

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

Факультет

вычислительной техники

Кафедра

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 систем электронной коммерции**

Студент	_____	Тепин Константин Сергеевич
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	Князев В.Н.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролер	_____	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)
Рецензент	Начальник отдела ООО	Афонин А.Ю.
	_____	(фамилия, инициалы)
	(подпись, дата)	

НТП «Криптософт»
(должность, место работы)

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

ОС	Операционная система
ПО	Программное обеспечение
UML	Unified modeling language
RUP	Rational unified process
API	Application Programming Interface
ВАК	Высшая аттестационная комиссия
БД	База данных
СУБД	Система управления базами данных

Содержание

Введение.....	6
1 Анализ предметной области и планирование	14
1.1 Понятие и классификация систем электронной коммерции	14
1.2 Обзор аналогов	19
1.2.1 Анализ системы электронной коммерции «Chrono24».....	19
1.2.2 Анализ системы электронной коммерции «Pull&Bear».....	20
1.2.3 Анализ системы электронной коммерции «ShapShop»	22
1.2.4 Подведение итогов сравнения аналогов	23
1.3 Анализ возможностей повышения эффективности разработки систем электронной коммерции	24
1.4 Сетевое планирование процесса разработки.....	26
1.4.1 Структурное планирование.....	26
1.4.2 Календарное планирование.....	34
1.4.3 Подведение итогов сетевого планирования	37
1.5 Выбор технологий и инструментальных средств разработки	37
Выводы	39
2 Разработка системы электронной коммерции и модифицированного алгоритма аутентификации	40
2.1 Анализ требований.....	40
2.1.1 Анализ функциональных требований	40
2.1.2 Анализ нефункциональных требований	44
2.2 Онтологическое моделирование предметной области.....	45
2.3 Имитационное моделирование объекта исследования	46
2.4 Оценка адекватности модели.....	58
2.5 Концептуальное и логическое проектирование базы данных.....	60
2.6 Проектирование серверной части системы электронной коммерции	62
2.7 Проектирование клиентской части системы электронной коммерции...	68

2.8 Разработка модифицированного алгоритма аутентификации	75
2.8.1 Проектирование алгоритма аутентификации.....	75
2.8.2 Разработка улучшенного алгоритма шифрования.....	83
Выводы	87
3 Реализация системы электронной коммерции с модифицированным алгоритмом аутентификации	89
3.1 Реализация модифицированного алгоритма аутентификация	89
3.2 Эксперимент по проверке преимуществ модифицированного алгоритма аутентификации.....	89
3.3 Реализация и развертывание базы данных	93
3.4 Реализация серверной части системы электронной коммерции.....	94
3.5 Реализация клиентской части системы электронной коммерции.....	96
3.6 Тестирование системы электронной коммерции.....	99
3.6.1 Нагрузочное тестирование	99
3.6.2 Функциональное тестирование.....	100
3.6.3 Модульное тестирование.....	102
3.7 Метрики кода.....	104
Выводы	108
Заключение	109
Список использованных источников	112
Приложение А. Глоссарий	120
Приложение Б. Исходный код имитационной модели	122
Приложение В. Описание сущностей базы данных	126
Приложение Г. Исходный код модифицированного алгоритма аутентификации.....	133
Приложение Д. Скрипты создания базы данных.....	138
Приложение Е. Исходный код серверной части системы электронной коммерции.....	143
Приложение Ж. Исходный код клиентского мобильного приложения	155
Приложение И. Результаты функционального тестирования системы	

электронной коммерции	174
Приложение К. Дипломы победителя международных конкурсов научных работ	182
Приложение Л. Свидетельство о государственной регистрации программы для ЭВМ	186

Введение

Актуальность темы. В современном мире наблюдается тенденция к постепенной цифровизации всех отраслей деятельности человека и всех сфер жизни общества. Данная тенденция обусловлена тем, что внедрение новых информационных технологий позволяет повысить эффективность практически любого экономического, социального или бытового процесса. Осознавая данный факт, правительство Российской Федерации в 2017 году утвердило программу «Цифровая экономика Российской Федерации». Данный документ определяет стратегию развития «...цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах социально-экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост» [1].

Еще одним документом, определяющим развитие информационных технологий в Российской Федерации, является указ Президента Российской Федерации «О стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы» от 9 мая 2017 года [2]. Документ также подчеркивает важность и необходимость следованию современным тенденциям развития автоматизированных и цифровых технологий.

Происходящее развитие и совершенствование информационных технологий поводит к постоянному росту доли интернет-торговли. Так, анализ мирового рынка показывает, что в 2020 году на долю интернет покупок приходилось 11,2% всего товарооборота в мире. Что на 0,7% больше аналогичного показателя в предыдущем году [3]. Данный способ ведения торговли имеет ряд преимуществ, как для покупателей, так и для торговых предприятий. Например, процесса покупок и возможность заказа любого представленного товара независимо от географического положения магазина и покупателя позволяет автоматизация значительно повысить количество покупок и, как следствие, экономические показатели торгового предприятия.

Системы электронной коммерции используются обширным классом предприятий: начиная от крупных сетей, специализирующихся на торговле, заканчивая малыми компаниями и индивидуальными предпринимателями. Предприятия последнего типа, ввиду ограниченности бюджета, не могут себе позволить заказы на разработку и обслуживание сложных информационных систем. Однако их потребность в автоматизации и цифровизации бизнес-процессов продаж делает актуальным разработку легких и относительно дешевых программных продуктов.

Статистическое исследование, проведенное компанией Яндекс, показало, что доля активных клиентов торговых сервисов растет с каждым годом. Так, в 2022 году половина пользователей интернет магазинов в возрасте от 16 до 55 лет совершают онлайн покупки чаще одного раза в год. При этом 20% клиентов совершают покупки чаще одного раза в месяц [4]. Данная статистика представлена на графиках на рисунке 1.



Рисунок 1 – Статистика аудитории интернет магазинов на 2021 и 2022 года

Опубликованное компанией CN Analytics ежегодное исследование показывает рост рынка мобильных приложений в 2022 году. Таким образом, в 2022 году показатель потраченных пользователями средств через мобильные приложения вырос на 21% по сравнению с 2021 годом [5]. Данный рост связан

с активным развитием мобильных приложений-сервисов, таких как интернет-магазины, службы доставки еды и сервисы услуг. Кроме того, приложения для совершения интернет-покупок находятся на втором месте в рейтинге популярности среди неигровых приложений, уступая только мобильным банковским и финансовым приложениям.

Исходя из всего вышеперечисленного следует, что тема диссертации «Модель и алгоритм функционирования программных средств систем электронной коммерции» является актуальной и перспективной.

Предметная область электронной коммерции предполагает работу с большим объёмом конфиденциальной информации пользователей, данная особенность подчёркивает актуальность создания улучшенного, более криптостойкого по сравнению с аналогами алгоритма аутентификации пользователей в системе.

Кроме того, целесообразно расширение существующей классификации систем электронной коммерции. Объединение существующих классификаций и добавление собственных признаков позволяет более эффективно проводить обзор аналогов и сравнение их с проектируемой системой электронной коммерции.

Исследования аналогов показало недостаточность использования подходов онтологического и имитационного моделирования в предметной области систем электронной коммерции. Онтологическая модель с расширенным, по сравнению с существующими моделями, набором классов предметной области позволяет повысить эффективность проектирования объекта исследования. Создание дискретно-событийных имитационных моделей позволяет оптимизировать параметры работы системы электронной коммерции с учётом взаимодействия программных средств и человеческого фактора.

Цель магистерской диссертационной работы состоит в анализе и исследовании предметной области, проектировании и реализации системы электронной коммерции на примере системы электронной торговли с

использованием средств онтологического, визуального и математического моделирования. В рамках работы поставлены следующие **задачи**:

- 1) проведение критического анализа предметной области;
- 2) создание онтологической модели предметной области, отличающейся от существующих расширенным набором сущностей и классов предметной области;
- 3) создание дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования реализованной системы электронной торговли;
- 4) разработка сетевого графика проекта по созданию программной системы электронной коммерции для последующей оценки сроков и бюджета данного проекта;
- 5) исследование, проектирование и реализация улучшенного алгоритма аутентификации, позволяющего достигнуть повышенного уровня защищенности данных, а также тестирование и эксперимент по проверке эффективности алгоритма;
- 6) разработка расширенной классификации систем электронной коммерции путем внедрения дополнительных признаков;
- 7) регистрация программы аутентификации для системы электронной коммерции и подготовка разработанной системы электронной коммерции к внедрению.

Объектом исследования является разрабатываемая система электронной торговли.

Предметом исследования являются модели и алгоритмы, применяемые в процессе проектирования и применения систем электронной торговли.

Научная новизна работы заключается в следующем.

1. Предложен модифицированный алгоритм, который отличается от известных алгоритмов наличием пяти ключей, а именно: двух пар открытых и закрытых ключей шифрования и одного публичного ключа аутентификации, а также комбинированным использованием JWT аутентификации и

модифицированного алгоритма асимметричного шифрования RSA, что позволяет повысить надежность процесса аутентификации в системе электронной коммерции.

2. Предложены две дискретно-событийные имитационные модели для оптимизации и определения оптимального режима работы системы электронной торговли. В отличие от известных имитационных моделей, они носят комплексный характер и позволяют учитывать особенность взаимодействия аппаратно-программных средств с пользователями.

3. Предложена расширенная онтологическая модель для предприятия электронной торговли. Данная модель отличается от известных расширенным набором сущностей и классов предметной области. Она используется для эффективного проектирования объекта исследования.

4. Классификация систем электронной коммерции, отличающаяся от существующих расширенным набором 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 метаматематического аппарата сетевого планирования и управления и программного продукта Microsoft Project было проведено структурное и календарное планирование программного проекта разработки системы электронной коммерции, в результате которого были определены срок и бюджет проекта, показавшие его экономическую целесообразность и практическую значимость.

Основные положения, выносимые на защиту.

1. Модифицированный алгоритм, который отличается от известных алгоритмов наличием пяти ключей, а именно: двух пар открытых и закрытых ключей шифрования и одного публичного ключа аутентификации, а также комбинированным использованием JWT аутентификации и модифицированного алгоритма асимметричного шифрования RSA.

2. Две дискретно-событийные имитационные модели для оптимизации и определения оптимального режима работы системы электронной торговли. В отличие от известных имитационных моделей, они носят комплексный характер и позволяют учитывать особенность взаимодействия аппаратно-программных средств с пользователями.

3. Расширенная онтологическая модель для системы электронной торговли. Данная модель отличается от известных расширенным набором сущностей и классов предметной области. Она используется для эффективного проектирования объекта исследования.

4. Предложена классификация систем электронной коммерции, отличающаяся от существующих расширенным набором 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, 2022.

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

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

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

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

6. Научно-технический семинар кафедры «МО и ПЭВМ» 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.

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

1. Тепин, К. С. Разработка мобильного приложения для поддержки бизнес-процессов торгового предприятия / К. С. Тепин, В. Н. Князев // Научные достижения и открытия 2021: сборник статей XX Международного научно-исследовательского конкурса. – Пенза: МЦНС «Наука и Просвещение», 2021. – С. 36 – 43.

2. Тепин, К. С. Исследование предприятия электронной коммерции средствами имитационного моделирования / К. С. Тепин, В. Н. Князев // Молодежный исследовательский потенциал: Сборник статей V Международного научно-исследовательского конкурса. – Петрозаводск: МЦНС «Новая наука», 2022. – С. 90 – 102.

3. Тепин, К. С. Исследование работы службы доставки предприятия электронной коммерции средствами имитационного моделирования / К. С. Тепин, В. Н. Князев // Материалы всероссийской (национальной) научной конференции «Фундаментальные и прикладные исследования. Актуальные проблемы и достижения». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 26 – 28.

4. Тепин, К. С. Имитационное моделирование бизнес-процессов системы электронной коммерции / К. С. Тепин, В. Н. Князев // Материалы международной научной конференции «Высокие технологии и инновации в науке». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 130 – 134.

5. Тепин, К. С. Проектирование программной системы электронной коммерции средствами языка UML / К. С. Тепин, В. Н. Князев // Материалы международной студенческой научной конференции «Поколение будущего». – СПб.: ГНИИ «Нацразвитие», 2022. – С. 85 – 89.

6. Тепин, К. С. Вопросы применения экспертных систем в электронной торговле / К. С. Тепин, А. Г. Михалев, В. Н. Князев // Сборник материалов IX Ежегодной всероссийской межвузовской научно-практической конференции

«Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза: Издательство ПГУ, 2022. – С. 91 – 94.

7. Тепин, К. С. Разработка алгоритма аутентификации в системе электронной коммерции / К. С. Тепин, 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. 398 – 402.

8. Тепин, К. С. Программа аутентификации в системе электронной коммерции // Свидетельство о государственной регистрации программы для ЭВМ. № 2023619790. Заявка № 2023619790, дата поступления 20.03.2023 г., дата государственной регистрации в Реестре программ для ЭВМ 16.05.2023 г. – М., Федеральный институт промышленной собственности, 2023.

9. Тепин, К. С. Разработка модифицированного алгоритма аутентификации для системы электронной коммерции / К. С. Тепин, В. Н. Князев // Международный научно-исследовательский журнал. – 2023. – №7 (перечень ВАК, в печати).

10. Тепин, К. С. Вопросы создания модифицированного алгоритма аутентификации для системы электронной коммерции / К. С. Тепин, В. Н. Князев // Молодой учёный: сборник статей XI Международного научно-исследовательского конкурса. – Пенза: МЦНС «Наука и Просвещение», 2023. – С. 24 – 28.

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.); диплом победителя I степени XI международного научно-исследовательского конкурса «Молодой ученый» (2023 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 и классификация систем электронной коммерции

Электронная коммерция – это собирательный термин, обозначающий совокупность всех видов предпринимательской деятельности с использованием интернет технологий, обеспечивающих максимальную цифровизацию и автоматизацию бизнес-процессов [6]. Одной из самых крупных групп предприятий электронной коммерции являются предприятия электронной торговли.

Система электронной коммерции – программная система, спроектированная для обеспечения автоматизации бизнес процессов предприятия электронной коммерции.

Существует две основные классификации модели организации электронной коммерции: Business-to-Business и Business-to-Client [7, 8]. В первом случае взаимодействие бизнес процессов предприятия происходит в отношении другого предприятия, второй случай является более распространённым – компания электронной коммерции предоставляет свои товары или услуги частным лицам.

Кроме того, на рынке электронной коммерции существуют более узкоспециализированные модели [9].

Business-to-Government модель, охватывающая все онлайн-транзакции между компаниями и органами государственного управления. В процессе электронных экономических взаимоотношений с государством, предприятие электронной коммерции может направить свою деятельность на выполнение заказов от государства.

Business-to-Employee модель позволяет организовать систему предоставления товаров и услуг работникам своей или же другой компании.

Business-to-Manager – это новый режим электронного бизнеса, действующими сторонами которого выступают предприятие – продавец продукции и профессиональный менеджер, оказывающий посреднические

услуги и на этом имеет свою комиссию. К примеру, предприятия продают товары через Интернет, а профессиональный менеджер оказывает услуги по их реализации конечным пользователям.

Consumer-to-Consumer – «Потребитель для потребителя». Эта отрасль электронной коммерции предназначена для заключения сделок между частными лицами по объявлениям, размещенным на сайтах площадок.

В большинстве случаев, предприятия электронной коммерции могут быть классифицированы по роду своей деятельности [10].

Модель интернет-магазина характерна для систем, обеспечивающих автоматизацию продаж широкого перечня товаров на специализированных ресурсах.

Онлайн-сервисы служат для оказания каких-либо услуг, например, приема коммунальных платежей, штрафов и регистрационных взносов в режиме online.

Система электронных торгов типа Business-to-Government позволяет проводить и регистрировать в виде тендеров, конкурсов и котировок. Наиболее наглядный пример этой системы – портал госзакупок;

Так же, система электронной коммерции может представлять из себя торговую площадку, на которой участники могут быть как заказчиками и покупателями, так и поставщиками и продавцами. На таких площадках публикуется и актуализируется информация о товарах, работах и услугах, размещается реклама и предоставляется возможность доставки.

Данные модели требуют различного подхода к маркетингу и организации бизнес процессов. Для рассматриваемого примера системы электронной коммерции интернет магазина продажи одежды характерна модель Business-to-Client.

Кроме того, для предприятия электронной коммерции в сфере продаж одежды характерна классификация товаров на различные категории. Для проектируемой имитационной модели выбрана следующая классификация

товаров: брендовая и люксовая одежда, повседневная одежда, обувь, аксессуары.

Использование классификации необходимо для построения более точной модели, так как одежда из разных категорий имеет разные показатели спроса, что может оказывать влияние на такие параметры модели, как частота появления заказов на определенную группу товаров.

Любая информационная система электронной коммерции это программная система, спроектированная для обеспечения автоматизации бизнес процессов предприятия электронной коммерции. Существующие классификации систем электронной коммерции представлены в таблице 1.

Таблица 1 – Классификация систем электронной коммерции

Название классификационного признака	Виды систем электронной коммерции
По модели организации электронной коммерции.	Business-to-Business, Business-to-Client, Business-to-Government, Business-to-Employee, Business-to-Manager и Consumer-to-Consumer.
По объемам продаж.	Информационная система электронной коммерции может быть направлена на: розничные и оптовые продажи.
По сфере применения.	Спецификация на продаже одежды, еды, автозапчастей, комплектующих ПК или же возможна универсальная система элитной коммерции.
По уровню реализации.	Web-приложения интернет магазина, мобильные приложения, рекламные web-страницы.

Самый часто приводимый в литературе критерий это классификации «по модели организации», то есть Business-to-Business, Business-to-Client и т. д. Данные критерии определяют способ организации и взаимодействия с пользователем системы электронной коммерции, так, например, пользователем может выступать клиент системы электронной коммерции, компания, правительство или же сотрудник определенной компании с определёнными компетенциями и потребностями [11].

Исходя из рассмотренных источников, получена общая классификация систем электронной коммерции. На рисунке 2 представлена данная общая классификация в виде древовидной структуры.

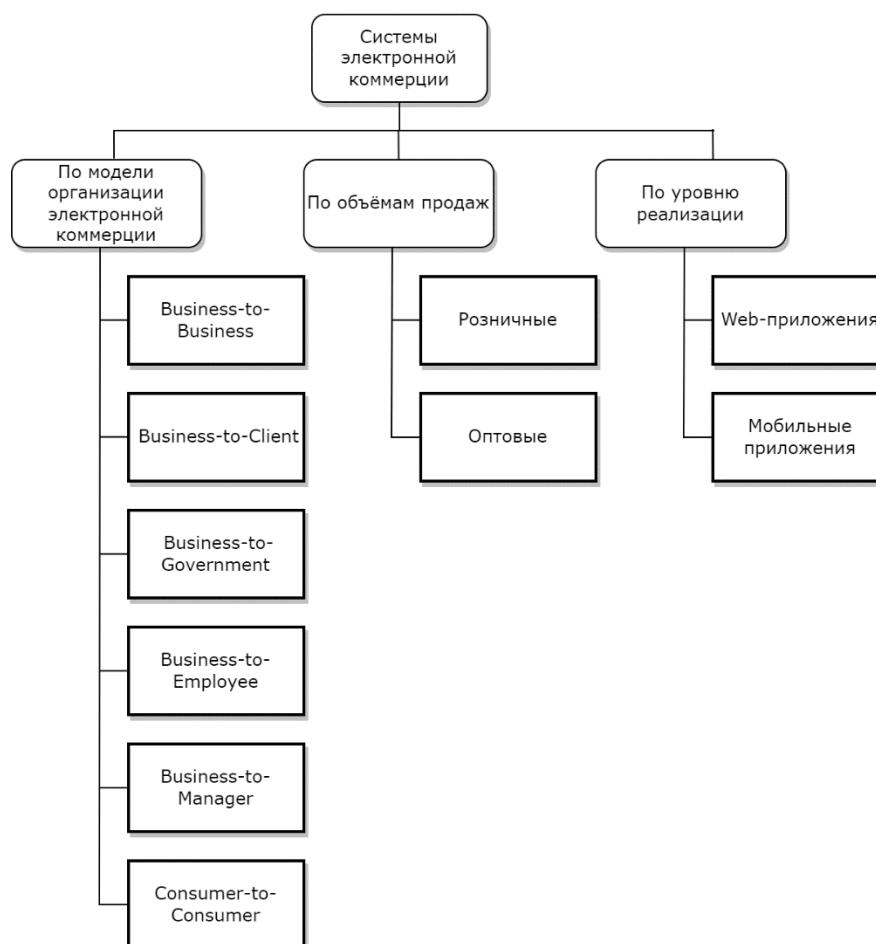


Рисунок 2 – Общая классификация систем электронной коммерции

Исходя из представленных признаков, было принято решение расширить классификация, дополнив её отдельными признаками, выявленными в процессе анализа, проектирования и реализации системы электронной коммерции.

Дополнительные критерии классификации представлены в таблице 2.

Таблица 2 – Дополнительные критерии классификации

Название классификационного признака	Виды систем электронной коммерции
По наличию интерактивной карты магазинов и пунктов выдачи.	Исходя из поставленных функциональных требований, система электронной коммерции может включать или не включать интерактивную карту.
По уровню организации службы товара.	Процесс доставки товара может быть различным: собственная служба доставки, собственные пункты выдачи, партнёрские программы известных служб доставки, отправка посылок почтовым отправлением.
По наличию пользовательского рейтинга.	Системы электронной коммерции могут включать или не включать функционал оценки товаров и качества услуг со стороны пользователя: очки рейтинга, отзывы пользователей.

На рисунке 3 представлена итоговая классификация. Для наглядности новые разработанные признаки, внедрённые в классификацию, выделены цветом.

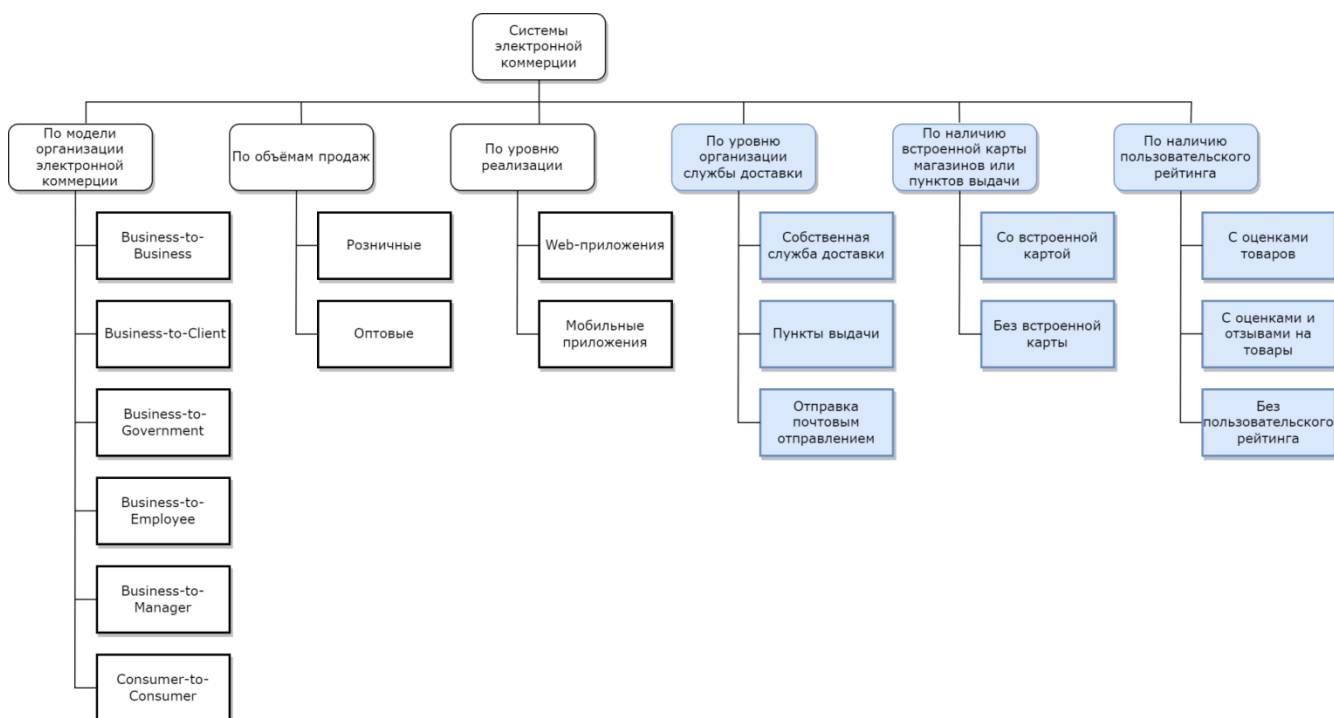


Рисунок 3 – Итоговая расширенная классификация системы электронной коммерции

Таким образом, расширенная система классификации позволяет проводить более подробное сравнение аналогов, выделение недостатков и позволяет устанавливать более качественные функциональные требования на проектирование и разработку систем электронной коммерции.

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

Классифицируем разрабатываемую в рамках данных работы систему электронной коммерции в соответствии с полученной классификацией. Разрабатываемая система электронной коммерции: по модели организации электронной коммерции – Business-to-Client, по объемам продаж – розничная, по уровню реализации – мобильное приложение, по уровню организации службы доставки – отправка почтовым отправлением, по наличию карты – со

встроенной картой, по наличию пользовательского рейтинга – с оценками товаров.

1.2 Обзор аналогов

На рынке систем электронной коммерции в виде мобильных приложений для сферы торговли можно встретить системы крупных торговых сетей, таких как «Aliexpress», «Amazon», «DNS», «Citilink». Однако, данные программные средства работают со сложными информационными системами крупных торговых сетей. Такие системы по объективным причинам не могут быть использованы малыми предприятиями. Проанализируем различные системы электронной коммерции, которые используются малыми предприятиями.

1.2.1 Анализ системы электронной коммерции «Chrono24»

Программные средства «Chrono24» является клиентской программой площадки онлайн-торговли часами «Chrono24» [12]. Существуют версии для операционных систем мобильной платформы Android и iOS. Интерфейс приложения показан на рисунке 4.

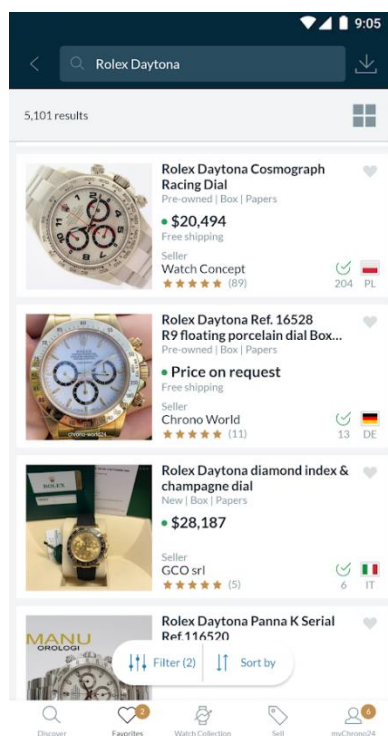


Рисунок 4 – Интерфейс приложения «Chrono24»

Данное приложение имеет дружелюбный пользовательский интерфейс, не перегруженный лишними деталями и сложными элементами. Каталог товаров реализован эргономичными графическими функциональными элементами.

К достоинствам приложения также относятся следующие возможности: возможность переписки с менеджером, наличие оповещений, возможность распознавания модели часов с помощью фото, полученных напрямую с камеры устройства. Главной особенностью приложения является то, что пользователь может выставить свой товар для продажи в приложении.

К недостаткам приложения относится неудобная система группировки выбранных пользователем товаров. Вместо корзины покупок с выбранными пунктами в приложении реализована система закладок, что может вызвать проблемы при попытке заказа сразу нескольких товаров. Кроме того, отсутствует возможность контроля состояния оплаченного заказа на товар.

Также, из-за особенности предметной области в которой создавалось данное приложение, его функционал ориентирован только на виртуальный интернет магазин. В приложении «Chrono24» отсутствует векторная карта или информация о расположении пунктов выдачи или магазинов.

1.2.2 Анализ системы электронной коммерции «Pull&Bear»

В ходе подробного обзора программных средств системы электронной коммерции «Pull&Bear» был сделан вывод, что основными функциями этого программного продукта является функционал совершения покупок, просмотра каталога товаров с возможностью фильтрации и оформлять заявки на товары в магазине одежды и аксессуаров «P&B» [13].

Также данное программное обеспечение обладает функционалом автоматического подбора подходящей одежды исходя из заданных пользователем размеров.

Данное приложение доступно как на Android начиная с 8 версии, так и на iOS начиная с 13 версии. Интерфейс программы представлен на рисунке 5.

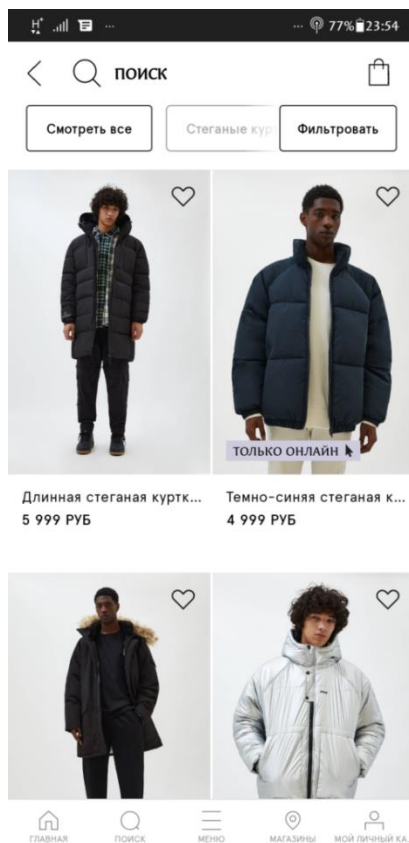


Рисунок 5 – Интерфейс приложения «Pull&Bear»

Из достоинств программной части системы можно отметить наличие карты с расположением магазинов, реализацию чата с продавцом магазина, наличие удобной корзины покупок. Также присутствует удобная работа с изображениями, возможность зума и прокрутки.

Основной недостаток приложения – крайне неудобный интерфейс. Мелкие шрифты в сочетании с большими контейнерами не способствуют легкому восприятию контента. В макете интерфейса использованы только линейные вертикальные контейнеры, что вызывает трудность при прокрутке большого количества объектов в контейнере. В пользовательских отзывах о приложении также преобладают жалобы на низкую эргономичность и общую запутанность интерфейса.

Кроме всего перечисленного, негативное впечатление от использования приложения «Pull&Bear» возникает из-за регулярных всплывающих оповещений об ошибках соединения с серверной частью приложения.

Также существенным недостатком является отсутствие функциональной возможности оставить отзыв на конкретный товар или поставить оценку.

1.2.3 Анализ системы электронной коммерции «Shopshap»

Программные средства системы электронной коммерции «Shopshap» позволяют совершать покупки в интернет магазине головных уборов [14]. Приложение доступно только для мобильной платформы под управлением Android. Интерфейс данного приложения показан на рисунке 6.

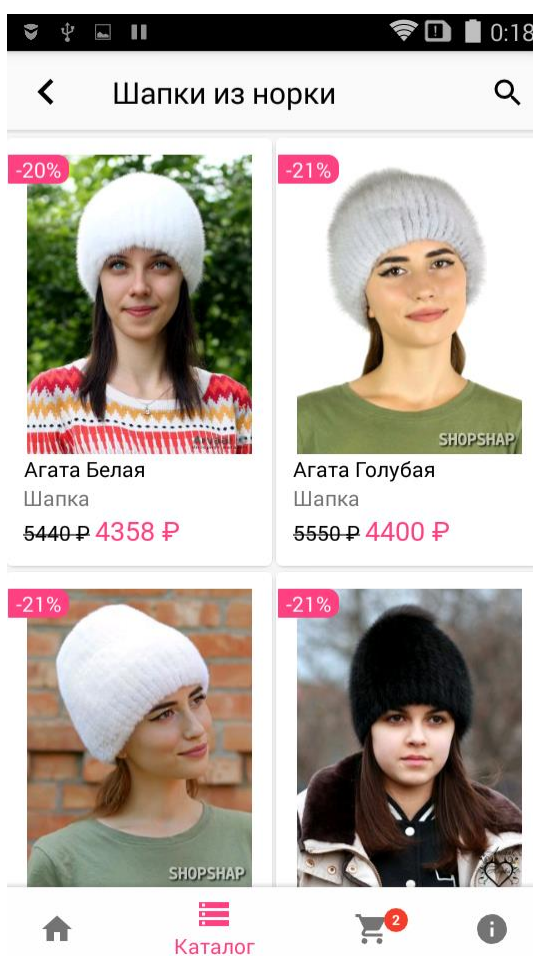


Рисунок 6 – Интерфейс приложения «Shopshap»

Приложение «Shopshap» обладает эргономичным и функционально достаточным интерфейсом. Например, категории товаров вынесены в боковое меню, что позволяет быстро перемещаться сразу в желаемый раздел.

Также, к достоинствам данного приложения можно отнести наличие удобной корзины выбранных товаров и меню оплаты банковской картой.

К недостаткам можно отнести отсутствие интерактивной векторной карты с отметками расположения магазина, прямой связи с продавцами или менеджерами магазина. Как и в предыдущих приложениях, серьезным недостатком является отсутствие возможности оставлять отзывы или оценки на товары.

1.2.4 Подведение итогов сравнения аналогичных решений

После рассмотрения данных программных средств систем электронной коммерции, можно сделать вывод об их функциональной недостаточности для использования в качестве универсального решения для предприятия торговли. Сравнение данных аналогов с разрабатываемой программой представлено в таблице 3 по каждому из критериев.

Таблица 3 – Сопоставление аналогов

Система электронной коммерции	Удобство и качество интерфейса (по оценкам пользователей)	Наличие карты расположения магазинов	Наличие корзины покупок	Связь с продавцом или менеджером в приложении	Возможность оставлять отзывы и оценки
Разрабатываемая	Удобный	Да	Да	Да	Да
«Chrono24»	Удобный	Нет	Нет	Да	Нет
«Pull&Bear»	Запутанный, не эргономичный	Да	Да	Нет	Нет
«Shapshop»	Удобный	Нет	Да	Нет	Нет

Во всех представленных аналогах наблюдается отсутствие такого важного функционального элемента, как возможность пользователю оставлять отзывы и оценки на приобретенные им товары. Наличие данной функции в разрабатываемом приложении может стать значительным преимуществом, так как она позволяет пользователям не только ставить оценки и писать отзывы на товары, но и выбирать товары исходя из рейтинга, выставленного самими пользователями.

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

На основе данного сравнения аналогов принято решение создать программный продукт системы электронной коммерции, лишенный выявленных недостатков и предоставляющий необходимый функционал для поддержки всех выделенных информационных задач в рамках данной предметной области.

1.3 Анализ возможностей повышения эффективности разработки систем электронной коммерции

В ходе проектирования и изучения организации предприятия электронной коммерции важным этапом является моделирование. Сбор данных и моделирование позволяют оценить большую часть рисков и определить оптимальный режим работы предприятия [15].

Один из самых эффективных методов моделирования является имитационное моделирование – метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности [16, 17]. Такую модель можно выполнить во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

В ходе изучения литературы и анализа аналогов был сделан вывод о том, что имитационное моделирование для предприятий электронной торговли применяется крайне редко.

Перед созданием имитационной модели обязательным этапом является формализация структуры предметной области. Для этого может быть эффективно использован инструментарий онтологического моделирования. Онтологическое моделирование позволяет создавать иерархию классов и

объектов предметной области. Между ними устанавливаются связи и создается онтологический граф.

Для системы электронной коммерции выдвигаются требования повышенной надежности и безопасности данных [18, 19, 20]. Это обусловлено тем, что в большинстве случаев система электронной коммерции содержит функционал хранения и обработки персональных данных пользователей, данных средств оплаты и информацию об электронных платежах.

Утечка таких данных может повлечь за собой крупные финансовые и репутационные потери [21], следовательно, актуальной является перспектива создания улучшенных алгоритмов безопасности и криптографических инструментов.

Для программных средств электронной коммерции в виде трехзвенной архитектуры с клиентом, сервером и базой данных возможны следующие пути повышения надежности [22, 23]:

- улучшение алгоритма аутентификации;
- применение улучшенного алгоритма шифрования при передаче данных между различными компонентами архитектуры программной части системы;
- повышение надежности хранения данных в базе данных посредством добавления улучшенного криптографического алгоритма хеширования.

Самым важным из перечисленных улучшений является повышение надежности алгоритма аутентификации. Так как уязвимость данного фрагмента может поставить всю систему под опасность неправомерного доступа. Роль аутентификации заключается в том, что актер для доступа к системе должен предъявить нечто, называемое фактором аутентификации. Строго говоря, фактор аутентификации – это определенный вид уникальной информации, предоставляемый субъектом системе при его аутентификации.

Так как программные средства системы электронной коммерции созданы с использованием технологий Java, самым оптимальным способом создания

алгоритма повышения надежности является построение аутентификации по стандарту JWT.

Данные пользователя, пересылаемые с использованием JWT-структуры подписываются. Таким образом, получатель может убедиться в подлинности сообщения. Однако, существенным недостатком JWT является то, что данные не зашифрованы и могут быть перехвачены и использованы злоумышленниками.

Перспективой улучшения надежности данного алгоритма является внедрение асинхронного шифрования данных в составе JWT [24]. При этом шифрованию подлежит только полезная нагрузка JWT-сообщения. Для этого может потребоваться создание программной надстройки над логикой работы с JWT-сообщениями. Данная программная надстройка должна будет выполнять функции шифрования и дешифрования.

Таким образом, пересылаемые данные смогут быть проверены на подлинность и одновременно быть защищены от неправомерного доступа. Это позволит достичь высокого уровня надежности при передаче данных между программными компонентами системы электронной коммерции.

1.4 Сетевое планирование процесса разработки

1.4.1 Структурное планирование

Структурное планирование заключается в выделении отдельных работ и выявлении временных связей между ними. Основным инструментом структурного планирования является сетевое планирование [25].

Сетевое планирование использует теорию графов для отображения всей структуры взаимосвязанных работ проекта. Процесс сетевого планирования позволяет построить сетевой график, представляющий собой ориентированный граф. Каждая вершина данного графа соответствует отдельному рабочему процессу в рамках проекта и хранит временное значение длительности данного процесса. Связи между работами носят строго последовательный характер. Так,

ни какая работа не может начаться пока не закончатся все работы, предшествующие ей. Пример сетевого графика представлен на рисунке 7.



Рисунок 7 – Пример сетевого графика

Кроме того, начало и конец проекта обозначаются специальными фиктивными вершинами, имеющими нулевую длительность. Таки вершины называются вехами [26].

Важнейшей характеристикой любого сетевого графика является критический путь. Критический путь – это такая последовательность вершин от начальной до конечной вехи графа, что задержка выполнения работы в любой вершине приведет к задержке сроков реализации проекта. При этом каждая работа, соответствующая вершина которой находится в критическом пути, называется критической работой.

Выявить критические работы можно по средствам использования метрик раннего времени начала и позднего время начала для каждой работы.

Данные метрики построены с помощью следующих формул [27].

Пусть t_i – время выполнения работы i , $T_p(i)$ – раннее время начала работы i , $T_n(i)$ – позднее время начала работы i . Тогда

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

$$T_n(i) = \min(T_n(j) - t_j) \quad j \in H,$$

где G – множество работ, предшествующих работе i ;

H – множество работ, следующих за работой i .

Используем метод сетевого планирования для сетевого планирования программного планирования разработки мобильного приложения. Первым шагом является выделение ресурсов и задач в рамках проекта.

В таблице 4 представлены выделенные задачи проекта.

Таблица 4 – Перечень задач проекта

№	Название задачи	Тип	Предшественники	Длительность (дней)
1	Начало проекта	Веха	-	0
2	Анализ предметной области и требований	Фаза	-	-
3	Изучение предметной области и постановка задачи	Задача	1	1
4	Обзор аналогичных решений на рынке мобильных приложений	Задача	3	2
5	Подведение итогов обзора аналогичных решений	Задача	4	1
6	Анализ требований	Задача	3	2
7	Построение концептуальной модели предметной области	Задача	5;6	1
8	Проектирование приложения	Фаза	-	-
9	Выбор технологий и средств разработки	Задача	6	3
10	Проектирование пользовательского интерфейса мобильного приложения	Задача	9	5
11	Проектирование архитектуры мобильного приложения	Задача	9	5
12	Проектирование архитектуры серверной части приложения	Задача	9	5
13	Логическое проектирование базы данных	Задача	7	3
14	Реализация	Фаза	-	-
15	Физическое моделирование и создание скриптов генерации базы данных	Задача	9,13	4
16	Реализация пользовательского интерфейса мобильного приложения	Задача	10	5
17	Реализация ПО мобильного приложения	Задача	11,16	5
18	Реализация ПО серверной части приложения	Задача	12;15	4
19	Тестирование	Фаза	-	-
20	Тестирование серверной части приложения	Задача	18	3
21	Тестирование мобильного приложения	Задача	17,18	4
22	Конец проекта	Веха	20,21	0

Таблица 1 показывает, что процесс разработки был распланирован на 22 отдельные задачи. Следующим шагом определим используемые ресурсы. В таблице 5 представлены ресурсы проекта.

В таблице 6 представлено назначение ресурсов на каждую задачу

Таблица 5 – Список ресурсов проекта

Ресурс	Тип	Ставка		Затраты
		Станд. Ставка	Ставка сверхур.	
Программист 1	Трудовой	20000 р./мес	400 р./ч	-
Программист 2	Трудовой	20000 р./мес	400 р./ч	-
Менеджер проекта	Трудовой	21000 р./мес	410 р./ч	-
Тестировщик	Трудовой	15000 р./мес	300 р./ч	-
Компьютер	Материальный	25000 р.	-	-
Роутер	Материальный	4000 р.	-	-
Интернет услуги	Затраты	-	-	1200 р.

Таблица 6 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	-	-
2	Анализ предметной области и требований	-	-
3	Изучение предметной области и постановка задачи	Менеджер проекта Компьютер Роутер	100% 3 шт 1 шт
4	Обзор аналогичных решений на рынке мобильных приложений	Менеджер проекта Программист1 Интернет услуги	50% 50% 1200 руб
5	Подведение итогов обзора аналогичных решений	Менеджер проекта	100%
6	Анализ требований	Программист2 Менеджер проекта	50% 50%
7	Построение концептуальной модели предметной области	Программист2	100%
8	Проектирование приложения	-	-
9	Выбор технологий и средств разработки	Программист2 Менеджер проекта	60% 40%
10	Проектирование пользовательского интерфейса мобильного приложения	Программист1	100%
11	Проектирование архитектуры мобильного приложения	Программист1	100%
12	Проектирование архитектуры серверной части приложения	Программист2	100%
13	Логическое проектирование базы данных	Программист1 Программист2	30% 70%
14	Реализация	-	-
15	Физическое моделирование и создание скриптов генерации базы данных	Программист1 Программист2	50% 50%
16	Реализация пользовательского интерфейса мобильного приложения	Программист1	100%
17	Реализация ПО мобильного приложения	Программист1 Программист2	75% 25%

Продолжение таблицы 6

№	Название задачи	Ресурс	Единицы
18	Реализация ПО серверной части приложения	Программист2	100%
19	Тестирование	-	-
20	Тестирование серверной части приложения	Тестировщик Программист2	85% 15%
21	Тестирование мобильного приложения	Тестировщик	100%
22	Конец проекта	-	-

На рисунке 8 показан полученный сетевой график, описывающий структуру процесса реализации анализируемого проекта.

