

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Направление подготовки 09.04.01 «Информатика и вычислительная техника»

Магистерская программа Прикладной искусственный интеллект

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

ИССЛЕДОВАНИЕ И РАЗРАБОТКА

АВТОМАТИЗИРОВАННЫХ СИСТЕМ ОБСЛУЖИВАНИЯ

АВИАПАССАЖИРОВ С ПРИМЕНЕНИЕМ ЭЛЕМЕНТОВ

ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Студент Староверов Илья Михайлович
(подпись, дата) (ФИО полностью)

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

Нормоконтролер Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент Петерский С.В.
(подпись, дата) (фамилия, инициалы)

ПензГТУ
(должность, место работы)

Работа допущена к защите (протокол заседания кафедры от 28.05.2025 № 16)

Заведующий кафедрой Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 05.06.2025 № 2)

Секретарь ГЭК Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза 2025

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

Факультет
Вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 29 » сентября 2023 г

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Староверов Илья Михайлович гр. 23ВВИм1 факультета ВТ

2. Тема работы Исследование и разработка автоматизированных систем обслуживания
авиапассажиров с применением элементов искусственного интеллекта

Тема утверждена приказом ПГУ № 986/о от « 29 » сентября 2023

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

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

1) *Исследовать и разработать автоматизированную систему обслуживания авиа-*
пассажиров, включающую в себя рекомендательную систему выбора авианаправления.

2) *Среда выполнения приложения: операционная система Android 9 и выше*
Среда выполнения сервера: операционная система Windows 10 и выше
Среда выполнения рекомендательной системы: ОС Windows 10 и выше
Языковые средства разработки: C#, Python, UML, SQL, Linq
Инструментальные средства разработки: Microsoft SQL Server 2019, Microsoft
Visual Studio 2019, Spyder, ArgoUML, Visual Paradigm, Microsoft Project 2007
Технологии разработки: RUP, ООП, MVC, MVVM, Entity Framework

3) *Основные функции мобильного приложения:*

а) осуществление бронирования и покупки авиабилета;

б) вывод рекомендованных авианаправлений;

в) осуществление выкупа и снятия брони;

г) осуществление выкупа и снятия множества броней;

д) осуществление возврата билета;

е) осуществление возврата множества билетов;

ж) изменение данных пассажира или покупателя.

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

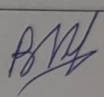
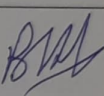
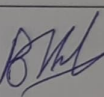
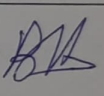
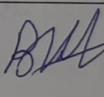
2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области постановка задачи	10%	29.09.2023 – 25.11.2023	
2	Проектирование	20%	26.11.2023 – 27.03.2024	
3	Реализация	30%	28.03.2024 – 12.10.2024	
4	Контроль качества программного обеспечения	20%	13.10.2024 – 20.02.2025	
5	Оформление пояснительной записки	20%	21.02.2025 – 25.05.2025	

Дата выдачи задания "29" сентября 2023

Руководитель магистерской диссертации

(подпись, дата)

Задание к исполнению принял студент

(подпись, дата)

Работу к защите допустить

Декан факультета

(подпись, дата)

Л.Р. Фионова

РЕФЕРАТ

Пояснительная записка содержит 215 листов, 69 рисунков, 41 таблица, 73 источника, 11 приложений.

АВИАБИЛЕТЫ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ANDROID, C#, PYTHON, RUP, UML, MS SQL SERVER, РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА, ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ, ОНТОЛОГИЧЕСКОЕ МОДЕЛИРОВАНИЕ, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Объектом разработки является автоматизированная система обслуживания авиапассажиров в виде мобильного приложения, рекомендательной системы и средств моделирования.

Цель работы – повышение эффективности процессов бронирования и продажи авиабилетов за счёт создания интеллектуальной системы, включающей рекомендательные механизмы и средства моделирования.

Разработка серверной и мобильной частей производилась на языке C# в среде Microsoft Visual Studio 2019 с использованием СУБД MS SQL Server 2019 на основе методологии RUP. Рекомендательная система разработана на языке Python в среде Spyder.

Приложение предназначено для ОС Android 9.0 и выше, серверная часть и рекомендательная система работают под управлением Windows 10 и выше.

Применены методы онтологического и дискретно-событийного имитационного моделирования, а также алгоритмы машинного обучения (SVD, PageRank, случайный лес) для реализации гибридной рекомендательной системы.

					ПГУ 1.09.04.01.12.001 ПЗ				
Изм.	Лист	№ докум.	Подп.	Дата	Исследование и разработка автоматизированных систем обслуживания авиапассажиров с применением элементов искусственного интеллекта. Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Староверов И.М.	<i>Ст</i>	28.05.25				215	
Пров.		Князев В.Н.	<i>ВН</i>	30.05.25			2		
Реценз.		Петерский	<i>Пет</i>	30.05.25		ФВТ., каф. ВТ, гр. 23ВВИм1			
Н. контр.		Бычков А.С.	<i>АБ</i>	30.05.25					
Утв.		Митрохин М.А.	<i>ММ</i>	30.05.25					

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ.....	5
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	6
ВВЕДЕНИЕ.....	7
1 Анализ предметной области исследования	17
1.1 Понятие и классификация средств обслуживания авиапассажиров	17
1.2 Анализ программных продуктов аналогичного назначения	22
1.2.1 «Аэрофлот».....	22
1.2.2 «Aviasales»	23
1.2.3 «Яндекс.авиабилеты»	25
1.2.4 Сравнение аналогов	26
1.3 Применение искусственного интеллекта в сфере авиаперевозок.....	27
1.4 Выбор технологий и инструментальных средств разработки.....	30
1.5 Анализ целевой аудитории	32
1.6 Структурное и календарное планирование для IT-проекта разработки системы обслуживания авиапассажиров.....	35
2 Разработка автоматизированной системы обслуживания авиапассажиров и гибридной рекомендательной системы	46
2.1 Онтологическая модель предметной области.....	46
2.2 Имитационное моделирование объекта исследования	47
2.3 Проверка адекватности имитационных моделей.....	56
2.4 Анализ требований.....	60
2.4.1 Анализ функциональных требований.....	60
2.4.2 Анализ нефункциональных требований.....	64
2.5 Проектирование интерфейса пользователя.....	65
2.6 Проектирование базы данных.....	76
2.6.1 Концептуальное проектирование базы данных.....	76
2.6.2 Логическое проектирование базы данных	77

2.7 Разработка гибридной рекомендательной системы для автоматизированных систем обслуживания авиапассажиров	79
3 Реализация автоматизированной системы обслуживания авиапассажиров	86
3.1 Реализация программного обеспечения	86
3.1.1 Реализация базы данных	86
3.1.2 Реализация серверной части приложения	91
3.1.3 Реализация мобильного приложения	93
3.1.4 Тестирование программного обеспечения	96
3.2 Разработка и тестирование программы, реализующей гибридную рекомендательную систему для авиаперелетов	104
ЗАКЛЮЧЕНИЕ	108
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	111
ПРИЛОЖЕНИЕ А Исходный код имитационных моделей	119
ПРИЛОЖЕНИЕ Б Исходный код скриптов базы данных	123
ПРИЛОЖЕНИЕ В Исходный код серверной части	135
ПРИЛОЖЕНИЕ Г Исходный код клиентской части	145
ПРИЛОЖЕНИЕ Д Результаты тестирования серверной части	167
ПРИЛОЖЕНИЕ Е Исходный код функций тестирования	177
ПРИЛОЖЕНИЕ Ж Результаты тестирования клиентской части	180
ПРИЛОЖЕНИЕ И Исходный код программы, реализующей гибридную рекомендательную систему для авиаперелетов	189
ПРИЛОЖЕНИЕ К Дипломы победителя международных конкурсов научных работ	195
ПРИЛОЖЕНИЕ Л Свидетельство о государственной регистрации программы для ЭВМ	197
ПРИЛОЖЕНИЕ М Презентация	199

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей работе применяются следующие термины с соответствующими определениями

Средства обслуживания авиапассажиров – оборудование и инфраструктура, предназначенные для обеспечения комфорта, безопасности и удобства пассажиров до и во время авиаперелетов

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

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяются следующие сокращения и обозначения

БД – база данных

ВАК – высшая аттестационная комиссия

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

API – Application Programming Interface

RUP – Rational Unified Process

UML – Unified Modeling Language

ВВЕДЕНИЕ

В настоящее время информационные ресурсы общества являются одним из самых главных факторов его развития. Они задают направления в различных сферах общества и оказывают огромное влияние на массовое сознание людей. В начале XXI века происходило развитие цифровых технологий. Ученые были уверены, что развитие технологий в будущем повлияет на все отношения между людьми, на экономику, на общество, на весь мир в целом. Так оно и произошло. Сегодня цифровая среда проникла во все сферы, как в область профессиональной деятельности, так и в повседневные практики.

Цифровизацию можно определить, как внедрение информационных технологий во все сферы деятельности в системах разного уровня, в том числе и в экономику [1].

Процесс внедрения информационных технологий во все сферы деятельности активно происходит и в России. Данный процесс поддерживается, в том числе, и на государственном уровне. Об этом может говорить программа «Цифровая экономика Российской Федерации», которая была утверждена 9 мая 2017 года правительством 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» [2].

Также 9 мая 2017 года был подписан Указ Президента Российской Федерации «О стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы». В нем были поставлены основные цели и задачи политики Российской Федерации в сфере информационных технологий, а также объявлен курс на формирование национальной цифровой экономики во всех сферах, включая такую сферу, как авиаперевозки [3].

Одной из важнейших особенностей развития информационных технологий в нашей стране и во всем мире является повсеместное распространение мобильных устройств с доступом к Интернету. При этом мобильный Интернет постепенно оттесняет на второй план традиционный доступ к сети с использованием стационарных компьютеров по причине своего существенно большего удобства. Так, например, как отмечено в государственной программе «Цифровая экономика», в 2016 году в нашей стране «на 100 человек приходилось 159,95 мобильного телефона и из 100 человек 71,29 человека использовали мобильный доступ к сети Интернет» [2]. В настоящее время эта тенденция лишь усиливается.

Повсеместное развитие искусственного интеллекта не остается без внимания и на государственном уровне. Так, в Указе Президента Российской Федерации «О развитии искусственного интеллекта в Российской Федерации» определяются «цели и основные задачи развития искусственного интеллекта в Российской Федерации, а также меры, направленные на его использование в целях обеспечения национальных интересов и реализации стратегических национальных приоритетов», а в Распоряжении Правительства Российской Федерации № 2715-р, утвержденном 5 октября 2023 года, устанавливается выделение 821 млн. рублей «на осуществление грантовой поддержки малых предприятий по разработке, применению и коммерциализации продуктов, сервисов и (или) решений с использованием технологий искусственного интеллекта» [4, 5].

Сегодня авиаперевозки как в мире, так и в России, являются одним из самых быстро развивающихся способов передвижения. Люди пользуются ими для быстрого перемещения на большие расстояния с разными целями. Кто-то перемещается с целью туризма, кто-то отправляется в командировку, кто-то навещает своих родственников. В связи с этим авиаперелеты для многих людей являются очень важной частью их жизни.

Информационные технологии распространяются, в том числе, и на отрасль воздушных перевозок. В частности, об этом может говорить

Постановление Правительства РФ «Об утверждении требований к автоматизированной информационной системе оформления воздушных перевозок...» [6]. В данном документе определяется, что в автоматизированной информационной системе оформления воздушных перевозок должно быть использовано только российское программное обеспечение.

Исходя из всего вышеперечисленного следует, что тема диссертации «Исследование и разработка автоматизированных систем обслуживания авиапассажиров с применением элементов искусственного интеллекта» является актуальной и перспективной.

Предметная область автоматизированных систем обслуживания авиапассажиров предполагает работу с большим объемом данных о предпочтениях и поведении пользователей, что подчеркивает актуальность разработки рекомендательной системы выбора авианаправления, способной более точно и эффективно учитывать индивидуальные запросы пассажиров по сравнению с существующими решениями.

Также представляется целесообразным актуализировать существующую классификацию средств обслуживания авиапассажиров путем ее расширения. Объединение уже имеющихся классификаций с добавлением новых, собственных критериев, позволяет более эффективно анализировать существующие системы и сравнивать их с разрабатываемой системой обслуживания.

Анализ существующих систем выявил, что в сфере автоматизированных систем обслуживания авиапассажиров недостаточно используются методы онтологического и имитационного моделирования. Расширенная онтологическая модель, включающая более широкий набор классов по сравнению с существующими решениями, способствует повышению эффективности проектирования исследуемой системы. Применение дискретно-событийного имитационного моделирования позволяет оптимизировать параметры функционирования

автоматизированных систем с учетом взаимодействия программных компонентов и человеческого фактора.

Цель магистерской диссертационной работы состоит в повышении эффективности процессов бронирования и продажи авиабилетов в аэропорту за счет разработки программного приложения, ориентированного как на сотрудников кассы, так и на конечных пользователей.

В рамках работы поставлены следующие задачи:

- проведение критического анализа предметной области,
- разработка расширенной классификации средств обслуживания авиапассажиров путем внедрения дополнительных классификационных признаков,
- создание онтологической модели предметной области, отличающейся от существующих расширенным набором сущностей и классов предметной области,
- создание дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования реализованной автоматизированной системы обслуживания авиапассажиров,
- разработка сетевого графика проекта по созданию программной автоматизированной системы обслуживания авиапассажиров для последующей оценки сроков и бюджета данного проекта,
- разработка рекомендательной системы в автоматизированной системе обслуживания авиапассажиров, реализация и тестирование разработанной системы,
- регистрация программной реализации разработанной рекомендательной системы и подготовка к внедрению разработанной и реализованной системы обслуживания авиапассажиров.

Объектом исследования является разрабатываемая автоматизированная система обслуживания авиапассажиров.

Предметом исследования являются модели и подходы к разработке рекомендательных систем, применяемые в процессе проектирования и применения автоматизированных систем обслуживания авиапассажиров.

В работе применяются методы онтологического, имитационного и визуального моделирования, математический аппарат сетевого планирования и управления, методы теории вероятностей и математической статистики, методы теории массового обслуживания, методы искусственного интеллекта, методы теории графов, современные методы анализа качества программного кода, методы проектирования программных средств.

Научная новизна работы заключается в следующем:

- предложен подход к разработке оригинальной гибридной рекомендательной системы, который отличается от традиционных подходов использованием сразу нескольких методов: коллаборативной фильтрации на основе сингулярного разложения матрицы (SVD), графовых моделей с алгоритмом PageRank и алгоритмов машинного обучения, в частности модели случайного леса. Система обрабатывает разнообразные данные о пользователях, включая историю просмотров городов, покупки и бронирования билетов, интересы, сезонность и популярность направлений. Благодаря комбинированному использованию этих методов и многоуровневому анализу данных, достигается более высокая точность и лучшая персонализация рекомендаций в автоматизированной системе обслуживания авиапассажиров,

- предложены две дискретно-событийные имитационные модели для оптимизации и определения оптимального режима работы автоматизированной системы обслуживания авиапассажиров. В отличие от известных имитационных моделей, разработанные модели обладают комплексным характером и позволяют учитывать как технические характеристики аппаратно-программных средств, таких как серверы обработки запросов, так и человеческий фактор, связанный с деятельностью кассиров и предпочтениями пользователей. Такой подход обеспечивает более

точное воспроизведение реальных условий функционирования системы и способствует принятию обоснованных решений по ее оптимизации,

- предложена расширенная онтологическая модель для автоматизированной системы обслуживания авиапассажиров. Данная модель отличается от известных расширенным набором сущностей и классов предметной области. Она используется для эффективного проектирования объекта исследования,

- предложена модифицированная классификация средств обслуживания авиапассажиров, отличающаяся от известных тем, что предлагается более полный расширенный перечень классификационных признаков и средств.

На основе материалов диссертационной работы было получено Свидетельство о государственной регистрации программы для ЭВМ в Федеральном институте промышленной собственности Российской Федерации. Проведены мероприятия по подготовке к внедрению результатов исследования. С использованием метаматематического аппарата сетевого планирования и управления, а также программного обеспечения Microsoft Project выполнено структурное и календарное планирование программного проекта по разработке системы обслуживания авиапассажиров. В результате были определены сроки реализации и бюджет проекта, что подтвердило его экономическую эффективность и практическую значимость.

Основные положения, выносимые на защиту:

- предложенный подход к разработке оригинальной гибридной рекомендательной системы, которая отличается от традиционных подходов использованием сразу нескольких методов: коллаборативной фильтрации на основе сингулярного разложения матрицы (SVD), графовых моделей с алгоритмом PageRank и алгоритмов машинного обучения, в частности модели случайного леса. Система обрабатывает разнообразные данные о пользователях, включая историю просмотров городов, покупки и бронирования билетов, интересы, сезонность и популярность направлений.

Благодаря комбинированному использованию этих методов и многоуровневому анализу данных, достигается более высокая точность и лучшая персонализация рекомендаций в автоматизированной системе обслуживания авиапассажиров,

- две дискретно-событийные имитационные модели для оптимизации и определения оптимального режима работы автоматизированной системы обслуживания авиапассажиров. В отличие от известных имитационных моделей, разработанные модели обладают комплексным характером и позволяют учитывать как технические характеристики аппаратно-программных средств, таких как серверы обработки запросов, так и человеческий фактор, связанный с деятельностью кассиров и предпочтениями пользователей. Такой подход обеспечивает более точное воспроизведение реальных условий функционирования системы и способствует принятию обоснованных решений по ее оптимизации,

- расширенная онтологическая модель для автоматизированной системы обслуживания авиапассажиров. Данная модель отличается от известных расширенным набором сущностей и классов предметной области. Она используется для эффективного проектирования объекта исследования,

- предложенная модифицированная классификация средств обслуживания авиапассажиров, отличающаяся от известных тем, что предлагается более полный расширенный перечень классификационных признаков и средств.

Основные результаты работы были доложены и обсуждены на следующих конференциях:

- Всероссийская научно-практическая конференция обучающихся и научно-педагогических работников «Актуальные проблемы науки и образования». – Пенза, октябрь 2023,

- XVIII Международная научно-практическая конференция «Современные тенденции развития науки и мирового сообщества в эпоху цифровизации». – Москва, октябрь 2023,

– 9-я Всероссийская национальная научно-практическая конференция «Проблемы развития современного общества». – Пятигорск, январь 2024,

– VII Всероссийская научно-практическая конференция «Современные тенденции развития фундаментальных и прикладных наук». – Брянск, январь 2024,

– XI Всероссийская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза, март 2024,

– Международная научно-практическая школа-конференция «Актуальные проблемы науки и образования». – Пенза, октябрь 2024,

– XXI Международная научно-техническая конференция «Новые информационные технологии и системы». – Пенза, ноябрь 2024,

– XII Всероссийская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза, март 2025.

По теме диссертационного исследования опубликовано 12 печатных работ, две из которых опубликованы в научном журнале, входящем в перечень ВАК:

– Староверов И.М. Разработка программных средств поддержки процессов резервирования и продажи авиабилетов // Достижения вузовской науки 2023: сборник статей международного научно-исследовательского конкурса. – 2023. – С. 35 – 41,

– Староверов И.М. Вопросы разработки автоматизированных систем обслуживания авиапассажиров // Сборник материалов XVIII международной научно-практической конференции «Современные тенденции развития науки и мирового сообщества в эпоху цифровизации». – 2023. – С. 69 – 74,

– Староверов И.М. Проектирование автоматизированных систем обслуживания авиапассажиров с применением современных информационных технологий // Наш выбор – наука! : Сборник статей IV Международного научно-исследовательского конкурса. – 2024. – С. 20 – 28,

– Староверов И.М. Разработка онтологической модели для автоматизированной системы обслуживания авиапассажиров // Проблемы развития современного общества: Сборник научных статей 9-й Всероссийской национальной научно-практической конференции. – 2024. – Т. 3. – С. 125 – 129,

– Староверов И.М., Князев В.Н. Современные подходы к классификации средств обслуживания авиапассажиров // Современные тенденции развития фундаментальных и прикладных наук: Материалы VII Всероссийской научно-практической конференции. – 2024. – С. 370 – 374,

– Староверов И.М., Князев В.Н. Вопросы разработки онтологической модели для системы автоматизированного обслуживания пассажиров воздушного транспорта // Информационные технологии в науке и образовании. Проблемы и перспективы: Сборник материалов XI Всероссийской научно-практической конференции. – 2024. – С. 167 – 169,

– Староверов И.М., Князев В.Н. Вопросы разработки рекомендательной системы для автоматизированной системы обслуживания авиапассажиров // Информационные технологии в науке и образовании. Проблемы и перспективы: Сборник материалов XI Всероссийской научно-практической конференции. – 2024. – С. 169 – 172,

– Староверов И.М., Князев В.Н. Комплексный подход к созданию рекомендательной системы для авиаперелетов // Вестник Пензенского государственного университета. – 2025. – № 1. – С 125 – 128,

– Староверов И.М., Князев В.Н. Имитационное моделирование автоматизированных систем обслуживания авиапассажиров // Международный научно-исследовательский журнал. – 2025. – №3 (153). – URL: <https://research-journal.org/archive/3-153-2025-march/10.60797/IRJ.2025.153.18> (дата обращения 2025-03-22). (Перечень ВАК),

– Староверов И.М., Князев В.Н. Программа персонализированных рекомендаций городов для авиапассажиров // Свидетельство о

государственной регистрации программы для ЭВМ № 2025617775. Заявка № 2025611693, дата поступления 01.02.2025 г., дата государственной регистрации в Реестре программ для ЭВМ 28.03.2025 г. – М., Федеральный институт промышленной собственности, 2025. – URL: <https://www.elibrary.ru/item.asp?id=80654096> (дата обращения 2025-05-19),

– Староверов И.М., Князев В.Н. Систематизация и классификация средств обслуживания авиапассажиров в контексте современных технологий // Информационные технологии в науке и образовании. Проблемы и перспективы: Сборник материалов XII Всероссийской научно-практической конференции. – 2025. (В печати),

– Староверов И.М., Князев В.Н. Разработка гибридной рекомендательной системы для авиапассажиров // Международный научно-исследовательский журнал. – 2025. – №6 (156). (Перечень ВАК). (В печати).

По теме магистерской диссертации получены следующие документы: диплом победителя Международного научно-исследовательского конкурса «Достижения вузовской науки 2023» (2023 г.); диплом I степени Международного научно-исследовательского конкурса «Наш выбор – наука!» (2024 г.).

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

1 Анализ предметной области исследования

1.1 Понятие и классификация средств обслуживания авиапассажиров

Единого определения средств обслуживания авиапассажиров не существует. Но есть несколько источников, на основе которых можно составить такое определение [7, 8]. Средства обслуживания авиапассажиров – это оборудование и инфраструктура, предназначенные для обеспечения комфорта, безопасности и удобства пассажиров до и во время авиаперелетов. В это понятие входят такие элементы, как средства продажи и бронирования билетов, терминалы и здания аэропортов, включающие зоны ожидания, регистрацию, паспортный контроль, обслуживание багажа, а также другие объекты, в которых обеспечивается безопасность и комфортные условия для авиапассажиров.

Перечислим основные классификационные признаки, по которым можно характеризовать средства механизации и автоматизации технологических процессов обслуживания авиапассажиров [9].

По месту расположения средства обслуживания авиапассажиров подразделяются на следующие виды:

- внутривокзальные,
- перонные,
- внутрисамолетные,
- офисы бронирования и продаж.

По объекту обслуживания средства обслуживания авиапассажиров делятся на следующие виды:

- обслуживание пассажиров,
- обслуживание багажа.

По принадлежности средства обслуживания авиапассажиров делятся на следующие виды:

- аэропортовая собственность,

- собственность авиакомпаний,
- собственность хэндлинговых и других сервисных компаний,
- собственность государственных служб.

По мобильности средства обслуживания авиапассажиров подразделяются на следующие виды:

- подвижные (мобильные),
- стационарные.

По этапу технологии обслуживания средства обслуживания авиапассажиров подразделяются на следующие виды:

- средства организации и оптимизации обслуживания,
- средства бронирования и продажи,
- средства контроля за авиабезопасностью,
- средства таможенного и паспортного контролей,
- средства регистрации,
- средства обработки багажа,
- средства посадки в воздушное судно,
- средства обслуживания на борту,
- средства дополнительных услуг.

В рамках выполнения данной работы на основе приведенных выше классификаций была создана общая классификация средств обслуживания авиапассажиров, представленная на рисунке 1.

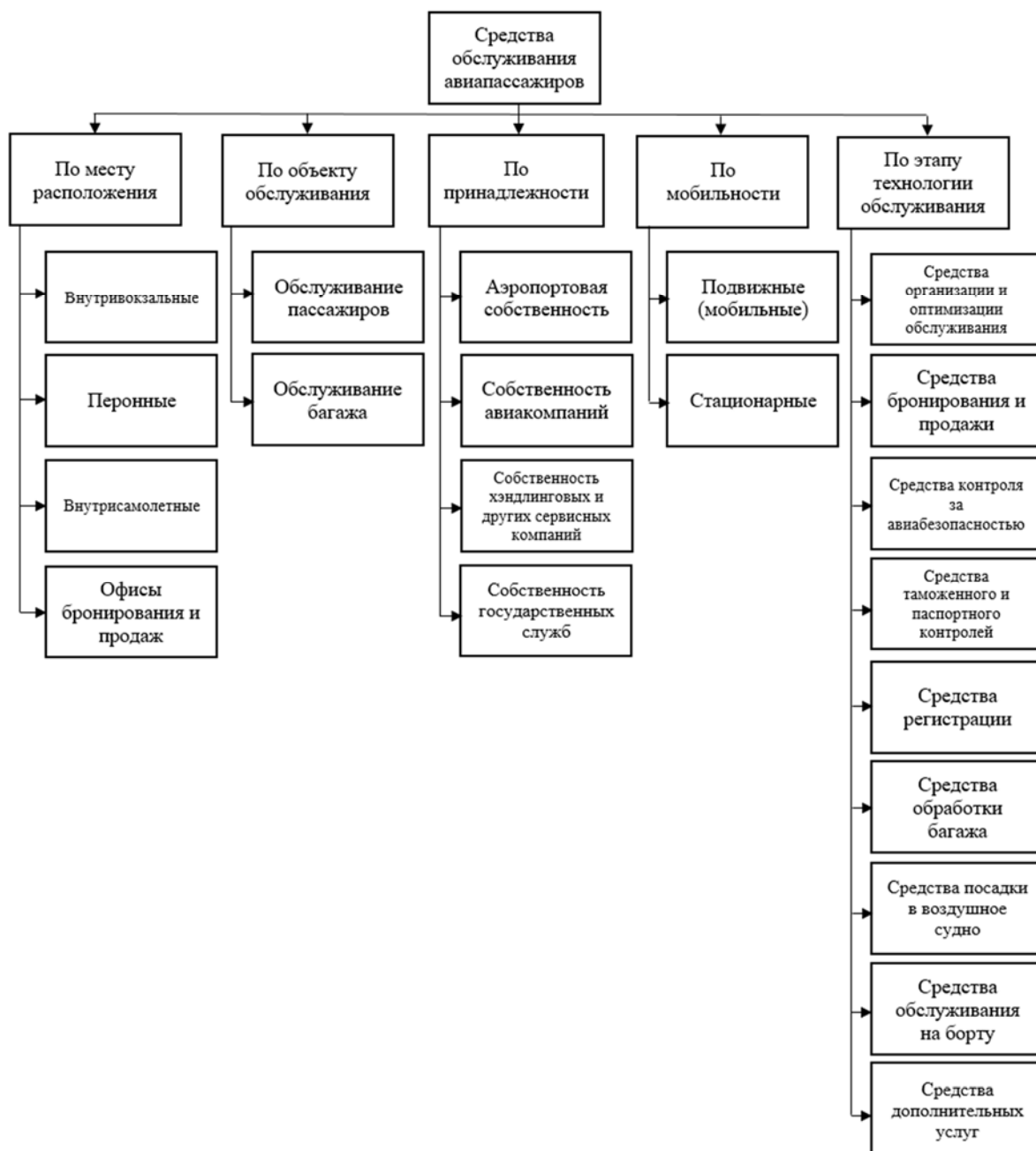


Рисунок 1 – Исходная общая классификация средств обслуживания авиапассажиров

По итогам выполнения анализа предметной области по теме диссертации было решено, что целесообразно расширение существующей классификации средств обслуживания авиапассажиров путем добавления новых классификационных признаков.

В предложенной расширенной классификации были добавлены еще пять классификационных признаков: требуемый пользовательский опыт,

предоставление аналитических данных, степень безопасности данных, степень автоматизации, применение искусственного интеллекта [10].

По требуемому пользовательскому опыту средства обслуживания авиапассажиров разделяются на два класса:

- требуется пользовательский опыт,
- не требуется пользовательский опыт.

По предоставлению аналитических данных средства обслуживания авиапассажиров разделяются на два класса:

- с предоставлением аналитических данных,
- без предоставления аналитических данных.

По степени безопасности данных средства обслуживания авиапассажиров разделяются на три класса:

- с высокой степенью безопасности,
- со средней степенью безопасности,
- с низкой степенью безопасности.

По степени автоматизации средства обслуживания авиапассажиров разделяются на три класса:

- автоматические,
- автоматизированные,
- ручные.

По применению искусственного интеллекта средства обслуживания авиапассажиров разделяются на два класса:

- с применением искусственного интеллекта,
- без применения искусственного интеллекта.

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

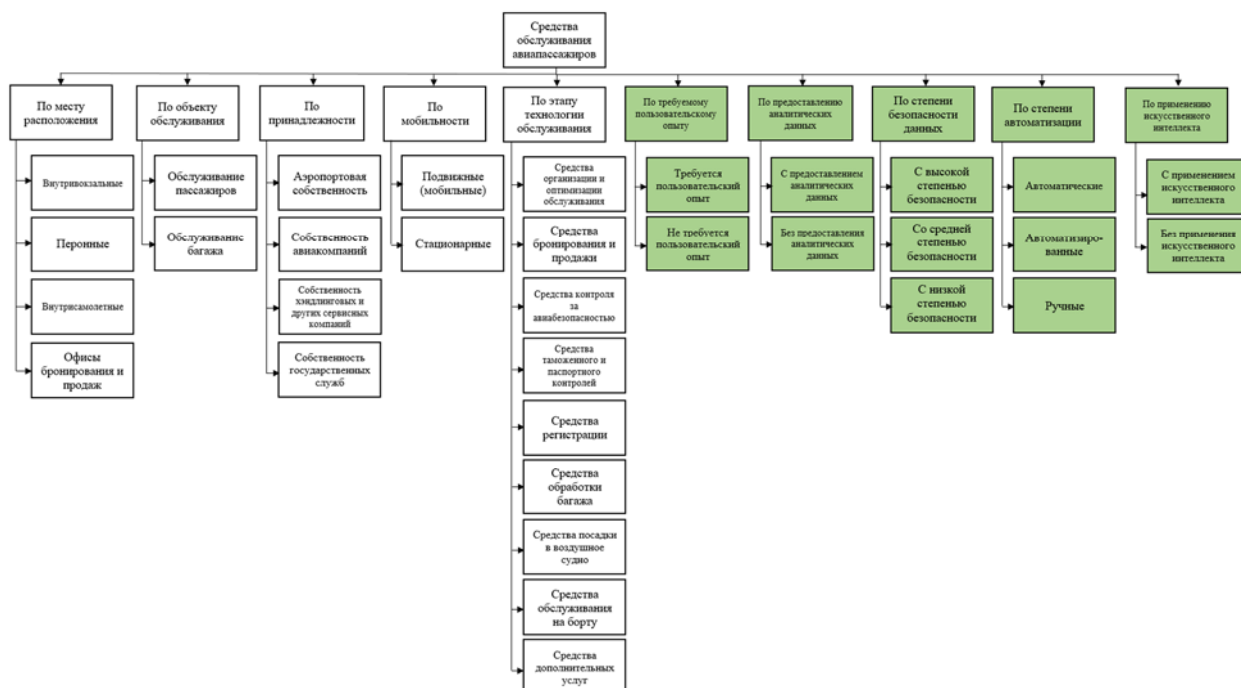


Рисунок 2 – Расширенная классификация средств обслуживания авиапассажиров

Предложенная расширенная классификация средств обслуживания авиапассажиров позволяет учитывать современные тенденции в области информационных систем и технологий.

Разрабатываемая система предполагает расположение в офисах бронирования и продаж, по объекту обслуживания является обслуживающей пассажиров, по принадлежности – аэропортовая собственность, по мобильности – подвижная (мобильная), по этапу технологии обслуживания – средство бронирования и продажи, по требуемому пользовательскому опыту – не требуется пользовательский опыт, по предоставлению аналитических данных – с предоставлением аналитических данных, по степени безопасности данных – со средней степенью безопасности, по степени автоматизации – автоматизированная, по применению искусственного интеллекта – с применением искусственного интеллекта.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время существует большое множество приложений, позволяющих бронировать и выкупать билеты на авиарейсы. Рассмотрим наиболее известные из них и проведем их сравнительный анализ.

1.2.1 «Аэрофлот»

Компания «Аэрофлот – Российские авиалинии» была основана в 1992 году для выполнения международных и внутренних пассажирских и грузовых воздушных перевозок. В настоящее время в группу «Аэрофлот» входят одноименная компания и несколько дочерних компаний, таких как «Донавиа», «Россия», «Победа» и другие [11]. Также у компании «Аэрофлот» имеется мобильное приложение. Главная страница данного приложения представлена на рисунке 3.

Через мобильное приложение «Аэрофлот» продает билеты на сумму более 1 млрд рублей в месяц. Общее число пользователей мобильного приложения «Аэрофлот» на всех платформах по состоянию на середину 2017 года составляет около 1,5 миллионов человек, которое продолжает расти [12]. При этом соотношение онлайн продаж билетов, включая продажи через мобильное приложение, на 2021 год составляло более 44 процентов [13].

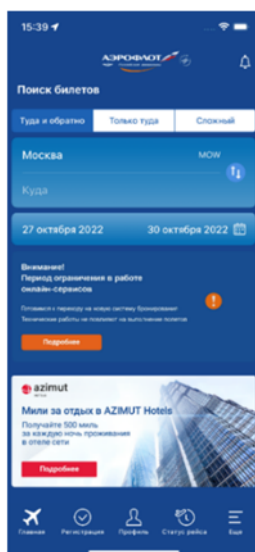


Рисунок 3 – Главная страница мобильного приложения «Аэрофлот»

На первой вкладке данного приложения расположена страница поиска авиабилетов на различные направления. Присутствует возможность выбора рейсов как в одну сторону, так и в обе сразу. Также имеется возможность поиска сложного маршрута, то есть с пересадками.

На второй вкладке имеется возможность пройти онлайн-регистрацию на рейс. В третьей вкладке можно посмотреть или изменить профиль пользователя. На четвертой вкладке можно узнать статус рейса, на который куплены или забронированы билеты. На последней вкладке представлена информация о приложении.

К достоинствам данного приложения можно отнести удобный интерфейс, позволяющий искать авиарейсы по разным направлениям. Также можно быстро проходить регистрацию на рейс и просматривать статус рейса. Еще через рассматриваемое приложение имеется возможность приобретать билеты на рейсы, проводящие другие авиакомпании.

К недостаткам приложения можно отнести медленную работу самого приложения при поиске рейсов, постоянную повторную авторизацию при выходе из приложения. Также при изменении личных данных пассажира купленный или забронированный билет можно будет только вернуть обратно, так как изменения в нем произвести нельзя. Следовательно, пассажира не пустят на рейс.

1.2.2 «Aviasales»

Aviasales – одно из самых популярных в России приложений для поиска авиабилетов. Данная платформа является лишь агрегатором, то есть на ней нельзя купить билеты, а только найти их и перейти на сайты авиакомпаний. На данный момент в базе агрегатора 728 авиакомпаний и около 200 агентств. В день совершается около 1.4 млн поисков [14]. У данного сервиса также существует мобильное приложение, главный экран которого приведен на рисунке 4.

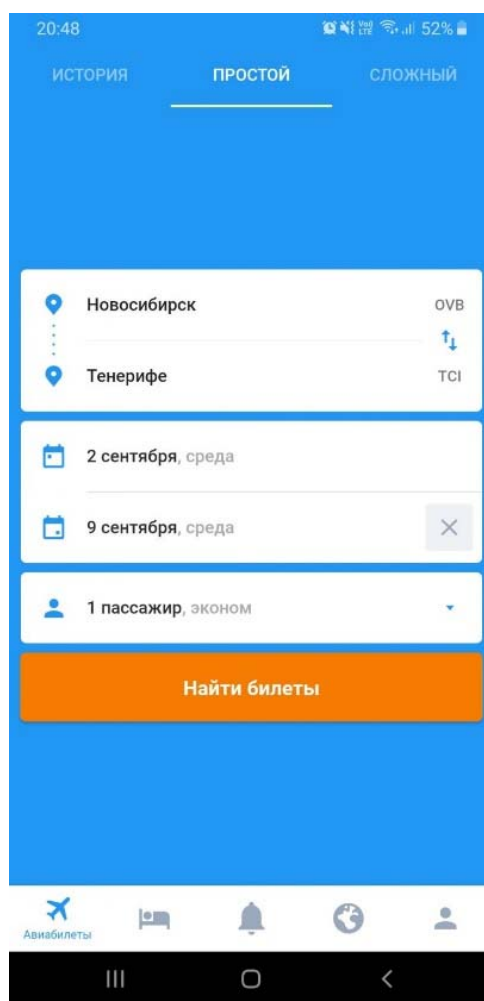


Рисунок 4 – Главная страница мобильного приложения «Aviasales»

В данном приложении имеется несколько вкладок. На первой вкладке можно осуществлять поиск авиарейсов (как простой, так и с пересадками) и просматривать историю предыдущих поисков. Также имеется возможность нахождения рейсов в обе стороны.

На второй вкладке также имеется возможность просмотра имеющихся в различных городах отелей и гостиниц и цен на них. На третьей вкладке можно включить или выключить напоминания о предстоящих рейсах и уведомления об изменении рейсов, на которые куплены билеты. Четвертая вкладка служит для поиска различных путешествий. На последней вкладке находится информация о пользователе, которую можно изменять.

Достоинством данного приложения является просмотр всех рейсов от разных авиакомпаний. Также плюсом приложения является быстрый поиск

рейсов от большого множества авиакомпаний, что позволяет пользователю сэкономить время и деньги.

Минусом является то, что именно данный сервис определяет, рейсы каких компании он будет показывать. На данный фактор пользователь не может никак повлиять. Также явным недостатком является то, что в приложении нельзя купить или забронировать билет: пользователю лишь предлагается перейти в приложение авиакомпании, которая выполняет выбранный рейс. Еще в приложении имеется платный контент, который позволяет получить лучшую обратную связь и другие преимущества.

1.2.3 «Яндекс.авиабилеты»

Яндекс.авиабилеты – продукт одной из самых известных поисковых систем «Яндекс». Яндекс.авиабилеты начал работать в 2009 году. При этом количество посетителей данного метапоисковика постоянно растет [15]. Данный сервис так же, как и Aviasales, позволяет только просматривать авиарейсы и осуществлять переход на приложения авиакомпаний, непосредственно осуществляющих рейсы. В 2016 году появилась мобильная версия данного продукта [16]. Главная страница приложения представлена на рисунке 5.

Как можно увидеть из рисунка 3, в приложении имеется 4 вкладки. На первой вкладке осуществляется поиск рейсов по направлению и дате вылета. Рейсы можно добавлять в избранное для отслеживания динамики цен. Избранные пользователем рейсы отображаются во второй вкладке. На третьей вкладке можно просмотреть информацию о рейсах, на которые были куплены или забронированы билеты. На четвертой вкладке можно изменять параметры пользователя приложения, а также настраивать уведомления пассажиров об изменении в расписании.

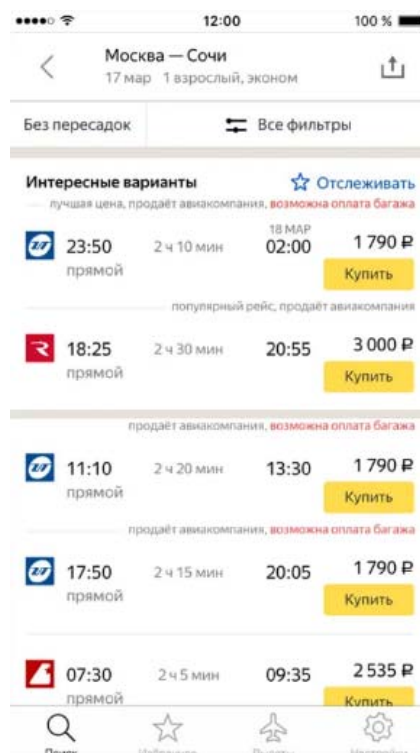


Рисунок 5 – Главная страница мобильного приложения «Яндекс.авиабилеты»

Плюсом приложения является довольно быстрый поиск рейсов от большого количества авиакомпаний.

Таким же минусом, как и у компании «Aviasales», является то, что осуществляется просмотр всех рейсов от разных авиакомпаний, на выбор которых пользователь никак не может повлиять. Вследствие этого, пользователь не может купить или забронировать билет из данного мобильного приложения. Также минусом в данном сервисе является то, что нельзя искать сложные (с пересадками) маршруты. Еще отсутствует функция автозаполнения документов, а также возможность вернуть купленный билет через приложение.

1.2.4 Сравнение аналогов

На данном этапе был произведен сравнительный анализ рассмотренных ранее систем обслуживания авиапассажиров. В результате анализа была сформирована таблица 1.

Таблица 1 – Сравнение аналогов

Система обслуживания авиапассажиров	Бронирование билетов	Покупка билетов	Автозаполнение данных	Платный функционал	Функция возврата билета
Аэрофлот	Есть	Есть	Нет	Нет	Есть
Aviasales	Нет	Нет	Нет	Есть	Есть
Яндекс.авиабилеты	Нет	Нет	Нет	Нет	Нет
Разрабатываемое приложение	Есть	Есть	Есть	Нет	Есть

Из данного сравнения можно сделать вывод, что в большинстве из рассмотренных приложений недоступна функция бронирования или покупки билетов. Также ни в одном из приложений нет возможности автозаполнения данных. В одном из сервисов имеется платный функционал, а в другом невозможно из приложения осуществить возврат билета.

Таким образом, в результате анализа программных продуктов аналогичного назначения было принято решение о разработке собственного приложения, которое будет иметь весь необходимый пользователю функционал и будет лишено недостатков продуктов-аналогов.

1.3 Применение искусственного интеллекта в сфере авиаперевозок

Применение искусственного интеллекта (ИИ) в сфере авиаперевозок имеет огромный потенциал для улучшения эффективности, безопасности и комфорта пассажиров. ИИ-технологии могут применяться в различных сферах авиаперевозок, таких как автоматизацию различных процессов, улучшение обслуживания клиентов, оптимизацию операций, повышение безопасности полетов и др.

Тем не менее, на данный момент «ИИ на авиационном рынке находится на начальной стадии» [17]. Это означает, что развитие и применение ИИ в авиации имеет большой потенциал для дальнейшего роста и улучшения.

Рассмотрим некоторые из ключевых областей, в которых ИИ может быть применен [18 – 24]:

– автоматизация операций. ИИ может помочь авиакомпаниям оптимизировать и автоматизировать различные операционные процессы. Например, системы ИИ могут использоваться для мониторинга и анализа данных о состоянии самолетов в реальном времени, что позволяет предсказывать поломки и проводить профилактические ремонтные работы. Это помогает авиакомпаниям увеличить доступность самолетов и снизить затраты на обслуживание. Также ИИ может использоваться для оптимизации планирования маршрутов, управления запасами топлива и прогнозирования спроса на авиабилеты, чтобы максимизировать загрузку судов и прибыльность авиакомпаний,

– улучшение комфорта пассажиров. Технологии ИИ могут значительно улучшить комфорт до и во время авиаперевозок. Например, системы ИИ могут использоваться для автоматической проверки багажа, что ускоряет процесс регистрации и сокращает время ожидания для пассажиров. Также, в аэропортах внедряются системы распознавания лиц на основе ИИ, которые позволяют осуществлять более быструю и безопасную идентификацию пассажиров. Это не только ускоряет процесс контроля безопасности, но и позволяет пассажирам избежать длительных очередей, упрощает процедуру прохождения контроля в аэропорту. Также технологии ИИ могут использоваться для персонализации обслуживания пассажиров, предлагая индивидуальные рекомендации о питании, развлечениях и других услугах во время полета. Также ИИ на основе рекомендательных систем может предлагать индивидуальные рекомендации о предпочитаемых направлениях авиаперелета, оптимальных маршрутах и времени пересадок, учитывая предпочтения пассажира и ограничения [25]. Это позволяет создавать более удобные маршруты с минимальными временными затратами и повышать уровень удовлетворенности клиентов,

– улучшение безопасности. ИИ имеет огромный потенциал для повышения безопасности авиаперевозок. Системы ИИ могут анализировать огромные объемы данных о полетах, погодных условиях, техническом

состоянии самолетов и других факторах, чтобы выявлять потенциальные угрозы и предотвращать аварии. Например, системы ИИ могут анализировать данные с бортовых датчиков для обнаружения нештатных ситуаций и предлагать пилотам рекомендации по действиям. Также системы ИИ могут использоваться для прогнозирования и предотвращения технических проблем, что способствует повышению надежности и безопасности полетов,

– развитие автономных систем. ИИ также играет важную роль в развитии автономных систем в авиации. Например, авионика на основе ИИ может управлять автопилотом и автономным летательным аппаратом, обеспечивая высокую точность и эффективность в осуществлении полетов. Это может снизить нагрузку на пилотов и улучшить безопасность полетов. Также ИИ-системы могут быть использованы для автоматического обнаружения и реагирования на угрозы, такие как беспилотные воздушные аппараты или сближение с другими самолетами, что повышает безопасность полетов,

– прогнозирование и планирование. Применение ИИ также полезно при прогнозировании и планировании авиаперевозок. Алгоритмы машинного обучения могут использоваться для анализа и прогнозирования спроса на авиабилеты, что помогает авиакомпаниям оптимизировать цены и заполнить самолеты наиболее эффективно. Также, ИИ может помочь в определении оптимальных маршрутов и расписания рейсов, учитывая различные факторы, такие как погодные условия, пересадки и доступность аэропортов.

Таким образом, применение искусственного интеллекта в сфере авиаперевозок позволяет улучшить всех аспекты авиации – от оптимизации операций и повышения безопасности до улучшения комфорта пассажиров и развития автономных систем. Использование ИИ позволяет авиационной промышленности работать более эффективно, улучшать сервис обслуживания пассажиров и создавать более безопасные авиаперевозки.

