МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет	*Кафедра*
вычислительной техники	МО и ПЭВМ

Направление подготовки	09.04.04 «Программная инженерия»
Магистерская программа	Программное обеспечение вычислительной техники и автоматизированных систем

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

Модель и алгоритм функционирования программных средств электронных библиотек

Студент _____ Ащеулов Игорь Олегович
(подпись, дата) (ФИО полностью)

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

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

Рецензент Начальник отдела ООО НТП «Криптософт» _____ Афонин А.Ю.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

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

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

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

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

Пенза, 2023

Список сокращений

ПС	Программные средства
ПО	Программное обеспечение
UML	Unified modeling language
RUP	Rational unified process
ООП	Объектно-ориентированное программирование
IT	Informational technologies
ВАК	Высшая аттестационная комиссия
MS	Microsoft
БД	База данных
ЭБ	Электронная библиотека

Содержание

Введение 6
1 Анализ предметной области исследования 14
1.1 Понятие электронной библиотеки и классификация электронных библиотек 14
1.2 Анализ программных продуктов аналогичного назначения 18
1.2.1 Научная электронная библиотека eLibrary 18
1.2.2 Национальная электронная библиотека (НЭБ) 19
1.2.3 КиберЛенинка 21
1.2.4 Общие выводы по анализу аналогов 22
1.3 Реализация поиска в современных электронных библиотеках 23
1.4 Технологии разработки 25
1.4.1 Экстремальное программирование (XP) 25
1.4.2 Технология MSF 27
1.4.3 Технология RUP 28
1.4.4 Выбор технологии разработки 30
1.5 Анализ целевой аудитории 30
1.6 Структурное и календарное планирование для IT-проекта разработки электронной библиотеки 33
Выводы 48
2 Разработка электронной библиотеки и алгоритма поиска в ней книг 49
2.1 Онтологическая модель предметной области 49
2.2 Имитационное моделирование объекта исследования 51
2.2.1 Проверка адекватности имитационных моделей 71
2.3 Анализ требований 74
2.3.1 Анализ функциональных требований 74
2.3.2 Анализ нефункциональных требований 78
2.4 Проектирование интерфейса пользователя 79
2.5 Проектирование базы данных 87

2.6 Разработка улучшенного алгоритма поиска книг в электронной библиотеке 91
Выводы 96
3 Реализация электронной научной библиотеки 98
3.1 Разработка прототипа электронной научной библиотеки 98
3.1.1 Реализация базы данных 98
3.1.2 Разработка серверной части 99
3.1.3 Разработка клиентской части 101
3.1.4 Тестирование серверной части приложения 103
3.1.5 Тестирование клиентской части приложения 106
3.1.5.1. Функциональное тестирование 106
3.1.5.2. Модульное тестирование 107
3.1.6 Метрики кода 109
3.1.6.1. Метрики размера программы 110
3.1.6.2. Метрики Холстеда 111
3.2 Разработка прототипа программы, реализующей улучшенный алгоритм поиска книг в электронной библиотеке 113
3.2.1 Разработка и тестирование программы 113
3.2.2 Эксперимент по проверке эффективности программы, реализующей предложенный модифицированный алгоритм 114
Выводы 117
Заключение 118
Список использованных источников 121
Приложение А. Глоссарий 131
Приложение Б. Исходный код имитационных моделей 133
Приложение В. Верификация имитационной модели 139
Приложение Г. Исходный код скриптов базы данных 145
Приложение Д. Исходный код серверной части 151
Приложение Е. Исходный код клиентской части 163
Приложение Ж. Результаты тестирования серверной части 187
Приложение И. Результаты тестирования клиентской части 194

Приложение К. Исходный код программы, реализующей модифицированный алгоритм поиска книг 204
Приложение Л. Дипломы победителя международных конкурсов научных работ 212
Приложение М. Свидетельство о государственной регистрации программы для ЭВМ 214

Введение

Актуальность темы. В наше время информационные технологии все шире распространяются на разнообразные сферы человеческой деятельности, кардинально изменяя 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. С этой целью в 2017 году были утверждены «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы» и государственная программа «Цифровая экономика». В них был взят курс на формирование национальной 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» [1, 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, в первую очередь мобильному. 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 информационных технологий организации, как библиотеки. Это связано с появлением электронных версий книг, имеющих перед классическими бумажными экземплярами несколько важных преимуществ. Во-первых, они не занимают много места и доступ к ним можно обеспечить практически откуда угодно при наличии планшета, смартфона или иного компактного гаджета.

Кроме того, получить электронную версию какой-либо книги можно гораздо быстрее, чем печатную, что сильно упрощает библиотекам обмен информацией между собой, а также с другими организациями, с которыми они сотрудничают. Это особенно ценно для научных библиотек, которым требуется постоянно обновлять свои коллекции в связи с частыми выходами новых научных трудов.

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» [3]. В ней среди важнейших задач, стоящих перед современными библиотеками, указывается их модернизация путем «перевода основных процессов, обеспечивающих жизнедеятельность библиотек, на цифровые технологии». Она подразумевает, помимо прочего, полномасштабную оцифровку книжных фондов библиотек [4]. Об этом же говорит и статистика, свидетельствующая об устойчивом росте посещаемости основных электронных научных библиотек России [5, 6].

Таким образом, на основании приведенных выше факторов можно сделать вывод о том, что тема диссертации «Модель и алгоритм функционирования программных средств электронных библиотек» весьма актуальна и перспективна в условиях современного общества.

В настоящее время к проблематике данного направления можно отнести тот факт, что недостаточное внимание уделяется проблеме

оптимальной организации поиска книг в составе электронной библиотеки, что предполагает разработку более эффективного алгоритма поиска.

Также целесообразно расширение существующей классификации электронных библиотек путем добавления новых классификационных признаков и видов библиотек.

Кроме того, в процессе разработки электронных научных библиотек недостаточно используется аппарат онтологического и имитационного моделирования для проектирования, комплексного исследования и оптимизации создаваемых электронных библиотек.

Цель магистерской диссертационной работы состоит в анализе, исследовании, проектировании и последующей реализации электронной библиотеки с использованием средств онтологического, визуального и математического моделирования на примере электронной научной библиотеки.

Задачи исследования. Для достижения поставленной цели необходимо решить следующие задачи.

1. Критический анализ предметной области по тематике диссертации.
2. Актуализация существующей классификации электронных библиотек путем добавления новых классификационных признаков и систем.
3. Разработка онтологической модели предметной области.
4. Создание набора дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования электронной научной библиотеки.
5. Разработка сетевого графика IT-проекта по созданию программных средств электронной библиотеки, определение срока и бюджета его выполнения.
6. Разработка модифицированного алгоритма поиска книжных источников в электронной библиотеке, реализация и тестирование разработанного алгоритма, проведение эксперимента для оценки его эффективности.

7. Регистрация программной реализации разработанного модифицированного алгоритма поиска и подготовка к внедрению разработанной и реализованной электронной научной библиотеки.

Объектом исследования является разрабатываемая электронная научная библиотека.

Предметом исследования являются модели и алгоритмы функционирования программных средств электронных библиотек.

Методы исследования основаны на методах онтологического, имитационного и визуального моделирования, математическом аппарате сетевого планирования и управления, методах теории вероятностей и математической статистики, методах теории массового обслуживания, современных методах анализа качества программного кода, методах проектирования программных средств.

Научная новизна работы заключается в следующем.

1. Предложен модифицированный алгоритм поиска книг в электронной библиотеке, отличающийся от алгоритма расширения выборки, на основе которого он создан, большей скоростью поиска при сохранении приемлемой надежности.

2. Предложен набор дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования электронной научной библиотеки. Важной особенностью предложенного набора моделей является то, что они, в отличие от известных аналогичных моделей, имеют комплексный характер и позволяют учитывать как особенности работы аппаратного обеспечения электронной библиотеки, так и человеческий фактор, связанный с деятельностью ее сотрудников.

3. Предложена расширенная онтологическая модель предметной области для электронных библиотек, отличающаяся от известных расширенным набором сущностей и классов предметной области, с целью формализации и систематизации информации для эффективного проведения проектирования объекта исследования.

4. Предложена модифицированная классификация электронных библиотек, отличающаяся от известных тем, что предлагается более полный расширенный перечень классификационных признаков и систем.

Практическая значимость исследований. По материалам диссертационной работы получено Свидетельство о государственной регистрации программы для ЭВМ в Федеральном институте промышленной собственности РФ. Проведена подготовка к внедрению результатов диссертационной работы. С помощью математического аппарата сетевого планирования и управления и программного продукта Microsoft Project было проведено структурное и календарное планирование программного проекта разработки электронной научной библиотеки, в результате которого были определены срок и бюджет проекта, показавшие его экономическую целесообразность и практическую значимость.

Основные положения, выносимые на защиту.

1. Предложенный модифицированный алгоритм поиска книг в электронной библиотеке, отличающийся от алгоритма расширения выборки, на основе которого он создан, большей скоростью поиска при сохранении приемлемой надежности.

2. Предложенный набор дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования электронной научной библиотеки. Важной особенностью предложенного набора моделей является то, что они, в отличие от известных аналогичных моделей, имеют комплексный характер и позволяют учитывать как особенности работы аппаратного обеспечения электронной библиотеки, так и человеческий фактор, связанный с деятельностью ее сотрудников.

3. Предложенная расширенная онтологическая модель предметной области для электронных библиотек, отличающаяся от известных расширенным набором сущностей и классов предметной области, созданная с целью формализации и систематизации информации для эффективного проведения проектирования объекта исследования.

4. Предложенная модифицированная классификация электронных библиотек, отличающаяся от известных тем, что предлагается более полный расширенный перечень классификационных признаков и систем.

Апробация работы. Основные результаты работы были доложены и обсуждены на следующих конференциях.

1. Всероссийская (национальная) научная конференция «Фундаментальные и прикладные исследования. Актуальные проблемы и достижения». – Санкт-Петербург, январь 2022.

2. Международная научная конференция «Высокие технологии и инновации в науке». – Санкт-Петербург, январь 2022.

3. Международная студенческая научная конференция «Поколение будущего». – Санкт-Петербург, январь 2022.

4. IX Ежегодная всероссийская межвузовская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза, март 2022.

5. Научно-технический семинар кафедры «МОиПЭВМ» ПГУ. – Пенза, октябрь 2022.

6. Научно-технический семинар кафедры «МОиПЭВМ» ПГУ. – Пенза, ПГУ, март 2023 г.

7. X Ежегодная Всероссийская межвузовская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза, ПГУ, март 2023 г.

8. XV Международная научно-практическая конференция «Современные тенденции развития науки и мирового сообщества в эпоху цифровизации». – Ростов-на-Дону, июнь 2023.

Публикации. По теме диссертационного исследования опубликовано 10 печатных работ, одна из которых опубликована в научном журнале, входящем в перечень ВАК.

1. 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.

2. Ащеулов, И.О. Исследование электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов // Молодежный исследовательский потенциал: сборник статей V Международного научно-исследовательского конкурса. – Петрозаводск: МЦНС «Новая наука», 2022. – С. 77 – 89.

3. Ащеулов, И.О. Исследование работы сервера электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Материалы всероссийской (национальной) научной конференции «Фундаментальные и прикладные исследования. Актуальные проблемы и достижения». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 11–13.

4. Ащеулов, И.О. Исследование работы читального зала электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Материалы международной научной конференции «Высокие технологии и инновации в науке». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 81–84.

5. Ащеулов, И.О. Исследование работы администраторов электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Материалы международной студенческой научной конференции «Поколение будущего». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 79–83.

6. Ащеулов, И.О. Исследование работы оцифровщиков книг электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Информационные технологии в науке и образовании. Проблемы и перспективы: сборник статей по материалам IX Всероссийской научно-практической конференции. – Пенза: Изд-во ПГУ, 2022. – С. 73–76.

7. Ащеулов И.О., Князев В.Н. Программа поиска книг в электронной библиотеке // Свидетельство о государственной регистрации

программы для ЭВМ № 2023616960. Заявка № 2023614793, дата поступления 13.03.2023 г., дата государственной регистрации в Реестре программ для ЭВМ 04.04.2023 г. – М., Федеральный институт промышленной собственности, 2023.

8. Ащеулов, И.О. Разработка алгоритма поиска книг в электронной библиотеке / И.О. Ащеулов, В.Н. Князев // Сборник материалов X Ежегодной Всероссийской межвузовской научно-практической конференции «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза: Издательство ПГУ, 2023. – С. 167 – 171.

9. Ащеулов, И.О. Вопросы создания модифицированного алгоритма поиска книг в электронной библиотеке / И.О. Ащеулов, В.Н. Князев // Сборник материалов XV Международной научно-практической конференции «Современные тенденции развития науки и мирового сообщества в эпоху цифровизации». – Махачкала: Издательство «Алеф», 2023. – С. 202 – 208.

10. Ащеулов, И.О. Разработка модифицированного алгоритма поиска книг в электронной библиотеке / И.О. Ащеулов, В.Н. Князев // Международный научно-исследовательский журнал. – 2023. – №7 (перечень ВАК) (в печати).

Конкурсы научных работ. По теме магистерской диссертации получены следующие документы:

- диплом победителя XX Международного научно-исследовательского конкурса «Научные достижения и открытия 2021» (2021 г.);
- диплом лауреата I степени V Международного научно-исследовательского конкурса «Молодежный исследовательский потенциал» (2022 г.).

Структура и объём работы. Диссертация состоит из введения, трех глав, заключения, списка использованных источников и приложений.

1 Анализ предметной области исследования

1.1 Понятие электронной библиотеки и классификация электронных библиотек

Для того чтобы сформулировать смысл понятия электронной библиотеки, обратимся к государственному стандарту в этой сфере – ГОСТ Р 7.0.96 – 2016 «Система стандартов по информации, библиотечному и издательскому делу. Электронные библиотеки. Основные виды. Структура. Технология формирования». Согласно указанному государственному стандарту, введенному впервые в Российской Федерации, электронная библиотека – это «информационная система, предназначенная для организации и хранения упорядоченного фонда электронных объектов, и обеспечения доступа к ним с помощью единых средств навигации и поиска» [7]. Фонд электронной библиотеки формируется как совокупность хранящихся в ней электронных документов и связанных с ними метаданных, описывающих структуру документа, его содержание и прочие характеристики и использующихся в процессе поиска.

Единой классификации электронных библиотек не существует. Есть несколько частичных классификаций, основанных на различных характеристиках тех или иных библиотек [7 – 9].

По месту расположения электронные библиотеки подразделяются на следующие виды:

– локальные;

– удалённые.

По доступности электронные библиотеки делятся на следующие виды:

– открытый доступ;

– частичный доступ;

– доступ на основе подписки.

По правообладанию электронные библиотеки подразделяются на следующие виды:

– государственные библиотеки, учрежденные органами государственной власти, в том числе федеральные библиотеки, библиотеки субъектов Российской Федерации, библиотеки министерств и иных федеральных органов исполнительной власти;

– муниципальные библиотеки, учрежденные органами местного самоуправления;

– библиотеки Российской академии наук, других академий, научно-исследовательских институтов, образовательных организаций;

– библиотеки предприятий, учреждений, организаций;

– библиотеки общественных объединений;

– частные библиотеки;

– библиотеки, учрежденные иностранными юридическими и физическими лицами, а также международными организациями в соответствии с международными договорами Российской Федерации.

По принадлежности к языковой среде электронные библиотеки подразделяются на следующие виды:

– русскоязычные, англоязычные и т.д.;

– мультиязыковые.

По способу создания электронные библиотеки подразделяются на следующие виды:

– генерируемые (электронные документы создаются держателями фонда);

– агрегируемые (создаются из уже готовых электронных документов или их коллекций);

– смешанные.

Самой распространенной является классификация электронных библиотек по назначению, делящая их на следующие виды [7 – 11]:

– универсальные;

– специализированные (например, научные библиотеки, библиотеки художественной литературы и т.д).

В рамках выполнения данной работы на основе приведенных выше частичных классификаций была создана общая классификация электронных библиотек, представленная на рисунке 1.

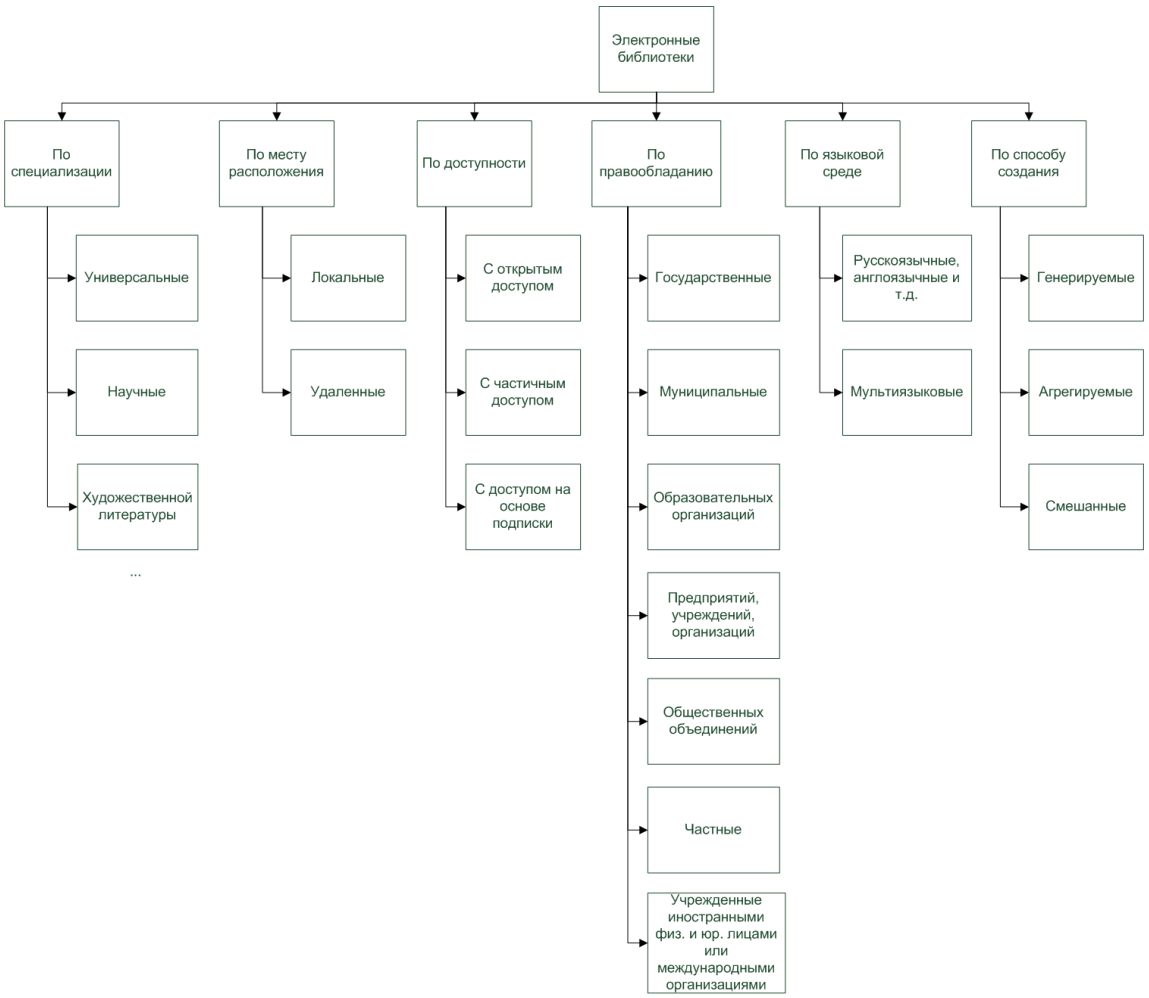


Рисунок 1 – Исходная общая классификация электронных библиотек

По итогам постановки задачи диссертационной работы был сделан вывод о целесообразности расширения существующей классификации электронных библиотек путем добавления новых классификационных признаков. В предложенной расширенной классификации был добавлен еще один классификационный признак: наличие электронного читального зала. По нему электронные библиотеки разделяются на два класса:

– электронные библиотеки с электронным читальным залом;
– электронные библиотеки без электронного читального зала.

Для полученной расширенной классификации общий список классификационных признаков и классов, выделяемых по каждому из них, представлен на рисунке 2.

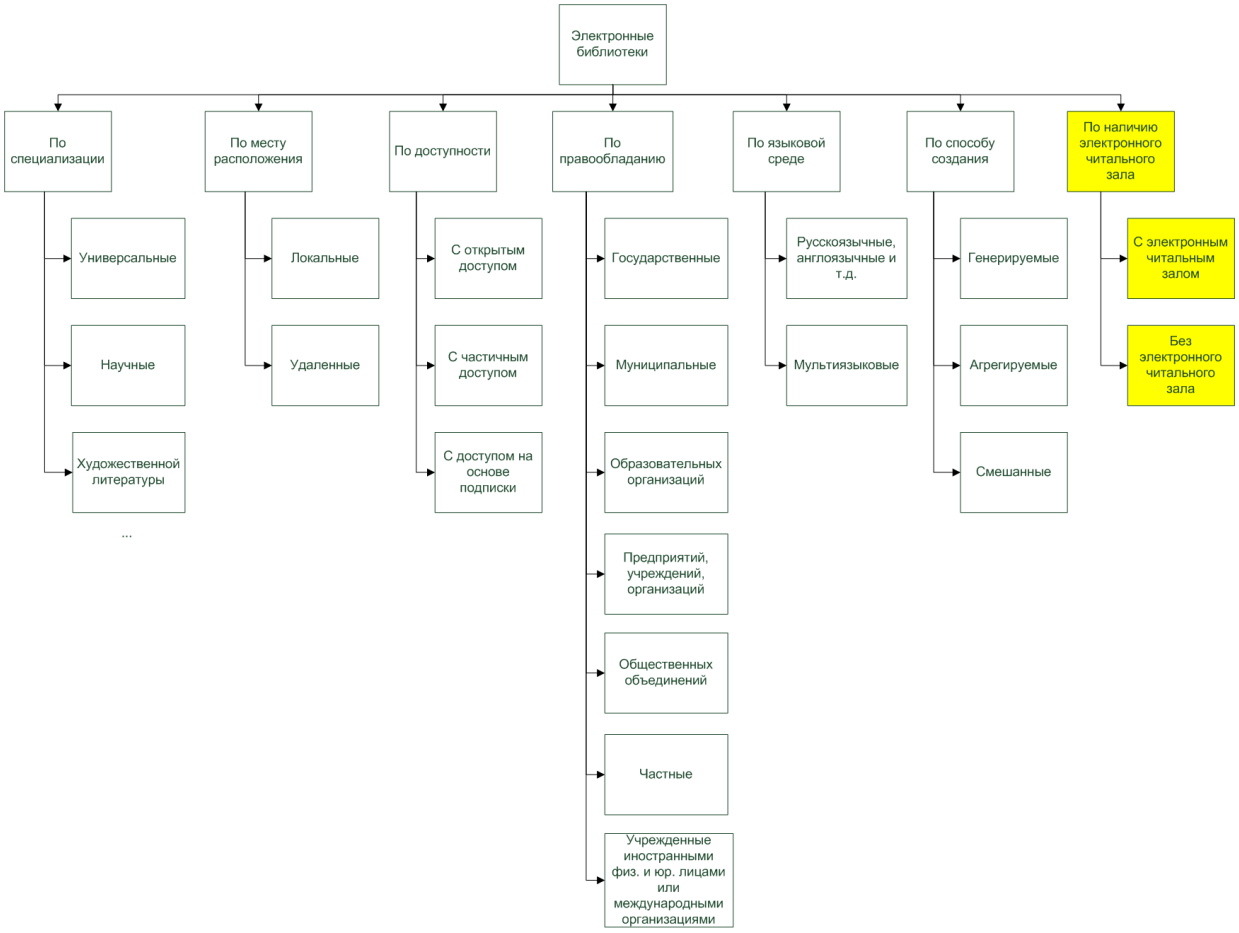


Рисунок 2 – Расширенная классификация электронных библиотек

Предложенная расширенная классификация электронных библиотек позволяет учитывать современные тенденции в области информационных систем и технологий и программной инженерии.

Созданная расширенная классификация была применена для классифицирования разрабатываемой электронной библиотеки. Разрабатываемая библиотека по специализации является научной, по месту расположения – удаленной, по доступности – библиотекой с частичным доступом, по правообладанию – частной, по языковой среде – русскоязычной, по способу создания – генерируемой, по наличию электронного читального зала – библиотекой без электронного читального зала.

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 с целью обеспечения доступа к иностранным научным изданиям для ученых нашей страны. В 2005 году она стала работать с русскоязычными публикациями, впоследствии став крупнейшей русскоязычной электронной научной библиотекой [12 – 14]. Также на основе этого портала был создан Российский индекс научного цитирования (РИНЦ).

Главная страница электронной библиотеки eLibrary представлена на рисунке 3.

Рисунок 3 – Главная страница электронной библиотеки eLibrary

На сегодня пользователям портала eLibrary доступны более 34 млн научных публикаций и патентов, включая свыше 5600 российских научно-технических журналов. В системе зарегистрировано более 3600 организаций и свыше 3,5 миллионов индивидуальных пользователей из 125 различных стран [15]. Ежегодно читатели просматривают более 12 миллионов полнотекстовых статей и свыше 90 миллионов аннотаций [16]. Для каждой книги содержится информация о ее наличие 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, в которую входят оцифрованные фонды множества российских библиотек. В число участников проекта входит ряд крупнейших библиотек нашей страны, таких как Российская государственная библиотека, Российская национальная библиотека, Российская государственная детская библиотека, Государственная публичная научно-техническая библиотека, Государственная публичная историческая библиотека и т.д. [17].

Главный экран приложения представлен на рисунке 4.

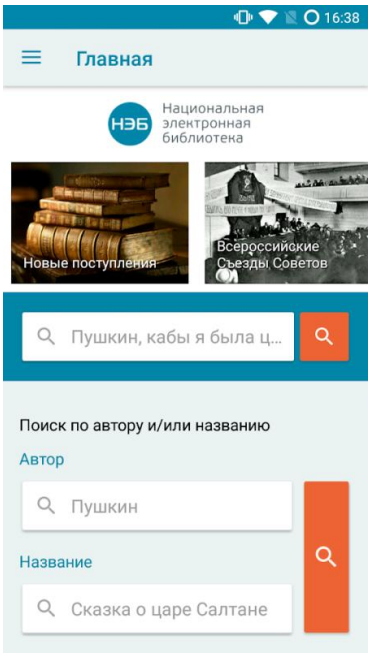


Рисунок 4 – Главный экран мобильного приложения «НЭБ»

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;
- периодические издания;
- авторефераты и диссертации;
- патентные документы;
- картографические издания;
- нотные издания.

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 [18].

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 приобретают дополнительные возможности, такие как создание закладок, личных подборок, заметок или сохранение истории поиска [19].

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 обеспечивает бесплатный открытый доступ к научным публикациям в электронном виде, размещаемым по договорённости с правообладателями по лицензии Creative Commons Attribution (CC-BY) [20].

Главная страница электронной библиотеки представлена на рисунке 5.

Рисунок 5 – 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 [21].

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 Общие выводы по анализу 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

Кроме того, во всех указанных аналогах отсутствует информация о проведении онтологического и имитационного моделирования, а также о разработке улучшенного алгоритма поиска книг.

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 Реализация поиска в современных электронных библиотеках

Поиск – важный компонент выявления объектов в какой-либо библиотечной системе. Он должен быть способен находить в ней информационные ресурсы по их свойствам или содержанию, при этом предоставляя пользователю как можно более широкий функционал.

Как правило, поиск начинается с формирования поисковых запросов. В простейшем случае это некий набор задаваемых пользователем параметров.

Обычно электронные библиотеки предоставляют следующие варианты поиска [22 – 24]:

- простой или стандартный поиск;
- расширенный поиск;
- профессиональный поиск.

Рассмотрим данные варианты подробнее.

Стандартный поиск применяется наиболее широко. Это простейший вариант поиска, предоставляющий некоторые минимальные возможности,

которые легки в использовании, но при этом достаточно полны. Он выполняется во всех текстах и описательных данных библиотеки.

В современных системах ЭБ можно сузить пространство поиска, выделив некоторое подмножество разыскиваемых информационных ресурсов по определенным характеристикам. Также в современных ЭБ обычно возможно производить многошаговый поиск, в котором новая итерация производится только среди результатов предыдущей, что может дать более релевантные результаты.

Для результатов поиска возможна сортировка по целому ряду характеристик на выбор. В роли варианта по умолчанию обычно выступает сортировка по релевантности.

Расширенный поиск предоставляет все возможности простого, а также дает некоторые дополнительные возможности, такие как использование в запросе логических операторов или одновременная фильтрация сразу по ряду критериев.

Профессиональный поиск предназначен для опытных пользователей. Он использует синтаксис поискового языка и осуществляется по следующим принципам:

– допустимы любые выражения, допустимые также в простом поиске;

– возможно наличие в поисковом запросе значений полей поиска, указанные по принципу «Имя – значение», а также логических операторов и специальных символов.

Как можно увидеть, организация поиска в современных электронных библиотеках – достаточно сложный процесс. При этом от его правильной организации очень сильно зависят как скорость выполнения поиска, так и релевантность его результатов.

В настоящее время существует множество популярных алгоритмов поиска, пригодных для самых различных ситуаций, однако они, как правило, не подходят для применения в электронной библиотеке. Так, алгоритм

последовательного поиска, основанный на поочередной проверке соответствия критериям поиска для всех элементов последовательности данных, прост в реализации и не требует каких-то предварительных условий, но крайне неэффективен по времени [25]. Другие алгоритмы, вроде алгоритма бинарного поиска, гораздо быстрее, но требуют предварительной сортировки данных по тому их атрибуту, по которому должен проводиться поиск, что в условиях современной электронной библиотеки невозможно, так как поиск может производиться сразу по нескольким критериям, никак не связанным друг с другом.

Таким образом, разработка такого алгоритма поиска книг внутри электронной библиотеки, который позволил бы находить нужную книгу с максимальной скоростью, при этом позволяя пользователю осуществлять поиск в соответствии с любыми нужными ему параметрами, является важным этапом создания качественной электронной библиотеки.

1.4 Технологии разработки

Технологией разработки программного обеспечения называется система принципов, понятий, методов и методологий, задающих стиль разработки ПО. Выбор для использования конкретной технологии разработки из множества существующих зависит от размера и слаженности команды разработчиков, а также от особенностей самого выполняемого проекта.

1.4.1 Экстремальное программирование (XP)

Суть методологии можно описать рядом принципов [26].

1. Тестирование – используется многоуровневая система тестирования программного кода, обеспечивающая максимально быстрое выявление и устранение в нем ошибок.

2. Четкое планирование – процесс начинается с постановки задания и создания примерного плана разработки и графика реализации каждой части плана. План дополняется в процессе уточнения задачи.

3. Плотный контакт с заказчиком – согласно философии XP под заказчиком подразумевается не тот, кто платит за работу, а конечный пользователь системы. Необходим постоянный контакт с пользователями, которые должны участвовать в тестировании промежуточных версий.

4. Парное программирование – особый способ организации рабочего коллектива. Над программированием системы работают пары специалистов за одним компьютером. Один из них занимается написанием самого кода, другой – наблюдает за точностью работы напарника. При необходимости пары меняются местами. Продолжительность работ каждой команды жестко не регламентируется. В процессе работы команды могут ротироваться, что позволяет добиться того, чтобы каждый программист имел представление не только о своем фрагменте системы, но и обо всей программе в целом.

5. Повторяющиеся итерации – обычно предполагается, что проект окажется запущен для выполнения только один раз, после того, как программный код будет полностью написан. XP же требует, чтобы работоспособность системы проверялась многократно, после выполнения каждого крупного шага, что позволит ускорить выявление ошибок и отладку приложения.

6. Рефакторинг или переработка кода – после выполнения каждой задачи призводится ревизия написанного кода и его последующая оптимизация, чтобы сделать систему быстрее, надежнее и безопаснее.

7. Ранний релиз конечного продукта – сторонники XP стараются всячески ускорить разработку системы, чтобы максимально быстро создать ее первую версию. Благодаря этому заказчик получает первые результаты (прибыль, оптимизацию рабочих процессов) куда быстрее, чем при других вариантах разработки. При этом разработчики, в свою очередь, быстрее получают первые отзывы пользователей и заказчика, что позволяет учесть мнение пользователей в процессе дальнейшей работы над системой. Это делает работу над новыми версиями приложения эффективнее, а само

разрабатываемое ПО – более дружественным для пользователей и полезным для заказчика.

8. Простое проектирование – проектирование системы выполняется в несколько этапов с учетом изменяющихся условий работы над проектом, что позволяет сэкономить время проектировании и увеличить его эффективность.

9. Разработка метафоры системы – представляет собой создание архитектуры системы, задание ее формы и внутренней структуры. В рамках экстремального программирования требуется как можно более четкое и детальное описание архитектуры системы, а в случае необходимости – коррекция соответствующей информации.

10. Жесткие стандарты оформления кода – программисты должны придерживаться одних и тех же стандартов оформления программного кода для обеспечения взаимопонимания между разработчиками и во избежание непредвиденных задержек, вызванных отсутствием такового.

11. Коллективная работа – специалистам предоставляется значительная свобода. Каждый может предлагать изменить тот или иной фрагмент кода. Ответственность за проект несет вся команда разработчиков целиком.

1.4.2 Технология MSF

Microsoft Solutions Framework (MSF) – технология разработки ПО от корпорациии Microsoft. Она применяется для создания приложений платформы .NET и прочих объектно-ориентированных программных средств. В основе данной технологии – гибкий процесс разработки, уделяющий при этом существенное внимание вопросам производительности и безопасности создаваемого программного продукта.

Базовые концепции и принципы модели процессов MSF [27]:

- единое видение проекта – все участники проекта должны четко понимать его цель;

- управление компромиссами – поиск компромиссов между ресурсами, сроками и функциональными требованиями проекта;
- гибкость – готовность к изменяющимся проектным условиям;
- концентрация на бизнес-приоритетах – сосредоточенность на выгоде, желаемой заказчиком;
- поощрение свободного общения внутри проекта;
- создание базовых версий – фиксация состояния каждого компонента разрабатываемой системы и эффективное управление вносимыми в проект изменениями.

Благодаря своей гибкости и масштабируемости MSF может удовлетворить потребности команды разработчиков любого размера.

1.4.3 Технология RUP

Rational Unified Process (RUP) – это технология разработки ПО, созданная компанией Rational Software. В RUP используется итеративный подход, то есть процесс разработки представляет собой последовательность итераций, включащих в себя ряд этапов разработки (анализ требований, проектирование, реализацию и т.д.) и преследущих некоторые четко заданные цели, в рамках которых в ходе данной итерации создается частично работающая версия разрабатываемой программы. Каждая последующая итерация развивает и совершенствует результаты предыдущих. Это продолжается вплоть до создания финальной версии разрабатываемого продукта.

Одним из важнейших достоинств RUP является возможность по необходимости регулировать степень формализации проекта. По окончании каждого этапа можно создавать как все требуемые документы, так и только необходимые, или же не создавать ничего. Такой подход делает технологию очень гибкой и, благодаря этому, крайне распространенной. Гибкость RUP позволяет одной и той же команде разработчиков работать над проектами, кардинально отличающимися друг от друга по требованиям и объему.

Модель жизненного цикла RUP представляет собой весьма сложную и проработанную итеративно-инкрементную модель с элементами каскада. Она включает в себя 4 фазы и 9 типов деятельности.

Основные фазы RUP:

- начало проекта – определяются основные цели, бюджет и средства выполнения проекта, составляются его предварительные планы, цель – достижение компромисса относительно задач проекта между всеми заинтересованными лицами;
- проработка – на базе основных требований разрабатывается архитектура продукта, которая в дальнейшем ляжет в основу создаваемой системы;
- построения – выполняется детальное уточнение требований и разработка соответствующей им системы на основе ранее спроектированной архитектуры;
- передача – окончательное развертывание системы в рабочей среде и доработка под нужды пользователей мелких деталей.

В рамках каждой из фаз может быть проведено несколько итераций, их количество зависит от сложности выполняемого проекта.

Основные процессы (деятельности) RUP делятся на рабочие и поддерживающие. Рабочие деятельности включают в себя:

- моделирование предметной области (бизнес-моделирование);
- определение требований;
- анализ и проектирование;
- реализация;
- тестирование.

К поддерживающим деятельностям относятся:

- развертывание;
- управление конфигурациями и изменениями;
- управление проектом;
- управление средой проекта [28].

Один из основных столпов RUP – создание моделей с использованием унифицированного языка моделирования (Unified Modeling Language, UML). Обычно этот язык ассоциируют с моделированием программных систем, использующих парадигму ООП, хотя на самом деле он применяется куда шире благодаря своей расширяемости [29]. Он не привязан к какой-то одной методологии и может применяться в любой из существующих. Основная идея этого языка – возможность моделирования ПО и других систем в виде наборов взаимодействующих объектов. UML-модели включают в себя две составляющих:

- статическая структура – описание основных типов объектов в системе и связей между ними;
- динамическое поведение – описание жизненных циклов объектов внутри системы и их взаимодействие в процессе ее функционирования.

1.4.4 Выбор технологии разработки

Исходя из всего выше сказанного, в данном проекте будут использоваться технология RUP и диаграммы языка UML. Это связано с тем, что технология RUP универсальнее остальных рассмотренных технологий разработки, потому что может одинаково эффективно применяться как в небольших проектах с минимумом формализма, так и в больших и сложных проектах, требующих высокого уровня формализации. Кроме того, RUP не требует постоянного взаимодействия с заказчиком.

1.5 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 6.

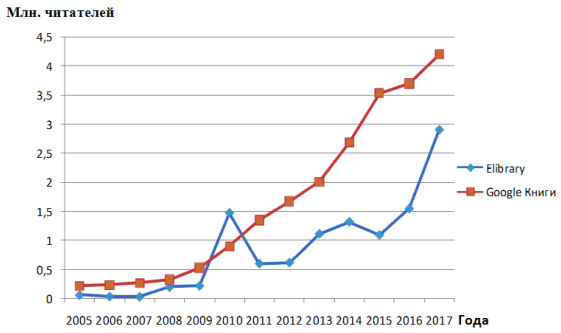


Рисунок 6 – Изменение числа пользователей электронной библиотеки elibrary

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 [5]. В последние годы эта тенденция сохранилась, и в настоящий момент число зарегистрированных пользователей elibrary превысило 4,5 млн. человек. Непрерывный рост посещаемости прослеживается и у других популярных электронных научных библиотек, таких как «КиберЛенинка» [6], а также у библиотек других направленностей, например, «Литрес», у которого количество выданных книг за последние 5 лет увеличилось в 8,5 раз [30].

На рисунке 7 представлены доли основных производителей смартфонов на мировом рынке за последние 2 года [31].

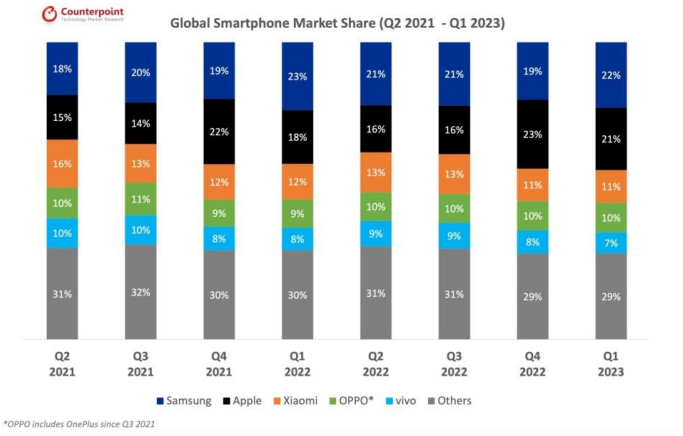


Рисунок 7 – Доли производителей смартфонов на мировом рынке

Можно заметить, что практически все основные производители смартфонов – Samsung, Xiaomi 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 [32]. 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 8 [33].

