

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

Программное обеспечение
автоматизированной системы построения
маршрута движения

Студент

(подпись, дата) Лебедев Константин Андреевич
(ФИО полностью)

Руководитель

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

Нормоконтролёр

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

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

Заведующий кафедрой

(подпись) Козлов А.Ю.
(фамилия, инициалы)

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

Секретарь ГЭК

(подпись) Попова Н.А.
(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 Лебедев Константин Андреевич гр. 20ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Программное обеспечение автоматизированной системы построения маршрута движения
- Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____
2. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н
3. Задание на работу (назначение разработки, исходные данные и т.п.):
- 1) Разработать программное обеспечение автоматизированной системы построения маршрута движения.
- 2) Среда выполнения приложения: операционная система Windows 10 и выше. Среда выполнения сервера: операционная система Linux.
- Языковые средства разработки: C#, JavaScript, PHP, SQL. Инструментальные средства разработки: Microsoft Visual Studio, Atom, MySQL Server, Microsoft Project, DrawIO.
- Технологии разработки: RUP, ООП.
- 3) Основные функции приложения:
- а) построение оптимального маршрута;
- б) просмотр информации о различных маршрутах;

в) возможность сохранения маршрута.

4. Перечень подлежащих разработке вопросов:

1) Анализ предметной области и постановка задачи;

2) Проектирование;

3) Реализация;

4) Контроль качества программного обеспечения;

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 116 листа, 42 рисунка, 16 таблиц, 32 использованных источников, 6 приложений.

ДЕСКТОПНОЕ ПРИЛОЖЕНИЕ, КАРТЫ, ПУТИ, ОПТИМИЗАЦИЯ МАРШРУТА, WINDOWS, MYSQL SERVER, C#, JAVASCRIPT

Объектом разработки является программное обеспечение автоматизированной системы построения маршрута движения.

Цель работы – разработка приложения для построения маршрута движения. Данное приложение позволит построить, сохранить, удалить, выбрать, а также изменить маршрут движения.

Разработка клиентской части происходила на языке программирования C#, JavaScript, серверной части – на языке PHP, SQL. В качестве среды разработки использовалась Microsoft Visual Studio 2022 и Atom. В качестве СУБД была использована СУБД MySQL Server.

Разработка приложения производилась на языке программирования C# и JavaScript. В качестве среды разработки использовались Microsoft Visual Studio 2022, Atom IDE, операционная система Windows 10. Приложение предназначено для использования под управлением ОС Windows 10 и выше, серверная часть использует операционную систему семейства Linux.

					ПГУ 09.03.04 – 08BP201.13 ПЗ			
И	Л	№ докум.	Под	Д	Программное обеспечение автоматизированной системы построения маршрута движения. Пояснительная записка	Лит.	Лист	Листов
Разраб.	Лебедев К.А.						4	116
Пров.	Князев В.Н.							
Н.	Такташкин Д.В.					гр. 20ВП1		
Утв.	Козлов А.Ю.							

Содержание

Введение.....	7
1 Анализ предметной области и требований	11
1.1 Анализ предметной области и постановка задачи	11
1.2 Анализ аналогичных программных продуктов	11
1.2.1 Программное обеспечение для навигации персонала на промышленной территории от компании «Университет 20.35»	12
1.2.2 Программное обеспечение интерактивной навигации «LOGICMAP» от «Zorgtech».....	14
1.2.3 Система indoor навигации от компании «Навигационные решения» ...	16
1.2.4 Сравнение аналогов	18
1.3 Анализ целевой аудитории	18
1.4 Анализ функциональных требований.....	21
1.5 Анализ нефункциональных требований.....	26
1.6 Планирование разработки и оценка бюджета.....	27
2 Проектирование программного обеспечения	39
2.1 Выбор технологий и инструментальных средств разработки.....	39
2.2 Проектирование архитектуры программного обеспечения	40
2.3 Проектирование интерфейса пользователя.....	41
2.4 Логическое проектирование базы данных	46
2.5 Проектирование серверной части приложения	49
3 Реализация программного обеспечения	51
3.1 Реализация базы данных	51
3.2 Разработка серверной части.....	52
3.3 Реализация клиентской части приложения	53
4 Тестирование программного продукта	56
4.1 Тестирование серверной части приложения	56
4.2 Тестирование клиентской части приложения.....	56
4.2.1 Функциональное тестирование	56

	6
4.2.2 Модульное тестирование	57
4.3 Метрики кода.....	59
4.3.1 Метрики размера программы	61
4.3.2 Метрики Холстеда	61
Заключение	64
Список использованных источников	65
Приложение А (справочное) Глоссарий	69
Приложение Б (рекомендуемое) Исходный код скриптов базы данных	73
Приложение В (рекомендуемое) Исходный код серверной части	77
Приложение Г (рекомендуемое) Исходный код приложения	80
Приложение Д (рекомендуемое) Результаты тестирования серверной части..	110
Приложение Е (рекомендуемое) Результаты тестирования приложения	113

Введение

Современный мир характеризуется широким внедрением информационных технологий во все сферы деятельности, тем самым сильно влияют на жизнь общества и изменяют ее.

Не остается в стороне от этой тенденции и наша страна. В ней также взят курс на максимальное внедрение информационных технологий во всех сферах деятельности, который поддерживается на государственном уровне. 9 мая 2017 года была утверждена «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». В ней были поставлены основные цели и задачи политики Российской Федерации в сфере информационных технологий, а также объявлен курс на формирование национальной цифровой экономики [1]. С этой же целью в том же 2017 году была принята государственная программа «Цифровая экономика». Она призвана обеспечить «развитие цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост и экономический суверенитет» [2].

Не осталась в стороне и программное обеспечение локальной логистики и построения пути. Это связано с тем что все больше промышленных предприятий переходят на автоматизированную логистику роботизированных транспортных систем. Информационные технологии играют ключевую роль в оптимизации и ускорении процессов в данной отрасли. Сейчас многие компании предприятия стремятся использовать информационные системы для управления складскими операциями, включая планирование маршрутов, управление запасами и мониторинг производительности роботов-транспортников. Это изменяет подходы к логистике, делая её более эффективной и гибкой благодаря автоматизации и роботизации. Предприятия используют программное обеспечение для управления логистикой, которое

позволяет оптимизировать процессы, сокращая время, увеличивая точность выполнения задач, а также уменьшает затраты на труд.

Промышленные предприятия, осознавая значение информационных технологий, инвестируют в разработку и внедрение программного обеспечения для автоматизации логистических процессов и построения маршрута. Это включает в себя использование робототехники для транспортировки грузов, что значительно улучшает эффективность и снижает затраты на труд.

В целом, переход к цифровому управлению логистикой на промышленных предприятиях является ответом на вызовы современности, позволяя предприятиям оставаться конкурентоспособными и эффективными в быстро меняющемся мире.

С развитием робототехники появляются новые возможности для автоматизации логистических процессов на промышленных предприятиях. Роботы могут выполнять задачи, связанные с перемещением грузов, что значительно упрощает и ускоряет процессы. Это позволяет предприятиям не только увеличивать объемы производства, но и делать их более гибкими и адаптивными к изменяющимся требованиям рынка.

Для успешной интеграции роботов в производственные процессы необходимо разработать специализированное программное обеспечение. Это ПО позволяет управлять движениями роботов, оптимизировать их работу и координировать действия с другими системами, это становится основой для создания эффективной и безопасной роботизированной среды, которая может работать без постоянного участия человека.

На складах активно применяется автоматизация для упрощения процесса хранения и извлечения товаров. Использование программного обеспечения для управления роботами-помощниками позволяет сократить время на выполнение этих задач и увеличить точность операций. Это не только повышает эффективность работы склада, но и снижает вероятность ошибок, что важно для обеспечения высокого качества обслуживания клиентов.

Программное обеспечение может использоваться для определения оптимальных маршрутов перемещения грузов, учитывая различные параметры, такие как размеры грузов, доступное пространство и расписание работы. Это помогает избежать лишних пробегов и экономит ресурсы, что особенно важно для предприятий с большим количеством складских помещений и сложной логистической сетью.

Программное обеспечение должно быть достаточно гибким, чтобы легко адаптироваться к изменяющимся условиям производства и масштабироваться при необходимости расширения производственных мощностей. Это позволяет предприятиям быстро реагировать на изменения в рыночных условиях и поддерживать высокий уровень конкурентоспособности.

Текущее состояние применения программного обеспечения (ПО) в области локальной логистики и построения маршрута на предприятиях характеризуется значительными изменениями и развитием технологий. Это отражает стремление компаний к повышению эффективности процессов управления складом, оптимизации цепочек поставок и улучшению качества обслуживания клиентов.

Внедрение ПО построения маршрута в систему автоматического погрузчика (AGV) и робототехники на склад, позволит не только увеличить скорость обработки грузов, но и сократить количество рабочих мест, требующих человеческого труда. Это приводит к снижению трудовых затрат и повышению безопасности работы на складе. Роботы могут работать круглосуточно без перерывов, что особенно актуально для ночной работы или выполнения задач в выходные дни.

Системы автоматической идентификации и сортировки (ASRS) используются для организации хранения большого количества мелких единиц товаров. Они позволяют быстро находить нужные товары, минимизируют время поиска и увеличивая эффективность использования пространства склада.

Интеграция различных систем и платформ через API становится все более распространенной практикой. Это позволяет создавать гибкие и масштабируемые решения.

Таким образом на основе выше перечисленных фактов можно сделать вывод о целесообразности разработки программного обеспечения построения маршрута, позволяя автоматизировать процессы нахождения и построения оптимального пути, эта тема весьма актуальна и перспективна в условиях современного общества.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Разрабатываемое программное обеспечение будет представлять собой автоматизированную систему построения маршрутов движения, предназначенную для облегчения процесса планирования движения по промышленным предприятиям и закрытым помещениям. С помощью данного ПО, пользователи смогут избежать необходимости вручную составлять маршруты, что существенно упрощает процесс поиска наиболее оптимальных вариантов передвижения. Для использования будет достаточно иметь ПК или планшет с установленным программным обеспечением и стабильное сетевое соединение, что позволит получать актуальную информацию о маршрутах.

Основные функциональные возможности программного обеспечения будут в себя включать:

- регистрацию новых пользователей и авторизацию входящих в систему;
- просмотр информации о различных маршрутах, включая детализацию расстояний;
- построение оптимального маршрута на основе заданных пользователем параметров, таких как начальная и конечная точки;
- возможность сохранения маршрута.

В дополнение к основным функциям, в системе должны быть предусмотрены права администраторов, которые позволят им управлять контентом и функционалом сервиса. Администраторы смогут добавлять новые маршруты, удалять устаревшие или ненужные.

1.2 Анализ аналогичных программных продуктов

В настоящее время существует несколько программных продуктов по локальной логистике и маршрутизации. Рассмотрим наиболее известные программы такого рода и произведем для них сравнительный анализ.

1.2.1 Программное обеспечение для навигации персонала на промышленной территории от компании «Университет 2035»

Компания «Университет 2035» представляет собой образовательную организацию, специализирующуюся на предоставлении качественного и своевременного развития каждого человека в условиях постоянных изменений через деятельность в области ИТ. Основная цель университета – подготовка специалистов, способных адаптироваться к быстро меняющемуся миру благодаря использованию цифровых технологий.

Далее приведены основные характеристики Университета 2035:

- глобальный и цифровой характер, университет активно использует цифровую платформу для обучения, что позволяет ему охватывать широкую аудиторию студентов по всему миру;
- большое количество пользователей и участников, за четыре года работы университета более 150 тысяч человек прошли обучение, а число пользователей платформы превышает 257 тысяч;
- разнообразие программ и курсов, университет предлагает более 25 квалификаций на основе данных, а также проводит проектные интенсивы, направленные на развитие навыков командной работы и решения сложных задач;
- инновационные подходы к обучению, внедрение новых форматов обучения, такие как «Супер-НИИ» в датацентричном образовании, и развивает сетевое образовательное пространство;
- прозрачность и открытость, университет активно делится своим опытом через лекторий, где представлены лекции и экспертные дискуссии, а также собирает отзывы о своих сервисах.

Компания «Университет 2035» стремится не только к обучению, но и к развитию инфраструктуры образования, созданию новых возможностей для студентов и профессионалов в области ИТ. Его работа направлена на

поддержание высоких стандартов образования и подготовку будущих лидеров в цифровом мире.

Компания «Университет 2035» представляет собой первый в России глобальный цифровой университет, активно развивающийся в области IT и цифровых технологий. Университет предлагает широкий спектр услуг и проектов, направленных на развитие компетенций в области новых технологий и искусственного интеллекта. Одним из ключевых направлений деятельности является разработка и внедрение программного обеспечения для автоматизации процессов на промышленных территориях, включая навигацию персонала.

Программное обеспечение для навигации персонала на промышленной территории от «Университета 2035» может быть частью более крупных проектов, связанных с цифровой трансформацией промышленности. Это ПО позволяет оптимизировать перемещения сотрудников внутри производственной площадки, учитывая расположение объектов, рабочих мест, складов и других важных точек. Такая система может значительно улучшить эффективность работы персонала, сокращая время на поиск необходимых мест и упрощая планирование рабочих процессов [5].

Это приложение представлено на рисунке 1.



Рисунок 1 – Пример внешнего вида приложения

Ниже приведены особенности приложения:

- основные особенности такого программного обеспечения могут включать;
- интерактивные карты промышленной территории с возможностью выбора маршрута;
- расчет оптимальных маршрутов движения с учетом текущей загрузки и трафика;
- возможность интеграции с другими системами управления предприятием для синхронизации данных о местонахождении сотрудников;
- функции для администраторов, позволяющие управлять доступом к различным частям территории и назначать специальные маршруты для отдельных категорий пользователей.

Такое программное обеспечение способствует повышению безопасности на производстве, снижению времени простоя оборудования и улучшению условий труда персонала. Разработка и внедрение подобных решений являются важной частью стратегии цифровой трансформации промышленных предприятий, направленной на повышение их конкурентоспособности и эффективности.

1.2.2 Программное обеспечение интерактивной навигации «LOGICMAP» от «Zorgtech»

Аэропорты, торговые центры, гостиницы, больницы, университеты и школы посещает множество людей, которые не знакомы с маршрутами внутри зданий и территорий. Интерактивная навигация помогает посетителям сориентироваться в больших пространствах. Системы интерактивной навигации имеют ряд преимуществ по сравнению со статическими картами и указателями, например, быстрое обновление карт, размещение дополнительной информации и рекламы. Программное обеспечение Zorgtech предлагает простое и интуитивно понятное комплексное решение, которое может быть установлено в общественных местах со сложной планировкой.

Интерактивная навигация «LOGICMAP» от компании Zorgtech позволяет посетителям с лёгкостью ориентироваться в торговом центре, служит для информирования об акциях, скидках и специальных предложениях. «LOGICMAP» отображает подробные описания и полный перечень распродаж в онлайн-режиме. Решение работает на цифровых носителях: сенсорных столах, информационных киосках, интерактивных панелях и стойках. Интерактивная навигация «LOGICMAP» имеет простую систему управления и редактирования информации [6].

Функциональные возможности программного обеспечения интерактивной навигации «LOGICMAP»:

- подробная интерактивная карта всех этажей с указанием магазинов, мест обслуживания и других необходимых точек интересов;
- функциональная система поиска маршрута к пункту назначения, интерактивный список точек интересов, отсортированный по категориям;
- поиск с учетом вариаций названий, ошибок и опечаток, а также поиск, по ключевым словам;
- современный привлекательный сенсорный интерфейс с элементами анимации;
- вспомогательная информация, включающая описание магазинов и интерактивные баннеры;
- отображение контекстной рекламы в виде слайд-шоу, анимации или видео;
- поиск необходимой информации;
- дружелюбный интерфейс администрирования: пошаговое выполнение действий позволяет быстро и эффективно работать в режиме редактирования, а также обеспечивает простое удаление и добавление информации;
- централизованное управление данными на всех терминалах одновременно;
- сбор статистики по различным показателям (выбор товарных групп, брендов, времени запросов и т. д.

Представление программного продукта изображено на рисунке 2.



Рисунок 2 – Пример внешнего вида приложения

«LOGICMAP» от «Zorgtech» является мощным инструментом для специалистов и организаций, работающих с географической информацией и требующих гибкого решения для создания и управления интерактивными картами и навигационными системами.

1.2.3 Система indoor навигации от компании «Навигационные решения»

Система навигации внутри помещений (indoor навигация) – это современная технология, которая помогает ориентироваться посетителям внутри зданий. Платформы для ее работы базируются на определении местоположения человека или объекта. Для этого в разных местах помещений устанавливают специальные маячки (биконы) или метки, которые с определенными интервалами рассылают радиосигналы. Как только человек с мобильным приложением появляется в зоне их действия, датчики тут же определяют его местоположение и отправляют сигналы с соответствующей информацией на считывающую систему.

Система indoor навигации может быть интегрирована в смартфоны на базе IOS или Android с предустановленным приложением. Для ее реализации используются различные технологии – Bluetooth Low Energy, Wi-Fi и другие, позволяющие определять местонахождение с точностью до 1–5 метров. Indoor навигацию можно применять в любых зданиях, будь то торговые центры, аэропорты, вокзалы, складские или промышленные объекты [7].

Внешний вид приложения Indoor представлен на рисунке 3.

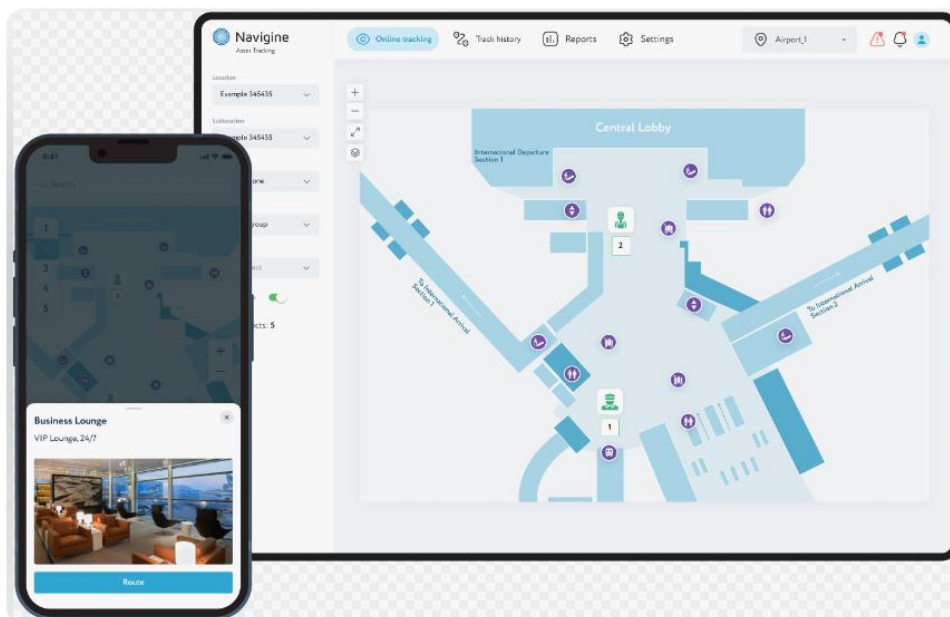


Рисунок 3 – Внешнего вида приложения Indoor

Функциональные возможности программного обеспечения indoor навигации:

- возможность быстро находить необходимые помещения внутри зданий;
- возможность прокладывать наиболее короткие и удобные маршруты;
- возможность получать уведомления с подсказками и полезной информацией;
- возможность получать клиентскую аналитику (на стороне менеджеров компании).

Благодаря системе навигации внутри помещений можно легко ориентироваться в сложных и запутанных зданиях. Пользователю достаточно открыть приложение, чтобы увидеть свое местоположение в режиме реального времени. После выбора пункта назначения приложение строит маршрут, который учитывает заданные параметры и настройки (например, максимальное использование для передвижения лифтов, а не лестниц).

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше систем навигации пользователей представлен в таблице 1.

Таблица 1 – Сравнение аналогов

ПО навигации	Техническое оснащение	Стоимость приложения	Универсальность приложения
ПО навигации персонала на промышленной территории от компании «Университет 20.35»	Информационные доски, интерактивные панели и стойки	Высокая	Разрабатывается под определенное производство
Программное обеспечение интерактивной навигации «LOGICMAP» от компании «Zorgtech»	Сенсорные столы, информационные киоски, интерактивные панели и стойки	Средняя	Может быть установлено в общественных местах со сложной планировкой
Indoor навигация от компании «Навигационные решения»	Необходима установка датчиков	Средняя	Может применяться в объектах с большим количеством помещений
Собственное ПО автоматизированных систем построения маршрута	Нужно подключение к сети интернет, иметь ПК или планшетный компьютер на базе ОС Windows	Низкая	Программное обеспечение может применяться в общественных и производственных местах.

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше систем навигации пользователей.

1.3 Анализ целевой аудитории

Анализ целевой аудитории для темы «программное обеспечение автоматизированной системы построения маршрута движения» включает в себя рассмотрение нескольких ключевых групп пользователей и их потребностей:

- компании транспортной отрасли;
- городские муниципалитеты и государственные организации;
- предприятия доставки;

- частные лица и малый бизнес;
- работники служб поддержки и технического обслуживания;
- исследовательские и образовательные учреждения.

Компании транспортной отрасли - перевозчики, логистические компании, автобусные и железнодорожные компании, которые нуждаются в эффективных алгоритмах для планирования маршрутов, оптимизации времени в пути и снижения затрат на топливо.

Городские муниципалитеты и государственные организации - местные власти могут использовать такое ПО для управления городским транспортом, планирования маршрутов общественного транспорта, учета трафика и улучшения качества жизни граждан.

Предприятия доставки - компании, занимающиеся доставкой товаров (например, курьерские службы), также могут найти применение в автоматизированных системах построения маршрутов для оптимизации доставок и повышения эффективности работы.

Частные лица и малый бизнес - индивидуальные пользователи или малые предприятия, которым необходимо регулярно перемещаться по большим расстояниям или осуществлять доставку товаров, могут использовать такие системы для планирования своих маршрутов, экономии времени и ресурсов.

Работники служб поддержки и технического обслуживания - специалисты, для которых важно иметь доступ к инструментам, позволяющим быстро находить оптимальные маршруты для выполнения задач, особенно если они связаны с обслуживанием оборудования или инвентаря.

Исследовательские и образовательные учреждения - университеты, научные центры и исследовательские группы, которые могут использовать системы для моделирования и анализа различных сценариев движения, изучения влияния различных факторов на эффективность маршрутизации.

Ключевые потребности рассмотренной целевой аудитории включают:

- эффективность маршрутизации;
- удобство использования;

– интерактивность.

Эффективность маршрутизации - оптимизация маршрутов для минимизации времени в пути и затрат.

Удобство использования - простота интерфейса и возможность быстрого создания и изменения маршрутов.

Интерактивность - возможность интеграции с другими системами (например, GPS, картографическими сервисами) для получения актуальной информации о дорожном состоянии и пробках.

Таким образом, разработка программного обеспечения для автоматизированной системы построения маршрута движения должна учитывать эти потребности, чтобы предложить продукт, который будет востребован и полезен для широкой аудитории.

На основе статистической информации, можно представить следующую диаграмму с процентным соотношением установленных операционных систем на стационарных компьютерах пользователей (рисунок 4).