1.4 Выбор технологий и инструментальных средств разработки

Было решено использовать одну из самых известных методологий разработки Rational Unified Process (RUP). Данная модель описывает общий процесс, которым должна руководствоваться проектная команда, чтобы создать программный продукт. В первую очередь, данная методология предполагает итеративную разработку. То есть вся разработка делится на четыре основных этапа, каждый из которых включает в себя одну или несколько итераций, продолжающихся от 2 до 6 недель [26]:

- начальная стадия – определяются границы проекта, требования, приводится экономическое обоснование,
- стадия уточнения – производится уточнение требований и обновление экономического обоснования, проектирование архитектуры,
- стадия построения – реализуется основная часть функциональности продукта,
- стадия внедрения – продукт дорабатывается до окончательной версии и передается заказчику, производится обучение пользователей.

Технология RUP была выбрана, так как обладает рядом преимуществ относительно других методологий. Во-первых, RUP обеспечивает гибкость процесса разработки. То есть в проект можно относительно быстро вносить корректировки в случае выявления ошибок на предыдущих стадиях или изменений требований заказчиком. Также в данной методологии производится оценка рисков уже на первых стадиях разработки, поэтому в случае его экономической нецелесообразности можно завершить проект еще на начальной стадии. Помимо этого, в отличие от других методологий, например, eXtreme Programming (XP), постоянное взаимодействие между заказчиком не требуется.

Технология RUP для анализа требований, предполагает использование прецедентов использования (use case). В связи с этим было решено

использовать диаграммы на языке моделирования UML. Для построения диаграмм использовалось средство UML-моделирования ArgoUML.

Для выполнения онтологического моделирования использовалась система Protege, представляющая собой одно из наиболее распространённых средств разработки онтологий, поддерживающее язык OWL и обладающее удобным графическим интерфейсом. С её помощью была разработана онтология предметной области, используемая для формализации знаний.

Для проведения имитационного моделирования применялась среда GPSS Studio, предназначенная для построения и анализа дискретно-событийных моделей. Данная система позволяет описывать процессы в виде последовательности событий с учётом вероятностных характеристик и временных задержек, что делает её удобной для анализа поведения сложных систем в условиях неопределённости.

В качестве системы управления базами данных была выбрана Microsoft SQL Server 2019, являющаяся одной из самых популярных СУБД [27]. Серверная часть взаимодействует с базой данных с помощью объектно-реляционного модуля Entity Framework Core 6. Данный модуль позволяет управлять сущностями базы данных через их проекцию – классы C#, описанные с использованием специальных атрибутов, а также предоставляет возможность выполнения SQL запросов через язык запросов Linq [28].

Серверная часть приложения будет реализована при помощи платформы ASP.NET, которая предоставляет широкий функционал для обработки HTTP-запросов и методы управления поведением сервера. Сервер будет работать под управлением операционной системы Windows 10.

Клиентская часть приложения будет реализована при помощи платформы Xamarin, предназначенной для построения современных производительных приложений для iOS, Android и Windows [29]. Целевой платформой в данной разработке будет являться Android.

Для реализации рекомендательной системы будет применяться язык программирования Python с использованием библиотек Scikit-learn, SciPy и

NetworkX. Запуск рекомендательной системы будет осуществляться под управлением операционной системы Windows 10.

1.5 Анализ целевой аудитории

Целевой аудиторией проекта являются кассиры аэропорта, использующие для своей работы мобильные устройства и обслуживающие людей, которым необходимо осуществить перелет из одного города в другой по различным целям, например, командировка, туризм, посещение родственников и другое, а также непосредственные клиенты, которые могут с использованием своего мобильного устройства произвести действия по покупке или бронированию авиабилетов. В настоящее время количество авиаперевозок имеет тренд на повышение. Этот вывод можно сделать на основании данных, предоставляемых сайтом Aviastat, который берет статистику из заключений Росстата и предоставляет ее в виде графиков [30, 31]. На рисунке 6 представлен график, на котором показаны выполненный пассажирооборот, тренд, а также прогноз в мире на пассажирские авиаперевозки за 1980 – 2025 годы.

Как видно из рисунка, мировой пассажирооборот в интервале с 1980 по 2020 годы постоянно растет с небольшими падениями в 2001 году после трагедии 11 сентября и в 2008 году в связи с мировым финансовым кризисом.

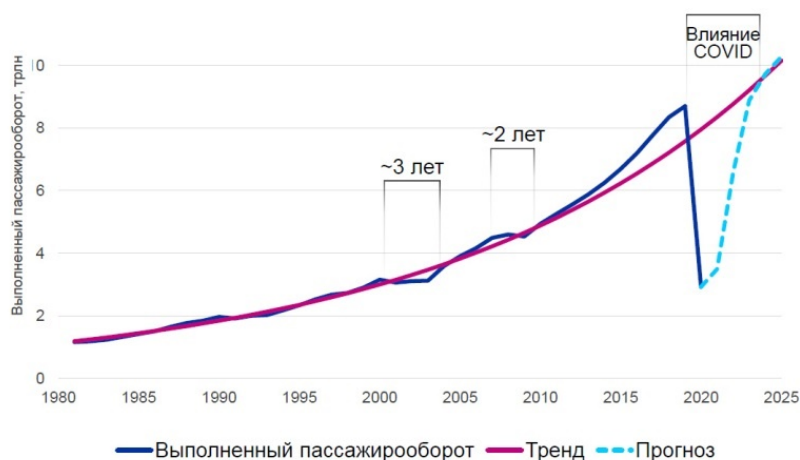


Рисунок 6 – Динамика пассажирооборота в 1980 – 2025 годах

В 2020 году произошло сильное падение мирового пассажирооборота почти в 3 раза, связанное с пандемией COVID-19. Но, как можно увидеть на графике, уже к 2025 году по прогнозам пассажирооборот вернется к прежнему уровню и даже возрастет. Это подтверждает статистика 2024 года, в котором мировой пассажирооборот на 3,8% превысил уровень допандемийного 2019 года [32].

На пассажирооборот воздушного транспорта в России пандемия COVID-19, как видно из графика (рисунок 7), также оказала довольно сильное влияние (особенно на международные перевозки) [33]. На графике видно небольшое падение пассажирооборота в 2022 году в связи с проведением специальной военной операции. Но в большей степени это сказывается на международных перевозках, когда как тренд на внутренний пассажирооборот растет.

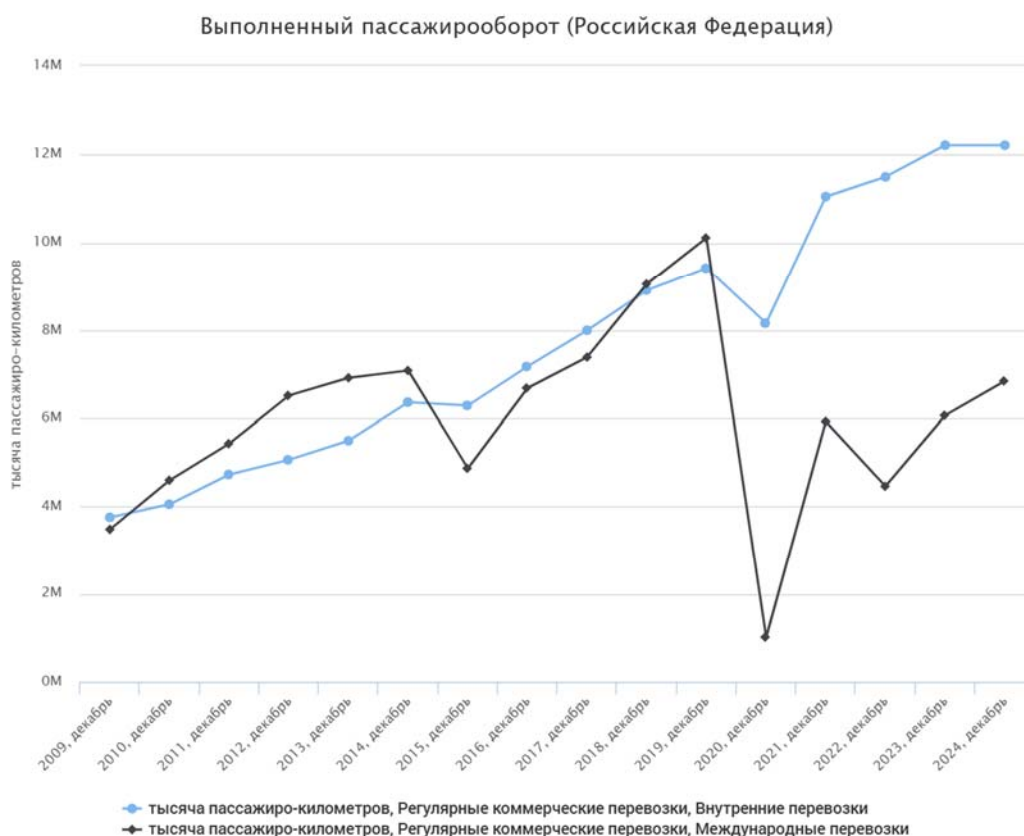


Рисунок 7 – Динамика пассажирооборота в России в 2009-2024 годах

На рисунке 8 представлены доли основных производителей смартфонов на мировом рынке за последние 2 года [34].

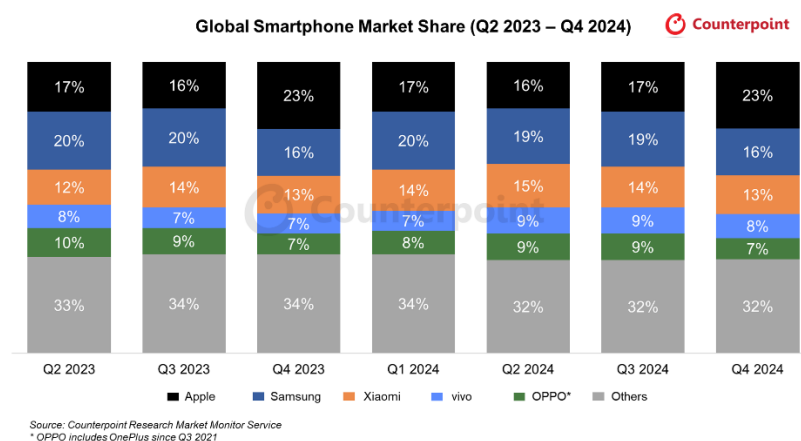


Рисунок 8 – Доли производителей смартфонов на мировом рынке

Из графика можно увидеть, что основную долю (около 70%) на мировом рынке производителей смартфонов занимают смартфоны с операционной системой Android, такие как Samsung, Xiaomi, vivo и другие. Компания же Apple, использующая операционную систему iOS, хоть и увеличила количество своих смартфонов на рынке до 23%, но даже не приблизилась ко своим конкурентам. Поэтому с целью увеличения количества пользователей, которые смогут использовать разрабатываемое мобильное приложение, было решено создавать его именно для операционной системы Android.

На рисунке 9 приведена статистика распространения разных версий системы Android [35].



Рисунок 9 – Статистика распространения разных версий системы Android

Из рисунка можно заметить, что 81,2% пользователей операционной системы Android используют версию не ниже 9.0. Поэтому было принято решение разрабатывать приложение, которое будет работать на версии 9.0 или выше.

1.6 Структурное и календарное планирование для IT-проекта разработки системы обслуживания авиапассажиров

Без правильного планирования невозможно качественно и в установленные сроки выполнить работы по любому проекту. Это касается и сферы информационных технологий. Планирование предполагает построение графика выполнения работ проекта таким образом, чтобы выполнить все поставленные задачи с приемлемым качеством за минимально возможное время с использованием минимума затрат [36 – 40].

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

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области и постановка задачи	3 дня	1
4	Анализ программных продуктов аналогичного назначения	9 дней	1
5	Анализ функциональных требований	6 дней	3;4
6	Анализ требований к интерфейсу	6 дней	5
7	Анализ целевой аудитории	3 дня	5
8	Анализ требований завершен	-	6;7
9	Выбор решения	-	-
10	Анализ существующих технологий разработки ПО	6 дней	8
11	Выбор технологий разработки ПО	3 дня	10
12	Выбор решения завершен	-	11
13	Проектирование программных средств	-	-
14	Проектирование архитектуры приложения	15 дней	12
15	Проектирование базы данных	9 дней	12
16	Проектирование серверной части приложения	9 дней	14;15
17	Проектирование автоматизированной системы	12 дней	14
18	Проектирование интерфейса пользователя	6 дней	14
19	Проектирование завершенно	-	16;17;18
20	Реализация программных средств	-	-
21	Реализация информационной системы	15 дней	14;15
22	Реализация серверной части приложения	57 дней	19;21
23	Реализация мобильного приложения	63 дня	19;21
24	Реализация завершенна	-	22;23
25	Тестирование	-	-
26	Изучение основных видов тестирования	3 дня	24

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
27	Функциональное тестирование	6 дней	26
28	Модульное тестирование	9 дней	26
29	Тестирование завершено	-	27;28
30	Составление и оформление документации	21 день	29
31	Конец проекта	-	30

Ниже в таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Перечень ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист 1	Трудовой	400 руб/ч	600 руб/ч
Программист 2	Трудовой	350 руб/ч	550 руб/ч
Аналитик	Трудовой	350 руб/ч	550 руб/ч
Дизайнер	Трудовой	300 руб/ч	400 руб/ч
Тестирующий	Трудовой	300 руб/ч	400 руб/ч
Компьютер	Материальный	2000 руб	-
Бумага	Материальный	500 руб	-
Интернет	Затраты	-	-

В таблице 4 приведено распределение ресурсов по задачам.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области и постановка задачи	Аналитик Компьютер Интернет	40% 3 1500 руб
4	Анализ программных продуктов аналогичного назначения	Аналитик	50%
5	Анализ функциональных требований	Программист 1 Программист 2 Аналитик	100% 100% 50%
6	Анализ требований к интерфейсу	Дизайнер Аналитик	100% 30%
7	Анализ целевой аудитории	Аналитик	50%
8	Анализ требований завершен	-	-
9	Выбор решения	-	-
10	Анализ существующих технологий разработки ПО	Программист 2	30%
11	Выбор технологий разработки ПО	Программист 1 Программист 2	50% 50%
12	Выбор решения завершен	-	-
13	Проектирование программных средств	-	-
14	Проектирование архитектуры приложения	Программист 1 Программист 2 Аналитик	100% 100% 50%
15	Проектирование базы данных	Программист 2 Аналитик	70% 50%
16	Проектирование серверной части приложения	Программист 1 Аналитик	100% 30%
17	Проектирование автоматизированной системы	Программист 2 Аналитик	100% 30%

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
18	Проектирование интерфейса пользователя	Дизайнер Аналитик	60% 50%
19	Проектирование завершено	-	-
20	Реализация программных средств	-	-
21	Реализация информационной системы	Программист 2	100%
22	Реализация серверной части приложения	Программист 1	100%
23	Реализация мобильного приложения	Программист 2	100%
24	Реализация завершена	-	-
25	Тестирование	-	-
26	Изучение основных видов тестирования	Программист 2	30%
27	Функциональное тестирование	Тестировщик	70%
28	Модульное тестирование	Тестировщик Программист 2	70% 50%
29	Тестирование завершено	-	-
30	Составление и оформление документации	Аналитик Программист 1 Программист 2 Бумага	100% 50% 50% 2
31	Конец проекта	-	-

Таким образом, реализацией проекта будут заниматься программист 1, программист 2, аналитик и дизайнер, которые в процессе работы используют три компьютера и доступ в интернет. Для вычисления длительности реализации проекта и его критического пути был построен сетевой график, представленный на рисунке 10.

Теперь необходимо рассчитать критический путь проекта. Для этого необходимо высчитать раннее и позднее времена начала работ для всех задач проекта, и на основе данных сведений определить резерв времени для каждой из задач.

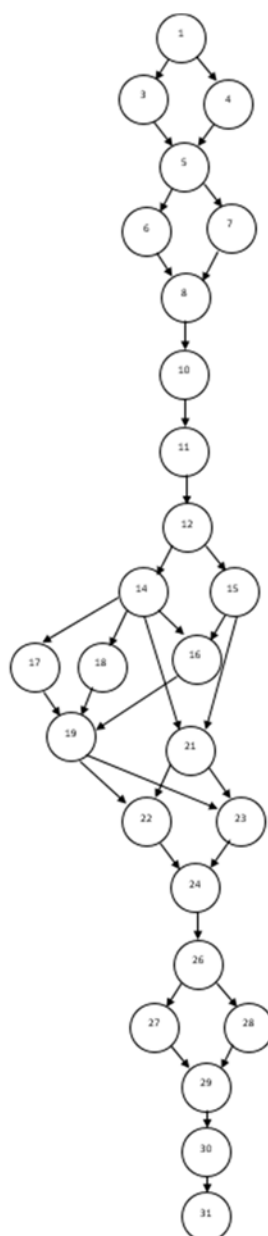


Рисунок 10 – Сетевой график проекта

В таблице 5 представлены расчеты раннего времени начала работ для его задач.

Таблица 5 – Расчеты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	1	0	0	0+0	0
5	3	0	3	Max(0+3;0+9)	9
	4	0	9		
6	5	9	6	9+6	15
7	5	9	6	9+6	15

Продолжение таблицы 5

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
8	6	15	6	Max(15+6;15+3)	21
	7	15	3		
10	8	21	0	21+0	21
11	10	21	6	21+6	27
12	11	27	3	27+3	30
14	12	30	0	30+0	30
15	12	30	0	30+0	30
16	14	30	15	Max(30+15;30+9)	45
	15	30	9		
17	14	30	15	30+15	45
18	14	30	15	30+15	45
19	16	45	9	Max(45+9;45+12;45+6)	57
	17	45	12		
	18	45	6		
21	14	30	15	Max(30+15;30+9)	45
	15	30	9		
22	19	57	0	Max(57+0;45+15)	60
	21	45	15		
23	19	57	0	Max(57+0;45+15)	60
	21	45	15		
24	22	60	57	Max(60+57;60+63)	123
	23	60	63		
26	24	123	0	123+0	123
27	26	123	3	123+3	126
28	26	123	3	123+3	126
29	27	126	6	Max(126+6;126+9)	135
	28	126	9		
30	29	135	0	135+0	135
31	30	135	21	135+21	156

Также было рассчитано позднее время начала работ для задач проекта (таблица 6).

Таблица 6 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
31	-	-	-	-	156
30	31	156	21	156-21	135
29	30	135	0	135-0	135
28	29	135	9	135-9	126
27	29	135	6	135-6	129
26	27	129	3	Min(129-3;126-3)	123
	28	126			
24	26	123	0	123-0	123
23	24	123	63	123-63	60
22	24	123	57	123-57	66

Продолжение таблицы 6

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
21	22 23	66 60	15	Min(66-15;60-15)	45
19	22 23	66 60	0	Min(66-0;60-0)	60
18	19	60	6	60-6	54
17	19	60	12	60-12	48
16	19	60	9	60-9	51
15	16 21	51 45	9	Min(51-9;45-9)	36
14	16 17 18 21	51 48 54 45	15	Min(51-15;48-15;54-15; 45-15)	30
12	14 15	30 36	0	Min(30-0;36-0)	30
11	12	30	3	30-3	27
10	11	27	6	27-6	21
8	10	21	0	21-0	21
7	8	21	3	21-3	18
6	8	21	6	21-6	15
5	6 7	15 18	6	Min(15-6;18-6)	9
4	5	9	9	9-9	0
3	5	9	3	9-3	6
1	3 4	6 0	0	Min(6-0;0-0)	0

По результатам нахождения раннего и позднего времени начала работ можно найти критический путь проекта. Вычисление критического пути представлено в таблице 7.

Таблица 7 – Вычисление критического пути проекта

Номер работы	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	6	6
4	0	0	0
5	9	9	0
6	15	15	0
7	15	18	3
8	21	21	0
10	21	21	0
11	27	27	0
12	30	30	0
14	30	30	0
15	30	36	6
16	45	51	6
17	45	48	3
18	45	54	9

Продолжение таблицы 7

Номер работы	Раннее время начала	Позднее время начала	Резерв времени
19	57	60	3
21	45	45	0
22	60	66	6
23	60	60	0
24	123	123	0
26	123	123	0
27	126	129	3
28	126	126	0
29	135	135	0
30	135	135	0
31	156	156	0

На основе полученных данных можно сделать вывод, что критический путь проходит через работы с номерами 1, 4, 5, 6, 8, 10, 11, 12, 14, 21, 23, 24, 26, 28, 29, 30, 31, а общее время выполнения проекта составляет 156 дней. Граф проекта, где пунктирными линиями указан критический путь, приведен на рисунке 11.

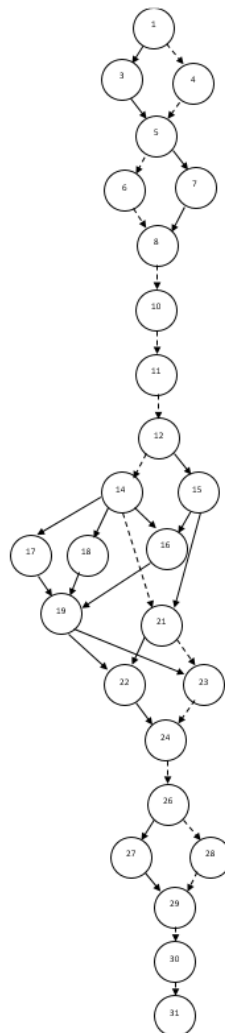


Рисунок 11 – Критический путь проекта

Теперь проверим правильность расчетов продолжительности проекта и вычислим величину его бюджета. Для этого воспользуемся системой управления проектами Microsoft Project 2007, которая позволяет автоматически произвести все необходимые вычисления [41]. Список составленных задач в системе Microsoft Project представлен на рисунке 12. В качестве времени начала проекта была выбрана дата 11.12.2023.

	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Пн 11.12.23	Пн 11.12.23		
2	- Анализ требований	21 дней	Пн 11.12.23	Вт 16.01.24		
3	Анализ предметной области и постановка задачи	3 дней	Пн 11.12.23	Ср 13.12.23	1	Аналитик[40%],Компьютер[3],Интернет[1 500,00 P]
4	Анализ программных продуктов аналогичного назначения	9 дней	Пн 11.12.23	Чт 21.12.23	1	Аналитик[50%]
5	Анализ функциональных требований	6 дней	Пт 22.12.23	Пт 29.12.23	3,4	Программист 1,Программист 2,Аналитик[50%]
6	Анализ требований к интерфейсу	6 дней	Вт 09.01.24	Вт 16.01.24	5	Дизайнер,Аналитик[30%]
7	Анализ целевой аудитории	3 дней	Вт 09.01.24	Чт 11.01.24	5	Аналитик[50%]
8	Анализ требований завершен	0 дней	Вт 16.01.24	Вт 16.01.24	6,7	
9	- Выбор решения	9 дней	Ср 17.01.24	Пн 29.01.24		
10	Анализ существующих технологий разработки ПО	6 дней	Ср 17.01.24	Ср 24.01.24	8	Программист 2[30%]
11	Выбор технологий разработки ПО	3 дней	Чт 25.01.24	Пн 29.01.24	10	Программист 1[50%],Программист 2[50%]
12	Выбор решения завершен	0 дней	Пн 29.01.24	Пн 29.01.24	11	
13	- Проектирование программных средств	27 дней	Вт 30.01.24	Чт 07.03.24		
14	Проектирование архитектуры приложения	15 дней	Вт 30.01.24	Пн 19.02.24	12	Программист 1,Программист 2,Аналитик[50%]
15	Проектирование базы данных	9 дней	Вт 30.01.24	Пт 09.02.24	12	Программист 2[70%],Аналитик[50%]
16	Проектирование серверной части приложения	9 дней	Вт 20.02.24	Пн 04.03.24	14,15	Программист 1,Аналитик[30%]
17	Проектирование мобильного приложения	12 дней	Вт 20.02.24	Чт 07.03.24	14	Программист 2,Аналитик[30%]
18	Проектирование интерфейса пользователя	6 дней	Вт 20.02.24	Ср 28.02.24	14	Дизайнер[60%],Аналитик[50%]
19	Проектирование завершенно	0 дней	Чт 07.03.24	Чт 07.03.24	16,17,18	
20	- Реализация программных средств	78 дней	Вт 20.02.24	Вт 18.06.24		
21	Реализация информационной системы	15 дней	Вт 20.02.24	Ср 13.03.24	14,15	Программист 2
22	Реализация серверной части приложения	57 дней	Чт 14.03.24	Пт 07.06.24	19,21	Программист 1
23	Реализация мобильного приложения	63 дней	Чт 14.03.24	Вт 18.06.24	19,21	Программист 2
24	Реализация завершена	0 дней	Вт 18.06.24	Вт 18.06.24	22,23	
25	- Тестирование	12 дней	Ср 19.06.24	Чт 04.07.24		
26	Изучение основных видов тестирования	3 дней	Ср 19.06.24	Пт 21.06.24	24	Программист 2[30%]
27	Функциональное тестирование	6 дней	Пн 24.06.24	Пн 01.07.24	26	Тестирующий[70%]
28	Модульное тестирование	9 дней	Пн 24.06.24	Чт 04.07.24	26	Тестирующий[70%],Программист 2[50%]
29	Тестирование завершено	0 дней	Чт 04.07.24	Чт 04.07.24	27,28	
30	Составление и оформление документации	21 дней	Пт 05.07.24	Пт 02.08.24	29	Аналитик,Программист 1[50%],Программист 2[50%],Бумага[2]
31	Конец проекта	0 дней	Пт 02.08.24	Пт 02.08.24	30	

Рисунок 12 – Список задач проекта в системе Microsoft Project

Построенная системой Microsoft Project диаграмма Ганта приведена на рисунке 13.

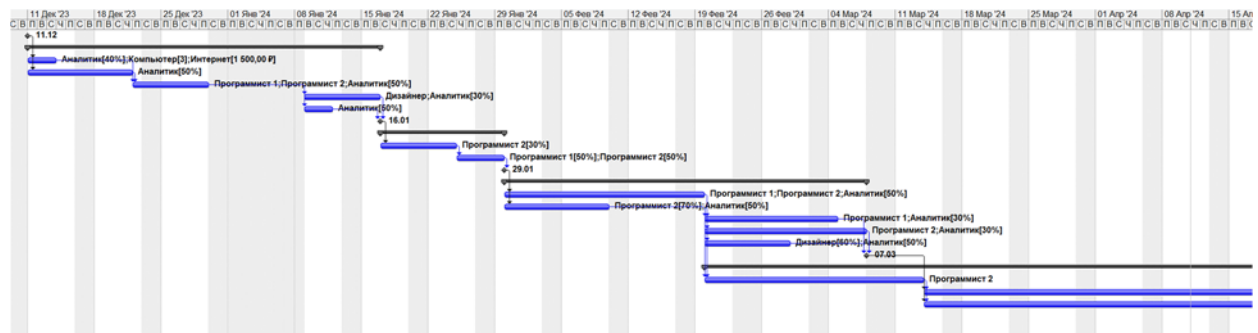


Рисунок 13 – Диаграмма Ганта

График загруженности ресурса «Программист 1» представлен на рисунке 14.

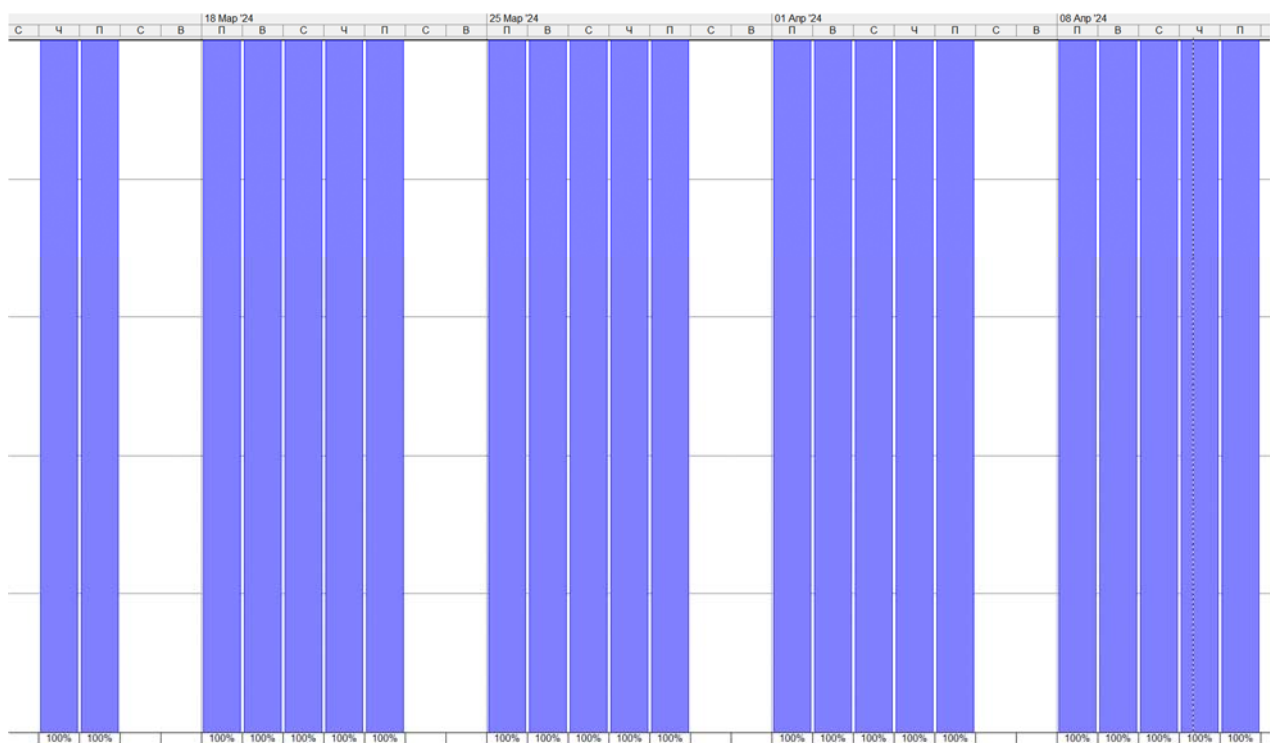


Рисунок 14 – График загрузки ресурса «Программист 1»

График загрузки ресурса «Программист 2» представлен на рисунке 15.

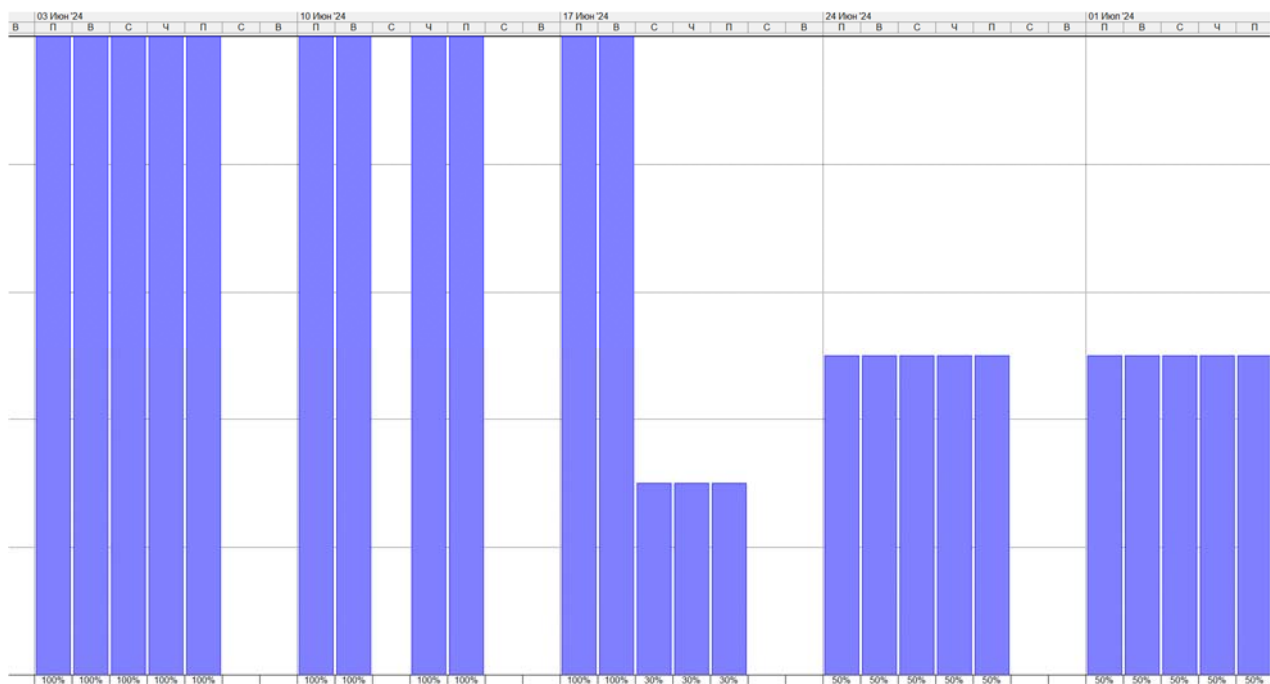


Рисунок 15 – График загрузки ресурса «Программист 2»

График загрузки ресурса «Аналитик» представлен на рисунке 16.

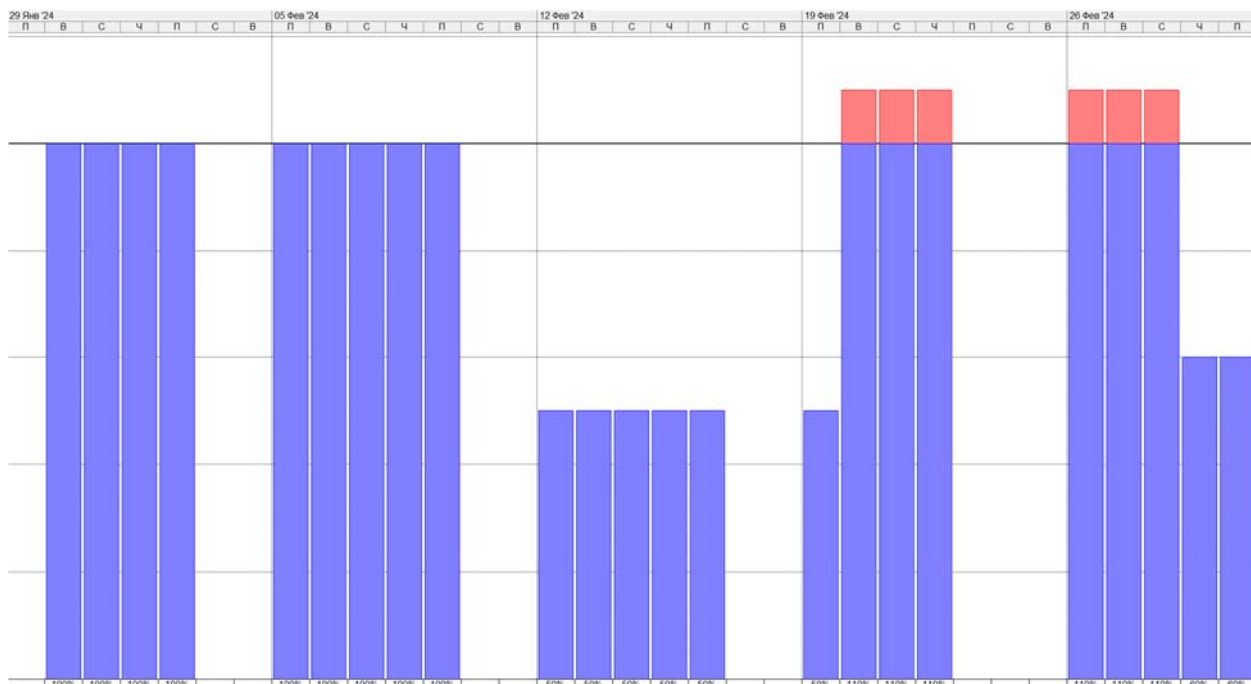


График загрузки ресурса «Дизайнер» представлен на рисунке 17.

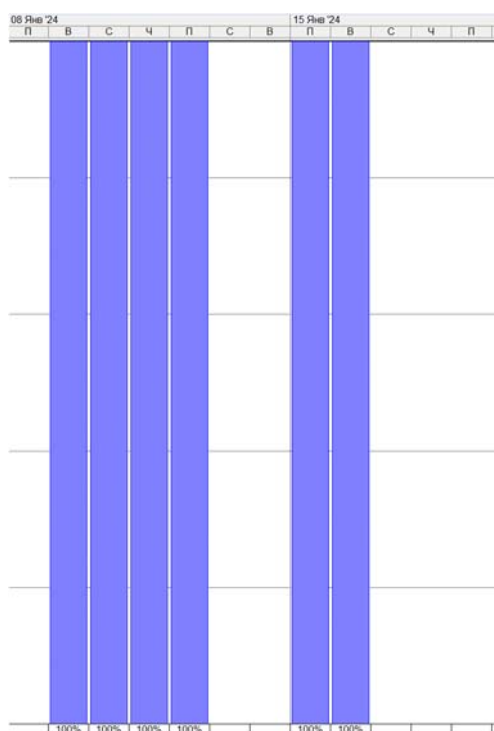


График загрузки ресурса «Тестировщик» представлен на рисунке 18.

На рисунке 19 представлен обзорный отчет проекта, составленный системой Microsoft Project. Из рисунка видно, что общая длительность

проекта составляет 156 дней (совпадает со значением, вычисленным вручную), а предполагаемые затраты – 903280 рублей.



Рисунок 18 – График загрузки ресурса «Тестировщик»

Даты			
Начало:	Пн 11.12.23	Окончание:	Пт 02.08.24
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	156 дней	Оставшаяся:	156 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	156 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	2 479,2 ч	Оставшиеся:	2 479,2 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	2 479,2 ч	Процент завершения:	0%
Затраты			
Текущий план:	903 280,00 Р	Оставшиеся:	903 280,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	903 280,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	31	Трудовые ресурсы:	3
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	3
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	31	Итого ресурсов:	8

Рисунок 19 – Обзорный отчет проекта

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

2 Разработка автоматизированной системы обслуживания авиапассажиров и гибридной рекомендательной системы

2.1 Онтологическая модель предметной области

Для эффективного проектирования автоматизированной системы обслуживания авиапассажиров необходимо провести ее предварительный анализ, структурировать и систематизировать информацию о предметной области. Для этого было проведено онтологическое моделирование. В среде Protege был разработан набор классов и свойств классов, а также взаимодействий между классами [42, 43]. Основываясь на этом, на языках OWL и SWRL была разработана онтология в среде проектирования Protege. Одной из целей данной научно-исследовательской работы являлось построение онтологической модели предметной области. Онтологический граф (рисунок 20), иерархии классов (рисунок 21) и свойств объектов модели (рисунок 22) представлены ниже.

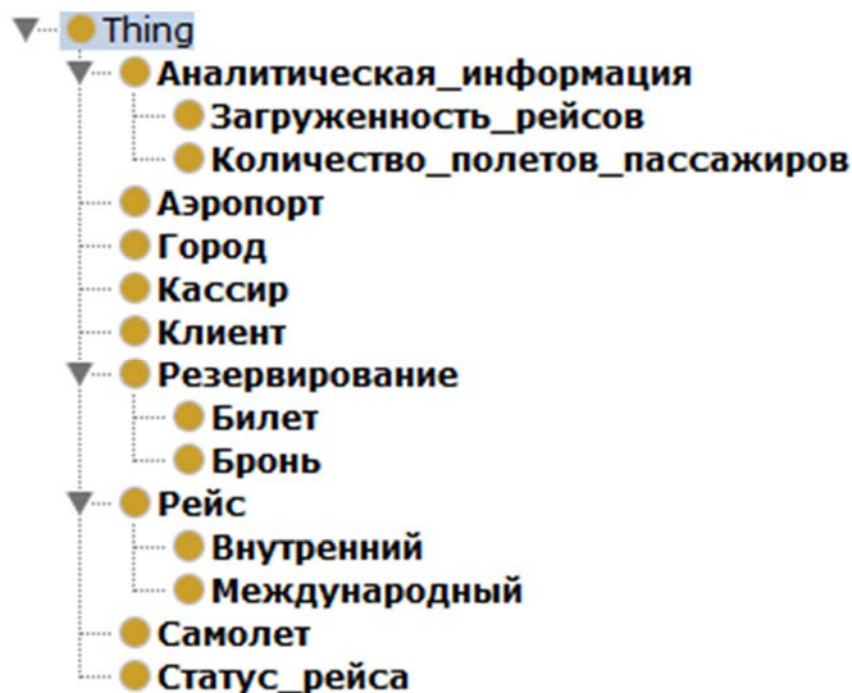


Рисунок 20 – Иерархия классов онтологической модели

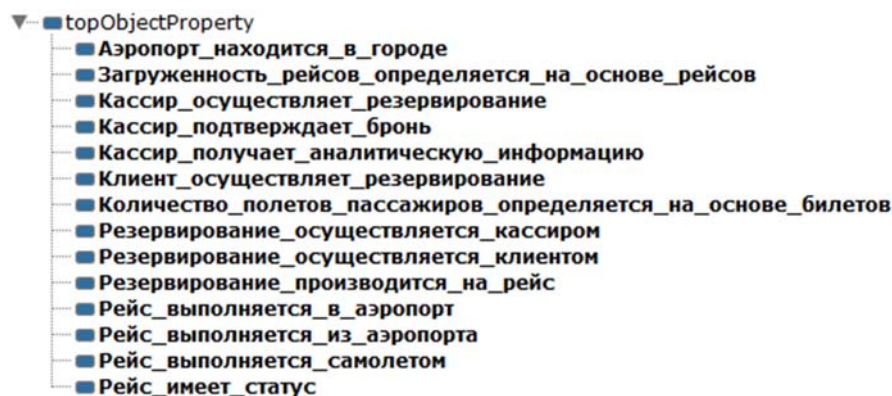


Рисунок 21 – Иерархия свойств объектов онтологической модели

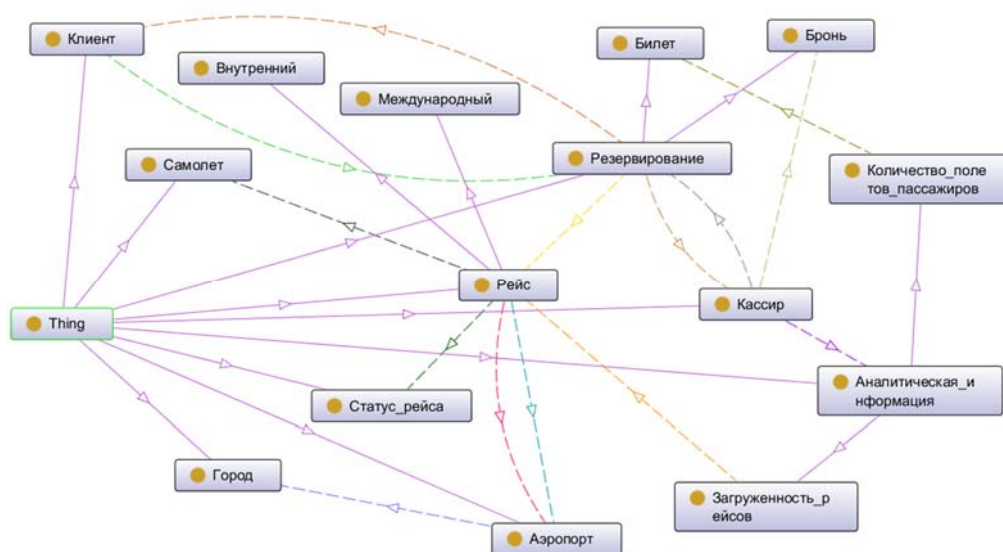


Рисунок 22 – Онтологический граф модели

Разработанная онтологическая модель в отличие от существующих моделей, которые представляют лишь поверхностное изучение онтологий, а также содержат значительно меньше классов и свойств [44], позволяет проводить эффективное проектирование автоматизированной системы обслуживания авиапассажиров, в том числе на основе визуального моделирования с применением языка UML.

2.2 Имитационное моделирование объекта исследования

Главная цель моделирования заключается в определении оптимальных параметров автоматизированной системы обслуживания авиапассажиров. Эти параметры включают в себя количество кассиров в аэропорту и время обработки запросов сервером. Важно учесть, что чрезмерно высокие

значения этих параметров могут привести к избыточным операционным расходам аэропорта, что экономически нецелесообразно, в то время как слишком низкие значения могут нарушить нормальное функционирование системы. Поэтому оптимальными будут минимальные значения этих параметров, при которых система продолжает работать удовлетворительно.

Для обеспечения эффективной работы аэропорта необходимо оптимизировать его структуру, которая может быть значительно улучшена с использованием технологий имитационного моделирования. Эти технологии в настоящее время широко применяются в научных исследованиях, управленческой деятельности, производстве, обучении и других областях.

Основная функция автоматизированной системы обслуживания авиапассажиров осуществляется через сервер, который обрабатывает запросы пользователей, такие как покупка и бронирование билетов, сдача билетов и т.п. Скорость обработки запросов сервером напрямую влияет на скорость получения ответа пользователем, что является ключевым аспектом для обеспечения удобства работы с системой.

Однако работа автоматизированной системы обслуживания авиапассажиров включает в себя и другие аспекты, связанные с человеческим фактором. В частности, не все авиапассажиры используют приложение для самостоятельного оказания услуг по авиаперевозкам. Некоторые люди предпочитают традиционные методы обслуживания, включая контакт с сотрудниками аэропорта [45]. Это может быть вызвано различными причинами, такими как отсутствие технической грамотности, неудобство использования мобильных приложений или просто предпочтение общения с живым человеком. Автоматизированные системы должны учитывать этот человеческий фактор. Таким образом, в штат аэропорта должны входить кассиры.

Существуют научные работы, в которых исследуются параметры систем обслуживания авиапассажиров [46], но в данных работах высчитываются другие параметры и применяются только теоретические

оценки, основанные на теории систем массового обслуживания без их имитации в специализированных средах.

С учетом определенных выше целей в среде GPSS Studio были разработаны две оригинальные дискретно-событийные имитационные GPSS-модели [42, 47 – 50]. Первая модель предназначена для симуляции обслуживания пассажиров кассирами, вторая – для моделирования обработки запросов сервером. Особенностью этих моделей является их комплексный характер, который позволяет учитывать как технические особенности оборудования автоматизированной системы обслуживания авиапассажиров, так и человеческий фактор, связанный с деятельностью персонала. Исходные коды моделей на языке GPSS представлены в приложении А. Рассмотрим структуру каждой из этих моделей более подробно.

Q-схема имитационной модели обслуживания пассажиров кассирами для автоматизированной системы обслуживания авиапассажиров представлена на рисунке 23.

