Universitatea Tehnică a Moldovei

DEZVOLTAREA UNEI APLICAȚII WEB PENTRU CITIREA ȘI VÂNZAREA CĂRȚILOR

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ЧТЕНИЯ И ПРОДАЖИ КНИГ

DEVELOPMENT OF WEB APPLICATION FOR READING AND SELLING BOOKS

Student: **gr. TI-209,**
Corman Alexandr

Coordonator: **Scorohodova Tatiana**
asistentă universitară

Chișinău, 2024

MINISTERUL EDUCAŢIEI ŞI CERCETĂRII

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

Admis la usţinere

Şef de departament:

Fiodorov I. dr., conf. univ.

„___” *mai*_____ 2024

Dezvoltarea unei aplicații web pentru citirea și vânzarea cărților

Proiect de licență

Student: _____ **Corman Alexandr, TI-209**

Coordonator: _____ **Poddukin Vladimir, asist. univ.**

Consultant: _____ **Scorohodova Tatiana, asist. univ.**

Chișinău, 2024

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, I ˆ rmatică şi Microelectronică Departamentul Ingineria Software şi Automatică Programul de studii Tehnologia Informației

Aprob

șef de departament

Fiodorov Ion, dr., conf. univ.

_____

„___ ” octombrie **2023**

CAIET DE SARCINI

pentru proiectul de licență al studentului

Corman Alexandr

(numele şi prenumele studentului)

1. Tema proiectului de licență *Dezvoltarea unei aplicații web pentru citirea și vânzarea cărților* **confirmată prin hotărârea Consiliului facultății nr.** *1* **din** „*30*” *octombrie 2023*

2. Termenul limită de prezentare a proiectului de licență *20.05.2024*

3. Date inițiale pentru elaborarea proiectului de licență *Sarcina pentru elaborarea proiectului de diplomă.*

4. Conținutul memoriului explictiv

Введение
1 Анализ предметной области
2 Требования к системе
3 Анализ и моделирование системы
4 Реализация
5 Оценка стоимости проекта

Заключение
Список использованных источников

5. Conținutul părții grafice a proiectului de licență

Определение и постановка задачи. Исследование существующих аналогов. Анализ и проектирование приложения. Реализация приложения. Тестирование приложения. Оценка стоимости проекта.

6. Lista consultanților:

Consultant	**Capitol**	**Confirmarea realizării activității**	
		Semnătura consultantului (data)	**Semnătura studentului (data)**
T. Scorohodova	*Standarde tehnologice, Controlul calității, Estimarea costului*		

7. Data înmânării caietului de sarcini *04.09.2023*

Coordonator *Poddukin Vladimir* _____

semnătura

Sarcina a fost luată pentru a fi executată de către studentul *Corman Alexandr*

_____ *04.09.2023*

semnătura, data

PLAN CALENDARISTIC

Nr. crt.	Denumirea etapelor de proiectare	Termenul de realizare a etapelor	Nota
1	*Описание предметной области*	*04.09.23 – 30.09.23*	*10%*
2	*Анализ аналогов. Определение и постановка задачи*	*01.10.23 – 30.11.23*	*20%*
3	*Моделирование и проектирование системы*	*01.12.23 – 25.12.23*	*20%*
4	*Реализация приложения*	*16.01.24 – 14.03.24*	*25%*
5	*Тестирование приложения*	*05.03.24 – 10.04.24*	*10%*
6	*Описание системы*	*11.03.24 – 20.04.24*	*5%*
7	*Стоимостная оценка*	*11.03.24 – 20.04.24*	*5%*
8	*Оформление пояснительной записки*	*02.04.24 – 02.05.24*	*5%*

Student *Corman Alexandr* *( )*

Coordonator de proiect de licență *Poddukin Vladimir* *( )*

DECLARAŢIA STUDENTULUI

Subsemnatul, Corman Alexandr, declar pe proprie răspundere că lucrarea de faţă este rezultatul muncii mele, pe baza propriilor cercetări şi pe baza informaţiilor obţinute din surse care au fost citate şi indicate, conform normelor etice, în note şi în bibliografie. Declar că lucrarea nu a mai fost prezentată sub această formă la nici o instituţie de învăţământ superior în vederea obţinerii unui grad sau titlu ştiinţific ori didactic.

Semnătura autorului

UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ
DEPARTAMENTUL INGINERIA SOFTWARE ŞI AUTOMATICĂ
PROGRAMUL DE STUDII TEHNOLOGIA INFORMAŢIEI

AVIZ

la proiectul de licență

Titlul: Разработка приложения для чтения и покупки книг

Студент Корман Алекснадр гр. TI-209

1. Actualitatea temei: В совремменном мире бумажные книги уходят на второй план с появлением интернета, поэтому у современного человека появляется потребность читать книги когда он хочет и какие хочет без привязке к бумаге. Данный дипломный проект позволяет это через веб-приложение, которое может быть доступно в любой части мира.

2. **Caracteristica proiectului de licență**: Приложение было создано для удобства чтения книг в любой точке земного шара.

3. Analiza prototipului: Приложение совмещает в себе возможность чтения книг, их покупки, а так же возможность выкладывать свои, получать комментарии и улучшать свои писательские навыки.

4. **Estimarea rezultatelor obținute:** Это простое в использовании и интуитивно понятное приложение для любого пользователя кто хочет встать на путь писателя или уже им является и хочет увеличить количество своих читателей.

5. Corectitudinea materialului expus: Материал представлен со ссылками на источники, используется проверенные источники, потому что написанны людьми с опытом работы в сфере информационных технологий.

6. Calitatea materialului grafic: Проект представлен с помощью: диаграмм, таблиц и интерфейсов приложения.

7. Valoarea practică a proiectului: Предназначен для ноутбуков, компьютеров, планшетов, смартфонов.

8. Observații și recomandări: Требования к дипломному проекту были полностью соблюдены, замечаний нет.

9. Caracteristica studentului și titlul conferit: Студент, Корман Александр, проявил индивидуальность и профессионализм при создании дипломного проекта, доказал целесообразность разработки продукта, смог соблюсти все поставленные департаментом цели и требования.

Din cele relatate, urmează că lucrarea de licență poate fi admisă spre susținere, cu nota _____

Lucrarea în forma electronică corespunde originalului prezentat către susținere publică.

Coordonatorul proiectului de licență _____ Poddukin Vladimir, asist. univ.

semnătura, data

АННОТАЦИЯ

Целью дипломного проекта было создание приложения для чтения и покупки книг. Итоговым продуктом стало приложение, предназначенное для любых компьютерных устройств, с простым и понятным интерфейсом и всем необходимым функционалом, а сам проект состоит из пяти глав, в которых поэтапно описан процесс создания приложения. Первая глава посвящена анализу предметной области, главной задачей было объяснить важность и необходимость выбранной темы с помощью аргументов. Следующий шаг это выявление и анализ работающих аналогов, создание требований к системе, к ее компонентам. Также предоставлены основные задачи и методы их реализации.

Вторая и третья главы дипломного проекта посвящены моделированию и проектированию, а точнее поведенческому и структурному описанию, что наглядно показывает все аспекты системы. Отношения между компонентами, варианты использования приложения и общее представление системы передаются через UML-диаграммы и их описание. Четвертая глава содержит описание практической составляющей дипломного проекта, реализацию задуманной системы, примеры кода, а также список всех технологий и инструментов, которые были использованны для создания указанного приложения. Часть реализации включает в себя тестирование системы, выявление ошибок и их последующее исправление.

В пятой главе была произведенена оцененка стоимости проекта и обоснованы затраты на внедрение программного продукта. В конце были подведены итоги всех этапов разработки системы, а также составлено заключение.

REZUMAT

Scopul proiectului de teză a fost crearea unei aplicații pentru citirea și cumpărarea de cărți. Produsul final a fost o aplicație concepută pentru orice dispozitiv de calculator, cu o interfață simplă și clară și toate funcționalitățile necesare, iar proiectul în sine este format din cinci capitole, care descriu pas cu pas procesul de creare a unei aplicații. Primul capitol este consacrat analizei temei, sarcina principală a fost de a explica importanța și necesitatea temei alese cu ajutorul argumentelor. Următorul pas este identificarea și analiza analogilor de lucru, crearea cerințelor pentru sistem și componentele acestuia. De asemenea, sunt furnizate principalele sarcini și metode pentru implementarea lor.

Al doilea și al treilea capitol al proiectului de teză sunt dedicate modelării și proiectării, sau mai degrabă descrierii comportamentale și structurale, care arată clar toate aspectele sistemului. Relațiile dintre componente, cazurile de utilizare a aplicațiilor și imaginea de ansamblu a sistemului sunt transmise prin diagrame UML și descrierile acestora. Al patrulea capitol conține o descriere a componentei practice a proiectului de diplomă, implementarea sistemului planificat, exemple de cod, precum și o listă a tuturor tehnologiilor și instrumentelor care au fost utilizate pentru a crea aplicația specificată. O parte a implementării implică testarea sistemului, identificarea erorilor și apoi corectarea acestora.

În capitolul al cincilea a fost evaluat costul proiectului și au fost justificate costurile de implementare a produsului software. La final, au fost rezumate rezultatele tuturor etapelor de dezvoltare a sistemului și s-a tras o concluzie.

ABSTRACT

The goal of the thesis project was to create an application for reading and purchasing books. The final product was an application designed for any computer device, with a simple and clear interface and all the necessary functionality, and the project itself consists of five chapters, which describe the process of creating an application step by step. The first chapter is devoted to the analysis of the subject area; the main task was to explain the importance and necessity of the chosen topic with the help of arguments. The next step is to identify and analyze working analogues, create requirements for the system and its components. The main tasks and methods for their implementation are also provided.

The second and third chapters of the thesis project are devoted to modeling and design, or rather behavioral and structural description, which clearly shows all aspects of the system. The relationships between components, application use cases, and the overall view of the system are conveyed through UML diagrams and their descriptions. The fourth chapter contains a description of the practical component of the diploma project, the implementation of the planned system, code examples, as well as a list of all the technologies and tools that were used to create the specified application. Part of the implementation involves testing the system, identifying errors and then correcting them.

In the fifth chapter, the cost of the project was assessed and the costs of implementing the software product were justified. At the end, the results of all stages of system development were summed up, and a conclusion was drawn up.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире, где интернет есть у каждого, где он заменяет человеку большую часть жизни, где компьютеры и телефоны стали неотъемлемой частью повседневной жизни, многие люди забывают что есть что-то вне интернета, например бумажные книги. Сейчас мало кто читает книги, тем более бумажные, ведь это так неудобно, книга громоздкая и ее нельзя брать куда захочется, но все же читающие люди остались и электронные книги стали для них спасением, ведь электронную книгу можно читать где угодно. С целью предоставить пользователям возможность читать те книги которые им нравятся, в данной дипломной работе будет проведена работа над созданием веб-приложения для чтения электронных книг.

В ходе написания дипломной работы будет рассмотрено устройство веб-приложения, которое позволит не только читать, а также покупать, писать, скачивать книги.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В этой главе идет речь об анализе предметной области, которая включает в себя анализ предметной области дипломного проекта. Осуществлен анализ существующей информации о аналогах разрабатываемого приложения, включая их краткое описание, преимущества и недостатки, представленные в сравнительной таблице. Этот этап предоставляет ответы на вопросы о характере приложения, его предполагаемой структуре, целевой платформе и области применения в рамках дипломного проекта. Анализ занимает важное место в процессе планирования системы, представляя собой комплексный обзор различных аналитических аспектов разрабатываемого приложения.

1.1 Важность темы

Веб-приложения для чтения книг является прогрессивным способом чтения книг. Один из важнейших аспектов сайтов для чтения книг - это обеспечение доступа к литературе разных стран и культур. Сайты предоставляют доступ к классическим произведениям, позволяя читателям познакомиться с литературным наследием разных эпох и культур. Это способствует культурному обогащению и пониманию разнообразия мировой литературы.

Сайты для чтения книг также помогают расширить аудиторию для авторов. Электронные книги доступны во всем мире и не ограничены тиражами и физическими магазинами. Это дает возможность молодым или малоизвестным авторам представить свои произведения широкой публике, а читателям - найти новых авторов и жанры, которые могли бы им понравиться. Сайты для чтения книг также играют важную роль в образовании.

Студенты, учителя и исследователи могут находить необходимую литературу онлайн и изучать ее без дополнительных затрат. Это также удобно для тех, кто изучает иностранные языки, так как они могут читать тексты на оригинальном языке и одновременно изучать перевод.

Популярность сайтов для чтения книг способствует сокращению использования бумаги и уменьшению экологической нагрузки. Электронные книги не требуют древесных ресурсов для производства и не создают отходов, связанных с утилизацией бумажных книг. Это способствует сохранению природных ресурсов и снижению воздействия на окружающую среду.

В моем случае я соберу все преимущества различных площадок в своем приложении убрав соответствующие минусы. В таком случае мое приложение будет в разы лучше выглядеть в глазах пользователей, чем другие.

1.2 Существующие аналоги

Перед тем как разработать веб-приложеие читалку, важно провести анализ уже существующих аналогов. Это позволит определить особенности, преимущества и недостатки конкурирующих продуктов.

Одним из аналогов, который был рассмотрен, является "Author Today" (рисунок 1.1) — веб-приложение с похожим функционалом и большой библиотекой книг. В "Author Today" пользователь может читать книги, используя веб-приложение или мобильное приложение. Книги могут платными и бесплатными, автор сам задает это, автор может ограничивать чтение по главам, одни бесплатные, другие нет. Автор может обьединять книги по циклам, писать свой блог. Читатель же может оценивать произведения, писать комментарии и благодарить автора.

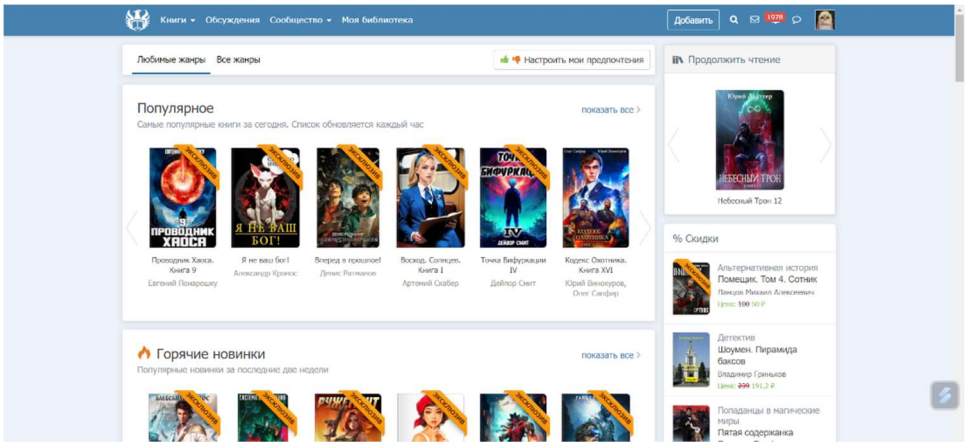

Рисунок 1.1 – "Author Today"

Еще одним аналогом является " Litres" (рисунок 1.2). Обычная читалка ничем особым не выделяющаяся. Посредственный дизайн усложняет взаимодействие пользователя с приложениям. Предоставляет стандартные функции для читателя такие как: чтения книг, комментирование.

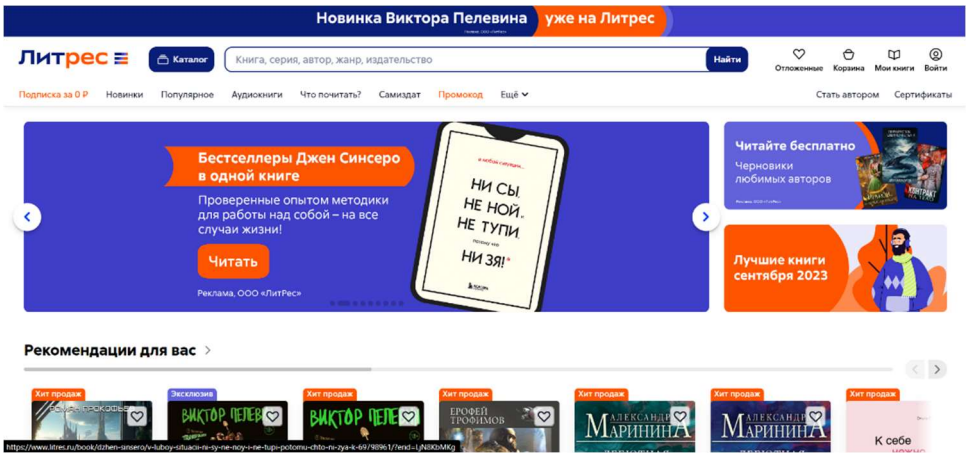

Рисунок 1.2 – "Litres"

Еще одним интересным аналогом является "Литнет" (рисунок 1.3). Приятный дизайн, множество понятных пользователю функции как в других читалках. Отличается совсем незначительно от других.

