

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства поддержки процессов
организации и функционирования
муниципальной электронной библиотеки**

Студент _____ Аникин Анатолий Игоревич _____
(подпись, дата) (ФИО полностью)

Руководитель _____ Князев В.Н. _____
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В. _____
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю. _____
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А. _____
(подпись) (фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«__» _____ 202_ г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент _____ Аникин Анатолий Игоревич _____ гр. 19ВП1 факультета ФВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства поддержки процессов организации и функционирования
муниципальной электронной библиотеки

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 2023

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать приложение для мобильных устройств, обеспечивающее поддержку
процессов организации и функционирования муниципальной электронной библиотеки

2) *Среда выполнения приложения: операционная система Android 8 и выше*

Среда выполнения сервера: операционная система Windows 10 и выше

Языковые средства разработки: C#, UML, XAML, SQL

Инструментальные средства разработки: Microsoft SQL Server 2019,

Microsoft Visual Studio 2022, Microsoft Project, StarUML, Visual Paradigm 17.0

Технологии разработки: RUP, ООП, MVC, XAMARIN, Entity Framework

3) *Основные функции мобильного приложения:*

а) скачивание книг из числа имеющихся в системе;

б) поиск книг среди имеющихся в соответствии с заданными пользователем

запросом;

в) регистрация и авторизация пользователя;

г) добавление оценок и комментариев к книгам;

д) сортировка перечня книг по автору, названию, средней оценке;

е) удаление книг с мобильного устройства;

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 133 листов, 65 рисунков, 16 таблиц, 32 использованных источника, 7 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ЭЛЕКТРОННАЯ МУНИЦИПАЛЬНАЯ БИБЛИОТЕКА, ANDROID, MS SQL SERVER, XAMARIN, MVC, API.NET CORE, C#, RUP, UML

Объектом разработки является мобильное приложение для поддержки сервисов электронной муниципальной библиотеки.

Цель работы – разработка мобильного приложения электронной муниципальной библиотеки, позволяющего регистрироваться и авторизовываться читателям, искать, скачивать и удалять с устройства уже имеющиеся книги и добавлять оценки и комментарии к этим книгам.

Разработка серверной части происходила на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2022. В качестве СУБД была использована СУБД MS SQL Server.

Разработка приложения производилась на языке программирования C#. В качестве среды разработки использовалась среда программирования Visual Studio 2022, с применением фреймворка для кроссплатформенной разработки мобильных приложений Xamarin. Приложение предназначено для использования под управлением ОС Android версии 8.0 и выше, а его серверная часть работает под управлением операционной системы Windows 10 и выше.

					ПГУ 09.03.04 – 08БР191.01 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программные средства поддержки процессов организации и функционирования муниципальной электронной библиотеки. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Аникин А.И.					4	133
Пров.		Князев В.Н				Группа 19ВП1		
Реценз.								
Н. контр.		Такташкин Д.В						
Утверд.		Козлов А.Ю.						

Содержание

Введение	7
1 Анализ предметной области и требований.....	10
1.1 Анализ предметной области и постановка задачи.....	10
1.2 Анализ программных продуктов аналогичного назначения.....	10
1.2.1 Пензенская областная библиотека имени М. Ю. Лермонтова	11
1.2.2 Муниципальное бюджетное учреждение «Кузнецкая центральная городская библиотека им. А.Н. Радищева»	13
1.2.3 Муниципальное бюджетное учреждение «Карпогорская межпоселенческая библиотека» Пинежского муниципального района Архангельской области.....	15
1.2.4 Сравнение аналогов	16
1.3 Анализ целевой аудитории	16
1.4 Анализ функциональных требований	20
1.5 Анализ нефункциональных требований	23
1.6 Планирование разработки и оценка бюджета.....	24
2 Проектирование мобильного приложения	36
2.1 Выбор технологий и инструментальных средств разработки	36
2.2 Проектирование архитектуры программного обеспечения.....	37
2.3 Проектирование интерфейса пользователя	39
2.4 Проектирование базы данных	43
2.5 Проектирование серверной части приложения.....	45
3 Реализация мобильного приложения	47
3.1 Реализация базы данных.....	47
3.2 Разработка серверной части	47
3.3 Реализация клиентской части приложения	50
4 Тестирование программного продукта.....	53
4.1 Тестирование серверной части приложения.....	53
4.2 Тестирование клиентской части приложения.....	54

4.2.1 Функциональное тестирование	54
4.2.2 Модульное тестирование	55
4.3 Метрики кода.....	57
4.3.1 Метрика размера программы	59
4.3.2 Метрики Мак-Кейба.....	59
4.3.3 Метрика Холстеда.....	60
Заключение.....	63
Список использованных источников	65
Приложение А. Глоссарий.....	69
Приложение Б. Диаграммы деятельности	72
Приложение В. Исходный код sql скриптов базы данных.....	75
Приложение Г. Исходный код серверной части.....	78
Приложение Д. Исходный код мобильного приложения	89
Приложение Е. Результаты тестирования серверной части.....	118
Приложение Ж. Результаты тестирования мобильного приложения	125

Введение

В наше время информационные технологии все больше проникают в различные сферы жизни человека. Компьютеры становятся неотъемлемой частью экономических процессов по всему миру, изменяя жизнь людей. Россия также идет по пути максимального использования информационных технологий во всех сферах деятельности, что поддерживается на государственном уровне. Для достижения этой цели была утверждена «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы», в которой были поставлены основные цели и задачи политики России в сфере информационных технологий [1]. Также была принята государственная программа «Цифровая экономика», призванная обеспечить «развитие цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост и экономический суверенитет» [2].

Одной из ключевых особенностей этого развития является распространение мобильных устройств с доступом к интернету, которые постепенно заменяют традиционный доступ к сети. Согласно государственной программе «Цифровая экономика» в 2016 году «на 100 человек приходилось 159,95 мобильного телефона и из 100 человек 71,29 человека использовали мобильный доступ к сети Интернет» [2]. Данная тенденция сохраняется и сейчас.

Библиотеки, как и любые другие организации, стараются идти в ногу со временем и адаптироваться к современным тенденциям. Электронные библиотеки становятся все более актуальными и популярными в наше время. С развитием технологий и доступности Интернета, люди все чаще обращаются к электронным источникам информации вместо традиционных бумажных книг.

Одним из главных преимуществ электронных библиотек является их доступность. Электронные книги можно скачивать и читать в любом месте и в любое время, не завися от наличия физической книги или доступности

библиотеки. Это особенно удобно для тех, кто живет в удаленных районах или не имеет возможности посещать библиотеки.

Кроме того, электронные библиотеки предлагают широкий выбор книг и других материалов, которые не всегда доступны в традиционных библиотеках. Это включает в себя редкие книги, научные статьи и журналы, а также материалы на разных языках.

Современные электронные библиотеки также предлагают удобный поиск и классификацию книг, что позволяет быстро найти нужную информацию и организовать свою работу с литературой.

Наконец, электронные библиотеки являются экологически более безопасными, так как не требуют использования бумажных книг и других материалов.

Важность внедрения информационных технологий в сфере библиотечного дела была признана в том числе и на государственном уровне. 13 марта 2021 года вышла «Стратегия развития библиотечного дела в Российской Федерации на период до 2030 года». В ней подчеркивается, что «изменение способов производства и потребления информации влечет за собой изменение моделей информационно-библиотечного обслуживания пользователей» [3]. Основной задачей современных библиотек стала их модернизация, включающая в себя разработку и внедрение новых форматов хранения и методов работы с литературой. Одной из главных тенденций современных библиотек является полномасштабная оцифровка их книжных фондов [4].

С учетом вышеперечисленных преимуществ, можно сделать вывод, что электронные библиотеки являются актуальным и важным инструментом для получения и организации информации в нашей современной эпохе.

Однако, чтобы использовать электронные библиотеки, пользователи должны иметь доступ к компьютеру или ноутбуку. С развитием мобильных устройств, все большее количество людей предпочитают использовать смартфоны и планшеты для чтения электронных книг.

В связи с сказанным выше, можно с уверенностью сказать, что создание мобильного приложения для поддержки процессов организации и функционирования муниципальной электронной библиотеки становится необходимостью.

Для данной работы были выделены следующие задачи:

- провести анализ предметной области;
- проанализировать требования к разрабатываемой программе;
- произвести проектирование программных средств и структур данных к разрабатываемому приложению;
- произвести планирование разработки и расчет бюджета;
- произвести реализацию программного продукта;
- провести тестирование разработанного средства;
- оценить качество программного кода.

Целью данной выпускной квалификационной работы является проектирование программных средств и структур данных для поддержки процессов организации и функционирования муниципальной электронной библиотеки, расчет бюджета для разрабатываемого приложения, реализация и тестирование мобильного приложения.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Цель разрабатываемого мобильного приложения – обеспечить удобный доступ к электронной муниципальной библиотеке, позволяя пользователям избежать необходимости посещения библиотеки для получения нужной книги. Выполнение этой задачи возможно благодаря мобильному устройству с установленным приложением и подключению к интернету. Разрабатываемое программное средство позволяет получить доступ к книгам из любого места, где есть мобильный интернет или Wi-Fi.

В приложении должен быть доступ к любой электронной книге из электронной библиотеки, а также предоставлен доступ к удобной системе поиска, которая поможет пользователям выбрать то, что их интересует. Кроме того, должна быть реализована возможность оставлять свои комментарии и оценки литературе, которая представлена в приложении.

Таким образом, в приложении должны быть представлены следующие основные функции:

- регистрация новых пользователей;
- вход зарегистрированных пользователей в систему;
- просмотр информации о книгах;
- скачивание книг;
- поиск книг;
- оставление оценок и комментариев.

Реализации данных функций будет достаточно для корректной работы программного средства и выполнения возложенных на него задач.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время существует целый ряд муниципальных электронных библиотек. Рассмотрим некоторые известные интернет-ресурсы такого рода и произведем для них сравнительный анализ.

1.2.1 Пензенская областная библиотека имени М. Ю. Лермонтова

Одним из базовых элементов культурной, образовательной и информационной инфраструктуры пензенского региона является Пензенская областная библиотека имени Михаила Юрьевича Лермонтова. Она выполняет важнейшие социальные и коммуникативные функции, а также является старейшим учреждением культуры Пензенской области. В 1892 году библиотека была основана как первая в мире именная Лермонтовская библиотека представителями пензенской интеллигенции (врачами, учителями, юристами) при поддержке местных органов государственной власти и общественного самоуправления с целью содействия образованию и самообразованию жителей Пензенской губернии путем чтения [5].

Главная страница Пензенской областной библиотеки имени М. Ю. Лермонтова представлена на рисунке 1.



Рисунок 1 – Главная страница Пензенской областной библиотеки имени М. Ю. Лермонтова

Фонд библиотеки составляет 2,2 млн. единиц хранения. Это рукописи, книги, периодические и нотные издания, нормативно-техническая документация, наглядные пособия, аудио и видеодокументы, электронные издания и документы на микроформах.

Фонд включает также издания для инвалидов по зрению. По языковому составу документов фонд представлен:

- документами на русском языке и на языках народов Российской Федерации;
- документами на иностранных языках: романо-германском (латиница), славянском (кириллица), языках стран СНГ и Балтии [6].

Ежегодное пополнение фонда составляет 14 тыс. экземпляров, в том числе книг и электронных изданий – 10 тыс.

На сегодня, пользователям данной электронной библиотеки доступны следующие функциональные возможности:

- регистрация пользователя;
- авторизация пользователя;
- поиск литературы по электронному каталогу;
- заказ литературы;
- контроль выполнения заказа.

Основным недостатком данной библиотеки является то, что электронный каталог содержит лишь информацию о наличии литературы в библиотеки и не позволяет получить электронную версию искомой литературы.

Также, хотя сайт достаточно удобен в использовании, поисковая система для литературы сделана через чур сложной и крайне неудобной для новых пользователей.

Кроме того, данный портал не только не имеет мобильного приложения, но и работает не стабильно на мобильных системах в принципе.

1.2.2 Муниципальное бюджетное учреждение «Кузнецкая центральная городская библиотека им. А.Н. Радищева»

Центральная городская библиотека имени А.Н. Радищева в Кузнецке была создана для обеспечения права граждан на свободный доступ к информации. Центральная городская библиотека была основана в 1897 году.

Основными направлениями деятельности библиотеки являются: краеведение, продвижение чтения среди детей и молодежи, гражданское и правовое просвещение, интеллектуальное и творческое развитие детей и молодежи, патриотическое воспитание, экологическое воспитание [7].

Библиотека регулярно организует встречи с местными авторами. Наиболее популярной формой совместной работы с детьми и молодежью является организация квестов.

Библиотеки активно создают и развивают веб-ресурсы по краеведению своего поколения. Базы данных, созданные библиотеками, доступны пользователям в отдаленных районах.

Ежегодно библиотекой пользуются более 23 000 человек. Фонд библиотеки насчитывает около 200 000 единиц хранения [8].

Главная страница МБУ «Кузнецкая центральная городская библиотека им. А.Н. Радищева» представлена на рисунке 2.

Пользователи этой электронной библиотеки могут искать литературу по электронному каталогу, просматривать электронные версии литературы (хотя список электронных версий очень ограничен), заказывать физические книги, сохранять поступившие отправления и просматривать списки будущих покупок библиотеки.

Поддерживается также версия для людей с ослабленным зрением.

Несмотря на указанные преимущества данного портала, он все же не лишен недостатков.

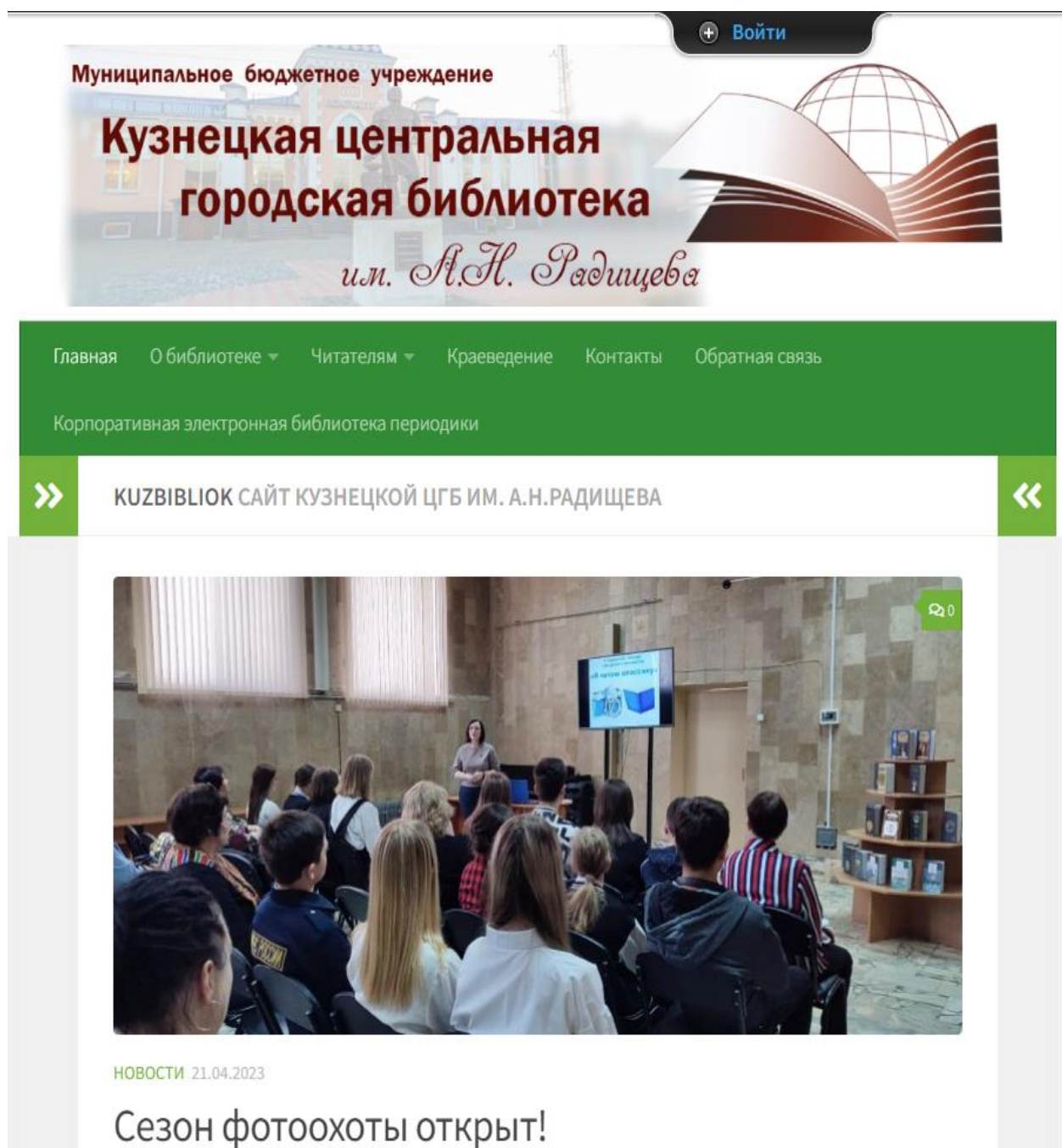


Рисунок 2 – Главная страница муниципального бюджетного учреждения
«Кузнецкая центральная городская библиотека им. А.Н. Радищева»

Основным из них можно считать то, что электронный каталог не позволяет получить электронные версии литературы, а лишь содержит информацию о наличии физических копий. Также неудобство поиска и отсутствие мобильного приложения может оттолкнуть многих читателей.

1.2.3 Муниципальное бюджетное учреждение «Карпогорская межпоселенческая библиотека» Пинежского муниципального района Архангельской области

Главная страница МБУ «Карпогорская библиотека» представлена на рисунке 3.

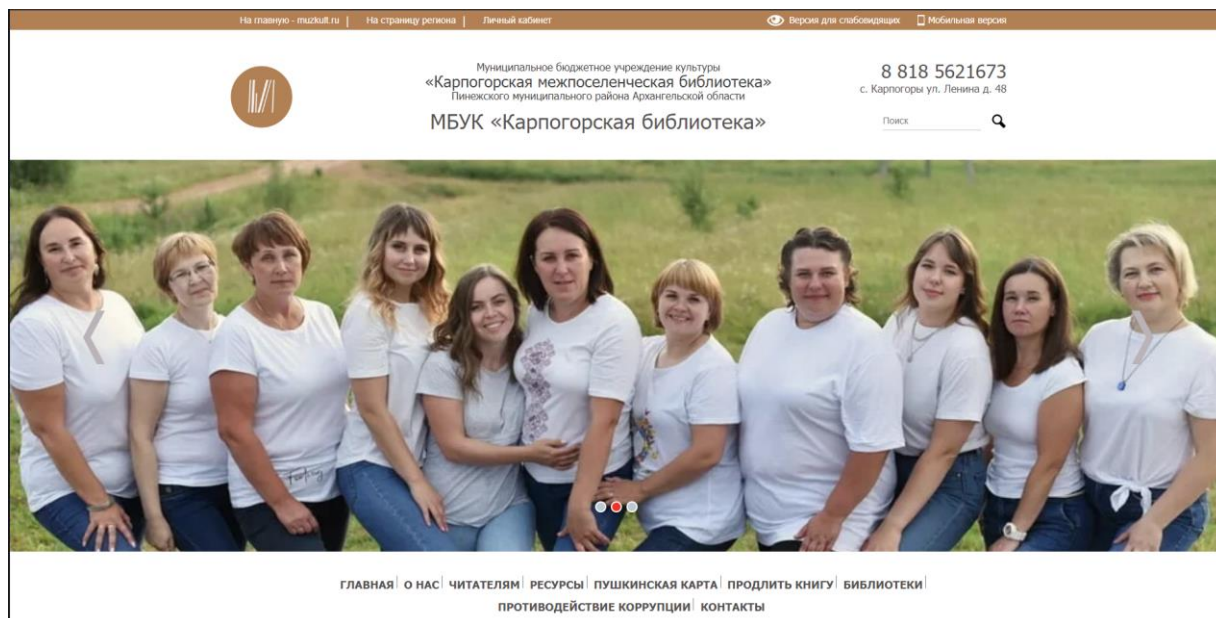


Рисунок 3 – Главная страница Карпогорской межпоселенческой библиотеки

Данный портал предоставляет своим пользователям доступ к достаточно обширному фонду литературы, позволяет просматривать электронные версии книг или получать физические копии в пункте выдачи. Кроме того, любой читатель может удобно использовать данный ресурс с мобильного устройства, так как данный портал имеет мобильную версию. Также осуществляется поддержка версии для слабовидящих [9].

Однако есть и недостатки. Основным из них является неудобство пользования самим порталом, так как многие функции работают нестабильно. В процессе работы было замечено периодическая неработоспособность функции просмотра электронной литературы онлайн, некорректность отображение страниц в мобильной версии сайта. Несмотря на наличие мобильной версии сайта, отсутствие полноценного мобильного приложения можно считать

существенным минусом. Помимо всего сказанного, стоит заметить неудобство системы поиска литературы в электронном каталоге.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше интернет-библиотек представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Интернет-библиотека	Наличие электронных экземпляров	Возможность скачивания литературы	Мобильное приложение	Система поиска
Пензенская областная библиотека имени М. Ю. Лермонтова	Нет	Нет	Отсутствует	Неудобная
МБУ «Кузнецкая центральная городская библиотека им. А.Н. Радищева»	Да, но перечень электронных экземпляров сильно ограничен	Нет	Отсутствует	Неудобная
МБУ «Карпогорская библиотека»	Да, но перечень электронных экземпляров сильно ограничен	Нет	Отсутствует, но есть мобильная версия сайта	Неудобная
Разрабатываемое приложение	Да	Да	Существует	Удобная

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше интернет-библиотек.

1.3 Анализ целевой аудитории

Целевой аудиторией проекта являются читатели муниципальных библиотек, которые предпочитают использовать мобильные средства для работы

с книгами, но столкнулись с ограниченностью функционирования электронных каталогов и отсутствием корректной поддержки мобильных устройств. Число пользователей библиотек, по данным главного информационно-вычислительного центра Министерства культуры РФ [10], представлено на рисунке 4.

	Население (тыс.чел.)	Число библиотек	Число жителей на одну библиотеку (тыс.чел.)		Библиотечный фонд (тыс.ед.)	Обеспеченность жителей фондами библиотек		Выдано (просмотрено) документов, (тыс.ед.)	Число пользователей тыс.чел.	% охвата населения библиотечным обслуживанием	
			показатель	место среди субъектов РФ		показатель	место среди субъектов РФ			показатель	место среди субъектов РФ
Всего по Российской Федерации	146 171,0	41 356	3,5		829 400,35	6		990 701,46	49 671,51	34,0	

Рисунок 4 – Данные о пользователях библиотек

В настоящее время число читателей, предпочитающих электронные экземпляры литературы бумажным, постоянно растет. Это связано в первую очередь с тем, что люди стремятся к удобству. Многие даже готовы платить деньги коммерческим организациям за удобное мобильное средство, с неограниченным функционально электронным каталогом произведений вместо того, чтобы получить искомую литературу в муниципальной библиотеке в физическом экземпляре или же электронном виде, в связи с отсутствием корректной поддержки мобильных устройств.

В качестве подтверждения роста числа читателей электронной литературы, приведем слова гендиректора Российской государственной библиотеки: «Я уверен, что очень многое во взаимодействии читателя и библиотеки должно быть переведено в цифровую среду. <...> Во время пандемии активное развитие получили электронные ресурсы «Ленинки». Это действительно удобно, и наш читатель уже привык ими пользоваться. <...> Конечно, мы идем в ногу со временем и делаем упор на цифровую трансформацию.». Значительно вырос среди россиян интерес и к онлайн-библиотекам. Как рассказали в пресс-службе книжного сервиса «Литрес», за пять лет количество выданных книг в проекте «Литрес: библиотека» увеличилось в 8,5 раза (866%), всего за этот период через проект было выдано более 9,5 млн книг [11].

По данным 2021 года объем российского рынка электронных и аудиокниг достиг 11 млрд рублей, увеличившись на 34% в сравнении с показателем годичной давности. Это больше темпов роста в 2020 году (+26%), свидетельствуют данные ГК «LitRes», которые обнародованы 29 декабря 2021 года [12]. В настоящий момент, рынок цифровой книги сохранил тенденцию развития и расширения, что также подтверждают слова гендиректора группы компаний «LitRes». По его словам, ожидается рост рынка цифровых книг в России на 20-30% в 2023 году, а емкость рынка составит более 13 миллиардов рублей [13].

Также, подтверждением популярности электронных экземпляров у читателей можно считать статистику посещений пользователями одной из крупнейших российских коммерческих электронных библиотек «LitRes» (рисунок 5) [14].

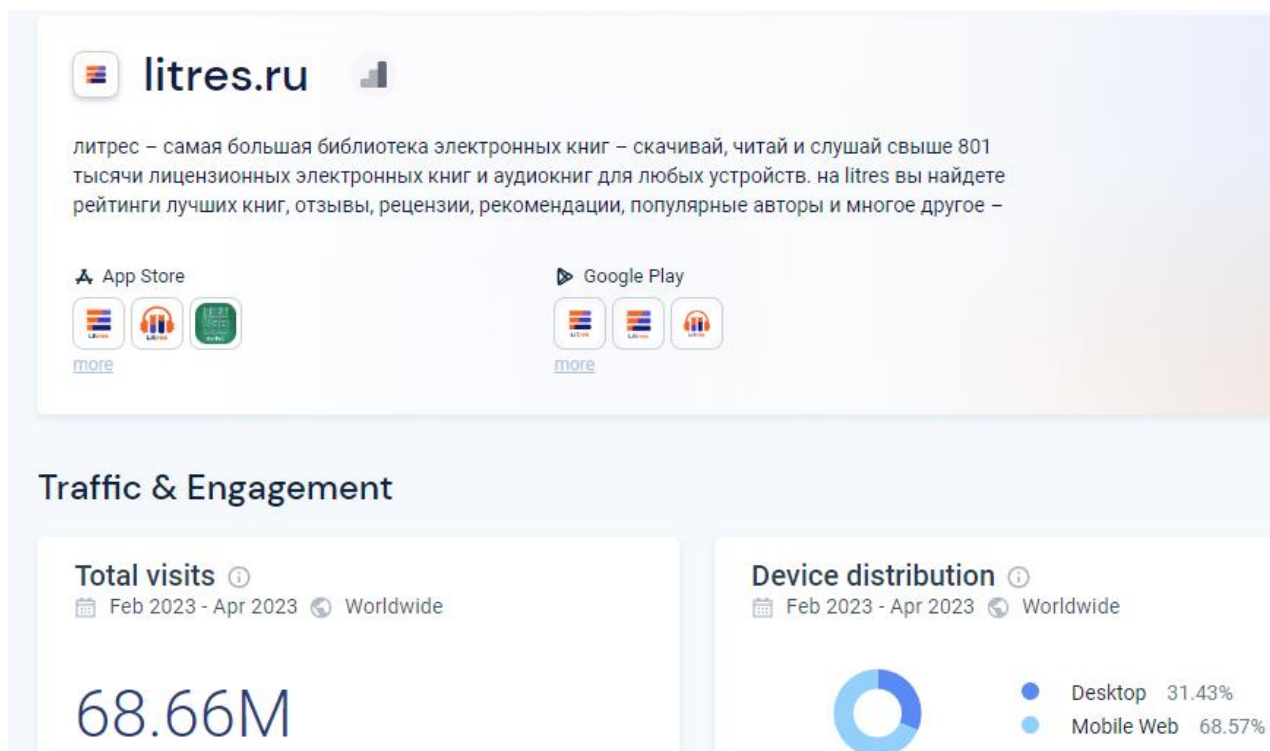


Рисунок 5 – Статистика посещения

Как можно увидеть, данная библиотека весьма популярна у читателей. Также, важно заметить, что практически 70% пользователей данного сайта используют мобильные средства. Помимо этого, данный сервис имеет

мобильное приложение, которое входит в топ 3 бесплатных средств в своей категории в России в каталоге «Google Play», а также обладает огромным количеством положительных отзывов и скачиваний.

На основании всего вышесказанного, можно с уверенностью заявлять о актуальности разрабатываемого приложения и наличия спроса у пользователей.

На рисунке 6 представлено распределение рынка смартфонов за период 2021 – 2022 года [15].

Vendor	2021 market share	2022 market share
Samsung	20%	22%
Apple	17%	19%
Xiaomi	14%	13%
OPPO	11%	9%
vivo	10%	9%
Others	28%	28%

Рисунок 6 – Доли производителей смартфонов на мировом рынке

Можно заметить, что практически все основные производители смартфонов – Samsung, Xiaomi, Huawei и т. д. – используют операционную систему Android. Единственное значимое исключение – компания Apple, использующая собственную систему iOS. Но, при всей популярности Apple, ее продукция даже в периоды пикового спроса составляет лишь в районе 20% от мирового рынка, что в несколько раз меньше, чем доля устройств с Android [16].

