**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет *ИСТ* Кафедра *ИВК*

К ЗАЩИТЕ ДОПУСТИТЬ

Зав. кафедрой

/ С.К. Киселев /

подпись инициалы, фамилия

« » 20 24 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема *Интеллектуальный помощник для работы с электронной библиотекой*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ *Х.А. Нгуен*

подпись инициалы, фамилия

Обозначение работы *ВКР-УлГТУ-09.03.02-17/1175-2024* Группа *ЦИСТбв-51*

Направление подготовки *09.03.02 «Информационные системы и технологии»*

код, наименование

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель ВКР / *В.М. Кандаулов* /

подпись, дата инициалы, фамилия

Консультанты:

*экономический раздел* / *М.В. Рыбкина* /

наименование раздела подпись, дата инициалы, фамилия

Ульяновск, 2024

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет  *ИСТ*  Кафедра  *ИВК* .

Направление подготовки *Информационные системы и технологии* .

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ |
|  | Зав. кафедрой |
|  | \_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / |
|  | подпись инициалы, фамилия |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

ЗАДАНИЕ

**на выпускную квалификационную работу**

обучающемуся  *Нгуену Хыу Ану* курса  *5*  группы  *ЦИСТбв-51*

фамилия, имя, отчество

1. Тема работы *Интеллектуальный помощник для работы с электронной библиотекой*утверждена приказом по университету №\_\_\_\_\_\_\_ от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.
2. Срок сдачи обучающимся законченной ВКР «*22*» *июня* 2024 г.
3. Исходные данные к работе *разработать систему* *персональной электронной библиотеки работающей в локальной сети*
4. Содержание пояснительной записки *задание на создание системы; информационное, алгоритмическое, программное обеспечение системы; тестирование системы*
5. Перечень графического материала (с точным указанием обязательных чертежей) *схема алгоритма обмена информацией между серверной частью и API ИИ, схема алгоритма добавления электронной книги в библиотеку.*
6. Календарный график работы над ВКР на весь период (с указанием сроков выполнения и содержания отдельных этапов)

|  |  |  |
| --- | --- | --- |
| № этапа | Содержание этапа | Срок выполнения |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
|  |  |  |

1. Консультанты

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Ф.И.О. консультанта | Подпись, дата | |
| Задание выдал | Задание принял |
| *Экономический* | Рыбкина М.В. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата выдачи задания «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_/ *В.М. Кандаулов* /

должность, учёная степень, ученое звание подпись инициалы, фамилия

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / *Х.А. Нгуен* /

подпись обучающегосяинициалы, фамилия

Содержание

[CПИСОК ИСПОЛЬЗОВАННЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ 7](#_Toc168055567)

[ВВЕДЕНИЕ 8](#_Toc168055568)

[1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ СИСТЕМЫ 10](#_Toc168055569)

[1.1 Назначение и цели создания системы 10](#_Toc168055570)

[1.2 Характеристика объекта автоматизации 10](#_Toc168055571)

[1.2.1 Общее описание 10](#_Toc168055572)

[1.2.2 Состав и порядок функционирования 10](#_Toc168055573)

[1.3 Общие требования к системе 11](#_Toc168055574)

[1.3.1 Требования к структуре и функционированию системы 11](#_Toc168055575)

[1.3.2 Дополнительные требования 12](#_Toc168055576)

[1.4 Требования к функциям, выполняемым системой 12](#_Toc168055577)

[1.5 Требования к видам обеспечения 12](#_Toc168055578)

[1.5.1 Требования к информационному обеспечению 12](#_Toc168055579)

[1.5.2 Требования к алгоритмическому обеспечению 13](#_Toc168055580)

[1.5.3 Требования к программному обеспечению 13](#_Toc168055581)

[1.6 Анализ аналогичных разработок 14](#_Toc168055582)

[1.6.1 Calibre 14](#_Toc168055583)

[1.6.2 Kavita 15](#_Toc168055584)

[1.6.3 Koodo 16](#_Toc168055585)

[2 Информационное обеспечение системы 18](#_Toc168055586)

[2.1 Выбор средств управления данными 18](#_Toc168055587)

[2.2 Проектирование базы данных 20](#_Toc168055588)

[2.3 Организация сбора, передачи, обработки и выдачи информации 24](#_Toc168055589)

[3 Алгоритмическое обеспечение системы 25](#_Toc168055590)

[3.1 Алгоритм обмена информации между приложением и API искусственного интеллекта 25](#_Toc168055591)

[3.2 Алгоритм добавления электронной книги в систему 25](#_Toc168055592)

[4 Программное обеспечение системы 26](#_Toc168055593)

[4.1 Структура программного обеспечения и функции его компонентов 26](#_Toc168055594)

[4.2 Выбор компонентов программного обеспечения 26](#_Toc168055595)

[4.2.1 Клиентская операционная система 26](#_Toc168055596)

[4.2.2 Инструментальное средство разработки и язык программирования 27](#_Toc168055597)

[4.2.3 Вспомогательное программное обеспечение 27](#_Toc168055598)

[4.3 Разработка прикладного программного обеспечения 28](#_Toc168055599)

[4.3.1 Структура прикладного программного обеспечения 28](#_Toc168055600)

[4.4 Особенности реализации, эксплуатации и сопровождения системы 30](#_Toc168055601)

[4.5 Руководство пользователя 31](#_Toc168055602)

[4.5.1 Требования к условиям эксплуатации 31](#_Toc168055603)

[4.5.2 Инсталляция и настройка 31](#_Toc168055604)

[4.5.3 Порядок и особенности работы 33](#_Toc168055605)

[5 ТЕСТИРОВАНИЕ СИСТЕМЫ 35](#_Toc168055606)

[5.1 Условия и порядок тестирования 37](#_Toc168055607)

[5.2 Исходные данные для контрольных примеров 37](#_Toc168055608)

[5.3 Результаты тестирования 37](#_Toc168055609)

[6 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 38](#_Toc168055610)

[6.1 Расчет показателя трудоёмкости для программного продукта 38](#_Toc168055611)

[6.2 Расчет затрат на материальные ресурсы 39](#_Toc168055612)

[6.3 Расчёт затрат на разработку системы 40](#_Toc168055613)

[6.4 Расчёт затрат на оплату труда 40](#_Toc168055614)

[6.5 Расчет отчислений на социальные нужды 41](#_Toc168055615)

[6.5 Себестоимость проекта 41](#_Toc168055616)

[6.6 Расчёт плановой прибыли 41](#_Toc168055617)

[6.7 Определение экономической эффективности проекта 43](#_Toc168055618)

[6.8 Выводы по технико-экономическому анализу и обоснованию проекта разработки 43](#_Toc168055619)

[ЗАКЛЮЧЕНИЕ 44](#_Toc168055620)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc168055621)

[ПРИЛОЖЕНИЕ А 46](#_Toc168055622)

# CПИСОК ИСПОЛЬЗОВАННЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ

ИИ – искусственный интеллект.

СУБД – система управления базами данных.

ПО – программное обеспечение.

Фреймворк (Software Framework) – программная платформа, определяющая структуры программного обеспечения.

API – интерфейс программирования приложения (Application Programming Interface).

.NET Core – модульная платформа для разработки программного обеспечения с открытым исходным кодом.

С# – объектно-ориентированный язык программирования общего назначения, разработанный компанией Microsoft.

DTO (Data Transfer Object) – объект переноса данных, нужная для передачи информации.

UML – аббревиатура от Unified Modeling Language, язык графический описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес процессов, системного проектирования и отображения организационных структур.

# ВВЕДЕНИЕ

В современном мире набирает всю большую популярность искусственный интеллект (ИИ). Его практически можно встретить во многих отраслях деятельности, таких как финансы, промышленность, торговле, образование, системы безопасности, робототехника и т.п. В быту можно встретить в обработке фотоизображений, распознаваний образов на фото, улучшение и восстановление качества фотоизображений, и транскрипция или перевод с одного естественного языка на другой. Всё благодаря развитию мощностей компьютерной техники, увеличение количества транзисторов на кристаллах интегральных схем, а также появлении искусственных нейронных сетей и генеративных ИИ. Также популярности ИИ способствовало появление чат-бота на базе больших языковых моделей LLM ChatGPT от компании OpenAI.

На данном этапе развития ИИ может использоваться в качестве инструмента автоматизации некоторых процессов. В некоторых сферах деятельности она может полностью заменить человека, в основном это сфера предоставления услуг конечным потребителям, такие как кассиры, банковские служащие, официанты и т.п. Также ИИ может использоваться в качестве автопилота транспортных средств, и самый известный пример — это электромобили от американской компаний Tesla. В военном деле ИИ используется для управления дронами, роботами-сапёрами и автономными комплексами вооружения.

В сфере разработки программного обеспечения искусственный интеллект, в основном на базе нейронных сетей, помогают в написании кода и быстрого поиска нужной информации. Такие ИИ-ассистенты не могут полностью заменить программиста человека, так как код всё также нужно анализировать на корректность работы логики в предметной области и на ошибки, особенно связанные с кибербезопасностью. В основном публичные ИИ-инструменты, такие как GitHub Copilot, обучающиеся на коде опубликованные в открытом доступе, у которых в большинстве случаев не было проведено ревью кода (code review), может генерировать устаревший код, так как код используемых библиотек может измениться со временем и/или генерировать код с неявными уязвимостями, связанные с безопасностью.

Из основных недостатков искусственного интеллекта являются

* Безопасность и конфиденциальность данных: использование ИИ может создать уязвимости в системах безопасности и привести к утечкам конфиденциальной информации.
* Потеря рабочих мест: автоматизация процессов с использованием ИИ может привести к сокращению рабочих мест, особенно в тех сферах, где задачи могут быть легко автоматизированы.
* Предвзятость и дискриминация: ИИ может унаследовать предвзятость из данных, на которых он обучается, что может привести к несправедливым или дискриминационным решениям.
* Ответственность и прозрачность: в некоторых случаях сложно объяснить, почему ИИ принял определенное решение, что затрудняет определение ответственности в случае ошибок или несправедливых действий.
* Этические вопросы: использование ИИ вызывает много этических вопросов, включая проблемы конфиденциальности, принципа справедливости и вопросы о том, кто несет ответственность за действия ИИ.
* Угроза безопасности: развитие ИИ также может создать новые угрозы для безопасности, такие как создание автономных систем, способных принимать вредные решения.

Цель данной дипломной работы является получение навыков разработки прикладного обеспечения, навыков работы с современными инструментами разработки, интеграции искусственного интеллекта в информационную систему.

# 1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ СИСТЕМЫ

## 1.1 Назначение и цели создания системы

Назначением системы:

Система предназначена для хранения электронных книг, в основном технической литературы, а также других документов в едином хранилище с возможностью их просмотра, редактирования описания книг, отслеживания активности пользователя при работе с документами.

Местом внедрения является внутренняя локальная сеть пользователя.

Цель создания системы:

* уменьшение времени доступа к работе с электронными книгами и документами;
* снижение нагрузки на дисковые системы устройств пользователя.

## 1.2 Характеристика объекта автоматизации

Объектом автоматизации является неавтоматизированное рабочее место читателя в локальной сети.

### 1.2.1 Общее описание

Объектом автоматизации нынешней системы является процесс работы с файлами книг и документов. Доступ к файлам можно получить только внутри локальной сети на сетевом хранилище по протоколу SMB, организованный с помощью программного обеспечения роутера.

### 1.2.2 Состав и порядок функционирования

Функционирование системы включает в себя набор следующих процессов:

* реализация общего доступа к хранилищу с книгами;
* отслеживание прогресса чтения для каждой книги для каждого пользователя;
* перевод выделенной части текста книги;
* обобщение выделенной части текста документа.

Алгоритм работы с файлами электронных книг следующий:

* открыть сетевой общий диск через любой файловый менеджер поддерживающий протокол обмена данными SMB;
* найти определенный файл книги;
* открыть файл книги в любом удобном для пользователя программы для просмотра документа;
* читать, находя неизвестные слова записываем их и по необходимости делается перевод через любые сторонние (веб-)сервисы.

## 1.3 Общие требования к системе

### 1.3.1 Требования к структуре и функционированию системы

Система должна представлять собой программный комплекс, в состав которого должны входить следующие подсистемы:

* подсистема сервера;
* подсистема клиента.

Подсистема серверной части должна состоять из следующих компонентов:

* учёт списка электронных книг;
* учёт списка пользователей;
* учёт (трекинг) взаимодействия пользователя с книгами.

Пользовательский интерфейс подсистемы клиентской части программного обеспечения должен состоять из:

* страница аутентификации (ввода логина и пароля);
* страница списка электронных книг;
* страница списка коллекции электронных книг;
* страница просмотра электронной книги.

Требования к режимам функционирования системы предполагают:

* обеспечения аутентификации и авторизации;
* обеспечение корректной работы на нескольких устройствах;
* обеспечение корректной работы отслеживания активности пользователя;
* обеспечение хранения файлов электронных книг.

Перспективы развития системы предполагают:

* возможность поддержки просмотра электронных книг других форматов по типу DJVU, EPUB и т.д.;
* возможность преобразование файла электронной книги в другой файловый формат;
* возможность нахождения данных о книге;
* возможность отображения советов и рекомендации;
* улучшение удобства чтения.

### 1.3.2 Дополнительные требования

Техническое обслуживание и администрирование программного обеспечения должно выполняться специалистами, имеющими соответствующую квалификацию и навыки выполнения работ.