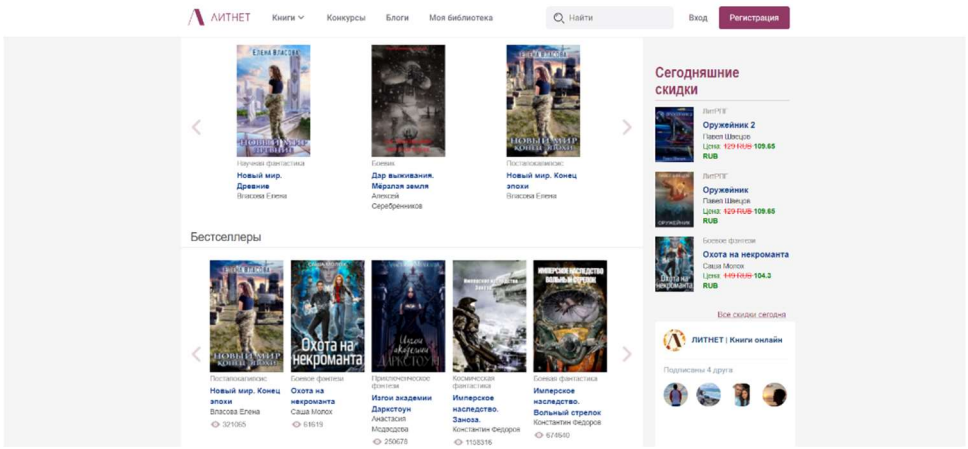

Рисунок 1.3 – "Литнет"

В таблице 1.1 представлены основные критерии, которые используются для сравнения нашего разрабатываемого приложения с уже существующими аналогами. Эта таблица предоставляет инструмент для более наглядного выявления преимуществ и недостатков наших систем относительно конкурентов.

Таблица 1.1 – Сравнительный анализ аналогов

Критерии	Author Today	Litres	Litnet	Разрабатываемое приложение
Чтение книги	**Да**	**Да**	**Да**	**Да**
Создание комментариев	**Да**	**Да**	**Да**	**Да**
Мобильное приложение	**Да**	**Нет**	**Нет**	**Да**
Назойливая реклама	**Да**	**Да**	**Нет**	**Нет**
Уведомления при выходе нового	**Да**	**Нет**	**Нет**	**Да**
Аудио книги	**Да**	**Нет**	**Нет**	**Да**
Умный подбор рекомендаций	**Нет**	**Нет**	**Нет**	**Да**

1.3 Цель, задачи и требования к системе

Цель дипломного проекта, является веб-приложения. Для создания приложения было принято решение выбрать для бэкенда “ASP .NET CORE” – кроссплатформенныый фреймворк для содания MVC, API приложений. В данном приложение используется MVC принцип разработки бэкенд приложения и немного RESTful API. Для фронтэнда используется чистый javascript совместно с Vue.js для реактивности и немного jQuery. В качестве библеотек для дизайна были выбраны Fomantic UI и Bootstrap.

Можно выделить несколько задач, по результату выполнения которых может быть достигнута поставленная цель:

- создание бэкенда;
- создание фронтенда;
- настройка базы данных;
- создание дополнительных возможностей;
- тестирование приложения на разных этапах реализации.

Создание сложных программных приложений требует разделения жизненного цикла программного обеспечения на определенные этапы. Жизненный цикл программного приложения включает следующие ключевые этапы:

- анализ предметной области и создание технического задания – на этом этапе проект существует лишь на бумаге, происходит изучение предметной области и формирование технического задания;
- анализ и проектирование структуры приложения – тот этап включает создание документации, описывающие внутренние взаимодействия и взаимодействие с системой;
- реализация – на данном этапе осуществляется непосредственная реализация системы в соответствии с проектной документацией;
- тестирование и отладка – этап проверки насколько проект соответствует требованиям, выявления ошибок и их последующего устранения;
- утилизация – непосредственно использование продукта;
- обновления приложения – этап подразумевает выход обновлений для игры, после её релиза и поддержка приложения в целом.

1.4 Язык программирования

Для создания веб-приложения будет использоваться C# (рисунок 1.4). C# — это мощный и универсальный объектно-ориентированный язык программирования, разработанный компанией Microsoft.

Объектно-ориентированный язык: C# построен вокруг концепций объектно-ориентированного программирования (ООП), что облегчает создание модульных и структурированных программ. Интеграция с платформой .NET: C# является основным языком для разработки приложений на платформе .NET (Microsoft .NET Framework и .NET Core). Это обеспечивает высокую переносимость кода и возможность работы на различных операционных системах. Сильная типизация: C# предоставляет строгую, статическую типизацию, что помогает обнаруживать ошибки на этапе компиляции, а не во время выполнения программы. Управление памятью: Язык C# использует систему управления памятью, основанную на сборке мусора, что облегчает разработку и предотвращает многие проблемы, связанные с утечками памяти. Многозадачность: C# поддерживает асинхронное программирование и многозадачность, что позволяет эффективно работать с асинхронными операциями и параллельными вычислениями. Богатая стандартная библиотека: C# поставляется с обширной стандартной библиотекой классов,

предоставляющей широкий спектр функциональности для работы с файлами, сетевыми протоколами, базами данных и многими другими аспектами программирования. Современные возможности языка: C# постоянно развивается, добавляя новые функции и синтаксические конструкции. Введение асинхронного программирования, лямбда-выражений, LINQ (Language Integrated Query) и других возможностей делает его современным и эффективным языком программирования. C# широко используется для создания различных типов приложений, включая веб-приложения, настольные приложения, мобильные приложения, игры и многое другое.

Рисунок 1.4 – Язык C#

1.5 Среда разработки

Для создания веб-приложения будет использоваться среда разработки Visual Studio (рисунок. 1.5).

Visual Studio - это интегрированная среда разработки (IDE) от Microsoft, предназначенная для создания различных типов приложений, включая веб-приложения, настольные приложения, мобильные приложения, игры и другие программные решения. Многозадачность и поддержка множества языков: Visual Studio предоставляет возможность разработки на различных языках программирования, таких как C#, Visual Basic, F#, C++, Python, и другие. Это позволяет разработчикам выбирать язык в зависимости от требований проекта. Интеграция с платформой .NET: Visual Studio тесно интегрирована с платформой .NET, что облегчает создание, отладку и

развертывание приложений для .NET Framework и .NET Core. Отладка и профилирование: Инструменты отладки Visual Studio обеспечивают широкие возможности для выявления и устранения ошибок в коде. Также предоставляются средства профилирования для анализа производительности приложений. Графический дизайнер интерфейса: Visual Studio содержит инструменты для визуального проектирования пользовательского интерфейса приложений, что облегчает создание привлекательных и функциональных пользовательских интерфейсов. Управление версиями и командная работа: Встроенная поддержка систем контроля версий, таких как Git, облегчает управление и отслеживание изменений в коде. Интеграция с платформой Azure DevOps обеспечивает совместную работу в командах. Расширяемость: Visual Studio поддерживает расширения и плагины, что позволяет разработчикам интегрировать сторонние инструменты и расширять функциональность среды разработки. Поддержка различных платформ: Visual Studio позволяет разрабатывать приложения для различных платформ, включая Windows, Linux, macOS, мобильные устройства и облачные сервисы. Кросс-платформенная разработка: С Visual Studio можно создавать приложения для различных операционных систем, в том числе использовать .NET Core для разработки кросс-платформенных приложений. Visual Studio является одним из наиболее распространенных инструментов разработки в индустрии программного обеспечения и широко используется профессиональными разработчиками.

Рисунок 1.5- Visual Studio

2 ТРЕБОВАНИЯ К СИСТЕМЕ

В данной главе разбираются функциональные и нефункциональные требования, а также поток данных к нашей системе, которые помогут лучше разобраться в том, как работает каждый режим игры и как он реализуется.

2.1 Функциональные требования

Функциональные требования - это часть спецификации системы или программного продукта, которая описывает функции, операции и возможности, которые должны быть реализованы в продукте. Они фокусируются на том, что система должна делать, и какие функциональные возможности она должна предоставить пользователю. Функциональные требования часто определяются как "что" система должна делать.

Примеры функциональных требований могут включать:

- Регистрация пользователя: Система должна предоставлять форму для регистрации новых пользователей. При этом требуется сбор информации, такой как имя, адрес электронной почты и пароль.
- Поиск функции: Система должна обеспечивать возможность поиска информации на основе заданных критериев с использованием поискового механизма.
- Добавление товара в корзину: В электронной коммерции система должна позволять пользователям добавлять товары в корзину покупок, а затем осуществлять оформление заказа.
- Выполнение арифметических операций: В приложении калькулятора функциональные требования могут включать выполнение базовых арифметических операций, таких как сложение, вычитание, умножение и деление.
- Генерация отчетов: Система управления базами данных должна предоставлять возможность генерации отчетов по определенным критериям.

В первую очередь система предоставляет пользователю возможность читать книги в онлайн, но только бесплатные. Для чтения платных книг, соответственно и их покупка доступна только зарегистрированным пользователям. Зарегистрированные пользователи так же могут оставлять комментарии, подписываться на обновления книг и авторов, скачивать книги, так же сами писать

свои книги. То есть можно смело утверждать, что незарегистрированный пользователь сильно ограничен в возможностях использования системы.

2.2 Поток данных

Поток данных (Data Flow) - это концепция, которая описывает передачу данных в системе, программе или процессе. Он представляет собой путь, по которому данные перемещаются от одного места к другому внутри системы. Концепция потока данных часто используется в контексте программирования, проектирования информационных систем, а также в области обработки данных и анализа. В программировании поток данных может быть представлен следующим образом:

- Ввод данных (Input): Получение данных из внешних источников, таких как пользовательский ввод, файлы или сенсоры.
- Обработка данных (Processing): Применение логики и алгоритмов к входным данным для получения желаемого результата. Этот этап может включать операции фильтрации, сортировки, вычислений и другие манипуляции с данными.
- Вывод данных (Output): Предоставление результатов обработки в виде вывода, который может быть направлен на экран, в файл, базу данных или другое место.

Проектирование потока данных позволяет легко представлять и отслеживать перемещение информации в системе, что важно для понимания работы приложений и оптимизации процессов обработки данных. В области обработки данных и анализа поток данных также может означать непрерывный поток информации, который поступает и обрабатывается в режиме реального времени. Например, в системах обработки потоков данных (stream processing) данные поступают и обрабатываются по мере их поступления, а не после того, как весь объем данных собран. Таким образом, поток данных представляет собой путь передвижения информации внутри системы, процесса или программы от начального источника до конечного пункта назначения, проходя через различные этапы обработки и манипуляций.

Система использует множество данных для своей работы. В первую очередь используется данные пользователя, которые попадают в систему через форму регистрации во фронтенд части и попадают в базу данных через бэкенд часть. Так же часть данных может изменяться или дополняться через форму аккаунта в личном кабинете. В систему попадает книги в полном обьеме загруженные в веб-приложение, комментарии к этим книгам, потом эти данные используются для отображения в некоторых частях веб-приложения. В системе используются алгоритмы для подбора

книг по предпочтениям для пользователей основываясь на уже прочитанных книгах, добавленных в библиотеку.

2.3 Нефункциональные требования

Нефункциональные требования - это аспекты, характеристики или ограничения системы, которые не описывают конкретные функции, которые система должна выполнять, а скорее определяют свойства, качества или ограничения, которые должны присутствовать в системе или в ее компонентах. Эти требования описывают "как" система должна выполнять свои функции, а не "что" она должна делать. Примеры нефункциональных требований включают:

- Производительность: Определение требований по скорости работы, времени отклика и обработки данных системы в различных условиях.
- Надежность: Указание на уровень стабильности, надежности и устойчивости системы, включая требования к отказоустойчивости и восстановлению после сбоев.
- Безопасность: Описывает требования по обеспечению защиты данных, аутентификации пользователей, управлению доступом и другим аспектам безопасности.
- Масштабируемость: Устанавливает требования по способности системы масштабироваться для обработки увеличения объема данных или пользователей.
- Удобство использования (Usability): Включает требования по дизайну интерфейса, простоте использования, интуитивности и удобству взаимодействия с пользователем.
- Сопровождаемость (Maintainability): Определение того, насколько легко можно поддерживать, модифицировать и обновлять систему.
- Совместимость: Описывает требования к совместимости системы с другими системами, стандартами и аппаратным обеспечением.
- Эффективность использования ресурсов: Определение эффективного использования ресурсов, таких как память, процессорное время, энергия и т.д.

Нефункциональные требования являются важной частью общего набора требований, так как они влияют на аспекты системы, которые могут быть критическими для ее успеха, надежности и удовлетворения потребностей пользователей и бизнеса.

Разрабатывая веб-приложение были выделенны нефункциональные требования необходимые для комфортного использования системы пользователями.

- Производительность:
  - Время загрузки страницы должно быть не более 2 секунд для основного контента.
  - Система должна поддерживать одновременное обслуживание не менее 500 активных пользователей без существенного снижения производительности.
- Доступность:
  - Сайт должен обеспечивать доступность не менее 99% времени в течение месяца.
  - Веб-приложение должно поддерживать чтение книги в оффлайн-режиме после их предварительной загрузки.
- Безопасность:
  - Все пользовательские данные должны передаваться по протоколу HTTPS.
  - Система должна обеспечивать защиту от угроз безопасности, такие как SQL-инъекции и межсайтовые атаки (XSS, CSRF).
- Удобство использования:
  - Интерфейс должен быть интуитивно понятным для пользователей всех уровней.
  - Система должна предоставлять возможности настройки шрифта, цветовой схемы и других параметров для удовлетворения индивидуальных потребностей пользователей.

3 АНАЛИЗ И МОДЕЛИРОВАНИЕ СИСТЕМЫ

Процесс моделирования является один из важнейших компонентов в разработке программного обеспечения. Моделирование системы является важным инструментом для эффективного проектирования, анализа и внедрения сложных систем, позволяя участникам проекта иметь общее понимание целевой системы. Моделирование системы помогает участникам проекта и заинтересованным сторонам лучше понять структуру, функциональность и взаимодействие компонентов системы.

3.1 Разработка логической модели БД

Чтобы посторить модель базы данных использовались мощности программы SSMS.

Для создания диаграммы зависимостей модели базы данных в SSMS необходимо сначала создать эту базу данных, задать все поля, так же добавить ключи для связей.