Поэтому наиболее выгодным решением с точки зрения доступности продукта для пользователей будет реализация разрабатываемого приложения именно для операционной системы Android.

Статистика распространения разных версий системы Android представлена на рисунке 7 [17].

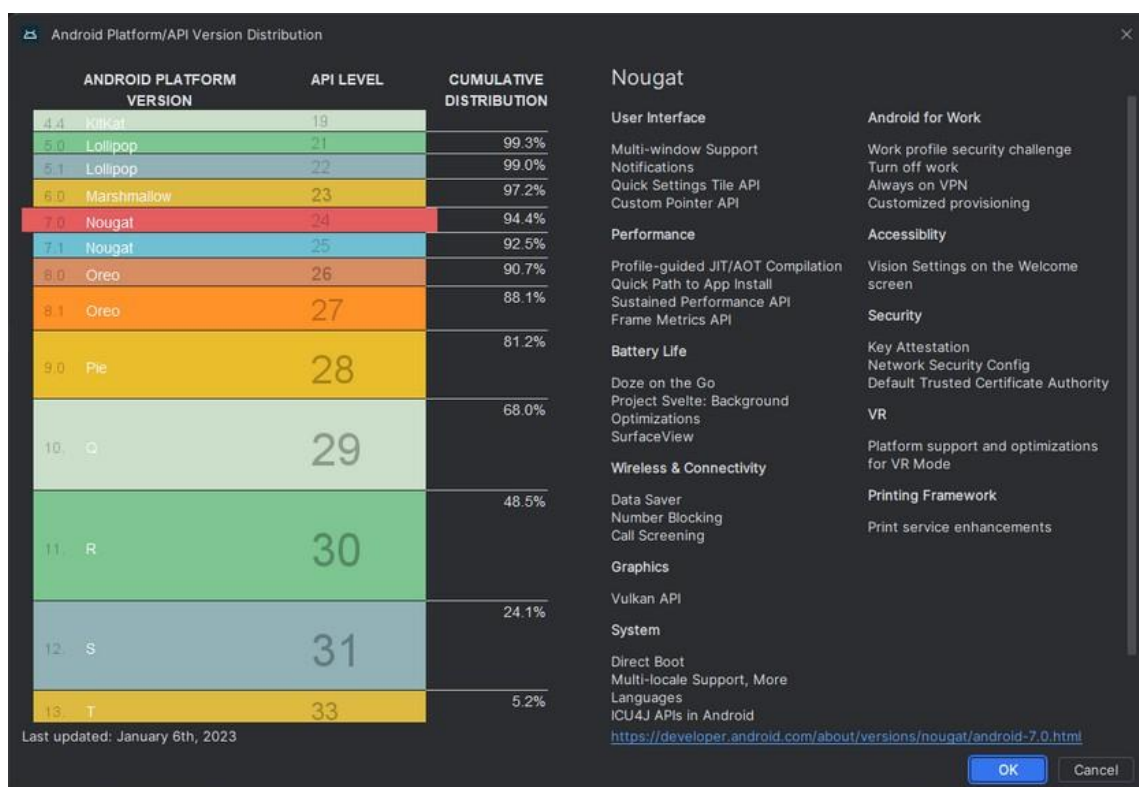


Рисунок 7 – Распространение различных версий системы Android

Можно увидеть, что 90% пользователей системы Android использует версии не старше 8.0. Поэтому наиболее оптимальным решением является разработка приложения таким образом, чтобы оно успешно функционировало на Android 8.0 и более новых версиях системы.

1.4 Анализ функциональных требований

На рисунке 8 отображена диаграмма вариантов использования, созданная в результате анализа требований к разрабатываемому средству.

Также составим спецификации вариантов использования «Зарегистрироваться», «Войти в систему», «Открыть перечень скаченных книг». Спецификация служит для описания последовательности действий пользователя и системы в рамках одного варианта использования, что позволяет однозначно и недвусмысленно понять требования к программному обеспечению, спроектировать и разработать программное средство без дополнительных уточнений и домыслов.

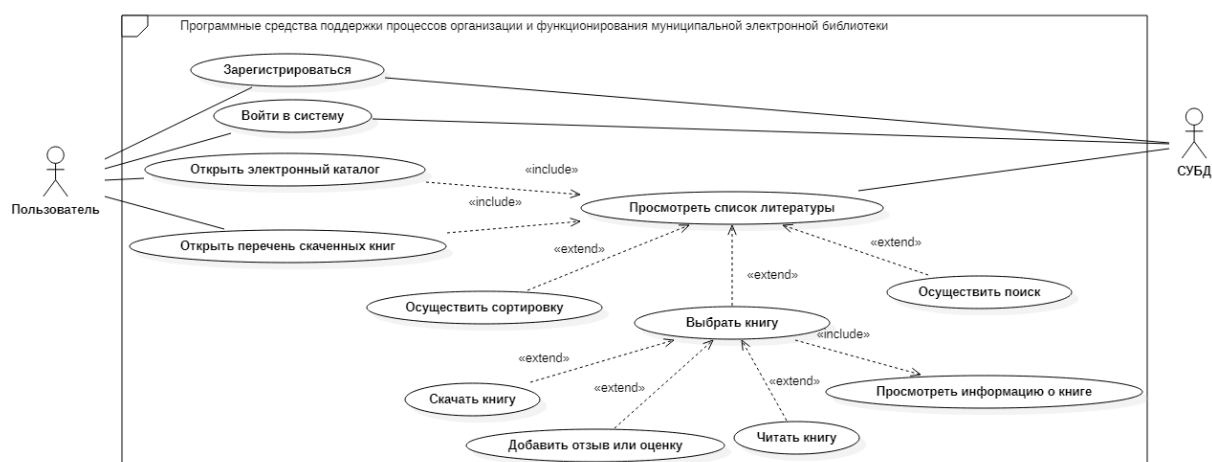


Рисунок 8 – Диаграмма вариантов использования мобильного приложения

Рассмотрим спецификации вариантов использования «Зарегистрироваться» (рисунок 9), «Войти в систему» (рисунок 10), «Открыть перечень скаченных книг» (рисунок 11).

Прецедент: зарегистрироваться
ID: 1
Краткое описание: Пользователь регистрируется в приложении
Главные актеры: Пользователь, СУБД
Предусловие: Пользователь ещё не зарегистрирован в системе
Основной поток: <ol style="list-style-type: none"> 1. Пользователь заполняет данные, необходимые для регистрации <ol style="list-style-type: none"> 1.1. Пользователь вводит адрес электронной почты 1.2. Пользователь вводит ФИО 1.3. Пользователь вводит данные паспорта 1.4. Пользователь задает логин и пароль 2. Пользователь посылает запрос системе на проведение регистрации по указанным данным 3. Система проверяет наличие и корректность введенных данных 4. Если данные введены корректно <ol style="list-style-type: none"> 4.1. Система запрашивает у СУБД сведения о наличии аккаунта с аналогичными паспортными данными, адресом электронной почты и логином 4.2. Если такой аккаунт существует <ol style="list-style-type: none"> 4.2.1. Система выводит сообщение о наличии данного пользователя в СУБД 4.2.2. Прерывает регистрацию пользователя 4.3. Иначе <ol style="list-style-type: none"> 4.3.1. Система проводит регистрацию пользователя 4.3.2. Сообщает пользователю о корректной регистрации
Постусловие: регистрация произведена

Рисунок 9 – Спецификация варианта использования «Зарегистрироваться»

Прецедент: войти в систему
ID: 2
Краткое описание: Пользователь авторизуется в системе
Главные актеры: Пользователь, СУБД
Предусловие: Пользователь
Основной поток: <ol style="list-style-type: none"> 1. Пользователь вводит логин и пароль 2. Система запрашивает у СУБД сведения о наличии аккаунта с указанным логином и паролем 3. СУБД возвращает сведения о наличии аккаунта с таким логином и паролем 4. Если такой аккаунт существует <ol style="list-style-type: none"> 4.1. Система проводит авторизацию пользователя 4.2. Система выводит сообщения о корректной авторизации 5. Иначе система сообщает о некорректности введенных данных
Постусловие: вход в систему осуществлен

Рисунок 10 – Спецификация варианта использования «Войти в систему»

Прецедент: открыть перечень скаченных книг
ID: 3
Краткое описание: Пользователь открывает перечень скаченных книг
Главные актеры: Пользователь, СУБД
Предусловие: Пользователь авторизован
Основной поток: <ol style="list-style-type: none"> 1. Пользователь выбирает список скаченной литературы 2. Система выводит перечень литературы, которую ранее скачал пользователь 3. Если скаченной литературы не обнаружено, система выводит соответствующее сообщение 4. Если пользователь хочет изменить критерий сортировки <ol style="list-style-type: none"> 4.1. Система получает критерий сортировки 4.2. Выводит отсортированный список по заданному критерию 5. Если пользователь выбирает конкретную книгу <ol style="list-style-type: none"> 5.1. Система запрашивает данные о выбранной литературе у СУБД 5.2. СУБД возвращает информацию о выбранной литературе 5.3. Система выводит краткую информацию о литературе 5.4. Система предоставляет возможность открыть литературу для прочтения 5.5. Система предоставляет возможность оставить отзыв и оценку литературе 6. Если пользователь хочет осуществить поиск <ol style="list-style-type: none"> 6.1. Система запрашивает критерий поиска 6.2. Пользователь вводит поисковый запрос 6.3. Система осуществляет поиск литературы
Постусловие: пользователь ознакомлен со списком скаченной им литературы

Рисунок 11 – Спецификация варианта использования «Открыть перечень скаченных книг»

Рассмотрим некоторые варианты использования подробнее. Для вариантов использования «Зарегистрироваться» и «Войти в систему» составлены диаграммы деятельности. Данные диаграммы представлены в приложении Б.

Для варианта использования «Открыть электронный каталог» была составлена диаграмма последовательности, представленная на рисунке 12.

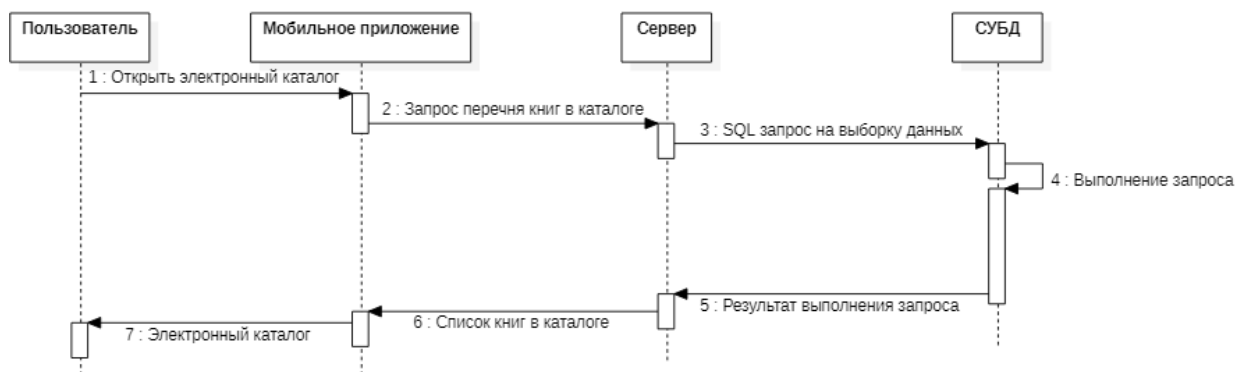


Рисунок 12 – Диаграмма последовательности варианта использования «Открыть электронный каталог»

В результате анализа функциональных требований были успешно выявлены основные функции разрабатываемого приложения. При этом для разрабатываемого приложения были построены диаграммы вариантов использования, деятельности и последовательности, а также для некоторых выявленных вариантов использования были составлены спецификации.

1.5 Анализ нефункциональных требований

Рассмотрев поставленную задачу, были сформулированы нефункциональные требования к разрабатываемому программному средству.

Должны быть использованы такие технологии разработки, как RUP, ООП, MVC.

Разработка серверной части приложения выполнена на языке программирования C#, в среде программирования Microsoft Visual Studio 2022. База данных, в которой будут храниться данные сервера, реализована с применением СУБД Microsoft SQL Server, запросы к базе данных

осуществляются на языке SQL. Серверная должны функционировать на операционной системе Windows 7 или выше.

Клиентская часть приложения выполнена на языке C#, в среде программирования Microsoft Visual Studio 2022. В процессе разработки также будет использован фреймворк для кроссплатформенной разработки мобильных приложений Xamarin. Клиентское приложение должно корректно функционировать на ОС Android 8.0 и выше.

1.6 Планирование разработки и оценка бюджета

Качественное выполнение работ по любому проекту, в том числе и в сфере информационных технологий, невозможно при отсутствии грамотного планирования, которое позволяет обеспечить выполнение работ над проектом таким образом, чтобы добиться выполнения всех поставленных целей за минимально возможное время, с минимально возможными затратами и с приемлемым качеством [18 – 20].

В процессе планирования разработки и оценки бюджета был выделен ряд задач, представляющих собой этапы работы над проектом, представленных в таблице 2.

Таблица 2 – Задачи проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области	7 дней	1
4	Анализ приложений-аналогов	4 дней	1
5	Анализ требований к функционалу	3 дней	3;4
6	Анализ требований к интерфейсу	3 дней	5
7	Постановка задачи	2 дня	6
8	Анализ требований завершен	-	7
9	Проектирование программных средств	-	-
10	Проектирование архитектуры приложения	3 дня	8
11	Проектирование базы данных	3 дня	8
12	Проектирование функционала	4 дня	10;11
13	Проектирование интерфейса пользователя	3 дня	10;11
14	Проектирование завершено	-	12;13
15	Реализация программных средств	-	-

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
16	Реализация базы данных	7 дней	10;11
17	Реализация серверной части приложения	14 дней	14;16
18	Реализация клиентской части приложения	12 дней	14;16
19	Реализация завершена	-	17;18
20	Тестирование	-	-
21	Тестирование работы серверной части приложения	4 дня	17
22	Тестирование работы клиентской части приложения	5 дней	18
23	Исправление ошибок	3 дня	19;21;22
24	Тестирование завершено	-	23
25	Внесение в систему электронных экземпляров литературы	3 дня	24
26	Оформление документации	7 дней	24
27	Конец проекта	-	25;26

Таким образом, для реализации проекта нужно выполнить 26 задач.

В таблице 3 приведено описание затраченных ресурсов.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист 1	Трудовой	300 руб/ч	550 руб/ч
Программист 2	Трудовой	300 руб/ч	550 руб/ч
Аналитик	Трудовой	250 руб/ч	500 руб/ч
Дизайнер	Трудовой	250 руб/ч	450 руб/ч
Компьютер	Материальный	350000 руб	-
Интернет	Затраты	-	-

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области	Аналитик Компьютер Интернет	40% 2 2000 руб
4	Анализ приложений-аналогов	Аналитик	60%
5	Анализ требований к функционалу	Аналитик Программист 1	100% 100%

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
6	Анализ требований к интерфейсу	Аналитик Дизайнер	100% 100%
7	Постановка задачи	Аналитик Дизайнер Программист 1 Программист 2	100% 25% 25% 25%
8	Анализ требований завершен	-	-
9	Проектирование программных средств	-	-
10	Проектирование архитектуры приложения	Аналитик Программист 1	50% 50%
11	Проектирование базы данных	Аналитик Программист 1	50% 50%
12	Проектирование функционала	Аналитик Программист 2	50% 50%
13	Проектирование интерфейса пользователя	Аналитик Дизайнер Программист 2	50% 100% 50%
14	Проектирование завершено	-	-
15	Реализация программных средств	-	-
16	Реализация базы данных	Программист 1	100%
17	Реализация серверной части приложения	Программист 2	100%
18	Реализация клиентской части приложения	Программист 1 Дизайнер	100% 100%
19	Реализация завершена	-	-
20	Тестирование	-	-
21	Тестирование работы серверной части приложения	Программист 2	50%
22	Тестирование работы клиентской части приложения	Программист 1	50%
23	Исправление ошибок	Программист 1 Программист 2	50% 50%
24	Тестирование завершено	-	-
25	Внесение в систему электронных экземпляров литературы	Программист 1 Программист 2	30% 30%
26	Оформление документации	Программист 1 Программист 2	70% 70%
27	Конец проекта	-	-

Таким образом, реализацией проекта будут заниматься программист 1, программист 2, аналитик и дизайнер, которым для работы нужно два компьютера и доступ в интернет. Для вычисления длительности реализации проекта и его критического пути построен сетевой график (рисунок 13).

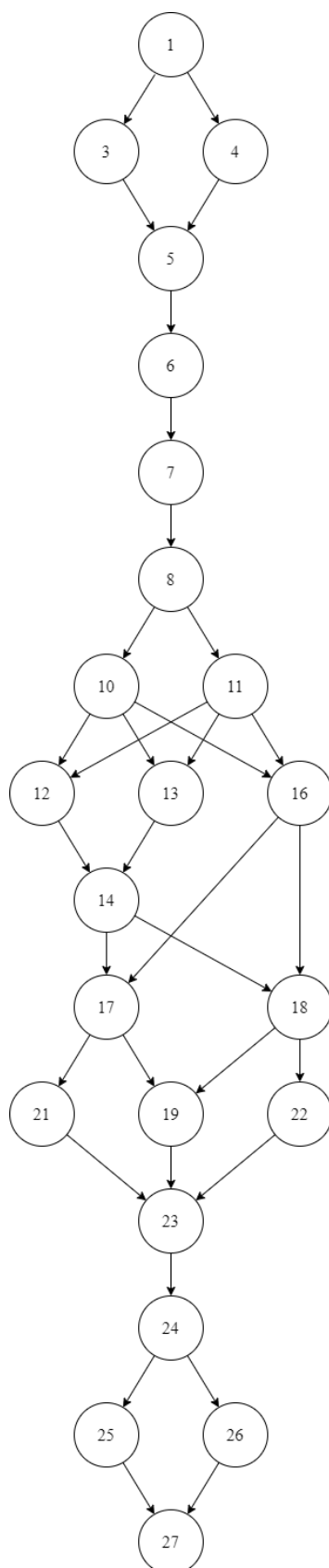


Рисунок 13 – Сетевой график проекта

Рассчитаем вручную критический путь проекта. В таблице 5 представлены расчеты раннего времени начала работ для его задач.

Таблица 5 – Расчеты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	1	0	0	0+0	0
5	3	0	7	Max(0+7;0+4)	7
	4	0	4		
6	5	7	3	7+3	10
7	6	10	2	10+2	12
8	7	12	2	12+2	14
10	8	14	0	14+0	14
11	8	14	0	14+0	14
12	10	14	3	14+3	17
	11				
13	10	14	3	14+3	17
	11				
14	12	17	4	Max(17+4;17+3)	21
	13	17	3		
16	10	14	3	14+3	17
	11				
17	14	21	0	Max(21+0;17+7)	24
	16	17	7		
18	14	21	0	Max(21+0;17+7)	24
	16	17	7		
19	17	24	14	Max(24+14;24+12)	38
	18		12		
21	17	24	14	24+14	38
22	18	24	12	24+12	36
23	19	38	0	Max(38+0;38+4;36+5)	42
	21	38	4		
	22	36	5		
24	23	42	3	42+3	45
25	24	45	0	45+0	45
26	24	45	0	45+0	45
27	25	45	3	Max(45+3;45+7)	52
	26	45	7		

В таблице 6 рассчитано позднее время начала работ задач.

Таблица 6 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
27	-	-	-	-	52
26	27	52	7	52-7	45
25	27	52	3	52-3	49
24	26 25	45 49	0	Min(45-0;49-0)	45
23	24	45	3	45-3	42
22	23	42	5	42-5	37
21	23	42	4	42-4	38
19	23	42	0	42-0	42
18	22 19	37 42	12	Min(37-12;42-12)	25
17	21 19	38 42	14	Min(38-14;42-14)	24
16	18 17	25 24	7	Min(25-7;24-7)	17
14	18 17	25 24	0	Min(25-0;24-0)	24
13	14	24	3	24-3	21
12	14	24	4	24-4	20
11	16 13 12	17 21 20	3	Min(17-3;21-3;20-3)	14
10	16 13 12	17 21 20	3	Min(17-3;21-3;20-3)	14
8	11 10	14	0	14-0	14
7	8	14	2	14-2	12
6	7	12	2	12-2	10
5	6	10	3	10-3	7
4	5	7	4	7-4	3
3	5	7	7	7-7	0
1	4 3	3 0	0	Min(3-0;0-0)	0

На основе найденных значений раннего и позднего времени начала работ можно найти критический путь. Соответствующие вычисления представлены в таблице 7.

Таблица 7 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	0	3	3
5	7	7	0
6	10	10	0
7	12	12	0
8	14	14	0
10	14	14	0
11	14	14	0
12	17	20	3
13	17	21	4
14	21	24	3
16	17	17	0
17	24	24	0
18	24	25	1
19	38	42	4
21	38	38	0
22	36	37	1
23	42	42	0
24	45	45	0
25	45	49	4
26	45	45	0
27	52	52	0

На основе полученных можно сделать вывод, что критический путь проходит через работы с номерами 1, 3, 5, 6, 7, 8, 10, 11, 16, 17, 21, 23, 24, 26, 27, а общее время выполнения проекта составляет 52 дня. На рисунке 14 представлен граф проекта, на котором пунктирными линиями указан критический путь.

Чтобы проверить корректность проведенных расчетов продолжительности проекта, вычислить необходимый бюджет, воспользуемся системой управления проектами Microsoft Project. Данная система позволяет автоматизировать все необходимые вычисления [21, 22]. Внесенные в систему Microsoft Project задачи из таблицы 2, представлен на рисунке 15. Временем начало проекта принята следующая дата: 10.01.2023.

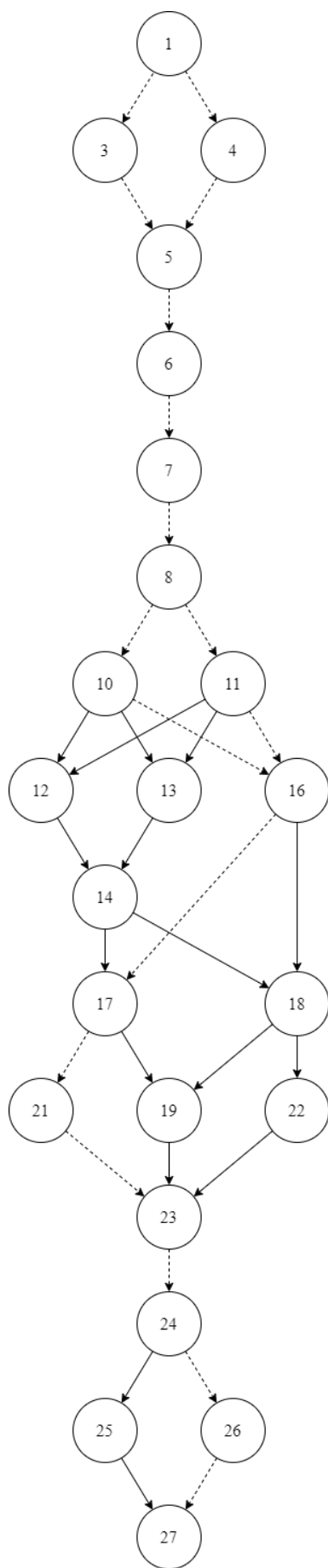


Рисунок 14 – Критический путь проекта

№	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Ср 12.04.23	Ср 12.04.23		
2	Анализ требований	15 дней	Ср 12.04.23	Вт 02.05.23		
3	Анализ предметной области	7 дней	Ср 12.04.23	Чт 20.04.23	1	Аналитик[40%];Компьютер[1 шт.];Интернет
4	Анализ приложений-аналогов	4 дней	Ср 12.04.23	Пн 17.04.23	1	Аналитик[60%]
5	Анализ требований к функциям	3 дней	Пт 21.04.23	Вт 25.04.23	3,4	Аналитик;Программист1
6	Анализ требований к интерфейсу	3 дней	Ср 26.04.23	Пт 28.04.23	5	Аналитик;Дизайнер
7	Постановка задачи	2 дней	Пн 01.05.23	Вт 02.05.23	6	Аналитик;Дизайнер[25%];Программист
8	Анализ требований завершено	0 дней	Вт 02.05.23	Вт 02.05.23	7	
9	Проектирование программных средств	7 дней	Ср 03.05.23	Чт 11.05.23		
10	Проектирование архитектуры	3 дней	Ср 03.05.23	Пт 05.05.23	8	Аналитик[50%];Программист1[50%]
11	Проектирование базы данных	3 дней	Ср 03.05.23	Пт 05.05.23	8	Аналитик[50%];Программист1[50%]
12	Проектирование функционала	4 дней	Пн 08.05.23	Чт 11.05.23	10,11	Аналитик[50%];Программист2[50%]
13	Проектирование интерфейса	3 дней	Пн 08.05.23	Ср 10.05.23	10,11	Аналитик[50%];Дизайнер;Программист
14	Проектирование завершено	0 дней	Чт 11.05.23	Чт 11.05.23	13,12	
15	Реализация программных средств	21 день	Пн 08.05.23	Пн 05.06.23		
16	Реализация базы данных	7 дней	Пн 08.05.23	Вт 16.05.23	10,11	Программист1
17	Реализация серверной части	14 дней	Ср 17.05.23	Пн 05.06.23	14,16	Программист2
18	Реализация клиентской части	12 дней	Ср 17.05.23	Чт 01.06.23	14,16	Программист1;Дизайнер
19	Реализация завершена	0 дней	Пн 05.06.23	Пн 05.06.23	17,18	
20	Тестирование	9 дней	Пт 02.06.23	Ср 14.06.23		
21	Тестирование работы сервера	4 дней	Вт 06.06.23	Пт 09.06.23	17	Программист2
22	Тестирование работы клиента	5 дней	Пт 02.06.23	Чт 08.06.23	18	Программист1
23	Исправление ошибок	3 дней	Пн 12.06.23	Ср 14.06.23	19,21,22	Программист1[50%];Программист2[50%]
24	Тестирование завершено	0 дней	Ср 14.06.23	Ср 14.06.23	23	
25	Внесение в систему электронных документов	3 дней	Чт 15.06.23	Пн 19.06.23	24	Программист1[30%];Программист2[30%]
26	Оформление документации	7 дней	Чт 15.06.23	Пт 23.06.23	24	Программист1[70%];Программист2[70%]
27	Конец проекта	0 дней	Пт 23.06.23	Пт 23.06.23	25,26	

Рисунок 15 – Задачи в Microsoft Project

На основе введенных данных, в среде Microsoft Project была построена диаграмма Ганта, на основании которой, можно считать, что критический путь проходит через задачи: 1, 3, 5, 6, 7, 8, 10, 11, 16, 17, 21, 23, 24, 26, 27. Можно заметить, что вычисленный программой критический путь совпал с критическим путем, вычисленным вручную. Диаграмма Ганта представлена на рисунке 16.

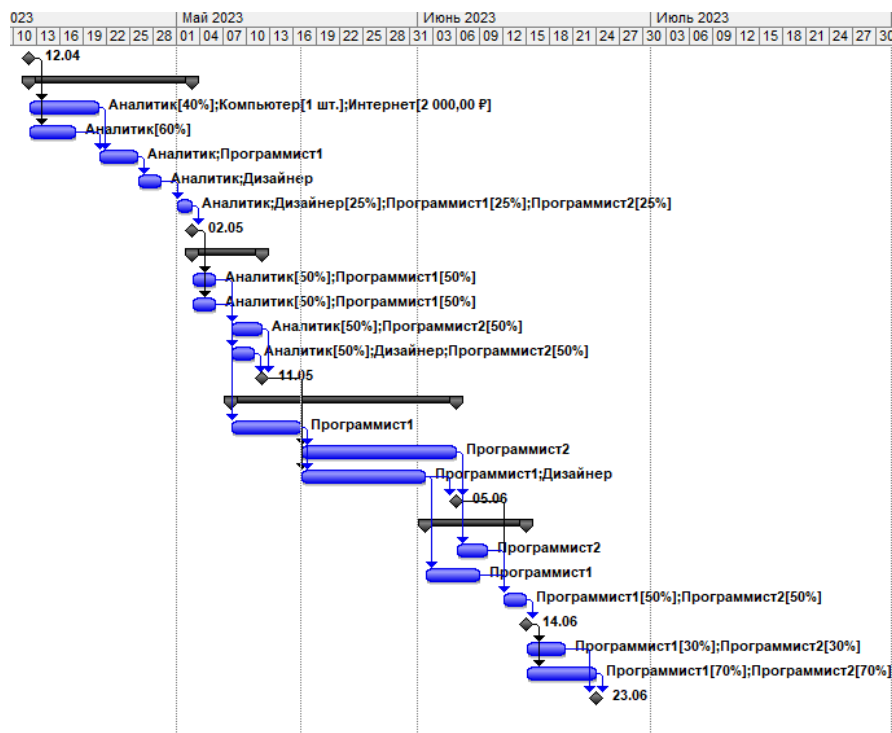


Рисунок 16 – Диаграмма Ганта

На рисунках 17 – 20 представлены графики трудовых ресурсов «Программист1», «Программист2», «Аналитик», «Дизайнер» соответственно.

