

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

Программные средства автоматизации
транспортной логистики грузоперевозок

Студент _____ Акмашев Сергей Витальевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Балашова И.Ю.
(подпись, дата) (фамилия, инициалы)

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

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

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

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

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

Пенза, 2024

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов «25»
декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Акмашев Сергей Витальевич гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства автоматизации транспортной логистики
грузоперевозок

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы _____ к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

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

1) *Назначение: программные средства для автоматизированного решения задач оптимизации транспортной логистики грузоперевозок.*

2) *Среда выполнения: операционная система Windows 10*

Языковые средства разработки: C#, UML

Инструментальные средства: MS Visual Studio 2022, PgAdmin 4, Draw.io,

Технологии разработки: ООП, принципы SOLID, шаблоны проектирования.

3) *Основные функции приложения:*

а) просмотреть путевые листы;

б) просмотреть расположение точек маршрута на карте;

в) сформировать путевой лист;

г) редактировать точки маршрута;

д) редактировать логистические ресурсы.

5. Перечень подлежащих разработке вопросов:

1) *Анализ предметной области и требований к системе;*

2) *Проектирование программных средств;*

3) Реализация программных средств;
4) Тестирование программных средств;
5) Оформление пояснительной записки.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

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

Реферат

Пояснительная записка содержит 84 листа, 40 рисунков, 16 таблиц и 15 использованных источников.

ТРАНСПОРТНАЯ ЛОГИСТИКА, ВИЗУАЛИЗАЦИЯ, ЗАДАЧА КОММИВОЯЖЕРА, МУРАВЬИНЫЙ АЛГОРИТМ, ГЕОГРАФИЧЕСКИЕ ДАННЫЕ.

Объект исследования – сервисы расчета и построения оптимальных логистических маршрутов, а также их визуальная интерпретация.

Цель выпускной квалификационной работы – спроектировать и разработать программные средства, позволяющие производить расчет оптимального маршрута на основе реальных географических данных.

Технологии разработки – разработка всех средств расчета и визуализации маршрута производится в среде разработки Microsoft Visual Studio 2022 на объектно-ориентированном языке программирования C#. Развертывание основного приложения осуществляется на операционных системах семейства Windows версии 10 и выше. В рамках обработки и хранения данных приложение взаимодействует с СУБД PostgreSQL версии 12. В качестве среды исполнения программных средств выступает персональный компьютер пользователя.

Результаты работы – в ходе выполнения выпускной квалификационной работы были спроектированы и разработаны программные средства, позволяющие осуществить расчет оптимального пути для заданных точек маршрута.

					ПГУ 09.03.04– 8BP201.02 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства автоматизации транспортной логистики грузоперевозок. Пояснительная записка			Лит.	Лист	Листов	
Разраб.		Акмашев С.В.									
Пров.		Балашова И.Ю.								4	84
								Группа 20ВП1			
Н. Контр.		Такташкин Д.В									
Утв.		Козлов А.Ю.									

Содержание

Введение.....	7
1 Анализ предметной области и постановка задачи на разработку программных средств автоматизации транспортной логистики грузоперевозок.....	8
1.1 Анализ предметной области транспортной логистики	8
1.2 Анализ алгоритмов оптимизации маршрутов.....	8
1.3 Анализ программного обеспечения аналогичного назначения	10
1.4 Постановка задачи на разработку программных средств.....	16
1.5 Анализ функциональных требований к программным средствам	17
1.6 Анализ нефункциональных требований к программным средствам	19
1.7 Планирование разработки и оценка бюджета.....	20
2 Проектирование программных средств автоматизации транспортной логистики грузоперевозок	23
2.1 Архитектура приложения.....	23
2.2 Проектирование структур данных	24
2.2.1 Концептуальная модель базы данных	24
2.2.2 Логическая модель базы данных	26
2.3 Проектирование программных средств.....	26
2.3 Проектирование пользовательского интерфейса.....	29
3 Реализация программных средств автоматизации транспортной логистики грузоперевозок	33
3.1 Реализация базы данных	33
3.2 Реализация алгоритмической части	35
3.3 Реализация основного приложения.....	40

4	Контроль качества программного обеспечения.....	42
4.1	Выбор режима тестирования	42
4.2	Тестирование алгоритмической части приложения.....	42
4.3	Тестирование пользовательской части приложения	44
4.4	Метрики кода.....	46
	Заключение	48
	Список использованных источников	49
	Приложение А (справочное) Глоссарий	51
	Приложение Б (рекомендуемое) Листинг приложения	53
	Приложение В (рекомендуемое) Листинг модульных тестов.....	74
	Приложение Г (рекомендуемое) Результаты тестирования	79

Введение

В современном мире транспортная логистика играет ключевую роль в обеспечении эффективной и безопасной доставки грузов. Программные средства автоматизации в этой области становятся все более важными для оптимизации процессов грузоперевозок и улучшения качества обслуживания клиентов. Они позволяют сократить время на планирование маршрутов, управление складскими запасами, контроль за транспортными средствами и мониторинг выполнения доставки. Для решения таких задач зачастую применяются алгоритмы по оптимизации маршрутов. Такие алгоритмы позволяют производить независимый расчет логистических ресурсов, которые будут использоваться в рамках грузоперевозок. Их высокая эффективность достигается за счет хорошо развитых технологий в области вычислительных устройств, что позволяет производить в достаточной степени точные расчеты на этапах планирования. В последние десятилетия муравьиный алгоритм стал одним из наиболее популярных методов оптимизации, применяемых в различных областях, включая транспортную логистику. В контексте транспортной логистики муравьиный алгоритм может быть применен для оптимизации маршрутов доставки грузов, планирования расписания транспортных средств, управления инвентаризацией и других задач, связанных с эффективным перемещением товаров.

Целью выпускной квалификационной работы является разработка программных средств автоматизации транспортной логистики грузоперевозок. Разработка включает в себя анализ требований, проектирование, реализацию и тестирование программного обеспечения. Данное приложение должно предоставлять удобный графический интерфейс для планирования маршрутов доставки грузов и нахождения наиболее оптимальных решений.

1 Анализ предметной области и постановка задачи на разработку программных средств автоматизации транспортной логистики грузоперевозок

1.1 Анализ предметной области транспортной логистики

С развитием информационных систем в области транспортной логистики значительно выросла необходимость оптимизации процессов доставки груза. На текущий момент вычисление оптимальных маршрутов становится неотъемлемой частью сервисов, использующих логистические алгоритмы. Конечное использование подобных технологий сводится к упрощенному пользовательскому интерфейсу с открытой картой местности, установки точек маршрута и их наглядного отображения. Развитие API картографических технологий позволяет вынести вычисления оптимизации маршрутов от пользовательских устройств к удаленным вычислительным центрам. Однако подобные технические решения имеют высокую стоимость и относительно небольшое разнообразие на рынке, в отличие от геоданных, которые могут предоставлять эти сервисы. В этом случае, применение муравьиного алгоритма будет недорогостоящим, достаточно простым и универсальным решением для задач различного спектра в рамках транспортной логистики [1].

В рамках транспортной логистики основное предназначение алгоритмов оптимизации маршрута является нахождение такого порядка посещения точек доставки, что общая длина пройденного при обходе точек пути была минимальна. Решение же данной задачи является ресурсоемким при использовании классических подходов перебора для большого числа точек выгрузки. Поэтому наиболее лучшим методом для вычисления оптимального маршрута будет использование метаэвристических алгоритмов.

1.2 Анализ алгоритмов оптимизации маршрутов

Одной из актуальных задач транспортной логистики является задача оптимизация маршрута транспорта при доставке грузов. Данная задача является

одной из фундаментальных задач комбинаторной оптимизации и известна как задача коммивояжера [2]. Задача коммивояжера является одной из наиболее известных NP-сложных задач и состоит в поиске кратчайшего гамильтонова цикла в графе. Одной из особенностей ее приложения в сфере логистики является большое число вершин графа, которые должен посетить коммивояжер, что влияет на эффективность применения известных алгоритмов решения данной задачи.

На рисунке 1 представлены распространенные подходы решения задачи коммивояжера.

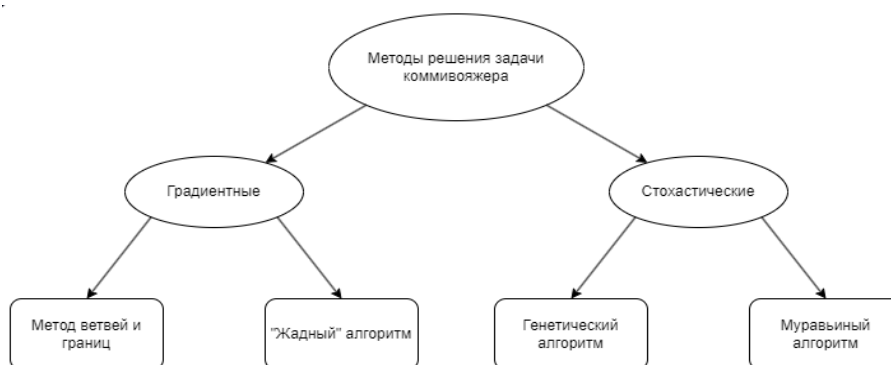


Рисунок 1 – Наиболее популярные подходы к решению задачи коммивояжера

Для сравнения стохастических и градиентных алгоритмов были выполнены вычислительные эксперименты для 10, 20, ..., 50 точек маршрута. Для программной реализации были взяты матрицы расстояний между точками. В качестве среды исполнения был использован персональный компьютер с 64-разрядной ОС Windows 10 с процессором AMD Ryzen 5 3600 с тактовой частотой 3.59 ГГц. В качестве показательных параметров для сравнения были выбраны L (км) – длина построенного маршрута за конечное число итераций для заданного количества точек и $T(c)$ – время, занимаемое алгоритмом для выполнения. Результаты подсчета приведены в таблице 1.

Таблица 1 – Результаты подсчета характеристик сравниваемых алгоритмов

	Градиентные алгоритмы				Стохастические алгоритмы			
	МВГ		ЖА		ГА		МА	
N	L(км)	T(c)	L(км)	T(c)	L(км)	T(c)	L(км)	T(c)
10	25,51	0,0942	26,09	0,0010	25,95	0,1110	26,20	0,1240

Продолжение таблицы 1

N	L(км)	T(с)	L(км)	T(с)	L(км)	T(с)	L(км)	T(с)
20	52,67	252,2904	73,87	0,0010	58,88	1,4030	59,01	1,0278
30	-	-	125,06	0,0011	97,91	3,9040	92,38	3,1569
40	-	-	197,77	0,0020	178,57	6,8710	169,62	5,3800
50	-	-	230,51	0,0021	212,29	11,4390	201,54	10,9205

Из таблицы 1 видно, что наилучшим образом на малом числе точек проявили себя градиентные алгоритмы. При большем числе точек маршрута, что типично для задач транспортной логистики, стохастические алгоритмы проявили себя лучше, в то время как использование градиентных алгоритмов было не применимо в силу несопоставимых затрат по времени исполнения. Среди стохастических алгоритмов МА находит оптимальные маршруты быстрее, чем ГА, что определяет преимущества его применения для решения задачи оптимизации маршрутов транспортной логистики [3].

1.3 Анализ программного обеспечения аналогичного назначения

Проанализировав рынок программных продуктов аналогичного назначения, можно выделить основные платформы, предоставляющие услуги и программные средства по автоматизации транспортной логистики грузоперевозок. К основным крупным платформам можно отнести следующие: Яндекс.Маршрутизация, Махотра, TMS LogistPRO, Шедекс и 1С Умная логистика. Общие сведения о рассматриваемых аналогах приведены в таблице 2.

Таблица 2 – Общие сведения о программном обеспечении аналогичного назначения

Название приложения	Организация	Общие сведения
Яндекс.Маршрутизация	ООО «Яндекс»	Облачная платформа для автоматического построения маршрутов и мониторинга их выполнения. Для визуального контроля планируемых маршрутов используется интеграция с сервисом «Яндекс.Карты»
Махотра	НПК «Маджента девелопмент»	Сервис для эффективного планирования маршрутов, контроля транспорта и выездных сотрудников. Подходит дистрибьюторским компаниям, а также ритейлерам и транспортным компаниям
TMS LogistPRO	ООО «С2Б ГРУПП»	Система автоматизации грузоперевозок. Может интегрировать сервисы с ERP-системами. По факту пользования предоставляется только функционал, который в последствии может дорабатываться со стороны клиента по необходимости

Продолжение таблицы 2

Название приложения	Организация	Общие сведения
Шедекс	ООО «Логистические программы»	Облачный сервис для оптимизации логистики. В составе имеет мобильное приложение для водителей, диспетчеров и менеджеров, а также включает систему отчетности и аналитики
1С Умная логистика	ООО «Умная Логистика»	Экосистема для автоматизации грузоперевозок. Подходит для транспортно-экспедиционных компаний и перевозчиков с собственным автопарком

Приведенные аналоги предоставляют различный функционал с различными средствами развертывания. Анализ данных приложений представляет собой выявление сильных и слабых сторон, а также общий вывод по использованию данных приложений.

Интерфейс для работы с приложением «Яндекс.Маршрутизация» представлен на рисунке 2.

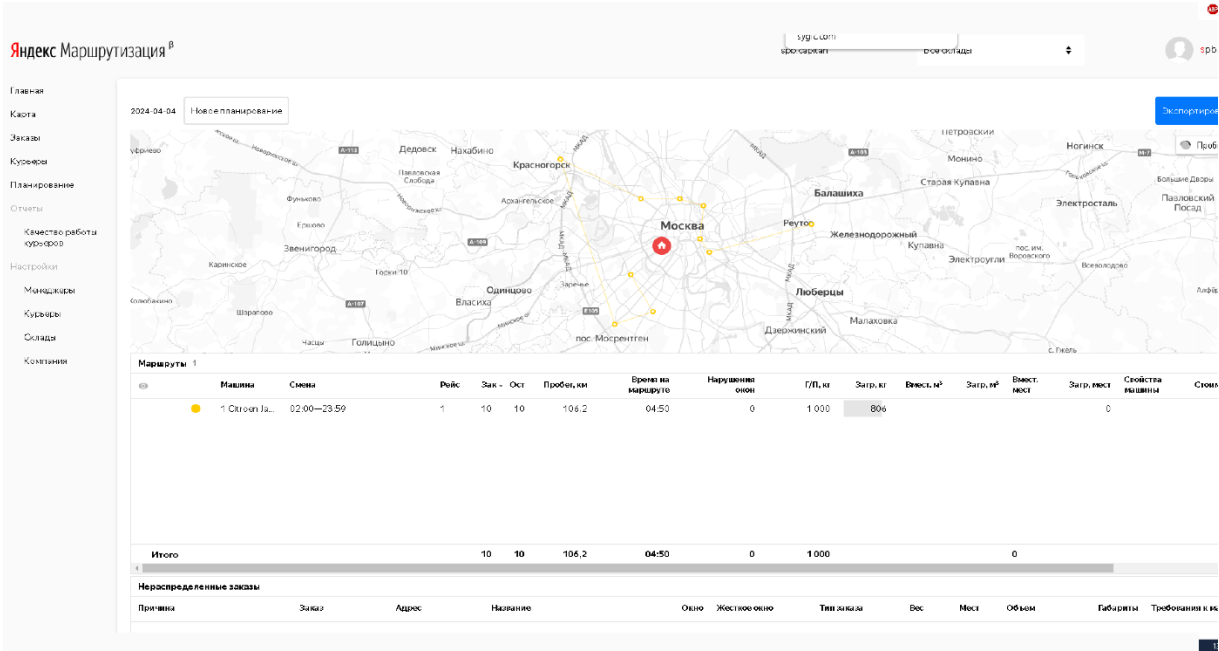


Рисунок 2 – Графический интерфейс приложения «Яндекс.Маршрутизация»

К основным преимуществам данного решения можно отнести следующие:

- наличие современного и удобного программного интерфейса;
- интеграция картографических сервисов для отслеживания маршрутов в реальном времени;
- прогноз загруженности дорог и учет актуальных географических данных о маршруте.

К недостаткам приложения можно отнести следующие:

- высокая стоимость программного решения;
- перегруженность функционала и большой порог для навыков пользования.

Сервис для эффективного планирования маршрутов Махортра предоставляет большой функционал в рамках специфики транспортной логистики. Графический интерфейс данного приложения приведен на рисунке 3.

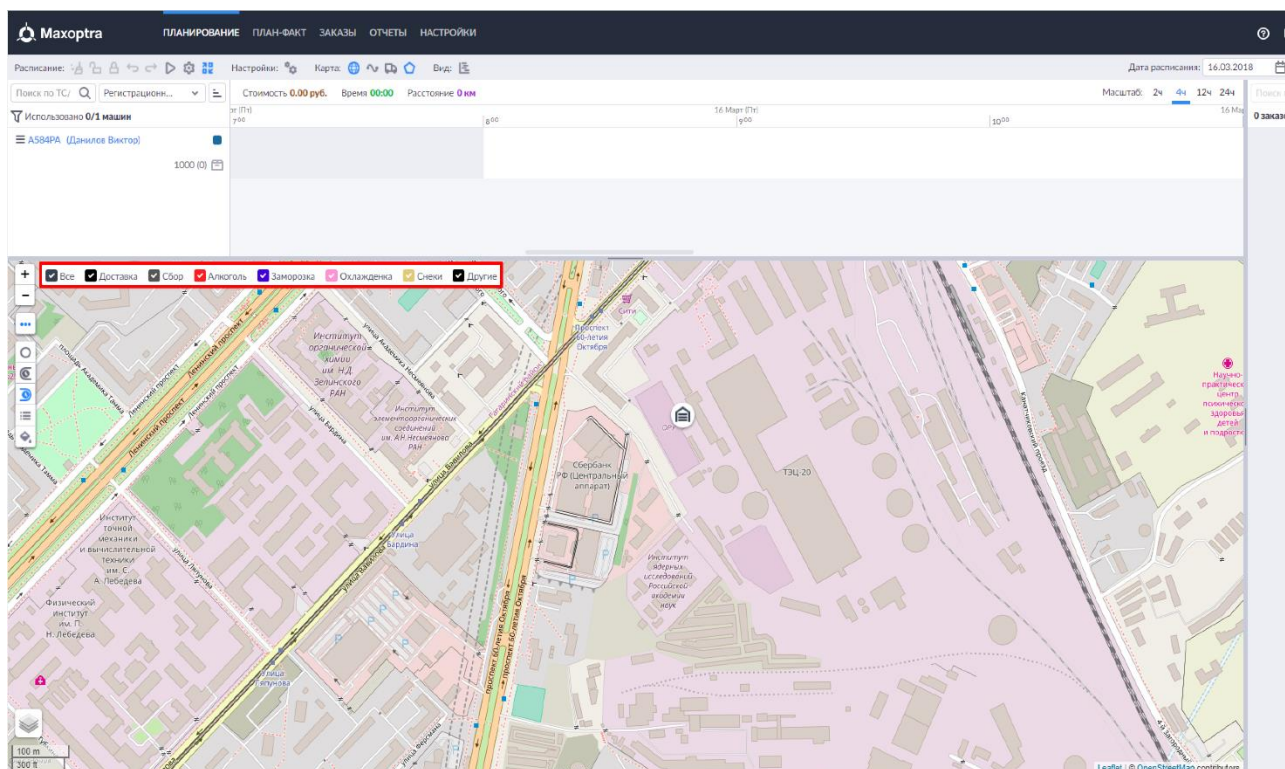


Рисунок 3 – Графический интерфейс приложения «Махортра»

К основным преимуществам данного решения можно отнести следующие:

- детальная аналитика в рамках перевозок;
- встроенная система оповещений о статусе перевозок;
- имеется возможность разделения зоны доставок для ограничения выстраиваемых маршрутов.

К недостаткам приложения можно отнести следующие:

- отсутствует дифференциальная стоимость для заказчика;
- имеется ограничение на количество исполнителей, новые исполнители увеличивают общую стоимость использования приложения.

Система автоматизации грузоперевозок «TMS LogistPRO» предоставляет эффективные функциональные возможности для планирования и контроля формируемых маршрутов. Графический интерфейс демонстрационной версии приложения представлен на рисунке 4.

ДЕМО

Заявки

Заказы

Запросы

Контрагенты

Подрядчики

Водители

Автомобили

Сотрудники

Юр.лица

Платежи

Зарегистрироваться

👤

⚙

Календарь

Таблица

< 25.03-31.03 08.04-14.04 >

📅 Сегодня

< 📅 Апрель 2024 >

< 2023 2025

	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота	Воскресенье
DAF - X 472 YY 25 20 т. / 33 плт. п: SCHMITZ - B 122 OP 237	1	2	3	4	5	6	
DAF - Y 447 YY 21 20 т. / 33 плт. п: SCHMITZ - P 768 YA 63	1	2	3	4	5	6	
DAF - P 782 KT 653 20 т. / 33 плт. п: SCHMITZ - P 668 MA 72	1	2	3	4	5	6	
FORD - M 725 EA 254 10 т. / 18 плт.	1	2	3	4	5	6	
FORD - M 317 KO 86 10 т. / 18 плт.	1	2	3	4	5	6	
	1	2	3	4	5	6	

Рисунок 4 – Графический интерфейс приложения «TMS LogistPRO»

К основным преимуществам данного решения можно отнести следующие:

- автоматическое распределение заявок по заданным критериям;
- возможность ведения электронного документооборота;
- наличие большой корпоративной базы перевозчиков.

