**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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 НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение   
высшего 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 | *Разработка и реализация системы* | *31.05.2024* |
| 2 | *Написание пояснительной записки* | *31.05.2024* |
| 3 | *Создание чертежей* | *02.06.2024* |
| 4 | *Прохождение нормоконтроля* | *?* |
| 5 | *Предварительная защита?* | *?* |
| 6 | *Защита* | *22.06.2024* |
|  |  |  |
|  |  |  |

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

**АННОТАЦИЯ**

Выпускная квалификационная работа Нгуен Хыу Ана по теме «Интеллектуальный помощник для работы с электронной библиотекой». Руководитель Кандаулов Валерий Михайлович. Работа защищена на кафедре «Измерительно-вычислительные комплексы» УлГТУ в 2024 году.

Пояснительная записка: 62 страницы, 6 разделов, 1 приложения, 43 рисунков, 14 таблиц, 16 источников.

Ключевые слова: электронная библиотека, Angular , ASP.NET Core, Entity Framework Core, TypeScript, C#, искусственный интеллект.

Разработанная автоматизированная система электронной библиотеки предназначена для централизованного хранения электронных книг и удобной работы с ними. Представляет собой программное обеспечение который использует веб-технологии. Программное обеспечение реализовано на языке программирования C# и TypeScript.

Содержание

[CПИСОК ИСПОЛЬЗОВАННЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ 8](#_Toc168503888)

[ВВЕДЕНИЕ 9](#_Toc168503889)

[1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ СИСТЕМЫ 11](#_Toc168503890)

[1.1 Назначение и цели создания системы 11](#_Toc168503891)

[1.2 Характеристика объекта автоматизации 11](#_Toc168503892)

[1.2.1 Общее описание 11](#_Toc168503893)

[1.2.2 Состав и порядок функционирования 11](#_Toc168503894)

[1.3 Общие требования к системе 12](#_Toc168503895)

[1.3.1 Требования к структуре и функционированию системы 12](#_Toc168503896)

[1.3.2 Дополнительные требования 13](#_Toc168503897)

[1.4 Требования к функциям, выполняемым системой 13](#_Toc168503898)

[1.4.1 Функция предоставления списка документов 13](#_Toc168503899)

[1.4.2 Функция просмотра документа 14](#_Toc168503900)

[1.4.3 Функция перевода текста 14](#_Toc168503901)

[1.4.4 Функция нахождения определения выбранного слова 14](#_Toc168503902)

[1.4.5 Функция обобщения текста 14](#_Toc168503903)

[1.5 Требования к видам обеспечения 14](#_Toc168503904)

[1.5.1 Требования к информационному обеспечению 14](#_Toc168503905)

[1.5.2 Требования к алгоритмическому обеспечению 14](#_Toc168503906)

[1.5.3 Требования к программному обеспечению 15](#_Toc168503907)

[1.6 Анализ аналогичных разработок 15](#_Toc168503908)

[1.6.1 Calibre 15](#_Toc168503909)

[1.6.2 Kavita 17](#_Toc168503910)

[1.6.3 Koodo 18](#_Toc168503911)

[2 Информационное обеспечение системы 20](#_Toc168503912)

[2.1 Выбор средств управления данными 20](#_Toc168503913)

[2.2 Проектирование базы данных 22](#_Toc168503914)

[2.3 Организация сбора, передачи, обработки и выдачи информации 25](#_Toc168503915)

[3 Алгоритмическое обеспечение системы 26](#_Toc168503916)

[3.1 Алгоритм обмена информации между приложением и API искусственного интеллекта 26](#_Toc168503917)

[3.2 Алгоритм добавления электронной книги в библиотеку 26](#_Toc168503918)

[4 Программное обеспечение системы 29](#_Toc168503919)

[4.1 Структура программного обеспечения и функции его компонентов 29](#_Toc168503920)

[4.2 Выбор компонентов программного обеспечения 29](#_Toc168503921)

[4.2.1 Клиентская операционная система 29](#_Toc168503922)

[4.2.2 Инструментальное средство разработки и язык программирования 30](#_Toc168503923)

[4.2.3 Вспомогательное программное обеспечение 31](#_Toc168503924)

[4.3 Разработка прикладного программного обеспечения 31](#_Toc168503925)

[4.3.1 Структура прикладного программного обеспечения серверной части 31](#_Toc168503926)

[4.3.2 Структура прикладного программного обеспечения клиентской части 33](#_Toc168503927)

[4.4 Особенности реализации, эксплуатации и сопровождения системы 33](#_Toc168503928)

[4.5 Руководство пользователя 34](#_Toc168503929)

[4.5.1 Требования к условиям эксплуатации 34](#_Toc168503930)

[4.5.2 Инсталляция и настройка 34](#_Toc168503931)

[4.5.3 Порядок и особенности работы 36](#_Toc168503932)

[5 ТЕСТИРОВАНИЕ СИСТЕМЫ 50](#_Toc168503933)

[5.1 Условия и порядок тестирования серверной части 51](#_Toc168503934)

[5.2 Условия и порядок тестирования клиентской части 53](#_Toc168503935)

[5.3 Результаты тестирования 53](#_Toc168503936)

[6 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 56](#_Toc168503937)

[6.1 Расчет показателя трудоёмкости для программного продукта 56](#_Toc168503938)

[6.2 Расчет затрат на материальные ресурсы 57](#_Toc168503939)

[6.3 Расчёт затрат на разработку системы 58](#_Toc168503940)

[6.4 Расчёт затрат на оплату труда 58](#_Toc168503941)

[6.5 Расчет отчислений на социальные нужды 59](#_Toc168503942)

[6.6 Себестоимость проекта 59](#_Toc168503943)

[6.7 Расчёт плановой прибыли 59](#_Toc168503944)

[6.8 Определение экономической эффективности проекта 60](#_Toc168503945)

[6.9 Выводы по технико-экономическому анализу и обоснованию проекта разработки 60](#_Toc168503946)

[ЗАКЛЮЧЕНИЕ 61](#_Toc168503947)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 62](#_Toc168503948)

[ПРИЛОЖЕНИЕ А 64](#_Toc168503949)

# 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, торговля, 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 работающий 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;
* функция обобщения текста.

Первоочередность реализации функций указана по порядку.

### 1.4.1 Функция предоставления списка документов

В системе должна быть реализована функция предоставления списка книг, в виде обычного списка и виде сетки из обложек.

### 1.4.2 Функция просмотра документа

В системе должна быть реализована функция просмотра документа книги с навигацией по страницам, возможность перехода по главам и возможность масштабирования.

### 1.4.3 Функция перевода текста

В системе должна быть реализована функция перевода выбранного пользователем текста в документе, с поддержкой разных естественных языков.

### 1.4.4 Функция нахождения определения выбранного слова

В системе должна быть реализована функция нахождения определения для выделенного пользователем в документе слова и вывода определения в виде отдельного (диалогового) окна или через меню.

### 1.4.5 Функция обобщения текста

В системе должна быть реализована функция обобщения выделенного пользователем текста в книге. Обобщение должно быть выведено в отдельном диалоговом окне.

## 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;
* 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 1.1.

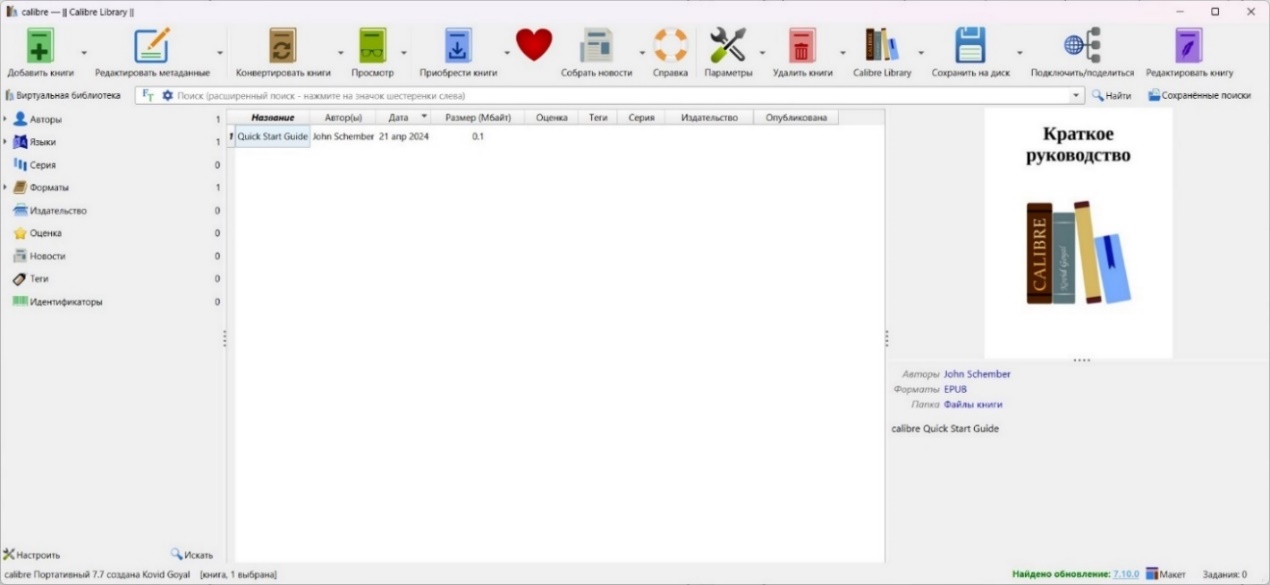


Рисунок 1.1 – Окно программы Calibre

### 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 1.2.

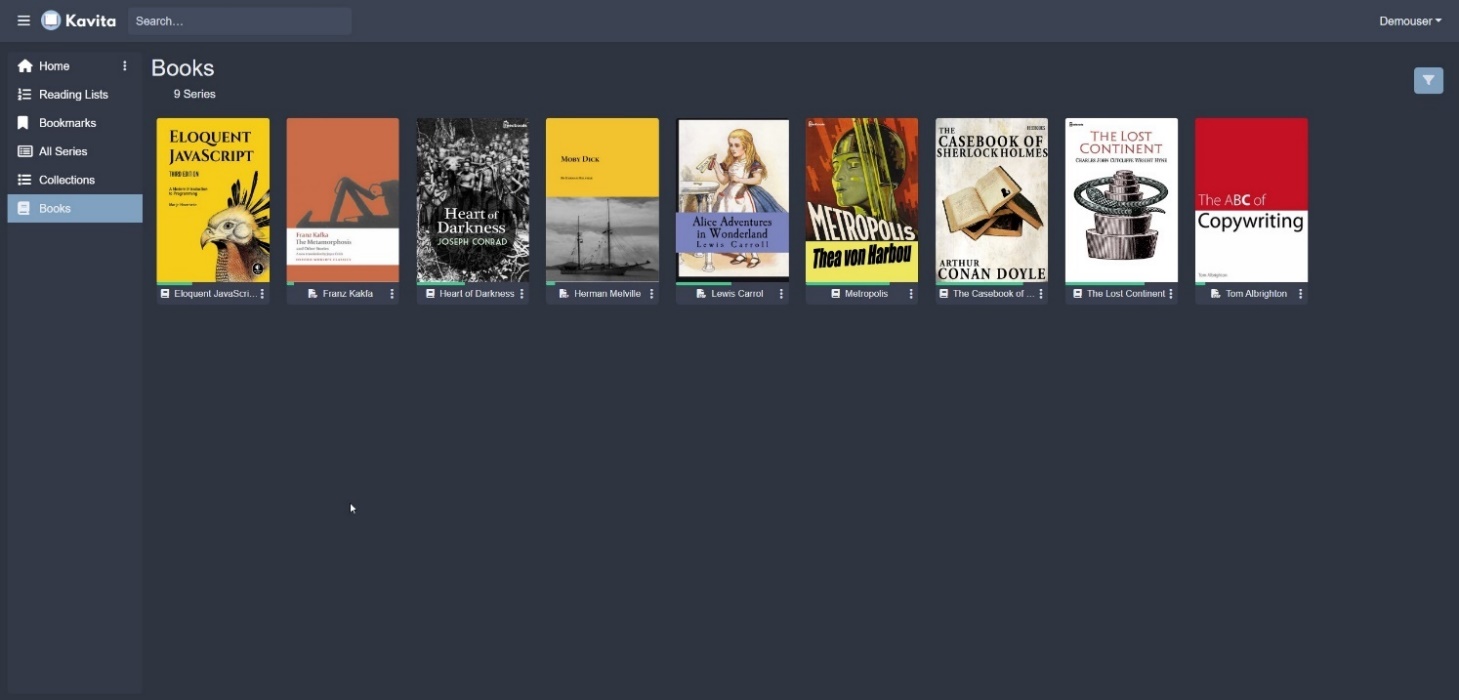


Рисунок 1.2 – Веб-страница программы Kavita

### 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 1.3.

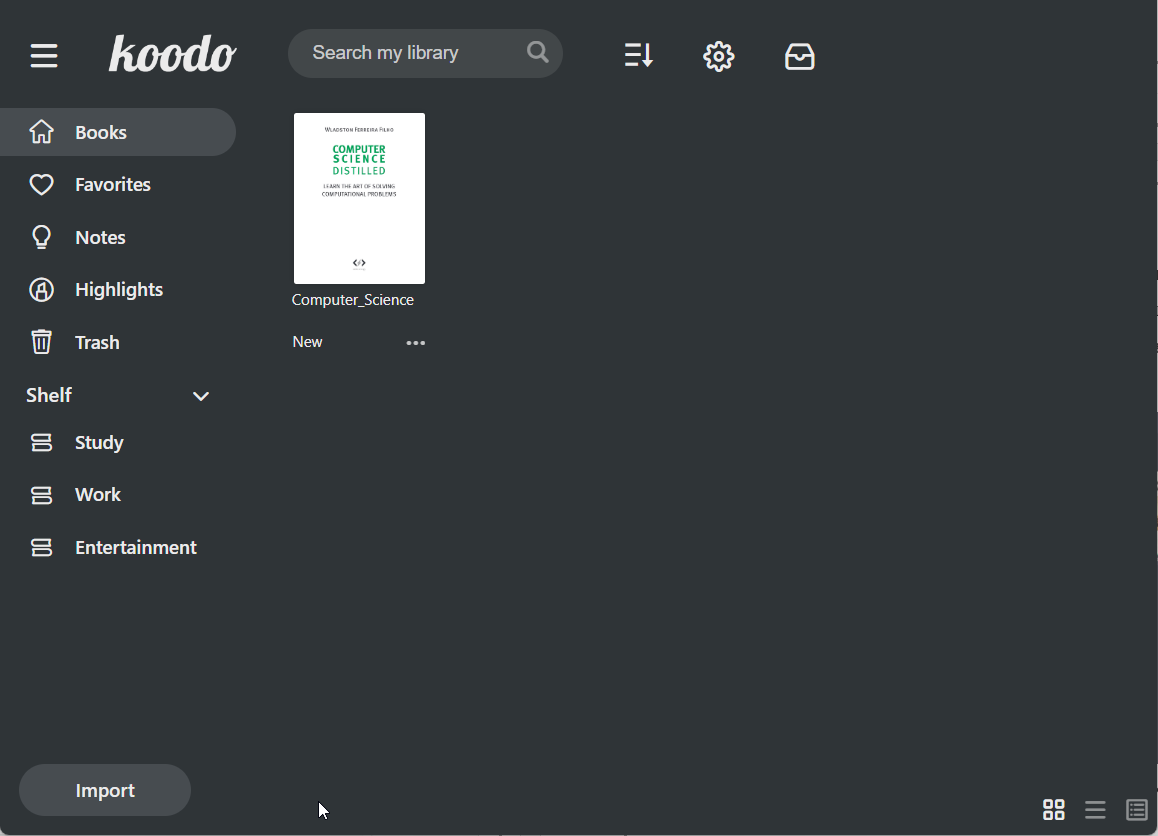


Рисунок 1.3 – Страница программы Kodoo

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

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, коллекции 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.

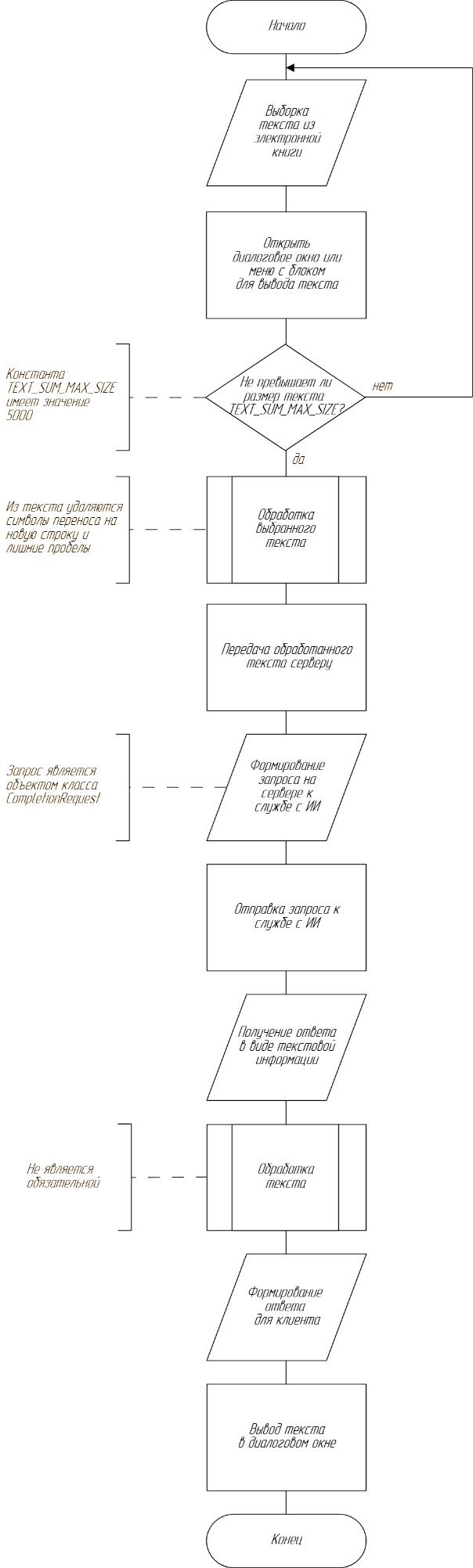


Рисунок 3.1 – Схема алгоритма обмена информации между приложением и API исскуственного интеллекта

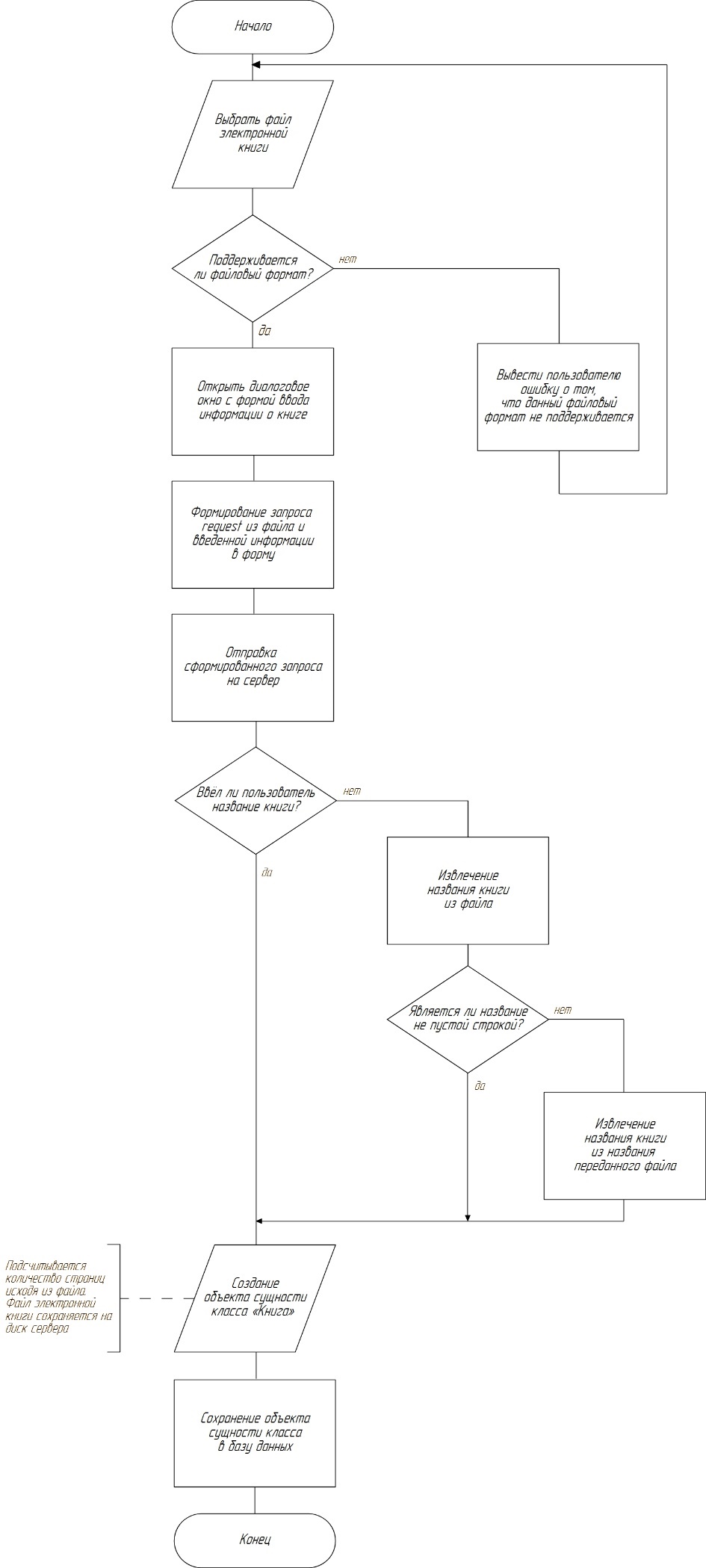


Рисунок 3.2 – Схема алгоритма добавления электронной книги в библиотеку

# 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.

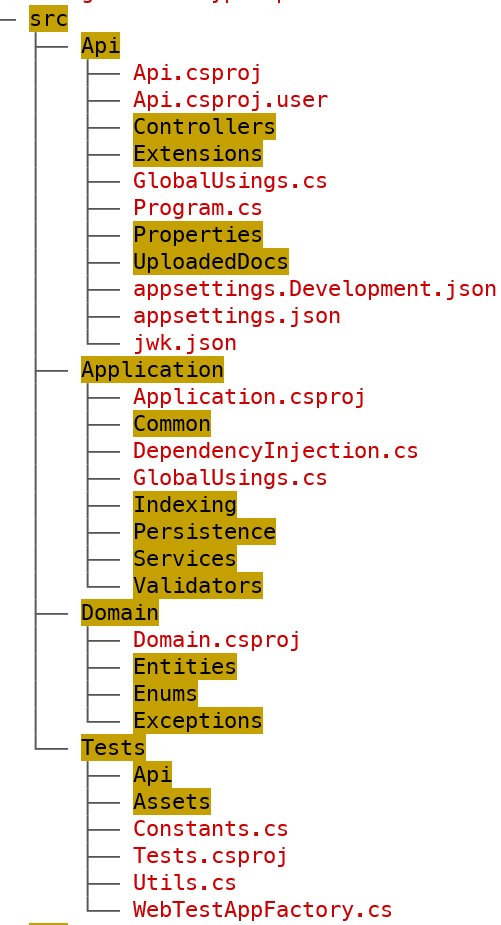


Рисунок 4.1 - Структура серверной части

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

В данном разделе будет описываться требования к условиям эксплуатации, установка и настройка серверной части системы, порядок и особенности работы с интерфейсной частью приложения.

### 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 следующий текст на рисунке 4.3.

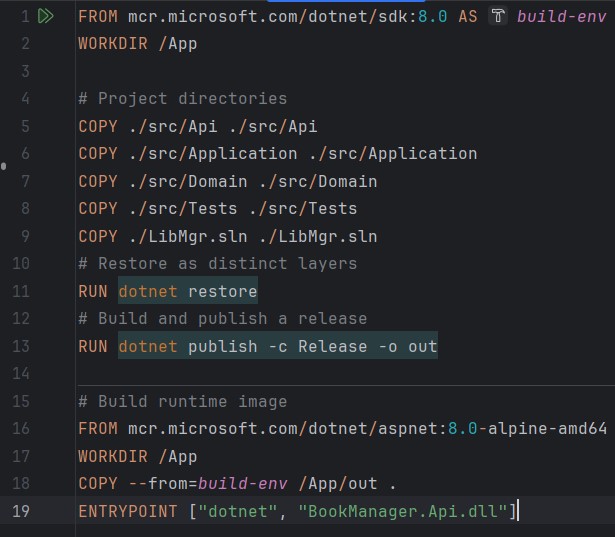


Рисунок 4.3 – Текст Dockerfile (сервер)

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.

Таким же образом было написан Dockerfile и для клиентской части.

### 4.5.3 Порядок и особенности работы

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

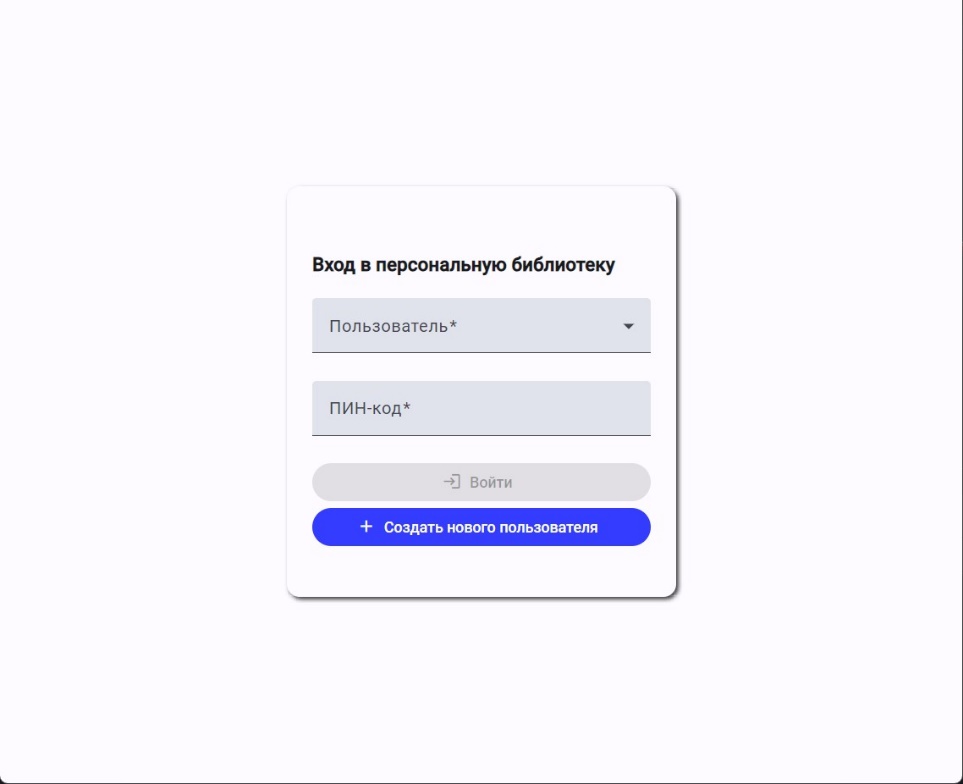


Рисунок 4.4 – Страница аутентификации

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.

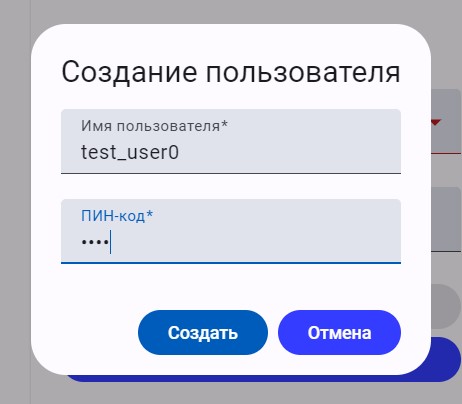


Рисунок 4.5 – Диалоговое окно создания аккаунта пользователя

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.

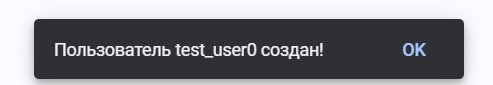


Рисунок 4.6 – Вывод сообщения об успешности создания пользователя

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

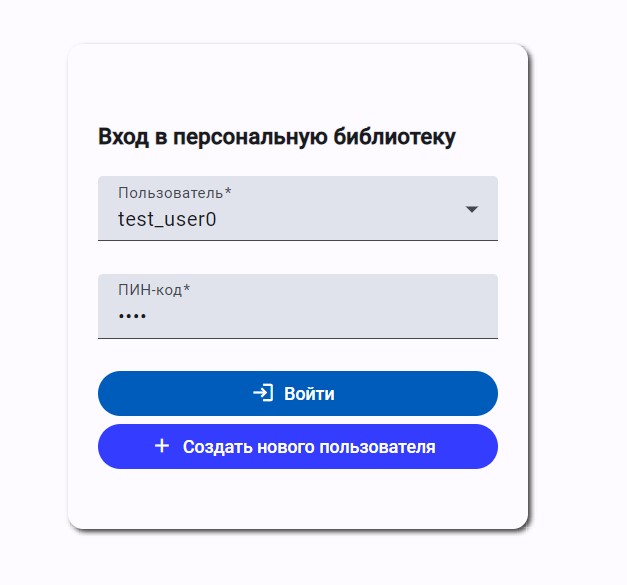


Рисунок 4.7 – Ввод имени пользователя и ПИН-кода

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).

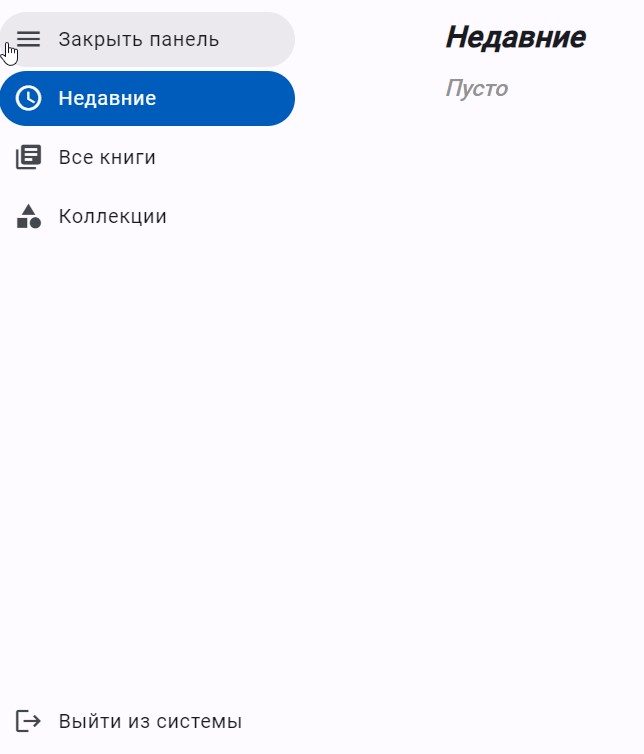


Рисунок 4.9 – Раскрытая панель навигации

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

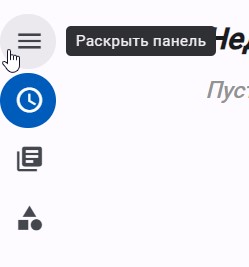


Рисунок 4.10 – Надпись рядом кнопкой навигации

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

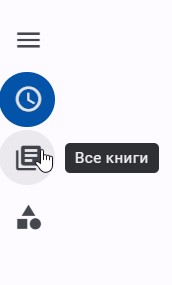


Рисунок 4.11 – Кнопка для перехода на страницу со списком электронных книг

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).

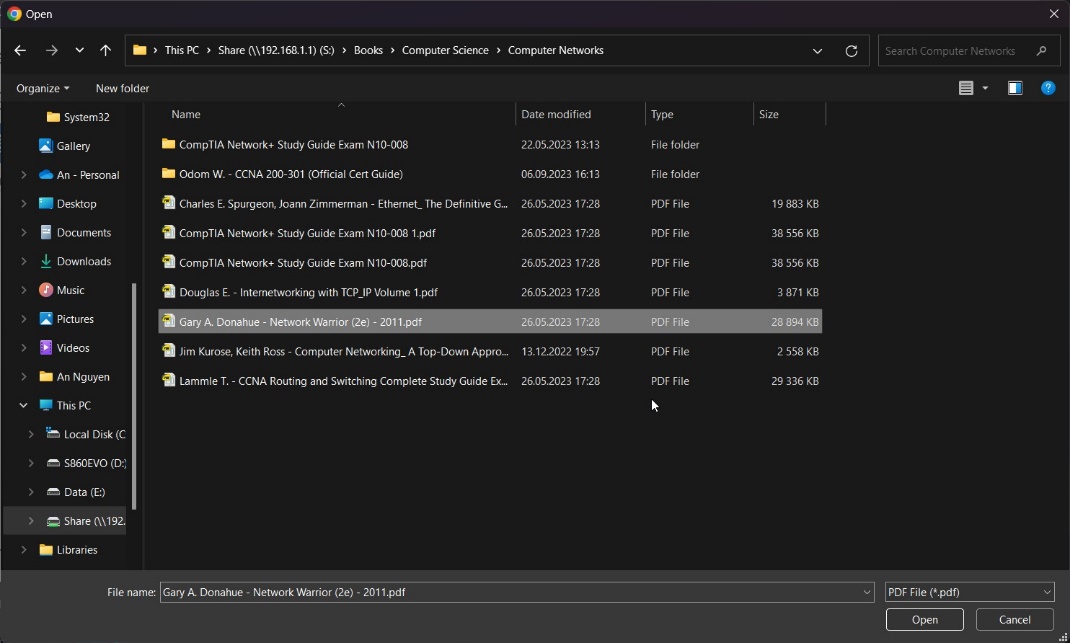


Рисунок 4.13 – Диалоговое окно выбора файла

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.

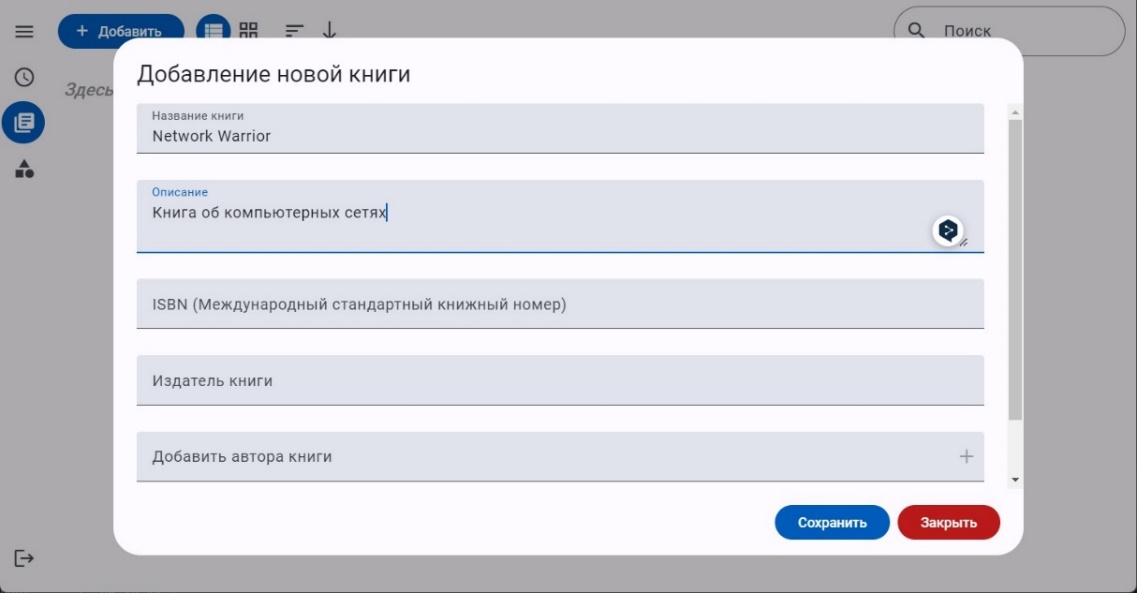


Рисунок 4.14 – Окно ввода данных об электронной книге

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).

