

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий  
Кафедра Информационных систем и технологий  
Специальность 1-40 01 01 «Программное обеспечение информационных технологий»  
Специализация 1-40 01 01 10 «Программное обеспечение информационных технологий  
(программирование интернет-приложений)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Web-приложение «Книжный магазин»

Выполнил студент Максимчикова Юлия Сергеевна  
(Ф.И.О.)

Руководитель проекта асс. Дубовик М.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты асс. Дубовик М.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер асс. Дубовик М.В.  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой \_\_\_\_\_

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
 Учреждение образования  
 «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»  
 Факультет информационных технологий  
 Кафедра информационных систем и технологий

Утверждаю  
 Заведующий кафедрой  
 \_\_\_\_\_ В.В. Смелов  
 подпись инициалы и фамилия  
 “    ” \_\_\_\_\_ 2020г.

**ЗАДАНИЕ**  
**к курсовому проектированию**  
**по дисциплине " Программирование в Internet"**

Специальность: 1-40 01 01 «Программное обеспечение Группы: 5  
информационных технологий»

Студент: Максимчикова Ю.С.

**Тема: WEB-приложение «Книжный магазин»**

**1. Срок сдачи студентом законченной работы: "7 декабря 2020 г."**

**2. Исходные данные к проекту:**

**2.1. Функционально ПС должно:**

- сохранение информации в централизованной базе данных;
- изменение и удаление уже существующих данных;
- наличие нескольких типов пользователей;
- фильтрация и сортировка данных;
- поиск данных.

**2.2.** Программное средство должно быть выполнено с использованием асинхронного программирования, взаимодействовать с базой данных, реализовано под разными платформами. Программное средство должно представлять собой web-приложение с асинхронным UI (React) и должно предоставлять API (REST). Приложение должно быть развернуто с помощью Docker. Docker должен быть установлен на Windows и Linux. Отображение, бизнес логика и хранилище данных должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, последовательности разработать на основе UML. Язык разработки проекта C#. Web-приложение должно быть логически завершенным. Управление программой должно быть интуитивно понятным и удобным. Листинги проекта должны содержать комментарии.

**3. Содержание расчетно-пояснительной записки**

- Введение (актуальность задачи)
- Постановка задачи (алгоритмы решения, обзор прототипов)
- Проектирование (выбор платформы, конфигурация, диаграмма вариантов использования, диаграмма последовательности (по необходимости),

логическая схема базы данных, описание базы данных)

- Разработка
- Тестирование (ручное или unit-тесты)
- Руководство программиста (как развернуть приложение)
- Тестирование
- Заключение
- Список используемых источников
- Приложения

#### 4. Форма представления выполненной курсовой работы:

- Теоретическая часть курсового проекта должны быть представлены в формате MS Word. Оформление записки должно быть согласно выданным правилам.
- Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio, Rational Rose, WS или копии экрана (интерфейс).
- Листинги программы представляются частично в приложении.
- К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и инсталляцию проекта.

#### Календарный план

№ п/п	Наименование этапов курсового проекта	Срок выполнения этапов проекта	Примечание
1	Введение	08.10.2020	
2	Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования	15.10.2020	
3	Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных)	29.10.2020	
4	Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса	05.11.2020	
5	Кодирование программного средства	19.11.2020	
6	Тестирования и отладка программного средства	26.11.2020	
7	Оформление пояснительной записки	03.12.2020	
8	Сдача проекта	07.12.2020	

5. Дата выдачи задания 25.09.2020

Руководитель \_\_\_\_\_ М.В. Дубовик  
(подпись)

Задание принял к исполнению \_\_\_\_\_

(дата и подпись студента)

## Содержание

Введение .....	5
Реферат.....	6
1 Постановка задачи .....	7
1.1 Аналитический обзор прототипов и литературных источников .....	7
1.2 Анализ требований к программному средству и разработка функциональных требований.....	8
2 Проектирование программного средства .....	11
2.1 Обзор используемых технологий.....	11
2.2 Проектирование архитектуры приложения .....	12
2.3 Проектирование базы данных .....	15
3 Реализация программного средства.....	17
3.1 Реализация сущностей.....	17
3.2 Реализация серверной части.....	17
3.3 Реализация клиентской части.....	19
3.4 Реализация аутентификации и авторизации .....	22
4 Тестирование, проверка работоспособности и анализ полученных результатов .	23
5 Руководство пользователя .....	27
6 Руководство программиста.....	35
Заключение .....	39
Список использованных источников.....	40
ПРИЛОЖЕНИЕ А.....	41
ПРИЛОЖЕНИЕ Б .....	42
ПРИЛОЖЕНИЕ В .....	43

## Введение

Стремительное развитие сети Интернет привело к тому, что сегодня всё больше и больше магазинов, от маленьких узкоспециализированных до крупных торговых сетей стремятся иметь своё представительство в сети Интернет, а многие вообще осуществляют свою деятельность исключительно в сети. Интернет предоставляет новые каналы сбыта продукции, открывает широкие возможности для рекламы и маркетинговых исследований.

Интернет-магазин — это виртуальный магазин с реальными товарами, инструмент для продажи товаров и услуг через Интернет. Он имеет ряд преимуществ перед обычным магазином. Информация о клиенте может храниться в системе управления магазином, доступ к витрине имеют покупатели со всего мира, имеется возможность расширения географии продаж. Кроме того, владелец интернет-магазина может изучать поведение покупателей на сайте, интерес к товарам, и таким образом подстраивать свой магазин под актуальные потребности потребителя. Все это позволит значительно увеличить продажи. Также интернет-магазин удобен и для покупателя. Большой ассортимент, удобный выбор, возможность сравнить товар и изучить характеристики, отзывы, удобные цены, круглосуточный заказ и доступ к магазину и товарам, удобная доставка на дом или в офис. Неудивительно, что выбор темы для данного курсового проекта остановился на разработке web-приложения, реализующего работу книжного магазина в сети Интернет.

Разработка данного приложения осуществлялась на языке C#. Для разработки базы данных использовалась СУБД MS SQL, а для разработки API-приложения использовались такие технологии как: ASP.Net Core, Entity Framework Core, MediatR, FluentValidation, NSwag – на стороне сервера, и библиотеки React, React Bootstrap, React Router, – на стороне клиента.

## Реферат

Пояснительная записка курсового проекта состоит из 43 страниц, 28 рисунков, 3 приложений, 6 источников литературы.

В первом разделе рассматриваются прототипы и актуальность задачи.

Во втором разделе описана архитектура курсового проекта.

В третьем разделе предоставлена информация о разработанных компонентах приложения.

Четвертый раздел содержит руководство пользователя для разработанного клиентского приложения.

В пятом разделе представлены результаты тестирования приложения.

В заключении описывается результат курсового проектирования и задачи, которые были решены в ходе разработки приложения.

## 1 Постановка задачи

В данной курсовой работе ставится задача разработки книжного интернет-магазина, предназначенного для реализации продажи книг в сети Интернет.

### 1.1 Аналитический обзор прототипов и литературных источников

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников. Анализ рынка книжной продукции показал, что на сегодняшний день разработано немалое количество подобных проектов. Среди всех решений наиболее популярными в Беларуси являются: oz.by, biblio.by и belkniga.by.

Остановимся подробнее на белорусской компании OZ и её книжном интернет-магазине oz.by. Появившись 20 лет назад, OZ.by стал одним из первых белорусских интернет-магазинов и первым книжным интернет-магазином. А с 2009 года компания начала открывать розничные магазины «OZ Книги», которые функционируют и как пункты выдачи заказов. К настоящему моменту в ассортименте магазина не только бумажные, но и электронные книги, а также канцтовары, настольные игры, сувенирная продукция и т.д. Интерфейс интернет-магазина oz.by представлен на рисунке 1.1.

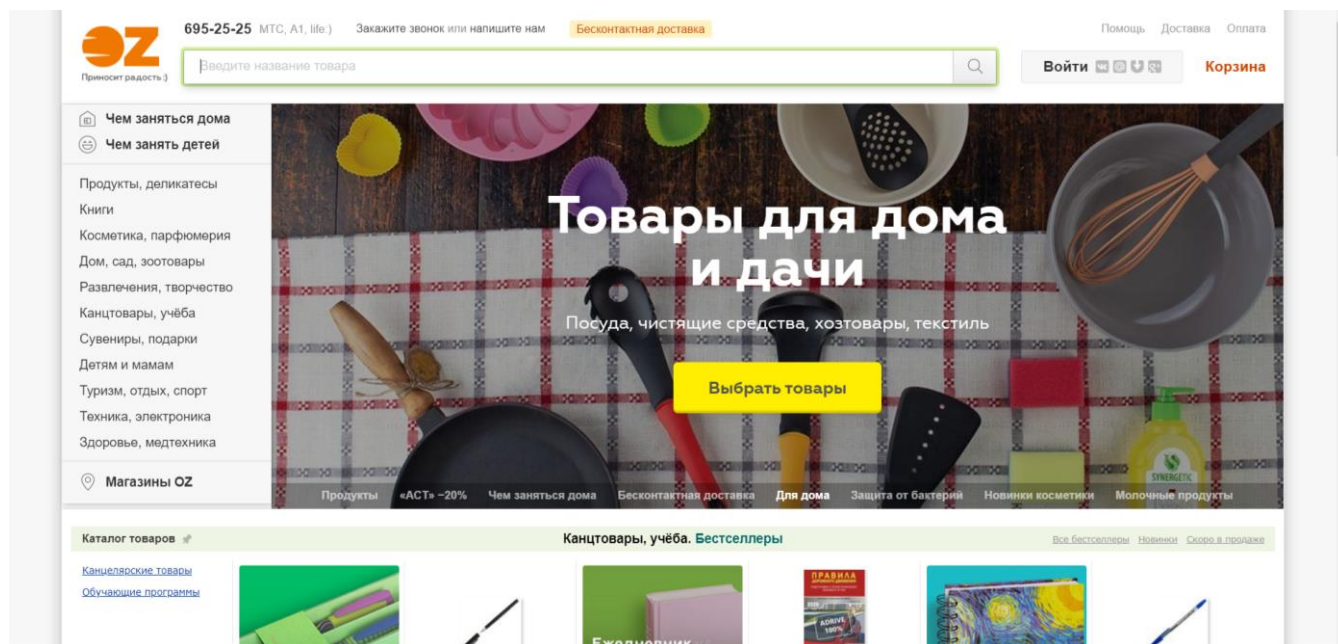


Рисунок 1.1 – Интерфейс oz.by

Интерфейс посетителя достаточно прост. Посетитель может просматривать информацию главной страницы, список товаров и сами товары. Купить товар посетитель может только зарегистрировавшись, тем самым перейдя в категорию пользователей.

Интерфейс пользователя даёт возможность совершать покупки, а также открывать доступ к личному кабинету, где отображается корзина, подготовленные заказы, а также история выполненных заказов. Помимо текстовой информации о

товаре покупателю предоставлена еще и графическая в виде фотографий товаров. В целом сайт оформлен в спокойном неагрессивном стиле, не отвлекающем пользователя от основной задачи — совершения покупок.

Проведённый аналитический обзор прототипов и литературных источников позволил сформировать общую схему работы приложения, которая присуща интернет-магазинам данного рода, а именно интернет-магазинам книжной продукции.

Каждому пользователю должна предоставляться возможность найти интересующие его книги, получить подробную информацию о книге, её авторе, издательстве, узнать достоверную цену и самое главное - осуществить покупку книги. Кроме этого, система должна предоставлять администратору интерфейс для добавления новых книг, редактирования информации уже имеющихся, а также их удаления. Зарегистрированным пользователям в системе должна быть предоставлена возможность редактирования личной информации, работы с разделом любимых книг, а также возможность просмотра истории купленных книг.

## **1.2 Анализ требований к программному средству и разработка функциональных требований**

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

- бизнес-требования;
- пользовательские требования;
- функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

- простота и лёгкость интерфейса;
- единый стиль оформления интерфейса;
- использование архитектурных шаблонов проектирования;



– использование системы управления базами данных (СУБД).