Рисунок 4 – Диаграмма использования ОС

Эта диаграмма отражает текущее распределение основных операционных систем (ОС) среди пользователей [8 – 12]. Далее мы можем увидеть подробную информацию об ОС:

- Windows 10 продолжает оставаться самой популярной операционной системой, установлена на 69.89% компьютеров;
- Windows 7, хотя и уступает Windows 10, всё ещё широко используется, составляя 2,81% от общего числа установок;
- Windows 11 недавно начала набирать популярность и установлена на 26,19% компьютеров;
- macOS установлена на 5,03% компьютеров;
- Windows 8.1 составляет 0,69% от общего числа установок;
- Linux системы, несмотря на свою относительно низкую долю (2,33%), продолжают набирать популярность, особенно среди тех, кто ценит открытость и гибкость этих систем.

Это распределение важно учитывать при разработке программного обеспечения, поскольку выбор определенной операционной системы может существенно повлиять на доступность и удобство использования продукта для потенциальных пользователей.

Можно увидеть, что более 70% пользователей используют операционные системы Windows, и большинство из них используют версию 10 и выше. Поэтому приложение будет разрабатываться таким образом, что оно успешно функционировало для ОС Windows 10 и более новых версий системы.

1.4 Анализ функциональных требований

Поскольку основным назначением ПО «автоматизированной системы построения маршрута движения» является работа с построением маршрута движения по карте возможных путей, именно путь будет основным объектом системы. Для каждого пути хранится его название и карта движения. Отдельно хранится список путей, прикрепленный к пользователю.

В системе должно быть два основных типа пользователей: обычный пользователь и администратор. Обычный пользователь должен иметь возможность построения маршрута, задать точку начала и конца для построения маршрута, так же удаление точек конца и начала. Также каждый

такой пользователь может сохранять или удалять маршруты в случае необходимости.

Администратор должен иметь все возможности, предусмотренные для обычных пользователей. Кроме того, администратор должен быть способен добавлять или удалять карты, находящиеся в базе данных.

В результате анализа требований задания была создана диаграмма вариантов использования (рисунок 5) [16].



Рисунок 5 – Диаграмма вариантов использования программного обеспечения

Составим спецификации вариантов использования «Зарегистрироваться» (рисунок 6), «Зайти в учетную запись» (рисунок 7), «Сохранить маршрут» (рисунок 8), «Удалить маршрут» (рисунок 9). «Построить маршрут» (рисунок 10).

Название: зарегистрироваться
Назначение: регистрация нового аккаунта пользователя
Актеры: пользователь, СУБД
Предусловие: программа запущена
Основной сценарий: <ol style="list-style-type: none"> 1. Пользователь выбирает регистрацию 2. Пользователь вводит логин и пароль. 3. Система проверяет логин и пароль на корректность. 4. Если логин и пароль корректны. <ol style="list-style-type: none"> 4.1. Система запрашивает сведения об аккаунте с введенными данными. 4.2. СУБД передает сведения о наличии аккаунта с введенным логином. 4.3. Если такого аккаунта нет <ol style="list-style-type: none"> 4.3.1. СУБД добавляет в БД информацию о новом пользователе. 4.3.2. Система осуществляет вход нового пользователя. 4.4. Иначе выводится сообщение о том, что выбранный логин уже занят. 5. Иначе система выводит сообщение о недопустимом логине или пароле.
Постусловие: регистрация выполнена

Рисунок 6 – Сценарий варианта использования «Зарегистрироваться»

Название: Зайти в учетную запись
Назначение: Вход на существующий аккаунт пользователя
Актеры: пользователь, СУБД
Предусловие: программа запущена
Основной сценарий: <ol style="list-style-type: none"> 1. Пользователь выбирает авторизацию 2. Пользователь вводит логин и пароль. 3. Система запрашивает сведения об аккаунте с введенными данными. 4. СУБД передает сведения о наличии аккаунта с введенными данными. 5. Если такой аккаунт существует <ol style="list-style-type: none"> 4.3.2. Система осуществляет вход нового пользователя. 4.4. Иначе система выводит сообщение, что выбранный логин уже занят. 6. Иначе система выводит сообщение о неверном логине или пароле.
Постусловие: вход в аккаунт выполнен

Рисунок 7 – Сценарий варианта использования «Зайти в учетную запись»

Название: Сохранить маршрут
Назначение: сохранение выбранного маршрута
Актеры: пользователь, СУБД
Предусловие: пользователь авторизован
Основной сценарий: 1. Пользователь выбирает точку начала и конца. 2. Система передает параметры серверу БД. 3. СУБД сохраняет поступившие параметры. 4. Система выводит информацию о маршруте.
Постусловие: поиск успешно завершен

Рисунок 8 – Сценарий варианта использования «Сохранить маршрут»

Название: Удалить маршрут
Назначение: удаление маршрута
Актеры: пользователь, СУБД
Предусловие: существует сохраненный маршрут
Основной сценарий: 1. Пользователь выбирает сохраненный маршрут. 2. Пользователь удаляет маршрут. 3. Система отправляет запрос на сервер БД для удаления.
Постусловие: Удаление маршрута завершено

Рисунок 9 – Сценарий варианта использования «Удалить маршрут»

Название: Построить маршрут
Назначение: построение маршрута.
Актеры: пользователь, СУБД
Предусловие: выбрана начальная и конечная точка
Основной сценарий: 1. Пользователь выбирает точку начала и конца. 2. Система строит путь из точки начала в точку конца
Постусловие: построение пути завершено

Рисунок 10 – Сценарий варианта использования «Построить маршрут»

Рассмотрим некоторые варианты использования поподробнее. Для варианта использования «Авторизоваться» составлена диаграмма деятельности, представленная на рисунке 11.

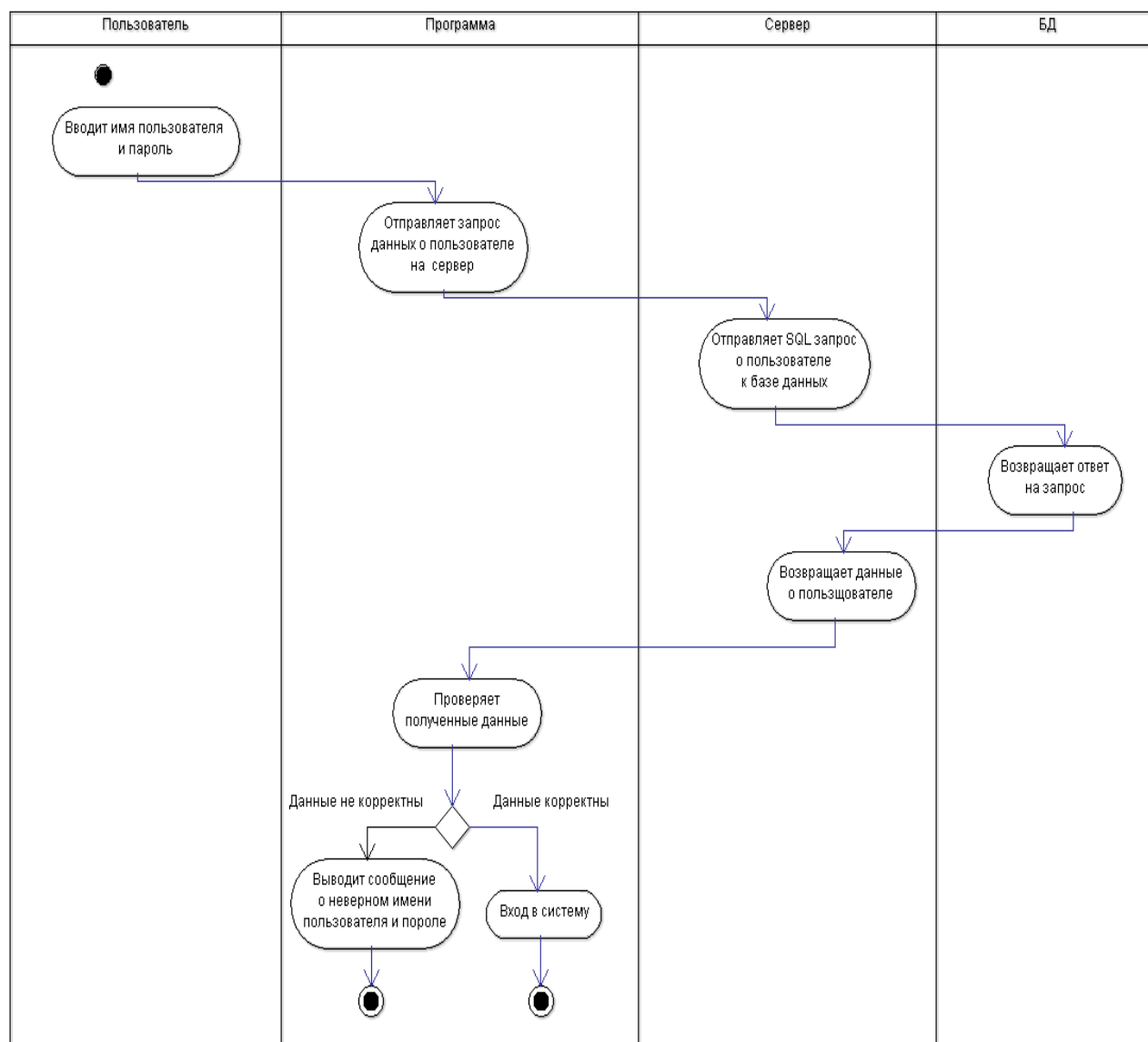


Рисунок 11 – Диаграмма деятельности варианта использования «Авторизоваться»

Данная диаграмма деятельности отображает представление поэтапных действий в системе авторизации пользователя.

Для варианта использования «Построения маршрута» была составлена диаграмма последовательности, представленная на рисунке 12.

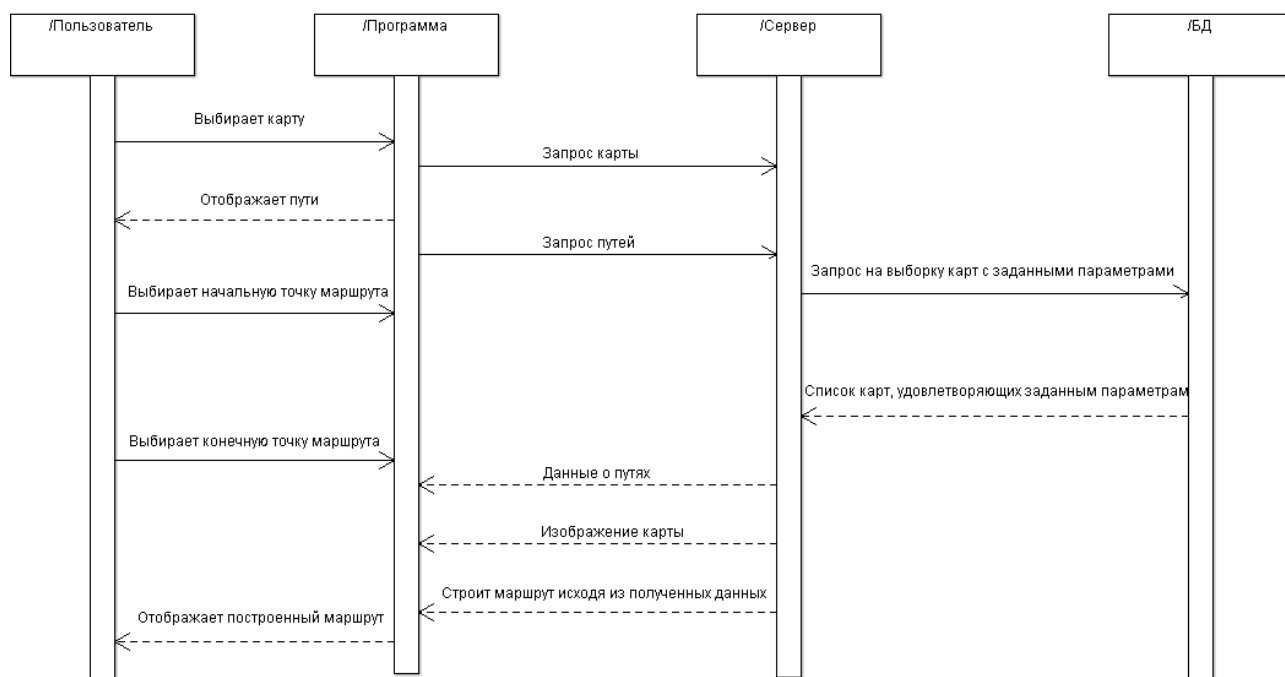


Рисунок 12 – Диаграмма последовательности варианта использования
«Построения маршрута»

В результате анализа функциональных требований были успешно показаны основные функции разрабатываемого программного обеспечения. Также для этого ПО были построены диаграммы вариантов использования, деятельности и последовательности.

1.5 Анализ нефункциональных требований

Разработка серверной части приложения предполагает использование языка программирования C# в рамках среды разработки Visual Studio 2022. На сервере для хранения данных применяется база данных, созданная с помощью СУБД MySQL Server, где все запросы к данным формируются с использованием языка SQL.

Клиентская сторона приложения разрабатывается на языке C#, используя Visual Studio 2022 как среду разработки. Цель состоит в создании приложения с дружелюбным и легко понятным пользовательским интерфейсом. Приложение должно быть протестировано и способно корректно функционировать на устройствах с операционной системой Windows 10 и выше.

1.6 Планирование разработки и оценка бюджета

Грамотное планирование является ключевым фактором для успешного осуществления любых проектов, включая те, что связаны с информационными технологиями. Без него невозможно добиться эффективного выполнения задач, необходимого для достижения поставленных целей в оптимальные сроки, с наименьшими расходами и соответствующим уровнем качества [13 – 17].

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

Таблица 2 – Перечень задач проекта

Номер	Название задачи	Длительность	Предшественники
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ приложений-аналогов	7 дней	1
4	Анализ вариантов построения оптимального маршрута	10 дней	3
5	Консультация с руководством предприятия	2 дня	4
6	Принятие окончательного решения	1 день	5
7	Анализ требований завершен	-	6
8	Выбор средств реализации проекта	-	-
9	Анализ возможных программных средств	4 дня	1
10	Выбор программных средств для реализации приложения	3 дня	9
11	Выбор средств завершен	-	10
12	Проектирование приложения	-	-
13	Проектирование базы данных	5 дней	7; 11
14	Проектирование архитектуры ПО	3 дня	13
15	Проектирование интерфейса пользователя	5 дней	7; 11
16	Проектирование ПО завершено	-	14; 15
17	Реализация проекта	-	-
18	Реализация базы данных	10 дней	13
19	Реализация серверной части приложения	15 дней	16;18
20	Реализация клиентской части приложения	20 дней	16;18
21	Реализация проекта завершена	-	19;20
22	Отладка	-	-
23	Тестирование работы серверной части приложения	5 дней	21

Продолжение таблицы 2

Номер	Название задачи	Длительность	Предшественники
24	Тестирование работы клиентской части приложения	7 дней	21
25	Исправление ошибок	10 дней	23; 24
26	Отладка завершена	-	25
27	Внедрение программного средства	-	-
28	Внесение в систему карт и путей	7 дней	26
29	Обучение сотрудников предприятия работе с системой	7 дней	28
30	Поиск и исправление ошибок при внедрении	10 дней	28
31	Внедрение завершено	-	29;30
32	Конец проекта	-	31

Таким образом, для реализации проекта нужно выполнить 32 задачи.

В таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Список ресурсов проекта

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

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

Номер	Название задачи	Ресурс	Единицы
1	Начало проекта		
2	Анализ требований		
3	Анализ приложений-аналогов	Программист1 Компьютер Интернет	50% 1 шт
4	Анализ алгоритмов построения маршрутов	Программист1 Программист2 Программист3 Компьютер Интернет	50% 50% 50% 1 шт
5	Консультация с руководством предприятия	Программист1 Компьютер Интернет	50% 1 шт

Продолжение таблицы 4

Номер	Название задачи	Ресурс	Единицы
6	Принятие окончательного решения	Программист1 Программист2 Программист3	50% 50% 50%
7	Анализ требований завершен		
8	Выбор средств реализации проекта		
9	Анализ возможных программных средств	Программист1 Программист2 Программист3 Компьютер Интернет	50% 50% 50% 1 шт
10	Выбор программных средств для реализации ПО	Программист1 Программист2 Программист3 Компьютер Интернет	50% 50% 50% 1 шт
11	Выбор средств завершен		
12	Проектирование приложения		
13	Проектирование базы данных	Программист1 Программист2 Компьютер Интернет	50% 25% 2 шт
14	Проектирование архитектуры приложения	Программист1 Программист2 Компьютер Интернет	50% 25% 2 шт
15	Проектирование интерфейса пользователя	Дизайнер Программист2 Программист3 Компьютер Интернет	50% 50% 75% 3 шт.
16	Проектирование приложения завершено		
17	Реализация проекта		
18	Реализация базы данных	Программист1 Программист2 Компьютер Интернет	100% 100% 2 шт.
19	Реализация серверной части приложения	Программист1 Программист2 Программист3 Компьютер Интернет	100% 100% 25% 3 шт.

Окончание таблицы 4

Номер	Название задачи	Ресурс	Единицы
20	Реализация клиентской части приложения	Программист1 Программист2 Программист3 Дизайнер Компьютер Мобильное устройство Интернет	25% 25% 100% 50% 4 шт. 2шт.
21	Реализация проекта завершена		
22	Отладка		
23	Тестирование работы серверной части приложения	Программист1 Программист2 Программист3 Компьютер Интернет	75% 75% 25% 3 шт.
24	Тестирование работы клиентской части приложения	Программист1 Программист2 Программист3 Компьютер Мобильное устройство Интернет	25% 25% 75% 3 шт. 1 шт.
25	Исправление ошибок	Программист1 Программист2 Программист3 Компьютер Интернет	75% 75% 75% 3 шт.
26	Отладка завершена		
27	Внедрение программного средства		
28	Внесение в систему электронных копий всех книг библиотечного фонда	Программист2 Программист3 Компьютер	50% 50% 1 шт.
26	Отладка завершена		
27	Внедрение программного средства		
28	Внесение в систему карт и путей	Программист2 Программист3 Компьютер	50% 50% 1 шт.
29	Обучение сотрудников предприятия работе с системой	Программист1 Компьютер Мобильное устройство	75% 1 шт. 1 шт.
30	Поиск и исправление ошибок при внедрении	Программист1 Программист2 Программист3 Компьютер Мобильное устройство Интернет	25% 50% 50% 2 шт. 1 шт.
31	Внедрение завершено		
32	Конец проекта		

Таким образом, реализацией проекта будут заниматься 3 программиста и 1 дизайнер, которым для работы нужны компьютеры, мобильное устройство и доступ к интернету.

Для вычисления длительности реализации проекта и его критического пути построен сетевой график, представленный на рисунке 13.

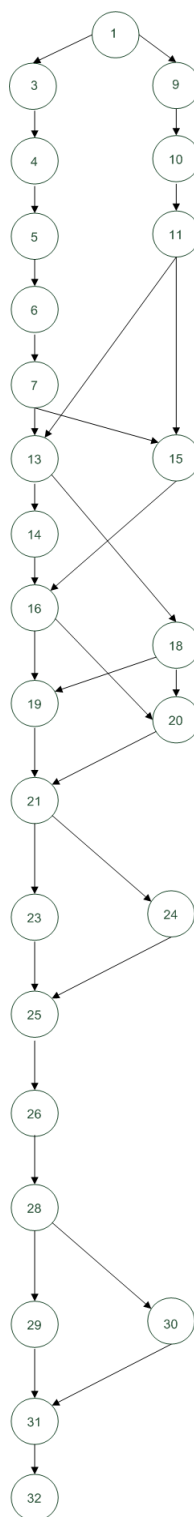


Рисунок 13 – Сетевой график проекта

Рассчитаем вручную критический путь проекта. Критический путь представляет собой последовательность задач, определяющих минимальное время завершения проекта. Все задачи на критическом пути имеют нулевой запас времени, что означает, что любое их задержание приведет к задержке всего проекта. В таблице 5 представлены расчеты раннего времени начала работ для его задач.

Таблица 5 – Расчеты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0
3	1	0	0	0+0	0
4	3	0	7	0+7	7
5	4	7	10	7+10	17
6	5	17	2	17+2	19
7	6	19	1	19+1	20
9	1	0	0	0+0	0
10	9	0	4	0+4	4
11	10	4	3	4+3	7
13	7	20	0	Max(20+0;7+0)	20
	11	7	0		
14	13	20	5	20+5	25
15	7	20	0	Max(20+0;7+0)	20
	11	7	0		
16	14	25	3	Max(25+3;20+5)	28
	15	20	5		
18	13	20	5	20+5	25
19	16	28	0	Max(28+0;25+10)	35
	18	25	10		
20	16	28	0	Max(28+0;25+10)	35
	18	25	10		
21	19	35	15	Max(35+15;35+20)	55
	20	35	20		
23	21	55	0	55+0	55
24	21	55	0	55+0	55
25	23	55	5	Max(55+5;55+7)	62
	24	55	7		
26	25	62	10	62+10	72
28	26	72	0	72+0	72

Продолжение таблицы 5

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
29	28	72	7	72+7	79
30	28	72	7	72+7	79
31	29 30	79 79	7 10	Max(79+7; 79+10)	89
32	31	89	0	89+0	89

В таблице 6 рассчитано позднее время начала работ задач.

Таблица 6 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
32	-	-	-	-	89
31	32	89	0	89-0	89
30	31	89	10	89-10	79
29	31	89	7	89-7	82
28	29 30	82 79	7 7	Min(82-7; 79-7)	72
26	28	72	0	72-0	72
25	26	72	10	72-10	62
24	25	62	7	62-7	55
25	26	72	10	72-10	62
24	25	62	7	62-7	55
23	25	62	5	62-5	57
21	24 23	55 57	0	Min(55-0; 57-0)	55
20	21	55	20	55-20	35
19	21	55	15	55-15	40
18	20 19	35 40	10	Min(35-10; 40-10)	25
16	20 19	35 40	0	Min(35-0; 40-0)	35
15	16	35	5	35-5	30
15	16	35	5	35-5	30
14	16	35	3	35-3	32
13	18 14	25 32	5	Min(25-5; 32-5)	20
11	15 13	30 20	0	Min(30-0; 20-0)	20
10	11	20	4	20-4	16

Продолжение таблицы 6

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
9	10	16	3	16–3	13
7	15 13	30 20	0	Min(30–0;20–0)	20
6	7	20	1	20–1	19
5	6	19	2	19–2	17
4	5	17	10	17–17	7
3	4	7	7	7–7	0
1	3 9	0 13	0	Min(0–0;13–0)	0

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

Таблица 7 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	7	7	0
5	17	17	0
6	19	19	0
7	20	20	0
9	0	13	13
10	4	16	12
11	7	20	13
13	20	20	0
14	25	32	7
15	20	30	10
16	28	35	7
18	25	25	0
19	35	40	5
20	35	35	0
21	55	55	0
23	55	57	2
24	55	55	0
25	62	62	0
26	72	72	0
28	72	72	0
29	79	82	3

Продолжение таблицы 7

Работа №	Раннее начала	время	Позднее начала	время	Резерв времени
30	79		79		0
31	89		89		0
32	89		89		0

На основе полученных данных можно увидеть, что срок равен 89 дням в итоге представить критический путь. Также можно увидеть, что критический путь проходит через работы с номерами 1, 3, 4, 5, 6, 7, 13, 18, 20, 21, 24, 25, 26, 28, 30, 31, 32. На рисунке 14 представлен граф проекта, на котором штриховыми линиями указан критический путь.