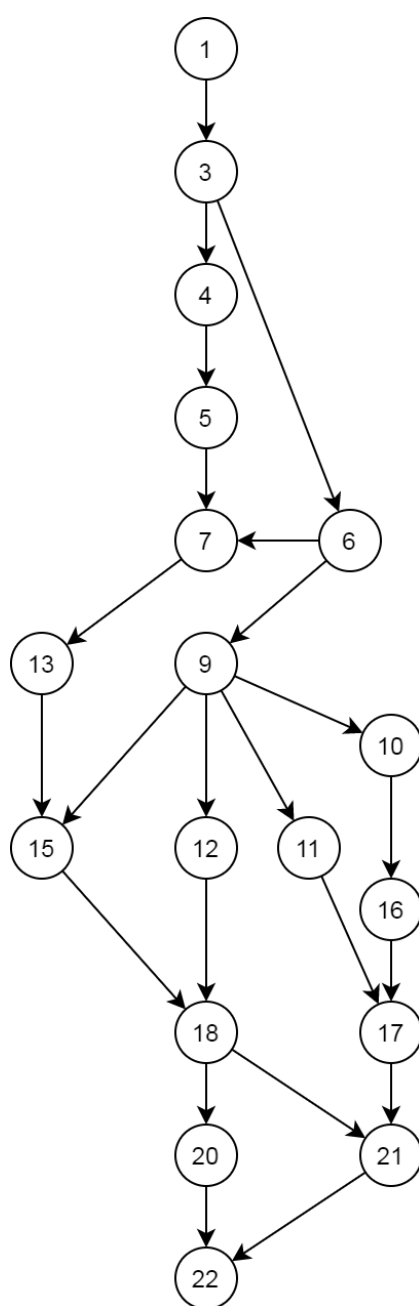


Рисунок 8 – Сетевой график проекта создания системы

Используя полученный сетевой график, в соответствии с формулами, рассчитаем раннее время начала и позднее время каждой работы проекта. Раннее время начала представлено в таблице 7. Позднее время начала представлено в таблице 8.

Таблица 7 – Расчет раннего времени начала каждой работы

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	3	0	1	0+1	1
5	4	1	2	1+2	3
6	3	0	1	0+1	1
7	5	3	1	Max(3+1;1+2)	4
	6	1	2		
9	6	1	2	1+2	3
10	9	3	3	3+3	6
11	9	3	3	3+3	6
12	9	3	3	3+3	6
13	7	4	1	4+1	5
15	9	3	3	Max(3+3;5+3)	8
	13	5	3		
16	10	6	5	6+5	11
17	11	6	5	Max(6+5;11+5)	16
	16	11	5		
18	12	6	5	Max(6+5;8+4)	12
	15	8	4		
20	18	12	4	12+4	16
21	17	16	5	Max(16+5;12+4)	21
	18	12	4		
22	20	16	3	Max(16+3;21+4)	25
	21	21	4		

Таблица 8 – Расчет позднего времени начала каждой работы

Работа	Следующая работы			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность предшеств.		
22	-	-	-	-	25
21	22	25	4	25-4	21
20	22	25	3	25-3	22
18	20 21	22 21	3	Min(22-3;21-3)	18
17	21	21	5	21-5	16
16	17	16	5	16-5	11
15	18	18	4	18-4	14
13	15	14	3	14-3	11
12	18	18	5	18-5	13
11	17	16	5	16-5	11
10	16	11	5	11-5	6
9	10 11 12 15	6 11 13 14	3	Min(6-3;11-2;13-2;14-2);	3
7	13	11	1	11-1	10
6	7 9	10 3	2	Min(10-2;3-2)	1
5	7	10	1	10-1	9
4	5	9	2	9-2	7
3	4 6	7 1	1	Min(7-1;1-1)	0
1	3	0	0	0-0	0

Используя значения раннего и позднего начала работы можно вычислить, является ли данная работа критической. Нахождение критических работ представлено в таблице 9.

Таблица 9 – Определение критических работ

№ работы	Раннее время начала работы	Позднее время начала работы	Запас времени для работы
1	0	0	0
3	0	0	0
4	1	7	6
5	3	9	6
6	1	1	0
7	4	10	6
9	3	3	0
10	6	6	0
11	6	11	5
12	6	13	7
13	5	11	6
15	8	14	6
16	11	11	0
17	16	16	0
18	12	18	6
20	16	22	6
21	21	21	0
22	25	25	0

Значение нуля в столбце запаса времени говорит нам, что данная вершина является критической. Соединив все критические вершины можно получить критический путь.

Сетевой график с критическим путем показан на рисунке 9.

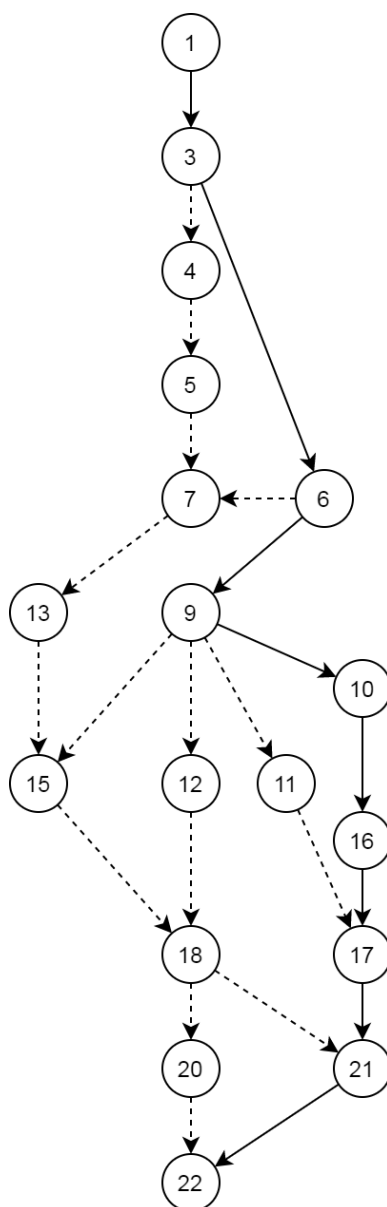


Рисунок 9 – Критический путь проекта

На данном рисунке критический путь обозначен сплошными дугами, а пунктирными линиями обозначены связи между работами, не входящими в критический путь. Длина критического пути проекта составляет 25 дней.

1.4.2 Календарное планирование и оценка бюджета проекта

Проведем планирование проекта с использованием выбранного средства управления проектами Microsoft Project 2007 [28 – 30]. Началом проекта была принята дата 01.11.2022.

На рисунке 10 представлен полученный список задач проекта с назначенными ресурсами.

№	Название задачи	Длительность	Начало	Окончание	Предшст	Названия ресурсов
1	Начало проекта	0 дней	Вт 01.11.22	Вт 01.11.22		
2	Анализ предметной области и требований	5 дней	Вт 01.11.22	Пн 07.11.22	1	
3	Анализ предметной области и постановка задачи	1 день	Вт 01.11.22	Вт 01.11.22	1	Компьютер[3];Роутер[1];Менеджер проекта
4	Обзор аналогичных решений на рынке мобильных прило	2 дней	Ср 02.11.22	Чт 03.11.22	3	Менеджер проекта[50%];Программист1[50%]
5	Подведение итогов анализа аналогичных решений	1 день	Пт 04.11.22	Пт 04.11.22	4	Менеджер проекта
6	Анализ требований	2 дней	Ср 02.11.22	Чт 03.11.22	3	Программист2[50%];Менеджер проекта[50%]
7	Построение концептуальной модели предметной област	1 день	Пн 07.11.22	Пн 07.11.22	5,6	Программист2[50%]
8	Проектирование приложения	8 дней	Вт 08.11.22	Чт 17.11.22		
9	Выбор технологий и средств разработки	3 дней	Вт 08.11.22	Чт 10.11.22	7	Менеджер проекта[60%];Программист2[40%]
10	Проектирование пользовательского интерфейса	5 дней	Вт 08.11.22	Пн 14.11.22	7	Программист1[50%]
11	Проектирование архитектуры приложения	5 дней	Пт 11.11.22	Чт 17.11.22	9	Программист2
12	Логическое проектирование базы данных	3 дней	Вт 08.11.22	Чт 10.11.22	7	Программист1[30%];Программист2[70%]
13	Реализация	24 дней	Пт 11.11.22	Ср 14.12.22		
14	Физическое моделирование базы данных	2 дней	Пт 11.11.22	Ср 16.11.22	9,12	Программист1[50%];Программист2
15	Разработка веб сервисов	2 дней	Пт 11.11.22	Ср 23.11.22	9	Программист2
16	Разработка дизайна пользовательского интерфейса	3 дней	Вт 15.11.22	Пн 21.11.22	10	Программист1[38%]
17	Разработка пользовательского интерфейса	3 дней	Вт 22.11.22	Чт 24.11.22	11,16	Программист1[60%];Программист2[40%]
18	Разработка ПО мобильного приложения	14 дней	Пт 25.11.22	Ср 14.12.22	15,17	Программист1[50%];Программист2[50%]
19	Тестирование	18 дней	Чт 24.11.22	Пн 19.12.22		
20	Тестирование серверной части приложения	3 дней	Чт 24.11.22	Пн 28.11.22	15	Тестирующий[85%];Программист1[15%]
21	Тестирование клиентской части приложения	3 дней	Чт 15.12.22	Пн 19.12.22	18	Тестирующий
22	Конец проекта	0 дней	Пн 19.12.22	Пн 19.12.22	19	

Рисунок 10 – Задачи проекта

На рисунке 11 представлена диаграмма Ганта, автоматически сгенерированная в среде Microsoft Project на основе построенного списка задач.



Рисунок 11 – Диаграмма Ганта проекта

Рассмотрим загрузку ресурсов «программист1» и «программист2». Полученные графики загрузки представлены на рисунке 12 и 13.

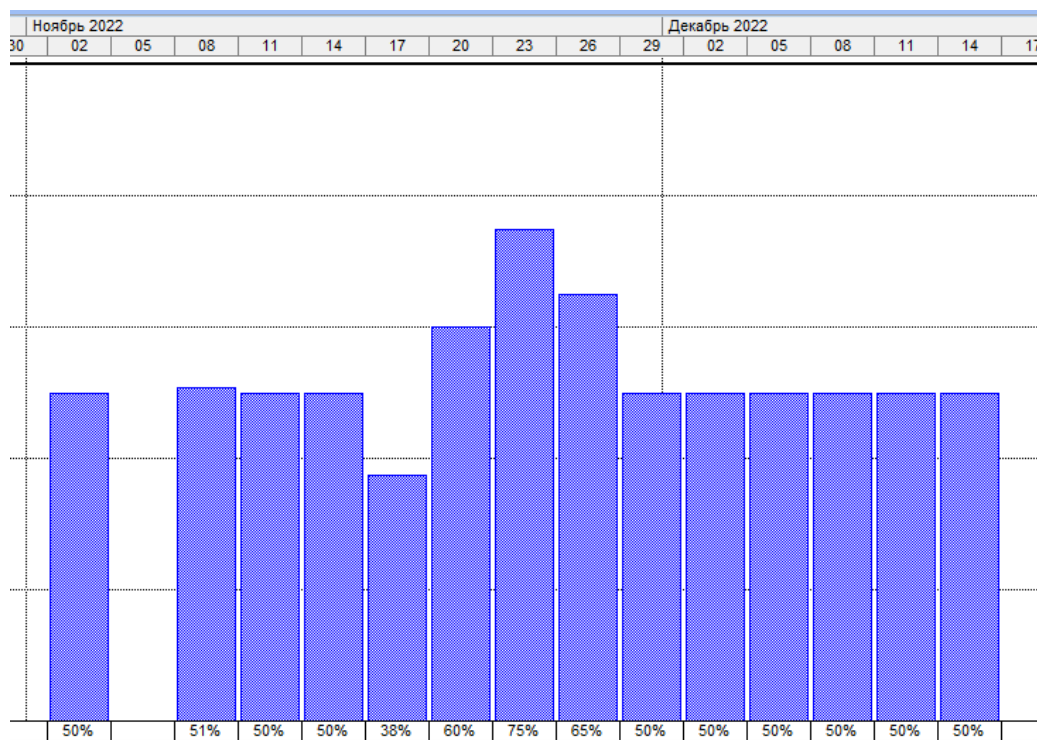


Рисунок 12 – Загрузка ресурса «Программист1»

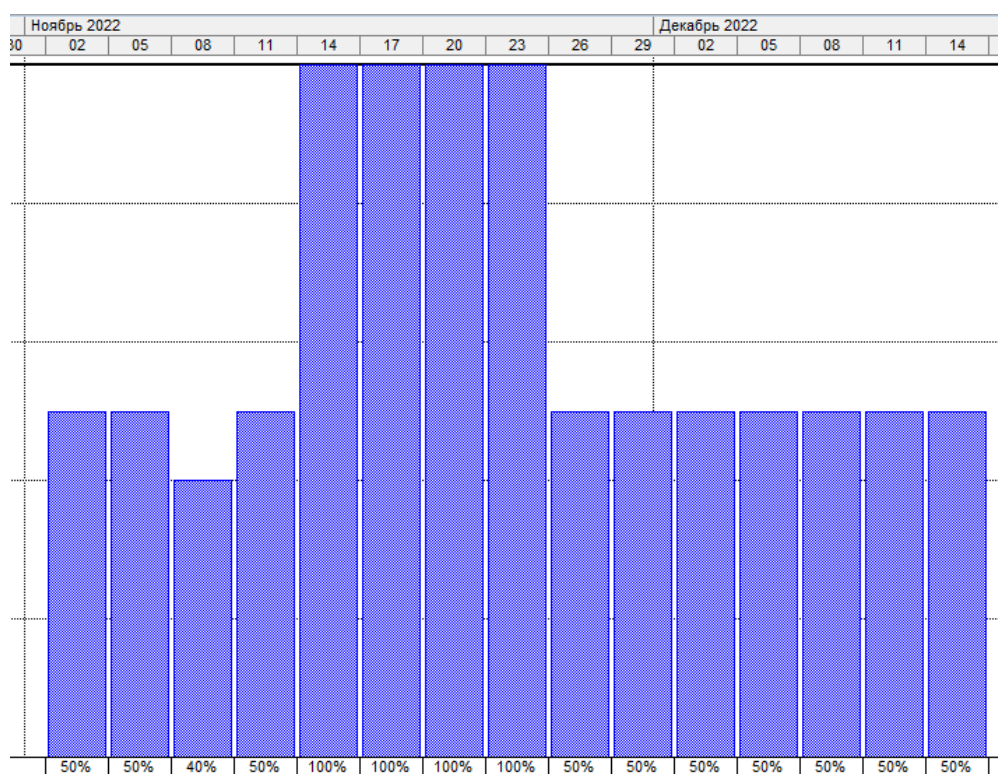


Рисунок 13 – Загрузка ресурса «Программист2»

Планирование в программе Microsoft Project показало соответствие в структуре плана и в сроках проекта с планированием, рассчитанным вручную.

1.4.3 Подведение итогов сетевого планирования

Для данного проекта было проведено структурное и календарное планирование, в том числе, с использованием программы Microsoft Project. Отчет, составленный в программе Microsoft Project, представлен на рисунке 14.

на дату Ср 22.06.22

Даты			
Начало:	Вт 01.11.22	Окончание:	Пн 19.12.22
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Откло н. начала:	0 дней	Откло н. окончания:	0 дней
Длительность			
Текущий план:	25 дней	Оставшаяся:	25 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	25 дней	Процент завершени:	0%
Трудозатраты			
Текущий план:	339,6 ч	Оставшиеся:	339,6 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	339,6 ч	Процент завершени:	0%
Затраты			
Текущий план:	120 552,50 р	Оставшиеся:	120 552,50 р
Базовый план:	0,00 р	Фактические:	0,00 р
Отклонение:	120 552,50 р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	22	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	0
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	22	Итого ресурсов:	7

Рисунок 14 – Отчет по проекту

Из отчета можно сделать вывод, что итоговая стоимость проекта составляет 120 550 рублей. Длительность проекта составляет 25 дней. Полученные данные свидетельствуют о том, что с экономической точки зрения проект целесообразен.

1.5 Выбор технологий и инструментальных средств разработки

В качестве методологии разработки выбрана модель Rational Unified Process (RUP). Данный подтип спиральной модели разработки предполагает последовательное выполнение итераций разработки. При этом на каждой

итерации проект разделяется на несколько фронтов работ, например, позволяет проводить независимо работы по разработке клиентского приложения и серверного приложения. Данная модель подходит для разработки программного продукта ULDBS и позволяет рассматривать разработку клиентского мобильного приложения как отдельный проект, процесс разработки которого встраивается в процесс разработки информационной системе согласно методологии RUP [31].

В качестве базы данных информационной системы была выбрана PostgreSQL. На сегодняшний день это одна из самых популярных и востребованных баз данных и СУБД на рынке. К ее преимуществам можно отнести надежные механизмы транзакций, механизм репликации и легкую масштабируемость [32].

Приложение связывается с базой данных через серверную часть, состоящую из RESTful сервисов, созданных с помощью технологий JavaEE на языке Java. Приложение серверной части развертывается на сервере Apache Tomee Plume 8.0.4. Данная архитектура отличается легкостью развертывания, администрирования, поддержки и расширяемости.

Клиентское мобильное приложение создано для платформы Android с использованием Android SDK. Android сейчас является крайне популярной мобильной платформой, так как позволяет разработчикам создавать удобные и гибкие решения. В качестве языка реализации выбран Kotlin – язык, транслируемый для Java Virtual Machine. Kotlin обладает рядом преимуществ перед классическим языком Java: более удобные синтаксические конструкции, отсутствие тяжелых встроенных модулей и оптимизация для работы в среде Android [33].

Для логического проектирования базы данных использовалась программа ERwin Data Modeler [34].

Для формализации проектирования программного обеспечения системы электронной коммерции был использован язык моделирования UML [35]. Унифицированный язык моделирования UML был использован для создания

диаграмм деятельности, последовательности, компонентов и классов. Так же UML был использован в процессе разработки модифицированного алгоритма аутентификации.

Среда управления проектами Microsoft Project была использована для планирования проекта разработки системы электронной коммерции. Было проведено сетевое планирование. Созданы задачи и ресурсы проекта. Результатом планирования является итоговая смета проекта, содержащая сроки и бюджет.

На этапе онтологического моделирования была использована среда Protégé. Для имитационного моделирования была использована среда GPSS Studio. В GPSS Studio были созданы две дискретно-событийные модели, позволяющие проводить оптимизацию режима функционирования системы электронной коммерции.

Выводы

Анализ предметной области и обзор аналогов показал ряд недостатков рассмотренных систем электронной коммерции. На основании этого было решено разработать собственную электронную систему электронной коммерции с модифицированным алгоритмом аутентификации, а также с применением технологий онтологического и имитационного моделирования.

Кроме того, на основании данных из нескольких источников была сформирована общая классификация систем электронной коммерции, которая была дополнена новыми признаками.

Были выбраны технологии и инструментальные средства реализации.

Было выполнено сетевое планирование проекта аналитическим путем и с помощью среды Microsoft Project. Результатом планирования получена смета проекта со сроком выполнения в 25 дней и бюджетом в 120 550 рублей. Данные результаты позволяют сделать вывод об экономической целесообразности проекта.

2 Разработка системы электронной коммерции и модифицированного алгоритма аутентификации

2.1 Анализ требований

2.1.1 Анализ функциональных требований

После проведения сравнительного анализа аналогов, с учётом выявленных недостатков и преимуществ, были сформулированы следующие пункционные требования для разрабатываемой системы электронной коммерции.

Разрабатываемая система электронной коммерции должна поддерживать функционал для пользователей в виде покупателя, менеджера, администратора.

Функции системы для каждой категории пользователей показаны на диаграмме вариантов использования на рисунке 15.



Рисунок 15 – Диаграмма вариантов использования системы электронной коммерции

Составим сценарий выполнения варианта использования «Оставлять отзыв на товар». Сценарий представлен на рисунке 16.

Наименование: Оставлять отзыв на товар
ID: 1
Краткое описание: На экране подробной информации о конкретном товаре в разделе отзывов о товаре пользователь имеет возможность оставить свой отзыв и оценку товару.
Действующие лица: Пользователь
Предусловие: Загружена страница с товаром из каталога, пользователь авторизован в приложении.
Основной поток: <ol style="list-style-type: none"> 1. Пользователь открывает раздел отзывов. 2. Пользователь выбирает оценку товара по пятибалльной шкале. 3. Пользователь вводит текстовый отзыв. 4. Пользователь нажимает кнопку «опубликовать». 5. ЕСЛИ в приложении есть информация о том, что пользователь покупал товар <ol style="list-style-type: none"> 5.1. Отправка данных пользовательского отзыва на сервер 5.2. Обновление страницы 6. ИНАЧЕ <ol style="list-style-type: none"> 6.1. Приложение выводит сообщение о том, что пользователь не может оставлять отзыв на товар, который он не покупал 6.2. Приложение предлагает пользователю купить данный товар

Рисунок 16 – Сценарий прецедента «Оставлять отзыв на товар»

Для варианта использования «Войти в систему» была создана диаграмма деятельности. Она продемонстрирована на рисунке 17.

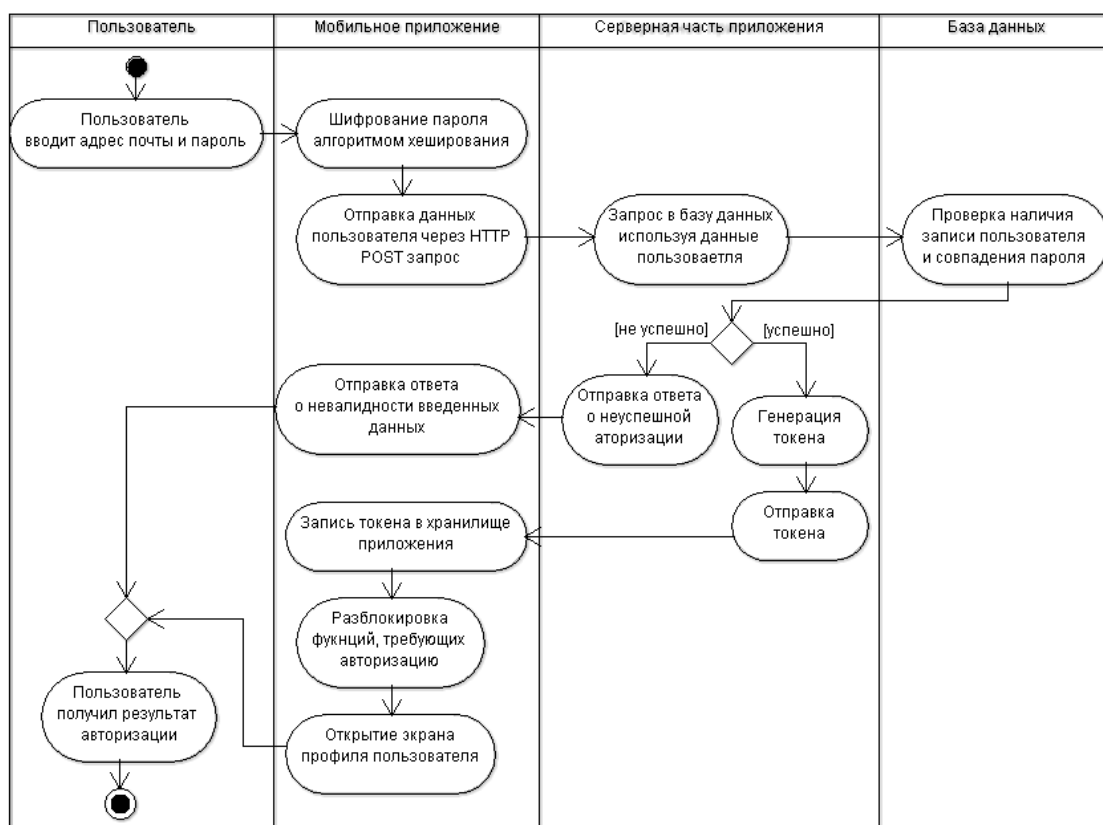


Рисунок 17 – Диаграмма деятельности прецедента «Войти в систему»

На рисунке 18 представлена диаграмма последовательности [13] варианта использования «Просматривать статистику».

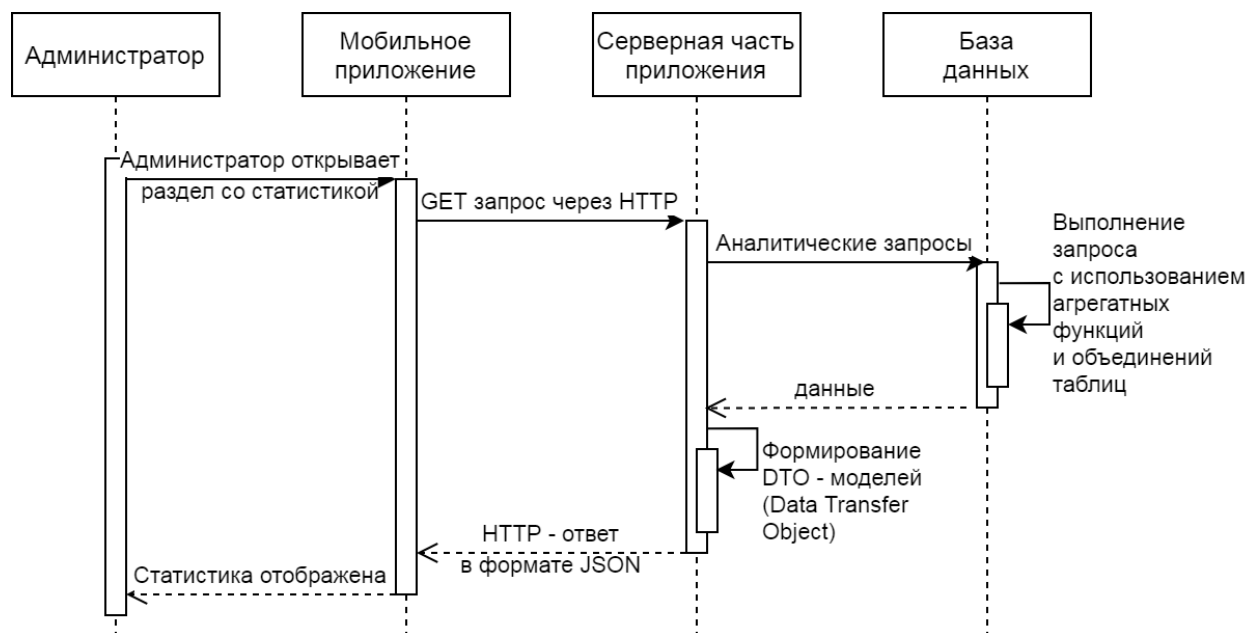


Рисунок 18 – Диаграмма последовательности прецедента «Просматривать статистику»

На рисунке 19 представлена диаграмма последовательности варианта использования «Отмечать выполнение заявок» [36].

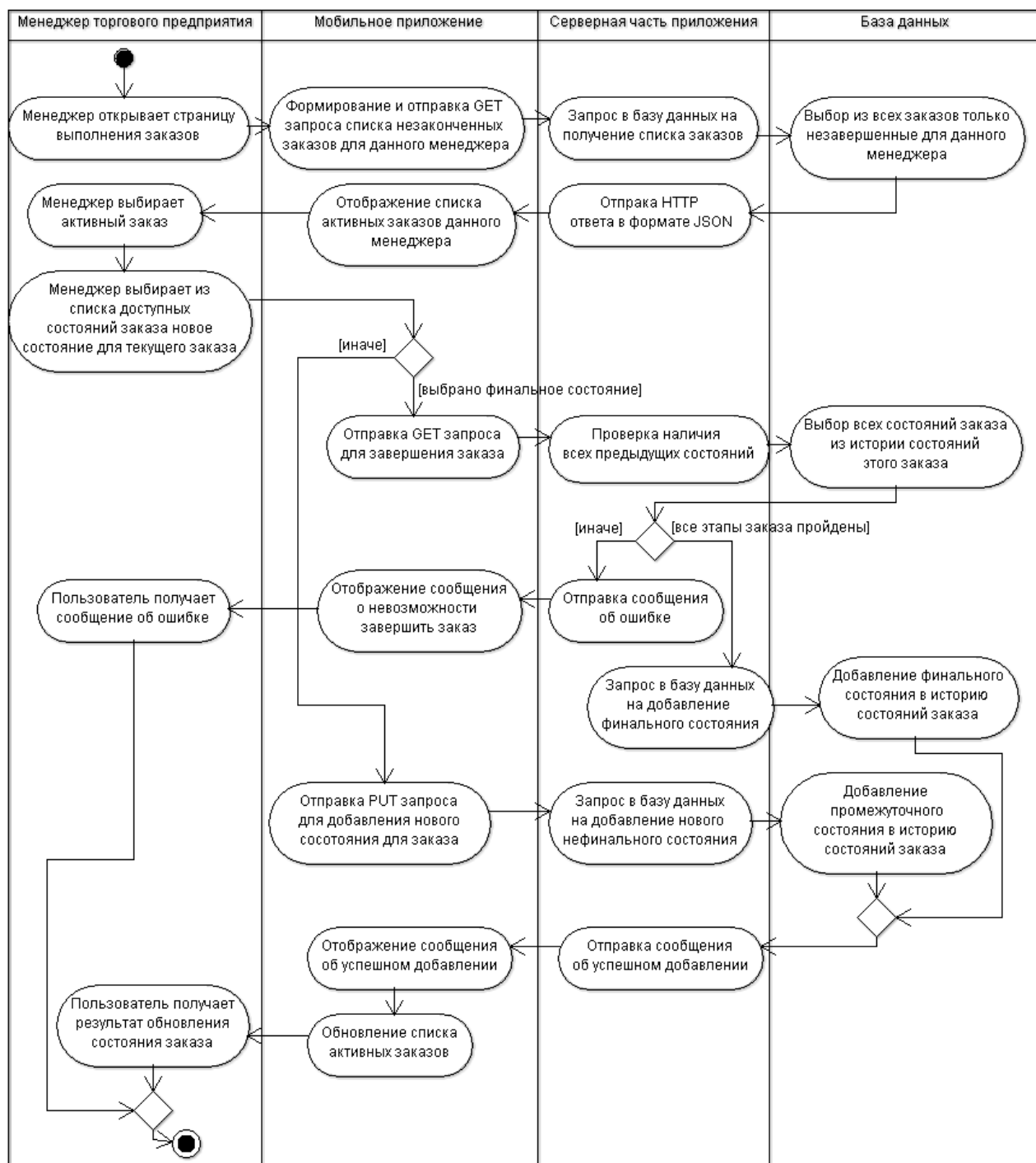


Рисунок 19 – Диаграмма деятельности прецедента «Отмечать выполнение заявок»

Полученные диаграммы деятельности и последовательности позволяют достаточно подробно описать взаимодействие между всеми слоями архитектуры приложения с учетом направления потоков данных.

2.1.2 Анализ нефункциональных требований

Исходя из поставленной задачи, был сформулирован ряд нефункциональных требований к разрабатываемой системе электронной коммерции.

Клиентская часть системы электронной коммерции представляет собой мобильное приложение, которое должно показывать стабильную работу на любых допустимых наборах выходных данных и при любых действиях пользователя в рамках пользовательского интерфейса.

Для разрабатываемого мобильного приложения должен быть разработан эргономичный пользовательский интерфейс, отвечающим ряд у требований [37]:

- простая структура и иерархия экранов;
- использование стандартных для Android приложений элементов навигации;
- локальное сохранение данных пользователя;
- узнаваемый дизайн и цветовая гамма;
- высокая скорость отклика при переключении экранов и использовании функциональных элементов.

Кроме того, приложение должно быть совместимо с операционными системами Android версии 8.0 и выше.

Серверная часть приложения должна быть легко развёртываемой и конфигурируемой. Данное требование является решающим на этапе выбора технологий реализации серверного приложения и базы данных.

Серверная часть приложения и база данных должны быть совместимы с операционными системами семейства Linux и его серверными дистрибутивами.

Кроме того, для базы данных должны быть включены функции распределенного хранения данных, индексации и резервного копирования.

2.2 Онтологическое моделирование предметной области

Проектирование системы электронной коммерции целесообразно проводить на основе предварительного анализа, структуризации и систематизации информации предметной области. Разработка онтологической модели предметной области начинается с онтологического анализа [38].

В среде Protege был разработан соответствующий задаче набор классов, свойств этих классов, а также набор связей и взаимодействий между ними [39]. На этой основе была разработана на языках OWL и SWRL онтология в среде Protege. Иерархия классов модели показана на рисунке 20. Свойства объектов показаны на рисунке 21.

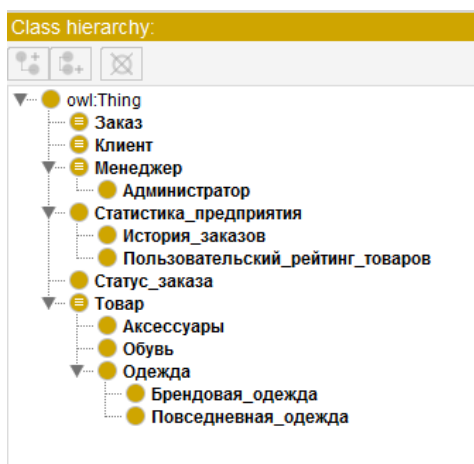


Рисунок 20 – Иерархия классов онтологической модели

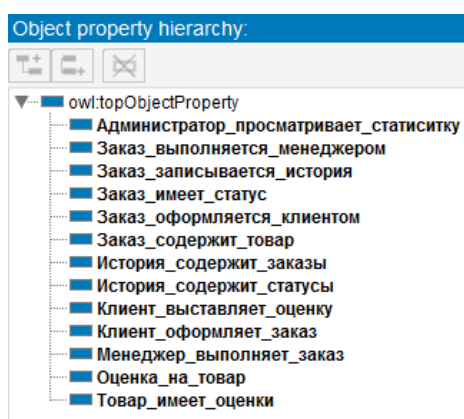


Рисунок 21 – Свойства объектов онтологической модели

Общая схема онтологической модели в виде онтологического графа представлена на рисунке 22.

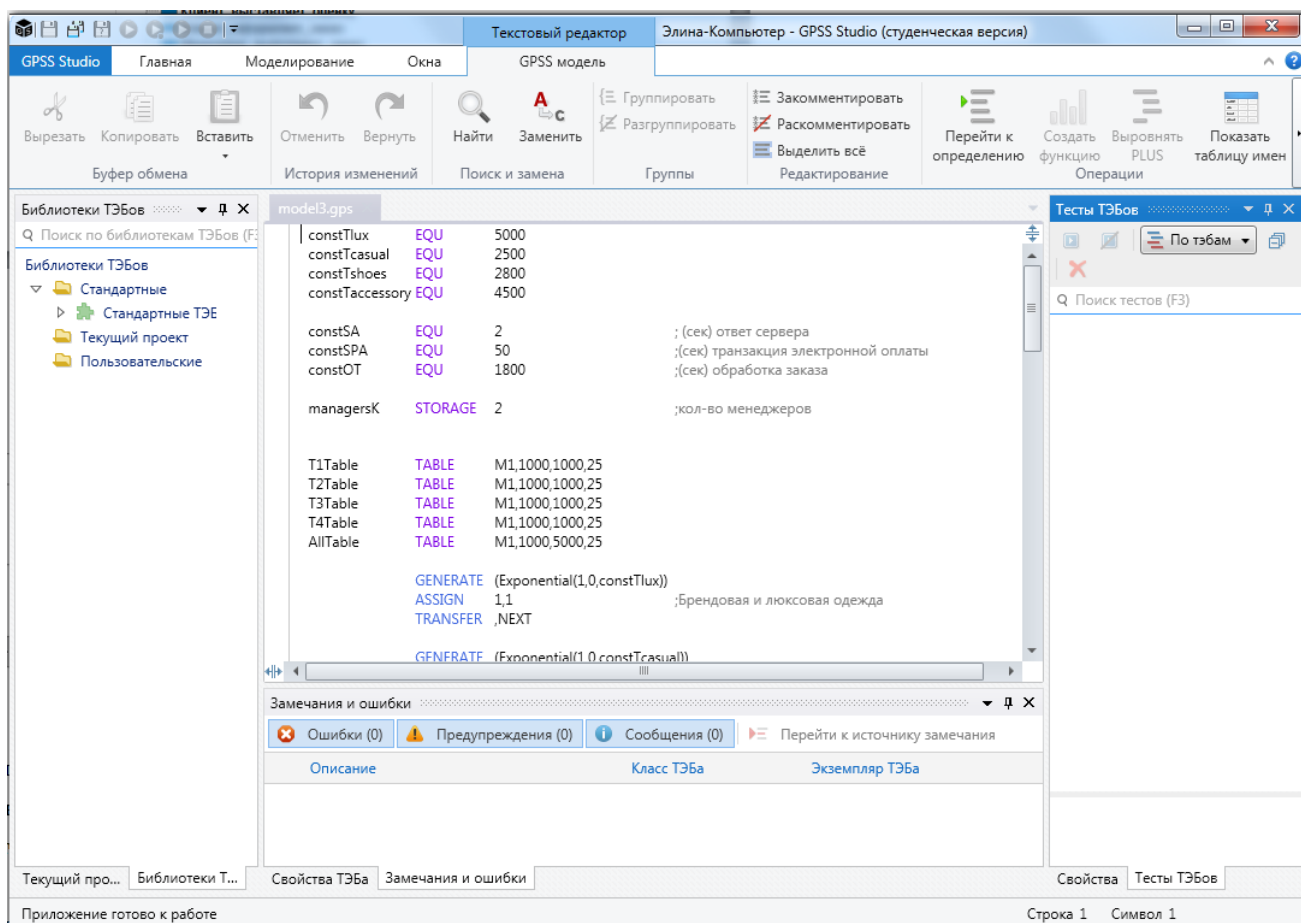


Рисунок 23 – Среда GPSS Studio

Первым этапом моделирования стало создание Q-схемы [41] модели, которая показана на рисунке 24.

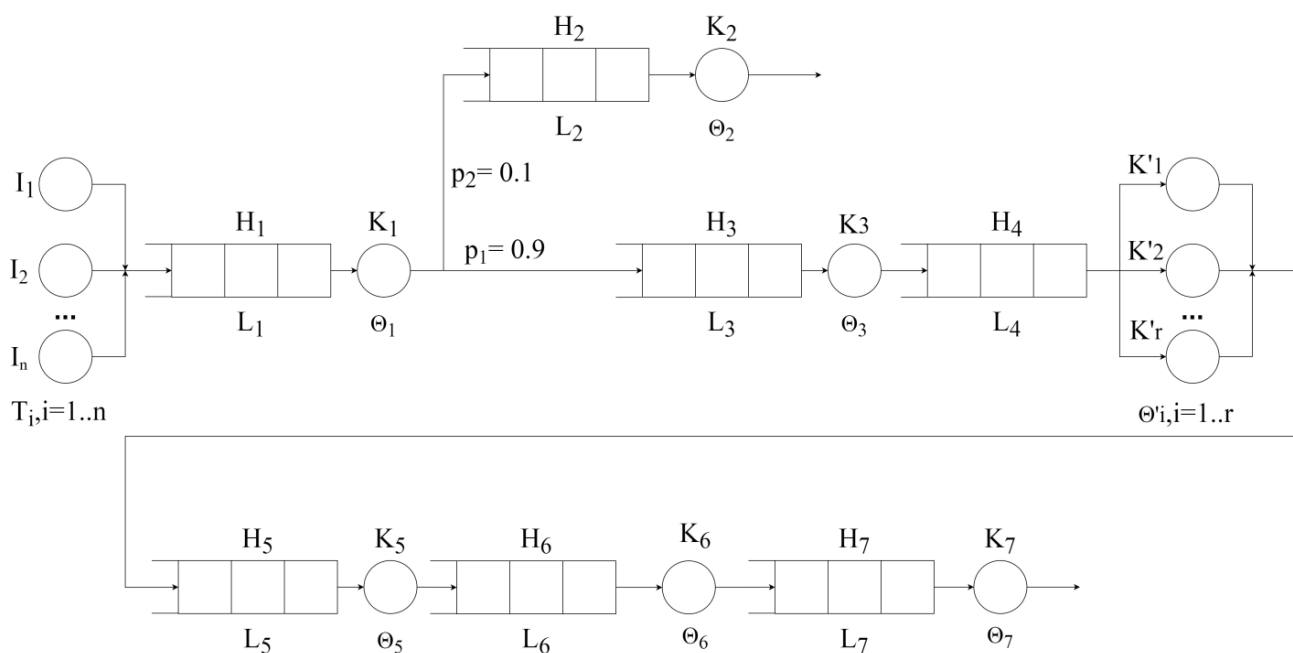


Рисунок 24 – Q-схема модели системы электронной коммерции

Полученная дискретно-событийная GPSS модель представляет собой многофазную сеть массового обслуживания, где потоки заявок проходят через единичные и множественные каналы обслуживания [42–48].

Поток заявок в систему обеспечивается источниками I_1 – I_n , где n – количество наименований товаров, предоставляемых предприятием. Фрагмент информационной системы, обеспечивающий просмотр информации о товаре покупателю, обозначена K_1 .

Вероятностью отсутствия товара выбрано значение:

$$p2 = \frac{1}{m} = 0.1,$$

где m – максимальное количество товаров на складе, то есть закупленная у поставщика партия.

Следовательно, вероятность наличия товара:

$$p1 = 1 - \frac{1}{m} = 0.9.$$

Таким образом, если выбранный пользователем товар отсутствует в наличии (переход с вероятностью $p2$), то информационная система предлагает оповестить пользователя при поступлении данного товара в магазин. Фрагмент информационной системы, ответственный за данный вариант использования представлен обработчиком K_2 .

В обратном случае, если товар есть в наличии (переход $p1$), то в компоненте K_3 происходит регистрация заявки в системе.

После обработки заявки информационной системой, в работу над этой заявкой включаются менеджеры K'_1 – K'_r , где r – число менеджеров. Менеджеры проверяют оплату и готовят товар к отправке покупателю.

Обработчиком K_5 обозначена подсистема контроля за состоянием заказа и отслеживанием отправок. Далее возможно использование различных способов доставки: курьером, службой доставки, почтой или самовывозом. K_6 обозначает службу доставки.

Финальным этапом K_7 является подтверждение получения пользователем товара через электронный механизм отслеживания посылок.