К недостаткам приложения можно отнести следующие:

- из представленных перевозчиков существует возможность создания заявок только для своих перевозчиков;
- дизайн интерфейса является устаревшим вариантом современных решений.

Облачный сервис для оптимизации логистики «Шедекс» предоставляет программное решение не только для диспетчеров логистики, но и для водителей. Графический интерфейс данного приложения представлен на рисунке 5.

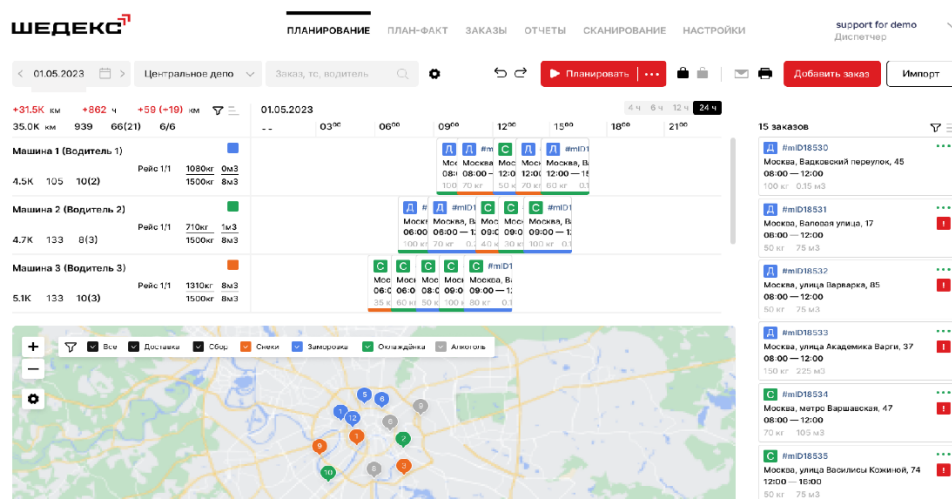


Рисунок 5 – Графический интерфейс приложения «Шедекс»

К основным преимуществам данного решения можно отнести следующие:

- прогноз времени прибытия с учетом пробок и других ограничений;
- фотофиксация этапов выполнения работ;
- дифференциальная стоимость услуг.

К недостаткам приложения можно отнести следующие:

- интуитивно сложный графический функционал;
- отсутствие электронного формирования документов.

Экосистема автоматизации грузоперевозок «1С Умная логистика» предоставляет инструменты для управления и планирования маршрутов грузоперевозок. Графический интерфейс данного приложения представлен на рисунке 6.

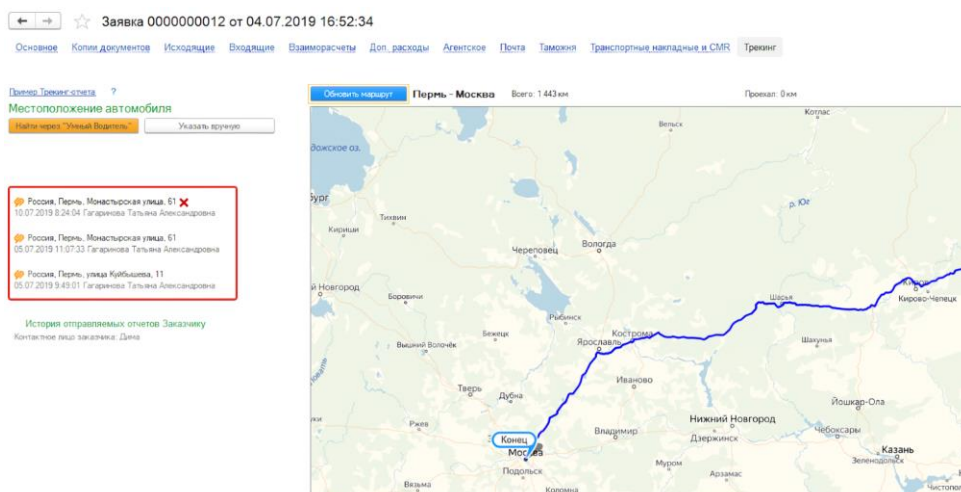


Рисунок 6 – Графический интерфейс приложения «1С Умная логистика»

К основным преимуществам данного решения можно отнести следующие:

- наличие электронного документооборота;
- формирование аналитических отчетов;
- настройка формирующихся закрывающих документов.

К недостаткам приложения можно отнести следующие:

- высокая стоимость использования;
- дизайн интерфейса является устаревшим вариантом современных решений.

Для проведения сравнительного анализа приложений аналогов был рассмотрен набор критериев, на основе которых производится сравнение. Для рассматриваемых критериев были установлены следующие значения баллов:

- 0 баллов – критерий не выполнен;
- 1 балл – критерий выполнен частично;
- 2 балла – критерий выполнен полностью.

Результаты сравнения по заданным критериям приведены в таблице 3.

Таблица 3 – Результаты сравнения разрабатываемых программных средств с аналогами

Критерий	Разрабатываемое приложение	Яндекс Маршрутизация	Махотра	TMS LogistPRO	Шедекс	1С Умная логистика
Современный интерфейс	2	2	2	1	2	0
Эффективность графического интерфейса	2	0	2	0	1	1
Поддержка и формирование электронной документации	1	2	2	1	0	2
Низкая стоимость использования	2	0	1	2	1	1
Отсутствие ограничений на создаваемые ресурсы	2	1	1	0	0	1
Интеграции с другими сервисами	2	2	2	1	2	2
Итого	11	7	10	5	6	7

На основе результатов сравнения можно сделать вывод, что разрабатываемые программные средства автоматизации транспортной

логистики грузоперевозок способны в достаточной мере выполнять необходимые задачи. Разрабатываемое приложение обладает современным интерфейсом, а также имеет низкую стоимость использования, что позволит эффективно выполнять планирование маршрутов без больших затрат.

1.4 Постановка задачи на разработку программных средств

Основной задачей на разработку программных средств автоматизации транспортной логистики грузоперевозок является создание приложения, позволяющего выполнять расчет оптимальных маршрутов для заданных координат. В качестве дополнительных условий пользования приложения можно выделить удобство и эффективность графического интерфейса для взаимодействия с пользователем.

Для формирования автоматического расчета данных и маршрутов, необходимо выделить логистические ресурсы, на основе которых будет произведен расчет. К основным логистическим ресурсам можно отнести задействованный транспорт и трудовые ресурсы. Используемый для доставки транспорт обладает рядом характеристик, которые необходимо учитывать для расчета затрат на передвижение по сформированному маршруту. К данным характеристикам можно отнести расход топлива на пройденный путь в 100 километров, грузоподъемность и габариты. На основании выделенных характеристик, можно обозначить основные наборы данных для использования – путевой лист, водитель и транспорт. Таким образом, для использования и хранения полученных в ходе расчетов сведений, целесообразно задействовать СУБД. Это позволит оперировать данными о предыдущих расчетах, а также назначать и контролировать логистические ресурсы.

Также, для более эффективного взаимодействия пользователя с графическими элементами управления расчетом, необходимо распределить элементы графического интерфейса по групповым признакам. На основе выделенных признаков принадлежности инструментов транспортной логистики, можно сформировать следующие разделы программы:

- главная страница;
- работа с графической картой;
- ресурсы;
- путевой лист.

Таким образом, на основе сформированных требований будут произведены проектирование и разработка программных средств автоматизации транспортной логистики грузоперевозок.

1.5 Анализ функциональных требований к программным средствам

Диаграмма вариантов использования используется для визуализации функциональных требований к системе или программному продукту [4]. Она позволяет понять, как пользователи будут взаимодействовать с системой, какие действия они будут выполнять, какие варианты использования системы существуют, а также кто участвует в этих вариантах использования. Диаграмма вариантов использования помогает лучше понять потребности пользователей и определить функциональные требования к системе. Разработанная диаграмма вариантов использования приведена на рисунке 7.

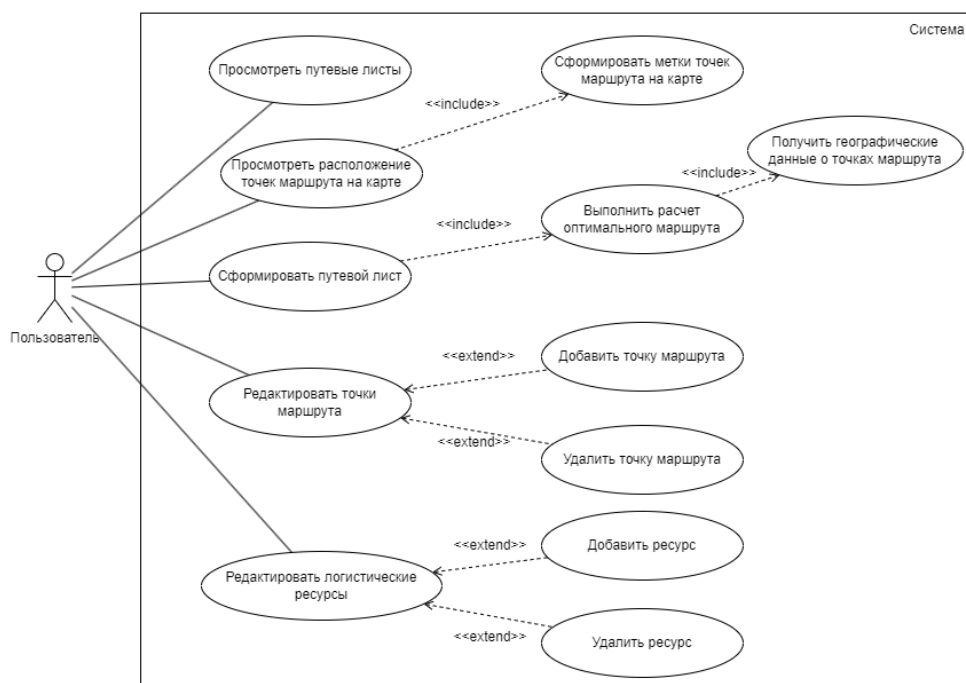


Рисунок 7 – Диаграмма вариантов использования приложения

Основным управляющим элементом системы является пользователь, который посредством взаимодействия с графическим интерфейсом может использовать предоставляемый функционал. Главные варианты использования представляют собой возможности использования разделов приложения. Спецификация варианта использования «Просмотреть расположение точек маршрута на карте» приведена на рисунке 8.

Прецедент: Просмотреть расположение точек маршрута на карте
ID: 2
Краткое описание: Пользователь просматривает графическую карту маршрута
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. В маршруте присутствует хотя бы одна точка. 2. Выбрана стартовая точка маршрута.
Основной поток: 1. Пользователь открывает раздел работы с графической картой. 2. Выполняется запрос на формирование карты. 3. Сформированная карта отображается в виде графического элемента.
Постусловия: 1. Карта отображает элементы маршрута.
Альтернативные потоки: Нет

Рисунок 8 – Спецификация варианта использования «Просмотреть расположение точек маршрута на карте»

Спецификация варианта использования «Сформировать путеводный лист» приведена на рисунке 9.

Прецедент: Сформировать путеводный лист
ID: 3
Краткое описание: Пользователь запускает расчет оптимального маршрута
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. В маршруте присутствует хотя бы одна точка. 2. Выбрана стартовая точка маршрута.
Основной поток: 1. Пользователь открывает раздел работы с графической картой. 2. Пользователь запускает расчет оптимального маршрута. 2.1. Система формирует запрос на получение географических координат точек маршрута. 2.2. Система производит расчет оптимального маршрута 3. Точки маршрута сортируются в соответствии с результатом расчета
Постусловия: 1. Карта маршрута обновлена.
Альтернативные потоки: Нет

Рисунок 9 – Спецификация варианта использования «Сформировать путеводный лист»

Спецификация варианта использования «Просмотреть путевые листы» приведена на рисунке 10.

Прецедент: Просмотреть путевые листы
ID: 1
Краткое описание: Пользователь просматривает список сформированных путевых листов
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Существует хотя бы один ранее сформированный путевой лист.
Основной поток: 1. Пользователь открывает раздел главной страницы 2. Система предлагает список всех сохраненных путевых листов.
Постусловия: 1. Отображен список путевых листов
Альтернативные потоки: Нет

Рисунок 10 – Спецификация варианта использования «Просмотреть путевые листы»

Таким образом, проанализированные функциональные возможности приложения определяют основные способы взаимодействия с пользователем. Это позволяет выделить бизнес-процессы приложения и сформировать минимальный набор реализуемых функций для корректной работоспособности.

1.6 Анализ нефункциональных требований к программным средствам

Разработка программных средств автоматизации транспортной логистики грузоперевозок должна быть выполнена в многофункциональной среде разработки Microsoft Visual Studio 2022 на объектно-ориентированном языке программирования C#. Графический интерфейс должен быть построен с использованием языка разметки XAML. В качестве СУБД должна выступать PostgreSQL. Взаимодействие с СУБД необходимо разработать с использованием фреймворка EntityFramework 6. Приложение должно разворачиваться и исполняться на операционных системах семейства Windows версии 10 и выше.

1.7 Планирование разработки и оценка бюджета

Для оценки временных затрат, а также наглядного планирования процесса разработки была построена диаграмма Ганта с использованием инструментов Project Libre. Общий процесс разработки приложения содержит четыре вехи. Каждый этап включает в себя некоторый набор подзадач, которые необходимо выполнить для перехода на новый этап. Для разработки данного приложения целесообразно выделить следующие этапы:

- 1) анализ предметной области и постановка задачи на разработку программных средств автоматизации транспортной логистики грузоперевозок;
- 2) проектирование программных средств автоматизации транспортной логистики грузоперевозок;
- 3) реализация программных средств автоматизации транспортной логистики грузоперевозок;
- 4) контроль качества программного обеспечения.

Краткое описание задач каждого этапа приведено на рисунке 11.

Название	Продолж...	Начало	Окончание
Начало проекта	0 дней	08.04.24 9:00	08.04.24 9:00
Анализ предметной области и постановка задачи на разр	5 дней	08.04.24 9:00	12.04.24 18:00
Анализ предметной области	0,375 дней	08.04.24 9:00	08.04.24 12:00
Анализ используемых алгоритмов	0,625 дней	08.04.24 12:00	08.04.24 18:00
Анализ программного обеспечения аналогичного назначения	1 день	09.04.24 9:00	09.04.24 18:00
Постановка задачи на разработку программных средств	1 день	10.04.24 9:00	10.04.24 18:00
Анализ функциональных требований	0,375 дней	11.04.24 9:00	11.04.24 12:00
Анализ нефункциональных требований	0,625 дней	11.04.24 12:00	11.04.24 18:00
Планирование разработки и оценка бюджета	1 день	12.04.24 9:00	12.04.24 18:00
Проектирование пользовательских средств автоматизац	9 дней	15.04.24 9:00	25.04.24 18:00
Архитектура приложения	3 дней	15.04.24 9:00	17.04.24 18:00
Проектирование структур данных	2 дней	18.04.24 9:00	19.04.24 18:00
Проектирование программных средств	2 дней	22.04.24 9:00	23.04.24 18:00
Проектирование пользовательского интерфейса	2 дней	24.04.24 9:00	25.04.24 18:00
Реализация программных средств автоматизации трансп	11 дней	26.04.24 9:00	10.05.24 18:00
Реализация базы данных	2 дней	26.04.24 9:00	29.04.24 18:00
Реализация алгоритмической части	3 дней	30.04.24 9:00	02.05.24 18:00
Реализация основного приложения	6 дней	03.05.24 9:00	10.05.24 18:00
Контроль качества программного обеспечения	8 дней	13.05.24 9:00	22.05.24 18:00
Выбор режима тестирования	1 день	13.05.24 9:00	13.05.24 18:00
Тестирование алгоритмической части приложения	4 дней	14.05.24 9:00	17.05.24 18:00
Тестирование пользовательской части приложения	2 дней	20.05.24 9:00	21.05.24 18:00
Метрики кода	1 день	22.05.24 9:00	22.05.24 18:00
Оформление отчета	8 дней	23.05.24 9:00	03.06.24 18:00
Оформление пояснительной записки	8 дней	23.05.24 9:00	03.06.24 18:00
Конец проекта	0 дней	03.06.24 18:00	03.06.24 18:00

Рисунок 11 – Сведения об этапах разработки приложения

	Название	Стандартная ставка	Ставка сверхурочных
	Аналитик	100/час	180/час
	Разработчик	250/час	500/час
	Тестировщик	120/час	240/час
	Компьютер	75 000/час	0/час
	Интернет	0/час	0/час

Рисунок 13 – Стоимость задействованных ресурсов

Общая оценка финансовых затрат проекта представлена на рисунке 14.

Название	Оставшаяся стоимость
Начало проекта	0 руб.
Анализ предметной области и постановка задачи на разрабо	4000 руб.
Анализ предметной области	300 руб.
Анализ используемых алгоритмов	500 руб.
Анализ программного обеспечения аналогичного назначения	800 руб.
Постановка задачи на разработку программных средств	800 руб.
Анализ функциональных требований	300 руб.
Анализ нефункциональных требований	500 руб.
Планирование разработки и оценка бюджета	800 руб.
Проектирование пользовательских средств автоматизации т	23600 руб.
Архитектура приложения	8400 руб.
Проектирование структур данных	5600 руб.
Проектирование программных средств	4000 руб.
Проектирование пользовательского интерфейса	5600 руб.
Реализация программных средств автоматизации транспорт	22000 руб.
Реализация базы данных	4000 руб.
Реализация алгоритмической части	6000 руб.
Реализация основного приложения	12000 руб.
Контроль качества программного обеспечения	9680 руб.
Выбор режима тестирования	960 руб.
Тестирование алгоритмической части приложения	3840 руб.
Тестирование пользовательской части приложения	1920 руб.
Метрики кода	2960 руб.
Оформление отчета	6400 руб.
Оформление пояснительной записки	6400 руб.
Конец проекта	0 руб.
	Итого: 65680 руб.

Рисунок 14 – Общая оценка финансовых затрат

Таким образом, исходя из оценки финансовых и временных затрат, видно, что предполагаемые затраты на разработку составляют 65 680 рублей, а общая продолжительность – 41 день.

2 Проектирование программных средств автоматизации транспортной логистики грузоперевозок

2.1 Архитектура приложения

Согласно выделенным требованиям к программным средствам, была разработана диаграмма коммуникаций, отражающая основной процесс взаимодействия компонентов приложения. Разработанная диаграмма приведена на рисунке 15.

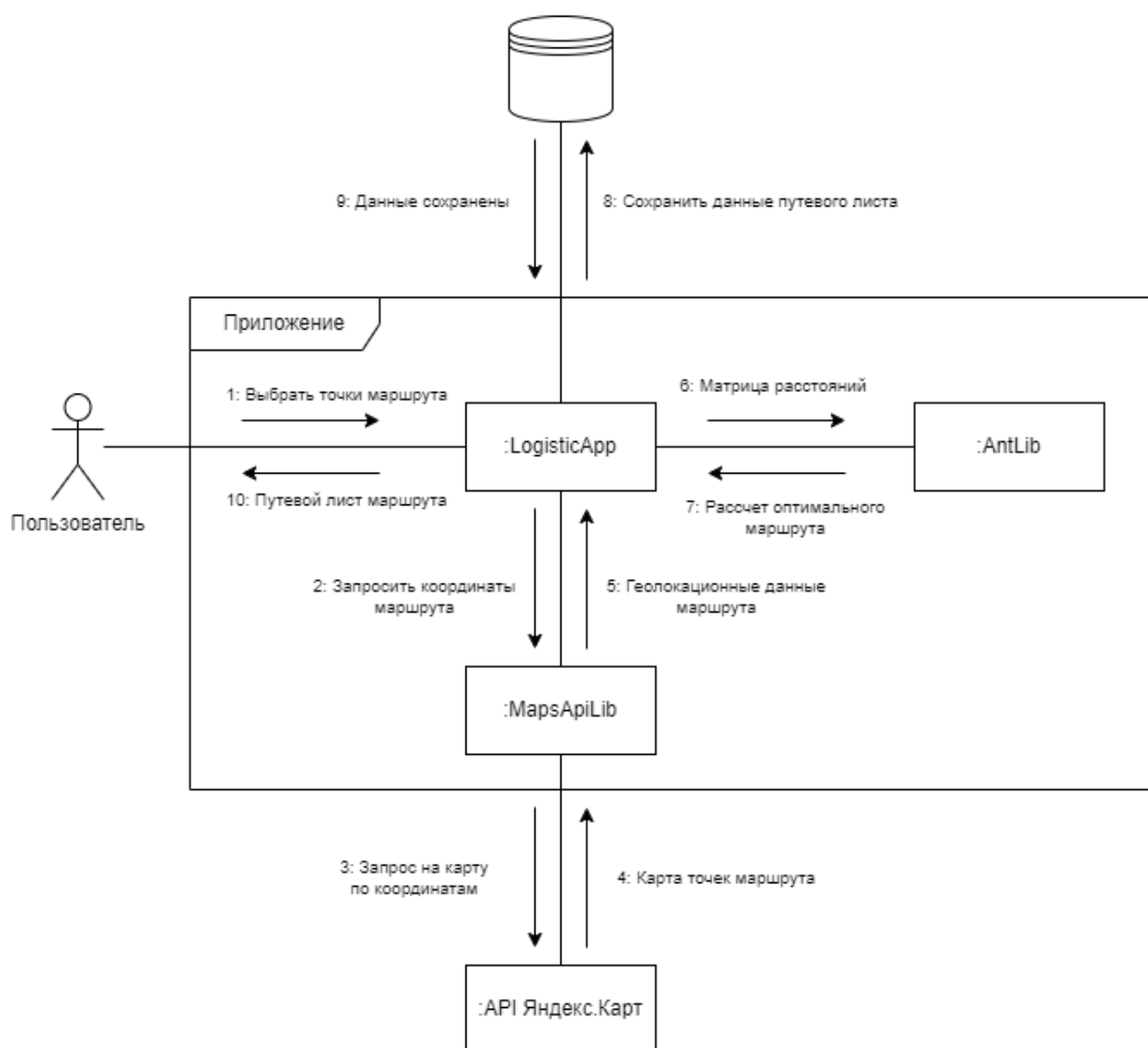


Рисунок 15 – Диаграмма коммуникаций приложения

Данная диаграмма отражает основные архитектурные особенности приложения, а также демонстрирует взаимодействие компонентов на разных

уровнях. Основной частью приложения является компонент LogisticApp, с которым взаимодействует пользователь. Исходными данными для работы приложения являются точки маршрута, которые задаются пользователем. Дальнейшая работа приложения предполагает перевод точек маршрута в координаты с помощью компонента MapsApiLib. Данный компонент перевод координаты согласно представленным данным с интерфейса Яндекс.Карт и позволяет сформировать матрицу расстояний. С помощью сгенерированной матрицы расстояний, компонент AntLib производит вычисления оптимального маршрута на основе применения муравьиного алгоритма, и выдает максимально приближенный результат. Конечным результатом работы программы является обработка найденного оптимального маршрута главным приложением и формирование формального путевого листа для представления пользователю. До вывода путевого листа пользователю, приложение сохраняет информацию о сформированном маршруте. Сохраняемые данные включают в себя сведения о самом содержании маршрута, а также сведения о предполагаемом водителе и транспорте. Ввод информации о логистических ресурсах может осуществляться пользователем предварительно. При расчете приложение автоматически подберет необходимые данные, если ручной ввод не был осуществлен.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Концептуальная модель данных представляет собой наглядное отражение организации данных на высоком уровне. Данное представление может демонстрировать описание сущностей, их атрибуты и отношения без указания технических деталей реализации [5].