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволят решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы.

Данное web-приложение подразумевает наличие нескольких ролей пользователей и разграничение прав, которые они имеют. Незарегистрированный пользователь данного программного решения должен иметь возможность:

- обратиться к ресурсу;
- зарегистрировать себя в системе и переходить в категорию зарегистрированных пользователей;
- просматривать данные об имеющихся в каталоге книгах;
- выполнять поиск книг, сортировку и фильтрацию;
- пользоваться функционалом корзины, а именно: добавлять книги в корзину и удалять.

Зарегистрированный пользователь в дополнение к разрешенным действиям незарегистрированного должен иметь возможность:

- входить в приложение, после ввода данных, необходимых для аутентификации;
- просматривать и редактировать личную информацию;
- пользоваться функционалом любимых книг, а именно: добавлять книги в любимые и удалять;
- осуществлять покупку книг, добавленных в корзину;
- просматривать историю покупок.

Администратор должен иметь возможность:

- входить в панель администратора, после ввода данных, необходимых для аутентификации;
- просматривать уже имеющиеся в каталоге книги;
- добавлять в каталог новые книги;
- редактировать информацию уже имеющихся книг;
- удалять книги из каталога.

Данные требования обобщены в виде диаграммы вариантов использования разрабатываемого программного средства, которая приведена в приложении А. Она отражает функциональность программного средства с точки зрения получения значимого результата для пользователя.

К последнему уровню требований относятся функциональные требования. Функциональные требования определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Данные требования могут быть описаны в виде утверждений, способов взаимодействия и методов реализаций. После проведения анализа были выявлены следующие функциональные требования:

- приложение должно разграничивать функционал для пользователей с разными ролями;

- приложение должно производить валидацию вводимых пользователем данных;

- приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;

- просмотр книг должен быть реализован в удобном виде с отображением графических данных;

- приложение должно выполнять поиск книг по названию или автору книги;

- приложение должно позволять выполнять сортировку и фильтрацию данных;

- приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;

- приложение должно предоставлять возможность пользователям проходить авторизацию и входить в систему под соответствующим введенным данным пользовательским именем;

- приложение должно предоставлять возможность администраторам добавлять книги с помощью отдельной формы, редактировать информацию о книгах, удалять уже имеющиеся в каталоге.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

## 2 Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ. Результатом данного раздела является обобщенная структура (архитектура) программного обеспечения.

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает:

- выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
- соединение выбранных элементов структуры и поведения во всё более крупные системы;
- архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

### 2.1 Обзор используемых технологий

Разработка данного приложения осуществлялась на языках C# и JavaScript. При реализации курсового проекта использовались технологии: ASP.Net Core, Entity Framework Core, MediatR, FluentValidation, NSwag и библиотеки React, React Bootstrap, React Router. MS SQL в качестве базы данных.

Для реализации серверной части web-приложения был выбран язык программирования C# с использованием технологии ASP.Net Core для обработки REST-запросов, работы с базой данных и выполнения основной бизнес-логики приложения.

Для работы с базой данных выбрана СУБД MS SQL, которая программно связана с бэкендом с помощью ORM технологии Entity Framework Core.

Entity Framework Core (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом, т. е. EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции. EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Для реализации клиентской части был выбран React, который предназначен только для работы с представлениями. Это решение представляет собой не платформу, а обычную библиотеку, поэтому для построения одностраничного приложения требуются дополнительные библиотеки. Существует несколько библиотек, которые можно использовать с React для создания многофункциональных одностраничных приложений.

Важной особенностью React является использование виртуальной модели DOM. Виртуальная модель DOM в React дает целый ряд преимуществ, в том числе производительность (виртуальная модель DOM может оптимизировать процесс обновления отдельных частей реальной модели DOM) и тестируемость (возможность тестировать библиотеку React и ее взаимодействие с такой виртуальной моделью DOM без использования браузера).

Кроме того, React реализует непривычный способ работы с HTML. Вместо строгого разделения между кодом и разметкой (например, со ссылками JavaScript в атрибутах HTML) React добавляет HTML непосредственно в код JavaScript в виде JSX. JSX — это синтаксис в стиле HTML, который может компилироваться в чистый код JavaScript.

Поскольку React не является полноценной платформой, для реализации таких возможностей, как маршрутизация, вызовы web-API и управление зависимостями, требуются другие библиотеки. Для маршрутизации была выбрана библиотека React Router, для вызовов API — Fetch API, который предоставляет интерфейс JavaScript для работы с запросами и ответами HTTP.

## 2.2 Проектирование архитектуры приложения

Перед тем как начать проектировать архитектуру приложения, необходимо определиться с типом разрабатываемого приложения. Для осуществления работы книжного магазина наиболее подходящим типом приложения является web-приложение.

На сегодняшний день существует два принципиальных подхода к созданию web-приложений: традиционные web-приложения, большая часть логики которых выполняется на сервере, а также одностраничные приложения, логика пользовательского интерфейса которых выполняется преимущественно в web-браузере, а взаимодействие с web-сервером осуществляется главным образом через web-API. Для данного продукта был выбран тип одностраничного приложения. Этот выбор можно аргументировать следующими пунктами:

- в приложении требуется полнофункциональный пользовательский интерфейс;
- в приложении должен предоставляться API для других внутренних или общедоступных клиентов.

При разработке архитектуры и проектировании данного программного решения важно придерживаться общих принципов проектирования:

- принцип разделения задач, под которым подразумевается разделение программного обеспечения на компоненты в соответствии с выполняемыми ими функциями;
- принцип инкапсуляции отдельных частей приложения, который позволяет изолировать их друг от друга;
- принцип инверсии зависимостей, которая является важной частью процесса создания слабо связанных приложений, так как детали реализации могут описывать зависимости и реализовывать абстракции более высокого уровня, а не компоненты того же уровня;

- принцип явных зависимостей, т.е. методы и классы должны явно требовать наличия всех совместно работающих объектов, которые необходимы для их корректного функционирования;

- принцип единственной обязанности, который подразумевает, что объекты должны иметь только одну обязанность и только одну причину для изменения;

- принцип «Не повторяйся», т.е. в приложении не следует определять поведение, связанное с конкретной концепцией, в нескольких расположениях, так как такой подход часто приводит к ошибкам;

- принцип независимости сохраняемости, который относится к типам, для которых требуется сохранение состояния, однако код которых не зависит от выбираемой для этих целей технологии;

- принцип ограниченных контекстов, который подразумевает разбиение приложения на отдельные концептуальные модули.

На сегодняшний день существует целый ряд архитектурных шаблонов, применяемых при проектировании приложений. К ним относятся такие шаблоны как клиент/сервер, многослойная архитектура, компонентная архитектура, архитектура, основанная на шине сообщений, и сервисноориентированная архитектура (service-oriented architecture, SOA).

Для проектирования книжного магазина была выбрана многослойная архитектура. Данная архитектура также известна и под другим названием — чистая архитектура.

В рамках чистой архитектуры центральным элементом приложения являются его бизнес-логика и модель. В этом случае бизнес-логика не зависит от доступа к данным или другим инфраструктурам, то есть стандартная зависимость инвертируется: инфраструктура и детали реализации зависят от ядра приложения. Это достигается путем определения абстракций или интерфейсов в ядре приложения, которые реализуются типами, определенными в слое инфраструктуры. Такую архитектуру обычно рисуют в виде серии окружностей с общим центром, которая внешне напоминает срез луковицы.

На рисунке 2.1 показан пример такого стиля представления архитектуры.

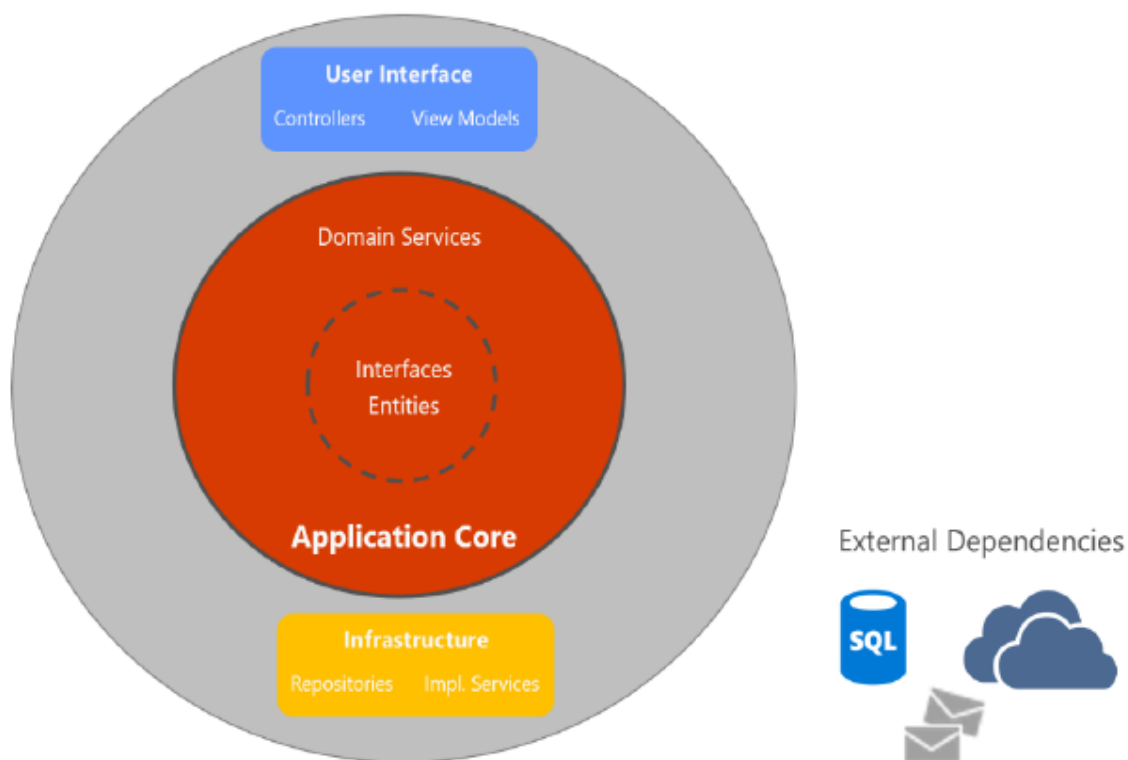


Рисунок 2.1 – Чистая архитектура (многослойное представление)

На этой схеме зависимости направлены из самой внутренней окружности. Ядро приложения называется так, потому что находится в самом центре этой схемы. Как видно на схеме, ядро приложения не имеет зависимостей от других слоев приложения. Сущности и интерфейсы приложения находятся в самом центре. Сразу после них, но все еще в пределах ядра приложения, расположены доменные службы, которые обычно реализуют интерфейсы, определенные во внутренней окружности. За пределами ядра приложения располагаются слои пользовательского интерфейса и инфраструктуры, которые зависят от ядра приложения, но не друг от друга (обязательно).

На рисунке 2.2 показана более привычная горизонтальная схема слоев, которая лучше отражает зависимости между слоем пользовательского интерфейса и другими слоями.