В рассмотренной статистике [49] малых предприятий электронной коммерции преобладают значения, представленные в таблице 10.

Таблица 10 – Параметры модели системы электронной коммерции

Обозначение в модели	Статистическое значение	Описание
n	80	Количество различных видов товаров.
m	10	Размер партии каждого товара при закупки у поставщика.
NM	3	Количество менеджеров.
E	1 сек	Единица модельного времени.
t	15 мин	Интервал поступления.
T	1 месяц	Временной интервал моделирования.
TD	3 дня	Средний срок доставки коммерческой службой доставки (с учетом доставки по всей стране).
SA	2 сек	Среднее время обработки операции в серверном приложении, запущенном на серверном фреймворке Apache TomEE 8.0
SPA	30 сек	Среднее время обработки транзакции оплаты через прикладной веб интерфейс онлайн банкинга.
OT	30 мин	Среднее время обработки заказ.

Входной поток заявок является стационарным потоком без последствия. В модели преобладают признаки показательного (он же экспоненциальный) закона распределения непрерывной величины, так как достаточно только получить время между текущей заявкой и следующей. Закон распределения описывается следующим образом:

$$P(x) = \begin{cases} 0, & x < 0 \\ 1 - e^{-\lambda x}, & x > 0 \end{cases},$$

где λ – интенсивность потока заявок, для данной модели $\lambda = \frac{1}{t}$.

Исходя из параметров созданной модели, была проведена ее реализация на языке GPSS средствами программы GPSS Studio.

Основной частью финансовых расходов в данной модели предметной области является показатель зарплаты менеджеров.

Следовательно, необходимо определить оптимальное количество менеджеров в построенной системе. Результат данного исследования приведен в таблице 11.

Таблица 11 – Оптимизация количества менеджеров

Количество менеджеров	Оценка длины очереди заказов	Оценка времени пребывания заказа в системе	Финансовые затраты на зарплаты включая налоги
2	Длина очереди достигает 140 заказов, что не является приемлемым.	Время обработки заказа достигает 2,5 часа, что не является приемлемым (рисунок 25).	80 тыс. руб.
3	Длина очереди приемлема.	Время пребывания заказа в системе достигает 15 мин. Данный показатель является приемлемым (рисунок 26).	120 тыс. руб.
4	Длина очереди приемлема.	Время пребывания заказа в системе достигает 10 мин. Данный показатель является приемлемым (рисунок 27).	160 тыс. руб.

Рассмотренные на рисунках ниже гистограммы рассчитаны средствами GPSS Studio.

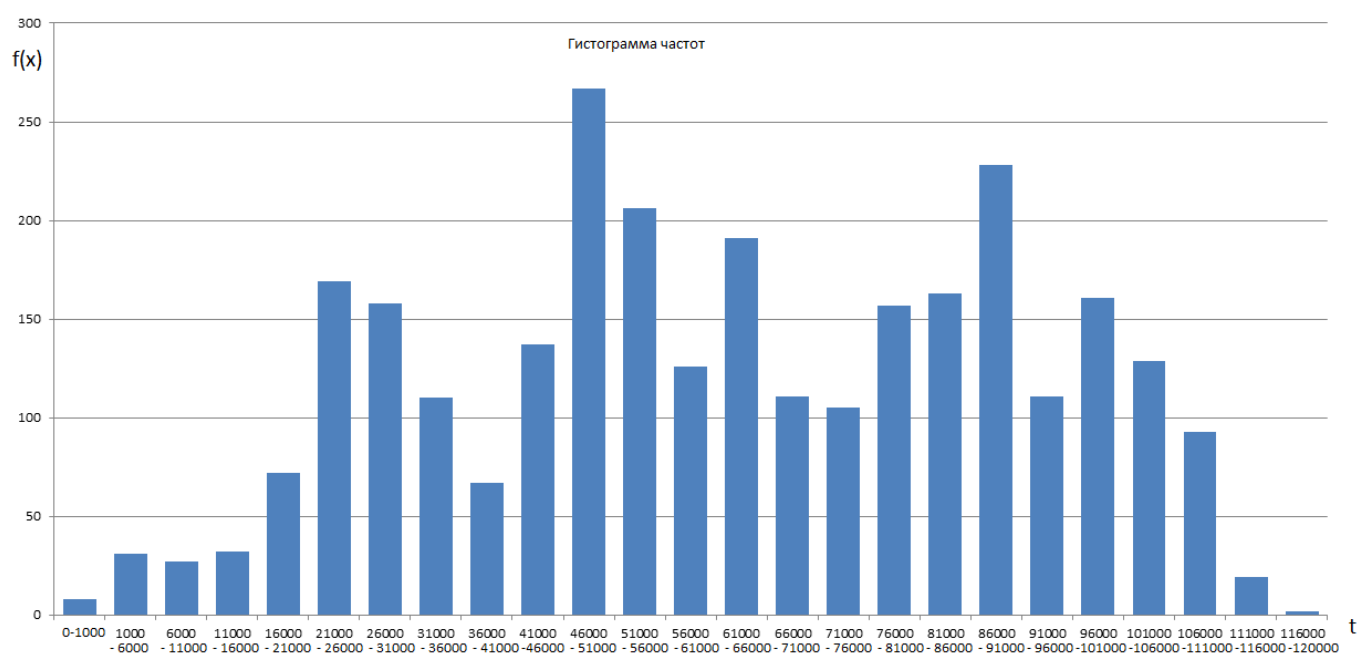


Рисунок 25 – Гистограмма частот времени пребывания заказа в системе при работе 2 менеджеров

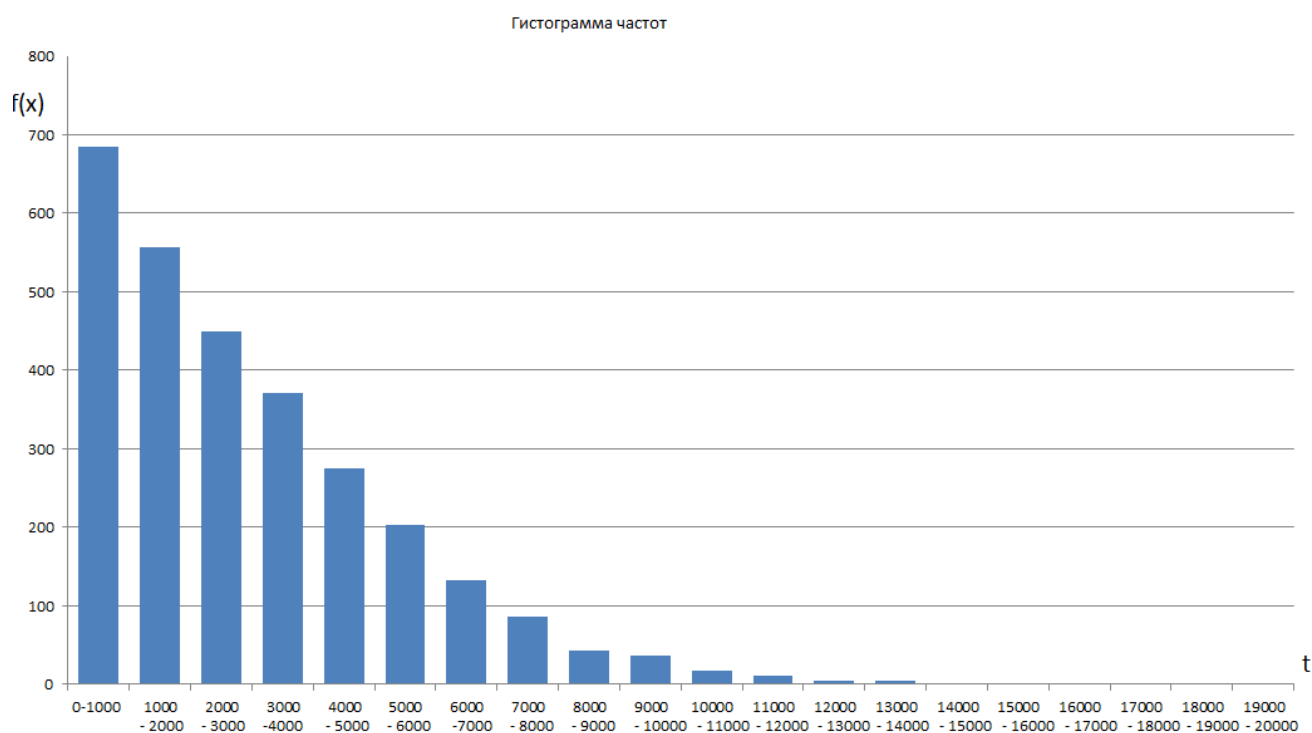


Рисунок 26 – Гистограмма частот времени пребывания заказа в системе при работе 3 менеджеров

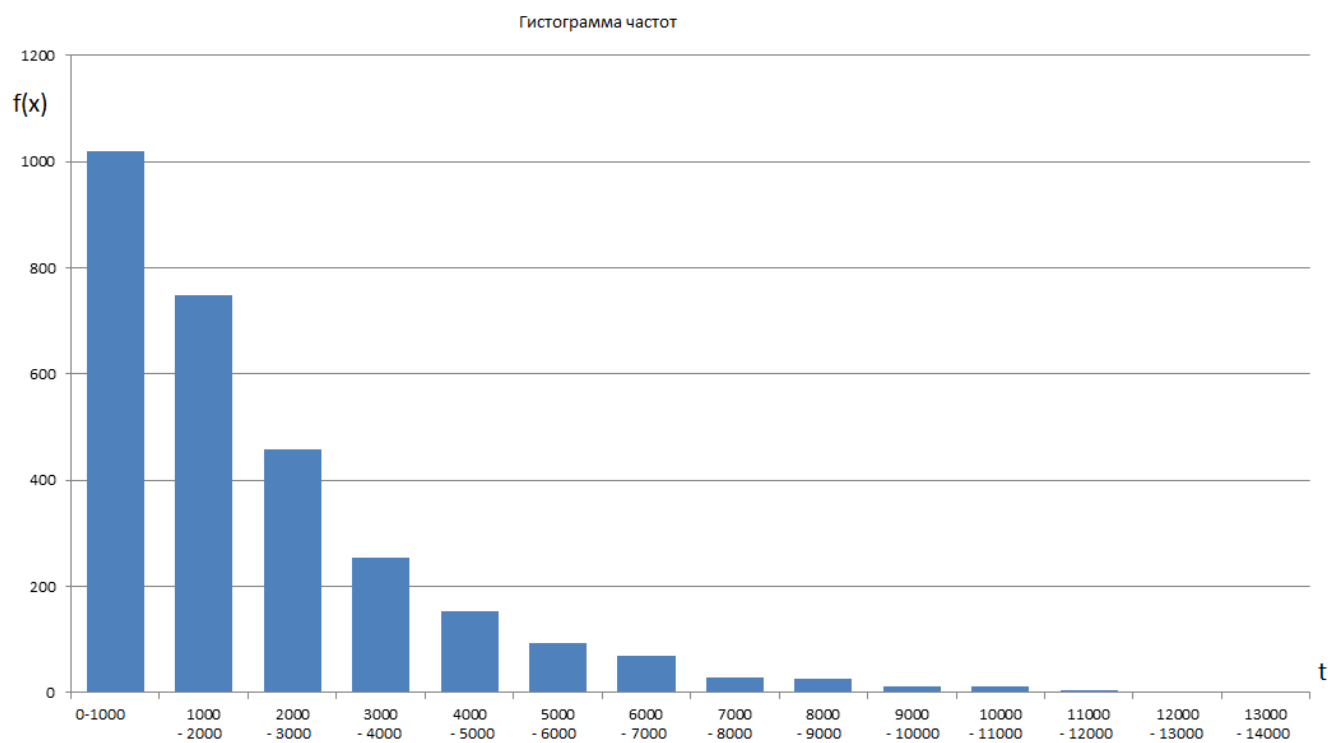


Рисунок 27 – Гистограмма частот времени пребывания заказа в системе при работе 4 менеджеров

Исходя из полученных показателей, можно сделать вывод, что оптимальным с финансовой точки зрения количеством менеджеров является три работника.

Также данная модель может быть использована для вычисления среднего времени пребывания заявок в системе по каждой категории товаров отдельно. Для оптимального режима работы системы с тремя менеджерами характерны следующие показатели.

На рисунке 28 представлена диаграмма частот заказов на товары категории «Брендовая и люксовая одежда».

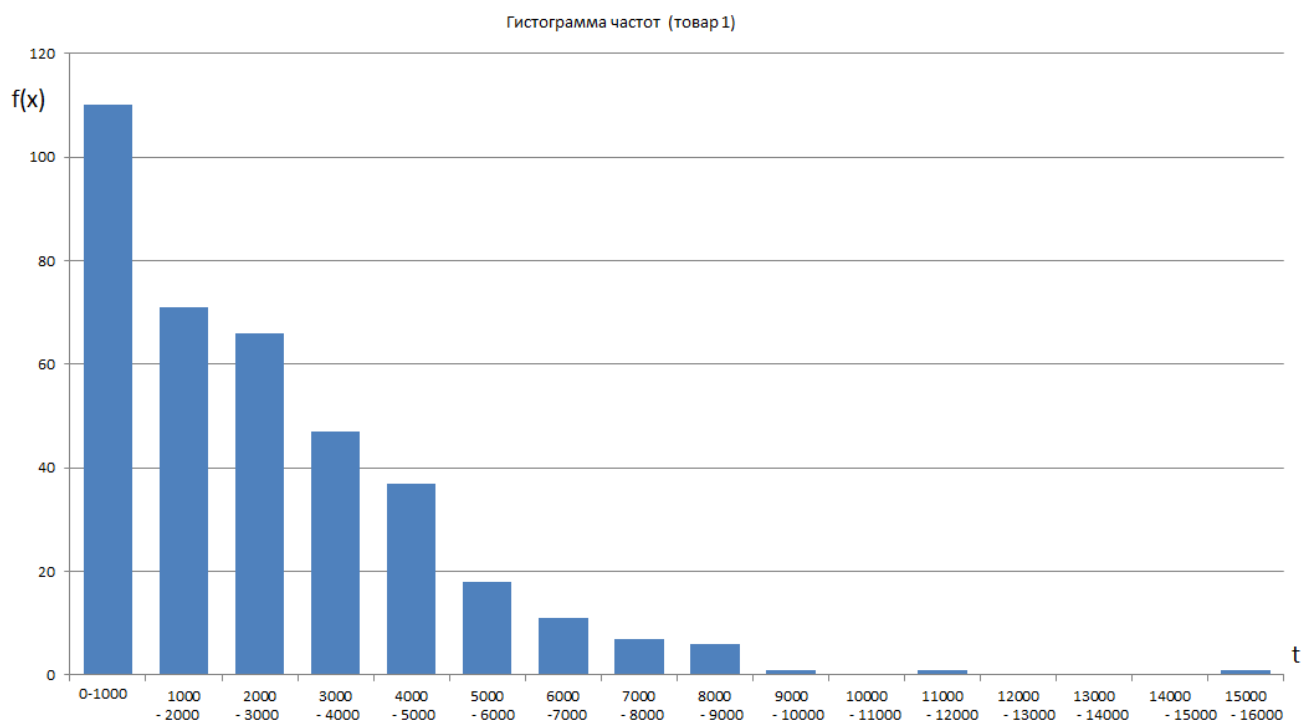


Рисунок 28 – Гистограмма частот времени пребывания в системе заказов на товары категории «Брендовая и люксовая одежда»

На следующем рисунке 29 представлена диаграмма частот заказов на товары категории «Повседневная одежда».

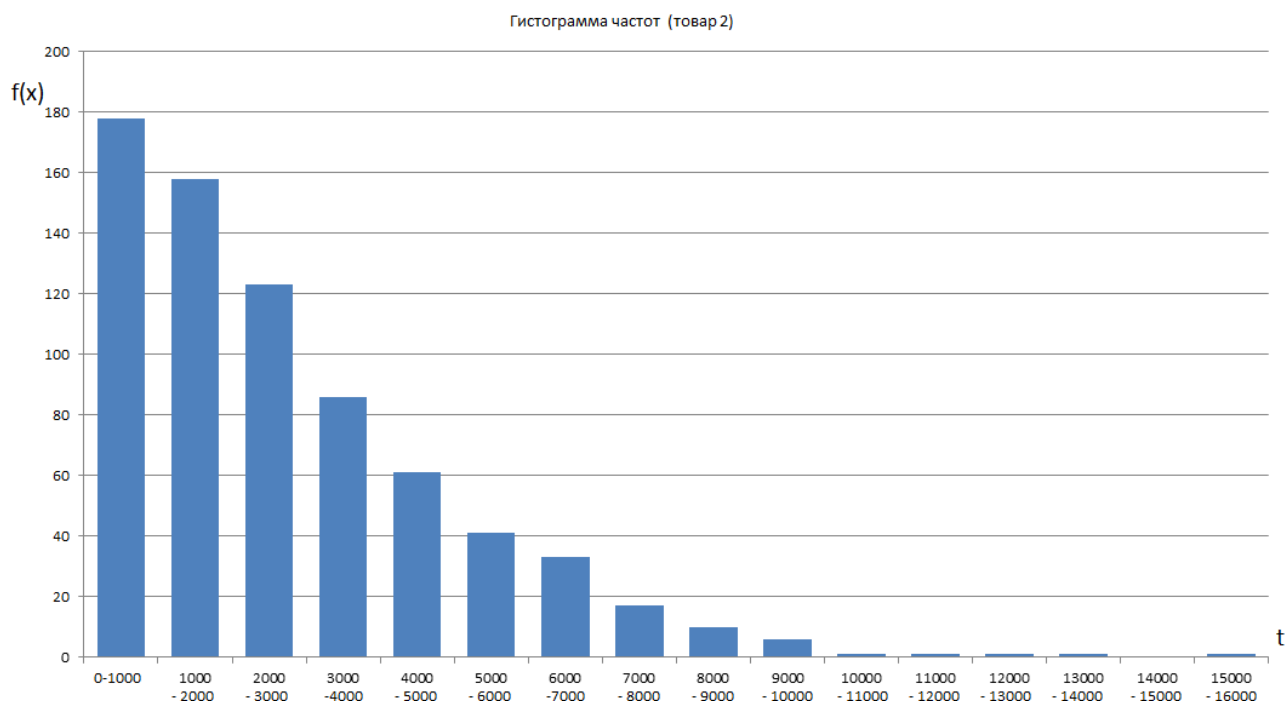


Рисунок 29 – Гистограмма частот времени пребывания в системе заказов на товары категории «Повседневная одежда»

На следующем рисунке 30 представлена диаграмма частот заказов на товары категории «Обувь».

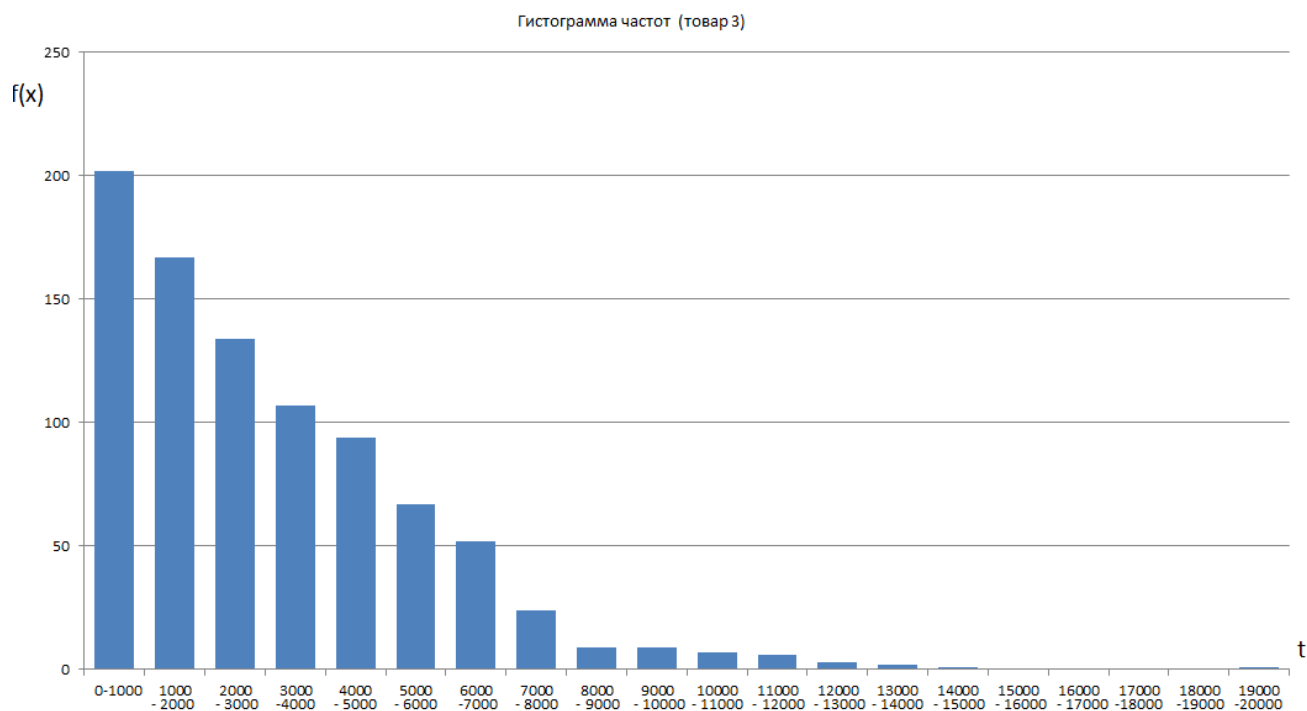


Рисунок 30 – Гистограмма частот времени пребывания в системе заказов на товары категории «Обувь»

На рисунке 31 представлена диаграмма частот заказов на товары категории «Аксессуары».

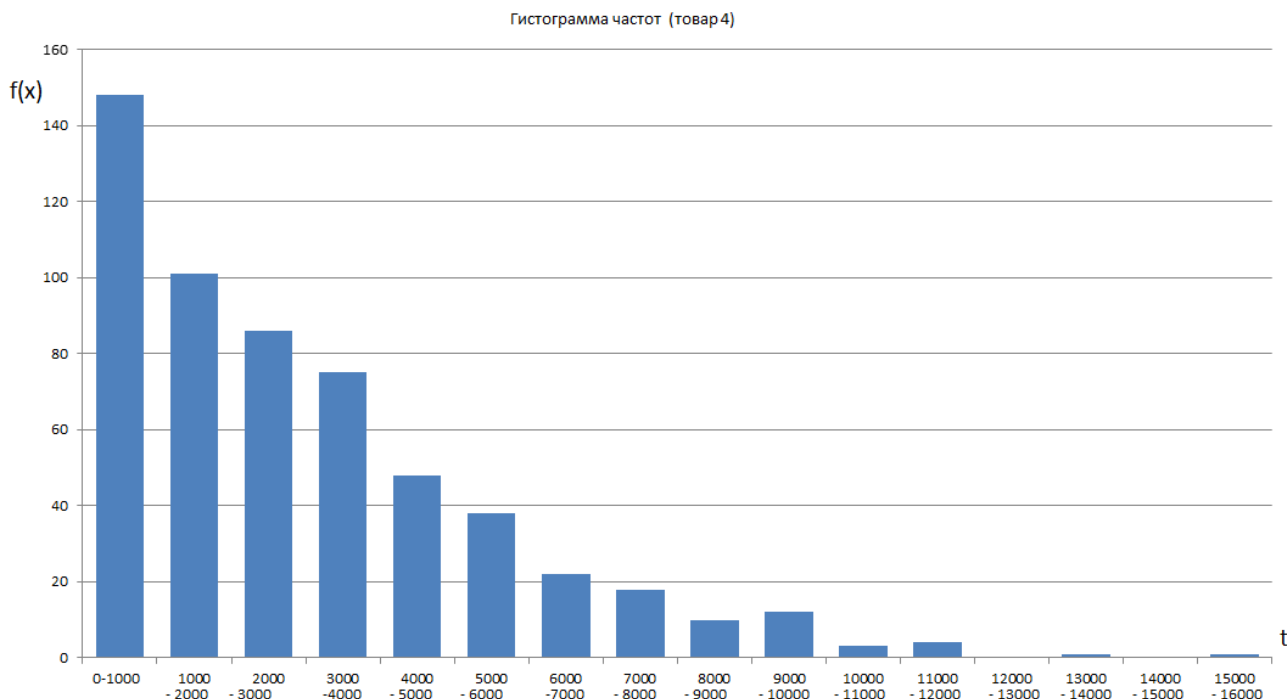


Рисунок 31 – Гистограмма частот времени пребывания в системе заказов на товары категории «Аксессуары»

На рисунке 32 показаны значения математического ожидания и среднеквадратического отклонения времени присутствия заявки в системе.



Рисунок 32 – График мат. ожидания и среднеквадратического отклонения

На рисунке 33 показан график эмпирической функции распределения для случайной величины времени генерации заявок.

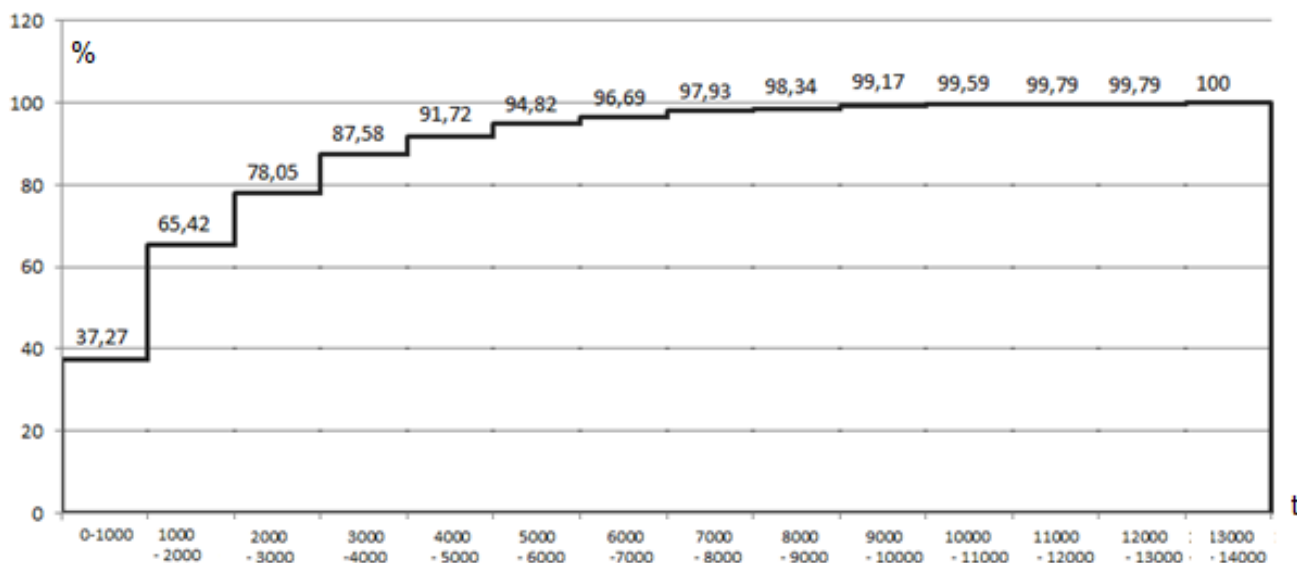


Рисунок 33 – Эмпирическая функция распределения

Еще одной возможностью применения имитационного моделирования в данной предметной области является оптимизация процесса доставки товара службой доставки предприятия электронной коммерции. Стоит отметить, что данное моделирование имеет своей целью оптимизацию количества работников службы доставки, поэтому моделирование целесообразно только для варианта доставки товара службой доставки. Другие способы доставки в данной модели не учитываются [50]. Q – схема модели службы доставки представлена на рисунке 34.

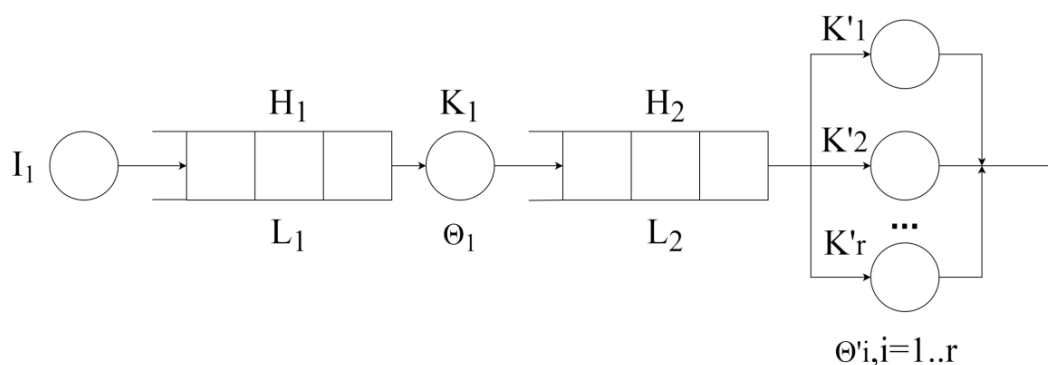


Рисунок 34 – Q-схема модели службы доставки

Поток заявок в данную модель обеспечивается источником I_1 . Фрагмент информационной системы, обеспечивающий регистрацию и управление доставкой товаров, обозначен обработчиком K_1 .

Очередь H_2 аккумулирует товары, предназначенные для доставки клиенту. Средняя длина данной очереди на прямую зависит от количества работников службы доставки, обозначенных обработчиками $K'_1 - K'_r$, где r – оптимизируемое число работников службы доставки.

Определим оптимальное количество доставщиков в моделируемой системе. Результат данного исследования приведен в таблице 12.

Таблица 12 – Оптимизация количества доставщиков

Количество доставщиков	Оценка длины очереди	Оценка времени пребывания заказа в системе	Финансовые затраты на зарплаты включая налоги
2	Длина очереди достигает 200 заказов, что не является приемлемым.	Время обработки заказа достигает 5 часов, что не является приемлемым.	60 тыс. руб.
3	Длина очереди приемлема.	Время пребывания заказа в системе достигает 18 мин. Данный показатель является приемлемым (рисунок 35).	90 тыс. руб.
4	Длина очереди приемлема. Рисунок.	Время пребывания заказа в системе достигает 10 мин. Данный показатель является приемлемым.	120 тыс. руб.

Данные показатели также рассчитаны средствами GPSS Studio.

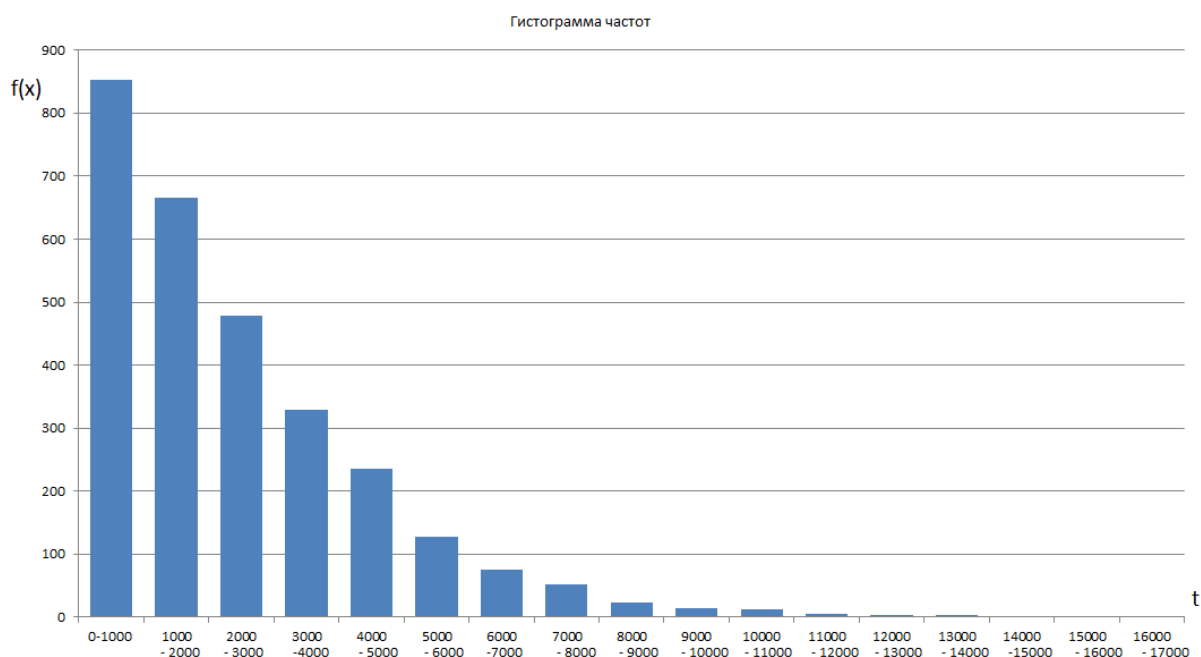


Рисунок 35 – Гистограмма частот времени пребывания заявки в системе

Таким образом, результаты моделирования показывают, что оптимальный режим работы службы доставки предприятия достигается при работе трех доставщиков.

На рисунке 36 представлены графики математического ожидания времени пребывания заявки в системе и среднеквадратического отклонения.



Рисунок 36 – График мат. ожидания и среднеквадратического отклонения

На рисунке 37 показан график эмпирической функции распределения для случайной величины времени генерации заявок.

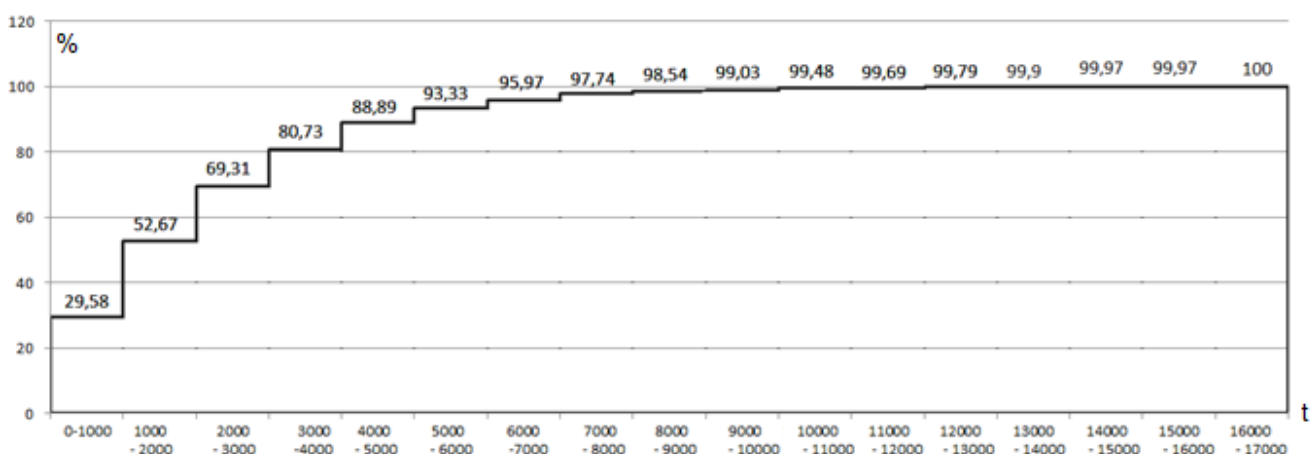


Рисунок 37 – График мат. ожидания и среднеквадратического отклонения

Исходя из данного графика, закон распределения – экспоненциальный.

Таким образом, полученные результаты имитационного моделирования позволяют выбрать оптимальные с экономической точки зрения значения количества работников – три менеджера и три работника службы доставки.

2.4 Оценка адекватности модели

Оценим адекватность имитационной модели службы доставки, так как данная модель сочетает особенности взаимодействия программных средств системы электронной коммерции и человеческого фактора работников службы доставки. Рассматриваемая модель представлена на рисунке 38.

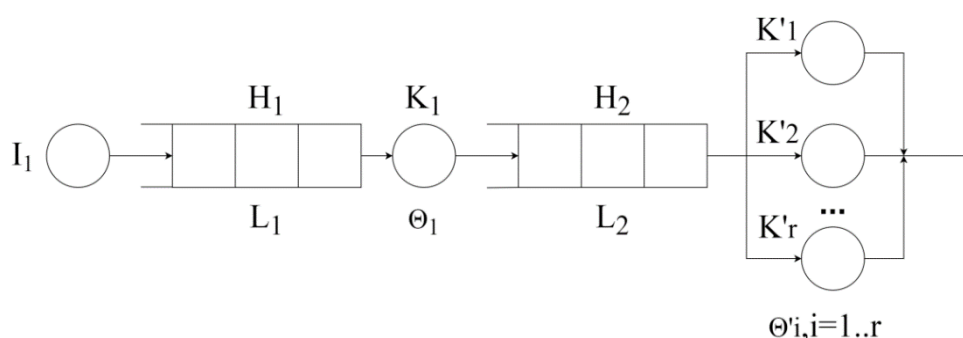


Рисунок 38 – Q-схема модели службы доставки

Данная модель представляет собой сеть массового обслуживания. Для оценки адекватности необходимо провести валидацию модели. Для этого рассчитаем теоретические значения результатов моделирования, а затем сравним их с данными, полученными в ходе вычислительного эксперимента в среде GPSS World. Для вычисления расчетных значений применим математический аппарат теории массового обслуживания [51,52].

Пусть интенсивность входящего потока:

$$\lambda = \frac{1}{900} = 0,0011.$$

Тогда, коэффициент загрузки системы:

$$\gamma = \frac{\lambda}{N\mu} = \frac{0,0011}{3 * 0.00055} = 0,66,$$

где μ – интенсивность потока обслуживания;

N – количество работников службы доставки, в данном случае $N = 3$.

Среднее число занятых доставщиков:

$$\bar{N}_{\text{зан}} = \alpha = \frac{\lambda}{\mu} = \frac{0,0011}{0,00055} = 2.$$

Среднее число свободных доставщиков:

$$\bar{N}_{\text{св}} = N - \bar{N}_{\text{зан}} = 3 - 2 = 1.$$

Вероятность задействия всех доставщиков без возникновения очереди:

$$P_N = P_3 = \frac{\frac{\alpha^3}{3!}}{\sum_{j=0}^3 \frac{\alpha^j}{j!} + \frac{\alpha^3}{3!} * \frac{\gamma}{1-\gamma}} = \frac{\frac{2^3}{3!}}{\sum_{j=0}^3 \frac{2^j}{j!} + \frac{2^3}{3!} * \frac{0,66}{1-0,66}} = 0,15.$$

Средняя длина очереди:

$$\bar{r} = P_N * \frac{\gamma}{(1-\gamma)^2} = 0,15 * \frac{0,66}{(1-0,66)^2} = 0,85.$$

Среднее время пребывания заявок в очереди:

$$\bar{T}_{\text{оч}} = \frac{r}{\lambda} = \frac{0,85}{0,0011} = 772,72 \text{ (сек.)}.$$

Среднее число заявок в системе:

$$\bar{N}_{\text{сис}} = \bar{N}_{\text{зан}} + \bar{r} = 2 + 0,85 = 2,85.$$

Среднее время пребывания заявок в системе:

$$\bar{T}_{\text{сис}} = \frac{\bar{N}_{\text{сис}}}{\lambda} = \frac{2,85}{0,0011} = 2591 \text{ (сек.)}.$$

Получив данные расчетные значения модели, рассмотренной в качестве сети массового обслуживания, сравним их с практическими. В таблице 13 представлено данное сравнение.

Таблица 13 – Валидация имитационной модели

Показатель	Расчетное значение	Модельное значение	Расхождение
Коэффициент загрузки системы	0,66	0,605	8,33%
Среднее число свободных доставщиков	1	1.1	10%
Среднее время пребывания требований в системе	2591	2432	6,14%

Результат валидации показывает расхождение результатов моделирования с расчетными теоретическими значениями менее 10% для каждого из верифицируемых значений, что подтверждает адекватность созданной имитационной модели службы доставки.

Аналогичная проверка адекватности путём валидации была проведена для второй модели, которая так же показала расхождение менее 10% и подтвердила адекватность имитационной модели.

2.5 Концептуальное и логическое проектирование базы данных

На основе анализа предметной области и функциональных требований была построена концептуальная модель базы данных системы электронной коммерции в виде диаграммы «сущность-связь», которая представлена на рисунке 39. Подробное описание сущностей представлено в приложении В.

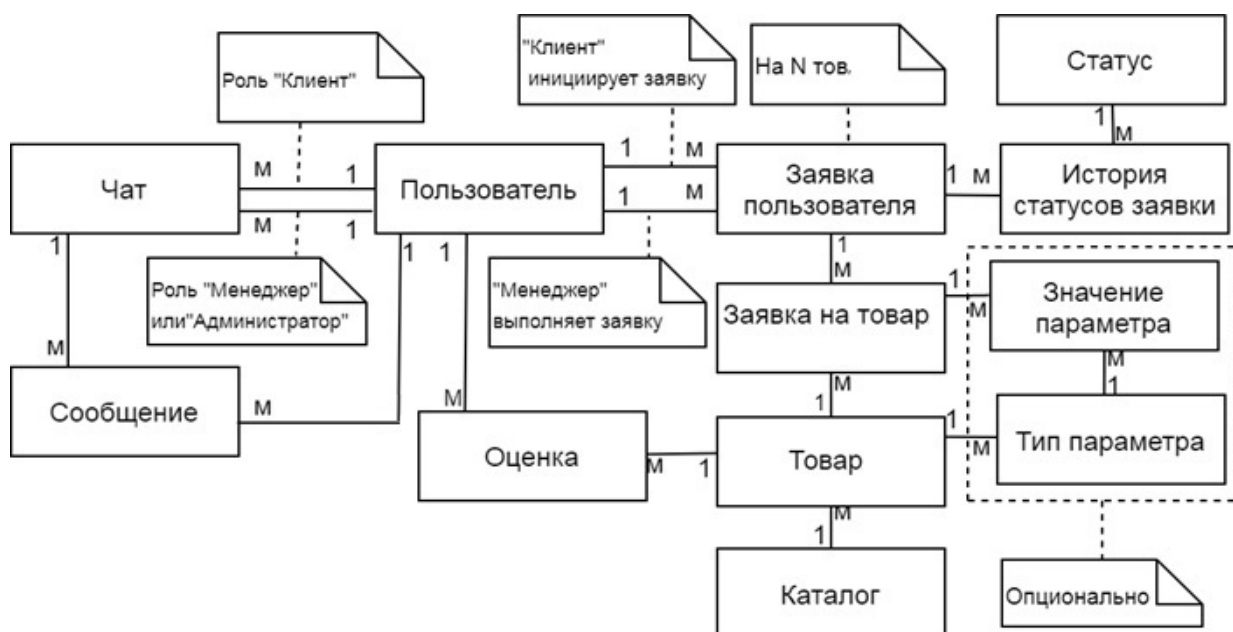


Рисунок 39 – Диаграмма БД в нотации «сущность-связь»

Подробное описание сущностей представлено в таблице В.1. Связи между сущностями описаны в таблице В.2.

Последним этапом разработки концептуальной модели было проведено проектирование атрибутов сущностей. Атрибуты сущностей представлены в таблицах В.3 – В.12.