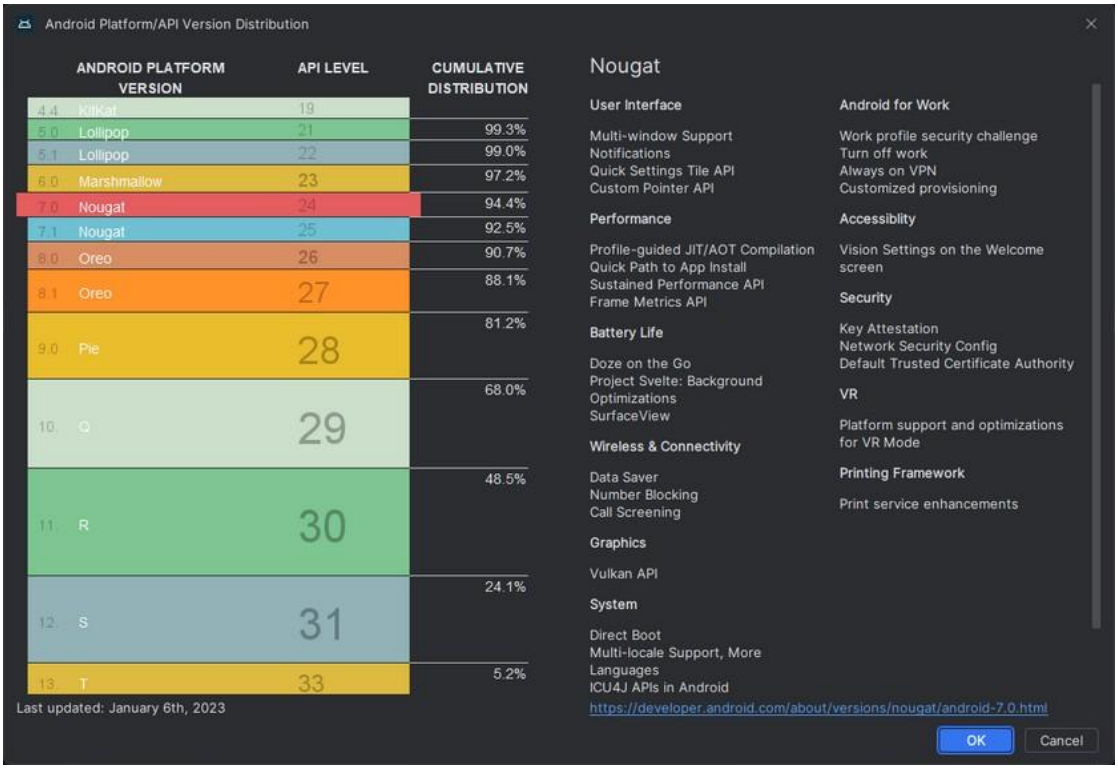


Рисунок 8 – Распространение различных версий системы Android

Можно увидеть, что 94,4 % пользователей системы Android использует версии не старее 7.0. Поэтому приложение будет разрабатываться таким образом, чтобы оно успешно функционировало для Android 7.0 и более новых версий системы.

1.6 Структурное и календарное планирование для IT-проекта разработки электронной библиотеки

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 [34 – 38].

Структурное планирование можно разделить на следующие этапы:

– разбиение проекта на совокупность отдельных работ, которые должны быть выполнены в ходе его реализации;

– построение сетевого графика, задающего порядок выполнения выделенных работ;

– оценка временных характеристик работ и анализ сетевого графика.

В результате структурного планирования должен быть построен сетевой график – ориентированный граф, вершины которого соответствуют работам проекта, а дуги – временным взаимосвязям между ними.

Сетевой график имеет следующие свойства:

– каждой работе соответствует только одна вершина;

– ни одна работа не может быть начата, пока не закончатся все непосредственно предшествующие ей работы, соответственно, пока некоторая работа не закончена, не может быть начата ни одна из следующих за ней работ;

– начало и конец проекта обозначаются работами с нулевой продолжительностью, которые называются вехами.

По сетевому графику можно определить критические работы проекта и его критический путь.

Критической называется работа, для которой задержка ее начала приведет к задержке срока окончания проекта в целом. Такие работы не

имеют запаса времени. Некритические же работы имеют некоторый запас времени, в пределах которого их начало может быть задержано.

Критический путь – это путь от начальной к конечной вершине сетевого графика, проходящий только через критические работы. Суммарная длительность работ критического пути определяет минимальное время реализации проекта. Для определения критического пути требуется найти критические работы проекта. Это выполняется в два этапа:

– вычисление раннего времени начала каждой работы проекта (времени, раньше которого она не может быть начата);

– вычисление позднего времени начала каждой работы (время, позже которого работа не может быть начата без увеличения общего времени выполнения всего проекта).

Значения раннего и позднего времени начала работ вычисляются следующим образом. Пусть t_i – время выполнения работы i, $T_p(i)$ – раннее время начала работы i, $T_{\text{п}}(i)$ – позднее время начала работы i. Тогда

$$T_p(i) = \max\left(T_p(j) + t_j\right), j \in G,$$

где G – множество работ, непосредственно предшествующих работе i. Раннее время начальной работы проекта принимается равным нулю.

Так как последняя работа – это веха с нулевой длительностью, раннее время ее начала совпадает с длительностью всего проекта. Далее нужно принять эту величину уже за позднее время начала последней работы, для остальных работ позднее время начала вычисляется по формуле:

$$T_{\text{п}}(i) = \min\left(T_{\text{п}}(j) - t_j\right), j \in H,$$

где H – множество работ, непосредственно следующих за работой i.

Определим критические работы и критический путь для программного проекта разработки электронной библиотеки.

Для начала требуется определить задачи, выполняемые в ходе работ по проекту, и ресурсы, необходимые для разработки приложения.

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

29	Обучение сотрудников библиотеки работе с системой	Программист1 Компьютер Мобильное устройство	75% 1 шт. 1 шт.
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 9).

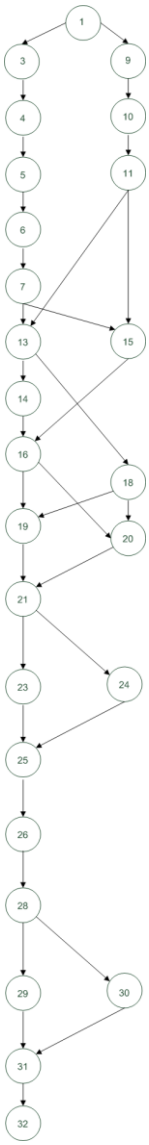


Рисунок 9 – Сетевой график проекта

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

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

Таблица 7 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	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 10 представлен граф проекта, на котором штриховыми линиями указан критический путь.

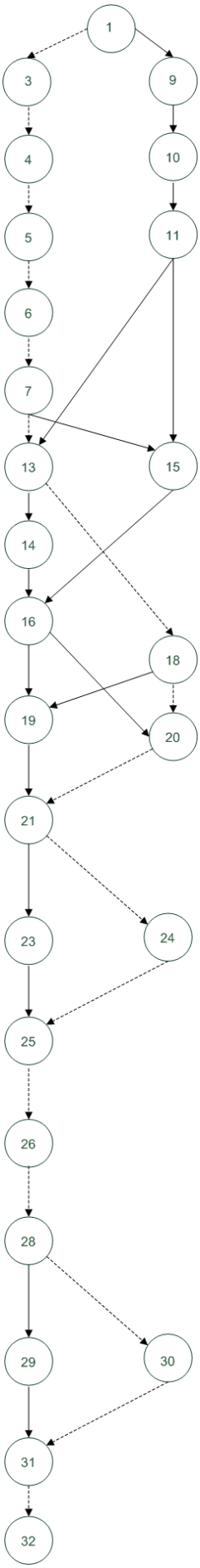


Рисунок 10 – Критический путь проекта

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 [39 – 42]. Список задач из таблицы 1, внесенный в систему Microsoft Project, представлен на рисунке 11. В качестве времени начала проекта была принята дата 06.12.2021.

	Название задачи	Длительность	Начало	Окончание	Предшественники
1	Начало проекта	0 дней	Пн 06.12.21	Пн 06.12.21	
2	**Анализ требований**	**20 дней**	**Пн 06.12.21**	**Пт 31.12.21**	
3	Анализ приложений-аналогов	7 дней	Пн 06.12.21	Вт 14.12.21	1
4	Анализ библиотечного фонда	10 дней	Ср 15.12.21	Вт 28.12.21	3
5	Консультация с руководством библиотеки	2 дней	Ср 29.12.21	Чт 30.12.21	4
6	Принятие окончательного решения	1 день	Пт 31.12.21	Пт 31.12.21	5
7	Анализ требований завершен	0 дней	Пт 31.12.21	Пт 31.12.21	6
8	**Выбор средств реализации проекта**	**7 дней**	**Пн 06.12.21**	**Вт 14.12.21**	
9	Анализ возможных программных средств	4 дней	Пн 06.12.21	Чт 09.12.21	1
10	Выбор программных средств для реализации приложения	3 дней	Пт 10.12.21	Вт 14.12.21	9
11	Выбор средств завершен	0 дней	Вт 14.12.21	Вт 14.12.21	10
12	**Проектирование приложения**	**8 дней**	**Пн 03.01.22**	**Ср 12.01.22**	
13	Проектирование базы данных	5 дней	Пн 03.01.22	Пт 07.01.22	7;11
14	Проектирование архитектуры системы	3 дней	Пн 10.01.22	Ср 12.01.22	13
15	Проектирование интерфейса пользователя	5 дней	Пн 03.01.22	Пт 07.01.22	7;11
16	Проектирование приложения завершено	0 дней	Ср 12.01.22	Ср 12.01.22	14;15
17	**Реализация проекта**	**30 дней**	**Пн 10.01.22**	**Пт 18.02.22**	
18	Реализация базы данных	10 дней	Пн 10.01.22	Пт 21.01.22	13
19	Реализация серверной части приложения	15 дней	Пн 24.01.22	Пт 11.02.22	16;18
20	Реализация клиентской части приложения	20 дней	Пн 24.01.22	Пт 18.02.22	16;18
21	Реализация проекта завершена	0 дней	Пт 18.02.22	Пт 18.02.22	19;20
22	**Отладка**	**17 дней**	**Пн 21.02.22**	**Вт 15.03.22**	
23	Тестирование работы серверной части системы	5 дней	Пн 21.02.22	Пт 25.02.22	21
24	Тестирование работы клиентской части системы	7 дней	Пн 21.02.22	Вт 01.03.22	21
25	Исправление ошибок	10 дней	Ср 02.03.22	Вт 15.03.22	23;24
26	Отладка завершена	0 дней	Вт 15.03.22	Вт 15.03.22	25
27	**Внедрение программного средства**	**17 дней**	**Ср 16.03.22**	**Чт 07.04.22**	
28	Внесение в систему электронных копий всех книг библиотечного фонда	7 дней	Ср 16.03.22	Чт 24.03.22	26
29	Обучение сотрудников библиотеки работе с системой	7 дней	Пт 25.03.22	Пн 04.04.22	28
30	Поиск и исправление ошибок при внедрении	10 дней	Пт 25.03.22	Чт 07.04.22	28
31	Внедрение завершено	0 дней	Чт 07.04.22	Чт 07.04.22	29;30
32	Конец проекта	0 дней	Чт 07.04.22	Чт 07.04.22	31

Рисунок 11 – 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 12.

Рисунок 12 – Диаграмма Ганта

На рисунках 13 – 15 представлены графики загрузки трудовых ресурсов «Программист1», «Программист2», «Программист3».

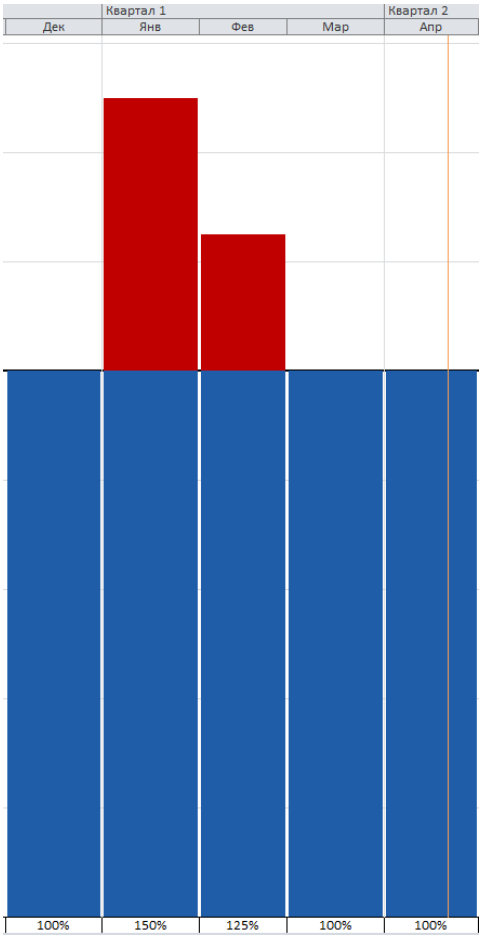


Рисунок 13 – График загрузки для ресурса «Программист1»

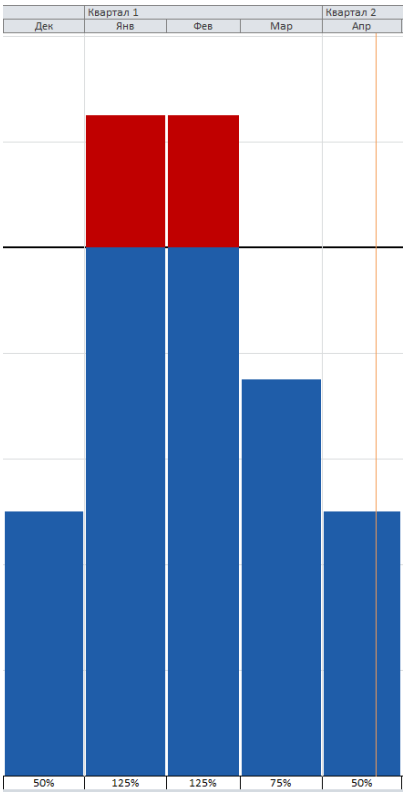


Рисунок 14 – График загрузки для ресурса «Программист2»

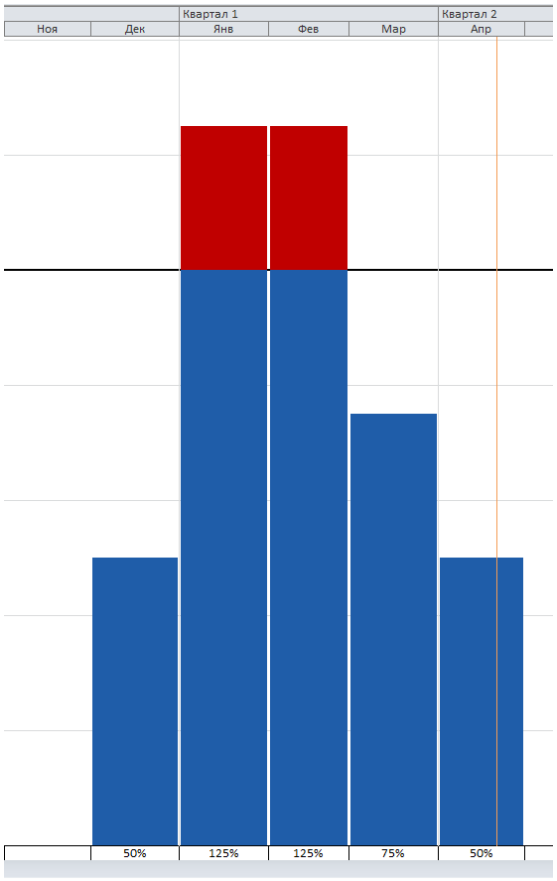


Рисунок 15 – График загрузки для ресурса «Программист3»

На рисунке 16 представлен график загрузки трудового ресурса «Дизайнер».

Рисунок 16 – График загрузки для ресурса «Дизайнер»

На рисунке 17 представлен график загрузки материального ресурса «Компьютер».

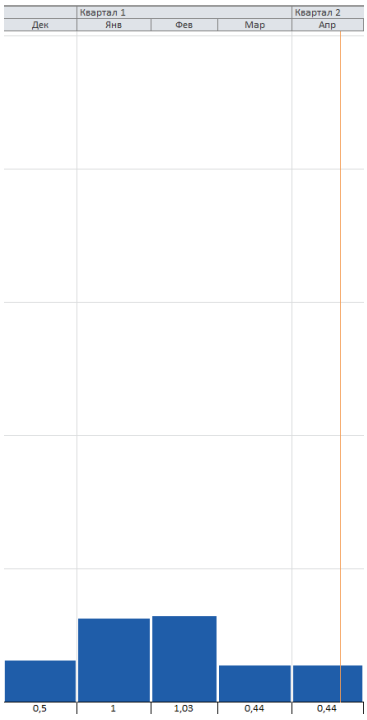


Рисунок 17 – График загрузки для ресурса «Компьютер»

На рисунке 18 представлен график загрузки материального ресурса «Мобильное устройство».

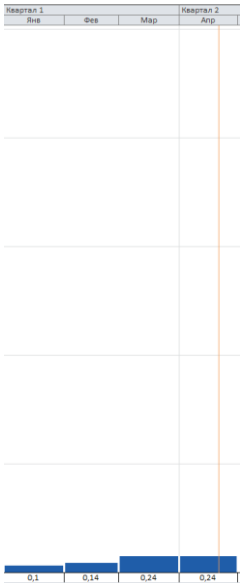


Рисунок 18 – График загрузки для ресурса «Мобильное устройство»

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

На рисунке 19 представлен обзорный отчет проекта, из которого видно, что общая длительность проекта составляет 89 дней (что совпадает со значением, вычисленным вручную), а предполагаемые затраты – 632 100 руб.

Электронная библиотека

на дату Пн 07.03.22

Даты

Начало:	Пн 06.12.21	Окончание:	Чт 07.04.22
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней

Длительность

Текущий план:	89 дней	Оставшаяся:	89 дней
Базовый план:	0 дней	Фактическая:	0 дней
Отклонение:	89 дней	Процент завершени	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 завершени	0%

Затраты

Текущий план:	632 100,00 ₽	Оставшиеся:	632 100,00 ₽
Базовый план:	0,00 ₽	Фактические:	0,00 ₽
Отклонение:	632 100,00 ₽		

Состояние задачи		Состояние ресурса	
Еще не начаты е задачи:	32	Трудовые ресурсы:	2
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	3
Завершенны е задачи:	0	Материальны е ресурсы:	2
Итого задач:	32	Итого ресурсов:	7

Рисунок 19 – Обзорный отчет проекта

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

Выводы

Анализ предметной области показал ряд слабых и сильных сторон современных электронных научных библиотек. Для того чтобы уменьшить негативные стороны и усилить позитивные, было решено разработать собственную электронную научную библиотеку с применением специально разработанного в рамках данной работы модифицированного алгоритма поиска книг, а также технологий онтологического и имитационного моделирования.

Также на основании данных из нескольких источников была сформирована общая классификация электронных библиотек, которая затем была дополнена добавлением нового классификационного признака.

Помимо этого, был проведен и обоснован выбор технологий и средств разработки.

Кроме того, было выполнено сетевое планирование проекта аналитическим путем и с помощью среды Microsoft Project, которое показало длительность реализации проекта – 89 дней и предполагаемые затраты – 632 100 руб. На основании этого можно сделать вывод об экономической целесообразности проекта.

2 Разработка электронной библиотеки и алгоритма поиска в ней книг

2.1 Онтологическая модель предметной области

Проектирование электронной научной библиотеки целесообразно проводить на основе предварительного анализа, структуризации и систематизации информации предметной области. Разработка онтологической модели предметной области начинается с онтологического анализа [43]. В среде Protege был разработан соответствующий задаче набор классов, свойств этих классов, а также набор связей и взаимодействий между ними. На этой основе была разработана на языках OWL и SWRL онтология в среде Protege. Одной из целей данной научно-исследовательской работы являлось построение онтологической модели предметной области. Онтологический граф модели представлена на рисунке 20.

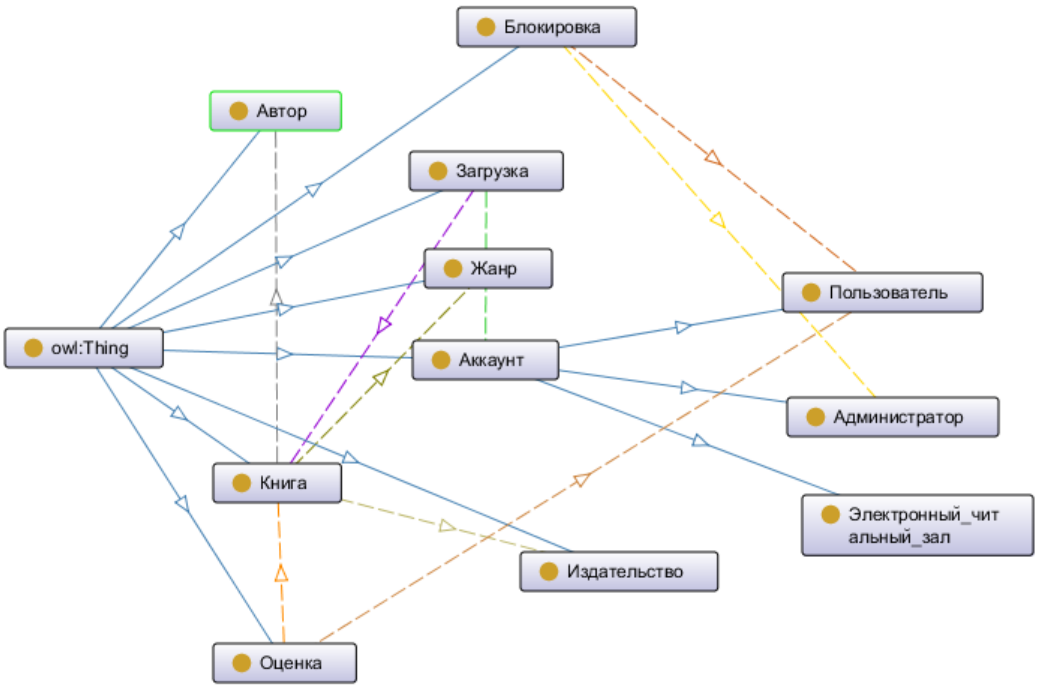


Рисунок 20 – Онтологический граф модели

Иерархия классов модели представлена на рисунке 21.

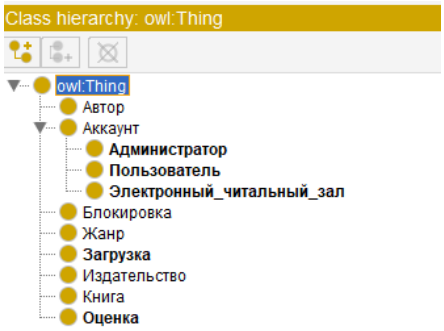


Рисунок 21 – Иерархия классов модели

Иерархия свойств объектов онтологической модели представлена на рисунке 22.

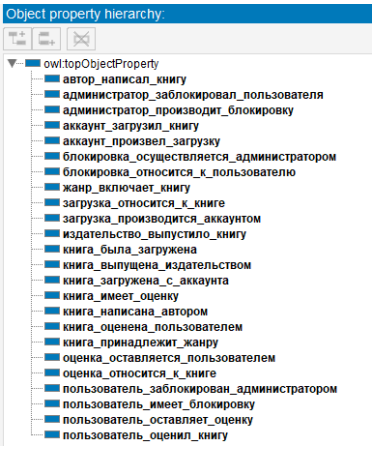


Рисунок 22 – Иерархия свойств объектов онтологической модели

Иерархия свойств данных онтологической модели представлена на рисунке 23.

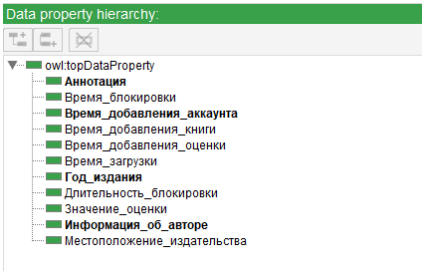


Рисунок 23 – Иерархия свойств данных онтологической модели

Онтологическая модель была построена с помощью системы Protege и позволяет на базе структуризации и систематизации информации предметной области проводить эффективное проектирование электронной научной

библиотеки, в том числе на основе визуального моделирования с применением языка UML.

Предложенная расширенная онтологическая модель предметной области для электронных библиотек отличается от известных расширенным набором сущностей и классов предметной области.

2.2 Имитационное моделирование объекта исследования

Для того чтобы обеспечить необходимую эффективность работы электронной библиотеки, нужно оптимальным образом организовать ее структуру. В этом большую помощь могут оказать технологии имитационного моделирования, которые на сегодня являются одним из наиболее востребованных инструментов в научных исследованиях, управленческой и производственной деятельности, обучении и других областях [44 – 51].

Основной функционал электронной библиотеки автоматически осуществляется ее сервером. Он обрабатывает запросы пользователей (поиск книг, загрузка необходимой книги и т.д.) и автоматически отвечает на них. От того, насколько быстро сервер обрабатывает приходящие к нему запросы, зависит, насколько быстро пользователю будет приходить ответ, что важно с точки зрения обеспечения максимально удобной работы пользователя с системой.

Однако работа электронной библиотеки включает в себя и другие аспекты, в отличие от автоматизированной обработки запросов пользователей, связанные с человеческим фактором. Одним из них является расширение книжного фонда библиотеки путем добавления в систему новых книг. Эту работу осуществляют администраторы электронной библиотеки. При этом стоит учесть, что не все книги поступают в библиотеку сразу в пригодной для непосредственного добавления в систему электронной форме. Книга также может быть и в бумажной версии, и тогда ее необходимо предварительно оцифровать, при этом, чем больше страниц в книге, тем дольше будет проходить процесс ее оцифровки. Таким образом, штат

электронной библиотеки, помимо администраторов, должен включать в себя и специалистов по оцифровке книг.

Кроме того, из-за вопросов, связанных с авторскими правами, в современных библиотеках, помимо произведений с открытым доступом, к которые может читать любой желающий, также присутствуют и книги с ограниченным доступом, которые можно просматривать только из электронных читальных залов. При этом для обеспечения функционирования электронного читального зала в нем должны присутствовать компьютеры, обеспечивающие удаленный доступ к книгам электронной библиотеки, а также библиотекари, регулирующие доступ посетителей к этим компьютерам.

Цель моделирования – определить оптимальные значения следующих характеристик электронной библиотеки: количество администраторов, количество оцифровщиков, число библиотекарей и компьютеров в электронном читальном зале, время обработки запроса сервером. Следует учесть, что слишком большие значения всех этих величин приведут к неоправданно высоким затратам на работу библиотеки, что невыгодно с экономической точки зрения, а слишком малые не позволят библиотеке нормально функционировать (сервер не будет справляться с запросами, новые книги не будут успевать добавляться в систему и т.д.). Поэтому оптимальным будет минимальное значение всех вышеупомянутых величин, при которых система работает удовлетворительно.

С учетом указанных целей исследования в среде GPSS Studio была проведена разработка четырех оригинальных дискретно-событийных имитационных GPSS-моделей: одной – для моделирования обработки сервером запросов, второй – для добавления книг администраторами, третьей – для процесса оцифровки бумажных книг перед их добавлением в библиотеку и четвертой – для моделирования работы электронного читального зала [52 – 56]. Важной особенностью этих моделей является то, что они, в отличие от известных аналогичных моделей, имеют комплексный характер и позволяют учитывать как особенности работы аппаратного

обеспечения электронной библиотеки, так и человеческий фактор, связанный с деятельностью ее сотрудников. Исходные коды моделей на языке GPSS представлены в приложении Б. Рассмотрим структуру каждой из моделей.

Q-схема модели для блока администраторов электронной библиотеки представлена на рисунке 24.

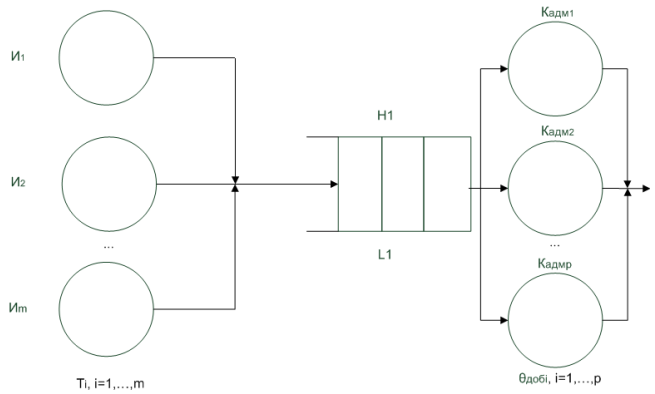


Рисунок 24 – Q-схема модели для блока администраторов

В модели отражены все этапы обработки заявок: поступление новых книг ($И_1$ – $И_m$, m – число типов книг в библиотеке), добавление книг в систему администраторами электронной библиотеки ($К_{адм1}$ – $К_{адмр}$, р – число администраторов).

Исходные количественные данные (интервал поступления заявок, время обработки заявок) для модели были получены на основе изучения статистики по распределению книг в электронных научных библиотеках по типам и по изменению числа книг в библиотеке.

Q-схема модели для блока оцифровщиков представлена на рисунке 25.

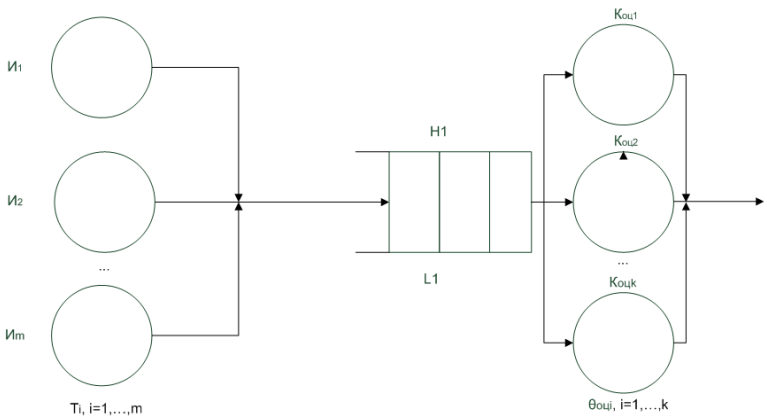


Рисунок 25 – Q-схема модели для блока оцифровщиков

Здесь также имеются все этапы обработки заявок: поступление новых бумажных книг ($И_1$ – $И_m$, m – число типов книг в библиотеке), оцифровка бумажных книг ($К_{оц1}$ – $К_{оцk}$, k – число оцифровщиков).

Исходные количественные данные (интервал поступления заявок, время обработки заявок) для модели были получены на основе изучения статистики по распределению книг в электронных научных библиотеках по типам и по изменению числа книг в библиотеке, а также по скорости работы оцифровщиков.

Q-схема модели для блока электронного читального зала представлена на рисунке 26.

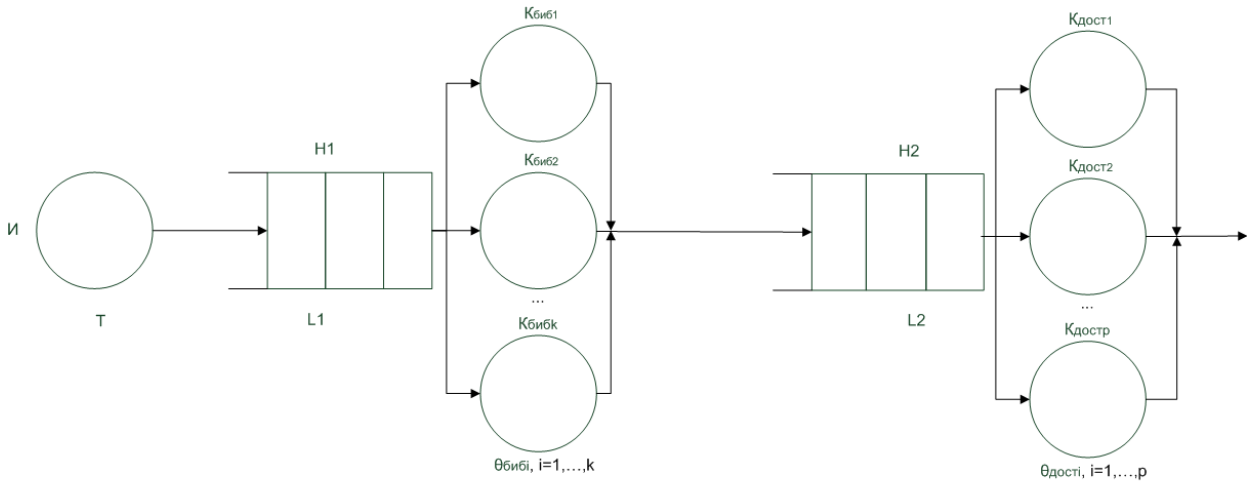


Рисунок 26 – Q-схема модели для блока электронного читального зала

В этой модели также присутствуют все этапы обработки заявок: прибытие новых читателей (И), предоставление библиотекарем доступа читателю к компьютеру электронного читального зала ($К_{биб1}$ – $К_{бибk}$, k – число библиотекарей), использование читателем компьютера для получения доступа к требуемой ему электронной книге ($К_{дост1}$ – $К_{достр}$, p – число компьютеров).

Исходные количественные данные (интервал поступления заявок, время обработки заявок) для модели были получены экспериментально путем мониторинга работы электронного читального зала библиотеки в течение нескольких рабочих дней, в результате чего были определены

средние значения указанных величин и сделано предположение об их распределении.

Q-схема модели для блока запросов к серверу представлена на рисунке 27.

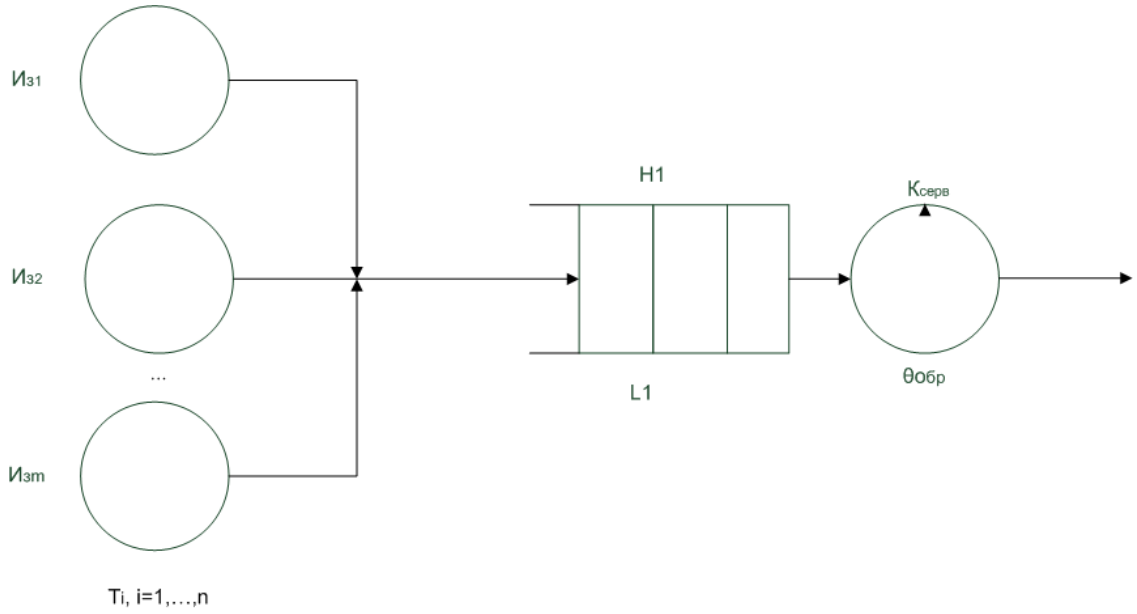


Рисунок 27 – Q-схема модели для блока запросов к серверу

В этом блоке также присутствуют все этапы обработки заявок: поступление новых запросов ($И_1 – И_m$, m – число типов книг в библиотеке) и обработка запросов сервером ($К_{серв}$).

Исходные количественные данные (интервал поступления заявок, время обработки заявок) для модели были получены на основе изучения статистики по распределению книг в электронных научных библиотеках по типам и по посещаемости электронных научных библиотек, а также по быстродействию современных серверов.

В качестве входного потока заявок используется простейший поток заявок с интервалом поступления, распределенным по экспоненциальному закону распределения, так как заявки поступают в совершенно случайный момент времени, и именно такой подход, как правило, применяется на практике при имитационном моделировании систем. Также экспоненциальное распределение принято для соответствующих длительностей обслуживания заявок разработанной многофазной сети

массового обслуживания [46, 47]. Для блоков администраторов и оцифровщиков единица модельного времени – 1 минута, а временной промежуток – 1 рабочая неделя (5 дней по 8 часов). Для блока электронного читального зала единица времени также равна 1 минуте, а временной промежуток составляет 1 рабочий день (8 часов). Для блока сервера единица времени – 1 миллисекунда, а временной промежуток равен 1 суткам.

Результаты проведенного имитационного моделирования для работы сервера представлены в таблицах 8 – 10 соответственно.

Первый эксперимент производится при следующих значениях среднего интервала времени поступления запросов на книги в миллисекундах:

- запрос на статью – 1577;
- запрос на сборник статей – 368286;
- запрос на автореферат диссертации – 473100;
- запрос на патент – 1227621;
- запрос на монографию – 46036;
- запрос на материал конференции – 1659;
- запрос на сборник трудов конференции – 122762;
- запрос на диссертацию – 47310;
- запрос на номер журнала – 13640.

Всего поступает примерно 120 000 запросов в день.

Результаты моделирования указаны в таблице 8.

Таблица 8 – Результаты моделирования обработки запросов сервером (эксперимент 1)

Время на обработку запроса на сервере, мс	Общее время обработки запроса
100	До 1.2 с
200	До 2.8 с
300	До 6.4 с
400	До 9.6 с

Пример исследования разработанной имитационной модели в среде имитационного моделирования GPSS Studio представлен на рисунке 28.

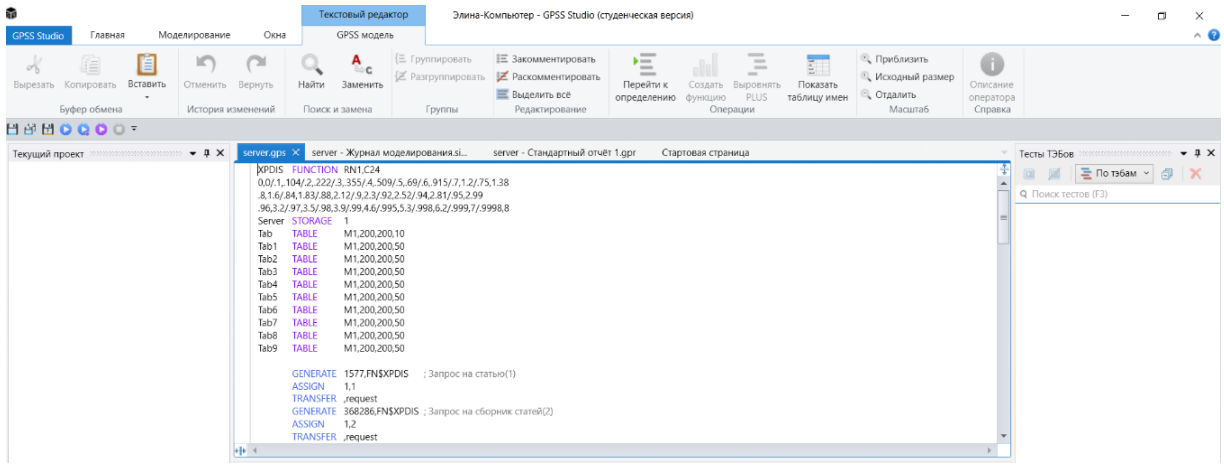


Рисунок 28 – Исследование модели в среде GPSS Studio

Также для этого эксперимента представлено несколько временных характеристик обработки заявок, полученных при оптимальных значениях параметров обработки (время обработки запроса – 100 мс). На рисунке 29 представлены средневзвешенное значение и среднеквадратичное отклонение времени обработки заявок.