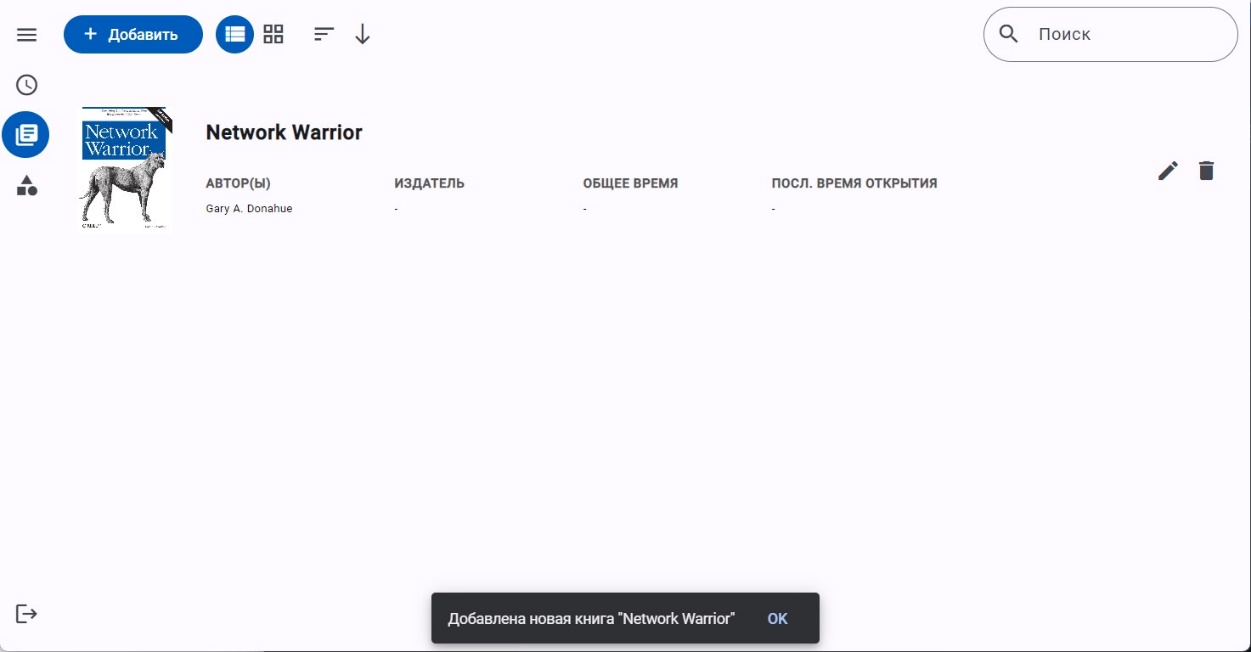


Рисунок 4.15 – Результат добавления новой книги

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.

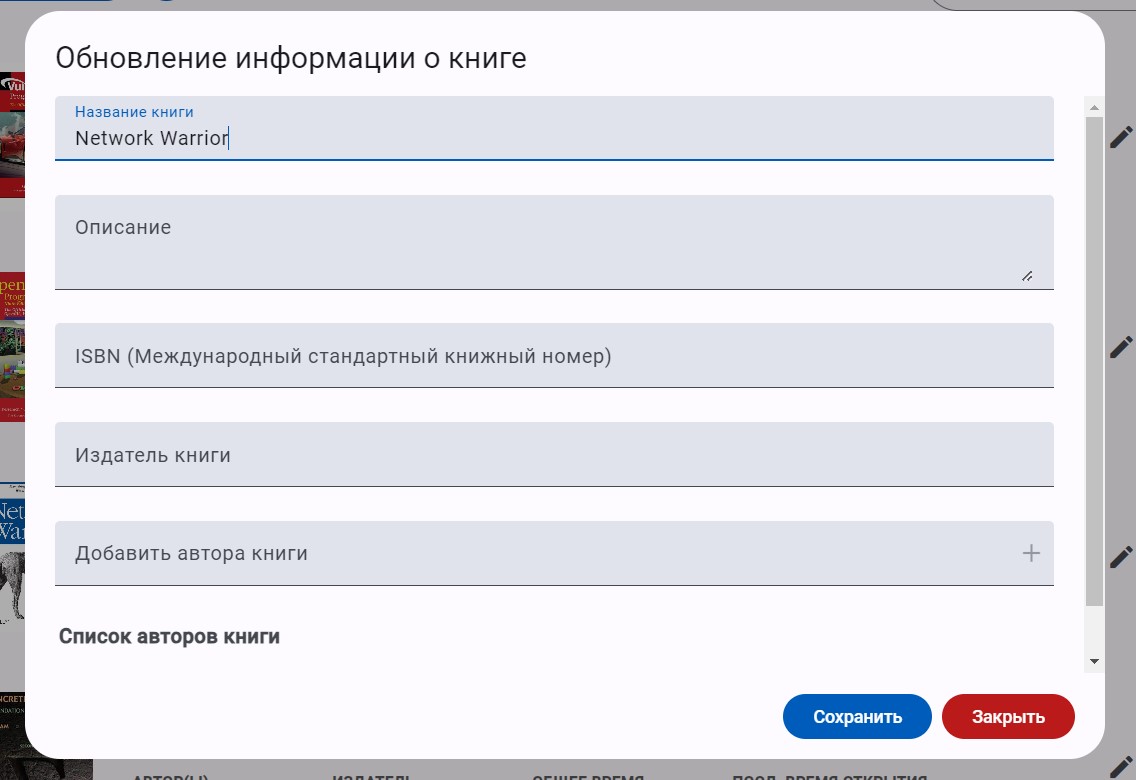


Рисунок 4.16 – Окно редактирование информации о книге

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).

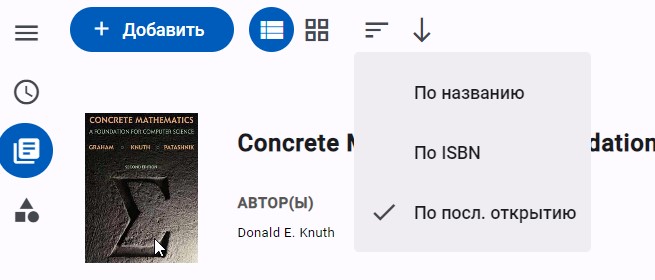


Рисунок 4.18 – Меню выбора сортировки по определенному свойству

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

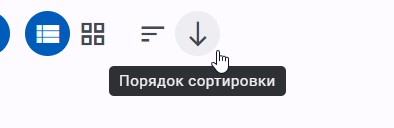


Рисунок 4.19 – Кнопка «Порядок сортировки»

Вид «сетка» представлена на рисунке 4.20.

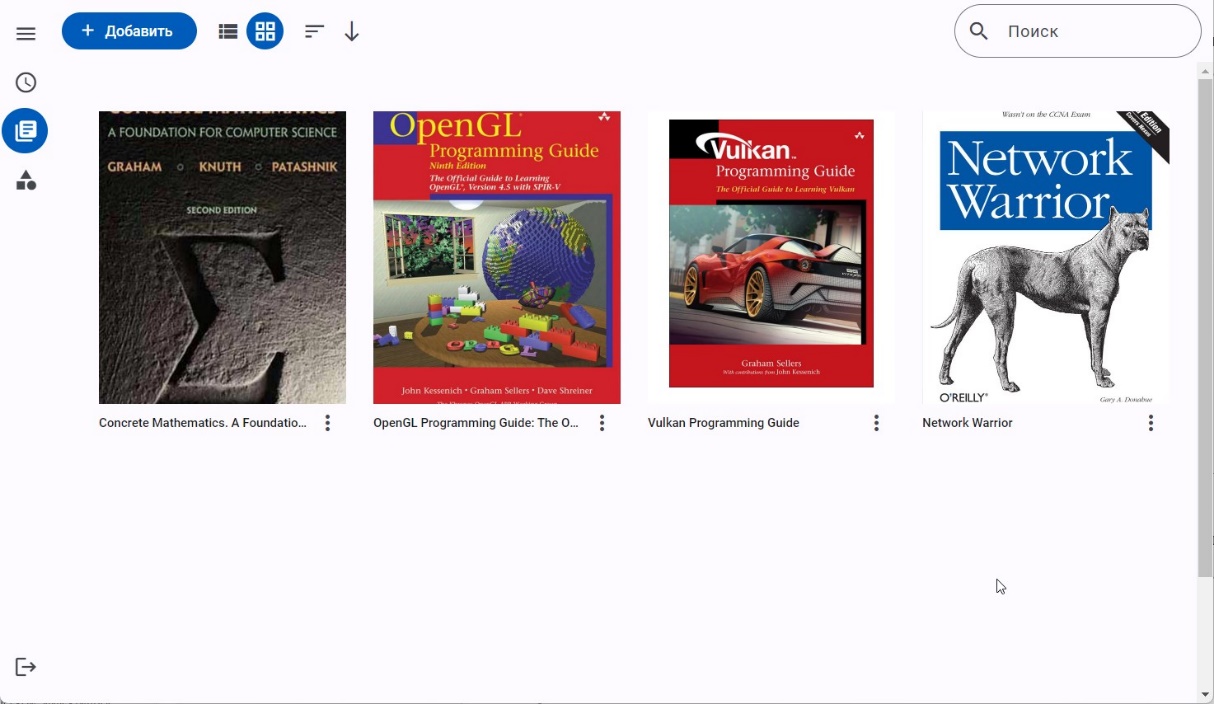


Рисунок 4.20 – Список книг в виде «сетки»

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).

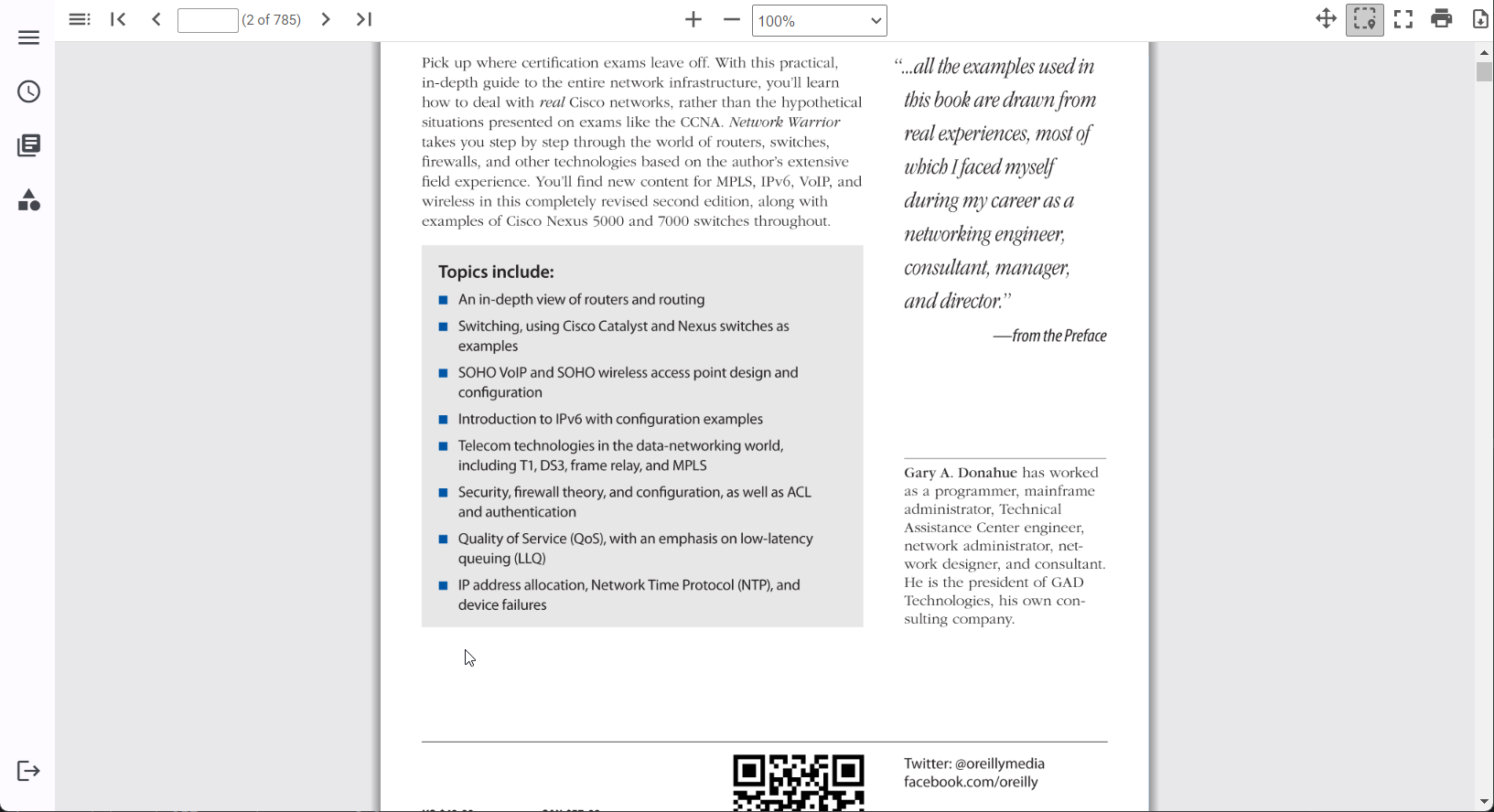


Рисунок 4.22 – Страница просмотра документа

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.

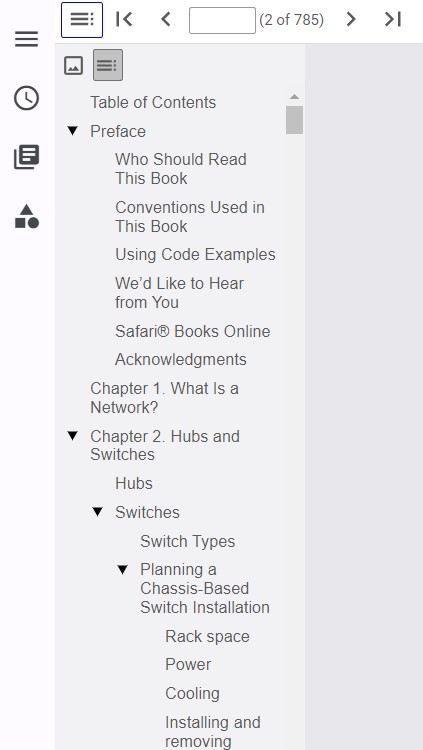


Рисунок 4.24 – Панель навигации в документе

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

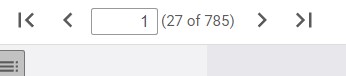


Рисунок 4.25 – Элементы управления страницей в документе

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.

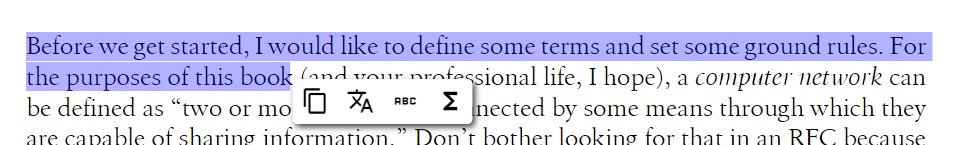


Рисунок 4.28 – Контекстное меню

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.

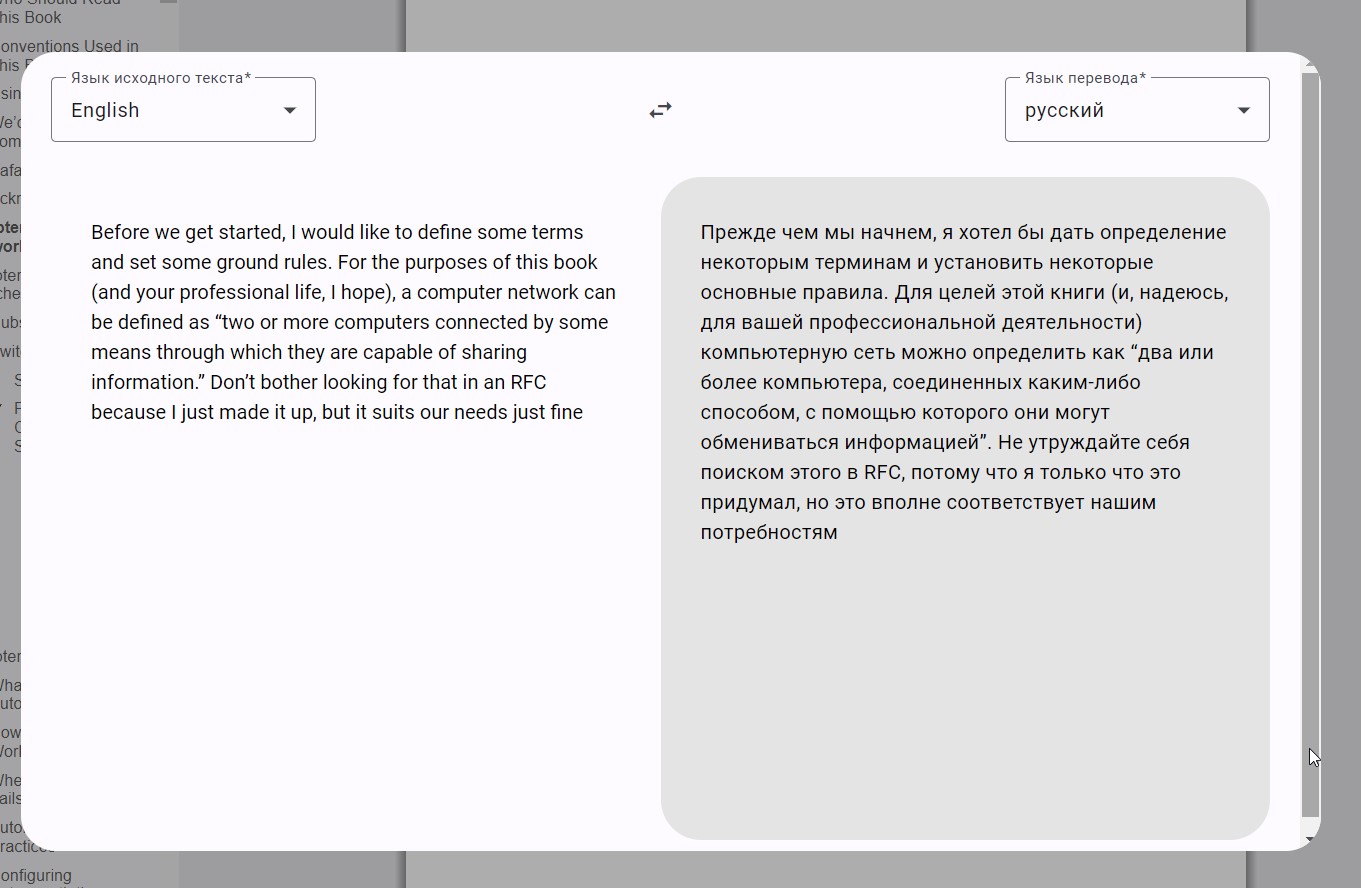


Рисунок 4.29 – Диалог перевода текста

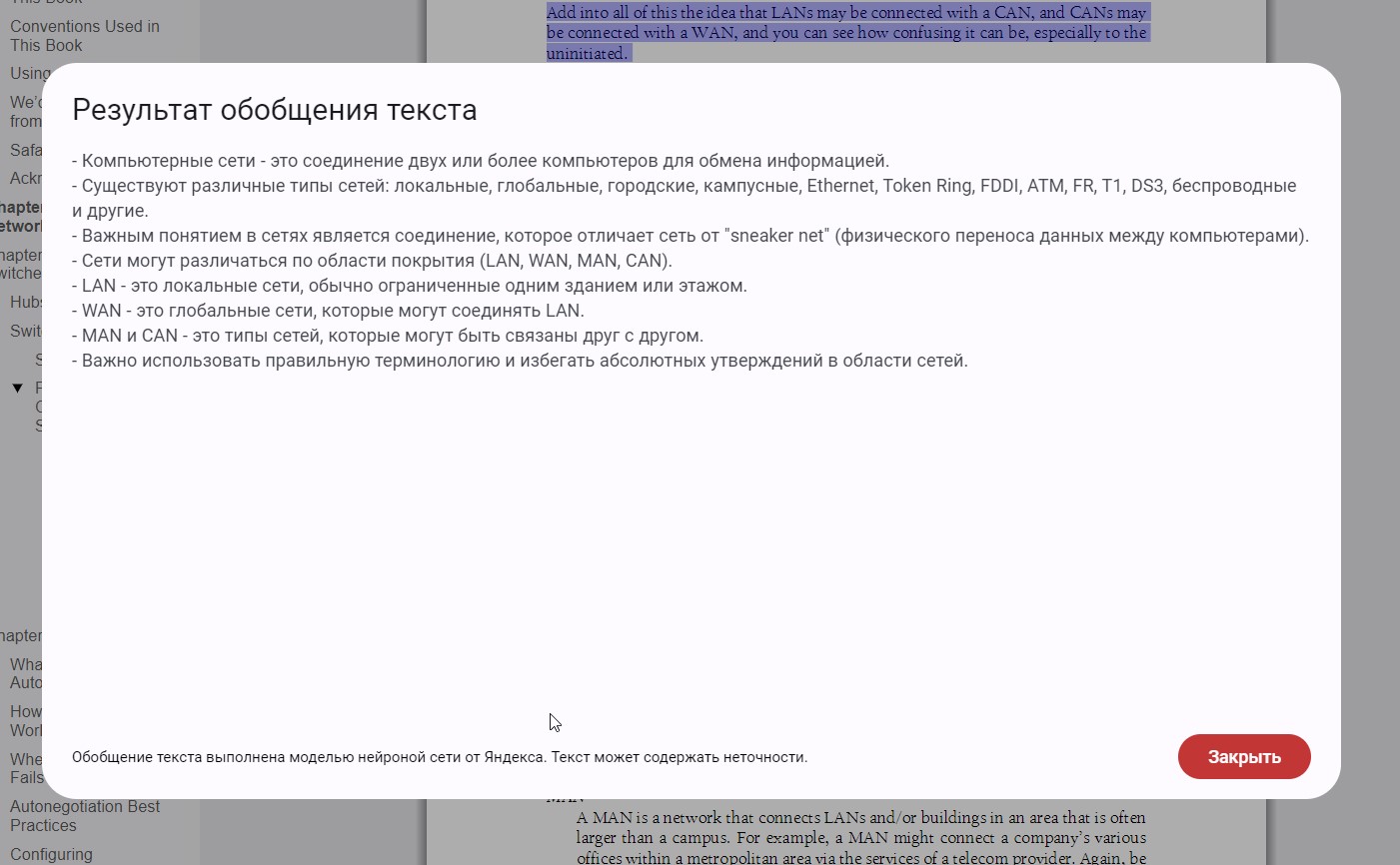


Рисунок 4.30 – Диалог обобщения текста

Нахождение определения для слова или фразы может осуществляться одними из двух служб – Merriam Webstar и Яндекса. Также как и с переводом и обобщением текста, запрос от клиента обрабатывается серверной частью, затем передаётся сторонней службе. Ответ затем передаётся обратно через сервер клиенту. Определения слова выводится в виде блока после меню (рисунок 4.31). Стороннюю службу можно выбирать в этом блоке. Если результат не будет найден, то выводится сообщение об этом в нижней части страницы и самом блоке с определениями.

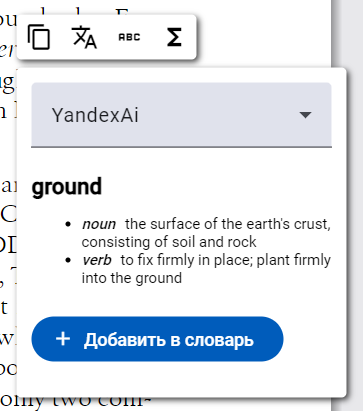


Рисунок 4.31 – Блок с определением слова

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 4.32).

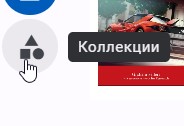


Рисунок 4.32 – Кнопка страницы «Коллекции» в панели навигации

Нажав на неё, открывается страница, представленная на рисунке 4.33.

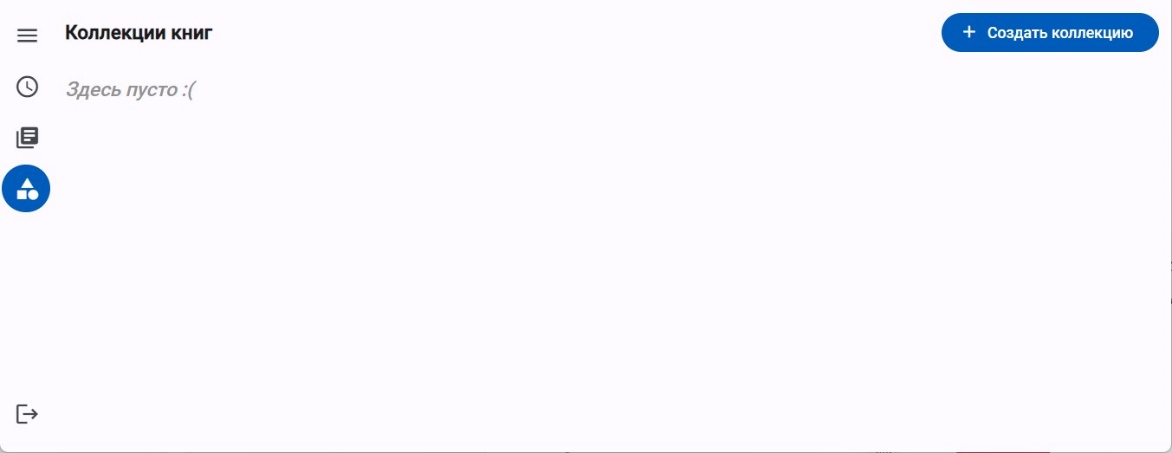


Рисунок 4.33 – Страница «Коллекции книг»

Чтобы создать коллекцию книг нужно нажать кнопку «Создать коллекцию» в верхнем правом углу страницы. Затем откроется диалоговое окно (рисунок 4.34).

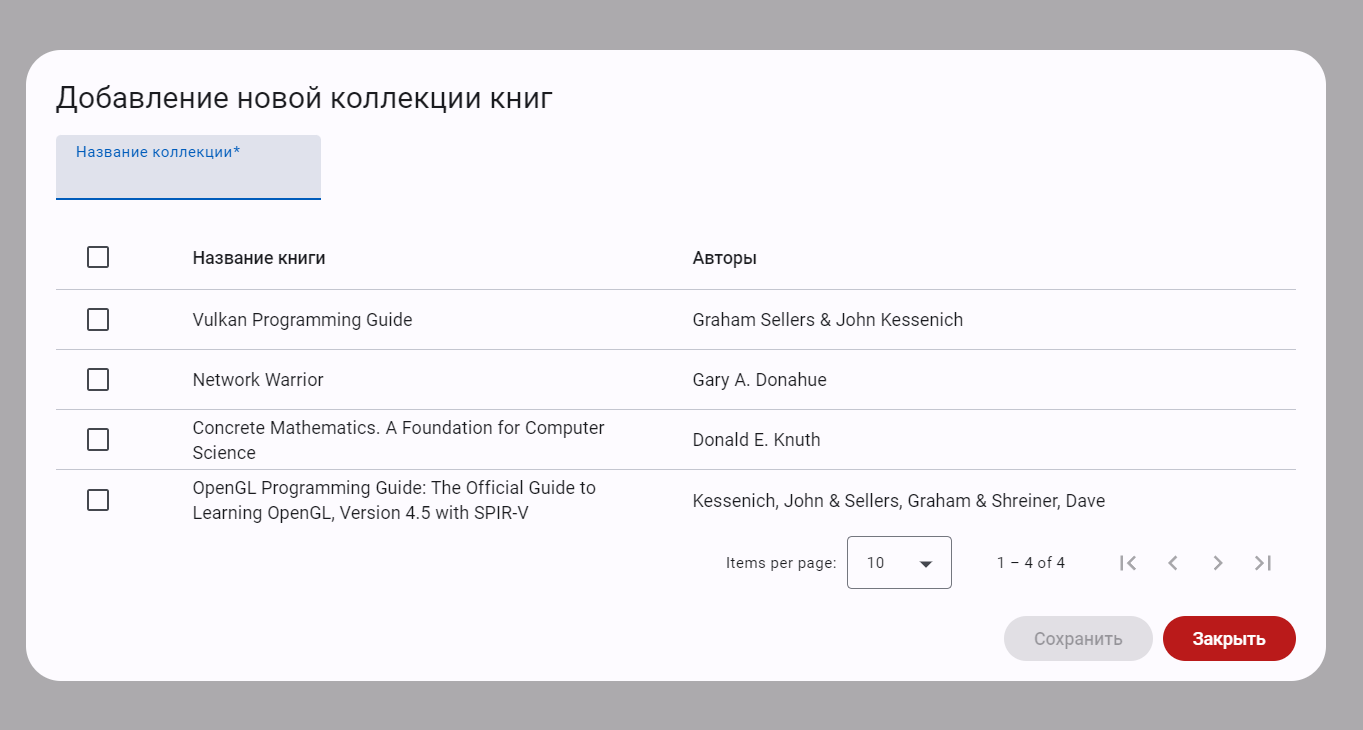


Рисунок 4.34 – Диалог создания коллекции книг

После ввода имени коллекции и выбора нужных книг, нужно нажать на кнопку «Сохранить». Имя коллекции должно быть уникальным. Если будет введено название, которые уже есть в списке, то поле окраситься в красный цвет (рисунок 4.35).

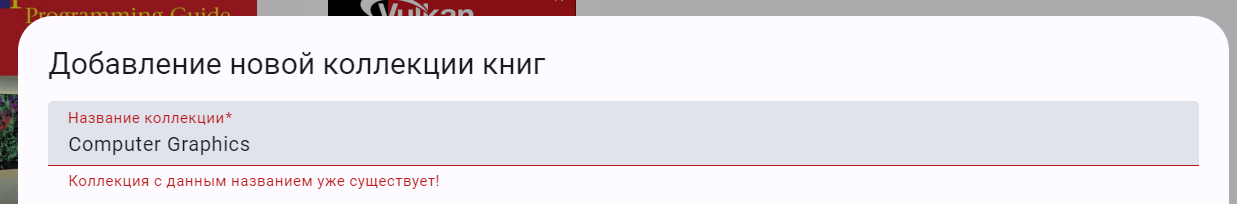


Рисунок 4.35 – Вывод ошибки об уникальности

# 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 их написание 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 сложной части системы. В данном случае тестируется web API сервера.

Алгоритм тестирования определенной функции системы имеет следующий вид:

* запустить сервер с конфигурацией для тестирования (осуществляется программным способом в коде для тестирования);
* авторизоваться в системе, если это необходимо;
* сформировать определенный HTTP-запрос;
* отправить запрос в тестируемую систему;
* получить ответ от системы;
* если ответ системы соответствует ожидаемому результату, например HTTP-ответ с кодом ошибки 400 Bad Request (плохой запрос), то тест считать 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

Эта команда сначала скомпилирует весь проект, и затем запустит процесс тестирования.

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».

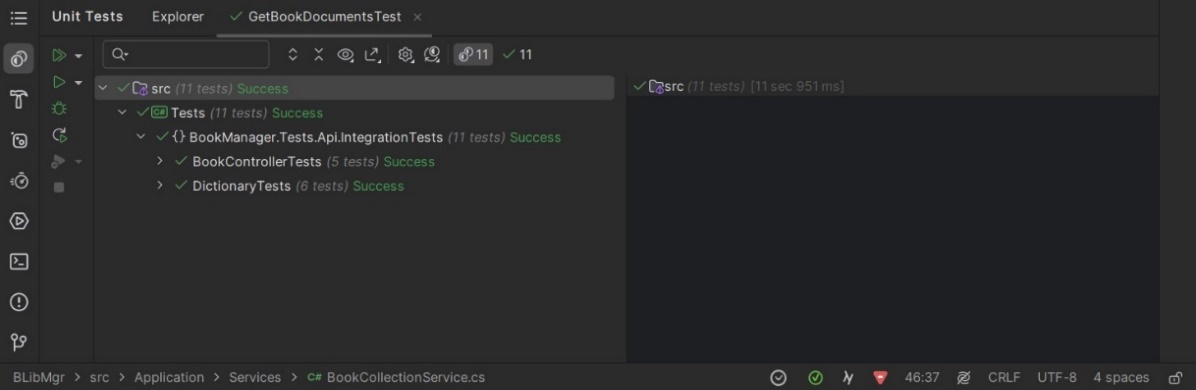


Рисунок 5.1 – Панель Unit Testing

Были написаны 4 класса для тестирования, которые в сумме содержат 19 методов. Перед тестами создаётся база данных для тестов. После всех тестов база данных удаляется. Список классов для тестирования:

* класс «BookTests»;
* класс «UserTests»;
* класс «DictionaryTests».

В классе «BookTests» содержатся тесты получения списка книг, добавление файла книги, изменения данных о книге, загрузка, удаления книги из базы данных. Всего тестов насчитывается 5 в этом классе.

В класс «UserTests» входят тесты, связанные с получения списка имен пользователей, созданием пользователя, изменением ПИН-кода, проверка на то, что один пользователь не может изменить ПИН-код другого пользователя, удаление другого пользователя.

Класс «DictionaryTests» содержит тесты, выведения списка сохраненных слов, поиска, добавления, изменения и удаления.

## 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

Серверная часть приложения была протестирована с помощью автоматизированных тестов через встраиваемую среду разработки. Все тесты были пройдены успешно (рисунок 5.2).

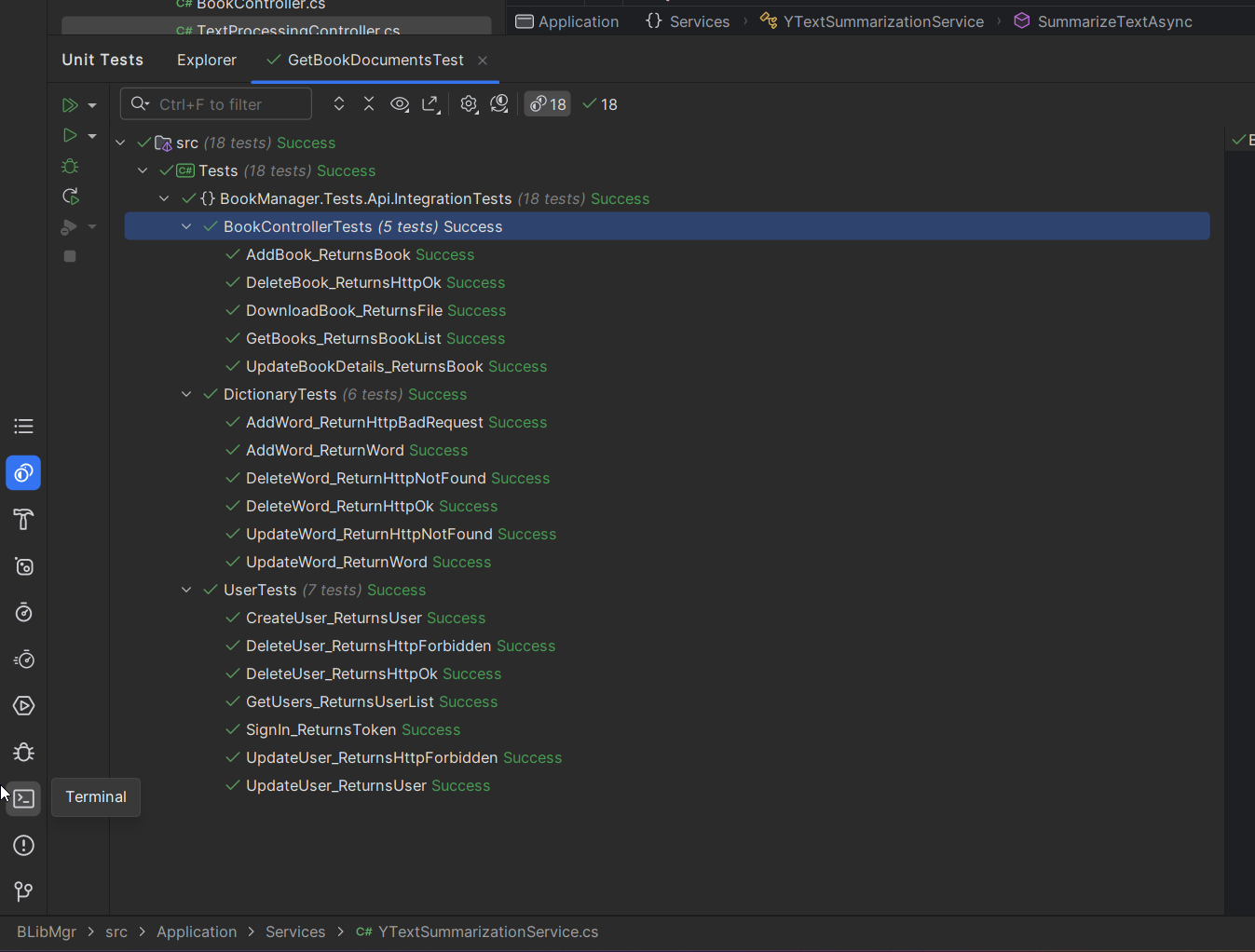


Рисунок 5.2 – Результаты тестирования серверной части

В клиентской части приложения были протестированы следующие функции:

* добавление электронной книги без ввода, связанной с ней данных;
* добавление электронной книги с вводом данных;
* переключение режим отображения списка документов;
* сортировка списка документов;
* изменение информации об электронной книге;
* удаление записи электронной книги;
* создание пользователя;
* открытие электронной книги;
* навигация по документу;
* масштабирование документа;
* отображение контекстного меню после выделения текста;
* выделение и копирование текста из документа;
* выделение и перевод текста из документа;
* выделение слова и нахождение определения;
* выделение и обобщение текста;
* подсчёт времени открытия книги;
* подсчёт времени, проведенного с книгой;
* создание пустой коллекции книг;
* создание коллекции с несколькими книгами;
* изменение состава коллекции книг;
* попытка создания коллекции с существующим уже названием;
* попытка изменения названия коллекции с уже существующим названием в списке коллекции книг;
* удаление коллекции книг.