Система является свободно распространяемой и не требует лицензии.

## 1.4 Требования к функциям, выполняемым системой

В системе должны быть реализованы следующие функции:

* функция предоставления списка документов;
* функция просмотра документа;
* функция перевода текста;
* функция нахождения определения выбранного слова;

Первоочередность реализации функций указана по порядку.

## 1.5 Требования к видам обеспечения

### 1.5.1 Требования к информационному обеспечению

В системе должны быть предусмотрены следующие требования:

* наличие системы управления базами данных PostgreSQL;
* реализация средств контроля вводимых данных;
* требования к архивации и шифровании отсутствуют.

### 1.5.2 Требования к алгоритмическому обеспечению

Необходимо реализовать алгоритм обмена информацией между серверной частью и API искусственного интеллекта.

Необходимо реализовать алгоритм добавления электронной книги в электронную библиотеку.

### 1.5.3 Требования к программному обеспечению

Требования к организации интерфейса:

* пользовательский интерфейс должен быть удобен, интуитивно понятен и прост в использовании;
* на главной странице расположить список недавно открытых электронных книг, кнопка добавления новых книг из системы пользователя, сортировка книг по разным критериям, поиск по разным критериям;
* на странице аутентификации пользователя расположить ввод пользователя и пароля в виде ПИН-кода, кнопка входа в систему, кнопка регистрации нового пользователя;
* на странице просмотра книги должно быть кнопки навигации по разным страницам, управляющие элементы изменения масштаба, возможность выделять текст, копирование в буфер обмена выделенного текста, также возможность нахождения перевода выделенного текста, нахождение определения выделенного текста;
* страница редактирования метаданных книги таких как название, теги, описание.

## 1.6 Анализ аналогичных разработок

Существует некоторое количество систем, выполняющих схожие разрабатываемой задачи. Их основной функционал направлен на предоставление электронной библиотеки и возможности их чтения.

### 1.6.1 Calibre

Calibre – кроссплатформенное, свободное и отрытое программное обеспечение для чтения и хранения электронных книг в электронной библиотеке для настольных систем на базе Windows, MacOS и Linux, разработанное в 2006 году. В данный момент продолжает поддерживаться. Распространяется под лицензией GNU GPL версии 3. Последняя версия программы – 7.9.0 (19 апреля 2024). Обладает довольно большим функционалом: можно конвертировать книги в разные форматы, собирать статьи с новостных сайтов и создавать из них электронную книгу, возможность искать электронные книги на разных сайтах магазинах и т.п. Большая часть кода программы написана на языке программирования Python и с помощью фреймворка PyQt. Визуальное представление программы Calibre представлена на рисунке 1.

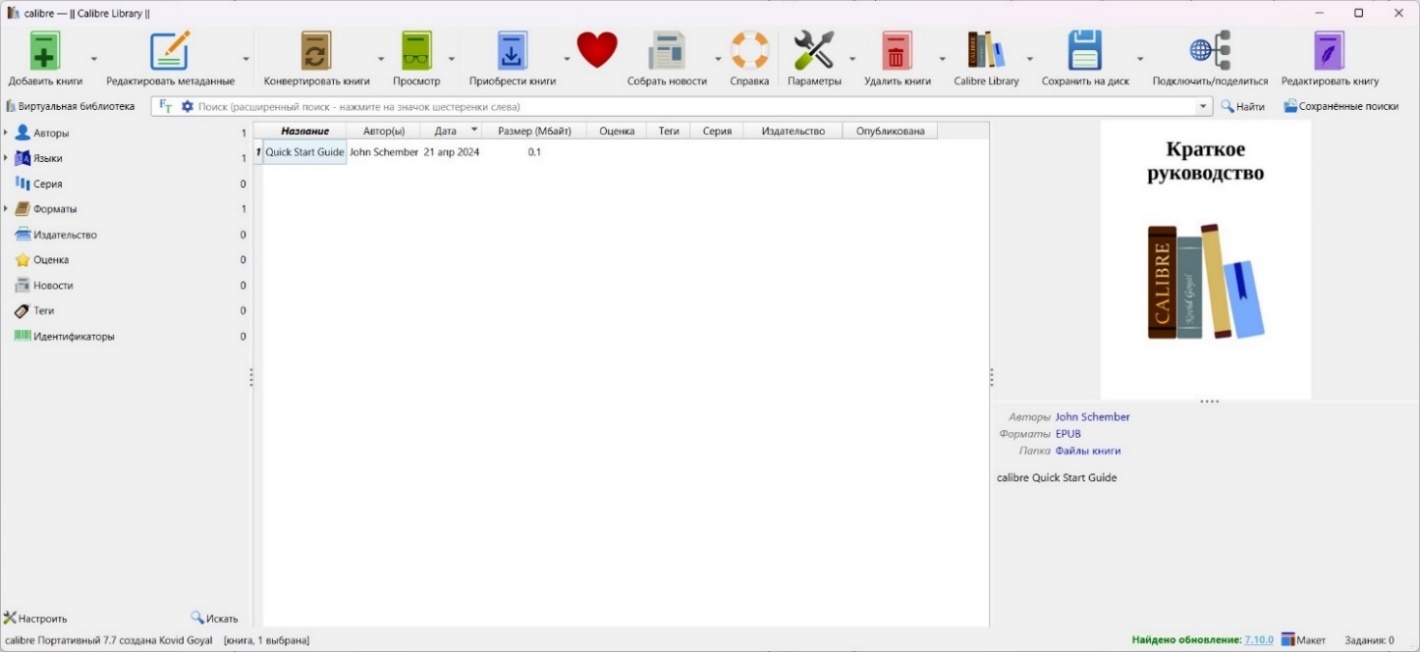


Рисунок 1 – Окно программы Calibre

### 1.6.2 Kavita

Kavita – открытое программное обеспечение для чтения и хранения книг в электронной библиотеке, созданное в 2020 году. Сделана в виде web-приложения. Написана на языках программирования C# и Typescript с помощью фреймворков ASP.NET и Angular. На данный момент ещё находится в разработке. Последняя версия – 0.8.1 (23 апреля 2024). Исходный код программы распространяется под лицензией GNU GPL версии 3. Визуальное представление программы Kavita изображена на рисунке 2.

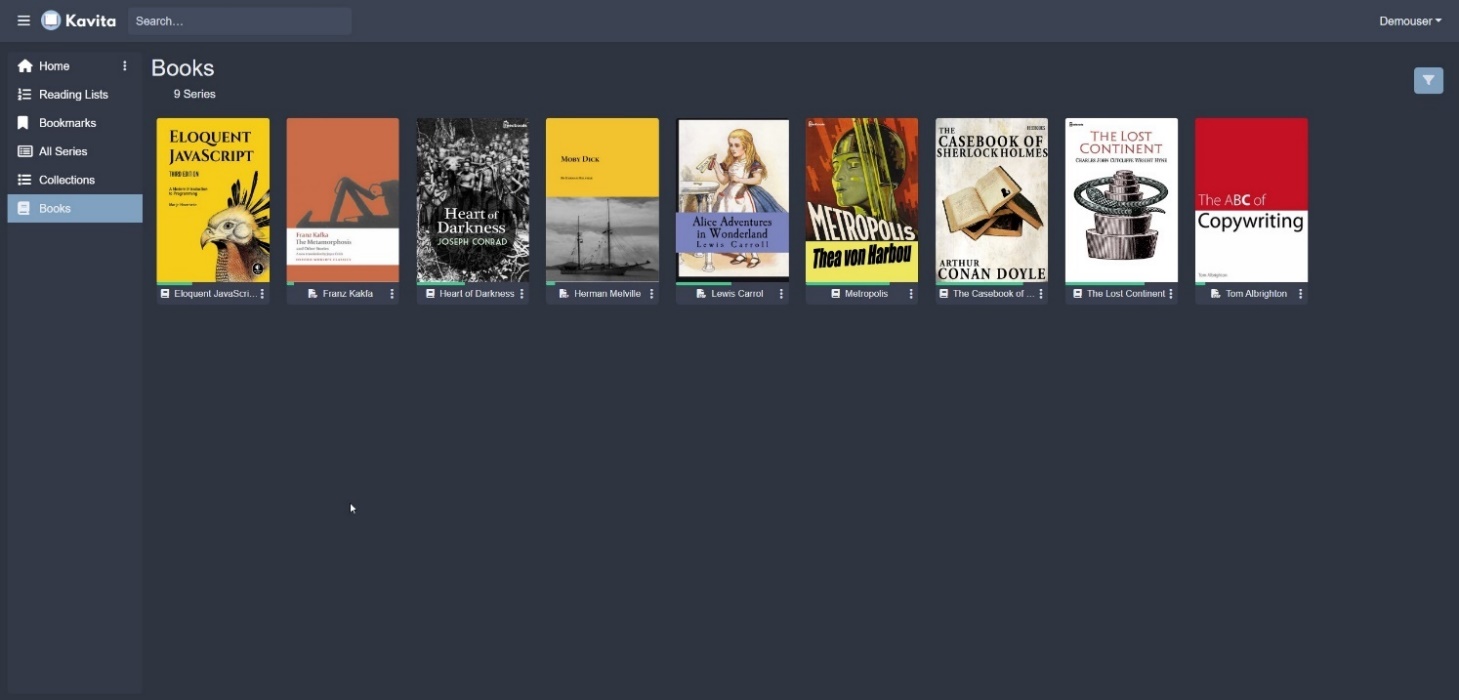


Рисунок 2 – Веб-страница программы Kavita

### 1.6.3 Koodo

Koodo – свободное программное обеспечение для чтения и хранения книг, разработанная в 2020 году. Написана на языке программирования JavaScript с использованием фреймворка React. Распространяется в виде настольного приложения на базе Electron и может работать как веб-приложение. Исходный код распространяется под лицензией AGPL версии 3. Последняя версия – 1.6.6 (6 апреля 2024). Визуальное представление программы Koodo изображена на рисунке 3.

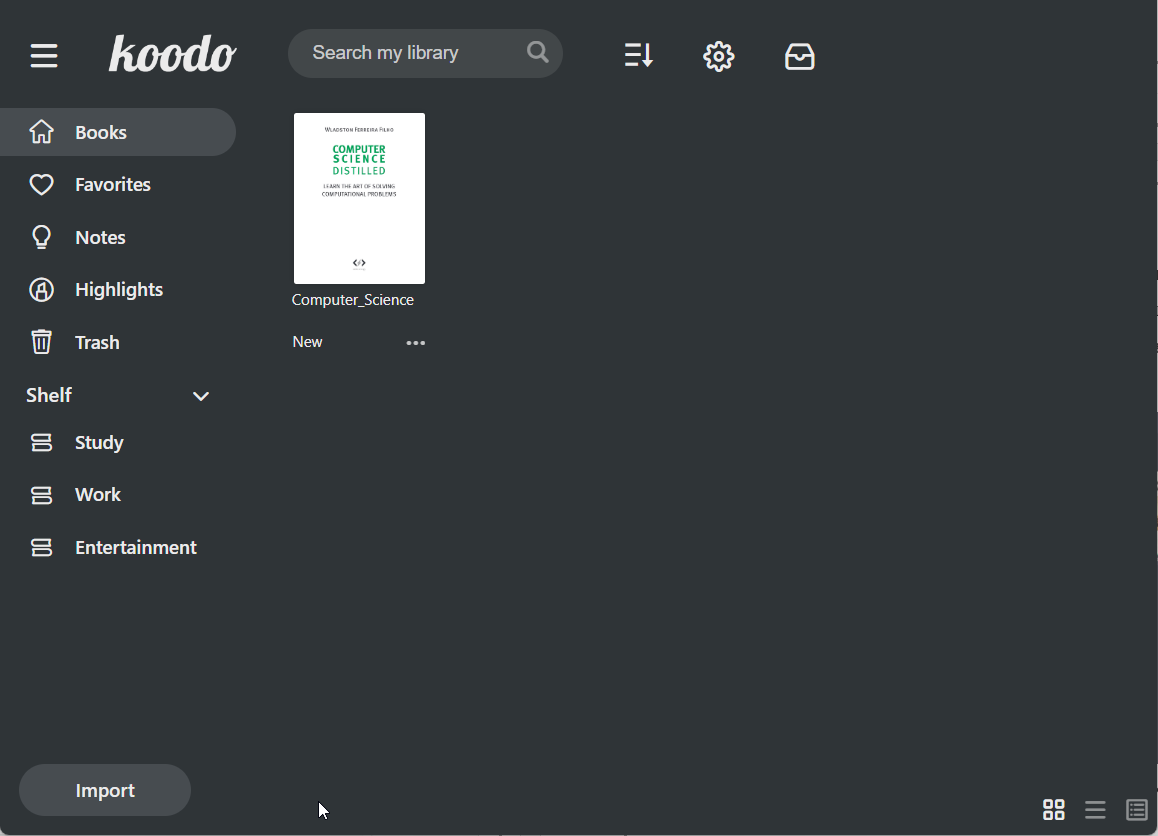


Рисунок 3 – Страница программы Kodoo

Сравнительный анализ аналогичных программных систем с разрабатываемым продуктом приведена в таблице 1.1.