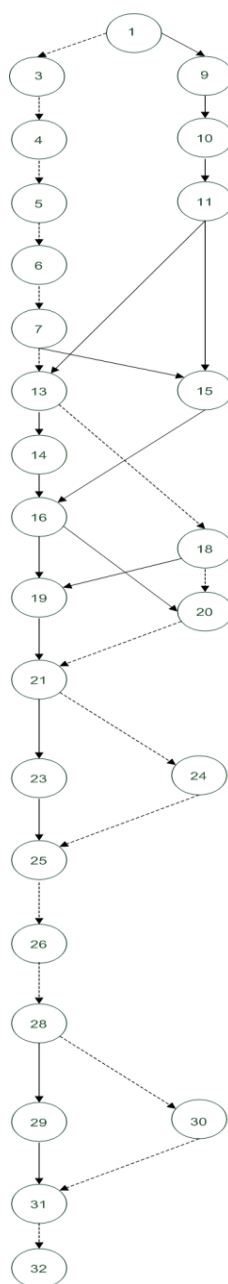


Рисунок 14 – Критический путь проекта

Для того, чтобы рассчитать продолжительность проекта, а также вычислить величину его бюджета, воспользуемся системой управления проектами Microsoft Project, которая позволяет автоматически произвести все необходимые вычисления [18 – 21]. На рисунке 15 представлен список задач из таблицы 1 внесенный в Microsoft Project. Началом проекта была принята дата 07.12.2023.

	Название задачи ▼	Длительно ▼	Начало ▼	Окончание ▼	Предшественники ▼
1	Начало проекта	0 дней	Чт 07.12.23	Чт 07.12.23	
2	▢ Анализ требований	20 дней	Ср 07.12.23	Пн 01.01.24	
3	Анализ приложений-аналогов	7 дней	Ср 07.12.23	Пт 15.12.23	1
4	Анализ вариантов построения оптимального маршрута	10 дней	Ср 16.12.23	Вт 29.12.23	3
5	Консультация с руководством предприятия	2 дней	Ср 30.12.23	Чт 31.12.23	4
6	Принятие окончательного решения	1 день	Пт 01.01.24	Пт 01.01.24	5
7	Анализ требований завершен	0 дней	Пт 01.01.24	Пт 01.01.24	6
8	▢ Выбор средств реализации проекта	7 дней	Ср 07.12.23	Вт 15.12.23	
9	Анализ возможных программных средств	4 дней	Ср 07.12.23	Чт 10.12.23	1
10	Выбор программных средств для реализации приложения	3 дней	Пт 11.12.23	Вт 15.12.23	9
11	Выбор средств завершен	0 дней	Вт 15.12.23	Вт 15.12.23	10
12	▢ Проектирование приложения	8 дней	Пн 04.01.24	Ср 13.01.24	
13	Проектирование базы данных	5 дней	Пн 04.01.24	Пт 08.01.24	7;11
14	Проектирование архитектуры ПО	3 дней	Пн 11.01.24	Ср 13.01.24	13
15	Проектирование интерфейса пользователя	5 дней	Пн 04.01.24	Пт 08.01.24	7;11
16	Проектирование приложения завершено	0 дней	Ср 13.01.24	Ср 13.01.24	14;15
17	▢ Реализация проекта	30 дней	Пн 11.01.24	Пт 19.02.24	
18	Реализация базы данных	10 дней	Пн 11.01.24	Пт 22.01.24	13
19	Реализация серверной части приложения	15 дней	Пн 25.01.24	Пт 12.02.24	16;18
20	Реализация клиентской части приложения	20 дней	Пн 25.01.24	Пт 19.02.24	16;18
21	Реализация проекта завершена	0 дней	Пт 19.02.24	Пт 19.02.24	19;20
22	▢ Отладка	17 дней	Пн 22.02.24	Вт 16.03.24	
23	Тестирование работы серверной части ПО	5 дней	Пн 22.02.24	Пт 26.02.24	21
24	Тестирование работы клиентской части ПО	7 дней	Пн 22.02.24	Вт 02.03.24	21
25	Исправление ошибок	10 дней	Ср 03.03.24	Вт 16.03.24	23;24
26	Отладка завершена	0 дней	Вт 16.03.24	Вт 16.03.24	25
27	▢ Внедрение программного средства	17 дней	Ср 17.03.24	Чт 08.04.24	
28	Внесение в систему карт и путей	7 дней	Ср 17.03.24	Чт 25.03.24	26
29	Обучение сотрудников предприятия	7 дней	Пт 26.03.24	Пн 05.04.24	28
30	Поиск и исправление ошибок при внедрении	10 дней	Пт 26.03.24	Пн 08.04.24	28
31	Внедрение завершено	0 дней	Пн 08.04.24	Пн 08.04.24	29;30
32	Конец проекта	0 дней	Пн 08.04.24	Пн 08.04.24	31

Рисунок 15 – Задачи проекта в среде Microsoft Project

На рисунке 16 представлена диаграмма Ганта, построенная в среде Microsoft Project на основе введенных данных.

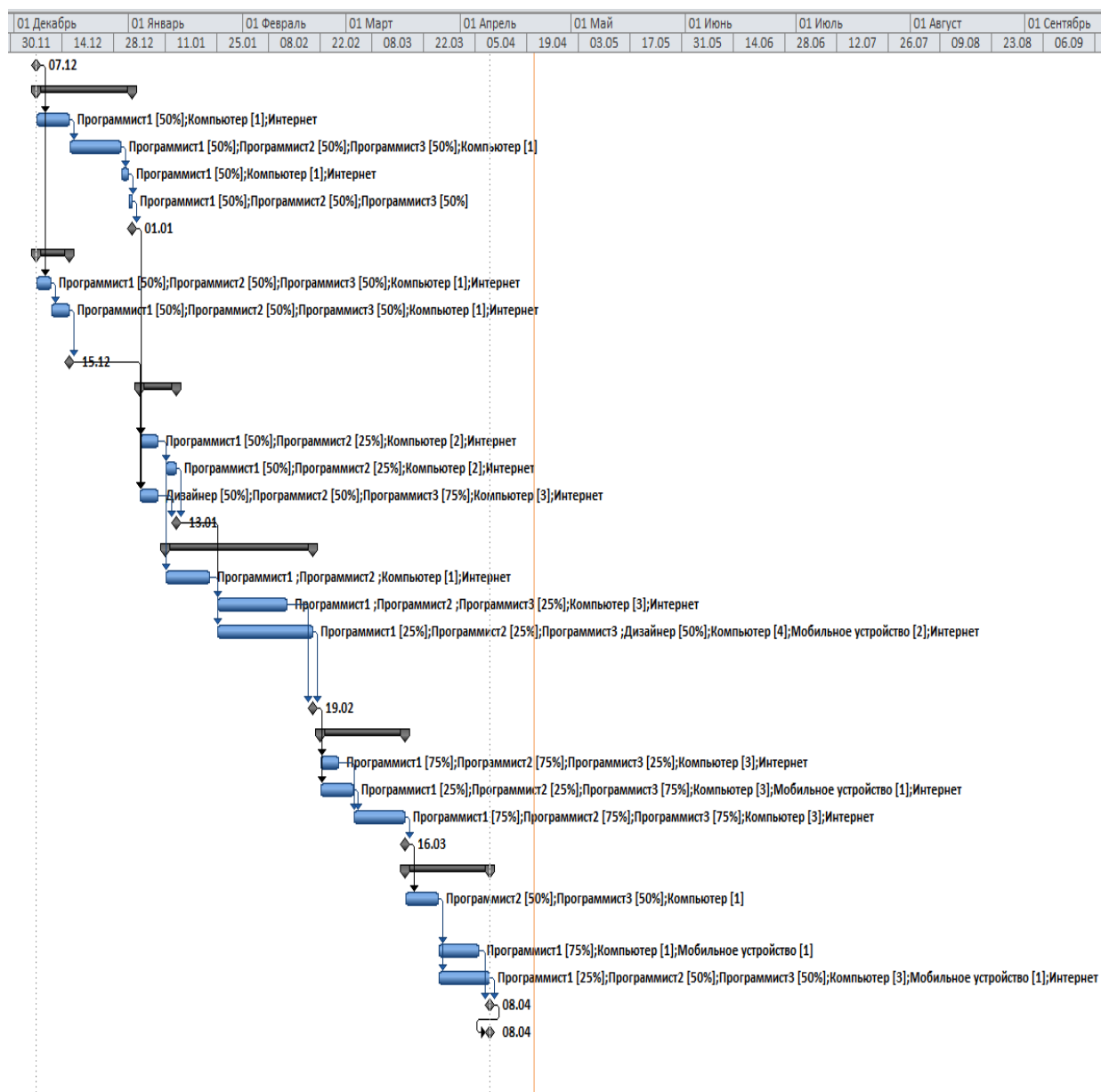


Рисунок 16 – Диаграмма Ганта

На ее основании можно понять, что критический путь проходит через задачи 1, 3, 4, 5, 6, 7, 13, 18, 20, 21, 24, 25, 26, 28, 30, 31, 32. Можно заметить, что вычисленный программой критический путь совпал с критическим путем, вычисленным вручную.

На рисунке 17 представлен обзор проекта. Из рисунка видно, что общая длительность проекта составляет 89 дней (что совпадает со значением, вычисленным вручную), а предполагаемые затраты – 1 632 100 руб.

НИР

на дату Вс 21.04.23

<u>Даты</u>			
Начало:	Чт 07.12.23	Окончание:	Пн 08.04.24
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
<u>Длительность</u>			
Текущий план:	89 дней	Оставшаяся:	89 дней
Базовый план:	0 дней	Фактическая:	0 дней
Отклонение:	89 дней	Процент завершени:	0%
<u>Трудозатраты</u>			
Текущий план:	1 551,6 ч	Оставшиеся:	1 551,6 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 551,6 ч	Процент завершени:	0%
<u>Затраты</u>			
Текущий план:	1 632 100,00 Р	Оставшиеся:	1 632 100,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	1 632 100,00 Р		
<u>Состояние задачи</u>		<u>Состояние ресурса</u>	
Еще не начатые задачи	32	Трудовые ресурсы:	2
Выполняющиеся задач	0	Трудовые ресурсы с прев. доп	3
Завершенные задачи:	0	Материальные ресурсы:	2
Итого задач:	32	Итого ресурсов:	7

Рисунок 17 – Обзорный отчет проекта

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

2 Проектирование программного обеспечения

2.1 Выбор технологий и инструментальных средств разработки

При разработке приложения была выбрана методология RUP, которая предусматривает итеративный подход. На каждой итерации устанавливаются конкретные цели, достижение которых требуется к её завершению. В результате, после каждой итерации получается промежуточная версия ПО. Методология RUP включает четыре основные фазы, каждая из которых может содержать одну или несколько итераций:

- фаза начала проекта – формирование общего представления о проекте и экономического обоснования;
- уточнение – анализ предметной области и проектирование архитектуры продукта;
- построение – реализация основного функционала продукта;
- внедрение – получение окончательной версии продукта и его передача заказчику, включающее бета-тестирование, исправление обнаруженных ошибок и обучение пользователей.

Каждая фаза заканчивается вехой, на которой команда оценивает выполнение задач и принимает ключевые решения о будущем направлении работы над продуктом.

Выбор методологии RUP обусловлен её преимуществами, такими как гибкость процесса разработки, возможность быстрых корректировок проекта при необходимости и выявление основных рисков на ранних этапах. В отличие от некоторых других методологий, например XP, RUP не требует непрерывного взаимодействия с заказчиком, что является важным фактором для данного проекта [22].

На этапе планирования проекта было принято решение использовать приложение Microsoft Project для эффективного управления всеми стадиями проекта. Оно позволяет структурировать проект, составлять календарный план и рассчитывать затраты на ресурсы.

Для анализа требований и моделирования структуры проекта согласно принципам RUP были использованы диаграммы на языке моделирования UML, построенные с помощью программы draw.io [23].

Для создания программы был выбран Visual Studio 2022, среда разработки для создания программ для операционной системы Windows, и язык программирования C#. Серверная часть приложения разрабатывалась на языке PHP с использованием Atom. Для хранения данных использовалась база данных, работающая на СУБД MySQL Server.

2.2 Проектирование архитектуры программного обеспечения

При разработке приложения было выбрано использование двухуровневой архитектуры «клиент-сервер».

В такой архитектуре серверная часть включает в себя базу данных, систему управления базами данных (СУБД) и серверную логику приложения, которая обеспечивает прямое взаимодействие с базой данных и обычно реализуется через API. Клиентская часть приложения предоставляет пользователю интуитивно понятный интерфейс для просмотра и взаимодействия с данными из базы данных, а также позволяет осуществлять удаленное подключение к серверу. В нашем случае, роль клиента выполняет программа.

Обмен между клиентом и сервером происходит следующим образом: клиент отправляет HTTP-запрос на сервер. Серверная часть приложения затем формирует SQL-запрос к базе данных на основе полученного запроса. СУБД обрабатывает этот запрос и возвращает результаты обратно на сервер, который, в свою очередь, формирует ответ для клиента на основе этих результатов. После получения ответа, клиент отображает его пользователю в удобном для восприятия виде.

Основное преимущество такой архитектуры заключается в том, что все данные хранятся на сервере, что позволяет сократить размер клиентского приложения. Также, благодаря независимости серверной части от клиентской,

возможно создание нескольких версий клиентских приложений, что может быть полезно при расширении функционала приложения на другие платформы, кроме Windows.

2.3 Проектирование интерфейса пользователя

Исходя из определенных функциональных требований, был создан интерфейс для приложения, который реализует страницы для входа, регистрации, поиск оптимального маршрута, отображение списка карт, данные о пользователе, страница администрирования.

Основной рабочий экран приложения будет разбит на две части: на область навигации и на область отображения данных. Макет представлен на рисунке 18.

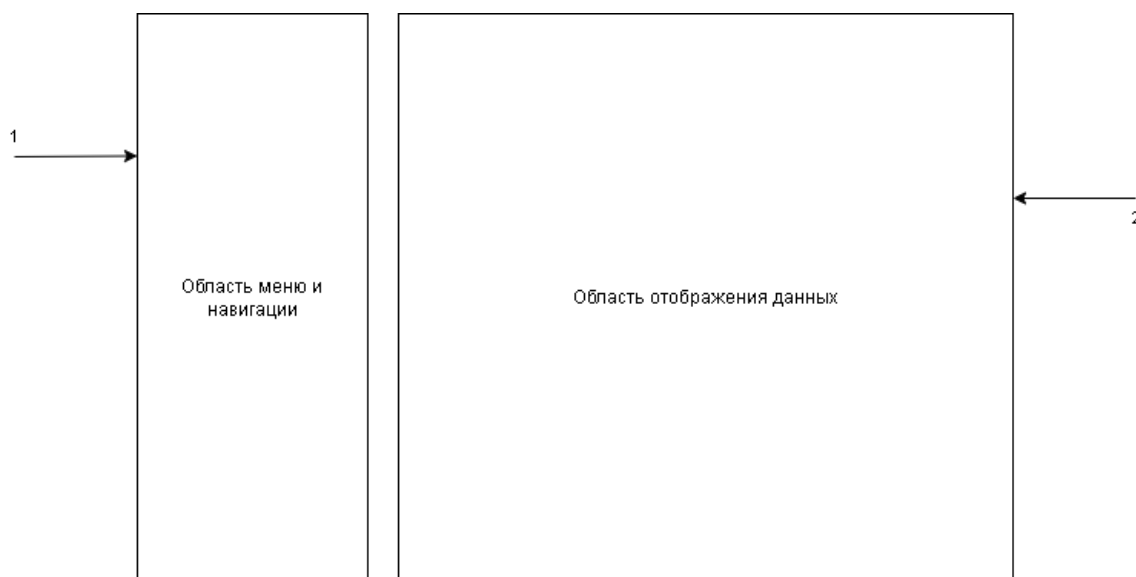


Рисунок 18 – Макет клиентской части

Страница входа, изображенная на рисунке 18, содержит следующие компоненты:

- 1 – меню навигации;
- 2 – отображение данных, при активации кнопки из области меню и навигации будет изменяться область отображения данных.

Сперва будет отображаться форма аутентификации пользователя. Система аутентифицирует пользователя, проверяя наличие учетной записи с

введенным логином и паролем. В случае отсутствия такой учетной записи, пользователю будет показано сообщение об ошибке, связанной с неправильным логином или паролем. Если же пользователь найден, система обновляет область навигации и меню в зависимости от роли пользователя. Макет интерфейса аутентификации представлен на рисунке 19.

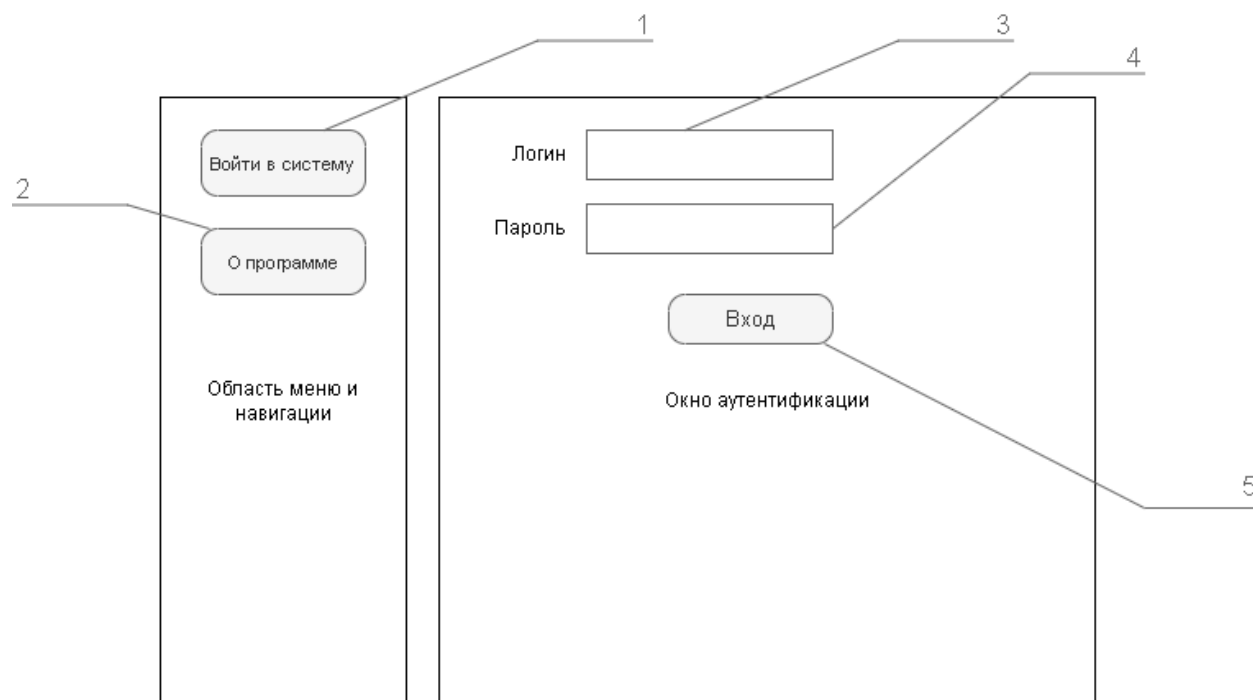


Рисунок 19 – Макет аутентификации

На ней имеются следующие элементы:

- 1 – кнопка перехода на страницу входа;
- 2 – кнопка перехода на страницу «О программе»;
- 3 – поле ввода логина пользователя;
- 4 – поле ввода пароля пользователя;
- 5 – кнопка входа в систему.

При нажатии кнопки «Войти в систему» в области данных появляется форма аутентификации, при нажатии на кнопку «Войти» система проверяет, являются ли введенные логин и пароль допустимыми. Если это условие не выполняется, будет выведено сообщение о недопустимом логине или пароле.

При нажатии на кнопку «О программе» в области отображения данных показывается информация о разработчике программы и ее назначении.

Макет интерфейса страницы о программе представлен на рисунке 20.

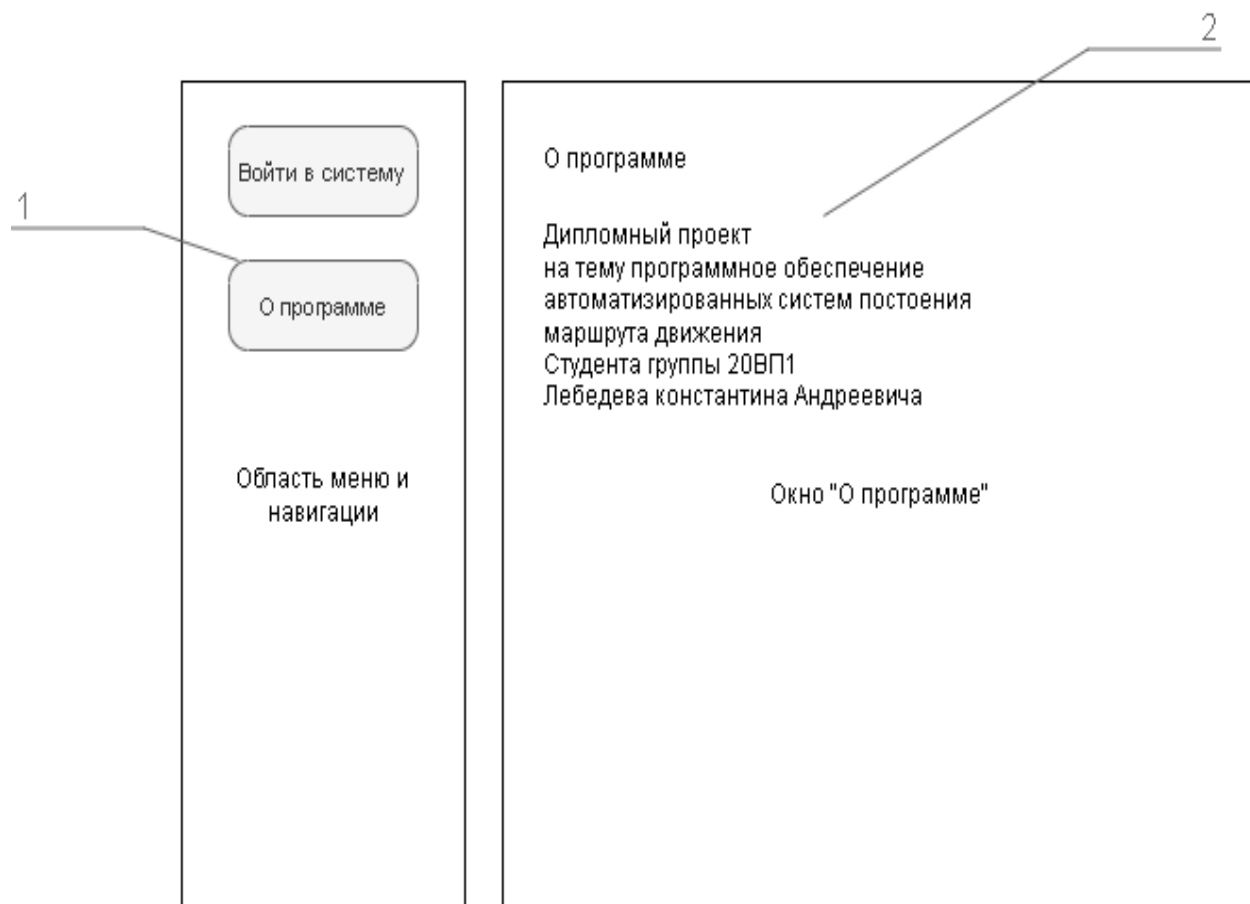


Рисунок 20 – Макет страницы «О программе»

На ней имеются следующие элементы:

- 1 – кнопка перехода на страницу «О программе»;
- 2 – информация о программе и авторе.