Согласно выделенным требованиям к приложению по хранению и организации данных была разработана диаграмма типа «сущность-связь». Данная диаграмма представлена на рисунке 16.

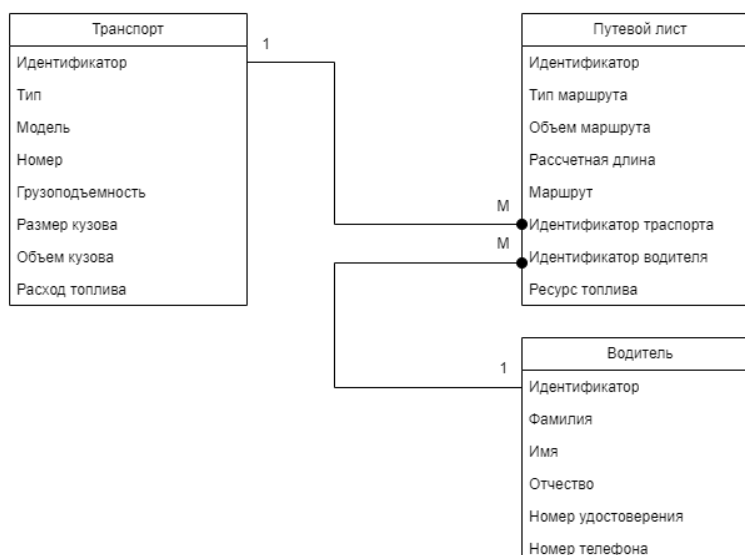


Рисунок 16 – Графическое представление концептуальной модели

Разработанная диаграмма наглядно отражает хранимые элементы приложения и их связи. В таблице 4 представлены сущности, сформированные согласно входным и выходным данным бизнес-процессов.

Таблица 4 – Описание сущностей

Наименование сущности	Идентификатор сущности	Назначение сущности
Путевой лист	Waybill	Содержит сведения о построенных маршрутах, их участниках и расходных ресурсах
Водитель	Driver	Содержит сведения о водителях, назначаемых на маршруты
Транспорт	Vehicle	Содержит сведения о транспортных единицах в рамках логистических грузоперевозок

В таблице 5 представлено описание связей между сущностями.

Таблица 5 – Описание связей

Наименование связи	Описание	Мощность
Vehicle-Waybill	Определяет принадлежность транспортных единиц к путевым листам	1:M
Driver-Waybill	Определяет принадлежность водителей к маршрутам путевых листов	1:M

Концептуальная модель данных построена в графической среде draw.io в нотации IDEF1X [6].

2.2.2 Логическая модель базы данных

Логическая модель данных является расширением концептуальной модели, в которой сущности, атрибуты и связи дополнительно детализированы и организованы. Логическая модель данных является основой физической модели, и раскрывает представление данных более детально с технической точки зрения [7].

На основе концептуальной модели данных была построена логическая модель, представленная на рисунке 17.

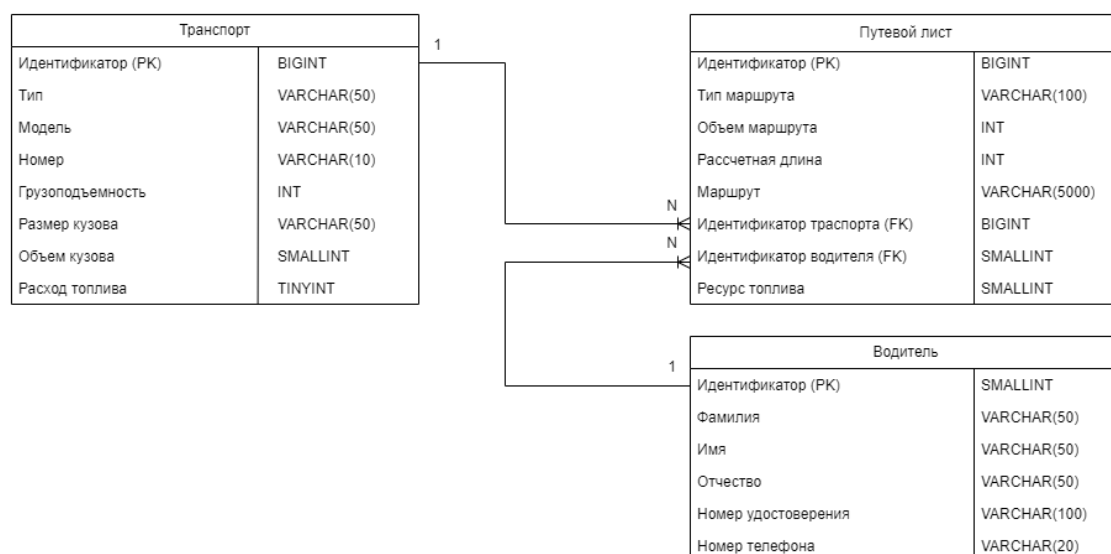


Рисунок 17 – Графическое представление логической модели

Логическая модель была построена в среде draw.io в нотации моделирования Crow's Foot [8].

Таким образом, спроектированные графические представления организации и хранения данных позволяют выполнить техническую реализацию обращения к базе данных приложения, а также наглядно отразить структурные связи между данными.

2.3 Проектирование программных средств

На основе архитектуры приложения была спроектирована диаграмма классов, приведенная на рисунке 18. Данная диаграмма позволяет наглядно

отобразить основные технические решения, применяемые в рамках дальнейшей разработки приложения.



Рисунок 18 – Диаграмма классов основного приложения

В таблице 6 представлена спецификация классов основного приложения.

Таблица 6 – Спецификация классов

Класс	Описание класса	Поля	Описание полей	Методы	Описание методов
MainWindow	Класс основного окна приложения	TransferData	Сведения о построенном маршруте	NavigateToHome()	Переход на главную страницу
		_currentPage	Текущая страница приложения	NavigateToMap()	Переход на страницу с картой маршрута
		_cache	Временные данные расчета маршрута	NavigateToProducts()	Переход на страницу склада
				NavigateToReport()	Переход на страницу путевого листа
				ChangeSettings()	Переход на страницу настроек

Продолжение таблицы 6

Класс	Описание класса	Поля	Описание полей	Методы	Описание методов
MapsTool	Класс модуля запросов геоанных	Distance Matrix[[]]	Матрица расстояний	GetGeoData()	Получить геоанные
		Length	Длина маршрута	GetMapImage()	Получить изображение карты
		MapImage	Изображение карты	SetMapSettings()	Установить настройки карты
		MapSize	Размер карты	GetDistanceData()	Получить сведения о расстоянии
		Map Resolution	Разрешение карты		
		apiQueryMap	Запрос на получение карты		
		apiQueryData	Запрос на получение геоанных		
Optimization Tool	Класс модуля расчета маршрута	alphaParam	Альфа-параметр алгоритма	GetBestSolution()	Рассчитать маршрут
		betaParam	Бета-параметр алгоритма	InitAnts()	Создать популяцию для расчета
		pDecFactor	Фактор уменьшения точности	ChangeTarget()	Изменить конечную точку маршрута
		pIncFactor	Фактор увеличения точности	BuildTrail()	Построить стартовый маршрут
		AntsCount	Популяция колонии	Update()	Обновить маршрут
		MaxIterations	Время расчета	BestTrail()	Улучшить маршрут
Optimization Data	Класс хранимой информации о маршруте	route[]	Маршрут	GetRoute()	Получить маршрут
		Length	Длина маршрута	OptimizationData()	Создать объект данных маршрута
		Resource Param	Параметр логистических ресурсов		
		Distance	Общая дистанция		
		StartPoint	Стартовая точка маршрута		
		Count	Число точек маршрута		

Спроектированная диаграмма классов главного приложения позволяет организовать структуру приложения, а также уточнить технические детали реализации.

2.3 Проектирование пользовательского интерфейса

Согласно выделенным функциональным требованиям к приложению был спроектирован графический интерфейс пользователя. Данный интерфейс должен предоставлять беспрепятственный доступ к основным возможностям приложения, а также располагать эффективными и простыми инструментами взаимодействия с приложением.

Проектируемое приложение является многостраничным. Это подразумевает наличие нескольких разделов доступа к отдельным модулям приложения. Данный выбор архитектуры пользовательского интерфейса обусловлен умеренным количеством размещаемых элементов управления по различным частям приложения, которые требуют уникального взаимодействия с пользователем. Проектируемый пользовательский интерфейс представляет собой четыре раздела управления. Раздел главной страницы представлен на рисунке 19.

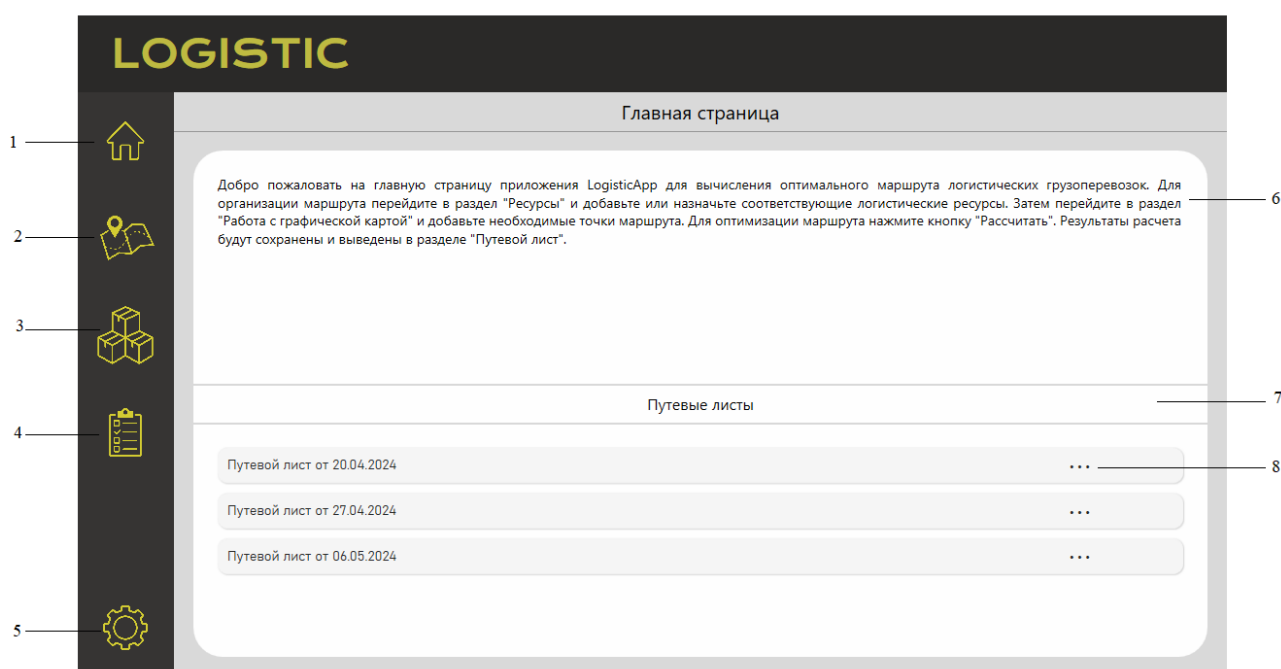


Рисунок 19 – Графический интерфейс раздела «Главная страница»

Главное окно приложения представлено следующими графическими элементами:

1 – кнопка перехода на главную страницу;

- 2 – кнопка перехода на страницу работы с графической картой;
- 3 – кнопка перехода на страницу назначения ресурсов;
- 4 – кнопка перехода на страницу путевого листа;
- 5 – кнопка перехода в настройки приложения;
- 6 – общее описание приложения;
- 7 – блок просмотра сформированных путевых листов;
- 8 – кнопка настройки и просмотра путевого листа.

Раздел страницы работы с графической картой представлен на рисунке 20.

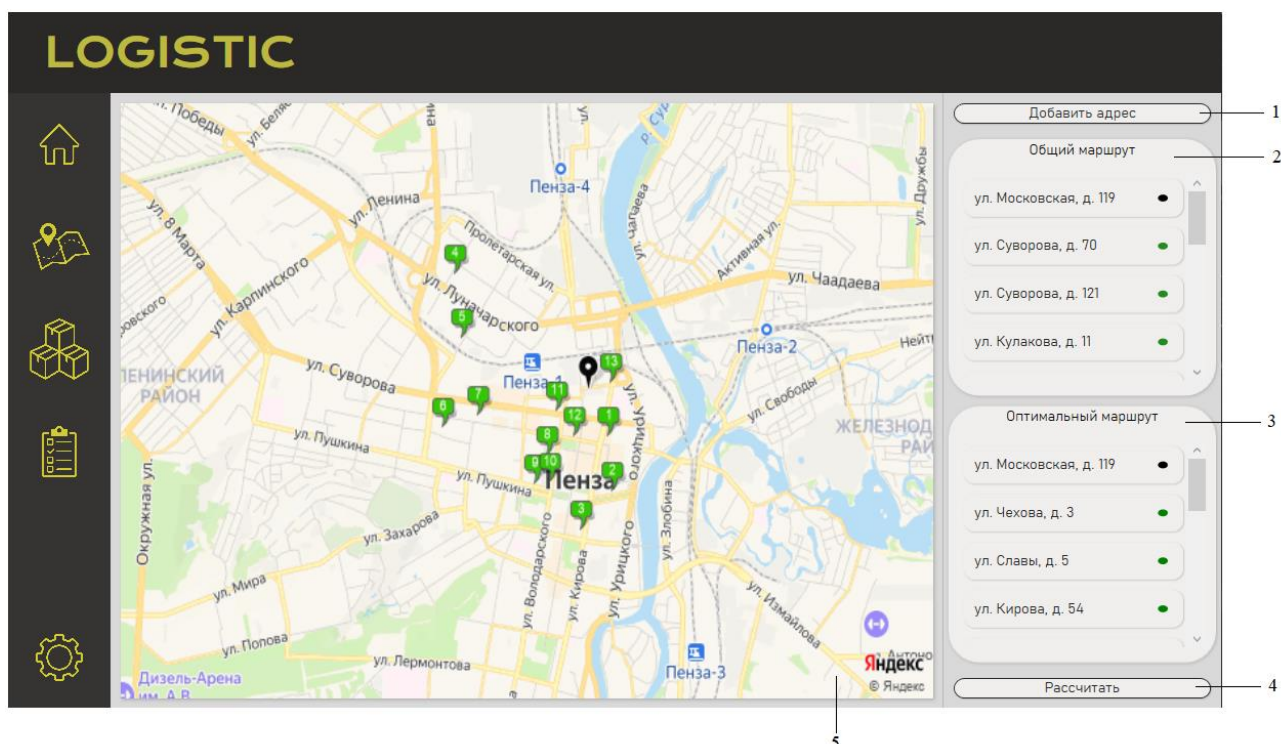


Рисунок 20 – Графический интерфейс раздела «Работа с графической картой»

Данный раздел представлен следующими графическими элементами:

- 1 – кнопка добавления нового адреса в маршрут;
- 2 – список добавленных точек маршрута;
- 3 – упорядоченный список точек оптимального маршрута;
- 4 – кнопка выполнения расчета оптимизации маршрута;
- 5 – карта для наглядного изображения порядка маршрута.

Работы с графической картой предоставляет возможность интерактивного просмотра маршрута пользователем. Это делает более эффективным планирование и создание новых маршрутов.

Раздел страницы ресурсов представлен на рисунке 21.

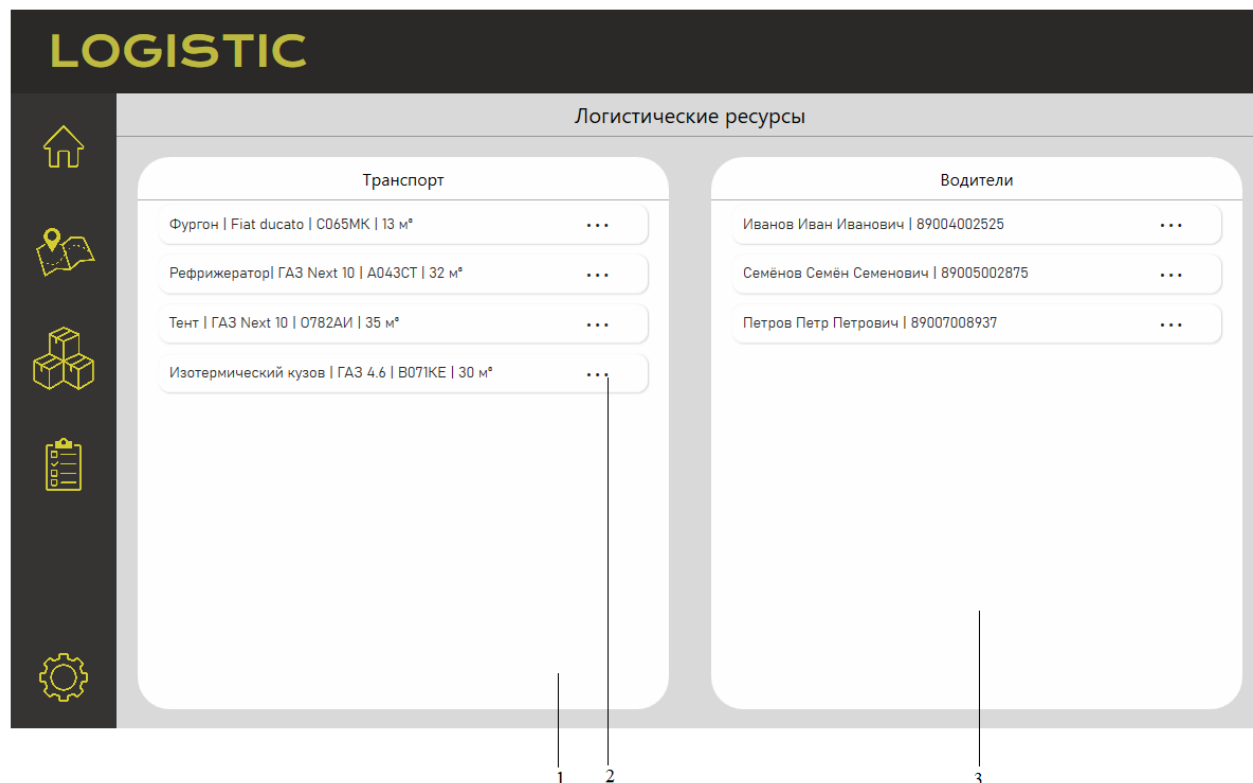


Рисунок 21 – Графический интерфейс раздела «Ресурсы»

Страница ресурсов представлена следующими графическими элементами:

- 1 – список используемых транспортных средств для доставки с краткой характеристикой;
- 2 – кнопка подробной настройки логистического ресурса;
- 3 – список задействованных водителей.

