

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ

«ЭЛЕКТРОННАЯ БИБЛИОТЕКА»

Студент

Духнов - 23.05.24 Духнов Олег Олегович
(подпись, дата)

Руководитель

Юрова 02.06.24 Юрова Ольга Викторовна
(подпись, дата)

Нормоконтролёр

Бычков 02.06.24 Бычков Андрей Станиславович
(подпись, дата)

Работа допущена к защите (протокол заседания. кафедры от 27.05.2024г. № 11)

Заведующий кафедрой

М.А. Митрохин
(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

10.06.2024 № 14)

Секретарь ГЭК

Гурин
(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 19 » 04 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

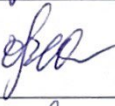
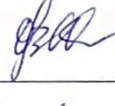
1. Студент Духнов Олег Олегович гр.20ВВП2 факультета вычислительной техники
2. Тема работы: Клиент-серверное приложение «Электронная библиотека»
Тема утверждена приказом ПГУ № 405/о от 19 апреля 2024 г.
3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О.В.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать клиент-серверное приложение «Электронная библиотека» для возможности чтения книг онлайн. Приложение должно позволять пользователям искать книги в библиотеки, загружать книги на сервер и скачивать их с сервера. Также приложение должно позволять осуществлять регистрацию и авторизацию пользователя, редактировать информацию о пользователе. Приложение должно быть спроектировано с использованием клиент-серверной архитектуры. Клиентская часть программы должна обладать понятным графическим интерфейсом. Сервер должен включать в себя сохранение данных в базу данных, передачу, редактирование и удаление их из нее. Клиентская программа должна позволять пользователю просматривать и читать книги и получать их с сервера.

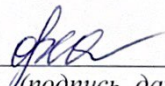
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки клиент-серверного приложения. Также необходимо изучить технологий для разработки приложения, такие как WPF, PostgreSQL и т.д.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2024-30.04.2024	
2.	Анализ требований	10	01.05.2024-02.05.2024	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2024-07.05.2024	
4.	Описание алгоритмов	40	08.05.2024-11.05.2024	
5.	Разработка клиент-серверного приложения «Электронная библиотека»	150	12.05.2024-29.05.2024	
6.	Оформление пояснительной записки	24	30.05.2024-31.05.2024	

Дата выдачи задания 19 апреля 2024 г.

Руководитель бакалаврской работы  Юрова О.В.
(подпись, дата)

Задание к исполнению принял студент  Духнов О.О.
(подпись, дата)

Работу к защите допустить
Декан факультета  06.06.2024 Фионова Л.Р.

Реферат

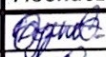
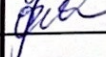
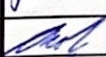
Бакалаврская работа содержит 105 л., 42 рис., 7 табл., 6 источн. и 3 прил.

КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, POSTGRESQL, PGADMIN4, VISUAL STUDIO, MVC, WPF, C#, .NET, JSON, NPGSQL

Цель работы – разработать клиент-серверное приложение, позволяющее читать книги онлайн.

Разработанное приложение должно позволять пользователям рекомендовать книги, иметь возможность искать книги по критериям и загружать книги на сервер. Также приложение должно позволять осуществлять регистрацию и авторизацию пользователя, редактировать информацию о пользователе.

Результатом проделанной работы является проект, который удовлетворяет поставленной задаче. А именно, разработано, реализовано и протестировано приложение для чтения книг пользователям. При разработке было использовано большое количество технологий, к которым относятся: язык программирования C#, платформа .NET, язык структурированных запросов SQL, библиотека для работы с базами данных в C#-приложениях NPGSQL, WPF для написания интерфейса приложения, среда разработки Visual Studio. Для хранения данных выбрана СУБД PostgreSQL, а для её администрирования применялся инструмент PGAdmin4.

					ПГУ 1.090301.28.001 ПЗ							
Изм.	Лист	№ докум.	Подпись	Дата	Клиент-серверное приложение «Электронная библиотека» Пояснительная записка			Лит.	Лист	Листов		
Разраб.		Духнов О.О.		2024							2	105
Провер.		Юрова О.В.		2024								
Н. Контр.		Бычков А.С.		2024								
Утв.		Митрохин М.А.		2024				ФВТ, каф. ВТ гр. 20ВВП2				

Содержание

Перечень сокращений и обозначений	7
Введение	8
1 Анализ предметной области	9
1.1 Анализ темы	9
1.2 Анализ аналогов приложений.	10
1.2.1 Онлайн-сервис «ЛитРес»	10
1.2.2 Онлайн-библиотека «MyBook»	11
1.2.3 Онлайн-сервис «Строки»	13
1.2.4 Вывод	14
1.3 Постановка задачи	16
2 Выбор технологий и средств разработки приложения	21
2.1 C#	21
2.2 .Net	22
2.3 Visual Studio	23
2.4 PostgreSQL	23
2.5 Npgsql	24
2.6 PgAdmin 4	24
2.7 Json	24
2.8 WPF	25
2.9 MaterialDesignThemes	25
3 Архитектура клиент-серверного приложения «Электронная библиотека». . .	26
3.1 Клиент – серверная архитектура	26
3.2 Архитектура приложения	27
4 Реализация приложения	30
4.1 Описание работы приложения	30
4.2 Структура БД	31

4.3	Программная реализация приложения	35
4.3.1	ClientGUI	35
4.3.2	MyHelper	37
4.3.3	Server	39
4.4	Основные алгоритмы взаимодействия клиента и сервера	41
5	Тестирование приложения	48
5.1	Регистрация пользователя	48
5.2	Авторизация пользователя	50
5.3	Просмотр, скачивание и чтение книги	52
5.4	Редактирование аккаунта пользователя	55
5.5	Поиск по критериям	58
5.6	Добавление книги в личную библиотеку	59
6	Безопасность жизнедеятельности	62
6.1	Вред здоровью при работе с ПК	62
6.2	Снижение вреда для здоровья при работе с ПК	62
6.3	Техника безопасности при работе с компьютером	63
	Заключение	64
	Список использованных источников	65
	Приложение А – UML-диаграммы	66
	Приложение А.1-Диаграмма последовательностей	67
	Приложение А.2-Диаграмма развертывания	68
	Приложение А.3-Диаграмма классов	69
	Приложение А.4-Диаграмма прецедентов	70
	Приложение Б – Листинг программы	71
	Приложение В - Презентация	97

Перечень сокращений и обозначений

БД – база данных

ВЦИОМ – всероссийский центр изучения общественного мнения

МТС – мобильные телесистемы

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базой данных

CLR (от англ. Common Language Runtime) – исполняющая среда для байт-кода, в которой компилируется программа

FTP (от англ. File Transder Protocol) – протокол передачи данных по сети

HTTP (от англ. Hypertext Transfer Protocol) – протокол прикладного уровня передачи данных

JSON (от англ. JavaScript Object Notation) – текстовый формат для обмена структурированными данными и для хранения их

MVC (от англ. Model-View-Controller) — шаблон (паттерн) программирования, разделяющий архитектуру приложения на три модуля

SQL (от англ. Structured Query Language) - язык структурированных запросов

TCP/IP (от англ. Transmission Control Protocol/Internet Protocol) – сетевая модель, описывающая процесс передачи цифровых данных

UML (от англ. Unified Modeling Language) – унифицированный язык моделирования

WPF (от англ. Windows Presentation Foundation) – система для построения клиентских приложений для ОС Windows

Введение

В современном мире, где информация и знания становятся всё более доступными, онлайн-библиотеки для чтения книг являются неотъемлемой частью нашей жизни. Они предоставляют читателям удобный доступ к обширным коллекциям литературы, позволяя выбирать книги по интересам, жанру или автору. В этой работе будут рассмотрены основные аспекты функционирования онлайн-библиотек, их преимущества и недостатки перед физическими аналогами, перспективы развития этого направления, а также разработка и реализация приложения.

Актуальность данной темы обусловлена растущей популярностью онлайн-библиотек благодаря своей удобной доступности и возможности получения информации в любое время и из любой точки мира.

Подводя итог, хочется отметить, что разработка приложения онлайн-библиотеки является востребованной задачей, так как позволяет облегчить процесс поиска, хранения и предоставления книг пользователям.

					ПГУ 1,090301,28.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Анализ темы

В настоящее время все больше и больше людей предпочитают электронную книгу – физической. Это не удивительно, так как благодаря современным информационным технологиям есть возможность пользоваться огромным количеством книг из любой точки мира.

Еще одним большим преимуществом электронной библиотеки перед физической – это хранение книг. Для хранения печатных книг требуется выделять отдельное место, а электронная хранится удалено на сервере, либо же у пользователя на устройстве. Из-за такого способа хранения книг у людей есть возможность получить доступ ко всей своей библиотеке, где угодно и на любое устройство.

Стоит отметить и недостатки приложений, а именно: необходимость подключения к интернету, так как без доступа к сети не удастся пользоваться приложением в полной мере; зависимость от сервера, если сервер выйдет из строя, то некоторые данные будут недоступны пользователю.

Кто бы что не говорил, современно общество охотно продолжает читать книги в любом формате. Согласно официальной статистике, ВЦИОМ от 26 мая 2022 года, 92% опрошенных Россиян говорят, что читают книги. Из них 58% читали в бумажном формате, а в электронном 71%. Статистика по устройствам следующая – с смартфонов читает каждый третий, а именно 35%. 20 % читают с экранов компьютеров. Электронные книги выбрали 16% [1].

Исходя из данных статистике ВЦИОМ следует отметить тот факт, что приложения для чтения книг в электронном формате постепенно начали вытеснять бумажные версии. Со временем данная тенденция будет сохраняться, и большинство людей и вовсе откажутся от книг в печатном формате.

Проанализировав данную тему хочется сказать, что приложение для чтения электронных книг будет требовать спросом среди людей.

					ПГУ 1,090301,28.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1.2 Анализ аналогов приложений

Электронные приложения для чтения книг — это цифровые платформы и программы, которые позволяют пользователям читать книги на различных устройствах, таких как смартфоны, планшеты и персональные компьютеры. На рынке существует много различных приложений, которые позволяют читать книги в электронном виде. Эти приложения предоставляют доступ к обширной библиотеке журналов, книг, газет и других текстовых материалов.

Для более правильной реализации приложения, необходимо проанализировать аналоги, выявить их слабые и сильные стороны, а также определить основные требования к разрабатываемому приложению.

В этом пункте будут рассмотрены три самых популярных приложения для чтения книг, а именно:

- онлайн-сервис «ЛитРес»;
- онлайн-библиотека «MyBook»;
- онлайн-сервис «Строки».

1.2.1 Онлайн-сервис «ЛитРес»

Интерфейс онлайн-сервиса «ЛитРес» представлен на рисунках 1 и 2.

Основная особенность данного сервиса — это подписочная система, которая предоставляет доступ к более чем 400 000 электронных и аудиокниг.

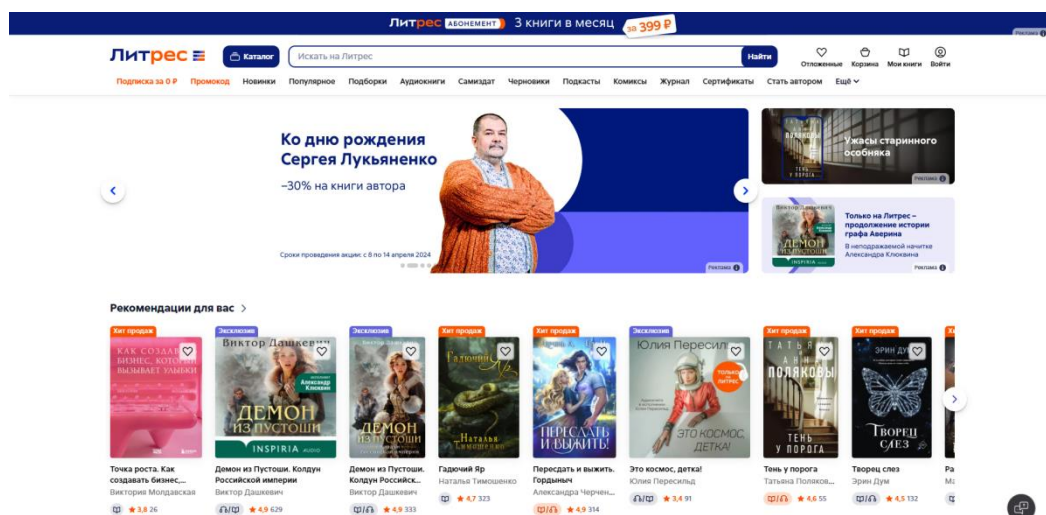


Рисунок 1 - Главная страница сервиса

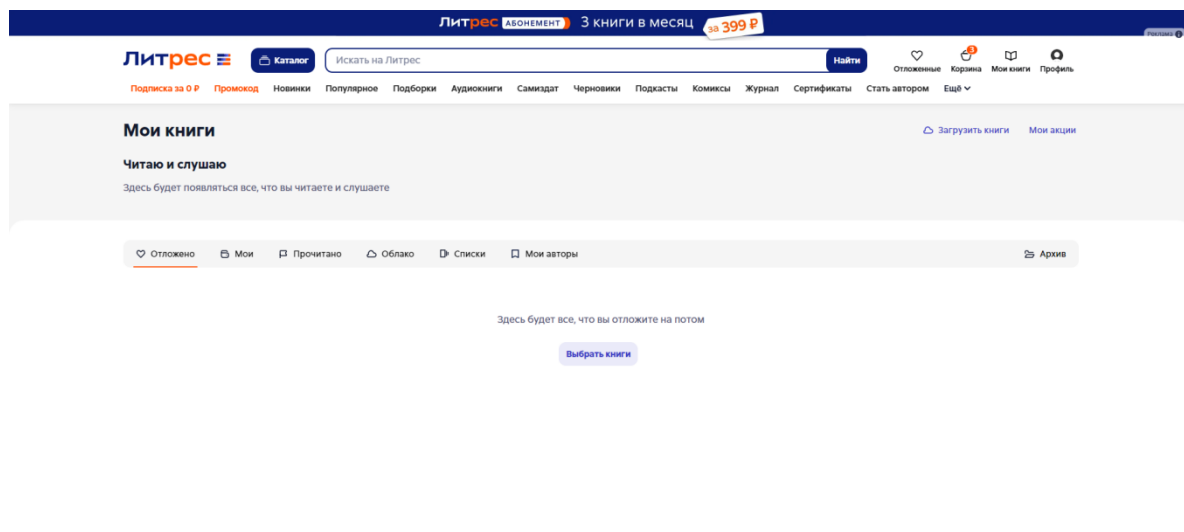


Рисунок 2 - Страница с библиотекой клиента

Онлайн-сервис «ЛитРес» имеет большое количество различных функций, предоставляемые пользователю. Ниже представлены основные из них:

- возможность чтения и прослушивания книг. У пользователя есть выбор, каким образом изучать материал;
- разнообразие контента. В «ЛитРес» собрано большое количество книг разных авторов, жанров и издательств;
- доступ к библиотеке книг. Пользователь может читать книги на любом устройстве, где есть выход в интернет;
- поиск книг. В приложении реализован удобный поиск, который может найти нужную книгу.
- Загрузка книг с устройства. В онлайн-сервисе «ЛитРес» реализована возможность загружать собственные книги в облако.

Основные недостатки данного онлайн-сервиса – это отсутствие бесплатных электронных книг и перегруженность интерфейса.

1.2.2 Онлайн-библиотека «MyBook»

Интерфейс онлайн-библиотеки «MyBook» представлен на рисунках 3 и 4.

Главные достоинства данной библиотеки – это оформление одной подписки для получения всех имеющихся книг в приложении, а также большой выбор бесплатных книг.

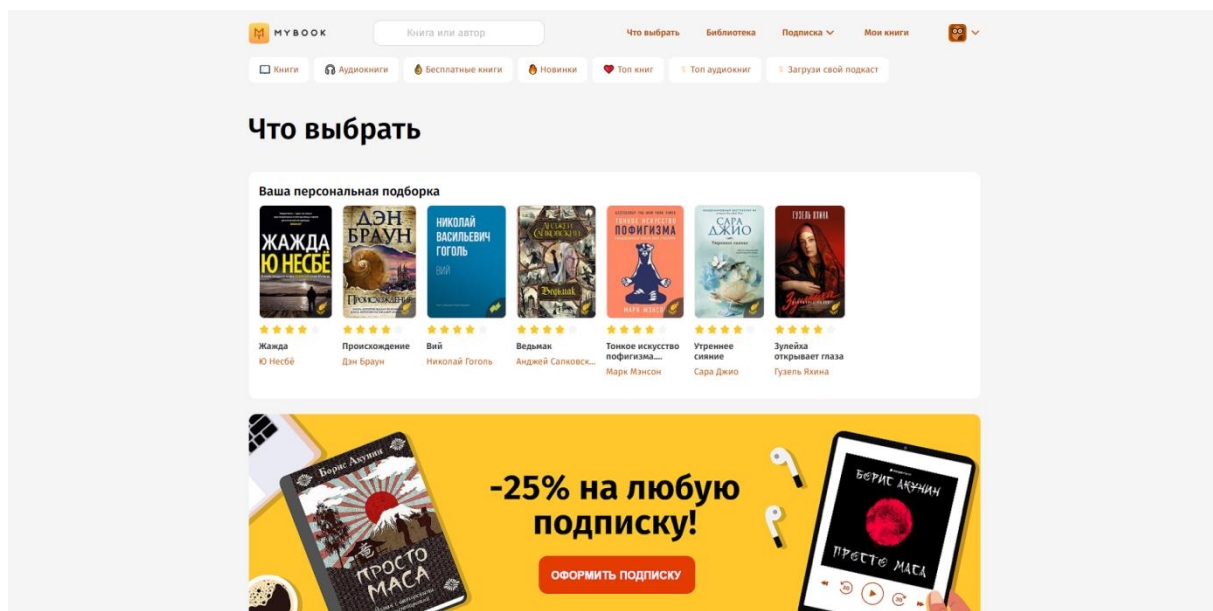


Рисунок 3 - Главная страница онлайн-библиотеки

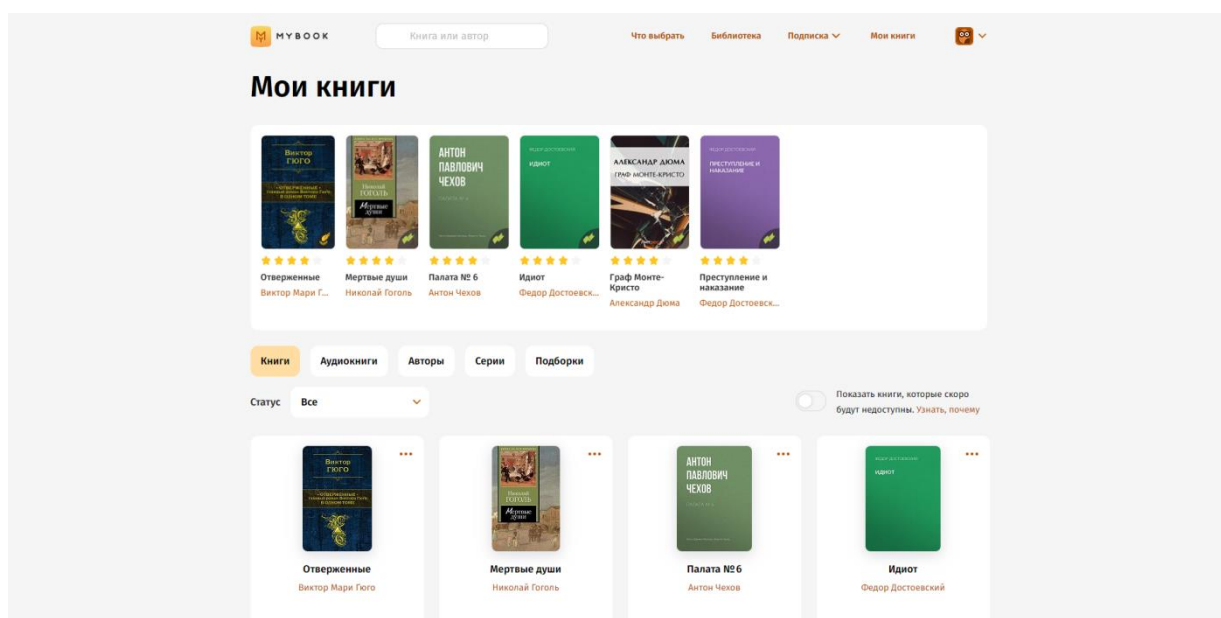


Рисунок 4 - Страница с библиотекой клиента

Онлайн-библиотека «MyBook» предоставляет большое количество различных услуг пользователю. Основные функции приложения представлены ниже:

- широкий выбор книг. Приложение «MyBook» предлагает большой выбор книг различных жанров и авторов;
- синхронизация. Онлайн-библиотека «MyBook» позволяет синхронизировать книги между разными устройствами;

- удобный интерфейс. Интерфейс данного приложения понятный и удобный;
- поиск книг. В приложении реализован удобный поиск по критериям;
- аудиокниги. В «MyBook» есть возможность прослушивать информацию в аудиоформате.

Основной недостаток данной библиотеке – отсутствие некоторых популярных книг, которые доступны в аналогах.

1.2.3 Онлайн-сервис «Строки».

