ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННОХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ОТОБРАЖЕНИЯ ГОРОДСКОГО ТРАНСПОРТА НА КАРТЕ

Выпускная квалификационная работа
обучающегося по направлению подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения,
группы 07001302
Катышева Антона Евгеньевича

Научный руководитель
доцент
Чашин Ю.Г.

БЕЛГОРОД 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одной из главных составляющих отраслей городской инфраструктуры является развитие транспортной системы. Стабильность функционирования городской транспортной системы способствует повышению экономического, культурного, социального и духовного уровней жизни, увеличению работоспособности организаций, учреждений и предприятий, своевременным обеспечением жителей товарами и услугами.

Из перечня пассажирских перевозок, больше всего занимают внутригородские перевозки, примерно 70-80%. С каждым годом растет потребность в предоставлении населению качественных услуг пассажирских перевозок, что приводит к росту количество транспортных средств и остановок общественного автотранспорта на долю города, а соответственно к перегрузке транспортной сети, снижению средней скорости движения и ухудшению уровня обслуживания транспортных средств. Исходя из этого, становится важен выбор оптимального маршрута и транспорта передвижения для экономии энергии, ресурсов и времени.

Всё выше сказанное приводит к возникновению потребности в автоматизированной информационной системы, работающей с данными о пассажирских перевозках в пределах города.

Данная система будет полезна как для организаций, осуществляющих пассажирские перевозки, позволяющая наблюдать, анализировать, формировать и изменять маршруты движения общественного транспорта, так и для населения города, помогая найти пути к определенному месту назначения, заранее распланировать маршрут передвижения по городу, получить дополнительную информацию об услугах пассажирских перевозок, сэкономить время.

Благодаря развитию информационных технологий, создание подобных системы становится не сложной задачей.

Объектом исследования данной работы является модель автотранспортной системы города.

Предметом исследования является автоматизация процесса формирования и представления данных о транспортной инфраструктуре в пределах города.

Целью выпускной квалификационной работы является рассмотрение модели системы городского пассажирского транспорта, формирования информационной составляющей данной модели и её представление, разработка автоматизированной информационной системы согласно тематике.

Задачами данной работы в связи с указанной целью являются:

- проанализировать существующие решения и выявить недостатки;
- установить требования к собственному решению;
- спроектировать информационную систему;
- разработать программную реализацию автоматизированной информационной системы;
- исследовать полученный продукт на работоспособность и соответствие с установленными требованиями.

Структура работы обусловлена предметом, целью и задачами исследования. Работа состоит из введения, трех глав и заключения.

Введение раскрывает актуальность, определяет объект, предмет, цель и задачи исследования, раскрывает значимость работы.

В первом разделе рассматриваются основные понятия, теоретические аспекты описанной проблемы, анализ и постановка требований к результирующему продукту.

Второй раздел проекта описывает выбор программного обеспечения для реализации и проектирование базы данных.

В третьем разделе описывается разработка базы данных и программная реализация автоматизированной информационной системы в виде веб-приложения и демонстрация интерфейса.

В четвертом разделе осуществляется подготовка тестовых данных и проведение испытаний для проверки работоспособности разработанной системы.

Выпускная квалификационная работа состоит из 52 листов, содержит 22 рисунка, 4 таблицы, 16 листингов.

1 ПОСТАНОВКА ЗАДАЧИ

Перед рассмотрением предметной области стоит ознакомиться с основные понятия в рамках данной работы.

Информационная система (ИС) — система, предоставляющая возможность хранения, обработки, поиска информации, и соответствующие организационные ресурсы, для обеспечения и распространения информации.

Информационная система предназначена для обеспечения своевременной надлежащей информацией определенных людей, другими словами для удовлетворения потребностей в конкретной информации в рамках определенной предметной области. Результатом функционирования информационных систем при этом является информационная продукция — информационные услуги, базы данных, нормативные и прочие документы, информационные массивы.

Для автоматизированных систем, используемых в управлении, исследованиях, проектировании и др., смысл которых заключается в обработке информации, дано такое определение:

Автоматизированная система (в информационных технологиях) — система, реализующая информационную технологию выполнения установленных функций с помощью персонала и комплекса средств автоматизации.

В этом случае автоматизированные системы рассматриваются как информационные системы. В целом АС — это система, состоящая из персонала и комплекса средств автоматизации его деятельности и реализует информационную технологию выполнения установленных функций.

База данных – это средство для реляционного и эффективного хранения информации. Такая база обеспечивает надежную защиту данных от случайной потери или порчи, экономно использует ресурсы (как человеческие, так и

технические) и снабжена механизмами поиска фильтрация и предоставления информации пользователю в удобном для него виде.

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

Клиент-серверная технология – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемые клиентами. Физически клиент и сервер — это программное обеспечение. Обычно они взаимодействуют через компьютерную сеть посредством сетевых протоколов и находятся на разных вычислительных машинах, но могут выполняться также и на одной машине.

API – application programming interface – (программный интерфейс приложения, интерфейс программирования приложений, интерфейс прикладного программирования) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

Веб-приложение — клиент-серверное приложение, в котором клиентом выступает браузер, а сервером — веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети.

1.1 Системный анализ предметной области

На начальном шаге проектирования информационной системы нужно выполнить анализ предметной области, т.е. выделить объекты предметной области и их связи между собой.

Во время выбора структуры и состава предметной области возможны несколько подходов: предметный и функциональный.

Функциональный подход реализует принцип движения «от условий задачи» и используется, когда определен комплекс задач, для обслуживания которых создается ИС. В этом случае можно определить минимальный набор объектов предметной области, которые необходимо описать.

В предметном подходе объекты предметной области определяются с таким расчетом, чтобы их можно было использовать при реализации множества разнообразных задач.

Чаще всего используют комбинацию этих двух подходов, который, с одной стороны, ориентирован на функциональные потребности пользователей или конкретные задачи, а с другой стороны, учитывает возможность наращивания новых приложений.

Опишем предметную область программного продукта, представленного в данной работе.

В рассматриваемой системе предполагается работа двух типов людей: обслуживающий персонал и пользователи.

Обслуживающий персонал формирует базовую составляющую информационных данных, имеет возможность её дополнять, изменять, удалять, следит за работоспособностью системы и формирует отчеты при необходимости.

Пользователи получают доступ к различным данным системы, без возможности каким-либо образов воздействовать на неё.

Общественный транспорт осуществляет посредничество между организациями, предоставляющими транспортные услуги и жителями города, пользующимися этими услугами.

Система должна предусмотреть режим ведения системного каталога, отражающего перечень транспорта, к которому имеются маршруты движения в базе. Внутри базы в систематическом каталоге могут быть уникальные

идентификационные номера и полные наименования. Каждый транспорт в базе может присутствовать в нескольких экземплярах. К объектам и атрибутам, позволяющим охарактеризовать отдельные экземпляры транспорта можно отнести:

1. Идентификационный номер транспорта
2. Тип транспорта
3. Модель транспорта
4. Количество посадочных мест
5. Компания, предоставляющая услуги
6. Государственный номер
7. Координаты местоположения

Следующий объект – маршрут движения транспортных средств, который следует по определенной траектории. В данном объекте можно выделить следующие атрибуты:

1. Идентификационный номер маршрута
2. Название маршрута
3. Список остановок на маршруте
4. Траектория маршрута
5. Интервал движения транспорта
6. Стоимость проезда

Также объектом будет являться остановка, которая определяется названием, направлением – на какой стороне улицы находится, и географическими координатами местоположения.

Все выделенные объекты связываются через отдельный объект – «Назначение», который каждому экземпляру транспорта ставит в соответствии маршрут движение. Таким образом на каждый маршрут можно назначить несколько единиц транспорта.

Необходимо учесть время назначения для исключения ситуации назначения определенного экземпляра транспорта на несколько маршрутов.

Это позволит по последнему назначению однозначно определить маршрут движения, по которому следуют транспорт и вести учёт маршрутов, на который когда-либо назначался транспорт.

Из рассмотренного представления объектов предметной области необходимо создание автоматизированной информационной системы для управления данными о них. Один из вариантов – это создание веб-приложения с использованием клиент-серверной технологии.

Такая система будет состоять из следующих модулей:

- серверная часть (база данных);
- пользовательский интерфейс (веб-приложение);

Благодаря такой структуре, пользователи не будут зависеть от конкретной операционной системы, что будет является кроссплатформенным решением. Также это предоставит быстрый доступ к информации, обновленной обслуживающим персоналом.

На рис. 1.1. изображена архитектура веб-приложения, согласно которому, клиенты через пользовательский интерфейс, по средствам интернета могут получить доступ к информации, хранящейся на серверах в виде базы данных.

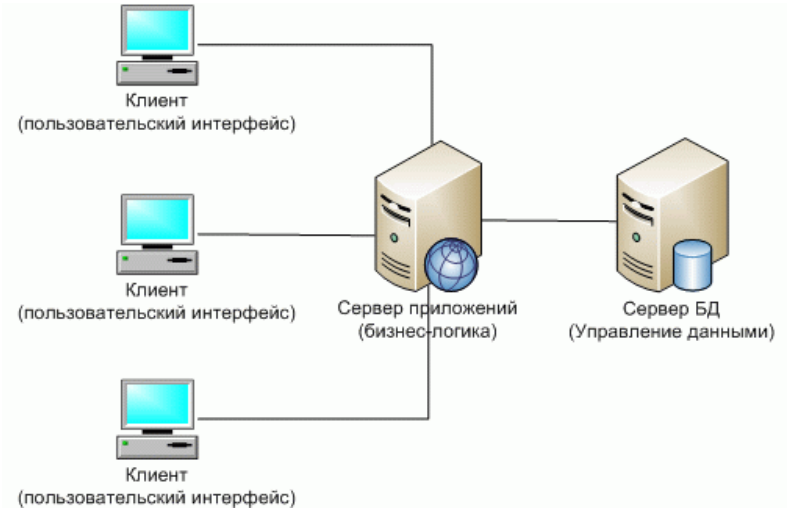

Рис. 1.1. Архитектура веб-приложения

1.2 Обзор информационных систем, реализующих подобные задачи

В рамках темы выпускной квалификационной работы было найдено и выбрано для рассмотрения несколько систем, реализующих решение определенных задач, в частности:

- Интерактивная информационная система, отображающая положение транспорта на карте города Томска [http://rasp.tomsk.ru];
- Интернет сервис центра управления городским автоэлектротранспортом города Новосибирска [https://nskgortrans.ru/];
- Интернет сервис «Умный транспорт» города Рязань [http://умный-транспорт.рф].

1.2.1 Центр организации и контроля пассажироперевозок г. Томска

Муниципальное бюджетное учреждение Города Томска «Центр организации и контроля пассажироперевозок» создано в июне 2012 г. в соответствии с постановлениями администрации Города Томска от 06.02.2012 №103 и от 11.04.2012 № 358. Полноценно вести хозяйственную деятельность учреждение начало с 01.10.2012 г.

Уполномоченным отраслевым органом (Учредителем) в отношении МБУ «Центр организации и контроля пассажироперевозок» определены департамент городского хозяйства администрации Города Томска и департамент экономического развития и управления муниципальной собственностью администрации Города Томска.

Основными задачами на момент создания учреждения были:

- обеспечение эффективного контроля и регулирования пассажирских перевозок с применением информационных технологий;

- внедрение современных систем управления перевозками;
- осуществление мониторинга и контроль навигационных параметров транспортных средств.

С 01.03.2014 г. на базе МБУ «Центр организации и контроля пассажироперевозок» создан единый навигационно-информационный центр Города Томска.

Главная страница данной системы (рис. 1.2.) представляет собой карту города и расположенные на ней функциональные панели.

