

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНТЕРНЕТ-
МАГАЗИНА КНИГ

Студент

(подпись, дата)

(ФИО полностью)

Руководитель

(подпись, дата)

И.В. Акифьев
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

А. С. Бычков
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 2.06.2023 г. № 14)

Заведующий кафедрой

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой

(протокол заседания ГЭК от

16.06.2023 г. № 5)

Секретарь ГЭК

(подпись)

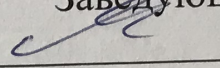
Е.И. Гурин
(фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
« 16 » 04 2023 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Долгов Дмитрий Петрович гр.19ВВВ1 факультета вычислительной техники.

2. Тема работы программное обеспечение интернет- магазина книг.

Тема утверждена приказом ПГУ № 458 от "26" апреля 2023г.

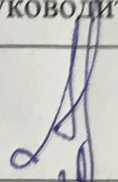
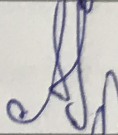
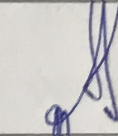
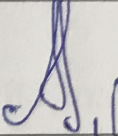
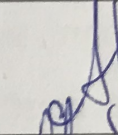
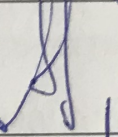
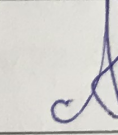
3. Руководитель бакалаврской работы к.э.н., доцент, каф. ВТ ПГУ Акифьев И.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

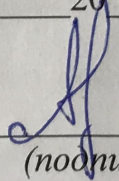
В ходе выполнения выпускной квалификационной работы необходимо разработать интернет-магазин, отвечающий всем требованиям функциональности, дизайна и структуры современного веб-ресурса, с целью предоставление пользователям удобной и доступной возможности приобретать книги созданные ИИ в онлайн режиме в электронном виде. Задачи, которые необходимо решить для достижения цели, поделить на три модуля: задачи исследования; задачи для разработки веб-сайта; задачи для разработки серверной части системы.

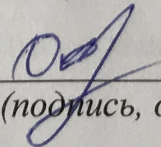
5. Перечень подлежащих разработке вопросов Требуется произвести анализ предметных областей: изучить существующие интернет магазины книг, выделить их достоинства и недостатки. Выбрать средства для разработки, которые подойдут для создания интернет магазина. Определить архитектуру магазина и вспомогательные средства разработки. Изучить функционал библиотек для взаимодействия с файлами.

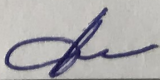
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	26.04.2023 – 28.05.2023	
2	Анализ требований	32	29.04.2023 – 30.05.2023	
3	Создание макета графического интерфейса	30	1.04.2023 – 3.05.2022	
4	Выбор технологий, применяемых при проектировании и разработке	40	4.05.2023 – 8.05.2023	
5	Реализация графического интерфейса	66	9.05.2023 – 15.05.2023	
6	Разработка основной части программы	70	16.05.2023 – 23.05.2023	
7	Оформление пояснительной записки	54	24.05.2023 – 13.05.2023	

Дата выдачи задания " 16. " апреля 20 23 г.

Руководитель бакалаврской работы  Акифьев И.В.
(подпись, дата)

Задание к исполнению принял студент  Долгов Д.П.
(подпись, дата)

Работу к защите допустить
Декан факультета  16.06.2023 Фионова Л.Р.
(подпись, дата)

Реферат

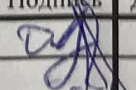
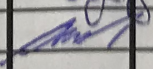
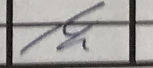
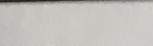
Выпускная квалификационная работа содержит 166 листов, 60 рисунков, 13 источников, 7 приложений.

КНИГИ, ИНТЕРНЕТ-МАГАЗИН, *JAVA*, *SPRING*, КОНТЕНТ, СКРИПТ, ХОСТИНГ, *JAVASCRIPT*, *HTML*, *API*, *FRONTEND*, *BACKEND*, *UML*

Цель работы – разработать ПО интернет-магазин, отвечающий всем требованиям функциональности, дизайна и структуры современного веб-ресурса, с целью предоставление пользователям удобной и доступной возможности приобретать книги созданные ИИ в онлайн режиме в электронном виде.

В результате проделанной работы было разработано ПО, которое позволяет пользователям легко и удобно редактировать свои профили, просматривать и выбирать книги из широкого ассортимента, производить поиск, просмотр и покупку книг в электронном виде, просматривать свою историю заказов, а также изменять персональную информацию. ПО обеспечивает полный набор функциональных возможностей для удобного и безопасного онлайн-шопинга книг в электронном виде.

Серверная часть была разработана на языке программирования *Java* с использованием библиотеки *Spring*, что обеспечило высокую стабильность и надежность работы. Клиентская программа разработано с помощью шаблонизатора *FMT*. Данные хранятся на сервере в БД *PostgreSQL*.

					ПГУ1.090301.08.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Долгов Д.П.			Программное обеспечение интернет-магазина книг Пояснительная записка		
Провер.		Акифьев И.В.					
Н. Контр		Бычков А.С.					
Утверд.		Митрохин М.А.					
					Лит.	Лист	Листов
						2	166
					ФВТ, каф. ВТ гр. 19BBB1		

Содержание

Перечень сокращений и обозначений	4
Введение	5
1 Программное обеспечение интернет-магазин книг	7
1.1 Анализ темы	7
1.1.1 Обоснование необходимости разработки	7
1.1.2 Анализ аналогов	9
1.1.3 Постановка задачи	15
1.1.4 Функциональные требования	17
1.2 Архитектура системы	20
1.3 Выбор инструментов разработки	24
1.4 Описание архитектуры клиентской части	25
1.4.1 Описание веб-страниц	26
1.4.2 URL-адреса веб-страниц	27
1.4.3 Навигация веб-страниц	28
1.4.4 Макет	29
1.4.5 Дизайн	31
1.4.6 Контент	36
1.5 Описание структуры базы данных	39
1.5.1 Таблицы в базе данных	39
1.5.2 Таблица «Книги» (Books)	39
1.5.3 Таблица «Содержание книги» (BookContent)	39
1.5.4 Таблица «Заказы» (Orders)	40
1.5.5 Таблица «Пользователи» (Users)	40

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.5.6	Хранение электронных книг.....	41
1.5.7	Передача заказа с электронными книгами.....	42
1.6	Описание архитектуры серверной части программы	43
1.6.1	Контроллеры	43
1.6.2	Сервисы.....	48
1.6.3	Репозитории.....	51
1.6.4	Модели	53
1.6.5	Конфигурация	58
1.6.6	Классы утилиты	60
1.7	Тестирование	61
1.7.1	Главная страница	62
1.7.2	Страница авторизации.....	63
1.7.3	Страница личного кабинета.....	64
1.7.4	Страницы продуктов	67
1.7.5	Страницы контактов.....	69
1.7.6	Страница корзины	69
1.7.7	Страница оформления заказа	70
1.7.8	Страница всех заказов.....	73
1.7.9	Страница просмотра книг	74
1.7.10	Страница с акциями.....	74
1.7.11	Страницы с новинками.....	75
1.8	Перспективы развития.....	76
2	Безопасность жизнедеятельности.....	77
2.1	Обеспечение электробезопасности	77

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

2.2	Ионизирующие излучения	79
2.3	Заключение по БЖД	82
	Заключение	83
	Список используемых источников	84
	Приложение А	85
	Приложение Б	156
	Приложение В	158
	Приложение Г	160
	Приложение Д	161
	Приложение Е	163
	Приложение Ж	165

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Перечень сокращений и обозначений

В настоящем отчете ВКР применяют следующие сокращения и обозначения

БД – база данных

ИИ – искусственный интеллект

ООП – объектно-ориентированное программирование

ПО – программное обеспечение

ССУ – современных систем управления

СУБД – система управления базами данных

API – программный интерфейс приложения

ISBN – международный стандартный книжный номер

FMT (*Free Marker Template*) – шаблонизатор веб-страниц

HTML (*Hyper Text Markup Language*) – язык гипертекстовой разметки

ORM (*Object-Relational Mapping*) – объектно-реляционное отображение

REST (*Representational State Transfer*) – передача репрезентативного состояния

MVC (*Model-View-Controller*) – «Модель-Вид-Контроллер»

UML (*Unified Modeling Language*) – унифицирован язык моделирования

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В Российской Федерации уровень информатизации и прогресса в области информационных технологий постоянно повышается. Причиной большого развития сферы информационных технологий стала конкурентная борьба в разработке веб приложений, сайтов, технических инструментов для сбора и анализа статистических данных.

Использование веб-технологий в сферах торговли и бизнеса позволяет значительно сократить расходы на содержание торговых площадей путем отказа от них. Значительное количество компаний осуществляют продажу товаров и услуг через интернет. Это упрощает поиск новых поставщиков и клиентов и дает конкурентное преимущество.

Значительный процент населения имеет возможность доступа к Интернет-ресурсам. В настоящее время человек получает большую часть новой информации из интернета: уведомления в социальных сетях, музыкальные новинки, культурные события, новости из творчества, а также новости о ситуации в мире. Поэтому наиболее популярным и удобным способом получения информации новинках литературы является поиск информации в сети Интернет. Обычно, пользователь заходит на сайт интересующего его магазина, предоставляющей услуги заказа, доставки литературы.

В основе современных систем управления (ССУ) веб-ресурсом лежат сложные программные комплексы, которые обеспечивают полную поддержку, работу и обслуживания интернет-ресурса. Большая часть ССУ, обладающих необходимой функциональностью, являются либо платными и имеют фиксированный набор компонентов, либо чрезмерно «сложными».

Вышесказанного подтверждает, что необходимость в интернет-ресурсах, которые создают конкуренцию уже существующим сервисам, предоставляющих онлайн торговлю, оснащенных системой управления содержимым оптимизированной непосредственно под торговли, еще

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

актуальна. При проектировании и разработке интернет-магазина, необходимо учесть удобство взаимодействия с интерфейсом как пользователя, так и персонала.

Сегодня многие области человеческой деятельности переживают период цифровой трансформации. Одной из таких областей является литературная индустрия, где современные технологии уже начинают проникать в традиционный процесс создания и продажи книг. С развитием искусственного интеллекта возникают новые возможности для создания литературных произведений, а также улучшения процессов редактирования и корректировки текста.

Одной из наиболее интересных и перспективных задач в области использования искусственного интеллекта в литературе является создание книг, написанных полностью ИИ. Это представляет собой небывалый подход к творчеству и может привести к появлению совершенно новых форм и жанров литературы.

В рамках данной работы будет рассмотрено применение искусственного интеллекта для создания книг, а также оценены возможности коммерциализации этого процесса. Будут проанализированы достоинства и недостатки такого подхода, а также потенциальные риски и вызовы, связанные с продажей книг, созданных ИИ.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1 Программное обеспечение интернет-магазин книг

1.1 Анализ темы

1.1.1 Обоснование необходимости разработки

Актуальность разработки книжного интернет-магазина «Book58» обусловлена непрерывным повышением роста продаж через интернет и прогрессивное развитие искусственного интеллекта. Разработка интернет-магазина книг, сгенерированных искусственным интеллектом (ИИ), имеет ряд преимуществ перед обычными магазинами:

- ИИ может создавать уникальные книги, которых еще нет на рынке. Такие книги могут привлечь новых покупателей, заинтересованных в новых авторах, жанрах или темах;

- написание и издание книг требует больших затрат времени и средств. Использование ИИ для создания книг может значительно сократить время и стоимость производства;

- использование ИИ для создания книг позволяет производить анализ данных о предпочтениях и поведении покупателей. Эта информация может быть использована для оптимизации производства, маркетинга и продаж, что может привести к увеличению прибыли;

- ИИ может создавать книги на различные темы и жанры. Это позволяет интернет-магазину предоставлять широкий ассортимент книг, достойный внимания каждого покупателя.

В связи с этим, разработка интернет-магазина книг, созданных ИИ, имеет большой потенциал для успешного бизнеса. Он может обеспечить конкурентное преимущество на рынке книжных магазинов и привлечь новых клиентов, заинтересованных в уникальном продукте.

Предметом данного исследования является разрабатываемое ПО.

Для определения бизнес-процессов, связанных с созданием и продажей книг, написанных ИИ, можно применить следующие методы и подходы:

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

– анализ литературного рынка и его потребностей - необходимо исследовать спрос на литературу, созданную ИИ, а также выявить особенности и требования к этому виду литературы со стороны покупателей;

– изучение процесса создания книг при использовании ИИ - необходимо разобраться в том, каким образом ИИ участвует в процессе создания книг, какие данные ему предоставляются, какие алгоритмы используются, и как это может отразиться на качестве результата;

– анализ процесса распространения и продажи книг - нужно изучить процесс продажи книг, созданных ИИ, и найти оптимальные способы продвижения и продажи таких книг на литературном рынке;

– оценка экономических показателей - необходимо изучить экономические показатели и прогнозируемую доходность проекта, связанного с созданием и продажей книг, написанных ИИ;

– определение ключевых шагов бизнес-процессов - необходимо выявить ключевые шаги в процессе создания и продажи книг, написанных ИИ, начиная от разработки концепции и заканчивая продвижением и продажей книг;

– изучение юридических аспектов - необходимо изучить правовые аспекты, связанные с авторскими правами на литературные произведения, созданные ИИ, а также учесть возможные ограничения и требования, накладываемые законодательством.

Комбинируя указанные методы и подходы, можно определить бизнес-процессы, связанные с созданием и продажей книг, написанных ИИ, и разработать стратегию для успешного продвижения таких книг на рынке. Поскольку способом поиска информации о книгах и авторах для потенциальных и предполагаемых покупателей являются поисковые системы и веб-сайты, то интерфейсом интернет-магазина должен являться веб-сайт.

Продажа книг, написанных искусственным интеллектом (ИИ), может быть актуальна по нескольким причинам.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Во-первых, ИИ способен генерировать уникальный контент, который может заинтересовать широкую аудиторию. Например, ИИ может создавать книги на основе анализа больших объемов данных, что позволяет ему выявлять новые тенденции в литературе и создавать произведения, соответствующие современным вкусам читателей.

Во-вторых, продажа книг, написанных ИИ, может стать конкурентным преимуществом интернет-магазина, который будет предлагать такой контент. Это поможет привлечь новых клиентов и увеличить объем продаж.

Кроме того, использование ИИ для создания книг может сократить затраты на оплату авторских гонораров, что может стать экономически выгодным для интернет-магазина. В то же время, это может вызвать этические и юридические вопросы, связанные с авторским правом.

Однако, хотя ИИ может быть полезным инструментом в литературе, он не заменит человеческое творчество и уникальность мышления. Таким образом, продажа книг, написанных ИИ, может стать интересным экспериментом для интернет-магазина, но не должна полностью вытеснять традиционную литературу и творческий потенциал писателей.

1.1.2 Анализ аналогов

При разработке интернет-магазина необходимо учесть возможности существующего на данный момент ПО и устранить недостатки, для повышения конкурентоспособности программного продукта.

Проанализировав наиболее важные работы отечественных и зарубежных исследователей, можно сделать вывод, что тема продажи книг через интернет магазин далеко не новая, а вот продажа книг, написанных ИИ, встречается только на торговой площадке amazon.com. Также стоит отметить наличие методических указаний по разработке интернет-магазинов, есть сравнительного анализа инструментов и способов реализации таких веб-приложений, что упрощает разработку.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.1.2.1 Интернет-магазин «Читай город»

Сайт «Читай город» – это всероссийский интернет-магазин книг, который предлагает широкий ассортимент книг различных жанров и направлений [1]. Основные характеристики сайта:

– дизайн и удобство использования. Сайт имеет современный и привлекательный дизайн, который не перегружен информацией. На главной странице представлены актуальные новинки, бестселлеры, а также разделы с книгами по жанрам и авторам. Навигация по сайту интуитивно понятна, что делает использование сайта удобным;

– ассортимент. Сайт предлагает огромный выбор книг различных жанров и направлений, включая художественную литературу, научно-популярные книги, бизнес-литературу, детские книги и многое другое. Кроме того, на сайте можно найти книги на разных языках;

– цены на книги на сайте вполне конкурентоспособны и часто ниже, чем в обычных книжных магазинах. Кроме того, на сайте часто проводятся акции и скидки, что делает покупку книг еще более выгодной;

– удобство заказа и доставки. Заказ книг на сайте происходит быстро и просто. Доставка осуществляется по всей России, а также в страны СНГ и за ее пределами. Кроме того, на сайте можно выбрать удобный способ оплаты;

– дополнительные возможности. На сайте есть раздел «Клуб читателей», где зарегистрированные пользователи могут получать бонусы за покупки и участвовать в различных акциях. Кроме того, на сайте можно найти информацию о книжных новинках, авторах и мероприятиях, связанных с книгами.

Недостатки сайта:

– непростой интерфейс. Сайт может показаться сложным для использования для некоторых пользователей, особенно для тех, кто не имеет опыта в онлайн-шопинге;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

- ограниченный выбор. Хотя сайт предлагает широкий ассортимент книг, некоторые пользователи могут не найти то, что ищут;
- дорогая доставка. Стоимость доставки может быть высокой, особенно для заказов из других регионов;
- низкая скорость загрузки. Сайт может загружаться медленно, что может оттолкнуть пользователей;
- отсутствие информации о товаре. Некоторые книги могут не иметь подробного описания, что может затруднить выбор покупателя;
- ограниченные способы оплаты, что может быть неудобно для некоторых пользователей.

В целом, сайт является удобным и надежным интернет-магазином книг, который предлагает широкий ассортимент книг, удобный заказ и доставку, а также дополнительные возможности для постоянных покупателей.

1.1.2.2 Интернет-магазин «Лексикон»

Сайт «Лексикон» – интернет-магазин книг, города Пензы [2]. Основные характеристики сайта:

- сайт имеет достаточно простую структуру, основанную на категориях и подкатегориях книг. Навигация по сайту достаточно удобна и интуитивно понятна;
- широкий выбор книг различных жанров и авторов. Однако, описания книг и обложки не всегда соответствуют действительности, что может вызывать разочарование у покупателей;
- дизайн сайта выглядит устаревшим и не очень привлекательным. Цветовая гамма и шрифты не всегда сочетаются между собой, что может создавать впечатление хаоса и неряшливости;
- некоторые проблемы с оптимизацией для поисковых систем. Например, отсутствуют мета-теги и заголовки страниц, что может затруднять индексацию сайта и его ранжирование в поисковых результатах. Это может

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

привести к проблемам с отображением сайта в различных браузерах и устройствах;

- отсутствуют ссылки на социальные сети, что может быть упущением в плане продвижения и привлечения новых клиентов;

- функционал корзины и оформления заказа работает, однако, дизайн и удобство использования могут быть улучшены.

1.1.2.3 Интернет-магазин «Моя книга»

Сайт «Моя книга» – интернет-магазин книг, города Саратов [3].
Характеристики сайт:

- имеет современный и привлекательный дизайн, который не перегружен информацией. На главной странице представлены актуальные новинки и бестселлеры, а также разделы с книгами по жанрам и авторам. Навигация по сайту в основном интуитивно понятна и удобна;

- широкий выбор книг различных жанров и направлений, включая художественную литературу, научно-популярные книги, бизнес-литературу, детские книги и многое другое. Кроме того, на сайте можно найти книги на разных языках;

- цены на книги на сайте вполне конкурентоспособны и часто ниже, чем в обычных книжных магазинах. Кроме того, на сайте часто проводятся акции и скидки;

- заказ книг на сайте происходит быстро и удобно. Для этого необходимо выбрать нужную книгу, добавить ее в корзину и оформить заказ. Доставка осуществляется по всей России;

- имеется раздел с отзывами покупателей, что позволяет ознакомиться с мнением других людей о книгах и сервисе сайта;

- есть раздел с подарочными сертификатами, а также возможность оформить подписку на журналы. Кроме того, на сайте можно найти информацию о книжных новинках и интересных мероприятиях, связанных с книгами.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Недостатки сайта:

- отзывы покупателей отсутствуют, что может затруднять принятие решения о покупке книги;
- магазин находится в Саратове, при доставке книг почтой или курьером, сроки доставки могут быть довольно длительными, что может быть неудобно для тех, кто хочет получить книгу быстро;
- доступны только несколько способов оплаты, что может быть неудобно для некоторых пользователей;
- не всегда предоставляется достаточно информации о книгах, такой как описание, автор, издательство и т.д. Это может затруднять выбор и покупку книги.

1.1.2.4 Интернет-магазин «Буквояд»

При анализе всероссийского интернет-магазина «Буквояд» можно отметить следующие моменты [4]:

- современный и привлекательный дизайн, который не перегружен информацией. На главной странице представлены актуальные новинки и бестселлеры, а также разделы с книгами по жанрам и авторам. Навигация по сайту интуитивно понятна и удобна;
- контент разделен на несколько категорий: книги, электронные книги, аудиокниги, подарки и сувениры, игры и развлечения, а также разделы для детей и подростков. Каждая категория содержит подкатегории, что упрощает поиск нужной информации;
- широкий выбор книг различных жанров и направлений, включая художественную литературу, научно-популярные книги, учебники и многое другое. Кроме того, на сайте можно найти аудиокниги, электронные книги и подарочные издания;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

– на главной странице сайта представлены новинки, бестселлеры, акции и скидки, что позволяет быстро ознакомиться с актуальными предложениями;

– цены на книги в целом соответствуют рыночным. Кроме того, на сайте часто проводятся акции и скидки;

– на сайте можно найти рекомендации по чтению, отзывы на книги, а также подписаться на рассылку новостей и акций. Кроме того, на сайте есть возможность оставить отзыв о книге и поставить ей оценку;

– удобный поиск по авторам, названиям книг и ISBN;

– есть разделы с литературными новостями, интервью с авторами, рецензиями на книги и другими материалами, связанными с миром книг;

– имеет мобильную версию.

Недостатки сайта:

– отсутствие отзывов и рейтингов. Это может снижать доверие пользователей к сайту и его товару;

– неоптимизированный дизайн для мобильных устройств. Если сайт не адаптирован для просмотра на мобильных устройствах, то пользователи могут испытывать трудности при просмотре сайта на своих смартфонах или планшетах.

1.1.2.5 Интернет-магазин «УчКоллектор»

Сайт «УчКоллектор» – интернет-магазин книг, города Пензы [5].

Основные характеристики сайта:

– дизайн сайта устарел и не очень удобен для пользователя. Некоторые элементы интерфейса не соответствуют современным требованиям, например, отсутствие адаптивности под мобильные устройства;

– представленная информация не является достаточно подробной и не дает полного представления о деятельности компании;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

– не оптимизирован для поисковых систем, отсутствуют мета-теги, заголовки страниц и другие элементы, которые помогают поисковым системам понимать содержание сайта;

– загружается довольно медленно, что может отталкивать пользователей и негативно сказываться на рейтинге сайта в поисковых системах;

– отсутствует SSL-сертификат, что может негативно сказаться на доверии пользователей к сайту и безопасности передачи данных.

В целом, сайт нуждается в обновлении и оптимизации, чтобы стать более удобным и привлекательным для пользователей.

Как видно из сравнения некоторые интернет-магазины используют устаревшие технологии, поэтому для конкурентно способности необходимо разработать ПО соответствующее современным стандартам.

1.1.3 Постановка задачи

В ходе выполнения выпускной квалификационной работы необходимо разработать интернет-магазин, отвечающий всем требованиям функциональности, дизайна и структуры современного веб-ресурса, с целью предоставление пользователям удобной и доступной возможности приобретать книги созданные ИИ в онлайн режиме в электронном виде. Задачи, которые необходимо решить для достижения цели, поделить на три модуля:

- задачи исследования;
- задачи для разработки веб-сайта;
- задачи для разработки серверной части системы.

Задачи исследования:

- изучение предметной области, определение потребностей целевой аудитории и требований к книжной продукции;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

– проведение сравнительного анализа существующих аналогов, осуществляющих продажу книжной продукции в сети Интернет, в целях выявления преимуществ и недостатков;

– определение функциональных и нефункциональных требования к проектируемому программному продукту, на базе которых разработать проект

книжного интернет-магазина;

– определение соответствующего окружение и инструментов для разработки книжного интернет-магазина;

– составление ключевых этапов программной реализации.

Задачи для разработки веб-сайта:

– разработка дизайна сайта, обеспечивающего максимально удобное и простое использование интернет-магазина;

– создание удобной навигации по сайту;

– разработка системы поиска, позволяющей находить книгу по названию и автору;

– создание функционала добавления книг в корзину и оформления заказа, а также интеграция с платежными системами;

– разработка функционала отзывов;

– разработка системы управления заказами и складом, чтобы обеспечить быструю и эффективную доставку книг клиентам;

– создание механизма акций и скидок.

Задачи, связанные с разработкой серверной части и базы данных:

– разработка серверной части интернет-магазина, обеспечивающая функционирование всех его элементов и интеграцию с другими системами.

– создание базы данных для хранения информации о книгах, клиентах, заказах и других объектах, связанных с функционированием интернет-магазина;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

- определение и реализация алгоритмов обработки запросов пользователя;
- тестирование серверной части на соответствие требованиям к работе сайта при большом количестве пользователей и высокой нагрузке;
- разработка системы защиты от атак и взломов, чтобы предотвратить возможность утечки конфиденциальных данных пользователей;
- создание системы бэкапов, чтобы обеспечить сохранность данных интернет-магазина и возможность быстрого восстановления в случае сбоя;
- обеспечение безопасности базы данных и защиты ее от взлома, а также установление доступов для сотрудников магазина, работающих с базой данных;
- реализация системы управления контентом интернет-магазина, позволяющей добавлять, редактировать или удалять информацию о книгах, авторах, издательствах и т.д.

1.1.4 Функциональные требования

Основные сценарии представлены на диаграмме вариантов использования (*use case*) (см. рисунок 1).

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

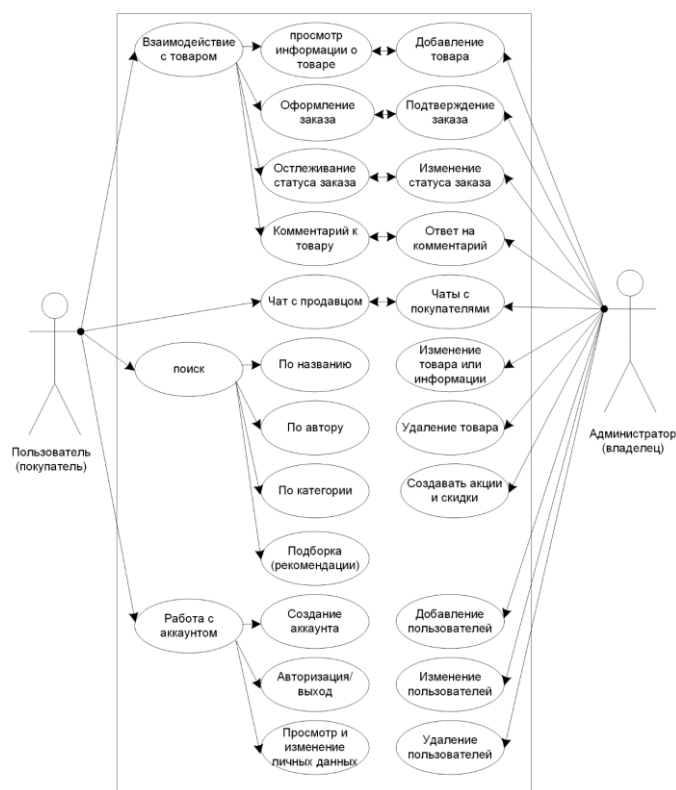


Рисунок 1 – Диаграмма вариантов использования

Ниже приведены варианты использования покупателя для интернет-магазина книг:

- покупка книг. Пользователь может искать и выбирать книги, добавлять их в корзину и удалять из нее, оформлять заказ и оплачивать покупку;
- просмотр информации о товаре. Пользователь может получить информацию о конкретной книге, такую как ее описание, автора, издательство, год издания, цену и отзывы других пользователей;
- поиск книг. Пользователь может искать книги по различным параметрам, таким как название, автор, категория и ключевые слова;
- создание аккаунта. Пользователь может создать аккаунт, чтобы получить доступ к дополнительным функциям, таким как сохранение адресов доставки, просмотр истории заказов и управление личным профилем;

- управление аккаунтом. Пользователь может изменять свою личную информацию, такую как адрес доставки, пароль и настройки уведомлений;
- отслеживание статуса заказа. Пользователь может отслеживать статус своих заказов, таких как подтверждение заказа, обработка, отправка и доставка;
- просмотр категорий. Пользователь может просматривать категории книг, такие как художественная литература, научно-популярная литература, бизнес-книги и многое другое;
- поиск по автору. Пользователь может искать книги, написанные определенным автором или выбрать из списка известных авторов;
- покупка электронных книг. Пользователь может покупать электронные книги в цифровом формате, которые могут быть загружены на устройство для чтения;
- подборки. Пользователь может просматривать подборки книг, составленные на основе различных критериев, таких как новинки, бестселлеры, рекомендации и т.д.

Администратор интернет-магазина книг имеет несколько важных задач, включая:

- просматривать и управлять заказами, обрабатывать платежи и обеспечивать быструю доставку книг;
- добавлять новые книги в каталог, редактировать существующие книги и удалять товары, которые уже не доступны для продажи;
- регулировать цены на товары, в зависимости от того, как меняется спрос и предложение на конкретные книги;
- создавать различные акции и скидки на книги, чтобы привлечь клиентов и повысить продажи;
- отвечать на вопросы клиентов, решать проблемы и общаться с ними через электронную почту или другие каналы связи.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Администратор интернет-магазина книг играет важную роль в успешном функционировании бизнеса, потому что обладает полномочиями для принятия стратегических решений, которые напрямую влияют на эффективность работы магазина и его успех.

1.2 Архитектура системы

Поставленная задача позволяет определить, что пользователей будет много, бизнес-цель – интернет-магазин книг.

Тип приложения – *Web*-приложение, которое работает на сервере и доступно для пользователей через браузер. Для разработки используются различные языки программирования и фреймворки. Обычно веб-приложение состоит из клиентской и серверной частей. Клиентская часть отвечает за отображение данных и взаимодействие с пользователем, а серверная часть обрабатывает запросы пользователя, выполняет бизнес-логику и взаимодействует с базой данных.

Потребителем интернет-магазина является любой пользователь с доступом в интернет, который заинтересован в покупке литературы. Пользователи могут быть как физическими лицами, так и юридическими лицами, которые хотят приобрести товары для своих личных нужд или для бизнеса соответственно.

Web-приложения имеют ряд минусов:

- зависимость от скорости интернет-соединения. Если соединение с интернетом медленное, то приложение может работать медленно и неудобно для пользователей;

- ограниченные возможности работы без интернета. *Web*-приложения не могут работать без доступа в интернет, что ограничивает их использование в некоторых случаях, например, когда нет подключения к сети;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

– безопасность. *Web*-приложения могут стать объектом хакерских атак и кражи данных пользователей, если система не обеспечена должной защитой;

– кросс-браузерность. Различные браузеры могут по-разному интерпретировать код, написанный на языках веб-технологий, что может привести к некорректному отображению или работе приложения в разных браузерах;

– невозможность доступа к ресурсам компьютера. *Web*-приложения не могут получить доступ к ресурсам компьютера пользователя, таким как файловая система, камера, микрофон и др., что может ограничить функциональность приложения в некоторых случаях.

Несмотря на эти минусы, *Web*-приложения имеют массу преимуществ и широко используются в современном мире.

В качестве технологий и библиотек для реализации интернет-магазина будут использоваться:

- ОС: *Linux*;
- серверный язык программирования: *Java*, библиотека *Spring*;
- фреймворки для веб-приложений: *Bootstrap*, *Free Marker Template*;
- СУБД: *PostgreSQL*;
- библиотеки для обработки платежей: *PayPal*;
- библиотеки для обеспечения безопасности: *bcrypt*.

Архитектурный стиль для такой системы будет трехуровневый. В трехуровневой архитектуре система разделена на три слоя:

- *Presentation Layer* – отвечает за отображение пользовательского интерфейса и обработку пользовательских запросов;
- *Business Layer* – содержит бизнес-логику приложения, которая выполняется после получения запроса от презентационного слоя;
- *Data Access Layer* – обеспечивает доступ к данным приложения.

Компоненты приложения:

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

- *Web-сервер* – принимает запросы от клиента, отображает страницы и передает данные между клиентом и сервером;
- СУБД (система управления базами данных) - хранит информацию о книгах, жанрах, авторах, заказах и пользователях;
- *Payment Gateway* - обрабатывает платежи при оплате заказа;
- *Email Service* - отправляет *email*-уведомления пользователю о статусе его заказа.