Страница ресурсов позволяет пользователю предварительно выбрать и назначить логистические ресурсы для планирования маршрута. При добавлении новых элементов, все ранее доступные ресурсы обновляются в базе данных и формируется новый список. Каждый элемент может редактироваться, что позволяет выполнять гибкую настройку и уточнение деталей ресурса.

Раздел страницы путевого листа приведен на рисунке 22.

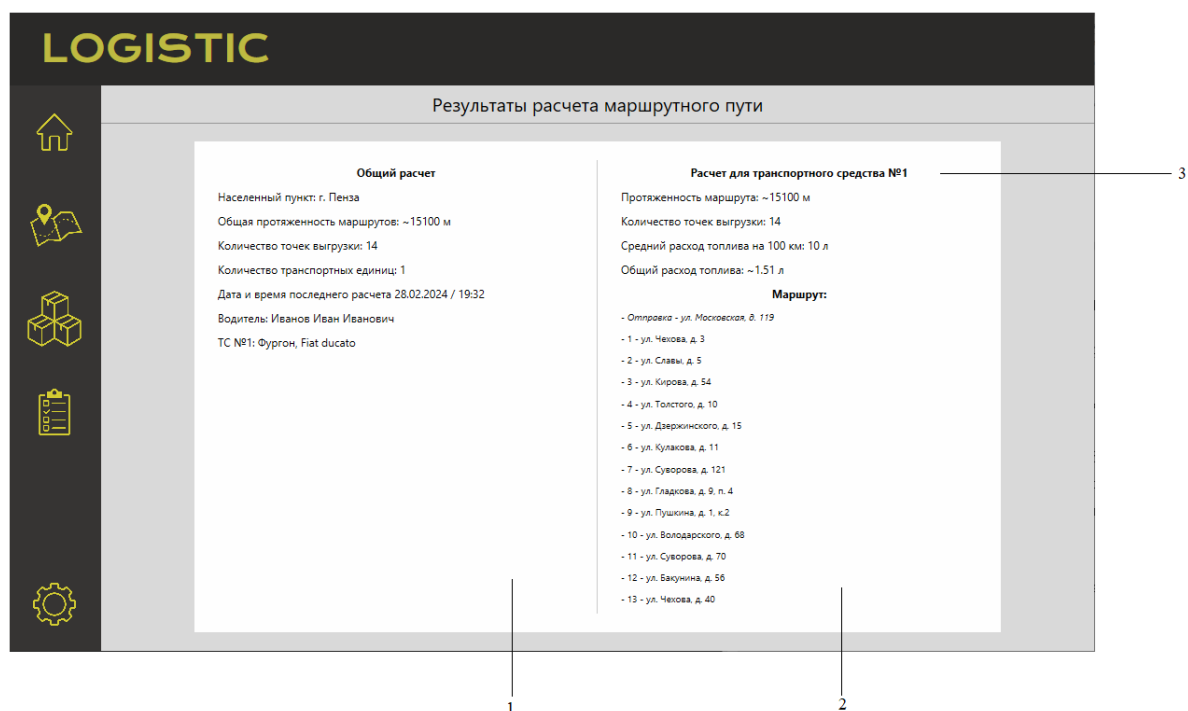


Рисунок 22 – Графический интерфейс раздела «Путевой лист»

Данный раздел представлен следующими графическими элементами:

- 1 – страница с общими данными о маршруте;
- 2 – блок сведений о расходах на используемое транспортное средство;
- 3 – блок сведений о посещаемых точках разгрузки.

Таким образом, спроектированный пользовательский интерфейс позволяет пользователю эффективно использовать основной функционал приложения в комфортном для представления виде.

3 Реализация программных средств автоматизации транспортной логистики грузоперевозок

3.1 Реализация базы данных

Физическая модель базы данных представляет собой логическую модель, выраженную в терминологии языка описания конкретной СУБД. В рамках данного приложения используется СУБД PostgreSQL.

Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне [9].

Согласно концептуальной и логической модели был разработана физическая модель данных, представленная на рисунке 23.

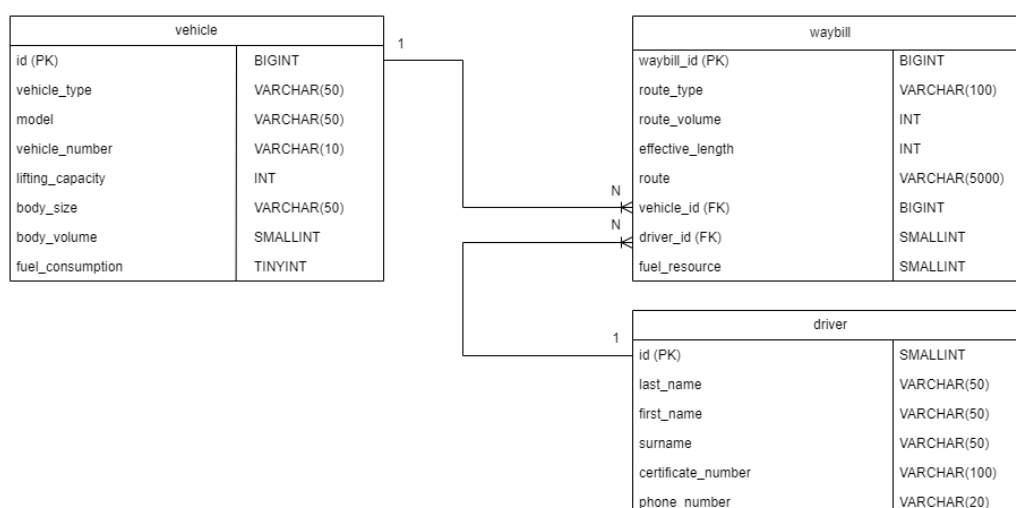


Рисунок 23 – Физическая модель базы данных

На основе физической модели, была разработана база данных с использованием СУБД PostgreSQL. В таблице 7 представлена спецификация сущности «Путевой лист».

Таблица 7 – Спецификация сущности «Путевой лист»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
waybill_id	Идентификатор	BIGINT	Нет	> 0	–	PK
route_type	Тип маршрута	VARCHAR(100)	Нет	Любое	–	–

Продолжение таблицы 7

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
route_volume	Объем маршрута	INT	Нет	> 0	–	–
effective_length	Длина маршрута	INT	Нет	> 0	–	–
route	Маршрут	VARCHAR(1000)	Нет	Любое	–	–
vehicle_id	Идентификатор транспорта	BIGINT	Нет	> 0	–	FK
driver_id	Идентификатор водителя	SMALLINT	Нет	> 0	–	FK
fuel_resource	Ресурс топлива	SMALLINT	Нет	> 0	–	–

В таблице 8 представлена спецификация сущности «Водитель».

Таблица 8 – Спецификация сущности «Водитель»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	SMALLINT	Нет	> 0	–	PK
last_name	Фамилия	VARCHAR(50)	Нет	Любое	–	–
first_name	Имя	VARCHAR(50)	Нет	Любое	–	–
middle_name	Отчество	VARCHAR(50)	Да	Любое	–	–
certificate_number	Номер водительского удостоверения	VARCHAR(100)	Нет	Любое	–	–
phone_number	Номер телефона	VARCHAR(20)	Да	Любое	–	–

В таблице 9 представлена спецификация сущности «Водитель».

Таблица 9 – Спецификация сущности «Транспорт»

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
id	Идентификатор	BIGINT	Нет	> 0	–	PK
vehicle_type	Тип транспортного средства ()	VARCHAR(50)	Нет	Любое	–	–
model	Модель транспортного средства	VARCHAR(50)	Нет	Любое	–	–

Продолжение таблицы 9

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
vehicle_number	Государственный номер	VARCHAR(10)	Нет	Любое	–	–
lifting_capacity	Грузоподъемность	INT	Нет	> 0	–	–
body_size	Размер кузова	VARCHAR(50)	Да	Любое	–	–
body_volume	Объем кузова	SMALLINT	Да	> 0	–	–
fuel_consumption	Расход топлива	TINYINT	Нет	> 0	–	–

Таким образом, реализованная физическая схема базы данных позволяет эффективно хранить сведения о произведенных расчетах для логистических грузоперевозок.

3.2 Реализация алгоритмической части

Алгоритмическая часть приложения представляет собой модуль расчета на основе муравьиного алгоритма. Особенностью реализации работы муравьиного алгоритма в разрабатываемом приложении состоит в том, чтобы алгоритм был независимым и масштабируемым элементом системы, поскольку логистические грузоперевозки не ограничиваются однотипными задачами.

Основой алгоритма является многократное выполнение обхода точек маршрута с промежуточным обновлением инструментов выбора маршрута. Данный принцип выполняется итеративно одновременно несколькими муравьями. Для муравья, находящегося в городе i , переход в город j зависит от нескольких факторов: виртуальный след феромона, маршрут и видимость.

Основной составляющей алгоритма является нахождение вероятности осуществления перехода муравья k из города i в город j на итерации t :

$$\begin{cases} P_{ij,k}(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in J_{i,k}} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta}, \text{ если } j \in J_{i,k}, \\ P_{ij,k}(t) = 0, \text{ если } j \notin J_{i,k} \end{cases}$$

где α и β – настроечные значения, устанавливающие вес оставляемого следа муравья для дальнейшего выбора маршрута;

$J_{i,k}$ – число еще не посещенных вершин;

η_{ij} – фактор видимости, выражающий коэффициент перехода муравья из города i в город j ;

$\tau_{ij}(t)$ – виртуальный след феромона между городами.

Количество откладываемого феромона на ребре (i, j) рассчитывается следующим образом:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \frac{Q}{L_k(t)}, & \text{если } (i, j) \in T_k(t), \\ 0, & \text{если } (i, j) \notin T_k(t) \end{cases},$$

где $T_k(t)$ – маршрут, пройденный муравьем k на итерации t ;

$L_k(t)$ – общая длина пройденного маршрута;

Q – настроечный параметр того же порядка, что и оптимальная длина маршрута.

Последним элементом нахождения оптимального маршрута является испарение феромона, позволяющее со временем исключать большие переходы с помощью вероятностного механизма выбора. Феромон испаряется согласно коэффициенту испарения $p \in [0,1]$ по следующему правилу:

$$\tau_{ij}(t+1) = (1-p) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t),$$

где

$$\Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij,k}(t),$$

а m – число особей в популяции.

В ходе выполнения алгоритма количество муравьев не изменяется. Чрезмерное число муравьев в колонии приведет к снижению точности поиска из-за усиления субоптимальных маршрутов, а нехватка муравьев приведет к быстрому испарению феромонов. По этой причине число виртуальных муравьев принимается равным количеству вершин [10].

Для наглядной демонстрации работы алгоритма, была построена диаграмма деятельности, отражающая поэтапное выполнения оптимизации маршрутов. За основу для данной диаграммы был взят базовый муравьиный

алгоритм без дополнительных улучшений. Данная диаграмма деятельности представлена на рисунке 24.

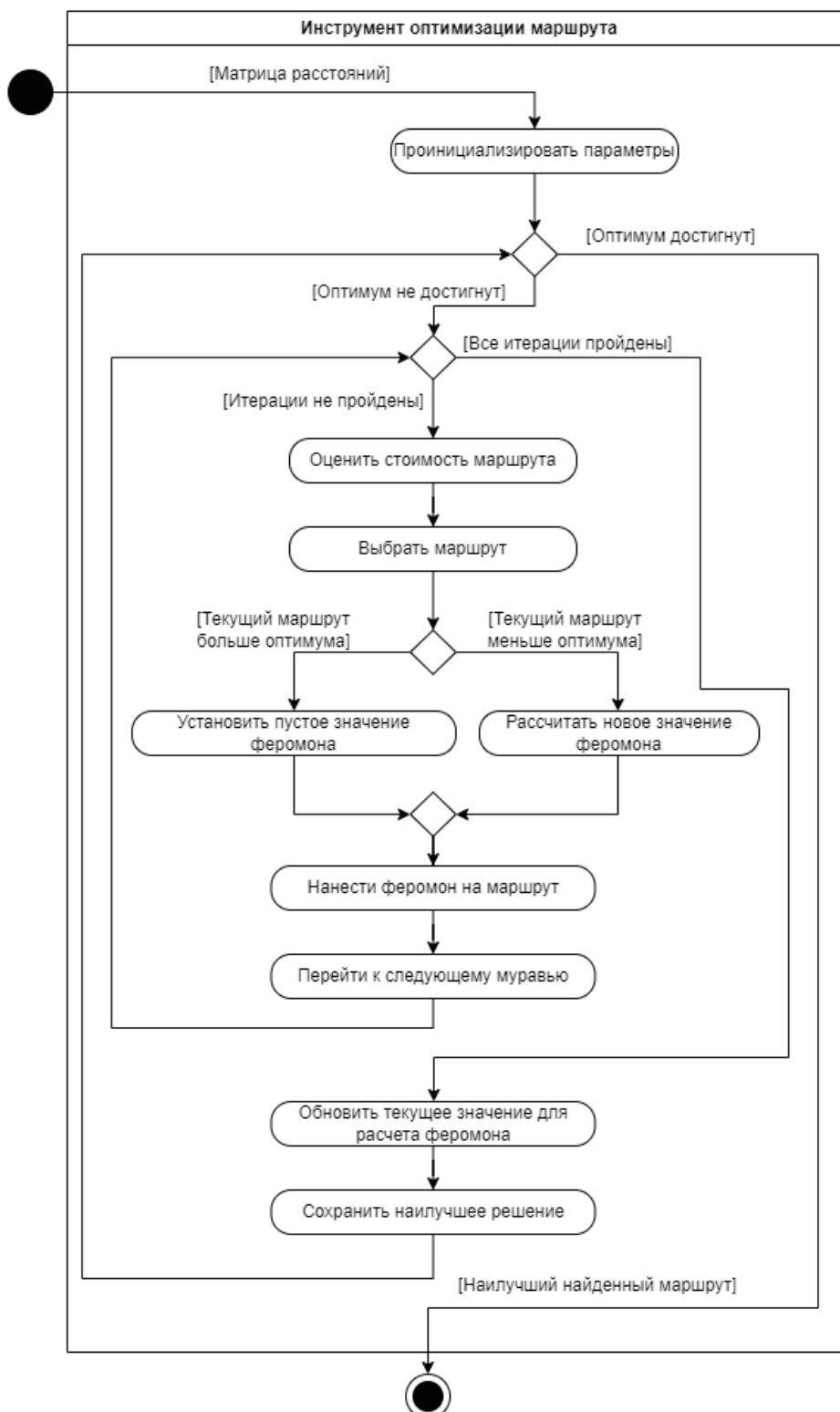


Рисунок 24 – Диаграмма деятельности муравьиного алгоритма

Данная диаграмма отражает общий процесс оптимизации маршрута по заданной матрице расстояний. В качестве демонстрации работы конечного

алгоритма, на рисунке 25 представлена карта города Пенза с установленными точками маршрута.

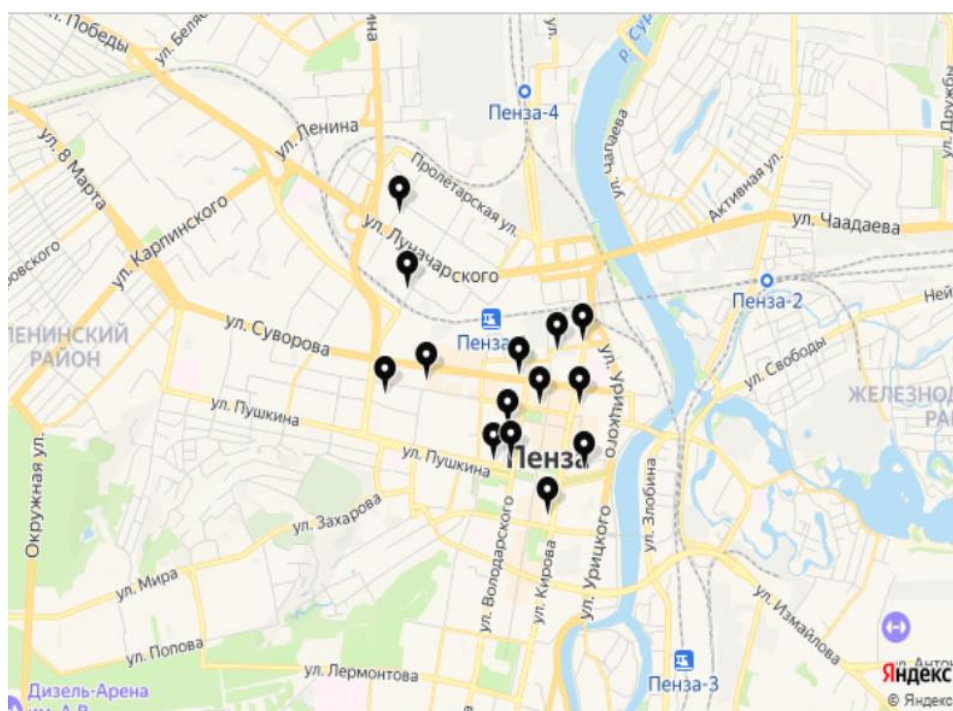


Рисунок 25 – Карта с установленными точками маршрута

На таблице 10 представлена матрица расстояний между выбранными точками, где значения расстояний указаны в километрах.

Таблица 10 – Матрица расстояний между точками маршрута

Порядковый номер точки	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	2.27	1.66	2.7	3.2	1.56	0.88	1.3	1.96	2.14	1.46	1.17	1.47	1.21
2	2.27	0	1.41	2.97	3.5	1.83	1.14	1.29	1.85	2.17	1.45	1.21	1.24	0.97
3	1.66	1.41	0	2.16	2.64	2.55	1.86	2.04	2.73	2.89	2.17	1.93	1.66	1.69
4	2.7	2.97	2.16	0	0.77	1.99	2.1	2.88	3.1	3.1	2.59	2.48	3	2.8
5	3.2	3.5	2.64	0.77	0	2.23	2.34	3.1	3.3	3.3	2.83	2.72	3.3	3
6	1.56	1.83	2.55	1.99	2.23	0	0.35	1.13	1.54	1.77	0.82	0.73	1.29	1.05
7	0.88	1.14	1.86	2.1	2.34	0.35	0	0.86	1.26	1.42	0.54	0.46	1.02	0.78
8	1.3	1.29	2.04	2.88	3.1	1.13	0.86	0	1.67	1.73	1.2	1.05	0.17	0.44
9	1.96	1.85	2.73	3.1	3.3	1.54	1.26	1.67	0	1.02	1.04	1.26	0.9	1.16
10	2.14	2.17	2.89	3.1	3.3	1.77	1.42	1.73	1.02	0	0.86	1.07	1.39	1.34
11	1.46	1.45	2.17	2.59	2.83	0.82	0.54	1.2	1.04	0.86	0	0.65	1.16	0.92
12	1.17	1.21	1.93	2.48	2.72	0.73	0.46	1.05	1.26	1.07	0.65	0	0.85	0.61
13	1.47	1.24	1.66	3	3.3	1.29	1.02	0.17	0.9	1.39	1.16	0.85	0	0.35
14	1.21	0.97	1.69	2.8	3	1.05	0.78	0.44	1.16	1.34	0.92	0.61	0.35	0

Для заданной матрицы расстояний была произведена серия расчетов оптимальных маршрутов. Число итераций составляет 500 проходов по

маршрутам, численность колонии – 100 муравьев. Результаты для 10 расчетов представлены на таблице 11.

Таблица 11 – Матрица расстояний между точками маршрута

№	Порядок посещения точек маршрута	Эффективная длина маршрута
1	5-4-6-7-11-10-9-13-8-14-12-2-3-1	11,93 км
2	5-4-3-2-14-13-8-1-12-6-7-11-10-9	11,79 км
3	5-4-3-2-14-13-8-1-7-6-12-11-10-9	11,62 км
4	5-4-6-7-11-12-10-9-14-13-8-2-3-1	12,43 км
5	5-9-10-11-6-7-1-8-13-14-12-2-3-4	14,44 км
6	5-4-1-7-12-14-13-8-6-11-10-9-2-3	13,03 км
7	5-4-3-2-14-8-13-9-10-12-11-6-7-1	11,61 км
8	5-3-2-14-8-13-9-10-11-7-6-12-1-4	13,89 км
9	5-4-3-2-9-10-11-6-7-12-14-13-8-1	12,12 км
10	5-4-3-1-7-6-11-10-9-13-8-14-12-2	11,84 км

По результатам расчета для заданной матрицы расстояний, наилучшим оказался маршрут под номером 7. Данный маршрут на карте приведен на рисунке 26.

