

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

**Программное обеспечение мобильной
поддержки сервисов электронной научной
библиотеки**

Студент

Игорь 18.06.21 Ащеулов Игорь Олегович
(подпись, дата) (ФИО полностью)

Руководитель

ВН 18.06.21 Князев В.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

ДВ 22.06.21 Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

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

Заведующий кафедрой

П.П. Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от 25.06.21 № 9)

Секретарь ГЭК

Н.А. Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2021

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
«21» января 2021 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Ащеулов Игорь Олегович гр. 17ВП1 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Программное обеспечение мобильной поддержки сервисов
электронной научной библиотеки

Тема утверждена приказом ПГУ № 002740 от «21» января 2021

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 сервисов электронной научной библиотеки.*

2) *Среда выполнения приложения: операционная система Android 6 и выше*

Среда выполнения сервера: операционная система Windows 10

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

*Инструментальные средства разработки: Android Studio, Microsoft SQL Server
2019, Microsoft Visual Studio 2019, Microsoft Project, ArgoUML, Erwin Data Modeler*

Технологии разработки: RUP, ООП, 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
критериями;*

в) добавление новых книг в систему и удаление имеющихся;

*г) добавление оценок и комментариев к книгам, удаление существующих
комментариев*

5. Перечень подлежащих разработке вопросов

1) *Анализ предметной области и постановка задачи*

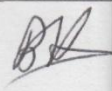
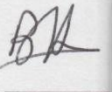
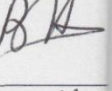
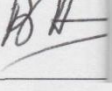
2) *Проектирование*

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

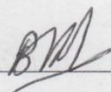
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области постановка задачи	15%	14.04.2021– 21.04.2021	
2	Проектирование	25%	22.04.2021– 05.05.2021	
3	Реализация	40%	06.05.2021– 30.05.2021	
4	Контроль качества программного обеспечения	5%	31.05.2021– 02.06.2021	
5	Оформление пояснительной записки	15%	03.06.2021– 14.06.2021	

Дата выдачи задания "28" декабря 2020

Руководитель бакалаврской работы

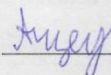
(подпись, дата)



28.12.2020

Задание к исполнению принял студент

(подпись, дата)

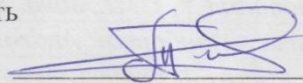


28.12.20

Работу к защите допустить

Декан факультета

(подпись, дата)



Реферат

Пояснительная записка содержит 138 листов, 83 рисунка, 16 таблиц, 36 использованных источников, 6 приложений.

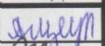
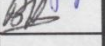
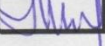
МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ЭЛЕКТРОННАЯ НАУЧНАЯ БИБЛИОТЕКА, ANDROID, MS SQL SERVER, JAVA, C#, RUP, UML

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

Цель работы – разработка мобильного приложения электронной научной библиотеки, позволяющего добавлять в систему новые электронные книги, а также искать, скачивать и удалять уже имеющиеся и добавлять оценки и комментарии к этим книгам.

Разработка серверной части происходила на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2019. В качестве СУБД была использована СУБД MS SQL Server.

Разработка приложения производилась на языке программирования Java. В качестве среды разработки использовалась среда программирования Android Studio, операционная система Windows 10. Приложение предназначено для использования под управлением ОС Android версии 6.0 и выше, а его серверная часть работает под управлением операционной системы Windows 7 и выше.

					ПГУ 09.03.04 – 08BP171.04 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программное обеспечение мобильной поддержки сервисов электронной научной библиотеки. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Ащеулов И.О.		18.06.20			3	138
Пров.		Князев В.Н.		18.06.20				
Реценз.								
Н. контр.		Такташкин Д.В.		22.06.20				
Утверд.		Макарычев П.П.				Группа 17ВП1		

Содержание

Введение.....	6
1 Анализ предметной области и требований	9
1.1 Анализ предметной области и постановка задачи.....	9
1.2 Анализ программных продуктов аналогичного назначения	10
1.2.1 Научная электронная библиотека eLibrary	10
1.2.2 Национальная электронная библиотека (НЭБ)	11
1.2.3 КиберЛенинка.....	14
1.2.4 Сравнение аналогов	15
1.3 Анализ целевой аудитории	15
1.4 Анализ функциональных требований.....	17
1.5 Анализ нефункциональных требований.....	22
1.6 Планирование разработки и оценка бюджета.....	22
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.....	38
2.4 Логическое проектирование базы данных	47
2.5 Проектирование серверной части приложения	51
3 Реализация мобильного приложения	53
3.1 Реализация базы данных	53
3.2 Разработка серверной части.....	53
3.3 Реализация клиентской части приложения	56
4 Тестирование программного продукта	59
4.1 Тестирование серверной части приложения	59
4.2 Тестирование клиентской части приложения	61
4.2.1 Функциональное тестирование.....	61
4.2.2 Модульное тестирование	63
4.3 Метрики кода.....	65
4.3.1 Метрики размера программы.....	66
4.3.2 Метрики Холстеда	66

Закключение	69
Список использованных источников	70
Приложение А. Глоссарий	74
Приложение Б. Исходный код скриптов базы данных	76
Приложение В. Исходный код серверной части.....	82
Приложение Г. Исходный код мобильного приложения.....	94
Приложение Д. Результаты тестирования серверной части.....	122
Приложение Е. Результаты тестирования мобильного приложения.....	129

Введение

В наше время информационные технологии все шире распространяются на разнообразные сферы человеческой деятельности. С каждым годом по всему миру компьютеры все активнее используют в различных отраслях экономики, и это кардинально меняет жизнь людей.

Не остается в стороне от этой тенденции и наша страна. В ней также взят курс на максимальное внедрение информационных технологий во всех сферах деятельности, который поддерживается на государственном уровне. 9 мая 2017 года была утверждена «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». В ней были поставлены основные цели и задачи политики Российской Федерации в сфере информационных технологий, а также объявлен курс на формирование национальной цифровой экономики [1]. С этой же целью в том же 2017 году была принята государственная программа «Цифровая экономика». Она призвана 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 важнейших особенностей развития информационных технологий в нашей стране, да и во всем современном мире, является повсеместное распространение мобильных устройств с доступом к интернету. При этом мобильный интернет постепенно оттесняет на второй план традиционный доступ к сети с использованием стационарных компьютеров по причине своего существенно большего удобства. В частности, как отмечено в государственной программе «Цифровая экономика», в 2016 году в нашей стране «на 100 человек приходилось 159,95 мобильного телефона и из 100 человек 71,29 человека использовали мобильный доступ к сети Интернет» [2]. Со временем эта тенденция лишь усиливается.

Вследствие этого для большинства интернет-ресурсов пользователи мобильных устройств стали играть ключевую роль, оттесняя пользователей ПК на второй план. Поэтому создатели веб-сервисов стараются обеспечить максимальное удобство использования своего продукта именно для мобильных платформ, с целью чего активно выпускаются специальные мобильные приложения.

Всеобщая компьютеризация затрагивает даже такие, казалось бы, далекие от компьютеров организации, как библиотеки. Это связано с тем, что появилась альтернатива классическим бумажным книгам в лице их электронных версий. При этом электронные книги имеют ряд преимуществ над классическими. В частности, бумажные книги тяжелые и занимают много места, что не дает всегда держать их под рукой. Электронные книги же могут быть прочитаны в практически любых условиях, достаточно, чтобы под рукой находился ноутбук, планшет, смартфон или другой подобный компактный гаджет. В ближайшее время подобные тенденции сохранятся, и электронные книги продолжают становиться все популярнее.

Библиотеки, как и любые другие организации, должны идти в ногу со временем, адаптируясь к новейшим достижениям техники и информационных технологий. Массовое распространение электронных книг приводит к тому, что для успешного функционирования им уже не достаточно простой коллекции печатных книг. Дело в том, что в случае, если библиотека работает исключительно с печатными книгами, потенциальный круг ее посетителей ограничивается исключительно людьми, живущими в относительной близости от нее. Это допустимо для небольших библиотек, ориентированных исключительно на жителей своего района, но не для крупных библиотек. Особенно важна максимальная доступность имеющихся в коллекции книг для всех желающих для библиотек с узкоспециализированной литературой, в частности, для научных библиотек, так как их потенциальными читателями является довольно небольшая доля населения. Поэтому им нужно искать свою аудиторию не только среди людей, живущих поблизости, но и среди тех, кто проживает на большом

расстоянии от места физического расположения библиотеки и не способен посетить ее лично. Единственный способ успешно решить эту проблему заключается в активном использовании библиотеками электронных версий книг, которые с легкостью можно просматривать удаленно.

Другой причиной, по которой для библиотек важно использование электронных книг, является возможность легкого обмена информацией между библиотеками, а также другими организациями (например, издательствами или журналами), с которыми они сотрудничают. Это связано с тем, что получить электронную версию той или иной книги можно гораздо быстрее, чем печатную. Особенно это важно для научных библиотек, которым необходимо регулярно обновлять свои коллекции, ведь новые научные труды выходят очень и очень часто.

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]. Поэтому одной из основных задач, стоящих перед современными библиотеками, является их модернизация. Она представляет собой «перевод основных процессов, обеспечивающих жизнедеятельность библиотек, на цифровые технологии», включающий в себя в том числе «разработку и внедрение принципиально новых форматов хранения и методов работы с книгами», т.е. использование наряду с классическими печатными версиями книг также и их электронных версий. В частности, одной из главных тенденций современных библиотек является полномасштабная оцифровка их книжных фондов [4].

Таким образом, на основании приведенных выше факторов можно сделать вывод о том, что разработка мобильного приложения для электронной научной библиотеки весьма актуальна и перспективна в условиях современного общества.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Разрабатываемое мобильное приложение должно обеспечить людям максимально удобный доступ к книгам из фонда библиотеки. Благодаря ему пользователям не потребуется лично приходить в библиотеку, чтобы получить физический экземпляр книги. Достаточно будет иметь мобильное устройство с установленным приложением и подключение к интернету. Это позволит получить доступ к электронной версии имеющихся в библиотечном фонде книг из любого места, где есть мобильный интернет или Wi-fi, что весьма удобно для читателей.

Приложение должно обеспечивать всем зарегистрированным пользователям доступ к любой электронной книге из числа имеющихся в системе. Также пользователям должна быть предоставлена удобная система поиска книг, позволяющая отбирать их в соответствии с желаемыми пользователем критериями, и возможность высказывать свою точку зрения относительно представленных в приложении научных трудов путем оставления к ним оценок и комментариев.

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.

Помимо рядовых пользователей в системе будут присутствовать и администраторы, которые будут иметь более широкие возможности действий. В частности, они должны быть способны добавлять в систему новые книги и удалять уже имеющиеся, а также удалять недопустимые

комментарии читателей к книгам (например, содержащие оскорбления или спам) и блокировать пользователей-нарушителей.

1.2 Анализ программных продуктов аналогичного назначения

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

1.2.1 Научная электронная библиотека eLibrary

Платформа eLibrary была создана в 1999 году для обеспечения российским ученым доступа к иностранным научным изданиям. С 2005 года eLibrary начала работу с русскоязычными публикациями и с тех пор является ведущей русскоязычной электронной научной библиотекой [5]. Также на основе этого портала был создан Российский индекс научного цитирования (РИНЦ).

Главная страница электронной библиотеки eLibrary представлена на рисунке 1.



Рисунок 1 – Главная страница электронной библиотеки eLibrary

На сегодня пользователям портала eLibrary доступны свыше 34 млн научных публикаций и патентов, в том числе более 5600 отечественных научно-технических журналов. Общее число зарегистрированных институциональных пользователей (организаций) – более 3600. Помимо этого, в системе зарегистрированы свыше 3,5 миллиона индивидуальных пользователей из 125 стран мира [6]. Ежегодно читатели получают из библиотеки более 12 миллионов полнотекстовых статей и просматривают более 90 миллионов аннотаций [7]. Для каждой книги содержится информация о ее наличии как в базе данных РИНЦ, так и в международных библиографических базах Scopus и Web of Science, а также о количестве цитирований в каждой из баз.

Сайт предоставляет достаточно мощные поисковые средства, позволяющие производить поиск требуемой книги по множеству критериев. В частности, публикации можно отбирать по типу, тематике, автору, дате публикации и доступности для пользователя. Поиск может производиться не только в названии публикации или ее тексте, но и, например, в списке цитируемой литературы.

Недостатком портала является его слабая приспособленность для мобильных платформ. В частности, сайт библиотеки имеет достаточно громоздкий интерфейс, который явно ориентирован на персональные компьютеры (ПК) и неудобен для использования на планшетах или смартфонах, а мобильное приложение для нее и вовсе отсутствует.

1.2.2 Национальная электронная библиотека (НЭБ)

НЭБ – это государственная информационная система, включающая в себя оцифрованные фонды целого ряда отечественных библиотек, в том числе таких гигантов, как Российская государственная библиотека, Российская национальная библиотека, Государственная публичная научно-техническая библиотека, Всероссийская государственная библиотека иностранной литературы, Государственная публичная историческая библиотека, Российская государственная библиотека искусств, Российская

государственная детская библиотека, Всероссийская патентно-техническая библиотека и т.д. [8].

Главный экран приложения представлен на рисунке 2.

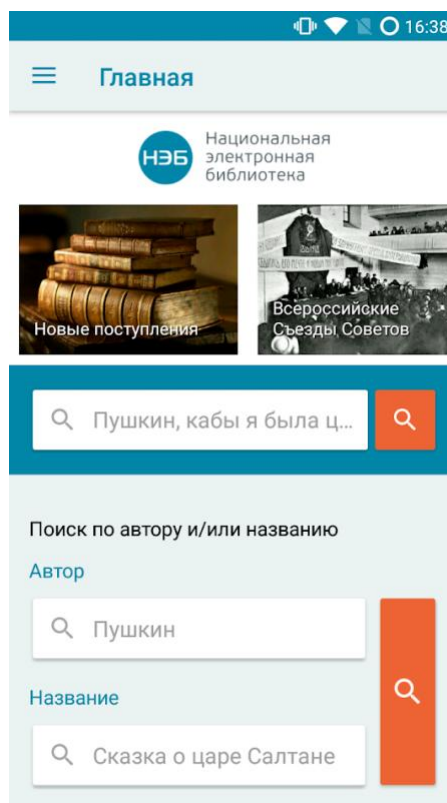


Рисунок 2 – Главный экран мобильного приложения «НЭБ»

Фонд Национальной электронной библиотеки включает в себя более 4 млн. электронных копий различных произведений. Вот лишь некоторые виды изданий, которые в ней имеются:

- научно-образовательная литература;
- художественная литература;
- нотные издания;
- картографические издания;
- патентные документы;
- авторефераты и диссертации;
- периодические издания (газеты, журналы, альманахи).

Большую часть фондов НЭБ составляют произведения, перешедшие в общественное достояние. Их можно читать, скачивать и копировать без ограничений. Однако доступ к книгам, охраняемым авторским правом,

ограничен. Для них доступна лишь информация об их наличии в библиотеках-участниках проекта НЭБ [9].

В НЭБ можно производить поиск по множеству критериев. В частности, возможен поиск произведений по их авторам, названию, дате и месту издания, по направленности (книга, периодическое издание, автореферат или диссертация и т.д.), по доступности книги (доступна в электронном виде или только в печатном) и даже по тексту самого произведения.

Использование НЭБ для пользователей и библиотек является бесплатным. Регистрация пользователей на портале необязательна и не требуется для доступа к тем или иным электронным ресурсам. Однако зарегистрированные пользователи получают в свое распоряжение дополнительный функционал, включающий в себя такие возможности, как создание закладок, личных подборок, заметок или сохранение истории поиска.

Одним из достоинств библиотеки является наличие для нее мобильного приложения, что значительно облегчает ее использование на телефонах и планшетах. Однако это приложение достаточно ненадежно в работе.

Недостатком НЭБ является то, что данный ресурс не является чистой научной библиотекой. Помимо научных трудов, здесь хранится также и множество книг других направлений, что может вызвать путаницу при поиске. Кроме того, часть функционала можно считать избыточной. Например, возможность отслеживания наличия книг в библиотеках в печатном виде приводит к появлению в системе большого числа книг, для которых электронная версия оказывается недоступной, а единственное, что присутствует на странице подобных книг, – это информация о том, в какой библиотеке ее можно найти в бумажном виде. Такие страницы зачастую оказываются бесполезны и лишь раздувают список результатов поиска. Также мобильное приложение системы работает достаточно нестабильно.

1.2.3 КиберЛенинка

КиберЛенинка поддерживает распространение знаний по модели открытого доступа (Open Access), обеспечивая бесплатный оперативный доступ к научным публикациям в электронном виде, которые в зависимости от договорённостей с правообладателем размещаются по лицензии Creative Commons Attribution (CC-BY) [10].

Главная страница электронной библиотеки представлена на рисунке 3.



Рисунок 3 – Главная страница электронной библиотеки «КиберЛенинка»

Библиотека комплектуется научными статьями, публикуемыми в журналах России и ближнего зарубежья. Ко всем этим статьям предоставляется бесплатный открытый доступ. При этом для каждой из статей хранятся сведения о ее наличии или отсутствии в ряде библиографических баз, таких как перечень ВАК, Scopus или ESCI.

Недостатком данной библиотеки является то, что ее фонд ограничен лишь статьями и не поддерживает другие типы научных изданий. Кроме того, хотя сайт библиотеки и имеет удобную мобильную версию, полноценного мобильного приложения для нее не существует. Также на

портале используется весьма сложная и неудобная система поиска, с которой крайне тяжело работать непривычным к ней пользователям.

1.2.4 Сравнение аналогов

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

Таблица 1 – Сравнение аналогов

Интернет-библиотека	Только научные издания	Типы научных изданий	Мобильное приложение	Система поиска
eLibrary	Да	Все	Отсутствует	Понятная
НЭБ	Нет	Все	Существует	Понятная
КиберЛенинка	Да	Только статьи	Отсутствует, но есть мобильная версия сайта	Сложная в использовании
Разрабатываемое приложение	Да	Все	Существует	Понятная

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 научных электронных библиотек, использующие для работы с книгами мобильные устройства. В настоящее время число пользователей научных электронных библиотек стабильно возрастает. В частности, подобный вывод можно сделать на основании статистики изменения числа пользователей у одной из крупнейших электронных научных библиотек нашей страны eLibrary. График, на котором показана динамика изменения числа пользователей этой библиотеки в 2005 – 2017 годах, представлен на рисунке 4.

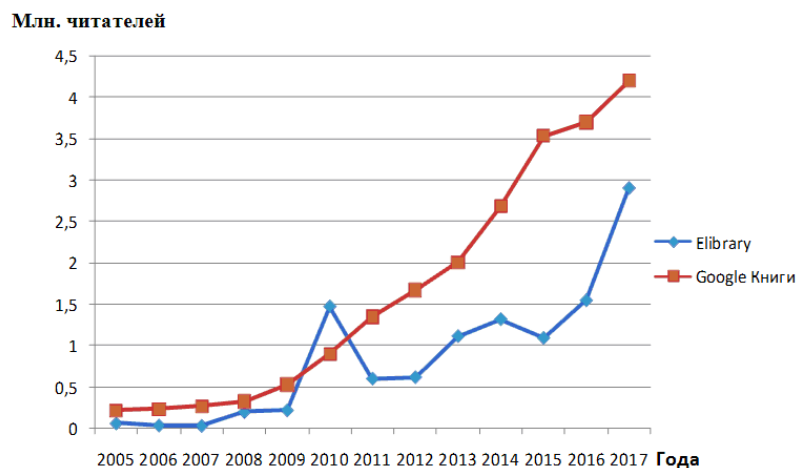


Рисунок 4 – Изменение числа пользователей электронной библиотеки elibrary

Как можно увидеть, в истории библиотеки бывали периоды временного падения посещаемости (например, отток пользователей в 2011 году, вызванный тем что серверы библиотеки оказались технически не готовы к наплыву новых читателей в 2010), однако в целом на протяжении этого отрезка число ее читателей устойчиво возрастало [11]. В последние годы эта тенденция сохранилась, и в настоящий момент число зарегистрированных пользователей elibrary превысило 3,5 млн. человек. Непрерывный рост посещаемости прослеживается и у других популярных электронных научных библиотек, таких как «КиберЛенинка» [12].

На рисунке 5 представлены доли основных производителей смартфонов на мировом рынке за последние 3 года [13].

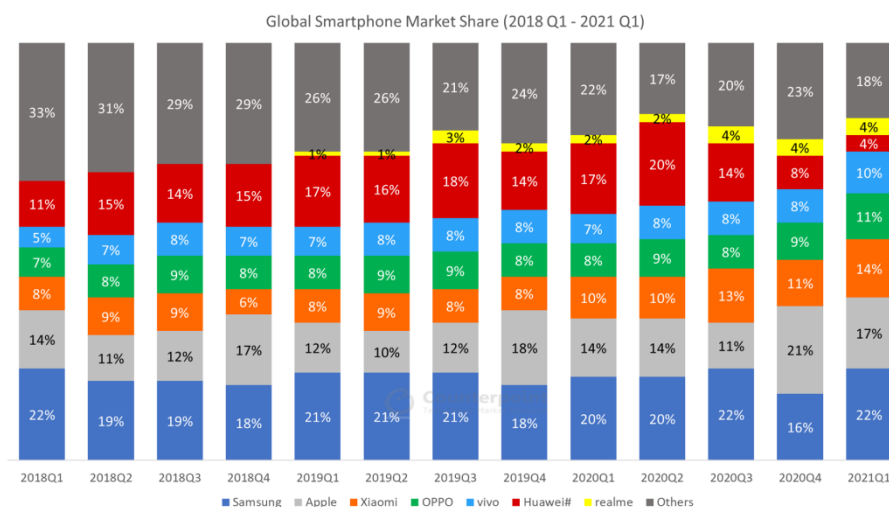


Рисунок 5 – Доли производителей смартфонов на мировом рынке

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. А в нашей стране доля смартфонов Apple даже ниже, чем в среднем по планете [14]. 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 6 [15].

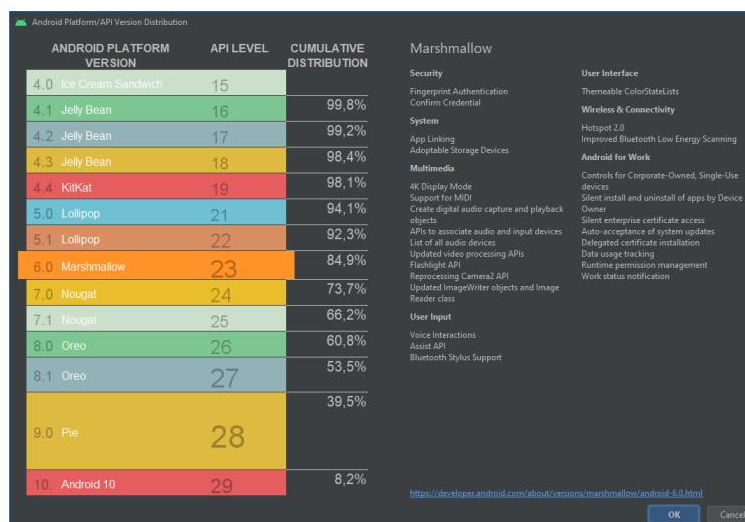


Рисунок 6 – Распространение различных версий системы Android

Можно увидеть, что 84,9 % пользователей системы Android использует версии не старше 6.0. Поэтому приложение будет разрабатываться таким образом, чтобы оно успешно функционировало для Android 6.0 и более новых версий системы.

1.4 Анализ функциональных требований

Поскольку основным назначением электронной научной библиотеки является работа с книгами, именно книги будут являться основными объектами системы. Для каждой из них, помимо ее названия и текста, будет

храниться дополнительная информация: автор, год издания, жанр, количество страниц и т.д., а также оценки и отзывы, оставленные пользователями.

В системе должно быть два основных типа пользователей: рядовые пользователи и администраторы. Рядовой пользователь должен иметь возможность поиска и скачивания нужных ему книг, а также оставления отзывов и оценок для прочитанных произведений.

Администраторы должны иметь все возможности, предусмотренные для рядовых пользователей. Кроме того, администратор должен быть способен удалять из системы любые книги или отзывы к ним при обнаружении нарушений, а также блокировать пользователей, допустивших нарушения.

В результате анализа требований задания была создана диаграмма вариантов использования (рисунок 7).

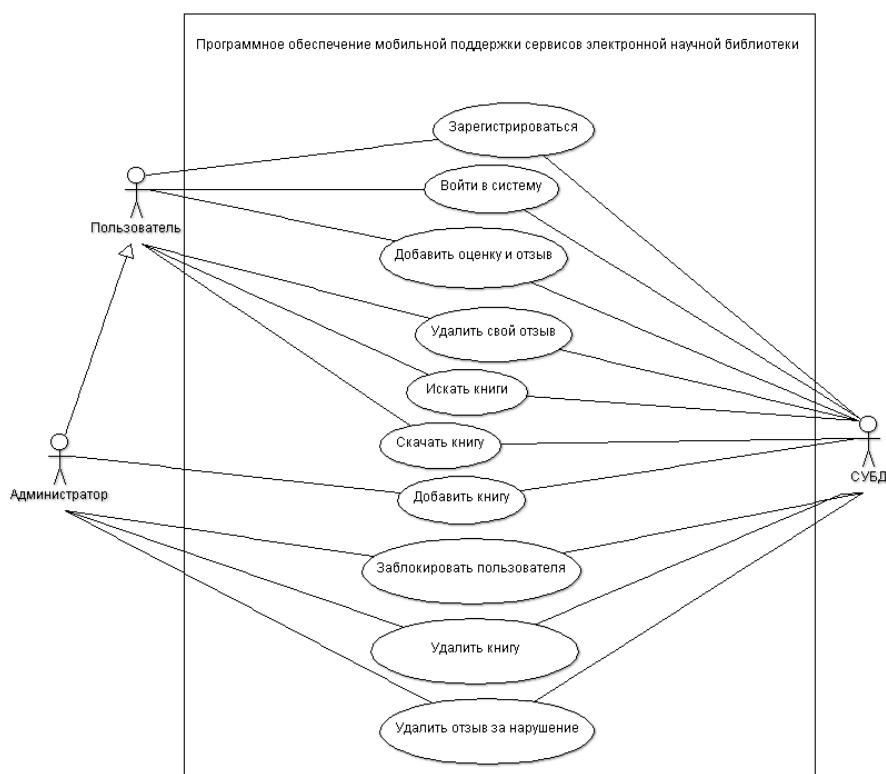


Рисунок 7 – Диаграмма вариантов использования мобильного приложения

Составим спецификации вариантов использования «Зарегистрироваться» (рисунок 8), «Войти в систему» (рисунок 9), «Искать книги» (рисунок 10), «Добавить оценку и отзыв» (рисунок 11).

Название: зарегистрироваться
Назначение: регистрация нового аккаунта пользователя
Актеры: пользователь, СУБД
Предусловие: приложение запущено
Основной сценарий: 1. Пользователь выбирает регистрацию 2. Пользователь вводит логин и пароль. 3. Система проверяет логин и пароль на корректность. 4. Если логин и пароль корректны. 4.1. Система запрашивает у СУБД сведения о наличии аккаунта с введенным логином. 4.2. СУБД передает сведения о наличии аккаунта с введенным логином. 4.3. Если такого аккаунта нет 4.3.1. СУБД добавляет в БД информацию о новом пользователе. 4.3.2. Система осуществляет вход нового пользователя. 4.4. Иначе система выводит сообщение о том, что выбранный логин занят. 5. Иначе система выводит сообщение о недопустимом логине или пароле.
Постусловие: регистрация выполнена

Рисунок 8 – Спецификация варианта использования «Зарегистрироваться»

Название: войти в систему
Назначение: вход в систему с существующего аккаунта пользователя
Актеры: пользователь, СУБД
Предусловие: приложение запущено
Основной сценарий: 1. Пользователь выбирает авторизацию 2. Пользователь вводит логин и пароль. 3. Система запрашивает у СУБД сведения о наличии аккаунта с введенным логином и паролем. 4. СУБД передает сведения о наличии аккаунта с такими логином и паролем 5. Если такой аккаунт существует 5.1. Система проверяет наличие у него активной блокировки. 5.2. Если активной блокировки нет 5.2.1. Система осуществляет вход нового пользователя. 5.3. Иначе система сообщает о наличии у пользователя блокировки. 6. Иначе система выводит сообщение о неверном логине или пароле.
Постусловие: вход в систему выполнен

Рисунок 9 – Спецификация варианта использования «Войти в систему»

Название: искать книги
Назначение: поиск книг в соответствии с заданными параметрами
Актеры: пользователь, СУБД
Предусловие: пользователь авторизован
Основной сценарий: <ol style="list-style-type: none"> 1. Пользователь выбирает поиск книг. 2. Система запрашивает параметры поиска. 3. Пользователь вводит параметры поиска 4. Система передает параметры поиска серверу БД. 5. СУБД отбирает книги в соответствии с заданными параметрами поиска 6. СУБД возвращает список найденных книг системе. 7. Система выводит информацию о найденных книгах на экран.
Постусловие: поиск успешно завершен

Рисунок 10 – Спецификация варианта использования «Искать книги»

Название: добавить оценку и отзыв
Назначение: добавление новых оценки и отзыва к книге.
Актеры: пользователь, СУБД
Предусловие: книга выбрана
Основной сценарий: <ol style="list-style-type: none"> 1. Пользователь выбирает добавление оценки и отзыва. 2. Пользователь вводит значение оценки и текст комментария. 3. Система отправляет оценку и комментарий на сервер БД. 4. СУБД сохраняет оценку и комментарий в БД.
Постусловие: добавление оценки и отзыва завершено

Рисунок 11 – Спецификация варианта использования «Добавить оценку и
ОТЗЫВ»

Рассмотрим некоторые варианты использования поподробнее. Для варианта использования «Зарегистрироваться» составлена диаграмма деятельности, представленная на рисунке 12.