В ходе тестирования были детально разобраны подсистема работы со списком электронных книг, интерфейс отображения списка книг и коллекции книг, просмотра документа, корректное отображение документа. Таким образом система корректно работает и соответствует требованиям технического задания.

# 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, также 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 материальные ресурсы, разработку системы, оплату труда, себестоимость проекта, расчёт рентабельности и плановой прибыли, определение экономической эффективности.

Также в данном проекте почти прошли весь жизненный цикл разработки программного обеспечения, такие как анализ предметной области проекта, разработка технического задания, проектирование архитектуры информационной системы, разработка программной системы и его тестирования, написание руководства пользователя, выпуск программного продукта.

Также в ходе данной работы были освоены современные инструменты разработки, исследованы возможности интеграции искусственного интеллекта 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. Роберт М. Чистая архитектура. Искусство разработки программного обеспечения / СПб.: Питер. – 2018 г. – 352 с.
10. Роберт М. Чистый код. Создание, анализ и рефакторинг. / СПб.: Питер. – 2022 г. – 872 с.
11. Теория и практика машинного обучения : учебное пособие / В. В. Воронина, А. В. Михеев, Н. Г. Ярушкина, К. В. Святов. –Ульяновск: УлГТУ, 2017. – 290 с.
12. A tour of C# language: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>, свободный.
13. Ajay D. Kshemklayni. Distributed Computing: Principles, Algorithms, and Systems / Ajay D. Kshemklayni, Mukesh Singhal. – Cambridge University Press, 2008.
14. Artificial Intelligence: A Modern Approach, 4th US ed. / Stuart J. Russell., Peter Norvig. – Prentice Hall, 2020.
15. Dependency Injection Principles, Practices, and Patterns / Mark S., Steven van Deursan. – Manning Publications, 2019.
16. Tannenbaum A.S. Distributed Systems / Marteen Van Steen, Tannenbaum A.S. – Pearson Education Inc., 2023.

# ПРИЛОЖЕНИЕ А

Файл AuthenticationController.cs:

﻿using BookManager.Application.Common;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces;

using Microsoft.AspNetCore.Mvc;

using SameSiteMode = Microsoft.AspNetCore.Http.SameSiteMode;

namespace BookManager.Api.Controllers;

[ApiController]

[Route("auth")]

public class AuthenticationController(

IAuthenticationService service)

: ControllerBase

{

[HttpPost]

[Route("sign-in")]

public async Task<IActionResult> SignIn([FromBody] AuthenticationRequestDto request)

{

var authenticationResponse = await service.SignIn(request);

if (authenticationResponse.RefreshToken != null)

{

Response.Cookies.Append(Constants.RefreshTokenCookieKey, authenticationResponse.RefreshToken, new CookieOptions

{

HttpOnly = true,

});

}

return authenticationResponse.Status switch

{

AuthenticationStatus.Failed => Unauthorized(),

AuthenticationStatus.Success => Ok(authenticationResponse),

\_ => BadRequest()

};

}

[HttpPost]

[Route("refresh-token")]

public async Task<IActionResult> RefreshToken()

{

var refreshToken = Request.Cookies[Constants.RefreshTokenCookieKey];

if (refreshToken == null) return Unauthorized();

var response = await service.RefreshToken(refreshToken);

return response.Status switch

{

AuthenticationStatus.Failed => Unauthorized(),

AuthenticationStatus.Success => Ok(response),

\_ => Unauthorized()

};

}

[HttpPost]

[Route("sign-out")]

public new IActionResult SignOut()

{

Response.Cookies.Delete(Constants.RefreshTokenCookieKey);

return Ok();

}

}

Файл BookCollectionController.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace BookManager.Api.Controllers;

[ApiController]

[Route("book-collections")]

public class BookCollectionController(IBookCollectionService service) : ControllerBase

{

[HttpGet]

[Authorize]

public async Task<IActionResult> GetAll()

{

var allCollections = await service.GetAllAsync();

foreach (var collection in allCollections)

{

UpdateThumbnailUrls(collection);

}

return Ok(allCollections);

}

[HttpPost]

[Authorize]

public async Task<IActionResult> CreateCollection([FromBody] BookCollectionModRequest request)

{

IActionResult result;

try

{

var addedCollection = await service.CreateAsync(request);

UpdateThumbnailUrls(addedCollection);

result = Ok(addedCollection);

}

catch (ArgumentException e)

{

result = BadRequest(e.Message);

}

return result;

}

[HttpPut]

[Authorize]

[Route("{id:guid}")]

public async Task<IActionResult> UpdateCollection(Guid id, [FromBody] BookCollectionModRequest request)

{

IActionResult result;

try

{

var addedCollection = await service.UpdateAsync(id, request);

UpdateThumbnailUrls(addedCollection);

result = Ok(addedCollection);

}

catch (ArgumentException e)

{

result = BadRequest(e.Message);

}

catch (EntityNotFoundException)

{

result = NotFound();

}

return result;

}

[HttpDelete]

[Authorize]

[Route("{id:guid}")]

public async Task<IActionResult> DeleteAsync(Guid id)

{

try

{

await service.DeleteAsync(id);

}

catch (EntityNotFoundException)

{

return NotFound();

}

return Ok();

}

private void UpdateThumbnailUrls(BookCollectionDto dto)

{

if (dto.Books == null) return;

foreach (var book in dto.Books)

{

book.DocumentDetails.ThumbnailUrl = GetImageUrl(book.DocumentDetails.Id);

}

}

private string GetImageUrl(Guid id)

{

return $"{Request.Scheme}://{Request.Host}{Request.PathBase}/books/cover/{id}";

}

}

Файл BookController.cs:

using BookManager.Application.Common.DTOs;

using BookManager.Domain.Entities;

using BrunoZell.ModelBinding;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

namespace BookManager.Api.Controllers;

[ApiController]

[Route("books")]

public class BookController(

IBookService service,

ISearchService searchService,

UserManager<User> userManager)

: ControllerBase

{

[HttpGet]

[Authorize]

public async Task<PageDto<BookDto>> GetBooks(

[FromQuery] int pageNumber,

[FromQuery] int pageSize,

[FromQuery] string? sortBy = null,

[FromQuery] SortOrder sortOrder = SortOrder.Asc

)

{

var user = await userManager.GetUserAsync(HttpContext.User);

var page = await service.GetPageAsync(new PageRequestDto(pageNumber, pageSize, sortBy, sortOrder), null, user);

foreach (var item in page.Items)

{

item.DocumentDetails.ThumbnailUrl = GetImageUrl(item.DocumentDetails.Id);

}

return page;

}

[HttpGet]

[Authorize]

[Route("download/{id:guid}")]

public async Task<IActionResult> DownloadBook(Guid id)

{

var user = await userManager.GetUserAsync(HttpContext.User);

if (user == null) return BadRequest();

var bookDto = await service.GetByIdAsync(id);

if (bookDto == null) return NotFound();

var stream = await service.GetBookFileStreamAsync(id, user);

return File(stream, bookDto.GetContentType(), true);

}

[HttpGet]

[Route("cover/{id:guid}")]

public async Task<IActionResult> GetBookCover(Guid id)

{

var imageStream = await service.GetBookCoverImageFileStream(id);

if (imageStream == null) return Ok(null);

return File(imageStream, "image/jpeg");

}

[HttpGet]

[Authorize]

[Route("{id:guid}")]

public async Task<IActionResult> GetBookById(Guid id)

{

var user = await userManager.GetUserAsync(HttpContext.User);

var bookDto = await service.GetByIdAsync(id, user?.Id);

if (bookDto == null)

return NotFound();

return Ok(bookDto);

}

[HttpPost]

[Route("search")]

public async Task<PageDto<BookDto>> Search([FromBody] SearchRequestDto request)

{

var user = await userManager.GetUserAsync(HttpContext.User);

var page = await searchService.SearchByBookDetailsAsync(request, user);

foreach (var item in page.Items)

{

item.DocumentDetails.ThumbnailUrl = GetImageUrl(item.DocumentDetails.Id);

}

return page;

}

[HttpPost]

[Route("full-text-search")]

public Task<PageDto<BookTextDto>> Search([FromBody] TextSearchRequestDto request)

{

return searchService.SearchByBookTexts(request);

}

[HttpPost]

[Authorize]

public async Task<BookDto> AddBook([ModelBinder(BinderType = typeof(JsonModelBinder))] BookMetadataDto bookMetadata,

IFormFile file)

{

await using var stream = file.OpenReadStream();

return await service.AddBookAsync(stream, bookMetadata);

}

[HttpPost]

[Authorize]

[Route("{bookId:guid}/update-total-time")]

public async Task<IActionResult> UpdateTotalTime(Guid bookId, [FromBody] TotalTimeUpdateRequestDto updateRequest)

{

var user = await userManager.GetUserAsync(HttpContext.User);

if (user == null) return BadRequest();

await service.UpdateTotalTimeAsync(bookId, user.Id, updateRequest.Seconds);

return Ok();

}

[HttpPost]

[Authorize]

[Route("{id:guid}/last-viewed-page")]

public async Task<IActionResult> UpdateLastViewedPage(Guid id, [FromBody] LastViewedPageUpdateRequest request)

{

var user = await userManager.GetUserAsync(HttpContext.User);

if (user == null) return BadRequest();

await service.UpdateLastViewedPageAsync(request.PageNumber, user.Id, id);

return Ok();

}

[HttpPut]

[Authorize]

[Route("{id:guid}")]

public async Task<BookDto> UpdateBookDetails(Guid id, [FromBody] BookDetailsUpdateDto details)

{

return await service.UpdateBookDetailsAsync(id, details);

}

[HttpDelete]

[Authorize]

[Route("{id:guid}")]

public async Task<IActionResult> DeleteBook(Guid id)

{

await service.DeleteBookAsync(id);

return Ok();

}

private string GetImageUrl(Guid id)

{

return $"{Request.Scheme}://{Request.Host}{Request.PathBase}/books/cover/{id}";

}

Файл TextProcessingController.cs:

﻿using BookManager.Application.Common.DTOs;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace BookManager.Api.Controllers;

[ApiController]

[Route("text-processing")]

public class TextProcessingController(

ITranslationService translationService,

ITextSummarizationService textSummarizationService): ControllerBase

{

[HttpGet]

[Route("list-languages")]

public async Task<IActionResult> ListLanguages()

{

return Ok(await translationService.ListLanguagesAsync());

}

[HttpPost]

[Authorize]

[Route("detect-language")]

public async Task<IActionResult> DetectLanguage([FromBody] DetectLanguageRequestDto request)

{

return Ok(await translationService.DetectLanguageAsync(request));

}

[HttpPost]

[Authorize]

[Route("translate")]

public async Task<IActionResult> Translate([FromBody] TranslationRequestDto request)

{

return Ok(await translationService.TranslateAsync(request));

}

[HttpPost]

[Authorize]

[Route("summarize-text")]

public async Task<IActionResult> SummarizeText([FromBody] TextSummarizationRequestDto request)

{

return Ok(await textSummarizationService.SummarizeTextAsync(request));

}

}

Файл UserController.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using BookManager.Domain.Entities;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using ArgumentException = System.ArgumentException;

namespace BookManager.Api.Controllers;

[ApiController]

[Route("users")]

public sealed class UserController(

IUserService service,

UserManager<User> userManager

) : ControllerBase

{

[HttpGet]

public IAsyncEnumerable<UserDto> GetAllUsers()

{

return service.GetAllUsers();

}

[HttpPost]

public async Task<IActionResult> CreateUser([FromBody] UserAddRequest request)

{

IActionResult actionResult;

try

{

var createdUser = await service.CreateUser(request);

actionResult = Ok(createdUser);

}

catch (ArgumentException e)

{

actionResult = BadRequest(e.Message);

}

return actionResult;

}

[HttpPut]

[Authorize]

[Route("{id:guid}")]

public async Task<IActionResult> UpdateUser(Guid id, [FromBody] UserAddRequest request)

{

var user = await userManager.GetUserAsync(HttpContext.User);

if (user == null || user.Id != id)

{

return Forbid();

}

IActionResult actionResult;

try

{

var updatedUser = await service.UpdateUser(id, request);

actionResult = Ok(updatedUser);

}

catch (ArgumentException e)

{

actionResult = BadRequest(e.Message);

}

catch (EntityNotFoundException)

{

actionResult = NotFound();

}

return actionResult;

}

[HttpDelete]

[Authorize]

[Route("{id:guid}")]

public async Task<IActionResult> Delete(Guid id)

{

var user = await userManager.GetUserAsync(HttpContext.User);

if (user == null || user.Id != id)

{

return Forbid();

}

try

{

await service.DeleteUser(id);

}

catch (EntityNotFoundException)

{

return NotFound();

}

return Ok();

}

}

Файл WebSocketController.cs:

﻿using System.Net.WebSockets;

using System.Text;

using System.Text.Json;

using BookManager.Application.Notification;

using Microsoft.AspNetCore.Mvc;

using NodaTime;

using NodaTime.Serialization.SystemTextJson;

namespace BookManager.Api.Controllers;

[ApiController]

public class WebSocketController(INotificationService notificationService) : ControllerBase

{

private readonly JsonSerializerOptions \_serializerOptions =

new JsonSerializerOptions(JsonSerializerDefaults.Web).ConfigureForNodaTime(DateTimeZoneProviders.Tzdb);

[Route("/ws")]

public async Task Ws()

{

if (HttpContext.WebSockets.IsWebSocketRequest)

{

using var webSocket = await HttpContext.WebSockets.AcceptWebSocketAsync();

await Handle(webSocket);

}

else

{

HttpContext.Response.StatusCode = StatusCodes.Status400BadRequest;

}

}

private async Task Handle(WebSocket webSocket)

{

var channel = await notificationService.SubscribeAsync(CancellationToken.None);

while (await channel.Reader.WaitToReadAsync())

{

if (webSocket.State != WebSocketState.Open)

{

break;

}

while (channel.Reader.TryRead(out var notification))

{

var jsonString = JsonSerializer.Serialize(notification, \_serializerOptions);

await webSocket.SendAsync(

new ArraySegment<byte>(Encoding.UTF8.GetBytes(jsonString)),

WebSocketMessageType.Text,

WebSocketMessageFlags.EndOfMessage,

CancellationToken.None

);

}

}

await notificationService.UnsubscribeAsync(channel);

}

}

Файл WordDictionaryController.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace BookManager.Api.Controllers;

[ApiController]

[Route("word-dictionary")]

public class WordDictionaryController(IWordDictionaryService service) : ControllerBase

{

[HttpGet]

[Route("list-third-party-providers")]

public IActionResult ListThirdPartyDictionaryProviderNames()

{

return Ok(service.GetThirdPartyProviderNames());

}

[HttpGet]

[Authorize]

[Route("{id}")]

public async Task<IActionResult> FindAsync(string id, [FromQuery] string? providerName)

{

IActionResult? result;

try

{

var words = await service.FindAsync(id);

result = Ok(words);

}

catch (ArgumentException)

{

result = BadRequest();

}

return result;

}

[HttpGet]

[Authorize]

[Route("third-party-dictionary/{id}")]

public async Task<IActionResult> FindInThirdPartyDictionaryAsync(string id, [FromQuery] string providerName)

{

IActionResult? result;

try

{

var words = await service.FindInExtDictAsync(id, providerName);

result = Ok(words);

}

catch (NotAvailableException e)

{

result = StatusCode(StatusCodes.Status503ServiceUnavailable, e.Message);

}

catch (ArgumentException)

{

result = BadRequest();

}

return result;

}

[HttpPost]

[Authorize]

public async Task<IActionResult> AddWordAsync([FromBody] WordDto word)

{

IActionResult result;

try

{

var addedWord = await service.AddWordAsync(word);

result = Ok(addedWord);

}

catch (ArgumentException e)

{

result = BadRequest(e.Message);

}

return result;

}

[HttpPut]

[Authorize]

[Route("{id}")]

public async Task<IActionResult> UpdateWordAsync(string id, [FromBody] WordDto word)

{

IActionResult result;

try

{

var updatedWord = await service.UpdateWordAsync(id, word);

result = Ok(updatedWord);

}

catch (ArgumentException e)

{

result = BadRequest(e.Message);

}

catch (EntityNotFoundException)

{

result = NotFound();

}

return result;

}

[HttpDelete]

[Authorize]

[Route("{id}")]

public async Task<IActionResult> DeleteWordAsync(string id)

{

IActionResult result;

try

{

await service.DeleteWordAsync(id);

result = Ok();

}

catch (EntityNotFoundException)

{

result = NotFound();

}

return result;

}

}

Файл Security.cs:

﻿using System.Text;

using Microsoft.IdentityModel.Tokens;

namespace BookManager.Api.Extensions;

public static class SecurityExtensions

{

public static IServiceCollection AddTokenBasedSecurity(this IServiceCollection services, IConfiguration configuration)

{

var jwtTokenOptionsSection = configuration.GetSection(JwtTokenOptions.Jwt);

var jwtTokenOptions = jwtTokenOptionsSection.Get<JwtTokenOptions>() ?? new JwtTokenOptions();

var jsonWebKey = new JsonWebKey(File.ReadAllText("./jwk.json", Encoding.UTF8));

var tokenValidationParameters = new TokenValidationParameters

{

ValidIssuer = jwtTokenOptions.Issuer,

ValidAudience = jwtTokenOptions.Audience,

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

IssuerSigningKey = jsonWebKey,

};

services.AddSingleton<SecurityKey, JsonWebKey>(\_ => jsonWebKey);

services.AddSingleton<TokenValidationParameters>(\_ => tokenValidationParameters);

services.Configure<JwtTokenOptions>(jwtTokenOptionsSection);

services.AddAuthentication().AddJwtBearer(options =>

{

options.TokenValidationParameters = tokenValidationParameters;

});

return services;

}

}

Файл YandexCloud.cs:

﻿using System.Security.Cryptography;

using System.Text.Json;

using System.Text.Json.Serialization;

using Microsoft.IdentityModel.JsonWebTokens;

using Microsoft.IdentityModel.Tokens;

using Yandex.Cloud;

using Yandex.Cloud.Credentials;

namespace BookManager.Api.Extensions;

public class JwtCredentialsProvider(string keyId, string serviceAccountId, string privateKeyFilePath): ICredentialsProvider

{

private readonly HttpClient \_httpClient = new();

private const string TokensUrl = "https://iam.api.cloud.yandex.net/iam/v1/tokens";

public string GetToken()

{

var now = DateTime.UtcNow;

var rsa = RSA.Create();

rsa.ImportFromPem(File.ReadAllText(privateKeyFilePath).ToCharArray());

var securityKey = new RsaSecurityKey(rsa)

{

KeyId = keyId

};

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.RsaSsaPssSha256);

var descriptor = new SecurityTokenDescriptor

{

Issuer = serviceAccountId,

Audience = TokensUrl,

IssuedAt = now,

Expires = now.AddMinutes(60),

SigningCredentials = credentials,

};

var jwtHandler = new JsonWebTokenHandler();

var encodedToken = jwtHandler.CreateToken(descriptor);

if (encodedToken == null) return string.Empty;

var content = new Dictionary<string, object?>()

{

["jwt"] = encodedToken

};

var request = new HttpRequestMessage(HttpMethod.Post, TokensUrl)

{

Content = JsonContent.Create(content)

};

var response = \_httpClient.Send(request);

var data = JsonSerializer.Deserialize<Data>(response.Content.ReadAsByteArrayAsync().Result);

return data?.IamToken ?? string.Empty;

}

private class Data

{

[JsonPropertyName("iamToken")]

public string IamToken { get; init; } = string.Empty;

[JsonPropertyName("expiresAt")]

public string ExpiresAt { get; init; } = string.Empty;

}

}

public static class YandexCloud

{

public static IServiceCollection AddYandexCloudSdk(this IServiceCollection services, IConfiguration configuration)

{

var ycOptions = configuration.GetSection(YandexCloudOptions.YandexCloud)

.Get<YandexCloudOptions>();

if (ycOptions == null) return services;

services.Configure<YandexCloudOptions>(configuration.GetSection(YandexCloudOptions.YandexCloud));

var credentialsProvider =

new JwtCredentialsProvider(ycOptions.KeyId, ycOptions.ServiceAccountId, ycOptions.PrivateKeyFilePath);

var sdk = new Sdk(credentialsProvider);

services.AddScoped<ICredentialsProvider>(\_ => credentialsProvider);

services.AddScoped<Sdk>(\_ => sdk);

return services;

}

}

Файл NotificationHub.cs:

﻿using Microsoft.AspNetCore.SignalR;

namespace BookManager.Api.Hubs;

public class NotificationHub : Hub;

Файл Program.cs:

using BookManager.Api.Extensions;

using BookManager.Api.Hubs;

using BookManager.Api.Services;

using BookManager.Application;

using BookManager.Application.Common;

using BookManager.Application.Notification;

var builder = WebApplication.CreateBuilder();

builder.Services

.Configure<FileStorageOptions>(builder.Configuration.GetSection(FileStorageOptions.FileStorage))

.ConfigureDataPersistence(builder.Configuration)

.AddApplicationServices(builder.Configuration);

builder.Services.AddControllers();

builder.Services.AddSignalR();

builder.Services.AddEndpointsApiExplorer()

.AddSwaggerGen();

builder.Services.AddCors(options =>

{

options.AddDefaultPolicy(policy => policy

.WithOrigins(

"https://localhost:4200",

"http://localhost:4200"

)

.AllowAnyMethod()

.AllowAnyHeader()

.AllowCredentials()

);

});

builder.Services.AddYandexCloudSdk(builder.Configuration);

builder.Services.AddTokenBasedSecurity(builder.Configuration);

builder.Services.AddSingleton<INotificationService, PushNotificationService>();

builder.WebHost.ConfigureKestrel(options =>

{

options.Limits.MaxRequestBodySize = Constants.Default.MaxFileSize;

});

var app = builder.Build();

app.DbMigrate();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseCors();

app.UseAuthentication();

app.UseAuthorization();

app.UseWebSockets();

app.UseDefaultFiles();

app.UseStaticFiles();

app.MapControllers();

app.MapHub<NotificationHub>("/notification");

app.Run();

// For tests

public abstract partial class Program;

Файл PushNotificationService.cs:

﻿using System.Text.Json;

using System.Threading.Channels;

using BookManager.Api.Hubs;

using BookManager.Application.Notification;

using Microsoft.AspNetCore.SignalR;

using NodaTime;

using NodaTime.Serialization.SystemTextJson;

namespace BookManager.Api.Services;

public class PushNotificationService(IHubContext<NotificationHub> hubContext) : INotificationService

{

private readonly JsonSerializerOptions \_serializerOptions =

new JsonSerializerOptions(JsonSerializerDefaults.Web).ConfigureForNodaTime(DateTimeZoneProviders.Tzdb);

public ValueTask<Channel<Notification>> SubscribeAsync(CancellationToken cancellationToken)

{

throw new NotImplementedException();

}

public ValueTask UnsubscribeAsync(Channel<Notification> channel)

{

throw new NotImplementedException();

}

public async ValueTask SendAsync(Notification notification, CancellationToken cancellationToken)

{

var jsonString = JsonSerializer.Serialize(notification, \_serializerOptions);

await hubContext.Clients.All.SendAsync("book-indexing", jsonString, cancellationToken);

}

}

Файл DictionaryOptions.cs:

﻿namespace BookManager.Application.Common.Config;

public sealed class DictionaryOptions

{

public const string Dictionary = "Dictionary";

public required string RapidApiKey { get; init; }

public string RapidApiHost { get; init; } = Constants.Default.Dictionary.RapidApiHost;

public string BaseUrl { get; init; } = Constants.Default.Dictionary.BaseUrl;

public string DefinitionUrl { get; init; } = Constants.Default.Dictionary.DefinitionUrl;

public string InfoUrl { get; init; } = Constants.Default.Dictionary.InfoUrl;

}

Файл FileStorageOptions.cs:

﻿namespace BookManager.Application.Common.Config;

public sealed class FileStorageOptions

{

public const string FileStorage = "FileStorage";

public string DirectoryPath { get; init; } = Constants.Default.DirectoryPath;

public long MaxFileSizeInBytes { get; init; } = Constants.Default.MaxFileSize;

}

Файл JwtTokenOptions.cs:

﻿namespace BookManager.Application.Common.Config;

public class JwtTokenOptions

{

public const string Jwt = "Jwt";

public string Issuer { get; init; } = Constants.Default.Issuer;

public string Audience { get; init; } = Constants.Default.Audience;

public uint AccessTokenLifetimeInMinutes { get; init; } = Constants.Default.AccessTokenLifetimeInMinutes;

public uint RefreshTokenLifetimeInMinutes { get; init; } = Constants.Default.RefreshTokenLifetimeInMinutes;

public string Key { get; init; } = string.Empty;

}

Файл MerriamWebsterOptions.cs:

﻿namespace BookManager.Application.Common.Config;

public class MerriamWebsterOptions

{

public const string MerriamWebster = "MerriamWebster";

public string ApiKey { get; init; }

}

Файл YandexCloudOptions.cs:

﻿namespace BookManager.Application.Common.Config;

public class YandexCloudOptions

{

public const string YandexCloud = "YandexCloud";

public string KeyId { get; init; } = string.Empty;

public string ServiceAccountId { get; init; } = string.Empty;

public string PrivateKeyFilePath { get; init; } = string.Empty;

public string ModelSummarizationUri { get; init; } = string.Empty;

public string GptModelUri { get; init; } = string.Empty;

public float ModelSummarizationTemperature { get; init; }

public float GptModelTemperature { get; init; }

}

Файл Constants.cs:

namespace BookManager.Application.Common;

public static class Constants

{

public static class Default

{

public const string DirectoryPath = "./";

public const int IndexingQueueCapacity = 2;

// 256 MiB in Bytes

public const long MaxFileSize = 268\_435\_456L;

// JWT options

public const string Issuer = "https://localhost:7013";

public const string Audience = "http://localhost:4200";

public const uint AccessTokenLifetimeInMinutes = 5;

public const uint RefreshTokenLifetimeInMinutes = 60;

public static class Dictionary

{

public const string RapidApiHost = "wordsapiv1.p.rapidapi.com";

public const string BaseUrl = "https://wordsapiv1.p.rapidapi.com";

public const string DefinitionUrl = "/words/incredible/definitions";

public const string InfoUrl = "/words";

}

}

public const string RefreshTokenCookieKey = "refresh\_token";

public const int ThumbnailPreviewWidth = 600;

public const int ThumbnailPreviewHeight = 800;

}

Файл AuthenticationRequestDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public record AuthenticationRequestDto(string Name, string PinCode);

Файл AuthenticationResponseDto.cs:

﻿using System.Text.Json.Serialization;

using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public class AuthenticationResponseDto

{

public AuthenticationStatus Status { get; set; } = AuthenticationStatus.Success;

public string? AccessToken { get; set; }

[JsonIgnore]

public string? RefreshToken { get; set; }

}

[TranspilationSource]

public enum AuthenticationStatus

{

Failed,

Success

}

Файл BookCollectionDTOs.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record BookCollectionDto

{

public Guid Id { get; init; }

public required string Name { get; init; }

public ICollection<BookDto>? Books { get; init; } = null!;

public BookCollection ToEntity()

{

var entity = new BookCollection

{

Id = Id,

Name = Name

};

if (Books is { Count: > 0 })

{

entity.Books = Books.Select(d => d.ToEntity()).ToList();

}

return entity;

}

}

[TranspilationSource]

public sealed record BookCollectionModRequest(string Name, ICollection<BookDto>? Books);

public static class BookCollectionExtensions

{

public static BookCollectionDto ToDto(this BookCollection bookCollection)

{

return new BookCollectionDto

{

Id = bookCollection.Id,

Name = bookCollection.Name,

Books = bookCollection.Books.Select(b => b.ToDto()).ToList()

};

}

}

Файл BookDetailsUpdateDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public record BookDetailsUpdateDto(

string? Title,

string? Description,

string? Isbn,

string? PublisherName,

IEnumerable<string>? Authors,

IEnumerable<string>? Tags

);

Файл BookDto.cs:

using System.Net.Mime;

using BookManager.Domain.Enums;

using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record BookDto

{

public required Details DocumentDetails { get; init; }

public required BookFileMetadata FileMetadata { get; init; }

public UserStats? Stats { get; init; }

public Book ToEntity()

{

return new Book

{

Id = DocumentDetails.Id,

Description = DocumentDetails.Description,

Isbn = DocumentDetails.Isbn,

PublisherName = DocumentDetails.PublisherName,

Title = DocumentDetails.Title,

Filename = string.Empty,

FileType = FileMetadata.Type,

FileSize = FileMetadata.Size,

ThumbnailFilename = string.Empty,

Authors = DocumentDetails.Authors,

Tags = DocumentDetails.Tags

};

}

public string GetContentType()

{

return FileMetadata.Type switch

{

BookFileType.Pdf => MediaTypeNames.Application.Pdf,

BookFileType.Epub => "application/epub+zip",

\_ => throw new ArgumentOutOfRangeException()

};

}

[TranspilationSource]

public class Details

{

public Guid Id { get; init; }

public string? Title { get; set; }

public string? Isbn { get; set; }

public string? Description { get; set; }

public string? PublisherName { get; set; }

public string? ThumbnailUrl { get; set; }

public int? PageCount { get; set; }

public string[]? Authors { get; set; }

public string[]? Tags { get; set; }

}

[TranspilationSource]

public class UserStats

{

public long TotalReadingTime { get; set; }

public DateTimeOffset? RecentAccessTime { get; set; }

public int? LastViewedPage { get; set; }

}

[TranspilationSource]

public record BookFileMetadata

{

public required BookFileType Type { get; init; }

public required long Size { get; init; }

}

}

public static class BookDocumentEntityExtensions

{

public static BookDto ToDto(this Book entity)

{

var stats = entity.Stats.Any() ? entity.Stats.FirstOrDefault() : null;

return new BookDto

{

DocumentDetails = new BookDto.Details

{

Id = entity.Id,

Description = entity.Description,

Isbn = entity.Isbn,

PublisherName = entity.PublisherName,

Title = entity.Title,

ThumbnailUrl = string.Empty,

PageCount = entity.PageCount,

Authors = entity.Authors,

Tags = entity.Tags,

},

FileMetadata = new BookDto.BookFileMetadata

{

Size = entity.FileSize,

Type = entity.FileType

},

Stats = stats != null ? new BookDto.UserStats

{

TotalReadingTime = stats.TotalReadingTime,

RecentAccessTime = stats.RecentAccess.ToDateTimeOffset(),

LastViewedPage = stats.LastViewedPage

} : null

};

}

}

Файл BookMetadataDto.cs:

﻿using BookManager.Domain.Enums;

using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record BookMetadataDto

{

public required string Filename { get; init; }

public required long FileSizeInBytes { get; init; }

public required BookFileType FileType { get; init; }

public string? Title { get; init; }

public string? Isbn { get; init; }

public string? Description { get; init; }

public string? PublisherName { get; init; }

public IEnumerable<string>? Authors { get; init; }

public IEnumerable<string>? Tags { get; init; }

}

Файл BookTextDto.cs:

﻿namespace BookManager.Application.Common.DTOs;

public record BookTextDto

{

public required Guid BookDocumentId { get; init; }

public required string Text { get; init; }

public int? PageNumber { get; init; }

}

public static class BookDocumentTextEntityExtensions

{

public static BookTextDto ToDto(this BookText entity)

{

return new BookTextDto

{

BookDocumentId = entity.BookDocumentId,

Text = entity.Text,

PageNumber = entity.PageNumber

};

}

}

Файл DetectLanguageDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record DetectLanguageRequestDto(string Text);

[TranspilationSource]

public sealed record DetectLanguageResponseDto(string DetectedLanguageCode);

Файл FileChunkDto.cs:

﻿namespace BookManager.Application.Common.DTOs;

public sealed record FileChunkDto

{

public required string Hash { get; init; }

public required byte[] Data { get; init; }

}

Файл LanguageDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record LanguageDto(string Code, string Name);

Файл LastViewedPageUpdateRequest.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public record LastViewedPageUpdateRequest(int PageNumber);

Файл MerriamWebsterDefDto.cs:

﻿using System.Text.Json.Serialization;

namespace BookManager.Application.Common.DTOs;

// https://dictionaryapi.com/products/json

public sealed record MerriamWebsterDefDto

{

public EntryMetadata Meta { get; init; }

[JsonPropertyName("hom")]

public int Homograph { get; init; }

[JsonPropertyName("fl")]

public string FunctionalLabel { get; init; } = string.Empty;

[JsonPropertyName("vrs")]

public Variant[] Variants { get; init; } = [];

[JsonPropertyName("hwi")]

public HeadwordInformation HeadwordInfo { get; init; }

[JsonPropertyName("def")]

public object[] DefinitionSection { get; init; } = [];

[JsonPropertyName("shortdef")]

public string[] ShortDefinition { get; init; } = [];

public record EntryMetadata(

string Id,

string Uuid,

string Sort,

string Src,

string Section,

string[] Stems,

bool Offensive

);

public record HeadwordInformation

{

[JsonPropertyName("hw")]

public string Headword { get; init; } = string.Empty;

[JsonPropertyName("prs")]

public Pronuciation[]? Pronunciations { get; init; }

}

public record Pronuciation

{

[JsonPropertyName("mw")]

public string WrittenPronunciation { get; init; } = string.Empty;

[JsonPropertyName("sound")]

public AudioInfo Sound { get; init; } = null!;

}

public record AudioInfo

{

[JsonPropertyName("audio")]

public string FileName { get; init; } = string.Empty;

public string Ref { get; init; } = string.Empty;

public string Stat { get; init; } = string.Empty;

}

public record Variant

{

public string Va { get; init; } = string.Empty;

[JsonPropertyName("vl")]

public string? VariantLabel { get; init; }

[JsonPropertyName("prs")]

public Pronuciation[]? Pronunciations { get; init; }

}

}

Файл PageDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public record PageDto<T>

{

public IEnumerable<T> Items { get; set; } = new List<T>();

public int PageSize { get; set; }

public int PageNumber { get; set; }

public int PageCount { get; set; }

public int TotalItemCount { get; set; }

public static int GetNormalizedPageNumber(int pageNumber)

{

return pageNumber <= 0 ? 1 : pageNumber;

}

public static int CountPage(int totalItemCount, int pageSize)

{

var normalizedTotalItemCount = totalItemCount == 0 ? 1 : totalItemCount;

var normalizedPageSize = pageSize == 0 ? 1 : pageSize;

return normalizedTotalItemCount / normalizedPageSize +

(normalizedTotalItemCount % normalizedPageSize > 0 ? 1 : 0);

}

public sealed class Builder

{

private readonly PageDto<T> \_pageDto = new();

private Builder()

{

}

public static Builder Create()

{

return new Builder();

}

public Builder SetItems(IEnumerable<T> items)

{

\_pageDto.Items = items;

return this;

}

public Builder SetPageSize(int pageSize)

{

\_pageDto.PageSize = pageSize;

return this;

}

public Builder SetPageNumber(int pageNumber)

{

\_pageDto.PageNumber = pageNumber;

return this;

}

public Builder SetPageCount(int pageCount)

{

\_pageDto.PageCount = pageCount;

return this;

}

public Builder SetTotalItemCount(int totalItemCount)

{

\_pageDto.TotalItemCount = totalItemCount;

return this;

}

public PageDto<T> Build()

{

return \_pageDto;

}

}

}

Файл PageRequestDto.cs:

﻿using System.Linq.Expressions;

using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record PageRequestDto(

int PageNumber,

int PageSize,

string? SortBy = null,

SortOrder SortOrder = SortOrder.Asc);

[TranspilationSource]

public enum SortOrder

{

Asc,

Desc

}

Файл RawImageDto.cs:

﻿namespace BookManager.Application.Common.DTOs;

public record RawImageDto(byte[] Data, int Width, int Height);

Файл SearchRequestDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed class SearchRequestDto

{

public int PageSize { get; set; }

public int PageNumber { get; set; }

public string? SortProperty { get; init; }

public SortOrder? SortOrder { get; init; }

public string? Title { get; set; }

public string? Description { get; set; }

public string? Isbn { get; set; }

public string? PublisherName { get; set; }

public string[]? Authors { get; set; }

}

Файл TextSearchRequestDto.cs:

﻿namespace BookManager.Application.Common.DTOs;

public record TextSearchRequestDto

{

public int PageSize { get; init; }

public int PageNumber { get; init; }

public string Pattern { get; init; } = string.Empty;

}

Файл TextSummarizationDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public record TextSummarizationRequestDto(string Text);

[TranspilationSource]

public record TextSummarizationResponseDto(string SummarizedText);

Файл TotalTimeUpdateRequestDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public class TotalTimeUpdateRequestDto

{

public long Seconds { get; init; }

}

Файл TranslationDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record TranslationRequestDto

{

public string? SourceLanguage { get; init; }

public string TargetLanguage { get; init; } = string.Empty;

public string SourceText { get; init; } = string.Empty;

}

[TranspilationSource]

public sealed record TranslationResponseDto

{

public string? DetectedSourceLanguage { get; init; }

public string TargetLanguage { get; init; } = string.Empty;

public string TranslatedText { get; init; } = string.Empty;

}

Файл UserAddRequest.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public record UserAddRequest

{

public required string Name { get; init; }

public required string PinCode { get; init; }

}

Файл UserDto.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record UserDto(Guid Id, string Name);

public static class UserEntityExtensions

{

public static UserDto ToDto(this User user)

{

return new UserDto(user.Id, user.UserName ?? "");

}

}

Файл WordDictionaryDTOs.cs:

﻿using Tapper;

namespace BookManager.Application.Common.DTOs;

[TranspilationSource]

public sealed record WordDefinitionDto(string PartOfSpeech, string SubjectName, string Definition)

{

public DictionaryWordDefinition ToEntity()

{

return new DictionaryWordDefinition

{

SubjectName = SubjectName,

PartOfSpeech = PartOfSpeech,

Definition = Definition

};

}

}

[TranspilationSource]

public sealed record WordAlias(string Alias)

{

public DictionaryWordAlias ToEntity()

{

return new DictionaryWordAlias

{

Alias = Alias

};

}

}

[TranspilationSource]

public sealed record WordDto(

string Word,

string? Transcription,

string? LanguageCode,

ICollection<WordAlias> Aliases,

ICollection<WordDefinitionDto> Definitions)

{

public DictionaryWord ToEntity()

{

var word = new DictionaryWord

{

Word = Word,

Transcription = Transcription,

LanguageCode = LanguageCode

};

foreach (var definition in Definitions.Select(def => def.ToEntity()))

{

word.Definitions.Add(definition);

}

return word;

}

}

public static class DictionaryWordExtension

{

public static WordDto ToDto(this DictionaryWord word)

{

return new WordDto(

word.Word,

word.Transcription,

word.LanguageCode,

word.Aliases.Select(a => a.ToDto()).ToList(),

word.Definitions.Select(wordDef => wordDef.ToDto()).ToList()

);

}

public static WordDefinitionDto ToDto(this DictionaryWordDefinition definition)

{

return new WordDefinitionDto(definition.PartOfSpeech, definition.SubjectName, definition.Definition);

}

public static WordAlias ToDto(this DictionaryWordAlias alias)

{

return new WordAlias(alias.Alias);

}

}

public record WordsApiWordDefinitionResponseDto

{

public string Word { get; init; } = null!;

public IEnumerable<WordDefinition> Definitions { get; } = new List<WordDefinition>();

public record WordDefinition(string Definition, string PartOfSpeech);

}

Файл EntityNotFoundException.cs:

﻿namespace BookManager.Application.Common.Exceptions;

public class EntityNotFoundException : Exception;

Файл FileAlreadyExistsException.cs:

namespace BookManager.Application.Common.Exceptions;

public class FileAlreadyExistsException : Exception

{

public FileAlreadyExistsException() : base("The file already exists!")