На данном уровне моделирования были спроектированы только атрибуты, несущие информационную нагрузку для каждой сущности, то есть, не рассматривая фиктивные атрибуты и внешние ключи.

Используя концептуальную модель, была создана логическая модель базы данных с помощью программы проектирования и документирования баз данных ERwin Data Modeler.

Созданная с помощью автоматизированных средств документации ERwin Data Modeler таблица сущностей логической модели базы данных представлена как таблица 14.

Таблица 14 – Таблица сущностей логической модели

Название	Описание
Заявка_на_товар	Заявка на конкретный товар. Обеспечивает связь между заявкой пользователя и товаром.
Заявка_пользователя	Включает в себя товары которые пользователь одновременно заказал
Значение_параметра	Значение для текущего товара в заявке
История_статусов	История изменения статусов заявки
Каталог	Сущность, объединяющая класс товаров
Оценка	Оценка и отзыв клиента о товаре
Пользователь	Записи пользователей информационной системы
Сообщение	Сообщение в процессе чата
Статус	Статус выполнения заявки
Тип_параметра	Желаемый параметр товара
Товар	Основной объект бизнес-процессов
Чат	Диалог между клиентом и менеджером

Полученная логическая модель в виде IDEF1X-диаграммы продемонстрирована на рисунке 40.

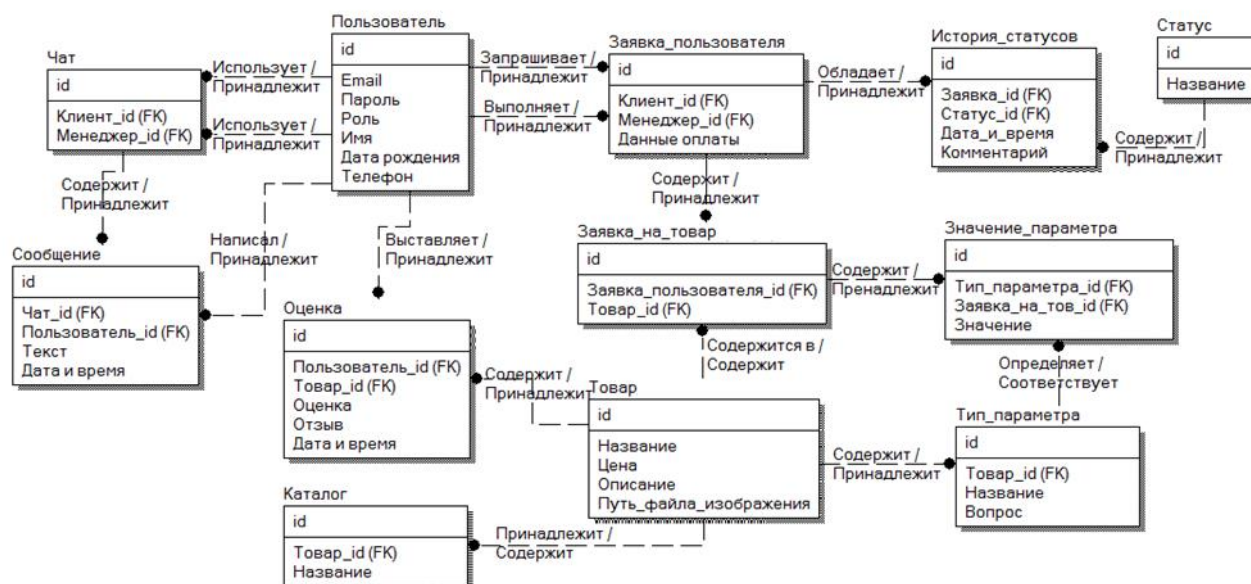


Рисунок 40 – IDEF1X-диаграмма логической модели базы данных

Стоит отметить, что данная модель является универсальной и, несмотря на то, что выражена в контексте реляционной модели базы данных, не отражает деталей конкретной реализации под определенную базу данных.

2.6 Проектирование серверной части системы электронной коммерции

В архитектуре разрабатываемой системы электронной коммерции серверная часть играет роль посредника между мобильным клиентским приложением и базой данных. Серверная часть выполнена на основе RESTful технологий Java Web Services [53]. Веб-сервисы предлагают удобный интерфейс передачи данных через прикладной протокол прикладного уровня HTTP с использованием формата данных JSON.

Кроме этого, технологии Java Web Services предоставляют инструменты для преобразования данных из формата JSON в объекты – модели. Преобразованием можно управлять по средствам классов-адаптеров.

Серверная часть системы представляет собой набор классов – сервисов, автоматически определяемых технологией Java Web Service. Каждый такой сервис может служить входной точкой в программу.

Исходя из выбранных технологий и требований, для серверной программы выбрана следующая архитектура. Все классы приложения делятся на три слоя: модели, фасады и веб-сервисы.

Веб-сервисы определяют точки входа в программу по средствам обращения к ним через HTTP запрос. Метод запроса, URL путь и возвращаемые данные определяются в сервисах через нотации из библиотеки JAX-RS.

Пример использования нотаций для объявления веб-сервиса представлен ниже:

```
@GET
@Path("getFullInfoForGood/{good}")
@Produces({MediaType.APPLICATION_JSON})
public FullGoodInfo getFullGoodInfo(@PathParam("good") UUID good) {
    return goodFacade.getFullInfoForGood(good, 50, 0);
}
```

Фасады предоставляют методы доступа к хранилищам данных, например, к базе данных или к файлам. Для запросов к базе данных использована библиотека Java Persistence. Библиотека включает в себя Object-Relational Mapping (ORM) технологию и предоставляет три возможных варианта доступа к базе данных [20]:

- прямой запрос в базу данных, используя язык SQL;
- запрос на языке Java Persistence Query Language (JPQL), язык JPQL является расширением языка SQL, позволяющим использовать в запросе сущности из контекста Java-программы;
- запрос через цепочку вызова методов ORM-библиотеки.

Пример запроса на языке JPQL с использованием Java Persistence:

```
EntityManager
.createQuery("SELECT mes FROM Message mes WHERE mes.chatUuid.uuid = :mess_p", Message.class)
.setParameter("mess_p", chat)
.getResultList();
```

Модели выполняют сразу несколько функций: являются моделями данных предметной области, управляют отражением сущностей базы данных в Java-объектах, управляют преобразованием объектов в JSON-строку и обратно.

Пример модели представлен ниже:

```
public class Status extends AbstractEntity implements Serializable
{
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "uuid")
    @JsonbTypeAdapter(UuidJsonConverter.class)
    @Convert("uuidConverter")
```

Как можно видеть, в модели использовано множество атрибутов.

На рисунке 41 представленная диаграмма классов серверной части системы электронной коммерции.

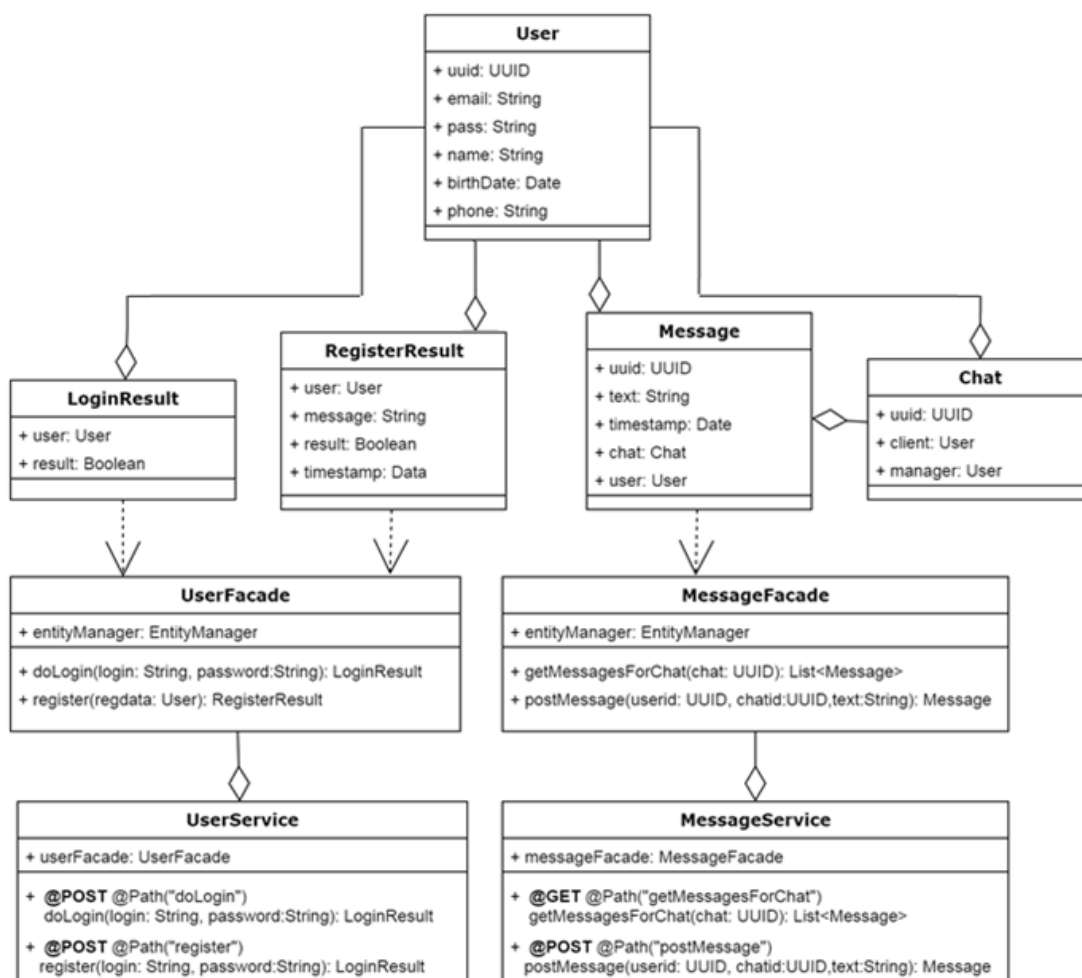


Рисунок 41 – Диаграмма классов серверной части системы электронной коммерции

Диаграмма создана с использованием языка UML. На данной диаграмме классы сгруппированы по признаку принадлежности их к слою архитектуры приложения. Таким образом, классы UserService, MessageService принадлежат к сервисам. Классы UserFacade, MessageFacade принадлежат к фасадам. Классы User, LoginResult, RegisterResult, Message, Chat являются моделями. Описание классов, присутствующих на диаграмме классов представлено в таблице 15.

Таблица 15 – Описание классов

Название класса	Описание
UserService	Сервис пользователя. Содержит методы для входа в систему и создания учетной записи пользователя, доступные для вызова через HTTP запрос. Содержит ссылку на соответствующий фасад пользователя.
MessageService	Сервис сообщений. Содержит метод загрузки сообщений для чата и метод отправки сообщения, доступные через HTTP запрос. Содержит ссылку на соответствующий фасад сообщений.
UserFacade	Фасад пользователя. Предоставляет доступ к запросам в базу данных для входа в систему. Содержит ссылку на объект типа EntityManager, предоставляющий доступ к базе данных.
MessageFacade	Фасад сообщений. Предоставляет доступ к запросам в базу данных для загрузки сообщений и отправки сообщений в базу данных. Содержит ссылку на объект типа EntityManager, предоставляющий доступ к базе данных.
User	Модель пользователя. Содержит UUID, логин, пароль, имя, дату рождения, телефонный номер пользователя.
LoginResult	DTO(Data Transfer Object), содержащий данные пользователя. Содержит ссылку на модель пользователя.
RegisterResult	DTO, содержащий информацию о создании учетной записи. Содержит ссылку на модель пользователя.
Message	Модель сообщения. Содержит ссылку на модель пользователя, текст сообщения, время отправки.
Chat	Модель чата.

В таблице 16 приведена спецификация веб-сервисов.

Таблица 16 – Описание веб-сервисов

Метод и название	Параметры	Возвращаемые данные	Краткое описание
GET chat/getChatsByClient	Заголовок запроса: Id пользователя	Список чатов клиента	Позволяет получить все чаты для текущего пользователя-клиента
GET chat/ getChatsByManager	Заголовок запроса: Id пользователя	Список чатов менеджера	Позволяет получить все чаты для текущего пользователя-менеджера
PUT chat/createChat	Заголовок запроса: Id клиента Id менеджера	Результат создания чата	Позволяет создать чат между клиентом и менеджером
PUT chat/autoCreateChat	Заголовок запроса: Id клиента	Результат создания чата	Альтернативный способ создания чата, не требующий наличие заявок у данного пользователя
POST feedback/postFeedback	Тело запроса: Объект отзыва в JSON формате	Добавленный отзыв	Позволяет добавить в систему пользовательский отзыв на товар с оценкой
GET file/{path}	URL параметр: Путь к файлу	Поток байт скачиваемого файла	Позволяет скачивать файлы с сервера. А именно изображения товаров
GET good/getGoods	—	Список товаров в случайном порядке	Позволяет загрузить товары без упорядочивания.
GET good/ getGoodsByPopularity	—	Список товаров в порядке по средней оценке	Позволяет загрузить товары, отсортированные по средней оценке.
GET good/ getGoodsByRequests	—	Список товаров в порядке по количеству заявок	Позволяет загрузить товары, отсортированные по количеству заявок.
GET good/ getFullInfoForGood/ {good}	URL параметр: Id товара	Подробная информация о товаре.	Позволяет получить информацию о товаре, и отзывах на него. Необходимо для отображения страницы товара в приложении.
POST good/CreateGood	Тело запроса: Объект товара в формате Json	Результат добавления товара	Добавляет товар в каталог товаров приложения
GET message/ getMessagesForChat	Заголовок: Id чата	Список сообщений чата.	Позволяет получить все сообщения для чата

Продолжение таблицы 16

Название	Параметры	Возвращаемые данные	Краткое описание
POST message/postMessage	Заголовок запроса: Id пользователя Id чата, Текст сообщения,	Созданное сообщение.	Добавить в систему сообщение. Вызывается каждый раз, когда пользователь отправляет сообщение в чате.
POST request/postRequest	Тело запроса: DTO объект в формате JSON	Результат добавления	Добавляет отзыв на товар
GET request/myRequests /{userid}	URL параметр: UUID пользователя	Список заявок для пользователя	Позволяет пользователю загрузить список своих заявок
GET request/ getAllUnfinishedFor Manager/{managid}	URL параметр: UUID пользователя	Список заявок для менеджера	Позволяет менеджеру загрузить список назначенных ему заявок
GET request/ getAllUnfinished	—	Список незавершенных заявок	Позволяет получить список всех незавершенных заявок на данный момент
POST statushistory/ createStatusHistory	Заголовок запроса: UUID заявки, UUID статуса, сообщение	Результат обновления статуса для заявки	Для заявки с указанным UUID в истории статусов заявок будет установлен новый статус с указанным UUID
GET user/getStatistics	—	DTO со статистикой	Получение статистики по выполнению заявок
POST user/doLogin	Заголовок запроса: логин, пароль	DTO результат входа в систему	Метод предоставляет возможность входа в систему.
POST user/register	Тело запроса: Модель user	DTO результат входа в систему	Метод предоставляет возможность создания учетной записи

Таким образом, данные сервисы полностью обеспечивают связь клиентского приложения с серверным приложением и базой данных.

2.7 Проектирование клиентской части системы электронной коммерции

Клиентская часть системы электронной коммерции представляет собой мобильное приложение для ОС Android [54]. Первым этапом стало проектирование пользовательского интерфейса.

Пользовательский интерфейс в Android реализуется в виде макетов, описанных с помощью языка разметки XML. В момент создания активности или фрагмента создается класс Inflater, который распознает структуру XML макета и генерирует свойства, соответствующие каждому элементу интерфейса.

Android использует специальный класс под названием NavController для запуска перехода на экран, идентификатор которого передается в метод navigate.

Иерархия экранов приложения представлена на рисунке 42.

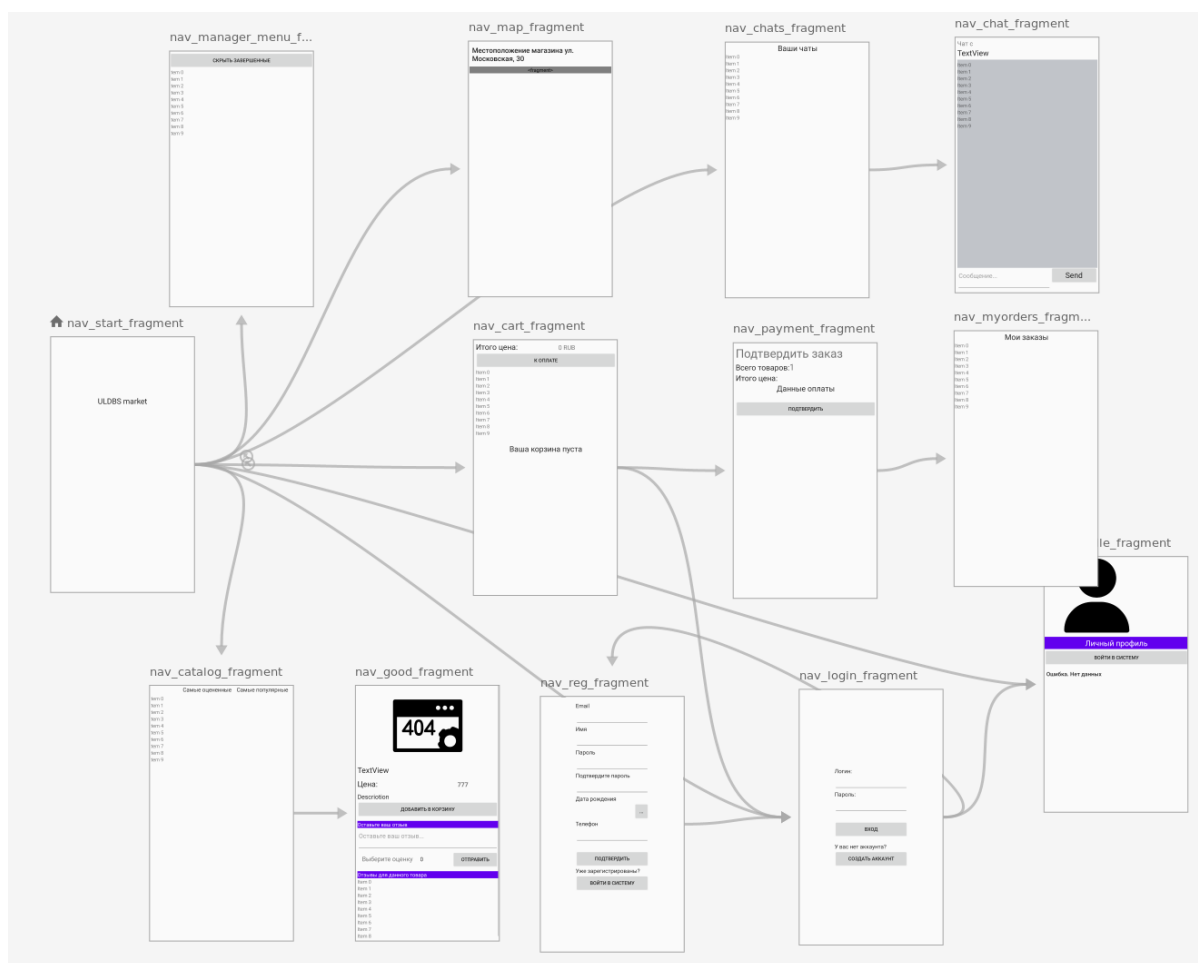


Рисунок 42 – Граф навигации экранов клиентского приложения

Описание экранов приложения представлено в таблице 17.

Таблица 17 – Описание экранов приложения

Название	Описание
nav_start_fragment	Стартовый экран приложения. На нем расположено приветствие и может быть настроено отображение различной информации для пользователя. Например, название и адрес предприятия. Настройка происходит по средствам XML файла ресурса строковых констант.
nav_login_fragment	Экран входа в систему пользователя
nav_register_fragment	Экран создания учетной записи пользователя
nav_profile_fragment	Экран личного кабинета пользователя. Содержит информацию о текущем пользователе и предоставляет функцию выхода из учетной записи.
nav_cart_fragment	Экран корзины с добавленными в нее товарами.
nav_catalog_fragment	Экран каталога товаров. Содержит два варианта загрузки списка товаров: упорядоченные по рейтингу оценок и упорядоченные по количеству заявок.
nav_good_fragment	Экран страницы товара
my_orders_fragment	Экран, на котором пользователь может просматривать свои заявки
nav_payment_fragment	Экран оплаты и регистрации заявки.
nav_chats_fragment	Экран, содержащий список доступных пользователю чатов.
nav_chat_fragment	Экран чата клиента с менеджером.
nav_admin_menu_fragment	Экран меню администратора с отображением статистики и функционалом контроля каталога
nav_manager_menu_fragment	Экран меню менеджера, который позволяет отмечать выполнение заявок
map_fragment	Экран с векторной картой

Описанные выше экраны были спроектированы с помощью встроенного в интегрированную среду разработки Android Studio конструктора макетов. Ниже представлено детальное описание данных макетов.

Рассмотрим макеты экранов создания учетной записи пользователя и входа в систему. Макеты продемонстрированы на рисунке 43.

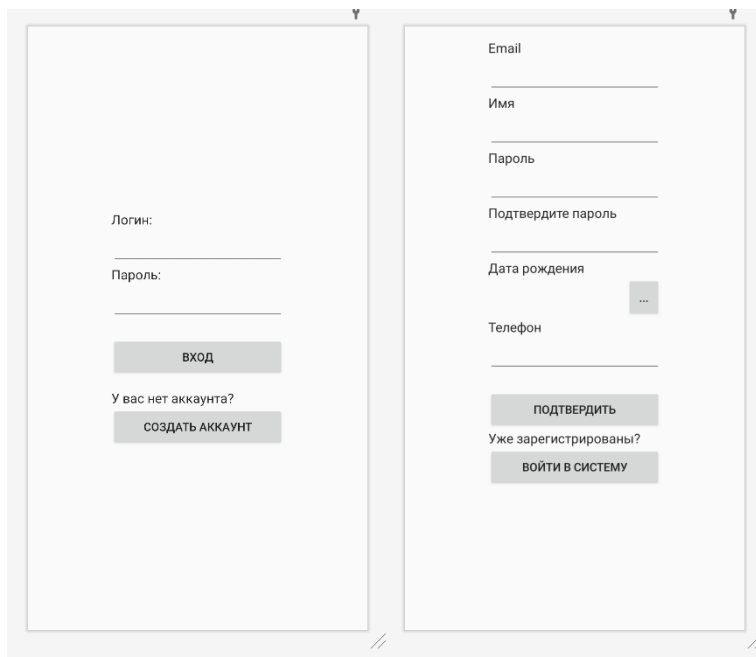


Рисунок 43 – Макеты экранов создания учетной записи и входа в систему

Макеты этих двух экранов содержат всего три типа функциональных элементов: текст, поле ввода и кнопка. На рисунке 44 показан макет экрана чата.

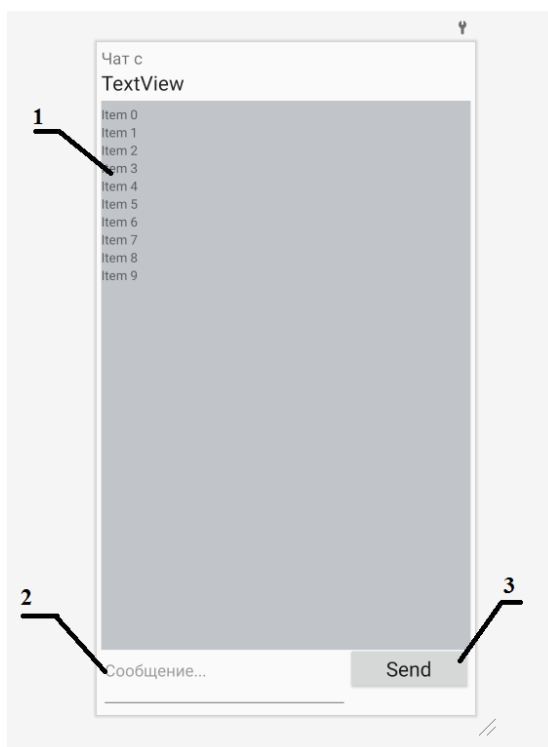


Рисунок 44 – Макет отображения товара в списке

Цифрой 1 на рисунке отображено текстовое поле, где при загрузке чата отображается имя пользователя, с которым происходит диалог. Цифра 2 показывает обновляемый список, настроенный так, чтобы корректно отображать сообщения. Цифра 3 показывает расширяемое поле для ввода сообщения. Цифрой 4 обозначена кнопка отправки сообщения.

На рисунке 45 представлен макет экрана страницы товара.

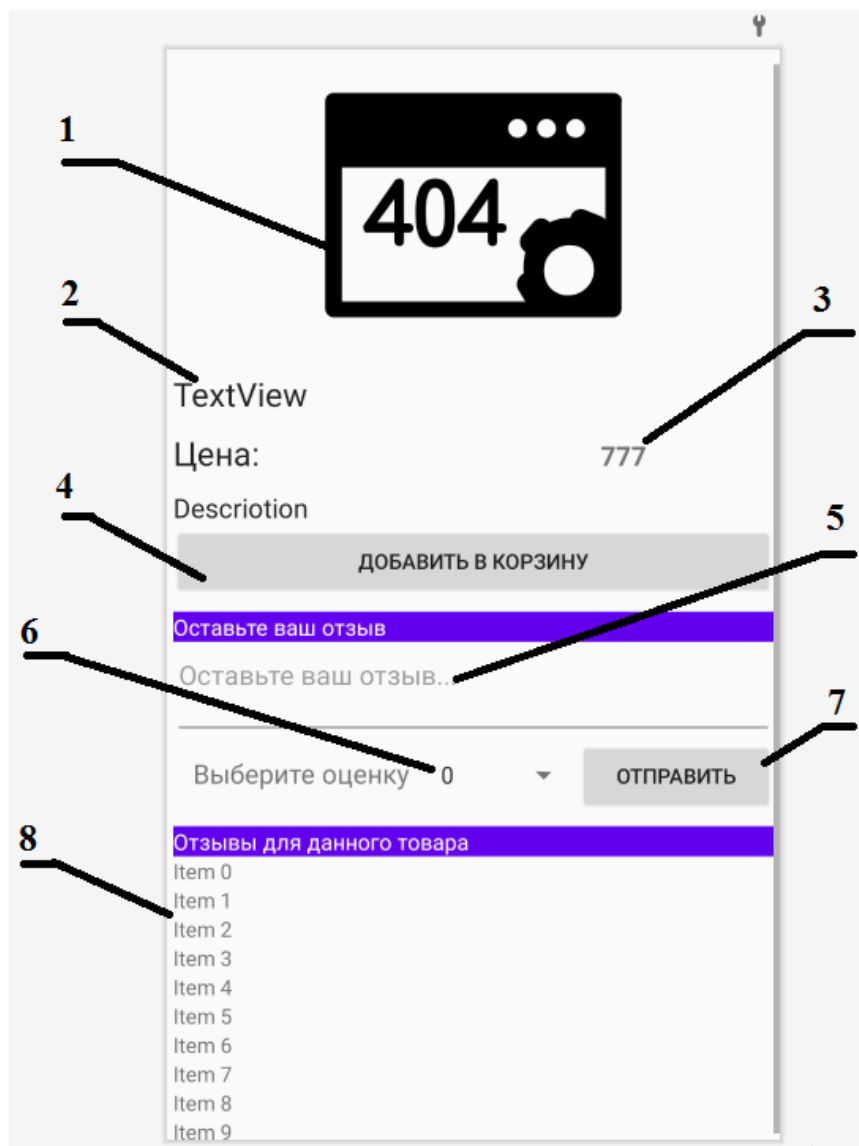


Рисунок 45 – Макет экрана страницы товара

Цифрой 1 обозначено динамически загружаемое изображение товара. Цифрой 2 обозначено текстовое поле названия товара. Цифрой 3 обозначено поле вывода стоимости товара. Цифрой 4 помечена кнопка добавления товара в корзину. Цифра 5 обозначает многострочное расширяемое поле ввода отзыва о

товаре. Цифрой 6 обозначено выпадающее меню, которое предлагает выбрать оценку товара по пятибалльной шкале. Цифрой 7 обозначена кнопка отправки отзыва. Цифрой 8 обозначен обновляемый список, содержащий экземпляры макета отзывов на товар.

На рисунке 46 показан макет экрана меню менеджера.

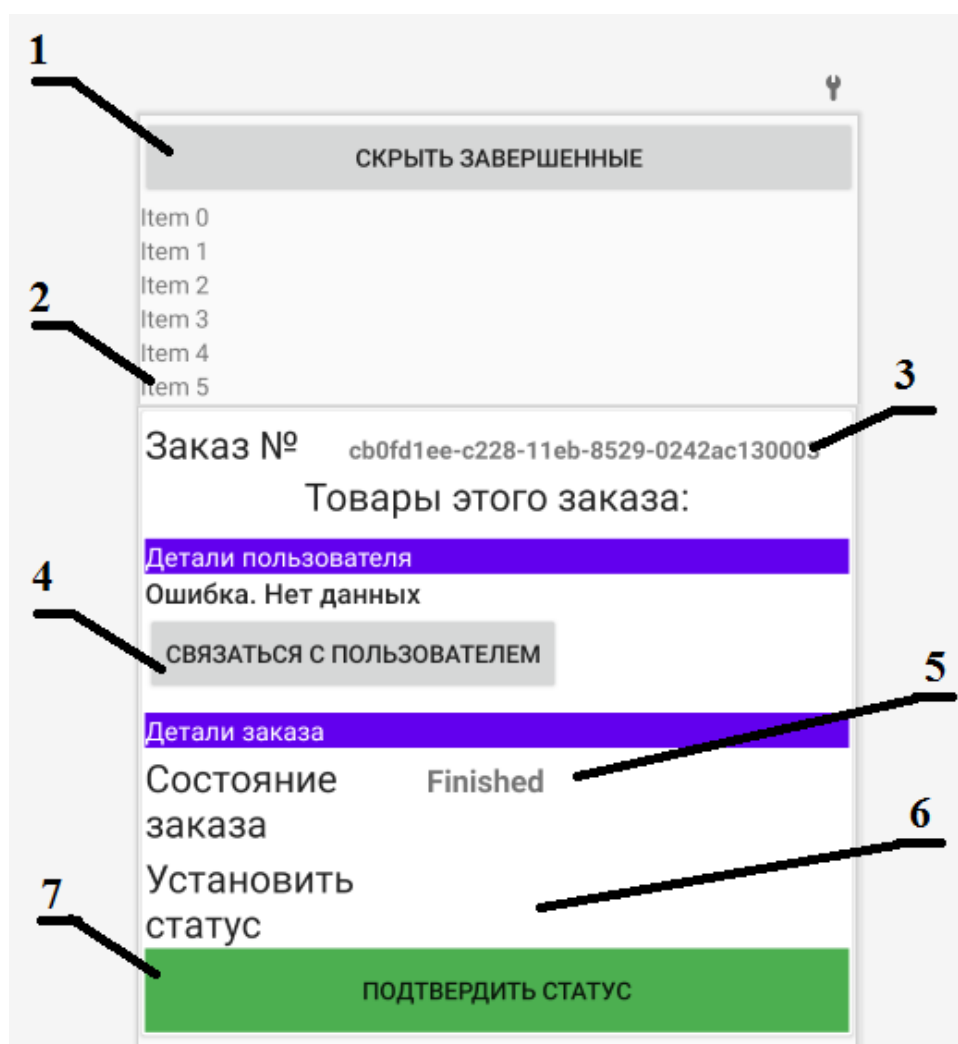


Рисунок 46 – Макет меню менеджера

На данном рисунке цифрой 1 обозначена кнопка фильтрации отображаемых заявок. Цифрой 2 обозначен список, в который загружаются заявки. Цифра 3 показывает поле отображения номера заявки. Цифра 4 обозначает конку перехода к чату с клиентом из этой заявки. Цифра 5 указывает на поле отображения текущего состояния заявки. Цифра 6 показывает на место для выпадающего списка, предоставляющего выбор

нового состояния для заявки. Кнопка 7 показывает на кнопку подтверждения выбора нового состояния для заявки.

Исходя из выбранных технологий, для реализации клиентского мобильного приложения системы электронной коммерции, было решено использовать архитектурный шаблон MVVM. Данный шаблон подразумевает разделение логики приложения на три уровня: view, viewmodel, model.

Слой view отвечает за отображение данных и управление графическими элементами на пользовательском интерфейсе. Слой model предоставляет функционал доступа к данным и структурам данных. Слой viewmodel отвечает за обработку бизнес-логики и за привязку данных на пользовательском интерфейсе.

Схема взаимодействия компонентов модели MVVM представлена на рисунке 47.

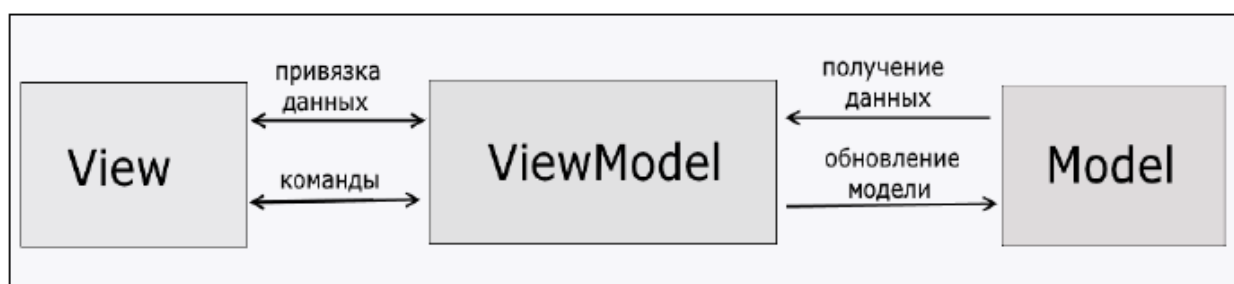


Рисунок 47 – Схема шаблона MVVM

Выбор данного подхода организации архитектурной структуры приложения обусловлен наличием в используемом инструменте Android SDK следующих компонентов [55].

Data Binding Library – библиотека динамической привязки данных на графическом пользовательском интерфейсе к статически скомпилированным программным кодам. Позволяет удобно реализовывать привязку компонента view к компоненту viewmodel по средствам применения поведенческого шаблона «наблюдатель».

LiveData – механизм, позволяющий создавать экземпляры шаблона «наблюдатель», оптимизированные к жизненному циклу компонентов мобильного приложения – активностей и фрагментов.

Абстрактный класс ViewModel – позволяет создавать компоненты viewmodel архитектуры MVVM, который также позволяет оптимально работать в контексте жизненного цикла и фрагментов мобильного приложения.

Диаграмма классов подсистемы приложения, предназначенной для создания учетной записи и входа в систему пользователей представлена на рисунке 48.

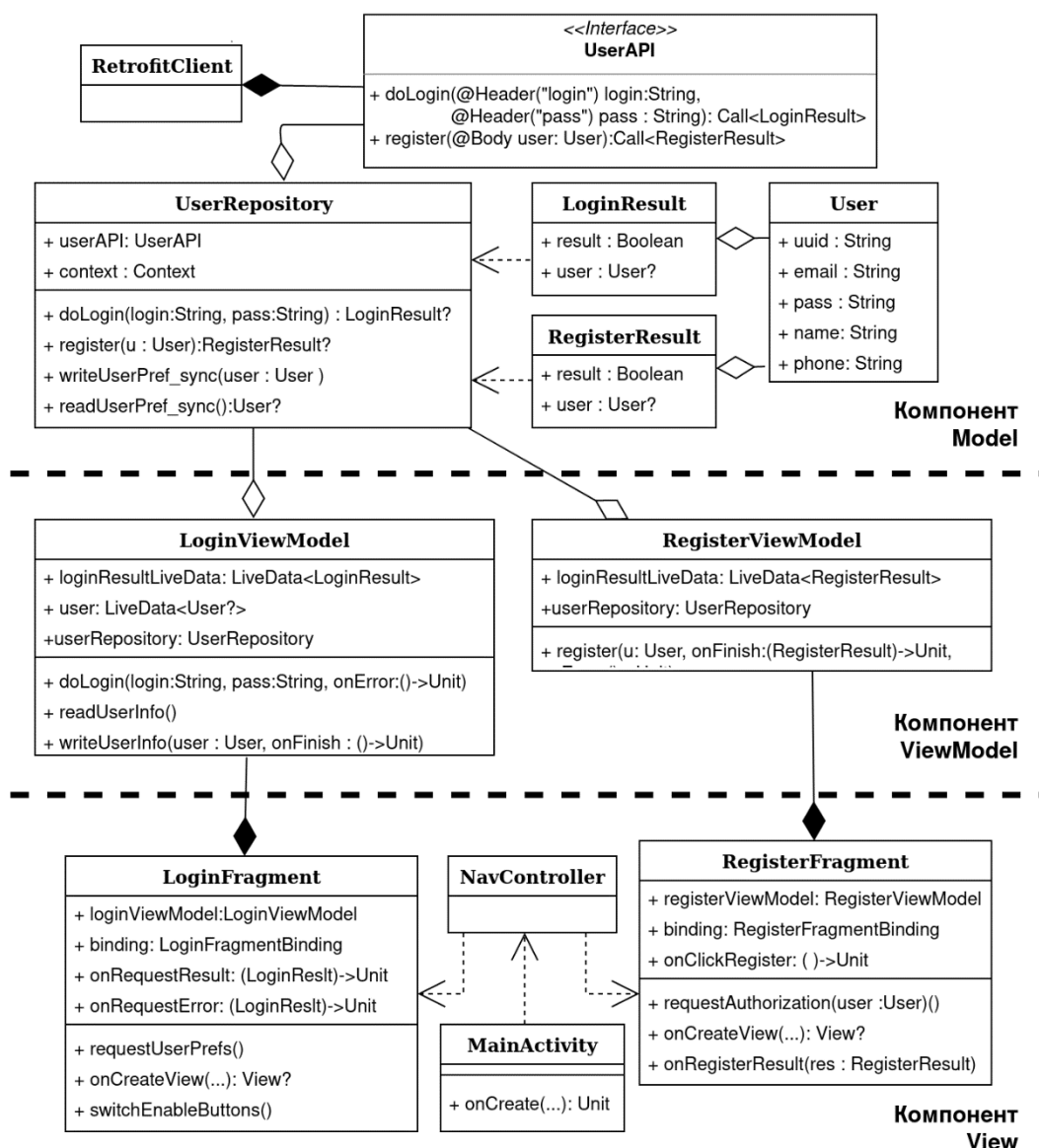


Рисунок 48 – Диаграмма классов компонента входа в систему и создания новой учетной записи

На диаграмме также показана пунктирными линиями декомпозиция классов на компоненты архитектуры MVVM. Таким образом,

спроектированная архитектура классов позволяет создать гибкую и расширяемую объектно-ориентированную структуру.

2.8 Разработка модифицированного алгоритма аутентификации

2.8.1 Проектирование алгоритма аутентификации

Специфика предметной области электронной коммерции предполагает обработку в информационных системах значительного объёма конфиденциальной информации. Утечка, модификация или несанкционированный доступ к информации такого типа, как правило, приводит к материальным или репарационным убыткам. Поэтому для программных средств электронной коммерции важнейшим требованием является обеспечение достаточного уровня информационной безопасности.

Основными потенциальными угрозами информационной безопасности систем электронной коммерции являются [56]:

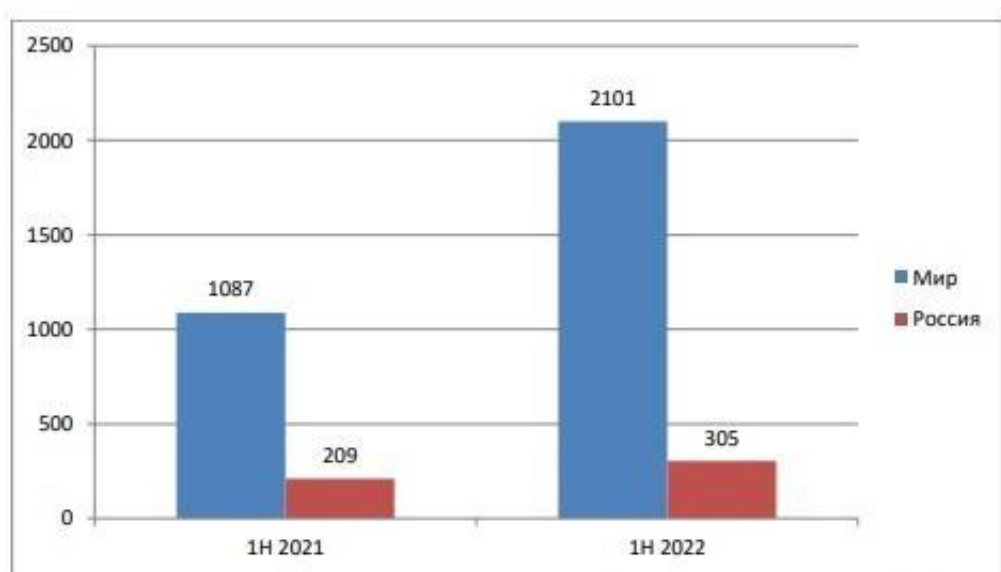
- DDoS-атаки, или же перегрузка веб-сервисов с целью генерации ошибок «отказ в обслуживании»;
- атака через посредника для перехвата интернет трафика между клиентом и сервером;
- использование скриптов для перебора паролей и несанкционированной аутентификации;
- использование архитектурных особенностей информационных структур для внедрение вредоносного ПО;
- использование уязвимостей в алгоритмах аутентификации пользователей.

Для противодействия угрозам существуют различные методы обеспечения информационной безопасности:

- аутентификация в виде алгоритма проверки подлинности пользователя;
- регламентирование доступа к системе в виде системы разделённого доступа, как уровня ОС, базы данных или уровня серверного приложения;

- шифрование данных передаваемых и хранимых данных;
- безопасное соединение;
- протоколы IPsec.

Осенью 2022 компания InfoWatch выпустила отчёт об исследовании утечек данных в крупных коммерческих информационных системах [57]. Исследование охватывало, как ситуацию в России, так и во всём мире. Как можно видеть на графике, представленном на рисунке 49, количество таких утечек в первом квартале 2022 г. значительно превышает количество аналогичных ситуаций в первом квартале 2021 г.



Число утечек: Мир – Россия, I полугодие 2021 г. – I полугодие 2022 г.

Рисунок 49 – График статистики утечек информации

Таким образом, недостаточный уровень применения методов информационной безопасности позволяет злоумышленникам неправомерно получать всё большее количество информации.

Использование криптографии позволяет защититься от подобных утечек и сопутствующего ущерба.

Криптография – это наука о методах обеспечения конфиденциальности и подлинности информации. Основными понятиями криптографии являются: ключ, шифрование, дешифрование. Шифрование представляет собой преобразование открытой информации в зашифрованный вид посредством

алгоритма и ключа. Ключ – параметр алгоритма шифрования. Дешифрование – процесс, позволяющий получить открытую информацию из закрытой.