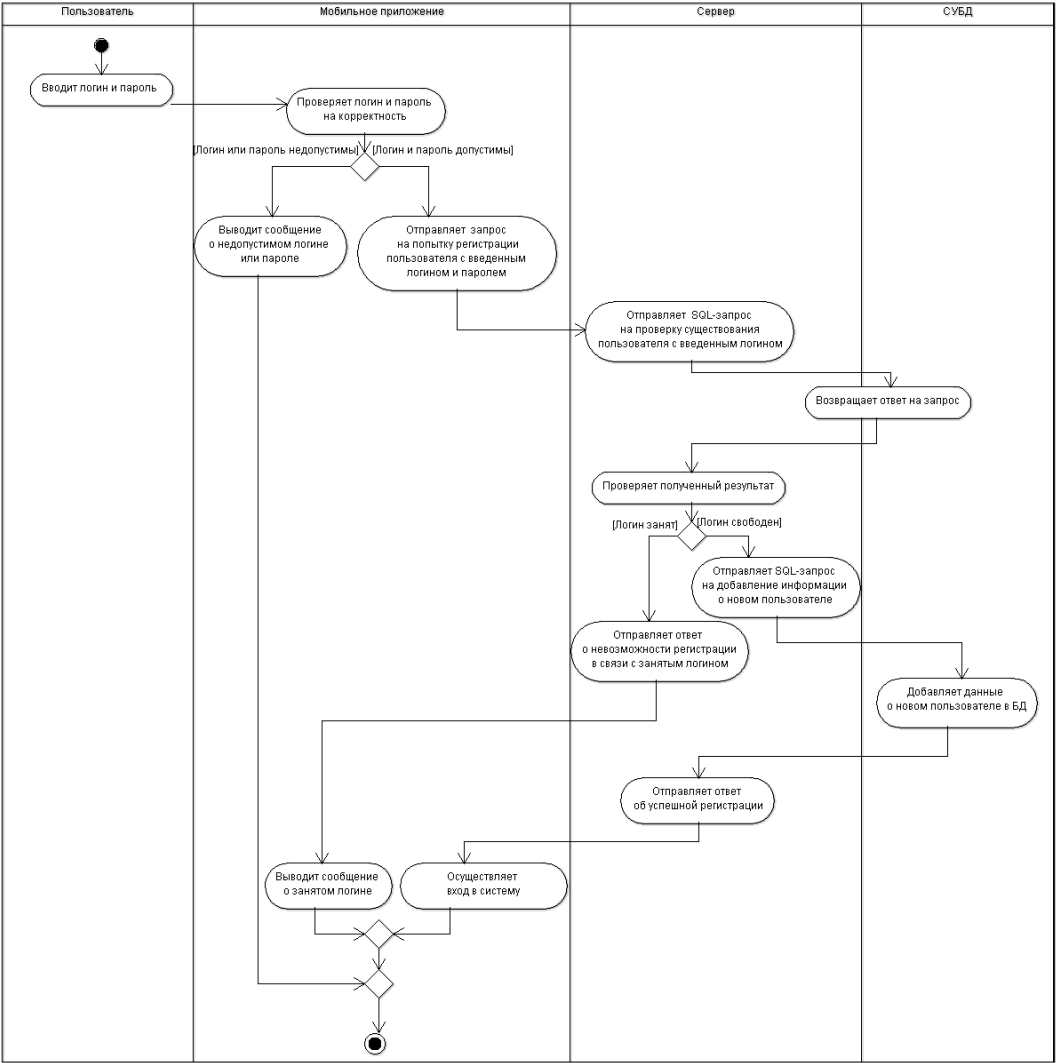


Рисунок 12 – Диаграмма деятельности варианта использования «Зарегистрироваться»

Для варианта использования «Искать книги» была составлена диаграмма последовательности, представленная на рисунке 13.

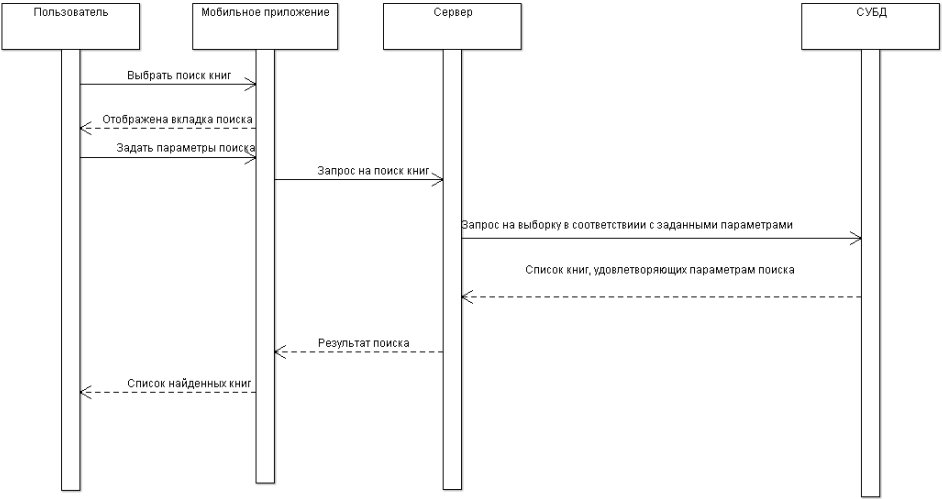


Рисунок 13 – Диаграмма последовательности варианта использования «Искать книги»

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

Разработка серверной части приложения должна быть выполнена на языке программирования C#, в среде программирования Microsoft Visual Studio 2019. Данные на сервере хранятся в базе данных, реализованной с использованием СУБД Microsoft SQL Server, запросы к базе данных осуществляются на языке SQL. Серверная должна работать под управлением операционной системы Windows 7 и выше.

Клиентская часть приложения должна быть реализована на языке программирования Java с использованием среды программирования Android Studio. Она должна представлять собой мобильное приложение с удобным и интуитивно понятным пользовательским интерфейсом. Клиентское приложение должно быть отлажено и должно успешно работать под управлением ОС Android 6.0 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 [16 – 20].

Задачи, представляющие собой этапы работы над проектом, представлены в таблице 2.

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ приложений-аналогов	7 дней	1
4	Анализ библиотечного фонда	10 дней	3
5	Консультация с руководством библиотеки	2 дня	4
6	Принятие окончательного решения	1 день	5
7	Анализ требований завершен	-	6
8	Выбор средств реализации проекта	-	-
9	Анализ возможных программных средств	4 дня	1
10	Выбор программных средств для реализации приложения	3 дня	9
11	Выбор средств завершен	-	10
12	Проектирование приложения	-	-
13	Проектирование базы данных	5 дней	7; 11
14	Проектирование архитектуры приложения	3 дня	13
15	Проектирование интерфейса пользователя	5 дней	7; 11
16	Проектирование приложения завершено	-	14; 15
17	Реализация проекта	-	-
18	Реализация базы данных	10 дней	13
19	Реализация серверной части приложения	15 дней	16;18
20	Реализация клиентской части приложения	20 дней	16;18
21	Реализация проекта завершена	-	19;20
22	Отладка	-	-
23	Тестирование работы серверной части приложения	5 дней	21
24	Тестирование работы клиентской части приложения	7 дней	21
25	Исправление ошибок	10 дней	23; 24
26	Отладка завершена	-	25
27	Внедрение программного средства	-	-
28	Внесение в систему электронных копий всех книг библиотечного фонда	7 дней	26
29	Обучение сотрудников библиотеки работе с системой	7 дней	28
30	Поиск и исправление ошибок при внедрении	10 дней	28
31	Внедрение завершено	-	29;30
32	Конец проекта	-	31

Таким образом, для реализации проекта нужно выполнить 32 задачи.

В таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист1	Трудовой	450 Р/ч	550 Р/ч
Программист2	Трудовой	350 Р/ч	450 Р/ч
Программист3	Трудовой	400 Р/ч	450 Р/ч
Дизайнер	Трудовой	300 Р/ч	350 Р/ч
Интернет	Затраты		
Компьютер	Материальный	500 Р	
Мобильное устройство	Материальный	200 Р	

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Анализ приложений-аналогов	Программист1 Компьютер Интернет	50% 1 шт
4	Анализ библиотечного фонда	Программист1 Программист2 Программист3 Компьютер Интернет	50% 50% 50% 1 шт
5	Консультация с руководством библиотеки	Программист1 Компьютер Интернет	50% 1 шт
6	Принятие окончательного решения	Программист1 Программист2 Программист3	50% 50% 50%
7	Анализ требований завершен		
8	Выбор средств реализации проекта		
9	Анализ возможных программных средств	Программист1 Программист2 Программист3 Компьютер Интернет	50% 50% 50% 1 шт
10	Выбор программных средств для реализации приложения	Программист1 Программист2 Программист3 Компьютер Интернет	50% 50% 50% 1 шт
11	Выбор средств завершен		
12	Проектирование приложения		
13	Проектирование базы данных	Программист1 Программист2 Компьютер Интернет	50% 25% 2 шт

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
14	Проектирование архитектуры приложения	Программист1 Программист2 Компьютер Интернет	50% 25% 2 шт
15	Проектирование интерфейса пользователя	Дизайнер Программист2 Программист3 Компьютер Интернет	50% 50% 75% 3 шт.
16	Проектирование приложения завершено		
17	Реализация проекта		
18	Реализация базы данных	Программист1 Программист2 Компьютер Интернет	100% 100% 2 шт.
19	Реализация серверной части приложения	Программист1 Программист2 Программист3 Компьютер Интернет	100% 100% 25% 3 шт.
20	Реализация клиентской части приложения	Программист1 Программист2 Программист3 Дизайнер Компьютер Мобильное устройство Интернет	25% 25% 100% 50% 4 шт. 2шт.
21	Реализация проекта завершена		
22	Отладка		
23	Тестирование работы серверной части приложения	Программист1 Программист2 Программист3 Компьютер Интернет	75% 75% 25% 3 шт.
24	Тестирование работы клиентской части приложения	Программист1 Программист2 Программист3 Компьютер Мобильное устройство Интернет	25% 25% 75% 3 шт. 1 шт.
25	Исправление ошибок	Программист1 Программист2 Программист3 Компьютер Интернет	75% 75% 75% 3 шт.
26	Отладка завершена		
27	Внедрение программного средства		
28	Внесение в систему электронных копий всех книг библиотечного фонда	Программист2 Программист3 Компьютер	50% 50% 1 шт.

Окончание таблицы 4

№	Название задачи	Ресурс	Единицы
30	Поиск и исправление ошибок при внедрении	Программист1 Программист2 Программист3 Компьютер Мобильное устройство Интернет	25% 50% 50% 2 шт. 1 шт.
31	Внедрение завершено		
32	Конец проекта		

Таким образом, реализацией проекта будут заниматься 3 программиста и 1 дизайнер, которым для работы нужны компьютеры, мобильные устройства и доступ к интернету. Для вычисления длительности реализации проекта и его критического пути построен сетевой график (рисунок 14).

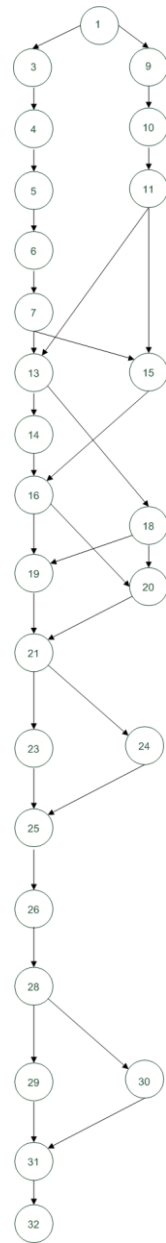


Рисунок 14 – Сетевой график проекта

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	3	0	7	0+7	7
5	4	7	10	7+10	17
6	5	17	2	17+2	19
7	6	19	1	19+1	20
9	1	0	0	0+0	0
10	9	0	4	0+4	4
11	10	4	3	4+3	7
13	7	20	0	Max(20+0;7+0)	20
	11	7	0		
14	13	20	5	20+5	25
15	7	20	0	Max(20+0;7+0)	20
	11	7	0		
16	14	25	3	Max(25+3;20+5)	28
	15	20	5		
18	13	20	5	20+5	25
19	16	28	0	Max(28+0;25+10)	35
	18	25	10		
20	16	28	0	Max(28+0;25+10)	35
	18	25	10		
21	19	35	15	Max(35+15;35+20)	55
	20	35	20		
23	21	55	0	55+0	55
24	21	55	0	55+0	55
25	23	55	5	Max(55+5;55+7)	62
	24	55	7		
26	25	62	10	62+10	72
28	26	72	0	72+0	72
29	28	72	7	72+7	79
30	28	72	7	72+7	79
31	29	79	7	Max(79+7; 79+10)	89
	30	79	10		
32	31	89	0	89+0	89

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		
32	-	-	-	-	89
31	32	89	0	89-0	89
30	31	89	10	89-10	79
29	31	89	7	89-7	82
28	29	82	7	Min(82-7; 79-7)	72
	30	79	7		
26	28	72	0	72-0	72
25	26	72	10	72-10	62
24	25	62	7	62-7	55
25	26	72	10	72-10	62
24	25	62	7	62-7	55
23	25	62	5	62-5	57
21	24	55	0	Min(55-0;57-0)	55
	23	57			
20	21	55	20	55-20	35
19	21	55	15	55-15	40
18	20	35	10	Min(35-10;40-10)	25
	19	40			
16	20	35	0	Min(35-0;40-0)	35
	19	40			
15	16	35	5	35-5	30
15	16	35	5	35-5	30
14	16	35	3	35-3	32
13	18	25	5	Min(25-5;32-5)	20
	14	32			
11	15	30	0	Min(30-0;20-0)	20
	13	20			
10	11	20	4	20-4	16
9	10	16	3	16-3	13
7	15	30	0	Min(30-0;20-0)	20
	13	20			
6	7	20	1	20-1	19
5	6	19	2	19-2	17
4	5	17	10	17-17	7
3	4	7	7	7-7	0
1	3	0	0	Min(0-0;13-0)	0
	9	13			

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	7	7	0
5	17	17	0
6	19	19	0
7	20	20	0
9	0	13	13
10	4	16	12
11	7	20	13
13	20	20	0
14	25	32	7
15	20	30	10
16	28	35	7
18	25	25	0
19	35	40	5
20	35	35	0
21	55	55	0
23	55	57	2
24	55	55	0
25	62	62	0
26	72	72	0
28	72	72	0
29	79	82	3
30	79	79	0
31	89	89	0
32	89	89	0

На основе полученных можно сделать вывод, что критический путь проходит через работы с номерами 1, 3, 4, 5, 6, 7, 13, 18, 20, 21, 24, 25, 26, 28, 30, 31, 32, а общее время выполнения проекта составляет 89 дней. На рисунке 15 представлен граф проекта, на котором штриховыми линиями указан критический путь.

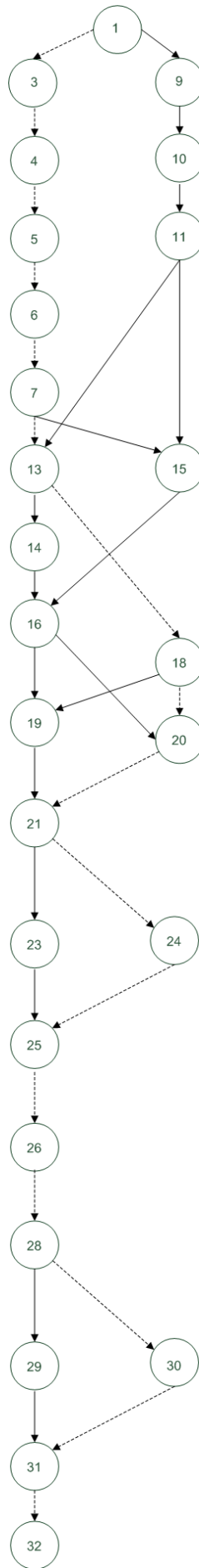


Рисунок 15 – Критический путь проекта

Для того, чтобы проверить корректность расчетов продолжительности проекта, а также вычислить величину его бюджета, воспользуемся системой управления проектами Microsoft Project, которая позволяет автоматически произвести все необходимые вычисления [21 – 24]. Список задач из таблицы 3, внесенный в систему Microsoft Project, представлен на рисунке 16. В качестве времени начала проекта была принята дата 07.12.2020.

	Название задачи	Длительно	Начало	Окончание	Предшественники
1	Начало проекта	0 дней	Пн 07.12.20	Пн 07.12.20	
2	Анализ требований	20 дней	Пн 07.12.20	Пт 01.01.21	
3	Анализ приложений-аналогов	7 дней	Пн 07.12.20	Вт 15.12.20	1
4	Анализ библиотечного фонда	10 дней	Ср 16.12.20	Вт 29.12.20	3
5	Консультация с руководством библиотеки	2 дней	Ср 30.12.20	Чт 31.12.20	4
6	Принятие окончательного решения	1 день	Пт 01.01.21	Пт 01.01.21	5
7	Анализ требований завершен	0 дней	Пт 01.01.21	Пт 01.01.21	6
8	Выбор средств реализации проекта	7 дней	Пн 07.12.20	Вт 15.12.20	
9	Анализ возможных программных средств	4 дней	Пн 07.12.20	Чт 10.12.20	1
10	Выбор программных средств для реализации приложения	3 дней	Пт 11.12.20	Вт 15.12.20	9
11	Выбор средств завершен	0 дней	Вт 15.12.20	Вт 15.12.20	10
12	Проектирование приложения	8 дней	Пн 04.01.21	Ср 13.01.21	
13	Проектирование базы данных	5 дней	Пн 04.01.21	Пт 08.01.21	7;11
14	Проектирование архитектуры приложения	3 дней	Пн 11.01.21	Ср 13.01.21	13
15	Проектирование интерфейса пользователя	5 дней	Пн 04.01.21	Пт 08.01.21	7;11
16	Проектирование приложения завершенно	0 дней	Ср 13.01.21	Ср 13.01.21	14;15
17	Реализация проекта	30 дней	Пн 11.01.21	Пт 19.02.21	
18	Реализация базы данных	10 дней	Пн 11.01.21	Пт 22.01.21	13
19	Реализация серверной части приложения	15 дней	Пн 25.01.21	Пт 12.02.21	16;18
20	Реализация клиентской части приложения	20 дней	Пн 25.01.21	Пт 19.02.21	16;18
21	Реализация проекта завершенна	0 дней	Пт 19.02.21	Пт 19.02.21	19;20
22	Отладка	17 дней	Пн 22.02.21	Вт 16.03.21	
23	Тестирование работы серверной части приложения	5 дней	Пн 22.02.21	Пт 26.02.21	21
24	Тестирование работы клиентской части приложения	7 дней	Пн 22.02.21	Вт 02.03.21	21
25	Исправление ошибок	10 дней	Ср 03.03.21	Вт 16.03.21	23;24
26	Отладка завершенна	0 дней	Вт 16.03.21	Вт 16.03.21	25
27	Внедрение программного средства	17 дней	Ср 17.03.21	Чт 08.04.21	
28	Внесение в систему электронных копий всех книг библиотечного фонда	7 дней	Ср 17.03.21	Чт 25.03.21	26
29	Обучение сотрудников библиотеки работе с системой	7 дней	Пт 26.03.21	Пн 05.04.21	28
30	Поиск и исправление ошибок при внедрении	10 дней	Пт 26.03.21	Чт 08.04.21	28
31	Внедрение завершенно	0 дней	Чт 08.04.21	Чт 08.04.21	29;30
32	Конец проекта	0 дней	Чт 08.04.21	Чт 08.04.21	31

Рисунок 16 – Задачи проекта в среде Microsoft Project

В среде Microsoft Project на основе введенных данных была построена диаграмма Ганта. На ее основании можно понять, что критический путь проходит через задачи 1, 3, 4, 5, 6, 7, 13, 18, 20, 21, 24, 25, 26, 28, 30, 31, 32. Можно заметить, что вычисленный программой критический путь совпал с критическим путем, вычисленным вручную. Диаграмма представлена на рисунке 17.

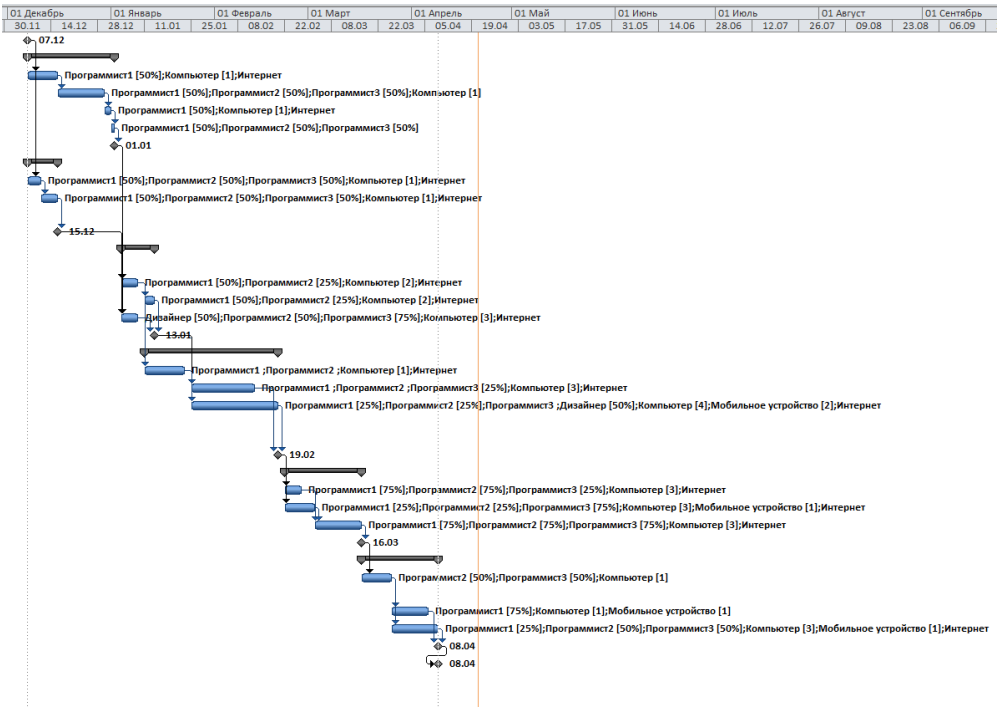


Рисунок 17 – Диаграмма Ганта

На рисунках 18 – 20 представлены графики трудовых ресурсов «Программист1», «Программист2», «Программист3».

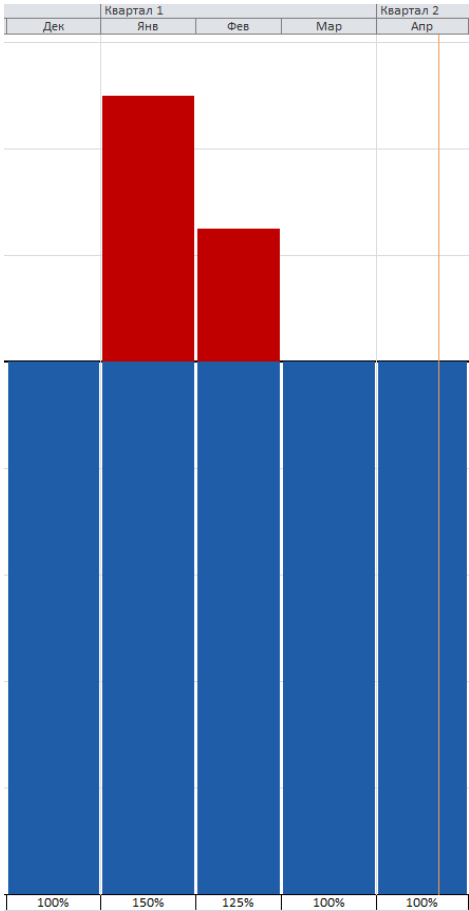


Рисунок 18 – График ресурсов для ресурса «Программист1»

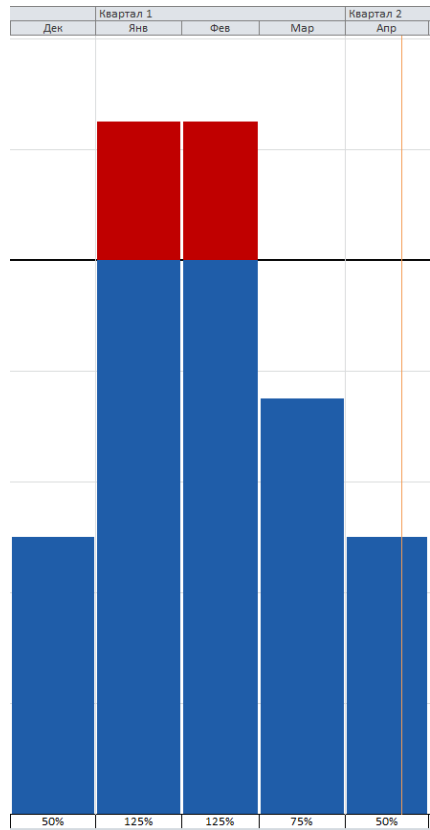


Рисунок 19 – График ресурсов для ресурса «Программист2»

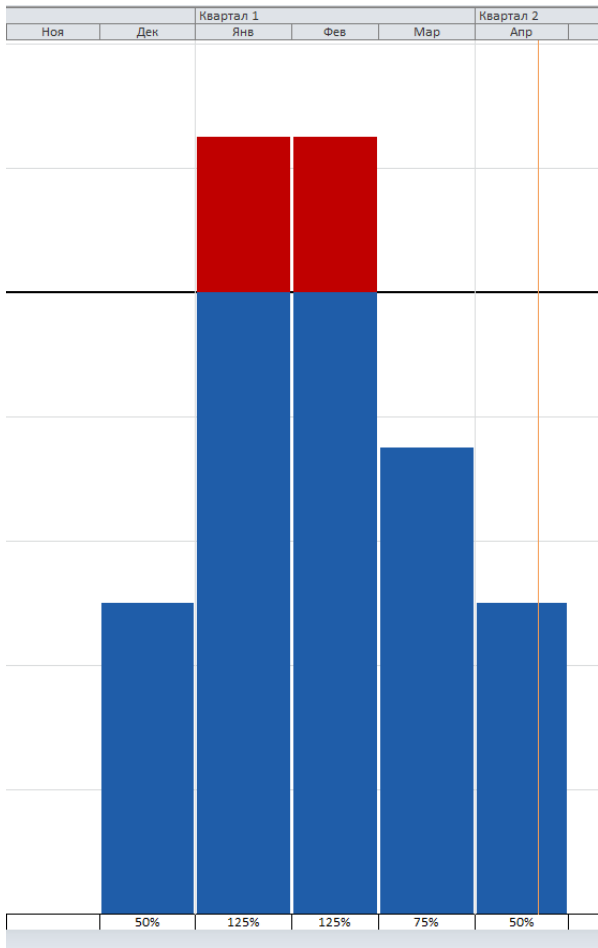


Рисунок 20 – График ресурсов для ресурса «Программист3»

На рисунке 21 представлен график трудового ресурса «Дизайнер».

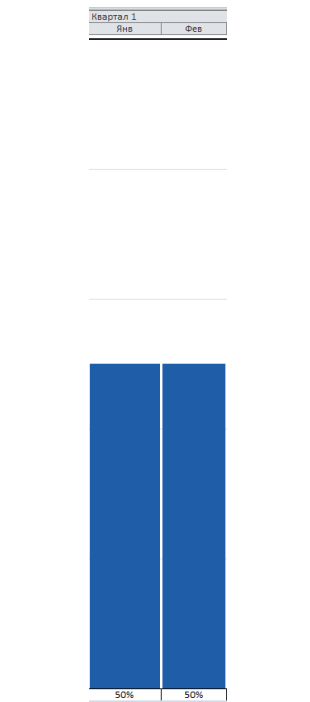


Рисунок 21 – График ресурсов для ресурса «Дизайнер»

На рисунке 22 представлен график материального ресурса «Компьютер».



Рисунок 22 – График ресурсов для ресурса «Компьютер»

На рисунке 23 представлен график материального ресурса «Мобильное устройство».

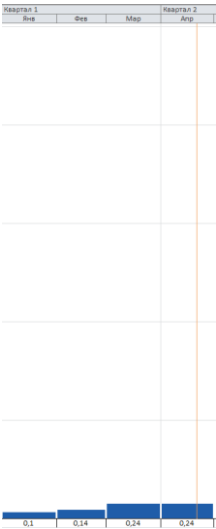


Рисунок 23 – График ресурсов для ресурса «Мобильное устройство»

Как можно заметить, у ресурсов «Программист1», «Программист2» и «Программист3» в январе и феврале наблюдается перегрузка.

На рисунке 24 представлен обзор проекта. Из рисунка видно, что общая длительность проекта составляет 89 дней (что совпадает со значением, вычисленным вручную), а предполагаемые затраты – 632 100 руб.

НИР

на дату Ср 21.04.21

Даты			
Начало:	Пн 07.12.20	Окончание:	Чт 08.04.21
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начал:	0 дней	Отклон. окончани	0 дней
Длительность			
Текущий план:	89 дней	Оставшаяся:	89 дней
Базовый план:	0 дней	Фактическая:	0 дней
Отклонение:	89 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	1 551,6 ч	Оставшиеся:	1 551,6 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 551,6 ч	Процент завершения:	0%
Затраты			
Текущий план:	632 100,00 Р	Оставшиеся:	632 100,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	632 100,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начаты задачи:	32	Трудовые ресурсы:	2
Выполняющиеся задач	0	Трудовые ресурсы с прев. доп	3
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	32	Итого ресурсов:	7

Рисунок 24 – Обзорный отчет проекта

На основе произведенных оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств разработки

В процессе разработки приложения было решено использовать методологию RUP. Разработка приложения в соответствии с методологией RUP носит итеративный характер. На каждой итерации перед разработчиками ставятся определенные цели, которые должны быть достигнуты к ее концу. В результате в конце каждой итерации должна получиться промежуточная функциональная версия разрабатываемого ПО. В методологии RUP существуют 4 основных фазы, каждая из которых может включать в себя одну или несколько итераций [25]:

- фаза начала проекта – формируется общее видение проекта и экономическое обоснование;
- уточнение – анализ предметной области и построение архитектуры продукта;
- построение – реализация основного функционала продукта;
- внедрение – получение финальной версии продукта и его передача заказчику, фаза включает в себя бета-тестирование и устранение выявленных в его процессе ошибок, а также обучение пользователей.

Каждая фаза завершается вехой, на которой команда разработчиков оценивает выполнение поставленных целей и принимает критически важные решения о ходе дальнейшей работы над продуктом.

Выбор в пользу данной методологии продиктован рядом ее достоинств. С одной стороны, методология RUP обеспечивает гибкость процесса разработки и позволяет быстро вносить коррективы в проект в случае выявления в нем ошибок или изменения требований к разрабатываемому приложению. Кроме того, благодаря данной методологии удастся выявить основные риски уже на ранних стадиях разработки. При этом RUP, в отличие от некоторых других популярных методологий, например XP, не требует постоянного взаимодействия с заказчиком, которое в рамках данного проекта невозможно.