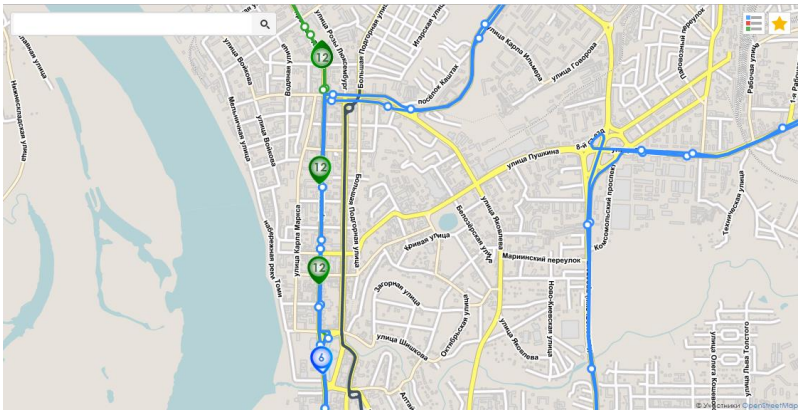

Рис. 1.2. Главная страница информационной системы г. Томска

В левом верхнем углу расположена панель поиска, по ключевым словам, при помощи которого можно найти остановку или маршрут автотранспорта, что изображено на рисунке 1.3. слева.

В правом верхнем углу расположена кнопка, которая раскрывает панель с перечнем маршрутов, по которым осуществляются перевозки (рис. 1.3. по центру). На данной панели можно выбрать маршруты, транспорт которых будет отображаться на карте, посмотреть траекторию маршрута и список остановок (рис. 1.3. справа), через которые проходит выбранный маршрут.

На карте транспортные средства отображаются в виде маркеров с цифрой, обозначающей номер маршрута, и определенным цветом для данного типа транспорта, как например, все автобусы на карте отображаются зелеными маркерами.

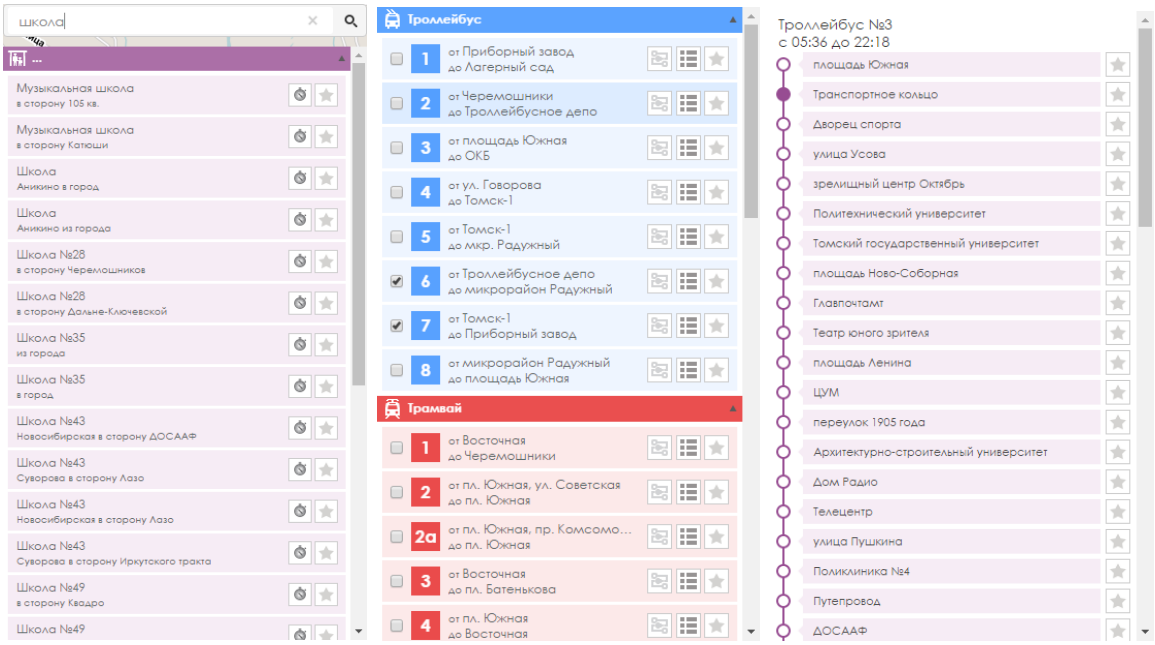

Рис. 1.3. Функциональные панели системы

Маршруты на карте, представлены в виде ломанной линии, также с определенным цветом.

Имеется возможность выбора нескольких маршрутов для отображения на карте, однако отличить один от другого для одного типа транспорта, не представляется возможности.

Использован адаптивный веб-дизайн, обеспечивающий правильное отображение на устройствах различного формата.

Дополнительная информация имеет небольшой объем. Отсутствует определение местоположения пользователя и отображение остановок на карте. Карта представлена в единственном виде.

1.2.2 Центр управления городским автоэлектротранспортом

Муниципальное предприятие «Центр управления городским автоэлектротранспортом» организовано 1 июля 1992 года (Постановление мэрии г. Новосибирска № 459 от 03.06.92г.).

Муниципальное предприятие «Центр управления городским автоэлектротранспортом» с 01.09.98г. реорганизовано в Муниципальное учреждение «Центр управления городским автоэлектротранспортом» приказом Комитета по управлению имуществом мэрии г. Новосибирска от 18.03.98 за № 270 и приказом по МП «ЦУГАЭТ» от 10.08.98г. за № 61-к.

Основные задачи, поставленные перед "Центром управления городским автоэлектротранспортом":

- Контроль и анализ выполнения помаршрутных и сменно-суточных планов работы пассажирского транспорта.
- Оперативное принятие мер по восстановлению движения в случае его сбоев и внесение изменений в организацию работы пассажирского транспорта.
- Оперативное перераспределение подвижного состава по маршрутам в зависимости от фактического выпуска.

Центром управления городским автоэлектротранспортом был создан интернет сервис, отображающий городской транспорт, интерфейс которого можно увидеть на рисунке 1.4.

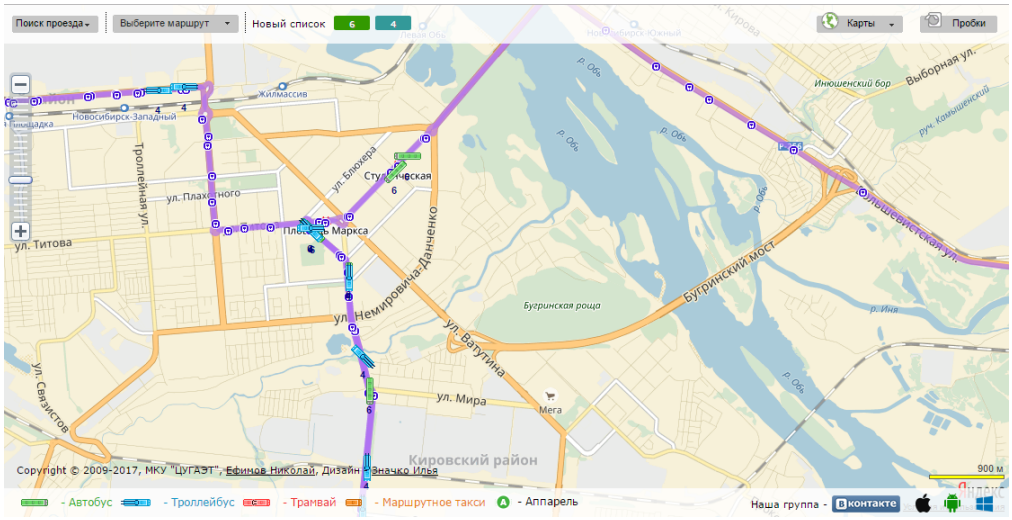

Рис. 1.4. Главная страница интернет сервиса

Данный сервис отображает на карте транспортные средства и маршруты их следования, с возможностью выбора нескольких. Разные типы транспорта имеют уникальные маркеры, но нет возможности визуально отличить по какому маршруту следует автотранспорт, если выбрано отображение нескольких маршрутов для одного типа транспортного средства.

Также при выборе нескольких маршрутов, приходится находить каждый индивидуально во всплывающем списке, который находиться слева на верхней панели. Количество выбранных маршрутов ограничено, система позволяет наблюдать только за пятью маршрутами.

Остановки отображаются только на выбранных маршрутах, что ограничивает информативность и доступность информации на сервисе.

Маркеры имеют мелкий размер, тем самым усложняют отслеживание маршрута или выбор остановки. Отсутствует определение местоположения клиента.

При открытии сервиса на устройствах с небольшим разрешением объекты сайта накладываются друг на друга, что очень усложняет его использование.

На данном сервисе предусмотрен выбор карты, в частности это карты Google и Яндекс, однако нет возможности просмотреть перечень остановок, через которые проходит маршрут.

1.2.3 Интернет сервис умный транспорт

Компания "Умный транспорт" основана в 2009 году, занимается разработкой программного обеспечения в сфере спутникового мониторинга транспорта.

В 2010 году был разработан и запущен интернет-сервис для пассажиров.

На данный момент компания реализовала и внедрила интернет-сервис для пассажиров в более, чем 30 городах России, Казахстана и Белоруссии.

Компания разрабатывает и внедряет проект по предоставлению городскому населению информации о передвижении муниципального и коммерческого транспорта. На рисунке 1.5. можно увидеть пример главной страницы рассматриваемого сервиса для города Рязань.

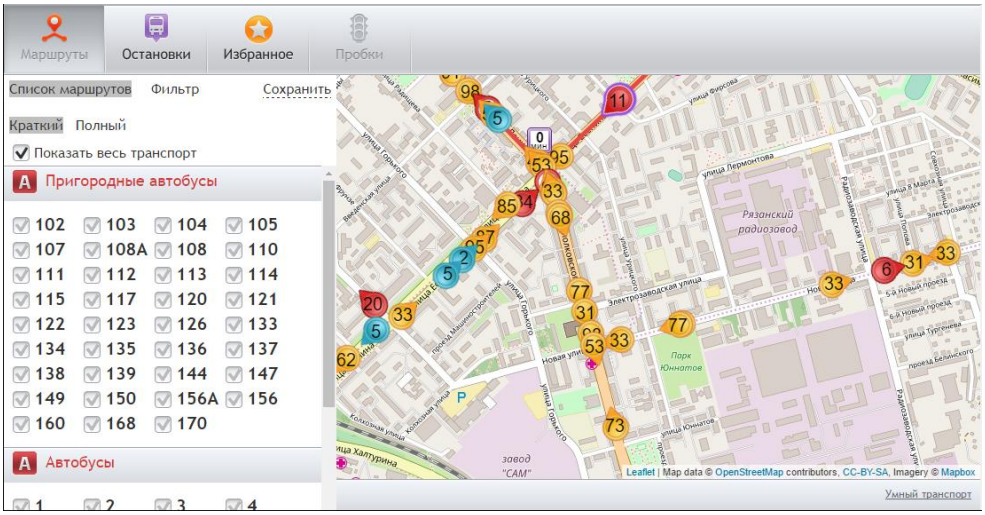

Рис. 1.5. Главная страница сервиса «Умный транспорт»

Как и в рассмотренных ранее системах, в этой также предоставляется возможность выборочного просмотра маршрутов. Визуализация транспортных средств выполнена в виде маркеров цвета, соответствующего типу транспортного средства и номера, обозначающего маршрут движения.

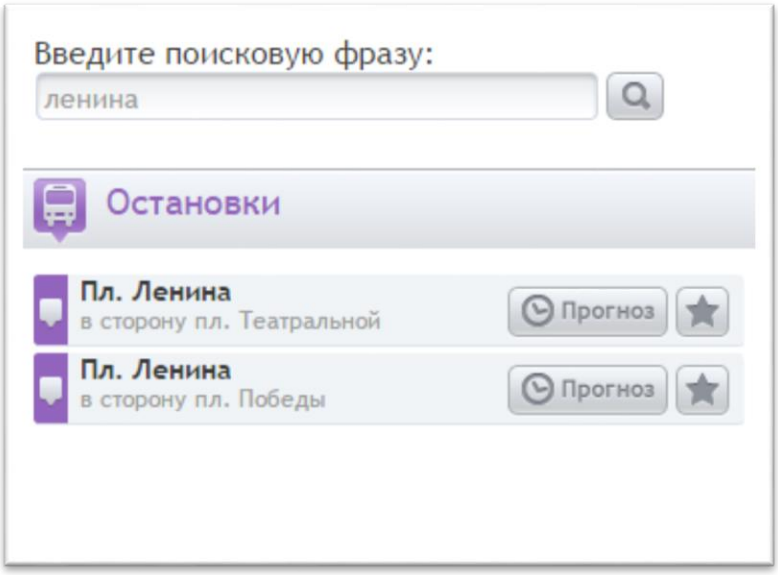

Рис. 1.6. Панель поиска остановки

Присутствует возможность множественного выбора маршрутов. Также реализован просмотр схемы движения маршрута. Имеется поиск остановок по ключевому слову (см. рис. 1.6.), но на карте они не отображаются, за исключением лишь одной, которою выбрали при поиске.