Интерфейс онлайн-сервиса «Строки» представлены на рисунках 5 и 6.

Основное преимущество данного сервиса перед конкурентами – эксклюзивность некоторых изданий, которые принадлежат издательству МТС.

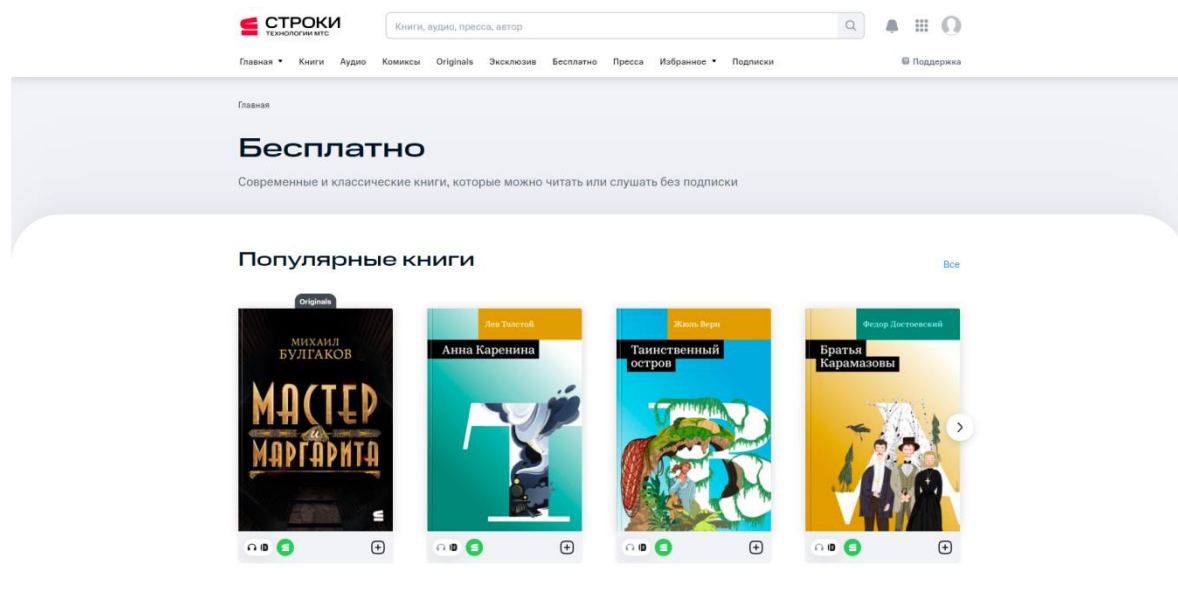


Рисунок 5- Главная страница сервиса

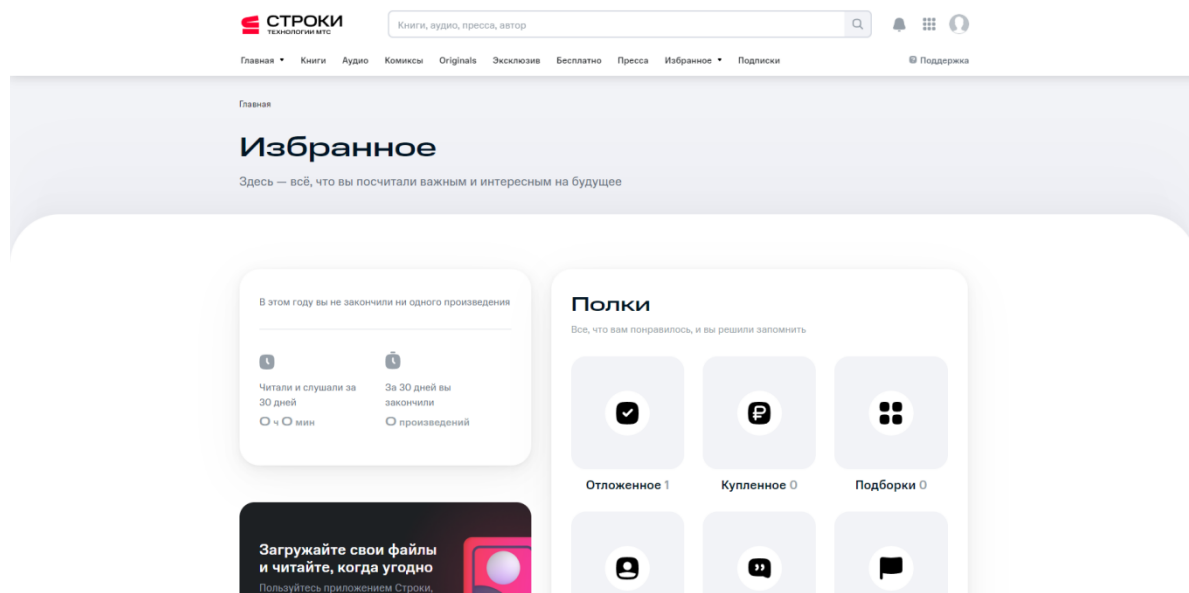


Рисунок 6 – Страница с избранными книгами.

Онлайн-сервис «Строки» оказывает большой выбор услуг для чтения электронных книг. Основные возможности перечислены ниже:

- синхронизация. Приложение позволяет синхронизировать контент между разными устройствами;
- наличие аудио книг. Некоторые книги доступны в аудио формате;
- поиск книг. В онлайн-сервисе «Строки» реализован понятный поиск по книгам и авторам;
- удобный интерфейс. Интерфейс онлайн-сервиса «Строки» понятный и интуитивный.

Главный недостаток онлайн-сервиса – чтение книг возможно только с приложения, установленного на телефон.

1.2.4 Вывод

Исходя из анализа аналогово следует отметить, что все они являются хорошими и конкурентноспособными приложениями для чтения электронных версий книг. У каждого есть свои плюсы и минусы, особые функции, которые выделяются среди других похожих приложений.

Результаты анализа существующих аналогов можно представить в таблице 1, показывающие минусы и плюсы рассматриваемых сервисов.

					ПГУ 1,090301,28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

Таблица 1 – Достоинства и недостатки онлайн-библиотек

Функционал	«ЛитРес»	«MyBook»	«Строки»
Удобный и понятный интерфейс	-	+	+
Загрузка собственных книг в облако	+	-	-
Чтение книг оффлайн	+	-	-
Чтение книг онлайн	+	+	-
Поиск по критериям	+	+	+
Наличие бесплатных книг	-	+	+

Проанализировав данные онлайн-библиотеки, было решено разработать собственное приложение для чтения электронных книг, которое будет называться DigitalLybrary. Данное приложение будет соответствовать основным требованиям, которые предоставляет пользователь к онлайн библиотеки, а именно: удобный интерфейс; все основные функции, необходимые для комфортного использования; возможность авторизованным пользователям просматривать книги онлайн или скачивать на устройство; возможность удобного поиска книг; возможность использовать приложение для чтения книг, не имея выхода в интернет.

Помимо базового функционала, будут реализованы функции, которые отсутствуют у аналогов и которые будут привлекательны конечному пользователю. А именно будет реализована система друзей, которая предоставляет следующие возможности: пользователю будут предлагаться в рекомендации книги друзей; будет возможность рекомендовать свои книги друзьям; возможность просматривать библиотеку у друзей.

Исходя из анализа темы, 20% людей используют персональные компьютеры в качестве устройства для чтения книг, поэтому было принято решение сделать приложение для этой целевой аудитории.

Конкуренция среди Web-приложений для чтения книг слишком высока, поэтому разрабатываться будет десктопное приложение, так как среди аналогов данных приложений нет. Есть читалки, которые позволяют просматривать в различных форматах книги, но десктопной онлайн-библиотеки на рынке нет.

Преимуществами «DigitalLibrary» перед другими онлайн-библиотеками и электронными читалками будет заключаться в том, что пользователю будет доступен функционал и онлайн-библиотек, и электронных читалок, а также данный функционал будет расширен своими интересными «фишками», которых нет в аналогах.

1.3 Постановка задачи

Приложение «Электронная библиотека» должно реализовывать следующий функционал:

Со стороны клиента:

- загрузка и скачивание книг с сервера;
- форма для авторизации и регистрации;
- возможность чтения книг онлайн и оффлайн;
- список прочтенных, желаемых и в процессе чтения книг;
- система друзей;
- ведение статистики за пользователем;
- редактирование профиля;
- поиск по критериям;
- загрузка собственных книг в облако;
- запись цитат.

Со стороны сервера:

- хранение всех данных клиента;
- предоставление данных о пользователе, книгах и его друзьях;
- добавление, редактирование и удаление пользователей, книг;
- возможность обслуживания несколько клиентов одновременно.

Минимальные системные требования:

- операционная система: Windows (7, 8, 10, 11);
- минимальная скорость интернета: 1 Мбит/сек;
- минимальный объем оперативной памяти: 2 ГБ (для Windows 11 требуется 4 ГБ оперативной памяти);

					<i>ПГУ 1,090301,28.001 ПЗ</i>	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

– минимальный объем свободной памяти на жестком диске: 120 МБ.

Данное приложение имеет функционал, который разделен на три категории, а именно:

- для неавторизованного пользователя;
- для авторизованного пользователя;
- для пользователя, у которого отсутствует подключение к серверу.

Диаграмма вариантов использования приложения для неавторизованного пользователя представлена на рисунке 7.

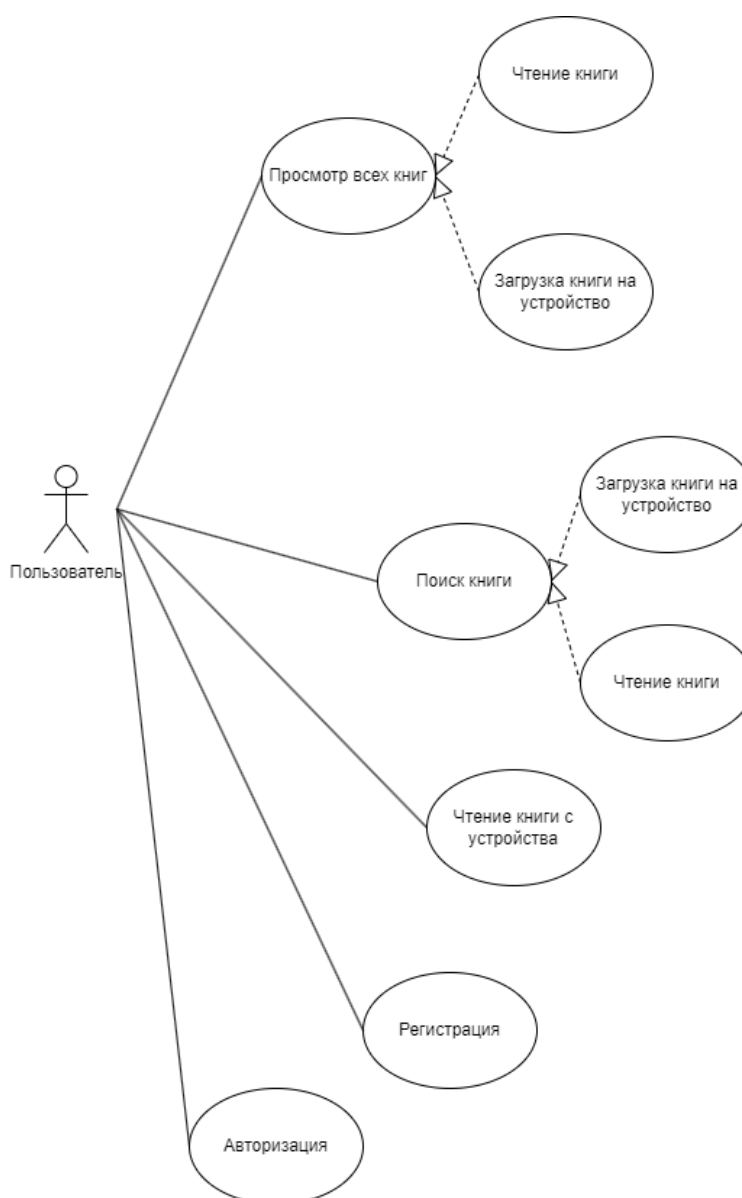


Рисунок 7 – UML диаграмма вариантов использования неавторизованного пользователя

Неавторизованный пользователь имеет возможность пользоваться ограниченным набором функций, а именно:

- просмотр списка доступных книг, чтение книг онлайн;
- авторизация в приложении;
- регистрация в приложении;
- чтение книги с устройства;
- загрузка книги на устройство;
- поиск по критериям книг.

Авторизированному пользователю будет доступен расширенный функционал работы программы. UML диаграмма представлена на рисунке 8.

					ПГУ 1,090301,28.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 8 – UML диаграмма вариантов использования авторизированного пользователя

Функциональные возможности, доступные авторизированному пользователю:

- просмотр списка доступных книг, чтение книг онлайн;
- настройка профиля;
- просмотр профиля;
- чтение книги с устройства;
- загрузка книги на устройство;
- добавление и удаление книги в библиотеку;
- добавление и удаление книг в список любимых;
- просмотр личной библиотеки;
- поиск по критериям книг.

У пользователя должна присутствовать возможность пользоваться приложением, даже если не удалось подключиться к серверу по причинам сбоя сервера или отсутствия подключения к интернету, поэтому должен быть реализован соответствующий функционал.

UML диаграмма представлена на рисунке 9.



Рисунок 9 – Диаграмма пользователя при отсутствующем подключении к серверу

В данном случае у пользователя будет возможность читать книги со своего устройства.

2 Выбор технологий и средств разработки приложения

В данном разделе будут рассмотрены технологии и средства разработки, которые были использованы для реализации программы онлайн-библиотеки «DigitalLibrary».

Ниже приведены технологии и средства разработки, которые были выбраны:

- C#;
- .NET;
- Visual Studio;
- PostgreSQL;
- Npgsql;
- pgAdmin4;
- Json;
- WPF;
- MaterialDesignThemes;

2.1 C#

Язык программирования C# – это объектно-ориентированный язык программирования, который был разработан компанией Microsoft. Этот язык является универсальным и его используют для разработки десктопных приложений под различные операционные системы, написания игр, веб-сервисов, нейросетей и мобильных приложений [2].

Данный язык программирования является очень мощным, востребованным и быстро развивающимся. Он был написан на языке C, и целью C# было заменить язык программирования Java, чтобы упростить разработку различных приложений под Windows. Также стоит отметить, что C# уже несколько лет подряд входит в топ 5 самых популярных языков программирования.

Компания Microsoft активно развивает и обновляет язык C#, с каждой новой версией появляется много различного функционала, особенностей и так называемого «синтаксического сахара».

Есть множество различных инструментов в виде других языков программирования, например, С, С++, Java, Python, но для разработки онлайн-библиотеки был выбран именно этот язык программирования, потому что компания Microsoft создала целую инфраструктуру для языка С# в виде различных библиотек и инструментов, с помощью которых можно быстро создать эффективное, надежное и удовлетворяющая всем требованиям приложение под операционную систему Windows.

Если сравнивать С# с Python или Java, то выбор пал именно на первый только потому, что приложение под операционную систему Windows будет работать быстрее за счет оптимизации своего инструмента компанией Microsoft. Если сравнивать с С или С++, то С# был выбран из-за того, что процесс разработки приложения будет происходить быстрее, так как присутствует большое количество синтаксических особенностей, которые упрощают разработку, а также не нужно беспокоиться за работу с памятью программы.

2.2 .Net

Платформа .Net является набором технологий, разработанных компанией Microsoft для создания и выполнения приложений на различных устройствах и операционных системах.

Данная платформа включает в себя набор библиотек и инструментов для удобной и быстрой разработки различного программного обеспечения любого уровня сложности.

Стоит отметить, что в .Net есть общезыковая среда исполнения CommonLanguageRuntime (CLR), которая упрощает разработку приложений и компонентов. Эта среда обеспечивает выполнение определенного функционала, такого как сборка мусора, обработка исключений, безопасное использование памяти и отладка программы.

В этом приложении с помощью данной платформы будет написана серверной часть приложения и интерфейс пользователя.

					ПГУ 1,090301,28.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

2.3 Visual Studio

VisualStudio – это интегрированная среда разработки (IDE), разработанная компанией Microsoft. Данная IDE является мощным инструментом разработки, отладки и тестирования программ, написанных на языках программирования, разработанных компанией Microsoft, а именно C, C++, C#, F#, Visual Basic.NET. Также есть возможность разрабатывать на Python, JavaScript и других языках. Используется для разработки мобильных, десктопных, веб-приложений, а также игр.

Данной приложение было выбрано в качестве основной среды разработки из-за того, что оно очень хорошо интегрировано с платформой .NET и языком программирования C#, в VisualStudio удобно тестировать и отлаживать свое приложение, устанавливать необходимые пакеты с помощью удобного инструмента NuGet, также есть возможность использовать систему контроля версий.

Главный конкурент VisualStudio – это Rider. Данное IDE от компании JetBrains является очень хорошим и конкурентно способным, но был выбран продукт от компании Microsoft, так как он является бесплатным.

2.4 PostgreSQL

PostgreSQL – это кроссплатформенная объектно-реляционная система управления базами данных (СУБД), которая предоставляет большой выбор возможностей и функций для управления, хранения, организации и анализа данных. Данное СУБД является бесплатным и его можно устанавливать на любую операционную систему.

Основные особенности PostgreSQL:

- расширяемость за счет создания новых типов данных и функций;
- управление доступом на основе ролей, группировка пользователей по уровню прав;
- работа с большими объемами данных;
- наследование;

- поддержка большого количества различных типов данных;
- резервное копирование.

PostgreSQL был использован в проекте из-за того, что он быстрый, надежный, бесплатный и имеет удобные приложения и библиотеки для работы с ним.

2.5 Npgsql

Npgsql – это библиотека с открытым исходным кодом для работы с PostgreSQL. Данная библиотека позволяет разработчикам следующее: выполнять запрос к базе данных; получать результат запроса; управлять транзакциями; работать с параметрами и типами данных.

2.6 PgAdmin 4

PgAdmin 4 – это бесплатное кроссплатформенное приложение для управления базами данных PostgreSQL. Оно представляет собой клиентское приложение, которое предоставляет удобный и интуитивно понятный интерфейс для работы с базами данных, администрирования и мониторинга.

2.7 Json

Json (JavaScript Object Notation) – это формат для передачи и получения данных. Json имеет понятный и читаемый способ для представления структурированных данных, таких как массивы, объекты, структуры в виде текстовой строки. Этот формат данных широко используется для обмена данными между клиентом и сервером.

Формат данных Json имеет простую структуру, что в свою очередь позволяет написать Json-объект даже вручную. Также этот формат поддерживает большое количество различных типов данных, такие как строки, числа, массивы, объекты, логические значения, что позволяет представлять данные в разнообразных форматах.

Данный формат представления данных используется в приложении, так как уже реализован основной функционал для работы с ним, который легко и удобно использовать для передачи данных по сети. А также из-за того, что он имеет самый

					<i>ПГУ 1,090301,28.001 ПЗ</i>	Лист
						22
Изм.	Лист	№ докум.	Подпись	Дата		

маленький объем Json-файла при сериализации объекта класса среди других форматов для представления данных. Поэтому этот файл очень быстро передается по сети, что позволяет ускорить работу программы.

2.8 WPF

Windows Presentation Foundation – это специальный инструмент, который позволяет разрабатывать красивые и понятные приложения, с помощью доступных инструментов под операционную систему Windows [3].

Данный инструмент предоставляет разработчикам множество различных инструментов для написания качественных визуальных приложений, а именно: различные графические элементы, шаблоны, стили.

WPF имеет следующие особенности, а именно: графические операции выполняются с использованием видеокарты, что позволяет ускорить работу приложения; дизайн и логика разделены, но синхронизированы между собой, что позволяет более эффективно делать приложения; имеется большое количество визуальных элементов, которые легко настраиваются и расширяются.

В платформе .NET есть конкурент WPF, а именно Windows Form. Был выбран первый из-за того, что он предоставляет более широкий выбор инструментов для разработки пользовательского интерфейса, а также есть возможность верстать с помощью xaml.

2.9 MaterialDesignThemes

MaterialDesignThemes является библиотекой, которая содержит в себе стили и темы для упрощения и создания дизайна пользовательского интерфейса. Данная библиотека была разработана компанией Google и является одной из самых популярных среди всех библиотек в своем направлении [4].

Была выбрана в проекте данная библиотека стилей, потому что с помощью нее можно легко и быстро создать красивый и удобный пользовательский интерфейс.

3 Архитектура клиент-серверного приложения «Электронная библиотека»

3.1 Клиент – серверная архитектура

Модель взаимодействия, в которой есть два основных лица, а именно клиент и сервер, которые взаимодействуют друг с другом, называется «Клиент-серверной архитектурой».

Была выбрана именно эта архитектура, так как было основное требование для приложения, чтобы пользователи в любом уголке планеты могли получить данные своего аккаунта, а также все книги, которые находятся в собственной библиотеки.

Взаимодействие между клиентом и сервером происходит за счет сетевых протоколов, таких как HTTP, TCP/IP, FTP. Данные протоколы определяют правила взаимодействия между клиентом и сервером. В основе этих правил лежит инициирование запросов от клиента и отправкой его на сервер. В свою очередь последний получает запрос, обрабатывает его, подготавливает данные и передает их в виде ответа клиенту.