Таблица №1.1 - Сравнительный анализ аналогичных программных продуктов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название программного продукта** | **Calibre** | **Kavita** | **Koodo** | **Разработанный программный продукт** |
| Многопользовательская система | - | + | - | + |
| Веб-интерфейс | - | + | + | + |
| Настольное приложение | + | - | + | +  (в виде PWA) |
| Трекинг проведенного времени | - | - | - | + |
| Краткий пересказ выделенного текста | - | - | - | + |
| Встроенный словарь | + | - | - | + |
| Лицензия исходного кода программы | GPL 3.0 | GPL 3.0 | AGPL 3.0 | MIT |

# 2 Информационное обеспечение системы

## 2.1 Выбор средств управления данными

На данный момент существуют множество систем управлений баз данных (СУБД). Одними из популярных СУБД на 2023 год считаются исходя опроса 76634 всех участников на сайте StackOverflow:

* PostgreSQL (45,55%).
* MySQL (41,09%).
* SQLite (30,9%).
* MongoDB (25,52%).
* Microsoft SQL Server (25,45%).

Проценты обозначают количество людей желающих пользоваться данными СУБД и тех людей, которые пользовались ими в течении последнего года.

Множество из этих решений являются реляционными СУБД и поддерживают язык запросов SQL для записи и чтения данных. Существуют также и NoSQL системы, такие как MongoDB, в котором данные хранятся в виде документов формата BSON (бинарный JSON). Набор полей в документах не имеют четкой структуры и одно и тоже поле может иметь значения разных типов. Он хорошо подходит для хранения плохо структурированной информации, а также для быстрого прототипирования программных систем. Запросы осуществляются через драйвер, которая по сути является библиотекой. Эта СУБД официально поддерживает все основные популярные языки программирования, а также существуют драйверы, написанные и поддерживаемые сообществом открытого программного обеспечения. Имеет лицензию Server Side Public License.

Из NoSQL БД можно считать СУБД Redis, которая позволяет довольно быстро записать и считывать данные. Она хранит всю информацию в оперативной памяти (in-memory database). В основном используется как брокер сообщений между сервисами системы. В данной работе она не подходит, так как у нас только один сервер.

Также есть вариант использования файловой СУБД SQLite, но так как в системе может работать несколько пользователей одновременно, этот вариант не является оптимальным в данной задаче. Из плюсов установка самой СУБД не требуется, так как он обычно поставляется вместе программным продуктом в виде сторонней библиотеки. В основном она используется в системах на стороне клиента.

Одной из самой популярной реляционной СУБД является MySQL, разработанная в 1995 году компанией Sun Microsystems. На данный момент компания Sun Microsystems поглощена Oracle Corporation в 2010=ых. Имеется open-source, называемой MySQL Community Edition, и коммерческие платные версии с технической поддержкой. Также существуют форки (производные проекты), такие как MariaDB и Percona. Широко применяется в малом бизнесе. Поддерживает все операционные системы.

Другой популярной СУБД является Microsoft SQL Server. Является проприетарной, то есть исходный код СУБД закрыт. Ранее она была имела только платные выпуски (edition), но недавно появились ещё и бесплатные выпуски, такие как Express и Developer. Эти варианты обладают не всеми возможностями представленные в платных выпусках. Developer версия в основном предназначена только для разработки и тестирования программных систем. Выпуск Express исходя из раздела про выпуски в документации к SQL Server (https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022?view=sql-server-ver16) предназначена для программных систем малых размеров и имеет некоторые ограничения.

В данной работе была выбрана реляционная СУБД PostgreSQL, так как она является наиболее популярным среди разработчиков программного обеспечения, а следовательно, существуют множество уже решенных проблем и других ресурсов, также обладает довольно хорошей документацией и относительно быстрее MongoDB, исходя отчёта по проведению теста производительности между PostgreSQL и MongoDB, компанией EnterpriseDB (Ссылка: https://info.enterprisedb.com/rs/069-ALB-339/images/PostgreSQL\_MongoDB\_Benchmark-WhitepaperFinal.pdf). Если сравнивать с MySQL PostgreSQL имеет больше типов данных для колонок такие как массив, JSON, перечисления (enum) и т.п. Последняя версия PostgreSQL – 16. Распространяется под лицензией PostgreSQL License, которая является open-source лицензией, схожая с BSD и MIT лицензией.

## 2.2 Проектирование базы данных

Диаграмма базы данных представлена на рисунке 2.1.

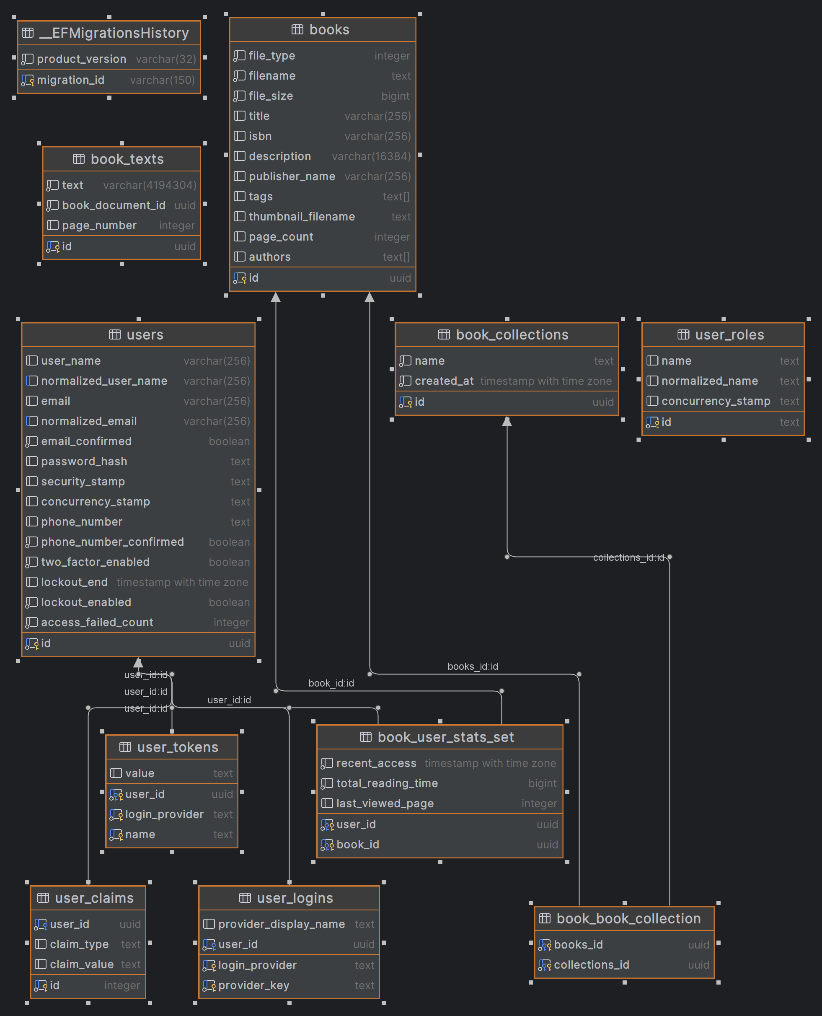


Рисунок 2.1 – Схема базы данных

Основной сущностью в данной предметной области является «Книга» (Book). Атрибуты сущности отображены в таблице 2.1.

Таблица 2.1 – Атрибуты сущности «Book»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | Id (ИД) | Guid | Идентификатор (ИД) |
| 2 | Title (Название книги) | string? | Название книги |
| 3 | Isbn | string? | Уникальный ИД книги в формате EAN-11 или EAN-13 |
| 4 | PublisherName | string? | Название компании издателя |
| 5 | Description | string? | Описания |
| 6 | Filename | string | Имя файла документа, обязателен |
| 7 | FileType | BookFileType | Тип файла документа, обязателен |
| 8 | FileSize | long | Размер файла, обязателен |
| 9 | Authors | string[]? | Список авторов |
| 10 | Tags | string[]? | Теги книги |
| 11 | PageCount | int? | Количество страниц |
| 12 | BookCollections | ICollection<BookCollection> | Список коллекции в которых эта книга входит |

Атрибуты сущности «Коллекции книг» (BookCollection) отображены в таблице 2.2.

Таблица 2.2 – Атрибуты сущности «BookCollection»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | Id (ИД) | Guid | Идентификатор (ИД) |
| 2 | Name (Название коллекции) | string | Название коллекции |
| 3 | Books | ICollection<Book> | Список книг в коллекции |
| 4 | CreatedAt (Когда было создано) | Instant | Время создание записи |

Сущность «Текст книги» (BookText) нужна для поиска по текстам книг (full-text search). Атрибуты сущности приведены в таблице 2.3.

Таблица 2.3 – Атрибуты сущности «BookText»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | Id (ИД) | Guid | Идентификатор (ИД) |
| 2 | Text (Текст) | string | Часть текста книги |
| 3 | PageNumber (Номер страницы) | int? | Номер страницы книги на котором расположена часть текста книги |
| 4 | BookDocumentId | Guid | Внешний ключ к сущности «Книга» |

Сущность «Статистика пользователя» (BookUserStats) показывает количество времени, который провёл пользователь с определенной книгой, на какой странице он остановился и когда последний раз её открывал. Все её атрибуты представлены в таблице 2.4.

Таблица 2.4 – Атрибуты сущности «BookUserStats»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | BookId (ИД книги) | Guid | Идентификатор книги |
| 2 | UserId (ИД пользователя) | Guid | Идентификатор пользователя |
| 3 | RecentAccess (Последнее время доступа) | Instant | Последнее время, когда пользователь открывал книгу |
| 4 | TotalReadingTime | long | Общее проведенное время с книгой в секундах |
| 5 | LastViewedPage | int? | Номер страницы, на котором остановился пользователь |

Атрибуты сущности «Словарного слова» (DictionaryWord) отбражена в таблице 2.5.

Таблица 2.5 – Атрибуты сущности «DictionaryWord»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | Word | string | Само слово; используется в качестве основного ключа таблицы |
| 2 | Transcription | string? | Транскрипция слова, необязателен |
| 3 | LanguageCode | string? | Кода языка, необязателен |
| 4 | Aliases | ICollection<DictionaryWordAlias> | Список слов в альтернативной форме |
| 5 | Definition | ICollection<DictionaryWordDefinition> | Список определении этого слова |

Сущность «Альтернативная форма слова» (DictionaryWordAlias) имеет зависимость многие к одному к сущности «Словарное слово» и само по себе не может существовать. Атрибуты сущности представлена на рисунке 2.6.

Таблица 2.6 – Атрибуты сущности «DictionaryWordAlias»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | Alias | string | Альтернативное форма слова; используется в качестве основного ключа таблицы |

Также как с сущностью «Альтернативная форма слова» сущность «Определение слова» (DictionaryWordDefinition) имеет зависимость многие к одному от сущности «Словарное слово» и само по себе не может существовать. Атрибуты сущности представлена на рисунке 2.7.

Таблица 2.7 – Атрибуты сущности «DictionaryWordDefinition»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | Id | Guid | Уникальный идентификатор, основный ключ |
| 2 | PartOfSpeech | string? | Часть речи, необязательно |
| 3 | SubjectName | string? | Имя деятельности, где используется определение; необязательно |
| 4 | Definition | string | Само определение слова |

Атрибуты сущности «Пользователь» (User) отображены в таблице 2.8.

Таблица 2.8 – Атрибуты сущности «Пользователь»

| **№** | **Название атрибута** | **Тип** | **Описание** |
| --- | --- | --- | --- |
| 1 | Id (ИД) | Guid | Идентификатор пользователя |
| 2 | UserName | string | Имя пользователя |
| 3 | NormalizedUserName | string | Имя пользователя записанная заглавными буквами |
| 3 | PasswordHash | string | Хеш пин-кода пользователя |

## 2.3 Организация сбора, передачи, обработки и выдачи информации

Сбор информации осуществляется путём ввода данных пользователем через веб-интерфейс системы. Выдача списка книг и их просмотр также производится через веб-интерфейс. Обеспечение достоверности информации и регламентные процедуры обслуживания не требуется.

# 3 Алгоритмическое обеспечение системы

## 3.1 Алгоритм обмена информации между приложением и API искусственного интеллекта

Общая характеристика: алгоритм предназначен для обмена информация между разрабатываемой системой и облачной службой Яндекса с заранее обученной моделью нейронной сети, который обобщает переданный текст. Вызывается конечным пользователем системы через веб-интерфейс на странице просмотра книги. Протокол обмена данными между двумя системами – HTTPS.

Результат выполнения: список строк обобщения, переданного на вход текста.

Логическое описание: схема алгоритма изображена на рисунке 3.1.

## 3.2 Алгоритм добавления электронной книги в библиотеку

Общая характеристика: алгоритм предназначен для добавления записи об электронной книге СУБД, а также сохранения файла на дисковую систему, и дальнейшей индексации всего текста книги. Выполняется, когда пользователь добавляет свою копию электронной книги в систему через веб-интерфейс на странице списка книг через диалоговое окно редактирования книги.

Результат выполнения: объект класса «BookDto», добавленная в базу данных.

Логическое описание: схема алгоритма изображена на рисунке 3.2.

# 4 Программное обеспечение системы

## 4.1 Структура программного обеспечения и функции его компонентов

Система создаётся для использования на настольных системах пользователей, а также возможна работа на мобильных устройствах.

## 4.2 Выбор компонентов программного обеспечения

В качестве основной встраиваемой среды разработки (IDE) используется JetBrains Rider.

Для работы серверной части требуется среда исполнения .NET версии 8 и выше. Далее перечислены сторонние пакеты серверной части:

* веб-фреймворк ASP.NET Core;
* пакет для работы с СУБД Entity Framework Core;
* пакет-драйвер для работы СУБД PostgreSQL Npgsql;
* пакет для работы с облачным сервисом Яндекс Yandex.Cloud;
* пакет для работы с PDF файлами PdfPig.