Отсутствует возможность просмотра списка остановок, через которые проходит маршрут. Карта представлена в единственном виде.

В таблице 1.1 приведено сравнение функционала рассмотренных сервисов по ключевым моментам согласно тематики данных сервисов.

Цифрой со звездочкой обозначены пункты, которые являются не ключевыми, но их наличие было бы полезным и удобным при использовании на ресурсах подобной тематики.

Под буквами «А», «Б», «В» подразумеваются описанные выше системы в порядке их рассмотрения, а «Г» разработанная нами система.

Таблица 1.1

Сравнение функционала

№	Пункты сравнения	А	Б	В	Г
1.	Отображение траектории маршрутов	Да	Да	Да	Да
2.	Отображение остановок на карте	Нет	Ограничено	Нет	Да
3.	Поиск по ключевым объектам	Да	Нет	Да	Да
4.	Выбор нескольких маршрутов для отображения	Да	Ограничено	Да	Да
5.	Список остановок на маршруте	Да	Нет	Нет	Да
6.	Дополнительная информация	Да	Нет	Нет	Да
7.	Возможность выбора вида карты	Нет	Да	Нет	Да
8.	Определение местоположения пользователя	Нет	Нет	Нет	Да
9.	Адаптивная верстка сайта	Да	Нет	Нет	Да
10.	Возможность управления картой при помощи клавиатуры	Нет	Нет	Да	Да
11.	Кнопки и шкала масштаба	Нет	Да	Нет	Да

1.3 Требования к информационной системе

Перед проектированием необходимо установить требования к информационной системе в общем. Информационная система должна обеспечить:

- полноту информации для каждого звена системы.
- точность и достоверность информации.
- своевременность и актуальность поступления информации.
- экономичность и эффективность обработки информации.

Кроме того, автоматизированная информационная система должна удовлетворять ряд таких технических требований, как быстродействие, при обработке, поиске и вводе информации, удобный пользовательский интерфейс рабочих мест, возможность развития системы.

Теперь стоит определить требования к результирующей системе в рамках темы выпускной квалификационной работы. Система, в первую очередь должна:

- Обеспечивать администрирование удаленной базы данных и управление данными этой базы с помощью клиент-серверной технологии. А именно, осуществлять хранение, дополнение, изменение и удаление информации. Причем, данный функционал должен быть доступен только для управляющего персонала системы, и закрыт для пользователей.
- Реализовывать отображение информации, хранящейся в базе данных, с возможностью её фильтрации по нескольких признакам, а также предоставлять возможность поиска по ключевым словам.
- Выполнять визуализацию основных объектов база данных на карте и предоставлять возможность выбора нескольких объектов по усмотрению пользователя.
- Реализовать интуитивно-понятный интерфейс для пользователей.

Рассмотрев аналогичные решения и выявив их недостатки необходимо усовершенствовать систему путем:

- Предоставления инструментов управления и отображения графическими объектами на карте;
- Максимизации полноты информации в системе;
- Предоставления пользователям возможности дополнять информативную базу системы;
- Разработки адаптивного интерфейса для корректного отображения на разных типах устройств;
- Добавления функции определения местоположения пользователя;
- Расширения методов управления системой отображения объектов.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

При построении архитектуры веб-приложений, необходимо оптимально уменьшить зависимость между структурными единицами. В общем случае любое веб-приложение состоит из трех структур:

- рабочее место пользователя, которое ограничивается лишь наличием веб-браузера и доступом в интернет;
- модуль, исполняемый на стороне веб-сервера;
- сервер базы данных.

Эти структурные единицы порождают два вида связей.

1. Связь между браузером и сервером.
2. Связь между сервером и базой данных.

Для получения оптимальной независимости между структурными единицами, нужно чтобы каждая такая единица оперировала только нужным ей набором данных.

Чтобы минимизировать зависимость веб-браузером и веб-сервером необходимо задействовать язык разметки HTML только в браузере, и чтобы веб-сервер предоставлял доступ для получения необходимых данных и отображения их на страницы.

2.1 Выбор программного обеспечения для реализации

Главным преимуществом веб-интерфейсов считается отсутствие необходимости установки дополнительных программных продуктов, так как основные операционные системы поставляются с уже установленным браузером.

Классическим и наиболее популярным методом создания веб-интерфейсов является использование HTML – языка для описания структуры

веб-страниц, с применением таблиц стилей CSS и скрипты JavaScript для обеспечения взаимодействия между элементами страницы.

Для реализации своего проекта была выбрана СУБД PostgreSQL. Эта СУБД предназначена для решения широкого круга задач. Она сочетает в себе высокую надежность и простоту установки, а также отвечает всем современным требованием ПО.

PostgreSQL – это мощная, компактная реляционная система управления базами данных. Она может выполняться на разнообразных серверных и клиентских платформах, включая Windows, Linux и на некоторых других платформах UNIX и Mac OC X. К тому же это свободная объектно-реляционная система управления базами данных. PostgreSQL мощный инструмент и используется в высоконагруженных проектах, он более строгий и делает больше проверок за нас, давая меньше вероятности ошибиться, что в перспективе выглядит как преимущество.

Для работы с СУБД PostgreSQL мы воспользуемся оболочкой pgAdmin III – это свободное кроссплатформенное (работающее и в *nix, и в Windows) программное обеспечение, предоставляющее графический интерфейс для разработки и администрирования базы данных.

Клиент-серверные СУБД позволяет обмениваться клиенту и серверу минимально необходимыми объёмами информации. При этом основная вычислительная нагрузка ложится на сервер. Клиент может выполнять функции предварительной обработки перед передачей информации серверу, но в основном его функции заключаются в организации доступа пользователя к серверу.

В качестве технологий для программирования серверной части веб-приложения и доступа к данным мы будем использовать язык программирования общего назначения – PHP, который с легкостью можно внедрять в HTML.

Для реализации картографического функционала используем Leaflet – это библиотека, написанная на JavaScript, с открытым исходным кодом, предназначенная для отображения карт на веб-сайтах

2.2 Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные этапы, на которые разбивается процесс проектирования базы данных информационной системы:

- Концептуальное или инфологическое проектирование.
- Логическое или даталогическое проектирование.

2.2.1 Инфологическое проектирование БД

Инфологическое проектирование – сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия: изучение предметной области и ее информационной структуры, выявление всех фрагментов, каждый из которых характеризуется информационными объектами и связями между ними, процессами над информационными объектами, моделирование и интеграция всех представлений

По окончании данного этапа получаем семантическую модель предметной области, то есть информационную модель наиболее высокого уровня абстракции.

Исходя из предметной области, мы выдели четыре основные сущности, и атрибуты, описывающие их:

- Транспорт (идентификатор транспорта, модель транспорта, государственный номер, широта, долгота, компания);

- Маршрут (идентификатор маршрута, название маршрута, координаты маршрута, список остановок, интервал движения, стоимость проезда);
- Остановка (идентификатор остановки, название остановки, направление, широта, долгота);
- Назначение (идентификатор назначения, транспорт, маршрут, время назначения).

Транспорт назначается на маршрут, которой следует по определенным траекториям, через контрольные точки – остановки. Сущность «Назначение» осуществляет связь между транспортами и маршрутами по их идентификаторам. Каждая из представленных сущностей имеет идентификатор – атрибут, однозначно определяющий один экземпляр каждой сущности и атрибуты, хранящие информацию об экземпляре. Однако в данном представлении разные экземпляры сущностей будут хранить одинаковую информацию, поэтому для избежание избыточности данных мы выделили еще три сущности:

- Тип транспортного средства (идентификатор типа, наименование транспортного средства);
- Модель транспортного средства (идентификатор модели, название модели, количество посадочных мест);
- Компания (идентификатор компании, название компании).

Весь транспорт делиться на несколько типов, как например, автобусы, троллейбусы и так далее. Каждый из типов имеет вариацию моделей, определяемых названием и количеством пассажиров, на которое рассчитана данная модель.

Услуги пассажирских перевозок предоставляют несколько компаний, за которыми закреплены определенные маршруты.

2.2.2 Даталогическое проектирование БД

Даталогическое проектирование является проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных.

На этапе логического проектирования указывают структуру сущности, т.е. набор схем отношений с указанием названия полей, выделением первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. Данная схема изображена на рисунке 2.1.

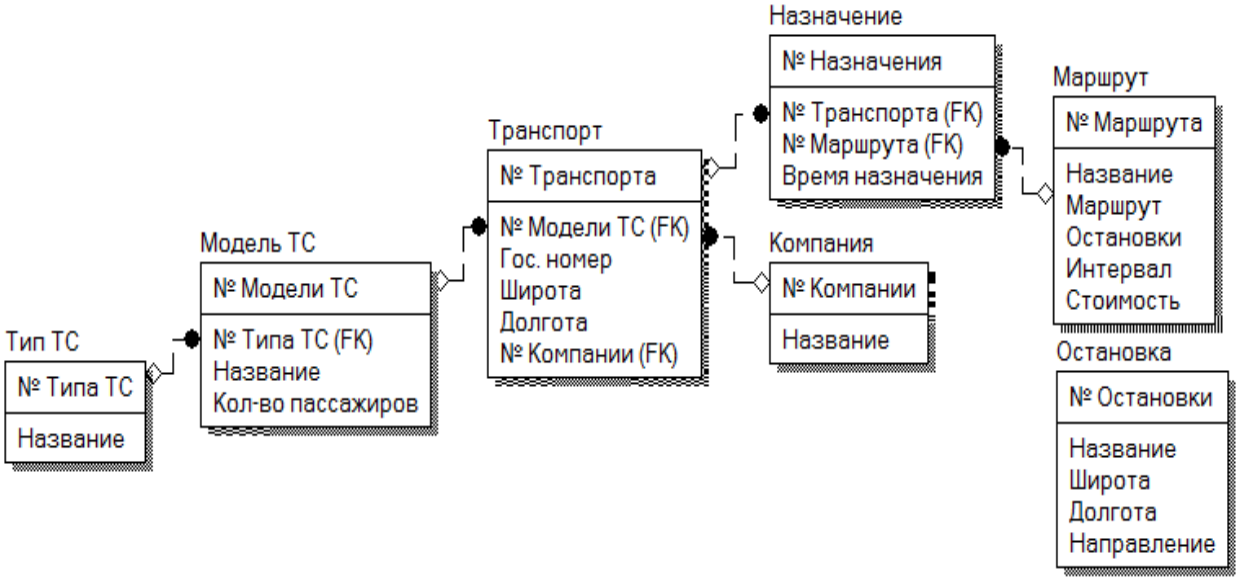

Рис. 2.1. Логическая схема базы данных

Процесс физического проектирования заключается в увязке логической структуры базы данных и физической среды хранения с целью наиболее эффективного размещения данных, т.е. отображении логической структуры базы в структуру хранения.

При таком проектировании указывается название полей, тип данных и размерность.

Физическая схема базы данных представлена на рисунке 2.2.

