Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Кафедра информационных систем и технологий
Кафедра прикладной информатики

Утверждена приказом проректора по учебной работе от «17» апреля 2020 г. № 481-О
Выполнена по заявке организации (предприятия)
ООО «Центр интеллектуальных технологий для робототехнических и компьютерных систем»

Допущена к защите
«_____» _____ 20___ г.
Зав. кафедрой информационных систем и технологий_____
(название кафедры)
д.физ.-мат. наук, проф. Дроздова В.И.
(уч. степень, уч. звание, Ф.И.О. зав. каф.)

_____
(подпись зав. кафедрой)

Зав. кафедрой прикладной информатикой_____
(название кафедры)
к. э. наук, доцент Азаров И.В.
(уч. степень, уч. звание, Ф.И.О. зав. каф.)

_____
(подпись зав. кафедрой)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ (ДИПЛОМНОМУ ПРОЕКТУ) НА ТЕМУ:

Разработка автоматизированной информационной системы «Управление транспортом»

(общая тема комплексной дипломной работы)

Руководитель (координатор) комплексной выпускной квалификационной работы
Новикова Елена Николаевна, к. физ.-мат. наук, доцент
(Ф.И.О., ученая степень, звание, должность)

Раздел: Разработка серверных приложений приема навигационных данных и мониторинга

Выполнил: Операйло Константин Витальевич
(Ф.И.О.)
Студент 2 **курса, группы** ИНС-м-о-181
направленность (профиль) / специализация

Руководитель: Новикова Елена Николаевна
(Ф.И.О.)
к. физ.-мат. наук, доцент
(Ф.И.О., ученая степень, звание, должность)

_____
(подпись)

09.04.02 «Информационные системы и технологии» профиль «Управление данными»

Очная форма обучения

Рецензент: Зайцева Ирина Васильевна
к. физ.-мат. наук, доцент кафедры
(Ф.И.О., ученая степень, звание, должность)

(подпись)

Раздел: Разработка API обмена данными между клиентскими и серверными приложениями АИС «Управление данными»

Выполнил: Якимов Михаил Алексеевич
(Ф.И.О.)
Студент 2 **курса, группы** ИНС-м-о-181
направленность (профиль) / специализация
09.04.02 «Информационные системы и технологии» профиль «Управление данными»
Очная форма обучения

Руководитель: Новикова Елена Николаевна
(Ф.И.О.)
к. физ.-мат. наук, доцент
(Ф.И.О., ученая степень, звание, должность)

(подпись)

Рецензент: Зайцева Ирина Васильевна
к. физ.-мат. наук, доцент кафедры
(Ф.И.О., ученая степень, звание, должность)

(подпись)

Раздел: Разработка клиентского приложения АИС «Управление транспортом»

Выполнил: Деркач Михаил Игоревич
(Ф.И.О.)
Студент 2 **курса, группы** ИНС-м-о-181
направленность (профиль) / специализация
09.04.02 «Информационные системы и технологии» профиль «Управление данными»
Очная форма обучения

Руководитель: Новикова Елена Николаевна
(Ф.И.О.)
к. физ.-мат. наук, доцент
(Ф.И.О., ученая степень, звание, должность)

(подпись)

Рецензент: Зайцева Ирина Васильевна
к. физ.-мат. наук, доцент кафедры
(Ф.И.О., ученая степень, звание, должность)

(подпись)

Раздел: Разработка модуля составления расписания

Выполнил: Синкевич Николай Владимирович
(Ф.И.О.)
Студент 2 **курса, группы** ПИ-м-о-182
направленность (профиль) / специализация
09.04.03 «Прикладная информатика» профиль «Математическое и информационное обеспечение экономической деятельности»
Очная форма обучения

Руководитель: Азаров Иван Валерьевич
(Ф.И.О.)
к.э.наук, доцент
(Ф.И.О., ученая степень, звание, должность)

(подпись)

Рецензент: Зайцева Ирина Васильевна
к. физ.-мат. наук, доцент кафедры
(Ф.И.О., ученая степень, звание, должность)

(подпись)

Консультанты по разделам:

Разработка серверных приложений
приема навигационных данных
и мониторинга

_____Е.Н. Новикова

наименование раздела подпись, инициалы, фамилия

Разработка API обмена данными
между клиентскими и серверными
приложениями АИС «Управление данными

_____Е.Н. Новикова

наименование раздела подпись, инициалы, фамилия

Разработка клиентского приложения
АИС «Управление транспортом»

_____Е.Н. Новикова

наименование раздела подпись, инициалы, фамилия

Разработка модуля составления расписания

_____И.В. Азаров

наименование раздела подпись, инициалы, фамилия

Нормоконтролер:
_Новикова Е.Н._____
(Ф.И.О. руководителя раздела)
_к. физ.-мат. наук, доцент_____
(ученая степень, звание, должность)

_____
(подпись)

Нормоконтролер:
_Новикова Е.Н._____
(Ф.И.О. руководителя раздела)
_к. физ.-мат. наук, доцент_____
(ученая степень, звание, должность)

_____
(подпись)

Нормоконтролер:
_Новикова Е.Н._____
(Ф.И.О. руководителя раздела)
_к. физ.-мат. наук, доцент_____
(ученая степень, звание, должность)

_____
(подпись)

Нормоконтролер:
Азаров И.В._____
(Ф.И.О. руководителя раздела)
К.э.наук, доцент_____
(ученая степень, звание, должность)

_____
(подпись)

Дата защиты
«» _____ 20___ г.

Оценка_____

Ставрополь, 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение

высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Кафедра информационных систем и технологий
Направление подготовки 09.04.02 «Информационные системы и технологии»
Направленность (профиль) «Управление данными»

«УТВЕРЖДАЮ»
Зав. кафедрой ИСТ

_____В.И. Дроздова
подпись, инициалы, фамилия
«_____»_____2020 г

ЗАДАНИЕ НА КОМПЛЕКСНУЮ ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ (КОМПЛЕКСНУЮ ДИПЛОМНУЮ РАБОТУ)

1.Тема комплексной выпускной квалификационной работы (комплексной дипломной работы) **Разработка автоматизированной информационной системы «Управление транспортом»**

2. Тема раздела комплексной выпускной квалификационной работы
Разработка серверных приложений приема навигационных данных и мониторинга

Утверждена приказом проректора по учебной работе от «17» апреля 2020 г. № 481-О
Студент Операйло Константин Витальевич _____группа ИНС-м-о-18-1
Ф.И.О.
Направление подготовки 09.04.02 Информационные системы и технологии
Направленность (профиль) Управление данными

2.1. Исходные данные для исследования материалы отчета по преддипломной практики, научно-техническая литература, заказ на разработку автоматизированной информационной системы «Управление данными», Единые функционально-технические требования на создание единой региональной системы по управлению автомобильным транспортом и городским наземным электрическим транспортом, осуществляющим регулярную перевозку пассажиров и багажа (АИС «Управления транспортом»)

2.2. Содержание комплексной дипломной работы:
2.2.1 Предпроектное исследование серверной части АИС
2.2.2 Разработка частного технического задания серверного приложения
2.2.3 Разработка архитектуры АИС
2.2.4 Выбор технологий, методов и средств разработки
2.2.5 Проектирование серверных приложений
2.2.6 Разработка серверных приложений
Приложение листинги кодов
Дата выдачи задания 5 марта 2020 г.
Руководитель (координатор) комплексной дипломной работы Е.Н. Новикова
Руководитель раздела комплексной дипломной работы _____Е.Н. Новикова
подпись *инициалы, фамилия*
Консультанты по разделам _____Е.Н. Новикова
подпись *инициалы, фамилия*
_____Е.Н. Новикова

подпись *инициалы, фамилия*

_____Е.Н. Новикова

подпись *инициалы, фамилия*

_____И.В. Азаров

подпись *инициалы, фамилия*

Задание к исполнению принял «05» марта 2020 г._____

подпись

3. Тема раздела комплексной выпускной квалификационной работы

Разработка API обмена данными между клиентскими и серверными приложениями АИС «Управление данными»

Утверждена приказом проректора по учебной работе от «17» апреля 2020 г. № 481-О

Студент Якимов Михаил Алексеевич _____группа ИНС-м-о-18-1

Ф.И.О.

Направление подготовки 09.04.02 Информационные системы и технологии

Направленность (профиль) Управление данными_____

3.1. Исходные данные для исследования материалы отчета по преддипломной практики, научно-техническая литература, заказ на разработку автоматизированной информационной системы «Управление данными», Единые функционально-технические требования на создание единой региональной системы по управлению автомобильным транспортом и городским наземным электрическим транспортом, осуществляющим регулярную перевозку пассажиров и багажа (АИС «Управления транспортом»)

3.2. Содержание комплексной дипломной работы:

3.2.1 Описание данных. Проектирование и разработка базы данных_____

3.2.2 Описание процессов извлечения данных _____

3.2.3 Разработка алгоритмов обработки данных_____

3.2.4 Разработка API обмена данными_____

Дата выдачи задания 5 марта 2020 г. _____

Руководитель (координатор) комплексной дипломной работы_____Е.Н. Новикова

Руководитель раздела комплексной дипломной работы _____Е.Н. Новикова

подпись *инициалы, фамилия*

Консультанты по разделам _____Е.Н. Новикова

подпись *инициалы, фамилия*

_____Е.Н. Новикова

подпись *инициалы, фамилия*

_____Е.Н. Новикова

подпись *инициалы, фамилия*

_____И.В. Азаров

подпись *инициалы, фамилия*

Задание к исполнению принял «05» марта 2020 г._____

подпись

4. Тема раздела комплексной выпускной квалификационной работы

Разработка клиентского приложения АИС «Управление транспортом»

Утверждена приказом проректора по учебной работе от «17» апреля 2020 г. № 481-О

Студент__Деркач Михаил Игоревич_____группа ИНС-м-о-18-1

Ф.И.О.

Направление подготовки 09.04.02 Информационные системы и технологии

Направленность (профиль) Управление данными

4.1. Исходные данные для исследования материалы отчета по преддипломной практики, научно-техническая литература, заказ на разработку автоматизированной информационной системы «Управление данными», Единые функционально-технические требования на создание единой региональной системы по управлению автомобильным транспортом и городским наземным электрическим транспортом, осуществляющим регулярную перевозку пассажиров и багажа (АИС «Управления транспортом»)

4.2. Содержание комплексной дипломной работы:
4.2.1 Предпроектное исследование клиентской части АИС «Управление транспортом»
4.2.2 Составление частного ТЗ разработки клиентской приложения АИС
4.2.3 Проектирование клиентского приложения АИС
4.2.4 Разработка клиентского приложения АИС: разработка модуля «Регулятор», разработка компонента «Карта», разработка компонента «Контрольная точка», разработка компонента «Маршруты»
Приложение листинги кодов
Дата выдачи задания 5 марта 2020 г.
Руководитель (координатор) комплексной дипломной работы Е.Н. Новикова
Руководитель раздела комплексной дипломной работы Е.Н. Новикова
подпись *инициалы, фамилия*
Консультанты по разделам Е.Н. Новикова
подпись *инициалы, фамилия*
Е.Н. Новикова
подпись *инициалы, фамилия*
Е.Н. Новикова
подпись *инициалы, фамилия*
И.В. Азаров
подпись *инициалы, фамилия*

Задание к исполнению принял «05» марта 2020 г.
подпись

5. Тема раздела комплексной выпускной квалификационной работы

Разработка модуля составления расписания

Утверждена приказом проректора по учебной работе от «17» апреля 2020 г. № 481-О
Студент Синкевич Николай Владимирович группа ПИ-м-о-18-2
Ф.И.О.
Направление подготовки 09.04.03 Прикладная информатика
Направленность (профиль) Математическое и информационное обеспечение экономической деятельности

5.1. Исходные данные для исследования материалы отчета по преддипломной практики, научно-техническая литература, заказ на разработку автоматизированной информационной системы «Управление данными», Единые функционально-технические требования на создание единой региональной системы по управлению автомобильным транспортом и городским наземным электрическим транспортом, осуществляющим регулярную перевозку пассажиров и багажа (АИС «Управления транспортом»)

5.2. Содержание комплексной дипломной работы:
5.2.1 Предпроектное исследование модуля составления расписания пассажирских перевозок
5.2.2 Разработка технического задания модуля составления расписания пассажирских перевозок
5.2.3 Проектирование модуля

5.2.4 Разработка алгоритмов расчета расписания

5.2.5 Разработка интерфейса модуля

Приложение листинги кодов

Дата выдачи задания 5 марта 2020 г.

Руководитель (координатор) комплексной дипломной работы Е.Н. Новикова

Руководитель раздела комплексной дипломной работы И.В. Азаров

подпись *инициалы, фамилия*

Консультанты по разделам Е.Н. Новикова

подпись *инициалы, фамилия*

Е.Н. Новикова

подпись *инициалы, фамилия*

Е.Н. Новикова

подпись *инициалы, фамилия*

И.В. Азаров

подпись *инициалы, фамилия*

Задание к исполнению принял «05» марта 2020 г.

подпись

6. Срок представления работы к защите «25» июня 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Направление подготовки 09.04.02 « Информационные системы и технологии»
Направленность (профиль) «Управление данными»

КАЛЕНДАРНЫЙ ПЛАН

Фамилия, имя, отчество (полностью) Операйло Константин Витальевич
Тема комплексной ВКР: Разработка автоматизированной информационной системы «Управление транспортом»

Раздел: Разработка серверных приложений приема навигационных данных и мониторинга
Руководитель (координатор)_ к.физ.-мат.наук, доцент Новикова Е.Н.
Руководитель раздела комплексной дипломной работы к.физ.-мат.наук, доцент Новикова Е.Н.
Консультанты: Нормоконтроль - к.физ.-мат.наук, доцент Новикова Е.Н.

№	Наименование этапов выпускной квалификационной работы	Срок выполнения работы	Примечание
1	Подготовка обзора научно-технической литературы	05.03.2020 - 20.03.2020	
2	Работа над разделами выпускной квалификационной работы	21.03.2014 - 01.06.2020	
3	Консультация по разделам	02.06.2020 - 09.06.2020	
4	Оформление работы	10.06.2020- 17.06.2020	
5	Прохождение нормоконтроля	17.06.2020 - 18.06.2020	
6	Предварительная защита, проверка ВКР системой Антиплагиат	19.06.2020	
8	Получение допуска к защите	19.06.2020-20.06.2020	
9	Отзыв руководителя	22.06.2020-23.06.2020	
10	Рецензирование	23.06.2014-25.06.2020	
11	Срок сдачи работы на кафедру	25.06.2020	
12	Защита выпускной квалификационной работы	1.07.2020-3.07.2020-	

Научный руководитель (координатор) _____Новикова Елена Николаевна

Научный руководитель раздела_____ Новикова Елена Николаевна
подпись, Ф.И.О.
Зав. кафедрой_____ Дроздова Виктория Игоревна
подпись, Ф.И.О.

05 марта 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Направление подготовки 09.04.02 « Информационные системы и технологии»
Направленность (профиль) «Управление данными»

КАЛЕНДАРНЫЙ ПЛАН

Фамилия, имя, отчество (полностью) Якимов Михаил Алексеевич
Тема комплексной ВКР: Разработка автоматизированной информационной системы «Управление транспортом»

Раздел: Разработка API обмена данными между клиентскими и серверными приложениями АИС «Управление данными»

Руководитель (координатор) к.физ.-мат.наук, доцент Новикова Е.Н.
Руководитель раздела комплексной дипломной работы к.физ.-мат.наук, доцент Новикова Е.Н.
Консультанты: Нормоконтроль - к.физ.-мат.наук, доцент Новикова Е.Н.

№	Наименование этапов выпускной квалификационной работы	Срок выполнения работы	Примечание
1	Подготовка обзора научно-технической литературы	05.03.2020 - 20.03.2020	
2	Работа над разделами выпускной квалификационной работы	21.03.2014 - 01.06.2020	
3	Консультация по разделам	02.06.2020 - 09.06.2020	
4	Оформление работы	10.06.2020- 17.06.2020	
5	Прохождение нормоконтроля	17.06.2020 - 18.06.2020	
6	Предварительная защита, проверка ВКР системой Антиплагиат	19.06.2020	
8	Получение допуска к защите	19.06.2020-20.06.2020	
9	Отзыв руководителя	22.06.2020-23.06.2020	
10	Рецензирование	23.06.2014-25.06.2020	
11	Срок сдачи работы на кафедру	25.06.2020	
12	Защита выпускной квалификационной работы	1.07.2020-3.07.2020-	

Научный руководитель (координатор) _____Новикова Елена Николаевна
Научный руководитель раздела_____ Новикова Елена Николаевна
подпись, Ф.И.О.
Зав. кафедрой_____ Дроздова Виктория Игоревна
подпись, Ф.И.О.

05 марта 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Направление подготовки 09.04.02 « Информационные системы и технологии»
Направленность (профиль) «Управление данными»

КАЛЕНДАРНЫЙ ПЛАН

Фамилия, имя, отчество (полностью) Деркач Михаил Игоревич
Тема комплексной ВКР: Разработка автоматизированной информационной системы «Управление транспортом»

Раздел: Разработка клиентского приложения АИС «Управление транспортом»

Руководитель (координатор) к.физ.-мат.наук, доцент Новикова Е.Н.

Руководитель раздела комплексной дипломной работы к.физ.-мат.наук, доцент Новикова Е.Н.
Консультанты: Нормоконтроль - к.физ.-мат.наук, доцент Новикова Е.Н.

№	Наименование этапов выпускной квалификационной работы	Срок выполнения работы	Примечание
1	Подготовка обзора научно-технической литературы	05.03.2020 - 20.03.2020	
2	Работа над разделами выпускной квалификационной работы	21.03.2014 - 01.06.2020	
3	Консультация по разделам	02.06.2020 - 09.06.2020	
4	Оформление работы	10.06.2020- 17.06.2020	
5	Прохождение нормоконтроля	17.06.2020 - 18.06.2020	
6	Предварительная защита, проверка ВКР системой Антиплагиат	19.06.2020	
8	Получение допуска к защите	19.06.2020-20.06.2020	
9	Отзыв руководителя	22.06.2020-23.06.2020	
10	Рецензирование	23.06.2014-25.06.2020	
11	Срок сдачи работы на кафедру	25.06.2020	
12	Защита выпускной квалификационной работы	1.07.2020-3.07.2020-	

Научный руководитель (координатор) _____Новикова Елена Николаевна
Научный руководитель раздела_____ Новикова Елена Николаевна
подпись, Ф.И.О.
Зав. кафедрой_____ Дроздова Виктория Игоревна
подпись, Ф.И.О.
05 марта 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий и телекоммуникаций
Направление подготовки 09.04.03 « Прикладная информатика»
Направленность (профиль) «Математическое и информационное обеспечение экономической деятельности»

КАЛЕНДАРНЫЙ ПЛАН

Фамилия, имя, отчество (полностью) Синкевич Николай Владимирович
Тема комплексной ВКР: Разработка автоматизированной информационной системы «Управление транспортом»
Раздел: Разработка модуля составления расписания
Руководитель (координатор)_ к.физ.-мат.наук, доцент Новикова Е.Н.

Руководитель раздела комплексной дипломной работы к.э.наук, доцент Азаров И.В.
Консультанты: Нормоконтроль - к.э.-н.наук, доцент Азаров И.В.

№	Наименование этапов выпускной квалификационной работы	Срок выполнения работы	Примечание
1	Подготовка обзора научно-технической литературы	05.03.2020 - 20.03.2020	
2	Работа над разделами выпускной квалификационной работы	21.03.2014 - 01.06.2020	
3	Консультация по разделам	02.06.2020 - 09.06.2020	
4	Оформление работы	10.06.2020- 17.06.2020	
5	Прохождение нормоконтроля	17.06.2020 - 18.06.2020	
6	Предварительная защита, проверка ВКР системой Антиплагиат	19.06.2020	
8	Получение допуска к защите	19.06.2020-20.06.2020	
9	Отзыв руководителя	22.06.2020-23.06.2020	
10	Рецензирование	23.06.2014-25.06.2020	
11	Срок сдачи работы на кафедру	25.06.2020	
12	Защита выпускной квалификационной работы	1.07.2020-3.07.2020	

Научный руководитель (координатор) _____Новикова Елена Николаевна
Научный руководитель раздела_____ Азаров Иван Валерьевич
подпись, Ф.И.О.
Зав. кафедрой_____ Азаров Иван Валерьевич
подпись, Ф.И.О.
05 марта 2020 г.

Содержание

Введение

В данной комплексной работе рассматривается проектирование и разработка АИС «Управление транспортом». Каждый раздел посвящен определенному модулю системы и подробно рассматривает каждый этап её разработки, от анализа предметной области до её реализации. Объектом, рассматриваемым в данной работе, является транспортная система. АИС не привязана к определенному городу и должна быть достаточно гибкой в отношении географического местоположения.

Актуальность данной работы достаточно высока. Современный мир продолжает осваивать и использовать информационные технологии все большими темпами. Каждая кампания и предприятие стараются опередить своих конкурентов и использование передовых программ стало уже не просто преимуществом, но и необходимостью.

Реализация приложения подобного масштаба позволит существенно упростить работу как управленческого персонала транспортной системы, так и водителей. Простой и легкий в освоении интерфейс позволит быстро изучить данную АИС и запустить в эксплуатацию.

Задачами данной работы являются:

– сбор, анализ научно-технической информации, отечественного и зарубежного опыта по тематике исследования;

– углубление знаний и навыков по проектированию, внедрению и сопровождению информационных систем и технологий на производстве;

– исследование процессов функционирования АИС «Управление транспортом»;

– выбор методов и средств моделирования процессов функционирования АИС «Управления транспортом»;

Целями выполнения данной работы являются:

– Теоретический анализ предметной области;

– Выбор среды разработки;

– Моделирование и построение алгоритмов;

– Разработка серверных приложений приема навигационных данных и мониторинга;

– Разработка API обмена данными между клиентскими и серверными приложениями АИС «Управление транспортом»;

– Разработка базы данных для АИС «Управления транспортом»;

– Разработка клиентского приложения АИС «Управление транспортом»;

– Разработка модуля составления расписания.

Разрабатываемая система должна соответствовать государственным и международным стандартам, а также требованиям, предъявляемым заказчиком.

В ходе выполнения работы необходимо использовать материал, полученный в ходе обучения и приобрести новые навыки разработки приложений. Необходимо проанализировать существующие решения и методики реализации поставленных задач и выбрать наиболее оптимальное решение.

Данная дипломная работа состоит из четырех частей. Первая часть описывает реализацию серверного приложения, направленного на прием, хранение и сортировку данных о перемещении транспортных средств. Вторая часть описывает API, направленный на прием и обработку запросов со стороны WEB-клиента. В третьей части работы описывается клиентская WEB-часть со всем её обширным функционалом. Четвертая часть описывает разработку интерфейса модуля составления расписания.

1 Разработка серверных приложений приема навигационных данных и мониторинга

1.1 Предпроектное исследование серверной части АИС

Автоматизированная информационная система «Управление транспортом» (АИС «Управление транспортом») представляет собой программно-аппаратный комплекс, обеспечивающий широкий спектр задач мониторинга и управления автомобильным транспортом и городским наземным электрическим транспортом, осуществляющим регулярную перевозку пассажиров и багажа, интегрирует передовые инфокоммуникационные технологии и технологии спутниковых систем мониторинга транспорта.

Основное назначение АИС «Управление транспортом» – повышение качества пассажирских перевозок, контроль и управление транспортными средствами и потоками, соблюдение безопасности пассажироперевозок, повышение уровня информированности населения.

Основными задачами данной системы являются:

– сбор, хранение и передача мониторинговой информации о контролируемом транспорте;

– осуществление мониторинга функционирования общественного транспорта;

– формирование электронной единой маршрутной сети общественного транспорта, с возможностью решения задач оптимизации маршрутной сети;

– осуществление диспетчерского управления общественным транспортом;

– получение информации о нештатных ситуациях;

– подсчета пассажиропотока и прогнозирование пассажиропотока в условиях изменяющейся инфраструктуры города.

Объектом автоматизации являются процессы управления пассажирским транспортом.

Серверная часть разрабатываемой АИС состоит из нескольких компонентов:

– баз данных, для хранения навигационных данных и информации о маршрутах, водителях, остановках и перевозчиках.

– серверного приложения, для сбора навигационных данных с автомобильных терминалов.

На данный момент на рынке не представлено продуктов, выполняющих необходимых серверных функций, однако есть ряд продуктов, предоставляющих услуги по получению и мониторингу навигационных данных. Список некоторых из них:

– ITOB – Позволяет получать навигационные данные со многих бортовых систем и ретранслировать их в базу данных. Затем выводить данные через учетную систему, такую как 1С.

– Runovo – Получение навигационных данных и вывод данных через собственный модуль данного разработчика.

– CityPoint – Закрытая система, позволяющая принимать и выводить навигационные данные.

Все описанные системы имеют одинаковые недостатки:

– Системы являются закрытыми, следовательно, нет возможности внесения исправлений в программный код.

– Все продукты предоставляются по подписке, без возможности приобретения их в постоянное пользование.

– Навязывание дополнительных ненужных функций, увеличивающих стоимость продукта.

Таким образом, создание собственного серверного приложения, является лучшим решением, так как на рынке нет компаний, предоставляющих услуг только по получению навигационных данных, а аренда целой системы обладает целым рядом недостатков.

1.2 Техническое задание на разработку сервера

1.2.1 Полное и краткое наименования информационной подсистемы

Полное наименование системы – Серверное приложение для управления навигационными данными АИС «Управление транспортом».

Краткое наименование системы – «Сервер».

1.2.2 Перечень документов, на основе которых создается Система

– Техническое задание

– Переписка между Сторонами.

1.2.3 Плановые сроки начала и окончания работ по созданию Системы

Определяются в соответствии с графиком и согласованными сроками проведения работ.

1.2.4 Назначение серверного приложения

Основным назначением серверного приложения является сбор и хранения навигационных данных автомобильных терминалов. Приложение осуществляет прием данных в круглосуточном режиме. При повышенной нагрузке строит очереди сообщений для обработки всех запросов. Получая данные от автомобилей, при необходимости, отправляет подтверждения о получении данных. Полученные данные сохраняются в базу данных. Приложения должно контролировать целостность данных в таблицах, с которыми осуществляет работу.

1.2.5 Категории пользователей

Администратор – осуществляет контроль работоспособности приложения. В случае возникновения ошибок устраняет их, или сообщает о них разработчику.

Разработчик – осуществляет исправление ошибок, требующих изменение программного кода, и добавляет новые функции.

1.2.6 Цели приложения:

– получение данных с автомобильных терминалов;

– обработка данных;

– сохранение данных в базу данных.

1.2.7 Состав приложения:

– основная программа, запускающая модули;

– модуль для работы по протоколу EGTS;

– модуль для работы по протоколу NavTelecom;

– модуль для ведения логов;

– модуль конфигурации;

– модуль для работы с брокером сообщений;

– модули для работы с базой данных.

1.2.8 Требования к защите информации от несанкционированного доступа

– поддержка ролевой модели для ограничения доступа и разделения функционала;

– идентификация, аутентификация и авторизация.

1.2.9 Платформа

Язык программирования: C++

СУБД: PostgreSQL

Операционная система: приложение должно работать на всех основных платформах (Linux/Windows/OS X)

1.3 Разработка архитектуры АИС

Основными принципами построения архитектуры системы АИС «Управление транспортом» являются:

– Модульность. Подобный принцип позволяет обеспечить оперативность, масштабируемость и эффективность системы по отношению к используемому аппаратно-программному обеспечению. Кроме того, подобное построение системы позволит сделать более эффективным процесс её разработки, модификации и тестирования, так как позволит разделить её на обособленные логические и программные модули, сузив процесс разработки и тестирования рамками самостоятельных частей. Каждый отдельный модуль может модифицироваться независимо от других модулей, предотвращая возможность нежелательного влияния вносимых изменений на другие части системы. Взаимодействие модулей системы имеет строго типизированный интерфейс,

отражающий логику обмена данными и сигналами системы в наглядной форме. Вместе с тем, подобное модульное разделение системы, позволяет размещать отдельные модули системы на отдельных аппаратных платформах. Взаимная удалённость подобных модулей может быть достаточно велика, поэтому важна роль программной платформы поддерживающей надёжность и защищённость подобных межмодульных коммуникаций.

– Кроссплатформенность позволит работать более чем на одной аппаратной платформе и/или операционной системе. Особенно актуальным этот принцип является для организации автоматизированных рабочих мест пользователей, которые могут работать на различных платформах. Например, такой принцип позволит работать как в операционных системах Linux и Windows одновременно.

АИС «Управление транспортом» условно состоит из нескольких модулей:

– базы данных;

– приложение для управления навигационными данными;

– API для взаимодействия с базой данных;

– клиентские web-приложения (административное приложение, приложение регулятора и так далее).

Клиентские приложения разрабатываются с использованием web-технологий, что позволяет использовать приложения независимо от платформы, так как, приложения будут одинаково работать на любом устройстве, где есть web-браузер. Подробное описание клиентской части приводится в главе 3.

API разрабатывается на языке PHP с использованием фреймворка FatFree, что позволит ему работать на любом web-сервере.

Серверное приложение разрабатывается на языке C++ с использованием фреймворка Qt, что позволяет разработать быстрое и кроссплатформенное приложение.

В качестве СУБД используется PostgreSQL, что позволяет создавать базы данных для любой платформы. Также PostgreSQL позволяет работать с большим

количеством данных, предоставляет высокую скорость работы и имеет хорошую документацию.

Архитектура системы представлена на рисунке 1.1.

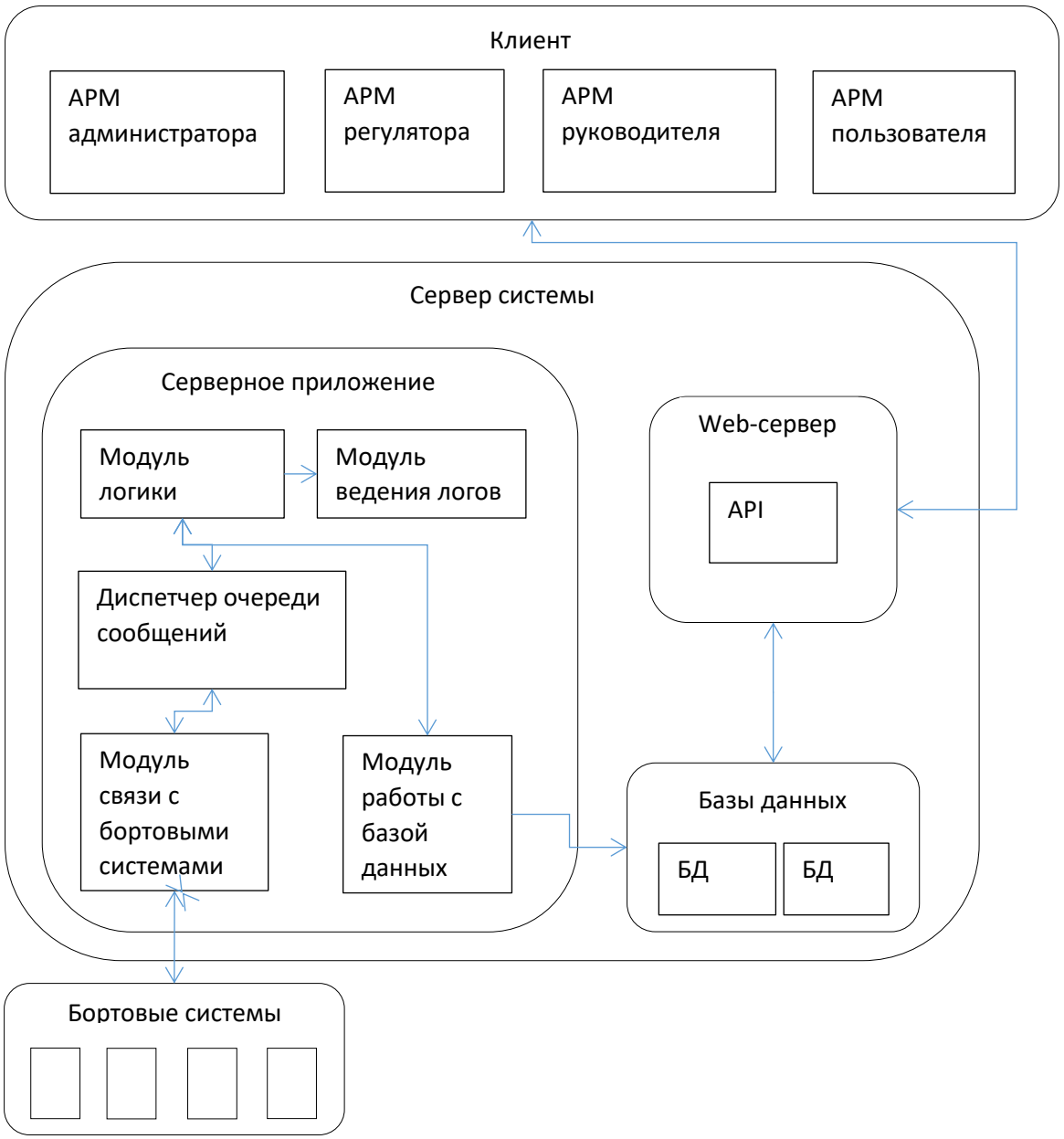

Рисунок 1.1 – Архитектура АИС «Управление транспортом»

Структура серверного приложения приведена на рисунке 1.2. Приложение состоит из нескольких модулей, таких как: модуль логирования, модуль конфигурирования, модуль для работы с очередями, модули для работы по протоколам EGTS и Navtelecom, модули для работы с базой данных.

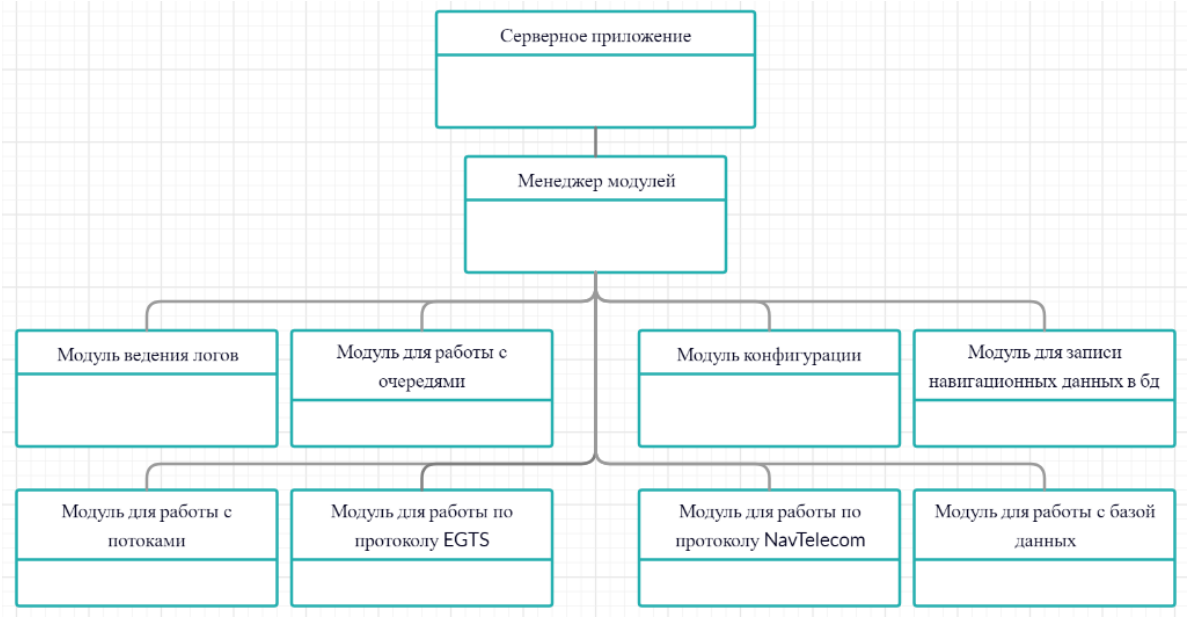

Рисунок 1.2 – Структура серверного приложения

Таким образом, была разработана архитектура и определены основные функции и цели создаваемой системы.

1.4 Выбор технологий, методов и средств разработки

Серверное приложение строится с использованием свободно распространяемых программных продуктов: СУБД – PostgreSQL, библиотеки Qt, реализованной на языке программирования C++, на базе операционной системы семейства Linux. В предлагаемой архитектуре системы для организации диспетчера очередей сообщений используется брокер сообщений RabbitMQ, который является отличным решением в построении сервисно-ориентированной архитектуры и хорошо справляется с распределенными ресурсоемкими задачами. RabbitMQ предоставляет полезный мост, который делает возможным для таких языков, как Java, Ruby, Python, PHP, JavaScript и C# совместно использовать данные в различных операционных системах и средах.

1.4.1 Язык программирования C++

Для разработки был выбран язык C++, поскольку он имеет ряд важных преимуществ:

1. Портативность

C ++ предлагает функцию переносимости или независимости от платформы, которая позволяет пользователю легко запускать одну и ту же программу в разных операционных системах или интерфейсах.

2. Объектно-ориентированный

Одним из главных преимуществ C ++ является особенность объектно-ориентированного программирования, которое включает в себя такие понятия, как классы, наследование, полиморфизм, абстрагирование данных и инкапсуляция, которые позволяют многократно использовать код и делают программу еще более надежной [4].

3. Манипуляция низкого уровня

Поскольку C ++ тесно связан с C, который является процедурным языком, тесно связанным с машинным языком, C ++ допускает низкоуровневое манипулирование данными на определенном уровне. Встроенные системы и компилятор создаются с помощью C ++.

4. Управление памятью

C ++ дает программисту полный контроль над управлением памятью. Это дает новые возможности для программиста, но увеличивает ответственность за управление памятью, а не за управление сборщиком мусора. Эта концепция реализована с помощью DMA (динамическое распределение памяти) с использованием указателей [4].

5. Масштабируемость

Масштабируемость относится к способности программы масштабироваться. Это означает, что программа на C ++ способна работать как в небольших масштабах, так и в больших объемах данных.

1.4.2 Фреймворк QT

В качестве кроссплатформенного инструмента разработки был выбран фреймворк Qt.

Qt – это бесплатный набор инструментов с открытым исходным кодом для создания графических пользовательских интерфейсов, а также кросс-

платформенных приложений, работающих на различных программных и аппаратных платформах, таких как Linux, Windows, MacOS, Android и других [5].

Qt используется для разработки графических пользовательских интерфейсов (GUI) и многоплатформенных приложений, которые работают на всех основных настольных платформах и на большинстве мобильных или встроенных платформ. Большинство программ с графическим интерфейсом, созданных с помощью Qt, имеют нативный интерфейс, и в этом случае Qt классифицируется как инструментарий виджетов. Также могут быть разработаны не-GUI программы, такие как инструменты командной строки и консоли для серверов [6].

Qt поддерживает различные компиляторы, включая компилятор GCC C ++.

Другие функции включают в себя SQL для доступа к базе данных, XML, JSON и поддержку сети [6].

Qt имеет следующий ряд преимуществ перед другими кроссплатформенными инструментами (такими как wxWidgets, KDevelop, Electron, GTK +):

– высокая производительность;

– Qt имеет много классов, не связанных с графическим интерфейсом;

– хорошая поддержка и обилие документации.

Самым большим недостатком является повышенная сложность сборки из-за использования компилятора метаобъектов.

Библиотека использует собственный формат проекта, именуемый .pro файлом, в котором собрана информация о том, какие файлы будут скомпилированы, по каким путям искать заголовочные файлы и много другой информации. Впоследствии при помощи утилиты qmake из них получаются makefile для make-утилиты компилятора [5].

1.4.3 RabbitMQ

В качестве брокера сообщений был выбран RabbitMQ.

RabbitMQ – программный брокер сообщений на основе стандарта AMQP – тиражируемое связующее программное обеспечение, ориентированное на обработку сообщений. Создан на основе системы Open Telecom Platform, написан на языке Erlang, в качестве движка базы данных для хранения сообщений использует Mnesia [7].

Состоит из сервера, библиотек поддержки протоколов HTTP, XMPP и STOMP, клиентских библиотек AMQP для Java и .NET Framework и различных плагинов (таких как плагины для мониторинга и управления через HTTP или веб-интерфейс или плагин «Shovel» для передачи сообщений между брокерами). Имеется реализация клиентов для доступа к RabbitMQ для целого ряда языков программирования, в том числе для Perl, Python, Ruby, PHP. Поддерживается горизонтальное масштабирование для построения кластерных решений [7].

Обладает следующими преимуществами в сравнении с другими брокерами сообщений (ActiveMq, Kafka и других):

- Высокая производительность;
- Удобная панель администрирования;
- Может принимать до 1 миллиона сообщений в секунду;
- Простота конфигурации.

Из минусов стоит отметить необходимость конфигурирования брокера заранее, (конфигурация не может быть осуществлена из кода) и необходимость работающего экземпляра среды выполнения Erlang.

1.4.4 PostgreSQL

В качестве СУБД была выбрана PostgreSQL.

PostgreSQL – это мощная система объектно-реляционных баз данных с открытым исходным кодом, которая использует и расширяет язык SQL в сочетании со многими функциями, которые безопасно хранят и масштабируют самые сложные рабочие нагрузки данных. Происхождение PostgreSQL восходит к 1986 году как часть проекта POSTGRES в Калифорнийском университете в Беркли и уже более 30 лет активно разрабатывает базовую платформу [9].

PostgreSQL заслужил прочную репутацию благодаря своей проверенной архитектуре, надежности, целостности данных, надежному набору функций, расширяемости и преданности сообщества программного обеспечения с открытым исходным кодом, стоящим за программным обеспечением, для последовательного предоставления эффективных и инновационных решений [8]. PostgreSQL работает во всех основных операционных системах, совместим с ACID с 2001 года и имеет мощные дополнения, такие как популярный расширитель геопространственных баз данных PostGIS.

Преимущества PostgreSQL:

– работа с любой ОС;

– поддержка БД неограниченного размера;

– мощные и надёжные механизмы транзакций и репликации;

– расширяемая система встроенных языков программирования и поддержка загрузки С-совместимых модулей;

– наследование;

– легкая расширяемость.

Единственным недостатком при сравнении с другими реляционными СУБД является меньшая скорость записи при выполнении некоторых операций. Однако описанные преимущества делают PostgreSQL отличным выбором для разрабатываемой АИС.

1.4.5 PostGIS

PostGIS – открытое программное обеспечение, которое добавляет поддержку географических объектов в объектно-реляционную базу данных PostgreSQL. PostGIS следует спецификации Simple Features for SQL от Open Geospatial Consortium (OGC) [10].

Особенности [10]:

– типы геометрии для точек, LineStrings, Polygons, MultiPoints, MultiLineStrings, MultiPolygons и GeometryCollections;

– пространственные предикаты для определения взаимодействий геометрий с использованием 3x3 DE-9IM (предоставлено библиотекой программного обеспечения GEOS);

– пространственные операторы для определения геопространственных измерений, таких как площадь, расстояние, длина и периметр;

– пространственные операторы для определения операций геопространственного набора, таких как объединение, разность, симметричная разность и буферы (предоставлено GEOS);

– R-tree-over-GiST (обобщенное дерево поиска) пространственные индексы для высокоскоростных пространственных запросов;

– поддержка селективности индекса для обеспечения высокопроизводительных планов запросов для смешанных пространственных/непространственных запросов;

– для растровых данных используется PostGIS WKT Raster (теперь интегрирован в PostGIS 2.0+ и переименован в PostGIS Raster).

Реализация PostGIS основана на «облегченных» геометриях и индексах, оптимизированных для уменьшения места на диске и памяти. Использование облегченной геометрии помогает серверам увеличить объем данных, перенесенных с физического дискового хранилища в оперативную память, существенно улучшив производительность запросов [10].

1.4.6 Протоколы обмена телеметрическими данными

Серверное приложение должно работать с автомобильными терминалами по протоколам EGTS и NavTelecom.

Протокол передачи данных – набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами. Эти соглашения задают единообразный способ передачи сообщений и обработки ошибок при взаимодействии программного обеспечения разнесённой в пространстве аппаратуры, соединённой тем или иным интерфейсом.

Для организации обмена данными между устройствами и системами, которые занимаются сбором и анализом данных, необходим общий язык или,

иначе говоря, определенный набор правил и команд, который будут понимать обе участвующие в обмене данными стороны. Такой язык называется протоколом передачи данных.

Существуют разные протоколы: некоторые закрытые, т.е. разработанные отдельными компаниями, для обмена данными между выпускаемыми ими самими устройствами, а другие стандартные, пусть даже и не до конца универсальные, но позволяющие обмениваться данными устройствам разных производителей.

В навигационных устройствах для всех вариантов передачи телематической информации по каналам связи используется множество протоколов:

- протокол ЕГТС;
- NTCB;
- NTCT;
- Portman GT2000;
- AutoGraph;
- Galileo;
- PGSM4;
- ITOB MTDS-350;
- Сторож;
- Navitech ASC-1;
- SOAP;
- и др.

Основным же протоколом является протокол EGTS, так как он законодательно закреплен на территории Российской Федерации. Тем не менее, используются и другие протоколы, такие как NTCB. Далее приводится подробное описание этих двух протоколов.

1.4.6.1 Протокол EGTS

Это один из множества протоколов, который применятся передачи телеметрических данных. Особенность его в том, что он законодательно закреплен на территории Российской Федерации.

Для описания протокола используется в основном 2 документа:

– ГОСТ Р 54619 - 2011.

– Приказ №285 Минтранса России от 31 июля 2012 года.

Первый документ, содержит описание межсетевого взаимодействия и структуры пакетов авторизации.

Второй документ описывает структуры пакетов, которые содержат непосредственно данные, такие как широта, долгота, скорость, состояния подключенных датчиков, уровень топлива и т.д.

Указанный протокол является протоколом транспортного уровня. Общая длина пакета протокола транспортного уровня не превышает значения 65535 байт, что соответствует максимальному значению параметра Windows Size (максимальный размер целого пакета, принимаемый на стороне приемника) заголовка протокола TCP [1].

В протоколе предусмотрено 3 типа пакетов [1]:

- EGTS_PT_APPDATA – предназначен для передачи одной или нескольких структур, содержащих информацию.
- EGTS_PT_RESPONSE – с помощью данного типа пакета осуществляется подтверждения пакета протокола транспортного уровня.
- EGTS_PT_SIGNED_APPDATA – отвечает за передачу с применением цифровой площади.

Взаимодействие абонентского терминала (АТ) с сервером происходит следующим образом:

- АТ отправляет пакет авторизации на сервер.
- Сервер отправляет ответ о получении пакета.
- Затем сервер отправляет пакет EGTS_SR_RESULT_CODE с результатом авторизации.

– Если авторизация пройдена, то АТ начинает отправку пакетов с данными.

– На каждый пакет сервер отвечает пакетом типа EGTS_SR_RECORD_RESPONSE.

Схематично процесс изображен на рисунке 1.3.

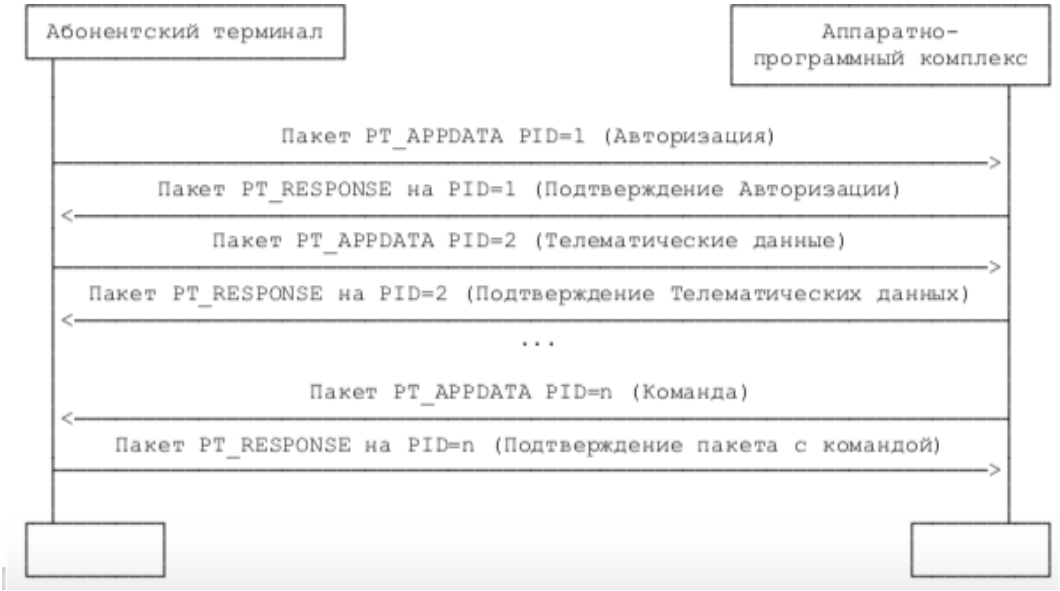

Рисунок 1.3 – Процесс взаимодействия АТ с сервером

Подробное описание процесса авторизации и пакетов которые участвуют в этом процессе, можно найти в ГОСТ Р 54619 – 2011 в пункте 6.7.2.9 Описание процедуры авторизации.

Каждый пакет состоит из 3-х частей [1]:

– заголовок;

– тело пакета (поле SRFD на схеме);

– контрольная сумма (поле SFRCS на схеме).

Схематично структура пакета изображена на рисунке 1.4.

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Тип	Типданны х	Размер, байт
PRV (Protocol Version)								M	BYTE	1
SKID (Security Key ID)								M	BYTE	1
PRF (Prefix)		RTE	ENA		CMP	PR		M	BYTE	1
HL (Header Length)								M	BYTE	1
HE (Header Encoding)								M	BYTE	1
FDL (Frame Data Length)								M	USHORT	2
PID (Packet Identifier)								M	USHORT	2
PT (Packet Type)								M	BYTE	1
PRA (Peer Address)								O	USHORT	2
RCA (Recipient Address)								O	USHORT	2
TTL (Time To Live)								O	BYTE	1
HCS (Header Check Sum)								M	BYTE	1
SFRD (Services Frame Data)								O	BINARY	0 … 65517
SFRCS (Services Frame Data Check Sum)								O	USHORT	0, 2

Рисунок 1.4 – Структура пакета протокола ЕГТС

Основные поля [2]:

– HL – Длина заголовка транспортного уровня с учетом поля HCS.

– FDL – Размер в байтах поля данных SFRD.

– PID – Номер пакета протокола транспортного уровня.

– PT – Тип пакета (типы указаны выше).

– HCS – Контрольная сумма заголовка (по алгоритму CRC-8)

– SFRD – Структура данных, зависящая от типа пакета и содержащая информацию протокола уровня поддержки услуг.

– SFRCS – Контрольная сумма поля SFRD (по алгоритму CRC-16).

Поле SFRD является набором структур вида (рисунок 1.5):

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Тип	Типданны х	Размер, байт
RL (Record Length)								M	USHORT	2
RN (Record Number)								M	USHORT	2
RFL (Record Flags)								M	BYTE	1
SSOD	RSOD	GRP	RPP		TMFE	EVFE	OBFE			
OID (Object Identifier)								O	UINT	4
EVID (Event Identifier)								O	UINT	4
TM (Time)								O	UINT	4
SST (Source Service Type)								M	BYTE	1
RST (Recipient Service Type)								M	BYTE	1
RD (Record Data)								M	BINARY	3...65498

Рисунок 1.5 – Структура поля SFRD

Самыми полезными тут являются OID – идентификатор устройства и RD – набор подзаписей данных, которые имеет следующий формат, указанный на рисунке 1.6.

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0	Тип	Типданных	Размер, байт
SRT (Subrecord Type)								M	BYTE	1
SRL (Subrecord Length)								M	USHORT	2
SRD (Subrecord Data)								O	BINARY	0… 65495

Рисунок 1.6 – Структура подзаписей поля RD

Как видно, структура простая – название типа записи, длина секции данных и сами данные.

Типы подзаписей в зависимости от типа пакета могут быть различные. Типы пакетов указаны в таблице 1.1.

Таблица 1.1 – Типы пакетов

Код	Название	Тип пакета	Документ
0	EGTS_SR_RECORD_RESPONSE	Авторизация	ГОСТ Р 54619
1	EGTS_SR_TERM_IDENTITY	Авторизация	ГОСТ Р 54619
2	EGTS_SR_MODULE_DATA	Авторизация	ГОСТ Р 54619
3	EGTS_SR_VEHICLE_DATA	Авторизация	ГОСТ Р 54619
6	EGTS_SR_AUTH_PARAMS	Авторизация	ГОСТ Р 54619
7	EGTS_SR_AUTH_INFO	Авторизация	ГОСТ Р 54619
8	EGTS_SR_SERVICE_INFO	Авторизация	ГОСТ Р 54619
9	EGTS_SR_RESULT_CODE	Авторизация	ГОСТ Р 54619
0	EGTS_SR_RECORD_RESPONSE	Данные	Приказ №285
16	EGTS_SR_POS_DATA	Данные	Приказ №285
17	EGTS_SR_EXT_POS_DATA	Данные	Приказ №285
18	EGTS_SR_AD_SENSORS_DATA	Данные	Приказ №285
19	EGTS_SR_COUNTERS_DATA	Данные	Приказ №285
20	EGTS_SR_STATE_DATA	Данные	Приказ №285
22	EGTS_SR_LOOPIN_DATA	Данные	Приказ №285
23	EGTS_SR_ABS_DIG_SENS_DATA	Данные	Приказ №285
24	EGTS_SR_ABS_AN_SENS_DATA	Данные	Приказ №285
25	EGTS_SR_ABS_CNTR_DATA	Данные	Приказ №285
26	EGTS_SR_ABS_LOOPIN_DATA	Данные	Приказ №285
27	EGTS_SR_LIQUID_LEVEL_SENSOR	Данные	Приказ №285
28	EGTS_SR_PASSENGERS_COUNTERS	Данные	Приказ №285

В таблице приведены типы пакетов и документы с их описанием.

1.4.6.2 Обзор протоколов NavTelecom

Протоколы NavTelecom часто используются на территории РФ наряду с ЕГТС.

В телематических устройствах (серии «Сигнал» и «Смарт») производства ООО «Навтелеком» для всех вариантов передачи телематической информации по каналам связи используется два протокола [3]:

– бинарный протокол NTCB (Navtelecom Binary) с расширением FLEX;

– текстовый протокол NTCT (Navtelecom Text).

Текстовый (символьный) протокол NTCT используется для передачи телеметрии через службу SMS операторов сотовой связи. Пакеты данного протокола ограничены длиной одного SMS сообщения (140 символов) и включают основную телематическую информацию об объекте контроля.

Для передачи полной информации об устройстве, смены настроек и внутреннего программного обеспечения, используется бинарный протокол NTCB обмена данными по каналам USB, GPRS [3].

Протокол NTCB разделен на прикладной и транспортный уровни.

При обмене по GPRS устройство поддерживает различные варианты представительского и транспортного уровней обмена.

Представительские уровни [3]:

1. ASCII – данные без шифрования с использованием соответствующей таблицы кодировки;

2. AES128 – данные зашифрованные симметричным алгоритмом блочного шифрования.

Транспортные уровни:

1. TCP – протокол обмена, обеспечивает передачу данных в сетях и подсетях TCP/IP;

2. UDP – протокол обмена для передачи данных в сетях IP без установления соединения.

Схема передачи данных на сервер (рисунок 1.7):

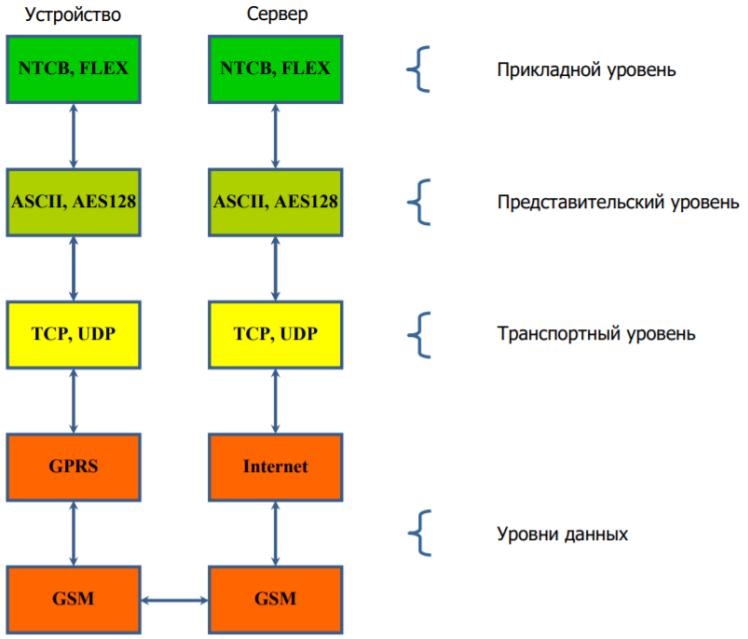

Рисунок 1.7 – Схема общения AT с сервером

NTCB является базовым протоколом со статическими телеметрическими пакетами. Под «статическими» подразумеваются структуры телеметрических записей: F1, F2, F3, F4, F5, F5.1, F5.2, F6. В данный момент часть протокола, касающаяся статических телеметрических сообщений, не используется, используется расширение протокола FLEX. Протокол FLEX представляет собой набор телеметрических сообщений, дающих возможность гибкой настройки передаваемой информации. Также FLEX включает в себя расширение набора команд и сообщений для работы с устройством и периферией. В протоколе NTCB допускаются пакеты длиной не более 65551 байт. Каждое сообщение в рамках данного протокола состоит из двух частей: транспортного заголовка и данных прикладного уровня (телеметрической информации) [3].

Структура сообщений похожа на структуру сообщений протокола ЕГТС (рисунок 1.8):

Поля сообщения NTCB	**<HEAD>(заголовок транспортного уровня)**	**<BODY> (данные прикладного уровня)**
Размер полей в байтах	16	от 0 до 65535

Рисунок 1.8 – Структура пакета NTCB

Структура заголовка так же не сильно отличается от ЕГТС (Рисунок 1.9).

Байты по порядку	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Поле заголовка	Преамбула				Идентификатор получателя IDr				Идентификатор отправителя IDs				Количество байт данных n		Контрольная сумма данных CSd	Контрольная сумма заголовка CSp
Размерность и формат данных поля	char[4]				U32				U32				U16		U8	U8
Значения полей пакета по умолчанию при его отправке от устройства к серверу без прикладных данных (n=0)	@NTC				1				0				0		0x00	0x18
	0x40 0x4e 0x54, 0x43				**0x01** 0x00 0x00 0x00				0x00 0x00 0x00 0x00				0x00 0x00		0x00	0x18
Значения полей пакета по умолчанию при его отправке от сервера к устройству без прикладных данных (n=0)	@NTC				0				1				0		0x00	0x18
	0x40 0x4e 0x54, 0x43				0x00 0x00 0x00 0x00				**0x01** 0x00 0x00 0x00				0x00 0x00		0x00	0x18

Рисунок 1.9 – Структура заголовка

Первые три поля (преамбула, идентификатор получателя, идентификатор отправителя) служат для однозначного определения устройства и сервера при попытке установить соединение. Значения данных полей задаются при настройке устройства. Если эти параметры при настройке устройства не заданы, то устройство будет использовать значения этих полей по умолчанию.

Количество байт данных указывает на количество байт данного пакета, следующих после данного 16-байтового заголовка. Количество байт не может превышать 65535. Контрольные суммы, использующиеся в заголовке, вычисляются по всей длине данных, указанных в предыдущем поле, по алгоритму «исключающего или» (XOR) [3].

При работе по каналу GPRS инициатором установления связи с сервером всегда является устройство. Общий случай процедуры обмена данными с сервером (рисунок 1.10):

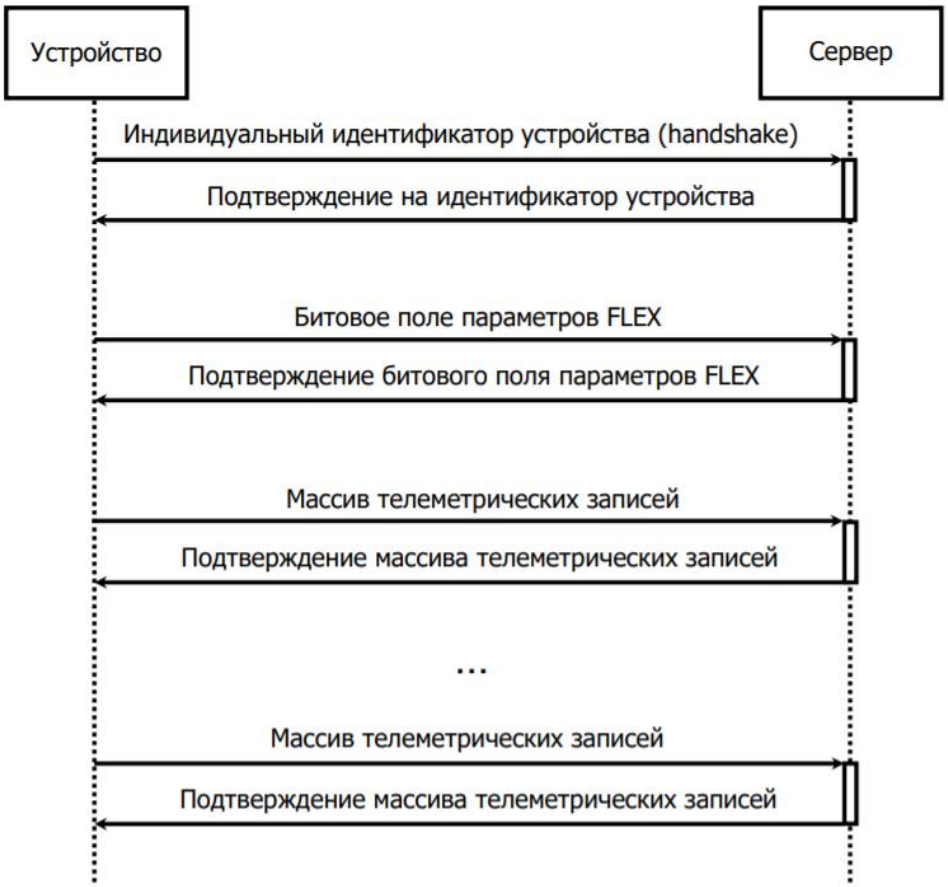

Рисунок 1.10 – Процедура обмена данными с сервером

После открытия соединения (сокета) устройство посылает пакет handshake (рисунок 1.11):

Сообщение	<HEAD>*>S:<s>	
Ответ от сервера	<HEAD>*<S	
Обозначения	**Расшифровка**	**Формат данных**
<HEAD>	16-ти байтовый заголовок пакета NTCB с преамбулой @NTC	U8[16]
*>S	0x2A 0x3E 0x53	char[3]
*<S	0x2A 0x3C 0x53	char[3]
<s>	Строка идентификатора *	char[15]

Рисунок 1.11 – Структура пакета handshake

Далее отсылается сообщение для согласования версий протоколов. В данном сообщении определяется состав и количество передаваемых данных, версия протокола FLEX, версия структуры данных (рисунок 1.12) [3].

Сообщение	<HEAD>*>FLEX<protocol><protocol_version><struct_version><data_size><bitfield[data_size/8]>	
Ответ от сервера	<HEAD>*<FLEX<protocol><protocol_version><struct_version>	
Обозначения	**Расшифровка**	**Формат данных**
<HEAD>	16-ти байтовый заголовок пакета NTCB с преамбулой @NTC	U8[16]
*>FLEX	0x2A 0x3E 0x46 0x4C 0x45 0x58	char[6]
*<FLEX	0x2A 0x3C 0x46 0x4C 0x45 0x58	char[6]
<protocol>	Условное обозначение протокола, в котором собирается работать устройство: 0xB0 – FLEX	U8
<protocol_version>	Версия протокола нужна для идентификации совместимости набора команд и формата пакетов на сервере и в устройстве. Для версии 1.0 - 10 (0x0A) Для версии 2.0 - 20 (0x14)	U8
<struct_version>	Версия структуры данных нужна для идентификации совместимости формата передаваемых данных на сервере и в устройстве. Для версии 1.0 - 10 (0x0A) Для версии 2.0 - 20 (0x14)	U8
<data_size>	Размер последующего конфигурационного поля <bitfield[data_size/8+(1)]> в битах Для структуры версии 1.0 - 69 Для структуры версии 2.0 - 122	U8
<bitfield[data_size/8+(1)]>	Битовый массив с информацией о передаваемых полях структуры данных. Передаваемому полю соответствует установленный бит, при нулевом бите соответствующее поле не передается. Значение массива определяется конфигурацией устройства. Длина в байтах вычисляется как целое число байт способное уместить указанное в поле <data_size> число. Для структуры версии 1.0 - 9 байт Для структуры версии 2.0 - 16 байт	[U8] (массив байт)

Рисунок 1.12 – Структура пакета согласования протоколов

После подтверждения, устройство присылает телеметрические данные (рисунок 1.13):

Сообщение	~A<size><x[0]-x[size-1]><crc8>	
Ответ от сервера	~A<size><crc8>	
Обозначения	**Расшифровка**	**Формат данных**
~A	0x7E 0x41	char[2]
<size>	Количество телеметрических записей, передаваемых в массиве	U8
<x[0]-x[size-1]>	Массив телеметрических записей, со структурой FLEX. Количество передаваемых параметров и размер в байтах соответствует значению в поле <bitfield[data_size/8+(1)]> пакета FLEX. Записи следуют друг за другом без каких-либо разграничителей.	-
<crc8>	8ми-разрядная побайтовая CRC8 символов ~A и полей <size> и <x[0]-x[size-1]>	U8

Рисунок 1.13 – Сообщение с телеметрическими данными

В теле сообщения такого сообщения могут находиться различные данные, самые главные из них: широта и долгота автомобиля, скорость, направление.

1.5 Проектирование серверных приложений

Проектирование серверного приложения следует начать с разработки UML-диаграмм.

1.5.1 Разработка функциональной модели

Модель работы представляет собой систему наборов работ, где каждая из работ – это совокупность нескольких объектов или наборов объектов [11]. На данной диаграмме (Рисунок 1.14):

1. На вход в качестве материала поступают данные с передатчиков;
2. Правила, которыми руководствуется работа, в данной диаграмме выступают ГОСТы и техническое задание;
3. Конечная цель – сохранение и хранение данных;
4. Ресурсы, которые выполняют работу:

– Администратор – сотрудник, который контролирует работу программы;

– Разработчик.

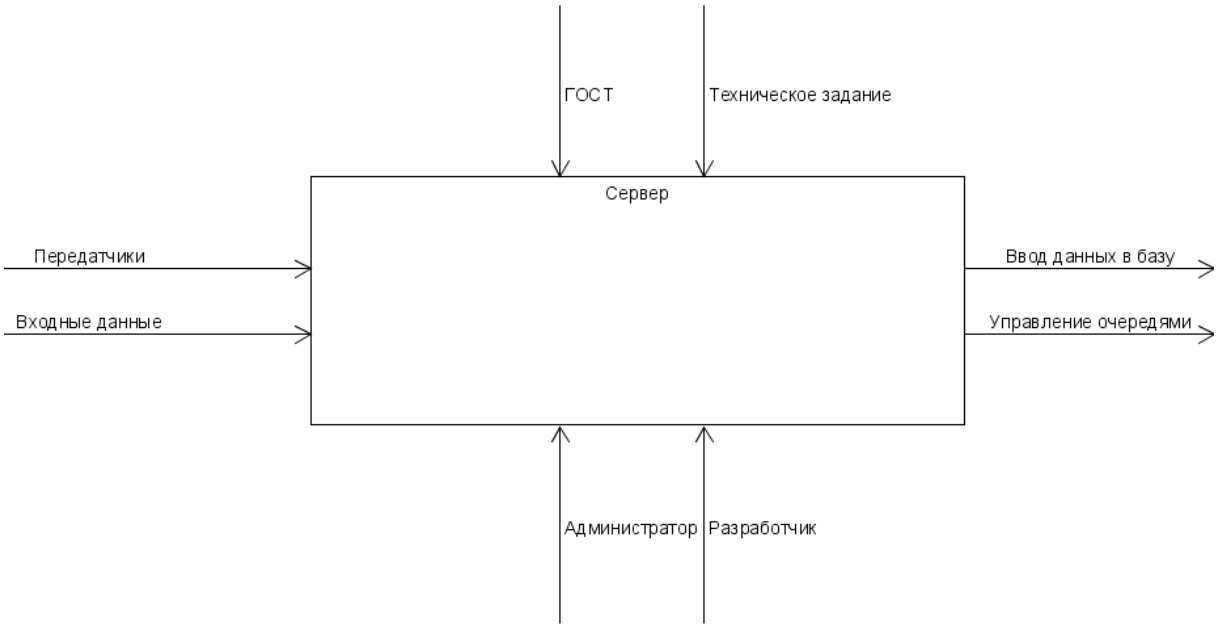

Рисунок 1.14 – Общая диаграмма

1.5.2 Разработка диаграммы прецедентов

Вариант использования представляет собой типичное взаимодействие пользователя и проектируемой системы (Рисунок 1.15).