{

}

}

Файл InvalidFileHashException.cs:

namespace BookManager.Application.Common.Exceptions;

public class InvalidFileHashException : Exception

{

public InvalidFileHashException() : base("The file integrity hash is not valid!")

{

}

}

Файл MaxFileSizeException.cs:

namespace BookManager.Application.Common.Exceptions;

public class MaxFileSizeException : Exception

{

public MaxFileSizeException() : base("The maximum file size has been exceeded.")

{

}

}

Файл NotAvailableException.cs:

﻿namespace BookManager.Application.Common.Exceptions;

public class NotAvailableException(string message): Exception($"NotAvailableException: {message}");

Файл UserAccountExceptions.cs:

﻿namespace BookManager.Application.Common.Exceptions;

public class UserCreationException(string message): Exception(message);

public class UserUpdateException(string message): Exception(message);

Файл IFileStorage.cs:

namespace BookManager.Application.Common.Interfaces;

public interface IFileStorage

{

Task<FileInfo> SaveFileAsync(string filename, Stream stream);

FileStream GetFileStream(string filename);

IAsyncEnumerable<byte[]> ReadFileAsync(string filename);

string GetFileHash(string filename);

void DeleteFile(string filename);

}

Файл IAppDbContext.cs:

namespace BookManager.Application.Common.Interfaces.Persistence;

public interface IAppDbContext

{

DbSet<BookCollection> BookCollections { get; }

DbSet<Book> Books { get; }

DbSet<BookText> BookTexts { get; }

DbSet<User> Users { get; }

DbSet<BookUserStats> BookUserStatsSet { get; }

DbSet<DictionaryWord> DictionaryWords { get; }

Task<int> SaveChangesAsync(CancellationToken cancellationToken = default);

}

Файл IAuthenticationService.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces;

public interface IAuthenticationService

{

public Task<AuthenticationResponseDto> SignIn(AuthenticationRequestDto request);

public Task<AuthenticationResponseDto> RefreshToken(string refreshTokenString);

}

Файл IBookCollectionService.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface IBookCollectionService

{

public Task<IEnumerable<BookCollectionDto>> GetAllAsync();

public Task<BookCollectionDto> CreateAsync(BookCollectionModRequest request);

public Task<BookCollectionDto> UpdateAsync(Guid id, BookCollectionModRequest request);

public Task DeleteAsync(Guid id);

}

Файл IBookFileHandler.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Domain.Enums;

namespace BookManager.Application.Common.Interfaces.Services;

public interface IBookFileHandler

{

public BookFileType FileType { get; }

public int? CountNumberOfPages(Stream bookStream);

public IEnumerable<string> GetAuthorList(Stream bookStream);

public RawImageDto? GetPreviewImage(Stream bookStream);

public Task<Stream> GetJpegImageAsync(RawImageDto rawImage);

public string? GetBookTitle(Stream bookStream);

public IAsyncEnumerable<BookText> StreamBookTexts(Guid bookId, Stream stream);

}

Файл IBookService.cs:

using System.Linq.Expressions;

using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface IBookService

{

public Task<PageDto<BookDto>> GetPageAsync(

PageRequestDto request,

Expression<Func<Book, bool>>? predicate = null,

User? user = null

);

public Task<BookDto?> GetByIdAsync(Guid bookId, Guid? userId = null);

public Task<BookDto> AddBookAsync(Stream fileStream, BookMetadataDto bookMetadata);

public Task<BookDto> UpdateBookDetailsAsync(Guid id, BookDetailsUpdateDto details);

public Task UpdateTotalTimeAsync(Guid bookId, Guid userId, long seconds);

public Task<FileStream> GetBookFileStreamAsync(Guid id, User user);

public Task<FileStream?> GetBookCoverImageFileStream(Guid bookId);

public Task DeleteBookAsync(Guid id);

public Task UpdateLastViewedPageAsync(int page, Guid userId, Guid bookId);

}

Файл IIndexingService.cs:

﻿namespace BookManager.Application.Common.Interfaces.Services;

public interface IIndexingService

{

Task IndexDocumentAsync(Guid bookId, CancellationToken cancellationToken);

Task<int> DeleteDocumentTextsAsync(Guid bookDocumentId, CancellationToken cancellationToken);

}

Файл ISearchService.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface ISearchService

{

public Task<PageDto<BookDto>> SearchByBookDetailsAsync(SearchRequestDto searchRequest, User? user = null);

public Task<PageDto<BookTextDto>> SearchByBookTexts(TextSearchRequestDto request);

}

Файл ITextSummarizationService.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface ITextSummarizationService

{

public Task<TextSummarizationResponseDto> SummarizeTextAsync(TextSummarizationRequestDto request);

}

Файл IThirdPartyDictionaryProvider.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface IThirdPartyDictionaryProvider

{

public string ProviderName { get; }

public Task<IEnumerable<WordDto>> GetDefinitionAsync(string word);

}

Файл ITranslationService.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface ITranslationService

{

public Task<IEnumerable<LanguageDto>> ListLanguagesAsync();

public Task<DetectLanguageResponseDto> DetectLanguageAsync(DetectLanguageRequestDto request);

public Task<TranslationResponseDto> TranslateAsync(TranslationRequestDto request);

}

Файл IUserService.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface IUserService

{

public IAsyncEnumerable<UserDto> GetAllUsers();

public Task<UserDto> CreateUser(UserAddRequest request);

public Task<UserDto> UpdateUser(Guid id, UserAddRequest request);

public Task DeleteUser(Guid id);

}

Файл IWordDictionaryService.cs:

﻿using BookManager.Application.Common.DTOs;

namespace BookManager.Application.Common.Interfaces.Services;

public interface IWordDictionaryService

{

public IEnumerable<string> GetThirdPartyProviderNames();

public Task<IEnumerable<WordDto>> FindAsync(string word);

public Task<IEnumerable<WordDto>> FindInExtDictAsync(string word, string thirdPartyProviderName);

public Task<WordDto> AddWordAsync(WordDto word);

public Task<WordDto> UpdateWordAsync(string wordId, WordDto word);

public Task DeleteWordAsync(string word);

}

Файл DependencyInjection.cs:

﻿using BookManager.Application.Common.Config;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces;

using BookManager.Application.Common.Interfaces.Services;

using BookManager.Application.Indexing;

using BookManager.Application.Notification;

using BookManager.Application.Persistence;

using BookManager.Application.Persistence.FileSystem;

using BookManager.Application.Services;

using BookManager.Application.Validators;

using FluentValidation;

using Microsoft.AspNetCore.Identity;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using NodaTime;

using NodaTime.Serialization.SystemTextJson;

using Npgsql;

using Constants = BookManager.Application.Common.Constants;

namespace BookManager.Application;

public static class DependencyInjection

{

public static IServiceCollection ConfigureDataPersistence(

this IServiceCollection services,

IConfiguration configuration)

{

var conStrBuilder = new NpgsqlConnectionStringBuilder(configuration.GetConnectionString("Main"))

{

Password = configuration["DbPassword"]

};

var connection = conStrBuilder.ConnectionString;

services.AddDbContextFactory<AppDbContext>(options =>

options.UseNpgsql(connection, o => o.UseNodaTime())

.UseSnakeCaseNamingConvention());

services.AddScoped<IAppDbContext>(provider => provider.GetRequiredService<AppDbContext>());

services.AddIdentityCore<User>(options =>

{

options.Password.RequireLowercase = false;

options.Password.RequireUppercase = false;

options.Password.RequireDigit = true;

options.Password.RequireNonAlphanumeric = false;

})

.AddRoles<IdentityRole<Guid>>()

.AddEntityFrameworkStores<AppDbContext>();

return services;

}

public static IServiceCollection AddApplicationServices(this IServiceCollection services,

IConfiguration configuration)

{

services.Configure<MerriamWebsterOptions>(configuration.GetSection(MerriamWebsterOptions.MerriamWebster));

services.AddSingleton<IFileStorage, FileStorage>();

services.AddSingleton<IIndexingTaskQueue>(\_ => new IndexingTaskQueue(Constants.Default.IndexingQueueCapacity));

services.ConfigureHttpJsonOptions(jsonOptions =>

{

jsonOptions.SerializerOptions.ConfigureForNodaTime(DateTimeZoneProviders.Tzdb);

});

services.AddScoped<IBookService, BookService>();

services.AddScoped<IUserService, UserService>();

services.AddScoped<ISearchService, SearchService>();

services.AddScoped<IIndexingService, IndexingService>();

services.AddScoped<IAuthenticationService, AuthenticationService>();

services.AddScoped<IBookFileHandler, PdfBookFileHandler>();

services.AddScoped<ITranslationService, YTranslationService>();

services.AddScoped<ITextSummarizationService, YTextSummarizationService>();

services.AddScoped<IWordDictionaryService, DictionaryService>();

services.AddScoped<IBookCollectionService, BookCollectionService>();

services.AddTransient<IThirdPartyDictionaryProvider, MerriamWebsterDictionaryProvider>();

services.AddTransient<IThirdPartyDictionaryProvider, YAiDictionaryProvider>();

services.AddScoped<IValidator<PageRequestDto>, PageRequestValidator>();

services.AddScoped<IValidator<UserAddRequest>, UserAddRequestValidator>();

services.AddScoped<IValidator<WordDto>, WordDtoValidator>();

services.AddHttpClient<ITextSummarizationService, YTextSummarizationService>(httpClient =>

{

httpClient.BaseAddress = new Uri("https://llm.api.cloud.yandex.net");

});

services.AddHttpClient<YAiDictionaryProvider>();

services.AddHttpClient<MerriamWebsterDictionaryProvider>();

services.AddHostedService<IndexingHostedService>();

return services;

}

Файл IndexingHostedService.cs:

﻿using BookManager.Application.Common.Interfaces.Services;

using BookManager.Application.Notification;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using NodaTime;

namespace BookManager.Application.Indexing;

public sealed class IndexingHostedService(

ILogger<IndexingHostedService> logger,

IEnumerable<INotificationService> notificationServices,

IIndexingTaskQueue queue,

IServiceProvider serviceProvider

) : BackgroundService

{

public override async Task StopAsync(CancellationToken stoppingToken)

{

logger.LogInformation("Queued Indexing Hosted Service is stopping.");

await base.StopAsync(stoppingToken);

}

protected override async Task ExecuteAsync(CancellationToken stoppingToken)

{

while (!stoppingToken.IsCancellationRequested)

{

var workItem = await queue.DequeueAsync(stoppingToken);

await HandleWorkItem(stoppingToken, workItem);

}

}

private async Task HandleWorkItem(CancellationToken stoppingToken, IndexingWorkItem workItem)

{

try

{

using var scope = serviceProvider.CreateScope();

var service = scope.ServiceProvider.GetRequiredService<IIndexingService>();

int deletedCount;

switch (workItem.OperationType)

{

case IndexingWorkItemOperationType.Created:

await service.IndexDocumentAsync(workItem.BookDocumentId, stoppingToken);

logger.LogInformation(

"The text of the document {} was added for full-text search.",

workItem.BookDocumentId

);

var bookIndexedNotification = new Notification.Notification(

$"The text of the document {workItem.BookDocumentId} was added for full-text search.",

SystemClock.Instance.GetCurrentInstant()

);

await Notify(bookIndexedNotification, stoppingToken);

break;

case IndexingWorkItemOperationType.Replaced:

deletedCount = await service.DeleteDocumentTextsAsync(workItem.BookDocumentId, stoppingToken);

logger.LogInformation(

"The text of the {} document was deleted successfully. (deleted rows = {})",

workItem.BookDocumentId,

deletedCount

);

await service.IndexDocumentAsync(workItem.BookDocumentId, stoppingToken);

logger.LogInformation(

"The text of the document {} was added for full-text search.",

workItem.BookDocumentId

);

var updatedNotification = new Notification.Notification(

$"The text of the document {workItem.BookDocumentId} was updated for full-text search.",

SystemClock.Instance.GetCurrentInstant()

);

await Notify(updatedNotification, stoppingToken);

break;

case IndexingWorkItemOperationType.Deleted:

deletedCount = await service.DeleteDocumentTextsAsync(workItem.BookDocumentId, stoppingToken);

logger.LogInformation(

"The text of {} for indexing was deleted successfully. (deleted rows = {})",

workItem.BookDocumentId,

deletedCount

);

var bookTextDeletedNotification = new Notification.Notification(

$"The text of {workItem.BookDocumentId} for indexing was deleted successfully.",

SystemClock.Instance.GetCurrentInstant()

);

await Notify(bookTextDeletedNotification, stoppingToken);

break;

default:

throw new ArgumentOutOfRangeException(nameof(stoppingToken));

}

}

catch (Exception ex)

{

logger.LogError(ex,

"Error occurred executing {WorkItem}.", nameof(workItem));

}

}

private async Task Notify(Notification.Notification notification, CancellationToken cancellationToken)

{

foreach (var service in notificationServices)

{

await service.SendAsync(notification, cancellationToken);

}

}

Файл IndexingService.cs:

﻿using BookManager.Application.Common.Exceptions;

using BookManager.Application.Common.Interfaces;

using BookManager.Application.Common.Interfaces.Services;

namespace BookManager.Application.Indexing;

public sealed class IndexingService(

IAppDbContext dbContext,

IEnumerable<IBookFileHandler> bookFileHandlers,

IFileStorage storage

) : IIndexingService

{

private const int MaxPageCountToSave = 100;

public async Task IndexDocumentAsync(Guid bookId, CancellationToken cancellationToken)

{

var book = await dbContext.Books.FindAsync([bookId], cancellationToken);

if (book == null)

throw new EntityNotFoundException();

var fileStream = storage.GetFileStream(book.Filename);

foreach (var bookFileHandler in bookFileHandlers)

{

if (bookFileHandler.FileType != book.FileType) continue;

var count = 0;

await foreach (var bookText in bookFileHandler.StreamBookTexts(book.Id, fileStream)

.WithCancellation(cancellationToken))

{

if (count >= MaxPageCountToSave)

{

count = 0;

await dbContext.SaveChangesAsync(cancellationToken);

}

dbContext.BookTexts.Add(bookText);

count++;

}

break;

}

await fileStream.DisposeAsync();

await dbContext.SaveChangesAsync(cancellationToken);

}

public async Task<int> DeleteDocumentTextsAsync(Guid bookDocumentId, CancellationToken cancellationToken)

{

return await dbContext.BookTexts

.Where(text => text.BookDocumentId == bookDocumentId)

.ExecuteDeleteAsync(cancellationToken);

}

}

Файл IndexingTaskQueue.cs:

﻿using System.Threading.Channels;

namespace BookManager.Application.Indexing;

public sealed class IndexingTaskQueue: IIndexingTaskQueue

{

private readonly Channel<IndexingWorkItem> \_queue;

public IndexingTaskQueue(int capacity)

{

var options = new BoundedChannelOptions(capacity)

{

FullMode = BoundedChannelFullMode.Wait

};

\_queue = Channel.CreateBounded<IndexingWorkItem>(options);

}

public async ValueTask QueueAsync(IndexingWorkItem workItem)

{

await \_queue.Writer.WriteAsync(workItem);

}

public async ValueTask<IndexingWorkItem> DequeueAsync(CancellationToken cancellationToken)

{

return await \_queue.Reader.ReadAsync(cancellationToken);

}

}

Файл IndexingWorkItem.cs:

﻿namespace BookManager.Application.Indexing;

public enum IndexingWorkItemOperationType

{

Created,

Replaced,

Deleted

}

public sealed record IndexingWorkItem(

IndexingWorkItemOperationType OperationType,

Guid BookDocumentId

);

Файл INotificationService.cs:

﻿

using System.Threading.Channels;

namespace BookManager.Application.Notification;

public interface INotificationService

{

public ValueTask<Channel<Notification>> SubscribeAsync(CancellationToken cancellationToken);

public ValueTask UnsubscribeAsync(Channel<Notification> channel);

public ValueTask SendAsync(Notification notification, CancellationToken cancellationToken);

}

Файл Notification.cs:

﻿using NodaTime;

namespace BookManager.Application.Notification;

public sealed record Notification(string Message, Instant DateTime);

Файл NotificationService.cs:

﻿using System.Threading.Channels;

namespace BookManager.Application.Notification;

public class NotificationService : INotificationService

{

private readonly List<Channel<Notification>> \_consumers = [];

public ValueTask<Channel<Notification>> SubscribeAsync(CancellationToken cancellationToken)

{

var consumerChannel = Channel.CreateUnbounded<Notification>();

\_consumers.Add(consumerChannel);

return ValueTask.FromResult(consumerChannel);

}

public ValueTask UnsubscribeAsync(Channel<Notification> channel)

{

\_consumers.Remove(channel);

return ValueTask.CompletedTask;

}

public async ValueTask SendAsync(Notification notification, CancellationToken cancellationToken)

{

foreach (var consumer in \_consumers)

{

await consumer.Writer.WriteAsync(notification, cancellationToken);

}

}

}

Файл AppDbContext.cs:

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

namespace BookManager.Application.Persistence;

public class AppDbContext(DbContextOptions<AppDbContext> options)

: IdentityUserContext<User, Guid>(options), IAppDbContext

{

public DbSet<BookCollection> BookCollections => Set<BookCollection>();

public DbSet<Book> Books => Set<Book>();

public DbSet<BookText> BookTexts => Set<BookText>();

public DbSet<BookUserStats> BookUserStatsSet => Set<BookUserStats>();

public DbSet<DictionaryWord> DictionaryWords => Set<DictionaryWord>();

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<User>(u => u.ToTable("users"));

modelBuilder.Entity<IdentityRole>(r => r.ToTable("user\_roles"));

modelBuilder.Entity<IdentityUserLogin<Guid>>(r => r.ToTable("user\_logins"));

modelBuilder.Entity<IdentityUserClaim<Guid>>(r => r.ToTable("user\_claims"));

modelBuilder.Entity<IdentityUserToken<Guid>>(r => r.ToTable("user\_tokens"));

modelBuilder.Entity<BookUserStats>()

.HasKey(s => new { s.BookId, s.UserId });

modelBuilder.Entity<BookText>()

.HasIndex(bdt => bdt.Text)

.HasMethod("GIN")

.IsTsVectorExpressionIndex("english");

modelBuilder.Entity<Book>()

.HasMany(b => b.Collections)

.WithMany(bc => bc.Books);

modelBuilder.Entity<DictionaryWord>()

.Navigation(w => w.Definitions)

.AutoInclude();

}

}

Файл FileStorage.cs:

using System.Buffers;

using Blake3;

using BookManager.Application.Common.Config;

using BookManager.Application.Common.Interfaces;

using Microsoft.Extensions.Options;

namespace BookManager.Application.Persistence.FileSystem;

public sealed class FileStorage : IFileStorage, IDisposable

{

private const int FileChunkBufferSize = 8192;

private readonly FileStorageOptions \_options;

private Hasher \_hasher;

public FileStorage(IOptions<FileStorageOptions> options)

{

\_options = options.Value;

\_hasher = Hasher.New();

InitDirectory();

}

public FileStream GetFileStream(string filename)

{

var filepath = Path.Combine(\_options.DirectoryPath, filename);

return File.OpenRead(filepath);

}

public async IAsyncEnumerable<byte[]> ReadFileAsync(string filename)

{

await using var stream = File.OpenRead(filename);

var pool = MemoryPool<byte>.Shared.Rent(FileChunkBufferSize);

while (await stream.ReadAsync(pool.Memory) > 0)

{

yield return pool.Memory.ToArray();

}

}

public string GetFileHash(string filename)

{

var filepath = Path.Combine(\_options.DirectoryPath, filename);

\_hasher.Reset();

\_hasher.Update(File.ReadAllBytes(filepath));

return \_hasher.Finalize().ToString();

}

public async Task<FileInfo> SaveFileAsync(string filename, Stream stream)

{

var filepath = Path.Combine(\_options.DirectoryPath, filename);

await using var fileStream = stream;

await using (var file = File.Create(filepath))

{

fileStream.Seek(0, SeekOrigin.Begin);

await fileStream.CopyToAsync(file);

}

return new FileInfo(filepath);

}

public void DeleteFile(string filename)

{

File.Delete(filename);

}

private void InitDirectory()

{

Directory.CreateDirectory(\_options.DirectoryPath);

}

public void Dispose()

{

\_hasher.Dispose();

}

}

Файл AuthenticationService.cs:

﻿using BookManager.Application.Common.Config;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces;

using Microsoft.AspNetCore.Identity;

using Microsoft.Extensions.Options;

using Microsoft.IdentityModel.JsonWebTokens;

using Microsoft.IdentityModel.Tokens;

using JwtRegisteredClaimNames = System.IdentityModel.Tokens.Jwt.JwtRegisteredClaimNames;

namespace BookManager.Application.Services;

public class AuthenticationService(

UserManager<User> userManager,

IOptions<JwtTokenOptions> options,

TokenValidationParameters tokenValidationParameters,

SecurityKey jsonWebKey)

: IAuthenticationService

{

private readonly JwtTokenOptions \_jwtOptions = options.Value;

private readonly JsonWebTokenHandler \_tokenHandler = new();

public async Task<AuthenticationResponseDto> SignIn(AuthenticationRequestDto request)

{

var user = await userManager.FindByNameAsync(request.Name);

if (user == null)

return new AuthenticationResponseDto { Status = AuthenticationStatus.Failed };

var verificationResult = userManager.PasswordHasher

.VerifyHashedPassword(user, user.PasswordHash!, request.PinCode);

if (verificationResult == PasswordVerificationResult.Failed)

return new AuthenticationResponseDto { Status = AuthenticationStatus.Failed };

var accessToken = GenerateToken(user, \_jwtOptions.AccessTokenLifetimeInMinutes);

var refreshToken = GenerateToken(user, \_jwtOptions.RefreshTokenLifetimeInMinutes);

return new AuthenticationResponseDto

{

AccessToken = accessToken,

RefreshToken = refreshToken

};

}

public async Task<AuthenticationResponseDto> RefreshToken(string refreshTokenString)

{

var token = \_tokenHandler.ReadJsonWebToken(refreshTokenString);

var validationResult = await \_tokenHandler.ValidateTokenAsync(token, tokenValidationParameters);

if (!validationResult.IsValid || validationResult.Claims[JwtRegisteredClaimNames.Sub] is not string userId)

return new AuthenticationResponseDto { Status = AuthenticationStatus.Failed };

var user = await userManager.FindByIdAsync(userId);

if (user == null) return new AuthenticationResponseDto { Status = AuthenticationStatus.Failed };

return new AuthenticationResponseDto

{

AccessToken = GenerateToken(user, \_jwtOptions.AccessTokenLifetimeInMinutes)

};

}

private string GenerateToken(User user, double lifetimeInMinutes)

{

var credentials = new SigningCredentials(jsonWebKey, SecurityAlgorithms.EcdsaSha256Signature);

var claims = new Dictionary<string, object>

{

[JwtRegisteredClaimNames.Name] = user.UserName!,

[JwtRegisteredClaimNames.Sub] = user.Id,

};

var descriptor = new SecurityTokenDescriptor

{

Issuer = \_jwtOptions.Issuer,

Audience = \_jwtOptions.Audience,

Claims = claims,

Expires = DateTime.Now.AddMinutes(lifetimeInMinutes),

SigningCredentials = credentials

};

return \_tokenHandler.CreateToken(descriptor);

}

}

Файл BookCollectionService.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using BookManager.Application.Common.Interfaces.Services;

namespace BookManager.Application.Services;

public class BookCollectionService(IAppDbContext dbContext) : IBookCollectionService

{

public async Task<IEnumerable<BookCollectionDto>> GetAllAsync()

{

return await dbContext.BookCollections

.Include(bc => bc.Books)

.Select(bc => bc.ToDto())

.ToListAsync();

}

public async Task<BookCollectionDto> CreateAsync(BookCollectionModRequest request)

{

if (string.IsNullOrEmpty(request.Name))

throw new ArgumentException(nameof(request.Name));

if (dbContext.BookCollections.Select(bc => bc.Name).Contains(request.Name))

throw new ArgumentException("A book collection with this name already exists.", nameof(request));

var existingBooks = request.Books is { Count: > 0 }

? await FilterBooksByIds(request.Books.Select(b => b.DocumentDetails.Id))

.ToListAsync()

: [];

var newCollection = new BookCollection

{

Name = request.Name,

Books = existingBooks

};

var entry = dbContext.BookCollections.Add(newCollection);

await dbContext.SaveChangesAsync();

return entry.Entity.ToDto();

}

public async Task<BookCollectionDto> UpdateAsync(Guid id, BookCollectionModRequest request)

{

if (string.IsNullOrEmpty(request.Name))

throw new ArgumentException(nameof(request.Name));

var isNameAlreadyExists = dbContext.BookCollections

.Where(bc => bc.Id != id)

.Select(bc => bc.Name)

.Contains(request.Name);

if (isNameAlreadyExists)

throw new ArgumentException("A book collection with this name already exists.", nameof(request));

var foundCollection =

await dbContext.BookCollections

.Include(bc => bc.Books)

.FirstOrDefaultAsync(bc => bc.Id == id);

if (foundCollection == null)

throw new EntityNotFoundException();

var existingBooks = request.Books is { Count: > 0 }

? await FilterBooksByIds(request.Books.Select(dto => dto.DocumentDetails.Id)).ToListAsync()

: [];

foundCollection.Name = request.Name;

foundCollection.Books.Clear();

foreach (var existingBook in existingBooks)

{

foundCollection.Books.Add(existingBook);

}

dbContext.BookCollections.Update(foundCollection);

await dbContext.SaveChangesAsync();

return foundCollection.ToDto();

}

public async Task DeleteAsync(Guid id)

{

var foundCollection = await dbContext.BookCollections.FindAsync(id);

if (foundCollection == null) throw new EntityNotFoundException();

dbContext.BookCollections.Remove(foundCollection);

await dbContext.SaveChangesAsync();

}

private IQueryable<Book> FilterBooksByIds(IEnumerable<Guid> ids)

{

return dbContext.Books.Where(b => ids.Any(id => id == b.Id));

}

}

Файл BookService.cs:

using System.Linq.Expressions;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using BookManager.Application.Common.Interfaces;

using BookManager.Application.Common.Interfaces.Services;

using BookManager.Application.Indexing;

using BookManager.Domain.Enums;

using NodaTime;

namespace BookManager.Application.Services;

public sealed class BookService(

IAppDbContext dbContext,

IFileStorage fileStorage,

IEnumerable<IBookFileHandler> bookFileHandlers,

IIndexingTaskQueue indexingTaskQueue)

: IBookService

{

private IIndexingTaskQueue IndexingTaskQueue { get; } = indexingTaskQueue;

private static readonly Dictionary<string, Expression<Func<Book, object>>> BookAvailableSortOptions = new()

{

["title"] = b => b.Title!,

["isbn"] = b => b.Isbn!,

["recent\_access"] = b => b.Stats.FirstOrDefault()!.RecentAccess

};

public async Task<PageDto<BookDto>> GetPageAsync(

PageRequestDto request,

Expression<Func<Book, bool>>? predicate = null,

User? user = null)

{

var normalizedPageNumber = PageDto<BookDto>.GetNormalizedPageNumber(request.PageNumber);

var query = dbContext.Books.AsQueryable();

if (predicate != null)

query = query.Where(predicate);

if (user != null)

query = query.Include(b => b.Stats.Where(u => u.UserId == user.Id));

var totalItemCount = await query.CountAsync();

var orderExpr = request.SortBy != null && BookAvailableSortOptions.TryGetValue(request.SortBy, out var expr)

? expr

: BookAvailableSortOptions["recent\_access"];

if (request.SortBy == "recent\_access")

{

Expression<Func<Book, object>> orderByNullExpr = b => b.Stats.FirstOrDefault().RecentAccess == null;

query = request.SortOrder == SortOrder.Desc

? query.OrderBy(orderByNullExpr).ThenByDescending(orderExpr)

: query.OrderBy(orderByNullExpr).ThenBy(orderExpr);

}

else

{

query = request.SortOrder == SortOrder.Desc ? query.OrderByDescending(orderExpr) : query.OrderBy(orderExpr);

}

query = query.Skip((normalizedPageNumber - 1) \* request.PageSize)

.Take(request.PageSize);

var pageCount = PageDto<BookDto>.CountPage(totalItemCount, request.PageSize);

var items = await query.Select(b => b.ToDto()).ToListAsync();

return PageDto<BookDto>.Builder.Create()

.SetPageNumber(normalizedPageNumber)

.SetPageSize(items.Count)

.SetTotalItemCount(totalItemCount)

.SetPageCount(pageCount)

.SetItems(items)

.Build();

}

public async Task<BookDto?> GetByIdAsync(Guid bookId, Guid? userId = null)

{

var query = dbContext.Books.AsQueryable();

if (userId != null)

{

query = query.Include(b => b.Stats.Where(s => s.UserId == userId));

}

var book = await query.FirstOrDefaultAsync(b => b.Id == bookId);

return book?.ToDto();

}

public async Task<BookDto> AddBookAsync(Stream fileStream, BookMetadataDto bookMetadata)

{

var id = Guid.NewGuid();

var filename = $"{id}-{bookMetadata.Filename}";

var fileInfo = await fileStorage.SaveFileAsync(filename, fileStream);

var fileType = DocumentFileTypeUtils.GetFileType(fileInfo.FullName);

var bookFileHandler = bookFileHandlers.FirstOrDefault(handler => handler.FileType == fileType) ??

throw new Exception();

await using var bookFileStream = fileStorage.GetFileStream(filename);

var extractedTitle = bookFileHandler.GetBookTitle(bookFileStream);

var thumbnailImage = await SaveThumbnailImage(bookFileHandler, bookFileStream, filename);

var book = new Book

{

Id = id,

Filename = fileInfo.Name,

FileSize = fileInfo.Length,

FileType = BookFileType.Pdf,

Title = !string.IsNullOrEmpty(bookMetadata.Title) ? bookMetadata.Title

: string.IsNullOrEmpty(extractedTitle) ? bookMetadata.Filename : extractedTitle,

ThumbnailFilename = thumbnailImage,

PageCount = CountNumberOfPages(fileInfo.FullName),

Authors = bookMetadata.Authors == null || !bookMetadata.Authors.Any()

? bookFileHandler.GetAuthorList(bookFileStream).ToArray()

: bookMetadata.Authors.ToArray()

};

var entry = dbContext.Books.Add(book);

await dbContext.SaveChangesAsync();

var document = entry.Entity;

await IndexingTaskQueue.QueueAsync(

new IndexingWorkItem(

IndexingWorkItemOperationType.Created,

document.Id

)

);

return document.ToDto();

}

Файл DictionaryService.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using BookManager.Application.Common.Interfaces.Services;

using FluentValidation;

namespace BookManager.Application.Services;

public sealed class DictionaryService(

IAppDbContext dbContext,

IEnumerable<IThirdPartyDictionaryProvider> thirdPartyDictionaryProviders,

IValidator<WordDto> validator) : IWordDictionaryService

{

public IEnumerable<string> GetThirdPartyProviderNames()

{

return thirdPartyDictionaryProviders.Select(provider => provider.ProviderName).ToList();

}

public async Task<IEnumerable<WordDto>> FindAsync(string word)

{

return await dbContext.DictionaryWords

.Where(w => w.Word == word || w.Aliases.Select(alias => alias.Alias).Contains(word))

.Select(w => w.ToDto())

.ToListAsync();

}

public async Task<IEnumerable<WordDto>> FindInExtDictAsync(string word, string thirdPartyProviderName)

{

var provider =

thirdPartyDictionaryProviders.FirstOrDefault(provider => provider.ProviderName == thirdPartyProviderName);

if (provider == null)

throw new ArgumentException(null, nameof(thirdPartyProviderName));

return await provider.GetDefinitionAsync(word);

}

public async Task<WordDto> AddWordAsync(WordDto word)

{

var validationResult = await validator.ValidateAsync(word);

if (!validationResult.IsValid) throw new ArgumentException("Invalid word", nameof(word));

var entry = dbContext.DictionaryWords.Add(word.ToEntity());

await dbContext.SaveChangesAsync();

return entry.Entity.ToDto();

}

public async Task<WordDto> UpdateWordAsync(string wordId, WordDto word)

{

var validationResult = await validator.ValidateAsync(word);

if (!validationResult.IsValid) throw new ArgumentException("Invalid word", nameof(word));

var foundEntity = await dbContext.DictionaryWords.FindAsync(wordId);

if (foundEntity == null) throw new EntityNotFoundException();

foundEntity.Transcription = word.Transcription;

foundEntity.LanguageCode = word.LanguageCode;

foundEntity.Definitions.Clear();

foreach (var wordDef in word.Definitions)

{

foundEntity.Definitions.Add(wordDef.ToEntity());

}

foundEntity.Aliases.Clear();

foreach (var alias in word.Aliases)

{

foundEntity.Aliases.Add(alias.ToEntity());

}

var entry = dbContext.DictionaryWords.Update(foundEntity);

await dbContext.SaveChangesAsync();

return entry.Entity.ToDto();

}

public async Task DeleteWordAsync(string word)

{

var foundEntity = await dbContext.DictionaryWords.FindAsync(word);

if (foundEntity == null) throw new EntityNotFoundException();

dbContext.DictionaryWords.Remove(foundEntity);

await dbContext.SaveChangesAsync();

}

}

Файл EpubBookFileHandler.cs:

﻿using System.Text;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces.Services;

using BookManager.Domain.Enums;

using HtmlAgilityPack;

using VersOne.Epub;

namespace BookManager.Application.Services;

public sealed class EpubBookFileHandler: IBookFileHandler

{

public BookFileType FileType => BookFileType.Epub;

public int? CountNumberOfPages(Stream bookStream)

{

return null;

}

public IEnumerable<string> GetAuthorList(Stream bookStream)

{

bookStream.Seek(0, SeekOrigin.Begin);

var book = EpubReader.ReadBook(bookStream);

return book.AuthorList;

}

public RawImageDto? GetPreviewImage(Stream bookStream)

{

bookStream.Seek(0, SeekOrigin.Begin);

var book = EpubReader.ReadBook(bookStream);

// book.CoverImage;

// TODO: Need to implement GetJpegImageAsync

return null;

}

public Task<Stream> GetJpegImageAsync(RawImageDto rawImage)

{

throw new NotImplementedException();

}

public IAsyncEnumerable<BookText> StreamBookTexts(Guid bookId, Stream stream)

{

stream.Seek(0, SeekOrigin.Begin);

var book = EpubReader.ReadBook(stream);

var contents = book

.ReadingOrder

.Select(textContent => new BookText

{

BookDocumentId = bookId,

Text = PrintTextContentFile(textContent),

}).ToAsyncEnumerable();

stream.Dispose();

return contents;

}

public string? GetBookTitle(Stream bookStream)

{

bookStream.Seek(0, SeekOrigin.Begin);

using var document = EpubReader.OpenBook(bookStream);

return string.IsNullOrEmpty(document.Title) ? null : document.Title;

}

private static string PrintTextContentFile(EpubLocalTextContentFile textContentFile)

{

HtmlDocument htmlDocument = new();

htmlDocument.LoadHtml(textContentFile.Content);

StringBuilder sb = new();

foreach (var node in htmlDocument.DocumentNode.SelectNodes("//text()"))

{

sb.AppendLine(node.InnerText.Trim());

}

var contentText = sb.ToString();

return contentText;

}

}

Файл MerriamWebsterDictionaryProvider.cs:

﻿using System.Text.Json;

using System.Text.RegularExpressions;

using BookManager.Application.Common.Config;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using BookManager.Application.Common.Interfaces.Services;

using Flurl;

using Microsoft.Extensions.Options;

namespace BookManager.Application.Services;

public partial class MerriamWebsterDictionaryProvider(

HttpClient httpClient,

IOptions<MerriamWebsterOptions> options)

: IThirdPartyDictionaryProvider

{

private const string RequestUri = "https://www.dictionaryapi.com/api/v3/references/collegiate/json";

private readonly MerriamWebsterOptions \_options = options.Value;

private readonly JsonSerializerOptions \_jsonSerializerOptions = new(JsonSerializerDefaults.Web);

private readonly Regex \_wordIdRegex = WordIdRegex();

public string ProviderName => "MerriamWebster";

public async Task<IEnumerable<WordDto>> GetDefinitionAsync(string word)

{

if (string.IsNullOrEmpty(\_options.ApiKey.Trim()))

throw new NotAvailableException(

$"The API key is not defined for {nameof(MerriamWebsterDictionaryProvider)} service."