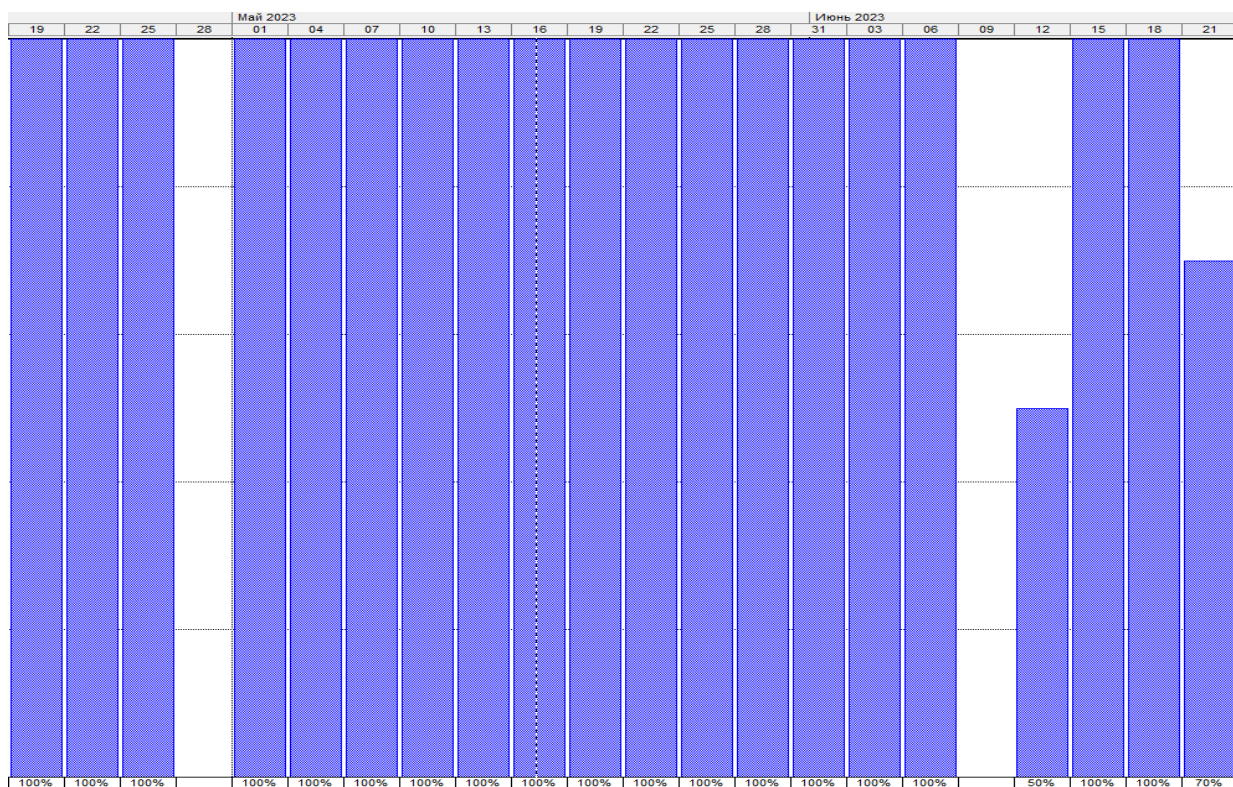


Рисунок 17 – График ресурсов для «Программист1»

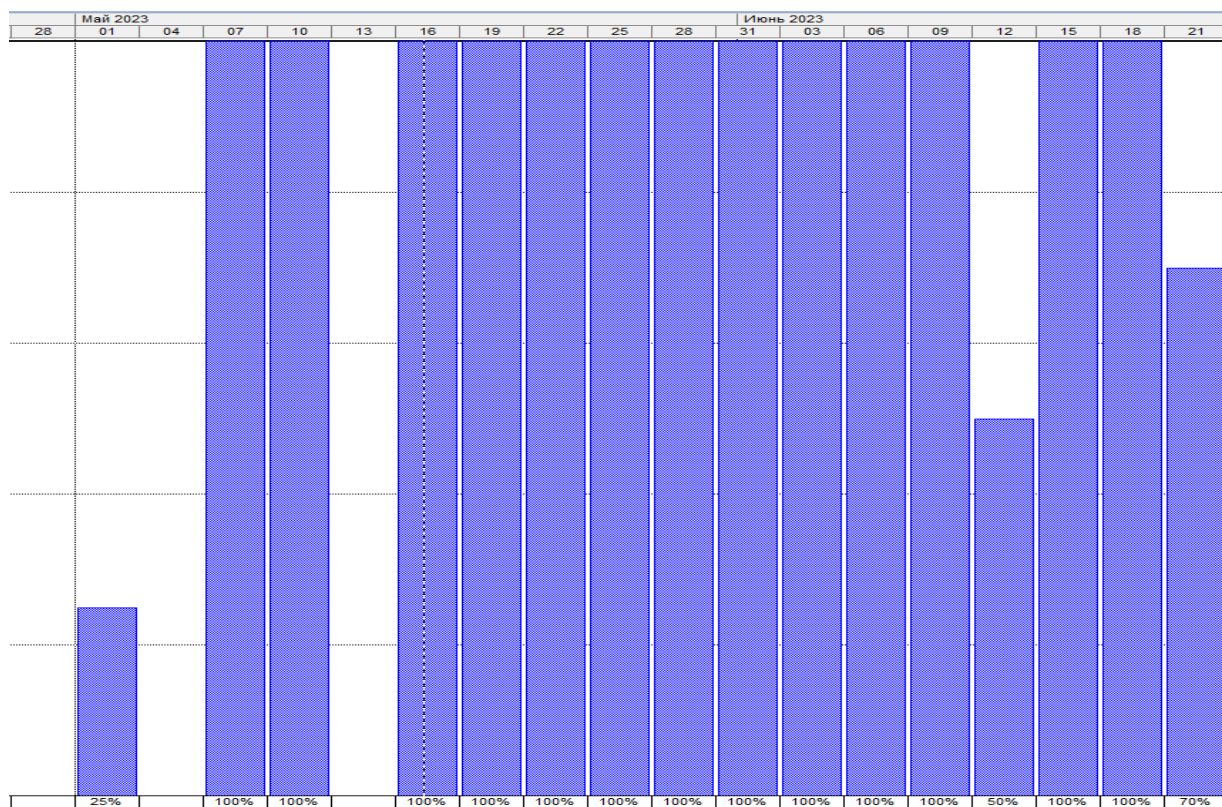


Рисунок 18 – График ресурсов для «Программист2»

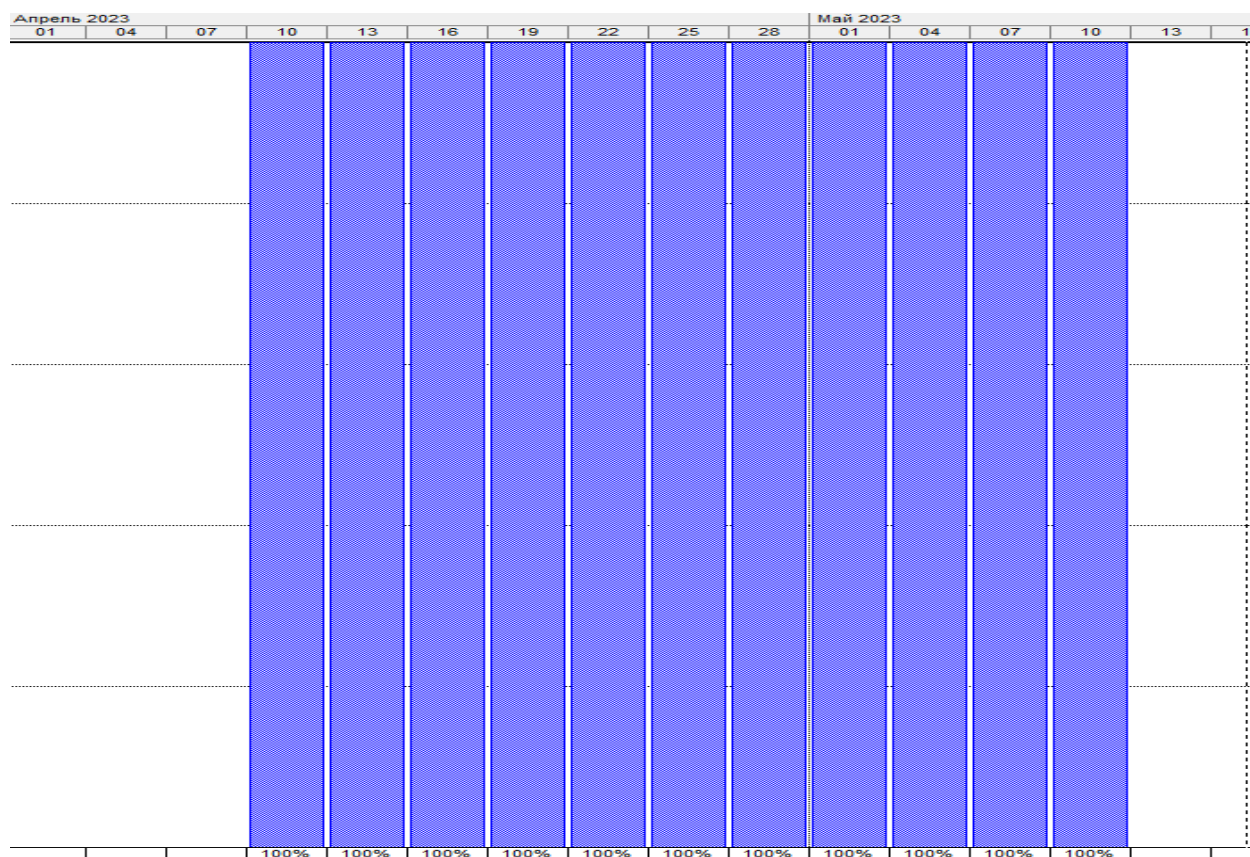


Рисунок 19 – График ресурсов для «Аналитик»

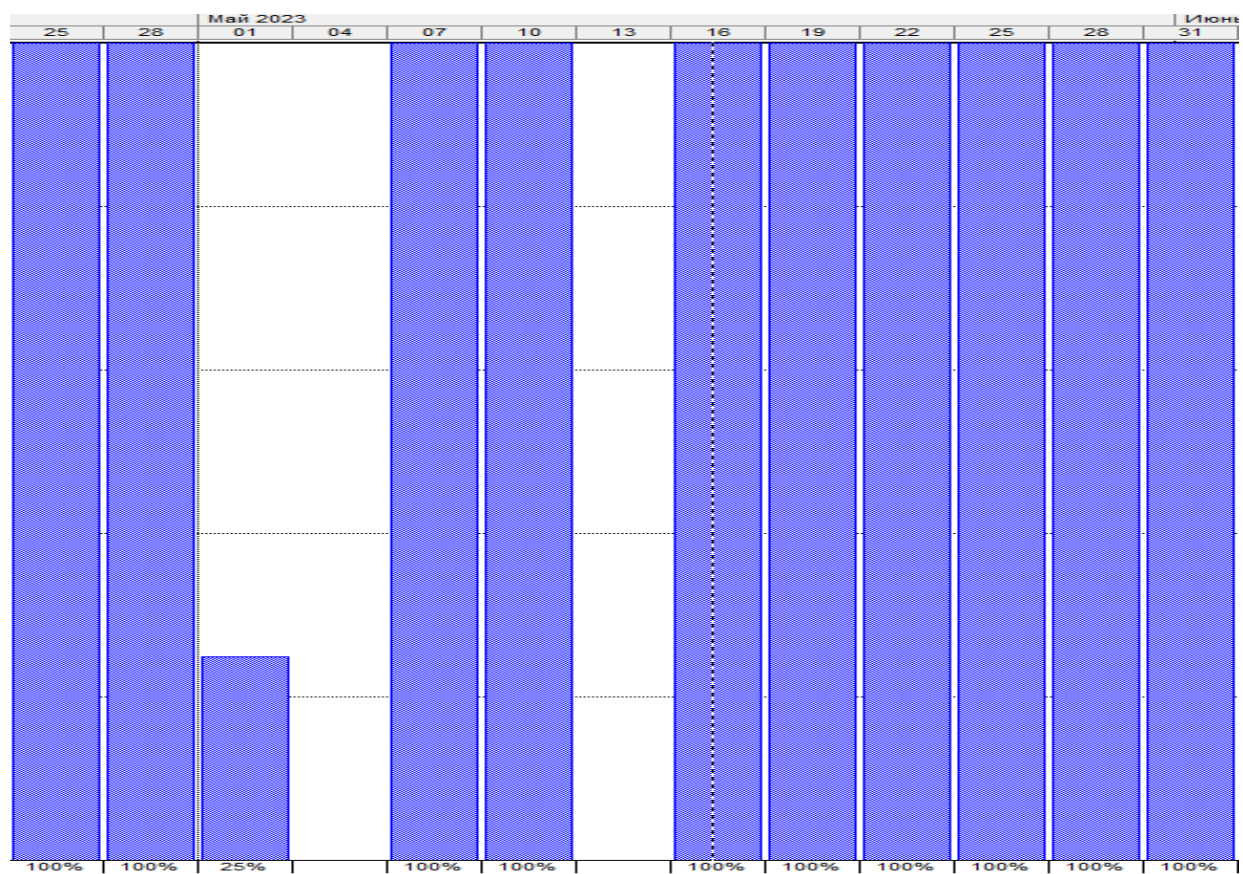


Рисунок 20 – График ресурсов для «Дизайнер»

На рисунке 21 представлен обзорный отчет проекта. Реализация займет 52 дня, а предполагаемые затраты составят 272040 рублей.

<u>Даты</u>			
Начало:	Ср 12.04.23	Окончание:	Чт 22.06.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
<u>Длительность</u>			
Текущий план:	52 дней	Оставш ася:	52 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	52 дней	Процент завершения:	0%
<u>Трудозатраты</u>			
Текущий план:	834,4 ч	Оставш иеся:	834,4 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	834,4 ч	Процент завершения:	0%
<u>Затраты</u>			
Текущий план:	272 040,00 Р	Оставш иеся:	272 040,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	272 040,00 Р		
<u>Состояние задачи</u>		<u>Состояние ресурса</u>	
Еще не начатые задачи	27	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	0
Завершенные задачи:	0	Материальные ресурсы:	1
Итого задач:	27	Итого ресурсов:	6

Рисунок 21 – Обзорный отчет проекта

На основании произведенных оценок сроков выполнения проекта и стоимости его реализации можно сделать вывод о его экономической целесообразности.

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств разработки

В процессе проектирования программного средства было решено применять одну из самых популярных методологий разработки Rational Unified Process (RUP). RUP – это методология разработки программного обеспечения, основанная на итеративной и инкрементальной модели. Она предлагает подход к разработке, который предполагает, что вся разработка делится на четыре основных этапа, каждый из которых включает в себя одну или несколько итераций [23].

К этапам разработки, по методологии RUP, относят:

- начальная стадия – формируется общее видение проекта и экономическое обоснование;
- стадия уточнения – анализ предметной области и построение архитектуры продукта;
- стадия построения – реализация основного функционала продукта;
- стадия внедрения – получение финальной версии продукта и его передача заказчику.

В каждой из этих фаз используются определенные методы и инструменты, которые помогают команде проекта создавать высококачественное ПО. Каждая фаза завершается вехой, на которой команда разработчиков оценивает выполнение поставленных целей и принимает критически важные решения о ходе дальнейшей работы над продуктом. RUP также включает в себя набор лучших практик, которые помогают снизить риски и повысить эффективность процесса разработки.

Методология разработки RUP была выбрана при разработке программного средства для поддержки процессов организации и функционирования муниципальной электронной библиотеки не случайно, а после рассмотрения основных преимуществ: гибкость процесса разработки, возможность быстрого внесения поправок в проект, возможность выявления основных рисков

разработки программного средства уже на ранних стадиях, данный метод разработки не требует постоянного взаимодействия с заказчиком. На основании указанных ранее особенностей, можно сделать вывод, что в условиях разработки данного программного средства, методология разработки ПО RUP является наиболее оптимальной и эффективной.

В соответствии с принципами RUP, для анализа требований и моделирования структуры проекта будет применяться язык моделирования UML. Язык UML – это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем [24].

Серверная и клиентская части приложения будут реализованы на языке программирования C#, в среде программирования Microsoft Visual Studio 2022. База данных, в которой будут храниться данные сервера, реализована с применением СУБД Microsoft SQL Server, запросы к базе данных осуществляются на языке SQL. В процессе разработки также будет использован фреймворк для кроссплатформенной разработки мобильных приложений Xamarin.

2.2 Проектирование архитектуры программного обеспечения

Для данной разработки было решено использовать в приложении трехуровневую архитектуру «клиент-сервер», рисунок 22.

В трёхуровневой архитектуре сервер баз данных, файловый сервер и другие представляют собой отдельный уровень, результаты работы которого использует сервер приложений. Логика данных и бизнес-логика находятся в сервере приложений. Все обращения клиентов к базе данных происходят через промежуточное программное обеспечение (middleware), которое находится на сервере приложений. Вследствие этого, повышается гибкость работы и производительность [25].

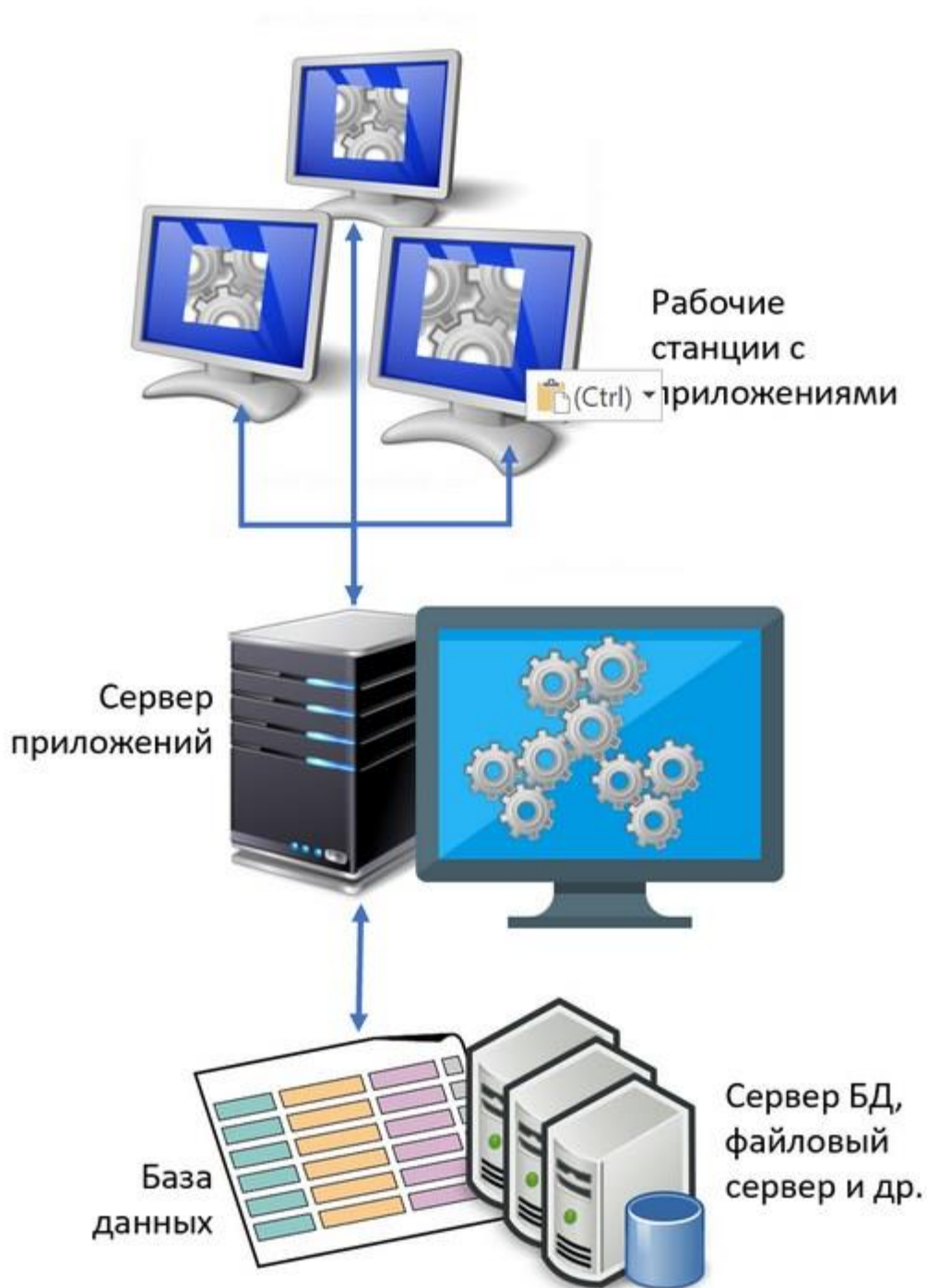


Рисунок 22 – Трёхуровневая архитектура «клиент-сервер»

Преимущества трёхуровневой архитектуры:

- целостность данных;
- более высокая безопасность, по сравнению с двухуровневой архитектурой;
- защищённость базы данных от несанкционированного проникновения.

Однако данная архитектура имеет недостаток, а именно более сложную структуру коммуникаций между клиентами и сервером.

2.3 Проектирование интерфейса пользователя

В современном мире пользовательский интерфейс является одним из ключевых аспектов разработки программного обеспечения. Именно он определяет удобство использования приложения, его функциональность и эстетический вид. Поэтому проектирование интерфейса пользователя является важным этапом разработки любого программного продукта, включая веб-сайты, мобильные приложения и десктопные приложения. В данной главе, на основе выявленных функциональных требований, будет спроектирован интерфейс страниц разрабатываемого мобильного приложения.

Для обеспечения всех заявленных функций приложения, необходимо разработать интерфейс страницы авторизации, регистрации, списка скаченных книг, электронного каталога, страница данных и отзывов о книге, поле для чтения книги.

Приведем модели интерфейса некоторых страниц приложения.

Модель страницы авторизации пользователя в систему представлена на рисунке 23.

На данной странице будут размещены следующие элементы:

- 1 – поле ввода логина;
- 2 – поле ввода пароля;
- 3 – кнопка «Вход»;
- 4 – кнопка перехода на страницу регистрации.

При нажатии на кнопку «Вход» система проверяет корректность введенных данных, а также наличие зарегистрированного пользователя в системе с указанными логином и паролем. Если такого пользователя не обнаружено или введенные данные некорректны, выводится сообщение о некорректности логина или пароля. В противном случае происходит авторизация пользователя и переход на главную страницу.

The diagram shows a login page model within a rectangular frame. It contains the following elements from top to bottom: a label 'Логин:' followed by a text input field (pointed to by arrow 1), a label 'Пароль:' followed by a text input field (pointed to by arrow 2), a 'Вход' button (pointed to by arrow 3), and a section titled 'Еще не зарегистрированы:' containing a 'Зарегистрироваться' button (pointed to by arrow 4).

Рисунок 23 – Модель страницы входа в систему

Страница регистрации пользователей представлена на рисунке 24.

The diagram shows a user registration page model within a rectangular frame. It contains the following elements from top to bottom: a label 'Введите логин:' followed by a text input field (pointed to by arrow 1), a label 'Введите адрес электронной почты:' followed by a text input field (pointed to by arrow 2), a label 'Введите ФИО:' followed by a text input field (pointed to by arrow 3), a label 'Введите пароль:' followed by a text input field (pointed to by arrow 4), and a 'Зарегистрироваться' button (pointed to by arrow 5).

Рисунок 24 – Модель страницы регистрации пользователя

На данной странице будут размещены следующие элементы:

1 – поле ввода логина;

2 – адреса электронной почты;

3 – поле ввода ФИО;

4 – поле ввода пароля;

5 – кнопка «Зарегистрироваться», по нажатию на которую происходит регистрация нового пользователя в системе по указанным данным.

В случае, если указанные данные не верны или уже зарегистрированы в системе, выводится соответствующее сообщение.

Главная страница приложения представлена на рисунке 25.



Рисунок 25 – Модель главной страницы приложения

На данной странице будут размещены следующие элементы:

1 – кнопка открытия электронного каталога, по нажатию на которую происходит переход на страницу с электронным каталогом;

2 – кнопка открытия списка скаченной литературы, по нажатию на которую происходит переход на страницу с списком скаченной литературы пользователем;

3 – кнопка выхода из профиля, по нажатию на которую происходит выход пользователя из системы и переход на страницу авторизации.

Страницы электронного каталога и скаченной литературы аналогичны с точки зрения интерфейса пользователя, однако ссылаются на разные части базы данных, интерфейс представлен на рисунке 26.

The diagram illustrates the layout of a literature list output page. It consists of several sections:

- Search Section:** At the top, there is a label "Введите поисковый запрос:" followed by a text input field (callout 1) and a button labeled "Искать" (callout 2).
- Search Criteria Section:** Below the search section is a label "Выберите критерий поиска:" followed by a dropdown menu (callout 3).
- Sorting Section:** Below the search criteria section is a label "Выберите критерий сортировки:" followed by another dropdown menu (callout 4).
- Search Results Section:** This section contains three rows, each representing a search result. Each row has a label "Название книги - Автор" and a button labeled "Открыть" (callout 5).
- Footer Section:** At the bottom of the page is a button labeled "Возврат на главную" (callout 6).

Рисунок 26 – Модель страницы вывода списка литературы

На данной странице будут размещены следующие элементы:

- 1 – поле ввода поискового запроса;
- 2 – кнопка «Искать», по нажатию на которую происходит поиск литературы в базе данных и переформирование списка литературы на странице;
- 3 – выпадающее окно выбора критерия поиска;
- 4 – выпадающее окно выбора критерия сортировки;
- 5 – кнопка перехода на страницу с подробной информацией о литературе;
- 6 – кнопка возврата на главную страницу приложения.

Спроектированные модели интерфейса являются предварительными и могут быть изменены в процессе реализации программного средства.

2.4 Проектирование базы данных

В современном мире, где огромное количество информации обрабатывается каждую секунду, базы данных играют важную роль в хранении и управлении этой информацией. Поэтому проектирование базы данных – это важный этап любого проекта.

На основе анализа предметной области и функциональных требований была построена концептуальная модель базы данных в виде диаграммы «сущность-связь», представленной на рисунке 27.

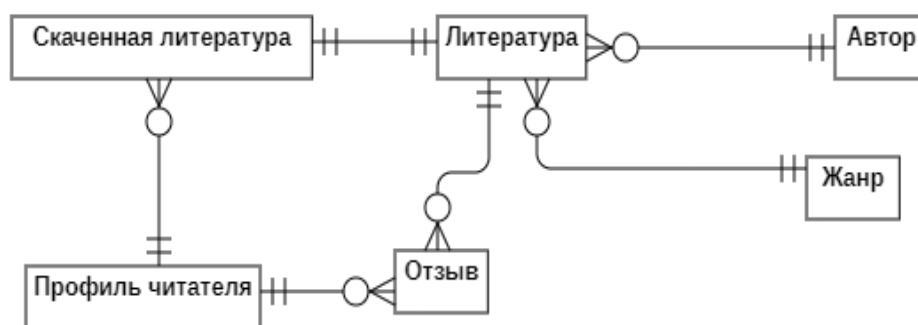


Рисунок 27 – Диаграмма «сущность-связь»

Сущности концептуальной модели данных представлены в таблице 8.

Таблица 8 – Сущности предметной области

№	Наименование	Описание	Количество экземпляров	Режим обновления
1	Профиль читателя	Список профилей читателей	10000	В систему добавляются новые профили читателей при регистрации
2	Скаченная литература	Список скаченной литературы пользователем	100	Обновляется после того, как пользователь скачивает литературу на свое устройство
3	Литература	Список всей литературы системы	10000	Обновляется администрацией библиотек
4	Автор	Список авторов литературы	1000	Обновляется администрацией библиотек по мере необходимости

Продолжение таблицы 8

№	Наименование	Описание	Количество экземпляров	Режим обновления
5	Жанр	Список жанров литературы	100	Обновляется администрацией библиотек по мере необходимости
6	Отзыв	Список отзывов о литературе	100	Обновляется по мере добавления отзывов пользователями

Также в таблице 9 представлено описание связей между сущностями.