При нажатии кнопки «Войти в систему» в области данных появляется форма аутентификации, при нажатии на кнопку «Войти» система проверяет, являются ли введенные логин и пароль допустимыми.

После успешной авторизации в качестве администратора на панели навигации появляется кнопки «Управление картами» и «Выход из системы». При нажатии на эту кнопку в области просмотра данных открывается редактор карт, где можно добавить новую, изменить существующую и удалить.

Макет интерфейса редактирования карт представлен на рисунке 21.

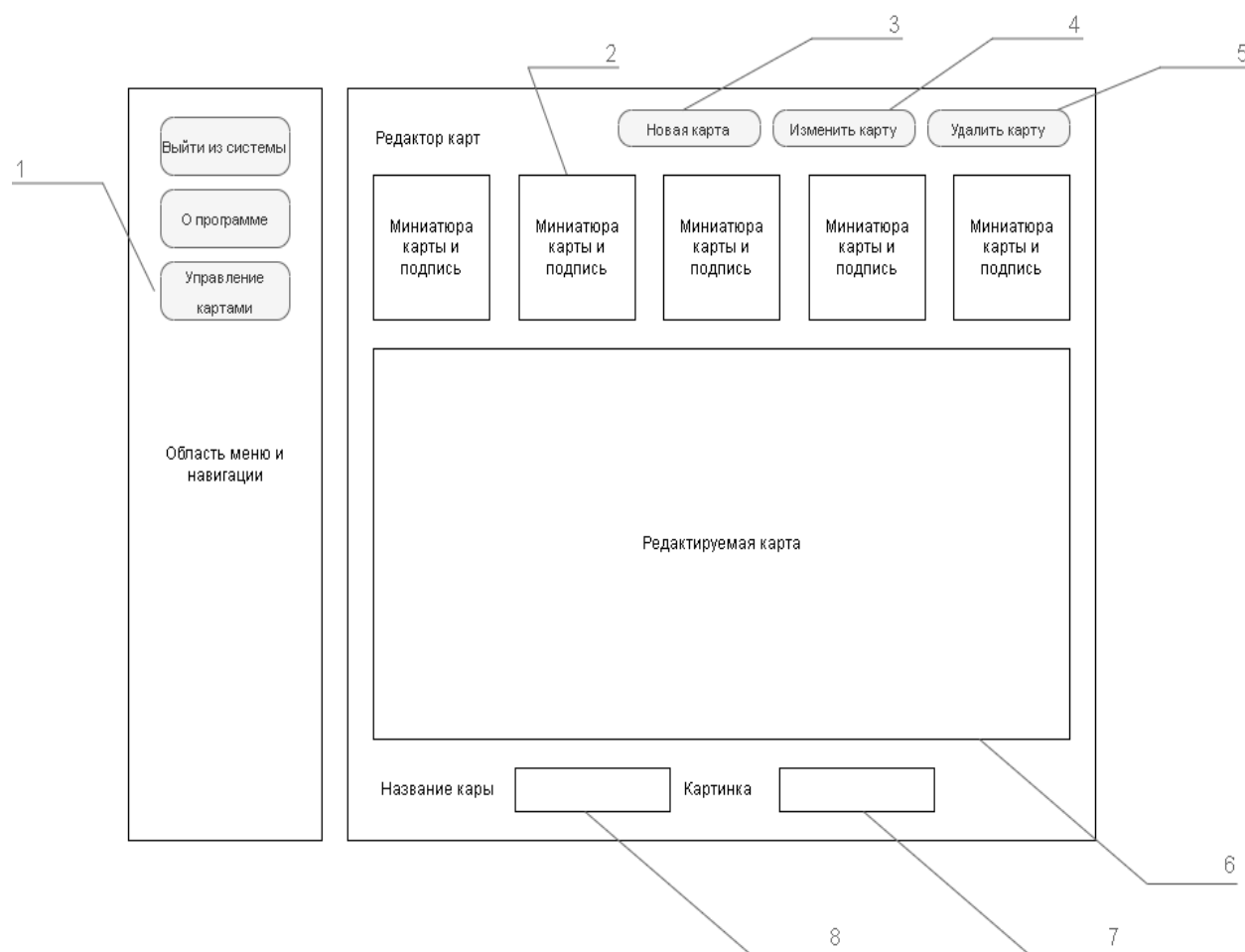


Рисунок 21 – Макет страницы «Редактирования карты»

На форме «Редактора карт», представленном на рисунке 21, присутствуют следующие элементы:

- 1 – кнопка «Управление картами»;
- 2 – отдельные элементы выбора карт;
- 3 – кнопка «Новая карта» для добавления карты;
- 4 – кнопка «Изменить карту» для изменения карты;
- 5 – кнопка «Удалить карту» для текущей удаления карты;
- 6 – редактируемая карта;
- 7 – поле для ввода картинки;
- 8 – поле для ввода названия карты.

После успешной авторизации в качестве обычного пользователя на панели навигации появляется кнопки «Выход из системы», «Выбрать карту» и «Редактировать профиль».

Макет формы пользователя представлен на рисунке 22.

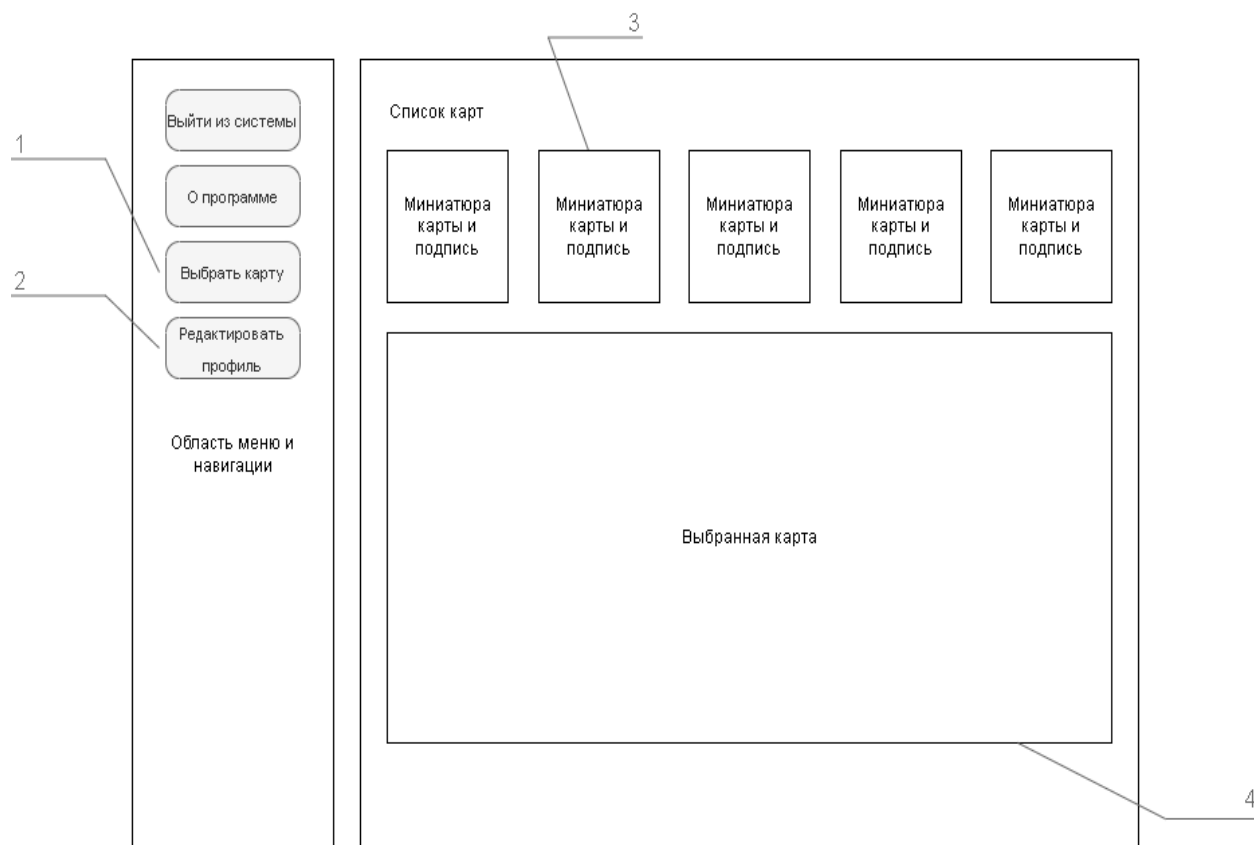


Рисунок 22 – Макет формы пользователя

На форме «Список карт» представленном на рисунке 22, присутствуют следующие элементы:

- 1 – кнопка «Выбрать карту»;
- 2 – кнопка «Редактировать профиль»;
- 3 – кнопки открытия карты;
- 4 – панель с выбранной картой для построения маршрута.

Макет формы редактирования профиля пользователя представлен на рисунке 23.

На форме «Редактора профиля» представленном на рисунке 23, присутствуют следующие элементы:

- 1 – кнопка «Редактора профиля»;
- 2 – поле имя пользователя;
- 3 – поле пароля пользователя;
- 4 – поле для Фамилии пользователя;

- 5 – поле для Имя пользователя;
- 6 – поле для Отчества пользователя;
- 7 – поле для отдела или подразделения;
- 8 – кнопка «Сбросить» для сброса изменений;
- 9 – кнопка «Отправить» для сохранения изменений.

Рисунок 23 – Макет формы редактирования профиля

По окончании этапа макетирования пользовательского интерфейса разрабатывалось графическое приложение.

2.4 Логическое проектирование базы данных

Концептуальная модель базы данных была разработана на основании изучения предметной области и определения функциональных требований. Эта модель представлена с помощью диаграммы «сущность-связь», которая представлена на рисунке 24.

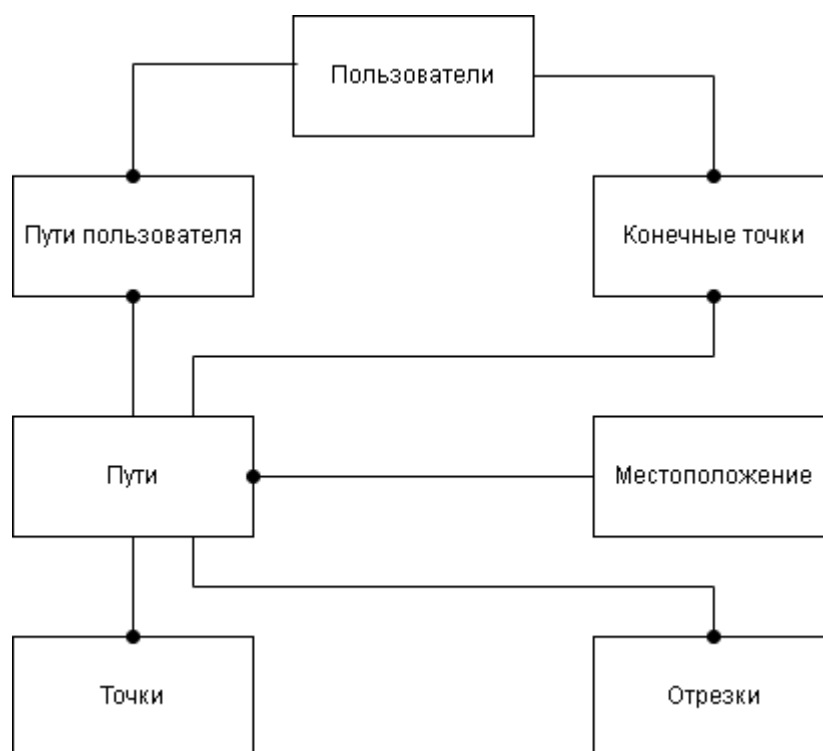


Рисунок 24 – Диаграмма «сущность-связь»

В таблице 8 описаны сущности концептуальной модели данных.

Таблица 8 – Сущности предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Список пользователей системы	Стержневая	10000	В систему добавляются новые пользователи при регистрации
2	Путь	Карта и набор путей на ней	Стержневая	10000	Администраторы могут добавлять пути в систему
3	Пути пользователя	Список путей, сохранённых пользователем	Стержневая	1000	Пользователь может добавить путь в быстрый доступ
4	Конечные точки	Начальная и конечная точки маршрута заданные пользователем	Стержневая	100000	Пользователь может добавить начало и конец маршрута
5	Точка	Составная часть пути с координатами	Стержневая	100000	Пользователь добавляет точку в путь
6	Отрезок	Часть пути, соединяющая точки	Стержневая	100000	Пользователь добавляет отрезок пути
7	Местоположение	Географическое положение пути	Стержневая	100	Пользователь может выбрать географическое местоположение пути

В таблице 9 представлено описание связей между сущностями предметной области.

Таблица 9 – Описание связей между сущностями предметной области

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Пользователь	Пути пользователя	Каждый пользователь может сохранять пути для быстрого доступа/ Каждый путь пользователя указывает на одного пользователя	1:M	Необязательная (пользователь может не иметь сохранённых путей)	
2	Роль пользователя	Пользователь	Каждую роль может иметь множество пользователей/ Каждый пользователь может иметь только одну роль	1:M	Необязательная (роль может не относиться ни к одному пользователю)	
3	Пути	Пути пользователя	Каждый пользователь может сохранить несколько путей/ Каждый сохранённый путь пользователя указывает на один путь	1:M	Необязательная (путь может быть не сохранён для быстрого доступа)	
	Пользователь	Конечные точки	Каждый пользователь может сохранять конечные точки пути / каждая конечная точка указывает на одного пользователя	1:M	Пользователь может не указывать конечные точки маршрута (необязательный)	
	Путь	Конечные точки	Для одного пути может быть указано несколько конечных точек для разных пользователей / каждая конечная точка указывает на один путь	1:M	Пользователь может не указывать конечные точки маршрута (необязательный)	
	Путь	Отрезок	Один путь содержит в себе несколько отрезков / один отрезок принадлежит к одному пути	1:M	Пользователь может не указывать состав пути (необязательный)	
	Путь	Точка	Один путь содержит в себе несколько точек / одна точка принадлежит к одному пути	1:M	Пользователь может не указывать состав пути (необязательный)	
	Местоположение	Путь	Одно местоположение может включать в себя несколько путей	1:M	Местоположение может быть не указано (необязательный)	

На основании концептуальной модели данных в среде Erwin Data Modeler была построена логическая модель базы данных. Ее структура описана диаграммой в нотации IDEF1X, представленной на рисунке 25.

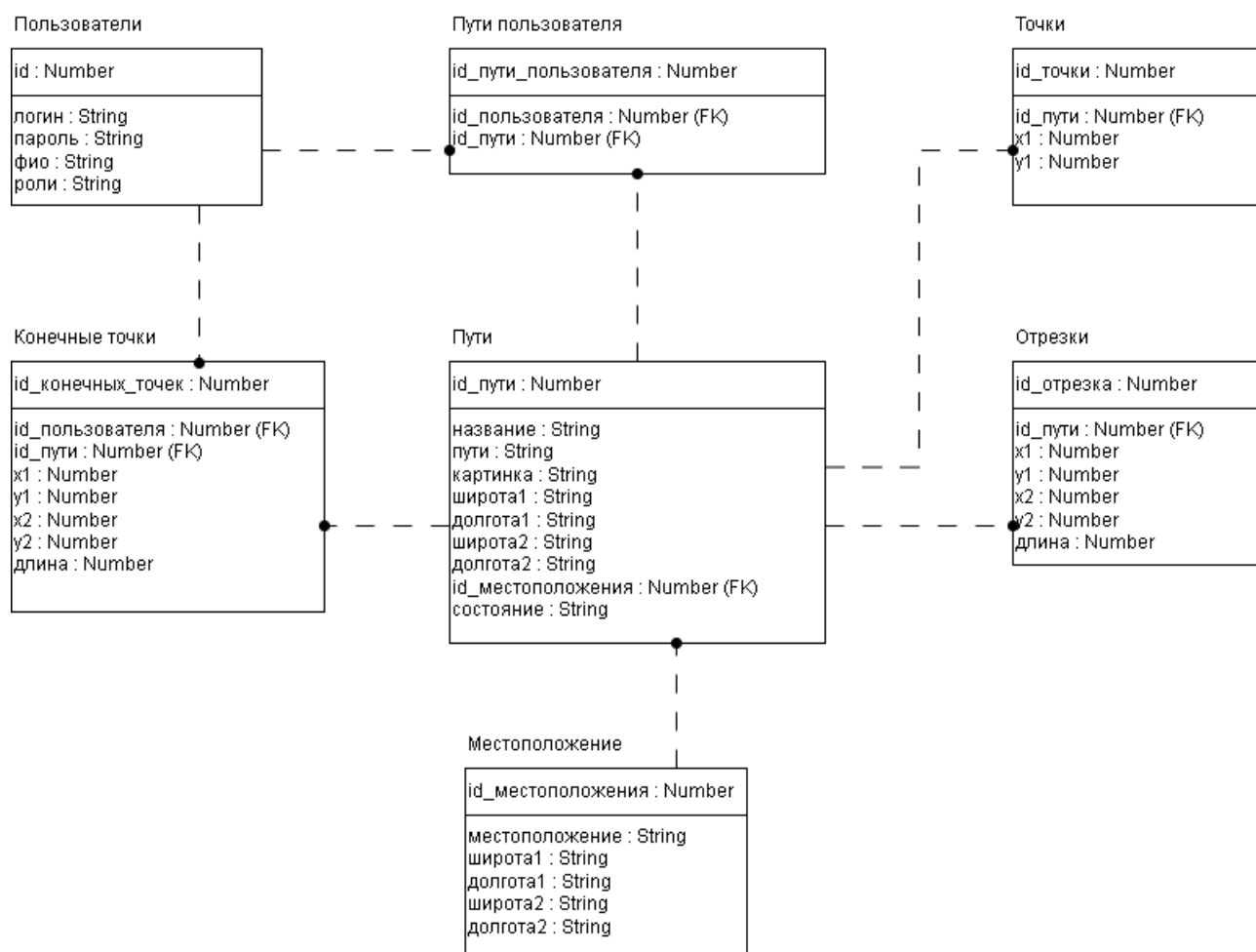


Рисунок 25 – Логическая модель базы данных

На представленной логической модели уже более подробно отображена спроектированная структура БД. Тут имеются не только типы сущностей и связи между ними, но и хранятся данные о конкретных атрибутах сущностей. Тем не менее сущности всё ещё представлены в привязки к конкретной реализации базы данных, а показаны в абстрактной форме.

2.5 Проектирование серверной части приложения

Приложение использует серверную компоненту, основанную на API. Для взаимодействия с данными применяется протокол HTTP/HTTPS, используя его базовые методы: GET для извлечения информации о ресурсах, POST для их создания, PUT для модификации и DELETE для удаления. Ресурсы обращаются через их уникальные URI.

Архитектурный подход приложения основывается на модели MVC (Model-View-Controller), что демонстрируется на рисунке 26.

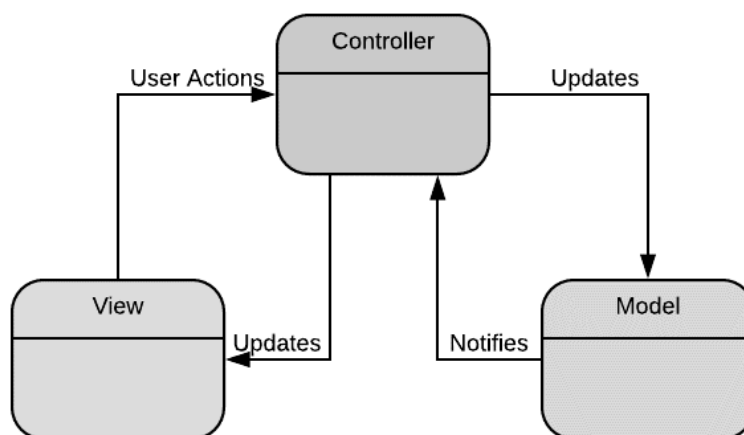


Рисунок 26 – Принцип работы MVC

Серверная составляющая включает классы модели, которые управляют связью с базой данных, и классы контроллера, обеспечивающие обработку запросов от клиентов и формирование соответствующих ответов. Взаимодействие с пользователем обеспечивается уровнем представления, который находится на стороне клиента.

3 Реализация программного обеспечения

3.1 Реализация базы данных

Используя систему Erwin и учитывая особенности СУБД MySQL Server, была разработана физическая модель базы данных на основании предварительно созданной логической модели. Эта модель изображена на рисунке 27.

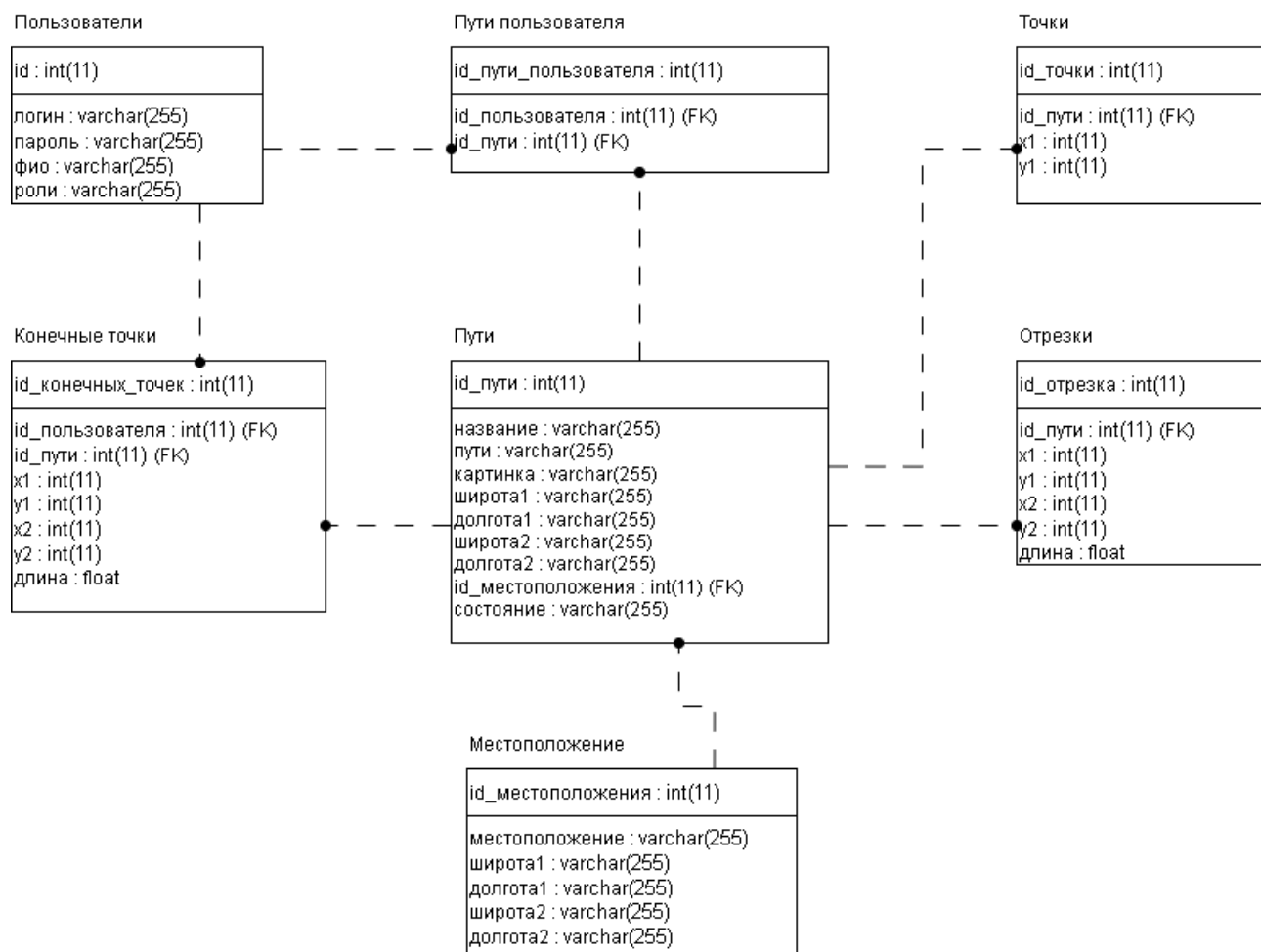


Рисунок 27 – Физическая модель базы данных

Созданный на основе этой физической модели скрипт на языке SQL предназначен для формирования базы данных. Код этого скрипта приведен в приложении Б. Процесс развертывания проектированной базы данных осуществлялся через выполнение указанного скрипта с помощью инструмента phpmyadmin, который применяется для управления базами данных СУБД MySQL Server.