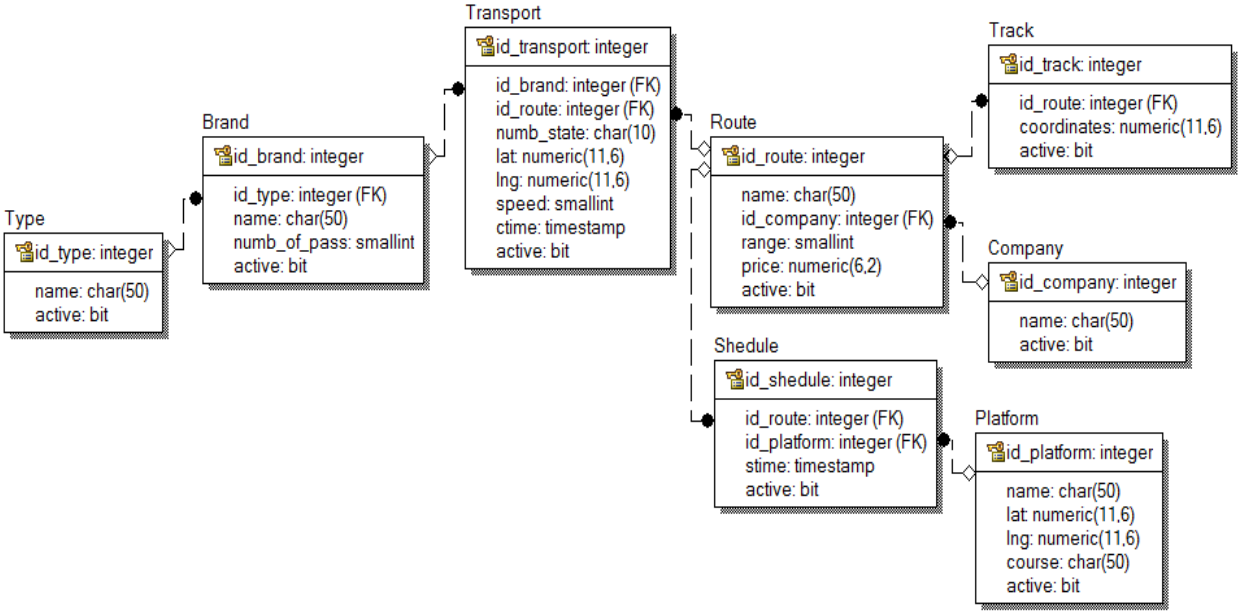

Рис. 2.2. Физическая схема базы данных

2.3 Разработка функциональной и модульной схем

Проектирование программного обеспечения зачастую начинается с разработки функциональной схемы решаемой задачи.

Функциональная схема задачи представляет из себя иерархическое разбиение сложной задачи на ряд задач попроще, которые в свою очередь могут разделиться на подзадачи до тех пор, пока каждая необходимая деталь не будет определена однозначно ясно.

Концепция иерархического разбиения задачи настолько тривиально, что человек даже не осознает, что сталкивается с этим очень часто в своей повседневной жизни.

Можно выделить несколько правил, которых следует придерживаться при построении иерархической функциональной схемы:

1. На каждом уровне задача должна иметь законченный вид.

2. На любом уровне иерархии, разбиение должно полностью охватывать некоторую задачу или функцию, которая соответствует этому уровню.

В рамках данного проекта можно выделить четыре основные задачи требующих решения:

1. Добавление информации в базу данных;
2. Удаление информации из базы данных;
3. Просмотр и поиск информации;
4. Отображение полученной из базы данных информации на карте.

На рисунке 2.3. представлена функциональная схема проектирования.

Рис. 2.3. Функциональная схема

Модульная схема – это схематическое размещение основных модулей разрабатываемого веб-приложения.

Модульная схема веб-приложения представлена на рисунке 2.4.

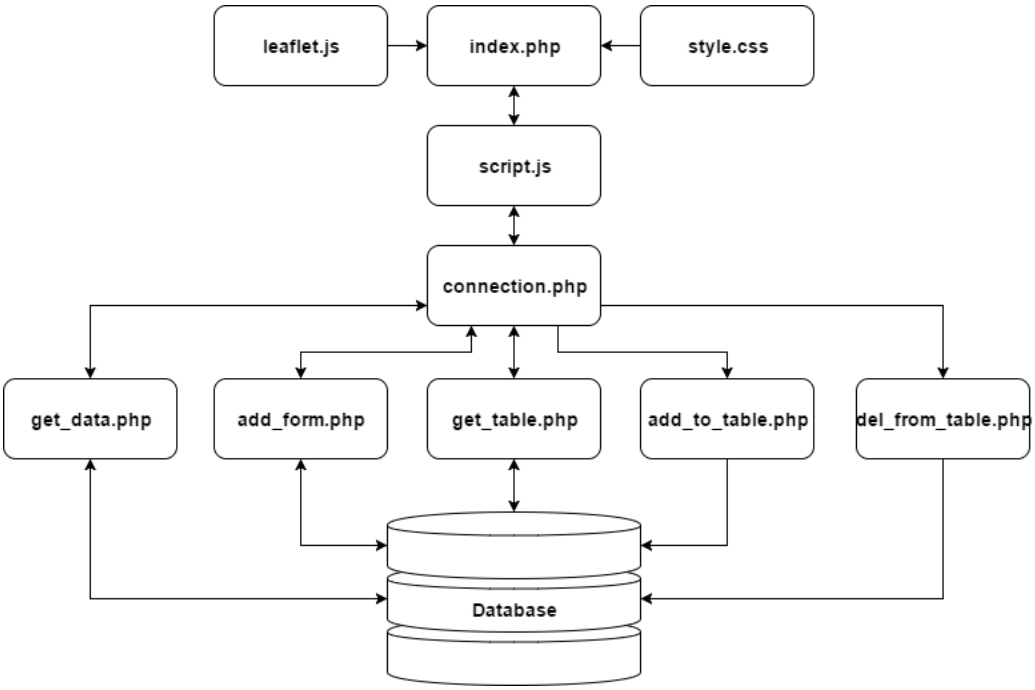

Рис. 2.4 Модульная схема

Разрабатывая модульную схему, мы создаем эскиз будущего проекта. Создание такой схемы приводит к лучшей сбалансированности и пропорциональности всех функциональных элементов проекта.

Таблица 2.1

Описание модульной схемы

Index.php	Веб-страница сайта, написанная с использованием html
Style.css	Таблица стилей для элементов веб-интерфейса
Script.js	Скрипты и ajax-функции написанные на JavaScript для работы с БД и управления элементами интерфейса
Leaflet.js	Подключаемая библиотека для реализации отображения карты, маркеров и линий на веб-странице
Connection.php	Подключение к СУБД
Get_data.php	Получение информации из базы данных
Add_form.php	Формирование форм на веб-странице для добавления, удаления и поиска информации
Get_table.php	Получение информации из базы данных и формирование таблиц с данными для отображения
Add_to_table.php	Добавление информации в базу данных
Del_from_table.php	Удаление информации из базы данных

3 РАЗРАБОТКА БАЗЫ ДАННЫХ И ВЕБ-ПРИЛОЖЕНИЯ

После инфологического и даталогического проектирования базы данных, определения функциональных возможностей проекта, разработки функциональной и модульной схемы, можно преступить к этапу разработки информационной системы.

Разработку информационной системы можно разбить на несколько логических частей:

- Разработка информационного обеспечения – базы данных
- Программирование на стороне сервера – бизнес-логика
- Разработка веб-приложения

3.1 Разработка информационного обеспечения

Для хранения информации была создана база данных в СУБД PosgreSQL «bustrack», содержащая семь. В листинге 3.1. показан код для создания базы данных

Листинг 3.1. Создание базы данных

```
CREATE DATABASE bustrack
  WITH OWNER = postgres
      ENCODING = 'UTF8'
      TABLESPACE = pg_default
      LC_COLLATE = 'Russian_Russia.1251'
      LC_CTYPE = 'Russian_Russia.1251';
```

Конец листинга 3.1.

Для реализации автоинкрементных полей были созданы «последовательности» для каждой таблицы (листинг 3.2.).

Листинг 3.2. Создание последовательности

```
CREATE SEQUENCE public.seq_id_brand START 1
```

Конец листинга 3.2.

Эти последовательности используются в качестве автоинкремента для поле идентификаторов, это позволяет минимизировать ошибки при вводе данных в таблицу.

Для каждой из сущностей, согласно проектированию, были созданы таблицы. Пример создание таблицы «transport», содержащей информацию о конкретном транспорте, представлен в листинге 3.3.

Листинг 3.3. Создание таблицы

```
CREATE TABLE public.transport (id_transport integer NOT NULL DEFAULT
nextval('seq_id_transport'), id_company integer,id_brand integer,numb_state
character varying(10),lat numeric(8,5),lng numeric(8,5),
  CONSTRAINT pk_transport PRIMARY KEY (id_transport),
  CONSTRAINT fk_brand_tr FOREIGN KEY (id_brand)
     REFERENCES public.brand (id_brand)
  CONSTRAINT fk_company_tr FOREIGN KEY (id_company)
     REFERENCES public.company (id_company));
```

Конец листинга 3.3.

Поле «id_transport» указано как первичный ключ для данной таблицы и в значение по умолчанию указана функция, которая при каждом добавлении данных увеличивает значение установленной последовательности на единицу. Также добавлены вторичные ключи для реализации связей между таблицами.

Аналогичным образом были созданы остальные таблицы. Результат продемонстрирован на рисунке 3.1.

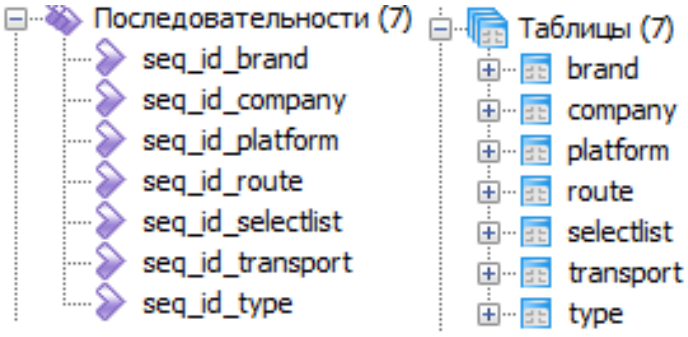

Рис. 3.1. Результат создания последовательностей и таблиц

3.2 Программирование на стороне сервера

Для реализации программного продукта на стороне сервера были созданы представления и хранимые процедуры.

Для каждой из таблиц созданы представления для просмотра информации, содержащейся в таблице. Листинг 3.4. демонстрирует реализацию создания представления для просмотра данных в таблице «transport».

Листинг 3.4. Создание представления

```
CREATE OR REPLACE VIEW public.get_transport
 AS SELECT id_transport, name_company, name_brand, numb_state, lat, lng
   FROM transport
    LEFT JOIN brand ON transport.id_brand = brand.id_brand
    LEFT JOIN company ON transport.id_company = company.id_company;
```

Конец листинга 3.4.

В данном представлении использовано объединение со связными таблицами для вывода понятной читабельной информации вместо идентификаторов. Результат работы рассмотренного представления можно увидеть на рисунке 3.2.

id in	name_company character varying(50)	name_brand character va	numb_state character va	lat numeric(8	lng numeric(8
1	ООО Белкомтранс	ПАЗ 32040	Т320НТ	50.59721	36.57979
2	МУП Городской пассажирский транспорт	ПАЗ 32040	О451ЕТ	50.59996	36.58022
3	МУП Городской пассажирский транспорт	ПАЗ 32054	Р945АХ	50.60348	36.58091
4	АО Пригородная пассажирская компания Черноземье	ПАЗ 32054	У324ХХ	50.60269	36.58769
5	АО Пригородная пассажирская компания Черноземье	АКСМ-321	С578ХА	50.60228	36.59061
6	МУП Городской пассажирский транспорт	ПАЗ 32051	Е578ХА	50.59819	36.59336

Рис. 3.2. Результат работы представления

Для добавления информации в базу данных были разработаны хранимые процедуры для каждой из таблиц, которые используя входные параметры формируют запись и добавляет её в соответствующую таблицу.

Пример создания хранимой процедуры для таблицы «transport» показан в листинге 3.5.