Существуют различные криптографические алгоритмы, наиболее лаконичная их классификация представлена на рисунке 50.

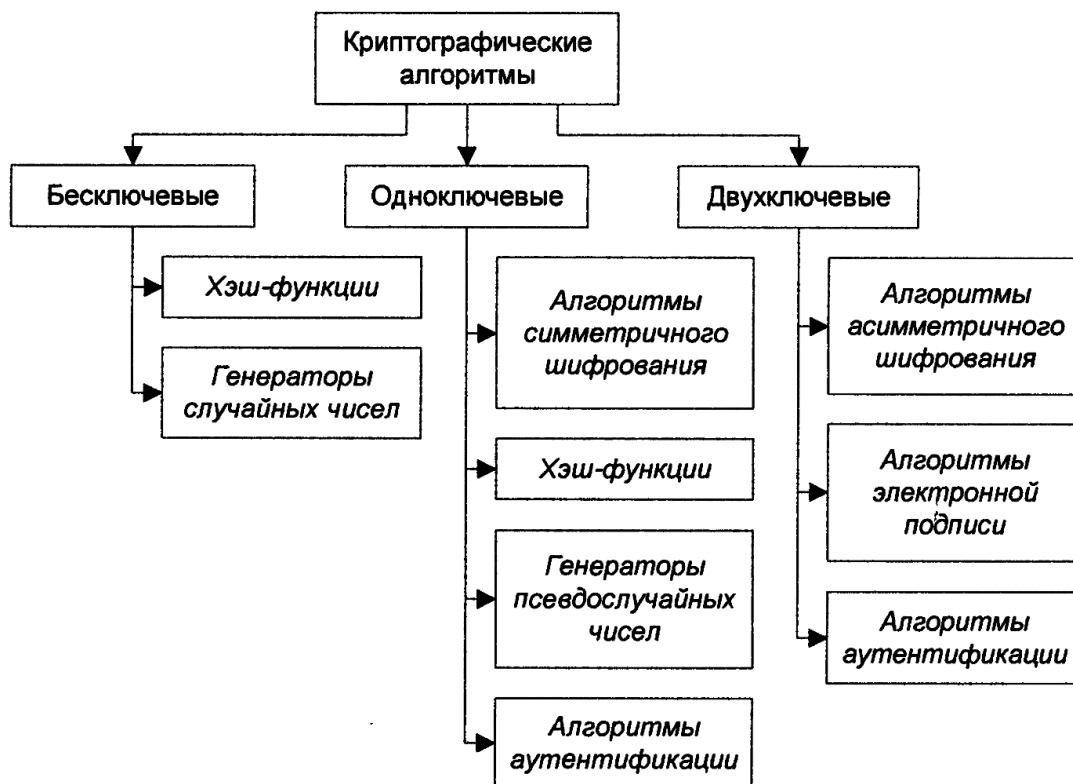


Рисунок 50 – Виды криптографических алгоритмов

Алгоритмы хеширования или же, хеш-функции, предназначены для преобразования входных данных в цепочку лексем таким образом, что при одних и тех же входных данных получаются одинаковые цепочки.

Алгоритмы шифрования разделяются на одноключевые (симметричные) с одним ключом для шифрования и дешифрования и двухключевые (асимметричные) алгоритмы, в которых шифрование производится публичным ключом, а дешифрование секретным ключом на стороне сервера.

Алгоритмы аутентификация обычно используют криптографическую подпись для проверки того, что обращающийся к ресурсу клиент тот, за кого себя выдает.

Рассмотрим программное обеспечение аналогичного назначения. В таблице 18 представлены программы аналогичного назначения и их отличия от разрабатываемой.

Таблица 18 – Сравнение аналогичных программ

Программа	Использование JWT	Алгоритм шифрования	Количество ключей	Длина ключа	Совмещение сигнатуры JWT подписи с шифром.
Разрабатываемая	Да	Модифицированный RSA	5	4096	Да
OAuth 2.0	Да	Стандартный RSA	4	2048	Нет
Javaх.crypto	Нет	Стандартный RSA	4	2048	Нет

Разрабатываемая программа позволяет использовать JWT подпись вместе с асимметричным алгоритмом шифрования. Таким образом, при использовании программы в системе электронной коммерции необходимо поддерживать 5 ключей: две пары для клиента и сервера, а также один секретный ключ JWT подписи.

Для программных средств электронной коммерции в виде трехзвенной архитектуры с клиентом, сервером и базой данных возможны следующие пути повышения надежности с использованием криптографии [58, 59]:

- модифицирование алгоритма аутентификации;
- применение модифицированного алгоритма шифрования при передаче данных между различными компонентами архитектуры программной части системы;
- повышение надежности хранения данных в базе данных посредством добавления улучшенного алгоритма хеширования.

Наиболее важным из перечисленных улучшений является фактор повышения надежности алгоритма аутентификации, так как уязвимость данного фрагмента может поставить всю систему под опасность неправомерного доступа. Роль аутентификации заключается в том, что

пользователь для доступа к системе должен предъявить фактор аутентификации, т.е. определенный вид уникальной информации.

Так как программные средства системы электронной коммерции созданы с использованием технологий Java, самым оптимальным способом создания алгоритма повышения надежности является построение аутентификации по стандарту JWT.

JSON Web Token (JWT) – это способ построения уникального ключа аутентификации в соответствии со стандартом RFC 7519. Данные пользователя, пересылаемые с использованием JWT-структуры подписываются. Таким образом, получатель может убедиться в подлинности сообщения. Однако, существенным недостатком JWT является то, что это алгоритм аутентификации в чистом виде. Это означает, что JWT не выполняет шифрование передаваемой информации.

JWT сообщение состоит из нескольких частей: заголовок, полезная нагрузка, подпись сообщения.

На рисунке 51 продемонстрирован алгоритм аутентификации с использованием JWT-сообщений.

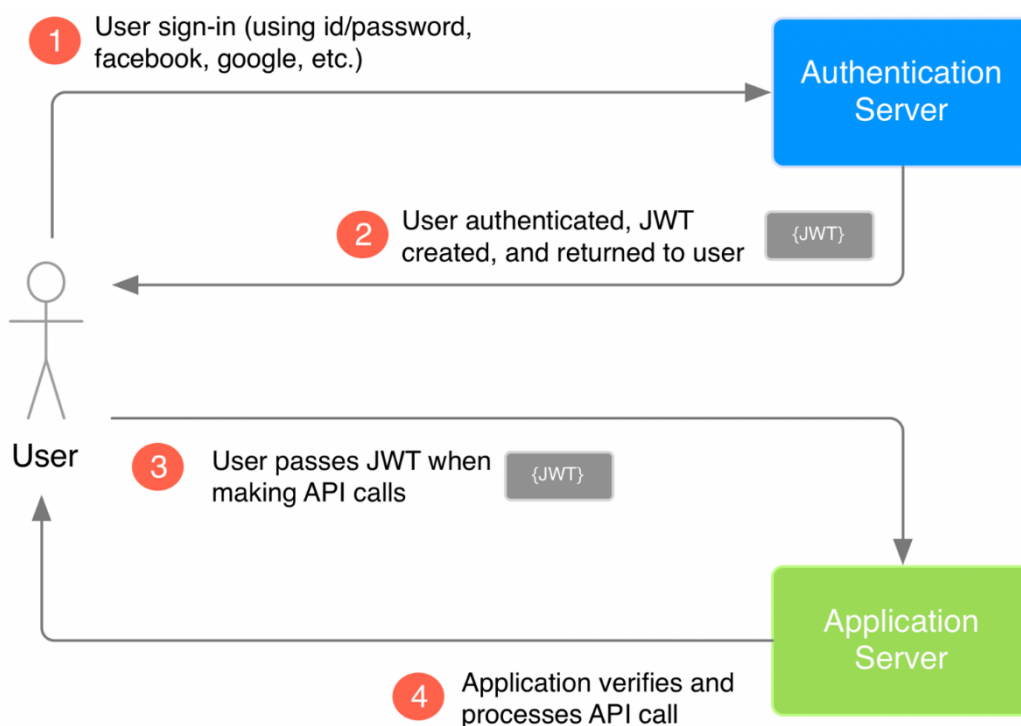


Рисунок 51 – Алгоритм аутентификации по стандарту JWT

Процесс аутентификации пользователя средствами JWT происходит следующим образом [60, 61]:

- клиент обращается на сервер с помощью публичного аутентификационного ключа;
- сервер создаёт JWT-токен и отправляет его в клиентское приложение;
- при последующих запросах клиент добавляет полученный токен, формируя таким образом JWT сообщение;
- при каждом запросе, сервер определяет, является ли пользователь тем, за кого себя выдает.

На последнем шаге серверное приложение использует криптографический алгоритм, который позволяет проверить, является ли входящее JWT сообщение именно тем, что было создано сервером для конкретного пользователя. Этот процесс называется JWT верификацией.

Пример использования JWT представлен в коде:

```
header = { "alg": "HS256", "typ": "JWT" }
payload = { "userId": "b08f86af-35da-48f2-8fab-cef3904660bd" }

const SECRET_KEY = 'cAtwalkkEy'
const unsignedToken = base64urlEncode(header) + '.' +
base64urlEncode(payload)
const signature = HMAC-SHA256(unsignedToken, SECRET_KEY)

const token = encodeBase64Url(header) + '.' +
encodeBase64Url(payload) + '.' + encodeBase64Url(signature)
```

Переменные header и payload содержат заголовок и полезную нагрузку JWT-сообщения. Далее, используя ключ SECRET_KEY, происходит шифрование алгоритмом хеширования SHA256 и формирование JWT-сообщения для передачи через API.

Особенность технологии JWT состоит в том, что она применяется не для сокрытия или маскировки данных, а только для подписи сообщения с целью аутентификации источника данных. Аутентификационный публичный ключ в

виде JWT-токена приходит в клиентское приложение в открытом виде и легко может быть перехвачен злоумышленником.

Исходя из этого, актуальная модификация алгоритма аутентификации заключается в добавлении дополнительных криптографических методов защиты информации в виде асимметричного алгоритма шифрования. В качестве данного алгоритма была предложена модификация популярного алгоритма RSA, которая позволяет повысить уровень криптостойкости шифра.

Таким образом, особенность данного алгоритма заключается в наличии пяти ключей, а именно: двух пар открытых и закрытых ключей шифрования и одного публичного ключа аутентификации, а также комбинированным использованием JWT аутентификации и модифицированного алгоритма асимметричного шифрования RSA [62, 63].

Диаграммы деятельности модифицированного алгоритма аутентификации показаны на рисунке 52 и рисунке 53.

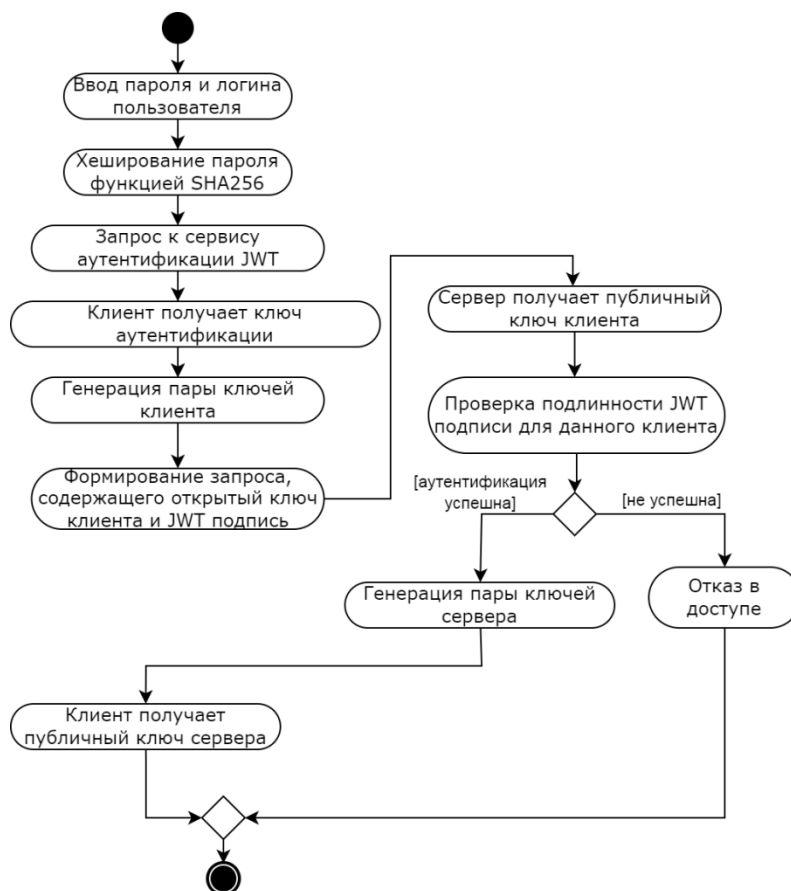


Рисунок 52 – Диаграмма деятельности на этапе обмена ключами

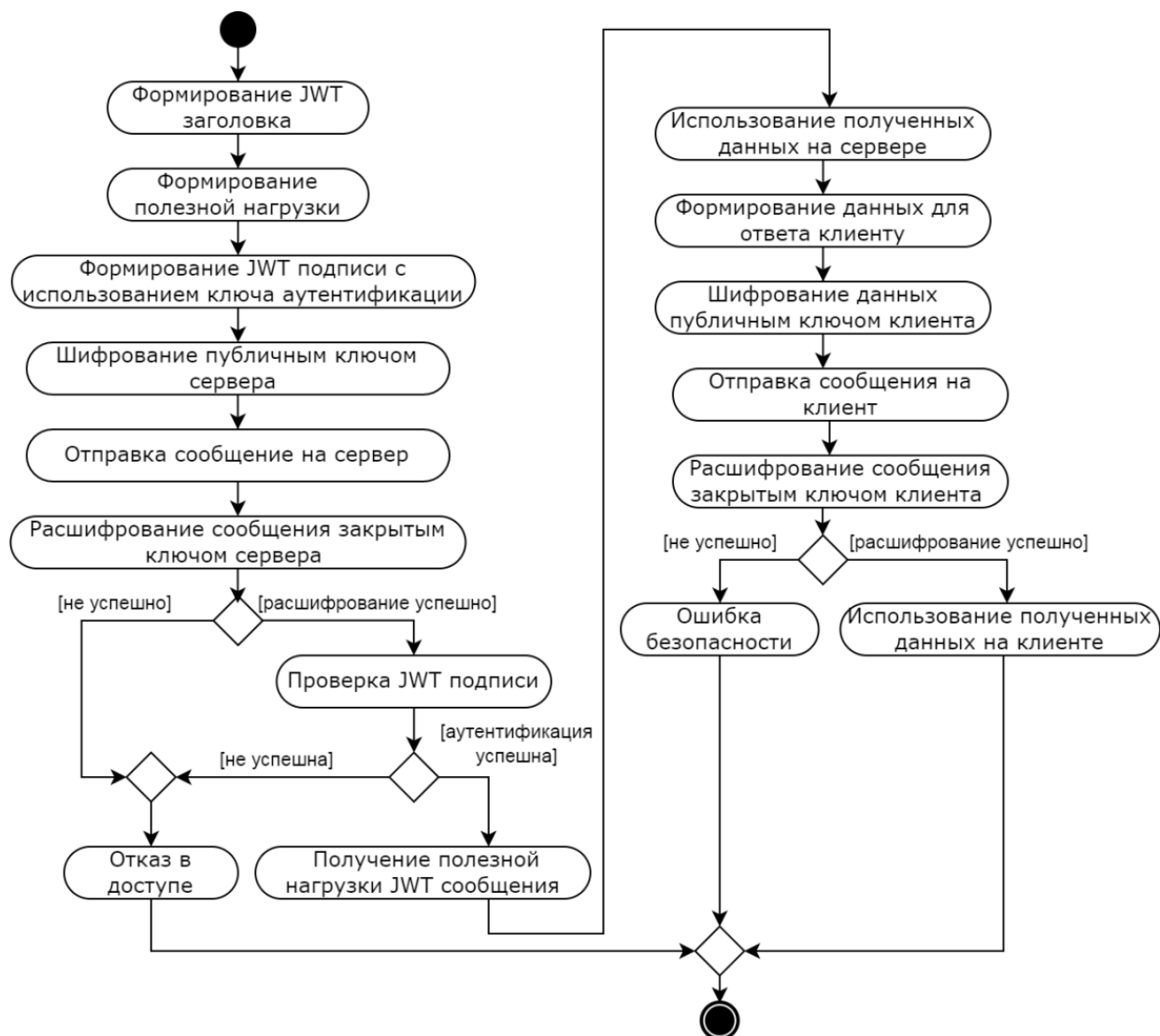


Рисунок 53 – Диаграмма деятельности модифицированного алгоритма аутентификации на этапе использования ключей

Таким образом, сообщения, которые пересылаются между компонентами системы, не только шифруются модифицированным асимметричным алгоритмом шифрования, но и имеют JWT-структуру.

Данная особенность позволяет повысить криптографическую стойкость процесса аутентификации. Исходя из этого, разработанный алгоритм аутентификации для системы электронной коммерции является обоснованным и позволяет сделать вывод о преимуществе системы электронной коммерции в безопасности аутентификации.

Кроме того, в данной модификации алгоритма используется ключ длиной в 4096 бит, что в два раза больше общепринятой длины в 2048 бит. Исходя из зависимости длины ключа к рейтингу криптостойкости (рисунок 54) можно сделать вывод о том, что данное изменение дополнительно повышает стойкость криптографического алгоритма [64].

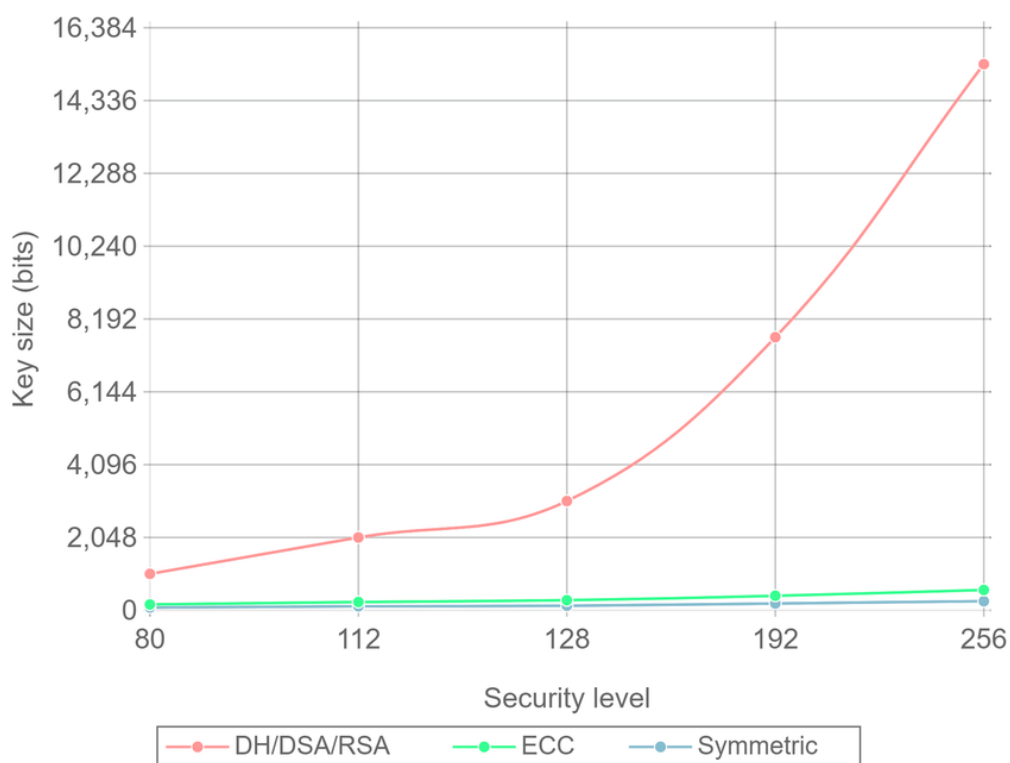


Рисунок 54 – График зависимости рейтинга криптостойкости от длины ключа RSA-алгоритма

Все полученные модификация позволяют предположить повышение криптографической стойкости получаемого шифра.

2.8.2 Разработка улучшенного алгоритма шифрования

Так как разрабатываемый алгоритм совмещает асимметричное шифрование с JWT-сообщением, то информацию о текущем пользователе можно передавать Headers запроса по стандарту HTTP (рисунок 55).

Http - request

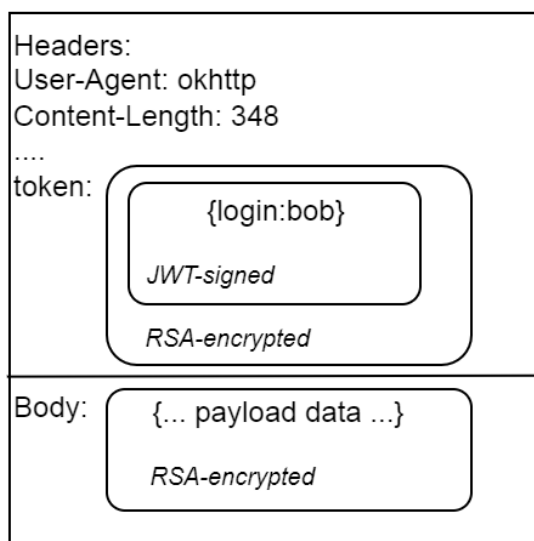


Рисунок 55 – Использование шифрования в различных частях HTTP запроса

RSA – это алгоритм шифрования с открытым ключом, что означает, что он использует два ключа: открытый ключ для шифрования сообщений и закрытый ключ для их расшифровки. Безопасность RSA основана на том факте, что большие целые числа трудно разложить на простые множители, что затрудняет получение закрытого ключа из открытого ключа [65 – 68].

Рассмотрим алгоритм генерации ключа. Первым шагом в использовании RSA является генерация пары открытого и закрытого ключей. Для этого нам нужно выбрать два больших простых числа, p и q , тогда

$$n = pq.$$

Далее необходимо вычислить значение функции Эйлера $\varphi(n)$. Полученное значение называется модулем.

Пусть функция $\text{НОД}(i,j)$ принимает значение наибольшего общего делителя чисел i и j . Найдём целое число e , удовлетворяющее условиям:

$$\begin{cases} 1 < e < \varphi(n) \\ \text{НОД}(e, \varphi(n)) = 1. \end{cases}$$

Найдём число d , обратное e по модулю $\varphi(n)$, то есть удовлетворяющее условию:

$$de \equiv 1 \pmod{\varphi(n)}.$$

Полученные пары (e,n) , (d,n) являются открытым и закрытым ключами RS алгоритма соответственно.

Рассмотрим процесс шифрования. Чтобы зашифровать сообщение m , мы сначала преобразуем его в целое число, используя схему обратимого кодирования, такую как ASCII или Unicode [9]. Шифрование производится по формуле:

$$c = E(m) = m^e \bmod n,$$

где c – полученный шифр.

Дешифрование производится по формуле:

$$c = D(m) = c^d \bmod n.$$

Данный стандартный алгоритм RSA может быть модифицирован несколькими способами в зависимости от конкретных потребностей или требований. Например, возможно использование криптографических дополнений, добавляющих ничего не значащие данные в процесс шифрования, использование ключей разных размеров, Использование разных законов распределения для генерации чисел p и q .

Однако важнейшей частью асимметричного алгоритма шифрования RSA является функция вычисления модуля, которая дополнительно рандомизирует псевдослучайные значения, генерируемые компьютером. Функция модуля обычно использует случайный характер расположения простых и составных чисел на числовой прямой. В стандартной реализации RSA для этого используется функция Эйлера, имеющая вид [69]:

$$\varphi(n) = \begin{cases} n - 1, & \text{если } n - \text{простое,} \\ \prod_{p|n}^n \left(1 - \frac{1}{p}\right), & \text{если } n - \text{любое составное,} \end{cases}$$

где n – случайное сгенерированное число, $n \in \mathbb{Z}$, $n > 0$;

p – простое число, итерирующее все разложения числа n на простые множители.

Для повышения разброса случайных чисел, формирующих модуль, целесообразна замена функции Эйлера на родственную функцию, дающую расходящуюся последовательность чисел. Такой цели может служить функция Кармайкла [69]:

$$\lambda(n) = \begin{cases} \frac{1}{2}\varphi(n), & \text{если } \exists k: n = 2^k, k \geq 3, \\ \varphi(n), & \text{в остальных случаях.} \end{cases}$$

Различие графиков функций Эйлера (слева) и функций Кармайкла (справа) представлено на рисунке 56.

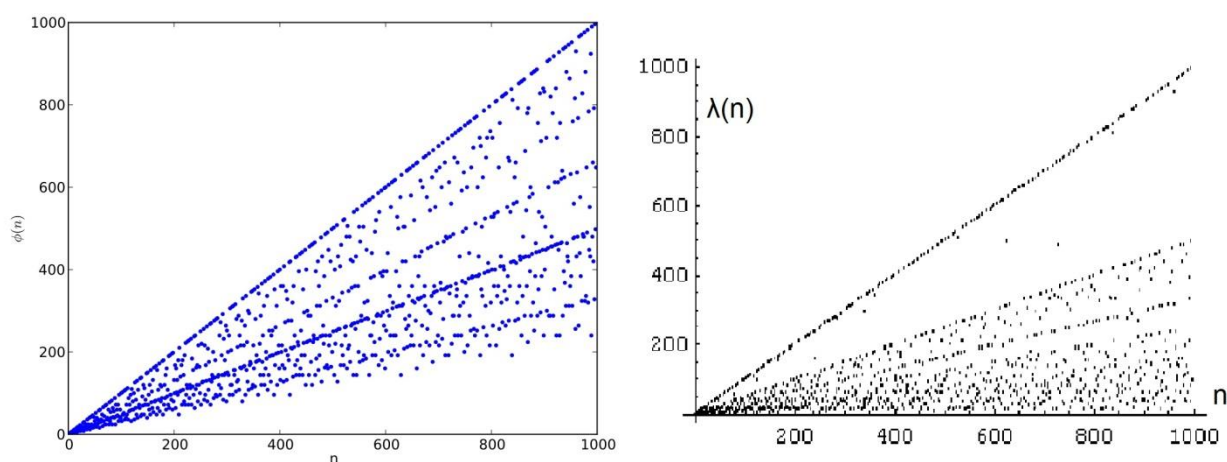


Рисунок 56 – График функции Эйлера и функций Кармайкла

Таким образом, замена функции вычисления модуля на функцию Крамайкла позволяет ожидать повышения разброса случайной величины, так как для неё на графике достигается больший разброс случайных чисел относительно прямой простых чисел – диагональная последовательность точек на графике. Получаемые значения модуля будут в сильнее отличаться для простых и непростых случайных чисел p и q .

Повышение разброса значения модуля для простых и составных значений аргумента позволяет ожидать повышения разброса значений для публичного и секретного ключей – пар значений (e, n) , (d, n) . Которые вычисляются с помощью операции взятия обратного по модулю. Данная модификация позволяет ожидать повышения криптостойкости получаемого шифра.

Выводы

На этапе анализ функциональных и нефункциональных требований были определены задачи на проектирование, реализацию и тестирование системы электронной коммерции. Функционал был разделён по категориям пользователей системы: покупатель, менеджер, администратор.

Создана онтологическая модель предметной области, которая отличается от известных расширенным набором сущностей и классов. Она используется для эффективного проектирования программных средств объекта исследования.

Созданы две дискретно-событийные имитационные модели для оптимизации и определения оптимального режима работы системы электронной торговли. В отличие от известных имитационных моделей, они носят комплексный характер и позволяют учитывать особенность взаимодействия аппаратно-программных средств с человеческим фактором. Данные имитационные модели были использованы для оптимизации режима работы системы электронной коммерции, а именно параметр количества необходимых работников. Получен результат – необходимо три менеджера и три работника службы доставки.

Была проведена оценка адекватности имитационных моделей. Адекватность моделей была подтверждена путем их валидации.

На этапе логического проектирования базы данных системы электронной коммерции, исходя из анализа предметной области и функциональных требований, была создана концептуальная модель базы данных. На основе концептуальной модели была создана логическая модель базы данных.

На этапе проектирования серверной части системы электронной коммерции, исходя из функциональных требований, было спроектировано серверное приложение системы электронной коммерции. Была выбрана архитектура RESTful сервисов. Так же для формализации процесса проектирования был применен UML.

Проведённое проектирование клиентского мобильного приложения системы электронной коммерции заключалось в выборе подхода к архитектуре

программных средств для эффективного функционирования их с API среды ОС Android. Был выбран архитектурный шаблон MVVM, с поддержкой встроенных компонентов Android. Для формализации процесса был использован UML.

Этап разработки модифицированного алгоритма аутентификации был разделен на два шага: проектирование алгоритма аутентификации и разработка модифицированного алгоритма шифрования для данного алгоритма аутентификации.

Спроектирован модифицированный алгоритм, который отличается от известных алгоритмов наличием пяти ключей, а именно: двух пар открытых и закрытых ключей шифрования и одного публичного ключа аутентификации. Алгоритм также отличается комбинированным использованием JWT аутентификации и модифицированного алгоритма асимметричного шифрования RSA, что позволяет повысить надежность процесса аутентификации в системе электронной коммерции.

Был разработан упомянутый выше модифицированный алгоритм асимметричного шифрования на основе RSA. Модификация алгоритма заключается в замене функции вычисления модуля с функции Эйлера в стандартной реализации на функцию Кармайкла. При сравнении графиков функций видно больший разброс значений функции Кармайкла для простых и составных значений аргумента. Данная модификация позволяет ожидать повышения криптостойкости получаемого шифра, а следовательно, повышения надежности алгоритма аутентификации.

3 Реализация системы электронной коммерции с модифицированным алгоритмом аутентификации

3.1 Реализация модифицированного алгоритма аутентификации

Программные средства алгоритма аутентификации были реализованы средствами языка Java и запакованы в Jar библиотеку, которая подключается к компонентам системы электронной коммерции.

Исходный код программных средств алгоритма аутентификации представлен в приложении Г.

3.2 Эксперимент по проверке преимущества модифицированного алгоритма аутентификации

Оценка криптографической стойкости алгоритма шифрования является сложной задачей, так как актуальные методы и инструменты взлома шифра, используемые хакерами, не публикуются. Однако, проверить надежность шифра RSA возможно. Вот несколько шагов, которые можно предпринять, чтобы проверить надежность шифра RSA:

Длина тестового ключа: надёжность RSA прямо пропорциональна длине используемого ключа. Как правило, ключи длиной не менее 2048 бит считаются безопасными для RSA.

Тестирование генерации ключей позволяет проверить, что алгоритм создает надежные случайные ключи. Это можно сделать, проанализировав распределение простых чисел, сгенерированных алгоритмом.

Тестирование шифрования/дешифрования позволяет проверить надежность алгоритмов шифрования и дешифрования, чтобы убедиться, что они не уязвимы для атак, таких как атаки по времени или атаки с выбранным открытым текстом.

В целом, чтобы проверить надежность шифра RSA, необходимо выполнить всесторонний анализ системы шифрования и ее компонентов, чтобы выявить любые потенциальные уязвимости и слабые места. Это можно сделать

с помощью комбинации ручного анализа, автоматизированного тестирования и тестирования на проникновение.

Из всех возможных вариантов наименее затратным по времени и ресурсам является математический или статический анализ случайных чисел, лежащих в основе процесса генерации криптографического шифра. Для стандартной и модифицированной реализации алгоритма RSA метод анализа закона распределения случайных чисел для пары изначальных случайных p и q не подходит, так как прежде чем сформировать пары ключей, данные числа используются в функции модуля. Далее именно функция модуля позволяет сформировать пары ключей (e, n) , (d, n) .

Таким образом, актуальной возможностью анализа созданных алгоритмом шифров является статический анализ значений функции модуля, генерирующихся каждый раз при создании пар ключей алгоритмом RSA. При проведении анализа двух реализаций алгоритма необходимо определить, в какой реализации значения функции модуля более независимы от генератора случайных чисел и независимы между собой. Данное утверждение базируется на том, что функции генерации подвержены уязвимости обратного анализа закона распределения случайной величины, а генераторы случайной величины уязвимы к анализу их зависимости от текущего момента времени (timestamp) в миллисекундах и зависимости от реализации датчика псевдослучайных чисел [70].

Рассмотрим инструментарий, который может нам помочь в данном эксперименте.

Чтобы проанализировать, какой набор случайных чисел является более случайным, можно использовать различные статистические эксперименты для измерения случайности или независимости чисел в каждом наборе. Вот некоторые из них:

Критерий хи-квадрат (Chi-squared test) – это статистический тест, который измеряет, насколько хорошо наблюдаемые данные соответствуют ожидаемым данным. Для случайных чисел можно рассчитать ожидаемую

частоту каждого значения и сравнить их с наблюдаемыми частотами с помощью данного критерия хи-квадрат. Если наблюдаемая и ожидаемая частоты близки, то набор чисел считается более случайным.

Тест запусков – это статистический тест, который измеряет появление шаблонов или одинаковых серий в последовательности чисел. Этот тест позволяет увидеть, имеет ли один набор случайных чисел больше шаблонов или серий, чем другой, что указывало бы на то, что он менее случайный.

Критерий согласия Колмогорова — это статистическая величина, которая измеряет соответствие между двумя распределениями или набором величин и заданным законом распределения. Она может быть использована, чтобы сравнить распределение чисел в каждом наборе и посмотреть, какое из них является более случайным.

Используем критерий согласия Колмогорова, чтобы исследовать значения функции модуля при генерации ключей в стандартной (с функцией Эйлера) и в модифицированной (с функцией Кармайка) реализации RSA-алгоритма. При этом в качестве распределения выбран нормальный закон распределения.

Таким образом, основное направление эксперимента можно сформулировать следующим образом: чем в большей степени выборка из значений функции модуля RSA алгоритма принадлежит нормальному закону распределения, тем более случайны эти значения и тем более стойкий шифр генерируется алгоритмом.

Пусть X_n – выборка значений модуля RSA алгоритма, $F_n(x)$ – неизвестная эмпирическая функция распределения модуля RSA, $F(x)$ – искомая функция распределения, в нашем случае это функция нормального распределения случайной величины. Тогда, критерий Колмогорова в общем виде [71]:

$$D_n = \sup_x |F_n(x) - F(x)|.$$

Сформируем и запишем в файл выборки по 100, 1000, 10000, 50000 значений функции модуля RSA алгоритма выводимых при генерации пары ключей. Проведём данную процедуру для стандартной реализации алгоритма RSA и для модифицированного алгоритма.

Создадим Python скрипт для вычисления критерия Колмогорова, используя библиотеку `scipy.stats`.

```
import numpy as np
from scipy.stats import kstest

with open('viborka12', 'r') as f:
    set = [float(line.strip()) for line in f]

    #Колмогоров с нормальным законом распределения
    test_statistic, p_value = kstest(set, 'norm')

    print('Test statistic:', test_statistic)
    print('p-value:', p_value)
```

Проведём эксперимент для всех выборок, результат представлен в таблице 19.

Таблица 19 – Критерий Колмогорова для значений функции модуля

Размер выборки	Стандартная реализация RSA	Модифицированный алгоритм шифрования RSA с применением функции Кармайкла
1500	0.01131651545	0.0282330365
5000	0.03362707476	0.05631364065
10000	0.14933495688	0.2381435760
50000	0.3329395680	0.41313043487

Таким образом, значения функций модуля модифицированного алгоритма шифрования показали большее соответствие нормальному закону распределения, что позволяет судить о выигрыше в криптостойкости генерируемого шифра в сравнении со стандартной реализацией. Кроме, того, модифицированный алгоритм шифрования использует ключ длиной 4096 в отличие от стандартного 2048, что так же позволяет сделать вывод о большей устойчивости к действиям по взлому защищённых данных.

Исходя из всего перечисленного, разработанная программа аутентификации для системы электронной коммерции с использованием модифицированного алгоритма аутентификации и модифицированного алгоритма шифрования, является обоснованной и позволяет говорить о

преимущество системы электронной коммерции в безопасности аутентификации.

3.3 Реализация и развертывание базы данных

Для работы с базой данных через СУБД PostgreSQL использована программа DBeaver, содержащая удобный редактор SQL скриптов, редактор таблиц баз данных с возможностью генерации диаграммы данных.

Используя созданную в среде CA ERwin Data Modeler физическую модель, был создан скрипт создания базы данных на языке PL/pgSQL, который представлен в листинге Д.1 в приложении Д.

На рисунке 57 представлена диаграмма данных развернутой базы данных, построенная в среде DBeaver.

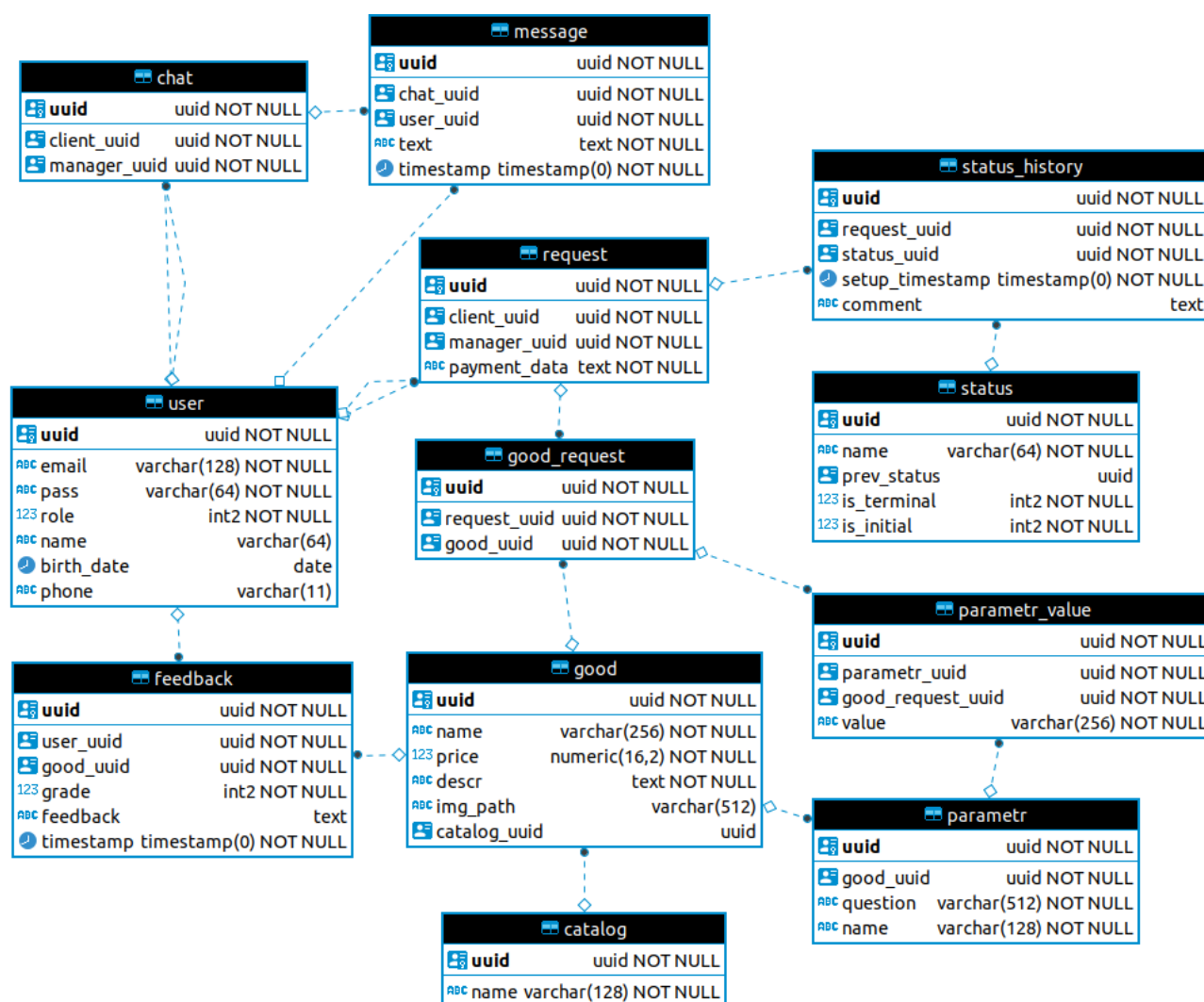


Рисунок 57 – Диаграмма данных развернутой базы данных

Построенная в среде DBeaver диаграмма данных отражает состояние базы данных после выполнения скрипта генерации данной базы данных и также отражает результат создания базы данных.

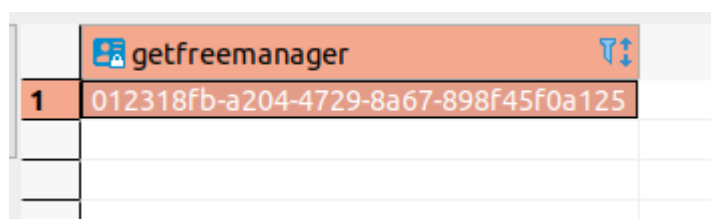
Диаграмма отражает таблицы и атрибуты с указанием использованных типов данных.

Кроме того, определенная логика была реализована на стороне базы данных. Была создана хранимая функция `getfreemanager`, которая возвращает UUID наименее загруженного менеджера. Код функции представлен в приложении Д.2. Данная функция принимает 2 параметра: код роли менеджера и UUID финального состояния заявок.

Рассмотрим работу функции `getfreemanager`. Для вызова функции необходимо выполнить данный запрос к базе данных:

```
select * from getfreemanager(2, 'ad631ed1-a650-4f9c-bfa1-70b84c6f0d10')
```

Результат запроса показан на рисунке 58.



	getfreemanager
1	012318fb-a204-4729-8a67-898f45f0a125

Рисунок 58 – Результат запроса к функции `getfreemanager`

Полученная база данных была развернута под управлением операционной системы Linux Ubuntu. Данный выбор был обусловлен тем, что большинство дистрибутивов операционной системы Linux включает пакет СУБД PostgreSQL по умолчанию и не требуют установки дополнительных программ.

3.4 Реализация серверной части системы электронной коммерции

Исходя из результатов проектирования программного обеспечения и выбора технологий реализации, была реализована серверная часть системы электронной коммерции на языке Java. Исходный код представлен в приложении Е.

Диаграмма основных классов серверной части системы электронной коммерции представлена на рисунке 59.