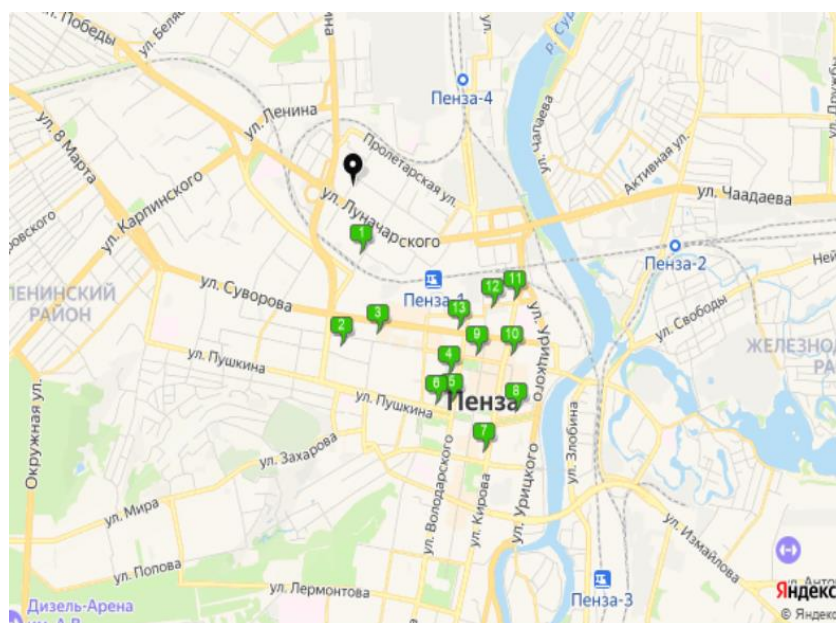


Рисунок 26 – Карта с установленными точками маршрута

Данный рисунок наглядно отражает основную суть применения алгоритма для решения задач автоматизации транспортной логистики грузоперевозок. По результатам реализации заданных формул был создан модуль расчета оптимального маршрута на основе работы муравьиного алгоритма, вычисляющий наилучший путь посещения точек доставки груза. Данный модуль

был реализован в качестве подключаемой библиотеки в среде разработки MS Visual Studio 2022 на языке C#.

3.3 Реализация основного приложения

Основное приложение было реализовано в среде разработки MS Visual Studio 2022 в качестве исполняемого проекта WPF. Windows Presentation Foundation (WPF) – аналог технологии WinForms, система для построения клиентских приложений Windows с визуально улучшенными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework, использующая язык XAML [11]. Использование данной технологии позволяет беспрепятственно воспроизвести необходимые графические элементы на языке разметки XAML для эффективного вывода информации в комфортном для пользователя формате. Для наглядного отражения взаимодействия модулей приложения была разработана диаграмма компонентов, приведенная на рисунке 27.

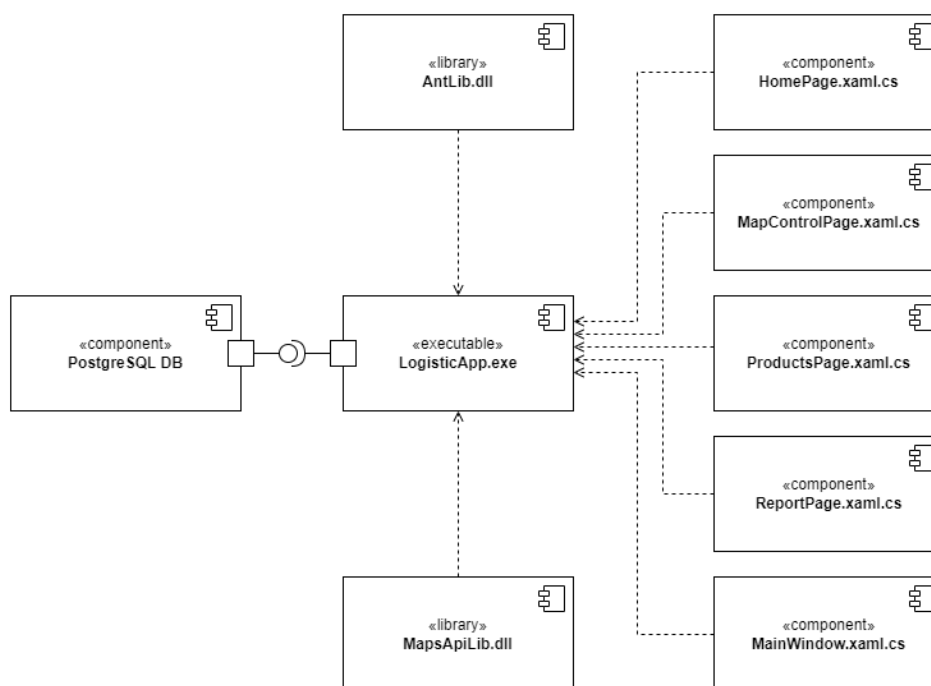


Рисунок 27 – Диаграмма компонентов разворачиваемого приложения

Основная работа приложения сосредоточена в компонентах, представляющих собой совокупность кода и интерфейсной разметки. С

помощью использования библиотек AntLib.dll и MapsApiLib.dll приложение производит расчет и запрос данных, для дальнейшего вывода на страницы пользователю. Результат работы приложения записываются в базе данных с использованием СУБД PostgreSQL.

Для демонстрации разворачивания используемых компонентов в приложении, была разработана диаграмма развертывания, приведенная на рисунке 28.

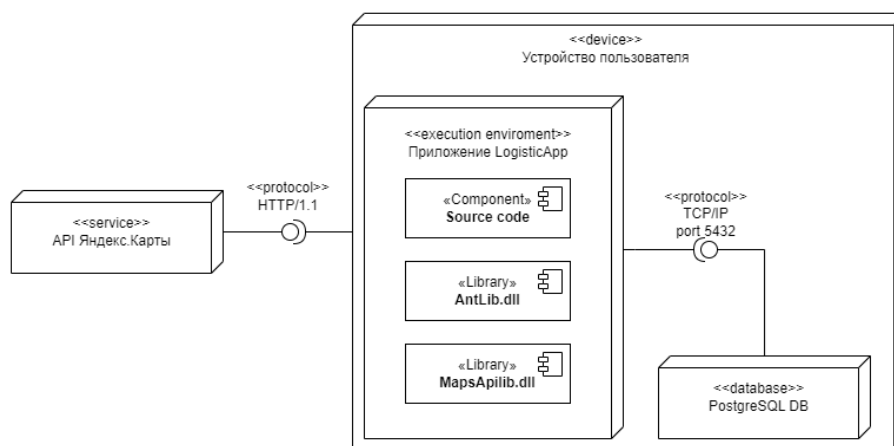


Рисунок 28 – Диаграмма развертывания приложения

Данная диаграмма отражает развертывание элементов разрабатываемого приложения в пространстве. Разработанное приложение может быть развернуто на персональных компьютерах с операционной системой семейства Windows версии 10 и выше. Разворачиваемая база данных с использованием СУБД PostgreSQL позволяет хранить данные результатов работы приложения и может располагаться как на устройстве пользователя, так и на внешнем вычислительном устройстве, так как взаимодействие с ней осуществляется по сетевому протоколу. Для эффективной работы основного приложения необходим сетевой доступ к API Яндекс.Карт, что позволяет получать актуальные сведения о геолокационных позициях точек маршрута. Совокупность разворачиваемых программных средств позволяет с высокой точностью производить расчет оптимальных маршрутов для решения задач логистических грузоперевозок. Код разработанных программных средств приведен в приложении Б.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Существует достаточно много видов тестирования программных продуктов [12]. Наиболее популярны следующие виды тестирования:

- функциональное тестирование;
- модульное тестирование;
- интеграционное тестирование.

Функциональное тестирование сосредотачивается на проверке соответствия бизнес-требованиям приложения путем проверки результатов действий без учета промежуточных состояний системы.

Юнит тестирование или модульное тестирование работает на очень низком уровне, близко к исходному коду приложения. Способ заключается в тестировании отдельных методов и функций классов, компонентов или модулей, используемых в программном обеспечении. Модульные тесты могут выполняться очень быстро, потому что не требуют больших расходов на автоматизацию [13].

Для полного охвата взаимодействия пользователя с приложением необходимо написать тест-кейсы с детальным описанием действий для проверки конкретных функций, основанные на пользовательских сценариях.

Интеграционное тестирование проверяет взаимодействие различных модулей и сервисов приложения, требуя запуска различных компонентов приложения для проведения тестов.

Решение о проведении функционального тестирования программы было принято из-за основного уклона требований к программе на функциональные аспекты, которые необходимо проверить.

4.2 Тестирование алгоритмической части приложения

В результате разработки программы было проведено модульное тестирование. Unit-тесты были разработаны с помощью встроенных средств

среды разработки MS Visual Studio. Исходный код модульных тестов приведен в приложении В. План тестирования приведен в таблице 12.

Таблица 12 – План модульного тестирования

Тест	Описание	Ожидаемый результат	Актуальный результат
NormalMapCityCount_Test	Проверка общей работоспособности алгоритма для реальных расстояний между точками маршрута, который необходимо оптимизировать	Пройден	Пройден
NormalMapBestTrail_Test	Проверка общей работоспособности алгоритма для расчета длины маршрута для реальных расстояний между точками маршрута, который необходимо оптимизировать	Пройден	Пройден
FailedParams_Test	Проверка работоспособности алгоритма при некорректных входных параметрах	Пройден	Пройден
FailedCityParam_Test	Проверка работоспособности алгоритма при некорректном количестве точек маршрута	Пройден	Пройден
FailedDistanceParam_Test	Проверка работоспособности алгоритма при некорректно заданных расстояниях между точками	Пройден	Пройден
AntAccuracy_Test	Проверка точности расчета алгоритма при заранее известном оптимальном маршруте	Пройден	Пройден

Результаты модульного тестирования приведены на рисунке 29. По результатам можно сделать вывод, что все тесты пройдены успешно и алгоритм работает корректно.

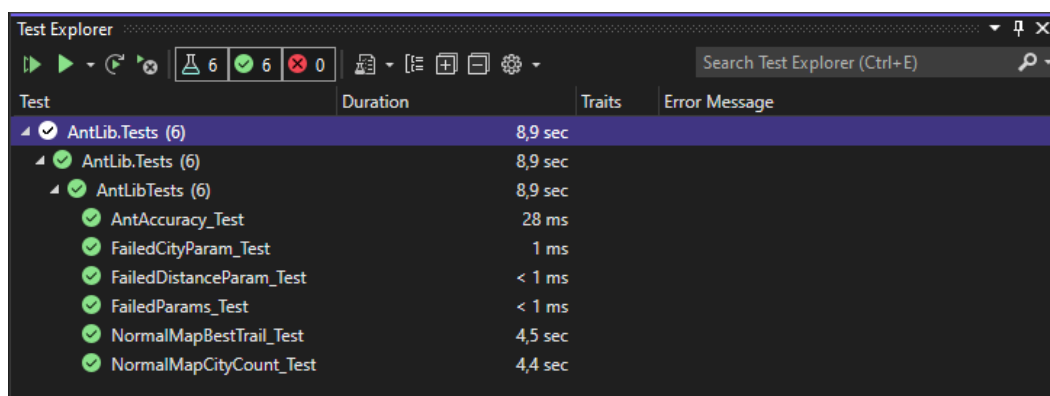


Рисунок 29 – Результат выполнения модульных тестов в среде разработки MS Visual Studio 2022

В ходе тестирования были выявлены и исправлены ошибки в коде алгоритмической части приложения.

4.3 Тестирование пользовательской части приложения

Для проведения функционального тестирования пользовательской части был выбран метод тестирования черным ящиком. Пользовательская часть приложения в рамках основного функционала представлена вводом адресов географических точек и их отображения на карте. Для ввода адреса имеются определенные требования, которым соответствуют выделенные классы эквивалентности в таблице 13.

Таблица 13 – Классы эквивалентности

Требование	Правильные КЭ	Неправильные КЭ
Длина адреса	10–100 (1)	0–9 (2) 101–∞ (3)
Структура	Начинается с буквы (4) Блоки разделены запятыми (5) Точки используются для сокращения (6) Дефисы соединяют буквы и цифры, или буквы и буквы (7)	Не начинается с буквы (8) Блоки не разделены запятыми (9) Точки используются не для сокращения (10) Дефисы соединяют не буквы и цифры, или буквы и буквы (11)
Содержание	Кириллические буквы (12) Цифры (13) Запяты (14) Точки (15) Дефисы (16)	Остальное (17)

Согласно выделенным классам эквивалентности были составлены тест-кейсы, приведенные в таблице 14. Результаты приведены в приложении Г.

Таблица 14 – Тест-кейсы для выделенных классов эквивалентности

№	КЭ	Входные данные	Выходные данные
1	1,4,5,6,7,12, 13,14,15,16	г. Санкт-Петербург, ул. Пушкина, д. 90	Координаты
2	1,4,5,6,11,12, 13,14,15,16	г. – Пенза, ул.–Пушкина,д.3	Ошибка: Неправильно поставленные дефисы (рисунок Г.1)
3	1,5,6,8,12,13,14,15	6, ул. Пушкина	Ошибка: Начинается с цифры (рисунок Г.2)
4	1,4,5,6,7,12, 13,14,15,17	г. Волгоград, ул. *?№», д. 3	Ошибка: Содержит недопустимые символы (рисунок Г.3)

Продолжение таблицы 14

№	КЭ	Входные данные	Выходные данные
5	3,4,12	aaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaa	Ошибка: Больше 100 символов
6	1,4,5,10,12, 13,14,15	г. Москва., ул. Т.олстого, д. 4	Ошибка: Точка используется не для сокращения (рисунок Г.4)
7	1,4,6,9,12,13,15	г. Москва ул. Толстого д. 4	Ошибка: Блоки не разделены запятыми
8	2,4,6,12,15	г. Реж	Ошибка: Меньше 10 символов

В дополнение к выделенным класса эквивалентности были составлены граничные условия к вводимым данным. Граничные условия к вводу адреса представлены в таблице 15.

Таблица 15 – Граничные условия

Границы	Содержание
Физические	Строка нулевой длины (Ф)
Логические	Указано несколько улиц (Л1) Указано несколько домов (Л2) Указано несколько городов (Л3) Не указан город (Л4) Не указана улица (Л5) Номер дома не является целым положительным числом (Л6)
Произвольные	Нарушена последовательность составления адреса (П)

Для выделенных граничных условий были составлены дополнительные тест-кейсы, приведенные в таблице 16.

Таблица 16 – Тест-кейсы для выделенных граничных условий

№	Границы	Входные данные	Выходные данные
1	Ф	{Пустая строка}	–
2	Л1	г. Пенза, ул. Пушкина, ул. Суворова, д. 3	Ошибка: Указано несколько улиц (рисунок Г.5)
3	Л2	г. Пенза, ул. Пушкина, д. 3, д. 4, д. 5	Ошибка: Указано несколько домов (рисунок Г.6)
4	Л3	г. Пенза, г. Москва, ул. Пушкина, д. 3	Ошибка: Указано несколько городов (рисунок Г.7)

Продолжение таблицы 16

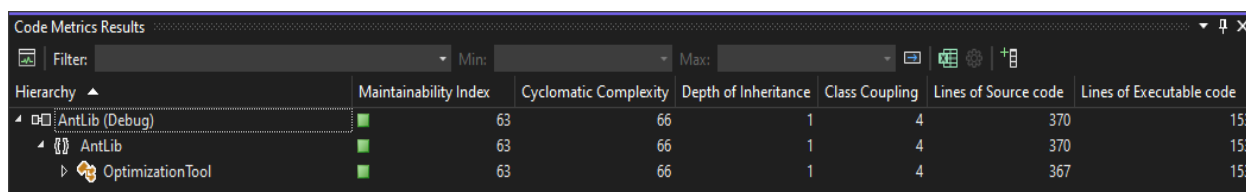
№	Границы	Входные данные	Выходные данные
5	Л4	ул. Пушкина, д. 3	Ошибка: Не указан город
6	Л5	г. Пенза, д. 3	Ошибка: Не указана улица (рисунок Г.8)
7	Л6	г. Пенза, ул. Пушкина, д. 00	Ошибка: Неверно указан номер дома (рисунок Г.9)
8	П	д.3, ул. Пушкина, г. Пенза	Ошибка: Нарушена последовательность адреса (рисунок Г.10)

Полученные результаты соответствуют ожидаемым, а значит, пользовательская часть приложения работает корректно.

4.4 Метрики кода

Метрика программного обеспечения – мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций [14].

Был проведен анализ качества кода реализованного алгоритма вычисления оптимального маршрута для заданных точек. С помощью встроенных средств среды разработки MS Visual Studio были получены результаты вычисления индекса доступности, цикломатической сложности, глубины наследования и общих характеристик программного кода. Результаты вычислений приведены на рисунке 30.



Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
AntLib (Debug)	63	66	1	4	370	153
AntLib	63	66	1	4	370	153
OptimizationTool	63	66	1	4	367	153

Рисунок 30 – Результаты вычисления индекса доступности и других характеристик программного кода

На рисунке 30 представлены следующие результаты метрик кода, вычисляемые в Microsoft Visual Studio:

– Maintainability Index (индекс доступности) – вычисляет значение индекса от 0 до 100, представляющее относительную простоту поддержания

кода. Высокое значение означает лучшую поддержку. Цветные закодированные оценки можно использовать для быстрого выявления проблемных мест в коде. Зеленый рейтинг составляет от 20 до 100 и указывает, что код имеет хорошую поддержку;

- Cyclomatic Complexity (цикломатическая сложность) – измеряет структурную сложность кода. Он создается путем вычисления количества различных путей кода в потоке программы. Программа, которая имеет сложный поток управления, требует больше тестов для достижения хорошего покрытия кода и менее поддерживается;

- Depth of Inheritance (глубина наследования) – указывает количество различных классов, наследуемых друг от друга, вплоть до базового класса. Для глубины наследования хорошо низкое значение;

- Class Coupling (объединение классов) – измеряет связь с уникальными классами с помощью параметров, локальных переменных, возвращаемых типов, вызовов методов, универсальных или шаблонных экземпляров, базовых классов, реализаций интерфейса, полей, определенных во внешних типах и оформлении атрибутов;

- Lines of Source code (строки исходного кода) – указывает точное количество строк исходного кода, присутствующих в исходном файле, включая пустые строки;

- Lines of Executable code (строки исполняемого кода) – указывает приблизительное количество строк или операций исполняемого кода. Это количество операций в исполняемом коде [15].

В данном случае все вычисленные значения находятся в приемлемом диапазоне, что означает, что код относительно удобен в поддержке.

Заключение

Результатом выполнения выпускной квалификационной работы стало приложение, позволяющее эффективно планировать и находить оптимальные решения для логистических задач. Была проведена оценка финансовых и временных затрат на разработку.

На стадии анализа требований проведен анализ предметной области транспортной логистики, сравнительный анализ программного обеспечения аналогичного назначения, а также выявлены функциональные и нефункциональные требования к разрабатываемым средствам.

Для проекта осуществлено структурное и календарное планирование, а также выполнена оценка стоимости разработки приложения. По итогу, на выполнение всех этап разработки проекта необходим 41 день и 65 680 рублей.

На стадии проектирования была разработана концептуальная, логическая и физическая модели базы данных, а также спроектирована архитектура программного продукта. Реализованные программные средства функционально соответствуют поставленным задачам и предоставляют возможности для хранения сведений о выполненных расчетах. Созданное приложение разворачивается непосредственно на персональном компьютере пользователя, что делает его легкодоступным и недорогим решением для оптимизации процессов транспортной логистики. Работоспособность приложения проверена несколькими видами тестирования, включая модульное и функциональное тестирования. Все тесты были успешно пройдены.

Таким образом, задача, поставленная по реализации программных средств автоматизации транспортной логистики грузоперевозок, выполнена в полном объеме.

Список использованных источников

1. Клименков А. Как муравьи решают проблемы коммивояжёров // Тематический блог «Tproger» [Электронный ресурс]. URL: <https://tproger.ru/articles/kak-muravi-reshajut-problemy-kommivojashjorov> (дата обращения 03.06.2024).
2. Ваулин А.Е. Метод ветвей и границ. Задача коммивояжера // Тематический блог «Хабр» [Электронный ресурс]. URL: <https://habr.com/ru/articles/560468/> (дата обращения 03.06.2024).
3. Акмашев, С.В. Сравнительный анализ стохастических и градиентных подходов к решению задачи оптимизации маршрутов транспортной логистики / С.В. Акмашев, И.Ю. Балашова // Новые информационные технологии и системы: материалы XX международной науч.-техн. конф. – Пенза: Изд-во Пензенского государственного университета, 2023. – С. 87 – 91.
4. Сырцев В. Использование диаграммы вариантов использования UML при проектировании программного обеспечения // Тематический блог «Хабр» [Электронный ресурс]. URL: <https://habr.com/ru/articles/566218/> (дата обращения 03.06.2024).
5. Концептуальная модель базы данных // Пензенский Государственный Университет Архитектуры и Строительства. [Электронный ресурс]. URL: <https://studfile.net/preview/9959196/page:2/> (дата обращения 03.06.2024).
6. Верников Г. Основы методологии IDEF1X // Проект «Корпоративный менеджмент» [Электронный ресурс]. URL: <https://www.cfin.ru/vernikov/idef/idef1x.shtml/> (дата обращения 03.06.2024).
7. Рудь В. Концептуальное логическое и физическое моделирование данных // Информационный портал компании «Марк Аврелий» [Электронный ресурс]. URL: https://marcus-aurelius.ru/articles/Data_modeling.html (дата обращения 03.06.2024).