Создание сущностей и добавление определений

В первую очередь определяется, что есть пользователь в системе. Это характеризуется таблицей User: UserId (Primary Key) – уникальный идентификатор пользователя, Username – имя пользователя в системе, Email – электронный почтовый адрес пользователя для связи с ним вне системы, Password – захэшированный пароль пользователя для входа в систему, CreatedAt – дата регистрации / создания аккаунта. На рисунке 3.1 представлен пример создания таблицы.

```
CREATE TABLE [dbo].[User](
    [UserId] [int] IDENTITY(1,1) NOT NULL,
    [Username] [nvarchar](255) NOT NULL,
    [Email] [nvarchar](255) NOT NULL,
    [Password] [nvarchar](255) NOT NULL,
    [CreatedAt] [datetime] NOT NULL,
 CONSTRAINT [PK_User] PRIMARY KEY CLUSTERED
(
    [UserId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Email] UNIQUE NONCLUSTERED
(
    [Email] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY],
 CONSTRAINT [IX_Username] UNIQUE NONCLUSTERED
(
    [Username] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Рисунок 3.1 – Пример создания таблицы для модели БД

Таблица Book характеризуется следующими полями: BookId (Primary Key) – уникальный идентификатор книги, Title – название книги, Description – аннотация книги, Author (Foreign Key) – внешний ключ ссылающийся на таблицу User (UserId) и характеризующий автора книги, CreatedAt – дата публикации книги.

Таблица Comment характеризуется следующими полями: CommentId (Primary Key) – уникальный идентификатор комментария, Text – содержание комментария, AuthorId (Foreign Key) – внешний ключ ссылающийся на таблицу User (UserId) и характеризующий автора комментария, BookId (Foreign Key) – внешний ключ ссылающийся на таблицу Book (BookId) и характеризующий к какой книге принадлежит комментарий, CreatedAt – дата публикации комментария.

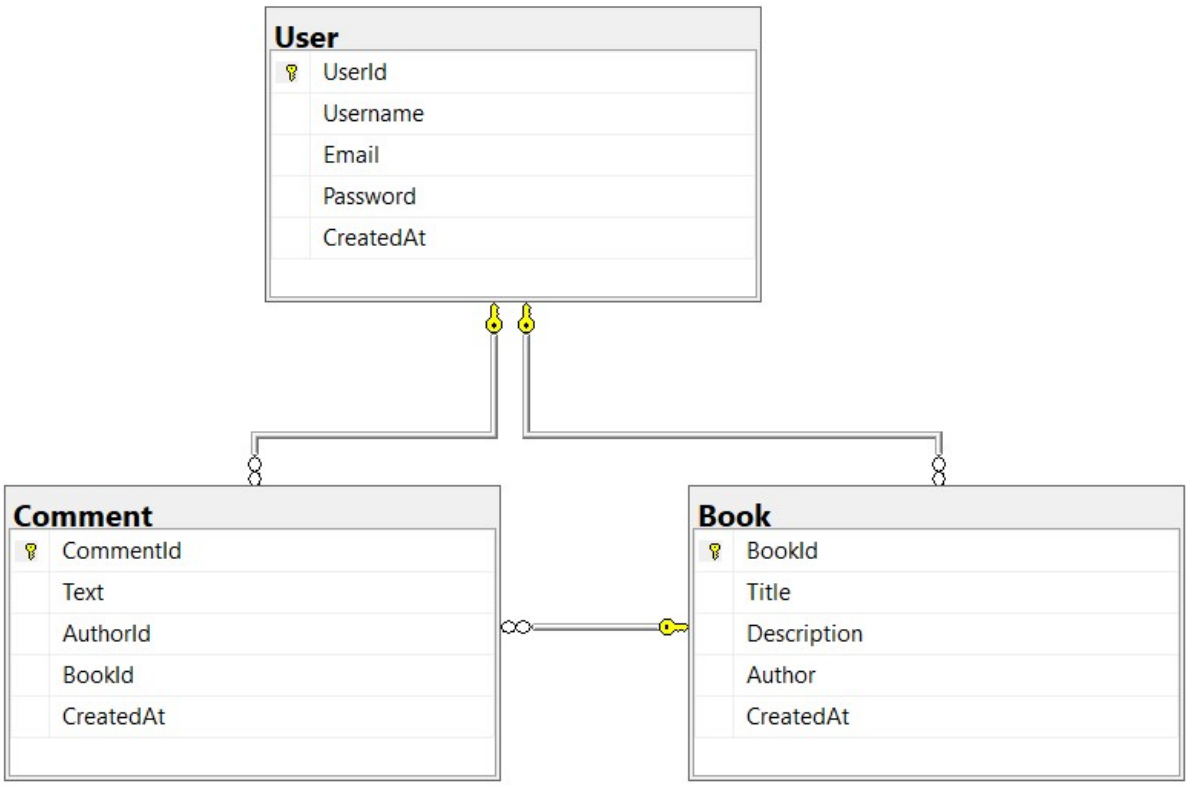

Рисунок 3.2 – Логическая схема базы данных

3.2 Разработка контекстной диаграммы и её декомпозиция

Контекстная диаграмма - это вид диаграммы, который позволяет визуализировать взаимодействие между системой и её окружением. Она описывает систему как один блок (чаще всего прямоугольник) и внешние сущности или компоненты, с которыми система взаимодействует. Контекстная диаграмма является высокоуровневым представлением и обычно используется на начальном этапе анализа системы. Контекстная диаграмма предоставляет участникам проекта и заинтересованным сторонам общее представление о системе и её месте в окружающей среде.

Контекстная диаграмма проекта (рисунок 3.3) состоит из требований необходимых веб-приложению для его разработки.

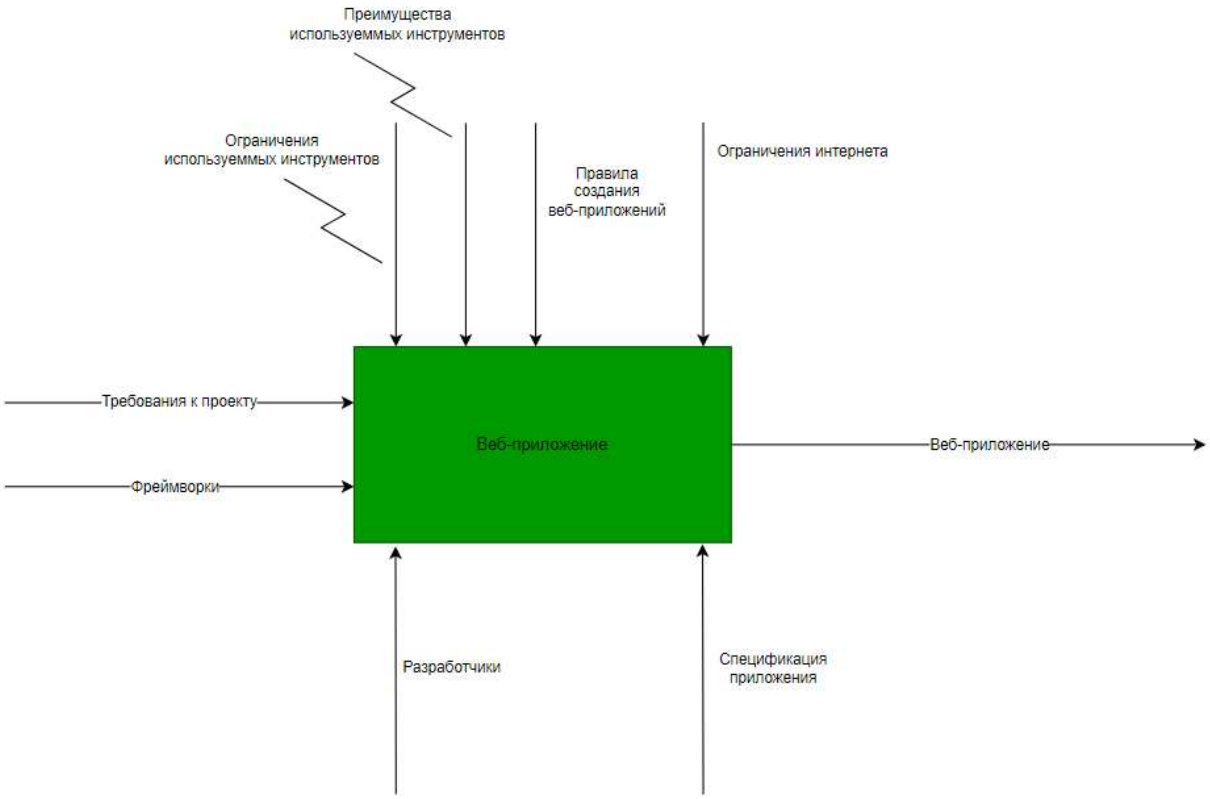

Рисунок 3.3 – Контекстная диаграмма

Входные данные:

- требования к проекту;
- фреймворки

Управляющие воздействия:

- ограничения используемых инструментов;
- преимузества используемых инструментов;

– правила создания веб-приложений;
– ограничения интернета;

Механизмы:

– разработчики;
– спецификация приложения.

Выходные данные:

– веб-приложение.

Декомпозиция контекстной диаграммы - это процесс разбиения общего представления системы на более детализированные компоненты, уточняя внутреннюю структуру и взаимосвязи между её частями (рисунок 3.4). Этот процесс позволяет перейти от высокоуровневого представления системы к более подробному описанию её компонентов и взаимосвязей, что является важным этапом в анализе и проектировании системы. Вот несколько причин, почему декомпозиция контекстной диаграммы может быть полезной:

a) Уточнение структуры системы: Декомпозиция позволяет более детально определить внутренние компоненты и подсистемы системы, что важно для точного понимания её структуры.
b) Более детальное проектирование: Детализация контекстной диаграммы является этапом, предшествующим более подробному проектированию компонентов системы.
c) Лучшее понимание требований: Декомпозиция контекстной диаграммы помогает лучше понять требования к системе, учитывая их внутренний контекст и взаимосвязи.
d) Определение зависимостей: Процесс декомпозиции помогает выявить зависимости между компонентами системы, что важно для управления изменениями и понимания влияния изменений на другие части системы.

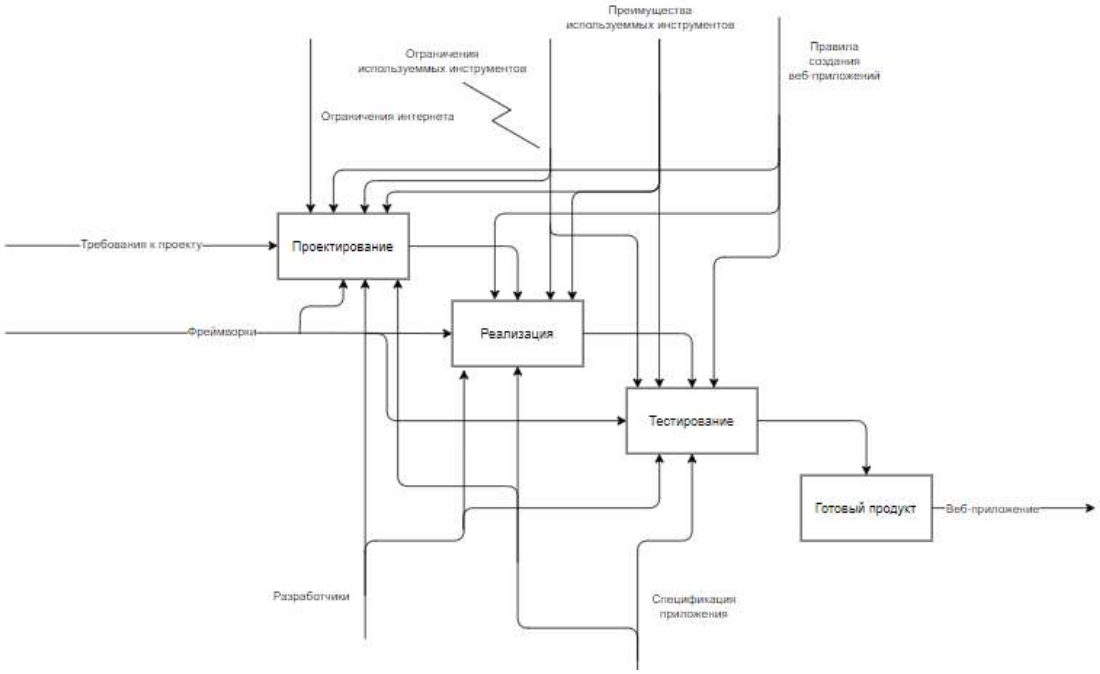

Рисунок 3.4 – Декомпозиция контекстной диаграммы

3.3 Функциональное назначение системы

Диаграммы вариантов использования предоставляют общее представление о том, как система взаимодействует с внешними акторами и какие функциональные возможности предоставляет. На диаграмме вариантов использования (рисунок 3.5) изображены действия, которые может выполнить пользователь в системе. Не зарегистрированный пользователь может только читать бесплатные книги, тот кто вошел в систему уже может оставлять комментарии, купить какое-то произведение.