);

var requestUri = RequestUri.AppendPathSegment(word.Trim('/'))

.AppendQueryParam("key", \_options.ApiKey)

.ToUri();

var requestMessage = new HttpRequestMessage

{

Method = HttpMethod.Get,

RequestUri = requestUri,

};

var responseMessage = await httpClient.SendAsync(requestMessage);

try

{

await using var contentStream = await responseMessage.Content.ReadAsStreamAsync();

var results = await JsonSerializer.DeserializeAsync<ICollection<MerriamWebsterDefDto>>(

contentStream,

\_jsonSerializerOptions

);

if (results == null) return new List<WordDto>();

var words = new List<WordDto>();

foreach (var def in results)

{

if (string.IsNullOrEmpty(def.HeadwordInfo.Headword)

|| def.ShortDefinition is { Length: 0 }

|| def.HeadwordInfo.Pronunciations is { Length: 0 }

|| string.IsNullOrEmpty(def.HeadwordInfo.Pronunciations?[0].WrittenPronunciation)

)

continue;

var definitions = def.ShortDefinition

.Select(definition => new WordDefinitionDto(def.FunctionalLabel, "", definition))

.ToList();

var transcription = def.HeadwordInfo.Pronunciations[0].WrittenPronunciation;

var normalizedWordId = \_wordIdRegex.Replace(def.Meta.Id, string.Empty);

var aliases = def.Meta.Stems.Select(stem => new WordAlias(stem)).ToList();

words.Add(new WordDto(normalizedWordId, transcription, string.Empty, Aliases: aliases,

Definitions: definitions));

}

return words;

}

catch (Exception)

{

return new List<WordDto>();

}

}

[GeneratedRegex(":\\d$")]

private static partial Regex WordIdRegex();

}

Файл PdfBookFileHandler.cs:

﻿using BookManager.Application.Common;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces.Services;

using BookManager.Domain.Enums;

using Docnet.Core;

using Docnet.Core.Models;

using SixLabors.ImageSharp;

using SixLabors.ImageSharp.PixelFormats;

using SixLabors.ImageSharp.Processing;

using UglyToad.PdfPig;

namespace BookManager.Application.Services;

public sealed class PdfBookFileHandler : IBookFileHandler

{

public BookFileType FileType => BookFileType.Pdf;

public int? CountNumberOfPages(Stream bookStream)

{

bookStream.Seek(0, SeekOrigin.Begin);

using var document = PdfDocument.Open(bookStream);

return document.NumberOfPages;

}

public IEnumerable<string> GetAuthorList(Stream bookStream)

{

bookStream.Seek(0, SeekOrigin.Begin);

using var document = PdfDocument.Open(bookStream);

var authors = document.Information.Author;

return new List<string>([authors]);

}

public RawImageDto? GetPreviewImage(Stream bookStream)

{

using var stream = new MemoryStream();

bookStream.Seek(0, SeekOrigin.Begin);

bookStream.CopyTo(stream);

using var docReader = DocLib.Instance.GetDocReader(

stream.ToArray(),

new PageDimensions(Constants.ThumbnailPreviewWidth, Constants.ThumbnailPreviewHeight

));

using var pageReader = docReader.GetPageReader(0);

var rawBytes = pageReader.GetImage();

if (rawBytes == null) return null;

var width = pageReader.GetPageWidth();

var height = pageReader.GetPageHeight();

return new RawImageDto(rawBytes, width, height);

}

public async Task<Stream> GetJpegImageAsync(RawImageDto rawImage)

{

var image = Image.LoadPixelData<Bgra32>(rawImage.Data, rawImage.Width, rawImage.Height);

image.Mutate(x => x.BackgroundColor(Color.White));

var stream = new MemoryStream();

await image.SaveAsJpegAsync(stream);

return stream;

}

public string? GetBookTitle(Stream bookStream)

{

bookStream.Seek(0, SeekOrigin.Begin);

using var document = PdfDocument.Open(bookStream);

return document.Information.Title;

}

public async IAsyncEnumerable<BookText> StreamBookTexts(Guid bookId, Stream stream)

{

stream.Seek(0, SeekOrigin.Begin);

var document = PdfDocument.Open(stream);

var pages = document.GetPages()

.Select(page => new BookText

{

BookDocumentId = bookId,

Text = page.Text.Replace("\u0000", string.Empty),

PageNumber = page.Number

}

);

await foreach (var bookText in pages.ToAsyncEnumerable())

{

yield return bookText;

}

document.Dispose();

}

}

Файл SearchService.cs:

﻿using System.Linq.Expressions;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces.Services;

using LinqKit;

namespace BookManager.Application.Services;

public sealed class SearchService(IAppDbContext dbContext, IBookService bookService) : ISearchService

{

public async Task<PageDto<BookDto>> SearchByBookDetailsAsync(SearchRequestDto searchRequest, User? user = null)

{

var predicate = PredicateBuilder.New<Book>();

foreach (var propInfo in searchRequest.GetType().GetProperties())

{

if (propInfo.GetValue(searchRequest) is not string stringValue

|| !typeof(Book).GetProperties().Select(p => p.Name).Contains(propInfo.Name))

continue;

var expr = GetStringContainsExpression<Book>(propInfo.Name, stringValue);

predicate = predicate.Or(expr);

}

if (searchRequest.Authors is { Length: > 0 })

{

predicate = predicate.Or(b => b.Authors.Intersect(searchRequest.Authors).Any());

}

var request = new PageRequestDto(

searchRequest.PageNumber,

searchRequest.PageSize,

searchRequest.SortProperty,

searchRequest.SortOrder ?? SortOrder.Asc

);

return await bookService.GetPageAsync(request, predicate, user);

}

public async Task<PageDto<BookTextDto>> SearchByBookTexts(TextSearchRequestDto request)

{

if (string.IsNullOrEmpty(request.Pattern)) throw new ArgumentException(request.Pattern);

var normalizedPageNumber = PageDto<BookTextDto>.GetNormalizedPageNumber(request.PageNumber);

var query = dbContext.BookTexts

.Where(t => EF.Functions.ToTsVector("english", t.Text).Matches(request.Pattern))

.OrderBy(t => t.PageNumber)

.Skip((normalizedPageNumber - 1) \* request.PageSize)

.Take(request.PageSize)

.Select(t => t.ToDto());

var totalItemCount = await query.CountAsync();

var pageCount = PageDto<BookTextDto>.CountPage(totalItemCount, request.PageSize);

var searchResults = await query.ToListAsync();

return PageDto<BookTextDto>.Builder.Create()

.SetPageCount(pageCount)

.SetPageNumber(normalizedPageNumber)

.SetTotalItemCount(totalItemCount)

.SetItems(searchResults)

.Build();

}

// Original: https://stackoverflow.com/questions/278684/how-do-i-create-an-expression-tree-to-represent-string-containsterm-in-c

// Also you can read: https://learn.microsoft.com/en-us/dotnet/csharp/advanced-topics/expression-trees/expression-trees-building

private static Expression<Func<T, bool>> GetStringContainsExpression<T>(string propertyName, string propertyValue)

{

var parameterExp = Expression.Parameter(typeof(T), "anything");

var propertyExp = Expression.Property(parameterExp, propertyName);

var method = typeof(string).GetMethod("Contains", [typeof(string)]);

var someValue = Expression.Constant(propertyValue, typeof(string));

var containsMethodExp = Expression.Call(propertyExp, method!, someValue);

return Expression.Lambda<Func<T, bool>>(containsMethodExp, parameterExp);

}

}

Файл UserService.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Exceptions;

using BookManager.Application.Common.Interfaces.Services;

using FluentValidation;

using Microsoft.AspNetCore.Identity;

namespace BookManager.Application.Services;

public sealed class UserService(

IAppDbContext dbContext,

IValidator<UserAddRequest> userAddRequestValidator,

UserManager<User> userManager,

IPasswordHasher<User> passwordHasher)

: IUserService

{

public IAsyncEnumerable<UserDto> GetAllUsers()

{

return dbContext.Users.Select(u => u.ToDto()).AsAsyncEnumerable();

}

public async Task<UserDto> CreateUser(UserAddRequest request)

{

var validationResult = await userAddRequestValidator.ValidateAsync(request);

if (!validationResult.IsValid) throw new ArgumentException("Invalid request", nameof(request));

var newUser = new User

{

UserName = request.Name,

};

newUser.PasswordHash = passwordHasher.HashPassword(newUser, request.PinCode);

var identityResult = await userManager.CreateAsync(newUser);

if (!identityResult.Succeeded)

throw new UserCreationException("Failed to create a user");

var user = await userManager.FindByNameAsync(newUser.UserName)

?? throw new UserCreationException("Failed to get user entity after an user account creation.");

return user.ToDto();

}

public async Task<UserDto> UpdateUser(Guid id, UserAddRequest request)

{

var validationResult = await userAddRequestValidator.ValidateAsync(request);

if (!validationResult.IsValid) throw new ArgumentException("Invalid request", nameof(request));

var foundUser = await dbContext.Users.FindAsync([id]);

if (foundUser == null || string.IsNullOrEmpty(foundUser.UserName)) throw new EntityNotFoundException();

foundUser.PasswordHash = passwordHasher.HashPassword(foundUser, request.PinCode);

var identityResult = await userManager.UpdateAsync(foundUser);

if (!identityResult.Succeeded)

throw new UserUpdateException("Failed to update user account.");

var updatedUser = await userManager.FindByNameAsync(foundUser.UserName)

?? throw new UserUpdateException("Failed to get user entity after an user account update.");

return updatedUser.ToDto();

}

public async Task DeleteUser(Guid id)

{

var found = await dbContext.Users.FindAsync([id]) ?? throw new EntityNotFoundException();

dbContext.Users.Remove(found);

await dbContext.SaveChangesAsync();

}

}

Файл YAiDictionaryProvider.cs:

﻿using System.Net;

using System.Net.Http.Headers;

using System.Net.Http.Json;

using System.Text.Json;

using BookManager.Application.Common.Config;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces.Services;

using Microsoft.Extensions.Logging;

using Microsoft.Extensions.Options;

using Yandex.Cloud.Ai.FoundationModels.V1;

using Yandex.Cloud.Credentials;

namespace BookManager.Application.Services;

public class YAiDictionaryProvider(

ILogger<YAiDictionaryProvider> logger,

HttpClient client,

ICredentialsProvider credentialsProvider,

IOptions<YandexCloudOptions> options)

: IThirdPartyDictionaryProvider

{

public string ProviderName => "YandexAi";

private readonly YandexCloudOptions \_options = options.Value;

private readonly JsonSerializerOptions \_serializerOptions = new(JsonSerializerDefaults.Web);

private const string RequestUri = "https://llm.api.cloud.yandex.net";

private const string JsonDivider = "```";

public async Task<IEnumerable<WordDto>> GetDefinitionAsync(string word)

{

if (!ValidateWord(word))

{

throw new ArgumentException("Invalid word", nameof(word));

}

var message = new Message

{

Role = "user",

Text = GetPromptString(word)

};

var completionRequest = new CompletionRequest

{

ModelUri = \_options.GptModelUri,

Messages = { message },

CompletionOptions = new CompletionOptions

{

Stream = false,

Temperature = \_options.GptModelTemperature,

MaxTokens = 1000

},

};

var token = credentialsProvider.GetToken();

if (token == null) throw new Exception("Failed to get IAM token to Yandex Cloud");

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

var responseMessage =

await client.PostAsync($"{RequestUri}/foundationModels/v1/completion",

JsonContent.Create(completionRequest));

if (responseMessage.StatusCode != HttpStatusCode.OK)

throw new Exception(responseMessage.ReasonPhrase);

var response = JsonSerializer.Deserialize<CompletionResponse>(await responseMessage.Content.ReadAsStreamAsync(),

\_serializerOptions);

if (response == null) throw new Exception("Failed to deserialize a response message");

if (!response.Result.Alternatives.Any()) return new List<WordDto>();

var resultText = response.Result

.Alternatives

.ElementAt(0)

.Message

.Text

.Replace("\n", "")

.Replace("/", "");

try

{

var startIndexOfJsonResult = resultText.IndexOf(JsonDivider, StringComparison.Ordinal);

var endIndexOfJsonResult = resultText.IndexOf(

JsonDivider,

startIndexOfJsonResult + JsonDivider.Length,

StringComparison.Ordinal

);

var jsonStringResult = startIndexOfJsonResult == -1 || endIndexOfJsonResult == -1

? resultText

: resultText.Substring(

startIndexOfJsonResult + JsonDivider.Length,

endIndexOfJsonResult - JsonDivider.Length

).Trim();

var wordDto = JsonSerializer.Deserialize<WordDto>(

jsonStringResult,

\_serializerOptions

);

return wordDto != null ? [wordDto] : new List<WordDto>();

}

catch (Exception e)

{

logger.LogError("The response message parsing error: {}", e.Message);

}

return new List<WordDto>();

}

private static bool ValidateWord(string word)

{

return word.Split(' ').Length < 3;

}

private static string GetPromptString(string word)

{

return

$"Найди определение для слова '{word}'. Верни результат в виде JSON объект на примере ```{{ \"word\": \"go\", \"transcription\": \"[gəu]\",\"definitions\":" +

"[{\"partOfSpeech\": \"verb\", \"definition\": \"to get dressed and prepare for the day or an event\" }," +

"{\"partOfSpeech\": \"phrasal verb\", \"definition\": \"to stand up after sitting or lying down\" } ]}```";

}

Файл YTextSummarizationService.cs:

﻿using System.Net;

using System.Net.Http.Headers;

using System.Net.Http.Json;

using System.Text;

using System.Text.Json;

using BookManager.Application.Common.Config;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces.Services;

using Microsoft.Extensions.Options;

using Yandex.Cloud.Ai.FoundationModels.V1;

using Yandex.Cloud.Credentials;

namespace BookManager.Application.Services;

public sealed class CompletionResponse

{

public ResponseResult Result { get; set; }

public class ResponseResult

{

public IEnumerable<Alternative> Alternatives { get; set; } = [];

public TokensUsage? Usage { get; set; }

public string? ModelVersion { get; set; }

public class Alternative

{

public AMessage Message { get; set; }

public string? Status { get; set; }

public record AMessage(string Role, string Text);

}

public record TokensUsage(string InputTextTokens, string CompletionTokens, string TotalTokens);

}

}

public sealed class YTextSummarizationService(

HttpClient client,

ICredentialsProvider credentialsProvider,

IOptions<YandexCloudOptions> options)

: ITextSummarizationService

{

private readonly YandexCloudOptions \_options = options.Value;

private readonly JsonSerializerOptions \_serializerOptions = new(JsonSerializerDefaults.Web);

public async Task<TextSummarizationResponseDto> SummarizeTextAsync(TextSummarizationRequestDto request)

{

var message = new Message

{

Role = "user",

Text = request.Text

};

var completionRequest = new CompletionRequest

{

ModelUri = \_options.ModelSummarizationUri,

Messages = { message },

CompletionOptions = new CompletionOptions

{

Stream = false,

Temperature = \_options.ModelSummarizationTemperature,

MaxTokens = 1000

},

};

var token = credentialsProvider.GetToken();

if (token == null) throw new Exception("Failed to get IAM token to Yandex Cloud");

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

var responseMessage =

await client.PostAsync("foundationModels/v1/completion", JsonContent.Create(completionRequest));

if (responseMessage.StatusCode != HttpStatusCode.OK)

{

throw new Exception(responseMessage.ReasonPhrase);

}

var response = JsonSerializer.Deserialize<CompletionResponse>(await responseMessage.Content.ReadAsStreamAsync(),

\_serializerOptions);

if (response == null) throw new Exception("Failed to deserialize a response message");

var sb = new StringBuilder();

sb.Append(string.Join('\n', response.Result.Alternatives.Select(a => string.Join('\n', a.Message.Text))));

sb.Append('\n');

return new TextSummarizationResponseDto(sb.ToString());

}

}

Файл YTranslationService.cs:

﻿using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces.Services;

using Microsoft.IdentityModel.Tokens;

using Yandex.Cloud;

using Yandex.Cloud.Ai.Translate.V2;

namespace BookManager.Application.Services;

public class YTranslationService(Sdk sdk) : ITranslationService

{

private readonly TranslationService.TranslationServiceClient \_client = sdk.Services.Ai.Translate.TranslationService;

public async Task<IEnumerable<LanguageDto>> ListLanguagesAsync()

{

var list = await \_client.ListLanguagesAsync(new ListLanguagesRequest());

return list.Languages.Select(l => new LanguageDto(l.Code, l.Name));

}

public async Task<DetectLanguageResponseDto> DetectLanguageAsync(DetectLanguageRequestDto request)

{

var response = await \_client.DetectLanguageAsync(new DetectLanguageRequest

{

Text = request.Text

});

return new DetectLanguageResponseDto(response.LanguageCode);

}

public async Task<TranslationResponseDto> TranslateAsync(TranslationRequestDto request)

{

var translationRequest = new TranslateRequest

{

TargetLanguageCode = request.TargetLanguage,

Texts = { request.SourceText }

};

if (request.SourceLanguage != null) translationRequest.SourceLanguageCode = request.SourceLanguage;

var response = await \_client.TranslateAsync(translationRequest);

if (response.Translations.IsNullOrEmpty())

{

throw new Exception("Failed to translate text");

}

var dto = new TranslationResponseDto

{

DetectedSourceLanguage = response.Translations[0].DetectedLanguageCode,

TargetLanguage = request.TargetLanguage,

TranslatedText = string.Join('\n', response.Translations.Select(t => t.Text))

};

return dto;

}

}

Файл PageRequestValidator.cs:

﻿using BookManager.Application.Common.DTOs;

using FluentValidation;

namespace BookManager.Application.Validators;

public class PageRequestValidator: AbstractValidator<PageRequestDto>

{

public PageRequestValidator()

{

RuleFor(p => p.PageNumber).GreaterThan(0);

RuleFor(p => p.PageSize).GreaterThan(0);

}

}

Файл UserAddRequestValidator.cs:

﻿using BookManager.Application.Common.DTOs;

using FluentValidation;

namespace BookManager.Application.Validators;

public class UserAddRequestValidator : AbstractValidator<UserAddRequest>

{

public UserAddRequestValidator()

{

RuleFor(user => user.Name).NotEmpty().MinimumLength(2).MaximumLength(32);

RuleFor(user => user.PinCode).Matches("^[0-9]{4,16}$").NotEmpty();

}

}

Файл WordDtoValidator.cs:

﻿using BookManager.Application.Common.DTOs;

using FluentValidation;

namespace BookManager.Application.Validators;

public class WordDtoValidator: AbstractValidator<WordDto>

{

public WordDtoValidator()

{

RuleFor(w => w.Word).Matches(@"^([\p{L}-]\*)( ?)([\p{L}-]\*)$").NotEmpty();

}

}

Файл Book.cs:

using System.ComponentModel.DataAnnotations;

using BookManager.Domain.Enums;

namespace BookManager.Domain.Entities;

public sealed class Book

{

[Key] public Guid Id { get; init; }

[Required] public required BookFileType FileType { get; init; }

[Required] public required string Filename { get; init; }

[Required] public required long FileSize { get; init; }

[MaxLength(256)] public string? Title { get; set; }

[MaxLength(256)] public string? Isbn { get; set; }

[MaxLength(16 \* 1024)] public string? Description { get; set; }

[MaxLength(256)] public string? PublisherName { get; set; }

public string[]? Authors { get; set; }

public string[]? Tags { get; set; }

public int? PageCount { get; set; }

public string? ThumbnailFilename { get; init; }

public ICollection<BookCollection> Collections { get; set; } = new List<BookCollection>();

public IEnumerable<BookUserStats> Stats { get; } = new List<BookUserStats>();

}

Файл BookCollection.cs:

﻿using System.ComponentModel.DataAnnotations;

using NodaTime;

namespace BookManager.Domain.Entities;

public class BookCollection

{

[Key]

public Guid Id { get; init; }

public required string Name { get; set; }

public ICollection<Book> Books { get; set; } = new List<Book>();

public Instant CreatedAt { get; init; }

}

Файл BookText.cs:

using System.ComponentModel.DataAnnotations;

namespace BookManager.Domain.Entities;

public sealed class BookText

{

[Key]

public Guid Id { get; init; }

// 4 MiB

[MaxLength(4194304)]

public required string Text { get; init; } = null!;

public Guid BookDocumentId { get; init; }

public int? PageNumber { get; init; }

}

Файл BookUserStats.cs:

﻿using NodaTime;

namespace BookManager.Domain.Entities;

public sealed class BookUserStats

{

public Guid UserId { get; init; }

public Guid BookId { get; init; }

public Instant RecentAccess { get; set; }

public long TotalReadingTime { get; set; }

public int? LastViewedPage { get; set; }

}

Файл DictionaryWord.cs:

﻿using System.ComponentModel.DataAnnotations;

namespace BookManager.Domain.Entities;

public sealed class DictionaryWord

{

[Key]

public string Word { get; init; } = null!;

public string? Transcription { get; set; } = string.Empty;

public string? LanguageCode { get; set; } = string.Empty;

public ICollection<DictionaryWordAlias> Aliases { get; } = new List<DictionaryWordAlias>();

public ICollection<DictionaryWordDefinition> Definitions { get; } = new List<DictionaryWordDefinition>();

}

Файл DictionaryWordAlias.cs:

﻿using System.ComponentModel.DataAnnotations;

namespace BookManager.Domain.Entities;

public class DictionaryWordAlias

{

[Key]

public string Alias { get; init; } = null!;

public DictionaryWord DictionaryWord { get; init; } = null!;

}

Файл DictionaryWordDefinition.cs:

﻿using System.ComponentModel.DataAnnotations;

namespace BookManager.Domain.Entities;

public sealed class DictionaryWordDefinition

{

[Key] public Guid Id { get; init; }

[MaxLength(1024)]

public required string SubjectName { get; init; }

[MaxLength(8192)]

public required string Definition { get; init; }

public string PartOfSpeech { get; init; } = null!;

public string DictionaryWordId { get; init; } = null!;

public DictionaryWord Word { get; init; } = null!;

}

Файл User.cs:

﻿using Microsoft.AspNetCore.Identity;

namespace BookManager.Domain.Entities;

public sealed class User: IdentityUser<Guid>

{

public IEnumerable<BookUserStats> Stats { get; } = new List<BookUserStats>();

}

Файл BookFileType.cs:

using BookManager.Domain.Exceptions;

using Tapper;

namespace BookManager.Domain.Enums;

[TranspilationSource]

public enum BookFileType

{

Pdf,

Epub

}

public static class DocumentFileTypeUtils

{

public static BookFileType GetFileType(string filename)

=> Path.GetExtension(filename).ToLower() switch

{

".pdf" => BookFileType.Pdf,

".epub" => BookFileType.Epub,

\_ => throw new UnsupportedFileTypeException()

};

}

Файл UnsupportedFileTypeException.cs:

namespace BookManager.Domain.Exceptions;

public class UnsupportedFileTypeException : Exception;

Файл ApiFixture.cs:

using System.Net.Http.Headers;

using BookManager.Application.Common.DTOs;

using BookManager.Application.Common.Interfaces;

using BookManager.Application.Common.Interfaces.Services;

using BookManager.Application.Persistence;

using Microsoft.AspNetCore.Mvc.Testing;

using Microsoft.EntityFrameworkCore;

using ServiceProviderServiceExtensions = Microsoft.Extensions.DependencyInjection.ServiceProviderServiceExtensions;

namespace BookManager.Tests.Api.IntegrationTests;

public class ApiFixture : IDisposable

{

private readonly WebTestAppFactory<Program> \_factory;

public ApiFixture()

{

\_factory = new WebTestAppFactory<Program>();

AddUser();

}

public void Dispose()

{

GC.SuppressFinalize(this);

using var scope = ServiceProviderServiceExtensions.CreateScope(\_factory.Services);

var dbContext = ServiceProviderServiceExtensions.GetRequiredService<AppDbContext>(scope.ServiceProvider);

dbContext.Database.EnsureDeleted();

\_factory.Dispose();

}

public HttpClient CreateClient()

{

return \_factory.CreateClient(new WebApplicationFactoryClientOptions

{

AllowAutoRedirect = false,

});

}

public HttpClient CreateAuthenticatedClient()

{

using var scope = ServiceProviderServiceExtensions.CreateScope(\_factory.Services);

var authService = ServiceProviderServiceExtensions.GetRequiredService<IAuthenticationService>(scope.ServiceProvider);

var token = authService.SignIn(new AuthenticationRequestDto(Constants.UserName, Constants.UserPinCode))

.Result

.AccessToken!;

var client = \_factory.CreateClient(new WebApplicationFactoryClientOptions

{

AllowAutoRedirect = false,

});

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token);

return client;

}

public void Cleanup()

{

using var scope = ServiceProviderServiceExtensions.CreateScope(\_factory.Services);

var dbContext = ServiceProviderServiceExtensions.GetRequiredService<AppDbContext>(scope.ServiceProvider);

dbContext.Books.ExecuteDelete();

dbContext.BookTexts.ExecuteDelete();

dbContext.DictionaryWords.ExecuteDelete();

}

private void AddUser()

{

using var scope = ServiceProviderServiceExtensions.CreateScope(\_factory.Services);

var userService = ServiceProviderServiceExtensions.GetRequiredService<IUserService>(scope.ServiceProvider);

var user = userService.CreateUser(new UserAddRequest

{

Name = Constants.UserName,

PinCode = Constants.UserPinCode,

}).Result;

Console.WriteLine($"The user ${user.Name} created.");

}

}

[CollectionDefinition("Api collection")]

public class ApiFixtureCollection : ICollectionFixture<ApiFixture>;

Файл app.component.html:

<app-main-layout>

<router-outlet></router-outlet>

</app-main-layout>

Файл app.component.ts:

import { ChangeDetectionStrategy, Component } from '@angular/core';

import { HubConnectionBuilder } from '@microsoft/signalr';

import { environment } from '../environments/environment';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrl: './app.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class AppComponent {

constructor() {

const connection = new HubConnectionBuilder()

.withUrl(`${environment.BASE\_URL}/notification`)

.build();

connection.on('book-indexing', (result) => {

console.log(result);

});

connection.start().then()

}

}

Файл app.module.ts:

import { CommonModule } from '@angular/common';

import { provideHttpClient, withInterceptors } from '@angular/common/http';

import { NgModule } from '@angular/core';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { MatButtonModule } from '@angular/material/button';

import { MatIconModule } from '@angular/material/icon';

import { MatInputModule } from '@angular/material/input';

import { MatListModule } from '@angular/material/list';

import { MatSelectModule } from '@angular/material/select';

import { MatSidenavModule } from '@angular/material/sidenav';

import { MatTooltipModule } from '@angular/material/tooltip';

import { BrowserModule } from '@angular/platform-browser';

import { provideAnimationsAsync } from '@angular/platform-browser/animations/async';

import {

provideRouter,

RouterLink,

RouterLinkActive,

RouterOutlet,

withEnabledBlockingInitialNavigation,

withViewTransitions,

} from '@angular/router';

import { MainLayoutComponent } from '@core/components/main-layout/main-layout.component';

import { ToolbarComponent } from '@core/components/toolbar/toolbar.component';

import { authInterceptor } from '@core/interceptors/auth.interceptor';

import { httpErrorInterceptor } from '@core/interceptors/http-error.interceptor';

import { provideAuthHttpClient } from '@core/providers/auth-http-client';

import { AppComponent } from './app.component';

import { routes } from './app.routes';

@NgModule({

declarations: [

AppComponent,

MainLayoutComponent,

ToolbarComponent,

],

imports: [

CommonModule,

FormsModule,

ReactiveFormsModule,

BrowserModule,

RouterOutlet,

RouterLink,

RouterLinkActive,

MatButtonModule,

MatIconModule,

MatListModule,

MatSelectModule,

MatInputModule,

MatTooltipModule,

MatSidenavModule,

],

providers: [

{

provide: Window,

useValue: window,

},

provideRouter(

routes,

withEnabledBlockingInitialNavigation(),

withViewTransitions(),

),

provideAnimationsAsync(),

provideHttpClient(

withInterceptors([httpErrorInterceptor, authInterceptor]),

),

provideAuthHttpClient(),

],

bootstrap: [AppComponent],

})

export class AppModule {

}

Файл app.routes.ts:

import { Routes } from '@angular/router';

import { CONSTANTS } from '@core/constants';

import { authGuard } from '@core/guards/auth.guard';

import { loginGuard } from '@core/guards/login.guard';

export const routes: Routes = [

{

path: CONSTANTS.ENDPOINTS.RECENT\_BOOKS,

canActivate: [authGuard],

loadComponent: () => import('@core/pages/recent-books/recent-books.component')

.then(m => m.RecentBooksComponent),

},

{

path: CONSTANTS.ENDPOINTS.EXPLORER,

canActivate: [authGuard],

loadComponent: () => import('@core/pages/library-explorer/library-explorer.component')

.then(m => m.LibraryExplorerComponent),

},

{

path: CONSTANTS.ENDPOINTS.BOOK\_COLLECTIONS,

canActivate: [authGuard],

loadComponent: () => import('@core/pages/book-collections/book-collections.component')

.then(m => m.BookCollectionsComponent)

},

{

path: CONSTANTS.ENDPOINTS.VIEWER,

canActivate: [authGuard],

loadComponent: () => import('@core/pages/book-viewer/book-viewer.component')

.then(m => m.BookViewerComponent),

},

{

path: `${CONSTANTS.ENDPOINTS.AUTH.PATH}/${CONSTANTS.ENDPOINTS.AUTH.SIGN\_IN}`,

canActivate: [loginGuard],

loadComponent: () => import('@core/pages/login/login.component')

.then(m => m.LoginComponent),

},

{

path: CONSTANTS.ENDPOINTS.NO\_CONNECTION,

loadComponent: () =>

import('./core/pages/no-connection/no-connection.component').then(

(m) => m.NoConnectionComponent

),

},

];

Файл book-file-type.ts:

import { BookFileType } from './dtos/BookManager.Domain.Enums';

export function getBookFileType(fileTypeString: string): BookFileType {

switch (fileTypeString) {

case 'application/pdf':

case '.pdf':

return BookFileType.Pdf;

case 'application/epub':

case '.epub':

return BookFileType.Epub;

default:

throw new Error(`This file type is not supported! ${fileTypeString}`);

}

}

Файл book-collection-list.component.html:

@for (collection of bookCollections(); track collection.name) {

<app-book-collection-list-item [bookCollection]="collection"

(bookOpenBtnClickEvent)="emitBookOpenClickEvent($event)"

(bookInfoBtnClickEvent)="emitBookInfoClickEvent($event)"

(editBtnClickEvent)="emitEditEvent(collection)"

(deleteBtnClickEvent)="emitDeleteEvent(collection)">

</app-book-collection-list-item>

}

Файл book-collection-list.component.ts:

import { ChangeDetectionStrategy, Component, input, output } from '@angular/core';

import { BookCollectionDto, BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { BookCollectionListItemComponent } from '@core/components/book-collection-list-item/book-collection-list-item.component';

@Component({

selector: 'app-book-collection-list',

standalone: true,

imports: [

BookCollectionListItemComponent,

],

templateUrl: './book-collection-list.component.html',

styleUrl: './book-collection-list.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookCollectionListComponent {

public bookCollections = input.required<BookCollectionDto[]>();

public bookInfoClickEvent = output<BookDto>();

public bookOpenClickEvent = output<BookDto>();

public collectionEditClickEvent = output<BookCollectionDto>();

public collectionDeleteClickEvent = output<BookCollectionDto>();

public emitEditEvent(collection: BookCollectionDto): void {

this.collectionEditClickEvent.emit(collection);

}

public emitDeleteEvent(collection: BookCollectionDto): void {

this.collectionDeleteClickEvent.emit(collection);

}

public emitBookInfoClickEvent(book: BookDto): void {

this.bookInfoClickEvent.emit(book);

}

public emitBookOpenClickEvent(book: BookDto): void {

this.bookOpenClickEvent.emit(book);

}

}

Файл book-collection-list-item.component.html:

<section class="collection-title-container">

<h3 class="title">{{ bookCollection().name }}</h3>

<button mat-icon-button [matMenuTriggerFor]="menu">

<mat-icon>more\_vert</mat-icon>

</button>

</section>

<section class="collection-grid">

@for (book of bookCollection().books; track $index) {

<app-book-grid-item [bookItem]="book"

[isEditButtonVisible]="false"

[isDeleteButtonVisible]="false"

(openEvent)="emitBookOpenBtnClickEvent(book)"

(infoEvent)="emitBookInfoBtnClickEvent(book)">

</app-book-grid-item>

}

</section>

<mat-menu #menu="matMenu">

<button mat-menu-item (click)="emitEditBtnClickEvent()">Изменить</button>

<button mat-menu-item (click)="emitDeleteBtnClickEvent()">Удалить</button>

</mat-menu>

Файл book-collection-list-item.component.ts:

import { ChangeDetectionStrategy, Component, computed, input, output } from '@angular/core';

import { BookCollectionDto, BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { BookGridItemComponent } from '@core/components/book-grid-item/book-grid-item.component';

import { MatIconButton } from '@angular/material/button';

import { MatIcon } from '@angular/material/icon';

import { MatMenu, MatMenuItem, MatMenuTrigger } from '@angular/material/menu';

@Component({

selector: 'app-book-collection-list-item',

standalone: true,

imports: [

BookGridItemComponent,

MatIconButton,

MatIcon,

MatMenu,

MatMenuTrigger,

MatMenuItem,

],

templateUrl: './book-collection-list-item.component.html',

styleUrl: './book-collection-list-item.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush

})

export class BookCollectionListItemComponent {

public bookCollection = input.required<BookCollectionDto>();

public bookOpenBtnClickEvent = output<BookDto>();

public bookInfoBtnClickEvent = output<BookDto>();

public editBtnClickEvent = output<void>();

public deleteBtnClickEvent = output<void>();

public emitBookOpenBtnClickEvent(book: BookDto): void {

this.bookOpenBtnClickEvent.emit(book);