Архитектурные компоненты:

- *Frontend Framework* – используется для создания пользовательского интерфейса приложения;
- *API (Application Programming Interface)* – предоставляет доступ к функциям бизнес-логики приложения;
- *ORM (Object-Relational Mapping)* – используется для работы с базой данных приложения, чтобы упростить доступ к данным из кода приложения;
- *Cache* – используется для ускорения работы приложения путем сохранения результатов запросов к базе данных в кэше.

Общий поток взаимодействия между слоями приложения:

- клиент отправляет запрос на сервер, используя веб-интерфейс;
- *Web-сервер* принимает запрос и направляет его на бизнес-логику приложения через *API*;
- бизнес-логика обрабатывает запрос, выполняет необходимые проверки и манипуляции с данными (например, поиск книги в базе данных) и возвращает результат клиенту через *API*;
- если клиент оформляет заказ и проходит процесс оплаты, то данные о заказе отправляются в *Payment Gateway* и *Email Service* для дальнейшей обработки.

Такая архитектура позволяет создать расширяемое и гибкое приложение, которое может быть легко модифицировано и дополнено новыми функциями в дальнейшем.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Для оценки времени и объема работ был разработан прототип дизайна приложения, представленный на рисунке 2, в котором отражен необходимый функционал. Технические риски отсутствуют.

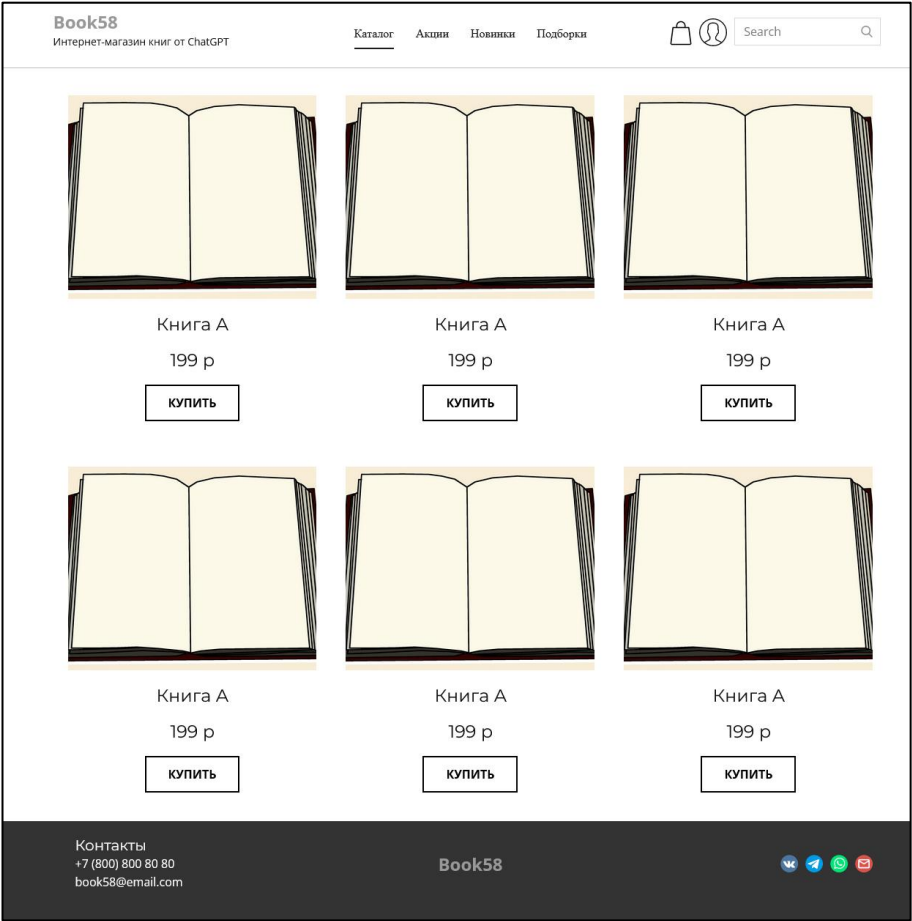


Рисунок 2 – Прототип дизайна веб-приложения, страница «каталог книг»

Прототип страницы сайта интернет-магазина является важным этапом разработки. Он помогает определить структуру и функциональность будущего сайта, а также понять потребности и предпочтения целевой аудитории.

На основании прототипа будет разрабатываться полноценный дизайн приложения и его верстка. Важно учесть не только функциональность, но и эстетические критерии: выбор цветовой гаммы, типографики, фотографий и других элементов дизайна. Также нужно учитывать требования к мобильной

версии сайта, которая должна быть адаптирована под различные размеры экранов смартфонов и планшетов

После создания сайта необходимо провести тестирование на пользователях, чтобы убедиться в его удобстве использования и эффективности. Это позволит сделать дополнительные корректировки и улучшения, если они будут необходимы. Профессионально спроектированный и продуманный дизайн сайта поможет привлечь больше клиентов и повысить уровень продаж в интернет-магазине.

1.3 Выбор инструментов разработки

Для реализации поставленных задач были рассмотрены следующие технологии и среда разработки:

Spring Boot – это современный и мощный фреймворк для разработки приложений на языке *Java*, который позволяет быстро создавать высокопроизводительные приложения [6].

Free Marker Template (FMT) – это шаблонизатор для *HTML*, который позволяет создавать динамические страницы веб-приложения, а также улучшает читаемость кода и облегчает его поддержку [7].

PostgreSQL – это мощная реляционная база данных с открытым исходным кодом, которая обладает высокой производительностью, надежностью и безопасностью [8].

IntelliJ IDEA – это одна из лучших интегрированных сред разработки (*IDE*) для *Java*, которая предоставляет широкий набор инструментов для создания, отладки и тестирования приложений.

Недостатки выбранных технологий:

– *Spring Boot* может быть достаточно сложным для начинающих разработчиков, так как требует хорошего знания основ *Java* и основных концепций фреймворка;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

– *FreeMarker Template* является мощным и гибким инструментом для создания шаблонов, однако у него есть несколько недостатков. Не обладает таким богатым функционалом, как другие шаблонизаторы, что может привести к трудностям при реализации некоторых функций. Не предоставляет возможности для прямого вывода данных на экран, что означает, что разработчику нужно дополнительно обрабатывать данные перед выводом. Может работать медленно при обработке больших объемов данных, особенно если используются сложные шаблоны и выражения;

– *PostgreSQL* может быть достаточно трудным для настройки и управления для некоторых пользователей, особенно если они не имеют опыта работы с реляционными базами данных;

– *IntelliJ IDEA* является коммерческой *IDE* и требует оплаты лицензии для полного использования всех ее функций.

В целом, выбранный стек технологий предоставляет хорошие возможности для создания интернет-магазина на языке *Java*, но требует от разработчика определенного уровня знаний и опыта в работе с этими технологиями.

1.4 Описание архитектуры клиентской части

Для реализации функционала, который был описан в постановке задач, могут понадобиться следующие веб-страницы:

- главная страница;
- страница авторизации;
- страница личного кабинета;
- страницы продуктов;
- страницы контактов;
- страница корзины;
- страница оформления заказа;
- страница оплаты;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

- страница подтверждения оплаты заказа
- страницы с акциями;
- страницы с новинками;
- страницы заказов пользователя;
- страница приобретенных книг пользователя;
- страница администрирования;
- страница добавления книги;
- страница заказов пользователей.

1.4.1 Описание веб-страниц

Главная страница: на этой странице можно разместить краткую информацию о продукте. Здесь же расположены основные элементы навигации, такие как меню и поиск.

Страница регистрации и авторизации: позволяет пользователям зарегистрироваться на сайте или войти в свой аккаунт, если они уже зарегистрировались. Авторизации необходима пользователям сайта, чтобы получить доступ к добавлению товаров в корзину или оформление заказа.

Страница профиля или личного кабинета с настройками профиля, истории заказов и сообщений.

Страницы продуктов, на которых подробнее описать книгу. Здесь размещены фотографии, цены, информация о содержании книги, жанре, авторе, стоимости и отзывах.

Страницы контактов: эта страница содержит информацию о том, как связаться с владельцами сайта или службой поддержки. Обычно здесь размещаются адреса, телефоны и формы обратной связи.

Страница корзины отображает выбранные пользователем товары с возможностью изменения количества книг, удаления из корзины и оформление заказа.

Страница оформления заказа содержит форму для заполнения пользователем необходимой информации.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Страницы с акциями обычно содержит информацию о текущих скидках, распродажах, бонусах и других специальных предложениях. На этой странице необходимо всегда предоставлять актуальную информацию.

Страницы с новинками обычно содержит информацию о последних изданиях книг.

На странице заказов пользователи могут посмотреть свою историю заказов, отслеживать статус выполнения заказа.

На странице приобретенных книг пользователи могут просмотреть список книг, которые они купили и скачали с сайта.

Страница оплаты: на этой странице пользователи могут выбрать способ оплаты и ввести необходимую информацию для оплаты.

Страница подтверждения оплаты заказа: после успешной оплаты пользователь перенаправляется на эту страницу, где он может увидеть подтверждение оплаты и основную информацию о своем заказе.

Страница администрирования: на этой странице только администраторы сайта могут управлять содержимым сайта, добавлять или удалить книги, изменять цены и т.д.

Страница добавления книги: только администраторы сайта имеют доступ к этой странице, где можно добавить новую книгу на сайт. На этой странице необходимо заполнить всю необходимую информацию о книге, такую как название, автор, описание, фотографии и т.д.

Страница заказов пользователей: на этой странице администраторы могут просмотреть список всех заказов от пользователей, изменять статусы заказов.

1.4.2 URL-адреса веб-страниц

Конечный URL-адрес для каждой страницы может зависеть от конкретного доменного имени. Так как разрабатывается локальное приложение на порту 8090, то URL будут выглядеть следующим образом:

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

- Главная страница: <http://localhost:8090/>;
- Страница авторизации: <http://localhost:8090/login>;
- Страница личного кабинета: <http://localhost:8090/account>;
- Страницы продуктов: <http://localhost:8090/products>;
- Страницы контактов: <http://localhost:8090/contacts>;
- Страница корзины: <http://localhost:8090/basket>;
- Страница оформления заказа: <http://localhost:8090/checkout>;
- Страницы с акциями: <http://localhost:8090/main?typePage=discount> ;
- Страницы с новинками: <http://localhost:8090/main?typePage=novelty> ;
- Страницы заказов пользователя: <http://localhost:8090/user/orders>;
- Страница приобретенных книг: <http://localhost:8090/user/books>;
- Страница оплаты: <http://localhost:8090/basket/payment>;
- Страница подтверждения оплаты заказа: <http://localhost:8090/basket/payment>;
- Страница администрирования: <http://localhost:8090/user>;
- Страница добавления книги: <http://localhost:8090/api>;
- Страница заказов пользователей: <http://localhost:8090/orders>.

1.4.3 Навигация веб-страниц

Для данного списка страниц, навигационная система включает в себя главное меню, отображаемое на каждой странице сайта, и которое может содержать ссылки на все основные страницы: “Каталог”, “Акции”, “Новинки”, “Корзина”. На рисунке 3 представлена схема карта веб-сайта интернет магазина.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

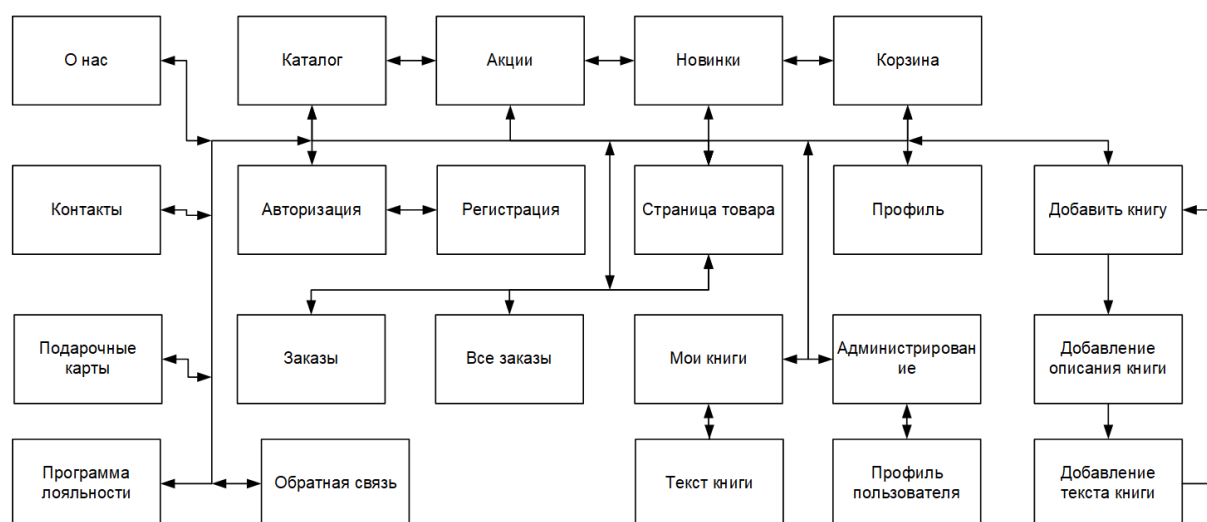


Рисунок 3 – Схема сайта интернет магазина

Также присутствует дополнительный блок навигации, который может размещаться в верхней части сайта. Этот блок содержит ссылки на страницы «Профиль», «Заказы» и «Мои книги».

Для удобства пользователей, на страницах продуктов, акций, новинок используются фильтры. В целом, навигация интуитивно понятная и доступная, так как количество страниц не большое.

1.4.4 Макет веб-страниц

Следующий шаг – создание макетов каждой страницы на основе описания навигации. Ниже представлены и описаны созданные макеты для четырех веб-страниц: каталога, страницы товара, корзины и страница текста книги.

На странице каталога предварительно расположена шапка с навигацией, эмблемой и кнопкой расширенного меню. Основной контент – список доступных товаров с описанием и изображениями. Перед товарами расположен поиск и фильтрация товара (см. рисунок 4).

На странице товара находится более подробная информация о выбранном товаре: название, цена, автор, описание, а также кнопка "Добавить в корзину" (см. рисунок 5).

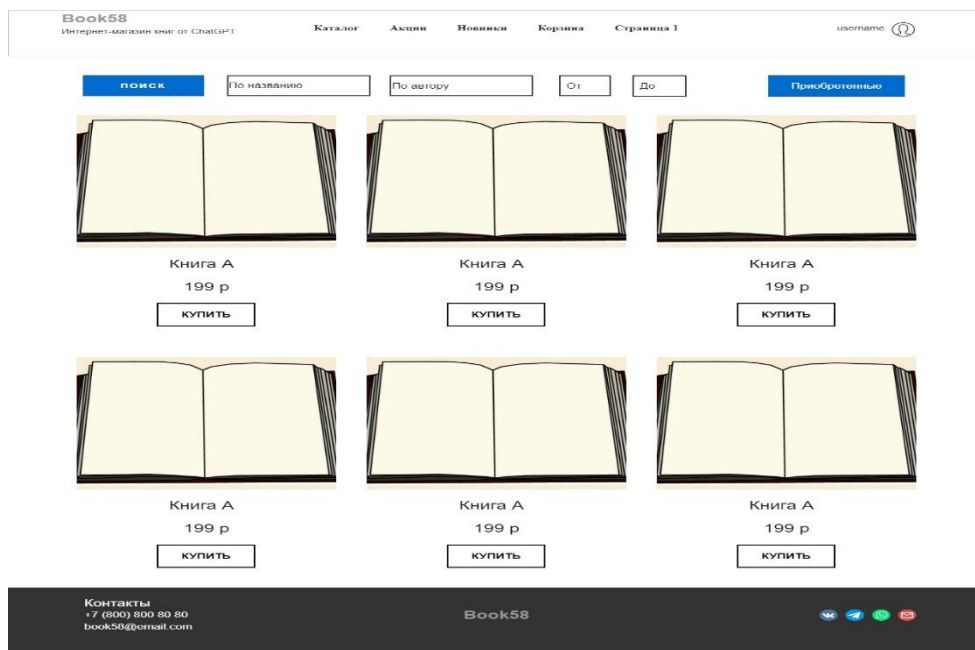


Рисунок 4 – Макет страницы каталога

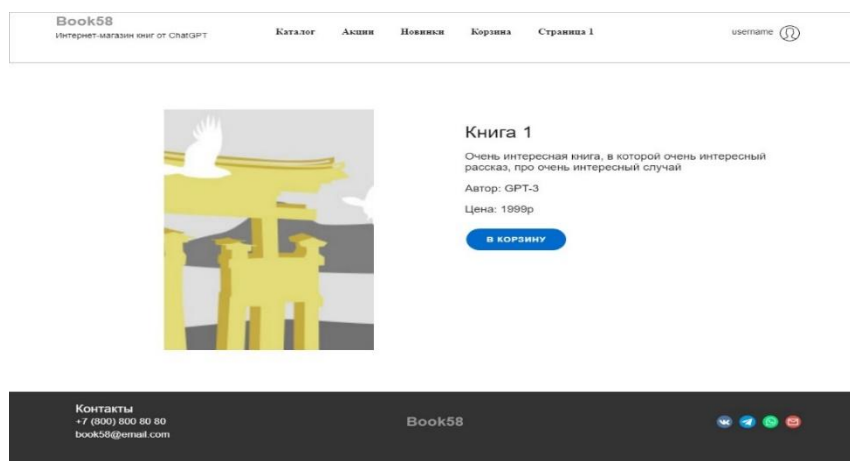


Рисунок 5 – Макет страницы товара

Корзина – это страница, на которой отображен список выбранных товаров. А также кнопку удаления товаров из корзины перед оформлением заказа. На этой странице есть цена каждого товара со скидкой или без, общая стоимость заказа и кнопка перехода на страницу оформления заказа и на страницу каталога (см. рисунок 6).

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

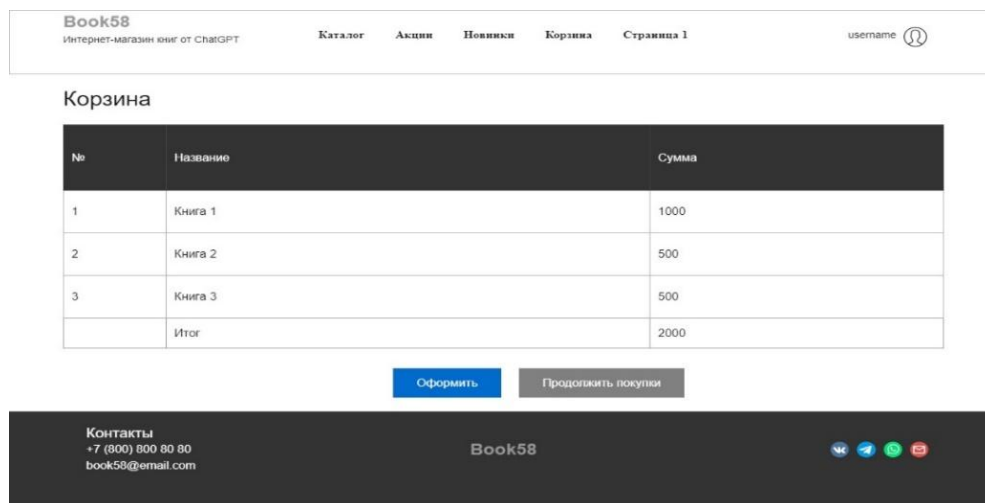


Рисунок 6 – Макет корзины

Страница текста книги – на ней представлено содержимое книги. Возможно, здесь также можно предоставить пользователю возможность выбора формата текста и шрифт (см. рисунок 7).

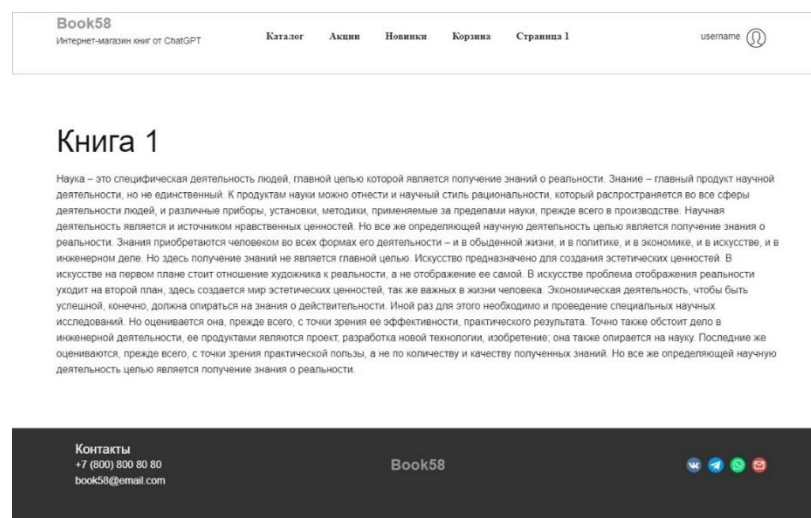


Рисунок 7 – Макет страницы просмотра текста

1.4.5 Дизайн веб-страниц

После этого этап к разработке дизайна сайта и функциональности, Следующим шагом было создание страниц каждой страницы на основе макетов и описания навигации. На рисунках 8–19 представлены основные веб-страницы сайта интернет магазина книг. После этого этап к разработке

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

дизайна сайта и функциональности, используя выбранный инструмент для создания сайтов.

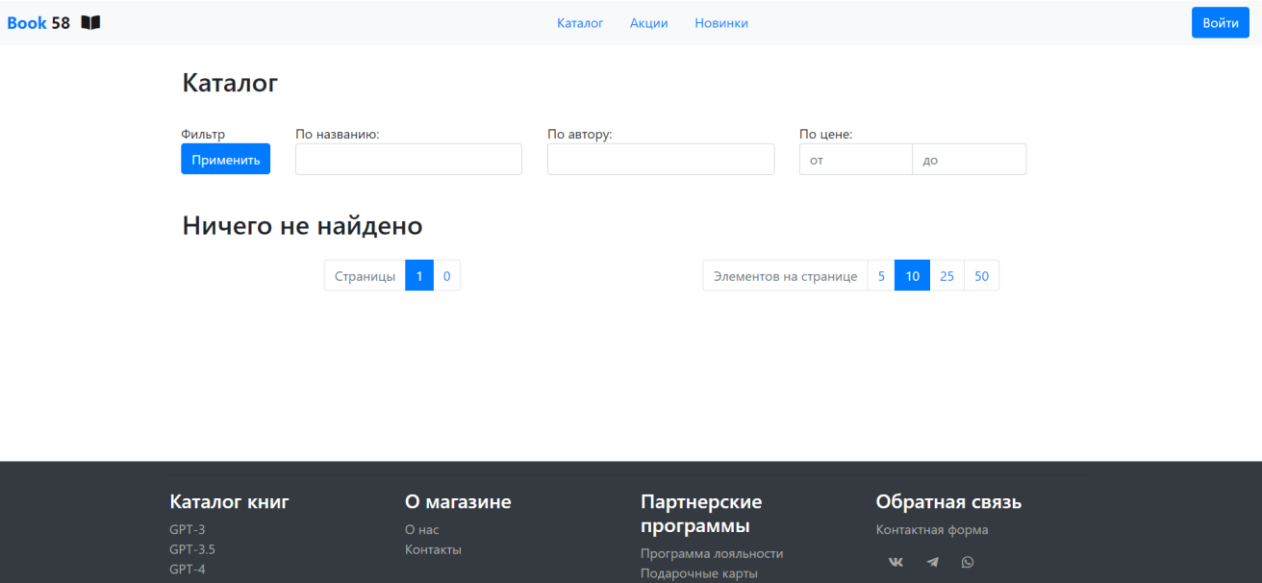


Рисунок 8 – Главная страница неавторизованного пользователя

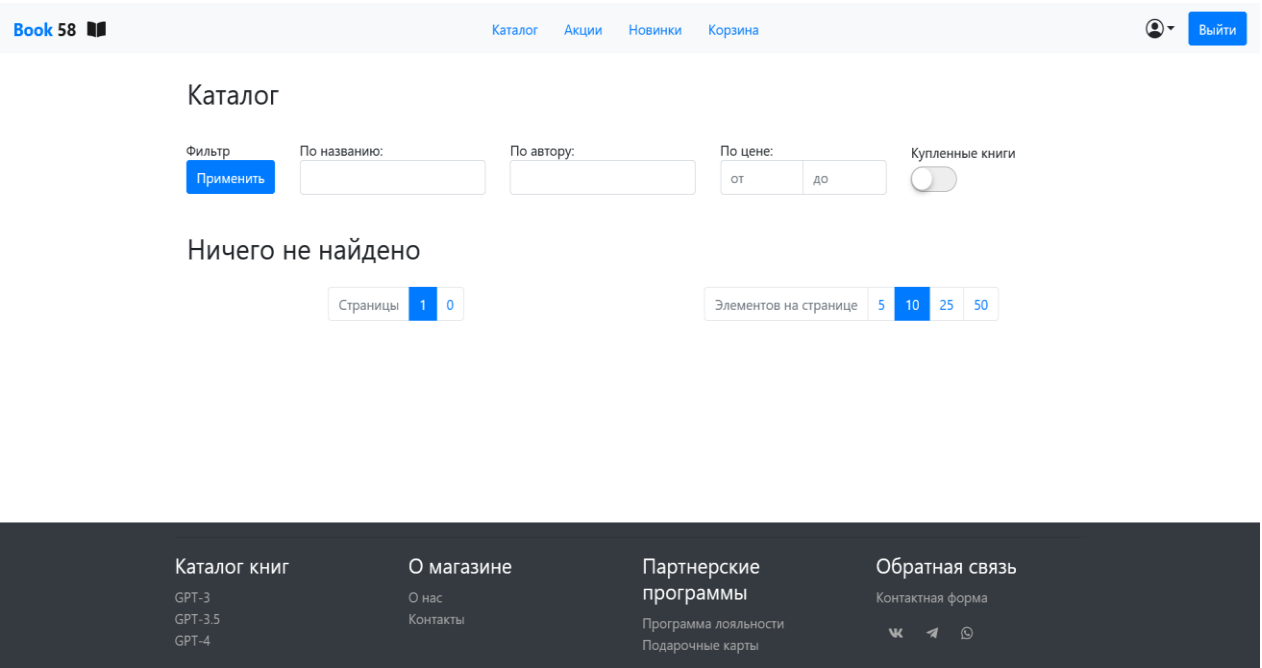


Рисунок 9 – Главная страница авторизованного пользователя

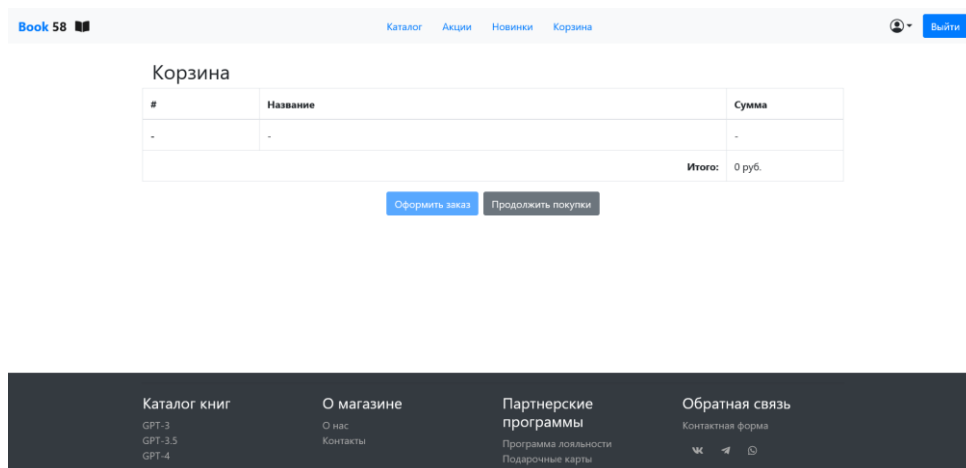


Рисунок 10 – Пустая корзина товаров

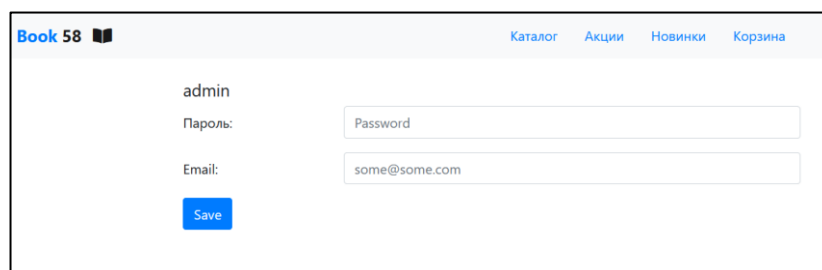


Рисунок 11 – Страница профиля пользователя

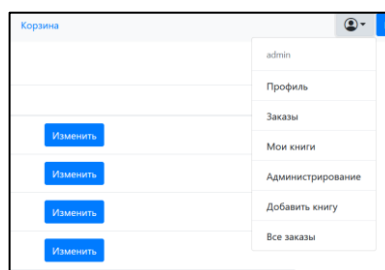


Рисунок 12 – Расширенное меню администратора

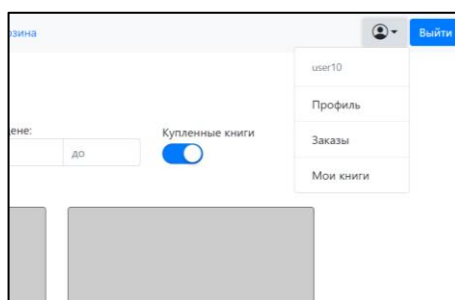


Рисунок 13 – Расширенное меню обычного пользователя

Book 58		Каталог	Акции	Новинки	Корзина	Выйти
Список пользователей						
Имя	Права					
admin	ADMIN, USER	Изменить				
user1	USER	Изменить				
user2	USER	Изменить				
user3	USER	Изменить				
user4	USER	Изменить				
user5	USER	Изменить				
user6	USER	Изменить				
user7	USER	Изменить				
user8	USER	Изменить				

Рисунок 14 – Список пользователей

Book 58		Каталог	Акции	Новинки	Корзина	Выйти
Список заказов						
Фильтр	По пользователю:	По книге:	По дате:	По статусу:		
Применить						
Пользователь	Книга	Дата заказа	Статус	Удалить	Написать	
user1	Книга 5	26.05.2023 14:25:30	Форм Выполни Закрыт			
user1	Книга 4	26.05.2023 14:25:30	Форм Выполни Закрыт			
user1	Книга 3	26.05.2023 14:25:30	Форм Выполни Закрыт			
admin	Книга 49	11.06.2023 21:28:20	Форм Выполни Закрыт			
<div> <div>Каталог книг GPT-3 GPT-3.5 GPT-4</div> <div>О магазине О нас Контакты</div> <div>Партнерские программы Программа лояльности Подарочные карты</div> <div>Обратная связь Контактная форма</div> </div>						

Рисунок 15 – Список заказов всех пользователей

Book 58

Каталог

Акции

Новинки

Корзина

Выйти

Корзина

#	Название	Сумма
1	Книга 50	<div>699 руб.</div>
2	Книга 48	<div>999 1.099 руб.</div>
		<div>Итого: 1 698 руб.</div>

Оформить заказ

Продолжить покупки

Каталог книг

GPT-3

GPT-3.5

GPT-4

О магазине

О нас

Контакты

Партнерские программы

Программа лояльности

Подарочные карты

Обратная связь

Контактная форма

ЧК

←

→

Рисунок 16 – Корзина с выбранными товарами

					ПГУ1.090301.08.001 ПЗ		Лист
Изм.	Лист	№ докум.	Подпись	Дата			

Book 58

КаталогАкцииНовинкиКорзина

Выйти

Оплата

Номер карты:

1234123412341234

Имя владельца карты:

111

Срок действия:

11

11

111

Общая сумма заказа:

1 698 руб.

ОплатитьНазад

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

ЧК

Рисунок 17 – Страница оплаты заказа

Book 58

КаталогАкцииНовинкиКорзина

Выйти

Ваш заказ успешно оформлен!

Спасибо за покупку! Мы свяжемся с вами в ближайшее время для уточнения деталей.