Таблица 9 – Описание связей между сущностями

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Профиль читателя	Скаченная литература	Каждый пользователь может производить множество загрузок / Каждая загрузка производится одним пользователем	1:M	Необязательная (пользователь может не иметь загруженных книг)	При повторной загрузке книги пользователем новая запись не создается
2	Профиль читателя	Отзыв	Каждый пользователь может оставлять множество отзывов / Каждый отзыв оставляется одним пользователем	1:M	Необязательная (Пользователь может не иметь оставленных отзывов о литературе)	При повторной попытке оставить отзыв к одной и той же книге происходит редактирование старого отзыва
3	Скаченная литература	Литература	Каждой скаченной пользователем литературе соотносится литература из электронного каталога	1:1	Обязательная	
4	Литература	Отзыв	К каждой литературе может быть написано множество отзывов / Каждый отзыв может относиться только к одной литературе	1:M	Необязательная (у литературы может и не быть отзывов)	
5	Литература	Автор	Каждая литература может иметь одного автора / У каждого автора может быть множество написанных им произведений	M:1	Обязательная	
6	Литература	Жанр	Каждая литература может иметь один жанр / К каждому жанру может относиться множество произведений	M:1	Обязательная	

На основании концептуальной модели данных была смоделирована логическая модель базы данных средствами Visual Paradigm. Её структура описана диаграммой логической модели, представленной на рисунке 28.

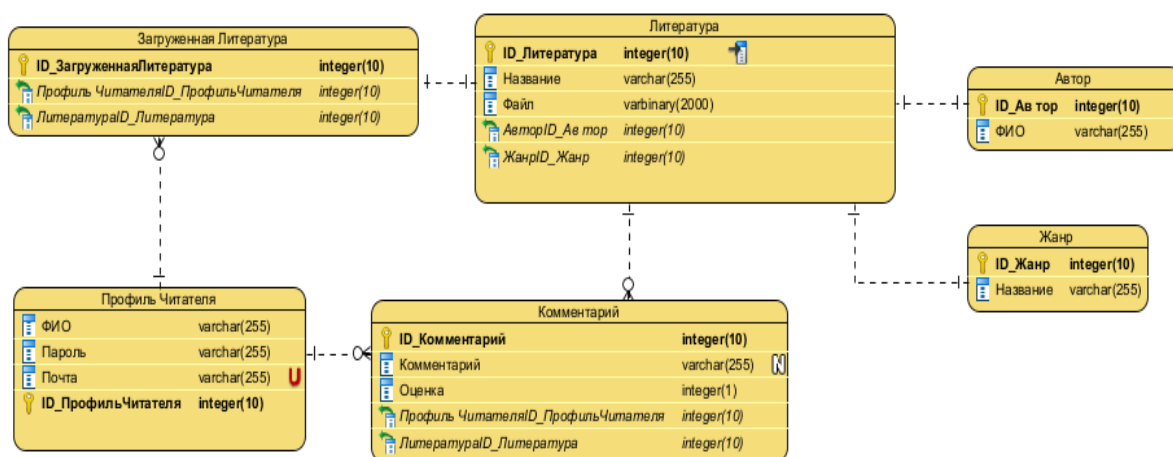


Рисунок 28 – Логическая модель базы данных

На представленной логической модели спроектированная структура БД отображена уже более подробно. Здесь не только присутствуют типы сущностей и связи между ними, но и хранится информация о конкретных атрибутах сущностей. Однако на данный момент сущности все еще представлены в абстрактной форме, без привязки к конкретной реализации базы данных.

2.5 Проектирование серверной части приложения

Для реализации серверной части приложения было решено использовать архитектуру REST API. Данная архитектура используется для доступа клиентских приложений к данным, хранящимся на сервере, для этого используются такие методы как: POST, GET, PUT и DELETE. При этом обмен сообщениями осуществляется обычно по протоколу HTTP.

Ключевой особенностью REST API является то, что обмен сообщениями происходит без хранения состояния. Каждое сообщение является самодостаточным и содержит всю информацию, необходимую для его обработки. Сервер не хранит результаты предыдущих сессий с клиентским приложением. Это повышает гибкость и масштабируемость серверной части, позволяет поддерживать асинхронное взаимодействие и реализовывать любые сложные алгоритмы. Кроме того, такая форма взаимодействия является

универсальной, не зависит от технологии, используемой клиентом и сервером, и не привязывает разработчика к конкретному провайдеру.

В процессе проектирования программных средств было решено использовать в приложении шаблон MVC (Model-View-Controller) [26]. Данный шаблон разделяет приложение на три модуля:

- модель (Model) – основная логика приложения, отвечающая за данные, методы работы с ними и структуры программы, модель реагирует на команды из контроллера и выдает информацию или изменяет свое состояние, после чего передает данные в представление;

- представление (View) – отвечает за визуализацию информации и отображение данных на уровне пользовательского интерфейса;

- контроллер (Controller) – обеспечивает взаимодействие с системой, обрабатывает действия пользователя, проверяет полученную информацию и передает её модели.

Логическая схема MVC–приложения приведена на рисунке 29.

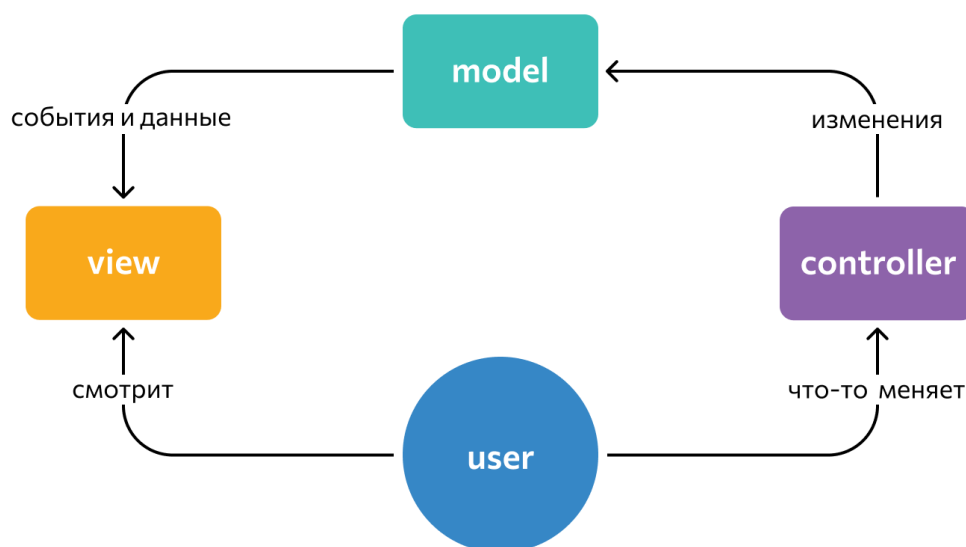


Рисунок 29 – Логическая схема MVC-приложения

Данный подход позволяет изменять каждый компонент независимо друг от друга для простой разработки и поддержки.

3 Реализация мобильного приложения

3.1 Реализация базы данных

На основе составленной ранее логической модели базы данных с учетом специфики СУБД Microsoft SQL Server была построена физическая модель базы данных, представленная на рисунке 30.

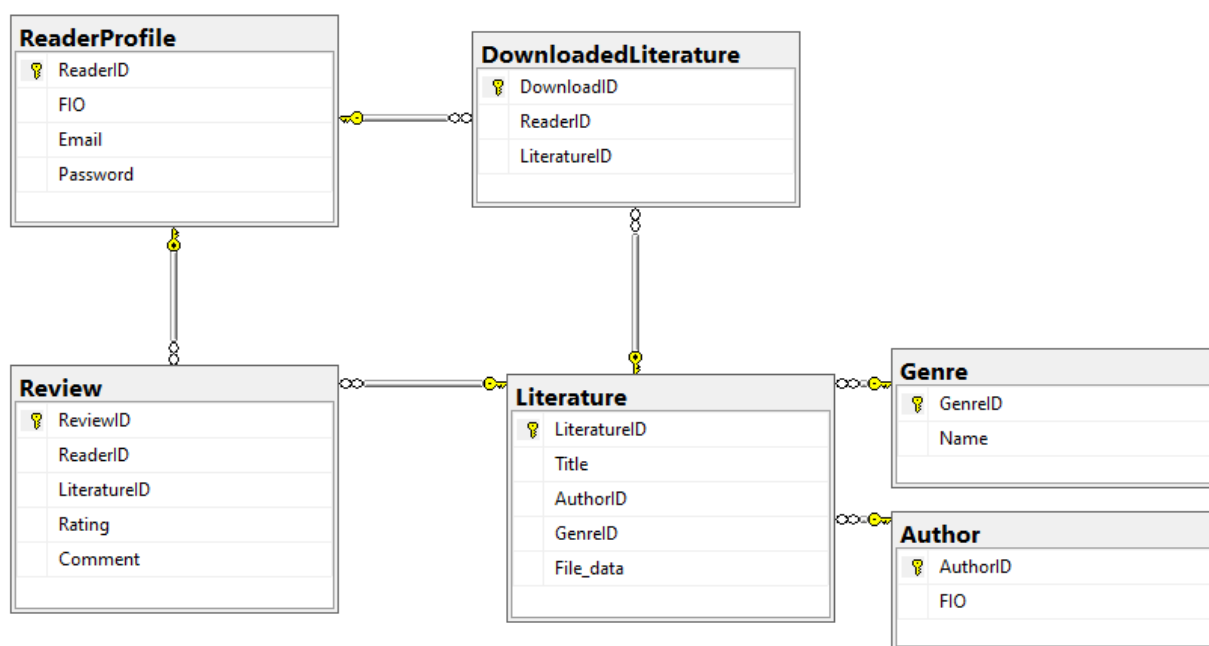


Рисунок 30 – Физическая модель базы данных

Создание базы данных было произведено с использованием скрипта на языке SQL. Его код представлен в приложении В. Развертывание построенной базы данных было произведено с использованием SQL Server Management Studio, данное программное средство используется для администрирования баз данных под управлением СУБД Microsoft SQL Server.

3.2 Разработка серверной части

Разработка серверной части разрабатываемого приложения производилась в среде программирования Microsoft Visual Studio 2022 на языке программирования C# с применением технологии Web API [27, 28]. Реализация взаимодействия с базой данных была выполнена с использованием технологии Entity Framework [29].

В процессе разработки была применена архитектура MVC, которая подразумевает, что все классы серверной части приложения можно разделить на два слоя: слой модели, предназначенной для организации взаимодействия с базой данных, и слой контроллеров, предназначенный для обработки HTTP-запросов к серверу со стороны клиентской части приложения.

Для лучшего представления устройства серверной части приложения были построены диаграммы классов уровней модели и контроллеров, представленные на рисунках 31, 32.

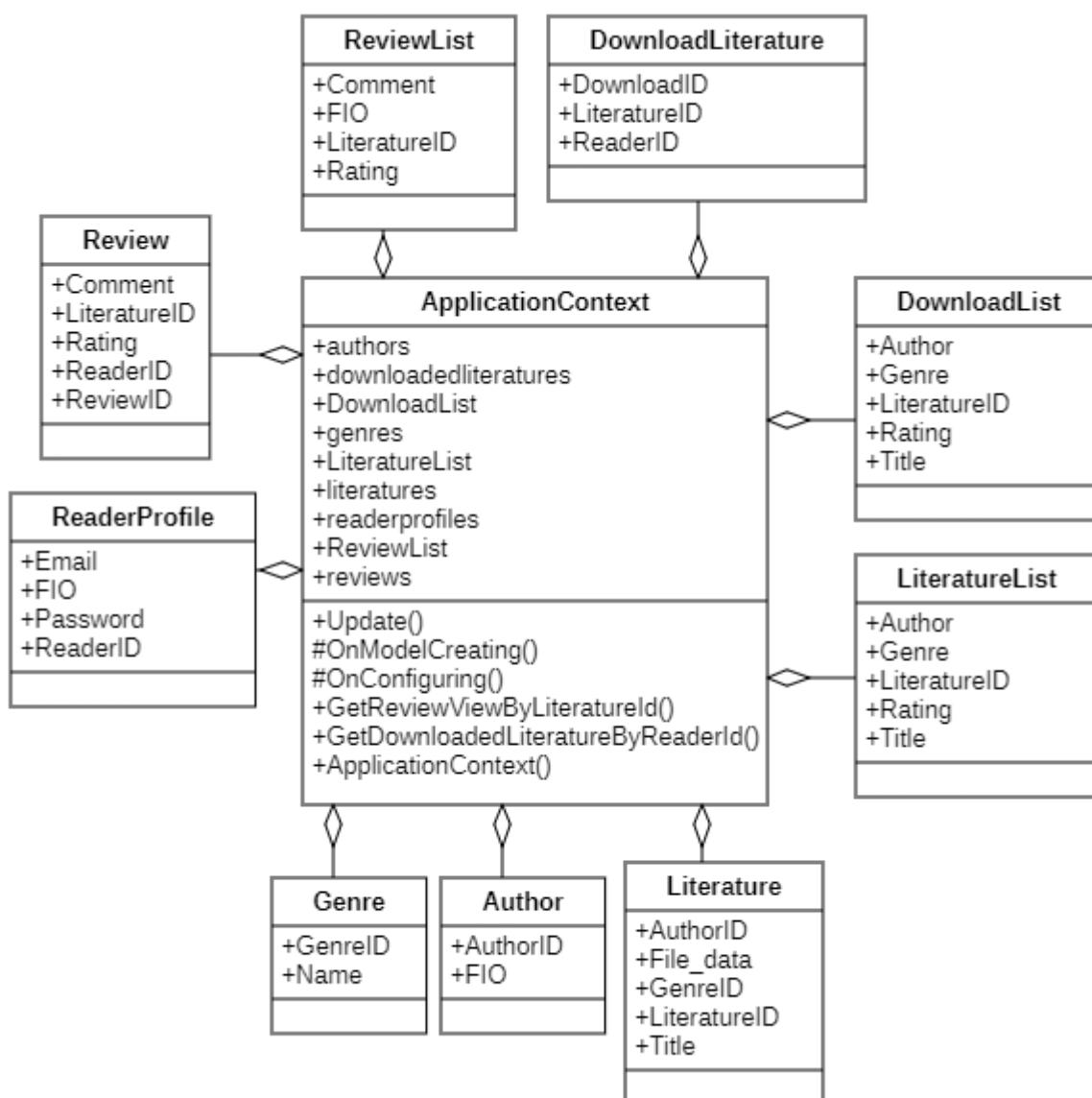


Рисунок 31 – Диаграмма классов серверной части (слой модели)

Классы `Author.cs`, `DownloadedLiterature.cs`, `Genre.cs`, `Literature.cs`, `ReaderProfile.cs`, `Review.cs` были созданы на основании одноименных таблиц

базы данных. Класс LiteratureList.cs же основываются на одноименных представлениях и позволяют облегчить выборку нужных данных из БД. В свою очередь, классы ReviewList.cs, DownloadList.cs используются для получения данных из функций, возвращающих табличные значения.



Рисунок 32 – Диаграмма классов серверной части (слой контроллеров)

Подробное описание классов слоя контроллеров было представлено в таблице 10.

Таблица 10 – Описание классов слоя контроллеров

Класс	Назначение
DownloadedLiteratureController.cs	Отвечает за обработку запросов на получение списка скаченной литературы, информации о конкретной скаченной книге
LiteratureController.cs	Отвечает за обработку запросов на получения списка всей литературы в электронном каталоге, сортировки по названию, автору и рейтингу, а также получение информации о конкретном экземпляре литературы и загрузки её в приложение пользователя
ReaderProfileController.cs	Отвечает за обработку запросов на авторизацию и регистрацию пользователей
ReviewController.cs	Отвечает за обработку запросов на получения списка всех отзывов на конкретную литературу, а также на добавление/изменение отзыва.

Листинг серверной части разрабатываемого программного средства представлен в приложении Г.

3.3 Реализация клиентской части приложения

В процессе разработки мобильного приложения с использованием среды программирования Visual Studio 2022 был применен фреймворк Xamarin для кроссплатформенной разработки мобильных приложений на языке C#. Компоненты программы отображены на диаграмме компонентов (рисунок 33), а физическое расположение компонентов системы описано на диаграмме развертывания (рисунок 34).

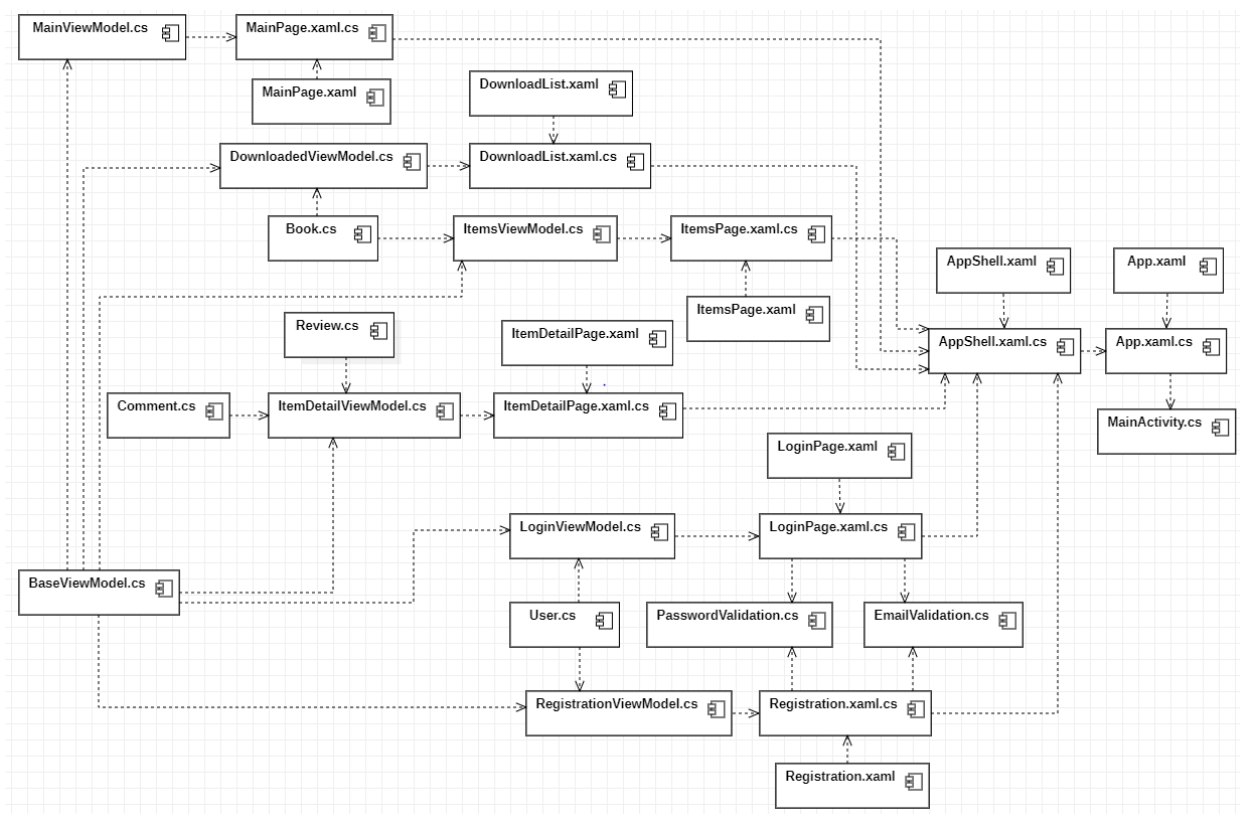


Рисунок 33 – Диаграмма компонентов клиентской части приложения

Описание основных компонентов представлено в таблице 11.

Таблица 11 – Компоненты клиентского приложения

Компонент	Назначение
MainViewModel.cs	Файл класса, реализующий функционал главной страницы
BaseViewModel.cs	Файл класса, реализующий базовый функционал страниц

Продолжение таблицы 11

DownloadedViewModel.cs	Файл класса, реализующий функционал страницы загруженной литературы
ItemDetailViewModel.cs	Файл класса, реализующий функционал страницы конкретной литературы
ItemsViewModel.cs	Файл класса, реализующий функционал страницы электронного каталога
LoginViewModel.cs	Файл класса, реализующий функционал страницы авторизации
RegistrationViewModel.cs	Файл класса, реализующий функционал страницы регистрации
MainPage.xaml	XAML-файл, реализующий интерфейс главной страницы
MainPage.xaml.cs	Файл класса, реализующий взаимодействие с элементами интерфейса главной страницы
DownloadList.xaml	XAML-файл, реализующий интерфейс страницы загруженной литературы
DownloadList.xaml.cs	Файл класса, реализующий взаимодействие с элементами интерфейса страницы загруженной литературы
ItemDetailPage.xaml	XAML-файл, реализующий интерфейс страницы конкретной литературы
ItemDetailPage.xaml.cs	Файл класса, реализующий взаимодействие с элементами интерфейса страницы конкретной литературы
LoginPage.xaml	XAML-файл, реализующий интерфейс страницы авторизации
LoginPage.xaml.cs	Файл класса, реализующий взаимодействие с элементами интерфейса страницы авторизации
RegistrationPage.xaml	XAML-файл, реализующий интерфейс страницы регистрации
RegistrationPage.xaml.cs	Файл класса, реализующий взаимодействие с элементами интерфейса страницы регистрации
App.xaml	XAML-файл, хранит определенные ресурсы для всего приложения
App.xaml.cs	Реализует взаимодействие с интерфейсом приложения
AppShell.xaml	XAML-файл, отвечает за интерфейс навигационного меню
AppShell.xaml.cs	Отвечает за реализацию навигационного меню
MainActivity.cs	Точка входа в программное средство
EmailValidation.cs	Вспомогательный класс, реализующий методы валидации почтового адреса
PasswordValidation.cs	Вспомогательный класс, реализующий методы валидации вводимого пароля

На диаграмме развертывания можно увидеть, что клиентское приложение работает на мобильном устройстве с операционной системой Android. На сервере, работающем под управлением ОС Windows, находятся API и база

данных. Для взаимодействия между клиентской и серверной частями приложения используется протокол HTTP.

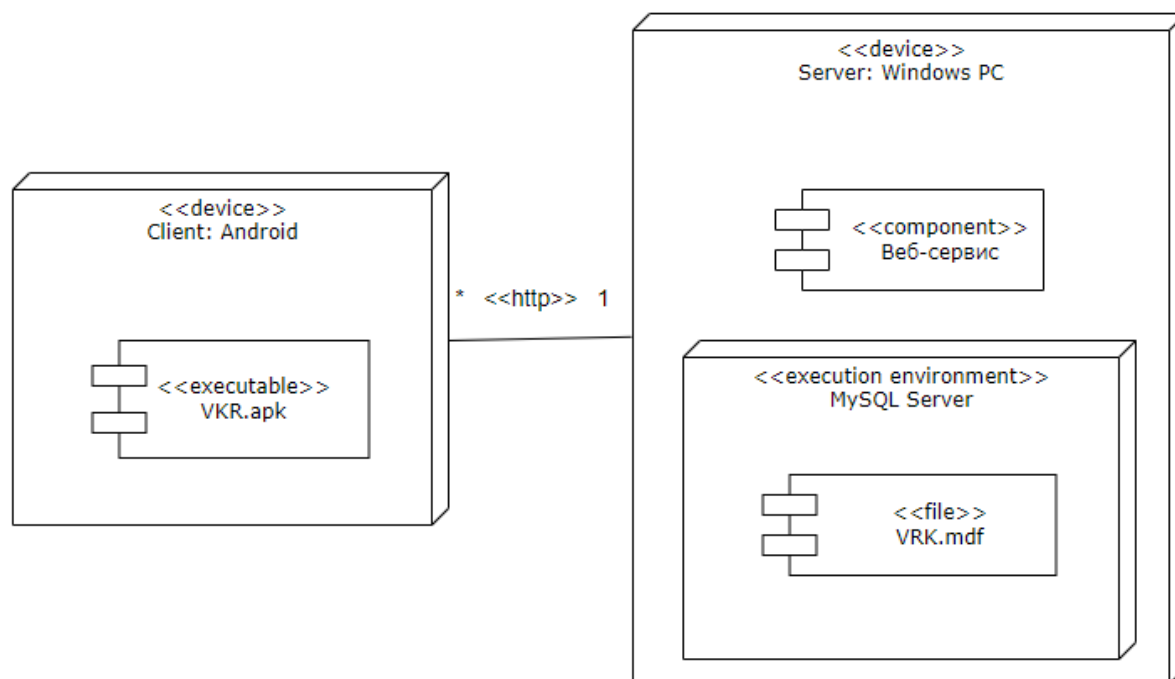


Рисунок 34 – Диаграмма развертывания разрабатываемого программного средства

Листинг клиентской части программного средства представлен в приложении Д.

4 Тестирование программного продукта

4.1 Тестирование серверной части приложения

Убедимся в корректности работы серверной части приложения. Для этого было произведено функционально тестирование с использованием встроенного в Visual Studio 2022 инструментального средства. Результаты проведенного тестирования представлены в таблице 12. Скриншоты проведенного тестирования вынесены в приложение Е.

Таблица 12 – Результаты тестирования основных запросов серверной части приложения

№	Описание теста	Запрос		Результат
		Метод	Параметры	
1	Регистрация пользователя	POST	«fio» «Косов», «email»: «kos@mail.ru», «password»: «Qwerty123»	Тест пройден. Пользователь успешно добавлен в систему. Id пользователя было возвращено Код ответа HTTP – 200 (рисунок Е.1).
2	Авторизация пользователя	POST	Email = anikin@mail.ru Password = Qwerty123	Тест пройден. Пользователь успешно авторизован в системе. Id пользователя было возвращено Код ответа HTTP – 200 (рисунок Е.2).
3	Получение информации о конкретной литературе	GET	Literature id = 1	Тест пройден. Информация о книге успешно возвращена Код ответа HTTP – 200 (рисунок Е.3).
4	Скачивание книги	GET	Literature id = 1 User id = 1	Тест пройден. Скачивание произошло успешно. Был возвращен список byte[] Код ответа HTTP – 200 (рисунок Е.4).
5	Получение списка всей литературы	GET		Тест пройден. Список книг возвращен. Код ответа HTTP – 200 (рисунок Е.5).
6	Получение списка скаченной литературы	GET		Тест пройден. Список скаченных книг возвращен. Код ответа HTTP – 200 (рисунок Е.6).
7	Удаление скаченной литературы	DELETE	Literature id=1 User id=1	Тест пройден. Книга удалена. Код ответа HTTP – 200 (рисунок Е.7).
8	Получение списка всех комментариев к данной литературе	GET	Literature id = 1	Тест пройден. Список комментариев возвращен. Код ответа HTTP – 200 (рисунок Е.8).
9	Добавление комментария	POST	«readerID»: 3, «literatureID»: 1, «rating»: 9, «comment»: «Комментари»	Тест пройден. Комментарий добавлен. Код ответа HTTP – 200 (рисунок Е.9).

Продолжение таблицы 12

№	Описание теста	Запрос		Результат
		Метод	Параметры	
10	Проверка, скачена ли литература	GET	Literature id = 1 Reader id = 1	Тест пройден. Код ответа HTTP – 200 (рисунок Е.10).
11	Получение списка всей литературы с сортировкой по названию	GET		Тест пройден. Список книг возвращен. Код ответа HTTP – 200 (рисунок Е.11).
12	Получение списка всей литературы с сортировкой по Автору	GET		Тест пройден. Список книг возвращен. Код ответа HTTP – 200 (рисунок Е.12).

По результатам тестов можно судить о корректной работе серверной части приложения.

4.2 Тестирование клиентской части приложения

4.2.1 Функциональное тестирование

Для проверки корректности работы клиентской части приложения было проведено функциональное тестирование. Его результаты представлены в таблице 13. Скриншоты тестов находятся в приложении Ж.

Таблица 13 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Авторизоваться	Авторизация пользователя с корректными данными Email: anikin@mail.ru Password: Qwerty123	Тест пройден, авторизация успешна (рисунок Ж.1, Ж.2).
	Авторизация пользователя с незарегистрированного в системе Email: an@mail.ru Password: Qwerty3	Тест пройден. Показано сообщение о ошибке (рисунок Ж.3).
Регистрация	Регистрация нового пользователя	Тест пройден. Регистрация успешна (рисунок Ж.4).
	Регистрация пользователя, используя уже зарегистрированный в системе email	Тест пройден. Получено сообщение о ошибке (рисунок Ж.5).
Открыть электронный каталог	Открытие электронного каталога и вывод перечня книг	Тест пройден. Книги получены (рисунок Ж.6).
Сортировка электронного каталога по названию	Переключение режима сортировки на сортировку по названию	Тест пройден. Книги отсортированы по названию (рисунок Ж.7).