}

public emitBookInfoBtnClickEvent(book: BookDto): void {

this.bookInfoBtnClickEvent.emit(book);

}

public emitEditBtnClickEvent(): void {

this.editBtnClickEvent.emit();

}

public emitDeleteBtnClickEvent(): void {

this.deleteBtnClickEvent.emit();

}

}

Файл book-grid-item.component.html:

<img [src]="bookItem().documentDetails.thumbnailUrl" alt="No thumbnail" />

<div class="info-container">

<span class="title">

{{ bookItem().documentDetails.title }}

</span>

<a

mat-icon-button

(click)="$event.stopPropagation()"

[matMenuTriggerFor]="actionMenu"

>

<mat-icon>more\_vert</mat-icon>

</a>

</div>

<mat-menu #actionMenu>

@if (isInfoButtonVisible()) {

<button mat-menu-item (click)="emitInfoClickEvent()">

<mat-icon>info</mat-icon>

Информация о книге

</button>

}

@if (isEditButtonVisible()) {

<button mat-menu-item (click)="emitEditEvent()">

<mat-icon>edit</mat-icon>

Редактировать детали

</button>

}

@if (isDeleteButtonVisible()) {

<button mat-menu-item (click)="emitDeleteEvent()">

<mat-icon>delete</mat-icon>

Удалить

</button>

}

</mat-menu>

Файл book-grid-item.component.ts:

import {

ChangeDetectionStrategy,

Component,

HostListener,

input,

output,

} from '@angular/core';

import { BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { MatIconAnchor } from '@angular/material/button';

import { MatMenu, MatMenuItem, MatMenuTrigger } from '@angular/material/menu';

import { MatIcon } from '@angular/material/icon';

import { NgOptimizedImage } from '@angular/common';

@Component({

selector: 'app-book-grid-item',

templateUrl: './book-grid-item.component.html',

styleUrl: './book-grid-item.component.scss',

standalone: true,

imports: [

MatIconAnchor,

MatMenuTrigger,

MatIcon,

MatMenu,

NgOptimizedImage,

MatMenuItem,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookGridItemComponent {

public bookItem = input.required<BookDto>();

public isEditButtonVisible = input<boolean>(true);

public isDeleteButtonVisible = input<boolean>(true);

public isInfoButtonVisible = input<boolean>(true);

public openEvent = output();

public infoEvent = output();

public editEvent = output();

public deleteEvent = output();

@HostListener('click')

public emitOpenEvent(): void {

this.openEvent.emit();

}

public emitEditEvent(): void {

this.editEvent.emit();

}

public emitDeleteEvent(): void {

this.deleteEvent.emit();

}

public emitInfoClickEvent() {

this.infoEvent.emit();

}

}

Файл book-grid-view.component.html:

@for (book of books(); track $index) {

<app-book-grid-item

[bookItem]="book"

(openEvent)="emitOpenItemEvent(book)"

(infoEvent)="emitInfoItemEvent(book)"

(editEvent)="emitEditItemEvent(book)"

(deleteEvent)="emitDeleteItemEvent(book)"

></app-book-grid-item>

}

<ng-content></ng-content>

Файл book-grid-view.component.ts:

import {

AfterViewInit,

ChangeDetectionStrategy,

Component,

DestroyRef,

ElementRef,

EventEmitter,

Output,

input,

output,

} from '@angular/core';

import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

import { BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { debounceTime, fromEvent } from 'rxjs';

import { BookGridItemComponent } from '@core/components/book-grid-item/book-grid-item.component';

@Component({

selector: 'app-book-grid-view',

templateUrl: './book-grid-view.component.html',

styleUrl: './book-grid-view.component.scss',

standalone: true,

imports: [

BookGridItemComponent,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookGridViewComponent implements AfterViewInit {

protected readonly RESIZE\_DEBOUNCE\_TIME = 200;

public books = input.required<BookDto[]>();

public openItemEvent = output<BookDto>();

public infoItemEvent = output<BookDto>();

public editItemEvent = output<BookDto>();

public deleteItemEvent = output<BookDto>();

@Output()

public numOfVisibleItemsChangeEvent = new EventEmitter<number>();

constructor(

private readonly \_hostElement: ElementRef,

private readonly \_destroyRef: DestroyRef

) {}

public ngAfterViewInit(): void {

fromEvent(window, 'resize')

.pipe(

debounceTime(this.RESIZE\_DEBOUNCE\_TIME),

takeUntilDestroyed(this.\_destroyRef)

)

.subscribe(() =>

this.numOfVisibleItemsChangeEvent.emit(this.\_calculateGridItemCount())

);

this.numOfVisibleItemsChangeEvent.emit(this.\_calculateGridItemCount());

}

public emitOpenItemEvent(book: BookDto): void {

this.openItemEvent.emit(book);

}

public emitInfoItemEvent(book: BookDto) {

this.infoItemEvent.emit(book);

}

public emitEditItemEvent(book: BookDto): void {

this.editItemEvent.emit(book);

}

public emitDeleteItemEvent(book: BookDto): void {

this.deleteItemEvent.emit(book);

}

private \_calculateGridItemCount(): number {

const hostStyles = getComputedStyle(this.\_hostElement.nativeElement);

const rowCount = hostStyles.gridTemplateRows.split(' ').length;

const colCount = hostStyles.gridTemplateColumns.split(' ').length;

return rowCount \* colCount;

}

}

Файл book-list-item.component.html:

<div class="thumbnail-container">

<img [ngSrc]="image()" alt="No thumbnail" fill />

@if (showProgression && progressionValue()) {

<mat-progress-bar [value]="progressionValue()"></mat-progress-bar>

}

</div>

<div class="info" [class.handset]="isHandset()">

<div class="book-title">{{ bookItem().documentDetails.title || '-' }}</div>

@if (!isHandset()) {

<div class="property-item">

<span class="name">Автор(ы)</span>

<span class="value">{{ bookItem().documentDetails.authors || '-' }}</span>

</div>

<div class="property-item">

<span class="name">Издатель</span>

<span class="value">{{ bookItem().documentDetails.publisherName || '-' }}</span>

</div>

<div class="property-item">

<span class="name">Общее время</span>

<span class="value">{{ totalReadingTime() }}</span>

</div>

<div class="property-item">

<span class="name">Посл. время открытия</span>

<span class="value">{{ recentAccessTime() }}</span>

</div>

<div class="actions">

@if (showEditButton) {

<button mat-icon-button (click)="handleEditEvent($event)">

<mat-icon>edit</mat-icon>

</button>

}

@if (showDeleteButton) {

<button mat-icon-button (click)="handleDeleteEvent($event)">

<mat-icon>delete</mat-icon>

</button>

}

</div>

} @else {

<div class="actions" [class.handset]="isHandset()">

<button mat-icon-button (click)="handleInfoClickEvent($event)">

<mat-icon>info</mat-icon>

</button>

@if (showEditButton) {

<button mat-icon-button (click)="handleEditEvent($event)">

<mat-icon>edit</mat-icon>

</button>

}

@if (showDeleteButton) {

<button mat-icon-button (click)="handleDeleteEvent($event)">

<mat-icon>delete</mat-icon>

</button>

}

</div>

}

</div>

Файл book-list-item.component.ts:

import {

ChangeDetectionStrategy,

Component,

computed,

EventEmitter,

Input,

input, output,

Output,

} from '@angular/core';

import { DomSanitizer } from '@angular/platform-browser';

import { BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { format, formatDuration, intervalToDuration, secondsToMinutes } from 'date-fns';

import { MatProgressBar } from '@angular/material/progress-bar';

import { MatIconButton } from '@angular/material/button';

import { MatIcon } from '@angular/material/icon';

import { NgOptimizedImage } from '@angular/common';

import { BreakpointObserver, Breakpoints } from '@angular/cdk/layout';

import { toSignal } from '@angular/core/rxjs-interop';

import { map } from 'rxjs';

import { ru } from 'date-fns/locale';

@Component({

selector: 'app-book-list-item',

templateUrl: './book-list-item.component.html',

styleUrl: './book-list-item.component.scss',

standalone: true,

imports: [

MatProgressBar,

MatIconButton,

MatIcon,

NgOptimizedImage,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookListItemComponent {

protected readonly EMPTY\_PLACEHOLDER = '-';

@Input()

public showEditButton: boolean = true;

@Input()

public showDeleteButton: boolean = true;

@Input()

public showProgression: boolean = true;

public readonly infoClickEvent = output();

public readonly editClickEvent = output();

public readonly deleteClickEvent = output();

public bookItem = input.required<BookDto>();

public image = computed(() => {

return (

this.\_domSanitizer.bypassSecurityTrustResourceUrl(

`${this.bookItem().documentDetails.thumbnailUrl}`,

) ?? '/assets/images/noimage.png'

);

});

public progressionValue = computed(() => {

const item = this.bookItem();

const lastViewedPage = item.stats?.lastViewedPage ?? 0;

const pageCount = item.documentDetails.pageCount ?? 0;

return Math.round((lastViewedPage / pageCount) \* 100);

});

public totalReadingTime = computed(() => {

const item = this.bookItem();

const totalReadingTimeInSec = item.stats?.totalReadingTime ?? 0;

if (totalReadingTimeInSec === 0) {

return '-';

}

const duration = intervalToDuration({ start: 0, end: totalReadingTimeInSec \* 1000 });

return formatDuration(duration, { zero: true, locale: ru });

});

public recentAccessTime = computed(() => {

const item = this.bookItem();

if (!item.stats || !item.stats.recentAccessTime)

return this.EMPTY\_PLACEHOLDER;

return format(item.stats.recentAccessTime, 'dd-MM-yyyy HH:mm');

});

public isHandset = toSignal(this.\_breakpointObserver.observe([Breakpoints.Handset])

.pipe(map((result) => result.matches)));

constructor(

private readonly \_breakpointObserver: BreakpointObserver,

private readonly \_domSanitizer: DomSanitizer,

) {

}

public handleEditEvent(event: MouseEvent): void {

event.stopPropagation();

this.editClickEvent.emit();

}

public handleDeleteEvent(event: MouseEvent): void {

event.stopPropagation();

this.deleteClickEvent.emit();

}

public handleInfoClickEvent(event: MouseEvent): void {

event.stopPropagation();

this.infoClickEvent.emit();

}

}

Файл book-list-view.component.html:

<mat-action-list id="book-list" [style.gap.px]="LIST\_ITEM\_GAP\_PX">

@for (book of books(); track $index) {

<button mat-list-item (click)="handleOpenItemEvent(book)">

<app-book-list-item

matTooltipPosition="above"

[style.height.px]="LIST\_ITEM\_HEIGHT\_PX"

[bookItem]="book"

[matTooltip]="book.documentDetails.title ?? ''"

[showEditButton]="showEditButton()"

[showDeleteButton]="showDeleteButton()"

(infoClickEvent)="handleInfoItemEvent(book)"

(editClickEvent)="handleEditItemEvent(book)"

(deleteClickEvent)="handleDeleteItemEvent(book)"

>

</app-book-list-item>

</button>

}

</mat-action-list>

Файл book-list-view.component.ts:

import {

AfterViewInit,

ChangeDetectionStrategy,

Component,

DestroyRef,

ElementRef,

input,

output,

} from '@angular/core';

import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

import { BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { debounceTime, fromEvent } from 'rxjs';

import { MatActionList, MatListItem } from '@angular/material/list';

import { MatTooltip } from '@angular/material/tooltip';

import { BookListItemComponent } from '@core/components/book-list-item/book-list-item.component';

@Component({

selector: 'app-book-list-view',

templateUrl: './book-list-view.component.html',

styleUrl: './book-list-view.component.scss',

standalone: true,

imports: [

MatActionList,

MatListItem,

MatTooltip,

BookListItemComponent,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookListViewComponent implements AfterViewInit {

public showEditButton = input(true);

public showDeleteButton = input(true);

public books = input.required<BookDto[]>();

public openItemEvent = output<BookDto>();

public infoItemEvent = output<BookDto>();

public editItemEvent = output<BookDto>();

public deleteItemEvent = output<BookDto>();

public numOfVisibleItemsChangeEvent = output<number>();

protected readonly RESIZE\_DEBOUNCE\_TIME = 200;

protected readonly LIST\_ITEM\_HEIGHT\_PX = 130;

protected readonly LIST\_ITEM\_GAP\_PX = 14;

constructor(

private readonly \_hostElement: ElementRef,

private readonly \_destroyRef: DestroyRef,

) {

}

public ngAfterViewInit(): void {

fromEvent(window, 'resize')

.pipe(

debounceTime(this.RESIZE\_DEBOUNCE\_TIME),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe(() =>

this.numOfVisibleItemsChangeEvent.emit(this.\_countVisibleItems()),

);

this.numOfVisibleItemsChangeEvent.emit(this.\_countVisibleItems());

console.log(this.\_countVisibleItems());

}

public handleOpenItemEvent(book: BookDto): void {

this.openItemEvent.emit(book);

}

public handleInfoItemEvent(book: BookDto): void {

this.infoItemEvent.emit(book);

}

public handleDeleteItemEvent(book: BookDto): void {

this.deleteItemEvent.emit(book);

}

public handleEditItemEvent(book: BookDto) {

this.editItemEvent.emit(book);

}

private \_countVisibleItems(): number {

const hostStyles = getComputedStyle(this.\_hostElement.nativeElement);

const height = parseInt(hostStyles.getPropertyValue('height'));

return Math.round(

height / (this.LIST\_ITEM\_HEIGHT\_PX + this.LIST\_ITEM\_GAP\_PX),

);

}

}

Файл icon-button.component.ts:

import { ChangeDetectionStrategy, Component, input } from '@angular/core';

import { MatRipple } from '@angular/material/core';

import { MatIcon } from '@angular/material/icon';

@Component({

selector: 'button[app-icon-button]',

templateUrl: './icon-button.component.html',

styleUrl: './icon-button.component.scss',

standalone: true,

imports: [MatIcon],

changeDetection: ChangeDetectionStrategy.OnPush,

hostDirectives: [MatRipple],

})

export class IconButtonComponent {

public active = input<boolean>(false);

}

Файл main-layout.component.html:

<mat-sidenav-container>

@if (!isHandset() && isSignedIn()) {

<mat-sidenav

[mode]="!isHandset() ? 'side' : 'over'"

[class.expanded]="isExpanded"

[fixedInViewport]="true"

[(opened)]="isExpanded"

>

<div class="internal-sidenav">

<mat-nav-list>

<a mat-list-item

matTooltip="Раскрыть панель"

matTooltipPosition="right"

[matTooltipDisabled]="isExpanded"

(click)="toggleSidenav()">

<mat-icon matListItemIcon>menu</mat-icon>

<div matListItemTitle \*ngIf="isExpanded">Закрыть панель</div>

</a>

@for (route of routeLinks; track $index) {

<a

mat-list-item

routerLinkActive="active"

matTooltipPosition="right"

[matTooltip]="route.name"

[matTooltipDisabled]="isExpanded"

[routerLinkActiveOptions]="{ exact: true }"

[routerLink]="route.link"

>

<mat-icon matListItemIcon>{{ route.iconCode }}</mat-icon>

<div matListItemTitle \*ngIf="isExpanded">{{ route.name }}</div>

</a>

}

</mat-nav-list>

<div class="spacer"></div>

<mat-nav-list>

<a mat-list-item

matTooltip="Выйти из системы"

matTooltipPosition="right"

[matTooltipDisabled]="isExpanded"

(click)="signOut()">

<mat-icon matListItemIcon>logout</mat-icon>

<div matListItemTitle \*ngIf="isExpanded">Выйти из системы</div>

</a>

</mat-nav-list>

</div>

</mat-sidenav>

}

<mat-sidenav-content

[class.handset]="isHandset()"

[class.expanded]="!isHandset() && isSignedIn() && isExpanded"

>

<ng-content></ng-content>

</mat-sidenav-content>

</mat-sidenav-container>

Файл main-layout.component.ts:

import { ChangeDetectionStrategy, Component } from '@angular/core';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Router } from '@angular/router';

import { CONSTANTS } from '@core/constants';

import { AuthService } from '@core/services/auth.service';

import { toSignal } from '@angular/core/rxjs-interop';

import { BreakpointObserver, Breakpoints } from '@angular/cdk/layout';

import { map } from 'rxjs';

@Component({

selector: 'app-main-layout',

templateUrl: './main-layout.component.html',

styleUrl: './main-layout.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class MainLayoutComponent {

public routeLinks = [

{

iconCode: 'schedule',

name: 'Недавние',

link: CONSTANTS.ENDPOINTS.RECENT\_BOOKS,

},

{

iconCode: 'library\_books',

name: 'Все книги',

link: CONSTANTS.ENDPOINTS.EXPLORER,

},

{

iconCode: 'category',

name: 'Коллекции',

link: CONSTANTS.ENDPOINTS.BOOK\_COLLECTIONS,

},

];

public isExpanded = false;

public isHandset = toSignal(this.\_breakpointObserver.observe([Breakpoints.Handset])

.pipe(map((result) => result.matches)));

public isSignedIn = toSignal(this.\_authService.isSignedIn$);

constructor(

private readonly \_authService: AuthService,

private readonly \_snackBar: MatSnackBar,

private readonly \_breakpointObserver: BreakpointObserver,

private readonly \_router: Router,

) {

}

public toggleSidenav(): void {

this.isExpanded = !this.isExpanded;

}

public signOut(): void {

this.\_authService.signOut()

.subscribe(() => {

this.\_snackBar.open('Вы вышли из системы.', 'OK', { duration: 1500 });

this.\_router.navigate([

CONSTANTS.ENDPOINTS.AUTH.PATH,

CONSTANTS.ENDPOINTS.AUTH.SIGN\_IN,

]);

});

}

}

Файл sort-menu.component.html:

<button

mat-icon-button

[matMenuTriggerFor]="sortMenu"

matTooltip="Меню сортировки по критерию"

>

<mat-icon>sort</mat-icon>

</button>

<button

mat-icon-button

(click)="changeSortOrder()"

matTooltip="Порядок сортировки"

>

<mat-icon> @if(sortOrder()) { north } @else { south } </mat-icon>

</button>

<mat-menu #sortMenu>

@for (option of sortOptions(); track $index) {

<button

mat-menu-item

class="sort-option"

(click)="onSortOptionSelected(option)"

>

<mat-icon> @if (sortOption() === option) { check } </mat-icon>

<span>{{ option.name }}</span>

</button>

}

</mat-menu>

Файл sort-menu.component.ts:

import {

ChangeDetectionStrategy,

Component,

input,

output,

} from '@angular/core';

import { SortOrder } from '@core/dtos/BookManager.Application.Common.DTOs';

import { MatIcon } from '@angular/material/icon';

import { MatIconButton } from '@angular/material/button';

import { MatTooltip } from '@angular/material/tooltip';

import { MatMenu, MatMenuItem, MatMenuTrigger } from '@angular/material/menu';

export interface SortOption {

value: string;

name: string;

}

@Component({

selector: 'app-sort-menu',

templateUrl: './sort-menu.component.html',

styleUrl: './sort-menu.component.scss',

standalone: true,

imports: [

MatIcon,

MatIconButton,

MatTooltip,

MatMenu,

MatMenuTrigger,

MatMenuItem,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class SortMenuComponent {

public sortOptions = input.required<SortOption[]>();

public sortOption = input<SortOption>();

public sortOrder = input<SortOrder>(SortOrder.Asc);

public sortOptionChange = output<SortOption>();

public sortOrderChange = output<SortOrder>();

public onSortOptionSelected(option: SortOption): void {

this.sortOptionChange.emit(option);

}

public changeSortOrder(): void {

this.sortOrderChange.emit(

this.sortOrder() === SortOrder.Asc ? SortOrder.Desc : SortOrder.Asc,

);

}

}

Файл toolbar.component.ts:

import { ChangeDetectionStrategy, Component } from '@angular/core';

@Component({

selector: 'app-toolbar',

templateUrl: './toolbar.component.html',

styleUrl: './toolbar.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class ToolbarComponent { }

Файл tooltip-menu.component.html:

<div [cdkMenuTriggerFor]="contextMenu"></div>

<ng-content></ng-content>

<ng-template #contextMenu>

<div class="context-menu-container">

<div class="tooltip-menu">

<button matTooltip="Перевести"

matTooltipPosition="above"

(click)="emitTextCopyEvent()">

<mat-icon>content\_copy</mat-icon>

</button>

<button matTooltip="Перевести"

matTooltipPosition="above"

(click)="emitTranslationBtnClickEvent()">

<mat-icon>translate</mat-icon>

</button>

<button matTooltip="Определение"

matTooltipPosition="above"

[disabled]="definitionLoading()"

(mouseup)="emitDefinitionBtnClickEvent()">

<mat-icon>abc</mat-icon>

</button>

<button matTooltip="Обобщение текста"

matTooltipPosition="above"

(click)="emitTextSumBtnClickEvent()">

<mat-icon>functions</mat-icon>

</button>

</div>

@if (isDefinitionMenuOpen()) {

<div class="word-definition-popup">

<div class="word-definition-container">

<mat-form-field>

<mat-select [(ngModel)]="selectedDefinitionProvider">

@for (provider of definitionProviders(); track $index) {

<mat-option [value]="provider">{{ provider }}</mat-option>

}

</mat-select>

</mat-form-field>

@if (!haveDefinitions()) {

<span>Нет результатов для данного определения</span>

}

@for (entry of wordDefinitionEntries(); track $index) {

<div class="word">{{ entry.word }}</div>

<div class="transcription">{{ entry.transcription }}</div>

<ul>

@for (definition of entry.definitions; track $index) {

<li>

<span class="part-of-speech">{{ definition.partOfSpeech }}</span>

<span class="definition">{{ definition.definition }}</span>

</li>

}

</ul>

}

</div>

<div>

<button mat-flat-button

color="primary"

[disabled]="definitionLoading() || !haveDefinitions()"

(click)="emitWordAddBtnClickEvent()">

<mat-icon>add</mat-icon>

Добавить в словарь

</button>

</div>

<app-loading-spinner-overlay [loading]="definitionLoading()"></app-loading-spinner-overlay>

</div>

}

</div>

</ng-template>

Файл tooltip-menu.component.ts:

import { CdkMenuTrigger } from '@angular/cdk/menu';

import {

ChangeDetectionStrategy,

Component,

computed,

HostListener,

input,

model,

output,

viewChild,

} from '@angular/core';

import { MatTooltip } from '@angular/material/tooltip';

import { MatIcon } from '@angular/material/icon';

import { WordDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { MatButtonModule } from '@angular/material/button';

import { LoadingSpinnerOverlayComponent } from '@shared/components/loading-spinner-overlay/loading-spinner-overlay.component';

import { MatFormField } from '@angular/material/form-field';

import { MatOption, MatSelect } from '@angular/material/select';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

@Component({

selector: '[app-tooltip-menu]',

templateUrl: './tooltip-menu.component.html',

styleUrl: './tooltip-menu.component.scss',

standalone: true,

imports: [

CdkMenuTrigger,

MatTooltip,

MatIcon,

MatButtonModule,

LoadingSpinnerOverlayComponent,

MatFormField,

MatSelect,

MatOption,

ReactiveFormsModule,

FormsModule,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TooltipMenuComponent {

protected readonly WORD\_BREAK\_PATTERN = new RegExp(`(-)\n`, 'gm');

protected readonly NEW\_LINE\_PATTERN = new RegExp(`(\n|\r\n|\r)`, 'gm');

protected readonly WORD\_DEFINITION\_POPUP\_HEIGHT\_PX = 240;

public contextMenuTrigger = viewChild(CdkMenuTrigger);

public selectedDefinitionProvider = model<string | null>(null);

public wordDefinitionEntries = input<WordDto[]>([]);

public haveDefinitions = computed(() => {

const entries = this.wordDefinitionEntries();

return entries.length > 0;

});

public definitionLoading = input<boolean>(false);

public definitionProviders = input<string[]>([]);

public isDefinitionMenuOpen = input<boolean>(false);

public selectionEvent = output<string | null>();

public textCopyEvent = output<string>();

public translationBtnClickEvent = output<string>();

public definitionBtnClickEvent = output<string>();

public textSumBtnClickEvent = output<string>();

public definitionAddBtnClickEvent = output<WordDto[]>();

private \_selectedText?: string;

constructor(

private readonly \_window: Window,

) {

}

@HostListener('mousedown')

public clearSelection(): void {

const selection = this.\_window.getSelection();

this.contextMenuTrigger()?.close();

selection?.removeAllRanges();

this.selectionEvent.emit(null);

}

@HostListener('mouseup', ['$event'])

public showTooltipMenu(e: MouseEvent) {

const selection = this.\_window.getSelection();

const selectedText = selection?.toString().trim();

const contextMenuTrigger = this.contextMenuTrigger()!;

this.selectionEvent.emit(selectedText ?? null);

if (!selection || !selectedText || selection.rangeCount === 0) {

return;

}

if (!contextMenuTrigger.menuPosition) {

contextMenuTrigger.menuPosition = [];

}

contextMenuTrigger.menuPosition.splice(

0,

contextMenuTrigger.menuPosition.length,

);

contextMenuTrigger.menuPosition.push(

{

originX: 'center',

originY: 'top',

overlayX: 'center',

overlayY: 'top',

offsetX: e.clientX,

offsetY: e.clientY,

},

{

originX: 'start',

originY: 'top',

overlayX: 'center',

overlayY: 'bottom',

offsetX: e.clientX,

offsetY: e.clientY,

},

);

this.\_selectedText = this.\_processText(selectedText);

contextMenuTrigger.open();

}

@HostListener('scroll', ['$event'])

@HostListener('wheel', ['$event'])

public closeContextMenu(): void {

if (this.contextMenuTrigger()?.isOpen()) {

this.contextMenuTrigger()?.close();

}

}

public willBeOffscreenByHeight(e: MouseEvent): boolean {

return e.clientY + this.WORD\_DEFINITION\_POPUP\_HEIGHT\_PX > window.innerHeight;

}

public emitTranslationBtnClickEvent(): void {

if (!this.\_selectedText) return;

this.translationBtnClickEvent.emit(this.\_selectedText);

this.contextMenuTrigger()?.close();

}

public emitTextCopyEvent(): void {

if (!this.\_selectedText) return;

this.textCopyEvent.emit(this.\_selectedText);

this.contextMenuTrigger()?.close();

}

public emitTextSumBtnClickEvent(): void {

if (!this.\_selectedText) return;

this.textSumBtnClickEvent.emit(this.\_selectedText);

}

public emitDefinitionBtnClickEvent(): void {

if (!this.\_selectedText) return;

this.definitionBtnClickEvent.emit(this.\_selectedText);

}

public emitWordAddBtnClickEvent(): void {

if (!this.\_selectedText) return;

this.definitionAddBtnClickEvent.emit(this.wordDefinitionEntries());

this.contextMenuTrigger()?.close();

}

private \_processText(value: string): string {

return value.trim()

}

}

Файл book-collection-edit-dialog.component.html:

<h2 mat-dialog-title>{{ dialogTitle() }}</h2>

<div mat-dialog-content class="book-add-form">

<mat-form-field>

<mat-label>Название коллекции</mat-label>

<input matInput type="text" [formControl]="nameFormControl" />

@if (nameFormControl.invalid) {

<mat-error>{{ errorMessage }}</mat-error>

}

</mat-form-field>

<div class="container">

<app-loading-spinner-overlay [loading]="isBooksLoading()"></app-loading-spinner-overlay>

<div class="books-table-container">

<table mat-table [dataSource]="books">

<ng-container matColumnDef="select">

<th mat-header-cell \*matHeaderCellDef>

<mat-checkbox (change)="$event ? toggleAllRows() : null"

[checked]="selection.hasValue() && isAllSelected()"

[indeterminate]="selection.hasValue() && !isAllSelected()"

[aria-label]="checkboxLabel(0)">

</mat-checkbox>

</th>

<td mat-cell \*matCellDef="let row; let index = index">

<mat-checkbox (click)="$event.stopPropagation()"

(change)="$event ? toggleBookItem(row) : null"

[checked]="isBookSelected(row)"

[aria-label]="checkboxLabel(index, row)">

</mat-checkbox>

</td>

</ng-container>

<ng-container matColumnDef="title">

<th mat-header-cell [style.width]="'400px'" \*matHeaderCellDef>Название книги</th>

<td mat-cell \*matCellDef="let book"> {{ book.documentDetails.title }}</td>

</ng-container>

<ng-container matColumnDef="authors">

<th mat-header-cell \*matHeaderCellDef>Авторы</th>

<td mat-cell \*matCellDef="let book"> {{ book.documentDetails.authors }}</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns; sticky: true"></tr>

<tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

</table>

</div>

<mat-paginator [length]="itemCount"

[pageSize]="10"

[pageSizeOptions]="pageSizeOptions"

showFirstLastButtons>

</mat-paginator>

</div>

</div>

<mat-dialog-actions>

<button mat-flat-button

color="primary"

[disabled]="nameFormControl.invalid"

(click)="save()">

Сохранить

</button>

<button mat-flat-button

color="warn"

(click)="close()">

Закрыть

</button>

</mat-dialog-actions>

Файл book-collection-edit-dialog.component.ts:

import {

AfterViewInit,

ChangeDetectionStrategy, ChangeDetectorRef,

Component,

computed,

DestroyRef,

Inject, OnInit,

signal,

ViewChild,

} from '@angular/core';

import { BookCollectionDto, BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { MAT\_DIALOG\_DATA, MatDialogActions, MatDialogModule, MatDialogRef } from '@angular/material/dialog';

import { MatIconModule } from '@angular/material/icon';

import { FormBuilder, ReactiveFormsModule, ValidatorFn, Validators } from '@angular/forms';

import { MatFormFieldModule } from '@angular/material/form-field';

import { MatInputModule } from '@angular/material/input';

import { DigitOnlyModule } from '@uiowa/digit-only';

import { MatTooltipModule } from '@angular/material/tooltip';

import { MatButtonModule } from '@angular/material/button';

import { MatListModule } from '@angular/material/list';

import { BookService } from '@core/services/book.service';

import { MatTableModule } from '@angular/material/table';

import { MatPaginator, MatPaginatorModule } from '@angular/material/paginator';

import { startWith, switchMap } from 'rxjs';

import { MatCheckbox } from '@angular/material/checkbox';

import { SelectionModel } from '@angular/cdk/collections';

import { MatProgressBar } from '@angular/material/progress-bar';

import { LoadingSpinnerOverlayComponent } from '@shared/components/loading-spinner-overlay/loading-spinner-overlay.component';

import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

import { CONSTANTS } from '@core/constants';

interface DialogData {

mode?: 'create' | 'update';

allBookCollections: BookCollectionDto[];

bookCollection?: Omit<BookCollectionDto, 'id'>;

}

@Component({

selector: 'app-book-collection-edit-dialog',

standalone: true,

imports: [

MatDialogModule,

MatIconModule,

ReactiveFormsModule,

MatFormFieldModule,

MatInputModule,

DigitOnlyModule,

MatTooltipModule,

MatDialogActions,

MatButtonModule,

MatListModule,

MatTableModule,

MatPaginatorModule,

MatCheckbox,

MatProgressBar,

LoadingSpinnerOverlayComponent,

],

templateUrl: './book-collection-edit-dialog.component.html',

styleUrl: './book-collection-edit-dialog.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookCollectionEditDialogComponent implements OnInit, AfterViewInit {

protected readonly CREATION\_DIALOG\_TITLE = 'Добавление новой коллекции книг';

protected readonly EDIT\_DIALOG\_TITLE = 'Изменить коллекцию книг';

protected readonly UNIQUENESS\_ERROR\_MESSAGE = 'Коллекция с данным названием уже существует!';

@ViewChild(MatPaginator)

public paginator!: MatPaginator;

public nameFormControl = this.\_fb.nonNullable.control<string>(

'',

[

Validators.required,

],

);

public isEditMode = signal<boolean>(false);

public dialogTitle = computed(() => {

return this.isEditMode()

? this.EDIT\_DIALOG\_TITLE

: this.CREATION\_DIALOG\_TITLE;

});

public isBooksLoading = signal<boolean>(false);

public itemCount = 0;

public pageSizeOptions = [10, 25, 50];

public displayedColumns = ['select', 'title', 'authors'];

public books: BookDto[] = [];

public selection = new SelectionModel<BookDto>(true,

[],

undefined,

(a, b) => a.documentDetails.id === b.documentDetails.id);

public errorMessage = '';

constructor(

@Inject(MAT\_DIALOG\_DATA) private readonly \_data: DialogData,

private readonly \_booksService: BookService,

private readonly \_cdr: ChangeDetectorRef,

private readonly \_fb: FormBuilder,

private readonly \_dialogRef: MatDialogRef<BookCollectionEditDialogComponent>,

private readonly \_destroyRef: DestroyRef,

) {

this.\_initForm();

this.isEditMode.set(this.\_data.mode === 'update');

}

public ngOnInit(): void {

this.nameFormControl.valueChanges

.pipe(takeUntilDestroyed(this.\_destroyRef))

.subscribe(() => this.\_updateErrorMessage());

}

public ngAfterViewInit(): void {

this.paginator.page

.pipe(

startWith({}),

switchMap(() => {

this.isBooksLoading.set(true);

return this.\_booksService.getPage(this.paginator.pageIndex + 1, this.paginator.pageSize);

}),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe((page) => {

this.books = page.items;

this.itemCount = page.totalItemCount;

this.isBooksLoading.set(false);

});

}

public isAllSelected() {

const numSelected = this.selection.selected.length;

const numRows = this.itemCount;

return numSelected === numRows;

}

public toggleAllRows() {

if (!this.selection.selected.length) {

this.selection.select(...this.books);

} else {

this.selection.clear();

}

}

Файл book-edit-dialog.component.html:

<h2 mat-dialog-title>{{ dialogTitle() }}</h2>

<div mat-dialog-content class="book-add-form" [formGroup]="bookForm">

<mat-form-field>

<mat-label>Название книги</mat-label>

<input matInput type="text" formControlName="title" />

</mat-form-field>

<mat-form-field>

<mat-label>Описание</mat-label>

<textarea matInput type="text" formControlName="description"></textarea>

</mat-form-field>

<mat-form-field>

<mat-label>ISBN (Международный стандартный книжный номер)</mat-label>

<input

matInput

type="text"

formControlName="isbn"

placeholder="0123456789"

matTooltip="Должен содержать 10-значное или 13-значное число"

digitOnly

/>

</mat-form-field>

<mat-form-field>

<mat-label>Издатель книги</mat-label>

<input matInput type="text" placeholder="Рога и копыта" formControlName="publisherName" />

</mat-form-field>

<mat-form-field>

<mat-label>Добавить автора книги</mat-label>

<input matInput type="text" [formControl]="authorNameInputControl">

<button mat-icon-button

matSuffix

[disabled]="authorNameInputControl.invalid || authorNameInputControl.pristine"

(click)="addAuthor()">

<mat-icon>add</mat-icon>

</button>

</mat-form-field>

<h3 class="list-title">Список авторов книги</h3>

<mat-list>

@for (author of bookForm.value.authors; track $index) {

<div matListItemLine class="author-item">

<div matListItemTitle class="author-name">{{ author }}</div>

<button mat-icon-button (click)="deleteAuthor($index)">

<mat-icon>delete</mat-icon>

</button>

</div>

}

</mat-list>

</div>

<mat-dialog-actions>

<button mat-flat-button

color="primary"

[disabled]="bookForm.invalid"

(click)="saveBook()">

Сохранить

</button>

<button mat-flat-button

color="warn"

(click)="close()">

Закрыть

</button>

</mat-dialog-actions>

Файл book-edit-dialog.component.ts:

import { ChangeDetectionStrategy, Component, computed, Inject, OnInit, signal } from '@angular/core';

import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';

import { MAT\_DIALOG\_DATA, MatDialogActions, MatDialogModule, MatDialogRef } from '@angular/material/dialog';

import { BookMetadataDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { MatFormFieldModule } from '@angular/material/form-field';

import { MatInputModule } from '@angular/material/input';

import { DigitOnlyModule } from '@uiowa/digit-only';

import { MatTooltipModule } from '@angular/material/tooltip';

import { MatButtonModule } from '@angular/material/button';

import { MatListModule } from '@angular/material/list';

import { MatIconModule } from '@angular/material/icon';

export interface BookEditDialogData {

mode?: 'create' | 'update';

bookDetails: Omit<

BookMetadataDto,

'filename' | 'fileSizeInBytes' | 'fileType'

>;

}

@Component({

selector: 'app-book-edit-dialog',

templateUrl: './book-edit-dialog.component.html',

styleUrl: './book-edit-dialog.component.scss',

standalone: true,

imports: [

MatDialogModule,

MatIconModule,

ReactiveFormsModule,

MatFormFieldModule,

MatInputModule,

DigitOnlyModule,

MatTooltipModule,

MatDialogActions,

MatButtonModule,

MatListModule,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookEditDialogComponent implements OnInit {

protected readonly CREATION\_DIALOG\_TITLE = 'Добавление новой книги';

protected readonly EDIT\_DIALOG\_TITLE = 'Обновление информации о книге';

public isEditMode = signal<boolean>(false);

public dialogTitle = computed(() => {

return this.isEditMode()