# Clean Architecture Layers

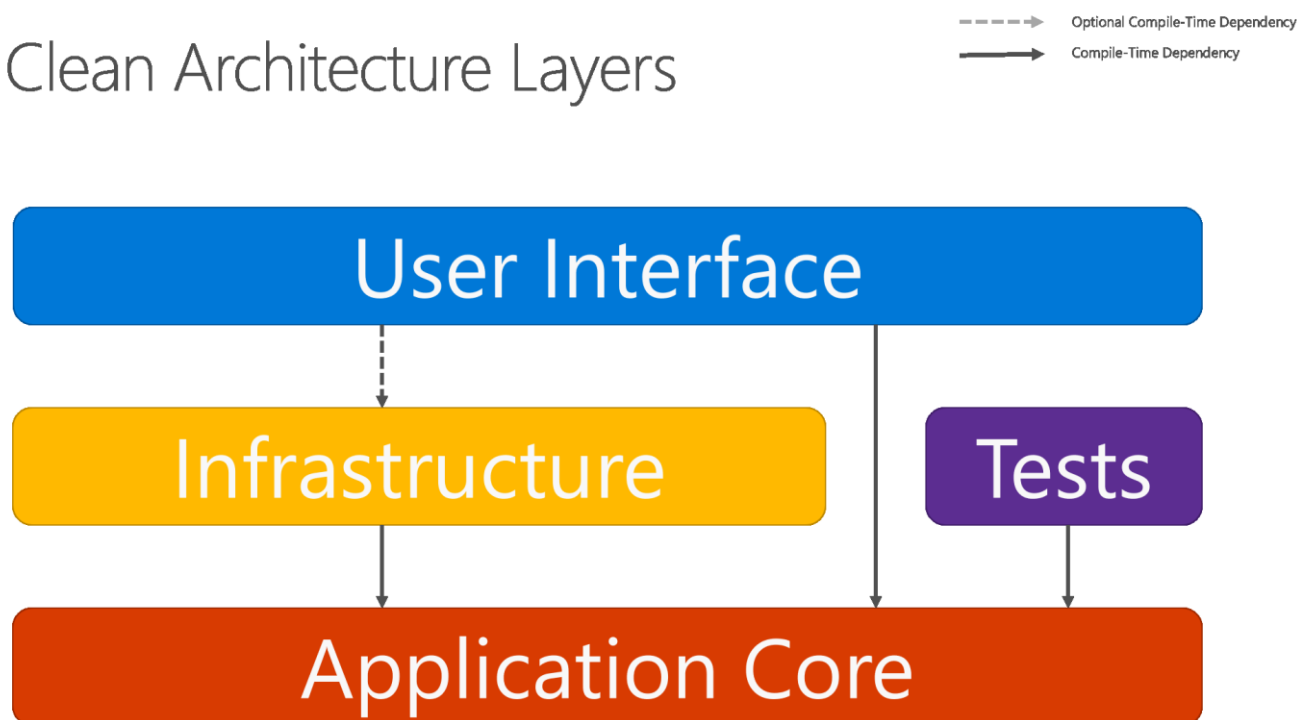


Рисунок 2.2 – Чистая архитектура (горизонтальное представление слоев)

На этой схеме сплошные стрелки соответствуют зависимостям времени компиляции, а пунктирные — зависимостям, которые существуют только во время выполнения. В рамках чистой архитектуры слой пользовательского интерфейса работает с интерфейсами, которые определены в ядре приложения во время компиляции, и в идеальном случае не должен знать ничего о типах реализации, определенных в слое инфраструктуры. Тем не менее во время выполнения эти типы реализации необходимы для выполнения приложения, поэтому они должны существовать и быть привязаны к интерфейсам ядра приложения посредством внедрения зависимостей.

## 2.3 Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

- обеспечение хранения в БД всей необходимой информации;
- обеспечение возможности получения данных по всем необходимым запросам;
- сокращение избыточности и дублирования данных;
- обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (дatalogическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. В результате этого этапа создаётся ER-

модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут.

Сущность — это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь — это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности — это любая деталь, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 9 сущностей:

- покупатель;
- книга;
- жанр;
- автор;
- издатель;
- обложка книги;
- заказ;
- детали заказа;
- любимая книга.

Также в ER-модели были определены необходимые связи. Например, между сущностями книга и обложка книги была установлена связь один-к-одному. Для каждой сущности были выделены атрибуты. Например, для сущности деталей заказа в качестве атрибутов были выделены такие характеристики, как идентификатор, количество книг, идентификатор заказа и идентификатор книги.

Логическое (дatalogическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных дatalogическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но не учитывается специфика конкретной СУБД. Дatalogическая модель базы данных представлена в приложении Б.



### 3 Реализация программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

#### 3.1 Реализация сущностей

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework Core.

Создание базы данных происходит из созданной вручную модели объектов. Созданные модели объектов частично совпадают с сущностями, которые были сформированы ранее в разделе 2.3. Каждая модель вынесена в отдельный файл, а уже все модели собраны в одной папке. Содержание папки Entities показано на рисунке 3.1.

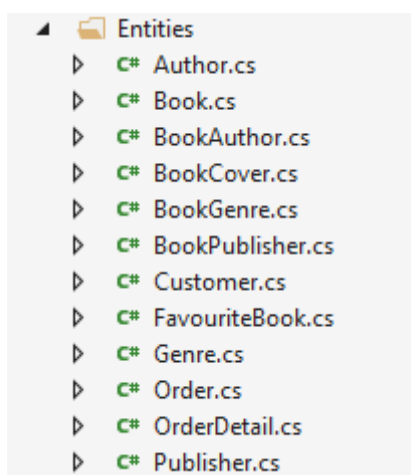


Рисунок 3.1 – Содержание папки Entities

Подробный код модели, соответствующей сущности книги, представлен в приложении В.

#### 3.2 Реализация серверной части

Серверная часть реализована с помощью кроссплатформенной технологии ASP.Net Core. Во время этапа проектирования было принято решение использовать архитектурный паттерн «чистая архитектура». Для реализации данной архитектуры серверная часть была представлена в виде решения, состоящего из шести отдельных проектов. Посмотрим на логическую структуру папок серверной части web-приложения. Она приведена на рисунке 3.2.

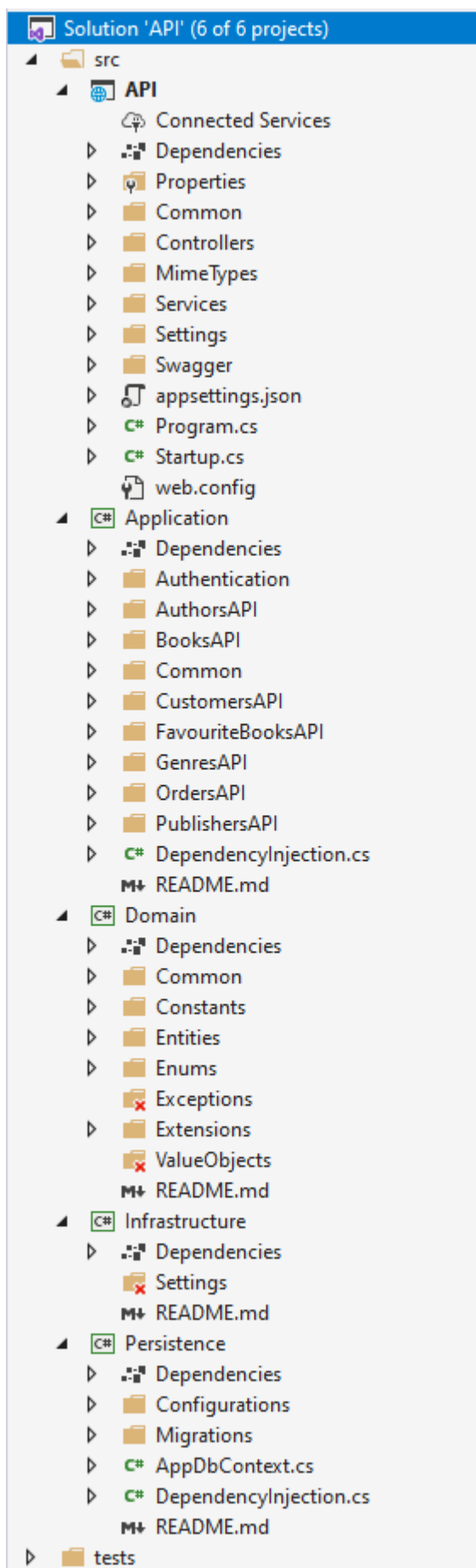


Рисунок 3.2 – Логическая структура папок серверной части

Имена проектов в папке src соответствуют слоям диаграммы чистой архитектуры, за исключением API, который представляет слой представления.

Проект Domain представляет уровень домена и включает сущности, перечисления, исключения, интерфейсы, типы и логику, специфичные для слоя домена. Этот слой полностью абстрагирован от внешних зависимостей.

Проект Application представляет собой слой приложения и содержит всю бизнес-логику. Этот проект реализует CQRS (разделение ответственности за запросы команд), при этом каждый вариант использования представлен одной командой или запросом. Этот слой зависит от слоя домена, но не зависит от других слоев или проектов. Этот уровень определяет интерфейсы, которые реализуются внешними слоями. Например, если приложению требуется доступ к службе уведомлений, в приложение будет добавлен новый интерфейс, а реализация будет создана в инфраструктуре.

Проект Infrastructure представляет собой слой инфраструктуры и содержит классы для доступа к внешним ресурсам, таким как файловые системы, web-службы, SMTP и т. д. Эти классы основаны на интерфейсах, определенных на слое приложения.

Проект Persistence представляет собой слой для доступа к базе данных. Этот слой конфигурирует DbContext, используемый в приложении.

Проект API представляет собой слой представления. Этот слой содержит проект, с которым пользователь может взаимодействовать. В контексте нашего приложения это означает, что он принимает входные данные в форме HTTP-запросов по сети (например, GET / POST / и т. д.) и возвращает свои выходные данные в виде контента, отформатированного как JSON.

Папка tests содержит множество модульных и интеграционных тестов.

Помимо ASP.NET Core, в этом решении используется множество технологий, в том числе:

- CQRS с помощью MediatR;
- валидация с помощью FluentValidation;
- сопоставление объектов с объектами с помощью AutoMapper;
- доступ к данным с помощью Entity Framework Core;
- API с использованием ASP.NET Core;
- спецификация OpenAPIc NSwag;
- безопасность с использованием ASP.NET Core Identity и IdentityServer;
- автоматизированное тестирование с помощью xUnit.net, Moq и Shouldly.

### 3.3 Реализация клиентской части

Клиентская часть разработана с помощью JavaScript-библиотеки React. React – это декларативная, эффективная и гибкая библиотека для построения пользовательских интерфейсов. Разработанный в этом курсовом проекте клиент представляет собой одностраничное приложение (SPA).

SPA-приложение, Single Page Application, или «приложение одной страницы» – это тип web-приложений, в которых загрузка необходимого кода происходит на

одну страницу, что позволяет сэкономить время на повторную загрузку одних и тех же элементов. Жизненный цикл SPA представлен на рисунке 3.3.

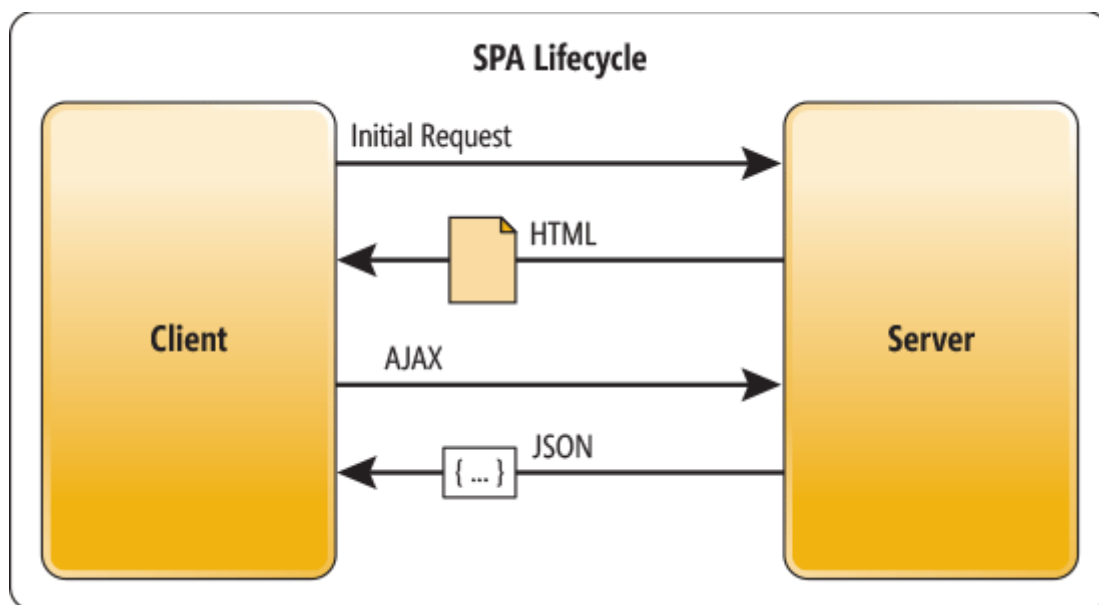


Рисунок 3.3 – Жизненный цикл SPA

Особенность архитектуры SPA заключается в том, что все элементы, необходимые для работы приложения: элементы CSS, скрипты, стили и прочее на одной странице. Они загружаются при инициализации. Также данный вид приложений загружает дополнительные модули после запроса от пользователя. Любая пользовательская активность фиксируется для удобства навигации. Это позволяет скопировать ссылку и открыть софт на том же этапе взаимодействия на другой вкладке, браузере или устройстве.