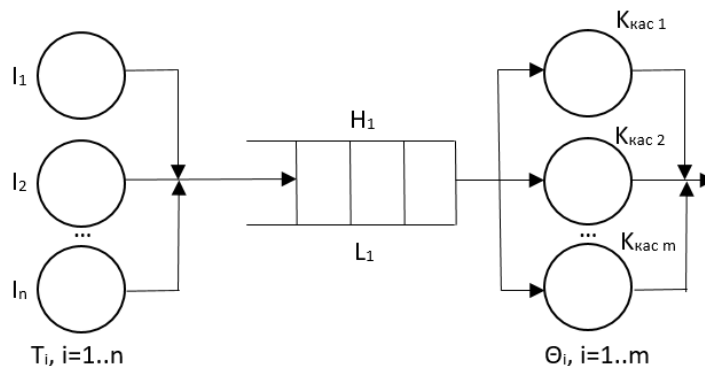


Рисунок 23 – Q-схема имитационной модели обслуживания пассажиров кассирами

В схеме показаны все этапы обслуживания: поступление заявок на обслуживание пассажиров ($I_1 - I_n$, n – число разновидностей обслуживания), обслуживание пассажира кассиром ($K_{\text{кас } 1} - K_{\text{кас } m}$, m – число кассиров).

Исходные количественные данные (интервалы поступления заявок, время обслуживания) для модели были получены экспериментальным способом путем изучения статистики по приходу пассажиров в кассы аэропорта, по типам обслуживания, а также по скорости работы кассиров.

Первый эксперимент производится при следующих значениях среднего интервала времени поступления заявок на обслуживание пассажиров в минутах:

- покупка билетов – 4,
- бронирование билетов – 6,
- выкуп брони – 7,
- снятие брони – 17,
- возврат билета – 80,
- подтверждение бронирования билетов – 2.

Всего поступает примерно 3800 заявок в неделю.

Результаты моделирования указаны в таблице 8.

Таблица 8 – Результаты моделирования обслуживания пассажиров кассирами (эксперимент 1)

Количество кассиров	Результат
10	Более 6% заявок обслуживаются более 20 мин. Средняя очередь 0,2.
9	Более 6% заявок обслуживаются более 20 мин (рисунок 24). Средняя очередь 0,6.
8	Более 8% заявок обслуживаются более 20 мин. Средняя очередь 1,4.
7	Более 11% заявок обслуживаются более 20 мин. Средняя очередь 3,6.

Оптимальным является вариант с 9 кассирами. Далее приведена гистограмма частот времени пребывания заявок в системе для количества кассиров 9, построенная на основе данных, полученных GPSS Studio.

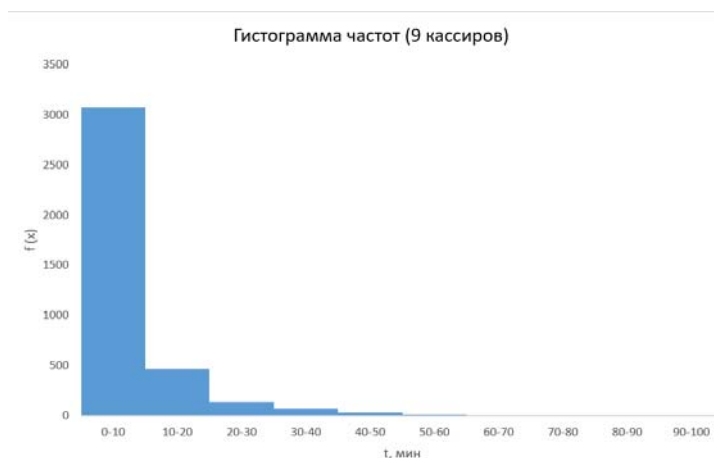


Рисунок 24 – Гистограмма частот времени пребывания заявки в системе при работе 9 кассиров

На рисунке 25 представлены математическое ожидание и среднеквадратическое отклонение обслуживания заявки при работе 9 кассиров.

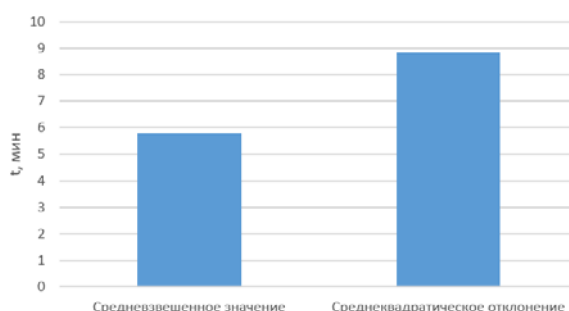


Рисунок 25 – Средневзвешенное значение и среднеквадратическое отклонение при работе 9 кассиров

Далее представлена эмпирическая функция распределения времени обработки заявок (рисунок 26).



Рисунок 26 – Эмпирическая функция распределения при работе 9 кассиров

Второй эксперимент производится при следующих значениях среднего интервала времени поступления заявок на обслуживание пассажиров в минутах:

- покупка билетов – 6,
- бронирование билетов – 9,
- выкуп брони – 10,
- снятие брони – 25,
- возврат билета – 120,
- подтверждение бронирования билетов – 3.

Всего поступает примерно 2500 заявок в неделю.

Результаты моделирования указаны в таблице 9.

Таблица 9 – Результаты моделирования обслуживания пассажиров кассирами (эксперимент 2)

Количество кассиров	Результат
7	Более 6% заявок обслуживаются более 20 мин. Средняя очередь 0,3.
6	Более 6% заявок обслуживаются более 20 мин. Средняя очередь 0,5.
5	Более 19% заявок обслуживаются более 20 мин. Средняя очередь 4,5.
4	Более 60% заявок обслуживаются более 20 мин. Средняя очередь 18,7.

Оптимальным является вариант с 6 кассирами.

Третий эксперимент производится при следующих значениях среднего интервала времени поступления заявок на обслуживание пассажиров в минутах:

- покупка билетов – 9,
- бронирование билетов – 13,
- выкуп брони – 15,
- снятие брони – 37,
- возврат билета – 180,
- подтверждение бронирования билетов – 4.

Всего поступает примерно 1800 заявок в неделю.

Результаты моделирования указаны в таблице 10.

Таблица 10 – Результаты моделирования обслуживания пассажиров кассирами (эксперимент 3)

Количество кассиров	Результат
6	Более 5% заявок обслуживаются более 20 мин. Средняя очередь 0,1.
5	Более 8% заявок обслуживаются более 20 мин. Средняя очередь 0,6.
4	Более 9% заявок обслуживаются более 20 мин. Средняя очередь 1,5.
3	Более 32% заявок обслуживаются более 20 мин. Средняя очередь 6,3.

Оптимальным является вариант с 5 кассирами.

Q-схема имитационной модели обработки запросов сервером для автоматизированной системы обслуживания авиапассажиров представлена на рисунке 27.

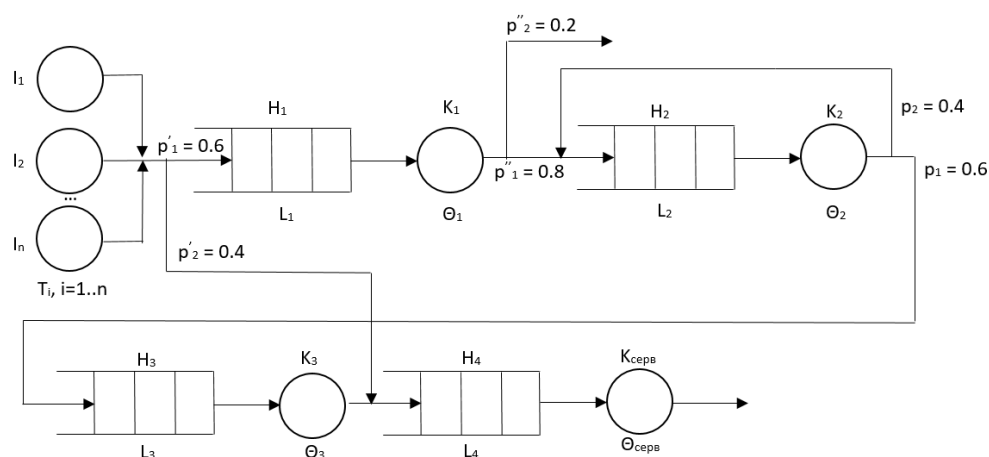


Рисунок 27 – Q-схема имитационной модели обработки запросов сервером

В данной модели поток заявок в систему осуществляется источниками $I_1 - I_n$, где n – число разновидностей обслуживания.

Затем прибывшая заявка с вероятностью $p'_2 = 0.4$ (вид обслуживания не покупка или бронирование билетов) поступает сразу на обработку сервером.

С вероятностью же $p'_1 = 0.6$ (вид обслуживания покупка или бронирование билетов) происходит переход сначала в канал обслуживания K_1 , в котором осуществляется просмотр рейсов на заданное направление. Далее с вероятностью $p''_2 = 0.2$ заявка покидает систему (не найден подходящий рейс). Иначе она обслуживается каналом K_2 , который отвечает за добавление в систему пассажира. Затем с вероятностью $p_2 = 0.4$ осуществляется добавление еще одного пассажира в K_2 и т.д. С вероятностью же $p_1 = 0.6$ происходит выход из данного цикла и переход к каналу K_3 (добавление покупателя в систему).

Далее осуществляется обработка запроса сервером в канале обслуживания $K_{серв}$.

Исходные количественные данные (интервалы поступления заявок, время обработки заявок) для модели были получены экспериментальным способом путем изучения статистики по использованию приложений обслуживания авиапассажиров, по типам обслуживания, а также по быстродействию современных серверов.

Первый эксперимент производится при следующих значениях среднего интервала времени поступления заявок в систему в секундах:

- запрос на покупку билетов – 20,
- запрос на бронирование билетов – 34,
- запрос на выкуп брони – 37,
- запрос на снятие брони – 83,
- запрос на возврат билета – 115.

Всего поступает примерно 77000 заявок в неделю.

Результаты моделирования указаны в таблице 11.

Таблица 11 – Результаты моделирования обработки запросов сервером (эксперимент 1)

Время на обработку запроса сервером, с	Среднее общее время обработки запроса, с
1	1,1
2	2,6
3	4,5
4	7,2

По результатам данного моделирования была составлена гистограмма частот времени пребывания одной заявки в системе (рисунок 28), средневзвешенное значение и среднеквадратичное отклонение времени обработки заявки (рисунок 29), а также эмпирическая функция распределения времени обработки заявки (рисунок 30) при времени на обработку запроса в 1 секунду.

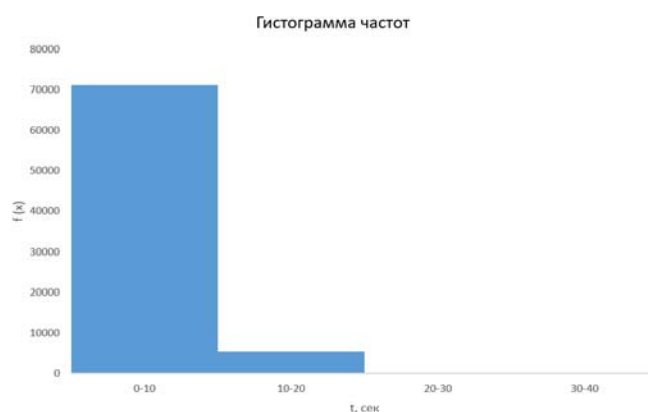


Рисунок 28 – Гистограмма частот времени пребывания заявки в системе при времени на обработку запроса в 1 секунду

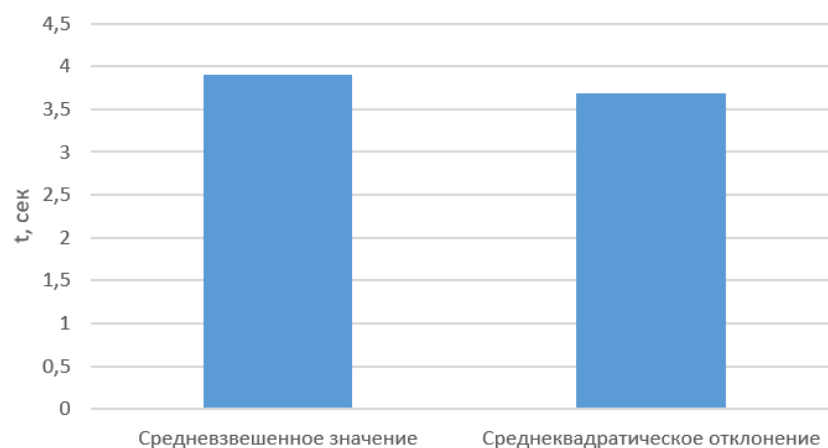


Рисунок 29 – Средневзвешенное значение и среднеквадратическое отклонение при времени на обработку запроса в 1 секунду



Рисунок 30 – Эмпирическая функция распределения при времени на обработку запроса в 1 секунду

Второй эксперимент производится при следующих значениях среднего интервала времени поступления заявок в систему в секундах:

- запрос на покупку билетов – 30,
- запрос на бронирование билетов – 51,
- запрос на выкуп брони – 55,
- запрос на снятие брони – 124,
- запрос на возврат билета – 172.

Всего поступает примерно 51000 заявок в неделю.

Результаты моделирования указаны в таблице 12.

Таблица 12 – Результаты моделирования обработки запросов сервером
(эксперимент 2)

Время на обработку запроса сервером, с	Среднее общее время обработки запроса, с
1	1,1
2	2,3
3	3,8
4	5,7

Третий эксперимент производится при следующих значениях среднего интервала времени поступления заявок в систему в секундах:

- запрос на покупку билетов – 45,
- запрос на бронирование билетов – 76,
- запрос на выкуп брони – 82,
- запрос на снятие брони – 186,
- запрос на возврат билета – 258.

Всего поступает примерно 34000 заявок в неделю.

Результаты моделирования указаны в таблице 13.

Таблица 13 – Результаты моделирования обработки запросов сервером
(эксперимент 3)

Время на обработку запроса сервером, с	Среднее общее время обработки запроса, с
1	1,1
2	2,2
3	3,5
4	5,0

Оптимальное время обработки запроса сервером составляет 1 секунду, так как при больших значениях время получения ответа составляет более 2 секунд, что некомфортно для пользователя.

2.3 Проверка адекватности имитационных моделей

Проверим адекватность имитационных моделей путем проведения их валидации и верификации [51, 52]. Для осуществления валидации возьмем модель обслуживания пассажиров кассирами при поступлении 3800 заявок в неделю.

Рассчитаем ряд теоретических значений для рассматриваемой модели, а затем сравним их с полученными результатами в среде GPSS Studio [53].

Интенсивность входящего потока:

$$\lambda = \frac{3800}{3360} = 1,131. \quad (1)$$

Коэффициент загрузки системы:

$$\gamma = \frac{\lambda}{N\mu} = \frac{1,131}{9*0,192} = 0,655, \quad (2)$$

где μ – интенсивность потока обслуживания,

N – количество кассиров (в данном случае $N = 9$).

Среднее число занятых администраторов:

$$\bar{N}_{\text{зан}} = \alpha = \frac{\lambda}{\mu} = \frac{1,131}{0,192} = 5,893. \quad (3)$$

Среднее число свободных администраторов:

$$\bar{N}_{\text{св}} = N - \bar{N}_{\text{зан}} = 9 - 5,893 = 3,107. \quad (4)$$

Вероятность задействия всех администраторов без возникновения очереди:

$$P_N = P_9 = \frac{\frac{\alpha^9}{9!}}{\sum_{j=0}^9 \frac{\alpha^j}{j!} + \frac{\alpha^9}{9!} * \frac{\gamma}{1-\gamma}} = \frac{\frac{5,893^9}{9!}}{\sum_{j=0}^9 \frac{5,893^j}{j!} + \frac{5,893^9}{9!} * \frac{0,655}{1-0,655}} = 0,062. \quad (5)$$

Средняя длина очереди:

$$\bar{r} = P_N * \frac{\gamma}{(1-\gamma)^2} = 0,062 * \frac{0,655}{(1-0,655)^2} = 0,342. \quad (6)$$

Среднее время пребывания заявок в очереди:

$$\bar{T}_{оч} = \frac{\bar{r}}{\lambda} = \frac{0,342}{1,131} = 0,302. \quad (7)$$

Среднее число заявок в системе:

$$\bar{N}_{сис} = \bar{N}_{зан} + \bar{r} = 5,893 + 0,342 = 6,235. \quad (8)$$

Среднее время пребывания заявок в системе:

$$\bar{T}_{сис} = \frac{\bar{N}_{сис}}{\lambda} = \frac{6,235}{1,131} = 5,513. \quad (9)$$

Теперь сравним некоторые из полученных теоретических значений показателей с практическими значениями, полученными в процессе моделирования (таблица 14).

Таблица 14 – Сравнение теоретических и практических значений характеристик модели

Показатель	Расчетное значение	Модельное значение	Расхождение
Коэффициент загрузки системы	0,655	0,654	0,2%
Среднее число свободных кассиров	3,107	3,114	0,2%
Среднее время пребывания заявки в системе	5,513	5,791	5%

Таким образом, в результате валидации имитационной модели обслуживания пассажиров кассирами было определено, что расхождение результатов теоретического и практического значений составляет не более 10%. Это подтверждает адекватность и корректность работы модели.

Аналогично валидация была проведена и для второй модели, в результате чего было получено расхождение менее 10%.

Теперь выполним верификацию созданных моделей, которая предполагает проверку на тех участках, где ошибки могут возникнуть легче всего. В разработанных моделях таким участком является место, где различные типы запросов отделяются друг из друга, позволяя вести статистику как по всем типам обслуживания в целом, так и для каждого типа в отдельности.

Для проверки корректности работы модели используется режим трассировки в GPSS Studio. Результаты проверок представлены ниже на рисунках 31 – 34.

The screenshot shows the GPSS Studio interface. On the left, the model code is displayed, with the 'GENERATE 6.FN\$XPDIS : Бронирование билетов(2)' line highlighted in yellow. On the right, the 'Run' window is open, showing the current time (2.56) and the current transaction (2 / 2). The 'Parameter transaction' table is empty.

Параметр транзакта	Значение
--------------------	----------

Рисунок 31 – Поступление заявки в модели обслуживания пассажиров кассирами (бронирование билетов)

The screenshot shows the GPSS Studio interface. On the left, the model code is displayed, with the 'TABULATE Tab2 : Время бронирования билетов (2)' line highlighted in yellow. On the right, the 'Run' window is open, showing the current time (8.44) and the current transaction (2 / 2). The 'Parameter transaction' table has two rows.

Параметр транзакта	Значение
1	2
2	10

Рисунок 32 – Результаты обработки заявки в модели обслуживания пассажиров кассирами (бронирование билетов)

The screenshot shows the GPSS Studio interface. On the left, the model code is displayed, with the 'GENERATE 20.FN\$XPDIS : Покупка билетов(1)' line highlighted in yellow. On the right, the 'Run' window is open, showing the current time (18.27) and the current transaction (1 / 1). The 'Parameter transaction' table is empty.

Параметр транзакта	Значение
--------------------	----------

Рисунок 33 – Поступление заявки в модели обработки запроса сервером (покупка билетов)

	ADVANCE RELEASE	1.FN\$XPDIS K3	: Добавление покупателя
work	QUEUE ENTER DEPART ADVANCE LEAVE	QH4 Server QH4 1.FN\$XPDIS Server	: Работа сервера
type	TABULATE	Tab	
	TEST E	P1,1,type2	
	TABULATE	Tab1	: Время покупки билетов(1)
	TRANSFER	.finish	
type2	TEST E	P1,2,type3	: Время бронирования билетов(2)
	TABULATE	Tab2	
	TRANSFER	.finish	
type3	TEST E	P1,3,type4	: Время выкупа брони(3)
	TABULATE	Tab3	
	TRANSFER	.finish	

Текущее время (AC1): 25,60
Текущий транзакт (XN1 / A1): 1 / 1
[Редактировать](#)

Параметр транзакта	Значение
1	1

[Редактировать](#)

Переменная / матрица	Значение
----------------------	----------

Рисунок 34 – Результаты обработки заявки в модели обработки запроса сервером (покупка билетов)

Как можно увидеть, каждая из заявок была обработана в соответствии со своим типом. Аналогично были проверены и другие типы заявок. При этом несовпадений обнаружено не было. Следовательно, обе модели работают корректно.

Таким образом, результаты валидации и верификации разработанных имитационных моделей подтвердили их адекватность.

2.4 Анализ требований

2.4.1 Анализ функциональных требований

Основными пользователями разрабатываемой системы обслуживания авиапассажиров являются кассир аэропорта и клиент. Они, работая с системой, должны, в первую очередь, иметь возможность оформления бронирования и продажи билета и, при необходимости, заносить информацию о пассажирах и покупателе в систему. Также им должна быть предоставлена возможность входа в систему и выхода из нее, и изменение данных пассажира или покупателя. Помимо этого, пользователям должна быть предоставлена возможность оформления как одиночного, так и множественного выкупа и снятия броней и возврата билетов. Еще для кассира должен быть обеспечен просмотр аналитической информации (средняя загруженность рейсов по дням и количество совершенных полетов пассажирами по месяцам) и возможность подтверждения броней. Для пассажира необходимо обеспечить просмотр списка оформленных билетов и

броней, а также вывод информации по наиболее предпочтительным направлениям для него.

На основе всего вышесказанного были созданы две диаграммы вариантов использования, представленные на рисунках 35 и 36 [54].

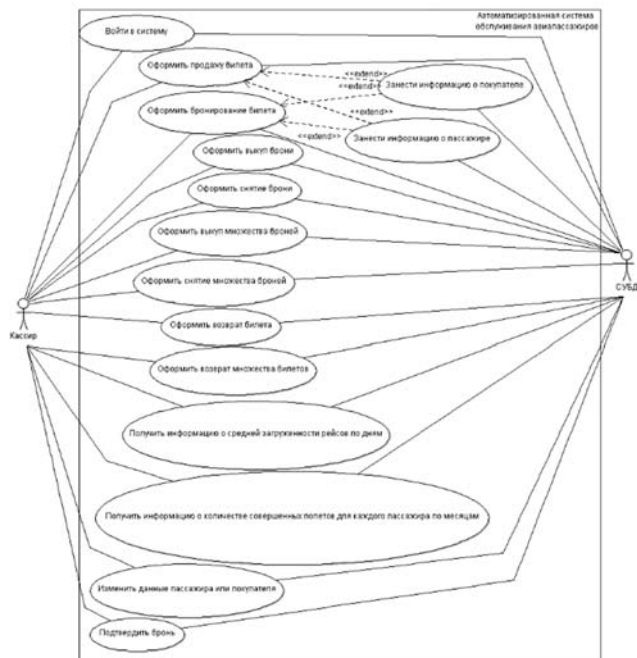


Рисунок 35 – Диаграмма вариантов использования автоматизированной системы обслуживания авиапассажиров для актера «Кассир»

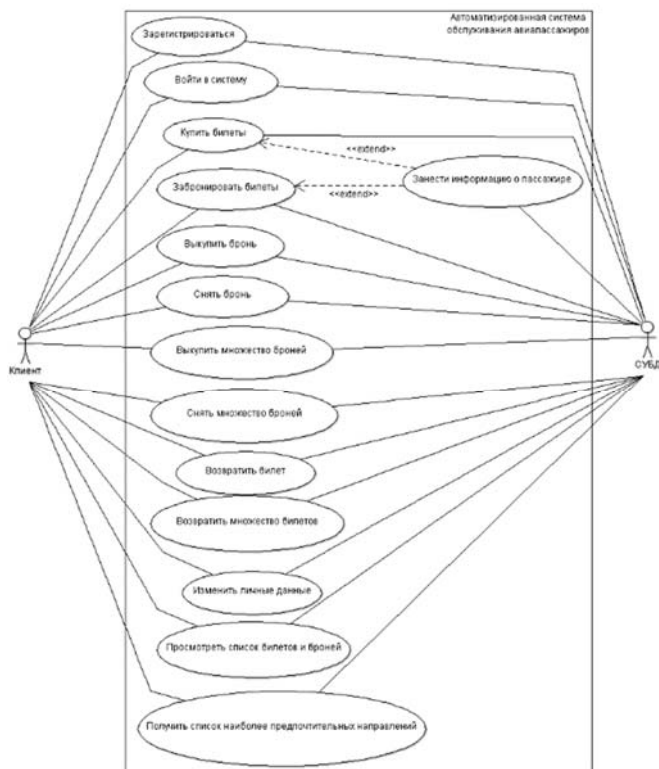


Рисунок 36 – Диаграмма вариантов использования автоматизированной системы обслуживания авиапассажиров для актера «Клиент»

Составим спецификации для основных вариантов использования: «Оформить продажу билета» для актера «Кассир» (рисунок 37), «Выкупить множество броней» для актера «Клиент» (рисунок 38), «Возвратить билет» для актера «Клиент» (рисунок 39).

Наименование: оформить продажу билета
ID: 1
Краткое описание: кассир продает билет на выбранный рейс
Действующие лица: кассир, СУБД
Предусловие: покупатель обратился к кассиру для покупки билета
Основной поток: <ol style="list-style-type: none"> 1. Кассир вводит дату рейса 2. Кассир вводит направление перелета 3. Система ищет информацию о рейсах по дате и направлению перелета 4. Система отображает найденную информацию о рейсах 5. Кассир выбирает рейс 6. ЕСЛИ информации о покупателе нет в БД 6.1. Кассир вводит информацию о покупателе 7. ИНАЧЕ 7.1. Кассир вводит номер паспорта покупателя 8. Система заносит информацию о покупателе в БД 9. ЕСЛИ информации о пассажирах нет в БД 9.1. Кассир вводит информацию о пассажирах 10. ИНАЧЕ 10.1. Кассир вводит номер паспорта пассажира 11. Кассир вводит номер места пассажира в самолете 12. Система заносит информацию о пассажире в БД 13. Кассир инициирует операцию продажи билета 14. Система заносит информацию о продаже в БД и сообщает об этом кассиру
Постусловие: билет был продан и занесен в БД

Рисунок 37 – Спецификация варианта использования «Оформить продажу билета» для актера «Кассир»

Наименование: выкупить множество броней
ID: 2
Краткое описание: клиент выкупает несколько броней
Действующие лица: клиент, СУБД
Предусловие: клиент вошел в систему
Основной поток: <ol style="list-style-type: none"> 1. Клиент вводит номер рейса 2. Система ищет информацию по номеру рейса и номеру паспорта клиента в БД 3. ЕСЛИ брони найдены 3.1. Система отображает найденную информацию клиенту 3.2. Клиент инициирует операцию выкупа броней 3.3. Система корректирует информацию в БД 4. ИНАЧЕ 4.1. Система сообщает клиенту, что броней с заданными параметрами в БД не найдены
Постусловие: множество броней было выкуплено

Рисунок 38 – Спецификация варианта использования «Выкупить множество броней» для актера «Клиент»

Наименование: вернуть билет
ID: 3
Краткое описание: клиент осуществляет возврат билета
Действующие лица: клиент, СУБД
Предусловие: клиент вошел в систему
Основной поток: <ol style="list-style-type: none"> 1. Клиент вводит номер билета 2. Система ищет информацию по номеру билета в БД 3. ЕСЛИ билет найден И он был оформлен клиентом <ol style="list-style-type: none"> 3.1. Система отображает найденную информацию клиенту 3.2. Клиент инициирует операцию возврата билета 3.3. Система корректирует информацию в БД 4. ИНАЧЕ <ol style="list-style-type: none"> 4.1. Система сообщает клиенту, что билет с заданным номером в БД не найден или он оформлен другим пользователем
Постусловие: билет был возвращен

Рисунок 39 – Спецификация варианта использования «Возвратить билет» для актера «Клиент»

Помимо спецификаций для некоторых прецедентов были составлены диаграмма деятельности и диаграмма последовательности. Для варианта использования «Войти в систему» составлена диаграмма деятельности, представленная на рисунке 40.

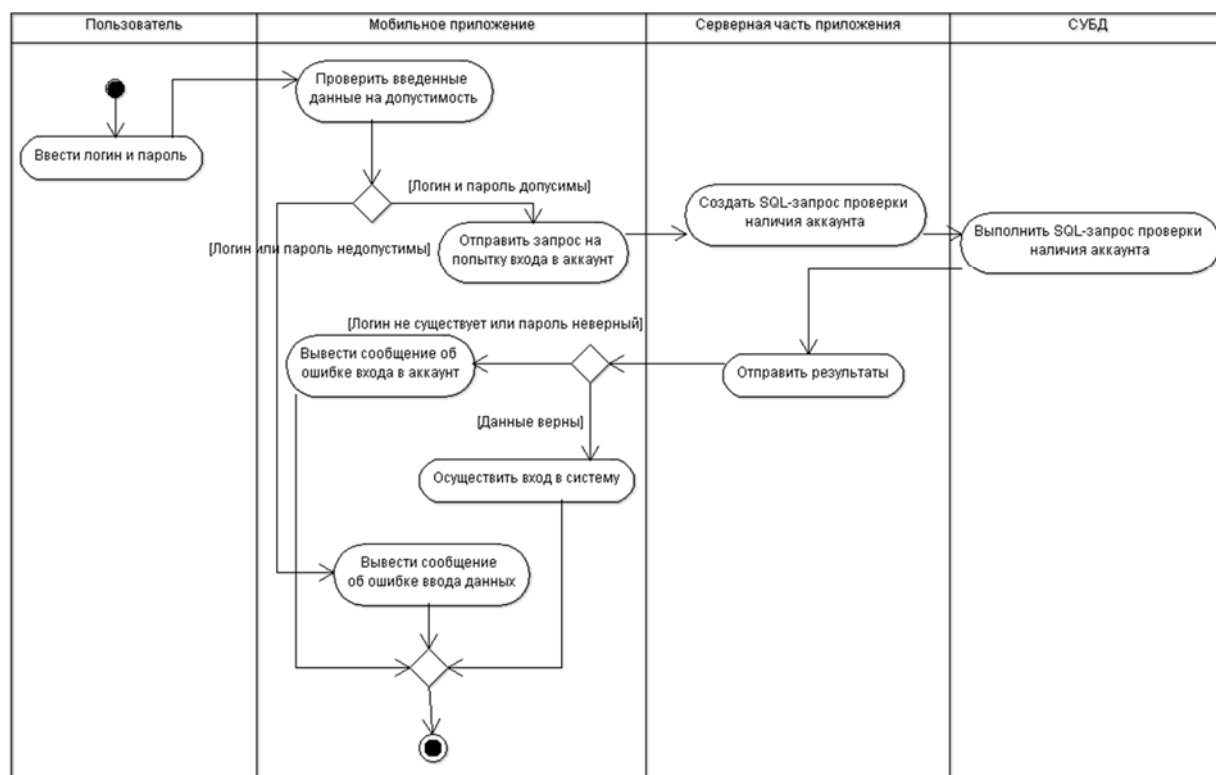


Рисунок 40 – Диаграмма деятельности варианта использования «Войти в систему»

Для варианта использования «Занести информацию о пассажире» была составлена диаграмма последовательности, приведенная на рисунке 41.

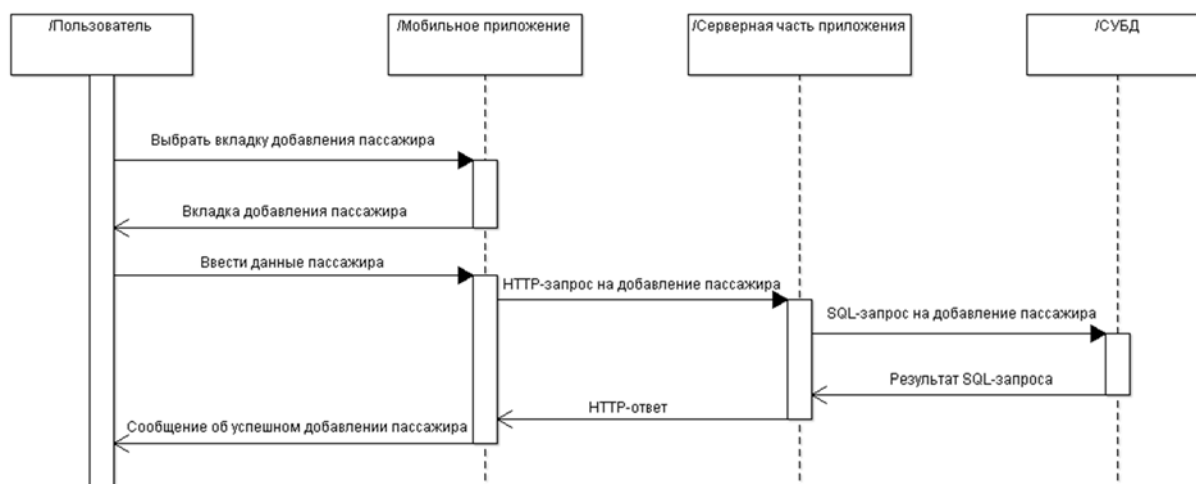


Рисунок 41 – Диаграмма последовательности варианта использования «Занести информацию о пассажире»

На данном этапе были определены основные функции разрабатываемой системы обслуживания авиапассажиров. При этом для данной системы были построены диаграммы вариантов использования, деятельности и последовательности, а также для некоторых вариантов использования были составлены спецификации.

2.4.2 Анализ нефункциональных требований

На данном этапе были определены нефункциональные требования к разрабатываемому программному обеспечению. Должны быть использованы такие технологии разработки, как RUP, ООП, MVC, MVVM [26, 55].

Серверная часть приложения должна быть разработана на языке программирования C# на платформе ASP.NET в среде программирования Microsoft Visual Studio 2019 [56]. Данные на сервере должны храниться в базе данных, реализованной с использованием СУБД MS SQL Server 2019, запросы к базе данных должны осуществляться на языке SQL с использованием фреймворка Entity Framework Core [57]. Серверная часть должна работать под управлением операционной системы Windows 10 и выше.

Клиентская часть приложения должна быть реализована на языке программирования C# в среде программирования Microsoft Visual Studio 2019. Она должна представлять собой мобильное приложение с удобным и интуитивно понятным пользовательским интерфейсом. Клиентское приложение должно быть отлажено и должно успешно работать под управлением операционной системы Android 9.0 и выше.

Для реализации рекомендательной системы должен применяться язык программирования Python с использованием библиотек Scikit-learn, SciPy и NetworkX и использоваться среда программирования Spyder. Запуск рекомендательной системы должен осуществляться под управлением операционной системы Windows 10 и выше.

2.5 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был спроектирован интерфейс мобильного приложения. Он состоит из 18 страниц, 4 из которых являются основными вкладками. Они располагаются на странице входа: «Продажа» для оформления бронирования и продажи билетов; «Выкуп брони» для оформления снятия и выкупа брони; «Возврат билет» для возврата купленных билетов; «Другое» для получения аналитической информации, изменения данных клиентов, подтверждения бронирования и просмотра списка билетов и брони.

Макет интерфейса страницы авторизации представлен на рисунке 42.

Вход

1 — Логин

2 — Пароль

3 — Войти

4 — Нет аккаунта? Зарегистрироваться

Рисунок 42 – Страница авторизации

На макете страницы авторизации имеются следующие элементы:

- 1 – поле ввода логина,
- 2 – поле ввода пароля,
- 3 – кнопка входа, по нажатию на которую осуществляется попытка авторизации,
- 4 – кнопка, по нажатию на которую осуществляется переход на страницу регистрации.

Макет интерфейса страницы продажи представлен на рисунке 43.

The wireframe shows a flight booking interface. At the top, there are three input fields labeled 'Откуда' (1), 'Куда' (2), and 'Дата вылета' (3). Below these is a button labeled 'Вывести список рейсов' (4). Underneath the button is a section titled 'Рекомендованные направления:' containing a list box with three items: 'Направление 1', 'Направление 2', and 'Направление 3' (5). At the bottom of the page is a horizontal navigation bar with four buttons: 'Продажа', 'Выкуп брони', 'Возврат билета', and 'Другое'.

Рисунок 43 – Страница продажи

На макете страницы продажи имеются следующие элементы:

- 1 – поле выбора города отправления рейса,
- 2 – поле выбора города прибытия рейса,
- 3 – поле выбора даты рейса,
- 4 – кнопка вывода списка рейсов, удовлетворяющих заданным критериям,
- 5 – список рекомендованных авианаправлений, который был получен в результате работы рекомендательной системы.

Макет интерфейса страницы выбора рейса представлен на рисунке 44.

Рисунок 44 – Страница выбора рейса

На макете страницы авторизации имеются следующие элементы:

- 1 – кнопка «Назад», по нажатию на которую осуществляется возврат на предыдущую страницу,
- 2 – результаты поиска рейсов по заданным критериям,
- 3 – кнопка перехода к оформлению билетов, по нажатию на которую осуществляется переход на страницу покупки/бронирования билетов.

Макет интерфейса страницы покупки/бронирования билетов представлен на рисунке 45.

Рисунок 45 – Страница покупки/бронирования билетов

На макете страницы покупки/бронирования билетов имеются следующие элементы:

1 – кнопка «Назад», по нажатию на которую осуществляется возврат на предыдущую страницу,

2 – кнопка добавления пассажира, по нажатию на которую осуществляется переход на страницу добавления пассажира,

3 – список с информацией о добавленных пассажирах,

4 – кнопка добавления покупателя, по нажатию на которую осуществляется переход на страницу добавления покупателя,

5 – надпись с информацией о добавленном покупателе,

6 – кнопка продажи билетов, по нажатию на которую осуществляется попытка оформления продажи билетов с введенными параметрами и происходит переход на предыдущую страницу,

7 – кнопка бронирования билетов, по нажатию на которую осуществляется попытка оформления бронирования билетов с введенными параметрами и происходит переход на предыдущую страницу.

Макет интерфейса страницы добавления ранее учтенного пассажира представлен на рисунке 46.

Рисунок 46 – Страница добавления ранее учтенного пассажира

На макете страницы добавления ранее учтенного пассажира имеются следующие элементы:

- 1 – поле ввода паспортных данных пассажира,
- 2 – поле выбора места в самолете,
- 3 – кнопка добавления пассажира, по нажатию на которую осуществляется поиск пассажира и добавление его к списку добавленных пассажиров,
- 4 – кнопка «Назад», по нажатию на которую осуществляется возврат на страницу покупки/бронирования билетов.

Макет интерфейса страницы добавления нового пассажира представлен на рисунке 47.

Фамилия	<input type="text"/>	1
Имя	<input type="text"/>	2
Отчество (при наличии)	<input type="text"/>	3
Дата рождения	<input type="text"/>	4
Номер телефона	<input type="text"/>	5
Паспортные данные	<input type="text"/>	6
Номер места	<div></div>	7
<div>Добавить пассажира</div>		8
<div>Назад</div>		9
<div>Ранее учтенный</div>		
<div>Новый</div>		

Рисунок 47 – Страница добавления нового пассажира

На макете страницы добавления нового пассажира имеются следующие элементы:

- 1 – поле ввода фамилии пассажира,
- 2 – поле ввода имени пассажира,
- 3 – поле ввода отчества пассажира,
- 4 – поле ввода даты рождения пассажира,
- 5 – поле ввода номера телефона пассажира,

6 – поле ввода паспортных данных пассажира,

7 – поле выбора места в самолете,

8 – кнопка добавления пассажира, по нажатию на которую осуществляется добавление пассажира в информационную систему и занесение его в список добавленных пассажиров,

9 – кнопка «Назад», по нажатию на которую осуществляется возврат на страницу покупки/бронирования билетов.

Макет интерфейса страницы добавления ранее учтенного покупателя представлен на рисунке 48.

Рисунок 48 – Страница добавления ранее учтенного покупателя

На макете страницы добавления ранее учтенного покупателя имеются следующие элементы:

1 – поле ввода паспортных данных пассажира,

2 – кнопка добавления покупателя, по нажатию на которую осуществляется поиск покупателя и его выбор,

3 – кнопка «Назад», по нажатию на которую осуществляется возврат на страницу покупки/бронирования билетов.

Макет интерфейса страницы добавления нового покупателя представлен на рисунке 49.

Фамилия

Имя

Отчество (при наличии)

Дата рождения

Номер телефона

Паспортные данные

Добавить пассажира

Назад

Ранее учтенный

Новый

Рисунок 49 – Страница добавления нового покупателя

На макете страницы добавления нового покупателя имеются следующие элементы:

- 1 – поле ввода фамилии покупателя,
- 2 – поле ввода имени покупателя,
- 3 – поле ввода отчества покупателя,
- 4 – поле ввода даты рождения покупателя,
- 5 – поле ввода номера телефона покупателя,
- 6 – поле ввода паспортных данных покупателя,
- 7 – кнопка добавления покупателя, по нажатию на которую осуществляется добавление покупателя в информационную систему и его выбор,
- 8 – кнопка «Назад», по нажатию на которую осуществляется возврат на страницу покупки/бронирования билетов.

Макет интерфейса страницы выкупа брони представлен на рисунке 50.

Продажа			
Выкуп брони			
Возврат билета			
Другое			

Рисунок 50 – Страница выкупа брони

На макете страницы выкупа брони имеются следующие элементы:

- 1 – поле ввода номера брони,
- 2 – кнопка снятия брони, по нажатию на которую осуществляется сброс брони с введенным номером,
- 3 – кнопка выкупа брони, по нажатию на которую осуществляется выкуп брони с введенным номером,
- 4 – поле ввода номера рейса,
- 5 – поле ввода паспортных данных оформителя броней,
- 6 – кнопка снятия множества броней, по нажатию на которую осуществляется сброс одной или нескольких броней на основе введенных данных,
- 7 – кнопка выкупа множества броней, по нажатию на которую осуществляется выкуп одной или нескольких броней на основе введенных данных.

Макет интерфейса страницы возврата билета представлен на рисунке 51.

Рисунок 51 – Страница возврата билета

На макете страницы возврата билета имеются следующие элементы:

- 1 – поле ввода номера билета,
- 2 – кнопка оформления возврата билета, по нажатию на которую осуществляется обнуление билета,
- 3 – поле ввода номера рейса,
- 4 – поле ввода паспортных данных покупателя билетов,
- 5 – кнопка оформления возврата множества билетов, по нажатию на которую осуществляется обнуление одного или нескольких билетов на основе введенных данных.

Макет интерфейса страницы «Другое» представлен на рисунке 52.

Рисунок 52 – Страница «Другое»

На макете страницы «Другое» имеются следующие элементы:

1 – кнопка, по нажатию на которую открывается страница с информацией о средней загруженности рейсов по дням,

2 – кнопка, по нажатию на которую открывается страница с информацией о количестве совершенных полетов для каждого пассажира по месяцам,

3 – кнопка, по нажатию на которую открывается страница выбора пассажира или покупателя,

4 – кнопка, по нажатию на которую открывается страница подтверждения бронирования,

5 – кнопка, по нажатию на которую открывается страница просмотра списка билетов и броней,

6 – кнопка выхода из аккаунта, по нажатию на которую происходит переход на страницу авторизации.

Макет интерфейса страницы выбора пассажира или покупателя представлен на рисунке 53.

Выбор пассажира или покупателя			
Паспортные данные			
Изменить данные пассажира или покупателя			
Продажа	Выкуп брони	Возврат билета	Другое

Рисунок 53 – Страница выбора пассажира или покупателя

На макете страницы выбора пассажира или покупателя имеются следующие элементы:

1 – кнопка «Назад», по нажатию на которую осуществляется возврат на предыдущую страницу,

2 – поле ввода паспортных данных пассажира или покупателя,

3 – кнопка, по нажатию на которую происходит поиск пассажира или покупателя и осуществляется переход на страницу изменения данных пассажира или покупателя.

Макет интерфейса страницы изменения данных пассажира или покупателя представлен на рисунке 54.

The screenshot shows a mobile application interface for changing passenger or buyer data. At the top, there is a title bar with a back arrow (1) and the text 'Изменение данных пассажира или покупателя'. Below the title bar, there are several input fields with labels: 'Фамилия' (2), 'Имя' (3), 'Отчество (при наличии)' (4), 'Дата рождения' (5), 'Номер телефона' (6), 'Паспортные данные' (7), 'Пляжный отдых' (8), 'Горные лыжи' (9), and 'Исторические места' (10). Each field is represented by a text box. Below these fields is a button labeled 'Изменить данные пассажира или покупателя' (11). At the bottom of the screen, there is a navigation bar with four buttons: 'Продажа', 'Выкуп брони', 'Возврат билета', and 'Другое'.

Рисунок 54 – Страница изменения данных пассажира или покупателя

На макете страницы изменения данных пассажира или покупателя имеются следующие элементы:

1 – кнопка «Назад», по нажатию на которую осуществляется возврат на предыдущую страницу,

2 – поле ввода фамилии пассажира или покупателя,

3 – поле ввода имени пассажира или покупателя,

4 – поле ввода отчества пассажира или покупателя,

5 – поле выбора даты рождения пассажира или покупателя,

6 – поле ввода номера телефона пассажира или покупателя,

7 – поле ввода паспортных данных пассажира или покупателя,

8 – поле ввода интересов пляжного отдыха пассажира или покупателя по десятибалльной шкале,

9 – поле ввода интересов горных лыж пассажира или покупателя по десятибалльной шкале,

10 – поле ввода интересов исторических мест пассажира или покупателя по десятибалльной шкале,

11 – кнопка, по нажатию на которую осуществляется сохранение введенных данных возврат на предыдущую страницу.

2.6 Проектирование базы данных

2.6.1 Концептуальное проектирование базы данных

На основе анализа предметной области была построена концептуальная модель базы данных. Графическое представление концептуальной модели БД в виде диаграммы «сущность-связь» представлено на рисунке 55.