Листинг 3.5. Создание функции добавления данных

```
CREATE OR REPLACE FUNCTION public.add_transport(IN id_company integer,IN id_brand integer,IN numb_state character varying(10),IN lat numeric(8,5),IN lng numeric(8,5))
RETURNS TABLE(id_transport integer, name_company character varying(50),name_brand character varying(50), numb_state character varying(10), lat numeric(8,5),lng numeric(8,5)) AS
$BODY$ INSERT INTO transport (id_company,id_brand,numb_state,lat,lng) VALUES($1,$2,$3,$4,$5); SELECT id_transport, name_company, name_brand, numb_state, lat, lng FROM transport LEFT JOIN brand ON transport.id_brand = brand.id_brand LEFT JOIN company ON transport.id_company = company.id_company; $BODY$ LANGUAGE sql VOLATILE;
```

Конец листинга 3.5.

При добавлении данных поле идентификатор «id_transport» не указывается, так как значение для него формируется автоматически при помощи созданной последовательности.

Для удаления данных из таблицы при создании хранимой процедуры достаточно указать только поле идентификатор записи, которую необходимо удалить. На листинг 3.6. показан пример создания функции для удаления данных из таблицы «transport».

Листинг 3.6. Создание функции удаления данных

```
CREATE OR REPLACE FUNCTION public.del_transport(id integer)
  RETURNS void AS
$BODY$
  DELETE FROM transport WHERE id_transport = $1;
$BODY$
  LANGUAGE sql VOLATILE;
```

Конец листинга 3.6.

Создание простых представление и хранимых процедур на добавление и удаление информации для остальных таблиц выглядит идентично, на рисунке 3.3. представлен список созданных представлений и процедур.

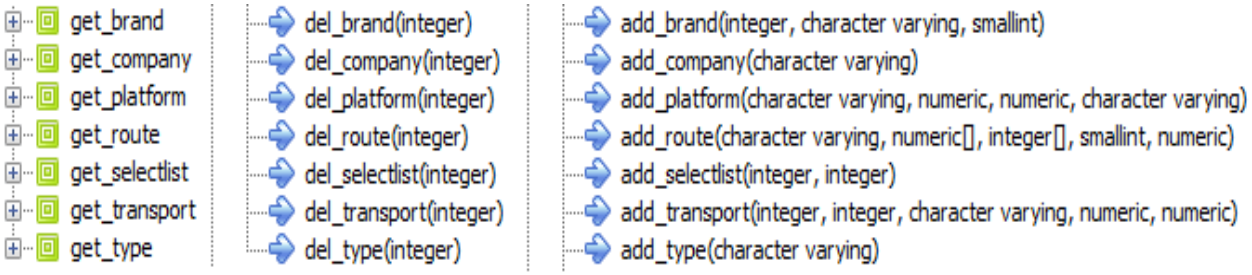

Рис. 3.3. Представления и функции

Также были созданы более сложные представления, как например вывод информации о действующих маршрутах с количеством транспорта, осуществляющих перевозку по данному маршруту. Представление «get_selectlist_full» выводит подробную информацию о назначениях транспортов на маршруты за все время используя объединение с несколькими таблицами. Представление «get_selectlist_full_with_max_time»

представленное на листинге 3.7. выводит последнее назначение транспорта на маршрут с подробной информацией.

Листинг 3.7. Создание представления

```
CREATE OR REPLACE VIEW public.get_selectlist_full_with_max_time
 AS SELECT * FROM get_selectlist_full AS stf
    WHERE select_time = (
    SELECT MAX(select_time) FROM get_selectlist_full AS stf2 WHERE stf.numb_state=stf2.numb_state);
```

Конец листинга 3.7.

Представление из листинга 3.8. вывод информации о действующих маршрутах с количеством транспорта, осуществляющих перевозку по данному маршруту Маршрут является действующим, если на него назначен хотя бы один транспорт и после этого, он не переназначался на любой другой маршрут.

Листинг 3.8. Создание представления

```
CREATE OR REPLACE VIEW public.get_route_with_count_transport
 AS SELECT count(name_brand),name_route,name_type,range,price
    FROM get_selectlist_full_with_max_time
      GROUP BY name_route,name_type,range,price
```

Конец листинга 3.8.

Также были разработаны функции для выборки определенной информации.

Разработанной базой данных установлено хранение для каждого маршрута списка остановок, через которые он проходит, в виде массива идентификатором остановок. Хранение списка остановок в такой форме позволяет сохранить их последовательность и избавиться от избыточности одинаковой информации.

Функция, представленная в листинге 3.9. по входному параметру, означающему номер маршрута, выбирает для него массив идентификаторов остановок, разворачивает его в список и выводит информацию об остановке по идентификатору. Таким образом функция возвращает список остановок с определяющей для нее информацией для конкретного маршрута.

Листинг 3.9. Создание функции

```
CREATE OR REPLACE FUNCTION public.get_platformlist_for_route(
   IN id_route integer) RETURNS
   TABLE(id_platform integer,name_platform character varying(50),lat
numeric(8,5),lng numeric(8,5),course character varying (50)) AS
$BODY$
SELECT id_platform, name_platform, lat, lng, course
     FROM (SELECT UNNEST(ARRAY(SELECT platform FROM route where
id_route = $1)) AS ind) AS plat
        LEFT JOIN platform ON ind = platform.id_platform
$BODY$ LANGUAGE sql VOLATILE;
```

Конец листинга 3.9.

Для поиска маршрута от одной остановки до другой была реализована функция поиска маршрутов, проходящих через две остановки, идентификаторы которых указаны в качестве входных параметров. Процесс создания данной функции показан на листинге 3.10.

Листинг 3.10. Создание функции

```
CREATE OR REPLACE FUNCTION public.search_route_on_platforms(
   IN id_platform1 integer,IN id_platform2 integer)
RETURNS
   TABLE(id_roure integer,name_route character varying(50),coordinates
numeric(8,5)[][],platform integer[], range smallint, price numeric(6,2)) AS
$BODY$
```

```
SELECT * FROM get_route
    WHERE $1= ANY(platform) AND $2= ANY(platform)
$BODY$ LANGUAGE sql VOLATILE;
```

Конец листинга 3.10.

Одной из задач, которых должна решать разрабатываемая система – это вывод транспорта, осуществляющего перевозку людей по определенному маршруту. Для решения данной задачи была разработана функция, которая по введенному идентификатору маршрута, выводит список транспортных средств с информацией, определяющей их. На листинге 3.11. продемонстрирован код разработанной функции.

Листинг 3.11. Создание функции

```
CREATE OR REPLACE FUNCTION public.get_transport_on_route(
   IN id_route integer) RETURNS TABLE(id_selectlist integer,name_company
character varying(50),name_type character varying(50),name_brand character
varying(50),numb_state character varying(10),lat numeric(8,5),lng
numeric(8,5),numb_of_pass smallint) AS
$BODY$ SELECT id_selectlist, name_company, name_type, name_brand,
numb_state, lat, lng, numb_of_pass
   FROM (SELECT * from get_selectlist_full_with_max_time) AS plat
      WHERE plat.name_route = (SELECT name_route FROM get_route WHERE
id_route = $1)
$BODY$ LANGUAGE sql VOLATILE;
```

Конец листинга 3.11.

3.3 Разработка веб-приложения

Для начала необходимо установить на сервер соответствующие драйвера pdo для php и сконфигурировать файл php.ini. Для реализации

подключения веб-приложения к базе данных был создан файл «connection.php», содержание которого можно увидеть на листинге 3.12.

Листинг 3.12. Доступ к БД

```
$host = 'localhost';
$port = 5432;
$dbname = 'bustrack';
$dsn = "pgsql:host=$host; port=$port; dbname=$dbname";
$user = "postgres";
$password = 'root'; // for stac pass OR ''
$dbconn = new PDO($dsn, $user, $password);
```

Конец листинга 3.12

Для отображения всей информации, получаемой из базы данных, был создан файл «index.php», содержание которого можно посмотреть в приложении. Данный файл хранит структуру веб-страницы, определяется элементы интерфейса, содержит блок-контейнер, в котором будет отображаться карта.

Для управления элементами страницы и интерфейса, а также выполнения ajax-функций, которые позволяют обратиться через PHP к базе данных не перезагружая страницу, и реализации функции управления элементами карты, был создан файл «script.js»

Выбранная нами библиотека Leaflet, написанная на JavaScript, позволяет отображать карты на веб-страницах, и организует отображение маркеров, ломанных линий и некоторых простых фигур.

Начало работы с данной библиотекой осуществляется путем создания на веб-странице средствами html «контейнера» для отображения карты и вызова на JavaScript функции инициализации карты.

В листинге 3.13. представлен пример инициализации карты. При этом устанавливает:

- имя блока – контейнера, в котором будет отображаться карта;
- координаты (широты, долготы), по которым произойдет первоначальная центровка карты;
- уровень масштабирования карты при первоначальной генерации.

Листинг 3.13. Инициализация карты

```
function initializeMap(idMap) {
    map = L.map(idMap, {
        center: [50.598077, 36.579920],
        zoom: 13,
        attributionControl: true
    });}
```

Конец листинга 3.13

Для комфортного пользования картой, расширения функциональности и информативности необходимо добавить панели инструментов:

- Сайдбар – панель для отображения блоков с формами и информацей из базы данных;
- Шкала масштаба,
- Функцию поиска местоположения пользователя при помощи геолокации;
- Панель добавления маркеров и ломанных линий на карту;
- Панель редактирования добавленных элементов;
- Панель выбора карты и отображения слоев на ней.

В листинге 3.14. приведен пример добавления панели выбора карт и добавления на панель слоя с картой, предоставляемой веб-картографическим

проектом OpenStreetMap в свободной и бесплатной форме. При этом указывается:

- Веб-адрес где располагаются тайлы (квадратные изображения карты в определенном масштабе);
- Уровень максимального масштаба;
- Ссылка на проект, предоставляющий карты;
- Название слоя для отображения на панели;
- Способ отображения;
- Размеры панели.

Листинг 3.14. Создание слоя для карты

```
var attrOsm = '<a href="http://osm.org/copyright">OpenStreetMap</a> &copy';
var urlOsm = 'http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';
var mapOsm = L.tileLayer(urlOsm, {attribution: attrOsm, maxZoom: 19});
var layerOSM = {"OSM Mapnik": mapOsm};
window.ExampleData = { Basemaps: [{groupName: "OSM",layers : layerOSM}]};
var options={container_width:"300px",position:'topright'};
layerControl=new L.Control.styledLayerControl(ExampleData.Basemaps,options);
map.addControl(layerControl);
layerControl.selectGroup(ExampleData.Basemaps[1].groupName);
```

Конец листинга 3.14.

Для получения информации об объектах, которые необходимо отображать на карте создана ajax-функция «addObjectToMap()», которая возвращает объект с данными. По установленному значению переменной «name» она обращается к файлу «get_data.php», которые подключается к базе данных и выполняет конкретное представление в зависимости от значения «name», после чего возвращает данные полученные при выполнении представления.

Код ajax-функция «addObjectToMap()» представлен в листинге 3.15.

Листинг 3.15. Функция addObjectToMap()

```
function addObjectToMap() {
$.ajax({
    type: 'POST',
    data: ({name:'selectlist_full_with_max_time'}),
    url: 'get_data.php',
    success: function(d){
      var d = JSON.parse(d);
      addLayerControlOnMap(d);}});}});}
```

Конец листинга 3.15.

В функцию «addLayerControlOnMap(d)» в качестве параметра передаются данные о действующих маршрутах. Данная функция осуществляет:

- создание слоев для каждого действующего маршрута;
- создание ломанных линий траекторий маршрутов;
- создание маркеров транспорта на маршруте;
- добавление линий и маркеров на слой;
- добавление созданных слоев на панель выбора.

В листинге 3.16. представлен пример создания маркеров и ломанных линий для последующего добавления на карту. При создании указываются свойства для отображения объекта на карте: координаты, цвет линии, толщина линии, уровень прозрачности и другие. Также создается popup-сообщение с информацией об объекте отображения, которое будет высвечиваться при выборе объекта на карте. После этого осуществляется добавление созданного объекта в определенный слой, соответствующий маршруту.