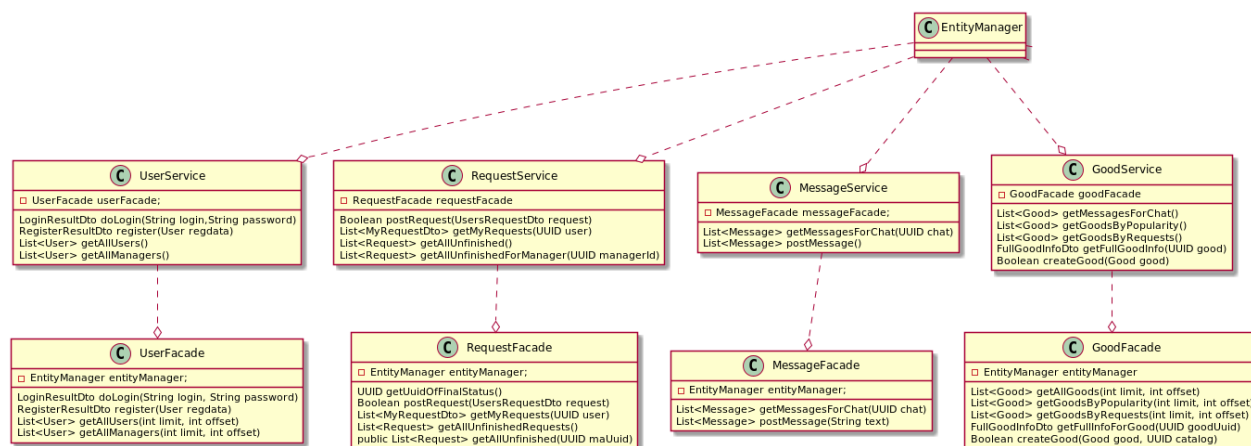


Рисунок 59 – Диаграмма классов серверной части системы электронной коммерции

На диаграмме можно видеть, что все классы разделены на два слоя: фасады и сервисы. Из них фасады UserFacade, RequestFacade, MessageFacade, GoodFacade предназначены для предоставления API веб-сервисов через протокол HTTP.

Сервисы предоставляют функции работы с сущностями объектно-реляционного представления JPA и базы данных. Серверная часть системы построена на основе архитектуры RESTful сервисов, данная особенность показана на диаграмме компонентов на рисунке 60.

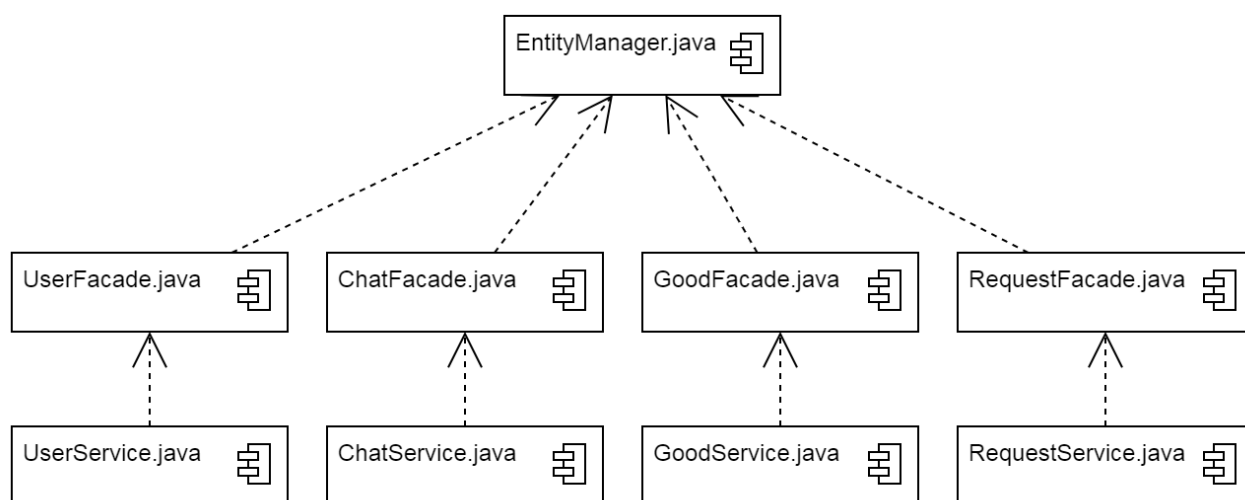


Рисунок 60 – Диаграмма компонентов RESTful сервисов

Таким образом, RESTful архитектура позволяет удобно реализовать клиент-серверное взаимодействие в разрабатываемой системе.

3.5 Реализация клиентской части системы электронной коммерции

Исходя из поставленных задач и функциональных требований, было реализовано клиентское мобильное приложение системы электронной коммерции на языке Kotlin. Исходный код мобильного приложения представлен в приложении Ж.

Диаграмма основных классов мобильного приложения представлена на рисунке 61.

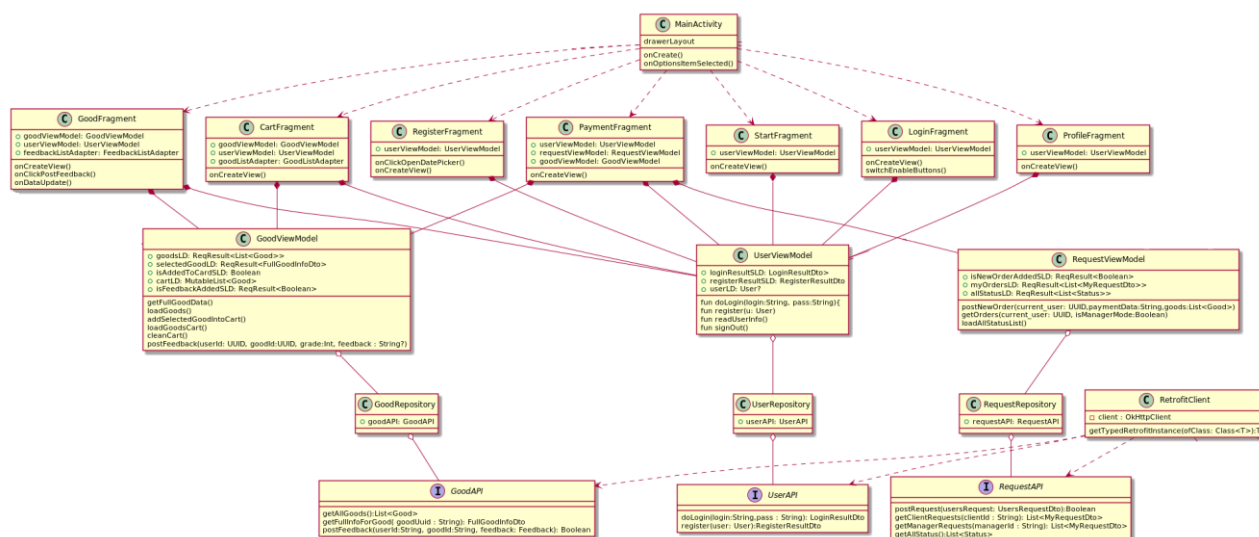


Рисунок 61 – Диаграмма классов клиентского мобильного приложения

Описание классов представлено в таблице 20.

Таблица 20 – Описание классов приложения

Название	Описание
MainActivity	Главная точка входа в приложение. Содержит инициализация ресурсов и классов навигации по фрагментам приложения.
GoodFragment	Фрагмент страницы товара.
CartFragment	Фрагмент страницы корзины покупок.
RegisterFragment	Фрагмент страницы создания учетной записи в приложении.
PaymentFragment	Фрагмент страницы оформления товара при заявке.
StartFragment	Фрагмент страницы запуска приложения

Продолжение таблицы 20

Название	Описание
LoginFragment	Фрагмент страницы входа в систему.
ProfileFragment	Фрагмент страницы профиля учетной записи пользователя.
ManagerMenuFragment	Фрагмент страницы менеджера с функционалом выполнения заявок клиентов.
MyOrdersFragment	Фрагмент страницы, отображающей активные заявки текущего пользователя.
GoodViewModel	ViewModel компонент содержащий актуальные данные для работы функционала, связанного с товарами.
UserViewModel	ViewModel компонент содержащий актуальные данные для работы функционала, связанного с аккаунтами пользователей.
RequestViewModel	ViewModel компонент содержащий актуальные данные для работы функционала, связанного с заявками на товар.
ChatViewModel	ViewModel компонент содержащий актуальные данные для работы функционала, связанного с чатами.
GoodRepository	Репозитории, предоставляющие доступ к ресурсам, включая локальное хранилище и API запросов к веб-сервисам.
UserRepository	
RequestRepository	
ChatRepository	
GoodAPI	Интерфейс подключения к веб-сервисам товаров.
UserAPI	Интерфейс подключения к веб-сервисам, предоставляющим функционал работы с аккаунтами пользователей.
RequestAPI	Интерфейс подключения к веб-сервисам заявок.
ChatAPI	Интерфейс подключения к веб-сервисам, предоставляющим функционал работы с аккаунтами пользователей.
RetrofitClient	Класс, использующий интерфейсы доступа к веб-сервисам для создания подключения.

Отдельно стоит выделить реализацию такого компонента архитектуры MVVM, как модель. Диаграмма классов слоя модели, использованного в мобильном приложении, представлена на рисунке 62.

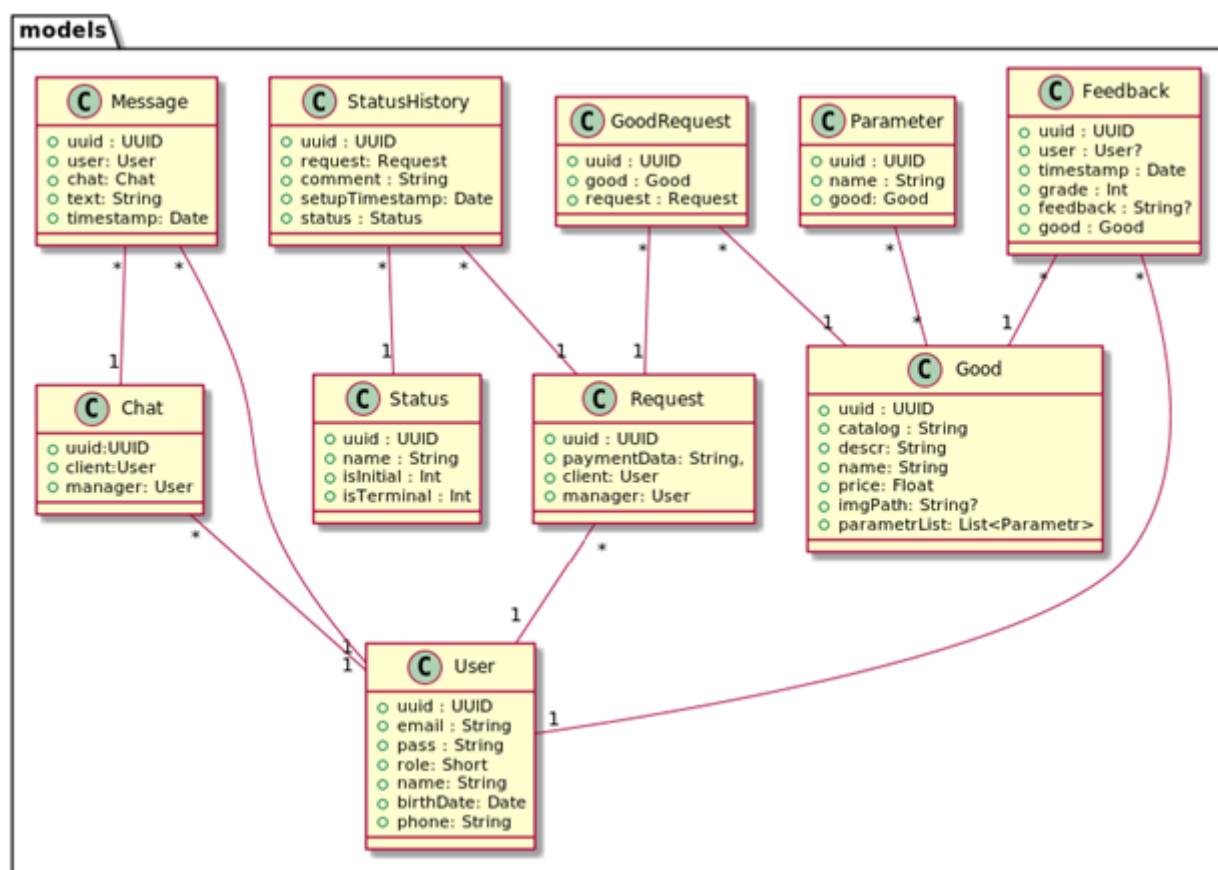


Рисунок 62 – Диаграмма классов слоя модели приложения

Слой модели представляет собой объектно-ориентированное отражение сущностей предметной области. Кроме того, классы в составе слоя модели могут содержать бизнес логику предметной области, методы конвертации одних сущностей в другие, клонирования и т. д.

Как можно видеть на полученной диаграмме классов, используемая в мобильном приложении модель по структуре сущностей идентична концептуальной модели базы данных, построенной на этапе проектирования приложения. Данная особенность обусловлена тем, что приложение полностью реализует интерфейс, предоставляемый веб-сервисами серверной части системы электронной коммерции.

Диаграмма развёртывания всей системы электронной коммерции, как системы отдельных компонентов показана на рисунке 63.

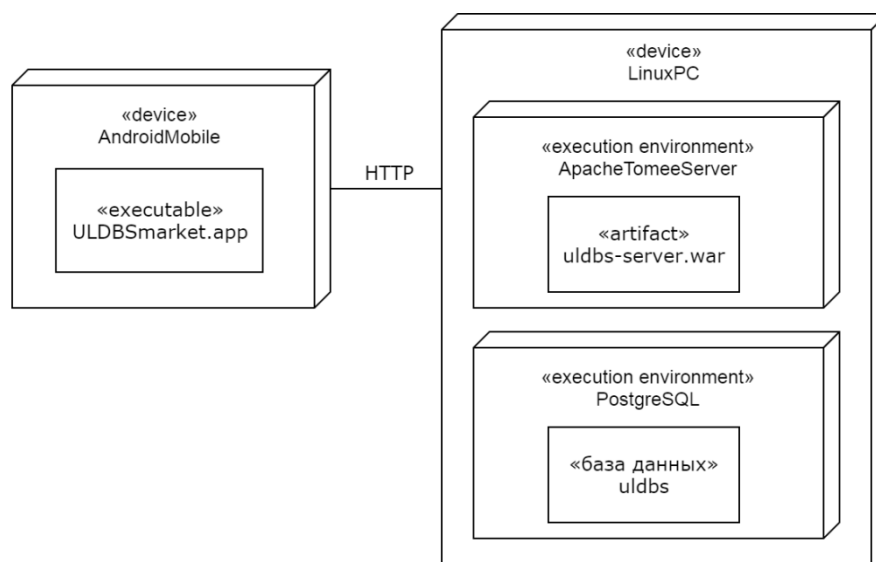


Рисунок 63 – Диаграмма развертывания системы электронной коммерции

Таким образом, на физическом устройстве сервера под управлением ОС Linux происходит развертывание сразу двух компонентов системы: серверной части системы и базы данных. Клиентское мобильное приложение связывается с серверной частью через HTTP.

3.6 Тестирование системы электронной коммерции

3.6.1 Нагрузочное тестирование

Для выполнения нагрузочного тестирования серверной части системы было принято решение использовать инструмент Jmeter [22]. На рисунке 64 представлен составленный план тестирования.

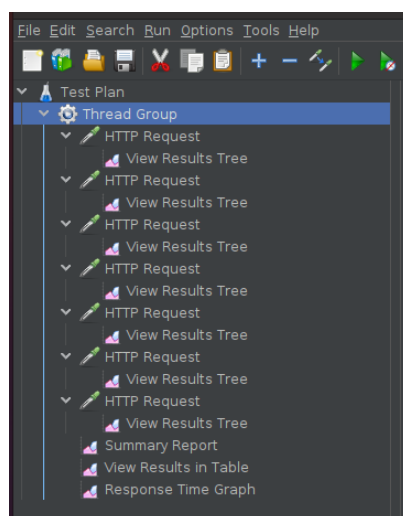


Рисунок 64 – План нагрузочного тестирования приложения

Конфигурация нагрузочного тестирования представлена на рисунке 65. Каждый сервис будет протестирован одновременно 50 пользователями 50 раз.

Рисунок 65 – Конфигурация нагрузочного тестирования приложения

На рисунке 66 представлен результат нагрузочного тестирования.

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes
1	05:08:31.373	Thread Group 1-1	HTTP Request	7	✓	241
2	05:08:31.381	Thread Group 1-1	HTTP Request	6	✓	2070
3	05:08:31.387	Thread Group 1-1	HTTP Request	4	✓	2090
4	05:08:31.391	Thread Group 1-1	HTTP Request	5	✓	666
5	05:08:31.394	Thread Group 1-2	HTTP Request	3	✓	241
6	05:08:31.396	Thread Group 1-1	HTTP Request	3	✓	241
7	05:08:31.399	Thread Group 1-2	HTTP Request	6	✓	2070
8	05:08:31.399	Thread Group 1-1	HTTP Request	6	✓	2070
9	05:08:31.407	Thread Group 1-2	HTTP Request	4	✓	2090
10	05:08:31.413	Thread Group 1-3	HTTP Request	3	✓	241
11	05:08:31.411	Thread Group 1-2	HTTP Request	8	✓	666
12	05:08:31.416	Thread Group 1-3	HTTP Request	4	✓	2070
13	05:08:31.419	Thread Group 1-2	HTTP Request	3	✓	241
14	05:08:31.420	Thread Group 1-3	HTTP Request	8	✓	2090
15	05:08:31.407	Thread Group 1-1	HTTP Request	22	✓	2090
16	05:08:31.429	Thread Group 1-1	HTTP Request	8	✓	666
17	05:08:31.428	Thread Group 1-3	HTTP Request	10	✓	666
18	05:08:31.433	Thread Group 1-4	HTTP Request	11	✓	241
19	05:08:31.438	Thread Group 1-3	HTTP Request	6	✓	241
20	05:08:31.422	Thread Group 1-2	HTTP Request	23	✓	2070
21	05:08:31.453	Thread Group 1-5	HTTP Request	3	✓	241
22	05:08:31.446	Thread Group 1-2	HTTP Request	10	✓	2090
23	05:08:31.456	Thread Group 1-2	HTTP Request	8	✓	666
24	05:08:31.456	Thread Group 1-5	HTTP Request	9	✓	2070
25	05:08:31.444	Thread Group 1-4	HTTP Request	21	✓	2070
26	05:08:31.444	Thread Group 1-3	HTTP Request	24	✓	2070

Рисунок 66 – Результат нагрузочного тестирования приложения

Проведенное нагрузочное тестирование показывает стабильную работу серверной части системы электронной коммерции под планируемой штатной нагрузкой в виде одновременным использованием 50 пользователями.

3.6.2 Функциональное тестирование

Основываясь на функциональных требованиях, определенных на этапе анализа и на основе бизнес-процессов предметной области был проведен набор тестов, представленных в таблице 21. Скриншоты большей части результатов тестирования представлены в приложении И.

Таблица 21 – Результаты функционального тестирования системы

Бизнес-процесс	Тест и параметры	Результат
Создать учетную запись	Ввести валидные данные учет. записи Логин test2@gmail.com Пароль test2	Тест пройден. Запись пользователя добавлена в систему.
	Ввести данные с отличающимися паролями. Логин test11@gmail.com Пароль test11 Повтор пароля testtest	Тест пройден. Выведено уведомление о несовпадении паролей (рисунок И.1).
	Ввести данные с логином существующего в системе пользователя. Логин test2@gmail.com	Тест пройден. Выведено уведомление о существовании записи.
	Попытаться создать логин не являющийся электронным почтовым адресом Логин andrey777	Тест пройден. Выведено уведомление о вводе не валидных данных
Войти в систему	Ввести валидные данные Логин test2@gmail.com Пароль test2	Тест пройден. Открыт доступ к странице профиля пользователя (рисунок И.2).
	Ввести не валидные данные логин test2@gmail.com пароль test1234	Тест пройден. Выведено уведомление.
Просматривать каталог товаров	Загрузит страницу каталога.	Тест пройден. Каталог отображен (рисунок И.3).
Просмотр подробной информации о товаре	Просматривать страницу товара с доступным фото товара.	Тест пройден. Фото товара динамически подгружено (рисунок И.5).
	Просматривать страницу товара без фото.	Тест пройден. Выведено изображение по умолчанию (рисунок И.6).
Просматривать страницу с картой	Загрузить карту с маркером расположения магазина	Тест пройден. Карта с маркером показана (рисунок И.7).
Оставить отзыв на товар	Оставить отзыв и оценку на товар от лица пользователя, заказывавшего данный товар	Тест пройден. Отзыв добавлен на страницу товара (рисунок И.8).
	Оставить только оценку на товар. Без текстового отзыва.	Тест пройден. На странице товара оценка.
	Попытаться оставить отзыв на товар от лица пользователя, не заказывавшего данный товар	Тест пройден. Отзыв не добавлен.

Продолжение таблицы 21

Бизнес-процесс	Тест и параметры	Результат
Оставлять заявку на товар	Попытка оформить заявку на товар Данные оплаты Карта 1111222233334444, Пароль 123, ФИО Alex Alexeev Alexeevich	Тест пройден. Заявка поступила в систему (рисунок И.9).
Использовать чат с менеджером	Выбирать чат из списка чатов при наличии нескольких активных чатов с менеджерами.	Тест пройден. Список чатов загружен (рисунок И.10).
	Загрузить список чатов пользователю без активных чатов и заявок.	Тест пройден. Вывод сообщения об отсутствии доступных чатов.
	Отправлять сообщение в чат	Тест пройден. Сообщение отображено в чате (рисунок И.11).
	Запрос чата с менеджером при отсутствии активных чатов. Чаты для данного пользователя отсутствуют, но у пользователя есть открытая заявка.	Тест пройден. На стороне сервера создан новый чат. Чат отображен в клиентском приложении
Отмечать выполнение заявок	Менеджер выставляет новый статус заявке	Тест пройден. Список заявок обновился с внесенными изменениями (рисунок И.12).
Просматривать статистику	Администратор заходит в меню администратора.	Тест пройден. Садистка загружена.
Управлять каталогом товаров	Администратор добавляет новый товар в каталог.	Тест пройден. В каталог добавлен товар.
	Администратор удаляет товар из каталога.	Тест пройден. Из каталога удален товар.

Результаты тестирования работоспособности показывают стабильную и корректную работу системы электронной коммерции на данном наборе тестов. Система выполняет все функциональные требования.

3.6.3 Модульное тестирование

В дополнение к функциональному тестированию, были созданы модульные тесты на языке Kotlin в среде разработки Android Studio. Пример модульного теста на языке Kotlin представлен в листинге ниже:

```
class UnitTest {
    //Тестовые данные 1
```

```

val test_chat_uuid = "3a7ac6a6-3587-4f70-9259-d2c5bd0655d4"
val test_managername = "Александр_test"
val test_client_uuid = "0aab3e7f-cde2-4b0d-ac93-141c987be013"

@Test fun testOpenChatFragment() {
    val bundle : Bundle = Bundle();
    bundle.putString("chatid",test_chat_uuid)
    bundle.putString("managername",test_managername)
    bundle.putString("userid", test_chat_uuid)
    activityRule.scenario.onActivity {
        it.findNavController(R.id.nav_host_fragment)
            .navigate(R.id.nav_chat_fragment, bundle)
    }
}

@Test fun testOpenGoodFragment() {
    val bundle : Bundle = Bundle();
    bundle.putString("goodid",good_id)
    activityRule.scenario.onActivity {
        it.findNavController(R.id.nav_host_fragment)
            .navigate(R.id.nav_good_fragment, bundle)
    }
}

@Test fun testViewModelFactory(){
    activityRule.scenario.onActivity {
        val factory = ViewModelFactory(it)
        factory.create(UserViewModel::class.java)
        factory.create(GoodViewModel::class.java)
        factory.create(RequestViewModel::class.java)
        factory.create(ChatViewModel::class.java)
    }
}
}

```

Результат тестирования представлен на рисунке 67.

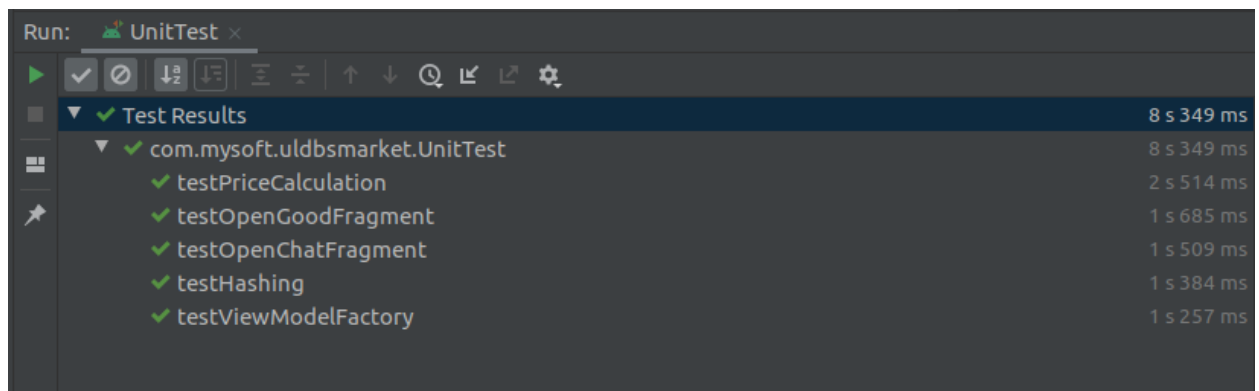


Рисунок 67 – Результат выполнения модульных тестов

В таблице 22 представлена спецификация модульных тестов.

Таблица 22 – Описание модульных тестов

Название	Описание
testPriceCalculation	Тестирование логики отображения суммарной стоимости товаров.
testOpenGoodFragment	Тестирование работы фрагмента страницы товара.
testOpenChatFragement	Тестирование работы фрагмента чатов.
tesHashing	Тестирование использования функции хеширования.
testViewModelFactory	Тестирование создания объектов компонента ViewModel

Модульные тесты показывают стабильную работу программной части системы электронной коммерции.

3.7 Метрики кода

Для оценки качества полученного исходного кода были вычислены различные метрики кода. Для анализа метрик кода используем метод `createChat`, который открывает HTTP запрос к веб-сервису и принимает ответ. Код данного метода представлен ниже.

```
suspend fun createChat(clientId: String,
managerId:String):ReqResult<Boolean>{
    val call : Call<Boolean>
    var res : Response<Boolean>? = null
    try {
        call = chatAPI.createChat(clientId,managerId)
        res = call.execute();
    }catch (e:Exception){
        Log.e(tag, "exception: " + e.toString())
        e.printStackTrace()
    }
    finally {
        if(res != null && res.isSuccessful) {
            return if(res.body() == null)
                ReqResult(false, R.string.response_empty_error, null)
            else
                ReqResult(true, "", res.body())
        }else{
            return ReqResult(false, R.string.request_err, null)}}}
```

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

Управляющий граф анализируемого кода представлен на рисунке 68.

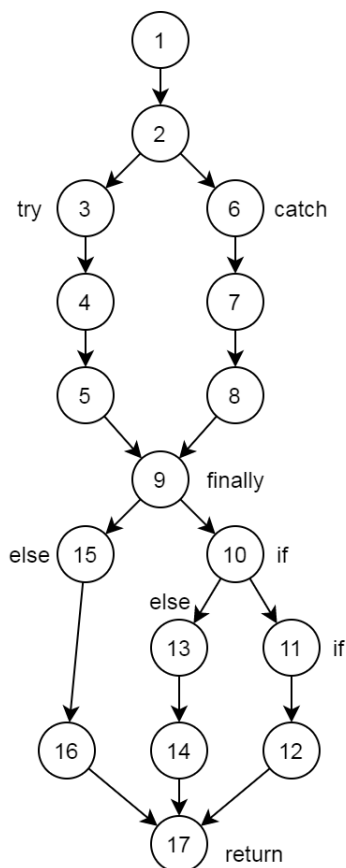


Рисунок 68 – Управляющий граф функции createChat

Для вычисления метрики необходимо подсчитать количество вершин и дуг данного графа.

Количество вершин $v = 17$, количество дуг $e = 19$.

Формулой для вычисления метрики Мак-Кейба является:

$$C = e - v + 2p,$$

где p – число компонент связности графа, то есть количество таких подмножеств вершин графа, что каждые две вершины в этом подмножестве соединены хотя бы одним путем. Для рассматриваемого графа:

$$p = 1.$$

Получили следующее значение метрики Мак-Кейба:

$$C = e - v + 2p = 4.$$

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

Еще одной важной метрикой кода является метрика Холстеда. Данный показатель составляется на основе синтаксической структуры кода. Он позволяет оценить потенциальный объем программы и сложность написания программы.

Первым этапом анализа кода с использованием метрики является составление словаря операторов и операндов исследуемого кода. Словарь представлен в таблице 23.

Таблица 23 – Словарь программы

Оператор	Число операторов	Операнд	Число операндов
val	1	call	3
:	3	Call	1
<	2	Boolean	2
>	2	res	10
var	1	Response	1
?	1	null	5
=	4	chatAPI	1
{	5	createChat	1
}	5	clientId	1
try	1	managerId	1
catch	1	execute	1
finally	1	e	3
.	8	Exception	1
(	13	Log	1
)	13	«строка»	4
“	4	printStackTrace	1
if	2	isSuccessful	1
else	2	body	2
!=	1	ReqResult	3
==	1	false	2
return	2	true	1
		toString	1

Составленный словарь позволяет получить следующие значения: Общее число операторов $N_1 = 73$, число уникальных операторов $n_1 = 21$, общее число операндов $N_2 = 47$, число уникальных операндов $n_2 = 22$.

Тогда объем словаря программы может быть вычислен, как:

$$n = n_1 + n_2 = 43.$$

Длина рассматриваемого кода будет равна:

$$N = N_1 + N_2 = 120.$$

Оценим теоретический объем программы (V) в условных единицах:

$$V = N \log_2 n \approx 651.$$

Также данные метрики позволяют оценить теоретическую длину программы, как:

$$C = n_1 \log_2(n_1) + n_2 \log_2(n_2) = 190.$$

Полученная метрика свидетельствует о том, что потенциальная сложность и объем программы приемлемы.

Для программы, написанной в ООП парадигме, актуальной метрикой оценки кода является ООП метрика Мартина. Используя классы, представленные на диаграмме классов на рисунке 32, можно вычислить такие показатели, как количество внешних зависимостей (C_a) и количество классов, зависящих от внешних абстракций (C_e).

$$C_a = 6,$$

$$C_e = 14.$$

Тогда метрика устойчивости Мартина будет равна:

$$I = \frac{C_e}{C_a + C_e} = 0.7.$$

Данный показатель меньше 1, что свидетельствует об относительной стабильности архитектуры классов приложения.

Все вычисленные метрики подтверждают отсутствие проблем с качеством программных средств системы электронной коммерции.

Выводы

После реализации на языке Java спроектированного ранее модифицированного алгоритма аутентификации, был проведен эксперимент с целью подтвердить его преимущество в надежности и криптостойкости. Суть эксперимента заключалась в получении выборок значений функции модуля. Выборки собирались для реализованного модифицированного алгоритма и для стандартной реализации алгоритма шифрования RSA. Далее, используя критерий Колмогорова была произведена оценка меры принадлежности полученных выборок нормальному закону распределение. Результат эксперимента показал, что значения модифицированного алгоритма в большей степени принадлежат закону нормального распределения, из чего можно сделать об их более случайном характере. Данный результат доказывает повышение криптостойкости модифицированного алгоритма аутентификации.

Произведена реализация базы данных на языке SQL и ее развертывание на СУБД Postgres.

Произведена реализация серверной части системы электронной коммерции в виде Java приложения с архитектурой RESTful сервисов.

Реализовано клиентское мобильное приложение системы электронной коммерции в виде Android приложение.

На этапе тестирования полученных программных средств было выполнено: нагрузочное тестирование серверной части системы электронной коммерции; функциональное тестирование системы электронной коммерции; модульное тестирование клиентского мобильного приложения системы электронной коммерции.

Результаты тестирования программных средств показали стабильную работу приложения на данных наборах тестов.

На этапе вычисления метрик кода были получены метрики Мак-Кейба и Мартина. Метрики подтверждают отсутствие проблем с качеством кода программных средств системы электронной коммерции.

Заключение

В результате выполнения диссертационной работы была разработана система электронной коммерции и модифицированный алгоритм аутентификации для системы электронной коммерции. Для достижения этих результатов был проведен анализ имеющихся решений, который выявил ряд недостатков в современных системах электронной коммерции. С целью исправления этих недостатков было решено реализовать собственную систему электронной коммерции и создать для нее собственный модифицированный алгоритм аутентификации, а также провести онтологическое и имитационное моделирование по теме предметной области.

В процессе выполнения диссертационной работы была разработана система электронной коммерции, отвечающая всем предъявленным функциональным и нефункциональным требованиям. Тестирование разработанной системы электронной коммерции показало отсутствие ошибок на тестовых наборах входных данных. Для программных средств разработанной системы электронной коммерции был вычислен ряд метрик кода, которые подтвердили, что код программный код имеет достаточно высокий уровень качества и удобства для дальнейшей поддержки. Был реализован модифицированный алгоритм аутентификации в системе электронной коммерции. 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 аутентификации в системе электронной коммерции, была зарегистрирована в государственном Реестре программ для ЭВМ, о чем свидетельствует соответствующее Свидетельство о регистрации, представленное в приложении Л.

Основные результаты диссертационной работы состоят в следующем.

1. Проведен расширенный и углубленный критический анализ предметной области по тематике магистерской диссертации.
2. Разработана и реализована система электронной коммерции с применением современных и эффективных информационных и программных технологий и методологий, включая онтологическое, имитационное и визуальное моделирование.
3. Был реализован модифицированный алгоритм аутентификации, который отличается от известных алгоритмов наличием пяти ключей, а именно: двух пар открытых и закрытых ключей шифрования и одного публичного ключа аутентификации. Алгоритм отличается также комбинированным использованием JWT аутентификации и модифицированного алгоритма асимметричного шифрования RSA, что позволяет повысить надежность процесса аутентификации в системе электронной коммерции.
4. Разработан сетевой график IT-проекта по созданию системы электронной коммерции, и определен срок выполнения и бюджет IT-проекта.
5. Предложены две дискретно-событийные имитационные модели для исследования и улучшения параметров функционирования системой электронной коммерции. Важной особенностью предложенного набора моделей является то, что он, в отличие от известных аналогичных моделей, имеет комплексный характер и позволяет учитывать как особенности работы программного обеспечения, так и человеческий фактор, связанный с деятельностью сотрудников системы электронной коммерции.
6. Предложена расширенная онтологическая модель предметной области системы электронной коммерции, 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 систем электронной коммерции, отличающаяся от известных тем, что предлагается более полный расширенный перечень классификационных признаков и систем.

8. По теме магистерской диссертации получены следующие документы:

- диплом лауреата I степени V Международного конкурса научных работ «Новая наука»;
- диплом победителя I степени XX международного научно-исследовательского конкурса «Наука и просвещение»;
- диплом победителя I степени XI международного научно-исследовательского конкурса «Молодой ученый»;
- свидетельство о государственной регистрации программы для ЭВМ в Федеральном институте промышленной собственности РФ.

Свидетельство о регистрации представлено в приложении Л.

Дипломы победителя в международных конкурсах представлены в приложении К.

9. По теме магистерской диссертации сделано 10 публикаций, включая статью в журнале, входящем в перечень ВАК, и проведена апробация на 7 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. 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: 01.06.2023).
2. 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: 01.06.2023).
3. Коновалов, В. Э. Тенденции рынка интернет-торговли в России / В. Э. Коновалов // Экономика и социум. – 2020. – №4(71). – С. 10 – 18.
4. Исследование российской аудитории интернет-магазинов // Аналитические отчеты Яндекса [Электронный ресурс]. – URL: <https://www.shopolog.ru/metodichka/analytics/issledovanie-rossiyskoy-auditorii-internet-magazinov/razrabotki> (дата обращения: 05.06.2023).
5. Обзор: Мобильность в бизнесе // CN Analytics [Электронный ресурс]. – URL: https://www.cnews.ru/reviews/mobilnost_v_biznese_2020/articles/rossijskij_rynok_mobilnoj_razrabotki (дата обращения: 05.06.2023).
6. Зайцева, Е. А. Управление бизнес-процессами торгового предприятия / Е. А. Зайцева // Международный научный журнал «Инновационная наука». – 2016. – №6. – С. 108 – 109.
7. Панкина, Т. В. Электронная коммерция и электронная торговля: подходы к классификации / Т. В. Панкина // Вестник Российской Экономической академии им. Г. В. Плеханова. – 2017. – №5(23). – С. 80 – 84.
8. Бейсенбаева, А. К. Теоритические основы электронной коммерции и перспективы / А. К. Бейсенбаева // Статистика, учет и аудит. – 2015. – №2(41). – С. 8 – 11.

9. Головенчик, Г. С. Сущность, классификация и особенности электронной коммерции / Г. С. Головенчик // Наука и инновации. – 2020. – №4(206). – С. 39 – 45.
10. Власова, А. С. Электронная коммерция / А. С. Власова // Современные тенденции в экономике и управлении: новый взгляд. – 2016. – №40-2. – С. 37 – 42.
11. Тепин, К. С. Разработка мобильного приложения для поддержки бизнес-процессов торгового предприятия / К. С. Тепин, В. Н. Князев // Научные достижения и открытия 2021: сборник статей XX Международного научно-исследовательского конкурса. – Пенза: МЦНС «Наука и Просвещение», 2021. – С. 36 – 43.
12. Chrono24 – электронный сервис предприятия по продаже часов // Chrono24 [Электронный ресурс]. – URL: <https://www.chrono24.com.ru/> (дата обращения: 21.12.2021)
13. Pull&Bear – онлайн магазин по продаже одежды // Pull&Bear [Электронный ресурс]. – URL: <https://www.pullandbear.com/ru/> (дата обращения: 21.12.2021)
14. Shopshap – интернет магазин по продаже головных уборов // Shopshap [Электронный ресурс]. – URL: <https://play.google.com/store/apps/details?id=com.shopshap.ru> (дата обращения: 21.12.2021)
15. Попова, Е. В. Моделирование бизнес-процессов промышленных организаций / Е. В. Попова // Инновации и инвестиции. – 2019. – №9. – С. 27 – 31.
16. Кабаева, И. И. Имитационное моделирование / И. И. Кабаева // International scientific review. – 2016. – №4. – С. 19– 20.
17. Плютенко, А. С. Имитационное моделирование / А. С. Плютенко // Вестник Московского государственного университета печати. – 2015. – №6. – С. 27 – 31.

18. Шилов, А. К. Анализ требований безопасности информационных систем / А. К. Шилов // Известия Южного федерального университета. Технические науки. – 2014. – №6(102). – С. 125 – 128.
19. Капустин, Ф. А. Информационная безопасность и защита информации в современном обществе / Ф. А. Капустин // Актуальные проблемы информатики. – 2016. – №4 – С. 738 – 740.
20. Алексеев, В. В. Доверие к информационной безопасности критических информационных систем / В. В. Алексеев // Труды Международного симпозиума «Надежность и качество» . – 2010. – С. 84 – 85.
21. Камбулов, Д. А. Анализ потенциальных угроз информационной безопасности и обзор существующих методов защиты / Д. А. Камбулов // Журнал E-Scio. – 2021. – №10(56) – С. 55 – 59.
22. Ажмухамедов, И. М. Принципы обеспечения комплексной безопасности информационных систем / И. М. Ажмухамедов // Вестник Астраханского государственного технического университета. – 2016. – №1 – С. 23 – 27.
23. Светлов, М. С. Методы повышения надежности программного обеспечения информационных систем / М. С. Светлов // Математические методы в технике и технологиях ММТТ. – 2015. – №10-2(59) – С. 109 – 113.
24. Introduction to JSON Web Tokens // Официальная документация технологии JWT – RFC 7519 [Электронный ресурс]. – URL: <https://jwt.io/introduction> (дата обращения: 06.06.2023)
25. Постовалов, А. А. Структурное планирование сетевыми методами / А. А. Постовалов, А. В. Авдоськина // Физико-математические науки. – 2016. – №12. – С. 145 – 149.
26. Гасанбеков, С. К. Сетевое планирование как инструмент управления проектами / С. К. Гасанбеков, Н. А. Лубенец // Известия МГТУ «МАМИ». – 2014. – №1(19). – С. 21 – 25.
27. Сетевое планирование и управление // ИНТУИТ Национальный Открытый Университет [Электронный ресурс]. – URL:

https://intuit.ru/studies/educational_groups/1697/courses/352/lecture/8389 (дата обращения: 25.05.2021)

28. Богданов, В. В. Управление проектами в Microsoft Project 2007. Учебный курс / В. В. Богданов. – М.: Питер, 2008. – 591 с.
29. Шкрыль, А. А. MS Project 2007 современное управление проектам / А. А. Шкрыль. – БХВ-Петербург. – 2007. – 256 с.
30. Кудрявцев, Е. М. Microsoft Project. Методы сетевого планирования и управления проектом / Е. М. Кудрявцев. – ДМК Пресс, 2005. – 240 с.
31. Введение в IBM Rational Unified Process // ИНТУИТ Национальный Открытый Университет [Электронный ресурс]. – URL: <https://intuit.ru/studies/courses/652/508/info> (дата обращения: 26.06.2023)
32. PostgresPro // Официальная документация СУБД PostgreSQL [Электронный ресурс]. URL: <https://postgrespro.ru/docs/postgresql/13/index> (дата обращения: 06.06.2023)
33. Kotlin Lang // Официальная документация языка Kotlin [Электронный ресурс]. – URL: <https://kotlinlang.ru> (дата обращения: 06.06.2023)
34. Атаманов, Ю. С. Обзор возможностей ERwin Data Modeler / Ю. С. Атаманов // Молодой ученый. – 2017. – № 9 (143). – С. 26 – 28
35. Джим Арлоу, Айла Нейштадт UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Джим Арлоу. – Санкт-Петербург, Изд-во Символ-Плюс, 2007.
36. Тепин, К.С. Проектирование программной системы электронной коммерции средствами языка UML / К. С. Тепин, В. Н. Князев // Материалы международной студенческой научной конференции «Поколение будущего». – СПб.: ГНИИ «Нацразвитие», 2022. – с. 85 – 89.
37. Демидов, Д. Г. Требования к современным пользовательским интерфейсам / Д. Г. Демидов // Вестник МГУП имени Ивана Федорова. – 2016. – №1. – С. 25 – 26.

38. Цуканова, Н. И. Онтологическая модель представления и организации знаний / Н. И. Цуканова – М.: Горячая линия-Телеком, 2015. – 272 с.
39. Невзорова, О. А. Инструментальные программные средства работы со знаниями / О. А. Невзорова // Образовательные технологии и общество. – 2016. – №4. – С. 19 – 20.
40. Тепин, К. С. Имитационное моделирование бизнес-процессов системы электронной коммерции / К. С. Тепин, В. Н. Князев // Материалы международной научной конференции «Высокие технологии и инновации в науке». – СПб.: ГНИИ «Нацразвитие», 2022. – с. 130 – 134.
41. Тепин, К. С. Исследование предприятия электронной коммерции средствами имитационного моделирования / К. С. Тепин, В. Н. Князев // Молодежный исследовательский потенциал: Сборник статей V Международного научно-исследовательского конкурса. – Петрозаводск: МЦНС «Новая наука», 2022. – С. 90 – 102.
42. 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).
43. Осипов, Г. С. Математическое и имитационное моделирование системы массового обслуживания / Г. С. Осипов. – М: Издательский дом Академии Естествознания, 2017. – 56 с.
44. Ослин, Б. Г. Моделирование. Имитационное моделирование СМО: учебное пособие / Б. Г. Ослин. – Томск: Издательство Томского политехнического университета, 2010. – 128 с.
45. 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. 71 – 84.
46. 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.