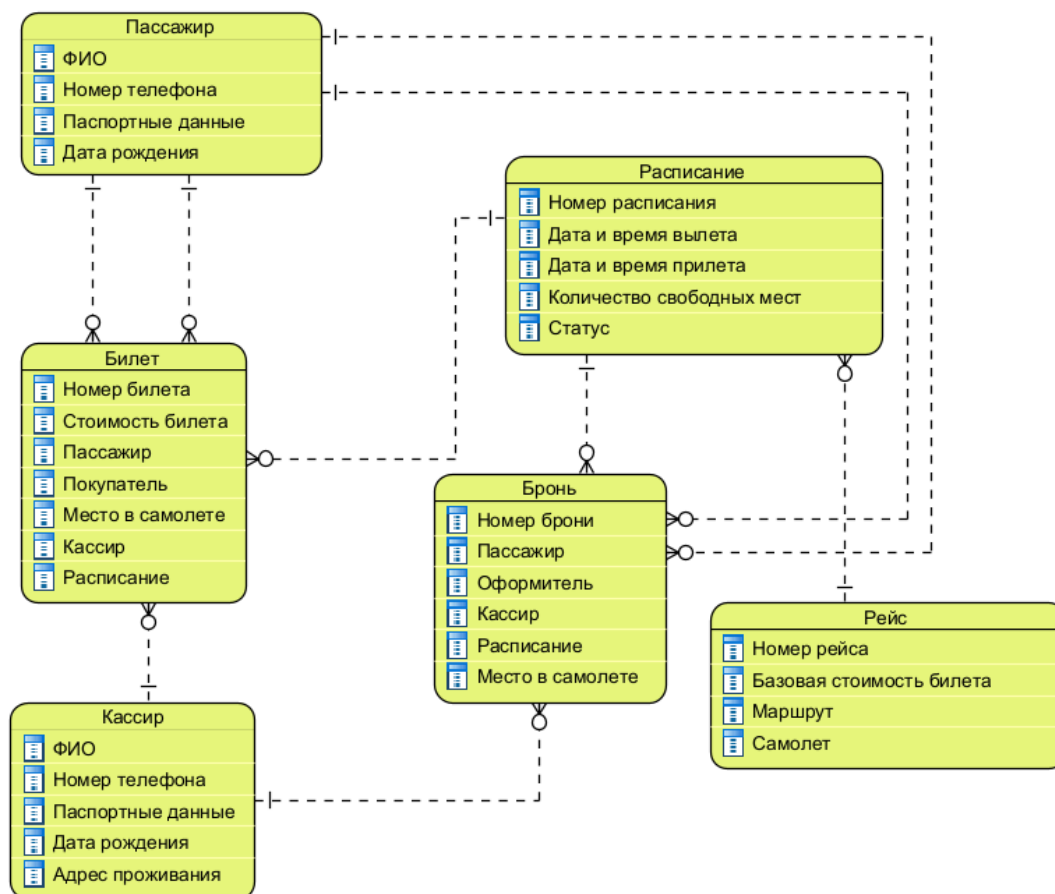


Рисунок 55 – Графическое представление концептуальной модели базы данных

Сущности концептуальной модели данных представлены в таблице 15.

Таблица 15 – Сущности концептуальной модели

№	Название	Описание	Количество экземпляров	Режим обновления
1	Пассажир	Список пассажиров и покупателей	1000000	При покупке билетов или оформлении брони клиентом, информация о котором отсутствует в системе
2	Рейс	Список рейсов	5000	При добавлении нового рейса
3	Расписание	Список расписаний рейсов	100000	При добавлении нового расписания рейса
4	Кассир	Список кассиров	100	При принятии на работу нового кассира
5	Билет	Список билетов	30000000	При продаже билета
6	Бронь	Список броней	30000000	При бронировании билета

В таблице 16 представлено описание связей между сущностями.

Таблица 16 – Описание связей между сущностями концептуальной модели

№	Связь	Описание	Мощность	Обязательность
1	Пассажир – Билет	Определяет билеты пассажира	1:M	Да
2	Пассажир – Билет	Определяет купленные билеты покупателя	1:M	Да
3	Пассажир – Бронь	Определяет брони пассажира	1:M	Да
4	Пассажир – Бронь	Определяет оформленные брони оформителя	1:M	Да
5	Рейс – Расписание	Определяет расписания для рейса	1:M	Нет
6	Расписание – Билет	Определяет билеты, проданные на расписание	1:M	Нет
7	Расписание – Бронь	Определяет брони, оформленные на расписание	1:M	Нет
8	Кассир – Билет	Определяет билеты, проданные кассиром	1:M	Нет
9	Кассир – Бронь	Определяет брони, оформленные кассиром	1:M	Нет

Таким образом, были спроектированы основные сущности с атрибутами и связями между ними, без включения атрибутов первичных и внешних ключей.

2.6.2 Логическое проектирование базы данных

На основании концептуальной модели была спроектирована логическая модель базы данных, представленная на рисунке 56.

На данной логической модели структура базы данных отображена уже более подробно. Здесь определены первичные и внешние ключи, ограничения атрибутов сущностей, а также уточнен их состав и связи между ними.

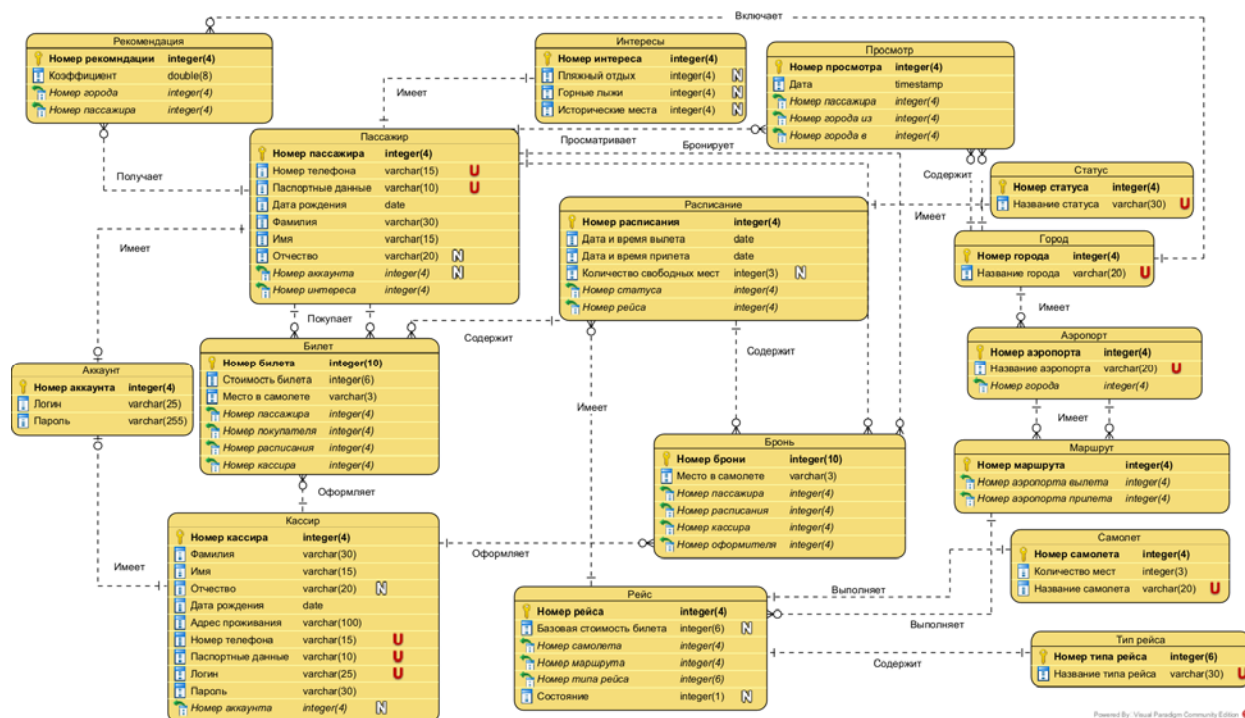


Рисунок 56 – Графическое представление логической модели базы данных

Сущности логической модели данных, не определенные в концептуальной модели, представлены в таблице 17.

Таблица 17 – Сущности логической модели

№	Название	Описание	Количество экземпляров	Режим обновления
1	Тип рейса	Список типов рейса	20	При добавлении новых типов рейса
2	Самолет	Список самолетов	100	При добавлении нового самолета
3	Маршрут	Список авиамаршрутов	5000	При добавлении нового маршрута
4	Аэропорт	Список аэропортов	2000	При добавлении нового аэропорта
5	Город	Список городов	1500	При добавлении нового города
6	Статус	Список статусов авиарейса	10	При добавлении нового статуса
7	Аккаунт	Список аккаунтов	1000000	При регистрации нового пользователя
8	Рекомендация	Список рекомендаций пассажира	5000000	При выполнении генерации новых рекомендаций
9	Интересы	Список интересов пассажиров	1000000	При указании пользователем своих интересов
10	Просмотр	Список просмотренных направлений	5000000	При осуществлении просмотра пассажиром рейса

Также в таблице 18 представлено описание связей между сущностями логической модели, не определенных в результате концептуального моделирования.

Таблица 18 – Описание связей между сущностями логической модели

№	Связь	Описание	Мощность	Обязательность
1	Рейс – Тип рейса	Определяет тип для рейса	1:1	Да
2	Рейс – Самолет	Определяет самолет, которым выполняется рейс	1:1	Да
3	Рейс – Маршрут	Определяет маршрут рейса	M:1	Да
4	Аэропорт – Маршрут	Определяет аэропорт вылета для маршрута	1:M	Да
5	Аэропорт – Маршрут	Определяет аэропорт прилета для маршрута	1:M	Да
6	Город – Аэропорт	Определяет аэропорты в городе	1:M	Да
7	Расписание – Статус	Определяет статус расписания	1:1	Да
8	Аккаунт – Кассир	Определяет аккаунт кассира	1:1	Нет
9	Аккаунт – Пассажир	Определяет аккаунт пассажира	1:1	Нет
10	Пассажир – Рекомендация	Определяет рекомендацию для пассажира	1:M	Да
11	Город – Рекомендация	Определяет город для рекомендации	1:M	Да
12	Интересы – Пассажир	Определяет интересы пассажира	1:1	Да
13	Пассажир – Просмотр	Определяет просмотр рейса пассажиром	1:M	Да
14	Город – Просмотр	Определяет город вылета при просмотре	1:M	Да
15	Город – Просмотр	Определяет город прилета при просмотре	1:M	Да

Несмотря на достаточную содержательность данной модели, сущности все еще представлены в абстрактной форме, без привязки к конкретной реализации базы данных.

2.7 Разработка гибридной рекомендательной системы для автоматизированных систем обслуживания авиапассажиров

Разработка эффективной рекомендательной системы включает в себя множество этапов. Рассмотрим общую схему взаимодействий между различными этапами системы, включая сбор данных, их обработку, обучение моделей и предоставление рекомендаций пользователям [58].

В частности, уже имеются рекомендательные системы для подбора релевантных авианаправлений, в которых для создания рекомендаций используется лишь алгоритм k-средних [59] или контентная фильтрация [60]. Также представлены работы, в которых обсуждаются и строятся рекомендательные системы для путешественников, передвигающихся по Тайланду, на основе нескольких методов машинного обучения [61]. В отличие от существующих решений, в данной работе предлагается

принципиально новый гибридный подход, основанный на оригинальной комбинации методов коллаборативной и контентной фильтрации, графовых моделей и машинного обучения.

Основные этапы разработки системы представлены на рисунке 57.



Рисунок 57 – Основные этапы разработки рекомендательной системы

На этапе сбора данных используются разнообразные источники информации:

- история просмотров городов: данные о том, какие города пользователи просматривали до этого в мобильном приложении для покупки и бронирования авиабилетов,
- покупки билетов: информация о купленных билетах, включая города отправления и назначения, даты,
- предварительные бронирования: данные о городах, на которые пользователи предварительно бронировали билеты,
- интересы пользователей: анкеты и профили пользователей, содержащие их предпочтения и интересы (например, пляжный отдых, исторические места и т.д.),
- сезонность: коэффициенты популярности городов в зависимости от текущего сезона,
- популярность: общие коэффициенты популярности городов.

Сбор данных осуществляется путем выполнения SQL-запросов к базе данных аэропорта.

На этапе машинного обучения используются различные модели для прогнозирования предпочтений пользователей:

- коллаборативная фильтрация: метод сингулярного разложения матрицы (SVD) для определения схожести между пользователями и городами,

- модель случайного леса: используется для прогнозирования вероятности выбора конкретного направления на основе профиля пользователя и его истории,

- линейная регрессия: объединяет результаты коллаборативной фильтрации и графовых моделей для создания финальной гибридной модели.

Графовые модели позволяют представить города и перелеты в виде ориентированного взвешенного графа:

- узлы представляют города, а ребра – вероятности перелетов между ними на основе предпочтений пользователя,

- алгоритм PageRank определяет ранжирование городов на основе вероятности посещения каждого города.

Гибридная модель объединяет результаты всех вышеуказанных методов, выполняя следующие этапы:

- ранжирование городов осуществляется на основе коллаборативной фильтрации и модели случайного леса,

- смешивание результатов выполняется с помощью линейной регрессии, которая комбинирует результаты коллаборативной фильтрации, графовых моделей и модели случайного леса.

Взаимодействие рекомендательной системы с мобильным приложением для покупки и бронирования авиабилетов осуществляется через API. Оно обеспечивает:

- получение рекомендаций для конкретного пользователя, основанные на его профиле и активности,

- предоставление обновленной информации о пользователе и его предпочтениях.

Рассмотрим подробно алгоритмы и модели, заложенные в основу оригинальной рекомендательной системы для авиаперелетов. Система

использует комбинацию нескольких методов, включая коллаборативную фильтрацию, графовые модели и машинное обучение. Это обеспечивает более точную и персонализированную выдачу рекомендаций для пользователей.

Коллаборативная фильтрация (Collaborative filtering) – это метод рекомендации, при котором анализируется только реакция пользователей на объекты: оценки, которые выставляют пользователи объектам [62 – 64]. Оценки могут быть как явными (пользователь явно указывает, на сколько «звездочек» он оценивает объект), так и неявными (например, количество просмотров одного ролика). Чем больше оценок собирается, тем точнее получаются рекомендации. В нашем случае используются неявные оценки.

Получается, что пользователи помогают друг другу в фильтрации объектов. Поэтому такой метод называется также совместной фильтрацией.

В нашем случае используется метод сингулярного разложения матрицы (SVD), который позволяет выявить скрытые факторы предпочтений пользователей.

Иллюстрацию работы алгоритма коллаборативной фильтрации можно увидеть на рисунке 58, который визуализирует процесс создания и разложения матрицы «пассажир-город» с помощью SVD.

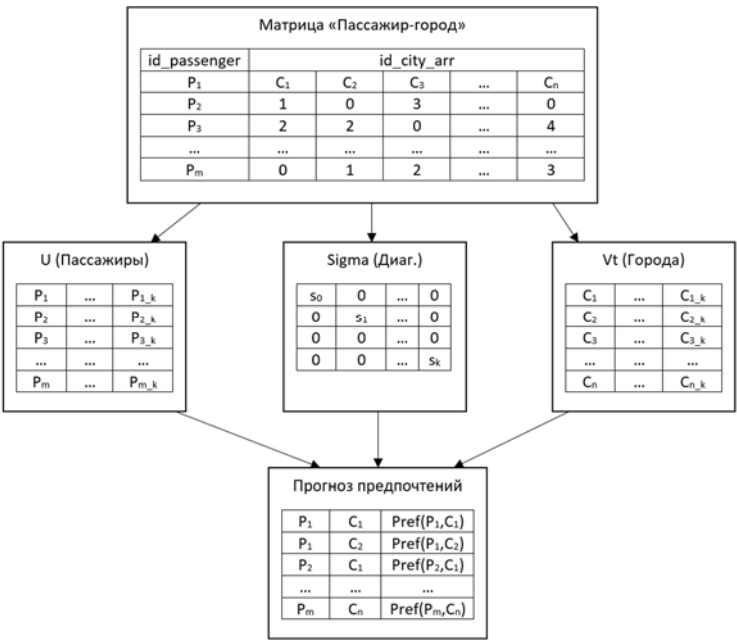


Рисунок 58 – Иллюстрация алгоритма коллаборативной фильтрации

Графовые модели в рекомендательных системах представляют собой мощный инструмент для моделирования и анализа сложных структур данных, где объекты и их взаимосвязи могут быть наглядно представлены в виде узлов и ребер графа [65]. Эти модели позволяют учесть не только характеристики отдельных элементов (например, пользователей или товаров), но и типы и силу их взаимодействий, что делает их особенно ценными для создания персонализированных рекомендаций.

В контексте рекомендательных систем, графовые модели используются для представления и анализа разнообразных данных, таких как социальные сети, пользовательские предпочтения, товарные каталоги и их взаимосвязи. Эти модели способствуют более глубокому пониманию структуры предпочтений и поведения пользователей, позволяя обнаруживать скрытые паттерны и предсказывать потенциальный интерес к определенным товарам или услугам.

В нашем случае графовые модели позволяют представить города и перелеты в виде ориентированного графа, где узлы – это города, а ребра – вероятности перелетов между ними. Веса ребер зависят от предпочтений пользователя и его активности.

Для ранжирования городов в графе предпочтений используется алгоритм PageRank. Он определяет вероятность посещения каждого города на основе случайных блужданий. Идея алгоритма заключается в определении важности веб-страницы, исходя из количества и качества ссылок на нее [66]. Если важная страница ссылается на другую страницу, то последняя также считается важной. Аналогично, в нашем случае PageRank используется для ранжирования городов в графе предпочтений пассажиров.

PageRank формально можно представить следующим образом. Пусть у нас есть граф $G = (V, E)$, где V – множество узлов (например, веб-страниц или городов), а E – множество ребер (ссылок между узлами или маршрутов между городами). Тогда значение PageRank для узла i вычисляется по формуле:

$$PR_i = (1 - d) + d \sum_j^{M_i} \frac{PR_j}{L_j}, \quad (10)$$

где PR_i – значение PageRank для узла i ,

d – коэффициент демпфирования или затухания (обычно принимается равным 0.85),

M_i – множество узлов, ссылающихся на узел i ,

L_j – количество исходящих ссылок из узла j .

Коэффициент демпфирования d определяет вероятность того, что пользователь продолжит переходить по ссылкам. Значение $(1 - d)$ отражает вероятность того, что пользователь начнет случайное блуждание с произвольной страницы.

Упрощенно созданный граф можно представить на рисунке 59. На нем изображено в качестве узлов 3 города (C_1 , C_2 и C_3), а также ребра, представляющие вероятность перехода из одного города в другой, веса которых высчитываются для каждого пассажира отдельно с использованием алгоритма случайного леса. Также в каждом узле определен PageRank по формуле, представленной выше. Как можно заметить, наименее релевантным городом в данном графе является C_3 .

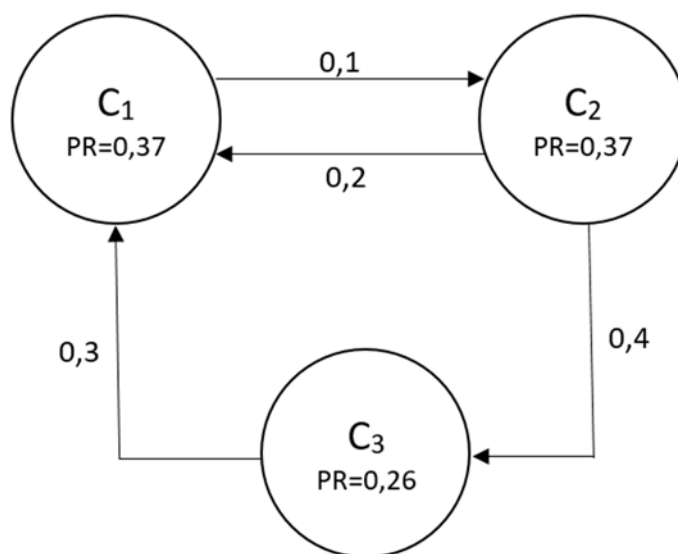


Рисунок 59 – Иллюстрация графовой модели

Случайный лес – это ансамблевый метод машинного обучения, использующий множество решающих деревьев для определения наилучшего решения через усреднение выводов отдельных деревьев [67]. Эта модель особенно эффективна для решения задач регрессии и классификации благодаря своей способности обрабатывать большие объемы данных с высокой точностью и устойчивостью к переобучению.

В контексте рекомендательных систем, случайный лес может использоваться для прогнозирования интересов пользователя, базируясь на его профиле и истории взаимодействий [68].

Таким образом, разработана оригинальная гибридная рекомендательная система для авиаперелетов, сочетающая в себе коллаборативную фильтрацию на основе SVD, графовые модели с алгоритмом PageRank и модель случайного леса. Система использует данные о просмотрах городов, покупках и бронированиях билетов, интересах пользователей, сезонности и популярности направлений, что обеспечивает более высокую точность и лучшую персонализацию рекомендаций.

3 Реализация автоматизированной системы обслуживания авиапассажиров

3.1 Реализация программного обеспечения

3.1.1 Реализация базы данных

На основании концептуальной и логической модели для СУБД Microsoft SQL Server 2019 была разработана физическая модель базы данных (рисунок 60).

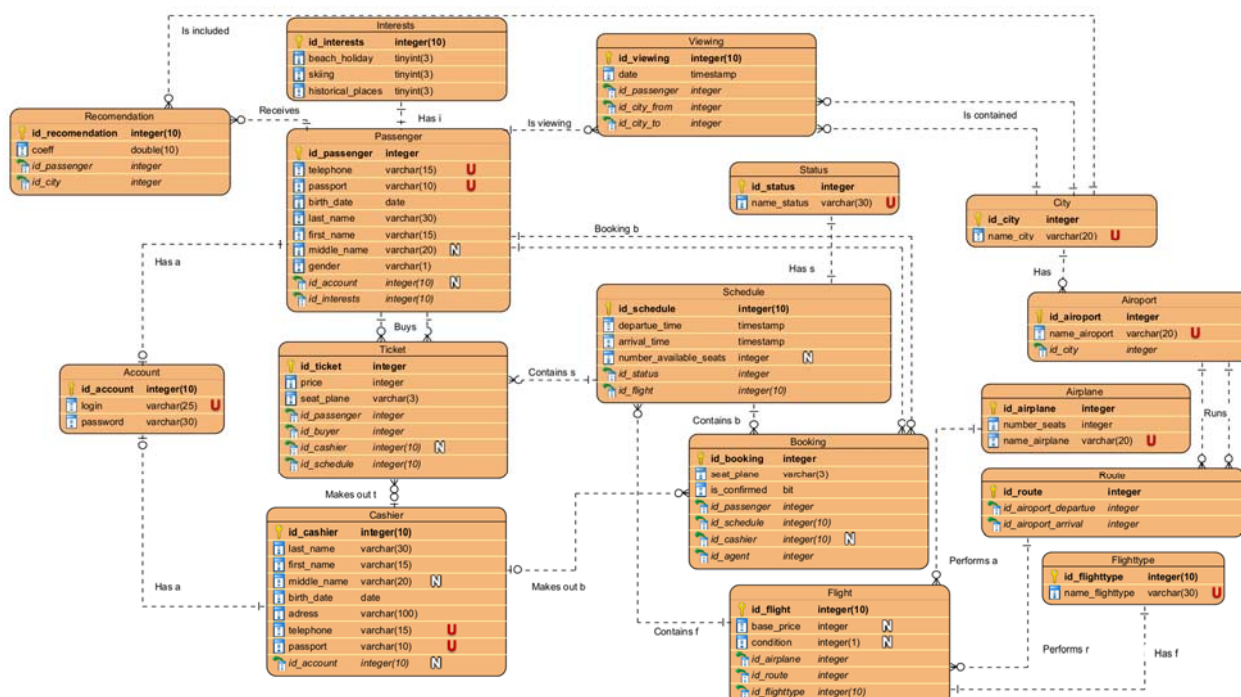


Рисунок 60 – Графическое представление физической модели базы данных

Описание атрибутов сущности «Пассажир» представлено в таблице 19.

Таблица 19 – Описание атрибутов сущности «Пассажир»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_passenger	Идентификатор	integer	Нет	Да	PK
telephone	Номер телефона	varchar(15)	Нет	Да	-
passport	Номер паспорта	varchar(10)	Нет	Да	-
birth_date	Дата рождения	date	Нет	Нет	-
last_name	Фамилия	varchar(30)	Нет	Нет	-
first_name	Имя	varchar(15)	Нет	Нет	-
middle_name	Отчество	varchar(20)	Да	Нет	-
gender	Пол	varchar(1)	Нет	Нет	-

Продолжение таблицы 19

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_account	Идентификатор аккаунта	integer	Да	Нет	FK
id_interests	Идентификатор интересов	integer	Нет	Нет	FK

Описание атрибутов сущности «Кассир» представлено в таблице 20.

Таблица 20 – Описание атрибутов сущности «Кассир»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_cashier	Идентификатор	integer	Нет	Да	PK
telephone	Номер телефона	varchar(15)	Нет	Да	-
passport	Номер паспорта	varchar(10)	Нет	Да	-
birth_date	Дата рождения	date	Нет	Нет	-
last_name	Фамилия	varchar(30)	Нет	Нет	-
first_name	Имя	varchar(15)	Нет	Нет	-
middle_name	Отчество	varchar(20)	Да	Нет	-
adress	Адрес проживания	varchar(100)	Нет	Нет	-
id_account	Идентификатор аккаунта	integer	Да	Нет	FK

Описание атрибутов сущности «Статус» представлено в таблице 21.

Таблица 21 – Описание атрибутов сущности «Статус»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_status	Идентификатор	integer	Нет	Да	PK
name_status	Наименование статуса рейса	varchar(30)	Нет	Да	-

Описание атрибутов сущности «Город» представлено в таблице 22.

Таблица 22 – Описание атрибутов сущности «Город»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_city	Идентификатор	integer	Нет	Да	PK
name_city	Наименование города	varchar(20)	Нет	Да	-

Описание атрибутов сущности «Аэропорт» представлено в таблице 23.

Таблица 23 – Описание атрибутов сущности «Аэропорт»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_airoport	Идентификатор	integer	Нет	Да	PK
name_airoport	Наименование аэропорта	varchar(20)	Нет	Да	-
id_city	Идентификатор города	integer	Нет	Нет	FK

Описание атрибутов сущности «Самолет» представлено в таблице 24.

Таблица 24 – Описание атрибутов сущности «Самолет»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_airplane	Идентификатор	integer	Нет	Да	PK
number_seats	Количество мест	integer	Нет	Нет	-
name_airplane	Наименование самолета	varchar(20)	Нет	Да	-

Описание атрибутов сущности «Маршрут» представлено в таблице 25.

Таблица 25 – Описание атрибутов сущности «Маршрут»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_route	Идентификатор	integer	Нет	Да	PK
id_airoport_departue	Идентификатор аэропорта вылета	integer	Нет	Нет	FK
id_airoport_arrival	Идентификатор аэропорта прилета	integer	Нет	Нет	FK

Описание атрибутов сущности «Тип рейса» представлено в таблице 26.

Таблица 26 – Описание атрибутов сущности «Тип рейса»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_flighttype	Идентификатор	integer	Нет	Да	PK
name_flighttype	Наименование типа рейса	varchar(30)	Нет	Да	-

Описание атрибутов сущности «Рейс» представлено в таблице 27.

Таблица 27 – Описание атрибутов сущности «Рейс»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_flight	Идентификатор	integer	Нет	Да	PK
base_price	Базовая стоимость	integer	Да	Нет	-
id_airplane	Идентификатор самолета	integer	Нет	Нет	FK
id_route	Идентификатор маршрута	integer	Нет	Нет	FK
id_flighttype	Идентификатор типа рейса	integer	Нет	Нет	FK
condition	Состояние рейса	integer	Да	Нет	-

Описание атрибутов сущности «Расписание» представлено в таблице 28.

Таблица 28 – Описание атрибутов сущности «Расписание»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_schedule	Идентификатор	integer	Нет	Да	PK
departure_time	Дата и время вылета	timestamp	Нет	Нет	-
arrival_time	Дата и время прилета	timestamp	Нет	Нет	-
number_available_seats	Количество свободных мест	integer	Да	Нет	-
id_status	Идентификатор статуса рейса	integer	Нет	Нет	FK
id_flight	Идентификатор рейса	integer	Нет	Нет	FK

Описание атрибутов сущности «Билет» представлено в таблице 29.

Таблица 29 – Описание атрибутов сущности «Билет»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_ticket	Идентификатор	integer	Нет	Да	PK
price	Стоимость	integer	Нет	Нет	-
seat_plane	Место в самолете	varchar(3)	Нет	Нет	-
id_passenger	Идентификатор пассажира	integer	Нет	Нет	FK
id_buyer	Идентификатор покупателя	integer	Нет	Нет	FK
id_cashier	Идентификатор кассира	integer	Нет	Нет	FK
id_schedule	Идентификатор расписания	integer	Нет	Нет	FK

Описание атрибутов сущности «Бронь» представлено в таблице 30.

Таблица 30 – Описание атрибутов сущности «Бронь»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_booking	Идентификатор	integer	Нет	Да	PK
seat_plane	Место в самолете	varchar(3)	Нет	Нет	-
is_confirmed	Флаг подтверждения бронирования	bit	Нет	Нет	-
id_passenger	Идентификатор пассажира	integer	Нет	Нет	FK
id_agent	Идентификатор оформителя	integer	Нет	Нет	FK
id_cashier	Идентификатор кассира	integer	Нет	Нет	FK
id_schedule	Идентификатор расписания	integer	Нет	Нет	FK

Описание атрибутов сущности «Аккаунт» представлено в таблице 31.

Таблица 31 – Описание атрибутов сущности «Аккаунт»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_account	Идентификатор	integer	Нет	Да	PK
login	Логин	varchar(25)	Нет	Да	-
password	Пароль	varchar(30)	Нет	Нет	-

Описание атрибутов сущности «Рекомендация» представлено в таблице 32.

Таблица 32 – Описание атрибутов сущности «Рекомендация»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_recomendation	Идентификатор	integer	Нет	Да	PK
coeff	Коэффициент рекомендации	double	Нет	Нет	-
id_passenger	Идентификатор пассажира	integer	Нет	Нет	FK
id_city	Идентификатор города	integer	Нет	Нет	FK

Описание атрибутов сущности «Интересы» представлено в таблице 33.

Таблица 33 – Описание атрибутов сущности «Интересы»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_interests	Идентификатор	integer	Нет	Да	PK
beach_holiday	Оценка пляжного отдыха	tinyint	Нет	Нет	-
skiing	Оценка горных лыж	tinyint	Нет	Нет	-
historical_places	Оценка исторических мест	tinyint	Нет	Нет	-

Описание атрибутов сущности «Просмотр» представлено в таблице 34.

Таблица 34 – Описание атрибутов сущности «Просмотр»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
id_viewing	Идентификатор	integer	Нет	Да	PK
date	Дата	timestamp	Нет	Нет	-
id_passenger	Идентификатор пассажира	integer	Нет	Нет	FK
id_city_from	Идентификатор города вылета	integer	Нет	Нет	FK
id_city_to	Идентификатор города прилета	integer	Нет	Нет	FK

На данной физической модели структура базы данных отображена с учетом спецификации типов СУБД Microsoft SQL Server 2019.

На основе физической модели БД был сгенерирован скрипт на языке SQL для создания базы данных. Его код представлен в приложении Б. Развертывание спроектированной базы данных было выполнено путем запуска данного скрипта в СУБД Microsoft SQL Server 2019.

3.1.2 Реализация серверной части приложения

Реализация серверной части приложения была произведена в среде программирования Microsoft Visual Studio 2019 на языке C# с применением технологии Web API на платформе ASP.NET [56, 69]. Для взаимодействия с созданной базой данных была использована объектно-ориентированная технология Entity Framework [57, 70].

На основании используемой архитектуры MVC все классы серверной части делятся на два слоя: слой модели, предназначенный для взаимодействия с базой данных, и слой контроллеров, предназначенный для обработки HTTP-запросов к серверу.

Диаграмма классов слоя модели представлена на рисунке 61.

Представленные на данной диаграмме классы были автоматически сгенерированы средой программирования Visual Studio с помощью технологии Entity Framework. Генерация производилась на основе реализованной до этого базы данных.

Как можно увидеть из диаграммы, классы Passenger, Ticket, Schedule, Cashier, Status, Booking, Flight, City, Airoport, Route, Airplane Flighttype, Account, Recommendation, Interests и Viewing были созданы на основе одноименных таблиц реализованной базы данных. Классы же PassengersFlightsCount, AverageFlightLoad, BookingsInformation и TicketsBookings основываются на одноименных представлениях и позволяют облегчить выборку нужных данных из базы.

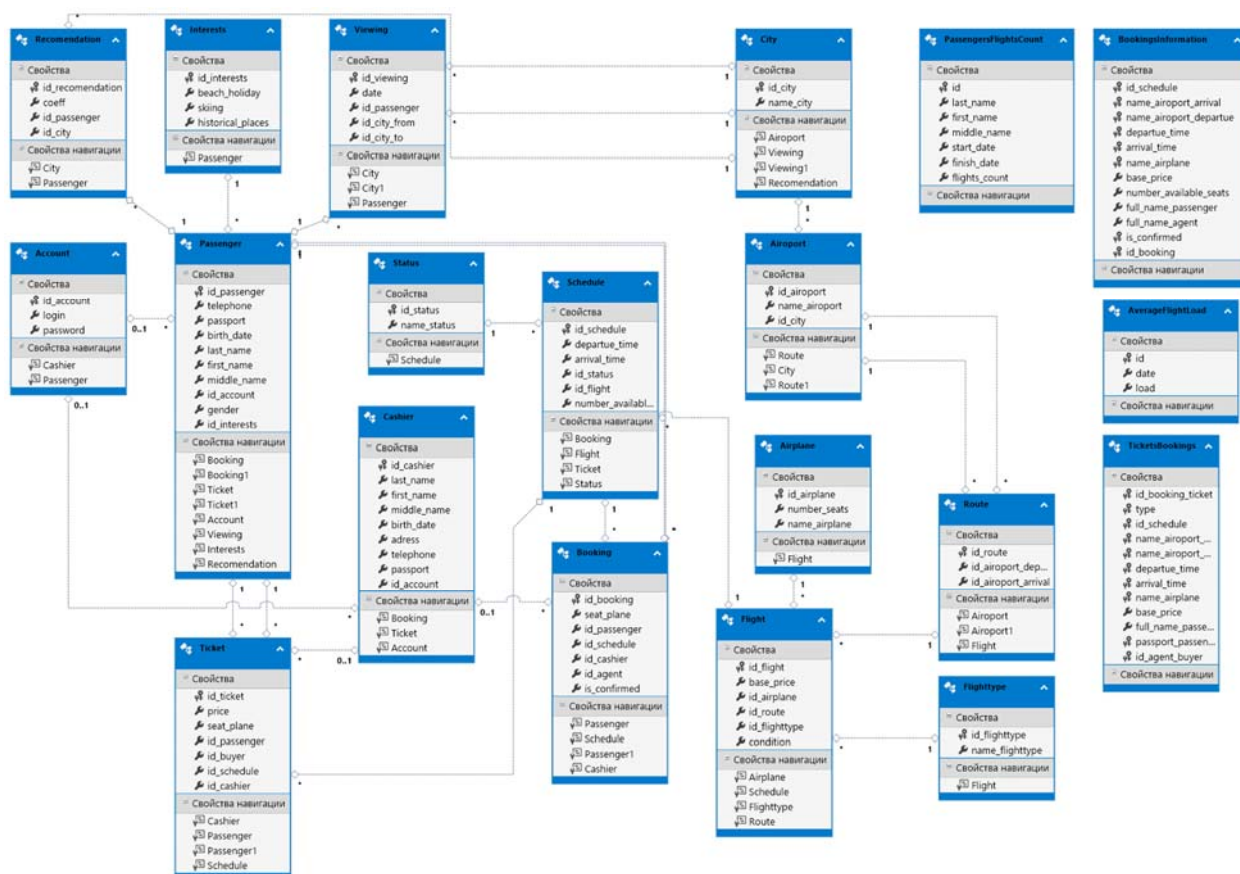


Рисунок 61 – Диаграмма классов серверной части слоя модели

Диаграмма классов слоя контроллеров представлена на рисунке 62.

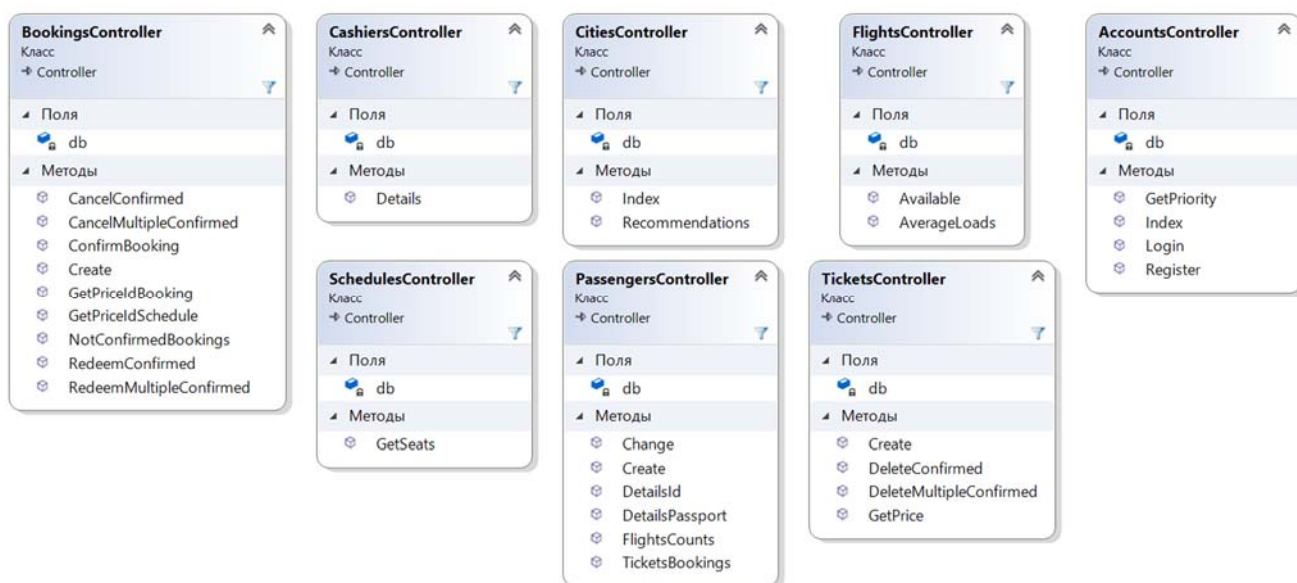


Рисунок 62 – Диаграмма классов серверной части слоя контроллеров

Описание классов слоя контроллеров приведено в таблице 35.

Таблица 35 – Описание классов слоя контроллеров

Класс	Назначение
BookingsController	Обрабатывает запросы добавления новых броней, выкупа и снятия одной и множества броней
CashiersController	Обрабатывает запросы получения информации о кассире
CitiesController	Обрабатывает запросы получения списка городов, а также получения списка рекомендованных городов авторизованному пользователю
FlightsController	Обрабатывает запросы получения доступных рейсов на определенное направление и дату, а также получения информации о средней загрузженности рейсов по дням
TicketsController	Обрабатывает запросы добавления новых билетов, возврата одного и множества билетов
SchedulesController	Обрабатывает запросы получения списка свободных мест на расписание
PassengersController	Обрабатывает запросы добавления нового пассажира или покупателя, изменения его данных, получения информации о пассажирах или покупателях и их номеров, а также информации о количестве совершенных полетов по месяцам
AccountsController	Обрабатывает запросы авторизации и регистрации пользователей и получения информации о них

Исходный код серверной части приведен в приложении В.

3.1.3 Реализация мобильного приложения

На данном этапе было реализовано мобильное приложение на языке C# в среде Visual Studio 2019. Исходный код представлен в приложении Г. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 63.

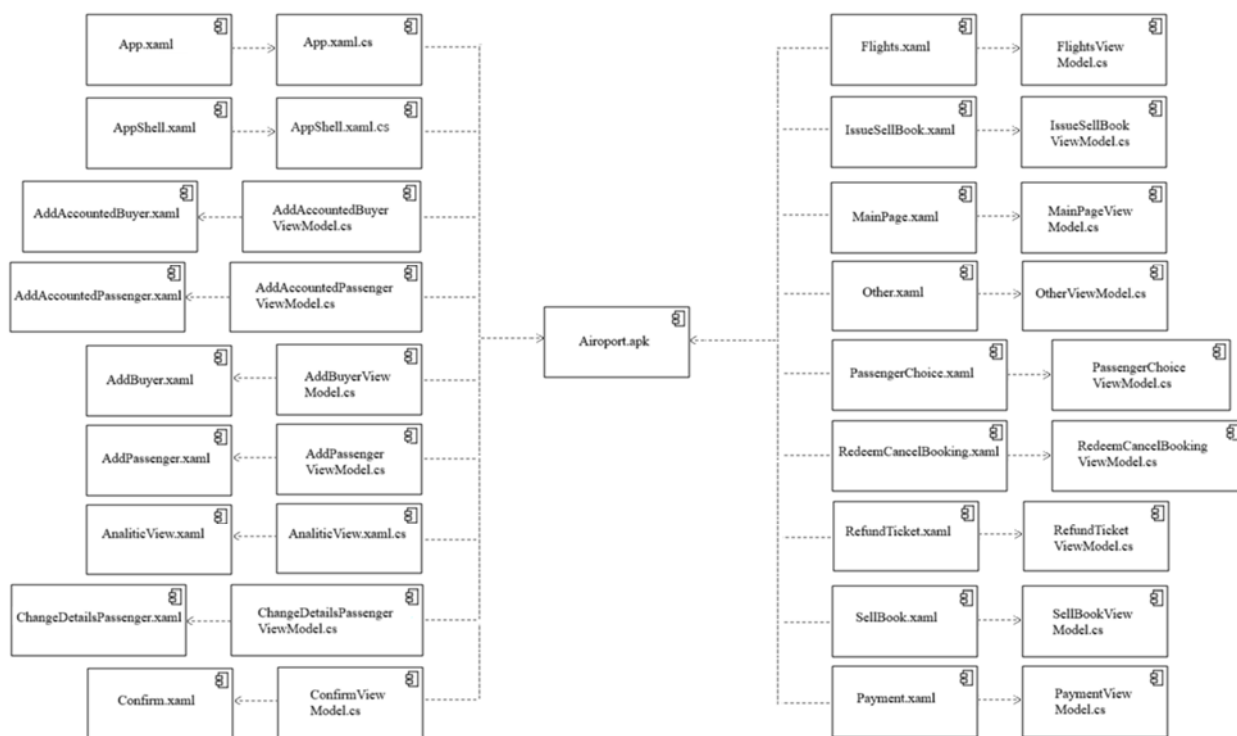


Рисунок 63 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 36.

Таблица 36 – Описание компонентов мобильного приложения

Компонент	Назначение
App.xaml	ХАМЛ-файл описания подключаемых ресурсов к приложению
App.xaml.cs	Начальная точка загрузки и закрытия приложения
AppShell.xaml	ХАМЛ-файл описания структуры интерфейса приложения
AppShell.xaml.cs	Регистрирует страницы приложения, не вошедшие в основную структуру
AddAccountedBuyer.xaml	ХАМЛ-файл описания интерфейса страницы добавления ранее учтенного покупателя
AddAccountedBuyerViewModel.cs	Управляет логикой и данными страницы добавления ранее учтенного покупателя
AddAccountedPassenger.xaml	ХАМЛ-файл описания интерфейса страницы добавления ранее учтенного пассажира
AddAccountedPassengerViewModel.cs	Управляет логикой и данными страницы добавления ранее учтенного пассажира
AddBuyer.xaml	ХАМЛ-файл описания интерфейса страницы добавления нового покупателя
AddBuyerViewModel.cs	Управляет логикой и данными страницы добавления нового покупателя
AddPassenger.xaml	ХАМЛ-файл описания интерфейса страницы добавления нового пассажира
AddPassengerViewModel.cs	Управляет логикой и данными страницы добавления нового пассажира
AnaliticView.xaml	ХАМЛ-файл описания интерфейса страницы с аналитической информацией
AnaliticView.xaml.cs	Управляет логикой и данными страницы с аналитической информацией

Продолжение таблицы 36

Компонент	Назначение
ChangeDetailsPassenger.xaml	ХАМЛ-файл описания интерфейса страницы изменения данных пассажира или покупателя
ChangeDetailsPassengerViewModel.cs	Управляет логикой и данными страницы изменения данных пассажира или покупателя
Flights.xaml	ХАМЛ-файл описания интерфейса страницы вывода списка рейсов
FlightsViewModel.cs	Управляет логикой и данными страницы вывода списка рейсов
IssueSellBook.xaml	ХАМЛ-файл описания интерфейса страницы покупки/бронирования билетов
IssueSellBookViewModel.cs	Управляет логикой и данными страницы покупки/бронирования билетов
MainPage.xaml	ХАМЛ-файл описания интерфейса страницы авторизации
MainPageViewModel.cs	Управляет логикой и данными страницы авторизации
Other.xaml	ХАМЛ-файл описания интерфейса страницы «Другое»
OtherViewModel.cs	Управляет логикой и данными страницы «Другое»
PassengerChoice.xaml	ХАМЛ-файл описания интерфейса страницы выбора пассажира или покупателя при изменении его данных
PassengerChoiceViewModel.cs	Управляет логикой и данными страницы выбора пассажира или покупателя при изменении его данных
RedeemCancelBooking.xaml	ХАМЛ-файл описания интерфейса страницы выкупа и снятия брони
RedeemCancelBookingViewModel.cs	Управляет логикой и данными страницы выкупа и снятия брони
RefundTicket.xaml	ХАМЛ-файл описания интерфейса страницы возврата билетов
RefundTicketViewModel.cs	Управляет логикой и данными страницы возврата билетов
SellBook.xaml	ХАМЛ-файл описания интерфейса страницы продажи и бронирования билетов
SellBookViewModel.cs	Управляет логикой и данными страницы продажи и бронирования билетов
Confirm.xaml	ХАМЛ-файл описания интерфейса подтверждения бронирования билета
ConfirmViewModel.cs	Управляет логикой и данными страницы подтверждения бронирования билета
Payment.xaml	ХАМЛ-файл описания интерфейса оплаты билетов
PaymentViewModel.cs	Управляет логикой и данными страницы оплаты билетов
Airopport.apk	Исполняемый файл мобильного приложения

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 64.

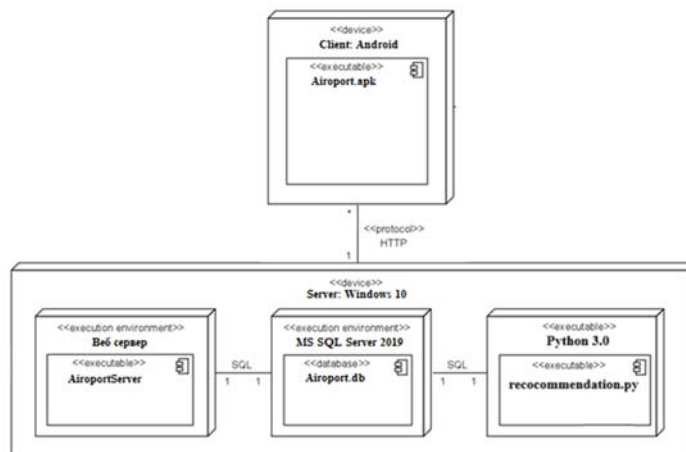


Рисунок 64 – Диаграмма развертывания программного обеспечения

Из данной диаграммы можно увидеть, что программное обеспечение состоит из четырех уровней: мобильное приложение, работающее под управлением ОС Android, серверная часть СУБД и рекомендательная система, работающие под управлением ОС Windows 10.