3.2 Разработка серверной части

Разработка серверной части приложения производилась в среде разработки Atom на языке PHP с применением технологии Ajax. Для взаимодействия с базой данных использовалась библиотека php-mysql.

В соответствии с используемой архитектурой MVC все классы серверной части можно разделить на два слоя: слой модели, предназначенный для взаимодействия с БД, и слой контроллеров, предназначенный для обработки HTTP-запросов к серверу [25 – 27].

Диаграмма классов слоя модели представлена на рисунке 28.

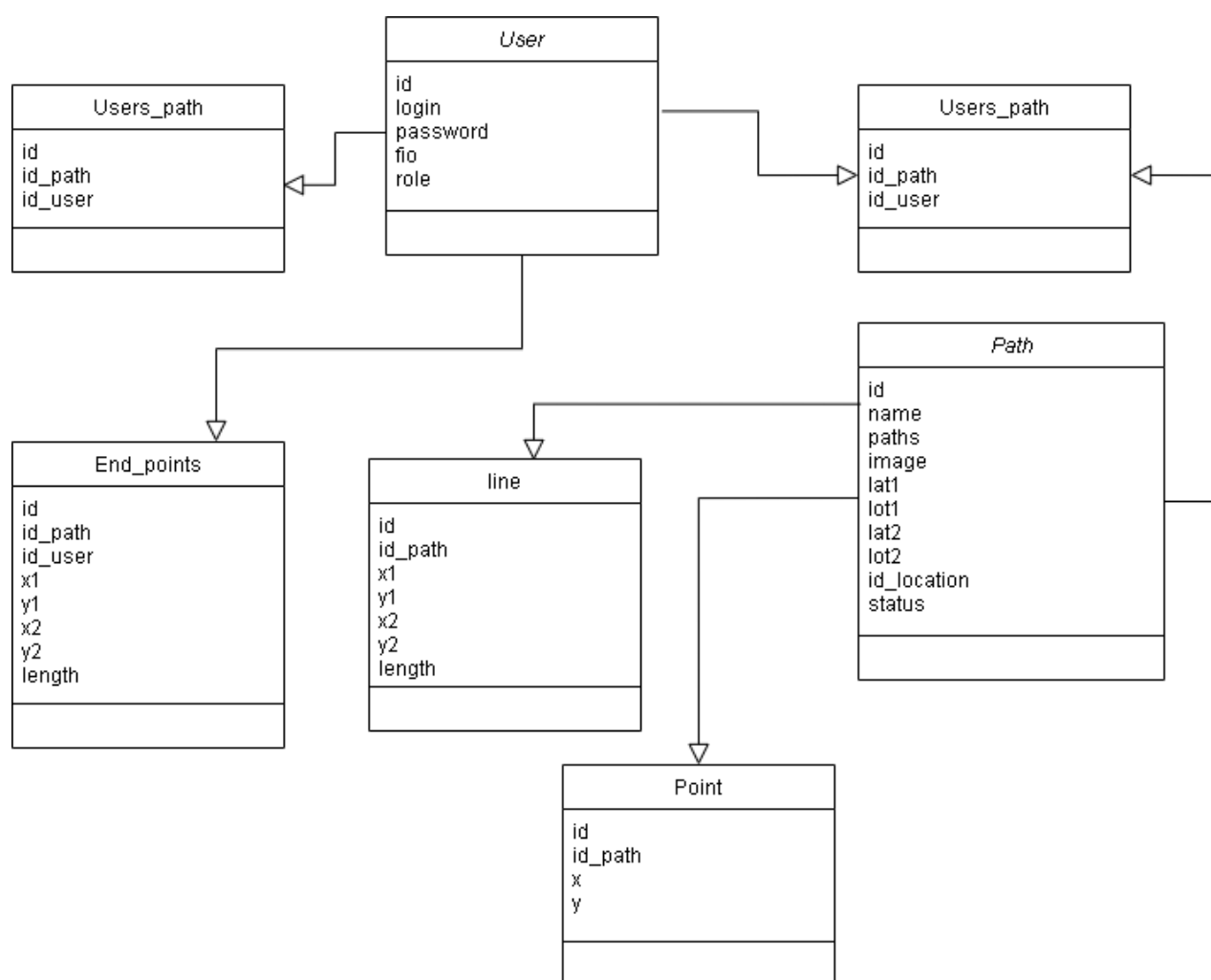


Рисунок 28 – Диаграмма классов серверной части

Как можно заметить, все классы были созданы на основании одноименных таблиц базы данных, что говорит о правильном построении схемы [28, 29].

Так же были реализованы методы взаимодействия с клиентской частью, посредством принятия запросов к БД и возвращению результатов запросов в виде json-строки.

Взаимодействия с клиентской частью производилось посредством протокола HTTP. Так же возможно использование протокола HTTPS при установке соответствующего сертификата безопасности.

Серверная часть получает запросы от клиентской части через механизм GET/POST HTTP-запросов. При этом используются параметры запросов, приведённые в таблице 10.

Таблица 10 – Параметры запроса к серверной части

№ п/п	Параметр запроса	Описание	Пример
1	key	Ключ безопасности, используется для доступа к API серверной части	key=lebedev
2	sql	Запрос к БД, результат которого преобразуется в json-объект и возвращается клиентской части приложения	sql=select+*+from+пути
3	username	Вспомогательный параметр, для указания контента возвращаемых данных	username=konstatin

Серверная часть получает запрос от клиентской части и проверяет ключ безопасности. Если ключ безопасности неверный – клиентской части выдаётся ошибка. Далее, после прохождения проверки безопасности, из полученного параметра с именем «sql» формируется запрос к базе данных. Полученный ответ от базы данных сериализуется в json-объект и выдаётся клиентской части в текстовом виде через протокол HTTP.

Листинг серверной части представлен в приложении В.

3.3 Реализация клиентской части приложения

Приложение было разработано на языке C#, PHP и JavaScript с применением среды разработки Visual Studio 2022 и Atom [24]. Структура программы отражена на диаграмме компонентов, которая представлена на рисунке 29.

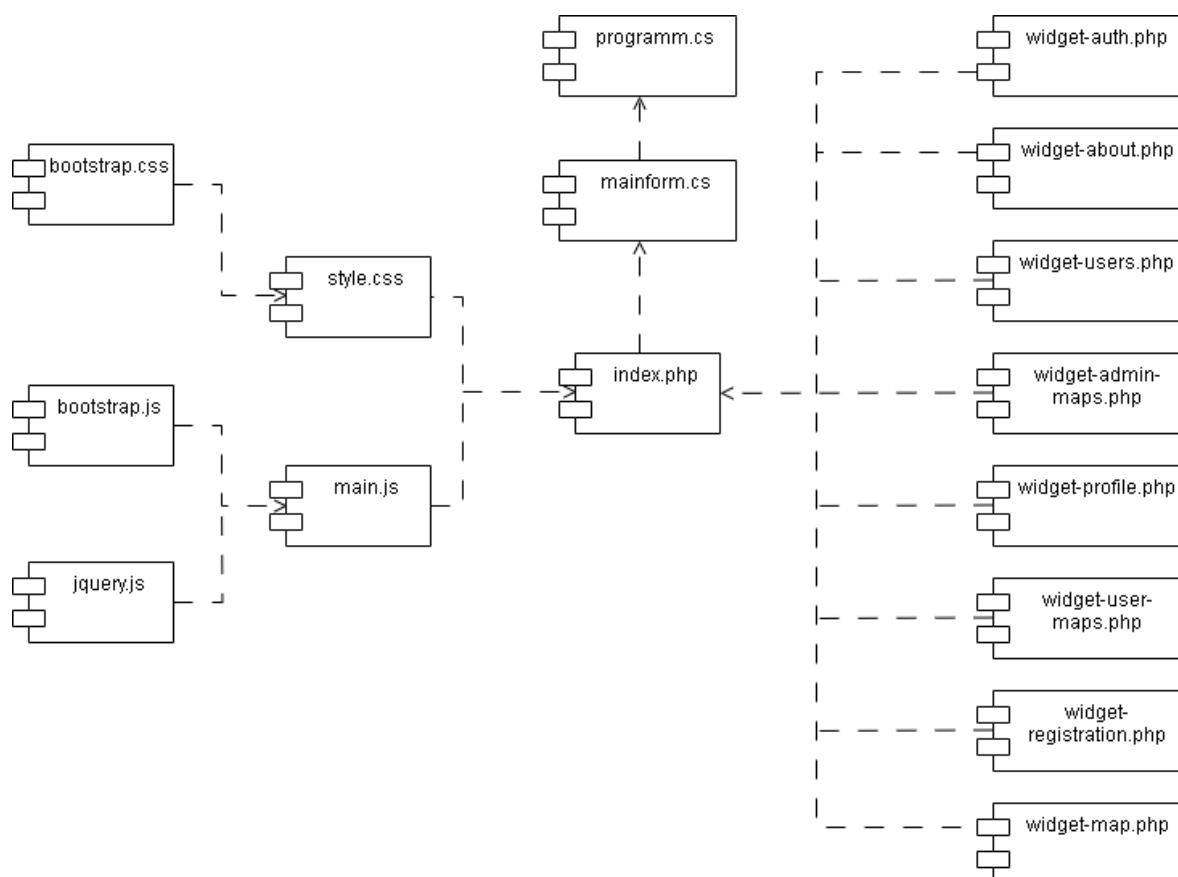


Рисунок 29 – Диаграмма компонентов приложения

Описание компонентов представлено в таблице 11.

Таблица 11 – Компоненты клиентского приложения

Компонент	Назначение
program.cs	Основной файл приложения.
mainform.cs	Файл основного окна приложения.
index.php	Файл начальной страницы приложения.
widget-auth.php	Файл страницы входа в систему.
widget-about.php	Файл страницы «О программе».
widget-users.php	Файл страницы отображающий список пользователей.
widget-admin-maps.php	Файл страницы редактирования карты администратором.
widget-profile.php	Файл страницы редактирования профиля пользователя.
widget-user-maps.php	Файл страницы со списком карт пользователя.
widget-map.php	Файл страницы с навигации по карте.
widget-registration.php	Файл страницы регистрации нового пользователя.
style.css	Файл стилей приложения.
bootstrap.css	Дополнительный файл стилей приложения.
main.js	Файл с основным программным кодом.
bootstrap.js	Файл с дополнительным программным кодом.
jquery.js	Файл с дополнительным программным кодом для управления событиями и манипуляцией элементами.

Физическое расположение компонентов системы показано на диаграмме развертывания, на которая представлена на рисунке 30.

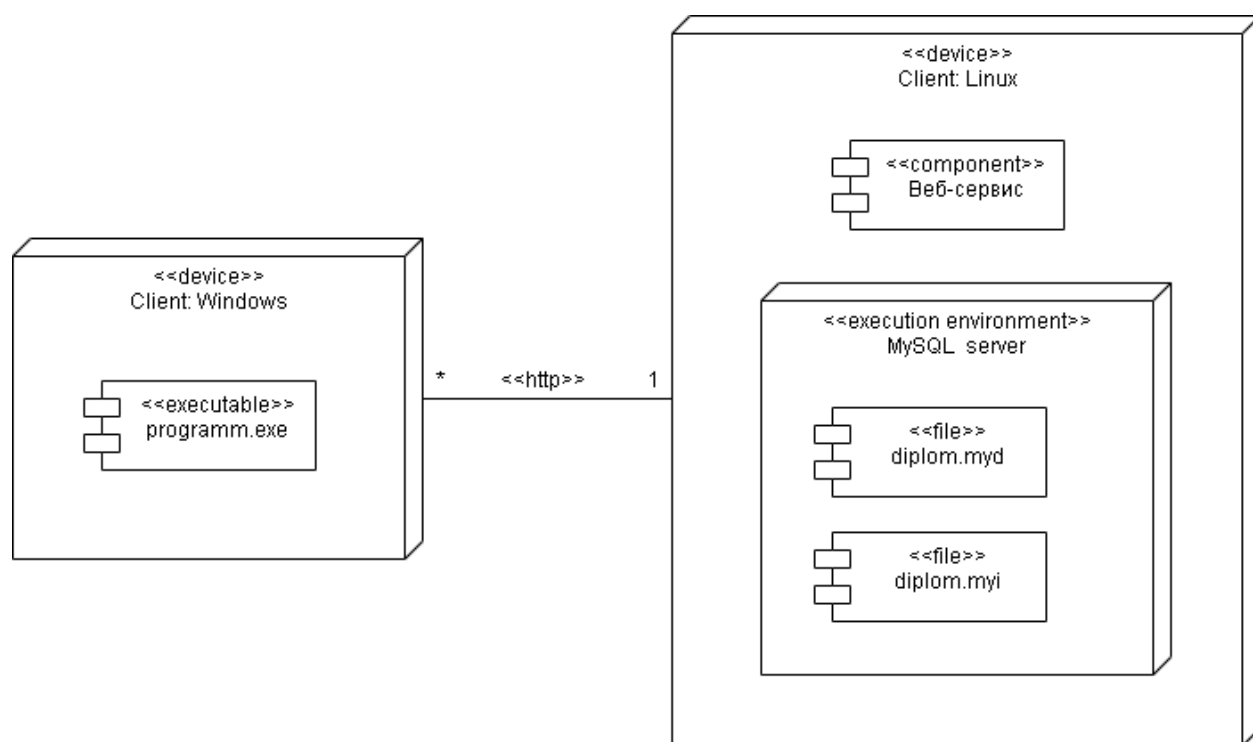


Рисунок 30 – Диаграмма развертывания программного обеспечения

На диаграмме показано, что клиентская программа функционирует на персональном компьютере с операционной системой Windows. Серверная составляющая и база данных размещены на сервере, использующем Linux. Обмен данными между сервером и клиентом осуществляется через протокол HTTP.

Код клиентского компонента программы представлен в приложении Г.

4 Тестирование программного продукта

4.1 Тестирование серверной части приложения

Важно проверить, что серверная часть приложения работает без ошибок. Для этого была проведена проверка работоспособности серверной части приложения. Результаты этой проверки отображены в таблице 12. Скриншоты результатов тестов приведены в приложении Д.

Таблица 12 – Результаты тестирования серверной части

№	Описание теста	Запрос		Результат
		Метод	Параметры	
1	Получение списка пользователей	GET	query=select+*+from+пользователи&key=lebedev	Тест пройден. Список пользователей успешно получен. Код ответа HTTP – 200 (рисунок Д.1).
2	Получение списка пользователей с неверным ключом API	GET	query=select+*+from+пользователи&key=nelebedev	Тест пройден. Список пользователей успешно не получен, т. к. указан неверный код. Код ответа HTTP – 200 (рисунок Д.2).
3	Получение списка путей	GET	query=select+*+from+пути&key=lebedev	Тест пройден. Список путей успешно получен. Код ответа HTTP – 200 (рисунок Д.3).
4	Получение списка сохраненных путей пользователей	GET	query=select+*+from+пути_пользователя&key=lebedev	Тест пройден. Список путей пользователей успешно получен. Код ответа HTTP – 200 (рисунок Д.4).
5	Получение списка сохраненных путей пользователем	GET	query=select+id_пути+from+пути_пользователя+where+id_пользователя=1&key=lebedev	Тест пройден. Список путей пользователя успешно получен. Код ответа HTTP – 200 (рисунок Д.5).

Тестирование было успешно пройдено, что говорит о правильной работе серверной части приложения. Далее производим тестированию клиентской части приложения.

4.2 Тестирование клиентской части приложения

4.2.1 Функциональное тестирование

Для проверки правильной работы клиентской части было проведено функциональное тестирование. Результаты тестирования приведены в таблице 13. Скриншоты результатов тестов приведены в приложении Е.

Таблица 13 – Функциональное тестирование

Вариант использования	Тест и параметры	Результат
Зарегистрироваться	Ввести правильные для регистрации. Логин: kostya. Пароль: 123456.	Тест пройден. Пользователь успешно зарегистрирован в системе (рисунок Д.1).
	Ввести данные с логином существующего в системе пользователя. Логин: user.	Тест пройден. Показано сообщение о том, что имя пользователя уже занято (рисунок Д.2).
Авторизоваться	Ввести правильные данные. Логин: kostya. Пароль: 123456..	Тест пройден. Авторизация прошла успешно (рисунок Д.3).
	Ввести неправильные данные. Логин: user2. Пароль: 12345678.	Тест пройден. Выведено уведомление о неверном логине или пароле (рисунок Д.4).

Функциональное тестирование показало правильную работу клиентской части приложения.

4.2.2 Модульное тестирование

Модульные тесты представляют собой автоматизированные проверки, которые анализируют правильность функционирования отдельных компонентов программы (чаще всего функций или методов), сопоставляя полученный результат с ожидаемым. Их основное преимущество заключается в возможности оперативно обнаруживать и исправлять ошибки, появляющиеся в процессе изменения исходного кода.

Для тестирования мы создадим класс `ValidationTest` с методами для проверки логина и пароля, который использует методы класса `RegisterActivity` `isValidLogin` и `isValidPassword`, проверяющие логин и пароль соответственно.

Исходный код модульных тестов был представлен в листинге 1.

Листинг 1 – Исходный код модульных тестов

```
class RegisterActivity {
    static isValidLogin(login) {
        const regex = /^[a-zA-Z0-9]{3,20}$/;
        return regex.test(login);
    }
}
```

```

    }

    static isValidPassword(password) {
        const regex = /^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{4,30}$/;
        return regex.test(password);
    }
}

class ValidationTest {
    constructor() {}

    validLogin() {
        console.log((RegisterActivity.isValidLogin("user1234"))?
'validLogin passed' : 'validLogin failed');
    }

    loginTooShort() {
        console.log((RegisterActivity.isValidLogin("xz"))? 'loginTooShort
passed' : 'loginTooShort failed');
    }

    loginTooLong() {
        console.log((RegisterActivity.isValidLogin("verylongloginverylongloginver
ylonglogin"))? 'loginTooLong passed' : 'loginTooLong failed');
    }

    disallowedSymbolLogin() {
        console.log((RegisterActivity.isValidLogin("№novaliglogin"))?
'disallowedSymbolLogin passed' : 'disallowedSymbolLogin failed');
    }

    validPassword() {
        console.log((RegisterActivity.isValidPassword("password1234"))?
'validPassword passed' : 'validPassword failed');
    }

    passwordTooShort() {
        console.log((RegisterActivity.isValidPassword("b1"))?
'passwordTooShort passed' : 'passwordTooShort failed');
    }

    passwordTooLong() {
        console.log((RegisterActivity.isValidPassword("passwordpasswordpasswordpa
ssword"))? 'passwordTooLong passed' : 'passwordTooLong failed');
    }

    disallowedSymbolPassword() {
        console.log((RegisterActivity.isValidPassword("my_password"))?
'disallowedSymbolPassword passed' : 'disallowedSymbolPassword failed');
    }

    noNumbersPassword() {
        console.log((RegisterActivity.isValidPassword("something"))?
'noNumbersPassword passed' : 'noNumbersPassword failed');
    }
}

```

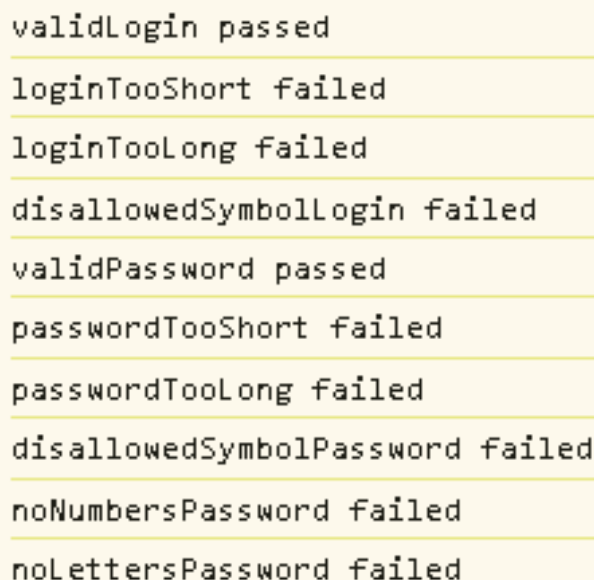
```

        noLettersPassword() {
            console.log((RegisterActivity.isValidPassword("1234567890"))?
            'noLettersPassword passed' : 'noLettersPassword failed');
        }
    }

    // Запускаем тесты
    const validationTest = new ValidationTest();
    validationTest.validLogin();
    validationTest.loginTooShort();
    validationTest.loginTooLong();
    validationTest.disallowedSymbolLogin();
    validationTest.validPassword();
    validationTest.passwordTooShort();
    validationTest.passwordTooLong();
    validationTest.disallowedSymbolPassword();
    validationTest.noNumbersPassword();
    validationTest.noLettersPassword();

```

Результаты модульных тестов приведены на рисунке 31



validLogin passed
loginTooShort failed
loginTooLong failed
disallowedSymbolLogin failed
validPassword passed
passwordTooShort failed
passwordTooLong failed
disallowedSymbolPassword failed
noNumbersPassword failed
noLettersPassword failed

Рисунок 31 – Результаты модульного тестирования

Все тесты были успешно пройдены, что говорит о правильной работе тестируемых методов.

4.3 Метрики кода

Метриками кода называются некоторые количественные величины, позволяющие подсчитать, насколько код программы обладает тем или иным свойством. Существует огромное количество самых разнообразных метрик,

имеющих различное назначение. Наиболее известными являются следующие группы метрик:

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

Рассмотрим метод `dijkstra`, отвечающий за поиск кратчайшего маршрута. Его код представлен в листинге 2.