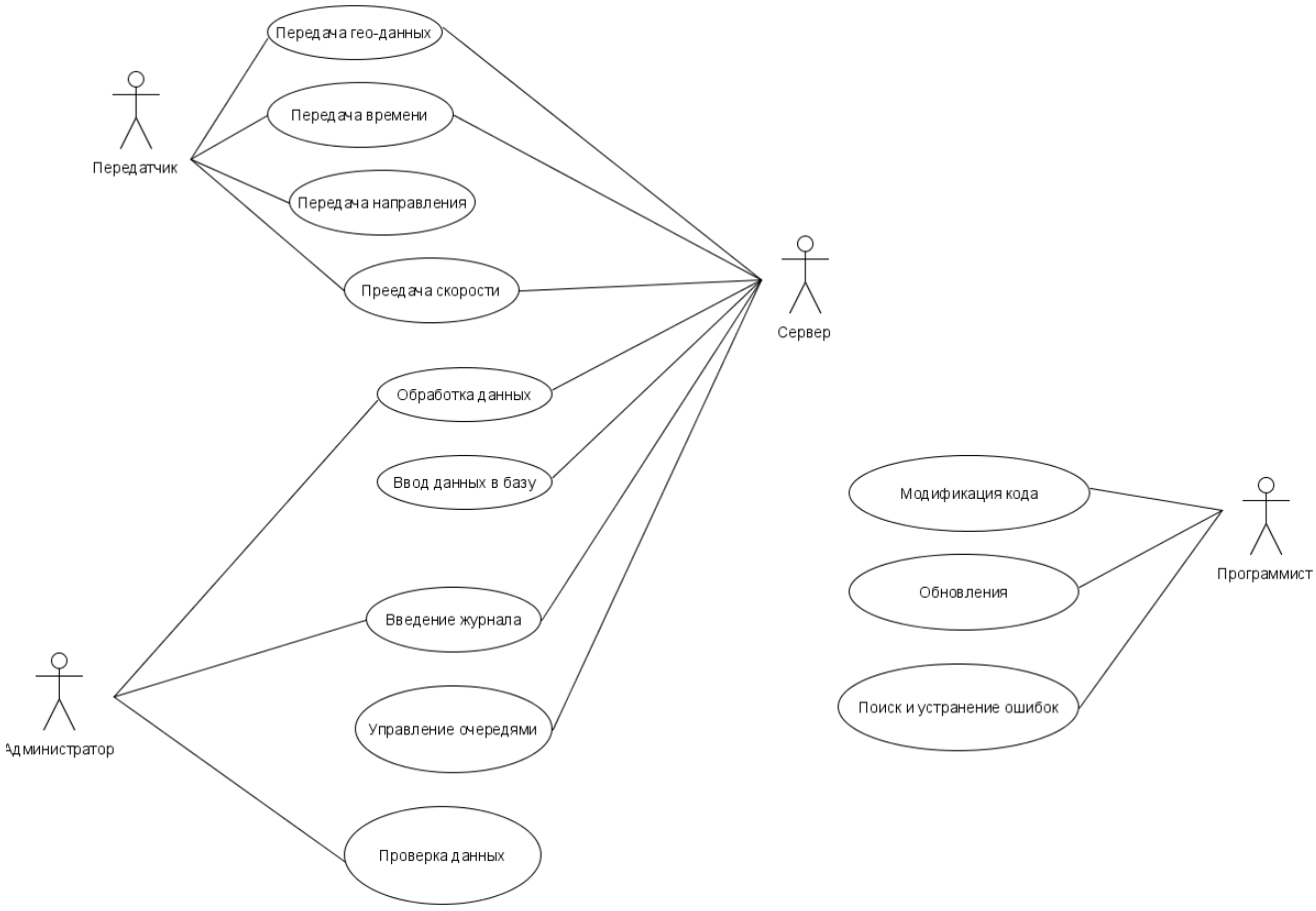

Рисунок 1.15 – Диаграмма прецедентов

На данной диаграмме вариантов использования актерами являются:

– администратор;

– передатчик;

– сервер;

– программист.

1.5.3 Разработка диаграммы объектов

Диаграмма объектов дает общее представление о структуре серверного приложения (рисунок 1.16) [12].

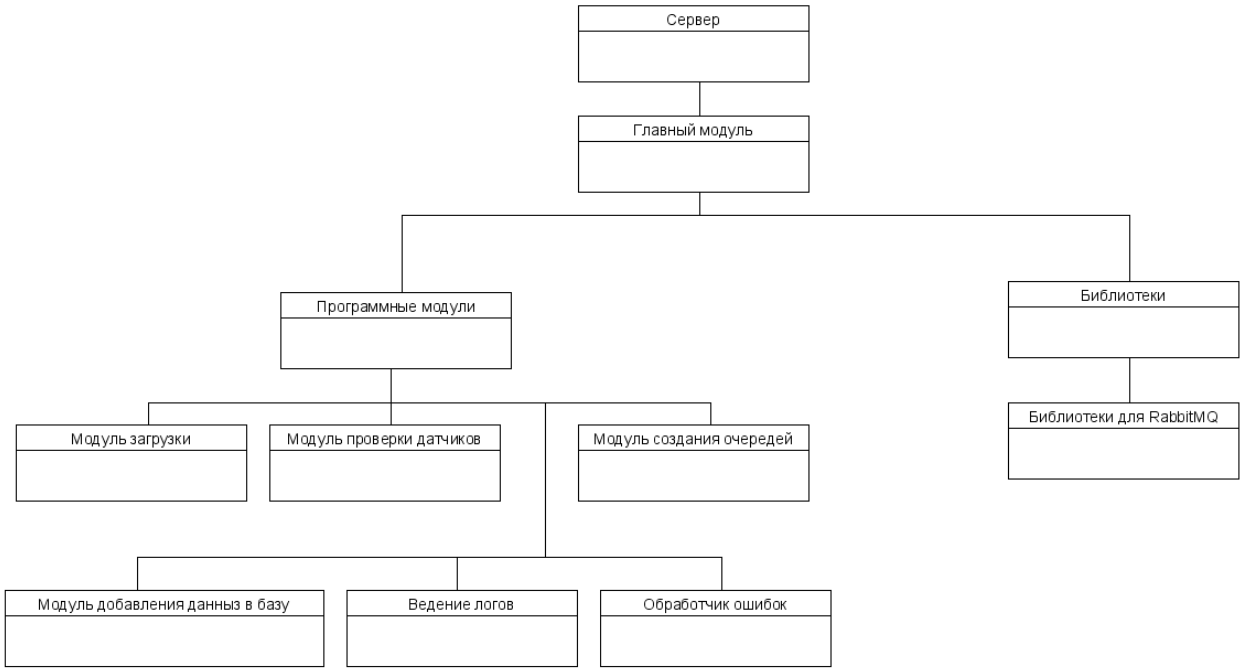

Рисунок 1.16 – Диаграмма объектов

Как видно из диаграммы (рисунок 1.16), программа разбита на несколько модулей, каждый из которых выполняет свою функцию. Определенные модули могут работать независимо (как, например, модуль ведения логов), некоторые же непосредственно связаны с другими модулями (например, модуль загрузки использует модуль ведения логов для сохранения записей о состоянии загрузки).

1.5.4 Разработка диаграммы состояний

Диаграмма состояний (Рисунок 1.17) описывает поведение системы.

На диаграмме представлены два особых состояния: вход и выход. Любое действие, связанное с событием входа, выполняется, когда объект входит в данное состояние [11].

Событие выхода выполняется в том случае, когда объект выходит из данного состояния.

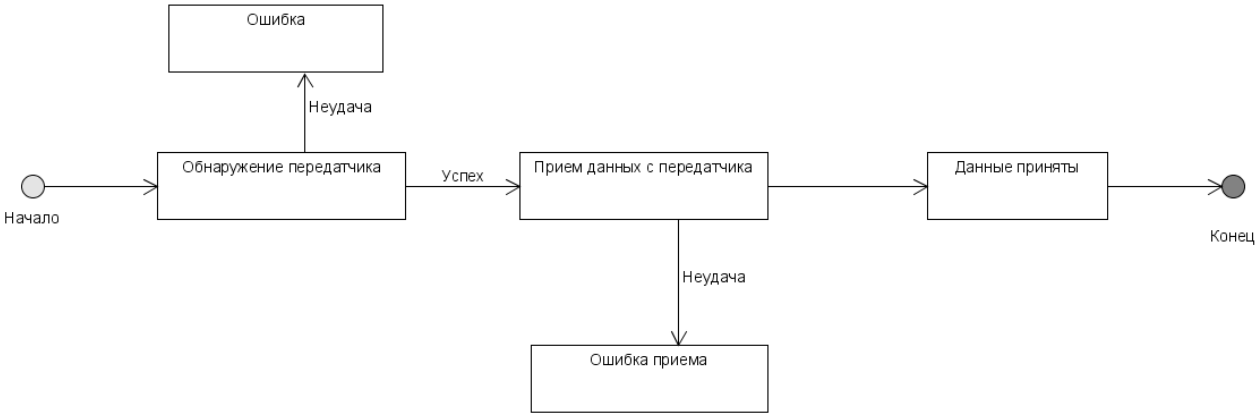

Рисунок 1.17 – Диаграмма состояний

1.5.5 Разработка диаграммы развертывания

Диаграмма развёртывания в UML моделирует физическое развертывание артефактов на узлах (рисунок 1.18) [13].

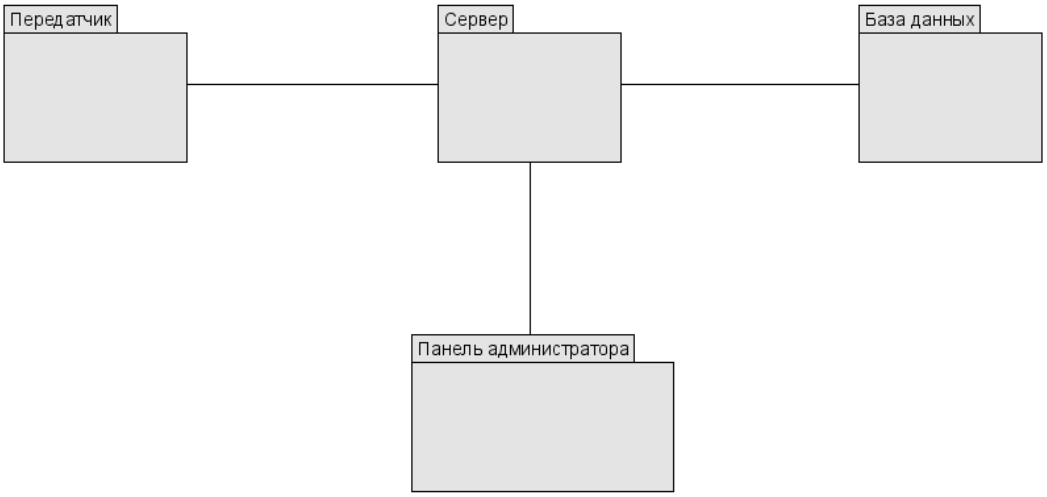

Рисунок 1.18 – Диаграмма развертывания

1.5.6 Проектирование базы данных

Для работы приложения понадобится база данных, содержащая следующие таблицы:

– таблица последних местоположений автомобилей, для быстрого получения информации об автомобиле, без необходимости перебора большого количества записей;

– параметры – дополнительных данные, которые может прислать бортовая система (например: пробег, суточное время стоянки, данные датчиков и так далее);

– записи – таблица, куда сохраняется история перемещений всех автомобилей;

– параметры записей – таблица, которая будет содержать дополнительные параметры для всех записей, которые пришли с сообщением.

Структура таблиц представлена на рисунке 1.19.

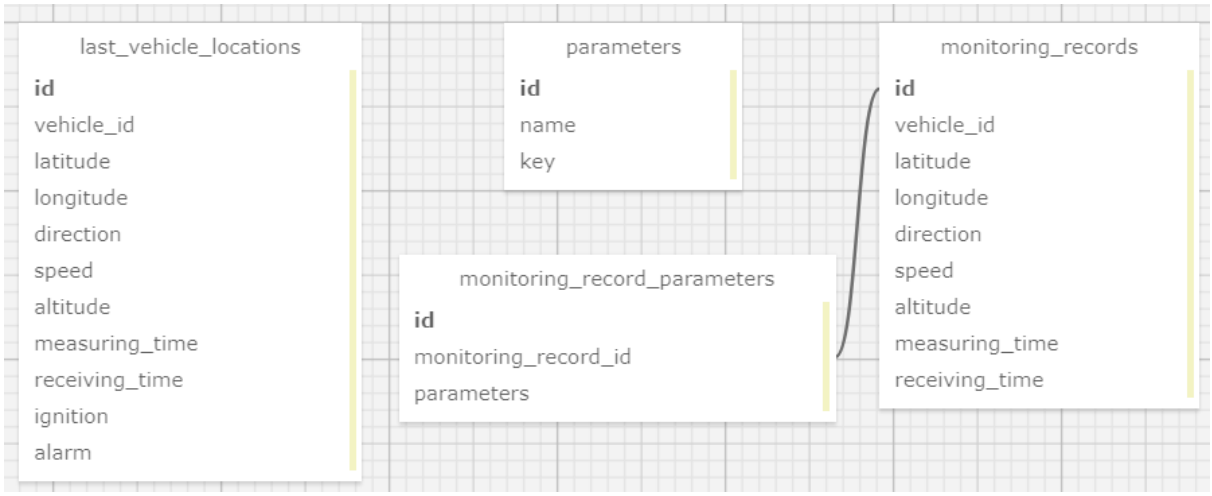

Рисунок 1.19 – Таблицы для навигационных данных

1.6 Разработка серверных приложений

1.6.1 Подготовка к разработке

Перед началом разработки, нужно установить необходимые для работы программные продукты:

- qt и qtcreator;
- postgresql
- postgis;
- rabbit-mq;
- pgadmin3.

Все программы за исключением rabbit-mq, можно установить с Улучшенным инструментарием пакетов (APT) Ubuntu. Подробная инструкция по установке всех компонентов приведена в приложении А.

После установки достаточно создать новый проект в Qt Creator.

1.6.2 Разработка модуль менеджера

Модуль менеджер отвечает за загрузку и запуск модулей программы, таких как: модуль работы с EGTS, модуль работы с NavTelecom, модуль логов, модуль работы с базой данных и так далее.

Структура модуль менеджера приведена на рисунке 1.20.

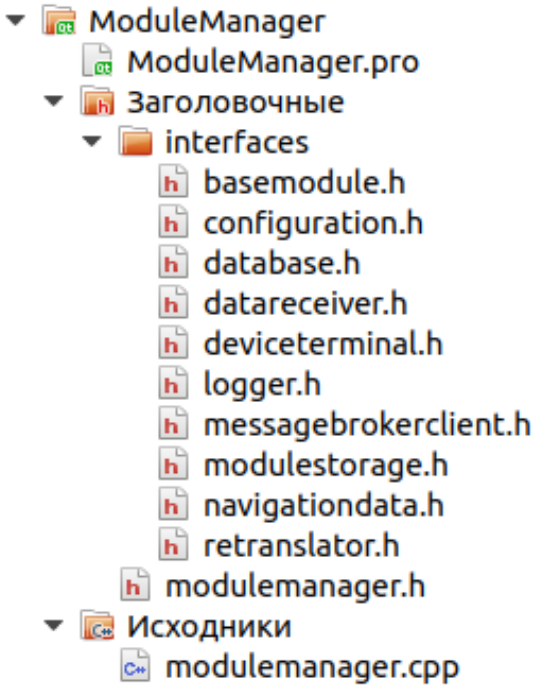

Рисунок 1.20 – Файлы модуль менеджера

Папка «interfaces» содержит некоторые интерфейсы для модулей. Описание этих интерфейсов:

– basemodule.h – Интерфейс базового модуля. Этот интерфейс должны поддерживать абсолютно все модули системы.

– configuration.h – Интерфейс модуля, предоставляющего функции конфигурации.

– database.h - Интерфейс модуля, предоставляющего функции доступа к базе данных.

– datareceiver.h – Интерфейс модуля, получающего навигационные данных от терминалов.

– deviceterminal.h – Интерфейс модуля, реализующего терминал для подключения приборов.

– logger.h - Интерфейс модуля, предоставляющего функции логирования.

– messagebrokerclient.h – Интерфейс модуля, связывающегося с диспетчером очереди сообщений.

– modelustorage.h – Интерфейс класса, предоставляющего модулям функции для обращения к другим модулям.

– navigationdata.h – Объявляет универсальную внутрисистемную структуру навигационных данных.

– retranslator.h – Интерфейс модуля, реализующего ретранслятор навигационных данных.

modulemanager.h – Интерфейс класса для работы с модулями.

modulemanager.cpp – Реализация класса, загружающего модули и реализующего функции взаимодействия модулей.

Основная задача модуль менеджера – загрузка и запуск модулей на выполнение. На рисунке 1.21 приведена функция для загрузки модулей из указанной папки.

```
/**
 * @brief Загружает модули из указанной папки
 * @return
 */
bool ModuleManager::loadModules(QDir modulesDir) {
    qDebug() << "Loading modules from " << qPrintable(modulesDir.absolutePath());

    foreach (QString fileName, modulesDir.entryList(QDir::Files)) {
        qDebug() << "Loading module " << qPrintable(modulesDir.absoluteFilePath(fileName));

        QPluginLoader loader(modulesDir.absoluteFilePath(fileName));
        QObject *module = loader.instance();
        if (!module) {
            qCritical("Error: %s", qPrintable(loader.errorString()));
            return false;
        }
        modules.append(module);
        moduleFiles[module] = fileName;
    }

    return true;
}
```

Рисунок 1.21 – Функция загрузки модулей

На рисунке 1.22 приведена функция для инициализации модулей.

```
/**
 * @brief Инициализирует все загруженные модули.
 * @return
 */
bool ModuleManager::initModules() {
    // список модулей для инициализации
    ModuleList remainingModules(modules);
    ModuleList initializedModules;

    // пробуем в цикле инициализировать все загруженные модули, удовлетворяя их зависимости
    while (true) {
        if (remainingModules.empty()) {
            return true;
        }
        // берем первый неинициализированный модуль
        QObject *module = remainingModules.at(0);
        // и инициализируем его и все его зависимости
        if (!initModuleAndAllRequirements(module, &remainingModules, &initializedModules)) {
            qCritical("Unable to initialize module %s", qPrintable(moduleFiles[module]));
            return false;
        }
    }
}
```

Рисунок 1.22 – Функция инициализации модулей

На рисунке 1.23 приведена функция для запуска модулей.

```
/**
 * @brief Запускает модули на выполнение.
 */
void ModuleManager::launchModules() {
    ModuleList::const_iterator i;
    for (i = modules.constBegin(); i != modules.constEnd(); i++) {
        Module module = *i;
        BaseModule *baseModule = qobject_cast<BaseModule *>(module);
        if (baseModule) {
            baseModule->launch();
        }
        else {
            qCritical("Unable to launch module %s", qPrintable(moduleFiles[module]));
        }
    }
}
```

Рисунок 1.23 – Функция для запуска модулей

В главном файле программы теперь достаточно указать папку, где находятся собранные модули, и модуль менеджер запустит их (рисунок 1.24).

```
#include <QCoreApplication>
#include <QDir>
#include <QtDebug>
#include "modulemanager.h"

int main(int argc, char *argv[]) {
    QCoreApplication a(argc, argv);

    QDir modulesDir = QDir(a.applicationDirPath());
    modulesDir.cdUp();
    modulesDir.cd("ServerModules/libs");

    ModuleManager moduleManager;

    // загружаем модули
    if (!moduleManager.loadModules(modulesDir)) {
        qCritical("Unable to load all required modules!");
        return -1;
    }

    // инициализируем модули
    if (!moduleManager.initModules()) {
        qCritical("Unable to init modules!");
        return -1;
    }

    // запускаем модули
    moduleManager.launchModules();

    return a.exec();
}
```

Рисунок 1.24 – Запуск модулей при запуске программы.

1.6.3 Разработка модуля конфигурации

Модуль конфигурации отвечает за чтение и запись настроек из файла типа «ini». Файл с конфигурацией содержит некоторые данные в формате ключ-значение. Конфигурационный файл будет хранить информацию для подключения к базе данных, настройки RabbitMQ, порты для запуска терминалов EGTS и NavTelecom, название файла логов и так далее. Пример файла конфигурации приведен на рисунке 1.25.

```
[granit_v3]
host=0.0.0.0
port=65100

[navtelecom]
host=0.0.0.0
port=65200

[egts]
host=0.0.0.0
port=65300
worker_threads=1

[log]
file_name=app.log

[postgres]
db=monitoring_2
host=localhost
password=postgres
port=5432
user=postgres
```

Рисунок 1.25 – Пример файла конфигурации

Модуль конфигурации должен уметь считывать и устанавливать значение по ключу. Данные функции приведены на рисунке 1.26.

```
/**
 * @brief       Возвращает значение из файла конфигурации по указанному ключу.
 * @param key   Ключ
 * @param value Значение
 * @return      Существование ключа в конфигурации
 */
bool IniConfiguration::getValue(QString key, QVariant &value) {
    if (!settings->contains(key)) {
        qCritical("File settings.ini doesn't contain key %s or fully empty" , qPrintable(key));
        return false;
    }
    value = settings->value(key);
    return true;
}

/**
 * @brief       Устанавливает значение в конфигурации по указанному ключу.
 * @param key   Ключ
 * @param value Знвчение
 */
void IniConfiguration::setValue(QString key, QVariant &value) {
    settings->setValue(key, value);
}
```

Рисунок 1.26 – Функции для возвращения и установки значения

1.6.4 Разработка модуля ведения логов

Основная задача модуля логов – предоставление функций для вывода сообщений в файл логов и на экран. Модуль содержит две основные функции:

функция инициализации (функция запускает модуль конфигурации, получает информацию о расположении файла логов и открывает его для чтения и записи), функция вывода сообщения (Рисунок 1.27).

```
/**
 * @brief           Выполняет фактический вывод сообщения в файл и на консоль.
 * @param type      Тип (уровень) сообщения.
 * @param message   Сообщение.
 */
void FileLogger::logMessage(QtMsgType type, const QString &message) {
    QMutexLocker locker(&mutex);
    QString finalMessage;
    QString messageType;

    switch (type) {
        case QtDebugMsg:
            messageType = "debug";
        break;
        case QtWarningMsg:
            messageType = "warning";
        break;
        case QtCriticalMsg:
            messageType = "critical";
        break;
        case QtFatalMsg:
            messageType = "fatal";
        break;
        default:
            messageType = "debug";
        break;
    }

    finalMessage = QString("%1 [%2] - %3").arg(QDateTime::currentDateTime().toString("dd.MM.yyyy hh:mm:ss")).arg(mes

    logFileStream << finalMessage << endl;
    logStdoutStream << finalMessage << endl;
}
```

Рисунок 1.27 – Функция вывода сообщения

1.6.5 Разработка модуля для работы с очередью сообщений

Модуль клиента RabbitMQ выполняет работу по созданию канала связи с брокером, объявляет очередь, запускает и останавливает прием сообщений на канале. Функция получения сообщения из очереди приведена на рисунке 1.28.

```
/**
 * @brief Выполняет получение сообщения из очереди.
 * @param channel   Канал.
 * @param message   Сообщение.
 * @param consumer_tag  Название приемника.
 * @param timeout   Таймаут получения сообщения.
 * @return Статус получения сообщения.
 */
bool RabbitMQClient::consumeMessage(void *channel, QByteArray *message, QString consumer_tag,
                                    int timeout) {
    AmqpClient::Channel::ptr_t clientChannel = *(AmqpClient::Channel::ptr_t *)channel;
    try {
        Envelope::ptr_t envelope;
        if (!clientChannel->BasicConsumeMessage(consumer_tag.toStdString(), envelope, timeout)) {
            return false;
        }
        amqp_bytes_t data = envelope->Message()->getAmqpBody();
        message->clear();
        message->append((char *)data.bytes, data.len);
    }
    catch (AmqpException &e) {
        logger->critical(QString("RabbitMQ error while message consuming: %1").arg(e.what()));
        return false;
    }

    return true;
}
```

Рисунок 1.28 – Функция получения сообщения из очереди

1.6.6 Разработка модуля подключения к базе данных

Модуль должен выполнять две основные задачи – подключение к базе данных и выполнение запросов к базе данных. Модуль содержит следующие функции:

функция инициализации – запускает модуль конфигурации, который позволяет получить необходимую информацию для подключения из конфигурационного файла и инициализирует подключение к базе данных.

функция выполнения запроса – получая параметризированную строку и параметры, выполняет запрос к базе данных и возвращает ответ. Функция выполнения запроса приведена на рисунке 1.29.

```
bool PostgresDB::query(QString queryString, QueryParameters *parameters, QueryResult *result) {
    QMutexLocker locker(&queryMutex);
    QSqlQuery query(db);
    if (!query.prepare(queryString)) {
        logger->warning(QString("Unable to prepare query: %1. Message: %2").arg(queryString)
                        .arg(query.lastError().text()));
        return false;
    }

    if (parameters != DatabaseQueryNoParameters) {
        QueryParameters::const_iterator i;
        for (i = parameters->constBegin(); i != parameters->constEnd(); i++) {
            query.addBindValue(*i);
        }
    }

    if (!query.exec()) {
        logger->warning(QString("Unable to exec query: %1. Message: %2")
                        .arg(getLastExecutedQuery(query)).arg(query.lastError().text()));
        return false;
    }

    if (result != DatabaseQueryNoResult) {
        result->clear();

        // получаем список полей в результате запроса
        QSqlRecord record = query.record();
        int fieldsCount = record.count();

        QList<QString> fields;
        for (int i = 0; i < fieldsCount; i++) {
            fields.append(record.field(i).name());
        }

        while (query.next()) {
            QueryResultRow row;
            for (int i = 0; i < fieldsCount; i++) {
                row[fields.at(i)] = query.value(i);
            }
            result->append(row);
        }
    }

    return true;
}
```

Рисунок 1.29 – Функция для выполнения запроса к базе данных

1.6.7 Реализация модуля для сохранения навигационных данных

Основной целью данного модуля является запись навигационных данных в базу данных. Для этого модуль должен иметь следующие функции:

функция инициализации – загружает модуль логирования, модуль связи с базой данных и модуль связи с диспетчером очереди сообщений

функция добавления данных в очередь – добавляет данные в очередь сообщений

функция записи данных из очереди – получает навигационные данные из очереди и отправляет их в базу данных.

На рисунке 30 приведена часть функции, отвечающая за считывание данных из очереди.

```
NavigationData data;
QByteArray rawData;
QDataStream rawDataStream(&rawData, QIODevice::ReadWrite);
rawDataStream.setVersion(QDataStream::Qt_5_3);

while (true) {
    if (!messageBrokerClient->consumeMessage(channel, &rawData, consumerTag)) {
        logger->critical("Unable to consume navigation data for data writer!");
        sleep(1);
        continue;
    }

    rawDataStream.device()->seek(0);

    rawDataStream >> data.vehicleId;
    rawDataStream >> data.deviceId;

    rawDataStream >> data.timestamp;
    rawDataStream >> data.latitude;
    rawDataStream >> data.longitude;
    rawDataStream >> data.direction;
    rawDataStream >> data.speed;
    rawDataStream >> data.altitude;

    rawDataStream >> data.ignition;
    rawDataStream >> data.alarm;

    rawDataStream >> data.additional;

    QueryParameters parameters;
    parameters.append(data.vehicleId);
    parameters.append(data.latitude);
    parameters.append(data.longitude);
    parameters.append(data.direction);
    parameters.append(data.speed);
    parameters.append(data.timestamp);
    parameters.append(data.altitude);
```

Рисунок 1.30 – Считывание данных из очереди

На рисунке 1.31 приведена часть функции, которая записывает полученные из очереди сообщений данные в базу данных и проверяет успешность операции.

```
// добавляем запись в таблицу monitoring
QueryResult result;
if (!db->query("INSERT INTO history.monitoring_records "
               "(vehicle_id, latitude, longitude, direction, speed, "
               "measuring_time, receiving_time, altitude) "
               "VALUES (?, ?, ?, ?, ?, "
               "to_timestamp(?) at time zone 'utc', now() at time zone 'utc', ?) "
               "RETURNING id;",
               &parameters, &result)) {
    logger->critical("Unable to insert navigation data!");
    continue;
}

if (result.count() == 0) {
    logger->critical("Couldn't get monitoring record id after insert.");
    continue;
}
```

Рисунок 1.31 – Запись навигационных данных в базу данных

1.6.8 Разработка модуля для работы с протоколом EGTS

Структура модуля EGTSTerminal имеет следующий вид (рисунок 1.32):

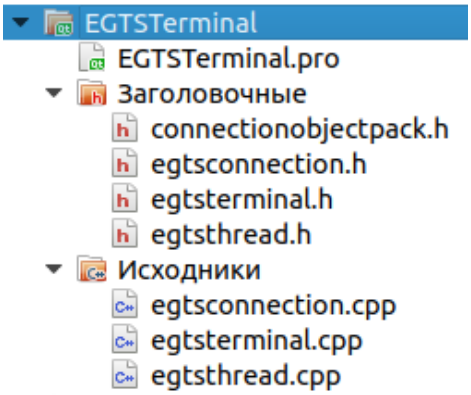

Рисунок 1.32 – Исходные файлы разрабатываемого модуля

Файл EGTSTerminal.pro содержит подключаемые модули, список ресурсов, тип файла после компиляции и т.д.

Файл connectiononobjectpack.h – небольшой интерфейс контейнера объектов, передающегося обработчику каждого соединения.

Файлы egtsthread.h и egtsthread.cpp – интерфейс и реализация класса потока, обслуживающего протокол EGTS. Главным образом служит для создания, обработки и очищения заданий по работе с пакетами.

Файлы egtsterminal.h и egtsterminal.cpp – интерфейс и реализация модуля терминала для протокола EGTS. Служит для инициализации всего модуля и обработки новых соединений. Обработчик события приема нового соединения проверяет наличие свободного потока, который должен принять данные от клиента (рисунок 1.33).

```
/**
 * @brief           Обработчик события приема нового соединения для терминала.
 * @param socket    Сокет.
 */
void EGTSTerminal::newConnection(qintptr handle) {
    logger->debug("New connection for EGTS accepted");

    EGTSThread *availableThread = threads[0];
    int lowestConnectionCount = INTMAX_MAX;

    for (int i = 0; i < threads.count(); i++) {
        if (lowestConnectionCount > threads[i]->getConnectionsCount()) {
            lowestConnectionCount = threads[i]->getConnectionsCount();
            availableThread = threads[i];
        }
    }

    QMetaObject::invokeMethod(availableThread, "newConnection", Q_ARG(qintptr, handle));
}
```

Рисунок 1.33 – Обработчик события приема нового соединения

Файлы egtsconnection.h и egtsconnection.cpp – интерфейс и реализация класса обработчика отдельного соединения по протоколу EGTS.

Для удобного разбора пакета удобно воспользоваться директивой #define, которая позволяет создать макросы. Например, создадим макросы для типов пакета транспортного уровня и различных сервисов (рисунок 1.34).

```
// Тип пакета Транспортного Уровня
#define EGTS_PT_RESPONSE                (0)
#define EGTS_PT_APPDATA                 (1)
#define EGTS_PT_SIGNED_APPDATA          (2)

// Сервисы
#define EGTS_AUTH_SERVICE               (1)
#define EGTS_TELEDATA_SERVICE           (2)
#define EGTS_COMMANDS_SERVICE           (4)
#define EGTS_FIRMWARE_SERVICE           (9)
#define EGTS_ECALL_SERVICE              (10)

#define EGTS_SR_RECORD_RESPONSE         (0)
#define EGTS_SR_TERM_IDENTITY           (1)
#define EGTS_SR_MODULE_DATA             (2)
#define EGTS_SR_VEHICLE_DATA            (3)
#define EGTS_SR_DISP_IDENTITY           (5)
#define EGTS_SR_AUTH_PARAMS             (6)
#define EGTS_SR_AUTH_INFO               (7)
#define EGTS_SR_SERVICE_INFO            (8)
#define EGTS_SR_RESULT_CODE             (9)
```

Рисунок 1.34 – Макросы

Для разбора каждого пакета удобно использовать структуры. Так как заранее известна длина каждого поля отдельного пакета, достаточно создать поля структуры различной длины. Например, первый байт заголовка сообщения всегда содержит версию протокола, следовательно, в структуре EGTS_header создаем поле PRV типа данных quint8 (или unsigned byte). На рисунке 1.35 приведен пример структуры для хранения заголовка сообщения.

```
#pragma pack(1)
typedef struct {
    quint8  PRV;        // (Protocol Version)
    quint8  SKID;       // (Security Key ID)

    // Flags:
    quint8  PR  : 2;
    quint8  CMP : 1;
    quint8  ENA : 2;
    quint8  RTE : 1;
    quint8  PRF : 2;

    quint8  HL;         // Длина заголовка Транс
    quint8  HE;         // метод кодирования сле
    quint16 FDL;        // размер в байтах поля
    quint16 PID;        // номер пакета Транспор
    quint8  PT;         // Тип пакета Транспортн
    quint8  HSC;        // Контрольная сумма заг

} EGTS_Header;
```

Рисунок 1.35 – Структура для заголовка

Таким же образом создаем структуры для всех остальных частей пакета: тело сообщения, контрольная сумма. Так как в зависимости от типа пакета, тело сообщения может быть разное, необходимо создать несколько структур для разных типов пакета. Например, на рисунке 1.36 приведен пример структуры для пакета авторизации, а на рисунке 1.37 пример структуры для пакета с данными о передвижении автомобиля.

```
typedef struct {
    EGTS_Header header;
    EGTS_RecordHeader recordHeader;
    EGTS_SubRecordHeader subRecordHeader;

    quint32 TID;

    quint8  HDIDE   : 1;
    quint8  IMEIE   : 1;
    quint8  IMSIE   : 1;
    quint8  LNGCE   : 1;
    quint8  SSRA    : 1;
    quint8  NIDE    : 1;
    quint8  BSE     : 1;
    quint8  MNE     : 1;

    quint8    IMEI[EGTS_IMEI_LEN];

    quint16 SFRCS;
} EGTS_authPacket;
```

Рисунок 1.36 – Структура для части с авторизацией

```
typedef struct {
    EGTS_Header header;
    EGTS_RecordHeader recordHeader;
    EGTS_SubRecordHeader subRecordHeader;

    quint32 NTM;            //Navigation Time
    quint32 LAT;
    quint32 LONG;

    quint8  VLD     : 1;
    quint8  FIX     : 1;
    quint8  CS      : 1;
    quint8  BB      : 1;
    quint8  MV      : 1;
    quint8  LAHS    : 1;
    quint8  LOHS    : 1;
    quint8  ALTE    : 1;

    quint8  SPDL;
    quint8  SPDH    : 6;
    quint8  ALTS    : 1;
    quint8  DIRH    : 1;

    quint8  DIR;
    quint8  ODM[3];
    quint8  DIN;
    quint8  SRC;

    quint8  ALT[3];

    quint16 SFRCS;
} EGTS_telemPacket;
```

Рисунок 1.37 – Структура для данных с навигацией

Необходимо создать небольшую функцию для определения типа сообщения. Функция приведена на рисунке 1.38.

```
/**
 * @brief   Парсит информационный пакет.
 * @return  Тип пакета.
 */
qint8 EGTSConnection::checkTypeOfPacket() {

    EGTS_DataPacket *dataPacket = (EGTS_DataPacket *)currentPacket.data();

    if(dataPacket->header.PT != EGTS_PT_APPDATA)
    {
        return -1;
    }

    return dataPacket->recordHeader.SST;
}
```

Рисунок 1.38 – Функция для определения типа пакета

После создания всех необходимых структур, необходимо создать функцию для парсинга пакета. На рисунке 1.39 приведена функция, которая разбирает

заголовок сообщения и в зависимости от типа пакета вызывает различные функции для последующего разбора.

```
bool EGTSConnection::parsePacket(QByteArray *data) {
    // убираем статус получения заголовка для обработки следующего пакета
    headerReceived = false;

    // добавляем данные к пакету
    currentPacket.append(*data);

    switch (checkTypeOfPacket()) {
    case EGTS_AUTH_SERVICE:
        if (parseAuthPacket()) {
            brokenPacketsCount = 0;
        }
        else {
            brokenPacketsCount++;
        }
        break;
    case EGTS_TELEDATA_SERVICE:
        if (parseTelemPacket()) {
            brokenPacketsCount = 0;
        }
        else {
            brokenPacketsCount++;
        }
        break;
    case (-1):
        logger->warning("Received not APPDATA EGTS Packet!");
    default:

        break;
    }

    return true;
}
```

Рисунок 1.39 – Функция разбора информационного пакета

На рисунках 1.40 и 1.41 приведена функция для разбора пакета с навигационными данными.

```
/**
 * @brief   Парсит информационный пакет с навигационными данными.
 * @return  Статус.
 */
bool EGTSConnection::parseTelemPacket() {
    NavigationData navData;
    EGTS_telemPacket *dataPacket = (EGTS_telemPacket *)currentPacket.data();
    if(dataPacket->header.PRV != 0x01 || dataPacket->header.PRF != 0x00)
    { logger->warning("Received unknown protocol version!");
        return false; }
    if(dataPacket->header.HL < 11 || dataPacket->header.HL > 16)
    { logger->warning("Received unknown header!");
        return false; }
    if(EGTS_CRC8((unsigned char *)&dataPacket->header, sizeof(EGTS_Header) - sizeof(quint8)) != dataPacket->header.HSC)
    { logger->warning("EGTS_PC_HEADERCRC_ERROR");
        return false; }
    if(dataPacket->header.FDL <= 0)
    { logger->warning("Received empty data!");
        return false; }

    lastPacketId = dataPacket->header.PID;
    packetParsed = true;

    if ((quint8)currentPacket.length() < sizeof(EGTS_telemPacket)) {
        logger->warning("Received unknown EGTS packet!");
        return false; }

    if (deviceId == -1) {
        deviceSerialNumber = QString::number(dataPacket->recordHeader.OID);

        if (!checkDeviceDbStatus()) {
            return false; }
    }
```

Рисунок 1.40 – Функция для разбора пакета с навигацией (Часть 1)

```
        navData.vehicleId = vehicleId;
        navData.deviceId = deviceSerialNumber;
        navData.additional[AdditionalNavigationDataKeys::DEVICE_IMEI] = deviceSerialNumber;

        navData.timestamp = dataPacket->NTM + 1262293200;
        navData.latitude = coordinateFromEgts(dataPacket->LAT, 90.0f);
        navData.longitude = coordinateFromEgts(dataPacket->LONG, 180.0f);
        navData.additional[GPS_DATA_VALID] = (bool)dataPacket->VLD;
        navData.altitude = 0;
        if (dataPacket->LOHS) {
            navData.longitude = -navData.longitude;
        }
        if (dataPacket->LAHS) {
            navData.latitude = -navData.latitude;
        }
        int speed = dataPacket->SPDL | ((dataPacket->SPDH << 8) & 0x3F);
        navData.speed = speed / 10;
        navData.direction = dataPacket->DIR | ((dataPacket->DIRH << 8) & 0x01);

        logger->debug(QString("EGTS device %1 time: %2, lat: %3, lon: %4")
                      .arg(deviceSerialNumber).arg(navData.timestamp).arg(navData.latitude)
                      .arg(navData.longitude));

        registerNewNavigationData(&navData);

    return true;
}
```

Рисунок 1.41 – Функция для разбора пакета с навигацией (Часть 2)

После разбора пакета и проверки его целостности, необходимо отправить ответ на бортовую систему и записать все данные в базу данных. На этом прием сообщения по протоколу EGTS завершен.

1.6.9 Разработка модуля для работы по протоколу NavTelecom

Обмен сообщениями по протоколу NavTelecom мало отличается от обмена по протоколу EGTS, значительную часть изменений представляет структура информационного пакета. Общая схема функционирования модуля такая же, как и у модуля EGTS – запускается сервер для терминала, который производит прием сообщений через указанный порт, каждое сообщение обрабатывается в соответствие с протоколом NavTelecom, программа отправляет сообщение о приеме пакета, программа записывает данные в базу данных.

Структура модуля NavTelecomTerminal имеет следующий вид (рисунок 1.42):

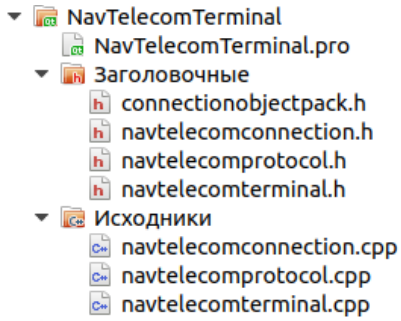

Рисунок 1.42 – Исходные файлы разрабатываемого модуля

Файл NavTelecomTerminal.pro содержит подключаемые модули, список ресурсов, тип файла после компиляции и т.д.

Файл connectiononobjectpack.h – небольшой интерфейс контейнера объектов, передающегося обработчику каждого соединения.

Файлы navtelecomterminal.h и navtelecomterminal.cpp – интерфейс и реализация модуля терминала для протокола NavTelecom. Служит для инициализации всего модуля и обработки новых соединений (рисунок 1.43).

```
/**
 * @brief           Обработчик события приема нового соединения для терминала.
 * @param socket    Сокет.
 */
void NavTelecomTerminal::newConnection(qintptr handle) {
    logger->debug("New connection for NavTelecom device accepted");

    NavTelecomConnection *newConnection = new NavTelecomConnection(handle, objectPack);
    connect(newConnection, SIGNAL(finished()), newConnection, SLOT(deleteLater()));
    newConnection->start();
}
```

Рисунок 1.43 – Обработчик события приема нового соединения

Файлы navtelecomconnection.h и navtelecomconnection.cpp – интерфейс и реализация класса обработчика отдельного соединения по протоколу NavTelecom. Класс обслуживает отдельное соединение, регистрирует прибор и навигационные данные в базе данных, парсит информационный пакет и отправляет ответ, используя функции класса navtelocomprotocol. На рисунке 44 приведена функция отправки навигационных данных в очередь сообщений.

```
/**
 * @brief       Регистрирует новые навигационные данные в приемниках.
 * @param data  Навигационные данные.
 */
void NavTelecomConnection::registerNewNavigationData(NavigationData *data) {
    QByteArray packedData;
    QDataStream dataStream(&packedData, QIODevice::ReadWrite);
    dataStream.setVersion(QDataStream::Qt_5_3);
    dataStream << data->vehicleId;
    dataStream << data->deviceId;

    dataStream << data->timestamp;
    dataStream << data->latitude;
    dataStream << data->longitude;
    dataStream << data->direction;
    dataStream << data->speed;
    dataStream << data->altitude;

    dataStream << data->ignition;
    dataStream << data->alarm;

    dataStream << data->additional;

    if (!messageBrokerClient->sendMessage(channel, NAVIGATION_DATA_EXCHANGE_NAME, "", &packedData)) {
        logger->critical("Unable to send NavTelecom navigation data to message broker!");
    }
}
```

Рисунок 1.44 – Регистрация навигационных данных

Файлы navtelecomprotocol.h и navtelecomprotocol.cpp – интерфейс и реализация класса для работы с сообщениями по протоколу NavTelocom. Класс предоставляет удобные функции для парсинга информационных пакетов различных типов и функции для составления пакета для отправки на терминал.

NavTelecom работает с несколькими типами сообщений. Любое сообщение имеет заголовок. На рисунке 1.45 приведена структура для парсинга заголовка).

```
#pragma pack(1)
struct NavTelecomHeader {
    quint8      preamble[4];
    quint32     receiverId;
    quint32     senderId;
    quint16     dataLength;
    quint8      dataCRC;
    quint8      headerCRC;
};
```

Рисунок 1.45 – Структура заголовка сообщения

Тело сообщения может отличаться в зависимости от типа пакета, поэтому перед парсингом программа должна определить тип полученного сообщения. Затем сообщение разбирается в зависимости от типа, например, на рисунке 1.46 приведена структура для парсинга пакета типа NavTelecomData_4

```
struct NavTelecomData_4 {
    quint8      packetType;
    quint32     recordId;
    quint16     eventId;
    quint8      generationTime[6];
    quint8      guardingStateAndAlarm;
    quint8      modulesStatus;
    quint8      gsmSignalStrength;
    quint8      outputState;
    quint16     inputState;
    quint16     mainPowerSupplyVoltage;
    quint16     reservedPowerSupplyVoltage;
    qint8       temperature1;
    qint8       temperature2;
    qint8       temperature3;
    qint8       temperature4;
};
```

Рисунок 1.46 – Структура тела сообщения

В зависимости от типа сообщения, в теле может находиться больше или меньше навигационных данных и информации о статусе автомобиля.

После парсинга пакета, информация отправляется в базу данных.

1.6.10 Проверка работоспособности программы

Для запуска программы необходимо создать новый экран в программе «screen», чтобы приложение могло работать без остановки. Сделать это можно следующей командой:

screen –S deviceServer

Затем необходимо добавить путь к библиотекам в переменную окружения. Для этого используется команда:

export LD_LIBRARY_PATH=~/projects/MonitoringSystem/DeviceTerminal

После проделанных действий можно запускать программу. Для этого можно использовать команды:

cd ~/projects/MonitoringSystem/DeviceTerminal

./DeviceTerminal

После запуска программа уведомит пользователя об успешной загрузке и инициализации модулей. Затем программа запустит TCP серверы для работы терминалов и начнет принимать подключения (рисунок 1.47).

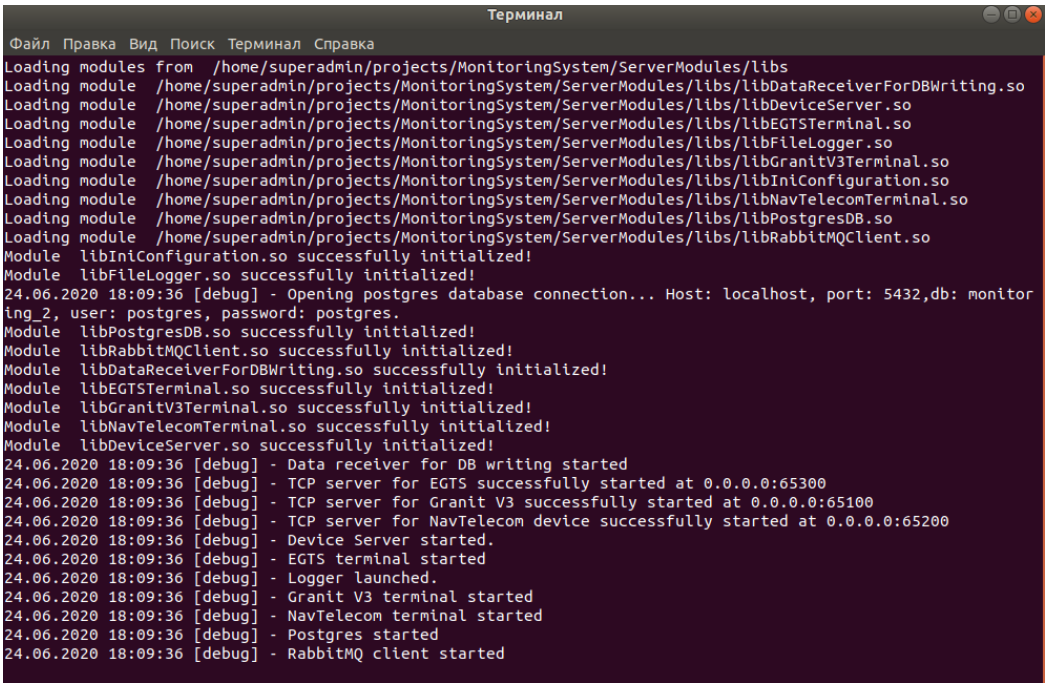

Рисунок 1.47 – Запуск программы

При получении нового сообщения от бортовой системы, программа выведет эту информацию на экран и отправит ее в базу данных (рисунок 1.48-1.49).

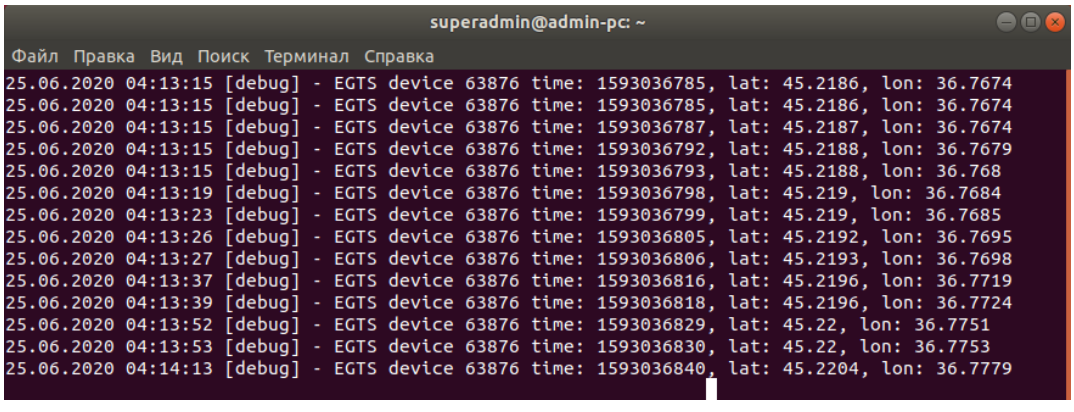

Рисунок 1.48 – Прием данных

	id [PK] serial	vehicle_id integer	latitude double precision	longitude double precision	direction integer	speed integer	altitude integer	measuring_time timestamp without time zone	receiving_time timestamp without
1	1635636	12	45.2310683148054	36.8455616563665	94	9	0	2020-06-24 22:17:45	2020-06-25 01:17:47
2	1635635	12	45.2311049856318	36.844023325677	89	10	0	2020-06-24 22:17:40	2020-06-25 01:17:43
3	1635634	12	45.2310166613271	36.8418749600746	83	10	0	2020-06-24 22:17:33	2020-06-25 01:17:34
4	1635633	12	45.2308549953696	36.8403499985208	78	10	0	2020-06-24 22:17:28	2020-06-25 01:17:29
5	1635632	12	45.2305866487395	36.8388466622771	73	10	0	2020-06-24 22:17:23	2020-06-25 01:17:28
6	1635631	12	45.2300033311429	36.8365199716847	68	10	0	2020-06-24 22:17:15	2020-06-25 01:17:26
7	1635630	12	45.2283549879744	36.8304449778121	69	10	0	2020-06-24 22:16:54	2020-06-25 01:16:56
8	1635629	12	45.2281233331254	36.8295716254109	69	10	0	2020-06-24 22:16:51	2020-06-25 01:16:52
9	1635628	12	45.2271683200326	36.8257799876914	70	10	0	2020-06-24 22:16:38	2020-06-25 01:16:49
10	1635627	2	44.1965476586597	43.1341612141902	308	0	0	2020-06-25 01:15:28	2020-06-25 01:16:42
11	1635626	12	45.225723321835	36.81961832494	71	10	0	2020-06-24 22:16:17	2020-06-25 01:16:18
12	1635625	12	45.2252249967366	36.8172449890564	76	10	0	2020-06-24 22:16:09	2020-06-25 01:16:12
13	1635624	12	45.2246399818046	36.8117666609613	81	10	0	2020-06-24 22:15:51	2020-06-25 01:15:53

Рисунок 1.49 – Сохраненные данные

Таким образом, приложение принимает информацию с терминалов в круглосуточном режиме и записывает ее в базу данных. В дальнейшем эти данные могут использоваться клиентами или для анализа.

2 Разработка API обмена данными между клиентскими и серверными приложениями АИС «Управление транспортом»

2.1 Теоретический анализ. Разработка базы данных

2.1.1 Теория и анализ необходимых средств разработки REST API

В данной работе рассматривается разработка REST API для АИС «Управление транспортом». АИС представляет собой набор нескольких компонентов, а именно:

- сервер, реализованный на Qt;
- клиентская часть web-приложения;
- серверная часть web-приложения;
- база данных на PostgreSQL.

В ходе разработки REST API был выбран язык PHP. Подобный выбор обусловлен тем, что на сегодняшний день PHP является наиболее распространенным языком веб-программирования. Подавляющие большинство сайтов и веб-сервисов в интернете написано именно с его помощью - по некоторым оценкам PHP применяется более чем на 80% сайтов, среди которых такие сервисы, как facebook.com, vk.com, baidu.com и другие. Такая популярность неудивительна, ведь простота языка в освоении позволяет быстро и легко создавать сайты и порталы, в то время как глубокий функционал позволяет решать задачи различной сложности.

PHP был создан в 1994 году датским программистом Расмусом Лердорфом и изначально представлял собой набор скриптов на другом языке Perl. Позже этот набор скриптов был переписан в интерпретатор на языке C. И с самого возникновения PHP (сокращение от PHP: Hypertext Preprocessor - PHP: Препроцессор гипертекста) представлял удобный набор инструментов для создания веб-сайтов и веб-приложений [1].

У PHP есть ряд неоспоримых преимуществ:

– для всех наиболее распространенных операционных систем (Windows, MacOS, Linux) есть свои версии пакетов разработки на PHP, то есть существует возможность создавать веб-сайты на любой из перечисленных ОС;

– PHP может работать в связке с различными веб-серверами: Apache, Nginx, IIS;

– простота и легкость освоения. Как правило, уже имея небольшой опыт в программировании на PHP, можно создавать простые веб-сайты;

– PHP поддерживает работу с множеством систем баз данных (MySQL, MSSQL, Oracle, Postgre, MongoDB и другие);

– распространенность хостинговых услуг и их дешевизна. Как правило, хостинговые компании размещают веб-сайты на PHP на веб-серверах Apache или Nginx, которые работают на одной из операционных систем семейства Linux. И веб-серверы, и операционные системы на базе Linux бесплатны, что снижает общую стоимость использования хостинга;

– постоянное развитие. PHP продолжает развиваться, выходят все новые версии, которые несут новые функции, адаптируя язык программирования к новым задачам. Как правило, переход на новую версию не составляет труда [1].

Популярность PHP в области построения веб-сайтов определяется так же наличием большого набора встроенных средств и дополнительных модулей для разработки веб-приложений. Основные из них:

1. автоматическое извлечение POST- и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
2. взаимодействие с большим количеством различных систем управления базами данных через дополнительные модули (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8);
3. автоматизированная отправка HTTP-заголовков;
4. работа с HTTP-авторизацией;
5. работа с cookies и сессиями;
6. работа с локальными и удалёнными файлами, сокетами;
7. обработка файлов, загружаемых на сервер;

8. работа с XForms.

Важным преимуществом языка PHP перед такими языками, как Perl и C заключается в возможности создания HTML документов с внедренными командами PHP. Значительным отличием PHP от какого-либо кода, выполняющегося на стороне клиента, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера.

PHP является языком программирования с динамической типизацией, не требующим указания типа при объявлении переменных, равно как и самого объявления переменных [2]. Преобразования между скалярными типами зачастую осуществляются неявно и без дополнительных усилий (впрочем, PHP предоставляет широкие возможности и для явного преобразования типов).

К скалярным типам данных относятся:

– целочисленный тип (integer);
– число с плавающей точкой (float, double);
– логический тип (boolean);
– строковый тип (string);

К нескалярным типам относятся:

– массив (array);
– объект (object);
– внешний ресурс (resource);
– неопределенное значение (null).

К псевдотипам относятся:

– mixed любой тип;
– number число (integer либо float);
– callback (анонимная функция, string или массив);
– void отсутствие параметров;

PHP является интерпретируемым языком – то есть, программы на PHP выполняются построчно, в отличие от компилируемых программ, где весь текст программы, перед запуском, анализируется и транслируется в машинный или

байт-код. PHP-скрипты обычно обрабатываются интерпретатором в порядке, обеспечивающем кроссплатформенность разработанного приложения:

1. лексический анализ исходного кода и генерация лексем;
2. синтаксический анализ полученных лексем;
3. генерация байт-кода;
4. выполнение байт-кода интерпретатором (без создания исполняемого файла).

Для увеличения быстродействия приложений возможно использование специального программного обеспечения, так называемых акселераторов. Принцип их работы заключается в кэшировании однажды сгенерированного байт-кода в памяти и/или на диске, таким образом, из процесса работы приложения исключаются этапы 1-3, что в общем случае ведёт к значительному ускорению работы [2].

Важной особенностью является то, что разработчику нет необходимости заботиться о распределении и освобождении памяти. Ядро PHP реализует средства для автоматического управления памятью; вся выделенная память возвращается системе после завершения работы скрипта.

REST – это архитектурный стиль взаимодействия компонентов распределенного приложения в сети. Он был разработан в диссертации Роя Филдинга в 200 году, как альтернатива SOAP, когда запрос клиента несет в себе исчерпывающую информацию о желаемом ответе сервера и сервер не обязан сохранять сессию взаимодействия с клиентом [3].

В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

Свойства архитектуры, которые зависят от ограничений, наложенных на REST-системы:

– Производительность — взаимодействие компонентов системы может являться доминирующим фактором производительности и эффективности сети с точки зрения пользователя;

– Масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов.

– Рой Филдинг — один из главных авторов спецификации протокола HTTP, описывает влияние архитектуры REST на масштабируемость следующим образом:

– Простота унифицированного интерфейса;

– Открытость компонентов к возможным изменениям для удовлетворения изменяющихся потребностей (даже при работающем приложении);

– Прозрачность связей между компонентами системы для сервисных служб;

– Переносимость компонентов системы путем перемещения программного кода вместе с данными;

– Надёжность, выражающаяся в устойчивости к отказам на уровне системы при наличии отказов отдельных компонентов, соединений или данных.

Существует шесть обязательных ограничений для построения распределённых REST-приложений по Филдингу [3].

Выполнение этих ограничительных требований обязательно для REST-систем.

1. Модель клиент-сервер. Первым ограничением, применимым к гибридной модели, является приведение архитектуры к модели клиент-сервер. Разграничение потребностей является принципом, лежащим в основе данного накладываемого ограничения. Отделение потребности интерфейса клиента от потребностей сервера, хранящего данные, повышает переносимость кода клиентского интерфейса на другие платформы, а упрощение серверной части улучшает масштабируемость. Наибольшее же влияние на всемирную паутину, пожалуй, имеет само разграничение, которое

позволяет отдельным частям развиваться независимо друг от друга, поддерживая потребности в развитии интернета со стороны различных организаций.

2. Отсутствие состояния. Протокол взаимодействия между клиентом и сервером требует соблюдения следующего условия: в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится (Stateless protocol или «протокол без сохранения состояния»). Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. *Состояние* сессии при этом сохраняется на стороне клиента. Информация о состоянии сессии может быть передана сервером какому-либо другому сервису (например, в службу базы данных) для поддержания устойчивого состояния, например, на период установления аутентификации. Клиент инициирует отправку запросов, когда он готов (возникает необходимость) перейти в новое состояние.

Во время обработки клиентских запросов считается, что клиент находится в переходном состоянии. Каждое отдельное состояние приложения представлено связями, которые могут быть задействованы при следующем обращении клиента.

3. Кэширование. Как и во Всемирной паутине, клиенты, а также промежуточные узлы, могут выполнять кэширование ответов сервера. Ответы сервера, в свою очередь, должны иметь явное или неявное обозначение как кэшируемые или некэшируемые с целью предотвращения получения клиентами устаревших или неверных данных в ответ на последующие запросы. Правильное использование кэширования способно полностью или частично устранить некоторые клиент-серверные взаимодействия, ещё больше повышая производительность и расширяемость системы.

4. Единообразие интерфейса. Наличие унифицированного интерфейса является фундаментальным требованием дизайна REST-сервисов. Унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо.

К унифицированным интерфейсам предъявляются следующие ограничительные условия:

– Идентификация ресурсов - все ресурсы идентифицируются в запросах, например, с использованием URI в интернет-системах. Ресурсы концептуально отделены от представлений, которые возвращаются клиентам. Например, сервер может отсылать данные из базы данных в виде HTML, XML или JSON, ни один из которых не является типом хранения внутри сервера.

– Манипуляция ресурсами через представление - если клиент хранит представление ресурса, включая метаданные — он обладает достаточной информацией для модификации или удаления ресурса.

– «Самоописываемые» сообщения - Каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать.

5. Слои. Клиент обычно не способен точно определить, взаимодействует он напрямую с сервером или же с промежуточным узлом, в связи с иерархической структурой сетей (подразумевая, что такая структура образует слои).

6. Код по требованию. REST может позволить расширить функциональность клиента за счёт загрузки кода с сервера в виде апплетов или сценариев.

RESTful - приложение имеет набор следующих преимуществ:

– надёжность;

– производительность;

– масштабируемость;

– прозрачность системы взаимодействия;

– простота интерфейсов;

– портативность компонентов;

– лёгкость внесения изменений;

– способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины).

RESTful API сводится к четырем базовым операциям:

- получение данных в удобном для клиента формате;

- создание новых данных;
- обновление данных;
- удаление данных.

REST функционирует поверх протокола HTTP, поэтому стоит упомянуть о его основных особенностях. Для каждой операции указанной выше используется свой собственный HTTP метод:

- GET – получение;
- POST – создание;
- PUT – обновление, модификация;
- DELETE – удаление.

Все эти методы в совокупности называют CRUD (create, read, update, delete) – (создать, прочитать, обновить, удалить) операциями [3].

Создание RESTful приложения с нуля может быть сложным процессом, создающим массу возможностей для ошибок и/или несовершенств реализации. Поэтому частой практикой является использование различных фреймворков.

Фреймворк (англ. Framework — остов, каркас, структура) — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Можно также говорить о каркасном подходе как о подходе к построению программ, где любая конфигурация программы строится из двух частей:

1. Постоянная часть — каркас, не меняющийся от конфигурации к конфигурации и несущий в себе гнёзда, в которых размещается вторая, переменная часть;

2. Сменные модули (или точки расширения).

На первый взгляд фреймворк очень похож на библиотеку. Однако, фреймворк отличается от библиотеки тем, что библиотека может быть использована в программном продукте просто как набор подпрограмм близкой функциональности, не влияя на архитектуру программного продукта и не накладывая на неё никаких ограничений. В то время как фреймворк диктует

правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию — «каркас», который нужно будет расширять и изменять, согласно указанным требованиям.

Также, в отличие от библиотеки, которая объединяет в себе набор близкой функциональности, — фреймворк может содержать в себе большое число разных по тематике библиотек.

Другим ключевым отличием фреймворка от библиотеки может быть инверсия управления: пользовательский код вызывает функции библиотеки (или классы) и получает управление после вызова. Во фреймворке пользовательский код может реализовывать конкретное поведение, встраиваемое в более общий, абстрактный код фреймворка. При этом фреймворк вызывает функции (классы) пользовательского кода.

На рынке представлено большое количество самых различных PHP-фреймворков (рисунок 2.1). Представляя большинство базовых инструментов, необходимых для решения задач web-программирования, каждый из фреймворков имеет так же более специфичные функции, направленные на решение более узких проблем. Так же разные фреймворки могут обладать разными спецификациями, такими как объем исходного кода, простота в освоении и тому подобное.

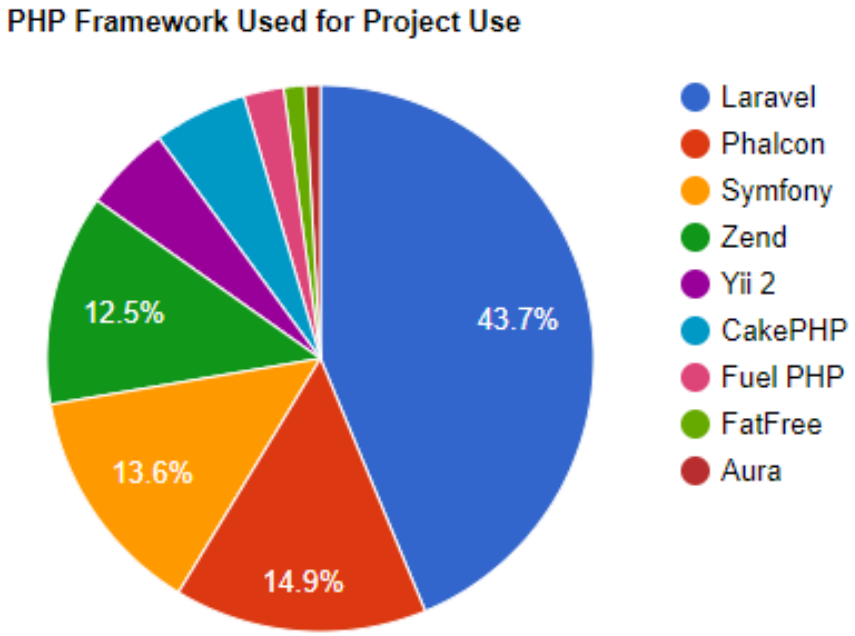

Рисунок 2.1 – Популярные фреймворки по результатам опросов

На основе запросов и скачиваний с GitHub был создан следующий график (рисунок 2.1) [4].

Самым популярным PHP-фреймворком является Laravel. Это обусловлено широким функционалом и обширной документацией. Однако, популярность сама по себе не может являться достаточной причиной для выбора платформы реализации проекта, предварительно необходимо взвесить достоинства и недостатки остальных решений и на основе исследования выбрать наиболее подходящий вариант.

Для начала, опишем выбранные фреймворки.

Laravel – бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (англ. Model View Controller – модель-представление-контроллер). Laravel выпущен под лицензией MIT.

Исходный код проекта размещается на GitHub. В результате опроса sitepoint.com в декабре 2013 года о самых популярных PHP-фреймворках Laravel занял место самого многообещающего проекта на 2014 год.

Одной из лучших функций Laravel является его способность обрабатывать структуры noSQL, такие как MongoDB или Redis. Начать работу с Laravel также легко благодаря его обширной документации, популярности и Laravel Udemy, содержащим популярные видео и учебные пособия призваны помочь разработчикам в Laravel.

К минусам Laravel можно отнести достаточно высокий порог вхождения с обширной документацией. Так же, иногда, после выхода нового обновления, появляются баги, способные при определенных обстоятельствах сильно усложнить разработку.

Phalcon — PHP фреймворк с открытым исходным кодом, написанный на Си. В данный момент поддерживается версия переписанная на Zephir.

Основывается на идеях MVC. Разрабатывается компанией Phalcon Team. Фреймворк Phalcon PHP распространяется по лицензии BSD с учетом «New BSD License».

Phalcon предлагает разработчикам инструменты для хранения данных, такие как собственный диалект SQL: PHQL, а также Object Document Mapping для MongoDB. Другие функции включают в себя механизмы шаблонов, конструкторы форм, простоту создания приложений с поддержкой международных языков и многое другое. Phalcon идеально подходит как для создания высокопроизводительных API-интерфейсов REST, так и для полноценных веб-приложений.

Symfony — свободный фреймворк, написанный на PHP, который использует паттерн Model-View-Controller.

Symfony предлагает быструю разработку и управление веб-приложениями, позволяет легко решать рутинные задачи веб-программиста. Работает только с PHP 5 и выше. Имеет поддержку множества баз данных (MySQL, PostgreSQL, SQLite или любая другая PDO-совместимая СУБД). Информация о реляционной базе данных в проекте должна быть связана с объектной моделью. Это можно сделать при помощи ORM инструмента. Symfony поставляется с двумя из них: Propel и Doctrine.

Symfony имеет репутацию очень стабильного, высокопроизводительного и хорошо документированного фреймворка. Symfony поддерживается французскими SensioLabs и был разработан ими и его сообществом как очень функциональная структура/

Symfony используется многими известными компаниями, такими как BBC и проекты с открытым исходным кодом, такими как Drupal и eZpublish. Его документация обширна, и сообщество столь же обширно. создание как высокопроизводительных API REST, так и полноценных веб-приложений.

CodeIgniter - это идеальная среда для быстрой разработки приложений. Это легковесная инфраструктура с небольшим размером, которую можно

установить, просто загрузив ее непосредственно на хостинг без специальных команд или установки дополнительного программного обеспечения.

Создание полноценных веб-приложений - это легкая задача с небольшой кривой обучения и многочисленными библиотеками. Документация CodeIgniter обширна с обширным и полезным сообществом. Code Igniter поддерживается также академической структурой: Технологическим институтом Британской Колумбии, который поможет обеспечить его дальнейшее развитие и рост.

Fat-Free - это модульный PHP-фреймворк с множеством пакетов, которые ставят его между настоящими микро-фреймворками и полноценными PHP-фреймворками, такими как Laravel. Разработчики любят называть его F3. Fat-Free поставляется с множеством пакетов для модульного тестирования, обработки изображений, проверки данных, Open ID и многого другого. Одной из ключевых его особенностей является объем – фреймворк весит всего 63 Кб, в отличие от более тяжеловесных аналогов [5].

Fat-free обладает встроенной поддержкой баз данных SQL и NoSQL и делает разработку многоязычных веб-приложений очень простой. Следуя минималистичному подходу, он стремится избегать добавления кода и структур, которые не являются строго необходимыми, в то же время сосредотачиваясь на более общих функциях. Одним из самых больших преимуществ Fat-free является то, что он очень легкий. Хотя он не подходит для массовых проектов, он является одним из лучших вариантов для малого и среднего масштаба.

F3 поддерживает как SQL, так и NoSQL базы данных без необходимости дополнительных модификаций: в списке поддерживаемых БД имеются MySQL, SQLite, MSSQL/Sybase, PostgreSQL и MongoDB.

Так же в F3 встроена защита от спама и DoS-атак с использованием проверок DNSBL. DNSBL - это списки хостов, хранимые с использованием системы архитектуры DNS. Почтовый сервер обращается к DNSBL и проверяет в нём наличие IP-адреса клиента, с которого он принимает сообщение. При положительном ответе считается, что происходит попытка приёма спам-сообщения. Серверу отправителя сообщается ошибка 5xx (неустранимая ошибка)

и сообщение не принимается. Почтовый сервер отправителя создаёт «отказную квитанцию» отправителю о недоставке почты.

Так как у каждого из приведенных фреймворков имеется свой набор плюсов и минусов, необходимо провести более детальный сравнительный анализ (таблица 2.1).