Для работы клиентской части нужен веб-сервер Nginx или Apache. В основном клиентская часть тестировалась на Nginx. Также как и с серверной частью использовались следующие сторонние пакеты:

* веб-фреймворк Angular 17;
* пакет UI-компонентов Angular Material 17;
* пакет для работы со временем/датами date-fns;
* пакет для просмотра PDF документов ngx-extended-pdf-viewer в Angular (сам этот пакет является «оберткой» над другим пакетом pdf.js).

### 4.2.1 Клиентская операционная система

Так как разрабатываемое решение представляет собой веб-приложение, то клиентская операционная система может быть любой, которая поддерживает любой современный браузер.

### 4.2.2 Инструментальное средство разработки и язык программирования

Исходя из требований к разрабатываемой системе, которые были зафиксированы в техническом задании в качестве языка программирования были выбраны С# и TypeScript.

C# - это кросс-платформенный язык программирования (ЯП) общего назначения, который работает в бесплатной, кросс-платформенной среде .NET, разработанная компанией Microsoft [7]. Синтаксис языка схож с языком программирования Java. Написанный код на C# компилируется в промежуточный язык IL (Intermediate Language), который затем запускается в runtime-среде .NET. Среда может на лету компилировать часть IL код в нативный машинный код (JIT-компиляция). С версии .NET 8 появилась возможность статической (AOT, Ahead-of-time) компиляции, которая значительно снижает время запуска приложения.

TypeScript – это типизированный высоко-уровневый интерпретируемый язык программирования общего назначения, также разработанная компанией Microsoft, и являющийся надмножеством ЯП JavaScript. Код TypeScript в конечном итоге компилируется в код на JavaScript. Этот код затем выполняется в движке JavaScript, который обычно встроен во всех современных веб-браузерах, а также может представлена в виде отдельной программы как NodeJS или Deno.

В качестве интегрированной среды разработки был выбран JetBrains Rider, который имеет большое количество возможностей рефакторинга (преобразования в определенный вид) кода, предоставляет удобство разработки и отладки кода. В основном предназначен для разработки проектов на C#, но также представляет возможность писать веб-приложения с использованием ЯП JavaScript/TypeScript, а также языка разметки интерфейса HTML и стилей CSS/SASS/LESS. Также имеет возможность для работы почти со всеми СУБД. Имеет систему расширений, который дополняет функционал среды разработки.

### 4.2.3 Вспомогательное программное обеспечение

Для работы с данной системой требуется сервисы Яндекса и Merriam-Webster.

## 4.3 Разработка прикладного программного обеспечения

Весь код разрабатываемой системы, серверной и клиентской части, находится в одном репозитории.

### 4.3.1 Структура прикладного программного обеспечения серверной части

Клиентская часть находится в директории “web”, а код серверной части находится в директории “src”. Структура решения серверной части изображена на рисунке 4.1.

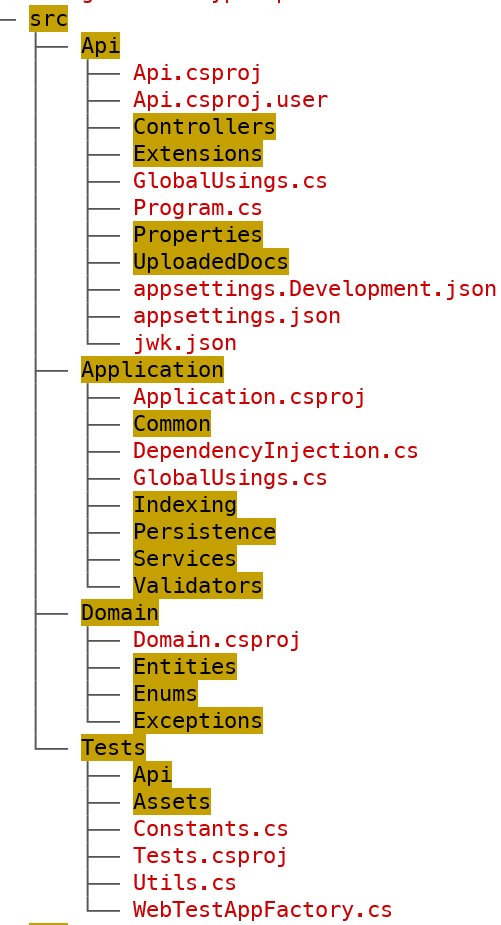


Рисунок 4.1 - Структура серверной части

Существуют 4 проекта C#, каждый из проектов содержит файл с расширением csproj, который представляет собой XML файл, описывающий настройки сборки проекта, пакеты от третьих лиц и другие внешние зависимости.

Проект «Domain» («Домен») содержит классы (в директории «Entities») и перечисления (в директории «Enums») относящийся к предметной области разрабатываемой системы. Не имеет зависимостей от проектов внутри системы.

Проект «Application» («Приложение») содержит бизнес-логику и обеспечивает работу со слоем хранения данных (Persistence layer). Имеет внешнюю зависимость от «Domain» проекта. В директории «Common» содержатся DTO (Data Transfer Object) классы, интерфейсы сервисных классов, статический класс констант, классы настроек конфигурации. В директории «Persistence» хранятся классы и интерфейсы, относящийся к работе с хранилищем данных и СУБД. Директория «Validators» содержит классы валидаторы, которые проверяют корректность введенных пользователем полей объектов определенных классов. В директории «Indexing» содержит классы относящийся к индексированию текста книг и их сохранения в базу данных.

Проект «Api» представляет собой ASP.NET Web API приложение, по сути, представляющее собой HTTP сервер, который производит обмен информацией в формате JSON между клиентами. Представляет собой слой представления. Имеет зависимость от проекта «Application».

Проект «Testing» содержит код теста серверной части приложения и имеет зависимость от проекта «Api».

Такая структура проекта позволяет легко масштабировать, то есть позволяет легко добавлять или изменять определенный функционал, и также тестировать отдельные части программы.

### 4.3.2 Структура прикладного программного обеспечения клиентской части

Далее на рисунке 4.2 изображена структура директории клиентской части TypeScript проекта.



Рисунок 4.2 – Структура клиентской части проекта

## 4.4 Особенности реализации, эксплуатации и сопровождения системы

Сложность реализации системы можно оценить, как высокую.

Разработчику системы необходимо иметь базовые знания языка программирования C# и TypeScript, и как работать со сторонними библиотеками такими как ASP.NET, Entity Framework и Angular. Также разработчик должен уметь работать с языком разметки HTML для описания интерфейса и языком описания стилей CSS.

Также от разработчика требуется умение работы с программной платформой Docker, которая предназначена для выполнения приложении в рабочей среде (production).

## 4.5 Руководство пользователя

### 4.5.1 Требования к условиям эксплуатации

От пользователя системы требуется базовый опыт работы с любым современным веб-браузером.

### 4.5.2 Инсталляция и настройка

Для серверной части приложения потребуется платформа для запуска программ в виде «контейнеров» Docker и СУБД PostgreSQL 16, а также API-ключи для доступа к веб-сервису словаря Merriam Webster и к облачной платформе от Яндекса.

Docker это открытая платформа для разработки, передачи, и запуска приложений. Он позволяет отделить приложение от инфраструктуры для того, чтобы передать программную систему конечному пользователю. Под инфраструктурой подразумевается вспомогательные программное обеспечение или библиотеки, которое требуется для работы программы, будь то среда выполнения .NET или NodeJS, библиотека OpenSSL и т.п. Приложение в Docker распространяется в виде «образов» (images). Образ Docker это стандартизированный пакет, в котором находятся файлы программного обеспечения и его вспомогательные библиотеки, необходимые для корректной работы приложения. Собранный образ является неизменяемым и состоит из слоёв (layer). Образ может быть построен на базе другого образа, который может содержать необходимые для определенного проекта пакеты, такие как Nginx. Все эти пакеты могут находится локально, либо в общем официальном реестре образов Docker Hub. [5]

Docker берёт образ и запускает их в виде «контейнеров» (containers), который по сути является изолированным процессом в операционной системе. Docker эмулирует операционную систему для контейнеров, используя ядро Linux, которое работает в основной операционной системе (host OS). И в этом заключается отличие от виртуальных машин, где эмулируется целая компьютерная система.

Если Docker нужно запустить в ОС Windows, то для этого потребуется подсистема Windows для Linux (WSL), который запускает среду Linux (Linux environment) на машине с Windows, без необходимости в отдельной виртуальной машине или dual booting (компьютерная система на котором установлены Windows и Linux на отдельных физических/логических разделах дисков). [7]

Для создания образа серверной части приложения потребуется файл Dockerfile, который содержит последовательность инструкции для сборки образа Docker. Dockerfile серверной части содержит текст на рисунке 4.3.

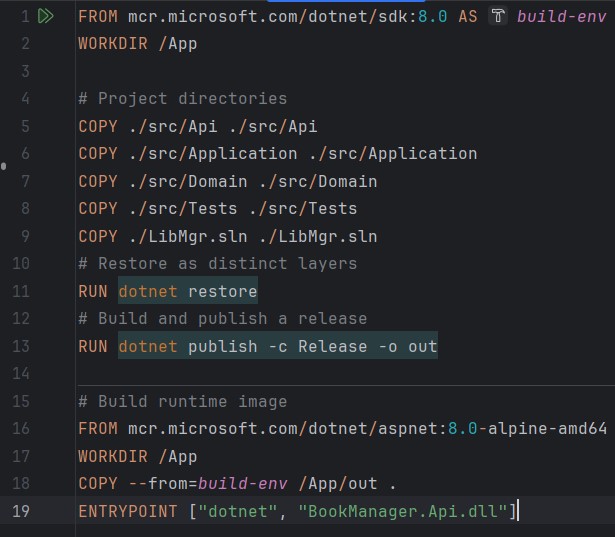


Рисунок 4.3 – Текст Dockerfile

Затем нужно запустить следующую команду находясь в директории с файлом Dockerfile:

*docker build -t libmgr-server .*

Аргумент «-t» отвечает за задание имени образа, не является обязательным. Под символом «.» точка подразумевается рабочая директория в котором находится пользователь в командном интерпретаторе. Если бы Dockerfile находился в другой директории, то нужно было бы задать путь до директории, где этот файл расположен. Также можно запустить скрипт PowerShell «build-docker-image.ps1» в директории «scripts» в корневой директории проекта.

При запуске образа приложения необходимо указать ему переменные окружения (environment variables), в котором будет содержать переменные с паролем пользователя «an» для доступа к СУБД.

### 4.5.3 Порядок и особенности работы

При переходе на веб-страницу приложения, система перенаправит на страницу аутентификации (рисунок 4.4).

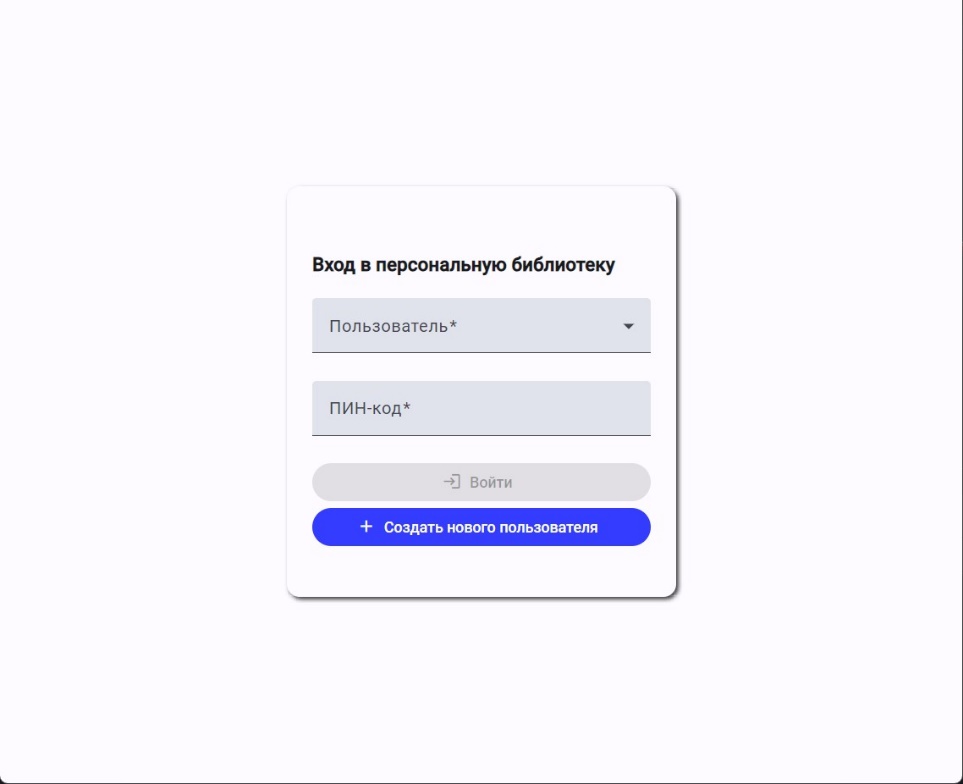


Рисунок 4.4 – Страница аутентификации

Далее нужно создать аккаунт пользователя. Для этого необходимо нажать на кнопку «Создать нового пользователя». Страница откроет диалоговое окно, на котором нужно ввести имя пользователя и ПИН-код. ПИН-код пользователя должен иметь длину в интервале от 4 до 12 цифр. Диалоговое окно создания аккаунта пользователя изображена на рисунке 4.5.