Можно выделить следующие основные преимущества клиент-серверной архитектуры: масштабируемость. Системы на основе данной архитектуры легко масштабируются, так как можно добавлять новых клиентов и серверы без значительных изменений в структуре системы; централизованность. Сервер обеспечивает централизованное управление данными и ресурсами, что облегчает контроль и безопасность системы; удобство обновлений. Данную архитектуру достаточно легко поддерживать и реализовывать новый функционал.

Основными недостаткам клиент-серверной архитектуры можно выделить: зависимость от сети. Клиенты должны постоянно иметь соединение с сервером, при отсутствии сети приложение не будет корректно функционировать; безопасность. Так как сервер хранит и обрабатывает важные данные пользователей, необходимо обеспечить защиту этих данных.

3.2 Архитектура приложения

В разделе «Архитектура приложения» будет рассмотрен паттерн программирования, а также взаимодействие компонентов в проекте. В качестве паттерна программирования был выбран шаблон MVC.

Одним из самых популярных шаблонов проектирования приложения является Model-View-Controller, который используется для разделения компонентов приложения на три основные части, а именно: модель, представление и контроллер. Каждая из этих частей выполняет свою определенную функцию, из-за чего упрощается разработка приложения и его поддержка. Архитектура данного проекта представлена на рисунке 10.

Модель отвечает за работу с данными приложения. Она представляет из себя структуру данных, бизнес-логику и методы доступа к данным и не зависит от представления или контроллера, а также обеспечивает изоляцию данных, что позволяет легко изменять данные без влияния на другие части приложения.

Представление отвечает за отображение данных для пользователя. Оно представляет собой интерфейс пользователя, который взаимодействует с моделью. Представление отображает данные из модели пользователя и передает пользовательские действия контроллеру.

Контроллер является связующим звеном между моделью и представлением. Он обрабатывает пользовательские запросы, взаимодействует с моделью для получения или изменения данных и передает данные в представление для отображения пользователю.

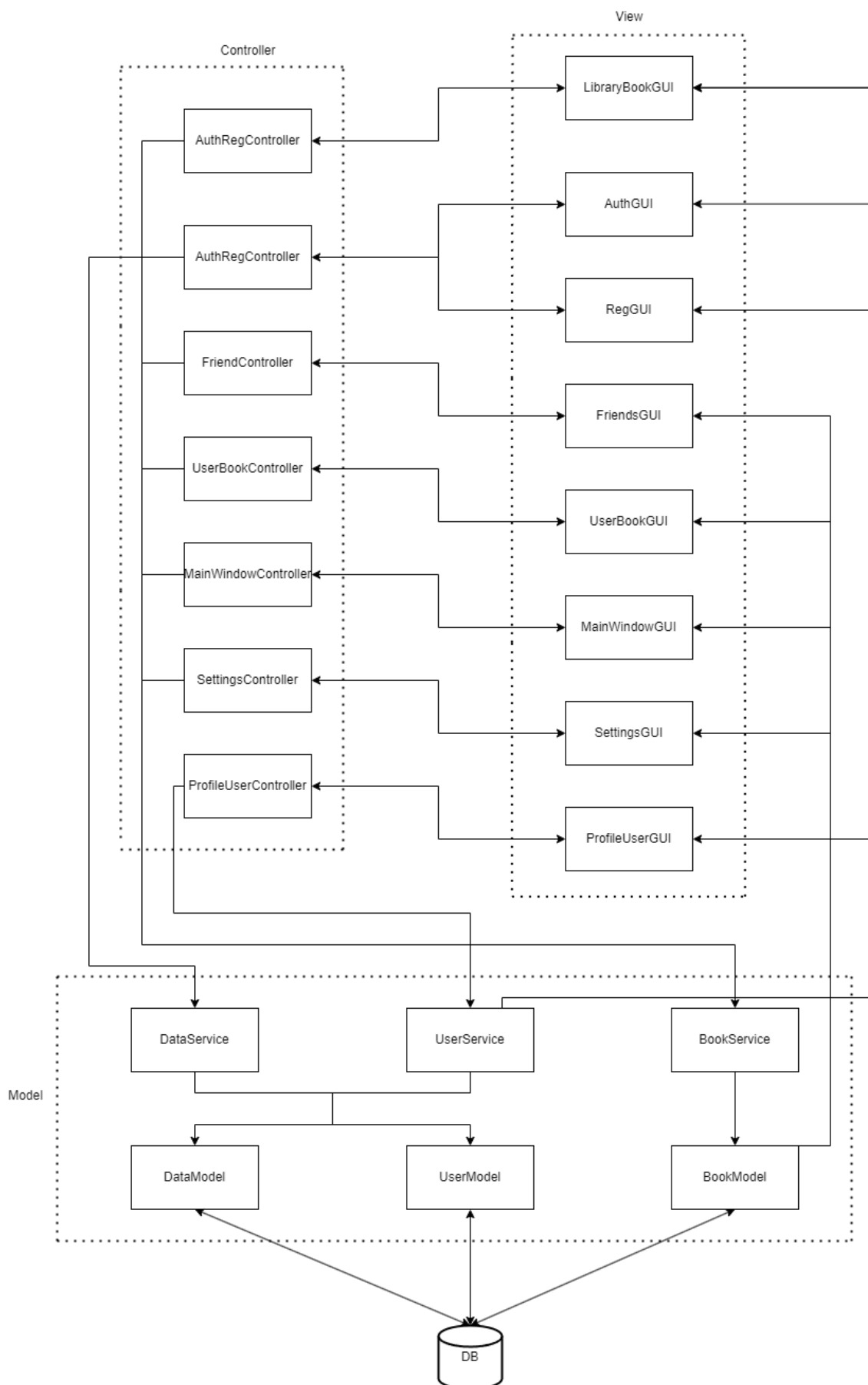


Рисунок 10 – Архитектура приложения

Основой реализации приложения будет: клиентская часть приложения разработана на WPF, и для взаимодействия с сервером используются сокеты; серверная часть приложения должна быть реализована на C# с использованием сокетов [5]; обмен данными между клиентом и сервером происходит с помощью Json файлов; взаимодействие с базой данных осуществляется через Npgsql, который предоставляет удобный доступ к базе данных PostgreSQL. Для управления базой данных также может использоваться pg4admin.

					ПГУ 1,090301,28.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

4 Реализация приложения

4.1 Описание работы приложения

Алгоритм работы приложения заключается в том, что пользователь взаимодействует с клиентской частью, написанной на WPF. В зависимости от того, какие действия выполняет пользователь, будет формироваться и отправляться запрос на сервер. Сервер в свою очередь будет обрабатывать запрос. Если в процессе обработки запроса происходит ошибка, сервер сообщает об этом клиенту. В случае успешной обработки запроса сервер взаимодействует с базой данных для выполнения операций, таких как создание, редактирование, получение или удаление данных. В случае успешного выполнения операции с базой данных, сервер возвращает клиенту необходимые данные или подтверждения выполнения операции. Если при обращении к базе данных возникает ошибка, сервер обрабатывает эту ошибку и возвращает информацию о ней клиенту.

Последовательность выполняемых действий авторизованным и не авторизованным пользователем представлена на рисунке 11.

					ПГУ 1,090301,28.001 ПЗ	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

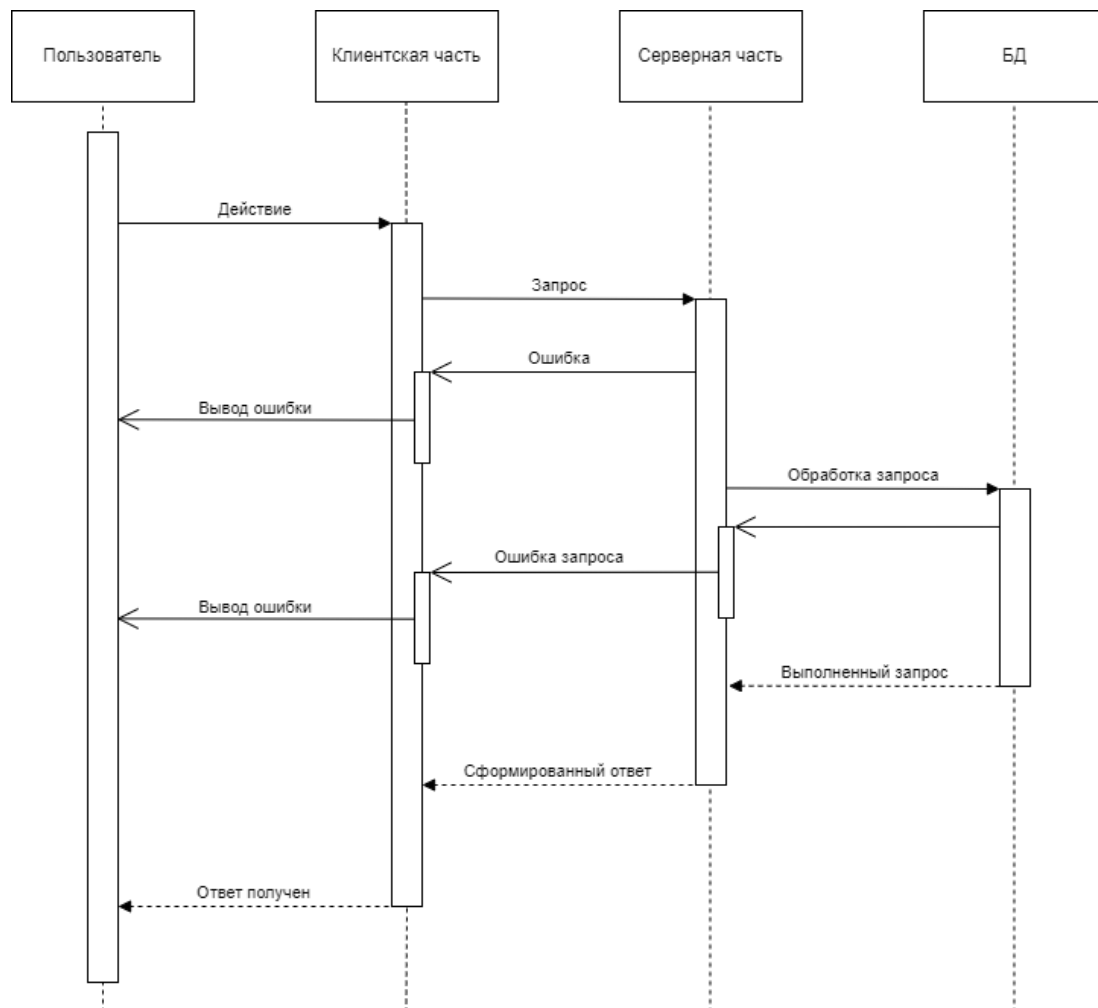


Рисунок 11 – UML диаграмма последовательности функционирования приложения

Топология системы и распределение ее компонентов по физическим узлам представлены на рисунке 12.

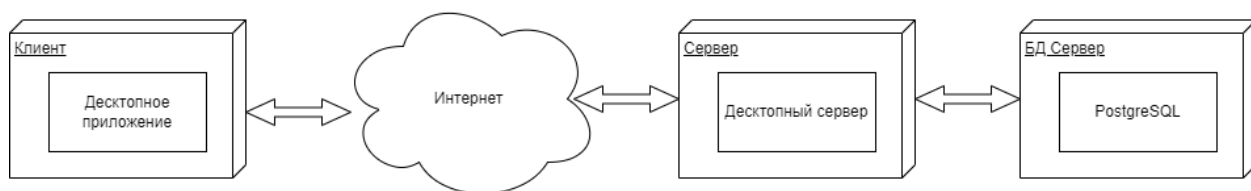


Рисунок 12 –UML диаграмма развертывания системы

4.2 Структура БД

Чтобы упростить хранение данных пользователя на серверной части приложения, необходимо использовать базу данных, которая будет хранить в себе

таблицы, сущности, поля и атрибуты [6]. Так как хорошо построенная база данных является одной из важнейших частей приложения, в данном разделе будут описана ее структура.

В данном приложении используется БД, которое будет содержать в себе следующие таблицы:

- таблица users, представленная в таблице 2. В ней будет содержаться основная информация о пользователях, которая будет использоваться для авторизации и идентификации пользователя в приложении;

- таблица books, представленная в таблице 3. Будет хранить основную информацию о книгах;

- таблица book_user, представленная в таблице 4. Данная таблица будет хранить личные книги пользователя;

- таблица favorite_book_user, представленная в таблице 5. Эта таблица будет хранить любимые книги пользователя;

- таблица statistics_user, представлена в таблице 6. В этой таблице будет содержаться личная статистика чтения книг пользователя;

- таблица quote_user, представлена в таблице 7. Здесь хранятся все цитаты пользователя;

- таблица friends_user, представлена в таблице 8. В ней хранится информация о друзьях пользователя.

Таблица 2 – Атрибуты сущности «users» (Пользователи)

Поле	Тип данных	Описание	Ключ
id_user	SERIAL	Идентификатор пользователя	РК
login	TEXT	Логин пользователя	
password	TEXT	Пароль пользователя	
name	TEXT	Имя пользователя	
last_name	TEXT	Фамилия пользователя	
icon	BYTEA	Иконка пользователя	

Таблица 3 – Атрибуты сущности «books» (Книги)

Поле	Тип данных	Описание	Ключ
id_book	SERIAL	Идентификатор книги	PK
author	TEXT	Автор книги	
book_name	TEXT	Название книги	
bytes	BYTEA	Содержимое книги	
description	TEXT	Описание книги	
genre	TEXT	Жанр книги	

Таблица 4 – Атрибуты сущности «book_user» (Книги пользователя) и «favorite_book_user» (Любимые книги пользователя)

Поле	Тип данных	Описание	Ключ
id	SERIAL	Идентификатор книги пользователя	PK
id_user	INTEGER	Идентификатор пользователя	FK
id_book	INTEGER	Идентификатор книги	FK

Таблица 5 – Атрибуты сущности «statistics_user» (Статистика пользователя)

Поле	Тип данных	Описание	Ключ
id	SERIAL	Идентификатор статистики пользователя	PK
reading_book	INTEGER	Прочитано книг	
reading_page	INTEGER	Прочитано страниц	
time_reading	DATE	Время чтения	
speed_reading	INTEGER	Скорость чтения	
id_user	INTEGER	Идентификатор пользователя	FK

Таблица 6 – Атрибуты сущности «quote_user» (Цитаты пользователя)

Поле	Тип данных	Описание	Ключ
id	SERIAL	Идентификатор цитат пользователя	PK
quote	TEXT	Цитата	
id_user	INTEGER	Идентификатор пользователя	FK

Таблица 7 – Атрибуты сущности «friends_user» (Друзья пользователя)

Поле	Тип данных	Описание	Ключ
id	SERIAL	Идентификатор	PK
status_friend	TEXT	Статус друга	
id_user	INTEGER	Идентификатор друга	FK

На рисунке 13 описана схема базы данных, разрабатываемого приложения, для наглядного рассмотрения взаимодействия таблиц.

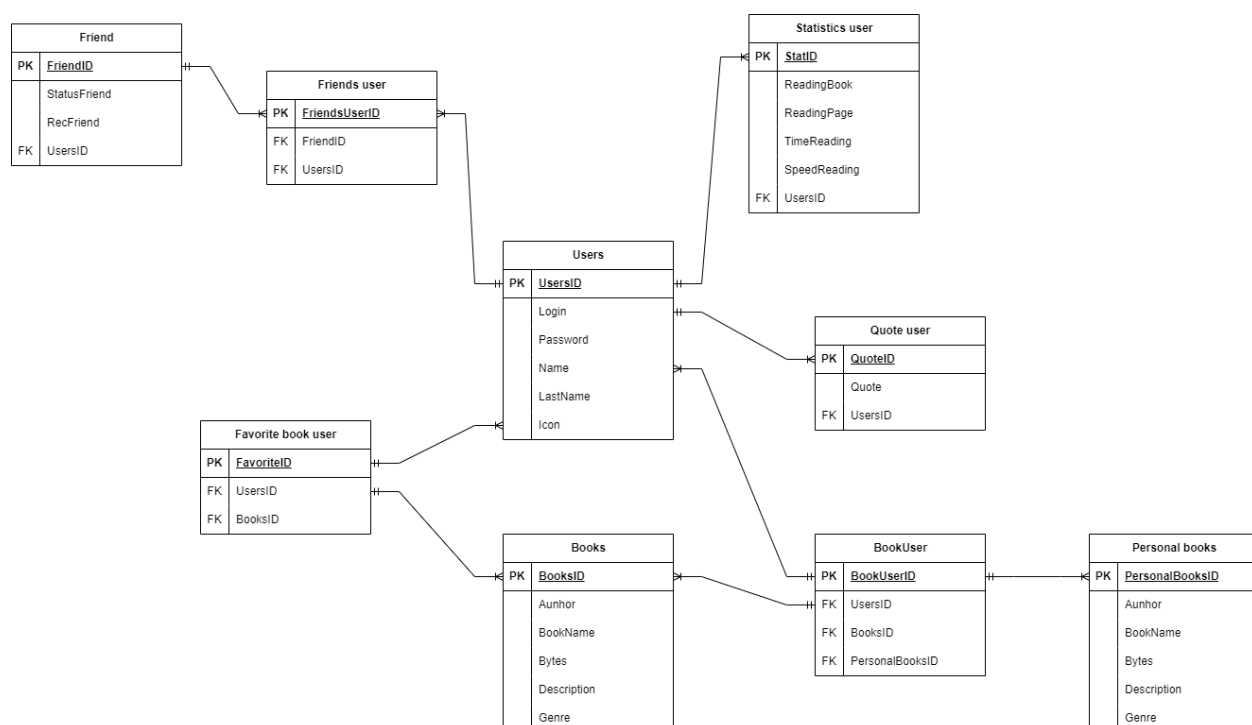


Рисунок 13 – Схема БД

4.3 Программная реализация приложения

Данное клиент-серверное приложение содержит три программных модуля:

- ClientGUI;
- MyHelper;
- Server.

ClientGUI – это программный модуль, который содержит реализацию компонентов представления и контроллера, а именно пользовательский интерфейс и классы для взаимодействия с сервером.

MyHelper – это модуль, представляющий собой dll библиотеку, которая реализует классы, которые используются и в программе сервера, и в программе клиента.

Server – программный модуль, содержащий набор классов для реализации компонента модели. Функционал этих классов заключается в обработке запросов пользователей, взаимодействия с базой данных и отправке ответа пользователю.

4.3.1 ClientGUI

Будет рассмотрен один из классов контроллера, а именно MainController. Этот класс является контроллером и отвечает за работу с представлением MainWindowGUI. Для этого реализованы следующие методы:

- ServerRequest();
- GetBook();
- OpenBookView();
- SetSourceImage().

Метод ServerRequest() отправляет на сервер запрос для получения информации о всех рекомендованных книг для пользователя. Реализация кода приведена в листинге 1.

Листинг 1. ServerRequest():

```
byte[] ServerRequest()  
{  
    Networking networking = new Networking();  
    byte[] request = new byte[1];  
    request[0] = (byte)UserRequests.GetRecommendation;  
    networking.SendData(request);  
}
```

					ПГУ 1,090301,28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		33

```

        var buffer = networking.RecvDataByte();
        networking.Close();
        return buffer;
    }

```

Метод GetBook() отправляет на сервер запрос для получения информации о выбранной книге пользователем, когда он взаимодействует с интерфейсом программы клиента. Реализация кода приведена в листинге 2.

Листинг 2. GetBook():

```

private Book GetBook(string nameBook)
{
    var buffer = Encoding.UTF8.GetBytes(nameBook);

    var resultBuffer = new byte[buffer.Length + 1];
    resultBuffer[0] = (byte)UserRequests.GetInfoBook;
    Array.Copy(buffer, 0, resultBuffer, 1, buffer.Length);
    Networking networking = new Networking();
    networking.SendData(resultBuffer);
    var requestBuffer = networking.RecvDataByte();
    networking.Close();
    Json.WriteJsonObject(requestBuffer);
    var book = (Book)Helper.ReadJsonObject<Book>();
    return book;
}

```

Метод OpenBookView() входной параметр класса Book для отображения выбранной книги пользователем. Реализация кода приведена в листинге 3.

Листинг 3. OpenBookView():

```

private void OpenBookView(Book book)
{
    UserBookGUI userBookGUI = new UserBookGUI();
    userBookGUI.SetData(Profile, book);
    userBookGUI.InitWindow();
    userBookGUI.Show();
}

```

Метод SetSourceImage() реализует функционал установки изображения в выбранный объект Image. Реализация кода приведена в листинге 4.

Листинг 4. SetSourceImage ():

```

void SetSourceImage(string pathImage, ref Image imageControl)
{
    BitmapImage bitmap = new BitmapImage();
    bitmap.BeginInit();
    bitmap.CacheOption = BitmapCacheOption.OnLoad;
}

```

					ПГУ 1,090301,28.001 ПЗ	Лист
						34
Изм.	Лист	№ докум.	Подпись	Дата		

```

        bitmap.StreamSource = new FileStream(pathImage, FileMode.Open,
        FileAccess.Read);
        bitmap.EndInit();
        imageControl.Source = bitmap;
        bitmap.StreamSource.Dispose();
    }

```

4.3.2 MyHelper

Данный компонент содержит множество классов, которые используются в приложениях клиента и сервера. В нем содержатся следующие классы:

- Book. Данный класс содержит в себе свойства, которые необходимы для отображения книги в интерфейсе пользователя, а именно название, автор, описание, жанр, иконка и содержимое книги;

- Json. Данный статический класс предоставляет методы для сериализации и десериализации данных, которое передаются между клиентом и пользователем по сети;

- Networking. Этот класс представляет собой интерфейс для взаимодействия по сети клиента с сервером;

- StatisticsBook. Класс содержит свойства, которые необходимы для отслеживания статистики за читаемой книгой пользователем. В нем содержатся такие свойства, как название книги, прочитанные страницы, потраченное время, количество прочитанных страниц, скорость чтения и статус прочтения книги.