Рисунок 29 – Средневзвешенное значение и среднеквадратичное отклонение

На рисунке 30 представлена гистограмма частот времени обработки заявок.

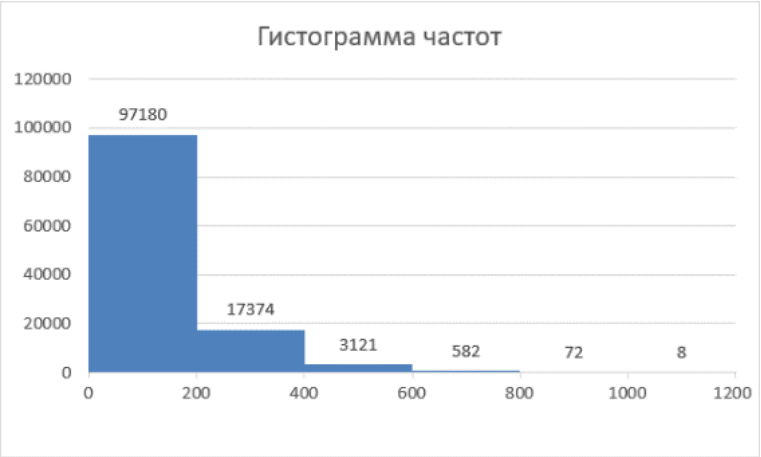


Рисунок 30 – Гистограмма частот

На рисунке 31 представлена эмпирическая функция распределения времени обработки заявок.

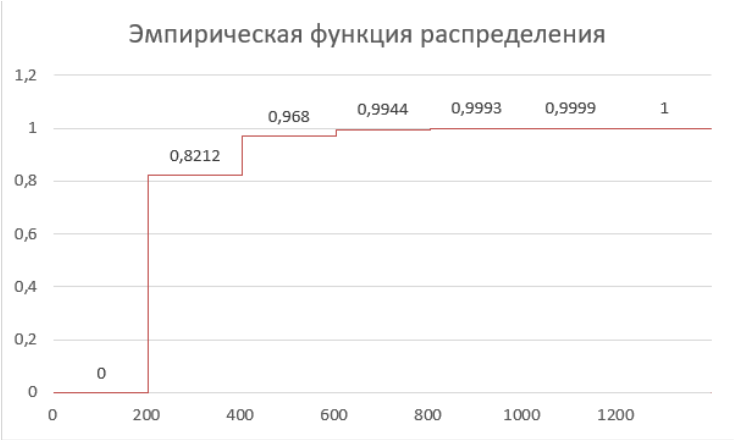


Рисунок 31 – Эмпирическая функция распределения

Второй эксперимент производится при следующих значениях среднего интервала времени поступления запросов на книги в миллисекундах:

– запрос на статью – 2103;
– запрос на сборник статей – 491048;
– запрос на автореферат диссертации – 630800;
– запрос на патент – 1636828;
– запрос на монографию – 61381;
– запрос на материал конференции – 2212;
– запрос на сборник трудов конференции – 163683;
– запрос на диссертацию – 63080;

– запрос на номер журнала – 18187.

Всего поступает примерно 90 000 запросов в день.

Результаты моделирования указаны в таблице 9.

Таблица 9 – Результаты моделирования обработки запросов сервером (эксперимент 2)

Время на обработку запроса на сервере, мс	Общее время обработки запроса
100	До 1.4 с
200	До 3 с
300	До 4.8 с
400	До 7.2 с

Третий эксперимент производится при следующих значениях среднего интервала времени поступления запросов на книги в миллисекундах:

– запрос на статью – 3154;

– запрос на сборник статей – 736572;

– запрос на автореферат диссертации – 946200;

– запрос на патент – 2455242;

– запрос на монографию – 92072;

– запрос на материал конференции – 3318;

– запрос на сборник трудов конференции – 245524;

– запрос на диссертацию – 94620;

– запрос на номер журнала – 27280.

Всего поступает примерно 60 000 запросов в день.

Результаты моделирования указаны в таблице 10.

Таблица 10 – Результаты моделирования обработки запросов сервером (эксперимент 3)

Время на обработку запроса на сервере, мс	Общее время обработки запроса
100	До 1.6 с
200	До 2.4 с
300	До 4.4 с
400	До 5.4 с

Таким образом, можно убедиться, что оптимальное время обработки запроса сервером из рассмотренных – 100 мс, т.к. при больших значениях время получения ответа на запрос может составить более чем 2 секунды, что некомфортно для пользователя.

Результаты проведенного имитационного моделирования для электронного читального зала представлены в таблицах 11 – 14.

В таблице 11 представлены результаты, полученные в результате моделирования при приходе в библиотеку в среднем 1 нового читателя библиотеки за 6 минут.

Таблица 11 – Результаты моделирования электронного читального зала (эксперимент 1)

Библиотекари	Компьютеры	Результат (библиотекари)	Результат (компьютеры)
4	8	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 3
3	8	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 3
2	8	Очередь не накапливается Макс. длина очереди 2	Очередь не накапливается Макс. длина очереди 3
1	8	Очередь 1 Макс. длина очереди 8	Очередь не накапливается Макс. длина очереди 8
4	7	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 3
3	7	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 3
2	7	Очередь не накапливается Макс. длина очереди 2	Очередь не накапливается Макс. длина очереди 5
1	7	Очередь не накапливается Макс. длина очереди 8	Очередь 2 Макс. длина очереди 9
4	6	Очередь не накапливается Макс. длина очереди 1	Очередь 1 Макс. длина очереди 15
3	6	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 6
2	6	Очередь не накапливается Макс. длина очереди 2	Очередь 2 Макс. длина очереди 4
1	6	Очередь не накапливается Макс. длина очереди 5	Очередь 2 Макс. длина очереди 3

Результат эксперимента показывает, что в этом случае оптимальным является вариант с 2 библиотекарями и 7 компьютерами.

Для этого эксперимента также представлено несколько временных характеристик обработки заявок, полученных при оптимальных значениях параметров обработки (2 библиотекаря и 7 компьютеров). На рисунке 32 представлены средневзвешенное значение и среднеквадратичное отклонение времени обработки заявок.

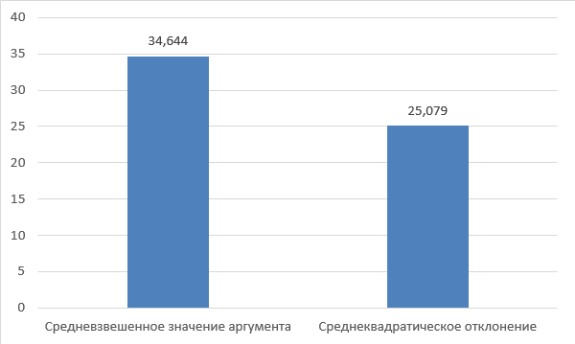


Рисунок 32 – Средневзвешенное значение и среднеквадратичное отклонение

На рисунке 33 представлена гистограмма частот времени обработки заявок.

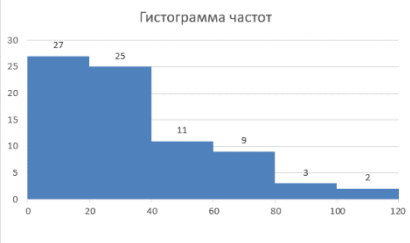


Рисунок 33 – Гистограмма частот

На рисунке 34 представлена эмпирическая функция распределения времени обработки заявок.

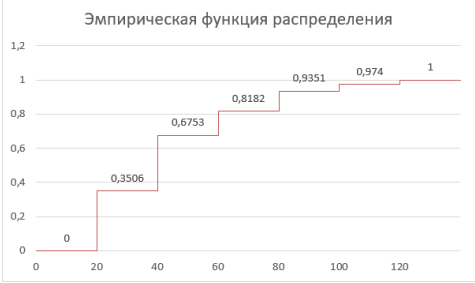


Рисунок 34 – Эмпирическая функция распределения

В таблице 12 представлены результаты моделирования, полученные при приходе в библиотеку в среднем 1 нового читателя библиотеки за 4 минуты.

Таблица 12 – Результаты моделирования электронного читального зала (эксперимент 2)

Библиотекари	Компьютеры	Результат (библиотекари)	Результат (компьютеры)
4	10	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 3
3	10	Очередь не накапливается Макс. длина очереди 2	Очередь не накапливается Макс. длина очереди 3
2	10	Очередь не накапливается Макс. длина очереди 5	Очередь не накапливается Макс. длина очереди 6
1	10	Очередь 2 Макс. длина очереди 17	Очередь 2 Макс. длина очереди 9
4	9	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 7
3	9	Очередь не накапливается Макс. длина очереди 2	Очередь не накапливается Макс. длина очереди 10
2	9	Очередь не накапливается Макс. длина очереди 6	Очередь не накапливается Макс. длина очереди 15
1	9	Очередь 10 Макс. длина очереди 17	Очередь 8 Макс. длина очереди 10
4	8	Очередь не накапливается Макс. длина очереди 1	Очередь 12 Макс. длина очереди 19
3	8	Очередь не накапливается Макс. длина очереди 2	Очередь не накапливается Макс. длина очереди 5
2	8	Очередь не накапливается Макс. длина очереди 6	Очередь 4 Макс. длина очереди 11
1	8	Очередь 2 Макс. длина очереди 17	Очередь 5 Макс. длина очереди 22

Результат эксперимента показывает, что в этом случае оптимальным является вариант с 2 библиотекарями и 9 компьютерами.

В таблице 13 представлены результаты моделирования, полученные при приходе в библиотеку в среднем 1 нового читателя библиотеки за 3 минуты.

Таблица 13 – Результаты моделирования электронного читального зала (эксперимент 3)

Библио-текари	Компью-теры	Результат (библиотекари)	Результат (компьютеры)
4	11	Очередь не накапливается Макс. длина очереди 1	Очередь не накапливается Макс. длина очереди 4
3	11	Очередь не накапливается Макс. длина очереди 2	Очередь не накапливается Макс. длина очереди 6
2	11	Очередь не накапливается Макс. длина очереди 5	Очередь 1 Макс. длина очереди 5
1	11	Очередь не накапливается Макс. длина очереди 14	Очередь 4 Макс. длина очереди 7
4	10	Очередь не накапливается Макс. длина очереди 1	Очередь 34 Макс. длина очереди 34
3	10	Очередь не накапливается Макс. длина очереди 2	Очередь не накапливается Макс. длина очереди 7
2	10	Очередь не накапливается Макс. длина очереди 5	Очередь 10 Макс. длина очереди 10
1	10	Очередь 11 Макс. длина очереди 17	Очередь не накапливается Макс. длина очереди 6
4	9	Очередь не накапливается Макс. длина очереди 1	Очередь 26 Макс. длина очереди 28
3	9	Очередь не накапливается Макс. длина очереди 3	Очередь 53 Макс. длина очереди 55
2	9	Очередь не накапливается Макс. длина очереди 5	Очередь 17 Макс. длина очереди 21
1	9	Очередь 16 Макс. длина очереди 21	Очередь 18 Макс. длина очереди 18

Результат эксперимента показывает, что в этом случае оптимальным является вариант с 2 библиотекарями и 11 компьютерами.

В таблице 14 представлены результаты моделирования, полученные при приходе в библиотеку в среднем 1 нового читателя библиотеки за 2 минуты.

Таблица 14 – Результаты моделирования электронного читального зала (эксперимент 4)

Библио-текари	Компью-теры	Результат (библиотекари)	Результат (компьютеры)
4	18	Очередь не накапливается Макс. длина очереди 5	Очередь не накапливается Макс. длина очереди 16
3	18	Очередь не накапливается Макс. длина очереди 3	Очередь не накапливается Макс. длина очереди 9
2	18	Очередь не накапливается Макс. длина очереди 11	Очередь не накапливается Макс. длина очереди 13
4	17	Очередь не накапливается Макс. длина очереди 3	Очередь не накапливается Макс. длина очереди 10
3	17	Очередь не накапливается Макс. длина очереди 4	Очередь 5 Макс. длина очереди 21
2	17	Очередь не накапливается Макс. длина очереди 14	Очередь 1 Макс. длина очереди 10
4	16	Очередь 3 Макс. длина очереди 5	Очередь 2 Макс. длина очереди 30
3	16	Очередь 1 Макс. длина очереди 4	Очередь 24 Макс. длина очереди 25
2	16	Очередь 2 Макс. длина очереди 14	Очередь не накапливается Макс. длина очереди 20

Результат эксперимента показывает, что в этом случае оптимальным является вариант с 3 библиотекарями и 18 компьютерами.

Результаты имитационного моделирования для добавления книг в систему администраторами представлены в таблицах 15 – 17.

Первый эксперимент производится при следующих значениях среднего интервала времени поступления новых книг для добавления в минутах:

– статья – 1.03;

– сборник статей – 240;
– автореферат диссертации – 45;
– патент – 800;
– монография – 30;
– материал конференции – 1.08;
– сборник трудов конференции – 80;
– диссертация – 31;
– номер журнала – 9.

Всего поступает примерно 5000 книг в неделю.

Результаты моделирования указаны в таблице 15.

Таблица 15 – Результаты моделирования работы администраторов (эксперимент 1)

Администраторы	Результат
10	Время добавления с ожиданием до 30 мин. Очередь до 8.
9	Время добавления с ожиданием до 30 мин. Очередь до 18.
8	Время добавления с ожиданием до 40 мин. Очередь до 26.
7	Очередь переполняется (до 55, 13 остается на конец недели).

Оптимальным является вариант с 9 администраторами.

Для этого эксперимента также представлено несколько временных характеристик обработки заявок, полученных при оптимальных значениях параметров обработки (9 администраторов). На рисунке 35 представлены средневзвешенное значение и среднеквадратичное отклонение времени обработки заявок.

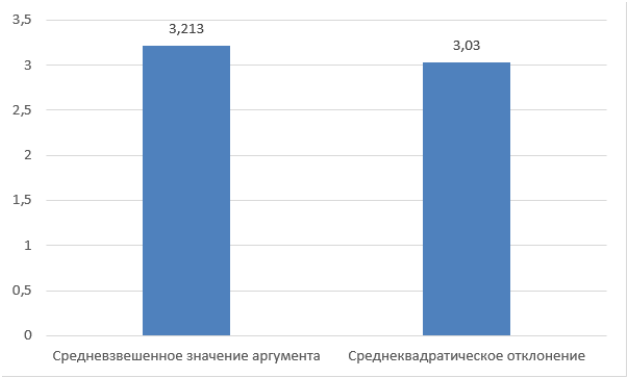


Рисунок 35 – Средневзвешенное значение и среднеквадратичное отклонение

На рисунке 36 представлена гистограмма частот времени обработки заявок.

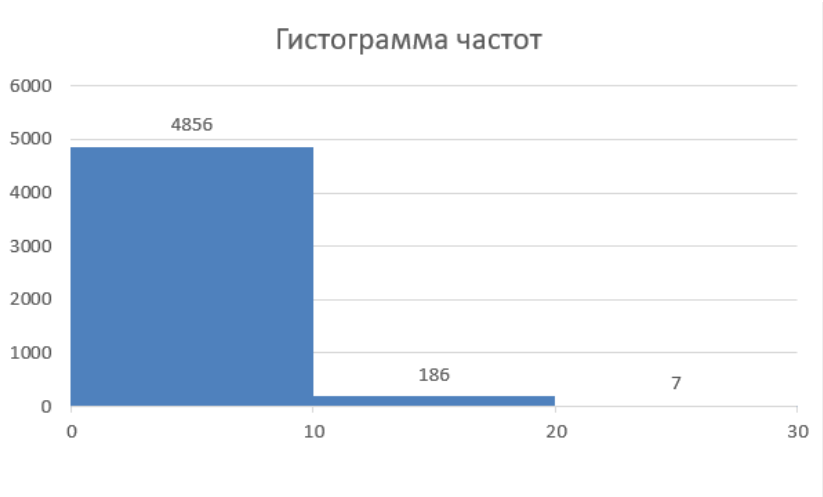


Рисунок 36 – Гистограмма частот

На рисунке 37 представлена эмпирическая функция распределения времени обработки заявок.

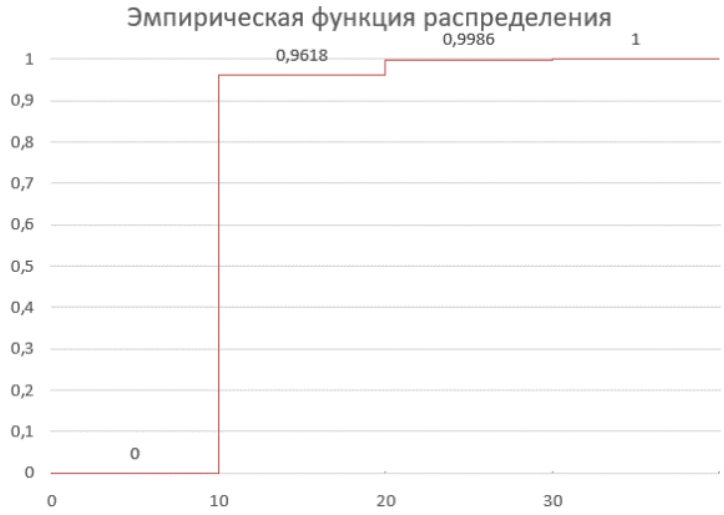


Рисунок 37 – Эмпирическая функция распределения

Второй эксперимент производится при следующих значениях среднего интервала времени поступления новых книг для добавления в минутах:

– статья – 0.69;
– сборник статей – 160;
– автореферат диссертации – 30;
– патент – 533;

– монография – 20;

– материал конференции – 0.72;

– сборник трудов конференции – 53;

– диссертация – 21;

– номер журнала – 6.

Всего поступает примерно 7500 книг в неделю.

Результаты моделирования указаны в таблице 16.

Таблица 16 – Результаты моделирования работы администраторов (эксперимент 2)

Администраторы	Результат
16	Время добавления с ожиданием до 30 мин. Очередь до 8
15	Время добавления с ожиданием до 30 мин. Очередь до 10
14	Время добавления с ожиданием до 30 мин. Очередь до 10, 6 остается в очереди на конец недели
13	Время добавления с ожиданием до 30 мин. Очередь до 25
12	Время добавления с ожиданием до 30 мин. Очередь до 25, 3 остается в очереди на конец недели
11	Время добавления с ожиданием до 30 мин. Очередь до 32
10	Время добавления с ожиданием до 50 мин. Очередь до 84, 1 остается в очереди на конец недели)
9	Очередь явно переполняется (сотни необработанных книг).

Оптимальным является вариант с 13 администраторами.

Третий эксперимент производится при следующих значениях среднего интервала времени поступления новых книг для добавления в минутах:

– статья – 0.52;

– сборник статей – 120;

– автореферат диссертации – 23;

– патент – 400;

– монография – 15;

– материал конференции – 0.54;

– сборник трудов конференции – 40;

– диссертация – 16;

– номер журнала – 5.

Всего поступает примерно 10000 книг в неделю.

Результаты моделирования указаны в таблице 17.

Таблица 17 – Результаты моделирования работы администраторов (эксперимент 3)

Администраторы	Результат
18	Время добавления с ожиданием до 30 мин. Очередь до 16.
17	Время добавления с ожиданием до 30 мин. Очередь до 17.
16	Время добавления с ожиданием до 30 мин. Очередь до 19.
15	Время добавления с ожиданием до 30 мин. Очередь до 30, 3 остается в очереди на конец недели.
14	Время добавления с ожиданием до 30 мин. Очередь до 44.
13	Время добавления с ожиданием до 40 мин. Очередь до 69, 8 остается в очереди на конец недели).
12	Очередь явно переполняется (сотни необработанных книг).

Оптимальным является вариант с 16 администраторами.

Результаты проведенного имитационного моделирования для оцифровки бумажных книг представлены в таблицах 18 – 20.

Первый эксперимент производится при следующих значениях среднего интервала времени поступления новых книг для оцифровки в минутах:

– сборник статей – 2400;

– автореферат диссертации – 624;

– патент – 8000;

– монография – 300;

– сборник трудов конференции – 800;

– диссертация – 312;

– номер журнала – 90.

Всего поступает примерно 50 книг в неделю.

Результаты моделирования указаны в таблице 18.

Таблица 18 – Результаты моделирования работы оцифровщиков (эксперимент 1)

Оцифровщики	Результат
5	Время оцифровки одной книги с учетом ожидания до 360 мин, очередь до 1, максимум 4 занято единовременно.
4	Время оцифровки одной книги с ожиданием до 360 мин, очередь до 1, максимум 4 занято единовременно.
3	Очередь 1-2, 1 книга остается не оцифрованной.
2	Очередь до 5, 2 книги остаются не оцифрованными.

Оптимальным является вариант с 4 оцифровщиками.

Для этого эксперимента также представлено несколько временных характеристик обработки заявок, полученных при оптимальных значениях параметров обработки (4 оцифровщика). На рисунке 38 представлены средневзвешенное значение и среднеквадратичное отклонение времени обработки заявок.

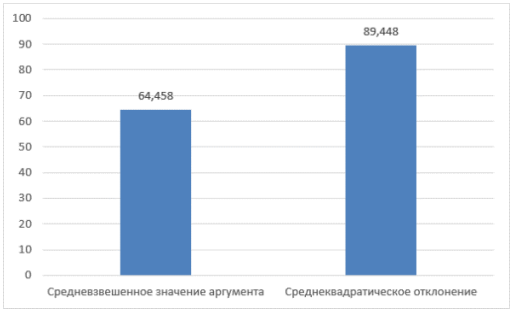


Рисунок 38 – Средневзвешенное значение и среднеквадратичное отклонение

На рисунке 39 – гистограмма частот времени обработки заявок.

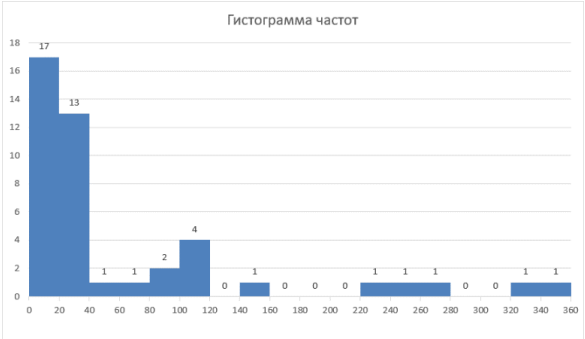


Рисунок 39 – Гистограмма частот

На рисунке 40 представлена эмпирическая функция распределения времени обработки заявок.

Рисунок 40 – Эмпирическая функция распределения

Второй эксперимент производится при следующих значениях среднего интервала времени поступления новых книг для оцифровки в минутах:

– сборник статей – 1600;
– автореферат диссертации – 416;
– ратент – 5333;
– монография – 200;
– сборник трудов конференции – 533;
– диссертация – 208;
– номер журнала – 60.

Всего поступает примерно 75 книг в неделю.

Результаты моделирования указаны в таблице 19.

Таблица 19 – Результаты моделирования работы оцифровщиков (эксперимент 2)

Оцифровщики	Результат
7	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 2
6	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 5
5	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 3
4	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 6, 3 книги остаются не оцифрованными
3	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 11
2	Очередь до 22, 20 книг остаются не оцифрованными

Оптимальным является вариант с 5 оцифровщиками.

Третий эксперимент производится при следующих значениях среднего интервала времени поступления новых книг для оцифровки в минутах:

– сборник статей – 1200;
– автореферат диссертации – 312;
– патент – 4000;
– монография – 150;
– сборник трудов конференции – 400;
– диссертация – 156;
– номер журнала – 45.

Всего поступает примерно 100 книг в неделю.

Результаты моделирования указаны в таблице 20.

Таблица 20 – Результаты моделирования работы оцифровщиков (эксперимент 3)

Оцифровщики	Результат
9	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 1
8	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 9
7	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 3
6	Время оцифровки одной книги с учетом ожидания до 580 мин, очередь до 4
5	Очередь до 4, 3 книги остаются не оцифрованными

Оптимальным является вариант с 6 оцифровщиками.

2.2.1 Проверка адекватности имитационных моделей

Для проверки адекватности разработанных имитационных моделей была проведена их валидация и верификация [57, 58]. В качестве примера приведено проведение валидации и верификации для модели блока администраторов в варианте с добавлением 5000 книг в неделю.

В рамках проведения валидации имитационной модели для оценки адекватности созданной модели был рассчитан ряд значений ожидаемых от модели характеристик систем массового обслуживания.

Интенсивность входящего потока:

$$\lambda = \frac{5000}{2400} = 2{,}083.$$

Коэффициент загрузки системы:

$$\gamma = \frac{\lambda}{N\mu} = \frac{2{,}083}{9 * 0{,}333} = 0{,}694.$$

Среднее число занятых администраторов:

$$\bar{N}_{\text{зан}} = \alpha = \frac{\lambda}{\mu} = \frac{2{,}083}{0{,}333} = 6{,}25.$$

Среднее число свободных администраторов:

$$\bar{N}_{\text{св}} = N - \bar{N}_{\text{зан}} = 9 - 6{,}25 = 2{,}75.$$

Вероятность задействования всех администраторов без возникновения очереди:

$$P_N = P_9 = \frac{\frac{\alpha^9}{9!}}{\sum_{j=0}^{9} \frac{\alpha^j}{j!} + \frac{\alpha^9}{9!} * \frac{\gamma}{1-\gamma}} = \frac{\frac{6{,}25^9}{9!}}{\sum_{j=0}^{9} \frac{6{,}25^j}{j!} + \frac{6{,}25^9}{9!} * \frac{0{,}694}{1-0{,}694}} =$$

$$= \frac{40{,}101}{465{,}068 + 40{,}101 * 2{,}268} = \frac{40{,}101}{556{,}028} = 0{,}072.$$

Средняя длина очереди:

$$\bar{r} = P_N * \frac{\gamma}{(1-\gamma)^2} = 0{,}072 * \frac{0{,}694}{(1-0{,}694)^2} = 0{,}537.$$

Среднее время пребывания заявок в очереди:

$$\bar{T}_{\text{оч}} = \frac{r}{\lambda} = \frac{0{,}537}{2{,}083} = 0{,}258.$$

Среднее число заявок в системе:

$$\bar{N}_{\text{сис}} = \bar{N}_{\text{зан}} + \bar{r} = 6{,}25 + 0{,}537 = 6{,}787.$$

Среднее время пребывания заявок в системе:

$$\bar{T}_{\text{сис}} = \frac{\bar{N}_{\text{сис}}}{\lambda} = \frac{6{,}787}{2{,}083} = 3{,}258.$$

Сравним некоторые из полученных теоретических значений показателей с практическими значениями, полученными в процессе моделирования (таблица 21).

Таблица 21 – Сравнение теоретических и практических значений характеристик модели

Показатель	Расчетное значение	Модельное значение	Расхождение
Коэффициент загрузки системы	0,694	0,701	1%
Среднее число свободных администраторов	2,75	2,691	2,19%
Среднее время пребывания требований в системе	3,258	3,213	1,4%

Отклонения незначительны (в пределах 10%), что свидетельствует о корректности работы модели.

Далее проведем верификацию разработанной модели. Нет необходимости проводить верификацию всей модели, достаточно проверить те ее места, где легко допустить ошибку. В данной модели, таким местом является заключительная часть модели, где заявки разных типов отделяют друг от друга, чтобы можно было вести статистику не только по всем типам книг в целом, но и для каждого из них по отдельности. Поэтому нужно провести корректность обработки заявок для каждого типа. Для проверки корректности работы модели используется режим трассировки в GPSS Studio. Результаты проверок представлены в приложении В.

Как можно увидеть, каждая из заявок была обработана в точном соответствии со своим типом. Следовательно, модель работает корректно для всех типов заявок.

Таким образом, результаты валидации и верификации разработанной имитационной модели подтвердили ее адекватность.

Аналогичным образом валидация и верификация были произведены и для остальных разработанных имитационных моделей. Для них также была подтверждена их адекватность.

2.3 Анализ требований

2.3.1 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 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, а также мочь удалять из системы любые книги или отзывы к ним при обнаружении нарушений и блокировать пользователей, допустивших нарушения.

В результате анализа требований задания была создана диаграмма вариантов использования (рисунок 41).

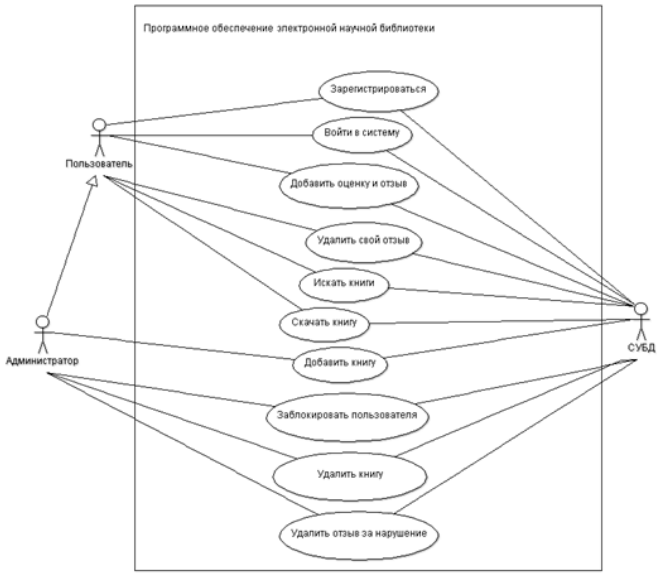


Рисунок 41 – Диаграмма вариантов использования электронной научной библиотеки

Составим спецификации вариантов использования «Зарегистрироваться» (рисунок 42), «Войти в систему» (рисунок 43), «Искать книги» (рисунок 44), «Добавить оценку и отзыв» (рисунок 45).

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 42 – 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 существующего 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 43 – 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: 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 44 – 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: 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 45 – Спецификация варианта использования «Добавить оценку и отзыв»

Рассмотрим некоторые варианты использования поподробнее. Для варианта использования «Зарегистрироваться» составлена диаграмма деятельности, представленная на рисунке 46.

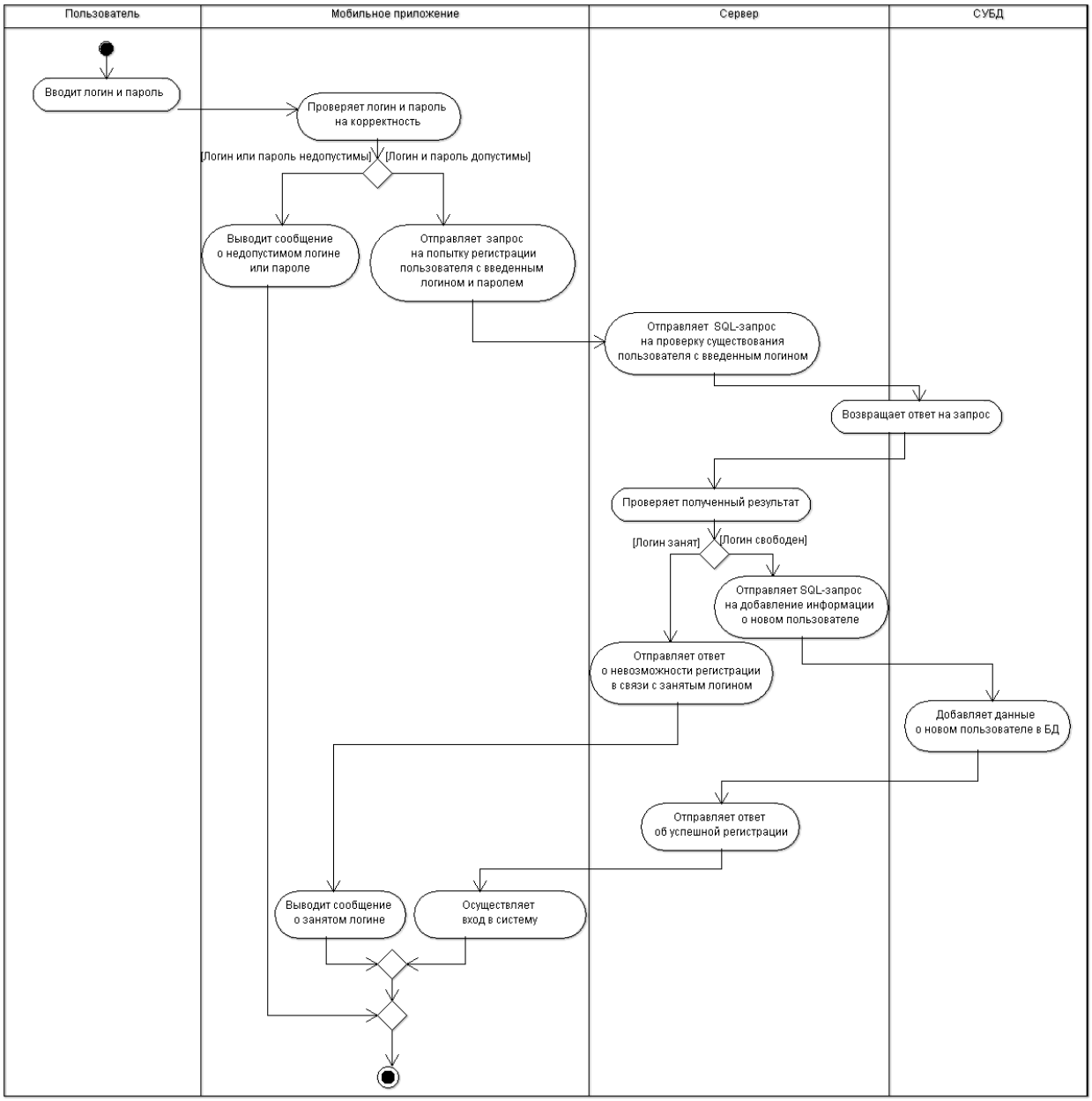


Рисунок 46– Диаграмма деятельности варианта использования «Зарегистрироваться»

Для варианта использования «Искать книги» была составлена диаграмма последовательности, представленная на рисунке 47.

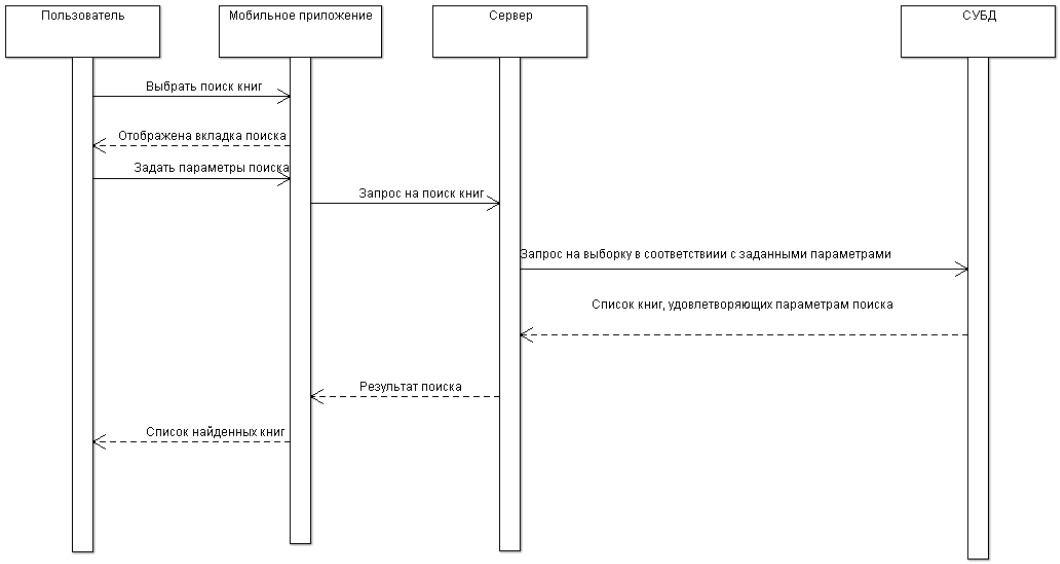


Рисунок 47 – Диаграмма последовательности варианта использования «Искать книги»

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.

2.3.2 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 часть должна работать под управлением операционной системы Windows 10 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 7.0 и выше.

Для проведения онтологического моделирования использовалась среда Protege, имитационного – GPSS Studio. Планирование проекта производилась с использованием системы управления проектами Microsoft Project, для построения диаграмм UML использовалось приложение ArgoUML.

2.4 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 48. 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.

Рисунок 48 – Страница входа

Страница регистрации представлена на рисунке 49. 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.

Рисунок 49 – Страница регистрации

Страница поиска представлена на рисунке 50. 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).

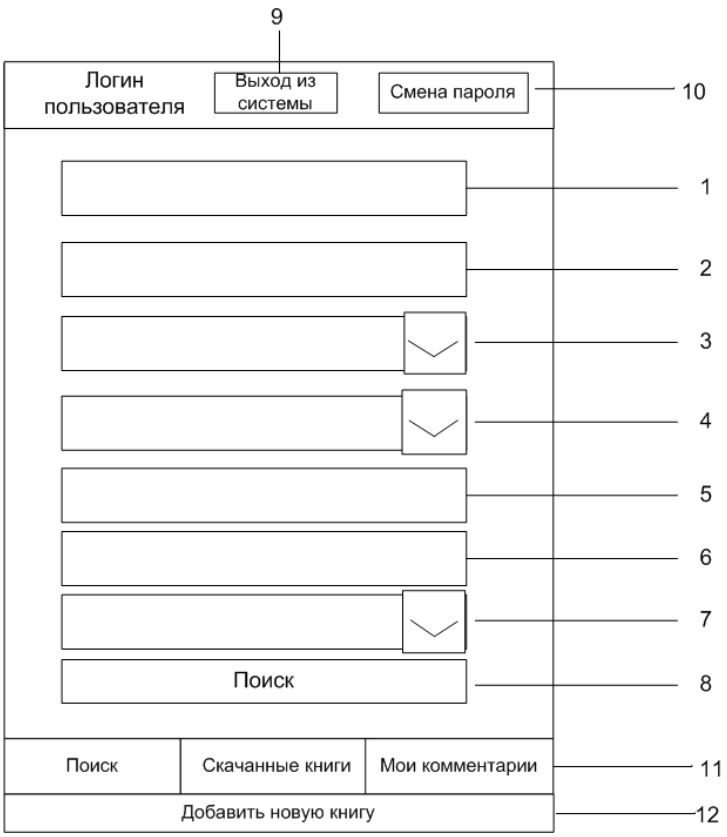


Рисунок 50 – Страница поиска

При нажатии на кнопку «Поиск» произойдет переход на страницу результатов поиска, представленную на рисунке 51. 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.

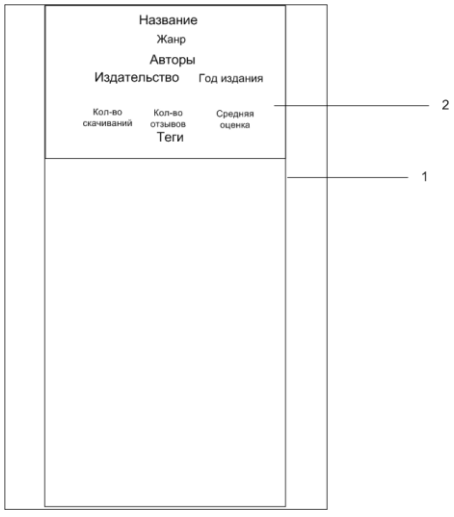


Рисунок 51 – Страница результатов поиска

На странице списка скачанных книг, которая представлена на рисунке 52, 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).