При загрузке новых модулей в SPA контент на них обновляется только частично, так как элементам, не нуждающимся в изменении, нет необходимости загружаться повторно, замедляя тем самым скорость ответа и передаваемый объем данных между браузером и сервером.

Теперь посмотрим на логическую структуру папок клиентской части web-приложения. Она приведена на рисунке 3.4.

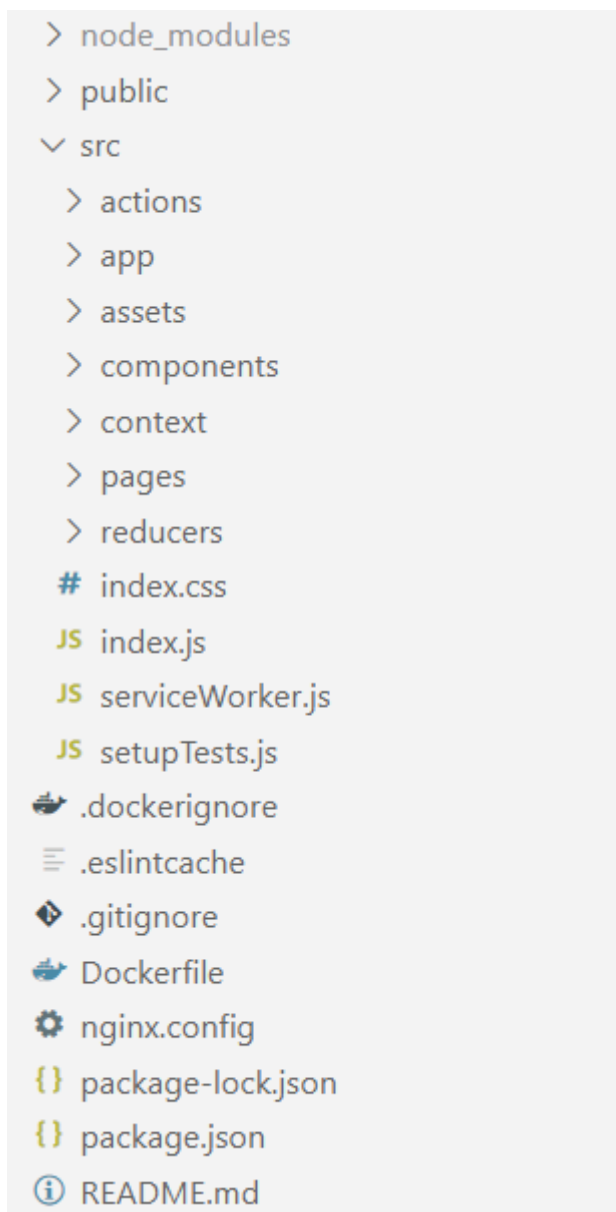


Рисунок 3.4 – Логическая структура папок клиентской части

Самое главное расположено в папке `src`. Пройдёмся по папкам, вложенным в эту папку.

В папке `actions` содержатся файлы с функциями необходимыми для обращения к API. Для запросов к серверу используется самостоятельно написанная обёртка для `Fetch API`.

Папка `app` содержит в себе главный компонент `App`, представляющий React-приложение. Отдельно выделен компонент, представляющий часть приложения, доступную только администраторам.

Папка `assets` предназначена для хранения различных графических ресурсов, используемых в приложении.

В папке `pages` расположены компоненты, представляющие собой страницы.

Папка `components` содержит все компоненты, используемые на различных страницах приложения.

Папка context хранит файлы с контекстами, используемыми в данном клиенте. В папке reducers находятся редюсеры.

Стоит отметить, что для стилизации приложения, был использован фреймворк React Bootstrap. Он содержит стили для основных элементов, которые применяются в верстке. Использование такого фреймворка значительно ускоряет процесс создания страниц. Стандартные стили легко менять, что обеспечивает гибкий и простой процесс создания макетов сайтов.

### **3.4 Реализация аутентификации и авторизации**

Аутентификация – это процедура проверки подлинности идентификации пользователя.

Аутентификация реализована с использованием JSON Web Token (JWT). JWT состоит из трех основных частей: заголовка (header), нагрузки (payload) и подписи (signature). Заголовок и нагрузка формируются отдельно, а затем на их основе вычисляется подпись.

Опишем схему работы аутентификации с использованием JWT. После первого логина, клиенту возвращается сгенерированный сервером JWT. При каждом следующем запросе, клиент должен передавать JWT установленным API способом. Сервер декодирует header и payload и проверяет зарезервированные поля. Если все в порядке, по указанному в header алгоритму составляется подпись. Если полученная подпись совпадает с переданной, пользователя авторизуют.

Для работы с JWT-токенами был применён Nuget пакет Microsoft.AspNetCore.Authentication.JwtBearer. Для встраивания функциональности JWT-токенов в конвейер обработки запроса использовался компонент JwtBearerAuthenticationMiddleware.

Авторизация – это процедура проверки прав аутентифицированного пользователя.

Данное web-приложение поддерживает три типа пользователей, а именно: гость, зарегистрированный пользователь и администратор. Каждый тип пользователя имеет свои определенные права. Ассоциация каждого пользователя с ролью осуществляется путем добавления к информации о пользователе в базе данных информации о роле. Роль, присущая пользователю, записана в JWT.

## 4 Тестирование, проверка работоспособности и анализ полученных результатов

Прежде всего были проведены тесты аутентификации: проверка на пустые и неверные данные. Валидация выполнена таким образом, что после нажатия на кнопку входа или регистрации выполняется проверка: в случае валидных данных они отправляются на сервер, в случае невалидных - данные с клиента не будут отправлены на сервер и будут выведены соответствующие сообщения об ошибках. При таком способе валидации шанс отправить невалидные данные на сервер минимален. Проверка работоспособности валидации на формах входа и регистрации представлена на рисунке 4.1.

The image shows a web application interface with two main sections: 'Sign In' and 'Join'. The 'Sign In' section has two input fields: 'Email address' and 'Password'. Both fields have red borders and red error messages below them: 'email is a required field' and 'password is a required field'. Below these fields is a blue button labeled 'Sign in'. Underneath the button is the text 'OR' and three social media icons: Google, Twitter, and Facebook. The 'Join' section has four input fields: 'First Name', 'Last Name', 'Email address', and 'Password'. Each field has a red border and a red error message below it: 'firstName is a required field', 'lastName is a required field', 'email is a required field', and 'password is a required field'. Below these fields is a blue button labeled 'Create your account'. The top of the page has a navigation bar with links 'All books', 'In stock', and 'Out of stock', and a shopping cart icon with '0 \$' and '0' items. The bottom of the page has a footer with the text '© 2020 Book Libraria I All rights reserved'.

Рисунок 4.1 – Валидация форм входа и регистрации

Кроме валидации полей, необходимо обрабатывать исключительные ситуации, возникающие на сервере. Таковыми, например, являются – попытка входа незарегистрированного в системе пользователя, попытка входа с неверным паролем, попытка регистрации уже зарегистрированного пользователя. Такие ошибки выводятся в виде уведомления. Пользователь имеет возможность закрыть данное уведомление и продолжить работу. Уведомление содержит в себе понятное для пользователя объяснение возникшей исключительной ситуации, чтобы в последствии устранить её.

Результат обработки исключительной ситуации, связанной с попыткой аутентификации незарегистрированного в системе пользователя, приведен ниже на рисунке 4.2.

Рисунок 4.2 – Попытка входа без регистрации

Результат обработки исключительной ситуации, связанной с попыткой аутентификации, используя неверный пароль, приведен на рисунке 4.3.

Рисунок 5.3 – Вход с неверным паролем

Результат обработки исключительной ситуации, связанной с попыткой регистрации нового пользователя, используя данные, принадлежащие уже имеющемуся пользователю в системе, приведен на рисунке 4.4.



The screenshot shows the Book Livraria website interface. At the top, there is a dark purple header with the Book Livraria logo, a search bar, and a 'Sign in/Join' button. Below the header, there are navigation links: 'All books', 'In stock', and 'Out of stock'. A red error message banner at the top of the main content area reads: 'You got an error! User with such email is already exist in system. Use Sign In form.' Below this, there are two forms: 'Sign In' and 'Join'. The 'Sign In' form has fields for 'Email address' and 'Password', a 'Sign in' button, and social media login options (Google, Twitter, Facebook). The 'Join' form has fields for 'First Name', 'Last Name', 'Email address', and 'Password'.

Рисунок 4.4 – Попытка регистрации уже существующего пользователя

Также были проведены тесты на работу валидации внутри приложения. Наличие валидации является обязательным в связи с тем, что в приложении постоянно ведётся работа с базой данных.

Валидация организована таким же образом, как и на формах регистрации и входа, т.е. после нажатия на кнопку отправки формы выполняется проверка: в случае валидных данных они отправляются на сервер, в случае невалидных - данные с клиента не будут отправлены на сервер и будут выведены соответствующие сообщения об ошибках. Все ошибки, возникшие при валидации, доступны и отображены удобным образом, что позволяет пользователю легко понять какие данные не валидны и быстро исправить их. Пример результата валидации данных внутри приложения приведен на рисунке 4.5.

The screenshot shows a form for adding a new book. The form is divided into two columns. The left column contains fields for 'ISBN', 'Title', 'Author', 'Language', 'Format', and 'Price'. The right column contains fields for 'Genres', 'Publisher', 'Publication date', 'Pages', and 'Quantity'. Each field has a red border and a red error message below it: 'Permission type is required'. A 'Save' button is located at the top right. On the left side of the form, there is a large text area that says 'No image available'.

Рисунок 4.5 – Демонстрация валидации внутри приложения

На данном этапе были выполнены тесты на проверку работоспособности приложения, а именно на проверку валидации. Были проведены анализы результатов, которые показали, что валидация в приложении работает в соответствии с тем, как она задумывалась.

## 5 Руководство пользователя

Главная страница даёт пользователю возможность перейти на любую из возможных страниц сайта. Её центральным элементом является панель, которая отображает 6 последних добавленных в каталог книг. Интерфейс главной страницы приведён ниже на рисунке 5.1.