Продолжение таблицы 13

Сортировка электронного каталога по автору	Переключение режима сортировки на сортировку по автору	Тест пройден. Книги отсортированы по автору (рисунок Ж.8).
Открыть список загруженных книг	Открытия страницы загруженных книг и вывод перечня скаченных пользователем книг	Тест пройден. Книги получены (рисунок Ж.9).
Открыть страницу книги	Открытие страницы книги, вывод информации о ней и комментариев	Тест пройден. Страница определенной книги выведена (рисунок Ж.10).
Оставить отзыв	Тестирование функционала комментирование литературы. Введенная оценка равна 5, комментарий, следующий: «Хорошая книга»	Тест пройден. Отзыв успешно добавлен (рисунок Ж.11–Ж.13).
Скачивание и открытие книги	Тестирование функционала загрузки литературы на устройство и его дальнейшее открытие.	Тест пройден. Книга успешно открыта (рисунок Ж.14).
Удаление литературы с устройства	Тестирование возможности удаления литературы	Тест пройден. Книга удалена (рисунок Ж.15).

Функциональное тестирование подтвердило корректность работы приложения.

4.2.2 Модульное тестирование

Для оценки корректности работы автономных единиц программного кода (как правило, подпрограмм) используются модульные тесты. Они сравнивают фактический результат работы с ожидаемым. Одним из главных преимуществ модульных тестов является возможность быстрого выявления и устранения ошибок при модификации кода. В связи с этим в рамках данного проекта было решено покрыть методы `isValidEmail` и `isValidPassword` модульными тестами. Данные методы проверяют корректность значений электронной почты и пароля, введенных пользователем при регистрации и авторизации.

Коды тестовых классов `EmailValidationTests` и `PasswordValidationTests`, которые были созданы для проведения модульных тестов, содержатся в листинге 1 и листинге 2 соответственной.

Листинг 1 – Исходный код модульных тестов EmailValidationTests

```
[TestClass()]
public class EmailValidationTests
{
    [TestMethod()]
    public void ValidEmail()
    {
        EmailValidation validation = new EmailValidation();
        Assert.AreEqual(true, validation.IsValidEmail("anikin@mail.ru"));
    }
    [TestMethod()]
    public void WithoutSpecialLetter()
    {
        EmailValidation validation = new EmailValidation();
        Assert.AreEqual(false, validation.IsValidEmail("anikmail.ru"));
    }
    [TestMethod()]
    public void EndWithDot()
    {
        EmailValidation validation = new EmailValidation();
        Assert.AreEqual(false,
            validation.IsValidEmail("anikin@mail.ru."));
    }
    [TestMethod()]
    public void WithSpace()
    {
        EmailValidation validation = new EmailValidation();
        Assert.AreEqual(false,
            validation.IsValidEmail("anikin@mail.ru "));
    }
    [TestMethod()]
    public void isCyrillic()
    {
        EmailValidation validation = new EmailValidation();
        Assert.AreEqual(false,
            validation.IsValidEmail("АНИКИН@mail.ru"));
    }
    [TestMethod()]
    public void WrongSpecialLetters()
    {
        EmailValidation validation = new EmailValidation();
        Assert.AreEqual(false,
            validation.IsValidEmail("anikin*$!*&)(?^dsa%dsa@mail.ru"));
    }
}
```

Листинг 2 – Исходный код модульных тестов PasswordValidationTests

```
[TestClass()]
public class PasswordValidationTests
{
    [TestMethod()]
    public void ValidPassword()
    {
        PasswordValidation validation = new PasswordValidation();
        Assert.AreEqual(true, validation.isValidPassword("Qwerty123"));
    }
    [TestMethod()]
    public void WithoutNumbers()
    {

```

```

        PasswordValidation validation = new PasswordValidation();
        Assert.AreEqual(false,
            validation.IsValidPassword("Qwertyafsafta"));
    }
    [TestMethod()]
    public void WithoutCapitalLetter()
    {
        PasswordValidation validation = new PasswordValidation();
        Assert.AreEqual(false, validation.IsValidPassword("qwerty123"));
    }
    [TestMethod()]
    public void tooShort()
    {
        PasswordValidation validation = new PasswordValidation();
        Assert.AreEqual(false, validation.IsValidPassword("Qty123"));
    }
    [TestMethod()]
    public void isCyrillic()
    {
        PasswordValidation validation = new PasswordValidation();
        Assert.AreEqual(false,
            validation.IsValidPassword("Qwerрусскийty123"));
    }
}

```

Результат проведения модульных тестов представлен на рисунке 35.

▲ EmailValidationTests (6)	3 мс
✓ EndWithDot	3 мс
✓ isCyrillic	< 1 мс
✓ ValidEmail	< 1 мс
✓ WithoutSpecialLetter	< 1 мс
✓ WithSpace	< 1 мс
✓ WrongSpecialLetters	< 1 мс
▲ PasswordValidationTests (...)	< 1 мс
✓ isCyrillic	< 1 мс
✓ tooShort	< 1 мс
✓ ValidPassword	< 1 мс
✓ WithoutCapitalLetter	< 1 мс
✓ WithoutNumbers	< 1 мс

Рисунок 35 – Результаты проведения модульных тестов методов «IsValidEmail» и «IsValidPassword»

Модульное тестирование подтвердило корректность работы проверяемых модулей.

4.3 Метрики кода

Для оценки качества исходного кода написанного в результате реализации программного продукта были вычислены различные метрики кода. Для вычисления метрик возьмём метод PostReview, который реализует обработку HTTP запроса на добавление комментария. Код данного метода приведен в листинге 3.

Листинг 3 – Добавление комментария

```
public async Task<ActionResult> PostReview (Review review)
{
    try
    {
        //Проверка на корректность полученных данных
        if (review == null || review.LiteratureID == null ||
            review.ReaderID == null || review.Rating == null ||
            review.Comment == null ) {
            return BadRequest("Введите запрошенные данные");
        }
        //Поиск комментария пользователя к указанной литературе
        var check = db.reviews.FirstOrDefault(x => (x.ReaderID ==
            review.ReaderID) && (x.LiteratureID ==
            review.LiteratureID));
        // Если комментарий найден, то заменяем его
        if (check != null)
        {
            check.Comment = review.Comment;
            check.Rating = review.Rating;
            db.SaveChanges();
        }
        // Если комментарий отсутствует, то добавляем его
        else
        {
            review.ReviewID = db.reviews.Max(x => x.ReviewID)+1;
            db.reviews.Add(review);
            db.SaveChanges();
        }
        // Успешное завершение
        return Ok(1);
    }
    catch
    {
        //Завершение с ошибкой
        return BadRequest();
    }
}
```

Данный метод находится в серверной части приложения и запускается при попытке добавления нового комментария к литературе. Он проверяет корректность вводимых данных, проверяет наличие оставленного комментария, если данный пользователь уже комментировал данное произведения, то происходит замена существующего комментария, иначе происходит добавление нового комментария.

Для данного метода будут вычислены три группы метрик: метрики размера программы, метрики Мак-Кейба и метрики Холстеда.

4.3.1 Метрика размера программы

Метрика размера программы представляет собой подсчет количества строк программного кода, а также числа строк комментариев. Результат вычисления данной метрики по приведенному выше колу программного средства представлен в таблице 14.

Таблица 14 – Метрики размера программы

Метрика	Количество
Количество строк кода	31
Строки-комментарии	6

Теперь вычислим процентное соотношение комментариев, оно равно $(6/31)*100 = 19.35\%$. Данный результат свидетельствует о том, что количество комментариев достаточное для хорошего понимания кода, притом отсутствует излишняя избыточность комментирования, которая могла бы излишне раздувать код программного средства.

4.3.2 Метрики Мак-Кейба

Метрика Мак-Кейба, также известная, как цикломатическая метрика, позволяет оценить структурную сложность программы на основе управляющего графа программы.

Построенный управляющий граф анализируемого кода представлен на рисунке 36.

Посчитаем количество вершин и дуг построенного графа: количество вершин $v=17$, количество дуг $e=17$. А также найдем компонент связности графа (p), для его нахождения необходимо посчитать количество таких подмножеств вершин графа, что каждые две вершины в этом подмножестве соединены хотя бы одной дугой. В данном графе $p=3$.

Вычислим метрику Мак-Кейба:

$$C=e-v+2p=6.$$

Исходя из того, что полученное значение меньше 10, можно сделать вывод о том, что анализируемый программный код обладает достаточно простой структурой потока управления и легок для восприятия.

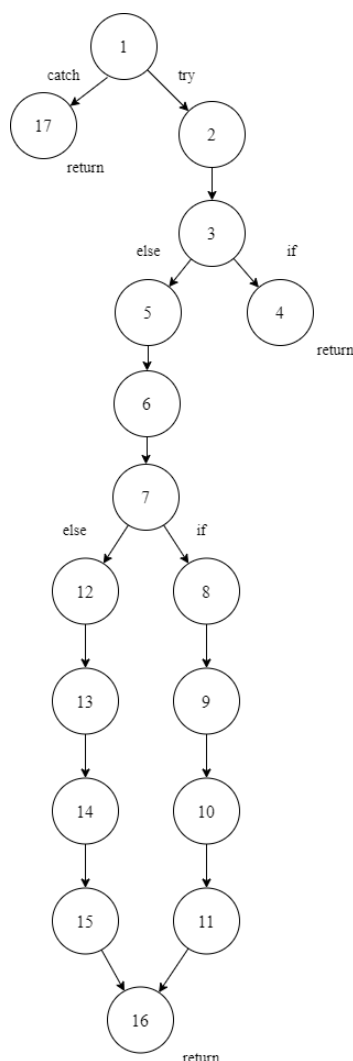


Рисунок 36 – Управляющий граф метода «PostReview»

Результаты вычисления данной метрики положительны, что свидетельствует о достаточно высоком качестве кода.

4.3.3 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы [30 – 32]:

– NUOprtr (Number of Unique Operators) – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);

– NUOprnd (Number of Unique Operands) – число уникальных операндов программы (словарь операндов);

– Noprtr (Number of Operators) – общее число операторов в программе;

– Noprnd (Number of Operands) – общее число операндов в программе.

Операторы и операнды приведены в таблицах 15 и 16 соответственно.

Таблица 15 – Словарь операторов

Оператор	Число операторов
==	7
=	4
!=	1
=>	2
&&	1
()	9
{}	6
;	10
.	22
if	2
return	3
try	1
catch	1
	4
+	1

Таблица 16 – Словарь операндов

Операнд	Число операндов
db	5
review	15
check	4

Таким образом, число уникальных операторов программы $n1 = 15$, число уникальных операндов программы $n2 = 3$, общее число операторов в программе $N1 = 74$, общее число операндов в программе $N2 = 24$.

Рассчитаем словарь программы: $n = 15 + 3 = 18$.

Рассчитаем длину программы $N = 74 + 24 = 98$.

Рассчитаем объем программы: $NV = 98 * \log_2 18 = 123$.

Потенциальный объем программы:

$$V^* = (2 + n^*) * \log_2(2 + n^*) = 4 * \log_2(4) = 8.$$

Рассчитаем сложность программы: $D = (15/2) * (24/3) = 60$.

Рассчитаем оценку интеллектуальных усилий: $E = 60 * 123 = 7380$.

Рассчитаем теоретический уровень программы по формуле:

$$L^* = V^* / HV = 8 / 123 = 0.065.$$

Полученные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки.

Заключение

В ходе выполнения выпускной квалификационной работы было спроектировано мобильное приложение для поддержки процессов организации и функционирования муниципальной электронной библиотеки. А также выполнена его реализация и тестирование

На этапе анализа требований и предметной области были проанализированы аналогичные программные продукты конкурентов, выявлены их основные сильные и слабые стороны, произведено сравнения с разрабатываемым программным средством, определены функциональные и нефункциональные требования к создаваемому приложению и проанализированы потенциальные потребители программного продукта.

Был составлен план разработки приложения и оценен бюджет. В результате общая продолжительность проекта составила 52 дня, а потенциальная стоимость разработки и внедрения – 272 040 рублей, что подтверждает экономическую целесообразность проекта программного обеспечения.

На этапе проектирования была составлена логическая модель базы данных и разработана архитектура мобильного приложения. В ходе реализации приложения, база данных была создана с использованием языка программирования SQL и развернута под управлением СУБД Microsoft SQL Server 2019. Также построена диаграмма, отражающая компоненты этой базы данных. В процессе разработки клиентской части приложения решено использовать такие технологии разработки, как RUP, ООП, MVC. Разработка серверной части приложения проводилась на языке C# с использованием среды программирования Microsoft Visual Studio 2022, а клиентская часть программного средства реализована на языке C# в среде программирования Microsoft Visual Studio 2022, с применением фреймворка для кроссплатформенной разработки мобильных приложений Xamarin.

В ходе тестирования было выполнено тестирование серверной и клиентской частей разрабатываемого программного средства. Успешное прохождение тестирования подтверждает корректность работы приложения.

Также были рассчитаны метрики программного кода, результаты расчетов свидетельствуют о достаточно высоком уровне качества кода программного средства.

Таким образом, в результате выполнения выпускной квалификационной работы было проведено проектирование программного средства, анализ аналогов и требований, составлен план разработки приложения и оценен необходимый бюджет, а также выполнена реализация и тестирование разрабатываемого программного продукта. Все поставленные задачи успешно выполнены.

Список использованных источников

1. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 10.05.2023).
2. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 10.05.2023).
3. Распоряжение Правительства Российской Федерации от 13 марта 2021 г. № 608-р «Стратегия развития библиотечного дела в Российской Федерации на период до 2030 года». [Электронный ресурс]. URL: <http://static.government.ru/media/files/NFWPpXpAAAEbPW60HiZiDvdZZ8AcNuu.pdf> (дата обращения: 10.05.2023).
4. Фазлиева, Л.Ю. Современные тенденции развития библиотек и библиотечного обслуживания населения / Л.Ю. Фазлиева // Стратегии развития социальных общностей, институтов и территорий: материалы Международной научно-практической конференции. – Екатеринбург:Изд-во Уральского федерального университета, 2015. – С. 103 – 105.
5. Лермонтовка: вектор развития // Пензенская областная библиотека имени М. Ю. Лермонтова liblermont.ru [Электронный ресурс]. URL: <http://liblermont.ru/index.php?page=bibl&id=13> (дата обращения: 10.05.2023).
6. Пензенская правда | Пензенская Лермонтовка // [PenzaInform penzainform.ru](http://PenzaInform.penzainform.ru) [Электронный ресурс]. URL: https://www.penzainform.ru/press/pp/2022/10/27/penzenskaya_lermontovka_12_etazhej_informatcii_i_balkon_lyubvi.html (дата обращения: 10.05.2023).
7. Кузнецкая центральная городская библиотека им. А.Н. Радищева // Культура.РФculture.ru [Электронный ресурс]. URL: <https://www.culture.ru>

w.culture.ru/institutes/24911/kuzneckaya-centralnaya-gorodskaya-biblioteka-im-a-n-radisheva (дата обращения: 10.05.2023).

8. Об учреждении // Кузнецкая центральная городская библиотека им. А.Н. Радищева kuzbibliok.ru [Электронный ресурс]. URL: <http://kuzbibliok.ru/об-учреждении/> (дата обращения: 10.05.2023).

9. Общая информация // Муниципальное бюджетное учреждение «Карпогорская межпоселенческая библиотека» Пинежского муниципального района Архангельской области. karpbibl.ru [Электронный ресурс]. URL: <https://karpbibl.ru/information> (дата обращения: 10.05.2023).

10. Коновалова Н. А. ОБЩЕДОСТУПНЫЕ БИБЛИОТЕКИ РОССИЙСКОЙ ФЕДЕРАЦИИ В ЦИФРАХ 2021 / Коновалова Н.А., Огородникова О.В. // ФГБУ «ГИВЦ Минкультуры России», 2021. – 213 с.

11. Гендиректор РГБ указал на рост популярности у россиян цифровых ресурсов библиотек // ТАСС tass.ru [Электронный ресурс]. URL: <https://tass.ru/kultura/14751121> (дата обращения: 10.05.2023).

12. Электронные книги (рынок России) // Tadviser Государство. Бизнес. Технологии tadviser.ru [Электронный ресурс]. URL: [https://www.tadviser.ru/index.php/Статья:Электронные_книги_\(рынок_России\)](https://www.tadviser.ru/index.php/Статья:Электронные_книги_(рынок_России)) (дата обращения: 10.05.2023).

13. «ЛитРес» предрекла рост российского рынка цифровых книг в 2023 году // Агентство экономической информации Прайм 1prime.ru [Электронный ресурс]. URL: <https://1prime.ru/business/20230209/839745243.html> (дата обращения: 10.05.2023).

14. Аналитика трафика и доля рынка litres.ru // Similarweb similarweb.com [Электронный ресурс]. URL: https://pro.similarweb.com/#/digitalsuite/websiteanalysis/overview/website-performance/*/999/3m?webSource=Total&key=litres.ru (дата обращения: 10.05.2023).

15. Итоговая статистика рынка смартфонов за 2022 год // mobiltelefon.ru [Электронный ресурс]. URL: https://mobiltelefon.ru/post_1674027176.html (дата обращения: 10.05.2023).
16. Android vs iOS market share – 2023 Statistics [Infographic] // Download Astro en.downloadastro.com [Электронный ресурс]. URL: <https://en.downloadastro.com/magazine/android-vs-ios-market-share/> (дата обращения: 10.05.2023).
17. Google обнародовала статистику версий Android // 4PDA 4pda.to [Электронный ресурс]. URL: <https://4pda.to/2023/01/19/408761> (дата обращения: 10.05.2023).
18. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.
19. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.
20. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.
21. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 стр.
22. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.
23. Процессы и стадии RUP // QA evolution qaevolution.ru [Электронный ресурс]. URL: <https://qaevolution.ru/metodologiya-menedzhment/rup/> (дата обращения: 10.05.2023).
24. Буч Г., Якобсон А., Рамбо Д. UML. 2-е издание Классика CS. Питер, 2005, – 736 с.
25. Архитектура «Клиент-Сервер» // Itelon itelon.ru [Электронный ресурс]. URL: <https://itelon.ru/blog/arkhitektura-klient-server/> (дата обращения: 10.05.2023).

26. MVC // Skillfactory Media skillfactory.ru [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/mvc> (дата обращения: 10.05.2023).
27. Введение в Web API // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/mvc/12.1.php> (дата обращения: 15.05.2023).
28. Андерсон Р., Ларкин К., Уоссон М. Создание веб-API с помощью ASP.NET Core и Visula Studio для Windows // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio> (дата обращения: 15.05.2023).
29. Руководство по Entity Framework // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/entityframework/2.4.php> (дата обращения: 15.05.2023).
30. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 15.05.2023).
31. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.
32. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.

Приложение А
ГЛОССАРИЙ
(справочное)

Android – популярная операционная система, построенная на ядре Linux. Используется в смартфонах, планшетах, электронных книгах, цифровых проигрывателях, наручных часах, фитнес-браслетах, игровых приставках, ноутбуках, нетбуках, телевизорах и других устройствах. Данная мобильная операционная система, разрабатываемая и выпускаемая американской компанией Google, используется на подавляющем большинстве современных мобильных устройств.

RUP (Rational Unified Process) – методология разработки программного обеспечения, в основе которой лежит итеративная модель разработки.

API (Application Programming Interface — программный интерфейс приложения, или интерфейс программирования приложений) — специальный протокол для взаимодействия компьютерных программ, который позволяет использовать функции одного приложения внутри другого.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент-сервер».

SQL (Structured Query Language) – это язык программирования для хранения и обработки информации в реляционной базе данных. Реляционная база данных хранит информацию в табличной форме со строками и столбцами, представляющими различные атрибуты данных и различные связи между значениями данных. Инструкции SQL можно использовать для хранения, обновления, удаления, поиска и извлечения информации из базы данных. Можно также использовать SQL для поддержания и оптимизации производительности базы данных.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Xamarin – это фреймворк для кроссплатформенной разработки мобильных приложений (iOS, Android, Windows Phone) с использованием языка C#.

MVC – шаблон (паттерн) программирования, разделяющий архитектуру приложения на три модуля: модель (Model), представление (View), контроллер (Controller). Он позволяет изменять каждый компонент независимо друг от друга для простой разработки и поддержки веб-приложений.

Электронная библиотека – упорядоченная коллекция разнородных электронных документов (например, книг и журналов), снабжённая средствами навигации и поиска.

Приложение Б
ДИАГРАММЫ ДЕЯТЕЛЬНОСТИ
(рекомендуемое)

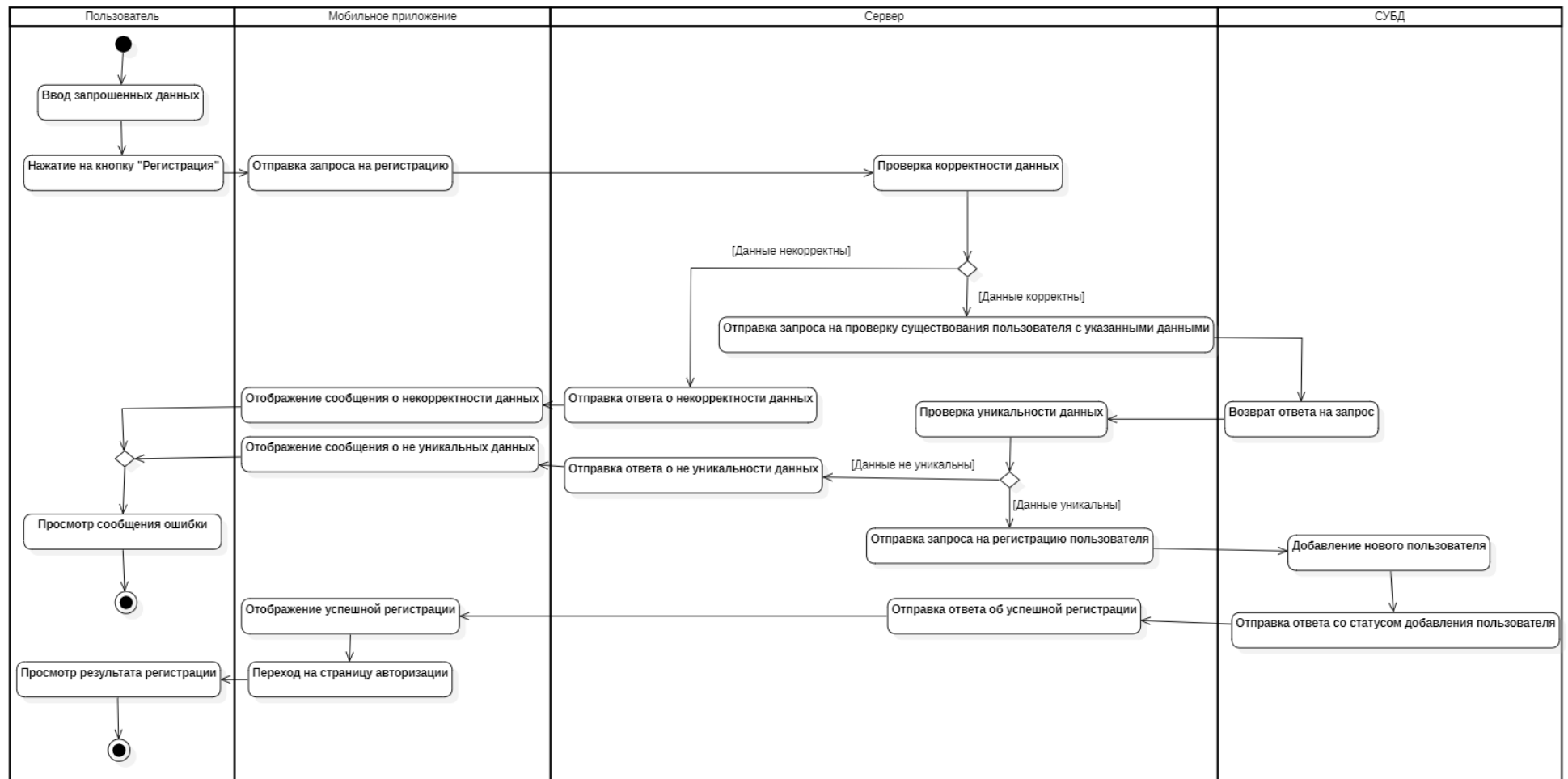


Рисунок Б.1 – Диаграмма деятельности варианта использования «Зарегистрироваться»

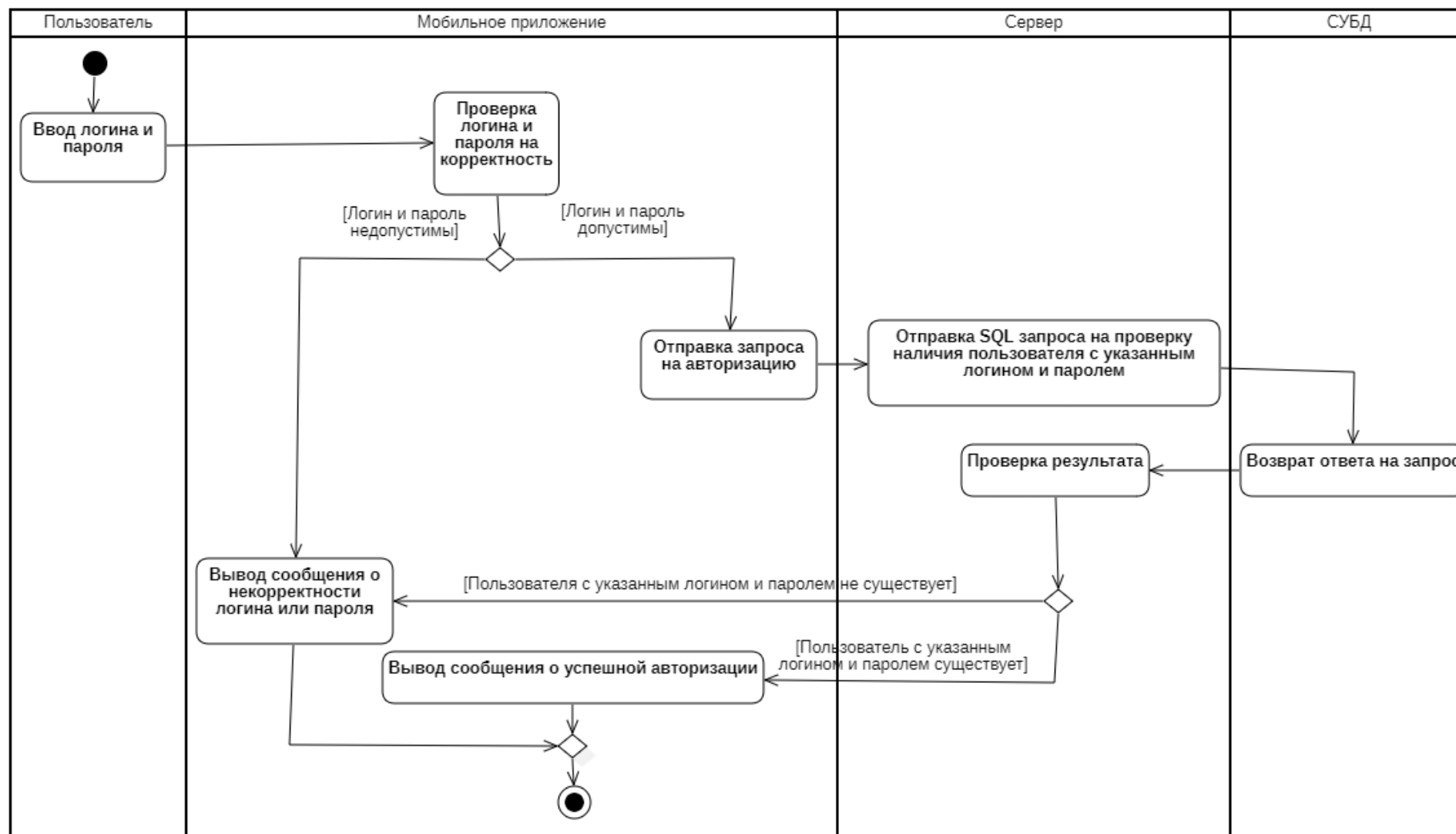


Рисунок Б.2 – Диаграмма деятельности варианта использования «Войти в систему»

Приложение В
ИСХОДНЫЙ КОД SQL СКРИПТОВ БАЗЫ ДАННЫХ
(рекомендуемое)