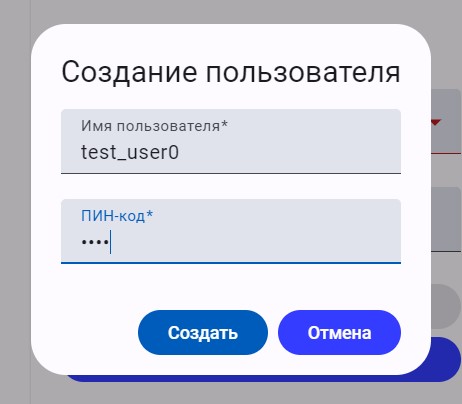


Рисунок 4.5 – Диалоговое окно создания аккаунта пользователя

После ввода данных в поля, нужно нажать на кнопку «Создать». После этого окно должно закрыться и вывести сообщение об успешности создания в нижней части приложения, как на рисунке 4.6.

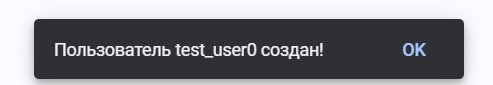


Рисунок 4.6 – Вывод сообщения об успешности создания пользователя

Подтверждение аккаунта пользователя не требуется. Дальше пользователь должен выбрать собственное имя пользователя и ввести ПИН-код (рисунок 4.7).

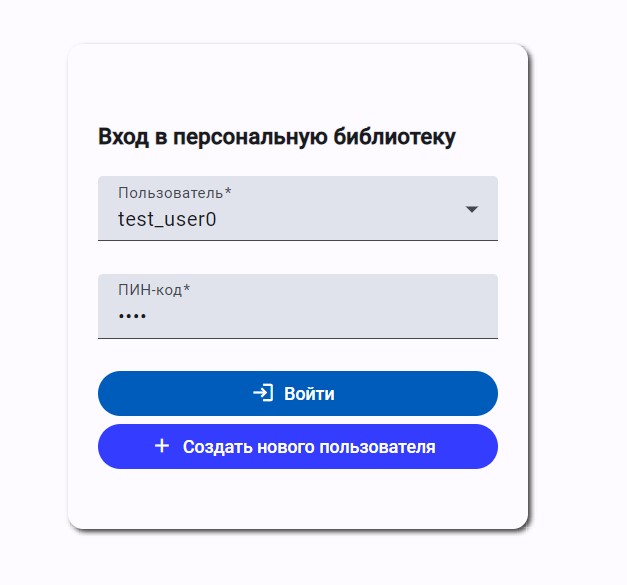


Рисунок 4.7 – Ввод имени пользователя и ПИН-кода

После этого кнопка «Войти» станет активным и далее нажимаем на него. После этого откроется страница с недавними открытыми книгами (рисунок 4.8).



Рисунок 4.8 – Страница недавно открытых книг (пустой)

После входа в систему, в левой части приложения появится панель навигации. Её можно раскрыть, нажав на кнопку с иконкой в виде трёх полос (рисунок 4.9).

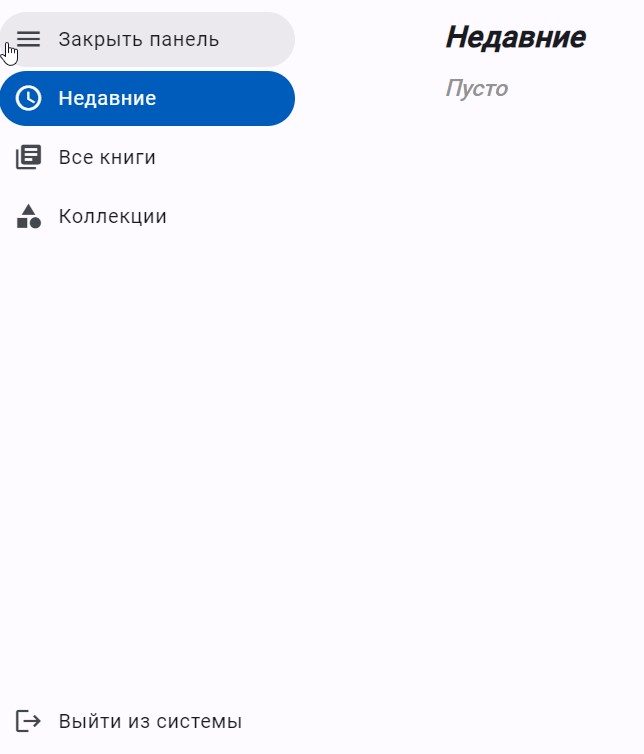


Рисунок 4.9 – Раскрытая панель навигации

Также при наведении на кнопки навигации в «скрытом состоянии» будет появляется блок с надписью назначения кнопки (рисунок 4.10).

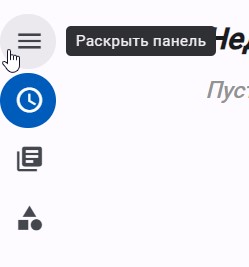


Рисунок 4.10 – Надпись рядом кнопкой навигации

Для добавления файла электронной книги нужно перейти на страницу со списком электронных книг нажав на кнопку с иконкой в виде «документа» (рисунок 4.11).

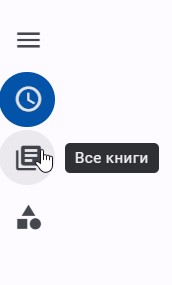


Рисунок 4.11 – Кнопка для перехода на страницу со списком электронных книг

Далее откроется страница со списком электронных книг, пока что пустая, изображенная на рисунке 4.12.



Рисунок 4.12 – Страница со списком книг (пустая)

Эта страница состоит из двух частей верхняя панель управления списком книг и основная часть со списком всех книг. В верхней панели нужно нажать на кнопку «Добавить» и затем откроется диалоговое окно выбора файла электронной книги (рисунок 4.13).

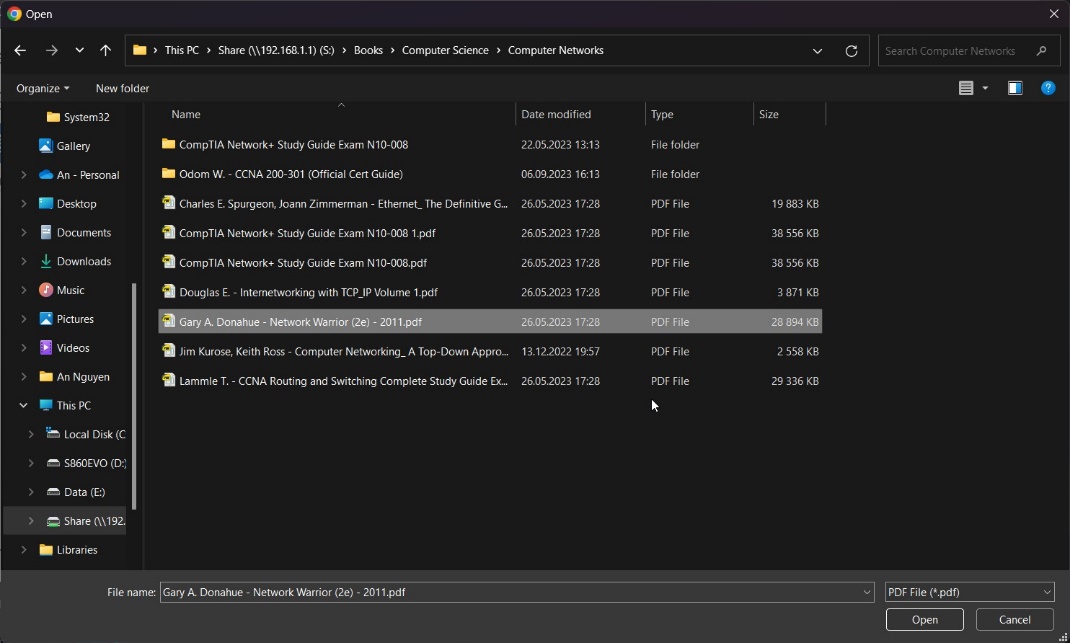


Рисунок 4.13 – Диалоговое окно выбора файла

Затем после выбора файла, откроется окно для ввода информации о книге, такие как название книги, описание, ISBN, авторы выбранной книги. Ввод всех этих данных не обязателен. Название книги и его авторов могут быть заполнены на стороне сервера. Окно ввода данных о книге изображено на рисунке 4.14.

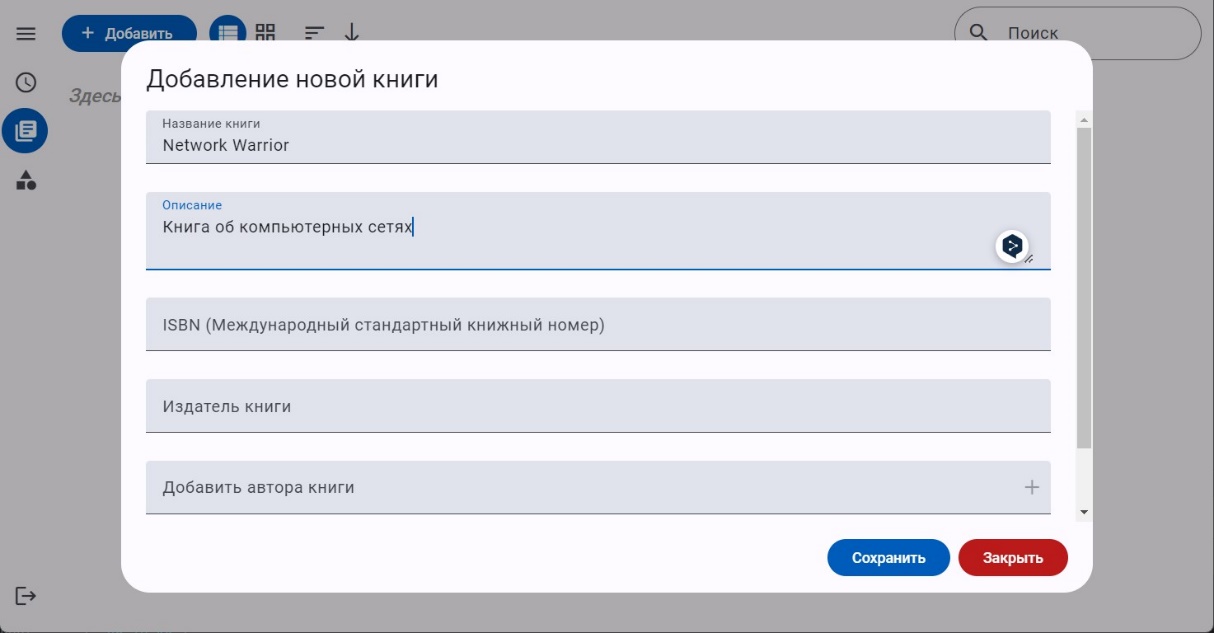


Рисунок 4.14 – Окно ввода данных об электронной книге

Далее нужно нажать на кнопку «Сохранить». После этого это окно должно закрыться и должно вывести сообщение об успешности создания в нижней части страницы (рисунок 4.15).

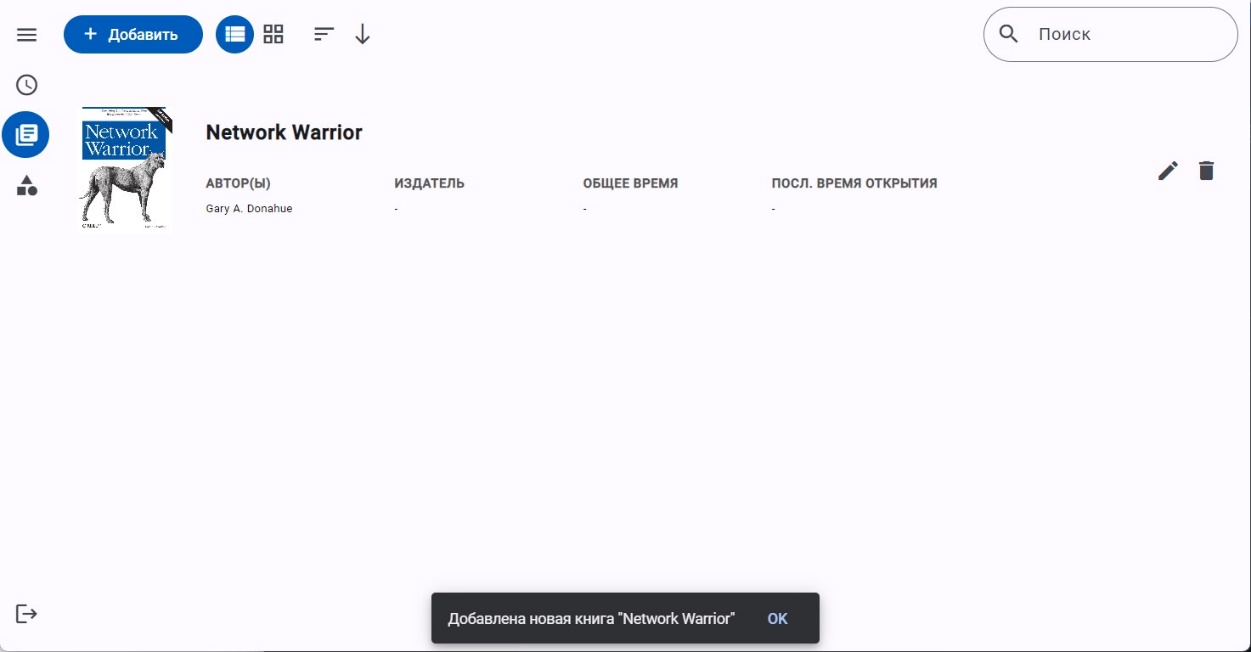


Рисунок 4.15 – Результат добавления новой книги

Каждую запись книги можно изменить или удалить. При нажатии на кнопку «Редактировать» (имеет изображение в виде «ручки»), открывается диалоговое окно аналогичное окну добавления книги. Окно редактирования изображена на рисунке 4.16.