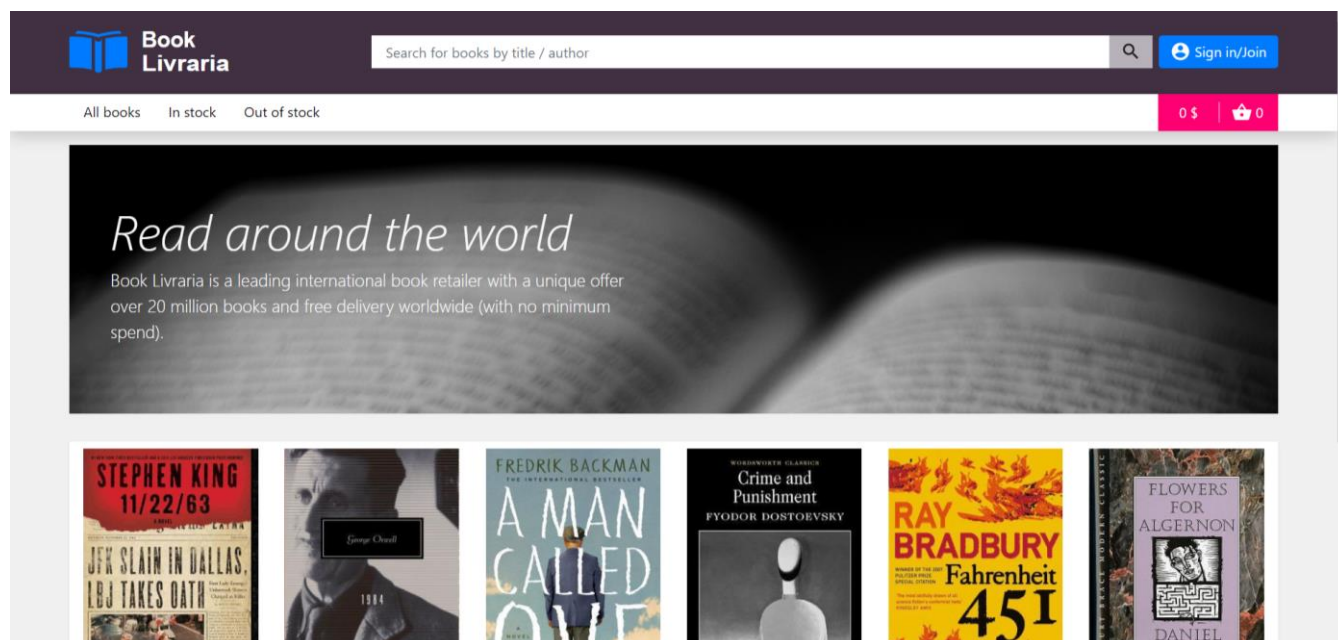


Рисунок 5.1 – Главная страница

Все страницы сайта имеют общий макет, состоящий из шапки и подвала. В шапке сайта расположен логотип, через который можно перейти на главную страницу, поисковое поле и в зависимости от типа пользователя либо кнопка перехода на страницу входа, либо кнопки для перехода в личный кабинет и для выхода из аккаунта. Интерфейс страницы входа в систему приведён на рисунке 5.2.

The screenshot shows the top navigation bar of the Book Livraria website. It includes the logo, a search bar with the placeholder text "Search for books by title / author", and a "Sign in/Join" button. Below the navigation bar, there are links for "All books", "In stock", and "Out of stock". On the right side of the header, there is a shopping cart icon showing "0 \$".

The main content area is divided into two columns. The left column is titled "Sign In" and contains fields for "Email address" and "Password", followed by a blue "Sign in" button. Below this, there is an "OR" separator and three social media icons (Google, Twitter, Facebook). The right column is titled "Join" and contains fields for "First Name", "Last Name", "Email address", and "Password", followed by a blue "Create your account" button.

At the bottom of the page, there is a footer with the text "© 2020 Book Livraria | All rights reserved".

Рисунок 5.2 – Страница для входа или регистрации

Если вход или регистрация успешна происходит перенаправление на страницу с личным кабинетом. Личный кабинет позволяет просматривать и редактировать пользовательскую информацию. Интерфейс страницы редактирования профиля представлен на рисунке 5.3.

The screenshot shows the "My Account" page of the Book Livraria website. The left sidebar contains a menu with "Personal Info", "Favourites", and "Order History". The main content area is titled "Personal info" and contains a form with the following fields:

- First name: Yuliya
- Last name: Maximchikova
- Email: m.yuliyas.00@gmail.com
- Country: (empty field)
- City: (empty field)
- Street: (empty field)
- Building No.: (empty field)
- Flat No.: (empty field)
- Postal Code: (empty field)
- Phone: +375 (empty field)

At the bottom of the page, there is a footer with the text "© 2020 Book Livraria | All rights reserved".

Рисунок 4.3 – Страница редактирования персональной информации пользователя

Кроме этого, в личном кабинете есть возможность просматривать список книг, добавленных в «Любимые». На этой странице пользователь может удалить книгу из раздела «Любимые» или добавить книгу в корзину. Данная страница представлена на рисунке 5.4.

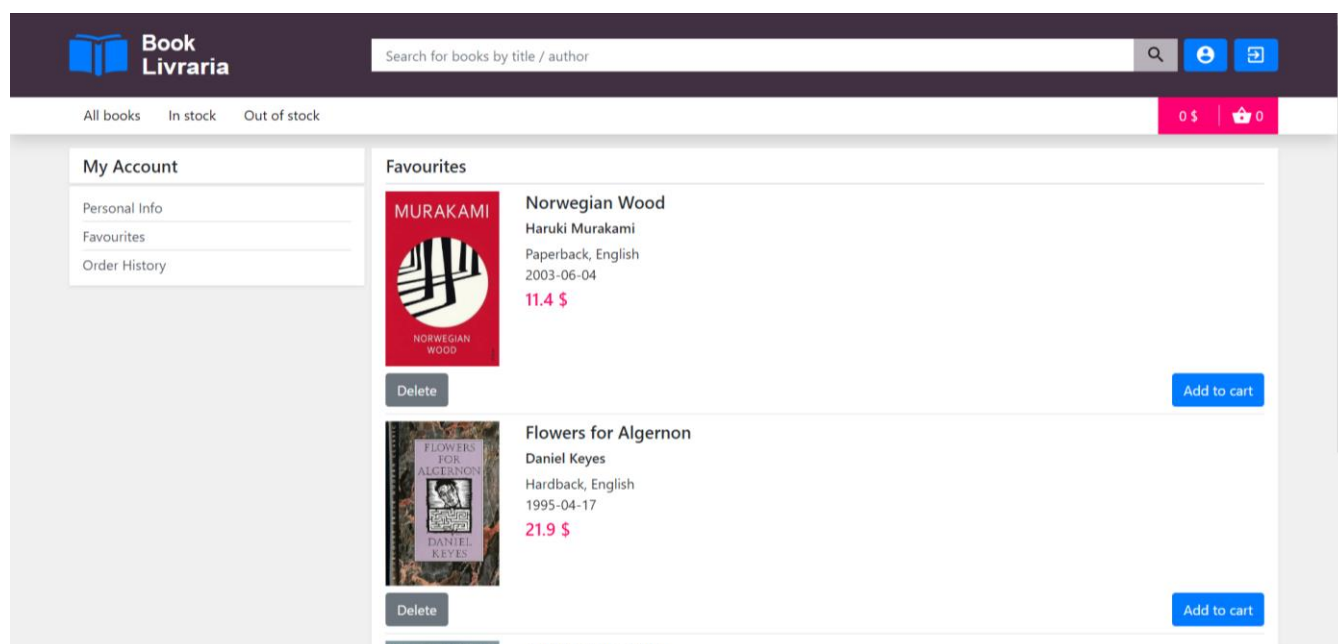


Рисунок 5.4 – Раздел «Любимые»

Последним функционалом, предоставляемым личным кабинетом, является просмотр истории покупок. Интерфейс страницы для просмотра истории покупок представлен ниже на рисунке 5.5.

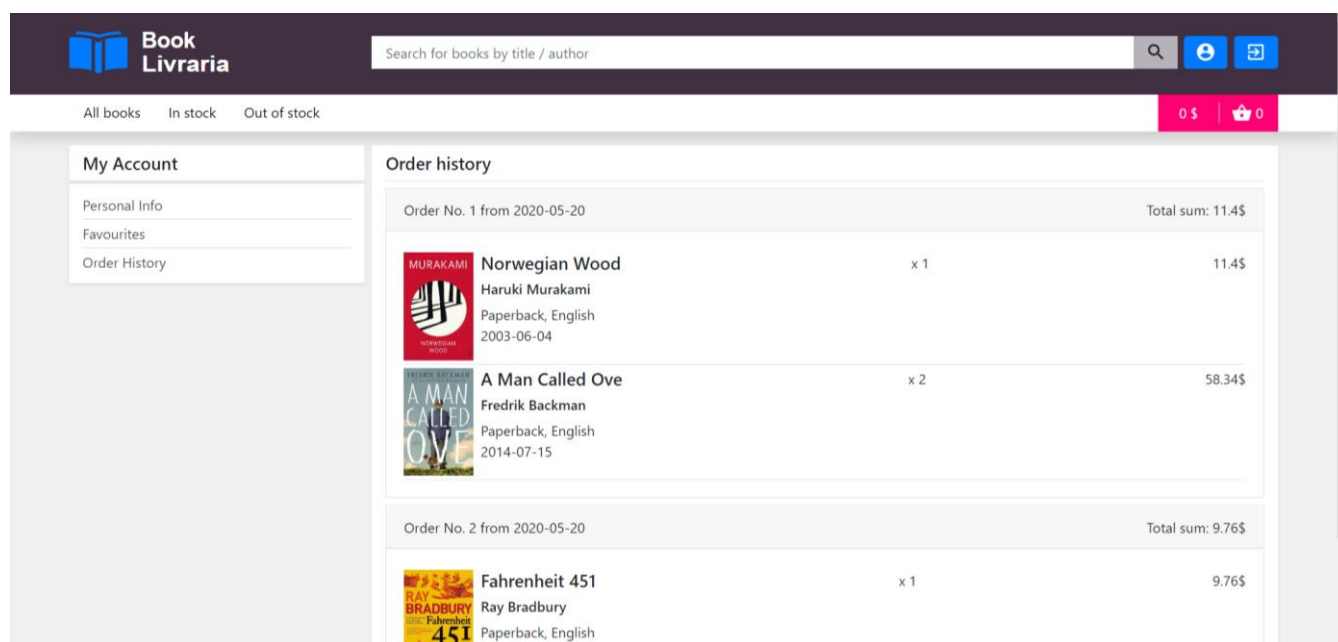


Рисунок 5.5 – История покупок

Перейдём к более интересной странице, а именно странице с каталогом имеющихся книг в магазине. Данная страница предоставляет широкий функционал работы. На этой странице можно выполнить поиск книг по автору или названию, фильтрацию книг по цене, жанру или доступности, а также сортировку по дате публикации или цене. На странице отображаются не все книги из базы данных, а лишь ограниченное количество. Это позволяет не замедлять загрузку страницы.

Чтобы просматривать остальные книги, выполнена пагинация. Данная страница показана на рисунке 5.6.

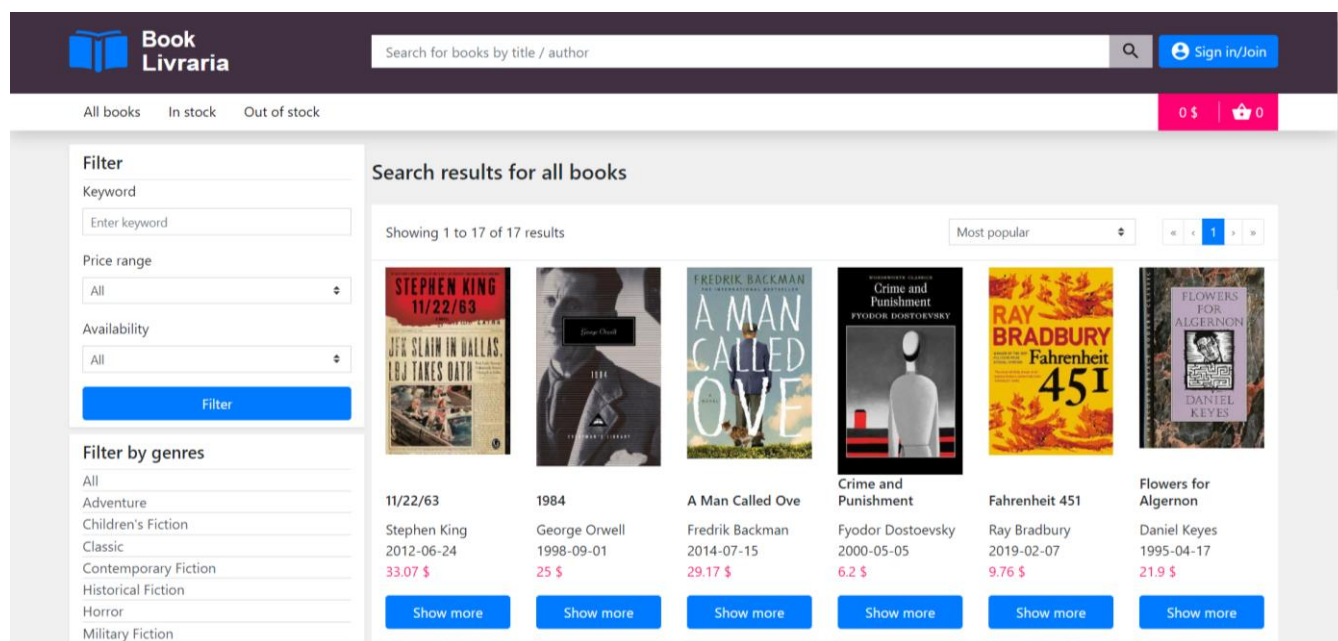


Рисунок 5.6 – Каталог книг

Из каталога книг, нажав на кнопку «Show more», мы переходим на страницу с описанием книги. Она весьма информативна, так как отображает всю имеющуюся информацию о книге. Также на этой странице можно добавить книгу в корзину. Интерфейс страницы приведён ниже на рисунке 5.7.