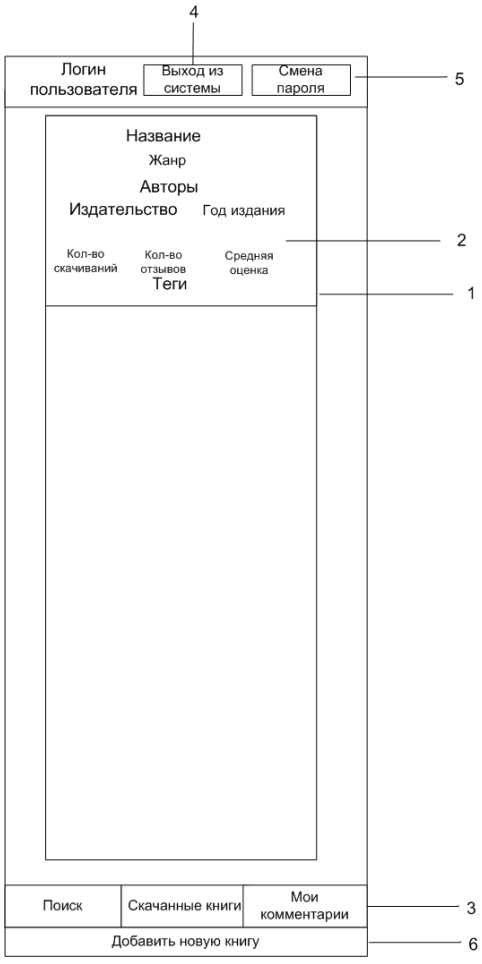


Рисунок 52 –Страница списка скачанных книг

Страница списка комментариев пользователя представлена на рисунке 53. 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).

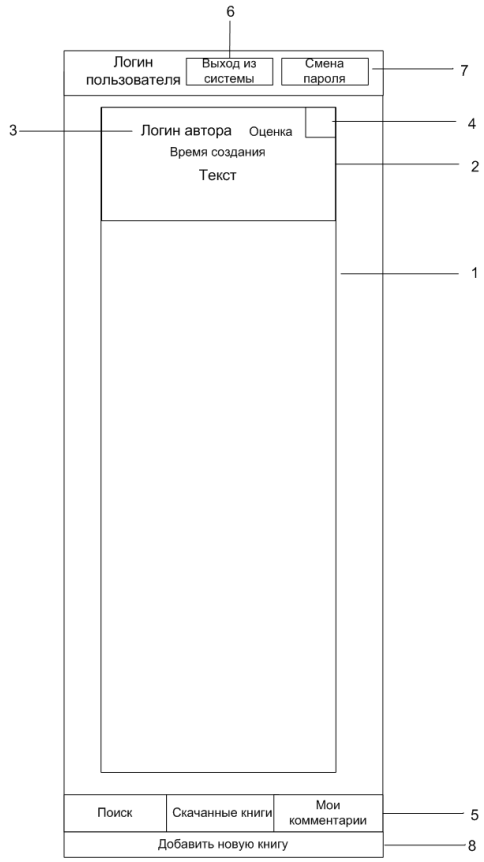


Рисунок 53 – Страница комментариев пользователя

Страница добавления новой книги представлена на рисунке 54. 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 дубликате. В противном случае информация о книге и ее файл сохраняются на сервере.

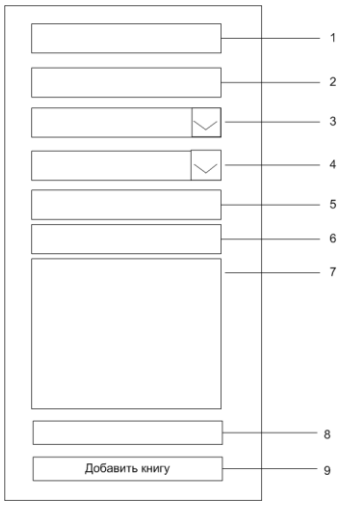


Рисунок 54 – Страница добавления новой книги

Страница смены пароля представлена на рисунке 55. 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 условие не выполняется, будет выведено сообщение об ошибке. В противном случае будут произведены смена пароля текущего пользователя и последующий возврат пользователя на предыдущую страницу.

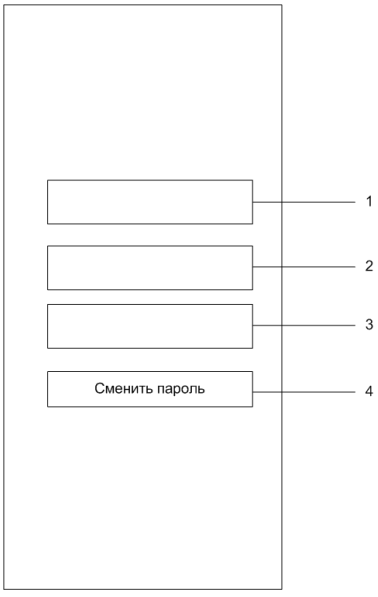


Рисунок 55 – Страница смены пароля

Если пользователь нажмет на любой из элементов в любом списке книг, откроется страница с информацией о выбранной книге, представленная на рисунке 56.

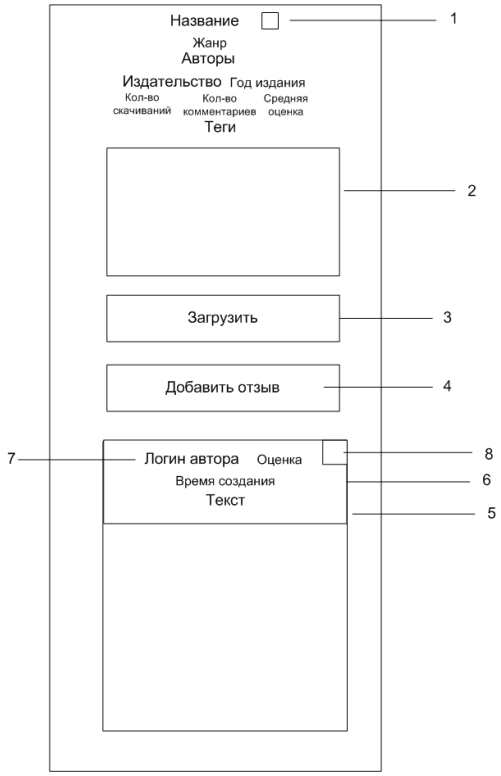


Рисунок 56 – Страница «Информация о книге»

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 57), на которой имеются следующие элементы: 1 – поле ввода значения оценки; 2 – поле ввода нового пароля пользователя; 3 – кнопка «Отправить».

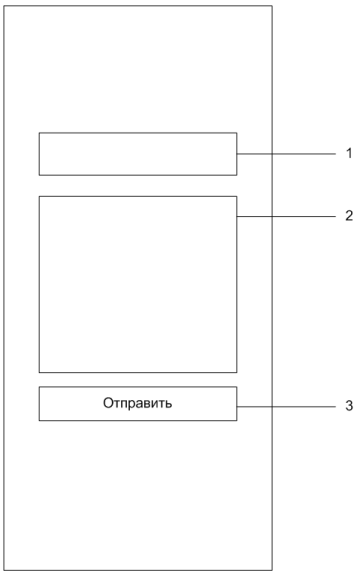


Рисунок 57 – Страница добавления отзыва

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 58), на которой представлены основные сведения о нем.

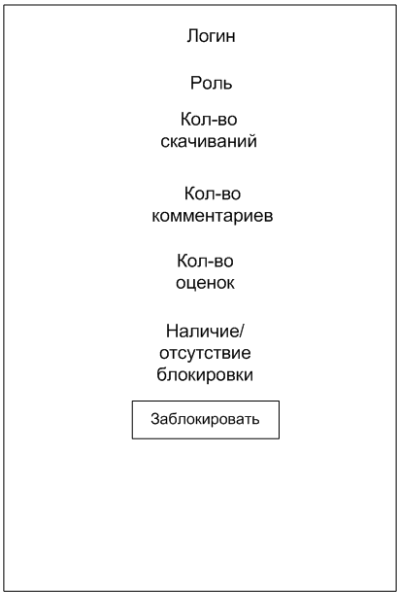


Рисунок 58 – Страница «Информация о пользователе»

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 59). Здесь имеются следующие элементы: 1 – поле ввода срока блокировки в днях; 2 – кнопка «Заблокировать».

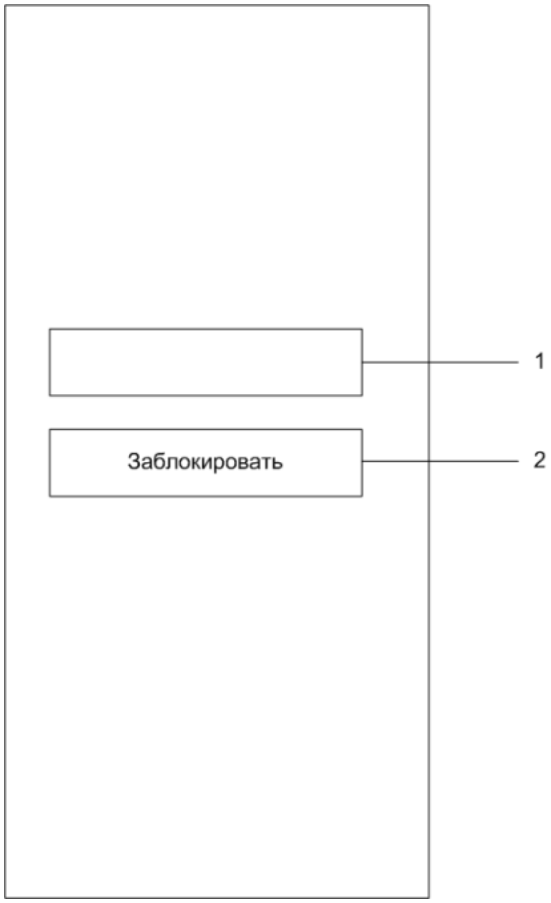


Рисунок 59 – Страница «Блокировка пользователя»

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

2.5 Проектирование базы данных

На основе анализа предметной области и функциональных требований была построена концептуальная модель базы данных [59]. Она представлена в виде диаграммы «сущность-связь» (рисунок 60).

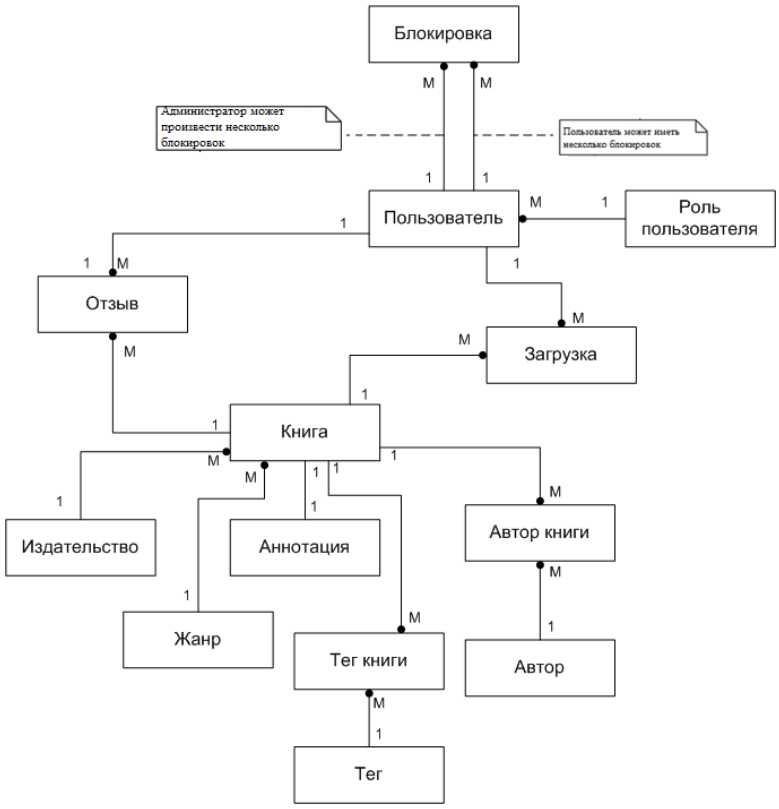


Рисунок 60 – Диаграмма «сущность-связь»

Сущности концептуальной модели данных описаны в таблице 22.

Таблица 22 – 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
6	Автор книги	Связь между авторами и книгами	Ассоциатив-ная	100000000	Обновляется при добавлении новой книги

Продолжение таблицы 22

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
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 23 представлено описание связей между сущностями предметной области.

Таблица 23 – Описание связей между сущностями предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мощ-ность	Обязатель-ность	Ограни-чения
1	Пользователь	Загрузка	Каждый пользователь может производить множество загрузок/ Каждая загрузка производится одним пользователем	1:M	Необязательная (пользователь может не иметь загруженных книг)	При повторной загрузке книги поль-зователем новая запись не создается
2	Роль пользователя	Пользователь	Каждую роль может иметь множество пользователей/ Каждый пользователь может иметь только одну роль	1:M	Необязательная (роль может не относится ни к одному пользователю)	
3	Книга	Загрузка	Каждая книга может быть загружена множество раз/ Каждая загрузка производится для одной книги	1:M	Необязательная (книга может не иметь загрузок)	При пов-торной заг-рузке книги пользовате-лем новая запись не создается

Продолжение таблицы 23

№	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	
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 61.

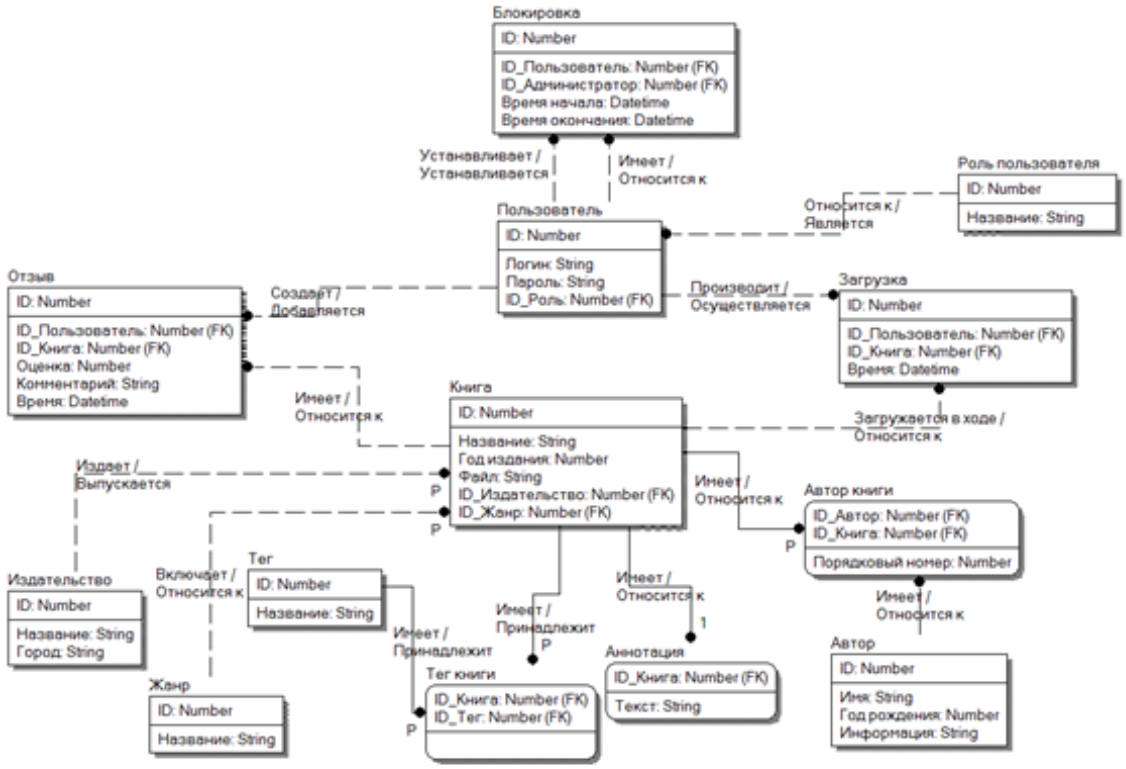


Рисунок 61 – Логическая модель базы данных

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.

2.6 Разработка улучшенного алгоритма поиска книг в электронной библиотеке

Для электронной библиотеки крайне важна оптимальная организация процесса поиска в ней книг. От нее очень сильно зависят как скорость выполнения поиска, так и релевантность его результатов. При этом стоит заметить, что эти две характеристики могут противоречить друг другу.

Наиболее быстрым является обычный текстовый поиск, который просто проверяет названия книг на соответствие введенному поисковому запросу и отбирает те из них, которые полностью ему удовлетворяют.

Однако такой поиск подходит далеко не всегда, так как поисковый запрос может содержать ошибки и неточности, из-за чего нужные книги могут быть не найдены. Чтобы решить эту проблему, используют алгоритмы нечеткого поиска, которые позволяют находить не только результаты, полностью соответствующие искомому запросу, но и результаты, относительно которых запрос содержит одну или несколько ошибок, хотя и ценой меньшей, по сравнению с точным поиском, скорости. Именно такой поиск и было решено реализовать в разрабатываемой электронной библиотеке.

Существует множество известных алгоритмов нечеткого поиска, каждый из которых имеет свои достоинства и недостатки [60 – 62]. К ним относятся:

– алгоритм расширения выборки;
– метод N-грамм;
– метод хеширования по сигнатуре;
– метод BK-деревьев;
– и т.д.

Рассмотрим вышеперечисленные алгоритмы поподробнее.

Метод N-грамм основан на гипотезе о том, что слова, совпадающие с учетом нескольких ошибок, должны иметь общую подстроку длины N. Такие подстроки называются N-граммами. При индексации словаря каждое слово в нем делится на N-граммы, и для всех N-грамм сохраняются списки содержащих их слов. Во время поиска запрос также разбивают на N-граммы, для каждой из которых получают список слов, в которые она входит. Принцип работы алгоритма наглядно показан на рисунке 62.

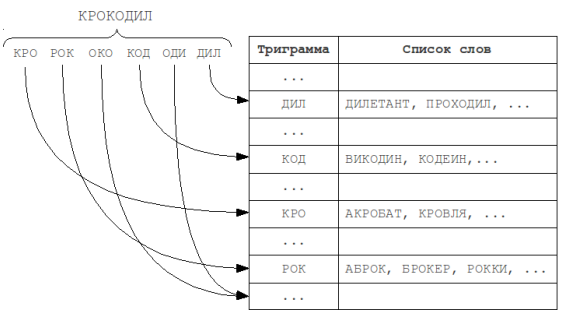
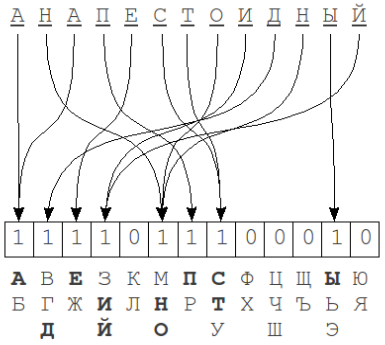


Рисунок 62 – Принцип работы метода N-грамм

На практике обычно используют триграммы (подстроки из 3 символов), так как при большом N алгоритм становится ненадежным.

Главным недостатком метода является то, что у запроса малой длины одна из букв может входить сразу во все его N-граммы, и всего одной ошибки в этой букве будет достаточно, чтобы запрос не дал нужного результата. Например, в слове «водка» буква «д» входит во все триграммы, и запрос «вотка» при использовании триграмм не найдет требуемого ответа. Вероятность того, что поиск закончится неудачей, тем выше, чем меньше длина запроса и больше количество ошибок в нем.

Метод хеширования по сигнатуре базируется на сопоставлении слов с последовательностями битов, называемыми хешами. Каждому из битов хеша сопоставляется группа символов в алфавите. При этом *i*-й бит хеша слова означает наличие (1) или отсутствие (0) в данном слове символов *i*-ой группы алфавита. Принцип построения хеша наглядно показан на рисунке 63.



Хеш	Список слов
0000000000000	-
...	
1111011100001	АВТОПРЕДПРИЯТИЕ, БЕЗОТРАДНАЯ, ВЕТЕРИНАРИЯ, ...
1111011100010	ПРЕВРАТНЫЙ, БЕЗРАССУДНЫЙ, АНАПЕСТОИДНЫЙ, ...
1111011100011	СОРИЕНТИРОВАТЬСЯ, БЕСПРЕПЯТСТВЕННЫЙ, ...
...	
1111111111111	ЛЕГКОИСЧЕРПЫВАЮЩИХСЯ, ВЫСОКОРАЗРЕШАЮЩИХ, ...

Рисунок 63 – Принцип построения хеша для метода хеширования по сигнатуре

При индексации словаря для каждого слова в нем вычисляется его хеш. Затем для каждого полученного в процессе индексации хеша сохраняется список соответствующих ему слов. Во время поиска вычисляется хеш,

соответствующий запросу. Затем в индексе ищутся все хранящиеся в нем хеши, отличающиеся от хеша запроса не более чем k битами, после чего для каждого из найденных хешей берется список соответствующих ему слов.

При удалении символа это либо не повлияет на хеш, либо изменит один из его битов на 0. Аналогично, при вставке нового символа в слово либо один бит в его хеше станет равным 1, либо ничего не поменяется. При замене символа хеш может остаться неизменным, изменить один бит или изменить 2 бита. При перестановке ничего не произойдет, так как при вычислении хеша порядок символов в слове не учитывается.

Главный недостаток алгоритма заключен в том, что из-за возможности изменения сразу 2 бит при замене одного символа найти все слова, получаемые из исходного при k ошибках, путем проверки всех хешей, отличающихся от хеша запроса на k бит, не получится из-за пропуска значительного числа слов с k заменами. Исправить это можно только проверкой также и хешей с большим числом различий, что потратит дополнительное время и найдет также множество ненужных результатов с числом ошибок, большим, чем k. При этом замена символа в слове будет изменять в хеше сразу 2 бита тем чаще, чем больше длина хеша. Кроме того, алгоритм хеширования по сигнатуре неприменим для префиксного поиска.

Метод BK-деревьев (деревья Баркхарда-Келлера) использует метрические деревья, построенные на основе свойства метрики отвечать неравенству треугольника. Данное свойство позволяет метрикам образовывать евклидовые или неевклидовые метрические пространства произвольной размерности. Используя его, можно построить структуру данных для поиска в метрическом пространстве, которой и являются BK-деревья. Некоторые метрики позволяют вычислять расстояние с ограничением, что дает возможность несколько ускорить поиск. Однако данный алгоритм в любом случае является на фоне аналогов неэффективным как по времени, так и по памяти.

В качестве основы для разрабатываемого модифицированного алгоритма поиска книг был взят алгоритм расширения выборки. Он на

основе исходного запроса создает набор запросов со всеми возможными ошибками, после чего для полученных «ошибочных» запросов производится точный поиск. Таким образом, задача о нечетком поиске оказывается сведена к обычному текстовому поиску. Принцип работы алгоритма наглядно показан на рисунке 64.

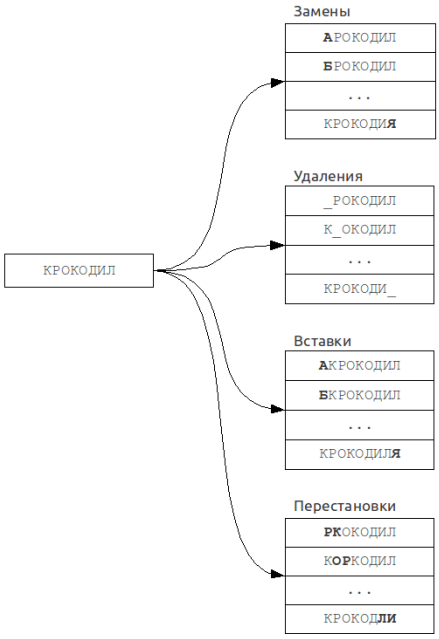


Рисунок 64 – Принцип работы алгоритма расширения выборки

Данный алгоритм решено взять за основу предлагаемой реализации поиска, так как он точнее и экономичнее по памяти большинства аналогичных алгоритмов, таких как хеширование по сигнатуре или метод N-грамм, а также довольно легок в модификации.

Главным его недостатком является низкая скорость поиска, поэтому разработанная модификация алгоритма призвана обеспечить его ускорение. Для этого из числа возможных вариантов ошибок поисковых запросов проверяются только наиболее вероятные. Маловероятные ошибки же игнорируются, что дает существенное увеличение скорости поиска ценой приемлемых потерь в точности. Вероятность ошибок определяется на основе расположения символов на клавиатуре, а также типичных орфографических ошибок русского языка («а» вместо «о», «с» вместо «з» и так далее).

Диаграмма деятельности разработанного модифицированного алгоритма поиска представлена на рисунке 65.

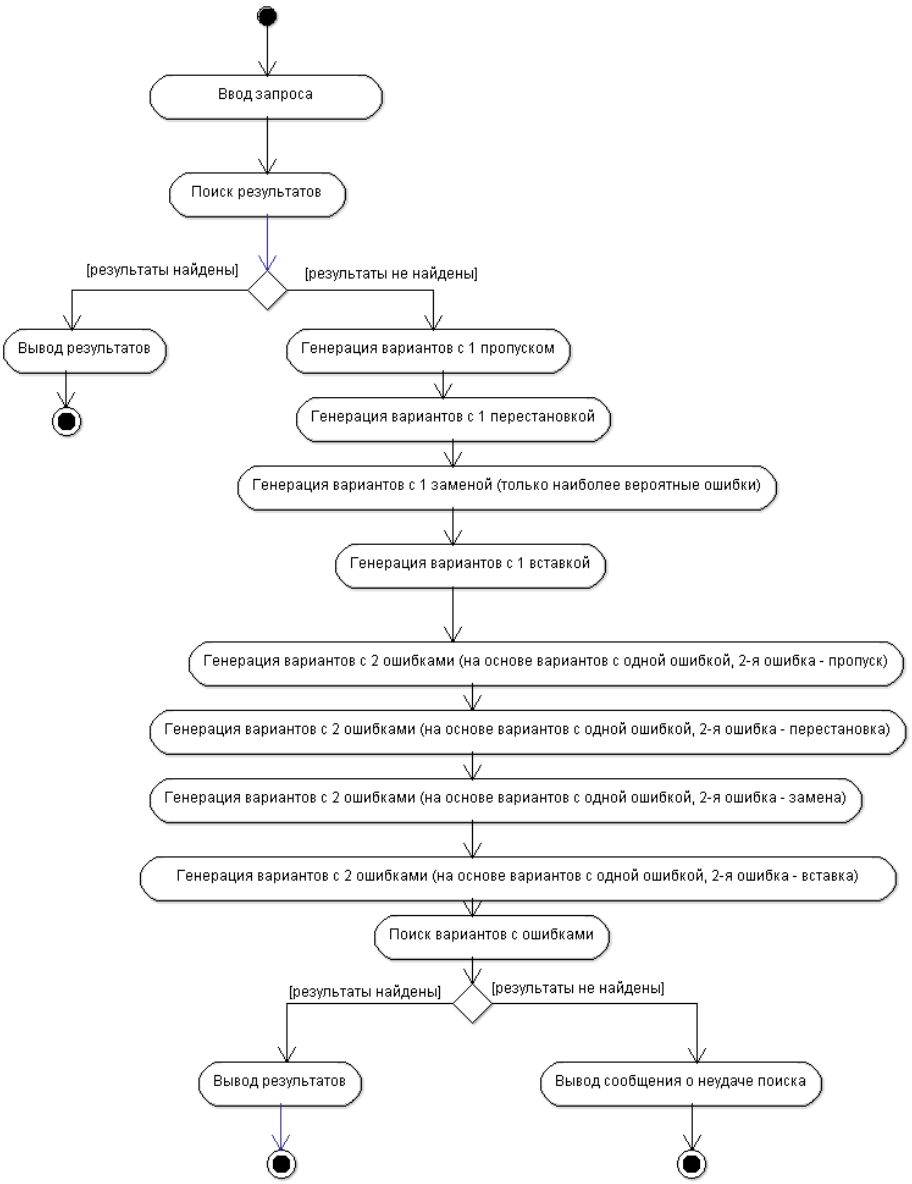


Рисунок 65 – Диаграмма деятельности модифицированного алгоритма

Таким образом, предложен модифицированный алгоритм поиска книг в электронной библиотеке, отличающийся от алгоритма расширения выборки, на основе которого он создан, большей скоростью поиска при сохранении приемлемой надежности.

Выводы

Проведены формирование и анализ функциональных и нефункциональных требований для разрабатываемой электронной библиотеки, проведено онтологическое моделирование рассматриваемой предметной области.

Для оптимизации параметров функционирования разрабатываемой электронной библиотеки было проведено имитационное моделирование в среде GPSS Studio. Для разработанных имитационных моделей произведена оценка адекватности, подтвердившая корректность их работы.

Созданы концептуальная и логическая модели базы данных разрабатываемой электронной библиотеки.

Разработан алгоритм поиска книг для проектируемой электронной библиотеки и определена выгода, которую должно позволить достичь его использование.

3 Реализация электронной научной библиотеки

3.1 Разработка прототипа электронной научной библиотеки

3.1.1 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 66.

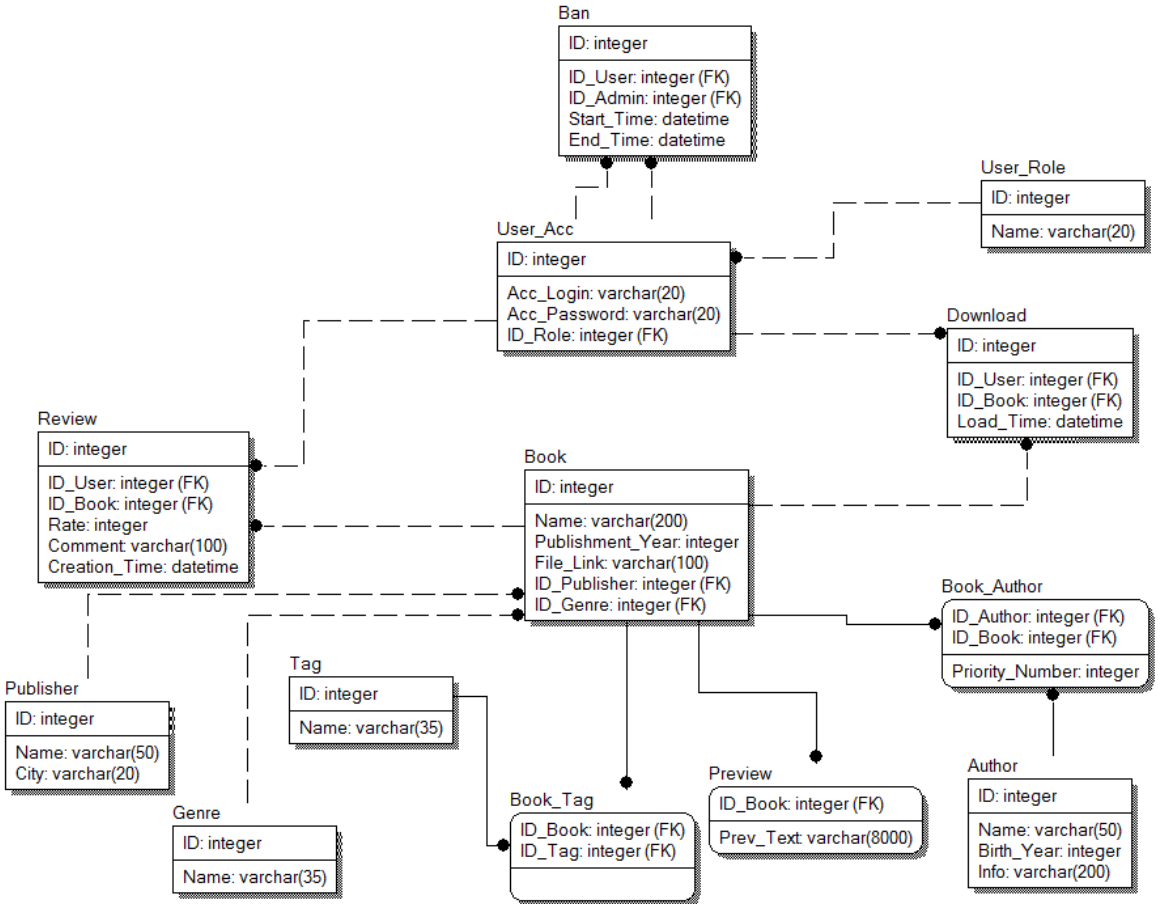


Рисунок 66 – Физическая модель базы данных

На основе составленной физической модели БД был сгенерирован скрипт на языке SQL для создания базы данных [63]. Его код представлен в приложении Г. 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.

3.1.2 Разработка серверной части

Разработка серверной части приложения производилась в среде программирования Microsoft Visual Studio 2019 на языке C# с применением технологии Web API [64, 65]. Для взаимодействия с базой данных использовалась технология Entity Framework [66].

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 67. Классы были автоматически сгенерированы средой программирования Visual Studio с использованием технологии Entity Framework. Генерация производилась на основе разработанной ранее базы данных (подход Database First) [67, 68].

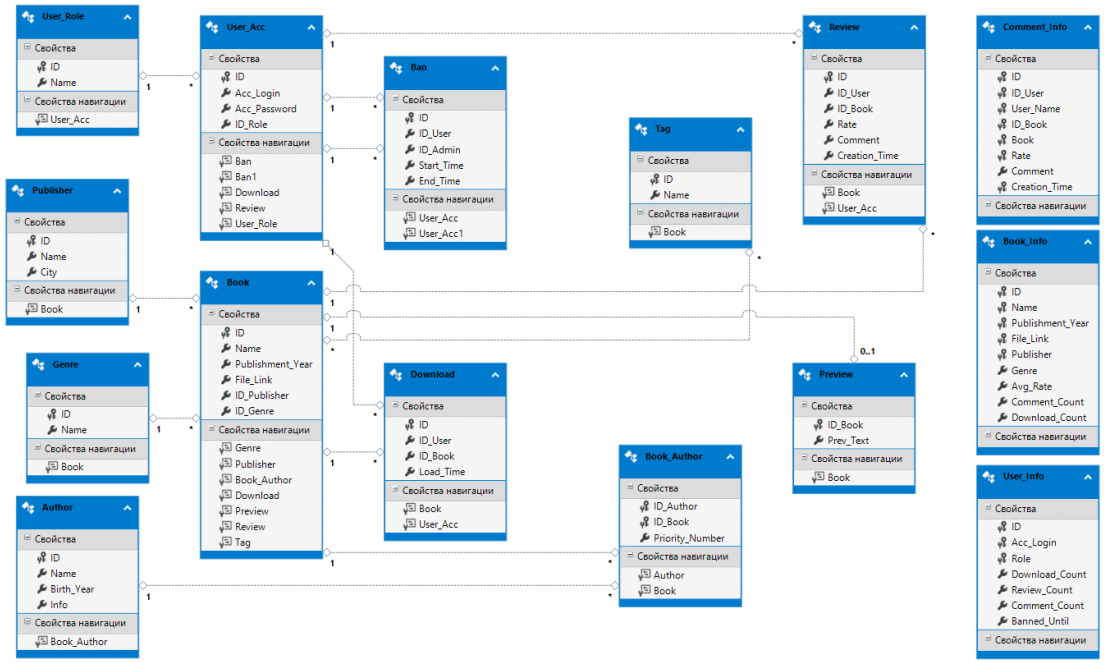


Рисунок 67 – Диаграмма классов серверной части (слой модели)

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 облегчить выборку нужных данных из базы.

Диаграмма классов слоя контроллеров представлена на рисунке 68.

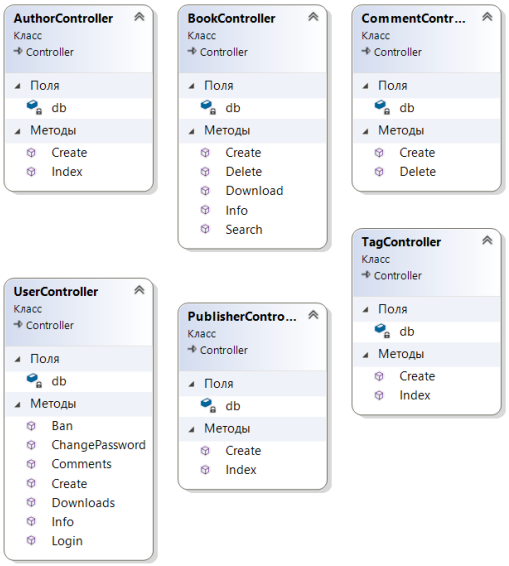


Рисунок 68 – Диаграмма классов серверной части (слой контроллеров)

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 24.

Таблица 24 – Описание классов слоя контроллеров

Класс	Назначение
AuthorController	Отвечает за обработку запросов на добавление в систему новых авторов и получение списка имеющихся
BookController	Отвечает за обработку запросов на добавление в систему новых книг и удаление уже имеющихся, на выборку книг по некоторым критериям, 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 удаление имеющихся
PublisherController	Отвечает за обработку запросов на добавление в систему новых издательств и получение списка имеющихся

Продолжение таблицы 24

Класс	Назначение
TagController	Отвечает за обработку запросов на добавление в систему новых тегов и получение списка имеющихся
UserController	Отвесает на обработку запросов на регистрацию нового пользователя, на вход в систему зарегистрированного, на смену пароля пользователя, на получение всех отзывов или загрузок пользователя, на получение сведений о пользователе и на его блокировку.

Листинг серверной части представлен в приложении Д.

3.1.3 Разработка клиентской части

Разработка мобильного приложения производилась на языке Java с использованием среды программирования Android Studio. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 69.

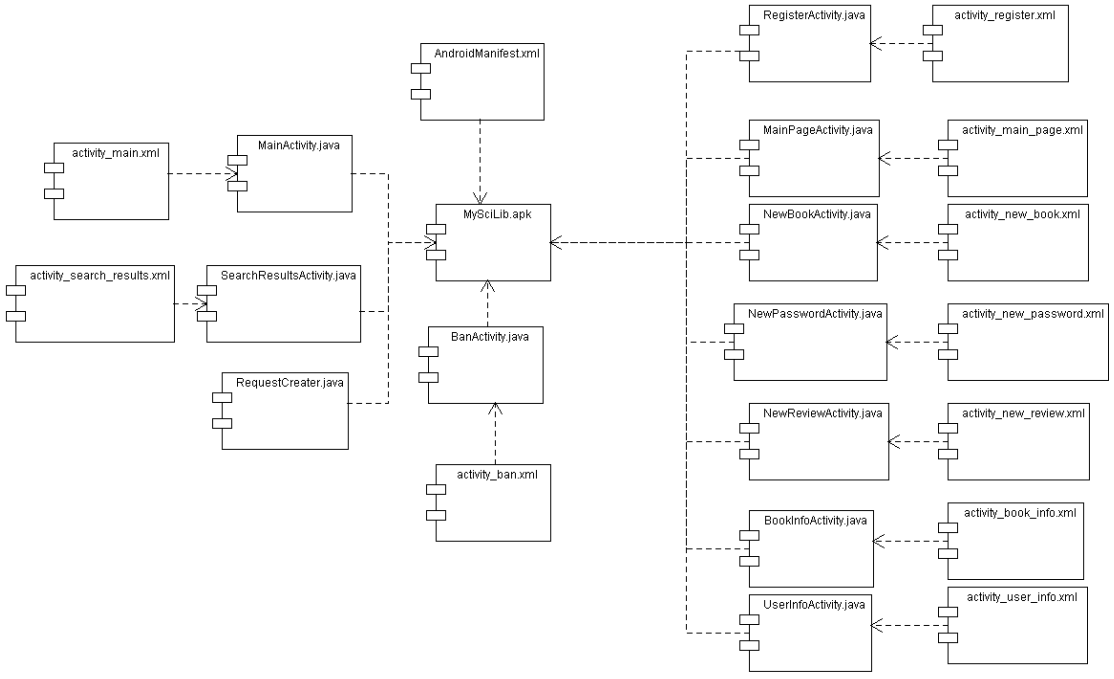


Рисунок 69 – Диаграмма компонентов клиентской части

Описание компонентов представлено в таблице 25.