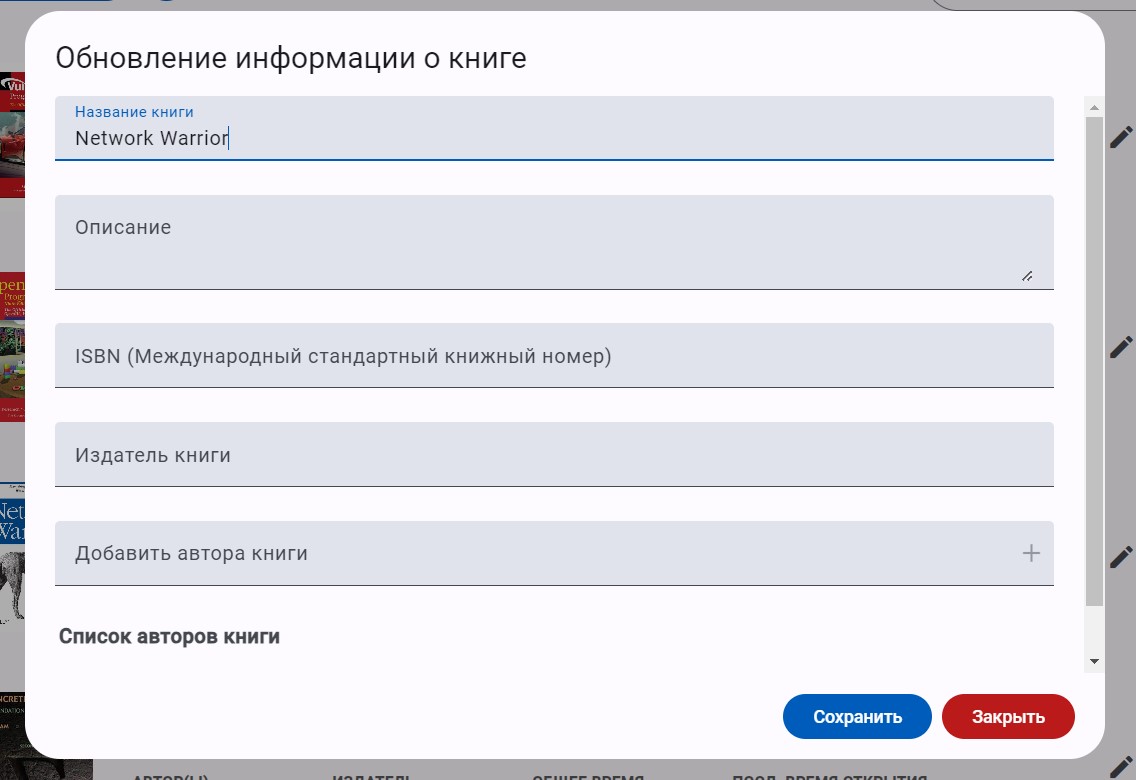


Рисунок 4.16 – Окно редактирование информации о книге

Также на странице есть возможности переключения вид вывода списка книг в верхней панели управления, выполненных в виде 2 кнопок после кнопки добавления книг (рисунок 4.17).



Рисунок 4.17 – Кнопки управление видом списка

Ещё есть возможность выбрать сортировку книг по названию, по ISBN или по последней дате открытия. Для этого нужно нажать на кнопку с иконкой в виде трёх линий разной длины (рисунок 4.18).

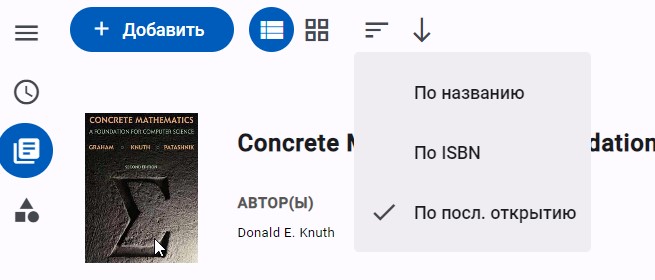


Рисунок 4.18 – Меню выбора сортировки по определенному свойству

За этой кнопкой находится кнопка с иконкой в виде стрелки, отвечающая за порядок сортировки книг по убыванию или возрастанию (рисунок 4.19).

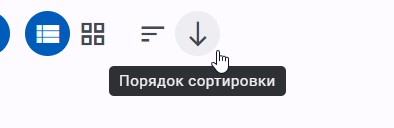


Рисунок 4.19 – Кнопка «Порядок сортировки»

Вид «сетка» представлена на рисунке 4.20.

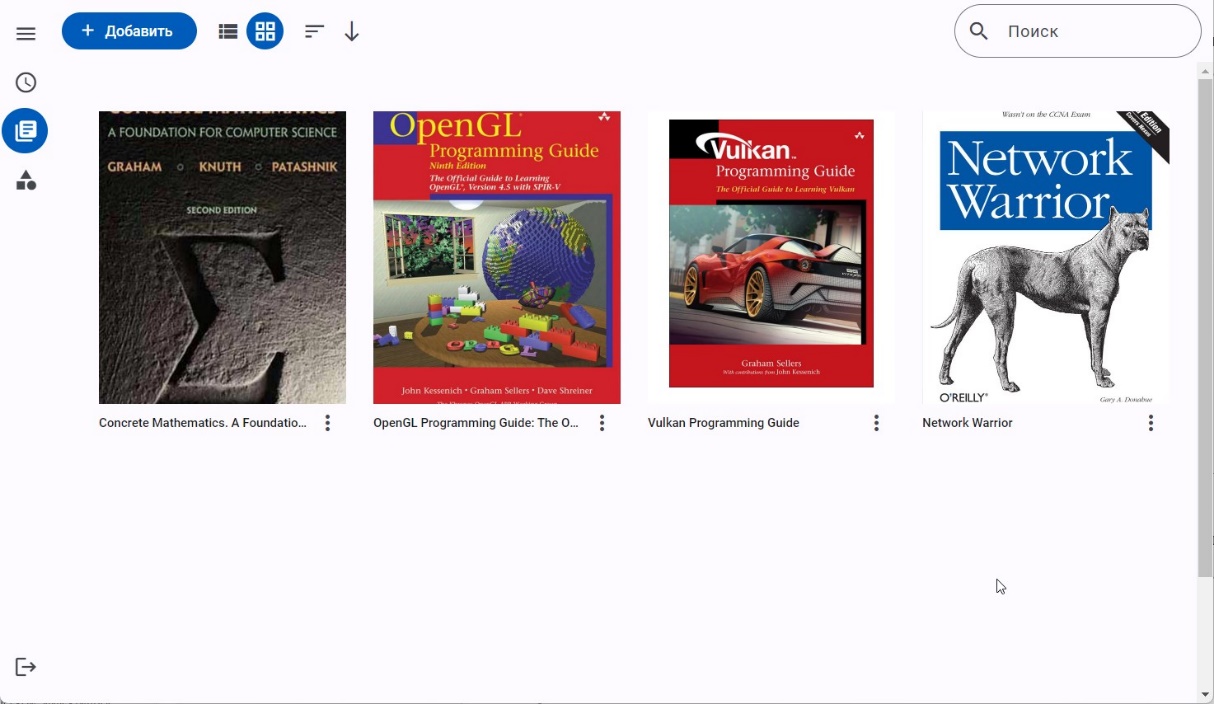


Рисунок 4.20 – Список книг в виде «сетки»

Чтобы открыть книгу, нужно просто нажать на обложку книги, если список в виде сетки, если же в виде обычно списка, то нужно просто навестись на любую подсвеченную часть записи документа (рисунок 4.21).



Рисунок 4.21 – Наведение на запись документа

После нажатия откроется страница просмотра книги (рисунок 4.22).

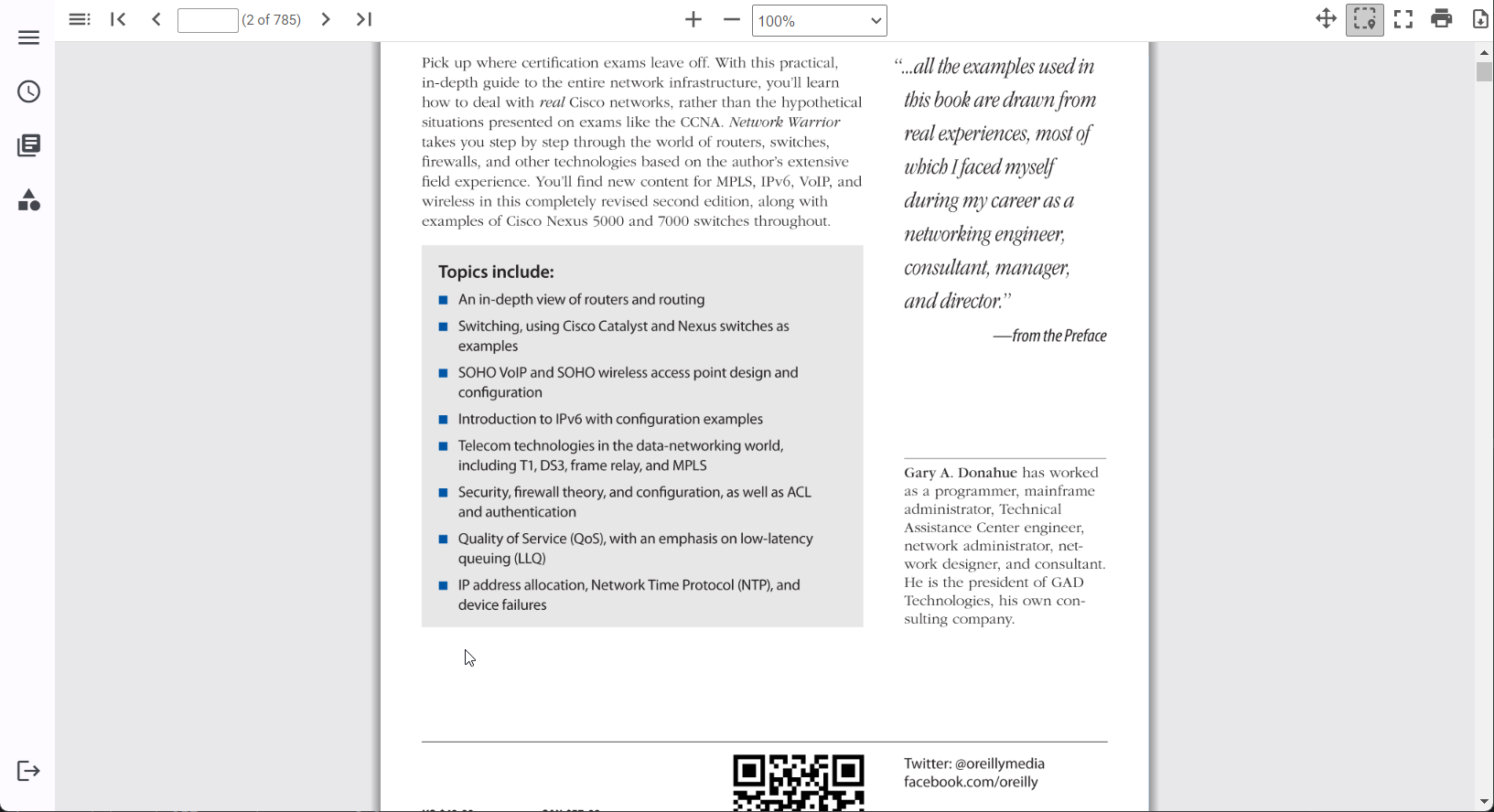


Рисунок 4.22 – Страница просмотра документа

Также как и со страницей списка книг, на этой странице есть верхняя панель управления и панель просмотра документа. В самой левой части панели управления находится кнопка в виде трёх линии с точками, которая открывает панель навигации по разделам в документе. Панель навигации по страницам изображена на рисунке 4.23.



Рисунок 4.23 – Кнопка панели навигации в документе

В нём находится список разделов в виде дерева, по которому можно перейти, нажав на любой их этих разделов. Она представлена на рисунке 4.24.

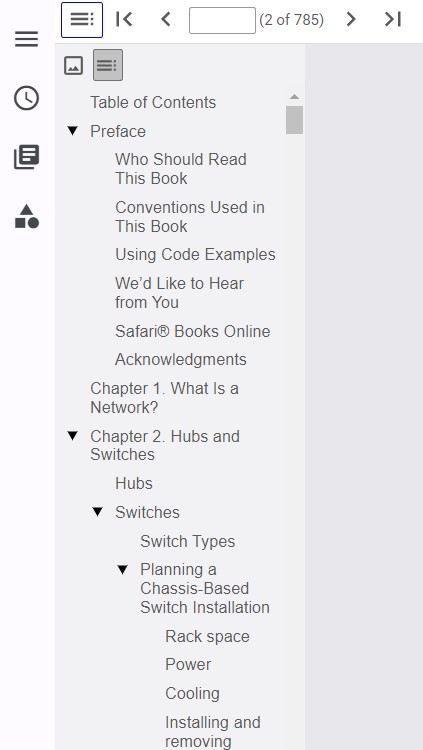


Рисунок 4.24 – Панель навигации в документе

Далее после кнопки открытия/закрытия панели навигации в документе находится элементы управления страницей (рисунок 4.25).