Листинг 3.16. Создание линий и маркеров

```
var p = new L.polyline(latlng, {color: 'blue',weight: 4,opacity: 0.8,smoothFactor: 1})
    .bindPopup("<strong>Номер маршрута "+strr+"</strong>")
```

```
    .addTo(routeGroup.layers[nameLayer]);
var m = new L.marker([d[j][4],d[j][5]])
      .bindPopup(d[j][6]+" "+d[j][3]+"<br /> Маршрут: "+d[j][8])
      .addTo(transportGroup.layers[nameLayer]);
```

Конец листинга 3.16.

Для отображения остановок на карте была разработана функция «addPlatformOnMap()», которая получает информацию об остановка из базы данных через ajax-функцию, аналогичную описанной ранее. Далее происходит создание слоя остановок и добавление маркеров на слой, созданных в соответствии с полученной информацией.

Для добавления новой информации в базу данных были созданы две функции «addForm(name)» и «addToTable(name)», у которых входной параметр «name» обозначает таблицу, в которую будет добавлена новая информация.

Функция «addForm(name)» обращается к файлу «add_form.php», который подключается к базе данных, получает из нее информацию об имеющейся в ней информации и создает строку html кода, генерируя форму, отображающуюся на веб-странице для ввода и добавления новой информации в конкретную таблицу базы данных, соответствующую входному параметру функции. После этого в функцию возвращается сформированная строка html кода и выполняется отображение полученных данных на веб-страницу в заранее размещенном блоке.

После ввода в форму на веб-странице данных и нажатия на кнопку «Сохранить» запускается функция «addToTable(name)», которая передает введенные данные файлу «add_to_table.php». Данные файл осуществляет подключение к базе данных и для таблицы соответствующей входному параметру функции «name», запускает функцию добавления данных,

разработанную ранее в СУБД, с информацией, введенной в форму на веб-странице.

Для удаления и поиска информации созданы функции «deleteFromTable(name,id)» и «searchToTable(name)» соответственно. Данные функции разработаны аналогично функции добавления данных.

3.4 Пользовательский интерфейс

Для реализованной информационной системы был разработан пользовательский веб-интерфейс, который можно увидеть на рисунке 3.4.

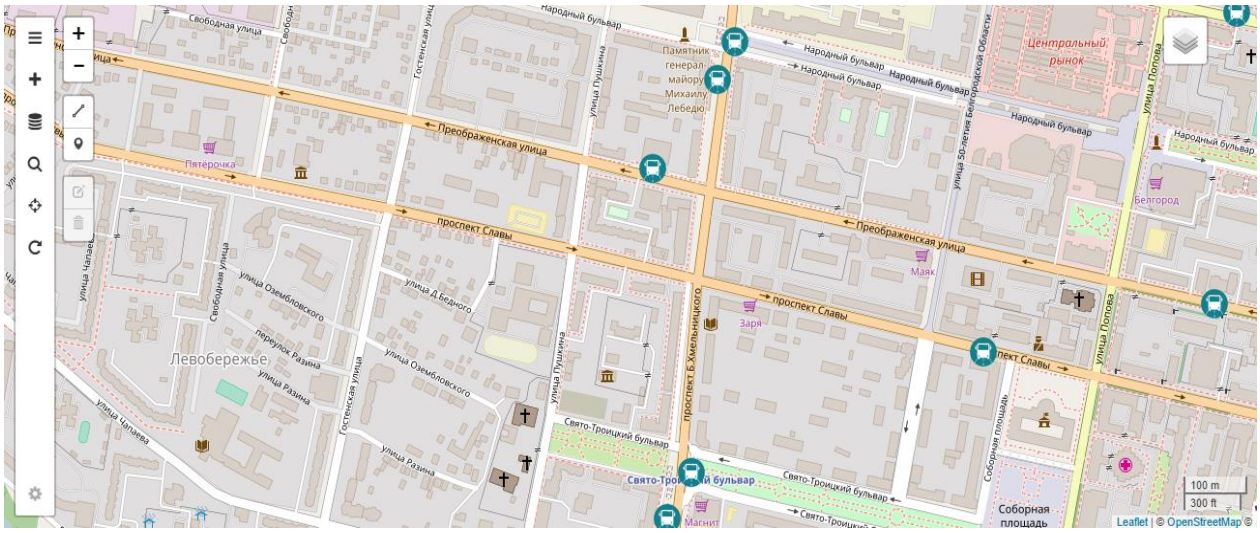

Рис. 3.4. Веб -интерфейс системы

На веб-странице расположены:

- Карта, с отображением необходимых пользователю объектов;
- Информационная панель (слайдер) в свернутом виде, которая организует доступ к таблицам и формам при помощи ссылок в виде иконок, при нажатии на которые панель открывается. Находится в левой части страницы;
- Кнопки масштабирования и рисования маркеров, ломанных линий. Находится с правого края от информационной панели;

- Панель выбора слоев, в свернутом виде, при наведении на которую она открывается, предоставляя пользователю выбрать вид карты и типы объектов для отображения на ней. Находится в верхнем правом углу;
- Шкала масштабирования в нижнем правом углу.

На информационной панели расположены иконки сверху вниз позволяющие:

1. Открыть блок с информацией о действующих маршрутах;
2. Открыть блок с формами для добавления данных;
3. Открыть блок с отображением содержания таблиц базы данных;
4. Открыть блок с формой поиска;
5. Определить место положение пользователя;
6. Вручную обновить содержание карты, если это необходимо.

Например, при нажатии на иконку в виде плюса информационная панель открывается с формами для добавления данных в базу данных (рис. 3.5.).

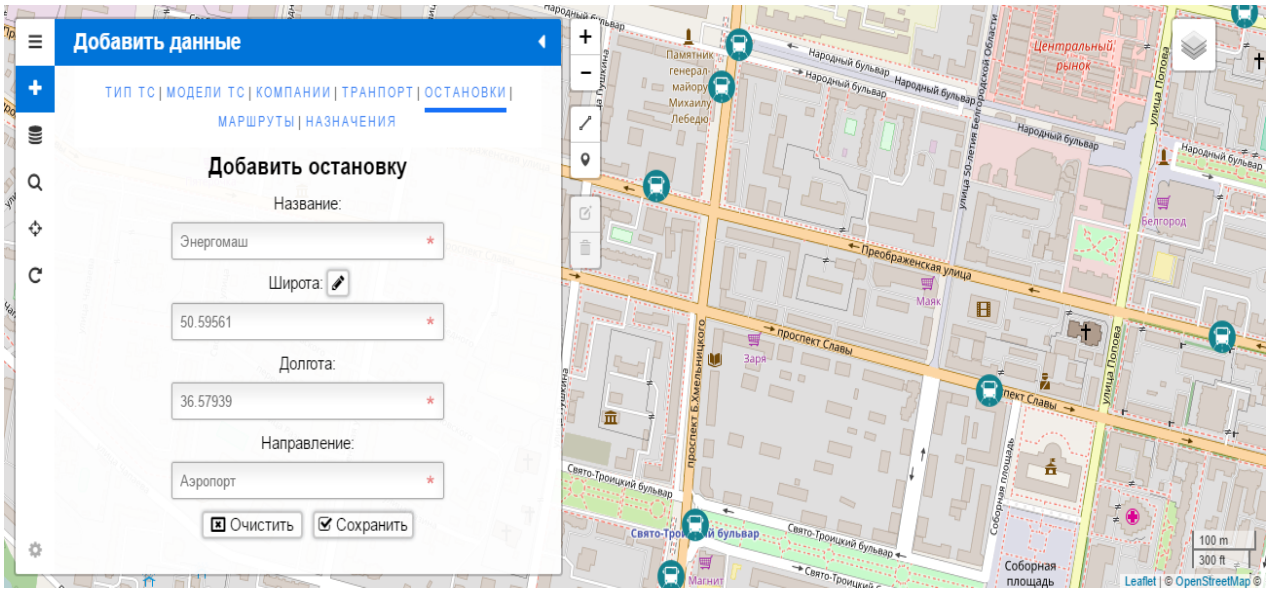

Рис. 3.5 Панель с формой добавления данных

При выборе блока с формами для добавления новой информации в базу данных в верху раскрытой панели имеется список, при помощи которого можно выбрать таблицу, в которую необходимо добавить данные. При

нажатии на какой-либо элемент списка автоматически подгружается соответствующая форма.

Разработанный интерфейс обладает адаптивным свойствам, что позволяет корректно отображать его на устройствах с разным форматом разрешения.

4 ТЕСТИРОВАНИЕ СИСТЕМЫ

В соответствии с установленными требованиями к разрабатываемой системе мы будем осуществлять проверку нашего приложения. Для этого нам необходимо подготовить тестовые данные и провести серию испытаний. Испытания включают в себя:

- Проверку отображения информации из таблиц базы данных;
- Проверку добавления и удаления информации из базы данных;
- Проверку поиска;
- Проверку отображения объектов на карте;

4.1 Тестовые данные

Для проверки отображения данных в веб-интерфейсе разработанной системы мы подготовили данные заполнив таблицу «Транспорт» в базе данных при помощи СУБД. Тестовые данные для проверки отображения данных представлены в таблице 4.1.

Таблица 4.1

Тестовые данные для таблицы «Транспорт»

№	Компания	Модель	Гос.номер
1	ООО Белкомтранс	ПАЗ 32040	Т320НТ
7	ООО Белкомтранс	ПАЗ 32054	Р000ХА
6	МУП Городской пассажирский транспорт	ПАЗ 32051	Е578ХА
2	МУП Городской пассажирский транспорт	ПАЗ 32040	О451ЕТ
3	МУП Городской пассажирский транспорт	ПАЗ 32054	Р945АХ
4	АО Пригородная пассажирская компания Черноземье	ПАЗ 32054	У324ХХ

При проверке поиска запросим информацию о транспорте на определенном маршруте. Для этого мы подготовили базу данных заполнив необходимой информацией и подготовили результат, который система должна выдать (таблица 4.2)

Таблица 4.2

Тестовые данные для поиска транспорта на маршруте «4»

Тип ТС	Модель	Гос. номер
Автобус	ПАЗ 32054	У324ХХ
Троллейбус	АКСМ-321	С578ХА
Автобус	ПАЗ 32051	Е578ХА

Для проверки отображения объектов на карте заполнили базу данных информацией об остановках, транспортах и маршрутах, при этом на карте

4.2 Тестовые испытания

Выполняем тестирование согласно установленным требованиям и подготовленными тестовыми данными

Для начала мы провели проверку отображения информации из таблицы базы данных «Транспорт», перейдя на панели в пункт «Таблицы БД» и выбрав из списка нужную нам таблицу. Результат отображения данных приведен на рисунке 4.1.

Рис. 4.1. Вывод таблицы

Здесь мы проверили процесс удаления данных выбрав транспорт под номером «7» и нажав на иконку корзины (рис. 4.2.).

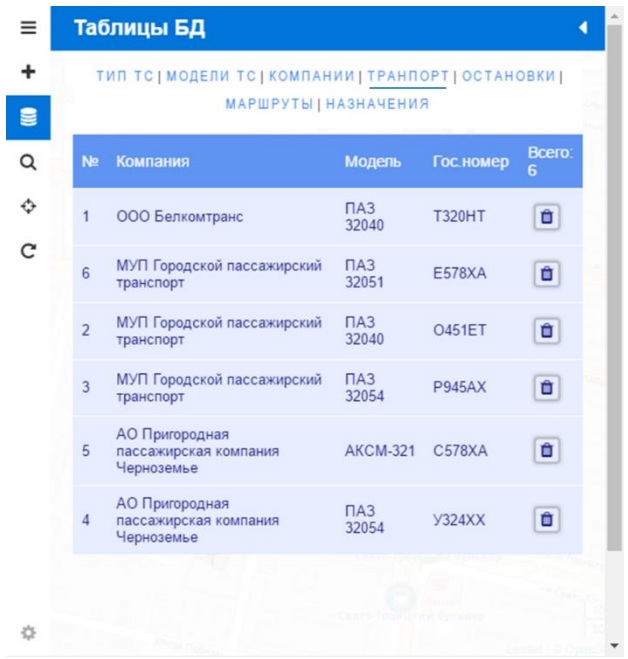

Рис. 4.2 Результат удаления записи.