? this.EDIT\_DIALOG\_TITLE

: this.CREATION\_DIALOG\_TITLE;

});

public bookForm = this.\_fb.group({

title: this.\_fb.control<string | null>(null),

description: this.\_fb.control<string | null>(null),

isbn: this.\_fb.control<string | null>(null, [

Validators.pattern('^([0-9]{10}|[0-9]{13})|null$'),

]),

publisherName: this.\_fb.control<string | null>(null),

authors: this.\_fb.control<string[]>([]),

});

public authorNameInputControl = this.\_fb.control<string | null>(null);

constructor(

private readonly \_fb: FormBuilder,

private readonly \_dialogRef: MatDialogRef<BookEditDialogComponent>,

@Inject(MAT\_DIALOG\_DATA) public dialogData: BookEditDialogData,

) {

this.\_initForm();

}

public ngOnInit(): void {

this.isEditMode.set(this.dialogData?.mode === 'update');

}

public addAuthor(): void {

if (this.authorNameInputControl.invalid || !this.authorNameInputControl.value) return;

this.bookForm.patchValue({

authors: [

...this.bookForm.value.authors ?? [],

this.authorNameInputControl.value,

],

});

this.authorNameInputControl.reset();

}

public deleteAuthor(index: number): void {

this.bookForm.patchValue({

authors: this.bookForm.value.authors?.filter((\_, i) => i !== index)

});

}

public saveBook(): void {

const values = this.bookForm.value;

if (this.bookForm.invalid) return;

const formData: BookEditDialogData = {

bookDetails: {

title: values.title ?? undefined,

description: values.description ?? undefined,

isbn: values.isbn ?? undefined,

publisherName: values.publisherName ?? undefined,

authors: values.authors ?? undefined,

},

};

this.\_dialogRef.close(formData);

}

public close(): void {

this.\_dialogRef.close();

}

private \_initForm(): void {

const dialogData = this.dialogData?.bookDetails;

if (!dialogData) return;

this.bookForm.setValue({

title: dialogData.title ?? null,

description: dialogData.description ?? null,

isbn: dialogData.isbn ?? null,

publisherName: dialogData.publisherName ?? null,

authors: dialogData.authors ?? [],

});

}

}

Файл book-info-dialog.component.html:

<h2 mat-dialog-title>Информация об электронной книге</h2>

<div mat-dialog-content>

<div class="property-item">

<span class="name">Автор(ы)</span>

<span class="value" [innerText]="bookAuthors"></span>

</div>

<div class="property-item">

<span class="name">Издатель</span>

<span class="value">{{ book.documentDetails.publisherName || '-' }}</span>

</div>

<div class="property-item">

<span class="name">Общее время</span>

<span class="value">{{ totalReadingTime }}</span>

</div>

<div class="property-item">

<span class="name">Посл. время открытия</span>

<span class="value">{{ book.stats?.recentAccessTime | date }}</span>

</div>

</div>

<div mat-dialog-actions>

<button mat-flat-button (click)="close()">

Закрыть

</button>

</div>

Файл book-info-dialog.component.ts:

import { ChangeDetectionStrategy, Component, computed, Inject } from '@angular/core';

import { MAT\_DIALOG\_DATA, MatDialogModule, MatDialogRef } from '@angular/material/dialog';

import { BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { B } from '@angular/cdk/keycodes';

import { DatePipe } from '@angular/common';

import { MatButtonModule } from '@angular/material/button';

import { formatDuration, intervalToDuration } from 'date-fns';

import { ru } from 'date-fns/locale';

interface DialogData {

book: BookDto

}

@Component({

selector: 'app-book-info-dialog',

standalone: true,

imports: [MatDialogModule, MatButtonModule, DatePipe],

templateUrl: './book-info-dialog.component.html',

styleUrl: './book-info-dialog.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush

})

export class BookInfoDialogComponent {

public book: BookDto;

public bookAuthors: string;

public totalReadingTime = '-'

constructor(

@Inject(MAT\_DIALOG\_DATA) public dialogData: DialogData,

private readonly \_dialogRef: MatDialogRef<BookInfoDialogComponent>

) {

this.book = dialogData.book;

this.bookAuthors = dialogData.book.documentDetails.authors?.join('\n') ?? '-'

const totalReadingTimeInSec = this.book.stats?.totalReadingTime ?? 0;

if (totalReadingTimeInSec !== 0) {

const duration = intervalToDuration({ start: 0, end: totalReadingTimeInSec \* 1000 });

this.totalReadingTime = formatDuration(duration, { zero: true, locale: ru });

}

}

public close(): void {

this.\_dialogRef.close();

}

}

Файл delete-confirmation-dialog.component.html:

<h3 mat-dialog-title>Удаление</h3>

<mat-dialog-content>

{{ message }}

</mat-dialog-content>

<mat-dialog-actions>

<button mat-flat-button color="accent" (click)="cancel()">Отмена</button>

<button mat-flat-button color="warn" (click)="confirm()">Да, удалить</button>

</mat-dialog-actions>

Файл delete-confirmation-dialog.component.ts:

import { ChangeDetectionStrategy, Component, Inject } from '@angular/core';

import {

MAT\_DIALOG\_DATA,

MatDialogActions,

MatDialogContent,

MatDialogRef,

MatDialogTitle,

} from '@angular/material/dialog';

import { MatButton } from '@angular/material/button';

interface DialogData {

message?: string;

}

@Component({

selector: 'app-delete-confirmation-dialog',

templateUrl: './delete-confirmation-dialog.component.html',

styleUrl: './delete-confirmation-dialog.component.scss',

standalone: true,

imports: [

MatDialogContent,

MatDialogTitle,

MatButton,

MatDialogActions,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class DeleteConfirmationDialogComponent {

public message?: string;

constructor(

@Inject(MAT\_DIALOG\_DATA) private readonly \_dialogData: DialogData,

private readonly \_dialogRef: MatDialogRef<DeleteConfirmationDialogComponent>,

) {

this.message = this.\_dialogData.message;

}

public confirm(): void {

this.\_dialogRef.close(true);

}

public cancel(): void {

this.\_dialogRef.close(false);

}

}

Файл dictionary-word-edit-dialog.component.html:

﻿<h2 mat-dialog-title>{{ dialogTitle() }}</h2>

<div mat-dialog-content class="word-definition-form" [formGroup]="dictionaryWordEditFormGroup">

<mat-form-field>

<mat-label>Транскрипция</mat-label>

<input matInput formControlName="transcription" />

</mat-form-field>

<mat-form-field>

<mat-label>Код языка</mat-label>

<input matInput formControlName="languageCode" />

</mat-form-field>

<div class="definition-form-control" [formGroup]="definitionFormGroup">

<mat-form-field>

<mat-label>Часть речи</mat-label>

<input matInput type="text" formControlName="" />

</mat-form-field>

</div>

<table mat-table [dataSource]="dictionaryWordEditFormGroup.value.definitions!">

<ng-container matColumnDef="partOfSpeech">

<th mat-header-cell \*matHeaderCellDef>Часть речи</th>

<td mat-cell \*matCellDef="let entry">{{ entry.partOfSpeech }}</td>

</ng-container>

<ng-container matColumnDef="subjectName">

<th mat-header-cell \*matHeaderCellDef>Область деятельности</th>

<td mat-cell \*matCellDef="let entry">{{ entry.subjectName }}</td>

</ng-container>

<ng-container matColumnDef="definition">

<th mat-header-cell \*matHeaderCellDef>Определение</th>

<td mat-cell \*matCellDef="let entry">{{ entry.definition }}</td>

</ng-container>

<tr mat-header-row \*matHeaderRowDef="displayedColumns"></tr>

<tr mat-row \*matRowDef="let row; columns: displayedColumns;"></tr>

</table>

</div>

<div mat-dialog-actions>

<button mat-flat-button

color="primary"

(click)="save()">

Сохранить

</button>

<button mat-flat-button

color="warn"

(click)="close()">

Отмена

</button>

</div>

Файл dictionary-word-edit-dialog.component.ts:

﻿import { ChangeDetectionStrategy, Component, computed, Inject, OnInit, signal } from '@angular/core';

import {

MAT\_DIALOG\_DATA,

MatDialogActions,

MatDialogContent,

MatDialogRef,

MatDialogTitle,

} from '@angular/material/dialog';

import { MatFormField, MatLabel } from '@angular/material/form-field';

import { MatInputModule } from '@angular/material/input';

import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';

import { MatButton } from '@angular/material/button';

import { DigitOnlyModule } from '@uiowa/digit-only';

import { WordDefinitionDto, WordDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { MatListModule } from '@angular/material/list';

import { MatColumnDef, MatTable, MatTableModule } from '@angular/material/table';

interface DialogData {

wordEntry: WordDto;

mode?: 'create' | 'update';

}

@Component({

selector: 'app-dictionary-word-edit-dialog',

standalone: true,

imports: [

MatDialogContent,

MatDialogTitle,

MatFormField,

MatInputModule,

MatLabel,

ReactiveFormsModule,

MatButton,

MatListModule,

MatDialogActions,

MatTableModule,

],

templateUrl: './dictionary-word-edit-dialog.component.html',

styleUrl: './dictionary-word-edit-dialog.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class DictionaryWordEditDialogComponent implements OnInit {

protected readonly CREATION\_DIALOG\_TITLE = 'Добавление слова';

protected readonly EDIT\_DIALOG\_TITLE = 'Обновление определения слова';

public displayedColumns = ['partOfSpeech', 'subjectName', 'definition'];

public isEditMode = signal<boolean>(false);

public dialogTitle = computed(() => {

return this.isEditMode() ? this.EDIT\_DIALOG\_TITLE : this.CREATION\_DIALOG\_TITLE;

});

public dictionaryWordEditFormGroup = this.\_fb.group({

transcription: this.\_fb.control<string | null>(null),

languageCode: this.\_fb.control<string | null>(null),

definitions: this.\_fb.nonNullable.control<WordDefinitionDto[]>([]),

});

public definitionFormGroup = this.\_fb.group({

partOfSpeech: this.\_fb.control<string | null>(null, [Validators.required]),

subjectName: this.\_fb.control<string | null>(null),

definition: this.\_fb.control<string | null>(null, [Validators.required]),

});

constructor(

@Inject(MAT\_DIALOG\_DATA) private readonly \_dialogData: DialogData,

private readonly \_dialogRef: MatDialogRef<DictionaryWordEditDialogComponent>,

private readonly \_fb: FormBuilder,

) {

this.initForm();

}

public ngOnInit(): void {

this.isEditMode.set(this.\_dialogData?.mode === 'update');

}

public initForm(): void {

const wordEntry = this.\_dialogData.wordEntry;

this.dictionaryWordEditFormGroup.patchValue({

transcription: wordEntry.transcription,

languageCode: wordEntry.languageCode,

definitions: wordEntry.definitions,

});

}

public save(): void {

this.\_dialogRef.close();

}

public close(): void {

this.\_dialogRef.close();

}

}

Файл text-sum-dialog.component.html:

<span mat-dialog-title>Результат обобщения текста</span>

<div mat-dialog-content>

<span [innerText]="summarizedText()"></span>

<app-loading-spinner-overlay [loading]="loading()"></app-loading-spinner-overlay>

</div>

<div class="dialog-actions" mat-dialog-actions>

<div>Обобщение текста выполнена моделью нейроной сети от Яндекса. Текст может содержать неточности.</div>

<div class="spacer"></div>

<button mat-flat-button color="warn" (click)="close()">

Закрыть

</button>

</div>

Файл text-sum-dialog.component.ts:

import { ChangeDetectionStrategy, Component, Inject, OnInit, signal } from '@angular/core';

import { MAT\_DIALOG\_DATA, MatDialogModule, MatDialogRef } from '@angular/material/dialog';

import { MatButtonModule } from '@angular/material/button';

import { TextProcessingService } from '@core/services/text-processing.service';

import { catchError, finalize, throwError } from 'rxjs';

import { MatSnackBar, MatSnackBarModule } from '@angular/material/snack-bar';

import { LoadingSpinnerOverlayComponent } from '@shared/components/loading-spinner-overlay/loading-spinner-overlay.component';

interface DialogData {

inputText: string;

}

@Component({

selector: 'app-text-sum-dialog',

standalone: true,

imports: [MatDialogModule, MatButtonModule, MatSnackBarModule, LoadingSpinnerOverlayComponent],

templateUrl: './text-sum-dialog.component.html',

styleUrl: './text-sum-dialog.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TextSumDialogComponent implements OnInit {

public loading = signal<boolean>(false);

public summarizedText = signal<string | null>(null);

constructor(

@Inject(MAT\_DIALOG\_DATA) public dialogData: DialogData,

private readonly \_dialogRef: MatDialogRef<TextSumDialogComponent>,

private readonly \_textProcessingService: TextProcessingService,

private readonly \_snackBar: MatSnackBar,

) {

}

public ngOnInit(): void {

this.loading.set(true);

this.\_textProcessingService.summarizeText({ text: this.dialogData.inputText })

.pipe(

catchError(err => {

this.\_snackBar.open('Произошла ошибка :(', 'OK');

return throwError(() => err);

}),

finalize(() => this.loading.set(false)),

)

.subscribe(response => {

console.log(response);

this.summarizedText.set(response.summarizedText);

});

}

public close(): void {

this.\_dialogRef.close();

}

}

Файл translation-dialog.component.html:

<div mat-dialog-content class="dialog-content" [formGroup]="languageFormGroup">

<div class="language-selector-section">

<div class="language-dropdown-container">

<mat-form-field appearance="outline" subscriptSizing="dynamic">

<mat-label>Язык исходного текста</mat-label>

<mat-select formControlName="sourceLanguage">

@for (language of languages; track $index) {

<mat-option [value]="language.code">{{ language.name }}</mat-option>

}

</mat-select>

</mat-form-field>

</div>

<button mat-icon-button (click)="swapLanguages()">

<mat-icon>swap\_horiz</mat-icon>

</button>

<div class="language-dropdown-container right">

<mat-form-field appearance="outline" subscriptSizing="dynamic">

<mat-label>Язык перевода</mat-label>

<mat-select formControlName="targetLanguage">

@for (language of languages; track $index) {

<mat-option [value]="language.code">{{ language.name }}</mat-option>

}

</mat-select>

</mat-form-field>

</div>

</div>

<div class="section">

<textarea

class="text-container"

cdkTextareaAutosize

formControlName="sourceText"

></textarea>

<div class="translated-text text-container">

{{ translatedText() }}

<app-loading-spinner-overlay [loading]="loading()">

</app-loading-spinner-overlay>

</div>

</div>

</div>

Файл translation-dialog.component.ts:

import { ChangeDetectionStrategy, Component, DestroyRef, Inject, OnInit, signal } from '@angular/core';

import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';

import { MAT\_DIALOG\_DATA, MatDialogContent, MatDialogTitle } from '@angular/material/dialog';

import { LanguageDto, TranslationRequestDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { TextProcessingService } from '@core/services/text-processing.service';

import { finalize } from 'rxjs';

import { MatFormField } from '@angular/material/form-field';

import { MatLabel, MatOption, MatSelect } from '@angular/material/select';

import { CdkTextareaAutosize } from '@angular/cdk/text-field';

import { LoadingSpinnerOverlayComponent } from '@shared/components/loading-spinner-overlay/loading-spinner-overlay.component';

import { MatButtonModule } from '@angular/material/button';

import { MatIcon } from '@angular/material/icon';

interface TranslationDialogComponentData {

sourceText: string;

targetLanguageCode: string;

}

@Component({

selector: 'app-translation-dialog',

templateUrl: './translation-dialog.component.html',

styleUrl: './translation-dialog.component.scss',

standalone: true,

imports: [

MatDialogTitle,

MatDialogContent,

MatLabel,

MatFormField,

MatSelect,

MatOption,

MatButtonModule,

CdkTextareaAutosize,

ReactiveFormsModule,

LoadingSpinnerOverlayComponent,

MatIcon,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class TranslationDialogComponent implements OnInit {

public loading = signal<boolean>(false);

public translatedText = signal<string | null>(null);

public languageFormGroup = this.\_fb.group({

sourceLanguage: this.\_fb.control<string | null>(null, [

Validators.required,

]),

sourceText: this.\_fb.control<string | null>(null, {

updateOn: 'blur',

validators: [Validators.required],

}),

targetLanguage: this.\_fb.control<string | null>(null, [

Validators.required,

]),

});

constructor(

@Inject(MAT\_DIALOG\_DATA) private \_data: TranslationDialogComponentData,

private readonly \_textProcessingService: TextProcessingService,

private readonly \_fb: FormBuilder,

private readonly \_destroyRef: DestroyRef,

) {

this.\_initFormGroup();

}

public get languages(): LanguageDto[] {

return this.\_textProcessingService.availableLanguages();

}

public ngOnInit(): void {

this.\_determineSourceTextLanguage();

this.\_subscribeOnFormChanges();

}

public translateText(): void {

const values = this.languageFormGroup.value;

if (this.languageFormGroup.invalid) return;

const request: TranslationRequestDto = {

sourceLanguage: values.sourceLanguage!,

sourceText: values.sourceText!,

targetLanguage: values.targetLanguage!,

};

this.loading.set(true);

this.\_textProcessingService

.translate(request)

.pipe(finalize(() => this.loading.set(false)))

.subscribe((response) =>

this.translatedText.set(response.translatedText),

);

}

public swapLanguages(): void {

const selectedSourceLangCode = this.languageFormGroup.value.sourceLanguage;

const selectedTargetLangCode = this.languageFormGroup.value.targetLanguage;

this.languageFormGroup.patchValue({

sourceLanguage: selectedTargetLangCode,

targetLanguage: selectedSourceLangCode,

});

}

private \_determineSourceTextLanguage(): void {

this.\_textProcessingService

.detectLanguage({ text: this.\_data.sourceText })

.subscribe((response) =>

this.languageFormGroup.patchValue({

sourceLanguage: response.detectedLanguageCode,

}),

);

}

private \_initFormGroup(): void {

this.languageFormGroup.patchValue({

sourceText: this.\_data.sourceText,

targetLanguage: this.\_data.targetLanguageCode,

});

}

private \_subscribeOnFormChanges() {

this.languageFormGroup.valueChanges

.pipe(takeUntilDestroyed(this.\_destroyRef))

.subscribe(() => this.translateText());

}

}

Файл user-registration-dialog.component.html:

<h2 mat-dialog-title>Создание пользователя</h2>

<mat-dialog-content [formGroup]="createUserForm">

<mat-form-field>

<mat-label>Имя пользователя</mat-label>

<input matInput formControlName="name" />

</mat-form-field>

<mat-form-field>

<mat-label>ПИН-код</mat-label>

<input matInput type="password" formControlName="pinCode" />

</mat-form-field>

</mat-dialog-content>

<mat-dialog-actions>

<button

mat-flat-button

color="primary"

[disabled]="createUserForm.pristine || createUserForm.invalid"

(click)="create()"

inputmode="numeric"

digitOnly

>

Создать

</button>

<button mat-flat-button color="accent" (click)="cancel()">Отмена</button>

</mat-dialog-actions>

Файл user-registration-dialog.component.ts:

import { ChangeDetectionStrategy, Component } from '@angular/core';

import { NonNullableFormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';

import { MatDialogActions, MatDialogContent, MatDialogRef, MatDialogTitle } from '@angular/material/dialog';

import { CONSTANTS } from '@core/constants';

import { MatFormField, MatLabel } from '@angular/material/form-field';

import { MatButton } from '@angular/material/button';

import { DigitOnlyModule } from '@uiowa/digit-only';

import { MatInputModule } from '@angular/material/input';

@Component({

selector: 'app-user-registration-dialog',

templateUrl: './user-registration-dialog.component.html',

styleUrl: './user-registration-dialog.component.scss',

standalone: true,

imports: [

MatDialogContent,

MatDialogTitle,

MatFormField,

MatInputModule,

MatLabel,

ReactiveFormsModule,

MatButton,

DigitOnlyModule,

MatDialogActions,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class UserRegistrationDialogComponent {

public createUserForm = this.\_fb.group({

name: this.\_fb.control('', [Validators.required]),

pinCode: this.\_fb.control('', [

Validators.required,

Validators.pattern(CONSTANTS.REGEX\_PATTERN.PIN\_CODE),

]),

});

constructor(

private readonly \_dialogRef: MatDialogRef<UserRegistrationDialogComponent>,

private readonly \_fb: NonNullableFormBuilder

) {}

public cancel(): void {

this.\_dialogRef.close();

}

public create(): void {

if (this.createUserForm.invalid) return;

this.\_dialogRef.close(this.createUserForm.value);

}

}

Файл BookManager.Domain.Enums.ts:

/\* THIS (.ts) FILE IS GENERATED BY Tapper \*/

/\* eslint-disable \*/

/\* tslint:disable \*/

/\*\* Transpiled from BookManager.Domain.Enums.BookFileType \*/

export enum BookFileType {

Pdf = 0,

Epub = 1,

}

Файл auth.guard.ts:

import { inject } from '@angular/core';

import { CanActivateFn, Router } from '@angular/router';

import { CONSTANTS } from '@core/constants';

import { AuthState } from '@core/stores/auth.state';

import { AuthService } from '@core/services/auth.service';

import { map } from 'rxjs';

export const authGuard: CanActivateFn = (\_, state) => {

const authState = inject(AuthState);

const authService = inject(AuthService);

const router = inject(Router);

if (authState.isSignedIn()) {

return true;

} else {

return authService.refreshToken()

.pipe(

map((isSignedIn) =>

router.parseUrl(

!isSignedIn

? `/${CONSTANTS.ENDPOINTS.AUTH.PATH}/${CONSTANTS.ENDPOINTS.AUTH.SIGN\_IN}`

: state.url,

)),

);

}

};

Файл login.guard.ts:

import { inject } from '@angular/core';

import { CanActivateFn, Router } from '@angular/router';

import { AuthState } from '@core/stores/auth.state';

export const loginGuard: CanActivateFn = () => {

const authState = inject(AuthState);

const router = inject(Router);

if (!authState.isSignedIn()) {

return true;

}

return router.parseUrl('/');

};

Файл auth.interceptor.ts:

import { HttpInterceptorFn } from '@angular/common/http';

import { inject } from '@angular/core';

import { AuthState } from '@core/stores/auth.state';

export const authInterceptor: HttpInterceptorFn = (req, next) => {

const authState = inject(AuthState);

if (authState.isSignedIn()) {

const headers = {

Authorization: `Bearer ${authState.accessToken}`,

};

const request = req.clone({ setHeaders: headers, withCredentials: true });

return next(request);

}

return next(req);

};

Файл http-error.interceptor.ts:

import {

HttpErrorResponse,

HttpEvent,

HttpHandlerFn,

HttpInterceptorFn,

HttpRequest,

HttpStatusCode,

} from '@angular/common/http';

import { EnvironmentInjector, inject, runInInjectionContext } from '@angular/core';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Router } from '@angular/router';

import { CONSTANTS } from '@core/constants';

import { AuthService } from '@core/services/auth.service';

import { AuthState } from '@core/stores/auth.state';

import {

catchError,

delay,

map,

mergeMap,

NEVER,

Observable,

retry,

retryWhen,

switchMap,

throwError,

timer,

} from 'rxjs';

const retryCount = 3;

const handleUnauthorizedError = (

req: HttpRequest<unknown>,

next: HttpHandlerFn,

err: HttpErrorResponse,

environmentInjector: EnvironmentInjector,

): Observable<HttpEvent<unknown>> => {

return runInInjectionContext(environmentInjector, () => {

const authService = inject(AuthService);

const authState = inject(AuthState);

const router = inject(Router);

return authService.refreshToken()

.pipe(

mergeMap((success) => {

if (!success) {

authState.accessToken = null;

routeToSignInPage(router);

return throwError(() => err);

}

return next(req);

}),

);

});

};

const routeToSignInPage = (router: Router) => {

router.navigate([

CONSTANTS.ENDPOINTS.AUTH.PATH,

CONSTANTS.ENDPOINTS.AUTH.SIGN\_IN,

]);

};

export const httpErrorInterceptor: HttpInterceptorFn = (req, next) => {

const environmentInjector = inject(EnvironmentInjector);

const router = inject(Router);

const snackBarService = inject(MatSnackBar);

return next(req)

.pipe(

retry({

count: retryCount,

delay: (error: HttpErrorResponse) => {

console.log('retry request');

if (error.status === HttpStatusCode.Unauthorized) {

return handleUnauthorizedError(req, next, error, environmentInjector)

.pipe(

delay(3000),

switchMap(() => next(req)),

);

}

return throwError(() => error);

},

}),

catchError((err: HttpErrorResponse) => {

if (err.status === 0) {

router.navigate([CONSTANTS.ENDPOINTS.NO\_CONNECTION]);

} else {

if (

err.status === HttpStatusCode.Accepted ||

err.status === HttpStatusCode.Created

) {

return NEVER;

} else if (err.status === HttpStatusCode.Unauthorized) {

return handleUnauthorizedError(req, next, err, environmentInjector);

} else {

snackBarService.open(

`HTTP error from API ${err.status}: ${err.statusText}`,

'OK',

);

}

}

return throwError(() => err);

}),

);

};

Файл book-collections.component.html:

<div class="toolbar">

<h2>Коллекции книг</h2>

<div class="spacer"></div>

<div>

<button mat-flat-button

color="primary"

(click)="openBookCollectionEditDialog()">

<mat-icon>add</mat-icon>

Создать коллекцию

</button>

</div>

</div>

<div class="collection-list-container">

<app-book-collection-list [bookCollections]="bookCollections()"

(bookOpenClickEvent)="openBook($event)"

(bookInfoClickEvent)="openBookInfoDialog($event)"

(collectionEditClickEvent)="openBookCollectionEditDialog($event)"

(collectionDeleteClickEvent)="openDeleteConfirmationDialog($event)">

</app-book-collection-list>

@if (!bookCollections().length) {

<div class="no-items">Здесь пусто :(</div>

}

</div>

Файл book-collections.component.ts:

import { ChangeDetectionStrategy, Component, DestroyRef, OnInit, signal } from '@angular/core';

import { BookCollectionService } from '@core/services/book-collection.service';

import { BookCollectionDto, BookDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { MatButton } from '@angular/material/button';

import { MatIconModule } from '@angular/material/icon';

import { MatDialog } from '@angular/material/dialog';

import { BookCollectionEditDialogComponent } from '@core/dialogs/book-collection-edit-dialog/book-collection-edit-dialog.component';

import { CONSTANTS } from '@core/constants';

import { BookCollectionListComponent } from '@core/components/book-collection-list/book-collection-list.component';

import { DeleteConfirmationDialogComponent } from '@core/dialogs/delete-confirmation-dialog/delete-confirmation-dialog.component';

import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

import { finalize, mergeMap, NEVER, of } from 'rxjs';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Router } from '@angular/router';

import { BookInfoDialogComponent } from '@core/dialogs/book-info-dialog/book-info-dialog.component';

@Component({

selector: 'app-book-collections',

standalone: true,

imports: [

MatButton,

MatIconModule,

BookCollectionListComponent,

],

templateUrl: './book-collections.component.html',

styleUrl: './book-collections.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookCollectionsComponent implements OnInit {

public bookCollections = signal<BookCollectionDto[]>([]);

constructor(

private readonly \_service: BookCollectionService,

private readonly \_router: Router,

private readonly \_snackBar: MatSnackBar,

private readonly \_dialog: MatDialog,

private readonly \_destroyRef: DestroyRef,

) {

}

public ngOnInit(): void {

this.\_loadCollections();

}

private \_loadCollections(): void {

this.\_service.getAll()

.subscribe((collections) => {

this.bookCollections.set(collections);

});

}

public openBookCollectionEditDialog(collection?: BookCollectionDto): void {

const dialogRef = this.\_dialog.open(BookCollectionEditDialogComponent, {

minWidth: CONSTANTS.SIZE.DIALOG\_MIN\_WIDTH,

data: {

mode: !collection ? 'create' : 'update',

allBookCollections: this.bookCollections(),

bookCollection: collection,

},

});

dialogRef.afterClosed()

.pipe(

mergeMap((requestObject) => {

if (!requestObject) return NEVER;

return !collection

? this.\_service.create(requestObject)

: this.\_service.update(collection.id, requestObject);

}),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe((newOrModCollection) => this.\_handleDialogResponse(collection, newOrModCollection));

}

public openDeleteConfirmationDialog(collection: BookCollectionDto): void {

const dialogRef = this.\_dialog.open(

DeleteConfirmationDialogComponent,

{

data: {

message: CONSTANTS.TEXTS.BOOK\_COLLECTION\_DEL\_CONFIRM\_MESSAGE,

},

},

);

dialogRef.afterClosed()

.pipe(

mergeMap((isConfirmed) => {

if (!isConfirmed) return of(null);

return this.\_service.delete(collection.id)

.pipe(

finalize(() => {

this.bookCollections.update((value) =>

value.filter((bc) => bc.id !== collection.id),

);

this.\_snackBar.open(`Коллекция ${collection.name} успешно удалена`, 'OK');

}),

);

}),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe();

}

private \_handleDialogResponse(collection?: BookCollectionDto, newOrModCollection?: BookCollectionDto) {

if (!newOrModCollection) {

return;

}

let message = `Коллекция ${newOrModCollection.name} успешно создана.`;

if (!collection) {

this.bookCollections.update((value) => [...value, newOrModCollection]);

} else {

this.bookCollections.update((value) => {

const copy = [...value];

const index = copy.findIndex(bc => bc.id === newOrModCollection.id);

if (index === -1) {

console.error('Failed to find a book collection');

return copy;

} else {

copy[index] = newOrModCollection;

}

return copy;

});

message = `Коллекция ${newOrModCollection.name} успешно изменена.`;

}

this.\_snackBar.open(message, 'OK');

}

public openBook(book: BookDto): void {

this.\_router.navigate(['viewer', book.documentDetails.id]);

}

public openBookInfoDialog(book: BookDto): void {

this.\_dialog.open(BookInfoDialogComponent, {

width: '100vw',

maxWidth: '100vw',

Файл book-viewer.component.html:

<div

id="pdf-viewer-container"

app-tooltip-menu

[selectedDefinitionProvider]="selectedDefinitionProvider()"

[definitionProviders]="definitionProviders()"

[wordDefinitionEntries]="wordEntries()"

[definitionLoading]="isDefinitionLoading"

[isDefinitionMenuOpen]="isDefinitionMenuOpen"

(selectionEvent)="clearWordEntries()"

(textCopyEvent)="copyText($event)"

(definitionBtnClickEvent)="openDefinitionMenu($event)"

(translationBtnClickEvent)="openTranslationDialog($event)"

(textSumBtnClickEvent)="openTextSummarizationDialog($event)"

(definitionAddBtnClickEvent)="addWordsToDictionary($event)"

(selectedDefinitionProviderChange)="changeDefinitionProvider($event)"

>

<ngx-extended-pdf-viewer

#pdfViewer

[(page)]="page"

[sidebarVisible]="false"

[customToolbar]="customToolbar"

[contextMenuAllowed]="false"

[textLayer]="true"

[src]="documentSource()"

[authorization]="bearerToken"

[showHandToolButton]="true"

[handTool]="false"

>

</ngx-extended-pdf-viewer>

</div>

<ng-template #customToolbar>

<div id="custom-toolbar">

<div class="left">

<pdf-toggle-sidebar></pdf-toggle-sidebar>

<pdf-first-page></pdf-first-page>

<pdf-previous-page></pdf-previous-page>

<pdf-page-number class="custom-page-number"></pdf-page-number>

<pdf-next-page></pdf-next-page>

<pdf-last-page></pdf-last-page>

<pdf-find-button></pdf-find-button>

</div>

<div class="mid">

<pdf-zoom-in></pdf-zoom-in>

<pdf-zoom-out></pdf-zoom-out>

<pdf-zoom-dropdown

[zoomLevels]="pdfViewer.zoomLevels"

></pdf-zoom-dropdown>

</div>

<div class="right">

<pdf-hand-tool></pdf-hand-tool>

<pdf-select-tool></pdf-select-tool>

<pdf-presentation-mode></pdf-presentation-mode>

<pdf-print></pdf-print>

<pdf-download></pdf-download>

<pdf-toggle-secondary-toolbar></pdf-toggle-secondary-toolbar>

</div>

</div>

</ng-template>

Файл book-viewer.component.ts:

@Component({

selector: 'app-book-viewer',

templateUrl: './book-viewer.component.html',

styleUrl: './book-viewer.component.scss',

standalone: true,

imports: [

NgxExtendedPdfViewerModule,

TooltipMenuComponent,

TranslationDialogComponent,

FormsModule,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class BookViewerComponent implements OnInit, AfterViewInit, OnDestroy {

protected readonly DEFAULT\_TARGET\_LANG\_CODE = 'ru';

private readonly SELECTED\_DEFINITION\_PROVIDER\_SESSION\_STORAGE\_KEY = 'selected\_definition\_provider';

@ViewChild(NgxExtendedPdfViewerComponent)

public pdfViewer!: NgxExtendedPdfViewerComponent;

public definitionProviders = signal<string[]>([]);

public documentSource = signal<ArrayBuffer | Uint8Array | URL>(

new ArrayBuffer(0),

);

public wordEntries = signal<WordDto[]>([]);

public selectedDefinitionProvider = signal<string | null>('MerriamWebster');

public bearerToken?: string;

public isDefinitionLoading = false;

public isDefinitionMenuOpen = false;

public selectedWord?: string;

private \_dictionaryWordRegex = new RegExp(CONSTANTS.REGEX\_PATTERN.DICTIONARY\_WORD, 'u');

private \_currentBook?: BookDto;

private \_page?: number;

private \_totalTimeInSec = 0;

private \_openPageDateTime!: Date;

constructor(

private readonly \_service: BookService,

private readonly \_dictionaryService: DictionaryService,

private readonly \_cdr: ChangeDetectorRef,

private readonly \_authState: AuthState,

private readonly \_route: ActivatedRoute,

private readonly \_snackBar: MatSnackBar,

private readonly \_dialog: MatDialog,

private readonly \_title: Title,

private readonly \_clipboard: Clipboard,

private readonly \_destroyRef: DestroyRef,

) {

}

public ngOnInit(): void {

pdfDefaultOptions.externalLinkTarget = 2;

pdfDefaultOptions.enableScripting = false;

this.\_subscribeToParamMap();

if (this.\_authState.accessToken) {

this.bearerToken = `Bearer ${this.\_authState.accessToken}`;

}

this.\_loadDefinitionProviders();

}

public ngAfterViewInit(): void {

this.\_openPageDateTime = new Date();

this.\_loadSettings();

}

public ngOnDestroy(): void {

this.\_updateLastViewedPage();

this.\_updateTotalTime();

this.\_saveSettings();

}

@HostListener('window:beforeunload', ['$event'])

public onBeforeUnload(): void {

this.\_updateLastViewedPage();

this.\_updateTotalTime();

this.\_saveSettings();

}

@HostListener('document:visibilitychange')

public onVisibilityChange() {

if (document.visibilityState === 'visible') {

this.\_openPageDateTime = new Date();

} else {

this.\_updateTotalTime();

}

}

public set page(value: number | undefined) {

this.\_page = value;

}

public get page(): number | undefined {

return this.\_page;

}

public copyText(selectedText: string): void {

this.\_clipboard.copy(selectedText);

this.\_snackBar.open('Текст скопирован', 'OK', { duration: 3000 });

}

public openTranslationDialog(selectedText: string): void {

if (!selectedText) return;

this.\_dialog.open(TranslationDialogComponent, {

minWidth: CONSTANTS.SIZE.TRANSLATION\_DIALOG\_MIN\_WIDTH,

minHeight: CONSTANTS.SIZE.TRANSLATION\_DIALOG\_MIN\_HEIGHT,

data: {

sourceText: selectedText,

targetLanguageCode: this.DEFAULT\_TARGET\_LANG\_CODE,

},

});

}

public openTextSummarizationDialog(selectedText: string): void {

if (!selectedText) return;

Файл library-explorer.component.html:

<div class="toolbar mat-primary" [class.handset]="isHandset()">

<div class="left-section">

@if (isHandset()) {

<button mat-mini-fab color="primary" (click)="bookFileInput.click()">

<mat-icon>add</mat-icon>

</button>

} @else {

<button mat-flat-button (click)="bookFileInput.click()">

<mat-icon>add</mat-icon>

Добавить

</button>

}

<input

#bookFileInput

class="book-file-input"

type="file"

accept="application/pdf"

(change)="onFileInputChange()"

/>

<div class="toggle-view-panel">

<button mat-icon-button

[class.active]="selectedViewMode() === ViewMode.List"

(click)="setViewMode(ViewMode.List)"

