**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет *ИСТ* Кафедра *ИВК*

К ЗАЩИТЕ ДОПУСТИТЬ

Зав. кафедрой

/ С.К. Киселев /

подпись инициалы, фамилия

« » 20 24 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема *Интеллектуальный помощник для работы с электронной библиотекой*

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

подпись инициалы, фамилия

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

Направление подготовки *09.03.02 «Информационные системы и технологии»*

код, наименование

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

Руководитель ВКР / *В.М. Кандаулов* /

подпись, дата инициалы, фамилия

Консультанты:

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

наименование раздела подпись, дата инициалы, фамилия

Ульяновск, 2024

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Факультет  *ИСТ*  Кафедра  *ИВК* .

Направление подготовки *Информационные системы и технологии* .

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

ЗАДАНИЕ

**на выпускную квалификационную работу**

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

фамилия, имя, отчество

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

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

1. Консультанты

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

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

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

должность, учёная степень, ученое звание подпись инициалы, фамилия

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

подпись обучающегосяинициалы, фамилия

Содержание

[CПИСОК ИСПОЛЬЗОВАННЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ 7](#_Toc167876924)

[ВВЕДЕНИЕ 8](#_Toc167876925)

[1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ СИСТЕМЫ 10](#_Toc167876926)

[1.1 Назначение и цели создания системы 10](#_Toc167876927)

[1.2 Характеристика объекта автоматизации 10](#_Toc167876928)

[1.2.1 Общее описание 10](#_Toc167876929)

[1.2.2 Состав и порядок функционирования 10](#_Toc167876930)

[1.3 Общие требования к системе 11](#_Toc167876931)

[1.3.1 Требования к структуре и функционированию системы 11](#_Toc167876932)

[1.3.2 Дополнительные требования 12](#_Toc167876933)

[1.4 Требования к функциям, выполняемым системой 12](#_Toc167876934)

[1.5 Требования к видам обеспечения 12](#_Toc167876935)

[1.5.1 Требования к информационному обеспечению 12](#_Toc167876936)

[1.5.2 Требования к алгоритмическому обеспечению 13](#_Toc167876937)

[1.5.3 Требования к программному обеспечению 13](#_Toc167876938)

[1.6 Анализ аналогичных разработок 14](#_Toc167876939)

[1.6.1 Calibre 14](#_Toc167876940)

[1.6.2 Kavita 15](#_Toc167876941)

[1.6.3 Koodo 16](#_Toc167876942)

[2 Информационное обеспечение системы 18](#_Toc167876943)

[2.1 Выбор средств управления данными 18](#_Toc167876944)

[2.2 Проектирование базы данных 20](#_Toc167876945)

[2.3 Организация сбора, передачи, обработки и выдачи информации 22](#_Toc167876946)

[3 Алгоритмическое обеспечение системы 23](#_Toc167876947)

[3.1 Алгоритм обмена информации между серверной частью и API искусственного интеллекта 23](#_Toc167876948)

[3.2 Алгоритм добавления книги в систему 23](#_Toc167876949)

[4 Программное обеспечение системы 24](#_Toc167876950)

[4.1 Структура программного обеспечения и функции его компонентов 24](#_Toc167876951)

[4.2 Выбор компонентов программного обеспечения 24](#_Toc167876952)

[4.2.1 Клиентская операционная система 24](#_Toc167876953)

[4.2.2 Инструментальное средство разработки и язык программирования 25](#_Toc167876954)

[4.2.3 Вспомогательное программное обеспечение 25](#_Toc167876955)

[4.3 Разработка прикладного программного обеспечения 25](#_Toc167876956)

[4.3.1 Структура прикладного программного обеспечения 25](#_Toc167876957)

[5 ТЕСТИРОВАНИЕ СИСТЕМЫ 28](#_Toc167876958)

[5.1 Условия и порядок тестирования 29](#_Toc167876959)

[5.2 Исходные данные для контрольных примеров 29](#_Toc167876960)

[5.3 Результаты тестирования 29](#_Toc167876961)

[6 ЭКОНОМИЧЕСКАЯ ЧАСТЬ 30](#_Toc167876962)