Листинг В.1 – Исходный код скрипта создания таблиц базы данных

```

CREATE TABLE ReaderProfile (
    ReaderID INT PRIMARY KEY,
    FIO VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Password VARCHAR(50) NOT NULL
);

CREATE TABLE Author (
    AuthorID INT PRIMARY KEY identity,
    FIO VARCHAR(50) NOT NULL
);

CREATE TABLE Genre (
    GenreID INT PRIMARY KEY identity,
    Name VARCHAR(50) NOT NULL,
);

CREATE TABLE Literature (
    LiteratureID INT PRIMARY KEY identity,
    Title VARCHAR(200) NOT NULL,
    AuthorID INT NOT NULL,
    GenreID INT NOT NULL,
    File_data VARBINARY(MAX) NOT NULL,
    FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID),
    FOREIGN KEY (GenreID) REFERENCES Genre(GenreID)
);

CREATE TABLE DownloadedLiterature (
    DownloadID INT PRIMARY KEY identity,
    ReaderID INT NOT NULL,
    LiteratureID INT NOT NULL,
    FOREIGN KEY (ReaderID) REFERENCES ReaderProfile(ReaderID),
    FOREIGN KEY (LiteratureID) REFERENCES Literature(LiteratureID)
);

CREATE TABLE Review (
    ReviewID INT PRIMARY KEY,
    ReaderID INT NOT NULL,
    LiteratureID INT NOT NULL,
    Rating INT NOT NULL,
    Comment VARCHAR(500) NOT NULL,
    FOREIGN KEY (ReaderID) REFERENCES ReaderProfile(ReaderID),
    FOREIGN KEY (LiteratureID) REFERENCES Literature(LiteratureID)
);

```

Листинг В.2 – Исходный код создания представления LiteratureView (основная информация о книге)

```

CREATE VIEW LiteratureView AS

SELECT Literature.LiteratureID, Literature.Title, Author.FIO as Author,
Genre.Name as Genre, (SELECT AVG(Review.Rating) FROM Review WHERE
Review.LiteratureID = Literature.LiteratureID) AS Rating

FROM Literature

INNER JOIN Author ON Literature.AuthorID = Author.AuthorID

INNER JOIN Genre ON Literature.GenreID = Genre.GenreID;

```

Листинг В.3 – Исходный код создания функции

GetDownloadedLiteratureByReaderId, возвращающей таблицу с основной информацией о скаченных книгах указанного по id пользователя

```
CREATE FUNCTION GetDownloadedLiteratureByReaderId (@readerId INT)
RETURNS TABLE
AS
RETURN
    SELECT l.LiteratureID, l.Title, a.FIO AS Author, g.Name AS Genre,
    (SELECT AVG(Review.Rating) FROM Review WHERE Review.LiteratureID =
    l.LiteratureID) AS Rating
    FROM DownloadedLiterature dl
    JOIN Literature l ON dl.LiteratureID = l.LiteratureID
    JOIN Author a ON l.AuthorID = a.AuthorID
    JOIN Genre g ON l.GenreID = g.GenreID
    WHERE dl.ReaderID = @readerId;
```

Листинг В.4 – Исходный код создания функции GetReviewViewByLiteratureID,

возвращающей таблицу с основной информацией о комментариях, оставленных к указанной по id книге

```
CREATE FUNCTION GetReviewViewByLiteratureID (@LiteratureId INT)
RETURNS TABLE
AS
RETURN
    SELECT Review.LiteratureID, ReaderProfile.FIO, Review.Comment, Review.Rating
    FROM Review
    INNER JOIN ReaderProfile ON Review.ReaderID = ReaderProfile.ReaderID
    WHERE Review.LiteratureID = @LiteratureId;
```

Приложение Г
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг Г.1 – Файл ApplicationContext.cs

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace WebServer.Models
{
    public class ApplicationContext : DbContext
    {
        public virtual DbSet<ReaderProfile> readerprofiles { get; set; }
        public virtual DbSet<Author> authors { get; set; }
        public virtual DbSet<Genre> genres { get; set; }
        public virtual DbSet<DownloadedLiterature> downloadedliteratures {
get; set; }
        public virtual DbSet<Literature> literatures { get; set; }
        public virtual DbSet<Review> reviews { get; set; }
        public virtual DbSet<LiteratureList> LiteratureLists { get; set; }
        public virtual DbSet<DownloadList> DownloadLists { get; set; }
        public virtual DbSet<ReviewList> ReviewLists { get; set; }
        public IQueryable<DownloadList> GetDownloadedLiteratureByReaderId
(int readerId) => FromExpression(() =>
GetDownloadedLiteratureByReaderId(readerId));
        public IQueryable<ReviewList> GetReviewViewByLiteratureID(int
LiteratureId) => FromExpression(() =>
GetReviewViewByLiteratureID(LiteratureId));
        public ApplicationContext(DbContextOptions<ApplicationContext>
optionsBuilder)
            : base(optionsBuilder)
        {
            Update();
        }
        public void Update()
        {
            readerprofiles.Load();
            authors.Load();
            genres.Load();
            downloadedliteratures.Load();
            literatures.Load();
            reviews.Load();
            LiteratureLists.Load();
        }
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<ReaderProfile>()
                .HasKey(m => new { m.ReaderID });
            modelBuilder.Entity<DownloadedLiterature>()
                .HasKey(m => new { m.ReaderID, m.LiteratureID, m.DownloadID
});
            modelBuilder.Entity<Genre>()
                .HasKey(m => new { m.GenreID });
            modelBuilder.Entity<Author>()
                .HasKey(m => new { m.AuthorID });
            modelBuilder.Entity<Literature>()
                .HasKey(m => new { m.LiteratureID, m.AuthorID, m.GenreID
});
            modelBuilder.Entity<Review>()
                .HasKey(m => new { m.ReviewID, m.ReaderID, m.LiteratureID
});

```

```

        modelBuilder.Entity<LiteratureList>(pc => { pc.HasNoKey();
pc.ToView("LiteratureView"); });
        modelBuilder.Entity<DownloadList>(pc => { pc.HasNoKey(); });
        modelBuilder.Entity<ReviewList>(pc => { pc.HasNoKey(); });

        modelBuilder.HasDbFunction(() =>
GetDownloadedLiteratureByReaderId(default));
        modelBuilder.HasDbFunction(() =>
GetReviewViewByLiteratureID(default));
    }
    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Data Source=DESKTOP-8F2BRIN;Initial
Catalog=VKR;Integrated Security=True; TrustServerCertificate=True;
Trusted_Connection=True;");
    }
}
}

```

Листинг Г.2 – Файл Author.cs

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServer.Models
{
    [Table("Author")]
    public class Author
    {
        [Column("AuthorID")]
        public int AuthorID { get; set; }
        [Column("FIO")]
        public string? FIO { get; set; }
    }
}

```

Листинг Г.3 – Файл DownloadedLiterature.cs

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServer.Models
{
    [Table("DownloadedLiterature")]
    public class DownloadedLiterature
    {
        [Column("DownloadID")]

        public int DownloadID { get; set; }
        [Column("ReaderID")]

        public int ReaderID { get; set; }
        [Column("LiteratureID")]

        public int LiteratureID { get; set; }
    }
}

```

Листинг Г.4 – Файл DownloadList.cs

```
namespace WebServer.Models
{
    public class DownloadList
    {
        public int LiteratureID { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public string Genre { get; set; }
        public int? Rating { get; set; }
    }
}
```

Листинг Г.5 – Файл Genre.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServer.Models
{
    [Table("Genre")]
    public class Genre
    {
        [Column("GenreID")]

        public int GenreID { get; set; }

        [Column("Name")]
        public string Name { get; set; }
    }
}
```

Листинг Г.6 – Файл Literature.cs

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServer.Models
{
    [Table("Literature")]
    public class Literature
    {
        [Column("LiteratureID")]

        public int LiteratureID { get; set; }
        [Column("Title")]
        public string Title { get; set; }
        [Column("AuthorID")]

        public int AuthorID { get; set; }
    }
}
```

```

        [Column("GenreID")]
        public int GenreID { get; set; }
        [Column("File_data")]
        public byte[]? File_data { get; set; }
    }
}

```

Листинг Г.7 – Файл LiteratureList.cs

```

using System.Reflection;

namespace WebServer.Models
{
    public class LiteratureList
    {
        public int LiteratureID { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public string Genre { get; set; }
        public int? Rating { get; set; }
    }
}

```

Листинг Г.8 – Файл ReaderProfile.cs

```

using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServer.Models
{
    [Table("ReaderProfile")]
    public class ReaderProfile
    {
        [Column("ReaderID")]
        [Key] public int ReaderID { get; set; }
        [Column("FIO")]
        public string? FIO { get; set; }
        [Column("Email")]
        public string? Email { get; set; }

        [Column("Password")]
        public string? Password { get; set; }
    }
}

```

Листинг Г.9 – Файл Review.cs

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace WebServer.Models
{
    [Table("Review")]
    public class Review
    {
        [Column("ReviewID")]

        public int ReviewID { get; set; }
        [Column("ReaderID")]

```

```

        public int ReaderID { get; set; }
        [Column("LiteratureID")]

        public int LiteratureID { get; set; }
        [Column("Rating")]
        public int Rating { get; set; }
        [Column("Comment")]
        public string Comment { get; set; }
    }
}

```

Листинг Г.10 – Файл ReviewList.cs

```

namespace WebServer.Models
{
    public class ReviewList
    {
        public int LiteratureID { get; set; }
        public string FIO { get; set; }
        public string Comment { get; set; }
        public int Rating { get; set; }
    }
}

```

Листинг Г.11 – Файл DownloadedLiteratureController.cs

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using Microsoft.EntityFrameworkCore;
using WebServer.Models;

namespace WebServer.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class DownloadedLiteratureController : ControllerBase
    {
        private readonly ApplicationDbContext db;
        public DownloadedLiteratureController(ApplicationContext context) { db
= context; }

        [HttpGet]
        [Route("Get")]
        public async Task<ActionResult<List<DownloadList>>> Get(int id)
        {
            return db.GetDownloadedLiteratureByReaderId(id).ToList();
        }
        [HttpGet]
        [Route("GetSortName")]
        public async Task<ActionResult<List<DownloadList>>> GetSortName(int
id)
        {
            return db.DownloadLists.FromSqlRaw($"SELECT * FROM
GetDownloadedLiteratureByReaderId({id}) ORDER BY Auth;").ToList();
        }
        [HttpGet]
        [Route("GetSortAuth")]
    }
}

```

```

        public async Task<ActionResult<List<DownloadList>>> GetSortAuth(int
id)
        {
            return db.DownloadLists.FromSqlRaw($"SELECT * FROM
GetDownloadedLiteratureByReaderId({id}) ORDER BY Auth;").ToList();
        }
        [HttpGet]
        [Route("GetSortMark")]
        public async Task<ActionResult<List<DownloadList>>> GetSortMark(int
id)
        {
            return db.DownloadLists.FromSqlRaw($"SELECT * FROM
GetDownloadedLiteratureByReaderId({id}) ORDER BY Auth;").ToList();
        }
        [HttpGet]
        [Route("LitGet")]
        public async Task<ActionResult<LiteratureList>> LitGet(string ch)
        {
            string[] l = ch.Split(new char[] { '|' },
StringSplitOptions.RemoveEmptyEntries);
            string id = l[0];
            LiteratureList literature = db.LiteratureLists.FirstOrDefault(x =>
x.LiteratureID == Int32.Parse(id));
            if (literature == null) { return NotFound(); }
            return new ObjectResult(literature);
        }
        [HttpDelete]
        [Route("Delete")]
        public async Task<ActionResult> Deleted(string str)
        {
            string[] l = str.Split(new char[] { '|' },
StringSplitOptions.RemoveEmptyEntries);
            string LitId = l[0];
            string ReadId = l[1];
            DownloadedLiterature downloadedLiterature =
db.downloadedliteratures.FirstOrDefault(x => (x.LiteratureID ==
Int32.Parse(LitId)) && (x.ReaderID == Int32.Parse(ReadId)));
            if (downloadedLiterature != null) {
                db.downloadedliteratures.Remove(downloadedLiterature);
                db.SaveChanges();
                return Ok(1);
            }
            return Ok(0);
        }
        [HttpGet]
        [Route("LitCheck")]
        public async Task<ActionResult> LitCheck(string ch)
        {
            string[] l = ch.Split(new char[] { '|' },
StringSplitOptions.RemoveEmptyEntries);
            string LitId = l[0];
            string ReadId = l[1];
            DownloadedLiterature downloadedLiterature =
db.downloadedliteratures.FirstOrDefault(x => (x.LiteratureID ==
Int32.Parse(LitId)) && (x.ReaderID == Int32.Parse(ReadId)));
            if (downloadedLiterature != null) { return Ok(1); }
            return Ok(0);
        }
    }
}

```

Листинг Г.12 – Файл LiteratureController.cs

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using WebServer.Models;

namespace WebServer.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LiteratureController : ControllerBase
    {
        private readonly ApplicationDbContext db;
        public LiteratureController(ApplicationDbContext context) { db = context; }

        [HttpGet]
        [Route("Get")]
        public async Task<ActionResult<List<LiteratureList>>> Get()
        {
            return db.LiteratureLists.ToList();
        }

        [HttpGet]
        [Route("GetSortName")]
        public async Task<ActionResult<List<LiteratureList>>> GetSortName()
        {
            return db.LiteratureLists.FromSqlRaw("SELECT * FROM LiteratureView
ORDER BY Title;").ToList();
        }

        [HttpGet]
        [Route("GetSortAuth")]
        public async Task<ActionResult<List<LiteratureList>>> GetSortAuth()
        {
            return db.LiteratureLists.FromSqlRaw("SELECT * FROM LiteratureView
ORDER BY Author;").ToList();
        }

        [HttpGet]
        [Route("GetSortMark")]
        public async Task<ActionResult<List<LiteratureList>>> GetSortMark()
        {
            return db.LiteratureLists.FromSqlRaw("SELECT * FROM LiteratureView
ORDER BY Rating;").ToList();
        }

        [HttpGet]
        [Route("LitGet")]
        public async Task<ActionResult<Literature>> LitGet(int id)
        {
            LiteratureList literature = db.LiteratureLists.FirstOrDefault(x =>
x.LiteratureID == id);
            if (literature == null) { return NotFound(); }
            return new ObjectResult(literature);
        }

        [HttpGet]
        [Route("Download")]
        public async Task<ActionResult<byte[]>> Download(string input)
        {
            string[] l = input.Split(new char[] { '|' },
StringSplitOptions.RemoveEmptyEntries);
            string rid = l[0];
            string lid = l[1];

```

```

        int readerid = int.Parse(rid);
        int id = int.Parse(lid);
        int downlid = db.downloadedliteratures.Max(x => x.DownloadID) + 1;
        Literature literature = db.literatures.FirstOrDefault(x =>
x.LiteratureID == id);
        if (literature == null) { return NotFound(); }
        DownloadedLiterature downloadedLiterature = new
DownloadedLiterature();
        downloadedLiterature.LiteratureID = id;
        downloadedLiterature.ReaderID = readerid;
        downloadedLiterature.DownloadID = downlid;
        db.downloadedliteratures.Add(downloadedLiterature);
        db.SaveChanges();
        return new ObjectResult(literature.File_data);
    }
}
}

```

Листинг Г.13 – Файл ReaderProfileController.cs

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Infrastructure;
using Newtonsoft.Json;
using System.Diagnostics.Eventing.Reader;
using System.Runtime.Serialization;
using System.Text.Json.Serialization;
using WebServer.Models;

namespace WebServer.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ReaderProfileController : ControllerBase
    {
        private readonly ApplicationContext db;
        public ReaderProfileController(ApplicationContext context) { db =
context; }
        [HttpGet]
        [Route("Get")]
        public async Task<ActionResult<ReaderProfile>> Get(int id)
        {
            ReaderProfile readerProfile = db.readerprofiles.FirstOrDefault(x
=> x.ReaderID == id);
            if (readerProfile == null) { return NotFound(); }
            return new ObjectResult(readerProfile);
        }

        [HttpPost]
        [Route("Authorization")]
        public async Task<ActionResult> PostAuthorization(string input//,
string password)
        {
            try
            {
                var Reader = new StreamReader(Request.Body);
                var content = await Reader.ReadToEndAsync();
                string[] l = input.Split(new char[] { '|' },
StringSplitOptions.RemoveEmptyEntries);
                string email = l[0];
                string password = l[1];
            }
        }
    }
}

```

```

        if (email == null || password == null) { return
BadRequest("Введите необходимые данные"); }
        var reader = db.readerprofiles.FirstOrDefault(x => (x.Email ==
email)&&(x.Password == password));
        if (reader == null) { return BadRequest("Пользователь не
обнаружен"); }
        else
        {
            return Ok(reader.ReaderID.ToString());
        }
    }
    catch { return BadRequest(); }
}
[HttpPost]
[Route("Registration")]
public async Task<ActionResult> PostRegistration(ReaderProfile
readerProfile)
{
    try
    {
        var Reader = new StreamReader(Request.Body);
        var content = await Reader.ReadToEndAsync();

        if(readerProfile == null || readerProfile.Email == null ||
readerProfile.FIO == null || readerProfile.Password == null) { return
BadRequest("Введите запрошенные данные"); }
        string email = readerProfile.Email;
        readerProfile.ReaderID = db.readerprofiles.Max(x =>
x.ReaderID) + 1; ;
        var reader = db.readerprofiles.FirstOrDefault(x => x.Email ==
email);

        if (reader != null) { return BadRequest("Email занят"); }

        db.readerprofiles.Add(readerProfile);
        db.SaveChanges();
        return Ok(readerProfile.ReaderID.ToString());
    }
    catch
    {
        return BadRequest();
    }
}
}
}

```

Листинг Г.14 – Файл ReviewController.cs

```

using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using WebServer.Models;

namespace WebServer.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ReviewController : ControllerBase
    {
        private readonly ApplicationContext db;
        public ReviewController(ApplicationContext context) { db = context; }
        [HttpGet]
        [Route("Get")]
    }
}

```

```

        public async Task<ActionResult<List<ReviewList>>> Get(int
LiteratureId)
        {
            db.GetReviewViewByLiteratureID(LiteratureId).ToList();
            return db.GetReviewViewByLiteratureID(LiteratureId).ToList();
        }
        [HttpPost]
        [Route("Post")]
        public async Task<ActionResult> PostReview (Review review)
        {
            try
            {
                if (review == null || review.LiteratureID == null ||
review.ReaderID == null || review.Rating == null || review.Comment == null ) {
return BadRequest("Введите запрошенные данные"); } // Проверка на корректность
полученных данных
                var check = db.reviews.FirstOrDefault(x => (x.ReaderID ==
review.ReaderID) && (x.LiteratureID == review.LiteratureID)); // Поиск
комментария пользователя к указанной литературе
                if (check != null) // Если комментарий найден, то заменяем его
                {
                    check.Comment = review.Comment;
                    check.Rating = review.Rating;
                    db.SaveChanges();
                }
                else // Если комментарий отсутствует, то добавляем его
                {
                    review.ReviewID = db.reviews.Max(x => x.ReviewID)+1;
                    db.reviews.Add(review);
                    db.SaveChanges();
                }
                return Ok(1); // Успешное завершение
            }
            catch
            {
                return BadRequest(); // Завершение с ошибкой
            }
        }
    }
}

```

Приложение Д
ИСХОДНЫЙ КОД МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Д.1 – Файл Book.cs

```
using System;
using System.Collections.Generic;

namespace VKR.Models
{
    public class Book
    {
        public string LiteratureID { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public string Rating { get; set; }
        public string Genre { get; set; }
        public Book() { }
        public Book(string id, string n, string a, string g, string m)
        {
            LiteratureID = id;
            Title = n;
            Author = a;
            Rating = m;
            Genre = g;
        }
    }
}
```

Листинг Д.2 – Файл Comment.cs

```
using System;

namespace VKR.Models
{
    public class Comm
    {
        public string Id { get; set; }
        public string LiteratureID { get; set; }
        public string UserID { get; set; }
        public string FIO { get; set; }
        public string Rating { get; set; }
        public string Comment { get; set; }

        public Comm() { }
        public Comm(string t, string d, string txt)
        {
            //Id = id;
            FIO = t;
            Rating = d;
            Comment = txt;
        }
    }
}
```

Листинг Д.3 – Файл Liter.cs

```
using System;
using System.Collections.Generic;

namespace VKR.Models
{
    public class Liter
    {

```

```

        public string LiteratureID { get; set; }
        public string Title { get; set; }
        public string Author { get; set; }
        public string Rating { get; set; }
        public string Genre { get; set; }
        public string Description { get; set; }
        public List<Comm> Comments { get; set; }
        public byte[] Content { get; set; }
        public Liter() { }
        public Liter(string id, string n, string a, string g, string m)
        {
            LiteratureID = id;
            Title = n;
            Author = a;
            Rating = m;
            Genre = g;
            Comments= new List<Comm>();
        }
    }
}

```

Листинг Д.4 – Файл Review.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace VKR.Models
{
    internal class Review
    {
        public int ReviewID { get; set; }

        public int ReaderID { get; set; }

        public int LiteratureID { get; set; }
        public int Rating { get; set; }
        public string Comment { get; set; }
    }
}

```

Листинг Д.5 – Файл User.cs

```

using System;
using System.Collections.Generic;

namespace VKR.Models
{
    public class User
    {
        public int ReaderID { get; set; }
        public string FIO { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }

        public User(int id, string name, string email, string password)
        {
            ReaderID = id;
            FIO = name;
            Email = email;
            Password = password;
        }
    }
}

```

```

    }
}

```

Листинг Д.6 – Файл MainViewModel.cs

```

using System;
using System.Windows.Input;
using VKR.Views;
using Xamarin.Essentials;
using Xamarin.Forms;

namespace VKR.ViewModels
{
    public class AboutViewModel : BaseViewModel
    {
        public AboutViewModel()
        {
            Title = "Главная страница";
        }

        public ICommand OpenWebCommand { get; }
    }
}

```

Листинг Д.7 – Файл MainPage.xaml.cs

```

using System;
using System.ComponentModel;
using VKR.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace VKR.Views
{
    public partial class AboutPage : ContentPage
    {
        public AboutPage()
        {
            InitializeComponent();
        }

        protected override void OnAppearing()
        {
        }

        private async void Button_Clicked1(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new ItemsPage());
        }

        private async void Button_Clicked2(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new DownloadList());
        }

        private async void Button_Clicked3(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new LoginPage());
        }
    }
}

```

Листинг Д.8 – Файл MainPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="VKR.Views.AboutPage"
    xmlns:vm="clr-namespace:VKR.ViewModels"
    Title="{Binding Title}"
    BackgroundColor="#96dlff">

    <ContentPage.BindingContext>
        <vm:AboutViewModel />
    </ContentPage.BindingContext>

    <ContentPage.Resources>
        <ResourceDictionary>
            <Color x:Key="Accent">#96dlff</Color>
        </ResourceDictionary>
    </ContentPage.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>

        <ScrollView Grid.Row="1">
            <StackLayout Orientation="Vertical" Padding="30,24,30,24"
                Spacing="10">

                <Label FontSize="16" Padding="0,24,0,0">
                    <Label.FormattedText>
                        <FormattedString>
                            <FormattedString.Spans>

                                </FormattedString.Spans>
                            </FormattedString>
                        </Label.FormattedText>
                    </Label>
                    <Button Margin="0,10,0,0" Text="Открыть электронный каталог"
                        Clicked="Button_Clicked1"
                        BackgroundColor="{StaticResource Primary}"
                        TextColor="White" />
                    <Button Margin="0,10,0,0" Text="Открыть список скаченной
литературы"
                        Clicked="Button_Clicked2"
                        BackgroundColor="{StaticResource Primary}"
                        TextColor="White" />
                    <Button Margin="0,10,0,0" Text="Выйти из профиля"
                        Clicked="Button_Clicked3"
                        BackgroundColor="{StaticResource Primary}"
                        TextColor="White" />
                </StackLayout>
            </ScrollView>
        </Grid>

    </ContentPage>
```

Листинг Д.9 – Файл BaseViewModel.cs

```
using System;
```

```

using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using VKR.Models;
using VKR.Services;
using Xamarin.Forms;

namespace VKR.ViewModels
{
    public class BaseViewModel : INotifyPropertyChanged
    {
        //public IDataStore<Item> DataStore =>
        DependencyService.Get<IDataStore<Item>>();

        bool isBusy = false;
        public bool IsBusy
        {
            get { return isBusy; }
            set { SetProperty(ref isBusy, value); }
        }

        string title = string.Empty;
        public string Title
        {
            get { return title; }
            set { SetProperty(ref title, value); }
        }

        protected bool SetProperty<T>(ref T backingStore, T value,
            [CallerMemberName] string propertyName = "",
            Action onChanged = null)
        {
            if (EqualityComparer<T>.Default.Equals(backingStore, value))
                return false;

            backingStore = value;
            onChanged?.Invoke();
            OnPropertyChanged(propertyName);
            return true;
        }

        #region INotifyPropertyChanged
        public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged([CallerMemberName] string
propertyName = "")
        {
            var changed = PropertyChanged;
            if (changed == null)
                return;

            changed.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
        #endregion
    }
}

```

Листинг Д.10 – Файл DownloadedViewModel.cs

```

//using Microsoft.Bot.Builder.Integration.AspNet.Core;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;

```

```

using System.Diagnostics;
using System.Linq;
using System.Net.Http.Headers;
using System.Net.Http;
using System.Threading.Tasks;
using VKR.Models;
using VKR.Views;
using Xamarin.Forms;

namespace VKR.ViewModels
{
    public class DownloadedViewModel : BaseViewModel
    {
        private Book _selectedItem;

        public ObservableCollection<Book> Items { get; }
        public ObservableCollection<Book> Books { get; }
        public Command LoadItemsCommand { get; }
        public Command AddItemCommand { get; }
        public Command<Book> ItemTapped { get; }

        public DownloadedViewModel()
        {
            Title = "Загруженная литература";
            Items = new ObservableCollection<Book>();
            Books = new ObservableCollection<Book>();
            LoadItemsCommand = new Command(() => ExecuteLoadItemsCommand());

            ItemTapped = new Command<Book>(OnItemSelected);
        }

        public async void ExecuteLoadItemsCommand()
        {
            IsBusy = true;

            try
            {
                Items.Clear();
                HttpResponseMessage response = await
App.client.GetAsync($"api/DownloadedLiterature/Get?id={App.iduser}");

                response.EnsureSuccessStatusCode();

                List<Book> who = await
response.Content.ReadAsAsync<List<Book>>();
                foreach (Book b in who)
                {
                    Items.Add(b);
                }
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex);
            }
            finally
            {
                IsBusy = false;
                foreach (var item in Items) { Books.Add(item); }
            }
        }
    }
}

```

```

public void OnAppearing()
{
    IsBusy = true;
    SelectedItem = null;
}

public Book SelectedItem
{
    get => _selectedItem;
    set
    {
        SetProperty(ref _selectedItem, value);
        OnItemSelected(value);
    }
}

async void OnItemSelected(Book item)
{
    if (item == null)
        return;

    // This will push the ItemDetailPage onto the navigation stack
    await
Shell.Current.GoToAsync($"{nameof(ItemDetailPage)}?{nameof(ItemDetailViewModel
1.ItemId)}={item.LiteratureID}");
    //
    await
Shell.Current.GoToAsync($"{nameof(ItemDetailPage)}?{nameof(ItemDetailViewModel
1.ItemId)}={item.Id}");
}

public async Task SortNameAsync()
{
    Items.Clear();
    Books.Clear();
    HttpResponseMessage response = await
App.client.GetAsync($"api/DownloadedLiterature/GetSortName?id={App.iduser}");

    response.EnsureSuccessStatusCode();

    List<Book> who = await response.Content.ReadAsAsync<List<Book>>();
    foreach (Book b in who)
    {
        Items.Add(b);
    }
    foreach (var item in Items) { Books.Add(item); }
}

public async Task SortAuthAsync()
{
    Items.Clear();
    Books.Clear();
    HttpResponseMessage response = await
App.client.GetAsync($"api/DownloadedLiterature/GetSortAuth?id={App.iduser}");

    response.EnsureSuccessStatusCode();

    List<Book> who = await response.Content.ReadAsAsync<List<Book>>();
    foreach (Book b in who)
    {
        Items.Add(b);
    }
    foreach (var item in Items) { Books.Add(item); }
}

public async Task SortMarkAsync()

```

```

        {
            Items.Clear();
            Books.Clear();
            HttpResponseMessage response = await
App.client.GetAsync($"api/DownloadedLiterature/GetSortMark?id={App.iduser}");

            response.EnsureSuccessStatusCode();