8. Васильева А. Нотации баз данных // Архив «О Visio по-русски» за 2011 – 2018 [Электронный ресурс]. URL: <https://rus-visio-blog-archive.github.io/2012/12/17/487.htm> (дата обращения 03.06.2024).
9. Карпова Т. Физические модели баз данных // НОУ «ИНТУИТ», курс лекций [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/1001/297/lecture/7415> (дата обращения 03.06.2024).
10. Штовба С.Д. Муравьиные алгоритмы // Портал научных исследований «ResearchGate» [Электронный ресурс]. URL: https://www.researchgate.net/publication/279535061_Stovba_SD_Muravinye_algoritmy_Exponenta_Pro_Matematika_v_prilozeniah_-_2003_-_No4_-_S_70-75 (дата обращения 03.06.2024).
11. Общие сведения о WPF // Официальная документация Microsoft [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8> (дата обращения 03.06.2024).
12. Куликов, С.С. Тестирование программного обеспечения. Базовый курс / С.С. Куликов – Минск.: EPAM Systems, 2021. – 300 с.
13. Академия Microsoft. Верификация программного обеспечения // НОУ «ИНТУИТ» [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/1040/209/lecture/5418/> (дата обращения 03.06.2024).
14. Рыжков Е. Программный код и его метрики // Тематический блог «Хабр». [Электронный ресурс]. URL: <https://habr.com/ru/companies/intel/articles/106082/> (дата обращения 03.06.2024).
15. Значения метрик кода // Официальная документация Microsoft [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-values?view=vs-2022> (дата обращения 03.06.2024).

Приложение А
(справочное)
Глоссарий

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

МА (Муравьиный алгоритм) – один из эффективных полиномиальных алгоритмов для нахождения приближенных решений задачи коммивояжера, а также решения аналогичных задач поиска маршрутов на графах.

МВГ (Метод ветвей и границ) – общий алгоритмический метод для нахождения оптимальных решений различных задач оптимизации, особенно дискретной и комбинаторной оптимизации.

ГА (Генетический алгоритм) – эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путем случайного подбора, комбинирования и вариации искоемых параметров с использованием механизмов, аналогичных естественному отбору в природе.

ЖА (Жадный алгоритм) – алгоритм, в котором на каждом шаге принимается решение, оцениваемое как наилучшее в данный момент, то есть локально оптимальное.

Crow`s foot (Нотация «Гусиные лапки») – широко используемый метод визуализации и проектирования схем баз данных. Он предоставляет представления моделей отношений между сущностями, которые важны для понимания структуры и взаимоотношений данных в системе баз данных.

БД (База данных) – это имеющая название совокупность данных, которая отражает состояние объектов и их отношений в рассматриваемой предметной области.

СУБД (Система управления базами данных) – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Приложение Б
(рекомендуемое)
Листинг приложения

Листинг Б.1 – Файл OptimizationTool.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AntLib
{
    public class OptimizationTool
    {
        private static Random random = new Random(0);

        private int _alphaParam = 3;

        private int _betaParam = 2;

        private double _pDecFactor = 0.01;
        private double pIncFactor = 2.0;
        public int AntsCount { get; private set; }
        public int MaxIterations { get; private set; }
        public void InitializeParams(int _alpha, int _beta,
            double _pheromoneDecFactor, double _pheromoneIncFactor,
            int _antsCount,
            int _maxTime)
        {
            _alphaParam = _alpha;
            _betaParam = _beta;
            _pDecFactor = _pheromoneDecFactor;
            pIncFactor = _pheromoneIncFactor;
            AntsCount = _antsCount;
            MaxIterations = _maxTime;
        }
        public int[] GetBestSolution(double[][] dMatrix, int cityCount,
            out double effectiveLength)
        {
            int[][] colony = InitAnts(AntsCount, cityCount);

            int[] bestWay = BestTrail(colony, dMatrix);

            effectiveLength = Length(bestWay, dMatrix);

            double[][] pheromones = InitPheromones(cityCount);

            int time = 0;
            while (time < MaxIterations)
            {
                UpdateAnts(colony, pheromones, dMatrix);
                UpdatePheromones(pheromones, colony, dMatrix);

                int[] tempBT = BestTrail(colony, dMatrix);
                double tempEL = Length(tempBT, dMatrix);
                if (tempEL < effectiveLength)
                {
                    effectiveLength = tempEL;
                    bestWay = tempBT;
                }
            }
        }
    }
}

```

```

        }
        time += 1;
    }

    return bestWay;
}

private int[][] InitAnts(int antsCount, int cityCount)
{
    int[][] colony = new int[antsCount][];
    for (int k = 0; k <= antsCount - 1; k++)
    {
        int startPos = random.Next(0, cityCount);
        colony[k] = RandomTrail(startPos, cityCount);
    }
    return colony;
}

private int[] RandomTrail(int start, int cityCount)
{
    int[] trail = new int[cityCount];

    for (int i = 0; i <= cityCount - 1; i++)
    {
        trail[i] = i;
    }

    for (int i = 0; i <= cityCount - 1; i++)
    {
        int r = random.Next(i, cityCount);
        int tmp = trail[r];
        trail[r] = trail[i];
        trail[i] = tmp;
    }

    int target = IndexOfTarget(trail, start);
    int temp = trail[0];
    trail[0] = trail[target];
    trail[target] = temp;

    return trail;
}

private int IndexOfTarget(int[] trail, int target)
{
    for (int i = 0; i <= trail.Length - 1; i++)
    {
        if (trail[i] == target)
        {
            return i;
        }
    }
    throw new Exception("Target not found in IndexOfTarget");
}

private double Length(int[] trail, double[][] dists)
{
    double result = 0.0;

```

```

        for (int i = 0; i <= trail.Length - 2; i++)
        {
            result += Distance(trail[i], trail[i + 1], dists);
        }
        return result;
    }

    private int[] BestTrail(int[][] colony, double[][] dists)
    {
        double effectiveLength = Length(colony[0], dists);
        int indexBT = 0;
        for (int k = 1; k <= colony.Length - 1; k++)
        {
            double len = Length(colony[k], dists);
            if (len < effectiveLength)
            {
                effectiveLength = len;
                indexBT = k;
            }
        }
        int numCities = colony[0].Length;
        int[] bestOptimizedTrail = new int[numCities];
        colony[indexBT].CopyTo(bestOptimizedTrail, 0);
        return bestOptimizedTrail;
    }

    private double[][] InitPheromones(int cityCount)
    {
        double[][] phs = new double[cityCount][];
        for (int i = 0; i <= cityCount - 1; i++)
        {
            phs[i] = new double[cityCount];
        }
        for (int i = 0; i <= phs.Length - 1; i++)
        {
            for (int j = 0; j <= phs[i].Length - 1; j++)
            {
                phs[i][j] = 0.01;
            }
        }
        return phs;
    }

    private void UpdateAnts(int[][] ants, double[][] pheromones,
double[][] dists)
    {
        int numCities = pheromones.Length;
        for (int k = 0; k <= ants.Length - 1; k++)
        {
            int start = random.Next(0, numCities);
            int[] newTrail = BuildTrail(k, start, pheromones, dists);
            ants[k] = newTrail;
        }
    }

    private int[] BuildTrail(int k, int start, double[][] pheromones,
double[][] dists)

```

```

    {
        int numCities = pheromones.Length;
        int[] trail = new int[numCities];
        bool[] visited = new bool[numCities];
        trail[0] = start;
        visited[start] = true;
        for (int i = 0; i <= numCities - 2; i++)
        {
            int cityX = trail[i];
            int next = NextCity(k, cityX, visited, pheromones,
dists);
            trail[i + 1] = next;
            visited[next] = true;
        }
        return trail;
    }

    private int NextCity(int k, int cityX, bool[] visited, double[][]
pheromones, double[][] dists)
    {
        double[] probabilityWays = MoveProbs(k, cityX, visited,
pheromones, dists);

        double[] cumul = new double[probabilityWays.Length + 1];
        for (int i = 0; i <= probabilityWays.Length - 1; i++)
        {
            cumul[i + 1] = cumul[i] + probabilityWays[i];
        }

        double p = random.NextDouble();

        for (int i = 0; i <= cumul.Length - 2; i++)
        {
            if (p >= cumul[i] && p < cumul[i + 1])
            {
                return i;
            }
        }
        throw new Exception("Invalid city error");
    }

    private double[] MoveProbs(int k, int cityX, bool[] visited,
double[][] pheromones, double[][] dists)
    {
        int cityCount = pheromones.Length;
        double[] tauetaParam = new double[cityCount];

        double sum = 0.0;

        for (int i = 0; i <= tauetaParam.Length - 1; i++)
        {
            if (i == cityX)
            {
                tauetaParam[i] = 0.0;
            }
            else if (visited[i] == true)

```

```

        {
            tauetaParam[i] = 0.0;
        }
        else
        {
            tauetaParam[i] = Math.Pow(pheromones[cityX][i],
            _alphaParam) * Math.Pow((1.0 / Distance(cityX, i, dists)), _betaParam);

            if (tauetaParam[i] < 0.0001)
            {
                tauetaParam[i] = 0.0001;
            }
            else if (tauetaParam[i] > (double.MaxValue /
            (cityCount * 100)))
            {
                tauetaParam[i] = double.MaxValue / (cityCount *
            100);
            }
            sum += tauetaParam[i];
        }

        double[] possibleRoutes = new double[cityCount];
        for (int i = 0; i <= possibleRoutes.Length - 1; i++)
        {
            possibleRoutes[i] = tauetaParam[i] / sum;
        }
        return possibleRoutes;
    }

    private void UpdatePheromones(double[][] phs, int[][] ants,
    double[][] dists)
    {
        for (int i = 0; i <= phs.Length - 1; i++)
        {
            for (int j = i + 1; j <= phs[i].Length - 1; j++)
            {
                for (int k = 0; k <= ants.Length - 1; k++)
                {
                    double length = Length(ants[k], dists);
                    double decreaseFactor = (1.0 - _pDecFactor) *

                    phs[i][j];

                    double increaseFactor = 0.0;
                    if (EdgeInTrail(i, j, ants[k]) == true)
                    {
                        increaseFactor = (pIncFactor / length);
                    }

                    phs[i][j] = decreaseFactor + increaseFactor;

                    if (phs[i][j] < 0.0001)
                    {
                        phs[i][j] = 0.0001;
                    }
                    else if (phs[i][j] > 100000.0)
                    {
                        phs[i][j] = 100000.0;
                    }
                }
            }
        }
    }

```

```

        }

        phs[j][i] = phs[i][j];
    }
}

}

private bool EdgeInTrail(int firstCity, int secondCity, int[]
route)
{
    int lastIndex = route.Length - 1;
    int target = IndexOfTarget(route, firstCity);

    if (target == 0 && route[1] == secondCity)
    {
        return true;
    }
    else if (target == 0 && route[lastIndex] == secondCity)
    {
        return true;
    }
    else if (target == 0)
    {
        return false;
    }
    else if (target == lastIndex && route[lastIndex - 1] ==
secondCity)
    {
        return true;
    }
    else if (target == lastIndex && route[0] == secondCity)
    {
        return true;
    }
    else if (target == lastIndex)
    {
        return false;
    }
    else if (route[target - 1] == secondCity)
    {
        return true;
    }
    else if (route[target + 1] == secondCity)
    {
        return true;
    }
    else
    {
        return false;
    }
}

private double Distance(int firstCity, int secondCity, double[][]
dists)
{
    return dists[firstCity][secondCity];
}

```

```

    }
}

```

Листинг Б.2 – Файл IRepository.cs

```

using LogisticApp.DataAccess.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess.Abstractions
{
    public interface IRepository <T> where T : class
    {
        public long Add(T item);
        public int Update(T item);
        public int Delete(long id);
        public IEnumerable<T> GetAll();
        public T GetById(long id);
    }
}

```

Листинг Б.3 – Файл Driver.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess.Models
{
    public class Driver
    {
        public long Id { get; set; }
        public string LastName { get; set; }
        public string FirstName { get; set; }
        public string? MiddleName { get; set; }
        public string CertificateNumber { get; set; }
        public string? PhoneNumber { get; set; }
    }
}

```

Листинг Б.4 – Файл Vehicle.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess.Models
{
    public class Vehicle
    {
        public long Id { get; set; }
    }
}

```

```

        public string VehicleType { get; set; }
        public string Model { get; set; }
        public string VehicleNumber { get; set; }
        public int LiftingCapacity { get; set; }
        public string BodySize { get; set; }
        public int BodyVolume { get; set; }
        public int FuelConsumption { get; set; }
    }
}

```

Листинг Б.5 – Файл Waybill.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess.Models
{
    public class Waybill
    {
        public long Id { get; set; }
        public string RouteType { get; set; }
        public int RouteVolume { get; set; }
        public int EffectiveLength { get; set; }
        public string? Route { get; set; }
        public Driver Driver { get; set; }
        public Vehicle Vehicle { get; set; }
        public int FuelResource { get; set; }
        public string? Place { get; set; }
    }
}

```

Листинг Б.6 – DriverRepository.cs

```

using LogisticApp.DataAccess.Abstractions;
using LogisticApp.DataAccess.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess.Repositories
{
    public class DriversRepository: IRepository<Driver>
    {
        private ApplicationContext _dataContext;
        public DriversRepository(ApplicationContext context)
        {
            this._dataContext = context;
        }
        public long Add(Driver item)
        {
            var result = _dataContext.Drivers.Add(item);
            _dataContext.SaveChanges();
        }
    }
}

```

```

        return result.Entity.Id;
    }

    public int Delete(long id)
    {
        var result = _dataContext.Drivers.FirstOrDefault(x => x.Id ==
id);
        if (result != null)
        {
            _dataContext.Drivers.Remove(result);
            _dataContext.SaveChanges();
            return 1;
        }

        return 0;
    }

    public IEnumerable<Driver> GetAll()
    {
        return _dataContext.Drivers.ToList();
    }

    public Driver GetById(long id)
    {
        return _dataContext.Drivers.FirstOrDefault(x => x.Id == id);
    }

    public int Update(Driver item)
    {
        var result = _dataContext.Drivers.FirstOrDefault(x => x.Id ==
item.Id);
        if (result != null)
        {
            result.CertificateNumber = item.CertificateNumber;
            result.FirstName = item.FirstName;
            result.LastName = item.LastName;
            result.MiddleName = item.MiddleName;
            result.PhoneNumber = item.PhoneNumber;
            _dataContext.SaveChanges();
            return 1;
        }

        return 0;
    }
}

```

Листинг Б.7 – VehiclesRepository.cs

```

using LogisticApp.DataAccess.Abstractions;
using LogisticApp.DataAccess.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess.Repositories
{

```

```

public class VehiclesRepository : IRepository<Vehicle>
{
    private ApplicationContext _dataContext;
    public VehiclesRepository(ApplicationContext context)
    {
        this._dataContext = context;
    }
    public long Add(Vehicle item)
    {
        var result = _dataContext.Vehicles.Add(item);
        _dataContext.SaveChanges();
        return result.Entity.Id;
    }

    public int Delete(long id)
    {
        var result = _dataContext.Vehicles.FirstOrDefault(x => x.Id
== id);
        if (result != null)
        {
            _dataContext.Vehicles.Remove(result);
            _dataContext.SaveChanges();
            return 1;
        }

        return 0;
    }

    public IEnumerable<Vehicle> GetAll()
    {
        return _dataContext.Vehicles.ToList();
    }

    public Vehicle GetById(long id)
    {
        return _dataContext.Vehicles.FirstOrDefault(x => x.Id == id);
    }

    public int Update(Vehicle item)
    {
        var result = _dataContext.Vehicles.FirstOrDefault(x => x.Id
== item.Id);
        if (result != null)
        {
            result.VehicleNumber = item.VehicleNumber;
            result.LiftingCapacity = item.LiftingCapacity;
            result.FuelConsumption = item.FuelConsumption;
            result.BodyVolume = item.BodyVolume;
            result.BodySize = item.BodySize;
            result.Model = item.Model;
            result.VehicleType = item.VehicleType;
            _dataContext.SaveChanges();
            return 1;
        }
        return 0;
    }
}

```

Листинг Б.8 – WaybillsRepository.cs

```

using LogisticApp.DataAccess.Abstractions;
using LogisticApp.DataAccess.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess.Repositories
{
    public class WaybillsRepository : IRepository<Waybill>
    {
        private ApplicationContext _dataContext;
        public WaybillsRepository(ApplicationContext context)
        {
            this._dataContext = context;
        }
        public long Add(Waybill item)
        {
            var result = _dataContext.Waybills.Add(item);
            _dataContext.SaveChanges();
            return result.Entity.Id;
        }

        public int Delete(long id)
        {
            var result = _dataContext.Waybills.FirstOrDefault(x => x.Id
== id);
            if (result != null)
            {
                _dataContext.Waybills.Remove(result);
                _dataContext.SaveChanges();
                return 1;
            }

            return 0;
        }

        public IEnumerable<Waybill> GetAll()
        {
            return _dataContext.Waybills.ToList();
        }

        public Waybill GetById(long id)
        {
            return _dataContext.Waybills.FirstOrDefault(x => x.Id == id);
        }

        public int Update(Waybill item)
        {
            var result = _dataContext.Waybills.FirstOrDefault(x => x.Id
== item.Id);
            if (result != null)
            {
                result.FuelResource = item.FuelResource;
                result.Route = item.Route;
            }
        }
    }
}

```

```

        result.RouteVolume = item.RouteVolume;
        result.EffectiveLength = item.EffectiveLength;
        result.Driver = item.Driver;
        result.RouteType = item.RouteType;
        result.Vehicle = item.Vehicle;
        _dataContext.SaveChanges();
        return 1;
    }
    return 0;
}
}
}

```

Листинг Б.9 – ApplicationContext.cs

```

using LogisticApp.DataAccess.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LogisticApp.DataAccess
{
    public class ApplicationContext : DbContext
    {
        public DbSet<Waybill> Waybills { get; set; }
        public DbSet<Driver> Drivers { get; set; }
        public DbSet<Vehicle> Vehicles { get; set; }
        public ApplicationContext()
        {
            Database.EnsureCreated();
        }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            optionsBuilder.UseNpgsql("Host=localhost;Port=5433;Database=logisticdb;Us
ername=postgres;Password=123");
        }
    }
}

```

Листинг Б.10 – ReportPage.xaml.cs

```

using LogisticApp.DataAccess;
using LogisticApp.DataAccess.Models;
using LogisticApp.DataAccess.Repositories;
using LogisticApp.Reports;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;

```

```

using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Markup.Localizer;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace LogisticApp.Pages
{
    /// <summary>
    /// Interaction logic for ReportPage.xaml
    /// </summary>
    public partial class ReportPage : Page
    {
        private WaybillsRepository _waybillsRepo;
        public long WaybillId { get; set; }
        public ReportPage(ApplicationContext db, long waybillId)
        {
            InitializeComponent();
            _waybillsRepo = new WaybillsRepository(db);
            WaybillId = waybillId;
            InitializeData();
        }
        public ReportPage()
        {
            InitializeComponent();
        }
        public void InitializeData()
        {
            var waybill = _waybillsRepo.GetById(WaybillId);
            if (waybill == null)
            {
                MainBlock.Items.Add(new Label() { Content = "Данные по заданному маршруту не найдены." });
                RouteBlock.Items.Add(new Label() { Content = "Данные по заданному маршруту не найдены." });
            }
            else
            {
                var mainReport = CreateMainReport(waybill);
                var routeReport = CreateRouteReport(waybill);
                DrawReport(mainReport, MainBlock);
                DrawReport(routeReport, RouteBlock);
            }
        }
        private MainReport CreateMainReport(Waybill waybill)
        {
            var reportItems = new List<ReportItem>();
            reportItems.Add(new ReportItem()
            {
                ItemName = "Населенный пункт",
                ItemValue = waybill.Place ?? "Не указано",
                ItemTag = "place"
            });
        }
    }
}

```

```

reportItems.Add(new ReportItem()
{
    ItemName = "Общая протяженность маршрута",
    ItemValue = $"{waybill.EffectiveLength} м",
    ItemTag = "effective_length"
});

reportItems.Add(new ReportItem()
{
    ItemName = "Количество точек выгрузки",
    ItemValue = $"{waybill.RouteVolume}",
    ItemTag = "route_volume"
});

reportItems.Add(new ReportItem()
{
    ItemName = "Дата и время учета",
    ItemValue = DateTime.Now.ToString("g"),
    ItemTag = "date_time"
});

reportItems.Add(new ReportItem()
{
    ItemName = "Водитель",
    ItemValue = $"{waybill.Driver.LastName}
{waybill.Driver.FirstName} {waybill.Driver.MiddleName}",
    ItemTag = "driver_name"
});

reportItems.Add(new ReportItem()
{
    ItemName = "Контактная информация",
    ItemValue = $"{waybill.Driver.PhoneNumber}",
    ItemTag = "driver_phone"
});

reportItems.Add(new ReportItem()
{
    ItemName = "ТС",
    ItemValue = $"{waybill.Vehicle.VehicleType},
{waybill.Vehicle.Model}",
    ItemTag = "vehicle_info"
});
var report = new MainReport($"Общий расчет <{waybill.Id}>",
reportItems);
return report;
}
private void DrawReport(ReportAbstract report, ItemsControl
block)
{
    var title = new Label()
    {
        Content = report.Title,
        FontSize = 14,
        HorizontalAlignment = HorizontalAlignment.Center,
    };