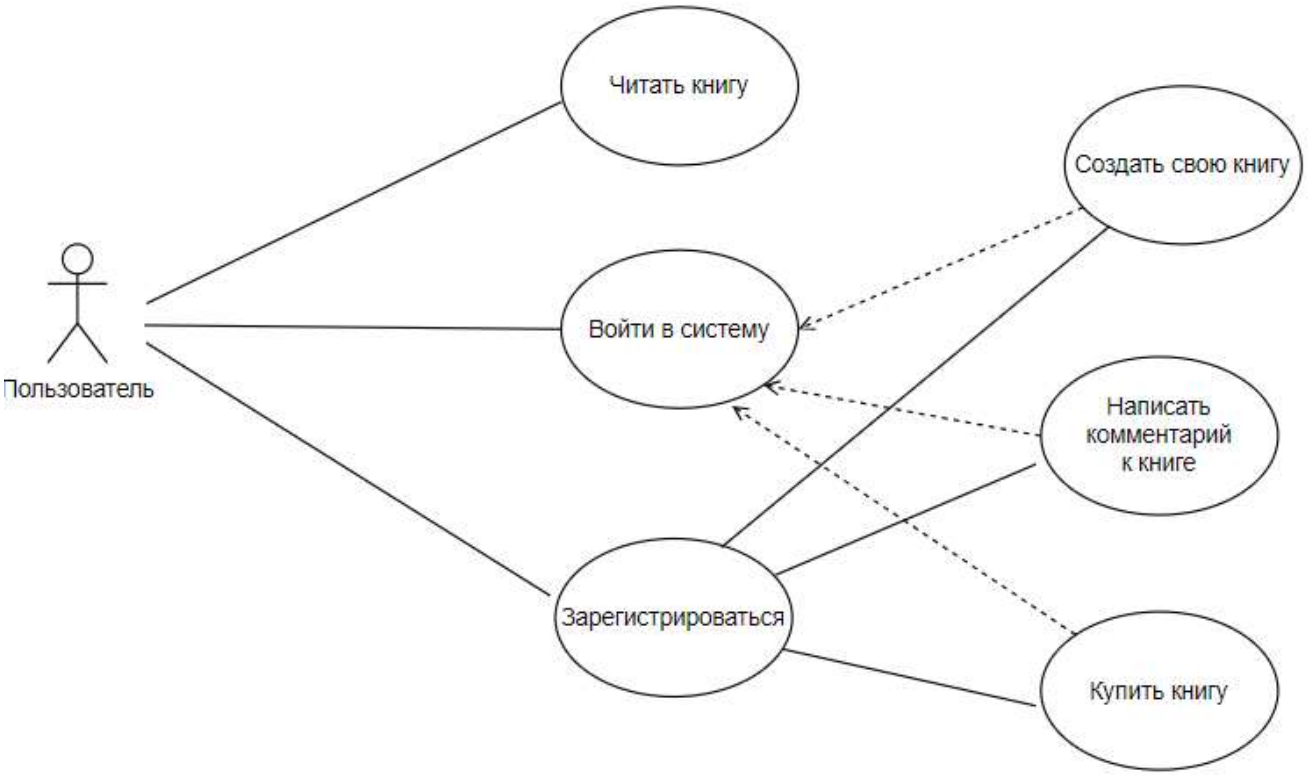

Рисунок 3.5 – Диаграмма прецедентов для взаимодействия с системой

3.4 Взаимодействие элементов системы

В информационном плане рассматривается взаимодействие между элементами, где коммуникация осуществляется обменом информацией между взаимодействующими объектами. Для моделирования такого взаимодействия в языке UML применяются соответствующие диаграммы взаимодействия. Диаграммы взаимодействия — это один из видов диаграмм в языке моделирования UML. Они используются для визуализации и описания взаимодействия между различными элементами системы, такими как объекты, компоненты и акторы.

На диаграмме последовательности (рисунок 3.6) показывается как происходит изменение счёта при соревновательной игре.

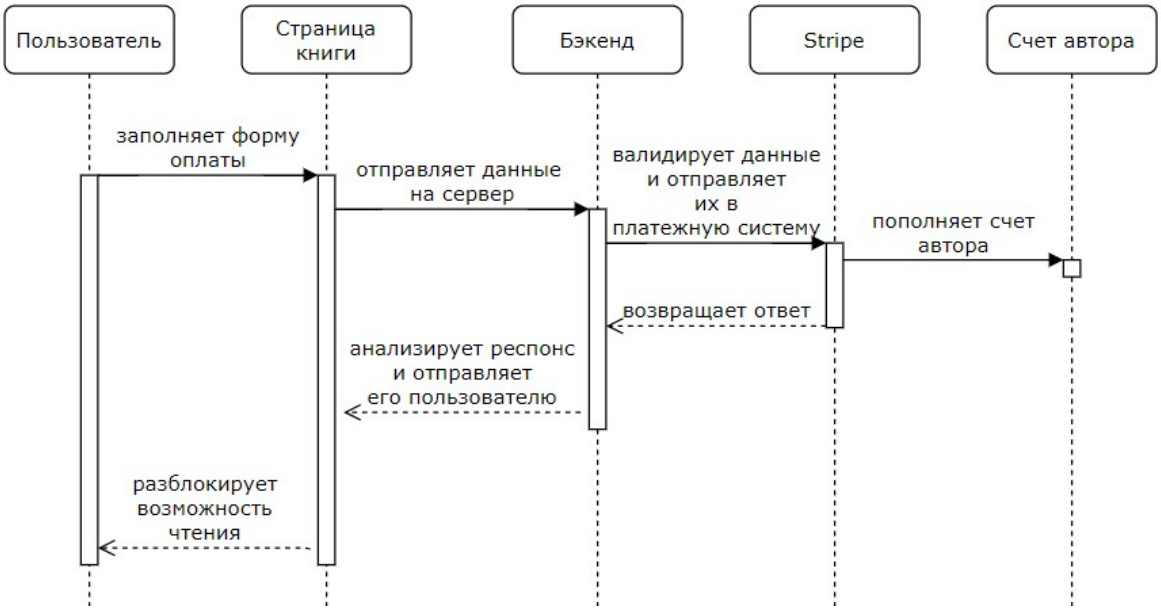

Рисунок 3.6 – Диаграмма последовательности оплаты книги

На рисунке 3.6 представлен процесс оплаты книги в веб-приложении. Сначала пользователь должен заполнить форму, где указывает данные своей карты. Эти данные отправляются на сервер, а потом в платежную систему Stripe, она их валидирует и пополняет счет автора. Далее Stripe предоставляет ответ серверу, который в свою очередь возвращает ответ на фроненд, где пользователю предоставляется возможность чтения.

На рисунке 3.7 представлен процесс начала чтения книги. Пользователь нажимает кнопку «Читать» потом его перенаправляет на страницу с текстовы содержанием книги. Но сначала нужно этот текст получить. Сначала фроненд делает запрос на бэкенд, который в свою очередь делает запрос в базу данных. От туда бэкенд достает уникальную ссылку на книгу и отправляет ее в хранилище Azure. Используя эту уникальную ссылку Azure находит файл с книгой и отправляет ее на обратно серверу, а он фронтенду, который отображает текст пользователю.

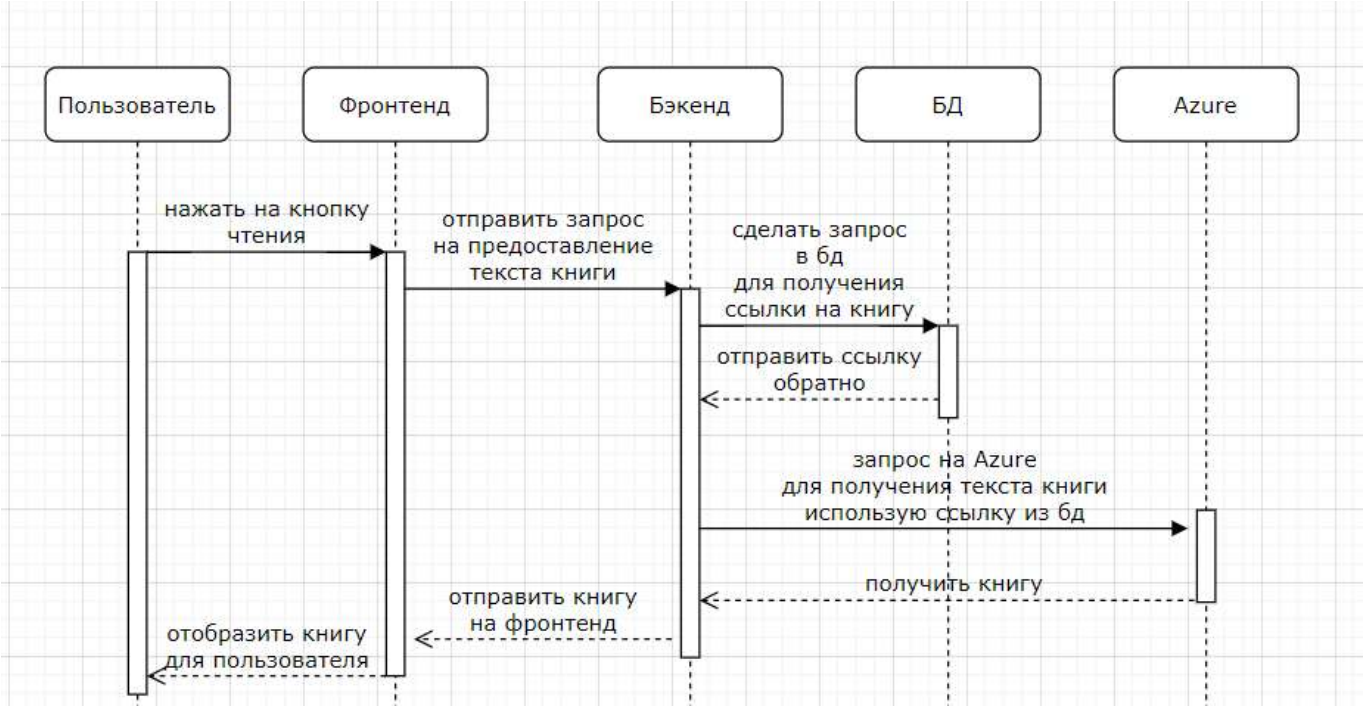

Рисунок 3.7 – Диаграмма последовательности начала чтения книги

На рисунке 3.8 представлена диаграмма кооперации для создания нового пользователя приложения. Сначала юзер должен заполнить форму на странице сайта, потом эти данные отправляются на сервер, там валидируются и после успешной валидации отправляются в базу данных. Потом ответ об успешной операции отправляется пользователю, где ему дается доступ к закрытой части приложения.

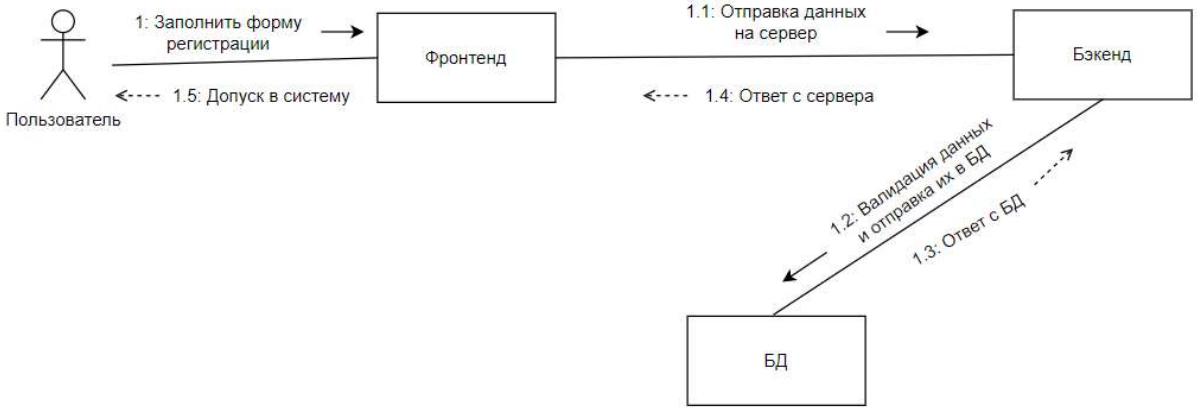

Рисунок 3.8 – Диаграмма кооперации регистрации нового пользователя

Диаграмма кооперации, представленная на рисунке 3.9 для создания нового комментария для книги в приложении. Сначала юзер должен заполнить поле ввода на странице книги, потом данные отправляются на сервер, там валидируются и после успешной валидации отправляются в базу данных. Потом ответ об успешной операции отправляется пользователю, где он уведомляется посредством пуш уведомления и сам комментарий отображается на странице.

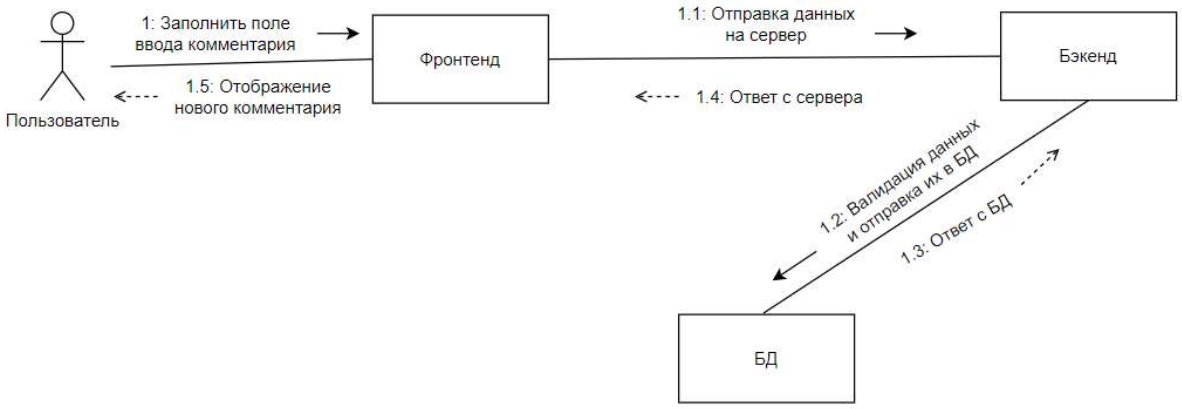

Рисунок 3.9 – Диаграмма кооперации добавления нового комментария

3.5 Статическая структура модели системы

Диаграмма классов (Class Diagram) - это один из ключевых видов диаграмм в языке моделирования UML. Диаграммы классов используются для визуализации структуры системы, описывая классы, их атрибуты, методы и отношения между ними. Эти диаграммы помогают моделировать структуру объектно-ориентированных систем и служат основой для разработки кода.

На рисунке 3.10 представлена диаграмма классов для начала чтения книги. Пользователь нажимает кнопку «Читать» на странице Book.cshtml. Отправляется запрос на бэкенд. Запрос попадает в BookController, потом данные отправляются в BookService, сервис делает запрос в базу данных для получения уникальной ссылки на файл с книгой через BookRepository. Потом сервис используя FileService (передавая ему уникальную ссылку на книгу) получает файл с Azure.

На рисунке 3.11 представлена диаграмма классов для создания нового аккаунта. Пользователь заполняет форму на странице SignIn.cshtml, запрос попадает в SignInController, потом данные отправляются в UserService и там валидируются, после валидации они отправляюся в бд с помощью UserRepository.

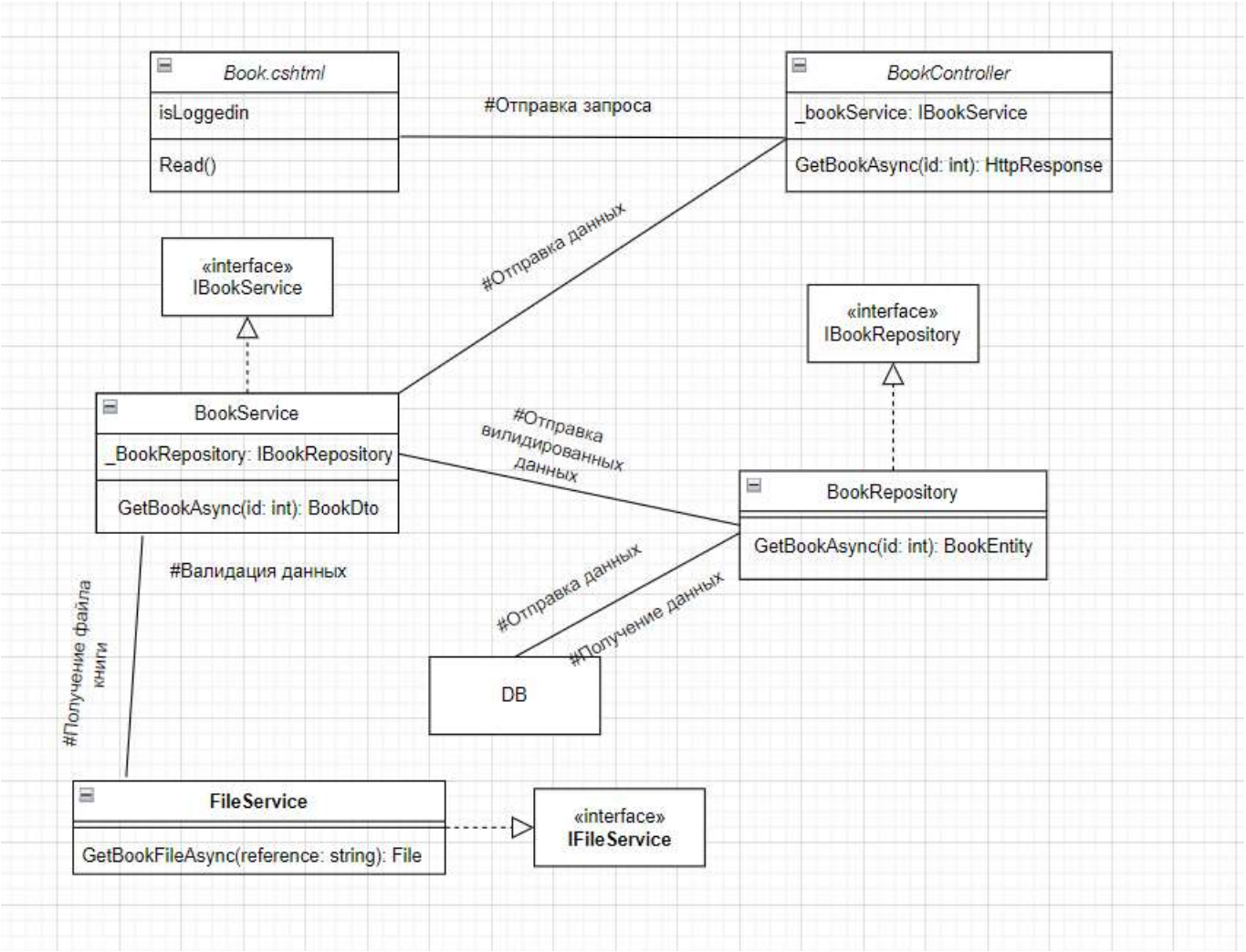

Рисунок 3.10 – Диаграмма классов начала чтения книги

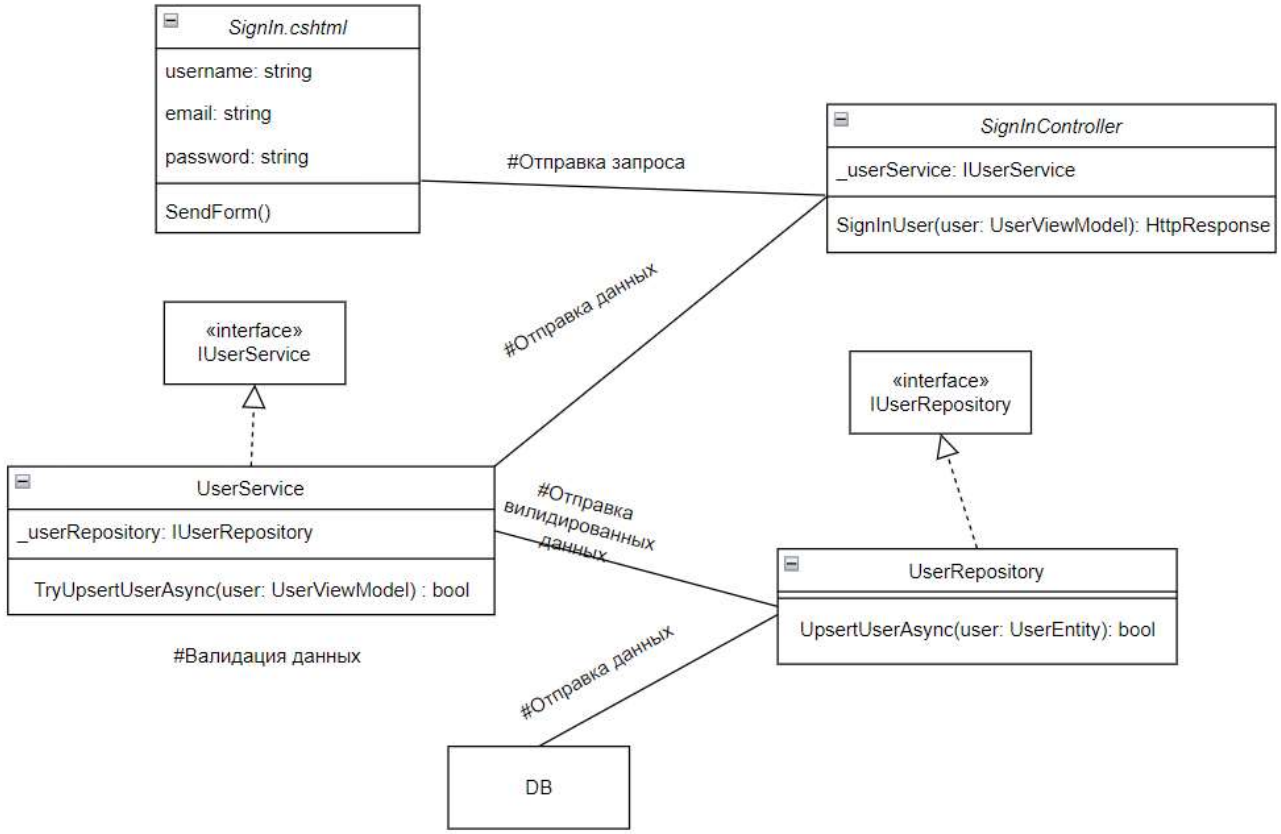

Рисунок 3.11 – Диаграмма для классов создание нового аккаунта

3.6 Моделирование поведения и процессов системы

Диаграммы состояний - это вид диаграмм в языке моделирования UML, который используется для визуализации и описания различных состояний, переходов между ними и событий, которые вызывают эти переходы в объекте или системе. Диаграммы состояний особенно полезны при моделировании поведения объектов, где важно представить их динамическое состояние. Диаграммы состояний являются мощным инструментом для моделирования и анализа динамического поведения объектов, а также обеспечивают важный вклад в разработку программных систем.