Для анализа требований, а также моделирования структуры проекта в соответствии с принципами RUP было решено использовать диаграммы на языке моделирования UML [26]. Для построения диаграмм использовалось средство UML-моделирования ArgoUML.

На этапе планирования проекта было решено использовать приложение Microsoft Project. Оно позволяет обеспечить эффективное управление проектом на всех стадиях его выполнения. Microsoft Project предоставляет возможность выполнить структуризацию проекта, составить его календарный план, а также рассчитать затраты трудовых, материальных и денежных ресурсов в процессе работы над проектом.

Для создания мобильного приложения было решено использовать среду программирования Android Studio, предназначенную для разработки программ для операционной системы Android, и язык программирования Java. Серверную часть было решено разрабатывать на языке C# с использованием среды программирования Visual Studio 2019. Для хранения данных используется БД, работающая под управлением СУБД Microsoft SQL Server. Проектирование базы данных выполнялось с использованием приложения Erwin Data Modeler. Реализация базы данных производилась с использованием языка программирования SQL.

2.2 Проектирование архитектуры программного обеспечения

Было принято решение использовать в приложении двухуровневую архитектуру «клиент-сервер». Ее схема представлена на рисунке 25.



Рисунок 25 – Двухуровневая архитектура «клиент-сервер»

В данной архитектуре на сервере располагаются база данных и СУБД, а также серверная часть приложения, которая отвечает за непосредственное взаимодействие с базой данных и, как правило, реализуется в виде интерфейса прикладного программирования (API).

Клиентская же часть разрабатываемого ПО обеспечивает для пользователя удобный интерфейс, позволяющий пользователю просматривать необходимые ему данные из БД и взаимодействовать с ними, а также реализует возможность удаленно обращаться к серверу. В нашем случае его роль выполняет мобильное приложение.

Взаимодействие клиента и сервера осуществляется следующим образом: клиент делает HTTP-запрос серверу. Серверная часть приложения, в свою очередь, на основе полученного запроса создает SQL-запрос к базе данных. СУБД обрабатывает SQL-запрос и возвращает его результаты серверу, который на основе полученных результатов формирует ответ на запрос клиента. Получив ответ, клиент представляет его пользователю в удобной для того форме.

Достоинством архитектуры является то, что все данные хранятся на стороне сервера. Это позволяет минимизировать объем клиентского приложения. Кроме того, поскольку серверная часть приложения не зависит от клиентской, можно создать несколько вариантов клиента, отличающихся друг от друга, что может быть выгодно в случае потенциального распространения приложения и на другие платформы, помимо Android.

2.3 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был разработан интерфейс мобильного приложения. Он включает в себя страницы входа, регистрации, поиска книг, результатов поиска, списка загруженных книг, списка оставленных комментариев, смены пароля, информации о книге, информации о пользователе, добавления отзыва и блокировки пользователя.

Страница входа представлена на рисунке 26. На ней имеются следующие элементы: 1 – поле ввода логина пользователя, 2 – поле ввода

пароля пользователя, 3 – кнопка «Вход», 4 – кнопка перехода на страницу регистрации. При нажатии на кнопку «Вход» система проверяет, существует ли на самом деле пользователь с таким логином и паролем. Если такого пользователя нет, будет выведено сообщение о неверном логине или пароле. В противном случае будет проверено, не заблокирован ли данный пользователь. Если он заблокирован, выводится сообщение о блокировке. Иначе авторизация будет считаться выполненной успешно.

The diagram shows a login page layout within a rectangular frame. It contains four main elements, each with a number and a leader line pointing to it from the right:

- 1: Points to the first empty rectangular input field.
- 2: Points to the second empty rectangular input field.
- 3: Points to a rectangular button containing the text "Войти".
- 4: Points to a rectangular button containing the text "Регистрация".

Between the button labeled "Войти" and the button labeled "Регистрация", there is a line of text: "Еще не зарегистрировались?".

Рисунок 26 – Страница входа

Страница регистрации представлена на рисунке 27. На ней имеются следующие элементы: 1 – поле ввода логина пользователя, 2 – поле ввода пароля пользователя, 3 – кнопка «Регистрация», 4 – кнопка перехода на страницу входа. При нажатии кнопки «Регистрация» система проверяет, являются ли введенные логин и пароль допустимыми. Если это условие не выполняется, будет выведено сообщение о недопустимом логине или пароле. В противном случае будет проверено, не занят ли данный логин. Если он занят, выводится соответствующее сообщение. Иначе авторизация будет считаться выполненной успешно.

The diagram shows a vertical rectangular frame containing four input fields. The first two are empty text boxes. The third is a button labeled 'Регистрация'. The fourth is a button labeled 'Войти', preceded by the text 'Уже зарегистрировались?'. Numbered lines point to each of these four elements from the right side.

Рисунок 27 – Страница регистрации

Страница поиска представлена на рисунке 28. На ней присутствуют следующие элементы: 1 – поле ввода названия требуемой книги (или его части), 2 – место для ввода списка имен авторов, 3 – выпадающий список выбора жанра, 4 – выпадающий список выбора издательства, 5 – поле ввода года издания, 6 – место для ввода списка тегов требуемых книг, 7 – выпадающий список выбора типа сортировки, 8 – кнопка «Поиск», запускающая процесс, 9 – кнопка выхода из системы, 10 – кнопка смены пароля, 11 – навигационная панель, 12 – кнопка добавления в систему новой книги (доступна только администраторам).

The diagram shows a search page layout. At the top, there is a header bar with three buttons: 'Логин пользователя', 'Выход из системы' (labeled 9), and 'Смена пароля' (labeled 10). Below the header is a main search area containing seven input fields. The first two are empty text boxes. The next two are dropdown menus with a downward arrow icon. The fifth is an empty text box. The sixth is an empty text box. The seventh is a dropdown menu with a downward arrow icon. Below these fields is a button labeled 'Поиск' (labeled 8). At the bottom of the page is a navigation bar (labeled 11) with three buttons: 'Поиск', 'Скачанные книги', and 'Мои комментарии'. Below the navigation bar is a button labeled 'Добавить новую книгу' (labeled 12).

Рисунок 28 – Страница поиска

При нажатии на кнопку «Поиск» произойдет переход на страницу результатов поиска, представленную на рисунке 29. На данной странице имеются следующие элементы: 1 – область вывода списка скачанных книг, 2 – кликабельный элемент списка, который отображает основную информацию о одной из найденных книг и дает возможность перейти на страницу с более полными сведениями о ней.

Название		
Жанр		
Авторы		
Издательство	Год издания	
Кол-во скачиваний	Кол-во отзывов	Средняя оценка
Теги		

Рисунок 29 – Страница результатов поиска

На странице списка скачанных книг, которая представлена на рисунке 30, отображается полный список книг, загруженных пользователем. На данной странице имеются следующие элементы: 1 – область вывода списка скачанных книг, 2 – кликабельный элемент списка, аналогичный такому же со страницы результатов поиска, 3 – навигационная панель, 4 – кнопка выхода из системы, 5 – кнопка смены пароля, 6 – кнопка добавления в систему новой книги (доступна только администраторам).

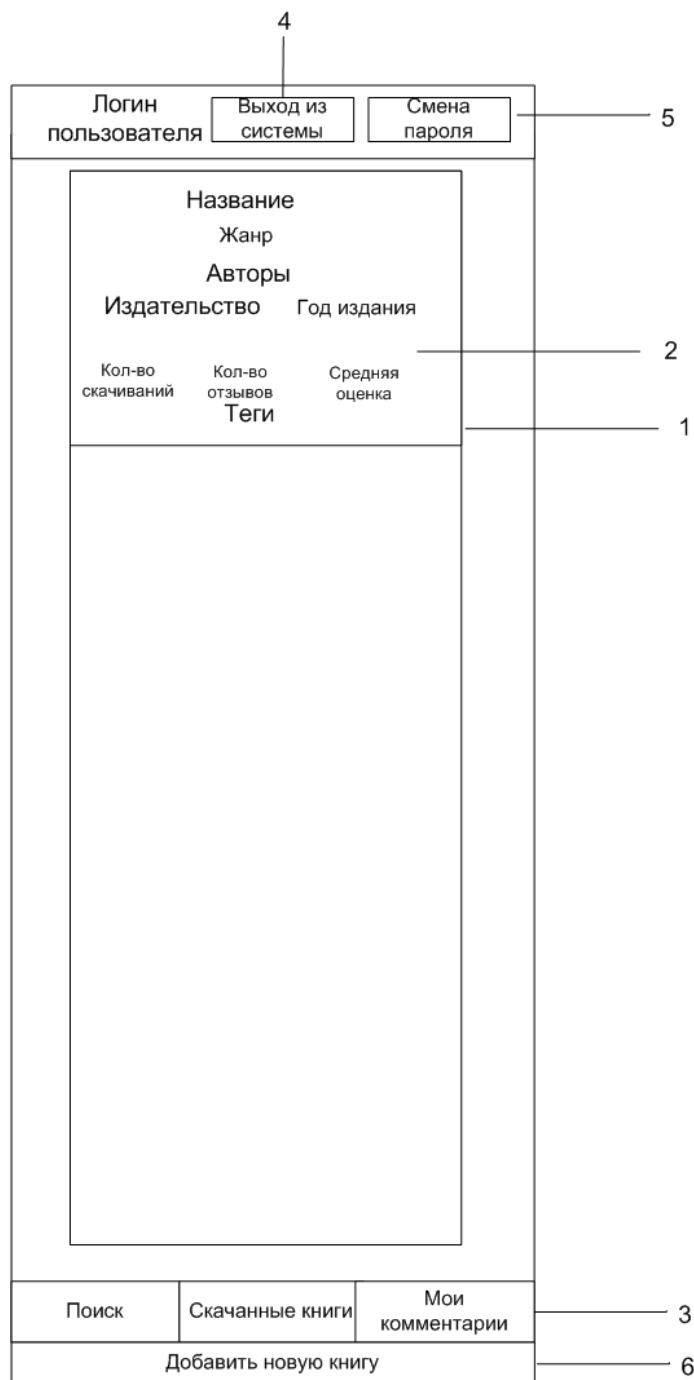


Рисунок 30 –Страница списка скачанных книг

Страница списка комментариев пользователя представлена на рисунке 31. На ней присутствуют следующие элементы: 1 – область вывода списка комментариев, 2 – отдельный элемент списка комментариев, 3 – ссылка на страницу автора данного комментария из списка, 4 – кнопка удаления комментария, 5 – навигационная панель, 6 – кнопка выхода из системы, 7 – кнопка смены пароля, 8 – кнопка добавления в систему новой книги (доступна только администраторам).

The diagram illustrates the layout of the 'User Comments' page. It features a top navigation bar (6) with buttons for 'Логин пользователя' (7), 'Выход из системы', and 'Смена пароля'. Below this is a header section (3) containing 'Логин автора', 'Оценка', 'Время создания', and 'Текст'. The main content area (1) is a large empty box for the comment. At the bottom, there is a footer bar (5) with buttons for 'Поиск', 'Скачанные книги', and 'Мои комментарии'. A button labeled 'Добавить новую книгу' (8) is located at the very bottom of the page.

Рисунок 31 – Страница комментариев пользователя

Страница добавления новой книги представлена на рисунке 32. Доступ к ней имеется лишь у администраторов электронной библиотеки. На ней присутствуют следующие элементы: 1 – поле ввода названия требуемой книги (или его части), 2 – место для ввода списка имен авторов, 3 – выпадающий список выбора жанра, 4 – выпадающий список выбора издательства, 5 – поле ввода года издания, 6 – место для ввода списка тегов требуемых книг, 7 – поле ввода текста аннотации книги, 8 – место задания файла добавляемой книги, 9 – кнопка «Добавить книгу», запускающая процесс добавления новой книги в систему. При нажатии кнопки «Добавить книгу» система проверяет задание всех обязательных атрибутов, а также файла книги. Если что-то из вышеперечисленного отсутствует, выводится сообщение об ошибке. В противном случае выполняется проверка на то, не существует ли в системе книга с такими же названием, издательством и годом издания. При ее наличии выводится сообщение о дубликate. В противном случае информация о книге и ее файл сохраняются на сервере.

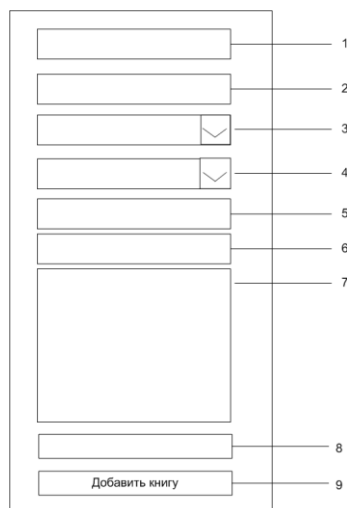


Рисунок 32 – Страница добавления новой книги

Страница смены пароля представлена на рисунке 33. На ней имеются следующие элементы: 1 – поле ввода старого пароля пользователя, 2 – поле ввода нового пароля пользователя, 3 – поле для повторения нового пароля, 4 – кнопка «Сменить пароль». При нажатии кнопки «Сменить пароль» система проверяет, являются ли введенный текущий пароль верным, а значения нового пароля, введенные в поля 1 и 2, – корректными и допустимыми. Если это одно из этих условий не выполняется, будет выведено сообщение об ошибке. В противном случае будут произведены смена пароля текущего пользователя и последующий возврат пользователя на предыдущую страницу.

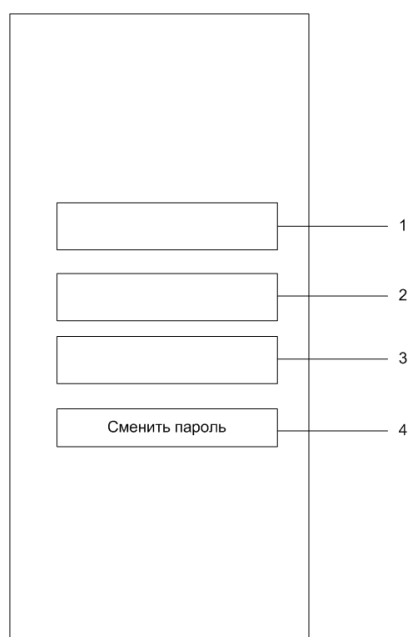


Рисунок 33 – Страница смены пароля

Если пользователь желает нажать на любой из элементов в любом списке книг, откроется страница с информацией о выбранной книге, представленная на рисунке 34.

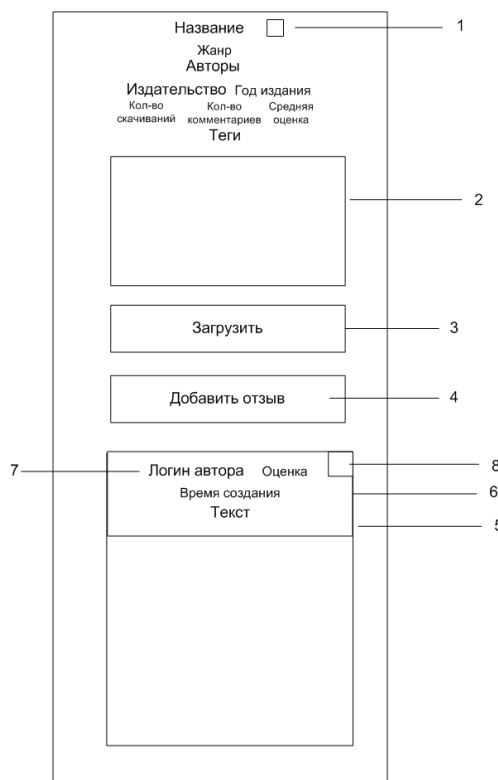


Рисунок 34 – Страница «Информация о книге»

На рисунке отмечены следующие компоненты страницы: 1 – кнопка удаления книги (доступна только автору и администраторам), 2 – область вывода аннотации к книге, 3 – кнопка «Загрузить», запускающая скачивание книги, 4 – кнопка «Добавить книгу», 5 – область вывода списка комментариев к книге, 6 – отдельный элемент списка комментариев, 7 – ссылка на страницу автора данного комментария из списка, 8 – кнопка удаления комментария (доступна только автору выбранного комментария и администраторам).

Кнопка добавления отзыва доступна любым пользователям, которые еще не оставляли отзывов к данной книге. При ее нажатии открывается страница добавления отзыва (рисунок 35), на которой имеются следующие элементы: 1 – поле ввода значения оценки, 2 – поле ввода нового пароля пользователя, 3 – кнопка «Отправить».

Рисунок 35 – Страница добавления отзыва

Пользователь должен выбрать оценку для книги и ввести текст комментария (при желании). После этого он должен нажать на кнопку «Отправить», что приведет к сохранению отзыва с добавлением текста комментария (при его наличии) в список комментариев о книге.

При нажатии на кнопку удаления книги произойдет ее удаление с последующим возвратом к странице поиска книг.

Нажатие на логин автора любого комментария позволяет открыть страницу с информацией об этом пользователе (рисунок 36), на которой представлена основная сведения о нем.

Рисунок 36 – Страница «Информация о пользователе»

Стоит заметить, что кнопка «Заблокировать» отображается только у администраторов и только на страницах пользователей, не являющихся администраторами и не имеющих активной блокировки. Ее нажатие приводит к открытию окна «Блокировка пользователя» (рисунок 37). Здесь имеются следующие элементы: 1 – поле ввода срока блокировки в днях, 2 – кнопка «Заблокировать».

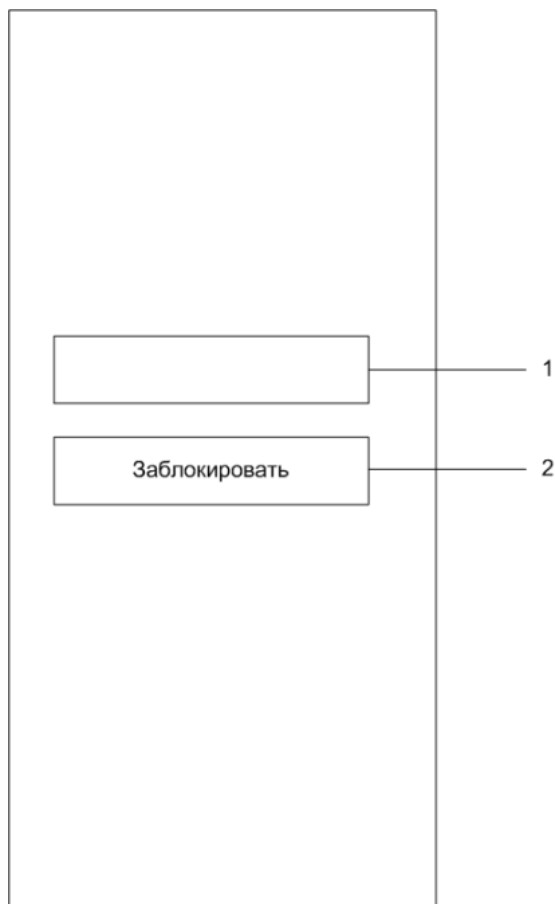


Рисунок 37 – Страница «Блокировка пользователя»

Администратор должен задать срок блокировки в днях и нажать на кнопку «Заблокировать», после чего выбранный пользователь будет заблокирован.

2.4 Логическое проектирование базы данных

На основе анализа предметной области и функциональных требований была построена концептуальная модель базы данных [27]. Она представлена в виде диаграммы «сущность-связь» (рисунок 38).

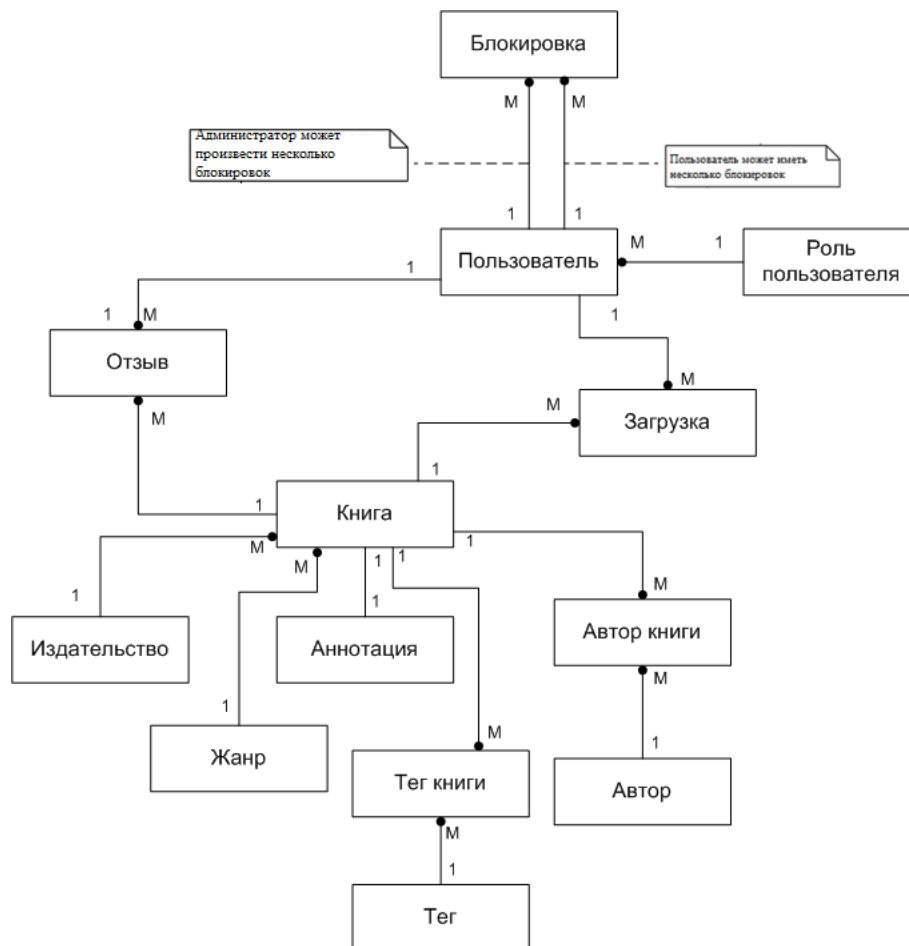


Рисунок 38 – Диаграмма «сущность-связь»

Сущности концептуальной модели данных описаны в таблице 8.

Таблица 8 – Сущности предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Список пользователей системы	Стержневая	10000	В систему добавляются новые пользователи при регистрации
2	Роль пользователя	Список ролей пользователей системы	Стержневая	10	Обновляются очень редко, только в случае добавления в систему новых типов пользователей.
3	Книга	Список книг системы	Стержневая	10000	Администраторы могут добавлять книги в систему
4	Отзыв	Список отзывов	Стержневая	100000000	Пользователь может оставить отзыв о книге, включающий в себя оценку, которая может сопровождаться комментарием
5	Автор	Список авторов	Стержневая	10000	Новые авторы добавляются при добавлении новых книг, авторы которых отсутствуют в системе

Продолжение таблицы 8

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
6	Автор книги	Связь между авторами и книгами	Ассоциативная	100000000	Обновляется при добавлении новой книги
7	Жанр	Список жанров книг	Стержневая	100	Новые жанры добавляются при добавлении новых книг, жанры которых отсутствуют в системе
8	Издательство	Список издательств книг	Стержневая	1000	Новые издательства добавляются при добавлении новых книг, издательств которых нет в системе
9	Тег	Список тегов системы	Стержневая	10000	Новые теги добавляются при добавлении новых книг, теги которых отсутствуют в системе
10	Тег книги	Связь между тегами и книгами	Ассоциативная	100000000	Обновляется при добавлении новой книги
11	Загрузка	Список загрузок	Стержневая	100000000	Пользователи могут загружать книги
12	Блокировка	Список блокировок	Стержневая	10000	Администраторы могут блокировать других пользователей при выявлении нарушений
13	Аннотация	Список аннотаций	Стержневая	10000	Администраторы могут добавлять аннотации к книгам

В таблице 9 представлено описание связей между сущностями предметной области.

Таблица 9 – Описание связей между сущностями предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Пользователь	Загрузка	Каждый пользователь может производить множество загрузок/ Каждая загрузка производится одним пользователем	1:М	Необязательная (пользователь может не иметь загруженных книг)	При повторной загрузке книги пользователем новая запись не создается
2	Роль пользователя	Пользователь	Каждую роль может иметь множество пользователей/ Каждый пользователь может иметь только одну роль	1:М	Необязательная (роль может не относиться ни к одному пользователю)	
3	Книга	Загрузка	Каждая книга может быть загружена множество раз/ Каждая загрузка производится для одной книги	1:М	Необязательная (книга может не иметь загрузок)	При повторной загрузке книги пользователем новая запись не создается

Продолжение таблицы 9

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
4	Книга	Отзыв	Каждой книге может соответствовать множество отзывов / Каждый отзыв относится к одной книге	1:M	Необязательная (книга может не иметь отзывов)	Не может быть несколько отзывов от одного пользователя
5	Пользователь	Отзыв	Каждый пользователь может оставить множество отзывов/ Каждый отзыв оставляется одним пользователем	1:M	Необязательная (пользователь может не оставлять отзывов)	Пользователь может оставить к одной книге только один отзыв
6	Автор	Книга	Каждый автор может написать множество книг / У каждой книги может быть как один автор, так и несколько	M:M	Обязательная	
7	Издательство	Книга	Издательство может выпустить множество книг / Каждая книга выпускается конкретным издательством	1:M	Обязательная	
8	Жанр	Книга	Жанр может соответствовать множеству книг/ Каждая книга принадлежит одному жанру	1:M	Обязательная	
9	Книга	Тег	Каждая книга может иметь множество тегов / Каждый тег может соответствовать множеству книг	M:M	Обязательная	
10	Пользователь	Книга	Каждый пользователь может добавить несколько книг / Каждая книга может быть добавлена только одним пользователем	1:M	Необязательная (пользователь может не добавлять книг)	
11	Пользователь	Блокировка	Пользователь может иметь несколько блокировок / Каждая блокировка относится к конкретному пользователю	1:M	Необязательная (пользователь может не иметь блокировок)	Пользователь не может иметь несколько активных блокировок
12	Администратор	Блокировка	Каждый администратор может произвести множество блокировок/ Каждая блокировка может быть произведена только одним администратором	1:M	Необязательная (администратор может никого не блокировать)	
13	Книга	Аннотация	Каждая книга может иметь одну аннотацию / Каждая аннотация относится к одной книге	1:1	Необязательная (книга может не иметь аннотации)	

На основании концептуальной модели данных в среде Erwin Data Modeler была построена логическая модель базы данных. Ее структура описана диаграммой в нотации IDEF1X, представленной на рисунке 39.

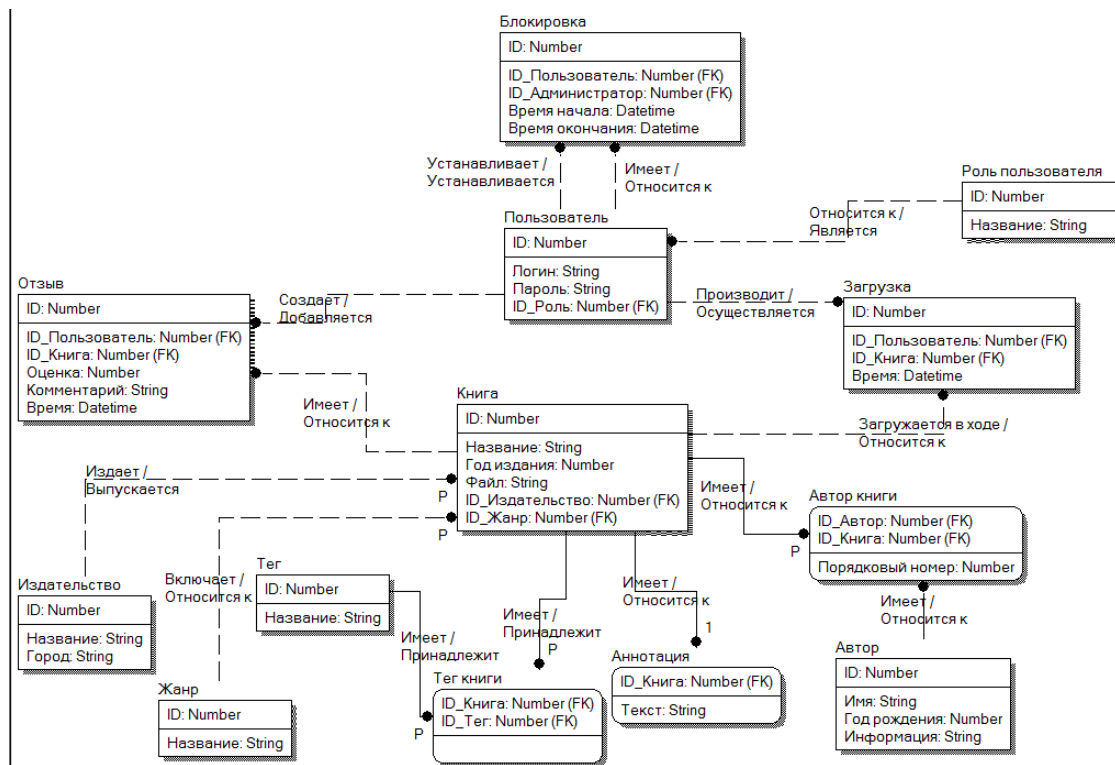


Рисунок 39 – Логическая модель базы данных

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

Серверная часть приложения представляет собой REST API. Доступ к данным осуществляется при помощи протокола HTTP с использованием его стандартных методов (GET применяется для получения данных ресурса, POST – для создания нового ресурса, PUT – для обновления существующего, DELETE – для удаления ресурса). Обращение к ресурсам происходит при помощи уникальных идентификаторов ресурсов (URI).

В приложение используется архитектура MVC (Model-View-Controller). Ее схема представлена на рисунке 40.