Перейти в каталогПерейти в раздел заказов

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

ЧК

Рисунок 18 – Страница подтверждения заказа

Book 58

КаталогАкцииНовинкиКорзина

Выйти

Заказы

Дата	Название	Сумма	Статус
11.06.2023 22:07:07	Книга 50	699 руб.	FORMED
11.06.2023 22:07:08	Книга 48	1 099 руб.	FORMED
11.06.2023 21:28:20	Книга 49	899 руб.	ISSUED

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

ЧК

Рисунок 19 – История заказов пользователя

Frontend на *Free Marker Template* для интернет-магазина книг был построен следующим образом:

- создан шаблоны *HTML*-страниц с помощью *Free Marker Template*, добавлены необходимые теги и переменные;
- созданы контроллеры, которые отвечают за обработку запросов пользователя и передачу данных в шаблоны;
- созданы сервисы, которые получают данные из базы данных и передают их контроллерам;
- использование фреймворк *Bootstrap* для быстрой разработки и *CSS*: для дополнительной кастомизации внешнего вида приложения;
- интеграция *JavaScript*: использован *JavaScript* для добавления интерактивных элементов, кнопки, выпадающие меню, слайдеры и т.д.;
- настройка маршрутизации, чтобы пользователи могли переходить между страницами и выполнять различные действия;
- проведено простое тестирование приложения в браузерах *Mozilla* и *Google Chrome*.

В целом, архитектура *Frontend* на *Free Marker Template* для интернет-магазина книг получилась достаточно гибкой и масштабируемой, что легко позволит вносить изменения и добавлять новые функциональные возможности.

1.4.6 Контент интернет-магазина

Кроме того, необходимо заняться контентом для сайта, написать тексты, подготовить изображения и другие материалы, которые будут размещены на страницах. Для примера заполнения была добавлена книга на рисунках 20–24.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Book 58

КаталогАкцииНовинкиКорзина

Выход

Добавить книгу

Название

Описание

Автор

Цена

Цена со скидкой

Выберите файл

Browse

Далее

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

Рисунок 20 – Страница добавления описания книги

Book 58

КаталогАкцииНовинкиКорзина

Выход

Генератор

Введите текст запроса:

напишите текст из 10 слов про луну

Выберите версию GPT:

GPT-3.5

Создать

Ответ ИИ:

Дополнить

Результат:

Сохранить

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

Рисунок 21 – Страница добавление текста книги, генерация

Book 58

КаталогАкцииНовинкиКорзина

Выход

Генератор

Введите текст запроса:

напишите предложение про полет на луну

Выберите версию GPT:

GPT-3.5

Создать

Ответ ИИ:

Дополнить

Результат:

Луна - спутник Земли, светит в ночном небе ярко.

Сохранить

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

Рисунок 22 – Страница добавление текста книги, добавление

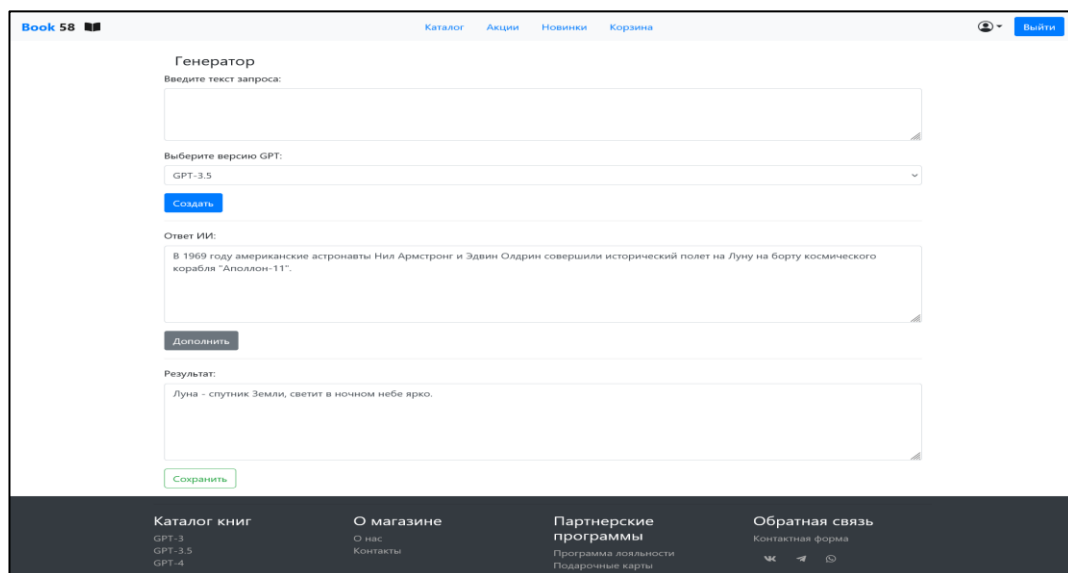


Рисунок 23 – Страница добавление текста, генерация нового текста

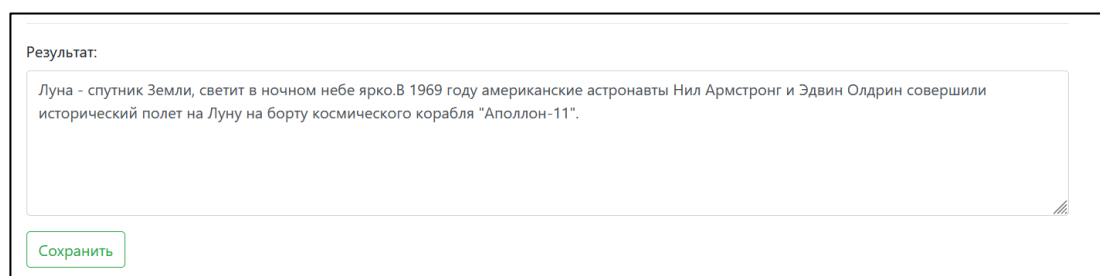


Рисунок 24 – Результат генерации текста

Контент-маркетинг является важной составляющей маркетинговых стратегий, поскольку позволяет предоставлять полезную информацию и привлекать новых пользователей.

Процесс создания качественного контента может быть длительным и трудоемким. Требуется провести много времени на исследование темы, написание текстов и подготовку изображений и других материалов.

Использование ИИ для генерации текстов на данный момент может быть ограниченным. Хотя существуют инструменты для генерации контента на основе алгоритмов ИИ, таких как механические турбины, результаты могут быть недостаточно точными или не соответствовать заданным требованиям.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.5 Описание структуры базы данных

1.5.1 Таблицы в базе данных

Таблицы и поля в базах данных используются для хранения и организации большого объема информации. Они позволяют структурировать данные таким образом, чтобы можно было легко находить нужную информацию и выполнять различные операции с ней [9]. Использование таблиц и полей позволяет обеспечить целостность и безопасность данных. Например, определение правил для заполнения полей или установить связи между различными таблицами, поможет избежать ошибок при работе с данными [10].

1.5.2 Таблица «Книги» (*Books*)

Предназначена для хранения информации о книге и содержит поля:

- *ID* книги (*BookID*);
- название книги (*Title*);
- описание (*Description*);
- автор (*Author*);
- цена (*Price*);
- обложка (*Image*);
- дата создания (*date_create*).

1.5.3 Таблица «Содержание книги» (*BookContent*)

Предназначена для хранения текстового содержания книги и имеет поле «*ID* книги», которое связывает ее с соответствующей записью в таблице «Книги». Содержит поля:

- *ID* книги (*BookID*);
- текст (*Content*).

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Таблицы «Книги» и «Содержание книги» имеют связь 1 к 1, что означает, что каждая запись в таблице «Книги» соответствует только одной записи в таблице «Содержание книги», а каждая запись в таблице «Содержание книги» соответствует только одной записи в таблице «Книги».

1.5.4 Таблица «Заказы» (*Orders*)

Предназначена для хранения информации о заказах и содержит поля:

- *ID* заказа (*OrderID*);
- дата заказа (*OrderDate*);
- статус заказа (*OrderStatus*);
- *ID* пользователя (*UserID*);
- *ID* книги (*BookID*).

1.5.5 Таблица «Пользователи» (*Users*)

Предназначена для хранения информации зарегистрированных пользователей и содержит поля:

- *ID* пользователя (*UserID*);
- роль пользователя (*Role*);
- имя (*FirstName*);
- фамилия (*LastName*);
- электронная почта (*Email*);
- пароль (*Password*).

Каждый пользователь может создавать несколько заказов, которые хранятся в таблице «Заказы». Для каждого заказа указывается *ID* пользователя (*UserID*), который сделал заказ.

Чтобы определить, какие конкретно книги были заказаны в рамках каждого заказа, используется связь между таблицами «Заказы» и «Книги». Для этого в таблице «Заказы» присутствует поле *ID* книги, которое ссылается на соответствующую запись в таблице «Книги». Таким образом, можно

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

узнать, какие книги были заказаны в каждом конкретном заказе. Для примера структура данных в БД представлена на рисунке 25.

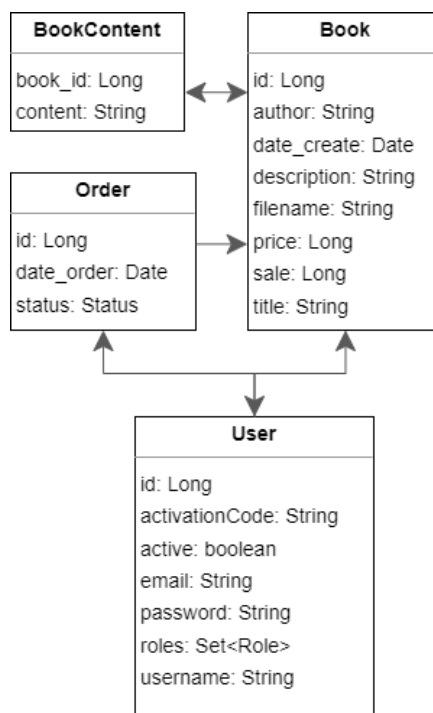


Рисунок 25 – Структура данных в БД, связь таблицы, поля таблиц

Дополнительно, таблица «Пользователи» связана с таблицей «Заказы» через поле *ID* пользователя (*UserID*), которое указывает на конкретного пользователя, который сделал заказ.

1.5.6 Хранение электронных книг

Хранение электронных книг может происходить на:

- локальном хранилище на сервере. Хранить электронные книги непосредственно на сервере приложения (см. рисунок 26). Однако это может привести к ограничению доступности и отказу в обслуживании, если количество пользователей и книг будет расти. Для раскрутки веб-сервиса будет достаточно, в последствии переход на облачное хранилище;
- облачном хранилище – например *Google Cloud Storage*, для хранения книг. Это позволит быстро расширять объем хранилища и

уменьшить затраты на поддержку инфраструктуры. Это вариант будет рассматривается при раскрученном интернет магазине.

	book_id	content
1	101	Сегодня утром, когда мой сервер был включен, я начал ра...
2	102	Каждый день я начинаю свою работу как только мой сервер...
3	103	Обычно люди стараются расслабиться и провести время с с...
4	104	В одном эксперименте исследователи использовали нейронн...
5	105	В одном эксперименте исследователи использовали нейронн...
6	106	В одном эксперименте исследователи использовали нейронн...
7	107	В одном эксперименте исследователи использовали нейронн...
8	108	В одном эксперименте исследователи использовали нейронн...
9	109	В одном эксперименте исследователи использовали нейронн...

Рисунок 26 – Текст книг в базе данных

1.5.7 Передача заказа с электронными книгами

Отдача электронных книг покупателю может быть осуществлена несколькими способами:

- предоставление ссылки для скачивания книги, который будет отправлена покупателю после оплаты. При скачивании файла книги необходимо использование соединения *HTTPS* для шифрования информации о покупателе;

- просмотр в браузере, с помощью использовать специальные-библиотеки для отображения контента в браузере, чтобы покупатель мог прочитать книгу без необходимости скачивать файл (см. рисунок 27). *PDF.js* – бесплатная библиотека *JavaScript*, разработанная *Mozilla* для работы с *PDF*-документами в браузере и может использоваться для отображения электронных книг.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

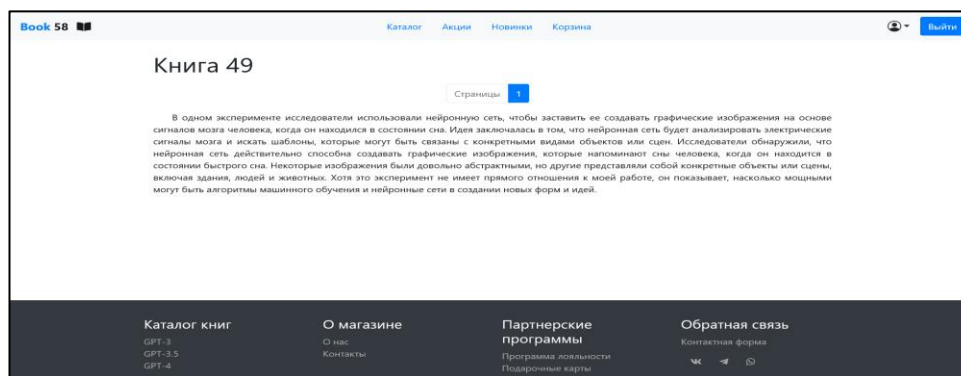


Рисунок 27 – Страница текста приобретенной книги

1.6 Описание архитектуры серверной части программы

Backend – это термин из области разработки программного обеспечения, который описывает часть системы или приложения, ответственную за обработку данных и бизнес-логики [11,12]. Ниже описана подробная архитектура *Backend* на *Spring Boot* для интернет-магазина книг.

Для реализации каталога, личного кабинета, навигации, корзины и отзывов в интернет-магазине книг понадобятся нижеперечисленные контроллеры, сервисы и репозитории.

1.6.1 Контроллеры

Контроллеры (*controllers*) – классы, которые обрабатывают запросы от клиентов и возвращают им ответы. Для интернет-магазина книг созданы следующие контроллеры.

ApiController – обрабатывает запросы для получения информации о книгах, добавления новых книг, обновления, добавления и генерации текста книг. Методы *api()* и *add()* отвечают за добавление новых книг в базу данных магазина. В методе *api()* пользователь может заполнить данные книги с помощью формы на веб-странице и загрузить изображение обложки, а затем отправить эти данные на сервер для сохранения. Метод *add()* обрабатывает эту форму и вызывает соответствующие сервисы для сохранения информации о книге. Метод *generate()* отвечает за генерацию описания книги

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

при помощи *OpenAI API*. Он принимает текстовый ввод пользователя и генерирует новый текст, который может быть добавлен к текущему описанию книги или сохранен в отдельном файле. Контроллер также имеет несколько аннотаций: *@Controller* указывает, что это класс, отвечающий за контроль веб-страниц, *@Autowired* используется для инъекции зависимостей, *@RequestMapping* указывает путь, по которому этот контроллер будет доступен, а *@PreAuthorize* указывает, что у пользователя должны быть права администратора для доступа к методам в этом контроллере.

ControllerUtils – контроллер, для работы с ошибками валидации данных в Spring-приложении. Он содержит метод *getErrors*, который принимает объект типа *BindingResult*, представляющий результат валидации данных на стороне сервера, и возвращает *Map<String, String>*, содержащую пары ключ-значение, где ключ – это имя поля, содержащего ошибку валидации, а значение – сообщение об ошибке. Для создания объекта *Map* используется метод *stream* из *Java 8*, с помощью которого выполняется маппинг списка ошибок в новый объект *Map*. Для этого используется объект типа *Collector*, созданный методом *toMap* из класса *Collectors*, который определяет правила маппинга. В данном случае, для каждой ошибки формируется ключ — это имя поля и строка «*Error*», а значение – сообщение об ошибке.

OrderController – контроллер для обработки запросов, связанных с заказами и книгами в магазине. Метод *getOrders()* отображает список заказов пользователя, которые были выданы или закрыты. Для этого метода используется сервис *bookService*, который получает соответствующие списки заказов из базы данных и преобразует их в объекты *BookDto*. Модель передается на веб-страницу, где она отображается в виде списка заказов. Метод *getBooks()* отображает список книг, которые были заказаны пользователем. Для этого также используется сервис *bookService*, который получает список книг из базы данных и преобразует его в объекты *BookDto*.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Модель передается на веб-страницу, где она отображается в виде списка книг. Методы *userEditForm()* и *saveStatus()* отвечают за отображение и изменение статуса заказа администратором магазина. Метод *userEditForm()* отображает список всех заказов на веб-странице и позволяет администратору изменять статус каждого заказа. Метод *saveStatus()* сохраняет выбранный новый статус заказа в базе данных магазина. Для доступа к некоторым методам контроллера требуются права администратора. В таком случае используется аннотация *@PreAuthorize("hasAuthority('ADMIN')")*, которая позволяет автоматически проверять права доступа к методу перед его выполнением.

ProductController – класс содержит методы для получения информации о продукте, оформления заказа, просмотра списка товаров, добавления товаров в корзину, удаления товаров из корзины и других действий, связанных с продуктами в магазине. Некоторые методы имеют аннотации *@GetMapping* и *@PostMapping*, которые указывают HTTP-методы, поддерживаемые каждым методом. Используется аннотация *@PreAuthorize("hasAuthority('ADMIN')")* для контроля доступа к методам только для пользователей с правами администратора. Метод *greeting(Map<String, Object> model)* просто перенаправляет пользователя на страницу */main*. Он используется для удобства пользователей, чтобы сразу попадать на главную страницу магазина после захода на сайт. Метод *main(...)* получает список всех книг и отображает его на странице */main*. Он также учитывает возможные фильтры и параметры постраничного вывода. Аннотация *@PageableDefault* указывает на параметры постраничного вывода по умолчанию, а параметр *pageable* — на текущие параметры. Метод *getBasket(...)* получает список книг, добавленных в корзину текущим пользователем, и отображает его на странице */basket*. Он также вычисляет общую стоимость товаров и передает ее на страницу. Метод *checkout(...)* отображает страницу оформления заказа. Вычисляется и передается общая

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

стоимость товаров, которые пользователь добавил в корзину. Метод *paymentPage(...)* отображает страницу с информацией о способах оплаты. Метод *checkout(...)* обрабатывает запрос на оформление заказа и добавляет его в базу данных. Этот метод вызывается при отправке формы с данными о заказе. Метод *bookService.confirmOrder(currentUser)* подтверждает заказ и добавляет его в базу данных. Метод *getBook(...)* получает информацию о книге и ее содержимое для текущего пользователя. Он учитывает параметры страничного вывода и передает информацию на страницу *orders/viewer*, на которой отображается содержимое книги. Метод *getProductInfo(...)* получает информацию о продукте по его идентификатору и отображает ее на странице */catalog/product*. Если продукта не существует, пользователь будет перенаправлен на главную страницу магазина. Метод *editProduct(...)* обрабатывает запрос на редактирование информации о книге. Его можно использовать только администратором. Метод *bookService.update(book)* обновляет информацию о книге в базе данных, а *bookService.delete(book)* удаляет книгу из базы данных. Метод *getProductInfo(...)* обрабатывает запрос на добавление/удаление товаров из корзины. Он вызывается при нажатии на кнопку добавления/удаления товара в корзину и изменяет соответствующие данные в базе данных. Также класс содержит некоторые дополнительные методы и поля, такие как *bookService*, *uploadPath* и другие.

RegistrationController – контроллер для обработки запросов, связанных с регистрацией и активацией пользователей в магазине. Метод *registration()* отображает страницу для регистрации нового пользователя на веб-странице. Метод *addUser()* обрабатывает форму регистрации нового пользователя. Он принимает данные пользователя из формы, проверяет их на корректность, выполняет проверку *reCAPTCHA* и добавляет нового пользователя в базу данных магазина. Если данные не проходят проверку или пользователь с таким именем уже существует, то метод возвращает ошибку и отображает соответствующее сообщение на веб-странице. Метод *activate()* обрабатывает

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

запрос на активацию пользователя по уникальному коду, отправленному на электронную почту при регистрации пользователя. Если код верен, то метод помечает пользователя как активированного в базе данных магазина и отображает сообщение об успешной активации на веб-странице. В противном случае метод отображает ошибку. Для работы с *API reCAPTCHA* используется библиотека *RestTemplate*. Код ключа *reCAPTCHA* хранится в файле *application.properties* и загружается в переменную *secret* с помощью аннотации *@Value*.

UserController – контроллер, который будет обрабатывать запросы на регистрацию нового пользователя, аутентификацию и авторизацию существующих пользователей, а также запросы на изменение персональной информации пользователя. Содержит несколько методов для обработки различных запросов: *userList* – получает список всех пользователей и передает его в модель представления для отображения на странице; *userEditForm* – получает данные о конкретном пользователе и передает их в модель представления для формирования страницы редактирования профиля пользователя; *userSave* – сохраняет изменения в профиле пользователя, основываясь на данных, переданных через параметры запроса; *getProfile* – получает данные о текущем пользователе и передает их в модель представления для отображения на странице профиля; *updateProfile* – сохраняет изменения в профиле текущего пользователя, основываясь на данных, переданных через параметры запроса. Контроллер также использует аннотации *@Controller*, *@RequestMapping*, *@GetMapping*, *@PostMapping*, *@RequestParam*, *@PathVariable*, *@PreAuthorize* и *@Autowired* для настройки маршрутов, обработки параметров запроса и выдачи доступа к определенным функциям только определенным пользователям, которые обладают определенными правами доступа.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.6.2 Сервисы

Сервисы (*services*) – классы, которые содержат бизнес-логику. Они вызываются контроллерами и выполняют задачи, связанные с работой с базой данных, проверкой прав доступа и другими действиями, необходимыми для обработки запросов. Созданы следующие сервисы:

- *MailSender*;
- *OpenAIService*;
- *UserService*;
- *BookService*.

MailSender – сервис для отправки электронных писем, используя *JavaMailSender* и *SimpleMailMessage* из *Spring Framework*. Значение переменной «*username*» получается из файла *application.properties*, который содержит настройки приложения, в том числе конфигурацию связанную с почтовым сервером. Метод «*send*» принимает параметры: адрес электронной почты получателя (*emailTo*), тему (*subject*) и текст сообщения (*message*). Он создает объект класса *SimpleMailMessage*, заполняет его данными и отправляет сообщение через *JavaMailSender*.

OpenAIService – является сервисным классом *OpenAIService*, который содержит метод *generateText()*, используемый для генерации текста с помощью *OpenAI*. Метод *generateText()* принимает строку *prompt* - фразу, слово или предложение, на основе которых будет генерироваться текст. Для отправки запроса на сервер *OpenAI* используется библиотека *OkHttp3* и класс *OkHttpClient* для создания клиента. Запрос выполняется методом *POST* с передачей *JSON*-запроса, который включает параметры модели и входных данных. Ключ авторизации передается в заголовке запроса. В случае успешного выполнения запроса, ответ сервера парсится с помощью библиотеки *Jackson ObjectMapper* в объект класса *Generator*, из которого получается сгенерированный текст. Если сервер не отвечает после шести попыток, метод возвращает строку «Нет ответа от сервера *Open AI*».

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

UserService - сервис, который обеспечивает работу с пользователями в приложении. Класс содержит метод *loadUserByUsername()*, который реализует интерфейс *UserDetailsService* из Spring Security и используется для аутентификации пользователей. Метод *addUser()* добавляет нового пользователя в базу данных и отправляет ему электронное письмо для активации учетной записи. Пользователь сохраняется с хэшированным паролем, случайно сгенерированным кодом активации и ролью *USER*. Методы *sendMessage()* и *activateUser()* используются для отправки сообщения на электронную почту пользователя и активации учетной записи соответственно. Метод *findAll()* возвращает всех зарегистрированных пользователей в системе. Метод *saveUser()* используется для изменения имени пользователя и его ролей в системе. Метод *updateProfile()* позволяет обновить профиль пользователя, в том числе изменить пароль или адрес электронной почты. Если адрес электронной почты пользователя изменен, то генерируется новый код активации и отправляется на новый адрес электронной почты. Кроме того, класс содержит переменные, которые инжектируются через аннотацию *@Value* из файла настроек: *hostname* - имя хоста, на котором запущено приложение, а также *PasswordEncoder* и *MailSender*, которые обеспечивают хеширование пароля и отправку электронной почты соответственно.

BookService – сервисный класс, который предоставляет методы для работы с книгами и заказами в приложении. В данном классе определены различные методы, которые обрабатывают запросы на получение, изменение и удаление данных о книгах в базе данных. Методы *messageList()* и *messageListOnlyAvailable()* помогают отфильтровать список книг для отображения пользователю, в зависимости от его прав доступа и интересов.

Также класс содержит методы для добавления/удаления книг из корзины пользователя, подтверждения заказа, обновления состояния заказа и т.д. Содержит следующие методов для обработки различных запросов:

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

- *messageList*. Получает список всех книг постранично, основываясь на параметрах *pageable*, *filter* и *user*;
- *messageListOnlyAvailable*. Получает список только доступных книг постранично, основываясь на параметрах *pageable*, *filter* и *user*;
- *getBasketForUser*. Получает список книг, которые пользователь добавил в корзину;
- *getOrderForUser*. Получает список книг, которые пользователь заказал;
- *getIssuedForUser*. Получает информацию о книге, которую пользователь арендовал, основываясь на параметрах *currentUser* и *bookId*;
- *getFormedForUser*. Получает список книг, которые пользователь заказал и которые были сформированы для доставки;
- *getCloseOrderForUser*. Получает список закрытых заказов, которые сделал пользователь;
- *getProductById*. Получает книгу по ее идентификатору (*id*);
- *orderListForUser*. Получает список заказов пользователя;
- *addOrDeleteIfExist*. Добавляет или удаляет книгу из корзины пользователя, основываясь на параметрах *currentUser* и *book*;
- *existByUserAndBook*. Проверяет, существует ли заказ на книгу от пользователя;
- *getOrderByUserProduct*. Получает заказ пользователя для конкретной книги;
- *confirmOrder*. Изменяет статус заказа пользователя с *Status.BASKET* на *Status.FORMED*;
- *update*. Обновляет статус заказа или информацию о книге, основываясь на параметрах *order_id*, *status* или *book*;
- *getOrders*. Получает список активных заказов для администратора;
- *delete*. Удаляет книгу из базы данных вместе со всеми связанными заказами и данными о контенте;

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

- *createBook*. Создает новую книгу;
- *saveBook*. Сохраняет информацию о книге и ее контенте;

Класс также содержит несколько поля, которые используются внутри методов, а также зависимости на репозитории для работы с данными: *BookRepo*; *OrderRepo*; *BookContentRepo*.

1.6.3 Репозитории

Репозитории (*repositories*) – классы, которые содержат код для работы с базой данных. Они предоставляют методы для выполнения операций *CRUD* (создание, чтение, обновление и удаление) и поиска книг, заказов, контента и других данных в базе данных. Для работы с БД созданы следующие репозитории:

- *BookRepo*;
- *OrderRepo*;
- *UserRepo*;
- *BookContentRepo*.

BookRepo – содержит методы для работы с объектами *Book* в базе данных. Является интерфейсом репозитория для доступа к данным объектов класса *Book*. Он наследуется от *CrudRepository<Book, Long>*, что означает, что предоставляет стандартные *CRUD*-методы (*create*, *read*, *update*, *delete*) для работы с базой данных. Кроме того, в этом интерфейсе определены несколько методов с аннотацией *@Query*, которые позволяют выполнить пользовательский запрос к базе данных при помощи *JPA* (*Java Persistence API*). Запросы используются для поиска книг с условиями фильтрации и сортировки, а также для получения заказов и их статусов. Каждый запрос создает список объектов класса *BookDto* с заданными полями: *book*, *status*, *date_order*. Некоторые из методов принимают объект *User* и/или *FilterForm*, чтобы добавить дополнительные параметры для поиска.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

BookContentRepo – интерфейс репозитория, который расширяет *CrudRepository*, предоставляя базовые методы для работы с данными в таблице *book_content* базы данных, предназначен для работы с текстами книг;

OrderRepo – содержит методы для работы с объектами *Order* в базе данных. Является репозиторием для работы с заказами (*Order*) в базе данных приложения. В этом интерфейсе определены методы для выполнения операций чтения/записи в базу данных, такие как получение списка всех заказов, получение заказов по статусу, получение заказов пользователя и т.д. Некоторые методы имеют аннотацию *@Query*, которая позволяет написать запрос на языке *SQL* или *JPQL* (языке запросов *Java Persistence API*), чтобы извлечь данные из базы данных. Метод *findByOrder()* использует запрос *JPQL* для поиска заказов, связанных с конкретным пользователем и книгой. Метод *setStatus()* использует запрос *JPQL* для обновления статуса заказа с текущего статуса на новый статус. Метод *deleteByBookId()* использует аннотацию *@Modifying*, которая указывает, что запрос изменяет данные в базе данных. Она используется в сочетании с аннотацией *@Query* для выполнения модифицирующих запросов, таких как удаление записей из базы данных. Интерфейс *OrderRepo* расширяет *CrudRepository*, что позволяет ему наследовать методы для создания, чтения, обновления и удаления данных в базе данных.

UserRepo – репозиторий, который содержит методы для работы с пользователями в БД. В этом интерфейсе определены два дополнительных метода: *findByUsername(String username)* – метод для поиска пользователя по имени пользователя (*username*), который используется для аутентификации пользователей; *findByActivationCode(String code)* – метод для поиска пользователя по коду активации (*activationCode*), который используется для активации учетной записи пользователя после регистрации. Каждый из этих методов возвращает объект *User*, соответствующий заданному условию поиска.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.6.4 Модели

Модели (*models*) - классы, которые определяют структуру объектов, используемых в приложении. В нашем случае, будут использованы следующие модели:

- книга (*Book*);
- заказ (*Order*);
- пользователь (*User*);
- роль (*Role*);
- текст книги (*BookContent*);
- статус (*Status*).

Модель *Book* – книга, которая содержит название, автора, издательство и другие атрибуты книги. который представляет сущность книги в приложении для интернет-магазина книг. Аннотация «*@Entity*» указывает, что данный класс является сущностью (*entity*) JPA, т.е. он будет отображаться на таблицу в базе данных. Поля класса соответствуют полям, которые составляют описание книги: название («*title*»), описание («*description*»), автор («*author*»), цена («*price*») и имя файла («*filename*»). Аннотации «*@Column*» указывают, что каждое из этих полей должно быть сохранено в отдельном столбце таблицы в базе данных. Для первичного ключа используется аннотация «*@Id*», а также аннотация «*@GeneratedValue*», которая указывает, каким образом будет генерироваться значение первичного ключа (в данном случае автоматически). В классе определены методы-конструкторы и метод «*set()*», который устанавливает значения полей на основе переданного объекта класса «*Book*». Аннотация «*@Data*» объединяет все поля класса в одну аннотацию, что позволяет автоматически создать геттеры/сеттеры, методы «*equals()*», «*hashCode()*» и «*toString()*». Валидация данных осуществляется с помощью аннотаций «*@NotBlank*», «*@Length*» и т.д., которые проверяют, что соответствующие значения не являются пустыми, не превышают определенной длины и т.д. В

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

классе есть поле «*bookContent*», связанное с другой сущностью «*BookContent*» через аннотацию «*@OneToOne*». Это означает, что каждая книга может иметь только один контент (текст книги), который будет храниться в базе данных в отдельной таблице.

Модель *Order*, которая содержит информацию о заказе – список книг, статус заказа, адрес доставки и т.д. Аннотация *@Table(name = «orders»)* указывает имя таблицы базы данных, которая будет соответствовать этой сущности. Конструктор *public Order(User user, Book book, Date dateOrder)* создает новый объект класса *Order* с тремя параметрами: пользователь (*user*), книга (*book*) и дата заказа (*dateOrder*). Аннотация *@Id* указывает, что *id* является первичным ключом таблицы базы данных. Аннотация *@GeneratedValue(strategy = GenerationType.AUTO)* указывает, как будет генерироваться значение для первичного ключа *id*. Аннотации *@ManyToOne* и *@JoinColumn* определяют связь между таблицами сущностей. В данном случае *Order* имеет связь *ManyToOne* с таблицами *User* и *Book*, причем *user_id* и *book_id* – это внешние ключи таблицы *orders*, которые связываются с первичными ключами таблиц *users* и *books*. Аннотация *@Enumerated(EnumType.STRING)* указывает, что поле *status* будет отображаться в базе данных как строка. Аннотация *@Temporal(TemporalType.TIMESTAMP)* указывает, что поле *date_order* будет храниться в базе данных как временная метка.