3.1.4 Тестирование программного обеспечения

3.1.4.1 Тестирование серверной части приложения

3.1.4.1.1 Функциональное тестирование

На данном этапе было выполнено функциональное тестирование в целях проверки правильности работы серверной части приложения. Результаты функционального тестирования серверной части приложения представлены в таблице 37.

Таблица 37 – Результаты функционального тестирования серверной части

№	Описание теста	Структура запроса	Результат
1	Авторизация кассира	Метод: GET URL: Cashiers/Login?login=Matveev &password=Matveevich1983	Тест пройден. Кассир авторизован (рисунок Д.1).
2	Получение информации о кассире по его идентификатору	Метод: GET URL: Cashiers/Details/4	Тест пройден. Информация получена (рисунок Д.2).
3	Получение списка городов	Метод: GET URL: Cities	Тест пройден. Список получен (рисунок Д.3).
4	Получение списка рейсов на определенное направление и дату	Метод: GET URL: Flights/Available?city_from=Пенза &city_to=Москва&date=2025-04-01 &priority=0&id_passenger=0	Тест пройден. Список получен (рисунок Д.4).
5	Получение списка свободных мест на определенный рейс по его идентификатору	Метод: GET URL: Schedules/GetSeats/3	Тест пройден. Список получен (рисунок Д.5).
6	Получение информации о пассажире по его идентификатору	Метод: GET URL: Passengers/DetailsId/10	Тест пройден. Информация получена (рисунок Д.6).
7	Изменение данных пассажира	Метод: PUT URL: Passengers/Change Body: { "id_passenger": 3, "telephone": "+79375091284", "passport": "5299100322", "birth_date": "1992-09-27", "last_name": "Богданов", "first_name": "Богдан", "middle_name": "Богданович", "priority": 0 }	Тест пройден. Данные изменены (рисунок Д.7).

Продолжение таблицы 37

№	Описание теста	Структура запроса	Результат
8	Добавление пассажира	Метод: POST URL: Passengers/Create Body: { "telephone": "+79378410924", "passport": "9288160325", "birth_date": "1993-08-14", "last_name": "Захаров", "first_name": "Захар", "middle_name": "Захарович" }	Тест пройден. Пассажир добавлен (рисунок Д.8).
9	Добавление брони	Метод: POST URL: Bookings/Create Body: { "seat_plane": "08c", "id_passenger": 10, "id_buyer": 10, "id_schedule": 5, "priority": 0 }	Тест пройден. Бронь добавлена (рисунок Д.9).
10	Снятие брони по ее идентификатору	Метод: DELETE URL: Bookings/Cancel?id_booking=224 &priority=0	Тест пройден. Бронь снята (рисунок Д.10).
11	Выкуп брони	Метод: POST URL: Bookings/Redeem Body: { "id_booking": 241, "priority": 0 }	Тест пройден. Бронь выкуплена (рисунок Д.11).
12	Снятие множества броней по номеру рейса и номеру паспорта оформителя броней	Метод: DELETE URL: Bookings/CancelMultiple? id_schedule=5&passport=9090761536 &priority=0	Тест пройден. Множество броней снято (рисунок Д.12).
13	Выкуп множества броней	Метод: POST URL: Bookings/RedeemMultiple Body: { "id_schedule": 63, "passport": "0878710132" }	Тест пройден. Множество броней выкуплено (рисунок Д.13).
14	Добавление билета	Метод: POST URL: Tickets/Create Body: { "seat_plane": "01b", "id_passenger": 3, "id_buyer": 4, "id_schedule": 13, "id_cashier": 2, "priority": 0 }	Тест пройден. Билет добавлен (рисунок Д.14).
15	Возврат билета по его идентификатору	Метод: DELETE URL: Tickets/Refund?id_ticket=1047 &priority=0	Тест пройден. Возврат билета осуществлен (рисунок Д.15).
16	Возврат множества билетов	Метод: DELETE URL: Tickets/RefundMultiple? id_schedule=41&passport=6279034552	Тест пройден. Возвращено множество билетов (рисунок Д.16).
17	Получение средней загруженности рейсов по дням за последнюю неделю	Метод: GET URL: Flights/AverageLoads	Тест пройден. Информация получена (рисунок Д.17).
18	Получение количества полетов пассажиров по месяцам за последний год	Метод: GET URL: Passengers/FlightsCounts	Тест пройден. Информация получена (рисунок Д.18).

Все тесты тестирования были пройдены. Значит можно говорить о правильности работы серверной части приложения.

3.1.4.1.2 Модульное тестирование

В дополнении к функциональному тестированию было проведено модульное тестирование серверной части для запроса получения списка рейсов на заданное направление и дату, а также для запроса авторизации

кассира. Оно выполнялось на языке программирования C# в среде разработки Visual Studio 2019. Исходный код тестов приведен в приложении Е, а его результаты – на рисунке 65.

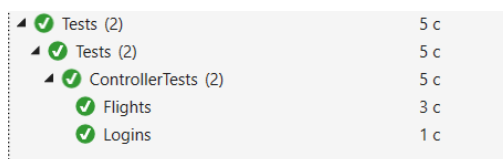


Рисунок 65 – Результаты модульного тестирования

Все тесты модульного тестирования прошли успешно. Следовательно, можно говорить о корректной работе данных функций.

3.1.4.2 Тестирование клиентской части приложения

Также было произведено функциональное тестирование мобильного приложения, результаты которого приведены в таблице 38.

Таблица 38 – Результаты функционального тестирования мобильного приложения

Вариант использования	Параметры теста	Результат
Авторизоваться	Авторизация кассира с корректными данными. Логин: Fedorov Пароль: Fedorovich1960	Тест пройден. Авторизация выполнена (рисунок Ж.1).
	Авторизация кассира с некорректными данными. Логин: Fed Пароль: Fed196	Тест пройден. Показано сообщение с ошибкой (рисунок Ж.2).
Оформить продажу билетов	Оформление продажи билетов	Тест пройден. Продажа оформлена (рисунок Ж.3).
Оформить бронирование билетов	Оформление бронирования билетов	Тест пройден. Бронирование оформлено (рисунок Ж.4).
Занести информацию о покупателе	Добавление нового покупателя	Тест пройден. Информация занесена (рисунок Ж.5).
Занести информацию о пассажире	Добавление нового пассажира	Тест пройден. Информация занесена (рисунок Ж.6).
Оформить выкуп брони	Оформление выкупа брони	Тест пройден. Выкуп брони оформлен (рисунок Ж.7).
Оформить снятие брони	Оформление снятия брони	Тест пройден. Снятие брони оформлено (рисунок Ж.8).

Продолжение таблицы 38

Вариант использования	Параметры теста	Результат
Оформить выкуп множества броней	Оформление выкупа множества броней	Тест пройден. Выкуп множества броней оформлен (рисунок Ж.9).
Оформить снятие множества броней	Оформление снятия множества броней	Тест пройден. Снятие множества броней оформлено (рисунок Ж.10).
Оформить возврат билета	Оформление возврата билета	Тест пройден. Возврат билета оформлен (рисунок Ж.11).
Оформить возврат множества билетов	Оформление возврата множества билетов	Тест пройден. Возврат множества билетов оформлен (рисунок Ж.12).
Получить информацию о средней загруженности рейсов по дням	Получение информации о средней загруженности рейсов по дням	Тест пройден. Информация о средней загруженности получена (рисунок Ж.13).
Получить информацию о количестве совершенных полетов для каждого пассажира по месяцам	Получение информации о количестве совершенных полетов для каждого пассажира по месяцам	Тест пройден. Информация о количестве полетов получена (рисунок Ж.14).
Изменить данные пассажира или покупателя	Изменение данных пассажира или покупателя	Тест пройден. Данные изменены (рисунок Ж.15).

Все тесты были выполнены успешно. Следовательно, можно говорить о корректной работе мобильного приложения.

3.1.4.3 Метрики кода

Метрики кода позволяют получить количественные величины, характеризующее программное обеспечение по тем или иным свойствам. Существует большое количество разнообразных метрик. Одной из самых простых метрик кода является количество строк кода (SLOC-метрики). Также одной из самых распространенных метрик, основанных на подсчете некоторых единиц в коде программы, относят метрики Холстеда.

Рассмотрим 3 метода класса Service, отвечающие за авторизацию кассира в системе (LoginCashierAsync), получение списка рейсов (GetFlightsAsync) и получения списка наименований городов (GetCitiesAsync). Коды данных методов приведены в листинге 1.

Листинг 1 – Коды методов класса Service

```
private static string url = "http://192.168.0.245:44317/";
private static HttpClientHandler ClientHandler = new HttpClientHandler();
private static HttpClient Client = new HttpClient(ClientHandler);

///  

/// <summary>
```

```

/// Функция для получения кассира по номеру паспорта
/// </summary>
/// <param name="login">Логин</param>
/// <param name="password">Пароль</param>
/// <returns>Объект типа Cashier или null, если номер паспорта не
найден</returns>
public async static Task<Cashier> LoginCashierAsync(string login, string
password)
{
    Cashier cashier = null;
    var response = await Client.GetAsync(url + "Cashiers/Login?login=" +
login + "&password=" + password);

    if (response.IsSuccessStatusCode) {
        string responseText = await response.Content.ReadAsStringAsync();
        cashier = JsonConvert.DeserializeObject<Cashier>(responseText);
    }
    return cashier;
}

/// <summary>
/// Функция для получения списка рейсов
/// </summary>
/// <param name="city_from">Город отправления</param>
/// <param name="city_to">Город прилета</param>
/// <param name="date">Дата отправления</param>
/// <returns>Список объектов типа Flight или null, если рейсы не
найжены</returns>
public async static Task<List<Flight>> GetFlightsAsync(string city_from,
string city_to, DateTime date)
{
    List<Flight> flights = null;
    string date_str = date.Date.ToString("yyyy-MM-dd");
    var response = await Client.GetAsync(url + "Flights/Available?city_from="
+ city_from + "&city_to=" + city_to + "&date=" + date_str);

    if (response.IsSuccessStatusCode)
    {
        string responseText = await response.Content.ReadAsStringAsync();
        flights = JsonConvert.DeserializeObject<List<Flight>>(responseText);
    }
    return flights;
}

/// <summary>
/// Функция для получения списка наименований имеющихся городов
/// </summary>
/// <returns>Список объектов типа string или пустой список, если города не
найжены</returns>
public static async Task<List<string>> GetCitiesAsync()
{
    List<City> cities = new List<City>();
    List<string> name_cities = new List<string>();
    var response = await Client.GetAsync(url + "Cities");

    if (response.IsSuccessStatusCode)
    {
        string responseText = await response.Content.ReadAsStringAsync();
        cities = JsonConvert.DeserializeObject<List<City>>(responseText);
        foreach (City city in cities)
        {
            name_cities.Add(city.name_city);
        }
    }
}

```

```

    return name_cities;
}

```

Данные методы находятся в клиентской части приложения. Функция для получения кассира запускается при попытке авторизации кассира в мобильном приложении. В ней выполняется HTTP-запрос к серверной части приложения для получения информации о кассире. Если данный запрос успешный, то возвращается данная информация вызываемому методу, иначе возвращается null. Функция для получения списка рейсов вызывается при поиске рейсов и работает аналогично функции получения информации о кассире. Функция получения списка наименований имеющихся городов вызывается после авторизации кассира и работает также, как и предыдущие, только возвращает не полную информацию о городах, а лишь их названия.

Для данных методов будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

3.1.4.3.1 Метрики размера программы

Данная группа метрик используется для измерения объема программы с помощью подсчета количества строк в тексте исходного кода. При этом часто выделяют методику подсчета физических строк кода и строк комментариев. Вычислим метрики размера программы для кода, приведенного в листинге 1 (таблица 39).

Таблица 39 – Метрики размера программы

Метрика	Количество
Количество строк кода	64
Количество строк комментариев	17

Теперь вычислим процент комментариев. Он равен $(17/64)*100\%=26\%$. Это означает, что количество комментариев в коде достаточно для его хорошего понимания, и при этом не слишком велико и не переполняет программу.

3.1.4.3.2 Метрики Холстеда

Метрика Холстеда вычисляется на основе анализа числа строк и синтаксических элементов исходного кода программы, а именно

используются характеристики: число уникальных операторов, число уникальных операндов, общее число операторов и общее число операндов [71 – 73]. Для измерения данных характеристик необходимо составить словарь операторов (таблица 40) и словарь операндов (таблица 41).

Таблица 40 – Словарь операторов

Оператор	Число вхождений
=	16
.	19
If	3
foreach	1
return	3
HttpClientHandler	1
HttpClient	1
Client.GetAsync	3
response.Content.ReadAsStringAsync	3
JsonConvert.DeserializeObject	3
date.Date.ToString	1
List	2
name_cities.Add	1
;	21
,	2
}	7
+	9

Таблица 41 – Словарь операндов

Операнд	Число вхождений
url	4
ClientHandler	2
Client	4
login	2
password	2
cashier	3
response	9
response.IsSuccessStatusCode	3
responseText	6
city_from	2
city_to	2
date	2
flights	3
date_str	2
cities	3
name_cities	3
city	2
"http://192.168.0.227:44307/"	1
null	2
"Cashiers/Login?passport="	1
"yyyy-MM-dd"	1
"Flights/Available?city_from="	1
"&city_to="	1
"&date="	1
"Cities"	1

Из вышеописанных таблиц получаем следующие данные: число уникальных операторов программы $n_1 = 17$, число уникальных операндов программы $n_2 = 25$, общее число операторов в программе $N_1 = 96$, общее число операндов в программе $N_2 = 63$.

Вычислим некоторые из существующих характеристик на основании полученных данных:

- длина программы (N):

$$N = N_1 + N_2 = 159, \quad (11)$$

- словарь программы (n):

$$n = n_1 + n_2 = 42, \quad (12)$$

- теоретический объем программы (V):

$$V = N \cdot \log_2 N \approx 1162,5, \quad (13)$$

- сложность программы (D):

$$D = \left(\frac{n_1}{2}\right) \cdot \left(\frac{N_2}{n_2}\right) \approx 21.4, \quad (14)$$

- оценка интеллектуальных усилий (E):

$$E = D \cdot V = 24877. \quad (15)$$

Вычисленные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшего усовершенствования поддержки.

3.2 Разработка и тестирование программы, реализующей гибридную рекомендательную систему для авиаперелетов

На основе разработанной ранее схемы взаимодействий между компонентами рекомендательной системы была создана программа, реализующая ее работу. Код программы представлен в приложении И. Для проверки корректности работы системы было проведено тестирование ее работы.

Примеры вывода рекомендованных авианаправлений из мобильного приложения представлены на рисунках 66 – 69. В первом случае пассажир периодически летает в Сочи и Пензу, поэтому ему предлагают данные направления. Но также он бывал несколько раз в Нижнем Новгороде. А так как помимо этого в его профиле указано, что он имеет высокую оценку предпочтения исторических мест, то ему также предлагается посетить Ереван, в котором он ни разу не был.

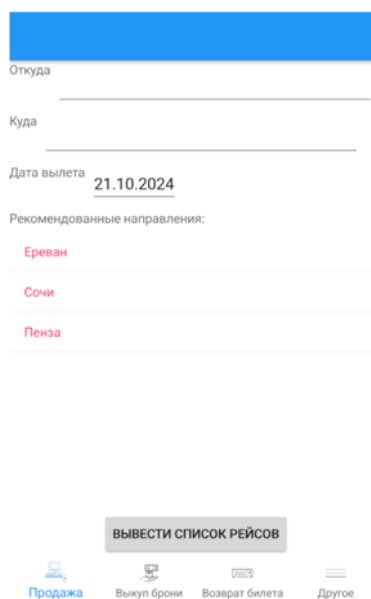


Рисунок 66 – Пример вывода рекомендованных авианаправлений для пассажира 1

Во втором случае (рисунок 67) пассажир очень часто летает в Казань и Афины через Москву. В других городах он ни разу не был, а интересы не выставлены в его профиле. Поэтому рекомендательная система предлагает ему только эти 3 города.

The screenshot shows a flight search interface. At the top, there is a blue header bar. Below it, the 'Откуда' (From) field is empty. The 'Куда' (To) field is also empty. The 'Дата вылета' (Departure date) is set to '21.10.2024'. Below this, the text 'Рекомендованные направления:' (Recommended directions:) is displayed. Underneath, three destinations are listed: 'Казань' (Kazan), 'Афины' (Athens), and 'Москва' (Moscow). At the bottom, there is a button labeled 'ВЫВЕСТИ СПИСОК РЕЙСОВ' (Show flight list). Below the button, there are four icons with labels: 'Продажа' (Sale), 'Выкуп брони' (Buy back booking), 'Возврат билета' (Return ticket), and 'Другое' (Other).

Рисунок 67 – Пример вывода рекомендованных авианаправлений для пассажира 2

В третьем случае (рисунок 68) пассажир имеет максимальную оценку пляжного отдыха, а также летал в Сочи и Симферополь. В связи с этим ему рекомендательная система предлагает рейсы в Баку и Махачкалу, где есть море и где он ни разу не был. А так как пассажир летал несколько раз в Москву и Санкт-Петербург, то система предполагает, что ему может быть интересен Великий Новгород, где он ни разу не был, несмотря на то, что предпочтения исторических мест у него выставлены достаточно низкие.

The screenshot shows a flight search form with the following elements:

- A blue header bar.
- Input fields for "Откуда" (From) and "Куда" (To), both currently empty.
- A date field labeled "Дата вылета" (Departure date) with the value "22.04.2025".
- A section titled "Рекомендованные направления:" (Recommended destinations:).
- A list of three recommended destinations: "Баку" (Baku), "Махачкала" (Makhachkala), and "Великий Новгород" (Veliky Novgorod), each on a separate line.
- A button labeled "ВЫВЕСТИ СПИСОК РЕЙСОВ" (Show flight list).
- A navigation bar at the bottom with four icons and labels: "Продажа" (Sale), "Выкуп брони" (Buy back booking), "Возврат билета" (Return ticket), and "Другое" (Other).

Рисунок 68 – Пример вывода рекомендованных авианаправлений для пассажира 3

В четвертом случае (рисунок 69) пассажир часто летает в Москву и из Москвы, поэтому ему рекомендательная система предлагает данное направление первым. Также в интересах пользователь установил очень высокое значение пляжного отдыха, а также он уже летал в Афины, в связи с чем вторым предложенным авианаправлением является Симферополь. Помимо этого, пассажир установил высокое значение интересов исторических мест, и он уже был в Казани и Великом Новгороде, из-за чего ему также предлагается полететь в Санкт-Петербург, где он еще не был.

Откуда

Куда

Дата вылета

22.04.2025

Рекомендованные направления:

Москва

Симферополь

Санкт-Петербург

ВЫВЕСТИ СПИСОК РЕЙСОВ

Продажа

Выкуп брони

Возврат билета

Другое

Рисунок 69 – Пример вывода рекомендованных авианаправлений для пассажира 4

Результаты тестирования показали корректность работы программы.

ЗАКЛЮЧЕНИЕ

В результате выполнения диссертационной работы была разработана автоматизированная система обслуживания авиапассажиров, а также интегрированная в данную систему гибридная рекомендательная система. Для достижения поставленных целей был проведен подробный анализ существующих решений, в ходе которого были выявлены недостатки современных систем обслуживания авиапассажиров. В связи с этим было принято решение о создании собственной системы и соответствующей рекомендательной подсистемы, а также о проведении онтологического и имитационного моделирования в рамках исследуемой предметной области и актуализации существующей классификации средств обслуживания авиапассажиров.

Разработанная система полностью удовлетворяет предъявленным функциональным и нефункциональным требованиям. В процессе тестирования на различных тестовых наборах входных данных ошибок выявлено не было, что подтверждает стабильную и корректную работу системы. Также была проведена оценка качества программного кода с использованием метрик, результаты которой показали высокий уровень его качества и пригодность к дальнейшему сопровождению. В состав системы была внедрена гибридная рекомендательная подсистема, эффективность которой была подтверждена в ходе тестирования. Помимо этого, были выполнены онтологическое и имитационное моделирование предметной области, а также разработана расширенная классификация средств обслуживания авиапассажиров.

Программа, реализующая рекомендательную подсистему, была официально зарегистрирована в государственном Реестре программ для ЭВМ Роспатента, что подтверждается соответствующим свидетельством, представленным в приложении Л.

Основные результаты диссертационной работы состоят в следующем:

– проведен расширенный и углубленный критический анализ предметной области по тематике магистерской диссертации;

– разработана, реализована и протестирована автоматизированная система обслуживания авиапассажиров с применением современных и эффективных информационных и программных технологий, включая рекомендательные механизмы, а также методы онтологического, имитационного и визуального моделирования. Проведена работа по подготовке к внедрению результатов работы;

– предложен подход к разработке гибридной рекомендательной системы, который отличается от известных подходов к разработке рекомендательных систем тем, что сочетает коллаборативную фильтрацию, графовые модели и методы машинного обучения. Предложенный подход позволяет повысить точность прогнозов и релевантность предложений;

– разработан сетевой график IT-проекта по созданию системы обслуживания авиапассажиров и определен срок выполнения и бюджет IT-проекта, что показало экономическую целесообразность и практическую значимость проекта;

– предложены две дискретно-событийные имитационные модели для исследования и улучшения параметров функционирования системы обслуживания авиапассажиров. Важной особенностью предложенного набора моделей является то, что он, в отличие от известных аналогичных моделей, имеет комплексный характер и позволяет учитывать как технические, так и человеческие аспекты работы аэропортового персонала;

– предложена расширенная онтологическая модель предметной области системы обслуживания авиапассажиров, отличающаяся от известных расширенным набором сущностей и классов предметной области, с целью формализации и систематизации информации для эффективного проведения проектирования объекта исследования;

– предложена модифицированная классификация средств обслуживания авиапассажиров, отличающаяся от известных тем, что

предлагается более полный расширенный перечень классификационных признаков и средств;

– по теме магистерской диссертации получены следующие документы:

1) диплом победителя Международного научно-исследовательского конкурса «Достижения вузовской науки 2023»;

2) диплом I степени Международного научно-исследовательского конкурса «Наш выбор – наука!»;

3) 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;

– по теме магистерской диссертации сделано 12 публикаций, включая две статьи в журнале, входящем в перечень ВАК, и проведена апробация на восьми научно-технических 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. Демура Н.А. Цифровизация: сущность и роль в развитии национальной экономики // Научный результат. Экономические исследования. – 2021. – №1. – С. 22 – 30.
2. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. – URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения 2023-12-12).
3. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». – URL: <http://publication.pravo.gov.ru/document/0001201705100002> (дата обращения 2023-12-12).
4. Указ Президента Российской Федерации от 10 октября 2019 г. № 490 «О развитии искусственного интеллекта в Российской Федерации». – URL: <http://publication.pravo.gov.ru/Document/View/0001201910110003> (дата обращения 2023-12-12).
5. Распоряжение Правительства Российской Федерации от 5 октября 2023 г. № 2715-р. – URL: <http://static.government.ru/media/files/nxkdhSkklndGCcrYovvXHLpW94TsOQ3D.pdf> (дата обращения 2023-12-12).
6. Постановление Правительства РФ от 8 августа 2022 г. № 1393 «Об утверждении требований к автоматизированной информационной системе оформления воздушных перевозок, к базам данных, входящим в ее состав, к информационно-телекоммуникационной сети, обеспечивающей работу указанной автоматизированной информационной системы, к ее оператору, а также мер по защите информации, содержащейся в ней, и порядка ее функционирования и изменении и признании утратившими силу некоторых актов правительства российской федерации». – URL: <http://>

static.government.ru/media/files/nf3IHO7O8Z9TASHtsA7fzvHv3tA3tbKU.pdf
(дата обращения 2023-12-12).

7. Приказ Минтранса России от 28 июня 2007 г. № 82 «Об утверждении Федеральных авиационных правил «Общие правила воздушных перевозок пассажиров, багажа, грузов и требования к обслуживанию пассажиров, грузоотправителей, грузополучателей». – URL: <https://base.garant.ru/191872/> (дата обращения 2023-12-12).

8. Романенко В.А. Организация и технология пассажирских воздушных перевозок: учебное пособие. – Самара: Самарский государственный аэрокосмический университет, 2004. – 100 с.

9. Маринцева К.В. Классификация средств механизации и автоматизации процессов обслуживания авиапассажиров // Транспортные системы и технологии перевозок. – 2012. – С. 56 – 61.

10. Староверов И.М., Князев В.Н. Современные подходы к классификации средств обслуживания авиапассажиров // Современные тенденции развития фундаментальных и прикладных наук: Материалы VII Всероссийской научно-практической конференции. – 2024. – С. 370 – 374.

11. История компании Аэрофлот. – URL: https://www.aeroflot.ru/ru-ru/about/aeroflot_today/aeroflot_history (дата обращения 2023-12-12).

12. Мобильное приложение Аэрофлота признано лучшим в мире. – URL: <https://www.aeroflot.ru/ru-ru/news/60445> (дата обращения 2023-12-12).

13. Годовой отчет Аэрофлота. – URL: https://ir.aeroflot.ru/fileadmin/user_upload/files/rus/common_info/gosa_doc_2022/Aeroflot_AR21_RUS_book_web.pdf (дата обращения 2023-12-12).

14. Aviasales – отзывы и описание принципа работы. – URL: <https://miroskop.ru/aviability/50-aviasales> (дата обращения 2023-12-12).

15. Авиакасса Яндекс.авиабилеты. Обзор и отзывы. – URL: <https://kajftur.ru/aviakassa/avia.yandex.ru.html> (дата обращения 2023-12-12).

16. Сервис «Яндекс.авиабилеты» стал доступен через мобильное приложение. – URL: <https://3dnews.ru/928991/servisyandeksaviabileti-stal-dostupen-cherez-mobilnoe-prilogenie> (дата обращения 2023-12-12).

17. Искусственный интеллект в авиации. Делает путешествия проще? – URL: https://dzen.ru/a/XVsLZUb0_wCtiktI (дата обращения 2023-12-12).

18. Роль искусственного интеллекта в авиационной промышленности. – URL: <https://ru.continuousdev.com/the-role-of-artificial-intelligence-in-the-aviation-industry-2-33247-12131> (дата обращения 2023-12-12).

19. Пассажирские авиаперевозки мигрируют из аналога в цифру. – URL: <https://cloud.vk.com/blog/passazhirskie-aviaperevozki-migriruyut-iz-analoga-v-cifru?ysclid=lnzobvz7j73480640> (дата обращения 2023-12-12).

20. Искусственный интеллект и авиаперелеты. – URL: <https://deeptech.net.io/archives/3409> (дата обращения 2023-12-12).

21. Компьютерное зрение в аэропортах: обнаружение багажа за счет искусственного интеллекта. – URL: <https://neuro-core.ru/blogs/airport-security-ai?ysclid=lnzoa2jdmk274067860> (дата обращения 2023-12-12).

22. Как технологии искусственного интеллекта могут ускорить предполетный досмотр в аэропорту? – URL: <https://aero.wekey.ru/blog/kak-tehnologii-iskusstvennogo-intellekta-mogut-uskorit-predpoletnyy-dosmotr-v-aeroportu.html?ysclid=lnzo9i09pi790238683> (дата обращения 2023-12-12).

23. Использование искусственного интеллекта в управлении частными самолетами и обслуживании клиентов. – URL: <https://compareprivateplanes.com/ru/articles/artificial-intelligence-in-private-jet-operations?ysclid=lnzo8kayq8479106439> (дата обращения 2023-12-12).

24. Blockchain и AI нацелились на авиацию. – URL: <https://habr.com/ru/articles/727060/> (дата обращения 2023-12-12).

25. Фальк К. Рекомендательные системы на практике / пер. с англ. Д.М. Павлова. – М.: ДМК Пресс, 2020. – 448 с.

26. Крачтен Ф. Введение в Rational Unified Process, 2-е издание // М.: Вильямс, 2002. – 240 с.
27. Документация по Microsoft SQL [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/sql/> (дата обращения 2023-05-04).
28. Основы Linq [Электронный ресурс]. – URL: <https://metanit.com/sharp/tutorial/15.1.php> (дата обращения 2023-05-04).
29. Документация по Xamarin [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/xamarin/> (дата обращения 2023-05-04).
30. Особенности рынка пассажирских и грузовых авиаперевозок в период пандемии [Электронный ресурс]. – URL: <https://www.aviastat.ru/statistics/108-osobennosti-rynka-passazhirskih-i-gruzovyh-aviaperevozok-v-period-pandemii> (дата обращения 2023-03-28).
31. Росстат [Электронный ресурс]. – URL: <https://rosstat.gov.ru/> (дата обращения 2023-03-28).
32. Пассажиропоток мировых аэропортов вырос в 2024 году на 9% / Aviastat [Электронный ресурс]. – 2025. – URL: <https://www.aviastat.ru/news/282562news-passazhiropotok-mirovyh-aeroportov-vyros-v-2024-godu-na-9> (дата обращения 2025-05-08).
33. Выполненный пассажирооборот / ЕМИСС. Государственная статистика [Электронный ресурс]. – 2023. – URL: <https://www.fedstat.ru/indicator/37020> (дата обращения 2023-10-17).
34. Global smartphone shipments market data [Электронный ресурс]. – URL: <https://www.counterpointresearch.com/insights/global-smartphone-share/> (дата обращения 2023-03-28).
35. Android Distribution Chart [Электронный ресурс]. – URL: <https://composables.com/android-distribution-chart> (дата обращения 2025-05-17).
36. Беркун С. Искусство управления IT-проектами. – СПб.: Питер, 2007. – 400 с.: ил.
37. Разу М.Л. Управление проектом. Основы проектного управления: учебник, 4-е издание. – М.: КНОРУС, 2015. – 756 с.

38. Гвоздев В.Е. Управление программными проектами: учебное пособие для вузов. – Москва: Издательство Юрайт, 2023. – 167 с.
39. Ехлаков Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие. – СПб.: Лань, 2019. – 244 с.
40. Иванов П.В. Управление проектами: Учебное пособие. – Рн/Д: Феникс, 2018. – 102 с.
41. Документация по Project. – URL: <https://learn.microsoft.com/ru-ru/project/> (дата обращения 2024-12-16).
42. Староверов И.М. Проектирование автоматизированных систем обслуживания авиапассажиров с применением современных информационных технологий // Наш выбор – наука!: Сборник статей IV Международного научно-исследовательского конкурса. – 2024. – С. 20 – 28.
43. Староверов И.М. Разработка онтологической модели для автоматизированной системы обслуживания авиапассажиров // Проблемы развития современного общества: Сборник научных статей 9-й Всероссийской национальной научно-практической конференции. – 2024. – Т. 3. – С. 125 – 129.
44. Луцкий М.Г. Разработка онтологии безопасности авиации // Инженерия программного обеспечения. – 2010. – № 4. – С. 56 – 63.
45. Особенности продажи авиабилетов в кассах. – URL: <https://agent.aero/blog/osobennosti-prodazhi-aviabiletov-v-kassah> (дата обращения 2024-05-16).
46. Романенко В.А. Приближённые модели авиатранспортных систем массового обслуживания с частичной взаимопомощью между обслуживающими каналами // Вестник Самарского университета. Аэрокосмическая техника, технологии и машиностроение. – 2022. – Т. 21. – № 4. – С. 118 – 130.
47. Вьюненко Л.Ф., Михайлов М.В., Первозванская Т.Н. Имитационное моделирование: учебник и практикум для вузов. – Москва: Издательство Юрайт, 2023. – 283 с.

48. Агапова Е.Г., Попова Т.М. Разработка имитационной модели участка транспортной сети в GPSS Studio // Международный журнал перспективных исследований. – 2021. – Т. 10. – № 4. – С. 145 – 150. doi:10.12731/2227-930X-2020-4-145-151.
49. Девятков В.В., Габалин А.В. Имитационное исследование бизнес-процессов с очередями с помощью среды моделирования GPSS Studio // Открытое образование. – 2020. – № 3 (24). – С. 67–77. doi.org/10.21686/1818-4243-2020-3-67-77.
50. Девятков В.В., Федотов М.В., Девятков Т.В. Имитационные исследования в среде моделирования GPSS STUDIO. Учебное пособие. – Москва: Вузовский учебник, 2019. – 283 с.
51. Богданова А.Ю., Теплянских Д.И., Выдрин В.А. Проведение валидации и верификации на примере имитационной модели телефонной станции // XIII Всероссийская научно-практическая конференция молодых ученых «РОССИЯ МОЛОДАЯ». – Кемерово: КузГТУ, 2021. – С. 31605.1 – 31605.8.
52. Лаврушина Е.Г., Гаевой С.С. Построение имитационной модели оптимизации количества сотрудников склада при отгрузке готовой продукции птицефабрики. – URL: <https://naukovedenie.ru/PDF/56EVN314.pdf> (дата обращения 2025-01-21).
53. Черушева Т.В., Зверовщикова Н.В. Теория массового обслуживания. – Пенза: Издательство ПГУ, 2021. – 224 с.
54. Рамбо Дж. UML 2.0. Объектно-ориентированное моделирование и разработка, 2-е издание. – СПб.: Питер, 2007. – 544 с.: ил.
55. Шаблон MVC [Электронный ресурс]. – URL: <https://docs.microsoft.com/ru-ru/aspnet/core/mvc> (дата обращения 2023-04-11).
56. Документация по ASP.NET [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/aspnet/core/> (дата обращения 2023-04-11).
57. Документация по Entity Framework [Электронный ресурс]. – URL: <https://docs.microsoft.com/ru-ru/ef> (дата обращения 2023-04-11).

58. Староверов И.М., Князев В.Н. Комплексный подход к созданию рекомендательной системы для авиаперелетов // Вестник Пензенского государственного университета. – 2024. – № 3. (В печати).
59. Kan H.Y., Dong D., Chau K. A Personalized Flight Recommender System Based on Link Prediction in Aviation Data // IEEE Access. – 2024. – Т. 12. – С. 36961 – 36973.
60. Muneer M., Rasheed U., Khalid S., Ahmad M. Tour Spot Recommendation System via Content-Based Filtering // 16th International Conference on Open Source Systems and Technologies (ICOSST). – 2022. – С. 1 – 6.
61. Kongpeng S., Hanskunatai A. Tourist Destination Recommendation System based on Machine Learning // Association for Computing Machinery. – 2024. – С. 58 – 67.
62. Гомзин А.Г., Коршунов А.В. Системы рекомендаций: обзор современных подходов. // Труды Института системного программирования РАН. – 2012. – С. 401 – 418.
63. Диденко А.А. Большие данные и рекомендательные системы // XII Всероссийская научно-практическая конференция молодых ученых «РОССИЯ МОЛОДАЯ». – 2020. – С. 21156.1 – 21156.7.
64. Погорельская Я.С. Обзор подходов к построению рекомендательных систем // Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем. – 2021. – № 1. – С. 278 – 282.
65. Понизовкин Д.М. Построение оптимального графа связей в системах коллаборативной фильтрации // Программные системы: теория и приложения. – 2011. – № 4. – С. 107 – 114.
66. Полякова О.С., Подлесный А.О. PageRank. Алгоритм ссылочного ранжирования // Наука и современность. – 2013. – № 20. – Т. 1. – С. 154 – 157.

67. Чистяков С.П. Случайные леса: обзор // Труды Карельского научного центра РАН. – 2013. – № 1. – С. 117 – 136.
68. Бородинов А.А. Разработка и исследование алгоритмов определения предпочитаемых пользователем остановок общественного транспорта в геоинформационной системе на основе методов машинного обучения // Компьютерная оптика. – 2020. – № 4. – Т. 44. – С. 646 – 652.
69. Введение в Web API [Электронный ресурс]. – URL: <https://metanit.com/sharp/mvc/12.1.php> (дата обращения 2025-03-23).
70. Руководство по Entity Framework [Электронный ресурс]. – URL: <https://metanit.com/sharp/efcore/> (дата обращения 2025-03-23).
71. Программный код и его метрики [Электронный ресурс]. – URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 2025-03-23).
72. Аверьянов А.В., Кошель И.Н., Кузнецов В.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.
73. Полякова Э.Р., Поляков А.А., Шустов А.Л., Придачкин Д.Г. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда // Cloud of Science. – 2020. – №4. – С. 950 – 959.

ПРИЛОЖЕНИЕ А
Исходный код имитационных моделей
(рекомендуемое)

Листинг А.1 – Исходный код имитационной модели обслуживания пассажиров кассирами

```

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

GENERATE      4,FN$XPDIS; Покупка билетов (1)
ASSIGN        1,1
ASSIGN        2,12
TRANSFER      ,service
GENERATE      6,FN$XPDIS; Бронирование билетов (2)
ASSIGN        1,2
ASSIGN        2,10
TRANSFER      ,service
GENERATE      7,FN$XPDIS; Выкуп брони (3)
ASSIGN        1,3
ASSIGN        2,4
TRANSFER      ,service
GENERATE      17,FN$XPDIS; Снятие брони (4)
ASSIGN        1,4
ASSIGN        2,2
TRANSFER      ,service
GENERATE      80,FN$XPDIS; Возврат билета (5)
ASSIGN        1,5
ASSIGN        2,3
TRANSFER      ,service
GENERATE      2,FN$XPDIS; Подтверждение бронирования билетов (6)
ASSIGN        1,6
ASSIGN        2,1
service       QUEUE       Och1
ENTER         Cashier
DEPART        Och1
ADVANCE       P2,FN$XPDIS; Обслуживание
LEAVE         Cashier
TABULATE      Tab; Время обслуживания клиента вне зависимости от типа
обслуживания
TEST E        P1,1,type2
TABULATE      Tab1; Время покупки билетов (1)
TRANSFER      ,finish
type2         TEST E      P1,2,type3
TABULATE      Tab2; Время бронирования билетов (2)
TRANSFER      ,finish
type3         TEST E      P1,3,type4
TABULATE      Tab3; Время выкупа брони (3)
TRANSFER      ,finish
type4         TEST E      P1,4,type5
TABULATE      Tab4; Время снятия брони (4)
TRANSFER      ,finish
type5         TEST E      P1,5,type6
TABULATE      Tab5; Время возврата билетов (5)
TRANSFER      ,finish
type6         TEST E      P1,6,finish

```

```

finish      TABULATE  Tab6; Время подтверждения бронирования билетов (6)
            TERMINATE

            GENERATE  3360; 7 дней / 1 минута
            TERMINATE 1
                START      1

```

Листинг А.2 – Исходный код имитационной модели обработки запросов сервером

```

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

    GENERATE  20,FN$XPDIS; Покупка билетов(1)
    ASSIGN 1,1
    TRANSFER ,next

    GENERATE  34,FN$XPDIS; Бронирование билетов(2)
    ASSIGN 1,2
TRANSFER ,next

    GENERATE  37,FN$XPDIS; Выкуп брони(3)
    ASSIGN 1,3
    TRANSFER ,work

GENERATE  83,FN$XPDIS; Снятие брони(4)
    ASSIGN 1,4
    TRANSFER ,work

    GENERATE  115,FN$XPDIS ; Возврат билета(5)
    ASSIGN 1,5
TRANSFER ,work

next QUEUE QH1
    SEIZE K1
    DEPART QH1
    ADVANCE  3,FN$XPDIS; Просмотр рейсов
    RELEASE K1
TRANSFER    .8,type

add_p  QUEUE QH2
    SEIZE K2
    DEPART QH2
    ADVANCE  1.5,FN$XPDIS; Добавление пассажира
    RELEASE K2
TRANSFER    .4,add_c,add_p

add_c  QUEUE QH3
    SEIZE K3
    DEPART QH3
    ADVANCE  1.5,FN$XPDIS; Добавление покупателя
    RELEASE K3

```

```

work QUEUE QH4
  ENTER Server
  DEPART QH4
  ADVANCE 2, FN$XPDIS; Работа сервера
  LEAVE Server

type TABULATE Tab

  TEST E P1,1,type2
  TABULATE Tab1
  TRANSFER ,finish

type2 TEST E P1,2,type3
  TABULATE Tab2
  TRANSFER ,finish

type3 TEST E P1,3,type4
  TABULATE Tab3
  TRANSFER ,finish

type4 TEST E P1,4,type5
  TABULATE Tab4
  TRANSFER ,finish

type5 TEST E P1,5,finish
  TABULATE Tab5

finish TERMINATE

GENERATE 604800 ; 7 дней / 1 секунда
  TERMINATE 1
  START 1

```

ПРИЛОЖЕНИЕ Б
Исходный код скриптов базы данных
(рекомендуемое)

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

```
CREATE TABLE Account (  
    id_account int IDENTITY NOT NULL CHECK(id_account > 0 and id_account <= 9999999),  
    login varchar(25) NOT NULL UNIQUE CHECK(login like replicate('[-a-zA-Z0-9!@#%]', len(login))),  
    password varchar(30) NOT NULL CHECK(password like replicate('[-a-zA-Z0-9!@#%]', len(password))),  
    PRIMARY KEY (id_account));  
  
CREATE TABLE Interests (  
    id_interests int IDENTITY NOT NULL,  
    beach_holiday tinyint NOT NULL CHECK(beach_holiday >= 1 and beach_holiday <= 10),  
    skiing tinyint NOT NULL CHECK(skiing >= 1 and skiing <= 10),  
    historical_places tinyint NOT NULL CHECK(historical_places >= 1 and historical_places <= 10),  
    PRIMARY KEY (id_interests));  
  
CREATE TABLE Viewing (  
    id_viewing int IDENTITY NOT NULL,  
    date datetime NOT NULL,  
    id_passenger int NOT NULL,  
    id_city_from int NOT NULL,  
    id_city_to int NOT NULL,  
    PRIMARY KEY (id_viewing));  
  
CREATE TABLE Recomendation (  
    id_recomendation int IDENTITY NOT NULL,  
    coeff float NOT NULL,  
    id_passenger int NOT NULL,  
    id_city int NOT NULL,  
    PRIMARY KEY (id_recomendation));  
  
CREATE TABLE Passenger (  
    id_passenger int IDENTITY NOT NULL,  
    telephone varchar(15) NOT NULL UNIQUE CHECK(telephone like replicate('[0-9+]', len(telephone)) and len(telephone) >= 12 and telephone like '+%'),  
    passport varchar(10) NOT NULL UNIQUE CHECK(passport like replicate('[IVXA-Я0-9]', len(passport))),  
    birth_date date NOT NULL CHECK(birth_date between '01.01.1910' and '31.12.2030'),  
    last_name varchar(30) NOT NULL CHECK(last_name like replicate('[a-яA-Я-]', len(last_name))),  
    first_name varchar(15) NOT NULL CHECK(first_name like replicate('[a-яA-Я-]', len(first_name))),  
    middle_name varchar(20) NULL CHECK(middle_name like replicate('[a-яA-Я-]', len(middle_name))),  
    gender varchar(1) NOT NULL CHECK(gender in ('м', 'ж')),  
    id_account int,  
    id_interests int NOT NULL,  
    PRIMARY KEY (id_passenger));  
  
CREATE TABLE Ticket (  
    id_ticket int IDENTITY NOT NULL,  
    price int NOT NULL CHECK(price >= 0 and price <= 9999999),  
    seat_plane varchar(3) NOT NULL CHECK(seat_plane like replicate('[a-z0-9]', len(seat_plane))),  
    id_passenger int NOT NULL,  
    id_buyer int NOT NULL,  
    id_schedule int NOT NULL,
```

```

    id_cashier int,
    PRIMARY KEY (id_ticket));

CREATE TABLE Booking (
    id_booking int IDENTITY NOT NULL,
    seat_plane varchar(3) NOT NULL CHECK(seat_plane like replicate('[a-z0-9]',
len(seat_plane))),
    id_passenger int NOT NULL,
    id_schedule int NOT NULL,
    id_cashier int,
    id_agent int NOT NULL,
    is_confirmed bit NOT NULL DEFAULT 1,
    PRIMARY KEY (id_booking));

CREATE TABLE Schedule (
    id_schedule int IDENTITY NOT NULL CHECK(id_schedule > 0 and id_schedule <=
9999),
    departue_time datetime NOT NULL CHECK(departue_time between '01.01.2010'
and '31.12.2040'),
    arrival_time datetime NOT NULL CHECK(arrival_time between '01.01.2010' and
'31.12.2040'),
    number_available_seats int CHECK(number_available_seats > 0 and
number_available_seats <= 999),
    id_status int NOT NULL,
    id_flight int NOT NULL,
    PRIMARY KEY (id_schedule));

CREATE TABLE Flight (
    id_flight int IDENTITY NOT NULL CHECK(id_flight > 0 and id_flight <= 9999),
    base_price int CHECK(base_price > 0 and base_price <= 999999),
    condition int,
    id_airplane int NOT NULL,
    id_route int NOT NULL,
    id_flighttype int NOT NULL,
    PRIMARY KEY (id_flight));

CREATE TABLE Flighttype (
    id_flighttype int IDENTITY NOT NULL CHECK(id_flighttype > 0 and
id_flighttype <= 9999),
    name_flighttype varchar(20) NOT NULL UNIQUE CHECK(name_flighttype like
replicate('[-a-яA-Яa-zA-Z ]', len(name_flighttype))),
    PRIMARY KEY (id_flighttype));

CREATE TABLE Airopport (
    id_airopport int IDENTITY NOT NULL CHECK(id_airopport > 0 and id_airopport <=
9999),
    name_airopport varchar(100) NOT NULL UNIQUE CHECK(name_airopport like
replicate('[( )-a-яA-Яa-zA-Z0-9. ]', len(name_airopport))),
    id_city int NOT NULL,
    PRIMARY KEY (id_airopport));

CREATE TABLE City (
    id_city int IDENTITY NOT NULL CHECK(id_city > 0 and id_city <= 9999),
    name_city varchar(20) NOT NULL UNIQUE CHECK(name_city like replicate('[-a-
яA-Яa-zA-Z ]', len(name_city))),
    PRIMARY KEY (id_city));

CREATE TABLE Cashier (
    id_cashier int IDENTITY NOT NULL CHECK(id_cashier > 0 and id_cashier <=
9999),
    last_name varchar(30) NOT NULL CHECK(last_name like replicate('[a-яA-Я-]',
len(last_name))),
    first_name varchar(15) NOT NULL CHECK(first_name like replicate('[a-яA-Я-
]', len(first_name))),