Таблица 2.1 – Сравнение фреймворков

Название фреймворка	Плюсы	Минусы
Laravel	Организация файлов и кода. Быстрая разработка приложений. Удобное тестирование. Способность перегружать динамические методы. Очень хорошее шифрование.	Баги после обновлений. Осуществляет много запросов к БД. Вес.
Phalcon	Скорость работы. Авто-загрузка. Уникальность структуры как C-расширения. Продвинутые встроенные средства безопасности. Глубокая документация. Дружелюбен к разработчику.	Не полностью open source. Не решенные баги.
Symphony	Благодаря кэшированию кода высокая производительность. Стабильность. Хорошая документация и поддержка.	Долгое обучение. Не поддерживает MVC.
CodeIgniter	Очень дружелюбен к разработчику. Не требует особых зависимостей. Поддержка множества баз данны, включая MSQL. Более производителен чем большая часть других фреймворков. Хорошая документация и поддержка.	Нет пространства имен. Неудобное тестирование. Небольшое количество встроенных библиотек.
Fat Free	Легковесный. Легко освоить. Очень хорошо оптимизирован под маршрутизацию URL. Хорош для многоязычных приложений. Встроенная поддержка SQL и No SQL.	Нет новых опций по сравнению с остальными

Итак, по результатам сравнения, мы можем сделать соответствующие выводы.

Laravel, несмотря на объем своего функционала, имеет два существенных недостатка – объем библиотек и большое количество запросов к БД. Поскольку, в АИС будет так же задействован второй сервер, который так же будет осуществлять запросы к БД, это может вызвать увеличенную нагрузку на аппаратную часть сервера.

Phalcon недостаточно надежен для выбранной системы, Symphony имеет слишком большой порог вхождения, а CodeIgniter недостаточно удобен в тестировании.

FatFree обладает достаточным для данного приложения функционалом, при этом его объем является крайне незначительным, а простота в освоении и большое количество документации позволит быстро и качественно осуществить разработку.

2.1.2 Описание данных. Проектирование и разработка базы данных

В данной работе также рассматривается проектирование и разработка базы данных. База должна включать в себя набор различных данных, описывающих водителей, транспортные средства, маршруты и т.д.

Принято выделять следующие основные этапы, на которые разбивается процесс проектирования базы данных информационной системы:

1. Концептуальное проектирование - сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия:

– обследование предметной области, изучение ее информационной структуры;

– выявление всех фрагментов, каждый из которых характеризуется пользовательским представлением, информационными объектами и связями между ними, процессами над информационными объектами;

– моделирование и интеграция всех представлений.

2. Логическое проектирование - преобразование требований к данным в структуры данных. На выходе получаем СУБД-ориентированную структуру базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

3. Физическое проектирование - определение особенностей хранения данных, методов доступа и т.д. [7].

В данном случае описывается база для АИС, автоматизирующей транспортные перевозки. Можно выделить следующие сущности:

– Перевозчики;

– Маршруты;

– Транспортные средства.

Поскольку система в будущем может значительно расширяться и получать заказы от различных фирм и организаций, разные перевозчики могут работать на разные организации. Это тоже следует учитывать при проектировке.

Маршруты в общем виде состоят из набора контрольные точек, которые следует редактировать. Так же следует учитывать возможность разных водителей работать в разных геозонах. Обновленный список сущностей будет выглядеть следующим образом:

– Перевозчики;

– Маршруты;

– Транспортные средства;

– Геозоны;

– Контрольные точки;

– Организации.

У каждой из этих сущностей существует свой набор полей и характеристик, которые следует учитывать при построении базы. Рассмотрим их подробнее.

Автоперевозчики должны включать в себя следующее:

– id - serial integer;

– name - Наименование автоперевозчика, varchar (NOT NULL);

– organization_id - id организации, integer (NOT NULL);

– manager - ФИО руководителя, varchar (допускается NULL);

– priority - Приоритетность в единой региональной системе, integer (допускается NULL);

– group_lease - Группа «Аренда», varchar (допускается NULL);

– legal_address - Юридический адрес, varchar (допускается NULL);

– actual_address - Фактический адрес, varchar (допускается NULL);

– phone - Телефон, varchar (допускается NULL);

– email - Эл. почта, varchar (допускается NULL);

– accounting_start_date - datetime (допускается NULL);

– accounting_end_date - datetime (допускается NULL).

Маршруты включают следующее:

– id - serial integer;

– name - Название маршрута, varchar;

– transport_type_id - тип транспорта (Все типы транспорта можно получить командой transport_types);

– color - цвет маршрута;

– operational_speed - средняя скорость транспорта;

– geozone_id - идентификатор геозоны.

– Типы транспорта представлены следующими данными:

– id - serial integer;

– name - Название типа транспорта, varchar;

– short_name - Короткое имя, varchar.

По такому же принципу строятся и дальнейшие таблицы.

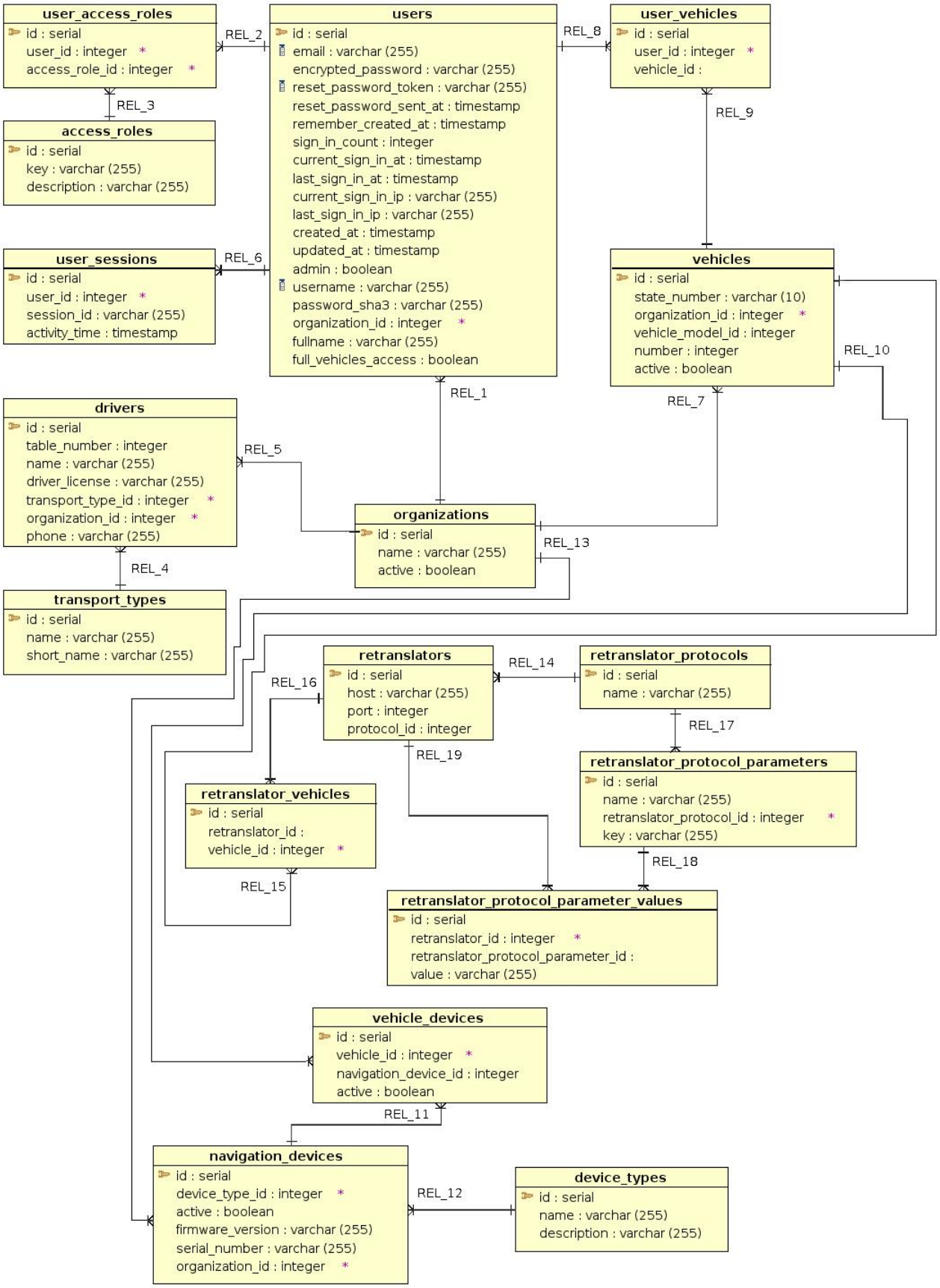

Рисунок 2.2 – Схема данных

Общая схема данных базы представлена на рисунке 2.2.

В качестве платформы для разработки БД был выбран PostgreSQL.

PostgreSQL— свободно распространяемая объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2011 [8].

Сильными сторонами PostgreSQL считаются:

– высокопроизводительные и надёжные механизмы транзакций и репликации;

– расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl, а также имеется поддержка загрузки модулей расширения на языке C;

– наследование;

– возможность индексирования геометрических объектов и наличие базирующегося на ней расширения PostGIS;

– встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;

– расширяемость. Расширяемость PostgreSQL означает, что пользователь может настраивать систему путем определения новых функций, агрегатов, типов,языков, индексов и операторов. Объектно-ориентированность PostgreSQL позволяет перенести логику приложения на уровень базы данных, что сильно упрощает разработку клиентов, так как вся бизнес логика находится в базе данных. Функции в PostgreSQL однозначно определяются названием, количеством и типами аргументов.

Функциями в PoostgreSQL называются блоки кода, исполняемые на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием одного из следующих языков:

– Встроенный процедурный язык PL/pgSQL, во многом аналогичный языку PL/SQL, используемому в СУБД Oracle;

- Скриптовые языки;
- Классические языки — C, C++,Java (через модуль PL/Java);
- Статистический язык R (через модуль PL/R).

PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса [16].

Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя.

Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие. С девятой версии возможно написание автономных блоков, которые позволяют выполнять код на процедурных языках без написания функций, непосредственно в клиенте.

PostgreSQL поддерживает большой набор встроенных типов данных:

– Целые;
– С фиксированной точкой;
– С плавающей точкой;
– Денежный тип (отличается специальным форматом вывода, а в остальном аналогичен числам с фиксированной точкой с двумя знаками после запятой);
– Символьные типы произвольной длины;
– Двоичные типы (включая BLOB);
– Типы «дата/время» (полностью поддерживающие различные форматы, точность, форматы вывода, включая последние изменения в часовых поясах);
– Булев тип;
– Перечисление;
– Геометрические примитивы;
– IP и IPv6-адреса;
– CIDR-формат;

- MAC-адрес;
- UUID-идентификатор;
- XML-данные;
- Массивы;
- JSON;
- Идентификаторы объектов БД;
- Псевдотипы.

2.2 Описание процессов извлечения и обработки данных. Разработка алгоритмов обработки данных

В модели каскадного цикла разработки программного обеспечения принято выделять следующие этапы:

1. Подготовка;
2. Проектирование;
3. Реализация;
4. Тестирование.

В данном разделе подробным образом будет рассмотрено проектирование и моделирование.

Моделирование является основой успешного функционирования любого программного продукта. Корректная и полная модель позволит получить требуемую систему.

Существуют разные средства и нотации для моделирования. Одной из них является IDEF0 – методология моделирования функций, позволяющая моделировать производственные функции, а так же используемая для анализа и разработки программного обеспечения (рисунок 2.3).

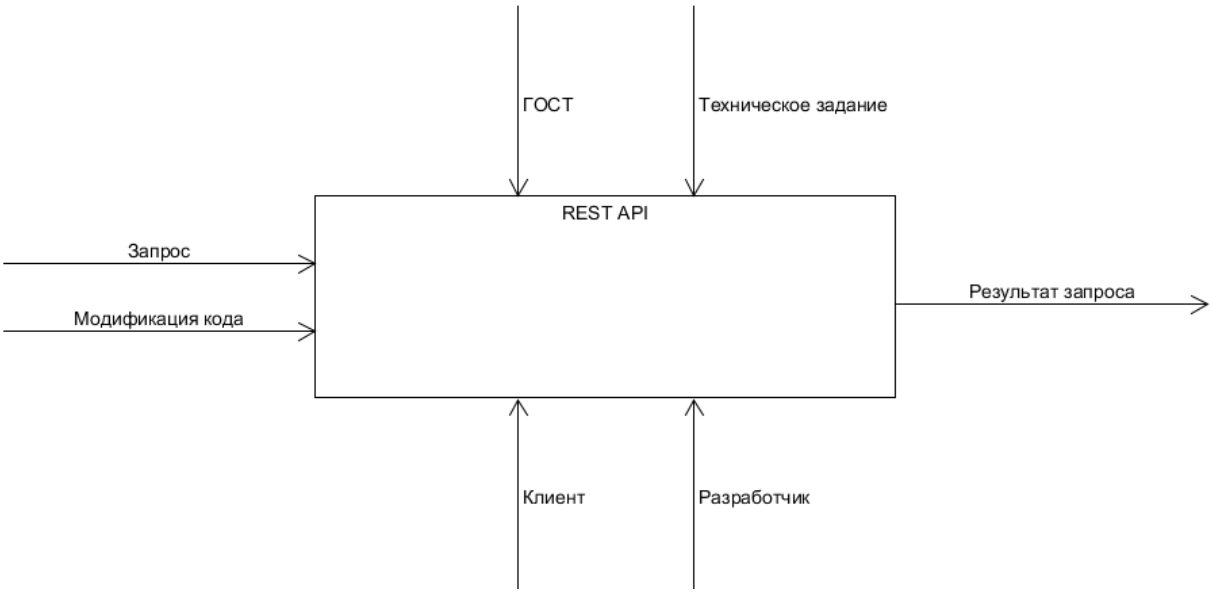

Рисунок 2.3 – Общая диаграмма

На рисунке 2.3 изображена функциональная модель верхнего уровня. Функциональная модель IDEF0 в общем виде представляет собой набор блоков, каждый из которых представляет собой «черный ящик» со входами и выходами, управлением и механизмами, которые в последствии декомпозируются до необходимого уровня. Функции соединяются между собой при помощи стрелок и описаний функциональных блоков. При этом каждый вид стрелки или активности имеет собственное значение. Данная модель позволяет описать все основные виды процессов, как административные, так и организационные.

Стрелки могут быть следующих видов:

1. Входящие – вводные, которые ставят определенную задачу;
2. Исходящие – выводящие результат деятельности;
3. Управляющие (сверху вниз) – механизмы управления (положения, инструкции и пр.);
4. Механизмы (снизу-вверх) – что используется для того, чтобы произвести необходимую работу.

В роли механизмов в рассматриваемом примере выступают клиент-приложение и разработчик, в роли управляющих механизмов выполняют ГОСТ и ТЗ, входящими стрелками представлены запросы клиента и задачи модификации кода, а исходящим результатом деятельности является результат запроса.

Таким образом, была составлена общая модель функционирования приложения. В дальнейшем необходимо провести декомпозицию модели и рассмотреть реализацию поставленных задач более подробно. Для этого используется функциональная модель второго уровня, расширяющая и дополняющая модель первого (рисунок 2.4).

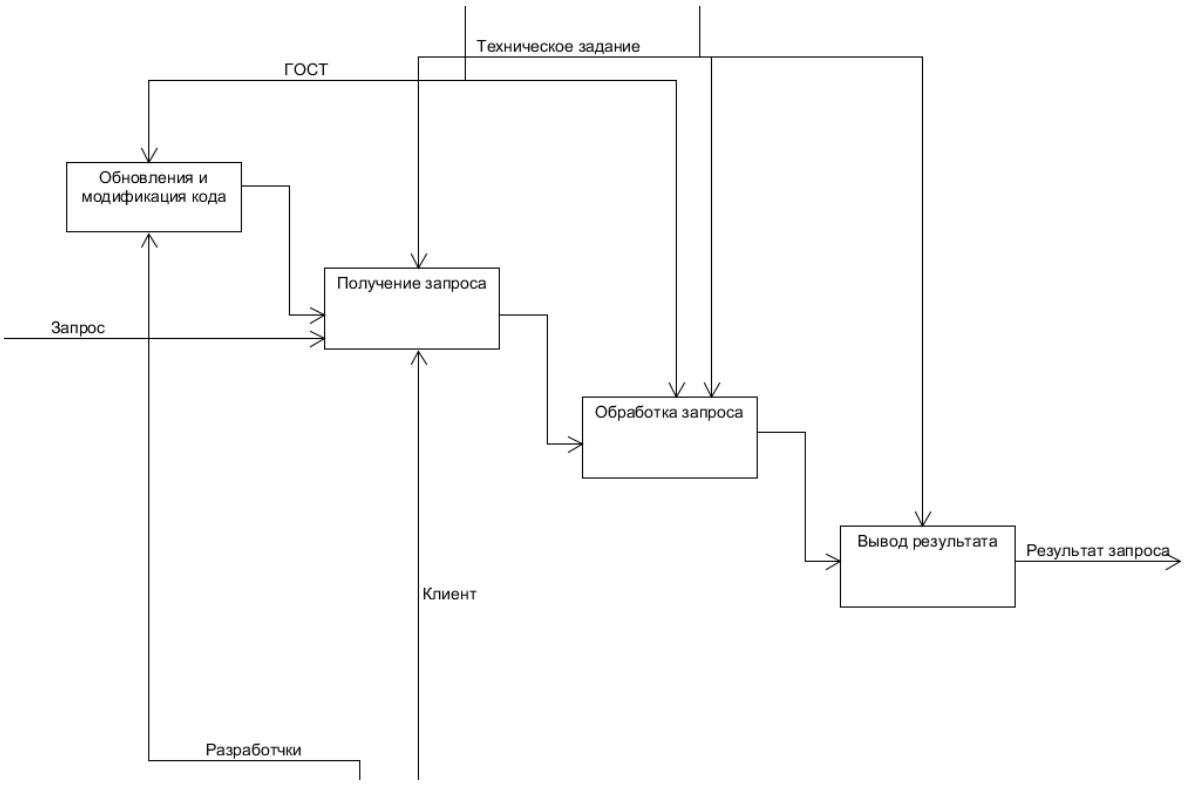

Рисунок 2.4 – IDEF0 модель второго уровня

Следующая рассматриваемая диаграмма не является частью стандарта IDEF0, но все равно является очень полезной частью проектирования ПО и называется деревом узлов. Диаграмма дерева узлов показывает иерархию работ в модели, позволяя рассмотреть всю модель целиком, но при этом не показывая взаимосвязи между работами. Процесс создания модели работ является итерационным, следовательно, работы могут менять своё положение в дереве узлов (рисунок 2.5).

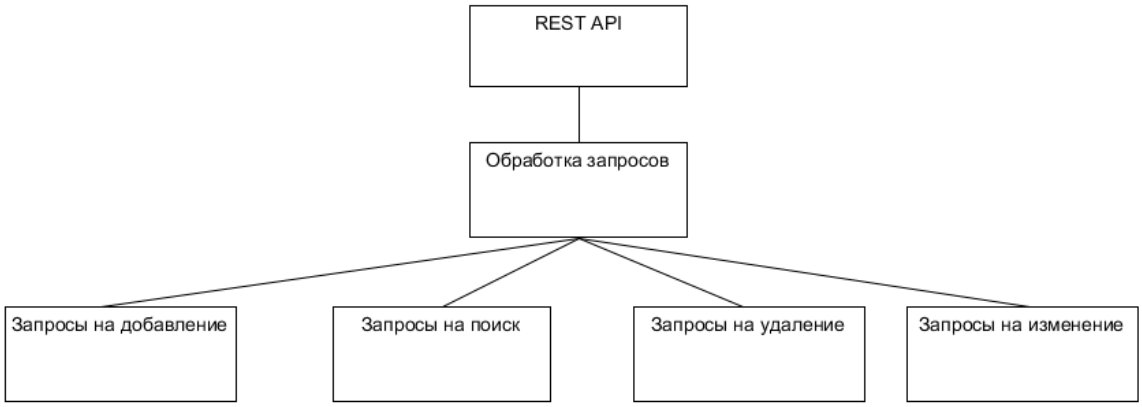

Рисунок 2.5 – Дерево узлов

Очень сильным инструментом при моделировании ПО является UML (Unified Modeling Language) – язык графического описания, предназначенный для обеспечения стандартного способа визуализации проектирования системы. Виды диаграмм UML можно разделить следующим образом:

– Внешней перспективы (диаграмма вариантов использования и диаграмма последовательности);

– Структурная перспектива (диаграмма классов и диаграмма развертывания);

– Перспектива взаимодействия (диаграмма последовательности);

– Поведенческая перспектива (диаграмма состояний)[9].

Диаграмма вариантов использования является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними (рисунок 2.6).

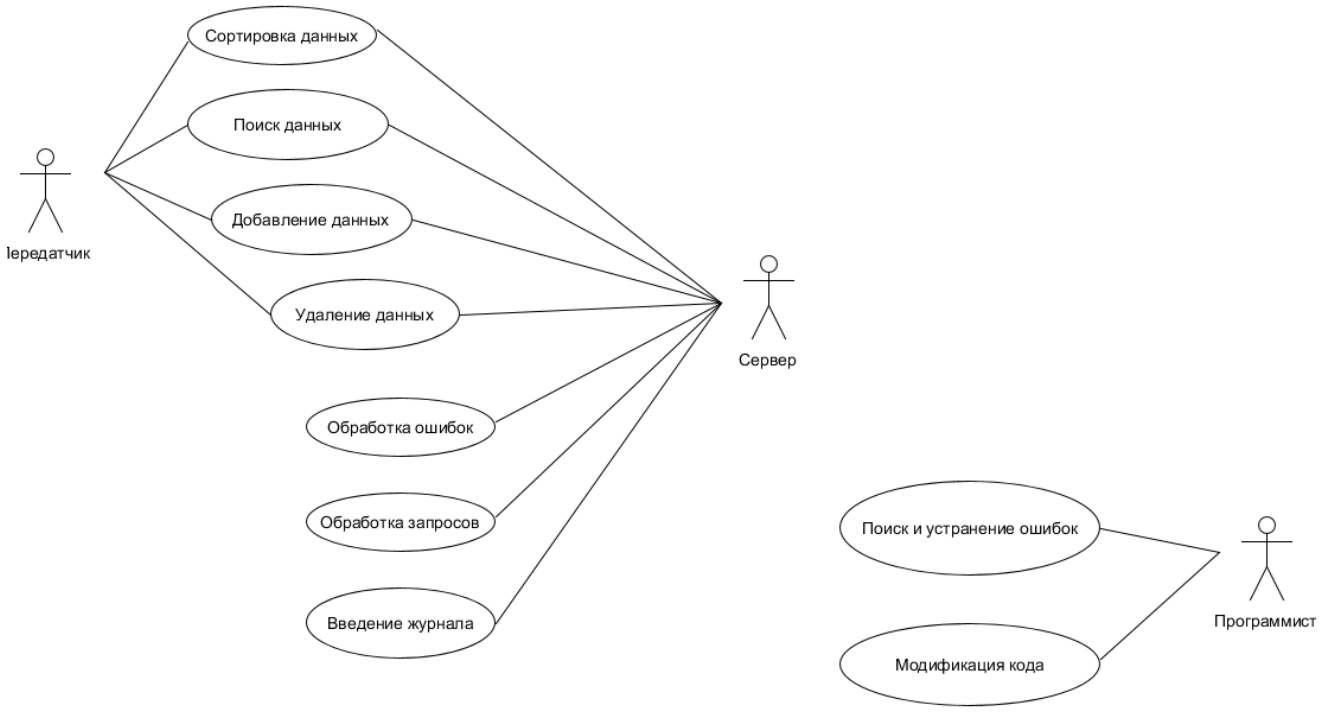

Рисунок 2.6 – Диаграмма вариантов использования

Диаграмма строится по следующему принципу: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру.

На рисунке 2.7 представлена диаграмма классов.

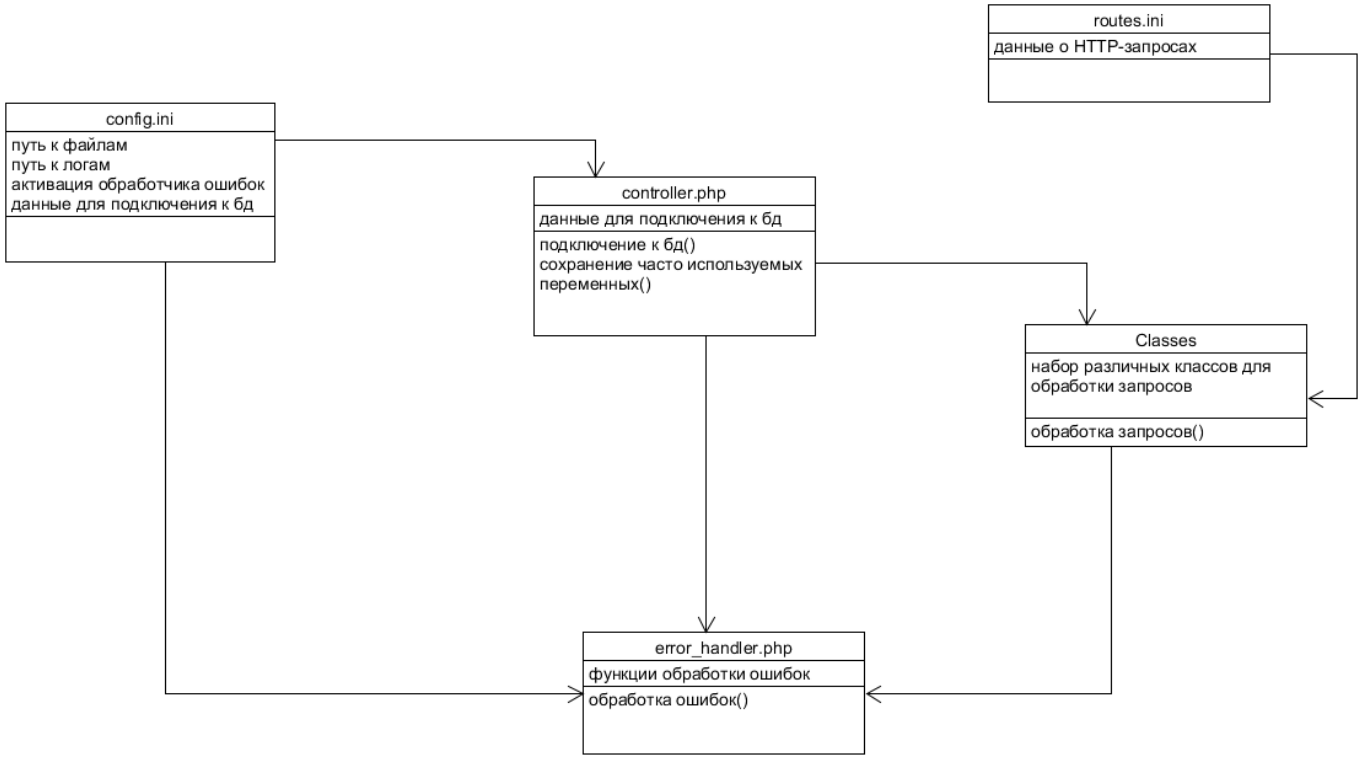

Рисунок 2.7 – Диаграмма классов

Диаграмма классов является структурной диаграммой, демонстрирующая общую структуру иерархии классов системы, их кооперацій, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Она широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

Диаграмма объектов (рисунок 2.8) показывает снимок подробного состояния системы в определенный момент времени.

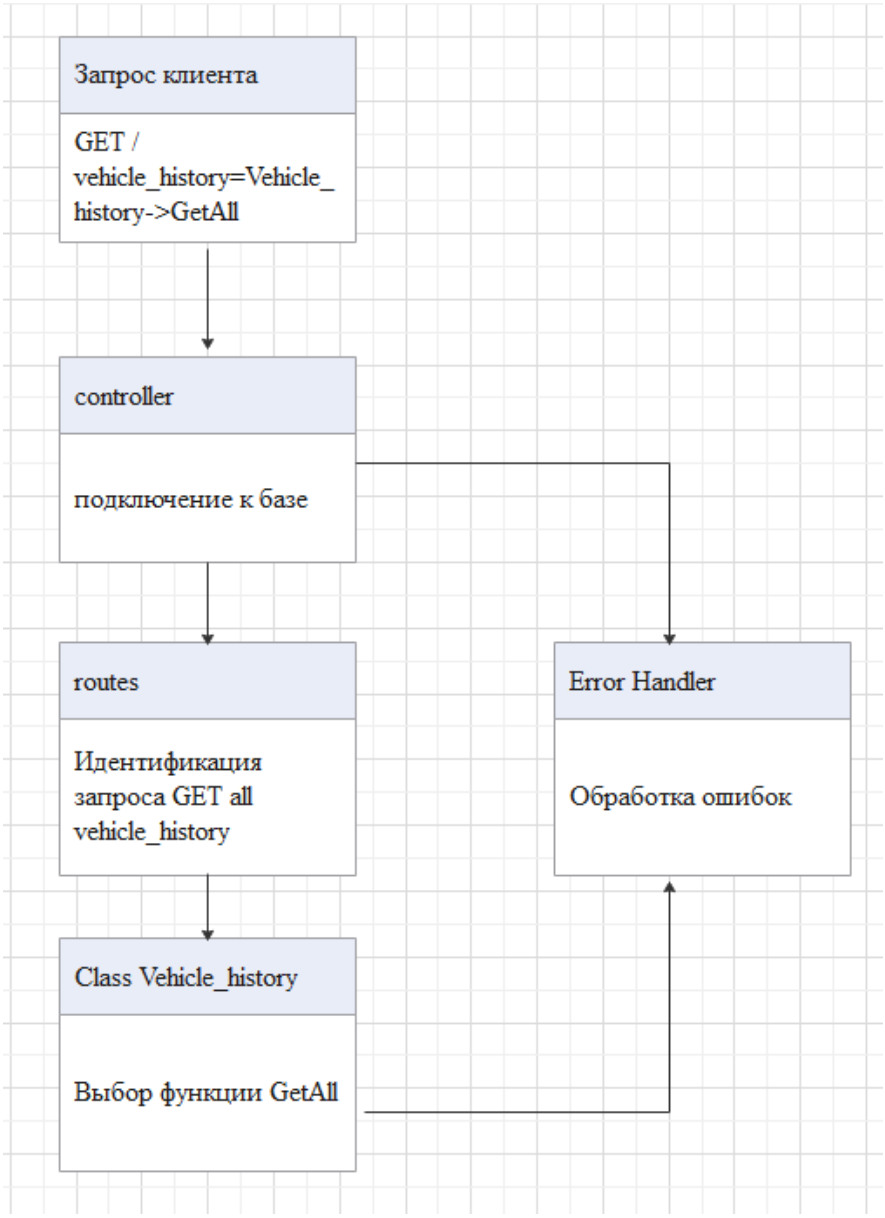

Рисунок 2.8 – Диаграмма объектов

Использование диаграмм объектов довольно ограничено, в основном демонстрацией примеров структуры данных.

На рисунке 2.9. представлена диаграмма состояний.

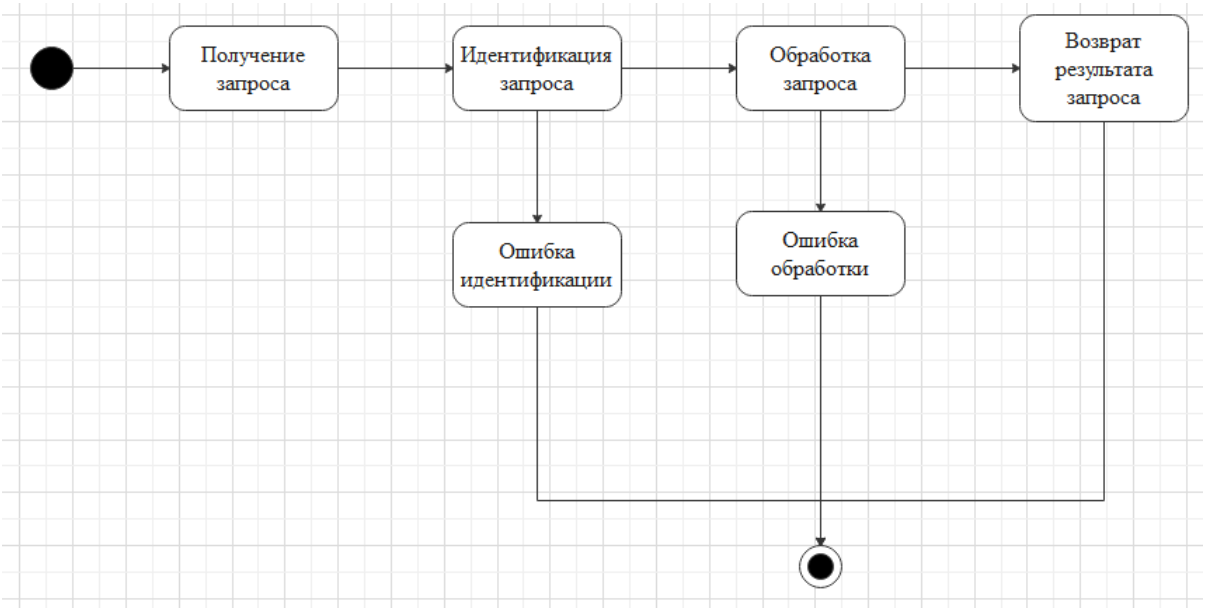

Рисунок 2.9 – Диаграмма состояний

Состояние - ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события. Состояние объекта определяется значениями некоторых его атрибутов и присутствием или отсутствием связей с другими объектами.

Диаграмма состояний показывает, как объект переходит из одного состояния в другое (рисунок 2.9). Диаграммы состояний служат для моделирования динамических аспектов системы (как и диаграммы последовательностей, кооперации, прецедентов и, как мы увидим далее, диаграммы деятельности).

Диаграмма активности – это UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний (рисунок 2.10).

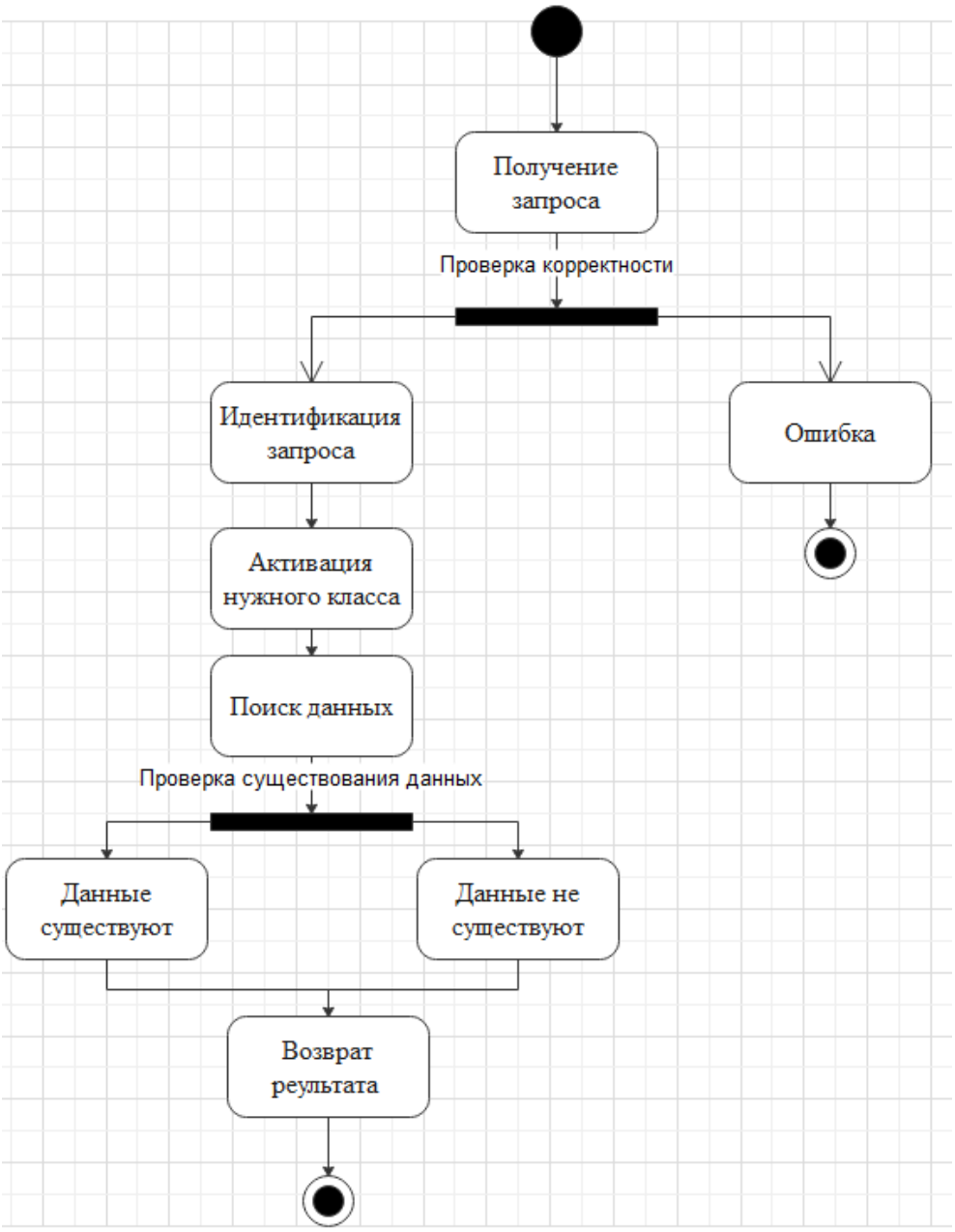

Рисунок 2.10 – Диаграмма активности

Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграмма развертывания (рисунок 2.11) предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения.

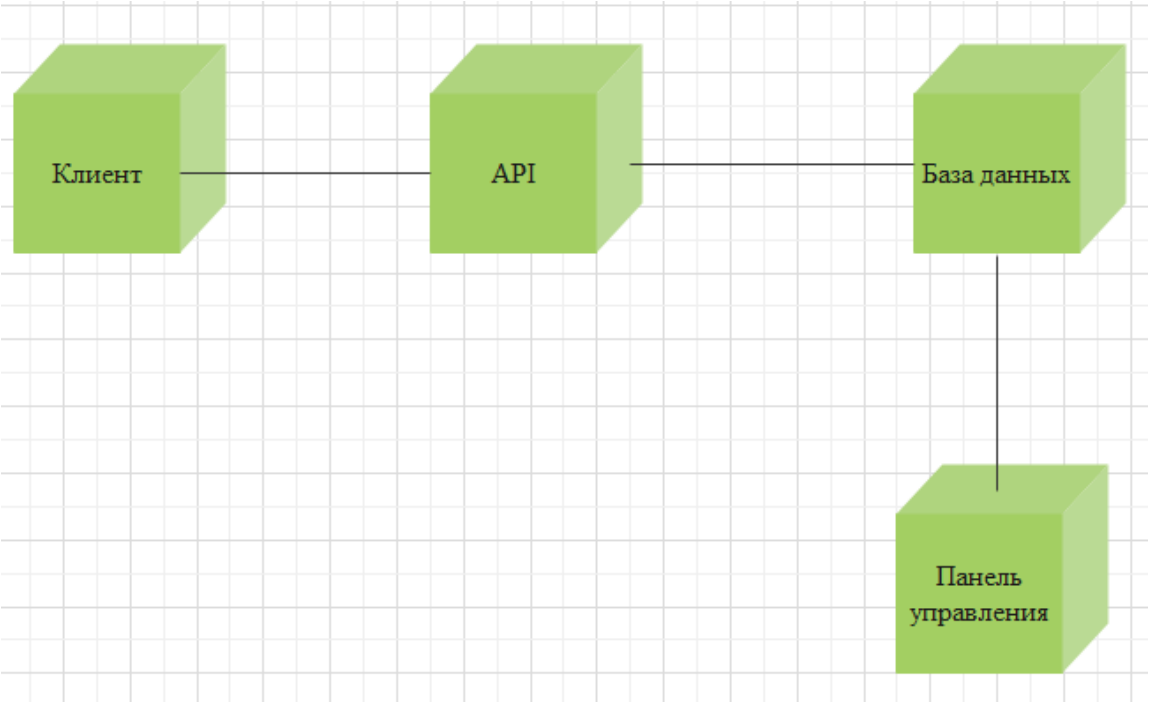

Рисунок 2.11 - Диаграмма развертывания

При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. Так, компоненты с исходными текстами программ могут присутствовать только на диаграмме компонентов. На диаграмме развертывания они не указываются.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними.

2.3 Разработка API обмена данными

Разработка проводится на языке PHP с использованием фреймворка FatFree. Главным предназначением API в данной работе является поиск, добавление, удаление и обновление записей данных в зависимости от запроса. Все запросы приходят с web-клиента, который может быть наделен различными правами доступа и обладать различным функционалом, в зависимости от пользователя и реализуемой задачи.

Принцип работы системы следующий: клиент посредством протокола HTTP передает запрос на сервер, где тот обрабатывается и, в зависимости от результатов, результат передается обратно через файл JSON.

JSON (JavaScript Object Notation) – это текстовый формат представления данных в нотации объекта JavaScript. Предназначен JSON, также как и некоторые другие форматы такие как XML и YAML, для обмена данными.

На рисунке 2.12 представлена общая схема взаимодействия REST API с клиентом и базой.

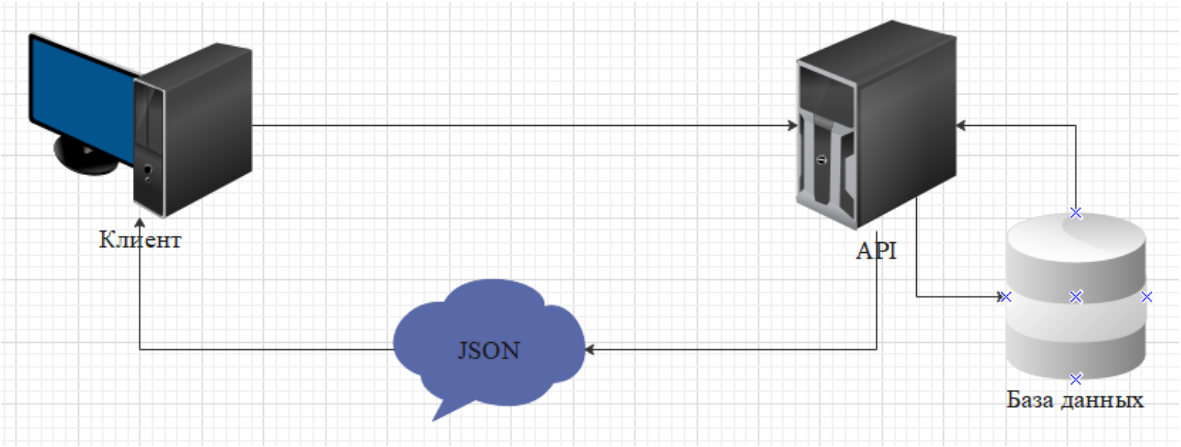

Рисунок 2.12 – Общая схема работы

Одним из преимуществ разработки на уже существующем фреймворке по сравнению с разработкой с нуля является автоматизация некоторых процессов. Часть файлов генерируются автоматически, что значительно ускоряет и облегчает процесс разработки. Рассмотрим файлы более подробно.

Config.ini представляет собой конфигурационный файл для API, в нем расположена информация о подключении к базе данных, о расположении файлов приложения, дата и время, а так же время на выделенную сессию (рисунок 2.13).

config.ini 747B

```
[globals]
; Where the framework autoloader will look for app files
AUTOLOAD="app/;app/classes/"
; Remove next line (if you ever plan to put this app in production)
DEBUG=2
; Where errors are logged
LOGS=tmp/
; Our custom error handler
ONERROR="Error_handler->error"
; Where the framework will look for templates and related HTML-support files
UI=ui/

; pgsql DSN
db="pgsql:host=localhost;port=5432;dbname=monitoring_2"
; db username
db_username="postgres"
; db password
db_password="postgres"
; User ID and password for accessing the back-end
user_id=admin
; Password: crypt('secret')
password="$1$o94.Rc0.$2GSt4JJsPx63u9UVpl0r//"
; Number of hours before session expires
expiry=24
; How timestamps look like on the pages
time_format="d M Y h:ia"
```

Рисунок 2.13 – config.ini

Файл controller.php обрабатывает HTTP-запрос и инициирует необходимый класс. На этом вопросе следует остановится подробнее.

Каждый запрос на сервер приходит в формате HTTP. В общем виде, запрос представляется следующим образом:

<HTTP-запрос>: url ссылка/ переменная.

Рассмотрим запросы подробнее:

– GET – позволяет получать один или несколько записей из базы;

– POST – позволяет добавлять записи;

– PUT – позволяет обновлять записи базы;

– DELETE – осуществляет удаление записи из базы.

Таким образом, для того, чтобы, например, запросить данные всех маршрутов, клиент должен отправить запрос:

GET http://194.58.104.20/routes=Routes.

У запроса GET существует определенная дифференциация – запрос GET All, позволяющий получить все данные из какой-либо таблицы, либо запрос GET One, позволяющий найти информацию по выбранному ID. Выше был приведен пример запроса GET All, запрос GET One будет выглядеть следующим образом:

GET http://194.58.104.20/routes/{id}.

Для разделения и корректной идентификации различных запросов требуется список требуемых функций, на основе которых контроллер будет загружать необходимый класс. Данную роль выполняет файл routes.ini (рисунок 14):

routes.ini 11KB

```
[routes]

; vehicle_history
GET /vehicle_history=Vehicle_history->GetAll
GET /vehicle_history/@id=Vehicle_history->GetOne

; routes
GET /routes=Routes->GetAll
GET /routes/@id=Routes->GetOne
POST /routes=Routes->Add
PUT /routes/@id=Routes->Update
DELETE /routes/@id=Routes->Delete

; vehicles
GET /vehicles=Vehicles->GetAll
GET /vehicles/@id=Vehicles->GetOne
POST /vehicles=Vehicles->Add
PUT /vehicles/@id=Vehicles->Update
DELETE /vehicles/@id=Vehicles->Delete

; vehicle last locations
GET /last_vehicle_locations=Vehicle_last_locations->GetAll
GET /last_vehicle_locations/@id=Vehicle_last_locations->GetOne
POST /last_vehicle_locations=Vehicle_last_locations->Add
PUT /last_vehicle_locations/@id=Vehicle_last_locations->Update
DELETE /last_vehicle_locations/@id=Vehicle_last_locations->Delete

; transport types
GET /transport_types=Transport_types->GetAll
GET /transport_types/@id=Transport_types->GetOne
POST /transport_types=Transport_types->Add
PUT /transport_types/@id=Transport_types->Update
DELETE /transport_types/@id=Transport_types->Delete

```

Рисунок 2.14 – Файл routes.ini

По мере необходимости, можно добавлять новые функции, что значительно облегчит осуществление масштабируемости приложения.

При реализации взаимодействия между двумя независимыми блоками, необходимо предусмотреть и настроить обработку возможных ошибок. При реализации архитектуры REST в этом поможет HTTP-статус коды. Рассмотрим это понятие более подробно.

Когда клиент отправляет запрос на HTTP-сервер - и сервер успешно принимает запрос - сервер должен уведомить клиента, был ли запрос успешно обработан или нет. HTTP выполняет это с пятью категориями кодов состояния:

– 100-й уровень (информационный) - сервер подтверждает запрос;

– 200-й уровень (успех) - сервер выполнил запрос, как и ожидалось;

– 300-й уровень (перенаправление) - клиенту необходимо выполнить дальнейшие действия для выполнения запроса;

– 400-й уровень (ошибка клиента) - клиент отправил неверный запрос;

– 500-й уровень (ошибка сервера) - серверу не удалось выполнить действительный запрос из-за ошибки сервера.

На основании кода ответа клиент может догадаться о результате определенного запроса. Самый простой способ обработки ошибок - ответить соответствующим кодом состояния.

Некоторые общие коды ответов включают в себя:

– 400 Bad Request - клиент отправил неверный запрос - например, отсутствует нужное тело запроса или параметр;

– 401 Unauthorized - Клиент не смог пройти аутентификацию на сервере;

– 403 Forbidden - клиент аутентифицирован, но не имеет доступа к запрошенному ресурсу;

– 404 Not Found - запрошенный ресурс не существует;

– 412 Precondition failed - одно или несколько условий в полях заголовка запроса оцениваются как ложные;

– 500 Internal Server Error - На сервере произошла общая ошибка;

– 503 Service unavailable - запрошенный сервис недоступен.

Как правило, клиент не оповещается об ошибках с кодом 500. 500-ые ошибки указывают на то, что на сервере возникли некоторые проблемы, такие как непредвиденное исключение в нашей службе REST при обработке запроса. Поэтому эта внутренняя ошибка не должна беспокоить клиента.

В данном случае, обработкой ошибок занимается файл error_handler.php (рисунок 2.15):

```php
<?php

//! Front-end processor
class Error_handler{

	function error($f3) {
		while (ob_get_level())
			ob_end_clean();

		header('Content-Type: application/json');

		echo json_encode(
			array(  "Code" => $f3->get('ERROR.code'),
				"Level" => $f3->get('ERROR.level'),
				"Status" => $f3->get('ERROR.status'),
				"Text" => $f3->get('ERROR.text'),
				"Trace" => $f3->get('ERROR.trace')
			), JSON_UNESCAPED_UNICODE);

		$log=new Log('error.log');
		$log->write($f3->get('ERROR.text'));
		foreach ($f3->get('ERROR.trace') as $frame)
			if (isset($frame['file'])) {
				// Parse each backtrace stack frame
				$line='';
				$addr=$f3->fixslashes($frame['file']).':'.$frame['line'];
				if (isset($frame['class']))
					$line.=$frame['class'].$frame['type'];
				if (isset($frame['function'])) {
					$line.=$frame['function'];
					if (!preg_match('/{.+}/',$frame['function'])) {
						$line.='(';
						if (isset($frame['args']) && $frame['args'])
							$line.=$f3->csv($frame['args']);
						$line.=')';
					}
				}
				// Write to custom log
				$log->write($addr.' '.$line);
			}
```

Рисунок 2.15 – Файл error_handler.php

С помощью функции json_encode, скрипт отправляет массив данных в файл JSON. Массив включает в себя код ошибки, её уровень, а также текстовое пояснение. Важным представляется ведение журнала ошибок, поэтому информация о них заносится в лог.

Файл controller.php представлен на рисунке 2.16.

```php
<?php

//! Base controller
class Controller {

	protected
		$db;

	//! HTTP route pre-processor
	function beforeroute($f3) {
		$db=$this->db;
	}

	//! HTTP route post-processor
	function afterroute() {

	}

	//! Instantiate class
	function __construct() {
		$f3=Base::instance();
		// Connect to the database
		$db=new DB\SQL($f3->get('db'), $f3->get('db_username'), $f3->get('db_password'));
		// Use database-managed sessions
		new DB\SQL\Session($db);
		// Save frequently used variables
		$this->db=$db;
	}
```

Рисунок 2.16 – Файл controller.php

Его назначением является инициация сессии подключения к базе данных. Его методы будут унаследованы другими классами.

Выше перечисленные файлы помогают осуществить маршрутизацию и управление запросами клиента. Однако, сами запросы они не выполняют и не осуществляют работу с данными в базе. Для этого существуют файлы, классифицируемые как «класс» (class). Рассмотрим их более подробно.

Из-за того, что приложение осуществляется по модульной архитектуре, располагать все функции в одном файле (или не оптимально распределять их по нескольким) представляется контрпродуктивным. Решением является распределение всех функций по отдельным файлам в зависимости от того, с каким именно сегментом базы данных осуществляется работа. Таким образом, каждый отдельный файл представляет собой набор функций для работы с той или иной таблицей.

В качестве примера можно рассмотреть файл carriers.php, работающий с данными автоперевозчиков.

Таблица автоперевозчиков включает в себя следующие поля:

– id - serial integer;

– name - Наименование автоперевозчика, varchar (NOT NULL);

– organization_id - id организации, integer (NOT NULL);

– manager - ФИО руководителя, varchar (допускается NULL);

– priority - Приоритетность в единой региональной системе, integer (допускается NULL);

– group_lease - Группа «Аренда», varchar (допускается NULL);

– legal_address - Юридический адрес, varchar (допускается NULL);

– actual_address - Фактический адрес, varchar (допускается NULL);

– phone - Телефон, varchar (допускается NULL);

– email - Эл. почта, varchar (допускается NULL);

– accounting_start_date - datetime (допускается NULL);

– accounting_end_date - datetime (допускается NULL);

Методами данного класса, являются функции, осуществляющие добавление, поиск, удаление и обновление данных в таблице.

Список команд:

1. <GET> : http://194.58.104.20/carriers
2. <GET> : http://194.58.104.20/carriers/{id}
3. <POST> : http://194.58.104.20/carriers
4. <PUT> : http://194.58.104.20/carriers/{id}
5. <DELETE> : http://194.58.104.20/carriers/{id}

```php
//! Front-end processor
class Carriers extends Controller {

	//! GET all method
	function GetAll($f3) {
		$db = $this->db;

		$res = $db->exec('select id, name, organization_id as organization, manager, priority, group_lease, legal_a

		foreach($res as &$value)
		{
			$res2 = $db->exec('select * from directory.organizations_2 WHERE id = ?', $value['organization']);
			$value['organization'] = $res2[0];
		}

		header('Content-Type: application/json');
		$f3->status(200);

		echo json_encode($res, JSON_UNESCAPED_UNICODE);
	}
```

Рисунок 2.17 – Метод GET all класса Carriers

Рассмотрим метод GET all (рисунок 2.17) более подробно. После инициации подключения к базе данных, метод подключения к которой он унаследовал от класса Controller, осуществляется SQL запрос Select, который выбирает необходимые поля – осуществляется выбор только тех полей, которые нужны пользователю.

После этого добавляются данные из связанной таблицы и, в случае если передача прошла успешно, данные добавляются в файл JSON.

Рассмотрим выполнение запроса GET one (рисунок 18):

```php
	//! GET by id
	function GetOne($f3) {
		$db = $this->db;

		$res = $db->exec('select id, name, organization_id as organization, manager, priority, group_lease, le

		$res2 = $db->exec('select * from directory.organizations_2 WHERE id = ?', $res[0]['organization']);

		$res[0]['organization'] = $res2[0];

		if($res == null)
		{
			$f3->error(404);
		}

		header('Content-Type: application/json');
		$f3->status(200);

		echo json_encode($res[0], JSON_UNESCAPED_UNICODE);
	}
```

Рисунок 2.18 – Метод GET all класса Carriers

Запрос Select был несколько видоизменен и теперь ищет запись с конкретным ID. Но если раньше код выполнялся без ошибок практически в любом случае (программа выдавала любое доступное количество записей), то теперь, из-за того, что ID искомой записи может быть не найден был добавлен обработчик ошибок, передающий код 404 в случае, если запись не может быть найдена.

Рассмотрим метод DELETE (рисунок 2.19):

```
//! DELETE method
function Delete($f3) {

        $db = $this->db;

        $object = new DB\SQL\Mapper($db,'directory.carriers');
        $object->load(array('@id=?',$f3->get('PARAMS.id')));
        if($object->dry())
        {
                $f3->error(404);
        }
        $object->erase();
        $f3->status(204);
}
```

Рисунок 2.19 – Метод DELETE класса Carriers

Данный метод удаляет записи по переданному ID и возвращает ошибку в случае, если подобных данных уже нет.

Рассмотрим метод POST (рисунок 2.20). Данный метод вносит данные, полученные со стороны клиента. Его отличительной особенностью является то, что если другие методы добавляли данные в файл JSON для дальнейшей работы с ними на клиенте, то в данном случае процесс обратный – функция разбирает полученные данные, записанные в файл самим клиентом и добавляет их в базу.

```php
//! POST data
function Add($f3) {

	$db = $this->db;

	$body = json_decode($f3->get('BODY'), TRUE);
	if($body == null)
	{
		$f3->status(500);
		exit;
	}

	$object = new DB\SQL\Mapper($db,'directory.carriers');

	$object->name = $body['name'];
	$object->organization_id = $body['organization_id'];
	$object->manager = $body['manager'];
	$object->priority = $body['priority'];
	$object->group_lease = $body['group_lease'];
	$object->legal_address = $body['legal_address'];
	$object->actual_address = $body['actual_address'];
	$object->phone = $body['phone'];
	$object->email = $body['email'];
	$object->accounting_start_date = $body['accounting_start_date'];
	$object->accounting_end_date = $body['accounting_end_date'];
	$object->save();
	$f3->status(204);
}
```

Рисунок 2.20 – Метод POST класса Carriers

Как и в прошлом примере, метод обновления (рисунок 2.21) записей считывает пользовательские данные из файла JSON, после чего ищет необходимую запись и обновляет её поля.

```
//! PUT data
function Update($f3) {
        $db = $this->db;

        $body = json_decode($f3->get('BODY'), TRUE);
        if($body == null)
        {
                $f3->status(500);
                exit;
        }

        $object = new DB\SQL\Mapper($db,'directory.carriers');
        $object->load(array('@id=?',$f3->get('PARAMS.id')));
        if($object->dry())
        {
                $f3->error(404);
        }

        if($body['name'] != null){
                $object->name = $body['name'];
        }
        if($body['organization_id'] != null){
                $object->organization_id = $body['organization_id'];
        }
        if($body['manager'] != null){
                $object->manager = $body['manager'];
        }
        if($body['priority'] != null){
                $object->priority = $body['priority'];
        }
        if($body['group_lease'] != null){
                $object->group_lease = $body['group_lease'];
        }
        if($body['legal_address'] != null){
                $object->legal_address = $body['legal_address'];
        }
        if($body['actual_address'] != null){
                $object->actual_address = $body['actual_address'];
        }
        if($body['phone'] != null){
                $object->phone = $body['phone'];
        }
        if($body['email'] != null){
                $object->email = $body['email'];
        }
        if($body['accounting_start_date'] != null){
                $object->accounting_start_date = $body['accounting_start_date'];
        }
        if($body['accounting_end_date'] != null){
```

Рисунок 2.21 – Метод PUT класса Carriers

Данный подход распространяется и на все остальные классы, что обеспечивает возможность расширения и усложнения системы без нарушения её структуры.

К примеру, класс Drivers должен осуществлять работу с данными водителей:

```
//! GET all method
function GetAll($f3) {
        $db = $this->db;

        $res = $db->exec('select id, table_number, full_name, age, phone, driving_experience_start_date,

        foreach($res as &$value)
        {
                $res2 = $db->exec('select * from directory.carriers WHERE id = ?', $value['carrier']);
                $value['organization'] = $res2[0];
        }

        header('Content-Type: application/json');
        $f3->status(200);

        echo json_encode($res, JSON_UNESCAPED_UNICODE);
}
```

Рисунок 2.22 – Метод GET all класса Drivers

Этот метод (рисунок 2.22) реализован по схожему с остальными методами принципу и позволяет получить все записи о водителях. Для получения информации об одном водителе используется другой метод (рисунок 2.23). В большинстве случаев, из-за разностей уровня доступа разных пользователей, некоторые функции, доступные одним, будут недоступны другим пользователям – это может быть обусловлено как спецификой выполняемой пользователем работы, так и уровнем его доступа к базе.

Метод GET all класса Drivers представлен на рисунке 2.23.

Метод POST для класса Drivers – на рисунке 2.24.

Метод PUT для класса Drivers – на рисунке 2.25.

Метод DELETE для класса Drivers – на рисунке 2.26.

```
//! GET by id
function GetOne($f3) {
        $db = $this->db;

        $res = $db->exec('select id, table_number, full_name, age, phone, driving_experience_sta

        $res2 = $db->exec('select * from directory.carriers WHERE id = ?', $res[0]['carrier']);

        $res[0]['carrier'] = $res2[0];

        if($res == null)
        {
                $f3->error(404);
        }

        header('Content-Type: application/json');
        $f3->status(200);

        echo json_encode($res[0], JSON_UNESCAPED_UNICODE);
}
```

Рисунок 2.23 - Метод GET all класса Drivers

```
//! POST data
function Add($f3) {

        $db = $this->db;

        $body = json_decode($f3->get('BODY'), TRUE);
        if($body == null)
        {
                $f3->status(500);
                exit;
        }

        $object = new DB\SQL\Mapper($db,'directory.drivers_2');

        $object->full_name = $body['full_name'];
        $object->table_name = $body['table_name'];
        $object->age = $body['age'];
        $object->phone = $body['phone'];
        $object->driving_experience_start_date = $body['driving_experience_start_date'];
        $object->driving_experience = $body['driving_experience'];
        $object->driving_experience_dcat_start_date = $body['driving_experience_dcat_start_date'];
        $object->driving_experience_dcat = $body['driving_experience_dcat'];
        $object->medical_certificate_valid_until = $body['medical_certificate_valid_until'];
        $object->carrier_id = $body['carrier_id'];
        $object->accounting_start_date = $body['accounting_start_date'];
        $object->accounting_end_date = $body['accounting_end_date'];
        $object->save();
        $f3->status(204);
}
```

Рисунок 2.24 - Метод POST для класса Drivers

```php
//! PUT data
function Update($f3) {
	$db = $this->db;

	$body = json_decode($f3->get('BODY'), TRUE);
	if($body == null)
	{
		$f3->status(500);
		exit;
	}

	$object = new DB\SQL\Mapper($db,'directory.drivers_2');
	$object->load(array('@id=?',$f3->get('PARAMS.id')));
	if($object->dry())
	{
		$f3->error(404);
	}

	if($body['full_name'] != null){
		$object->full_name = $body['full_name'];
	}
	if($body['table_name'] != null){
		$object->table_name = $body['table_name'];
	}
	if($body['age'] != null){
		$object->age = $body['age'];
	}
	if($body['phone'] != null){
		$object->phone = $body['phone'];
	}
	if($body['driving_experience_start_date'] != null){
		$object->driving_experience_start_date = $body['driving_experience_start_date'];
	}
	if($body['driving_experience'] != null){
		$object->driving_experience = $body['driving_experience'];
	}
	if($body['driving_experience_dcat_start_date'] != null){
		$object->driving_experience_dcat_start_date = $body['driving_experience_dcat_start_date'];
	}
	if($body['driving_experience_dcat'] != null){
		$object->driving_experience_dcat = $body['driving_experience_dcat'];
	}
	if($body['medical_certificate_valid_until'] != null){
		$object->medical_certificate_valid_until = $body['medical_certificate_valid_until'];
	}
```

Рисунок 2.25 – Метод PUT для класса Drivers

```
//! DELETE method
function Delete($f3) {

        $db = $this->db;

        $object = new DB\SQL\Mapper($db,'directory.drivers_2');
        $object->load(array('@id=?',$f3->get('PARAMS.id')));
        if($object->dry())
        {
                $f3->error(404);
        }
        $object->erase();
        $f3->status(204);
}
```

Рисунок 2.26 – Метод DELETE для класса Drivers

Также, в некоторых случаях, API позволяет исключительно один вид операций. Например, это касается модуля vehicle history, рассматривающего историю перемещения транспортных средств. Это обусловлено как защитой данных от недобросовестных пользователей, так и отсутствием практической необходимости.

Метод GET all для vehicle history представлен на рисунке 2.27.

Метод GET one для vehicle history – на рисунке 2.28.

```php
//! GET all method
function GetAll($f3) {
	$db = $this->db;

	$body = json_decode($f3->get('BODY'), TRUE);
	if($body == null)
	{
		$f3->status(500);
		exit;
	}

	$from = new DateTime($body['from']);
	$to = new DateTime($body['to']);

	$res = $db->exec("SELECT id, latitude, longitude, direction, speed, altitude,
				extract(epoch from measuring_time::timestamp without time zone)::integer
				as timestamp
				FROM history.monitoring_records
				WHERE vehicle_id = ?
				AND measuring_time >=
				CAST (to_timestamp(?) at time zone 'utc' AS timestamp without time zone)
				AND measuring_time <=
				CAST (to_timestamp(?) at time zone 'utc' AS timestamp without time zone)
				ORDER BY measuring_time ASC", array($body['id'], $from->getTimestamp(), $to->getTimestam

	header('Content-Type: application/json');
	$f3->status(200);

	echo json_encode($res, JSON_UNESCAPED_UNICODE);
}
```

Рисунок 2.27 – Метод GET all для vehicle history

```php
//! GET by id
function GetOne($f3) {
	$db = $this->db;

	$body = json_decode($f3->get('BODY'), TRUE);
	if($body == null)
	{
		$f3->status(500);
		exit;
	}

	$from = new DateTime($body['from']);
	$to = new DateTime($body['to']);

	$res = $db->exec("SELECT id, latitude, longitude, direction, speed, altitude,
				extract(epoch from measuring_time::timestamp without time zone)::integer
				as timestamp
				FROM history.monitoring_records
				WHERE vehicle_id = ?
				AND measuring_time >=
				CAST (to_timestamp(?) at time zone 'utc' AS timestamp without time zone)
				AND measuring_time <=
				CAST (to_timestamp(?) at time zone 'utc' AS timestamp without time zone)
				ORDER BY measuring_time ASC", array($f3->get('PARAMS.id'), $from->getTimestamp(), $to->getTimestamp()));

	if($res == null)
	{
		$f3->error(404);
	}

	header('Content-Type: application/json');
	$f3->status(200);

	echo json_encode($res, JSON_UNESCAPED_UNICODE);
}
```

Рисунок 2.28 – Метод GET one для vehicle history

После того, как работа с данными будет завершена, сервер сохраняет данные в формате JSON. Примерный результат представлен на рисунке 2.29:


```
[
    {
        "id": 1,
        "name": "ИП Алейникова Л.В.",
        "parent_organization_id": 0,
        "is_blocked": false,
        "created_at": "2019-10-20 00:00:00",
        "expired_at": "2020-10-20 00:00:00"
    },
    {
        "id": 2,
        "name": "ИП Сотикова М.Д.",
        "parent_organization_id": 0,
        "is_blocked": false,
        "created_at": "2019-10-20 00:00:00",
        "expired_at": "2020-10-20 00:00:00"
    }
]
```


Рисунок 2.29 – Результат успешного выполнения запроса

Однако, в некоторых случаях, клиент и сам может передавать запросы на сервер в данном формате.

3 Разработка клиентского приложения АИС «Управление транспортом»

3.1 Предпроектное исследование клиентской части АИС «Управление транспортом»

Внедрение АИС «Управление транспортом» позволит создать единую информационную платформу региона по мониторингу, контролю и управлению пассажирским транспортом, координировать работы различных служб, предприятий и организаций, а также предоставлять актуальную информацию в on-line режиме по маршрутам, расписанию, обстановке на дорогах жителям области.

Основными задачами данной системы являются:

– осуществление мониторинга функционирования общественного транспорта;

– формирование и оптимизация единой маршрутной сети общественного транспорта;

– осуществление диспетчерского управления общественным транспортом;

– информирование водителей об изменении режима движения;

– информирование пассажиров на остановках о времени прибытия транспорта;

– получение информации о нештатных ситуациях, подсчета пассажиропотока.

Осуществление мониторинга функционирования и управления пассажирским транспортом в рамках вышеуказанной системы позволит исполнительным органам государственной власти и органам местного самоуправления муниципальных образований:

- проводить полный учет и контроль, а также и хранение информации об объектах пассажирского транспорта, его инфраструктуре;

- вести статистический анализ актуальной информации по пассажирскому транспорту;

− применять мониторинговую информацию при проведении анализа и принятии управленческих решений в сфере пассажирского транспорта;

− повысить эффективность пассажироперевозок за счет оптимизации маршрутной сети.

АИС «Управление транспортом» позволит дополнительно расширить функциональность РСЭП в результате автоматизации следующих процессов, связанных с предоставлением Услуг в электронной форме:

1. Процесс формирования информации о транспортной сети и транспортной инфраструктуре региона в графическом и семантическом виде.
2. Процесс формирования маршрутных расписаний. Оптимизация транспортной сети на основании данных пассажиропотока. Определение потребного количества транспортных средств и распределение их по маршрутам.
3. Процесс долгосрочного и оперативного суточного планирования пассажирских перевозок.
4. Процесс оперативного контроля и управления пассажирскими перевозками.
5. Процесс получения оценки качества перевозочного процесса.
6. Процесс загрузке/выгрузке на порталы государственных услуг Российской Федерации (ЕПГУ и/или РПГУ) актуальной информации о работе автомобильного транспорта и городским наземным электрическим транспортом, осуществляющего регулярную перевозку пассажиров и багажа маршрутов в разрезе муниципальных образований субъекта Российской Федерации.
7. Процесс предоставления пользователям доступ к Порталу в части получения информации о маршрутной сети и времени прибытия транспортных средств на остановочные пункты.
8. Процесс предоставление пользователям (через мобильное устройство) доступа к Порталу в части получения информации о маршрутной сети и времени прибытия транспортных средств на остановочные пункты.

Модуль «Пассажир» нацелен на работу с пользователями, которым нужна информация о маршруте, текущем местоположении транспорта и справочной

информации о ресурсе и юридических документах, регулирующих деятельность перевозчиков в Ставропольском крае.