На рисунке 3.12 представлена диаграмма состояния. Для регистрации пользователь должен заполнить форму регистрации, потом она отправляется на сервер и там валидируется, если валидация проходит успешно, то пользователь вносится в базу данных, если нет, то ему высвечивает ошибку.

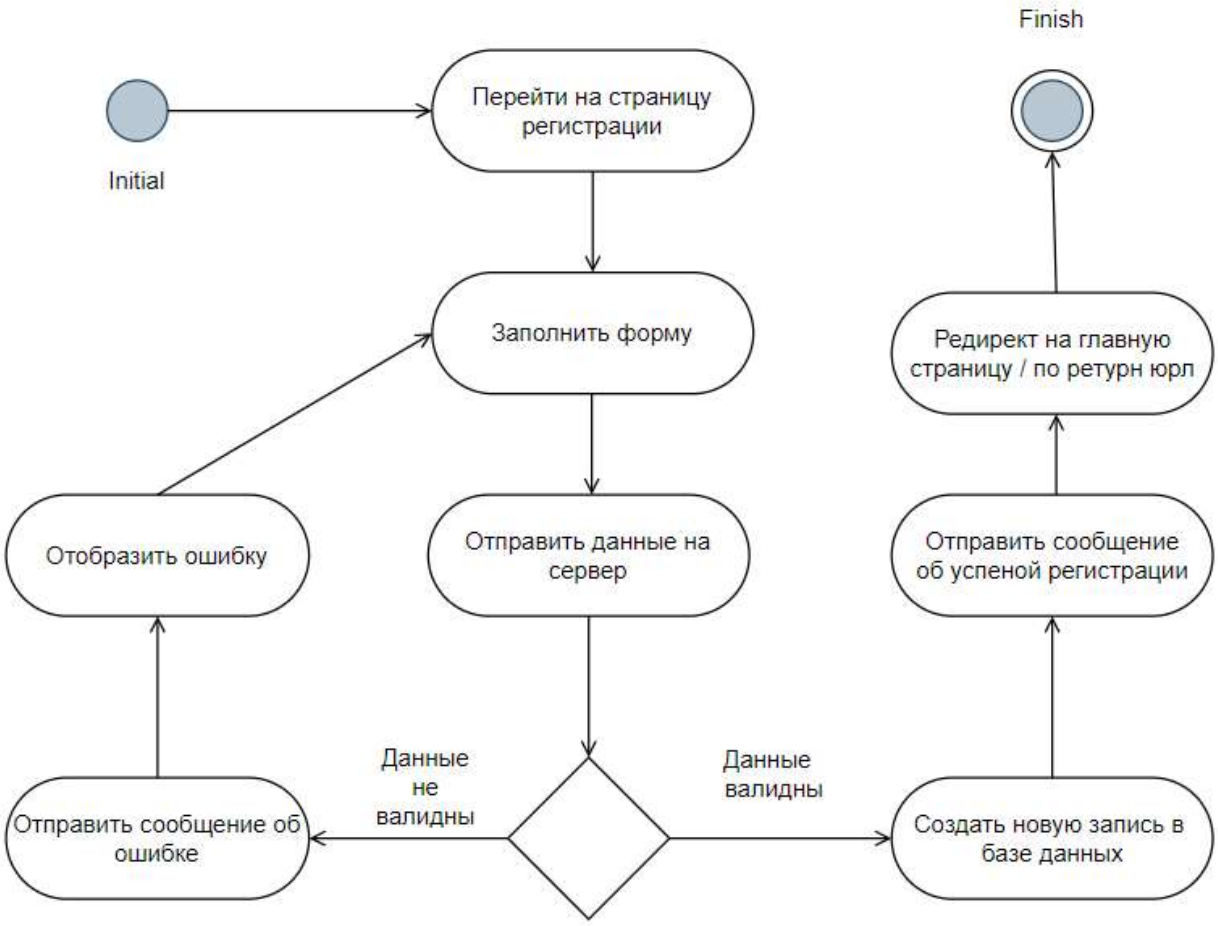

Рисунок 3.12 – Диаграмма состояния для создания нового аккаунта

Диаграмма состояния (рисунок 3.13) отображает процесс входа в аккаунт. Пользователь попадает на страницу авторизации, заполняет форму, потом она отправляется на сервер и там валидируется, если валидация проходит успешно, то созадется куки/токен авторизации, если нет, то ему высвечивает ошибку. Куки/токен сохраняется в браузере, чтобы дальнейшие запросы не нуждались в повторной авторизации.

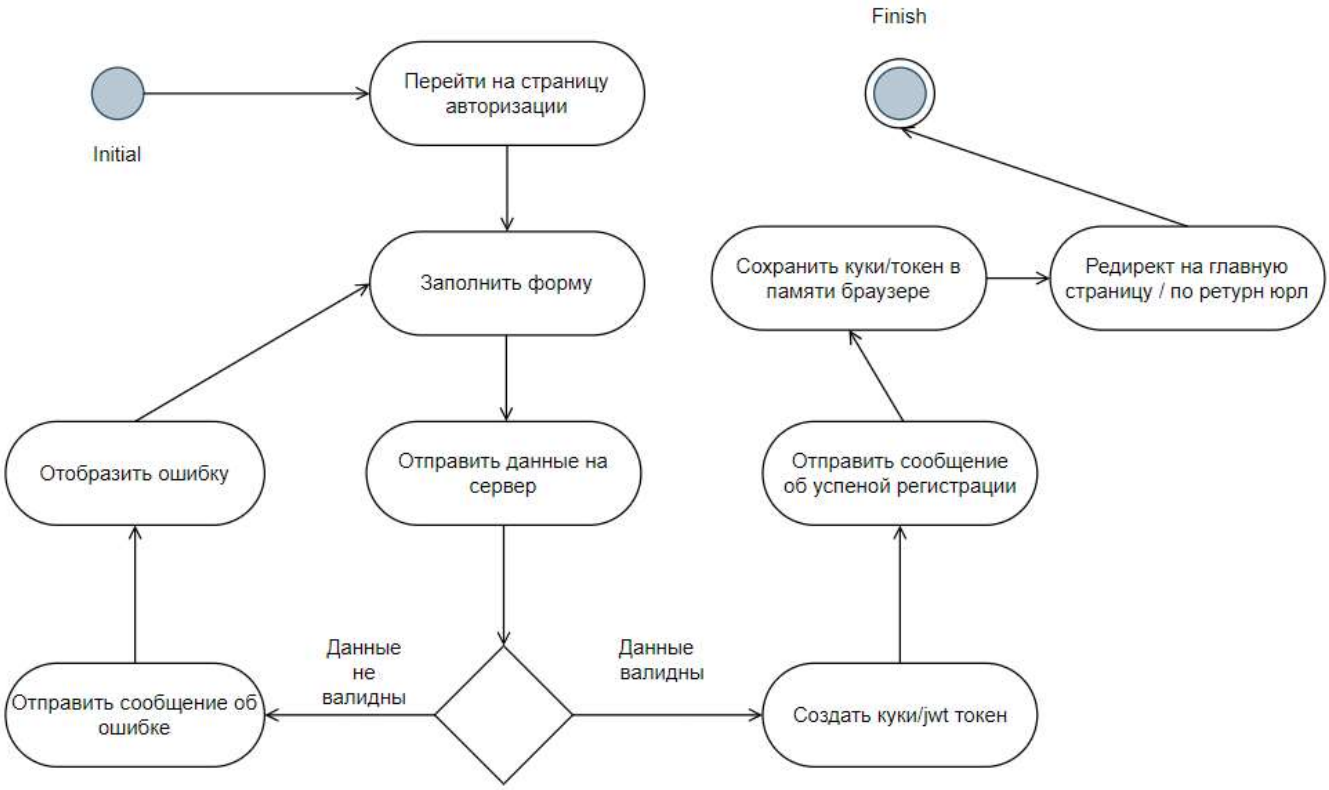

Рисунок 3.13 – Диагрмма состояния для входа в аккаунт

3.7 Моделирование физического представления системы

Диаграммы компонентов являются частью языка моделирования UML и используются для визуализации и описания высокоуровневой структуры системы, выделяя компоненты, их взаимосвязи и взаимодействие. Диаграммы компонентов часто применяются на этапе проектирования системы для представления её модульной архитектуры. Диаграммы компонентов являются важным инструментом в процессе проектирования и разработки программных систем, предоставляя архитекторам и разработчикам ясное представление о структуре системы.

На рисунке 3.14 представлена диаграмма компонентов страницы книги. По верхнему меню (нав бар) пользователь может открывать другие отделы веб-приложения. Это стандартная часть интерфейса для любой страницы приложения (кроме страницы регистрации/авторизации). Пользователь может посмотреть краткую информацию об авторе, почитать коментарии, посмотреть аннотацию к книге, ее жанр, тэги, чтобы понять, стоит ли ее читать или покупать.

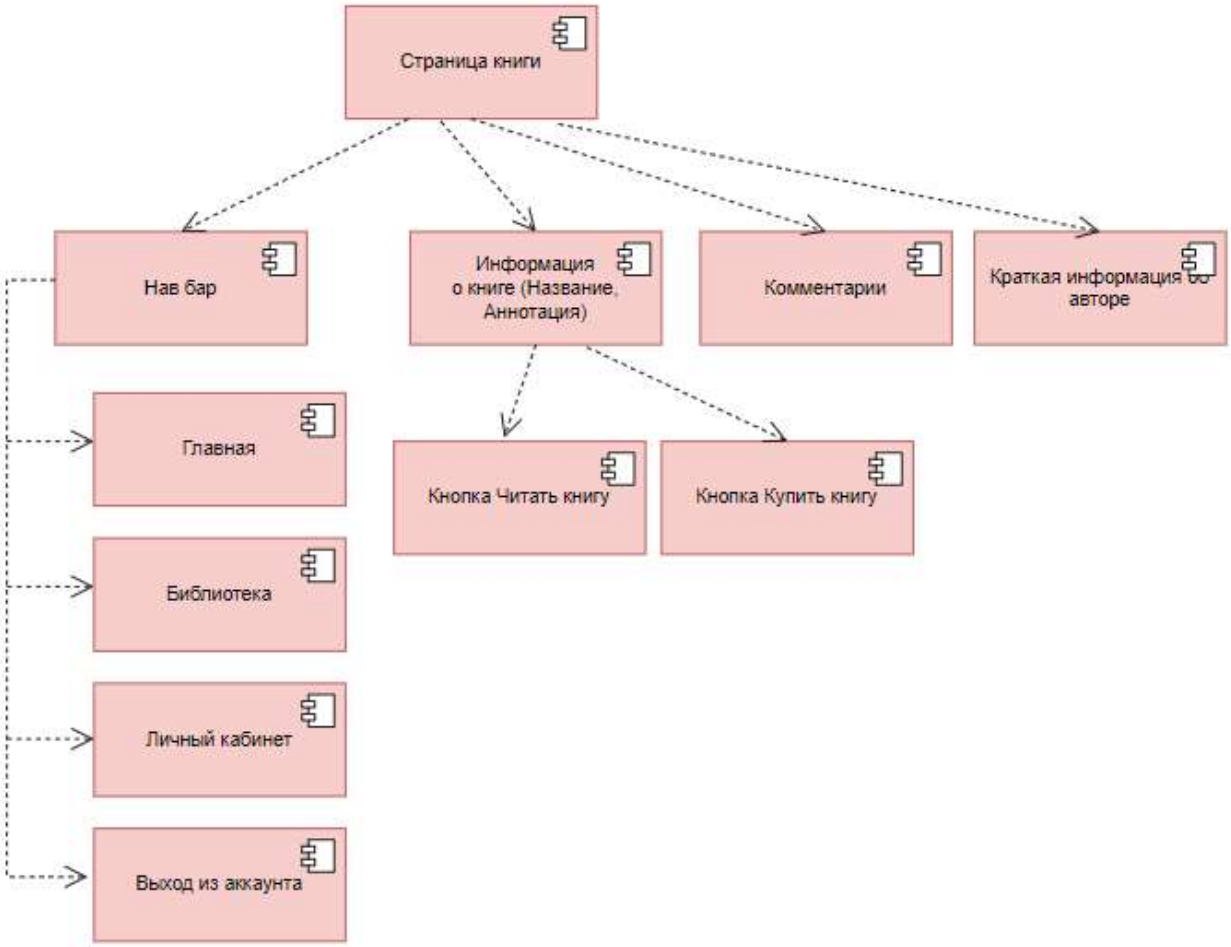

Рисунок 3.14 – Диаграмма компонентов меню

Диаграмма компонентов регистрации/авторизации (рисунок. 3.15) предостовляет нам описание интрефейса регистрации/авторизации. Пользователь может переключаться между авторизацией и регистрацией используя вкладки над формами (рисунок. 3.16), это позволяет использовать одну страницу для двух действий.

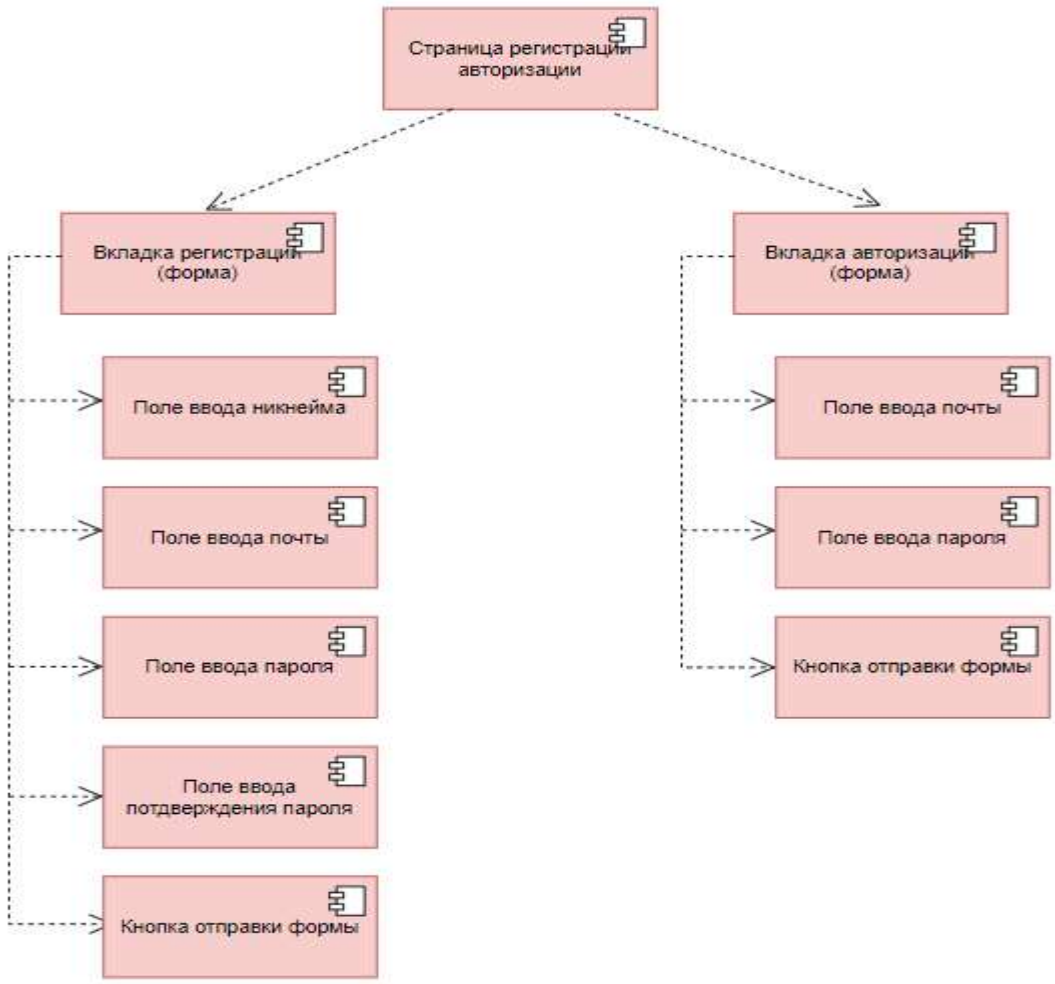

Рисунок 3.15 – Диаграмма компонентов регистрации/авторизации

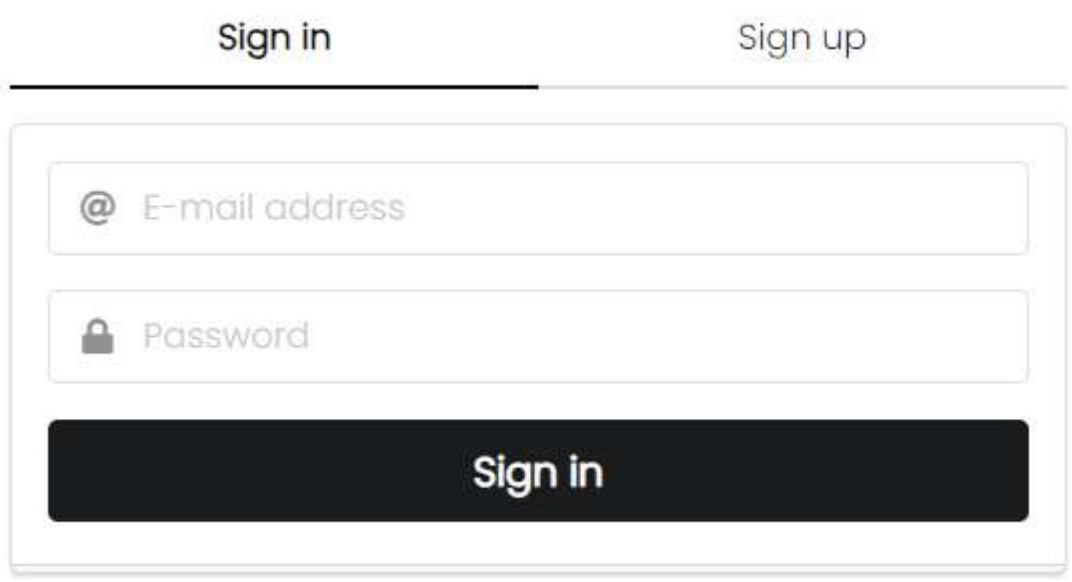

Рисунок 3.16 – Форма авторизации/регистрации

4 РЕАЛИЗАЦИЯ СИСТЕМЫ

Для реализации серверной части был выбран язык программирования C#. Использование языка программирования C# для разработки веб-приложений обосновано рядом весомых преимуществ. Во-первых, C# представляет собой язык, разработанный Microsoft, что обеспечивает высокую степень интеграции с экосистемой продуктов компании. Это позволяет разработчикам эффективно использовать инструменты и технологии, предоставляемые Microsoft, для упрощения процесса разработки и поддержки.

Одним из ключевых преимуществ C# для веб-разработки является его использование в рамках платформы .NET. Это обеспечивает платформенную независимость, позволяя приложениям, написанным на C#, успешно функционировать на различных операционных системах, таких как Windows, macOS и Linux. Таким образом, разработчики могут создавать веб-приложения, которые могут быть легко развернуты в разнообразных средах.

Вторым значительным аспектом является поддержка современных возможностей программирования, включая LINQ (Language Integrated Query), асинхронное программирование и многопоточность. Эти функциональности облегчают разработку высокопроизводительных и отзывчивых веб-приложений, способствуя эффективному использованию ресурсов сервера.

Сильная статическая типизация C# способствует выявлению ошибок на этапе компиляции, что повышает надежность кода и упрощает его поддержку. Библиотека .NET Framework предоставляет обширный набор функциональности, что ускоряет процесс разработки и уменьшает объем написания повторяющегося кода.

Наконец, постоянные обновления и адаптации C# к последним тенденциям в веб-разработке обеспечивают актуальность языка, делая его привлекательным выбором для создания современных и инновационных веб-приложений.

4.1 Страница регистрации-авторизации

Страница реистрации и авторизации были обьеденены в одну страницу, чтобы уменьшить количество повторяемого кода и по дизайнерским причинам (рисунок. 4.1). Переход между вкладками осуществляется по средством использования JavaScript, библиотеки Fomantic UI.

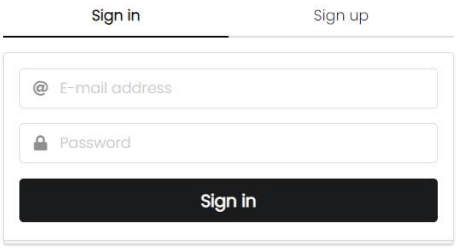

Рисунок 4.1 – Страница регистрации/авторизации