Таблица 25 – Компоненты клиентской части

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 пользова-теле.
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.
RequestCreater.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 70.

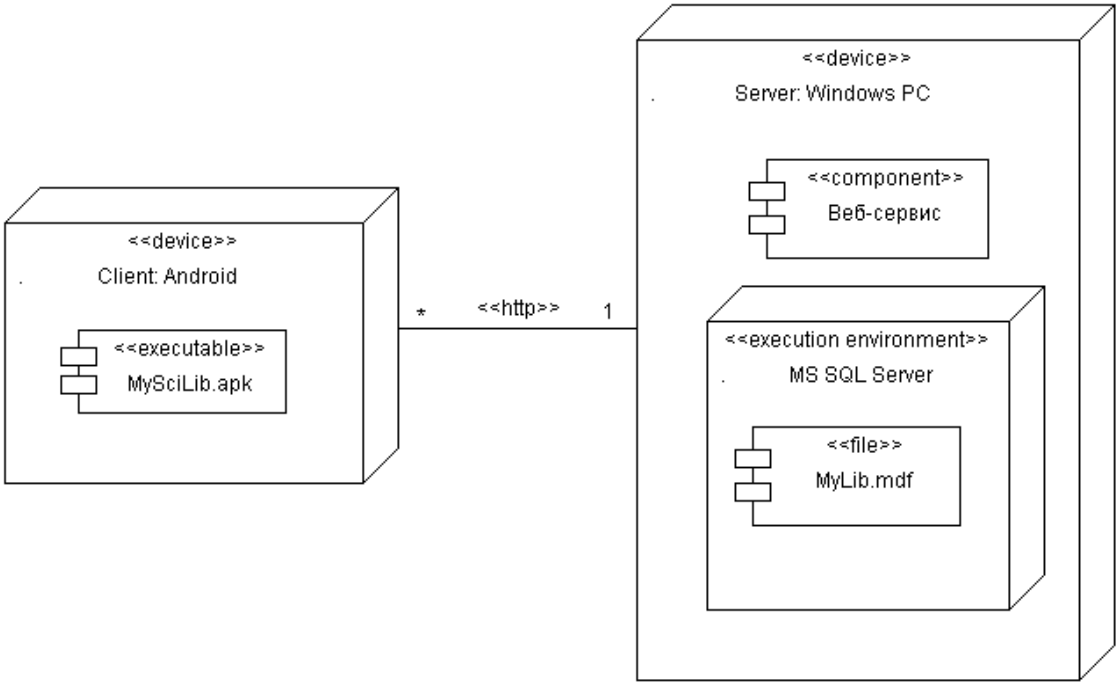


Рисунок 70 – Диаграмма развертывания электронной библиотеки

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 представлен в приложении Е.

3.1.4 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 26, скриншоты тестов находятся в приложении Ж.

Таблица 26 – 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.
2	Попытка входа в систему пользователя с логином someuser и паролем password12	GET	login = some-user, password= password 12	Тест пройден. Пользователь успешно авторизован. Код ответа HTTP – 200.
3	Получение сведений о пользователе с идентификатором 1	GET	id = 1	Тест пройден. Информация о поль-зователе успешно получена. Код ответа HTTP – 200.
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.
5	Получение списка комментариев, оставленных пользователем с идентификатором 1	GET	id=5	Тест пройден. Список комментариев, оставленных поль-зователем с идентификатором 1, успешно получен. Код ответа HTTP – 200.
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.
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.
8	Добавление отзыва от пользователя с идентификатором 1 к книге с идентификатором 1.	POST	userID=1, bookID=1, comment=Поз-навательно, rate=5	Тест пройден. Отзыв успешно добавлен в систему. Код ответа HTTP – 201.
9	Удаление отзыва с идентификатором 23	DELE-TE	id = 23	Тест пройден. Отзыв успешно удален. Код ответа HTTP – 200.
10	Добавление инфор-мации о загрузке пользователем книги с идентификатором 1.	POST	bookID=1, userID=1	Тест пройден. Информация о загрузке успешно добавлена. Код ответа HTTP – 201.

Продолжение таблицы 26

№	Описание теста	Запрос		Результат
		Метод	Параметры	
11	Получение списка издательств	GET		Тест пройден. Получен список издательств, сохраненных в системе. Код ответа HTTP – 200.
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.
13	Получение списка авторов	GET		Тест пройден. Получен список авторов, сохраненных в системе. Код ответа HTTP – 200.
14	Добавление в систему автора по имени Анастасия Новицкая	POST	name=Анастасия Новицкая	Тест пройден. Автор успешно добавлен. Код ответа HTTP – 201.
15	Получение списка тегов	GET		Тест пройден. Получен список тегов, сохраненных в системе. Код ответа HTTP – 200.
16	Добавление тега «C#»	POST	name=C#	Тест пройден. Тег успешно добавлен. Код ответа HTTP – 201.
17	Получение полной информации о книге с идентификатором 1	GET	id = 1	Тест пройден. Информация о книге успешно получена. Код ответа HTTP – 200.
18	Удаление книги с идентификатором 1	DELETE	id =1	Тест пройден. Книга успешно удалена. Код ответа HTTP – 200.
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.

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

3.1.5 Тестирование клиентской части приложения

3.1.5.1. Функциональное тестирование

Для проверки корректности работы клиентской части приложения было проведено функциональное тестирование. Его результаты представлены в таблице 27, скриншоты тестов находятся в приложении И.

Таблица 27 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Ввести валидные данные для регистрации. Логин: mylogin. Пароль: qwerty1234.	Тест пройден. Пользователь зарегистрирован в системе.
	Ввести данные с логином существующего в системе пользователя. Логин: mylogin.	Тест пройден. Выведено уведомление о том, что логин уже занят.
	Попытаться задать логин и пароль с недопустимыми символами. Логин: логин. Пароль: пароль.	Тест пройден. Выведено уведомление о вводе недопустимых данных.
	Попытаться задать логин и пароль недопустимой длины. Логин: m. Пароль: qwe1.	Тест пройден. Выведено уведомление о вводе недопустимых данных.
	Попытаться задать пароль без букв. Логин: mylogin1 Пароль: qwertyuiop	Тест пройден. Выведено уведомление о вводе недопустимых данных.
Авторизоваться	Ввести валидные данные. Логин: mylogin. Пароль: qwerty1234..	Тест пройден. Авторизация выполнена успешно.
	Ввести неверные данные. Логин: mylogin. Пароль: qwerty123456.	Тест пройден. Выведено уведомление о неверном логине или пароле.
	Ввести данные заблокированного пользователя.	Тест пройден. Выведено сообщение о блокировке.
Искать книги	Ввести поисковые данные: Название: Проектирование программ и программирование на С++. Авторы: Ольга Леонидовна Викентьева, Ольга Андреевна Полякова . Жанр: матеиал 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».

Продолжение таблицы 27

Вариант использования	Тест и параметры	Результат
Добавить оценку и отзыв	Поставить оценку с отзывом. Оценка: 5. Отзыв: интересно .	Тест пройдён. Оценка сохранена в системе, отзыв добавлен в список отзывов о книге.
	Оставить оценку без отзыва. Оценка: 4.	Тест пройден. Оценка сохранена в системе.
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.
	Попытаться добавить книгу без задания названия, авторов и тегов.	Тест пройден. Выдано сообщение о неполных данных.
Добавить книгу	Попытаться добавить книгу без указания ее файла.	Тест пройден. Выдано сообщение об отсутствии файла книги.
Скачать книгу	Загрузить через приложение файл книги, добавленной в предыдущем примере.	Тест пройден. Книга успешно открыта.
Удалить отзыв	Удалить отзыв, оставленный в ходе предшествующих тестов.	Тест пройден. Отзыв успешно удален.
Удалить книгу	Удалить книгу, добавленную в ходе предшествующих тестов.	Тест пройден. Книга успешно удалена.
Заблокировать пользователя	Заблокировать пользователя mylogin на 3 дня.	Тест пройден. Пользователь успешно заблокирован.

Функциональное тестирование показало корректность работы клиентской части приложения.

3.1.5.2. Модульное тестирование

Модульные тесты – автоматизированные тесты, позволяющие оценить корректность работы некоторой автономной единицы программного кода (обычно подпрограммы) путем сравнения фактического результата работы с ожидаемым [69 – 71]. Основное достоинство модульных тестов – то, 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 соотвественно.

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("l"));
    }
    @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 71.

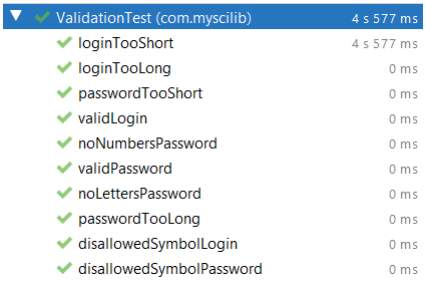


Рисунок 71 – Результаты модульного тестирования

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

3.1.6 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.

3.1.6.1. 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 28.

Таблица 28 – 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.

3.1.6.2. Метрики Холстеда

Для вычисления метрик Холстеда необходимо измерить 4 характеристики программы [72 – 74]:

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

С целью вычисления этих характеристик были составлен словарь операторов (таблица 29) и словарь операндов (таблица 30).

Таблица 29 – Словарь операторов

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

Таблица 30 – 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.

3.2 Разработка прототипа программы, реализующей улучшенный алгоритм поиска книг в электронной библиотеке

3.2.1 Разработка и тестирование программы

На основе разработанного ранее модифицированного алгоритма поиска книг в электронной библиотеке была создана программа, реализующая его работу [75 – 76]. Код программы представлен в приложении К. Для проверки корректности работы программы был проведено тестирование ее работы для различных типов ошибок (рисунки 72, 73).

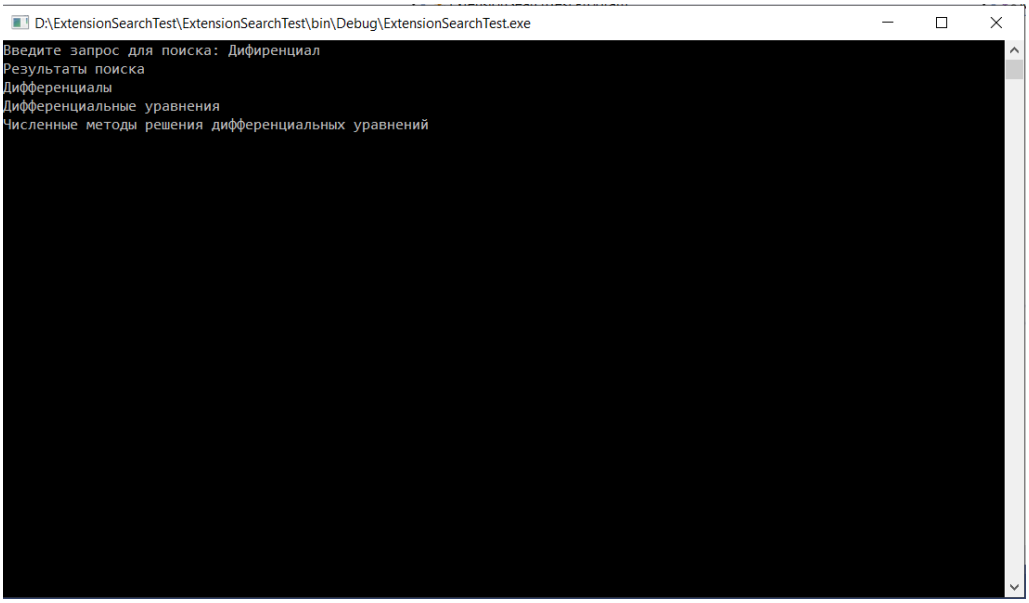


Рисунок 72 – Тестирование алгоритма (неверные и пропущенные символы)

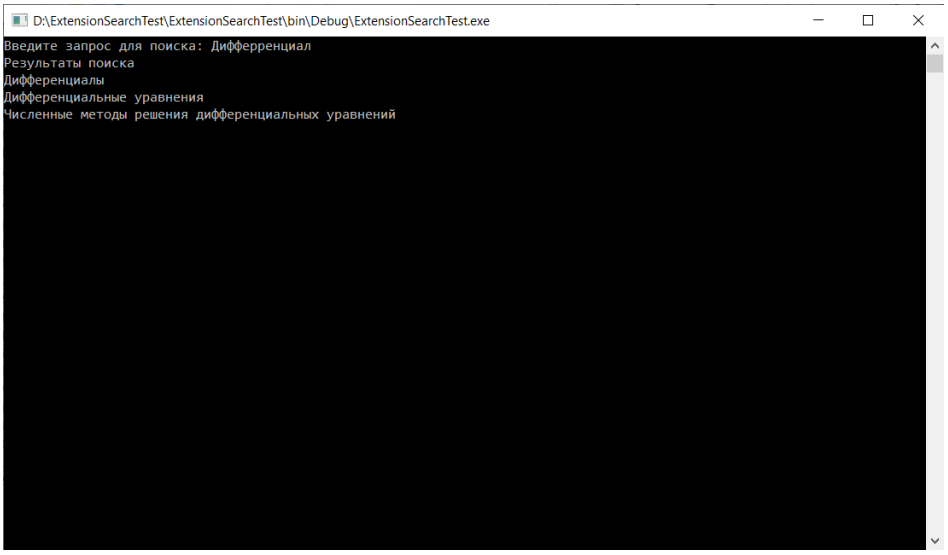


Рисунок 73 – Тестирование алгоритма (лишние символы)

Результаты тестирования показали корректность работы программы.

3.2.2 Эксперимент по проверке эффективности программы, реализующей предложенный модифицированный алгоритм

Для доказательства эффективности предлагаемых улучшений в модифицированном алгоритме поиска был организован и проведен соответствующий эксперимент. Для этого, наряду с программой, реализующей модифицированный алгоритм поиска, была разработана также программа, производящая поиск по классическому алгоритму расширения выборки. Результаты сравнения времени поиска для классического и модифицированного алгоритмов для одних и тех же запросов и на одних и тех же тестовых данных представлены на рисунке 74.

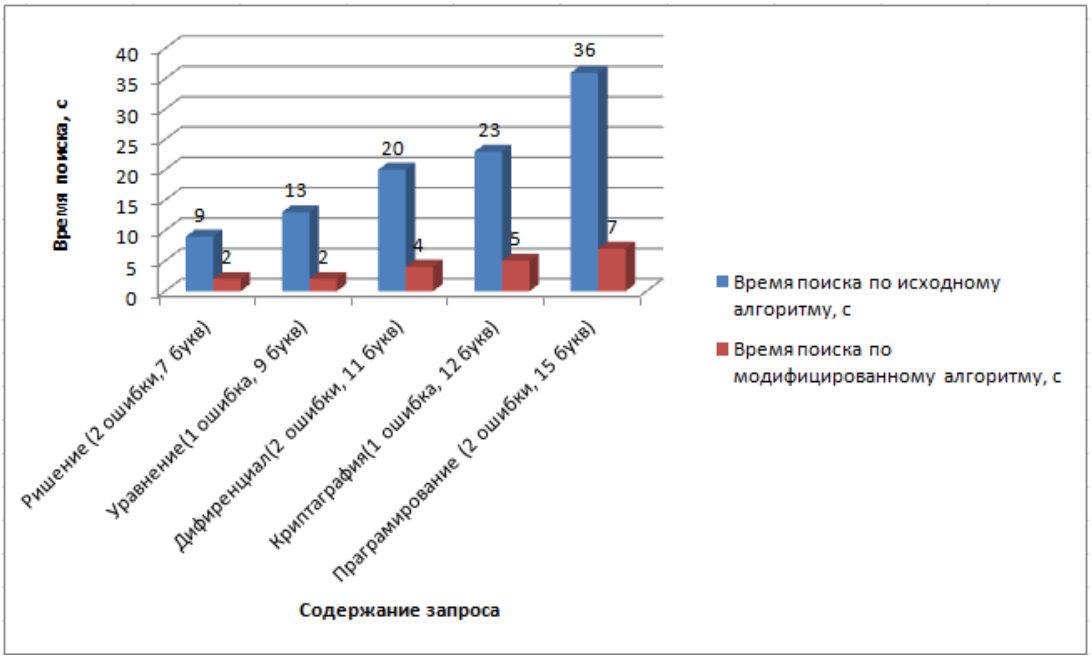


Рисунок 74 –Результаты проведенного эксперимента (время поиска)

Таким образом, как можно заметить, модифицированный алгоритм производит поиск в среднем примерно в 5 раз быстрее классического.

Для проверки сохранения надежности поиска для модифицированного алгоритма в приемлемых пределах используем F-меру Ван Ризбергена. Для ее вычисления сначала нужно найти точность и полноту результатов поиска. Они вычисляются по следующим формулам:

$$Presicion = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

где $Presicion$ – точность, $Recall$ – полнота, TP – число истинно-положительных результатов, FP – число ложно-положительных результатов, FN – число ложно-отрицательных результатов.

F-мера Ван Ризбергена вычисляется по формуле:

$$F = (\beta^2 + 1)\frac{Presicion \times Recall}{\beta^2\, Presicion + Recall},$$

где β – некоторый коэффициент, предназначенный для задания приоритета точности или полноты. При 0< β <1 больший упор делается на точность, при β >1 – на полноту, при β =1 (сбалансированная F-мера) точность и полнота считаются равнозначными. В рамках эксперимента использовалась сбалансированная F-мера, для которой формула вычисления принимает вид:

$$F = \frac{2(Presicion \times Recall)}{Presicion + Recall}.$$

Результаты эксперимента по сравнению F-меры поиска для классического и модифицированного алгоритмов для одних и тех же запросов и на одних и тех же тестовых данных, вычисленные в среде табличного процессора Excel по приведенным выше формулам, представлены на рисунке 75.

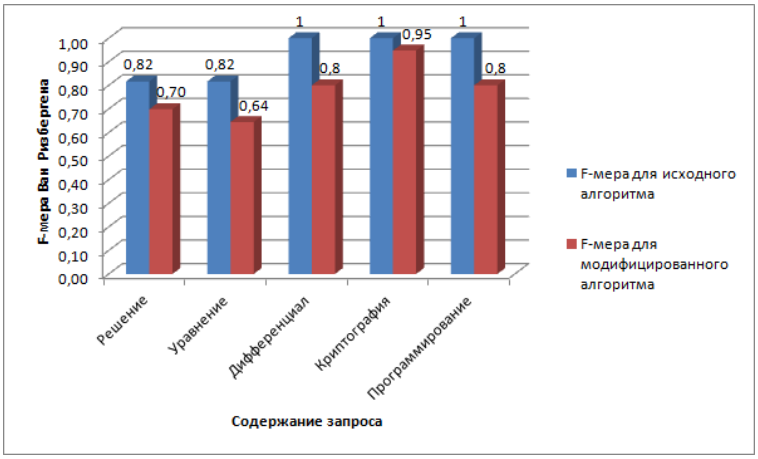


Рисунок 75 – Результаты проведенного эксперимента (F-мера)

Таким образом, для модифицированного алгоритма F-мера в среднем на 0,15 меньше, чем для классического, что свидетельствует о приемлемых потерях в надежности поиска.

Работа программ, используемых для проведения экспериментов представлена на рисунке 76 (классический алгоритм) и рисунке 77 (модифицированный алгоритм).

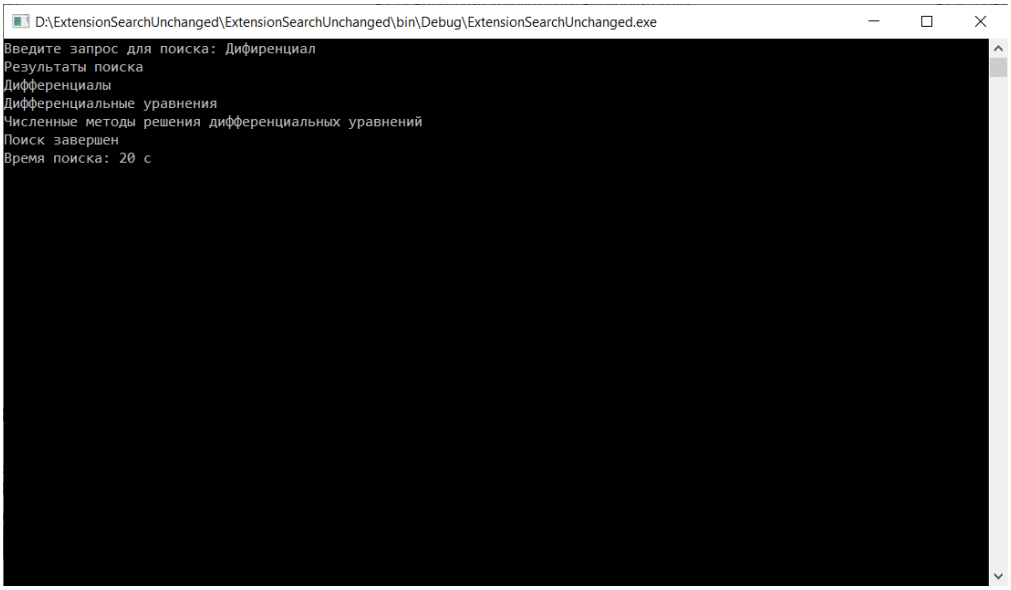


Рисунок 76 – Пример работы программы для проведения экспериментов (классический алгоритм)

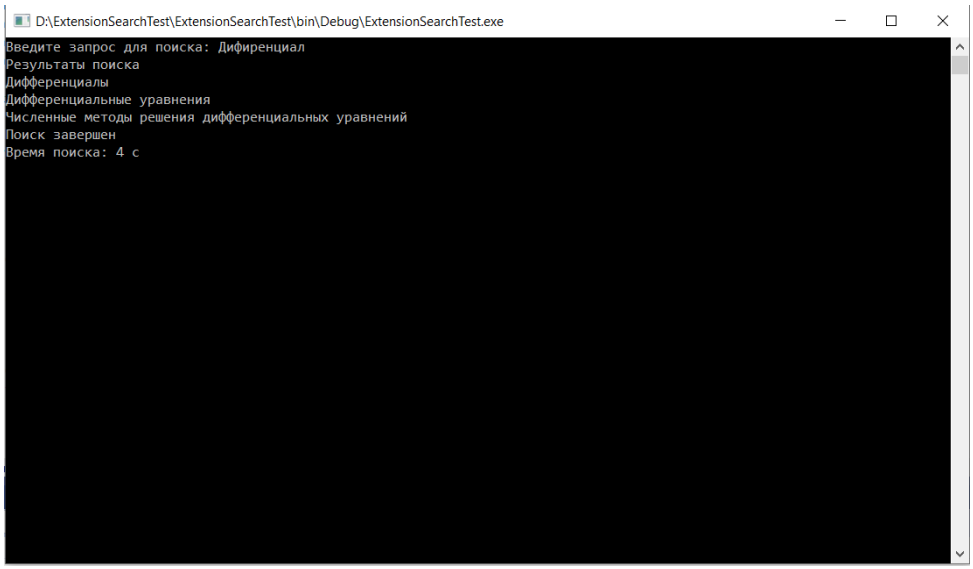


Рисунок 77 – Пример работы программы для проведения экспериментов (модифицированный алгоритм)

На основании полученных в ходе проведения экспериментов результатов можно сделать вывод, что сделанные аналитическим путем выводы об улучшениях, достигаемых при применении модифицированного алгоритма поиска книг, являются полностью достоверными.

Выводы

В рамках реализации были разработаны диаграммы классов, компонентов и развертывания для создаваемой электронной библиотеки. Реализована серверная и клиентская часть системы на языках высокого уровня C# и Java соответственно, а также развернута база данных для серверной части под управлением СУБД MS SQL Server. Затем обе части приложения были протестированы различными способами. Результаты тестирования показали корректность работы как серверной, так и клиентской части. Для одной из подпрограмм серверной части приложения был вычислен ряд метрик кода, которые подтвердили, что код программы имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки.

Также была создана программа, реализующая работу разработанного ранее модифицированного алгоритма поиска книг. Тестирование этой программы показало корректность ее работы, а эффективность внесенных в модифицированный алгоритм улучшений была подтверждена в результате проведенного эксперимента.

Заключение

В результате выполнения диссертационной работы были разработаны электронная научная библиотека и алгоритм поиска книг в электронной библиотеке. Для достижения этих результатов был проведен анализ имеющихся решений, который выявил ряд недостатков в современных электронных библиотеках. С целью исправления этих недостатков было решено реализовать собственную электронную научную библиотеку и создать для нее собственный модифицированный алгоритм поиска книг, а также провести онтологическое и имитационное моделирование по теме предметной области.

В процессе выполнения диссертационной работы была разработана электронная научная библиотека, отвечающая всем предъявленным функциональным и нефункциональным требованиям. Тестирование разработанной электронной библиотеке показало отсутствие ошибок на тестовых наборах входных данных. Для одной из подпрограмм серверной части приложения был вычислен ряд метрик кода, которые подтвердили, что код программы имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки. Также реализован и протестирован разработанный алгоритм поиска книг в электронной библиотеке. Для него был проведен эксперимент, подтверждающий эффективность внесенных модификаций. Кроме того, было проведено онтологическое и имитационное моделирование по теме предметной области и составлена расширенная классификация электронных библиотек.

Программа, реализующая разработанный модифицированный алгоритм поиска книг в электронной библиотеке, была зарегистрирована в государственном Реестре программ для ЭВМ, о чем свидетельствует соответствующее Свидетельство о регистрации, представленное в приложении М.

Основные результаты диссертационной работы состоят в следующем.

1. Проведен критический анализ предметной области по тематике магистерской диссертации.

2. Разработана и реализована полнофункциональная электронная научная библиотека с применением современных и эффективных информационных и программных технологий и методологий, включая онтологическое, имитационное и визуальное моделирование.

3. Разработан и реализован модифицированный алгоритм поиска книг в электронной научной библиотеке, отличающийся от алгоритма расширения выборки, на основе которого он создан, большей скоростью поиска при сохранении приемлемой надежности.

4. Разработан сетевой график IT-проекта по созданию электронной научной библиотеки, и определены срок выполнения и бюджет IT-проекта.

5. Предложен набор дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования электронной научной библиотеки. Важной особенностью предложенного набора моделей является то, что они, в отличие от известных аналогичных моделей, имеют комплексный характер и позволяют учитывать как особенности работы аппаратного обеспечения электронной библиотеки, так и человеческий фактор, связанный с деятельностью ее сотрудников.

6. Предложена расширенная онтологическая модель предметной области для электронных библиотек, отличающаяся от известных расширенным набором сущностей и классов предметной области, с целью формализации и систематизации информации для эффективного проведения проектирования объекта исследования.

7. Предложена модифицированная классификация электронных библиотек, отличающаяся от известных тем, что предлагается более полный расширенный перечень классификационных признаков и систем.

8. По теме магистерской диссертации получены следующие документы:

– Диплом победителя XX Международного научно-исследовательского конкурса «Научные достижения и открытия 2021»,

- Диплом лауреата I степени V Международного научно-исследовательского конкурса «Молодежный исследовательский потенциал»,
- Свидетельство о государственной регистрации программы для ЭВМ в Федеральном институте промышленной собственности РФ.

9. По теме магистерской диссертации сделано 10 публикаций, включая статью в журнале, входящем в Перечень ВАК, и проведена апробация на 8 конференциях различного уровня.

Таким образом, в ходе выполнения магистерской диссертационной работы был разработан программный продукт, полностью удовлетворяющий всем предъявленным требованиям и включающий в себя элементы научной новизны.

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: 16.12.2021).

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: 16.12.2021).

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: 16.12.2021).

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. 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.

6. 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: 16.12.2021).

7. ГОСТ Р 7.0.96 – 2016 «Система стандартов по информации, библиотечному и издательскому делу. Электронные библиотеки. Основные виды. Структура. Технология формирования». [Электронный ресурс]. URL:

https://www.rst.gov.ru/portal/gost/home/standarts/catalognational?portal:componentId=3503536e-2ac1-4753-8ed1-09a92fee02de&portal:isSecure=false&portal:portletMode=view&navigationalstate=JBPNS_rO0ABXc5AAZhY3Rpb24AAAABABBjb25jcmV0ZURvY3VtZW50AAZkb2NfaWQAAAABAAQ1MDI2AAdfX0VPRl9f (дата обращения: 16.12.2021).

8. Электронные ресурсы в современных библиотеках // Крымский федеральный университет [Электронный ресурс]. URL: https://cfuv.ru/wp-content/uploads/2016/02/013-elektron-resurs.pdf (дата обращения 16.12.2021).

9. Городнова, А.А. Электронные библиотеки, электронные каталоги и базы данных: учебное пособие / А.А. Городнова. – Нижний Новгород: Изд-во Волго-Вятской академии госслужбы, 2009. – 176 с.

10. Классификация электронных библиотек // Studbooks.net [Электронный ресурс]. URL: https://studbooks.net/1034622/kulturologiya/klassifikatsiya_elektronnyh_bibliotek#86 (дата обращения: 16.12.2021).

11. Злобина А. Что такое электронные библиотеки // iKirov – последние новости Кирова и Кировской области [Электронный ресурс]. URL: https://www.ikirov.ru/news/10408-chto-takoe-elektronnye-biblioteki (дата обращения: 16.12.2021).

12. 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.

13. Макаренко, И.А. Полнотекстовые ресурсы интернета в обслуживании пользователей муниципальной общедоступной библиотеки / И.А. Макаренко, Л.А. Шелякина. – Архангельск: АОНБ им. Н. А. Добролюбова, 2017. – 19 с.

14. Горелова, С.В. Методические рекомендации по использованию информационно-поисковых систем открытого доступа и электронного каталога OPAC-Global / С.В. Горелова, О.А. Абрамова. – Тула: ГОУ ДПО ТО «ИПК и ППРО ТО», 2017. – 28 с.

15. Статистика по пользователям библиотеки // Научная электронная библиотека Elibrary.ru [Электронный ресурс]. URL: https://elibrary.ru/stat_users.asp (дата обращения: 16.12.2021).

16. О проекте elibrary.ru // Научная электронная библиотека Elibrary.ru [Электронный ресурс]. URL: https://elibrary.ru/elibrary_about.asp (дата обращения: 16.12.2021).

17. Что такое Национальная электронная библиотека (НЭБ)? // Казанский национальный исследовательский технологический университет [Электронный ресурс]. URL: https://www.kstu.ru/event.jsp?id=97398 (дата обращения 16.12.2021).

18. Вопросы и ответы // НЭБ – национальная электронная библиотека [Электронный ресурс]. URL: https://rusneb.ru/faq/ (дата обращения 16.12.2021).

19. Агашина И.В. Ресурсы Национальной электронной библиотеки и использование их в работе библиотек: методические рекомендации // Ярославская областная универсальная научная библиотека имени Н.А. Некрасова [Электронный ресурс]. URL: http://www.rlib.yar.ru/media/Download/metod/2020/NEB.pdf (дата обращения: 16.12.2021).

20. О научной библиотеке открытого доступа // Бесплатная научная библиотека «КиберЛенинка» [Электронный ресурс]. URL: https://cyberleninka.ru/about (дата обращения: 16.12.2021).

21. Как добиться пользы от электронной библиотеки «КиберЛенинка»? // Научная библиотека РАНХиГС [Электронный ресурс]. URL: https://lib.ranepa.ru/ru/novosti/391-kak-dobitsya-polzy-ot-elektronnoj-biblioteki-kiberleninka (дата обращения: 16.12.2021).

22. Резниченко, В. А. О функции поиска в электронной библиотеке / В.А. Резниченко, Г.Ю. Проскудина // Труды 12ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции». – Казань: Изд-во Казанского (Приволжского) федерального университета, 2010. – С. 332 – 326.

23. Блохина Е.С. Алгоритм поиска информации и правила составления запроса в электронном каталоге библиотеки ОмГУ на платформе WEB-ИРБИС // Омский государственный университет [Электронный ресурс]. URL: https://omsu.ru/about/structure/science/ub/20170418_%D0%AD%D0%9A%20web%20%D0%98%D1%80%D0%B1%D0%B8%D1%81%20%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC.pdf (дата обращения 16.12.2021).

24. eLibrary.ru. Руководство пользователя // Оренбургский государственный педагогический университет [Электронный ресурс]. URL: https://ospu.ru/assets/resources/articles/biblioteka/manual_elibrary_for_user.pdf (дата обращения: 16.12.2021).

25. 6 алгоритмов поиска на Java: от простого к сложному // Библиотека программиста [Электронный ресурс]. URL: https://proglib.io/p/6-search-algorithms-java (дата обращения: 16.12.2021).

26. Что такое экстремальное программирование // Университет «Синергия» [Электронный ресурс]. URL: https://synergy.ru/akademiya/programming/chto_takoe_ekstremalnoe_programmirovanie (дата обращения: 05.07.2022).

27. Марчуков А., Савельев А. Знакомство с методологией Microsoft Solution Framework (MSF), основные компоненты и принципы методологии, дисциплина управления проектом MSF // НОУ «Интуит» [Электронный ресурс]. URL: https://intuit.ru/studies/courses/499/355/lecture/8449 (дата обращения: 05.07.2022).

28. Кролл, П. Rational Unified Process – это легко. Руководство по RUP для практиков / П. Кролл, Ф. Крачтен. – М.: КУДИЦ-ОБРАЗ, 2004. – 432 с.

29. Орлов, С. А. Технологии разработки программного обеспечения / С. А. Орлов, Б. Я. Цилькер. – СПб: Питер, 2012. – 464 с.

30. Гендиректор РГБ указал на рост популярности у россиян цифровых ресурсов библиотек // Рамблер [Электронный ресурс]. URL: https://kino.rambler.ru/movies/48728449-gendirektor-rgb-ukazal-na-rost-populyar

nosti-u-rossiyan-tsifrovyh-resursov-bibliotek/?ysclid=lhnfnqagq4963920893 (дата обращения: 07.06.2023).

31. Global Smartphone Market Share: Q2 2021 to Q1 2023 // CounterPoint Research [Электронный ресурс]. URL: https:// www.counterpointresearch.com/global-smartphone-share/ (дата обращения: 07.06.2023).

32. Продажи смартфонов в России, по производителям // ТопикРу [Электронный ресурс]. URL: https://topic.ru/statistics/tehnologies-and-telecommunications/consumer-electronics/prodazhi-smartfonov-v-rossii-po-proizvoditelyam/ (дата обращения: 07.06.2023).

33. Всё грустно. Google обнародовала статистику версий Android // 4PDA [Электронный ресурс]. URL: https://4pda.to/2023/01/19/408761/vsyo_grustno_google_obnarodovala_statistiku_versij_android/ (дата обращения: 07.06.2023).

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

35. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. –102 с.

36. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.

37. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.

38. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. – СПб.: Питер, 2019. – 208 с.

39. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 с.

40. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб:БХВ-Петербург, 2013. – 432 с.

41. Джонсон, Т. Microsoft Project 2016. Шаг за шагом / Т. Джонсон, К. Четфилд. – М., Изд-во «КТК Галактика», 2018. – 566 с.

42. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.

43. Цуканова, Н.И. Онтологическая модель представления и организации знаний / Н.И. Цуканова. – М.: Горячая линия-Телеком, 2015. – 272 с.

44. Советов, Б.Я. Моделирование систем: учебник / Б.Я. Советов, С.А. Яковлев. – М., Издательство Юрайт, 2021. – 343 с.

45. Серова Е.Г. Имитационное моделирование в современном менеджменте // Портал GPSS.ru [Электронный ресурс]. URL: http://www.gpss.ru/immod07/doklad/37.html (дата обращения: 05.06.22).

46. Осипов, Г.С. Математическое и имитационное моделирование системы массового обслуживания / Г.С. Осипов. – М: Издательский дом Академии Естествознания, 2017. – 56 с.

47. Ослин, Б.Г. Моделирование. Имитационное моделирование СМО: учебное пособие / Б.Г. Ослин. – Томск: Издательство Томского политехнического университета, 2010. – 128 с.

48. Бочаров, Е.П. Имитационная модель производственного процесса как элемент системы управления промышленным предприятием / Е.П. Бочаров, О.Н. Алексенцева, Д.В. Ермошин // Прикладная информатика. – 2007. – №3 (9). – С. 3 – 11.

49. Бабина, О.И. Имитационное моделирование процессов планирования на промышленном предприятии: монография / О.И. Бабина, Л.И. Мошкович. – Красноярск: Изд-во Сибирского федерального университета, 2014. – 200 с.

50. Эльберг, М. С. Имитационное моделирование : учеб. пособие / М. С. Эльберг, Н. С. Цыганков. – Красноярск : Сиб. федер. ун-т, 2017. – 128 с.

51. Прокимнов, Н.Н. Зарубежная практика промышленного применения технологий имитационного моделирования / Н.Н. Прокимнов // Прикладная информатика. – 2015. – №1 (55). – С. 71–84.

52. Ащеулов, И.О. Исследование электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов // Молодежный исследовательский потенциал: сборник статей V Международного научно-исследовательского конкурса. – Петрозаводск: МЦНС «Новая наука», 2022. – С. 77 – 89.

53. Ащеулов, И.О. Исследование работы сервера электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Материалы всероссийской (национальной) научной конференции «Фундаментальные и прикладные исследования. Актуальные проблемы и достижения». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 11–13.

54. Ащеулов, И.О. Исследование работы читального зала электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Материалы международной научной конференции «Высокие технологии и инновации в науке». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 81–84.

55. Ащеулов, И.О. Исследование работы администраторов электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Материалы международной студенческой научной конференции «Поколение будущего». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 79–83.

56. Ащеулов, И.О. Исследование работы оцифровщиков книг электронной научной библиотеки с использованием средств имитационного моделирования / И.О. Ащеулов, В.Н. Князев // Информационные технологии в науке и образовании. Проблемы и перспективы: сборник статей по материалам IX Всероссийской научно-практической конференции. – Пенза: Изд-во ПГУ, 2022. – С. 73–76.

57. Богданова А.Ю., Теплянских Д.И., Выдрин В.А. Проведение валидации и верификации на примере имитационной модели телефонной станции // Россия молодая: сборник материалов XIII Всероссийской научно-практической конференции с международным участием (2021).

[Электронный ресурс]. URL: https://science.kuzstu.ru/wp-content/Events/Conference/RM/2021/RM21/pages/Articles/031605.pdf (дата обращения: 26.03.2023).

58. Стародубов, А.Н. Верификация и валидация имитационных моделей систем: методические указания / А.Н.Стародубов, В.В. Зиновьев, П.И. Николаев, И.С. Кузнецов. – Кемерово: Изд-во КузГТУ, 2018. – 25 с.

59. 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.

60. Беленький, В. М. Анализ поисковых алгоритмов при решении задач идентификации объектов в слабоструктурированных базах данных / В. М. Беленький, Д. С. Карахтанов // Молодой ученый. – 2010. – № 8 (19). – Т. 1. – С. 150 – 155.

61. Сметанин Н. Нечёткий поиск в тексте и словаре // Хабр [Электронный ресурс]. URL: https://habr.com/ru/post/114997/ (дата обращения: 25.12.2022).