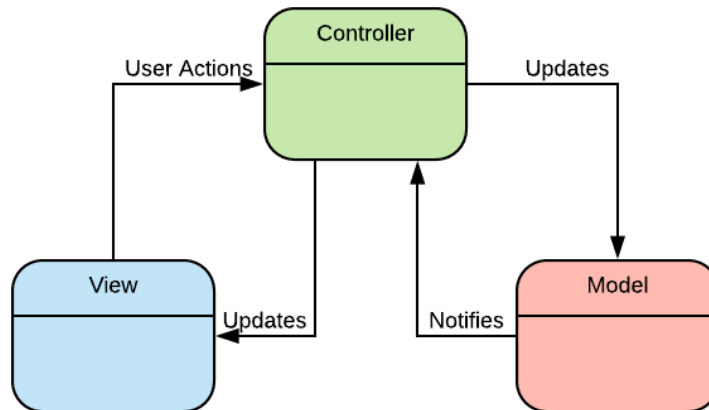


Рисунок 40 – Архитектура MVC

Серверная часть приложения включает в себя классы Модели, отвечающие за организацию доступа к данным (т.е. за непосредственное взаимодействие с БД) и классы Контроллера, отвечающие за организацию взаимодействия пользователя с клиентом. Именно они обрабатывают HTTP-запросы клиента и формируют ответы на них. К клиенту же отойдет только уровень Представления, отвечающий за взаимодействие с пользователем.

3 Реализация мобильного приложения

3.1 Реализация базы данных

На основе составленной ранее логической модели БД с учетом специфики СУБД Microsoft SQL Server при помощи системы Erwin была построена физическая модель. Ее структура представлена на рисунке 41.

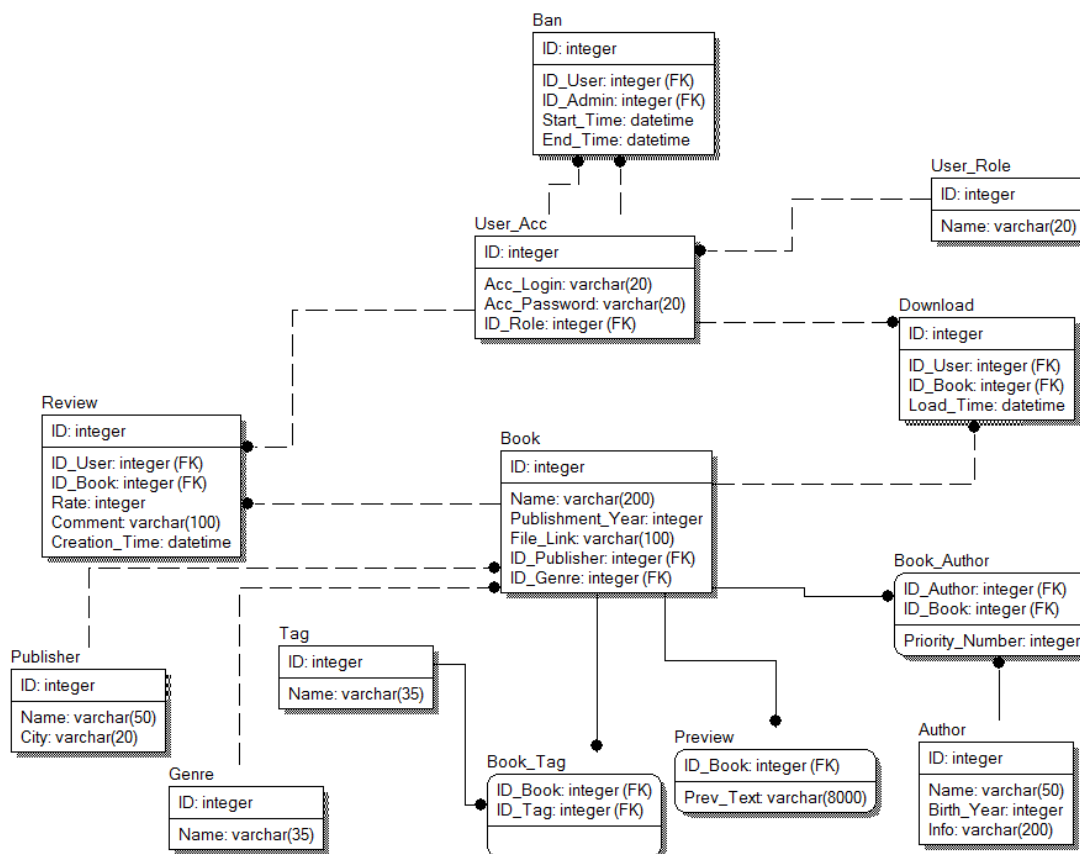


Рисунок 41 – Физическая модель базы данных

На основе составленной физической модели БД был сгенерирован скрипт на языке SQL для создания базы данных [28]. Его код представлен в приложении Б. Развертывание спроектированной базы данных было выполнено путем запуска данного скрипта с использованием утилиты SQL Server Management Studio, используемой для администрирования баз данных под управлением СУБД Microsoft SQL Server.

3.2 Разработка серверной части

Разработка серверной части приложения производилась в среде программирования Microsoft Visual Studio 2019 на языке C# с применением

технологии Web API [29 – 30]. Для взаимодействия с базой данных использовалась технология Entity Framework [31]. Диаграмма классов серверной части представлена на рисунке 41.

В соответствии с используемой архитектурой MVC все классы серверной части можно разделить на два слоя: слой модели, предназначенный для взаимодействия с БД, и слой контроллеров, предназначенный для обработки HTTP-запросов к серверу.

Диаграмма классов слоя модели представлена на рисунке 42. Все эти классы были автоматически сгенерированы средой программирования Visual Studio с использованием технологии Entity Framework. Генерация производилась на основе предварительно разработанной до этого базы данных (подход Database First) [32 – 33].

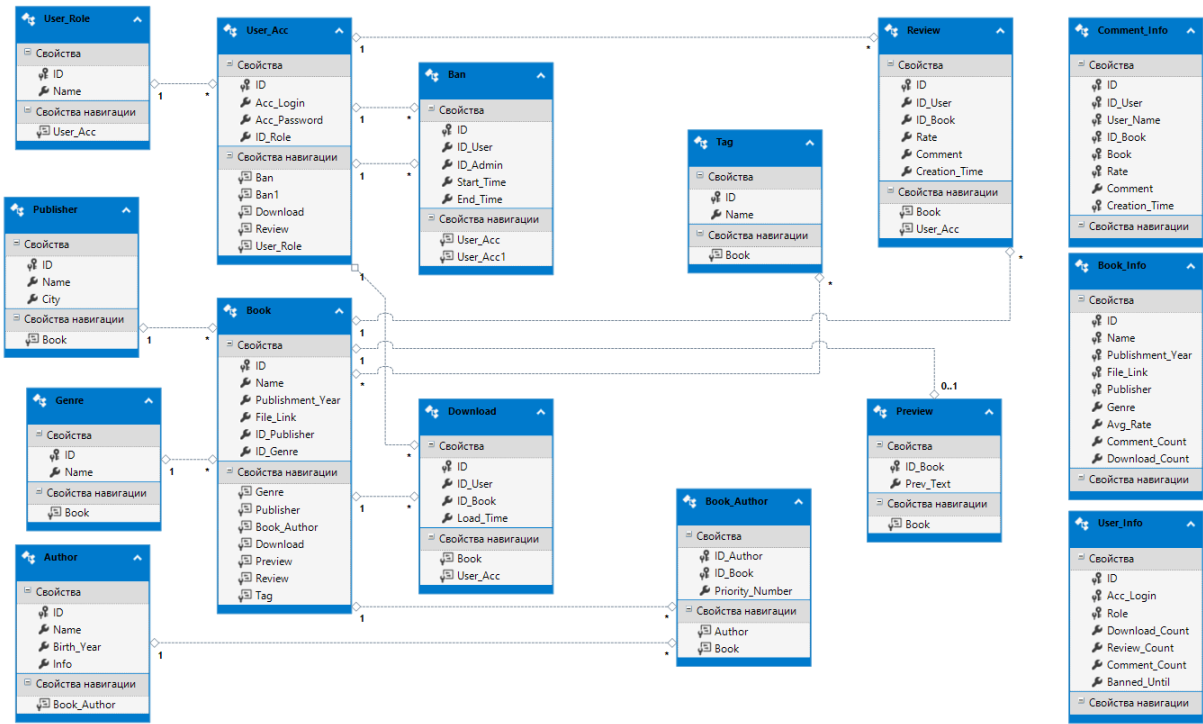


Рисунок 42 – Диаграмма классов серверной части (слой модели)

Как можно заметить, классы User_Role, User_Acc, Ban, Tag, Review, Publisher, Book, Preview, Genre, Author, Book_Author, Preview, Download были созданы на основании одноименных таблиц базы данных. Классы Comment_Info, Book_Info и User_Info же основываются на одноименных

представлениях и позволяют несколько облегчить выборку нужных данных из базы.

Диаграмма классов слоя контроллеров представлена на рисунке 43.

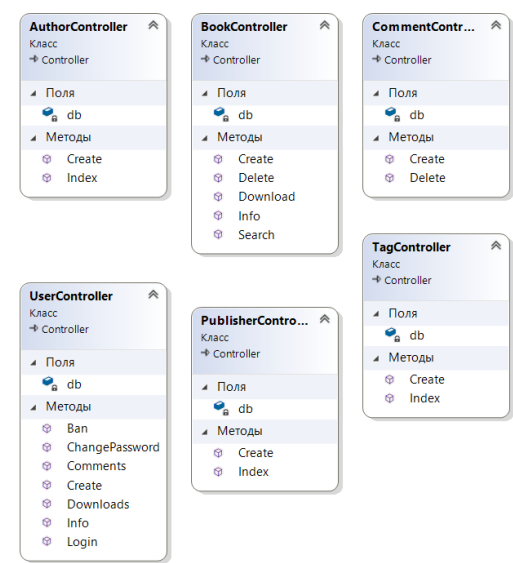


Рисунок 43 – Диаграмма классов серверной части (слой контроллеров)

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 10.

Таблица 10 – Описание классов слоя контроллеров

Класс	Назначение
AuthorController	Отвечает за обработку запросов на добавление в систему новых авторов и на получение списка имеющихся
BookController	Отвечает за обработку запросов на добавление в систему новых книг и на удаление уже имеющихся, а также на выборку книг в соответствии с некоторыми критериями, на получение полной информации о любой из имеющихся книг и на скачивание некоторой книги
CommentController	Отвечает за обработку запросов на добавление в систему новых комментариев и на удаление уже имеющихся
PublisherController	Отвечает за обработку запросов на добавление в систему новых издательств и на получение списка имеющихся

Продолжение таблицы 10

Класс	Назначение
TagController	Отвечает за обработку запросов на добавление в систему новых тегов и на получение списка имеющихся
UserController	Отвечает на обработку запросов на регистрацию нового пользователя и на вход в систему уже зарегистрированного, на смену пароля пользователя, на получение полных списков отзывов и загрузок некоторого пользователя, на получение информации о пользователе и на его блокировку.

Листинг серверной части представлен в приложении В.

3.3 Реализация клиентской части приложения

Разработка мобильного приложения производилась на языке Java с использованием среды программирования Android Studio. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 44.

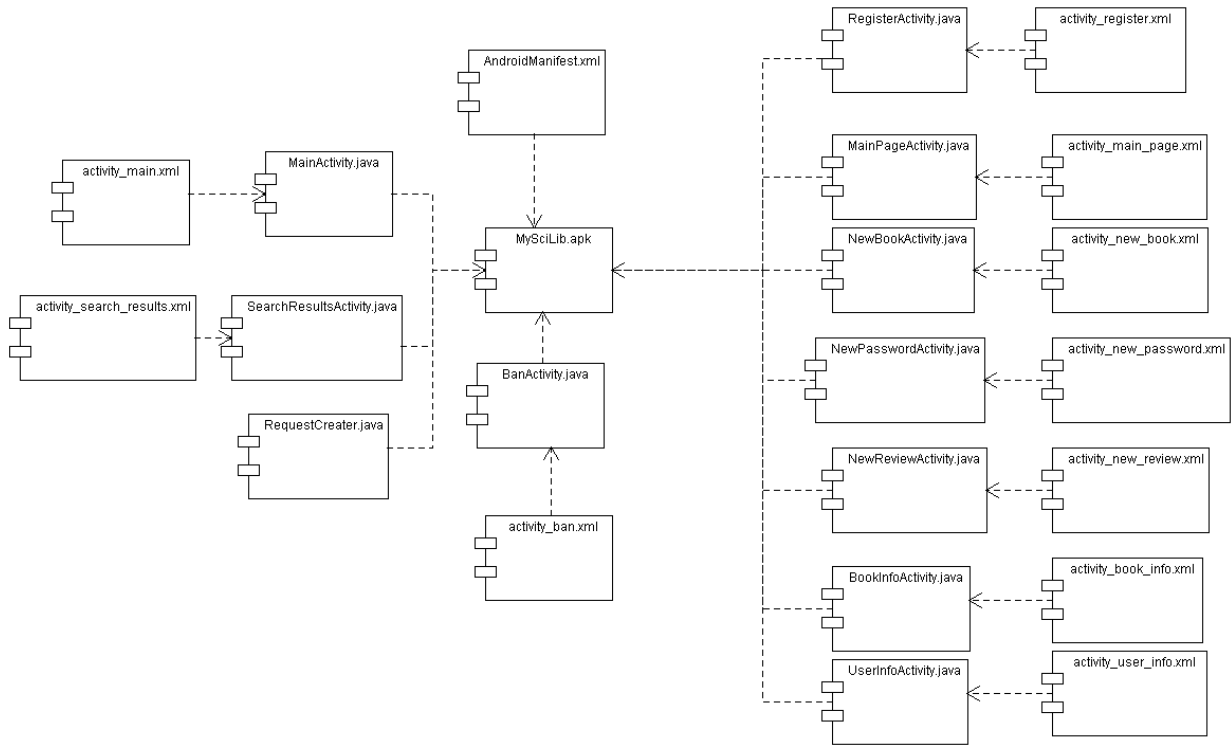


Рисунок 44 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 11.

Таблица 11 – Компоненты клиентского приложения

Компонент	Назначение
MainActivity.java	Файл класса начальной страницы приложения (страница входа в систему).
activity_main.xml	XML-файл, реализующий отображение страницы входа в систему.
RegisterActivity.java	Файл класса страницы регистрации.
activity_register.xml	XML-файл, реализующий отображение страницы регистрации.
MainPageActivity.java	Файл класса главной страницы приложения (объединяет в себе страницу поиска, страницу списка скачанных книг и страницу оставленных комментариев).
activity_main_page.xml	XML-файл, реализующий отображение главной страницы приложения.
NewBookActivity.java	Файл класса страницы добавления новой книги.
activity_new_book.xml	XML-файл, реализующий отображение страницы новой книги.
NewPasswordActivity.java	Файл класса страницы смены пароля.
activity_new_password.xml	XML-файл, реализующий отображение страницы смены пароля.
NewReviewActivity.java	Файл класса страницы добавления нового отзыва.
activity_new_review.xml	XML-файл, реализующий отображение страницы добавления нового отзыва.
BookInfoActivity.java	Файл класса страницы с информацией о книге.
activity_book_info.xml	XML-файл, реализующий отображение страницы с информацией о книге.
UserInfoActivity.java	Файл класса страницы с информацией о пользователе.
activity_user_info.xml	XML-файл, реализующий отображение страницы с информацией о пользователе.
BanActivity.java	Файл класса страницы блокировки пользователя.
activity_ban.xml	XML-файл, реализующий отображение страницы с информацией о книге.
SearchResultsActivity.java	Файл класса страницы результатов поиска.
activity_search_results.xml	XML-файл, реализующий отображение страницы результатов поиска.
RequestCreator.java	Файл класса, организующего запросы к серверу.
AndroidManifest.xml	XML-файл с параметрами проекта.
MySciLib.apk	Исполняемый файл приложения.

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 45.

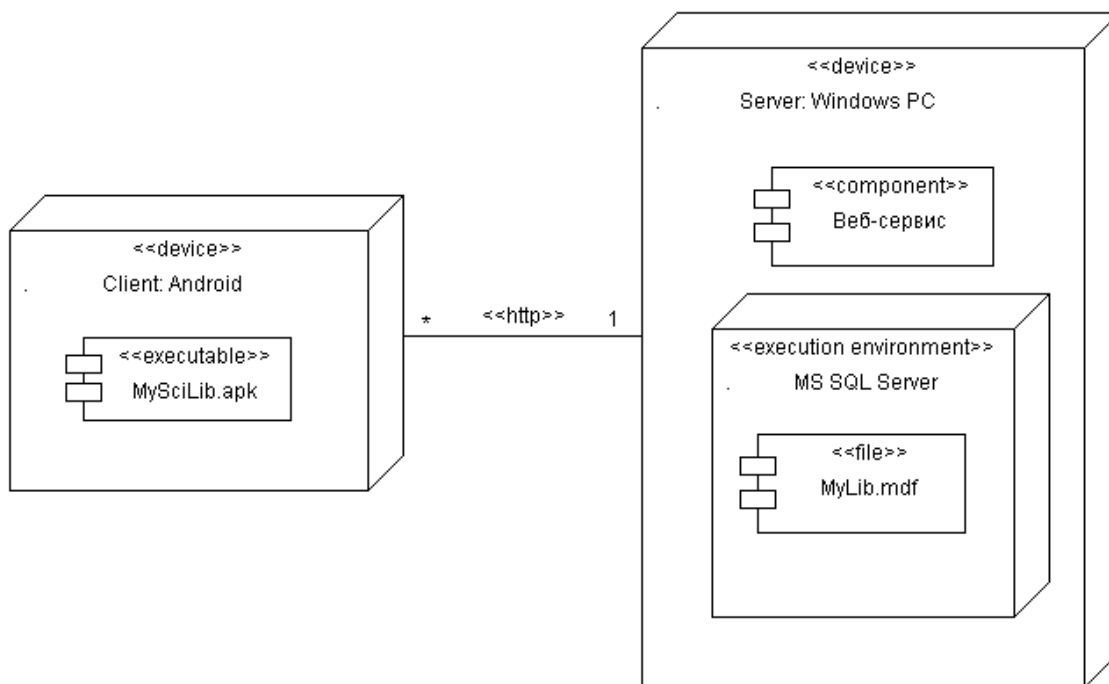


Рисунок 45 – Диаграмма развертывания мобильного приложения

Как можно увидеть на этой диаграмме, клиентское приложение работает на мобильном устройстве под управлением системы Android. API и база данных располагаются на сервере, работающем под управлением ОС Windows. Взаимодействие между серверной и клиентской частями приложения происходит с использованием протокола HTTP.

Листинг клиентской части программы представлен в приложении Г.

4 Тестирование программного продукта

4.1 Тестирование серверной части приложения

Для начала необходимо убедиться в корректности работы серверной части приложения. С этой целью было произведено тестирование серверной части приложения с использованием инструментального средства Postman. Результаты тестирования функционала серверной части представлены в таблице 12. Скриншоты тестов находятся в приложении Д.

Таблица 12 – Результаты тестирования серверной части

№	Описание теста	Запрос		Результат
		Метод	Параметры	
1	Добавление в систему пользователя с логином someuser и паролем password12	POST	login = someuser, password= password12	Тест пройден. Пользователь успешно добавлен в систему. Код ответа HTTP – 201 (рисунок Д.1).
2	Попытка входа в систему пользователя с логином someuser и паролем password12	GET	login = someuser, password= password 12	Тест пройден. Пользователь успешно авторизован. Код ответа HTTP – 200 (рисунок Д.2).
3	Получение сведений о пользователе с идентификатором 1	GET	id = 1	Тест пройден. Информация о пользователе успешно получена. Код ответа HTTP – 200 (рисунок Д.3).
4	Изменение пароля пользователя с идентификатором 8 с password12 на newpwd1234	POST	id= 8, oldpassword= password12, newpassword= newpwd1234	Тест пройден. Пароль успешно изменен. Код ответа HTTP – 200 (рисунок Д.4).
5	Получение списка комментариев, оставленных пользователем с идентификатором 1	GET	id=5	Тест пройден. Список комментариев, оставленных пользователем с идентификатором 1, успешно получен. Код ответа HTTP – 200 (рисунок Д.5).
6	Получение списка книг, загруженных пользователем с идентификатором 1	GET	id=5	Тест пройден. Список книг, загруженных пользователем с идентификатором 1, успешно получен. Код ответа HTTP – 200 (рисунок Д.6).
7	Блокировка пользователя с идентификатором 8 администратором с идентификатором 1 на 3 дня	POST	adminID=1, userID=8, length=3	Тест пройден. Пользователь успешно заблокирован. Код ответа HTTP – 200 (рисунок Д.7).

Продолжение таблицы 12

№	Описание теста	Запрос		Результат
		Метод	Параметры	
8	Добавление отзыва от пользователя с идентификатором 1 к книге с идентификатором 1.	POST	userID=1, bookID=1, comment=Познавательно, rate=5	Тест пройден. Отзыв успешно добавлен в систему. Код ответа HTTP – 201 (рисунок Д.8).
9	Удаление отзыва с идентификатором 23	DELETE	id = 23	Тест пройден. Отзыв успешно удален. Код ответа HTTP – 200 (рисунок Д.9).
10	Добавление информации о загрузке пользователем с идентификатором 1 книги с идентификатором 1.	POST	bookID=1, userID=1	Тест пройден. Информация о загрузке успешно добавлена. Код ответа HTTP – 201 (рисунок Д.10).
11	Получение списка издательств	GET		Тест пройден. Получен список издательств, сохраненных в системе. Код ответа HTTP – 200 (рисунок Д.11).
12	Добавление в систему издательства «Тольяттинский государственный университет» из города Тольятти	POST	name=Тольяттинский государственный университет, city=Тольятти	Тест пройден. Издательство успешно добавлено. Код ответа HTTP – 201 (рисунок Д.12).
13	Получение списка авторов	GET		Тест пройден. Получен список авторов, сохраненных в системе. Код ответа HTTP – 200 (рисунок Д.13).
14	Добавление в систему автора по имени Анастасия Новицкая	POST	name=Анастасия Новицкая	Тест пройден. Автор успешно добавлен. Код ответа HTTP – 201 (рисунок Д.14).
15	Получение списка тегов	GET		Тест пройден. Получен список тегов, сохраненных в системе. Код ответа HTTP – 200 (рисунок Д.15).
16	Добавление тега «С#»	POST	name=С#	Тест пройден. Тег успешно добавлен. Код ответа HTTP – 201 (рисунок Д.16).
17	Получение полной информации о книге с идентификатором 1	GET	id = 1	Тест пройден. Информация о книге успешно получена. Код ответа HTTP – 200 (рисунок Д.17).
18	Удаление книги с идентификатором 1	DELETE	id =1	Тест пройден. Книга успешно удалена. Код ответа HTTP – 200 (рисунок Д.18).

Окончание таблицы 12

19	Поиск книг издательства с идентификатором 6 жанра с идентификатором 5 с названием, включающим в себя «Проектирование программ», 2012 года издания.	POST	name=«Проектирование программ», genre=5, publisher=6, publisher=2012	Тест пройден. Получен список книг, подходящих условиям поиска. Код ответа HTTP – 200 (рисунок Д.19).
----	--	------	---	--

Все тесты были пройдены успешно, что свидетельствует о корректности работы серверной части приложения и позволяет перейти к тестированию клиентской части.

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 Е.

Таблица 13 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Ввести валидные данные для регистрации. Логин: mylogin. Пароль: qwerty1234.	Тест пройден. Пользователь зарегистрирован в системе (рисунок Е.1).
	Ввести данные с логином существующего в системе пользователя. Логин: mylogin.	Тест пройден. Выведено уведомление о том, что логин уже занят (рисунок Е.2).
	Попытаться задать логин и пароль с недопустимыми символами. Логин: логин. Пароль: пароль.	Тест пройден. Выведено уведомление о вводе недопустимых данных (рисунок Е.3).
	Попытаться задать логин и пароль недопустимой длины. Логин: m. Пароль: qwe1.	Тест пройден. Выведено уведомление о вводе недопустимых данных (рисунок Е.4).

Продолжение таблицы 13

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Произвести попытку задать пароль без букв. Логин: mylogin1 Пароль: qwertyuiop	Тест пройден. Выведено уведомление о вводе недопустимых данных (рисунок Е.5).
Авторизоваться	Ввести валидные данные. Логин: mylogin. Пароль: qwerty1234..	Тест пройден. Авторизация выполнена успешно (рисунок Е.6).
	Ввести неверные данные. Логин: mylogin. Пароль: qwerty123456.	Тест пройден. Выведено уведомление о неверном логине или пароле (рисунок Е.7).
	Ввести данные заблокированного пользователя.	Тест пройден. Выведено сообщение о блокировке (рисунок Е.8).
Искать книги	Ввести поисковые данные: Название: Проектирование программ и программирование на С++. Авторы: Ольга Леонидовна Викентьева, Ольга Андреевна Полякова . Жанр: материал конференции. Издатель: НИЦ «Академия Естествознания». Год издания:2012. Теги: С++.	Тест пройден. Найдены книги «Проектирование программ и программирование на С++: объектно-ориентированное программирование», «Проектирование программ и программирование на С++: структурное программирование» (рисунок Е.9).
Добавить оценку и отзыв	Поставить оценку с отзывом. Оценка: 5. Отзыв: интересно .	Тест пройден. Оценка сохранена в системе, отзыв добавлен в список отзывов о книге (рисунок Е.10).
	Оставить оценку без отзыва. Оценка: 4.	Тест пройден. Оценка сохранена в системе (рисунок Е.11).
Добавить книгу	Добавить книгу со следующими данными: Автор: Анастасия Новицкая. Название: Информационные методы моделирования процессов в экономических системах. Жанр: материал конференции Издательство: Тольяттинский государственный университет. Год издания: 2021 Теги: моделирование, IDEF0, IDEF1X, UML.	Тест пройден. Книга успешно добавлена (рисунок Е.12).
	Попытаться добавить книгу без задания названия, авторов и тегов .	Тест пройден. Выдано сообщение о неполных данных (рисунок Е.13).

Окончание таблицы 13

Вариант использования	Тест и параметры	Результат
Добавить книгу	Попытаться добавить книгу без указания ее файла.	Тест пройден. Выдано сообщение об отсутствии файла книги (рисунок Е.14).
Скачать книгу	Загрузить через приложение файл книги, добавленной в предыдущем примере..	Тест пройден. Книга успешно открыта (рисунок Е.15).
Удалить отзыв	Удалить отзыв, оставленный в ходе предшествующих тестов.	Тест пройден. Отзыв успешно удален (рисунок Е.16).
Удалить книгу	Удалить книгу, добавленную в ходе предшествующих тестов.	Тест пройден. Книга успешно удалена (рисунок Е.17).
Заблокировать пользователя	Заблокировать пользователя mylogin на 3 дня.	Тест пройден. Пользователь успешно заблокирован (рисунок Е.18).

Функциональное тестирование показало корректность работы клиентской части мобильного приложения.

4.2.2 Модульное тестирование

Модульные тесты – автоматизированные тесты, позволяющие оценить корректность работы некоторой автономной единицы программного кода (обычно подпрограммы) путем сравнения фактического результата работы с ожидаемым. Основное достоинство модульных тестов состоит в том, что они позволяют быстро выявить и устранить ошибки, возникающие при модификации кода.

В рамках данного проекта было решено покрыть модульными тестами методы класса RegisterActivity isValidLogin и isValidPassword, проверяющие на корректность предложенные пользователем при регистрации значения логина и пароля соответственно.

Для проведения модульных тестов был разработан тестовый класс ValidationTest, код которого представлен в листинге 1.

Листинг 1 – Исходный код модульных тестов

```
package com.myscilib;
import org.junit.Test;
import static org.junit.Assert.*;

public class ValidationTest {
    @Test
    public void validLogin() {
        assertTrue(RegisterActivity.isValidLogin("valid_login_1234"));
    }
    @Test
    public void loginTooShort() {
        assertFalse(RegisterActivity.isValidLogin("1"));
    }
    @Test
    public void loginTooLong() {
        assertFalse(RegisterActivity.isValidLogin("validlogin_validlogin"));
    }
    @Test
    public void disallowedSymbolLogin() {
        assertFalse(RegisterActivity.isValidLogin("validlogin!"));
    }
    @Test
    public void validPassword() {
        assertTrue(RegisterActivity.isValidPassword("password1234"));
    }
    @Test
    public void passwordTooShort() {
        assertFalse(RegisterActivity.isValidPassword("pwd123"));
    }
    @Test
    public void passwordTooLong() {
        assertFalse(RegisterActivity.isValidPassword(
            "passwordpasswordpassword"));
    }
    @Test
    public void disallowedSymbolPassword() {
        assertFalse(RegisterActivity.isValidPassword("my_password"));
    }
    @Test
    public void noNumbersPassword() {
        assertFalse(RegisterActivity.isValidPassword("something"));
    }
    @Test
    public void noLettersPassword() {
        assertFalse(RegisterActivity.isValidPassword("1234567890"));
    }
}
```

Результат проведения модульных тестов представлен на рисунке 46.

ValidationTest (com.myscilib)	4 s 577 ms
loginTooShort	4 s 577 ms
loginTooLong	0 ms
passwordTooShort	0 ms
validLogin	0 ms
noNumbersPassword	0 ms
validPassword	0 ms
noLettersPassword	0 ms
passwordTooLong	0 ms
disallowedSymbolLogin	0 ms
disallowedSymbolPassword	0 ms

Рисунок 46 – Результаты модульного тестирования

Все тесты были успешно пройдены, что свидетельствует о корректной работе тестируемых методов.

4.3 Метрики кода

Метриками кода называются некоторые количественные величины, позволяющие подсчитать, насколько код программы обладает тем или иным свойством. Существует огромное количество самых разнообразных метрик, имеющих различное назначение. Наиболее известными являются следующие группы метрик:

- метрики размера программы (SLOC-метрики);
- метрики сложности потока управления программ (метрики Холстеда и цикломатическая сложность).

Рассмотрим метод `Login` класса `UserController`, отвечающий за обработку запроса на вход в систему существующего пользователя нового пользователя на сервере. Его код представлен в листинге 2.

Листинг 2 – Вход в систему

```
public ActionResult Login(string login, string password)
{
    // Поиск пользователя с заданными логином или паролем
    User_Acc acc = db.User_Acc.Where(u => u.Acc_Login == login
        && u.Acc_Password == password).First();
    //Пользователь существует
    if (acc != null)
    {
        User_Info user = db.User_Info.FirstOrDefault
            (u => u.ID == acc.ID);
        //Пользователь не заблокирован
        if (user.Banned_Until == null)
        {
            return Json(user, JsonRequestBehavior.AllowGet);
        }
        //Пользователь заблокирован
        else
        {
            return new HttpStatusCodeResult
                (HttpStatusCode.Forbidden);
        }
    }
    //Пользователь не существует
    return new HttpUnauthorizedResult();
}
```

Данный метод находится в серверной части приложения и запускается при попытке произвести вход пользователя в систему. Он проверяет существование пользователя с заданными логином и паролем. В случае нахождения такого пользователя он также выясняет, не заблокирован ли он, и, если блокировка отсутствует, подтверждает вход пользователя в систему. При неверном логине или пароле или наличии у пользователя активной блокировки отправляется соответствующее сообщение об ошибке.