```

```

        block.Items.Add(title);

        foreach (var item in report.Items)
        {
            var labelItem = new Label();
            switch (item.ItemTag)
            {
                case "header":
                {
                    labelItem.Content = item.ItemName + ": " +
item.ItemValue;

                    labelItem.FontWeight = FontWeights.Bold;
                    labelItem.FontSize = 12;
                    labelItem.HorizontalAlignment =
HorizontalAlignment.Center;
                    break;
                }
                case "start_point":
                {
                    labelItem.Content = item.ItemName + ": " +
item.ItemValue;

                    labelItem.FontSize = 10;
                    labelItem.FontStyle = FontStyles.Italic;
                    break;
                }
                case "route_point":
                {
                    labelItem.Content = item.ItemName + ": " +
item.ItemValue;

                    labelItem.FontSize = 10;
                    break;
                }
                default:
                {
                    labelItem.Content = item.ItemName + ": " +
item.ItemValue;

                    labelItem.FontSize = 12;
                    labelItem.HorizontalAlignment =
HorizontalAlignment.Left;
                    break;
                }
            }

            block.Items.Add(item);
        }
    }
    private RouteReport CreateRouteReport(Waybill waybill)
    {
        var reportItems = new List<ReportItem>();
        reportItems.Add(new ReportItem
        {
            ItemName = "Протяженность маршрута",
            ItemValue = $"{waybill.EffectiveLength} м",
            ItemTag = "effective_length"
        });
        reportItems.Add(new ReportItem()
        {
            ItemName = "Количество точек выгрузки",

```

```

        ItemValue = $"{waybill.RouteVolume}",
        ItemTag = "route_volume"
    });
    reportItems.Add(new ReportItem()
    {
        ItemName = "Средний расход топлива на 100 км",
        ItemValue = $"{waybill.FuelResource}",
        ItemTag = "fuel_resource"
    });

    reportItems.Add(new ReportItem()
    {
        ItemName = "Общий расход топлива",
        ItemValue = $"{GetFuelRateBy100(waybill.EffectiveLength,
waybill.FuelResource)} л",
        ItemTag = "fuel_rate"
    });

    reportItems.Add(new ReportItem()
    {
        ItemName = "Маршрут",
        ItemValue = string.Empty,
        ItemTag = "header"
    });
    var report = new RouteReport($"Расчет для ТС <{waybill.Id}>",
reportItems);
    try
    {
        var routeItems =
JsonSerializer.Deserialize<List<string>>(waybill.Route);
        if (routeItems != null)
        {
            report.PrepareRouteData(routeItems);
        }
    }
    catch (Exception ex)
    {
        report.RouteItems.Add(new ReportItem()
        {
            ItemName = "Ошибка загрузки данных, повторите попытку
позднее.",
            ItemValue = $"Data error",
            ItemTag = "error"
        });
        Trace.WriteLine(ex);
    }

    return report;
}

private double GetFuelRateBy100(int effectiveLength, int
fuelResource)
{
    return (effectiveLength * fuelResource) / 100000D;
}
}

```

Листинг Б.11 – RestManager.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace MapsApiLib
{
    public enum HttpVerb
    {
        GET,
        POST,
        PUT,
        DELETE
    }

    public class RestManager
    {
        private int _requestTimeout = 10000;
        public string EndPoint { get; set; }
        public HttpVerb Method { get; set; }
        public string ContentType { get; set; }
        public string PostData { get; set; }

        public RestManager()
        {
            EndPoint = "";
            Method = HttpVerb.GET;
            ContentType = "application/json";
            PostData = "";
        }

        public RestManager(string endpoint)
        {
            EndPoint = endpoint;
            Method = HttpVerb.GET;
            ContentType = "application/json";
            PostData = "";
        }

        public RestManager(string endpoint, HttpVerb method)
        {
            EndPoint = endpoint;
            Method = method;
            ContentType = "application/json";
            PostData = "";
        }

        public RestManager(string endpoint, HttpVerb method, string
postData)
        {
            EndPoint = endpoint;
            Method = method;
            ContentType = "application/json";
            PostData = postData;
        }

        public string MakeRequest()

```

```

    {
        return MakeRequest("");
    }

    public string MakeRequest(string parameters)
    {
        var request = (HttpWebRequest)WebRequest.Create(EndPoint +
parameters);

        request.Method = Method.ToString();
        request.ContentLength = 0;
        request.ContentType = ContentType;
        request.Timeout = _requestTimeout;

        if (!string.IsNullOrEmpty(PostData) && Method ==
HttpVerb.POST)
        {
            var encoding = new UTF8Encoding();
            var bytes = Encoding.GetEncoding("iso-8859-
1").GetBytes(PostData);
            request.ContentLength = bytes.Length;

            using (var writeStream = request.GetRequestStream())
            {
                writeStream.Write(bytes, 0, bytes.Length);
            }

            using (var response = (HttpWebResponse)request.GetResponse())
            {
                var responseValue = string.Empty;

                if (response.StatusCode != HttpStatusCode.OK)
                {
                    var message = String.Format("Request failed. Received
HTTP {0}", response.StatusCode);
                    throw new ApplicationException(message);
                }
                using (var responseStream = response.GetResponseStream())
                {
                    if (responseStream != null)
                        using (var reader = new
StreamReader(responseStream))
                        {
                            responseValue = reader.ReadToEnd();
                        }
                }
                return responseValue;
            }
        }
    }
}

```

Листинг Б.12 – MainWindow.xaml

```

<Window x:Class="LogisticApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

```

```

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:local="clr-namespace:LogisticApp"
mc:Ignorable="d"
Title="MainWindow" Height="720" Width="1200">
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="100"/>
    <ColumnDefinition/>
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition Height="80"/>
    <RowDefinition/>
  </Grid.RowDefinitions>
  <Grid x:Name="TopPanelGrid" Grid.ColumnSpan="2" Grid.Row="0"
Background="#2A2928">
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition Width="70"/>
      <ColumnDefinition Width="70"/>
      <ColumnDefinition Width="70"/>
    </Grid.ColumnDefinitions>
    <Label Foreground="#CFCA44" FontSize="40" Margin="30,0,0,0"
VerticalAlignment="Center" FontFamily="Krona One" FontWeight="Normal"
FontStyle="Normal" Opacity="0.9">LOGISTIC</Label>
  </Grid>

  <Grid x:Name="ToolsPanelGrid" Grid.Row="1" Grid.Column="0"
Background="#363433">
    <Grid.RowDefinitions>
      <RowDefinition/>
      <RowDefinition Height="100"/>
    </Grid.RowDefinitions>
    <ItemsControl Grid.Row="0">
      <Border x:Name="HomePageButton" Height="100" Width="100"
MouseDown="HomePageButton_MouseDown" Cursor="Hand">
        <Image Source="Resources/MainPage.png" Margin="30">

          </Image>
        </Border>

        <Border x:Name="MapPageButton" Height="100" Width="100"
MouseDown="MapPageButton_MouseDown" Cursor="Hand">
          <Image Source="Resources/MapCalculate.png"
Margin="20">

            </Image>
          </Border>

          <Border x:Name="ProductsPageButton" Height="100"
Width="100" MouseDown="ProductsPageButton_MouseDown" Cursor="Hand">
            <Image Source="Resources/Products.png" Margin="20">

              </Image>
            </Border>

            <Border x:Name="ReportPageButton" Height="100"
Width="100" MouseDown="ReportPageButton_MouseDown" Cursor="Hand">
              <Image Source="Resources/Details.png" Margin="25">

```

```

        </Image>
    </Border>

    </ItemsControl>
    <Border x:Name="SettingsButton" Height="100" Width="100"
Grid.Row="1" MouseDown="SettingsButton_MouseDown" Cursor="Hand">
        <Image Source="Resources/Settings.png" Margin="25"
IsHitTestVisible="False">

            </Image>
        </Border>
    </Grid>

    <Grid x:Name="PagePanelGrid" Grid.Row="1" Grid.Column="1"
Background="#F0EFED">
        <Frame x:Name="MainFrame"
NavigationUIVisibility="Hidden"></Frame>
    </Grid>
</Grid>
</Window>

```

Приложение В
(рекомендуемое)
Листинг модульных тестов

Листинг В.1 – Файл AntLibTests.cs

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Collections.Generic;

namespace AntLib.Tests
{
    [TestClass]
    public class AntLibTests
    {
        private TestContext testContextInstance;
        public TestContext TestContext
        {
            get { return testContextInstance; }
            set { testContextInstance = value; }
        }
        /// <summary>
        /// Проверка общей работоспособности алгоритма для реальных
расстояний
        /// между точками маршрута, который необходимо оптимизировать
        /// </summary>
        [TestMethod]
        public void NormalMapCityCount_Test()
        {
            OptimizationTool optimizationTool = new OptimizationTool();

            optimizationTool.InitializeParams(3, 2, 0.01, 2, 150, 500);
            double[][] distances = new double[14][];
            distances[0] = new double[14] { 0, 2.27, 1.66, 2.7, 3.2,
1.56, 0.88, 1.3, 1.96, 2.14, 1.46, 1.17, 1.47, 1.21 };
            distances[1] = new double[14] { 2.27, 0, 1.41, 2.97, 3.5,
1.83, 1.14, 1.29, 1.85, 2.17, 1.45, 1.21, 1.24, 0.97 };
            distances[2] = new double[14] { 1.66, 1.41, 0, 2.16, 2.64,
2.55, 1.86, 2.04, 2.73, 2.89, 2.17, 1.93, 1.66, 1.69 };
            distances[3] = new double[14] { 2.7, 2.97, 2.16, 0, 0.77,
1.99, 2.1, 2.88, 3.1, 3.1, 2.59, 2.48, 3, 2.8 };
            distances[4] = new double[14] { 3.2, 3.5, 2.64, 0.77, 0,
2.23, 2.34, 3.1, 3.3, 3.3, 2.83, 2.72, 3.3, 3 };
            distances[5] = new double[14] { 1.56, 1.83, 2.55, 1.99, 2.23,
0, 0.35, 1.13, 1.54, 1.77, 0.82, 0.73, 1.29, 1.05 };
            distances[6] = new double[14] { 0.88, 1.14, 1.86, 2.1, 2.34,
0.35, 0, 0.86, 1.26, 1.42, 0.54, 0.46, 1.02, 0.78 };
            distances[7] = new double[14] { 1.3, 1.29, 2.04, 2.88, 3.1,
1.13, 0.86, 0, 1.67, 1.73, 1.2, 1.05, 0.17, 0.44 };
            distances[8] = new double[14] { 1.96, 1.85, 2.73, 3.1, 3.3,
1.54, 1.26, 1.67, 0, 1.02, 1.04, 1.26, 0.9, 1.16 };
            distances[9] = new double[14] { 2.14, 2.17, 2.89, 3.1, 3.3,
1.77, 1.42, 1.73, 1.02, 0, 0.86, 1.07, 1.39, 1.34 };
            distances[10] = new double[14] { 1.46, 1.45, 2.17, 2.59,
2.83, 0.82, 0.54, 1.2, 1.04, 0.86, 0, 0.65, 1.16, 0.92 };
            distances[11] = new double[14] { 1.17, 1.21, 1.93, 2.48,
2.72, 0.73, 0.46, 1.05, 1.26, 1.07, 0.65, 0, 0.85, 0.61 };
            distances[12] = new double[14] { 1.47, 1.24, 1.66, 3, 3.3,
1.29, 1.02, 0.17, 0.9, 1.39, 1.16, 0.85, 0, 0.35 };

```

```

        distances[13] = new double[14] { 1.21, 0.97, 1.69, 2.8, 3,
1.05, 0.78, 0.44, 1.16, 1.34, 0.92, 0.61, 0.35, 0 };

        int[] expectedPath =
optimizationTool.GetBestSolution(distances, 14, out double bestLength);

        bool isNormalPath = true;
        for (int i = 0; i < expectedPath.Length; i++)
        {
            if (expectedPath[i] < 0 || expectedPath[i] > 13)
            {
                isNormalPath = false;
                break;
            }
        }

        string route = (expectedPath[0] + 1).ToString();

        for (int i = 1; i < expectedPath.Length; i++)
        {
            route += "-" + (expectedPath[i] + 1);
        }
        TestContext.WriteLine(route + " " + Math.Round(bestLength, 2)
+ " км");
        Assert.IsTrue(isNormalPath);
    }
    private void ShowLength(double[][] distances, List<int> route)
    {
        double sum = 0;
        for (int i = 1; i < route.Count; i++)
        {
            sum += distances[route[i] - 1][route[i - 1] - 1];
        }
        TestContext.WriteLine(sum + " км");
    }
    /// <summary>
    /// Проверка общей работоспособности алгоритма для расчета длины
маршрута
    /// для реальных расстояний между точками маршрута, который
необходимо оптимизировать
    /// </summary>
[TestMethod]
    public void NormalMapBestTrail_Test()
    {
        OptimizationTool optimizationTool = new OptimizationTool();

        optimizationTool.InitializeParams(3, 2, 0.01, 2, 200, 1000);
        double[][] distances = new double[14][];
        distances[0] = new double[14] { 0, 2.27, 1.66, 2.7, 3.2,
1.56, 0.88, 1.3, 1.96, 2.14, 1.46, 1.17, 1.47, 1.21 };
        distances[1] = new double[14] { 2.27, 0, 1.41, 2.97, 3.5,
1.83, 1.14, 1.29, 1.85, 2.17, 1.45, 1.21, 1.24, 0.97 };
        distances[2] = new double[14] { 1.66, 1.41, 0, 2.16, 2.64,
2.55, 1.86, 2.04, 2.73, 2.89, 2.17, 1.93, 1.66, 1.69 };
        distances[3] = new double[14] { 2.7, 2.97, 2.16, 0, 0.77,
1.99, 2.1, 2.88, 3.1, 3.1, 2.59, 2.48, 3, 2.8 };
        distances[4] = new double[14] { 3.2, 3.5, 2.64, 0.77, 0,
2.23, 2.34, 3.1, 3.3, 3.3, 2.83, 2.72, 3.3, 3 };
    }

```

```

        distances[5] = new double[14] { 1.56, 1.83, 2.55, 1.99, 2.23,
0, 0.35, 1.13, 1.54, 1.77, 0.82, 0.73, 1.29, 1.05 };
        distances[6] = new double[14] { 0.88, 1.14, 1.86, 2.1, 2.34,
0.35, 0, 0.86, 1.26, 1.42, 0.54, 0.46, 1.02, 0.78 };
        distances[7] = new double[14] { 1.3, 1.29, 2.04, 2.88, 3.1,
1.13, 0.86, 0, 1.67, 1.73, 1.2, 1.05, 0.17, 0.44 };
        distances[8] = new double[14] { 1.96, 1.85, 2.73, 3.1, 3.3,
1.54, 1.26, 1.67, 0, 1.02, 1.04, 1.26, 0.9, 1.16 };
        distances[9] = new double[14] { 2.14, 2.17, 2.89, 3.1, 3.3,
1.77, 1.42, 1.73, 1.02, 0, 0.86, 1.07, 1.39, 1.34 };
        distances[10] = new double[14] { 1.46, 1.45, 2.17, 2.59,
2.83, 0.82, 0.54, 1.2, 1.04, 0.86, 0, 0.65, 1.16, 0.92 };
        distances[11] = new double[14] { 1.17, 1.21, 1.93, 2.48,
2.72, 0.73, 0.46, 1.05, 1.26, 1.07, 0.65, 0, 0.85, 0.61 };
        distances[12] = new double[14] { 1.47, 1.24, 1.66, 3, 3.3,
1.29, 1.02, 0.17, 0.9, 1.39, 1.16, 0.85, 0, 0.35 };
        distances[13] = new double[14] { 1.21, 0.97, 1.69, 2.8, 3,
1.05, 0.78, 0.44, 1.16, 1.34, 0.92, 0.61, 0.35, 0 };

        optimizationTool.GetBestSolution(distances, 14, out double
bestLength);

        Assert.IsTrue(bestLength > 0);
    }
    /// <summary>
    /// Проверка работоспособности алгоритма при некорректных входных
параметрах
    /// </summary>
    [TestMethod]
    [ExpectedException(typeof(System.IndexOutOfRangeException),
"Введенные параметры могут быть обработаны")]
    public void FailedParams_Test()
    {
        OptimizationTool optimizationTool = new OptimizationTool();

        optimizationTool.InitializeParams(-1, -2, 5, 6, 0, 0);
        double[][] distances = new double[4][];
        distances[0] = new double[4] { 0, 1, 2, 5 };
        distances[1] = new double[4] { 1, 0, 1, 3 };
        distances[2] = new double[4] { 2, 1, 0, 1 };
        distances[3] = new double[4] { 5, 3, 1, 0 };

        optimizationTool.GetBestSolution(distances, 4, out double
bestLength);
    }
    /// <summary>
    /// Проверка работоспособности алгоритма при некорректном
количестве точек маршрута
    /// </summary>
    [TestMethod]
    [ExpectedException(typeof(System.IndexOutOfRangeException),
"Введено корректное количество городов")]
    public void FailedCityParam_Test()
    {
        OptimizationTool optimizationTool = new OptimizationTool();

        optimizationTool.InitializeParams(3, 2, 0.01, 2, 200, 100);
        double[][] distances = new double[4][];

```

```

        distances[0] = new double[4] { 0, 1, 2, 5 };
        distances[1] = new double[4] { 1, 0, 1, 3 };
        distances[2] = new double[4] { 2, 1, 0, 1 };
        distances[3] = new double[4] { 5, 3, 1, 0 };

        optimizationTool.GetBestSolution(distances, 10, out double
bestLength);
    }
    /// <summary>
    /// Проверка работоспособности алгоритма при некорректно заданных
расстояниях между точками
    /// </summary>
[TestMethod]
[ExpectedException(typeof(System.IndexOutOfRangeException),
"Введены корректные расстояния")]
public void FailedDistanceParam_Test()
{
    OptimizationTool optimizationTool = new OptimizationTool();

    optimizationTool.InitializeParams(3, 2, 0.01, 2, 200, 100);
    double[][] distances = new double[4][];
    distances[0] = new double[4] { 0, -1, 2, 5 };
    distances[1] = new double[4] { 1, 0, -1, 3 };
    distances[2] = new double[4] { 2, -1, 0, 1 };
    distances[3] = new double[4] { -5, 3, 1, 0 };

    optimizationTool.GetBestSolution(distances, 10, out double
bestLength);
}
/// <summary>
/// Проверка точности расчета алгоритма при заранее известном
оптимальном маршруте
/// </summary>
[TestMethod]
public void AntAccuracy_Test()
{
    OptimizationTool optimizationTool = new OptimizationTool();

    optimizationTool.InitializeParams(3, 2, 0.01, 2, 200, 100);
    double[][] distances = new double[4][];
    distances[0] = new double[4] { 0, 1, 2, 5 };
    distances[1] = new double[4] { 1, 0, 1, 3 };
    distances[2] = new double[4] { 2, 1, 0, 1 };
    distances[3] = new double[4] { 5, 3, 1, 0 };

    optimizationTool.GetBestSolution(distances, 4, out double
bestLength);

    int expectedTrailValue = 3;

    TestContext.WriteLine("Лучшая протяженность маршрута: " +
expectedTrailValue + ", Найденная протяженность: " + bestLength);

    Assert.IsTrue(bestLength == expectedTrailValue);
}
}
}

```

Приложение Г
(рекомендуемое)
Результаты тестирования

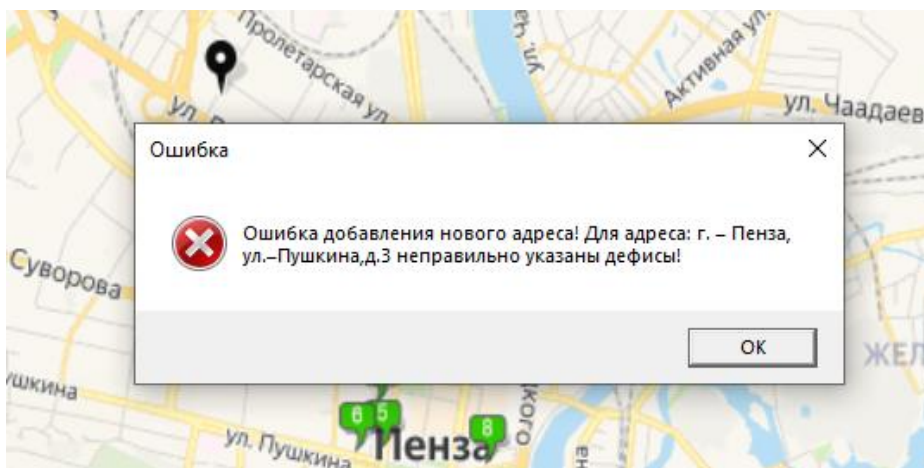


Рисунок Г.1 – Результаты тестирования класса эквивалентности «Неправильно поставленные дефисы»

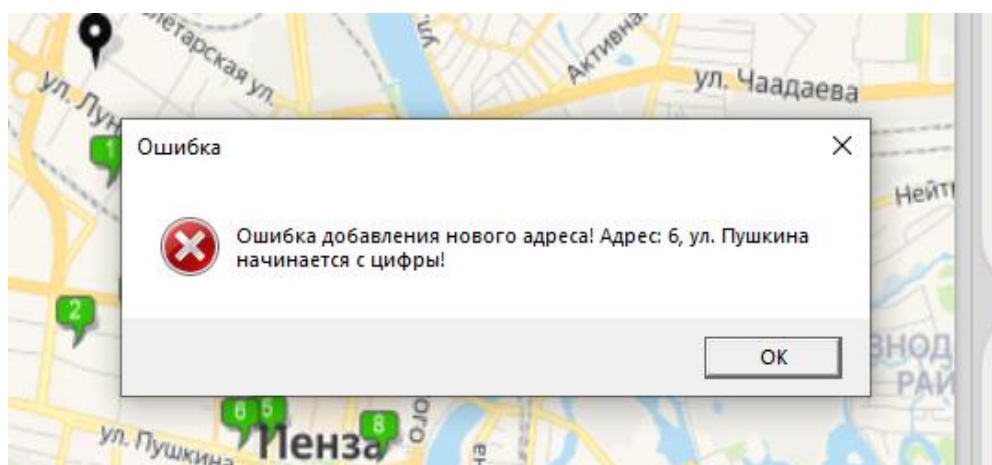


Рисунок Г.2 – Результаты тестирования класса эквивалентности «Начинается с цифры»