62. Лиманова, Н.И. Алгоритм нечеткого поиска в базах данных и его практическая реализация / Н.И. Лиманова, М.Н. Седов // Информационные технологии и нанотехнологии (ИТНТ-2017): сборник трудов III международной конференции и молодежной школы. – Самара: Изд-во «Новая техника», 2017. – С. 1893 – 1897.

63. 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.

64. Введение в Web API // Metanit.com [Электронный ресурс]. URL: https://metanit.com/sharp/mvc/12.1.php (дата обращения: 26.03.2023).

65. 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: 26.03.2023).

66. Руководство по Entity Framework // Metanit.com [Электронный ресурс]. URL: https://metanit.com/sharp/entityframework/2.4.php (дата обращения: 26.03.2023).

67. Database First // Microsoft [Электронный ресурс]. URL: https:// docs.microsoft.com/ru-ru/ef/ef6/modeling/designer/workflows/database-first (дата обращения: 26.03.2023).

68. Entity Framework 6. Использование Database First // Professor Web [Электронный ресурс]. URL: https://professorweb.ru/my/entity-framework/6/level1/1_6.php (дата обращения: 26.03.2023).

69. Лабун, Б. Дружеское знакомство с тестированием программ / Б. Лабун. – СПб.: БХВ-Петербург, 2022. – 288 с.

70. Майерс, Г. Искусство тестирования программ / Г. Майерс, Т. Баджетт, К. Сандлер. – Москва : Санкт-Петербург : Диалектика, 2019. – 271 с.

71. Куликов, С.С. Тестирование программного обеспечения. Базовый курс / С.С. Куликов. – Минск: Четыре четверти, 2020 – 312 с.

72. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: https://habr.com/ru/company/intel/blog/106082/ (дата обращения: 26.03.2023).

73. 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.

74. 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.

75. Ащеулов, И.О. Разработка алгоритма поиска книг в электронной библиотеке / И.О. Ащеулов, В.Н. Князев // Сборник материалов X Ежегодной Всероссийской межвузовской научно-практической конференции «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза: Издательство ПГУ, 2023. – С. 167 – 171.

76. Ащеулов, И.О. Вопросы создания модифицированного алгоритма поиска книг в электронной библиотеке / И.О. Ащеулов, В.Н. Князев // Сборник материалов XV Международной научно-практической конференции «Современные тенденции развития науки и мирового сообщества в эпоху цифровизации». – Махачкала: Издательство «Алеф», 2023. – С. 202 – 208.

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 ИМИТАЦИОННЫХ МОДЕЛЕЙ

(рекомендуемое)

Листинг Б.1 – Исходный код модели блока администраторов

```
XPDIS FUNCTION RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99
.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
Admin       STORAGE   9
Tab         TABLE     M1,10,10,30
Tab1        TABLE     M1,10,10,30
Tab2        TABLE     M1,10,10,30
Tab3        TABLE     M1,10,10,30
Tab4        TABLE     M1,10,10,30
Tab5        TABLE     M1,10,10,30
Tab6        TABLE     M1,10,10,30
Tab7        TABLE     M1,10,10,30
Tab8        TABLE     M1,10,10,30
Tab9        TABLE     M1,10,10,30

;цифровые копии
            GENERATE  1.03,FN$XPDIS; Статья(1)
            ASSIGN    1,1
            TRANSFER  ,add
            GENERATE  240,FN$XPDIS; Сборник статей(2)
            ASSIGN    1,2
            TRANSFER  ,add
            GENERATE  45,FN$XPDIS; Автореферат диссертации(3)
            ASSIGN    1,3
            TRANSFER  ,add
            GENERATE  800,FN$XPDIS;Патент (4)
            ASSIGN    1,4
            TRANSFER  ,add
            GENERATE  30,FN$XPDIS; Монография (5)
            ASSIGN    1,5
            TRANSFER  ,add
            GENERATE  1.08,FN$XPDIS; Материал конференции (6)
            ASSIGN    1,6
            TRANSFER  ,add
            GENERATE  80,FN$XPDIS; Сборник трудов конференции (7)
            ASSIGN    1,7
            TRANSFER  ,add
            GENERATE  31,FN$XPDIS; Диссертация (8)
            ASSIGN    1,8
            TRANSFER  ,add
            GENERATE  9,FN$XPDIS; Номер журнала (9)
            ASSIGN    1,9
add         QUEUE     Och1
            ENTER     Admin
            DEPART    Och1
            ADVANCE   3,FN$XPDIS; Добавление книги
            LEAVE     Admin
            TABULATE  Tab; Время работы с электронными книгами вне зависимости
от типа книг
            TEST E    P1,1,type2
            TABULATE  Tab1; Время работы со статьями(1)
            TRANSFER  ,finish
type2       TEST E    P1,2,type3
            TABULATE  Tab2; Время работы со сборниками статей(2)
            TRANSFER  ,finish
type3       TEST E    P1,3,type4
            TABULATE  Tab3; Время работы с авторефератами диссертацийй(3)
            TRANSFER  ,finish
type4       TEST E    P1,4,type5
            TABULATE  Tab4; Время работы с патентами (4)
```

```
            TRANSFER  ,finish
type5       TEST E    P1,5,type6
            TABULATE  Tab5; Время работы с монографиями (5)
            TRANSFER  ,finish
type6       TEST E    P1,6,type7
            TABULATE  Tab6; Время работы с материалами конференций (6)
            TRANSFER  ,finish
type7       TEST E    P1,7,type8
            TABULATE  Tab7; Время работы со сборниками трудов конференций (7)
            TRANSFER  ,finish
type8       TEST E    P1,8,type9
            TABULATE  Tab8; Время работы с диссертациями (8)
            TRANSFER  ,finish
type9       TEST E    P1,9,finish
            TABULATE  Tab9; Время работы с номерами журналов (9)
finish      TERMINATE


            GENERATE  2400
            TERMINATE 1
            START     1
```

Листинг Б.2 – Исходный код модели блока оцифровщиков

```
XPDIS FUNCTION RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99
.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
Pages       FVARIABLE P2#FN$XPDIS
EstTime     FVARIABLE V$Pages#0.6
Digitizer   STORAGE   4
Tab         TABLE     M1,20,20,30
Tab2        TABLE     M1,20,20,30
Tab3        TABLE     M1,20,20,30
Tab4        TABLE     M1,20,20,30
Tab5        TABLE     M1,20,20,30
Tab7        TABLE     M1,20,20,30
Tab8        TABLE     M1,20,20,30
Tab9        TABLE     M1,20,20,30
;бумажные копии
            GENERATE  2400,FN$XPDIS; Сборник статей(2)
            ASSIGN    1,2
            ASSIGN    2,200
            TRANSFER  ,digitize
            GENERATE  624,FN$XPDIS; Автореферат диссертации(3)
            ASSIGN    1,3
            ASSIGN    2,25
            TRANSFER  ,digitize
            GENERATE  8000,FN$XPDIS;Патент (4)
            ASSIGN    1,4
            ASSIGN    2,5
            TRANSFER  ,digitize
            GENERATE  300,FN$XPDIS; Монография (5)
            ASSIGN    1,5
            ASSIGN    2,200
            TRANSFER  ,digitize
            GENERATE  800,FN$XPDIS; Сборник трудов конференции (7)
            ASSIGN    1,7
            ASSIGN    2,150
            TRANSFER  ,digitize
            GENERATE  312,FN$XPDIS; Диссертация(8)
            ASSIGN    1,8
            ASSIGN    2,180
            TRANSFER  ,digitize
```

```
            GENERATE  90,FN$XPDIS; Номер журнала (9)
            ASSIGN    1,9
            ASSIGN    2,200
digitize    QUEUE     Och1
            ENTER     Digitizer
            DEPART    Och1
            ADVANCE   V$EstTime,FN$XPDIS; Оцифровка
            LEAVE     Digitizer
            TABULATE  Tab; Время работы с бумажными книгами вне зависимости от
типа книг
            TEST E    P1,2,type3
            TABULATE  Tab2; Время работы со сборниками статей(2)
            TRANSFER  ,finish
type3       TEST E    P1,3,type4
            TABULATE  Tab3; Время работы с авторефератами диссертациий(3)
            TRANSFER  ,finish
type4       TEST E    P1,4,type5
            TABULATE  Tab4; Время работы с патентами (4)
            TRANSFER  ,finish
type5       TEST E    P1,5,type7
            TABULATE  Tab5; Время работы с монографиями (5)
            TRANSFER  ,finish
type7       TEST E    P1,7,type8
            TABULATE  Tab7; Время работы со сборниками трудов конференций (7)
            TRANSFER  ,finish
type8       TEST E    P1,8,type9
            TABULATE  Tab8; Время работы с диссертациями (8)
            TRANSFER  ,finish
type9       TEST E    P1,9,finish
            TABULATE  Tab9; Время работы с номерами журналов (9)
finish      TERMINATE

            GENERATE  2400
            TERMINATE 1
            START     1
```

Листинг Б.3 – Исходный код модели блока электронного читального зала

```
XPDIS FUNCTION RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99
.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
Librarian  STORAGE   2
Computer   STORAGE   7
Tab        TABLE     M1,20,20,30
           GENERATE  6,FN$XPDIS; Приход посетителя
           QUEUE     Och1
           ENTER     Librarian
           DEPART    Och1
           ADVANCE   3,FN$XPDIS; Библиотекарь общается с посетителем
           LEAVE     Librarian
           QUEUE     Och2
           ENTER     Computer
           DEPART    Och2
           ADVANCE   30,FN$XPDIS; Посетитель работает за компьютером
           LEAVE     Computer
           TABULATE  Tab; Время пребывания пользователя в библиотеке
           TERMINATE

           GENERATE  480
           TERMINATE 1
           START     1
```

Листинг Б.4 – Исходный код модели блока запросов к серверу

```
XPDIS FUNCTION RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99
.96,3.2/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
Server  STORAGE   1
Tab     TABLE     M1,200,200,10
Tab1    TABLE     M1,200,200,50
Tab2    TABLE     M1,200,200,50
Tab3    TABLE     M1,200,200,50
Tab4    TABLE     M1,200,200,50
Tab5    TABLE     M1,200,200,50
Tab6    TABLE     M1,200,200,50
Tab7    TABLE     M1,200,200,50
Tab8    TABLE     M1,200,200,50
Tab9    TABLE     M1,200,200,50

        GENERATE  1577,FN$XPDIS; Запрос на статью(1)
        ASSIGN    1,1
        TRANSFER  ,request
        GENERATE  368286,FN$XPDIS; Запрос на сборник статей(2)
        ASSIGN    1,2
        TRANSFER  ,request
        GENERATE  473100,FN$XPDIS; Запрос на автореферат диссертации(3)
        ASSIGN    1,3
        TRANSFER  ,request
        GENERATE  1227621,FN$XPDIS; Запрос на патент (4)
        ASSIGN    1,4
        TRANSFER  ,request
        GENERATE  46036,FN$XPDIS; Запрос на монографию (5)
        ASSIGN    1,5
        TRANSFER  ,request
        GENERATE  1659,FN$XPDIS; Запрос на материал конференции (6)
        ASSIGN    1,6
        TRANSFER  ,request
        GENERATE  122762,FN$XPDIS; Запрос на сборник трудов конференции (7)
        ASSIGN    1,7
        TRANSFER  ,request
        GENERATE  47310,FN$XPDIS; Запрос на диссертацию(8)
        ASSIGN    1,8
        TRANSFER  ,request
        GENERATE  13640,FN$XPDIS; Запрос на номер журнала (9)
        ASSIGN    1,9
request QUEUE     Och
        ENTER     Server
        DEPART    Och
        ADVANCE   100,FN$XPDIS; Обработка запроса
        LEAVE     Server
        TABULATE  Tab; Время обработки запросов вне зависимости от типа книг
        TEST E    P1,1,type2
        TABULATE  TAB1; Время обработки запросов на статью(1)
        TRANSFER  ,finish
type2   TEST E    P1,2,type3
        TABULATE  TAB2; Время обработки запросов на сборник статей(2)
        TRANSFER  ,finish
type3   TEST E    P1,3,type4
        TABULATE  TAB3; Время обработки запросов на автореферат
диссертации(3)
        TRANSFER  ,finish
type4   TEST E    P1,4,type5
        TABULATE  TAB4; Время обработки запросов на патент (4)
        TRANSFER  ,finish
```

```
type5   TEST E    P1,5,type6
        TABULATE  TAB5; Время обработки запросов на монографию (5)
        TRANSFER  ,finish
type6   TEST E    P1,6,type7
        TABULATE  TAB6; Время обработки запросов на материал конференции (6)
        TRANSFER  ,finish
type7   TEST E    P1,7,type8
        TABULATE  TAB7; Время обработки запросов на сборник трудов
конференции (7)
        TRANSFER  ,finish
type8   TEST E    P1,8,type9
        TABULATE  TAB8; Время обработки запросов на диссертацию(8)
        TRANSFER  ,finish
type9   TEST E    P1,9,finish
        TABULATE  TAB9; Время обработки запросов на номер журнала (9)
finish  TERMINATE

        GENERATE  86400000
        TERMINATE 1
        START     1
```

Приложение В

ВЕРИФИКАЦИЯ ИМИТАЦИОННОЙ МОДЕЛИ

(рекомендуемое)

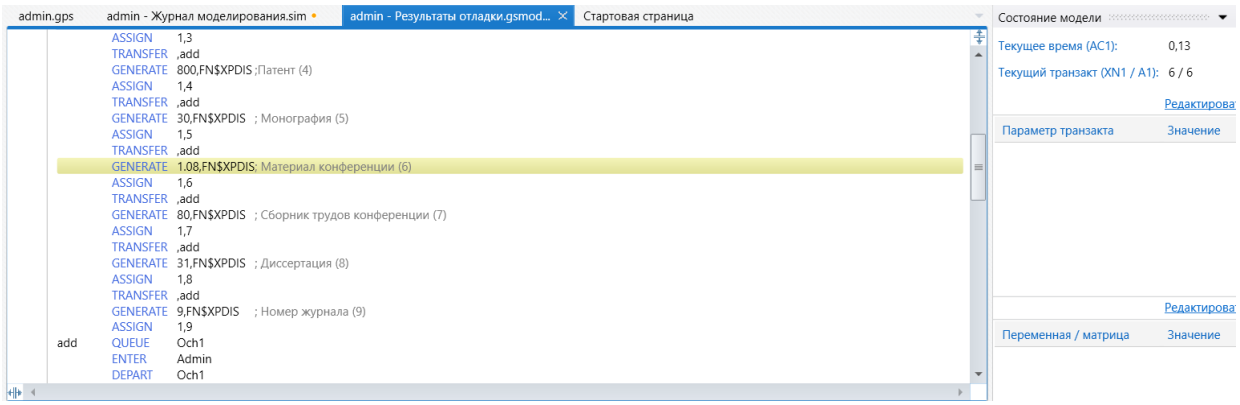


Рисунок В.1 – Поступление заявки (материал конференции)

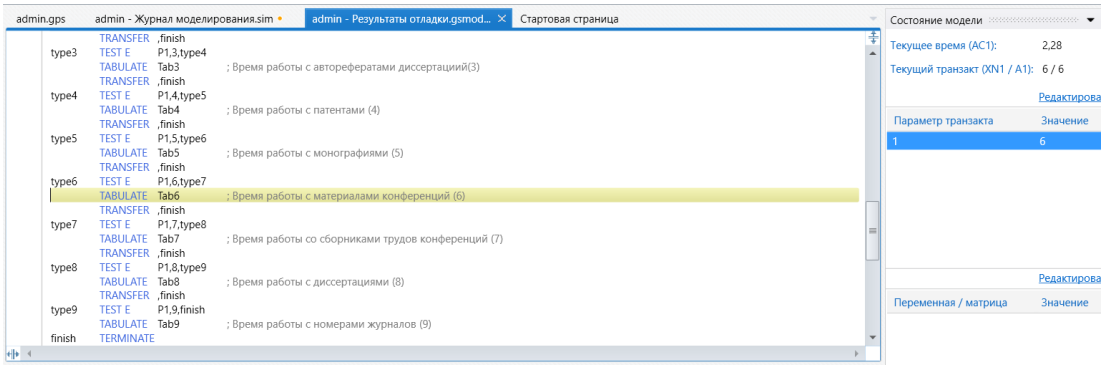


Рисунок В.2 – Результат обработки заявки (материал конференции)

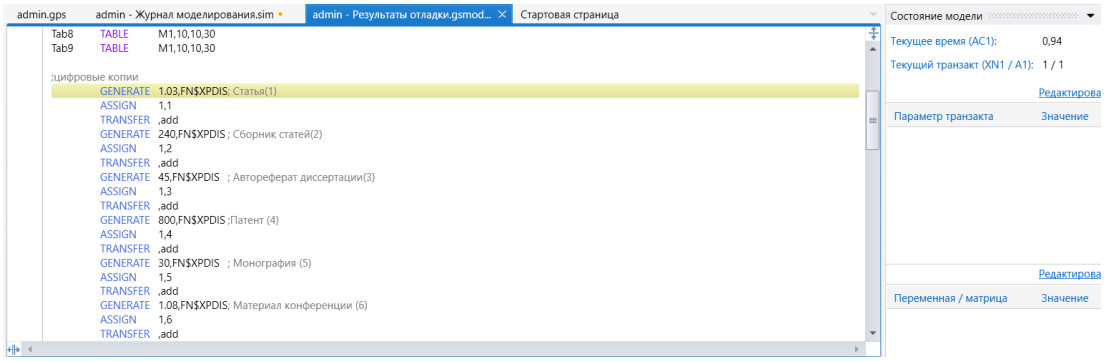


Рисунок В.3 – Поступление заявки (статья)

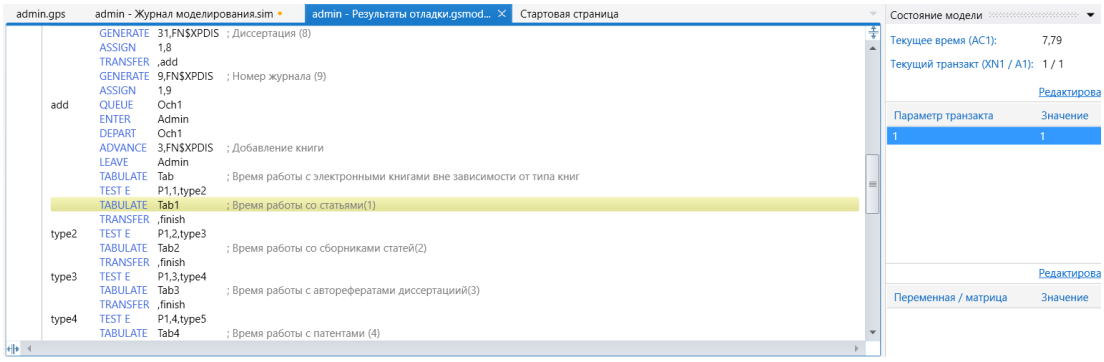


Рисунок В.4 – Результат обработки заявки (статья)

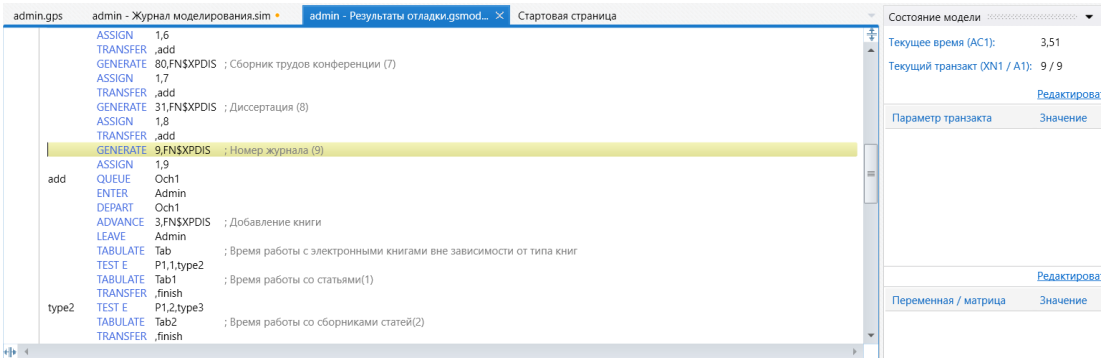


Рисунок В.5 – Поступление заявки (номер журнала)

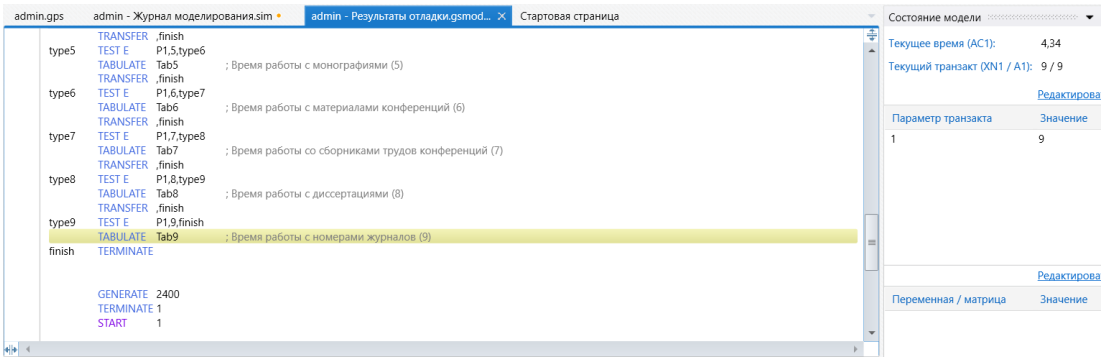


Рисунок В.6 – Результат обработки заявки (номер журнала)

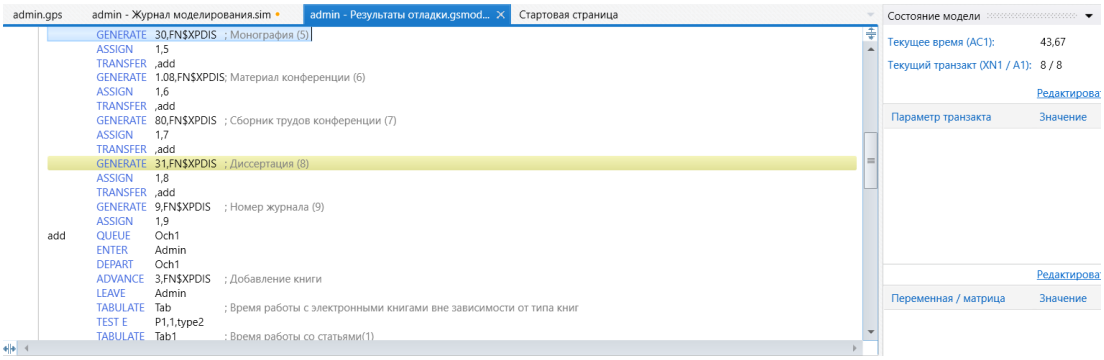


Рисунок В.7 – Поступление заявки (диссертация)

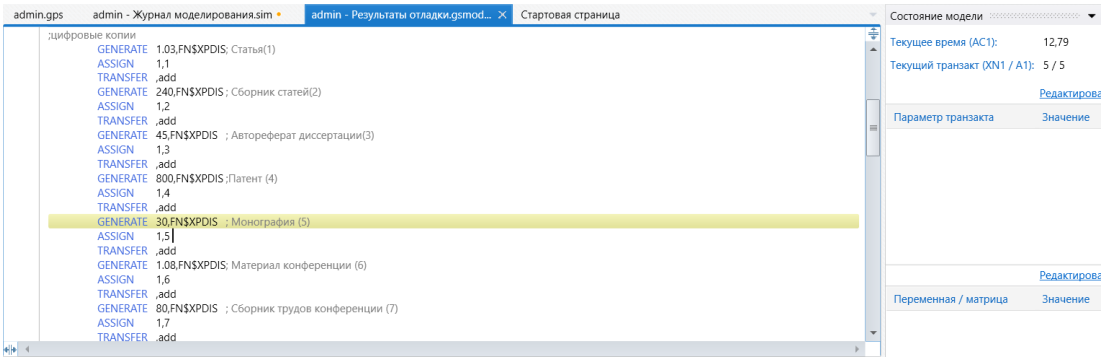


Рисунок В.8 – Результат обработки заявки (диссертация)

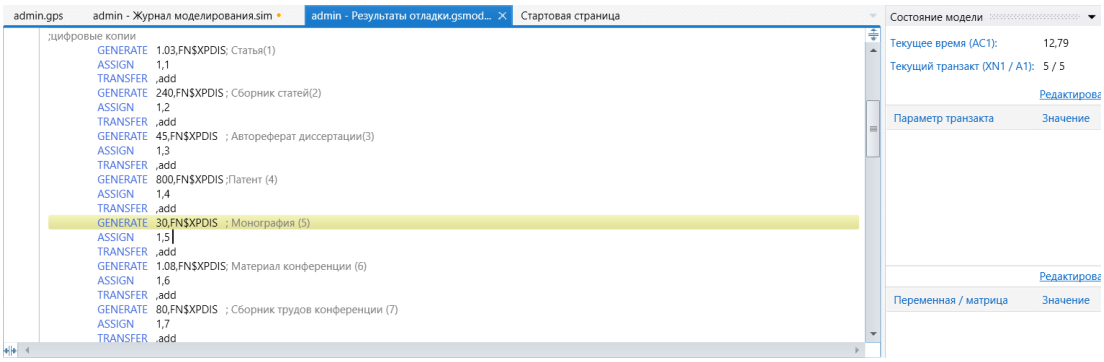


Рисунок В.9 – Поступление заявки (монография)

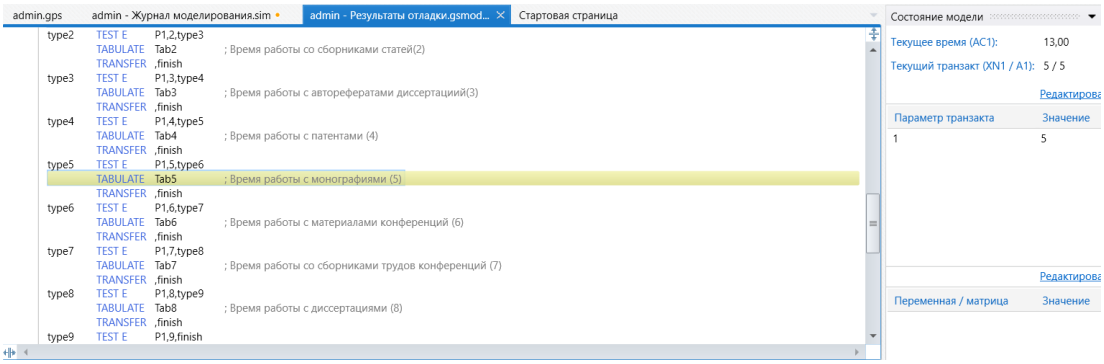


Рисунок В.10 – Результат обработки заявки (монография)

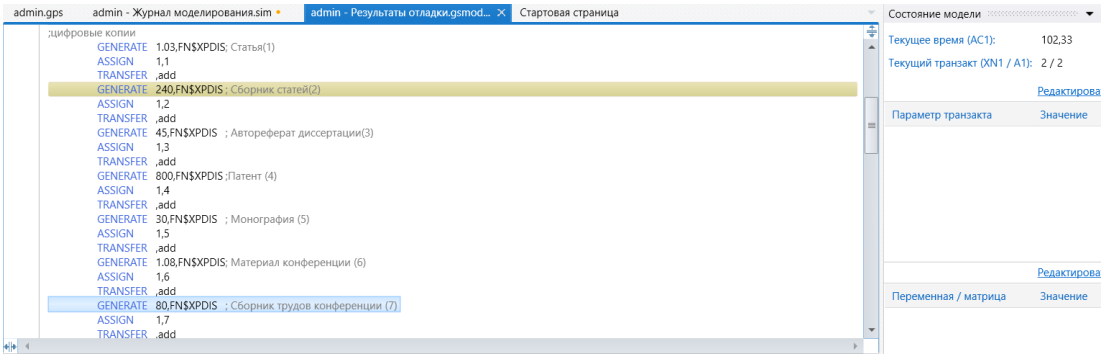


Рисунок В.11 – Поступление заявки (сборник статей)

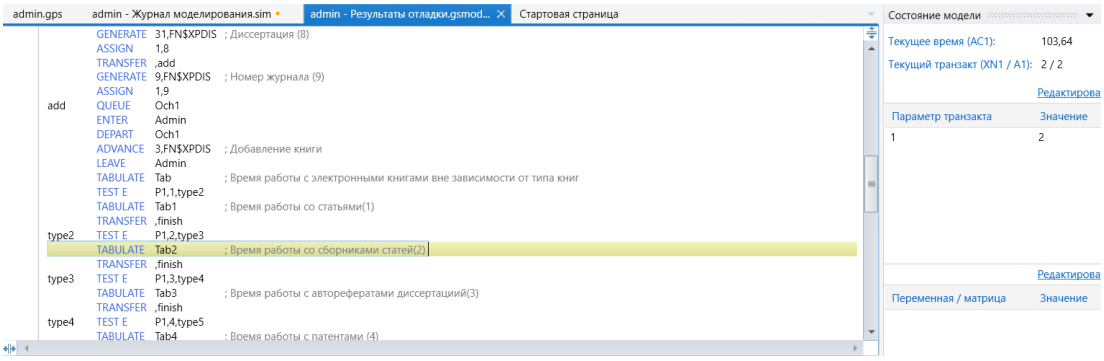


Рисунок В.12 – Результат обработки заявки (сборник статей)

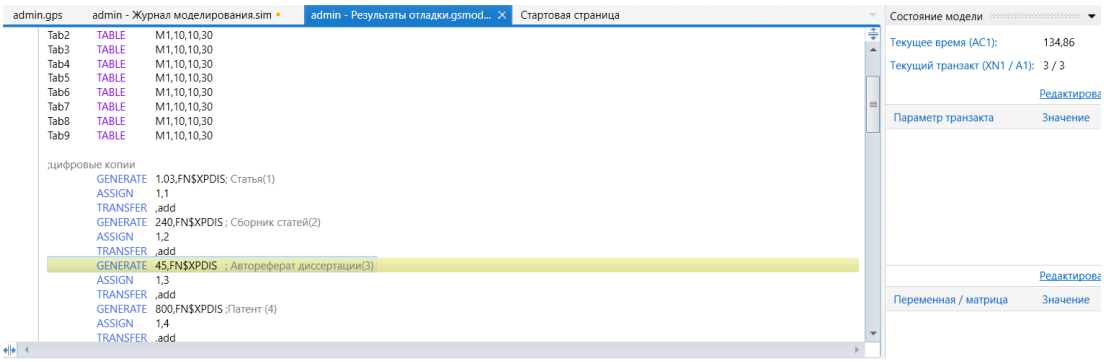


Рисунок В.13 – Поступление заявки (автореферат)

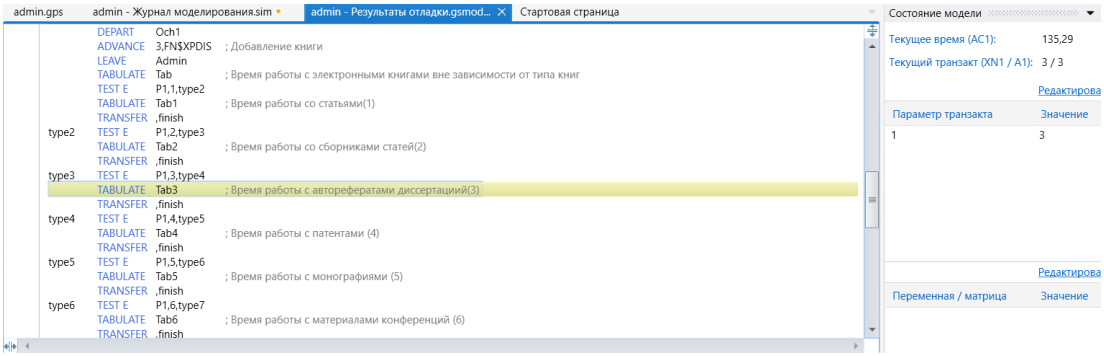


Рисунок В.14 – Результат обработки заявки (автореферат)

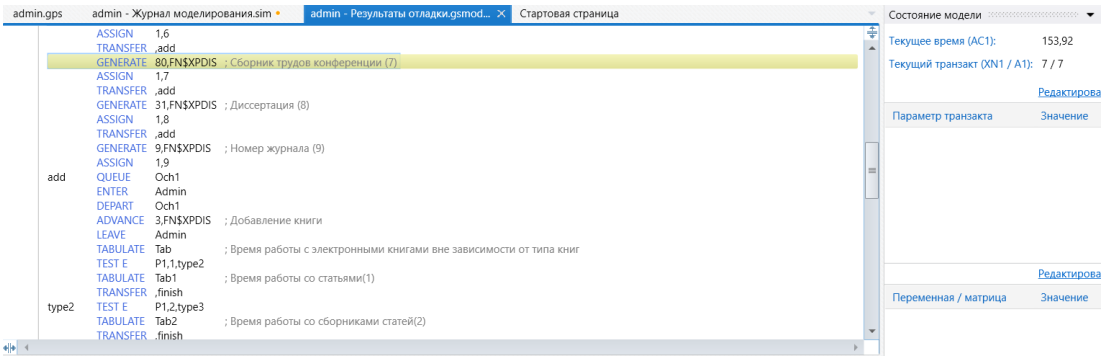


Рисунок В.15 – Поступление заявки (сборник конференции)

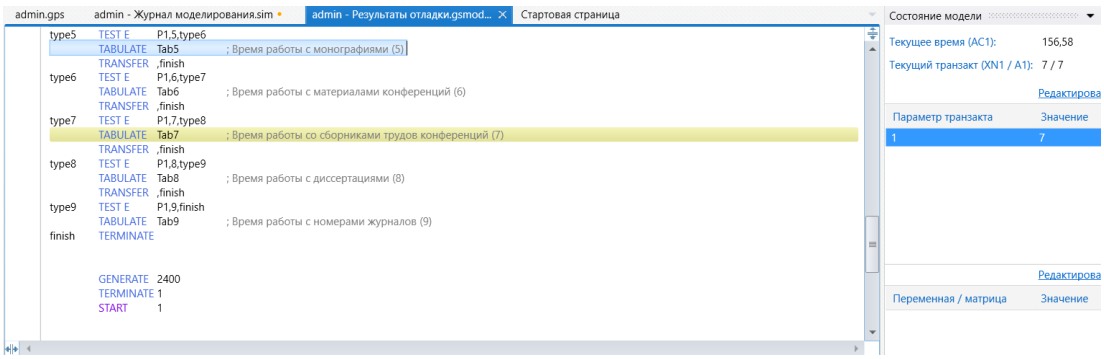


Рисунок В.16 – Результат обработки заявки (сборник конференции)

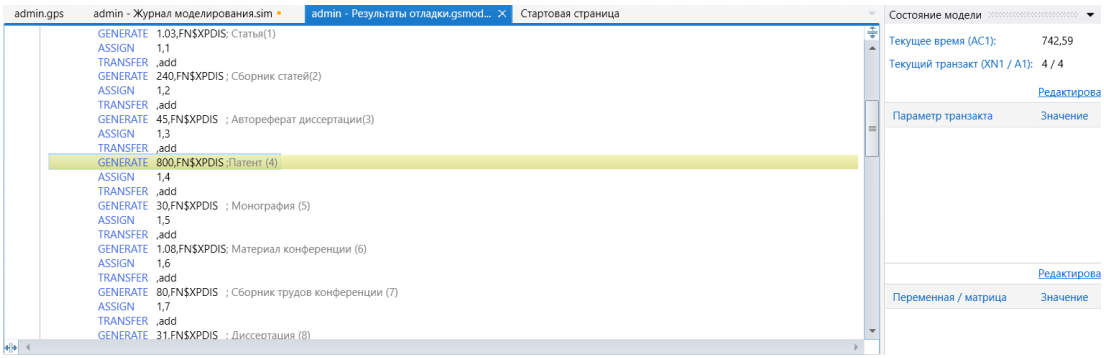


Рисунок В.17 – Поступление заявки (патент)

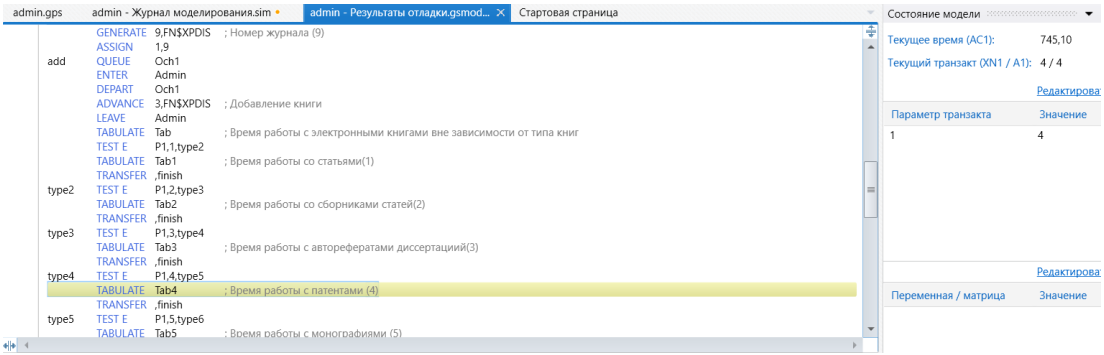


Рисунок В.18 – Результат обработки заявки (патент)

Приложение Г

ИСХОДНЫЙ КОД СКРИПТОВ БАЗЫ ДАННЫХ

(рекомендуемое)

Листинг Г.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 '%[^A-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 '%[^A-zA-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 '%[^A-z0-9_]%'),
      Acc_Password varchar(20) NOT NULL
      CONSTRAINT Validation_Password
            CHECK  (Acc_Password NOT LIKE '%[^A-z0-9]%' 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 '%[^A-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 '%[^A-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,
      Publishment_Year integer  NOT NULL
      CONSTRAINT Validation_Book_Publishment
            CHECK  ( [Publishment_Year]>=(1980) AND [Publishment_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]<=(100000000) ),
      ID_User  integer  NOT NULL FOREIGN KEY REFERENCES User_Acc(ID)
      CONSTRAINT Load_User_ID_Validation
```

```
            CHECK  ( [ID_User]>=(1) AND [ID_User]<=(100000000) ),
      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 Г.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 Г.3 – Исходный код создания представления Comment_Info (основная информация о комментарии)

```
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 Г.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 Г.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 Г.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 Г.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 Д

ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ

(рекомендуемое)

Листинг Д.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)
        {
            var preview = context.Preview.Find(id);
            if (preview != null)
                  Preview = preview.Prev_Text;
            else
                  Preview = null;
            Comments = context.Comment_Info.Where(c => c.ID_Book ==
                  id).ToList();
        }
    }
}
```

Листинг Д.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 Д.3 – Файл PublisherInfo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
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 Д.4 – Файл TagInfo.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 Д.5 – Файл UserInfo.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<DateTime> Banned_Until { get; set; }
    }
}
```

Листинг Д.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 Д.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 Д.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.SaveChanges();
            transaction.Commit();
            string rootPath = Directory.GetParent
                (Server.MapPath("")).FullName;
            string savePath = Path.Combine(rootPath, path);
            file.SaveAs(savePath);
            return new HttpStatusCodeResult(HttpStatusCode.Created);
        }
        catch
        {
            return new HttpStatusCodeResult
                (HttpStatusCode.BadRequest);
        }
    }
}
// DELETE: Book/Delete/5
[HttpDelete]
public ActionResult Delete(int id)
{
    Book book = db.Book.FirstOrDefault(b => b.ID == id);
    if (book != null)
    {
        db.Book.Remove(book);
        db.SaveChanges();
        return new EmptyResult();
    }
    return HttpNotFound();
}
// 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 Д.9 – Файл CommentController.cs