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

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

49. Ревина, С. Ю. Тенденции и перспективы российской электронной коммерции / С. Ю. Ревина // Современные тенденции в экономике и управлении: новый взгляд. – 2016. – №40-2. – С. 37 – 42.

50. Тепин, К. С. Исследование работы службы доставки предприятия электронной коммерции средствами имитационного моделирования / К. С. Тепин, В. Н. Князев // Материалы всероссийской (национальной) научной конференции «Фундаментальные и прикладные исследования. Актуальные проблемы и достижения». – СПб.: ГНИИ «Нацразвитие», 2022. – с. 26 – 28.

51. Богданова А.Ю., Тепляных 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 (2023). [Электронный ресурс]. URL: <https://science.kuzstu.ru/wpcontent/Events/Conference/RM/2021/RM21/pages/Articles/031605.pdf> (дата обращения: 26.03.2023).

52. 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.

53. Арно Лоре, Проектирование Веб-API / Пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2020. – 440 с.

54. Тепин, К. С. Разработка мобильного приложения для поддержки бизнес-процессов торгового предприятия / К. С. Тепин, В. Н. Князев // Научные достижения и открытия 2021: сборник статей XX Международного

научно-исследовательского конкурса. – Пенза: МЦНС «Наука и Просвещение», 2021. – С. 36 – 43.

55. Android Developer // Официальная документация API Android [Электронный ресурс]. URL: <https://developer.android.com/> (дата обращения: - 06.06.2023)

56. Колисниченко, Д. Безопасный Android. Защищаем свои деньги и данные от кражи / Д. Колисниченко. – М.: БХВ-Петербург, 2020. – 160 с.

57. InfoWatch аналитика [Электронный ресурс]. – URL: <https://www.infowatch.ru/analytics/analitika/v-2021-stalo-bolshe-umyshlennykh-utechek> (дата обращения: 06.06.2023).

58. Зубов, А. Н. Математика кодов аутентификации / А. Н. Зубов. – М.: Гелиос АРВ, 2017. – 959 с.

59. Малюк, А. А. Теория защиты информации / А. А. Малюк. – М.: Горячая линия – Телеком, 2020. – 114 с.

60. Лукашкин, Е. В. Разработка аутентификации, базирующейся на JWT-токенах / Е. В. Лукашкин // Материалы XXIII Республиканской научной конференции студентов и аспирантов. – Гомель. – 2020. – С. 268 – 269.

61. Макаров, Д. А. Механизм авторизации с использованием технологии JWT / Д. А. Макаров // Теория и практика современной науки. – 2020. – №1(55) – С. 474 – 476

62. Светлов, М. С. Методы повышения надежности программного обеспечения информационных систем / М. С. Светлов // Математические методы в технике и технологиях ММТТ. – 2015. – №10-2(59) – С. 109 – 113

63. Шаньгин, В. Ф. Информационная безопасность и защита информации / В. Ф. Шаньгин. – Москва: Мир, 2017. – 826 с.

64. Баричев, С. Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. – Москва: СПб. [и др.] : Питер, 2020. – 176 с.

65. Герман, О. Н. Теоретико-числовые методы в криптографии / О.Н. Герман, Ю.В. Нестеренко. – М.: Академия, 2021. – 272 с.

66. Зубов, А. Ю. Криптографические методы защиты информации. Совершенные шифры / А.Ю. Зубов. – М.: Гелиос АРВ, 2017. – 192 с.
67. Осмоловский, С. А. Стохастическая информатика. Инновации в информационных системах / С.А. Осмоловский. – М.: Горячая линия – Телеком, 2018. – 322 с.
68. Стохастические методы и средства защиты информации в компьютерных системах и сетях / Под редакцией И.Ю. Жукова. – М.: КУДИЦ-Пресс, 2021. – 512 с
69. Чичикин, Г. Я. Криптосистема RSA / Г. Я. Чичикин, Д. А. Семенов // Наука, образование и культура . – 2019. – №3(63) – С. 56 – 58.
70. Замула, А. А. Генераторы псевдослучайных чисел, основанные на дискретном логарифме / А. А. Замула, Д. А. Семченко // Технологический аудит и резервы производства. – 2019. – №5(45) – С. 28 – 33.
71. Мицель А. А./ Прикладная математическая статистика. – Томск: ТУСУР. – 2015. – 81с.

Приложение А
ГЛОССАРИЙ
(справочное)

Электронная коммерция – собирательный термин, обозначающий сферу цифровой экономики включая финансовые и торговые транзакции, осуществляемые при помощи компьютерных сетей.

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) – итеративная методология разработки программного обеспечения.

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.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, широко используемый в веб-системах.

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

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) – асимметричный алгоритм шифрования, основывающийся на вычислительной сложности задачи определения делителей больших простых чисел.

RESTful – архитектурный шаблон, определяющий организацию кода серверного приложения. Обеспечивает легкий обмен данными с клиентским приложением и возможность масштабирования.

Приложение Б
ИСХОДНЫЙ КОД ИМИТАЦИОННЫХ МОДЕЛЕЙ
(рекомендуемое)

Листинг Б.1 – Исходный код имитационной модели системы электронной коммерции

```

constTlux EQU 5000
  constTcasual EQU 2500
  constTshoes EQU 2800
  constTaccessory EQU 4500

constSA EQU 2 ; (сек) ответ сервера
constSPA EQU 50 ; (сек) транзакция электронной оплаты
constOT EQU 1800 ; (сек) обработка заказа

managersK STORAGE 2 ; кол-во менеджеров

T1Table TABLE M1,1000,1000,25
T2Table TABLE M1,1000,1000,25
T3Table TABLE M1,1000,1000,25
T4Table TABLE M1,1000,1000,25
AllTable TABLE M1,1000,5000,25

GENERATE (Exponential(1,0,constTlux))
ASSIGN 1,1 ;Брендовая и люксовая одежда
TRANSFER ,NEXT

GENERATE (Exponential(1,0,constTcasual))
ASSIGN 1,2 ;Повседневная одежда
TRANSFER ,NEXT

GENERATE (Exponential(1,0,constTshoes))
ASSIGN 1,3 ;Обувь
TRANSFER ,NEXT

GENERATE (Exponential(1,0,constTaccessory))
ASSIGN 1,4 ;Аксессуары
TRANSFER ,NEXT

NEXT QUEUE QH1
  SEIZE K1
  DEPART QH1
  ADVANCE (Exponential(1,0,constSA)) ;Обработка запроса на
просмотр товара
  RELEASE K1

```

```

QUEUE QH3
SEIZE K3
DEPART QH3
ADVANCE (Exponential(1,0,constSPA)) ;Онлайн оплата
RELEASE K3

QUEUE QH4
ENTER managersK
DEPART QH4
ADVANCE (Exponential(1,0,constOT)) ;Обработка заказа
менеджером
LEAVE managersK

QUEUE QH5
SEIZE K5
DEPART QH5
ADVANCE (Exponential(1,0,constSA))
RELEASE K5

TABULATE AllTable

TEST E P1,1,Mark2
TABULATE T1Table
TRANSFER ,FIN

Mark2 TEST E P1,2,Mark3
TABULATE T2Table
TRANSFER ,FIN

Mark3 TEST E P1,3,Mark4
TABULATE T3Table
TRANSFER ,FIN

Mark4 TABULATE T4Table

FIN TERMINATE 1
START 2880 ; 1мес / 15мин

```

Листинг Б.2 – Исходный код имитационной модели службы доставки для системы электронной коммерции.

```

constT EQU 900

constSE EQU 2 ; (сек) ответ сервера
constSD EQU 1800 ;доставка товара
deliverersK STORAGE 3 ;кол-во доставщиков

AllTable TABLE M1,1000,1000,25

GENERATE (Exponential(1,0,constT))

QUEUE QH1
SEIZE K1
DEPART QH1
ADVANCE (Exponential(1,0,constSE)) ;Обработка запроса на
просмотр
RELEASE K1

QUEUE QH2
ENTER deliverersK
DEPART QH2
ADVANCE (Exponential(1,0,constSD)) ;Обработка заказа менеджером
LEAVE deliverersK

TABULATE AllTable

TERMINATE 1
START 2880 ; 1мес / 15мин

```

Приложение В
ОПИСАНИЕ СУЩНОСТЕЙ БАЗЫ ДАННЫХ
(рекомендуемое)

Таблица В.1 – Описание сущностей предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь (клиент, менеджер, администратор)	Записи пользователей информационной системы	Стержневая	15000	Каждый раз когда регистрируется новый пользователь.
2	Чат	Диалог между клиентом и менеджером	Ассоциативная	10000000	Когда пользователь начинает общение с менеджером.
3	Сообщение	Сообщение в процессе чата	Стержневая	10000000	Каждый раз когда отправляется какое-либо сообщение.
4	Товар	Основной объект бизнес-процессов	Стержневая	1500	Редко
5	Каталог	Сущность, объединяющая класс товаров	Стержневая	100	Редко
6	Заявка пользователя	Включает в себя товары которые пользователь одновременно заказал.	Стержневая	15000	Каждый раз, когда пользователь оформляет заявку
7	Заявка на товар	Заявка на конкретный товар. Обеспечивает связь М:М между заявкой пользователя и товаром	Ассоциативная	30000	Каждый раз, когда пользователь оформляет заявку
8	Параметр заявки	Параметры заявки. Предлагается для заполнения пользователю на этапе оформления	Стержневая	50	Редко
9	Оценка	Оценка и отзыв клиента о товаре.	Стержневая	15000	Если пользователь захочет оценить товар

Таблица В.2 – Описание связей между сущностями предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мно- жес- тво	Обязательность	Ограничения
1	Пользователь(клиент)	Чат	Пользователь класса клиент может имеет один чат с автоматически определенным для него менеджером	1 : 1	Необязательное (если у клиента нет вопросов)	Роль пользователя – клиент
2	Пользователь (менеджер)	Чат	Менеджер может вести диалоги со множеством клиентов.	1:M	Необязательно (если клиенты не ведут диалог)	Роль пользователя - менеджер
3	Чат	Пользователь (клиент)	Каждый чат связан как с клиентом, так и с менеджером	M:1	Необязательное	Необходимо избегать дублирования чатов для конкретного клиента и конкретного менеджера.
4		Пользователь (менеджер)				
5	Пользователь(клиент)	Заявка пользователя	Каждый клиент может оставлять множество заявок	1:M	Необязательно	Роль пользователя – клиент
6	Пользователь (менеджер)	Заявка пользователя	Каждый менеджер обрабатывает множество заявок	1:M	Необязательно	Роль пользователя - менеджер
7	Чат	Сообщение	В чате может быть множество сообщений.	1:M	Обязательно	Необходимо наличие худобы одного сообщения
8	Заявка пользователя	Заявка на товар	В заявке пользователя может содержаться множество товаров	1:M	Обязательной	Хотябы 1 пункт в заявке
9	Товар	Заявка на товар	Один объект может входить в множество разных заявок	1:M	Необязательно	
10	Пользователь	оценка	Пользователь (клиент) может оставить оценку товару	1:M	Необязательно	Пользователь заказывал товар
11	Товар	Оценка	Один товар может иметь множество оценок от разных пользователей	1:M	Необязательно	
12	Каталог	Товар	В каталоге может содержаться множество товаров	1:M	Необязательно	При развертывании БД задается начальный каталог.
13	Заявка пользователя	История статусов заявки	Каждая заявка обладает несколькими записями изменениями их состояния.	1:M	Обязательно	Заявка должна иметь хотябы начальное состояние
14	Статус	История статусов заявки	Каждый статус может использоваться во множестве записей историй статусов для конкретных товаров	1:M	Необязательно	

Таблица В.3 – Атрибуты сущности «Пользователь»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример	Примечание
Email	Логин пользователя	Символьный	20	A-z, 0-9, ., @	Нет	vladimir99@mail.ru	Должен удовлетворять паттерну email адреса
Пароль	Пароль пользователя	Символьный	20	A-z, 0-9	Нет	Qwerty111	Не меньше 6 символов
Роль	Роль пользователя	Перечисление		Клиент, Менеджер, Администратор	нет	Клиент	
Имя	Имя пользователя	Символьный	20	A-z, A-я	Нет	Константин	
Дата рождения	Дата рождения пользователя	Символьный	10	##.##.#### где # - это 0-9	Да	12.12.1999	Соответствие шаблону
Номер телефона	Номер телефона	Числовой	11	0-9	Да	89004002233	

Таблица В.4 – Атрибуты сущности «Сообщение»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Текст	Текст сообщения	Символьный	510	символы UTF-8	Нет	Здравствуйте, у меня возникла проблема!
Дата и время	Метка времени когда было отправлено сообщение	Дата и время (метка времени)	8 байт		Нет	13.11.2014:00

Таблица В.5 – Атрибуты сущности «Каталог»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Название	Название каталога	Символьный	20	символы UTF-8	Нет	Наши товары

Таблица В.6 – Атрибуты сущности «Товар»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Название	Название товара	Символьный	20	символы UTF-8	Нет	Размещение рекламы
Цена	Цена	Числовой с плав. точкой 2 знака после запятой	5-17 байт	0.00 - *.99	Да	1250.00
Описание	Текстовое описание объекта бизнес-процесса	Символьный	500	символы UTF-8	Да	В услугу входит обработка, и предоставление данных
Картинка	Путь в файловой системе	Символьный	255	A-z - . /	Да	/res/img/32aarr.png

Таблица В.7 – Атрибуты сущности «Заявка пользователя»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример	Примечание
Данные для оплаты	Номер карты и код оплаты	Символьный	19	0-9	Нет	1111222 2333344 4123	Использование шифрования

Примечание: дату поступления заявки можно посчитать из истории статусов заявки, найдя дату, когда было установлен первый статус заявки.

Таблица В.8 – Атрибуты сущности «Оценка»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Оценка	Оценка товара	Числовой	1 байт	{0,1,2,3,4,5}	Нет	5
Отзыв	Отзыв пользователя о товаре	Символьный	255	Любые символы принятой кодировки	Да	Очень понравился товар!
Дата и время	Время, когда была поставлена оценка	Дата и время (метка времени)	8 байт		Нет	13.11.20 14:00

Таблица В.9 – Атрибуты сущности «Тип параметра»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Название параметра		Символьный	64	Любые символы принятой кодировки	Нет	Цвет
Вопрос пользователю	Что выводить пользователю при вопросе параметра	Символьный	255	Любые символы принятой кодировки	Нет	Введите желаемый цвет: доступно красный/черный/белый

Таблица В.10 – Атрибуты сущности «Значение параметра»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Значение	Значение определенного параметра –	Символьный	64	Любые символы принятой кодировки	Нет	Красный

Таблица В.11 – Атрибуты сущности «Статус»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Название статуса	Статус заявки	Символьный	64	Любые символы принятой кодировки	Нет	Выполняется

Таблица В.12 – Атрибуты сущности «История статусов заявки»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения null	Пример
Дата и время установления	Метка времени, когда был установлен данный статус заявки	(метка времени)	8 байт		Нет	12.11.2020
Комментарий	Комментарий, при установлении статуса заявки.	Символьный	64	Любые символы принятой кодировки	Да	Заявка не может быть выполнена. Реквизиты не верные.

Приложение Г
ИСХОДНЫЙ КОД МОДИФИЦИРОВАННОГО АЛГОРИТМА
АУТЕНТИФИКАЦИИ
(рекомендуемое)

Листинг Г.1 – Файл Main.java

```
package com.penzasoft.uldbscryptolib;

import java.nio.charset.StandardCharsets;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.spec.InvalidKeySpecException;

/**
 *
 * @author ktepin
 */
public class Main {

    private static int KEY_LENGTH = 2048; //1024 or 2048 or 4096

    public static void main(final String[] args){
        try {

            CustomRSA rsa = new CustomRSA(KEY_LENGTH);
            String payload = "RSA Test text abc123456";
            byte[] cipher =
rsa.encrypt(payload.getBytes(StandardCharsets.UTF_8));

            System.out.println("\nTesting RSA:");
            System.out.println("Example:");
            System.out.println(payload);
            System.out.println("Cipher:");
            System.out.println(cipher);

            byte[] ex = rsa.decrypt(cipher);
            System.out.println("Decrypt:");
            System.out.println(new String(ex, StandardCharsets.UTF_8));

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Листинг Г.2 – Файл CustomRandom.java

```
package com.penzasoft.uldbscryptolib;
import java.util.Random;

public class CustomRandom extends Random {
    private static final long serialVersionUID = 1L;
    private static final int SEED_MULTIPLIER = 1103515245;
    private static final int SEED_INCREMENT = 12345;
    private static final int MODULUS = 2147483647;
    private static final int OFFSET = 123456789;

    private int seed;

    public CustomRandom(int seed) {
        this.seed = seed;
    }
}
```

```

@Override
protected int next(int bits) {
    seed = (SEED_MULTIPLIER * seed + SEED_INCREMENT) % MODULUS;
    return seed >>> (32 - bits);
}

@Override
public synchronized void setSeed(long seed) {
    this.seed = (int) (seed ^ OFFSET);
}
}

```

Листинг Г.3 – Файл CustomRSA.java

```

package com.penzasoft.uldbcryptolib;
import java.math.BigInteger;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.RSAPrivateKeySpec;
import java.security.spec.RSAPublicKeySpec;
import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
/**
 *
 * @author ktepin
 * RSA algorithm is based on several mathematical formulas, including:
 * Euler's totient function:  $\phi(n) = (p - 1) * (q - 1)$ 
 * Modular exponentiation:  $a^b \bmod n$ 
 * Modular inverse:  $a^{-1} \bmod n$ 
 * These formulas are used to generate the public and private keys, encrypt and
 * decrypt messages, and perform other operations in the RSA algorithm.
 */
public class CustomRSA {

    private PublicKey publicKey;
    private PrivateKey privateKey;

    public CustomRSA(int bitLength) throws NoSuchAlgorithmException,
    NoSuchProviderException, InvalidKeySpecException {
        SecureRandom random = new SecureRandom(); //TODO custom random

        // Generate two large prime numbers, p and q
        BigInteger p = BigInteger.probablePrime(bitLength / 2, random);
        BigInteger q = BigInteger.probablePrime(bitLength / 2, random);

        // Calculate modulus  $n = p * q$ 
        BigInteger modulus = p.multiply(q);

        // Calculate Euler's totient function  $\phi(n) = (p - 1) * (q - 1)$ 
        BigInteger phi =
        (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

```

```

        // Choose public key e such that 1 < e < phi(n) and gcd(e, phi(n)) = 1
        BigInteger publicKeyExponent = BigInteger.probablePrime(bitLength / 4,
random);
        while (phi.gcd(publicKeyExponent).compareTo(BigInteger.ONE) > 0 &&
publicKeyExponent.compareTo(phi) < 0) {
            publicKeyExponent = publicKeyExponent.add(BigInteger.ONE);
        }

        // Calculate private key d such that d * e ≡ 1 (mod phi(n))
        BigInteger privateKeyExponent = publicKeyExponent.modInverse(phi);

        // Generate PublicKey and PrivateKey objects using the modulus and
exponents
        RSAPublicKeySpec publicKeySpec = new RSAPublicKeySpec(modulus,
publicKeyExponent);
        KeyFactory publicKeyFactory = KeyFactory.getInstance("RSA");
        publicKey = publicKeyFactory.generatePublic(publicKeySpec);

        RSAPrivateKeySpec privateKeySpec = new RSAPrivateKeySpec(modulus,
privateKeyExponent);
        KeyFactory privateKeyFactory = KeyFactory.getInstance("RSA");
        privateKey = privateKeyFactory.generatePrivate(privateKeySpec);
    }

    public byte[] encrypt(byte[] message) throws NoSuchAlgorithmException,
NoSuchPaddingException, InvalidKeyException, InvalidKeySpecException,
IllegalBlockSizeException, BadPaddingException {
        // Encrypt the message using the public key
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, publicKey);
        return cipher.doFinal(message);
    }

    public byte[] decrypt(byte[] ciphertext) throws NoSuchAlgorithmException,
NoSuchPaddingException, InvalidKeyException, InvalidKeySpecException,
IllegalBlockSizeException, BadPaddingException {
        // Decrypt the ciphertext using the private key
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        return cipher.doFinal(ciphertext);
    }

    public PublicKey getPublicKey() {
        return publicKey;
    }

    public PrivateKey getPrivateKey() {
        return privateKey;
    }
}

```

Листинг Г.4 – Файл CustomJWT.java

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.penzasoft.uldbscryptolib;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import io.jsonwebtoken.Claims;

```

```

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

/**
 *
 * @author ktepin
 */
public class CustomJWT {
    private static final String TOKEN_ISSUER = "admin";

    private String secretKey = "default_key54321";
    private byte[] secretKeyBytes = secretKey.getBytes();
    private int tokenExpTime = 36000; // in seconds

    public CustomJWT() {
    }

    public CustomJWT(String secretKey, int tokExpTime) {
        this.secretKey = secretKey;
        this.secretKeyBytes = this.secretKey.getBytes();
        this.tokenExpTime = tokExpTime;
    }

    public String generateToken(String subject) {
        Date now = new Date();
        Date expiration = new Date(now.getTime() + tokenExpTime * 1000);
        Map<String, Object> claims = new HashMap<>();
        claims.put("sub", subject);
        return Jwts.builder()
            .setClaims(claims)
            .setIssuer(TOKEN_ISSUER)
            .setIssuedAt(now)
            .setExpiration(expiration)
            .signWith(Keys.hmacShaKeyFor(secretKeyBytes),
SignatureAlgorithm.HS256)
            .compact();
    }

    public boolean verifyToken(String token) {
        try {

Jwts.parserBuilder().setSigningKey(Keys.hmacShaKeyFor(secretKeyBytes)).build().p
arseClaimsJws(token);
            return true;
        } catch (Exception e) {
            return false;
        }
    }

    public String getSubjectFromToken(String token) {
        Jws<Claims> claims =
Jwts.parserBuilder().setSigningKey(Keys.hmacShaKeyFor(secretKeyBytes)).build().p
arseClaimsJws(token);
        return claims.getBody().getSubject();
    }
}

```

Приложение Д
СКРИПТЫ СОЗДАНИЯ БАЗЫ ДАННЫХ
(рекомендуемое)

Листинг Д.1 – Скрипт создания таблиц

```

CREATE TABLE public."user" (
    uuid uuid NOT NULL,
    email varchar(128) NOT NULL,
    pass varchar(64) NOT NULL,
    "role" smallint NOT NULL,
    "name" varchar(64) NULL,
    birth_date date NULL,
    phone varchar(11) NULL,
    CONSTRAINT user_pk PRIMARY KEY (uuid)
);

CREATE TABLE public.chat (
    uuid uuid NOT NULL,
    client_uuid uuid NOT NULL,
    manager_uuid uuid NOT NULL,
    CONSTRAINT chat_pk PRIMARY KEY (uuid),
    CONSTRAINT chat_fk_client FOREIGN KEY (client_uuid) REFERENCES
public."user"(uuid),
    CONSTRAINT chat_fk_manager FOREIGN KEY (manager_uuid) REFERENCES
public."user"(uuid)
);

CREATE TABLE public.message (
    uuid uuid NOT NULL,
    chat_uuid uuid NOT NULL,
    user_uuid uuid NOT NULL,
    "text" text NOT NULL,
    "timestamp" timestamp(0) NOT NULL,
    CONSTRAINT message_pk PRIMARY KEY (uuid),
    CONSTRAINT message_fk_chat FOREIGN KEY (chat_uuid) REFERENCES
public.chat(uuid),
    CONSTRAINT message_fk_user FOREIGN KEY (user_uuid) REFERENCES
public."user"(uuid)
);

CREATE TABLE public.status (
    uuid uuid NOT NULL,
    "name" varchar(64) NOT NULL,
    "is_terminal" smallint NOT NULL default 0,
    "is_initial" smallint NOT NULL default 0,
    CONSTRAINT status_pk PRIMARY KEY (uuid)
);

CREATE TABLE public.request (

```

```

        uuid uuid NOT NULL,
        client_uuid uuid NOT NULL,
        manager_uuid uuid NOT NULL,
        payment_data text NOT NULL,
        CONSTRAINT request_pk PRIMARY KEY (uuid),
        CONSTRAINT request_fk_client FOREIGN KEY (client_uuid) REFERENCES
public."user"(uuid),
        CONSTRAINT request_fk_manager FOREIGN KEY (manager_uuid) REFERENCES
public."user"(uuid)
);

```

```

CREATE TABLE public.status_history (
    uuid uuid NOT NULL,
    request_uuid uuid NOT NULL,
    status_uuid uuid NOT NULL,
    setup_timestamp timestamp(0) NOT NULL,
    "comment" text NULL,
    CONSTRAINT status_history_pk PRIMARY KEY (uuid),
    CONSTRAINT status_history_fk_request FOREIGN KEY (request_uuid)
REFERENCES public.request(uuid),
    CONSTRAINT status_history_fk_status FOREIGN KEY (status_uuid)
REFERENCES public.status(uuid)
);

```

```

CREATE TABLE public."catalog" (
    uuid uuid NOT NULL,
    "name" varchar(128) NOT NULL,
    CONSTRAINT catalog_pk PRIMARY KEY (uuid)
);

```

```

CREATE TABLE public.good (
    uuid uuid NOT NULL,
    "name" varchar(128) NOT NULL,
    price numeric(2) NOT NULL,
    descr text NOT NULL,
    img_path varchar(512) NULL,
    CONSTRAINT good_pk PRIMARY KEY (uuid)
);

```

```

CREATE TABLE public.good (
    uuid uuid NOT NULL,
    name varchar(256) NOT NULL,
    price numeric(16,2) NOT NULL,
    descr text NOT NULL,
    img_path varchar(512) NULL,
    CONSTRAINT good_pk PRIMARY KEY (uuid)
);

```

```

);
ALTER TABLE public.good ADD catalog_uuid uuid NULL;
ALTER TABLE public.good ADD CONSTRAINT good_fk_catalog FOREIGN KEY
(catalog_uuid) REFERENCES public."catalog"(uuid);

CREATE TABLE public.feedback (
    uuid uuid NOT NULL,
    user_uuid uuid NOT NULL,
    good_uuid uuid NOT NULL,
    grade smallint NOT NULL,
    feedback text NULL,
    "timestamp" timestamp(0) NOT NULL,
    CONSTRAINT feedback_pk PRIMARY KEY (uuid),
    CONSTRAINT feedback_fk_user FOREIGN KEY (user_uuid) REFERENCES
public."user"(uuid),
    CONSTRAINT feedback_fk_good FOREIGN KEY (good_uuid) REFERENCES
public.good(uuid)
);

CREATE TABLE public.good_request (
    uuid uuid NOT NULL,
    request_uuid uuid NOT NULL,
    good_uuid uuid NOT NULL,
    CONSTRAINT good_request_pk PRIMARY KEY (uuid),
    CONSTRAINT good_request_fk_request FOREIGN KEY (request_uuid)
REFERENCES public.request(uuid),
    CONSTRAINT good_request_fk_good FOREIGN KEY (good_uuid) REFERENCES
public.good(uuid)
);

CREATE TABLE public.parametr (
    uuid uuid NOT NULL,
    good_uuid uuid NOT NULL,
    question varchar(512) NOT NULL,
    "name" varchar(128) NOT NULL,
    CONSTRAINT parametr_pk PRIMARY KEY (uuid),
    CONSTRAINT parametr_fk_good FOREIGN KEY (good_uuid) REFERENCES
public.good(uuid)
);

CREATE TABLE public.parametr_value (
    uuid uuid NOT NULL,
    parametr_uuid uuid NOT NULL,
    good_request_uuid uuid NOT NULL,
    value varchar(256) NOT NULL,
    CONSTRAINT parametr_value_pk PRIMARY KEY (uuid),

```

```

        CONSTRAINT parametr_value_fk_parametr FOREIGN KEY (parametr_uuid)
REFERENCES public.parametr(uuid),
        CONSTRAINT parametr_value_fk_good_req FOREIGN KEY
(good_request_uuid) REFERENCES public.good_request(uuid)
);

```

Листинг Д.2 – Скрипт создания хранимой функции

```

CREATE OR REPLACE FUNCTION getfreemanager(mng_role_code integer,
rqst_finish_stat uuid)
RETURNS uuid
LANGUAGE sql
AS $$

select mangr_uuid from(

select u.uuid as mangr_uuid,
(select count(*) from request r where not exists (
        select 1 from status_history sh where
            sh.request_uuid = r.uuid
            and sh.status_uuid = rqst_finish_stat
    )
    and r.manager_uuid = u.uuid
) as not_finished_requests
from "user" u where u."role" = mng_role_code

) as analyzing order by not_finished_requests limit 1;

$$;

```

Приложение Е
ИСХОДНЫЙ КОД СЕРВЕРНОЙ ЧАСТИ СИСТЕМЫ ЭЛЕКТРОННОЙ
КОММЕРЦИИ
(рекомендуемое)

Листинг Е.1 – Исходный код серверной части системы электронной коммерции

```
//Листинг класса ChatService
@Stateless
@Path("chat")
public class ChatService {

    @Inject
    private ChatFacade chatFacade;

    @GET
    @Path("getChatsByClient")
    @Produces({MediaType.APPLICATION_JSON})
    public List<Chat> getChatsByClientId(@HeaderParam("client") UUID
clientId){
        return chatFacade.getChatsByClientId(clientId);
    }

    @GET
    @Path("getChatsByManager")
    @Produces({MediaType.APPLICATION_JSON})
    public List<Chat> getChatsByManagerId(@HeaderParam("manager") UUID
managerId){
        return chatFacade.getChatsByManagerId(managerId);
    }

    @GET
    @Path("createChat")
    @Produces({MediaType.APPLICATION_JSON})
    public Boolean createChat(
        @HeaderParam("client") UUID client,
        @HeaderParam("manager") UUID manager){
        return chatFacade.createChat(client, manager);
    }

    @GET
    @Path("autoCreateChat")
    @Produces({MediaType.APPLICATION_JSON})
    public Boolean autoCreateChat(@HeaderParam("client") UUID client){
        return chatFacade.createChatWithRandomManager(client);
    }
}
```

```
//Листинг класса FeedbackService
@Stateless
@Path("feedback")
public class FeedbackService {
    @Inject
    private FeedbackFacade feedbackFacade;

    @POST
    @Path("postFeedback")
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
    public Boolean postFeedback(
        @HeaderParam("user") UUID user,
        @HeaderParam("good") UUID good,
        Feedback feedback){
        return feedbackFacade.addFeedback(feedback, user, good);
    }

    @GET
    @Path("getAllFeedbacks")
    @Produces({MediaType.APPLICATION_JSON})
    public List<Feedback> getAllFeedbacks(){
        return feedbackFacade.getAllFeedbacks(100, 0);
    }
}
```

```
//Листинг класса GoodService
@Stateless
@Path("good")
public class GoodService {
    @Inject
    private GoodFacade goodFacade;

    @GET
    @Path("getGoods")
    @Produces({ MediaType.APPLICATION_JSON})
    public List<Good> getMessagesForChat(){
        return goodFacade.getAllGoods(100, 0);
    }

    @GET
    @Path("getGoodsByPopularity")
    @Produces({MediaType.APPLICATION_JSON})
    public List<Good> getGoodsByPopularity(){
        return goodFacade.getGoodsByPopularity(100, 0);
    }

    @GET
    @Path("getGoodsByRequests")
    @Produces({MediaType.APPLICATION_JSON})
    public List<Good> getGoodsByRequests(){
```

```

        return goodFacade.getGoodsByRequests(100, 0);
    }

    @GET
    @Path("getFullInfoForGood/{good}")
    @Produces({MediaType.APPLICATION_JSON})
    public FullGoodInfoDto getFullGoodInfo(@PathParam("good") UUID good){
        return goodFacade.getFullInfoForGood(good, 50, 0);
    }

    @POST
    @Path("createGood")
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
    public Boolean createGood(@HeaderParam("catalog") UUID catalogId, Good
good){
        return goodFacade.createGood(good, catalogId);
    }
}

//Листинг класса ChatFacade
@Stateless
public class ChatFacade {

    private static final Logger logger =
Logger.getLogger(ChatFacade.class.getName());

    @PersistenceContext(unitName = "testPU")
    private EntityManager entityManager;

    public List<Chat> getChatsByClientId(UUID clientId){
        return entityManager
            .createQuery("SELECT ch FROM Chat ch WHERE ch.client.uuid =
:cl_id", Chat.class)
            .setParameter("cl_id", clientId)
            .getResultList();
    }

    public List<Chat> getChatsByManagerId(UUID managerId){
        return entityManager
            .createQuery("SELECT ch FROM Chat ch WHERE ch.manager.uuid =
:mn_id", Chat.class)
            .setParameter("mn_id", managerId)
            .getResultList();
    }

    public Boolean createChat(UUID userId, UUID managerId){
        try{
            // Поиск дубликатов. Если чат уже есть то мы не можем создать новый
            List<Chat> chats = entityManager

```

```

        .createQuery("SELECT ch FROM Chat ch WHERE ch.manager.uuid =
:mn_id AND ch.client.uuid = :cl_id", Chat.class)
        .setParameter("mn_id", managerId)
        .setParameter("cl_id", userId)
        .getResultList();
    if(!chats.isEmpty())
        return false;

    User client = entityManager.find(User.class, userId);
    User manager = entityManager.find(User.class, managerId);
    if(client == null || manager == null)
        return false;

    Chat newChat = new Chat();
    newChat.setUuid(UUID.randomUUID());
    newChat.setClient(client);
    newChat.setManager(manager);
    entityManager.persist(newChat);
    entityManager.flush();
    return true;
}

catch(Exception e){
    //result.setMessage(result.getMessage()+" "+e.getMessage());
    logger.log(Level.SEVERE, e.getMessage(), e);
    return false;
}

}

public Boolean createChatWithRandomManager(UUID userId){
    try{
        List<Chat> chats = getChatsByClientId(userId);
        if(chats.size() > 0)
            return false; //У пользователя уже есть чаты

        Query q = entityManager.createNativeQuery(
            "select * from getfreemanager(?::integer, ?::uuid);"
        );
        q.setParameter(1, MANAGER_ROLE_CODE);
        q.setParameter(2, FINAL_STATUS_UUID);
        String freeManagerUuid = q.getSingleResult().toString();
        if(freeManagerUuid != null)
            return createChat(userId, UUID.fromString(freeManagerUuid));
        else
            return false;
    }

    catch(Exception e){
        //result.setMessage(result.getMessage()+" "+e.getMessage());
        logger.log(Level.SEVERE, e.getMessage(), e);
        return false;
    }
}

```

```

    }

    //Листинг класса FeedbackFacade
    @Stateless
    public class FeedbackFacade {
        @PersistenceContext(unitName = "testPU")
        private EntityManager entityManager;

        private static final Logger logger =
        Logger.getLogger(FeedbackFacade.class.getName());

        public Boolean addFeedback(Feedback feedback, UUID user, UUID good){
            try{
                feedback.setUuid(UUID.randomUUID());
                User u = entityManager.find(User.class, user);
                Good g = entityManager.find(Good.class, good);
                if(u == null || g == null)
                    return false;
                List<Feedback> chats = entityManager
                    .createQuery("SELECT fb FROM Feedback fb WHERE fb.user.uuid
= :usr_id AND fb.good.uuid = :good_id",Feedback.class)
                    .setParameter("usr_id", u.getUuid())
                    .setParameter("good_id", g.getUuid())
                    .getResultList();
                if(!chats.isEmpty())
                    return false;

                feedback.setUser(u);
                feedback.setGood(g);
                feedback.setTimestamp(new Date());
                entityManager.persist(feedback);
                entityManager.flush();
                return true;
            }
            catch(Exception e){
                //result.setMessage(result.getMessage()+" : "+e.getMessage());
                logger.log(Level.SEVERE, e.getMessage(),e);
                return false;
            }
        }

        public List<Feedback> getAllFeedbacks(int limit, int offset){
            return entityManager.createQuery(
                "SELECT ff FROM Feedback ff", Feedback.class)
                .setMaxResults(limit - offset)
                .setFirstResult(offset)
                .getResultList();
        }
    }
}

```

```
//Листинг класса GoodFacade
@Stateless
public class GoodFacade {

    private static final Logger logger =
Logger.getLogger(GoodFacade.class.getName());

    @PersistenceContext(unitName = "testPU")
    private EntityManager entityManager;

    public List<Good> getAllGoods(int limit, int offset){
        return entityManager.createQuery(
            "SELECT gg FROM Good gg",Good.class)
            .setMaxResults(limit - offset)
            .setFirstResult(offset)
            .getResultList();
    }

    public List<Good> getGoodsByPopularity(int limit, int offset){
        return entityManager.createQuery(
            "SELECT gg FROM Good gg GROUP BY gg.uuid ORDER BY
COUNT(gg.feedbackList) DESC NULLS LAST")
            .setMaxResults(limit - offset)
            .setFirstResult(offset)
            .getResultList();
    }

    public List<Good> getGoodsByRequests(int limit, int offset){
        return entityManager.createQuery(
            "SELECT gg FROM Good gg GROUP BY gg.uuid ORDER BY
COUNT(gg.goodRequestList) DESC")
            .setMaxResults(limit - offset)
            .setFirstResult(offset)
            .getResultList();
    }

    public FullGoodInfoDto getFullInfoForGood(UUID goodUuid, int feedback_limit,
int feedback_offset){
        FullGoodInfoDto result = new FullGoodInfoDto();
        Good g = entityManager
            .createQuery("SELECT gg FROM Good gg WHERE gg.uuid = :idd",
Good.class)
            .setParameter("idd", goodUuid )
            .getSingleResult();
        result.setGood(g);
        List<Feedback> feedbacks = entityManager
            .createQuery("SELECT ff FROM Feedback ff WHERE ff.good.uuid =
:idd", Feedback.class)
            .setMaxResults(feedback_limit - feedback_offset)
```

```

        .setFirstResult(feedback_offset)
        .setParameter("idd", goodUuid )
        .getResultList();
    result.setFeedbacks(feedbacks);
    return result;
}

public Boolean createGood(Good good, UUID catalog){
    try{
        good.setUuid(UUID.randomUUID());
        good.setCatalog(new Catalog(catalog));
        entityManager.persist(good);
        entityManager.flush();
        return true;
    }
    catch(Exception e){
        //result.setMessage(result.getMessage()+" "+e.getMessage());
        logger.log(Level.SEVERE, e.getMessage(),e);
        return false;
    }
}
}

//Листинг класса Catalog
@Entity
@Table(name = "catalog", schema = "public")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Catalog.findAll", query = "SELECT c FROM Catalog c"),
    @NamedQuery(name = "Catalog.findByName", query = "SELECT c FROM Catalog c
WHERE c.name = :name")})
public class Catalog extends AbstractEntity implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "uuid")
    @JsonbTypeAdapter(UuidJsonConverter.class)
    @Convert("uuidConverter")
    private UUID uuid;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 128)
    @Column(name = "name")
    private String name;
    @OneToMany(mappedBy = "catalog")
    private List<Good> goodList;

    public Catalog() {
    }

    public Catalog(UUID uuid) {

```

```

        this.uuid = uuid;
    }
    public Catalog(UUID uuid, String name) {
        this.uuid = uuid;
        this.name = name;
    }
    public UUID getUuid() {
        return uuid;
    }
    public void setUuid(UUID uuid) {
        this.uuid = uuid;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    @XmlTransient
    public List<Good> getGoodList() {
        return goodList;
    }
    public void setGoodList(List<Good> goodList) {
        this.goodList = goodList;
    }
}

//Листинг класса Chat
@Entity
@Table(name = "chat", schema = "public")
@NamedQueries({@NamedQuery(name = "Chat.findAll", query = "SELECT c FROM Chat c")})
public class Chat extends AbstractEntity implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "uuid")
    @JsonbTypeAdapter(UuidJsonConverter.class)
    @Convert("uuidConverter")
    private UUID uuid;
    @JoinColumn(name = "client_uuid", referencedColumnName = "uuid")
    @ManyToOne(optional = false)
    private User client;
    @JoinColumn(name = "manager_uuid", referencedColumnName = "uuid")
    @ManyToOne(optional = false)
    private User manager;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "chat")
    private List<Message> messageList;

    public Chat() {
    }

```

```

public Chat(UUID uuid) {
    this.uuid = uuid;
}

public UUID getUuid() {
    return uuid;
}

public void setUuid(UUID uuid) {
    this.uuid = uuid;
}

public User getClient() {
    return client;
}

public void setClient(User clientUuid) {
    this.client = clientUuid;
}

public User getManager() {
    return manager;
}

public void setManager(User managerUuid) {
    this.manager = managerUuid;
}

@JsonbTransient
@JsonIgnore
@XmlTransient
public List<Message> getMessageList() {
    return messageList;
}

public void setMessageList(List<Message> messageList) {
    this.messageList = messageList;
}
}

```

```

//Листинг класса Feedback
@Entity
@Table(name = "feedback", schema = "public")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Feedback.findAll", query = "SELECT f FROM Feedback f"),
    @NamedQuery(name = "Feedback.findByGrade", query = "SELECT f FROM Feedback f
WHERE f.grade = :grade"),
    @NamedQuery(name = "Feedback.findByFeedback", query = "SELECT f FROM
Feedback f WHERE f.feedback = :feedback"),
    @NamedQuery(name = "Feedback.findByTimestamp", query = "SELECT f FROM
Feedback f WHERE f.timestamp = :timestamp"))
public class Feedback extends AbstractEntity implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "uuid")
    @JsonbTypeAdapter(UuidJsonConverter.class)
    @Convert("uuidConverter")
    private UUID uuid;
    @Basic(optional = false)
    @NotNull
    @Column(name = "grade")
    private short grade;
    @Size(max = 2147483647)
    @Column(name = "feedback")
    private String feedback;
    @Basic(optional = false)
    @NotNull
    @Column(name = "timestamp")
    @Temporal(TemporalType.TIMESTAMP)
    private Date timestamp;
    @JoinColumn(name = "good_uuid", referencedColumnName = "uuid")
    @ManyToOne(optional = false)
    private Good good;
    @JoinColumn(name = "user_uuid", referencedColumnName = "uuid")
    @ManyToOne(optional = false)
    private User user;

    public Feedback() {
    }
    public Feedback(UUID uuid) {
        this.uuid = uuid;
    }
    public Feedback(UUID uuid, short grade, Date timestamp) {
        this.uuid = uuid;
        this.grade = grade;
        this.timestamp = timestamp;
    }
}