>

<mat-icon>view\_list</mat-icon>

</button>

<button mat-icon-button

[class.active]="selectedViewMode() === ViewMode.Grid"

(click)="setViewMode(ViewMode.Grid)"

>

<mat-icon>grid\_view</mat-icon>

</button>

</div>

<app-sort-menu

[sortOptions]="SORT\_OPTIONS"

[(sortOption)]="selectedSortOption"

[(sortOrder)]="selectedSortOrder"

>

</app-sort-menu>

</div>

<div class="right-section">

<mat-form-field class="search-form-field"

appearance="outline"

subscriptSizing="dynamic"

[@searchFocus]="isInSearchMode() && !isHandset()"

>

<input matInput

type="text"

placeholder="Поиск"

[formControl]="searchFormControl"

(focus)="onSearchInputFocus()"

(blur)="onSearchInputBlur()"

/>

<mat-icon matPrefix>search</mat-icon>

@if (searchFormControl.value) {

<button mat-icon-button matSuffix (click)="searchFormControl.reset()">

<mat-icon>clear</mat-icon>

</button>

}

</mat-form-field>

</div>

</div>

<div #viewContainer

class="view-container"

infiniteScroll

[infiniteScrollDistance]="1"

[infiniteScrollUpDistance]="1.5"

[infiniteScrollThrottle]="50"

[scrollWindow]="false"

(scrolled)="loadNextPage()"

>

@if (isBooksEmpty()) {

<div class="no-items">Здесь пусто :(</div>

}

@switch (selectedViewMode()) {

@case (ViewMode.List) {

<app-book-list-view

[books]="books()"

(openItemEvent)="openBook($event)"

(infoItemEvent)="openBookInfoDialog($event)"

(editItemEvent)="editBook($event)"

(deleteItemEvent)="deleteBook($event)"

(numOfVisibleItemsChangeEvent)="handleNumOfVisibleItemsChange($event)"

>

<app-loading-spinner-overlay [loading]="isLoading()">

</app-loading-spinner-overlay>

</app-book-list-view>

}

@case (ViewMode.Grid) {

<app-book-grid-view

[books]="books()"

(openItemEvent)="openBook($event)"

(infoItemEvent)="openBookInfoDialog($event)"

(editItemEvent)="editBook($event)"

(deleteItemEvent)="deleteBook($event)"

(numOfVisibleItemsChangeEvent)="handleNumOfVisibleItemsChange($event)"

>

<app-loading-spinner-overlay [loading]="isLoading()">

</app-loading-spinner-overlay>

</app-book-grid-view>

}

}

</div>

Файл library-explorer.component.ts:

import {

ChangeDetectionStrategy,

Component, computed,

DestroyRef,

ElementRef,

OnDestroy,

OnInit, Signal,

signal,

viewChild,

} from '@angular/core';

import { takeUntilDestroyed, toSignal } from '@angular/core/rxjs-interop';

import { FormControl, ReactiveFormsModule, Validators } from '@angular/forms';

import { MatDialog } from '@angular/material/dialog';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Router } from '@angular/router';

import { getBookFileType } from '@core/book-file-type';

import { SortMenuComponent, SortOption } from '@core/components/sort-menu/sort-menu.component';

import { CONSTANTS } from '@core/constants';

import {

BookEditDialogComponent,

BookEditDialogData,

} from '@core/dialogs/book-edit-dialog/book-edit-dialog.component';

import { DeleteConfirmationDialogComponent } from '@core/dialogs/delete-confirmation-dialog/delete-confirmation-dialog.component';

import {

BookDetailsUpdateDto,

BookDto,

BookMetadataDto, SearchRequestDto,

SortOrder,

} from '@core/dtos/BookManager.Application.Common.DTOs';

import { BookService } from '@core/services/book.service';

import { debounceTime, finalize, map, mergeMap, Observable, of, tap } from 'rxjs';

import { MatSort } from '@angular/material/sort';

import { MatIcon } from '@angular/material/icon';

import { MatButton, MatButtonModule, MatIconButton } from '@angular/material/button';

import { MatFormField, MatPrefix, MatSuffix } from '@angular/material/form-field';

import { MatInput } from '@angular/material/input';

import { InfiniteScrollModule } from 'ngx-infinite-scroll';

import { LoadingSpinnerOverlayComponent } from '@shared/components/loading-spinner-overlay/loading-spinner-overlay.component';

import { BookGridViewComponent } from '@core/components/book-grid-view/book-grid-view.component';

import { BookListViewComponent } from '@core/components/book-list-view/book-list-view.component';

import { MatLabel } from '@angular/material/select';

import { animate, state, style, transition, trigger } from '@angular/animations';

import { BreakpointObserver, Breakpoints } from '@angular/cdk/layout';

import { AsyncPipe } from '@angular/common';

import { BookInfoDialogComponent } from '@core/dialogs/book-info-dialog/book-info-dialog.component';

enum ViewMode {

List,

Grid,

}

@Component({

selector: 'app-library-explorer',

templateUrl: './library-explorer.component.html',

styleUrl: './library-explorer.component.scss',

standalone: true,

imports: [

InfiniteScrollModule,

LoadingSpinnerOverlayComponent,

MatFormField,

MatIcon,

MatButtonModule,

MatInput,

MatLabel,

MatSort,

ReactiveFormsModule,

SortMenuComponent,

BookGridViewComponent,

BookListViewComponent,

BookEditDialogComponent,

DeleteConfirmationDialogComponent,

MatPrefix,

MatSuffix,

AsyncPipe,

],

animations: [

trigger('searchFocus', [

state('true', style({ width: '100%' })),

state('false', style({ width: '240px' })),

transition('false => true', animate('400ms')),

transition('true => false', animate('200ms')),

]),

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class LibraryExplorerComponent implements OnInit, OnDestroy {

public ViewMode = ViewMode;

public readonly SORT\_OPTIONS: SortOption[] = [

{ value: 'title', name: 'По названию' },

{ value: 'isbn', name: 'По ISBN' },

{ value: 'recent\_access', name: 'По посл. открытию' },

];

protected readonly DEFAULT\_PAGE\_SIZE = CONSTANTS.PAGE\_SIZE;

protected readonly DEFAULT\_SORT\_OPTION = {

value: 'recent\_access',

name: 'По посл. открытию',

};

protected readonly DEFAULT\_SORT\_ORDER = SortOrder.Asc;

protected readonly SORT\_OPTION\_KEY = 'library-explorer-sort-option';

protected readonly SORT\_ORDER\_KEY = 'library-explorer-sort-order';

protected readonly VIEW\_MODE\_KEY = 'library-explorer-view-mode';

public fileInputElement = viewChild<ElementRef<HTMLInputElement>>('bookFileInput');

public books = signal<BookDto[]>([]);

public isBooksEmpty = computed(() => {

return this.books().length === 0;

})

public currentPageNumber = signal<number>(1);

public pageSize = signal<number>(this.DEFAULT\_PAGE\_SIZE);

public selectedViewMode = signal<ViewMode>(ViewMode.List);

public isLoading = signal<boolean>(false);

public searchFormControl = new FormControl<string | null>(null, [Validators.minLength(3)]);

public isSearchInFocus = signal<boolean>(false);

public isInSearchMode: Signal<boolean> = computed(() => {

const isInFocus = this.isSearchInFocus();

return isInFocus || !!this.searchFormControl.value;

});

public isHandset = toSignal(this.\_breakpointObserver.observe([Breakpoints.Handset])

.pipe(map((result) => result.matches)));

private \_selectedSortOption = this.DEFAULT\_SORT\_OPTION;

private \_selectedSortOrder = this.DEFAULT\_SORT\_ORDER;

private \_pageCount = 0;

constructor(

private readonly \_bookService: BookService,

private readonly \_dialog: MatDialog,

private readonly \_router: Router,

private readonly \_snackBar: MatSnackBar,

private readonly \_destroyRef: DestroyRef,

private readonly \_breakpointObserver: BreakpointObserver,

) {

}

public ngOnInit(): void {

this.\_loadViewSettings();

this.\_subscribeToSearchChanges();

}

public ngOnDestroy(): void {

this.\_saveViewSettings();

}

public get selectedSortOption(): SortOption {

return this.\_selectedSortOption;

}

public get selectedSortOrder(): SortOrder {

return this.\_selectedSortOrder;

}

public set selectedSortOption(value: SortOption) {

this.\_selectedSortOption = value;

this.\_loadPageOfBookList(1, this.pageSize() \* this.currentPageNumber());

}

public set selectedSortOrder(value: SortOrder) {

this.\_selectedSortOrder = value;

this.\_loadPageOfBookList(1, this.pageSize() \* this.currentPageNumber());

}

public onSearchInputFocus(): void {

this.isSearchInFocus.set(true);

}

public onSearchInputBlur(): void {

this.isSearchInFocus.set(false);

}

public handleNumOfVisibleItemsChange(numOfVisibleItems: number) {

this.pageSize.set(Math.round(numOfVisibleItems \* 2));

this.\_loadPageOfBookList(1, this.pageSize() \* this.currentPageNumber());

}

public onFileInputChange() {

const files = this.fileInputElement()!.nativeElement.files!;

const file = files[0];

if (!file || !(file instanceof File)) {

return;

}

this.openBookAddDialog(file);

}

public setViewMode(mode: ViewMode): void {

this.selectedViewMode.set(mode);

}

public openBookInfoDialog(book: BookDto): void {

this.\_dialog.open(BookInfoDialogComponent, {

width: '100vw',

maxWidth: '100vw',

height: '100vh',

maxHeight: '100vh',

panelClass: 'fullscreen',

data: {

book,

},

});

}

public openBookAddDialog(file: File): void {

this.\_dialog

.open(BookEditDialogComponent, { minWidth: CONSTANTS.SIZE.DIALOG\_MIN\_WIDTH })

.afterClosed()

.pipe(

mergeMap((data: BookEditDialogData | undefined) => {

this.isLoading.set(true);

if (!data) return of(null);

const bookMetadata: BookMetadataDto = {

...data.bookDetails,

filename: file.name,

fileSizeInBytes: file.size,

fileType: getBookFileType(file.type),

};

return this.\_bookService.addBook(bookMetadata, file);

}),

finalize(() => {

this.\_resetFileInput();

this.isLoading.set(false);

}),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe((book) => {

if (!book) return;

this.\_loadPageOfBookList(1, this.pageSize() \* this.currentPageNumber());

this.\_snackBar.open(

`Добавлена новая книга "${book.documentDetails.title}"`,

'OK',

{ duration: 3000 },

);

});

}

public editBook(book: BookDto): void {

const details = book.documentDetails;

const data: BookEditDialogData = {

mode: 'update',

bookDetails: {

title: details.title,

description: details.description,

isbn: details.isbn,

publisherName: details.publisherName,

authors: details.authors,

tags: details.tags,

},

};

this.\_dialog

.open(BookEditDialogComponent, {

data,

minWidth: CONSTANTS.SIZE.DIALOG\_MIN\_WIDTH,

})

.afterClosed()

.pipe(

mergeMap((dialogReturnData: BookEditDialogData) => {

this.isLoading.set(true);

if (!dialogReturnData) return of(null);

const modifiedDetails = dialogReturnData.bookDetails;

const request: BookDetailsUpdateDto = {

title: modifiedDetails.title,

isbn: modifiedDetails.isbn,

description: modifiedDetails.description,

publisherName: modifiedDetails.publisherName,

authors: modifiedDetails.authors,

tags: modifiedDetails.tags,

};

return this.\_bookService.updateBookDetails(

book.documentDetails.id,

request,

);

}),

finalize(() => {

this.\_resetFileInput();

this.isLoading.set(false);

}),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe((book) => {

if (!book) return;

this.\_loadPageOfBookList(1, this.pageSize() \* this.currentPageNumber());

this.\_snackBar.open(

`Обновлены данные о книге "${book.documentDetails.title}"`,

'OK',

{ duration: 3000 },

);

});

}

public deleteBook(book: BookDto): void {

const dialogRef = this.\_dialog.open(

DeleteConfirmationDialogComponent,

{

data: {

message: CONSTANTS.TEXTS.BOOK\_DELETE\_CONFIRMATION\_MESSAGE,

},

},

);

dialogRef

.afterClosed()

.pipe(

mergeMap((isConfirmed: boolean) => {

if (!isConfirmed) return of(null);

return this.\_bookService.deleteBook(book.documentDetails.id);

}),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe(() => {

this.\_loadPageOfBookList(1, this.pageSize() \* this.currentPageNumber());

this.\_snackBar.open(

`Книга "${book.documentDetails.title} успешно удалена."`,

'OK',

{ duration: 3000 },

);

});

}

public async openBook(book: BookDto): Promise<void> {

await this.\_router.navigate(['viewer', book.documentDetails.id]);

}

public loadNextPage(): void {

if (this.currentPageNumber() > this.\_pageCount - 1) return;

const nextPage = this.currentPageNumber() + 1;

this.currentPageNumber.set(nextPage);

this.\_loadPageOfBookList(nextPage, this.pageSize(), false);

}

private \_loadPageOfBookList(

pageNumber: number,

pageSize: number = this.pageSize(),

shouldReset: boolean = true,

): void {

this.isLoading.set(true);

const searchValue = this.searchFormControl.value;

const page$ = !searchValue ?

this.\_bookService.getPage(

pageNumber,

pageSize,

this.selectedSortOption.value,

this.selectedSortOrder,

)

: this.\_bookService.searchByBookDetails(this.\_createSearchRequest(pageNumber, pageSize, searchValue));

page$

.pipe(

tap((page) => (this.\_pageCount = page.pageCount)),

finalize(() => this.isLoading.set(false)),

)

.subscribe((page) => {

if (!shouldReset) {

this.books.update((prevItems) => [...prevItems, ...page.items]);

} else {

this.books.set(page.items);

}

});

}

private \_subscribeToSearchChanges(): void {

this.searchFormControl.valueChanges

.pipe(

tap(() => this.searchFormControl.markAsTouched()),

debounceTime(500),

takeUntilDestroyed(this.\_destroyRef),

)

.subscribe(() => {

if (this.searchFormControl.invalid) return;

this.currentPageNumber.set(1);

this.\_loadPageOfBookList(1, this.pageSize());

});

}

private \_loadViewSettings(): void {

const sortOptionName = sessionStorage.getItem(this.SORT\_OPTION\_KEY);

const sortOption = this.SORT\_OPTIONS.find(

(option) => option.name === sortOptionName,

);

if (sortOption) this.\_selectedSortOption = sortOption;

const sortOrderStringNumber = sessionStorage.getItem(this.SORT\_ORDER\_KEY);

if (sortOrderStringNumber)

this.\_selectedSortOrder = parseInt(sortOrderStringNumber);

const viewModeStringNumber = sessionStorage.getItem(this.VIEW\_MODE\_KEY);

if (viewModeStringNumber)

this.selectedViewMode.set(parseInt(viewModeStringNumber));

}

private \_saveViewSettings(): void {

sessionStorage.setItem(this.SORT\_OPTION\_KEY, this.selectedSortOption.name);

sessionStorage.setItem(this.SORT\_ORDER\_KEY, `${this.\_selectedSortOrder}`);

sessionStorage.setItem(this.VIEW\_MODE\_KEY, `${this.selectedViewMode()}`);

}

private \_resetFileInput(): void {

this.fileInputElement()!.nativeElement.value = '';

}

private \_createSearchRequest(pageNumber: number, pageSize: number, value: string): SearchRequestDto {

return {

pageNumber,

pageSize,

sortProperty: this.\_selectedSortOption.value,

sortOrder: this.\_selectedSortOrder,

title: value,

description: value,

publisherName: value,

authors: [value],

};

}

}

Файл login.component.html:

<div class="login-form" [formGroup]="loginForm">

<h2>Вход в персональную библиотеку</h2>

<mat-form-field>

<mat-label>Пользователь</mat-label>

<mat-select formControlName="selectedUser">

@for (user of users(); track $index) {

<mat-option [value]="user">{{ user.name }}</mat-option>

}

</mat-select>

</mat-form-field>

<mat-form-field>

<mat-label>ПИН-код</mat-label>

<input

matInput

type="password"

formControlName="pinCode"

inputmode="numeric"

minlength="4"

maxlength="16"

digitOnly

/>

</mat-form-field>

<button

mat-flat-button

color="primary"

[disabled]="loginForm.invalid || loginForm.pristine"

(click)="login()"

>

<mat-icon>login</mat-icon>

Войти

</button>

<button mat-flat-button color="accent" (click)="createUser()">

<mat-icon>add</mat-icon>

Создать нового пользователя

</button>

</div>

Файл login.component.ts:

import { HttpErrorResponse } from '@angular/common/http';

import { ChangeDetectionStrategy, Component, DestroyRef, OnInit, signal } from '@angular/core';

import { takeUntilDestroyed } from '@angular/core/rxjs-interop';

import { NonNullableFormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';

import { MatDialog } from '@angular/material/dialog';

import { MatSnackBar } from '@angular/material/snack-bar';

import { Router } from '@angular/router';

import { CONSTANTS } from '@core/constants';

import { UserRegistrationDialogComponent } from '@core/dialogs/user-registration-dialog/user-registration-dialog.component';

import { UserAddRequest, UserDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { AuthService } from '@core/services/auth.service';

import { UserService } from '@core/services/user.service';

import { AuthState } from '@core/stores/auth.state';

import { catchError, mergeMap, NEVER, Observable, throwError } from 'rxjs';

import { DigitOnlyModule } from '@uiowa/digit-only';

import { MatOption } from '@angular/material/core';

import { MatFormField, MatLabel, MatSelect } from '@angular/material/select';

import { MatInput } from '@angular/material/input';

import { MatButton } from '@angular/material/button';

import { MatIcon } from '@angular/material/icon';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrl: './login.component.scss',

standalone: true,

imports: [

DigitOnlyModule,

MatButton,

MatFormField,

MatInput,

MatLabel,

MatOption,

MatSelect,

MatIcon,

ReactiveFormsModule,

UserRegistrationDialogComponent,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class LoginComponent implements OnInit {

public users = signal<UserDto[]>([]);

public loginForm = this.\_fb.group({

selectedUser: this.\_fb.control<UserDto | null>(null, [Validators.required]),

pinCode: this.\_fb.control<string | null>(null, [

Validators.required,

Validators.pattern(CONSTANTS.REGEX\_PATTERN.PIN\_CODE),

]),

});

constructor(

private readonly \_service: UserService,

private readonly \_authState: AuthState,

private readonly \_authService: AuthService,

private readonly \_snackBar: MatSnackBar,

private readonly \_dialog: MatDialog,

private readonly \_fb: NonNullableFormBuilder,

private readonly \_router: Router,

private readonly \_destroyRef: DestroyRef

) {}

public ngOnInit(): void {

if (this.\_authState.isSignedIn()) {

this.\_routeToMainPage();

}

this.loadUsers();

}

public loadUsers(): void {

this.\_service

.getUsers()

.pipe(

catchError((err) => {

this.\_snackBar.open(

'Не удалось загрузить список пользователей.',

'OK'

);

return throwError(() => err);

})

)

.subscribe((users) => {

this.users.set(users);

});

}

public login(): void {

const values = this.loginForm.value;

const username = values.selectedUser?.name;

const password = values.pinCode;

if (this.loginForm.invalid) {

this.\_snackBar.open('The username or password field is empty!', 'OK');

return;

}

if (!!username && !!password) {

this.\_authService

.signIn(username, password)

.pipe(catchError((err) => this.\_handleSignInError(err)))

.subscribe(() => {

this.\_routeToMainPage();

});

}

}

public createUser(): void {

const dialogRef = this.\_dialog.open(UserRegistrationDialogComponent);

dialogRef

.afterClosed()

.pipe(

mergeMap((newUser: UserAddRequest | undefined) => {

if (!newUser) return NEVER;

return this.\_service.addUser(newUser);

}),

catchError((err) => {

console.error(`Failed to create user: ${err}`);

this.\_snackBar.open(

`Произошла ошибка при создании пользователя: ${err}`

);

return throwError(() => err);

}),

takeUntilDestroyed(this.\_destroyRef)

)

.subscribe((user) => {

this.\_snackBar.open(`Пользователь ${user.name} создан!`, 'OK');

this.loadUsers();

});

}

private \_routeToMainPage() {

this.\_router.navigate([CONSTANTS.ENDPOINTS.RECENT\_BOOKS]);

}

private \_handleSignInError(err: unknown): Observable<never> {

if (err instanceof HttpErrorResponse && err.status === 401) {

const errorMessage = (err.error as Record<string, string>)['message'];

this.loginForm.setErrors({

invalidUsernameOrPassword: errorMessage,

});

this.\_snackBar.open(errorMessage, 'OK');

}

return throwError(() => err);

}

}

Файл no-connection.component.html:

<h2>No connection</h2>

<p>Please, check a network connection or server availability</p>

Файл no-connection.component.ts:

import { ChangeDetectionStrategy, Component } from '@angular/core';

@Component({

selector: 'app-no-connection',

standalone: true,

imports: [],

templateUrl: './no-connection.component.html',

styleUrl: './no-connection.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class NoConnectionComponent {}

Файл recent-books.component.html:

<h1>Недавние</h1>

@if (mostRecentBook()) {

<div class="most-recent-book">

<div class="item">

<div class="thumbnail-container">

<img [src]="mostRecentBook()!.documentDetails.thumbnailUrl" alt="no-image" />

</div>

<div class="info">

<h2 class="title">

{{ mostRecentBook()!.documentDetails.title }}

</h2>

@if (mostRecentBook()!.stats?.lastViewedPage) {

<div>

Вы остановились на странице

{{ mostRecentBook()!.stats?.lastViewedPage }} /

{{ mostRecentBook()!.documentDetails.pageCount }}

</div>

}

</div>

<div class="spacer"></div>

<button mat-flat-button (click)="openBook(mostRecentBook()!)">

Продолжить

</button>

</div>

@if (mostRecentBookProgressValue()) {

<mat-progress-bar

[value]="mostRecentBookProgressValue()"

></mat-progress-bar>

}

</div>

}

@if (isBooksEmpty()) {

<span class="no-items">Пусто</span>

}

<app-book-list-view

[books]="otherBooks()"

[showEditButton]="false"

[showDeleteButton]="false"

(openItemEvent)="openBook($event)"

></app-book-list-view>

Файл recent-books.component.ts:

import {

ChangeDetectionStrategy,

Component,

OnInit,

computed,

signal,

} from '@angular/core';

import { Router } from '@angular/router';

import {

BookDto,

SortOrder,

} from '@core/dtos/BookManager.Application.Common.DTOs';

import { BookService } from '@core/services/book.service';

import { MatButton } from '@angular/material/button';

import { MatProgressBar } from '@angular/material/progress-bar';

import { NgOptimizedImage } from '@angular/common';

import { BookListViewComponent } from '@core/components/book-list-view/book-list-view.component';

@Component({

selector: 'app-recent-books',

templateUrl: './recent-books.component.html',

styleUrl: './recent-books.component.scss',

standalone: true,

imports: [

MatButton,

MatProgressBar,

NgOptimizedImage,

BookListViewComponent,

],

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class RecentBooksComponent implements OnInit {

public otherBooks = signal<BookDto[]>([]);

public mostRecentBook = signal<BookDto | null>(null);

public isBooksEmpty = computed(() => {

return this.otherBooks().length === 0;

})

protected readonly DEFAULT\_PAGE\_SIZE = 5;

protected readonly DEFAULT\_PAGE\_NUMBER = 0;

protected readonly SORT\_BY = 'recent\_access';

protected readonly SORT\_ORDER = SortOrder.Desc;

public mostRecentBookProgressValue = computed(() => {

const mostRecentBook = this.mostRecentBook();

if (!mostRecentBook) return 0;

const pageCount = mostRecentBook?.documentDetails.pageCount ?? 0;

const pageNumber = mostRecentBook.stats?.lastViewedPage ?? 0;

return Math.round((pageNumber / pageCount) \* 100);

});

constructor(

private readonly \_service: BookService,

private readonly \_router: Router

) {}

public ngOnInit(): void {

this.\_loadBooks();

}

public openBook(book: BookDto): void {

this.\_router.navigate(['viewer', book.documentDetails.id]);

}

private \_loadBooks(): void {

this.\_service

.getPage(

this.DEFAULT\_PAGE\_NUMBER,

this.DEFAULT\_PAGE\_SIZE,

this.SORT\_BY,

this.SORT\_ORDER

)

.subscribe((page) => {

this.otherBooks.set(page.items.slice(1));

this.mostRecentBook.set(page.items[0]);

});

}

}

Файл auth-http-client.ts:

import { HttpBackend, HttpClient } from '@angular/common/http';

import { InjectionToken, Provider } from '@angular/core';

export const AUTH\_HTTP\_CLIENT = new InjectionToken<HttpClient>(

'A http client without interceptors'

);

export function provideAuthHttpClient(): Provider {

return {

provide: AUTH\_HTTP\_CLIENT,

deps: [HttpBackend],

useFactory: (httpBackend: HttpBackend) => new HttpClient(httpBackend),

};

}

Файл auth.service.ts:

import { HttpClient, HttpErrorResponse } from '@angular/common/http';

import { Inject, Injectable } from '@angular/core';

import { CONSTANTS } from '@core/constants';

import {

AuthenticationResponseDto,

AuthenticationStatus,

} from '@core/dtos/BookManager.Application.Common.DTOs';

import { AUTH\_HTTP\_CLIENT } from '@core/providers/auth-http-client';

import { AuthState } from '@core/stores/auth.state';

import { BehaviorSubject, Observable, catchError, map, of, tap } from 'rxjs';

import { environment } from '../../../environments/environment';

@Injectable({

providedIn: 'root',

})

export class AuthService {

private static readonly URL =

environment.BASE\_URL + '/' + CONSTANTS.SERVER\_URL.AUTH.PATH;

public isSignedIn$ = new BehaviorSubject<boolean>(false);

constructor(

private readonly \_state: AuthState,

private readonly \_httpClient: HttpClient,

@Inject(AUTH\_HTTP\_CLIENT) private readonly \_authHttpClient: HttpClient

) {}

public signIn(name: string, pinCode: string): Observable<void> {

return this.\_authHttpClient

.post<AuthenticationResponseDto>(

AuthService.URL + '/' + CONSTANTS.SERVER\_URL.AUTH.SIGN\_IN,

{

name,

pinCode,

},

{ withCredentials: true }

)

.pipe(

map((token) => {

if (token.accessToken) {

this.\_state.accessToken = token.accessToken;

this.isSignedIn$.next(true);

}

})

);

}

public refreshToken(): Observable<boolean> {

return this.\_authHttpClient

.post<AuthenticationResponseDto>(

AuthService.URL + '/' + CONSTANTS.SERVER\_URL.AUTH.REFRESH\_TOKEN,

undefined,

{

// A refreshToken is stored in http only cookie

withCredentials: true,

}

)

.pipe(

map((response) => {

if (

response.status === AuthenticationStatus.Success &&

response.accessToken

) {

this.\_state.accessToken = response.accessToken;

this.isSignedIn$.next(true);

}

return response.status === AuthenticationStatus.Success;

}),

catchError((err) => {

if (err instanceof HttpErrorResponse) {

console.log(err.message);

}

return of(false);

})

);

}

public signOut(): Observable<void> {

return this.\_httpClient

.post<void>(

AuthService.URL + '/' + CONSTANTS.SERVER\_URL.AUTH.SIGN\_OUT,

undefined,

{

withCredentials: true,

}

)

.pipe(

tap(() => {

this.clearAccessToken();

})

);

}

public clearAccessToken(): void {

this.\_state.accessToken = null;

}

}

Файл book-collection.service.ts:

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { environment } from '../../../environments/environment.development';

import { Observable } from 'rxjs';

import { BookCollectionDto, BookCollectionModRequest } from '@core/dtos/BookManager.Application.Common.DTOs';

@Injectable({

providedIn: 'root',

})

export class BookCollectionService {

private readonly \_url: string = `${environment.BASE\_URL}/book-collections`;

constructor(

private readonly \_httpClient: HttpClient,

) {

}

public getAll(): Observable<BookCollectionDto[]> {

return this.\_httpClient.get<BookCollectionDto[]>(this.\_url);

}

public create(request: BookCollectionModRequest): Observable<BookCollectionDto> {

return this.\_httpClient.post<BookCollectionDto>(this.\_url, request);

}

public update(id: string, request: BookCollectionModRequest): Observable<BookCollectionDto> {

return this.\_httpClient.put<BookCollectionDto>(`${this.\_url}/${id}`, request);

}

public delete(id: string): Observable<void> {

return this.\_httpClient.delete<void>(`${this.\_url}/${id}`);

}

}

Файл book.service.ts:

import { HttpClient, HttpParams } from '@angular/common/http';

import { Injectable } from '@angular/core';

import {

BookDetailsUpdateDto,

BookDto,

BookMetadataDto,

LastViewedPageUpdateRequest,

PageDto,

SearchRequestDto,

SortOrder,

} from '@core/dtos/BookManager.Application.Common.DTOs';

import { Observable } from 'rxjs';

import { environment } from '../../../environments/environment.development';

@Injectable({

providedIn: 'root',

})

export class BookService {

private readonly \_url: string = `${environment.BASE\_URL}/books`;

constructor(private readonly \_httpClient: HttpClient) {

}

public getPage(

pageNumber: number,

pageSize: number,

sortBy?: string,

sortOrder?: SortOrder,

): Observable<PageDto<BookDto>> {

let queryParams = new HttpParams({ fromObject: { pageNumber, pageSize } });

if (sortBy) queryParams = queryParams.append('sortBy', sortBy);

if (sortOrder) queryParams = queryParams.append('sortOrder', sortOrder);

return this.\_httpClient.get<PageDto<BookDto>>(this.\_url, {

params: queryParams,

});

}

public getBookById(id: string): Observable<BookDto> {

return this.\_httpClient.get<BookDto>(`${this.\_url}/${id}`);

}

public addBook(

bookMetadata: BookMetadataDto,

file: File,

): Observable<BookDto> {

const formData = new FormData();

formData.append('bookMetadata', JSON.stringify(bookMetadata));

formData.append('file', file);

return this.\_httpClient.post<BookDto>(`${this.\_url}`, formData);

}

public updateBookDetails(

id: string,

details: BookDetailsUpdateDto,

): Observable<BookDto> {

return this.\_httpClient.put<BookDto>(`${this.\_url}/${id}`, details);

}

public updateTotalTime(

bookId: string,

totalTimeSeconds: number,

): Observable<void> {

return this.\_httpClient.post<void>(

`${this.\_url}/${bookId}/update-total-time`,

{ seconds: totalTimeSeconds },

);

}

public deleteBook(id: string): Observable<void> {

return this.\_httpClient.delete<void>(`${this.\_url}/${id}`);

}

public updateLastViewedPage(

id: string,

pageNumber: number,

): Observable<void> {

return this.\_httpClient.post<void>(`${this.\_url}/${id}/last-viewed-page`, {

pageNumber,

} as LastViewedPageUpdateRequest);

}

public getBookDownloadUrl(id: string): URL {

return new URL(`${this.\_url}/download/${id}`);

}

public searchByBookDetails(request: SearchRequestDto): Observable<PageDto<BookDto>> {

return this.\_httpClient.post<PageDto<BookDto>>(`${this.\_url}/search`, request);

}

}

Файл dictionary.service.ts:

import { Injectable } from '@angular/core';

import { HttpClient, HttpParams } from '@angular/common/http';

import { Observable } from 'rxjs';

import { WordDto } from '@core/dtos/BookManager.Application.Common.DTOs';

import { environment } from '../../../environments/environment';

@Injectable({

providedIn: 'root',

})

export class DictionaryService {

private readonly \_url = `${environment.BASE\_URL}/word-dictionary`;

constructor(

private readonly \_httpClient: HttpClient,

) {

}

public listThirdPartyProviders(): Observable<string[]> {

return this.\_httpClient.get<string[]>(`${this.\_url}/list-third-party-providers`);

}

public find(word: string): Observable<WordDto[]> {

const url = `${this.\_url}/${word}`;

return this.\_httpClient.get<WordDto[]>(url);

}

public findInExtDict(word: string, providerName: string): Observable<WordDto[]> {

const url = `${this.\_url}/third-party-dictionary/${word}`;

const params = new HttpParams().set('providerName', providerName);

return this.\_httpClient.get<WordDto[]>(url, { params });

}

public addWord(word: WordDto): Observable<WordDto> {

return this.\_httpClient.post<WordDto>(this.\_url, word);

}

public updateWord(word: string, updateRequest: WordDto): Observable<WordDto> {

return this.\_httpClient.put<WordDto>(`${this.\_url}/${word}`, updateRequest);

}

public deleteWord(word: string): Observable<void> {

return this.\_httpClient.delete<void>(`${this.\_url}/${word}`);

}

}

Файл text-processing.service.ts:

import { HttpClient } from '@angular/common/http';

import { Injectable, signal } from '@angular/core';

import {

DetectLanguageRequestDto,

DetectLanguageResponseDto,

LanguageDto, TextSummarizationRequestDto, TextSummarizationResponseDto,

TranslationRequestDto,

TranslationResponseDto,

} from '@core/dtos/BookManager.Application.Common.DTOs';

import { Observable } from 'rxjs';

import { environment } from '../../../environments/environment';

@Injectable({

providedIn: 'root',

})

export class TextProcessingService {

private readonly \_url: string = `${environment.BASE\_URL}/text-processing`;

public availableLanguages = signal<LanguageDto[]>([]);

constructor(private readonly \_client: HttpClient) {

this.\_loadLanguages();

}

public listLanguages(): Observable<LanguageDto[]> {

return this.\_client.get<LanguageDto[]>(`${this.\_url}/list-languages`);

}

public translate(

request: TranslationRequestDto

): Observable<TranslationResponseDto> {

return this.\_client.post<TranslationResponseDto>(

`${this.\_url}/translate`,

request

);

}

public detectLanguage(

request: DetectLanguageRequestDto

): Observable<DetectLanguageResponseDto> {

return this.\_client.post<DetectLanguageResponseDto>(

`${this.\_url}/detect-language`,

request

);

}

public summarizeText(request: TextSummarizationRequestDto): Observable<TextSummarizationResponseDto> {

return this.\_client.post<TextSummarizationResponseDto>(`${this.\_url}/summarize-text`, request);

}

private \_loadLanguages(): void {

this.listLanguages().subscribe((languages) => {

this.availableLanguages.set(languages);

});

}

}

Файл user.service.ts:

import { HttpClient } from '@angular/common/http';

import { Injectable } from '@angular/core';

import {

UserAddRequest,

UserDto,

} from '@core/dtos/BookManager.Application.Common.DTOs';

import { Observable } from 'rxjs';

import { environment } from '../../../environments/environment.development';

@Injectable({

providedIn: 'root',

})

export class UserService {

private readonly \_url: string = `${environment.BASE\_URL}/users`;

constructor(private readonly \_httpClient: HttpClient) {}

public getUsers(): Observable<UserDto[]> {

return this.\_httpClient.get<UserDto[]>(this.\_url);

}

public addUser(request: UserAddRequest): Observable<UserDto> {

return this.\_httpClient.post<UserDto>(this.\_url, request);

}

public updateUser(id: string, request: UserAddRequest): Observable<UserDto> {

return this.\_httpClient.put<UserDto>(`${this.\_url}/${id}`, request);

}

public deleteUser(id: string): Observable<void> {

return this.\_httpClient.delete<void>(`${this.\_url}/${id}`);

}

}

Файл auth.state.ts:

import { Injectable, computed, signal } from '@angular/core';

@Injectable({

providedIn: 'root',

})

export class AuthState {

public isSignedIn = computed<boolean>(() => this.\_signedIn());

private \_signedIn = signal(false);

private \_accessToken = signal<string | null>(null);

public get accessToken(): string | null {

return this.\_accessToken();

}

public set accessToken(value: string | null) {

this.\_accessToken.set(value);

this.\_signedIn.set(!!value);

}

}

Файл loading-spinner-overlay.component.html:

@if (loading()) {

<div class="loading-overlay">

<mat-progress-spinner color="primary" mode="indeterminate"></mat-progress-spinner>

</div>

}

Файл loading-spinner-overlay.component.ts:

import { ChangeDetectionStrategy, Component, input } from '@angular/core';

import { MatProgressSpinner } from '@angular/material/progress-spinner';

@Component({

selector: 'app-loading-spinner-overlay',

standalone: true,

imports: [MatProgressSpinner],

templateUrl: './loading-spinner-overlay.component.html',

styleUrl: './loading-spinner-overlay.component.scss',

changeDetection: ChangeDetectionStrategy.OnPush,

})

export class LoadingSpinnerOverlayComponent {

public loading = input.required<boolean>();

}

Файл index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<title>App</title>

<base href="/" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link

href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap"

rel="stylesheet"

/>

<link

href="https://fonts.googleapis.com/icon?family=Material+Icons"

rel="stylesheet"

/>

</head>

<body class="mat-typography">

<app-root></app-root>

</body>

</html>