Для данного метода будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.3.1 Метрики размера программы

Данные метрики представляют собой подсчет количества строк программного кода, а также числа строк с комментариями. Вычислим метрики размера программы для приведенного выше программного кода. Результаты представлены в таблице 14.

Таблица 14 – Метрики размера программы

Метрика	Количество
Количество строк кода	25
Строки-комментарии	5

Также вычислим процент комментариев. Он равен $(5/25)*100=20$. Это свидетельствует о достаточном количестве комментариев в коде для его хорошего понимания при отсутствии избыточных комментариев, переполняющих программу.

4.3.2 Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить 4 характеристики программы [34 – 36]:

- число уникальных операторов;
- число уникальных операндов;
- общее число операторов;
- общее число операндов.

С целью вычисления этих характеристик были составлен словарь операторов (таблица 15) и словарь операндов (таблица 16).

Таблица 15 – Словарь операторов

Оператор	Число операторов
if	2
==	4
;	5
()	10
=	2
{}	4
return	3
new	2
&&	1
=>	1
!=	1
+	1

Таблица 16 – Словарь операндов

Операнд	Число операндов
db	2
user	2
u	2
acc	2
login	1
password	1
null	2

Таким образом, число уникальных операторов программы $n1 = 12$, число уникальных операндов программы $n2 = 7$, общее число операторов в программе $N1 = 36$, общее число операндов в программе $N2 = 12$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них:

- словарь программы (n):

$$n = n1 + n2;$$

- длина программы(N):

$$N = N1 + N2;$$

- объем программы (V):

$$V = N * \log_2 n;$$

- сложность программы (D):

$$D = (n1/2) * (N2/n2);$$

- оценка интеллектуальных усилий (E):

$$E = D * V.$$

Рассчитаем словарь программы: $n = 12+7 = 19$.

Рассчитаем длину программы $N = 36+12=48$.

Рассчитаем объем программы: $V = 48 * \log_2 19 = 203,9$.

Рассчитаем сложность программы: $D = (12/2) * (12/7) = 20,29$.

Рассчитаем оценку интеллектуальных усилий: $E=20,29 * 360 = 4137,13$.

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

Заключение

В ходе выполнения выпускной квалификационной работы было разработано мобильное приложение для поддержки сервисов электронной научной библиотеки.

На этапе анализа требований и предметной области был выполнен анализ программных продуктов аналогичного назначения, в результате которого были выявлены их основные достоинства и недостатки, определены функциональные и нефункциональные требования к создаваемому приложению и произведен анализ целевой аудитории.

Было осуществлено планирование разработки приложения и оценка бюджета. Результаты планирования показали, что общая длительность проекта составляет 89 дней, а предполагаемые затраты на реализацию – 632 100 рублей, что свидетельствует об экономической целесообразности данного программного проекта.

На этапе проектирования была составлена логическая модель базы данных и разработана архитектура мобильного приложения. База данных была создана с использованием языка программирования SQL и развернута под управлением СУБД Microsoft SQL Server 2019. Серверная часть приложения была разработана на языке C# с использованием среды программирования Microsoft Visual Studio 2019. Клиентская часть была разработана на языке Java с использованием среды программирования Android Studio.

Было выполнено тестирование серверной и клиентской частей разрабатываемого приложения. Оно показало корректность его работы. Также рассчитаны метрики программного кода для оценки его качества.

В дальнейшем планируется опубликовать разработанное приложение в магазине мобильных приложений для системы Android Google Play и развернуть его серверную часть на облачном сервисе, а также создать версию мобильного приложения для операционной системы iOS.

Таким образом, в ходе выполнения выпускной квалификационной работы были успешно выполнены все поставленные задачи.

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: 12.05.2021).
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: 12.05.2021).
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: 12.05.2021).
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. Богданова, И.Ф. Электронные библиотеки: история и современность / И.Ф. Богданова, Н.Ф. Богданова // Информационное общество: образование, наука, культура и технологии будущего. – 2017. – №1. – С. 133 – 154.
6. Статистика по пользователям библиотеки // Научная электронная библиотека Elibrary.ru [Электронный ресурс]. URL: https://elibrary.ru/stat_users.asp (дата обращения: 12.05.2021).
7. О проекте elibrary.ru // Научная электронная библиотека Elibrary.ru [Электронный ресурс]. URL: https://elibrary.ru/elibrary_about.asp (дата обращения: 12.05.2021).

8. Что такое Национальная электронная библиотека (НЭБ)? // Казанский национальный исследовательский технологический университет [Электронный ресурс]. URL: <https://www.kstu.ru/event.jsp?id=97398> (дата обращения 12.05.2021).

9. Вопросы и ответы // НЭБ – национальная электронная библиотека [Электронный ресурс]. URL: <https://rusneb.ru/faq/> (дата обращения 12.05.2021).

10. О научной библиотеке открытого доступа// Бесплатная научная библиотека «КиберЛенинка» [Электронный ресурс]. URL: <https://cyberleninka.ru/about> (дата обращения: 12.05.2021).

11. Саханский, Ю. В. Современные электронные библиотеки: задачи их деятельности и современные пути их развития / Ю. В. Саханский, Д. Р. Касаева // Инновационная наука. – 2017. – №12. – С. 58 – 62.

12. Аудитория научной онлайн-библиотеки «КиберЛенинка» выросла на 80% и насчитывает 52 млн пользователей// Ведомости [Электронный ресурс]. URL: <https://kp.vedomosti.ru/news/2020/02/25/823785-kiberleninka-vishla-samookupaemost> (дата обращения: 15.05.2021).

13. Global Smartphone Market Share: By Quarter // CounterPoint Research [Электронный ресурс]. URL: <https://www.counterpointresearch.com/global-smartphone-share/> (дата обращения: 15.05.2021).

14. Xiaomi Overtakes Samsung In Russia Online Smartphone Sales in Q4 2020 // CounterPoint Research [Электронный ресурс]. URL: <https://www.counterpointresearch.com/xiaomi-overtakes-samsung-russia-online-smartphone-sales-q4-2020/> (дата обращения: 15.05.2021).

15. Статистика версий Android // Новости IT-индустрии [Электронный ресурс]. URL: <https://3dnews.ru/1008195> (дата обращения: 15.05.2021).

16. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.

17. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.
18. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.
19. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.
20. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. - СПб.: Питер, 2019. – 208 с.
21. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 стр.
22. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб:БХВ-Петербург, 2013. – 432 с.
23. Джонсон, Т., Четфилд К. Microsoft Project 2016. Шаг за шагом / Т.Джонсон, К.Четфилд. – М., Изд-во "КТК Галактика", 2018. – 566 с.
24. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.
25. Унифицированный процесс Rational (RUP) // Интернет-технологии.ру [Электронный ресурс]. – URL: <https://www.internet-technologies.ru/articles/newbie/unificirovannyy-process-rational-rup.html> (дата обращения: 25.05.2021).
26. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.
27. Ащеулов, И.О. Разработка базы данных для web-приложения электронной научной библиотеки / И.О. Ащеулов // Научные достижения и открытия 2021: сборник статей XX Международного научно-исследовательского конкурса. – Пенза: МЦНС «Наука и Просвещение», 2021. – С. 29 – 35.
28. Точилкина, Т.Е. Принципы создания информационных систем и моделирования бизнес-процессов с использованием пакета программ

AllFusion Modeling Suite. Часть II. Моделирование данных и проектирование баз данных с ERwin Data Modeler. Учебное пособие./ Т.Е. Точилкина – М.: Академия бюджета и казначейства, 2009 – 167 с.

29. Введение в Web API // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/mvc/12.1.php> (дата обращения: 25.05.2021).

30. Андерсон Р., Ларкин К., Уоссон М. Создание веб-API с помощью ASP.NET Core и Visual 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: 25.05.2021).

31. Руководство по Entity Framework // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/entityframework/2.4.php> (дата обращения: 25.05.2021).

32. Database First // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef/ef6/modeling/designer/workflows/database-first> (дата обращения: 25.05.2021).

33. Entity Framework 6. Использование Database First // Professor Web [Электронный ресурс]. URL: https://professorweb.ru/my/entity-framework/6/level1/1_6.php (дата обращения: 25.05.2021).

34. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 25.05.2021).

35. 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.

36. 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 – мобильная операционная система для смартфонов, планшетов и других мобильных устройств, разрабатываемая и выпускаемая американской компанией Google и используемая на подавляющем большинстве современных мобильных устройств.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

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

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных, работающей под управлением соответствующей системой управления базами данных, а также для управления этими данными.

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

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

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

Приложение Б
ИСХОДНЫЙ КОД СКРИПТОВ БАЗЫ ДАННЫХ
(рекомендуемое)

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

```

CREATE TABLE Author
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_ID_Author
        CHECK ( [ID]>=(1) AND [ID]<=(10000) ),
    Name varchar(50) NOT NULL
    CONSTRAINT Validation_Author_Name
        CHECK ( Name NOT LIKE '%[^А-я. !-']%' escape '!'),
    Birth_Year integer
    CONSTRAINT Author_Birth_Validation
        CHECK ( [Birth_Year]>=(1500) AND [Birth_Year]<=(2000) ),
    Info varchar(200)
);

CREATE TABLE User_Role
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_Role_ID
        CHECK ( [ID]>=(1) AND [ID]<=(10) ),
    Name varchar(20) NOT NULL
    CONSTRAINT Validation_Rolename
        CHECK (Name NOT LIKE '%[^А-зА-я!-]%' escape '!')
);

CREATE TABLE User_Acc
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_User_ID
        CHECK ( [ID]>=(1) AND [ID]<=(10000) ),
    Acc_Login varchar(20) NOT NULL UNIQUE
    CONSTRAINT Validation_Login
        CHECK ( Acc_Login NOT LIKE '%[^А-з0-9_]%' ),
    Acc_Password varchar(20) NOT NULL
    CONSTRAINT Validation_Password
        CHECK (Acc_Password NOT LIKE '%[^А-з0-9]%' AND LEN(Acc_Password)
>= 8),
    ID_Role integer NOT NULL FOREIGN KEY REFERENCES User_Role(ID)
    CONSTRAINT Validation_Role_Key
        CHECK ( [ID_Role]>=(1) AND [ID_Role]<=(10) )
);

CREATE TABLE Ban
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_ID_Ban
        CHECK ( [ID]>=(1) AND [ID]<=(10000) ),
    ID_User integer NOT NULL FOREIGN KEY REFERENCES User_Acc(ID)
    CONSTRAINT Blocked_User_ID_Validation
        CHECK ( [ID_User]>=(1) AND [ID_User]<=(10000) ),
    ID_Admin integer NOT NULL FOREIGN KEY REFERENCES User_Acc(ID)
    CONSTRAINT Ban_Admin_ID_Validation
        CHECK ( [ID_Admin]>=(1) AND [ID_Admin]<=(10000) ),
    Start_Time datetime NOT NULL
    CONSTRAINT Validation_Ban_Start
        CHECK ( [Start_Time] >= '01-01-2020' AND [Start_Time] <= '31-12-
2050'),
    End_Time datetime NOT NULL
    CONSTRAINT Validation_Ban_End
        CHECK ( [End_Time] >= '01-01-2020' AND [End_Time] <= '31-12-
2050')
);

CREATE TABLE Genre

```

```

(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_Genre
        CHECK ( [ID]>=(1) AND [ID]<=(100) ),
    Name varchar(35) NOT NULL
    CONSTRAINT Validation_Genre_Name
        CHECK ( Name NOT LIKE '%[^А-я !-]%' escape '!')
);

CREATE TABLE Publisher
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_Publisher_ID
        CHECK ( [ID]>=(1) AND [ID]<=(1000) ),
    Name varchar(20) NOT NULL,
    City varchar(20) NOT NULL
    CONSTRAINT Validation_City_Name
        CHECK ( City NOT LIKE '%[^А-я ''!-]%' escape '!')
);

CREATE TABLE Book
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_ID_Book
        CHECK ( [ID]>=(1) AND [ID]<=(10000) ),
    Name varchar(80) NOT NULL,
    Publication_Year integer NOT NULL
    CONSTRAINT Validation_Book_Publication
        CHECK ( [Publication_Year]>=(1980) AND [Publication_Year]<=(2050)
),
    File_Link varchar(100) NOT NULL,
    ID_Publisher integer NOT NULL FOREIGN KEY REFERENCES Publisher(ID)
    CONSTRAINT Validation_Book_Publisher_ID
        CHECK ( [ID_Publisher]>=(1) AND [ID_Publisher]<=(1000) ),
    ID_Genre integer NOT NULL FOREIGN KEY REFERENCES Genre(ID)
    CONSTRAINT Validation_Book_Genre_ID
        CHECK ( [ID_Genre]>=(1) AND [ID_Genre]<=(100) )
);

CREATE TABLE Book_Author
(
    ID_Author integer FOREIGN KEY REFERENCES Author(ID)
    CONSTRAINT Validation_ID_Book_Author
        CHECK ( [ID_Author]>=(1) AND [ID_Author]<=(10000) ),
    ID_Book integer FOREIGN KEY REFERENCES Book(ID) ON DELETE CASCADE
    CONSTRAINT Validation_ID_Author_Book
        CHECK ( [ID_Book]>=(1) AND [ID_Book]<=(10000) ),
    Priority_Number integer NOT NULL
    CONSTRAINT Validation_Author_Number
        CHECK ( [Priority_Number]>=(1) AND [Priority_Number]<=(10) ),
    PRIMARY KEY (ID_Author, ID_Book)
);

CREATE TABLE Download
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_ID_Load
        CHECK ( [ID]>=(1) AND [ID]<=(1000000000) ),
    ID_User integer NOT NULL FOREIGN KEY REFERENCES User_Acc(ID)
    CONSTRAINT Load_User_ID_Validation
        CHECK ( [ID_User]>=(1) AND [ID_User]<=(1000000000) ),
    ID_Book integer NOT NULL FOREIGN KEY REFERENCES Book(ID) ON DELETE
        CASCADE

```

```

CONSTRAINT Load_Book_ID_Validation
    CHECK ( [ID_Book]>=(1) AND [ID_Book]<=(10000) ),
Load_Time datetime NOT NULL
CONSTRAINT Validation_Date_Load
    CHECK ([Load_Time] >= '01-01-2020' AND [Load_Time] <= '31-12-
2050')
);

CREATE TABLE Preview
(
    ID_Book integer PRIMARY KEY FOREIGN KEY REFERENCES Book(ID) ON DELETE
    CASCADE
    CONSTRAINT Validation_ID_Prev_Book
        CHECK ( [ID_Book]>=(1) AND [ID_Book]<=(10000) ),
    Prev_Text varchar(8000) NOT NULL
);

CREATE TABLE Review
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Validation_ID_Review
        CHECK ( [ID]>=(1) AND [ID]<=(100000000) ),
    ID_User integer NOT NULL FOREIGN KEY REFERENCES User_Acc(ID)
    CONSTRAINT Validation_Review_User_ID
        CHECK ( [ID_User]>=(1) AND [ID_User]<=(10000) ),
    ID_Book integer NOT NULL FOREIGN KEY REFERENCES Book(ID) ON DELETE
    CASCADE
    CONSTRAINT Validation_Book_User_ID
        CHECK ( [ID_Book]>=(1) AND [ID_Book]<=(10000) ),
    Rate integer NOT NULL
    CONSTRAINT Validation_Rate
        CHECK ( [Rate]>=(1) AND [Rate]<=(5) ),
    Comment varchar(100),
    Creation_Time datetime NOT NULL
    CONSTRAINT Validation_Date_Review
        CHECK ( [Creation_Time] >= '01-01-2020' AND [Creation_Time]<=
'31-12-2050' )
);

CREATE TABLE Tag
(
    ID integer IDENTITY ( 1,1 ) PRIMARY KEY
    CONSTRAINT Avg_ID_Tag
        CHECK ( [ID]>=(1) AND [ID]<=(10000) ),
    Name varchar(35) NOT NULL UNIQUE
);

CREATE TABLE Book_Tag
(
    ID_Book integer FOREIGN KEY REFERENCES Book(ID) ON DELETE CASCADE
    CONSTRAINT Validation_ID_Tag_Book
        CHECK ( [ID_Book]>=(1) AND [ID_Book]<=(10000) ),
    ID_Tag integer FOREIGN KEY REFERENCES Tag(ID)
    CONSTRAINT Validation_ID_Book_Tag
        CHECK ( [ID_Tag]>=(1) AND [ID_Tag]<=(10000) ),
    PRIMARY KEY(ID_Book, ID_Tag)
);

```

Листинг Б.2 – Исходный код создания представления Book_Info (основная информация о книге)

```

CREATE VIEW Book_Info AS
SELECT Book.ID, Book.Name, Publishment_Year, File_Link, Publisher.Name AS
Publisher, Genre.Name AS Genre, Avg_Rate, Comment_Count, Download_Count

```

```

FROM Book JOIN Publisher ON Book.ID_Publisher = Publisher.ID
LEFT JOIN Genre ON Book.ID_Genre = Genre.ID
LEFT JOIN (SELECT ID_Book, AVG( CONVERT (float, Rate)) AS Avg_Rate,
COUNT (Comment) AS Comment_Count FROM Review GROUP BY ID_Book) AS
Reviews ON Book.ID = Reviews.ID_Book
LEFT JOIN (SELECT ID_Book, COUNT(*) AS Download_Count FROM Download
GROUP BY ID_Book) AS Downloads ON Book.ID = Downloads.ID_Book;

```

Листинг Б.3 – Исходный код создания представления Comment_Info (основная информация о комментариях)

```

CREATE VIEW Comment_Info AS
SELECT Review.ID, ID_User, Acc_Login AS User_Name, ID_Book, Book.Name AS
Book, Rate, Comment, Creation_Time
FROM Review JOIN User_Acc ON Review.ID_User = User_Acc.ID
JOIN Book ON Review.ID_Book = Book.ID

```

Листинг Б.4 – Исходный код создания представления User_Info (информация о пользователе)

```

CREATE VIEW User_Info AS
SELECT User_Acc.ID, Acc_Login, User_Role.Name AS Role, Download_Count,
Review_Count, Comment_Count, Banned_Until
FROM User_Acc JOIN User_Role ON User_Acc.ID_Role = User_Role.ID
LEFT JOIN (SELECT ID_User, COUNT(*) AS Download_Count FROM Download
GROUP BY ID_User) AS Downloads ON User_Acc.ID = Downloads.ID_User
LEFT JOIN (SELECT ID_User, COUNT(*) AS Review_Count, COUNT (Comment) AS
Comment_Count FROM Review GROUP BY ID_User) AS Reviews ON
User_Acc.ID = Reviews.ID_User
LEFT JOIN (SELECT ID_User, End_Time AS Banned_Until FROM Ban WHERE
End_Time > GETDATE()) AS Active_Bans ON User_Acc.ID =
Active_Bans.ID_User;

```

Листинг Б.5 – Исходный код хранимой процедуры Create_Ban (блокировка пользователя)

```

CREATE PROCEDURE Create_Ban
@User_ID int, @Admin_ID int, @Length int
AS
INSERT INTO Ban
(ID_User, ID_Admin, Start_Time, End_Time)
VALUES
(@User_ID, @Admin_ID, GETDATE(), DATEADD (DAY, @Length, GETDATE()))

```

Листинг Б.6 – Исходный код хранимой процедуры Create_Download (сохранение данных о загрузке)

```

CREATE PROCEDURE Create_Download
@User_Id Int, @Book_Id Int
AS
INSERT INTO Download
(ID_User, ID_Book, Load_Time)
VALUES
(@User_Id, @Book_Id, GETDATE())

```

Листинг Б.7 – Исходный код хранимой процедуры Create_Review
(сохранение отзыва)

```
CREATE PROCEDURE Create_Review
@User_Id int, @Book_Id int, @Rate int, @Comment VARCHAR(100) = NULL
AS
INSERT INTO Review
    (ID_User, ID_Book, Rate, Comment, Creation_Time)
VALUES
    (@User_Id, @Book_Id, @Rate, @Comment, GETDATE())
```

Приложение В
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

Листинг В.1 – Файл BookPage.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MyScienceLibrary.Models
{
    public class BookPage: BookPanel
    {
        public string Preview { get; set; }
        public List<Comment_Info> Comments { get; set; }
        public BookPage(MySciLibContext context, int id): base(context, id)
        {
            Preview = context.Preview.Find(id).Prev_Text;
            Comments = context.Comment_Info.Where(c => c.ID_Book ==
                id).ToList();
        }
    }
}
```

Листинг В.2 – Файл BookPanel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MyScienceLibrary.Models
{
    public class BookPanel
    {
        public Book_Info info { get; set; }
        public List<AuthorInfo> authors { get; set; }
        public List<TagInfo> tags { get; set; }
        public BookPanel(MySciLibContext context , int id)
        {
            info = context.Book_Info.FirstOrDefault(b => b.ID == id);
            var bookAuthors = context.Book_Author.Where(ba => ba.ID_Book ==
                id).OrderBy(ba => ba.Priority_Number);
            authors = new List<AuthorInfo>();
            foreach (Book_Author ba in bookAuthors)
            {
                authors.Add(new AuthorInfo(ba.Author));
            }
            tags = new List<TagInfo>();
            var bookTags = context.Book.Find(id).Tag;
            foreach (Tag t in bookTags)
            {
                tags.Add(new TagInfo(t));
            }
        }
    }
}
```

Листинг В.3 – Файл Publisher_Info.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```

namespace MyScienceLibrary.Models
{
    public class PublisherInfo
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public string City { get; set; }
        public PublisherInfo(Publisher p)
        {
            ID = p.ID;
            Name = p.Name;
            City = p.City;
        }
    }
}

```

Листинг В.4 – Файл Tag_Info.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MyScienceLibrary.Models
{
    public class TagInfo
    {
        public int ID { get; set; }
        public string Name { get; set; }
        public TagInfo(Tag t)
        {
            ID = t.ID;
            Name = t.Name;
        }
    }
}

```

Листинг В.5 – Файл User_Info.cs

```

namespace MyScienceLibrary.Models
{
    using System;
    using System.Collections.Generic;

    public partial class User_Info
    {
        public int ID { get; set; }
        public string Acc_Login { get; set; }
        public string Role { get; set; }
        public Nullable<int> Download_Count { get; set; }
        public Nullable<int> Review_Count { get; set; }
        public Nullable<int> Comment_Count { get; set; }
        public Nullable<System.DateTime> Banned_Until { get; set; }
    }
}

```

Листинг В.6 – Файл MyLibModel.context.cs

```

namespace MyScienceLibrary.Models
{
    using System;

```

```

using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Core.Objects;
using System.Linq;

public partial class MySciLibContext : DbContext
{
    public MySciLibContext()
        : base("name=MySciLibContext")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        throw new UnintentionalCodeFirstException();
    }

    public virtual DbSet<Author> Author { get; set; }
    public virtual DbSet<Ban> Ban { get; set; }
    public virtual DbSet<Book> Book { get; set; }
    public virtual DbSet<Book_Author> Book_Author { get; set; }
    public virtual DbSet<Download> Download { get; set; }
    public virtual DbSet<Genre> Genre { get; set; }
    public virtual DbSet<Preview> Preview { get; set; }
    public virtual DbSet<Publisher> Publisher { get; set; }
    public virtual DbSet<Review> Review { get; set; }
    public virtual DbSet<Tag> Tag { get; set; }
    public virtual DbSet<User_Acc> User_Acc { get; set; }
    public virtual DbSet<User_Role> User_Role { get; set; }
    public virtual DbSet<Book_Info> Book_Info { get; set; }
    public virtual DbSet<Comment_Info> Comment_Info { get; set; }
    public virtual DbSet<User_Info> User_Info { get; set; }

    public virtual int Create_Ban(Nullable<int> user_ID, Nullable<int>
        admin_ID, Nullable<int> length)
    {
        var user_IDParameter = user_ID.HasValue ?
            new ObjectParameter("User_ID", user_ID) :
            new ObjectParameter("User_ID", typeof(int));

        var admin_IDParameter = admin_ID.HasValue ?
            new ObjectParameter("Admin_ID", admin_ID) :
            new ObjectParameter("Admin_ID", typeof(int));

        var lengthParameter = length.HasValue ?
            new ObjectParameter("Length", length) :
            new ObjectParameter("Length", typeof(int));

        return ((IObjectContextAdapter)this).ObjectContext.
            ExecuteFunction("Create_Ban", user_IDParameter,
                admin_IDParameter, lengthParameter);
    }

    public virtual int Create_Download(Nullable<int> user_Id,
        Nullable<int> book_Id)
    {
        var user_IdParameter = user_Id.HasValue ?
            new ObjectParameter("User_Id", user_Id) :
            new ObjectParameter("User_Id", typeof(int));
        var book_IdParameter = book_Id.HasValue ?
            new ObjectParameter("Book_Id", book_Id) :
            new ObjectParameter("Book_Id", typeof(int));

        return ((IObjectContextAdapter)this).ObjectContext.
            ExecuteFunction("Create_Download", user_IdParameter,
                book_IdParameter);
    }
}

```

```

    }

    public virtual int Create_Review(Nullable<int> user_Id, Nullable<int>
        book_Id, Nullable<int> rate, string comment)
    {
        var user_IdParameter = user_Id.HasValue ?
            new ObjectParameter("User_Id", user_Id) :
            new ObjectParameter("User_Id", typeof(int));
        var book_IdParameter = book_Id.HasValue ?
            new ObjectParameter("Book_Id", book_Id) :
            new ObjectParameter("Book_Id", typeof(int));
        var rateParameter = rate.HasValue ?
            new ObjectParameter("Rate", rate) :
            new ObjectParameter("Rate", typeof(int));
        var commentParameter = comment != null ?
            new ObjectParameter("Comment", comment) :
            new ObjectParameter("Comment", typeof(string));

        return ((IObjectContextAdapter)this).ObjectContext.
            ExecuteFunction("Create_Review", user_IdParameter,
                book_IdParameter, rateParameter, commentParameter);
    }
}

```

Листинг В.7– Файл AuthorController.cs

```

using System;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using MyScienceLibrary.Models;
namespace MyScienceLibrary.Controllers
{
    public class AuthorController : Controller
    {
        MySciLibContext db = new MySciLibContext();
        // GET: Author
        public ActionResult Index()
        {
            var authors = db.Author;
            List<AuthorInfo> authorList = new List<AuthorInfo>();
            foreach (Author a in authors)
            {
                authorList.Add(new AuthorInfo(a));
            }
            return Json(authorList, JsonRequestBehavior.AllowGet);
        }

        // POST: Author/Create
        [HttpPost]
        public ActionResult Create(string name, string info, string birth)
        {
            Author newAuthor = new Author();
            newAuthor.Name = name;
            newAuthor.Info = info;
            if (birth != null)
            {
                newAuthor.Birth_Year = Convert.ToInt32(birth);
            }
            else
            {

```

```

        newAuthor.Birth_Year = null;
    }
    db.Author.Add(newAuthor);
    db.SaveChanges();
    return new HttpStatusCodeResult(HttpStatusCode.Created);
}

}
}

```

Листинг В.8 – Файл BookController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text.Json;
using System.Web;
using System.Web.Mvc;
using MyScienceLibrary.Models;
using System.IO;

namespace MyScienceLibrary.Controllers
{
    public class BookController : Controller
    {
        MySciLibContext db = new MySciLibContext();
        // GET: Book/Info/5
        public ActionResult Info(int id)
        {
            BookPage book = new BookPage(db, id);
            return Json(book, JsonRequestBehavior.AllowGet);
        }

        // POST: Book/Create
        [HttpPost]
        public ActionResult Create(string name, int genreID, int publisherID,
            int pubYear, string tagsJSON, string authorsJSON, string preview,
            HttpPostedFileBase file)
        {
            using (var transaction = db.Database.BeginTransaction())
            {
                try
                {
                    List<int> tags = JsonSerializer.Deserialize<List<int>>
                        (tagsJSON);
                    List<int> authors = JsonSerializer.Deserialize<List<int>>
                        (authorsJSON);
                    Book book = new Book()
                    {
                        Name = name,
                        ID_Genre = genreID,
                        ID_Publisher = publisherID,
                        Publishment_Year = pubYear,
                        File_Link = "tmp",
                    };
                    db.Book.Add(book);
                    db.SaveChanges();
                    int bookID = book.ID;
                    foreach (int tagID in tags)
                    {
                        Tag tag = db.Tag.Find(tagID);
                        book.Tag.Add(tag);
                    }
                }
                int i = 1;
            }
        }
    }
}

```

```

        foreach (int authorID in authors)
        {
            Book_Author ba = new Book_Author
            {
                ID_Author = authorID,
                ID_Book = bookID,
                Priority_Number = i
            };
            db.Book_Author.Add(ba);
            i++;
        }
        string path = string.Format("books/{0}.pdf", bookID);
        book.File_Link = path;
        if (preview != null)
        {
            Preview bookPreview = new Preview()
            {
                ID_Book = bookID,
                Prev_Text = preview
            };
            db.Preview.Add(bookPreview);
        }
        db.Preview.Add(bookPreview);
        db.SaveChanges();
        transaction.Commit();
        file.SaveAs(path);
        return new HttpStatusCodeResult(HttpStatusCode.Created);
    }
    catch
    {
        transaction.Rollback();
        return new HttpStatusCodeResult
            (HttpStatusCode.BadRequest);
    }
}

// DELETE: Book/Delete/5
[HttpDelete]
public ActionResult Delete(int id)
{
    Book book = db.Book.FirstOrDefault(b => b.ID == id);
    File.Delete(book.File_Link);
    db.Book.Remove(book);
    db.SaveChanges();
    return new EmptyResult();
}

// POST: Book/Download
[HttpPost]
public ActionResult Download(int bookID, int userID)
{
    if (db.Book.Find(bookID) != null && db.User_Acc.Find(userID) !=
        null)
    {
        Download download = db.Download.FirstOrDefault(dl =>
dl.ID_Book == bookID && dl.ID_User == userID);
        if (download == null)
        {
            db.Create_Download(userID, bookID);
            return new HttpStatusCodeResult(HttpStatusCode.Created);
        }
        return new EmptyResult();
    }
    return HttpNotFound();
}

```