- User. Этот класс содержит свойства, которые необходимы для работы с данными пользователя. А именно класс содержит такие свойства, как логин, пароль, имя, фамилия и данные иконки пользователя;

- UserRequests. Данный класс содержит перечисление запросов, которые отправляет пользователь.

Будут рассмотрены основные методы данных классов, а именно:

- WriteJsonFile();
- ReadJsonFile();
- RecvDataByte();
- SendData().

Метод WriteJsonFile() является шаблонным, и он сереализует передаваемый объект в json файл для удобной передачи данных по сети. Реализация кода приведена в листинге 5.

Листинг 5. WriteJsonFile():

```
public static void WriteJsonFile<T>(string path, T obj)
{
    var jsonFormatter = new DataContractJsonSerializer(typeof(T));
    using (var file = new FileStream(path, FileMode.Create))
    {
        jsonFormatter.WriteObject(file, obj);
    }
}
```

Метод ReadJsonFile() является шаблонным, и он десереализует json файл в объект, тип которого был указан при вызове. Реализация кода приведена в листинге 6.

Листинг 6. ReadJsonFile():

```
public static object ReadJsonFile<T>(string path)
{
    var jsonFormatter = new DataContractJsonSerializer(typeof(T));
    object jsonObj;
    using (var file = new FileStream(path, FileMode.Open))
    {
        jsonObj = (T)jsonFormatter.ReadObject(file);
    }
    return jsonObj;
}
```

Метод RecvDataByte() реализует получение данных с сервера и возврат этих данных в виде массива байт . Реализация кода приведена в листинге 7.

Листинг 7. RecvDataByte():

```
public byte[] RecvDataByte()
{
    var buffer = new byte[65536 * 2048];
    int size = 0;
    int streamSize = 0;
    do
    {
        size = tcpSocket.Receive(buffer, streamSize, buffer.Length -
streamSize, SocketFlags.None);
        streamSize += size;
    }
}
```

					ПГУ 1,090301,28.001 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

```

    } while (tcpSocket.Available > 0);
    byte[] newBuffer = new byte[streamSize];
    Array.Copy(buffer, newBuffer, streamSize);

    return newBuffer;
}

```

Метод `SendData()` реализует отправку данных, в качестве передаваемого параметра принимается строка, эта строка преобразуется в массив байт и передается серверу. Реализация кода приведена в листинге 8.

Листинг 8. `SendData()`:

```

public void SendData(string message)
{
    var data = Encoding.UTF8.GetBytes(message);
    tcpSocket.Send(data);
}

```

4.3.3 Server

Данный компонент содержит набор классов для реализации программы сервера, а также класс для взаимодействия с базой данных.

Будут рассмотрены основные методы данных классов, а именно:

- `StartServer()`;
- `GetAllBook()`;
- `LoadRegisteredUsers()`;
- `AddBook()`;
- `DeleteBook()`.

Метод `StartServer()` реализует прослушивание подключаемых клиентов. Когда клиент подключается, сервер создает новый объект класса для обработки запроса пользователя, и запускает данную обработку в отдельном потоке. Реализация кода приведена в листинге 9.

Листинг 9. `StartServer()`:

```

public void StartServer()
{
    while (true)
    {
        TcpClient client = listener.AcceptTcpClient();
        ClientObject clientObject = new ClientObject(client);
        Thread clientThread = new Thread(new
        ThreadStart(clientObject.Process));
    }
}

```

					ПГУ 1,090301,28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		37

```

        clientThread.Start();
    }
}

```

Метод GetAllBook() реализует обращение к базе данных PostgreSQL с целью получения всей таблицы книг. Возвращает метод таблицу этих книг. Реализация кода приведена в листинге 10.

Листинг 10. GetAllBook():

```

public DataTable GetAllBook()
{
    npgsqlCommand.CommandText = "SELECT * FROM books";
    NpgsqlDataReader dataReader = npgsqlCommand.ExecuteReader();
    DataTable dataTable = new DataTable();
    if (dataReader.HasRows)
    {
        dataTable.Load(dataReader);
    }
    return dataTable;
}

```

Метод LoadRegisteredUsers() реализует обращение к базе данных PostgreSQL с целью добавления нового пользователя. Для этого в метод нужно передать пароль и логин нового пользователя. Реализация кода приведена в листинге 11.

Листинг 11. LoadRegisteredUsers():

```

public void LoadRegisteredUsers(string login, string password)
{
    npgsqlCommand.CommandText = String.Format("INSERT INTO
users(login,password) VALUES ('{0}','{1}']", login, password);
    NpgsqlDataReader dataReader = npgsqlCommand.ExecuteReader();
    if (dataReader.HasRows)
    {
        DataTable dataTable = new DataTable();
        dataTable.Load(dataReader);
    }
}

```

Метод AddBook() реализует добавление книги пользователя в личную библиотеку. Для этого программа клиента передает книгу, которую необходимо добавить, данный метод получает эту книгу и загружает в базу данных. Реализация кода приведена в листинге 11.

Листинг 11. AddBook():

```

private void AddBook()

```

					ПГУ 1,090301,28.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    var userInfo = Encoding.UTF8.GetString(data);
    string bufferJson;
    string nameUser;

    using (TextReader reader = new StringReader(userInfo))
    {
        nameUser = reader.ReadLine();
        bufferJson = reader.ReadLine();
    }
    var byteBufferJson = Encoding.UTF8.GetBytes(bufferJson);
    Helper.WriteJsonObject(byteBufferJson);
    Book book = (Book)Helper.ReadJsonObject<Book>();

    var postgresql = new PostgreSQL();
    postgresql.LoadNewBookUser(book, nameUser);
    postgresql.Close();
}

```

Метод DeleteBook() реализует удаление книги пользователя из личной библиотеки. Для этого программа клиента передает идентификатор книги, которую необходимо удалить, данный метод данный идентификатор и передает его в метод удаления книги из базы данных пользователя. Реализация кода приведена в листинге 12.

Листинг 12. DeleteBook():