```

```

    middle_name varchar(20) NULL CHECK(middle_name like replicate('[a-яA-Я-]',
len(middle_name))),
    birth_date date NOT NULL CHECK(birth_date between '01.01.1940' and
'31.12.2030'),
    adress varchar(100) NOT NULL CHECK(adress like replicate('[a-яA-Я0-9-., ]',
len(adress))),
    telephone varchar(15) NOT NULL UNIQUE CHECK(telephone like replicate('[0-
9+]', len(telephone)) and len(telephone) >= 12 and telephone like '+%'),
    passport varchar(13) NOT NULL UNIQUE CHECK(passport like replicate('[-0-9A-
Z]', len(passport))),
    id_account int,
    PRIMARY KEY (id_cashier));

CREATE TABLE Airplane (
    id_airplane int IDENTITY NOT NULL CHECK(id_airplane > 0 and id_airplane <=
9999),
    number_seats int NOT NULL CHECK(number_seats > 0 and number_seats <= 999),
    name_airplane varchar(20) NOT NULL UNIQUE CHECK(name_airplane like
replicate('[-a-яA-Яa-zA-Z0-9 ]', len(name_airplane))),
    PRIMARY KEY (id_airplane));

CREATE TABLE Route (
    id_route int IDENTITY NOT NULL CHECK(id_route > 0 and id_route <= 9999),
    id_airoport_departure int NOT NULL,
    id_airoport_arrival int NOT NULL,
    PRIMARY KEY (id_route));

CREATE TABLE Status (
    id_status int IDENTITY NOT NULL CHECK(id_status > 0 and id_status <= 99),
    name_status varchar(30) NOT NULL UNIQUE CHECK(name_status like
replicate('[a-яA-Я- ]', len(name_status))),
    PRIMARY KEY (id_status));

ALTER TABLE Ticket ADD CONSTRAINT Buys FOREIGN KEY (id_passenger) REFERENCES
Passenger (id_passenger);
ALTER TABLE Booking ADD CONSTRAINT [Booking b] FOREIGN KEY (id_passenger)
REFERENCES Passenger (id_passenger);
ALTER TABLE Ticket ADD CONSTRAINT [Contains t] FOREIGN KEY (id_schedule)
REFERENCES Schedule (id_schedule);
ALTER TABLE Booking ADD CONSTRAINT [Contains b] FOREIGN KEY (id_schedule)
REFERENCES Schedule (id_schedule);
ALTER TABLE Route ADD CONSTRAINT FKRoute832681 FOREIGN KEY
(id_airoport_arrival) REFERENCES Airoport (id_airoport);
ALTER TABLE Airoport ADD CONSTRAINT Has FOREIGN KEY (id_city) REFERENCES City
(id_city);
ALTER TABLE Ticket ADD CONSTRAINT [Makes out t] FOREIGN KEY (id_cashier)
REFERENCES Cashier (id_cashier);
ALTER TABLE Booking ADD CONSTRAINT [Makes out b] FOREIGN KEY (id_cashier)
REFERENCES Cashier (id_cashier);
ALTER TABLE Flight ADD CONSTRAINT [Performs a] FOREIGN KEY (id_airplane)
REFERENCES Airplane (id_airplane);
ALTER TABLE Route ADD CONSTRAINT Runs FOREIGN KEY (id_airoport_departure)
REFERENCES Airoport (id_airoport);
ALTER TABLE Ticket ADD CONSTRAINT FKTicket963229 FOREIGN KEY (id_buyer)
REFERENCES Passenger (id_passenger);
ALTER TABLE Booking ADD CONSTRAINT FKBooking113727 FOREIGN KEY (id_agent)
REFERENCES Passenger (id_passenger);
ALTER TABLE Flight ADD CONSTRAINT [Performs r] FOREIGN KEY (id_route)
REFERENCES Route (id_route);
ALTER TABLE Flight ADD CONSTRAINT [Has f] FOREIGN KEY (id_flighttype)
REFERENCES Flighttype (id_flighttype);
ALTER TABLE Schedule ADD CONSTRAINT [Has s] FOREIGN KEY (id_status)
REFERENCES Status (id_status);

```

```

ALTER TABLE Schedule ADD CONSTRAINT [Contains f] FOREIGN KEY (id_flight)
REFERENCES Flight (id_flight);
ALTER TABLE Cashier ADD CONSTRAINT [Has a] FOREIGN KEY (id_account)
REFERENCES Account (id_account);
ALTER TABLE Passenger ADD CONSTRAINT [Has a2] FOREIGN KEY (id_account)
REFERENCES Account (id_account);
ALTER TABLE Passenger ADD CONSTRAINT [Has i] FOREIGN KEY (id_interests)
REFERENCES Interests (id_interests);
ALTER TABLE Viewing ADD CONSTRAINT [Is viewing] FOREIGN KEY (id_passenger)
REFERENCES Passenger (id_passenger);
ALTER TABLE Viewing ADD CONSTRAINT [Is contained f] FOREIGN KEY
(id_city_from) REFERENCES City (id_city);
ALTER TABLE Viewing ADD CONSTRAINT [Is contained t] FOREIGN KEY (id_city_to)
REFERENCES City (id_city);
ALTER TABLE Recomendation ADD CONSTRAINT [Receives] FOREIGN KEY
(id_passenger) REFERENCES Passenger (id_passenger);
ALTER TABLE Recomendation ADD CONSTRAINT [Is included] FOREIGN KEY (id_city)
REFERENCES City (id_city);

```

Листинг Б.2 – Исходный код создания представления PassengersFlightsCount (информация о количестве совершенных полетов пассажиров по месяцам)

```

create view PassengersFlightsCount as
with tbl (num)
as
(select 0 union select 1 union select 2 union select 3 union select 4
union select 5 union select 6 union select 7 union select 8 union
select 9 union select 10 union select 11)
select (tbl.num + 1 + (p.id_passenger - 1) * 12) as "id",
p.last_name "last_name",
p.first_name "first_name",
p.middle_name "middle_name",
DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -tbl.num, GETDATE()))), 0)
"start_date",
DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -tbl.num + 1, GETDATE()))),
0) - 1 "finish_date",
count (t.id_ticket) "flights_count"
from Ticket t
join Schedule s
on s.id_schedule = t.id_schedule
join Status st
on st.id_status = s.id_status
right join Passenger p
on p.id_passenger = t.id_passenger
cross join tbl
where st.name_status = 'Выполнен'
and s.departue_time >= DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -
tbl.num, GETDATE()))), 0)
and s.departue_time < DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -
tbl.num + 1, GETDATE()))), 0)
group by tbl.num, p.id_passenger, p.last_name, p.first_name, p.middle_name,
DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -tbl.num, GETDATE()))), 0),
DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -tbl.num + 1, GETDATE()))),
0)
union
select (tbl.num + 1 + (p.id_passenger - 1) * 12),
p.last_name,
p.first_name,
p.middle_name,
DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -tbl.num, GETDATE()))), 0),
DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -tbl.num + 1, GETDATE()))),
0) - 1,
0

```

```

from Ticket t
join Schedule s
on s.id_schedule = t.id_schedule
join Status st
on st.id_status = s.id_status
right join Passenger p
on p.id_passenger = t.id_passenger
cross join tbl
where not exists
(select 1
from Ticket t
join Schedule s
on s.id_schedule = t.id_schedule
join Status st
on st.id_status = s.id_status
right join Passenger p0
on p0.id_passenger = t.id_passenger
cross join tbl tb
where st.name_status = 'Выполнен'
and s.departue_time >= DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -
tbl.num, GETDATE()))), 0)
and s.departue_time < DATEADD(month, DATEDIFF(month, 0, DATEADD(month, -
tbl.num + 1, GETDATE()))), 0)
and p0.id_passenger = p.id_passenger
and tb.num = tbl.num);

```

Листинг Б.3 – Исходный код создания представления AverageFlightLoad (информация о средней загрузке рейсов по дням)

```

create view AverageFlightLoad as
with tbl (num)
as
(select 0 union select 1 union select 2 union select 3 union select 4
union select 5 union select 6)
select tbl.num as "id",
convert (date, GETDATE() - tbl.num) "date",
1 - convert (float, sum(s.number_available_seats)) / sum(a.number_seats)
"load"
from Schedule s join Flight f
on s.id_flight = f.id_flight
join Airplane a
on f.id_airplane = a.id_airplane
right join tbl
on convert (date, s.departue_time) = convert (date, GETDATE() - tbl.num)
and s.number_available_seats is not null
group by tbl.num, convert (date, GETDATE() - tbl.num);

```

Листинг Б.4 – Исходный код создания хранимой функции fullAge (получение целого числа лет, прошедшего между двумя датами)

```

create function fullAge
(@startdate date, @enddate datetime)
returns int
as
begin
return (datediff(year, @startdate, @enddate) -
case
when month(@startdate) < month(@enddate)
then 0
when month(@startdate) > month(@enddate)
then 1
when day(@startdate) > day(@enddate)

```

```

        then 1
        else 0
    end)
end;

```

Листинг Б.5 – Исходный код создания хранимой функции get_price (получение стоимости билетов с учетом льгот)

```

create function get_price
(@id_passenger int,
@id_schedule int)
returns int
as
begin
    declare @price int;
    select @price = case when dbo.fullAge (p.birth_date, getdate())
between 0 and 1 then 0
    when dbo.fullAge (p.birth_date, getdate()) between 2 and 11 then 0.5 *
f.base_price
    else f.base_price end
    from Flight f join Schedule s
    on f.id_flight = s.id_flight
    join Passenger p
    on p.id_passenger = @id_passenger
    where @id_schedule = s.id_schedule;
    return @price;
end;

```

Листинг Б.6 – Исходный код создания хранимой процедуры get_price_ticket (получение стоимости билетов с учетом льгот)

```

create procedure get_price_ticket
(@id_passenger int,
@id_schedule int,
@price int output)
as
begin
    select @price = case when dbo.fullAge (p.birth_date, getdate()) between
0 and 1 then 0
    when dbo.fullAge (p.birth_date, getdate()) between 2 and 11 then 0.5 *
f.base_price
    else f.base_price end
    from Flight f join Schedule s
    on f.id_flight = s.id_flight
    join Passenger p
    on p.id_passenger = @id_passenger
    where @id_schedule = s.id_schedule;
end;

```

Листинг Б.7 – Исходный код создания хранимой процедуры sell_ticket (продажа билета)

```

create procedure sell_ticket
(@seat_plane varchar(3), @id_passenger int, @id_buyer int, @id_schedule int,
@id_cashier int = null, @price int output)
as
begin
    declare @count_seats int;
    select @count_seats = number_available_seats from Schedule where
id_schedule = @id_schedule;
    if @count_seats < 1

```

```

begin
    set @price = -1;
    return;
end;

select @price = dbo.get_price (p.id_passenger, s.id_schedule)
from Schedule s join Passenger p
on p.id_passenger = @id_passenger
where @id_schedule = s.id_schedule;

insert into Ticket (price, seat_plane, id_passenger, id_buyer,
id_schedule, id_cashier)
values
    (@price, @seat_plane, @id_passenger, @id_buyer, @id_schedule,
    @id_cashier);
end;

```

Листинг Б.8 – Исходный код создания хранимой процедуры book_ticket (бронирование билета)

```

create procedure book_ticket
(@seat_plane varchar(3), @id_passenger int, @id_agent int, @id_schedule int,
    @id_cashier int = null)
as
begin
    declare @number_available_seats int, @number_seats int, @is_confirmed
    bit = 1;
    select @number_available_seats = number_available_seats from Schedule
    where id_schedule = @id_schedule;
    if @number_available_seats < 1
        return -1;
    select @number_seats = a.number_seats
    from Schedule s join Flight f on s.id_flight = f.id_flight
    join Airplane a on a.id_airplane = f.id_airplane
    if (cast(@number_available_seats as float) / @number_seats < 0.2)
        set @is_confirmed = 0;
    insert into Booking (seat_plane, id_passenger, id_agent, id_schedule,
    id_cashier, is_confirmed)
    values
        (@seat_plane, @id_passenger, @id_agent, @id_schedule, @id_cashier,
        @is_confirmed);
    return @number_available_seats - 1;
end;

```

Листинг Б.9 – Исходный код создания хранимой процедуры add_passenger (добавление пассажира или покупателя)

```

create procedure add_passenger
(@telephone varchar(15), @passport varchar(10), @birth_date date, @last_name
    varchar(30), @first_name varchar(15), @middle_name varchar(20), @id_account
    int = null)
as
begin
    if exists (select 1 from Passenger where passport = @passport and
    telephone = @telephone) and @id_account is not null
        update Passenger
        set id_account = @id_account
        where passport = @passport and telephone = @telephone;
    else
        insert into Passenger (telephone, passport, birth_date, last_name,
        first_name, middle_name, id_account, gender, id_interests)
        values

```

```

        (@telephone, @passport, @birth_date, @last_name, @first_name,
        @middle_name, @id_account, 'M', 1);
end;

```

Листинг Б.10 – Исходный код создания хранимой процедуры add_account (добавление аккаунта)

```

create procedure add_account
(@login varchar(25), @password varchar(30))
as
begin
    insert into Account (login, password)
    values
        (@login, @password);
end;

```

Листинг Б.11 – Исходный код создания хранимой процедуры change_details_passenger (изменение данных пассажира или покупателя)

```

create procedure change_details_passenger
(@id_passenger int, @telephone varchar(15), @passport varchar(10),
@birth_date date, @last_name varchar(30), @first_name varchar(15),
@middle_name varchar(20))
as
begin
    update Passenger
    set telephone = @telephone,
        passport = @passport,
        birth_date = @birth_date,
        last_name = @last_name,
        first_name = @first_name,
        middle_name = @middle_name
    where id_passenger = @id_passenger;
end;

```

Листинг Б.12 – Исходный код создания хранимой процедуры redeem_booking (выкуп брони)

```

create procedure redeem_booking
(@id_booking int,
@price int output)
as
begin
    declare @is_confirmed bit;
    select @is_confirmed = is_confirmed
    from Booking
    where id_booking = @id_booking;
    if @is_confirmed = 0
        return -1;

    declare @seat_plane varchar(3), @id_passenger int, @id_buyer int,
    @id_schedule int, @id_cashier int;
    select @seat_plane = b.seat_plane,
        @id_passenger = b.id_passenger,
        @id_buyer = b.id_agent,
        @id_schedule = b.id_schedule,
        @id_cashier = b.id_cashier,
        @price = dbo.get_price(p.id_passenger, s.id_schedule)
    from Booking b join Schedule s
    on b.id_schedule = s.id_schedule
    join Passenger p

```

```

        on p.id_passenger = b.id_passenger
        where @id_booking = b.id_booking;

        insert into Ticket(price, seat_plane, id_passenger, id_buyer,
id_schedule, id_cashier)
        values
        (@price, @seat_plane, @id_passenger, @id_buyer, @id_schedule,
@id_cashier);

        delete from Booking
        where id_booking = @id_booking;

        return 0;
end;

```

Листинг Б.13 – Исходный код создания хранимой процедуры cancel_booking (снятие брони)

```

create procedure cancel_booking
(@id_booking int)
as
begin
    delete from Booking
    where id_booking = @id_booking;
end;

```

Листинг Б.14 – Исходный код создания хранимой процедуры redeem_multiple_booking (выкуп множества билетов)

```

create procedure redeem_multiple_booking
(@id_schedule int, @id_agent int, @sum_price int output)
as
begin
    declare @seat_plane varchar(3), @id_passenger int, @id_buyer int,
    @id_cashier int, @id_booking int, @prev_id_booking int, @price int,
    @is_confirmed bit;
    set @id_booking = 0;
    set @prev_id_booking = 0;
    set @sum_price = 0;

    while (1 = 1)
    begin
        set @prev_id_booking = @id_booking;
        select top 1 @id_booking = b.id_booking,
        @seat_plane = b.seat_plane,
        @id_passenger = b.id_passenger,
        @id_buyer = b.id_agent,
        @id_cashier = b.id_cashier,
        @price = dbo.get_price(p.id_passenger, s.id_schedule)
        from Booking b join Schedule s
        on b.id_schedule = s.id_schedule
        join Passenger p
        on p.id_passenger = b.id_passenger
        where b.id_schedule = @id_schedule
        and b.id_agent = @id_agent;

        if (@id_booking = @prev_id_booking)
            return @sum_price;

        select @is_confirmed = is_confirmed
        from Booking
        where id_booking = @id_booking;
    end
end;

```

```

        if @is_confirmed = 0
            continue;

        insert into Ticket(price, seat_plane, id_passenger, id_buyer,
id_schedule, id_cashier)
            values
                (@price, @seat_plane, @id_passenger, @id_buyer, @id_schedule,
@id_cashier);

        delete from Booking
        where id_booking = @id_booking;

        set @sum_price = @sum_price + @price;
    end
end;

```

Листинг Б.15 – Исходный код создания хранимой процедуры cancel_multiple_booking (снятие множества броней)

```

create procedure cancel_multiple_booking
(@id_schedule int, @id_agent int)
as
begin
    delete from Booking
    where id_schedule = @id_schedule
    and id_agent = @id_agent;
end;

```

Листинг Б.16 – Исходный код создания хранимой процедуры refund_ticket_new (возврат билета)

```

create procedure refund_ticket
(@id_ticket int,
@price int output)
as
begin
    select @price = price from Ticket where id_ticket = @id_ticket;
    delete from Ticket
    where id_ticket = @id_ticket;
end;

```

Листинг Б.17 – Исходный код создания хранимой процедуры refund_multiple_ticket (возврат множества билетов)

```

create procedure refund_multiple_ticket
(@id_schedule int, @id_buyer int, @sum_price int output)
as
begin
    declare @id_ticket int, @prev_id_ticket int, @price int;
    set @id_ticket = 0;
    set @prev_id_ticket = 0;
    set @sum_price = 0;

    while (1 = 1)
    begin
        set @prev_id_ticket = @id_ticket;
        select top 1 @id_ticket = id_ticket,
@price = price
        from Ticket
        where id_schedule = @id_schedule
        and id_buyer = @id_buyer;
    end
end;

```

```

        if (@id_ticket = @prev_id_ticket)
            return @sum_price;

        delete from Ticket
        where id_ticket = @id_ticket;
        set @sum_price = @sum_price + @price;
    end
end;

```

Листинг Б.18 – Исходный код создания хранимой процедуры get_flights (получение списка рейсов по заданным параметрам)

```

create procedure get_flights
@city_from varchar(20),
@city_to varchar(20),
@date date
as
select s.id_schedule,
ad.name_airoport name_airoport_from,
aa.name_airoport name_airoport_to,
s.departue_time,
s.arrival_time,
air.name_airplane,
f.base_price,
s.number_available_seats
from Schedule s join Flight f
on s.id_flight = f.id_flight
join Route r
on r.id_route = f.id_route
join Airopport ad
on ad.id_airoport = r.id_airoport_departure
join Airopport aa
on aa.id_airoport = r.id_airoport_arrival
join Airplane air
on air.id_airplane = f.id_airplane
join City cd
on cd.id_city = ad.id_city
join City ca
on ca.id_city = aa.id_city
join Flighttype ft
on ft.id_flighttype = f.id_flighttype
join Status st
on st.id_status = s.id_status
where ft.name_flighttype not in ('Грузоперевозки', 'Специальный рейс')
and (st.name_status = 'По расписанию' or st.name_status = 'Задерживается')
and cd.name_city = @city_from
and ca.name_city = @city_to
and convert(date, s.departue_time) = @date;

```

ПРИЛОЖЕНИЕ В
Исходный код серверной части
(рекомендуемое)

Листинг В.1 – Файл Cashier.cs

```
namespace Airopport.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Cashier
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Cashier()
        {
            this.Booking = new HashSet<Booking>();
            this.Ticket = new HashSet<Ticket>();
        }

        public int id_cashier { get; set; }
        public string last_name { get; set; }
        public string first_name { get; set; }
        public string middle_name { get; set; }
        public System.DateTime birth_date { get; set; }
        public string address { get; set; }
        public string telephone { get; set; }
        public string passport { get; set; }
        public Nullable<int> id_account { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Booking> Booking { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Ticket> Ticket { get; set; }
        public virtual Account Account { get; set; }
    }
}
```

Листинг В.2 – Файл Ticket.cs

```
namespace Airopport.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Ticket
    {
        public int id_ticket { get; set; }
        public int price { get; set; }
        public string seat_plane { get; set; }
        public int id_passenger { get; set; }
        public int id_buyer { get; set; }
        public int id_schedule { get; set; }
        public Nullable<int> id_cashier { get; set; }

        public virtual Cashier Cashier { get; set; }
        public virtual Passenger Passenger { get; set; }
        public virtual Passenger Passenger1 { get; set; }
        public virtual Schedule Schedule { get; set; }
    }
}
```

Листинг В.3 – Файл Booking.cs

```
namespace Airopport.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Booking
    {
        public int id_booking { get; set; }
        public string seat_plane { get; set; }
        public int id_passenger { get; set; }
        public int id_schedule { get; set; }
        public Nullable<int> id_cashier { get; set; }
        public int id_agent { get; set; }
        public bool is_confirmed { get; set; }

        public virtual Passenger Passenger { get; set; }
        public virtual Schedule Schedule { get; set; }
        public virtual Passenger Passenger1 { get; set; }
        public virtual Cashier Cashier { get; set; }
    }
}
```

Листинг В.4 – Файл AverageFlightLoad.cs

```
namespace Airopport.Models
{
    using System;
    using System.Collections.Generic;

    public partial class AverageFlightLoad
    {
        public int id { get; set; }
        public Nullable<System.DateTime> date { get; set; }
        public Nullable<double> load { get; set; }
    }
}
```

Листинг В.5 – Файл PassengersFlightsCount.cs

```
namespace Airopport.Models
{
    using System;
    using System.Collections.Generic;

    public partial class PassengersFlightsCount
    {
        public int id { get; set; }
        public string last_name { get; set; }
        public string first_name { get; set; }
        public string middle_name { get; set; }
        public Nullable<System.DateTime> start_date { get; set; }
        public Nullable<System.DateTime> finish_date { get; set; }
        public Nullable<int> flights_count { get; set; }
    }
}
```

Листинг В.6 – Файл BookingsController.cs

```
using System.Collections.Generic;
using System.Data.Entity.Core.Objects;
using System.Linq;
using System.Net;
using System.Web.Mvc;
using Airopport.Models;

namespace Airopport.Controllers
{
    public class BookingsController : Controller
    {
        private AiropportEntities db = new AiropportEntities();

        // POST: Bookings/Create
        [HttpPost]
        public ActionResult Create(string seat_plane, int id_passenger, int
id_buyer, int id_schedule, int priority)
        {
            int cashierId = 0;
            if (priority == 0)
            {
                var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
                if (string.IsNullOrEmpty(token))
                {
                    return new
HttpStatusCodeResult(HttpStatusCode.BadRequest, "Missing or invalid
authorization token");
                }

                int accountId = JwtService.GetItem(token);
                if (accountId < 0)
                {
                    return HttpNotFound();
                }
                cashierId = db.Cashier.FirstOrDefault(c => c.id_account ==
accountId).id_cashier;
            }

            int count;
            if (priority == 0)
            {
                count = db.book_ticket(seat_plane, id_passenger, id_buyer,
id_schedule, cashierId);
            }
            else
            {
                count = db.book_ticket(seat_plane, id_passenger, id_buyer,
id_schedule, null);
            }
            if (count > 0)
            {
                return new EmptyResult();
            }
            else
            {
                return HttpNotFound();
            }
        }
    }
}
```

```

// GET: Bookings/GetPriceIdBooking/5
public ActionResult GetPriceIdBooking(int? id_booking)
{
    if (id_booking == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }

    var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
    if (string.IsNullOrEmpty(token))
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest,
"Missing or invalid authorization token");
    }

    int accountId = JwtService.GetItem(token);
    if (accountId < 0)
    {
        return HttpNotFound();
    }
    Cashier cashier = db.Cashier.FirstOrDefault(c => c.id_account ==
accountId);
    dynamic result;
    if (cashier == null)
    {
        Passenger passenger = db.Passenger.FirstOrDefault(p =>
p.id_account == accountId);
        if (passenger == null)
        {
            return HttpNotFound();
        }
        result = (from s in db.Schedule
                    join b in db.Booking on s.id_schedule equals
b.id_schedule
                    join p in db.Passenger on b.id_passenger equals
p.id_passenger
                    where b.id_booking == id_booking && b.id_agent ==
passenger.id_passenger
                    select new { PassengerId = p.id_passenger,
ScheduleId = s.id_schedule })
                    .FirstOrDefault();
    }
    else
    {
        result = (from s in db.Schedule
                    join b in db.Booking on s.id_schedule equals
b.id_schedule
                    join p in db.Passenger on b.id_passenger equals
p.id_passenger
                    where b.id_booking == id_booking
                    select new { PassengerId = p.id_passenger,
ScheduleId = s.id_schedule })
                    .FirstOrDefault();
    }
    if (result == null)
    {
        return new HttpNotFoundResult();
    }

    int id_passenger = result.PassengerId;
    int id_schedule = result.ScheduleId;
    ObjectParameter price = new ObjectParameter("price",
typeof(int));

```

```

        db.get_price_ticket(id_passenger, id_schedule, price);
        return Json(price.Value, JsonRequestBehavior.AllowGet);
    }

    // GET: Bookings/GetPriceIdSchedule/5
    public ActionResult GetPriceIdSchedule(int? id_schedule, string
passportAgent)
    {
        if (id_schedule == null || passportAgent == "")
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }

        Passenger agent = db.Passenger.FirstOrDefault(p => p.passport ==
passportAgent);
        if (agent == null)
        {
            return new HttpNotFoundResult();
        }

        IList<int> id_passengers = (from b in db.Booking
                                where b.id_agent == agent.id_passenger &&
b.id_schedule == id_schedule
                                select b.id_passenger).ToList();
        if (id_passengers.Count == 0)
        {
            return new HttpNotFoundResult();
        }
        int sum_price = 0;
        foreach (int id_passenger in id_passengers)
        {
            ObjectParameter price = new ObjectParameter("price",
typeof(int));
            db.get_price_ticket(id_passenger, id_schedule, price);
            sum_price += (int)price.Value;
        }
        return Json(sum_price, JsonRequestBehavior.AllowGet);
    }

    // DELETE: Bookings/Cancel/5
    [HttpDelete, ActionName("Cancel")]
    public ActionResult CancelConfirmed(int id_booking, int priority)
    {
        if (priority >= 1)
        {
            var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
            if (string.IsNullOrEmpty(token))
            {
                return new
HttpStatusCodeResult(HttpStatusCode.BadRequest, "Missing or invalid
authorization token");
            }

            int accountId = JwtService.GetItem(token);
            if (accountId < 0)
            {
                return HttpNotFound();
            }
            int passengerId = db.Passenger.FirstOrDefault(p =>
p.id_account == accountId).id_passenger;
            if (db.Booking.FirstOrDefault(b => b.id_booking == id_booking
&& b.id_agent == passengerId) == null)
            {

```

```

        return new HttpUnauthorizedResult();
    }
}
int count = db.cancel_booking(id_booking);
if (count > 0)
{
    return new EmptyResult();
}
else
{
    return HttpNotFound();
}
}

// POST: Bookings/Redeem
[HttpPost, ActionName("Redeem")]
public ActionResult RedeemConfirmed(int id_booking, int priority)
{
    if (priority >= 1)
    {
        var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
        if (string.IsNullOrEmpty(token))
        {
            return new
HttpStatusCodeResult(HttpStatusCode.BadRequest, "Missing or invalid
authorization token");
        }

        int accountId = JwtService.GetItem(token);
        if (accountId < 0)
        {
            return HttpNotFound();
        }
        int passengerId = db.Passenger.FirstOrDefault(p =>
p.id_account == accountId).id_passenger;
        if (db.Booking.FirstOrDefault(b => b.id_booking == id_booking
&& b.id_agent == passengerId) == null)
        {
            return new HttpUnauthorizedResult();
        }
    }
    try
    {
        ObjectParameter price = new ObjectParameter("price",
typeof(int));
        int count = db.redeem_booking(id_booking, price);
        if (count > 0)
        {
            return Json(price.Value, JsonRequestBehavior.AllowGet);
        }
        else
        {
            return HttpNotFound();
        }
    }
    catch
    {
        return HttpNotFound();
    }
}

// DELETE: Bookings/CancelMultiple/
[HttpDelete, ActionName("CancelMultiple")]

```

```

        public ActionResult CancelMultipleConfirmed(int id_schedule, string
passport, int priority)
        {
            if (priority >= 1)
            {
                var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
                if (string.IsNullOrEmpty(token))
                {
                    return new
HttpStatusCodeResult(HttpStatusCode.BadRequest, "Missing or invalid
authorization token");
                }

                int accountId = JwtService.GetItem(token);
                if (accountId < 0)
                {
                    return HttpNotFound();
                }
                int passengerId = db.Passenger.FirstOrDefault(p =>
p.id_account == accountId).id_passenger;
                if (db.Booking.FirstOrDefault(b => b.id_schedule ==
id_schedule && b.id_agent == passengerId) == null)
                {
                    return new HttpUnauthorizedResult();
                }
            }
            Passenger passenger = db.Passenger.FirstOrDefault(p => p.passport
== passport);
            int count = db.cancel_multiple_booking(id_schedule,
passenger.id_passenger);
            if (count > 0)
            {
                return Json(count - 1, JsonRequestBehavior.AllowGet);
            }
            else
            {
                return HttpNotFound();
            }
        }

        // POST: Bookings/RedeemMultiple
        [HttpPost, ActionName("RedeemMultiple")]
        public ActionResult RedeemMultipleConfirmed(int id_schedule, string
passport)
        {
            try
            {
                Passenger passenger = db.Passenger.FirstOrDefault(p =>
p.passport == passport);
                ObjectParameter sum_price = new ObjectParameter("sum_price",
typeof(int));
                int count = db.redeem_multiple_booking(id_schedule,
passenger.id_passenger, sum_price);
                if (count > 0)
                {
                    return Json((count / 4, (int)sum_price.Value),
JsonRequestBehavior.AllowGet);
                }
                else
                {
                    return HttpNotFound();
                }
            }
        }

```

```

        catch
        {
            return HttpNotFound();
        }
    }

    // GET: Bookings/NotConfirmedBookings
    public ActionResult NotConfirmedBookings()
    {
        var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
        if (string.IsNullOrEmpty(token))
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest,
"Missing or invalid authorization token");
        }

        int accountId = JwtService.GetItem(token);
        if (accountId < 0 || db.Cashier.FirstOrDefault(c => c.id_account
== accountId) == null)
        {
            return HttpNotFound();
        }
        var bookings = db.BookingsInformation.Where(bi => bi.is_confirmed
== false).ToList();
        return Json(bookings, JsonRequestBehavior.AllowGet);
    }

    // PUT: Bookings/ConfirmBooking
    [HttpPut]
    public ActionResult ConfirmBooking(int id_booking)
    {
        var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
        if (string.IsNullOrEmpty(token))
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest,
"Missing or invalid authorization token");
        }

        int accountId = JwtService.GetItem(token);
        if (accountId < 0 || db.Cashier.FirstOrDefault(c => c.id_account
== accountId) == null)
        {
            return HttpNotFound();
        }
        var booking = db.Booking.FirstOrDefault(b => b.id_booking ==
id_booking);
        if (booking != null)
        {
            booking.is_confirmed = true;
            db.SaveChanges();
        }
        else
        {
            return HttpNotFound();
        }
        return new EmptyResult();
    }

    protected override void Dispose(bool disposing)
    {
        if (disposing)
        {

```

```

        db.Dispose();
    }
    base.Dispose(disposing);
}
}
}

```

Листинг В.7 – Файл CashiersController.cs

```

using System.Text;
using System.Web.Mvc;
using Airopport.Models;
using Airopport.Properties;
using Microsoft.IdentityModel.Tokens;

namespace Airopport.Controllers
{
    public class CashiersController : Controller
    {
        private AiropportEntities db = new AiropportEntities();

        // GET: Cashiers/Details
        public ActionResult Details()
        {
            var token =
Request.Headers["Authorization"]?.ToString().Replace("Bearer ", "");
            if (string.IsNullOrEmpty(token))
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest,
"Missing or invalid authorization token");
            }

            var cashierId = JwtService.GetItem(token);
            if (cashierId < 0)
            {
                return HttpNotFound();
            }
            Cashier cashier = db.Cashier.Find(cashierId);

            cashier.Booking.Clear();
            cashier.Ticket.Clear();
            if (cashier.Account != null)
            {
                cashier.Account.Cashier.Clear();
                cashier.Account.Passenger.Clear();
            }
            return Json(cashier, JsonRequestBehavior.AllowGet);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}

```

ПРИЛОЖЕНИЕ Г
Исходный код клиентской части
(рекомендуемое)

Листинг Г.1 – Файл AppShell.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Shell xmlns="http://xamarin.com/schemas/2014/forms"
        xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:d="http://xamarin.com/schemas/2014/forms/design"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:view="clr-namespace:Airopport.Views"
        mc:Ignorable="d"
        x:Class="Airopport.AppShell">
    <TabBar>
        <ShellContent Title="Продажа"
            Icon="sell.png"
            Route="SellBook"
            ContentTemplate="{DataTemplate view:SellBook}" />
        <ShellContent Title="Выкуп брони"
            Icon="redeem.png"
            Route="RedeemCancelBooking"
            ContentTemplate="{DataTemplate
view:RedeemCancelBooking}" />
        <ShellContent Title="Возврат билета"
            Icon="ticket.png"
            Route="RefundTicket"
            ContentTemplate="{DataTemplate view:RefundTicket}" />
        <ShellContent Title="Другое"
            Icon="other.png"
            Route="Other"
            ContentTemplate="{DataTemplate view:Other}" />
    </TabBar>

    <TabBar>
        <ShellContent Title="Ранее учтенный"
            Icon="old_passenger.png"
            Route="AddAccountedPassenger"
            ContentTemplate="{DataTemplate
view:AddAccountedPassenger}" />
        <ShellContent Title="Новый"
            Icon="new_passenger.png"
            Route="AddPassenger"
            ContentTemplate="{DataTemplate view:AddPassenger}" />
    </TabBar>

    <TabBar>
        <ShellContent Title="Ранее учтенный"
            Icon="old_passenger.png"
            Route="AddAccountedBuyer"
            ContentTemplate="{DataTemplate view:AddAccountedBuyer}"
/>
        <ShellContent Title="Новый"
            Icon="new_passenger.png"
            Route="AddBuyer"
            ContentTemplate="{DataTemplate view:AddBuyer}" />
    </TabBar>

    <ShellContent Route="MainPage"
        ContentTemplate="{DataTemplate view:MainPage}" />
    <ShellContent Route="Register"
        ContentTemplate="{DataTemplate view:Register}" />
</Shell>
```

Листинг Г.2 – Файл AppShell.xaml.cs

```
using Airopport.Views;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace Airopport
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class AppShell : Shell
    {
        public AppShell()
        {
            InitializeComponent();

            Routing.RegisterRoute(nameof(Flights), typeof(Flights));
            Routing.RegisterRoute(nameof(IssueSellBook),
typeof(IssueSellBook));
            Routing.RegisterRoute(nameof(AnaliticView),
typeof(AnaliticView));
            Routing.RegisterRoute(nameof(PassengerChoice),
typeof(PassengerChoice));
            Routing.RegisterRoute(nameof(ChangeDetailsPassenger),
typeof(ChangeDetailsPassenger));
            Routing.RegisterRoute(nameof(Confirm), typeof(Confirm));
            Routing.RegisterRoute(nameof(TicketsBookingsList),
typeof(TicketsBookingsList));
            Routing.RegisterRoute(nameof(Payment), typeof(Payment));
        }
    }
}
```

Листинг Г.3 – Файл Service.cs

```
using Airopport.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Essentials;

namespace Airopport.Services
{
    public static class Service
    {
        private static string url = "http://192.168.0.245:44317/";
        private static HttpClientHandler ClientHandler = new
HttpClientHandler();
        private static HttpClient Client = new HttpClient(ClientHandler);

        public async static Task<bool> LoginedAsync()
        {
            var token = await SecureStorage.GetAsync("jwt_token");
            if (token is null)
            {
                return false;
            }
            else
            {
            }
        }
    }
}
```

```

        {
            return true;
        }
    }

    public async static Task<bool> LoginAsync(string login, string
password)
    {
        var content = new StringContent("{\"login\":\"" + login + "\""
+ ", \"password\":\"" + password + "\""
, Encoding.UTF8, "application/json");
        var response = await Client.PostAsync(url + "Accounts/Login",
content);

        if (response.IsSuccessStatusCode) {
            string responseText = await
response.Content.ReadAsStringAsync();
            var tokenResponse =
JsonConvert.DeserializeObject<String>(responseText);
            await SecureStorage.SetAsync("jwt_token", tokenResponse);
            return true;
        }
        return false;
    }

    public async static Task<bool> RegisterAsync(string login, string
password)
    {
        var content = new StringContent("{\"login\":\"" + login + "\""
+ ", \"password\":\"" + password + "\""
, Encoding.UTF8, "application/json");
        var response = await Client.PostAsync(url + "Accounts/Register",
content);

        if (response.IsSuccessStatusCode)
        {
            string responseText = await
response.Content.ReadAsStringAsync();
            var tokenResponse =
JsonConvert.DeserializeObject<String>(responseText);
            await SecureStorage.SetAsync("jwt_token", tokenResponse);
            return true;
        }
        return false;
    }

    public async static Task LogoutAsync()
    {
        SecureStorage.Remove("jwt_token");
    }

    public async static Task<Cashier> GetCashierAsync()
    {
        var token = await SecureStorage.GetAsync("jwt_token");
        Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
        Cashier cashier = null;
        var response = await Client.GetAsync(url + "Cashiers/Details");

        if (response.IsSuccessStatusCode)
        {
            string responseText = await
response.Content.ReadAsStringAsync();

```

```

        cashier
    JsonConvert.DeserializeObject<Cashier>(responseText);
    }
    return cashier;
}

public async static Task<List<Flight>> GetFlightsAsync(string
city_from, string city_to, DateTime date, int priority, int idPassenger)
{
    List<Flight> flights = null;
    string date_str = date.Date.ToString("yyyy-MM-dd");
    var response = await Client.GetAsync(url +
"Flights/Available?city_from=" + city_from + "&city_to=" + city_to + "&date="
+ date_str
+ "&priority="
+ priority + "&id_passenger=" + idPassenger);

    if (response.IsSuccessStatusCode)
    {
        string responseText = await
response.Content.ReadAsStringAsync();
        flights =
JsonConvert.DeserializeObject<List<Flight>>(responseText);
    }
    return flights;
}

public static async Task<List<string>> GetCitiesAsync()
{
    List<City> cities = new List<City>();
    List<string> name_cities = new List<string>();
    var response = await Client.GetAsync(url + "Cities");

    if (response.IsSuccessStatusCode)
    {
        string responseText = await
response.Content.ReadAsStringAsync();
        cities =
JsonConvert.DeserializeObject<List<City>>(responseText);
        foreach(City city in cities)
        {
            name_cities.Add(city.name_city);
        }
    }
    return name_cities;
}

public static async Task<List<string>> GetRecommendationsAsync(int
id_passenger, int n)
{
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);

    var response = await Client.GetAsync(url +
$"Cities/Recommendations?id_passenger={id_passenger}&n={n}");
    if (response.IsSuccessStatusCode)
    {
        string responseText = await
response.Content.ReadAsStringAsync();
        List<string> recommendations =
JsonConvert.DeserializeObject<List<string>>(responseText);
        return recommendations;
    }
}

```

```

        else
        {
            return new List<string> { "Москва", "Пенза", "Санкт-
Петербург" };
        }

        public static async Task<List<string>> GetAvailableSeatsAsync(int
id_schedule, string[] occupidSeats)
        {
            List<string> availableSeats = new List<string>();
            var response = await Client.GetAsync(url + "Schedules/GetSeats/"
+ id_schedule);

            if (response.IsSuccessStatusCode)
            {
                string responseText = await
response.Content.ReadAsStringAsync();
                availableSeats =
JsonConvert.DeserializeObject<List<string>>(responseText);
                foreach (string occupidSeat in occupidSeats)
                {
                    availableSeats.Remove(occupidSeat);
                }
            }
            return availableSeats;
        }

        public static async Task<int> GetPriorityAsync()
        {
            var token = await SecureStorage.GetAsync("jwt_token");
            Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
            var response = await Client.GetAsync(url +
"Accounts/GetPriority");
            if (response.IsSuccessStatusCode)
            {
                string responseText = await
response.Content.ReadAsStringAsync();
                int priority =
JsonConvert.DeserializeObject<int>(responseText);
                return priority;
            }
            else
            {
                return -1;
            }
        }

        public static async Task<int> AddPassengerAsync(string telephone,
string passport, DateTime birth_date, string last_name, string first_name,
string middle_name)
        {
            int priority = await GetPriorityAsync();
            if (priority < 0)
            {
                return -1;
            }

            string date_str = birth_date.Date.ToString("yyyy-MM-dd");
            var content = new StringContent("{ \"telephone\": \"" + telephone +
"\"
            + ", \"passport\": \"" + passport + "\"
            + ", \"birth_date\": \"" + date_str + "\"

```

```

        + ", \"last_name\": \"" + last_name + "\""
        + ", \"first_name\": \"" + first_name + "\""
        + ", \"middle_name\": \"" + middle_name + "\""
        + ", \"priority\": \"" + priority + "\"}",
        Encoding.UTF8, "application/json");
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
    var response = await Client.PostAsync(url + "Passengers/Create",
content);

    if (response.IsSuccessStatusCode)
    {
        return 0;
    }
    return -1;
}

public static async Task<int> GetPassengerIdAsync(string passport)
{
    Passenger passenger = null;
    var response = await Client.GetAsync(url +
"Passengers/DetailsPassport?passport=" + passport);

    if (response.IsSuccessStatusCode)
    {
        string responseText = await
response.Content.ReadAsStringAsync();
        passenger =
JsonConvert.DeserializeObject<Passenger>(responseText);
    }
    return passenger.id_passenger;
}

public static async Task<Passenger> GetPassengerAsync(int
id_passenger)
{
    Passenger passenger = null;
    var response = await Client.GetAsync(url +
"Passengers/DetailsId/" + id_passenger);

    if (response.IsSuccessStatusCode)
    {
        string responseText = await
response.Content.ReadAsStringAsync();
        passenger =
JsonConvert.DeserializeObject<Passenger>(responseText);
    }
    return passenger;
}

public static async Task<int> GetYourselfPassengerIdAsync()
{
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
    var response = await Client.GetAsync(url +
"Passengers/DetailsPassport");

    if (response.IsSuccessStatusCode)
    {
        string responseText = await
response.Content.ReadAsStringAsync();

```

```

        Passenger passenger =
        JsonConvert.DeserializeObject<Passenger>(responseText);
        return passenger.id_passenger;
    }
    return -1;
}

public static async Task<Passenger> GetYourselfPassengerAsync()
{
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
    AuthenticationHeaderValue("Bearer", token);
    Passenger passenger = null;
    var response = await Client.GetAsync(url +
    "Passengers/DetailsPassport");

    if (response.IsSuccessStatusCode)
    {
        string responseText = await
        response.Content.ReadAsStringAsync();
        passenger =
        JsonConvert.DeserializeObject<Passenger>(responseText);
    }
    return passenger;
}

public static async Task<int> SellTicketsAsync()
{
    if (!Purchase.IsComplete())
    {
        return -1;
    }
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
    AuthenticationHeaderValue("Bearer", token);
    int id_buyer = Purchase.id_buyer, id_schedule =
    Purchase.id_schedule;
    Dictionary<int, string> passengers = Purchase.passengers;
    int sumPrice = 0;

    int priority = await GetPriorityAsync();
    foreach (int id in passengers.Keys)
    {
        var content = new StringContent("{\"seat_plane\":\"" +
        passengers[id] + "\"
        + ", \"id_passenger\":\"" + id + "\"
        + ", \"id_buyer\":\"" + id_buyer + "\"
        + ", \"id_schedule\":\"" + id_schedule + "\"
        + ", \"priority\":\"" + priority + "\"}",
        Encoding.UTF8, "application/json");
        var response = await Client.PostAsync(url + "Tickets/Create",
        content);

        if (!response.IsSuccessStatusCode)
        {
            return -1;
        }
        string responseText = await
        response.Content.ReadAsStringAsync();
        int price = JsonConvert.DeserializeObject<int>(responseText);
        sumPrice += price;
    }
    return sumPrice;
}

```

```

public static async Task<bool> BookTicketsAsync()
{
    if (!Purchase.IsComplete())
    {
        return false;
    }
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
    int id_buyer = Purchase.id_buyer, id_schedule =
Purchase.id_schedule;
    Dictionary<int, string> passengers = Purchase.passengers;

    int priority = await GetPriorityAsync();
    foreach (int id in passengers.Keys)
    {
        var content = new StringContent("{\"seat_plane\":\"" +
passengers[id] + "\"
        + ", \"id_passenger\":\"" + id + "\"
        + ", \"id_buyer\":\"" + id_buyer + "\"
        + ", \"id_schedule\":\"" + id_schedule + "\"
        + ", \"priority\":\"" + priority + "\"}",
        Encoding.UTF8, "application/json");
        var response = await Client.PostAsync(url +
"Bookings/Create", content);

        if (!response.IsSuccessStatusCode)
        {
            return false;
        }
    }
    return true;
}

public static async Task<int> GetPriceAsync()
{
    if (!Purchase.IsComplete())
    {
        return -1;
    }
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
    int id_schedule = Purchase.id_schedule;
    Dictionary<int, string> passengers = Purchase.passengers;
    int sumPrice = 0;

    foreach (int id in passengers.Keys)
    {
        var response = await Client.GetAsync(url +
"Tickets/GetPrice?id_passenger=" + id + "&id_schedule=" + id_schedule);
        if (!response.IsSuccessStatusCode)
        {
            return -1;
        }
        string responseText = await
response.Content.ReadAsStringAsync();
        int price = JsonConvert.DeserializeObject<int>(responseText);
        sumPrice += price;
    }
    return sumPrice;
}

```