Листинг 2 – Поиск маршрута методом Дейстра

```
//Функция Дейкстры
function dijkstra(graph, startNode) {
    const distances = {};
    const prev = {};
    const pq = new PriorityQueue();

    // Инициализация расстояний от начального узла до всех остальных
    // узлов
    for (const node in graph) {
        distances[node] = Infinity;
        prev[node] = null;
    }
    distances[startNode] = 0;
    pq.enqueue([startNode, 0]);
    //Цикл проверки на пустоту
    while (!pq.isEmpty()) {
        const shortestPath = pq.dequeue();
        const currentNode = shortestPath[0];
        //Проход по графам
        for (const neighbor in graph[currentNode]) {
            const altDistance = distances[currentNode] +
graph[currentNode][neighbor];
            if (altDistance < distances[neighbor]) {
                distances[neighbor] = altDistance;
                prev[neighbor] = currentNode;
                pq.enqueue([neighbor, altDistance]);
            }
        }
    }

    return { distances, prev };
}
```

Данный метод находится в клиентской части приложения и запускается при выборе пользователем начальной и конечной точек маршрута. Он анализирует имеющиеся пути на карте и формирует кратчайший маршрут из начальной точки в конечную.

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

4.3.1 Метрики размера программы

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

Таблица 14 – Метрики размера программы

Метрика	Количество
Количество строк кода	30
Строки-комментарии	4

Также вычислим процент комментариев. Он равен $(4/30)*100=13\%$. Данный результат говорит о том, что в коде сбалансированное количество комментариев и строк кода для его понимания.

4.3.2 Метрики Холстеда

Метрики Холстеда – это набор количественных показателей, используемых для анализа и оценки сложности, объема и качества программного обеспечения [30 – 32].

Для вычисления метрик Холстеда необходимо измерить 4 характеристики программы:

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

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

Таблица 15 – Словарь операторов

Оператор	Число операторов
if	1
!	1
;	14
()	10
=	11
{}	8
return	1
new	1
<	1
+	1

Таблица 16 – Словарь операндов

Операнд	Число операндов
graph	3
distances	7
prev	4
pq	5
altDistance	4
currentNode	5
shortestPath	2
startNode	2

Таким образом, число уникальных операторов программы $n1 = 10$, число уникальных операндов программы $n2 = 8$, общее число операторов в программе $N1 = 49$, общее число операндов в программе $N2 = 32$.

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

- словарь программы (n):

$$n = n1 + n2;$$

- длина программы (N):

$$N = N1 + N2;$$

- объем программы (V):

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

- сложность программы (D):

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

- оценка интеллектуальных усилий (E):

$$E = D * V.$$

Рассчитаем словарь программы: $n = 10+8 = 18$.

Рассчитаем длину программы $N = 49+32=81$.

Рассчитаем объем программы: $V = 81 * \log_2 18 = 102$.

Рассчитаем сложность программы: $D = (10/2) * (32/8) = 20$.

Рассчитаем оценку интеллектуальных усилий: $E=20 * 102 = 2040$.

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

Заключение

В процессе выполнения дипломной работы была создана программа для компьютеров, которая помогает автоматически строить маршруты перемещения.

В начале работы проводился анализ существующих программ с подобными функциями, чтобы узнать их сильные и слабые стороны, а также определить, какие функции должны быть у нового приложения и какова его целевая аудитория.

Затем был спланирован процесс разработки и рассчитан бюджет. Результаты планирования показали, что работа займет примерно 89 дней и её стоимость составит 1 632 100 рублей, что делает проект экономически оправданным.

На стадии проектирования разработаны структура базы данных и архитектура программы. База данных была создана с использованием языка SQL и работает на основе MySQL Server, а серверная часть написана на PHP с помощью среды разработки Atom. Клиентская часть сделана на C# и JavaScript, используя среду разработки Visual Studio 2022 и Atom.

Проведено тестирование всех компонентов программы, которое подтвердило ее правильную работу, а также измерены параметры качества кода.

В дальнейшем планируется разработать мобильную версию приложения и использовать приложение в системах автоматической навигации на производстве.

Таким образом, все цели дипломной работы достигнуты.

Список использованных источников

1. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 12.05.2024).
2. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 12.05.2024).
3. Указ о национальных целях развития Российской Федерации на период до 2030 года и на перспективу до 2036 года. [Электронный ресурс]. URL: <http://www.kremlin.ru/events/president/news/73986> (дата обращения 12.05.2024).
4. Постановление Правительства Российской Федерации от 15.04.2014 г. № 313. [Электронный ресурс]. URL: <http://government.ru/docs/all/91296/> (дата обращения 12.05.2024).
5. Программное обеспечение для навигации персонала на промышленной территории // pt.2035.university [Электронный ресурс]. URL: <https://pt.2035.university/project/bpla-dla-stroitelno-go-kontrola>.
6. Интерактивная навигация «LOGICMAP» от «Zorgtech» // kiosks.ru [Электронный ресурс]. URL: <https://kiosks.ru/index.php/product/interaktivnaya-navigaciya-logicmap/>.
7. Навигация внутри помещений: полное руководство // nvgn.ru [Электронный ресурс]. URL: <https://nvgn.ru/blog/navigacziya-vnutri-pomeshhenij-polnoe-rukovodstvo/>.
8. Операционные системы (мировой рынок) // tadviser.ru [Электронный ресурс]. URL: [https://www.tadviser.ru/index.php/Статья:Операционные_системы_\(мировой_рынок\)](https://www.tadviser.ru/index.php/Статья:Операционные_системы_(мировой_рынок))

9. Статистика: Windows 10 — самая популярная ОС в мире, на втором месте уже W11 // amd.news [Электронный ресурс]. URL: <https://amd.news/high-tech-news/article/040223-dolya-versiy-windows/>
10. Сравнение доли рынка Linux и Windows // komyounity.com [Электронный ресурс]. URL: <https://komyounity.com/sravnenie-doli-rinka-linux-i-windows/>
11. Исследование. Какие ОС популярны в России, и кто ими пользуется // hi-tech.mail.ru [Электронный ресурс]. URL: <https://hi-tech.mail.ru/news/26991-os-pop-in-russia/>
12. Статистическая информация об использовании операционных систем на май 2024 года. [Электронный ресурс]. URL: <https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide>
13. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков. – СПб.: Лань, 2019. – 244 с.
14. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов. – Рн/Д: Феникс, 2018. – 102 с.
15. Гонтарева, И.В. Управление проектами / И.В. Гонтарева, Р.М. Нижегородцев, Д.А. Новиков. – М.: КД Либроком, 2018. – 384 с.
16. Ньютон, Р. Управление проектами от А до Я / Р. Ньютон. – М.: Альпина Паблишер, 2018. – 180 с.
17. Вольфсон, Б.Л. Гибкое управление проектами и продуктами / Б.Л. Вольфсон. – СПб.: Питер, 2019. – 208 с.
18. Скороход, С.В. Управление проектами средствами Microsoft Project. Курс лекций. / С.В. Скороход. – М.: Интуит НОУ, 2016. – 319 стр.
19. Куперштейн, В.И. Microsoft Project 2013 в управлении проектами / В.И. Куперштейн. – СПб:БХВ-Петербург, 2013. – 432 с.
20. Джонсон, Т. Microsoft Project 2016. Шаг за шагом // Т.Джонсон, К.Четфилд. – М., Изд-во «КТК Галактика», 2018. – 566 с.

21. Акатова, Н.А. Планирование и контроль выполнения проектов с использованием MS Project / Н.А. Акатова, М.Г. Коротких. – М.: МИСиС, 2019. – 74 с.
22. Унифицированный процесс Rational (RUP) // Интернет-технологии.ру [Электронный ресурс]. – URL: <https://www.internet-technologies.ru/articles/newbie/unificirovannyu-process-rational-rup.html> (дата обращения: 16.05.2024).
23. Арлоу, Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2-е издание / Д. Арлоу, А. Нейштадт – СПб: Символ Плюс, 2007. – 624 с.
24. Microsoft Visual Studio creator manual (Начало работы с Visual Studio IDE) [Электронный ресурс]. URL: <https://visualstudio.microsoft.com/ru/vs/getting-started/>
25. Введение в Web API // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/mvc/12.1.php> (дата обращения: 25.05.2024).
26. Андерсон Р. Создание веб-API с помощью ASP.NET Core и Visual Studio для Windows // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio> (дата обращения: 19.05.2024).
27. Руководство по Entity Framework // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/entityframework/2.4.php> (дата обращения: 19.05.2024).
28. Database First // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef/ef6/modeling/designer/workflows/database-first> (дата обращения: 19.05.2024).
29. Entity Framework 6. Использование Database First // Professor Web [Электронный ресурс]. URL: https://professorweb.ru/my/entity-framework/6/level1/1_6.php (дата обращения: 19.05.2024).

30. Программный код и его метрики // Habr.com [Электронный ресурс]. URL: <https://habr.com/ru/company/intel/blog/106082/> (дата обращения: 19.05.2024).

31. Аверьянов, А.В. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ / А.В. Аверьянов, И.Н. Кошель, В.В. Кузнецов // Известия высших учебных заведений. Приборостроение. – 2019. – №11. – С. 970 – 975.

32. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л.Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4.– С. 950 – 959

Приложение А
(справочное)
Глоссарий

Windows – десктопная операционная система для персональных компьютеров, планшетов и других электронных устройств, разрабатываемая и выпускаемая американской компанией Microsoft и используемая на подавляющем большинстве современных персональных компьютерах.

API (Application Programming Interface) – это набор правил и протоколов, которые позволяют различным программам взаимодействовать друг с другом. API определяет методы и структуры данных, которые можно использовать для запроса функций или информации от другой программы или сервиса.

RUP (Rational Unified Process) – это итеративный подход к разработке программного обеспечения, который включает в себя четыре основные фазы: начало, уточнение, конструирование и переход. RUP акцентирует внимание на управлении рисками и изменчивости требований в процессе разработки.

SQL (Structured Query Language) – это язык управления данными, используемый для работы с базами данных. С помощью SQL можно создавать, модифицировать и извлекать данные из баз данных, а также управлять их структурой.

HTTP (Hypertext Transfer Protocol) – это протокол передачи гипертекста, который используется для передачи данных по Интернету. HTTP является основой любого обмена данными в Всемирной паутине и определяет методы и форматы сообщений, которые используются для общения между клиентами и серверами.

UML (Unified Modeling Language) – это стандартный язык моделирования, используемый для спецификации, визуализации, построения и документирования артефактов программных систем. UML предоставляет набор диаграмм для представления различных аспектов системы, таких как классы, объекты, действия и взаимодействия между ними.

.NET – это кросс-платформенная среда выполнения и фреймворк разработки от Microsoft, позволяющий создавать различные типы приложений, включая веб-приложения, мобильные приложения, игры и IoT-решения,

поддерживает несколько языков программирования, таких как C#, F# и Visual Basic, и предлагает богатую библиотеку классов для упрощения разработки.

JavaScript – это высокоуровневый, интерпретируемый язык программирования, который поддерживает объектно-ориентированное, императивное и функциональное стили программирования. JavaScript позволяет добавлять динамическое поведение к HTML-контенту, обрабатывать пользовательский ввод и взаимодействовать с браузером и другими веб-технологиями.

PHP (Hypertext Preprocessor) – это открытый скриптовый язык программирования, специально разработанный для веб-разработки. Он может быть встроен в HTML-код и выполняется на стороне сервера, что делает его идеальным инструментом для создания динамических веб-сайтов и приложений. PHP поддерживает широкий спектр баз данных, включая MySQL, PostgreSQL, Oracle и другие, а также предоставляет возможности для работы с файлами, обработки форм и многое другое.

Маршрут – это заранее определенный план перемещения от одной точки к другой, который включает в себя последовательность промежуточных точек или остановок. Маршруты могут быть разработаны для различных видов транспорта (например, автомобильные маршруты, велосипедные маршруты) и обычно учитывают такие факторы, как расстояние, время пути, условия дороги и предпочтительные направления движения.

Карта – это графическое представление географической области, которое показывает относительное расположение объектов и мест на поверхности Земли или других планет. Карты используются для навигации, планирования путешествий и изучения географии. Они могут быть представлены в различных масштабах и форматах, включая бумажные карты, цифровые карты и интерактивные карты онлайн.

Путь – это конкретное направление или траектория, по которой происходит движение от начальной до конечной точки. Путь может быть прямым или извилистым, коротким или длинным, и его выбор зависит от

многих факторов, таких как цель путешествия, доступность транспортных средств, состояние дорог и личные предпочтения.

Приложение Б
(рекомендуемое)
Исходный код скриптов базы данных

```

-- phpMyAdmin SQL Dump
-- version 4.9.7
-- https://www.phpmyadmin.net/
--
-- Хост: 10.0.0.94:3306
-- Время создания: Июн 15 2024 г., 20:23
-- Версия сервера: 10.3.37-MariaDB-log
-- Версия PHP: 7.4.11

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- База данных: `lebedev_diplom`
--
CREATE DATABASE IF NOT EXISTS `lebedev_diplom` DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci;
USE `lebedev_diplom`;

--
-- Структура таблицы `пользователи`
--
DROP TABLE IF EXISTS `пользователи`;
CREATE TABLE IF NOT EXISTS `пользователи` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `логин` varchar(255) NOT NULL,
  `пароль` varchar(255) NOT NULL,
  `фио` varchar(255) NOT NULL,
  `роли` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_general_ci;

--
-- Дамп данных таблицы `пользователи`
--
INSERT INTO `пользователи` (`id`, `логин`, `пароль`, `фио`, `роли`)
VALUES(1, 'admin', 'admin', 'Админ Админович Админов', 'administrator');
INSERT INTO `пользователи` (`id`, `логин`, `пароль`, `фио`, `роли`)
VALUES(2, 'user', 'user', 'Юзеров Юзер Юзерович', 'user');

--

```

```
-- Структура таблицы `пути`
```

```
--
```

```
DROP TABLE IF EXISTS `пути`;
CREATE TABLE IF NOT EXISTS `пути` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `название` varchar(255) NOT NULL,
  `пути` text NOT NULL,
  `картинка` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8
COLLATE=utf8_general_ci;
```

```
--
```

```
-- Дамп данных таблицы `пути`
```

```
--
```

```
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(1,
'Tестовый',
'{\"road1\": [{\"x\":20,\"y\":80},{\"x\":35,\"y\":75},{\"x\":35,\"y\":63},
{\"x\":20,\"y\":58},{\"x\":20,\"y\":45},{\"x\":61,\"y\":30},{\"x\":61,\"y\":10}],
\"road2\": [{\"x\":55,\"y\":100},{\"x\":55,\"y\":20},{\"x\":28,\"y\":12}],
\"road3\": [{\"x\":90,\"y\":70},{\"x\":68,\"y\":68},{\"x\":68,\"y\":57},
{\"x\":38,\"y\":57},{\"x\":10,\"y\":50}]}'
, 'https://icdn.lenta.ru/images/2022/06/28/15/20220628154810835/square_1280_0b4bf135536eec17791c50d039613ff9.jpg');
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(2, 'X',
'{\r\n  \"road1\": [\r\n    {\r\n      \"x\": 0,\r\n      \"y\": 0\r\n    },\r\n    {\r\n      \"x\": 100,\r\n      \"y\": 100\r\n    } \r\n  ],\r\n  \"road2\": [\r\n    {\r\n      \"x\": 100,\r\n      \"y\": 0\r\n    },\r\n    {\r\n      \"x\": 0,\r\n      \"y\": 100\r\n    } \r\n  ] \r\n}',
'https://nypost.com/wp-content/uploads/sites/2/2021/12/ukraine-crisis-russia-military-01.jpg');
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(3,
'Промзона 36',
'',
'https://www.roscosmos.ru/media/gallery/big/38645/6426344147.jpg');
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(4,
'Промзона 45',
'',
'https://cdnimages.absatz.media/images/2022.06/original/1920_62bafd61b09c60d6cef0cea6.jpg');
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(5,
'Промзона 11',
'',
'https://ozrenmedia.com/wp-content/uploads/sites/6/2022/06/roskosmos-satelit.jpg');
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(6,
'Промзона 36',
'',
'http://www.un-autre-regard-sur-la-terre.org/document/blogUARST/Evenements/Russia%202018%20-%20FIFA%20-%20World%20cup%20-%20Mondial/Russia%202018%20-%20Satellite%20Pleiades%20-%20Moscou%20-%20Stade%20Luzhniki%20-%2007-05-2018.jpg');
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(7,
'Промзона 29',
'',
'https://cdnstatic.rg.ru/uploads/photogallery/2023/05/17/2023-05-17t161258z_1707905307_rc2xz0aozesl_rtrmadp_3_ukraine-crisis-bakmut_9e9.jpeg');
INSERT INTO `пути` (`id`, `название`, `пути`, `картинка`) VALUES(8,
'Промзона 5',
'',
'http://cdn.globalso.com/chinagearreducer/application-07.jpg');
```

```
-- -----
```

```
--
-- Структура таблицы `пути_пользователя`
--

DROP TABLE IF EXISTS `пути_пользователя`;
CREATE TABLE IF NOT EXISTS `пути_пользователя` (
  `id_пользователя` int(11) NOT NULL,
  `id_пути` int(11) NOT NULL,
  PRIMARY KEY (`id_пользователя`,`id_пути`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_general_ci;

--
-- Дамп данных таблицы `пути_пользователя`
--

INSERT INTO `пути_пользователя` (`id_пользователя`, `id_пути`) VALUES(1,
1);
INSERT INTO `пути_пользователя` (`id_пользователя`, `id_пути`) VALUES(1,
2);
INSERT INTO `пути_пользователя` (`id_пользователя`, `id_пути`) VALUES(1,
5);
INSERT INTO `пути_пользователя` (`id_пользователя`, `id_пути`) VALUES(1,
6);
INSERT INTO `пути_пользователя` (`id_пользователя`, `id_пути`) VALUES(1,
8);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Приложение В
(рекомендуемое)
Исходный код серверной части

Листинг В.1 – Файл Db/index.php

```

<?php

// Параметры подключения к базе данных
$host = 'localhost';
$db    = 'lebedev_diplom';
$user  = '045767453_diplom';
$pass  = 'xLNF23U76dANx';

// Получаем JSON-тело запроса
$jsonData = file_get_contents('php://input');
file_put_contents('_jsonData.txt', var_export($jsonData, true));

file_put_contents('_REQUEST.txt', var_export($_REQUEST, true));
file_put_contents('_POST.txt', var_export($_POST, true));
file_put_contents('_GET.txt', var_export($_GET, true));
file_put_contents('_COOKIE.txt', var_export($_COOKIE, true));

try
{

    // Получаем все заголовки запроса
    $headers = getallheaders();

    // Проверяем заголовок Content-Type
    if (isset($headers['Content-Type']))
    {
        $contentType = $headers['Content-Type'];

        // Обработка JSON
        if (strpos($contentType, 'application/json') !== false)
        {
            // Получаем JSON-тело запроса
            $jsonData = file_get_contents('php://input');

            // Декодируем JSON в массив PHP
            $data = json_decode($jsonData, true);

            // Здесь ваш код для обработки JSON-данных
            throw new Exception('JSON данные получены. Данные: ' . var_export($data));
            //echo "JSON данные получены. Данные: ";
            //var_dump($data);
        }

        // Обработка application/x-www-form-urlencoded
        elseif (strpos($contentType, 'application/x-www-form-urlencoded') !== false)
        {
            // Здесь ваш код для обработки данных формы
            parse_str(file_get_contents('php://input'), $formData);

            echo "Данные формы получены. Данные: ";
            var_dump($formData);
        }

        // Если Content-Type не соответствует ни одному из ожидаемых

```

```

        else
        {
            throw new Exception('Неизвестный Content-Type:
' . $contentType);
        }
    }
    else
    {
        throw new Exception('Заголовок Content-Type не найден.');
```

// Проверяем запрос на ключ

```

$key = $_REQUEST['key'];
if ($key != 'lebedev')
{
    throw new Exception('Неправильный ключ');
}

// Подключение к базе данных
$pdo = new PDO("mysql:host=$host;dbname=$db", $user, $pass);

// Извлечение SQL-запроса из REQUEST-параметра
if (!isset($_REQUEST['query'])) {
    throw new Exception('SQL query parameter is missing');
}
$sqlQuery = $_REQUEST['query'];

// Выполнение SQL-запроса
$stmt = $pdo->prepare($sqlQuery);
$stmt->execute();

// Получение результатов
$results = $stmt->fetchAll(PDO::FETCH_ASSOC);

// Формирование ответа в формате JSON
header('Content-Type: application/json');
echo json_encode(['status' => 'success', 'data' => $results]);
}
catch (Exception $e)
{
    // Отправка ошибки в формате JSON
    header('Content-Type: application/json');
    echo json_encode(['status' => 'error', 'message' => $e-
>getMessage()]);
}

```

Приложение Г
(рекомендуемое)
Исходный код приложения

Листинг Г.1 – Файл Program.cs

```
namespace diplom
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high
            DPI settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new MainForm());
        }
    }
}
```

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace diplom
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }
    }
}
```

Листинг Г.2 – Файл MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace diplom
{
    public partial class MainForm : Form
```

```

    {
        public MainForm()
        {
            InitializeComponent();
        }
    }
}

```

Листинг Г.3 – Файл MainForm.Designer.cs

```

namespace diplom
{
    partial class MainForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
        disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            webView21 = new Microsoft.Web.WebView2.WinForms.WebView2();

            ((System.ComponentModel.ISupportInitialize)webView21).BeginInit();
            SuspendLayout();
            //
            // webView21
            //
            webView21.AllowExternalDrop = true;
            webView21.Anchor = AnchorStyles.Top | AnchorStyles.Bottom |
AnchorStyles.Left | AnchorStyles.Right;
            webView21.CreationProperties = null;
            webView21.DefaultBackgroundColor = Color.White;
            webView21.Location = new Point(0, 0);
            webView21.Name = "webView21";
            webView21.Size = new Size(583, 361);
            webView21.Source =
            Uri("https://airsouffle.ru/kostya/diplom/", UriKind.Absolute);
        }
    }
}

```

new

```

webView21.TabIndex = 0;
webView21.ZoomFactor = 1D;
//
// MainForm
//
AutoScaleDimensions = new.SizeF(7F, 15F);
AutoScaleMode = AutoScaleMode.Font;
ClientSize = new.Size(584, 361);
Controls.Add(webView21);
Name = "MainForm";
StartPosition = FormStartPosition.CenterScreen;
Text = "Автоматизированная система построения маршрута
движения";

((System.ComponentModel.ISupportInitialize)webView21).EndInit();
ResumeLayout(false);
}

#endregion

private Microsoft.Web.WebView2.WinForms.WebView2 webView21;
}
}

```

Листинг Г.4 – Файл index.php

```

<!doctype html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <!--link
href="//cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMHjY6hW+ALEwIH"
crossorigin="anonymous"-->
    <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

    <title>Диплом</title>

    <link rel="stylesheet" href="98-3/98.css">
    <link rel="stylesheet" href="win9x.css">
    <link href="style.css" rel="stylesheet">
    <!--link rel="stylesheet" type="text/css"
href="95/assets/win95.css~"-->

    <script src="vendor/jquery/js/jquery.min.js"></script>
    <!--script
src="https://cdn.jsdelivr.net/npm/jquery/dist/jquery.min.js"></script-->
    <!--script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.0/jquery.min.js"
integrity="sha512-
suUtsPqYmFd5Ls30Nz6bjDX+TCcfEzhFfqqijfdggsaFZoylvTj+2odBzshs0TCwYrYZhQeC
gHgJEknCb2YVQ==" crossorigin="anonymous" referrerpolicy="no-
referrer"></script-->

```

```

</head>

<body class="">
  <div class="container-fluid p-0">
    <div class="title-bar d-none">
      <div class="title-bar-text">
        Диплом Константина Лебедева
      </div>