Модель *User* – класс, который хранит информацию о пользователе. ФИО, *email*, адрес доставки, телефон и т.д. Представляет сущность пользователя в приложении для интернет-магазина книг. Аннотация «*@Table(name = “usr”)*» указывает, что соответствующая таблица в базе данных называется «*usr*». Поля класса соответствуют полям, которые составляют описание пользователя: идентификатор («*id*»), имя пользователя («*username*»), пароль («*password*»), активность («*active*»), *email*, код активации («*activationCode*») и роли («*roles*»). В данном классе реализован

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

интерфейс *UserDetails* для обеспечения аутентификации и авторизации пользователей в Spring Security. Поле «*roles*» представлено как множество ролей (*Set<Role>*), связанных с данным пользователем. Роли хранятся в отдельной таблице «*user_role*» с использованием аннотаций «*@ElementCollection*», «*@CollectionTable*» и «*@Enumerated*». Есть также поля «*books*» и «*orders*», связанные через аннотации «*@OneToMany*» с другими сущностями «*Book*» и «*Order*». Метод «*equals()*» и «*hashCode()*» переопределены для сравнения объектов класса «*User*» по полю «*id*». Пользователь может иметь несколько ролей, в том числе администраторскую. Метод «*isAdmin()*» проверяет, содержит ли множество ролей данного пользователя роль «*ADMIN*». Этот класс также содержит переопределенные методы из интерфейса *UserDetails*: «*getUsername()*», «*isAccountNonExpired()*», «*isAccountNonLocked()*», «*isCredentialsNonExpired()*» и «*isEnabled()*».

Модель *Role* – используется для определения ролей пользователей в приложении для интернет-магазина книг. Класс «*Role*» реализует интерфейс «*GrantedAuthority*», который определяет, что каждый экземпляр этого класса является одной из возможных ролей пользователей в системе. Перечисленные константы «*USER*» и «*ADMIN*» представляют собой две роли пользователей: обычного пользователя и администратора соответственно. Метод «*getAuthority()*» возвращает имя роли как строку. Такое использование перечисления позволяет удобно и безопасно хранить роли пользователей в базе данных или передавать их между слоями приложения.

Модель *BookContent* – класс, который хранит текст книги. Представляет сущность контента книги в приложении для интернет-магазина книг. Аннотация «*@Table(name = "book_content")*» указывает, что соответствующая таблица в базе данных называется «*book_content*». Поля класса соответствуют полям, которые составляют описание контента книги: идентификатор книги («*book_id*»), книга («*book*») и содержание («*content*»). В

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

данном случае связь между классами «*Book*» и «*BookContent*» установлена через аннотацию «*@OneToOne(fetch = FetchType.LAZY)*», которая указывает, что для каждой книги может быть только один объект контента, и он будет загружен из базы данных только при необходимости (лениво). Поле «*book_id*» используется как первичный ключ таблицы «*book_content*», который ссылается на первичный ключ таблицы «*book*». Это достигается с помощью аннотации «*@MapsId*» в сочетании с полем «*book*» типа «*Book*», которое является объектом, связанным с этим контентом. В классе определены методы-конструкторы, в том числе конструктор, принимающий объекты «*book*» и «*content*», что позволяет установить связь между книгой и ее контентом.

Модель *Status* – перечисление, которое определяет возможные статусы заказов, которые могут быть присвоены экземпляру класса *Order*. Метод *get()* возвращает строковое значение для каждого элемента перечисления, соответствующее значению статуса. Методы типа *isXXX()* проверяют, является ли текущий экземпляр перечисления определенным статусом. Например, метод *isBasket()* проверяет, является ли текущий экземпляр перечисления статусом *BASKET*. Перечисление *Status* содержит также элемент *UNKNOWN*, который используется, если статус неизвестен или не указан.

Для обработки ответов от *OpenAI* используются следующие классы:

- *CompletionChoice*;
- *Generator*;
- *Message*;
- *Usage*.

CompletionChoice – представляет объект, содержащий информацию о выборе пользователя в процессе завершения определенного действия или процесса. Аннотация *@JsonProperty* используется для указания имени соответствующего поля в *JSON* формате, что позволяет *Jackson* (библиотека,

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

используемая в *Spring Framework* для преобразования данных между объектами *Java* и *JSON*) правильно отображать данные. Аннотация *@Data* из *Lombok* генерирует стандартные методы *toString*, *equals*, *hashCode* и геттеры/сеттеры для всех полей класса автоматически. Класс имеет три поля:

- *Message*. Объект, содержащий сообщение, связанное с выбором пользователя;
- *finishReason*. Строка, содержащая причину завершения действия или процесса;
- *index*. Целочисленное значение, содержащее индекс выбора пользователя.

Generator. Класс для обработки ответа на запрос. Класс имеет 6 полей:

- *id*. Строка, содержащая уникальный идентификатор;
- *object*. Строка, содержащая тип объекта;
- *created*. Целочисленное значение, содержащее дату и время создания;
- *model*. Строка, содержащая модель ответа;
- *choices*. Массив объектов типа *CompletionChoice*, содержащий информацию о выборе пользователя в процессе завершения определенного действия или процесса;
- *usage*. Объект типа *Usage*, содержащий информацию об использовании ответа.

- *Message*. Представляет объект сообщения, используемый в определенном API. Класс *Message* имеет два поля:

- *Role*. Строковое значение, которое содержит тип сообщения;
- *Content*. Строковое значение, которое содержит текст сообщения.

Usage. Используется для десериализации *JSON*-ответа от *API*, которая возвращает информацию о количестве токенов, использованных при генерации текста с помощью модели языка (*OpenAI GPT-3*). Каждое из трех полей («*promptTokens*», «*completion_tokens*» и «*totalTokens*») соответствует

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

количеству токенов, использованных на разных этапах работы с моделью - для начальной части текста (prompt), для сгенерированного текста (завершения) и в общей сложности.

1.6.5 Конфигурация

Конфигурация (*configuration*) – классы, которые содержат настройки приложения, такие как подключение к базе данных и другие параметры. Созданы конфигурационные классы, которые содержат информацию о базе данных, порте сервера, ключах доступа и прочих параметрах, необходимые для запуска и работы приложения:

- *WebSecurityConfig*;
- *MvcConfig*;
- *MailConfig*;
- *EncriptionConfig*.

WebSecurityConfig – конфигурационный класс, который позволяет настраивать защиту веб-приложения. Он содержит методы для настройки механизмов аутентификации и авторизации, а также определяет разрешенные и запрещенные *URL*-адреса, доступные для пользователей. Для работы *Spring Security* добавлена аннотация *@EnableWebSecurity*, которая сообщает *Spring* о том, что этот класс будет использоваться для настройки безопасности веб-приложения. Аннотация *@EnableGlobalMethodSecurity (prePostEnabled = true)* позволяет использовать аннотации для задания прав доступа непосредственно к методам контроллеров. В данном коде определены два метода *configure(HttpSecurity http)* и *configure(AuthenticationManagerBuilder auth)*, которые определяют настройки для аутентификации и авторизации. Методы *antMatchers()* и *anyRequest()* определяют *URL*-адреса, для которых требуется авторизация, а *permitAll()* указывает, что эти адреса доступны всем пользователям. Также используется инъекция зависимостей (аннотации *@Autowired*) для получения экземпляров

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

UserService и *PasswordEncoder*, которые будут использоваться при настройке аутентификации.

MailConfig – конфигурационный класс для настройки отправки электронной почты в Spring-приложении. Класс *MailConfig* содержит метод *getMailSender*, который создает и возвращает объект *JavaMailSender*, предоставляемый *Spring Framework* для отправки почты. Внутри метода *getMailSender* определены поля класса, которые получают значения из файла конфигурации *application.properties* при помощи аннотации *@Value*. Эти значения представляют параметры подключения к серверу электронной почты, такие как имя хоста (*host*), порт (*port*) и протокол (*protocol*). Далее, объект *JavaMailSenderImpl* создается и настраивается с использованием значений полей класса, а также свойств *JavaMail API*, таких как протокол передачи почты и режим отладки. Настройки сохраняются в *Properties* и устанавливаются для объекта *JavaMailSenderImpl*. Метод *getMailSender* возвращает настроенный объект *JavaMailSender*, который используется для отправки писем электронной почты из приложения.

MvcConfig – конфигурационный класс для *Spring MVC*, который позволяет настраивать обработку запросов и представлений веб-приложения. Он реализует интерфейс *WebMvcConfigurer* и содержит методы для настройки контроллеров, обработки ошибок, ресурсов и других настроек. В данном коде определены три метода:

- метод *addViewControllers()* позволяет использовать предварительно определенные представления без необходимости создания отдельных контроллеров. В данном случае, когда пользователь будет переходить по адресу «/login», он будет автоматически перенаправлен на страницу «login»;

- метод *addResourceHandlers()* определяет местоположение статических ресурсов (CSS, JavaScript, изображения) и указывает, где они должны быть найдены. В данном примере все статические файлы будут искаться в директории «*classpath:/static/*» и в директории, заданной

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

параметром «*upload.path*», который был получен из файла *application.properties*;

– метод *addInterceptors()* добавляет перехватчик *RedirectInterceptor()*, который будет перенаправлять пользователя на другую страницу, если выполнено определенное условие. В данном примере он просто перенаправляет всех пользователей на главную страницу.

EncryptionConfig – является конфигурационным классом для настройки шифрования паролей в Spring-приложении. В классе определен метод *getPasswordEncoder()*, который создает и возвращает объект *BCryptPasswordEncoder*. *BCryptPasswordEncoder* – это реализация интерфейса *PasswordEncoder* в *Spring Security*, который использует сложный алгоритм хэширования солью для шифрования паролей пользователей. Чем выше значение параметра *strength*, тем более сложным будет алгоритм и тем больше времени нужно будет для его выполнения. В данном случае объект *BCryptPasswordEncoder* создается со значением параметра *strength* равным 8. Это означает, что алгоритм будет использовать 2^8 итераций при генерации хэша пароля. Чем больше итераций, тем безопаснее пароль, но и производительность работы может ухудшаться. Метод *getPasswordEncoder()* помечен аннотацией *@Bean*, что указывает *Spring Framework* на то, что он должен быть исполнен и настроен как бин *Spring*. Это позволяет другим классам в приложении получать доступ к объекту *PasswordEncoder* через механизм внедрения зависимостей (*Dependency Injection*).

1.6.6 Классы утилиты

Обработчики ошибок (*error handlers*) - классы, которые обрабатывают ошибки, возникающие в процессе работы приложения и возвращают соответствующие сообщения об ошибках клиентам. Для упрощения создан глобальный обработчик ошибок, который перехватывает все исключения,

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

возникающие в приложении, и возвращает соответствующие сообщения об ошибках. Классы утилиты представлены ниже.

Обработчики ошибок (*error handlers*) - классы, которые обрабатывают ошибки, возникающие в процессе работы приложения и возвращают соответствующие сообщения об ошибках клиентам. Для упрощения создан глобальный обработчик ошибок, который перехватывает все исключения, возникающие в приложении, и возвращает соответствующие сообщения об ошибках.

RedirectInterceptor – реализует интерфейс *HandlerInterceptorAdapter* из библиотеки Spring Framework. Он используется для перехвата HTTP-запросов и ответов веб-приложения на основе Spring Framework. Конкретно этот класс называется *RedirectInterceptor* и выполняет перенаправление запросов с помощью заголовка «*Turbolinks-Location*». Используется в ситуациях, когда необходимо изменить URL-адрес, по которому пользователь делает запросы, без фактической перезагрузки страницы. Метод *postHandle()* перехватывает ответ от сервера, проверяет наличие объекта *ModelAndView* (который предназначен для передачи данных между контроллером и представлением) и, если он существует, создает новый URL на основе текущего запроса и добавляет заголовок «*Turbolinks-Location*» в ответ. Это позволяет клиентскому коду обновлять содержимое страницы, не перезагружая ее полностью.

1.7 Тестирование

После завершения разработки и запуска сайта, необходимо провести тестирование и оптимизацию, чтобы убедиться, что сайт работает быстро и без ошибок, а пользователи могут легко и удобно пользоваться с сайтом.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.7.1 Главная страница

На главной странице необходимо проверить: загрузку страницы, наличие основных элементов дизайна (шапка, футер, меню), корректных ссылок на другие страницы и проверка работы поисковой строки (см. рисунок 28).

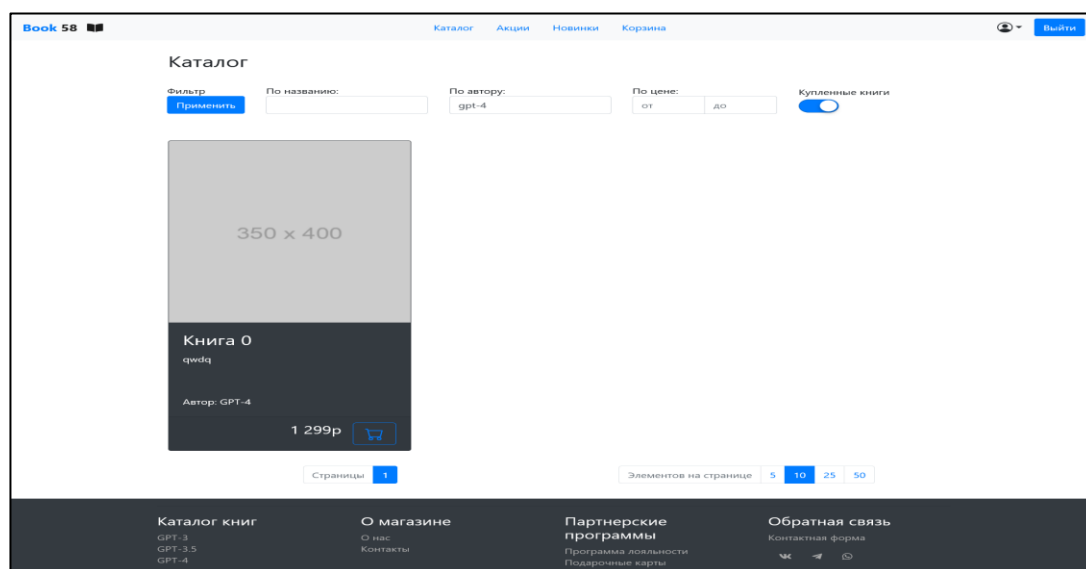


Рисунок 28 – Поиск по автору

Проверка фильтров товаров (по цене, категории, названию и т.д.) на рисунках 29-31.

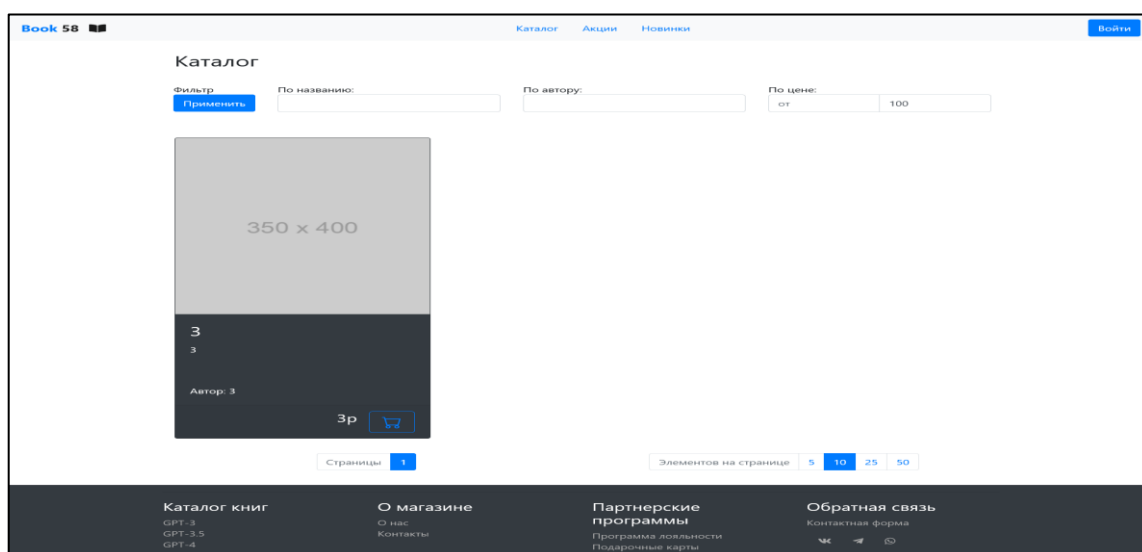


Рисунок 29 – Фильтр по цене до «100»

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

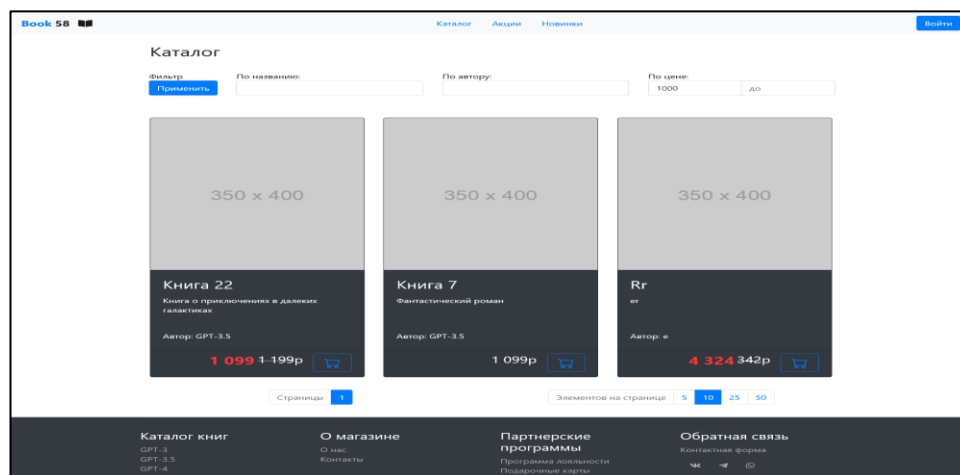


Рисунок 30 – Фильтр по цене от «1000»

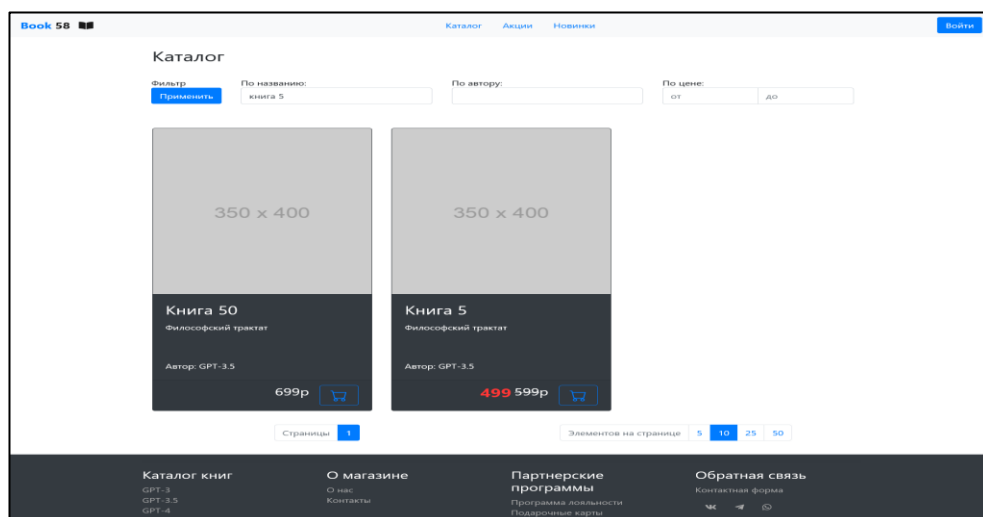


Рисунок 31 – Поиск по названию «книга 5»

1.7.2 Страница авторизации

На страницу авторизации можно попасть двумя способами: по адресу «/login» или при переходе на адреса, для которых требуется авторизация. Проверить необходимо загрузку страницы, наличия формы для ввода логина и пароля (см. рисунок 32).

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Рисунок 32 – Проверка страницы входа

Проверка функционала входа с правильными/неправильными данными на рисунке 33.

Рисунок 33 – Ввод неправильных данных

Проверка перенаправления пользователя на другую страницу после успешной авторизации работает корректно.

1.7.3 Страница личного кабинета

На странице проверена загрузка страницы наличие информации о пользователе (имя, электронная почта и т.д.) Также работает функционал изменения личной информации (см. рисунок 34-39).

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

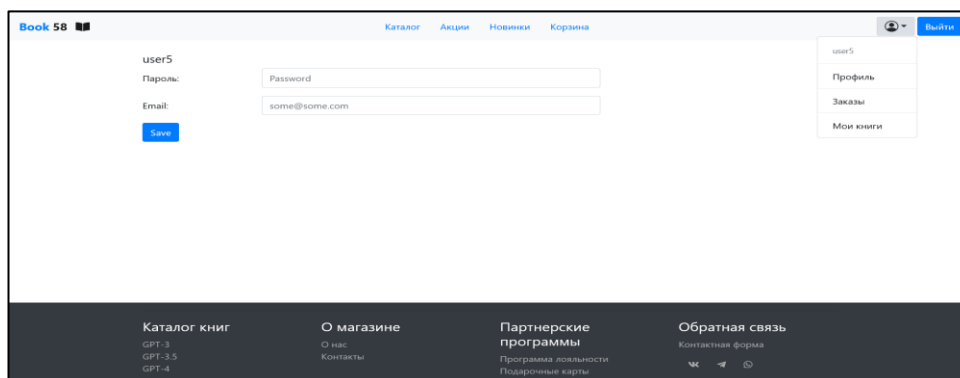


Рисунок 34 – Страница личного кабинета

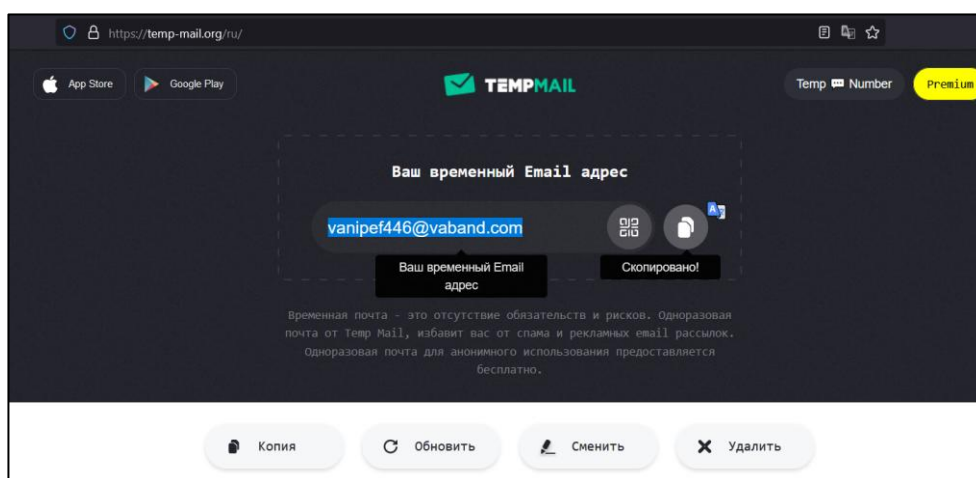


Рисунок 35 – Создание временной email почты

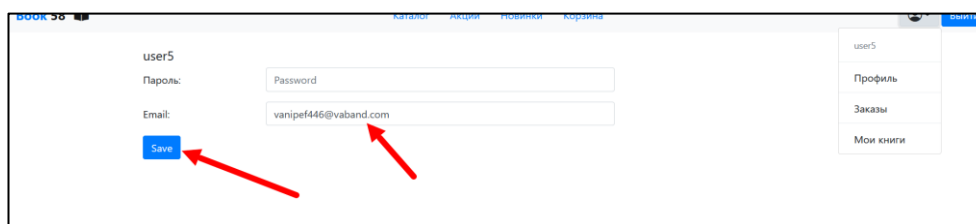


Рисунок 36 – Изменение электронной почты

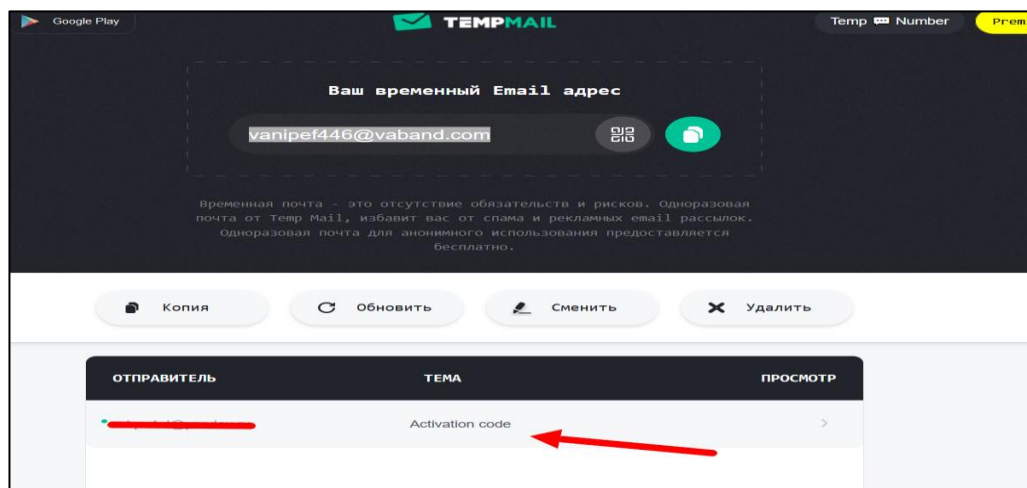


Рисунок 37 – Получение письма с кодом активации

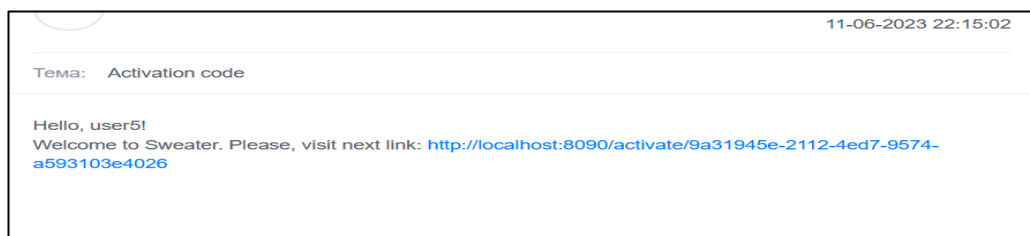


Рисунок 38 – Ссылка для проверки почты

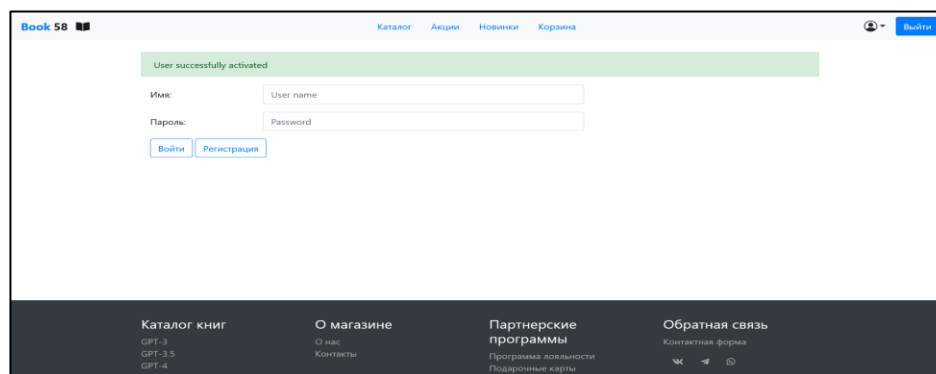


Рисунок 39 – Подтверждение успешной проверки почты

Проверка возможности выхода из аккаунта представлена на рисунках 40, 41.

					ПГУ1.090301.08.001 ПЗ		Лист
Изм.	Лист	№ докум.	Подпись	Дата			

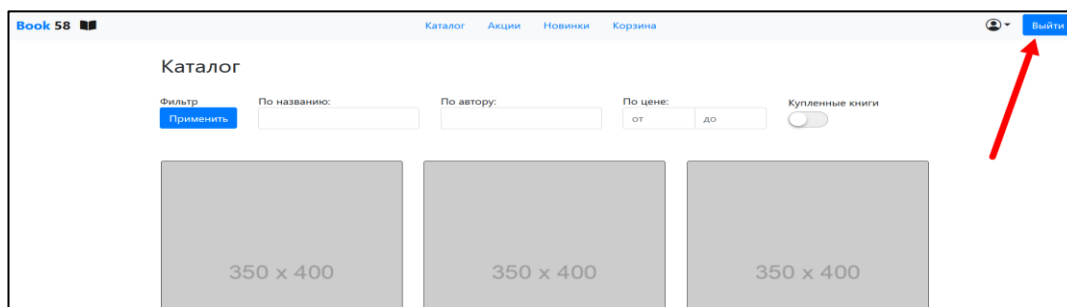


Рисунок 40 – Нажатие на кнопку «Выйти»

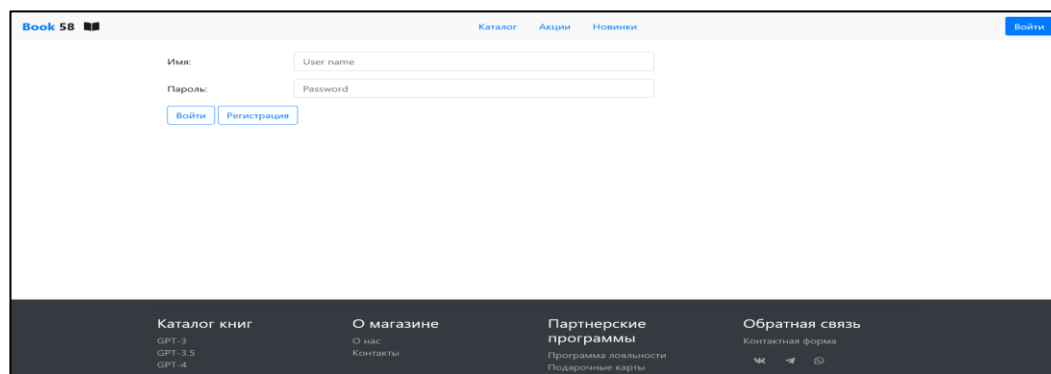


Рисунок 41 – Перенаправление на страницу авторизации

1.7.4 Страницы продуктов

На рисунке 42 проверка загрузки страницы продукта «Книга 47» для пользователя *User10*.

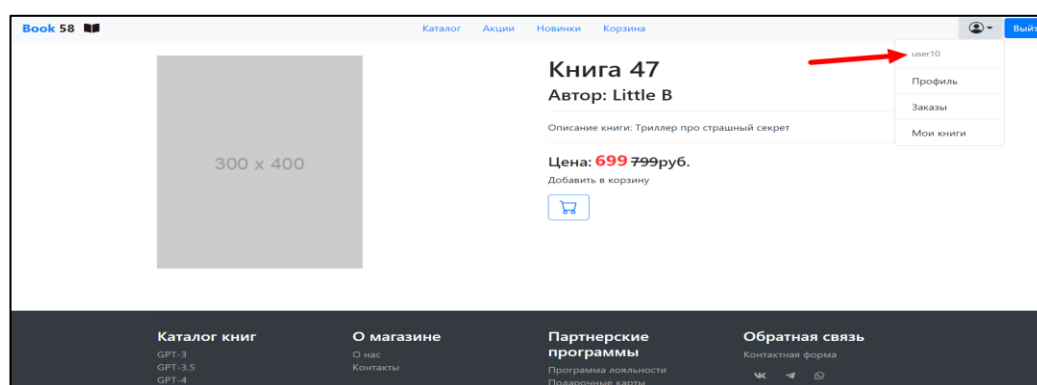


Рисунок 42 – Страница продукта для обычного пользователя

Проверка возможности добавления товара в корзину на рисунках 43, 44.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

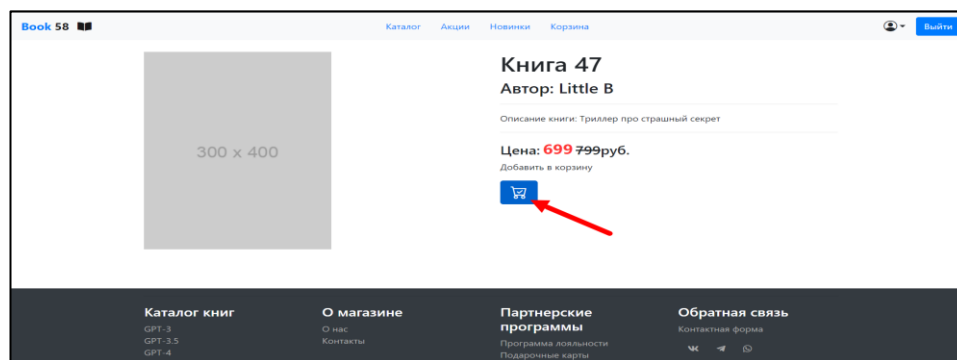


Рисунок 43 – Проверка кнопки добавление в корзину

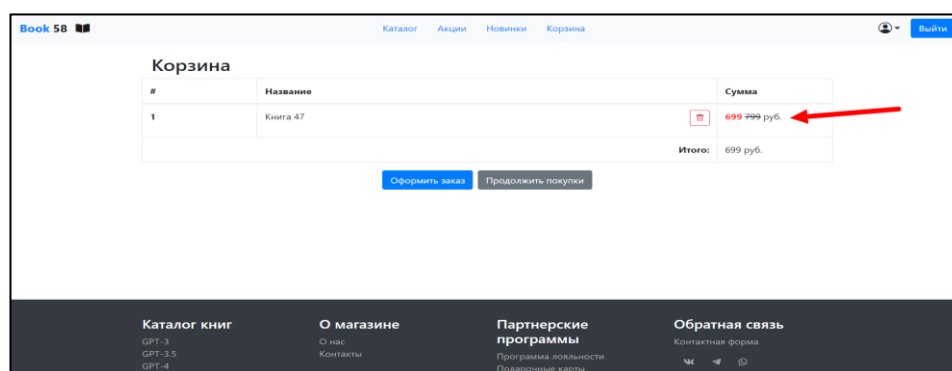


Рисунок 44 – Проверка добавленного товара в корзине

Страница продукта для администратора сайта приведена на рисунке 45. Отличие заключается в возможности изменения и обновление информации о продукте.