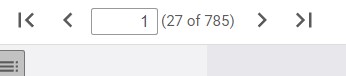


Рисунок 4.25 – Элементы управления страницей в документе

Список функции элементов слева направо, изображенных на рисунке 4.25:

* переход на первую страницу документа;
* переход на предыдущую страницу;
* поле ввода номера страницы;
* переход на следующую страницу;
* переход на последнюю страницу документа.

В центральной части панели управления элементы масштабирования страницы (рисунок 4.26).



Рисунок 4.26 – Управление масштабированием страницы

В правой части панели управления находятся следующие кнопки (рисунок 4.27):

* включение Hand Tool, курсор используется для перемещения по документу;
* включение режим выделения текста;
* просмотр документа в полноэкранном режиме;
* печать документа;
* скачать файл документа.



Рисунок 4.27 – Правая часть панели управления просмотра

При выделении текста появляется контекстное меню. Оно состоит из 4 кнопок: копирование текста, перевод, нахождение определения слова и обобщения текста. Контекстное меню изображено на рисунке 4.28.

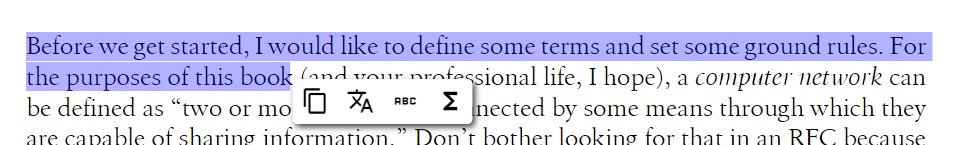


Рисунок 4.28 – Контекстное меню

Перевод и обобщение текста осуществляется моделями нейронных сетей от Яндекса. После нажатия на одну из этих кнопок открывается диалоговое окно и отправляется запрос серверной части приложения, а затем там формируется запрос к облачным серверам Яндекса. После получения ответа она передаётся клиентской части. Обобщение текста имеет ограничение на максимальное количество входных токенов. Токен это смысловые отрывки или часто встречающихся последовательностей символов, характерных для естественного языка [8]. Пример перевода текста и обобщения текста изображены на рисунке 4.29 и 4.30.

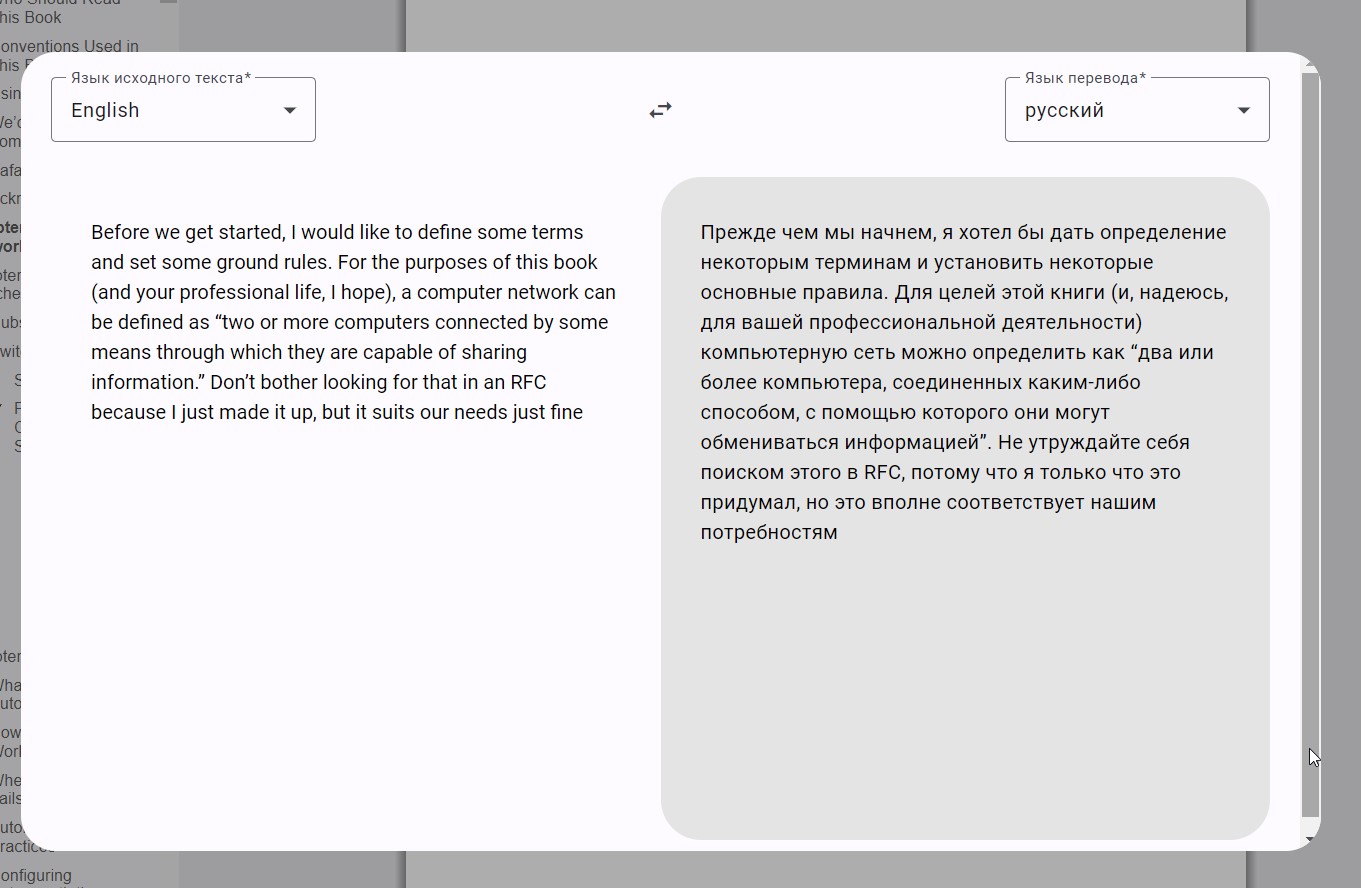


Рисунок 4.29 – Диалог перевода текста

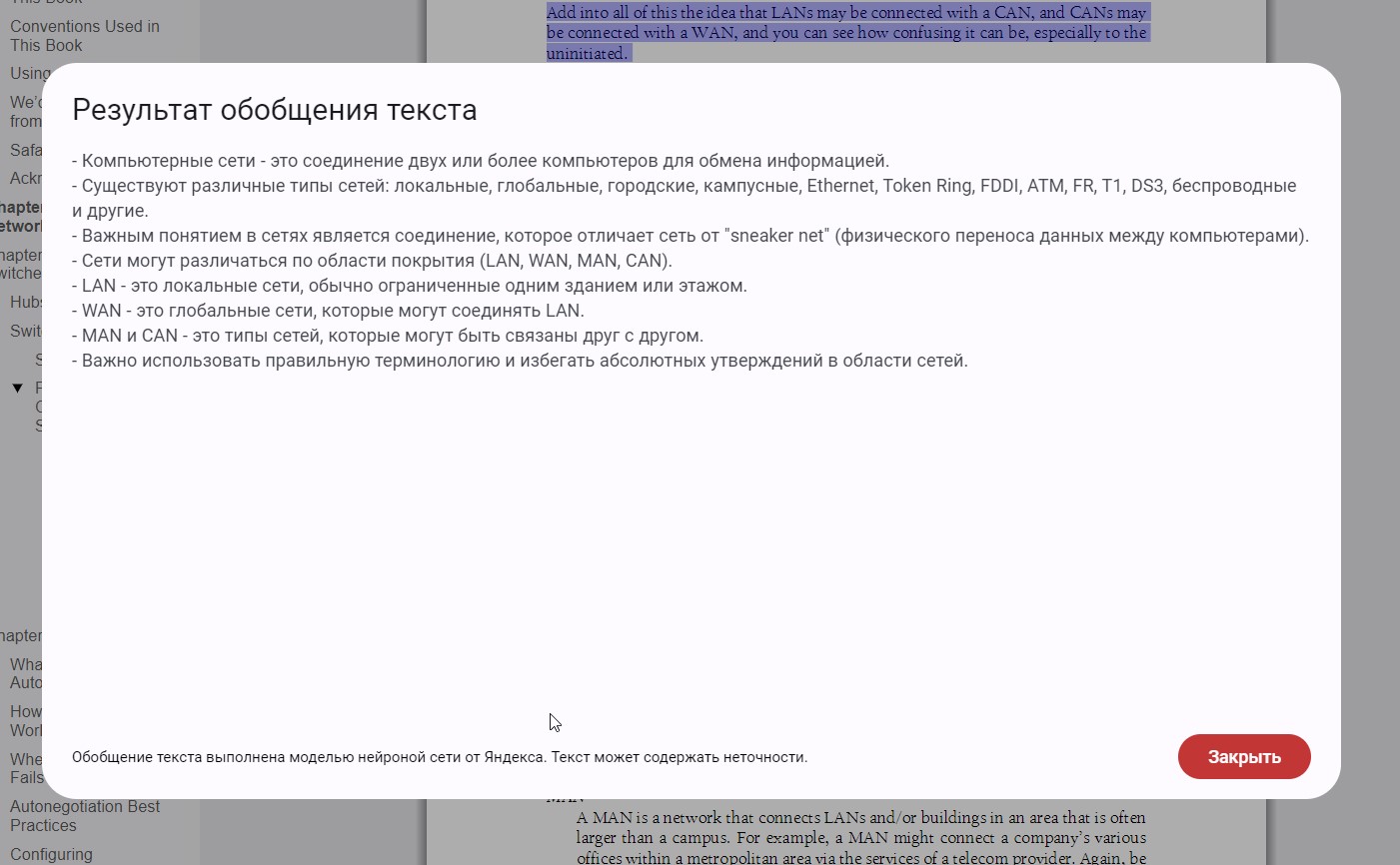


Рисунок 4.30 – Диалог обобщения текста

Также имеется возможность объединения книг в виде «коллекции». Для этого нужно перейти на страницу коллекции книг, нажав на кнопку с иконками с геометрическими фигурами (рисунок 4.31).

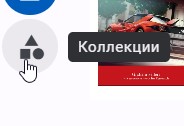


Рисунок 4.31 – Кнопка страницы «Коллекции» в панели навигации

Нажав на неё, открывается страница, представленная на рисунке 4.x.

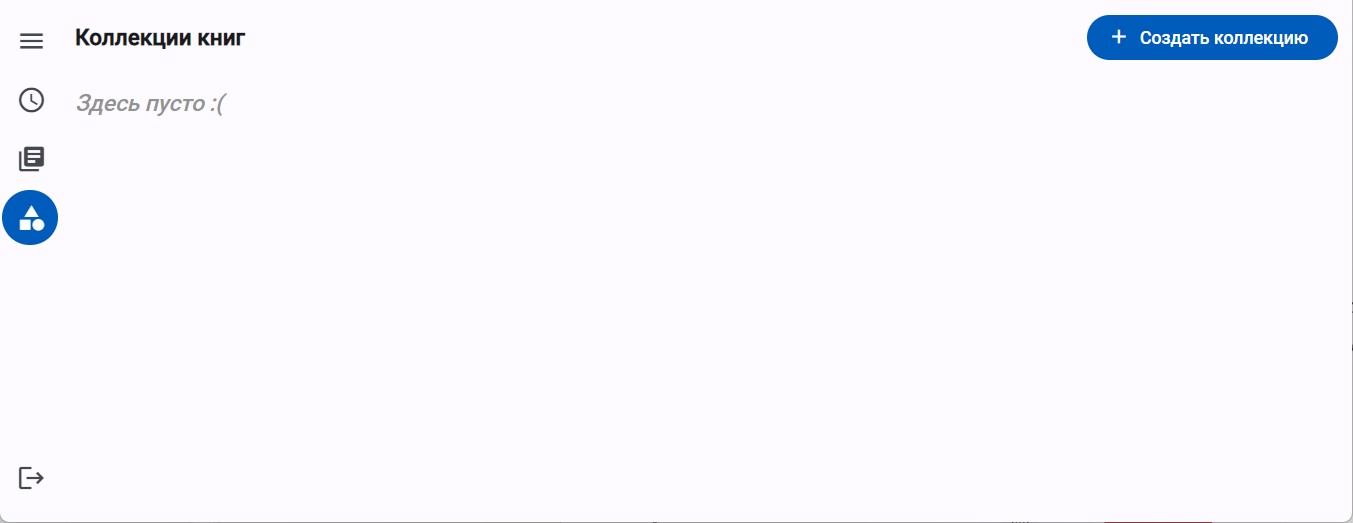


Рисунок 4.x – Страница «Коллекции книг»

# 5 ТЕСТИРОВАНИЕ СИСТЕМЫ

Тестирование — это проверка логики программной системы на правильность работы в соответствии с техническими требованиями. Существуют два способа тестирования: ручной и автоматизированный. Под ручным подразумевается, что разработчик самостоятельно тестирует систему напрямую или через специальные программы, такие как Postman (для HTTP API). Автоматизированные тесты, по сути, представляют собой отдельные написанные программы для проверки корректности поведения системы в целом или отдельных его частей. Существуют три основных вида автоматизированных тестов:

* Юнит тестирование (Unit testing).
* Интеграционное тестирование (integration testing).
* Функциональное тестирование (functional testing).