[6.1 Расчет показателя трудоёмкости для программного продукта 30](#_Toc167876963)

[6.2 Расчет затрат на материальные ресурсы 31](#_Toc167876964)

[6.3 Расчёт затрат на разработку системы 32](#_Toc167876965)

[6.4 Расчёт затрат на оплату труда 32](#_Toc167876966)

[6.5 Расчет отчислений на социальные нужды 33](#_Toc167876967)

[6.5 Себестоимость проекта 33](#_Toc167876968)

[6.6 Расчёт плановой прибыли 33](#_Toc167876969)

[6.7 Определение экономической эффективности проекта 34](#_Toc167876970)

[6.8 Выводы по технико-экономическому анализу и обоснованию проекта разработки 34](#_Toc167876971)

[ЗАКЛЮЧЕНИЕ 35](#_Toc167876972)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc167876973)

[ПРИЛОЖЕНИЕ А 37](#_Toc167876974)

# CПИСОК ИСПОЛЬЗОВАННЫХ ОБОЗНАЧЕНИЙ, СОКРАЩЕНИЙ И ТЕРМИНОВ

ИИ – искусственный интеллект.

СУБД – система управления базами данных.

ПО – программное обеспечение.

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

API – интерфейс программирования приложения (Application Programming Interface).

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

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

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

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

# ВВЕДЕНИЕ

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

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

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

Из основных недостатков искусственного интеллекта являются

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

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

# 1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА СОЗДАНИЕ СИСТЕМЫ

## 1.1 Назначение и цели создания системы

Назначением системы:

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

Местом внедрения является внутренняя локальная сеть пользователя.

Цель создания системы:

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

## 1.2 Характеристика объекта автоматизации

Объектом автоматизации является неавтоматизированное рабочее место читателя в локальной сети.

### 1.2.1 Общее описание

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

### 1.2.2 Состав и порядок функционирования

Функционирование системы включает в себя набор следующих процессов:

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

Алгоритм работы с файлами электронных книг следующий:

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

## 1.3 Общие требования к системе

### 1.3.1 Требования к структуре и функционированию системы

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

* подсистема сервера;
* подсистема клиента.

Подсистема серверной части должна состоять из следующих компонентов:

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

Пользовательский интерфейс подсистемы клиентской части программного обеспечения должен состоять из:

* страница аутентификации;
* страница списка электронных книг;
* страница просмотра электронной книги.

Требования к режимам функционирования системы предполагают:

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

Перспективы развития системы предполагают:

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

### 1.3.2 Дополнительные требования

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

Система является свободно распространяемой и не требует лицензии.

## 1.4 Требования к функциям, выполняемым системой

В системе должны быть реализованы следующие функции:

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

Первоочередность реализации функций указана по порядку.

## 1.5 Требования к видам обеспечения

### 1.5.1 Требования к информационному обеспечению

В системе должны быть предусмотрены следующие требования:

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

### 1.5.2 Требования к алгоритмическому обеспечению

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

Необходимо реализовать алгоритм добавления электронной книги в электронную библиотеку.

### 1.5.3 Требования к программному обеспечению

Требования к организации интерфейса:

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

Требования к написанию программного кода:

* язык программирования C#, JavaScript/TypeScript;
* веб-фреймворк ASP.NET, Angular;
* язык разметки HTML;
* текстовый редактор Visual Studio Code;
* интеграционная среда программирования JetBrains Rider.

Требования к пользователю системы:

* операционная система Windows NT не ниже 10 версии или дистрибутив Linux с любой графической оболочкой.