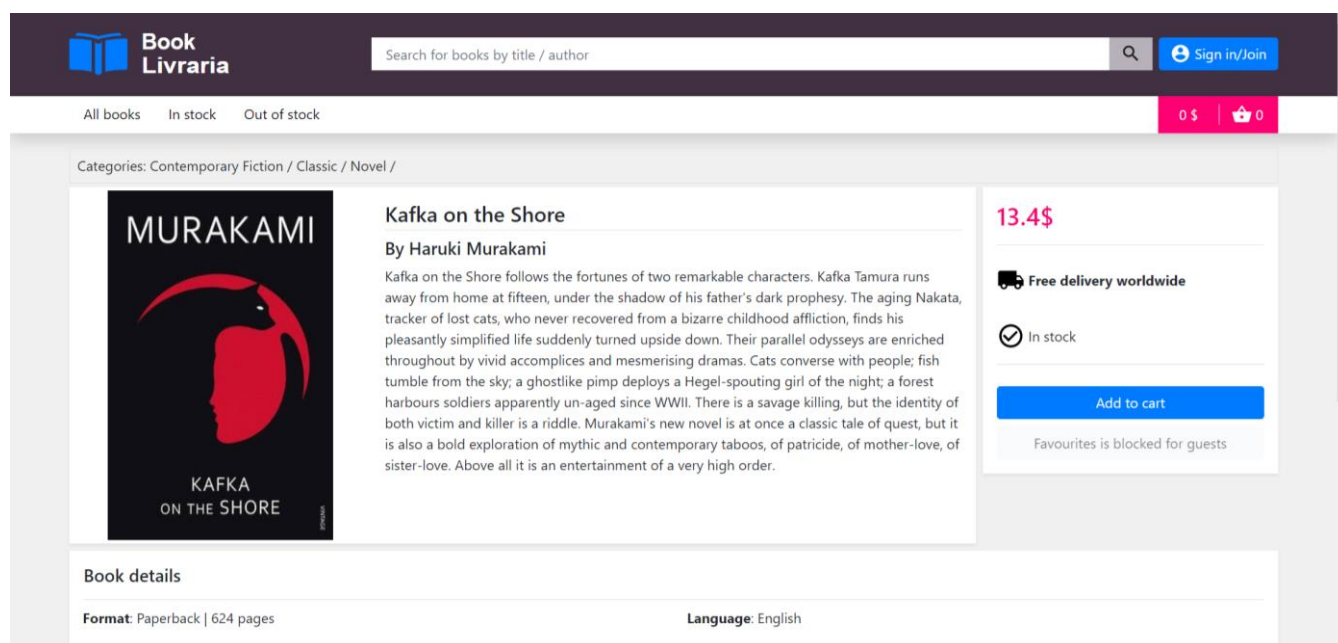


Рисунок 5.7 – Страница просмотра информации о книге

Посмотрим непосредственно на страницу с самой корзиной. На ней мы можем увеличить или уменьшить количество каждой книги в корзине, вообще удалить книгу из корзины или перейти к оформлению покупки. Оформить покупку

возможно лишь в том случае, если пользователь авторизован. Интерфейс страницы приведён ниже на рисунке 5.8.

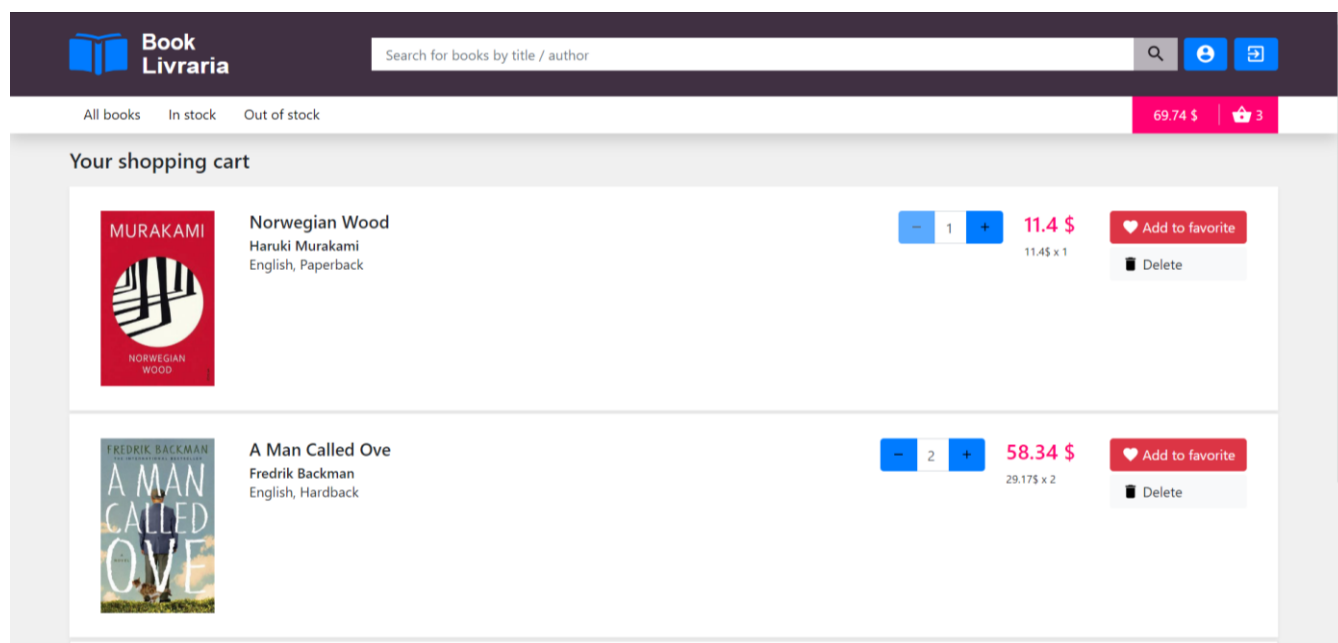


Рисунок 5.8 – Корзина

Страница для оформления покупки состоит из формы для заполнения информацией о покупателе и панели с данными о покупке. Её интерфейс приведён ниже на рисунке 5.9.

The screenshot shows the checkout page on the Book Livraria website. The left side contains a form for customer information, and the right side shows an order summary.

**Fill with valid information**

First name: Yuliya

Last name: Maximchikova

Email: m.yuliyas.00@gmail.com

Country: Belarus City: Gomel

Street: Pinskaya Building: 20 Flat: 1

Postal code: 246027

**Order Summary**

Item	Price
11/22/63 Stephen King x 2	66.14 \$
Crime and Punishment Fyodor Dostoevsky x 3	18.6 \$
Sub-total	84.74 \$
Delivery	Free
Total	84.74 \$

Рисунок 5.9 – Оформление покупки

Перейдём к части сайта, которая доступна только пользователям с ролью администратора. Переход к данной части приложения выполнен секретным путём. Это одно из решений для сокращения попыток несанкционированного доступа к

панели администратора. Чтобы авторизоваться как администратор существует отдельная страница входа. Она представлена на рисунке 5.10.

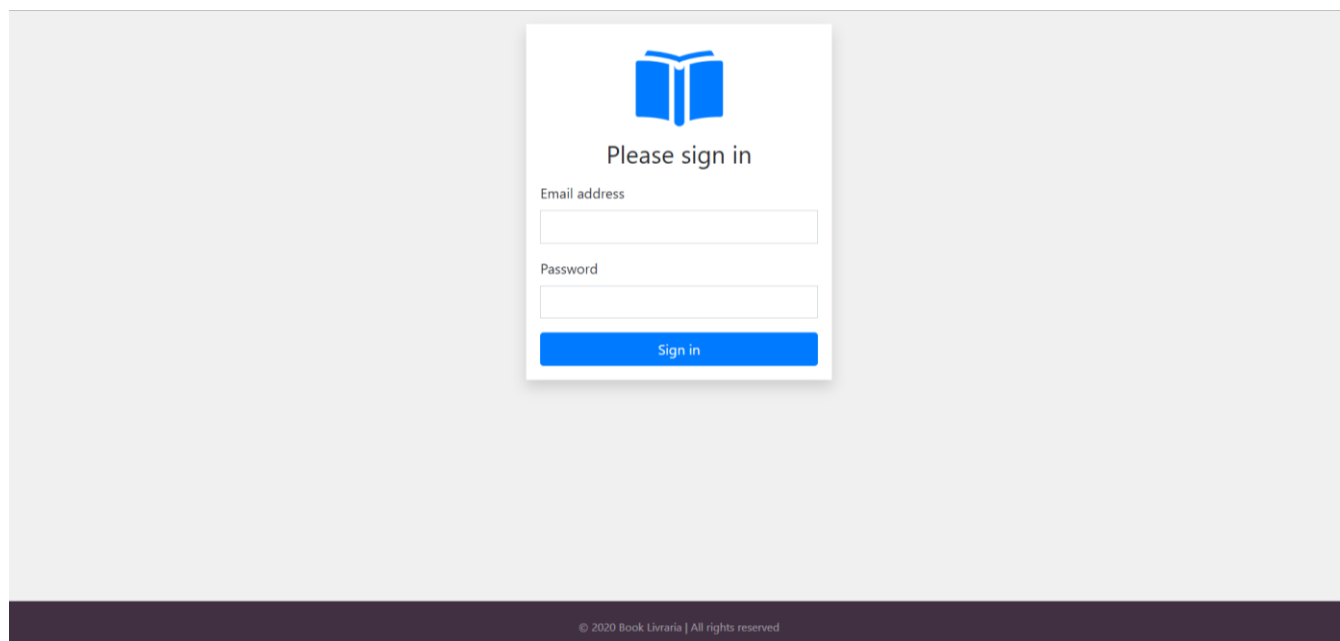


Рисунок 5.10 – Страница входа в панель администратора

В случае успешной аутентификации администратор попадает на страницу с панелью администратора. Все страницы для администратора также имеют общий макет, а именно шапку и подвал. На шапке расположена одна единственная кнопка, с помощью которой администратор может выполнить выход. С помощью панели администратора можно просмотреть все имеющиеся в каталоге книги. Они оформлены в виде таблицы с пагинацией, чтобы не замедлять загрузку страницы. На этой таблицы ISBN у книг являются ссылками, нажав на такую ссылку выполняется переход на страницу редактирования книги. Также с помощью данной страницы можно перейти на страницу добавления новой книги. Интерфейс панели администратора представлен на рисунке 5.11.



The screenshot shows the 'Books' administration panel. At the top, a dark purple header contains the title 'Books' and a subtitle 'You currently have 17 in the catalog!'. Below the header, the 'Overview' section features four summary cards: '17 ALL BOOKS' (blue), '15 IN STOCK' (orange), '0 OUT OF STOCK' (red), and '+ NEW BOOK' (green). The main section, 'Books (17)', displays a table with 17 books. Each row includes the ISBN, Title, Author, Publication Date, Status, and Price.