```

private void DeleteBook()
{
    var userInfo = Encoding.UTF8.GetString(data);
    string nameBook;
    string nameUser;
    using (TextReader reader = new StringReader(userInfo))
    {
        nameUser = reader.ReadLine();
        nameBook = reader.ReadLine();
    }
    var postgresql = new PostgreSQL();
    postgresql.DeleteUserBook(nameUser, nameBook);
    postgresql.Close();
}

```

4.4 Основные алгоритмы взаимодействия клиента и сервера

UML диаграммы деятельности разработаны для наглядного отображения последовательности действий между клиентской и серверной частями в процессе взаимодействия.

					ПГУ 1,090301,28.001 ПЗ	Лист
						39
Изм.	Лист	№ докум.	Подпись	Дата		

Для начала будет рассмотрен алгоритм регистрация и авторизация. Для того, чтобы пользователь имел возможности использовать все возможности приложения, предлагаемых приложением, необходимо выполнить регистрацию и дальнейший вход в систему. Если пользователь не имеет учетной записи, он может зарегистрироваться, перейдя на соответствующую страницу и заполнив необходимую информацию. Введенные данные отправляются на сервер для обработки и проверки уникальности. Если данные уже существуют в базе данных, пользователю будет отправлено уведомление об этом. В ином случае данные будут сохранены, и пользователь будет перенаправлен на главную страницу приложения.

При попытке авторизации с неправильными учетными данными пользователю будет выведено сообщение о неверном вводе. В случае успешной авторизации пользователь получит доступ ко всем функциям приложения.

Последовательность описанных действий представлены на рисунке 14.

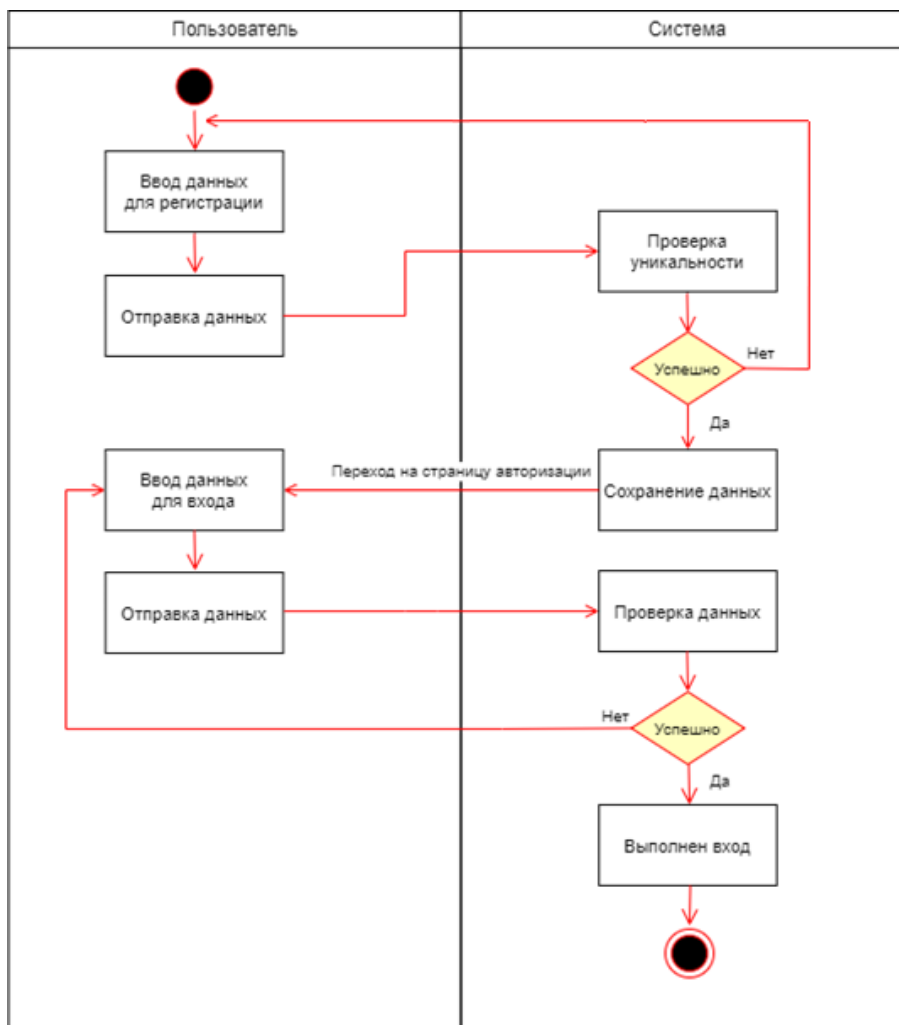


Рисунок 14 – UML диаграмма деятельности регистрации и авторизации

Далее рассмотрен алгоритм чтения книги. Чтобы пользователю начать читать книгу, ему необходимо выбрать нужную книгу и нажать кнопку «читать». Далее передается управление контроллеру, который в свою очередь обращается к серверу для получения данных книги. На стороне сервера проверяется, авторизован пользователь или нет. Если пользователь авторизован, то данные книги берутся из его личной библиотеки книг, если нет, то из общей. Далее идет проверка целостности данных книги, если целостность нарушена, то сервер передает об этом сообщение контроллеру, который в свою очередь выводит предупреждение клиенту о том, что книгу не удалось открыть. Иначе сервер отправляет данные книги.

После удачной проверки книги и принятия данных книги от сервера, контроллер формирует книгу для удобного прочтения. Далее выводит содержимое пользователю.

Последовательность описанных действий представлены на рисунке 15.

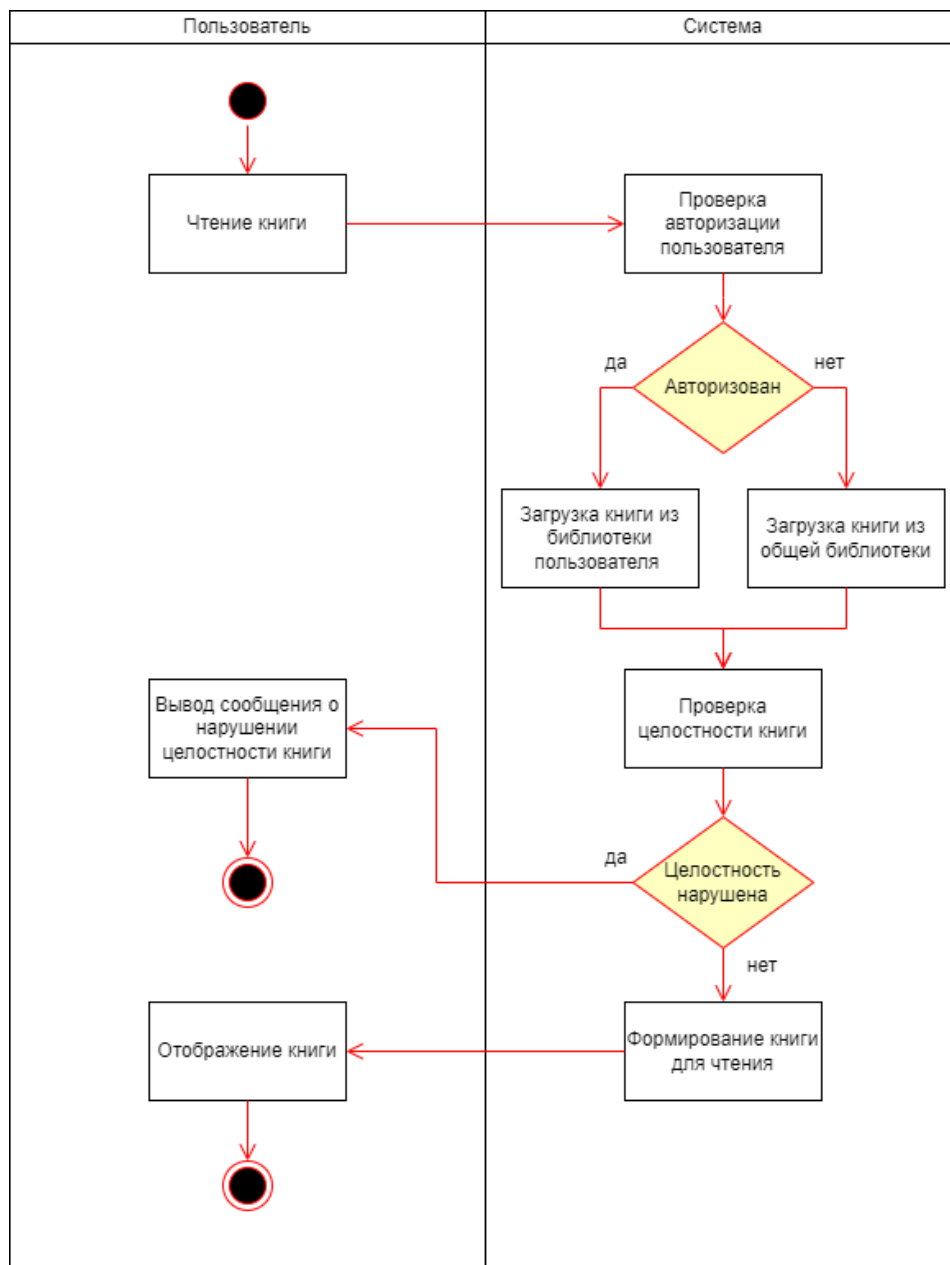


Рисунок 15 – UML диаграмма деятельности чтения книги

На рисунке 16 представлена UML диаграмма деятельности поиска книги. Данный алгоритм работает следующий образом, пользователь вводит текст для поиска нужной книги и нажимает кнопку поиска. Далее управление передается контроллеру, который отправляет запрос серверу о том, что необходимо найти

книгу в библиотеке. Сервер в свою очередь ищет в базе данных нужную книгу, поиск происходит по названию, автору и жанру книги. Если она отсутствует в библиотеке, то сервер сообщает об этом контроллеру, который в свою очередь сообщает об этом клиенту. Если книга находится в библиотеке, то сервер отправляет информацию о ней контроллеру, после чего последний выводит ее клиенту.

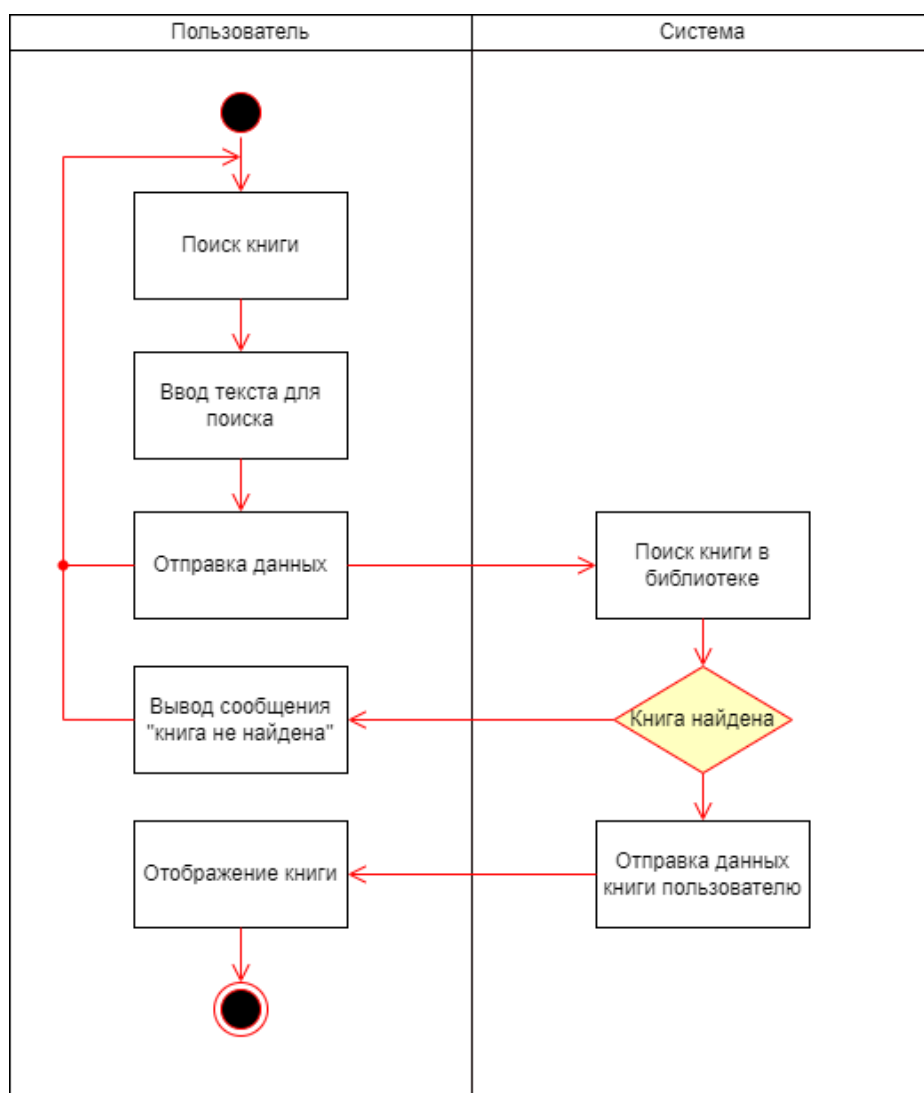


Рисунок 16 – UML диаграмма деятельности поиска книги

Далее рассмотрена возможность просмотра списка добавленных книг пользователем, а также непосредственно добавление книг в личную библиотеку. Для просмотра списка добавленных книг нужно перейти на страницу "Мои книги". При этом система проверит авторизацию пользователя в приложении. Если

пользователь не авторизован, его перенаправят на страницу входа. После успешной авторизации он сможет перейти на страницу своей личной библиотеки.

На странице "Мои книги" проверяется, есть ли у пользователя книги, загруженные в базу данных. Если у пользователя нет личных книг, то он может добавить их из своего устройства или из приложения.

Как было сказано, у пользователя есть два варианта добавления книг, а именно из библиотеки приложения и загрузка своей собственной. Чтобы добавить книгу из имеющийся библиотеки приложения, необходимо ее найти, алгоритм поиска был описан выше и представлен на рисунке 16. После этого пользователь должен нажать на кнопку «Добавить книгу в библиотеку», чтобы ее добавить. Второй вариант – это добавление с устройства, для этого нужно перейти в настройки профиля и выбрать книгу для загрузки с устройства. После этого пойдет загрузка на сервер.

Если пользователь авторизован и у него есть книги в библиотеке, контроллер получит список книг пользователя и выведет их в интерфейс.

Последовательность действий по просмотру личной библиотеки и добавлению книг представлены на рисунке 17.

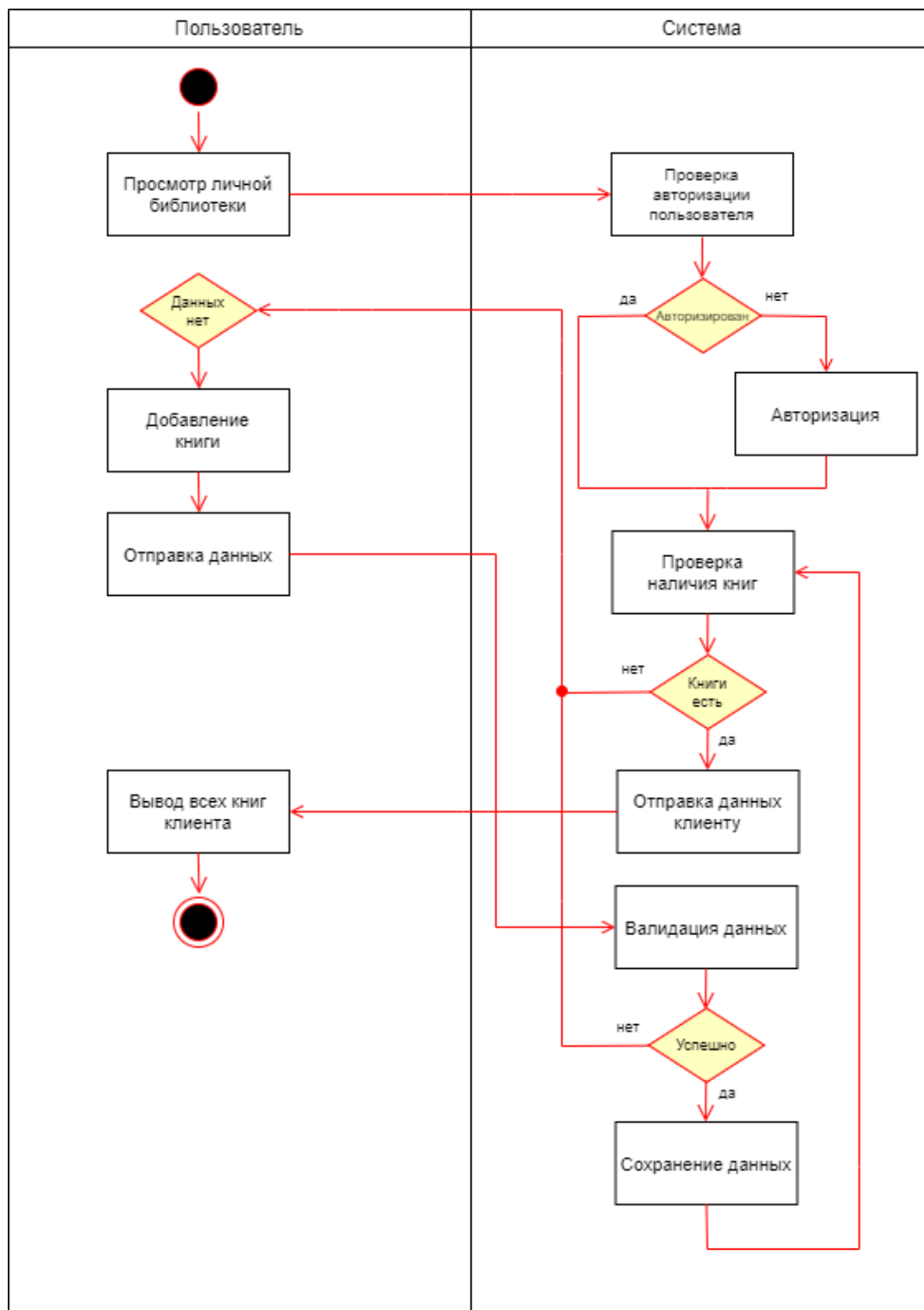


Рисунок 17 – UML диаграмма просмотра личной библиотеки и добавления
КНИГИ

5 Тестирование приложения

Для проверки работоспособности клиент-серверного приложения «Электронная библиотека» необходимо провести его тестирование.

Тестирование будет состоять из следующих действий:

- регистрация пользователя;
- авторизация пользователя;
- просмотр, скачивание и чтение книги;
- редактирование аккаунта пользователя;
- поиск по критериям;
- добавление книги в личную библиотеку.

5.1 Регистрация пользователя

Чтобы протестировать регистрацию пользователя, нужно перейти на страницу с формой регистрации и ввести следующие поля: логин, пароль, подтверждение пароля. После того, как пользователь заполнит необходимые данные, в базу данных внесется новая запись о пользователе.

На рисунке 18 представлен ввод данных при регистрации.

Рисунок 18 – Регистрация пользователя

На рисунках 19 показана таблица до занесения нового пользователя, а на рисунке 20 после регистрации пользователя.

Data Output Сообщения Notifications						
	id [PK] integer	логин text	пароль text	имя text	фамилия text	фотография text
1	0	admin	admin	Олег	Духнов	.\Users\admin\admin.jpg

Рисунок 19 – Таблица пользователей до регистрации

	id [PK] integer	логин text	пароль text	имя text	фамилия text	фотография text
1	0	admin	admin	Олег	Духнов	.\Users\admin\admin.jpg
2	13	NewUser	NewUser	[null]	[null]	[null]

Рисунок 20 – Таблица пользователей после регистрации

Если попытаться зарегистрировать нового пользователя с уже существующим логином, то программа выведет сообщение о том, что такой пользователь уже существует, на рисунке 21 это показано.

Рисунок 21 – Регистрация пользователя с уже существующим именем

5.2 Авторизация пользователя

Для тестирования авторизации необходимо перейти на соответствующее окно и ввести логин и пароль. На рисунке 22 представлен ввод данных.

Рисунок 22 – Авторизация пользователя

После успешной авторизации пользователь попадет на главную страницу приложения. На рисунке 23 это отражено.

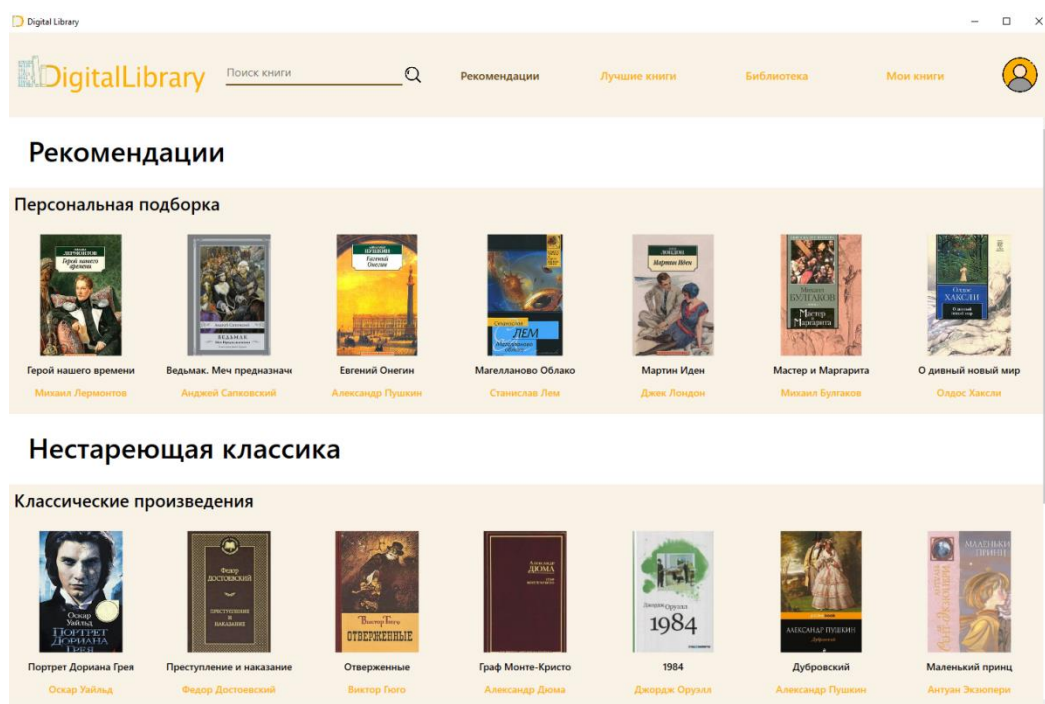


Рисунок 23 – Главная страница приложения после успешной авторизации

Если ввести неверные данные, то программа выведет предупреждение, это показано на рисунке 24.

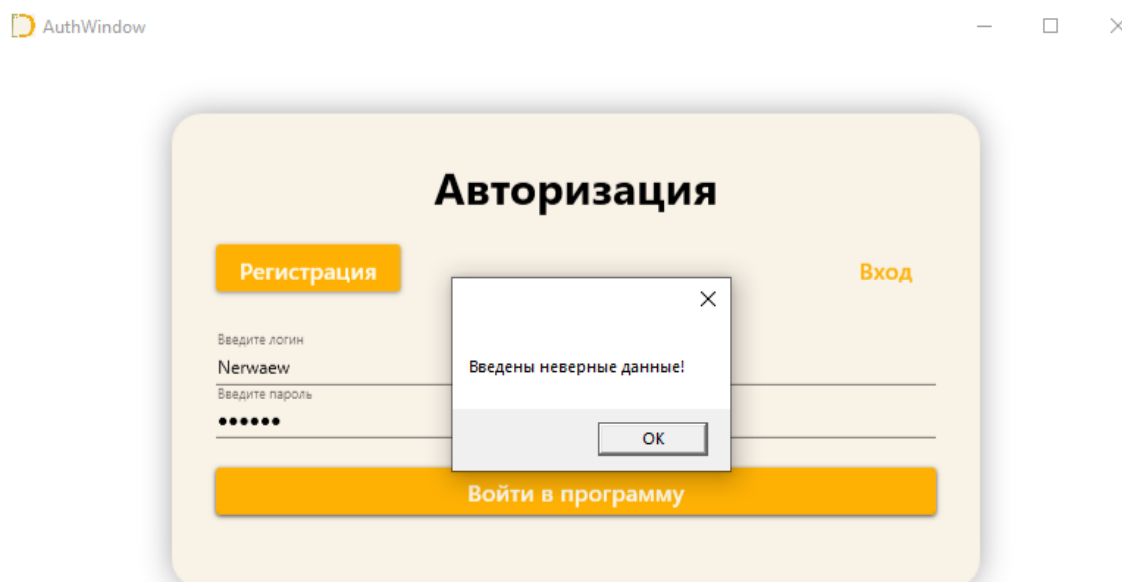


Рисунок 24 – Вывод предупреждения при неверном вводе данных

5.3 Просмотр, скачивание и чтение книги

Для того, чтобы протестировать функционал просмотра книги, необходимо выбрать нужную книгу нажав на нее левой кнопкой мыши. Книги выбирать можно как из всей библиотеки, так и из личной. Пример книг продемонстрирован на рисунке 23.

После выбора книги, выведется пользовательский интерфейс просмотра, который представлен на рисунке 25.

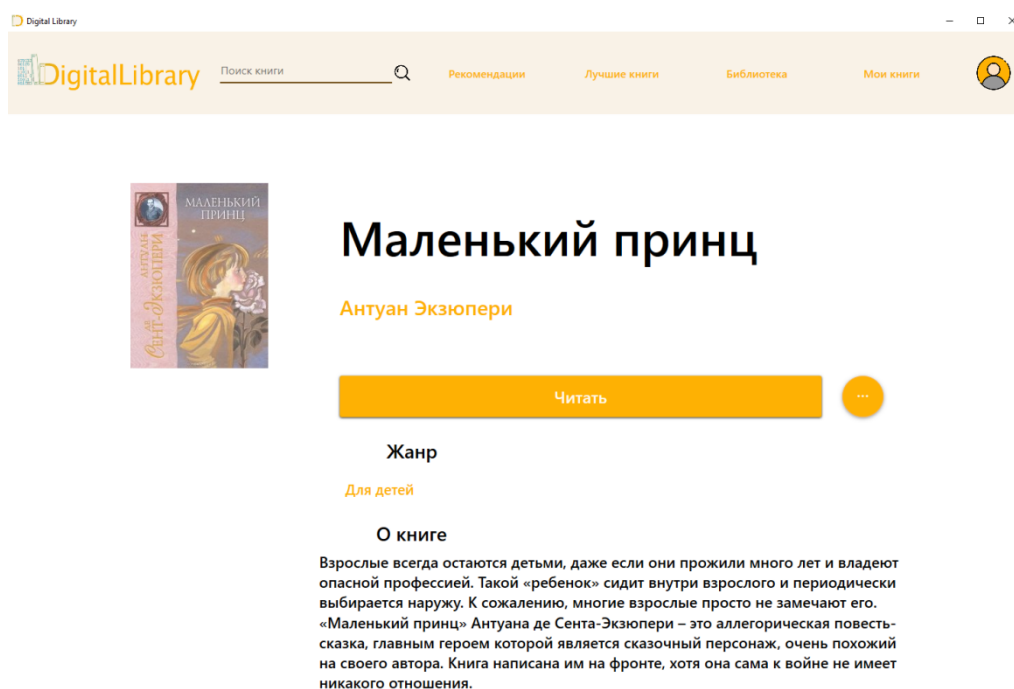


Рисунок 25 – Просмотр книги

Для скачивания книги на устройства необходимо нажать на кнопку с тремя точками и выбрать «Скачать книгу на устройство». После этого программа предложит сохранить книгу на устройстве, пользователю необходимо выбрать путь к книге и ее название, после чего она сохранится.

Процесс выбора пункта меню «Скачать на устройство» представлен на рисунке 26. На рисунках 27 и 28 представлены выбор пути для сохранения и результат этого сохранения.

					ПГУ 1,090301,28.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		



Маленький принц

Антуан Экзюпери

Читать

Жанр

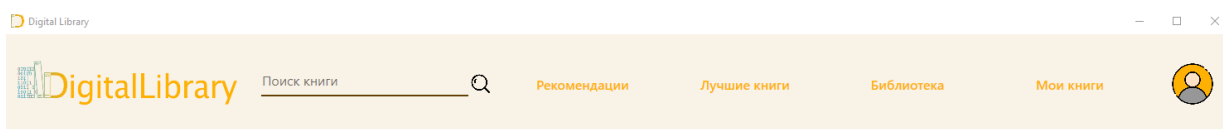
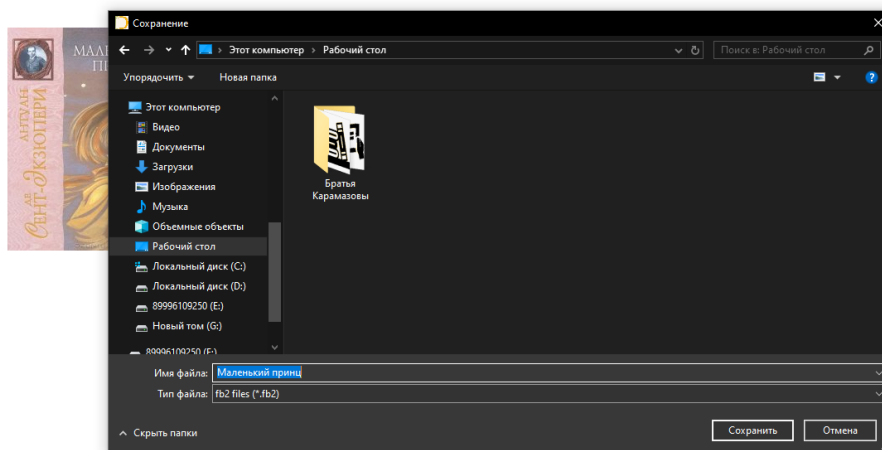
Для детей

О книге

Взрослые всегда остаются детьми, даже если они прожили много лет и владеют опасной профессией. Такой «ребенок» сидит внутри взрослого и периодически выбирается наружу. К сожалению, многие взрослые просто не замечают его. «Маленький принц» Антуана де Сента-Экзюпери – это аллегорическая повесть-сказка, главным героем которой является сказочный персонаж, очень похожий на своего автора. Книга написана им на фронте, хотя она сама к войне не имеет никакого отношения.

Добавить книгу в библиотеку
Скачать книгу на устройство

Рисунок 26 - Выбор пункта «Скачать на устройство»



Взрослые всегда остаются детьми, даже если они прожили много лет и владеют опасной профессией. Такой «ребенок» сидит внутри взрослого и периодически выбирается наружу. К сожалению, многие взрослые просто не замечают его. «Маленький принц» Антуана де Сента-Экзюпери – это аллегорическая повесть-сказка, главным героем которой является сказочный персонаж, очень похожий на своего автора. Книга написана им на фронте, хотя она сама к войне не имеет никакого отношения.

Рисунок 27 – Выбор пути и названия для сохранения

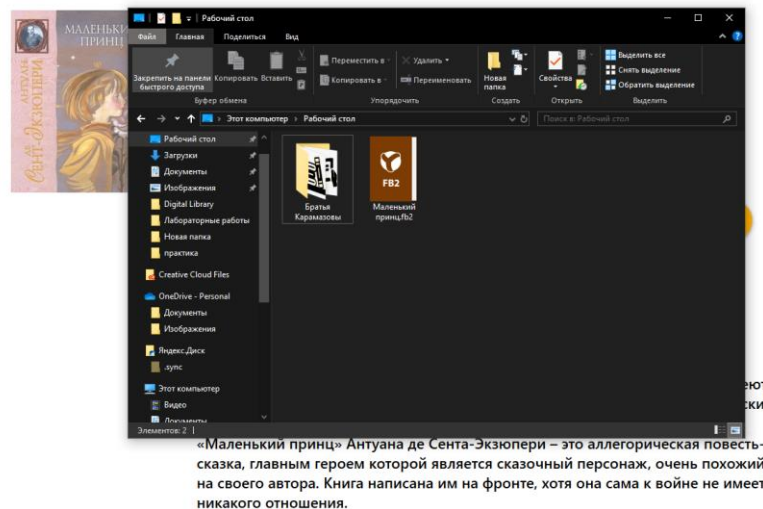


Рисунок 28 – Результат сохранения книги

Для того, чтобы начать читать книгу, необходимо выбрать соответствующую кнопку, которая представлена на рисунке 25. После нажатия на кнопку откроется пользовательский интерфейс для чтения книги. Результат представлен на рисунке 29.

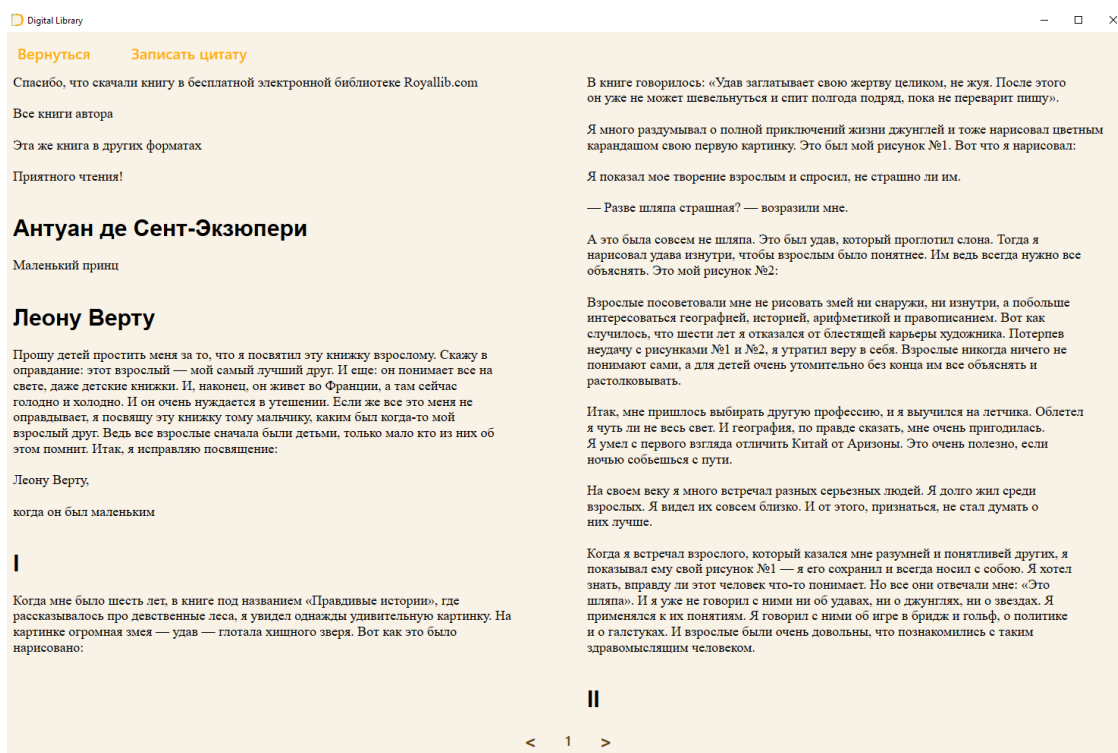


Рисунок 29 – чтение книги

5.4 Редактирование аккаунта пользователя

Чтобы протестировать настройки пользователя, необходимо выбрать пункт «Настройки», который показан на рисунке 30.

Результат перехода на страницу редактирования пользователя представлен на рисунке 31.

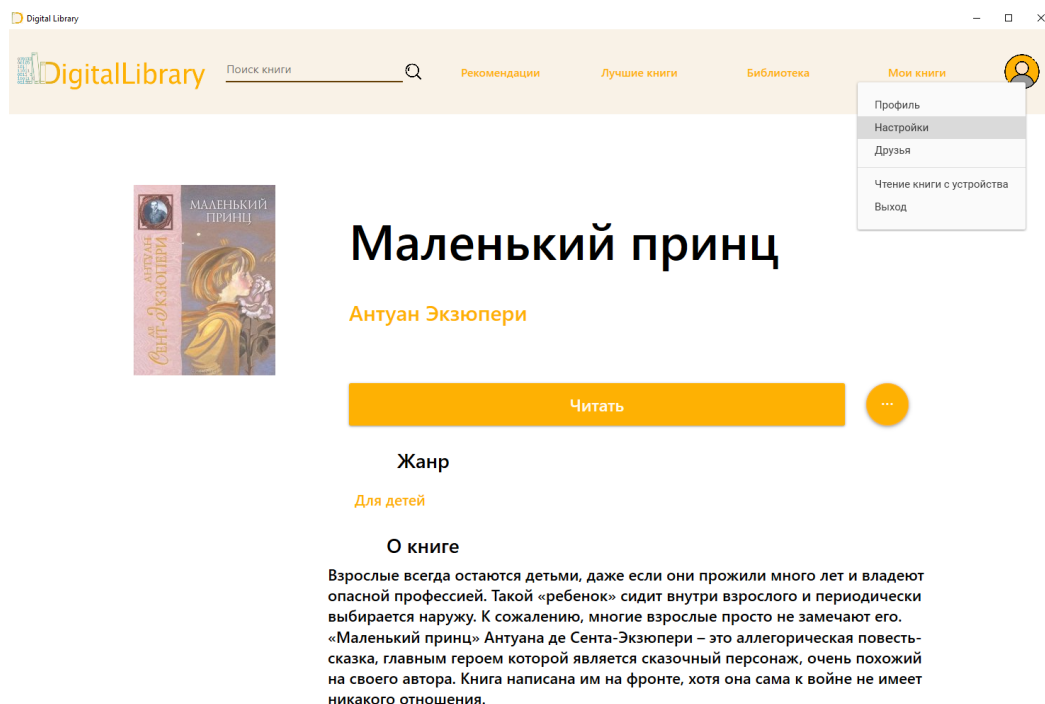


Рисунок 30 – Выбор меню «Настройки»

В редактирование профиля пользователя доступен следующий функционал, которые необходимо протестировать:

- изменение пароля;
- изменение персональных данных;
- смена фотографии профиля;
- запись цитат;
- добавление книги в личную библиотеку.

Для изменения пароля необходимо ввести новый пароль и повторить его, данный процесс представлен на рисунке 31.

До изменения пароль представлен на рисунке 21, после изменения на рисунке 32.

Для смены личной фотографии профиля необходимо выбрать с устройства картинку в формате jpg. После загрузки картинки изменения будут видны в профиле, результат этого представлен на рисунке 34. Также изменения будут и на сервере, а именно сохранится в базе данных новая картинка пользователя, результат отображен на рисунке 35.

Рисунок 34 – Результат изменения фотографии профиля в приложении