В Ставропольском крае на сегодняшний день функционирует ГИС «Управление транспортом», находящаяся по адресу https://transport.stavregion.ru. (рисунок 3.1) [11]

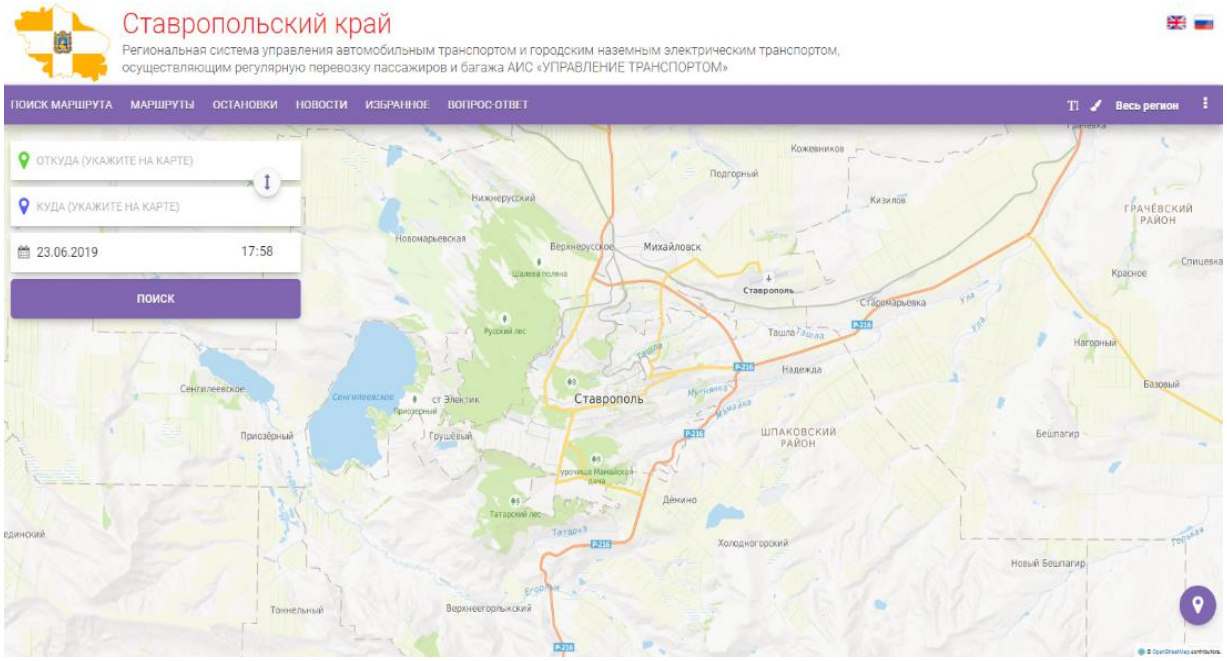

Рисунок 3.1 – АИС «Управление транспортом» в Ставропольском крае

Характеристики клиентской части вынесем в таблицу (Таблица 3.1).

Таблица 3.1 – Характеристики клиентской части

№ п/п	Основные элементы	Доп. элементы	Опр. местоположения	Доп. возможности
1	Поиск маршрута	Автоперевозчики	Есть	Нет
2	Маршруты	Информация о маршрутах		
3	Остановки	Контролирующие органы		
4	Избранное	Правила пользования		
5	Весь регион	О системе		

6		Статистика		
7		Загрузить мобильное приложение		

В Москве функционирует ГИС «Московский транспорт» и предоставляет достаточно широкий круг функциональности. К системе можно обратиться по адресу http://transport.mos.ru/.[12] Приветственный экран выглядит следующим образом (рисунок 3.2):

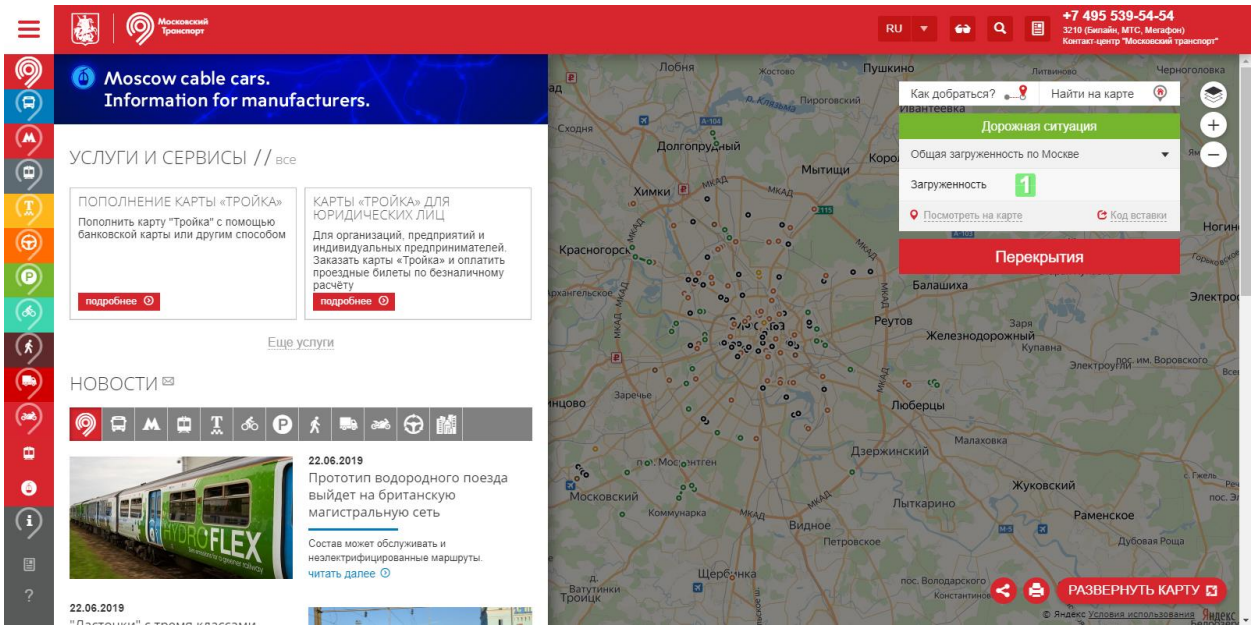

Рисунок 3.2 – ГИС «Московский транспорт»

Следующим образом выглядит система, если развернуть карту (рисунок 3.3):

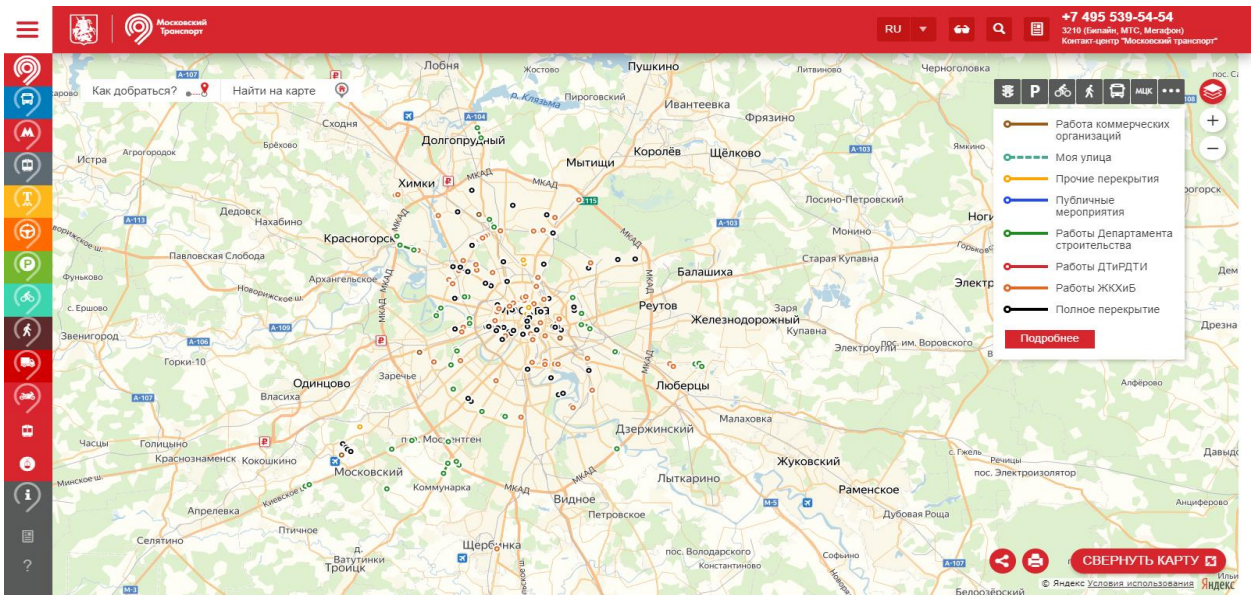

Рисунок 3.3 – ГИС «Московский транспорт»

Таблица 3.2 - Характеристики клиентской части вынесем в таблицу:

№ п/п	Основные элементы	Доп. элементы	Опр. местоположения	Доп. возможности
1	Московский транспорт	Развернуть карту	Нет	Версия для слабовидящих
2	Наземный транспорт	Поделиться картой		Учет пробок
3	Метрополитен и МЦК	Распечатать карту		Расписание маршрута
4	Пригородное сообщение	Как добраться		Расчет маршрута по общественному транспорту или такси
5	Такси	Найти на карте		Исчерпывающая справочная информация в каждом из разделов основных элементов управление
6	Каршеринг	Дорожная ситуация		
7	Паркинг	Поиск		
8	Велотранспорт	Версия для слабовидящих		
9	Пешеходы	Переключение языка		
10	Грузовая логистика			

11	Мототранспорт			
12	Московские центральные диаметры			
13	Правила перевозки			
14	Оставить обращение			
15	Справка			

ГИС «Портал общественного транспорта Санкт-Петербурга» является одной из наиболее функциональных транспортных систем. Отличительной чертой данной системы является отображение положения всех транспортных средств на карте с обновлением в режиме реального времени, а также возможность выбора отображения ТС, которые оборудованы для перевозок людей с ограниченными возможностями. К системе можно обратиться по адресу http://transport.orgp.spb.ru/Portal/transport/main (рисунок 3.4).[13]

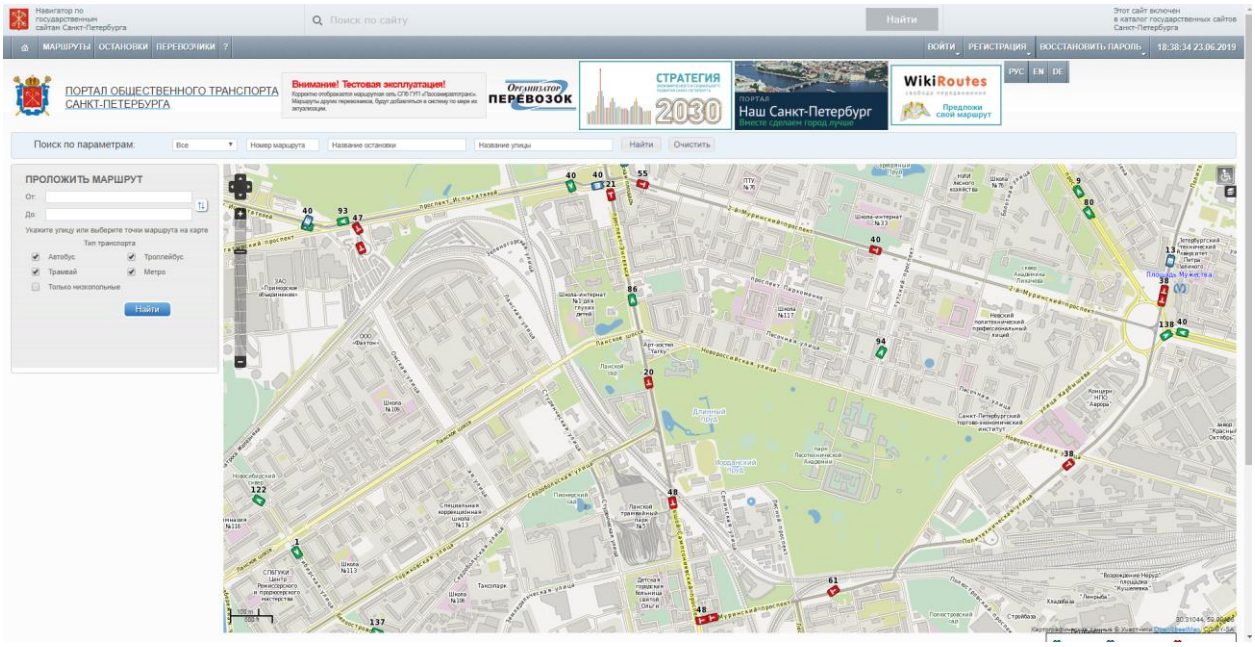

Рисунок 3.4 – ГИС «Портал общественного транспорта Санкт-Петербурга»

Существенным недостатком этой системы, по моему мнению, является то, что необходимо прокрутить экран вниз, чтобы увидеть легенду карты. Таким образом, рабочая область портала выглядит как представлено на рисунке 3.5:

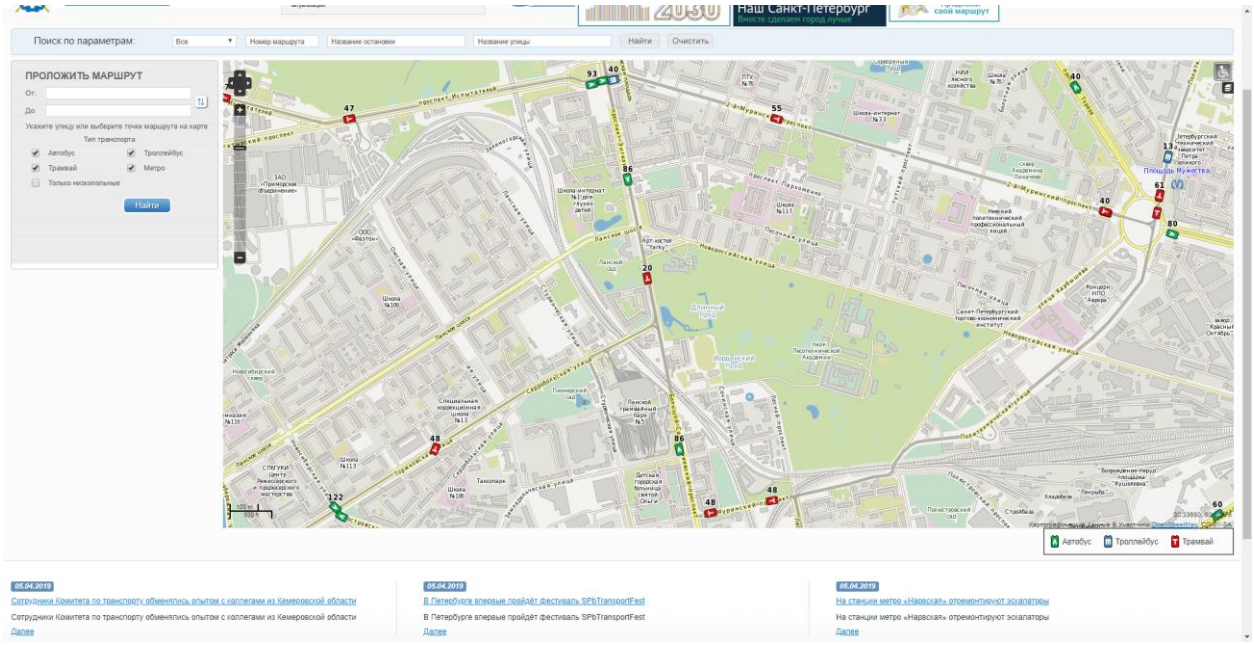

Рисунок 3.5 – ГИС «Портал общественного транспорта Санкт-Петербурга», рабочая область

Таблица 3.3 – Характеристики клиентской части вынесем в таблицу:

№ п/п	Основные элементы	Доп. элементы	Опр. местоположения	Доп. возможности
1	Маршруты	Войти	Нет	Отображение низкопольных ТС для инвалидов
2	Остановки	Регистрация		Обновление информации на карте в режиме реального времени
3	Перевозчики	Восстановить пароль		Личный кабинет
4	Справка			Переключение языка
5	Поиск по параметрам			Легенда карты
6	Проложить маршрут			

ГИС «Официальный портал транспортного информирования граждан» в Волгоградской области по своему внешнему оформлению является одной из самых лаконичных. Обратиться к системе можно по адресу http://transport.volganet.ru/main.php.[6] Приветственный экран выглядит следующим образом (рисунок 3.6):

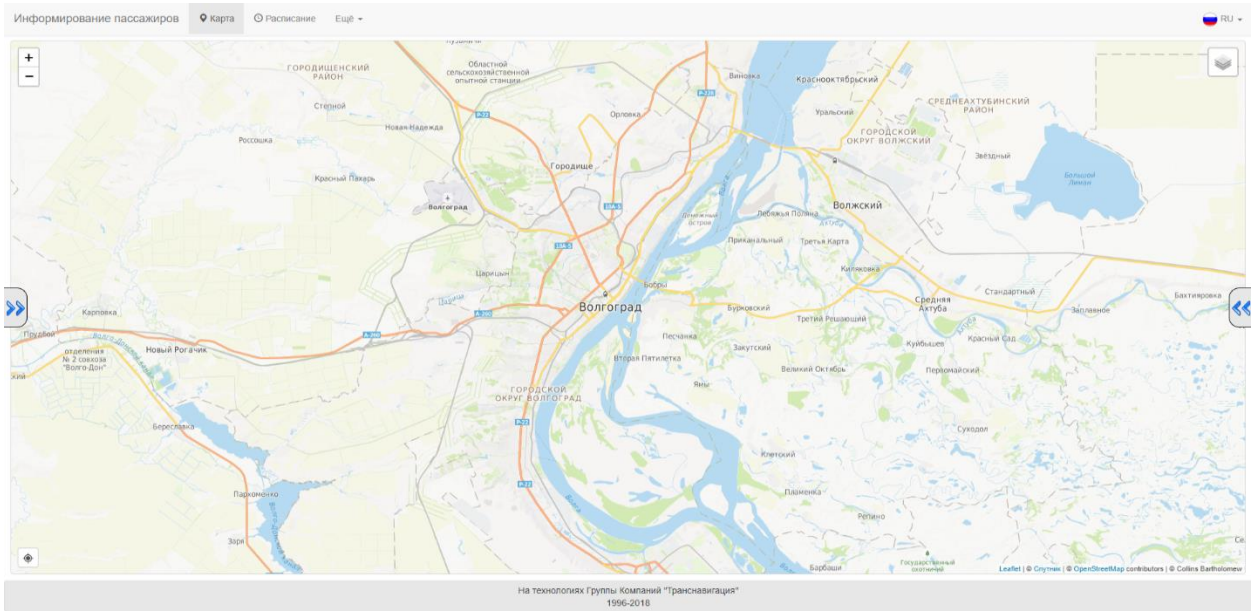

Рисунок 3.6 – ГИС «Официальный портал транспортного информирования граждан» в Волгоградской области

Как видно из снимка экрана, пользователь сразу видит перед собой карту, а все элементы управления спрятаны в меню. Если их развернуть, то рабочая область выглядит следующим образом (рисунок 3.7):

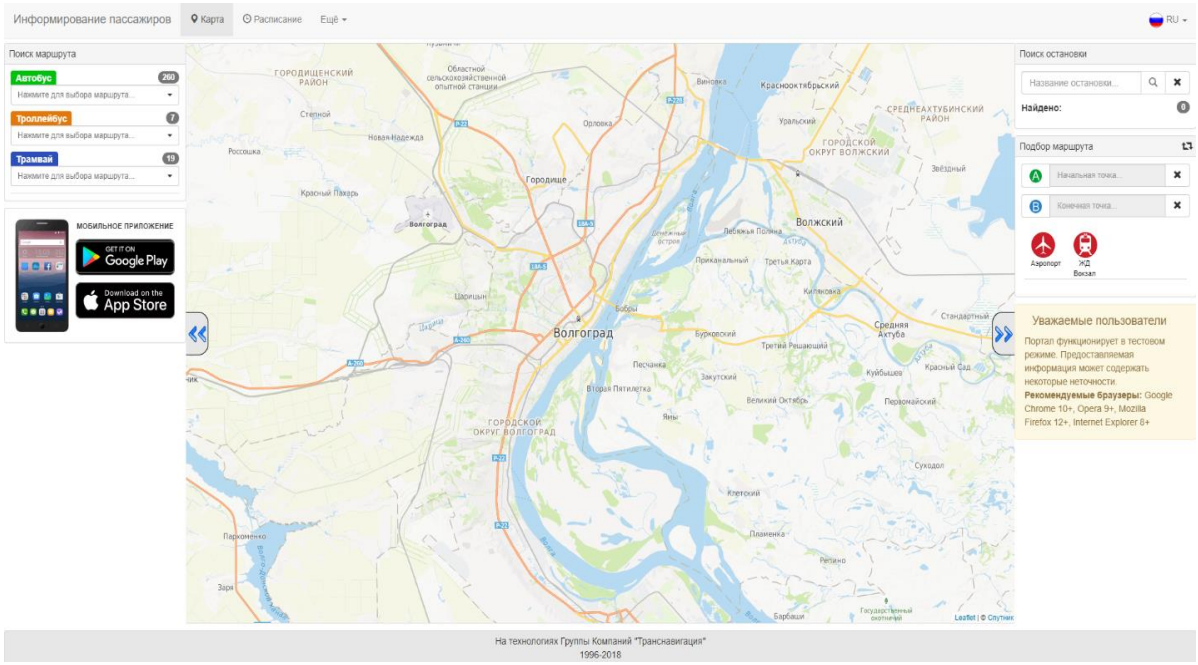

Рисунок 3.7 – ГИС «Официальный портал транспортного информирования граждан», рабочая область.

Таблица 3.4 – Характеристики клиентской части вынесем в таблицу:

№ п/п	Основные элементы	Доп. элементы	Опр. местоположения	Доп. возможности
1	Карта	Справка	Нет	Нет
2	Расписание	Настройки		
3	Поиск маршрута (стрелки слева)	Панели		
4	Поиск остановки (стрелки справа)	Написать разработчикам		
5	Подбор маршрута	Мобильная версия		
6	Переключение языка	О портале		

С точки зрения структурного анализа, путем декомпозиции, в данных системах можно выделить общие объекты в структуре в клиентской части:

1) Карта
2) Поиск маршрута
3) Поиск остановок

4) Справочное меню и меню дополнительной возможностей

5) Кнопка определения местоположения

Таким образом, модуль «Пассажир» необходимо представить в следующей совокупности объектов (рисунок 3.8):

Рисунок 3.8 – Диаграмма модуля «Пассажир»

3.2 Составление частного ТЗ разработки клиентской приложения АИС

Для составление частного ТЗ на разработку клиентского приложения было обработано задание на оказание услуг по созданию региональной информационной системы управления автомобильным транспортом и городским наземным электрическим транспортом, используемым для предоставления услуг на территории ставропольского края в целях предоставления услуг в электронной форме на портале государственных и муниципальных услуг (функций), предоставляемых (исполняемых) органами исполнительной власти и органами местного самоуправления муниципальных образований ставропольского края для обеспечения государственных нужд ставропольского края.

Частное техническое задание будет разработано согласно ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы:

1. Общие сведения

1.1. Наименование системы

1.1.1. Полное наименование системы

Полное наименование: региональная информационная система управления автомобильным транспортом и городским наземным электрическим транспортом, используемая для предоставления услуг на территории Ставропольского края в целях предоставления услуг в электронной форме на портале государственных и муниципальных услуг (функций), предоставляемых (исполняемых) органами исполнительной власти и органами местного самоуправления муниципальных образований Ставропольского края для обеспечения государственных нужд Ставропольского края

1.1.2. Краткое наименование системы

Краткое наименование: автоматизированная информационная система «Управление транспортом».

1.2. Основания для проведения работ

Создание АИС «Управление транспортом» ведется на основании следующих документов:

– Федеральный закон от 13.07.2015 г. № 220-ФЗ «Об организации регулярных перевозок пассажиров и багажа автомобильным транспортом и городским наземным электрическим транспортом в Российской Федерации и о внесении изменений в отдельные законодательные акты Российской Федерации».

– Федеральный закон от 8 ноября 2007 года N 259-ФЗ «Устав автомобильного транспорта и городского наземного электрического транспорта».

– Федеральный закон от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации».

– Федеральный закон от 27 июля 2006 г. № 152-ФЗ «О персональных данных».

– Федеральный закон от 27 июля 2010 года № 210-ФЗ «Об организации предоставления государственных и муниципальных услуг».

– Постановление Правительства Российской Федерации от 8 сентября 2010 г. №697 «О единой системе межведомственного электронного взаимодействия».

– Постановление Правительства Российской Федерации от 8 июня 2011г. №451 «Об инфраструктуре, обеспечивающей информационно-технологическое взаимодействие информационных систем, используемых для предоставления государственных и муниципальных услуг в электронной форме».

– Постановление Правительства Российской Федерации от 27 сентября 2011 г. №797 «О взаимодействии между многофункциональными центрами предоставления государственных (муниципальных) услуг и федеральными органами исполнительной власти, органами государственных внебюджетных фондов, органами государственной власти субъектов Российской Федерации, органами местного самоуправления».

– Постановление Правительства Российской Федерации от 24 октября 2011 г. №861 «О федеральных государственных информационных системах, обеспечивающих предоставление в электронной форме государственных и муниципальных услуг (осуществление функций)».

– Постановление Правительства Российской Федерации от 28 ноября 2011 г. №977 «О федеральной государственной информационной системе «Единая система идентификации и аутентификации в инфраструктуре, обеспечивающей информационно-технологическое взаимодействие информационных систем, используемых для предоставления государственных и муниципальных услуг в электронной форме».

– Постановление Правительства Российской Федерации от 22 декабря 2012 г. № 1376 «Об утверждении Правил организации деятельности многофункциональных центров предоставления государственных и муниципальных услуг».

– Приказ Министерства связи и массовых коммуникаций Российской Федерации от 13 апреля 2012 г. № 107 «Об утверждении Положения о федеральной государственной информационной системе «Единая система идентификации и аутентификации в инфраструктуре, обеспечивающей информационно-технологическое взаимодействие информационных систем, используемых для предоставления государственных и муниципальных услуг в электронной форме».

– Федеральный закон от 07.07.2003 № 126-ФЗ «О связи».

– Постановление Правительства Российской Федерации от 17 декабря 2013 г. № 1177 "Об утверждении Правил организованной перевозки группы детей автобусами".

– Постановление Правительства РФ от 25.08.2008 N 641 (ред. от 17.12.2010) "Об оснащении транспортных, технических средств и систем аппаратурой спутниковой навигации ГЛОНАСС или ГЛОНАСС/GPS".

– Указ Президента Российской Федерации от 17 мая 2007 г. N 638 "Об использовании глобальной навигационной спутниковой системы ГЛОНАСС в интересах социально-экономического развития Российской Федерации".

– Приказ Минэкономразвития РФ от 01.04.2010 г. №123 "Об определении видов оборудования, используемого при проведении геодезических и кадастровых услуг и подлежащего оснащению аппаратурой спутниковой навигации ГЛОНАСС или ГЛОНАСС/GPS".

– Приказ Мининформсвязи РФ от 19.06.2007 N 68 «Об утверждении Правил применения оборудования автоматизированных систем управления и мониторинга сетей электросвязи. Часть II. Правила применения оборудования автоматизированных систем управления и мониторинга средств связи, выполняющих функции цифровых транспортных систем».

2. Цели создания системы

Целями создания АИС «Управление транспортом» являются:

1. Принятие управленческих решений по повышению уровня информирования населения о услуге автомобильного и городского наземного электрического транспорта, осуществляющего регулярную перевозку пассажиров и багажа в Ставропольском крае.

2. Принятие управленческих решений по повышению привлекательности использования транспорта общего пользования населением Ставропольского края.

3. Принятие управленческих решений по обеспечению возможности повышения качества обслуживания маршрутов регулярных перевозок пассажиров и багажа за счет получения объективной информации о движении транспортных средств, обслуживающих маршруты.

4. Принятие управленческих решений по снижению количества личного автомобильного транспорта на дорогах Ставропольского края за счет увеличения количества перевозимых пассажиров.

5. Принятие управленческих решений по обеспечению информационной открытости и прозрачности деятельности органа исполнительной власти, ответственного за организацию пассажирских перевозок, за счет автоматизации процессов информирования населения.

6. Информационное сопровождение принятия управленческих решений при организации регулярных перевозок пассажиров и багажа автомобильным транспортом и городским наземным электрическим транспортом, в том числе связанных с установлением, изменением, отменой маршрутов регулярных перевозок, допуском юридических лиц и индивидуальных предпринимателей к осуществлению регулярных перевозок, использованием для осуществления регулярных перевозок объектов транспортной инфраструктуры, а также с организацией контроля за осуществлением регулярных перевозок.

3. Характеристика объектов автоматизации

Объектом автоматизации являются процессы принятия управленческих решений по направлениям:

1. Обеспечение деятельности органов исполнительной власти Ставропольского края и органов местного самоуправления по ведению реестров маршрутов регулярных перевозок, оформлению, переоформлению свидетельств об осуществлении перевозок по маршруту регулярных перевозок, карт маршрута регулярных перевозок;

2. Обеспечение выполнения контрольных функций органов исполнительной власти Ставропольского края и органов местного самоуправления за осуществлением регулярных перевозок;

3. Повышение уровня информирования населения о услуге автомобильного и городского наземного электрического транспорта, осуществляющего регулярную перевозку пассажиров и багажа в Ставропольском крае;

4. Повышение привлекательности использования транспорта общего пользования населением Ставропольского края;

5. Обеспечение возможности повышения качества обслуживания маршрутов регулярных перевозок пассажиров и багажа за счет получения объективной информации о движении транспортных средств, обслуживающих маршруты;

6. Снижение количества личного автомобильного транспорта на дорогах Ставропольского края за счет увеличения количества перевозимых пассажиров.

7. Обеспечение оказания перевозчиками государственных услуг, связанных с установленным в соответствии с Федеральным Законом №220-ФЗ порядком организации регулярных перевозок пассажиров и багажа автомобильным транспортом и городским наземным электрическим транспортом.

Объекты автоматизации состоят из следующих функциональных блоков:

1. Выполнение органами исполнительной власти Ставропольского края и органами местного самоуправления функций по регистрации маршрута в реестре маршрутов регулярных перевозок, учета и сопровождения государственных или муниципальных контрактов, оформления свидетельства об осуществлении перевозок по маршруту регулярных перевозок, карт маршрута регулярных перевозок.

2. Формирование органами исполнительной власти Ставропольского края и органами местного самоуправления единой базы данных по маршрутам регулярных перевозок, размещение сведений, включенных в реестры маршрутов регулярных перевозок на официальных сайтах.

3. Информирование о услуге транспорта общего пользования: расписания движения по маршрутам, трассы маршрутов, ожидаемое время прибытия транспортного средства, находящегося на маршруте, на определенный остановочный пункт.

4. Информирование населения об изменениях в режиме работы маршрутов регулярных перевозок пассажиров и багажа в Ставропольском крае.

5. Информирование населения о внештатных ситуациях, возникающих на маршруте: заторы, задержки, пробки, аварии и др.

6. Предоставление общей информации о услуге транспорта общего пользования: протяженность маршрутной сети, количество транспортных средств на маршрутах, характеристики транспортных средств на маршрутах.

7. Предоставление перевозчиками, с которыми заключен государственный или муниципальный контракт либо которым выдано свидетельство об осуществлении перевозок по маршруту регулярных перевозок, ежеквартальных отчетов об осуществлении регулярных перевозок.

4. Требования к системе

АИС «Управление транспортом» должна функционировать на всей территории, охваченной маршрутами автомобильного и городского наземного электрического транспорта субъекта Российской Федерации, включая маршруты внутри региона и межрегиональные маршруты.

Архитектура (интерфейс, протоколы) системы АИС «Управление транспортом» должна реализовываться на базе современных web-технологий, быть открытой и обеспечивать следующие принципы:

– модульный принцип построения, обеспечивающий возможность добавления/изменения/удаления модулей без необходимости изменения архитектуры в целом;

– построение по принципу многоуровневой архитектуры клиент-сервер с использованием в качестве клиента современных веб-браузеров.

3.3 Проектирование клиентского приложения АИС

В рамках анализа существующих систем, были выделены основные составляющие модуля «Пассажир». Так как АИС «Управление транспортом» представляет собой портал, то доступ к нему будет осуществляться через web-браузер. Использование web-браузера влечет за собой использование следующих технологий и языков программирования:

1) HTML;
2) CSS;
3) Javascript.

Проводя анализ существующих систем, мною был отмечен тот факт, что у большинства из анализируемых порталов есть следующие недостатки:

1) Неудобная навигация по порталу;
2) Интерфейс не интуитивно понятен;
3) Неудачный выбор цветов, из-за чего нет акцента на основной функциональности клиентской части.

Исходя из вышеуказанного, был проведен визуальный анализ таких систем, как Google Map, Yandex.Maps и 2Gis (рисунки 3.9 – 3.11).

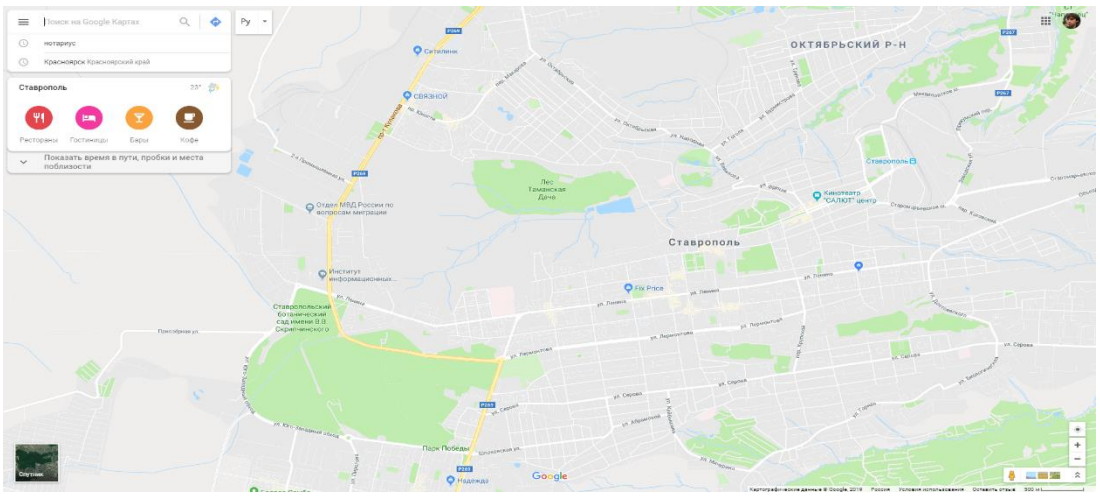

Рисунок 3.9 – Google Maps

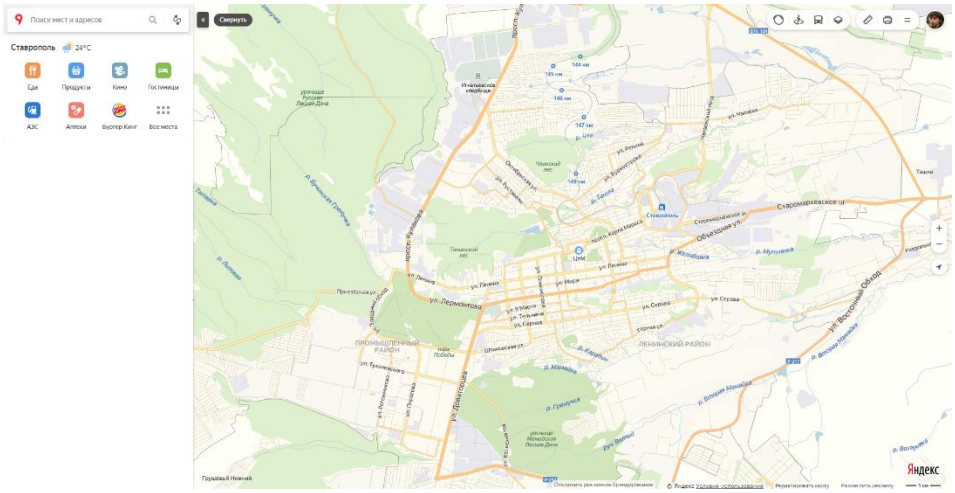

Рисунок 3.10 – Yandex.Maps

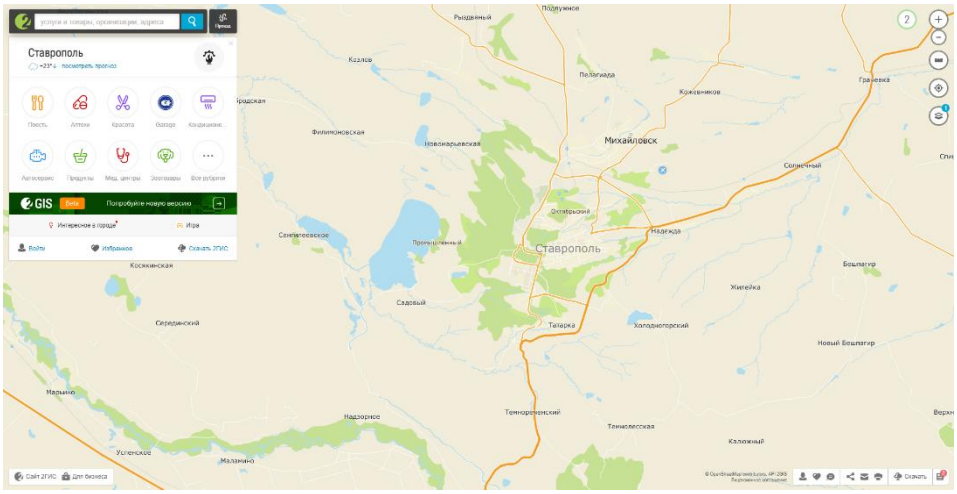

Рисунок 3.11 – 2Gis

У данных систем наблюдается следующие общие черты:

1) Карта занимает весь экран;

2) Отсутствие полос прокрутки;

3) Элементы управления картой находятся в противоположной части экрана;

4) Основные элементы управления находятся в левом верхнем углу экрана;

5) Элементы управления и навигации отделяются тенью под элементами или используют перекрытие карты в полную высоту занимаемой области, что создает контраст.

Исходя из этого, был подготовлен прототип дизайна в следующем виде (рисунок 3.12):

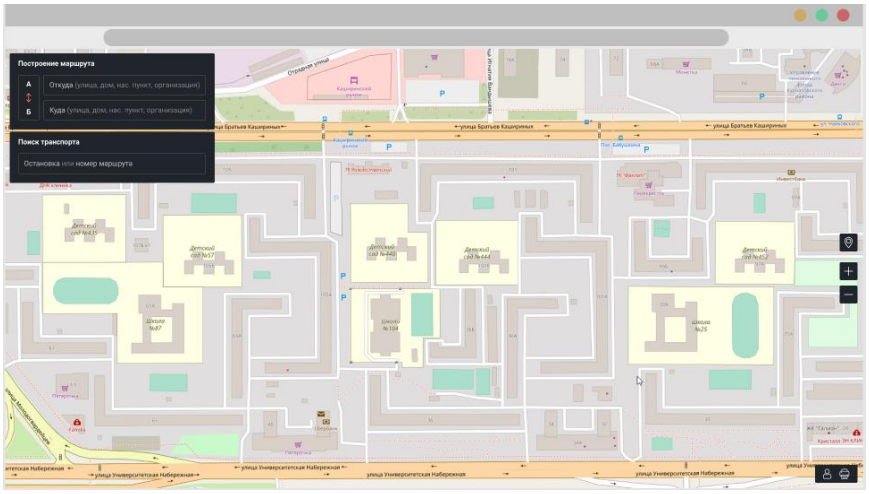

Рисунок 3.12 – Главное окно модуля «Пассажир»

В том случае, когда найден маршрут, на экран выводится блок со списком маршрутов (рисунок 3.13):

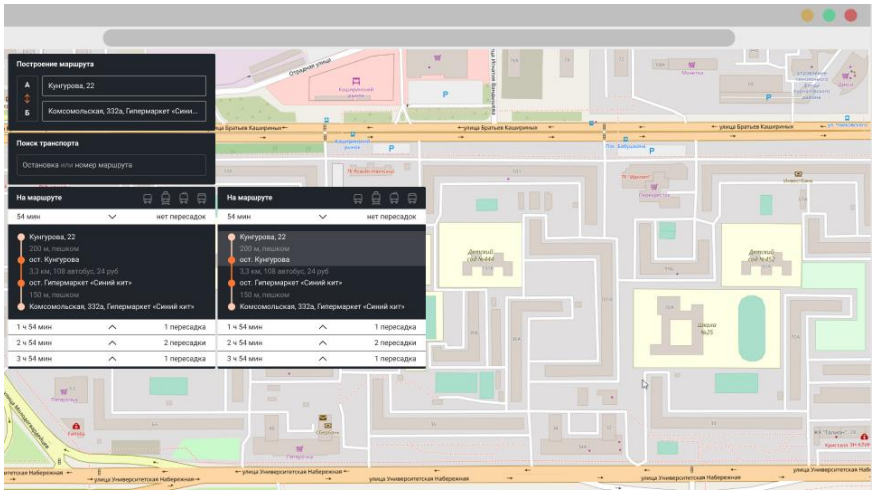

Рисунок 3.13 – Список маршрутов

В шапке этого блока находится фильтр по типам транспорта: автобус, маршрутка, трамвай, троллейбус. Так на, на рисунке изображен блок с поведением элемента списка маршрута при наведении.

В случае, если производится поиск маршрута по номеру маршрута, то на экран выводится следующий вариант блока, иллюстрированного выше (рисунок 3.14):

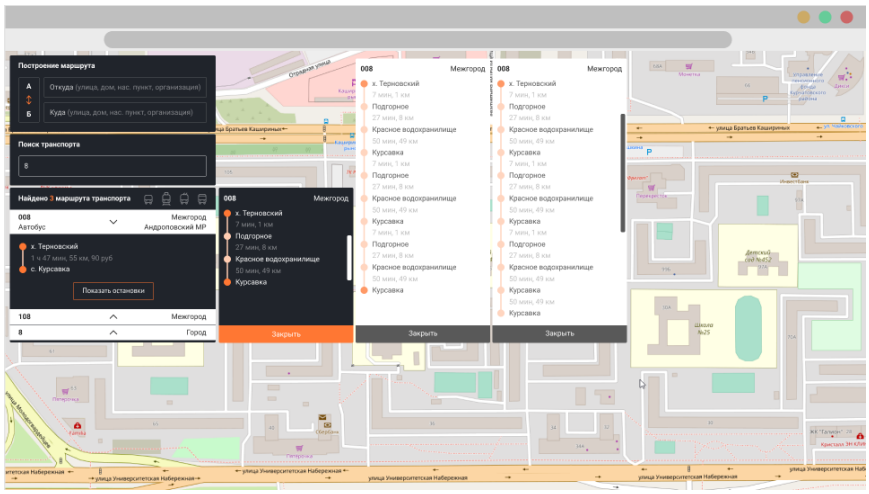

Рисунок 3.14 – Список маршрутов по номеру маршрута

На данном рисунке так же иллюстрируется полный список остановок на маршруте, в случае если маршрут междугородний, либо остановок слишком много, чтобы уместить его в родительский блок.

Как уже было сказано выше, использование web-браузера влечет за собой использование следующих технологий и языков программирования:

1) HTML;
2) CSS;
3) Javascript.

В качестве сервис-провайдера карты будем использовать бесплатный сервис OpenStreetMap. В руководстве разработчика на официальном сайте OpenStreetMap[7] указаны библиотеки, предоставляющие доступ к API сервиса. Для модуля разрабатываемого модуля будет использоваться Javascript библиотека «Leaflet».[8]

Исходя из ТЗ, большинство получаемых модулем данных, а также данные по работе с картами должны обновляться динамически на стороне клиента. Исходя из этого, в разработке модуля будет использоваться методология реактивного программирования.

Реактивное программирование — методология программирования, ориентированная на потоки данных и распространение изменений. Это означает, что должна существовать возможность легко выражать статические и динамические потоки данных, а также то, что нижележащая модель исполнения должна автоматически распространять изменения благодаря потоку данных.

К примеру, в императивном программировании присваивание a = b + c будет означать, что переменной a будет присвоен результат выполнения операции b + c, используя текущие (на момент вычисления) значения переменных. Позже значения переменных b и c могут быть изменены без какого-либо влияния на значение переменной a.

В реактивном же программировании значение a будет автоматически пересчитано, основываясь на новых значениях. Т.е. реактивное программирование — программирование с асинхронными потоками данных. [18] Соответственно, необходимо найти инструмент, предоставляющий данную функциональность и облегчит разработку модуля и разработку интерфейса модуля.

В качестве данного инструмента будет использовать Javascript-фреймворк Vue.js.

Фреймворк можно рассматривать как некий каркас для создания приложений на определенном языке программирования. Он включает в себя набор библиотек, которые позволяют значительно упростить и убыстрить разработку приложений. Также цель фреймворка – это предоставление такой среды разработки, которая позволит дорабатывать и расширять функционал проекта.

Фреймворк использует определенную архитектуру приложения для того, чтобы проект был разделен на логические части, модули. Например, схема разделения Model-View-Controller (MVC) подразумевает разделение на три сегмента: модель, представление и контроллер, каждый из которых можно изменять независимо от других.

Если говорить более упрощенно, то существует множество типовых задач, которые приходится решать разработчикам – и поэтому часто используются библиотеки. Использование фреймворка позволяет избежать отдельного «складирования» полезного материала где-нибудь в отдельной папке либо постоянного написания однотипного кода: фреймворк позволяет использовать встроенные классы для валидации, логирования и многих других процессов, а также имеет структуру, которую в итоге получает проект. [19]

Vue — это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications). [11]

Далее рассмотрим бизнес-процессы в модуле. Бизнес-процесс – совокупность различных видов деятельности, в рамках которой «на входе» некоторый ресурс, и в результате этой деятельности «на выходе» создается продукт, представляющий ценность для потребителя. [12]

Бизнес-логика — это реализация предметной области в информационной системе. [13]

Так как модуль «Пассажир» будет состоять из нескольких блоков, то для каждого из них будет описана бизнес-логика (рисунок 3.15).

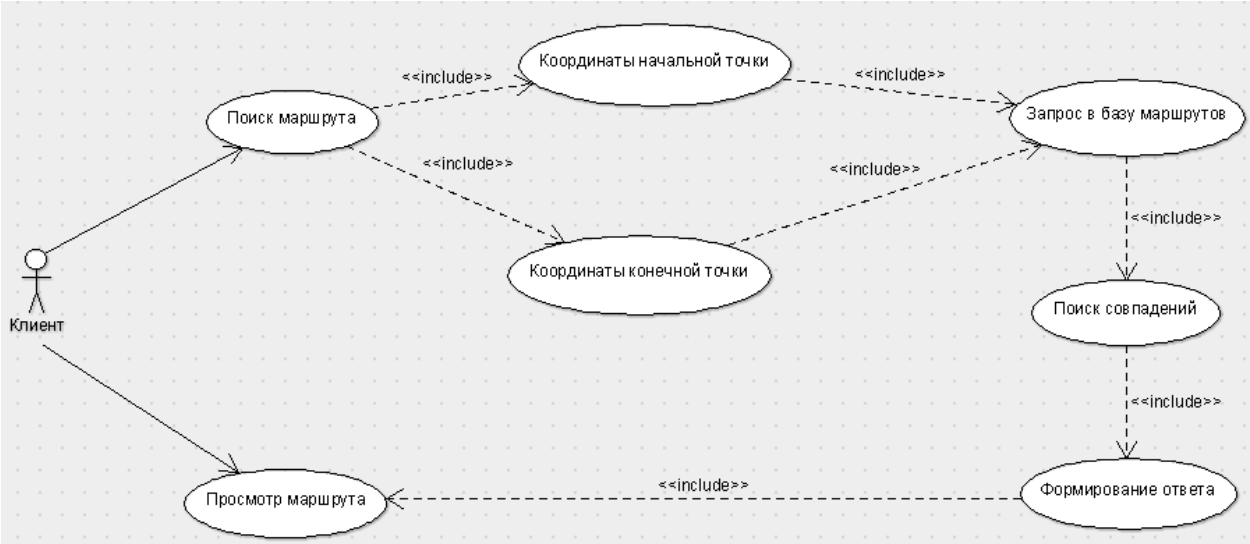

Рисунок 3.15 – Бизнес-логика поиска маршрута из точки А в точку Б.

Бизнес-логика поиска маршрута из точки А в точку Б выглядит следующим образом:

1) пользователь задает координаты точек
2) модуль обращается в базу маршрутов
3) база обрабатывает запрос и подбирает релевантные маршруты
4) формируется ответ базы
5) отправка ответа пользователю

Далее, рассмотрим бизнес-логику поиска маршрута по номеру (рисунок 3.16).

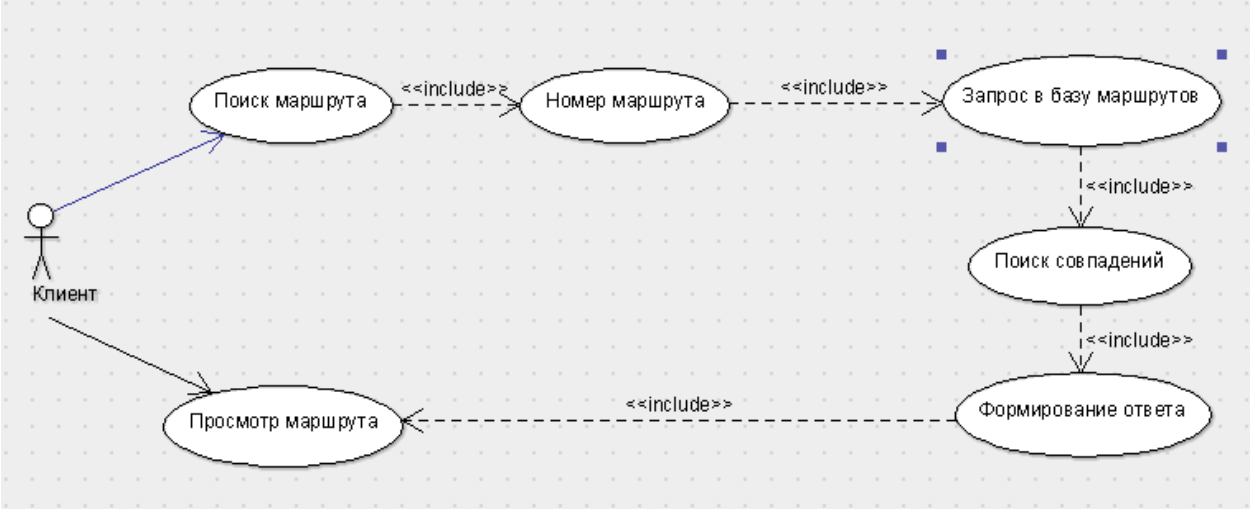

Рисунок 3.16 – Бизнес-логика поиска маршрута по номеру.

Бизнес-логика поиска маршрута по номеру выглядит следующим образом:

1) пользователь задает номер маршрута;

2) модуль обращается в базу маршрутов;

3) база обрабатывает запрос и ищет маршруты, включающие данный номер;

4) формируется ответ базы;

5) отправка ответа пользователю.

Отличие данной бизнес-логики от предыдущей состоит в том, что:

1. В случае отсутствия маршрута по запросу, пользователю могут быть предоставлены наиболее похожие маршруты, включающие номер из запроса;

2. Ответ предоставляется согласно типу маршрута: междугородний или городской.

Бизнес-логика поиска местоположения пользователя основывается на Geolocation API. Geolocation API позволяет пользователю предоставлять свое местоположение web-приложению, если пользователь даст согласие на это действие. Из соображений конфиденциальности, у пользователя будет запрошено разрешение на предоставление информации о местоположении. Функция поиска местоположения инициирует асинхронный запрос для обнаружения местоположения пользователя, и запрашивает аппаратные средства позиционирования, чтобы получить последнюю актуальную информацию (рисунок 3.17).

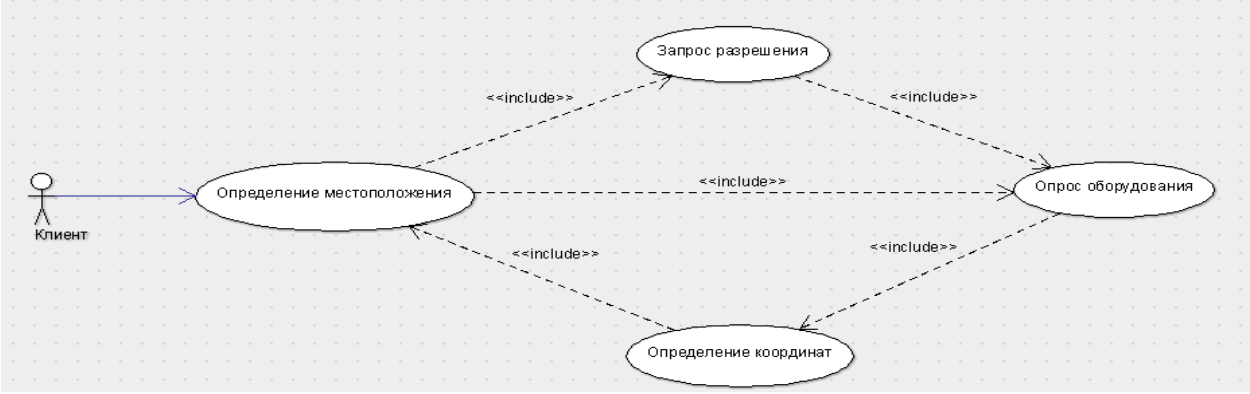

Рисунок 3.17 – Бизнес-логика определения местоположения.

В общем виде взаимодействие модулей и карты выглядит как показано на рисунке 3.18:

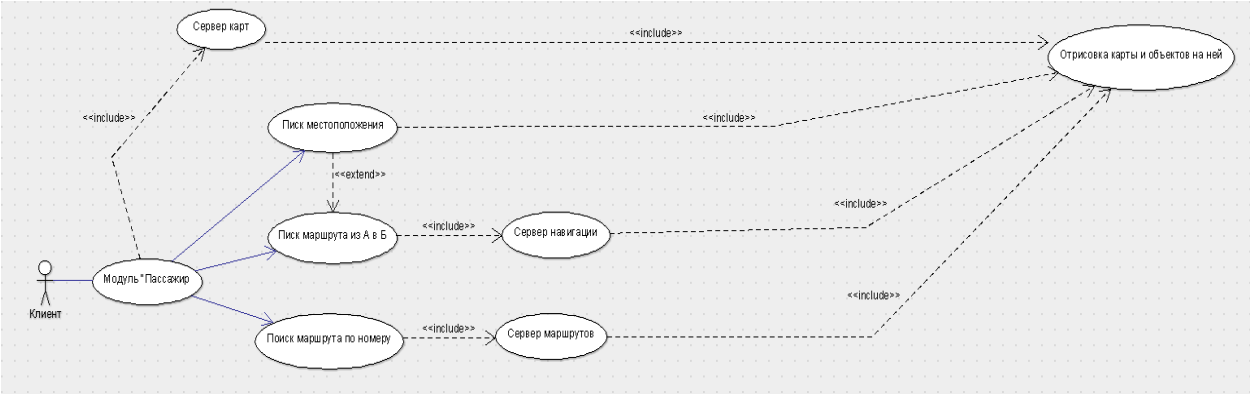

Рисунок 3.18 – Взаимодействие модулей с картой

3.4 Разработка клиентского приложения АИС

3.4.1 Разработка модуля «Регулятор»

Архитектура web-приложения обусловлена лучшим практикам разработки web-приложений на Vue.js. Модуль «Регулятор» выглядит следующим образом (рисунок 3.19):

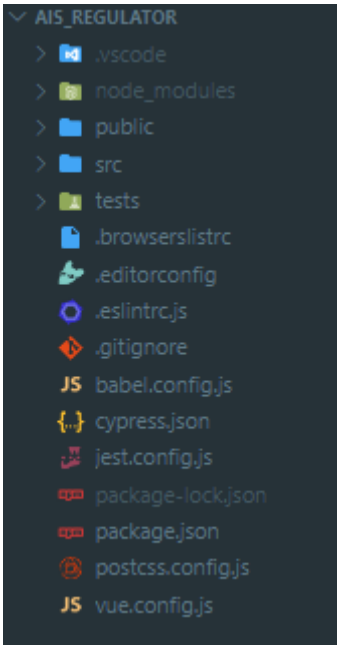

Рисунок 3.19 – Архитектура модуля «Регулятор»

На рисунке отображены следующие директории:

– *.vscode* – настройки проекта для редактора Visual Studio Code [13];

– *node_modules* – модули менеджера пакетов Node Package Manager[14], необходимые для разработки модуля;

– *public* – директория с файлами модуля, готовыми для размещения на хостинге;

– *src* – директория с исходными файлами проекта;

– *tests* – директория, содержащие необходимые инструменты для тестирования компонентов модуля.

Остальные файлы несут вспомогательный характер, т.к. содержат себе некоторые настройки, необходимые для работы модулей или содержащие правила для программиста, соблюдение которых будет контролироваться средой разработки.

В структуре директории *src* находятся следующие директории (3.20):

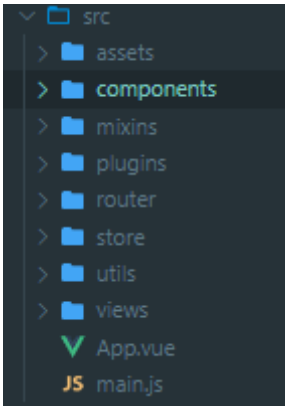

Рисунок 3.20 – Структура исходных файлов модуля «Регулятор»

– *assets* – содержит статические файлы проекта, такие как изображения и иконки;

– *components* – содержит компоненты приложения, такие как карта, кнопки управления, виджеты, баблы, поп-апы, таблицы и т.п.;

– *mixins* – содержит миксины; участки кода, которые часто используются в других компонентах и страницах;

– *plugins* – содержит программные надстройки проекта, такие как провайдер карт, библиотека для работа с API и AJAX-запросами, настройки библиотеки визуальных компонентов и т.п.;

– *router* – содержит маршрутизатор, обеспечивающий навигацию внутри модуля;

– *store* – содержит настройки хранилища данных;

– *utils* – то же, что и плагины, но необходимые только в режиме разработки;

– *views* – страницы приложения;

– *App.vue* – файл, отвечающий за отрисовку приложения на клиенте;

– *main.js* – файл, отвечающий за функционирование проекта в режиме разработки.

После обработки ТЗ заказчика, было решено создать интерфейс модуля регулятора согласно следующей структуре (рисунок 3.21):

1. Блок навигации, в котором размещаются разделы, сгруппированные в категории;
2. Блок отображения страницы.

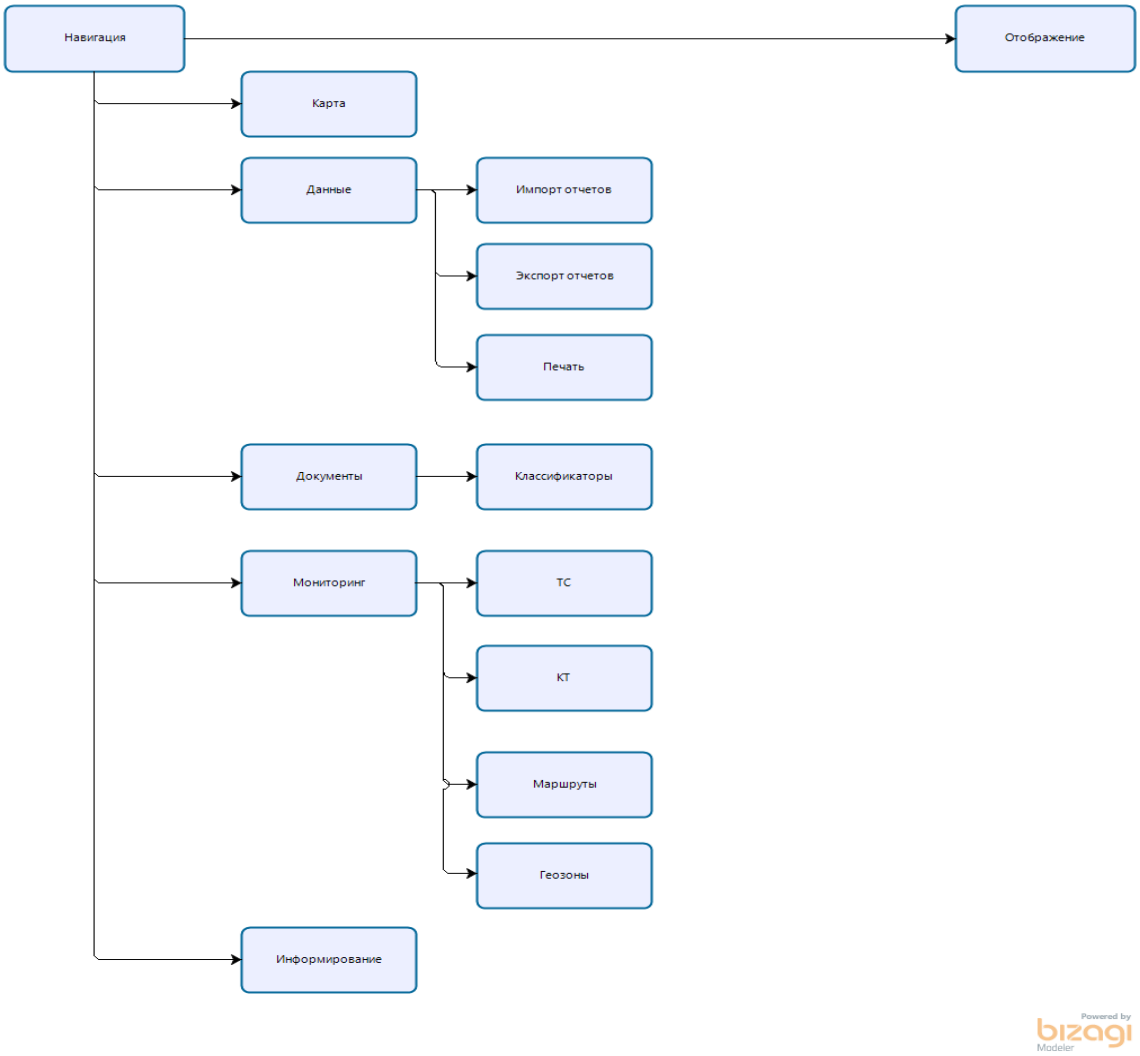

Рисунок 3.21 – Схема логики модуля «Регулятор»

Логика достаточно проста: если элемент навигации является категорией, то необходимо развернуть список инструментов, а если это непосредственно инструмент – перенаправить пользователя на необходимую страницу инструмента (рисунок 3.22). Т.к. страницы являются самостоятельными инструментами, маршрутизация (обработка HTTP-запросов) осуществляется с помощью Vue Router – официальной библиотеки маршрутизации для Vue.js [15].

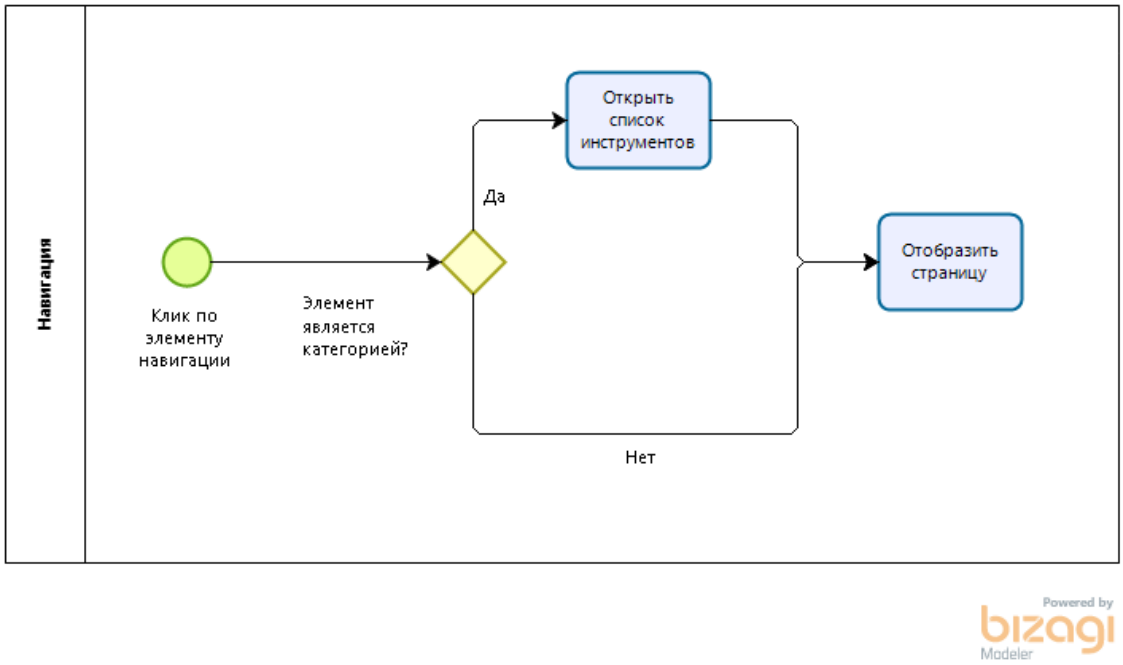

Рисунок 3.22 – Логика навигации

Так как данный модуль будет использоваться только для администраторов, было решено использовать библиотеку UI компонентов *Vuetify*.

Библиотеки UI – это коллекции таких компонентов, которые реализуют определенные рекомендации по стилю и предоставляют необходимые инструменты для построения многофункциональных веб-приложений. *Vuetify* использует концепцию *Material Design* – стиль графического дизайна интерфейсов программного обеспечения и приложений, разработанный компанией Google. Данный стиль используется в ОС *Android*, браузере *Google Chrome*, *YouTube* и другими продуктами компании.

Каждое приложение начинается с создания нового экземпляра *Vue* с помощью функции *Vue*:

```
var vm = new Vue({
  // опции
})
```

Переменную с экземпляром *Vue* традиционно именуют vm (сокращённо от *ViewModel*).

При создании экземпляра *Vue* необходимо передать объект опций.

Приложение *Vue* состоит из корневого экземпляра *Vue*, создаваемого с помощью *new Vue*, опционально организованного в дерево вложенных, повторно используемых компонентов.

Когда экземпляр *Vue* создан, он добавляет все свойства, найденные в опции *data*, в систему реактивности *Vue*. Поэтому представление будет «реагировать» на их изменения, обновляясь в соответствии с новыми значениями.

Каждый экземпляр *Vue* при создании проходит через последовательность шагов инициализации — например, настраивает наблюдение за данными, компилирует шаблон, монтирует экземпляр в DOM, обновляет DOM при изменении данных. Между этими шагами вызываются функции, называемые хуками жизненного цикла, с помощью которых можно выполнять свой код на определённых этапах. Ниже представлена диаграмма жизненного цикла экземпляра (рисунок 3.23):

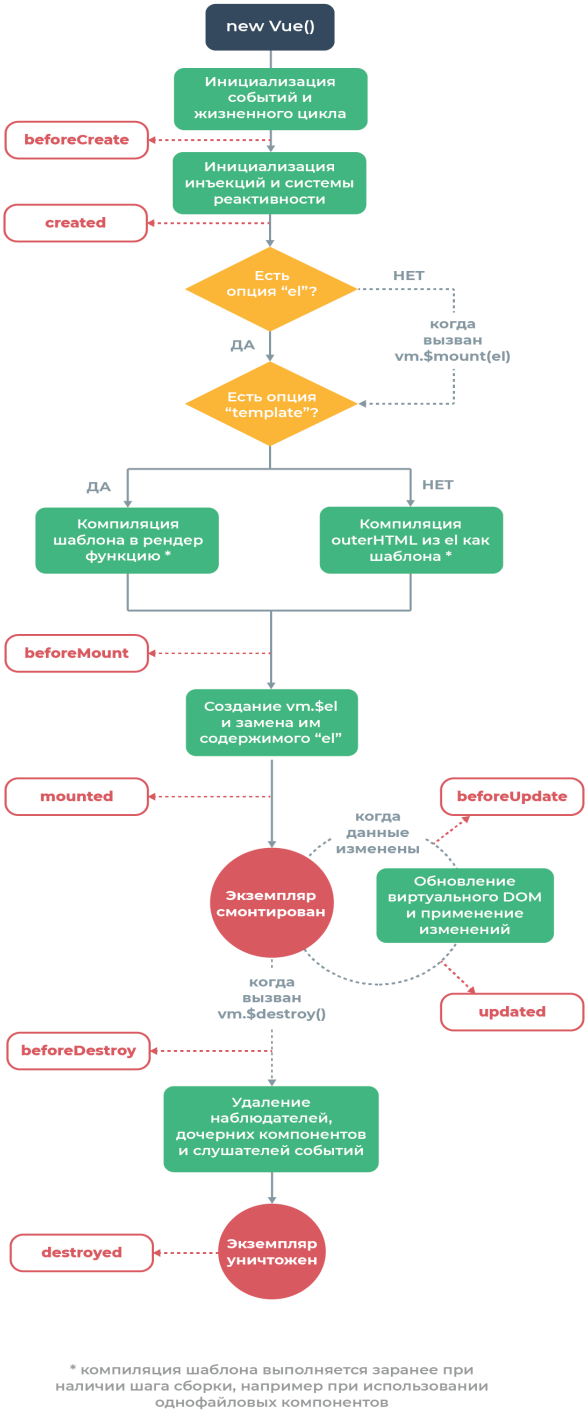

Рисунок 3.23 – Жизненный цикл экземпляра

Для связывания DOM с данными экземпляра *Vue* использует синтаксис, основанный на HTML. Все шаблоны *Vue* являются валидным HTML-кодом, который могут распарсить все HTML-парсеры и браузеры.

Для работы *Vue* компилирует шаблоны в render-функции виртуального DOM. В сочетании с системой реактивности, *Vue* умеет определять минимальное

число компонентов для повторной отрисовки и применяет минимальное количество манипуляций к DOM при изменении состояния приложения.