ISBN	TITLE	AUTHOR	PUBLICATION DATE	STATUS	PRICE
9781451627299	11/22/63	Stephen King	2012-06-24	In stock 314	33.07\$
9780679417392	1984	George Orwell	1998-09-01	In stock 212	25\$
9781476738017	A Man Called Ove	Fredrik Backman	2014-07-15	In stock 237	29.17\$
9781840224306	Crime and Punishment	Fyodor Dostoevsky	2000-05-05	In stock 37	6.2\$
9780006546061	Fahrenheit 451	Ray Bradbury	2019-02-07	In stock 73	9.76\$
9780151001637	Flowers for Algernon	Daniel Keyes	1995-04-17	In stock 195	21.9\$
9780552996099	Forrest Gump	Winston Groom	1995-01-01	In stock 57	10.5\$
9780099494096	Kafka on the Shore	Haruki Murakami	2005-10-06	In stock 97	13.4\$

Рисунок 5.11 – Главная страница панели администратора

На странице добавления новой книги расположена форма, с помощью которой, введя валидную информацию и нажав на кнопку «Save», можно добавить книгу в каталог. Данная страница показана на рисунке 5.12.

The screenshot shows the 'New book' form. The header is dark purple with the title 'New book' and the instruction 'Fill form with valid data'. The main content area is titled 'Add new book'. On the left, there is a placeholder for a book image with the text 'No image available'. The form itself consists of several input fields and dropdown menus arranged in two columns. A blue 'Save' button is located in the top right corner of the form area.

Add new book	
ISBN	Genres
<input type="text" value="ISBN"/>	<input type="text" value="Select..."/>
Title	
<input type="text" value="Title"/>	
Author	Publisher
<input type="text" value="Select..."/>	<input type="text" value="Select..."/>
Language	Publication date
<input type="text" value="Language"/>	<input type="text" value="Publication date"/>
Format	Pages
<input type="text" value="Format"/>	<input type="text" value="Pages"/>
Price	Quantity
<input type="text" value="Price"/>	<input type="text" value="Quantity"/>

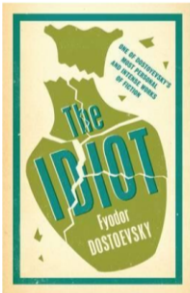
Рисунок 5.12 – Страница добавления новой книги в каталог

Страница для редактирования данных о книге имеет такой же внешний вид, как и страница для добавления новой книги. Отличие лишь в том, что форма уже заполнена данными и что кроме кнопки «Save» на форме присутствует кнопка для удаления книги из каталога. Страница для редактирования информации о книге представлена на рисунке 5.13

Book

ISBN: 9781847493439

Update book



ISBN

9781847493439

Genres

Classic x Novel x

Title

The Idiot

Author

Fyodor Dostoevsky

Publisher

Alma Books Ltd

Language

English

Publication date

2018-05-30

Format

Paperback

Pages

704

Price

Quantity

Save

Рисунок 5.13 – Страница редактирования информации о книге

На этом руководство пользователя заканчивается, так как были рассмотрены все страницы и весь описанный функционал web-приложения.

## 6 Руководство программиста

В данной главе будут представлены шаги, описывающие как развернуть приложение. Разработанное web-приложение было развёрнуто на платформе Docker.

Docker – платформа для разработки, доставки и эксплуатации приложений. Основное назначение – упростить развертывание приложения.

Нам понадобится Docker последней версии. В Docker Hub заранее были загружены images, поэтому понадобится лишь загрузить их при помощи команды `docker pull`. На рисунке 6.1 представлена загрузка image для клиентской и серверной части приложения.

```
user@DESKTOP-KMJFC9G:~$ docker pull 25082000/bookstore-backend
Using default tag: latest
latest: Pulling from 25082000/bookstore-backend
Digest: sha256:b1fe5499635aa4b3718af4c331d09d641b0cc2df9f9ea367a81dff2b653844fa
Status: Image is up to date for 25082000/bookstore-backend:latest
docker.io/25082000/bookstore-backend:latest
user@DESKTOP-KMJFC9G:~$ docker pull 25082000/bookstore-frontend
Using default tag: latest
latest: Pulling from 25082000/bookstore-frontend
Digest: sha256:f503a68bcc32d7b71b655fceb991e227d348ea7d51fd7dea2fadcd3fc5993bc3
Status: Image is up to date for 25082000/bookstore-frontend:latest
docker.io/25082000/bookstore-frontend:latest
```

Рисунок 6.1 – Загрузка image из Docker Hub

Для проверки, что images были загружены успешно, можно использовать команду `docker images`. Результат работы команды представлен на рисунке 6.2.

```
user@DESKTOP-KMJFC9G:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
25082000/bookstore-frontend	latest	91437c98c520	2 days ago	33.2MB
25082000/bookstore-backend	latest	bc353cdf3c83	2 days ago	257MB
mcr.microsoft.com/dotnet/core/sdk	3.1-buster	0b14e54e08b0	2 days ago	710MB
mcr.microsoft.com/dotnet/core/aspnet	3.1-buster-slim	96df76fbb586	3 days ago	207MB
nginx	alpine	0fde4fb87e47	3 days ago	22.3MB
node	alpine	f4268de957a6	4 days ago	109MB
mcr.microsoft.com/mssql/server	2019-latest	5ced205176bc	2 months ago	1.43GB
mcr.microsoft.com/mssql/server	latest	5ced205176bc	2 months ago	1.43GB

Рисунок 6.2 – Результат команды `docker images`

Таким образом наше приложение состоит уже как минимум из двух контейнеров, а ещё не стоит забывать про контейнер для образа базы данных, поэтому для запуска и конфигурирования нескольких контейнеров была использована команда `compose`.

Команда `compose` – это инструмент для Docker, используемый для определения и запуска нескольких приложений-контейнеров, в которых файл `compose` используется для определения необходимых для приложения сервисов. В листинге 6.1 приведено содержимое файла `docker-compose.yml`.

```

version: '3.9'

services:
  bookstore-database:
    image: "mcr.microsoft.com/mssql/server"
    environment:
      SA_PASSWORD: "Password12345!"
      ACCEPT_EULA: "Y"
    container_name: database
  bookstore-backend:
    image: 25082000/bookstore-backend
    build:
      context: ./bookstore-backend
      dockerfile: API/Dockerfile
    container_name: backend
    restart: on-failure
    ports:
      - "45763:80"
      - "45764:443"
    depends_on:
      - bookstore-database
  bookstore-frontend:
    image: 25082000/bookstore-frontend
    build:
      context: ./bookstore-frontend
      dockerfile: Dockerfile
    container_name: frontend
    ports:
      - "3001:80"
    depends_on:
      - bookstore-backend

```

Листинг 6.1 – Содержимое файла docker-compose.yml

Для запуска контейнера используется команда `docker compose up`. Результатом работы данной команды является работающее приложение, которое было развёрнуто с помощью Docker. Проверить успех создание контейнера приложения можно с помощью графического интерфейса программы Docker Desktop. В случае успеха на вкладке Containers/Apps будет отображен наш контейнер. Содержимое данной вкладки представлено на рисунке 6.3.

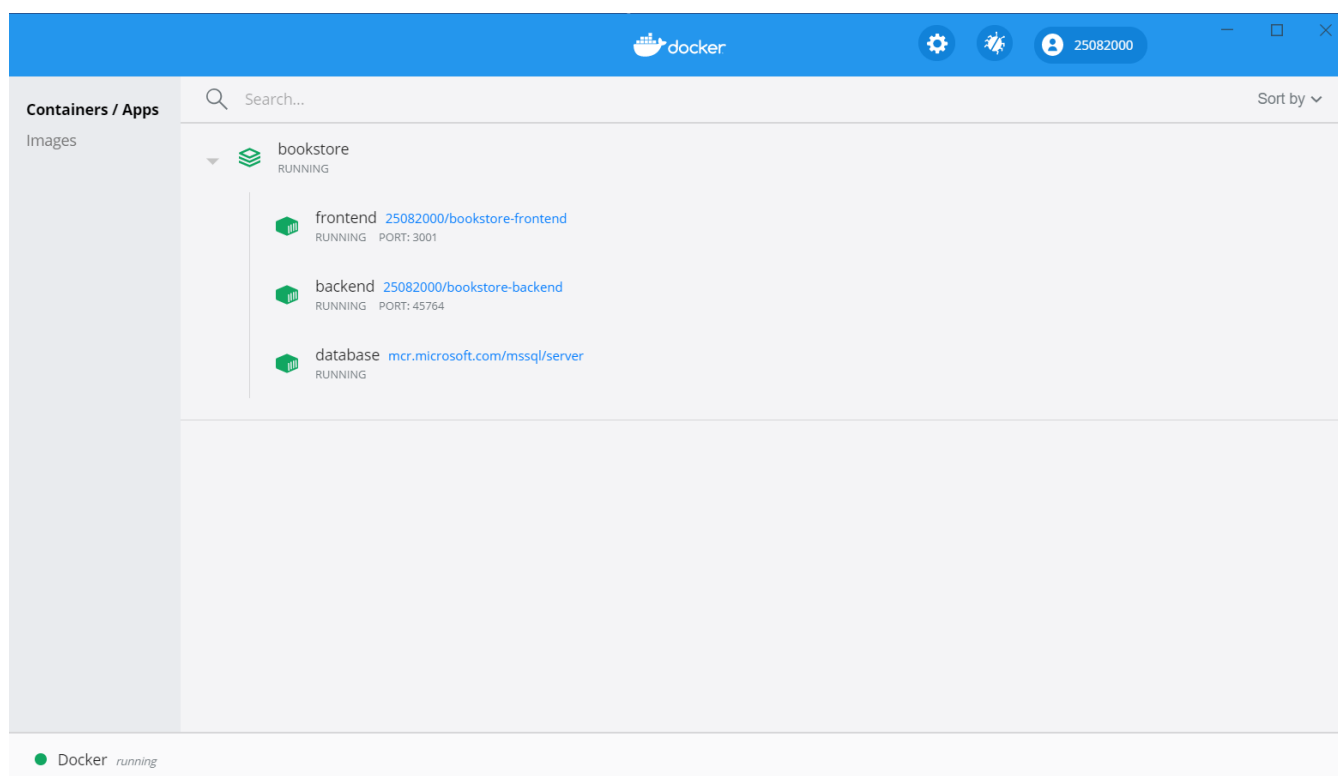


Рисунок 6.3 – Containers/Apps в Docker Desktop

Для формирования image необходимо было создать Dockerfile, который использовал следующие ключевые слова:

- FROM – «из чего» собираем наш image;
- WORKDIR – рабочая дериктория;
- EXPOSE – сообщает Docker о том, что контейнер прослушивает указанные сетевые порты во время выполнения;
- RUN – запуск команды;
- COPY – копироание файлов;
- ENTRYPOINT – команда, которая будет выполнена при запуске.

В листинге 6.2 находится исходный код Dockerfile для ASP.NET Core приложения.

```
FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-
slim AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build
WORKDIR /src
COPY ["API/API.csproj", "API/"]
COPY ["Persistence/Persistence/Persistence.csproj", "Persistenc
e/Persistence/"]
COPY ["Application/Application.csproj", "Application/"]
COPY ["Domain/Domain.csproj", "Domain/"]
COPY ["Infrastructure/Infrastructure.csproj", "Infrastructure/"]
]
```

```

RUN dotnet restore "API/API.csproj"
COPY . .
WORKDIR "/src/API"

RUN chmod +x "entrypoint.sh"
CMD /bin/bash "entrypoint.sh"

RUN dotnet build "API.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "API.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "API.dll"]

```

### Листинг 6.2 – Dockerfile для ASP.NET Core приложения

В листинге 6.3 находится исходный код Dockerfile React приложения.