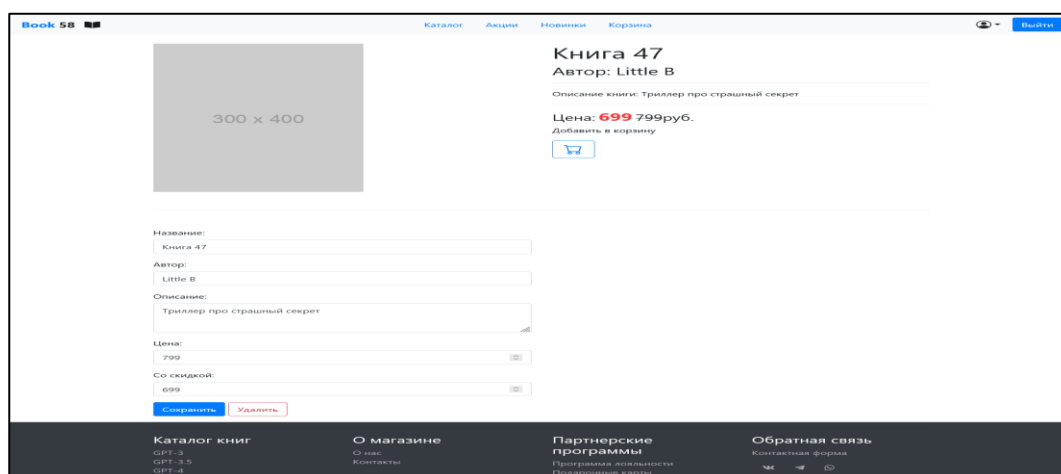


Рисунок 45 – Страница продукта для администратора

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.7.5 Страницы контактов

Проверка наличия контактной информации: адрес, телефон, электронная почта на рисунке 46. Проверка возможности отправки сообщения через форму обратной связи на рисунке 47.

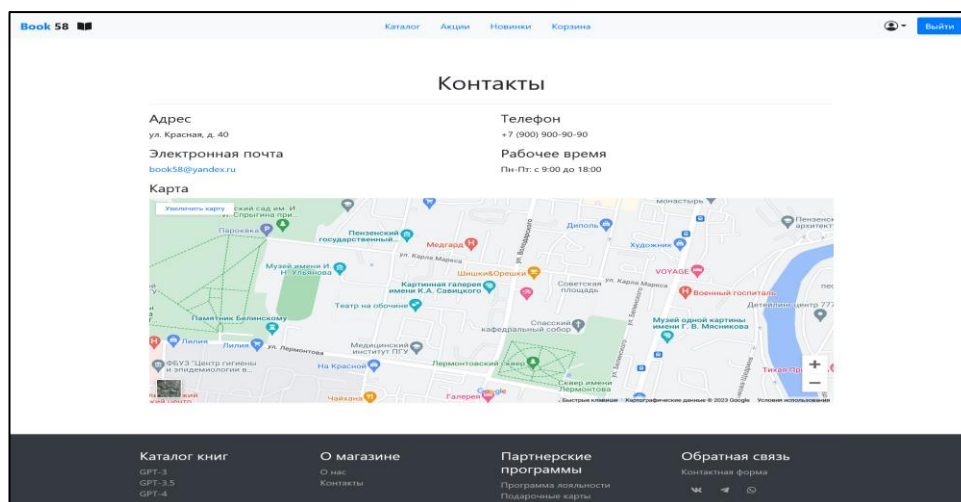


Рисунок 46 – Страница контактов

Рисунок 47 – Страница обратной связи

1.7.6 Страница корзины

Проверка страницы корзины: наличия списка товаров, добавленных в корзину; возможности изменения количества товаров и удаления товаров из корзины (см. рисунок 48 - 50).

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Корзина

#	Название	Сумма
-	-	-
		Итого: 0 руб.

Оформить заказ
Продолжить покупки

Рисунок 48 – Пустая корзина

Корзина

#	Название		Сумма
1	Книга 47		699 799 руб.
2	Книга 46		799 899 руб.
3	Книга 45		499 599 руб.
			Итого: 1 997 руб.

Оформить заказ
Продолжить покупки

Рисунок 49 – Добавлено 3 товара

Корзина

#	Название		Сумма
1	Книга 47		699 799 руб.
2	Книга 45		499 599 руб.
			Итого: 1 198 руб.

Оформить заказ
Продолжить покупки

Рисунок 50 – Удаление 1 товара из корзины

1.7.7 Страница оформления заказа

На странице оформления заказа необходимо проверить наличие информации о сумме заказа, выбора оплаты и форму ввода данных для оплаты, а также правильность заполнения формы (см. рисунки 51, 52). На

рисунках 53, 54 представлена отработка ошибка оплаты (ввод не корректных данных).

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Book 58

КаталогАкцииНовинкиКорзина

Войти

Оплата

Номер карты:

1234123412341234

Имя владельца карты:

Petrov

Срок действия:

11

25

111

Общая сумма заказа:

1 198 руб.

ОплатитьНазад

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

☒ ⚙️ 📧

Рисунок 51 – Страница оплаты заказа

Book 58

КаталогАкцииНовинкиКорзина

Войти

Ваш заказ успешно оформлен!

Спасибо за покупку! Мы свяжемся с вами в ближайшее время для уточнения деталей.

Перейти в каталогПерейти в раздел заказов

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

☒ ⚙️ 📧

Рисунок 52 – Страница подтверждения оплаты, успех

Book 58

КаталогАкцииНовинкиКорзина

Войти

Оплата

Номер карты:

1234123412341234

Имя владельца карты:

Petrov

Срок действия:

11

25

222

Общая сумма заказа:

799 руб.

ОплатитьНазад

Каталог книг

GPT-3
GPT-3.5
GPT-4

О магазине

О нас
Контакты

Партнерские программы

Программа лояльности
Подарочные карты

Обратная связь

Контактная форма

☒ ⚙️ 📧

Неправильные данные карты

Рисунок 53 – Страница оплаты, ввод некорректных данных

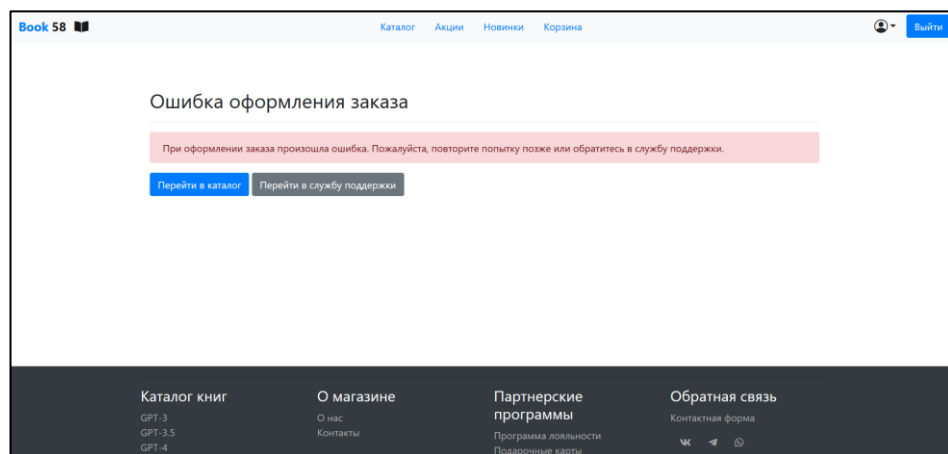


Рисунок 54 – Страница подтверждения оплаты, отклонен платеж

1.7.8 Страница всех заказов

Администратор может просматривать все заказы, и после проверки оплаты предоставлять доступ к товарам (см. рисунки 55, 56).

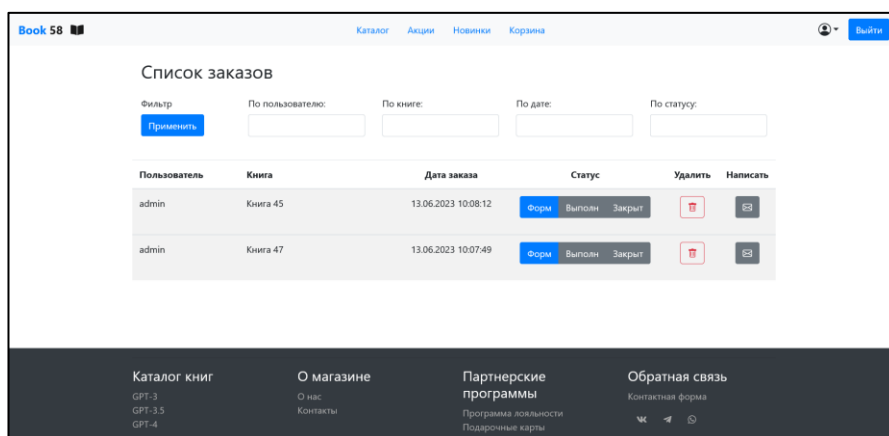


Рисунок 55 – Страница заказов пользователей

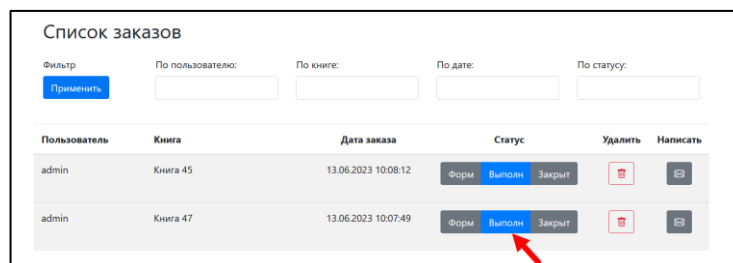


Рисунок 56 – Предоставление доступа к товару

1.7.9 Страница просмотра книг

После оплаты и подтверждения администратора, пользователю доступны оплаченные книги (см. рисунки 57, 58).

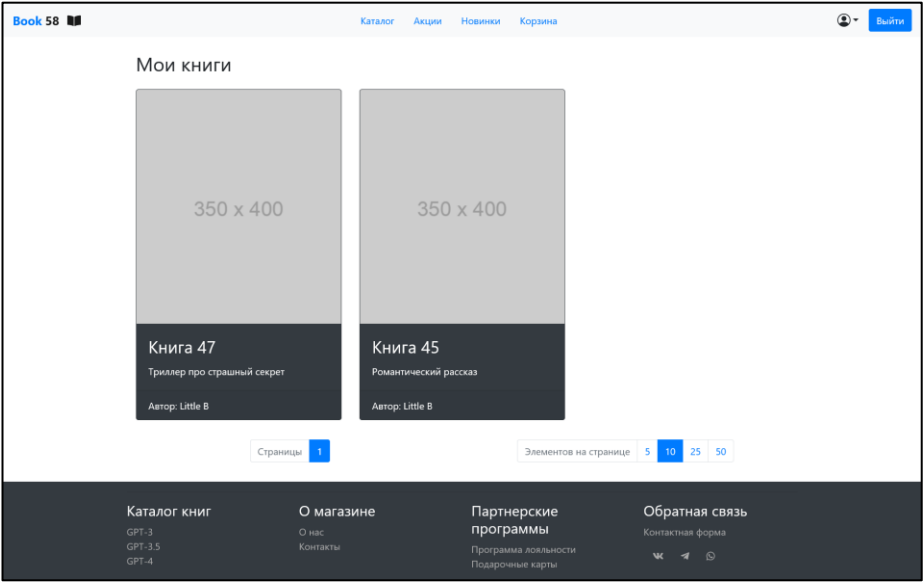


Рисунок 57 – Страница приобретенных книг

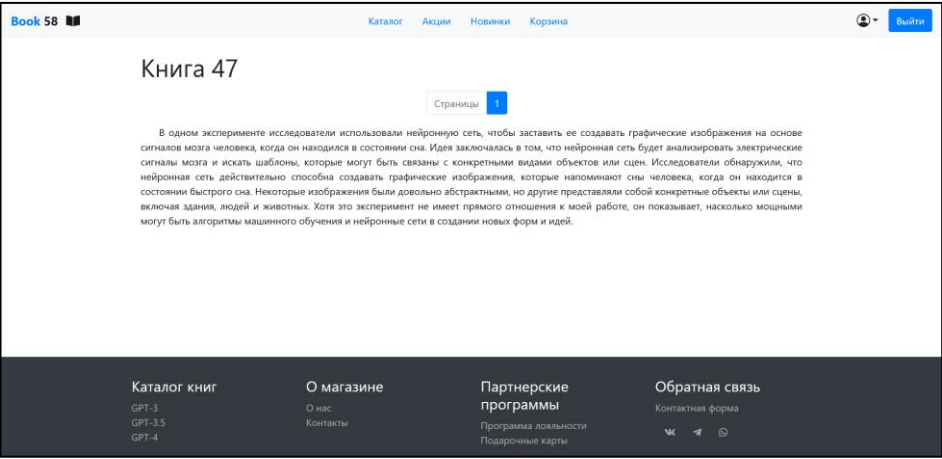


Рисунок 58 – Страница просмотрщика, текст приобретенной книги

1.7.10 Страница с акциями

Проверка наличия информации о текущих акциях и скидках на рисунке 59.

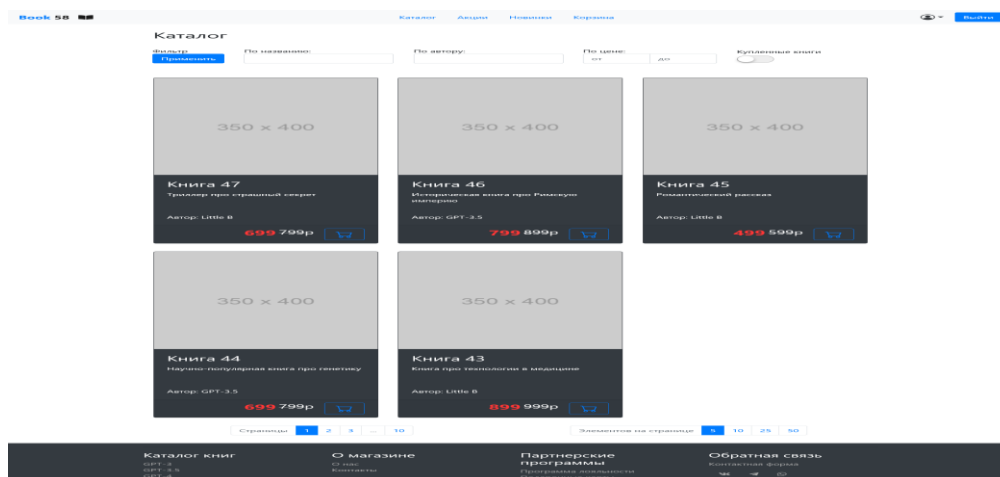


Рисунок 59 – Страница товарами по акции

1.7.11 Страницы с новинками

Проверка наличия информации о новых товарах и возможности добавления нового товара в корзину на рисунке 60.

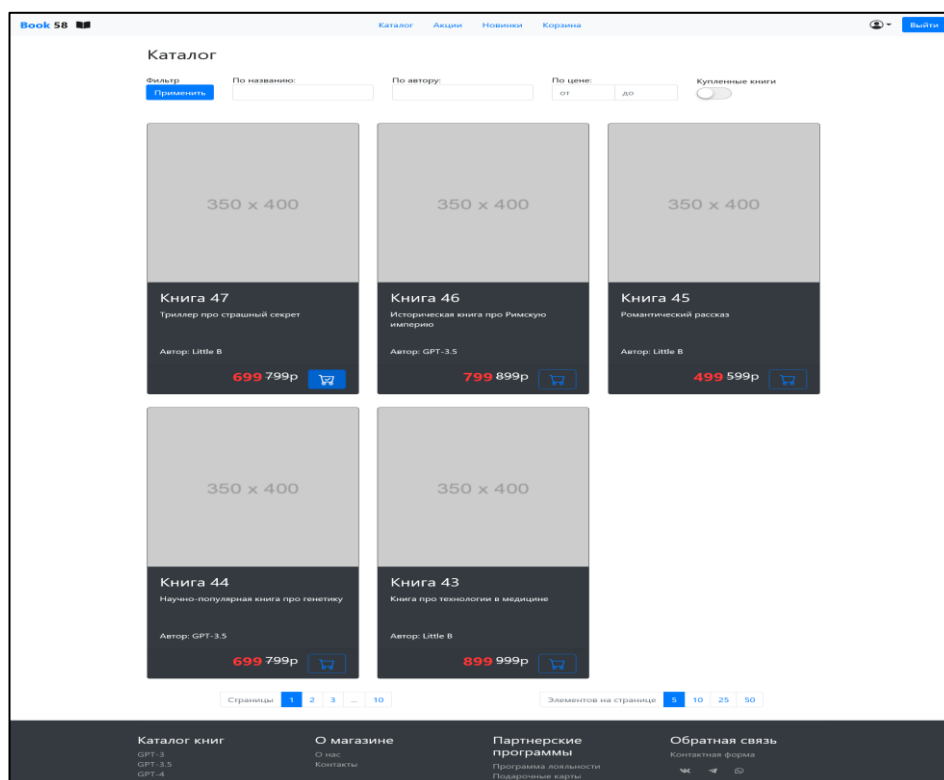


Рисунок 60 – Страница новых книг

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.8 Перспективы развития

К перспективам развития: запустить интернет-магазин и обеспечить его стабильную работу. Улучшать и развивать интернет-магазин, добавляя новый функционал и улучшая существующий на основе отзывов пользователей и аналитики работы магазина.

Улучшение поиска товаров на сайте. Можно добавить возможность поиска книг по жанру, автору или *ISBN*. На главной странице добавить возможность сортировки товаров.

Разработка мобильного приложения для интернет-магазина. Это позволит клиентам быстро и удобно делать покупки через свои мобильные устройства. На странице регистрации и авторизации: возможно добавить возможность входа через социальные сети.

Внедрение системы рекомендаций. Эта функция поможет клиентам находить книги, которые им могут быть интересны, на основе их предыдущих покупок и просмотров. Добавление системы отзывов и рабочей программы лояльности.

Сделать сайт более интерактивным и привлекательным для пользователей, добавив возможность просмотра отрывков из книг или видеообзоров. Предоставление более широкого выбора способов оплаты. Добавить новые альтернативные методы оплаты, чтобы сделать процесс оплаты более удобным для клиентов.

Создание блога или раздела новостей, где будут публиковаться информация о новинках, скидках и акциях. Это поможет привлечь новых клиентов и удержать старых.

Расширение ассортимента книг. Можно добавить новые жанры, авторов или языки, чтобы привлечь больше разных клиентов. Для создания книг добавить различные ИИ, способные генерировать тексты. Добавление генерации изображения обложки книги. Оптимизация программного кода.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

2 Безопасность жизнедеятельности

1.9 Обеспечение электробезопасности

Действие электрического тока на живую ткань носит разносторонний и своеобразный характер [13]. Проходя через организм человека, электроток производит термическое, электролитическое, механическое и биологическое действия. Термическое действие тока проявляется ожогами отдельных участков тела, нагревом до высокой температуры органов, расположенных на пути тока, вызывая в них значительные функциональные расстройства. Электролитическое действие тока выражается в разложении органической жидкости, в том числе крови, в нарушении ее физико-химического состава. Механическое действие тока приводит к расслоению, разрыву тканей организма в результате электродинамического эффекта, а также мгновенного взрывоподобного образования пара из тканевой жидкости и крови. Биологическое действие тока проявляется раздражением и возбуждением живых тканей организма, а также нарушением внутренних биологических процессов.

Электротравмы условно разделяют на общие и местные. К общим относят электрический удар, при котором процесс возбуждения различных групп мышц может привести к судорогам, остановке дыхания и сердечной деятельности. Остановка сердца связана с фибрилляцией — хаотическим сокращением отдельных волокон сердечной мышцы (фибрилл). К местным травмам относят ожоги, металлизацию кожи, механические повреждения, электроофтальмии. Металлизация кожи связана с проникновением в нее мельчайших частиц металла при его расплавлении под влиянием чаще всего электрической дуги.

Исход поражения человека электротоком зависит от многих факторов: силы тока и времени его прохождения через организм, характеристики тока (переменный или постоянный), пути тока в теле человека, при переменном

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

токе — от частоты колебаний. Ток, проходящий через организм, зависит от напряжения прикосновения, под которым оказался пострадавший, и суммарного электрического сопротивления, в которое входит сопротивление тела человека. Величина последнего определяется в основном сопротивлением рогового слоя кожи, составляющим при сухой коже и отсутствии повреждений сотни тысяч ом. Если эти условия состояния кожи не выполняются, то ее сопротивление падает до 1 кОм. При высоком напряжении и значительном времени протекания тока через тело сопротивление кожи падает еще больше, что приводит к более тяжелым последствиям поражения током. Внутреннее сопротивление тела человека не превышает нескольких сотен ом и существенной роли не играет.

На сопротивление организма воздействию электрического тока оказывает влияние физическое и психическое состояние человека. Нездоровье, утомление, голод, опьянение, эмоциональное возбуждение приводят к снижению сопротивления.

Допустимым считается ток, при котором человек может самостоятельно освободиться от электрической цепи. Его величина зависит от скорости прохождения тока через тело человека: при длительности действия более 10 с — 2 мА, при 10 с и менее — 6 мА. Ток, при котором пострадавший не может самостоятельно оторваться от токоведущих частей, называется не отпускающим. Переменный ток опаснее постоянного, однако при высоком напряжении (более 500 В) опаснее постоянный ток. Из возможных путей протекания тока через тело человека (голова — рука, голова — ноги, рука — рука, нога — рука, нога — нога и т. д.) наиболее опасен тот, при котором поражается головной мозг (голова — руки, голова — ноги), сердце и легкие (руки — ноги). Неблагоприятный микроклимат (повышенная температура, влажность) увеличивает опасность поражения током, так как влага (пот) понижает сопротивление кожных покровов.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

При гигиеническом нормировании ГОСТ 12.1.038—82* устанавливает предельно допустимые напряжения прикосновения и токи, протекающие через тело человека (рука — рука, рука — нога) при аварийном режиме работы электроустановок производственного и бытового назначения постоянного и переменного тока частотой 50 и 400 Гц.

1.10 Ионизирующие излучения

Ионизирующее излучение вызывает в организме цепочку обратимых и необратимых изменений. Пусковым механизмом воздействия являются процессы ионизации и возбуждения атомов и молекул в тканях. Диссоциация сложных молекул в результате разрыва химических связей — прямое действие радиации. Существенную роль в формировании биологических эффектов играют радиационно-химические изменения, обусловленные продуктами радиолиза воды. Свободные радикалы водорода и гидроксильной группы, обладая высокой активностью, вступают в химические реакции с молекулами белка, ферментов и других элементов биоткани, что приводит к нарушению биохимических процессов в организме. В результате нарушаются обменные процессы, замедляется и прекращается рост тканей, возникают новые химические соединения, не свойственные организму. Это приводит к нарушению деятельности отдельных функций и систем организма. Индуцированные свободными радикалами химические реакции развиваются с большим выходом, вовлекая в процесс сотни и тысячи молекул, не задействованных излучением. В этом состоит специфика действия ионизирующего излучения на биологические объекты. Эффекты развиваются в течение разных промежутков времени: от нескольких секунд до многих часов, дней, лет.

Ионизирующая радиация при воздействии на организм человека может вызвать два вида эффектов, которые клинической медициной относятся к болезням: детерминированные пороговые эффекты (лучевая болезнь,

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

лучевой ожог, лучевая катаракта, лучевое бесплодие, аномалии в развитии плода и др.) и стохастические (вероятностные) беспороговые эффекты (злокачественные опухоли, лейкозы, наследственные болезни).

Острые поражения развиваются при однократном равномерном гамма-облучении всего тела и поглощенной дозе выше 0,5 Гр. При дозе 0,25...0,5 Гр могут наблюдаться временные изменения в крови, которые быстро нормализуются. В интервале дозы 0,5...1,5 Гр возникает чувство усталости, менее чем у 10 % облученных может наблюдаться рвота, умеренные изменения в крови. При дозе 1,5...2,0 Гр наблюдается легкая форма острой лучевой болезни, которая проявляется продолжительной лимфопенией, в 30...50 % случаев — рвота в первые сутки после облучения. Смертельные исходы не регистрируются. Лучевая болезнь средней тяжести возникает при дозе 2,5...4,0 Гр.

Почти у всех облученных в первые сутки наблюдаются тошнота, рвота, резко снижается содержание лейкоцитов в крови, появляются подкожные кровоизлияния, в 20 % случаев возможен смертельный исход, смерть наступает через 2...6 недель после облучения. При дозе 4,0...6,0 Гр развивается тяжелая форма лучевой болезни, приводящая в 50 % случаев к смерти в течение первого месяца. При дозах, превышающих 6,0 Гр, развивается крайне тяжелая форма лучевой болезни, которая почти в 100 % случаев заканчивается смертью вследствие кровоизлияния или инфекционных заболеваний. Приведенные данные относятся к случаям, когда отсутствует лечение. В настоящее время имеется ряд противолучевых средств, которые при комплексном лечении позволяют исключить летальный исход при дозах около 10 Гр.

Хроническая лучевая болезнь может развиваться при непрерывном или повторяющемся облучении в дозах, существенно ниже тех, которые вызывают острую форму. Наиболее характерными признаками хронической лучевой болезни являются изменения в крови, ряд симптомов со стороны

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

нервной системы, локальные поражения кожи, поражения хрусталика, пневмосклероз (при ингаляции плутония-239), снижение иммунореактивности организма.

Степень воздействия радиации зависит от того, является облучение внешним или внутренним (при попадании радиоактивного изотопа внутрь организма). Внутреннее облучение возможно при вдыхании, заглатывании радиоизотопов и проникновении их в организм через кожу. Некоторые вещества поглощаются и накапливаются в конкретных органах, что приводит к высоким локальным дозам радиации. Кальций, радий, стронций и другие накапливаются в костях, изотопы иода вызывают повреждение щитовидной железы, редкоземельные элементы — преимущественно опухоли печени. Равномерно распределяются изотопы цезия, рубидия, вызывая угнетение кроветворения, атрофию семенников, опухоли мягких тканей. При внутреннем облучении наиболее опасны альфа-излучающие изотопы полония и плутония. Способность вызывать отдаленные последствия — лейкозы, зло-качественные новообразования, раннее старение — одно из коварных свойств ионизирующего излучения.

Гигиеническая регламентация ионизирующего излучения осуществляется Нормами радиационной безопасности НРБ—99 (Санитарными правилами СП 2.6.1.758—99). Основные дозовые пределы облучения и допустимые уровни устанавливаются для следующих категорий облучаемых лиц:

- персонал — лица, работающие с техногенными источниками (группа А) или находящиеся по условиям работы в сфере их воздействия (группа Б);
- все население, включая лиц из персонала, вне сферы и условий их производственной деятельности.

Для категорий облучаемых лиц устанавливают три класса нормативов: основные пределы доз (ПД), допустимые уровни, соответствующие основным пределам доз, и контрольные уровни.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

1.11 Заключение по БЖД

В подразделах 2.1 и 2.2 рассматривается важность соблюдения мер безопасности при работе с электричеством и влияние ионизирующего излучения на организм человека.

Электрический ток, проходя через человеческое тело, может вызвать различные виды повреждений, такие как ожоги, металлизацию кожи и нарушение внутренних биологических процессов. Для обеспечения безопасности работы с электрическим оборудованием следует соблюдать допустимые предельные значения напряжения и тока, установленные в соответствующих нормах и стандартах.

Излучение вызывает цепочку обратимых и необратимых изменений, которые могут привести к нарушению биохимических процессов в организме, замедлению и прекращению роста тканей, а также появлению новых химических соединений, несвойственных организму. Степень поражения зависит от вида облучения (внешнего или внутреннего), дозы и продолжительности воздействия. Для обеспечения безопасности людей, работающих с техногенными источниками, установлены нормы радиационной безопасности, включающие основные пределы доз и допустимые уровни, соответствующие основным пределам доз. Важно соблюдать эти нормы, чтобы минимизировать риски для здоровья и жизни

Компьютеры и другие электронные устройства излучают электромагнитное излучение (ЭМИ), которое является формой неионизирующего излучения. Это излучение может быть вредным для здоровья, если оно превышает допустимые нормы экспозиции и длительности. Обычно уровень излучения от компьютеров и других электронных устройств настолько низок, что не представляет риска для здоровья человека. Тем не менее, при работе за компьютером имеет смысл соблюдать правила эргономики, чтобы избежать утомления глаз и

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

позвоночника. Кроме того, стоит периодически делать перерывы, чтобы размяться и потянуться.

Заключение

В результате выполнения выпускной квалификационной работы был разработано ПО интернет-магазина книг «Book58», позволяющий осуществлять продажи электронных изданий в сети Интернет. Проведен анализ других интернет-магазинов, сравнение их особенностей и постановка задач на разработку собственного магазина. ПО обеспечивает возможность регистрации и авторизации пользователей, изучения подробной информации о той или иной книге, совершения поиска книг на сайте по названию, автору и цене, а также оформления заказа и просмотр приобретенных книг. Разработка на базе *Spring* достаточно сложная задача и требовала глубоких знаний и опыта в области разработки веб-приложений.

В результате разработки интернет-магазина можно сделать следующие выводы, что комбинация *Spring Boot*, *FreeMarker Template* и *PostgreSQL* позволила быстро разработать интернет-магазин, который может удовлетворить потребности большинства клиентов, а также иметь высокую производительность и надежность.

В целом, архитектура получилась достаточно гибкой и масштабируемой, что легко позволит вносить изменения и добавлять новые функциональные возможности.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Сайт магазина «Читай город» – <https://www.chitai-gorod.ru/>
2. Сайт магазина «Лексикон» – <https://knigi58.ru/>
3. Сайт магазина «Моя книга» – <https://moya-kniga.ru/>
4. Сайт магазина «Буквоед» – <https://www.bookvoed.ru/>
5. Сайт магазина «УчКоллектор» – <http://uchkollektor.sura.ru/>
6. Официальная документация *Spring Boot*: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
7. Официальный сайт *FreeMarker*: <https://freemarker.apache.org/>
8. Официальная документация PostgreSQL: <https://www.postgresql.org/docs/>
9. Книга «*PostgreSQL: Up and Running*» от Regina Obe и Leo Hsu: <https://www.oreilly.com/library/view/postgresql-up-and/9781491963418/>
10. Видеокурс «*PostgreSQL for Beginners*» на сайте *Pluralsight*: <https://www.pluralsight.com/courses/postgresql-beginners>
11. Статья «*Creating a RESTful Web Service with Spring Boot*» на сайте *Baeldung*: <https://www.baeldung.com/spring-boot-restful-web-service-tutorial>
12. Статья «*Building Microservices with Spring Boot – Part 1: Getting Started*» на сайте *DZone*: <https://dzone.com/articles/building-microservices-with-spring-boot-part-1-gett>
13. Безопасность жизнедеятельности. /Под ред. Н.А. Белова - М.: Знание, 2000 - 364с.