      <div class="title-bar-controls">
        <button aria-label="Minimize"></button>
        <button aria-label="Maximize"></button>
        <button aria-label="Close"></button>
      </div>
    </div>
    <div class="row g-2 m-0">
      <div id="sidebar" class="col col-md-4 col-lg-3 col-xl-2 col-
xxl-1 p-0 m-0 g-2">
        <div class="menu bg" data-role="guest">
          <h2 class="w-100">Меню гостя</h2>
          <a class="btn btn-primary" data-widget="auth">Вход в
систему</a>
          <a class="btn btn-primary" data-widget="about">О
программе</a>
        </div>
        <div class="menu bg-primary-subtle" data-role="admin">
          <h2 class="w-100">Меню администратора</h2>
          <a class="btn btn-primary" data-
widget="users">Управление пользователями</a>
          <a class="btn btn-primary" data-widget="admin-
maps">Управление картами</a>
        </div>
        <div class="menu bg-success-subtle" data-role="user">
          <h2 class="w-100">Меню пользователя</h2>
          <a class="btn btn-primary" data-
widget="profile">Управление профилем</a>
          <a class="btn btn-primary" data-widget="user-
maps">Выбрать карту</a>
        </div>
      </div>

      <div id="main" class="col col-md-8 col-lg-9 col-xl-10 col-
xxl-11 p-2 m-0 g-2" style="display: none;">
        <p class="jambo d-none">
          Здесь набор виджетов меняющийся при нажатии кнопки из
меню
        </p>
        <?php include_once('widget-auth.php') ?>
        <?php include_once('widget-about.php') ?>
        <?php include_once('widget-users.php') ?>
        <?php include_once('widget-admin-maps.php') ?>
        <?php include_once('widget-profile.php') ?>
        <?php include_once('widget-user-maps.php') ?>
        <?php include_once('widget-map.php') ?>
        <?php include_once('widget-registration.php') ?>
        <button class="d-none">Проверка</button>

```

```

        </div>
    </div>
</div>

    <!-- : Bootstrap Bundle with Popper -->
    <!--script
src="//cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.
js"                                integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script-->
    <script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
    <script src="main.js"></script>

</body>
</html>

```

Листинг Г.5 – Файл widget-about.php

```

<div class="widget cener" data-widget="about" data-role="guest">
    <h3>0 программе</h3>
    <p>Дипломный проект Лебедева К.А. Группа XXX</p>
    <p>Пенза, 2024</p>
</div>

```

Листинг Г.6 – Файл widget-admin-maps.php

```

<div class="widget" data-widget="admin-maps" data-role="admin" data-
onshow="загрузить_карты">
    <h3>Редактор карт</h3>
    <div class="maps"></div>
    <div class="map"></div>
</div>

```

Листинг Г.7 – Файл widget-auth.php

```

<div class="widget" data-widget="auth" data-role="guest">
    <h3>Авторизация в системе</h3>
    <form class="auth_form" action="" method="post">
        <input value="auth" name="action" type="hidden" />
        <input value="select" name="sql" type="hidden" />
        <div class="field-row-stacked">
            <label for="auth_username">Имя пользователя</label>
            <input id="auth_username" name="username" type="text" />
        </div>
        <div class="field-row-stacked">
            <label for="auth_password">Пароль</label>
            <input id="auth_password" name="password" type="password" />
        </div>
        <div class="field-row">
            <input type="submit" />
            <input type="reset" />
        </div>
        <div class="form-result"></div>
    </form>
    <br>
    <button class="default" data-
widget="registration">Регистрация</button>
</div>

```

Листинг Г.8 – Файл widget-map.php

```
<div class="widget" data-widget="map" data-role="user">
    Построение маршрута
</div>
```

Листинг Г.9 – Файл widget-profile.php

```
<div class="widget" data-widget="profile" data-role="user">
    <h3>Карточка пользователя</h3>
    <form class="profile_form" action="" method="post">
        <div class="field-row-stacked">
            <label for="profile_username">Имя пользователя</label>
            <input id="profile_username" type="text" />
        </div>
        <div class="field-row-stacked">
            <label for="profile_password">Пароль</label>
            <input id="profile_password" type="password" />
        </div>
        <div class="field-row-stacked">
            <label for="profile_fio_f">Фамилия</label>
            <input id="profile_fio_f" type="text" />
        </div>
        <div class="field-row-stacked">
            <label for="profile_fio_i">Имя</label>
            <input id="profile_fio_i" type="text" />
        </div>
        <div class="field-row-stacked">
            <label for="profile_fio_o">Отчество</label>
            <input id="profile_fio_o" type="text" />
        </div>
        <div class="field-row-stacked">
            <label for="profile_otdel">Отдел/подразделение</label>
            <input id="profile_otdel" type="text" />
        </div>
        <div class="field-row">
            <input type="submit" />
            <input type="reset" />
        </div>
    </form>
</div>
```

Листинг Г.10 – Файл widget-registration.php

```
<div class="widget" data-widget="registration" data-role="guest">
    <h3>Регистрация в системе</h3>
    <form class="registration_form" action="" method="post">
        <input id="action" type="hidden" value="registration"/>
        <div class="field-row-stacked">
            <label for="registration_username">Имя пользователя</label>
            <input id="registration_username" type="text" />
        </div>
        <div class="field-row-stacked">
            <label for="registration_password">Пароль</label>
            <input id="registration_password" type="password" />
        </div>
        <div class="field-row">
```

```

        <input type="submit" />
        <input type="reset" />
    </div>
    <div class="form-result"></div>
</form>
</div>

```

Листинг Г.11 – Файл widget-user-maps.php

```

<div class="widget" data-widget="user-maps" data-role="user">
    Список карт для пользователя
</div>

```

Листинг Г.12 – Файл widget-users.php

```

<div class="widget" data-widget="users" data-role="admin">
    <h3>Добавление пользователя</h3>
    <form class="user_add_form" action="" method="post">
        <div class="field-row-stacked">
            <label for="user_add_username">Имя пользователя</label>
            <input id="user_add_username" type="text" />
        </div>
        <div class="field-row-stacked">
            <label for="user_add_password">Пароль</label>
            <input id="user_add_password" type="password" />
        </div>
        <div class="field-row">
            <input type="submit" />
            <input type="reset" />
        </div>
    </form>
    <hr>
    <h3>Список пользователей</h3>
    <?php for ($i=0; $i < 11; $i++)
    {
        ?>

        <div class="field-row">
            <label for="user_update_username_<?=$i?>">Имя</label>
            <input id="user_update_username_<?=$i?>" type="text" />
            <label for="user_update_password_<?=$i?>">Пароль</label>
            <input id="user_update_password_<?=$i?>" type="password" />
            <input type="submit" value="Обновить"/>
            <input type="submit" value="Удалить"/>
        </div>

        <?php
    } ?>
</div>

```

Листинг Г.13 – Файл style.css

```

/*@import url("win9x.css");*/

.main {
    overflow-y: scroll;
}

```

```

.widget > form
{
    width: 200px;
}

.bg
{
    background-color: #aaa;
}

.widget.center
{
    display: flex;
    justify-content: center;
    align-items: center;
}

.maps {
    width: 100%;
    display: flex;
    overflow-x: auto;
}

.card.map-item
{
    width: 8rem;
}

.card-img, .card-img-bottom, .card-img-top
{
    object-fit: cover;
    object-position: center center;
    aspect-ratio: 1;
}

.carousel {
    overflow-x: auto;
}

.carousel-item {
    flex-shrink: 0; /* Предотвращает уменьшение размера карточки */
}

```

Листинг Г.14 – main.js

```

// Глобальные переменные
let role = 'guest'; // Начальное значение роли
const roles = ['guest', 'user', 'admin']; // Массив ролей
let user_id = 0; // текущий пользователь
let user_name = ''; // текущий пользователь

document.addEventListener('DOMContentLoaded', function() {
    const mainElement = document.getElementById('main');

```

```

// Обработчик события двойного клика
mainElement.addEventListener('dblclick', function() {
    // Находим индекс текущей роли в массиве
    let currentIndex = roles.indexOf(role);

    // Переход к следующей роли, или возвращение к первой, если
    // текущая роль последняя
    currentIndex = (currentIndex + 1) % roles.length;

    // Обновляем значение роли
    role = roles[currentIndex];

    // Обновление видимости элементов
    updateVisibility();

});

});

// Функция для обновления видимости элементов
function updateVisibility()
{
    // Обновление видимости элементов с атрибутом data-role и
    // классом.menu
    updateMenuVisibility();

    // Отображение первого элемента с классом.widget и скрывание остальных
    updateWidgetVisibility();
}

// Функция для обновления видимости меню
function updateMenuVisibility()
{
    const menuElements = document.querySelectorAll('[data-role].menu');

    menuElements.forEach(function(element) {
        element_role = element.getAttribute('data-role');
        if ((element_role === role) || (element_role === 'guest')) {

```

```

        element.style.display = ''; // Показываем элемент
    } else {
        element.style.display = 'none'; // Скрываем элемент
    }
});
}

// Функция для обновления видимости виджетов
function updateWidgetVisibility()
{
    спрятать_виджеты();
    const widgetElements = document.querySelectorAll('[data-
role="" + role + "'].widget');
    const widget = widgetElements[0];

    if (widget) {
        widget.style.display = '';
    }

    /*
    // Скрываем все элементы с классом.widget
    widgetElements.forEach(function(widget, index) {
        widget.style.display = 'none';

        // Показываем только первый элемент
        if (index === 0) {
            widget.style.display = '';
        }
    });
    */
}

// Вызываем функцию сразу после загрузки DOM для начального добавления
обработчиков
document.addEventListener('DOMContentLoaded', function() {
    bindMapItemClickEvent();
});

```

```

function bindMapItemClickEvent() {
    // Получаем все элементы, соответствующие переданному селектору
    const mapItems = document.querySelectorAll('.map-item');

    // Добавляем обработчик событий click для каждого элемента
    mapItems.forEach(function(item) {
        item.addEventListener('click', function() {
            // Вызываем карту на редактирование
            console.log('click .map-item');
            // Находим внутри.map-item изображение с классом.map-mini-
            image
            const miniImage = this.querySelector('.map-mini-image');

            if (miniImage) {
                // Получаем путь к изображению.map-mini-image
                const newSrc = miniImage.src;

                // Находим главное изображение .map и меняем его src
                // Это нужно для корректного размера элемента .map
                const mainImage = document.querySelector('.map img');
                if (mainImage) {
                    //mainImage.style.display = ""; // Показываем
                    изображение
                    mainImage.src = newSrc;
                    //mainImage.style.width = `auto`;
                    //mainImage.style.height = `auto`;
                    //mainImage.style.visibility = "hidden"; // Скрываем
                    изображение, сохраняя его в DOM
                    const { width, height } = mainImage;
                    const graph = document.getElementById('graph');
                    if (graph) {
                        graph.setAttribute('viewBox', `0 0 ${width}
                        ${height}`);
                    }
                }

                // Находим карту и меняем её фоновое изображение
                const mapElement = document.querySelector('.map');
                if (mapElement) {

```

```

        //mapElement.style.backgroundImage
        `url(${newSrc})`;

        // Находим главное изображение и получаем его
        естественные размеры
        const mainImage = document.querySelector('.map img');
        if (mainImage) {
            //const { width, height } = mainImage;

            // Применяем размеры к элементу.map
            //mapElement.style.width = `${width}px`;
            //mapElement.style.height = `${height}px`;

            //mainImage.style.width = `0px`;
            //mainImage.style.height = `0px`;
            //mainImage.style.display = "none"; // Скрываем
            изображение
        }
    }
    });
});
}

$( document ).ready( function() {

    createCookie('user_id', 1, 5); // сохраняем user_id на 5 дней
    спрятать_виджеты();

    // Устанавливаем события на кнопки меню
    $(".menu > .btn[data-widget], button[data-widget]").click( function()
    {
        //alert('click');
        console.log('menu click');
        показать_виджет(this.dataset.widget);
    });

    // Настраиваем виджеты

```

```

    //const widgetElement = document.querySelector('.widget[data-
widget="admin-maps"][data-role="admin"]');
    //const widgetElement = document.querySelector('.widget');
    //настройка_виджета(widgetElement);
    // Настройка виджетов для всех элементов с классом.widget
    document.querySelectorAll('.widget').forEach(widgetElement => {
        настройка_виджета(widgetElement);
    });

    $("#main").show();

});

function показать_виджет(widget, role="all")
{
    console.log('function показать_виджет');
    спрятать_виджеты();
    widgetElement = document.querySelector('.widget[data-
widget="'+widget+'"]');
    $(widgetElement).show();
    выполнить_onshow(widgetElement);
}

// убираем все виджеты
function спрятать_виджеты(role="all")
{
    if (role == 'all')
    {
        $(".widget").hide();
    } else {
        $(".widget[data-role='"+role+"']").hide();
    }
}

// Объявление глобальной переменной url_db
// Эта переменная будет содержать начальную часть URL для запроса к PHP-
скрипту
// Она объявляется вне любых функций, что делает её доступной во всем
скрипте

```

```

let url_db = 'http://airsouffle.ru/kostya/diplom/db/index.php';

// Функция для получения изображений по ID пользователя
function fetchImagesByUserId(userId, selector) {

    const sql_query = `SELECT id, название, картинка FROM пути`;
    const apikey = 'lebedev';

    // Кодирование userId для безопасной передачи в запросе
    // Это предотвращает проблемы с символами, которые могут нарушить
    структуру запроса
    const encodedUserId = encodeURIComponent(userId);

    // Формирование тела запроса в формате JSON
    // Здесь мы используем объект, где ключ 'query' содержит наш SQL-
    запрос
    const requestBody = JSON.stringify({
        //query: `SELECT * FROM пути WHERE user_id=${encodedUserId}`,
        query: sql_query,
        key: apikey
    });

    const requestData = new FormData();
    //data.append("json_data", requestBody);
    requestData.append('query', sql_query);
    requestData.append('key', apikey);

    // Определение настроек для fetch
    // Метод POST указывает, что мы хотим отправить данные на сервер
    // Headers содержат тип контента, который мы отправляем
    (application/x-www-form-urlencoded)
    // Body содержит сами данные запроса, преобразованные в строку
    const requestOptions = {
        method: 'POST',
        //headers: { 'Content-Type': 'application/json' },
        //body: requestBody
        body: requestData
    };

```

```

// Выполнение асинхронного POST-запроса к указанному URL
// Первый аргумент - URL, на который отправляется запрос
// Второй аргумент - настройки запроса
fetch(url_db, requestOptions)
.then(response => {
    // Проверка успешности выполнения запроса
    // Если запрос выполнен успешно, преобразование ответа в JSON
    return response.json();
})
.then(data => {
    // Проверка статуса ответа
    // Если статус 'success', значит данные были получены успешно
    if (data.status === 'success') {
        // Получение ссылки на элемент списка изображений в HTML-
документе

        const mapList = document.querySelector(selector);
        // Очистка списка перед добавлением новых элементов
        mapList.innerHTML = '';

        // Проход по каждому элементу данных, полученным в ответе
        data.data.forEach(item => {
            // Создание нового div элемента для каждой карты
            const col = document.createElement('div');
            col.className = 'col'; // Присвоение класса для
стилизации

            // Создание нового div элемента для каждой карты
            const card = document.createElement('div');
            card.className = 'card window map-item'; // Присвоение
класса для стилизации

            const card_body = document.createElement('div');
            card_body.className = 'card-body'; // Присвоение класса
для стилизации

            /*
            <div class="card" style="width: 18rem;">
            
            <div class="card-body">

```

```

        <h5 class="card-title">Card title</h5>
        <p class="card-text">Some quick example text to build on
the card title and make up the bulk of the card's content.</p>
        <a href="#" class="btn btn-primary">Go somewhere</a>
    </div>
</div>
*/
// Создание img элемента и установка его источника и alt
текста

const img = document.createElement('img');
img.src = item.картинка;
img.alt = item.название;
img.className = 'card-img-top map-mini-image'; //
Присвоение класса для стилизации

// Создание p элемента для подписи к изображению
const caption = document.createElement('p');
caption.textContent = 'Править карту
''+item.название+''';
caption.className = 'card-text text-center m-0';

const link = document.createElement('a');
link.href = '#';
link.dataset.map_id = item.id;
link.className = 'stretched-link'; // Присвоение класса
для стилизации

// Добавление img и p элементов в div
col.appendChild(card);
card.appendChild(img);
card.appendChild(card_body);
card_body.appendChild(caption);
card_body.appendChild(link);

// Добавление div с изображением и подписью в список
изображений

mapList.appendChild(col);
});

```

```

        bindMapItemClickEvent(); // привязываем событие на нажатие
карточки
    } else {
        // Вывод сообщения об ошибке, если данные не были получены
успешно
        console.error('Ошибка при получении данных:', data.message);
    }
})
.catch(error => {
    // Вывод сообщения об ошибке, если произошла ошибка при
выполнении запроса
    console.error('Ошибка при выполнении запроса:', error);
});
}

function загрузить_карты(user_id=1)
{
    console.log('function загрузить_карты, user_id=' + user_id);
    fetchImagesByUserId(user_id, '.widget[data-widget="admin-maps"]
.maps'); // Загружаем карты с сервера
}

// Функция для создания куки
function createCookie(name, value, days) {
    var expires = '',
    date = new Date();
    if (days) {
        date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000));
        expires = '; expires=' + date.toGMTString();
    }
    document.cookie = name + '=' + value + expires + '; path=/';
}

// Функция для чтения куки по имени
function readCookie(name) {
    var nameEQ = name + '=',
    allCookies = document.cookie.split(';'),
    i,

```

```

    cookie;
    for (i = 0; i < allCookies.length; i += 1) {
        cookie = allCookies[i];
        while (cookie.charAt(0) === ' ') {
            cookie = cookie.substring(1, cookie.length);
        }
        if (cookie.indexOf(nameEQ) === 0) {
            return cookie.substring(nameEQ.length, cookie.length);
        }
    }
    return null;
}

// Функция для удаления куки по имени
function eraseCookie(name) {
    createCookie(name, '', -1);
}

/*
// Пример использования
var idhome = '123';
var room = 'Apt 4B';
var renthome = 'Yes';

// Сохраняем переменные в куки
createCookie('idhome', idhome, 7);
createCookie('room', room, 7);
createCookie('renthome', renthome, 7);

// Читаем куки
console.log('ID Home:', readCookie('idhome'));
console.log('Room:', readCookie('room'));
console.log('Rental Home:', readCookie('renthome'));

// Удаляем куки
eraseCookie('idhome');
eraseCookie('room');
eraseCookie('renthome');
*/

```

```

// Функция для выполнения процедур
function executeProcedures(proceduresArray) {
    proceduresArray.forEach(procedure => {
        // Здесь должна быть логика выполнения процедуры
        // Например, вызов функции с именем procedure, если они
определены в глобальном scope
        window[procedure] && window[procedure]();
    });
}

function выполнить_onshow(widgetElement) {
    // Извлечение значения атрибута data-onshow
    const dataOnShowValue = widgetElement.getAttribute('data-onshow');
    // Проверка, есть ли значение атрибута
    if (dataOnShowValue) {
        // Разделение строки на отдельные процедуры
        const proceduresArray =
dataOnShowValue.split(';').filter(Boolean);
        // Выполняем процедуры
        executeProcedures(proceduresArray);
    }
}

// Функция настройка_виджета для выполнения процедур, указанных в
атрибуте data-onshow элемента
function настройка_виджета(widgetElement) {
    // Логирование имени виджета для отладки
    console.log(`Настройка виджета: ${widgetElement.className}`);
    // Вывод значения атрибута data-widget в консоль для отладки
    console.log(`Настройка виджета с data-widget:
${widgetElement.dataset.widget}`);

    // Извлечение значения атрибута data-onshow
    const dataOnShowValue = widgetElement.getAttribute('data-onshow');

    // Проверка, есть ли значение атрибута
    if (dataOnShowValue) {
        // Разделение строки на отдельные процедуры

```

```

const      proceduresArray      =
dataOnShowValue.split(';').filter(Boolean);

// Проверяем, является ли элемент видимым
if (window.getComputedStyle(widgetElement).display !== 'none') {
    // Если элемент уже видим, немедленно выполняем процедуры
    executeProcedures(proceduresArray);
} else {
    // Если элемент скрыт, добавляем обработчик события
visibilitychange
    widgetElement.addEventListener('visibilitychange', () => {
        if (document.visibilityState === 'visible') {
            // Если документ становится видимым, выполняем
процедуры
            executeProcedures(proceduresArray);
        }
    });
    // Или, если вы хотите привязать к этому событию конкретный
элемент, используйте селектор этого элемента вместо 'body'
    $(widgetElement).on('visibilitychange', function() {
        if (document.visibilityState === 'visible') {
            // Код, который следует выполнить, когда элемент
становится видимым
            console.log('Элемент стал видимым');
        }
    });
}
console.log('Атрибут data-onshow присутствует у элемента.');
```

```

} else {
    console.warn('Атрибут data-onshow отсутствует у элемента.');
```

```

}

function обновить_поле(selector, value)
{
    const elements = document.querySelectorAll(selector);
    elements.forEach(function(element) {
        try {
            element.value = value;

```

```

    } catch (e) {

    } finally {

    }

  });
}

```

Листинг Г.15 – forms.js

```

// назначаем обработчик для формы авторизации
document.addEventListener('DOMContentLoaded', function() {
  // Находим все формы на странице
  const forms = document.querySelectorAll('.auth_form');

  // Добавляем обработчики событий
  forms.forEach(form => {
    form.addEventListener('submit', function(event) {
      event.preventDefault(); // Предотвращаем стандартное
// поведение формы

      console.log('auth_form submit');
      //alert('auth_form submit');

      // Создаем объект FormData для сбора данных из формы
      let formData = new FormData(this);

      // Преобразуем FormData в JSON
      let object = {};
      formData.forEach(function(value, key){
        object[key] = value;
      });

      object.логин = object.логин.trim();

      let json = JSON.stringify(object);

      console.log(json);
    });
  });
}

```

```

const api_url =
`${url_db}?query=select+*+from+пользователи+where+логин='${object.логин}'
+and+пароль='${object.пароль}'&key=lebedev`;

console.log(api_url);

// Отправляем данные на сервер
fetch(api_url,
{
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: json
})
.then(response => response.json())
.then(data => {
  console.log('Данные успешно получены:', data);
  // Проверяем полученные данные
  if (data.data.length == 1) {
    // Авторизация прошла успешно
    iziToast.success({
      title: 'Вход в систему',
      message: 'Добро пожаловать, ' +
data.data[0].фамилия + ' ' + data.data[0].имя + ' ' +
data.data[0].отчество,
    });
    role = data.data[0].роли.split(' ')[0];
    user_id = data.data[0].id;
    user_name = data.data[0].логин;
    обновить_поле('input[name="логин"]', user_name);
    обновить_поле('input[name="фамилия"]',
data.data[0].фамилия);
    обновить_поле('input[name="имя"]', data.data[0].имя);
    обновить_поле('input[name="отчество"]',
data.data[0].отчество);
    обновить_поле('input[name="отдел"]',
data.data[0].отдел);
    updateVisibility();
  }
});

```

```

    } else {
        // Авторизация не удалась
        iziToast.error({
            title: 'Вход в систему',
            message: 'Имя пользователя или пароль указаны
неверно',

        });
    }
})
.catch((error) => {
    console.error('Ошибка при получении данных:', error);
    iziToast.error({
        title: 'Ошибка при получении данных',
        message: error.toLocaleString(),