            List<Book> who = await response.Content.ReadAsAsync<List<Book>>();
            foreach (Book b in who)
            {
                Items.Add(b);
            }
            foreach (var item in Items) { Books.Add(item); }
        }

public void SearchBook(string s)
{
    if (s == "") {
        Items.Clear();
        foreach (Book book in Books)
        {
            Items.Add(book);
        }
        return;
    }
    foreach (Book book in Items)
    {
        if (!(book.Title.Contains(s)) && (!book.Author.Contains(s))) {
            Items.Remove(book);
        }
    }
}
}
}

```

Листинг Д.11 – Файл ItemDetailViewModel.cs

```

using Java.Util;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Linq;
using System.Net.Http;
using System.Threading.Tasks;
using VKR.Models;
using Xamarin.Forms;
using static Android.Content.ClipData;
using static Android.Telephony.CarrierConfigManager;

namespace VKR.ViewModels
{
    [QueryProperty(nameof(ItemId), nameof(ItemId))]
    public class ItemDetailViewModel : BaseViewModel
    {
        private string itemId;
        private string text1;
        private string text2;
        private string text3;
        private string text4;
        private string description;
        public Liter Book1;
    }
}

```

```

public string Id { get; set; }
// public string Name { get; set; }
public string Name
{
    get => text1;
    set => SetProperty(ref text1, value);
}

// public string Auth { get; set; }
public string Auth
{
    get => text2;
    set => SetProperty(ref text2, value);
}
// public string Gen { get; set; }
public string Gen
{
    get => text3;
    set => SetProperty(ref text3, value);
}
//public string Rat { get; set; }
public string Rat
{
    get => text4;
    set => SetProperty(ref text4, value);
}
private string textButton;
public string TextButton
{
    get => textButton;
    set => SetProperty(ref textButton, value);
}
// public string Desc { get; set; }
public ObservableCollection<Comm> Com { get; }
public List<Comm> c;
public Command LoadItemCommand { get; }

/* public string Text
{
    get => text;
    set => SetProperty(ref text, value);
}

public string Description
{
    get => description;
    set => SetProperty(ref description, value);
}*/

public string ItemId
{
    get
    {
        return itemId;
    }
    set
    {
        itemId = value;
        fill();
    }
}

public async Task fill() {
    IsBusy = true;

```

```

        HttpResponseMessage response2 = await
App.client.GetAsync($"api/Literature/LitGet?id={Int32.Parse(itemId)}");

        response2.EnsureSuccessStatusCode();

        Book who = await response2.Content.ReadAsAsync<Book>();
        Auth = who.Author;
        Name = who.Title;
        Gen = who.Genre;
        Rat = who.Rating;
        //Desc = who.

        string str = itemId + "|" + App.iduser.ToString();
        HttpResponseMessage response = await
App.client.GetAsync($"api/DownloadedLiterature/LitCheck?ch={str}");
        response.EnsureSuccessStatusCode();
        int check = -5;
        check = await response.Content.ReadAsAsync<int>();

        if (check == 0)
        {
            TextButton = "Скачать";
        }
        else
        {
            TextButton = "Открыть";
        }

        HttpResponseMessage response3 = await
App.client.GetAsync($"api/Review/Get?LiteratureId={itemId}");
        response3.EnsureSuccessStatusCode();

        c = await response3.Content.ReadAsAsync<List<Comm>>();
        Com.Clear();
        foreach (Comm co in c)
        {
            Com.Add(co);
        }
        IsBusy = false;
    }
    public ItemDetailViewModel()
    {
        Com = new ObservableCollection<Comm>();
        LoadItemCommand = new Command(async () => await fill());
    }

    public async void LoadItemId(string itemId)
    {
        try
        {
            foreach (Comm co in c)
            {
                Com.Add(co);
            }
        }
        catch (Exception)
        {
            Debug.WriteLine("Failed to Load Item");
        }
    }
}

```

```

    }
}

```

Листинг Д.12 – Файл ItemsViewModel.cs

```

using System;
using System.Collections.ObjectModel;
using System.Diagnostics;
using System.Net.Http;
using System.Threading.Tasks;
using VKR.Models;
using VKR.Views;
using Xamarin.Forms;
using System.Collections;
using System.Collections.Generic;

namespace VKR.ViewModels
{
    public class ItemsViewModel : BaseViewModel
    {
        private Book _selectedItem;

        public ObservableCollection<Book> Items { get; }
        public ObservableCollection<Book> Books { get; }
        public Command LoadItemsCommand { get; }
        public Command AddItemCommand { get; }
        public Command<Book> ItemTapped { get; }

        public ItemsViewModel()
        {
            Title = "Электронный каталог";
            Items = new ObservableCollection<Book>();
            Books = new ObservableCollection<Book>();
            LoadItemsCommand = new Command(async () => await
ExecuteLoadItemsCommand());

            ItemTapped = new Command<Book>(OnItemSelected);

            AddItemCommand = new Command(OnAddItem);
        }

        async Task ExecuteLoadItemsCommand()
        {
            IsBusy = true;

            try
            {
                Items.Clear();
                // User u = new User(0, fio.Text, email.Text, login.Text,
password.Text);
                // HttpResponseMessage resp = await App.client.PostAsync();
                // HttpResponseMessage response = await
App.client.PostAsync($"api/ReaderProfile/Authorization?login={email.Text}&pas
sword={password.Text}");

                HttpResponseMessage response = await
App.client.GetAsync($"api/Literature/Get");

                response.EnsureSuccessStatusCode();

                List<Book> who = await
response.Content.ReadAsAsync<List<Book>>();
                foreach (Book b in who)

```

```

        {
            Items.Add(b);
        }
    }
    catch (Exception ex)
    {
        Debug.WriteLine(ex);
    }
    finally
    {
        IsBusy = false;
        foreach (var item in Items) { Books.Add(item); }
    }
}

public void OnAppearing()
{
    IsBusy = true;
    SelectedItem = null;
}

public Book SelectedItem
{
    get => _selectedItem;
    set
    {
        SetProperty(ref _selectedItem, value);
        OnItemSelected(value);
    }
}

private async void OnAddItem(object obj)
{
    await Shell.Current.GoToAsync(nameof(NewItemPage));
}

async void OnItemSelected(Book item)
{
    if (item == null)
        return;

    // This will push the ItemDetailPage onto the navigation stack
    await
Shell.Current.GoToAsync($"{nameof(ItemDetailPage)}?{nameof(ItemDetailViewModel
1.ItemId)}={item.LiteratureID}");
}

public async Task SortNameAsync()
{
    Items.Clear();
    Books.Clear();
    HttpResponseMessage response = await
App.client.GetAsync($"api/Literature/GetSortName");

    response.EnsureSuccessStatusCode();

    List<Book> who = await response.Content.ReadAsAsync<List<Book>>();
    foreach (Book b in who)
    {
        Items.Add(b);
    }
    foreach (var item in Items) { Books.Add(item); }
}

```

```

public async Task SortAuthAsync()
{
    Items.Clear();
    Books.Clear();
    HttpResponseMessage response = await
App.client.GetAsync($"api/Literature/GetSortAuth");

    response.EnsureSuccessStatusCode();

    List<Book> who = await response.Content.ReadAsAsync<List<Book>>();
    foreach (Book b in who)
    {
        Items.Add(b);
    }
    foreach (var item in Items) { Books.Add(item); }
}

public async Task SortMarkAsync()
{
    Items.Clear();
    Books.Clear();
    HttpResponseMessage response = await
App.client.GetAsync($"api/Literature/GetSortMark");

    response.EnsureSuccessStatusCode();

    List<Book> who = await response.Content.ReadAsAsync<List<Book>>();
    foreach (Book b in who)
    {
        Items.Add(b);
    }
    foreach (var item in Items) { Books.Add(item); }
}

public void SortCom()
{
}

public void SearchBook(string s)
{
    if (s == "")
    {
        Items.Clear();
        foreach (Book book in Books)
        {
            Items.Add(book);
        }
        return;
    }
    foreach (Book book in Books)
    {
        if (! (book.Title.ToUpper().Contains(s.ToUpper())) &&
(!book.Author.ToUpper().Contains(s.ToUpper())))
        {
            Items.Remove(book);
        }
    }
}
}

```

Листинг Д.13 – Файл DownloadList.xaml.cs

```

using System;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Threading.Tasks;
using VKR.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace VKR.Views
{
    public partial class DownloadList : ContentPage
    {
        DownloadedViewModel _viewModel;

        public DownloadList()
        {
            InitializeComponent();
            this.BindingContext = _viewModel = new DownloadedViewModel();
        }

        async void Handle_ItemTapped(object sender, ItemTappedEventArgs e)
        {
            if (e.Item == null)
                return;

            await DisplayAlert("Item Tapped", "An item was tapped.", "OK");

            //Deselect Item
            ((ListView)sender).SelectedItem = null;
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            _viewModel.OnAppearing();
        }

        private void Picker_SelectedIndexChanged(object sender, EventArgs e)
        {
            Picker picker = (Picker)sender;
            switch (picker.SelectedIndex)
            {
                case 0:
                    _viewModel.SortAuthAsync();
                    break;

                case 1:
                    _viewModel.SortNameAsync();
                    break;

                case 2:
                    _viewModel.SortMarkAsync();
                    break;

                default:
                    break;
            }
        }
    }
}

```

```

private void SearchBar_TextChanged(object sender, TextChangedEventArgs
e)
{
    SearchBar searchBar = (SearchBar)sender;
    string search = searchBar.Text;
    // DisplayAlert("Уведомление", search, "Ок");
    _viewModel.SearchBook(search);
}
}
}

```

Листинг Д.14 – Файл DownloadList.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewmodels="clr-namespace:VKR.ViewModels"
    xmlns:model="clr-namespace:VKR.Models"
    x:DataType="viewmodels:DownloadedViewModel"
    xmlns:local="clr-namespace:VKR.ViewModels"
    BackgroundColor="#96d1ff"
    x:Class="VKR.Views.DownloadList">

    <RefreshView x:DataType="local:DownloadedViewModel" Command="{Binding
LoadItemsCommand}" IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
        <StackLayout>
            <SearchBar TextChanged="SearchBar_TextChanged" />
            <Picker Title="Сортировать:"
                TitleColor="Black"
                HorizontalTextAlignment="Center"
                VerticalTextAlignment="Center"
                SelectedIndexChanged="Picker_SelectedIndexChanged"
                >

                <Picker.ItemsSource>
                    <x:Array Type="{x:Type x:String}">
                        <x:String>По автору</x:String>
                        <x:String>По названию</x:String>
                        <x:String>По рейтингу</x:String>
                    </x:Array>
                </Picker.ItemsSource>
            </Picker>

            <CollectionView x:Name="DownloadedView"
                ItemsSource="{Binding Items}"
                SelectionMode="None">
                <CollectionView.ItemTemplate>
                    <DataTemplate>
                        <StackLayout Padding="10" x:DataType="model:Book">
                            <Label Text="{Binding Title}"
                                LineBreakMode="NoWrap"
                                Style="{DynamicResource ListItemTextStyle}"
                                FontSize="16" />
                            <Label Text="{Binding Author}"
                                LineBreakMode="NoWrap"
                                Style="{DynamicResource ListItemDetailTextStyle}"
                                FontSize="13" />
                            <Label Text="{Binding Rating}"
                                LineBreakMode="NoWrap"
                                Style="{DynamicResource ListItemDetailTextStyle}"
                                FontSize="13" />
                        </StackLayout.GestureRecognizers>
                    </DataTemplate>
                </CollectionView.ItemTemplate>
            </CollectionView>
        </StackLayout>
    </RefreshView>
</ContentPage>

```



```

        ColumnDefinitions="Auto, *"
        BackgroundColor="Azure">
            <Label Text="{Binding FIO}"
VerticalTextAlignment="Center" FontSize="Small" Grid.Column="0"
Grid.ColumnSpan="2"/>
            <Label Text="{Binding Rating}"
FontSize="Small" Grid.Column="2" VerticalTextAlignment="Center"
Grid.ColumnSpan="1"/>
            <!-- <Label Text="{Binding Comment}" FontSize="Small"
Grid.Column="4" Grid.ColumnSpan="2"/> -->
            <Label VerticalTextAlignment="Center"
Text="{Binding Comment}" FontSize="Small" Grid.Column="3"
Grid.ColumnSpan="2"/>
        </Grid>
    </StackLayout>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>

</StackLayout>
</RefreshView>
</ContentPage>

```

Листинг Д.16 – Файл ItemDetailPage.xaml.cs

```

using Android.Content;
using Android.Graphics.Pdf;
using Newtonsoft.Json.Linq;
using Plugin.Permissions;
using Plugin.Permissions.Abstractions;
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using VKR.Models;
using VKR.ViewModels;
using Xamarin.Essentials;
using Xamarin.Forms;

namespace VKR.Views
{
    public partial class ItemDetailPage : ContentPage
    {
        ItemDetailViewModel vm;
        Context context;
        public interface IFileHelper
        {
            Task<string> SaveFileAsync(string fileName, byte[] fileBytes);
            void ShowPDFTXTFromLocal(string filename);
        };

        public ItemDetailPage()
        {
            InitializeComponent();
            BindingContext = new ItemDetailViewModel();
            BindingContext = vm = new ItemDetailViewModel();
        }

        private async Task DownloadAsync(int id)
        {

```

```

        string str = App.iduser + "|" + id;
        HttpResponseMessage response = await
App.client.GetAsync($"api/Literature/Download?input={str}");
        response.EnsureSuccessStatusCode();
        byte[] files = await response.Content.ReadAsAsync<byte[]>();
        string filename = vm.Name + ".pdf";

        var status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Storage);

        if (status !=
Plugin.Permissions.Abstractions.PermissionStatus.Granted)
        {
            if (await
CrossPermissions.Current.ShouldShowRequestPermissionRationaleAsync(Permission
.Storage))
            {
                await
Application.Current.MainPage.DisplayAlert("Permission Required", "Storage
permission is required to save the PDF file.", "OK");
            }

            var results = await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Storage);

            if (results.ContainsKey(Permission.Storage) &&
results[Permission.Storage] !=
Plugin.Permissions.Abstractions.PermissionStatus.Granted)
            {
                await
Application.Current.MainPage.DisplayAlert("Permission Denied", "Unable to save
the PDF file without storage permission.", "OK");
                return;
            }
        }

        var downloadsPath =
Android.OS.Environment.GetExternalStoragePublicDirectory(Android.OS.Environment
.DirectoryDownloads).AbsolutePath;
        var filePath = Path.Combine(downloadsPath, filename);

        using (var fileStream = new FileStream(filePath,
FileMode.OpenOrCreate))
        {
            fileStream.Write(files, 0, files.Length);
        }

        await Application.Current.MainPage.DisplayAlert("File Saved",
$"The PDF file has been saved to {filePath}.", "OK");
    }
    private async void Funct1()
    {
        await DownloadAsync(int.Parse(vm.ItemId));
    }
    public async void OpenPdf(string uri)
    {
        await Launcher.OpenAsync(new OpenFileRequest { File = new
ReadOnlyFile(uri) });
    }

    private void Button_Clicked(object sender, System.EventArgs e)
    {

```

```

        if (vm.TextButton == "Скачать")
        {
            Funct1();// DownloadAsync(int.Parse(vm.ItemId));
        }
        else
        {
            }
            string filename = vm.Name + ".pdf";
            string filePathAndName =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal),
filename);
            OpenPdf(filePathAndName);
            // OpenPdf(filePathAndName);
            // OpenPdf();
        }

private async void Button_Clicked_1(object sender, EventArgs e)
{
    string result = "-1";
    string res = "";
    while (true)
    {
        result = await DisplayPromptAsync("Оценка", "Какую оценку от 1
до 10 хотите установить?", "OK", "Отменить", "Оценка", initialValue: "5",
keyboard: Keyboard.Numeric);
        if (result == "-1") { break; }
        if ((int.Parse(result) > 10) || (int.Parse(result) < 0)) { await
DisplayAlert("Введена некорректная оценка", result, "OK"); } else {
            while (true) {
                res = await DisplayPromptAsync("Отзыв", "Введите
отзыв", "OK", "Отменить", "Отзыв", initialValue: " ", keyboard: Keyboard.Text);
                if(res != " ") { break; }
            }
            break;
        }
    }
    if(res!=" " && result != "-1")
    {
        Review commennt = new Review();
        commennt.Comment = res;
        commennt.Rating = int.Parse(result);
        commennt.LiteratureID = int.Parse(vm.ItemId);
        commennt.ReaderID = App.iduser;
        commennt.ReviewID = 0;

        HttpResponseMessage response = await
App.client.PostAsJsonAsync($"api/Review/Post", commennt);

        response.EnsureSuccessStatusCode();
        int who = await response.Content.ReadAsAsync<int>();
    }
}

private async void Button_Clicked_2(object sender, EventArgs e)
{
    string str = vm.ItemId + "|" + App.iduser.ToString();
    HttpResponseMessage response = await
App.client.GetAsync($"api/DownloadedLiterature/LitCheck?ch={str}");
    response.EnsureSuccessStatusCode();
    int check = -5;
    check = await response.Content.ReadAsAsync<int>();
}

```

```

        if (check == 0)
        {
            await DisplayAlert("Файл не скачен", vm.Name, "OK");
        }
        else
        {
            response = await
App.client.DeleteAsync($"api/DownloadedLiterature/Delete?str={str}");
            response.EnsureSuccessStatusCode();
            await response.Content.ReadAsAsync<int>();
            string filename = vm.Name + ".pdf";
            var downloadsPath =
Android.OS.Environment.GetExternalStoragePublicDirectory(Android.OS.Environment.DirectoryDownloads).AbsolutePath;
            var filePath = Path.Combine(downloadsPath, filename);
            System.IO.File.Delete(filePath);
        }
    }
}
}

```

Листинг Д.17 – Файл ItemsPage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VKR.Models;
using VKR.ViewModels;
using VKR.Views;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace VKR.Views
{
    public partial class ItemsPage : ContentPage
    {
        ItemsViewModel _viewModel;

        public ItemsPage()
        {
            InitializeComponent();

            BindingContext = _viewModel = new ItemsViewModel();
            ///Routing.RegisterRoute(nameof(ItemDetailPage),
typeof(ItemDetailPage));
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            _viewModel.OnAppearing();
        }

        private void Picker_SelectedIndexChanged(object sender, EventArgs e)
        {
            Picker picker = (Picker)sender;
            switch (picker.SelectedIndex)
            {
                case 0:

```

```

        _viewModel.SortAuthAsync();
        break;

    case 1:
        _viewModel.SortNameAsync();
        break;

    case 2:
        _viewModel.SortMarkAsync();
        break;

    default:
        break;
    }
}

private void SearchBar_TextChanged(object sender, TextChangedEventArgs
e)
{
    SearchBar searchBar = (SearchBar)sender;
    string search = searchBar.Text;
    // DisplayAlert("Уведомление", search, "Ок");
    _viewModel.SearchBook(search);
}
}

```

Листинг Д.18 – Файл ItemsPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="VKR.Views.ItemsPage"
    Title="{Binding Title}"
    xmlns:local="clr-namespace:VKR.ViewModels"
    xmlns:model="clr-namespace:VKR.Models"
    x:Name="BrowseItemsPage"
    BackgroundColor="#96d1ff">

    <ContentPage.ToolbarItems>
        <!-- <ToolbarItem Text="Add" Command="{Binding AddItemCommand}" /> -->
    </ContentPage.ToolbarItems>
    <!--
        x:DataType enables compiled bindings for better performance and compile
        time validation of binding expressions.
        https://docs.microsoft.com/xamarin/xamarin-forms/app-fundamentals/data-
        binding/compiled-bindings
    -->
    <RefreshView x:DataType="local:ItemsViewModel" Command="{Binding
LoadItemsCommand}" IsRefreshing="{Binding IsBusy, Mode=TwoWay}">
        <StackLayout>
            <SearchBar TextChanged="SearchBar_TextChanged" />
            <Picker Title="Сортировать:"
                TitleColor="Black"
                HorizontalTextAlignment="Center"
                VerticalTextAlignment="Center"
                SelectedIndexChanged="Picker_SelectedIndexChanged"
            >

            <Picker.ItemsSource>
                <x:Array Type="{x:Type x:String}">
                    <x:String>По автору</x:String>

```

```

        <x:String>По названию</x:String>
        <x:String>По рейтингу</x:String>
    </x:Array>
</Picker.ItemsSource>
</Picker>

<CollectionView x:Name="ItemsListView"
    ItemsSource="{Binding Items}"
    SelectionMode="None">
<CollectionView.ItemTemplate>
    <DataTemplate>
        <StackLayout Padding="10" x:DataType="model:Book">

            <Label Text="{Binding Title}"
                LineBreakMode="NoWrap"
                Style="{DynamicResource ListItemTextStyle}"
                FontSize="16" />
            <Label Text="{Binding Author}"
                LineBreakMode="NoWrap"
                Style="{DynamicResource ListItemDetailTextStyle}"
                FontSize="13" />
            <Label Text="{Binding Rating}"
                LineBreakMode="NoWrap"
                Style="{DynamicResource ListItemDetailTextStyle}"
                FontSize="13" />
            <StackLayout.GestureRecognizers>
                <TapGestureRecognizer
                    NumberOfTapsRequired="1"
                    Command="{Binding Source={RelativeSource
AncestorType={x:Type local:ItemsViewModel}}, Path=ItemTapped}"
                    CommandParameter="{Binding .}" />
            </TapGestureRecognizer>
            </StackLayout.GestureRecognizers>
        </StackLayout>
    </DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</StackLayout>
</RefreshView>
</ContentPage>

```

Листинг Д.19 – Файл LoginPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:d="http://xamarin.com/schemas/2014/forms/design"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:local="clr-namespace:VKR"

    mc:Ignorable="d"
    x:Class="VKR.Views.LoginPage"
    Shell.NavBarIsVisible="False"
    BackgroundColor="#96d1ff">
<ContentPage.Content>
    <StackLayout Padding="10,0,10,0" VerticalOptions="Center">
        <Entry x:Name="email" Placeholder="Email"
            Keyboard="Email">
            <Entry.Behaviors>
                <local:EmailValidation/>
            </Entry.Behaviors>
    </StackLayout>
</ContentPage.Content>

```

```

        </Entry>
        <Entry IsPassword="True" x:Name="password" Placeholder="Минимум 8
СИМВОЛОВ, один заглавный и есть числа">
            <Entry.Behaviors>
                <local:PasswordValidation/>
            </Entry.Behaviors>
        </Entry>
        <Button VerticalOptions="Center" Text="Авторизация"
Clicked="Button_Clicked"/>
        <Button VerticalOptions="Center" Text="Регистрация"
Clicked="Button_Clicked2"/>

    </StackLayout>
</ContentPage.Content>
</ContentPage>

```

Листинг Д.20 – Файл LoginPage.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VKR.ViewModels;
using VKR;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Net.Http;
using VKR.Models;
using static System.Net.Mime.MediaTypeNames;

namespace VKR.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LoginPage : ContentPage
    {
        public LoginPage()
        {
            InitializeComponent();
        }

        private async void Button_Clicked(object sender, EventArgs e)
        {
            EmailValidation EmailValid = new EmailValidation();
            PasswordValidation PassValid = new PasswordValidation();
            try
            {
                if (email.Text != "" && password.Text != "")
                {
                    if (EmailValid.IsValidEmail(email.Text) &&
PassValid.IsValidPassword(password.Text))
                    {
                        // User u = new User(0, fio.Text, email.Text,
login.Text, password.Text);
                        // HttpResponseMessage resp = await
App.client.PostAsync();
                        //HttpResponseMessage response = await
App.client.PostAsync($"api/ReaderProfile/Authorization?login={email.Text}&pas
sword={password.Text}");
                        string str = email.Text + "|" + password.Text;

```

```

        HttpResponseMessage response = await
App.client.PostAsJsonAsync($"api/ReaderProfile/Authorization?input={str}",
str);

        response.EnsureSuccessStatusCode();

        int who = await response.Content.ReadAsAsync<int>();
        App.iduser = who;

        await DisplayAlert("Авторизация", "Успешно", "Ok");

        await Navigation.PushAsync(new AboutPage());
    }
}
}
catch { await DisplayAlert("Авторизация", "Не успешно", "Ok"); }
}

private async void Button_Clicked2(object sender, EventArgs e) {
    await Navigation.PushAsync(new Registration());
}
}
}

```

Листинг Д.21 – Файл Registration.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using VKR.ViewModels;
using VKR;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using VKR.Models;
using System.Net.Http;

namespace VKR.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class Registration : ContentPage
    {
        public Registration()
        {
            InitializeComponent();
        }
        private async void Button_Clicked(object sender, EventArgs e)
        {
            EmailValidation EmailValid = new EmailValidation();
            PasswordValidation PassValid = new PasswordValidation();
            if (email.Text != "" && password.Text != "")
            {
                try
                {
                    if (EmailValid.IsValidEmail(email.Text) &&
PassValid.IsValidPassword(password.Text))
                    {
                        User u = new User(0, fio.Text, email.Text,
password.Text);
                        // HttpResponseMessage resp = await
App.client.PostAsync();

```

```

        HttpResponseMessage response = await
App.client.PostAsJsonAsync("api/ReaderProfile/Registration", u);
        response.EnsureSuccessStatusCode();

        int who = await response.Content.ReadAsAsync<int>();
        App.iduser = who;

        await DisplayAlert("Регистрация", "Успешна",
"Продолжить");
        await Navigation.PushAsync(new AboutPage());
    }
    }
    catch (Exception ex) { await DisplayAlert("Регистрация", "Email
занят", "Ok"); }
    } else { return; }
    }

    /*private async void Button_Clicked2(object sender, EventArgs e) {
        await Navigation.PushAsync(new AboutPage());
    }*/
}
}

```

Листинг Д.22 – Файл Registration.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:d="http://xamarin.com/schemas/2014/forms/design"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:xct="http://xamarin.com/schemas/2020/toolkit"
    xmlns:local="clr-namespace:VKR"

    mc:Ignorable="d"
    x:Class="VKR.Views.Registration"
    Shell.NavBarIsVisible="False"
    BackgroundColor="#96d1ff">
    <ContentPage.Content>
        <StackLayout Padding="10,0,10,0" VerticalOptions="Center">

            <Entry x:Name="email" Placeholder="Email"
                Keyboard="Email">
                <Entry.Behaviors>
                    <local:EmailValidation/>
                </Entry.Behaviors>
            </Entry>
            <Entry x:Name="fio" Placeholder="ФИО"
                Keyboard="Text">
            </Entry>
            <Entry IsPassword="True" x:Name="password" Placeholder="Минимум 8
символов, один заглавный и есть числа">
                <Entry.Behaviors>
                    <local>PasswordValidation/>
                </Entry.Behaviors>
            </Entry>
            <Button VerticalOptions="Center" Text="Регистрация"
Clicked="Button_Clicked"/>

        </StackLayout>
    </ContentPage.Content>

```

```
</ContentPage>
```

Листинг Д.23 – Файл EmailValidation.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using Xamarin.Forms;

namespace VKR
{
    public class EmailValidation : Behavior<Entry>
    {
        protected override void OnAttachedTo(Entry entry)
        {
            entry.TextChanged += OnEntryTextChanged;
            base.OnAttachedTo(entry);
        }
        protected override void OnDetachingFrom(Entry entry)
        {
            entry.TextChanged -= OnEntryTextChanged;
            base.OnAttachedTo(entry);
        }
        void OnEntryTextChanged(object sender, TextChangedEventArgs args)
        {
            bool isValid = IsValidEmail(args.NewTextValue);
            ((Entry)sender).TextColor = isValid ? Color.Green : Color.Red;
        }
        public bool IsValidEmail(string email)
        {
            var trimmedEmail = email.Trim();

            if (trimmedEmail.EndsWith("."))
            {
                return false;
            }
            try
            {
                var addr = new System.Net.Mail.MailAddress(email);
                return addr.Address == trimmedEmail;
            }
            catch
            {
                return false;
            }
        }
    }
}
```

Листинг Д.24 – Файл PasswordValidation.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;
using Xamarin.Forms;

namespace VKR
{
    public class PasswordValidation : Behavior<Entry>
    {
```

```

protected override void OnAttachedTo(Entry entry)
{
    entry.TextChanged += OnEntryTextChanged;
    base.OnAttachedTo(entry);
}
protected override void OnDetachingFrom(Entry entry)
{
    entry.TextChanged -= OnEntryTextChanged;
    base.OnAttachedTo(entry);
}
void OnEntryTextChanged(object sender, TextChangedEventArgs args)
{
    bool isValid = isValidPassword(args.NewTextValue);
    ((Entry)sender).TextColor = isValid ? Color.Green : Color.Red;
}
public bool isValidPassword(string password)
{
    var hasNumber = new Regex(@"[0-9]+");
    var hasUpperChar = new Regex(@"[A-Z]+");
    var hasMinimum8Chars = new Regex(@".{8,}");
    var hasCyrillicChars = new Regex(@"\p{IsCyrillic}");
    return !hasCyrillicChars.IsMatch(password)
hasNumber.IsMatch(password)      &&      hasUpperChar.IsMatch(password)      &&
hasMinimum8Chars.IsMatch(password);
}
}
}