Наиболее простой способ связывания данных — это текстовая интерполяция с использованием синтаксиса Mustache (двойных фигурных скобок):

```
<span>Сообщение: {{ msg }}</span>
```

Выражение в фигурных скобках будет заменено значением свойства *msg* соответствующего объекта данных. Кроме того, оно будет обновлено при любом изменении этого свойства.

Значение выражения в двойных фигурных скобках подставляется как простой текст, а не как HTML. Для HTML необходимо использовать директиву *v-html*:

```
<p>Двойные фигурные скобки: {{ rawHtml }}</p>
<p>Директива v-html: <span v-html="rawHtml"></span></p>
```

Содержимое тега span будет заменено значением свойства *rawHtml*, интерпретированного как обычный HTML — все привязки данных игнорируются. Однако, мы не можем использовать *v-html* для вложения шаблонов друг в друга, потому что движок шаблонов *Vue* не основывается на строках. Вместо этого нужно использовать компоненты, позволяющие сочетать и переиспользовать элементы пользовательского интерфейса.

Синтаксис двойных фигурных скобок не работает с HTML-атрибутами. вместо него используется директива *v-bind* или сокращение вида «:*имя_директивы*»:

```
<div v-bind:id="dynamicId"></div>
```

Или

```
<div :id="dynamicId"></div>
```

При использовании с булевыми атрибутами (когда их наличие уже означает true) v-bind работает немного иначе. В этом примере:

```
<button:disabled="isButtonDisabled">Кнопка</button>
```

Если значением *isButtonDisabled* будет *null*, *undefined* или *false*, то атрибут *disabled* не добавится в элемент <button>

Само понятие директив будет рассмотрено ниже.

Директивы — это специальные атрибуты с префиксом «v-». В качестве значения они принимают одно выражение JavaScript. Директива реактивно применяет к DOM изменения при обновлении значения этого выражения. Например, директива «*v-if*» удалит или вставит элемент <p> в зависимости от истинности значения выражения *seen*.

```
<p v-if="seen">Сейчас меня видно</p>
```

Компоненты — это переиспользуемые экземпляры *Vue* со своим именем. Так как компоненты это переиспользуемые экземпляры *Vue*, то они принимают те же опции что и *new Vue*, такие как *data*, *computed*, *watch*, *methods*, хуки жизненного цикла.

Обычно приложение организуется в виде дерева вложенных компонентов (рисунок 3.24):

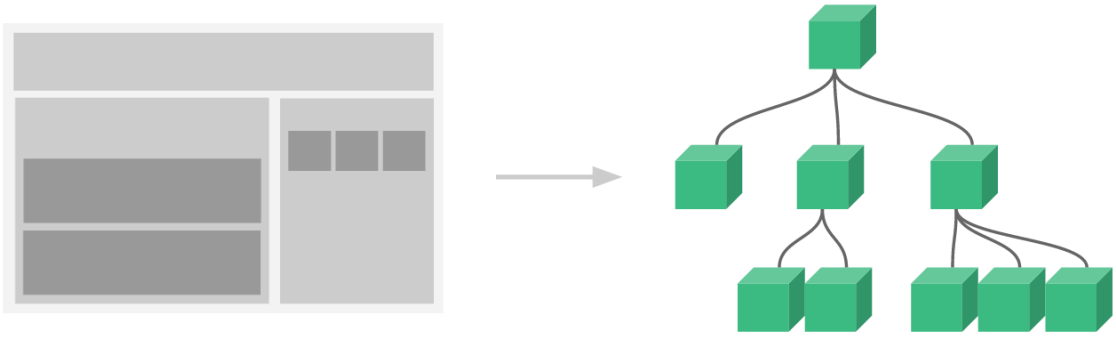

Рисунок 3.24 – Организация компонентов приложения

В нашем приложении будет использоваться локальная регистрация компонентов, чтобы снизить использование ресурсов на стороне клиента. В этом случае они будут определяться как объект Javascript и подключаться в объекте *components* компонента *Vue*. Так как в проекте используется сборщик *Webpack*, подключение компонентов будет выглядеть так:

```
import ComponentA from './ComponentA.vue'
export default {
  components: {
    ComponentA
  },
  // ...
}
```

Подробно о том, что такое *Webpack* описывать не имеет смысла, но сказать о том, что это инструмент разработчика, который выполняет очень большой объем операций, позволяющий сосредоточиться исключено на написании программного кода. Например, он переводит файлы с расширением *.vue* в HTML, если они являются страницами, объединяет javascript код в единый файл, проводит его обфускацию – уменьшает размер итогового файла, делает его

нечитабельным для человека, что усиливает защиту от взлома и кражи кода, и многое другое.

Сам по себе модуль «Регулятор» состоит из корневого модуля приложения – компонента *App.vue,* и компонента *AppBar.vue.*

Чтобы приложение начало работу, необходимо произвести первоначальную настройку. Для этого нам необходимо обратиться в *src/main.js*.

В данном файле мы подключим необходимые библиотеки для начала работы приложения, которые мы подробно рассмотрим чуть позже.

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import './plugins'
import vuetify from './plugins/vuetify'
import { sync } from 'vuex-router-sync'
import core from 'core-js'

sync(store, router)

Vue.config.productionTip = false
export const userLocationBus = new Vue()

new Vue({
  core,
  router,
  store,
  vuetify,
  render: h => h(App),
}).$mount('#app')
```

В данном файле мы указываем *App.vue* как основную страницу, подключаем маршрутизацию для перехода между страницами, хранилище состояния, библиотеку элементов пользовательского интерфейса, библиотеку обеспечения поддержки новейших свойств javascript для браузеров и инициализируем экземпляр *newVue,* куда передаем вышеуказанные данные в качестве опций и вызываем render-функцию. На данном этапе приложение полностью готово к дальнейшей разработке и работе.

Настройки элементов навигации находятся в файлах конфигурации библиотеки маршрутизации приложения *index.js* и *paths.js*.

Настройка маршрутизации выполнена таким образом, чтобы автоматически генерировать маршруты к файлам, указанным в файле *paths.js*.

```
/**
 * Vue Router
 *
 * @library
 *
 * https://router.vuejs.org/en/
 */

// Lib imports
import Vue from 'vue'
import VueAnalytics from 'vue-analytics'
import Router from 'vue-router'
import Meta from 'vue-meta'

// Routes
import paths from './paths'

function route (path, view, name) {
  return {
    name: name || view,
    path,
    component: resovle => import(`@/views/${view}.vue`).then(resovle),
  }
}

Vue.use(Router)

// Create a new router
const router = new Router({
  mode: 'history',
```

```
  routes: paths
    .map(path => route(path.path, path.view, path.name))
    .concat([{ path: '*', redirect: '/' }]),
  scrollBehavior (to, from, savedPosition) {
    if (savedPosition) {
      return savedPosition
    }
    if (to.hash) {
      return { selector: to.hash }
    }
    return { x: 0, y: 0 }
  },
})

Vue.use(Meta)

// Bootstrap Analytics
// Set in .env
// https://github.com/MatteoGabriele/vue-analytics
if (process.env.GOOGLE_ANALYTICS) {
  Vue.use(VueAnalytics, {
    id: process.env.GOOGLE_ANALYTICS,
    router,
    autoTracking: {
      page: process.env.NODE_ENV !== 'development',
    },
  })
}

export default router
```

Рассмотрим файл более конкретно в части, необходимой в корректной навигации между страницами приложения.

Сперва подключаем несколько библиотек, необходимых для конфигурации: Vue – корневая библиотека по управлению всем приложением, VueAnalytics – библиотека, необходимая для подключения метрик и телеметрии, Router – библиотека маршрутизации между страницами приложения, Meta – библиотека по управлению метаданными, сообщающие ту или иную информацию о странице браузеру, например, имя страницы, отображаемое во вкладке браузера и т.д.

Далее, происходит импорт массива объектов из файла *paths.js*, который содержит описание каждой из страниц приложения. Этот объект рассмотрим позже.

В функции route мы передаем в маршрутизатор информацию о страницах, импортированных из файла. Данная функция работает таким образом, что автоматически ищет файлы страниц папке *views* и подключает их в приложение, если они указаны в файле *paths.js*.

Paths.js, как уже было сказано ранее, содержит в себе массив объектов. Каждый объект представляет собой атрибуты страницы, такие как:

- path – путь, отображаемый в браузере;
- view – файл, содержащий в себе логику и разметку страницы;
- name – заголовок страницы, отображаемый в браузере.

```
/**
 * Define all of your application routes here
 * for more information on routes, see the
 * official documentation https://router.vuejs.org/en/
 */

export default [
  {
    path: '',
    // Relative to /src/views
    view: 'Client',
    name: 'Клиент',
  },
```

```
  {
    path: '/login',
    view: 'Login',
    name: 'Вход',
  },
  {
    path: '/monitor-car',
    name: 'Доступные транспортные стредства',
    view: 'Monitoring--Car',
  },
  {
    path: '/monitor-path',
    name: 'Маршруты',
    view: 'Monitoring--Path',
  },
  {
    path: '/monitor-ct',
    name: 'Управление контрольными точками',
    view: 'Monitoring--CT',
  },
  {
    path: '/classifiers',
    name: 'Классификаторы',
    view: 'Classifiers',
  },
  {
    path: '/geozones',
    name: 'Геозоны',
    view: 'Geozones',
  },
]
```

В *App.vue* происходит подключение нескольких компонентов, необходимых для функционирования приложения.

```
<template>
  <v-app>
    <core-app-bar v-if="$route.name !== 'К л и е н т' && $route.name !== 'В х о д'" />

    <core-drawer v-if="$route.name !== 'К л и е н т' && $route.name !== 'В х о д'" />

    <core-view />
  </v-app>
</template>

<script>
  export default {
    components: {
      CoreDrawer: () => import('@/components/core/Drawer'),
      CoreAppBar: () => import('@/components/core/AppBar'),
      CoreView: () => import('@/components/core/View'),
    },
    created: function () {
    },
  }
</script>

<style lang="scss">
html {
  overflow-y: auto !important;
}
</style>
```

В блоке <template /> находятся несколько компонентов. Навигация внутри приложении находится в компоненте *core-drawer*, отображение которого на странице происходит, если страница не содержит имя «Клиент» а так же имя «Вход».

В объекте components блока <script /> мы подключаем компоненты из блока <template />, которые являются функциями.

Таким образом, модуль «Регулятор» выглядит следующим образом (рисунок 3.25):

Рисунок 3.25 – Модуль «Регулятор»

Однако, рисунок выше не совсем соответствует тому, что описано в листингах кода, поэтому необходимо рассмотреть компонент core-drawer.

Данный компонент находится в файле *Drawer.vue*. На рисунке это можно выделить как блок в левой части экрана.

```
<template>
  <!-- color="grey darken-2" -->
  <v-navigation-drawer
    id="app-drawer"
    v-model="inputValue"
    style="z-index: 500"
    app
    dark
    floating
    mobile-break-point="991"
    persistent
    width="260"
  >
```

```
<template v-slot:img="attrs">
  <v-img
    v-bind="attrs"
    gradient="to top, rgba(0, 0, 0, .7), rgba(0, 0, 0, .7)"
  />
</template>

<v-list-item two-line>
  <v-list-item-avatar color="white">
    <v-img
      src="https://cdn.vuetifyjs.com/images/logos/v.png"
      height="34"
      contain
    />
  </v-list-item-avatar>

  <v-list-item-title class="title">
    AIS
  </v-list-item-title>
</v-list-item>

<v-divider class="mx-3 mb-3" />

<v-list nav>
  <!-- Bug in Vuetify for first child of v-list not receiving proper border-radius -->
  <div />
  <div
    v-for="(link, i) in links"
    :key="i"
    class="nav-links"
  >
    <v-list-group
      v-if="link.subMenu"
      no-action
      :prepend-icon="link.icon"
```

```
        >
          <template v-slot:activator>
            <v-list-item-title>{{ link.text }}</v-list-item-title>
          </template>
          <v-list-item
            v-for="(subLink, j) in link.subMenu"
            :key="j"
            :to="subLink.to"
            active-class="primary white--text"
          >
            <v-list-item-content>
              <v-list-item-title
                v-text="subLink.text"
              />
            </v-list-item-content>
          </v-list-item>
        </v-list-group>
        <v-list-item
          v-else
          :to="link.to"
          active-class="primary white--text"
        >
          <v-list-item-action>
            <v-icon>{{ link.icon }}</v-icon>
          </v-list-item-action>

          <v-list-item-title v-text="link.text" />
        </v-list-item>
      </div>
    </v-list>
  </v-navigation-drawer>
</template>

<script>
  // Utilities
```

```
import { mapMutations, mapState } from 'vuex'

export default {
  props: {
    opened: {
      type: Boolean,
      default: false,
    },
  },
  data: () => ({
    links: [
      {
        to: '/',
        icon: 'mdi-map-marker',
        text: 'К л и е н т с к а я  к а р т а',
      },
      {
        icon: 'mdi-database-edit',
        text: 'Д а н н ы е',
        subMenu: [
          {
            to: '#',
            text: 'И м п о р т  о т ч е т о в',
          },
          {
            to: '#',
            text: 'Э к с п о р т  о т ч е т о в',
          },
          {
            to: '#',
            text: 'П е ч а т ь',
          },
        ],
      },
      {
```

```
  icon: 'mdi-file-document-box-multiple',
  text: 'Документы',
  subMenu: [
    {
      to: '/classifiers',
      text: 'Классификаторы',
    },
  ],
},
{
  icon: 'mdi-laptop-chromebook',
  text: 'Мониторинг',
  subMenu: [
    {
      to: '/monitor-car',
      text: 'Транспортные средства',
    },
    {
      to: '/monitor-path',
      text: 'Маршруты',
    },
    {
      to: '/monitor-ct',
      text: 'Контрольные точки',
    },
    {
      to: '/geozones',
      text: 'Геозоны',
    },
  ],
},
{
  icon: 'mdi-alert-box',
  text: 'Информирование',
},
```

```
      ],
    }),

    computed: {
      ...mapState('app', ['image', 'color']),
      inputValue: {
        get () {
          return this.$store.state.app.drawer
        },
        set (val) {
          this.setDrawer(val)
        },
      },
    },

    methods: {
      ...mapMutations('app', ['setDrawer', 'toggleDrawer']),
      openSubMenu (e) {
        console.log(e)
      },
    },
  }
</script>
```

Блок <template /> содержит компонент *v-navigation-drawer* библиотеки компонентов *Vuetify*.

Каждая ссылка отправляется компонент *v-list-item* или в *v-list-item-group*, если это категория, согласно бизнес-процессу, указанному на рисунке 22. Сами ссылки и категории находятся в массиве *links* объекта *data*. Например, категория в массиве представляет собой объект со следующим набором атрибутов:

- icon – иконка категории;
- text – отображаемый текст;
- subMenu – массив объектов, описывающих ссылки.

Ссылки похожи на категории, однако не имеют *subMenu* и имеют *to* – атрибут, содержащий ссылку на страницу

3.4.3 Разработка модуля «Карта»

Карта – основа всей системы. Для подключения карты, в папке *views* создадим *Map.vue*. Код страницы выглядит следующим образом:

```
< <template>
  <l-map
    ref="osm"
    :style="$route.name === 'К л и е н т' ? clientStyle : adminStyle"
    :zoom="zoom"
    :center="initialLocation"
    :options="{ zoomControl: false }"
  >
    <l-marker
      ref="myGeo"
      :lat-lng="marker"
    />
    <slot name="cars-markers" />
    <slot name="cars-tracks" />
    <slot name="control-points" />
    <slot name="route-paths" />
    <l-tile-layer :url="url" />
  </l-map>
</template>

<script>
  import { mapGetters } from 'vuex'

  export default {
    data: () => ({
      zoom: 13,
      clientStyle: 'height: 100vh; width: 100%',
      adminStyle: 'height: calc(100vh - 75px); width: 100%',
      url: 'http://{s}.tile.osm.org/{z}/{x}/{y}.png',
      initialLocation: [45.044502, 41.969065],
      marker: [-100, -100],
    }),
    computed: {
      ...mapGetters(['geolocation']),
    },
    watch: {
      geolocation (newGeo, oldGeo) {
        if (newGeo !== this.initialLocation) {
```

```
            this.marker = this.$store.state.geolocation
            this.initialLocation = this.$store.state.geolocation
          }
        },
      },
      methods: {
        zoomUpdated (zoom) {
          this.zoom = zoom
        },
        centerUpdated (center) {
          this.initialLocation = center
        },
      },
    }
</script>
```

В блоке <template /> находится html-код и компоненты, которые фреймворк преобразует посредством javascript в html-код.

В блоке <script /> находятся настройки *Vue.js*. Конкретно нас интересует объект *data*, который возвращает ссылку на изображение карты, координаты маркера геопозиции, масштаб карты и координаты центра карты.

В блоке <style /> содержится css-код, который написан на препроцессоре scss.

Визуально, карта в модуле выглядит следующим образом (рисунок 3.26):

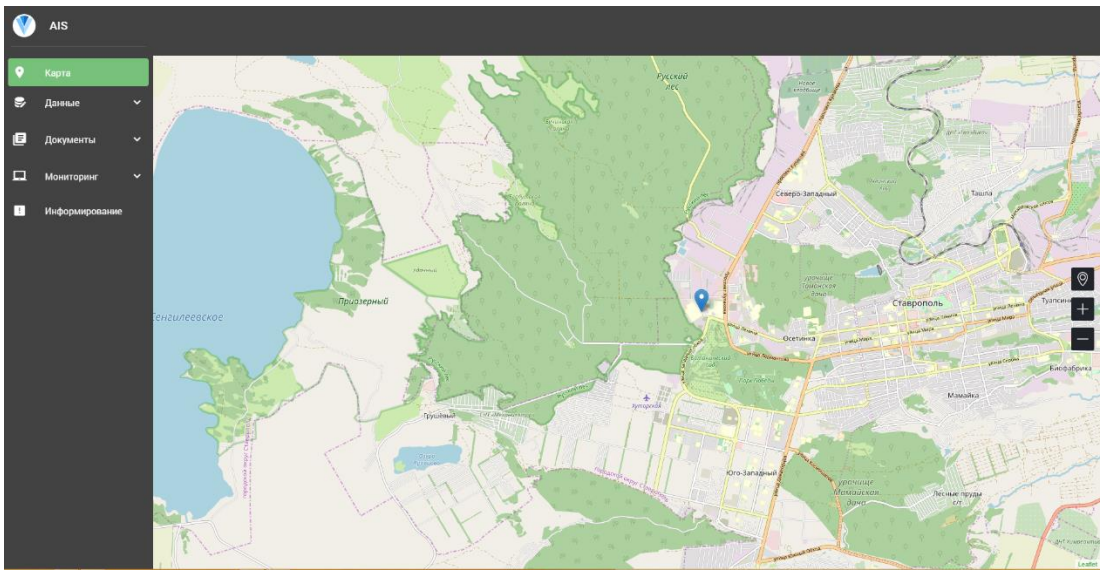

Рисунок 3.26 – Подключение карты в модуле «Регулятор»

Так как карта будет использоваться на многих страницах, то она будет сделана в виде компонента, включающего в себя другие компоненты:

1. *l-map* – компонент карты библиотеки *Leaflet*.
2. *l-marker* – компонент маркера библиотеки *Leaflet*.
3. *l-tile-layer* – компонент *Leaflet*, который принимает ссылку на изображения, которые составляют карту.

В объекте *props* находятся параметры, которые передаются от основного компонента дочерним, указанным выше. Эти данные проходят валидацию, согласно документации *Vue*. Так, в этих параметрах находится прямое указание, какой тип данных принимает каждый параметр: строка, массив или объект, или просто массив, а также значение по умолчанию. Значение по умолчанию необходимо для того, чтобы объекты без ожидания изменений выполняли свою задачу.

В модуле карты мы можем увидеть методы *zoomUpdated*, *centerUpdated*, вычисляемое свойство *geolocation*, данные опции будут описаны в следующем разделе, так как используют хранение состояний и связаны с компонентом элементов управления картой.

В блоке <template /> мы можем увидеть тег, который отсутствует в HTML – <slot />. Слоты предлагают удобный способ распространения контента из родительского компонента в дочерний компонент. Этот контент может быть любым из текста, HTML или даже других компонентов.

Иногда бывает полезно подумать о слотах, как о средствах инъекции контента непосредственно в шаблон дочернего компонента.

Слоты особенно полезны, когда состав компонентов под родительским компонентом не всегда одинаковый и модуль карты как раз попадает в этот случай, т.к. используется и на стороне клиента, и на некоторых страницах для регулятора и должен обладать разным набором инструментов, необходимых для выполнения тех или иных функций.

Так, слоты в модуле карты зарезервированы для следующих элементов:

1) маркеры машин;
2) треки маршрутов;
3) треки машин;

4) контрольные точки.

Данные слоты являются именованными, они работают аналогично отдельным слотам, но вместо этого позволяют распространять контент в разных частях в шаблоне дочернего компонента, что дает нам полный контроль необходимого содержимого в зависимости от нужд.

3.4.4 Управление состоянием приложения на примере компонента элементов управления картой

Сложность больших приложений нередко возрастает из-за распределения кусочков состояния по многим компонентам и связям между ними. Так, например, компоненты могут быть независимы друг от друга и использоваться как вместе на какой-либо странице и отдельно. В проекте такая ситуация возникает с компонентом карты на стороне клиента – пользователь должен иметь возможность использовать свое текущее местоположение как точку расчета маршрута. Однако, на стороне регулятора в определении местоположения нет необходимости. Ввиду этого было принято решение изолировать элементы управления картой отдельно от самой карты и не выделять для них слот. Однако, нужно сообщать приложению о том, что со стороны пользователя произошли изменения: изменилась геопозиция, изменен масштаб карты и т.п. Но ввиду того, что компоненты изолированы возникает проблема передачи данных из компонента в компонент. Для небольших приложений подойдет функциональность, которую предлагает фреймворк. Основа работы фреймворка — однонаправленный поток данных. Это значит, что данные из родительских компонентов передаются в дочерние через входные параметры, т.е. *props*, что и используется для компонента карты.

А для обратной связи наверх используются события (дочерние компоненты уведомляют о произошедшем событии и, возможно, передают какие-то данные). Но такой вариант нам не подходит, т.к. это предполагает размещение компонентов в одной странице и устанавливает прочную связь «родитель – потомок», а компонент карты не может быть установлен в такую

зависимость, т.к. пришлось бы создавать постоянно новую карту на каждой странице, где она необходимо. Это вызывает проблемы и избыточным размером приложения и возможным падением производительности, а так же нарушением одного из основных принципов программирования – DRY, что в переводе означает «не повторяй себя».

Для решения этой проблемы, *Vue* предлагает *Vuex*: собственную библиотеку управления состоянием.

Когда мы говорим о состоянии в компонентном приложении, то, как правило, оно имеет 2 типа:

- Локальное состояние компонента, ограниченное рамками одного компонента;
- Общее (глобальное) состояние приложения, доступное для всех компонентов приложения.

Для локального состояния характерен однонаправленый поток данных (рисунок 3.27), о что было сказано выше.

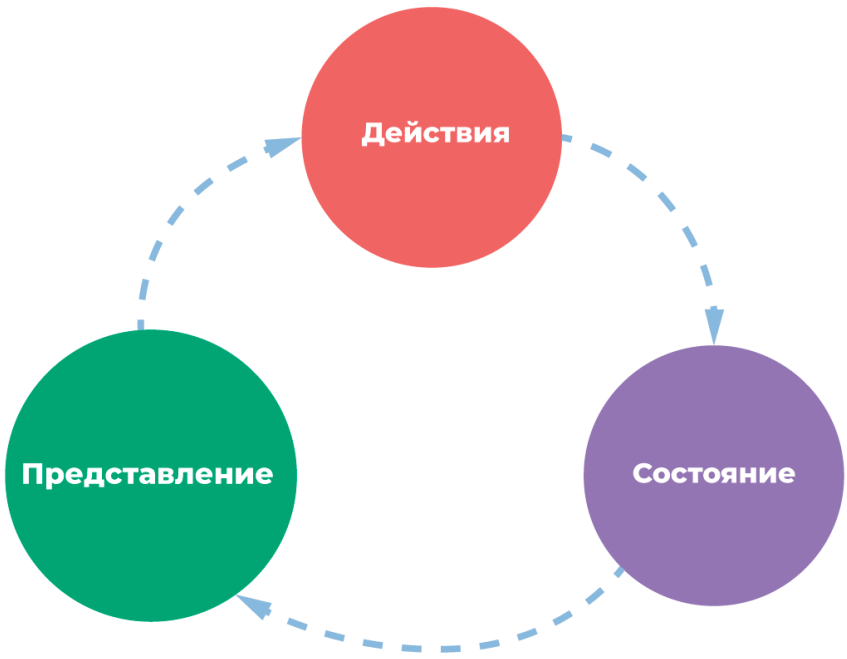

Рисунок 3.27 – Состояние приложения с однонаправленным потоком данных

Однако простота быстро исчезает, когда у нас появляется нескольких компонентов, основывающихся на одном и том же состоянии:

- Несколько представлений могут зависеть от одной и той же части состояния приложения;
- Действия из разных представлений могут оказывать влияние на одни и те же части состояния приложения.

Решая первую проблему, придётся передавать одни и те же данные входными параметрами в глубоко вложенные компоненты. Это часто сложно и утомительно, а для соседних компонентов такое и вовсе не сработает. Решая вторую проблему, можно прийти к таким решениям, как обращение по ссылкам к родительским/дочерним экземплярам или попыткам изменять и синхронизировать несколько копий состояния через события. Оба подхода хрупки и быстро приводят к появлению кода, который невозможно поддерживать.

Мы используем приложение как Single Page Application, где переход между страницами происходит без перезагрузки, соответственно, нам так же быстро необходимо передавать данные между компонентами и снизить количество запросов на сервер.

Исходя из этого было принято решение управлять состоянием приложения глобально. При этом наше дерево компонентов становится одним большим «представлением», и любой компонент может получить доступ к состоянию приложения или вызывать действия для изменения состояния, независимо от того, где они находятся в дереве.

Чётко определяя и разделяя концепции, возникающие при управлении состоянием, и требуя соблюдения определённых правил, которые поддерживают независимость между представлениями и состояниями, мы лучше структурируем код и облегчаем его поддержку (рисунок 3.28).

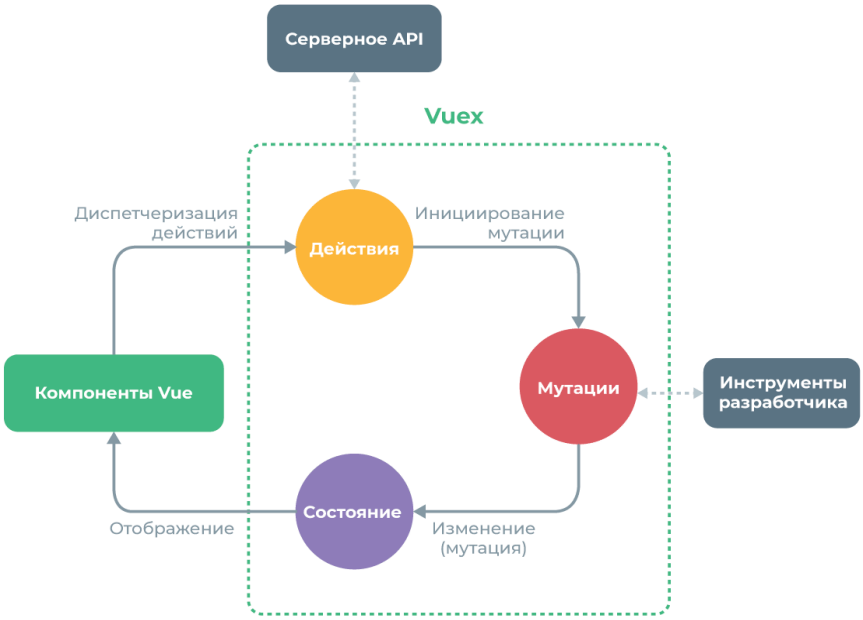

Рисунок 3.28 – Схема управления состояния приложения в глобальном пространстве

Создание хранилища приложения мы осуществим в *src/store/index.js* (рисунок 3.29)

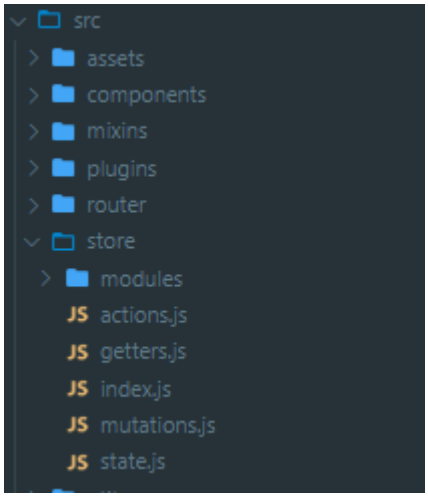

Рисунок 3.29 – Архитектура хранилища приложения

Содержимое выглядит следующим образом:

```
/**
```

```
 * Vuex
 *
 * @library
 *
 * https://vuex.vuejs.org/en/
 */

// Lib imports
import Vue from 'vue'
import Vuex from 'vuex'

// Store functionality
import actions from './actions'
import getters from './getters'
import modules from './modules'
import mutations from './mutations'
import state from './state'

Vue.use(Vuex)

// Create a new store
const store = new Vuex.Store({
  actions,
  getters,
  modules,
  mutations,
  state,
})

export default store
```

В данном файле мы подключаем указываем импорт самого фреймворка, библиотеки состояний и её составных частей, которые находятся в одноименных файлах:

1) Mutations – способом изменения состояния хранилища. Мутации во Vuex очень похожи на события: каждая мутация имеет строковый тип и функцию-обработчик. В этом обработчике и происходят, собственно, изменения состояния, переданного в функцию первым аргументом;

2) Getters – функции, кэширующие и возвращающие состояние хранилища. Геттеры являются вычисляемыми свойствами хранилища;

3) Actions – функции, вызывающие мутации и необходимые для асинхронных операций;

4) State – состояние, непосредственно хранилище.

На примере компонента элементов управления картой мы рассмотрим работу хранилища в нашем приложении (рисунок 3.30).

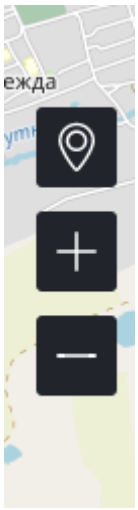

Рисунок 3.30 – Внешний вид компонента элементов управления картой: геолокация, увеличение и уменьшение масштаба карты

Код компонента:

```
<template>
  <div class="wrapper__controls_map">
    <button
      class="btn__control_map --locatecontrol"
      @click="findLocation"
    >
      <svg
        class="btn__control_map_svg"
        width="18"
        height="26"
        viewBox="0 0 18 26"
        fill="none"
        xmlns="http://www.w3.org/2000/svg"
      >
        <path
          class="is-filled"
          d="M9.0341 4.47754C11.9494 4.47754 14.3213 6.84932 14.3213 9.76463C14.3213 12.6799 11.9494 15.0517 9.0341 15.0517C6.11879 15.0517 3.74691 12.6799 3.74691 9.76463C3.74691 6.84932 6.11879 4.47754 9.0341 4.47754ZM9.0341 13.6295C11.1653 13.6295 12.8992 11.8957 12.8992 9.76454C12.8992 7.63337 11.1653 5.89964 9.0341 5.89964C6.90294 5.89964 5.16901 7.63347 5.16901 9.76463C5.16901 11.8958 6.90284 13.6295 9.0341 13.6295Z"
          fill="white"
        />
```

```
        <path
          class="is-filled"
          d="M9.03449 0.799072C13.9781 0.799072 18 4.82097 18 9.76458C18 12.6156 15.609 16.8632 14.1837 19.1414C13.246 20.6402 12.2383 22.0666 11.3464 23.1579C9.91468 24.9093 9.40225 25.0559 9.0344 25.0559C8.66105 25.0559 8.17725 24.9092 6.75003 23.1568C5.85609 22.059 4.84792 20.6334 3.91113 19.1425C2.47613 16.8588 0.0688934 12.6036 0.0688934 9.76448C0.0688934 4.82097 4.09079 0.799072 9.03449 0.799072ZM4.89703 18.0352C6.7359 21.02 8.42413 23.037 9.04293 23.5519C9.66496 23.0528 11.3258 21.0839 13.1904 18.0446C15.3431 14.5354 16.5778 11.5174 16.5778 9.76458C16.5779 5.60512 13.194 2.22118 9.03449 2.22118C4.87503 2.22118 1.49109 5.60512 1.49109 9.76458C1.491 11.5073 2.73249 14.5218 4.89703 18.0352Z"
          fill="white"
        />
      </svg>
    </button>
    <button
      class="btn__control_map"
      @click="zoomIn"
    >
      <svg
        class="btn__control_map_svg"
        width="20"
        height="20"
        viewBox="0 0 20 20"
        fill="none"
        xmlns="http://www.w3.org/2000/svg"
      >
        <path
          class="is-filled"
          d="M19.2188 9.21875H10.7812V0.78125C10.7812 0.349766 10.4315 0 10 0C9.56852 0 9.21875 0.349766 9.21875 0.78125V9.21875H0.78125C0.349766 9.21875 0 9.56852 0 10C0 10.4315 0.349766 10.7812 0.78125 10.7812H9.21875V19.2188C9.21875 19.6502 9.56852 20 10 20C10.4315 20 10.7812 19.6502 10.7812 19.2188V10.7812H19.2188C19.6502 10.7812 20 10.4315 20 10C20 9.56852 19.6502 9.21875 19.2188 9.21875Z"
          fill="white"
        />
      </svg>
    </button>
    <button
      class="btn__control_map"
      @click="zoomOut"
    >
      <svg
        class="btn__control_map_svg"
        width="20"
        height="2"
        viewBox="0 0 20 2"
        fill="none"
        xmlns="http://www.w3.org/2000/svg"
      >
        <path
          class="is-filled"
```

```
          d="M19.2188 0.21875H0.78125C0.349766 0.21875 0 0.568516 0 1C0 1.43148 0.349766 1.
78125 0.78125 1.78125H19.2188C19.6502 1.78125 20 1.43148 20 1C20 0.568516 19.6502 0.21875 1
9.2188 0.21875Z"
          fill="white"
        />
      </svg>
    </button>
  </div>
</template>

<script>
  import { mapGetters } from 'vuex'

  export default {
    data: () => ({
      name: 'controls',
    }),
    computed: {
      ...mapGetters(['geolocation']),
    },
    watch: {
      geolocation (newGeo, oldGeo) {
        if (newGeo) {
          this.findLocation()
        }
      },
    },
    methods: {
      zoomIn () {
        this.$parent.$children[0].$refs.osm.mapObject.zoomIn()
      },
      zoomOut () {
        this.$parent.$children[0].$refs.osm.mapObject.zoomOut()
      },
      findLocation () {
        const self = this
        if (navigator.geolocation) {
          const cordsArray = []
          navigator.geolocation.getCurrentPosition((position) => {
            const geoposition = position.coords
            cordsArray.push(geoposition.latitude, geoposition.longitude)
            self.$store.commit('setGeolocation', cordsArray)
          })
        }
      },
    },
  }
</script>

<style lang="scss">
.wrapper {
```

```
  &__controls {
    &_map {
      position: absolute;
      top: 50%;
      right: 10px;
      display: flex;
      flex-direction: column;
      z-index: 400;
      transform: translateY(-50%);
    }
  }
}

.btn {
  &__control {
    &_map {
      display: flex;
      justify-content: center;
      align-items: center;
      width: 40px;
      height: 40px;
      margin-bottom: 12px;
      border-color: #21242b;
      background: #21242b;
      border-radius: 3px;

      &:hover {
        .btn__control_map_svg {
          .is-filled {
            fill: #ffccb2;
          }
        }
      }

      &:focus {
        .btn__control_map_svg {
          .is-filled {
            fill: #ff7733;
          }
        }
      }
    }
  }
}
</style>
```

Тут необходимо заострить внимание на нескольких вещах. В компонент мы импортируем функцию возврата геолокации, а также отслеживаем её. То же самое мы используем в модуле карты.

В обобщенном виде это выглядит следующим образом для модуля карты:

```
<script>
  import { mapGetters } from 'vuex'

  export default {
    computed: {
      ...mapGetters(['geolocation']),
    },
    watch: {
      geolocation (newGeo, oldGeo) {
        if (newGeo !== this.initialLocation) {
          this.marker = this.$store.state.geolocation
          this.initialLocation = this.$store.state.geolocation
        }
      },
    },
  }
</script>
```

И для компонента элементов управления картой:

```
<script>
  import { mapGetters } from 'vuex'

  export default {
    computed: {
      ...mapGetters(['geolocation']),
    },
    watch: {
      geolocation (newGeo, oldGeo) {
        if (newGeo) {
          this.findLocation()
        }
      },
    },
    methods: {
      findLocation () {
        const self = this
        if (navigator.geolocation) {
          const cordsArray = []
          navigator.geolocation.getCurrentPosition((position) => {
```

```
            const geoposition = position.coords
            cordsArray.push(geoposition.latitude, geoposition.longitude)
            self.$store.commit('setGeolocation', cordsArray)
          })
        }
      },
    },
  }
</script>
```

Функция *mapGetters* просто передает геттеры хранилища в локальные вычисляемые свойства компонента, конкретно в нашем случае *geolocation* и данный геттер выглядит следующим образом:

```
export default {
  geolocation: state => state.geolocation
}
```

Как видно из кода выше, он обращается в хранилище, чтобы получить состояние *geolocation*. По умолчанию, данный геттер вернет из хранилища *state.js* пустой массив:

```
export default {
  geolocation: []
}
```

Логика определения местоположения отображена на рисунке 17. В программном виде за это отвечает метод *findLocation.*

```
findLocation () {
        const self = this
        if (navigator.geolocation) {
          const cordsArray = []
          navigator.geolocation.getCurrentPosition((position) => {
            const geoposition = position.coords
            cordsArray.push(geoposition.latitude, geoposition.longitude)
            self.$store.commit('setGeolocation', cordsArray)
          })
        }
      },
```

Данный метод срабатывает, если у устройства есть аппаратные средства геопозиционирования, получает широту и долготу и помещает во временный массив *cordsArray* и записывает этот массив в хранилище как значение *geolocation*.

В *watch* у обоих компонентов мы следим за состоянием местоложения пользователя. Если в компоненте элементов управления картой при смене местоположения вызывается метод *findLocation*, то в модуле карты – запись в компонент координат местоположения из хранилища, центрирование карты по координатам и установку в эти координаты маркера

3.4.5 Разработка компонента «Контрольная точка»

Компонент «Контрольная точка» позволяет пользователю АИС, наделенному специальными правами, создавать точки на карте, которые будут содержать информацию, необходимую для составления маршрутов по совокупности таких точек.

Код компонента выглядит следующим образом:

```
<template>
  <div class="point-buble__wrapper">
    <v-card
      dark
      class="point-buble__card"
    >
      <v-form @submit.prevent="createMarker($event)">
        <v-container fluid>
          <v-row>
            <v-col cols="12">
              <span v-
if="isChangeMode">И з м е н е н и е  т о ч к и  м а р ш р у т а</span>
              <span v-else>С о з д а н и е  т о ч к и  м а р ш р у т а</span>
            </v-col>
            <v-col cols="12">
              <v-row>
                <v-col
                  v-for="(input, i) in inputs"
                  :key="i"
                  cols="12"
```

```
        :md="input.isGeozone ? '12' : '4'"
      >
        <template v-if="input.isGeozone">
          <v-text-field
            v-model="point.geozone"
            dark
            disabled
            :label="input.label"
          />
        </template>
        <template v-else-if="!input.isSelect">
          <v-text-field
            v-model="point[input.name]"
            dark
            :label="input.label"
          />
        </template>
        <template v-else-if="input.isSelect">
          <v-select
            v-model="point[input.name]"
            :value="point[input.name]"
            dark
            :items="input.options"
            :label="input.label"
          />
        </template>
      </v-col>
    </v-row>
  </v-col>
  <v-col cols="12">
    <div class="point-buble__footer">
      <button
        class="point-buble__button"
        type="submit"
        data-action="submit"
      >
        О к
      </button>
      <button
        v-if="isChangeMode"
        class="point-buble__button"
        type="button"
        data-action="delete"
        @click="deleteMarker()"
      >
        У д а л и т ь
      </button>
      <button
        class="point-buble__button"
        type="button"
        data-action="cancel"
```

```
                @click="closeMarker()"
              >
                Отмена
              </button>
            </div>
          </v-col>
        </v-row>
      </v-container>
    </v-form>
  </v-card>
 </div>
</template>

<script>
 import axios from 'axios'
 import { mapGetters } from 'vuex'

 export default {
  name: 'Buble',
  props: {
   isChangeMode: {
    type: Boolean,
    default: false,
   },
   point: {
    type: Object,
    default: () => ({
     name: '',
     type: '',
     direction: '',
     longtitude: 0,
     latitude: 0,
     radius: 0,
     address: '',
     id: null,
    }),
   },
  },
  data () {
   return {
    inputs: [
     {
      name: 'name',
      label: 'Имя остановки',
      isSelect: false,
     },
     {
      name: 'type',
      label: 'Тип КТ',
      isSelect: true,
      options: ['Остановка', 'Участок дороги'],
```

```
          },
          {
            name: 'direction',
            label: 'Направление',
            isSelect: true,
            options: ['Прямое', 'Обратное'],
          },
          {
            name: 'longitude',
            label: 'Широта',
            isSelect: false,
          },
          {
            name: 'latitude',
            label: 'Долгота',
            isSelect: false,
          },
          {
            name: 'radius',
            label: 'Радиус',
            isSelect: false,
          },
          {
            name: 'address',
            label: 'Адрес',
            isSelect: false,
          },
          {
            name: 'geozone_id',
            label: 'Геозона',
            isSelect: false,
            isGeozone: true,
          },
        ],
        geozones: [],
      }
    },
    computed: {
      ...mapGetters(['sessionID']),
    },
    created: function () {},
    methods: {
      createMarker: function (ev) {
        console.log(this.point)
        const uri = 'http://194.58.104.20/control_points' + (this.point.id === -
2 ? '' : '/' + this.point.id)
        axios({
          method: this.point.id === -2 ? 'post' : 'put',
          url: uri,
          data: JSON.stringify({
            name: this.point.name,
```

```
            type: this.point.type === 'Остановка' ? 0 : 1,
            latitude: this.point.latitude,
            longitude: this.point.longitude,
            radius: this.point.radius,
            address: this.point.address,
            direction_id: this.point.direction === 'Прямое' ? 0 : 1,
            geozone_id: 0,
          }),
          headers: {
            'Content-Type': 'application/json',
          },
        })
          .then(res => {
            this.$emit('close')
          })
      },
      closeMarker: function () {
        this.$emit('close')
      },
      deleteMarker: function () {
        axios
          .delete(`http://194.58.104.20/control_points/${this.point.id}`)
          .then(res => {
            console.log(res)
            if (res.status === 204) {
              this.$emit('refresh')
            }
          })
      },
    },
  }
</script>

<style lang="scss" scoped>
.point {
  &-buble {
    &__wrapper {
      //позиционирование только для демо режима
      z-index: 500;
      border-radius: 10px;
      overflow: hidden;
    }
    &__card {
      border-radius: 10px;
      width: 100%;
      height: 100%;
      font-size: 17px;
    }

    &__footer {
      display: flex;
```

```
      justify-content: center;
      align-items: center;
      width: 100%;
    }

    &__button {
      display: flex;
      justify-content: center;
      align-items: center;
      width: 100%;
      max-width: 180px;
      height: 40px;
      margin: 10px 5px 0 5px;
      border-radius: 3px;
      color: #fff;
      border: 1px solid transparent;
      background: #ff884d;

      &:hover {
        cursor: pointer;
        background-color: #ff7733;
      }

      &:last-child {
        background: #20242b;
        border-color: #ff884d;

        &:hover {
          cursor: pointer;
          background: #303540;
          border-color: #ff7733;
        }
      }
    }
  }
}
</style>
```

В блоке <template /> находится html-код и компоненты, которые фреймворк преобразует посредством javascript в html-код.

В блоке <script /> находятся настройки vuejs. Объект *data* содержит массив объектов *inputs*, которые отвечает за именование полей и их атрибуты, входящие в состав модуля. Каждые объект содержит следующие атрибуты:

– *label* – имя поля, выводимое в интерфейсе;

– *name* – имя поля, необходимое для обращения к API сервера и рендеринга необходимой версии компонента поля ввода;

– *isSelect* – булевое значение, которое указывает приложению, является ли поле ввода выпадающим списком с опциями.

Дополнительные возможные атрибуты:

options – необходимое указать как массив строк, если *isSelect* принимает значение *true.*

Объект props содержит информацию для динамических данных.

Так, параметр *isChangeMode* объекта *props* устанавливает, что параметры, приходящие с этим именем данные переводят компонент в режим редактирования, а также, что данные обязательно должны быть булевого типа и по умолчанию режим редактирования выключен.

Параметр *point* объекта *props* устанавливает, что входящие данные с данным именем должны является объектом и по умолчанию иметь следующие атрибуты:

– *name* – имя точки, строка;

– *type* – тип точки, строка;

– *direction* – направление точки (прямое, обратное), строка;

– *longtitude* – широта, число;

– *latitude* – долгота, число;

– *radius* – радиус контрольной точки, число;

– *address* – адрес точки, строка;

– *id* – идентификатор точки в базе данных, число.

В блоке <style /> содержится css-код, который написан на препроцессоре scss.

Визуально, компонент при интеграции с картой выглядит следующим образом (рисунок 3.33):

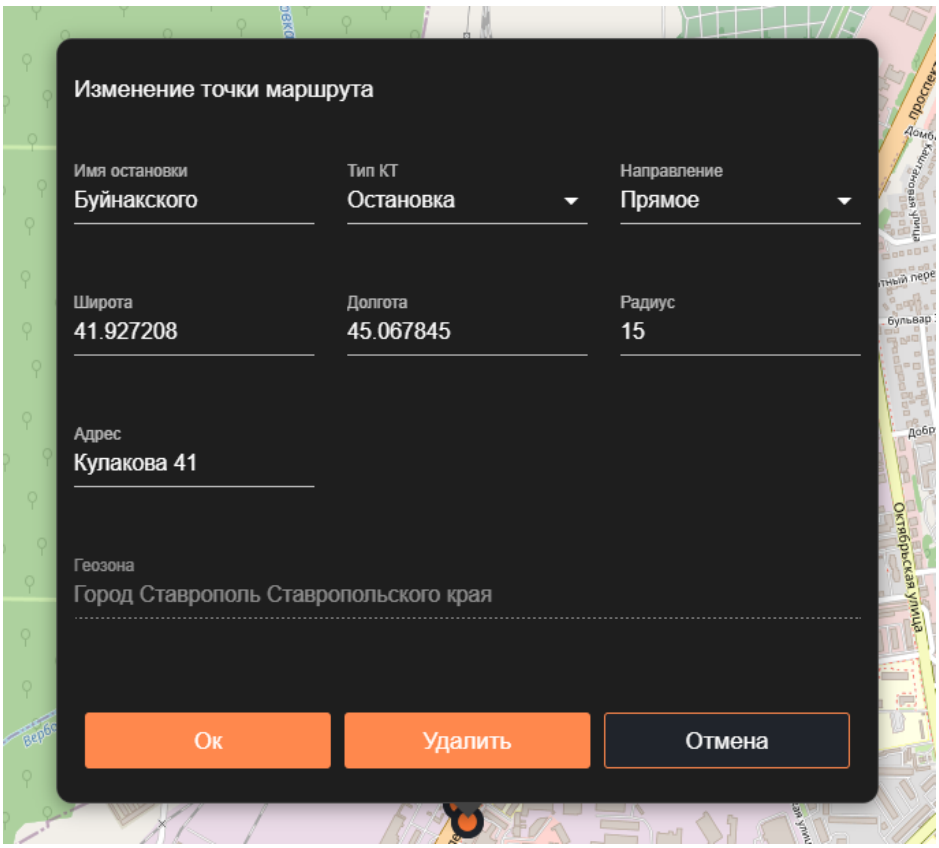

Рисунок 3.33 - Компонент «Контрольная точка» в режиме редактирования

3.4.6 Разработка компонента «Маршруты»

Маршруты являются одной из масштабных функций, которую предоставляет модуль «Регулятор». При разработке данного компонента, необходимо разрешить большой круг задач:

1) Интерфейс, который будет вмещать список контрольный точек, маршрутов и не мешает отображению контрольных точек и маршрутов на карте;

2) Разработать механизм объединения контрольных точек в маршруты;

3) Каким образом наносить маршруты на карту;

4) Организация клиент-серверного общения для получения, сохранения, удаления и редактирования маршрутов.

Ввиду того, что составной единицей маршрута является контрольная точка, было принято решение вынести список данных точек в левую часть экрана

и объединить их по территориальному признаку (3.34). То же самое сделать и для маршрутов.

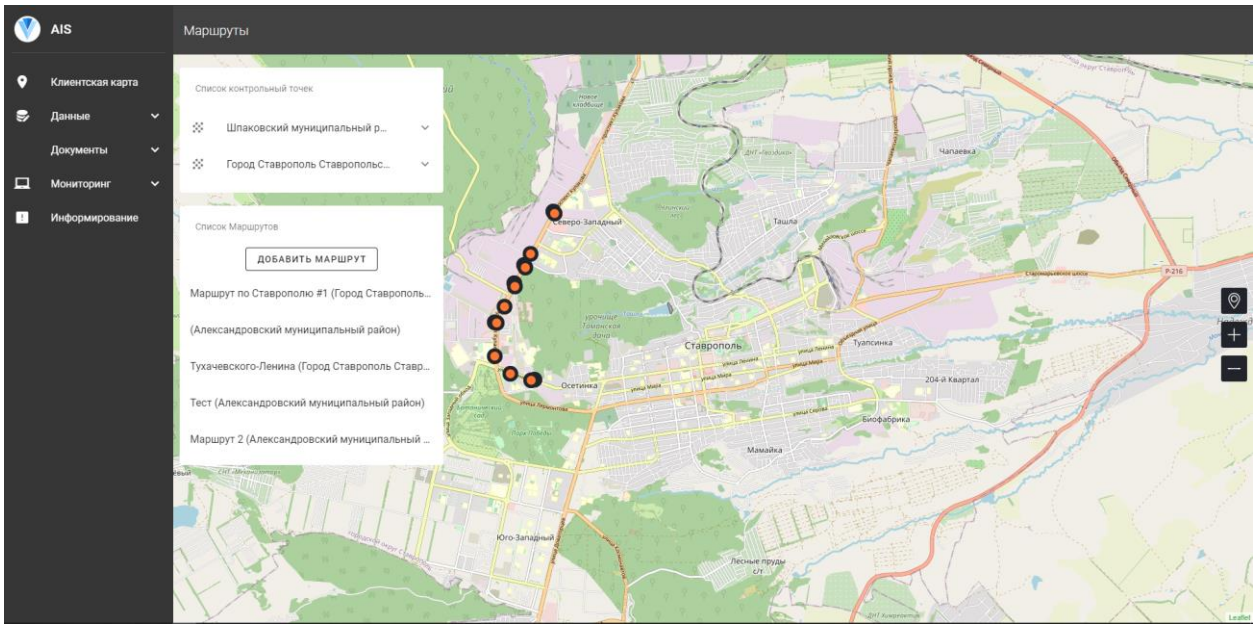

Рисунок 3.34 – Компонент «Маршруты» со списком маршрутов и контрольных точек

Данные списки работают по аналогичному принципу с меню в модуле «Регулятор».

На рисунке выше видны некоторые готовые тестовые маршрутки и контрольные точки, с которым можно работать.

Прежде чем создать маршрут, на необходимо иметь контрольные точки. Ввиду того, что контрольные точки являются остановками, это поставило дополнительную задачу по представлению прямого и обратного направления маршрута и вследствие этого в компоненте контрольной точки существует опция, отображающая тип направления (рисунок 3.35).

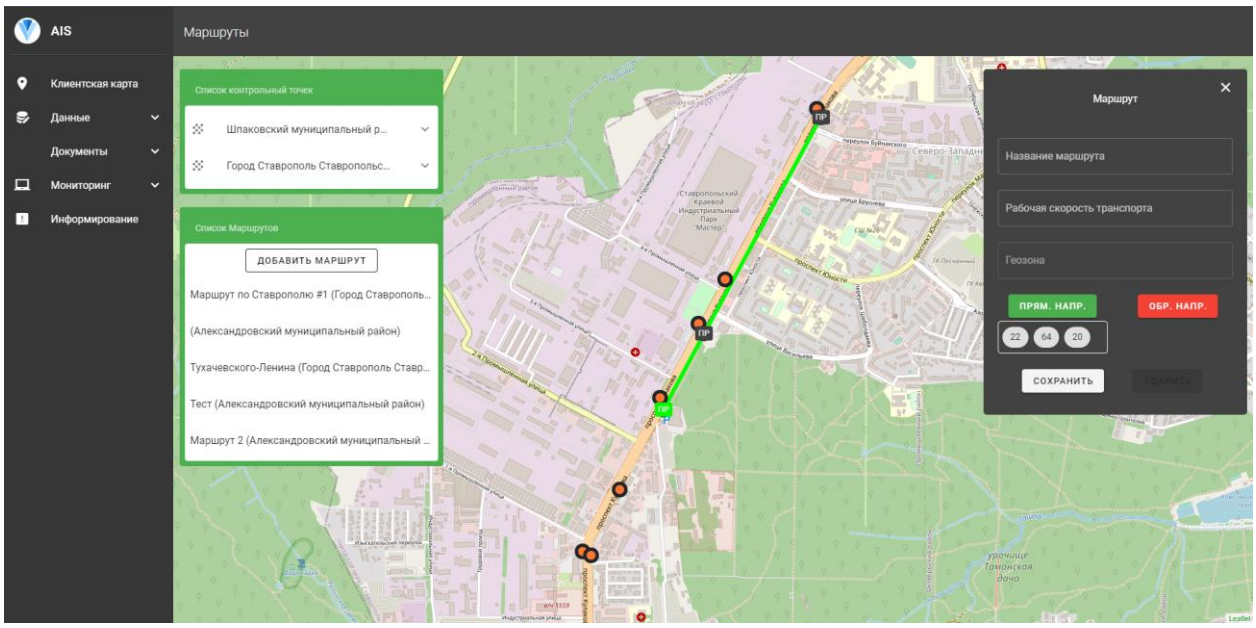

Рисунок 3.35 – Компонент «Маршруты» в режиме составления маршрута прямого направления

Чтобы перейти в режим создания маршрута, нам необходимо нажать на кнопку «Добавить маршрут». Далее, в правой части экрана появляется панель с настройками маршрута, где можно указать его имя, установить рабочую скорость ТС, привязать его к геозоне и выбрать направления.

После этого необходимо нажать на желаемый тип направления, на рисунке 30 – прямое направление, и поочередно нажимать на контрольные точки. В случае, если точки связаны между собой, они соединяются отрезком, контрольная точка заменяется на значок «ПР», последняя выделенная точка имеет значок «ПР» на зеленом фоне. Под идентификатором направления выводится блок с идентификатор точки, с которым она хранится на сервере.

В случае, если выбрано направление обратное, будет то же самое, но с тем отличием, что панели слева, отрезок между точками и фон активной контрольной точки будет иметь красный фон. Так же, клик по отрезку маршрута переключает редактирование маршрута в соответствующий режим, а удаление точек маршрута осуществляется повторным кликом по последней выбранной контрольной точке, чтобы исключить вероятность ошибочного построения маршрута, который будет проиллюстрирован ниже (рисунок 3.36):

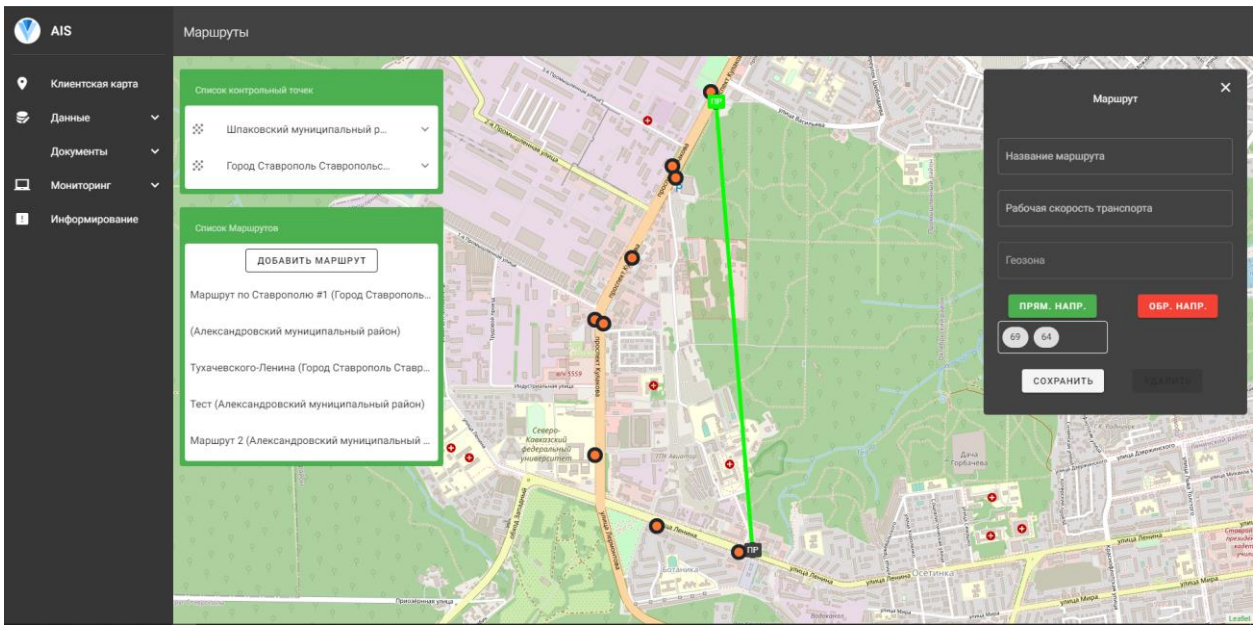

Рисунок 3.36 – Неверное построение маршрута между контрольными точками

Конечный вид готового маршрута имеет следующий вид (рисунок 3.37):

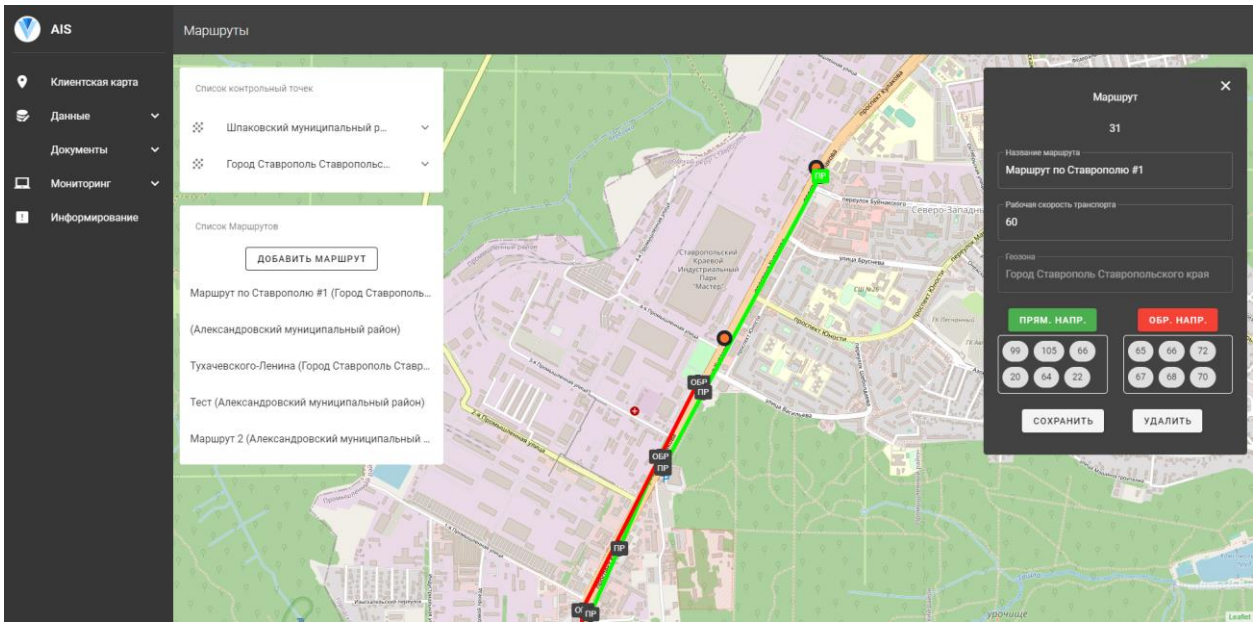

Рисунок 3.37 – Компонент «Маршруты» в режиме составления или просмотра готового маршрута

При просмотре готово маршрута в панели настроек над его название выводится идентификатор маршрута, по с которым он хранится на сервере.

Контрольные точки и отрезки маршрутов помещаются на карту с помощью директивы *v-slot*, которая была рассмотрена выше. При разработке данного модуля появилась необходимость размещения динамических элементов на карту и именно следствием этой необходимости стало внедрение именованных слотов в модуль карты.

Код разметки, отвечающий за отображение контрольных точек и отрезков маршрутов:

```
<core-map>
  <template v-slot:control-points>
    <l-marker
      v-for="(point, i) in points"
      :key="i"
      :lat-lng="[point.latitude, point.longitude]"
      :options="{title: `${point.name} (${point.direction_id ? 'О б р а т н о е' : 'П р я м о е'}) (${point.id})`}"
      @click="addPointToRoute(point)"
    >
      <l-icon
        :icon-anchor="pointMarkerAnchor"
      >
        <span
          v-if="choosenPath.straight_direction.find(x => x.control_point_id === point.id)"
          :class="{
            pointDir: true,
            str: true,
            sMain: choosenPath.straight_direction.findIndex(x => x.control_point_id === point.id) ===
              choosenPath.straight_direction.length - 1}"
        >
          П Р
        </span>
        <span
          v-if="choosenPath.reverse_direction.find(x => x.control_point_id === point.id)"
          :class="{
            pointDir: true,
            rvrs: true,
            rMain: choosenPath.reverse_direction.findIndex(x => x.control_point_id === point.id) ===
              choosenPath.reverse_direction.length - 1}"
        >
          О Б Р
        </span>
        <img
          width="24"
```

```
          src="../assets/gps.svg"
          alt="marker"
        >
      </l-icon>
    </l-marker>
  </template>
  <template v-slot:route-paths>
    <l-polyline
      :lat-lngs="straightPath"
      :color="'#07fc03'"
      :weight="6"
      @click="changeMode(1)"
    />
    <l-polyline
      :lat-lngs="reversePath"
      :color="'#ff0000'"
      :weight="6"
      @click="changeMode(2)"
    />
  </template>
</core-map>
```

Однако, работа данного модуля была бы невозможна без обращений к API сервера.

Для этого в приложения используется библиотека *Axios* – библиотека HTTP запросов, которая поддерживает последние стандарты веб-разработки. Библиотека основывается на *Promise* – это специальный объект, который содержит своё состояние. Вначале *pending* («ожидание»), затем – одно из: *fulfilled* («выполнено успешно») или *rejected* («выполнено с ошибкой»). Это означает, что мы можем выполнять запросы асинхронно, т.е. работать с приложением в ожидании ответа от сервера, а в случае ошибки приложение не будет остановлено и продолжит работу с имеющимся набором данных.

Способ использования, в общих чертах, такой:

1. Код, которому надо сделать что-то асинхронно, создаёт объект *promise* и возвращает его;

2. Внешний код, получив *promise*, подключает к нему обработчики событий;

3. По завершении процесса асинхронный код переводит *promise* в состояние *fulfilled* (с результатом) или *rejected* (с ошибкой). При этом автоматически вызываются соответствующие обработчики во внешнем коде.

Если проводить аналогии не с программированием, то это можно объяснить как подписку на рассылку – мы знаем, что получим уведомление о чем-либо по готовности ответа, как только подпишемся на рассылку и находимся в состоянии ожидания.

Подключение библиотеки в проекте мы осуществили на старте в *main.js*, а сама библиотека находится в папке *plugins/axios.js*.

В компоненте маршрутов рассмотрим подключение и пример запроса к API сервера:

```
<script>
  // ...
  import axios from 'axios'
  // ...

  export default {
    // ...
    created: function () {
      axios
        .get('http://194.58.104.20/geozones')
        .then(res => {
          console.log(res)
          this.geozones = res.data
        }).then(() => {
          this.reloadPoints()
          this.reloadPaths()
        })
    },
    // ...
  }
</script>
```

В данном участке кода, мы имопртируем библиотеку компонент, чтобы к ней в даленйшем обращаться. В хуке жизненного цикла компонента *created*, который выполняется как только готова объектная модель HTML-страницы (DOM), но еще не отображена пользователю, мы выполняем запрос к API сервера

методом *.get()* объекта *axios*. Затем, как только получен ответ от сервера, вы выводим его в консоль браузера с помощью *console.log()*.

Ответ от сервера имеет следующий вид (рисунок 3.38):

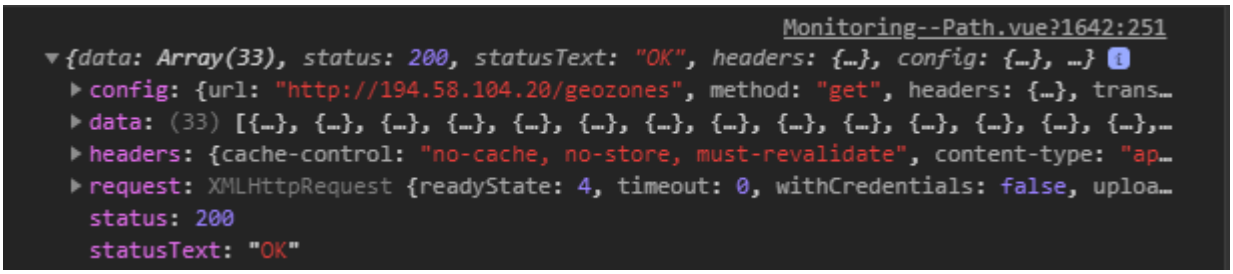

Рисунок 3.38 – Ответ от сервера в консоли браузера

И в качестве примера приведу тело ответа для данного запроса, содержащее 1 геозону:


```
[
   {
      "id": 1,
      "name": "Александровский муниципальный район"
   },
]
```


Если ответ получен, мы сохраняем тело ответа в переменную и выполняем загрузку контрольных точек и маршрутов.

Мы получаем список геозон, так как маршруты и контрольные точки не могут существовать сами по себе, а должны иметь привязку к географическому объекту, например, городу или области.

4.1 Предпроектное исследование модуля составления рассписания пассажирских перевозок

4.1.1 Алгоритмы составления расписания для городского транспорта

Одним из важных критериев при организации работы распределенной вычислительной системы является планирование выполнений заданий. То есть необходимо составить расписание, в котором будет указываться, какие задания на каких узлах и в какое время выполняются.

Теория расписаний – это раздел исследования операций, в котором строятся и анализируются математические модели календарного планирования (т.е. упорядочивания во времени) различных целенаправленных действий с учетом целевой функции и различных ограничений.

Методы и алгоритмы решения задач теории расписаний применяются для решения задач комбинаторной оптимизации. С 50-х годов двадцатого века началось активное теоретическое исследование задач теории расписаний.

Одним из главных вопросов нового направления была классификация задач и установление их сложности. Подавляющее большинство изученных задач теории расписаний являются NP-трудными. Несмотря на это, практика требует решения таких задач. Для этого существует несколько подходов. Первым подходом является разработка полиномиальных эвристических алгоритмов. Для некоторых эвристических алгоритмов известны оценки погрешности получаемого решения. Такие алгоритмы называются приближенными.

В настоящее время широкое распространение имеют метаэвристические алгоритмы, которые находят “хорошее” решение, близкое к оптимальному, за приемлемое время. Недостатком таких алгоритмов является отсутствие оценок качества полученного решения. Неизвестно, насколько решение отличается от оптимального в наихудшем случае. Точным методам решения NP-трудных задач также уделено большое внимание в работах по теории расписаний. Наибольшее

распространение получили методы сокращенного перебора, называемые методами ветвей и границ. Для сокращения перебора находятся нижние оценки целевой функции (в случае ее минимизации) и используются комбинаторные свойства задач. Также для решения задач теории расписаний широко применяется метод динамического программирования. Часто задачи теории расписаний могут быть сформулированы как задачи целочисленного линейного программирования.

В последние годы широкое распространение получил метод программирования в ограничениях. Одной из областей его успешного применения является теория расписаний. Некоторые сложные задачи теории расписаний могут быть оптимально решены с помощью алгоритмов, использующих элементы сразу нескольких методов. Одно из их названий - “гибридные алгоритмы”. Данное направление является одним из перспективных.

Содержательно многие задачи теории расписаний являются оптимизационными, т.е. состоят в поиске среди множества допустимых расписаний (расписаний, допускаемых условиями задачи) тех решений, на которых достигается “оптимальное” значение целевой функции. Обычно под “оптимальностью” понимается минимальное или максимальное значение некоторой целевой функции. Допустимость расписания понимается в смысле его осуществимости, а оптимальность — в смысле его целесообразности.

Метод решения задач дискретной оптимизации – это общая идея, применимая к широкому классу задач. Алгоритм решения – это реализация метода решения для конкретной задачи. Различают следующие методы решения:

- Эвристические алгоритмы (в т.ч. вероятностные алгоритмы и локальный поиск);
- Метод динамического программирования;
- Метод Ветвей и Границ;
- Метаэвристические методы;
- Графический метод и т.д.

Эвристические алгоритмы – алгоритмы, основанные на правдоподобных, но не обоснованных математически предположениях о свойствах оптимального решения задачи. Фактически, в эвристическом алгоритме учитывается одно или несколько свойств оптимального решения, на основе которых производится сокращение перебора возможных решений.

Метод динамического программирования получил большое распространение при решении некоторых задач дискретной оптимизации. Далее приводится формальное описание сути динамического программирования.

Разобраться с этим описанием легче после ознакомления с алгоритмом динамического программирования для задачи о упаковки. Динамическое программирование – раздел математического программирования, посвященный исследованию многошаговых задач принятия оптимальных решений. При этом многошаговость задачи отражает реальное протекание процесса принятия решений во времени, либо вводится в задачу искусственно за счет расчленения процесса принятия однократного решения на отдельные этапы, шаги. Цель такого представления состоит в сведении исходной задачи высокой размерности к решению на каждом шаге задачи меньшей размерности.

Графический метод – это модификация метода динамического программирования.

Классификация алгоритмов решения по трудоемкости:

- Полиномиальные алгоритмы;

- Псевдо-полиномиальные алгоритмы;

- Экспоненциальные алгоритмы.

Классификация по точности:

- Результатом работы точного алгоритма всегда является оптимальное решение. Как правило, в этих алгоритмах применяют различные способы сокращения перебора;

- Приближенные алгоритмы находят решение без каких-либо гарантий точности получаемых решений. Обычно это быстрые полиномиальные

алгоритмы, основанные на какой-либо догадке о свойствах оптимальных решений. Часто такие алгоритмы называют эвристическими;

- Приближенные алгоритмы, с гарантированными оценками качества. Их отличие от алгоритмов из предыдущего пункта заключается в том, что можно оценить абсолютную или относительную погрешность, т.е. отклонение полученного значения целевой функции от оптимального;

- Приближенные алгоритмы, с регулируемой точностью. Время работы таких алгоритмов зависит от выбранной пользователем необходимой точности. Например, нас может устроит отклонение от оптимума не больше чем на 1%.

На практике для решения NP-сложных задач довольно широко используют эвристические методы, которые не гарантируют нахождение оптимального решения, но позволяющие достаточно быстро получать решения приемлемого качества. К наиболее эффективным и популярным эвристическим методам относятся так называемые метаэвристики - обобщенные стратегии поиска оптимума в пространстве решений. Начальная их реализация, как правило, достаточно проста и позволяет быстро получить практический результат.

Можно выделить следующую классификацию метаэвристических алгоритмов.

Эволюционные методы. К таким методам относятся генетические алгоритмы, метод рассеивания, метод дифференциальной эволюции, метод, имитирующий поведение кукушки. Так же есть методы, имитирующие иммунные системы организмов – методы искусственных иммунных систем.

Методы «роевого» интеллекта. Например, метод муравьиных колоний, метод пчелиных колоний, алгоритм, имитирующий поведение лягушек и т.д.

Методы, имитирующие физические процессы. Наиболее популярным методом в данном классе является алгоритм имитации отжига. Так же можно отметить метод гравитационной кинематики.

Мультистартовые методы. Жадный адаптивный метод случайного поиска и метод направленного табу-поиска. Данные методы являются менее распространёнными и, как правило, не используется для построений расписаний.

На сегодняшний день наибольший интерес для изучения и развития представляет такой мета-эвристический алгоритм, как генетический. Так как они хорошо зарекомендовали себя и с точки зрения быстроты нахождения решения, и с точки зрения качества найденного решения.

Мета-эвристические алгоритмы – методы, в которых изучаются наиболее перспективные части пространства решений. Главным условием успешного мета-эвристического алгоритма является получение результатов, качественно превосходящих классические эвристики. Генетический алгоритм является эффективным для решения NP-трудных задач с большим количеством данных. Простота расчетов и быстродействие выполняемых операций позволяет производить большое количество итераций, тем самым найти эффективное решение в условиях ограниченного времени. Разработка эффективных алгоритмов позволяет рационально использовать ресурсы.

Генетический алгоритм основан на моделировании биологических механизмов популяционной генетики. Биологическая популяция эволюционирует несколько поколений по законам естественного отбора, который подчиняется принципу «выживает наиболее приспособленный». Приспособленность особи при решении задач определяет целевая функция, которая сформулирована следующим образом: чем меньше значение целевой функции, тем более особь приспособлена.

Терминология, используемая при решении задач с помощью генетического алгоритма:

1) Особь или хромосома – последовательность пунктов для посещения с начальной и конечной точкой на складе.

2) Ген – пункт для посещения (клиент).

3) Локус – местоположение конкретного пункта в последовательности посещения клиентов.

4) Генофонд – конечное множество всех последовательностей пунктов для посещения.

5) Популяция – набор всех пробных решений.

Операторы и процедуры генетического алгоритма:

- Скрещивание (процесс размножения): создается исходная популяция A и B со значением целевой функции L1 и L2; проводятся процедуры скрещивания между брачной парой, в результате которой создается новая особь C со значением целевой функции L'; проводится корректировка, удаление дублирующихся пунктов и добавление недостающих. Так как в процессе скрещивания некоторые пункты могут дублироваться, а какие-то пропасть из цепочки.

- Мутация: создается исходная популяция, из которой одна хромосома A3 с целевым значением функции L3; при проведении одноточечной мутации случайным образом выбирается один ген и переносится в другое место также случайным образом в результате чего появляется новая особь A4 с целевым значением функции L4. При удачном проведении мутации L3 должно быть больше, чем L4.

Рассмотрим оптимизационную задачу маршрутизации, в которой требуется составить такой план перевозок, при котором все запасы всех поставщиков вывозятся полностью, запросы всех потребителей удовлетворены полностью и суммарные затраты на перевозку всех грузов минимальны.

Целевая функция поставленной задачи (1) выражает требование обеспечить минимум суммарных затрат на перевозку всех грузов:

$$Z(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \rightarrow \min \quad (1)$$

С помощью первой группы из m уравнений (2) описывается тот факт, что запасы всех m поставщиков вывозятся полностью:

$$\sum_{j=1}^{n} x_{ij} = a_i, i = 1, \dots, m \quad (2)$$

Вторая группа из n уравнений (3) выражает требование полностью удовлетворить запасы всех n потребителей:

$$\sum_{i=1}^{m} x_{ij} = b_i, i = 1, \dots, n \quad (3)$$

Условиями неотрицательности всех переменных задач является неравенство (4):

$$x_{ij} \geq 0, i = 1, \dots, m, j = 1, \dots, n, \quad (4)$$

где m – поставщики груза в объемах a_1, a_2, …, a_m;

n – потребители, которым необходим груз в объеме b_1, b_2, …, b_m;

c_{ij} (i = 1, …, m; j = 1, …, n) – стоимость перевозки единицы груза откаждого i-го поставщика каждому j-му потребителю;

x_{ij} (i = 1, …, m; j = 1, …, n) – объемы перевозок от каждого i-го поставщика каждому j-му потребителю.

Рассмотрим пошаговую процедуру решения задачи маршрутизации при помощи генетического алгоритма:

1. Генерируется начальная популяция A_i, где i = 0, …, n.

2. Вычисляется общая длина маршрута для каждой из особей L_i, где i = 0, …, n.

3. Выбирается пара хромосом-родителей A_1 и A_2.

4. Проводится кроссинговер двух родителей.

5. С определенной степенью вероятности P проводится мутация.

6. Шаги 3-5 повторяются пока не будет сгенерировано новое поколение популяции.

7. Шаги 2-6 повторяются, пока не будет достигнут критерий окончания.

Из всего вышеизложенного можно сделать вывод, что генетический алгоритм целесообразно использовать для логистических информационных систем при расчете рациональных маршрутов транспортных средств. Простота структуры алгоритма, быстродействие выполняемых операций позволяют гарантировать выполнение большого количество итераций за приемлемое время,

допустимое для оперативного планирования маршрутов, что позволяет использовать алгоритм для решения задач маршрутизации на практике. Реализация данного алгоритма в виде программного модуля позволит провести вычислительный эксперимент и исследовать эффективность получаемых расчетов (рисунок 4.1).

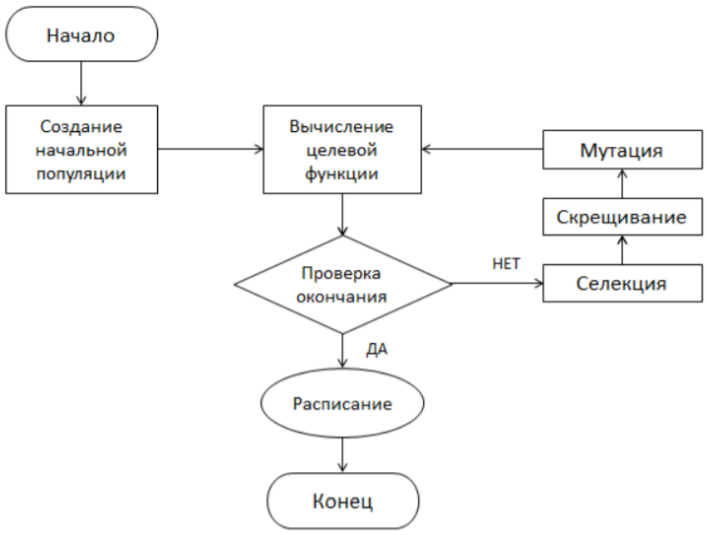

Рисунок 4.1 - Блок-схема генетического алгоритма

Итак, генетические, как и эволюционные алгоритмы в целом, основываются на использовании механизмов естественной эволюции. Эволюция по Дарвину осуществляется в результате взаимодействия трех основных факторов: изменчивости, наследственности, естественного отбора.

Изменчивость играет роль основой появления новых признаков и особенностей в строении и функциях организма. Наследственность закрепляет эти признаки. Естественный отбор устраняет организмы, плохо приспособленные к условиям существования.

Алгоритм предусматривает последовательность жизненных циклов популяции – поколений. Каждый цикл состоит из следующих операций: селекции наиболее приспособленных особей, выбора родительских пар, их размножения и мутации. Процесс повторяется до тех пор, пока на протяжении заданного количества поколений не будет появляться особь с лучшим значением функции приспособленности.

4.1.2 Анализ алгоритмов составления расписания для междугородних пассажирских перевозок

Для совершенствования оперативного планирования работы пассажирских перевозах на междугородных маршрутах необходимо иметь методику, адекватно отвечающую современным условиям функционирования и потребностям участников рынка транспортных услуг.

При этом следует руководствоваться принципом системного, комплексного подхода, суть которого заключается не в установлении соответствия того или иного параметра действующим нормам и требованиям, а в нахождении решения, позволяющего не только формально устранить отмеченный недостаток, но и найти вариант улучшения этого показателя, повысить эффективность использования каждого элемента имеющегося технологического потенциала, достичь наилучшего результата. В связи с чем надо не только изучать существующее положение объекта исследования, но и рассматривать перспективы его развития.

Предприятие, работающее в автотранспортной отрасли, должно правильно определять провозную возможность парка, а именно, количество пассажирских перевозок, необходимых для выполнения заданного объема работ в установленные сроки.

Обеспечение выполнения заказов в полном объеме и в срок при минимальных затратах может быть достигнуто созданием оптимальных провозных возможностей АТП и повышением их эффективности.

Целью функционирования обслуживающей системы в целом является удовлетворение требований на перевозку, поэтому важным показателем является производительность системы и входящий поток требований, поступающих в систему. В случае превышения числа поступающих требований над пропускной способностью в системе возникает очередь требований на выполнение перевозки. Очередь может образовываться перед каждой

подсистемой в отдельности, следовательно, каждая из подсистем может блокировать работу всей системы.

Время заявки (срочность ее выполнения), расстояние перевозки (длина маршрута), продолжительность перевозки влияют на величину пропускной способности системы, поскольку являются составляющими входящего потока требований. На величину пропускной способности системы также влияют и внутренние факторы этой системы, такие как организация работы отделов системы и технологический процесс, и производственные мощности предприятия, имеющиеся на конкретный момент времени (по количеству и состоянию).

При исследовании системы необходимо учитывать влияние отдельных подсистем на работу всей системы. Система может успешно справляться с возложенными на нее задачами только при условии, что пропускная способность системы превышает суммарный входящий поток требований на все виды перевозок. Излишнее количество пассажирских перевозок связано с большими капитальными вложениями, что приводит к замораживанию средств предприятия, поэтому необходимо точно определять наиболее выгодную (оптимальную) величину резерва пассажирских перевозок.

Оптимальная величина резерва производственных мощностей системы может быть выявлена по экономическому критерию - обеспечение минимума затрат или максимума прибыли.

В современных условиях функционирования транспортных компаний остается актуальной задача точного определения выработки подвижного состава и ресурсов, необходимых для осуществления доставки грузов. Одним из направлений рационального использования ресурсов на автомобильном транспорте является оптимизация затрат методами математического программирования, с помощью которых возможно проектирование материально-технической базы и оптимальных маршрутов доставки грузов, распределение подвижного состава и погрузочно-разгрузочных средств и многое другое.

Большая протяженность маршрутов приводит к необходимости отдавать предпочтение при выборе подвижного состава пассажирскому транспорту с большим количеством посадочных мест для выполнения перевозок меньшим количеством оборотов. Этот фактор был учтен в алгоритме распределения подвижного состава по заявкам на междугородных маршрутах.

В основу методики положена фиксация временного интервала «время выполнения j-ой заявки» и моментов начала и окончания выполнения работы идентифицированного пассажирского транспорта в операторе учета времени.

При автоматизированном учете в методике таких факторов, как объем груза, заявленного к перевозке, время заявки, время оборота на маршруте, занятость пассажирских перевозок на других заявках, производительность транспорта на заявке, распределение пассажирских перевозок по заявкам происходит с высокой оперативностью, точностью, эффективностью и с меньшими трудозатратами.

В методике учитывается специфическая особенность междугородных перевозок: большая протяженность маршрутов приводит к необходимости отдавать предпочтение при выборе подвижного состава автомобилям большей грузоподъемности для выполнения заявки меньшим количеством оборотов.

Методика позволяет сочетать в себе и выполнять одновременно операции по определению рациональной грузоподъемности пассажирских перевозок относительно требований заявки, расчету потребного количества пассажирских перевозок с учетом оптимальной производительности подвижного состава и затрат на перевозку, учету затрат на перевозку и расчету технико-эксплуатационных показателей работы подвижного состава, определению занятости парка пассажирских перевозок в любой момент времени. Для осуществления оперативного планирования по приведенному алгоритму необходимо сформировать базы данных (БД) исходной информации «$З_j$» (Поступающие заявки), «$А_{iксоб}$» (Собственный парк пассажирских перевозок), «$А_{iкпривл}$» (Привлеченный парк пассажирских перевозок) и базы данных результирующей информации «ТЭП $_j$» (ТЭП по заявке), «ТЭП $_{ц\ расч}$» (ТЭП за

«время цикла расчетное»), «$С_{пер\ j}$» (Затраты на перевозку по заявке), «$С_{пер\ ц\ расч}$» (Затраты за «время цикла расчетное»).

Планирование может выполняться автотранспортным или предприятием, эксплуатирующим собственный или привлеченный пассажирский транспорт. Для оценки и прогнозирования разных параметров междугородних пассажирских перевозок можно использовать методы экспертных оценок. На основе мнения экспертной группы создается суммарная информация об изучаемом объекте (маршруте) и создается решение, которое задается целью экспертизы. Во время работы над индивидуальными мнениями экспертов группы применяют разные количественные и качественные методики. Решение о том, какую выбрать методику, обуславливают сложности проблемы, которую надо решить, формы, в которой представлены оценки экспертной группы и целей экспертизы.

Для того, чтобы обработать итоги оценки результатов экспертами применяются методики математической статистики.

От того, какие цели преследует экспертиза во время обработки оценок, зависит, как будут решаться следующие проблемы:

- образование совокупной оценки;
- установление относительных весов объектов;
- определение степени согласованной договоренности точек зрения экспертной группы и т. д.

Отметим методы измерения объектов, которые чаще применяются во время оценки по порядковой или интервальной сетке: распределение/ранжирование, парное суждение, методика последовательных суждений, непринужденная оценка.

1. Ранжирование представляет собой расстановку объектов в порядке возрастания или убывания какого-нибудь характерного им признака. Ранжирование дает право выбрать из анализируемой совокупности причин, более значимую.

Причем всех экспертов призывают расстановить эти свойства в порядке предпочтительности. Цифрой 1 указывается более значимый признак, цифрой 2 – последующий за ним по значимости и так далее.

Итог ранжирования это ранжировка.

Если присутствует n объектов, то вследствие их ранжирования j-ым экспертом каждый объект приобретает оценку x_{ij} – ранг, который приписывается i-му объекту j-ым экспертом.

Значения x_{ij} располагаются на интервальном расстоянии от 1 до n. Ранг самого важного фактора равно единице, а менее важного – числу n.

Далее, благодаря методике математической статистики получают совокупное мнение экспертов.

Преимущества методики выражаются в следующем:

- относительная простота процесса получения оценок;
- меньшее количество экспертов относительно прочих методик при оценке одного и того же набора признаков.

Минусы:

- заранее считается, что оценки распределены равномерно;
- снижение значимости признаков считается тоже равномерным, при этом на практике такого не существует.

Более того, во время оценки существенного числа объектов экспертам весьма сложно сформировать ранжированный ряд, так как надо брать в расчет большое число сложных связей.

От подобного неудобства свободна следующая методика – парное сравнение, которое представляет собой определение предпочтительности объектов при сопоставлении всех допустимых пар. Тут необходимо, как при ранжировании, установить порядок всех объектов, нужно в любой из пар обнаружить более значимый объект или определить их равные позиции. То есть все факторы попарно сравниваются между собой. На основе парных сравнений

с помощью последующей обработки оказываются далее оценки каждого признака.

Парное сравнение проводится при множестве объектов, и в тех ситуациях, где различие среди объектов такое незначительное, что почти невыполнимо их ранжирование.

При применении методики, как правило, формируется матрица размером n x n, где n – количество сравниваемых объектов. Как выглядят матрицы парных сравнений в общем виде, указано в таблице 1.

При сравнении объектов матрица заполняется элементами a_{ij} следующим образом:

– 2, в случае, когда объект i предпочтительнее объекта j ($i > j$),

– 1, в случае, когда определено равенство объектов ($i = j$),

– 0, в случае, когда объект j предпочтительнее объекта i ($i < j$).

Общая сумма $\sum_{j=1}^{n} a_{ij}$ (по строке) в текущей ситуации предоставляет возможность для оценки сравнительной важности объектов. Объект, для которого сумма будет большей, может признаваться более значимым.

Сложение можно осуществлять и по столбцам ($\sum_{j=1}^{n} a_{ij}$), тогда самым значимым будет фактор, который набрал меньшее количество баллов.

Таблица 4.1 - Общий вид матрицы парных сравнений

Объекты			...		...		
1	0	1	2	1	2	0	1
2	2	1	0	2	1	0	2
…	1	0	2	1	2	0	1
i	2	1	0	0	2	1	2
…	1	0	2	2	0	1	1
n	0	2	1	0	2	2	1

Половина таблицы, которая расположена выше диагонали, является отражением той половины таблицы, которая ниже. Для того, чтобы не запутать, не спровоцировать эксперта для вычисления одной половины таблицы по другой, чтобы сократить количество операций, логично и правильнее заполнять только одну часть таблицы (выше или ниже диагонали).

3. Методика последовательных сравнений заключается в следующем:

– эксперт придает надлежащий порядок всем признакам в порядке уменьшения их важности: $A_1 > A_2 > A_3 > ... > A_n$;

– дает первому признаку значение, которое равно единице: $A_1 = 1$, остальным признакам ставит весовые коэффициенты в долях единицы;

– поддает сравнению значение первого признака с суммой всех последующих. Допустимы три варианта:

$$A_1 > A_2 + A_3 + ... + A_n$$
$$A_1 = A_2 + A_3 + ... + A_n$$
$$A_1 < A_2 + A_3 + ... + A_n$$

Эксперт выбирает наиболее подходящий, с его точки зрения, вариант и приводит в соответствующую форму с ним оценку первого события;

– сравнивает значение первого признака с суммой всех последующих за исключением самого крайнего признака. Делает оценку первого признака согласно выбранному из трех вариантов неравенству:

$$A_1 > A_2 + A_3 + ... + A_{n-1}$$
$$A_1 = A_2 + A_3 + ... + A_{n-1}$$
$$A_1 < A_2 + A_3 + ... + A_{n-1}$$

– процесс имеет повторения до сравнения A_1 с $A_2 + A_3$.

Вслед за тем, как экспертный специалист дал оценку первого признака согласно выбранному им неравенству из трех возможных:

$$A_1 > A_2 + A_3$$
$$A_1 = A_2 + A_3$$
$$A_1 < A_2 + A_3$$

он следует к уточнению оценки второго признака A_2 по такому же механизму действий, что и в первом случае, т. е. сравнивается оценка второго признака с общей суммой последующих.

Преимущество ее выражается в том, что эксперт в ходе проведения оценки признаков сам проводит анализ собственные оценки. Вместо назначения коэффициентов появляется творческий процесс формирования данных коэффициентов.

Минусы методики, следующие:

– ее сложная составляющая; эксперту, который ранее не был знаком с ней, она составит трудности при работе с этим процессом; так как, взамен того, чтобы уточнять собственные изначальные оценки, он будет путаться в них;
– ее громоздкость; для того, чтобы оценить один и тот же набор признаков ей нужно в 4 раза больше действий, нежели методике обычной ранжировки (суть в том, что для одной и той же работы надо в 4 раза больше экспертов).

4. Методика непосредственной оценки. Зачастую существует требование не только упорядочить (ранжировать объекты анализа), но и установить, на сколько какой-то один фактор более важен, нежели прочие.

В таком случае диапазон изменения в характеристиках объекта разделяется на определенные интервалы, где каждому предоставляется определенная оценка (балл), к примеру, от 0 до 10.

Как раз из-за этого методику непосредственной оценки часто называют также балльной методикой.

Суть методики заключается в том, что эксперт устанавливает любой из рассматриваемых объектов в определенный интервал (добавляет балл). Причем измерителем здесь будет уровень обладания объекта каким-либо определенным свойством.

Число интервалов, на которые разделяется диапазон изменения свойства, бывает разным для различных экспертов. Более того, методика допускает делать одну и ту же оценку (то есть устанавливать в один и тот же интервал) разным объектам.

После проведения оценки показателей экспертной группой проводится обработка итогов. Исходными сведениями для нее служат числовые данные, которые выражают предпочтения экспертов, и глубокое подтверждение таковых предпочтений. Целесообразностью обработки считается получение общих сведений и новой информации, которая содержится в скрытой форме в оценках эксперта. На базе результатов обработки формируются общие оценки работ (результатов) прогнозируемых показателей.

Присутствие числовых данных и суждений экспертов создает условия, которые требуют использование качественных и количественных методик обработки итогов коллективного экспертного оценивания. Удельный вес этих методик сильно зависит от класса проблем, которые решаются с помощью оценки экспертов. Рассмотрим методики обработки проблем первого класса, которые характеризуются необходимым информационным потенциалом. Как правило, данные проблемы больше распространяются на практике принятия решений.

От того, какие цели преследует экспертиза во время обработки оценок, зависит, как будут решаться следующие задачи:

- установление согласованной договоренности мнений экспертной группы;
- создание общей суммарной оценки объектов;

– установление зависимости среди мнений экспертов;
– установление относительных весов объектов;
– оценка надежности результатов экспертизы.

Установление согласованности оценок среди экспертной группы нужно для того, чтобы подтвердить верность гипотезы о том, что эксперты — это самые верные измерители, и выявить допустимые группировки в группе экспертов. Оценка согласованности мнений экспертов группы происходит с помощью высчитывания количественной меры, характеризующую уровень схожести мнений каждого из экспертов. Анализ значений меры согласованности дает возможности для того, чтобы выработать верное мнение об общем уровне знаний по проблеме, которая решается, и выявить группировки суждений экспертов, которые обусловлены различием точек зрения, направлений, существованием научных школ, характером профессиональной деятельности.

Задача формирования общей оценки объектов по индивидуальным мнениям экспертов появляется во время коллективной экспертной оценки. В случае, когда эксперты выполняли оценку объектов в количественной шкале, тогда задача формирования коллективной оценки состоит в том, чтобы определить среднее значение или медианы оценки. В процессе измерения в порядковой сетке методикой ранжирования или парного сравнения целью обработки оценок каждого из экспертов группы служит построение суммарной порядковой организации объектов на базе осреднения оценок экспертов.

Обработкой итогов экспертной оценки определяется зависимость среди мнений разных экспертов. Обнаружение данных зависимостей дает возможности для определения степень близости в оценках экспертов группы. Существенная значимость имеет установление зависимости среди оценок объектов, которые построены по разным показателям сопоставления. Это способствует определению взаимосвязанных между собой показателей сопоставления и осуществлению их группировки по степени взаимной связи.

Когда нужно найти решения для большей части задач, мало, просто, составлять упорядочивание объектов по одному или по группе показателей.

Лучше, таким же образом, иметь количественные значения относительной значимости объектов. Для того, чтобы решить такую задачу надо, не задумываясь, использовать методику непосредственной оценки. Тем не менее подобную задачу во время каких-либо отдельных обстоятельствах допускается решить с помощью обработки итогов ранжировок или парных сравнений экспертной группы.

Оценки проектов, которые получаются вследствие обработки, выражаются в случайных величинах, в связи с этим наиважнейшей задачей служит установление их верности и точности, то есть необходимо понимать, что результаты оценивания надежные.

Изучим ряд методик решения каждой из перечисленных задач:

1) Создание общей оценки.

Допустим, пусть экспертная группа сделала оценку какому-то проекту, тогда x_j – оценка *j*-го эксперта, *j* = 1, *m*, где *m* – число экспертов.

Для того, чтобы сформировать общую оценку экспертной группы, как правило, применяются средние величины. К примеру, медиана (ME), за которую считается такая оценка, где количество больших оценок равно числу меньших.

Возможно, применять таким же образом, и точечную оценку для экспертной группы, которая вычисляется, как среднее арифметическое:

$$\bar{x}_э = \frac{\sum_{j=1}^{m} x_j}{m}$$

2) Установление относительных весов объектов.

Порой необходимо установить, в какой степени определенный фактор (объект) значим (существенен) с позиции какого-нибудь критерия. Тогда говорят, что требуется определение веса каждого фактора.

Одна из методик определения весов заключается в следующем. Пусть x_{ij} - оценка фактора *i*, данная *j*-ым экспертом, *i*=1, *n*, *j*=1,*m*, *n* – число сопоставляемых

объектов, m – число экспертов. Тогда вес i-го объекта, который подсчитан вследствие оценок всех экспертов w_i, равен:

$$w_i = \frac{\sum_{j=1}^{m} w_{ij}}{m},$$

где w_{ij} - вес i-го объекта, который подсчитан по оценкам j-го эксперта, равен

$$w_{ij} = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}},\ j = \overline{1,m}, i = \overline{1,n}$$

3) Определение степени согласованности оценок экспертов группы.

В ситуации участия в опросе ряда экспертов различия в их оценках являются неизбежным фактором, но величина этих различий имеет существенное значение. Коллективная оценка считается очень надежной только тогда, когда хорошо сформирована согласованность ответов отдельных экспертов.

Для того, чтобы анализировать разброс и согласованность оценок используются статистические характеристики – меры разброса.

Вариационный размах

$$(R) = x_{\max} - x_{\min}$$

где $x_{\max}$ - максимальная оценка объекта; $x_{\min}$ - минимальная оценка объекта.

Среднее квадратическое отклонение, которое вычисляется по известной всем формуле:

$$\sigma = \sqrt{\frac{\sum_{j=1}^{m} (x_\rho - \bar{x})^2}{m-1}},$$

где x_j - оценка, данная j-ым экспертом; m – количество экспертов.

Коэффициент вариации (V), который, как правило, указывается в процентах:

$$V = \frac{\sigma}{\bar{x}} \cdot 100\%$$

Обладают специфичностью подходы к проверке согласованности, которые применяются во время оценки объектов методикой ранжирования.

Тогда результатом работы эксперта служит ранжировка, являющаяся последовательностью рангов (для эксперта j): $x_{1j}, x_{2j}, ..., x_{nj}$.

Согласованность между ранжировками 2-х экспертных специалистов допускается определить с помощью коэффициента ранговой корреляции Спирмэна:

$$\rho = 1 - \frac{6\sum_{i=1}^{n}(x_{ij} - x_{ik})^2}{n(n^2 - 1)} = 1 - \frac{6\sum_{i=1}^{n} d_j^2}{n(n-1)}$$

где x_{ij} – ранг, присвоенный i-му объекту j-ым экспертом; ik – ранг, присвоенный i-му объекту k-ым экспертом; i – разница между рангами, присвоенными i-му объекту.

Величина может меняться в диапазоне от –1 до +1. Во время однозначного совпадении оценок коэффициент равен единице. Равенство коэффициента минус единице наблюдается при наибольшем расхождении в суждениях экспертов.

Более того, расчет коэффициента ранговой корреляции может использоваться в виде оценки взаимоотношений между каким-нибудь фактором и результативным признаком в тех ситуациях, где признаки не могут быть измерены точно, но их можно упорядочить.

В данной ситуации показатель коэффициента Спирмэна может быть интерпретирован как значение коэффициента парной корреляции. Положительное значение демонстрирует прямую взаимосвязь среди факторов,

отрицательное – об обратном, причем, чем ближе абсолютное значение коэффициента к единице, тем более тесная взаимосвязь.

В случае, где требуется установление согласованности в ранжировании большего (свыше двух) количества экспертов, рассчитывается, так сказать, коэффициент конкордаци-хи, который представляет из себя общий коэффициент ранговой корреляции для группы, состоящей из *m* экспертов:

$$W = \frac{12 \cdot S}{m^2 (n^3 - n)},$$

где,

$$S = \sum_{i-1}^{n} \left(\sum_{j-1}^{m} x_{ij} - \frac{1}{2} m(n+1) \right)^2$$

Обратим внимание на то, что вычитается в скобках выступает именно в роли средней суммы рангов (при суммировании для всех объектов по отдельности), полученных *i* объектами от экспертов.

Коэффициент *W* может преобразовываться в диапазоне 0-1. Его равенство единице обозначает, что все эксперты присвоили объектам один и тот же ранг. Насколько ближе значение коэффициента к нулю, настолько меньше согласованными выступают оценки экспертов.

С целью определения весомости разных критериев, во время выбора ограниченной совокупности показателей для реализации сравнительной оценки при применении методики средневзвешенного показателя, требуется решение двух задач. Первая задача заключается в определении номенклатуры показателей качества, допускается трех типов. Первый тип – выбрать из некого числа значимых в каком-то смысле показателей из заданного исходной списка. Второй тип – организовать список значимых показателей. Третий тип – выбрать из заданного списка определенного числа значимых, «весомых», показателей. Вторая задача заключается в определении методики оценки показателей, то есть их весомости.

При применении экспертных методик привлечение множества экспертов увеличивает объективность итогов оценки. Однако, применение множества экспертов и высокая трудоемкость экспертных работ поднимают стоимость проведения оценок качества. С целью уменьшения трудоемкости работ экспертов применяют самый малотрудоемкую методику – методику рангов, которая предусматривает лишь ранжирование показателей, а не их численное определение экспертами.

При осуществлении этой методики эксперты обязаны реализовать прямую оценку важности каждого критерия по шкале относительной значимости оценок от 1 до 10. Для определения коэффициентов весомости оцениваемых свойств применяют формулу:

$$M_{ij} = R_{ij} \Big/ \sum_{i=1}^{n} R_{ij}$$

где, R_{ij} - абсолютное значение оценки i критерия, определенное по 10 бальной шкале j экспертом;

n – количество критериев;

m – количество экспертов.

На основе полученных коэффициентов весомости устанавливают наиболее важные единичные критерии, для которых выполняется условие:

$$M_{ij} > \frac{1}{2}$$

Т. к. сумма $M_j = 1$, то после исключения наименее важных критериев коэффициенты весомости остальных пересчитывают по формуле [36]:

$$M_{i0} = M_i^* \Big/ \sum_{i=1}^{k} M_i$$

4.2 Разработка технического задания модуля составления расписания пассажирских перевозок

Целью технического задания является разработка модуля составления расписания пассажирских перевозок. В итоге должен получиться работоспособный модуль, на котором располагается информация о составленном расписании и контактной информацией.

Модуль создается для упрощения и делегирования некоторых задач, которые в данный момент лежат на администраторе. А именно, информация о составлении расписания пассажирских перевозок, контактной информации.

В результате изучения теоретических вопросов в направлении изучения особенностей анализа и планирования пассажирских перевозок были определены следующие задачи, решение которых позволит создать модуль расписания общественного транспорта:

- выбрать технологии реализации приложения;
- определить источники данных для получения данных;
- разработать модель базы данных приложения;
- осуществить физическое моделирование базы данных приложения;
- разработать структуру приложения;
- разработать алгоритмы приложения;
- проверить работоспособность созданного приложения.

Требования к функциональным характеристикам:

- Предоставление информации о транспорте
- Предоставление информации о маршрутах
- Предоставление информации о контрольных точках
- Предоставление информации о водителях
- Предоставление информации о сформированном расписании

Входные данные программы должны быть организованы в виде вводимого в специальную форму текста или файла, соответствующего определенному шаблону. Данные, вводимые вручную, проверяются на корректность после попытки сохранения; данные, вводимые из файла, проверяются в ходе анализа и размещения данных.

Файлы указанного формата должны размещаться (храниться) на локальных или съемных носителях, отформатированных согласно требованиям операционной системы. Каждый день происходит резервирование полученной информации на отдельный носитель, для возможности восстановления информации в случае ошибки программы или поломки оборудования.

Выходные данные программы должны быть организованы в виде отчёта. Отчеты делятся на несколько групп по предназначению определенной группе пользователей. Доступ к таблицам может быть только у администратора.

Файлы указанного формата должны храниться на локальных или съемных носителях, отформатированных согласно требованиям операционной системы. Отчеты формируются в режиме реального времени и передаются пользователю. Отчеты, являются временными и стираются по завершению работы программы, могут быть сформированы заново при следующем запуске компьютера. При желании любой отчет можно сохранить отдельно.

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- организацией бесперебойного питания технических средств;
- выполнением рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;

- выполнением требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов;
- необходимым уровнем квалификации сотрудников профильных подразделений.

В Системе должен быть обеспечен надлежащий уровень защиты информации в соответствии с законом о защите персональной информации и программного комплекса в целом от несанкционированного доступа - “ Об информации, информатизации и защите информации” РФ N 24-ФЗ от 20.02.95.

Программа должна храниться на или ПК с возможностью подключения к Интернет сети.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

- разработка, согласование и утверждение программы и методики испытаний;
- проведение приемо-сдаточных испытаний;
- корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию.

4.3 Проектирование модуля

4.3.1 Разработка диаграмм: UML, IDEF

Для описания функциональных требований будут использованы возможности диаграмм вариантов использования (Use Case) которые позволяют отобразить пользователей системы (актеров) и функции, которые выполняет каждый актер.

Диаграмма вариантов использования, которая является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и

отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (действующим лицом, актантом, актером) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий. В данной модели показаны актеры с различными вариантами взаимодействия.

Основные элементы диаграммы - участник (actor) и прецедент (вариант). Участник — это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности. Графически участник изображается “человечком”.

Прецедент (use case) - описание множества последовательных событий (включая варианты), выполняемых системой, которые приводят к наблюдаемому участником результату. Прецедент представляет поведение сущности, описывая взаимодействие между участниками и системой. Прецедент не показывает, “как” достигается некоторый результат, а только “что” именно выполняется. Прецеденты обозначаются очень простым образом - в виде эллипса, внутри которого указано его название. На рисунке 4.2. приведена диаграмма вариантов использования.

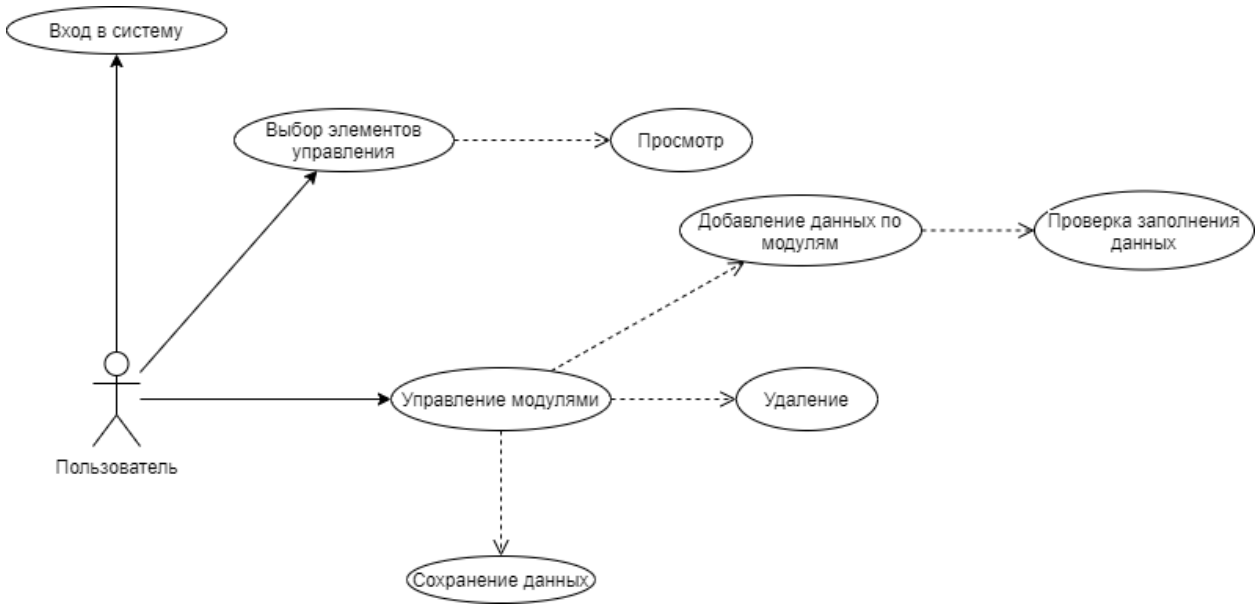

Рисунок 4.2 - Use-Case диаграмма

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Впрочем, часто возвращаемые результаты обозначают лишь в том случае, если это не очевидно из контекста.

Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов). Сообщения (вызовы методов) - линиями со стрелками. Возвращаемые результаты - пунктирными линиями со стрелками. Прямоугольники на вертикальных линиях под каждым из объектов показывают “время жизни” (фокус) объектов. Впрочем, довольно часто их не изображают на диаграмме, все это зависит от индивидуального стиля проектирования. На рисунке 2.3 представлена диаграмма последовательности загрузки маршрута.

Рисунок 4.3 – Диаграмма последовательности (Sequence diagram)

Для более полного анализа процессов построения маршрутов использованы возможности нотации IDEF0. Нотация IDEF0 — это методология моделирования, дающая возможность создания функциональной модели, отображающей структуру и функции системы, а также потоки данных и материальных объектов, связывающих данные функции. Бизнес-процессы в нотации IDEF0 представляются в форме прямоугольника, а стрелки отражают связь с другими процессами и внешней средой.

Методология IDEF0 является эффективным средством анализа, проектирования, а также представления деловых процессов. Структурной единицей модели IDEF0 есть диаграмма, которая представляет собой графическое описание модели предметной области, а также и ее части. Основным компонентом диаграммы IDEF0 является блок. Блоки диаграммы отображают работы, процессы, функции и задачи, выполняемые в определенное время.

Методология создания моделей бизнес-процессов представляет собой сочетание принципов и способов организации деятельности и технологий, которые применяются для анализа, описания и создания бизнес-моделей.

Методология IDEF0 использует метод функционального моделирования сложных систем, являющийся частью методологии структурного анализа и проектирования SADT (Structured Analysis and Design Technique). IDEF0 позволяет строить функциональные модели, которые описывают бизнес-процессы в виде иерархической системы взаимосвязанных функций.

Моделирование бизнес-процессов затрагивает многие аспекты деятельности объекта исследования:

- изменение внутренних нормативных документов и технологии проведения операций;
- изменение организационной структуры;
- новые требования к автоматизации выполняемых процессов и т. д.;
- оптимизацию функций подразделений и сотрудников;
- перераспределение прав и обязанностей руководителей.

Моделирование бизнес-процессов включает два этапа структурное и детальное. Структурное моделирование бизнес-процессов может выполняться в нотации IDEF0 с использованием специализированного инструментария.

Модели в нотации IDEF0 предназначены для высокоуровневого описания процессов в функциональном аспекте.

Методология создания моделей бизнес-процессов представляет собой сочетание принципов и способов организации деятельности и технологий, которые применяются для анализа, описания и создания бизнес-моделей.

Моделирование бизнес-процессов позволяет не только определить, как система работает в целом, как взаимодействует с внешними факторами, но и как организована деятельность на каждом рабочем месте. Моделирование бизнес-процессов — это эффективное средство поиска пути оптимизации системы, средство прогнозирования и минимизации рисков.

Части функционального блока представляют собой стороны прямоугольника, каждая из которых имеет разное назначение, так левая сторона – это вход, правая – выход, верхняя управление, нижняя – механизмы. Существующее взаимодействие с внешним миром описывается с помощью дуг.

Структурным моделированием бизнес-процесса отражается следующее:

- существующая организационная структура;
- документация и другие сущности, которые используются при выполнении моделируемых бизнес-процессов и необходимы для того, чтобы моделировать документооборот, с характеристиками их основного смысла;
- структура бизнес-процессов, которая отражает их иерархию от наиболее общих групп к частным бизнес-процессам;
- диаграммы взаимодействий для конечных бизнес-процессов, которые отражают поэтапность формирования и перемещения документации (информации, материалов, ресурсов и тому подобное) между действующими лицами.

Подробное моделирование бизнес-процессов исполняется в той же модели и должно показывает необходимую детализацию, а также должна обеспечить прямое представление о работе предприятия. Необходимость моделировать бизнес-процессы состоит в характеристике методов преобразования исходных данных в отчеты и диаграммы.

Так как стандарт IDEF0 является наиболее широко используемой методологией характеристики бизнес-процессов. Модели в нотации IDEF0необходимы для высокоуровневого описания бизнеса фирмы в функциональном аспекте.

С их помощью можно эффективно отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах. При этом широта и глубина обследования процессов в системе определяется самим разработчиком, что позволяет не перегружать создаваемую модель излишними данными.

Модель IDEF0 всегда начинается с представления системы как единого целого - одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой, и

обозначается идентификатором “A-0”. В процессе декомпозиции, функциональный блок, который в контекстной диаграмме отображает систему как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы и называется дочерней по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется дочерним блоком). В свою очередь, функциональный блок - предок называется родительским блоком по отношению к дочерней диаграмме, а диаграмма, к которой он принадлежит - родительской диаграммой.

Каждая IDEF0-диаграмма содержит блоки и дуги. Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними.

Построение модели начинается с представления всей системы в виде одного блока и дуг, изображающих интерфейсы с функциями вне системы. Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Каждая детальная диаграмма является декомпозицией блока из диаграммы предыдущего уровня. На каждом шаге декомпозиции диаграмма предыдущего уровня называется родительской для более детальной диаграммы.

На рынке компьютерных технологий в настоящее время представлены несколько специальных программ, позволяющих обследовать предприятие и построить модель. Существуют стандартизированные и проверенные методологии и инструментальные средства, с помощью которых можно обследовать предприятие и построить его модель. Ключевое их преимущество - простота и доступность в освоении.

Основу многих современных методологий моделирования бизнес-процессов составила методология SADT. В настоящее время наиболее широко используемая методология описания бизнес-процессов - стандарт США IDEF.

Преимущество анализа бизнес-процессов компании путём моделирования — это универсальность его модели. Моделирование бизнес-процессов — это ответ на многие вопросы, касающиеся совершенствования деятельности предприятия и повышения его конкурентоспособности. Руководитель или руководство предприятия, внедрившие у себя конкретную методологию, будет иметь информацию, которая позволит самостоятельно совершенствовать свое предприятие и прогнозировать его будущее [18].

Основу многих современных методологий моделирования бизнес-процессов составили методология SADT (Structured Analysis and Design Technique - метод структурного анализа и проектирования) [5]. Методология SADT может использоваться для моделирования широкого круга систем и определения требований и функций, а затем для разработки системы, которая удовлетворяет этим требованиям и реализует эти функции. Для уже существующих систем SADT может быть использована для анализа функций, выполняемых системой, а также для указания механизмов, посредством которых они осуществляются. Наиболее популярная нотация моделирования бизнес-процессов, основанная на методологии структурного анализа SADT- нотация IDEF0. Модели в нотации IDEF0 предназначены для высокоуровневого описания бизнеса компании в функциональном аспекте [3].

Нотация IDEF0 — это методология моделирования, которая позволяет создать функциональную модель, отображающую структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции. Бизнес-процессы в нотации IDEF0 представляются в форме прямоугольника, а стрелки отражают связь с другими процессами и внешней средой. Особенностью нотации является:

- Возможность декомпозировать процессы на подпроцессы и, таким образом, строить иерархические модели бизнес-процессов.
- Выделение четыре типов стрелок: три типа входов — вход, управление и механизм (это позволяет более гибко описывать логику

использования входов в процессе в целях последующего анализа), и выход.

Нотация IDEF0 используется для создания верхнего уровня модели бизнес-процессов. Построение IDEF0-диаграммы верхнего уровня обеспечивает наиболее общее или абстрактное описание объекта моделирования.

Методология SADT — это лучший способ уменьшить количество дорогостоящих ошибок за счет структуризации на ранних этапах создания системы, улучшения контактов между пользователями и разработчиками и сглаживания перехода от анализа к проектированию. Поэтому при выборе нотации для моделирования бизнес-процессов компании решающим фактором является процесс создания непротиворечивой и полезной системы описаний, который можно разбить на следующие этапы:

1. Сбор информации об исследуемой области.
2. Документирование полученной информации.
3. Представление ее в виде модели.
4. Уточнение модели посредством итеративного рецензирования.

IDEF0 - методология функционального моделирования, являющаяся составной частью SADT и позволяющая описать бизнес-процесс в виде иерархической системы взаимосвязанных функций. В методе IDEF0 можно выделить такие составляющие, как концепция метода, графический язык, процедура чтения диаграммы, метод построения модели, критерии оценки качества и др.

Основным преимуществом IDEF0 является возможность декомпозиции (расшифровки более подробно) каждого компонента модели бизнес-процесса на другой диаграмме для детального анализа каждой функции или потока.

В основе деятельности любой современной организации лежат цели, которые она ставит перед собой, чтобы лучше представлять, куда ей двигаться и к чему стремиться. Для достижения любой цели требуется решить некоторый набор задач. Задачи, как известно, в управлении организацией играют важную роль и указывают исполнителям, какие необходимо использовать ресурсы и

какие необходимо осуществить действия для решения поставленных ранее целей. Поэтому именно нотация IDEF0 является ключом в поиске решений поставленных задач. Она наглядно демонстрирует элементы бизнес-системы в цепочке организационных взаимодействий.

IDEF0-модель описывает: что система делает, что она производит, какая информация используется для управления, какие ресурсы и средства применяются для исполнения ее функций. Именно эти возможности позволяют рассмотреть бизнес-процессы предприятия со всех сторон и правильно выявить процессы, протекающие недостаточно эффективно, чтобы в дальнейшем провести их реинжиниринг. Поэтому удобнее всего использовать именно эту нотацию для описания бизнес-процессов предприятия.

Данная нотация является подходящей для описания бизнес-процессов предприятия по ряду причин:

- обеспечение возможности обмена информацией о рассматриваемом объекте на языке, понятном не только аналитику и разработчику системы, но и специалисту-эксперту в предметной области, пользователю, руководителю и любому сотруднику фирмы;
- описание бизнес-процессов осуществляется в форме, которая проста для восприятия сотрудников компании в специализированных программах (таких как Business Studio, Ramus Educational);
- нотация является экономически эффективной, так как программное обеспечение, позволяющее использовать нотацию, имеет не высокую цену либо распространяется бесплатно;
- набор элементов программ для описания процессов, а также возможности для создания отчетов, позволяют использовать возможности программ в полной мере и по назначению.

Методология IDEF0 использует метод функционального моделирования сложных систем, являющийся частью методологии структурного анализа и проектирования SADT (Structured Analysis and Design Technique). IDEF0

позволяет строить функциональные модели, которые описывают бизнес-процессы в виде иерархической системы взаимосвязанных функций.

Моделирование бизнес-процессов затрагивает многие аспекты деятельности объекта исследования:

- изменение внутренних нормативных документов и технологии проведения операций;
- изменение организационной структуры;
- новые требования к автоматизации выполняемых процессов и т. д.;
- оптимизацию функций подразделений и сотрудников;
- перераспределение прав и обязанностей руководителей.

Для моделирования сложных систем существуют определенные методологии и стандарты. К таким стандартам относятся методологии семейства IDEF. С их помощью можно эффективно отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах. При этом широта и глубина обследования процессов в системе определяется самим разработчиком, что позволяет не перегружать создаваемую модель излишними данными. Методология IDEF0 является эффективным средством анализа, проектирования, а также представления деловых процессов. Структурной единицей модели IDEF0 есть диаграмма, которая представляет собой графическое описание модели предметной области, а также и ее части. Основным компонентом диаграммы IDEF0 является блок. Части функционального блока представляют собой стороны прямоугольника, каждая из которых имеет разное назначение, так левая сторона – это вход, правая – выход, верхняя управление, нижняя – механизмы [3].

Модель IDEF0 всегда начинается с представления системы как единого целого - одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой, и обозначается идентификатором “A-0”. В процессе декомпозиции, функциональный блок, который в контекстной диаграмме отображает систему

как единое целое, подвергается детализации на другой диаграмме. Получившаяся диаграмма второго уровня содержит функциональные блоки, отображающие главные подфункции функционального блока контекстной диаграммы и называется дочерней по отношению к нему (каждый из функциональных блоков, принадлежащих дочерней диаграмме, соответственно называется дочерним блоком). В свою очередь, функциональный блок - предок называется родительским блоком по отношению к дочерней диаграмме, а диаграмма, к которой он принадлежит - родительской диаграммой.

Для создания моделей бизнес-процессов построения пассажирских маршрутов использована программа «AllFusion Process Modeler». Это инструмент для моделирования, анализа, документирования и оптимизации бизнес-процессов. «AllFusion Process Modeler» можно использовать для графического представления бизнес-процессов.

С помощью программы можно управлять сложными бизнес-процессами путем автоматизации процесса проектирования и ряда диаграмм для разных технологий проектирования.

Программа является мощным инструментов моделирования, которая разработана компанией Computer Associates Technologies. Программа используется для анализа, документирования и реорганизации сложных бизнес-процессов и автоматизирует ряд задач, которые обычно связанные с построением моделей процессов, обеспечивая семантическую точность, необходимую для гарантии правильных и согласованных результатов [10].

Модель, созданная средствами BPwin, позволяет четко документировать различные аспекты деятельности - действия, которые необходимо предпринять, способы их осуществления, требующиеся для этого ресурсы.

Модели в программе предоставляют возможность изучить и проанализировать бизнес-процессы и события, происходящие в рамках изучаемых процессов.

На рисунках 4.4 и 4.5 представлены диаграммы составления расписания по маршрутам пассажирских перевозок.

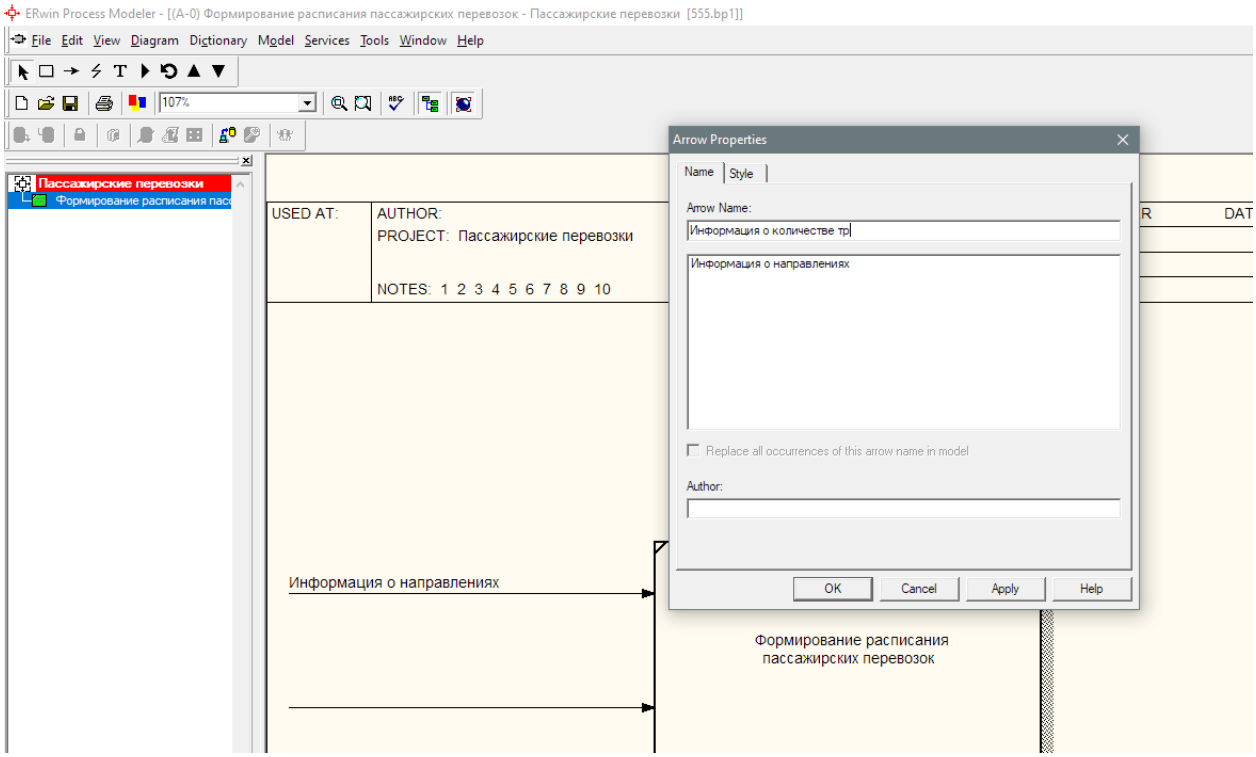

Рисунок 4.4 - Рабочее окно программы ERWin Process Modeler

Программа AllFusion Process Modeler включает три стандартные методологии: IDEF0 (функциональное моделирование), DFD (моделирование потоков данных) и IDEF3 (моделирование потоков работ). Эти методологии по-своему уникальны. Каждая из них может быть выполнена отдельно с помощью Process Modeler, но их совокупность заключённая в модель даёт аналитику полную картину предметной области клиента.

Программа отслеживает связи в диаграммах, сохраняя их целостность при внесении изменений в модель. Динамическая "подсветка" объектов служит подсказкой при построении модели и предостерегает от повторения распространенных ошибок в моделировании. Кроме этого, программа поддерживает заданные пользователем свойства, что позволяет вносить соответствующую вашим потребностям информацию.

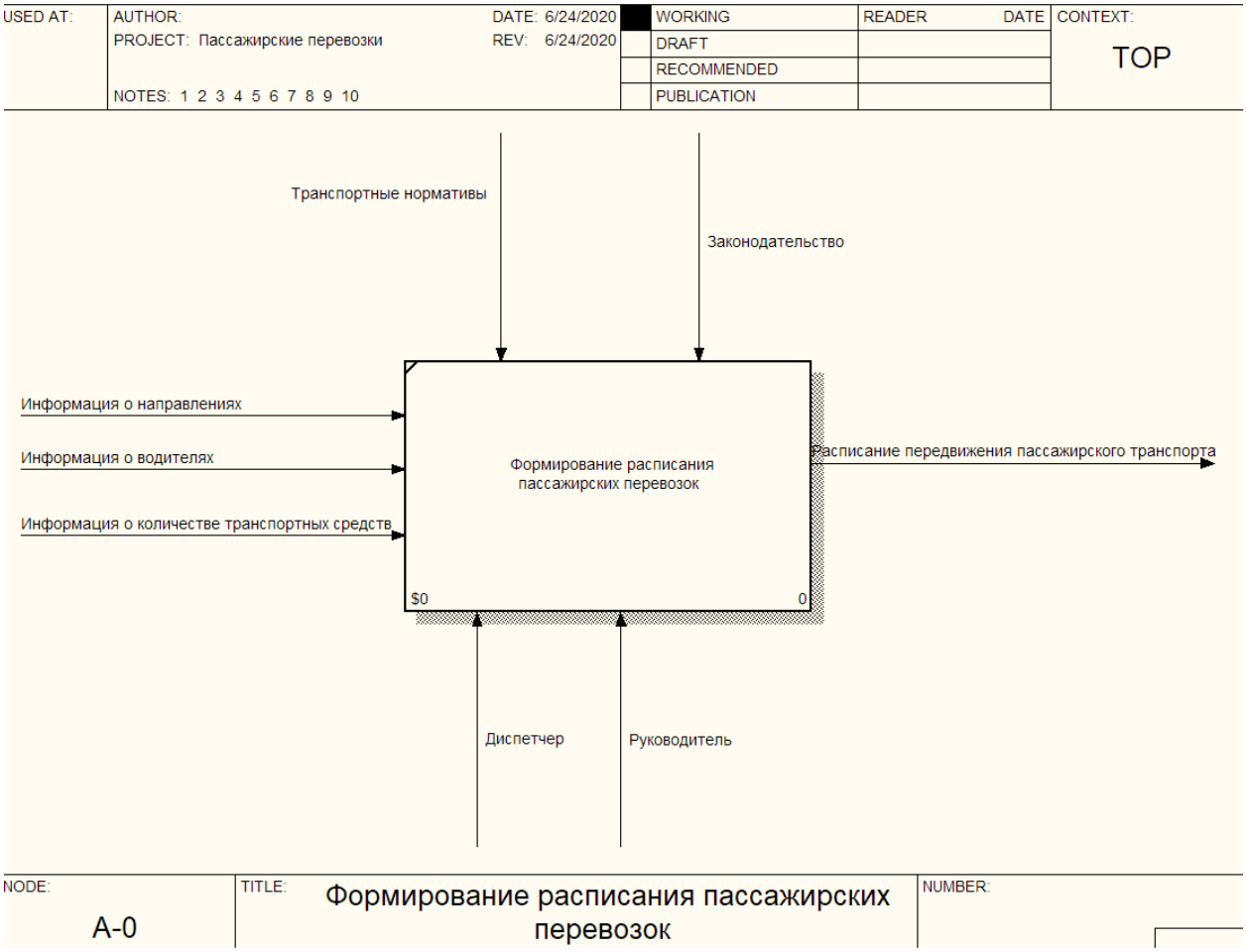

Рисунок 4.5 – Контекстная диаграмма бизнес-процесса составления расписания

4.3.2 Проектирование базы данных

Система управления базами данных (СУБД) представляет собой набор взаимосвязанных данных и набор программ для доступа к этим данным. Коллекция данных, обычно называемая базой данных (БД), содержит информацию, относящуюся к специфическому уровню бизнес-логики программного обеспечения. Основная цель СУБД состоит в том, чтобы обеспечить удобный и эффективный способ хранения и извлечения информации из БД.

Сегодня выбор СУБД может играть критическую роль в разработке ПО. Объемы накопленных человечеством данных, нашей «новой нефти», в ближайшие годы будут стремительно увеличиваться, по оценке IDC — с 33 зеттабайт (зеттабайт — 10 в 21 степени байт) в 2018 году до 175 зеттабайт в 2025-

м. Соответственно, растет и роль систем управления базами данных как центрального звена корпоративной инфраструктуры [16].

MySQL — это система управления реляционными базами данных (РСУБД). MySQL используется большинством современных программ, приложений, веб-сайтов и веб-сервисов как удобное и быстрое решение для хранения и поиска больших объемов данных. Простым примером элементов, которые могут храниться в БД MySQL, может быть имя зарегистрированного на сайте пользователя с соответствующим паролем (зашифрованным для безопасности), дата регистрации пользователя, количество посещений и т. д. Доступ к управлению БД с помощью MySQL возможен с помощью многих инструментов. С БД MySQL можно связаться через PHP, Javascript, с помощью других языков программирования.

БД — это структурированный набор данных, который используется различными программами, приложениями, прикладными системами конкретного предприятия, который управляется системой управления базами данных (СУБД). В общем случае БД можно представить как набор таблиц, которые связаны друг с другом. Язык структурированных запросов (SQL) — это язык БД. SQL поддерживает небольшой, но очень мощный набор операторов для управления данными в БД, а также их защиты. Почти каждый сервер БД поддерживает SQL, или же диалект этого языка. В настоящее время продукты SQL доступны для компьютеров любого типа, от небольших портативных компьютеров до больших серверв, а также для любых операционных систем, включая Microsoft Windows, Mac и многие дистрибутивы UNIX. Язык структурированных запросов (SQL) — это язык реляционных БД, который позволяет создавать, удалять БД, обращаться к БД и манипулировать ими. Ниже приведен список основных операций, которые можно сформулировать с помощью запросов SQL:

- создание новых БД;
- удаление БД;
- создание новых таблиц в БД;

- удаление таблиц из БД;
- создание и удаление пользователей (контроль доступа к БД);
- выполнение запросов к БД для извлечения данных;
- создание записей в БД;
- создание хранимых процедур в БД;
- установка разрешений для таблиц и процедур;
- создание отношений между таблицами.

SQL — наиболее популярный язык для серверов реляционных БД. К концу 1986 года использование языка SQL в качестве основного для работы с данными в СУБД стало практически повсеместным. IBM, Oracle, Sybase и Gupta использовали схожий синтаксис языка SQL для отправки сообщений от клиентской части СУБД (front end) к серверной (back end), что позволяло сочетать клиентские и серверные части разных производителей [20]. Реляционная модель данных была предложена в 1970 г. математиком Эдгаром Коддом (Codd E.F.). РСУБД является наиболее широко распространенной моделью данных и единственной из трёх основных моделей данных, для которой разработан теоретический базис с использованием теории множеств. Базовой структурой РСУБД является отношение, основанное на декартовом произведении доменов. Домен – это множество значений, которое может принимать элемент данных (например, множество целых чисел, множество дат, множество комбинаций символов длиной N и т. п.). Домен может задаваться перечислением элементов, указанием диапазона значений, функцией и т. Д [9].

СУБД MySQL сравнительно просто интегрируется в процесс создания ПО с использованием различных сред разработки и языков программирования. PHP — это серверный, встроенный в HTML язык сценариев, который может использоваться для создания динамических веб-страниц. PHP доступен для большинства операционных систем и веб-серверов и может обращаться к большинству распространенных БД, включая MySQL. PHP может быть запущен как отдельная программа или скомпилирован как модуль для использования с веб-сервером. В зависимости от версии PHP, есть 2-3 PHP API для доступа к БД

MySQL. Пользователи PHP 5 имеют возможность выбора между устаревшим расширением mysql, mysqli или PDO_MySQL. PHP 7 не поддерживает расширение mysql, оставляя только последние два варианта используемых инструментов.

Одной из важнейших современных технологий веб-разработки является Javascript. Для того, чтобы использовать Javascript вместе с MySQL, требуется использование дополнительного middleware-софта. Так, модуль Node.js с названием Knex.js позволяет создавать запросы к БД прямо в коде Javascript. Knex.js также предоставляет возможность использования таких возможностей БД как проведение транзакций, создание точек сохранения (savepoints) БД и др. Использование дополнительного middleware-обеспечения, такого как axios (модуль для Node.js) позволяет отправлять запросы SQL с клиента приложения на сервер. Таким образом, сегодня практически не существует ни одной важной технологии разработки ПО, которая не могла бы быть успешно интегрирована с СУБД MySQL.

Однако, существуют и некоторые ограничения использования MySQL. БД MySQL показывает низкую эффективность при использовании ее как хранилища данных, это отчасти связано с неспособностью использовать несколько процессоров для обработки одного запроса. К тому же MySQL часто критикуют за то, что эта СУБД имеет различия со стандартом SQL относительно интерпретации значения NULL и т. д. Существуют значительные проблемы безопасности данной СУБД, которые могут быть успешно эксплуатировала злоумышленниками. Так, существует возможность авторизации с неправильным паролем с вероятностью 1/256, так как MySQL считает, что пришедший токен от пользователя и ожидаемое значение равны Точек вхождения в чужую базу MySQL не так уж и много по сравнению с другими СУБД: SQL Injection, поиск логинов и паролей на GitHub, брутфорс, уязвимость к багам из паблика. К методам постэксплуатации можно еще дополнительно отнести повышение привилегий (goo.gl/UM5L6R), DoS-атаки (goo.gl/q7VDuY), применение триггеров и хранимых процедур. Правда, отдельные из них относятся к частным

случаям, которые можно встретить довольно редко либо для которых нужны очень специфичные условия [8].

MySQL — одна из наиболее популярных СУБД. Вместе с тем, на рынке СУБД присутствует значительная конкуренция. В зависимости от совокупных характеристик (преимуществ и недостатков, ограничений) продукта, его ориентированности на выполнение конкретного функционала и задач, а также особенностей лицензирования и использования, пользователи выбирают тот или иной продукт СУБД. Некоторые из сравнительных характеристик современных СУБД на базе SQL (реляционного типа) представлены в Таблице 2.

Таблица 4.2 - Сравнение СУБД MySQL с другими SQL (реляционными) СУБД

Название СУБД	Преимущества	Недостатки	Тип, характеристики
SQLite	легковесность (менее 600 КБ пространства) минимальная конфигурация одна база — один файл	рекомендуется для небольших БД (до 1 ТБ) только одна операция с базой в данный промежуток времени; serverless-архитектура: нужно online подключение	Тип: реляционная; свободное ПО; макс. размер БД: 140 ТБ; макс. кол-во колонок — 2 000
MySQL	простота использования; встроенный скрипт безопасности; быстрота; возможность репликации	ограниченный SQL (отсутствие FULL JOIN и др.); свободное ПО MySQL более ограниченное, чем платное; снижение темпов разработки	Тип: реляционная; свободное ПО, проприетарное; макс. размер БД: 65,536ТБ; макс. кол-во колонок — 1024, 30000
PostgreSQL	максимальная совместимость SQL; расширяемость (дополнения); большое коммьюнити и поддержка (свободная разработка)	ограничения памяти, выделенной на процесс (10Мб); непопулярная СУБД на хостингах; сложная настройка (необходимая для целостности данных и т. д.)	Тип: объектно-реляционная; свободное ПО; макс. размер БД: макс. размер БД: нет; макс. кол-во колонок — 250-1600
Oracle	сложная система безопасности;	высокая стоимость продукта; сложность конфигурации	Тип: объектно-реляционная;

	внешняя аутентификация; полнотекстовый поиск; наличие развитых инструментов отладки, администрирования		коммерческая лицензия; макс. размер БД: 128 ТБ; макс. кол-во колонок — 1000

Следует отметить, что несколько иную ветвь программных продуктов — СУБД — составляют нереляционные СУБД (NoSQL). Основным отличием реляционных и нереляционных СУБД является организация хранения данных. Если в реляционных СУБД данные хранятся в таблицах, в нереляционных СУБД нет связи между таблицами (данные могут храниться даже в файлах или в иерархических структурах). Простейшее представление данных в таких СУБД — это «ключ-значение». Примеры NoSQL СУБД: MongoDB, Redis, Cassandra, HBase, CouchDB, GemFire и др [12].

Таблица 4.3 - Сравнение СУБД MySQL с NoSQL (нереляционными) СУБД

Тип СУБД	MySQL	NoSQL
Тип	Реляционная БД	Нереляционная БД
Дизайн	Основное понятия — Table (таблица)	Основное понятие — Document (документ)
Масштабируемость	В целом сложный процесс для больших БД	Более легкое масштабирование больших БД
Модель	Подробная модель БД должна быть предоставлена до ее создания	Нет необходимости разрабатывать подробную модель БД
Сообщество	Обширное и экспертное сообщество	Сообщество относительно мало по сравнению с MySQL
Стандартизация	SQL это стандартный язык	Отсутствие стандартного языка запросов
Схема	Схема жесткая	Динамическая схема является ключевым преимуществом NoSQL
Гибкость	Не очень гибкий дизайн, новая вставка столбца или поля влияет на дизайн	Новый столбец или поля могут быть вставлены без изменения дизайна

В настоящее время БД типа NoSQL становятся основной частью ландшафта рынка БД, и, обладая некоторыми преимуществами, они могут стать триггером изменений в данной области в целом. Более низкая стоимость, более легкая масштабируемость и функции для с открытым исходным кодом делают

NoSQL привлекательным вариантом для многих компаний, желающих интегрироваться в такие области разработки ПО как, например, Big Data. Тем не менее, NoSQL — все еще относительно молодая технология без набора стандартов, которые предлагают СУБД SQL, такие как MySQL. В конце концов, выбор между NoSQL и SQL зависит от сложных бизнес-потребностей организации или индивидуального пользователя, а также от объема и разнообразия данных, которые используются БД.

Для работы с MySQL предназначена среда MySQL Workbench — это графический инструмент для работы с серверами и БД MySQL. MySQL Workbench полностью поддерживает сервер MySQL версии 5.6 и выше. Данный продукт также совместим со старыми версиями MySQL server 5.x, за исключением определенных случаев (например, отображение списка процессов) из-за изменения системных таблиц. Инструмент не поддерживает версии сервера MySQL 4.x.

Функциональность MySQL Workbench охватывает пять основных направлений использования продукта:

- разработка с использованием языка SQL — позволяет создавать и управлять подключениями к серверам БД. Наряду с возможностью настройки параметров соединения, MySQL Workbench предоставляет возможность выполнять SQL-запросы к соединениям с БД с помощью встроенного редактора SQL (см. Рисунок 1);
- моделирование данных — позволяет создавать модели-схемы БД графически, осуществлять обратный и прямой инжиниринг между схемой и действующей БД, а также редактировать все аспекты БД с помощью редактора таблиц. Редактор таблиц предоставляет простые в использовании средства для редактирования таблиц, столбцов, индексов, триггеров, параметров, вставок и привилегий, подпрограмм и представлений;
- администрирование сервера: позволяет администрировать экземпляры сервера MySQL путем администрирования

пользователей, выполнения резервного копирования и восстановления, проверки данных и аудита, просмотра состояния БД и мониторинга производительности сервера MySQL;

- миграция данных — позволяет выполнять миграцию из Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostgreSQL и других реляционных СУБД, объектов и данных в MySQL. Миграция также поддерживает переход с более ранних версий MySQL на последние версии;
- поддержка MySQL Enterprise: поддержка корпоративных продуктов, таких как MySQL Enterprise Backup, MySQL Firewall и MySQL Audit.

Среда работы с СУБД MySQL MySQL Workbench доступна в двух версиях: Community Edition (версия сообщества) и Commercial Edition (коммерческая версия). Community Edition предполагает бесплатный доступ к продукту. Коммерческая версия предоставляет дополнительные корпоративные функции, такие как доступ к MySQL Enterprise Backup, MySQL Firewall и MySQL Audit. MySQL Workbench Community Edition – интегрированная среда для проектировщиков, разработчиков и администраторов БД, реализующая функции визуального проектирования, разработки и эксплуатации БД MySQL (рисунок 4.6). Предшественником данного программного продукта является DBDesigner 4 от FabForce. Программа распространяется под свободной лицензией GNU GPL. Разработчик – компания Oracle [5].

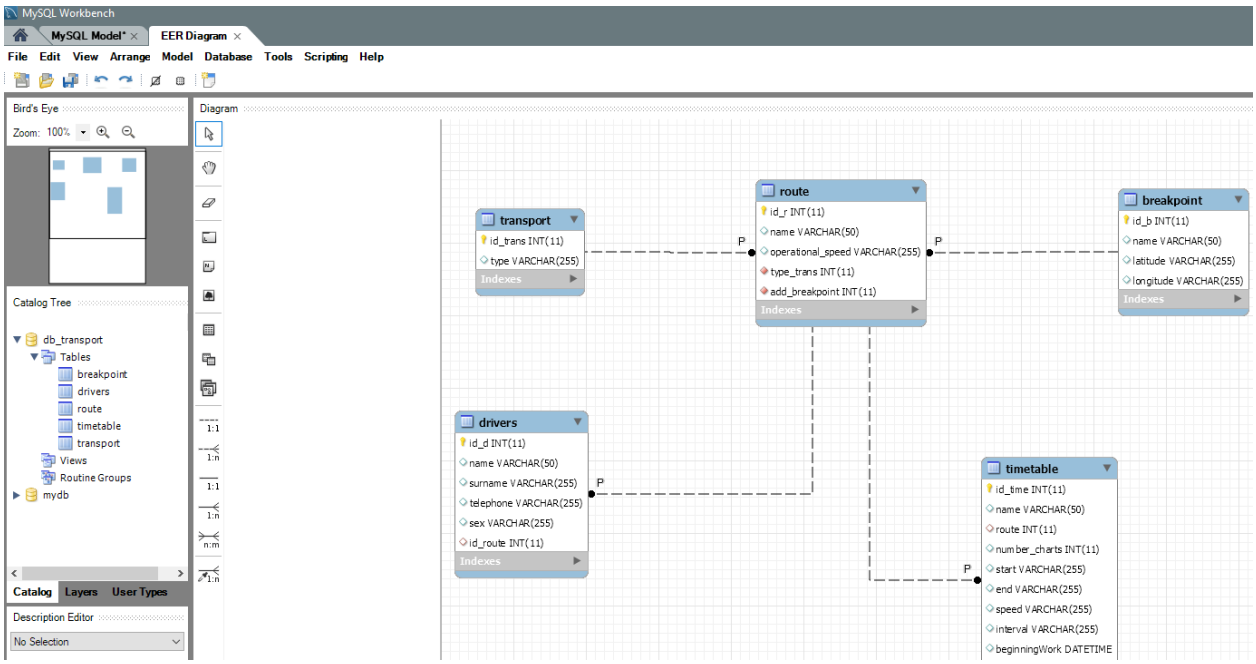

Рисунок 4.6 - Среда разработки MySQL Workbench (окно моделирования)

MySQL Workbench [21] — далеко не единственный в своем роде инструмент графически-ориентированного управления СУБД. Схожий функционал предоставляют продукты Microsoft SQL Server Management Studio, MySQL Workbench, Oracle Enterprise Manager, pgAdmin, phpMyAdmin и др. Преимуществами графического интерфейса управления БД являются возможность влиять на логику БД без предварительной сложной конфигурации, возможность быстро получать информацию о состоянии БД и производимых в ней операциях, ролях пользователей, структурах БД и т. д. Вместе с тем, полноценное управление БД MySQL с помощью исключительно графических средств, вероятно, не может быть достигнуто.

Таким образом, система управления базами данных (СУБД) должна удовлетворять требованиям простоты использования, бесплатности (доступности), стандартизации, защищенности и наличии достаточного количества справочного материала. Поэтому, множество пользователей (индивидуальных, корпоративных) делают выбор в сторону в пользу MySQL. Во-первых, данная СУБД является бесплатным open-source продуктом, то есть не оказывает никакой финансовой нагрузки при ее использовании. Во-вторых, сопутствующий продукт MySQL Workbench позволяет легко взаимодействовать

с установленной БД, предоставляя возможность через удобный интерфейс пользователя выполнять любые манипуляции с данными. И наконец, эта СУБД является одной из наиболее распространенных, занимая второе место по популярности в мире, уступая только корпоративной СУБД от Oracle. Именно поэтому в сети имеется большое количество материалов, как справочных, включая официальным сайтом, так и учебных, где рассматривается широкий спектр проблем и пути их решения при использовании данной БД,

Вместе с тем MySQL — это еще один продукт на рынке СУБД, пусть и один из наиболее популярных. Так, постепенное развитие других СУБД, а также целых NoSQL-парадигм может привести к постепенному снижению количества пользователей MySQL. Разумеется, во многом это обстоятельство может быть обусловлено и сменой парадигмы разработки ПО, и появлением специфических потребностей массового пользователя, таких как поддержка более сложных backend-систем, распространение распределенных систем различного типа и т. д.

Так или иначе, основные преимущества MySQL остаются неоспоримыми. Среди таких преимуществ:

- открытый исходный код и бесплатность;
- продуманность и быстродействие;
- относительная легковесность — занимает немного дискового пространства;
- кроссплатформенность — работает на многих операционных системах, таких как Windows, Unix-подобных ОС и др.;
- наличие множества материалов, в которых рассматривается решения определенных вопросов и проблем, возникающих при работе с СУБД;
- совместимость с небольшими приложениями (важно для индивидуальных пользователей.

Как было отмечено выше, для проектирования базы данных использованы возможности среды MySQL Workbench, которая была использована для физического моделирования. Логическая модель базы данных создана с помощью Software Ideas Modeler.

Процесс проектирования базы данных состоит из концептуального, логического и физического уровней. Концептуальный уровень – это сущности, атрибуты, связи. Концептуальная модель – это модель, которая отображает знания в предметной области об ее объектах и их взаимосвязях, процессах и результатах деятельности. На этом этапе могут быть использованы тексты, таблицы, графики, графы, блок–схемы.

Логическое проектирование – процесс преобразования требований к данным в структуре данных. На выходе получается СУБД–ориентированная структура базы данных и спецификации прикладных программ. На этом этапе могут моделироваться базы данных относительно различных СУБД и производиться сравнительный анализ моделей. Логический уровень (представление программиста) это записи, элементы данных, связи между записями.

Большинство современных подходов к проектированию баз данных базируется на использовании разновидностей так называемой ER–модели. Моделирование предметной области основывается на использовании графических диаграмм, которые включают небольшое число компонентов разного рода.

Основными понятиями ER–модели являются сущность, связь и атрибут. Сущность – это реальный или представляемый объект, информация о котором должна быть сохранена и доступна. В диаграммах ER–модели сущность представлена в виде прямоугольника, который содержит имя сущности. Каждый экземпляр сущности должен отличаться от любого другого экземпляра той же сущности. Связь – это графически представляемая ассоциация, которая устанавливается между двумя сущностями. В любой связи выделяются два конца, на каждом из которых должно быть указано имя конца связи, степень конца связи и обязательность связи. Атрибутом сущности является любая деталь, служащая для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности. Имена атрибутов

записываются в прямоугольник, который изображает сущность, под именем сущности и изображаются малыми буквами.

Уникальным идентификатором сущности может быть атрибут, комбинация атрибутов, комбинация связей или комбинация связей и атрибутов, которая уникально отличает любой экземпляр сущности от других экземпляров сущности того же типа.

На рисунке 4.7 приводится логическая структура модели базы данных составления расписания пассажирских перевозок.

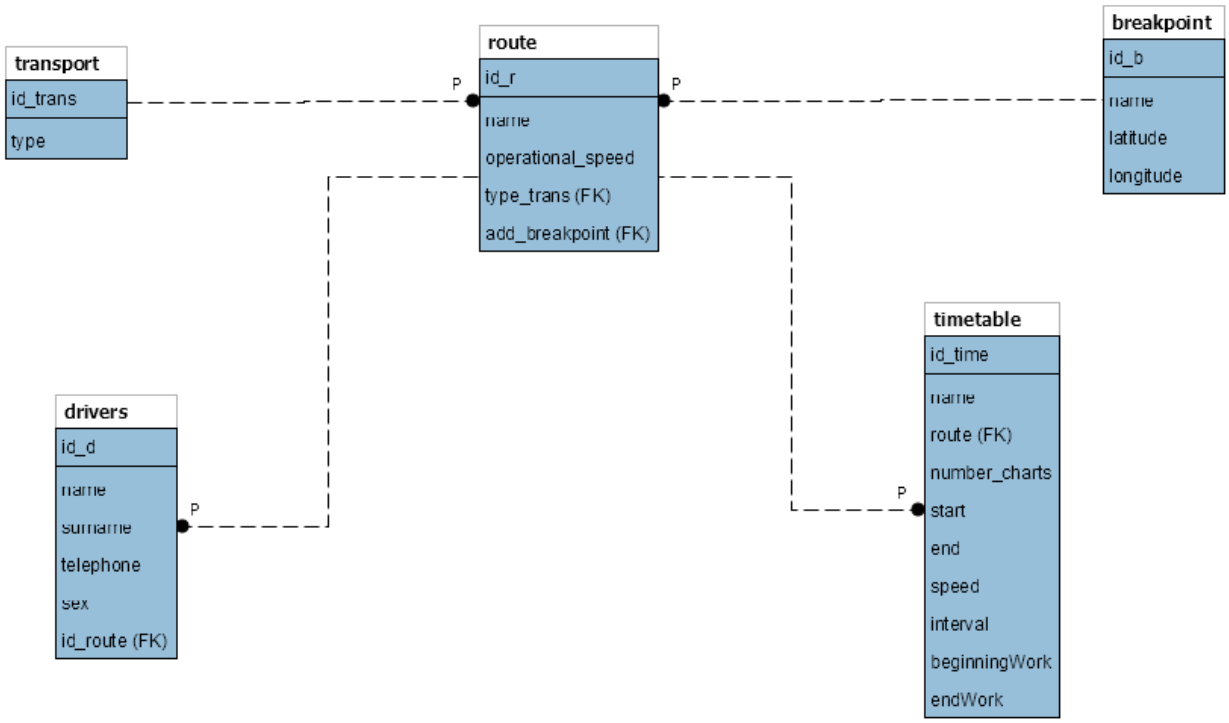

Рисунок 4.7 - Логическая модель базы данных

Рисунки 4.8 – 4.12 отображают физическое построение базы данных, разрабатываемого модуля.

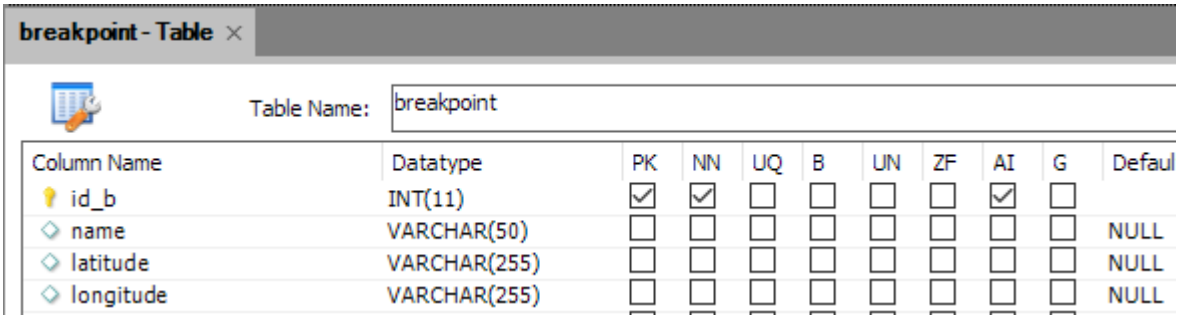

Рисунок 4.8 - Создание таблицы Breakpoint

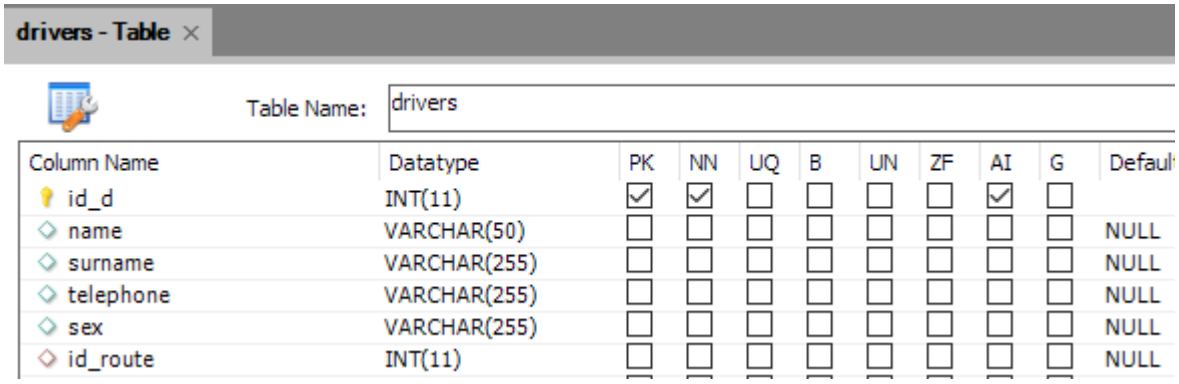

Рисунок 4.9 - Создание таблицы drivers

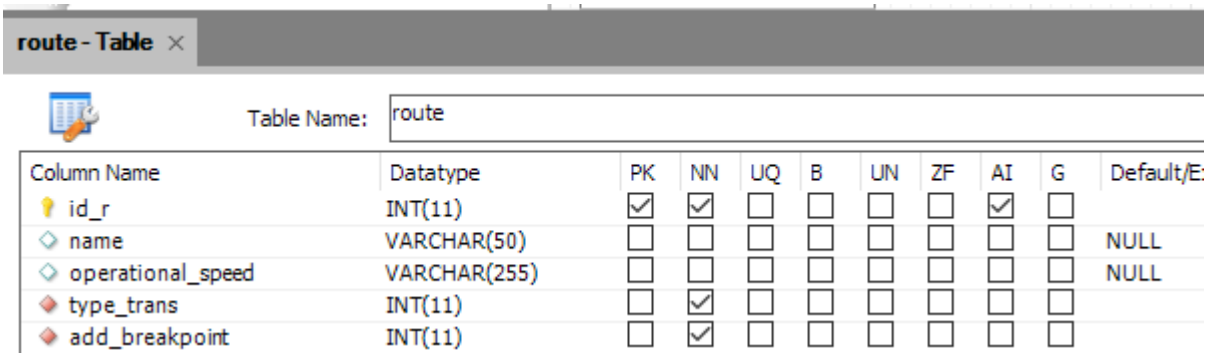

Рисунок 4.10 - Создание таблицы route

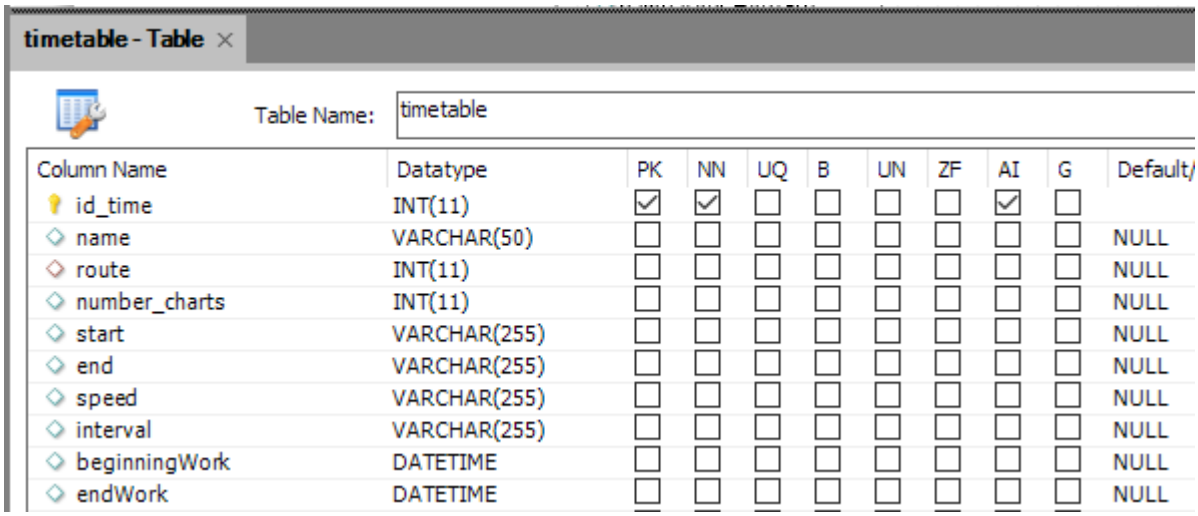

Рисунок 4.11 - Создание таблицы timetable

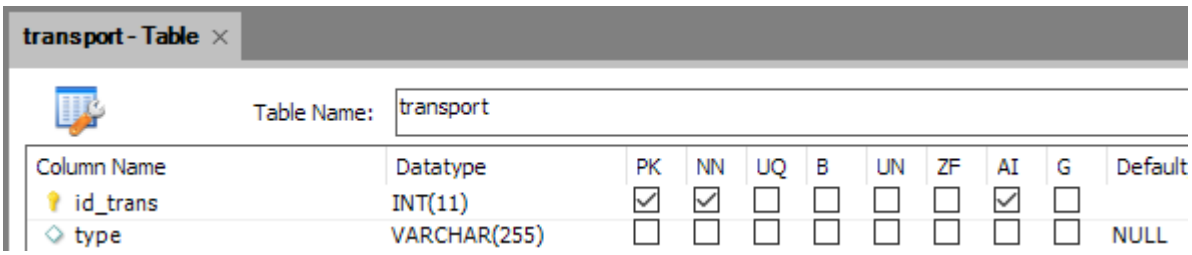

Рисунок 4.12 - Создание таблицы transport

После моделирования таблиц были построены связи между ними и создана физическая модель база данных (рисунок 4.13).

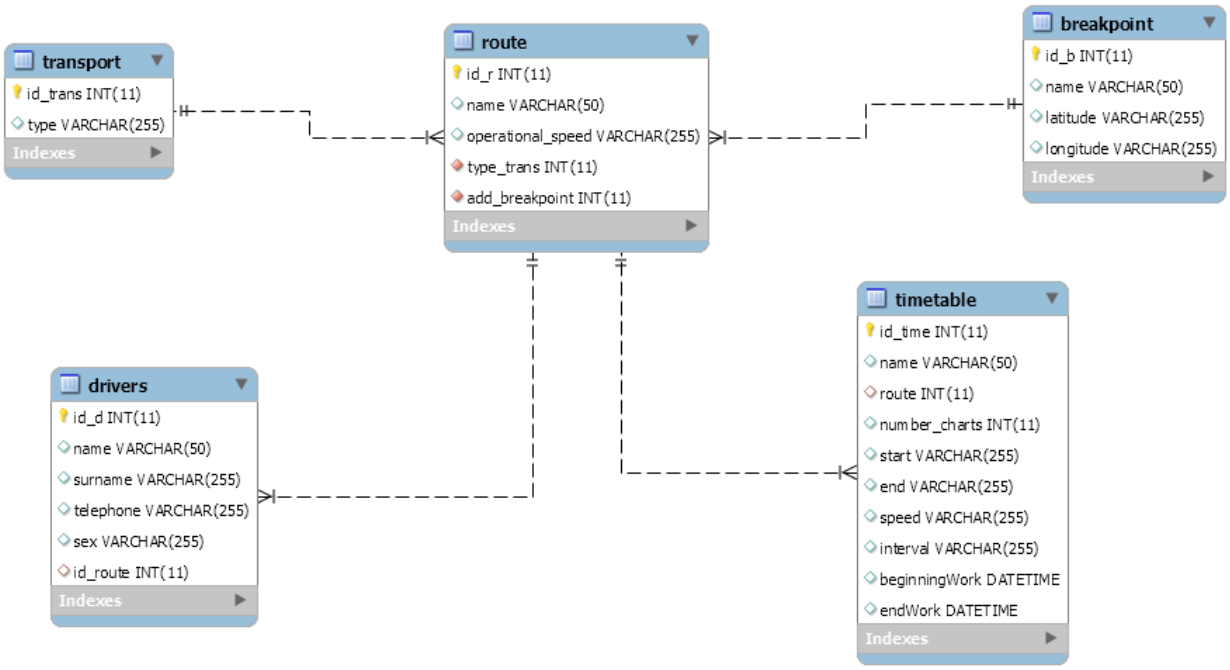

Рисунок 4.13 - Физическая модель базы данных

4.4 Разработка программного средства прогнозирования расписания общественного транспорта

4.4.1 Проектирование структуры и функционала модуля

После проектирования базы данных была спроектирована структура проекта, которая показана на рисунке 4.14.

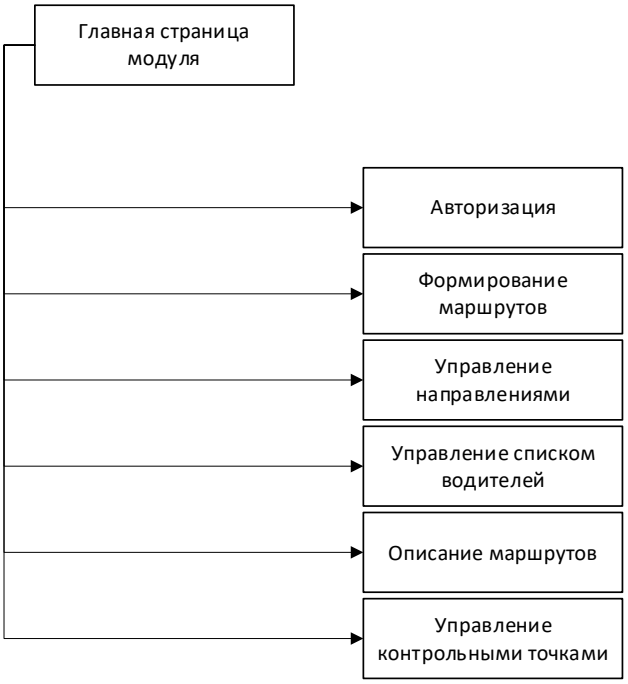

Рисунок 4.14 – Структура модуля

Функционал системы отображен на алгоритмах, которые показаны на рисунках 4.15 – 4.18. Алгоритм работы приложения переход в главное окно приведена на рисунке 4.15.

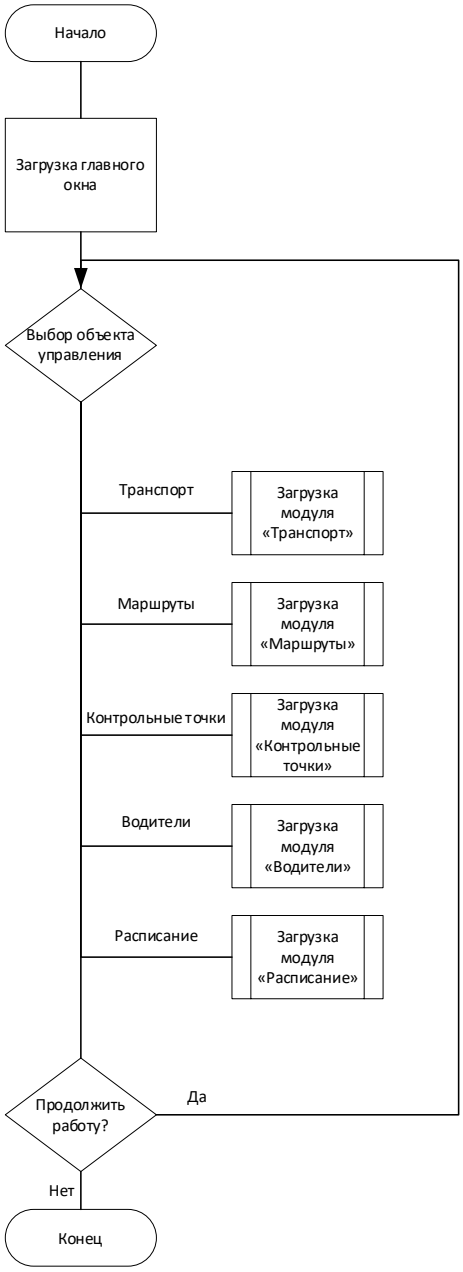

Рисунок 4.15 - Главное окно

Схема управления расписанием показана на рисунке 4.16.

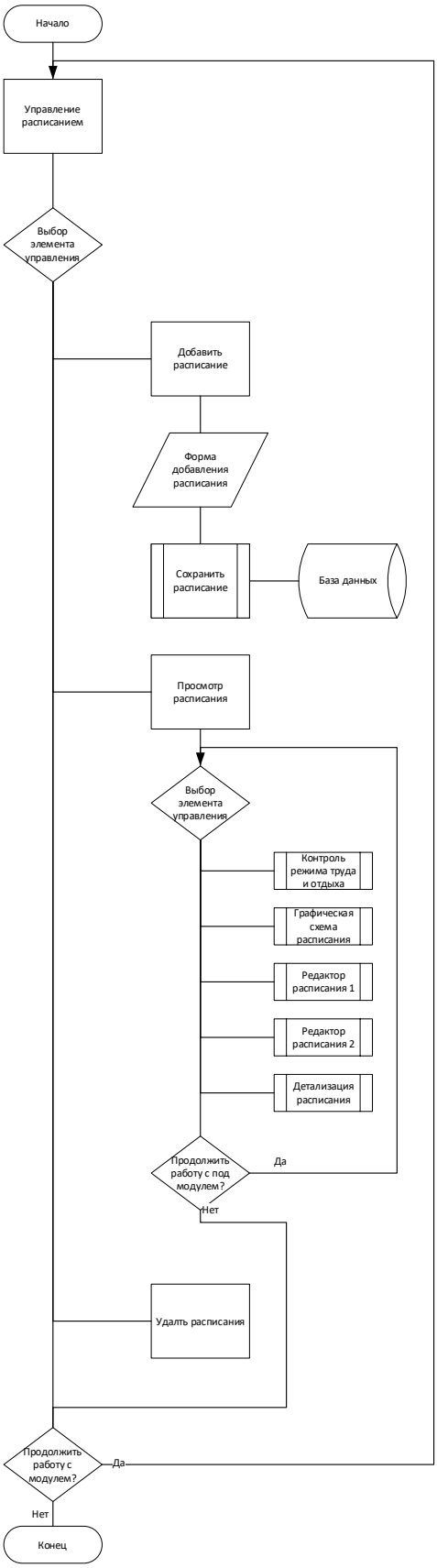

Рисунок 4.16 - Алгоритм управление расписанием

Алгоритм управления маршрутами показан на рисунке 4.17.

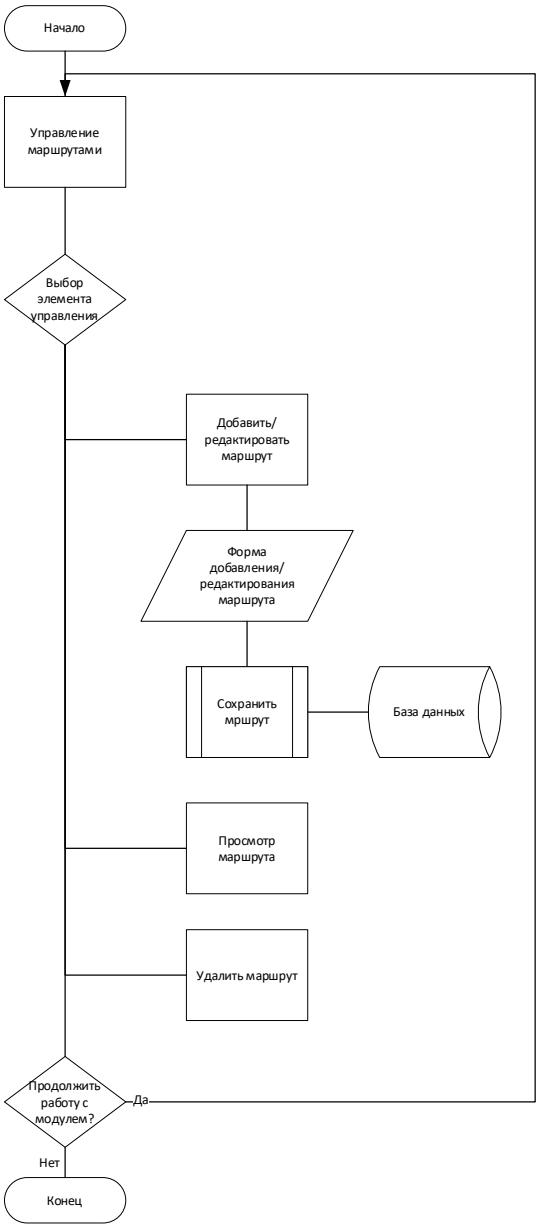

Рисунок 4.17 - Управление маршрутами

Схема управления контрольными точками отображена на рисунке 4.18.

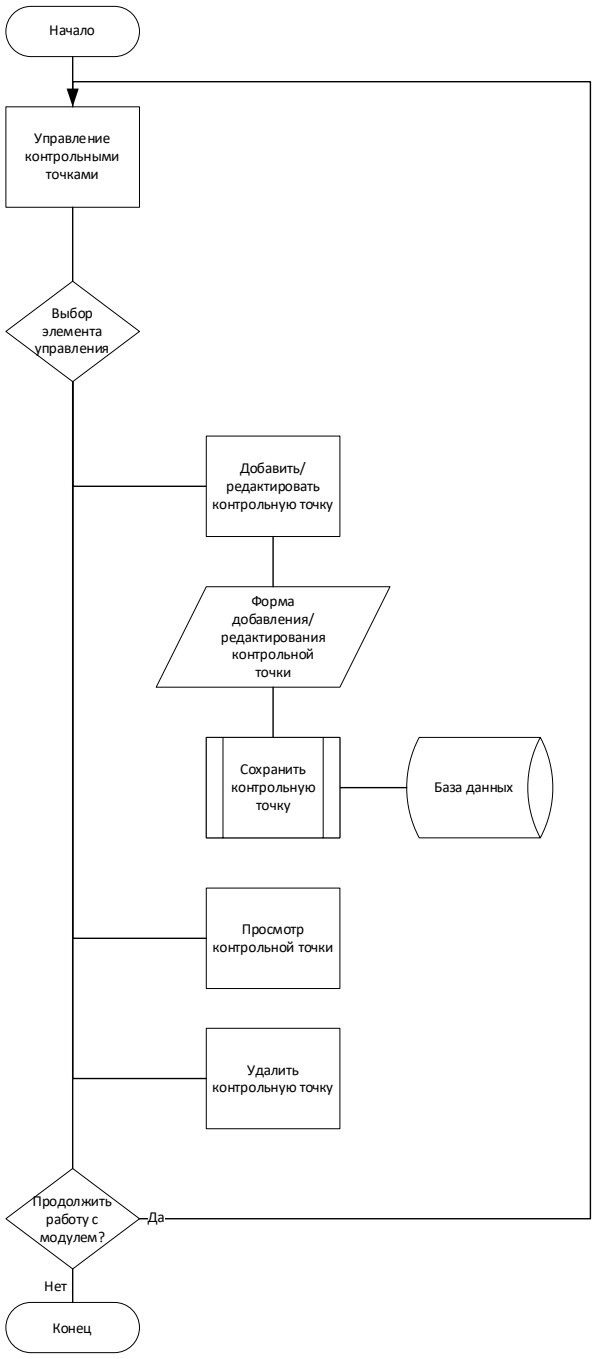

Рисунок 4.18 - Управление контрольными точками

4.4.2 Выбор архитектурного решения

В основу архитектуры проекта выбрана клиент-серверная архитектура. В базовой модификации клиент-сервер все процессы в распределенных системах разделяются на две вероятно перекрывающиеся группы. Процессы, реализующие определенную службу, к примеру, службу файловой системы или базы данных, называются серверами (servers). Процессы, запрашивающие

службы у серверов методом посылки запроса и дальнейшего ожидания отклика от сервера, называются клиентами (clients). Для реализации проекта выбрана трёхуровневая архитектура.

В трехуровневой архитектуре программы, образующие часть уровня обработки, выносятся на самостоятельный сервер, но дополнительно могут частично находиться и на компьютерах клиентов и серверов. Архитектура формируется из последующих модулей:

– клиент — это интерфейсный (обычно графический) элемент, который представляет первый уровень, в сущности, приложение для конечного пользователя. Первый уровень не должен характеризоваться прямыми связями с базой данных (по требованиям безопасности), быть нагруженным главной бизнес-логикой (по требованиям масштабируемости) и сохранять состояние приложения (по требованиям надежности). На первый уровень может быть вынесена и традиционно выносится простая бизнес-логика: интерфейс авторизации, алгоритмы шифрования, наладка вводимых значений на допустимость и соответствие формату, несложные операции (сортировка, группировка, подсчет значений) с информацией, уже загруженными на терминал.

– сервер приложений располагается на втором уровне. На втором уровне сосредоточена очень крупная часть бизнес-логики. За пределами его держатся фрагменты, экспортируемые на терминалы, а также погруженные в третий уровень хранимые процедуры и триггеры.

– сервер базы данных гарантирует хранение данных и выносится на третий уровень. Традиционно это стандартная реляционная или объектно-ориентированная СУБД. Если третий уровень предполагает собой базу данных вместе с хранимыми процедурами, триггерами и схемой, описывающей приложение в терминах реляционной модели, то второй уровень основывается как программный интерфейс, объединяющий клиентские компоненты с прикладной логикой базы данных.

Схема структуры показана на рисунке 4.19.

Рисунок 4.19 - Трехуровневая архитектура

Взаимодействие компонентов в данной архитектуре происходит благодаря использованию протокола HTTP (HyperTextTransfer Protocol). Он представляет собой протокол прикладного уровня для передачи гипертекста. Является протоколом высокого уровня, который используется в сетях TCP/IP для извлечения файлов с соответствующих ресурсов. При этом HTTP не хранит информацию о своем состоянии: каждая транзакция рассматривается независимо от остальных. Следовательно, в типичной реализации создается новое TCP-соединение между клиентом и сервером для каждой транзакции, которое прерывается сразу после ее завершения.

Для реализации проекта выбран язык программирования PHP. Язык программирования PHP специально разработана для создания веб-приложений. Имеет разные функции, которые предназначены для работы с сервером. Не имеет избыточные данные и методы. - один из немногих языков программирования, созданных специально для разработки веб-приложений. Поэтому он включает в себя все функции, необходимые именно для работы на веб-сервере, и при этом лишен избыточности, свойственной многим его конкурентам.

Очень приятная особенность PHP - то, что его команды включаются в обычные HTML-страницы с помощью специальных тегов, которые и заставляют PHP-машину выполнять на сервере нужные действия. Программам на PHP не нужны специальные CGI-директории с особыми правами доступа. Более того, на одной страничке можно произвольно чередовать "простой" HTML и PHP-код.

PHP не зависит от платформы. PHP прекрасно интегрируется во все популярные веб-серверы: Apacce и IIS, Zens и Netscape Enterprise Server, работает

под Windows и OS/2, MacOS и практически всеми UNIX-подобными системами. Как следствие - PHP работает практически у всех хостеров, разрешающих собственные выполняемые скрипты.

Замечательная особенность PHP - его интегрированность практически со всеми современными интернет-технологиями. PHP поддерживает большинство современных веб-протоколов: IMAP, FTP, POP, XML, SNMP и другие. PHP прекрасно работает с базами данных. Трудно найти СУБД, поддержка которой не была бы реализована в PHP. MySQL и MS SQL Server, PostgreSQL и Oracle, Sybase и Interbase... Один только перечень баз данных, поддерживаемых PHP, займет, наверное, целый экран.

PHP включает в себя огромное количество встроенных функций: обработки строк и массивов, работы с файловой системой и с HTTP, электронной почтой, датой и временем, кириллицей и другими национальными алфавитами... Когда я впервые начал программировать на PHP, то был просто поражен обилием встроенных функций! Благодаря им многие алгоритмы, требующие в большинстве языков написания программного кода размером в несколько экранов, реализуются на PHP одной командой (точнее, вызовом одной функции).

Современные тенденции развития языков программирования не обошли стороной и PHP. Средства объектно-ориентированного программирования появились еще в PHP3. А в объектной модели PHP4 в полном объеме реализованы классические понятия объектно-ориентированного программирования: наследование, инкапсуляция и полиморфизм.

PHP (Hypertext PreProcessor, препроцессор гипертекста) – язык программирования, исполняемый на стороне веб-сервера, спроектированный Расмусом Лердорфом (Rasmus Lerdorf) в качестве инструмента создания динамических и интерактивных веб-сайтов.

Этот язык оказался достаточно гибким и мощным, поэтому приобрёл большую популярность и используется в проектах любого масштаба: от простого блога до крупнейших веб-приложений в Интернете:

Со временем к работе над ним подключились разработчики со всего мира. PHP — один из старейших языков в рамках open source-проекта. Сейчас его поддерживает и разрабатывает группа энтузиастов во главе с компанией Zend Technologies. Ей руководят Зеев Сураски и Энди Гутманс: в 1997 году они создали третью версию PHP и активно развивают язык по сегодняшний день.

Язык программирования PHP широко применяется для построения веб-сайтов. Программы, написанные на PHP, запускают на веб-сервере, это дает возможность иметь доступ к приложению многим людям. И они спокойно могут работать на своих рабочих местах. Данный язык состоит из серверных скриптов. PHP является процессором HTML. Файлы с расширением PHP в сервере обрабатывается процессором. Такая проверка обеспечивает правильное выполнения файлов и действий приложения. Результаты запроса со страницы php выводятся в виде HTML-страницы. Скрипт написанную на php вставляют в готовую HTML-страницу. Чтобы сервер понимал, что ему просмотреть используется тег. <? PHP и?>.

PHP является языком серверных скриптов. JavaScript и Vbscript. являются языком клиентских скриптов. Это означает, что PHP-скрипт выполняется на сервере, а пользователь может видит только результат работы. Основные характеристики языка программирования PHP:

PHP язык скриптов, которая была разработана для веб-программистов, и используется для создания динамических страниц.

Когда PHP анализирует файл, тогда едет чтения содержания до тех пор, пока не встретится один из специальных операторов (тег открытия <? PHP), который будет интерпретировать текст как код PHP. Затем идет выполнение кода до тех пор, пока не встретится специальный оператор (тег закрытия?>) После чего продолжается чтение кода. Таким образом PHP можно ввести в любой момент в любой код HTML.

Обязательным правилом создания кода PHP является, строка должна обязательно закрываться «,» название функции не чувствительны к регистру (неважно СИЖ использованием больших или маленьких букв) имена

переменных чувствительны к регистру (разница между использованием больших и маленьких букв) скобки используются двойные «» или одинарные » (использование таких вариантов «» или «» будет ошибкой) код PHP ограничен одним набором тега открытия и закрытия

<? PHP текст закрытия? »- Рекомендуемая конструкция <? script language = «php»?> текст скрипта </ script>

<? Текст скрипта? »- Применение требует настройки PHP сервера.

В PHP существует три способа с помощью которых можно вставить комментарии, которые не будут отображаться в браузере, с целью придать необходимую информацию программисту. Все что следует после символов # или // считается комментарием.

Переменные в PHP представляют собой контейнер данных, который имеет имя и которому может быть присвоено значение, которое может меняться несколько раз и может быть сохранено в базе данных.

Имя данных переменной начинается с символа $ а потом с маленькой (a — z) или с большой (A — Z) или с символа «_». Имя переменной не может начинаться с цифры, и имя переменной не содержит пробелы.

Увеличение числа на единицу называется инкремента, а уменьшение на единицу декремента. Каждый PHP скрипт представляет собой набор конструкций. Конструкцией может быть присваивания, вызов функции, цикл (повтор кода), сравнение, а также конструкция, которая ничего не делает (пустой оператор). Конструкция завершается точкой с запятой.

Кроме того, конструкции объединяются в блоки заключением их в фигурные скобки. Блок конструкций — это также конструкция. В программировании PHP-конструкции позволяют определить условие а затем запустить некоторые операции в зависимости от запроса. if. else. elseif. switch.

Одной из важных функций PHP является добычи данных из форм HTML и их обработка. Элементы, с которыми вы будете работать, когда будете делать добычи такие атрибуты action и методы post и get функция — это часть

программного кода, названа уникальным именем (с точностью до регистра букв), основное назначение функции — это решение определенного задания.

Вызов функции проходит по имени в разных местах программы, позволяет многократно выполнять фрагмент с указанным именем. Конечно, что положительный эффект такого решения, в том, что блок кода пишется только раз, а потом меняется по мере необходимости.

Класс является основным понятием объектно-ориентированного программирования. Класс — это описание методов и свойств. Класс создается и не используется в программе.

Объект — это переменная, которая имеет свойства и методы, описанные в классе, от которого он создается. Можно создать несколько обек5тив от одного класса.

Свойство — это та самая переменная только внутри объекта.

Метод — это та же функция, только внутри объекта.

Язык PHP идеально подходит для создания сайтов и веб-приложений любой сложности. Блоги, интернет-магазины, лендинги (одностраничные сайты), API и прочие веб-сервисы - всё это можно сделать на PHP.

Также PHP подходит для написания консольных скриптов. Запускать их можно как разово, так и с определённой периодичностью.

Главная характеристика PHP — интерпретируемость. В отличие от Java, которая компилируется, а затем запускается в работу, PHP создается во время обращения к нему. Человек открывает сайт, на сервер посылается запрос, и в это время компилируется код. Каждый скрипт компилируется в реальном времени, а затем выполняется.

Синтаксис PHP очень похож на синтаксис C или Perl. Люди, знакомые с программированием, очень быстро смогут начать писать программы на PHP. В этом языке нет строгой типизации данных и нет необходимости в действиях по выделению/освобождению памяти.

Программы, написанные на PHP, достаточно легко читаемы. Написанный PHP – код легко зрительно прочитать и понять. В таблице 4 описание сравнение языков программирования Php и Python.

Таблица 4.4 – Сравнение языков программирования

Сравнение языков программирования PHP и Python		
Фреймворки	Python имеет меньшее количество фреймворков	Пользователи PHP имеют доступ к популярным фреймворкам
Популярность	Широко используется в искусственном интеллекте, науке о данных и научном сообществе	Язык, выбираемый для веб-разработки
Подключение к БД	Не поддерживает подключение к базе данных так широко, как PHP	Можно получить доступ к более чем 20 различным базам данных
Поддержка сообщества	В настоящее время растет быстрыми темпами	Широкая поддержка сообщества
Кривая обучения	Python лучше в долгосрочных проектах	У PHP очень низкая кривая обучения, и с ним легко начать.
Читабельность	Python использует очень строгие правила отступов. Это делает его более читаемым, чем PHP	Язык PHP хорошо документирован и придерживается классического подхода.
Тип языка	Язык программирования общего назначения	Специализируется, как язык программирования для веб-разработки
Синтаксис	Очень понятный и лаконичный синтаксис кодов	Встроенная библиотека имеет широкий спектр соглашений об именах
Известная компания, использующая эту технологию	Uber, Pinterest, Mozilla	Hootsuite, Viber, Appcelerator
Ключевая особенность	Быстрая разработка, динамическая типизация и красивый код.	Открытый исходный код, простота развертывания, постоянные улучшения.
Зарплата	Средняя зарплата профессионального разработчика Python в США составляет 120 024 доллара в год.	Средняя заработная плата разработчика PHP в США составляет 86 017 долларов в год.

Для удобного работа над кодом приложения требуется среда разработки. Одной из наиболее популярных сред для PHP является PhpStorm. Такж хорошо справляется с этой задачей и бесплатная NetBeans, о которой знаем уже так немало. Противостояние этих двух IDE сложилось исторически: сегодня вы

узнаете, почему все произошло именно так. Покажем основные сильные стороны каждой из сред разработки и увидим, какая из них больше подходит для PHP-разработчика: PhpStorm или NetBeans.

NetBeans — это одна из наиболее распространенных сред разработки, которая имеет открытый исходный код и доступна для бесплатного скачивания. В ней собраны все основные языки программирования, и PHP не является исключением. Универсальность — это ее основной competitive edge. В наши дни ониактивно конкурируют со средами для написания программ на Java, C/C++, Python и многих других, как и PHP. При установке одним из стандартных пакетов будут инструменты для работы с препроцессором.

Над NetBeans работает независимое сообщество, которое имеет доступ ко всем логам ошибок от пользователей. Они способны обрабатывать терабайты информации ежедневно. И это одно из первых преимуществ, которые успеем рассмотреть. Конечно, нет никакой гарантии, что это сделает продукт лучше, но позиция занята более выгодно, как показывают последние обновления.

Теперь перейдем к PhpStorm. Это один из продуктов от компании JetBrains. Именно они конкурируют с NetBeans в плане Java-разработки. Вместе со своей коммерческой средой IDEA они показали, что за качество стоит платить.

Такой же подход они решили применить и в сфере PHP-разработки. На базе продукта, который уже успел стать успешным, они разработали и IDE для взаимодействия с шаблонизатором. Взяв за основу IntelliJ IDEA, они переработали ее для PHP и выпустили в продажу. Результат удовлетворил ожидания. PHPStorm стал одной из наиболее востребованных сред, которые использовались в промышленных масштабах.

Свою роль сыграла и политика относительно распространения в учебных заведениях: JetBrains поставляют среду в IT-школы абсолютно бесплатно. Таким образом, молодые специалисты привыкают к интерфейсу и уже не хотят возвращаться к бесплатным решениям. Такая идея была позаимствована у Microsoft. Так они распространяют свою среду Visual Studio. Но разница в том,

что годовая подписка на VS стоит значительно дороже, и ее выбирают только высокооплачиваемые профессионалы.

Среди других недостатков, по сравнению с NetBeans, пользователи называют недостаточное количество плагинов, которые совместимы со Storm. Но здесь все понятно: их не может создать сообщество, есть лишь официальные релизы. В то же время качество иногда уступает open-source решениям. Еще программисты жалуются на недостаток интуитивности при реализации горячих клавиш. Но чему здесь удивляться: Vim вообще неинтуитивен, и ему все равно нет равных среди текстовых редакторов.

Таким образом, для создания проекта выбран язык программирования PHP и среда разработки PHPStorm.

4.5 Разработка интерфейса модуля

4.5.1 Создание базы данных проекта

После моделирования базы данных был создан SQL-скрипт для реализации базы данных на сервере MySQL:

```
CREATE TABLE IF NOT EXISTS `db_transport`.`breakpoint` (
  `id_b` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50) NULL DEFAULT NULL,
  `latitude` VARCHAR(255) NULL DEFAULT NULL,
  `longitude` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id_b`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
CREATE TABLE IF NOT EXISTS `db_transport`.`transport` (
  `id_trans` INT(11) NOT NULL AUTO_INCREMENT,
  `type` VARCHAR(255) NULL DEFAULT NULL,
  PRIMARY KEY (`id_trans`))
```

```
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
CREATE TABLE IF NOT EXISTS `db_transport`.`route` (
  `id_r` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50) NULL DEFAULT NULL,
  `operational_speed` VARCHAR(255) NULL DEFAULT NULL,
  `type_trans` INT(11) NOT NULL,
  `add_breakpoint` INT(11) NOT NULL,
  PRIMARY KEY (`id_r`),
  INDEX `fk_route_transport1_idx` (`type_trans` ASC) ,
  INDEX `fk_route_breakpoint1_idx` (`add_breakpoint` ASC) ,
  CONSTRAINT `fk_route_transport1`
    FOREIGN KEY (`type_trans`)
    REFERENCES `db_transport`.`transport` (`id_trans`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_route_breakpoint1`
    FOREIGN KEY (`add_breakpoint`)
    REFERENCES `db_transport`.`breakpoint` (`id_b`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
CREATE TABLE IF NOT EXISTS `db_transport`.`drivers` (
  `id_d` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50) NULL DEFAULT NULL,
  `surname` VARCHAR(255) NULL DEFAULT NULL,
  `telephone` VARCHAR(255) NULL DEFAULT NULL,
  `sex` VARCHAR(255) NULL DEFAULT NULL,
  `id_route` INT(11) NULL DEFAULT NULL,
```

```
  PRIMARY KEY (`id_d`),
  INDEX `FK_drivers` (`id_route` ASC) ,
  CONSTRAINT `FK_drivers`
    FOREIGN KEY (`id_route`)
    REFERENCES `db_transport`.`route` (`id_r`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
CREATE TABLE IF NOT EXISTS `db_transport`.`timetable` (
  `id_time` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50) NULL DEFAULT NULL,
  `route` INT(11) NULL DEFAULT NULL,
  `number_charts` INT(11) NULL DEFAULT NULL,
  `start` VARCHAR(255) NULL DEFAULT NULL,
  `end` VARCHAR(255) NULL DEFAULT NULL,
  `speed` VARCHAR(255) NULL DEFAULT NULL,
  `interval` VARCHAR(255) NULL DEFAULT NULL,
  `beginningWork` DATETIME NULL DEFAULT NULL,
  `endWork` DATETIME NULL DEFAULT NULL,
  PRIMARY KEY (`id_time`),
  INDEX `FK_timetable` (`route` ASC) ,
  CONSTRAINT `FK_timetable`
    FOREIGN KEY (`route`)
    REFERENCES `db_transport`.`route` (`id_r`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

Выполнение данного скрипта представлено на рисунке 4.20

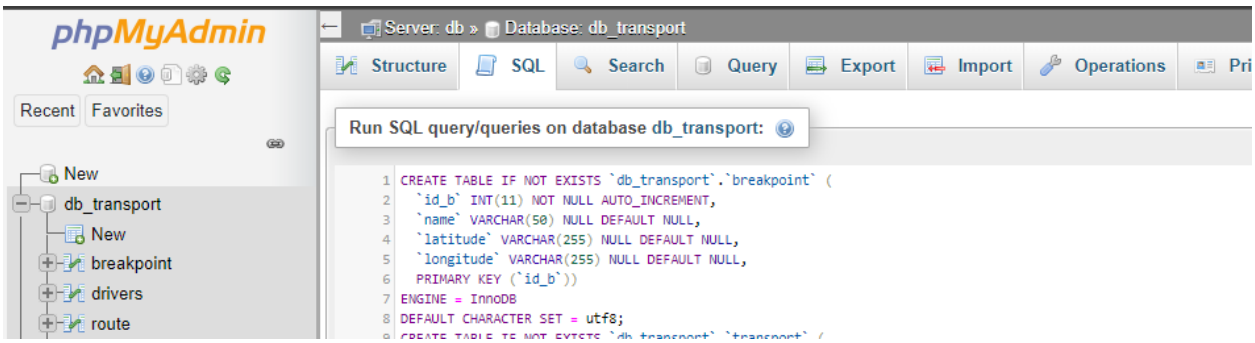

Рисунок 4.20 – Выполнение SQL-скрипта

После выполнения запроса на сервере MySQL были созданы таблицы, которые требуются для работы модуля. Показаны таблицы на рисунках 4.21 – 4.24

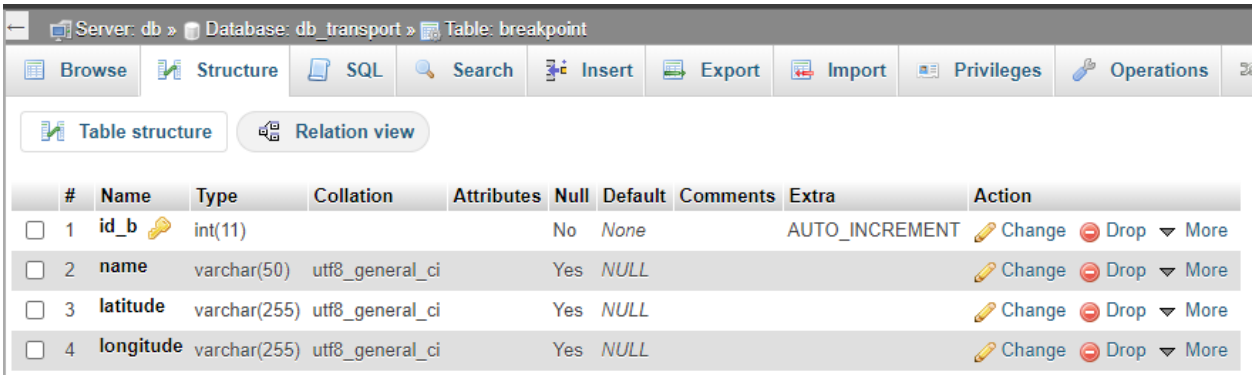

Рисунок 4.21 – Таблица Breakpoint

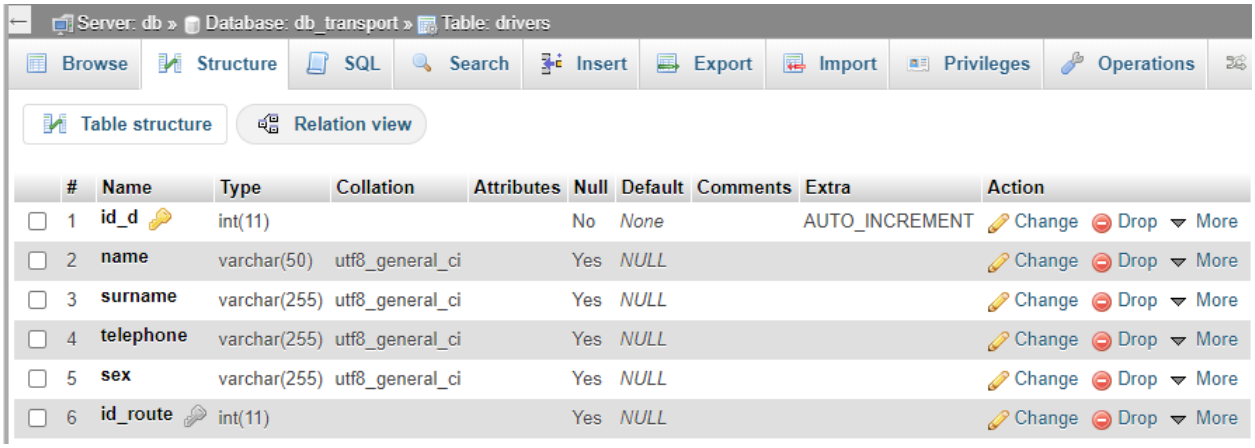

Рисунок 4.22 – Таблица drivers

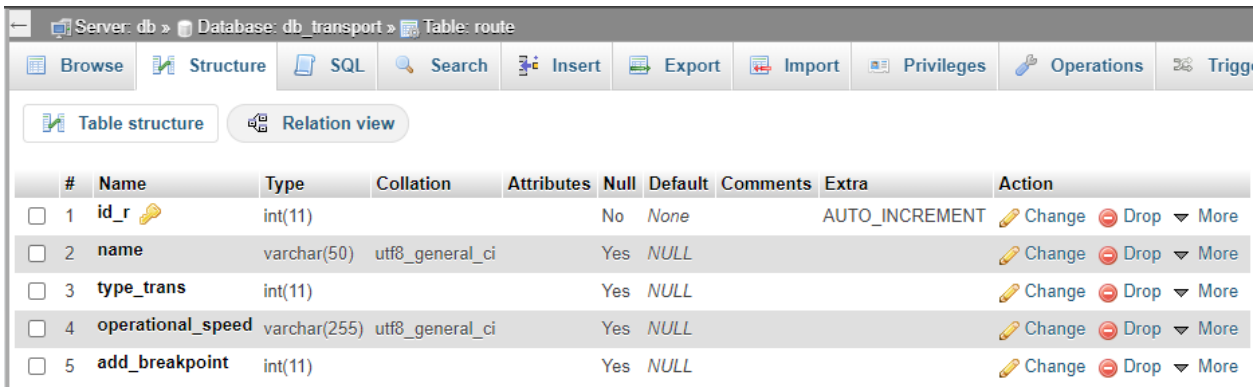

Рисунок 4.23 - Таблицаroute

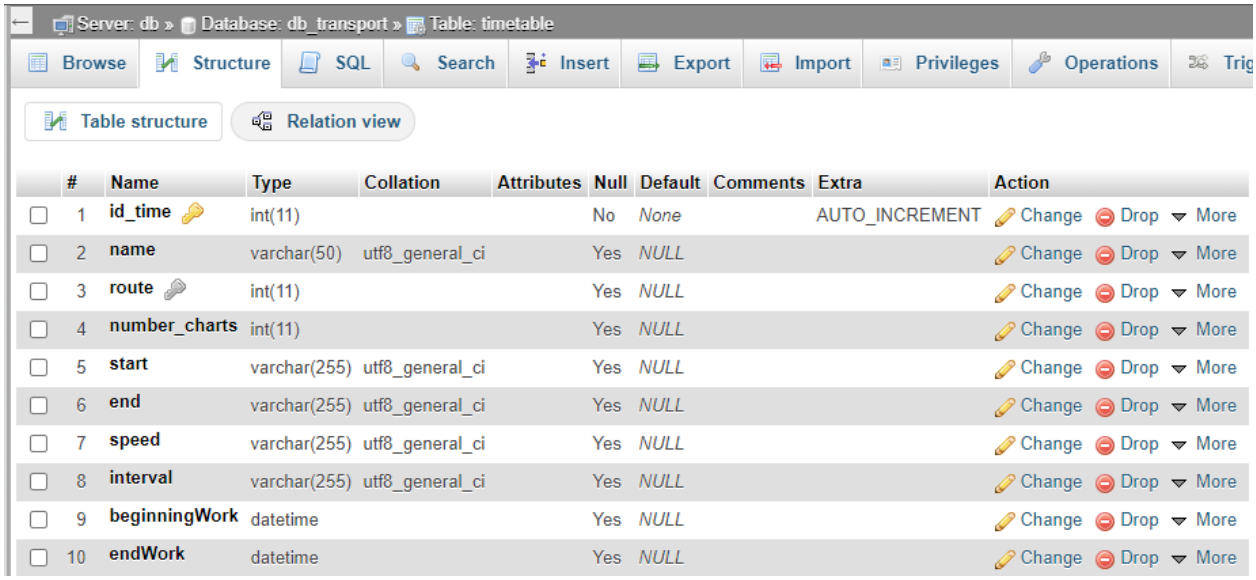

Рисунок 4.24 – Таблица timetable

Таким образом была создана база данных, которая предназначена для хранения данных модуля расписания пассажирских перевозок (рис. 25).