```

        public static async Task<int> GetPriceAsync(int id_booking)
        {
            var token = await SecureStorage.GetAsync("jwt_token");
            Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
            var response = await Client.GetAsync(url +
$"Bookings/GetPriceIdBooking?id_booking={id_booking}");
            if (!response.IsSuccessStatusCode)
            {
                return -1;
            }
            string responseText = await response.Content.ReadAsStringAsync();
            int price = JsonConvert.DeserializeObject<int>(responseText);
            return price;
        }

        public static async Task<int> GetPriceAsync(int id_schedule, string
passport)
        {
            var token = await SecureStorage.GetAsync("jwt_token");
            Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
            var response = await Client.GetAsync(url +
$"Bookings/GetPriceIdSchedule?id_schedule={id_schedule}&passportAgent={passpo
rt}");
            if (!response.IsSuccessStatusCode)
            {
                return -1;
            }
            string responseText = await response.Content.ReadAsStringAsync();
            int price = JsonConvert.DeserializeObject<int>(responseText);
            return price;
        }

        public static Task<List<AverageFlightLoad>>
GetAverageFlightLoadAsync()
        {
            List<AverageFlightLoad> loads = new List<AverageFlightLoad>();
            var response = await Client.GetAsync(url +
"Flights/AverageLoads");
            if (response.IsSuccessStatusCode)
            {
                string responseText = await
response.Content.ReadAsStringAsync();
                loads =
JsonConvert.DeserializeObject<List<AverageFlightLoad>>(responseText);
            }
            return loads;
        }

        public static Task<List<PassengersFlightsCount>>
GetPassengersFlightsCountAsync()
        {
            List<PassengersFlightsCount> passengersFlights = new
List<PassengersFlightsCount>();
            var response = await Client.GetAsync(url +
"Passengers/FlightsCounts");
            if (response.IsSuccessStatusCode)
            {
                string responseText = await
response.Content.ReadAsStringAsync();

```

```

        passengersFlights
JsonConvert.DeserializeObject<List<PassengersFlightsCount>>(responseText);
    }
    return passengersFlights;
}

public async static Task<int> RedeemBookingAsync(int id_booking)
{
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
    int priority = await GetPriorityAsync();
    var content = new StringContent("{\"id_booking\":\"" + id_booking
+ "\", \"priority\":\"" + priority + "\"", Encoding.UTF8,
"application/json");
    var response = await Client.PostAsync(url + "Bookings/Redeem",
content);

    if (response.IsSuccessStatusCode)
    {
        int price;
        string responseText = await
response.Content.ReadAsStringAsync();
        price = JsonConvert.DeserializeObject<int>(responseText);
        return price;
    }
    return -1;
}

public async static Task<bool> CancelBookingAsync(int id_booking)
{
    var token = await SecureStorage.GetAsync("jwt_token");
    Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
    int priority = await GetPriorityAsync();
    var response = await Client.DeleteAsync(url +
$"Bookings/Cancel?id_booking={id_booking}&priority={priority}");

    return response.IsSuccessStatusCode;
}

public async static Task<(int, int)> RedeemMultipleBookingAsync(int
id_schedule, string passport)
{
    var content = new StringContent("{\"id_schedule\":\"" + id_schedule
+ "\", \"passport\":\"" + passport + "\"", Encoding.UTF8, "application/json");
    var response = await Client.PostAsync(url +
"Bookings/RedeemMultiple", content);

    if (response.IsSuccessStatusCode)
    {
        int count, price;
        string responseText = await
response.Content.ReadAsStringAsync();
        (count, price) = JsonConvert.DeserializeObject<(int,
int)>(responseText);
        return (count, price);
    }
    return (0, -1);
}

public async static Task<int> CancelMultipleBookingAsync(int
id_schedule, string passport)
{

```

```

        var token = await SecureStorage.GetAsync("jwt_token");
        Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
        int priority = await GetPriorityAsync();
        var response = await Client.DeleteAsync(url +
$"Bookings/CancelMultiple?id_schedule={id_schedule}&passport={passport}&prior
ity={priority}");

        if (response.IsSuccessStatusCode)
        {
            string responseText = await
response.Content.ReadAsStringAsync();
            int count = JsonConvert.DeserializeObject<int>(responseText);
            return count;
        }
        return 0;
    }

    public async static Task<int> RefundTicketAsync(int id_ticket)
    {
        var token = await SecureStorage.GetAsync("jwt_token");
        Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
        int priority = await GetPriorityAsync();
        var response = await Client.DeleteAsync(url +
$"Tickets/Refund?id_ticket={id_ticket}&priority={priority}");

        if (response.IsSuccessStatusCode)
        {
            int price;
            string responseText = await
response.Content.ReadAsStringAsync();
            price = JsonConvert.DeserializeObject<int>(responseText);
            return price;
        }
        return -1;
    }

    public async static Task<(int, int)> RefundMultipleTicketAsync(int
id_schedule, string passport)
    {
        var token = await SecureStorage.GetAsync("jwt_token");
        Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
        var response = await Client.DeleteAsync(url +
$"Tickets/RefundMultiple?id_schedule={id_schedule}&passport={passport}");

        if (response.IsSuccessStatusCode)
        {
            int count, price;
            string responseText = await
response.Content.ReadAsStringAsync();
            (count, price) = JsonConvert.DeserializeObject<(int,
int)>(responseText);
            return (count, price);
        }
        return (0, -1);
    }

    public static async Task<int> ChangePassengerAsync(int id_passenger,
string telephone, string pasport, DateTime birth_date, string last_name,
string first_name, string middle_name)
    {
        string date_str = birth_date.Date.ToString("yyyy-MM-dd");

```

```

        var content = new StringContent("{ \"id_passenger\": \"" +
id_passenger + "\"\" +
        + ", \"telephone\": \"" + telephone + "\"\" +
        + ", \"passport\": \"" + pasport + "\"\" +
        + ", \"birth_date\": \"" + date_str + "\"\" +
        + ", \"last_name\": \"" + last_name + "\"\" +
        + ", \"first_name\": \"" + first_name + "\"\" +
        + ", \"middle_name\": \"" + middle_name + "\"}",
        Encoding.UTF8, "application/json");
        var response = await Client.PutAsync(url + "Passengers/Change",
content);

        if (response.IsSuccessStatusCode)
        {
            return 0;
        }
        return -1;
    }

    public async static Task<List<TicketsBookings>>
GetTicketsBookingsAsync()
    {
        var token = await SecureStorage.GetAsync("jwt_token");
        Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
        var response = await Client.GetAsync(url +
"Passengers/TicketsBookings");
        List<TicketsBookings> ticketsBookings = new
List<TicketsBookings>();

        if (response.IsSuccessStatusCode)
        {
            string responseText = await
response.Content.ReadAsStringAsync();
            ticketsBookings =
JsonConvert.DeserializeObject<List<TicketsBookings>>(responseText);
        }
        return ticketsBookings;
    }

    public async static Task<List<BookingsInformation>>
GetNotConfirmedBookingsAsync()
    {
        var token = await SecureStorage.GetAsync("jwt_token");
        Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
        var response = await Client.GetAsync(url +
"Bookings/NotConfirmedBookings");
        List<BookingsInformation> bookingsInformation = new
List<BookingsInformation>();

        if (response.IsSuccessStatusCode)
        {
            string responseText = await
response.Content.ReadAsStringAsync();
            bookingsInformation =
JsonConvert.DeserializeObject<List<BookingsInformation>>(responseText);
        }
        return bookingsInformation;
    }

    public async static Task<bool> ConfirmBookingAsync(int id_booking)
    {
        var token = await SecureStorage.GetAsync("jwt_token");

```

```

        Client.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);
        var content = new StringContent("{\"id_booking\":\" + id_booking +
\"}\", Encoding.UTF8, "application/json");
        var response = await Client.PutAsync(url +
"Bookings/ConfirmBooking", content);

        return response.IsSuccessStatusCode;
    }
}

```

Листинг Г.4 – Файл MainPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:d="http://xamarin.com/schemas/2014/forms/design"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d"
    x:Class="Airoport.Views.MainPage"
    Title="Вход">
    <StackLayout VerticalOptions="Center">
        <Entry Text="{Binding Login}"
            WidthRequest="350"
            Placeholder = "Логин"
            FontSize="Large"
            VerticalOptions="Center"
            HorizontalOptions="Center"/>
        <Entry Text="{Binding Password}"
            WidthRequest="350"
            Placeholder = "Пароль"
            FontSize="Large"
            IsPassword = "True"
            VerticalOptions="Start"
            HorizontalOptions="Center" />
        <Button Text="Войти"
            VerticalOptions="Start"
            HorizontalOptions="Center"
            Command="{Binding EnterCommand}" />
        <StackLayout HorizontalOptions="Center"
            Orientation="Horizontal">
            <Label Text="Нет аккаунта?"/>
            <Label Text="Зарегистрироваться">
                <Label.GestureRecognizers>
                    <TapGestureRecognizer Command="{Binding RegisterCommand}"
                />
            </Label.GestureRecognizers>
        </Label>
    </StackLayout>
</StackLayout>
</ContentPage>

```

Листинг Г.5 – Файл MainPageViewModel.cs

```

using Airoport.Services;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;

```

```

using System.Windows.Input;
using Xamarin.Forms;

namespace Airoport.ViewModels
{
    public class MainPageViewModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        public ICommand EnterCommand { get; }
        public ICommand RegisterCommand { get; }
        private Page page;
        private string login;
        private string password;
        protected void OnPropertyChanged([CallerMemberName] string
propertyName = null)
        {
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
        }

        public MainPageViewModel(Page page)
        {
            EnterCommand = new Command(async () => await Enter());
            RegisterCommand = new Command(async () => await Register());
            this.page = page;
        }

        public string Login
        {
            get => login;
            set
            {
                login = value;
                OnPropertyChanged();
            }
        }

        public string Password
        {
            get => password;
            set
            {
                password = value;
                OnPropertyChanged();
            }
        }

        private async Task Enter()
        {
            bool res = await Service.LoginAsync(login, password);
            if (res == false)
            {
                await page.DisplayAlert("Вход", "Неверный логин или пароль",
"OK");
                return;
            }
            else
            {
                Login = "";
                Password = "";
                await Shell.Current.GoToAsync($"//SellBook");
            }
        }
    }
}

```

```

        private async Task Register()
        {
            await Shell.Current.GoToAsync($"//Register");
        }
    }
}

```

Листинг Г.6 – Файл Flights.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:d="http://xamarin.com/schemas/2014/forms/design"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d"
    x:Class="Airoport.Views.Flights"
    Title="Выбор рейса">
    <ContentPage.Content>
        <StackLayout>
            <ListView ItemsSource="{Binding Flights}" SelectedItem="{Binding
Flight}" IsVisible="True" RowHeight="220">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <ViewCell>
                            <StackLayout Orientation="Vertical">
                                <Label Text="{Binding Path=id_schedule,
StringFormat='Номер рейса: {0}'}" />
                                <Label Text="{Binding
Path=name_airoport_from, StringFormat='Аэропорт вылета: {0}'}" />
                                <Label Text="{Binding Path=name_airoport_to,
StringFormat='Аэропорт прилета: {0}'}" />
                                <Label Text="{Binding Path=departue_time,
StringFormat='Время вылета: {0}'}" />
                                <Label Text="{Binding Path=arrival_time,
StringFormat='Время прилета: {0}'}" />
                                <Label Text="{Binding Path=name_airplane,
StringFormat='Самолет: {0}'}" />
                                <Label Text="{Binding Path=base_price,
StringFormat='Цена: {0}'}" />
                                <Label Text="{Binding
Path=number_available_seats, StringFormat='Количество свободных мест: {0}'}"
/>
                            </StackLayout>
                        </ViewCell>
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
            <Button Text="Перейти к оформлению билетов на выбранный рейс"
                VerticalOptions="Start"
                HorizontalOptions="Center"
                Command="{Binding IssueCommand}" />
        </StackLayout>
    </ContentPage.Content>
</ContentPage>

```

Листинг Г.7 – Файл FlightsViewModel.cs

```

using Airoport.Models;
using Airoport.Services;
using System;

```

```

using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;
using System.Web;
using Xamarin.Forms;

namespace Airopport.ViewModels
{
    [QueryProperty(nameof(City_from), "city_from")]
    [QueryProperty(nameof(City_to), "city_to")]
    [QueryProperty(nameof(Date), "date")]
    class FlightsViewModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        private Page page;
        private string city_from, city_to;
        private DateTime date;
        private ObservableCollection<Flight> flights;
        private Flight flight;
        public Command IssueCommand { get; set; }
        protected void OnPropertyChanged([CallerMemberName] string
propertyName = null)
        {
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
        }
        public string City_from
        {
            set
            {
                city_from = HttpUtility.UrlDecode(value);
            }
        }
        public string City_to
        {
            set
            {
                city_to = HttpUtility.UrlDecode(value);
            }
        }
        public string Date
        {
            set
            {
                date = DateTime.Parse(value);
            }
        }
        public ObservableCollection<Flight> Flights
        {
            get => flights;
            set
            {
                flights = value;
                OnPropertyChanged();
            }
        }
        public Flight Flight
        {
            get => flight;
            set
            {
                flight = value;
            }
        }
    }
}

```

```

        OnPropertyChanged();
    }
}

public FlightsViewModel (Page page)
{
    Flights = new ObservableCollection<Flight>();
    IssueCommand = new Command(async () => await OnIssueClicked());
    this.page = page;
    page.Appearing += OnPageAppearing;
}

private async void OnPageAppearing (object sender, EventArgs e)
{
    if (city_from != "" && city_to != "" && date != null)
    {
        int id = 0;
        int priority = await Service.GetPriorityAsync();
        if (priority == 1)
        {
            id = await Service.GetYourselfPassengerIdAsync();
        }
        var _flights = await Service.GetFlightsAsync(city_from,
city_to, date, priority, id);
        if (!(_flights is null)) {
            Flights = new ObservableCollection<Flight>(_flights);
        }
    }
}

private async Task OnIssueClicked()
{
    if (flight == null)
    {
        await page.DisplayAlert("Выбор рейса", "Не выбран рейс",
"OK");
        return;
    }
    int priority = await Service.GetPriorityAsync();
    if (priority < 0)
    {
        return;
    }
    if (priority > 1)
    {
        if (await page.DisplayAlert("Выбор рейса", "Перед
броированием билетов необходимо ввести личные данные. Вы хотите сейчас это
сделать?", "OK", "Отмена"))
        {
            await
Shell.Current.GoToAsync($"./ChangeDetailsPassenger");
        }
        return;
    }
    int id_schedule = flight.id_schedule;
    Purchase.id_schedule = id_schedule;
    Purchase.passengers = new Dictionary<int, string>();
    if (priority == 1)
    {
        Purchase.id_buyer = await
Service.GetYourselfPassengerIdAsync();
        await Shell.Current.GoToAsync($"./IssueSellBook?id_buyer=" +
Purchase.id_buyer);
    }
}

```

```

        else
        {
            Purchase.id_buyer = 0;
            await Shell.Current.GoToAsync($"../IssueSellBook");
        }
    }
}
}

```

Листинг Г.8 – Файл SellBook.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:d="http://xamarin.com/schemas/2014/forms/design"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    mc:Ignorable="d"
    x:Class="Airoport.Views.SellBook">
    <ContentPage.Content>
        <StackLayout>
            <StackLayout VerticalOptions="Start"
                HorizontalOptions="Start"
                Orientation="Horizontal">
                <Label x:Name="labelFrom"
                    Text="Откуда"
                    HorizontalOptions="Start" />
                <Picker ItemsSource="{Binding Cities}"
                    SelectedItem="{Binding FromCity}"
                    WidthRequest="350"
                    FontSize="Large"
                    HorizontalOptions="End" />
            </StackLayout>
            <StackLayout VerticalOptions="Start"
                HorizontalOptions="Start"
                Orientation="Horizontal">
                <Label x:Name="labelTo"
                    Text="Куда"
                    HorizontalOptions="Start" />
                <Picker ItemsSource="{Binding Cities}"
                    SelectedItem="{Binding ToCity}"
                    WidthRequest="350"
                    FontSize="Large"
                    HorizontalOptions="End" />
            </StackLayout>
            <StackLayout VerticalOptions="Start"
                HorizontalOptions="Start"
                Orientation="Horizontal">
                <Label x:Name="labelDate"
                    Text="Дата вылета"
                    HorizontalOptions="Start" />
                <DatePicker Date="{Binding FlightDate}"
                    HorizontalOptions="End" />
            </StackLayout>
            <Label Text="Рекомендованные направления:"
                HorizontalOptions="Start" />
            <ListView x:Name="listViewRecommendedCities"
                ItemsSource="{Binding RecommendedCities}"
                HorizontalOptions="FillAndExpand">
                <ListView.ItemTemplate>
                    <DataTemplate>
                        <TextCell Text="{Binding .}" />
                    </DataTemplate>
                </ListView.ItemTemplate>
            </ListView>
        </ContentPage.Content>
    </ContentPage>
</ContentPage>

```

```

        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
<Button Text="Вывести список рейсов"
        VerticalOptions="Start"
        HorizontalOptions="Center"
        Command="{Binding GetFlightsCommand}" />
</StackLayout>
</ContentPage.Content>
</ContentPage>

```

Листинг Г.9 – Файл SellBookViewModel.cs

```

using Airopport.Services;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;
using System.Windows.Input;
using Xamarin.Forms;

namespace Airopport.ViewModels
{
    class SellBookViewModel : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        public ICommand GetFlightsCommand { get; }
        private Page page;
        private ObservableCollection<string> cities;
        private ObservableCollection<string> recommendedCities = new
ObservableCollection<string>();
        private string fromCity;
        private string toCity;
        private DateTime flightDate;
        protected void OnPropertyChanged([CallerMemberName] string
propertyName = null)
        {
            PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
        }

        public SellBookViewModel(Page page)
        {
            GetFlightsCommand = new Command(async () => await GetFlights());
            InitializeCities();
            InitializeRecommendations();
            FlightDate = DateTime.Today;
            this.page = page;
        }

        public ObservableCollection<string> Cities
        {
            get => cities;
            set
            {
                cities = value;
                OnPropertyChanged();
            }
        }
    }
}

```

```

public ObservableCollection<string> RecommendedCities
{
    get => recommendedCities;
    set
    {
        recommendedCities = value;
        OnPropertyChanged();
    }
}

public string FromCity
{
    get => fromCity;
    set
    {
        fromCity = value;
        OnPropertyChanged();
    }
}

public string ToCity
{
    get => toCity;
    set
    {
        toCity = value;
        OnPropertyChanged();
    }
}

public DateTime FlightDate
{
    get => flightDate;
    set
    {
        flightDate = value;
        OnPropertyChanged();
    }
}

private async void InitializeCities()
{
    List<string> citiesList = await Service.GetCitiesAsync();
    Cities = new ObservableCollection<string>(citiesList);
}

private async void InitializeRecommendations()
{
    int priority = await Service.GetPriorityAsync();
    if (priority == 2)
    {
        RecommendedCities.Clear();
        return;
    }

    int passengerId = await Service.GetYourselfPassengerIdAsync();
    if (passengerId != -1)
    {
        List<string> recommendations = await
Service.GetRecommendationsAsync(passengerId, 3); // Выбираем топ-3
рекомендации
        RecommendedCities = new
ObservableCollection<string>(recommendations);
    }
}

```

```

    }

    private async Task GetFlights()
    {
        if (string.IsNullOrWhiteSpace(fromCity) ||
            string.IsNullOrWhiteSpace(toCity))
        {
            await page.DisplayAlert("Продажа/бронирование билетов", "Не  
выбрано направление авиаперелета", "OK");
            return;
        }
        await
        Shell.Current.GoToAsync($"Flights?&city_from={fromCity}&city_to={toCity}&date  
={flightDate.ToString("d")}");
    }
}

```

ПРИЛОЖЕНИЕ Д
Результаты тестирования серверной части
(рекомендуемое)



Рисунок Д.1 – Авторизация аккаунта



Рисунок Д.2 – Получение информации о кассире по его идентификатору



Рисунок Д.3 – Получение списка городов

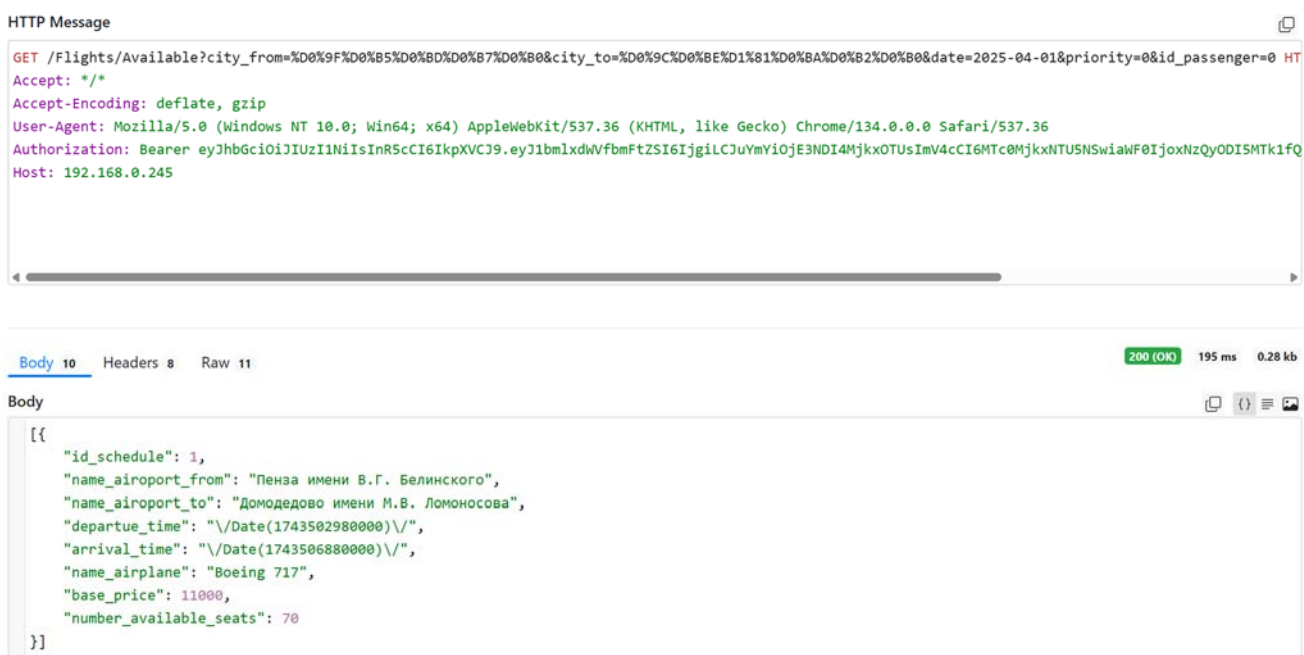


Рисунок Д.4 – Получение списка рейсов на определенное направление и дату



Рисунок Д.7 – Изменение данных пассажира



Рисунок Д.8 – Добавление пассажира



Рисунок Д.9 – Добавление брони



Рисунок Д.10 – Снятие брони по ее идентификатору



Рисунок Д.11 – Выкуп брони

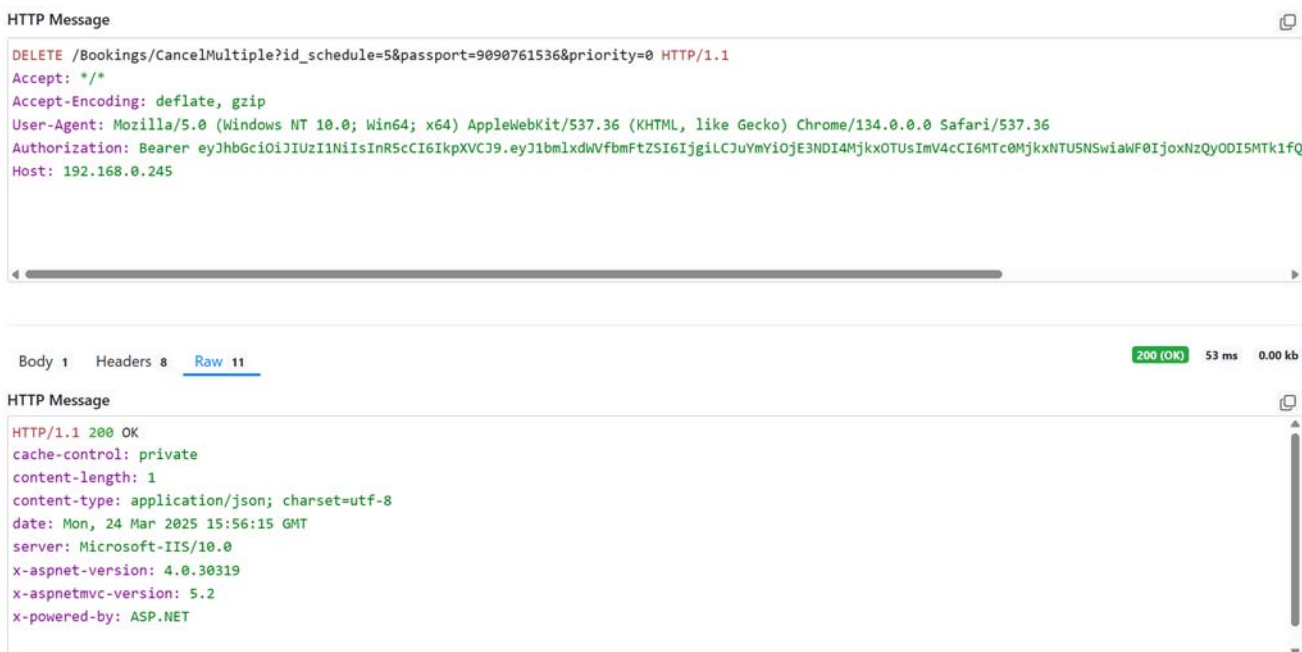


Рисунок Д.12 – Снятие множества броней по номеру рейса и номеру паспорта оформителя броней



Рисунок Д.13 – Выкуп множества брони



Рисунок Д.14 – Добавление билета



Рисунок Д.15 – Возврат билета по его идентификатору



Рисунок Д.16 – Возврат множества билетов



Рисунок Д.17 – Получение средней загруженности рейсов по дням за последнюю неделю



Рисунок Д.18 – Получение количества полетов пассажиров по месяцам за последний год

ПРИЛОЖЕНИЕ Е
Исходный код функций тестирования
(рекомендуемое)

Листинг Е.1 – Файл ControllerTests.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using ex.Controllers;
using System.Collections.Generic;
using System.Web.Mvc;
using ex.Models;

namespace Tests
{
    [TestClass]
    public class ControllerTests
    {
        public static IEnumerable<object[]> GetFlights()
        {
            yield return new object[] { true, "Пенза", "Москва",
DateTime.Parse("2023-05-03") };
            yield return new object[] { false, "Москва", "Пенза",
DateTime.Parse("2023-05-03") };
            yield return new object[] { false, "Пенза", "Москва",
DateTime.Parse("2023-05-05") };
            yield return new object[] { false, "Москва", "Пенза",
DateTime.Parse("2023-05-05") };
        }

        [DataTestMethod]
        [DynamicData(nameof(GetFlights), DynamicDataSourceType.Method)]
        public void Flights(bool isExists, string city_from, string city_to,
DateTime date)
        {
            var controller = new FlightsController();
            var result = controller.Available(city_from, city_to, date);
            if (!isExists)
            {
                Assert.IsInstanceOfType(result, typeof(HttpNotFoundResult));
                Assert.AreEqual(404,
((HttpNotFoundResult)result).StatusCode);
            }
            else
            {
                Assert.IsInstanceOfType(result, typeof(JsonResult));
                var r = ((JsonResult)result).Data;
                Assert.AreNotEqual(0, ((List<get_flights_Result>)r).Count);
            }
        }

        public static IEnumerable<object[]> GetLogins()
        {
            yield return new object[] { true, "Igorev", "Igorevich1998" };
            yield return new object[] { false, "Igorev", "Igorevich1999" };
            yield return new object[] { false, "Igore", "Igorevich1998" };
            yield return new object[] { false, "", "Igorevich1998" };
            yield return new object[] { false, "Igorev", "" };
            yield return new object[] { false, "", "" };
        }

        [DataTestMethod]
        [DynamicData(nameof(GetLogins), DynamicDataSourceType.Method)]
        public void Logins(bool isExists, string login, string password)
        {
            var controller = new CashiersController();
        }
    }
}
```

```

        var result = controller.Login(login, password);
        if (!isExists)
        {
            Assert.IsInstanceOfType(result,
typeof(HttpUnauthorizedResult));
            Assert.AreEqual(401,
((HttpUnauthorizedResult)result).StatusCode);
        }
        else
        {
            Assert.IsInstanceOfType(result, typeof(JsonResult));
            var r = ((JsonResult)result).Data;
            Assert.IsNotNull((Cashier)r);
        }
    }
}

```

ПРИЛОЖЕНИЕ Ж
Результаты тестирования клиентской части
(рекомендуемое)

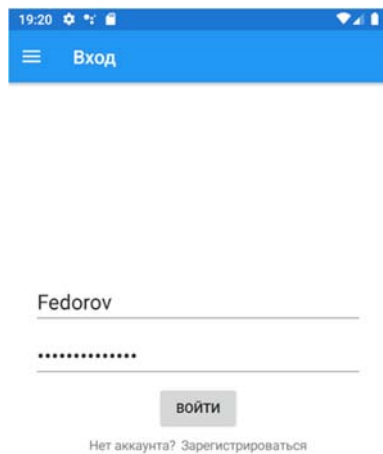


Рисунок Ж.1 – Авторизация кассира с корректными данными

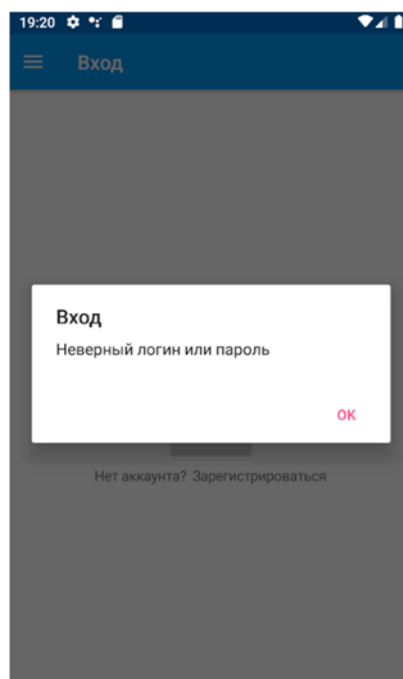


Рисунок Ж.2 – Авторизация кассира с некорректными данными

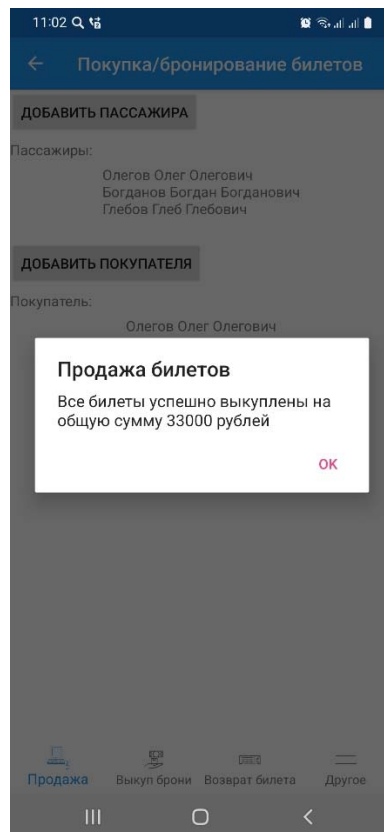


Рисунок Ж.3 – Оформление продажи билетов

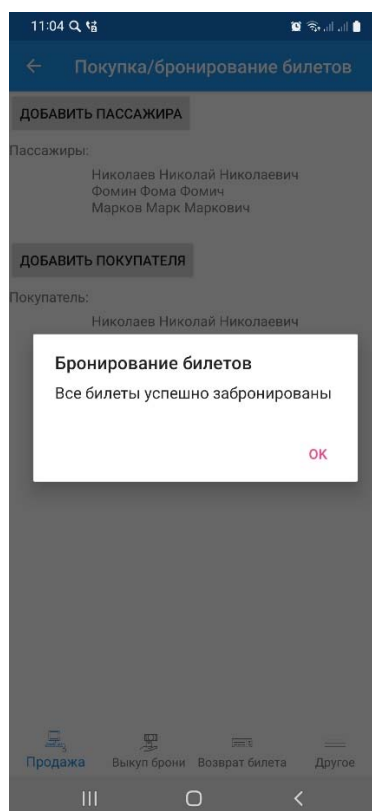


Рисунок Ж.4 – Оформление бронирования билетов

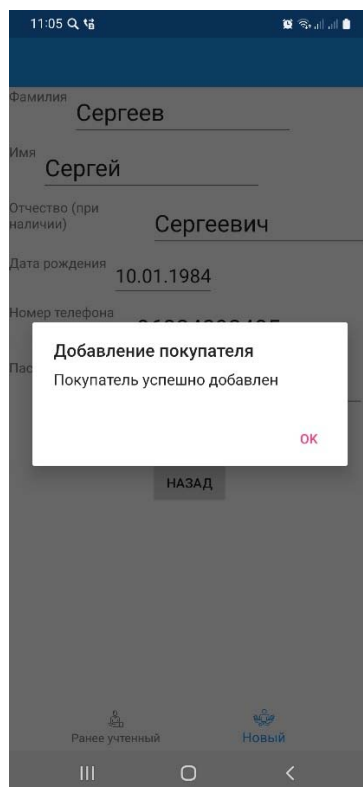


Рисунок Ж.5 – Добавление нового покупателя

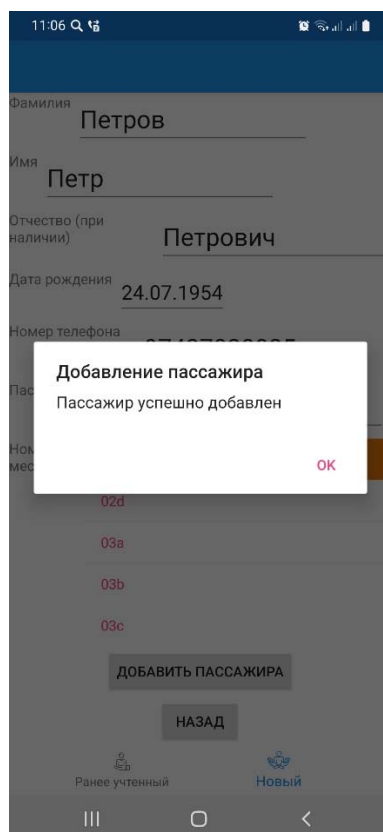


Рисунок Ж.6 – Добавление нового пассажира

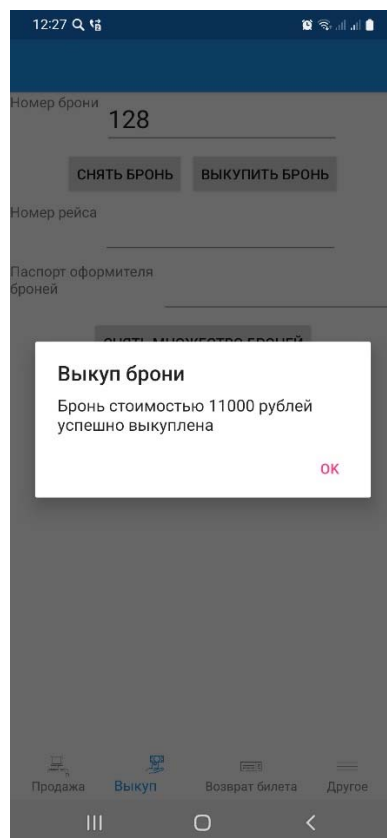


Рисунок Ж.7 – Осуществление выкупа брони

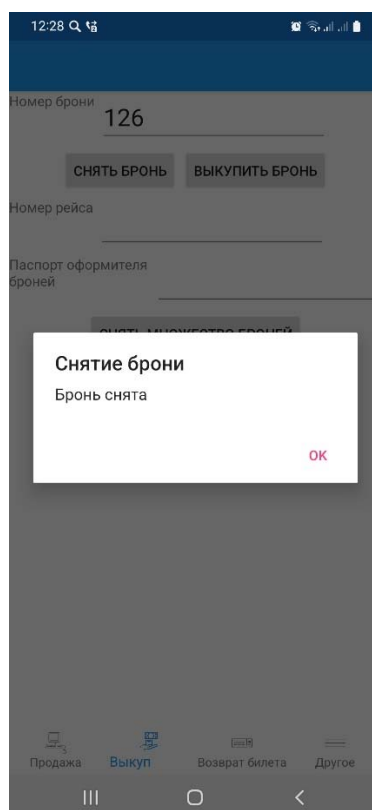


Рисунок Ж.8 – Осуществление снятия брони

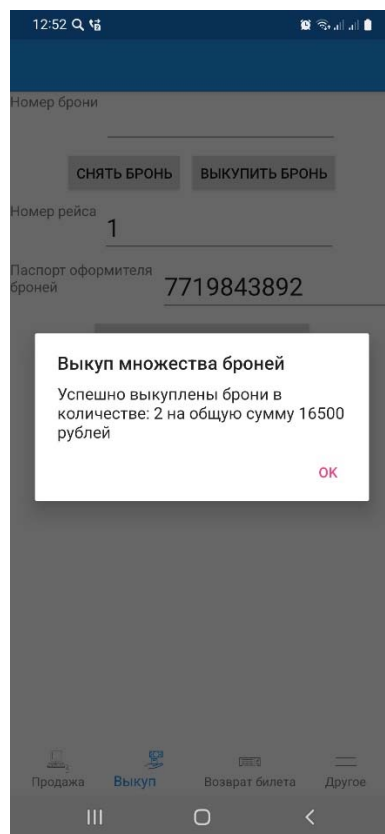


Рисунок Ж.9 – Осуществление выкупа множества броней

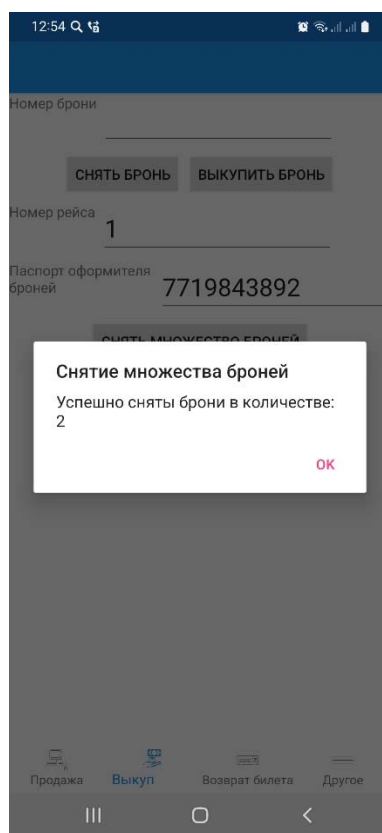


Рисунок Ж.10 – Осуществление снятия множества броней

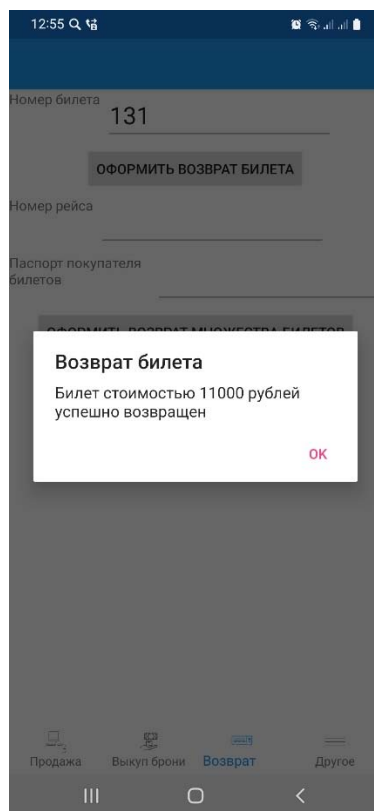


Рисунок Ж.11 – Оформление возврата билета

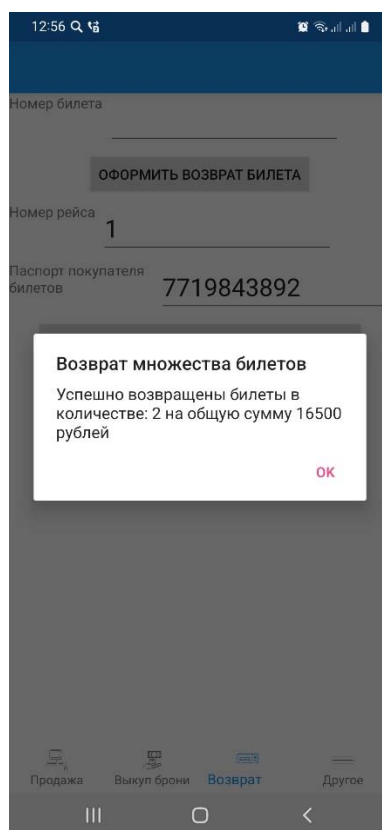


Рисунок Ж.12 – Оформление возврата множества билетов

19:21

← Аналитика

Данные

Дата	Загруженность
24.03.2025	0,245436105476673
23.03.2025	0,140909090909091
22.03.2025	0,33974358974359
21.03.2025	
20.03.2025	0,211538461538462
19.03.2025	0,116465863453815
18.03.2025	

Продажа Выкуп брони Возврат билета Другое

Рисунок Ж.13 – Получение информации о средней загруженности рейсов по дням

19:22

← Аналитика

Данные

Фамилия	Имя	Отчество	Нач. дата	Кон. дата	Кол. рейсов
Грачёва	Арина	Степановна	01.04.2024	30.04.2024	0
Грачёва	Кира	Петровна	01.05.2024	31.05.2024	0
Беляев	Роберт	Павлович	01.05.2024	31.05.2024	0
Макарова	Ника	Макаровна	01.11.2024	30.11.2024	0
Фёдорова	Кира	Степановна	01.12.2024	31.12.2024	0

Продажа Выкуп брони Возврат билета Другое

Рисунок Ж.14 – Получение информации о количестве совершенных полетов для каждого пассажира по месяцам

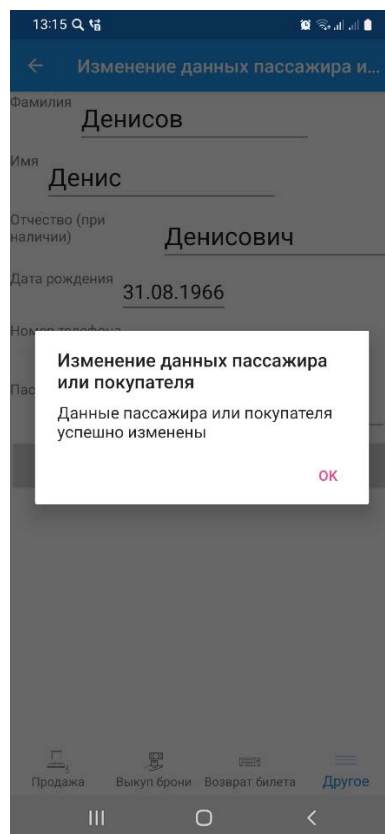


Рисунок Ж.15 – Изменение данных пассажира или покупателя

ПРИЛОЖЕНИЕ И

**Исходный код программы, реализующей гибридную рекомендательную
систему для авиаперелетов
(рекомендуемое)**

Листинг И.1 – Файл recommendation.py

```
import networkx as nx
import pyodbc
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from scipy.sparse.linalg import svds
from sklearn.preprocessing import OneHotEncoder

# Подключение к базе данных
conn = pyodbc.connect('DRIVER={SQL Server};SERVER=ILYA\\SQLEXPRESS;DATABASE=Airopport_new')
cursor = conn.cursor()

# SQL запрос для загрузки списка всех городов
query_cities = "SELECT id_city FROM City"
cities_data = pd.read_sql_query(query_cities, conn)
all_cities = cities_data['id_city'].unique()

# Коэффициент 5 для покупки, 3 для бронирования, 1 для просмотра
# Используются коэффициенты популярности городов, сезонности и предпочтений пользователей
query_purchase = """
    WITH tbl as
    (SELECT p.id_passenger, cd.id_city id_city_departure, ca.id_city
    id_city_arrival, count (*) * 5 * ss.coefficient * pp.coefficient coeff
    FROM Passenger p
    JOIN Ticket t ON p.id_passenger = t.id_passenger
    JOIN Schedule s ON t.id_schedule = s.id_schedule
    JOIN Flight f ON s.id_flight = f.id_flight
    JOIN Route r ON f.id_route = r.id_route
    JOIN Airopport aa ON r.id_airopport_arrival = aa.id_airopport
    JOIN City ca ON aa.id_city = ca.id_city
    JOIN Airopport ad ON r.id_airopport_departure =
    ad.id_airopport
    JOIN City cd ON ad.id_city = cd.id_city
    JOIN Seasonality ss ON Month(getdate()) = ss.month_number AND ca.id_city
    = ss.id_city_arrival AND cd.id_city = ss.id_city_departure
    JOIN Popularity pp ON ca.id_city = pp.id_city_arrival AND cd.id_city =
    pp.id_city_departure
    GROUP BY p.id_passenger, cd.id_city, ca.id_city, ss.coefficient,
    pp.coefficient),
    tbl2 as
    (SELECT p.id_passenger, cd.id_city id_city_departure, ca.id_city
    id_city_arrival, count (*) * 3 * ss.coefficient * pp.coefficient coeff
    FROM Passenger p
    JOIN Booking b ON p.id_passenger = b.id_passenger
    JOIN Schedule s ON b.id_schedule = s.id_schedule
    JOIN Flight f ON s.id_flight = f.id_flight
    JOIN Route r ON f.id_route = r.id_route
    JOIN Airopport aa ON r.id_airopport_arrival = aa.id_airopport
    JOIN City ca ON aa.id_city = ca.id_city
    JOIN Airopport ad ON r.id_airopport_departure =
    ad.id_airopport
    JOIN City cd ON ad.id_city = cd.id_city
    JOIN Seasonality ss ON Month(getdate()) = ss.month_number AND ca.id_city
    = ss.id_city_arrival AND cd.id_city = ss.id_city_departure
    JOIN Popularity pp ON ca.id_city = pp.id_city_arrival AND cd.id_city =
    pp.id_city_departure
```