```

Листинг Д.25 – Файл App.xaml.cs

```

using System.Collections.Generic;
using VKR.Models;
using VKR.Services;
using VKR.Views;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using System.Net;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;

namespace VKR
{
    public partial class App : Application
    {
        public interface IDocumentView
        {
            void DocumentView(string file, string title);
        }
        public static HttpClient client = new HttpClient();
        public static string path = "http://192.168.0.123:3000/";
        public static int iduser;

        public App()
        {
            InitializeComponent();
            client.BaseAddress = new Uri(path);
            client.DefaultRequestHeaders.Accept.Clear();
            client.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));
            // DependencyService.Register<MockDataStore>();
            MainPage = new AppShell();
        }
    }
}

```

```

        // MainPage = new ItemsPage();
    }

    protected override void OnStart()
    {
    }

    protected override void OnSleep()
    {
    }

    protected override void OnResume()
    {
    }

    /* static async Task RunAsync()
    {
        // Update port # in the following line.
        client.BaseAddress = new Uri("http://localhost:64195/");
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));
    }*/
    }
}

```

Листинг Д.26 – Файл App.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="VKR.App">

    <!--
        Define global resources and styles here, that apply to all pages in
        your app.
    -->
    <Application.Resources>
        <ResourceDictionary>
            <Color x:Key="Primary">#128201</Color>
            <Style TargetType="Button">
                <Setter Property="TextColor" Value="White"></Setter>
                <Setter Property="VisualStateManager.VisualStateGroups">
                    <VisualStateGroupList>
                        <VisualStateGroup x:Name="CommonStates">
                            <VisualState x:Name="Normal">
                                <VisualState.Setters>
                                    <Setter Property="BackgroundColor"
Value="{StaticResource Primary}" />
                                </VisualState.Setters>
                            </VisualState>
                            <VisualState x:Name="Disabled">
                                <VisualState.Setters>
                                    <Setter Property="BackgroundColor"
Value="#332196F3" />
                                </VisualState.Setters>
                            </VisualState>
                        </VisualStateGroup>
                    </VisualStateGroupList>
                </Setter>
            </Style>
        </ResourceDictionary>
    </Application.Resources>
</Application>

```

Приложение Е
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Curl

```
curl -X 'POST' \
  'http://192.168.0.123:3000/api/ReaderProfile/Registration' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "readerID": 0,
    "fio": "Kocob",
    "email": "kos@mail.ru",
    "password": "Qwerty123"
  }'
```

Request URL

http://192.168.0.123:3000/api/ReaderProfile/Registration

Server response

Code	Details
200	<p>Response body</p> <pre>3</pre> <p>Response headers</p> <pre>content-type: text/plain; charset=utf-8 date: Mon,05 Jun 2023 10:36:00 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Code	Description	Links
200	Success	No links

Рисунок Е.1 – Регистрация нового пользователя

Curl

```
curl -X 'POST' \
  'http://192.168.0.123:3000/api/ReaderProfile/Authorization?input=anikin%40mail.ru%7CQwerty123' \
  -H 'accept: */*' \
  -d ''
```

Request URL

http://192.168.0.123:3000/api/ReaderProfile/Authorization?input=anikin%40mail.ru%7CQwerty123

Server response

Code	Details
200	<p>Response body</p> <pre>1</pre> <p>Response headers</p> <pre>content-type: text/plain; charset=utf-8 date: Mon,05 Jun 2023 10:38:35 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Code	Description	Links
200	Success	No links

Рисунок Е.2 – Авторизация пользователя

Curl

```
curl -X 'GET' \
'http://192.168.0.123:3000/api/Literature/LitGet?id=1' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.0.123:3000/api/Literature/LitGet?id=1
```

Server response

Code

Details

200

Response body

```
{
  "literatureID": 1,
  "title": "Война и Мир",
  "author": "Толстой Л.Н.",
  "genre": "Роман",
  "rating": 6
}
```

 Download

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 05 Jun 2023 10:43:10 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Рисунок Е.3 – Получение информации о литературе

[illegible]

Рисунок Е.4 – Скачивание литературы

Curl

```
curl -X 'GET' \
'http://192.168.0.123:3000/api/Literature/Get' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.0.123:3000/api/Literature/Get
```

Server response

Code Details

200

Response body

```
[
  {
    "literatureID": 1,
    "title": "Война и Мир",
    "author": "Толстой Л.Н.",
    "genre": "Роман",
    "rating": 7
  },
  {
    "literatureID": 2,
    "title": "Евгений Онегин",
    "author": "Пушкин А.С.",
    "genre": "Роман",
    "rating": null
  },
  {
    "literatureID": 3,
    "title": "Капитанская дочка",
    "author": "Пушкин А.С.",
    "genre": "Роман",
    "rating": null
  }
]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon,05 Jun 2023 11:09:27 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Рисунок Е.5 – Получение списка литературы в электронном каталоге

Curl

```
curl -X 'GET' \
'http://192.168.0.123:3000/api/DownloadedLiterature/Get?id=1' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.0.123:3000/api/DownloadedLiterature/Get?id=1
```

Server response

Code Details

200

Response body

```
[
  {
    "literatureID": 1,
    "title": "Война и Мир",
    "author": "Толстой Л.Н.",
    "genre": "Роман",
    "rating": 6
  }
]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon,05 Jun 2023 10:50:51 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Рисунок Е.6 – Получение списка скаченной литературы

Curl

```
curl -X 'DELETE' \
'http://192.168.0.123:3000/api/DownloadedLiterature/Delete?str=1%7C1' \
-H 'accept: */*'
```

Request URL

```
http://192.168.0.123:3000/api/DownloadedLiterature/Delete?str=1%7C1
```

Server response

Code	Details
200	<p>Response body</p> <pre>1</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon,05 Jun 2023 10:50:45 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Рисунок Е.7 – Удаление скаченной литературы

Curl

```
curl -X 'GET' \
'http://192.168.0.123:3000/api/Review/Get?LiteratureId=1' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.0.123:3000/api/Review/Get?LiteratureId=1
```

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "literatureID": 1, "fio": "Kamenev A.I.", "comment": "Some text to comment", "rating": 5 }, { "literatureID": 1, "fio": "Anikin A.I.", "comment": " Some text", "rating": 7 }]</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon,05 Jun 2023 10:55:24 GMT server: Kestrel transfer-encoding: chunked</pre>

Рисунок Е.8 – Получение комментариев к заданной литературе

Curl

```
curl -X 'POST' \
  'http://192.168.0.123:3000/api/Review/Post' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "reviewID": 0,
    "readerID": 3,
    "literatureID": 1,
    "rating": 9,
    "comment": "Комментарий"
  }'
```

Request URL

<http://192.168.0.123:3000/api/Review/Post>

Server response

Code	Details
200	<p>Response body</p> <pre>1</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon,05 Jun 2023 10:57:51 GMT server: Kestrel transfer-encoding: chunked</pre>

Responses

Рисунок Е.9 – Добавление комментария

Curl

```
curl -X 'GET' \
  'http://192.168.0.123:3000/api/DownloadedLiterature/LitCheck?ch=1%7C1' \
  -H 'accept: */*'
```

Request URL

<http://192.168.0.123:3000/api/DownloadedLiterature/LitCheck?ch=1%7C1>

Server response

Code	Details
200	<p>Response body</p> <pre>1</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Mon,05 Jun 2023 10:54:47 GMT server: Kestrel transfer-encoding: chunked</pre>

Рисунок Е.10 – Проверка загружена ли литература пользователем

Curl

```
curl -X 'GET' \
'http://192.168.0.123:3000/api/Literature/GetSortName' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.0.123:3000/api/Literature/GetSortName
```

Server response

Code Details

200

Response body

```
[
  {
    "literatureID": 1,
    "title": "Война и Мир",
    "author": "Толстой Л.Н.",
    "genre": "Роман",
    "rating": 7
  },
  {
    "literatureID": 2,
    "title": "Евгений Онегин",
    "author": "Пушкин А.С.",
    "genre": "Роман",
    "rating": null
  },
  {
    "literatureID": 3,
    "title": "Капитанская дочка",
    "author": "Пушкин А.С.",
    "genre": "Роман",
    "rating": null
  }
]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon,05 Jun 2023 11:10:29 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Рисунок Е.11 – Получение списка литературы, отсортированного по названию

Curl

```
curl -X 'GET' \
'http://192.168.0.123:3000/api/Literature/GetSortAuth' \
-H 'accept: text/plain'
```

Request URL

```
http://192.168.0.123:3000/api/Literature/GetSortAuth
```

Server response

Code Details

200

Response body

```
[
  {
    "literatureID": 2,
    "title": "Евгений Онегин",
    "author": "Пушкин А.С.",
    "genre": "Роман",
    "rating": null
  },
  {
    "literatureID": 3,
    "title": "Капитанская дочка",
    "author": "Пушкин А.С.",
    "genre": "Роман",
    "rating": null
  },
  {
    "literatureID": 1,
    "title": "Война и Мир",
    "author": "Толстой Л.Н.",
    "genre": "Роман",
    "rating": 7
  }
]
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon,05 Jun 2023 11:13:59 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Рисунок Е.12 – Получение списка литературы, отсортированного по автору

Приложение Ж
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

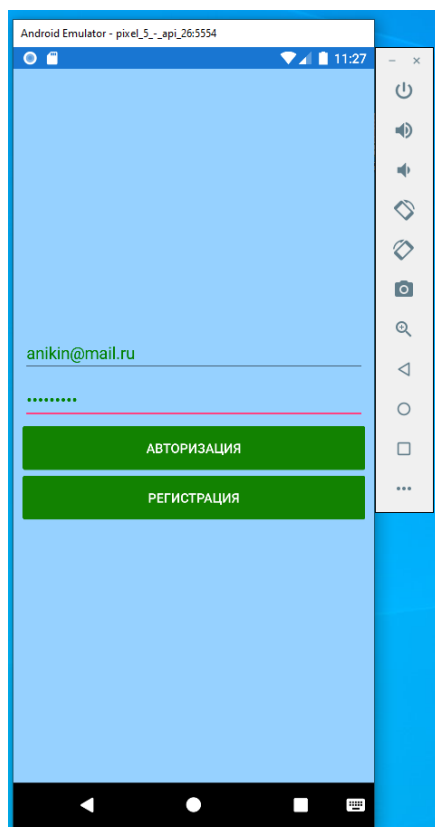


Рисунок Ж.1 – Авторизация

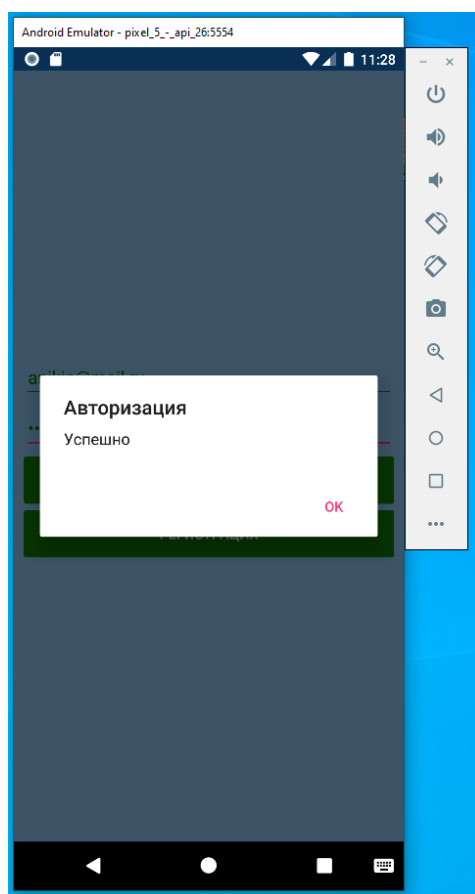


Рисунок Ж.2 – Успешная авторизация

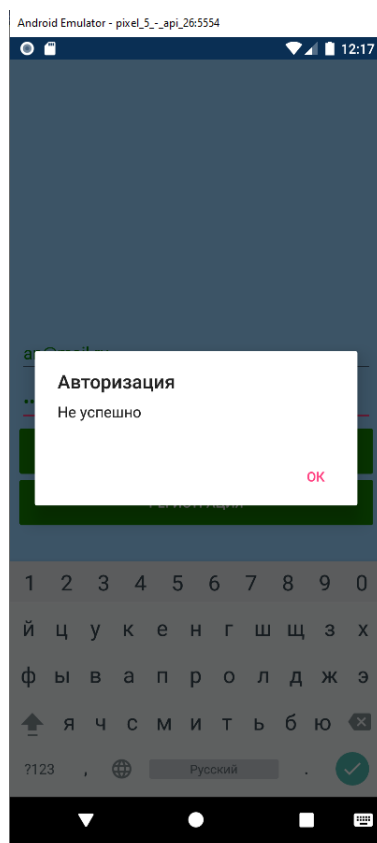


Рисунок Ж.3 – Неуспешная авторизация

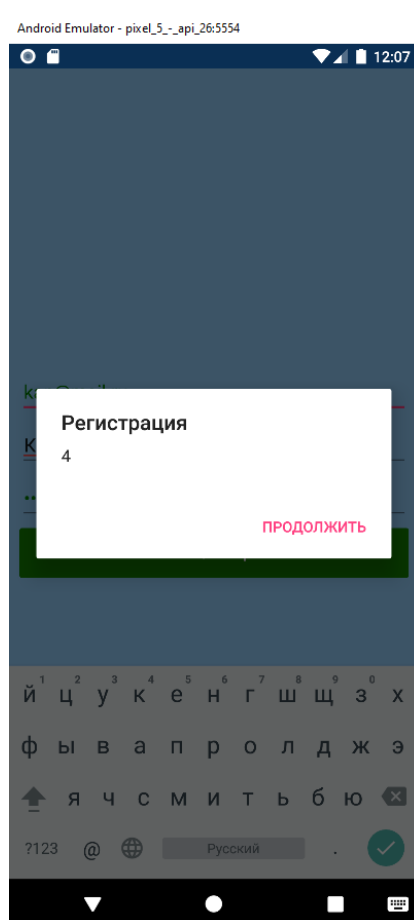


Рисунок Ж.4 – Успешная регистрация

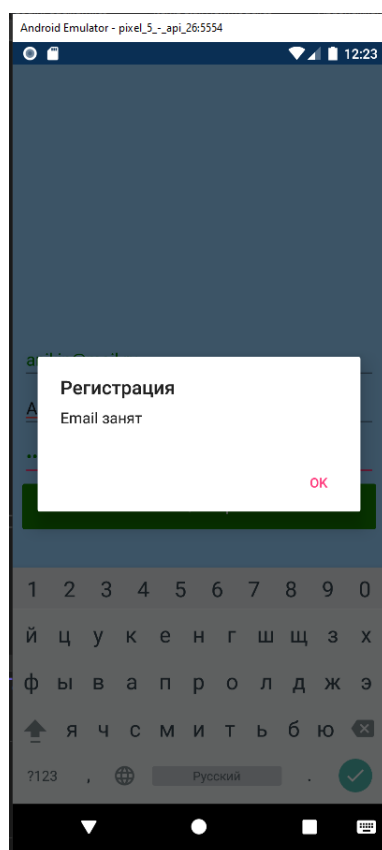


Рисунок Ж.5 – Неуспешная регистрация

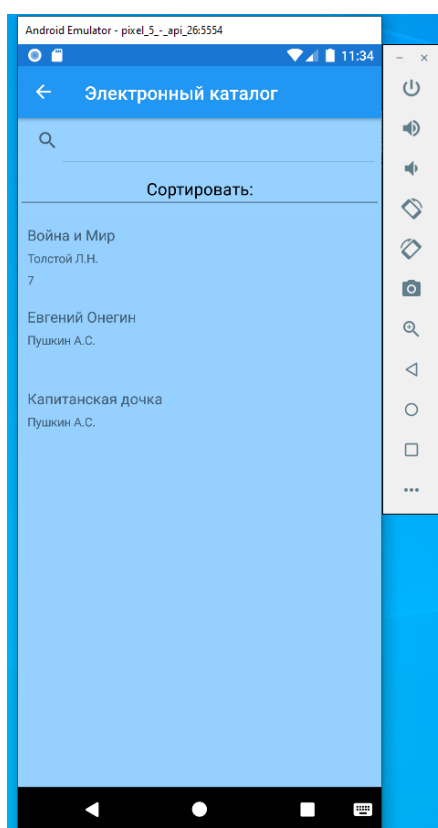


Рисунок Ж.6 – Открытый электронный каталог\

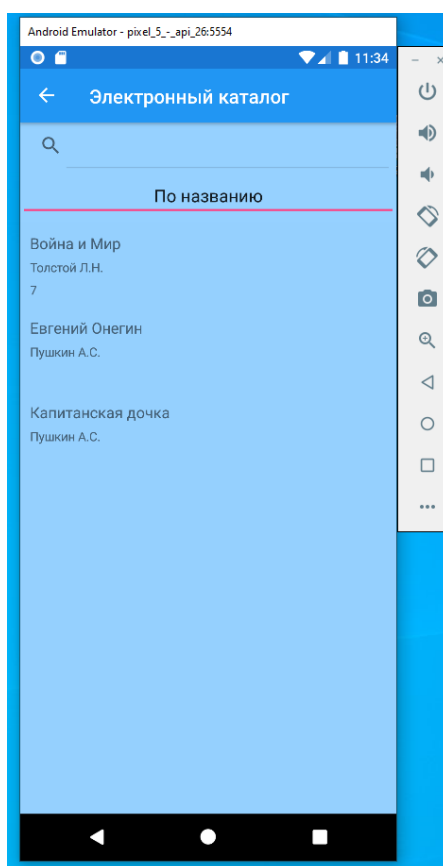


Рисунок Ж.7 – Сортировка по названию

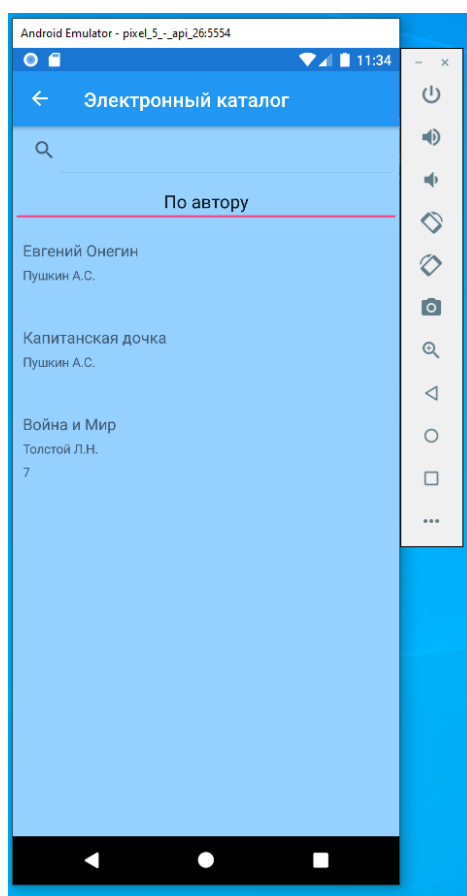


Рисунок Ж.8 – Сортировка по автору

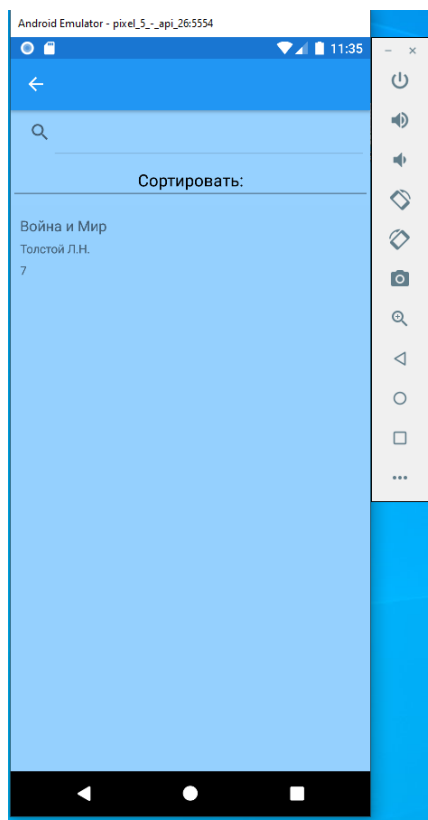


Рисунок Ж.9 – Загруженная литература

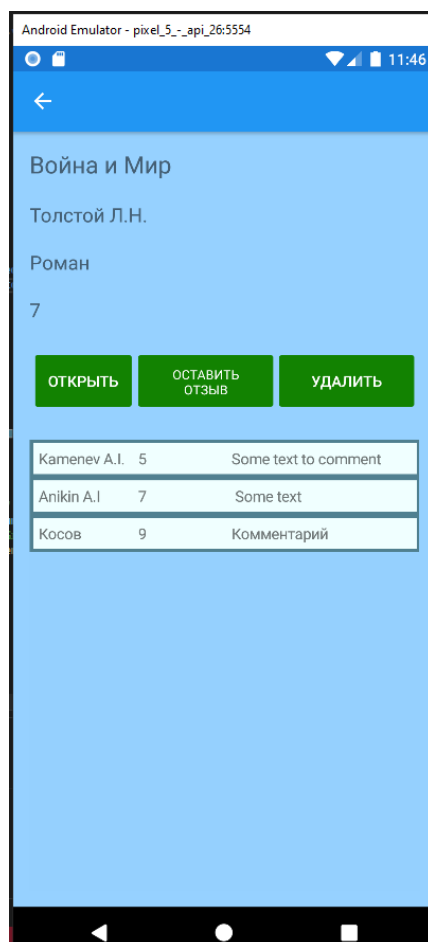


Рисунок Ж.10 – Страница литературы

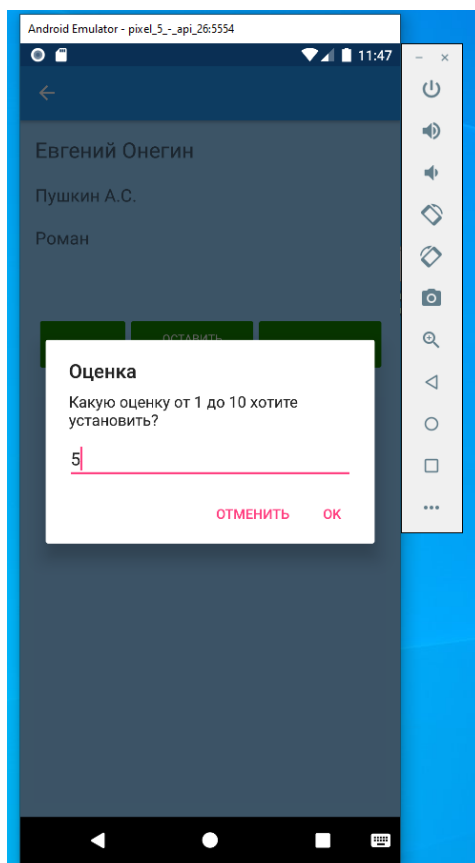


Рисунок Ж.11 – Отзыв 1

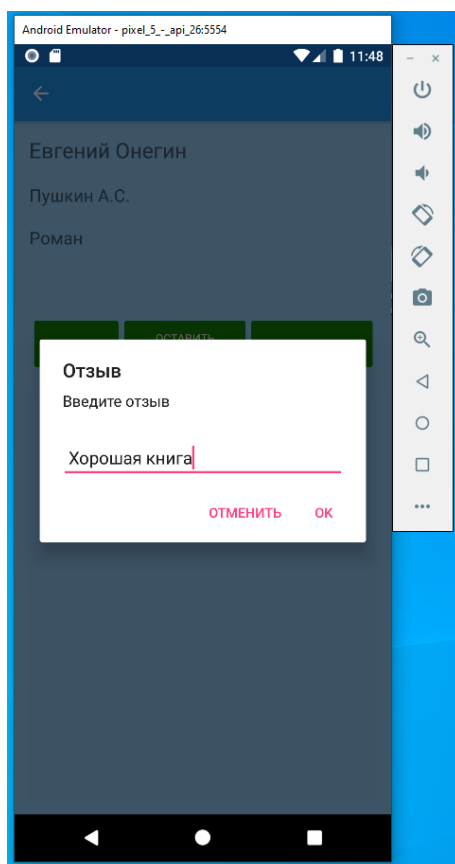


Рисунок Ж.12 – Отзыв 2

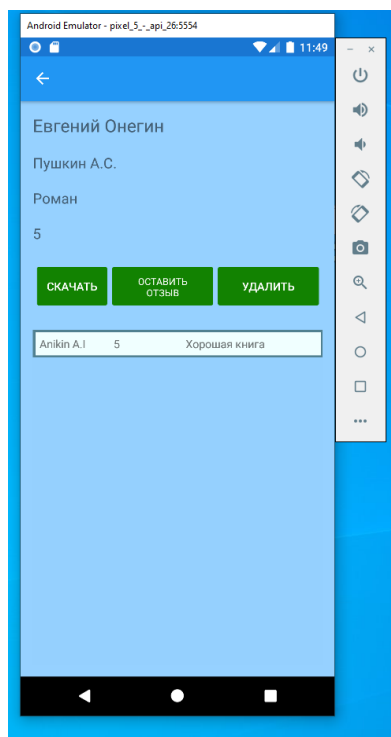


Рисунок Ж.13 – Отзыв 3

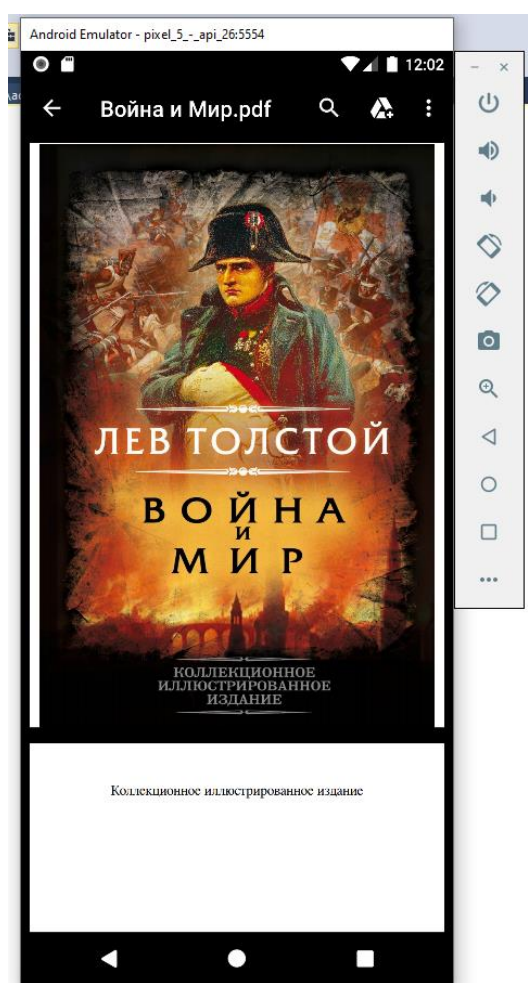


Рисунок Ж.14 – Открытая книга

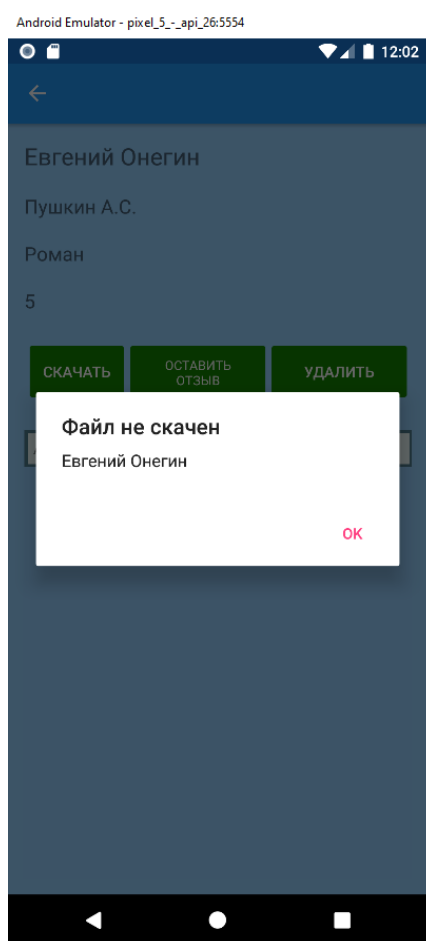


Рисунок Ж.15 – Удаление книги