```

```

public UUID getUuid() {
    return uuid;
}
public void setUuid(UUID uuid) {
    this.uuid = uuid;
}
public short getGrade() {
    return grade;
}
public void setGrade(short grade) {
    this.grade = grade;
}
public String getFeedback() {
    return feedback;
}
public void setFeedback(String feedback) {
    this.feedback = feedback;
}
public Date getTimestamp() {
    return timestamp;
}
public void setTimestamp(Date timestamp) {
    this.timestamp = timestamp;
}
public String getGood() {
    if(good== null)
        return "null";
    else
        return good.getUuid().toString();
}
public void setGood(Good goodUuid) {
    this.good = goodUuid;
}
public User getUser() {
    User u = new User();
    u.setUuid(UUID.randomUUID());
    u.setName(user.getName());
    return u;
}
public void setUser(User userUuid) {
    this.user = userUuid;
}

```

Приложение Ж

ИСХОДНЫЙ КОД КЛИЕНТСКОГО МОБИЛЬНОГО ПРИЛОЖЕНИЯ
(рекомендуемое)

Листинг Ж.1 – Исходный код клиентского мобильного приложения

```
//Листинг класса MainActivity
class MainActivity : AppCompatActivity(),
    NavigationView.OnNavigationItemSelectedListener {
    lateinit var drawerLayout: DrawerLayout

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        drawerLayout = findViewById<DrawerLayout>(R.id.drawer_layout)

        val navHostFragment =
            supportFragmentManager.findFragmentById(R.id.nav_host_fragment) as
            NavHostFragment
        val navController = navHostFragment.navController
        //        findViewById<NavigationView>(R.id.nav_view)
        //        .setupWithNavController(navController)

        //слушатель нажатия на кнопки навигации

        findViewById<NavigationView>(R.id.nav_view).setNavigationItemSelectedListener(th
            is);

        //Это те фрагменты в которых кнопка будет в виде полосок.
        val appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.nav_login_fragment,
                R.id.nav_profile_fragment,
                R.id.nav_cart_fragment,
                R.id.nav_chats_fragment_client,
                R.id.nav_chats_fragment_manager,
                R.id.nav_myorders_fragment,
                R.id.nav_catalog_fragment,
                R.id.nav_map_fragment
            ),
            drawerLayout
        )
        setupActionBarWithNavController(navController, appBarConfiguration)

        //        val navHostFragment : NavHostFragment =
        supportFragmentManager.findFragmentById(R.id.nav_host_fragment) as
        NavHostFragment
        //        val navController = navHostFragment.navController
        //        val bottomNavigation =
        findViewById<BottomNavigationView>(R.id.nav_view);
        //        bottomNavigation.setupWithNavController(navController)

    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return when (item.itemId) {
            android.R.id.home -> {
                drawerLayout.openDrawer(GravityCompat.START)
                true
            }
        }
    }
}
```

```

        }
        else -> super.onOptionsItemSelected(item)
    }
}

override fun onBackPressed() {
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START)
    } else {
        super.onBackPressed()
    }
}

//Нажатие на итемы в drawer
override fun onNavigationItemSelected(item: MenuItem): Boolean {
    val id =item.itemId;
    drawerLayout.closeDrawer(GravityCompat.START)
    when(id){
        R.id.nav_chats_fragment_client ->{
            val bundle : Bundle = Bundle();
            bundle.putString("mode","CLIENT")
            this.findNavController(R.id.nav_host_fragment)
                .navigate(R.id.nav_chats_fragment, bundle);
        }
        R.id.nav_chats_fragment_manager->{
            val bundle : Bundle = Bundle();
            bundle.putString("mode","MANAGER")
            this.findNavController(R.id.nav_host_fragment)
                .navigate(R.id.nav_chats_fragment, bundle);
        }
        else -> this.findNavController(R.id.nav_host_fragment).navigate(id);
    }
    return true;
}

companion object {
    //Исходя из роли пользователя (админ, менеджер) делает видимыми
    соответств. разделы в меню
    public fun showMenuGroups(currentUser: User, navigationView:
    NavigationView){
        val MANAGER:Short = 2;
        val ADMIN:Short = 3;
        if(currentUser.role != MANAGER && currentUser.role != ADMIN)
            return;

        val nav_Menu: Menu = navigationView.menu
        if(currentUser.role == MANAGER){
            nav_Menu.findItem(R.id.manager_menu_group).isVisible = true;
        }
        if(currentUser.role == ADMIN){
            nav_Menu.findItem(R.id.manager_menu_group).isVisible = true;
            nav_Menu.findItem(R.id.admin_menu_group).isVisible = true;
        }
    }

    public fun hideMenuGroups(navigationView: NavigationView){

```

```

        val nav_Menu: Menu = navigationView.menu
        nav_Menu.findItem(R.id.manager_menu_group).isVisible = false;
        nav_Menu.findItem(R.id.admin_menu_group).isVisible = false;
    }
}

//Листинг класса ChatViewModel
class ChatViewModel(private val chatRepository: ChatRepository): ViewModel() {
    private val _chatsLD = MutableLiveData< ReqResult<List<Chat>> >();
    val chatsLD: LiveData<ReqResult< List<Chat>> >
        get() = _chatsLD;

    private val _messagesLD = MutableLiveData< ReqResult<List<Message>> >();
    val messagesLD : LiveData< ReqResult<List<Message>> >
        get() = _messagesLD;

    //Чат создан
    private val _isChatCreatedSLD = SingleLiveEvent<ReqResult<Boolean>>();
    val isChatCreatedSLD : LiveData<ReqResult<Boolean>>
        get() = _isChatCreatedSLD;

    fun loadChats(userUuid : UUID, asManager:Boolean = false){
        viewModelScope.launch(Dispatchers.IO) {
            val res = chatRepository.getChats(userUuid.toString(), asManager);
            _chatsLD.postValue(res)
        }
    }

    fun loadMessages(chatId: UUID){
        if(chatsLD.value == null || !(chatsLD.value!!.isSuccess) ||
chatsLD.value!!.entity == null )
            return;

        viewModelScope.launch(Dispatchers.IO) {
            chatsLD.value!!.entity!!.find { it.uuid == chatId }.also {
                if(it != null){
                    val res =
chatRepository.getMessageByChat(it.uuid.toString())
                    _messagesLD.postValue(res)
                }
            }
        }
    }

    fun clearMessages(){
        _messagesLD.value = ReqResult(false,"",null)
    }

    fun createChat(clientUuid:UUID, managerUuid:UUID){
        viewModelScope.launch(Dispatchers.IO) {
            val res =
chatRepository.createChat(clientUuid.toString(),managerUuid.toString());
            _isChatCreatedSLD.postValue(res);
        }
    }
}

```

```

fun autoCreateChat(clientUuid: UUID) {
    viewModelScope.launch(Dispatchers.IO) {
        val res = chatRepository.autoCrateChat(clientUuid.toString());
        _isChatCreatedSLD.postValue(res);
    }
}

fun postMessage(userId: String, chatId: String, text:String){
    viewModelScope.launch(Dispatchers.IO) {
        val text_formatted = text.replace("\n", " ")
        val res = chatRepository.postMessage(userId, chatId, text_formatted)
        _messagesLD.postValue(res);
    }
}

}

//Листинг класса GoodViewModel
class GoodViewModel(private val goodRepository: GoodRepository):ViewModel() {
    //Список товаров
    private val _goodsLD = MutableLiveData<ReqResult<List<Good>>>();
    val goodsLD: LiveData<ReqResult<List<Good>>>
        get () = _goodsLD;

    //Выбранный товар
    private val _selectedGoodLD = MutableLiveData<ReqResult<FullGoodInfoDto>>();
    val selectedGoodLDDto : LiveData<ReqResult<FullGoodInfoDto>>
        get() = _selectedGoodLD;

    //Удачно или неудачно (уже в корзине) добавление в корзину
    private val _isAddedToCardSLD = SingleLiveEvent<Boolean>();
    val isAddedToCardSLD : LiveData<Boolean>
        get() = _isAddedToCardSLD;

    //Список товаров в корзине
    private val _cartLD = MutableLiveData< MutableList<Good> >();
    val cartLD : LiveData< MutableList<Good> >
        get() = _cartLD;

    private val _isFeedbackAddedSLD = SingleLiveEvent< ReqResult<Boolean>>()
    val isFeedbackAddedSLD: LiveData< ReqResult<Boolean> >
        get() = _isFeedbackAddedSLD

    var m_GoodId : String? = null

    fun getFullGoodData(){
        viewModelScope.launch(Dispatchers.IO) {
            if(m_GoodId != null) {
                val res: ReqResult<FullGoodInfoDto> =
goodRepository.getGoodInfo(m_GoodId!!);
                _selectedGoodLD.postValue(res)
            }
        }
    }
}

```

```

fun loadGoods() {
    viewModelScope.launch(Dispatchers.IO) {
        val result = goodRepository.getAllGoods();
        _goodsLD.postValue(result);
    }
}

fun addSelectedGoodIntoCart() {
    viewModelScope.launch(Dispatchers.IO) {
        if(selectedGoodLDDto.value != null
            && selectedGoodLDDto.value!!.isSuccess
            && selectedGoodLDDto.value!!.entity != null
        ){
            goodRepository.addGoodToCart(selectedGoodLDDto.value!!.entity!!.good)
            _isAddedToCardSLD.postValue(true);
        }
    }
}

//Загрузит из хранилища данные о товарах в корзине
fun loadGoodsCart() {
    viewModelScope.launch(Dispatchers.IO) {
        val goods = goodRepository.getGoodsInCart();
        if(goods != null){
            _cartLD.postValue(goods!!);
        }
    }
}

fun cleanCart() {
    viewModelScope.launch(Dispatchers.IO) {
        goodRepository.clearCart()
        _cartLD.postValue(mutableListOf())
    }
}

fun getSummPrice():Float{
    if(cartLD.value != null){
        return Good.SummPrice(cartLD.value!!)
    }
    else{
        return 0.0f
    }
}

fun postFeedback(userId: UUID, goodId:UUID, grade:Int, feedback : String?){
    viewModelScope.launch(Dispatchers.IO) {
        val fb :String? = feedback?.replace("\n", " ")
        val res = goodRepository.postFeedback(
            userId.toString(),
            goodId.toString(),
            Feedback(
                UUID.randomUUID(),
                null,
                Date(),
            )
        )
    }
}

```

```

        grade,
        feedback,
        goodId.toString()
    )
    );
    _isFeedbackAddedSLD.postValue(res)
}
}
}
//Листинг класса RequestViewModel
class RequestViewModel(private val requestRepository: RequestRepository):
ViewModel() {

    //Удачно или неудачно заказ добавлен в систему
    private val _isNewOrderAddedSLD = SingleLiveEvent<ReqResult<Boolean>>();
    val isNewOrderAddedSLD : LiveData<ReqResult<Boolean>>
        get() = _isNewOrderAddedSLD;

    private val _myOrdersLD = MutableLiveData<ReqResult<List<MyRequestDto>>>();
    val myOrdersLD: LiveData<ReqResult<List<MyRequestDto>>>
        get () = _myOrdersLD;

    private val _allStatusLD = MutableLiveData<ReqResult<List<Status>>>();
    val allStatusLD: LiveData<ReqResult<List<Status>>>
        get() = _allStatusLD;

    fun postNewOrder(
        current_user: UUID,
        paymentData:String,
        goods:List<Good>
    ){
        viewModelScope.launch(Dispatchers.IO) {
            val res =
requestRepository.postNewOrder(UsersRequestDto(current_user,paymentData,goods))
            _isNewOrderAddedSLD.postValue(res)
        }
    }

    fun getOrders(current_user: UUID, isManagerMode:Boolean){
        viewModelScope.launch(Dispatchers.IO) {
            val res = requestRepository.getMyOrders(current_user,
isManagerMode);
            _myOrdersLD.postValue(res);
        }
    }

    fun loadAllStatusList(){
        viewModelScope.launch(Dispatchers.IO) {
            val res = requestRepository.getAllStatusList()
            _allStatusLD.postValue(res)
        }
    }
}

//Листинг класса ViewModelFactory

```

```

class ViewModelFactory(private val context : Context) :
    ViewModelProvider.Factory {

    override fun <T : ViewModel?> create(modelClass: Class<T>): T {
        val user_repository = UserRepository(
            RetrofitClient.getTypedRetrofitInstance(UserAPI::class.java) , context)
        val chat_repository = ChatRepository(
            RetrofitClient.getTypedRetrofitInstance(ChatAPI::class.java),
            RetrofitClient.getTypedRetrofitInstance(MessageAPI::class.java),
            context
        )
        val good_repository =
            GoodRepository(RetrofitClient.getTypedRetrofitInstance(GoodAPI::class.java),
                context);
        val request_repository =
            RequestRepository(RetrofitClient.getTypedRetrofitInstance(RequestAPI::class.java)
                , context)
        when {
            modelClass.isAssignableFrom(UserViewModel::class.java) -> {
                return UserViewModel(user_repository) as T
            }
            modelClass.isAssignableFrom(ChatViewModel::class.java)->{
                return ChatViewModel(chat_repository) as T;
            }
            modelClass.isAssignableFrom(GoodViewModel::class.java)->{
                return GoodViewModel(good_repository) as T
            }
            modelClass.isAssignableFrom(RequestViewModel::class.java)->{
                return RequestViewModel(request_repository) as T
            }
            else -> {
                throw IllegalArgumentException("Unknown ViewModel")
            }
        }
    }
}

//Листинг класса ChatRepository
class ChatRepository(
    private val chatAPI : ChatAPI,
    private val messageAPI: MessageAPI,
    private val context: Context,
) {
    private val tag: String = "ChatRepository-network";

    suspend fun getChats(userId : String, asManager:Boolean):ReqResult<
        List<Chat> >{
        val call : Call<List<Chat>>
        var res : Response<List<Chat>>? = null;
        try {
            call = if(asManager)
                chatAPI.getChatsByManager(userId)
            else
                chatAPI.getChatsByClient(userId);
            res = call.execute();
        }catch (e:Exception){

```

```

        Log.e(tag, "exception: " + e.message);
        Log.e(tag, "exception: " + e.toString());
        e.printStackTrace()
    }
    finally {
        if(res != null && res.isSuccessful) {
            return if(res.body() == null)
                ReqResult(false,
context.getString(R.string.response_empty_error), null)
            else
                ReqResult(true, "", res.body())
        }else{
            return ReqResult(false, context.getString(R.string.request_err),
null)
        }
    }
}

suspend fun getMessagesByChat(chatId : String):ReqResult< List<Message> >{
    val call : Call<List<Message>>
    var res : Response<List<Message>>? = null;
    try {
        call = messageAPI.getMessagesForChat(chatId);
        res = call.execute();
    }catch (e:Exception){
        Log.e(tag, "exception: " + e.message);
        Log.e(tag, "exception: " + e.toString());
        e.printStackTrace()
    }
    finally {
        if(res != null && res.isSuccessful) {
            return if(res.body() == null)
                ReqResult(false,
context.getString(R.string.response_empty_error), null)
            else
                ReqResult(true, "", res.body())
        }else{
            return ReqResult(false, context.getString(R.string.request_err),
null)
        }
    }
}

suspend fun createChat(clientId: String,
managerId:String):ReqResult<Boolean>{
    val call : Call<Boolean>
    var res : Response<Boolean>? = null;
    try {
        call = chatAPI.createChat(clientId,managerId);
        res = call.execute();
    }catch (e:Exception){
        Log.e(tag, "exception: " + e.message);
        Log.e(tag, "exception: " + e.toString());
        e.printStackTrace()
    }
}

```

```

        finally {
            if(res != null && res.isSuccessful) {
                return if(res.body() == null)
                    ReqResult(false,
context.getString(R.string.response_empty_error), null)
                    else
                        ReqResult(true, "", res.body())
            }else{
                return ReqResult(false, context.getString(R.string.request_err),
null)
            }
        }
    }

suspend fun autoCrateChat(clientId: String):ReqResult<Boolean>{
    val call : Call<Boolean>
    var res : Response<Boolean>? = null;
    try {
        call = chatAPI.autoCreateChat(clientId);
        res = call.execute();
    }catch (e:Exception){
        Log.e(tag, "exception: " + e.message);
        Log.e(tag, "exception: " + e.toString());
        e.printStackTrace()
    }
    finally {
        if(res != null && res.isSuccessful) {
            return if(res.body() == null)
                ReqResult(false,
context.getString(R.string.response_empty_error), null)
                else
                    ReqResult(true, "", res.body())
        }else{
            return ReqResult(false, context.getString(R.string.request_err),
null)
        }
    }
}

suspend fun postMessage(clientId: String, chatId: String, text:
String):ReqResult< List<Message> > {
    val call : Call<List<Message>>
    var res : Response<List<Message>>? = null;
    try {
        call = messageAPI.postRequest(clientId,chatId,text);
        res = call.execute();
    }catch (e:Exception){
        Log.e(tag, "exception: " + e.message);
        Log.e(tag, "exception: " + e.toString());
        e.printStackTrace()
    }
    finally {
        if(res != null && res.isSuccessful) {
            return if(res.body() == null)
                ReqResult(false,
context.getString(R.string.response_empty_error), null)

```

```

        else
            ReqResult(true, "", res.body())
        }else{
            return ReqResult(false, context.getString(R.string.request_err),
null)
        }
    }
}

//Листинг класса GoodRepository
class GoodRepository(private val goodAPI: GoodAPI, private val context :
Context) {
    private val tag: String = "GoodRepository-network";

    private val CART_PREF_KEY : String = "goodscart";
    private val SHARED_PREFERENCES_CART_FILE = "cart"

    suspend fun getAllGoods(): ReqResult<List<Good>> {
        val call : Call<List<Good>>
        var res : Response<List<Good>>? = null;
        try {
            call = goodAPI.getAllGoods();
            res = call.execute();
        }catch (e:Exception){
            Log.e(tag, "exception: " + e.message);
            Log.e(tag, "exception: " + e.toString());
            e.printStackTrace();
        }
        finally {
            if(res != null && res.isSuccessful) {
                return if(res.body() == null)
                    ReqResult(false,
context.getString(R.string.response_empty_error), null)
                else
                    ReqResult(true, "", res.body())
            }else{
                return ReqResult(false, context.getString(R.string.request_err),
null)
            }
        }
    }

    suspend fun getGoodInfo(goodId : String): ReqResult<FullGoodInfoDto> {
        val call : Call<FullGoodInfoDto>
        var res : Response<FullGoodInfoDto>? = null;
        try {
            call = goodAPI.getFullInfoForGood(goodId);
            res = call.execute();
        }catch (e:Exception){
            Log.e(tag, "exception: " + e.message);
            Log.e(tag, "exception: " + e.toString());
            e.printStackTrace();
        }
        finally {
            if(res != null && res.isSuccessful) {

```

```

        return if(res.body() == null)
            ReqResult(false,
context.getString(R.string.response_empty_error), null)
        else
            ReqResult(true, "", res.body())
    }else{
        return ReqResult(false, context.getString(R.string.request_err),
null)
    }
}

suspend fun addGoodToCart(good :Good){
    val sharedPref =
context.getSharedPreferences(SHARED_PREFERENCES_CART_FILE,
Context.MODE_PRIVATE);

    //Уже добавленные товары
    var goods_list : MutableList<Good>? = getGoodsInCart();
    if(goods_list == null){
        goods_list = mutableListOf(good)
    }
    else{
        if(goods_list.find { it.uuid == good.uuid } == null)
            goods_list.add(good)
    }

    //Сохранение
    val json : String = Gson().toJson(goods_list)
    with(sharedPref.edit()){
        putString(CART_PREF_KEY,json)
        commit()
    }
}

suspend fun getGoodsInCart():MutableList<Good>?{
    val sharedPref =
context.getSharedPreferences(SHARED_PREFERENCES_CART_FILE,
Context.MODE_PRIVATE);

    val found = sharedPref.getString(CART_PREF_KEY,null)
    if(sharedPref.contains(CART_PREF_KEY) && found != null){
        val good_list : MutableList<Good> = Gson().fromJson(found,
Array<Good>::class.java).toMutableList();
        return good_list;
    }
    else{
        return null
    }
}

suspend fun clearCart(){
    val sharedPref =
context.getSharedPreferences(SHARED_PREFERENCES_CART_FILE,
Context.MODE_PRIVATE);
    if(sharedPref.contains(CART_PREF_KEY))

```

```

        with(sharedPref.edit()){
            remove(CART_PREF_KEY)
            clear()
            commit()
        }
    }

    suspend fun postFeedback(userId:String, goodId:String, feedback:
Feedback):ReqResult< Boolean > {
    val call : Call<Boolean>
    var res : Response<Boolean>? = null;
    try {
        call = goodAPI.postFeedback(userId, goodId, feedback);
        res = call.execute();
    }catch (e:Exception){
        Log.e(tag, "exception: " + e.message);
        Log.e(tag, "exception: " + e.toString());
        e.printStackTrace()
    }
    finally {
        if(res != null && res.isSuccessful) {
            return if(res.body() == null)
                ReqResult(false,
context.getString(R.string.response_empty_error), null)
            else
                ReqResult(true, "", res.body())
        }else{
            return ReqResult(false, context.getString(R.string.request_err),
null)
        }
    }
}

//Листинг интерфейса ChatAPI
interface ChatAPI {
    @GET("/uldb-back/chat/getChatsByClient")
    fun getChatsByClient(@Header("client") clientId:String):Call<List<Chat>>

    @GET("/uldb-back/chat/getChatsByManager")
    fun getChatsByManager(@Header("manager") managerId:String):Call<List<Chat>>

    @GET("/uldb-back/chat/createChat")
    fun createChat(
        @Header("client") clientId:String,
        @Header("manager") managerId:String):Call<Boolean>

    @GET("/uldb-back/chat/autoCreateChat")
    fun autoCreateChat(@Header("client") clientId:String):Call<Boolean>
}

//Листинг интерфейса GoodAPI
interface GoodAPI {
    @GET("/uldb-back/good/getGoods")
    fun getAllGoods(): Call<List<Good>>

    @GET("/uldb-back/good/getFullInfoForGood/{goodid}")

```

```

        fun getFullInfoForGood(@Path("goodid") goodUuid : String):
        Call<FullGoodInfoDto>

        @POST("/uldbbs-back/feedback/postFeedback")
        fun postFeedback(
            @Header("user") userId:String,
            @Header("good") goodId:String,
            @Body feedback: Feedback
        ): Call<Boolean>
    }
    //Листинг интерфейса MessageAPI
    interface MessageAPI {
        @GET("/uldbbs-back/message/getMessagesForChat")
        fun getMessagesForChat(@Header("chat") chatId:String): Call<List<Message>>

        @POST("/uldbbs-back/message/postMessage")
        fun postRequest(
            @Header("userid") userId:String,
            @Header("chatid") chatId:String,
            @Body text:String
        ):Call<List<Message>>
    }

    //Листинг класса RetrofitClient
    class RetrofitClient {
        companion object {
            public const val IMAGE_DOWNLOAD_URL = "http://192.168.0.83:8081/uldbbs-
            back/file"
            private const val BASE_URL = "http://192.168.0.83:8081/"

            private val client = OkHttpClient.Builder()
                .connectTimeout(100, TimeUnit.SECONDS)
                .readTimeout(100, TimeUnit.SECONDS).build()

            fun <T: Any> getTypedRetrofitInstance(ofClass: Class<T>):T{
                return Retrofit.Builder()
                    .baseUrl(BASE_URL).client(client)

                .addConverterFactory(GsonConverterFactory.create(GsonBuilder().create()))
                    .build()
                    .create(ofClass)
            }
        }
    }
    //Листинг класса Chat
    data class Chat(
        @SerializedName("uuid")
        val uuid : UUID,
        @SerializedName("client")
        val client: User,
        @SerializedName("manager")
        val manager: User
    )
    //Листинг класса Feedback

```

```

data class Feedback (
    @SerializedName("uuid")
    val uuid : UUID,
    @SerializedName("user")
    val user : User?,
    @SerializedName("timestamp")
    @JsonAdapter(DateTypeAdapter::class)
    val timestamp : Date,
    @SerializedName("grade")
    val grade : Int,
    @SerializedName("feedback")
    var feedback : String?,
    @SerializedName("good")
    val good : String
)

//Листинг класса Good
data class Good (
    @SerializedName("uuid")
    val uuid : UUID,
    @SerializedName("catalog")
    val catalog : String,
    @SerializedName("descr")
    val descr: String,
    @SerializedName("name")
    val name: String,
    @SerializedName("price")
    val price: Float,
    @SerializedName("imgPath")
    val imgPath: String?,
    @SerializedName("parametrList")
    val parametrList: List<Parametr>
){
    companion object {
        fun SummPrice(goods:List<Good>):Float{
            var result : Float = 0.0f;
            goods.forEach{
                result += it.price;
            }
            return result
        }
    }
}

//Листинг класса GoodRequest
data class GoodRequest (
    @SerializedName("uuid")
    val uuid : UUID,
    @SerializedName("good")
    val good : Good,
    @SerializedName("request")
    val request : UUID,
)

```

```

//Листинг класса CatalogFragment
class CatalogFragment : Fragment() {
    private val binding by lazy{
        ItemCatalogFragmentBinding.inflate(layoutInflater);
    }
    private lateinit var goodViewModel: GoodViewModel;

    private lateinit var goodListAdapter: GoodListAdapter;

    private val onItemSelect : (Good) -> Unit = {
        val bundle : Bundle = Bundle();
        bundle.putString("goodid",it.uuid.toString())

    findNavController().navigate(R.id.action_nav_catalog_fragment_to_nav_good_fragment, bundle)
    }

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        goodViewModel = ViewModelProviders.of(
            requireActivity(),
            ViewModelFactory(requireActivity().applicationContext)
        ).get(GoodViewModel::class.java)

        binding.grade.paintFlags = binding.grade.paintFlags or
        Paint.UNDERLINE_TEXT_FLAG
        binding.button8.paintFlags = binding.button8.paintFlags or
        Paint.UNDERLINE_TEXT_FLAG

        //Recycler view
        goodListAdapter = GoodListAdapter(this, onItemSelect);
        binding.catalogRv.adapter = goodListAdapter;
        binding.catalogRv.layoutManager = LinearLayoutManager(context)

        //Observer
        goodViewModel.goodsLD.observe(viewLifecycleOwner, Observer {
            if(it.isSuccess){
                goodListAdapter.setGoods(it.entity!!)
            }
            else{
                Toast.makeText(requireContext(), it.message,
                Toast.LENGTH_SHORT).show()
            }
        })

        goodViewModel.loadGoods();
        return binding.root;
    }
}

//Листинг класса ChatFragment
class ChatFragment : Fragment() {
    private val binding by lazy{
        ChatFragmentBinding.inflate(layoutInflater);
    }
    private lateinit var chatViewModel : ChatViewModel;

```

```

private lateinit var messageListAdapter : MessageListAdapter;

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
    chatViewModel = ViewModelProviders.of(
        requireActivity(),
        ViewModelFactory(requireActivity().applicationContext)
    ).get(ChatViewModel::class.java)
    chatViewModel.clearMessages()

    binding.contactorName.text = arguments?.getString("contactor_name");
    val chatid = arguments?.getString("chatid");
    val userid = arguments?.getString("userid");

    if(chatid != null && userid != null){
        //Recycler view
        val lm = LinearLayoutManager(context);
        lm.stackFromEnd = true
        binding.recyclerChat.layoutManager = lm
        messageListAdapter = MessageListAdapter(userid!!);
        binding.recyclerChat.adapter = messageListAdapter;

        //Observer
        chatViewModel.messagesLD.observe(viewLifecycleOwner, Observer {
            if(it.isSuccess && it.entity != null) {
                messageListAdapter.setMessages(it.entity)
                if(messageListAdapter.itemCount > 1)

binding.recyclerChat.smoothScrollToPosition(messageListAdapter.itemCount - 1);
            }
            else
                Toast.makeText(requireActivity().applicationContext,
it.message, Toast.LENGTH_SHORT).show()

        })

        binding.button6.setOnClickListener{
            binding.editTextChat.clearFocus();

chatViewModel.postMessage(userid, chatid, binding.editTextChat.text.toString())
        }

        chatViewModel.loadMessages(UUID.fromString(chatid))
    }
    return binding.root;
}
}

//Листинг класса ManagerMenuFragment
class ManagerMenuFragment : Fragment() {
    private val binding by lazy{
        ManagerMenuFragmentBinding.inflate(layoutInflater);
    }

    private lateinit var requestViewModel: RequestViewModel

```

```

private lateinit var userViewModel: UserViewModel

private lateinit var orderListAdapter: ManagerOrderListAdapter

private val onClickGoToChat: (Button, UUID)->Unit = {
    button, uuid ->
}

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
    requestViewModel = ViewModelProviders.of(
        requireActivity(),
        ViewModelFactory(requireActivity().applicationContext)
    ).get(RequestViewModel::class.java)
    userViewModel = ViewModelProviders.of(
        requireActivity(),
        ViewModelFactory(requireActivity().applicationContext)
    ).get(UserViewModel::class.java)
    binding.hideshow.setOnClickListener(onClickHideShowFinished)

    //Recycler view
    orderListAdapter =
ManagerOrderListAdapter(requireContext(), onClickGoToChat);
    binding.orderslistrv.adapter = orderListAdapter;
    binding.orderslistrv.layoutManager = LinearLayoutManager(context)

    if(userViewModel.userLD.value == null) {
        Toast.makeText(requireContext(), getString(R.string.not_authorized),
Toast.LENGTH_SHORT).show()
        findNavController().navigate(R.id.nav_login_fragment)
    }else{
        requestViewModel.allStatusLD.observe(viewLifecycleOwner, Observer {
            if(it.isSuccess && it.entity != null)
                loadOrders()
            else
                Toast.makeText(requireActivity().applicationContext,
it.message, Toast.LENGTH_SHORT).show()
        })
        requestViewModel.loadAllStatusList();
    }

    return binding.root;
}

private fun loadOrders(){
    requestViewModel.myOrdersLD.observe(viewLifecycleOwner, Observer {
        if(it.isSuccess && it.entity != null)
            orderListAdapter.setOrders(it.entity)
        else
            Toast.makeText(requireActivity().applicationContext, it.message,
Toast.LENGTH_SHORT).show()
        })
    requestViewModel.getOrders(userViewModel.userLD.value!!.uuid, true)
}
}

```

```
//Листинг класса MapFragment
class MapFragment : Fragment(), OnMapReadyCallback {
    private lateinit var binding : MapFragmentBinding;

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
        binding = MapFragmentBinding.inflate(inflater);

        (childFragmentManager.findFragmentById(R.id.map) as SupportMapFragment)
            .getMapAsync(this)

        return binding.root;
    }

    override fun onMapReady(map: GoogleMap) {
        val testMarker = LatLng(53.194846, 45.015790);
        map.addMarker(MarkerOptions()
            .position(testMarker)
            .title("Test")
        )
        map.moveCamera(CameraUpdateFactory.newLatLngZoom(testMarker, 15.0f))
    }
}
```

Приложение И

РЕЗУЛЬТАТЫ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ СИСТЕМЫ
ЭЛЕКТРОННОЙ КОММЕРЦИИ
(рекомендуемое)

Register

Email
test3@gmail.com

Имя
Константин

Пароль

Подтвердите пароль

Дата рождения
18 июня, 1995

Телефон
89045558866

ПОДТВЕРДИТЬ

Уже зарегистрированы?

ВОЙТИ В СИСТЕМУ

Пароли не совпадают

Рисунок И.1 – Тест создания записи пользователя

Your profile

Личный профиль

ВЫЙТИ ИЗ СИСТЕМЫ

Имя: Екатерина
Логин: test2@gmail.com
Телефон: 85554443244

Дата рождения: 12 апр. 1995

Аккаунт: User

Для звонков и уведомлений включен вибросигнал.

Рисунок И.2 – Тест входа в систему

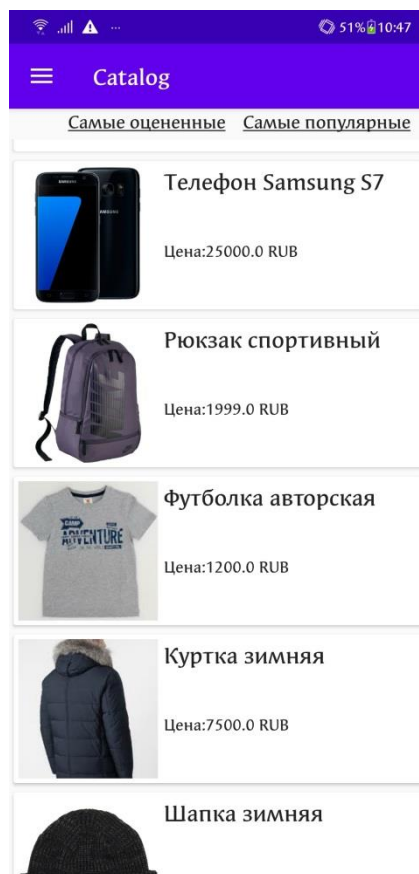


Рисунок И.3 – Тест просмотра каталога

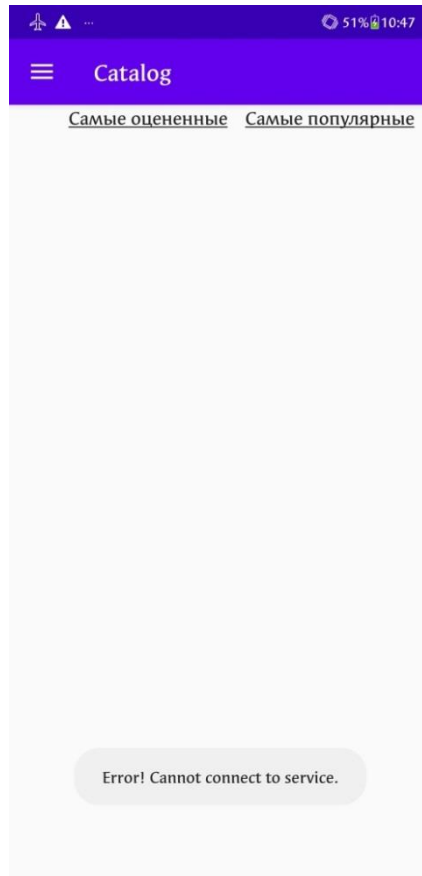


Рисунок И.4 – Тест отсутствия соединения



Рисунок И.5 – Тест просмотра страницы товара с фото

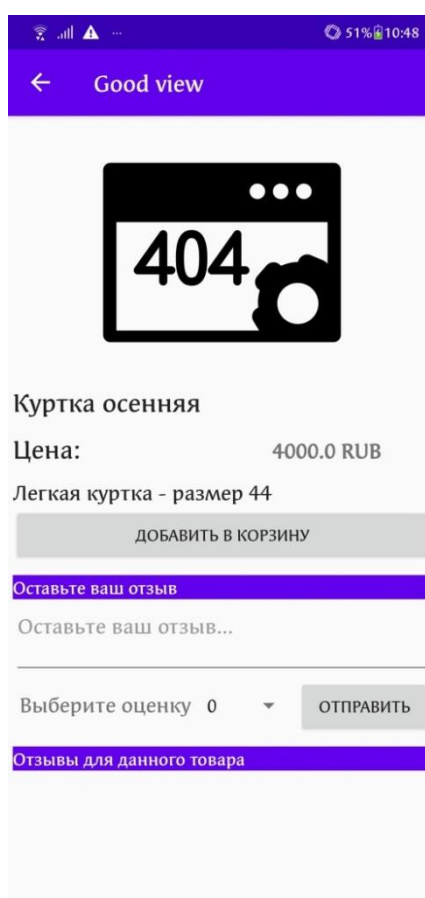


Рисунок И.6 – Тест просмотра страницы товара без фото



Рисунок И.7 – Тест просмотра страницы с картой

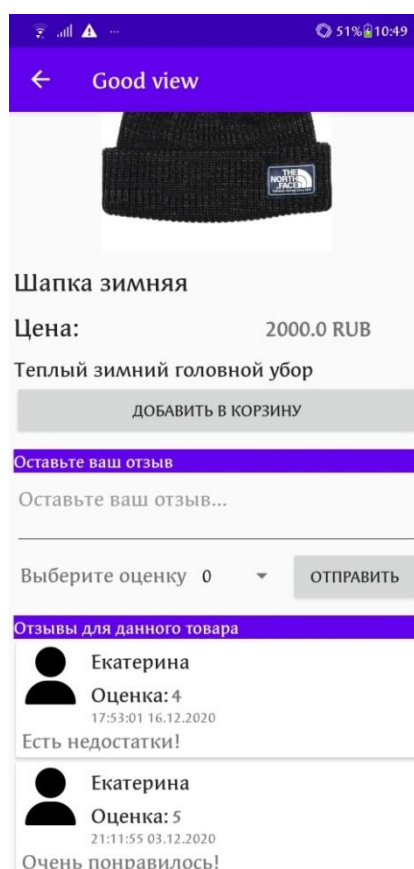


Рисунок И.8 – Тест отзыва на товар

11:01

← Order payment

Всего товаров: 1
Итого цена:

Данные оплаты

Cardholder Name
Alex

Card Number
1111 2222 3334 4444
Card number is invalid

Expiration Date CVV
12 / 21 111

Postal Code

Country Code Mobile Number
8 (904) 853-5522
SMS is required on this number

ПОДТВЕРДИТЬ

Рисунок И.9 – Тест оставления заявки на товар

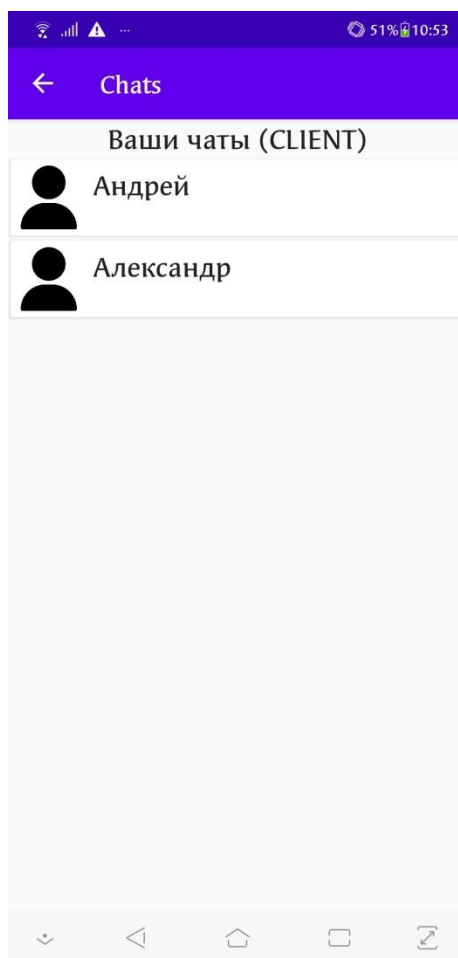


Рисунок И.10 – Тест страницы списка чатов



Рисунок И.11 – Тест чата

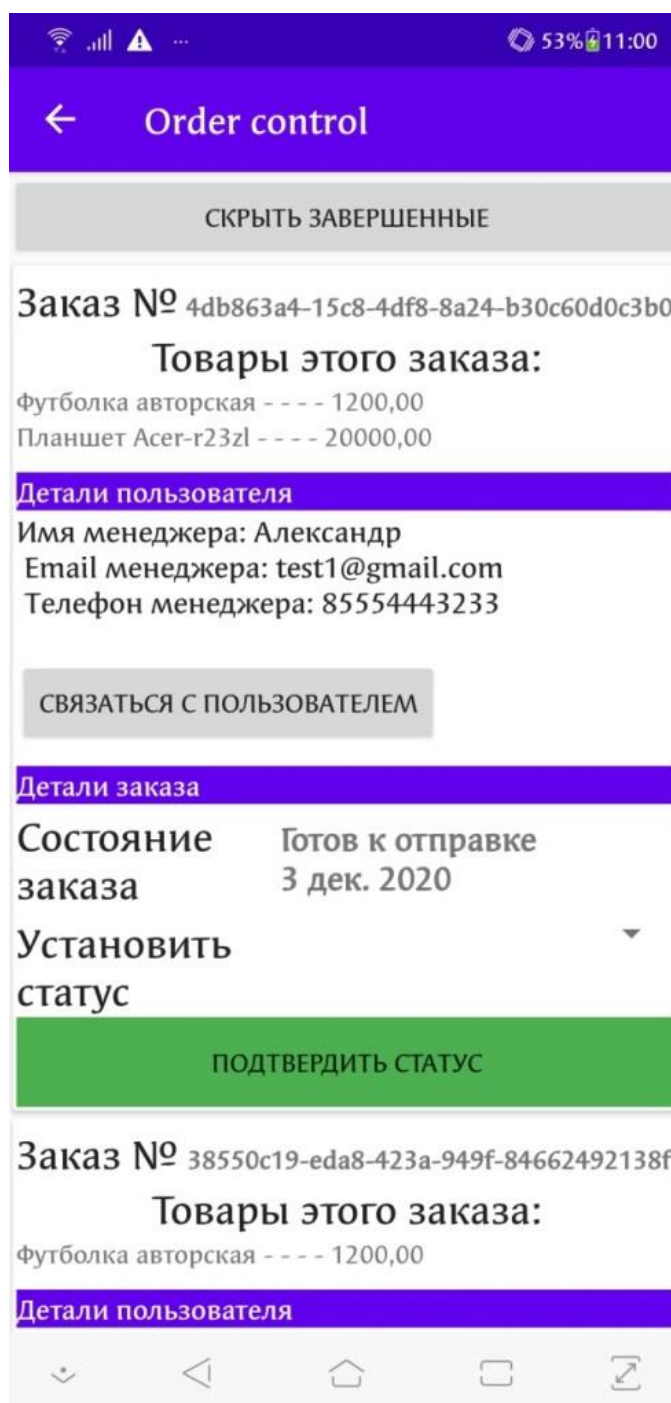


Рисунок И.12 – Тест выполнения заявок

Приложение К
ДИПЛОМЫ ПОБЕДИТЕЛЯ МЕЖДУНАРОДНЫХ КОНКУРСОВ
НАУЧНЫХ РАБОТ
(рекомендуемое)



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



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



Рисунок К.3 – Диплом победителя I степени XI Международного научно-исследовательского конкурса «Молодой ученый»

Приложение Л
СВИДЕТЕЛЬСТВО О ГОСУДАРСТВЕННОЙ РЕГИСТРАЦИИ
ПРОГРАММЫ ДЛЯ ЭВМ
(рекомендуемое)

РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023619790

**Программа аутентификации пользователей в системе
электронной коммерции**

Правообладатель: *Тепин Константин Сергеевич (RU)*

Авторы: *Тепин Константин Сергеевич (RU), Князев Виктор
Николаевич (RU)*



Заявка № 2023615266

Дата поступления 20 марта 2023 г.

Дата государственной регистрации

в Реестре программ для ЭВМ 16 мая 2023 г.

*Руководитель Федеральной службы
по интеллектуальной собственности*

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 68e80077b14e40903a94e0bd24145d5c7
Владелец *Зубов Юрий Сергеевич*
Действителен с 20.03.2022 по 26.05.2023

Ю.С. Зубов