Data Output							
Сообщения							
Notifications							
	id	логин	пароль	имя	фамилия	фотография	
	[PK] integer	text	text	text	text	text	
1	0	admin	admin	Олег	Духнов	.\Users\admin\admin.jpg	
2	13	NewUser	NewPassword	Антуан	Экзюпери	.\Users\NewUser\NewUser.jpg	

Рисунок 35 – Результат изменения фотографии профиля в БД

При тестировании записи цитаты необходимо в соответствующее поле записать понравившуюся цитату, и нажать кнопку сохранить. После чего цитата появится в профиле. Результат представлен на рисунке 36.

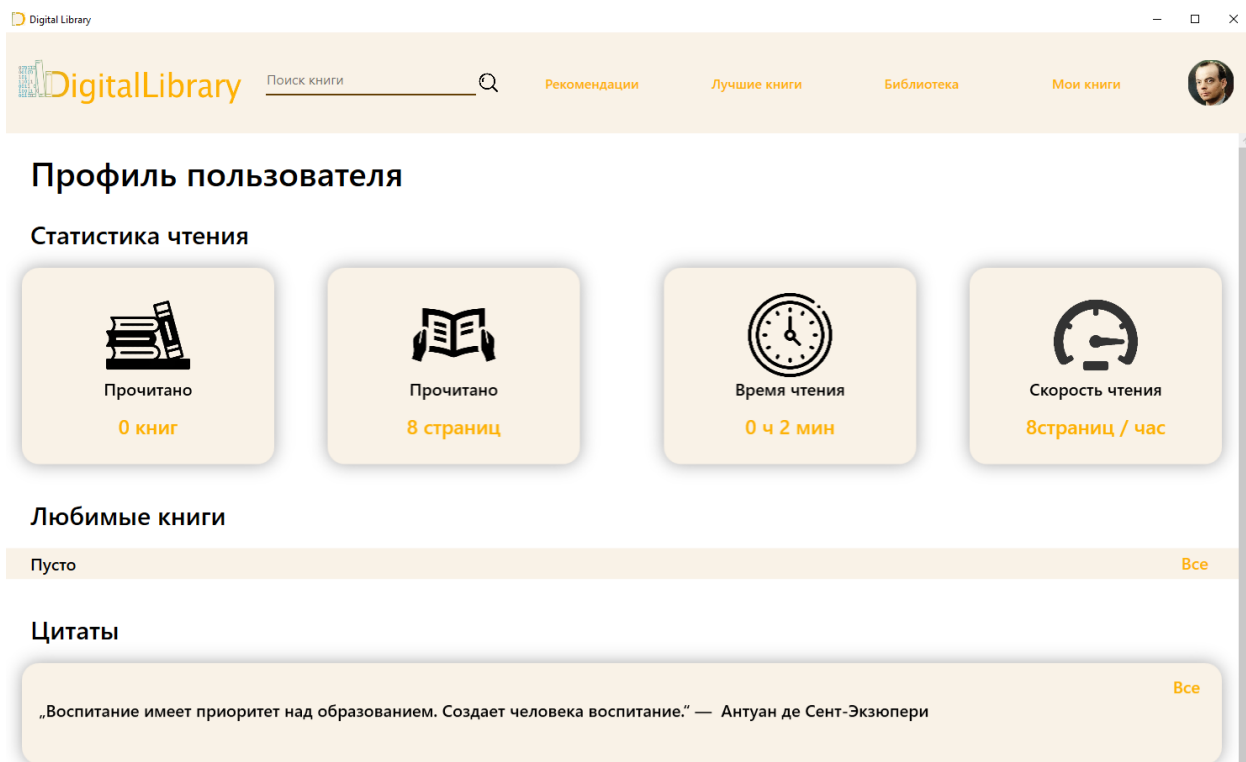


Рисунок 36 – Новая записанная цитата

Добавление книги в личную библиотеку будет протестировано дальше.

5.5 Поиск по критериям

Для тестирования поиска по критериям, необходимо в строку «Поиск книги» написать требуемый запрос. Для примера будет введен «пушкин», представлено это на рисунке 37. После этого необходимо нажать на значок поиска, которая находится рядом. В результате будет выведен список книг, который написал А.С. Пушкин. Это показано на рисунке 38.

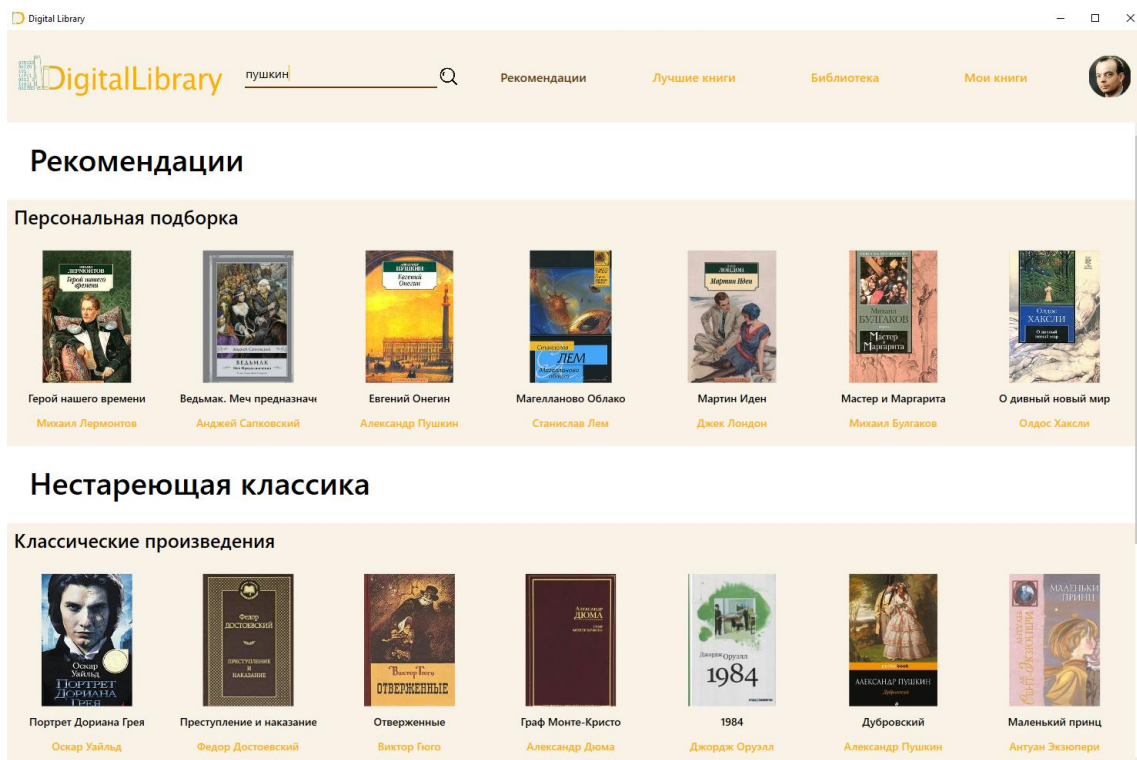


Рисунок 37 – Поиск книг

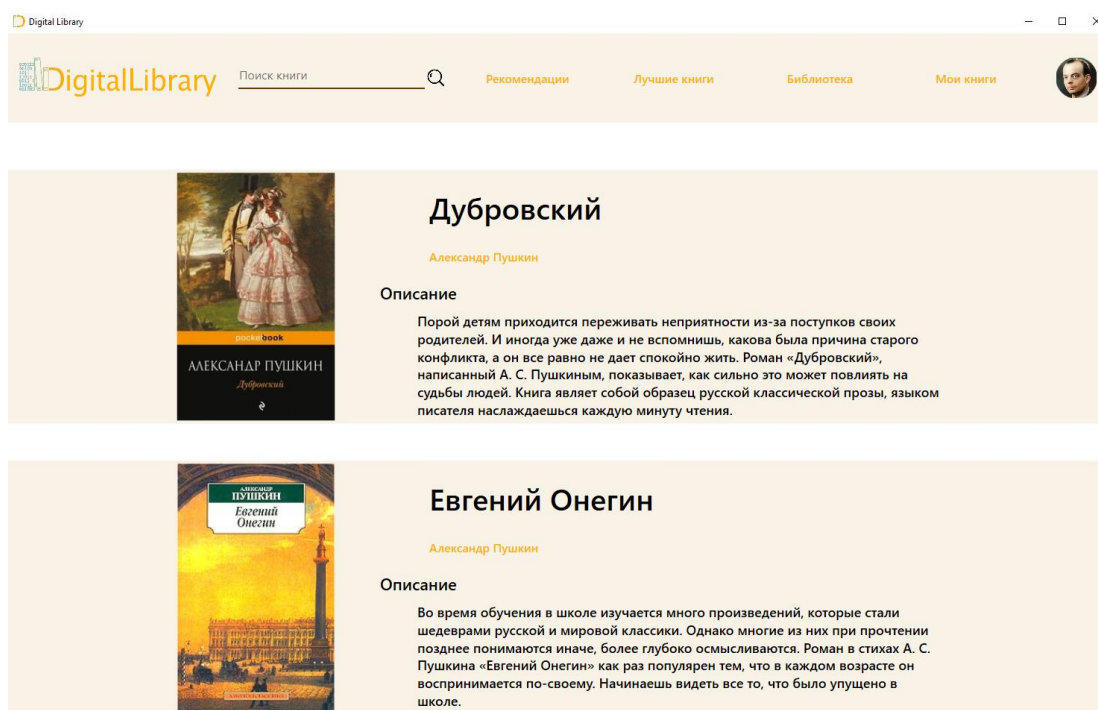


Рисунок 38 – Результат поиска

5.6 Добавление книги в личную библиотеку

Пользователь может добавить книги в личную библиотеку двумя способами, а именно из библиотеки приложения и с устройства.

					ПГУ 1,090301,28.001 ПЗ	Лист 57
Изм.	Лист	№ докум.	Подпись	Дата		

Изначально, если зайти на вкладку «Мои книги», то будет видно, что библиотека пользователя пуста, так как книги еще не добавлены. Это показано на рисунке 39.

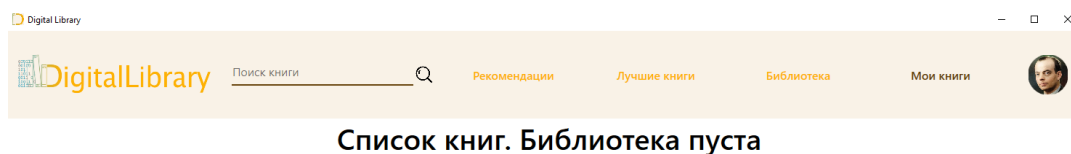


Рисунок 39 – Библиотека пользователя до добавления книг

Чтобы добавить книгу из библиотеки приложения, необходимо перейти в просмотр книги и нажать на пункт меню «Добавить книгу в библиотеку». Это представлено на рисунке 40.

После добавления данная книга появится в списке «Мои книги». Результат этого представлен на рисунке 42.

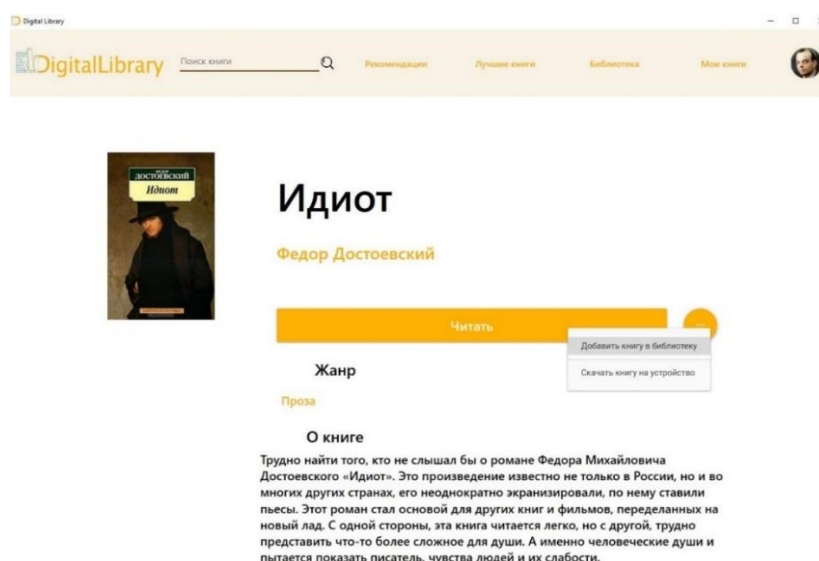


Рисунок 40 – Добавление книги в библиотеку из приложения

Для добавления книги со своего устройства, необходимо перейти в настройки профиля и заполнить поля для выбранной операции, показано это на рисунке 41.

Результат добавления книги с устройства в личную библиотеку представлен на рисунке 42.

Digital Library
Поиск книги
Рекомендации
Лучшие книги
Библиотека
Мои книги

Редактирование профиля

Сменить фотографию

Требуется загрузить картинку в формате jpg

Изменение пароля

Введите пароль

Повторите пароль

Сохранить

Добавление книги в библиотеку

Введите путь к файлу книги *

C:\Users\Olen\Desktop\Братья Карамазовы\Братья Карамазов

Выбрать путь к иконке

Введите путь к файлу иконки *

C:\Users\Olen\Desktop\Братья Карамазовы\avideaders.png

Выбрать путь к файлу

Введите название книги *

Братья Карамазовы

Введите автора книги

Федор Достоевский

Введите описание книги

Роман Федора Достоевского «Братья Карамазовы» можно без

Введите жанр книги

Проза

Сохранить

* обязательное поле

Запись цитаты

Введите цитату

Сохранить

Изменение персональных данных

Имя

Антун

Фамилия

Экзопери

Сохранить

Рисунок 41 – Добавление книги с устройства

Digital Library
Поиск книги
Рекомендации
Лучшие книги
Библиотека
Мои книги

Список книг

Идиот

Федор Достоевский

Описание

Трудно найти того, кто не слышал бы о романе Федора Михайловича Достоевского «Идиот». Это произведение известно не только в России, но и во многих других странах, его неоднократно экранизировали, по нему ставили пьесы. Этот роман стал основой для других книг и фильмов, переделанных на новый лад. С одной стороны, эта книга читается легко, но с другой, трудно представить что-то более сложное для души. А именно человеческие души и

Братья Карамазовы

Федор Достоевский

Описание

Роман Федора Достоевского «Братья Карамазовы» можно без сомнений назвать шедевром, невероятно глубоким и неоднозначным. В нем столько мыслей, подтекстов, намеков. Его можно читать несколько раз, каждый раз види что-то новое, получая новые эмоции, осознавая нечто иное. Не зря он десятки раз экранизировался в самых разных версиях – от близкой к тексту до более вольной интерпретации. И несмотря ни на что он сохраняет духовную силу и

Рисунок 42 – Результат добавления книг

Процесс тестирования прошел успешно, все функции приложения работают правильно.

6 Безопасность жизнедеятельности

При взаимодействии с компьютером, необходимо соблюдать меры безопасности, чтобы не было неприятных последствий для здоровья после использования. Если пользователь ПК будет пренебрегать техникой безопасности, то проблемы со здоровьем будут неизбежны.

6.1 Вред здоровью при работе с ПК

Проблемы со здоровьем в процессе взаимодействия с компьютером имеют свойство накапливаться, то есть здоровье ухудшается не сразу, а постепенно. Наиболее опасными факторами являются:

- нагрузка на суставы. При неправильной посадке за компьютером, на суставы будет идти большая нагрузка, а именно кистей рук, из-за чего может произойти даже онемение рук;
- нагрузка на спину. Опять же, если принять неправильно положение осанки при работе за компьютером, может появиться искривление позвоночника и даже грыжа межпозвоночного диска;
- нагрузка на органы зрения. При долгой работе с мониторами будет ухудшаться здоровья глаз, впоследствии могут появиться синдромы «красного глаза» и «сухого глаза»;
- нагрузка на психику и нервную. При долгом взаимодействии с ПК, у пользователя могут возникать проблемы с памятью, а также может появиться депрессия и даже бессонница.

6.2 Снижение вреда для здоровья при работе с ПК

При соблюдении простых правил можно избежать серьезных проблем со здоровьем, для этого нужно:

- соблюдать необходимое расстояние до монитора. Правильным расстоянием до дисплея является расстояние, при котором пользователь может коснуться указательным пальцем до верхнего края монитора;

- соблюдать оптимальный угол экрана. Оптимальным углом является наклон монитора в 30 градусов;
- клавиатуру ставить в нужное положение. Клавиатура должна находиться на расстоянии 20-30 сантиметров от края стола;
- делать перерыв. Примерно каждый час необходимо делать разминку для органов зрения, а именно оторваться от монитора и делать гимнастику для глаз: рисовать различные фигуры глазами, фокусировать внимание на дальние объекты и переводить фокус с одних предметов на другие. Также примерно каждые полтора-два часа необходимо выполнять зарядку;
- выбрать правильный стул или кресло. При выборе данной мебели необходимо приобрести такую, которая будет обеспечивать прямую осанку, при которой спина будет опираться в спинку.

6.3 Техника безопасности при работе с компьютером

При работе с компьютером необходимо соблюдать следующие правила, которые связаны с техникой безопасности:

- не оставлять компьютер без присмотра на длительное время;
- не использовать компьютер, который работает от питания без заземления;
- не работать за ПК, у которого есть повреждение в системном блоке или в его периферийных устройствах;
- не включать компьютер, если в помещении слишком высокая влажность;
- не класть на корпус компьютера или монитора посторонние предметы.

Заключение

В ходе выполнения данной выпускной квалификационной работы были получены навыки написания приложения для чтения книг онлайн под операционную систему Windows, изучен стек технологий и библиотек платформы .NET. Результатом работы является разработанное клиент-серверное приложение «Электронная библиотека», которое называется DigitalLibrary, а также отчёт, в котором представлены результаты тестирования разработанного программного обеспечения, описание разработки и подробный разбор его основных функциональных особенностей.

Разработанное приложение позволяет пользователю читать книги онлайн, загружать собственные книги на сервер, скачивать их на устройство, а также осуществлять поиск по всей библиотеке. После проведения тестирования было выяснено, что разработанное приложение соответствует всем поставленным в начале работы требованиям и работает корректно, не вызывая критических ошибок в системе.

В дальнейшем приложение можно усовершенствовать путем добавления поддержки шифрования данных пользователя, расширения библиотеки книг, а также возможности прослушивать аудиокниги. Разработанный интерфейс пользователя можно улучшить путем усовершенствования дизайна для повышения комфорта использования.

Список использованных источников

1. Книжная культура в России. [Электронный ресурс]/Режим доступа: <https://wciom.ru/analytical-reviews/analiticheskii-obzor/knizhnaja-kultura-2022>, свободный. (дата обращения 23.04.2024).
2. Стиллен Эндрю, Грин Дженнифер. Head First. Изучаем С# — Москва: Питер, 2022. — 42 с.
3. Документация по Windows Presentation Foundation. [Электронный ресурс]/Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netdesktop-8.0>, свободный. (дата обращения 20.04.2024).
4. Material Design In XAML. [Электронный ресурс]/Режим доступа: <http://materialdesigninxaml.net/>, свободный. (дата обращения 14.05.2024)
5. Клиент-серверное взаимодействие в С# и .NET. [Электронный ресурс]/Режим доступа: <https://metanit.com/sharp/net/3.2.php>, свободный. (дата обращения 13.05.2024)
6. Ржеуцкая С.Ю. Базы данных. Язык SQL: учеб. пособие / С.Ю. Ржеуцкая. – Вологда: ВоГТУ, 2010. – 17-22 с.

Приложение А

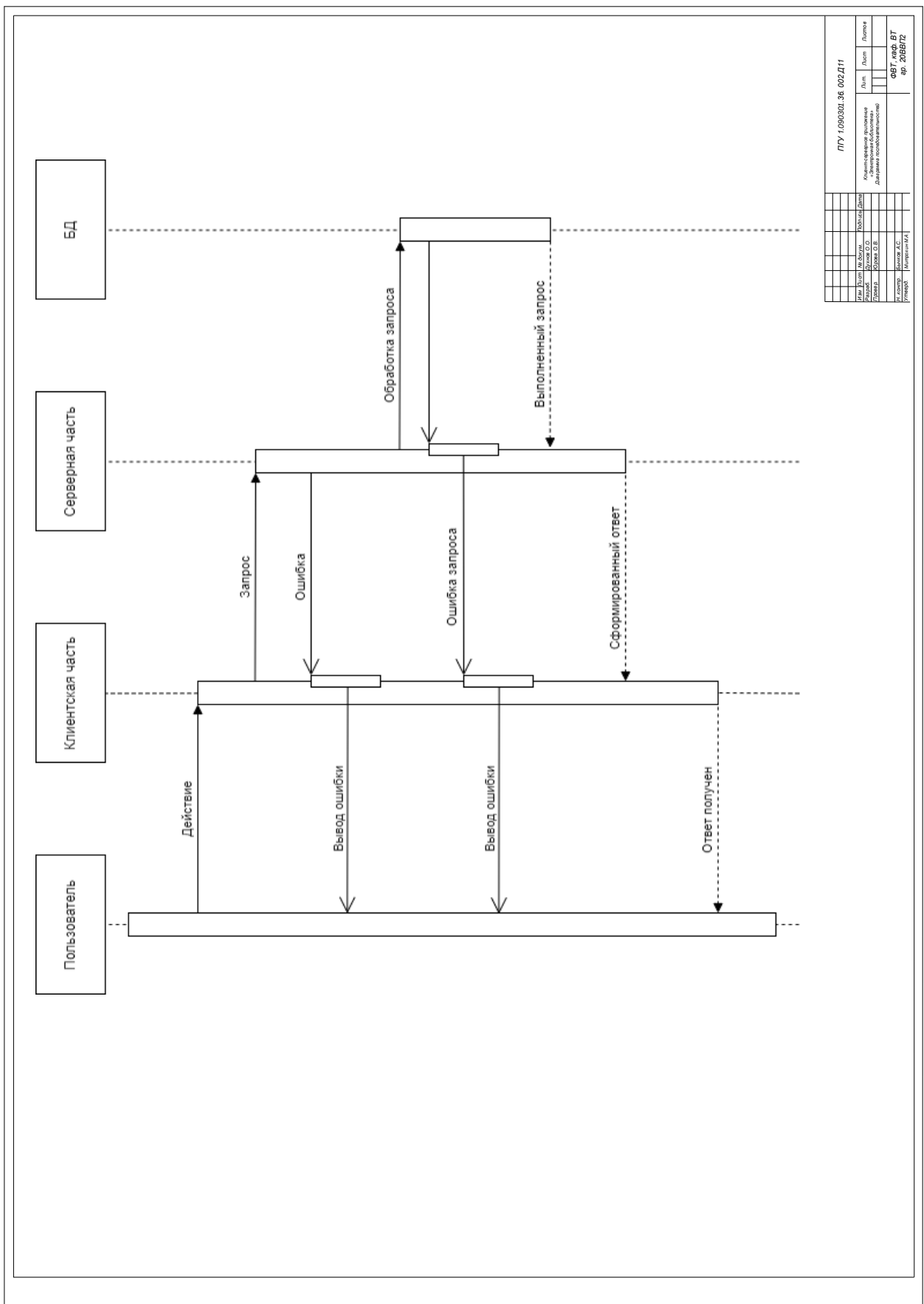
(Обязательное)

Клиент серверное приложение «Книжный интернет-магазин»

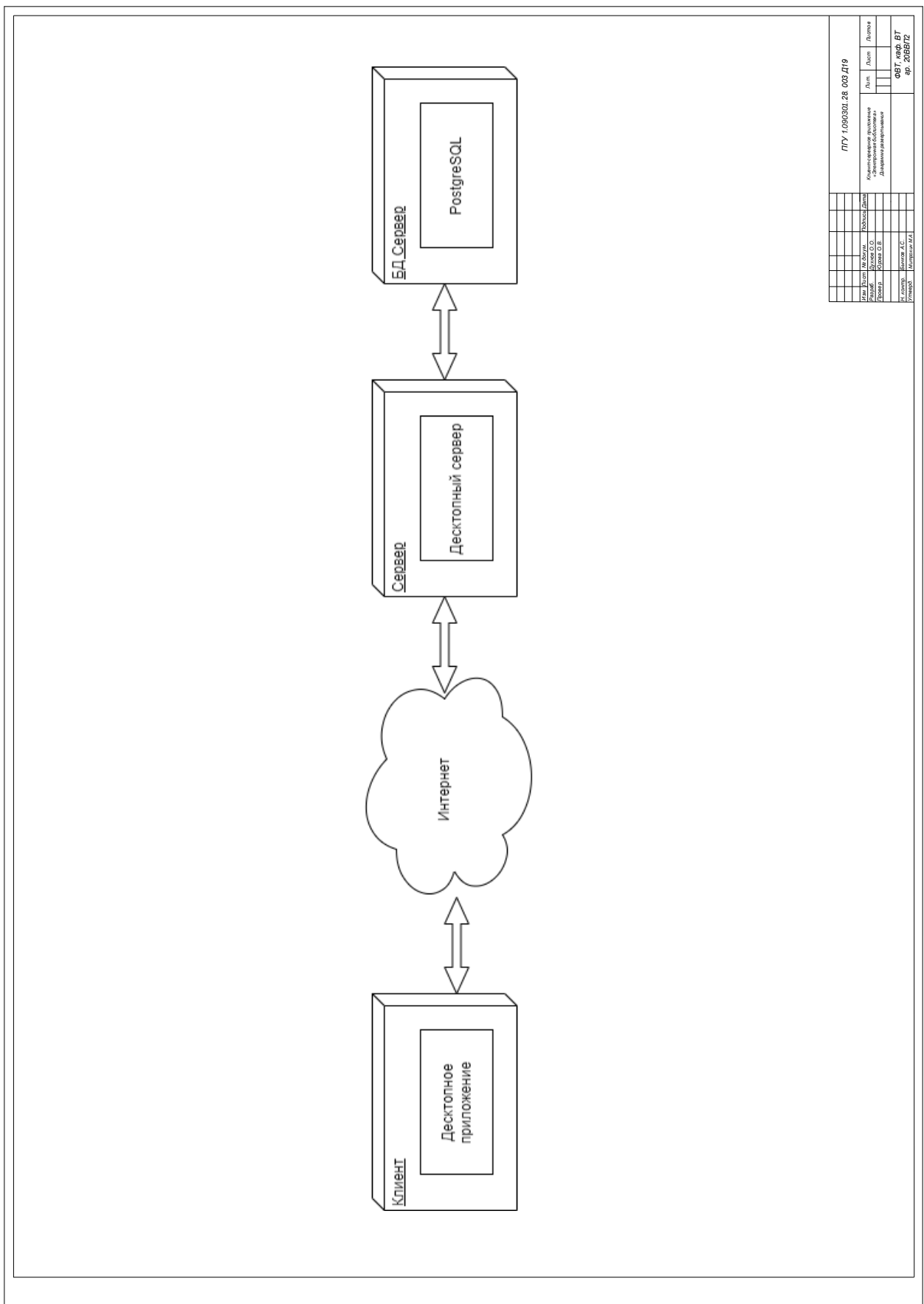
UML-диаграммы

					ПГУ 1,090301,28.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1-Диаграмма последовательностей



Приложение А.2-Диаграмма развертывания



Приложение А.3-Диаграмма классов

[illegible]

Приложение А.4-Диаграмма прецедентов



					ПГУ 1.090301.28.005 Д13			
Изм.	Лист	№ докум.	Подпись	Дата	Клиент-серверные приложения «Электронная библиотека» Дизайнерские прототипы	Лит.	Лист	Листов
Разраб.		Духов О.О.						
Провер.		Юрова О.В.						
М. контр.		Бачалов А.С.				ФВТ, каф. ВТ ар. 20.08/12		
Исполн.		М.И.И.И.						

Приложение Б

(Обязательное)

Клиент серверное приложение «Электронная библиотека»

Листинг программы

Файл Server.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace Server
{
    class Server
    {
        private static TcpListener listener;

        public Server()
        {
            const string ip = "127.0.0.1";
            const int port = 8080;

            var tcpEndPoint = new IPEndPoint(IPAddress.Parse(ip),
port);
            listener = new TcpListener(tcpEndPoint);
            listener.Start();
        }

        public void Start()
        {
            while (true)
            {
                TcpClient client = listener.AcceptTcpClient();
                ClientObject clientObject = new ClientObject(client);

                Thread clientThread = new Thread(new
ThreadStart(clientObject.Process));
                clientThread.Start();

                var endpoint = client.Client.RemoteEndPoint;
                Console.WriteLine("Пользователь подключился, его ip -
>" + endpoint.ToString());
            }
        }
    }
}
```