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А

(Обязательное)

Программное обеспечение интернет магазина книг

Листинг программы

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Класс ApiController.java

```
package com.example.bookstore.controller;

import com.example.bookstore.domain.Book;
import com.example.bookstore.domain.User;
import com.example.bookstore.service.BookService;
import com.example.bookstore.service.OpenAIService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import javax.validation.Valid;
import java.io.File;
import java.io.IOException;
import java.util.UUID;

@Controller
@RequestMapping («/api»)
@PreAuthorize («hasAuthority('ADMIN')»)
public class ApiController {

    @Autowired
    private BookService bookService;

    @Autowired
    private OpenAIService openAIService;

    @Value («${upload.path}»)
    private String uploadPath;

    @GetMapping
    public String api(
        @RequestParam(value = «title», required = false,
            defaultValue = «») String title,
        @RequestParam(value = «description», required =
            false, defaultValue = «») String description,
```

Лист

ПГУ1.090301.08.001 ПЗ

Изм. Лист № докум. Подпись Дата

```

        @RequestParam(value = «author», required = false,
defaultValue = «») String author,
        @RequestParam(value = «price», required = false,
defaultValue = «») String price,
        @RequestParam(value = «coverImage», required =
false) MultipartFile coverImage,
        @RequestParam(value = «input-text», required =
false, defaultValue = «») String inputText,
        @RequestParam(value = «output-text», required =
false, defaultValue = «») String outputText,
        Model model
    ) {

        model.addAttribute(«title», title);
        model.addAttribute(«description», description);
        model.addAttribute(«author», author);
        model.addAttribute(«price», price);
        if (coverImage != null)
            model.addAttribute(«coverImage», coverImage);

        return «catalog/api»;
    }

    @PostMapping()
    public String add(
        @RequestParam(value = «coverImage», required =
false) MultipartFile coverImage,
        @AuthenticationPrincipal User user,
        @Valid Book book,
        BindingResult bindingResult,
        Model model,
        @RequestParam(required = false, defaultValue = «»)
String filter,
        @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
    ) throws IOException {

        //      if (bindingResult.hasErrors()) {
        //          Map<String, String> errorsMap =
ControllerUtils.getErrors(bindingResult);
        //
        //          model.mergeAttributes(errorsMap);
        //          model.addAttribute(«message», message);
        //      } else {
            if (coverImage != null &&
!coverImage.getOriginalFilename().isEmpty()) {
                File uploadDir = new File(uploadPath);

                if (!uploadDir.exists()) {
                    uploadDir.mkdir();
                }
            }
        }
    }

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        String uuidFile = UUID.randomUUID().toString();
        String resultFilename = uuidFile + «.» +
coverImage.getOriginalFilename();

        coverImage.transferTo(new File(uploadPath + «/» +
resultFilename));

        book.setFilename(resultFilename);
    }

    bookService.createBook(book);

    model.addAttribute(«temp», ««);
    model.addAttribute(«output», ««);

    return «catalog/generate»;
}

@PostMapping(«/generate»)
public String generate(
    @RequestParam(value = «input-text», required =
false, defaultValue = ««) String inputText,
    @RequestParam(value = «temp-text», required = false,
defaultValue = ««) String tempText,
    @RequestParam(value = «output-text», required =
false, defaultValue = ««) String outputText,
    @RequestParam(value = «action») String action,
    Model model
) throws RuntimeException {

    if (action.equals(«create»)) {
        tempText = openAIService.generateText(inputText);
    } else if (action.equals(«add»)) {
        outputText = outputText + tempText;
        tempText = ««;
    } else if (action.equals(«save»)) {
        bookService.saveBook(outputText);
        return «redirect:/api»;
    } else {
        throw new RuntimeException(«Unknown command» +
action);
    }

    model.addAttribute(«temp», tempText);
    model.addAttribute(«output», outputText);

    return «catalog/generate»;
}
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Класс ControllerUtils.java

```
package com.example.bookstore.controller;

import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;

import java.util.Map;
import java.util.stream.Collectors;

public class ControllerUtils {
    static Map<String, String> getErrors(BindingResult bindingResult) {
        Collector<FieldError, ?, Map<String, String>> collector
        = Collectors.toMap(
            fieldError -> fieldError.getField() + «Error»,
            FieldError::getDefaultMessage
        );
        return
        bindingResult.getFieldErrors().stream().collect(collector);
    }
}
```

Класс OrderController.java

```
package com.example.bookstore.controller;

import com.example.bookstore.domain.*;
import com.example.bookstore.domain.dto.BookDto;
import com.example.bookstore.domain.dto.OrderDto;
import com.example.bookstore.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.security.access.prepost.PreAuthorize;
import
org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class OrderController {

    @Autowired
    private BookService bookService;
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Value («${upload.path}»)
private String uploadPath;

@GetMapping («/user/orders»)
public String getOrders(
    @AuthenticationPrincipal User currentUser,
    Model model,
    @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
) {
    Page<BookDto> pageIssued =
bookService.getIssuedForUser(pageable, currentUser);
    Page<BookDto> pageFormed =
bookService.getFormedForUser(pageable, currentUser);
    Page<BookDto> pageClose =
bookService.getCloseOrderForUser(pageable, currentUser);

    model.addAttribute («userChannel», currentUser);
    model.addAttribute («pageIssued», pageIssued);
    model.addAttribute («pageFormed», pageFormed);
    model.addAttribute («pageClose», pageClose);

    return «orders/userOrder»;
}

@GetMapping («/user/books»)
public String getBooks(
    @AuthenticationPrincipal User currentUser,
    Model model,
    @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
) {
    Page<BookDto> page =
bookService.getIssuedForUser(pageable, currentUser);

    model.addAttribute («userChannel», currentUser);
    model.addAttribute («page», page);
    model.addAttribute («url», «/user/books»);
    return «orders/userBooks»;
}

@PreAuthorize («hasAuthority('ADMIN')»)
@GetMapping («/orders»)
public String userEditForm(
    @AuthenticationPrincipal User currentUser,
    Model model,
    @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
) {

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        Page<OrderDto> page = bookService.getOrders(pageable);

        model.addAttribute(«userChannel», currentUser);
        model.addAttribute(«roles», Role.values());
        model.addAttribute(«page», page);

        return «orders/orders»;
    }

    @PreAuthorize(«hasAuthority('ADMIN')»)
    @PostMapping(«/orders»)
    public String saveStatus(
        @AuthenticationPrincipal User currentUser,
        @RequestParam(«status») String selectedSwitch,
        @RequestParam(«order_id») Long order_id,
        Model model
    ) throws Exception {

        if (selectedSwitch.equals(«formed»)) {
            bookService.update(order_id, Status.FORMED);
        } else if (selectedSwitch.equals(«issued»)) {
            bookService.update(order_id, Status.ISSUED);
        } else if (selectedSwitch.equals(«closed»)) {
            bookService.update(order_id, Status.CLOSED);
        } else if (selectedSwitch.equals(«unknown»)) {
            bookService.update(order_id, Status.UNKNOWN);
        } else {
            // Обработка ситуации, когда ни один из
            переключателей не выбран
            throw new Exception(«No select switch status»);
        }
        return «redirect:/orders»;
    }
}

```

Класс ProductController.java

```

package com.example.bookstore.controller;

import com.example.bookstore.domain.Book;
import com.example.bookstore.domain.Order;
import com.example.bookstore.domain.User;
import com.example.bookstore.domain.dto.BookContentDto;
import com.example.bookstore.domain.dto.BookDto;
import com.example.bookstore.domain.dto.FilterForm;
import com.example.bookstore.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

import org.springframework.data.web.PageableDefault;
import org.springframework.security.access.prepost.PreAuthorize;
import
org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import
org.springframework.web.servlet.mvc.support.RedirectAttributes;
import org.springframework.web.util.UriComponents;
import org.springframework.web.util.UriComponentsBuilder;

import javax.validation.Valid;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;

@Controller
public class ProductController {

    @Autowired
    private BookService bookService;

    @Value («${upload.path}»)
    private String uploadPath;

    @GetMapping («/»)
    public String greeting(Map<String, Object> model) {
        return «redirect:/main»;
    }

    @GetMapping («/main»)
    public String main(
        @ModelAttribute FilterForm filterForm,
        @RequestParam(name = «button_filter», required =
false) String button_filter,
        Model model,
        @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable,
        @AuthenticationPrincipal User user
    ) {

        boolean check = bookService.isViewAllBooks();
        if (filterForm.getMyCheckBox() != null) {
            check = filterForm.getMyCheckBox();
        }

        Page<BookDto> books;
        if (check) {

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        books = bookService.messageList (pageable,
filterForm, user);
    } else {
        books =
bookService.messageListOnlyAvailable (pageable, filterForm,
user);
    }
    check = bookService.isViewAllBooks ();

    model.addAttribute («books», books);
    model.addAttribute («checkBox», check);
    if (user != null) {
        model.addAttribute («user», user);
    }
    model.addAttribute («url», «/main»);
    model.addAttribute («filter», filterForm);
    model.addAttribute («bookFilter»,
filterForm.getBookFilter () != null ? filterForm.getBookFilter ()
: «»);
    model.addAttribute («authorFilter»,
filterForm.getAuthorFilter () != null ?
filterForm.getAuthorFilter () : «»);
    model.addAttribute («priceFilterFrom»,
filterForm.getStringPriceFilterFrom ());
    model.addAttribute («priceFilterTo»,
filterForm.getStringPriceFilterTo ());

    return «/catalog/main»;
}

@GetMapping («/basket»)
public String getBasket (
    @AuthenticationPrincipal User currentUser,
    Model model,
    @PageableDefault (sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
) {
    Page<BookDto> page =
bookService.getBasketForUser (pageable, currentUser);

    int totalPrice = 0;
    for (BookDto obj : page) {
        totalPrice += obj.getPrice ().intValue ();
    }

    model.addAttribute («userChannel», currentUser);
    model.addAttribute («page», page);
    model.addAttribute («totalPrice», totalPrice);
    model.addAttribute («url», «/user-messages/» +
currentUser.getId ());

    return «orders/basket»;
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }

    @GetMapping («/basket/checkout»)
    public String checkout(
        @AuthenticationPrincipal User currentUser,
        Model model,
        @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
    ) {
        Page<BookDto> page =
bookService.getOrderForUser(pageable, currentUser);

        int totalPrice = 0;
        for (BookDto obj : page) {
            totalPrice += obj.getPrice().intValue();
        }

        model.addAttribute («userChannel», currentUser);
        model.addAttribute («subscriptionsCount», 0);
        model.addAttribute («subscribersCount», 0);
        model.addAttribute («isSubscriber», 0);
        model.addAttribute («page», page);
        model.addAttribute («isCurrentUser»,
currentUser.equals(currentUser));
        model.addAttribute («order», 0);
        model.addAttribute («totalPrice», totalPrice);
        model.addAttribute («url», «/user-messages/» +
currentUser.getId());

        return «orders/checkout»;
    }

    @GetMapping («/basket/payment»)
    public String paymentPage(
        @AuthenticationPrincipal User currentUser,
        Model model
    ) {
        model.addAttribute («userChannel», currentUser);
        return «orders/payment»;
    }

    @PostMapping («/basket/payment»)
    public String checkout(
        @AuthenticationPrincipal User currentUser,
        Model model
    ) {
        bookService.confirmOrder(currentUser);
        model.addAttribute («userChannel», currentUser);
        return «orders/confirmation»;
    }

    @GetMapping («/user/book»)

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

    public String getBook(
        @RequestParam(«bookId») Long book_id,
        @RequestParam(value = «page», required = false,
defaultValue = «1») Long page,
        @AuthenticationPrincipal User currentUser,
        Model model,
        @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
    ) {
        BookContentDto bookContentDTO =
bookService.getIssuedForUser(currentUser, book_id);
        Book book = bookService.getProductById(book_id);
        int PAGE_SIZE = 5000;
        int textSize =
bookContentDTO.getBookContent().getContent().length();
        String content = «»;
        if (textSize > PAGE_SIZE) {
            if (page != null) {
                if (textSize > (int) (page * PAGE_SIZE))
                    content =
bookContentDTO.getBookContent().getContent().substring((int)
((page - 1) * PAGE_SIZE), (int) (page * PAGE_SIZE));
                else
                    content =
bookContentDTO.getBookContent().getContent().substring((int)
((page - 1) * PAGE_SIZE), textSize);
            } else
                content =
bookContentDTO.getBookContent().getContent().substring(0,
PAGE_SIZE);
        } else {
            content =
bookContentDTO.getBookContent().getContent();
        }

        int currentPage = Math.toIntExact(page);

        int totalPages = (int) Math.ceil((double) textSize /
PAGE_SIZE); // Общее количество страниц

        Set<Integer> pg = new TreeSet<>();
        pg.add(currentPage);
        pg.add(totalPages);
        pg.add(1);
        pg.add(Math.min(totalPages, currentPage + 2));
        pg.add(Math.min(totalPages, currentPage + 1));
        pg.add(Math.max(1, currentPage - 1));
        pg.add(Math.max(1, currentPage - 2));

        model.addAttribute(«userChannel», currentUser);
        model.addAttribute(«book», book);
        model.addAttribute(«bookContent», bookContentDTO);
    }

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        model.addAttribute(«content», content);
        model.addAttribute(«pages», pg);
        model.addAttribute(«currentPage», page);
        model.addAttribute(«pageSize»,
String.valueOf(PAGE_SIZE));
        model.addAttribute(«size»,
String.valueOf(bookContentDTO.getBookContent().getContent().length()));
        return «orders/viewer»;
    }

    @GetMapping(«/product/{product_id}»)
    public String getProductInfo(
        @AuthenticationPrincipal User currentUser,
        @PathVariable Long product_id,
        Model model,
        @PageableDefault(sort = {«id»}, direction =
Sort.Direction.DESC) Pageable pageable
    ) {
        Book book = bookService.getProductById(product_id);
        Order order =
bookService.getOrderByUserProduct(currentUser, book);
        if (book != null) {
            model.addAttribute(«book», book);
            model.addAttribute(«bought», order != null);
            return «catalog/product»;
        } else
            return «redirect:/main»;
    }

    @PreAuthorize(«hasAuthority('ADMIN')»)
    @PostMapping(«/product/{product_id}/edit»)
    public String editProduct(
        @AuthenticationPrincipal User currentUser,
        @Valid Book book,
        @RequestParam(value = «action») String action,
        RedirectAttributes redirectAttributes,
        @RequestHeader(required = false) String referer
    ) {

        if (action.equals(«update»)) {
            bookService.update(book);
        } else if (action.equals(«delete»)) {
            bookService.delete(book);
            return «redirect:/main»;
        } else {
            throw new RuntimeException(«Unknown command» +
action);
        }
    }

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        UriComponents components =
UriComponentsBuilder.fromHttpRequest(referer).build();

        components.getQueryParams()
            .entrySet()
            .forEach(pair ->
redirectAttributes.addAttribute(pair.getKey(),
pair.getValue()));

        return «redirect:» + components.getPath();
    }

@GetMapping(«/product/{product_id}/buy»)
public String getProductInfo(
    @AuthenticationPrincipal User currentUser,
    @PathVariable Long product_id,
    RedirectAttributes redirectAttributes,
    @RequestHeader(required = false) String referer
) {

    bookService.addOrDeleteIfExists(currentUser, product_id);

    UriComponents components =
UriComponentsBuilder.fromHttpRequest(referer).build();

    components.getQueryParams()
        .entrySet()
        .forEach(pair ->
redirectAttributes.addAttribute(pair.getKey(),
pair.getValue()));

    return «redirect:» + components.getPath();
}
}

```

Класс RegistrationController.java

```

package com.example.bookstore.controller;

import com.example.bookstore.domain.User;
import com.example.bookstore.domain.dto.CaptchaResponseDto;
import com.example.bookstore.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

import org.springframework.web.client.RestTemplate;

import javax.validation.Valid;
import java.util.Collections;
import java.util.Map;

@Controller
public class RegistrationController {
    private final static String CAPTCHA_URL =
«https://www.google.com/recaptcha/api/siteverify?secret=%s&respo
nse=%s»;

    @Autowired
    private UserService userService;

    @Value («${recaptcha.secret}»)
    private String secret;

    @Autowired
    private RestTemplate restTemplate;

    @GetMapping («/registration»)
    public String registration() {
        return «auth/registration»;
    }

    @PostMapping («/registration»)
    public String addUser(
        @RequestParam («password2») String passwordConfirm,
        @RequestParam («g-recaptcha-response») String
captchaResponse,
        @Valid User user,
        BindingResult bindingResult,
        Model model
    ) {
        String url = String.format (CAPTCHA_URL, secret,
captchaResponse);
        CaptchaResponseDto response =
restTemplate.postForObject (url, Collections.emptyList(),
CaptchaResponseDto.class);

        if (!response.isSuccess()) {
            model.addAttribute («captchaError», «Fill captcha»);
        }

        boolean isConfirmEmpty =
StringUtils.isEmpty (passwordConfirm);

        if (isConfirmEmpty) {
            model.addAttribute («password2Error», «Password
confirmation cannot be empty»);
        }
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (user.getPassword() != null &&
!user.getPassword().equals(passwordConfirm)) {
            model.addAttribute(«passwordError», «Passwords are
different!»);
        }

        if (isConfirmEmpty || bindingResult.hasErrors() ||
!response.isSuccess()) {
            Map<String, String> errors =
ControllerUtils.getErrors(bindingResult);

            model.mergeAttributes(errors);

            return «auth/registration»;
        }

        if (!userService.addUser(user)) {
            model.addAttribute(«usernameError», «User exists!»);
            return «login/registration»;
        }

        return «redirect:/login»;
    }

    @GetMapping(«/activate/{code}»)
    public String activate(Model model, @PathVariable String
code) {
        boolean isActivated = userService.activateUser(code);

        if (isActivated) {
            model.addAttribute(«messageType», «success»);
            model.addAttribute(«message», «User successfully
activated»);
        } else {
            model.addAttribute(«messageType», «danger»);
            model.addAttribute(«message», «Activation code is
not found!»);
        }

        return «login»;
    }
}

```

Класс UserController.java

```

package com.example.bookstore.controller;

import com.example.bookstore.domain.Role;
import com.example.bookstore.domain.User;
import com.example.bookstore.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

import org.springframework.security.access.prepost.PreAuthorize;
import
org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.Map;

@Controller
@RequestMapping(«/user»)
public class UserController {
    @Autowired
    private UserService userService;

    @PreAuthorize(«hasAuthority('ADMIN')»)
    @GetMapping
    public String userList(Model model) {
        model.addAttribute(«users», userService.findAll());

        return «manage/userList»;
    }

    @PreAuthorize(«hasAuthority('ADMIN')»)
    @GetMapping(«{user}»)
    public String userEditForm(@PathVariable User user, Model
model) {
        model.addAttribute(«user», user);
        model.addAttribute(«roles», Role.values());
        model.addAttribute(«orders», user.getOrders());

        return «manage/editUser»;
    }

    @PreAuthorize(«hasAuthority('ADMIN')»)
    @PostMapping
    public String userSave(
        @RequestParam String username,
        @RequestParam Map<String, String> form,
        @RequestParam(«userId») User user
    ) {
        userService.saveUser(user, username, form);

        return «redirect:/user»;
    }

    @GetMapping(«profile»)
    public String getProfile(Model model,
@AuthenticationPrincipal User user) {
        model.addAttribute(«username», user.getUsername());
        model.addAttribute(«email», user.getEmail());
    }

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return «manage/profile»;
    }

    @PostMapping(«profile»)
    public String updateProfile(
        @AuthenticationPrincipal User user,
        @RequestParam String password,
        @RequestParam String email
    ) {
        userService.updateProfile(user, password, email);

        return «redirect:/user/profile»;
    }
}

```

Класс CompletionChoice.java

```

package com.example.bookstore.domain.api;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.Data;

@Data
public class CompletionChoice {
    public CompletionChoice() {
    }

    @JsonProperty(«message»)
    private Message message;

    @JsonProperty(«finish_reason»)
    private String finishReason;

    @JsonProperty(«index»)
    private int index;
}

```

Класс Generator.java

```

package com.example.bookstore.domain.api;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.Data;

@Data
public class Generator {
    public Generator() {
    }

    @JsonProperty(«id»)
    private String id;
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

@JsonProperty(«object»)
private String object;

@JsonProperty(«created»)
private int created;

@JsonProperty(«model»)
private String model;

@JsonProperty(«choices»)
private CompletionChoice[] choices;

@JsonProperty(«usage»)
private Usage usage;
}

```

Класс Message.java

```

package com.example.bookstore.domain.api;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.Data;

@Data
public class Message {
    @JsonProperty(«role»)
    private String role;

    @JsonProperty(«content»)
    private String content;
}

```

Класс Usage.java

```

package com.example.bookstore.domain.api;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.Data;

@Data
public class Usage {
    public Usage() {
    }

    @JsonProperty(«prompt_tokens»)
    private int promptTokens;

    @JsonProperty(«completion_tokens»)
    private int completion_tokens;
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

    @JsonProperty(«total_tokens»)
    private int totalTokens;
}

```

Класс BookContentDto.java

```

package com.example.bookstore.domain.dto;

import com.example.bookstore.domain.BookContent;
import com.example.bookstore.domain.Book;
import lombok.Data;

@Data
public class BookContentDto {

    private Book book;

    private BookContent bookContent;

    public BookContentDto(Book book, BookContent bookContent) {
        this.book = book;
        this.bookContent = bookContent;
    }
}

```

Класс BookDto.java

```

package com.example.bookstore.domain.dto;

import com.example.bookstore.domain.Book;
import com.example.bookstore.domain.Status;
import lombok.Data;

import java.util.Date;

@Data
public class BookDto {
    private Long id;
    private String title;
    private String description;
    private String author;
    private Long price;
    private String filename;

    private Status status;
    private Date date;

    public BookDto(Book book) {
        this.id = book.getId();
        this.title = book.getTitle();
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        this.description = book.getDescription();
        this.author = book.getAuthor();
        this.price = book.getPrice();
        this.filename = book.getFilename();
        this.status = Status.UNKNOWN;
        this.date = null;
    }

    public BookDto(Book book, Status status, Date date, boolean
ignoreOrder) {
        this.id = book.getId();
        this.title = book.getTitle();
        this.description = book.getDescription();
        this.author = book.getAuthor();
        this.price = book.getPrice();
        this.filename = book.getFilename();
        if (ignoreOrder) {
            this.status = Status.UNKNOWN;
            this.date = null;
        } else {
            this.status = status;
            this.date = date;
        }
    }

    public BookDto(Book book, Status status, Date date) {
        this.id = book.getId();
        this.title = book.getTitle();
        this.description = book.getDescription();
        this.author = book.getAuthor();
        this.price = book.getPrice();
        this.filename = book.getFilename();
        this.status = status;
        this.date = date;
    }
}

```

Класс FilterForm.java

```

package com.example.bookstore.domain.dto;

import lombok.Data;

@Data
public class FilterForm {

    private String bookFilter;
    private String authorFilter;
    private Long priceFilterFrom;
    private Long priceFilterTo;
    private Boolean myCheckBox;
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

    public FilterForm(String bookFilter, String authorFilter,
String priceFilterFrom, String priceFilterTo, String myCheckBox)
{
    this.bookFilter = bookFilter;
    this.authorFilter = authorFilter;

    if (priceFilterFrom != null &&
!priceFilterFrom.isEmpty()) {
        priceFilterFrom = removeScape(priceFilterFrom);
        this.priceFilterFrom =
Long.valueOf(priceFilterFrom);
    } else {
        this.priceFilterFrom = null;
    }
    if (priceFilterTo != null && !priceFilterTo.isEmpty()) {
        priceFilterTo = removeScape(priceFilterTo);
        this.priceFilterTo = Long.valueOf(priceFilterTo);
    } else {
        this.priceFilterTo = null;
    }

    if (myCheckBox != null && !myCheckBox.isEmpty()) {
        this.myCheckBox = Boolean.valueOf(myCheckBox);
    } else {
        this.myCheckBox = null;
    }
}

    public String removeScape(String str)
    {
        return str.replaceAll("<<\\u00a0>>", "<<>").replaceAll("<< <<,"
<<>);
    }

    public boolean isEmpty() {
        if (bookFilter != null && !bookFilter.isEmpty()) {
            return false;
        }
        if (authorFilter != null && !authorFilter.isEmpty()) {
            return false;
        }
        if (priceFilterFrom != null) {
            return false;
        }
        if (priceFilterTo != null) {
            return false;
        }
        return true;
    }

    public String getBookFilter() {

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (bookFilter != null)
            return bookFilter.toLowerCase();
        return null;
    }

    public String getAuthorFilter() {
        if (authorFilter != null)
            return authorFilter.toLowerCase();
        return null;
    }

    public String getStringPriceFilterFrom() {
        if (priceFilterFrom != null)
            return removeScape(String.valueOf(priceFilterFrom));
        return ««;
    }

    public String getStringPriceFilterTo() {
        if (priceFilterTo != null)
            return removeScape(String.valueOf(priceFilterTo));
        return ««;
    }
}

```

Класс OrderDto.java

```

package com.example.bookstore.domain.dto;

import com.example.bookstore.domain.Book;
import com.example.bookstore.domain.Order;
import com.example.bookstore.domain.Status;
import com.example.bookstore.domain.User;
import lombok.Data;

import java.util.Date;

@Data
public class OrderDto {
    private Long id;
    private User author;
    private Book book;
    private Status status;
    private Date date;

    public OrderDto(Order order) {
        this.id = order.getId();
        this.book = order.getBook();
        this.author = order.getAuthor();
        this.status = order.getStatus();
        this.date = order.getDateOrder();
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Класс Book.java

```
package com.example.bookstore.domain;

import lombok.Data;
import org.hibernate.validator.constraints.Length;

import javax.persistence.*;
import javax.validation.constraints.NotBlank;

@Entity
@Data
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column
    @NotBlank(message = «Please fill the message»)
    @Length(max = 255, message = «Message too long (more than
255)»)
    private String title;

    @Column
    @Length(max = 2048, message = «Message too long (more than
2kB)»)
    private String description;

    @Column
    @Length(max = 255, message = «Message too long (more than
255)»)
    private String author;

    @Column
    private Long price;

    @Column
    private String filename;

    @OneToOne(mappedBy = «book», cascade = CascadeType.ALL,
orphanRemoval = true)
    private BookContent bookContent;

    public Book() {
    }

    public Book(String title, String description, String author,
Long price, String filename) {
        this.title = title;
        this.description = description;
        this.author = author;
    }
}
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        this.price = price;
        this.filename = filename;
    }

    public void set(Book book) {
        this.title = book.getTitle();
        this.description = book.getDescription();
        this.author = book.getAuthor();
        this.price = book.getPrice();
        this.filename = book.getFilename();
    }
}

```

Класс BookContent.java

```

package com.example.bookstore.domain;

import lombok.Data;

import javax.persistence.*;
import java.io.Serializable;

@Data
@Entity
@Table(name = «book_content»)
public class BookContent{

    @Id
    private Long book_id;

    @OneToOne(fetch = FetchType.LAZY)
    @MapsId
    private Book book;

    @Column(name = «content»)
    private String content;

    public BookContent() {
    }

    public BookContent(Long book_id, Book book, String content)
    {
        this.book_id = book_id;
        this.book = book;
        this.content = content;
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Класс Order.java

```
package com.example.bookstore.domain;

import lombok.Data;

import javax.persistence.*;
import java.util.Date;

@Data
@Entity
@Table(name = «orders»)
public class Order {
    public Order() {
    }

    public Order(User user, Book book, Date dateOrder) {
        this.author = user;
        this.book = book;
        this.status = Status.BASKET;
        this.date_order = dateOrder;
    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = «user_id»)
    private User author;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = «book_id»)
    private Book book;

    @Enumerated(EnumType.STRING)
    private Status status;

    @Temporal(TemporalType.TIMESTAMP)
    private Date date_order;
}
```

Класс Role.java

```
package com.example.bookstore.domain;

import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority {
    USER, ADMIN;

    @Override
    public String getAuthority() {
    }
}
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return name();
    }
}

```

Класс Status.java

```

package com.example.bookstore.domain;

public enum Status {
    UNKNOWN,
    BASKET,
    FORMED,
    ISSUED,
    CLOSED;

    public String get() {
        switch (this) {
            case BASKET:
                return «В корзине»;
            case FORMED:
                return «В обработке»;
            case ISSUED:
                return «Готов»;
            case CLOSED:
                return «Закрыт»;
        }
        return «Неизвестный»;
    }

    public boolean isUnknown() {
        return this.equals(UNKNOWN);
    }
    public boolean isBasket() {
        return this.equals(BASKET);
    }
    public boolean isFormed() {
        return this.equals(FORMED);
    }
    public boolean isIssued() {
        return this.equals(ISSUED);
    }
    public boolean isClosed() {
        return this.equals(CLOSED);
    }
}

```

Класс User.java

```

package com.example.bookstore.domain;

import lombok.Data;
import org.springframework.security.core.GrantedAuthority;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

import
org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import java.util.Collection;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

@Data
@Entity
@Table(name = «usr»)
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @NotBlank(message = «Username cannot be empty»)
    private String username;
    @NotBlank(message = «Password cannot be empty»)
    private String password;
    private boolean active;

    @Email(message = «Email is not correct»)
    @NotBlank(message = «Email cannot be empty»)
    private String email;
    private String activationCode;

    @ElementCollection(targetClass = Role.class, fetch =
FetchType.EAGER)
    @CollectionTable(name = «user_role», joinColumns =
@JoinColumn(name = «user_id»))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;

    @OneToMany(mappedBy = «author», cascade = CascadeType.ALL,
fetch = FetchType.LAZY)
    private Set<Book> books;

    @OneToMany(mappedBy = «author», cascade = CascadeType.ALL)
    private Set<Order> orders = new HashSet<>();

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        User user = (User) o;
        return Objects.equals(id, user.id);
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Override
public int hashCode() {

    return Objects.hash(id);
}

public boolean isAdmin() {
    return roles.contains(Role.ADMIN);
}

public String getUsername() {
    return username;
}

@Override
public boolean isAccountNonExpired() {
    return true;
}

@Override
public boolean isAccountNonLocked() {
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    return true;
}

@Override
public boolean isEnabled() {
    return isActive();
}

public void setUsername(String username) {
    this.username = username;
}

@Override
public Collection<? extends GrantedAuthority>
getAuthorities() {
    return getRoles();
}
}

```

Класс BookContentRepo.java

```

package com.example.bookstore.repos;

import com.example.bookstore.domain.BookContent;
import org.springframework.data.repository.CrudRepository;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```
public          interface          BookContentRepo          extends
CrudRepository<BookContent, Long> {
}
```

Класс BookRepo.java

```
package com.example.bookstore.repos;

import com.example.bookstore.domain.Book;
import com.example.bookstore.domain.Status;
import com.example.bookstore.domain.User;
import com.example.bookstore.domain.dto.BookDto;
import com.example.bookstore.domain.dto.FilterForm;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

public interface BookRepo extends CrudRepository<Book, Long> {

    @Query(«select                                                                                               new
com.example.bookstore.domain.dto.BookDto(m,                                                                 o.status,
o.date_order) « +
        «from Book m Left JOIN Order as o « +
        «ON m.id = o.book.id and o.author = :user « +
        «group by m, o.status, o.date_order»)
    Page<BookDto> findAll(Pageable pageable, @Param(«user») User
user);

    @Query(«select                                                                                               new
com.example.bookstore.domain.dto.BookDto(m) « +
        «from Book m « +
        «group by m»)
    Page<BookDto> findAll(Pageable pageable);

    @Query(«select                                                                                               new
com.example.bookstore.domain.dto.BookDto(b,                                                                 o.status,
o.date_order) « +
        «from Book b Left JOIN Order as o « +
        «ON b.id = o.book.id and o.author = :user « +
        «where      (:#{#filter.bookFilter}      is      null      or
lower(b.title) like %:#{#filter.bookFilter}%) « +
        «and      (:#{#filter.authorFilter}      is      null      or
lower(b.author) like %:#{#filter.authorFilter}%) « +
        «and      (:#{#filter.priceFilterFrom} is null or b.price
>= :#{#filter.priceFilterFrom}) « +
        «and      (:#{#filter.priceFilterTo} is null or b.price
<= :#{#filter.priceFilterTo}) « +
        «group by b, o.status, o.date_order»)
}
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        Page<BookDto> findByFilter(Pageable pageable, @Param(«user»)
User user,
                                @Param(«filter»)          FilterForm
filter);

```

```

        @Query(«select
com.example.bookstore.domain.dto.BookDto(b,
o.date_order) « +
                                new
                                o.status,
                                «from Book b Left JOIN Order as o « +
                                «ON b.id = o.book.id « +
                                «where (:#{#filter.bookFilter} is null or
lower(b.title) like %:#{#filter.bookFilter}%) « +
                                «and (:#{#filter.authorFilter} is null or
lower(b.author) like %:#{#filter.authorFilter}%) « +
                                «and (:#{#filter.priceFilterFrom} is null or b.price
>= :#{#filter.priceFilterFrom}) « +
                                «and (:#{#filter.priceFilterTo} is null or b.price
<= :#{#filter.priceFilterTo}) « +
                                «group by b, o.status, o.date_order»)
        Page<BookDto>          findByFilter(Pageable pageable,
@Param(«filter») FilterForm filter);

```