```
using System.Linq;
using System.Net;
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 Д.10 – Файл PublisherController.cs

```
using System.Collections.Generic;
using System.Net;
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 Д.11 – Файл TagController.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
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 Д.12 – Файл UserController.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Web.Mvc;
using MyScienceLibrary.Models;
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.Where(u => u.Acc_Login == login
                && u.Acc_Password == password).FirstOrDefault();
            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(user);
            }
            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 Е

ИСХОДНЫЙ КОД КЛИЕНТСКОЙ ЧАСТИ

(рекомендуемое)

Листинг Е.1 – Файл BanActivity.java

```
package com.myscilib;

import androidx.appcompat.app.AppCompatActivity;

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);
        Bundle args = getIntent().getExtras();
        id = args.getInt("id");
        userID = args.getInt("userID");
    }

    public void ButtonClick(View v){
        new Thread(()-> {
            try {
                EditText inputBan = findViewById(R.id.editTextLength);
                String lengthString = inputBan.getText().toString();
                if (lengthString.length() > 0) {
                    int length = Integer.parseInt(lengthString);
                    if (RequestCreater.banUser(id, userID, length)) {
                        setResult(RESULT_OK);
                        MessageMaker.finishOnSuccess(this, "Пользователь
                            успешно заблокирован");
                    } else {
                        MessageMaker.error(this, "Не удалось заблокировать
                            пользователя");
                    }
                } else {
                    MessageMaker.error(this, "Недопустимый формат длины
                        блокировки");
                }
            } catch (Exception e) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
                finish();
            }
        }).start();
    }
}
```

Листинг Е.2 – Файл BookInfoActivity.java

```
package com.myscilib;

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

import android.content.Intent;
import android.net.Uri;
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 int bookID;
    private BookPage book;

    public static boolean canMakeComment(int userID, Comment[] comments) {
        if (comments == null) {
            return true;
        }
        else{
            for (Comment comment : comments) {
                if (comment.getUserID() == userID) {
                    return false;
                }
            }
        }
        return true;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_book_info);
        new Thread(()-> {
            try {
                id = getIntent().getExtras().getInt("id");
                boolean isAdmin =
                     getIntent().getExtras().getBoolean("isAdmin");
                bookID = getIntent().getExtras().getInt("bookID");
                ImageButton delButton = findViewById(R.id.delBookButton);
                if (!isAdmin) {
                    delButton.setVisibility(View.INVISIBLE);
                }
                book = RequestCreater.getBook(bookID);
                if (book != null) {
                    runOnUiThread(() -> {
                        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);
                        StringBuilder authors = new StringBuilder();
                        Author[] authorList = book.getAuthors();
                        for (Author author : authorList) {
                            authors.append(author.getName()).append(" ");
                        }
                        editAuthors.setText(authors.toString());
                        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: " + avgRate);
                    } else {
                        editRate.setText("Средняя оценка: нет");
                    }
                    TextView editTags = findViewById(R.id.textViewTags);
                    Tag[] tagList = book.getTags();
                    StringBuilder tags = new StringBuilder("Теги: ");
                    for (Tag tag : tagList) {
                        tags.append(tag.getName()).append(" ");
                    }
                    editTags.setText(tags.toString());
                    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 (!canMakeComment(id, comments)) {
                            Button commentButton =
                                findViewById(R.id.buttonReviewAdd);
                            commentButton.setVisibility(View.INVISIBLE);
                        }
                        FragmentManager manager =
                            getSupportFragmentManager();
                        for (Comment comment : comments) {
                            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",
                                 comment);
                            fragment.setArguments(fragmentArgs);
                            transaction.add(R.id.commentsContainer,
                                fragment);
                            transaction.commit();
                        }
                    }
                });
            }
            else {
                MessageMaker.errorToast(this,
                    "Ошибка подключения к сети");
                finish();
            }
        } catch (Exception e) {
            MessageMaker.errorToast(this, "Ошибка подключения к сети");
            finish();
        }
    }).start();
```

```
    }

    public void delButtonClick(View v){
        new Thread(()-> {
            try {
                if (RequestCreater.deleteBook(bookID)) {
                    MessageMaker.finishOnSuccess(this, "Книга удалена");
                } else {
                    MessageMaker.error(this,
                        "Не удалось удалить книгу. Повторите попытку позже");
                }
            } catch (Exception e) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
            }
        }).start();
    }

    public void downloadClick (View v) {
        new Thread(()-> {
            try {
                if (RequestCreater.downloadBook(id, bookID)) {
                    String fileURI = String.format(
                        "http://10.0.2.2/MyScienceLibrary/%s",
                        book.Info.getFileLink()
                    );
                    startActivity(new Intent(Intent.ACTION_VIEW,
                       Uri.parse(fileURI)));
                } else {
                    MessageMaker.error(this, "Не удалось загрузить книгу.
                       Повторите попытку позже");
                }
            } catch (Exception e) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
            }
        }).start();
    }

    public void addReviewClick(View v){
        Intent intent = new Intent(this, NewReviewActivity.class);
        intent.putExtra("id", id);
        intent.putExtra("bookID", bookID);
        startActivityForResult(intent, 1);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
        Intent data){
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1 && resultCode == RESULT_OK){
            recreate();
        }
    }
}
```

Листинг Е.3 – Файл MainActivity.java

```
package com.myscilib;

import androidx.appcompat.app.AppCompatActivity;

import android.view.View;
import android.os.Bundle;
import android.content.Intent;
import android.widget.EditText;
public class MainActivity extends AppCompatActivity {
```

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

    public void loginClick(View v) {
        new Thread(()-> {
            try {
                EditText loginInput = findViewById(R.id.editTextLogin);
                EditText passwordInput = findViewById(R.id.editTextPassword);
                String login = loginInput.getText().toString();
                String password = passwordInput.getText().toString();
                User userAcc = RequestCreater.getUserByLoginPassword
                    (login, password);
                if (userAcc == null) {
                    MessageMaker.error(this, "Неверный логин или пароль");
                } else {
                    if (userAcc.getBannedUntil() != null) {
                        MessageMaker.error(this,
                            "К сожалению, ваш аккаунт заблокирован");
                    } 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) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
            }
        }).start();
    }

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

Листинг Е.4 – MainPageActivity.java

```
package com.myscilib;

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

import java.util.Arrays;
```

```
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){
        new Thread(()-> {
            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.getSelectedItemPosition();
                if (publisherID == 0) {
                    publisherID = null;
                }
                BookPanel[] books = RequestCreater.searchBooks(name,
```

```
                searchAuthorIDs, genreID, publisherID, searchTagIDs,
                pubYear);
            if (books != null && books.length > 0) {
                Spinner orderChoice = findViewById(R.id.spinnerSortBy);
                int order = orderChoice.getSelectedItemPosition();
                switch (order) {
                    case 1:
                        Arrays.sort(books, (book1, book2) ->
                             book1.getInfo().getName().compareTo
                                (book2.getInfo().getName()));
                        break;
                    case 2:
                        Arrays.sort(books, (book1, book2) -> {
                            Integer count2 =
                                book2.getInfo().getDownloadCount();
                            return count2.compareTo
                                (book1.getInfo().getDownloadCount());
                        });
                        break;
                    case 3:
                        Arrays.sort(books, (book1, book2) -> {
                            Double rate2 = book2.getInfo().getAvgRate();
                            return rate2.compareTo
                                (book1.getInfo().getAvgRate());
                        });
                        break;
                    case 4:
                        Arrays.sort(books, (book1, 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) {
            MessageMaker.error(this,
                "В процессе поиска произошла ошибка");
        }
    }).start();
}

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){
    super.onActivityResult(requestCode, resultCode, data);
```

```
        if (resultCode == RESULT_OK) {
            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");
            }
        }
    }
}
```

Листинг Е.5 – Файл NewBookActivity.java

```
package com.myscilib;

import androidx.appcompat.app.AppCompatActivity;

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

import java.io.FileNotFoundException;
import java.io.InputStream;

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

    @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) {
        new Thread(() -> {
            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();
            if (allReady(name, pubYear, genreID, publisherID)) {
                if (uri != null) {
                    InputStream input = getContentResolver().
                        openInputStream(uri);
                    if (RequestCreater.uploadBook(name, authorIDs,
                        genreID, publisherID, tagIDs, pubYear,
                        preview, input)) {
                        MessageMaker.finishOnSuccess(this,
                            "Книга добавлена");
                    } else {
                        MessageMaker.error(this, "Не удалось добавить
                            книгу. Повторите попытку позже");
                    }
                } else {
                    MessageMaker.error(this, "Файл книги не указан или
                        указан неверно");
                }
            } else {
                MessageMaker.error(this, "Должны быть введены все
                    требуемые данные. Кроме того,год издания должен
                    находиться в промежутке от 1980 до 2021");
            }
        } catch (NumberFormatException e) {
            MessageMaker.error(this, "Недопустимое значение года
                издания");
        }
        catch (FileNotFoundException e){
            MessageMaker.error(this, "Не удалось открыть файл");
        }
        catch (Exception e) {
            MessageMaker.errorToast(this, "Ошибка подключения к сети");
        }
    }).start();
}

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);
}

public void editPathClick(View v){
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("application/pdf");
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    startActivityForResult(intent, 3);
}

@Override
public void onActivityResult(int requestCode, int resultCode,
    Intent data){
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == RESULT_OK) {
        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 if (requestCode == 3) {
                TextView editPath = findViewById(R.id.editPath);
                uri = data.getData();
                editPath.setText(uri.getPath());
            }
        }
    }
}
```

Листинг Е.6 – Файл NewPasswordActivity.java

```
package com.myscilib;

import androidx.appcompat.app.AppCompatActivity;

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) {
        new Thread(() -> {
            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 (RequestCreater.changePassword
                            (id, oldPwd, newPwd)) {
                            MessageMaker.finishOnSuccess(this,
                                "Пароль успешно изменен");
                        } else {
                            MessageMaker.error(this,
                                "Исходный пароль неверен");
                        }
                    } else {
                        MessageMaker.error(this, "Недопустимый новый пароль.
                            Каждый пароль должен включать в себя цифры и
                            буквы, длина - от 8 до 20 символов");
                    }
```

```
                } else {
                    MessageMaker.error(this, "Пароли не совпадают");
                }
            } catch (Exception e) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
            }
        }).start();
    }
}
```

Листинг Е.7 – Файл NewReviewActivity.java

```
package com.myscilib;

import androidx.appcompat.app.AppCompatActivity;

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) {
        new Thread(()-> {
            try {
                Spinner rateSpinner = findViewById(R.id.spinnerRate);
                int rate = rateSpinner.getSelectedItemPosition() + 1;
                EditText commentInput = findViewById(R.id.editTextReview);
                String comment = commentInput.getText().toString();
                if (RequestCreater.uploadReview(id, bookID, rate, comment)) {
                    setResult(RESULT_OK);
                    MessageMaker.finishOnSuccess(this,
                        "Отзыв успешно добавлен");
                } else {
                    MessageMaker.error(this, "Не удалось одобавить отзыв.
                        Попробуйте еще раз");
                }
            } catch (Exception e) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
            }
        }).start();
    }
}
```

Листинг Е.8 – Файл RegisterActivity.java

```
package com.myscilib;

import android.content.Intent;
import android.os.Bundle;
```

```
import android.view.View;
import android.widget.EditText;

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 !='_'){
                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) {
        new Thread(()-> {
```

```
            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 = RequestCreater.createNewUser(newLogin,
                        newPwd);
                    if (userAcc == null) {
                        MessageMaker.error(this, "Данный логин уже занят");
                    } 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 {
                    MessageMaker.error(this, "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");
                }
            } catch (Exception e) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
            }
        }).start();
    }
}
```

Листинг Е.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.TextView;

import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class UserInfoActivity extends AppCompatActivity {

    private int id;
    private int userID;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_info);
        new Thread(() -> {
            try {
                id = getIntent().getExtras().getInt("id");
                boolean isAdmin = getIntent().getExtras().
                    getBoolean("isAdmin");
                userID = getIntent().getExtras().getInt("userID");
```

```
                User userInfo = RequestCreater.getUserByID(userID);
                if (userInfo != null) {
                    boolean isBannable = isAdmin &&
                        (userInfo.getBannedUntil() == null) &&
                        (!userInfo.getRole().equals("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) {
                MessageMaker.errorToast(this, "Ошибка подключения к сети");
                finish();
            }
        }).start();
    }
    public void buttonClick(View v) {
        Intent intent = new Intent(this, BanActivity.class);
        intent.putExtra("id", id);
        intent.putExtra("userID", userID);
        startActivityForResult(intent, 1);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
        Intent data){
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1 && resultCode == RESULT_OK){
            recreate();
        }
    }
}
```

Листинг Е.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;
import android.view.View;
import android.widget.TextView;

public class SearchResultsActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_results);
        Bundle args = getIntent().getExtras();
        int id = args.getInt("id");
        boolean isAdmin = args.getBoolean("isAdmin");
        if (savedInstanceState == null) {
            BookPanel[] results = (BookPanel[]) args.get("results");
            if (results != null && results.length > 0) {
                TextView noResultsText = findViewById(R.id.textNoResults);
                noResultsText.setVisibility(View.GONE);
                FragmentManager manager = getSupportFragmentManager();
                for (BookPanel result : results) {
                    FragmentTransaction transaction =
                        manager.beginTransaction();
                    BookFragment fragment = new BookFragment();
                    Bundle fragmentArgs = new Bundle();
                    fragmentArgs.putInt("id", id);
                    fragmentArgs.putBoolean("isAdmin", isAdmin);
                    fragmentArgs.putSerializable("book", result);
                    fragment.setArguments(fragmentArgs);
                    transaction.add(R.id.resultsContainer, fragment);
                    transaction.commit();
                }
            }
        }
    }
}
```

Листинг Е.11 – RequestCreater.java

```
package com.myscilib;

import com.google.gson.Gson;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;

public class RequestCreater {
    public static User getUserByLoginPassword(String login, String password)
            throws IOException
    {
        User userAcc = null;
        String requestURI = String.format("http://10.0.2.2/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 IOException
    {
        String requestURI = "http://10.0.2.2/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 IOException
    {
        String requestURI = String.format("http://10.0.2.2/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();
        return gson.fromJson(json, BookPanel[].class);
    }
    else {
        return null;
    }
}
public static BookPanel[] searchBooks (String name, int[] authorIDs,
    Integer genreID, Integer publisherID, int[] tagIDs, Integer pubYear)
        throws IOException
{
    String requestURI = "http://10.0.2.2/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();
            return gson.fromJson(json, BookPanel[].class);
        }
        else{
            return null;
        }
    }

    public static Comment[] getMyComments (int id)
            throws IOException
    {
        String requestURI = String.format("http://10.0.2.2/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 (Comment comment : comments) {
                comment.deserializeCreationTime();
            }
            return comments;
        }
        else{
            return null;
        }
    }

    public static boolean uploadBook (String name, int[] authorIDs,
        int genreID, int publisherID, int[] tagIDs, int pubYear,
        String preview, InputStream fileInput)
            throws IOException
    {
        String boundary = "----****hhghgfgfhgghgh7788678677687*";
        String requestURI = "http://10.0.2.2/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=\"temp.pdf\"\r\n");
        out.writeBytes("Content-Type: application/pdf\r\n");
        out.writeBytes("\r\n");
        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();
        return responseCode == HttpURLConnection.HTTP_CREATED;
    }
```

```
public static boolean downloadBook(int userID, int bookID)
        throws IOException
{
    String requestURI = "http://10.0.2.2/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();
    return responseCode == HttpURLConnection.HTTP_CREATED ||
        responseCode == HttpURLConnection.HTTP_OK;
}

public static boolean deleteBook(int id)
        throws IOException
{
    String requestURI = String.format("http://10.0.2.2/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 IOException
{
    String requestURI = "http://10.0.2.2/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 IOException
```

```
{
    String requestURI = String.format("http://10.0.2.2/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 (Comment comment : comments) {
            comment.deserializeCreationTime();
        }
        return book;
    }
    else{
        return null;
    }
}

public static boolean deleteReview (int id)
        throws IOException
{
    String requestURI = String.format("http://10.0.2.2/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 IOException
{
    String requestURI = "http://10.0.2.2/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 IOException
{
    String requestURI = String.format("http://10.0.2.2/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 IOException
{
    String requestURI = "http://10.0.2.2/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 IOException
{
    String requestURI = "http://10.0.2.2/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, (t1, t2) -> {
                Integer id1 = t1.getId();
                return id1.compareTo(t2.getId());
            });
            return tags;
        }
        else{
            return null;
        }
    }

    public static Author[] getAuthors()
            throws IOException
    {
        String requestURI = "http://10.0.2.2/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, (a1, a2) -> {
                Integer id1 = a1.getId();
                return id1.compareTo(a2.getId());
            });
            return authors;
        }
        else{
            return null;
        }
    }