Файл ClientObject.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using MyHelper;
```

					ПГУ 1,090301,28.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

```

using System.Drawing;
using System.IO;

namespace Server
{
    class ClientObject
    {
        private TcpClient client;
        private NetworkStream stream;
        private byte[] data;

        public ClientObject(TcpClient tcpClient)
        {
            client = tcpClient;
            stream = client.GetStream();
        }

        public void Process()
        {
            try
            {
                var buffer = ReadDataByte();
                data = new byte[buffer.Length - 1];
                Array.Copy(buffer, 1, data, 0, data.Length);
                UserRequests userRequests = (UserRequests)buffer[0];
                ActionSelection(userRequests);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            finally
            {
                if (client != null)
                {
                    client.Close();
                }
            }
        }
    }
}

```

					<i>ПГУ 1,090301,28.001 ПЗ</i>	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

        }
        if (stream != null)
        {
            stream.Close();
        }
    }

private byte[] ReadDataByte()
{
    var buffer = new byte[65536 * 2048];
    int size = 0;
    int streamSize = 0;
    do
    {
        size = stream.Read(buffer, streamSize, buffer.Length -
streamSize);

        streamSize += size;
    } while (stream.DataAvailable);
    byte[] newBuffer = new byte[streamSize];
    Array.Copy(buffer, newBuffer, streamSize);
    return newBuffer;
}

private void ActionSelection(UserRequests userRequests)
{
    switch (userRequests)
    {
        case UserRequests.Register:
            RegisterSelection();
            break;

        case UserRequests.Authorization:
            AuthorizationSelection();
            break;

        case UserRequests.GetRecommendation:
            GetRecommendation();
            break;
    }
}

```

					ИГУ 1,090301,28.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

```

case UserRequests.GetBestBook:
    GetBestBook();
    break;
case UserRequests.GetInfoBook:
    GetInfoBook();
    break;
case UserRequests.GetAllBook:
    GetAllBook();
    break;
case UserRequests.GetMyBook:
    GetMyBook();
    break;
case UserRequests.GetSizePageAllBook:
    GetSizePageAllBook();
    break;
case UserRequests.GetSizePageMyBook:
    GetSizePageMyBook();
    break;
case UserRequests.GetMyStatistics:
    GetMyStatistics();
    break;
case UserRequests.GetQuote:
    GetQuote();
    break;
case UserRequests.ReadingBook:
    ReadingBook();
    break;
case UserRequests.GetFavoriteBook:
    GetFavoriteBook();
    break;
case UserRequests.GetSizePageFavoriteBook:
    GetSizePageFavoriteBook();
    break;
case UserRequests.AvailabilityBook:
    AvailabilityBook();
    break;

```

					<i>ПГУ 1,090301,28.001 ПЗ</i>	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

```

case UserRequests.AvailabilityFavoriteBook:
    AvailabilityFavoriteBook();
    break;
case UserRequests.DeleteBook:
    DeleteBook();
    break;
case UserRequests.DeleteFavoriteBook:
    DeleteFavoriteBook();
    break;
case UserRequests.AddBook:
    AddBook();
    break;
case UserRequests.AddUserBook:
    AddUserBook();
    break;
case UserRequests.AddStatisticsBook:
    AddStatisticsBook();
    break;
case UserRequests.AddQuote:
    AddQuote();
    break;
case UserRequests.AddFavoriteBook:
    AddFavoriteBook();
    break;
case UserRequests.UpdateStatisticsBook:
    UpdateStatisticsBook();
    break;
case UserRequests.SaveData:
    SaveData();
    break;
case UserRequests.SaveImage:
    SaveImage();
    break;
case UserRequests.SavePassword:
    SavePassword();
    break;

```

```

        case UserRequests.SearchText:
            SearchText();
            break;
    }
}

private void SearchText()
{
    var searchText =
Encoding.UTF8.GetString(data).ToLower().Trim();
    var postgresql = new PostgreSQL();
    var dataTable = postgresql.LoadAllBook();
    postgresql.Close();
    List<Book> books = new List<Book>();
    foreach (DataRow row in dataTable.Rows)
    {
        string nameBook = row[1].ToString().ToLower();
        string authBook = row[2].ToString().ToLower();
        string genreBook = row[4].ToString().ToLower();
        if (nameBook.Contains(searchText) ||
authBook.Contains(searchText) || genreBook.Contains(searchText))
        {
            string pathIcon = @"..\books\" + row[1].ToString()
+ "\\\" + row[1].ToString() + ".jpg";
            byte[] bufferIcon = new byte[65536 * 128];
            int size = 0;
            using (var streamFile = new FileStream(pathIcon,
FileMode.Open))
            {
                size = streamFile.Read(bufferIcon, 0, 65536 *
128);
            }

            byte[] newBufferIcon = new byte[size];
            Array.Copy(bufferIcon, newBufferIcon, size);
            Book book = new Book();
            book.Name = row[1].ToString();
            book.Author = row[2].ToString();

```

					ИГУ 1,090301,28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		75

```

        book.Description = row[3].ToString();
        book.Genre = row[4].ToString();
        book.Icon = new BufferIcon;
        books.Add(book);
    }
}

Helper.WriteJsonObject(books);
var bufferJson = new byte[65536 * 1024];
int sizeJson = 0;
using (FileStream fileStream = new FileStream("temp.json",
FileMode.Open))
{
    sizeJson = fileStream.Read(bufferJson, 0, 65536 *
1024);
}
stream.Write(bufferJson, 0, sizeJson);
}

private void GetFavoriteBook()
{
    byte numberPage = data[0];
    byte[] bufferLogin = new byte[data.Length - 1];
    Array.Copy(data, 1, bufferLogin, 0, bufferLogin.Length);
    string userLogin = Encoding.UTF8.GetString(bufferLogin);
    var postgresql = new PostgreSQL();
    var dataTable = postgresql.GetFavoriteBookUser(userLogin);
    postgresql.Close();
    var books = new List<Book>();
    int startBook = numberPage * 10;
    int endBook = startBook + 10 < dataTable.Rows.Count ?
startBook + 10 : dataTable.Rows.Count;

    for (; startBook < endBook; startBook++)
    {
        var row = dataTable.Rows[startBook];
        Book tempBook = new Book();
        tempBook.Name = row[1].ToString();
    }
}

```

					ИГУ 1,090301,28.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

```

        tempBook.Author = row[2].ToString();
        tempBook.Description = row[3].ToString();
        tempBook.Genre = row[4].ToString();
        string pathDirectory = @"..\books\" + tempBook.Name;
        var pathIcon = "";
        if (Directory.Exists(pathDirectory))
        {
            pathIcon = @"..\books\" + tempBook.Name + "\\\" +
tempBook.Name + ".jpg";
        }
        else
        {
            pathIcon = @"..\Users\" + userLogin + "\\\" +
tempBook.Name + "\\\" + tempBook.Name + ".jpg";
        }
        byte[] bufferIcon = new byte[65536 * 128];
        int size = 0;
        using (var streamFile = new FileStream(pathIcon,
FileMode.Open))
        {
            size = streamFile.Read(bufferIcon, 0, 65536 *
128);

        }
        byte[] newBufferIcon = new byte[size];
        Array.Copy(bufferIcon, newBufferIcon, size);
        tempBook.Icon = newBufferIcon;
        books.Add(tempBook);
    }
    Helper.WriteJsonObject(books);
    var bufferJson = new byte[65536 * 256];
    int sizeJson = 0;

    using (var streamFile = new FileStream("temp.json",
FileMode.Open))
    {
        sizeJson = streamFile.Read(bufferJson, 0, 65536 *
256);

    }
    stream.Write(bufferJson, 0, sizeJson);

```

					ПГУ 1,090301,28.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

```

    }

private void DeleteFavoriteBook()
{
    var userInfo = Encoding.UTF8.GetString(data);
    string nameUser;
    string nameBook;
    using (TextReader reader = new StringReader(userInfo))
    {
        nameUser = reader.ReadLine();
        nameBook = reader.ReadLine();
    }
    var postgresql = new PostgreSQL();
    postgresql.DeleteFavoriteBook(nameUser, nameBook);
    postgresql.Close();
}

private void AddFavoriteBook()
{
    var userInfo = Encoding.UTF8.GetString(data);
    string bufferJson;
    string nameUser;
    using (TextReader reader = new StringReader(userInfo))
    {
        nameUser = reader.ReadLine();
        bufferJson = reader.ReadLine();
    }
    var byteBufferJson = Encoding.UTF8.GetBytes(bufferJson);
    Helper.WriteJsonObject(byteBufferJson);
    Book book = (Book)Helper.ReadJsonObject<Book>();
    var postgresql = new PostgreSQL();
    postgresql.LoadFavoriteBookUser(nameUser, book);
    postgresql.Close();
}

private void GetQuote()

```

```

{
    var nameUser = Encoding.UTF8.GetString(data);
    var postgresql = new PostgreSQL();
    var table = postgresql.GetTableQuotesUsers(nameUser);
    postgresql.Close();
    List<string> quoteUser = new List<string>();
    foreach(DataRow row in table.Rows)
    {
        quoteUser.Add(row[1].ToString());
    }
    Helper.WriteJsonObject(quoteUser);
    var buffeQuote = new byte[65536];
    int size = 0;
    using (FileStream fileStream = new FileStream("temp.json",
        FileMode.Open))
    {
        size = fileStream.Read(buffeQuote, 0, 65536);
    }
    stream.Write(buffeQuote, 0, size);
}

private void AddQuote()
{
    var userInfo = Encoding.UTF8.GetString(data);
    string nameUser;
    string quote;
    using (TextReader reader = new StringReader(userInfo))
    {
        nameUser = reader.ReadLine();
        quote = reader.ReadToEnd();
    }
    var postgresql = new PostgreSQL();
    postgresql.LoadQuotesUser(nameUser, quote);
    postgresql.Close();
}

private void GetMyStatistics()

```

```

    {
        var nameUser = Encoding.UTF8.GetString(data);
        var postgresql = new PostgreSQL();
        var tableStatistics =
postgresql.GetTableStatisticsUsers(nameUser);
        postgresql.Close();
        StatisticsBook statisticsTable = new StatisticsBook();
        foreach (DataRow row in tableStatistics.Rows)
        {
            statisticsTable.ReadingBook = row[2].ToString();
            statisticsTable.ReadingPage = row[3].ToString();
            statisticsTable.WastedTime = row[4].ToString();
            statisticsTable.SpeedReading = row[5].ToString();
        }
        Helper.WriteJsonObject(statisticsTable);
        var bufferJson = new byte[65536];
        int size = 0;
        using (FileStream fileStream = new FileStream("temp.json",
FileMode.Open))
        {
            size = fileStream.Read(bufferJson, 0, 65536);
        }
        var response = new byte[size];
        Array.Copy(bufferJson, 0, response, 0, size);
        stream.Write(response, 0, response.Length);
    }

private void UpdateStatisticsBook()
{
    var userInfo = Encoding.UTF8.GetString(data);
    string nameUser;
    string strJson;
    using (TextReader reader = new StringReader(userInfo))
    {
        nameUser = reader.ReadLine();
        strJson = reader.ReadToEnd();
    }
}