```
function setupTabs() {
    $('.menu .item').tab({
        history: true,
        onVisible: (tabPath) => {
            if (!!tabPath && tabPath === 'sign-in') {
                changePageTitle('Sign in');
            }
            else if (!!tabPath && tabPath === 'sign-up') {
                changePageTitle('Sign up');
            }
        }
    });
}
```

Используя функцию setupTabs настраивается как вкладки на странице будут себя вести. Определяется callback функция которая исполняется при отображении вкладки, она меняет название страницы для соотвествия открытой вкладки.

Для взаимодействия с сервером используется функция bindForm, которая подписывается на ивент нажатия кнопки для отправки данных с формы на сервер. Для каждой вкладки есть свой bindForm.

```
bindFormSubmit({
    btnId: 'sign-in-btn',
    formId: 'sign-in-form',
    methodFn: () => {
        return 'post';
    },
    urlFn: () => {
```

```
            return '/account/sign-in';
        },
        onSuccessFn: (response) => {
            const searchParams = new URLSearchParams(window.location.search);
            const returnUrl = searchParams.get('returnUrl');

            if (!!returnUrl) {
                goTo(returnUrl);
            } else {
                goTo('/');
            }
        }
});
```

На сервере запрос принимает endpoint который принаджит контроллеру аккаунтов. Endpoint отправляет запрос дальше по иерархии к сервису.

```
[AllowAnonymous]
[Route("/account")]
[ApiExplorerSettings(IgnoreApi = true)]
public class AccountController : BaseController
{
    private readonly IAccountService _accountService;

    public AccountController(IAccountService accountService)
    {
        _accountService = accountService;
    }

    [HttpGet("login")]
    public IActionResult SignInUp([FromQuery] string returnUrl)
    {
        if (User.Identity != null && User.Identity.IsAuthenticated)
        {
            return RedirectToHome();
        }

        return View();
    }

    [HttpPost("sign-up")]
    public async Task<IActionResult> SignUp(SignUpViewModel viewModel)
    {
        return DynamicResultResponse(await
_accountService.TrySignUpAsync(viewModel));
    }

    [HttpPost("sign-in")]
    public async Task<IActionResult> SignIn(SignInViewModel viewModel)
    {
        return DynamicResultResponse(await
_accountService.TrySignInAsync(viewModel));
    }

    [Route("sign-out")]
    [Authorize]
    public async Task<IActionResult> Logout()
    {
```

```
            if (_accountService.IsUserAuthenticated())
            {
                await _accountService.SignOutAsync();
                return RedirectToHome();
            }

            return ErrorResponse(null);
        }
    }
```

4.2 Страница создания книги

На страницу могут зайти только аунтефицированные пользователи. На странице представлена форма создания новой книги, Название, Аннотация, Файл с книгой, Файл с обложкой (рисунок. 4.2).

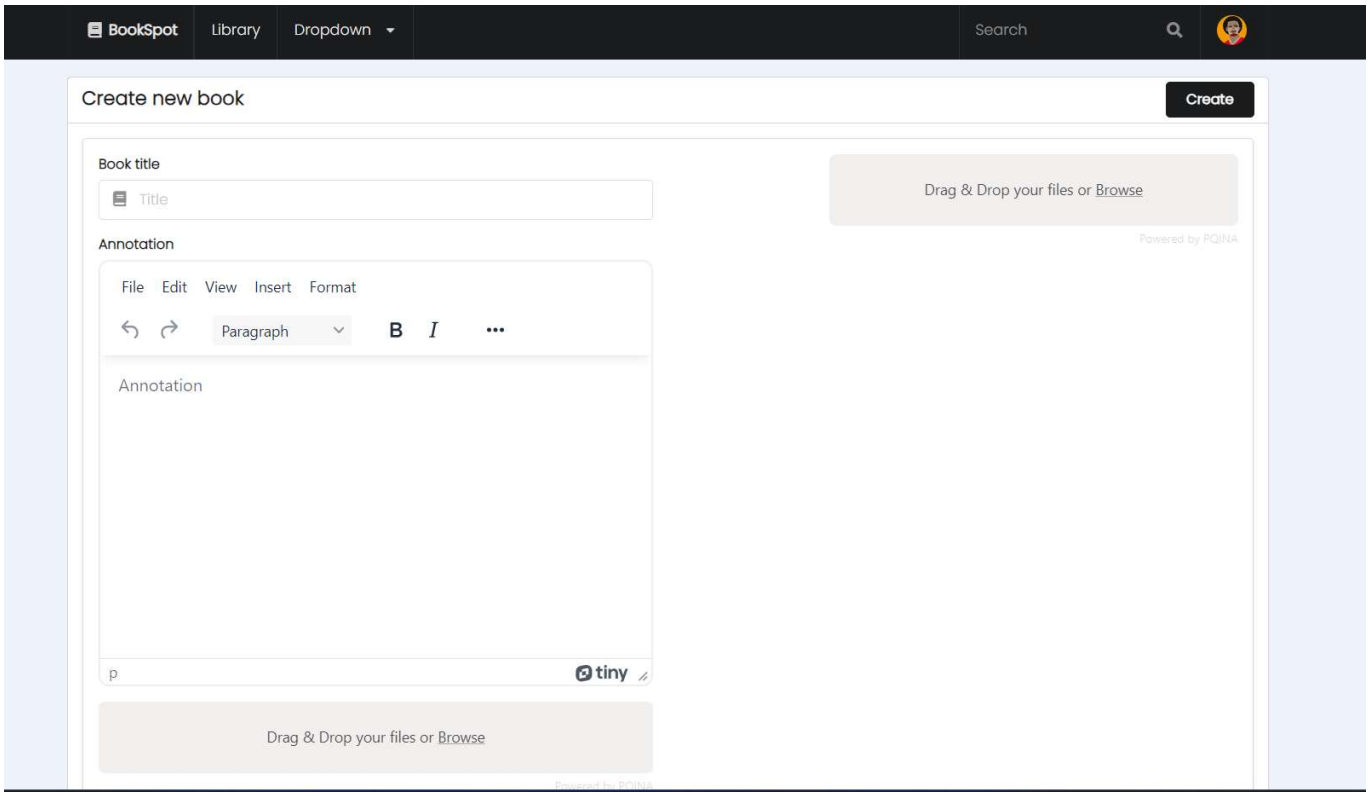

Рисунок 4.2 – Страница создания книги

Для пользовательского интерефейса использована JavaScript библиотека Fomantic UI, но для аннотации была взята библиотека TinyMCE для создания wysiwyg эдитора. Для работа с файлами была использванна библиотека Filepond.

Инициализация инстанса TinyMCE необходимо несколько шагов. Сначала добавить html код на страницу, потом через JavaScript указать фреймворку что этот html код для него.

```
<div class="field">
```

```
    <label>Annotation</label>
    <div>
        <textarea id="annotation" type="text" name="description"
placeholder="Annotation"></textarea>
    </div>
</div>

tinymce.init({
    selector: 'textarea#annotation'
});
```

Для инициализации Filepond использовалось практически тоже самое с небольшими отличиями.

```
<div class="field">
    <input type="file"
           id="book-file"
           class="filepond"
           name="filepond"

accept="@string.Join(",",appSettings.FileValidationSettings.BookFile.Allo
wedExtensionTypes
.Concat(appSettings.FileValidationSettings.BookFile.AllowedAttachmentType
s))" />
</div>


function registerBookFileFilepond() {
    FilePond.registerPlugin(
        FilePondPluginFileValidateType,
        FilePondPluginFileValidateSize
    );

    bookFileFilepond = FilePond.create(
        document.getElementById('book-file'),
        {
            maxFileSize: _get('bookFileMaxSizeInMb'),
            fileValidateTypeDetectType: (source, type) =>
                new Promise((resolve, reject) => {
                    if (source.name.toLowerCase().indexOf('.fb2') !== -1)
{
                        return resolve('application/x-fictionbook');
                    }

                    resolve(type);
                }),
            fileValidateTypeLabelExpectedTypes: 'Expects {allButLastType}
or {lastType}'
        }
    );
```

Чтобы задать необходимые ограничения на файл html инпуту предоставлялись доспустимые mime типы и расширения файла с сервера. Использовался C# код, т.к расширение html файла - .cshtml. Это позволяет использовать C# код в нем. Функция registerBookFileFilepond использовалась для инициализации инстанса Filepond. Необходимо сначала указать используемые

плагины, потом использовать функцию create. Из-за проблем с расширением .fb2, что является электронной книгой задан собственный обработчик валидации файла.

Для отправки на сервер всей информации использовался уже известная функция bindForm.