## 1.6 Анализ аналогичных разработок

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

### 1.6.1 Calibre

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

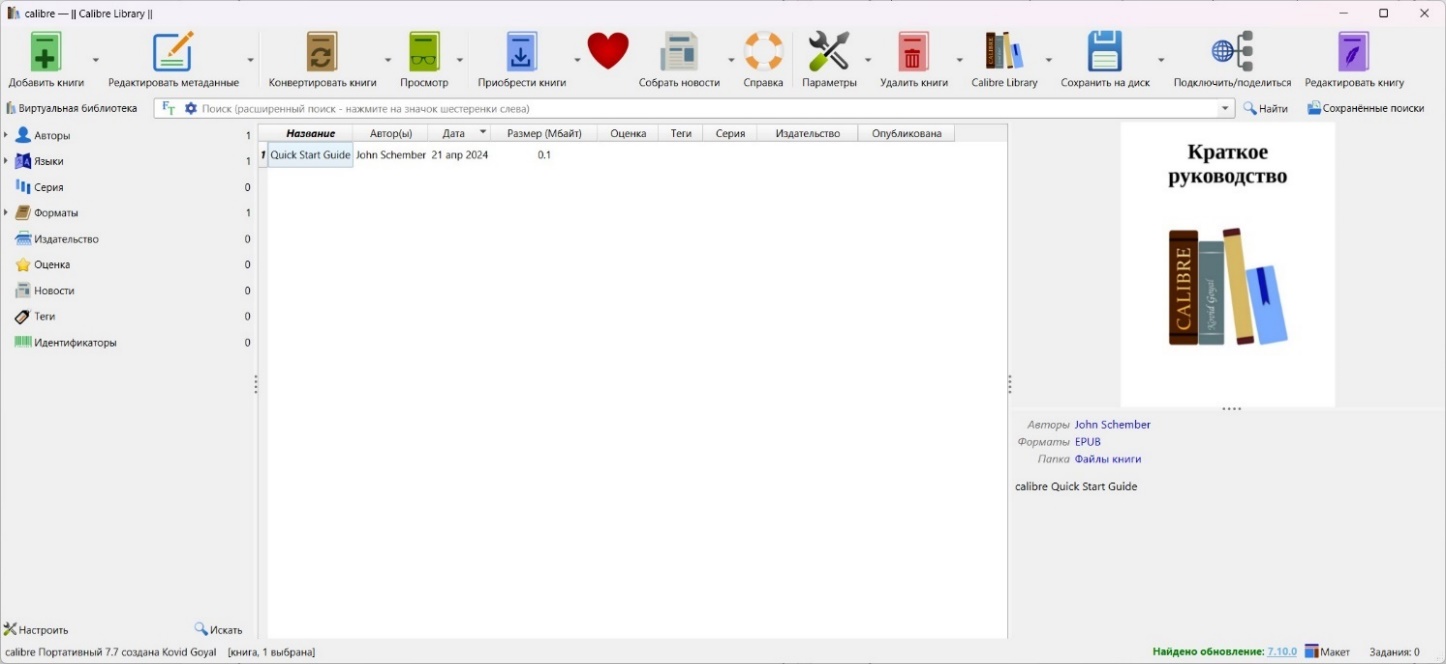


Рисунок 1 – Окно программы Calibre

### 1.6.2 Kavita

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

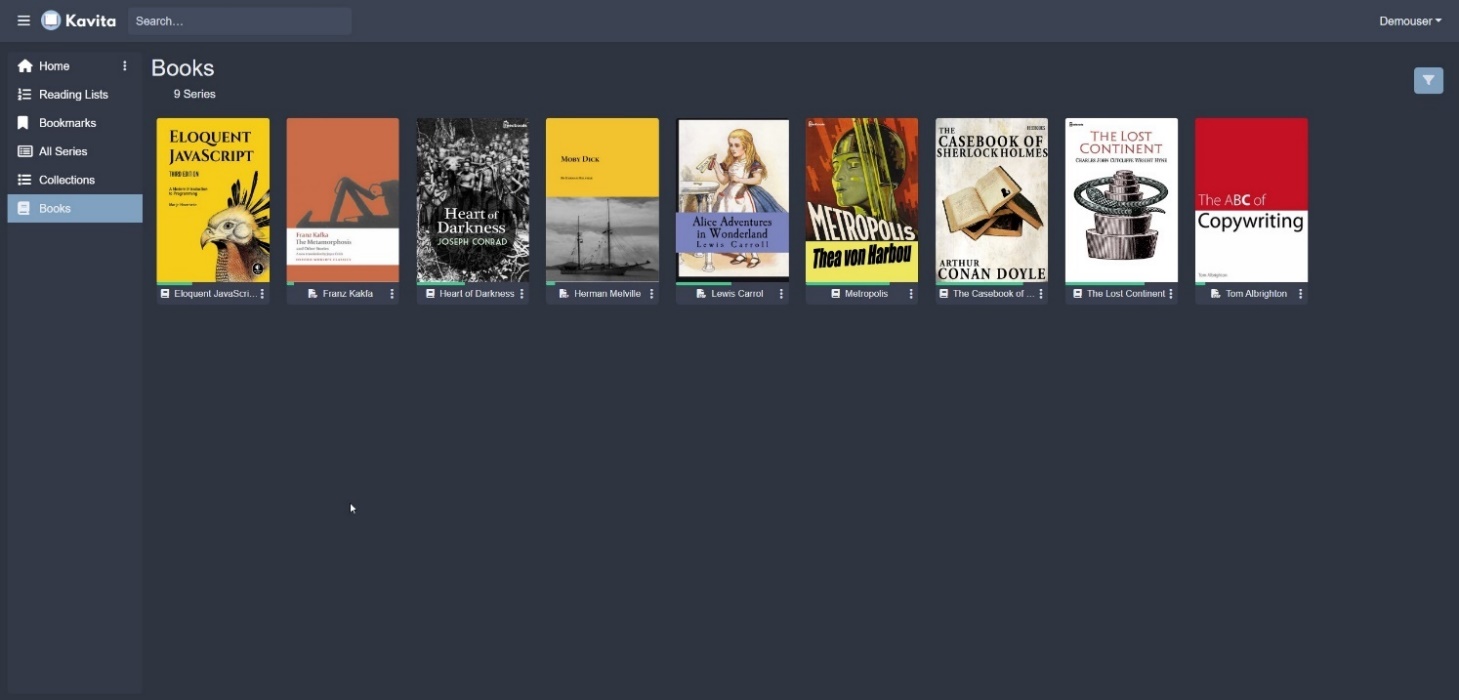


Рисунок 2 – Веб-страница программы Kavita

### 1.6.3 Koodo

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

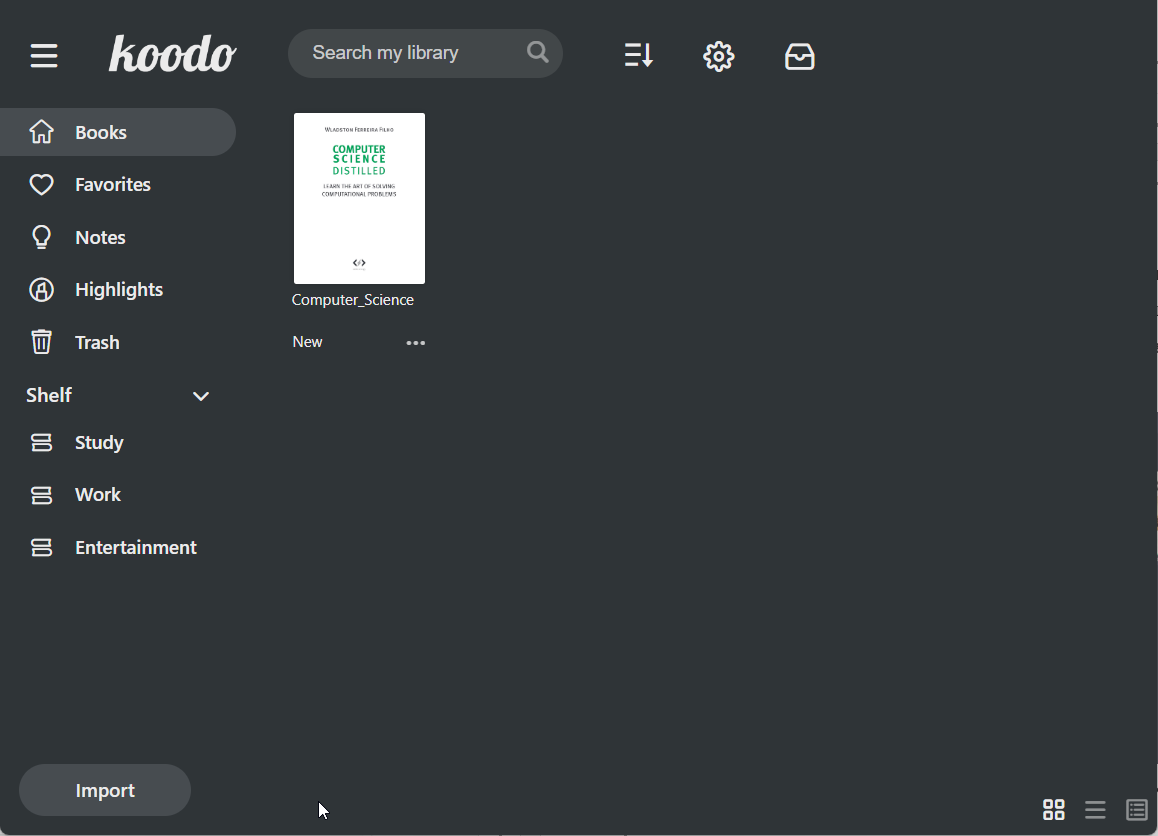


Рисунок 3 – Страница программы Kodoo

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

Таблица №1.1 - Сравнительный анализ аналогичных программных продуктов

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

# 2 Информационное обеспечение системы

## 2.1 Выбор средств управления данными

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

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

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

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

Из NoSQL БД можно считать СУБД Redis, которая позволяет быстро записать и считывать данные. Она хранит данные 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 SQL Server от Microsoft (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 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 приложении малых размеров.

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 такие как массив, JSON, перечисления (enum) и т.п. Последняя версия PostgreSQL – 16. Распространяется под лицензией PostgreSQL License, которая является open-source лицензией, схожая с BSD и MIT лицензией.

## 2.2 Проектирование базы данных

Диаграмма базы данных представлена на рисунке 4.