Для проверки поиска необходимо перейти на панели в пункт «поиск» и выбрать из списка нужную форму согласно подготовленным тестовым данным (ри.4.3.).

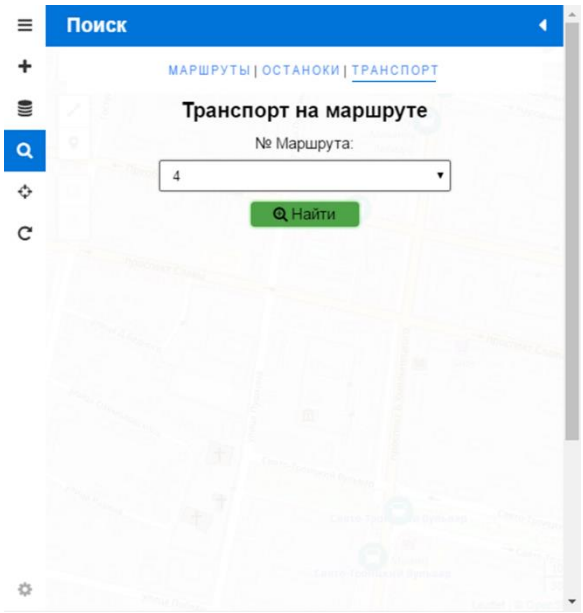

Рис. 4.3. Поиск транспорта по маршруту

Далее необходимо на форме из выпадающего списка выбрать маршрут под номером 4 и нажать кнопку «Найти». На рисунке 4.4. демонстрируется результат работы системы.

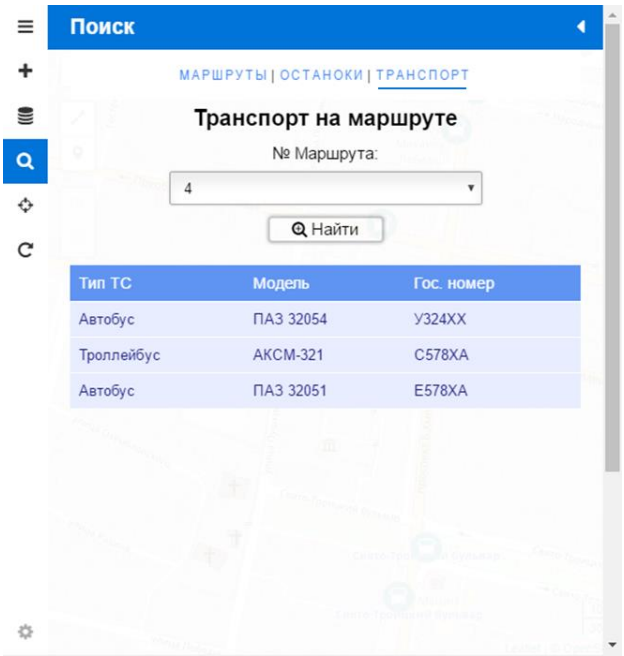

Рис.4.4 Результат поиска транспорта

Перейдя в пункт «Добавить данные» и выбрав таблицу «Компании», в появившемся поле мы добавили строку «Компания “Новая”» и нажали кнопку «Сохранить» (рис. 4.5).

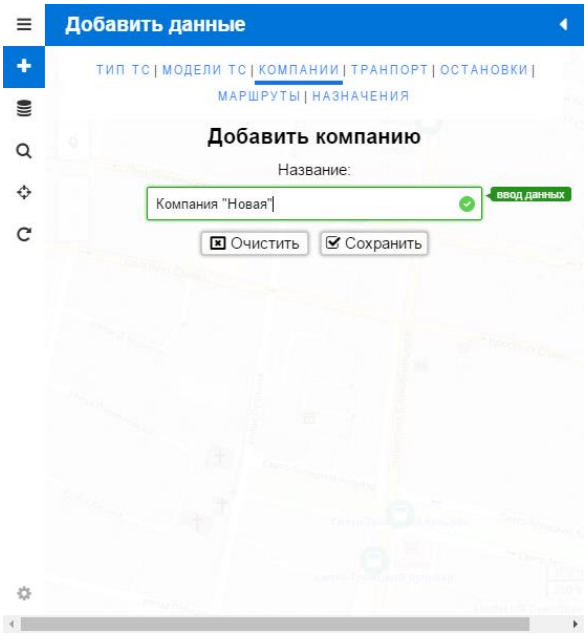

Рис.4.5. Добавление данных в таблицу «Компании»

После этого мы отобразили содержимое таблицы «Компании», которая до добавления данных содержала три записи (рис 4.6.).

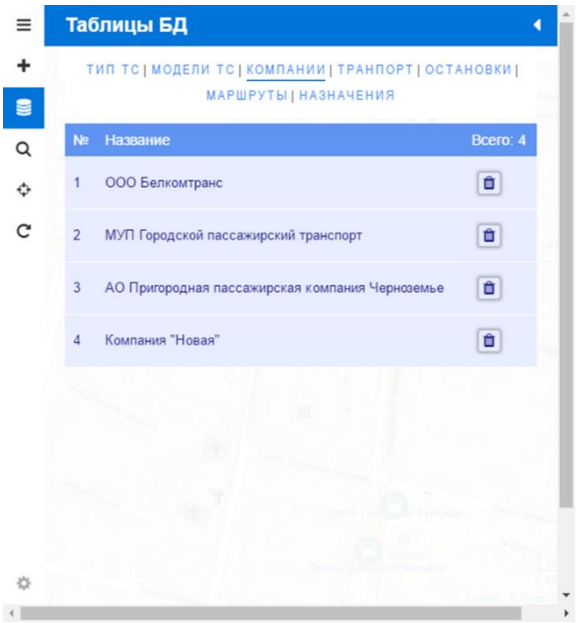

Рис.4.6. Результат добавления данных

Для проверки отображения мы подготовили тестовые данные с информацией о маршрутах и транспортных средствах, которые следуют по маршрутам. Необходимо перейти в пункт «Маршруты» и выбрать необходимый маршрут или выбрать маршрут на панели выбора слоев, расположенной в верхнем правом углу. В результате система отобразит на карте траекторию маршрута в виде ломанной линии и транспорт в виде маркеров, нажав на которые, можно получить краткую информацию об объекте. Пример отображения данных на карте представлен на рисунке 4.7.

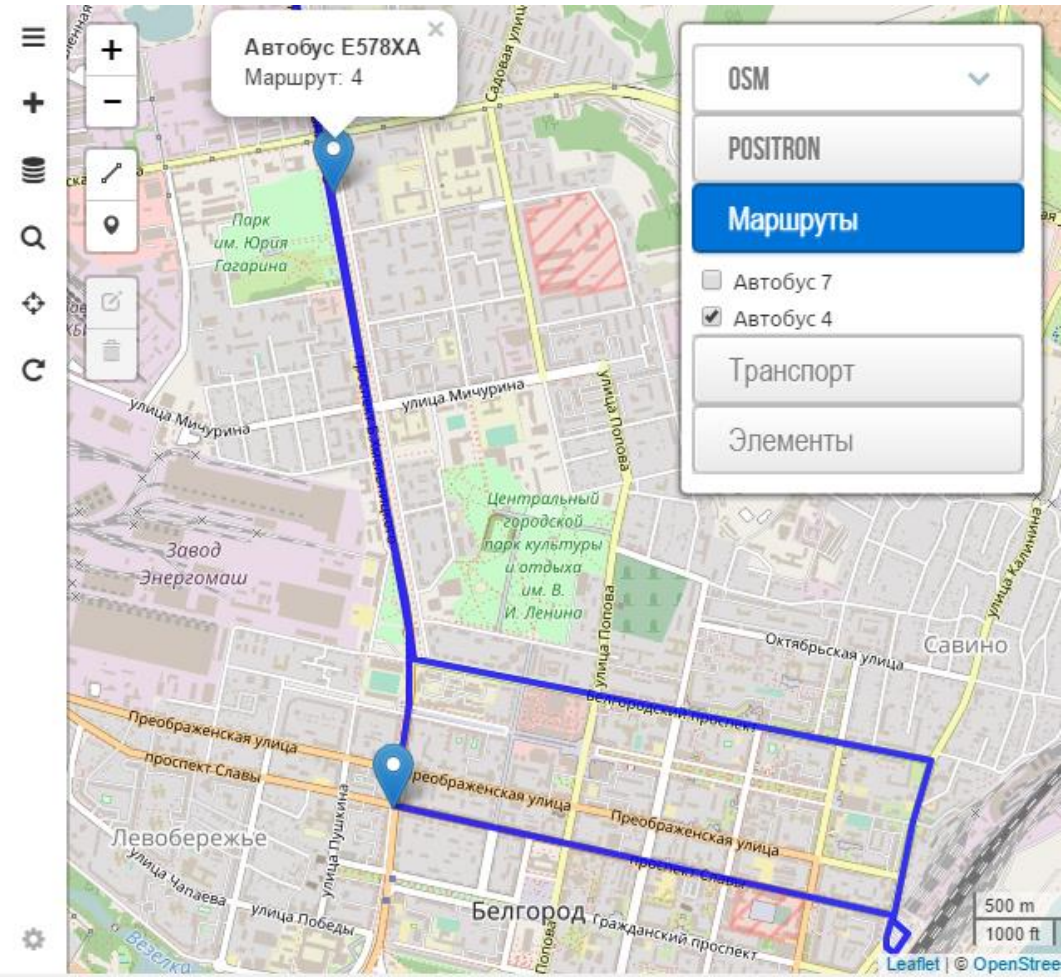

Рис. 4.7. Отображение объектов на карте

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе была поставлена цель разработка автоматизированной информационной системы отображения городского транспорта на карте. Для этого были выявлены объект и предмет исследования, проведен системный анализ предметной области, поставлены задачи, рассмотрены системы реализующие решения подобных задач с сравнением их функционала, выбран формат результирующей системы и установлены требования к ней.

Также была спроектирована база данных и представлены функциональная и модульная схемы информационной системы, разработана база данных в СУБД PstgreSQL.

Используя средства HTML, CSS, JavaScript и PHP данная система была реализована в веди веб-приложения с использованием клиент-серверной технологии и наличием адаптивного интерфейса.

Были подготовлены тестовые данные и проведены испытания для проверки разработанного веб-приложения, которые подтвердили правильную работу системы.

Таким образом, была достигнута поставленная цель выпускной квалификационной работы.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. ГОСТ 33707-2016 (ISO/IEC 2382:2015) Информационные технологии (ИТ). Словарь.

2. ГОСТ 7.0-99. Система стандартов по информации, библиотечному и издательскому делу.

3. ДСТУ 2941-94 Системы обработки информации. Разработка систем. Термины и определения.

4. ГОСТ 34.003-90 Комплекс стандартов на автоматизированные системы.

5. Дж. Уорсли, Дж. Дрейк, «PostgreSQL. Для профессионалов» – Питер, 2003.

6. Марк Пилгрим, «Погружение в HTML5» – БХВ-Петербург, 304 с.

7. Беэр Бибо, Иегуда Кац, «jQuery. Подробное руководство по продвинутому JavaScript» – Символ-плюс, 2011, 624 с.