    });
});
});
});
});
});

```

```

// назначаем обработчик для формы регистрации
document.addEventListener('DOMContentLoaded', function() {
    // Находим все формы на странице
    const forms = document.querySelectorAll('.registration_form');

    // Добавляем обработчики событий
    forms.forEach(form => {
        form.addEventListener('submit', function(event) {
            event.preventDefault();    //      Предотвращаем      стандартное
поведение формы

            console.log('registration_form submit');
            //alert('registration_form submit');

            // Создаем объект FormData для сбора данных из формы
            let formData = new FormData(this);

            // Преобразуем FormData в JSON

```

```

let object = {};
formData.forEach(function(value, key){
    object[key] = value;
});

object.логин = object.логин.trim();

let json = JSON.stringify(object);

console.log(json);

const api_url =
`${url_db}?query=insert+into+пользователи+(логин,пароль,роли)+values+('${object.логин}', '${object.пароль}', 'user')&key=lebedev`;

console.log(api_url);

// Отправляем данные на сервер
fetch(api_url,
{
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: json
})
.then(response => response.json())
.then(data => {
    console.log('Данные успешно получены:', data);

    if (data.status == 'success') {
        // Проверяем полученные данные
        if (data.data.length == 0) {
            // Регистрация прошла успешно
            iziToast.success({
                title: 'Регистрация выполнена',
                message: 'Добро пожаловать в систему',
            });
            role = 'user';
        }
    }
});

```

```

        user_id = -1;
        user_name = object.логин;
        обновить_поле('input[name="логин"]', user_name);
        updateVisibility();
    }
}

if (data.status == 'error') {
    console.error('Ошибка из серверной части:',
data.message);

    iziToast.error({
        title: 'Ошибка регистрации',
        message: 'Имя пользователя уже занято',
    });
}

})

.catch((error) => {
    console.error('Ошибка при получении данных:', error);
    iziToast.error({
        title: 'Ошибка при получении данных',
        message: error.toLocaleString(),
    });
});

});

});

});

// назначаем обработчик для профиля пользователя
document.addEventListener('DOMContentLoaded', function() {
    // Находим все формы на странице
    const forms = document.querySelectorAll('.profile_form');

    // Добавляем обработчики событий
    forms.forEach(form => {
        form.addEventListener('submit', function(event) {
            event.preventDefault(); // Предотвращаем стандартное
поведение формы

```

```

console.log('profile_form submit');
//alert('profile_form submit');

// Создаем объект FormData для сбора данных из формы
let formData = new FormData(this);

// Преобразуем FormData в JSON
let object = {};
formData.forEach(function(value, key){
    object[key] = value;
});

let json = JSON.stringify(object);

console.log(json);

const api_url = `${url_db}?query=update пользователи set
пароль='${object.пароль}',
фамилия='${object.фамилия}', имя='${object.имя}', отчество='${object.отчест
во}', отдел='${object.отдел}' where логин='${object.логин}'&key=lebedev`;

console.log(api_url);

// Отправляем данные на сервер
fetch(api_url,
{
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: json
})
.then(response => response.json())
.then(data => {
    console.log('Данные успешно получены:', data);

    if (data.status == 'success') {
        // Проверяем полученные данные
        if (data.data.length == 0) {

```

```

        // Регистрация прошла успешно
        iziToast.success({
            title: 'Обновление выполнено',
            message: 'Данные профиля успешно обновлены',
        });
    }
}

if (data.status == 'error') {
    console.error('Ошибка из серверной части:',
data.message);

    iziToast.error({
        title: 'Ошибка обновления',
        message: 'Данные профиля успешно обновлены',
    });
}

})

.catch((error) => {
    console.error('Ошибка при получении данных:', error);
    iziToast.error({
        title: 'Ошибка при получении данных',
        message: error.toLocaleString(),
    });
});

});

});
});

```

Листинг Г.16 – dijkstra.js

```

class PriorityQueue {
    constructor() {
        this.collection = [];
    }

    enqueue(element) {
        if (this.isEmpty()) {
            this.collection.push(element);
        } else {
            let added = false;
            for (let i = 0; i < this.collection.length; i++) {
                if (element[1] < this.collection[i][1]) {
                    this.collection.splice(i, 0, element);
                }
            }
        }
    }
}

```

```

        added = true;
        break;
    }
}
if (!added) {
    this.collection.push(element);
}
}

}

dequeue() {
    return this.collection.shift();
}

isEmpty() {
    return (this.collection.length === 0) ? true : false;
}
}

function dijkstra(graph, startNode) {
    const distances = {};
    const prev = {};
    const pq = new PriorityQueue();

    // Инициализация расстояний от начального узла до всех остальных
    узлов
    for (const node in graph) {
        distances[node] = Infinity;
        prev[node] = null;
    }
    distances[startNode] = 0;
    pq.enqueue([startNode, 0]);

    while (!pq.isEmpty()) {
        const shortestPath = pq.dequeue();
        const currentNode = shortestPath[0];

        for (const neighbor in graph[currentNode]) {
            const altDistance = distances[currentNode] +
graph[currentNode][neighbor];
            if (altDistance < distances[neighbor]) {
                distances[neighbor] = altDistance;
                prev[neighbor] = currentNode;
                pq.enqueue([neighbor, altDistance]);
            }
        }
    }

    return { distances, prev };
}

/*
// Пример использования:
graph = {
    A: { B: 1, C: 3 },
    B: { A: 1, C: 2 },
    C: { A: 3, B: 2 }
};

```

```

result = dijkstra(graph, 'A');
console.log('Distances:', result.distances);
console.log('Previous nodes:', result.prev);

// Пример использования с индексными переменными:
graph = {
  0: { 1: 1, 2: 3 },
  1: { 0: 1, 2: 2 },
  2: { 0: 3, 1: 2 },
  3: { 4: 1, 5: 3 },
  4: { 3: 1, 5: 2 },
  5: { 3: 3, 4: 2 },
  6: { 7: 1, 8: 3 },
  7: { 6: 1, 8: 2 },
  8: { 6: 3, 7: 2 },
  9: { 8: 1 }
};

result = dijkstra(graph, 0);
console.log('Distances:', result.distances);
console.log('Previous nodes:', result.prev);
*/

```

Приложение Д
(рекомендуемое)
Результаты тестирования серверной части

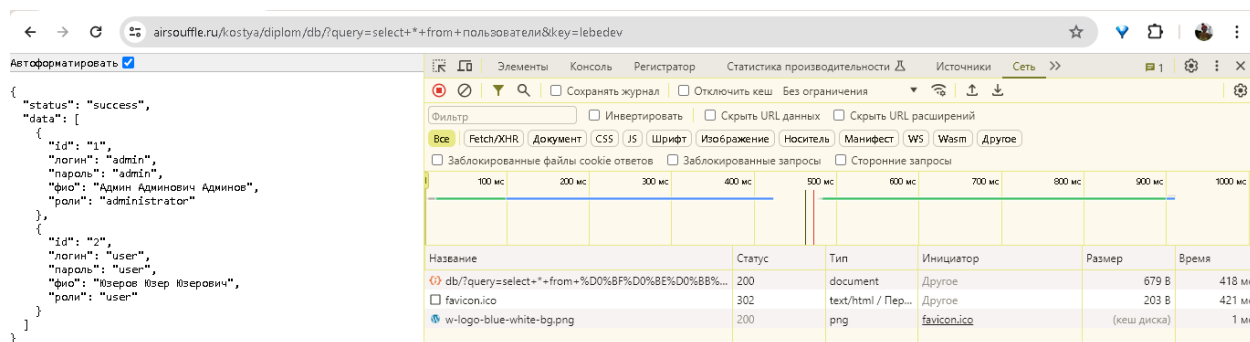


Рисунок Д.1 – Получение списка пользователей

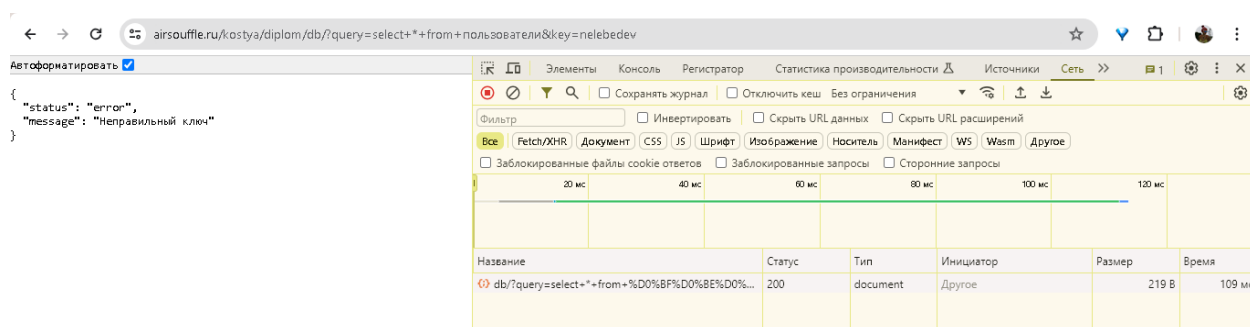


Рисунок Д.2 – Получение списка пользователей с неверным ключом API

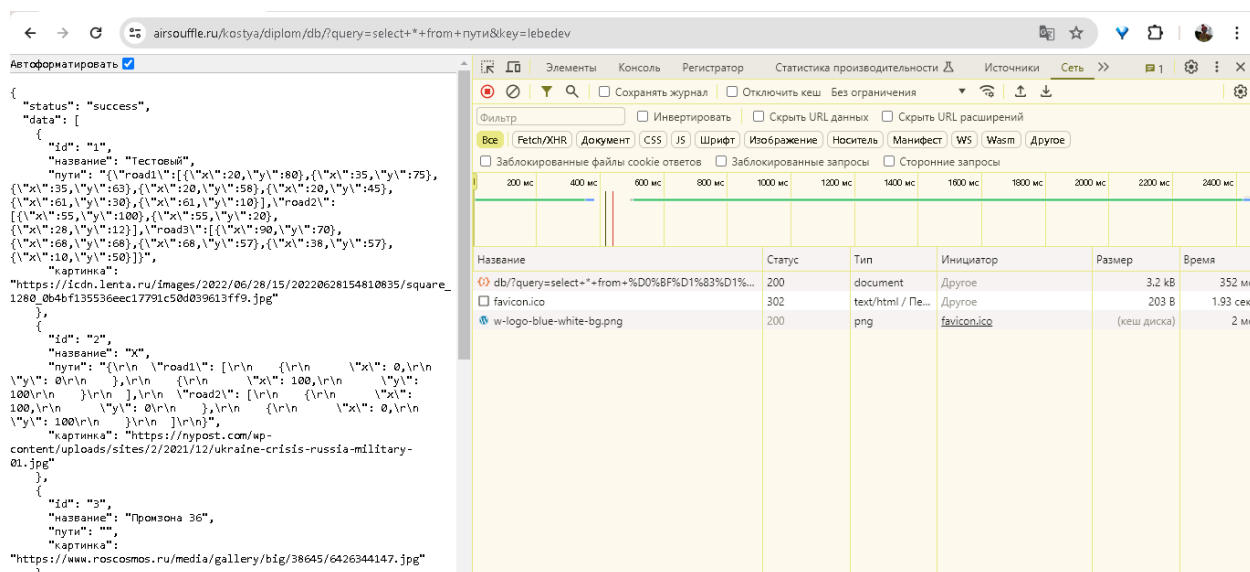


Рисунок Д.3 – Получение списка путей

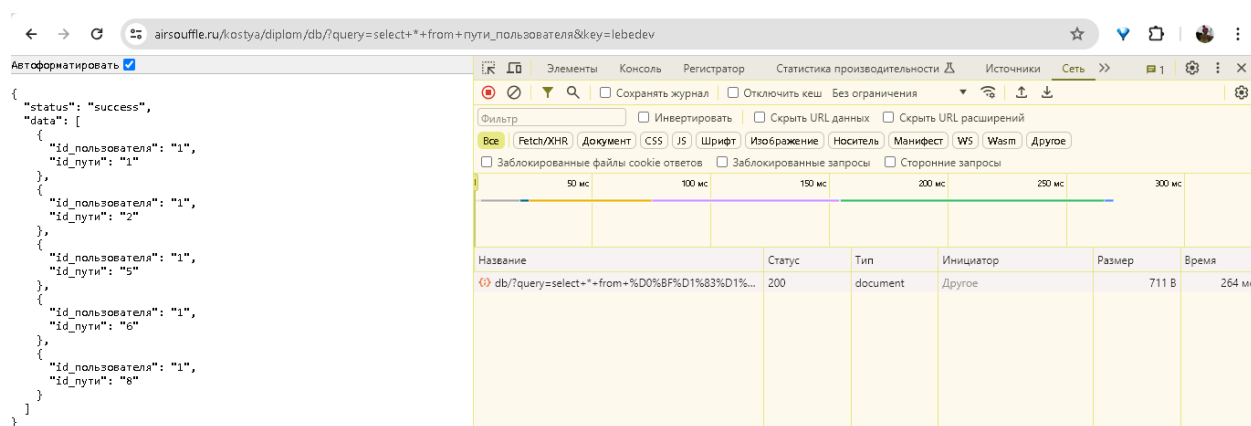


Рисунок Д.4 – Получение списка сохраненных путей пользователей

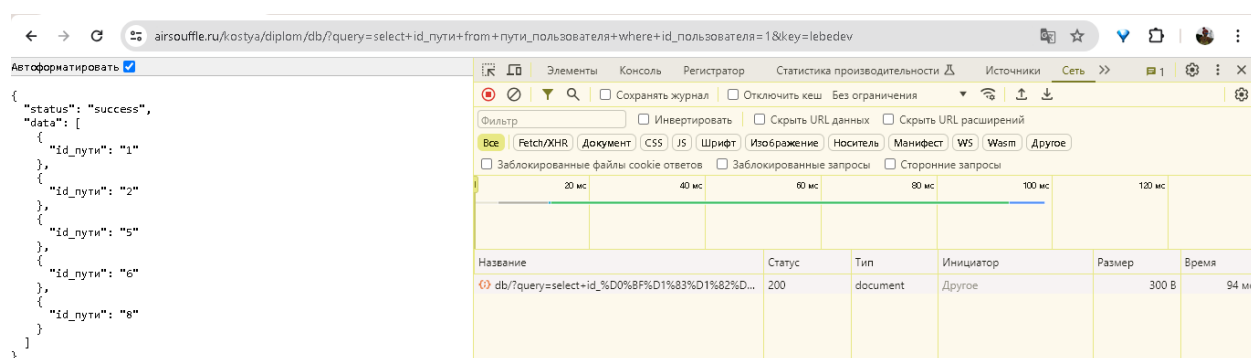


Рисунок Д.5 – Получение списка пользовательских путей

Приложение Е
(рекомендуемое)
Результаты тестирования приложения

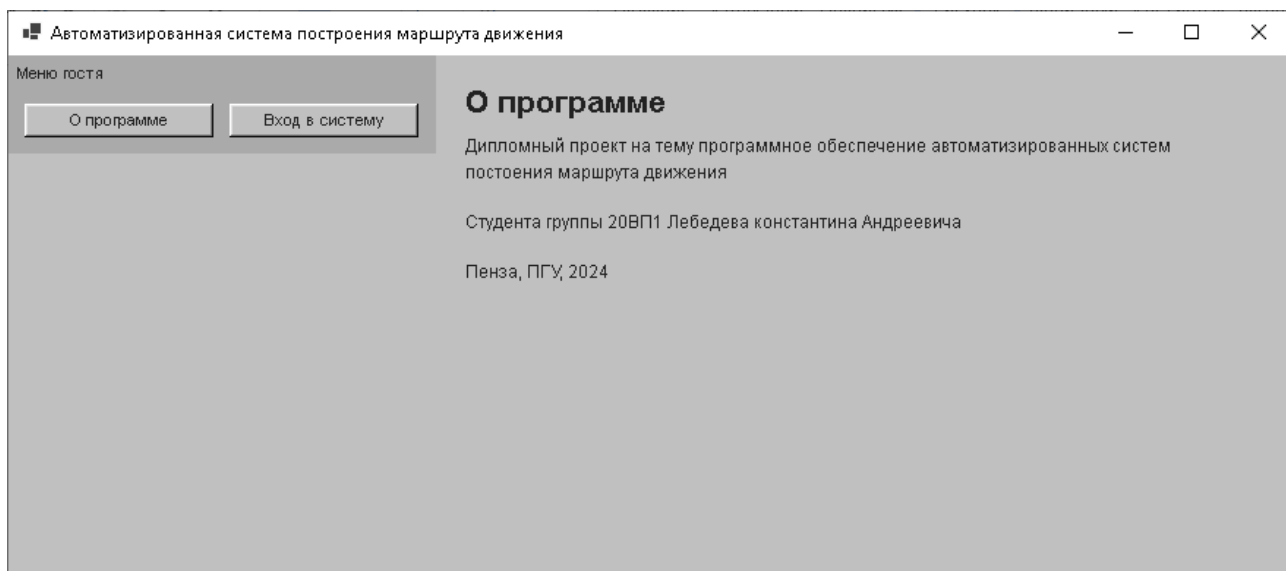


Рисунок Е.1 – Информация о программе и разработчике

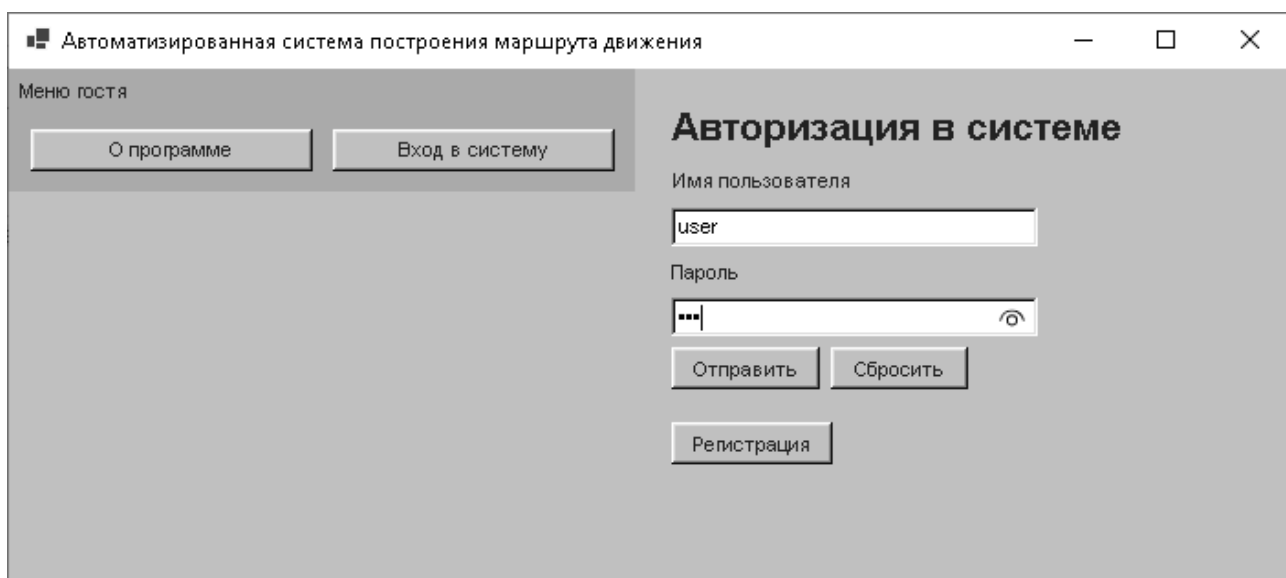


Рисунок Е.2 – Авторизация в системе

Автоматизированная система построения маршрута движения

Меню гостя

О программе Вход в систему

Регистрация в системе

Имя пользователя
konstantin

Пароль

Отправить Сбросить

Рисунок Е.3 – Регистрация в системе

Автоматизированная система построения маршрута движения

Меню пользователя

Выбрать карту Управление профилем

Карточка пользователя

Имя пользователя
konstantin

Пароль

Фамилия
Iebedev

Имя
konstantin

Отчество
andreevich

Отдел/подразделение
VT

Отправить Сбросить

Рисунок Е.4 – Карточка пользователя

Автоматизированная система построения маршрута движения

Меню администратора

Управление картами Управление пользователями

Добавление пользователя

Имя пользователя
user2

Пароль

Отправить Сбросить

Список пользователей

Имя	Пароль	Обновить	Удалить
user	*****	Обновить	Удалить
konstantin	*****	Обновить	Удалить
user3	*****	Обновить	Удалить

Рисунок Е.5 – Добавление и редактирование списков пользователей



Рисунок Е.6 – Главная форма редактирования карты администратором

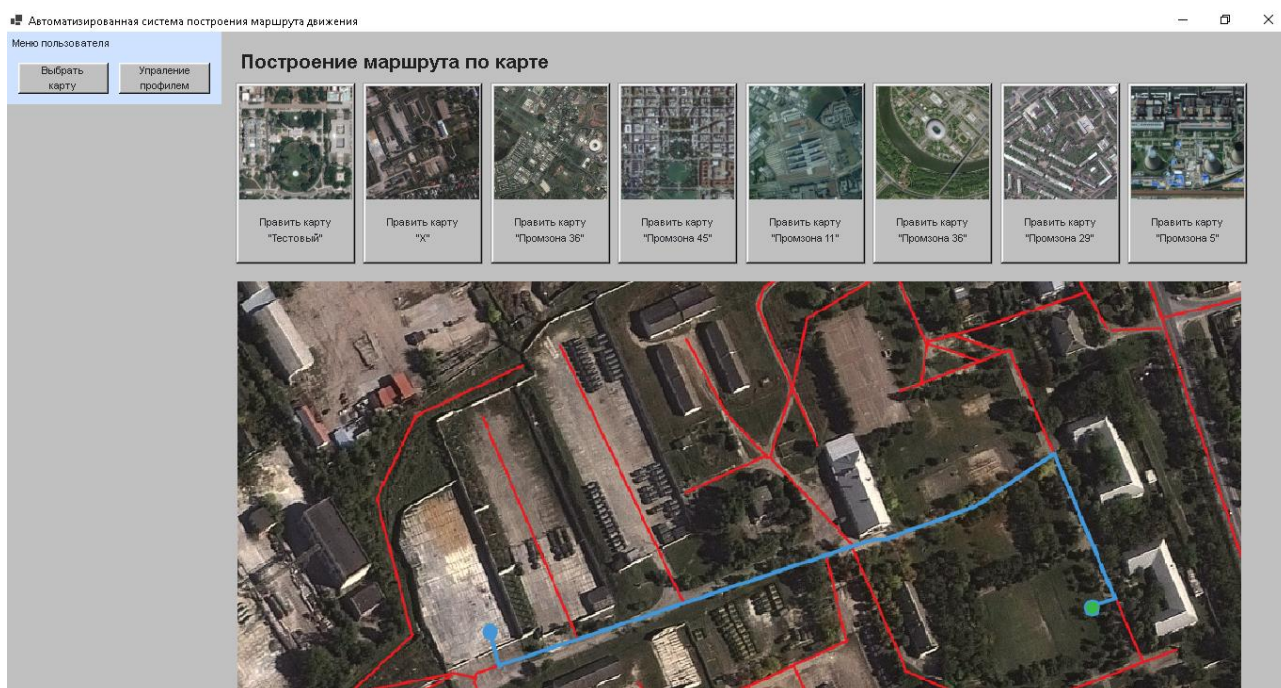


Рисунок Е.7 – Построение оптимального маршрута движения