```
function bindForms() {
    bindFormSubmit({
        btnId: 'create-book-btn',
        formId: 'book-form',
        methodFn: () => {
            return 'post';
        },
        urlFn: () => {
            return `/book`;
        },
        onSuccessFn: (response) => {
            showSuccess('Success');

            if (response.data) {
                goTo(`/book/${response.data.bookId}/view`)
            }
        },
        formDataBeforeSubmitFn: (formData)  => {
            let bookCoverFiles = bookCoverFilepond.getFiles();
            if (!!bookCoverFiles[0]) {
                formData.append('bookCoverImage',
bookCoverFiles[0].file);
            }

            let bookFiles = bookFileFilepond.getFiles();
            if (!!bookFiles[0]) {
                formData.append('bookFile', bookFiles[0].file);
            }

            return formData;
        },
        beforeFormValidationFn: () => {
            let tinymceContent = tinymce.activeEditor.getContent();
            $('#annotation').val(tinymceContent);
        }
    });
```

В данном случае, чтобы отправить данные с формы на сервер необходимо сначала вытазить данные из tinymce инстанса и двух filepond инстансов. Для этого использовались функции-коллбэки formDataBeforeSubmitFn и beforeFormValidationFn.

4.3 Страница книги

Страница книги (рисунок. 4.3) содежит несколько важных отделов. Отдел описания книги: название, автор, аннотация, обложка. Отдел автора (еще не реализовано). Отдел комментариев: где

использован инстанс tinymce, а так же инстанс Vue для реактивности. Отдел глав: где использован еще один инстанс Vue.

```
<div id="comment-app" class="m-top-15">
    <div v-if="comments.length">
        <div class="ui comments" v-for="comment in comments">
            <div class="comment">
                <a class="avatar">
                    <img src="/images/avatar/small/joe.jpg">
                </a>
                <div class="content">
                    <a class="author">{{ comment.authorName }}</a>
                    <div class="metadata">
                        <span class="date">{{ comment.createdAt }}</span>
                    </div>
                    <div class="text">
                        <div v-html="comment.text"></div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div v-else>
        <div class="ui segment">
            There are no comments yet.
        </div>
    </div>
</div>

commentApp = new Vue({
    el: '#comment-app',
    data: {
        bookId: _get('bookId'),
        comments: _get('comments'),
    },
    mounted: function () {
        bindForms();
        setupFormValidation();
    },
    methods: {}
});
```

Данный код использован для отображения комментариев на странице используя реактивность Vue. Каждый раз когда с сервера приходит положительный ответ о создании нового комментария – изменяется переменная comments в инстансе Vue commentApp. Инстанс Vue видит изменение и перегенерирует html для указанного куска страницы. Также использовалась функция mounted инстанса Vue для применения функций после генерации инстанса. В данном случае bindForm для отправки формы нового комментария на сервер и setupFormValidation чтобы задать правила валидации для этой формы. По html коду видно использование директивы v-for для отображения коллекций на странице, в данном случае коллекции comments, а также директивы v-if для отображения комментариев только тогда когда есть хоть однин комментарий, в противном должна отображаться заглушка.

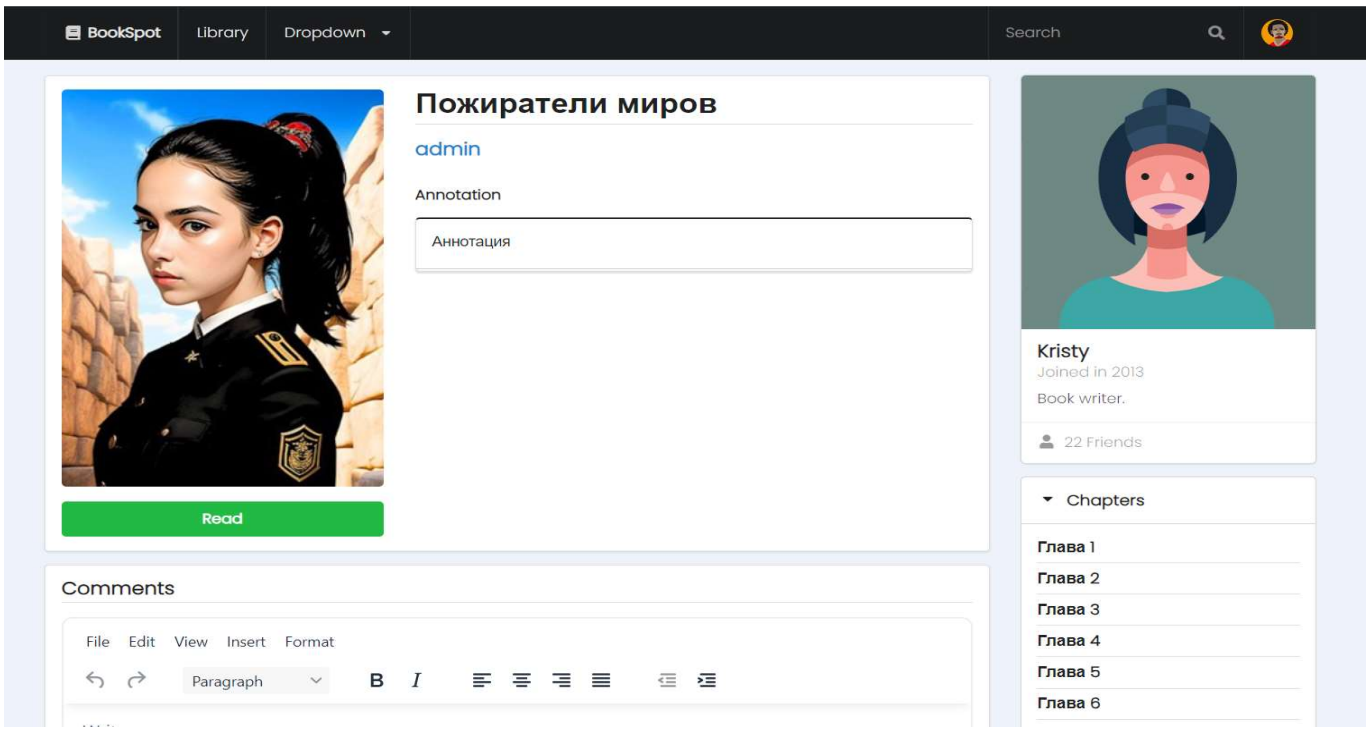

Рисунок. 4.3 – Страница книги

4.4 Страница главы

Страница книги (рисунок. 4.4) состит из двух важных отделов: хедера, который отличается от обычного хедера сайта; текущий хедер создан конкретно для удобства чтения пользователя; и самого отдела с контентом книги.

Хедер содержит в себе навигационное меню, которое позволяет вернуться назад к читаемой книге, передвигаться по главам, благодаря выпадающему списку с этими самыми главами и кнопкой настроек справа (еще не реализованно).

Контент в себе содердит название главы и содержание главы, которое в свою очередь является html кодом хранящемся в базе данных. Так же в конце страницы находятся две кнопки для навигации между рядом стоящими главами (рисунок. 4.5).

```
<div class="ui main text container">
    <div class="ui black stacked segment">
        <h1 class="ui header center aligned">@chapter.Title</h1>
    </div>
    <div class="ui segment">
        @Html.Raw(chapter.Content)
    </div>
    <div id="nav-btns-app" class="ui segment">
        <div class="actions-container">
            <div class="actions spaced">
                <button class="ui labeled icon black button"
                        :class="{ disabled: isFirstChapter }"
```

```
                        @@click="goToChapter(@chapter.BookId,
@previousChapterId)">
                    <i class="left arrow spinner icon"></i>
                    Back
                </button>
                <button class="ui right labeled icon black button"
                        :class="{ disabled: isLastChapter }"
                        @@click="goToChapter(@chapter.BookId,
@nextChapterId)">
                    <i class="right arrow icon"></i>
                    Next
                </button>
            </div>
        </div>
    </div>
</div>

navigationBtnsApp = new Vue({
    el: '#nav-btns-app',
    data: {
        isFirstChapter: _get('isFirstChapter'),
        isLastChapter: _get('isLastChapter'),