```

        @Query(«select
com.example.bookstore.domain.dto.BookDto(m,
o.date_order) « +
                                new
                                o.status,
                                «from Book m left join Order as o « +
                                «ON m.id = o.book.id « +
                                «where m.id = o.book.id and o.author = :user « +
                                «group by m, o.status, o.date_order»)
        Page<BookDto>          findByUser(Pageable pageable, @Param(«user»)
User user);

```

```

        Page<Book> findById(Pageable pageable, Long product_id);

```

```

        @Query(«select
com.example.bookstore.domain.dto.BookDto(b,
o.date_order) « +
                                new
                                o.status,
                                «from Book b left join Order as o « +
                                «ON b.id = o.book.id « +
                                «where o IS NULL or o.author = :user and o.status =
:status « +
                                «group by b, o.status, o.date_order»)
        Page<BookDto>          findForUserByStatus(Pageable pageable,
@Param(«user») User currentUser, @Param(«status») Status
status);

```

```

        @Query(«SELECT
com.example.bookstore.domain.dto.BookDto(b, « +
                                new
                                « o.status, « +
                                «o.date_order, « +
                                «(CASE WHEN o.author = :user and (o.status = :status
or o.status = :status2) THEN false ELSE true END)) « +

```

						Лист
					ПГУ1.090301.08.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

        «FROM Book b LEFT JOIN Order AS o « +
        «ON b.id = o.book.id « +
        «WHERE o IS NULL or o.author = :user and (o.status =
:status or o.status = :status2) or o.author != :user « +
        «GROUP BY b.id, o.id»)

```

```

        Page<BookDto> findForUserByStatus(Pageable pageable,
@Param(«user») User currentUser, @Param(«status») Status status,
@Param(«status2») Status status2);

```

```

        @Query(«SELECT
com.example.bookstore.domain.dto.BookDto(b, « +
        « o.status, « +
        «o.date_order, « +
        «(CASE WHEN o.author = :user and (o.status = :status
or o.status = :status2) THEN false ELSE true END)) « +
        «FROM Book b LEFT JOIN Order AS o « +
        «ON b.id = o.book.id « +
        «WHERE (o IS NULL or o.author = :user and (o.status
= :status or o.status = :status2) or o.author != :user ) and « +
        «(:#{#filter.bookFilter} = ' ' or lower(b.title) like
%:#{#filter.bookFilter}%) « +
        «and (:#{#filter.authorFilter} = ' ' or
lower(b.author) like %:#{#filter.authorFilter}%) « +
        «and (:#{#filter.priceFilterFrom} is null or b.price
>= :#{#filter.priceFilterFrom})) « +
        «and (:#{#filter.priceFilterTo} is null or b.price
<= :#{#filter.priceFilterTo})) « +
        «GROUP BY b.id, o.id»)

```

```

        Page<BookDto> findForUserFilterByStatus(
        Pageable pageable,
        @Param(«user») User currentUser,
        @Param(«filter») FilterForm filter,
        @Param(«status») Status status,
        @Param(«status2») Status status2
);

```

```

        @Query(«SELECT
com.example.bookstore.domain.dto.BookDto(b, « +
        «o.status, « +
        «o.date_order, « +
        «false) « +
        «FROM Book b LEFT JOIN Order AS o « +
        «ON b.id = o.book.id « +
        «WHERE « +
        «(:#{#filter.bookFilter} = ' ' or lower(b.title) like
%:#{#filter.bookFilter}%) « +
        «and (:#{#filter.authorFilter} = ' ' or
lower(b.author) like %:#{#filter.authorFilter}%) « +
        «and (:#{#filter.priceFilterFrom} is null or b.price
>= :#{#filter.priceFilterFrom})) « +
        «and (:#{#filter.priceFilterTo} is null or b.price
<= :#{#filter.priceFilterTo})) « +

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        «GROUP BY b.id, o.id»)
    Page<BookDto> findForUserFilterByStatus(
        Pageable pageable,
        @Param(«filter») FilterForm filter
    );
}

```

Класс OrderRepo.java

```

package com.example.bookstore.repos;

import com.example.bookstore.domain.Book;
import com.example.bookstore.domain.Order;
import com.example.bookstore.domain.Status;
import com.example.bookstore.domain.User;
import com.example.bookstore.domain.dto.OrderDto;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

public interface OrderRepo extends CrudRepository<Order, Long> {

    @Query(«select
com.example.bookstore.domain.dto.OrderDto(o) « +
        «from Order o « +
        «group by o»)
    Page<OrderDto> findAll(Pageable pageable);

    @Query(«select
com.example.bookstore.domain.dto.OrderDto(o) « +
        «from Order o « +
        «where o.status != :status1 and o.status != :status2
« +
        «group by o»)
    Page<OrderDto> findForActiveOrder(Pageable pageable,
@Param(«status1») Status status1, @Param(«status2») Status
status2);

    @Query(«select
com.example.bookstore.domain.dto.OrderDto(o) « +
        «from Order o « +
        «where o.author = :author « +
        «group by o»)
    Page<OrderDto> findByOrder(Pageable pageable,
@Param(«author») User currentUser);

    @Query(«from Order o « +

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        «where o.author = :author AND o.book = :book « +
        «group by o»)
    Order    findByOrder(@Param(«author»)    User    currentUser,
    @Param(«book») Book book);

    @Query(«select                                new
com.example.bookstore.domain.dto.OrderDto(o) « +
        «from Order o « +
        «where o.author = :user AND o.status = :status « +
        «group by o»)
    Page<OrderDto>    findByStatus(Pageable    pageable,
    @Param(«user»)    User    currentUser,    @Param(«status»)    Status
    status);

    @Modifying
    @Query(«UPDATE Order as o « +
        «SET o.status = :newStatus « +
        «WHERE    o.author    =    :user    AND    o.status    =
:currentStatus»)
    void    setStatus(@Param(«user»)    User    currentUser,
    @Param(«currentStatus»)    Status    currentUser,
    @Param(«newStatus») Status newStatus);

    @Modifying
    @Query(«DELETE FROM Order o WHERE o.book.id = :bookId»)
    void deleteByBookId(@Param(«bookId») Long bookId);
}

```

Класс UserRepo.java

```

package com.example.bookstore.repos;

import com.example.bookstore.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepo extends JpaRepository<User, Long> {
    User findByUsername(String username);

    User findByActivationCode(String code);
}

```

Класс BookService.java

```

package com.example.bookstore.service;

import com.example.bookstore.domain.*;
import com.example.bookstore.domain.dto.BookContentDto;
import com.example.bookstore.domain.dto.BookDto;
import com.example.bookstore.domain.dto.FilterForm;
import com.example.bookstore.domain.dto.OrderDto;
import com.example.bookstore.repos.BookContentRepo;
import com.example.bookstore.repos.BookRepo;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

import com.example.bookstore.repos.OrderRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.EntityNotFoundException;
import java.sql.Timestamp;
import java.util.Date;

@Service
public class BookService {
    @Autowired
    private BookRepo bookRepo;

    @Autowired
    private OrderRepo orderRepo;

    @Autowired
    private BookContentRepo bookContentRepo;

    private Book book;

    private boolean viewAllBooks = false;

    public Page<BookDto> messageList(Pageable pageable,
    FilterForm filter, User user) {

        viewAllBooks = true;
        if (user != null) {
            if (filter != null && !filter.isEmpty()) {
                viewAllBooks = filter.getMyCheckBox() == null ||
filter.getMyCheckBox();
                return bookRepo.findByFilter(pageable, user,
filter);
            } else {
                return bookRepo.findAll(pageable, user);
            }
        } else {
            if (filter != null && !filter.isEmpty()) {
                return
bookRepo.findForUserFilterByStatus(pageable, filter);
            } else {
                return bookRepo.findAll(pageable);
            }
        }
    }

    public Page<BookDto> messageListOnlyAvailable(Pageable
pageable, FilterForm filter, User user) {

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        viewAllBooks = false;
        if (user != null) {
            if (filter != null && !filter.isEmpty()) {
                viewAllBooks = filter.getMyCheckBox() != null &&
filter.getMyCheckBox();
                return
bookRepo.findForUserFilterByStatus(pageable,      user,      filter,
Status.UNKNOWN, Status.BASKET);
            } else {
                return      bookRepo.findForUserByStatus(pageable,
user, Status.UNKNOWN, Status.BASKET);
            }
        } else {
            if (filter != null && !filter.isEmpty()) {
                return
bookRepo.findForUserFilterByStatus(pageable, filter);
            } else {
                return bookRepo.findAll(pageable);
            }
        }
    }

    public Page<BookDto>   getBasketForUser(Pageable  pageable,
User currentUser) {
        return      bookRepo.findForUserByStatus(pageable,
currentUser, Status.BASKET);
    }

    public Page<BookDto> getOrderForUser(Pageable pageable, User
currentUser) {
        return bookRepo.findByUser(pageable, currentUser);
    }

    public Page<BookDto>   getIssuedForUser(Pageable  pageable,
User currentUser) {
        return      bookRepo.findForUserByStatus(pageable,
currentUser, Status.ISSUED);
    }

    public BookContentDto   getIssuedForUser(User  currentUser,
Long bookId) {
        Book book = getProductById(bookId);

        Order order = getOrderByUserProduct(currentUser, book);
        if (order.getStatus().isIssued()) {
            return      new      BookContentDto(book,
bookContentRepo.findById(bookId).orElseThrow(()      ->      new
EntityNotFoundException(bookId.toString())));
        }
        throw new EntityNotFoundException(bookId.toString());
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

    public Page<BookDto> getFormedForUser(Pageable pageable,
User currentUser) {
        return bookRepo.findForUserByStatus(pageable,
currentUser, Status.FORMED);
    }

    public Page<BookDto> getCloseOrderForUser(Pageable pageable,
User currentUser) {
        return bookRepo.findForUserByStatus(pageable,
currentUser, Status.CLOSED);
    }

    public Page<Book> getProductById(Pageable pageable, Long
productId) {
        return bookRepo.findById(pageable, productId);
    }

    public Book getProductById(Long productId) {
        Book book = bookRepo.findById(productId).orElse(null);
        return book;
    }

    public Page<OrderDto> orderListForUser(Pageable pageable,
User currentUser) {
        return orderRepo.findByOrder(pageable, currentUser);
    }

    public void addOrDeleteIfExist(User currentUser, Book book)
{
        Order order = orderRepo.findByOrder(currentUser, book);

        if (order != null) {
            orderRepo.deleteById(order.getId());
        } else {
            Order order2 = new Order(currentUser, book, new
Timestamp(new Date().getTime()));
            orderRepo.save(order2);
        }
    }

    public void addOrDeleteIfExist(User currentUser, Long
productId) {
        Book book = bookRepo.findById(productId).orElse(null);
        if (book != null) {
            addOrDeleteIfExist(currentUser, book);
        } else {
            throw new RuntimeException(«Product by index « +
productId + « not found «);
        }
    }

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

    public boolean existByUserAndBook(User currentUser, Book
book) {
        return orderRepo.findByOrder(currentUser, book) != null;
    }

    public Order getOrderByUserProduct(User currentUser, Book
book) {
        return orderRepo.findByOrder(currentUser, book);
    }

    @Transactional
    public void confirmOrder(User currentUser) {
        orderRepo.setStatus(currentUser, Status.BASKET,
Status.FORMED);
    }

    @Transactional
    public void update(Long order_id, Status status) {
        Order order =
orderRepo.findById(order_id).orElseThrow(() -> new
EntityNotFoundException(order_id.toString()));
        order.setStatus(status);
        orderRepo.save(order);
    }

    @Transactional
    public void update(Book book) {
        Book existingBook =
bookRepo.findById(book.getId()).orElseThrow(() -> new
EntityNotFoundException(book.getId().toString()));
        existingBook.set(book);
        bookRepo.save(existingBook);
    }

    public Page<OrderDto> getOrders(Pageable pageable) {
        return orderRepo.findForActiveOrder(pageable,
Status.UNKNOWN, Status.BASKET);
    }

    @Transactional
    public void delete(Book book) {
        orderRepo.deleteByBookId(book.getId());
        bookContentRepo.deleteById(book.getId());
        bookRepo.deleteById(book.getId());
    }

    public void createBook(Book book) {
        this.book = book;
    }

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        public void saveBook(String text) {
            bookRepo.save(book);

            BookContent bookContent = new BookContent(book.getId(),
book, text);
            bookContentRepo.save(bookContent);
        }

        public boolean isViewAllBooks() {
            return viewAllBooks;
        }

        public void setViewAllBooks(boolean viewAllBooks) {
            this.viewAllBooks = viewAllBooks;
        }
    }

```

Класс MailSender.java

```

package com.example.bookstore.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;

@Service
public class MailSender {
    @Autowired
    private JavaMailSender mailSender;

    @Value («${spring.mail.username}»)
    private String username;

    public void send(String emailTo, String subject, String
message) {
        SimpleMailMessage mailMessage = new SimpleMailMessage();

        mailMessage.setFrom(username);
        mailMessage.setTo(emailTo);
        mailMessage.setSubject(subject);
        mailMessage.setText(message);

        mailSender.send(mailMessage);
    }
}

```

Класс OpenAIService.java

```

package com.example.bookstore.service;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

import com.example.bookstore.domain.api.Generator;
import com.fasterxml.jackson.databind.ObjectMapper;
import okhttp3.*;
import org.springframework.stereotype.Service;

import java.io.IOException;
import java.net.SocketTimeoutException;
import java.util.concurrent.TimeUnit;

/**
 * сервисный класс MyOpenAIService, который содержит метод
 generateText(),
 * используемый для генерации текста.
 * Метод принимает массив строк prompts,
 * которые представляют собой фразы,
 * слова или предложения, на основе которых будет генерироваться
 текст.
 */
@Service
public class OpenAIService {

    private static final String OPENAI_API_URL_2 =
«https://api.openai.com/v1/chat/completions»;
    private static final String OPENAI_API_KEY = «secret_key»;

    public String generateText(String prompt) {

        OkHttpClient client = new OkHttpClient
            .Builder()
            .connectTimeout(60, TimeUnit.SECONDS)
            .build();

        MediaType mediaType =
MediaType.parse(«application/json»);

        RequestBody body = RequestBody.create(mediaType, «{\n» +
            «    \»model\»: \»gpt-3.5-turbo\»,\n» +
            «    \»messages\»: [{\»role\»: \»user\»,
\»content\»: \»« + prompt + «\»}],\n» +
            «    \»max_tokens\»: 512,\n» +
            «    \»top_p\»: 1,\n» +
            «    \»temperature\»: 0.5,\n» +
            «    \»frequency_penalty\»: 0,\n» +
            «    \»presence_penalty\»: 0\n» +
            «}»);

        Request request = new Request.Builder()
            .url(OPENAI_API_URL_2)
            .post(body)
            .addHeader(«Content-Type», «application/json»)

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        .addHeader («Authorization», «Bearer « +
OPENAI_API_KEY)
        .build();

        for (int i = 0; i < 6; i++) {
            try {
                Response response =
client.newCall(request).execute();
                assert response.body() != null;
                String bodyText = response.body().string();
                System.out.println(bodyText);
                ObjectMapper mapper = new ObjectMapper();
                Generator generator = mapper.readValue(bodyText,
Generator.class);
                return
generator.getChoices()[0].getMessage().getContent();

                } catch (SocketTimeoutException ex) {
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }
        return «Нет ответа от сервера Open AI»;
    }
}

```

Класс UserService.java

```

package com.example.bookstore.service;

import com.example.bookstore.domain.Role;
import com.example.bookstore.domain.User;
import com.example.bookstore.repos.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService
;
import
org.springframework.security.core.userdetails.UsernameNotFoundEx
ception;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

import java.util.*;
import java.util.stream.Collectors;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Service
public class UserService implements UserDetailsService {
    @Autowired
    private UserRepo userRepo;

    @Autowired
    private MailSender mailSender;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Value («${hostname}»)
    private String hostname;

    @Override
    public UserDetails loadUserByUsername(String username)
throws UsernameNotFoundException {
        User user = userRepo.findByUsername(username);

        if (user == null) {
            throw new UsernameNotFoundException («User not
found»);
        }

        return user;
    }

    public boolean addUser(User user) {
        User userFromDb = userRepo.findByUsername(user.getUsername());

        if (userFromDb != null) {
            return false;
        }

        user.setActive(true);
        user.setRoles(Collections.singleton(Role.USER));
        user.setActivationCode(UUID.randomUUID().toString());

        user.setPassword(passwordEncoder.encode(user.getPassword()));

        userRepo.save(user);

        sendMessage(user);

        return true;
    }

    private void sendMessage(User user) {
        if (!StringUtils.isEmpty(user.getEmail())) {
            String message = String.format(
                «Hello, %s! \n» +

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        «Welcome to Sweater. Please, visit
next link: http://%s/activate/%s»,
        user.getUsername(),
        hostname,
        user.getActivationCode()
    );

    mailSender.send(user.getEmail(), «Activation code»,
message);
    }
}

public boolean activateUser(String code) {
    User user = userRepo.findByActivationCode(code);

    if (user == null) {
        return false;
    }

    user.setActivationCode(null);

    userRepo.save(user);

    return true;
}

public List<User> findAll() {
    return userRepo.findAll();
}

public void saveUser(User user, String username, Map<String,
String> form) {
    user.setUsername(username);

    Set<String> roles = Arrays.stream(Role.values())
        .map(Role::name)
        .collect(Collectors.toSet());

    user.getRoles().clear();

    for (String key : form.keySet()) {
        if (roles.contains(key)) {
            user.getRoles().add(Role.valueOf(key));
        }
    }

    userRepo.save(user);
}

public void updateProfile(User user, String password, String
email) {
    String userEmail = user.getEmail();

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        boolean isEmailChanged = (email != null &&
!email.equals(userEmail)) ||
        (userEmail != null && !userEmail.equals(email));

        if (isEmailChanged) {
            user.setEmail(email);

            if (!StringUtils.isEmpty(email)) {
user.setActivationCode(UUID.randomUUID().toString());
            }
        }

        if (!StringUtils.isEmpty(password)) {
            user.setPassword(password);
        }

        userRepo.save(user);

        if (isEmailChanged) {
            sendMessage(user);
        }
    }
}

```

Класс RedirectInterceptor.java

```

package com.example.bookstore.util;

import org.springframework.web.servlet.ModelAndView;
import
org.springframework.web.servlet.handler.HandlerInterceptorAdapte
r;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RedirectInterceptor extends
HandlerInterceptorAdapter {
    @Override
    public void postHandle(HttpServletRequest request,
HttpServletResponse response, Object handler, ModelAndView
modelAndView) throws Exception {
        if (modelAndView != null) {
            String args = request.getQueryString() != null ?
request.getQueryString() : «»;
            String url = request.getRequestURI().toString() +
«?» + args;
            response.setHeader(«Turbolinks-Location», url);
        }
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Класс Application.java

```
package com.example.bookstore;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Файл V1__Init_DB.sql

```
create sequence hibernate_sequence start 1 increment 1;

create table book
(
    id            int8            not null,
    title         varchar(255) not null,
    description   varchar(2048),
    author        varchar(255),
    price         int8,
    filename      varchar(255),
    primary key (id)
);

create table orders
(
    id            int8 not null,
    user_id       int8 not null,
    book_id       int8 not null,
    status        varchar(255),
    date_order    TIMESTAMP,
    primary key (id)
);

CREATE TABLE book_content
(
--    id            int8 not null,
    book_id int8 not null,
    content text,
    primary key (book_id)
);

create table user_role
(
    user_id int8 not null,
    roles   varchar(255)
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

);

create table usr
(
    id            int8            not null,
    activation_code varchar(255),
    active        boolean         not null,
    email         varchar(255),
    password      varchar(255) not null,
    username      varchar(255) not null,
    primary key (id)
);

alter table if exists orders
    add constraint orders_fk
    foreign key (book_id) references book;

alter table if exists book_content
    add constraint content_fk
    foreign key (book_id) references book;

alter table if exists user_role
    add constraint user_role_user_fk
    foreign key (user_id) references usr;

```

Файл V2__Add_admin.sql

```

insert into usr (id, username, password, active)
values (1, 'admin', '123', true)

insert into user_role (user_id, roles)
values (1, 'USER'), (1, 'ADMIN');

```

Файл V3__Encode_passwords.sql

```

create extension if not exists pgcrypto;

update usr set password = crypt(password, gen_salt('bf', 8));

```

Файл style.css

```

html {
    height: 100%;
}

body {
    display: grid;
    grid-template-rows: auto 1fr auto;
    margin: 0;
    color: #113;
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        min-height: 100%;
    }

.checkbox-ios {
    display: inline-block;
    height: 28px;
    line-height: 28px;
    margin-right: 10px;
    position: relative;
    vertical-align: middle;
    font-size: 14px;
    user-select: none;
}

.checkbox-ios .checkbox-ios-switch {
    position: relative;
    display: inline-block;
    box-sizing: border-box;
    width: 56px;
    height: 28px;
    border: 1px solid rgba(0, 0, 0, .1);
    border-radius: 25%/50%;
    vertical-align: top;
    background: #eee;
    transition: .2s;
}

.checkbox-ios .checkbox-ios-switch:before {
    content: '';
    position: absolute;
    top: 1px;
    left: 1px;
    display: inline-block;
    width: 24px;
    height: 24px;
    border-radius: 50%;
    background: white;
    box-shadow: 0 3px 5px rgba(0, 0, 0, .3);
    transition: .15s;
}

.checkbox-ios input[type=checkbox] {
    display: block;
    width: 0;
    height: 0;
    position: absolute;
    z-index: -1;
    opacity: 0;
}

.checkbox-ios input[type=checkbox]:not(:disabled):active +
.checkbox-ios-switch:before {
    box-shadow: inset 0 0 2px rgba(0, 0, 0, .3);
}

.checkbox-ios input[type=checkbox]:checked + .checkbox-ios-
switch {

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        background: #007bff;
    }
    .checkboxbox-ios input[type=checkbox]:checked + .checkboxbox-ios-switch:before {
        transform:translateX(28px);
    }

    /* Hover */
    .checkboxbox-ios input[type=checkbox]:not(:disabled) + .checkboxbox-ios-switch {
        cursor: pointer;
        border-color: rgba(0, 0, 0, .3);
    }

    /* Disabled */
    .checkboxbox-ios input[type=checkbox]:disabled + .checkboxbox-ios-switch {
        filter: grayscale(70%);
        border-color: rgba(0, 0, 0, .1);
    }
    .checkboxbox-ios input[type=checkbox]:disabled + .checkboxbox-ios-switch:before {
        background: #eee;
    }

    /* Focus */
    .checkboxbox-ios.focused .checkboxbox-ios-switch:before {
        box-shadow: inset 0px 0px 4px #ff5623;
    }

```

Файл registration.ftl

```

<#import «../parts/common.ftl» as c>
<#import «../parts/login.ftl» as l>

<@c.page>

    <div class=«container my-auto»>
        <h2>Регистрация</h2>
    </div>
    ${book?ifExists}
    <@l.login «/registration» true />
</@c.page>

```

Файл api.ftl

```

<#import «../parts/common.ftl» as c>
<#include «../parts/security.ftl»>
<@c.page>

    <h4 class=«font-weight-normal»>Добавить книгу</h4>
    <div class=«form-group mt-3»>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <form                action=«/api»                method=«post»
enctype=«multipart/form-data»
        <div class=«form-group»>
            <input            type=«text»                class=«form-control
${(textError??)?string('is-invalid', '')}}»
            <#if                title??>value=«${title}»</#if>
name=«title» placeholder=«Название»/>
            <#if textError??>
                <div class=«invalid-feedback»>
                    ${textError}
                </div>
            </#if>
        </div>
        <div class=«form-group»>
            <input type=«text» class=«form-control»
            <#if
description??>value=«${description}»</#if>                name=«description»
placeholder=«Описание»/>
        </div>
        <div class=«form-group»>
            <input type=«text» class=«form-control»
            <#if                author??>value=«${author}»</#if>
name=«author» placeholder=«Автор»/>
        </div>
        <div class=«form-group»>
            <input type=«text» class=«form-control»
            <#if                price??>value=«${price}»</#if>
name=«price» placeholder=«Цена»/>
            <#if tagError??>
                <div class=«invalid-feedback»>
                    ${tagError}
                </div>
            </#if>
        </div>
        <div class=«form-group»>
            <div class=«custom-file»>
                <input            type=«file»                name=«coverImage»
id=«customFile» <#if coverImage??>value=${coverImage}</#if>/>
                <label                class=«custom-file-label»
for=«customFile»>Выберите файл</label>
            </div>
        </div>
        <button type=«submit» id=«save-button» class=«btn
btn-primary» name=«action» value=«save»>Далее</button>

        <input                type=«hidden»                name=«_csrf»
value=«${_csrf.token}»/>
    </form>
</div>

</@c.page>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Файл generate.ftl

```
<#import «../parts/common.ftl» as c>
<#include «../parts/security.ftl»>
<@c.page>
    <div class=«container»>
        <form action=«/api/generate» method=«post»>
            <div class=«container my-auto»>
                <h4>Генератор</h4>
            </div>
            <div class=«form-group»>
                <label for=«input-text»>Введите текст
запроса:</label>
                <textarea class=«form-control» name=«input-text»
id=«input-text» rows=«5»></textarea>
            </div>
            <div class=«btn-group»>
                <button type=«submit» class=«btn btn-primary mr-
2» name=«action» value=«create»>Создать</button>
            </div>
            <hr>
            <div class=«form-group»>
                <label for=«output-text»>Ответ ИИ:</label>
                <textarea class=«form-control» name=«temp-text»
id=«temp-text» rows=«5»>${temp}</textarea>
            </div>
            <div class=«btn-group»>
                <button type=«submit» class=«btn btn-secondary»
name=«action» value=«add»>Дополнить</button>
            </div>
            <hr>
            <div class=«form-group»>
                <label for=«output-text»>Результат:</label>
                <textarea class=«form-control» name=«output-
text» id=«output-text» rows=«5»>${output}</textarea>
            </div>
            <button type=«submit» id=«save-button» class=«btn
btn-outline-success» name=«action» value=«save»>Сохранить
            </button>
            <input type=«hidden» name=«_csrf»
value=«${_csrf.token}»/>
        </form>
    </div>
</@c.page>
```

Файл main.ftl

```
<#import «../parts/common.ftl» as c>
<#import «../parts/pager.ftl» as p>
<#import «../parts/card_book.ftl» as cb>

<@c.page>
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

<div class=«container my-auto»>
  <h2>Каталог</h2>
</div>
<div class=«form-row»>
  <div class=«form-group <>
    <form action=«/main» method=«get» class=«form-
inline»>
      <div class=«container my-4»>
        <div class=«row justify-content-center
align-items-center»>
          <div class=«col-auto»>
            <div class=«d-flex flex-column
justify-content-start <>
              <label class=«d-flex justify-
content-start ml-1» for=«user-filter»>Фильтр</label>
              <button type=«submit» class=«btn
btn-primary ml-1» id=«button_filter»
name=«button_filter»
value=«apply»>Применить
            </button>
          </div>
        </div>
      <div class=«col»>
        <div class=«d-flex flex-column
justify-content-start <>
          <label class=«d-flex justify-
content-start» for=«book-filter»>По названию:</label>
          <input type=«text» class=«form-
control» id=«book-filter» name=«bookFilter»
value=«${bookFilter}»>
        </div>
      </div>
      <div class=«col»>
        <div class=«d-flex flex-column
justify-content-start <>
          <label class=«d-flex justify-
content-start» for=«author-filter»>По автору:</label>
          <input type=«text» class=«form-
control» id=«author-filter» name=«authorFilter»
value=«${authorFilter}»>
        </div>
      </div>
      <div class=«col»>
        <div class=«d-flex flex-column
justify-content-start <>
          <label class=«d-flex justify-
content-start» for=«price-filter»>По цене:</label>
          <div class=«input-group»>
            <input
class=«form-control» id=«price-filter-from»

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

name=«priceFilterFrom»                                placeholder=«от»
value=«${priceFilterFrom}»

                                pattern=«[0-9]*»>
                                <input                type=«text»
class=«form-control» id=«price-filter-to» name=«priceFilterTo»
                                placeholder=«до»
pattern=«[0-9]*» value=«${priceFilterTo}»>
                                </div>
                                </div>
                                </div>
                                <#if user??>
                                <div class=«col»>
                                <label class=«d-flex justify-
content-start» for=«book-filter >>Купленные книги</label>
                                <div class=«form-group mt-1»>
                                <label class=«checkbox-ios»>
                                <#if checkBox>
                                <input type=«hidden»
id=«my-checkbox» name=«myCheckBox» value=«0»>
                                </#if>
                                <input type=«checkbox»
id=«my-checkbox» name=«myCheckBox» value=«1»
                                <#if
checkBox>checked</#if>
onclick=«button_filter.click()»>
                                <span class=«checkbox-
ios-switch»></span>
                                </label>
                                </div>
                                </div>
                                </#if>
                                </div>
                                <input type=«hidden» name=«_csrf»
value=«${_csrf.token}»/>
                                </div>
                                </form>
                                </div>
                                </div>
                                <div class=«container»>
                                <div class=«row >
                                <#list books.content as book>
                                <@cb.cardbook book/>
                                <#else>
                                <div class=«container my-auto»>
                                <#if filter.isEmpty()>
                                <h2>Нет книг</h2>
                                <#else>
                                <h2>Ничего не найдено</h2>
                                </#if>
                                </div>
                                </div>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        </#list>
    </div>
</div>
<@p.pager books url />
</@c.page>

```

Файл product.ftl

```

<#import «../parts/common.ftl» as c>
<#import «../parts/basket_button.ftl» as bb>
<#include «../parts/security.ftl»>

<@c.page>
    <div class=«row»>
        <div class=«col-md-6»>
            <img src=«https://placeholder.it/300x400» alt=«Обложка
книги»>
        </div>
        <div class=«col-md-6»>
            <h1>${book.title}</h1>
            <h3>Автор: ${book.author}</h3>
            <p>Описание книги: ${book.description} </p>
            <h4>Цена: ${book.price} руб.</h4>
            <hr>
            <h4>Отзывы:</h4>
            <p>Добавить в корзину</p>
            <@bb.basket_btn bought «/product/${book.id}/buy»/>
        </div>
    </div>
    <#if isAdmin>
        <hr class=«my-5»>
        <div class=«row mt-4»>
            <div class=«col-md-6»>
                <form
                                method=«POST»
action=«/product/${book.id}/edit»>
                    <div class=«form-group»>
                        <label for=«title»>Название:</label>
                        <input
                                type=«text»
                                name=«title»
id=«title» class=«form-control» value=«${book.title}» required>
                    </div>

                    <div class=«form-group»>
                        <label for=«author»>Автор:</label>
                        <input
                                type=«text»
                                name=«author»
id=«author» class=«form-control» value=«${book.author}»
                                required>
                    </div>

                    <div class=«form-group»>
                        <label
for=«description»>Описание:</label>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

<textarea                                name=«description»
id=«description» class=«form-control»

required>${book.description}</textarea>
</div>

<div class=«form-group»>
    <label for=«price»>Цена:</label>
    <input      type=«number»      name=«price»
id=«price» class=«form-control» value=«${book.price}»
required>
</div>

<button type=«submit» class=«btn btn-
primary» name=«action» value=«update»>Сохранить</button>
<button type=«submit» class=«btn btn-
outline-danger» name=«action» value=«delete»>Удалить</button>
<input      type=«hidden»      name=«id»
value=«${book.id}» />
<input      type=«hidden»      name=«_csrf»
value=«${_csrf.token}»>/>
</form>
</div>
</div>
</#if>

</@c.page>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Файл loyalty.ftl

```
<#import «../parts/common.ftl» as c>

<@c.page>
    <div class=«container»>
        <h1 class=«text-center mb-4»>Программа лояльности</h1>
        <hr>
    </div>
    <div class=«container mb-4»>
        <div class=«row»>
            <div class=«col-md-6»>
                <h2>Преимущества участия в программе лояльности</h2>
                <ul>
                    <li>Скидки до 20% на все товары книжного магазина;</li>
                    <li>Бонусные баллы за покупки, которые можно обменять на книги;</li>
                    <li>Получение эксклюзивных предложений и специальных акций;</li>
                    <li>Доступ к закрытым мероприятиям и мастер-классам для членов программы.</li>
                </ul>
            </div>
            <div class=«col-md-6»>
                <h2>Как стать участником программы лояльности</h2>
                <ol>
                    <li>Зарегистрируйтесь на сайте книжного магазина.</li>
                    <li>Совершайте покупки на сумму от 1000 рублей.</li>
                    <li>Накапливайте бонусные баллы и получайте скидки на следующие покупки.</li>
                </ol>
            </div>
        </div>
    </div>
</@c.page>
```