```

// POST: Book/Search
[HttpPost]
public ActionResult Search(string name, string genre, string
    publisher, string published, string tagsJSON, string authorsJSON)
{
    try
    {
        int genreID = -1;
        int publisherID = -1;
        int pubYear = -1;
        List<int> tags = new List<int>();
        List<int> authors = new List<int>();
        if (genre != null)
        {
            genreID = Convert.ToInt32(genre);
        }
        if (publisher != null)
        {
            publisherID = Convert.ToInt32(publisher);
        }
        if (published != null)
        {
            pubYear = Convert.ToInt32(published);
        }
        if (tagsJSON != null)
        {
            tags = JsonSerializer.Deserialize<List<int>>(tagsJSON);
        }
        if (authorsJSON != null)
        {
            authors = JsonSerializer.Deserialize<List<int>>
                (authorsJSON);
        }
        var books = db.Book.Where(b =>
            (name == null || b.Name.StartsWith(name)) &&
            (genreID < 0 || b.ID_Genre == genreID) &&
            (publisherID < 0 || b.ID_Publisher == publisherID) &&
            (pubYear < 0 || b.Publishment_Year == pubYear) &&
            (tagsJSON == null || tags.All(t => b.Tag.Any(bt => bt.ID
                == t))) &&
            (authorsJSON == null || authors.All(a =>
                b.Book_Author.Any(ba => ba.ID_Author == a)))
        );
        List<BookPanel> results = new List<BookPanel>();
        foreach (Book b in books)
        {
            BookPanel item = new BookPanel(db, b.ID);
            results.Add(item);
        }
        return Json(results, JsonRequestBehavior.AllowGet);
    }
    catch
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
}
}

```

Листинг В.9 – Файл CommentController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Net;
using System.Web;
using System.Web.Mvc;
using MyScienceLibrary.Models;

namespace MyScienceLibrary.Controllers
{
    public class CommentController : Controller
    {
        MySciLibContext db = new MySciLibContext();
        // POST: Comment/Create
        [HttpPost]
        public ActionResult Create(int userID, int bookID, int rate, string
            comment)
        {
            if (db.Book.Find(bookID) != null && db.User_Acc.Find(userID) !=
                null)
            {
                if (db.Review.FirstOrDefault(r => r.ID_Book == bookID &&
r.ID_User == userID) == null)
                {
                    db.Create_Review(userID, bookID, rate, comment);
                    return new HttpStatusCodeResult(HttpStatusCode.Created);
                }
                else
                {
                    return new HttpStatusCodeResult
                        (HttpStatusCode.Forbidden);
                }
            }
            return HttpNotFound();
        }

        // GET: Comment/Delete/5
        [HttpDelete]
        public ActionResult Delete(int id)
        {
            Review review = db.Review.FirstOrDefault(r=>r.ID == id);
            if (review != null)
            {
                db.Review.Remove(review);
                db.SaveChanges();
                return new EmptyResult();
            }
            return HttpNotFound();
        }
    }
}

```

Листинг В.10 – Файл PublisherController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using MyScienceLibrary.Models;

namespace MyScienceLibrary.Controllers
{
    public class PublisherController : Controller
    {
        MySciLibContext db = new MySciLibContext();

```

```

// GET: Publisher
public ActionResult Index()
{
    var publishers = db.Publisher;
    List<PublisherInfo> publisherList = new List<PublisherInfo>();
    foreach (Publisher p in publishers)
    {
        publisherList.Add(new PublisherInfo(p));
    }
    return Json(publisherList, JsonRequestBehavior.AllowGet);
}

// POST: Publisher/Create
[HttpPost]
public ActionResult Create(string name, string city)
{
    Publisher newPublisher = new Publisher();
    newPublisher.Name = name;
    newPublisher.City = city;
    db.Publisher.Add(newPublisher);
    db.SaveChanges();
    return new HttpStatusCodeResult(HttpStatusCode.Created);
}
}
}

```

Листинг В.11 – Файл TagController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using MyScienceLibrary.Models;

namespace MyScienceLibrary.Controllers
{
    public class TagController : Controller
    {
        MySciLibContext db = new MySciLibContext();
        // GET: Tag
        public ActionResult Index()
        {
            var tags = db.Tag;
            List<TagInfo> tagList = new List<TagInfo>();
            foreach (Tag t in tags)
            {
                tagList.Add(new TagInfo(t));
            }
            return Json(tagList, JsonRequestBehavior.AllowGet);
        }

        // POST: Tag/Create
        [HttpPost]
        public ActionResult Create(string name)
        {
            Tag existingTag = db.Tag.FirstOrDefault(t => t.Name == name);
            if (existingTag == null)
            {
                Tag newTag = new Tag();
                newTag.Name = name;
                db.Tag.Add(newTag);
                db.SaveChanges();
            }
        }
    }
}

```

```

    }
    return new HttpStatusCodeResult(HttpStatusCode.Created);
}
}
}

```

Листинг В.12 – Файл UserController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using MyScienceLibrary.Models;
using System.Text.Json;

namespace MyScienceLibrary.Controllers
{
    public class UserController : Controller
    {
        MySciLibContext db = new MySciLibContext();

        // GET: User/Info/1
        public ActionResult Info(int id)
        {
            User_Info user = db.User_Info.FirstOrDefault(u=>u.ID == id );
            if (user != null)
            {
                return Json(user, JsonRequestBehavior.AllowGet);
            }
            return HttpNotFound();
        }

        // GET: User/Login/
        public ActionResult Login(string login, string password)
        {
            User_Acc acc = db.User_Acc. FirstOrDefault(u => u.Acc_Login ==
                login && u.Acc_Password == password);
            if (acc != null)
            {
                User_Info user = db.User_Info.FirstOrDefault(u => u.ID ==
                    acc.ID);
                if (user.Banned_Until == null)
                {
                    return Json(user, JsonRequestBehavior.AllowGet);
                }
                else
                {
                    return new HttpStatusCodeResult
                        (HttpStatusCode.Forbidden);
                }
            }
            return new HttpUnauthorizedResult();
        }

        // GET: User/Comments/5
        public ActionResult Comments(int id)
        {
            User_Acc user = db.User_Acc.Find(id);
            if (user != null)
            {
                var comments = db.Comment_Info.Where(c => c.ID_User == id);
                return Json(comments, JsonRequestBehavior.AllowGet);
            }
            return HttpNotFound();
        }
    }
}

```

```

    }

    // POST: User/Create
    [HttpPost]
    public ActionResult Create(string login, string password)
    {
        User_Acc acc = db.User_Acc.FirstOrDefault(u => u.Acc_Login ==
            login);
        if (acc == null)
        {
            acc = new User_Acc()
            {
                Acc_Login = login,
                Acc_Password = password,
                ID_Role = 2
            };
            db.User_Acc.Add(acc);
            db.SaveChanges();
            User_Info user = db.User_Info. FirstOrDefault(u => u.Acc_Login
                == login);
            Response.StatusCode = (int) HttpStatusCode.Created;
            return Json(acc);
        }
        return new HttpUnauthorizedResult();
    }
    [HttpPost]
    // Put: User/ChangePassword/5
    public ActionResult ChangePassword(int id, string oldpassword, string
        newpassword)
    {
        User_Acc user = db.User_Acc.Find(id);
        if (user.Acc_Password == oldpassword)
        {
            user.Acc_Password = newpassword;
            db.SaveChanges();
            return new EmptyResult();
        }
        return new HttpUnauthorizedResult();
    }
    // GET: User/Downloads/5
    public ActionResult Downloads(int id)
    {
        User_Acc user = db.User_Acc.Find(id);
        if (user != null)
        {
            var downloads = db.Download.Where(dl => dl.ID_User == id);
            List<BookPanel> books = new List<BookPanel>();
            foreach (Download dl in downloads)
            {
                BookPanel book = new BookPanel(db, dl.ID_Book);
                books.Add(book);
            }
            return Json(books, JsonRequestBehavior.AllowGet);
        }
        return HttpNotFound();
    }
    // POST: User/Create
    [HttpPost]
    public ActionResult Ban(int adminID, int userID, int length)
    {
        db.Create_Ban(userID, adminID, length);
        return new EmptyResult();
    }
}
}

```

Приложение Г
ИСХОДНЫЙ КОД МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Г.1 – Файл BanActivity.java.xml

```

package com.myscilib;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
import android.view.View;
import android.os.Bundle;
import android.widget.EditText;

public class BanActivity extends AppCompatActivity {

    private int id;
    private int userID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ban);
    }

    public void ButtonClick(View v){
        try{
            EditText inputBan = findViewById(R.id.editTextLength);
            String lengthString = inputBan.getText().toString();
            if (lengthString.length() > 0)
            {
                int length = Integer.parseInt(lengthString);
                if (RequestCreator.banUser(id, userID, length)){
                    AlertDialog.Builder builder = new
                        AlertDialog.Builder(this);
                    builder.setTitle("Успех");
                    builder.setMessage("Пользователь успешно
                        заблокирован");
                    builder.setPositiveButton("OK", new
                        DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface
                                dialog, int which) {
                                    finish();
                                }
                        });
                    AlertDialog alert = builder.create();
                    alert.show();
                }
            }
            else {
                AlertDialog.Builder builder = new
                    AlertDialog.Builder(this);
                builder.setTitle("Ошибка");
                builder.setMessage("Не удалось заблокировать
                    пользователя");
                builder.setPositiveButton("OK", new
                    DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface
                            dialog, int which) {
                                dialog.cancel();
                            }
                    });
                AlertDialog alert = builder.create();
                alert.show();
            }
        }
    }
}

```

```

        else {
            AlertDialog.Builder builder = new
                AlertDialog.Builder(this);
            builder.setTitle("Ошибка");
            builder.setMessage("Недопустимый формат длины
                блокировки");
            builder.setPositiveButton("OK", new
                DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
                        which) {
                        dialog.cancel();
                    }
                });
            AlertDialog alert = builder.create();
            alert.show();
        }
    }
    catch (Exception e) {
        Toast.makeText(getApplicationContext(), "Ошибка подключения к
            сети", Toast.LENGTH_SHORT).show();
    }
}

```

Листинг Г.2 – Файл BookInfoActivity.java

```

package com.myscilib;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;

public class BookInfoActivity extends AppCompatActivity {

    private int id;
    private boolean isAdmin;
    private int bookID;
    private BookPage book;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        try{
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_book_info);
            id = getIntent().getExtras().getInt("id");
            isAdmin = getIntent().getExtras().getBoolean("isAdmin");
            bookID = getIntent().getExtras().getInt("bookID");
            ImageButton delButton = findViewById(R.id.delBookButton);
            if (!isAdmin){
                delButton.setVisibility(View.INVISIBLE);
            }
            book = RequestCreator.getBook(bookID);
            TextView editName = findViewById(R.id.textViewBookName);
            editName.setText(book.getInfo().getName());
            TextView editGenre = findViewById(R.id.textViewGenre);

```

```

editGenre.setText(book.getInfo().getGenre());
TextView editAuthors = findViewById(R.id.textBookAuthorList);
String authors = "";
Author[] authorList = book.getAuthors();
for (int i = 0; i < authorList.length; i++)
{
    authors += authorList[i].getName() + " ";
}
editAuthors.setText(authors);
TextView editPublication = findViewById
(R.id.textViewBookPublishment);
editPublication.setText(book.getInfo().getPublisher() + ", " +
    book.getInfo().getPublishmentYear());
TextView editDownloads =
findViewById(R.id.textViewBookDownloads);
editDownloads.setText("Скачиваний: " + book.getInfo().
    getDownloadCount());
TextView editComments = findViewById(R.id.textViewBookComments);
editComments.setText("Комментариев: " + book.getInfo().
    getCommentCount());
TextView editRate = findViewById(R.id.textViewAvgRate);
Double avgRate = book.getInfo().getAvgRate();
if (avgRate != null) {
    editRate.setText("Средняя оценка: " +
        book.getInfo().getAvgRate());
}
else {
    editRate.setText("Средняя оценка: нет");
}
TextView editTags = findViewById(R.id.textViewTags);
Tag[] tagList = book.getTags();
String tags = "Теги: ";
for(int i = 0; i < tagList.length; i++){
    tags+= tagList[i].getName() + " ";
}
editTags.setText(tags);
TextView editPreview = findViewById(R.id.textViewPreview);
String preview = book.getPreview();
if (preview != null) {
    editPreview.setText(preview);
}
else{
    editPreview.setText("Аннотация недоступна");
}
Comment[] comments = book.getComments();
if (comments != null)
{
    if (!RequestCreator.canMakeComment(id, bookID)){
        Button commentButton = findViewById
            (R.id.buttonReviewAdd);
        commentButton.setVisibility(View.INVISIBLE);
    }
    FragmentManager manager = getSupportFragmentManager();
    for (int i=0; i < comments.length; i++){
        FragmentTransaction transaction = manager.
            beginTransaction();
        ReviewFragment fragment = new ReviewFragment();
        Bundle fragmentArgs = new Bundle();
        fragmentArgs.putInt("id", id);
        fragmentArgs.putBoolean("isAdmin", isAdmin);
        fragmentArgs.putBoolean("pathToBook", false);
        fragmentArgs.putSerializable("comment", comments[i]);
        fragment.setArguments(fragmentArgs);
        transaction.add(R.id.commentsContainer, fragment);
        transaction.commit();
    }
}

```

```

    }
}
catch (Exception e) {
    Toast.makeText(getApplicationContext(), "Ошибка подключения к
        сети", Toast.LENGTH_SHORT).show();
    finish();
}
}

public void delButtonClick(View v){
    try{
        if (RequestCreator.deleteBook(bookID)){
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("Успех");
            builder.setMessage("Книга удалена");
            builder.setPositiveButton("ОК", new
                DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which){
                        finish();
                    }
                });
            AlertDialog alert = builder.create();
            alert.show();
        }
        else {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("Ошибка");
            builder.setMessage("Не удалось удалить книгу. Повторите
                попытку позже");
            builder.setPositiveButton("ОК", new
                DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which){
                        dialog.cancel();
                    }
                });
            AlertDialog alert = builder.create();
            alert.show();
        }
    }
    catch (Exception e) {
        Toast.makeText(getApplicationContext(), "Ошибка подключения к
            сети", Toast.LENGTH_SHORT).show();
    }
}

public void downloadClick (View v) {
    try{
        if (RequestCreator.downloadBook(id, bookID,
            book.info.getFileLink())){
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setTitle("Успех");
            builder.setMessage("Книга успешно загружена");
            builder.setPositiveButton("ОК", new
                DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which){
                        dialog.cancel();
                    }
                });
            AlertDialog alert = builder.create();
            alert.show();
        }
        else {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);

```



```

        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
    AlertDialog alert = builder.create();
    alert.show();
}
else {
    if (userAcc.getBannedUntil() != null) {
        AlertDialog.Builder builder = new
            AlertDialog.Builder(this);
        builder.setTitle("Ошибка");
        builder.setMessage("К сожалению, ваш аккаунт
            заблокирован");
        builder.setPositiveButton("OK", new
            DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog,
                    int which) {
                    dialog.cancel();
                }
            });
        AlertDialog alert = builder.create();
        alert.show();
    }
    else {
        Intent intent = new Intent(this,
            MainPageActivity.class);
        intent.putExtra("id", userAcc.getId());
        intent.putExtra("login", userAcc.getLogin());
        intent.putExtra("isAdmin",
            userAcc.getRole().equals("Администратор"));
        startActivity(intent);
    }
}
}
catch (Exception e) {
    Toast.makeText(getApplicationContext(), "Ошибка подключения к
        сети", Toast.LENGTH_SHORT).show();
}

}

public void registerClick(View v) {
    Intent intent = new Intent(this, RegisterActivity.class);
    startActivity(intent);
}
}

```

Листинг Г.4 – MainPage.Activity.java

```

package com.myscilib;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;

import com.google.android.material.bottomnavigation.BottomNavigationView;
import androidx.appcompat.app.AlertDialog;

```

```

import androidx.appcompat.app.AppCompatActivity;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import java.util.Arrays;
import java.util.Comparator;

public class MainPageActivity extends AppCompatActivity {

    private int id;
    private boolean isAdmin;
    private int[] searchTagIDs;
    private int[] searchAuthorIDs;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main_page);
        BottomNavigationView navView = findViewById(R.id.nav_view);
        AppBarConfiguration appBarConfiguration = new
            AppBarConfiguration.Builder(
                R.id.navigation_search, R.id.navigation_downloads,
                R.id.navigation_comments)
            .build();
        NavController navController = Navigation.findNavController(this,
            R.id.nav_host_fragment);
        NavigationUI.setupActionBarWithNavController(this, navController,
            appBarConfiguration);
        NavigationUI.setupWithNavController(navView, navController);
        TextView loginText = findViewById(R.id.textViewUsername);
        Bundle args = getIntent().getExtras();
        String login = args.getString("login");
        loginText.setText(login);
        id = args.getInt("id");
        isAdmin = args.getBoolean("isAdmin");
    }

    public void logoutClick(View v) {
        finish();
    }

    public void changePwdClick(View v) {
        Intent intent = new Intent(this, NewPasswordActivity.class);
        intent.putExtra("id", id);
        startActivity(intent);
    }

    public void addClick (View v) {
        Intent intent = new Intent(this, NewBookActivity.class);
        startActivity(intent);
    }

    public void searchClick(View v){
        try {
            EditText nameInput = findViewById(R.id.editTextSearchName);
            String name = nameInput.getText().toString();
            Spinner genreChoice = findViewById(R.id.spinnerGenre);
            Spinner publisherChoice = findViewById(R.id.spinnerPublisher);
            EditText yearInput = findViewById(R.id.editTextSearchYear);
            String yearString = yearInput.getText().toString();
            Integer pubYear = null;
            if (yearString.length() > 0){
                pubYear = Integer.parseInt(yearString);
            }
            Integer genreID = genreChoice.getSelectedItemPosition();
            if (genreID == 0) {
                genreID = null;
            }
        }
    }
}

```

```

Integer publisherID = publisherChoice.getSelectedItemId();
if (publisherID == 0)
{
    publisherID = null;
}
BookPanel[] books = RequestCreator.searchBooks(name,
    searchAuthorIDs, genreID, publisherID, searchTagIDs,
    pubYear);
Spinner orderChoice = findViewById(R.id.spinnerSortBy);
int order = orderChoice.getSelectedItemId();
switch (order){
    case 1:
        Arrays.sort(books, new Comparator<BookPanel>(){
            public int compare(BookPanel book1, BookPanel book2){
                return book1.getInfo().getName().compareTo
                    (book2.getInfo().getName());
            }
        });
        break;
    case 2:
        Arrays.sort(books, new Comparator<BookPanel>(){
            public int compare(BookPanel book1, BookPanel book2){
                Integer count2 = book2.getInfo().
                    getDownloadCount();
                return count2.compareTo(book1.getInfo().
                    getDownloadCount());
            }
        });
        break;
    case 3:
        Arrays.sort(books, new Comparator<BookPanel>(){
            public int compare(BookPanel book1, BookPanel book2){
                Double rate2 = book2.getInfo().getAvgRate();
                return rate2.compareTo(book1.getInfo().
                    getAvgRate());
            }
        });
        break;
    case 4:
        Arrays.sort(books, new Comparator<BookPanel>(){
            public int compare(BookPanel book1, BookPanel book2){
                Integer count2 = book2.getInfo().
                    getCommentCount();
                return count2.compareTo(book1.getInfo().
                    getCommentCount());
            }
        });
        break;
    default:
        break;
}
Intent intent = new Intent(this, SearchResultsActivity.class);
intent.putExtra("id", id);
intent.putExtra("isAdmin", isAdmin);
intent.putExtra("results", books);
startActivity(intent);
}
catch (Exception e) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Ошибка");
    builder.setMessage("В процессе поиска произошла ошибка");
    builder.setPositiveButton("OK", new
        DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        }
    );
}

```

```

        }
    });
    AlertDialog alert = builder.create();
    alert.show();
}

}

public void editAuthorsClick(View v) {
    Intent intent = new Intent(this, GetAuthorsActivity.class);
    startActivityForResult(intent, 1);
}

public void editTagsClick(View v){
    Intent intent = new Intent(this, GetTagsActivity.class);
    startActivityForResult(intent, 2);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data){
    if (requestCode == 1){
        TextView authorList = findViewById
            (R.id.textViewSearchAuthorList);
        String authorNames = data.getStringExtra("authornames");
        authorList.setText(authorNames);
        searchAuthorIDs = data.getIntArrayExtra("authorlist");
    }
    else if (requestCode == 2) {
        TextView tagList = findViewById(R.id.textViewSearchTagList);
        String tagNames = data.getStringExtra("tagnames");
        tagList.setText(tagNames);
        searchTagIDs = data.getIntArrayExtra("taglist");
    }
    else {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
}

```

Листинг Г.5 – Файл NewBookActivity.java

```

package com.myscilib;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.TextureView;
import android.view.View;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import java.io.File;

public class NewBookActivity extends AppCompatActivity {
    private int[] authorIDs;
    private int[] tagIDs;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_book);
        authorIDs = null;
        tagIDs = null;
    }
}

```

```

    }

    private boolean allReady(String name, int pubYear, int genreID, int pubID) {
        return name.length() > 0 && pubYear >= 1980 && pubYear <= 2021 &&
            genreID > 0 && pubID > 0 &&
            authorIDs != null && tagIDs != null;
    }

    public void uploadClick(View v) {
        try {
            EditText nameInput = findViewById(R.id.editTextBookName);
            String name = nameInput.getText().toString();
            Spinner genreChoice = findViewById(R.id.spinnerBookGenre);
            Spinner publisherChoice = findViewById(
                (R.id.spinnerBookPublisher));
            EditText yearInput = findViewById(R.id.editTextBookYear);
            int pubYear = Integer.parseInt(yearInput.getText().toString());
            EditText previewInput = findViewById(R.id.editTextPreview);
            String preview = (previewInput.getText().toString());
            int genreID = genreChoice.getSelectedItemPosition();
            int publisherID = publisherChoice.getSelectedItemPosition();
            EditText fileInput = findViewById(R.id.editPath);
            String path = fileInput.getText().toString();
            if (allReady(name, pubYear, genreID, publisherID)) {
                File file = new File(path);
                if (file.exists()) {
                    if (RequestCreator.uploadBook(name, authorIDs, genreID,
                        publisherID, tagIDs, pubYear, preview, path)) {
                        AlertDialog.Builder builder = new AlertDialog.
                            Builder(this);
                        builder.setTitle("Успех");
                        builder.setMessage("Книга добавлена");
                        builder.setPositiveButton("OK", new
                            DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int
                                    which) {
                                    dialog.cancel();
                                }
                            });
                        AlertDialog alert = builder.create();
                        alert.show();
                        finish();
                    }
                } else {
                    AlertDialog.Builder builder = new
                        AlertDialog.Builder(this);
                    builder.setTitle("Ошибка");
                    builder.setMessage("Не удалось добавить книгу.
                        Повторите попытку позже");
                    builder.setPositiveButton("OK", new
                        DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int
                                which) {
                                dialog.cancel();
                            }
                        });
                    AlertDialog alert = builder.create();
                    alert.show();
                }
            }
        } else {
            AlertDialog.Builder builder = new
                AlertDialog.Builder(this);
            builder.setTitle("Ошибка");

```

```

        builder.setMessage("Файл книги не указан или указан неверно");
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
}
else {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Ошибка");
    builder.setMessage("Должны быть введены все требуемые данные. Кроме того, " + "год издания должен находиться в промежутке от 1980 до 2021");
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
    AlertDialog alert = builder.create();
    alert.show();
}
}
catch (NumberFormatException e) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Ошибка");
    builder.setMessage("Недопустимое значение года издания");
    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
    AlertDialog alert = builder.create();
    alert.show();
}

catch (Exception e) {
    Toast.makeText(getApplicationContext(), "Ошибка подключения к сети", Toast.LENGTH_SHORT).show();
}
}

public void editAuthorsClick(View v) {
    Intent intent = new Intent(this, GetAuthorsActivity.class);
    startActivityForResult(intent, 1);
}

public void editTagsClick(View v){
    Intent intent = new Intent(this, GetTagsActivity.class);
    startActivityForResult(intent, 2);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data){
    if (requestCode == 1){

```

```

        TextView authorList = findViewById(R.id.textViewBookAuthorList);
        String authorNames = data.getStringExtra("authornames");
        authorList.setText(authorNames);
        authorIDs = data.getIntArrayExtra("authorlist");
    }
    else if (requestCode == 2) {
        TextView tagList = findViewById(R.id.textViewBookTagList);
        String tagNames = data.getStringExtra("tagnames");
        tagList.setText(tagNames);
        tagIDs = data.getIntArrayExtra("taglist");
    }
    else {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
}

```

Листинг Г.6 – Файл NewPasswordActivity.java

```

package com.myscilib;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class NewPasswordActivity extends AppCompatActivity {

    private int id;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_password);
        Bundle args = getIntent().getExtras();
        id = args.getInt("id");
    }

    public void saveClick(View v) {
        try{
            TextView oldPwdEdit = findViewById(R.id.editTextOldPwd);
            String oldPwd = oldPwdEdit.getText().toString();
            TextView newPwdEdit = findViewById(R.id.editTextNewPwd);
            String newPwd = newPwdEdit.getText().toString();
            TextView repeatPwdEdit = findViewById(R.id.editTextRepeatPwd);
            String repeatPwd = repeatPwdEdit.getText().toString();
            if (newPwd.equals(repeatPwd))
            {
                if (RegisterActivity.isValidPassword(newPwd))
                {
                    if (RequestCreator.changePassword(id, oldPwd,
                        newPwd)){
                        AlertDialog.Builder builder = new
                            AlertDialog.Builder(this);
                        builder.setTitle("Успех");
                        builder.setMessage("Пароль успешно изменен");
                        builder.setPositiveButton("OK", new
                            DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface
                                    dialog, int which) {
                                    finish();
                                }
                            }
                        );
                    }
                }
            }
        }
    }
}

```

```

        }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }
    else {
        AlertDialog.Builder builder = new
        AlertDialog.Builder(this);
        builder.setTitle("Ошибка");
        builder.setMessage("Исходный пароль неверен");
        builder.setPositiveButton("OK", new
            DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface
                    dialog, int which) {
                    dialog.cancel();
                }
            });
        AlertDialog alert = builder.create();
        alert.show();
    }
}
else {
    AlertDialog.Builder builder = new
        AlertDialog.Builder(this);
    builder.setTitle("Ошибка");
    builder.setMessage("Недопустимый новый пароль. Каждый
        пароль должен включать в себя цифры и буквы,
        длина - от 8 до 20 символов");
    builder.setPositiveButton("OK", new
        DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int
                which) {
                dialog.cancel();
            }
        });
    AlertDialog alert = builder.create();
    alert.show();
}
}
else {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Ошибка");
    builder.setMessage("Пароли не совпадают");
    builder.setPositiveButton("OK", new
        DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which){
                dialog.cancel();
            }
        });
    AlertDialog alert = builder.create();
    alert.show();
}
}
catch (Exception e) {
    Toast.makeText(getApplicationContext(), "Ошибка подключения к
        сети", Toast.LENGTH_SHORT).show();
}
}
}

```

Листинг Г.7 – Файл NewReviewActivity.java

```

package com.myscilib;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Spinner;
public class NewReviewActivity extends AppCompatActivity {

    private int id;
    private int bookID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_review);
        id = getIntent().getExtras().getInt("id");
        bookID = getIntent().getExtras().getInt("bookID");
    }

    public void sendClick(View v) {
        try{
            Spinner rateSpinner = findViewById(R.id.spinnerRate);
            int rate = rateSpinner.getSelectedItemPosition() + 1;
            EditText commentInput = findViewById(R.id.editTextReview);
            String comment = commentInput.getText().toString();
            if (RequestCreator.uploadReview(id, bookID, rate, comment)){
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                builder.setTitle("Успех");
                builder.setMessage("Отзыв успешно добавлен");
                builder.setPositiveButton("OK", new
                    DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog,int which){
                            finish();
                        }
                    });
                AlertDialog alert = builder.create();
                alert.show();
            }
            else {
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                builder.setTitle("Ошибка");
                builder.setMessage("Не удалось одобавить отзыв. Попробуйте
                    еще раз");
                builder.setPositiveButton("OK", new
                    DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog,int which){
                            dialog.cancel();
                        }
                    });
                AlertDialog alert = builder.create();
                alert.show();
            }
        }
        catch (Exception e) {
            Toast.makeText(getApplicationContext(), "Ошибка подключения к
                сети", Toast.LENGTH_SHORT).show();
        }
    }
}

```

Листинг Г.8 – Файл RegisterActivity.java

```

package com.myscilib;

import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

public class RegisterActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
    }

    private static boolean isLatinLetter(char c)
    {
        return c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z';
    }

    public static boolean isValidLogin(String login)
    {
        if (login.length() > 20 || login.length() < 3)
        {
            return false;
        }
        for (int i=0; i<login.length(); i++)
        {
            char symbol = login.charAt(i);
            if (!isLatinLetter(symbol) && !Character.isDigit(symbol) &&
                symbol != '_' && symbol != '-') {
                return false;
            }
        }
        return true;
    }

    public static boolean isValidPassword (String password)
    {
        if (password.length() > 20 || password.length() < 8)
        {
            return false;
        }
        boolean hasDigits = false;
        boolean hasLetters = false;
        for (int i=0; i < password.length(); i++)
        {
            char symbol = password.charAt(i);
            if (isLatinLetter(symbol)) {
                hasLetters = true;
            }
            else if (Character.isDigit(symbol)) {
                hasDigits = true;
            }
            else {
                return false;
            }
        }
        return (hasDigits && hasLetters);
    }
}

```