Алгоритм автоматизированного теста представляет собой следующее:

* создаётся входная информация/объект для тестируемой функции;
* входная информация/объект передаётся этой функции;
* тестируемая функция возвращает определенный результат;
* полученный результат сравнивается с ожидаемым результатом и, если они не равны, выводится ошибка, иначе выводится, что тест успешно пройден.

Иногда перед началом теста или тестов требуется вызвать определенную функцию, которая может создать схему базу данных, добавить туда записи или удалить их и т.п. Обычно сторонний пакет (библиотека) для тестирования, позволяет это сделать.

В разработке программ не всегда целесообразно писать автоматизированные тесты, так как написание автоматизированных тестов также занимает время, как и написание самого программного обеспечения.

## 5.1 Условия и порядок тестирования серверной части

Для серверной части были написаны интеграционные тесты. Под интеграционным тестом подразумевается тестирование какой-либо отдельной части системы. В данном случае тестируется web API сервер.

Алгоритм тестирования определенной функции системы имеет следующий вид:

* запуск серверной части приложения с конфигурацией для тестирования (осуществляется программным способом в коде для тестирования);
* авторизоваться в системе, если это необходимо;
* формируется определенный HTTP-запрос;
* отправка запроса в тестируемую систему;
* получение ответа от системы;
* если ответ системы соответствует ожидаемому результату, например HTTP-ответ с кодом ошибки 400 Bad Request, то тест считается успешно пройденным.

Такие тесты реализованы в виде методов класса. Код тестов расположен в проекте «src/Testing». Для тестов используется пакет «XUnit» версии 2.8.0. Для их запуска нужно ввести следующую команду в консоли находясь в корневой директории проекта:

*dotnet test*

Либо это можно осуществить через встраиваемую среду разработки. На примере JetBrains Rider, нужно в левой панели навигации нужно нажать на кнопку с иконкой  «Unit Testing», затем в раскрывшийся панели (рисунок 5.1) нажать на кнопку  «Run All Tests».

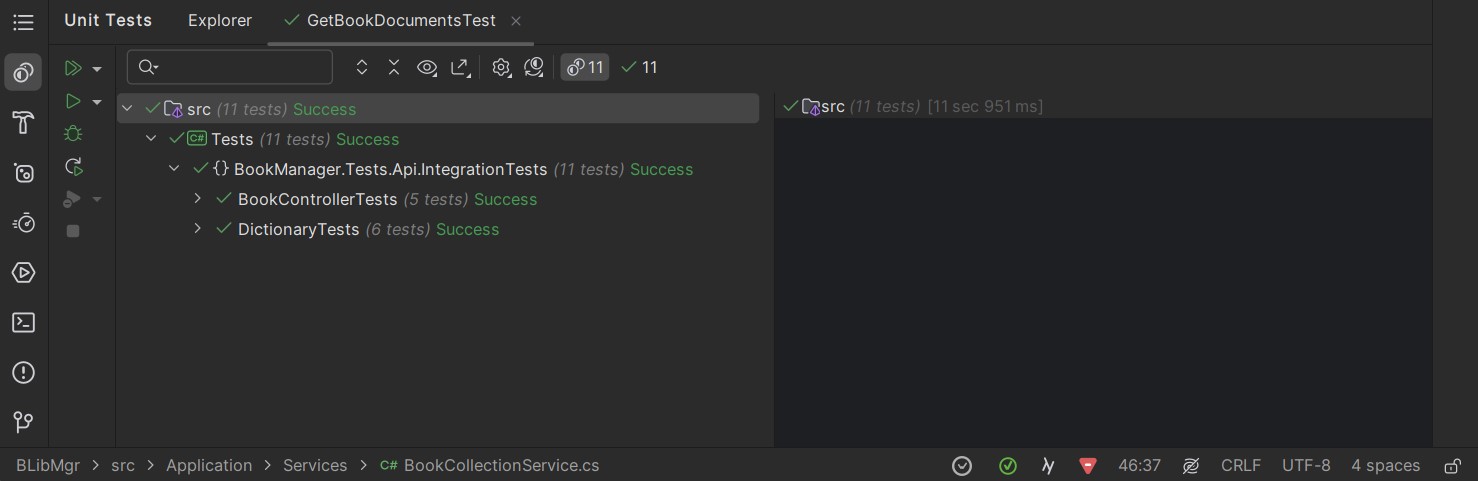


Рисунок 5.1 – Панель Unit Testing

## 5.2 Условия и порядок тестирования клиентской части

Все тесты для клиентской части системы выполнены вручную с использованием методологии черного ящика. Алгоритм ручного тестирования выглядит следующим образом:

* запустить серверную и клиентскую часть приложения в режиме отладки;
* открыть веб-браузер со следующим URL адресом http://localhost:4200;
* проверить определенный функционал приложения через веб-браузер, и если поведение не соответствует определенным правилам, то тест считается проваленным, иначе тест пройден успешно.

## 5.3 Результаты тестирования

-

# 6 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## 6.1 Расчет показателя трудоёмкости для программного продукта

Величина параметра трудоёмкости для разрабатываемого программного продукта состоит из суммы значений трудоёмкости для каждого этапа разработки и рассчитывается по формуле:

,

где – общая трудоемкость разработки программного продукта;

– трудоемкость работ на i-й стадии разработки;

n – общее количество этапов разработки.

В таблице 11 приведены данные о расчете величины трудоемкости для каждого из этапов разработки и для всего проекта в целом.

Таблица 6.1 – Поэтапная и общая оценка трудоемкости разработки

|  |  |  |
| --- | --- | --- |
| **Этап работы** | **Вид работы** | **Трудоемкость (чел. \* час)** |
| Формирование требований к системе | Исследование объекта предметной области  Обоснование необходимости создания системы  Анализ требований к системе | 20 |
| Техническое задание | Разработка и утверждение технического задания на систему | 30 |
| Разработка системы | Разработка системы на языке программирования | 360 |
| Тестирование системы | Проведение тестирования разработанной системы на тестовых данных.  Устранение ошибок. | 80 |
| Рабочая документация | Разработка рабочей документации на систему | 60 |
| Общая нагрузка | | 550 |

## 6.2 Расчет затрат на материальные ресурсы

Расчет затрат на материальные ресурсы производится по формуле:

,

где – расход i-го вида материального ресурса, натуральные единицы;

– цена за единицу i-го вида материального ресурса;

i – вид материального ресурса;

n – общее количество всех видов материальных ресурсов.

Результаты расчетов затрат на материальные ресурсы приведены   
в таблице 6.2.

Таблица 6.2 – Затраты на материальные ресурсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование** | **Единица измерения** | **Количество израсходованного материала** | **Цена за единицу, руб.** | **Сумма, руб.** |
| Компьютер | шт. | 1 | 70 000 | 70 000 |
| Монитор | шт. | 1 | 20 000 | 20 000 |
| Итого | | | | 90 000 |

При разработке системы не возникло необходимости в использовании расходных материалов.

Общая сумма затрат на электроэнергию рассчитывается по формуле:

,

где – паспортная мощность i-го электрооборудования, кВт;

– время работы i-го оборудования за весь период разработки, ч;

Ц – тариф электроэнергии, руб./кВт × ч;

i – вид электрооборудования;

n – количество электрооборудования.

Необходимые расчеты затрат на электроэнергию приведены в таблице 6.3.

Таблица 6.3 – Затраты на электроэнергию

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Наименование** | **Паспортная мощность,**  **кВт** | **Время работы оборудования** | **Тариф электроэнергии,**  **руб./кВтч** | **Сумма,**  **руб.** |
| Компьютер | 0,5 | 600 | 4,7 | 1410 |
| Монитор | 0,10 | 600 | 4,7 | 282 |
| Освещение | 0,04 | 200 | 4,7 | 37,6 |
| Итого | | | | 1 729,6 |

## 6.3 Расчёт затрат на разработку системы

Определение затрат на разработку производится путем составления соответствующей сметы, которая включает следующие статьи:

* затраты на оплату труда;
* отчисления на социальные нужды;
* амортизация основных фондов.

## 6.4 Расчёт затрат на оплату труда

Общая сумма затрат на оплату труда определяется по формуле:

,

где – часовая ставка i-го работника, руб.;

– время на разработку системы, ч;

i – порядковый номер работника;

n – количество работников.

Часовая заработная плата программиста рассчитывается по формуле:

,

где – среднемесячная заработная плата разработчика системы, руб.;

– среднемесячный фонд рабочего времени.

Стоимость одного часа работы программиста равна 300 руб.

Общая сумма затрат на оплату труда равна:

550 ч. × 300 руб. = 165 000 руб.

## 6.5 Расчет отчислений на социальные нужды

Данные об отчислениях на социальные нужды представлены в таблице 14.

Таблица 6.4 – Отчисления на социальные нужды

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид** | **Начислено заработной платы, руб.** | **Отчисления**  **%** | **Сумма,**  **руб.** |
| Фонд социального страхования РФ | 165 000 | 2,9 | 4785 |
| Фонд обязательного медицинского страхования | 5,1 | 8415 |
| Пенсионный фонд РФ | 22 | 36 300 |
| Итого | | | 49 500 |

## 6.6 Себестоимость проекта

В таблице 6.5 представлен расчет себестоимости проекта.

Таблица 6.5 – Себестоимость проекта

|  |  |
| --- | --- |
| **Статьи затрат** | **Сумма, руб.** |
| Затраты на материальные ресурсы | 90 000 |
| Затраты на оплату труда | 165 000 |
| Отчисление на социальные нужды | 49 500 |
| Итого по смете | 304 500 |

## 6.7 Расчёт плановой прибыли

Прибыль П рассчитывается по формуле

,

где – полная себестоимость, руб.;

– норма рентабельности.

При норме рентабельности 50% плановая прибыль составить:

П = 304 500 50% / 100% = 152 250 руб.

Полная стоимость проекта определяется как сумма себестоимости проекта и прибыли:

= 304 500 + 152 250 = 456 750 руб.

С учетом налога на прибыль 20% доход составит

152 250 – 152 250  0,2 = 121 800 руб.

## 6.8 Определение экономической эффективности проекта

Определение экономической эффективности проекта проводилось по методу расчета экономического эффекта от прибыли по следующей формуле:

,

где – экономический эффект, %;

– себестоимость, руб.;

П – прибыль (за вычетом налога на прибыль), руб.

Экономический эффект равен:

= (121 800 / 456 750) 100% = 26,66 %.

## 6.9 Выводы по технико-экономическому анализу и обоснованию проекта разработки

В результате проведения технико-экономического анализа проекта, были рассчитаны несколько показателей, с помощью которых было получено технико-экономическое обоснование разработки. Себестоимость проекта составила   
321 600 руб. Полная стоимость проекта составила 456 750 руб. Доход от внедрения системы – 121 800 руб. Экономический эффект составил 26,66 %.

Исходя из показателей, разработка системы является эффективной и принесет прибыль.

# ЗАКЛЮЧЕНИЕ

В данной дипломной работе была разработана программная система персональной библиотеки и в него был интегрирован искусственный интеллект, которые дополняет или/и заменяет некоторый функционал системы, были построены схемы алгоритмов обмена информацией между приложением и API искусственного интеллекта и схема алгоритма добавления электронной книги, разработана руководство пользователя. Также была описана экономическая составляющая проекта, такие как затраченное количество человеко-часов на написание технического задания, программного обеспечения, тестирования, подсчет затрат на весь проект, себестоимость проекта, расчёт рентабельности и плановой прибыли.

Также в данном проекте мы почти прошли весь жизненный цикл разработки программного обеспечения от анализа предметной области проекта до выпуска программного продукта.

Также в ходе данной работы были освоены современные инструменты разработки, исследованы возможности интеграции ИИ в разрабатываемой системе, получены навыки разработки, написания технической документации и внедрение программной системы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. – введ. 01.01.92. – М.: Изд-во стандартов, 2010. – 24 с.
2. Документация Angular Material: [Электронный ресурс] / Google LLC, 2024. – Режим доступа: <https://material.angular.io/components/categories>, свободный.
3. Документация Angular: [Электронный ресурс] / Google LLC, 2024. – Режим доступа: <https://angular.io/docs>, свободный.
4. Документация ASP.NET: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0>, свободный.
5. Документация Docker: [Электронный ресурс] / Docker Inc., 2024. – Режим доступа: <https://docs.docker.com/guides>, свободный.
6. Документация Entity Framework Core: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/ef/core/>, свободный.
7. Документация Windows Subsystem for Linux: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/windows/wsl/about>, свободный.
8. Определение слова «Токен»: [Электронный ресурс] / Yandex, 2024 – Режим доступа: <https://yandex.cloud/ru/docs/foundation-models/concepts/yandexgpt/tokens?utm_referrer=https%3A%2F%2Fwww.google.com%2F>, свободный.
9. A tour of C# language: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>, свободный.
10. Ajay D. Kshemklayni. Distributed Computing: Principles, Algorithms, and Systems / Ajay D. Kshemklayni, Mukesh Singhal. – Cambridge University Press, 2008.
11. Artificial Intelligence: A Modern Approach, 4th US ed. / Stuart J. Russell., Peter Norvig. – Prentice Hall, 2020
12. Distributed Systems / Marteen Van Steen, Tannenbaum A.S.

# ПРИЛОЖЕНИЕ А

Код «Program.cs»

-