```

GROUP BY p.id_passenger, cd.id_city, ca.id_city, ss.coefficient,
pp.coefficient),
tbl3 as
(SELECT p.id_passenger, cd.id_city id_city_departure, ca.id_city
id_city_arrival, count (*) * 1 * ss.coefficient * pp.coefficient coeff
FROM Passenger p
JOIN Viewing v ON p.id_passenger = v.id_passenger
JOIN City ca ON v.id_city_to = ca.id_city
JOIN City cd ON v.id_city_from = cd.id_city
JOIN Seasonality ss ON Month(getdate()) = ss.month_number AND ca.id_city
= ss.id_city_arrival AND cd.id_city = ss.id_city_departure
JOIN Popularity pp ON ca.id_city = pp.id_city_arrival AND cd.id_city =
pp.id_city_departure
GROUP BY p.id_passenger, cd.id_city, ca.id_city, ss.coefficient,
pp.coefficient)
SELECT ISNULL (t.id_passenger, ISNULL (t2.id_passenger, t3.id_passenger))
id_passenger,
ISNULL (t.id_city_departure, ISNULL
(t2.id_city_departure, t3.id_city_departure)) id_city_departure,
ISNULL (t.id_city_arrival, ISNULL
(t2.id_city_arrival, t3.id_city_arrival)) id_city_arrival,
ISNULL (t.coeff, 0) + ISNULL (t2.coeff, 0) +
ISNULL (t3.coeff, 0) coeff
FROM tbl t
RIGHT JOIN tbl2 t2 ON t.id_passenger =
t2.id_passenger AND t.id_city_departure = t2.id_city_departure AND
t.id_city_arrival = t2.id_city_arrival
RIGHT JOIN tbl3 t3 ON t.id_passenger =
t3.id_passenger AND t.id_city_departure = t3.id_city_departure AND
t.id_city_arrival = t3.id_city_arrival
UNION
SELECT ISNULL (t.id_passenger, ISNULL (t2.id_passenger, t3.id_passenger))
id_passenger,
ISNULL (t.id_city_departure, ISNULL
(t2.id_city_departure, t3.id_city_departure)) id_city_departure,
ISNULL (t.id_city_arrival, ISNULL
(t2.id_city_arrival, t3.id_city_arrival)) id_city_arrival,
ISNULL (t.coeff, 0) + ISNULL (t2.coeff, 0) +
ISNULL (t3.coeff, 0) coeff
FROM tbl t
LEFT JOIN tbl2 t2 ON t.id_passenger =
t2.id_passenger AND t.id_city_departure = t2.id_city_departure AND
t.id_city_arrival = t2.id_city_arrival
LEFT JOIN tbl3 t3 ON t.id_passenger =
t3.id_passenger AND t.id_city_departure = t3.id_city_departure AND
t.id_city_arrival = t3.id_city_arrival
"""

# SQL запрос для загрузки данных о пассажирах с возрастом, полом и интересами
query_passengers = """
SELECT id_passenger,
dbo.fullAge (birth_date, getdate()) age,
i.beach_holiday,
i.skiing,
i.historical_places
FROM Passenger p JOIN Interests i ON p.id_interests = i.id_interests
"""

passengers_data = pd.read_sql_query(query_passengers, conn)
purchases_data = pd.read_sql_query(query_purchase, conn)

# Объединение данных о пассажирах и покупках/бронированиях
data = pd.merge(passengers_data, purchases_data, on='id_passenger')

```

```

# Заполнение пропущенных значений в данных
data['coeff'] = data['coeff'].fillna(0)

# Преобразование категориальных признаков в One-Hot Encoding
encoder = OneHotEncoder(drop='first', sparse=False)
cities_encoded = encoder.fit_transform(data[['id_city_departure',
'id_city_arrival']])
cities_encoded_df = pd.DataFrame(cities_encoded,
columns=encoder.get_feature_names_out(['id_city_departure',
'id_city_arrival']))

# Подготовка данных для обучения алгоритма классификации
features = pd.concat([data[['age', 'beach_holiday', 'skiing',
'historical_places']], cities_encoded_df], axis=1)
target = data['coeff']

# Создание и обучение модели классификации
RandomForestRegressor = RandomForestRegressor()
RandomForestRegressor.fit(features, target)

# Создание матрицы взаимодействий пассажир-город
passenger_city_matrix = data.pivot_table(index='id_passenger',
columns='id_city_arrival', values='coeff', fill_value=0)

# Вычисление сходства между пассажирами
passenger_similarity_matrix = 1 -
passenger_city_matrix.T.corr(method='pearson')

# Вычисление сингулярного разложения для коллаборативной фильтрации
U, sigma, Vt = svds(passenger_city_matrix.to_numpy(), k=10)
sigma = np.diag(sigma)
passenger_factors = U @ sigma
city_factors = Vt.T

# Функция для прогнозирования предпочтений пассажира к городам с
использованием SVD
def predict_city_preferences_svd(passenger_id):
    passenger_idx = passenger_city_matrix.index.get_loc(passenger_id)
    passenger_vector = passenger_factors[passenger_idx, :]
    scores = passenger_vector @ city_factors.T
    return pd.Series(scores, index=passenger_city_matrix.columns)

# Очистка таблицы Recommendation перед добавлением новых данных
cursor.execute("DELETE FROM Recommendation")
conn.commit()

insert_query = """
    INSERT INTO Recommendation (id_passenger, id_city, coeff)
    VALUES (?, ?, ?)
    """

# Преобразуем данные в стандартные типы Python
def to_python(value):
    if isinstance(value, (np.integer, np.int_)):
        return int(value)
    elif isinstance(value, (np.floating, np.float_)):
        return float(value)
    else:
        return value

city_feature_names = ['id_city_departure', 'id_city_arrival']
X_train = []
y_train = []
graphs = {}

```

```

for id_passenger in data['id_passenger'].unique():
    # Создание графа
    G = nx.DiGraph()

    # Добавление вершин
    G.add_nodes_from(all_cities)

    passenger_data = data[(data['id_passenger'] == id_passenger)].iloc[[0]]

    # Добавление ребер с весами, учитывая предпочтения пользователя
    for city_departure in all_cities:
        for city_arrival in all_cities:
            if city_departure == city_arrival:
                continue
            source = city_departure
            target = city_arrival
            row = data[(data['id_passenger'] == id_passenger) &
                (data['id_city_departure'] == city_departure) & (data['id_city_arrival'] ==
                city_arrival)]

            encoded_input = pd.DataFrame([[city_departure, city_arrival]],
                columns=city_feature_names)
            encoded_input = encoder.transform(encoded_input)

            input_data = pd.DataFrame([[passenger_data.iloc[0]['age'],
                passenger_data.iloc[0]['beach_holiday'], passenger_data.iloc[0]['skiing'],
                passenger_data.iloc[0]['historical_places']],
                columns=['age', 'beach_holiday', 'skiing', 'historical_places'])
            input_data = pd.concat([input_data, pd.DataFrame(encoded_input,
                columns=encoder.get_feature_names_out(['id_city_departure',
                'id_city_arrival']))], axis=1)

            coeff = RandomForestRegressor.predict(input_data)[0]
            G.add_edge(source, target, weight=coeff)
            graphs[id_passenger] = G

    # Вычисление PageRank с учетом случайного блуждания
    pagerank = nx.pagerank(G, alpha=0.85)

    # Предсказания на основе SVD
    svd_predictions =
predict_city_preferences_svd(id_passenger).sort_values(ascending=False)

    # Определение значений пассажира на основе матрицы сходства
    passenger_favorites =
passenger_city_matrix.loc[id_passenger].sort_values(ascending=False).index.to
list()
    passenger_similarities =
passenger_similarity_matrix[id_passenger].sort_values(ascending=False).fillna
(0)

    # Заполнение данных для обучения линейной регрессии
    for city in passenger_favorites:
        if data[(data['id_passenger'] == id_passenger) &
            (data['id_city_arrival'] == city)].shape[0] == 0:
            continue
        pagerank_score = pagerank[city]
        similarity_score = passenger_similarities.get(city, 0)
        svd_score = svd_predictions.get(city, 0)
        X_train.append([pagerank_score, similarity_score, svd_score])
        y_train.append(data[(data['id_passenger'] == id_passenger) &
            (data['id_city_arrival'] == city)]['coeff'].values[0])

```

```

# Обучение линейной регрессии
linearRegressor = LinearRegression()
linearRegressor.fit(X_train, y_train)

for id_passenger in data['id_passenger'].unique():
    G = graphs[id_passenger]
    pagerank = nx.pagerank(G, alpha=0.85)

    # Предсказания на основе SVD
    svd_predictions =
predict_city_preferences_svd(id_passenger).sort_values(ascending=False)

    # Определение значений пассажира на основе матрицы сходства
    passenger_favorites =
passenger_city_matrix.loc[id_passenger].sort_values(ascending=False).index.to
list()
    passenger_similarities =
passenger_similarity_matrix[id_passenger].sort_values(ascending=False).fillna
(0)

    hybrid_recommendations = []
    for city in passenger_favorites:
        pagerank_score = pagerank[city]
        similarity_score = passenger_similarities.get(city, 0)
        svd_score = svd_predictions.get(city, 0)
        predicted_coeff = linearRegressor.predict([[pagerank_score,
similarity_score, svd_score]])[0]
        hybrid_recommendations.append((city, predicted_coeff))
    hybrid_recommendations = sorted(hybrid_recommendations, key=lambda x:
x[1], reverse=True)

    # Добавление рекомендаций в таблицу Recommendation
    for city, predicted_coeff in hybrid_recommendations:
        cursor.execute(
            insert_query,
            (
                to_python(id_passenger),
                to_python(city),
                to_python(predicted_coeff)
            )
        )
    conn.commit()

    print(f"Обработан пассажир {id_passenger}")

# Закрытие соединения с базой данных
cursor.close()
conn.close()

```

ПРИЛОЖЕНИЕ К
Дипломы победителя международных конкурсов научных работ
(рекомендуемое)



Рисунок К.1 – Диплом победителя Международного научно-исследовательского конкурса «Достижения вузовской науки 2023»



Рисунок К.2 – Диплом I степени Международного научно-исследовательского конкурса «Наш выбор – наука!»

ПРИЛОЖЕНИЕ Л

Свидетельство о государственной регистрации программы для ЭВМ (рекомендуемое)

РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025617775

**Программа персонализированных рекомендаций
городов для авиапассажиров**

Правообладатель: **Староверов Илья Михайлович (RU)**

Авторы: **Староверов Илья Михайлович (RU), Князев Виктор
Николаевич (RU)**



Заявка № 2025611693

Дата поступления **01 февраля 2025 г.**

Дата государственной регистрации

в Реестре программ для ЭВМ **28 марта 2025 г.**

*Руководитель Федеральной службы
по интеллектуальной собственности*

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат: 0692e703063000b542401670b0a2026
Владелец: **Зубов Юрий Сергеевич**
Действителен с 01.01.2024 по 03.10.2025

Ю.С. Зубов

ПРИЛОЖЕНИЕ М

Презентация (рекомендуемое)

Тема магистерской диссертации
Исследование и разработка автоматизированных систем
обслуживания авиапассажиров с применением элементов
искусственного интеллекта

Направление подготовки: 09.04.01 – «Информатика и
вычислительная техника»

Студент: группа 23ВВИм1

Староверов Илья Михайлович

Научный руководитель: к.т.н., доцент

Князев Виктор Николаевич

Пенза 2025

Анализ предметной области

Объект и предмет исследования

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

Цель и задачи исследования

Задачи исследования:

- 1) проведение критического анализа предметной области,
- 2) разработка расширенной классификации средств обслуживания авиапассажиров путем внедрения дополнительных классификационных признаков,
- 3) создание онтологической модели предметной области, отличающейся от существующих расширенным набором сущностей и классов предметной области,
- 4) создание дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования реализованной автоматизированной системы обслуживания авиапассажиров,
- 5) разработка сетевого графика проекта по созданию программной автоматизированной системы обслуживания авиапассажиров для последующей оценки сроков и бюджета данного проекта,
- 6) разработка рекомендательной системы в автоматизированной системе обслуживания авиапассажиров, реализация и тестирование разработанной системы,
- 7) регистрация программной реализации разработанной рекомендательной системы и подготовка к внедрению разработанной и реализованной системы обслуживания авиапассажиров.

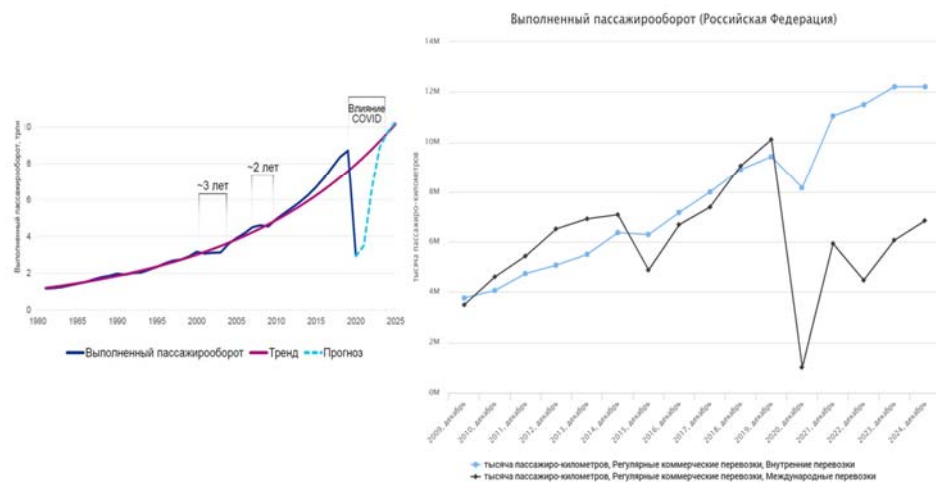
3

Научная новизна исследования

- 1) Предложен подход к разработке оригинальной гибридной рекомендательной системы, который отличается от традиционных подходов использованием сразу нескольких методов, за счет чего достигается более высокая точность и лучшая персонализация рекомендаций.
- 2) Предложен набор дискретно-событийных имитационных моделей для исследования и улучшения параметров функционирования автоматизированной системы обслуживания авиапассажиров. Эти модели, в отличие от известных аналогичных моделей, имеют комплексный характер и позволяют учитывать как технические характеристики, так и человеческий фактор, связанный с деятельностью кассиров.
- 3) Предложена расширенная онтологическая модель предметной области для систем обслуживания авиапассажиров, отличающаяся от известных расширенным набором сущностей и классов предметной области, с целью формализации и систематизации информации для эффективного проведения проектирования объекта исследования.
- 4) Предложена модифицированная классификация средств обслуживания авиапассажиров, отличающаяся от известных тем, что предлагается более полный расширенный перечень классификационных признаков и средств.

4

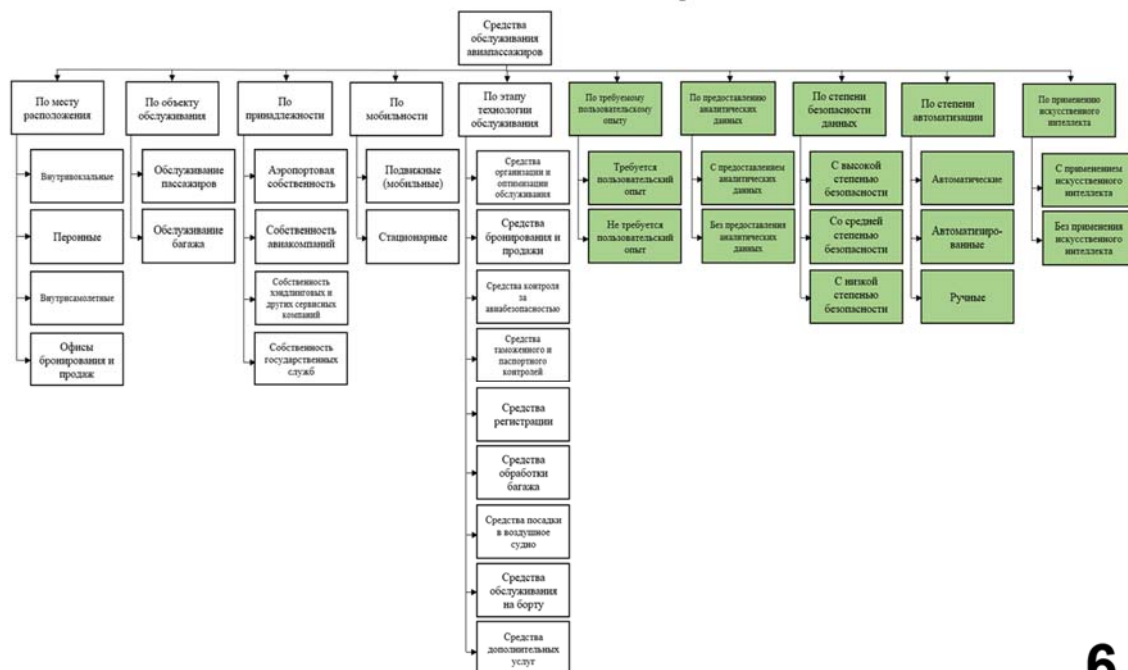
Актуальность исследования



5

Анализ предметной области

Классификация средств обслуживания авиапассажиров



6

Обзор аналогов

Система обслуживания авиапассажиров	Бронирование билетов	Покупка билетов	Автозаполнение данных	Платный функционал	Функция возврата билета
Аэрофлот	Есть	Есть	Нет	Нет	Есть
Aviasales	Нет	Нет	Нет	Есть	Есть
Яндекс.авиабилеты	Нет	Нет	Нет	Нет	Нет
Разрабатываемое приложение	Есть	Есть	Есть	Нет	Есть

7

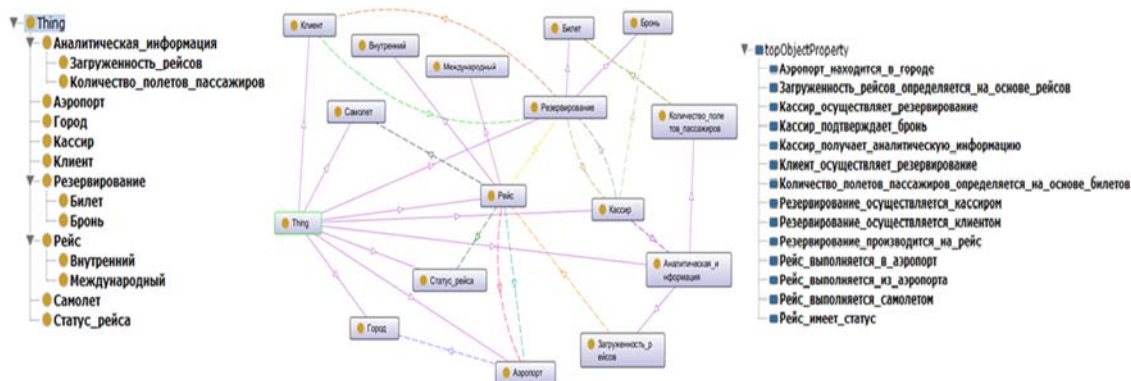
Анализ предметной области

Сетевое планирование и управление

Даты			
Начало:	Пн 11.12.23	Окончание:	Пт 02.08.24
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	156 дней	Оставшаяся:	156 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	156 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	2 479,2 ч	Оставшиеся:	2 479,2 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	2 479,2 ч	Процент завершения:	0%
Затраты			
Текущий план:	903 280,00 Р	Оставшиеся:	903 280,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	903 280,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи:	31	Трудовые ресурсы:	3
Выполняющиеся задачи:	0	Трудовые ресурсы с прев. дост:	3
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	31	Итого ресурсов:	8

8

Онтологическое моделирование



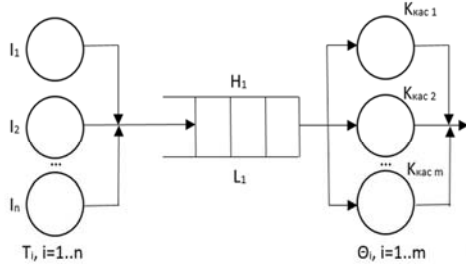
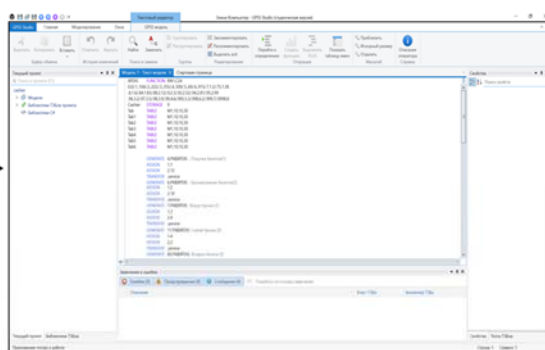
9

Разработка

Имитационное моделирование

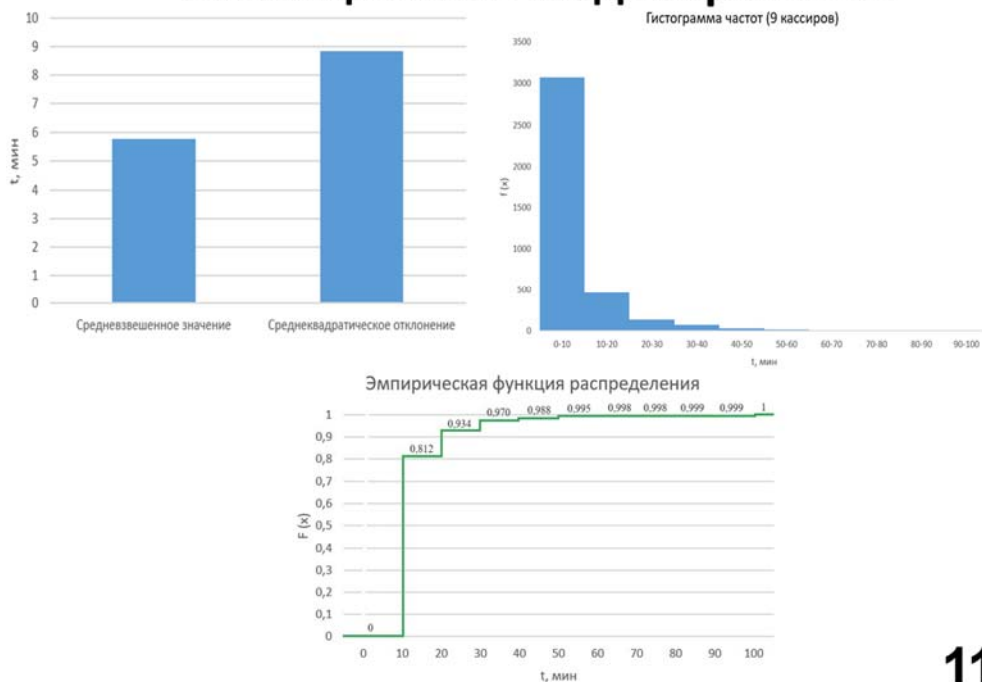
Q-схема модели

Моделирование в среде GPSS Studio



10

Имитационное моделирование



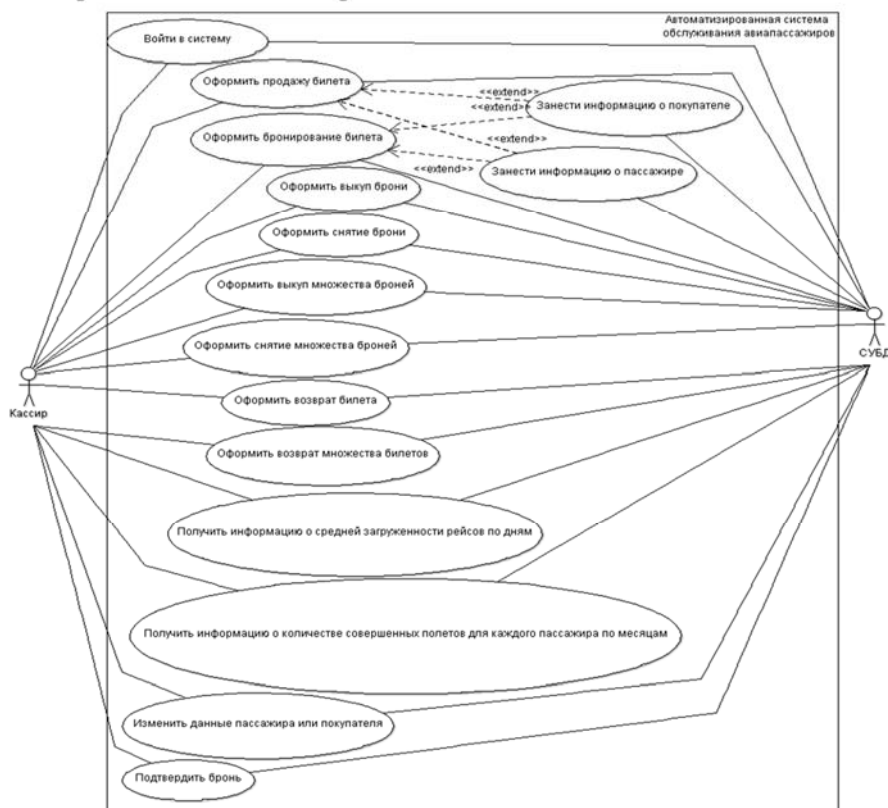
11
Разработка

Имитационное моделирование

Показатель	Расчетное значение	Модельное значение	Расхождение
Коэффициент загрузки системы	0,655	0,654	0,2%
Среднее число свободных кассиров	3,107	3,114	0,2%
Среднее время пребывания заявки в системе	5,513	5,791	5%

12

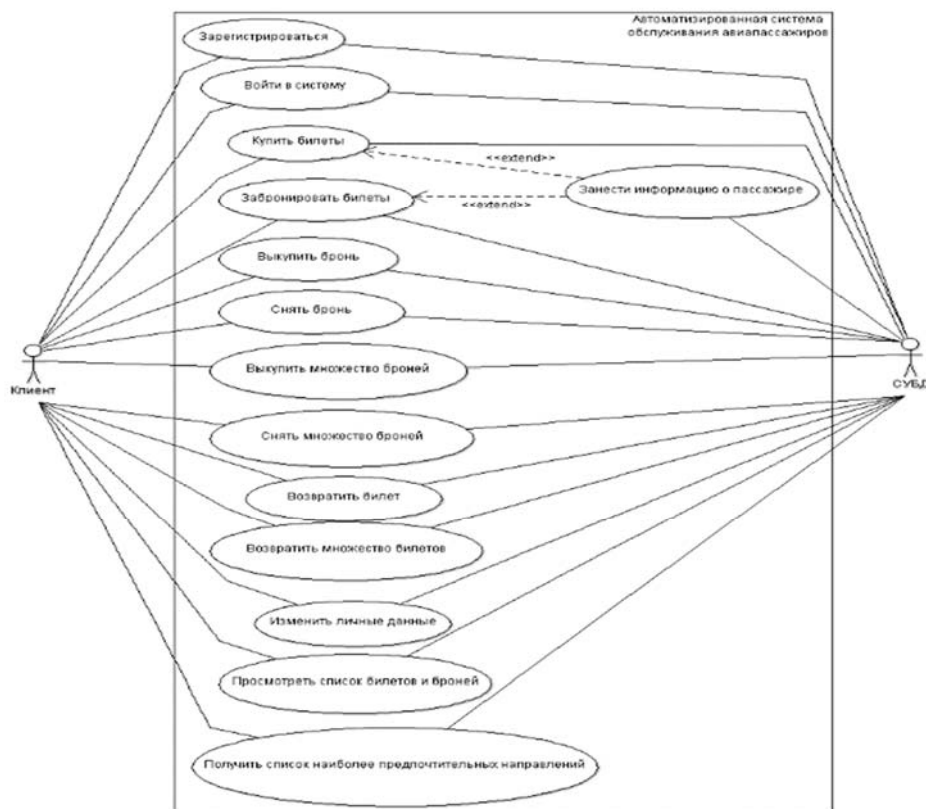
Диаграмма вариантов использования



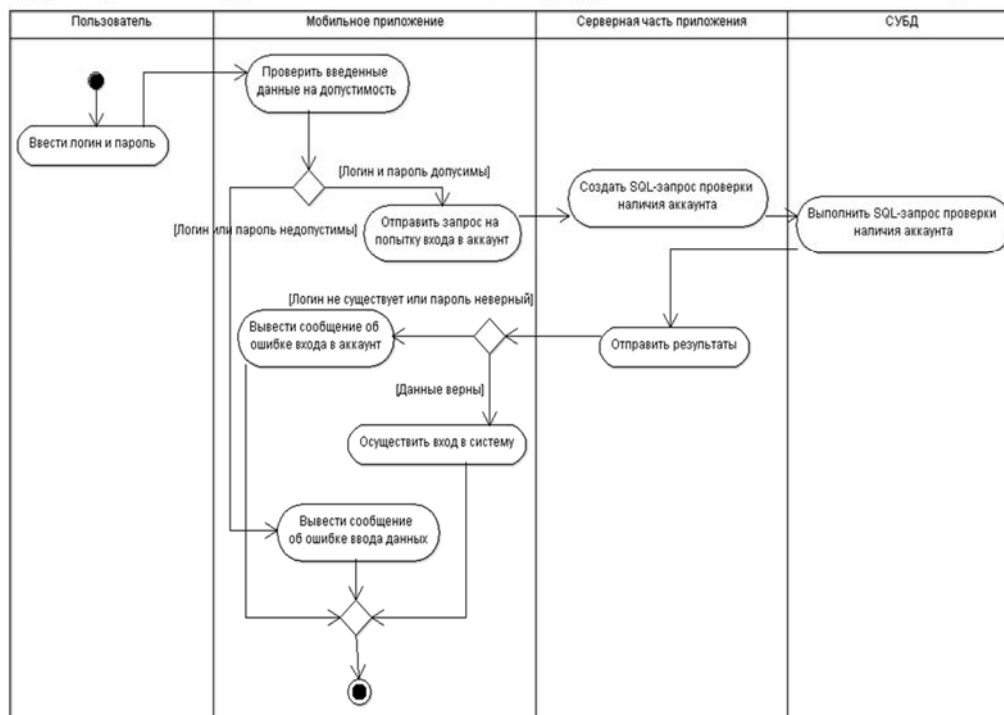
13

Разработка

Диаграмма вариантов использования



14



Разработка

Используемые технологии и средства разработки

Инструментальные средства:

- Visual Studio 2019;
- Spyder;
- Microsoft Project;
- GPSS Studio;
- Protégé.

- GPSS Studio;

- C#;
- Python;
- UML;
- OWL.

- Protégé.



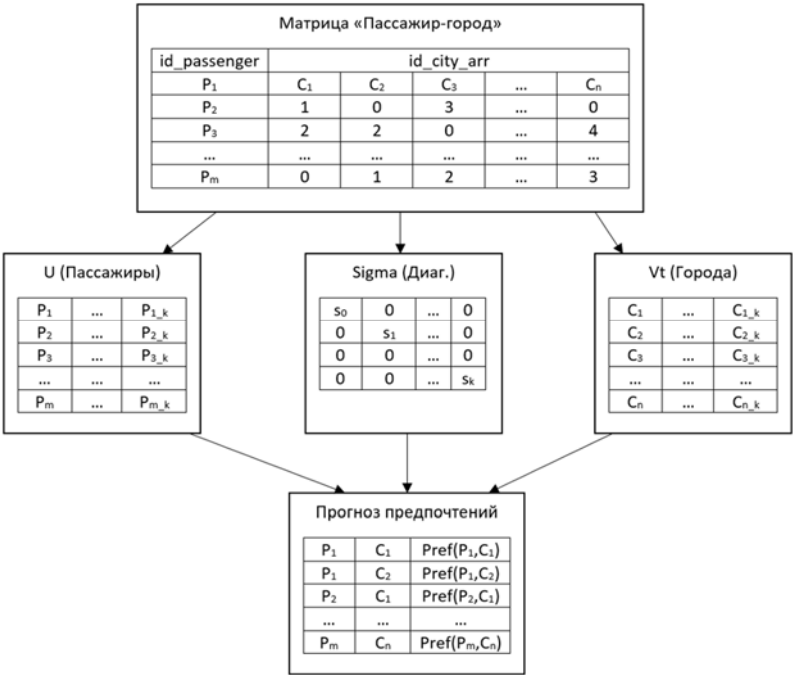
16

Гибридная рекомендательная система



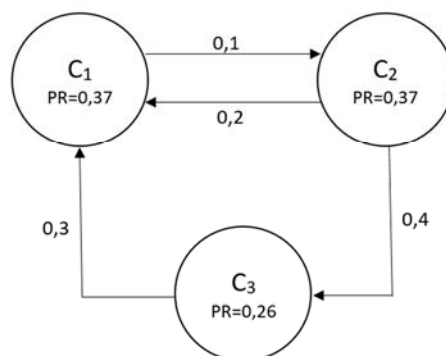
17
Разработка

Гибридная рекомендательная система



18

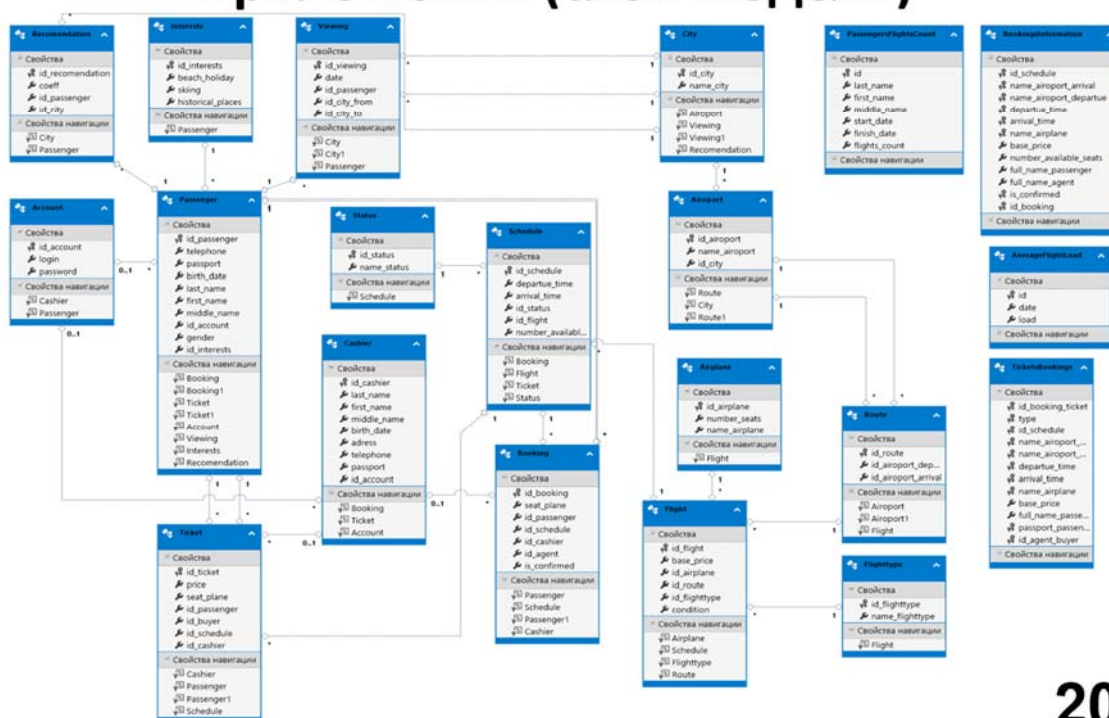
Гибридная рекомендательная система



19

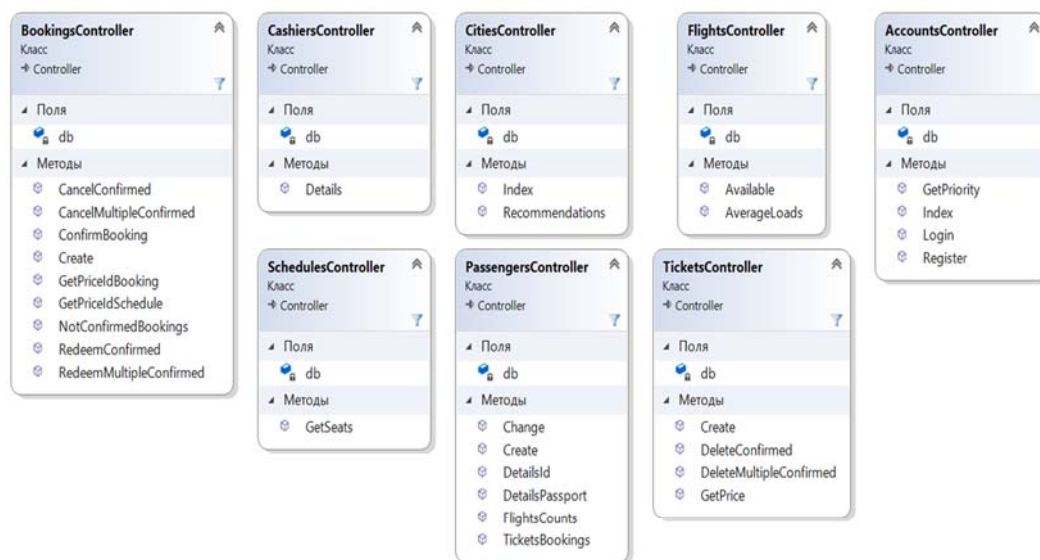
Реализация

Реализация серверной части приложения (слой модели)



20

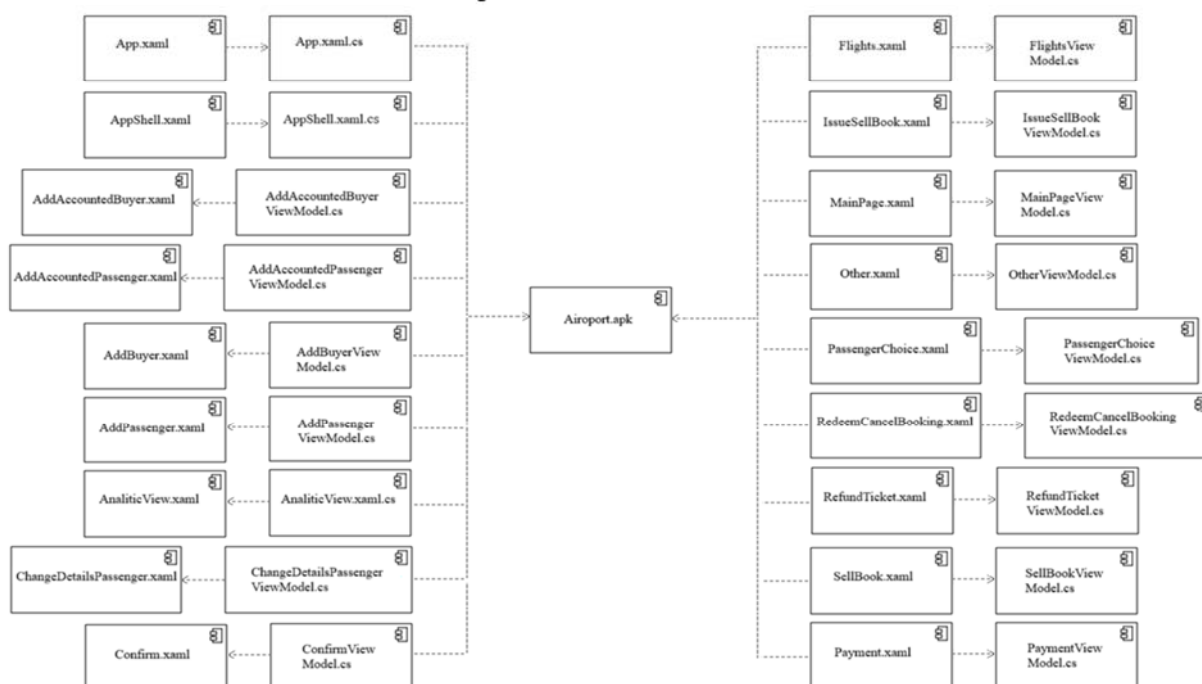
Реализация серверной части приложения (слой контроллеров)



21

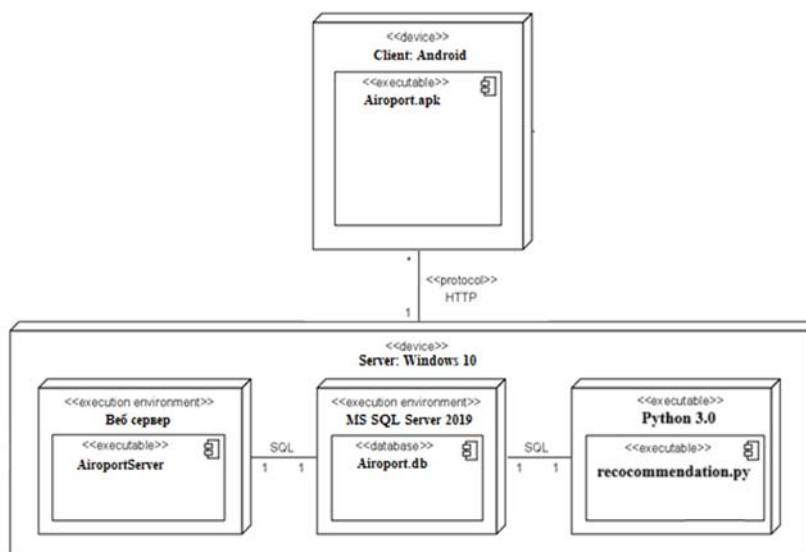
Реализация

Реализация клиентской части приложения



22

Диаграмма развертывания



23

Реализация

Тестирование гибридной рекомендательной системы

Откуда

Куда

Дата вылета 21.10.2024

Рекомендованные направления:

- Ереван
- Сочи
- Пенза

Откуда

Куда

Дата вылета 21.10.2024

Рекомендованные направления:

- Казань
- Афины
- Москва

ВЫВЕСТИ СПИСОК РЕЙСОВ

ВЫВЕСТИ СПИСОК РЕЙСОВ

24

Результаты исследования

1. Проведен расширенный и углубленный критический анализ предметной области по тематике магистерской диссертации.
2. Разработана, реализована и протестирована автоматизированная система обслуживания авиапассажиров с применением современных и эффективных информационных и программных технологий, включая рекомендательные механизмы, а также методологии онтологического, имитационного и визуального моделирования. Проведена работа по подготовке к внедрению результатов работы.
3. Предложен подход к разработке гибридной рекомендательной системы, который отличается от известных подходов к разработке рекомендательных систем тем, что сочетает коллаборативную фильтрацию, графовые модели и методы машинного обучения. Предложенный подход позволяет повысить точность прогнозов и релевантность предложений.
4. Разработан сетевой график IT-проекта по созданию системы обслуживания авиапассажиров и определен срок выполнения и бюджет IT-проекта.

25

Результаты исследования

5. Предложены две оригинальные дискретно-событийные имитационные модели для исследования и улучшения параметров функционирования системы обслуживания авиапассажиров.
6. Предложена расширенная онтологическая модель предметной области системы обслуживания авиапассажиров с целью формализации и систематизации информации для эффективного проведения проектирования объекта исследования.
7. Предложена модифицированная классификация средств обслуживания авиапассажиров.
8. По теме магистерской диссертации получены следующие документы:
 - a) диплом победителя Международного научно-исследовательского конкурса «Достижения вузовской науки 2023»,
 - b) диплом I степени Международного научно-исследовательского конкурса «Наш выбор – наука!»,
 - c) Свидетельство о государственной регистрации программы для ЭВМ в Федеральном институте промышленной собственности РФ.
9. По теме магистерской диссертации сделано 12 публикаций, включая две статьи в журнале, входящем в **перечень ВАК**, и проведена апробация на восьми конференциях различного уровня.

26

Апробация работы

1. Всероссийская научно-практическая конференция обучающихся и научно-педагогических работников «Актуальные проблемы науки и образования». – Пенза, октябрь 2023.
2. XVIII международная научно-практическая конференция «Современные тенденции развития науки и мирового сообщества в эпоху цифровизации». – Москва, октябрь 2023.
3. 9-я Всероссийская национальная научно-практическая конференция «Проблемы развития современного общества». – Пятигорск, январь 2024.
4. VII Всероссийская научно-практическая конференция «Современные тенденции развития фундаментальных и прикладных наук». – Брянск, январь 2024.
5. XI Всероссийская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза, март 2024.
6. Международная научно-практическая школа-конференция «Актуальные проблемы науки и образования». – Пенза, октябрь 2024.
7. XXI Международная научно-техническая конференция «Новые информационные технологии и системы». – Пенза, ноябрь 2024.
8. XII Всероссийская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы». – Пенза, март 2025.

27

Публикации по теме работы

1. Староверов И.М. Разработка программных средств поддержки процессов резервирования и продажи авиабилетов // Достижения вузовской науки 2023: сборник статей международного научно-исследовательского конкурса. – 2023. – С. 35 – 41.
2. Староверов И.М. Вопросы разработки автоматизированных систем обслуживания авиапассажиров // Сборник материалов XVIII международной научно-практической конференции «Современные тенденции развития науки и мирового сообщества в эпоху цифровизации». – 2023. – С. 69 – 74.
3. Староверов И.М. Проектирование автоматизированных систем обслуживания авиапассажиров с применением современных информационных технологий // Наш выбор – наука! : Сборник статей IV Международного научно-исследовательского конкурса. – 2024. – С. 20 – 28.
4. Староверов И.М. Разработка онтологической модели для автоматизированной системы обслуживания авиапассажиров // Проблемы развития современного общества: Сборник научных статей 9-й Всероссийской национальной научно-практической конференции. – 2024. – Т. 3. – С. 125 – 129.

28

Публикации по теме работы

5. Староверов И.М., Князев В.Н. Современные подходы к классификации средств обслуживания авиапассажиров // Современные тенденции развития фундаментальных и прикладных наук: Материалы VII Всероссийской научно-практической конференции. – 2024. – С. 370 – 374.
6. Староверов И.М., Князев В.Н. Вопросы разработки онтологической модели для системы автоматизированного обслуживания пассажиров воздушного транспорта // Информационные технологии в науке и образовании. Проблемы и перспективы: Сборник материалов XI Всероссийской научно-практической конференции. – 2024. – С. 167 – 169.
7. Староверов И.М., Князев В.Н. Вопросы разработки рекомендательной системы для автоматизированной системы обслуживания авиапассажиров // Информационные технологии в науке и образовании. Проблемы и перспективы: Сборник материалов XI Всероссийской научно-практической конференции. – 2024. – С. 169 – 172.
8. Староверов И.М., Князев В.Н. Комплексный подход к созданию рекомендательной системы для авиаперелетов // Вестник Пензенского государственного университета. – 2025. – № 1. (В печати).

29

Публикации по теме работы

9. Староверов И.М., Князев В.Н. Имитационное моделирование автоматизированных систем обслуживания авиапассажиров // Международный научно-исследовательский журнал. – 2025. – №3 (153). **(Перечень ВАК)**.
10. Староверов И.М., Князев В.Н. Программа персонализированных рекомендаций городов для авиапассажиров // Свидетельство о государственной регистрации программы для ЭВМ № 2025617775. Заявка № 2025611693, дата поступления 01.02.2025 г., дата государственной регистрации в Реестре программ для ЭВМ 28.03.2025 г. – М., Федеральный институт промышленной собственности, 2025.
11. Староверов И.М., Князев В.Н. Систематизация и классификация средств обслуживания авиапассажиров в контексте современных технологий // Информационные технологии в науке и образовании. Проблемы и перспективы: Сборник материалов XII Всероссийской научно-практической конференции. – 2025. (В печати).
12. Староверов И.М., Князев В.Н. Разработка гибридной рекомендательной системы для авиапассажиров // Международный научно-исследовательский журнал. – 2025. – №6 (156). **(Перечень ВАК)**. (В печати).

30

Участие в Международных конкурсах научных работ



31

Свидетельство о государственной регистрации программы для ЭВМ



32