8. Проектирование информационно-аналитической системы [Электронный ресурс].– Режим доступа: https://www.scienceforum.ru/2015/1351/9581

9. Справочник PHP [Электронный ресурс]. – Режим доступа: http://php.net

10. Документация Leaflet.js [Электронный ресурс]. – Режим доступа: http://leafletjs.com/reference-1.0.3.html

ПРИЛОЖЕНИЯ

Файл script.js

```
var map = null;
var sidebar = null;
var layerControl = null;
var drawControl = null;
var overlay = new Array();
var drawItems = null;
var platformLayer = null;
var platformGroupName = "Элементы";
var routeGroupName = "Маршруты";
var transportGroupName = "Транспорт";
var routeGroup = new Object();
var transportGroup = new Object();
var delTable = "";
var delId = "";

$(document).ready(function (){
        initializeMap('map');
        addSidebarOnMap();
  addScaleControlOnMap();
  addCoordinateControlOnMap();
        addObjectToMap();
        getTable('type');
        addForm('type');
  getRoute('get_route_transport');
  searchForm('searchRouts');
  $('#redraw').on('click', function() {
   repaintMap();
  });
});

function addObjectToMap() {
          $.ajax({
    type: 'POST',
    data: ({name:'type'}),
    url: 'get_data.php',
    success: function(t){
      t = JSON.parse(t);
      $.ajax({
        type: 'POST',
        data: ({name:'selectlist_full_with_max_time'}),
        url: 'get_data.php',
        success: function(d){
          var q = JSON.parse(d);
```

```
                addLayerControlOnMap(t,q);
                addPlatformOnMap();
                                        addDrawControlOnMap();
              }
             });
           }
          });
        }

        function repaintMap() {
          map.removeLayer(platformLayer);
          map.removeLayer(drawItems);
          for(var propName in routeGroup.layers){
             map.removeLayer(routeGroup.layers[propName]);
             map.removeLayer(transportGroup.layers[propName]);
          }
          layerControl.removeGroup(platformGroupName);
          layerControl.removeGroup(routeGroupName);
          layerControl.removeGroup(transportGroupName);
          map.removeControl(drawControl);
                addObjectToMap();
        }

        function initializeMap(idMap) {

                map = L.map(idMap, {
                        center: [50.598077, 36.579920],
                        zoom: 13,
                        attributionControl: true
                });
          var options = {
            container_width   : "200px",
            group_maxHeight     : "80px",
            exclusive         : false,
            collapsed : true,
            position: 'topright'
          };
          layerControl = new L.Control.styledLayerControl(ExampleData.Basemaps, overlay,
options);
          map.addControl(layerControl);
          layerControl.selectGroup("OSM");
        }

        function addSidebarOnMap() {
                sidebar = L.control.sidebar('sidebar').open("home").addTo(map);
        }

        function addScaleControlOnMap() {
          var options = {position: 'bottomright'};
          L.control.scale(options).addTo(map);
```

```
        }

        function addCoordinateControlOnMap() {
         function onMapMouseMove(e) {
          $('#viewLatLng').html(e.latlng.toString());
         }
         map.on('mousemove', onMapMouseMove);
         function onLocationFound(e) {
          var radius = (e.accuracy/40000).toFixed(2);
          L.marker(e.latlng)
           .addTo(map)
            .bindPopup("Вы находитесь здесь.<br>В радиусе " + radius + "
метров.<br>Координаты: "+e.latlng.toString())
            .openPopup();
         }
         map.on('locationfound', onLocationFound);
         function onLocationError(e) {
          alert(e.message);
         }
         map.on('locationerror', onLocationError);
         $('#geoloc').on('click', function() {
          sidebar.close();
          map.locate({setView: true, maxZoom: 16});
         });
        }

        function addLayerControlOnMap(t,d) {
         routeGroup.layers = new Object();
         transportGroup.layers = new Object();
                for (var i = 0 ; i < t.length ; i++) {
          var str = t[i][1];
          for (var j = 0; j < d.length ; j++) {
           if (str == d[j][6]) {
            var strr = d[j][8];
            nameLayer = str+' '+strr;
            routeGroup.layers[nameLayer] = new L.LayerGroup();
            transportGroup.layers[nameLayer] = new L.LayerGroup();
                        var arrF = d[j][9].replace(/{/g,'').replace(/}/g,'').split(',');
                        var latlng = new Array();
                        var count = arrF.length/2;
                        for (var k = 0 ; k < count ; k++) {
                         latlng[k] = new Array(arrF[k*2], arrF[k*2+1]);
                        }
                        var p = new L.polyline(
                         latlng, {
                          color: 'blue',
                          weight: 4,
                          opacity: 0.8,
                          smoothFactor: 1})
                         .bindPopup("<strong>Номер маршрута "+strr+"</strong>")
```

```
                                .addTo(routeGroup.layers[nameLayer]);
                                    var transportIcon = L.Icon.extend({
                                     options: {
                                       iconSize:     [32, 32]
                                     }
                                    });
                    var m = new L.marker([d[j][4],d[j][5]])
                      .bindPopup("<strong>"+d[j][6]+"      "+d[j][3]+"</strong><br />  Маршрут:
"+d[j][8]).openPopup()
                      .addTo(transportGroup.layers[nameLayer]);
                   }
                  }
                 }
                 for(var propName in routeGroup.layers){
                  layerControl.addOverlay( routeGroup.layers[propName], propName , {groupName :
routeGroupName} );
                  layerControl.addOverlay( transportGroup.layers[propName], propName , {groupName
: transportGroupName} );
                 }
                 map.addControl(layerControl);
                }

                function addDrawControlOnMap() {
                 drawItems = new L.featureGroup();
                 layerControl.addOverlay( drawItems, "Draw" , {groupName : platformGroupName} );
                 drawItems.addTo(map);
                       drawControl = new L.Control.Draw({
                  draw: {
                       marker: true,
                   polygon: false,
                   rectangle: false,
                   circle: false
                  },
                  edit: {
                   featureGroup: drawItems,
                   poly: {
                    allowIntersection: false
                   }
                  }
                 });
                 map.addControl(drawControl);
                 function getDataFromDraw() {
                  var arrayLatLng = drawItems.toGeoJSON();
                  if (!arrayLatLng.features.length) {delDataFromDraw()}
                  else{
                       var typeFeatures = arrayLatLng.features[0].geometry.type;
                   arrayLatLng = drawItems.toGeoJSON().features[0].geometry.coordinates;
                   if (typeFeatures!=="Point"){
                          for (var i = 0; i < arrayLatLng.length; i++) {
```

```
                    var buf = arrayLatLng[i][0].toFixed(5); arrayLatLng[i][0] = arrayLatLng[i][1].toFixed(5); arrayLatLng[i][1] = buf;
                    }
                }else{
                    var buf = arrayLatLng[0]; arrayLatLng[0] = arrayLatLng[1]; arrayLatLng[1] = buf;
                    $('#markerLat').val(arrayLatLng[0].toFixed(5));
                    $('#markerLng').val(arrayLatLng[1].toFixed(5));
                }
                $('#polyRoute').val(arrayLatLng.join(","));
            }
        }
        function delDataFromDraw() {
          drawItems.clearLayers();
        }
        map.on(L.Draw.Event.CREATED, function (event) {
            var layer = event.layer;
            drawItems.addLayer(layer);
        });
        map.on(L.Draw.Event.DRAWSTART, function (event) {
            drawItems.clearLayers();
        });
        map.on(L.Draw.Event.DRAWSTOP, function (event) {
          getDataFromDraw();
          sidebar.open('profile');
        });
        map.on(L.Draw.Event.EDITSTOP, function (event) {
          getDataFromDraw();
          sidebar.open('profile');
        });
        map.on(L.Draw.Event.DELETESTOP, function (event) {
          getDataFromDraw();
          sidebar.open('profile');
        });
        var _round = function(num, len) {
          return Math.round(num*(Math.pow(10, len)))/(Math.pow(10, len));
        };
        var getPopupContent = function(layer) {
          if (layer instanceof L.Polyline) {
            var latlngs = layer._defaultShape ? layer._defaultShape() : layer.getLatLngs(),
              distance = 0;
            if (latlngs.length < 2) {
              return "Дистанция: N/A";
            } else {
              for (var i = 0; i < latlngs.length-1; i++) {
                distance += latlngs[i].distanceTo(latlngs[i+1]);
              }
              return "Дистанция: "+_round(distance, 2)+" м";
            }
          }
```

```
            return null;
          };
          map.on(L.Draw.Event.CREATED, function(event) {
            var layer = event.layer;
            var content = getPopupContent(layer);
            if (content !== null) {
              layer.bindPopup(content);
            }
            drawItems.addLayer(layer);
          });
          map.on(L.Draw.Event.EDITED, function(event) {
            var layers = event.layers,
              content = null;
                layers.eachLayer(function(layer) {
              content = getPopupContent(layer);
              if (content !== null) {
                layer.setPopupContent(content);
              }
            });
          });
        }

        function addPlatformOnMap() {
          $.ajax({
            type: 'POST',
            data: ({name:'platform'}),
            url: 'get_data.php',
            success: function(data){
              data = JSON.parse(data);
              paintPlatform(data);
            }
          });
          platformLayer = new L.LayerGroup();
                layerControl.addOverlay( platformLayer, "Остановки", {groupName :
platformGroupName} );
          platformLayer.addTo(map);
          var platformIcon = L.Icon.extend({
            options: {
              iconUrl: 'ico/platform.png',
              iconSize:     [32, 32]
            }
          });
          function paintPlatform(data){
            for (var i = 0 ; i < data.length ; i++) {
              L.marker([data[i][2],data[i][3]], {icon: new platformIcon})
                .bindPopup("<strong>"+data[i][1]+"</strong><br />Направление: "+data[i][4])
                .addTo(platformLayer);
            }
          }
        }
```

```
function addForm(name) {
  $.ajax({
    type: 'POST',
    data: ({name:name}),
    url: 'add_form.php',
    success: function(data){
      $('#addForm').html(data);
                              $('.drawMarker').on('click', function() {
                                      sidebar.close();
                                drawControl._toolbars.draw._modes.marker.handler.enable();
                              });
                              $('.drawRoute').on('click', function() {
                                drawControl._toolbars.draw._modes.polyline.handler.enable();
                                sidebar.close();
                              });
                              $('.delete').on('click', function() {
                                document.getElementById('addToTable').reset();
                                drawItems.clearLayers();
                              });
    }
  });
}
function searchForm(name) {
  $.ajax({
    type: 'POST',
    data: ({name:name}),
    url: 'add_form.php',
    success: function(data){
      $('#searchForm').html(data);
      $('#searchInfo').html('');
    }
  });
}
function getRoute(name) {
  $.ajax({
    type: 'POST',
    data: ({table:name}),
    url: 'get_table.php',
    success: function(data){
      $('#routeInfo').html(data);
      $('.check').on('click', function() {
        if (this.checked) {
          routeGroup.layers[this.value].addTo(map);
        }else{
          map.removeLayer(routeGroup.layers[this.value]);
        }
      });
    }
  });
```

```
}

function getTable(name) {
        $.ajax({
                type: 'POST',
                data: ({table:name}),
   url: 'get_table.php',
   success: function(data){
                        $('#tableInfo').html(data);
    $('.trash').on('click', function(){
        delId = this.dataset.value;
        delTable = this.dataset.table;
        $("#delete-confirm").dialog("open");
        });
   }
        });
}

function deleteFromTable(name,id) {
   $.ajax({
   type: 'POST',
   data: ({"name":name,"id":id}),
   url: 'delete_from_table.php',
   success: function(){
     getTable(name);
     getRoute('get_route_transport');
   }
 });
}

function addToTable(name) {
 var form = $("#addToTable");
 var data = form.serialize();
 $.ajax({
   type: 'POST',
   url: 'add_to_table.php',
   data: data,
   success: function(e){
        document.getElementById('addToTable').reset();
     repaintMap();
     getTable(name);
     getRoute('get_route_transport');
   }
 });
}

function searchToTable(name) {
 var form = $("#searchToTable");
 var data = form.serialize();
 $.ajax({
```

```
    type: 'POST',
    url: 'get_table.php',
    data: data,
    success: function(data){
          $('#searchInfo').html(data);
    }
  });
}
$( function() {
  $( "#delete-confirm" ).dialog({
    modal: true,
    autoOpen: false,
    resizable: true,
    height: 'auto',
    width: 300,

    buttons: {
      Подтвердить: function() {
        $( this ).dialog( "close" );
        if ((delTable !== "") && (delId !== "")) {
          deleteFromTable(delTable,delId);
        }
      },
      Отмена: function() {
        $( this ).dialog( "close" );
      }
    }
  });
} );

$( function() {
  $( "#dialog" ).dialog({
    autoOpen: false,
    show: {
      effect: "blind",
      duration: 1000
    },
    hide: {
      effect: "explode",
      duration: 1000
    }
  });

  $( "#deleted" ).on( "click", function() {
    $( "#delete-confirm" ).dialog( "open" );
  });
});
```