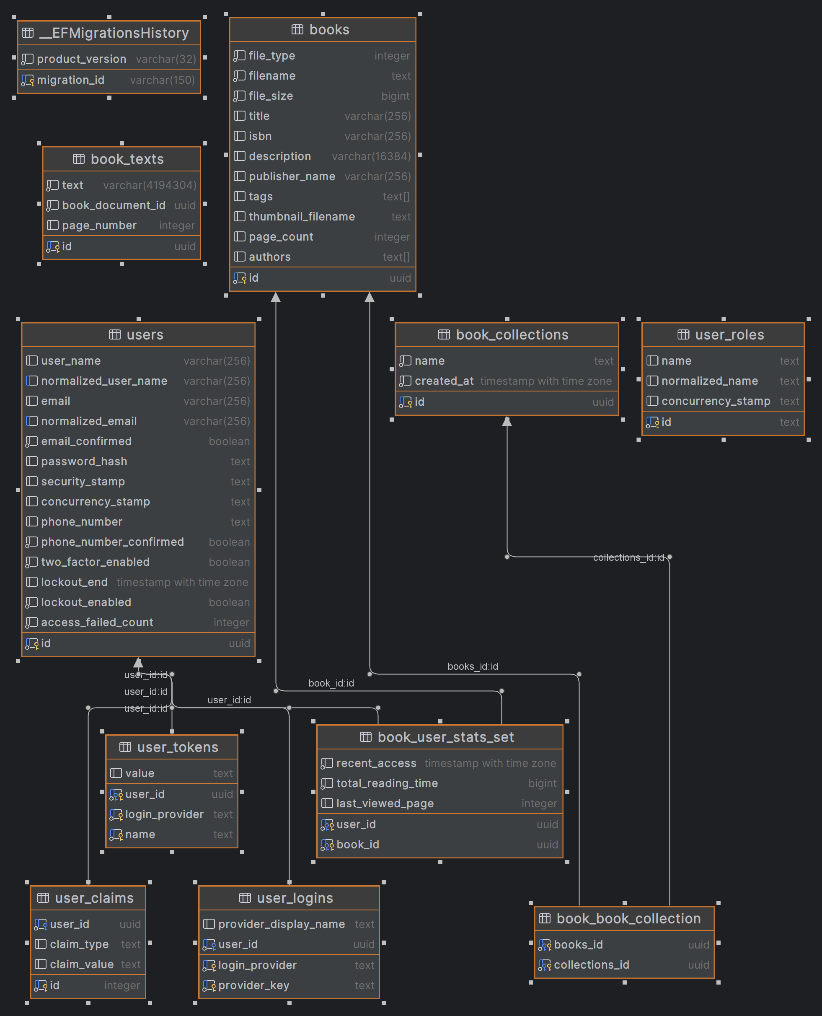


Рисунок 4 – Схема базы данных

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

Таблица №2.1 – Атрибуты сущности «Book»

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

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

Таблица №2.2 – Атрибуты сущности «BookCollection»

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

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

Таблица №2.3 – Атрибуты сущности «BookText»

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

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

Таблица №2.4 – Атрибуты сущности «BookUserStats»

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

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

Таблица №2.5 – Атрибуты сущности «DictionaryWord»

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

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

Таблица №2.6 – Атрибуты сущности «DictionaryWordAlias»

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

Также как с сущностью «Альтернативная форма слова» сущность «Определении 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 №.

Таблица № – Атрибуты сущности «Пользователь»

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

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

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

# 3 Алгоритмическое обеспечение системы

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

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

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

Логическое описание: схема алгоритма №

## 3.2 Алгоритм добавления электронной книги в систему

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 №

# 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 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 наибольшие возможности рефакторинга (преобразования в определенный вид) кода и 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 разрабатываемой системой не требуется вспомогательное ПО.

## 4.3 Разработка прикладного программного обеспечения

### 4.3.1 Структура прикладного программного обеспечения

Весь код разрабатываемой системы, серверной и клиентской части, находится в одном репозитории. Клиентская часть находится в директории “web”, а код серверной части находится в директории “src”. Структура решения серверной части изображена на рисунке #.

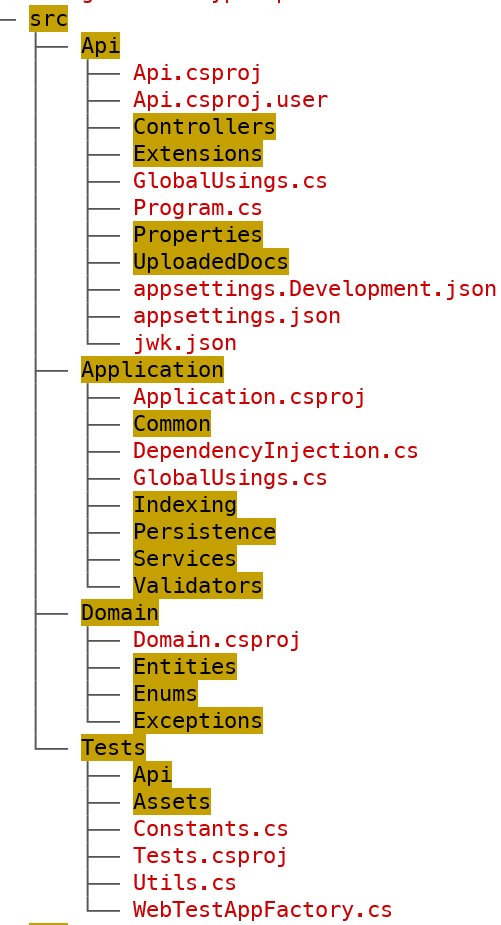


Рисунок # - Структура серверной части

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 и перечисления 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.

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

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

## 4.4 Особенности реализации, эксплуатации и сопровождения системы

Сложность реализации системы можно оценить, как высокую.

Разработчику системы необходимо знать языки программирования C#, 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.

## 4.5 Руководство пользователя

### 4.5.1 Требования к условиям эксплуатации

-

### 4.5.2 Инсталляция и настройка

Для серверной части приложения потребуется платформа для запуска приложении в виде «контейнеров» Docker.