}
```

Приложение Ж

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СЕРВЕРНОЙ ЧАСТИ

(рекомендуемое)

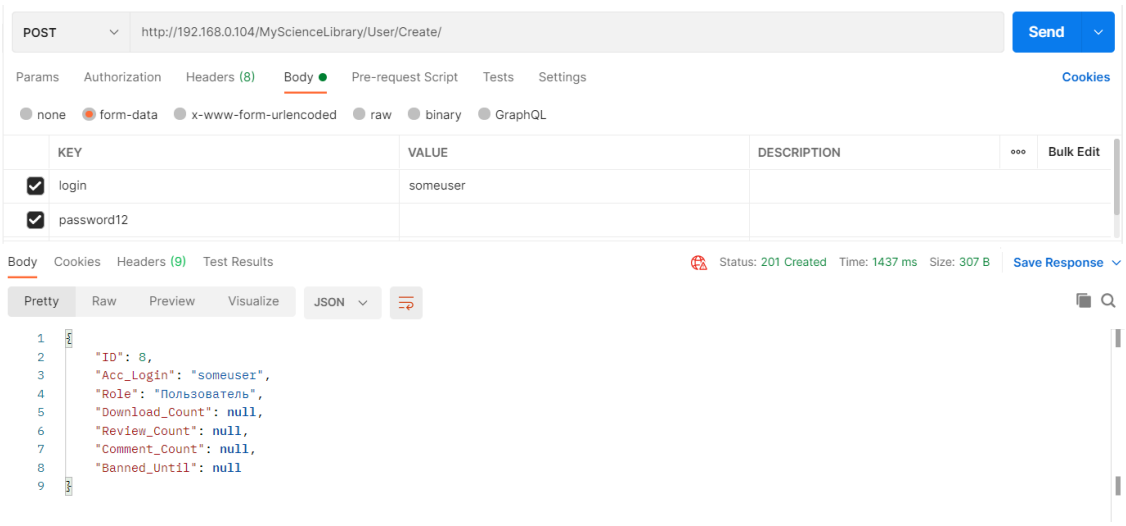


Рисунок Ж.1 – Добавление нового пользователя

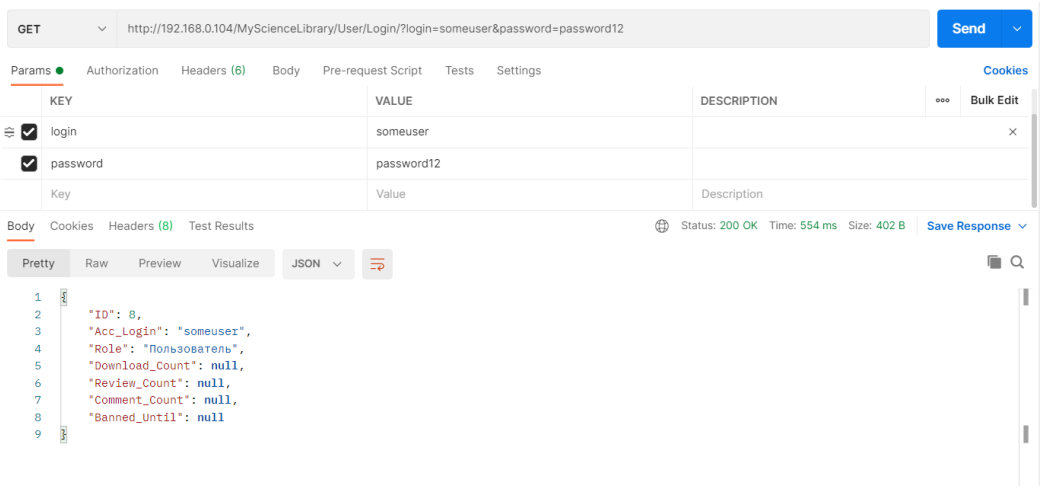


Рисунок Ж.2 – Вход существующего пользователя в систему

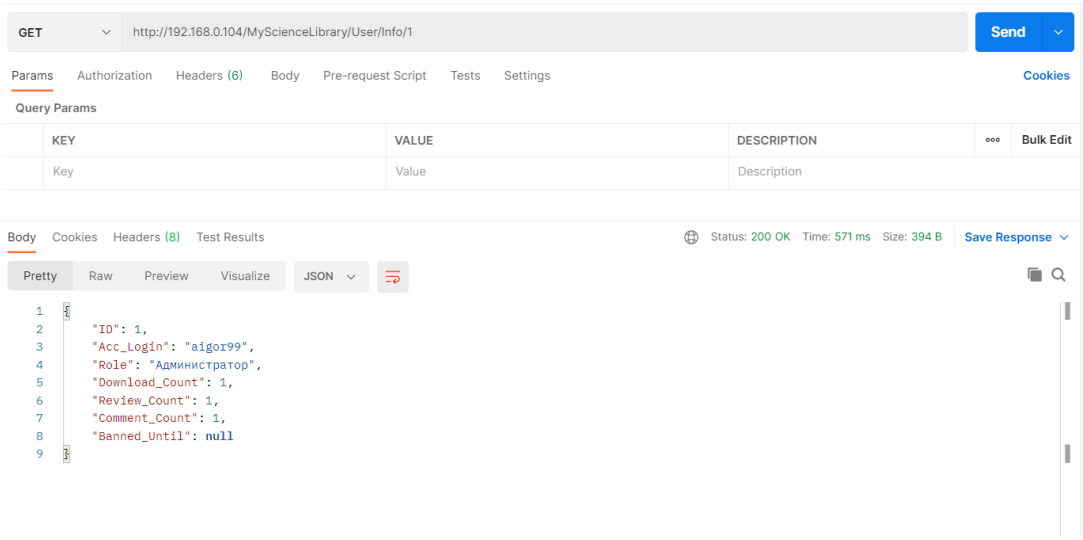


Рисунок Ж.3 – Получение информации о пользователе

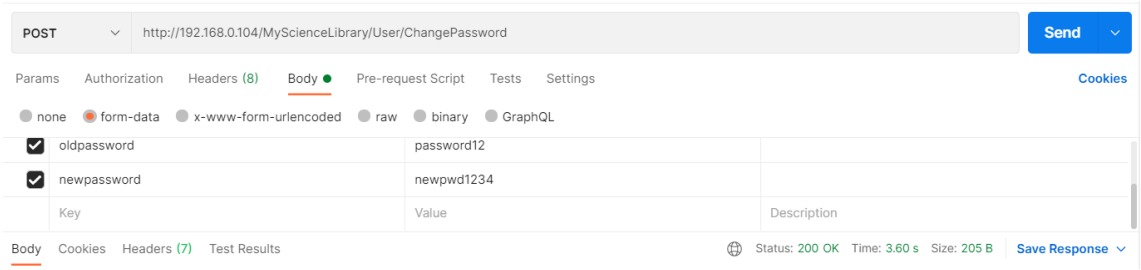


Рисунок Ж.4 – Изменение пароля пользователя

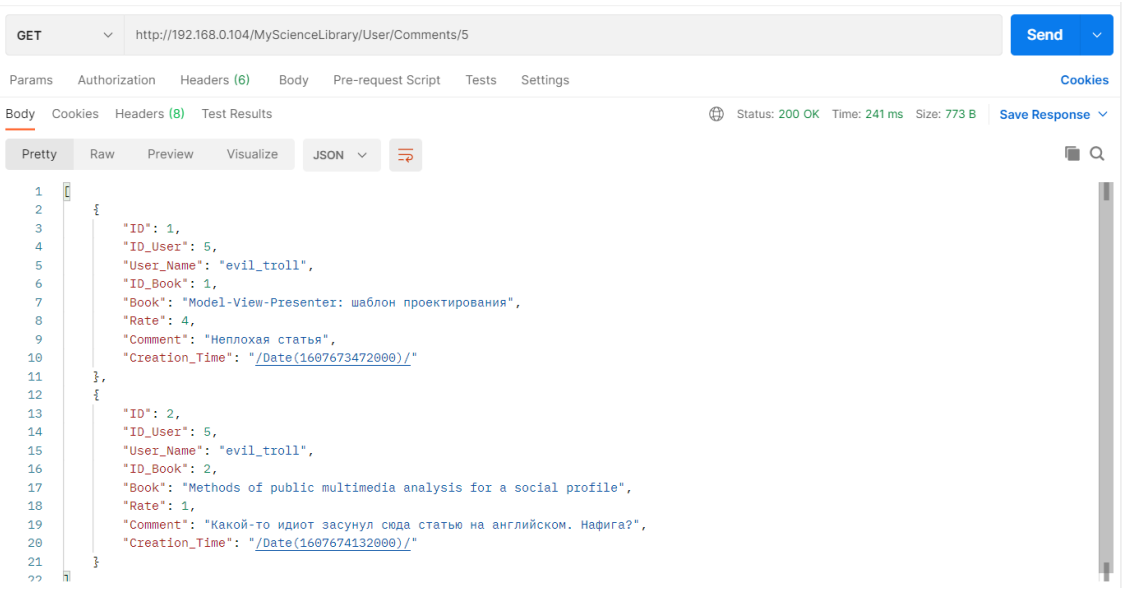


Рисунок Ж.5 – Получение списка комментариев пользователя

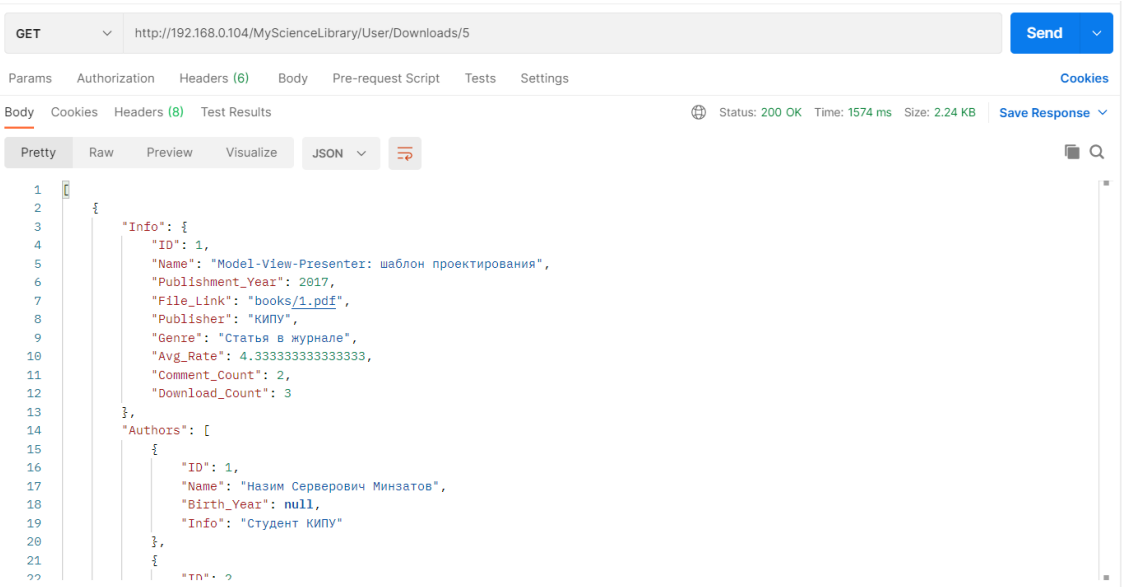


Рисунок Ж.6 – Получение списка загрузок пользователя

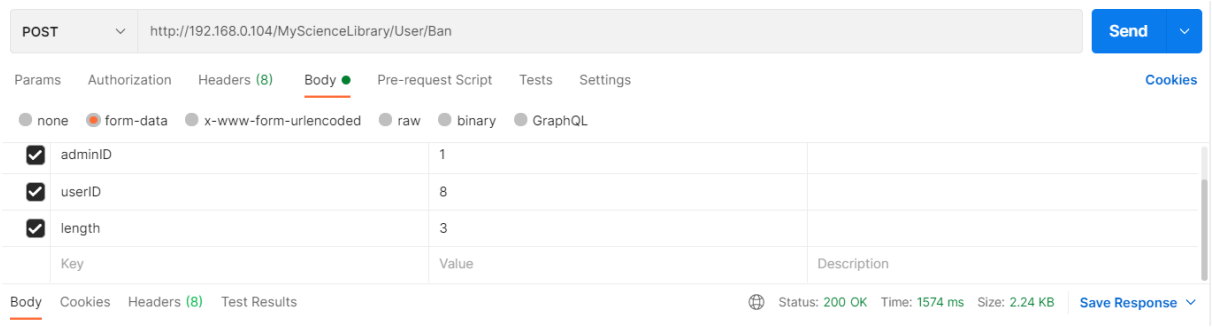


Рисунок Ж.7 – Блокировка пользователя

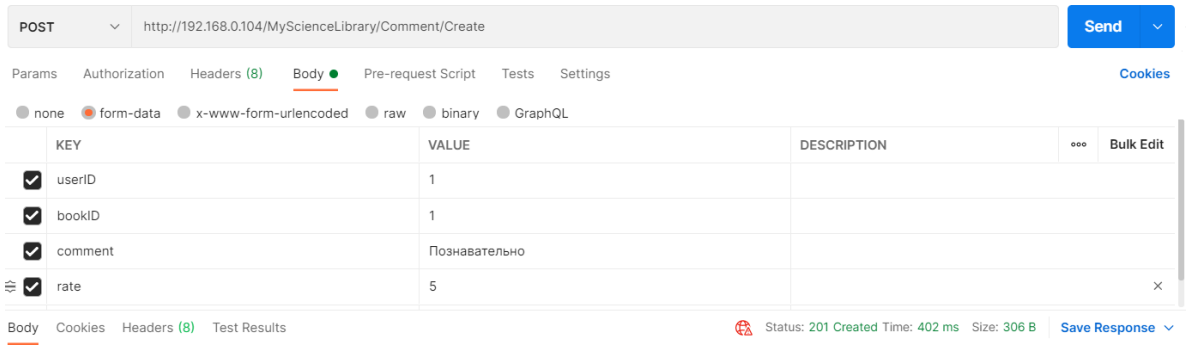


Рисунок Ж.8 – Добавление отзыва

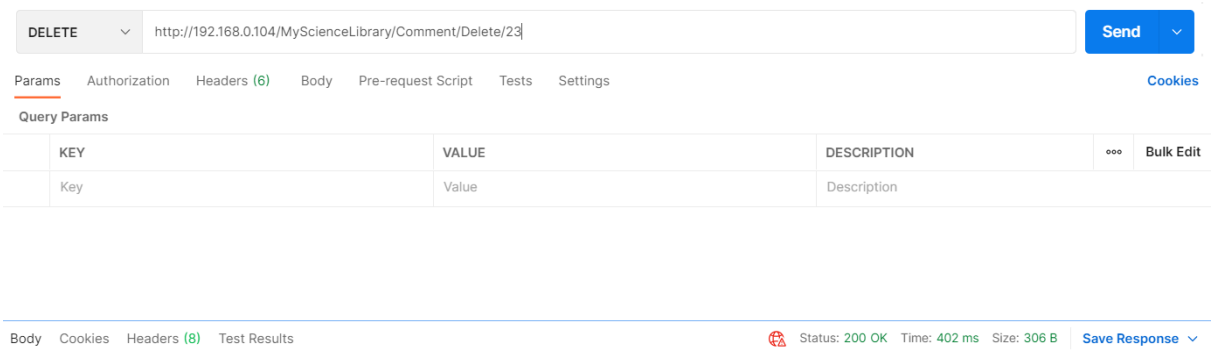


Рисунок Ж.9 – Удаление отзыва

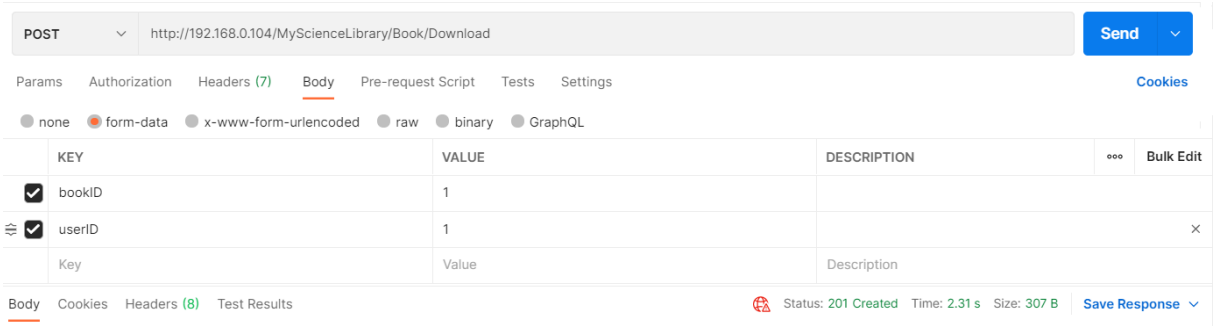


Рисунок Ж.10 – Сохранение информации о загрузке

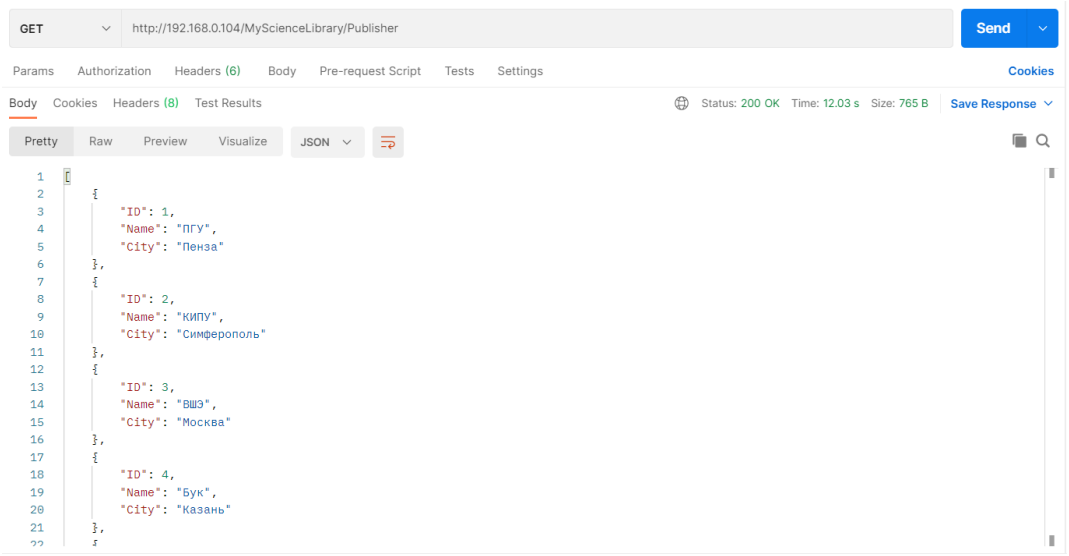


Рисунок Ж.11 – Получение списка издательств

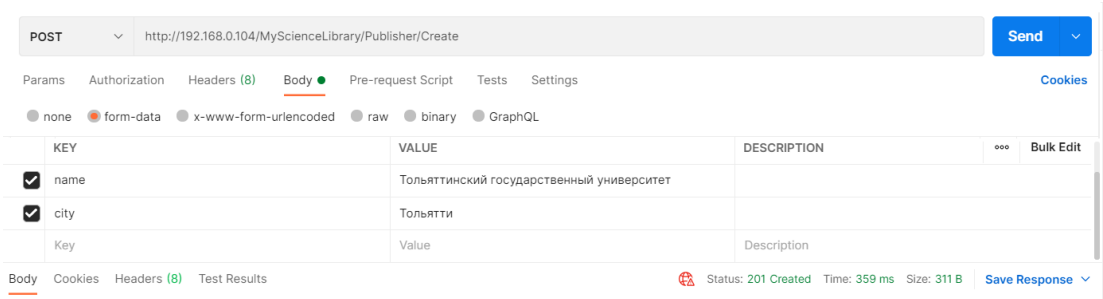


Рисунок Ж.12 – Добавление нового издательства

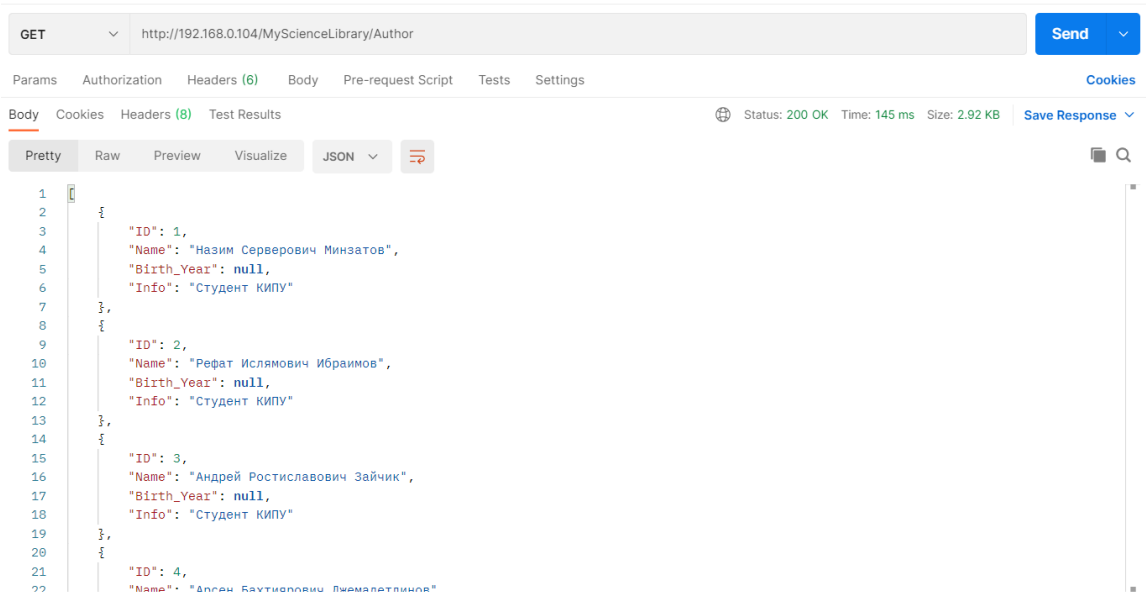


Рисунок Ж.13 – Получение списка авторов

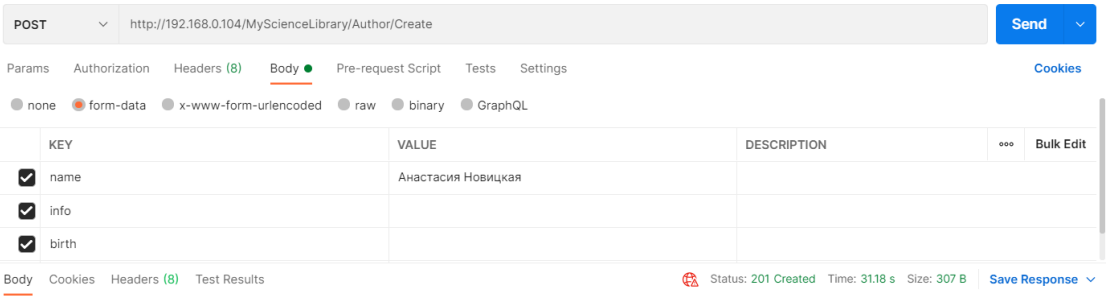


Рисунок Ж.14 – Добавление нового автора

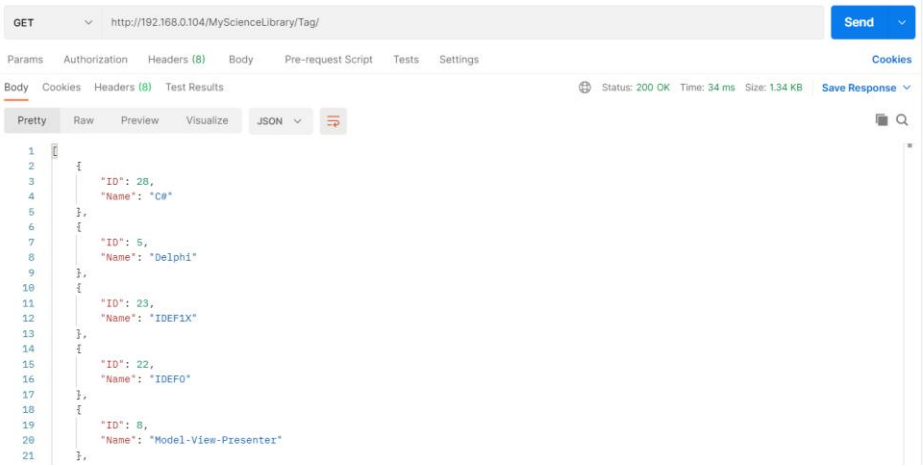


Рисунок Ж.15 – Получение списка тегов

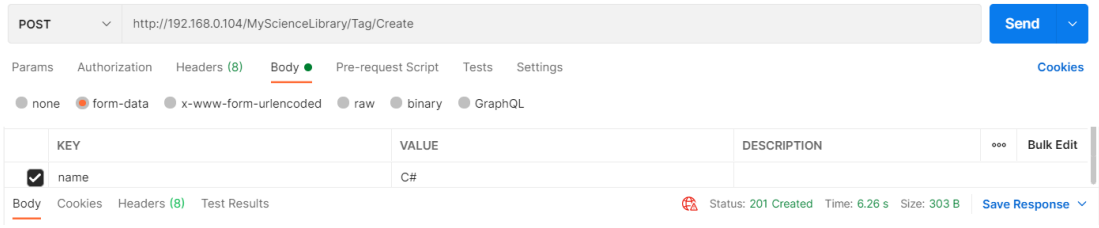


Рисунок Ж.16 – Добавление нового тега

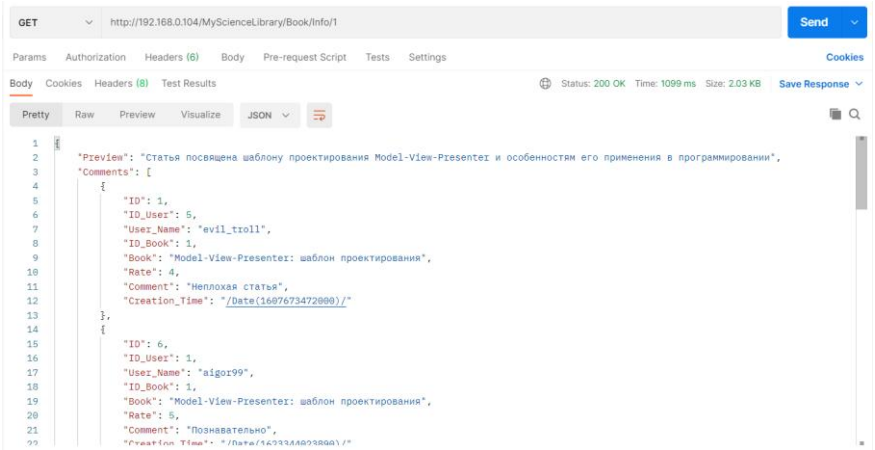


Рисунок Ж.17 – Получение информации о книге

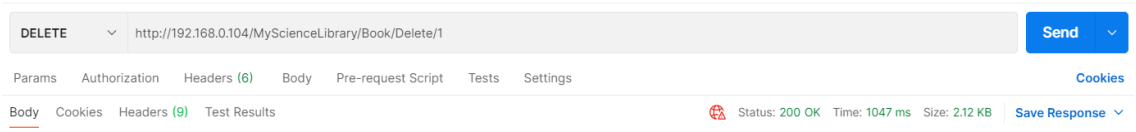


Рисунок Ж.18 – Удаление книги

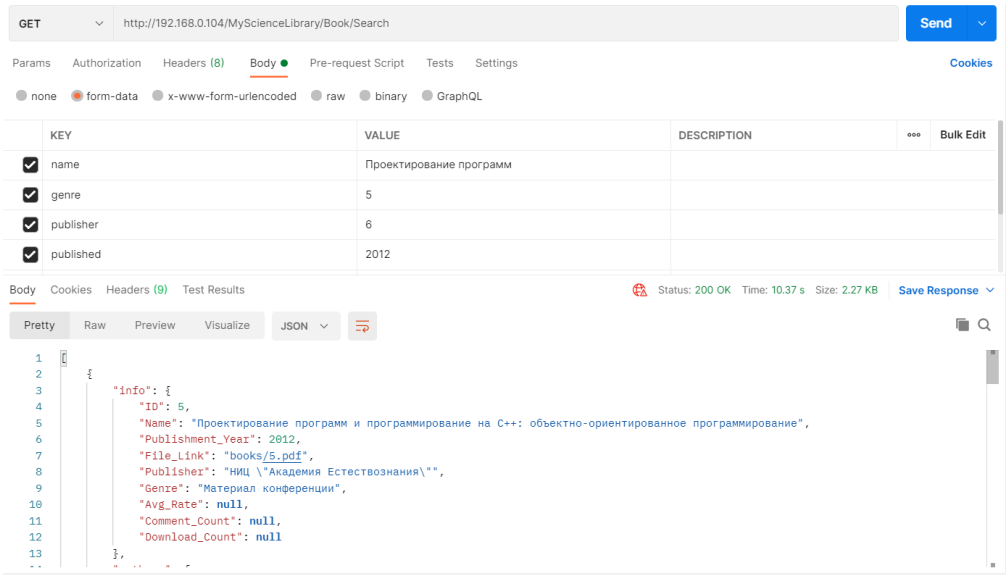


Рисунок Ж.19 – Поиск книг

Приложение И

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ КЛИЕНТСКОЙ ЧАСТИ

(рекомендуемое)

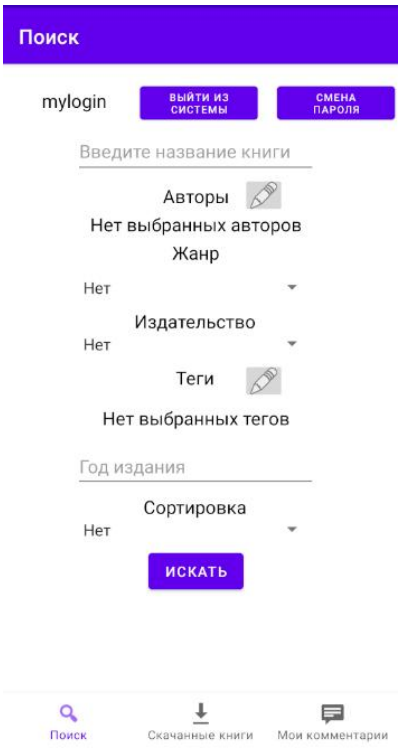


Рисунок И.1 – Успешная регистрация в системе

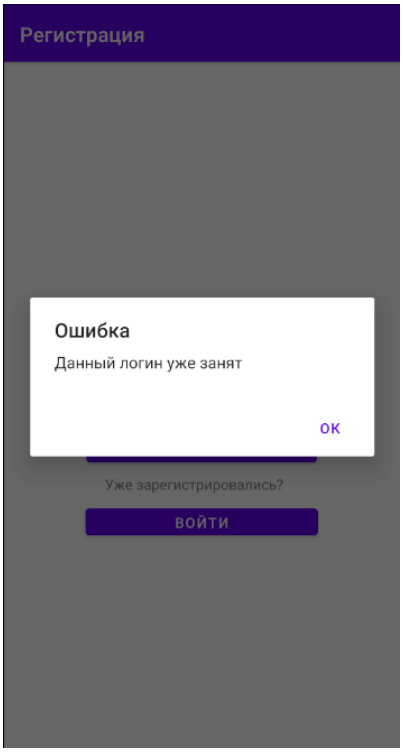


Рисунок И.2 – Логин занят

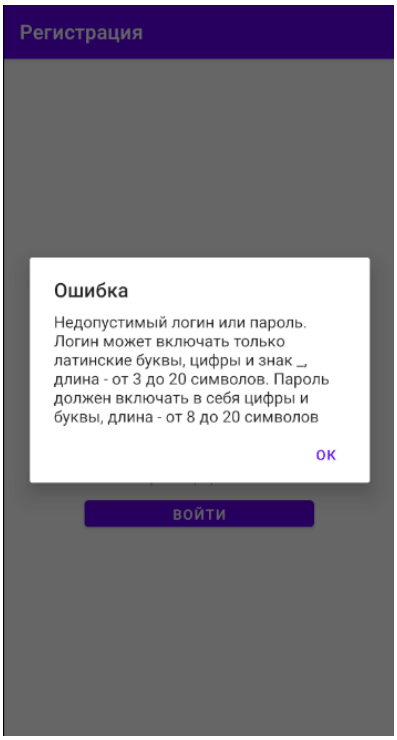


Рисунок И.3 – Недопустимые символы в логине и пароле

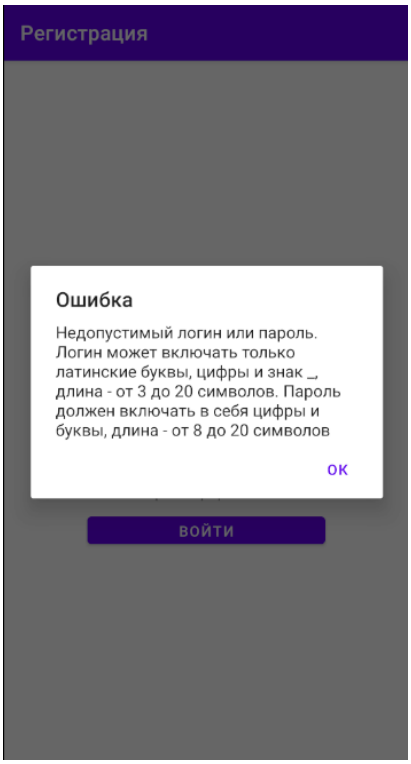


Рисунок И.4 – Логин и пароль недопустимой длины

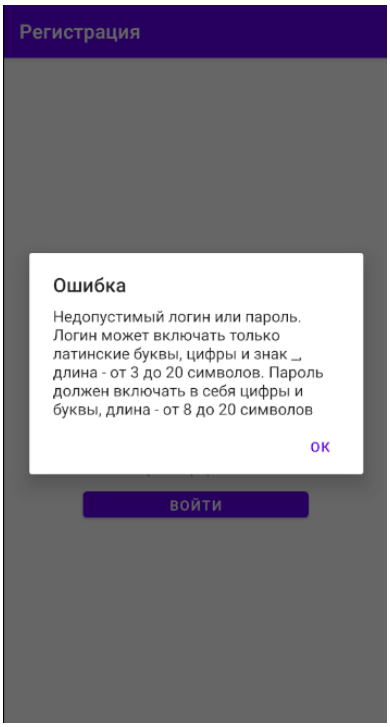


Рисунок И.5 – Пароль без букв

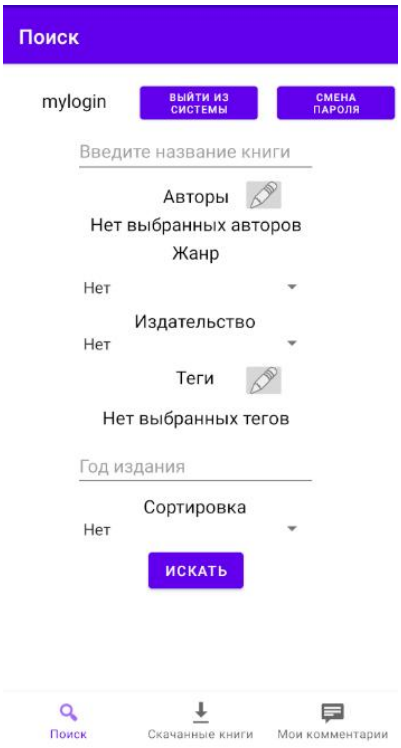


Рисунок И.6 – Успешная авторизация в системе

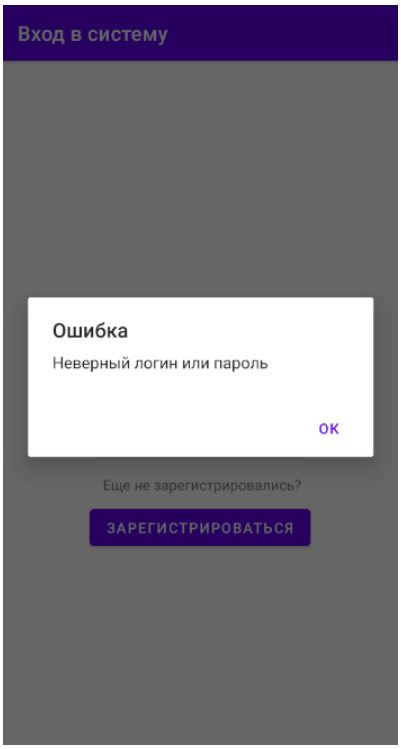


Рисунок И.7 – Неверный логин или пароль

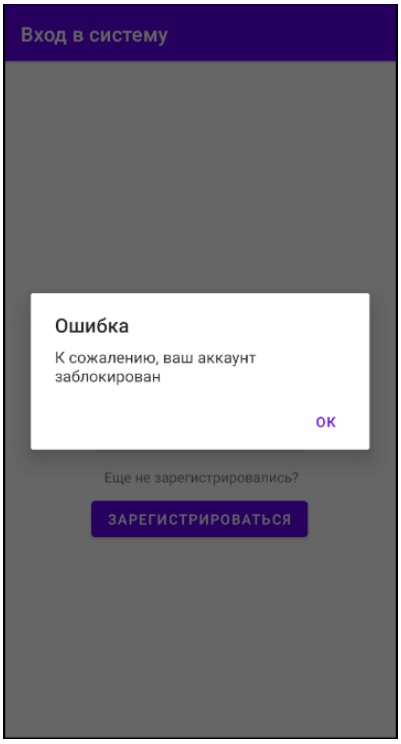


Рисунок И.8 – Аккаунт заблокирован

Рисунок И.9 – Поиск

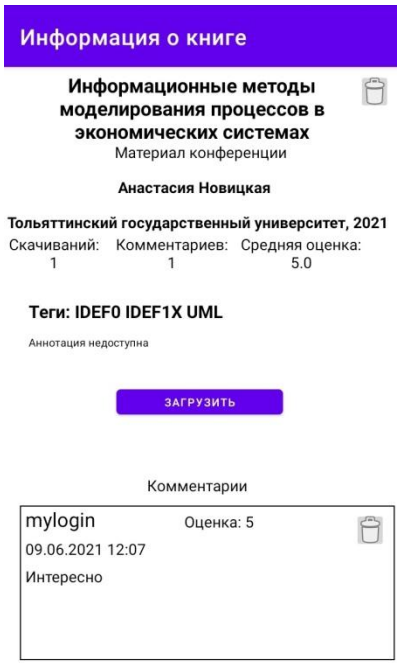


Рисунок И.10 – Оценка с отзывом

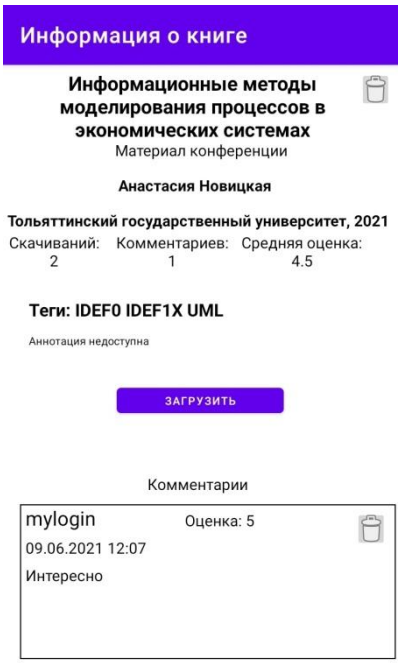


Рисунок И.11 – Оценка без отзыва

Рисунок И.12– Добавление книги

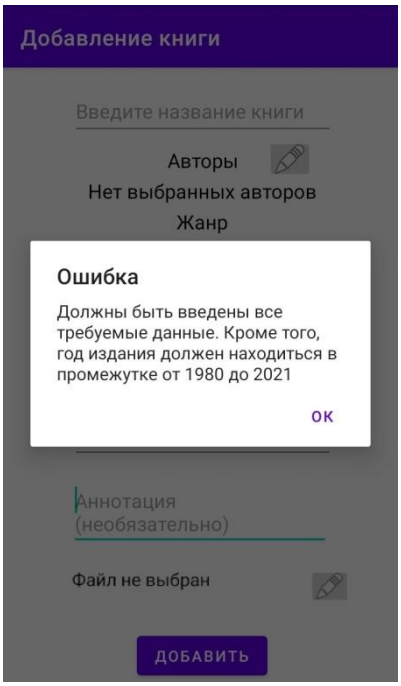


Рисунок И.13 – Неполные данные

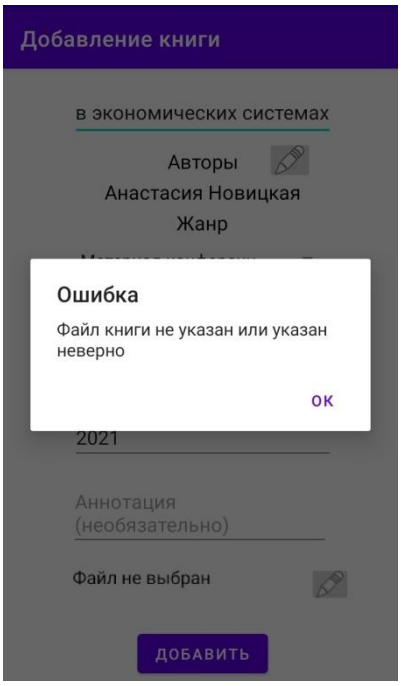


Рисунок И.14 – Нет файла

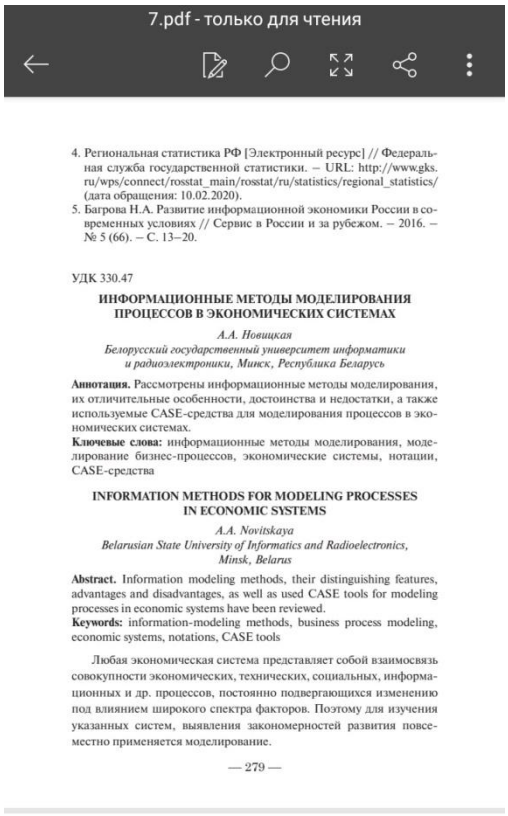


Рисунок И.15 – Скачивание книги

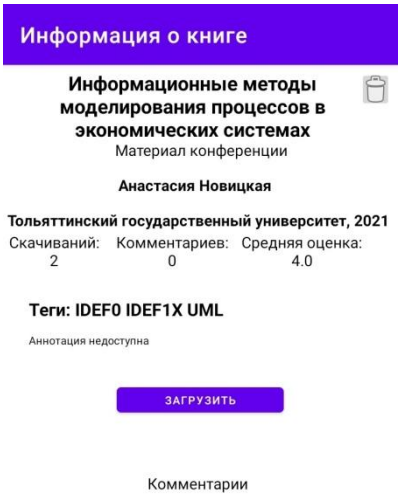


Рисунок И.16 – Удаление отзыва

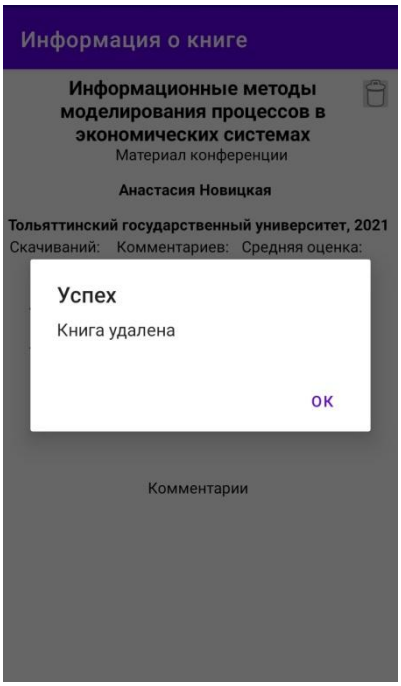


Рисунок И.17 – Удаление книги

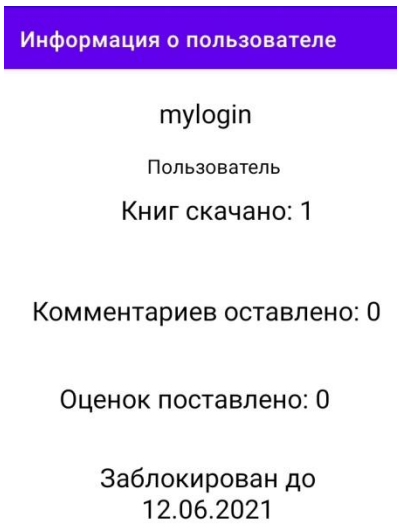


Рисунок И.18 – Блокировка пользователя

Приложение К

ИСХОДНЫЙ КОД ПРОГРАММЫ, РЕАЛИЗУЮЩЕЙ МОДИФИЦИРОВАННЫЙ АЛГОРИТМ ПОИСКА КНИГ

(рекомендуемое)

Листинг К.1 – Файл Program.cs

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

namespace ExtensionSearchTest
{
    class Program
    {
        static void Main(string[] args)
        {
            ExtensionSearcher searcher = new ExtensionSearcher();
            Console.Write("Введите запрос для поиска: ");
            string query = Console.ReadLine();
            List<string> result = searcher.Search(query);
            Console.WriteLine("Результаты поиска");
            foreach (string book in result)
            {
                Console.WriteLine(book);
            }
            Console.ReadKey();
        }
    }
}
```

Листинг К.2 – Файл ExtensionSearcher.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace ExtensionSearchTest
{
      public class ExtensionSearcher
      {
            private const int maxDistance = 2;
            private CommonMistakesList commonMistakes;
            private AcceptedAlphabet alphabet ;
            public ExtensionSearcher()
            {
                  commonMistakes = new CommonMistakesList();
                  alphabet = new AcceptedAlphabet();
            }
             public List<string> Search(string query)
            {
                  List<string> result = new List<string>();
                  StringBuilder stringBuilder = new
                        StringBuilder(query.ToLower());
                  //поиск исходного запроса
                  Populate(stringBuilder, result);
                  //запрос не найден - ищем варианты с 1-2 ошибками
                  if (maxDistance > 0 && result.Count == 0)
                        DoExtension(stringBuilder, 0, result,
                              maxDistance - 1);
                  return result;
            }
            private void DoExtension(StringBuilder stringBuilder, int start,
                        List<string> set, int depth)
            {
                  if (start >= stringBuilder.Length)
                        return;
```

```
char ch = stringBuilder[start];
stringBuilder.Remove(start, 1);

// Удаления
for (int i = start; i < stringBuilder.Length; i++)
{
      //поиск сгенерированного варианта
      Populate(stringBuilder, set);
      if (depth > 0)
            DoExtension(stringBuilder, i, set, depth - 1);
      char temp = stringBuilder[i];
      stringBuilder[i] = ch;
      ch = temp;
}
Populate(stringBuilder, set);
if (depth > 0)
      DoExtension(stringBuilder, stringBuilder.Length, set,
                  depth - 1);
stringBuilder.Append(ch);

// Перестановки
for (int i = start; i < stringBuilder.Length - 1; i++)
{
      char thisCh = stringBuilder[i];
      char nextCh = stringBuilder[i + 1];
      stringBuilder[i] = nextCh;
      stringBuilder[i + 1] = thisCh;
      //поиск сгенерированного варианта
      Populate(stringBuilder, set);
      if (depth > 0)
            DoExtension(stringBuilder, i + 2, set,
                  depth - 1);
      stringBuilder[i] = thisCh;
      stringBuilder[i + 1] = nextCh;
}

// Замены
for (int i = start; i < stringBuilder.Length; i++)
{
      ch = stringBuilder[i];
      //для каждого символа - рассматриваются только
      //наиболее вероятные замены
      foreach (char sym in
            commonMistakes.GetCommonReplacements(ch))
      {
            stringBuilder[i] = sym;
            //поиск сгенерированного варианта
            Populate(stringBuilder, set);
            if (depth > 0)
                  DoExtension(stringBuilder, i + 1, set,
                        depth - 1);
      }
      stringBuilder[i] = ch;
}

// Вставки
stringBuilder.Insert(start, char.MinValue);
for (int i = start; i < stringBuilder.Length; i++)
{
      foreach (char sym in alphabet.GetAlphabetForPos
            (stringBuilder, i))
      {
            stringBuilder[i] = sym;
            Populate(stringBuilder, set);
```

```
                        if (depth > 0)
                              DoExtension(stringBuilder, i + 1, set,
                                    depth - 1);
                  }
                  if (i + 1 < stringBuilder.Length)
                        stringBuilder[i] = stringBuilder[i + 1];
            }
            stringBuilder.Remove(stringBuilder.Length - 1, 1);
      }


      private void Populate(StringBuilder query, List<string> set)
      {
            string[] names = File.ReadAllLines("names.txt");
            foreach (string name in names)
            {
                  if (name.IndexOf(query.ToString(),
                        StringComparison.CurrentCultureIgnoreCase) >= 0
                        && !set.Contains(name))
                  {
                        set.Add(name);
                   }
            }
      }
}
}
```

Листинг К.3 – Файл CommonMistakesList.cs

```
using System.Collections.Generic;

namespace ExtensionSearchTest
{
      public class CommonMistakesList
      {
            private Dictionary<char, char[]> mistakesList;
            public CommonMistakesList()
            {
                  mistakesList = new Dictionary<char, char[]>()
                  {
                        //русские буквы
                        {'а', new char[]{ 'у', 'в', 'ч', 'с', 'м', 'п', 'е',
                              'к', 'о', 'я' } },
                        {'б', new char[]{ 'ь', 'ъ', 'л', 'д', '.', 'ю', ' ',
                              'ж', 'п' } },
                        {'в', new char[]{ 'у', 'к', 'а', 'с', 'ч', 'я', 'ы',
                              'ц', 'ф' } },
                        {'г', new char[]{ 'н', 'р', 'о', 'л', 'ш', '7', '6',
                              '5', 'к', 'х' } },
                        {'д', new char[]{ 'б', 'ю', 'ж', 'з', 'щ', 'ш', 'л',
                              'ь', 'т' } },
                        {'е', new char[]{ 'к', 'а', 'п', 'р', 'н', '6', '5',
                              '4', 'и', 'э', 'ё', 'о', 'я' } },
                        {'ё', new char[]{ 'к', 'а', 'п', 'р', 'н', '6', '5',
                              '4', 'и', 'э', 'е', 'о', 'я' } },
                        {'ж', new char[]{ 'ю', 'э', 'х', 'з', 'щ', 'д', 'б',
                              'ш' } },
                        {'з', new char[]{ 'щ', 'д', 'ж', 'э', 'х', '0', '9',
                              '8', 'с' } },
                        {'и', new char[]{ 'м', 'п', 'р', 'о', 'т', ' ', 'е',
                              'я', 'й', 'ы' } },
                        {'й', new char[]{ '1', '2', 'ц', 'ы', 'ф', 'и' } },
                        {'к', new char[]{ 'у', 'в', 'а', 'п', 'е', '5', '4',
                              '3', 'г' } },
```

```
{'л', new char[]{ 'ь', 'б', 'д', 'щ', 'ш', 'г', 'о'}},
{'м', new char[]{ 'и', 'р', 'п', 'а', 'с', ' ' } },
{'н', new char[]{ 'е', 'п', 'р', 'о', 'г', '6', '5'}},
{'о', new char[]{ 'т', 'ь', 'л', 'ш', 'г', 'н', 'р',
      'и', 'а', 'е', 'ё' } },
{'п', new char[]{ 'м', 'и', 'р', 'н', 'е', 'ё', 'к',
      'а', 'с', 'б' } },
{'р', new char[]{ 'и', 'т', 'о', 'г', 'н', 'е', 'ё',
      'п', 'м' } },
{'с', new char[]{ 'ч', 'в', 'а', 'п', 'м', ' ', ',',
      'з' } },
{'т', new char[]{ 'и', 'р', 'о', 'л', 'ь', 'ъ', ' ',
      'д' } },
{'у', new char[]{ 'к', 'а', 'в', 'ы', 'ц', '2', '3',
      '4', 'ю' } },
{'ф', new char[]{ 'я', 'ы', 'ц', 'й', 'в' } },
{'х', new char[]{ 'э', 'ж', 'з', '9', '0', 'г' } },
{'ц', new char[]{ 'у', 'в', 'ы', 'ф', 'й', '1', '2',
      '3' } },
{'ч', new char[]{ 'я', 'ы', 'в', 'а', 'с', ' ', ',',
      'ш' } },
{'ш', new char[]{ 'г', 'о', 'л', 'д', 'щ', '8', '7',
      '6', 'ж', 'ч' } },
{'щ', new char[]{ 'ш', 'л', 'д', 'ж', 'з', '9', '8',
      '7', 'ш' } },
{'ъ', new char[]{ 'т', 'о', 'л', 'д', 'б', '.', ' ',
      'ь' } },
{'ы', new char[]{ 'у', 'в', 'ч', 'я', 'ф', 'й', 'ц',
      'и' } },
{'ь', new char[]{ 'т', 'о', 'л', 'д', 'б', '.', ' ',
      'ъ' } },
{'э', new char[]{ 'ю', 'ж', 'з', 'х', 'е' } },
{'ю', new char[]{ 'б', 'д', 'ж', 'э', '.', 'у' } },
{'я', new char[]{ 'ч', 'в', 'ы', 'ф', ',', 'и', 'е',
      'а' } },
//английские буквы
{'a', new char[]{ 'z', 's', 'w', 'q' } },
{'b', new char[]{ 'v', 'g', 'h', 'j', 'n', ' ' } },
{'c', new char[]{ 'x', 'd', 'f', 'g', 'v', ' ' } },
{'d', new char[]{ 'e', 'r', 'f', 'c', 'x', 'z', 's'}},
{'e', new char[]{ 'r', 'd', 's', 'w', '2', '3', '4'}},
{'f', new char[]{ 'd', 'x', 'c', 'v', 'g', 't', 'r'}},
{'g', new char[]{ 'v', 'b', 'h', 'y', 't', 'f', 'c'}},
{'h', new char[]{ 'b', 'n', 'j', 'u', 'y', 'g', 'v'}},
{'i', new char[]{ 'u', 'j', 'k', 'o', '9', '8', '7'}},
{'j', new char[]{ 'n', 'm', 'k', 'i', 'u', 'h', 'b'}},
{'k', new char[]{ 'm', 'l', 'o', 'i', 'j', 'n' } },
{'l', new char[]{ 'p', 'o', 'k', 'm' } },
{'m', new char[]{ 'n', 'j', 'k', 'l', '.', ' ' } },
{'n', new char[]{ 'b', 'h', 'j', 'k', 'm', '.', ' '}},
{'o', new char[]{ 'i', 'k', 'l', 'p', '0', '9', '8'}},
{'p', new char[]{ 'o', 'l', '0', '9' } },
{'q', new char[]{ '1', '2', 'w', 's', 'a' } },
{'r', new char[]{ 'e', 'd', 'f', 't', 'e', '5', '4',
      '3' } },
{'s', new char[]{ 'e', 'd', 'x', 'z', 'a', 'w' } },
{'t', new char[]{ 'r', 'f', 'g', 'y', '6', '5', '4'}},
{'u', new char[]{ 'y', 'h', 'j', 'i', '8', '7', '6'}},
{'v', new char[]{ 'b', 'h', 'g', 'f', 'c', ' ' } },
{'w', new char[]{ 'e', 's', 'a', 'q', '1', '2', '3'}},
{'x', new char[]{ 'z', 's', 'd', 'f', 'c', ' ', ','}},
{'y', new char[]{ 't', 'g', 'h', 'u', '7', '6', '5'}},
{'z', new char[]{ 'x', 'd', 's', 'a', ' ', ',' } },
//цифры
{'0', new char[]{ '9', 'o', 'p', 'з', 'х' } },
```

```
                        {'1', new char[]{ '2', 'q', 'й', 'w', 'ц' } },
                        {'2', new char[]{ '1', 'q', 'й', 'w', 'ц', 'e', 'у',
                              '3' } },
                        {'3', new char[]{ '2', 'w', 'ц', 'e', 'у', 'r', 'к',
                              '4' } },
                        {'4', new char[]{ '3', 'e', 'у', 'r', 'к', 't', 'е',
                              '5' } },
                        {'5', new char[]{ '4', 'r', 't', 'e', 'у', 'н', '6'}},
                        {'6', new char[]{ '5', 't', 'у', 'н', 'u', 'г', '7'}},
                        {'7', new char[]{ '6', 'у', 'u', 'г', 'i', 'ш', 'щ',
                              '8' } },
                        {'8', new char[]{ '7', 'u', 'i', 'ш', 'o', 'щ', 'з',
                              '9' } },
                        {'9', new char[]{ '8', 'i', 'o', 'щ', 'p', 'з', 'х',
                              '0' } } ,
                        //прочие символы
                        {' ', new char[]{ 'z', 'x', 'ч', 'c', 'c', 'v', 'м',
                              'b', 'и', 'n', 'т', 'm', 'ь', 'ъ','б',',','.'}},
                        {',', new char[]{ 'z', 'x', 'я', 'ч', 'c', ' ' } },
                        {'.', new char[]{ 'n', 'm', 'ь', 'ъ', 'б', 'ю', ' '}},
                  };
            }
            public char[] GetCommonReplacements(char c)
            {
                  if (mistakesList.ContainsKey(c))
                        return mistakesList[c];
                  else
                        return null;
            }
      }
}
```

Листинг К.4 – Файл AcceptedAlphabet.cs

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

namespace ExtensionSearchTest
{
      public class AcceptedAlphabet
      {
            private const int engAlphabetSize = 26;
            private const int rusAlphabetSize = 32;
            private const int digitCount = 10;
            private char[] frequentEngChars;
            private char[] frequentRusChars;
            private char[] digits;
            public AcceptedAlphabet()
            {
                  char[] engChars = new char[engAlphabetSize];
                  int i = 0;
                  for (char c = 'a'; c <= 'z'; c++)
                  {
                        engChars[i] = c;
                        i++;
                  }
                   i = 0;
                  char[] rusChars = new char[rusAlphabetSize];
                  for (char c = 'a'; c <= 'я'; c++)
                  {
                        rusChars[i] = c;
                        i++;
                  }
```

```
        //наименее распространенные английские символы
        char[] rareEngChars = new char[] { 'j', 'q', 'x', 'z', 'k',
              'v', 'b', 'p', 'y' };
        // распространенные английские символы
        frequentEngChars = engChars.Except(rareEngChars).ToArray();
        //наименее распространенные русские символы
        char[] rareRusChars = new char[] { 'ъ', 'э', 'щ', 'ц', 'ф',
              'ш', 'ж', 'х', 'ю', 'ч', 'й', 'ь', 'ы', 'г', 'б','з'};
        // распространенные русские символы
        frequentRusChars = rusChars.Except(rareRusChars).ToArray();
        digits = new char[digitCount];
         i = 0;
        for (char c = '0'; c <= '9'; c++)
        {
              digits[i] = c;
              i++;
        }
}
public bool IsEnglish(char c)
{
       return c >= 'a' && c <= 'z';
}
public bool IsRussian(char c)
{
       return c >= 'a' && c <= 'я';
}
public char[] GetAlphabetForPos(StringBuilder sb, int pos)
{
      List<char> alphabet = new List<char>() { ' ' };
      if (pos == 0)
      {
            char nextSym = sb[pos + 1];
             if (IsEnglish(nextSym))
            {
                  alphabet.AddRange(frequentEngChars);
            }
            else if (IsRussian(nextSym))
            {
                  alphabet.AddRange(frequentRusChars);
            }
            else if(char.IsDigit(nextSym))
            {
                  alphabet.AddRange(digits);
            }
            if (!alphabet.Contains(nextSym))
            {
                  alphabet.Add(nextSym);
            }
      }
      else if (pos == sb.Length - 1)
      {
            char prevSym = sb[pos - 1];
            if (IsEnglish(prevSym))
            {
                  alphabet.AddRange(frequentEngChars);
            }
            else if (IsRussian(prevSym))
            {
                  alphabet.AddRange(frequentRusChars);
            }
            else if (char.IsDigit(prevSym))
            {
                  alphabet.AddRange(digits);
            }
             if (!alphabet.Contains(prevSym))
```

```
                        {
                              alphabet.Add(prevSym);
                        }
                  }
                  else
                  {
                        char prevSym = sb[pos - 1];
                        char nextSym = sb[pos + 1];
                        if (IsEnglish(prevSym) || IsEnglish(nextSym))
                        {
                              alphabet.AddRange(frequentEngChars);
                        }
                        if (IsRussian(prevSym) || IsRussian(nextSym))
                        {
                              alphabet.AddRange(frequentRusChars);
                        }
                        if (char.IsDigit(prevSym) || char.IsDigit(nextSym))
                        {
                              alphabet.AddRange(digits);
                        }
                        if (!alphabet.Contains(nextSym))
                        {
                              alphabet.Add(nextSym);
                        }
                        if (!alphabet.Contains(prevSym))
                        {
                              alphabet.Add(prevSym);
                        }
                  }
             return alphabet.ToArray();
            }
      }
}
```

Приложение Л

ДИПЛОМЫ ПОБЕДИТЕЛЯ МЕЖДУНАРОДНЫХ КОНКУРСОВ НАУЧНЫХ РАБОТ

(рекомендуемое)

Рисунок Л.1 – Диплом победителя XX Международного научно-исследовательского конкурса «Научные достижения и открытия 2021»

Рисунок Л.2 – Диплом лауреата I степени V Международного научно-исследовательского конкурса «Молодежный исследовательский потенциал»

Приложение М

СВИДЕТЕЛЬСТВО О ГОСУДАРСТВЕННОЙ РЕГИСТРАЦИИ ПРОГРАММЫ ДЛЯ ЭВМ

(рекомендуемое)

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023616960

Программа поиска книг в электронной библиотеке

Правообладатель: ***Ащеулов Игорь Олегович (RU)***

Авторы: ***Ащеулов Игорь Олегович (RU), Князев Виктор Николаевич (RU)***

Заявка № **2023614793**

Дата поступления **13 марта 2023 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***04 апреля 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Ю.С. Зубов