```

FROM node:alpine AS builder
WORKDIR /bookstore-frontend
COPY package.json package-lock.json ./
RUN npm install
ENV PATH="./node_modules/.bin:$PATH"
COPY . ./
RUN npm run build

FROM nginx:alpine
RUN apk --no-cache add curl
RUN curl -
L https://github.com/a8m/envsubst/releases/download/v1.1.0/envsubst-
`uname -s`-`uname -m` -o envsubst && \
    chmod +x envsubst && \
    mv envsubst /usr/local/bin
COPY ./nginx.config /etc/nginx/nginx.template
CMD ["/bin/sh", "-
c", "envsubst < /etc/nginx/nginx.template > /etc/nginx/conf.d/default
.conf && nginx -g 'daemon off;'"
COPY --from=builder /bookstore-
frontend/build /usr/share/nginx/html

```

### Листинг 6.3 – Dockerfile для React приложения

В результате данной главы были получены знания и практические умения по контейнеризации приложений с помощью Docker.

## Заключение

В процессе решения поставленной задачи была достигнута начальная цель по созданию web-приложения для книжного магазина. Во время выполнения данной курсовой работы было изучено немало теоретического материала, а также были просмотрены и изучены несколько уже готовых решений тех или иных задач. При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения. Таким образом, была достигнута цель и создано функционирующее рабочее приложение. Данный проект соответствует поставленным требованиям и все задачи, поставленные перед разработкой программы, выполнены.

Список выполненных задач:

- сохранение информации в централизованной базе данных;
- изменение и удаление уже существующих данных;
- наличие нескольких типов пользователей;
- фильтрация и сортировка данных;
- поиск данных.

Разработка приложения – трудоемкое занятие, выполнение которого требует всестороннего знания той предметной области, к которой относится тема разрабатываемого приложения. Был реализован удобный интерфейс, позволяющий любому пользователю быстро его освоить и без труда пользоваться функциями приложения.

Программное средство было успешно протестировано. Также к программному средству прилагается руководство пользователя.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

Данный проект соответствует поставленным требованиям и все задачи, поставленные перед разработкой программы, выполнены.

По итогу работы продукт был протестирован и отлажен, проанализированы его возможности и, внесены дополнения и улучшения с возможностью дальнейшего расширения проекта.

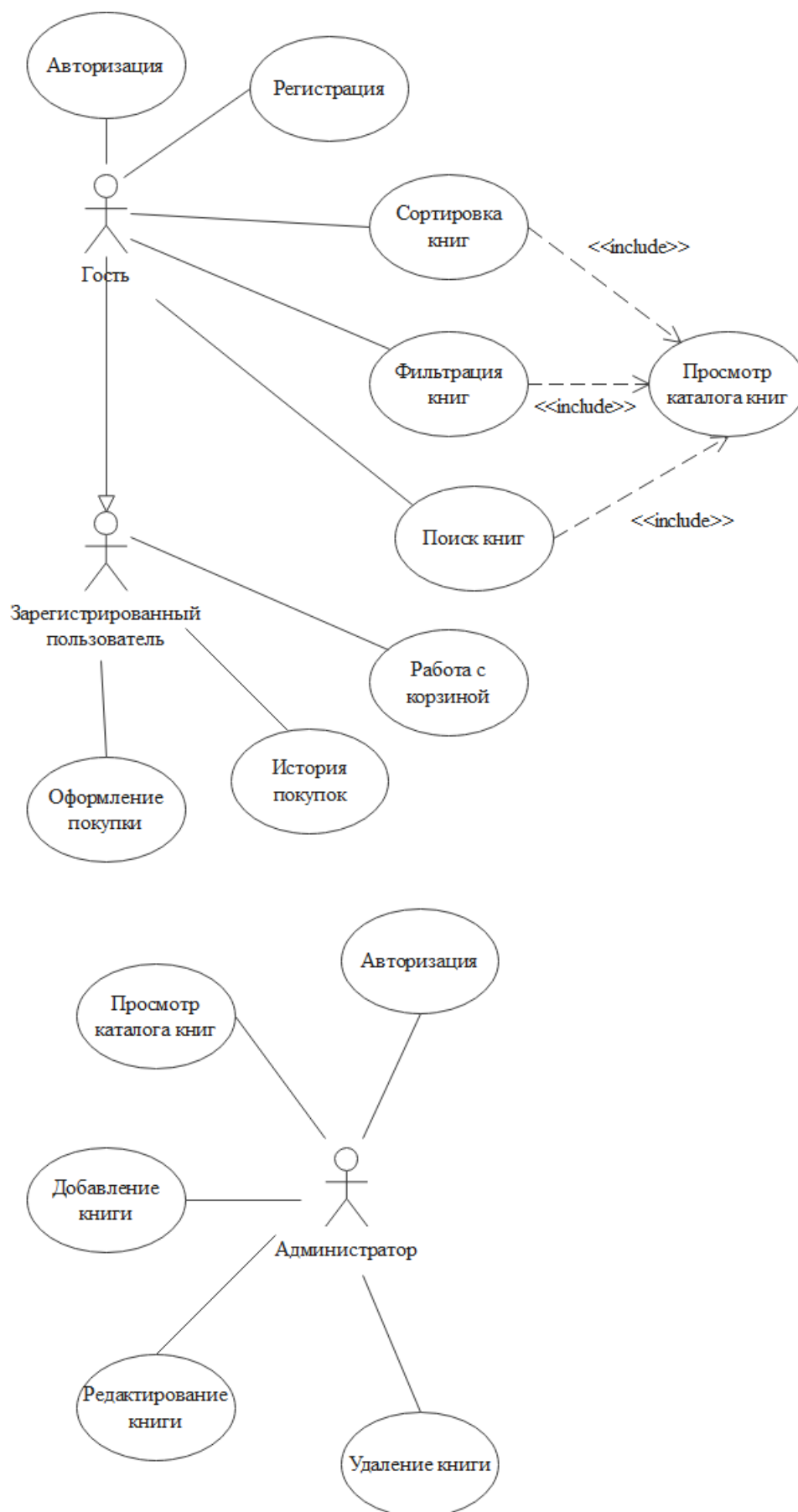
## Список использованных источников

1. Habr [Электронный ресурс]. / Режим доступа: <https://habr.com/ru>. Дата доступа: 12.11.2020.
2. ASP.NET Core 3.1 - Role Based Authorization Tutorial with Example API [Электронный ресурс]. / Режим доступа: <https://jasonwatmore.com/post/2019/10/16/aspnet-core-3-role-based-authorization-tutorial-with-example-api>. Дата доступа: 13.11.2020.
3. React – JavaScript-библиотека для создания пользовательских интерфейсов [Электронный ресурс] / Режим доступа: <https://ru.reactjs.org>. Дата доступа: 24.11.2020.
4. React Bootstrap [Электронный ресурс] / Режим доступа: <https://react-bootstrap.github.io>. Дата доступа: 25.11.2020.
5. Metanit.com. [Электронный ресурс]. / Режим доступа: <https://metanit.com/sharp/aspnet5/27.1.php>. Дата доступа: 14.11.2020.
6. Entity Framework Core. [Электронный ресурс]. / Режим доступа: <https://docs.microsoft.com/en-us/ef/core>. Дата доступа: 15.11.2020



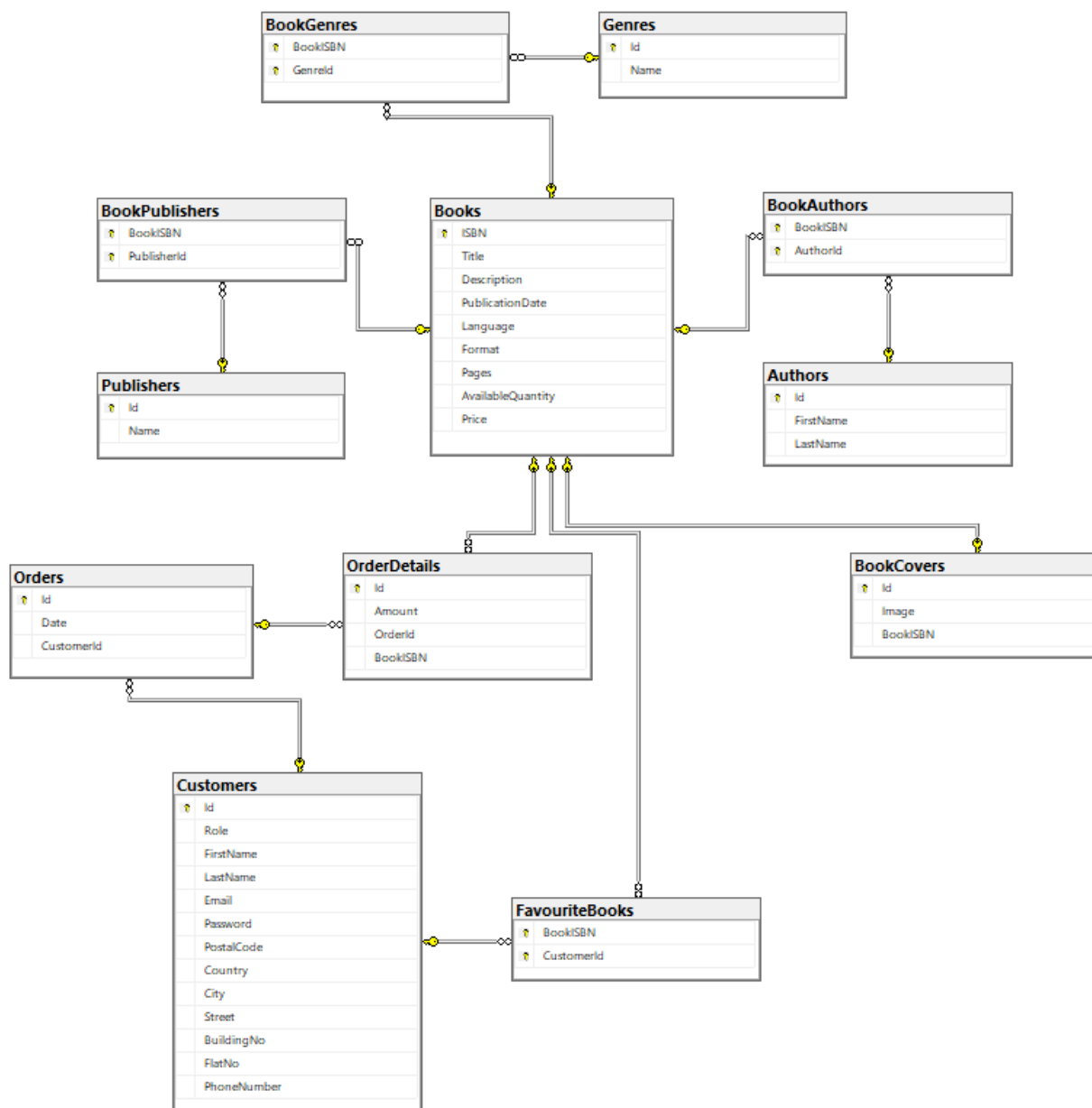
## ПРИЛОЖЕНИЕ А

Диаграмма использования



## ПРИЛОЖЕНИЕ Б

### Даталогическая модель базы данных



## ПРИЛОЖЕНИЕ В

### Исходный код модели, соответствующей сущности книги

```
public class Book
{
    [Key]
    [MaxLength(13)]
    public string ISBN { get; set; }
    [Required]
    [MaxLength(100)]
    public string Title { get; set; }
    public string Description { get; set; }
    [Column(TypeName = "date")]
    public DateTime PublicationDate { get; set; }
    [Required]
    [MaxLength(15)]
    public string Language { get; set; }
    [Required]
    public string Format { get; set; }
    public int Pages { get; set; }
    public int AvailableQuantity { get; set; }
    [Column(TypeName = "decimal(6, 2)")]
    public float Price { get; set; }
    public BookCover BookCover { get; set; }
    public IList<BookAuthor> BookAuthors { get; set; }
    public IList<BookPublisher> BookPublishers { get; set; }
    public IList<BookGenre> BookGenres { get; set; }
    public IList<FavouriteBook> FavouriteBooks { get; set; }
    public IList<OrderDetail> OrderDetails { get; set; }
}
```