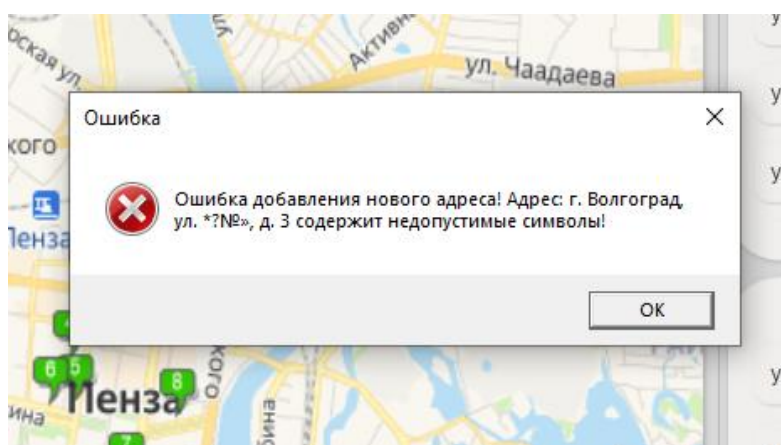


Рисунок Г.3 – Результаты тестирования класса эквивалентности «Содержит недопустимые символы»

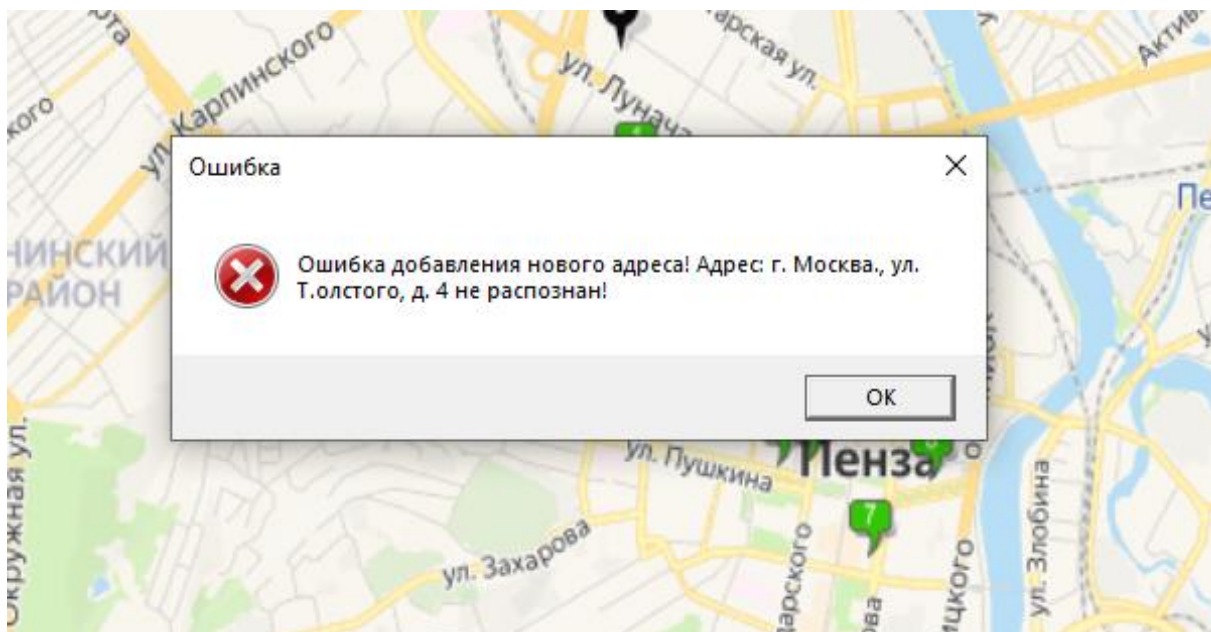


Рисунок Г.4 – Результаты тестирования класса эквивалентности «Точка используется не для сокращения»

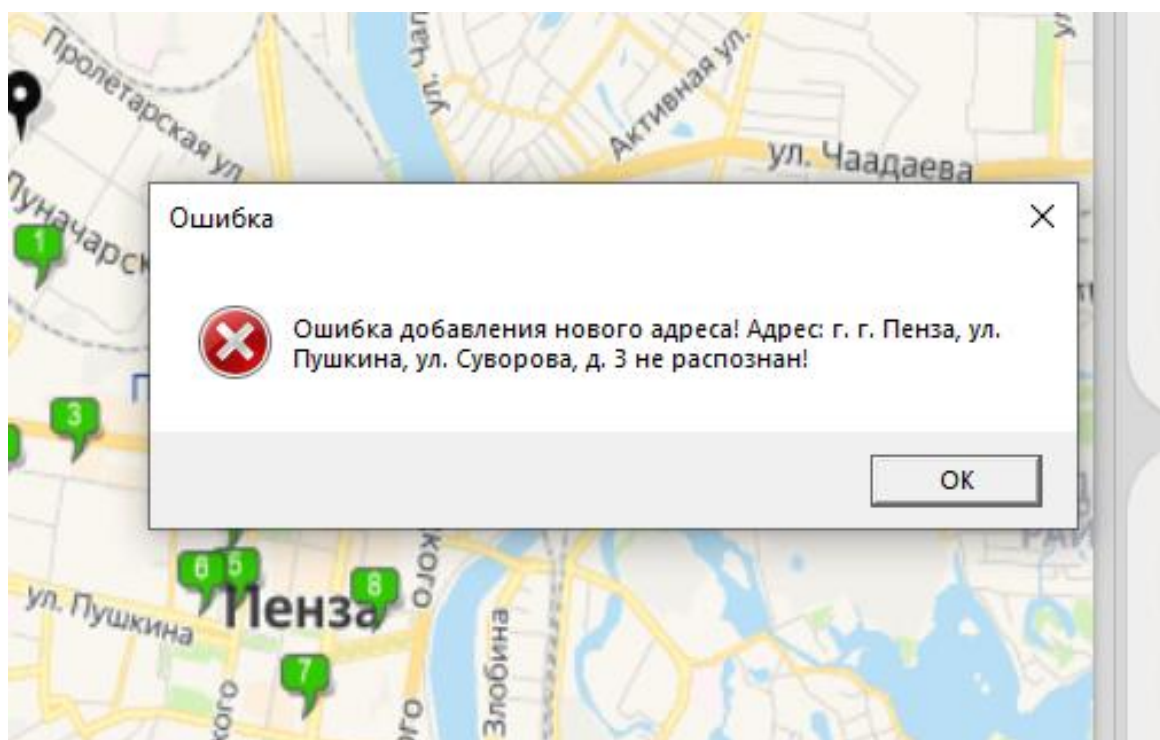


Рисунок Г.5 – Результаты тестирования логической границы «Указано несколько улиц»

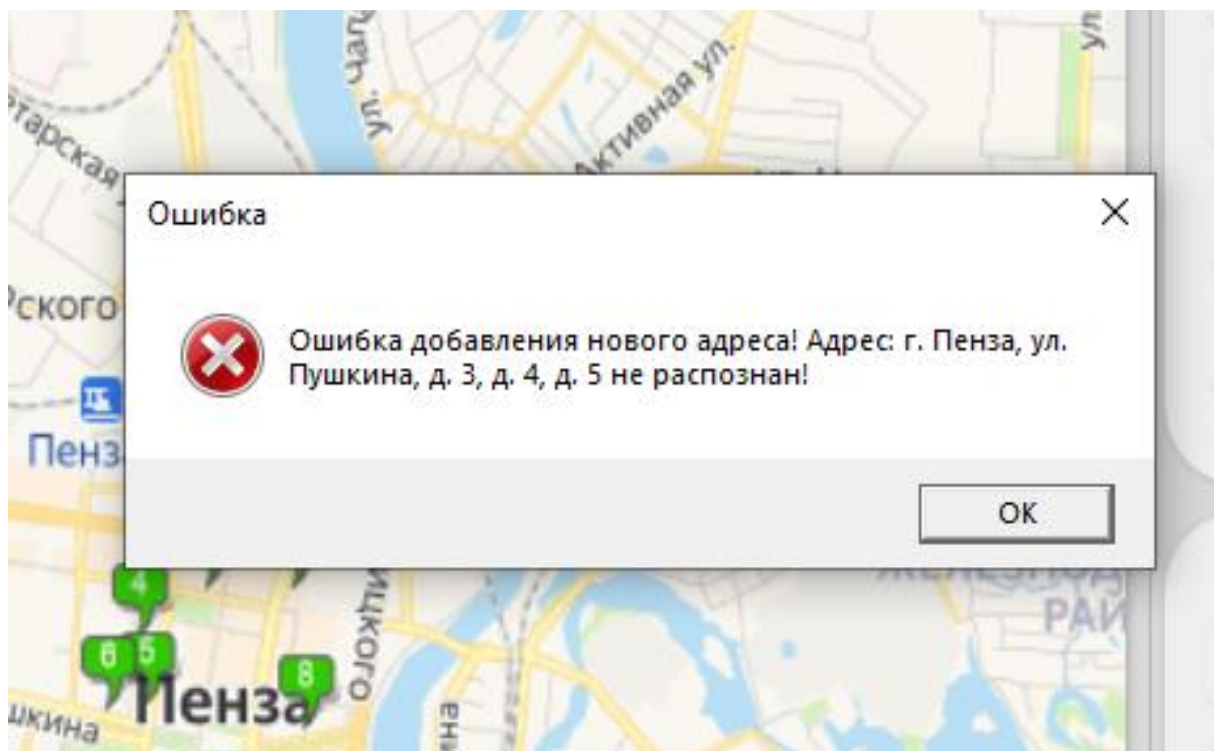


Рисунок Г.6 – Результаты тестирования логической границы «Указано несколько домов»

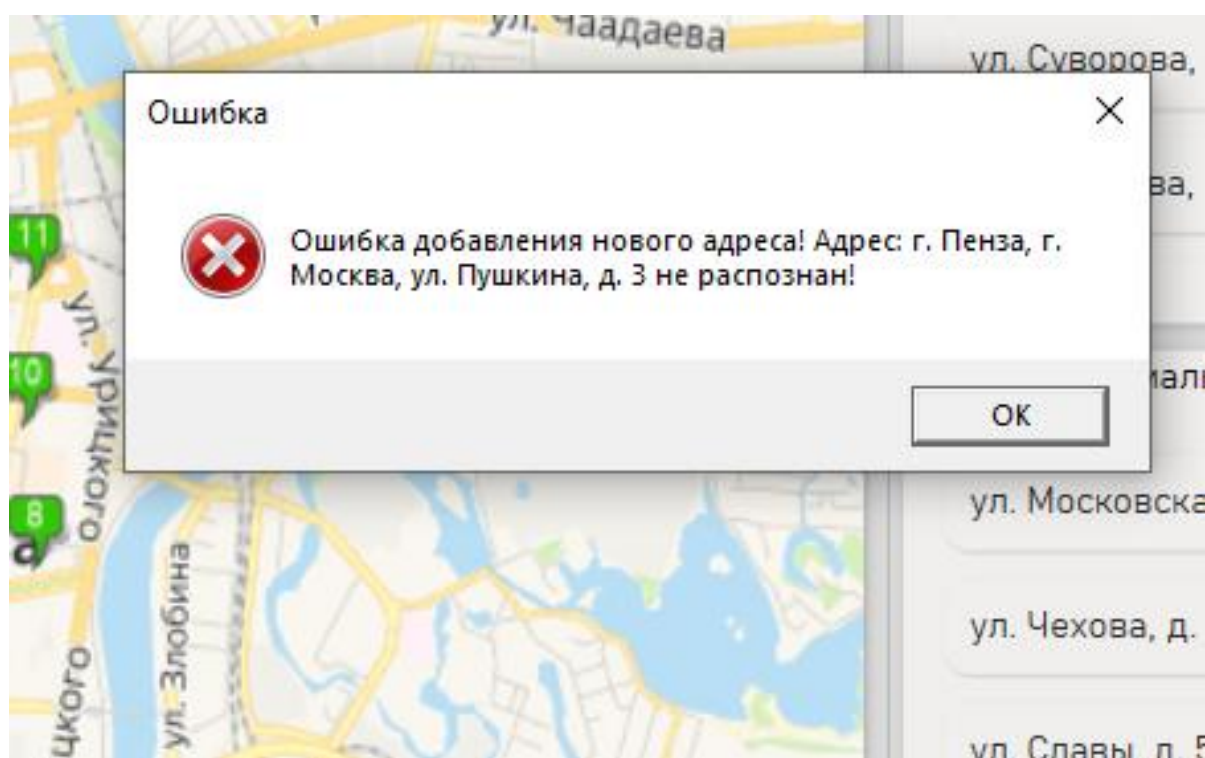


Рисунок Г.7 – Результаты тестирования логической границы «Указано несколько городов»

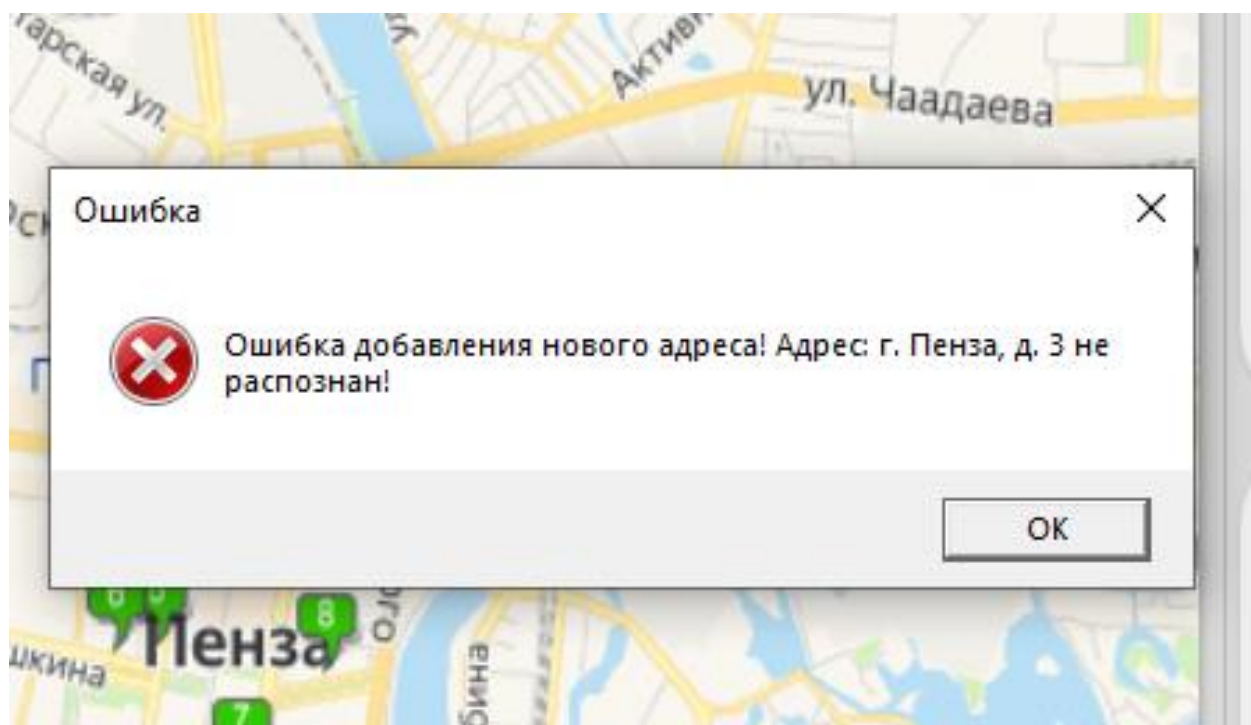


Рисунок Г.8 – Результаты тестирования логической границы «Не указана улица»

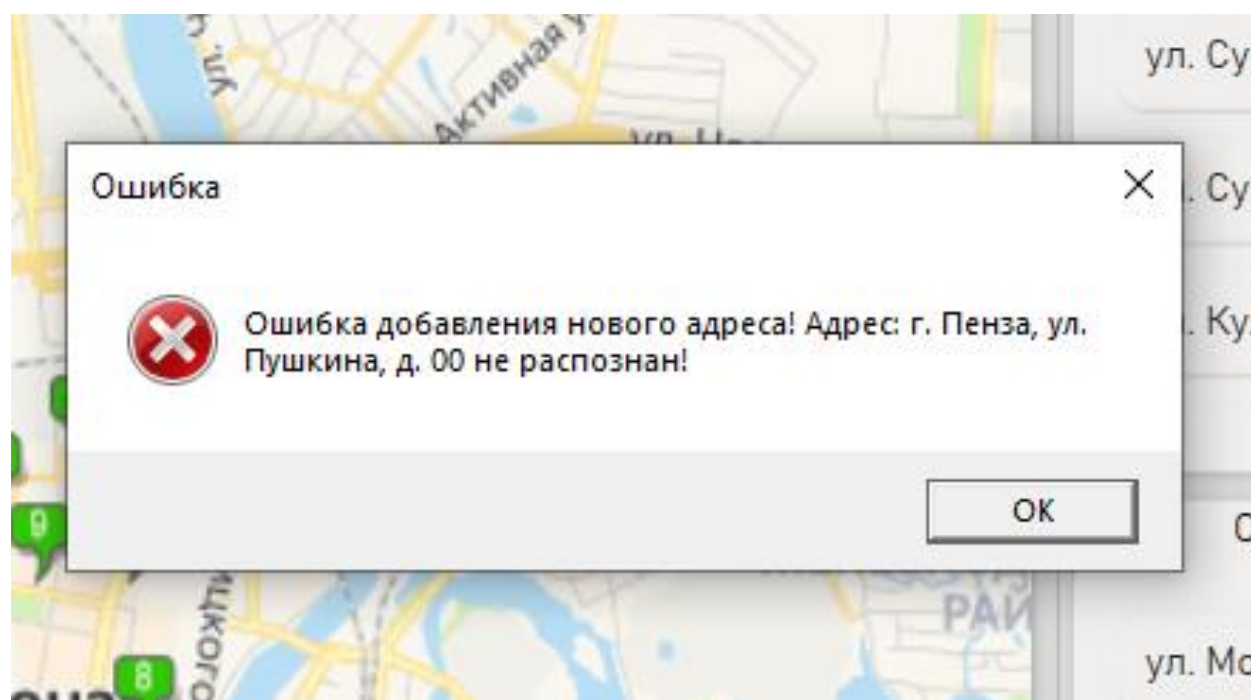


Рисунок Г.9 – Результаты тестирования логической границы «Неверно указан номер дома»

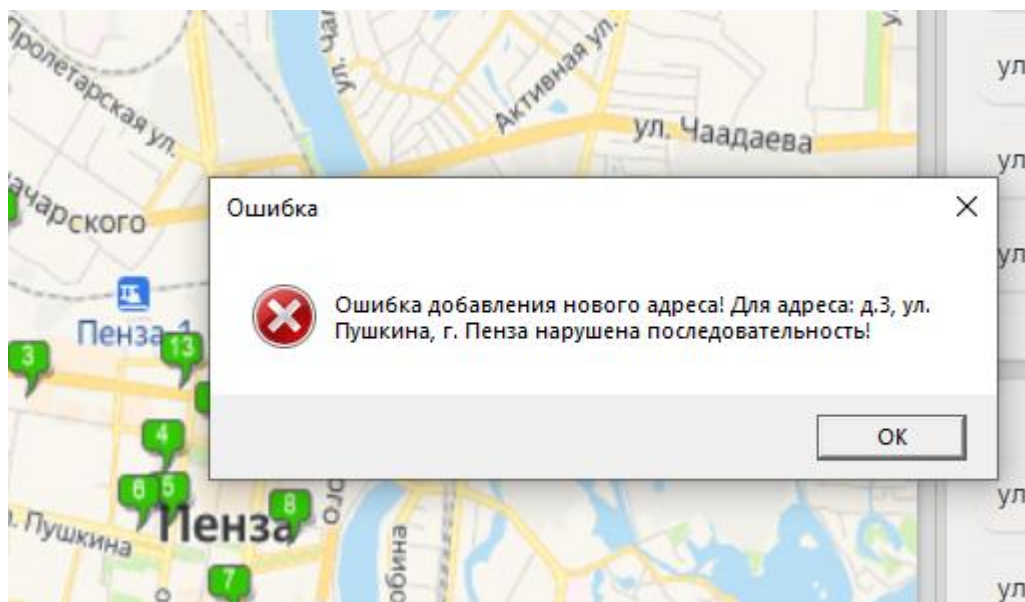


Рисунок Г.10 – Результаты производственной границы «Нарушена последовательность адреса»