```

public void loginClick(View v) {
    Intent intent = new Intent(this, MainActivity.class);
    startActivity(intent);
}

public void registerClick(View v) {
    try{
        EditText loginInput = findViewById(R.id.editRegisterLogin);
        EditText passwordInput = findViewById(R.id.editRegisterPassword);
        String newLogin = loginInput.getText().toString();
        String newPwd = passwordInput.getText().toString();
        if (isValidLogin(newLogin) && isValidPassword(newPwd)){
            User userAcc = RequestCreator.createNewUser(newLogin,
                newPwd);
            if (userAcc == null) {
                AlertDialog.Builder builder = new
                    AlertDialog.Builder(this);
                builder.setTitle("Ошибка");
                builder.setMessage("Данный логин уже занят");
                builder.setPositiveButton("OK", new
                    DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int
                            which) {
                            dialog.cancel();
                        }
                    });
                AlertDialog alert = builder.create();
                alert.show();
            }
        }
        else {
            Intent intent = new Intent(this,
                MainPageActivity.class);
            intent.putExtra("id", userAcc.getId());
            intent.putExtra("login", userAcc.getLogin());
            intent.putExtra("isAdmin",
                userAcc.getRole().equals("Администратор"));
            startActivity(intent);
        }
    }
    else {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Ошибка");
        builder.setMessage("Недопустимый логин или пароль. Логин
            может включать только латинские буквы, цифры и знак _,
            длина - от 3 до 20 символов. Пароль должен включать в
            себя цифры и буквы, длина - от 8 до 20 символов");
        builder.setPositiveButton("OK", new
            DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog,int which){
                    dialog.cancel();
                }
            });
        AlertDialog alert = builder.create();
        alert.show();
    }
}
}
}

```

Листинг Г.9 – UserInfoActivity.java

```

package com.myscilib;
import androidx.appcompat.app.AppCompatActivity;

```

```

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class UserInfoActivity extends AppCompatActivity {

    private int id;
    private boolean isAdmin;
    private int userID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        try{
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_user_info);
            id = getIntent().getExtras().getInt("id");
            isAdmin = getIntent().getExtras().getBoolean("isAdmin");
            userID = getIntent().getExtras().getInt("userID");
            User userInfo = RequestCreator.getUserByID(userID);
            boolean isBannable = isAdmin && (userInfo.getBannedUntil() ==
                null) && (userInfo.getRole() != "Администратор");
            if (!isBannable)
            {
                Button banButton = findViewById(R.id.buttonBan);
                banButton.setVisibility(View.GONE);
            }
            TextView editLogin = findViewById(R.id.textViewLogin);
            editLogin.setText(userInfo.getLogin());
            TextView editRole = findViewById(R.id.textViewRole);
            editRole.setText(userInfo.getRole());
            TextView editDownloads = findViewById(R.id.textViewDownloads);
            editDownloads.setText("Книг скачано: " +
                userInfo.getDownloadCount());
            TextView editComments = findViewById(R.id.textViewComments);
            editComments.setText("Комментариев оставлено: " +
                userInfo.getCommentCount());
            TextView editRates = findViewById(R.id.textViewRates);
            editRates.setText("Оценок поставлено: " +
                userInfo.getReviewCount());
            TextView editBan = findViewById(R.id.textViewBan);
            if (userInfo.getBannedUntil() != null) {
                String pattern = "dd.MM.yyyy";
                DateFormat df = new SimpleDateFormat(pattern);
                String banString = df.format(userInfo.getBannedUntil());
                editBan.setText("Заблокирован до " + banString);
            }
            else {
                editBan.setText("Активных блокировок нет");
            }
        }
        catch (Exception e) {
            Toast.makeText(getApplicationContext(), "Ошибка подключения к
                сети", Toast.LENGTH_SHORT).show();
        }
    }

    public void buttonClick(View v) {
        Intent intent = new Intent(this, BanActivity.class);
        intent.putExtra("id", id);
        intent.putExtra("userID", userID);
        startActivity(intent);
    }
}

```

```

        recreate();
    }
}

```

Листинг Г.10 – SearchResultsActivity.java

```

package com.myscilib;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import android.os.Bundle;

public class SearchResultsActivity extends AppCompatActivity {

    private int id;
    private boolean isAdmin;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_results);
        Bundle args = getIntent().getExtras();
        id = args.getInt("id");
        isAdmin = args.getBoolean("isAdmin");
        if (savedInstanceState == null) {
            BookPanel[] results = (BookPanel[]) args.get("results");
            FragmentManager manager = getSupportFragmentManager();
            for (int i = 0; i < results.length; i++) {
                FragmentTransaction transaction = manager.beginTransaction();
                BookFragment fragment = new BookFragment();
                Bundle fragmentArgs = new Bundle();
                fragmentArgs.putInt("id", id);
                fragmentArgs.putBoolean("isAdmin", isAdmin);
                fragmentArgs.putSerializable("book", results[i]);
                fragment.setArguments(fragmentArgs);
                transaction.add(R.id.results_container, fragment);
                transaction.commit();
            }
        }
    }
}

```

Листинг Г.11 – RequestCreator.java

```

package com.myscilib;

import com.google.gson.Gson;
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.*;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Date;

```

```

public class RequestCreator {
    public static User getUserByLoginPassword(String login, String password)
        throws MalformedURLException, IOException
    {
        User userAcc = null;
        String requestURI =
String.format("http://192.168.0.104/MyScienceLibrary/
user/login/?login=%s&password=%s", login, password);
        URL myURL = new URL(requestURI);
        HttpURLConnection connection = (HttpURLConnection)
            myURL.openConnection();
        connection.setRequestMethod("GET");
        int responseCode = connection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            StringBuilder content = new StringBuilder();
            BufferedReader reader = new BufferedReader(new
                InputStreamReader(connection.getInputStream()));
            String input;
            while((input = reader.readLine()) != null) {
                content.append(input);
            }
            String json = content.toString();
            Gson gson = new Gson();
            userAcc = gson.fromJson(json, User.class);
            userAcc.deserializeBanUntil();
        }
        else if(responseCode == HttpURLConnection.HTTP_FORBIDDEN) {
            userAcc = new User();
            userAcc.setBannedUntil(new Date());
        }
        return userAcc;
    }
    public static User createNewUser(String login, String password)
        throws MalformedURLException, IOException
    {
        String requestURI =
"http://192.168.0.104/MyScienceLibrary/User/Create";
        URL myURL = new URL(requestURI);
        HttpURLConnection connection = (HttpURLConnection)myURL.
            openConnection();
        connection.setRequestMethod("POST");
        connection.setDoOutput(true);
        String params = String.format("login=%s&password=%s", login,
            password);
        byte[] postData = params.getBytes(StandardCharsets.UTF_8);
        connection.setRequestProperty("Content-
            Length", String.valueOf(postData.length));
        connection.connect();
        try(OutputStream out = connection.getOutputStream()) {
            out.write(postData);
        }
        int responseCode = connection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_CREATED) {
            StringBuilder content = new StringBuilder();
            BufferedReader reader = new BufferedReader(new
                InputStreamReader(connection.getInputStream()));
            String input;
            while((input = reader.readLine()) != null) {
                content.append(input);
            }
            String json = content.toString();
            Gson gson = new Gson();
            User userAcc = gson.fromJson(json, User.class);
            userAcc.deserializeBanUntil();
            return userAcc;
        }
    }
}

```

```

    }
    else {
        return null;
    }
}
public static BookPanel[] getMyBooks (int id)
    throws MalformedURLException, IOException
{
    String requestURI = String.format("http://192.168.0.104/
        MyScienceLibrary/ user/downloads/%d", id);
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("GET");
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK){
        StringBuilder content = new StringBuilder();
        BufferedReader reader = new BufferedReader(new
            InputStreamReader(connection.getInputStream()));
        String input;
        while((input = reader.readLine()) != null){
            content.append(input);
        }
        String json = content.toString();
        Gson gson = new Gson();
        BookPanel[] books = gson.fromJson(json, BookPanel[].class);
        return books;
    }
    else {
        return null;
    }
}

public static BookPanel[] searchBooks (String name, int[] authorIDs,
    Integer genreID, Integer publisherID, int[] tagIDs, Integer pubYear)
    throws MalformedURLException, IOException
{
    String requestURI = "http://192.168.0.104/MyScienceLibrary/
        Book/Search";
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("POST");
    connection.setDoOutput(true);
    ArrayList<String> params = new ArrayList<String>();
    if (!name.isEmpty()){
        params.add(String.format("name=%s", name));
    }
    if (authorIDs != null){
        Gson gson = new Gson();
        String json = gson.toJson(authorIDs);
        params.add(String.format("authorsJSON=%s", json));
    }
    if (genreID != null){
        params.add(String.format("genre=%d", genreID));
    }
    if (publisherID != null){
        params.add(String.format("publisher=%d", publisherID));
    }
    if (tagIDs != null){
        Gson gson = new Gson();
        String json = gson.toJson(tagIDs);
        params.add(String.format("tagsJSON=%s", json));
    }
    if (pubYear != null){
        params.add(String.format("published=%s", pubYear));
    }
}

```

```

StringBuilder paramsMaker = new StringBuilder();
if (!params.isEmpty()) {
    paramsMaker.append(params.get(0));
    for (int i = 1; i < params.size(); i++) {
        paramsMaker.append("&");
        paramsMaker.append(params.get(i));
    }
}
byte[] postData = paramsMaker.toString().getBytes(
    StandardCharsets.UTF_8);
connection.setRequestProperty("Content-Length",
    String.valueOf(postData.length));
connection.connect();
try (OutputStream out = connection.getOutputStream()) {
    out.write(postData);
}
int responseCode = connection.getResponseCode();
if (responseCode == HttpURLConnection.HTTP_OK) {
    StringBuilder content = new StringBuilder();
    BufferedReader reader = new BufferedReader(new
        InputStreamReader(connection.getInputStream()));
    String input;
    while ((input = reader.readLine()) != null) {
        content.append(input);
    }
    String json = content.toString();
    Gson gson = new Gson();
    BookPanel[] books = gson.fromJson(json, BookPanel[].class);
    return books;
}
else {
    return null;
}
}

public static Comment[] getMyComments (int id)
    throws MalformedURLException, IOException
{
    String requestURI = String.format("http://192.168.0.104/
        MyScienceLibrary/user/comments/%d", id);
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("GET");
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK) {
        StringBuilder content = new StringBuilder();
        BufferedReader reader = new BufferedReader(new InputStreamReader
            (connection.getInputStream()));
        String input;
        while ((input = reader.readLine()) != null) {
            content.append(input);
        }
        String json = content.toString();
        Gson gson = new Gson();
        Comment[] comments = gson.fromJson(json, Comment[].class);
        for (int i=0; i<comments.length; i++) {
            comments[i].deserializeCreationTime();
        }
        return comments;
    }
    else {
        return null;
    }
}
}

```

```

public static boolean uploadBook (String name, int[] authorIDs, int
    genreID, int publisherID, int[] tagIDs, int pubYear, String preview,
    String path)
    throws MalformedURLException, IOException
{
    String boundary = "----**hhghgfgfhgghgh7788678677687*";
    String requestURI = "http://192.168.0.104/MyScienceLibrary/
        Book/Create";
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("POST");
    connection.setDoOutput(true);
    connection.setDoInput(true);
    connection.setUseCaches(false);
    connection.setRequestProperty("Connection", "Keep-Alive");
    connection.setRequestProperty("Content-Type", "multipart/form-
        data;boundary=" + boundary);
    DataOutputStream out = new DataOutputStream
        (connection.getOutputStream());
    out.writeBytes(String.format("--%s\r\n", boundary));
    out.writeBytes("Content-Disposition: form-
        data;name=\"name\"\\r\\n\\r\\n");
    out.write(String.format("%s\r\n", name) .
        getBytes(StandardCharsets.UTF_8));
    out.writeBytes(String.format("--%s\r\n", boundary));
    Gson gson = new Gson();
    String json = gson.toJson(authorIDs);
    out.writeBytes("Content-Disposition: form-
        data;name=\"authorsJSON\"\\r\\n\\r\\n");
    out.writeBytes(String.format("%s\r\n", json));
    out.writeBytes(String.format("--%s\r\n", boundary));
    out.writeBytes("Content-Disposition: form-
        data;name=\"genreID\"\\r\\n\\r\\n");
    out.writeBytes(String.format("%d\r\n", genreID));
    out.writeBytes(String.format("--%s\r\n", boundary));
    out.writeBytes("Content-Disposition: form-
        data;name=\"publisherID\"\\r\\n\\r\\n");
    out.writeBytes(String.format("%d\r\n", publisherID));
    out.writeBytes(String.format("--%s\r\n", boundary));
    String json2 = gson.toJson(tagIDs);
    out.writeBytes("Content-Disposition: form-
        data;name=\"tagsJSON\"\\r\\n\\r\\n");
    out.writeBytes(String.format("%s\r\n", json2));
    out.writeBytes(String.format("--%s\r\n", boundary));
    out.writeBytes("Content-Disposition: form-
        data;name=\"pubYear\"\\r\\n\\r\\n");
    out.writeBytes(String.format("%d\r\n", pubYear));
    out.writeBytes(String.format("--%s\r\n", boundary));
    if (!preview.isEmpty()){
        out.writeBytes("Content-Disposition: form-
            data;name=\"preview\"\\r\\n\\r\\n");
        out.write(String.format("%s\r\n",
            preview).getBytes(StandardCharsets.UTF_8));
        out.writeBytes(String.format("--%s\r\n", boundary));
    }
    out.writeBytes(String.format("--%s\r\n", boundary));
    out.writeBytes("Content-Disposition: form-
        data;name=\"file\";filename=\"%s\"\\r\\n",path));
    out.writeBytes("Content-Type: application/pdf\r\n");
    out.writeBytes("\\r\\n");
    FileInputStream fileInput = new FileInputStream(new File(path));
    int maxSize = 1024 * 1024;
    int bytesAvailable = fileInput.available();
    int bufferSize = Math.min(bytesAvailable, maxSize);
    byte[] buffer = new byte[bufferSize];

```

```

int count = fileInput.read(buffer, 0, bufferSize);
while (count > 0){
    out.write(buffer, 0, bufferSize);
    bytesAvailable = fileInput.available();
    bufferSize = Math.min(bytesAvailable, maxSize);
    count = fileInput.read(buffer, 0, bufferSize);
}
out.writeBytes("\r\n");
out.writeBytes(String.format("--%s--\r\n", boundary));

int responseCode = connection.getResponseCode();
System.out.println(responseCode);

    BufferedReader reader = new BufferedReader(new InputStreamReader
        (connection.getErrorStream()));
    String input;
    while((input = reader.readLine()) != null){
        System.out.println(input);
    }
    return responseCode == HttpURLConnection.HTTP_OK;
}
public static boolean downloadBook(int userID, int bookID, String link)
    throws MalformedURLException, IOException
{
    String requestURI = "http://192.168.0.104/MyScienceLibrary/
        Book/Download";
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("POST");
    connection.setDoOutput(true);
    String params = String.format("userID=%d&bookID=%d", userID, bookID);
    byte[] postData = params.getBytes(StandardCharsets.UTF_8);
    connection.setRequestProperty("Content-Length",
        String.valueOf(postData.length));
    connection.connect();
    try(OutputStream out = connection.getOutputStream()){
        out.write(postData);
    }
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_CREATED || responseCode ==
        HttpURLConnection.HTTP_CREATED){
        String fileURI = String.format("http://192.168.0.104/
            MyScienceLibrary/%s", link);
        String destDir = String.format
            ("/storage/emulated/0/Download/myscilib_%d.pdf", bookID);
        File file = new File(destDir);
        int i = 0;
        while (file.exists()){
            i++;
            destDir = String.format ("/storage/emulated/0/Download/
                myscilib_%d(%d).pdf", bookID, i);
            file = new File(destDir);
        }
        try(BufferedInputStream in = new BufferedInputStream(new
            URL(fileURI).openStream());
            FileOutputStream out = new FileOutputStream(destDir)){
            byte data[] = new byte[1024];
            int count;
            while ((count = in.read(data, 0, 1024)) != -1){
                out.write(data, 0, count);
            }
        }
        return true;
    }
    else{

```

```

        return false;
    }
}

public static boolean deleteBook(int id)
    throws MalformedURLException, IOException
{
    String requestURI = String.format("http://192.168.0.104/
        MyScienceLibrary/Book/Delete/%d", id);
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("DELETE");
    int responseCode = connection.getResponseCode();
    return responseCode == HttpURLConnection.HTTP_OK;
}

public static boolean uploadReview (int userID, int bookID, int rate,
    String comment)
    throws MalformedURLException, IOException
{
    String requestURI =
"http://192.168.0.104/MyScienceLibrary/Comment/Create";
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("POST");
    connection.setDoOutput(true);
    StringBuilder paramsMaker = new StringBuilder();
    paramsMaker.append(String.format("userID=%d&bookID=%d&rate=%d",
        userID, bookID, rate));
    if (!comment.isEmpty()){
        paramsMaker.append(String.format("&comment=%s", comment));
    }
    byte[] postData = paramsMaker.toString().getBytes
        (StandardCharsets.UTF_8);
    connection.setRequestProperty("Content-Length",String.valueOf
        (postData.length));
    connection.connect();
    try(OutputStream out = connection.getOutputStream()){
        out.write(postData);
    }
    int responseCode = connection.getResponseCode();
    return responseCode == HttpURLConnection.HTTP_CREATED;
}

public static BookPage getBook (int id)
    throws MalformedURLException, IOException
{
    String requestURI = String.format("http://192.168.0.104/
        MyScienceLibrary/book/info/%d", id);
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("GET");
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK) {
        StringBuilder content = new StringBuilder();
        BufferedReader reader = new BufferedReader(new
            InputStreamReader(connection.getInputStream()));
        String input;
        while((input = reader.readLine()) != null){
            content.append(input);
        }
        String json = content.toString();
        Gson gson = new Gson();
    }
}

```

```

        BookPage book = gson.fromJson(json, BookPage.class);
        Comment[] comments = book.getComments();
        for (int i = 0; i < comments.length; i++) {
            comments[i].deserializeCreationTime();
        }
        return book;
    }
    else {
        return null;
    }
}

public static boolean deleteReview (int id)
    throws MalformedURLException, IOException
{
    String requestURI = String.format("http://192.168.0.104/
        MyScienceLibrary/Comment/Delete/%d", id);
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("DELETE");
    int responseCode = connection.getResponseCode();
    return responseCode == HttpURLConnection.HTTP_OK;
}

public static boolean changePassword (int id, String oldPwd,
    String newPwd)
    throws MalformedURLException, IOException
{
    String requestURI = "http://192.168.0.104/MyScienceLibrary/User/
        ChangePassword/";
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("POST");
    connection.setDoOutput(true);
    String params = String.format("id=%d&oldpassword=%s&newpassword=%s",
        id, oldPwd, newPwd);
    byte[] postData = params.getBytes(StandardCharsets.UTF_8);
    connection.setRequestProperty("Content-Length", String.valueOf
        (postData.length));
    connection.connect();
    try (OutputStream out = connection.getOutputStream()) {
        out.write(postData);
    }
    int responseCode = connection.getResponseCode();
    return responseCode == HttpURLConnection.HTTP_OK;
}

public static User getUserByID(int id)
    throws MalformedURLException, IOException
{
    String requestURI = String.format("http://192.168.0.104/
        MyScienceLibrary/user/info/%d", id);
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("GET");
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK) {
        StringBuilder content = new StringBuilder();
        BufferedReader reader = new BufferedReader(new InputStreamReader
            (connection.getInputStream()));
        String input;
        while ((input = reader.readLine()) != null) {
            content.append(input);
        }
    }
}

```

```

    }
    String json = content.toString();
    Gson gson = new Gson();
    User userAcc = gson.fromJson(json, User.class);
    userAcc.deserializeBanUntil();
    return userAcc;
}
else{
    return null;
}
}

public static boolean banUser(int adminID, int userID, int length)
    throws MalformedURLException, IOException
{
    String requestURI = "http://192.168.0.104/MyScienceLibrary/User/Ban";
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("POST");
    connection.setDoOutput(true);
    String params = String.format("adminID=%d&userID=%d&length=%d",
        adminID, userID, length);
    byte[] postData = params.getBytes(StandardCharsets.UTF_8);
    connection.setRequestProperty("Content-Length",String.valueOf
        (postData.length));
    connection.connect();
    try(OutputStream out = connection.getOutputStream()){
        out.write(postData);
    }
    int responseCode = connection.getResponseCode();
    return responseCode == HttpURLConnection.HTTP_OK;
}

public static Tag[] getTags()
    throws MalformedURLException, IOException
{
    String requestURI = "http://192.168.0.104/MyScienceLibrary/tag";
    URL myURL = new URL(requestURI);
    HttpURLConnection connection = (HttpURLConnection)
        myURL.openConnection();
    connection.setRequestMethod("GET");
    int responseCode = connection.getResponseCode();
    if (responseCode == HttpURLConnection.HTTP_OK) {
        StringBuilder content = new StringBuilder();
        BufferedReader reader = new BufferedReader(new InputStreamReader
            (connection.getInputStream()));
        String input;
        while((input = reader.readLine()) != null){
            content.append(input);
        }
        String json = content.toString();
        Gson gson = new Gson();
        Tag[] tags = gson.fromJson(json, Tag[].class);
        Arrays.sort(tags, new Comparator<Tag>(){
            @Override
            public int compare(Tag t1, Tag t2){
                Integer id1 = t1.getId();
                return id1.compareTo(t2.getId());
            }
        });
        return tags;
    }
    else{
        return null;
    }
}

```

```

    }

    public static Author[] getAuthors()
        throws MalformedURLException, IOException
    {
        String requestURI = "http://192.168.0.104/MyScienceLibrary/author";
        URL myURL = new URL(requestURI);
        HttpURLConnection connection = (HttpURLConnection)
            myURL.openConnection();
        connection.setRequestMethod("GET");
        int responseCode = connection.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            StringBuilder content = new StringBuilder();
            BufferedReader reader = new BufferedReader(new InputStreamReader
                (connection.getInputStream()));
            String input;
            while((input = reader.readLine()) != null){
                content.append(input);
            }
            String json = content.toString();
            Gson gson = new Gson();
            Author[] authors = gson.fromJson(json, Author[].class);
            Arrays.sort(authors, new Comparator<Author>(){
                @Override
                public int compare(Author a1, Author a2){
                    Integer id1 = a1.getId();
                    return id1.compareTo(a2.getId());
                }
            });
            return authors;
        }
        else{
            return null;
        }
    }