```

```

        var bufferJson = Encoding.UTF8.GetBytes(strJson);
        Helper.WriteJsonObject(bufferJson);

        StatisticsBook bookUser =
(StatisticsBook)Helper.ReadJsonObject<StatisticsBook>();

        var postgreSqlAdd = new PostgreSql();
        var tableStatistics =
postgreSqlAdd.GetTableStatisticsUser(nameUser, bookUser.NameBook);
        postgreSqlAdd.Close();

        StatisticsBook statisticsTable = new StatisticsBook();
        foreach (DataRow row in tableStatistics.Rows)
        {
            statisticsTable.NameBook = row[1].ToString();
            statisticsTable.ReadingPage = row[2].ToString();
            statisticsTable.WastedTime = row[3].ToString();
            statisticsTable.NumberReadingPage = row[4].ToString();
            statisticsTable.SpeedReading = row[5].ToString();
        }

        if (statisticsTable.ReadingBook == null ||
statisticsTable.ReadingBook.Equals("No"))
        {
            StatisticsBook generalStatistics = new
StatisticsBook();

            int startReadingPage =
GetLinesLenght(statisticsTable.ReadingPage);

            string strReadingPage =
GetUniqueString(statisticsTable.ReadingPage, bookUser.ReadingPage);

            int numberReadingPage =
GetLinesLenght(strReadingPage);

            int numberPageRead = numberReadingPage -
startReadingPage;

            DateTime dateTimeTable =
DateTime.Parse(statisticsTable.WastedTime);

            DateTime dateTimeUser =
DateTime.Parse(bookUser.WastedTime);

            int secondTable = GetSecond(dateTimeTable);
            int secondUser = GetSecond(dateTimeUser);
            int secondResult = secondTable + secondUser;

            TimeSpan resultTime =
TimeSpan.FromSeconds(secondResult);

```

```

        TimeSpan secondUserTime =
TimeSpan.FromSeconds(secondUser);

        statisticsTable.ReadingPage = strReadingPage;
        statisticsTable.WastedTime = resultTime.ToString();

        if (numberReadingPage ==
Convert.ToInt32(bookUser.NumberReadingPage))
        {
            statisticsTable.ReadingBook = "Yes";
            generalStatistics.ReadingBook = "1";
        }
        else
        {
            statisticsTable.ReadingBook = "No";
            generalStatistics.ReadingBook = "0";
        }

        generalStatistics.ReadingPage =
numberPageRead.ToString();

        generalStatistics.WastedTime =
secondUserTime.ToString();

        generalStatistics.NameBook = statisticsTable.NameBook;
        var updateStatisticsUser = new PostgreSQL();
updateStatisticsUser.UpdateStatisticsBookUser(statisticsTable,
nameUser);

        updateStatisticsUser.Close();
        UpdateStatisticsGeneral(generalStatistics, nameUser);
    }
}

private int GetLinesLenght(string text)
{
    int count = 0;
    using (TextReader reader = new StringReader(text))
    {
        string line;
        while ((line = reader.ReadLine()) != null)
        {
            count++;
        }
    }
}

```

```

        }

        return count;
    }

    private void AddStatisticsBook()
    {
        var userInfo = Encoding.UTF8.GetString(data);
        string nameBook;
        string nameUser;
        using (TextReader reader = new StringReader(userInfo))
        {
            nameBook = reader.ReadLine();
            nameUser = reader.ReadLine();
        }
        var postgresql = new PostgreSQL();
        var checkAvailability =
        postgresql.GetAvailabilityStatisticsUser(nameUser, nameBook);
        postgresql.Close();
        if (checkAvailability == false)
        {
            var postgresqlAdd = new PostgreSQL();
            postgresqlAdd.LoadStatisticsUser(nameUser, nameBook);
            postgresqlAdd.Close();
        }
    }

    private void AddUserBook()
    {
        var userInfo = Encoding.UTF8.GetString(data);
        string bufferJson;
        string nameUser;
        using (TextReader reader = new StringReader(userInfo))
        {
            nameUser = reader.ReadLine();
            bufferJson = reader.ReadLine();
        }
        var byteBufferJson = Encoding.UTF8.GetBytes(bufferJson);
    }

```

```

        Helper.WriteJsonObject(byteBufferJson);
        Book book = (Book)Helper.ReadJsonObject<Book>();
        var postgresql = new PostgreSQL();
        postgresql.LoadNewBookUser(book, nameUser);
        postgresql.Close();
        Directory.CreateDirectory(@".\Users\" + nameUser + "\\\" +
book.Name);
        string pathFile = @".\Users\" + nameUser + "\\\" +
book.Name + "\\\" + book.Name + ".fb2";
        string pathIcon = @".\Users\" + nameUser + "\\\" +
book.Name + "\\\" + book.Name + ".jpg";
        using (FileStream fileStream = new FileStream(pathFile,
FileMode.Create))
        {
            fileStream.Write(book.BookByte, 0,
book.BookByte.Length);
        }
        using (FileStream fileStream = new FileStream(pathIcon,
FileMode.Create))
        {
            fileStream.Write(book.Icon, 0, book.Icon.Length);
        }
    }

    private void SaveData()
    {
        Helper.WriteJsonObject(data);
        User user = (User)Helper.ReadJsonObject<User>();
        var postgresql = new PostgreSQL();
        postgresql.UpdateUserData(user.Name, user.Surname,
user.Login);
        postgresql.Close();
    }

    private void SavePassword()
    {
        Helper.WriteJsonObject(data);
        User user = (User)Helper.ReadJsonObject<User>();

```

```

        var postgresql = new PostgreSQL();
        postgresql.UpdateUserPassword(user.Password, user.Login);
        postgresql.Close();
    }

    private void SaveImage()
    {
        var userInfo = Encoding.UTF8.GetString(data);
        string strIcon;
        string nameUser;
        using (TextReader reader = new StringReader(userInfo))
        {
            nameUser = reader.ReadLine();
            strIcon = reader.ReadToEnd();
        }
        var bufferName = Encoding.UTF8.GetBytes(nameUser);
        byte[] bufferIcon = new byte[data.Length -
bufferName.Length - 1];
        Array.Copy(data, bufferName.Length + 1, bufferIcon, 0,
bufferIcon.Length);
        string pathIcon = @".\Users\" + nameUser + "\\\" + nameUser
+ ".jpg";
        using (FileStream fileStream = new FileStream(pathIcon,
FileMode.Create))
        {
            fileStream.Write(bufferIcon, 0, bufferIcon.Length);
        }
        var postgresql = new PostgreSQL();
        postgresql.UpdateImageUser(pathIcon, nameUser);
        postgresql.Close();
    }

    private void DeleteBook()
    {
        var userInfo = Encoding.UTF8.GetString(data);
        string nameBook;
        string nameUser;
        using (TextReader reader = new StringReader(userInfo))

```

					ИГУ 1,090301,28.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            nameUser = reader.ReadLine();
            nameBook = reader.ReadLine();
        }
        var postgresql = new PostgreSQL();
        postgresql.DeleteUserBook(nameUser, nameBook);
        postgresql.Close();
    }

    private void AddBook()
    {
        var userInfo = Encoding.UTF8.GetString(data);
        string bufferJson;
        string nameUser;
        using (TextReader reader = new StringReader(userInfo))
        {
            nameUser = reader.ReadLine();
            bufferJson = reader.ReadLine();
        }
        var byteBufferJson = Encoding.UTF8.GetBytes(bufferJson);
        Helper.WriteJsonObject(byteBufferJson);
        Book book = (Book)Helper.ReadJsonObject<Book>();

        var postgresql = new PostgreSQL();
        postgresql.LoadNewBookUser(book, nameUser);
        postgresql.Close();
    }

    private void GetMyBook()
    {
        byte numberPage = data[0];
        byte[] bufferLogin = new byte[data.Length - 1];
        Array.Copy(data, 1, bufferLogin, 0, bufferLogin.Length);
        string userLogin = Encoding.UTF8.GetString(bufferLogin);
        var postgresql = new PostgreSQL();
        var dataTable = postgresql.LoadMyBook(userLogin);
    }

```

```

        postgresql.Close();

        var books = new List<Book>();

        int startBook = numberPage * 10;

        int endBook = startBook + 10 < dataTable.Rows.Count ?
startBook + 10 : dataTable.Rows.Count;

        for (; startBook < endBook; startBook++)
        {
            var table = dataTable.Rows[startBook];

            string nameBook = table[1].ToString();

            string authBook = table[2].ToString();

            string descriptionBook = table[3].ToString();

            string genreBook = table[4].ToString();

            string pathFolder = @".\books\" + nameBook;

            string pathIcon = "";

            if (Directory.Exists(pathFolder))
            {
                pathIcon = String.Format(@"D:\8
сестрп\BKP\Digital Library\Server\bin\Debug\books\{0}\{1}.jpg",
nameBook, nameBook);
            }

            else
            {
                pathIcon = @".\Users\" + userLogin + "\\\" +
nameBook + "\\\" + nameBook + ".jpg";
            }

            Bitmap image = new Bitmap(pathIcon);

            ImageConverter converter = new ImageConverter();

            byte[] bImage = (byte[])converter.ConvertTo(image,
typeof(byte[]));

            Book book = new Book(nameBook, authBook,
descriptionBook, genreBook, bImage);

            books.Add(book);
        }

        byte[] buffer = new byte[65536 * 256];

        Helper.WriteJsonObject(books);

        int size;

        using (var streamFile = new FileStream("temp.json",
FileMode.Open))
        {

```

					ИГУ 1,090301,28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

```

        size = streamFile.Read(buffer, 0, 65536 * 256);
    }

    byte[] newBuffer = new byte[size];
    Array.Copy(buffer, newBuffer, size);
    stream.Write(newBuffer, 0, size);
    var bufferResponse = ReadDataByte();
}

private void ReadingBook()
{
    try
    {
        var userInfo = Encoding.UTF8.GetString(data);
        string nameUser;
        string nameBook;

        using (TextReader reader = new StringReader(userInfo))
        {
            nameBook = reader.ReadLine();
            nameUser = reader.ReadLine();
        }

        string pathDirectory = @"..\books\" + nameBook;
        var pathBook = "";

        if (Directory.Exists(pathDirectory))
        {
            pathBook = @"..\books\" + nameBook + "\\\" +
nameBook + ".fb2";
        }
        else
        {
            pathBook = @"..\Users\" + nameUser + "\\\" +
nameBook + "\\\" + nameBook + ".fb2";
        }

        byte[] bufferBook = new byte[65536 * 512];

```

					ИГУ 1,090301,28.001 ПЗ	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		

```

        int size;

        using (var streamFile = new FileStream(pathBook,
FileMode.Open))
        {
            size = streamFile.Read(bufferBook, 0, 65536 *
512);
        }

        byte[] newBuffer = new byte[size];
        Array.Copy(bufferBook, newBuffer, size);

        stream.Write(newBuffer, 0, newBuffer.Length);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}

private void GetInfoBook()
{
    try
    {
        var userInfo = Encoding.UTF8.GetString(data);
        string nameUser;
        string nameBook;
        using (TextReader reader = new StringReader(userInfo))
        {
            nameBook = reader.ReadLine();
            nameUser = reader.ReadLine();
        }
        var postgresql = new PostgreSQL();
        var dataTable = postgresql.SearchBook(nameBook);
        postgresql.Close();
        if (dataTable.Rows.Count == 0)
        {

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1,090301,28.001 ПЗ

Лист

89

```

        var postgresqlnew = new PostgreSQL();
        dataTable = postgresqlnew.SearchBookUser(nameBook,
nameUser);

        postgresqlnew.Close();
    }
    string authorBook = "";
    string descriptionBook = "";
    string genreBook = "";
    foreach (DataRow row in dataTable.Rows)
    {
        authorBook = row[2].ToString();
        descriptionBook = row[3].ToString();
        genreBook = row[4].ToString();
    }
    Book book = new Book(nameBook, authorBook,
descriptionBook, genreBook);
    Helper.WriteJsonObject(book);
    byte[] buffer = new byte[65536 * 64];
    int size;
    using (var streamFile = new FileStream("temp.json",
FileMode.Open))
    {
        size = streamFile.Read(buffer, 0, 65536 * 64);
    }
    byte[] newBuffer = new byte[size];
    Array.Copy(buffer, newBuffer, size);
    stream.Write(newBuffer, 0, size);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
}

private void RegisterSelection()
{
    Helper.WriteJsonObject(data);
}

```

```

var user = (User)Helper.ReadJsonObject<User>();
try
{
    PostgreSQL postgresql = new PostgreSQL();
    var dataTable = postgresql.UploadRegisteredUsersAll();
    var checkResult = CheckAvailNameUser(dataTable, user);
    if (!checkResult)
    {
        stream.WriteByte((byte)UserRequests.Ok);
        postgresql.LoadRegisteredUsers(user.Login,
user.Password);

        postgresql.Close();
        Directory.CreateDirectory(@".\Users\" +
user.Login);

        PostgreSQL postgresqlCreate = new PostgreSQL();
        postgresqlCreate.CreateTableUser(user.Login);
        postgresqlCreate.Close();

        PostgreSQL addStatistics = new PostgreSQL();
        addStatistics.LoadNewStatisticsUser(user.Login);
        addStatistics.Close();

        PostgreSQL creatStatisticsUser = new PostgreSQL();

        creatStatisticsUser.CreateTableStatisticsUser(user.Login);
        creatStatisticsUser.Close();

        PostgreSQL creatFavoriteBooksUser = new
PostgreSQL();

        creatFavoriteBooksUser.CreateTableFavoriteBook(user.Login);
        creatFavoriteBooksUser.Close();

        PostgreSQL creatQuotesUser = new PostgreSQL();
        creatQuotesUser.CreateTableQuotesUser(user.Login);
        creatQuotesUser.Close();
    }
    else
    {
        stream.WriteByte((byte)UserRequests.Error);
    }
}

```

```

        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            stream.WriteByte((byte)UserRequests.Error);
        }
    }
}

```

Файл Networking.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;

namespace MyHelper
{
    public class Networking
    {
        private Socket tcpSocket;

        public Networking()
        {
            const string ip = "127.0.0.1";
            const int port = 8080;

            var tcpEndPoint = new IPEndPoint(IPAddress.Parse(ip),
port);

            tcpSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
            tcpSocket.Connect(tcpEndPoint);
        }

        public void SendData(string message)
        {
            var data = Encoding.UTF8.GetBytes(message);

```

					ИГУ 1,090301,28.001 ПЗ	Лист
						92
Изм.	Лист	№ докум.	Подпись	Дата		

```

        tcpSocket.Send(data);
    }

    public void SendData(byte[] data)
    {
        tcpSocket.Send(data);
    }

    public StringBuilder RecvDataString()
    {
        var buffer = new byte[65536];
        var size = 0;
        var dataRecv = new StringBuilder();
        do
        {
            size = tcpSocket.Receive(buffer);
            dataRecv.Append(Encoding.UTF8.GetString(buffer, 0,
size));
        }
        while (tcpSocket.Available > 0);
        return dataRecv;
    }

    public byte[] RecvDataByte()
    {
        var buffer = new byte[65536 * 2048];
        int size = 0;
        int streamSize = 0;
        do
        {
            size = tcpSocket.Receive(buffer, streamSize,
buffer.Length - streamSize, SocketFlags.None);
            streamSize += size;
        } while (tcpSocket.Available > 0);
        byte[] newBuffer = new byte[streamSize];
        Array.Copy(buffer, newBuffer, streamSize);
        return newBuffer;
    }

```

					ПГУ 1,090301,28.001 ПЗ	Лист
						93
Изм.	Лист	№ докум.	Подпись	Дата		

```

        public void Close()
        {
            tcpSocket.Shutdown(SocketShutdown.Both);
            tcpSocket.Close();
        }
    }
}

```

Файл Json.cs

```

using System;
using System.IO;
using System.Runtime.Serialization.Json;

namespace MyHelper
{
    static class Json
    {
        public static void WriteJsonFile<T>(string path, T obj)
        {
            var jsonFormatter = new
DataContractJsonSerializer(typeof(T));
            using (var file = new FileStream(path, FileMode.Create))
            {
                jsonFormatter.WriteObject(file, obj);
            }
        }

        public static object ReadJsonFile<T>(string path)
        {
            var jsonFormatter = new
DataContractJsonSerializer(typeof(T));
            object jsonObj;
            using (var file = new FileStream(path, FileMode.Open))
            {
                jsonObj = (T)jsonFormatter.ReadObject(file);
            }
            return jsonObj;
        }
    }
}

```

					ПГУ 1,090301,28.001 ПЗ	Лист
						94
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Клиент-серверное приложение «Электронная библиотека»

Презентация

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ «ЭЛЕКТРОННАЯ БИБЛИОТЕКА»

Автор работы:

студент группы 20ВВП2 Духнов Олег Олегович

Научный руководитель:

к.т.н., доцент, Юрова Ольга Викторовна

1

Цель и объекты работы

Цель работы – разработка клиент-серверного приложения для чтения книг онлайн.

Объектами разработки являются программы клиента и сервера, которые выполняют следующие функции:

- Обработка пользовательских данных и передачи их от программы сервера к клиенту;
- Отображение пользовательских данных через интерфейс клиентской программы;
- Чтение/добавление/скачивание книг;

2

					ПГУ 1,090301,28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		96

Актуальность темы

В настоящее время онлайн-библиотеки являются одним из основных источников потребления книг и с развитием информационных технологий пользователи всё чаще обращаются к электронным приложениям для чтения книг.

Согласно официальной статистики ВЦИОМ, 72% россиян читают книги в электронном виде, из них 20% читают на персональном компьютере. Исходя из данной статистики, можно сделать вывод, что тема актуальная, а также присутствует охват аудитории.

3

Стек используемых технологий

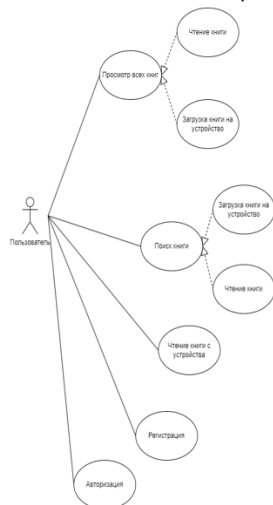


4

					ПГУ 1,090301,28.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		97

Варианты использования приложения

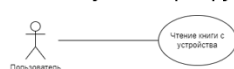
Пользователь не авторизован



Пользователь авторизован

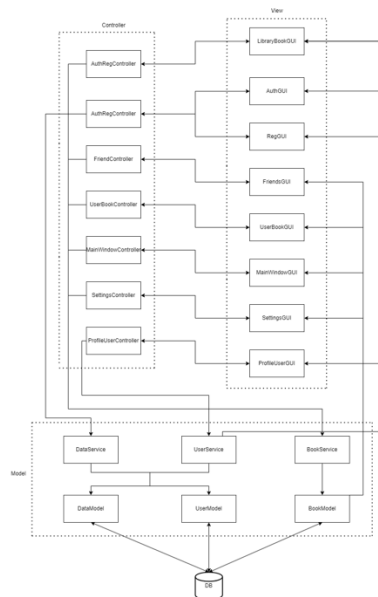


Нет доступа к серверу



5

Архитектура приложения



Шаблон проектирования был выбран Model-View-Controller (MVC).

Model предоставляет данные предметной области представлению и реагирует на команды контроллера, изменяя свое состояние.

View отвечает за отображение данных предметной области (модели) пользователю, реагируя на изменения модели.

Controller интерпретирует действия пользователя, оповещая модель о необходимости изменений.

6

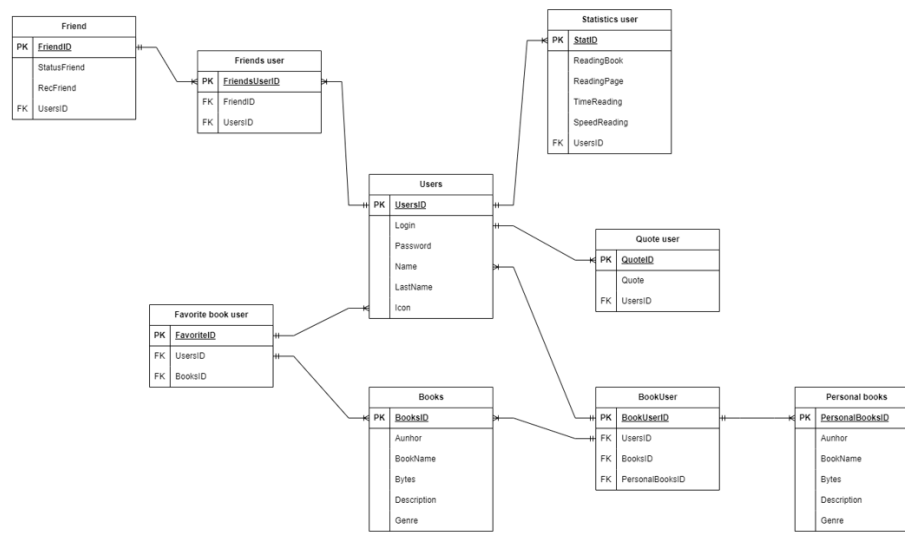
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1,090301,28.001 ПЗ

Лист

98

Структура базы данных



7

Заключение

В ходе выполнения данной выпускной квалификационной работы были разработаны программы сервера и клиента, которые позволяют читать книги онлайн.

Разработанное приложение позволяет пользователю читать книги онлайн, загружать собственные книги на сервер, скачивать их на устройство, а также осуществлять поиск по всей библиотеке.

8

Спасибо за внимание!

9

					ПГУ 1,090301,28.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		