### 4.5.3 Порядок и особенности работы

# 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;
* тестируемой функция возвращает определенный результат;
* полученный результат сравнивается с ожидаемым результатом и, если они не равны, выводится ошибка, иначе выводится, что тест успешно пройден.

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

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

Для серверной части были написаны интеграционные тесты. Код тестов расположен в проекте «src/Testing». Для их работы используется пакет XUnit версии 2.8.0.

Все тесты для клиентской части системы выполнены вручную с использованием методологии черного ящика.

## 5.1 Условия и порядок тестирования

## 5.2 Исходные данные для контрольных примеров

-

## 5.3 Результаты тестирования

-

# 6 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## 6.1 Расчет показателя трудоёмкости для программного продукта

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

,

где – общая трудоемкость разработки программного продукта;

– трудоемкость работ на i-й стадии разработки;

n – общее количество этапов разработки.

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

Таблица 6.1 – Поэтапная и общая оценка трудоемкости разработки

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

## 6.2 Расчет затрат на материальные ресурсы

Расчет затрат на материальные ресурсы производится по формуле:

,

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

– цена за единицу i-го вида материального ресурса;

i – вид материального ресурса;

n – общее количество всех видов материальных ресурсов.

Результаты расчетов затрат на материальные ресурсы приведены   
в таблице 6.2.

Таблица 6.2 – Затраты на материальные ресурсы

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

При разработке системы не возникло необходимости в использовании расходных материалов.

Общая сумма затрат на электроэнергию рассчитывается по формуле:

,

где – паспортная мощность i-го электрооборудования, кВт;

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

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

i – вид электрооборудования;

n – количество электрооборудования.

Необходимые расчеты затрат на электроэнергию приведены в таблице 6.3.

Таблица 6.3 – Затраты на электроэнергию

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

## 6.3 Расчёт затрат на разработку системы

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

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

## 6.4 Расчёт затрат на оплату труда

Общая сумма затрат на оплату труда определяется по формуле:

,

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

– время на разработку системы, ч;

i – порядковый номер работника;

n – количество работников.

Часовая заработная плата программиста рассчитывается по формуле:

,

где – среднемесячная заработная плата разработчика системы, руб.;

– среднемесячный фонд рабочего времени.

Стоимость одного часа работы программиста равна 300 руб.

Общая сумма затрат на оплату труда равна:

550 ч. × 300 руб. = 165 000 руб.

## 6.5 Расчет отчислений на социальные нужды

Данные об отчислениях на социальные нужды представлены в таблице 14.

Таблица 6.4 – Отчисления на социальные нужды

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

## 6.5 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 |

## 6.6 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.

## 6.7 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 %.

## 6.8 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, рассчитана экономическая часть проекта.

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

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. – введ. 01.01.92. – М. : Изд-во стандартов, 2010. – 24 с.
2. Документация Angular Material: [Электронный ресурс] / Google LLC, 2024. – Режим доступа: <https://material.angular.io/components/categories>, свободный.
3. Документация Angular: [Электронный ресурс] / Google LLC, 2024. – Режим доступа: <https://angular.io/docs>, свободный.
4. Документация ASP.NET: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0>, свободный.
5. Обзор Docker. Документация Docker: [Электронный ресурс] / Docker Inc., 2024. – Режим доступа: [https://docs.docker.com/get-started/overview](https://docs.docker.com/get-started/overview/), свободный.
6. Документация Entity Framework Core: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/ef/core/>, свободный.
7. A tour of C# language: [Электронный ресурс] / Microsoft, 2024. – Режим доступа: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>, свободный.
8. Ajay D. Kshemklayni. Distributed Computing: Principles, Algorithms, and Systems / Ajay D. Kshemklayni, Mukesh Singhal. – Cambridge University Press, 2008.
9. Artificial Intelligence: A Modern Approach, 4th US ed. / Stuart J. Russell., Peter Norvig. – Prentice Hall, 2020
10. Distributed Systems / Marteen Van Steen, Tannenbaum A.S.

# ПРИЛОЖЕНИЕ А

Код «Program.cs»

-