    public static boolean canMakeComment(int UserID, int bookID)
        throws MalformedURLException, IOException
    {
        String requestURI = "http://192.168.0.104/MyScienceLibrary/comment/
            available";
        URL myURL = new URL(requestURI);
        HttpURLConnection connection = (HttpURLConnection)
            myURL.openConnection();
        connection.setRequestMethod("GET");
        int responseCode = connection.getResponseCode();
        return responseCode == HttpURLConnection.HTTP_OK;
    }
}

```

Приложение Д
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ
(рекомендуемое)

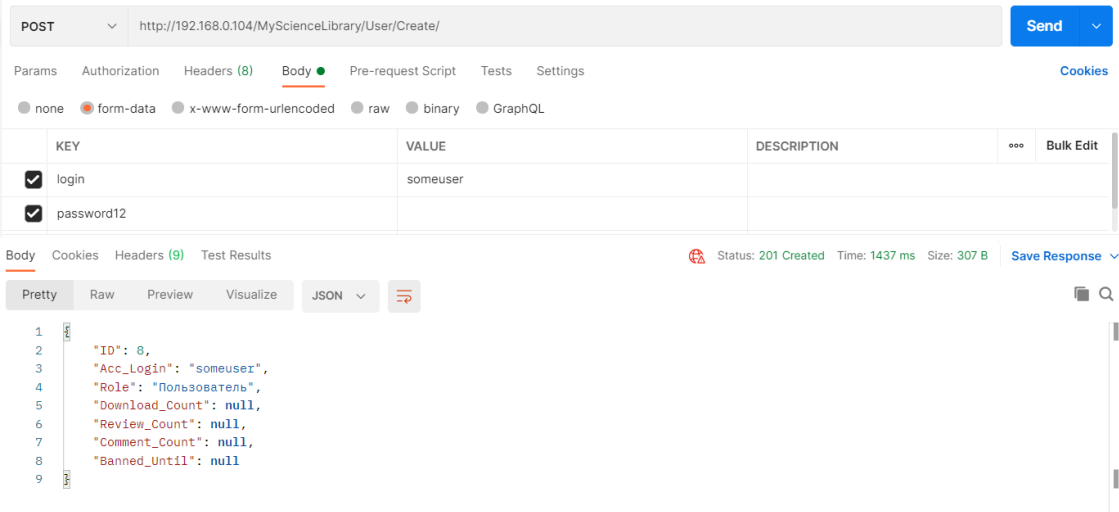


Рисунок Д.1 – Добавление нового пользователя

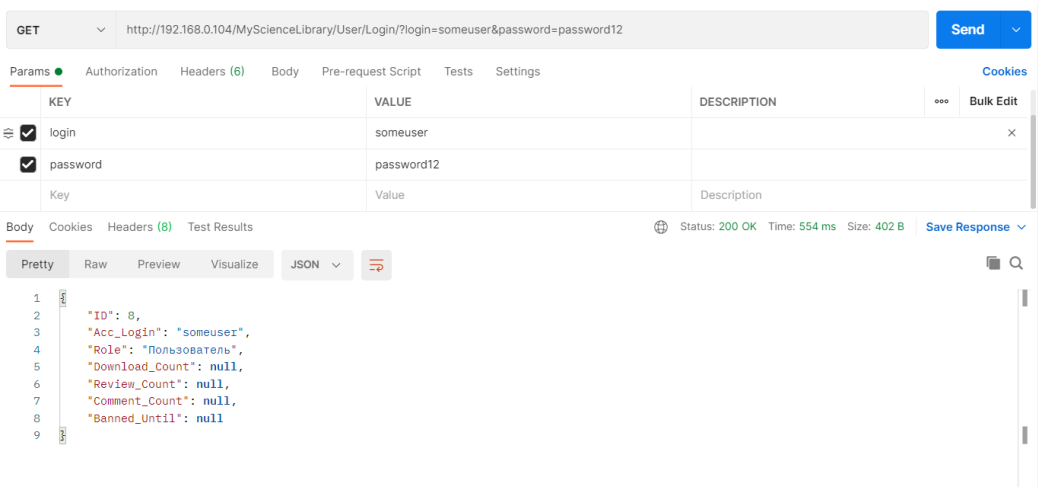


Рисунок Д.2 – Вход существующего пользователя в систему

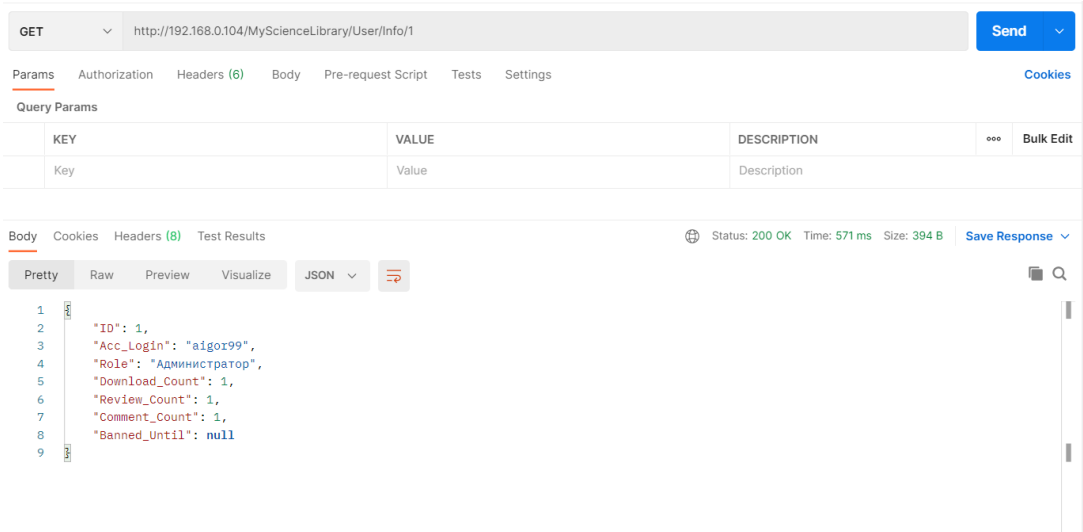


Рисунок Д.3 – Получение информации о пользователе

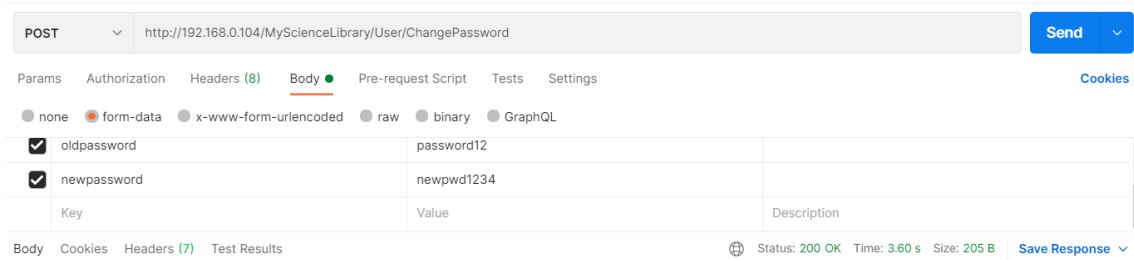


Рисунок Д.4 – Изменение пароля пользователя

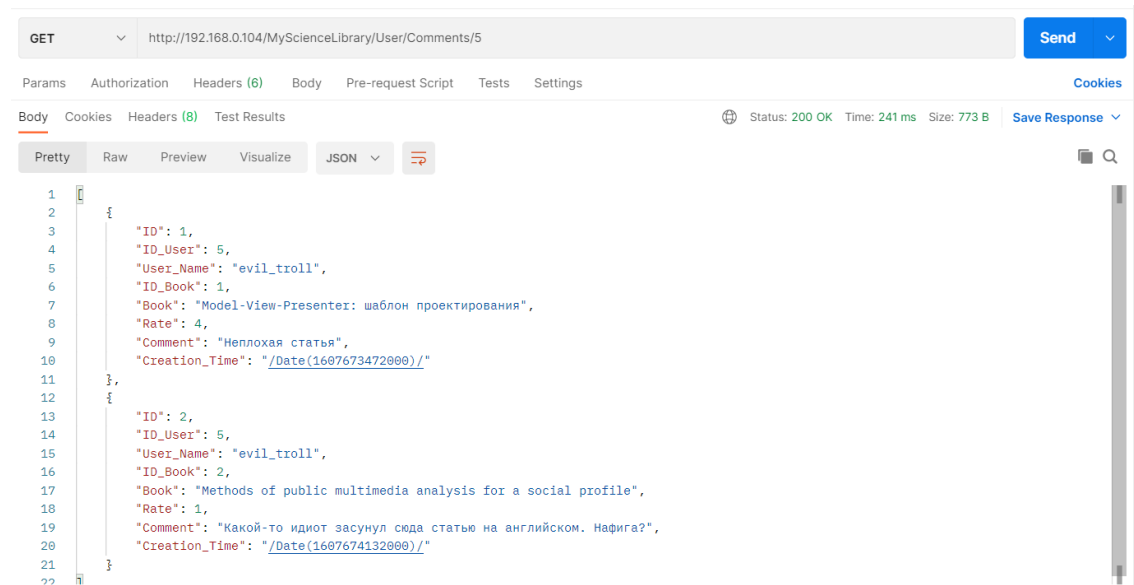


Рисунок Д.5 – Получение списка комментариев пользователя

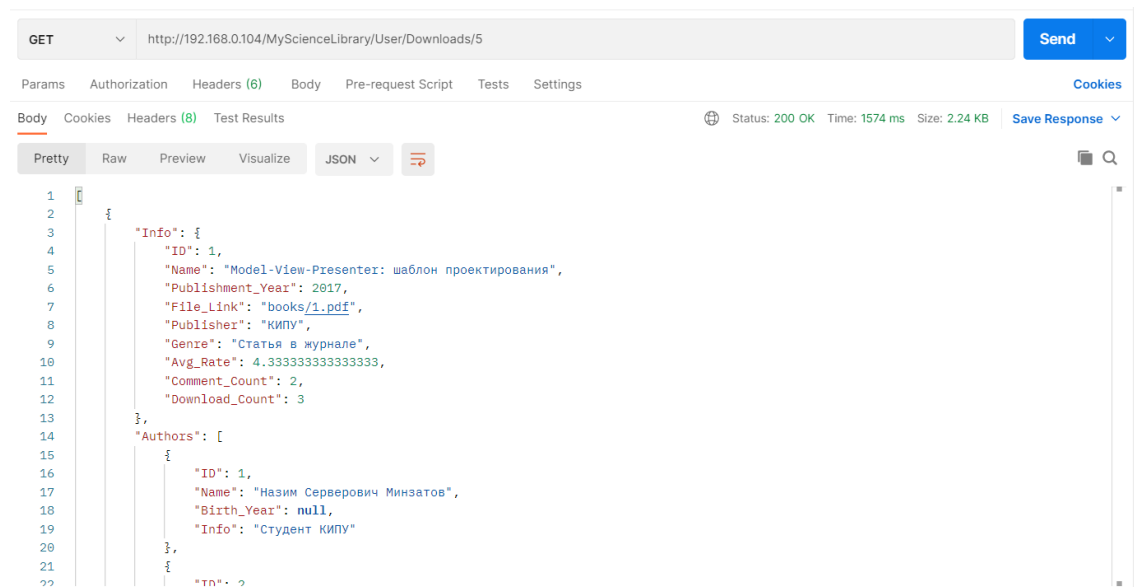


Рисунок Д.6 – Получение списка загрузок пользователя

POST

http://192.168.0.104/MyScienceLibrary/User/Ban

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

☒	adminID	1	
☒	userID	8	
☒	length	3	
	Key	Value	Description

Body

Cookies

Headers (8)

Test Results

Status: 200 OK Time: 1574 ms Size: 2.24 KB Save Response

Рисунок Д.7 – Блокировка пользователя

POST

http://192.168.0.104/MyScienceLibrary/Comment/Create

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
☒	userID	1			
☒	bookID	1			
☒	comment	Познавательно			
☒	rate	5			
	Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

Status: 201 Created Time: 402 ms Size: 306 B Save Response

Рисунок Д.8 – Добавление отзыва

DELETE

http://192.168.0.104/MyScienceLibrary/Comment/Delete/23

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

Status: 200 OK Time: 402 ms Size: 306 B Save Response

Рисунок Д.9 – Удаление отзыва

POST

http://192.168.0.104/MyScienceLibrary/Book/Download

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
☒	bookID	1			
☒	userID	1			
	Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

Status: 201 Created Time: 2.31 s Size: 307 B Save Response

Рисунок Д.10 – Сохранение информации о загрузке

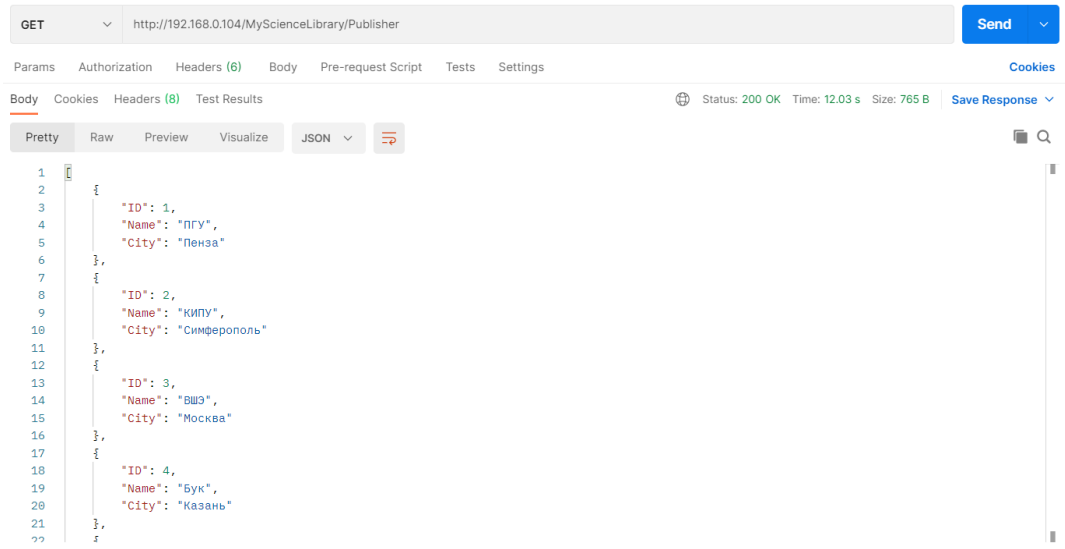


Рисунок Д.11 – Получение списка издательств

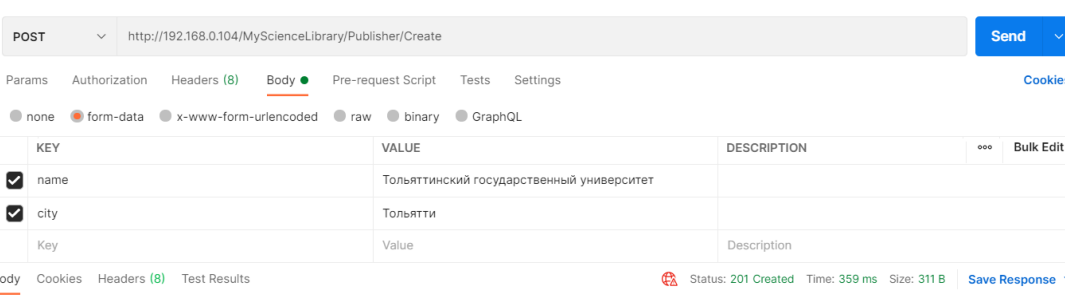


Рисунок Д.12 – Добавление нового издательства

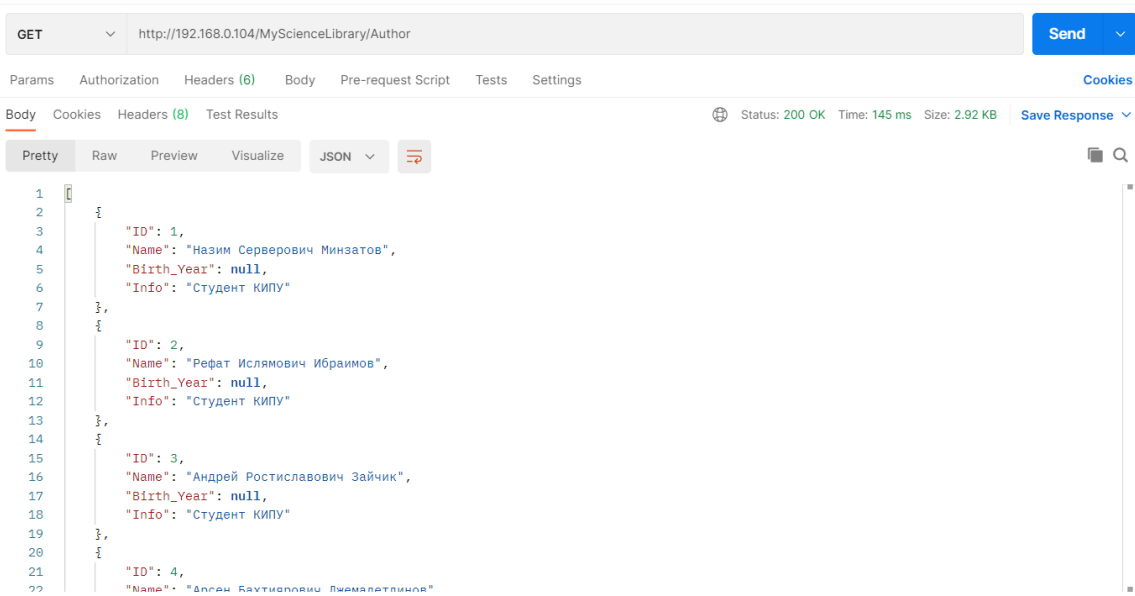


Рисунок Д.13 – Получение списка авторов

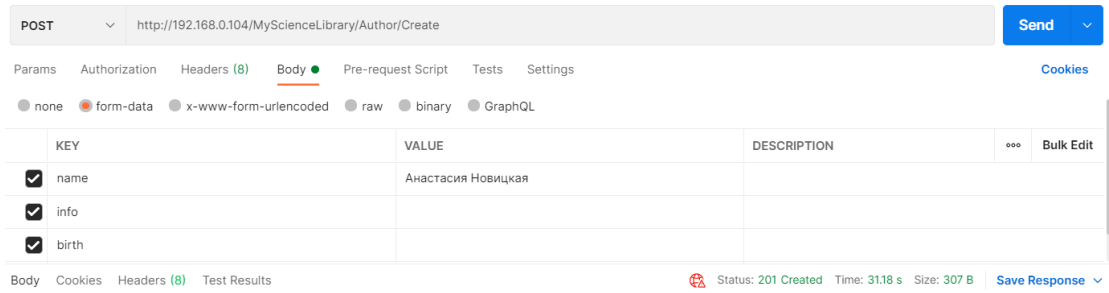


Рисунок Д.14 – Добавление нового автора

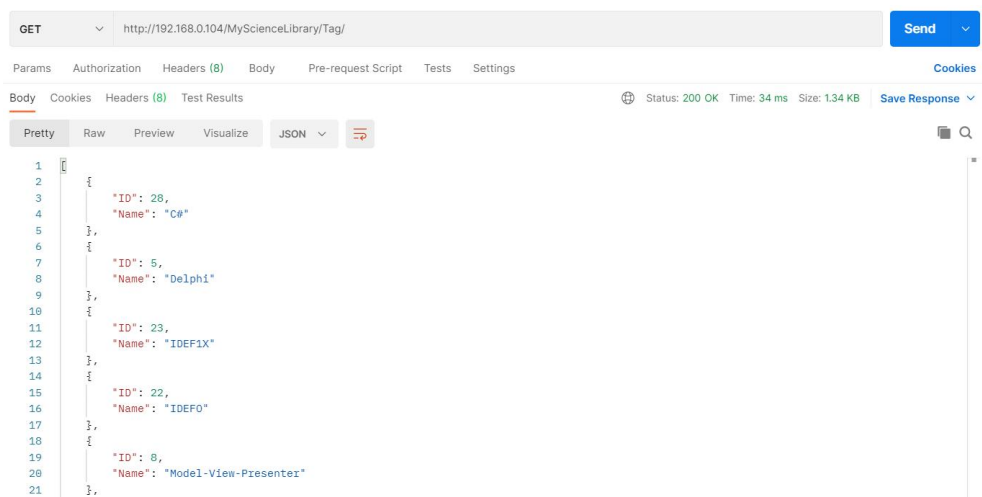


Рисунок Д.15 – Получение списка тегов

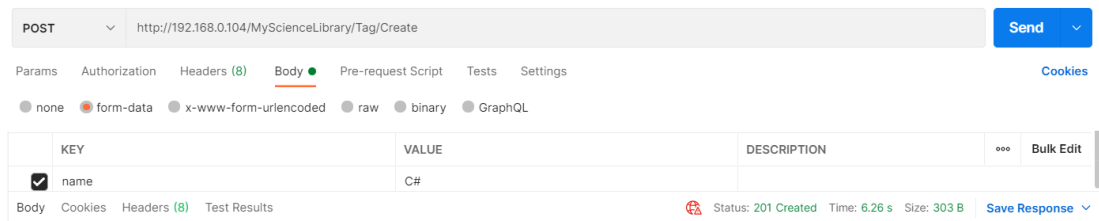


Рисунок Д.16 – Добавление нового тега

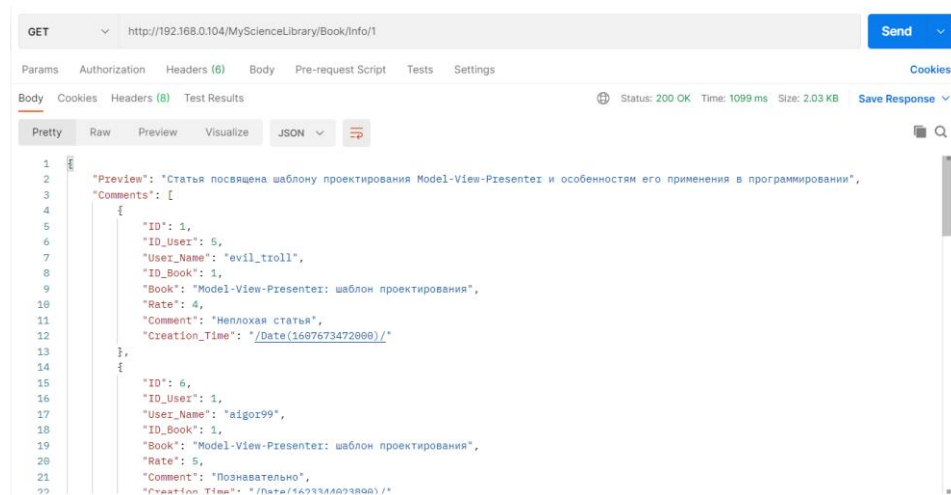


Рисунок Д.17 – Получение информации о книге

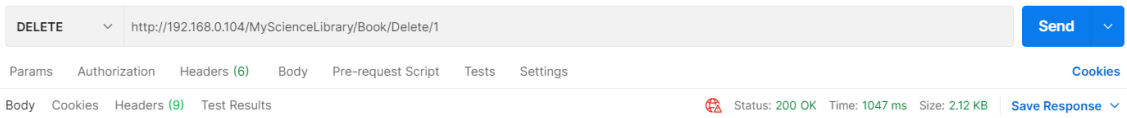


Рисунок Д.18 – Удаление книги

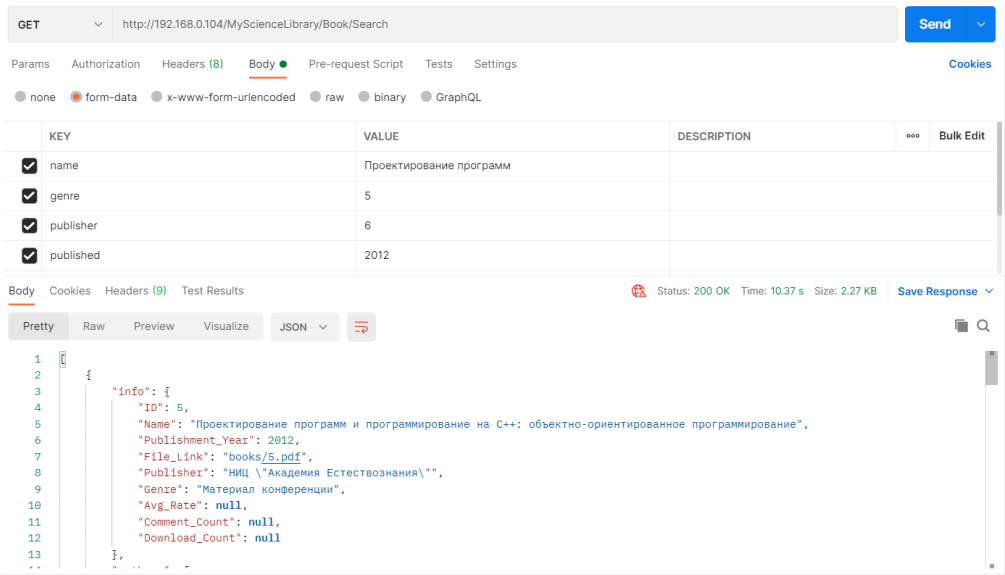


Рисунок Д.19 – Поиск книг

Приложение Е
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

The screenshot shows a search interface with a purple header labeled "Поиск". Below the header, there is a "mylogin" label and two buttons: "ВЫЙТИ ИЗ СИСТЕМЫ" and "СМЕНА ПАРОЛЯ". A search input field is labeled "Введите название книги". Below this, there are several filter sections: "Авторы" with a pencil icon and the text "Нет выбранных авторов", "Жанр" with a dropdown menu showing "Нет", "Издательство" with a dropdown menu showing "Нет", "Теги" with a pencil icon and the text "Нет выбранных тегов", "Год издания" with a text input field, and "Сортировка" with a dropdown menu showing "Нет". A purple button labeled "ИСКАТЬ" is at the bottom of the filters. At the very bottom, there is a navigation bar with three icons and labels: "Поиск" (with a magnifying glass icon), "Скачанные книги" (with a download icon), and "Мои комментарии" (with a speech bubble icon).

Рисунок Е.1 – Успешная регистрация в системе

The screenshot shows a registration screen with a purple header labeled "Регистрация". In the center, there is a white error dialog box with the title "Ошибка" and the message "Данный логин уже занят". There is a purple "ОК" button in the bottom right corner of the dialog box. Below the dialog box, there is a link "Уже зарегистрировались?" and a purple button labeled "ВОЙТИ".

Рисунок Е.2 – Логин занят

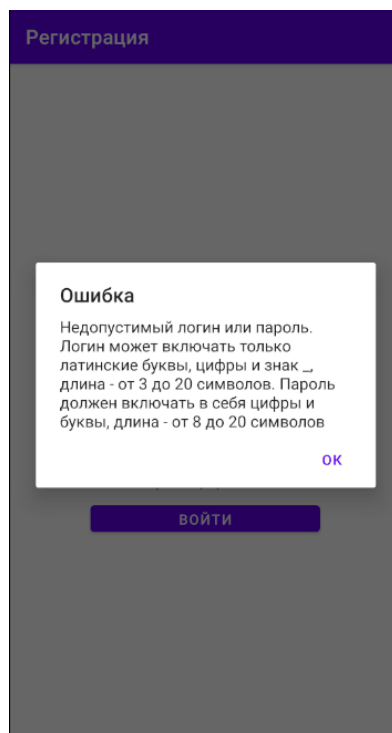


Рисунок Е.3 – Недопустимые символы в логине и пароле

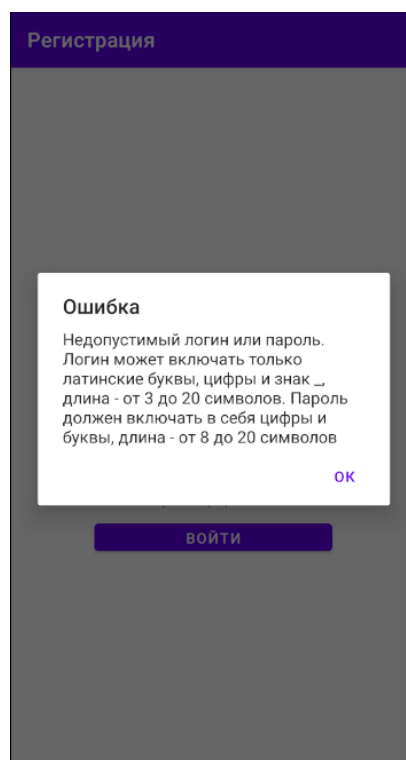


Рисунок Е.4 – Логин и пароль недопустимой длины

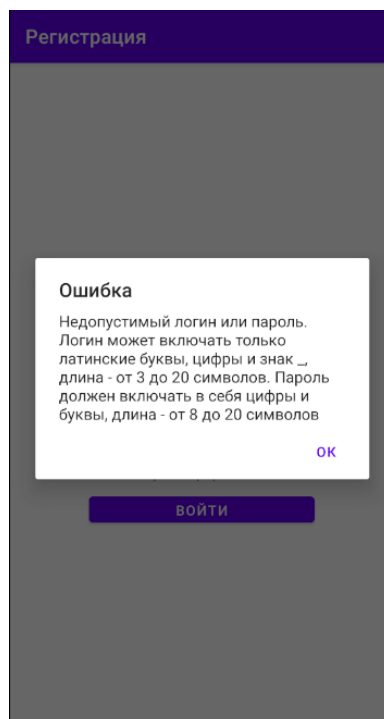


Рисунок Е.5 – Пароль без букв

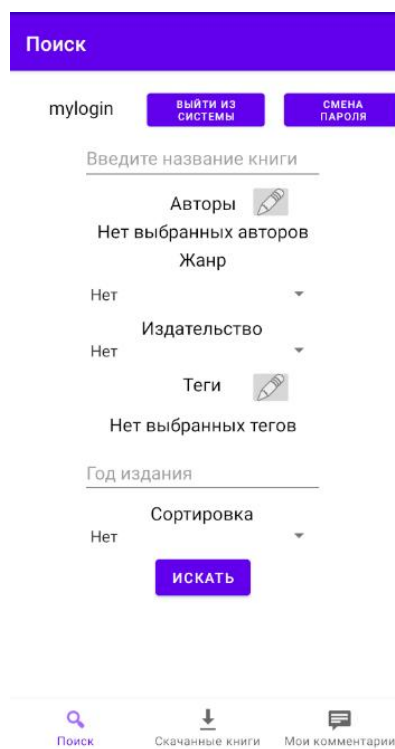


Рисунок Е.6 – Успешная авторизация в системе

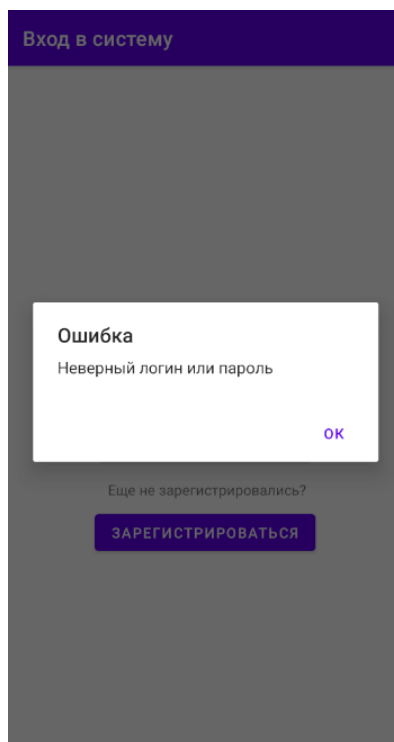


Рисунок Е.7 – Неверный логин или пароль

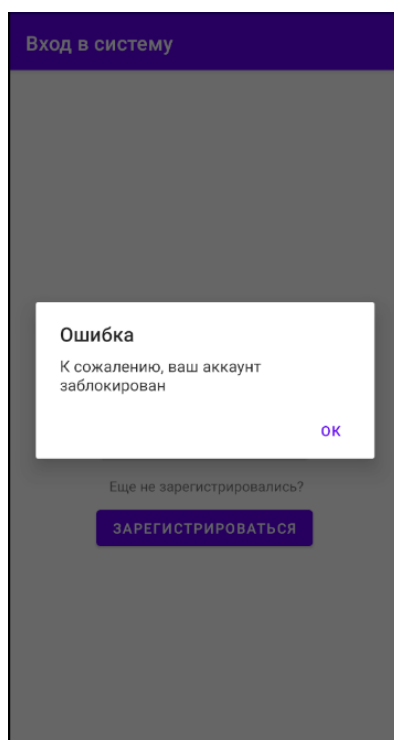


Рисунок Е.8 – Аккаунт заблокирован

Результаты поиска

Проектирование программ и программирование на C++: объектно-ориентированное программирование
 Материал конференции
 Ольга Леонидовна Викентьева
 Ольга Андреевна Полякова
 НИЦ "Академия естествознания", 2012
 Скачиваний: 0 Комментариев: 0 Средняя оценка: нет
 Теги: Программирование C++ ООП

Проектирование программ и программирование на C++: структурное программирование
 Материал конференции
 Ольга Леонидовна Викентьева
 Ольга Андреевна Полякова
 НИЦ "Академия естествознания", 2012
 Скачиваний: 0 Комментариев: 0 Средняя оценка: нет
 Теги: Программирование C++ Структурное программирование

Рисунок Е.9 – Поиск

Информация о книге

Информационные методы моделирования процессов в экономических системах 
 Материал конференции
 Анастасия Новицкая
 Тольяттинский государственный университет, 2021
 Скачиваний: 1 Комментариев: 1 Средняя оценка: 5.0
 Теги: IDEF0 IDEF1X UML
 Аннотация недоступна

ЗАГРУЗИТЬ

Комментарии

mylogin	Оценка: 5	
09.06.2021 12:07		
Интересно		

Рисунок Е.10 – Оценка с отзывом

Информация о книге

**Информационные методы
моделирования процессов в
экономических системах**
Материал конференции



Анастасия Новицкая

Тольяттинский государственный университет, 2021

Скачиваний: 2 Комментариев: 1 Средняя оценка: 4.5

Теги: IDEF0 IDEF1X UML

Аннотация недоступна

ЗАГРУЗИТЬ

Комментарии

mylogin	Оценка: 5	
09.06.2021 12:07		
Интересно		

Рисунок Е.11 – Оценка без отзыва

Информация о книге

**Информационные методы
моделирования процессов в
экономических системах**
Материал конференции

Анастасия Новицкая

Тольяттинский государственный университет, 2021

Скачиваний: 0 Комментариев: 0 Средняя оценка: нет

Теги: IDEF0 IDEF1X UML

Аннотация недоступна

ЗАГРУЗИТЬ

ДОБАВИТЬ ОТЗЫВ

Комментарии

Рисунок Е.12– Добавление книги

Добавление книги

Введите название книги

Авторы

Нет выбранных авторов

Жанр

Ошибка

Должны быть введены все требуемые данные. Кроме того, год издания должен находиться в промежутке от 1980 до 2021

OK

Аннотация (необязательно)

Путь к файлу

ДОБАВИТЬ

Рисунок Е.13 – Неполные данные

Добавление книги

в экономических системах

Авторы

Анастасия Новицкая

Жанр

Ошибка

Файл книги не указан или указан неверно

OK

2021

Аннотация (необязательно)

Путь к файлу

ДОБАВИТЬ

Рисунок Е.14 – Нет файла

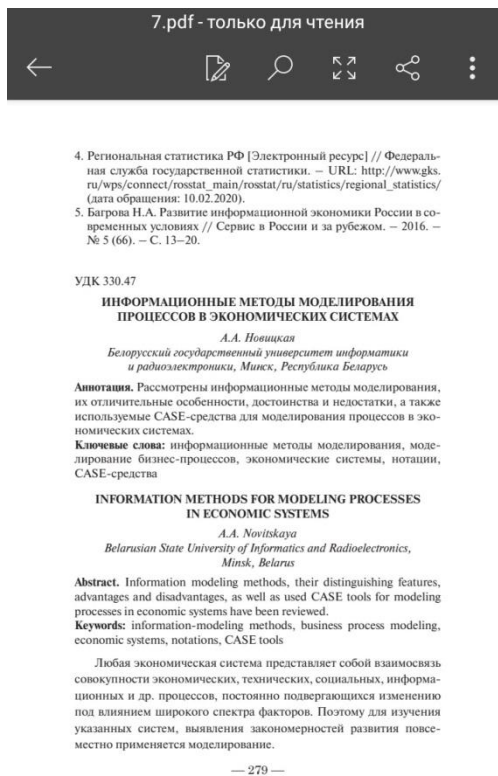


Рисунок Е.15 – Скачивание книги

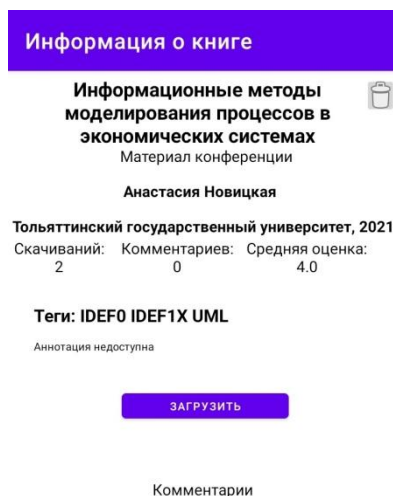


Рисунок Е.16 – Удаление отзыва

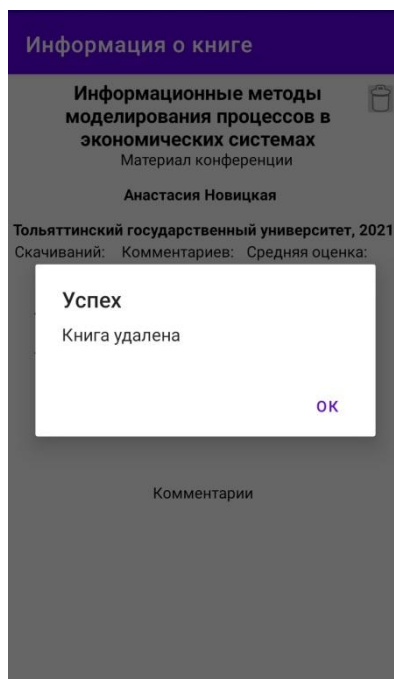


Рисунок Е.17 – Удаление книги

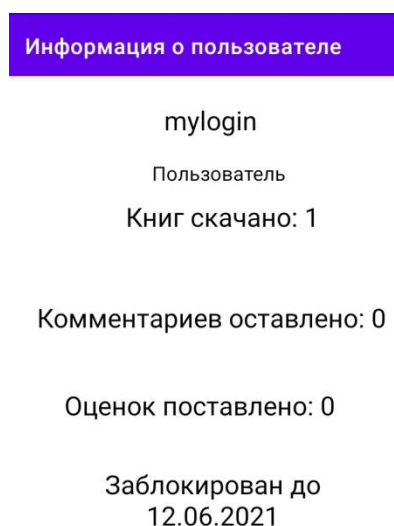


Рисунок Е.18 – Блокировка пользователя