    },
    mounted: function () {
    },
    methods: {
        goToChapter: function (bookId, chapterId) {
            goTo(`/book/${bookId}/chapter/${chapterId}/view`);
        }
    }
});
```

Здесь можно увидеть инстанс Vue для работы кнопок навигации. Функция goToChapter обеспечивает эту навигацию, но для доступа к этой функции, которая активируется по средством нажатия на кнопку на странице, необходимо чтобы кнопка была активна. За работоспособность кнопок отвечает две переменные isFirstChapter и isLastChapter. Они предотвращают возможность пользователя переходить к главе которой не существует.

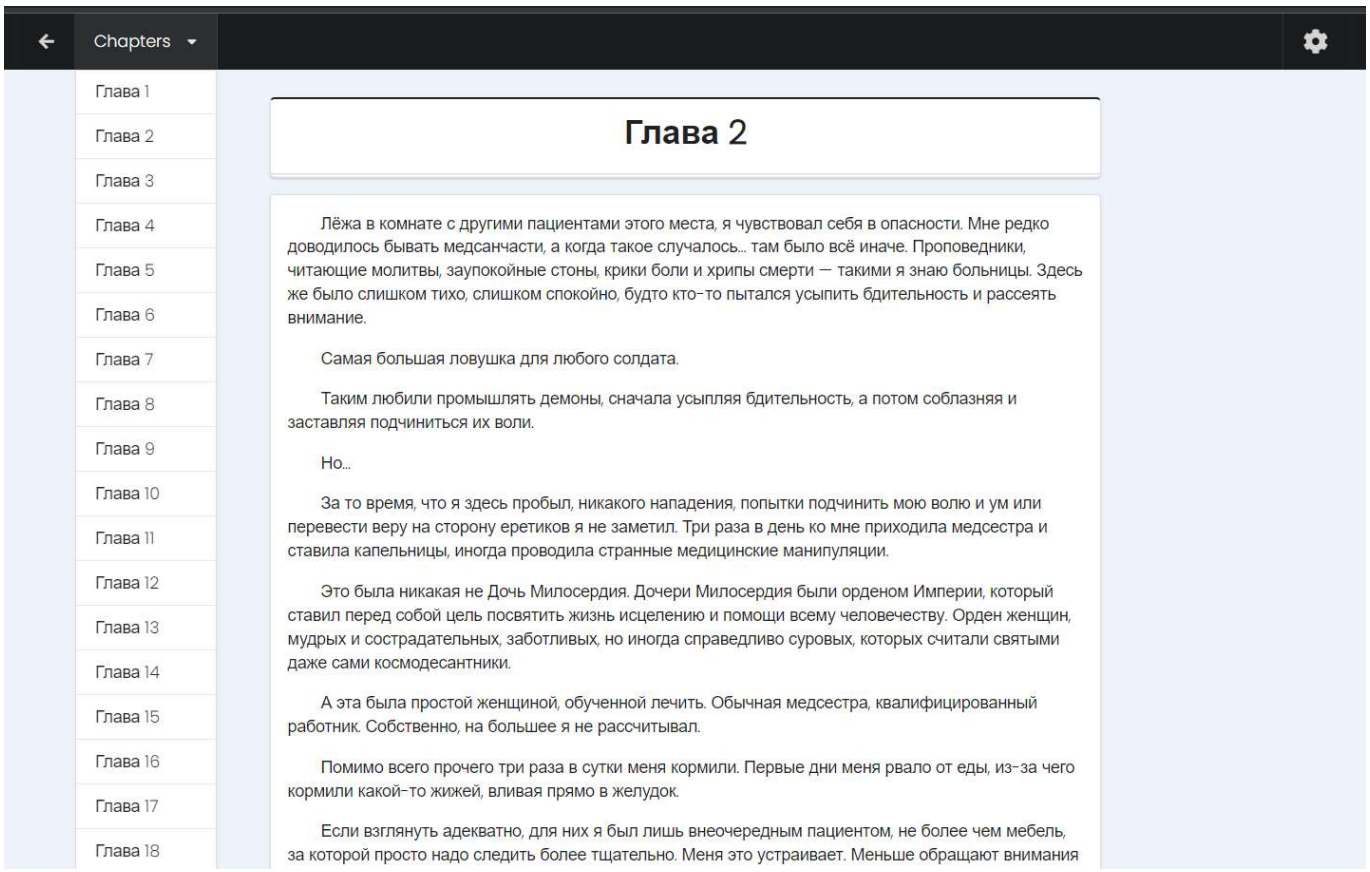

Рисунок. 4.4 – Страницы главы

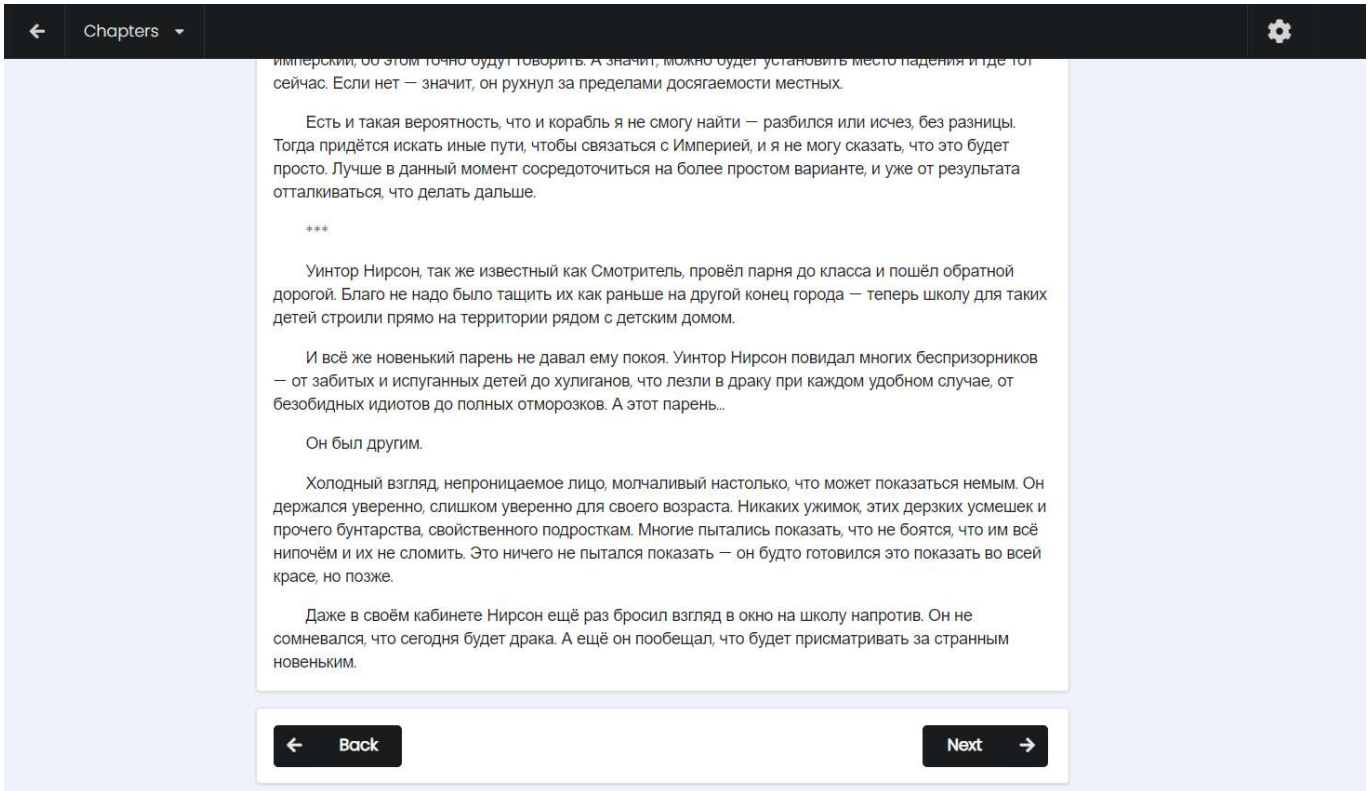

Рисунок. 4.5 – Кнопки навигации на странице главы

5 ОЦЕНКА СТОИМОСТИ

Work Breakdown Structure (WBS) — это методология структурирования и организации проекта на более управляемые и управляемые компоненты. WBS представляет собой древовидную декомпозицию проекта на более мелкие и управляемые элементы, что облегчает понимание общего объема работы и планирование ресурсов. Каждый уровень WBS представляет собой более детализированный уровень задач.

5.1 Декомпозиция разрабатываемой системы

Разработка практического любого веб-приложения (мой проект не исключение) состоит из 4-х основных этапов (рисунок 5.1):

- разработка фроненда: разработка пользовательской части приложения, например UI (визуальная часть приложения)
- разработка бэкенда: разработка серверной части приложение, которое обеспечивает прием запросов с фроненда, их обработки и ответ. Отвечает за взаимодействие с базой данных и сторонними сервисами
- проектирование базы данных: является одним из важнейших оспектов разработки приложения, хорошо спроектированная база данных влияет на производительность всего приложения, а на это влияет оптимизированность запросов, оптимизированность sql, правильные индексы, правильная иерархия таблиц
- интеграция сторонних сервисов и библиотек: важный этап, который описывает как приложение взаимодействует с другими приложениями, например Azure DevOps, Microsoft Entra ID

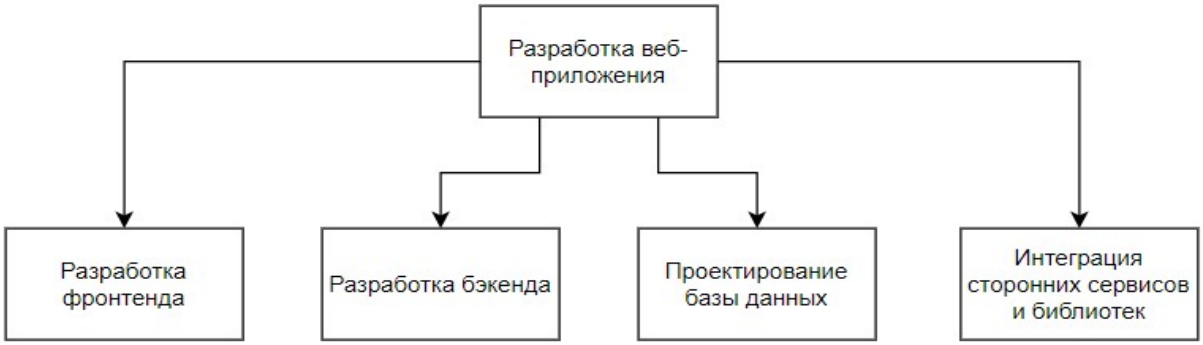

Рисунок 5.1 – Основные этапы разработки веб-приложения

Ниже представлена диаграмма декомпозиции проекта на иерархию (рисунок 5.2), где есть подготовка (проектирвоание) проекта, проектирование базы данных, разработка проекта и его тестирование с последующим запуском.

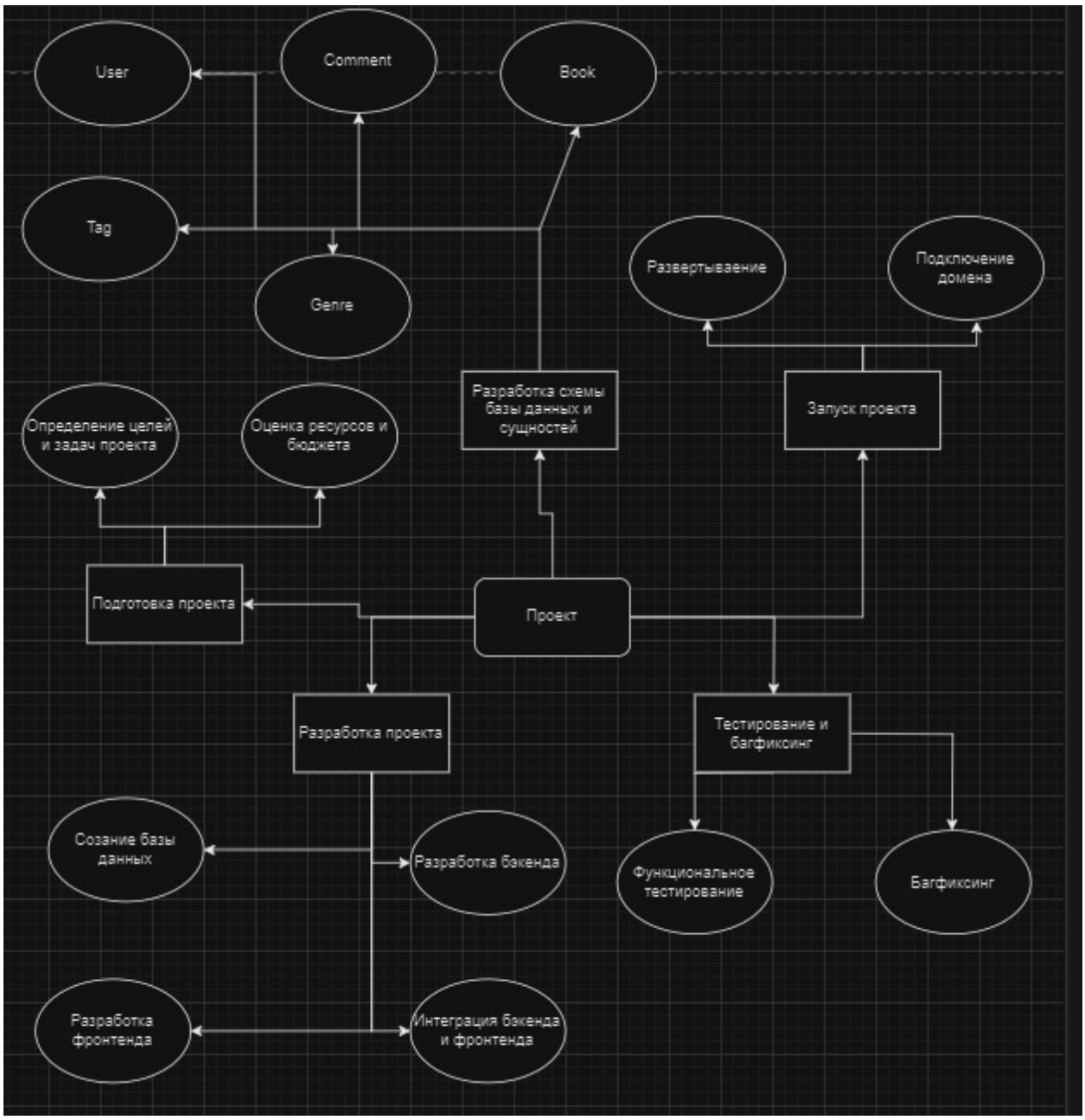

Рисунок 5.2 – WBS (Work Breakdown Structure) диаграмма декомпозиции проекта на иерархию

5.2 Планирование реализации требований

На данном этапе определяем временные рамки для каждого этапа разработки. Выделяем такие ресурсы как разработчики и дизайнеры, а также проводим тестирование нашего продукта. Ниже, на рисунке 5.3, представлена диаграмма Ганта по WBS, на которой каждому этапу выделили определённое время на его реализацию.

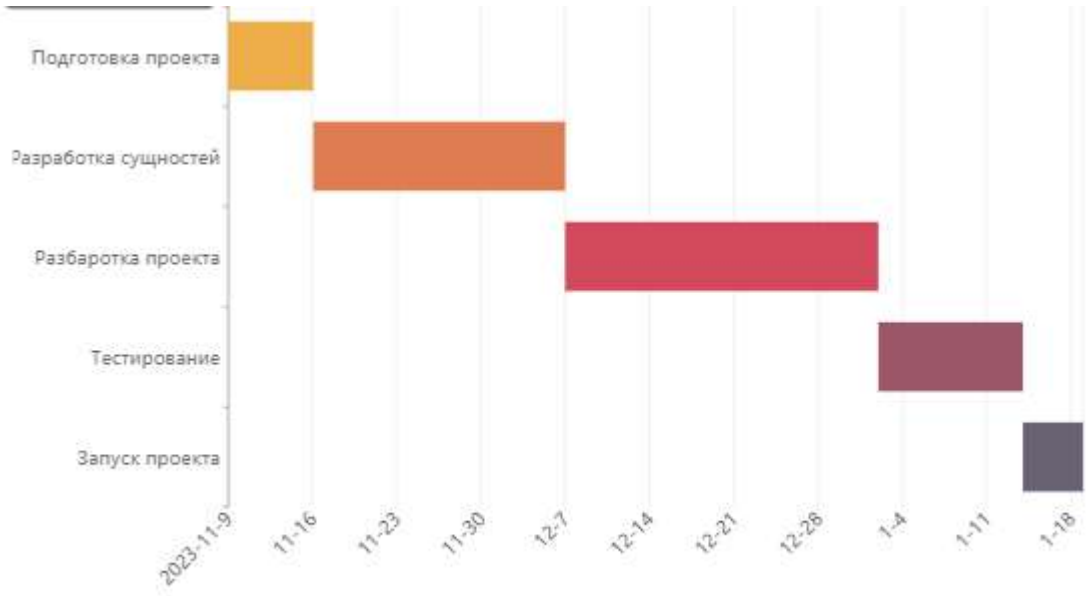

Рисунок 5.3 – Диаграмма Ганта по WBS

5.3 Оценка стоимости проекта

Определение стоимости проекта — это процесс оценки и вычисления всех расходов, необходимых для выполнения проекта. Этот процесс включает в себя анализ всех затрат, включая трудозатраты, материальные ресурсы, оборудование, программное обеспечение, внешние услуги и другие расходы, связанные с реализацией проекта. Цель определения стоимости проекта заключается в создании бюджета, который будет использоваться для управления финансами, контроля затрат и обеспечения финансовой устойчивости проекта.

Расчет

Для каждого этапа разработки нужно подсчитать дни и часы, затраченные одним работником. В среднем один рабочий день длится 8 часов. Оценим один час работы в $10.

- подготовка проекта: 7 дней × 8 часов/день × $10/час = $560
- разработка базы данных и сущностей: 21 дней × 8 часов/день × $10/час = $1680
- разработка проекта: 26 день × 8 часов/день × $10/час = $2080
- тестирование и багфиксинг: 12 дней × 8 часов/день × $10/час = $960
- запуск проекта: 5 дней × 8 часов/день × $10/час = $400

Общая стоимость за разработку функциональности составляет:

$$О = П + Р1 + Р2 + Т + З, \quad (4.1)$$

где

О – общая стоимость зарплат;

П – проектирование;

Р1 – разработка базы данных;

Р2 – разработка самого проекта

Т – тестирование;

З – запуск проекта.

О = $560 + $2080 + $1680 + $960 + $400 = $5680

Оценочную стоимость проекта можно вычислить по формуле:

C = S + Э + E + Eprob,

где

C – оценочная стоимость;

S – затраты на заработную плату за разработку функциональности и дизайна;

E – лицензии или оборудование;

Eprob – затраты на дополнительные расходы.

Затраты на электричество

Для определения окончательной стоимости проекта необходимо учесть не только расходы на оплату труда сотрудников, но также затраты на электроэнергию, особенно учитывая, что в рамках данного проекта, связанного с разработкой программного продукта, существуют затраты на рабочие ноутбуки, потребляющие значительное количество электроэнергии. При оценке стоимости проекта следует учесть эти факторы. Для расчета затрат на электроэнергию важно учитывать следующие аспекты:

- потребление энергии каждым устройством, задействованным в проекте;
- время, в течение которого каждое устройство будет использоваться в рамках проекта;
- стоимость потребленной электроэнергии на местном энергорынке.

Для определения затрат на электроэнергию в рамках данного проекта необходимо выполнить следующие шаги:

- составить список всех используемых в проекте устройств и указать их потребление электроэнергии в ваттах (W) или киловатт-часах (kWh);
- определить общее время использования каждого устройства в рамках проекта;

– установить стоимость потребленной электроэнергии на местном рынке, уточнив цену за киловатт-час или за тарифную единицу.

Путем анализа этих данных можно определить примерные затраты на электроэнергию и включить их в общую стоимость проекта.

Затраты на электроэнергию определяются с учётом тарифа в Бельцах – 4 лея за 1 КВт*ч и по формуле (4.2):

$$q = W * t, \quad (4.2)$$

где

q – расчёт затраченной электроэнергии за все время;

W – мощность одного компьютера, кВт;

t – время работы в часах.

t = 568 часов,

q = W *t = 0,60 кВт * 568 часа = 346,48 кВт*час,

$$Э = Т * q, \quad (4.3)$$

где

Э – затраты на электроэнергию;

Т – тариф на электроэнергию.

На основе полученной информации можно подсчитать расходы в молдавских леях на электроэнергию, которая была затрачена в процессе разработки проекта. Используя формулу (4.3):

Э = Т * q = 4 * 346,48 = 1385,92 (лея).

Для разработки дипломного проекта затраты по электроэнергии составляют 1385,92 лей.

Оценка дополнительных расходов

Здесь учитываются различные дополнительные расходы, которые могут возникнуть в результате непредвиденных обстоятельств, например, поломка компьютера:

– нижняя оценка: 0% от затрат на оплату труда;
– наиболее вероятная оценка: 10% от затрат на оплату труда;
– верхняя оценка: 25% от затрат на оплату труда.

Оценка дополнительных расходов вычисляется по формуле (4.4):

$$Eprob = 0 * A + * 0.1 * A + 0.25 * A, \quad (4.4)$$

где

A – затраты, которые включаю в себя затраты на заработную плату сотрудников, лицензии и электроэнергию. Так как в разработке проекта использовались только бесплатные инструменты, библиотеки и программы, то затраты на лицензию составляют 0 лей. Так же, в разработке проекта не нужны компоненты и материалы, поэтому E = 0 лей.

A = S + Э + E;

A = 102240 + 1385,92 + 0 = 103625,92 лей.

Подставляя полученные затраты (A) в формулу (4.4), можно рассчитать стоимость дополнительных расходов:

Eprob = (0% * 103625,92 + 10% * 103625,92 + 25% * 103625,92) = 36268,98 (лей).

Учитывая все перечисленные затраты, такие как заработные платы сотрудников, расходы на электроэнергию, затраты на лицензии для создания программного продукта, а также неожиданные расходы, окончательная оценочная стоимость проекта может быть рассчитана с использованием формулы (1), где происходит суммирование всех учтенных компонентов.

C = 102240 + 1385,92 + 0 + 36268,98 = 139894,93 (лей).

Таким образом, окончательная стоимость данной логической мобильной игры оценивается в 139894,93 лей.

ЗАКЛЮЧЕНИЕ

В заключение, разработка веб-приложения на языке программирования C# для чтения книг представляет собой результат комплексного инженерного подхода. Реализованный проект предоставляет пользователю удобный интерфейс для доступа к литературным произведениям, обеспечивая при этом эффективное взаимодействие с контентом. Отдельное внимание уделено анализу требований и проектированию системы, что позволило создать структурированное и масштабируемое веб-приложение.

Процесс разработки включал в себя эффективное использование технологий и принципов языка C#, что способствовало высокой производительности и надежности приложения. Результаты тестирования подтвердили соответствие функциональности приложения заявленным требованиям, а внедрение современных практик безопасности обеспечивает устойчивость к возможным угрозам.

Обнаруженные в процессе разработки проблемы были тщательно исследованы, и приняты меры по их решению. Отмечается, что разработанное веб-приложение предоставляет пользователю интуитивно понятный интерфейс, а использование технологии C# дает возможность эффективного масштабирования и дальнейшего развития системы.

В целом, создание веб-приложения на C# для чтения книг подчеркивает важность современных методологий разработки и применения передовых технологий для достижения высокого уровня функциональности и удовлетворения потребностей пользователей. Полученные результаты могут служить основой для дальнейших исследований в области веб-разработки и оптимизации пользовательского опыта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ, Описание предметной области, Обзор существующих программных продуктов, Выбор и обоснование варианта реализации задач. Студенческая библиотека онлайн, ©2023.
Режим доступа: https://studbooks.net/2247494/informatika/analiz_predmetnoy_oblasti

[2] Microsoft Documentation, Microsoft. ASP.NET Core Documentation.
Режим доступ: https://docs.microsoft.com/en-us/aspnet/core/

[3] Metanit. ASP.NET, Руководство по ASP.NET Core 8.
Режим доступ: https://metanit.com/sharp/aspnet6/

[4] Metanit. Полное руководство по языку программирования C# 12 и платформе .NET 8
Режим доступ: https://metanit.com/sharp/tutorial/

[5] БЕШЛИУ, В., КИРЕВ, П., СКОРОХОДОВА, Т. Методическое руководство по реализации лабораторных работ по курсу: Проектирование информационных систем (инструменты, руководства и методологические вопросы). Кишинёв, 2023. с.2-73.