Файл support.ftl

```
<#import «../parts/common.ftl» as c>

<@c.page>
    <div class=«container»>
        <h1>Нужна помощь? Мы здесь, чтобы ее оказать!</h1>
        <p>Обратитесь за помощью: наша команда готова помочь!</p>
        <hr>
    </div>
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

<div class=«container my-4»>
  <form action=«/support» method=«post»>
    <div class=«form-group»>
      <label for=«name»>Ваше имя</label>
      <input type=«text» class=«form-control»
id=«name» name=«name» required>
    </div>
    <div class=«form-group»>
      <label for=«email»>Email адрес</label>
      <input type=«email» class=«form-control»
id=«email» name=«email» required>
    </div>
    <div class=«form-group»>
      <label for=«message»>Сообщение</label>
      <textarea class=«form-control» id=«message»
name=«message» rows=«5» required></textarea>
    </div>
    <button type=«submit» class=«btn btn-
primary»>Отправить</button>
    <input type=«hidden» name=«_csrf»
value=«${_csrf.token}»/>
  </form>
</div>
</@c.page>

```

Файл editUser.ftl

```

<#import «../parts/common.ftl» as c>

<@c.page>
User editor

<form action=«/user» method=«post»>
  <input type=«text» name=«username» value=«${user.username}»
/>

  <#list roles as role>
  <div>
    <label><input type=«checkbox» name=«${role}»
${user.roles?seq_contains(role)?string(«checked», «») }
/>${role}</label>
  </div>
</#list>
  <#list orders as order>
  <div>
    <label>${order.book.getTitle}</label>
  </div>
</#list>

  <input type=«hidden» value=«${user.id}» name=«userId» />
  <input type=«hidden» value=«${_csrf.token}» name=«_csrf» />
  <button type=«submit»>Save</button>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

</form>

Файл profile.ftl

```
<#import «../parts/common.ftl» as c>
```

```
<@c.page>
```

```
<h5>${username}</h5>
```

```
${book?ifExists}
```

```
<form method=«post»>
```

```
    <div class=«form-group row»>
```

```
        <label class=«col-sm-2 col-form-label»>Password:</label>
```

```
        <div class=«col-sm-6»>
```

```
            <input type=«password» name=«password» class=«form-control» placeholder=«Password» />
```

```
        </div>
```

```
    </div>
```

```
    <div class=«form-group row»>
```

```
        <label class=«col-sm-2 col-form-label»>Email:</label>
```

```
        <div class=«col-sm-6»>
```

```
            <input type=«email» name=«email» class=«form-control» placeholder=«some@some.com» value=«${email!''}» />
```

```
        </div>
```

```
    </div>
```

```
    <input type=«hidden» name=«_csrf» value=«${_csrf.token}» />
```

```
    <button class=«btn btn-primary» type=«submit»>Save</button>
```

```
</form>
```

```
</@c.page>
```

Файл userList.ftl

```
<#import «../parts/common.ftl» as c>
```

```
<@c.page>
```

```
List of users
```

```
<table>
```

```
    <thead>
```

```
        <tr>
```

```
            <th>Name</th>
```

```
            <th>Role</th>
```

```
            <th></th>
```

```
        </tr>
```

```
    </thead>
```

```
    <tbody>
```

```
        <#list users as user>
```

```
            <tr>
```

```
                <td>${user.username}</td>
```

```
                <td><#list user.roles as role>${role}<#sep>,</td>
```

```
            </td>
```

```
                <td><a href=«/user/${user.id}»>edit</a></td>
```

```
            </tr>
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        </#list>
      </tbody>
    </table>
  </@c.page>

```

Файл basket.ftl

```

<#import «../parts/common.ftl» as c>

<@c.page>
  <div class=«container my-auto»>
    <h2>Корзина</h2>
  </div>

  <table class=«table table-bordered»>
    <thead>
      <tr>
        <th scope=«col»>#</th>
        <th scope=«col»>Название</th>
        <th scope=«col»>Сумма</th>
      </tr>
    </thead>
    <tbody>
      <#list page.content as book>
        <tr>
          <th scope=«row»>${book?index+1}</th>
          <td>
            <div class=«d-flex justify-content-between»>
              <a
                href=«/product/${book.id}»
                style=«text-decoration: none;color: inherit;»>
                ${book.title}</a>
              <a href=«/books/${book.id}/order»
                type=«button» class=«btn btn-outline-
danger btn-sm»>
                <i class=«bi bi-trash»></i>
              </a>
            </div>
          </td>
          <td>${book.price} руб.</td>
        </tr>
      <#else>
        <tr>
          <th scope=«row»>-</th>
          <td>-</td>
          <td>-</td>
        </tr>
      </#list>
    </tbody>
    <tfoot>
      <tr>
        <td colspan=«2» class=«text-right font-weight-
bold»>Итого:</td>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <td>${totalPrice} руб.</td>
    </tr>
</tfoot>
</table>
<div class=«text-center»>
    <a href=«/basket/payment» class=«btn btn-primary mr-2»>Оформить заказ</a>
    <a href=«/main» class=«btn btn-secondary»>Продолжить покупки</a>
</div>
</@c.page>

```

Файл checkout.ftl

```

<#import «../parts/common.ftl» as c>

<@c.page>
    <div class=«container my-auto»>
        <h2>Оформление заказа</h2>
    </div>

    <form>
        <div class=«form-group»>
            <label for=«name»>Имя:</label>
            <input type=«text» class=«form-control» id=«name»
required
                value=«${userChannel.username}>»
            </div>

        <div class=«form-group»>
            <label for=«email»>Email:</label>
            <input type=«email» class=«form-control» id=«email»
required
                placeholder=«email@example.com»
                value=«<#if
userChannel.email??>${userChannel.email}<#else></#if><»
            </div>

        <div class=«form-group»>
            <label for=«phone»>Телефон:</label>
            <input type=«tel» class=«form-control» id=«phone»
required
                placeholder=«+7 (xxx) xxx-xx-xx»
                value=«+79534449955»
            </div>

        <div class=«form-group»>
            <label for=«address»>Адрес доставки:</label>
            <textarea class=«form-control» id=«address» rows=«3»
required></textarea>
        </div>

        <div class=«form-group»>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <label for=«payment»>Способ оплаты:</label>
        <select class=«form-control» id=«payment» required>
            <option value=«»>Выберите способ оплаты</option>
            <option value=«visa»>Visa</option>
            <option value=«mastercard»>MasterCard</option>
            <option value=«paypal»>PayPal</option>
        </select>
    </div>

    <a href=«/basket/confirmation» type=«submit» class=«btn
btn-primary»>Оформить заказ</a>
</form>

```

</@c.page>

Файл confirmation.ftl

```

<#import «../parts/common.ftl» as c>

<@c.page>
    <h2>Ваш заказ успешно оформлен!</h2>
    <p>Спасибо за покупку! Мы свяжемся с вами в ближайшее время
для уточнения деталей.</p>
    <hr>
    <a href=«/main» class=«btn btn-primary»>Перейти в
каталог</a>
    <a href=«/user/orders» class=«btn btn-secondary»>Перейти в
раздел заказов</a>
</@c.page>

```

Файл orders.ftl

```

<#import «../parts/common.ftl» as c>

<@c.page>
    <div class=«container my-auto»>
        <h2>Список заказов</h2>
    </div>

    <form action=«/orders» method=«get»>

        <div class=«container my-4»>
            <div class=«row g-2 justify-content-center»>
                <div class=«col-2»>
                    <div class=«form-group»>
                        <label for=«user-filter»>Фильтр</label>
                        <button type=«button» class=«btn btn-
primary» style=«display: block;»>Применить</button>
                    </div>
                </div>
                <div class=«col»>

```

						Лист
					ПГУ1.090301.08.001 ПЗ	
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <div class=«form-group»>
            <label                    for=«user-filter»>По
пользователю:</label>
            <input  type=«text»  class=«form-control»
id=«user-filter»>
        </div>
    </div>
    <div class=«col»>
        <div class=«form-group»>
            <label                    for=«book-filter»>По
книге:</label>
            <input  type=«text»  class=«form-control»
id=«book-filter»>
        </div>
    </div>
    <div class=«col»>
        <div class=«form-group»>
            <label                    for=«book-filter»>По
date:</label>
            <input  type=«text»  class=«form-control»
id=«book-filter»>
        </div>
    </div>
    <div class=«col»>
        <div class=«form-group»>
            <label                    for=«book-filter»>По
статусу:</label>
            <input  type=«text»  class=«form-control»
id=«book-filter»>
        </div>
    </div>
    </div>
    <input                    type=«hidden»                    name=«_csrf»
value=«${_csrf.token}»/>
</div>
</form>

```

```

<table class=«table table-striped text-center align-middle»>
    <thead>
        <tr>
            <th          scope=«col»          class=«col-2          text-
left»>Пользователь</th>
            <th          scope=«col»          class=«col-md-auto          text-
left»>Книга</th>
            <th scope=«col» class=«col-2»>Дата заказа</th>
            <th scope=«col» class=«col-3»>Статус</th>
            <th scope=«col» class=«col-1»>Удалить</th>
            <th scope=«col» class=«col-1»>Написать</th>
        </tr>
    </thead>
    <#list page.content as order>
        <tbody>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <tr>
            <td
                class=«text-
left»>${order.getAuthor().username}</td>
            <td
                class=«text-left»><a
                    style=«text-decoration:
none;color: inherit;»>${order.getBook().title}</a></td>
            <td>${order.getDate()}</td>
            <td>
                <form action=«/orders» method=«post»>
                    <div class=«form-group»>
                        <div class=«btn-group btn-group-
toggle» data-toggle=«buttons»>
                            <label class=«btn <#if
order.status.isFormed()>btn-primary active<#else>btn-
secondary</#if><»
                                <input type=«radio»
name=«status» id=«formed» value=«formed»
onclick=«this.form.submit()»
                                    >Форм
                                </label>
                                <label class=«btn <#if
order.status.isIssued()>btn-primary active<#else>btn-
secondary</#if><»
                                    <input type=«radio»
name=«status» id=«issued» value=«issued»
onclick=«this.form.submit()»
                                        >Выполн
                                    </label>
                                    <label class=«btn <#if
order.status.isClosed()>btn-primary active<#else>btn-
secondary</#if><»
                                        <input type=«radio»
name=«status» id=«closed» value=«closed»
onclick=«this.form.submit()»
                                            >Закрыт
                                        </label>
                                    </div>
                                </div>
                                <input type=«hidden» name=«_csrf»
value=«${_csrf.token}»/>
                                <input type=«hidden» name=«order_id»
value=«${order.id}»/>
                            </form>
                        </td>
            <td>
                <form action=«/orders» method=«post»>
                    <button type=«submit» class=«btn btn-
outline-danger»

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

name=«status» id=«formed»
value=«unknown»>
        <i class=«bi bi-trash»></i>
        </button>
        <input type=«hidden» name=«_csrf»
value=«${_csrf.token}»/>
        <input type=«hidden» name=«order_id»
value=«${order.id}»/>
        </form>
    </td>
    <td>
        <button type=«button» class=«btn btn-
secondary»>
            <i class=«bi bi-envelope»></i>
        </button>
    </td>
</tr>
</tbody>
</#list>
</table>
</@c.page>

```

Файл payment.ftl

```

<#import «../parts/common.ftl» as c>

<@c.page>
    <h2>Оплата</h2>
    <form action=«/orders» method=«post»>
        <div class=«form-group»>
            <label for=«card-number»>Номер карты:</label>
            <input type=«text» class=«form-control» id=«card-
number» name=«card-number»
                placeholder=«XXXX XXXX XXXX XXXX»
                required pattern=«[0-9]{4}[0-9]{4}[0-9]{4}[0-
9]{4}»>
        </div>
        <div class=«form-group»>
            <label for=«card-name»>Имя владельца карты:</label>
            <input type=«text» class=«form-control» id=«card-
name» name=«card-name»
                placeholder=«Введите имя как на карте»
            </div>
        <div class=«form-group»>
            <label for=«expiry-date»>Срок действия:</label>
            <div class=«row»>
                <div class=«col-2 ml-2»>
                    <div class=«row»>
                        <div class=«col»>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <input      type=«text»      class=«form-
control» id=«expiry-month» name=«expiry-month» placeholder=«MM»
                                required pattern=«[0-9]{2}»>
    </div>
    <div class=«col»>
        <input      type=«text»      class=«form-
control» id=«expiry-year-first» name=«expiry-year»
                                placeholder=«ГГ»      required
pattern=«[0-9]{2}»>
    </div>
</div>
<div class=«col-1»>
    <input      type=«text»      class=«form-control»
id=«cvv» name=«cvv» placeholder=«CVV» required
                                pattern=«[0-9]{3}»>
</div>
    <input      type=«hidden»      id=«expiry-year»
name=«expiry-year»>
</div>
</div>
    <button      type=«submit»      class=«btn      btn-
primary»>Оплатить</button>
    <a href=«#» class=«btn btn-secondary»>Назад</a>
</form>
</@c.page>

```

Файл userBooks.ftl

Файл userOrder.ftl

Файл viewer.ftl

Файл basket_button.ftl

Файл card_book.ftl

Файл common.ftl

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

Файл login.ftl

Файл navbar.ftl

Файл pager.ftl

Файл security.ftl

Файл login.ftl

Файл application

```
spring.datasource.url=jdbc:postgresql://localhost/bookstore
spring.datasource.username=postgres
spring.datasource.password=****
spring.jpa.generate-ddl=false
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=validate

spring.freemarker.expose-request-attributes=true
server.port=8090

upload.path=/home/dru/uploads

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation
=true

spring.mail.host=smtp.yandex.ru
spring.mail.username=uw@dru.su
spring.mail.password=12345678
spring.mail.port=465
spring.mail.protocol=smtps
mail.debug=false

recaptcha.secret=*****

spring.session.jdbc.initialize-schema=always
spring.session.jdbc.table-name=SPRING_SESSION

spring.jpa.properties.hibernate.jdbc.time_zone=UTC
```

Файл pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0">
```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        xmlns:xsi=«http://www.w3.org/2001/XMLSchema-instance»
        xsi:schemaLocation=«http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd»>
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.example</groupId>
        <artifactId>sweater</artifactId>
        <version>1.0-SNAPSHOT</version>

        <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>2.0.0.RELEASE</version>
        </parent>

        <repositories>
            <repository>
                <id>openai-repo</id>
                <url>https://packages.openai.com/maven/</url>
            </repository>
        </repositories>

        <dependencies>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
freemarker</artifactId>
            </dependency>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
                <optional>true</optional>
            </dependency>

            <!-- DataBase support -->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-
jpa</artifactId>
            </dependency>
            <dependency>
                <groupId>org.postgresql</groupId>
                <artifactId>postgresql</artifactId>
                <scope>runtime</scope>
            </dependency>
            <dependency>
                <groupId>org.flywaydb</groupId>
                <artifactId>flyway-core</artifactId>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

</dependency>

<!-- security -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-
security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-jdbc</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>

<!-- JAXB -->
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-core</artifactId>
    <version>2.3.0</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-impl</artifactId>
    <version>2.3.0</version>
</dependency>

<!-- testing -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <version>RELEASE</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>jakarta.persistence</groupId>
    <artifactId>jakarta.persistence-api</artifactId>
    <version>2.2.3</version>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

</dependency>

<!-- API chatGPT      -->
<dependency>
  <groupId>com.squareup.okhttp3</groupId>
  <artifactId>okhttp</artifactId>
  <version>4.9.3</version>
</dependency>
<dependency>
  <groupId>org.jetbrains.kotlin</groupId>
  <artifactId>kotlin-stdlib</artifactId>
  <version>1.3.70</version>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
</dependency>
<dependency>
  <groupId>net.minidev</groupId>
  <artifactId>json-smart</artifactId>
  <version>2.4.10</version>
  <scope>compile</scope>
</dependency>

</dependencies>

<properties>
  <java.version>1.8</java.version>
</properties>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-
plugin</artifactId>
    </plugin>
    <plugin>
      <groupId>org.jetbrains.kotlin</groupId>
      <artifactId>kotlin-maven-plugin</artifactId>
      <version>1.3.70</version>
      <executions>
        <execution>
          <id>compile</id>
          <phase>process-sources</phase>
          <goals>
            <goal>compile</goal>
          </goals>
          <configuration>
            <sourceDirs>

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

        <source>src/main/java</source>
        <source>target/generated-
sources/annotations</source>
        </sourceDirs>
    </configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
</project>

```

Класс EncryptionConfig.java

```

package com.example.bookstore.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder
;
import
org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
public class EncryptionConfig {
    @Bean
    public PasswordEncoder getPasswordEncoder() {
        return new BCryptPasswordEncoder(8);
    }
}

```

Класс MailConfig.java

```

package com.example.bookstore.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.JavaMailSenderImpl;

import java.util.Properties;

@Configuration
public class MailConfig {
    @Value («${spring.mail.host}»)
    private String host;

    @Value («${spring.mail.username}»)
    private String username;

    @Value («${spring.mail.password}»)

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

private String password;

@Value («${spring.mail.port}»)
private int port;

@Value («${spring.mail.protocol}»)
private String protocol;

@Value («${mail.debug}»)
private String debug;

@Bean
public JavaMailSender getMailSender() {
    JavaMailSenderImpl mailSender = new
JavaMailSenderImpl();

    mailSender.setHost(host);
    mailSender.setPort(port);
    mailSender.setUsername(username);
    mailSender.setPassword(password);

    Properties properties =
mailSender.getJavaMailProperties();

    properties.setProperty («mail.transport.protocol»,
protocol);
    properties.setProperty («mail.debug», debug);

    return mailSender;
}
}

```

Класс MvcConfig.java

```

package com.example.bookstore.config;

import com.example.bookstore.util.RedirectInterceptor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;
import
org.springframework.web.servlet.config.annotation.InterceptorReg
istry;
import
org.springframework.web.servlet.config.annotation.ResourceHandle
rRegistry;
import
org.springframework.web.servlet.config.annotation.ViewController
Registry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigur
er;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Configuration
public class MvcConfig implements WebMvcConfigurer {
    @Value («${upload.path}»)
    private String uploadPath;

    @Bean
    public RestTemplate getRestTemplate() {
        return new RestTemplate();
    }

    public void addViewControllers (ViewControllerRegistry
registry) {

registry.addViewController («/login»).setViewName («login»);
    }

    @Override
    public void addResourceHandlers (ResourceHandlerRegistry
registry) {
        registry.addResourceHandler («/img/**»)
            .addResourceLocations («file://» + uploadPath +
«/»);
        registry.addResourceHandler («/static/**»)
            .addResourceLocations («classpath:/static/»);
    }

    @Override
    public void addInterceptors (InterceptorRegistry registry) {
        registry.addInterceptor (new RedirectInterceptor());
    }
}

```

Класс WebSecurityConfig.java

```

package com.example.bookstore.config;

import com.example.bookstore.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.bu
ilders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.method.configurat
ion.EnableGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.Http
Security;
import
org.springframework.security.config.annotation.web.configuration
.EnableWebSecurity;

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

```

import
org.springframework.security.config.annotation.web.configuration
.WebSecurityConfigurerAdapter;
import
org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {
    @Autowired
    private UserSevice userService;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Override
    protected void configure(HttpSecurity http) throws Exception
    {
        http
            .authorizeRequests()
                .antMatchers («/main»,»/», «/registration»,
«/static/**», «/activate/*»).permitAll()
                .anyRequest().authenticated()
            .and()
                .formLogin()
                .loginPage («/login»)
                .permitAll()
            .and()
                .rememberMe()
            .and()
                .logout()
                .permitAll();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth)
throws Exception {
        auth.userDetailsService(userService)
            .passwordEncoder(passwordEncoder);
    }
}

```

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		

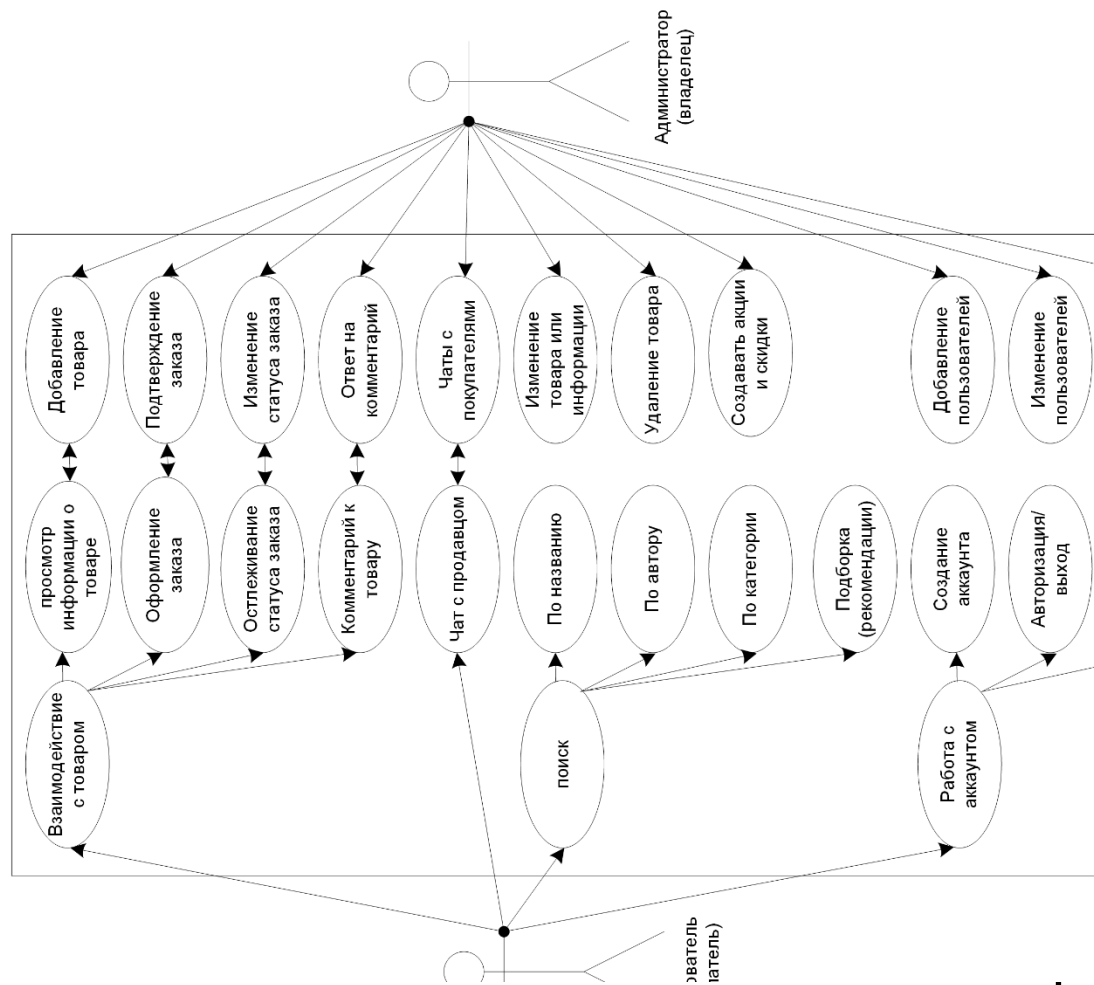
Приложение Б

(Обязательное)

Программное обеспечение интернет магазина книг

UML – диаграмма прецедентов

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		



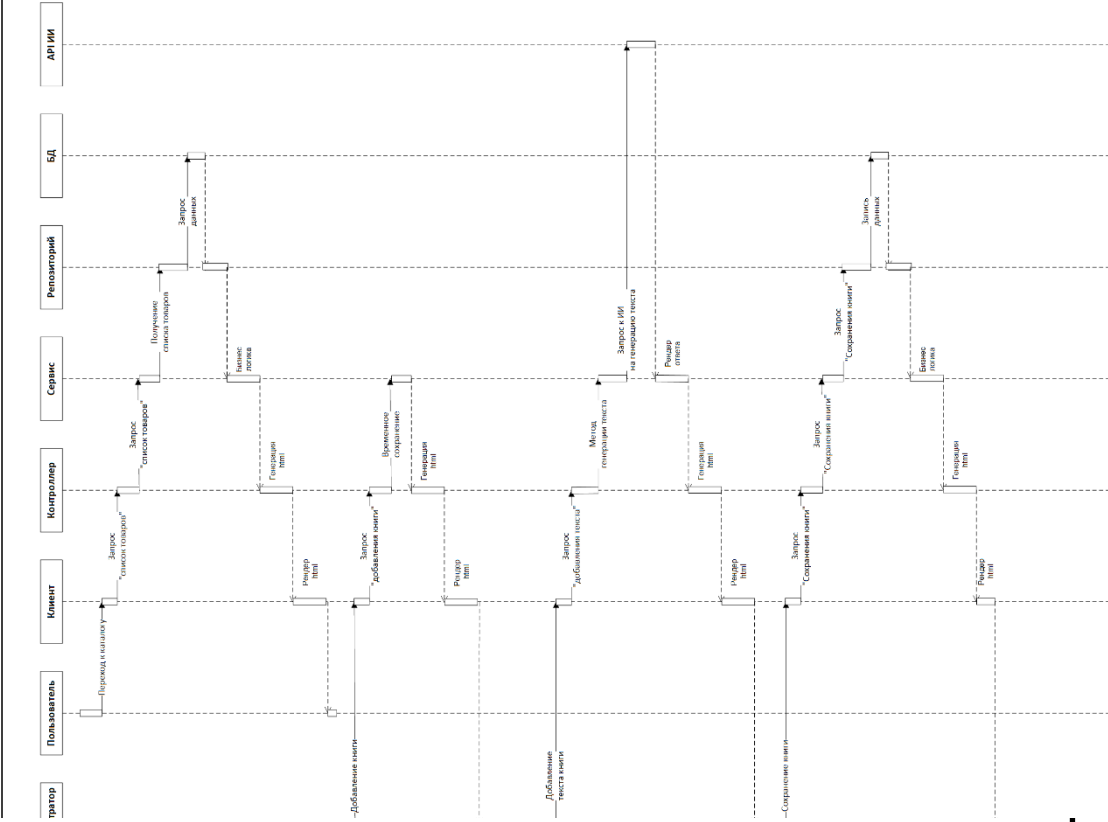
Приложение В

(Обязательное)

Программное обеспечение интернет магазина книг

UML – диаграмма последовательности

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		



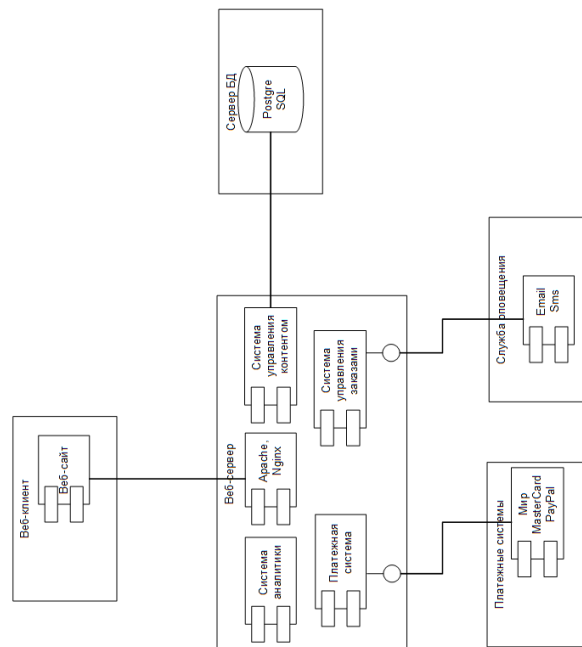
Приложение Г

(Обязательное)

Программное обеспечение интернет магазина книг

UML – диаграмма развертывания

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		



Приложение Д

(Обязательное)

Программное обеспечение интернет магазина книг

UML – диаграмма классов

ШУ 1.090301					Программное обеспечение интернет-магазина книг. UML – диаграмма развертывания				
Изм.	Лист	№ докум.	Подпись	Дата	Изм.	Лист	№ докум.	Подпись	Дата
					Разработчик		Должность И.П.		
					Проектировщик		Должность И.П.		
					Т. контр.		Должность И.П.		
					И. контр.		Должность И.П.		

					ПГУ1.090301.08.001 ПЗ					Лист
Изм.	Лист	№ докум.	Подпись	Дата						

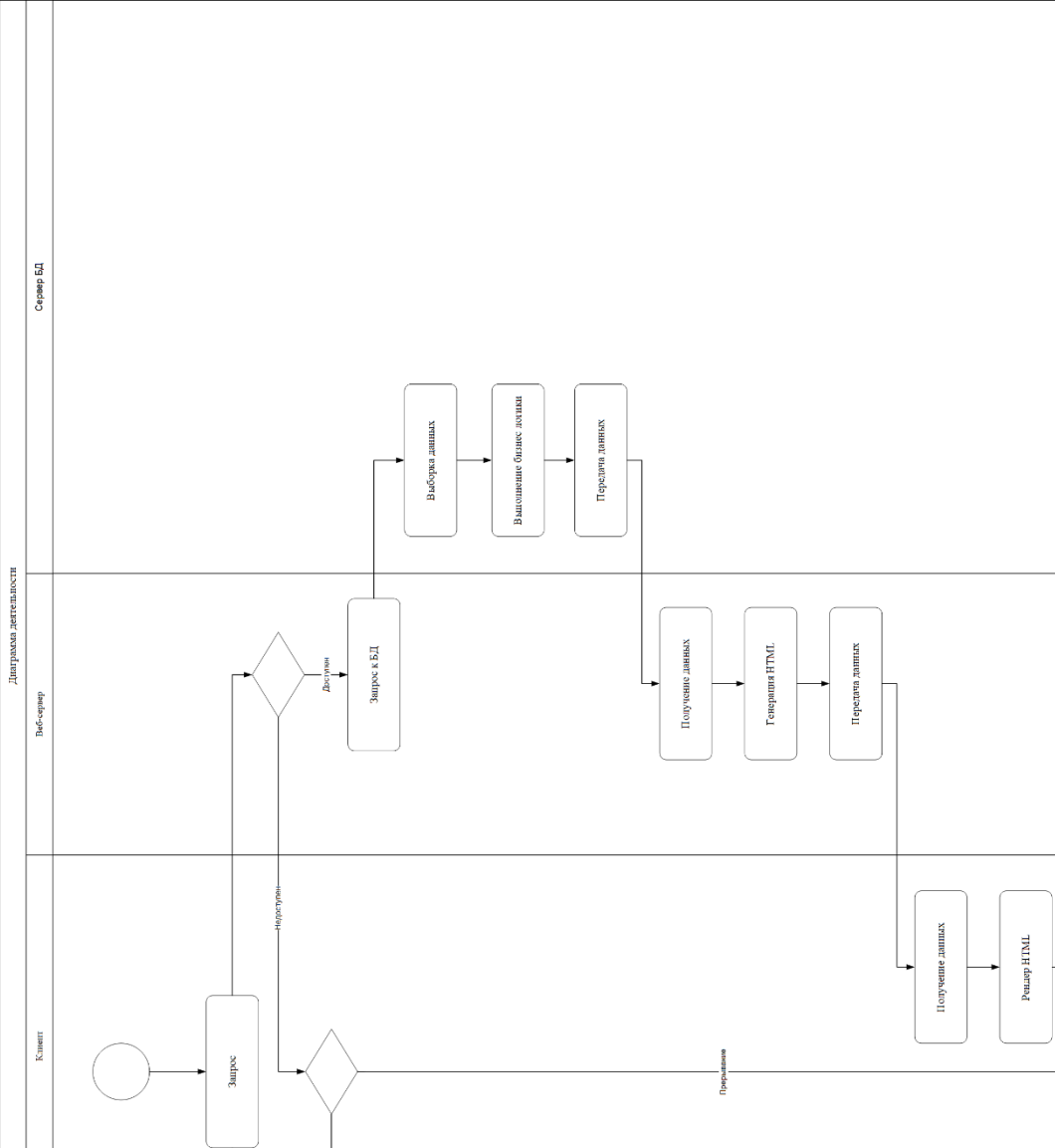
Приложение Е

(Справочное)

Программное обеспечение интернет магазина книг

UML – диаграмма деятельности

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		



Приложение Ж

(Справочное)

Программное обеспечение интернет магазина книг

Презентация

					ПГУ1.090301.08.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		