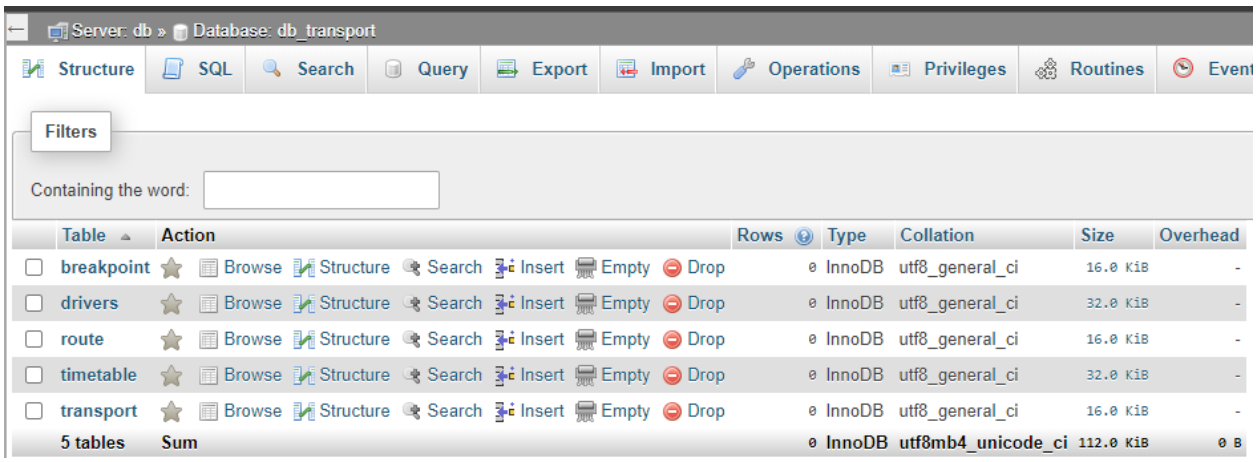

Рисунок 4.25 – База данных модуля расписания

4.5.2 Реализация модулей программного средства

После завершения процесса проектирования и моделирования базы данных и структуры приложения была осуществлена реализация модуля. Ход работ по реализации модуля показан на рисунках 4.26 – 4.31

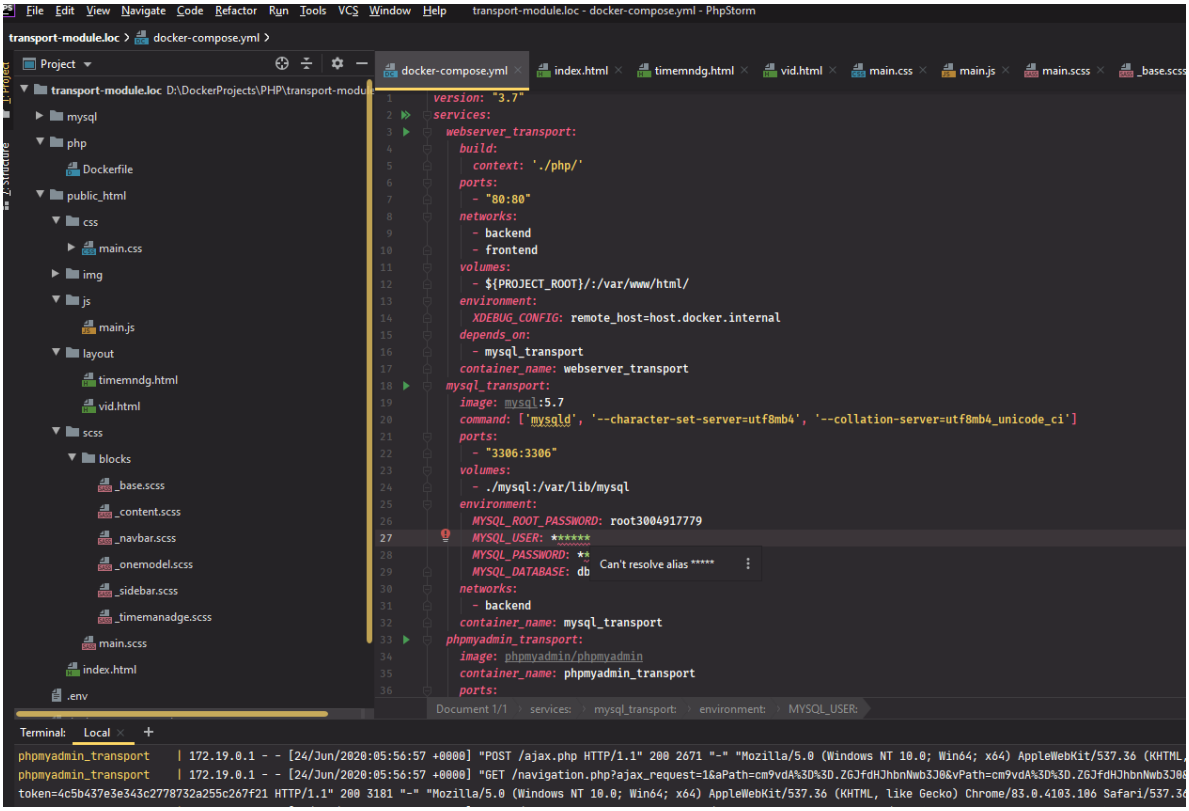

Рисунок 4.26 – Создание конфигурации сервисов для обеспечения работы модуля

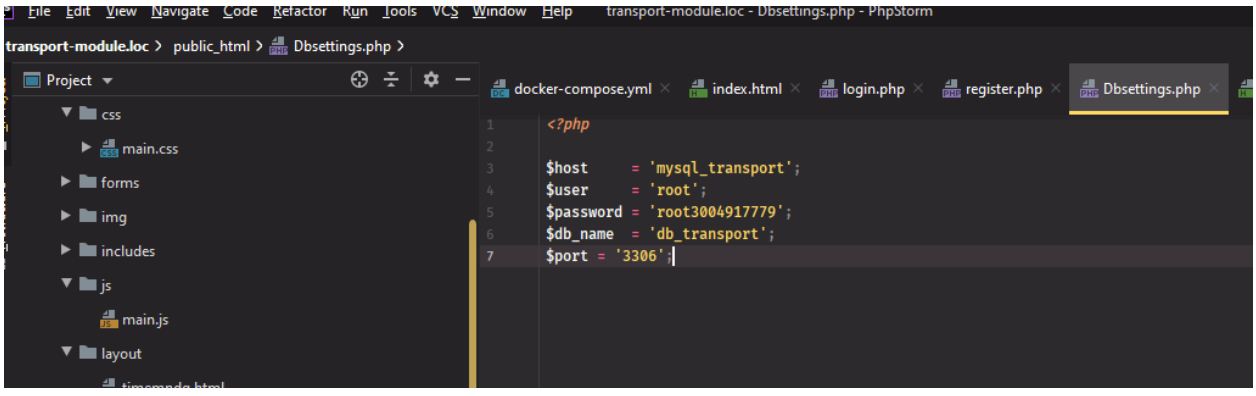

Рисунок 4.27 – Настройка соединения с базой данных

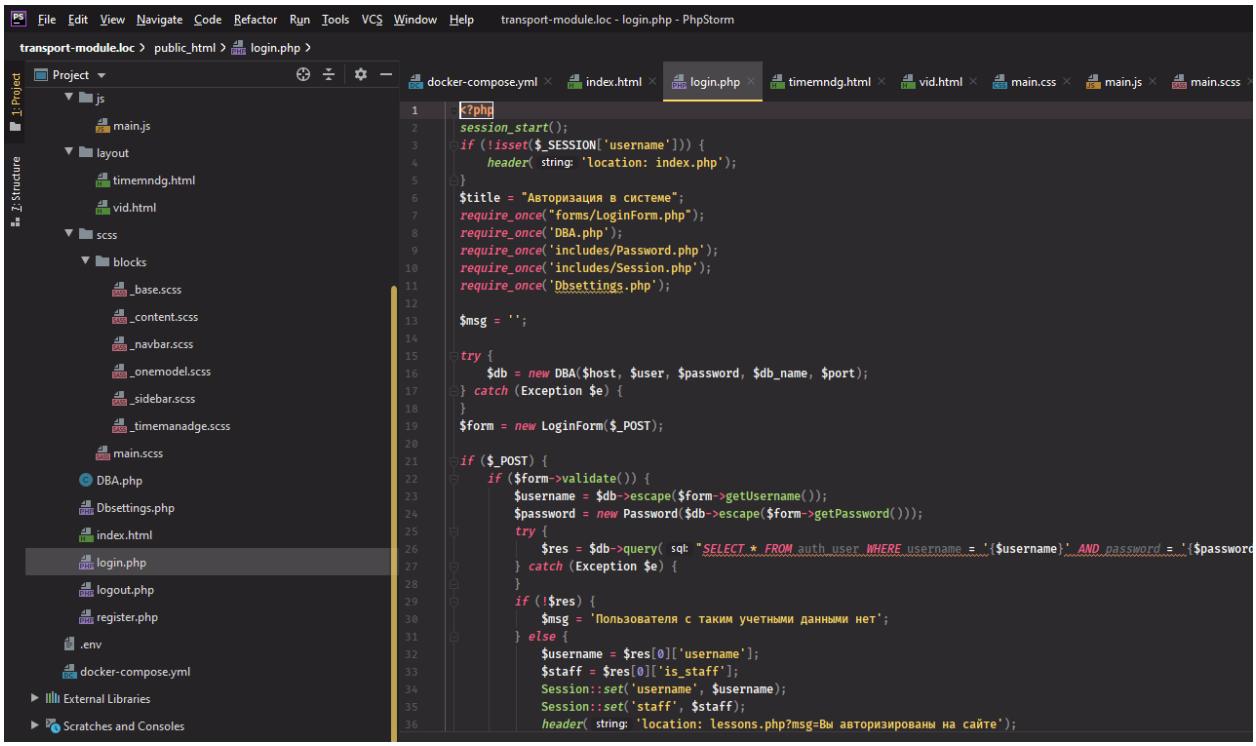

Рисунок 4.28 – Работа над компонентом авторизации в системе

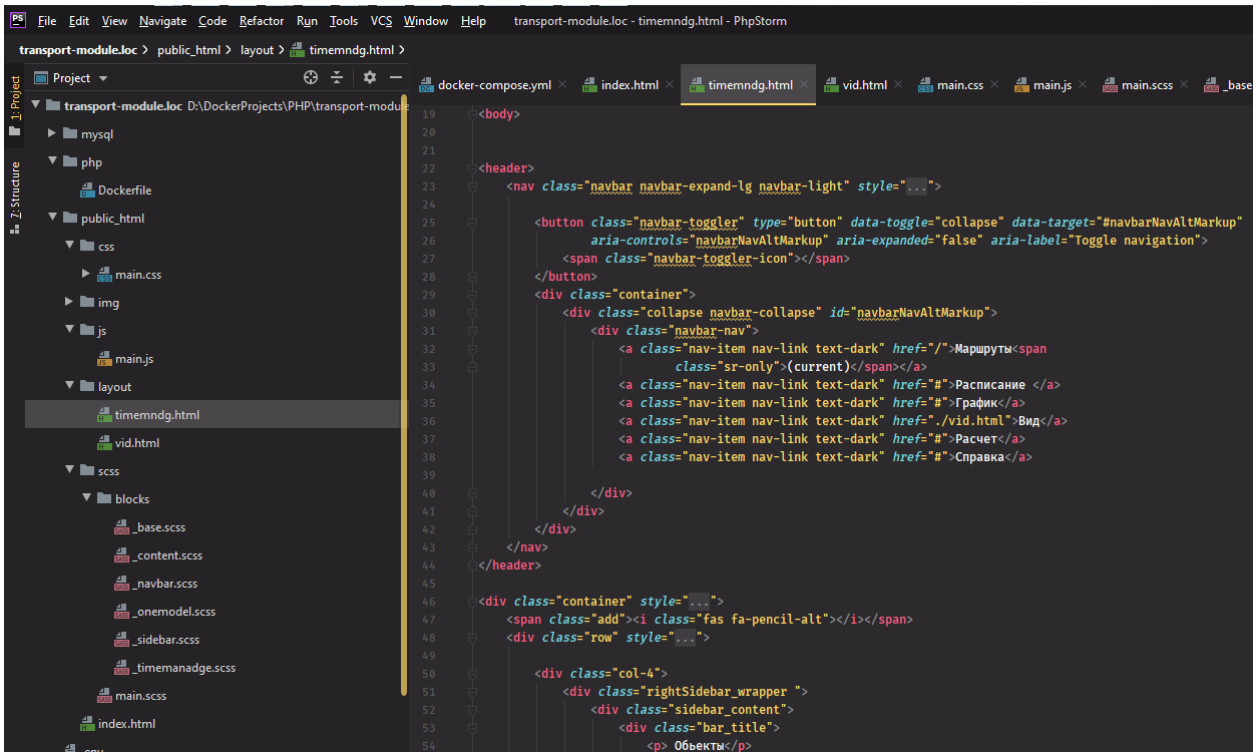

Рисунок 4.29 – Работа над страницей расписания пассажирских перевозок

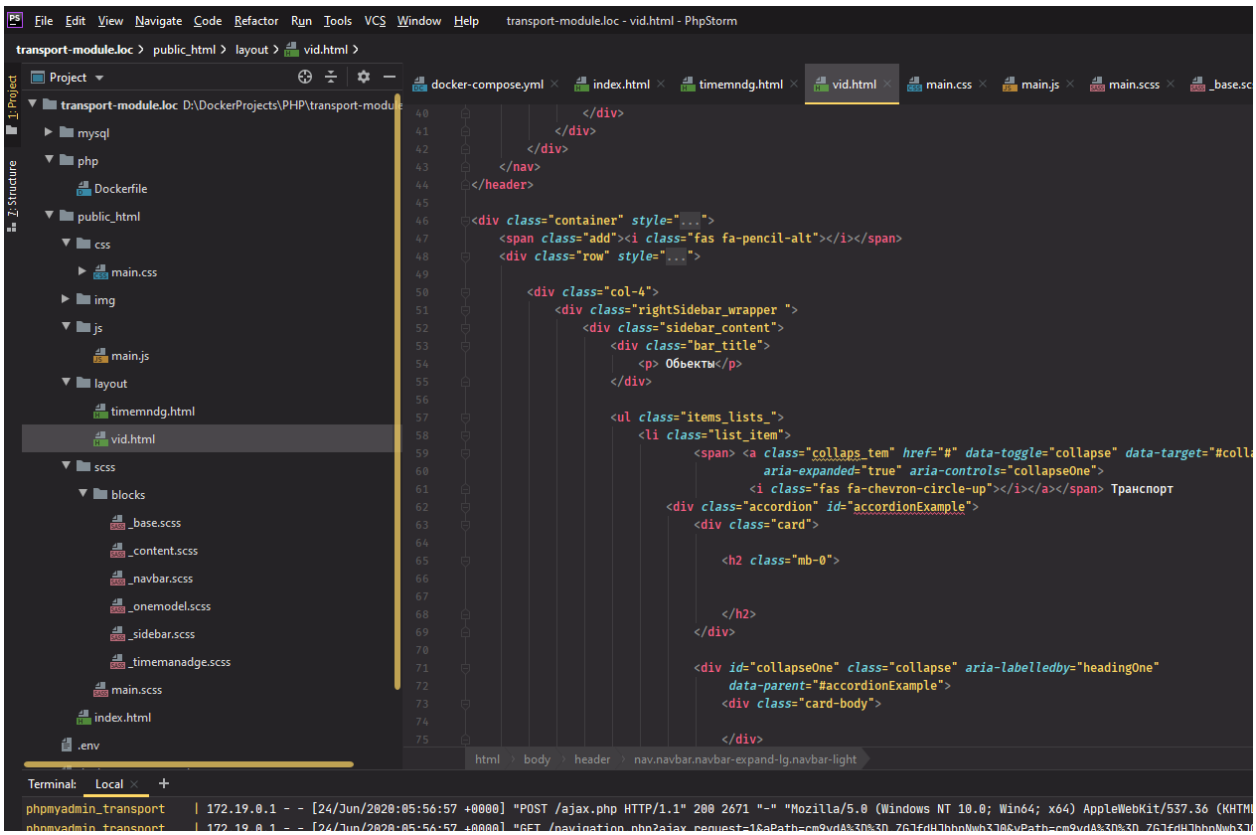

Рисунок 4.30 – Работа над страницей просмотра данных контрольной точки

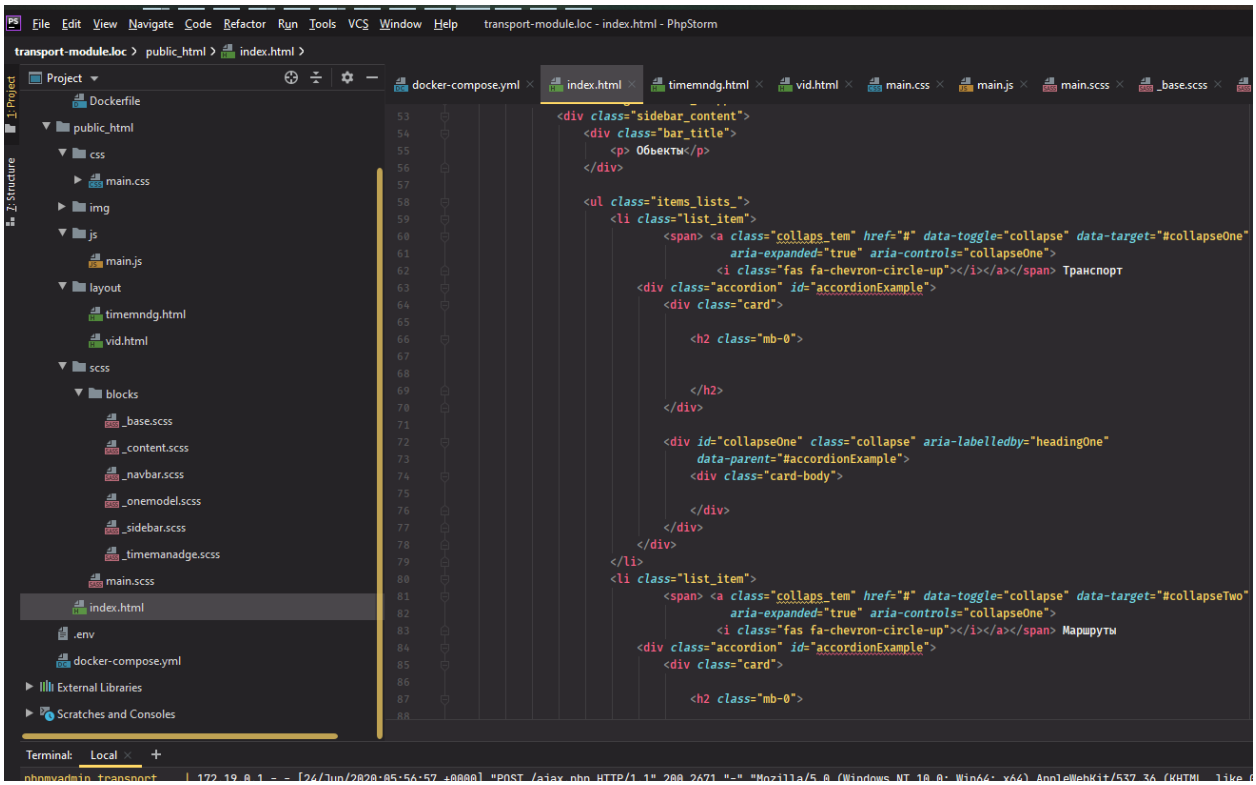

Рисунок 4.31 – Работа над главной страницей проекта

4.5.3 Тестирования разработанной системы

В проектах по разработке программного обеспечения помимо основной задачи по реализации заявленной функциональности существует не менее важная задача по обеспечению качества ПО. Качество ПО (Software quality) — это совокупность характеристик программного обеспечения, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Одним из устоявшихся способов контроля качества является тестирование. Тестирование ПО (Software testing) — проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. Далее будут описаны тесты на проверку структуры исходного кода, качества PHP-кода, валидации HTML и CSS, функциональные тесты. Ниже приведены результаты проверки работоспособности разработанного приложения. Рисунок 4.32 отображает внешний вид исходного кода приложения.

```
                <div class="navbar-nav">
                    <a id="nav_topbar" class="nav-item nav-link text-dark" href="/">Маршруты <span
                            class="sr-only">(current)</span></a>
                    <a id="nav_topbar" class="nav-item nav-link text-dark" href="./layout/timemndg.html">Расписание
                    </a>
                    <a id="nav_topbar" class="nav-item nav-link text-dark" href="#">График</a>
                    <a id="nav_topbar" class="nav-item nav-link text-dark" href="./layout/vid.html">Вид</a>
                    <a id="nav_topbar" class="nav-item nav-link text-dark" href="#">Расчет</a>
                    <a id="nav_topbar" class="nav-item nav-link text-dark" href="#">Справка</a>

                </div>
            </div>
        </div>
    </nav>
</header>

<div class="container" style="...">
    <span class="add"><i class="fas fa-pencil-alt"></i></span>
    <div class="row" style="...">

        <div class="col-4">
            <div class="rightSidebar_wrapper ">
                <div class="sidebar_content">
                    <div class="bar_title">
                        <p> Объекты</p>
                    </div>

                    <ul class="items_lists_">
                        <li class="list_item">
                                <span> <a class="collaps_tem" href="#" data-toggle="collapse" data-target="#collapseOne"
                                        aria-expanded="true" aria-controls="collapseOne">
                                    <i class="fas fa-chevron-circle-up"></i></a></span> Транспорт
                            <div class="accordion" id="accordionExample">
                                <div class="card">
```

Рисунок 4.32 - Оформление исходного кода приложения

Как видно из рисунка, исходный код приложения хорошо структурирован, содержит строки комментариев и ссылки на используемые программные библиотеки, для отслеживания изменений используется система контроля версий git.

После проверки параметров исходного кода были проверен функционал, который реализован в исходном коде. В процессе тестирования программного были проведены следующие виды тестов: тест сборки; функциональные тесты.

Тест сборки является копией «дымового» тестирования, которое выполняется для подтверждения того, что после старта приложения, оно выполняет все свои основные функции. Данное тестирование выполняется программистами, после создания программы, и если программа не проходит эти тесты, то в дальнейшем тестировании нет смысла, так как эти тесты указывают на то, что программе не работает.

Примером того, что программе не прошла «smoke testing» является то, что она не запустилась, также этот тест не проходится по причине ошибки подключения к базе данных. В результате «smoke testing» разработанная программа запустилась, и соединение с базой данных установлено успешно. Прохождение этого теста указывает на возможность проведения функционального и конфигурационного тестирования.

Функциональное тестирование представляет собой проверку того, выполняет ли программа те функции, которые предусмотрены техническим задание, а также степень точности выполнения проверяемых функций.

Таблица 1.5 - Тест главной страницы

Название:	Тест главной страницы	
Функция:	Отображение меню веб-приложения	
Действие	**Ожидаемый результат**	**Результат теста:**
Предусловие:		
Откройте программу	Открытое главное окно программы	пройден
Шаги теста:		

Выберите пункты из предоставленного меню	Страница для открытия выбрана	пройден
Нажмите кнопку перехода на другую страницу приложения	Переход на выбранную страницу осуществлен	пройден

Таблица 4.6 - Тест управления маршрутами

Название:	Тест регистрации	
Функция:	Регистрация в системе	
Действие	**Ожидаемый результат**	**Результат теста:**
Предусловие:		
Откройте программу	Открытое главное окно программы	пройден
Шаги теста:		
Выбрать пункт маршруты	Страница управления маршрутами открыт	пройден

Таблица 4.7 - Тест управление расписанием

Название:	Тест авторизации	
Функция:	Авторизация в системе	
Действие	**Ожидаемый результат**	**Результат теста:**
Предусловие:		
Откройте программу	Открытое главное окно программы	пройден
Шаги теста:		
Выбрать пункт расписание	Страница управление расписанием открыт	пройден

Таблица 4.8 - Тест водители

Название:	Тест авторизации	
Функция:	Авторизация в системе	
Действие	**Ожидаемый результат**	**Результат теста:**
Предусловие:		
Откройте программу	Открытое главное окно программы	пройден
Шаги теста:		
Выбрать пункт водители	Откыта информация о водителях	пройден

Действие	Ожидаемый результат	Результат теста:

Таблица 4.9 – Контрольные точки

Название:	Тест добавления новости	
Функция:	Добавление новости в административной панели	
Действие	**Ожидаемый результат**	**Результат теста:**
Предусловие:		
Откройте программу	Открытое главное окно программы	пройден
Шаги теста:		
Выбрать пункт Контрольные точки	Страница Контрольные точки открыта	пройден
Нажать кнопку «Добавить»	Открытие формы добавления новости	пройден
Ввести данные	Нужные поля введены	пройдет
Нажать кнопку «Сохранить»	Сохранение новых точек	пройден

При конфигурационном тестировании определено, что созданная программа успешно функционирует в разных браузерах (табл. 10).

Таблица 4.10 - Работа приложения в разных версиях ОС семейства Windows

Параметр	Google Chrome	Mozilla Firefox	Opera
Открытие главного модуля программы	+	+	+
Отображение меню программы	+	+	+
Выбор модуля программы	+	+	+
Отображение списка элементов из выбранного модуля	+	+	+
Добавление записей в выбранном модуле	+	+	+
Редактирование записей в выбранном модуле	+	+	+
Удаление записей в выбранном модуле	+	+	+

При проверке удобства пользования, определено, что программа понятна и удобна пользователя, при этом отвечает следующим параметрам:

- быстрое открытие модулей программы;
- подходящее стилевой оформление;
- удобное и понятное расположение пунктов навигации по системе;
- однотипные шрифты во всех модулях системы;
- отсутствие отвлекающих блоков и информации;
- удобное управление записями в модулях системы.

На рисунках 4.33 – 4.36 показан внешний вид окон созданного модуля.

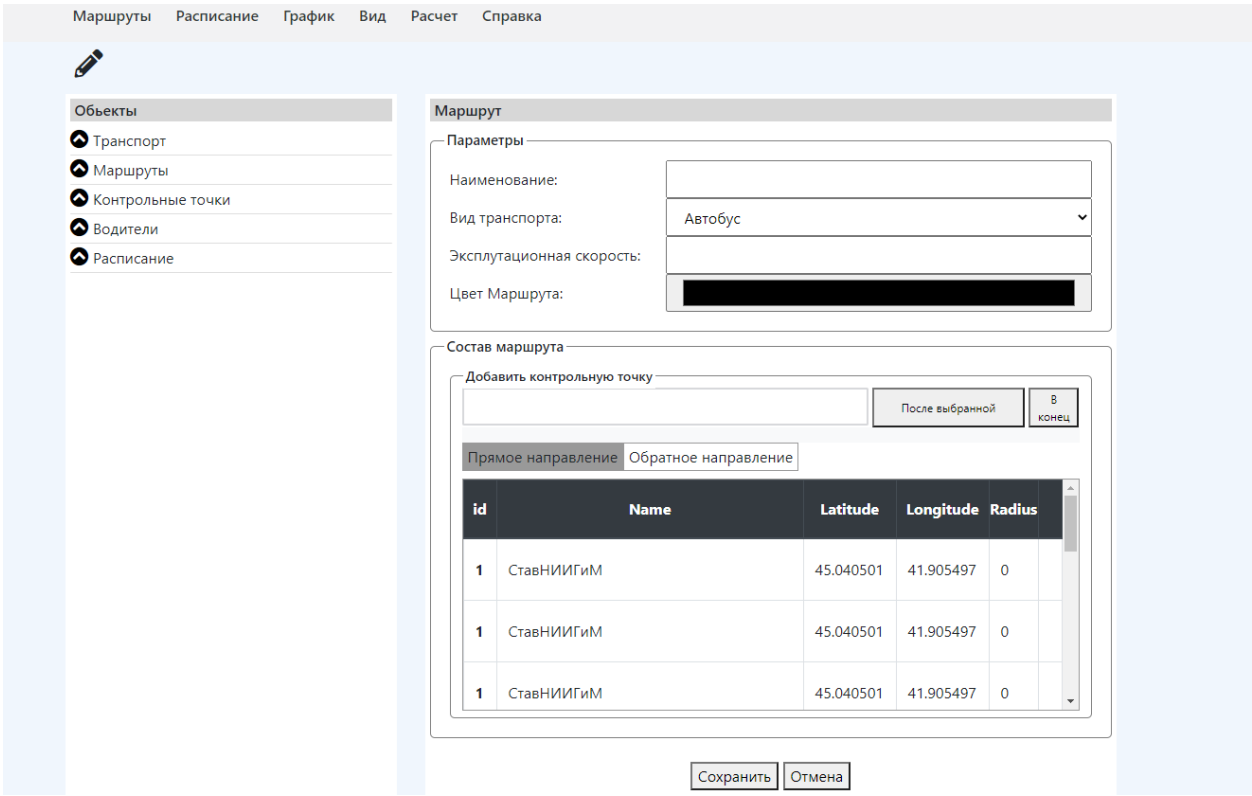

Рисунок 4.33 – Главаня старница. Прямое направление

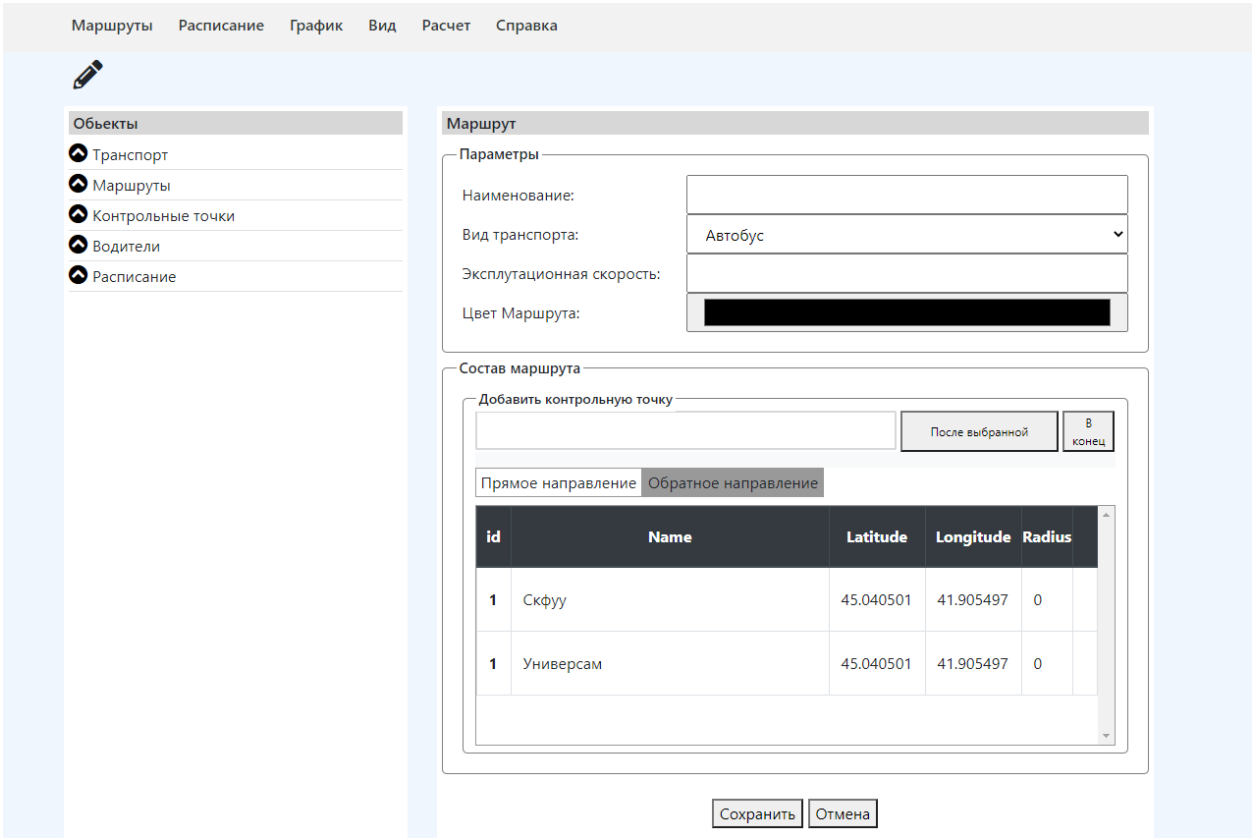

Рисунок 4.34 – Главаня страница. Обратное направление

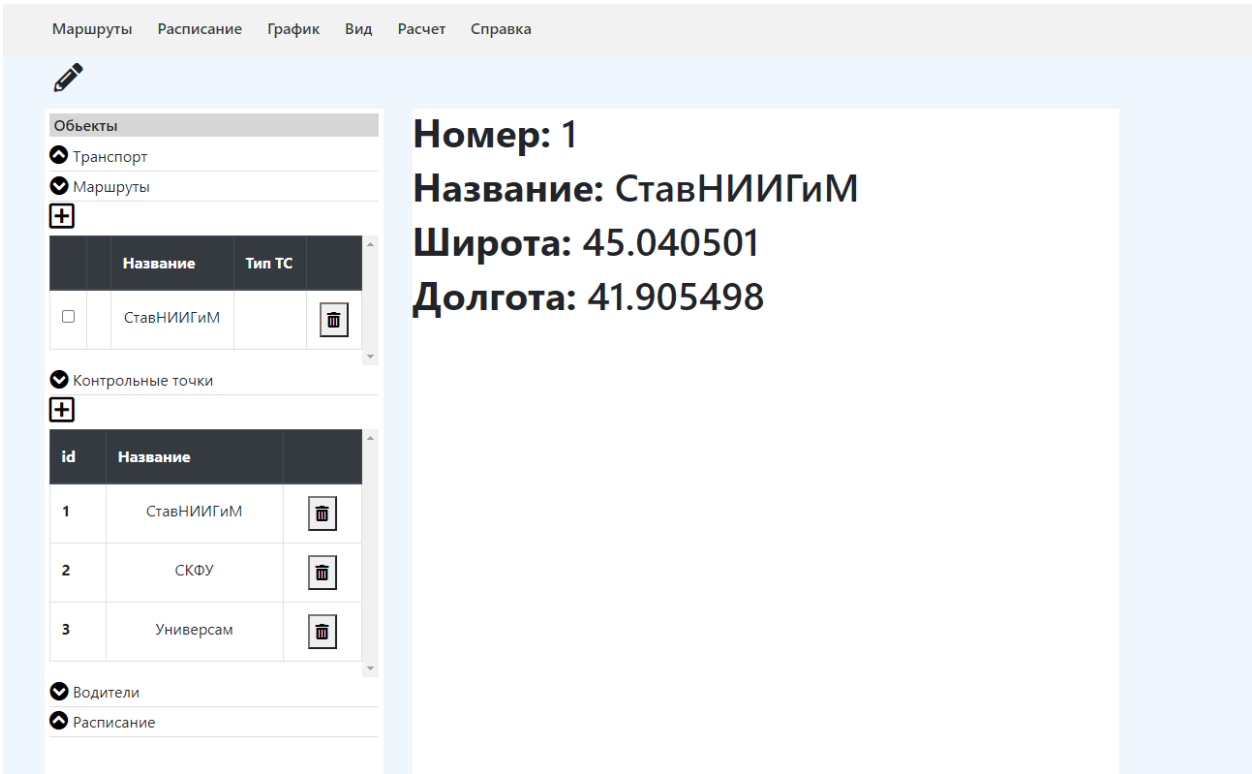

Рисунок 4.35 – Вид контрольной точки

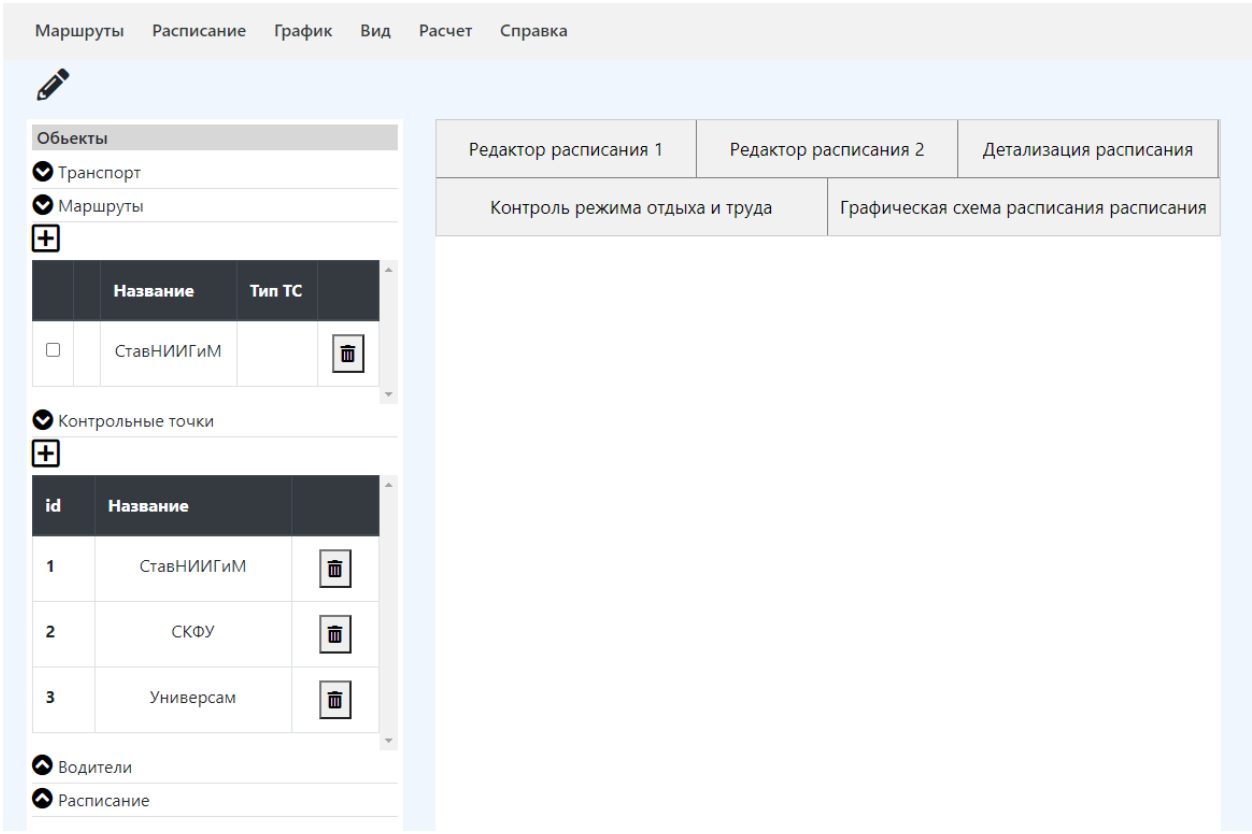

Рисунок 4.36 – Управление расписанием

Заключение к главе 4

В результате выполнения работы достигнута поставленная цель: разработан модуль расписания общественного транспорта. При достижении цели решены следующие задачи:

- проведен анализ алгоритмов составления расписания пассажирских перевозок;
- разработан проект модуля расписания общественного транспорта;
- реализован прототип программного средства прогнозирования расписания общественного транспорта.

Основными технологиями создания модуля выбраны: язык гипертекстовой разметки HTML, который служит для наглядного и хорошо структурированного представления информации; SQL - универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных; Docker – как инструмент организации окружения для реализации проекта; phpMyAdmin - веб-приложение с открытым кодом, написанное на языке PHP, представляющее собой веб-интерфейс для администрирования СУБД MySQL.

Список использованных источников

1. ГОСТ Р 54619-2011. Протоколы обмена данными автомобильной системы вызова экстренных оперативных служб с инфраструктурой системы экстренного реагирования при авариях [Текст]. – Москва: Стандартинформ, 2013. -67с.

2. Об утверждении требований к средствам навигации, функционирующим с использованием навигационных сигналов системы ГЛОНАСС или ГЛОНАСС/GPS и предназначенным для обязательного оснащения транспортных средств категории M, используемых для коммерческих перевозок пассажиров, и категории N, используемых для перевозки опасных грузов [Текст]: Приказ Минтранса РФ от 31 июля 2012 г. № 285. -2012. -82с.

3. Протокол информационного обмена оборудования ООО «Навтелеком» [Электронный ресурс] Режим доступа: https://navtelecom.ru/images/documentation/protocol/protocol_of_information_exchange_navtelecom_v5.5_%28160817%29.pdf, свободный

4. Advantages and Disadvantages of C++ | Make your Next Move! // Data Flair [Электронный ресурс] Режим доступа: https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/, свободный

5. Шлее Макс Qt 5.10. Профессиональное программирование на C++ / Макс Шлее. – СПб.: БХВ-Петербург, 2018. -1072 с.

6. Qt (software) // Wikipedia [Электронный ресурс] Режим доступа: https://en.wikipedia.org/wiki/Qt_(software), свободный

7. RabbitMQ // Wikipedia [Электронный ресурс] Режим доступа: https://en.wikipedia.org/wiki/RabbitMQ/, свободный

8. Моргунов, Е. П. PostgreSQL. Основы языка SQL: учебное пособие / Е. П. Моргунов. - СПб.: БХВ-Петербург, 2018. -336 с.

9. Новиков, Б. А. Основы технологий баз данных: учебное пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева. - М.: ДМК Пресс, 2020. -582 с.

10. PostGIS 3.0.2dev Manual [Электронный ресурс] Режим доступа: https://postgis.net/stuff/postgis-3.0.pdf, свободный

11. Леоненков, А. В. Самоучитель UML / А. В. Леоненков. – СПб.: БХВ-Петербург, 2007. -417 с.

12. Исаев Г.Н. Проектирование информационных систем: Учебное пособие / Г.Н. Исаев. - М.: Омега-Л, 2013. -424 с.

13. Мюллер, Р.Дж. Базы данных и UML. Проектирование / Р.Дж. Мюллер. - М.: ЛОРИ, 2017. -420 с.

14. Зандастра М. PHP. Объекты, шаблоны и методики программирования // И.: Вильямс, 2011. - 560 с.

15. Кузнецов М., Симдянов И. Самоучитель PHP 7 // И.: БХВ-Петербург, 2018. – 450 с.

16. [Интернет-ссылка] https://www.kennethlange.com/books/The-Little-Book-on-REST-Services.pdf // REST API

17. [Интернет-ссылка] https://ozvid.com/blog/46/reasons-why-laravel-framework-is-more-preferred-over-other-php-frameworks // Исследование фреймворков.

18. [Интернет-ссылка] What are the best PHP frameworks for building a RESTfull API // Slant URL: https://www.slant.co/topics/6956/~php-frameworks-for-building-a-restful-api.

19. [Интернет-ссылка] Fat-Free Framework // URL: []

20. [Интернет-ссылка] Панченко И. PostgreSQL: вчера, сегодня, завтра // URL: https://www.osp.ru/os/2015/03/13046900/

21. Г.-Ю. Шениг PostgreSQL 11. Мастерство разработки // И.: ДМК Пресс, 2019, 352 с.

22. Д. Арлоу, А. Нейштадт UML 2 и Унифицированный процесс: практический объектно-ориентированный анализ и проектирование, 2-е издание// И.: БХВ-Петербург, 2014, 257с.

23. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем Интернет-университет информационных технологий - 2-е изд. – М.: Бином. Лаборатория знаний Интуит Серия: Основы информационных технологий, 2008. – 300 с.

24. ГИС «Управление транспортом» [Электронный ресурс]. Режим доступа: https://transport.stavregion.ru

25. ГИС «Московский транспорт» [Электронный ресурс]. Режим доступа: http://transport.mos.ru/

26. ГИС «Московский транспорт» [Электронный ресурс]. Режим доступа: http://transport.mos.ru/

27. ГИС «Портал общественного транспорта Санкт-Петербурга» [Электронный ресурс]. Режим доступа: http://transport.orgp.spb.ru/Portal/transport/main

28. ГИС «Официальный портал транспортного информирования граждан» [Электронный ресурс]. Режим доступа: http://transport.volganet.ru/main.php

29. Develop – OpenStreetMap Wiki [Электронный ресурс]. Режим доступа: https://wiki.openstreetmap.org/wiki/Develop

30. Leaflet - a JavaScript library for interactive maps [Электронный ресурс]. Режим доступа: https://leafletjs.com/

31. Реактивное программирование — Википедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Реактивное_программирование

32. Обзор Vue.js – База знаний Timeweb Community [Электронный ресурс]. Режим доступа: https://timeweb.com/ru/community/articles/obzor-vue-js-1

33. Введение — Vue.js [Электронный ресурс]. Режим доступа: https://ru.vuejs.org/v2/guide/

34. Хаммер М., Чампи Дж. Х 18 Реинжиниринг корпорации: Манифест революции в бизнесе. Пер. с англ. — СПб.: Издательство С.-Петербургского университета, 1997. — 332 с.

35. Бизнес-логика [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Бизнес-логика

36. Гаврилов А. В. Анализ функциональных возможностей бесплатных CASE-средств проектирования баз данных // Открытое образование. 2016. — С. 39-43.

37. Гридина Е. Г. Использование поисковых систем для увеличения посещаемости федеральной системы информационно-образовательных ресурсов / Е. Г. Гридина, Д. С. Лысенко // Открытое образование. 2009. №3. С. 43–48.

38. Душкина М. Р. PR и продвижение в маркетинге. Коммуникациии воздействие, технологии и психология. - СПБ: Питер, 2010. - 560 с.

39. Зашеловский А. Е. Среды разработки программного обеспечения, ориентированные на процессы [Электронный ресурс] / А. Е. Зашеловский, О. Ф. Абрамова. – Режим доступа: http://www.scienceforum.ru/2015/pdf/15128.pdf (дата обращения 18.11.2010).

40. Играем мускулами: методы и средства взлома баз данных MySQL [Электронный ресурс]. — URL: https://xakep.ru/static/best-hacks/MYSQL.pdf (дата обращения: 22.06.2020).

41. Карпова И. П. Базы данных. Учебное пособие. – Московский государственный институт электроники и математики (Технический университет). – М., 2009. — 131 с.

42. Косенко В. В. Популярные языки программирования с позиций системного программирования / В. В. Косенко // Информационные технологии и вычислительные системы. – 2013. - № 1. – С. 54-59

43. Коцюба И. Ю. Основы проектирования информационных систем. Учебное пособие / И. Ю. Коцюба, А. В. Чунаев, А. Н. Шиков. – СПб: Университет ИТМО, 2015. – 206 с.

44. Либерти Д. Программирование на C#. Создание .NET приложений. Программирование на C# / Д. Либерти. – М.: Бином 2010. – 684 с.

45. Малыхина, М.П. Базы данных: основы, проектирование, использование: учебное пособие / М. П. Малыхина, Санкт-Петербург: БХВ-Петербург, 2004, 512 с.

46. Неретина Е. А. Web-сайт ВУЗа как важный инструмент маркетинговых коммуникаций // Вестник ЮУрГУ. Серия: Экономика и менеджмент. 2009. №41 (174). [Электронный ресурс] URL: http://cyberleninka.ru/article/n/web-sayt-vuza-

kak-vazhnyy-instrument-marketingovyh-kommunikatsiy (Дата обращения: 2. 05.2020).

47. Пинягина О. В. Практикум по курсу "Базы данных": [учебное пособие] / О. В. Пинягина, И. А. Фукин; Казан. (Приволж.)федер. ун-т, Казань: Казанский университет, 2012, 91 с.

48. Понимание NoSQL // Spring [Электронный ресурс]. — URL: https://spring-projects.ru/understanding/nosql/ (дата обращения: 22.06.2020).

49. Релевантность: [Электронный ресурс] // SEO-COPYWRITING. Уникальный контент. URL: http://www.seo-copywrite.ru/15/ (Дата обращения: 18. 05.2020).

50. Советов Б.Я. Базы данных: теория и практика: учебник для бакалавров: для студентов вузов, обучающихся по направлениям "Информатика и вычислительная техника" и "Информационные системы" / Б.Я.Советов, В.В.Цехановский, В.Д.Чертовской, Издание 2-е,Москва:Юрайт, 2012,463 с

51. Туманов В. Е. Проектирование хранилищ данных для систем бизнес-аналитики: учебное пособие / В.Е.Туманов, Москва: Интернет-Университет Информационных Технологий: БИНОМ. Лаборатория знаний, 2011, 615 с.

52. Что нужно знать о развитии СУБД // Cnews [Электронный ресурс]. — URL: https://www.cnews.ru/articles/2019-08-26_chto_nuzhno_znat_o_razvitii_subd (дата обращения: 22.06.2020).

53. Шипулина Ю. С. SEO-оптимизация: основные ошибки в её применении / Ю. С. Шипулина, М. С. Проноза // Экономические проблемы устойчивого развития: материалы Международной научно-практической конференций, посвященной памяти проф. Балацкого Е. Ф. (г. Суммы, 24-26 апреля 2013 г.): в 4 т. общ. ред. А. В. Прокопенко. - Сумы: Сумский государственный университет, 2013. - Т. 4. - С. 152-153.

54. Явич М. П. Сравнение процессов оптимизации сайта в поисковых системах с использованием PHP и HTML / М. П. Явич, Г. Ю. Иашвили // Современная техника и технологии. 2014. № 3 [Электронный ресурс]. URL: http://technology.snauka.ru/2014/03/3233 (дата обращения: 18.02.2016).

55. Microsoft SQL Server // Википедия [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki/Microsoft_SQL_Server (дата обращения: 22.06.2020).

56. MySQL Workbench // Википедия [Электронный ресурс]. — URL: https://ru.wikipedia.org/wiki/MySQL_Workbench (дата обращения: 22.06.2020).

ПРИЛОЖЕНИЕ А

1.1. Установка QT и Qt-creator

1. Установить, выполнив:

sudo apt-get install build-essential

sudo apt-get install qt5-default

sudo apt-get install qtcreator

sudo apt-get install g++ gcc git qtbase5-dev libqt5sql5-sqlite libqt5multimediaquick-p5 libqt5multimedia5-plugins libqt5multimedia5 libqt5qml5 qtdeclarative5-dev libqt5quick5 libqt5opengl5-dev

sudo apt-get install libqt5svg5*

1.2. Установка RabbitMQ

1. Установить Erlang:

apt install gpg-agent

gpg-agent –daemon

sudo apt-get install gnupg

wget https://packages.erlang-solutions.com/erlang-solutions_1.0_all.deb

sudo dpkg -i erlang-solutions_1.0_all.deb

wget https://packages.erlang-solutions.com/ubuntu/erlang_solutions.asc

sudo apt-key add erlang_solutions.asc

sudo apt-get update

sudo apt-get install esl-erlang

2. Установить Rabbitmq-Server:

wget -O- https://dl.bintray.com/rabbitmq/Keys/rabbitmq-release-signing-key.asc | sudo apt-key add -

wget -O- https://www.rabbitmq.com/rabbitmq-release-signing-key.asc | sudo apt-key add -

echo "deb https://dl.bintray.com/rabbitmq/debian $(lsb_release -sc) main" | sudo tee /etc/apt/sources.list.d/rabbitmq.list

sudo apt update

sudo apt -y install rabbitmq-server

sudo systemctl status rabbitmq-server.service

sudo systemctl enable rabbitmq-server

sudo rabbitmq-plugins enable rabbitmq_management

1.3. Установка RabbitMQ-C

1. Установить cmake:

sudo apt-get install cmake

2. Скачать последнюю версию по ссылке:

https://github.com/alanxz/rabbitmq-c/releases/latest

3. Открыть распакованную папку в терминале.

4. Выполнить:

mkdir build && cd build

sudo cmake -DENABLE_SSL_SUPPORT=OFF ..

sudo cmake --build . --target install

1.4. Установка SimpleAmqpClient

1. Установить boost:

sudo apt-get install libboost-all-dev

2. Скачать пакет по ссылке:

https://github.com/alanxz/SimpleAmqpClient

3. Открыть распакованную папку в терминале.

4. Выполнить:

mkdir simpleamqpclient-build

cd simpleamqpclient-build

cmake ..

make

sudo make install

1.5. Установка PostgreSQL

1. Установить postgresql:

sudo apt update

sudo apt install postgresql postgresql-contrib

sudo systemctl enable postgresql

sudo -i -u postgres

psql

\password postgres

Ввести пароль postgres

\q

exit

2. Установить phppgadmin:

sudo apt-get install phppgadmin

3. Чтобы того чтобы получить доступ к PhpPgAdmin через ваш браузер, веб – сервер Apache должен быть настроен. Поэтому откроем файл phppgadmin.conf:

sudo nano /etc/apache2/conf-available/phppgadmin.conf

Закомментируем линию ‘Require local‘, добавив ‘#’ в передней части линии. Тогда под той самой линии, добавьте ‘Allow From all‘. Сохраните и закройте файл. После редактирования секции должен выглядеть следующим образом:

Only allow connections from localhost:

#Require local

Allow From all

4. Теперь настроить некоторые параметры для PhpPgAdmin. Откройте файл config.inc.php:

sudo nano /etc/phppgadmin/config.inc.php

5. Найдите ‘$conf[‘extra_login_security’] = true;‘ и измените значение с true на false , так что вы можете войти в PhpPgAdmin как пользователь Postgres. Сохраните и закройте файл.

6. Теперь перезапустите PostgreSQL и Apache. Затем включите их запуск при загрузке. Все это можно сделать с помощью следующих команд:

sudo systemctl restart postgresql

sudo systemctl restart apache2

sudo systemctl enable apache2

7. Теперь откройте веб-браузер и войдите в PhpPgAdmin, перейдя по ссылке http://your_server_IP/phppgadmin/.

Вы можете войти, используя пользователь Postgres и пароль, настроенный ранее.

8. Установить pgadmin:

sudo apt install pgadmin3

1.6. Разворачивание базы данных

1. Создать пользователя sandro через phppgadmin с правами суперпользователя.

2. Установить postgis:

sudo apt install postgresql-10

sudo apt install postgresql-10-postgis-2.4

sudo apt install postgresql-10-postgis-scripts

pg_lsclusters

sudo apt install postgresql-10-pgrouting

3. Установить psgl драйвер для Qt:

sudo apt install libqt5sql5-psql

4. Создать базу данных monitoring.

5. Скопировать файл monitoring_before_stress_test.backup на компьютер.

6. Развернуть backup бд:

psql –v ON_ERROR_STOP=1 -h localhost -p 5432 -U postgres -f "/Путь_к_файлу_monitoring_before_stress_test.backup" "monitoring"

ПРИЛОЖЕНИЕ Б

Файл «main.cpp»

```
/*************************************************************
 * @file       main.cpp
 * @date       2014-2020
 * @version         0.1
 * @brief      Основной файл для сервера приложений системы
мониторинга пассажирского транспорта.
*************************************************************/

#include <QCoreApplication>
#include <QDir>
#include <QtDebug>
#include "modulemanager.h"

int main(int argc, char *argv[]) {
       QCoreApplication a(argc, argv);

       QDir modulesDir = QDir(a.applicationDirPath());
    modulesDir.cdUp();
    modulesDir.cd("ServerModules/libs");

       ModuleManager moduleManager;

       // загружаем модули
       if (!moduleManager.loadModules(modulesDir)) {
          qCritical("Unable to load all required modules!");
          return -1;
    }

       // инициализируем модули
    if (!moduleManager.initModules()) {
          qCritical("Unable to init modules!");
          return -1;
    }

       // запускаем модули
       moduleManager.launchModules();

       return a.exec();
}
```

Файл «modulemanager.cpp»

```
/*************************************************************
 * @file       modulemanager.cpp
 * @date       2014-2020
 * @version         0.1
 * @brief      Реализация класса, загружающего модули и
реализующего функции взаимодействия модулей.
*************************************************************/

#include <QDir>
```

```
#include <QPluginLoader>
#include <QDebug>
#include "modulemanager.h"
#include "interfaces/basemodule.h"


/**
 * @brief Загружает модули из указанной папки
 * @return
 */
bool ModuleManager::loadModules(QDir modulesDir) {
    qDebug() << "Loading modules from " <<
qPrintable(modulesDir.absolutePath());

       foreach (QString fileName,
modulesDir.entryList(QDir::Files)) {
        qDebug() << "Loading module " <<
qPrintable(modulesDir.absoluteFilePath(fileName));

          QPluginLoader
loader(modulesDir.absoluteFilePath(fileName));
          QObject *module = loader.instance();
          if (!module) {
            qCritical("Error: %s",
qPrintable(loader.errorString()));
            return false;
          }
          modules.append(module);
          moduleFiles[module] = fileName;
       }

       return true;
}


/**
 * @brief Инициализирует все загруженные модули.
 * @return
 */
bool ModuleManager::initModules() {
       // список модулей для инициализации
       ModuleList remainingModules(modules);
       ModuleList initializedModules;

       // пробуем в цикле инициализировать все загруженные модули,
удовлетворяя их зависимости
       while (true) {
          if (remainingModules.empty()) {
               return true;
          }
          // берем первый неинициализированный модуль
          QObject *module = remainingModules.at(0);
          // и инициализируем его и все его зависимости
```

```
            if (!initModuleAndAllRequirements(module,
&remainingModules, &initializedModules)) {
                qCritical("Unable to initialize module %s",
qPrintable(moduleFiles[module]));
              return false;
            }
        }
}


/**
 * @brief Запускает модули на выполнение.
 */
void ModuleManager::launchModules() {
        ModuleList::const_iterator i;
        for (i = modules.constBegin(); i != modules.constEnd(); i++)
{
            Module module = *i;
            BaseModule *baseModule = qobject_cast<BaseModule
*>(module);
            if (baseModule) {
                 baseModule->launch();
            }
            else {
                 qCritical("Unable to launch module %s",
qPrintable(moduleFiles[module]));
            }
        }
}


/**
 * @brief Инициализирует указанный модуль и все его зависимости.
Зависимости инициализируются рекурсивно!
 * @param module         Указатель на модуль для инициализации.
 * @param remaining      Список модулей, оставшихся
неинициализированными.
 * @param initialized    Список уже инициализированных модулей.
 * @return
 */
bool ModuleManager::initModuleAndAllRequirements(QObject *module,
ModuleList *remaining, ModuleList *initialized) {
        BaseModule *baseModule = qobject_cast<BaseModule *>(module);
        if (!baseModule) {
            return false;
        }

        baseModule->registerModuleStorage((ModuleStorage *)this);

        // запрашиваем требования модуля
        ModuleRequirements requirements = baseModule-
>getRequirements();
```

```
        // если требования пусты - пытаемся инициализировать модуль
        if (requirements.empty()) {
           if (baseModule->init()) {
             qDebug() << "Module " <<
qPrintable(moduleFiles[module]) << "successfully initialized!";
                remaining->removeAll(module);
                initialized->append(module);
                return true;
           }
           else {
             qCritical("Module %s failed!",
qPrintable(moduleFiles[module]));
             remaining->removeAll(module);
             initialized->append(module);
             return false;
           }
        }

        // если требования у модуля присутствуют -
        // пытаемся найти модуль с нужным интерфейсом в списке уже
инициализированных модулей
        ModuleRequirements::const_iterator i;
        // проходим по списку требований
        for (i = requirements.constBegin(); i !=
requirements.constEnd(); i++) {
           QString requiredInterface = *i;

           bool interfaceFound = false;
           ModuleList::const_iterator j;
           // проходим по списку уже инициализированных модулей
           for (j = initialized->constBegin(); j != initialized-
>constEnd(); j++) {
                QObject *initializedModule = *j;
                // если модуль поддерживает необходимый интерфейс
                if (initializedModule-
>inherits(requiredInterface.toLocal8Bit())) {
                     // говорим, что интерфейс найден и выходим из
цикла
                     interfaceFound = true;
                     break;
                }
           }

           // если интерфейс в списке инициализированных модулей не
найден
           if (!interfaceFound) {
                // проходим по списку еще не инициализированных
модулей
                for (j = remaining->constBegin(); j != remaining-
>constEnd(); j++) {
                     QObject *uninitializedObject = *j;
                     // если модуль поддерживает необходимый
интерфейс
```

```
                    if (uninitializedObject-
>inherits(requiredInterface.toLocal8Bit())) {
                        // пытаемся инициализировать этот модуль и
удовлетворить все его требования
                        if
(!initModuleAndAllRequirements(uninitializedObject, remaining,
initialized)) {
                       return false;
                        }
                        // если инициализировали модуль с нужным
интерфейсом, говорим что интерфейс найден
                        // и продолжаем перебирать модули, чтобы
инициализировать все модули с нужным интерфейсом
                        else {
                             interfaceFound = true;
                             // Обновляем итератор, поскольку
remaining изменился внутри рекурсивного вызова.
                             // Использовать здесь
QMutableListIterator не представляется возможным, потому что
                             // в один момент времени только один
экземпляр QMutableListIterator может работать
                             // со списком, а при рекурсии каждый
раз создается новый и static не прокатывает =)
                             j = remaining->constBegin();
                        }
                   }
              }
              if (!interfaceFound) {
                   qCritical("Module %s requires interface %s!",
qPrintable(moduleFiles[module]), qPrintable(requiredInterface));
                   return false;
              }
         }
      }

      // если мы дошли досюда, значит все требования
удовлетворены, можно инициализировать модуль
      if (baseModule->init()) {
       qDebug() << "Module " << qPrintable(moduleFiles[module])
<< "successfully initialized!";
         remaining->removeAll(module);
         initialized->append(module);
         return true;
      }
      else {
       qCritical("Module %s failed!",
qPrintable(moduleFiles[module]));
       remaining->removeAll(module);
       initialized->append(module);
       return false;
      }

    return true;
```

```
}


/**
 * @brief Возвращает первый модуль из списка с указанным
интерфейсом.
 * @param interface Интерфейс модуля.
 * @return              Модуль, реализующий требуемый интерфейс.
 */
Module ModuleManager::getModule(QString iface) {
       ModuleList::const_iterator i;
       for (i = modules.constBegin(); i != modules.constEnd(); i++)
{
          Module module = *i;
          if (module->inherits(iface.toLocal8Bit())) {
               return module;
          }
       }

       // если в списке нет модуля с необходимым интерфейсом,
возвращаем NULL.
       return NULL;
}


/**
 * @brief Возвращает список модулей, реализующих указанный
интерфейс.
 * @param interface Интерфейс модулей.
 * @return              Список модулей.
 */
ModuleList ModuleManager::getModules(QString iface) {
       ModuleList list;
       ModuleList::const_iterator i;
       for (i = modules.constBegin(); i != modules.constEnd(); i++)
{
          Module module = *i;
          if (module->inherits(iface.toLocal8Bit())) {
               list.append(*i);
          }
       }
       return list;
}
```

Файл «filelogger.cpp»

```
/*************************************************************
 * @file       filelogger.cpp
 * @date       2014-2020
 * @version         0.1
 * @brief      Реализация модуля вывода логов в файл.

*************************************************************/
```

```
#include <QMutexLocker>
#include <QDateTime>
#include "filelogger.h"


FileLogger::FileLogger() :
          logFileStream(&logFile), logStdoutStream(stdout) {
       // добавляем требование наличия модуля конфигурации
       requirements.append(ConfigurationIId);
}


/**
 * @brief Инициализирует модуль логирования.
 * @return     Статус инициализации.
 */
bool FileLogger::init() {
       // загружаем модуль конфигурации
       Module configurationModule = moduleStorage-
>getModule(ConfigurationIId);
       if (!configurationModule) {
        qCritical("Failed %s!", "Configuration module");
          return false;
       }
       configuration = qobject_cast<Configuration
*>(configurationModule);
       if (!configuration) {
        qCritical("Failed %s!", "Configuration");
          return false;
       }

       // получаем имя файла с логом
       QVariant logFileName;

       if (!configuration->getValue(QString("log/file_name"),
logFileName)) {
        qCritical("Failed %s!", "getLog");
          return false;
       }

       // открываем файл лога
       logFile.setFileName(logFileName.toString());
       if (!logFile.open(QIODevice::WriteOnly | QIODevice::Append))
{
        qCritical("Failed %s!", "opentLog");
          return false;
       }

       return true;
}


/**
```

```
 * @brief Запускает модуль логирования.
 */
void FileLogger::launch() {
	debug("Logger launched.");
}


/**
 * @brief		Выводит сообщение на уровне "Отладка".
 * @param message	Сообщение.
 */
void FileLogger::debug(const QString &message) {
	logMessage(QtDebugMsg, message);
}


/**
 * @brief		Выводит сообщение на уровне "Предупреждение"
 * @param message	Сообщение.
 */
void FileLogger::warning(const QString &message) {
	logMessage(QtWarningMsg, message);
}


/**
 * @brief		Выводит сообщение на уровне "Критическое".
 * @param message	Сообщение.
 */
void FileLogger::critical(const QString &message) {
	logMessage(QtCriticalMsg, message);
}


/**
 * @brief		// Выводит сообщение на уровне "Фатальное".
 * @param message	// Сообщение.
 */
void FileLogger::fatal(const QString &message) {
	logMessage(QtFatalMsg, message);
}


/**
 * @brief		Выполняет фактический вывод сообщения в файл и
на консоль.
 * @param type		Тип (уровень) сообщения.
 * @param message	Сообщение.
 */
void FileLogger::logMessage(QtMsgType type, const QString
&message) {
	QMutexLocker locker(&mutex);
	QString finalMessage;
```

```
        QString messageType;

        switch (type) {
           case QtDebugMsg:
                messageType = "debug";
           break;
           case QtWarningMsg:
                messageType = "warning";
           break;
           case QtCriticalMsg:
                messageType = "critical";
           break;
           case QtFatalMsg:
                messageType = "fatal";
           break;
         default:
             messageType = "debug";
         break;
        }

        finalMessage = QString("%1 [%2] -
%3").arg(QDateTime::currentDateTime().toString("dd.MM.yyyy
hh:mm:ss")).arg(messageType).arg(message);

        logFileStream << finalMessage << endl;
        logStdoutStream << finalMessage << endl;
}
```

Файл «iniconfiguration.cpp»

```
/*************************************************************
 * @file       iniconfiguration.cpp
 * @date       2014-2020
 * @version         0.1
 * @brief      Реализация модуля загрузки конфигурации из ini-
файла.
 *************************************************************/

#include <QCoreApplication>
#include "iniconfiguration.h"


IniConfiguration::IniConfiguration() {
        QString settingsFileName =
QCoreApplication::applicationDirPath() + "/settings.ini";
        settings = new QSettings(settingsFileName,
QSettings::IniFormat);
}


IniConfiguration::~IniConfiguration() {
        delete settings;
}
```

```
bool IniConfiguration::init() {
	return true;
}

void IniConfiguration::launch() {

}

/**
 * @brief	Возвращает значение из файла конфигурации по
указанному ключу.
 * @param key	Ключ
 * @param value	Значение
 * @return	Существование ключа в конфигурации
 */
bool IniConfiguration::getValue(QString key, QVariant &value) {
	if (!settings->contains(key)) {
	 qCritical("File settings.ini doesn't contain key %s or
fully empty" , qPrintable(key));
	   return false;
	}
	value = settings->value(key);
	return true;
}


/**
 * @brief	Устанавливает значение в конфигурации по указанному
ключу.
 * @param key	Ключ
 * @param value	Знвчение
 */
void IniConfiguration::setValue(QString key, QVariant &value) {
	settings->setValue(key, value);
}
```

Файл «postgresdb.cpp»

```
/*************************************************************
 * @file	postgresdb.cpp
 * @date	2014-2020
 * @version	0.1
 * @brief	Реализация модуля подключения к базе данных
PostgreSQL.
*************************************************************/

#include <QtSql>
#include <QMutexLocker>
#include "postgresdb.h"

PostgresDB::PostgresDB() {
	// добавляем требование наличия модуля
	requirements.append(LoggerIId);
	// добавляем требование наличия модуля конфигурации
```

```
        requirements.append(ConfigurationIId);
}


PostgresDB::~PostgresDB() {

}


/**
 * @brief Инициализирует модуль.
 * @return Статус инициализации.
 */
bool PostgresDB::init() {
        // загружаем модуль логирования
        Module loggerModule = moduleStorage-
>getModule(QString(LoggerIId));
        if (!loggerModule) {
           return false;
        }
        logger = qobject_cast<Logger *>(loggerModule);
        if (!logger) {
           return false;
        }

        // загружаем модуль конфигурации
        Module configurationModule = moduleStorage-
>getModule(ConfigurationIId);
        if (!configurationModule) {
           return false;
        }
        configuration = qobject_cast<Configuration
*>(configurationModule);
        if (!configuration) {
           return false;
        }

        // загружаем конфигурацию подключения к базе данных
        QVariant value;
        if (!configuration->getValue("postgres/host", value)) {
           logger->critical("Unable to find database host for
Postgres!");
           return false;
        }
        host = value.toString();

        if (!configuration->getValue("postgres/port", value)) {
           logger->critical("Unable to find database port for
Postgres!");
           return false;
        }
        bool conversionOk;
        port = value.toInt(&conversionOk);
```

```
        if (!conversionOk) {
            logger->critical("Unable to convert database port to int
for Postgres!");
            return false;
        }

        if (!configuration->getValue("postgres/user", value)) {
            logger->critical("Unable to find database user for
Postgres!");
            return false;
        }
        user = value.toString();

        if (!configuration->getValue("postgres/password", value)) {
            logger->critical("Unable to find database password for
Postgres!");
            return false;
        }
        password = value.toString();

        if (!configuration->getValue("postgres/db", value)) {
            logger->critical("Unable to find database name for
Postgres!");
            return false;
        }
        dbName = value.toString();

        // инициализируем подключение к базе данных
        db = QSqlDatabase::addDatabase("QPSQL");
        db.setHostName(host);
        db.setPort(port);
        db.setDatabaseName(dbName);
        db.setUserName(user);
        db.setPassword(password);

        logger->debug(QString("Opening postgres database
connection... Host: %1, port: %2,"
                              "db: %3, user: %4, password:
%5.").arg(host).arg(port).arg(dbName).
                      arg(user).arg(password));
        if (!db.open()) {
            logger->critical("Unable to open postgres database
connection!");
            logger->critical(db.lastError().text());
            return false;
        }

        return true;
}


/**
 * @brief Запускает модуль на выполнение.
```

```
 */
void PostgresDB::launch() {
	logger->debug("Postgres started");
}


/**
 * @brief		Выполняет запрос к базе данных.
 * @param queryString	Параметризованная строка запроса.
 * @param parameters	Параметры.
 * @param result	Результат запроса.
 * @return		Успешность выполнения запроса.
 */
bool PostgresDB::query(QString queryString, QueryParameters
*parameters, QueryResult *result) {
	QMutexLocker locker(&queryMutex);
	QSqlQuery query(db);
	if (!query.prepare(queryString)) {
	   logger->warning(QString("Unable to prepare query: %1.
Message: %2").arg(queryString)
				.arg(query.lastError().text()));
	   return false;
	}

	if (parameters != DatabaseQueryNoParameters) {
	   QueryParameters::const_iterator i;
	   for (i = parameters->constBegin(); i != parameters-
>constEnd(); i++) {
		query.addBindValue(*i);
	   }
	}

	if (!query.exec()) {
	   logger->warning(QString("Unable to exec query: %1.
Message: %2")

	.arg(getLastExecutedQuery(query)).arg(query.lastError().text
()));
	   return false;
	}

	if (result != DatabaseQueryNoResult) {
	   result->clear();

	   // получаем список полей в результате запроса
	   QSqlRecord record = query.record();
	   int fieldsCount = record.count();

	   QList<QString> fields;
	   for (int i = 0; i < fieldsCount; i++) {
		fields.append(record.field(i).name());
	   }
```

```
            while (query.next()) {
                 QueryResultRow row;
                 for (int i = 0; i < fieldsCount; i++) {
                      row[fields.at(i)] = query.value(i);
                 }
                 result->append(row);
            }
       }

       return true;
}


QString PostgresDB::getLastExecutedQuery(const QSqlQuery& query) {
       QString sql = query.executedQuery();
       const int nbBindValues = query.boundValues().size();

       for(int i = 0, j = 0; j < nbBindValues; ++j)
       {
          i = sql.indexOf(QLatin1Char('?'), i);
          if (i <= 0)
          {
               break;
          }
          const QVariant &var = query.boundValue(j);
          QSqlField field(QLatin1String(""), var.type());
          if (var.isNull())
          {
               field.clear();
          }
          else
          {
               field.setValue(var);
          }
          QString formatV = query.driver()->formatValue(field);
          sql.replace(i, 1, formatV);
          i += formatV.length();
       }

       return sql;
}
```

Файл «deviceserver.cpp»

```
/*************************************************************
 * @file       deviceserver.cpp
 * @date       2014-2020
 * @version         0.1
 * @brief      Реализация модуля, обслуживающего подключения
терминалов.

*************************************************************/

#include "deviceserver.h"
```

```
DeviceServer::DeviceServer() {
	// добавляем требование наличия модуля логирования
	requirements.append(LoggerIId);
	// добавляем требование наличия модулей терминалов для
приборов
	requirements.append(DeviceTerminalIId);
}


DeviceServer::~DeviceServer() {
	qDeleteAll(terminals);
	qDeleteAll(servers);
}


/**
 * @brief Инициализирует модуль.
 * @return	Статус инициализации.
 */
bool DeviceServer::init() {
	// загружаем модуль логирования
	Module loggerModule = moduleStorage-
>getModule(QString(LoggerIId));
	if (!loggerModule) {
	   return false;
	}
	logger = qobject_cast<Logger *>(loggerModule);
	if (!logger) {
	   return false;
	}

	// получаем список модулей, реализующих интерфейс терминалов
для приборов
	ModuleList modules = moduleStorage-
>getModules(DeviceTerminalIId);

	// заполняем список терминалов
	ModuleList::const_iterator i;
	for (i = modules.constBegin(); i != modules.constEnd(); i++)
{
	   Module module = *i;
	   DeviceTerminal *terminal = qobject_cast<DeviceTerminal
*>(module);
	   if (!terminal) {
	        return false;
	   }
	   terminals.append(terminal);
	}

	return true;
}
```

```
/**
 * @brief Запускает модуль.
 */
void DeviceServer::launch() {
       // проходим по списку терминалов
       TerminalList::const_iterator i;
       for (i = terminals.constBegin(); i != terminals.constEnd();
i++) {
          DeviceTerminal *terminal = *i;
          // создаем tcp-сервера для каждого терминала
          DeviceTcpServer *server = new DeviceTcpServer(this);

          // отслеживаем события сервера
          connect(server, SIGNAL(newIncomingConnection(qintptr)),
this, SLOT(newConnection(qintptr)));
          connect(server,
SIGNAL(acceptError(QAbstractSocket::SocketError)),
                  this,
SLOT(serverError(QAbstractSocket::SocketError)));

          // запускаем сервер для приема соединений
          if (!server->listen(QHostAddress(terminal->getTcpHost()),
terminal->getTcpPort())) {
               logger->critical(QString("Unable to launch TCP
server for %1 at %2:%3")
                                    .arg(terminal-
>getDeviceType()).arg(terminal->getTcpHost())
                                    .arg(QString::number(terminal-
>getTcpPort())));
               delete server;
          }
          else {
               logger->debug(QString("TCP server for %1
successfully started at %2:%3")
                                .arg(terminal-
>getDeviceType()).arg(terminal->getTcpHost())
                                .arg(QString::number(terminal-
>getTcpPort())));

               // добавляем сервер в списки
               servers.append(server);
               serverToTerminalRelations[server] = terminal;
          }
       }

       logger->debug("Device Server started.");

       start();
}
```

```
/**
 * @brief Обработчик события приема нового соединения.
 * @param handle    Дескриптор сокета.
 */
void DeviceServer::newConnection(qintptr handle) {
	// определяем источник события
	QObject *obj = sender();
	DeviceTcpServer *server = dynamic_cast<DeviceTcpServer
*>(obj);
	if (!server) {
	   logger->critical(QString("Unable to cast sender to TCP
server while accepting new connection!"));
	   return;
	}
	if (!servers.contains(server)) {
	   logger->critical(QString("Unable to find TCP server while
accepting new connection!"));
	   return;
	}

	// выясняем к какому терминалу относится новое соединение
	if (!serverToTerminalRelations.keys().contains(server)) {
	   logger->critical(QString("Unable to find terminal for new
connection!"));
	   return;
	}
	DeviceTerminal *terminal =
serverToTerminalRelations[server];

	// передаем сокет на обработку нужному терминалу
	terminal->newConnection(handle);
}


/**
 * @brief      Обработчик события возникновения ошибки в сервере.
 * @param error Ошибка.
 */
void DeviceServer::serverError(QAbstractSocket::SocketError error)
{
	Q_UNUSED(error);

	// определяем источник события
	QObject *obj = sender();
	DeviceTcpServer *server = dynamic_cast<DeviceTcpServer
*>(obj);
	if (!server) {
	   logger->critical(QString("Unable to cast sender to TCP
server while processing error!"));
	   return;
	}
	if (!servers.contains(server)) {
```

```
            logger->critical(QString("Unable to find TCP server while
processing error!"));
            return;
        }

        // выясняем к какому терминалу относится ошибка
        if (!serverToTerminalRelations.keys().contains(server)) {
            logger->critical(QString("Unable to find terminal while
processing TCP server error!"));
            return;
        }
        DeviceTerminal *terminal =
serverToTerminalRelations[server];

        logger->critical(QString("Error occured in server for %1.
Message: %2")
                              .arg(terminal-
>getDeviceType()).arg(server->errorString()));
}
```

Код четвертой части:

```
version: "3.7"
services:
  webserver_transport:
    build:
      context: './php/'
    ports:
      - "80:80"
    networks:
      - backend
      - frontend
    volumes:
      - ${PROJECT_ROOT}/:/var/www/html/
    environment:
      XDEBUG_CONFIG: remote_host=host.docker.internal
    depends_on:
      - mysql_transport
    container_name: webserver_transport
  mysql_transport:
    image: mysql:5.7
    command: ['mysqld', '--character-set-server=utf8mb4', '--collation-server=utf8mb4_unicode_ci']
    ports:
      - "3306:3306"
    volumes:
      - ./mysql:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: root3004917779
      MYSQL_USER: yurijmo
```

```
      MYSQL_PASSWORD: 3004917779
      MYSQL_DATABASE: db_transport
    networks:
      - backend
    container_name: mysql_transport
  phpmyadmin_transport:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin_transport
    ports:
      - 2323:80
    links:
      - mysql_transport:db
    networks:
      - backend
networks:
  frontend:
  backend:
volumes:
  data:

FROM php:7.3-apache
RUN docker-php-ext-install mysqli
RUN pecl install xdebug-2.7.0
RUN docker-php-ext-enable xdebug
RUN echo "xdebug.remote_enable=1" >> /usr/local/etc/php/php.ini
RUN docker-php-ext-install mysqli pdo pdo_mysql
RUN echo "extension=pdo_mysql" >> /usr/local/etc/php/php.ini
```

```
<!doctype html>
<html lang="en">

<head>
    <title>Title</title>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.1.1/css/all.css"
        integrity="sha384-O8whS3fhG2OnA5Kas0Y9l3cfpmYjapjI0E4theH4iuMD+pLhbf6JI0jIMfYcK3yZ"
crossorigin="anonymous">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
    <link rel="stylesheet" href="./css/main.css">
    <link rel="stylesheet" href="./js/main.js">
</head>

<body>


<header>
```

```
<nav class="navbar navbar-expand-lg navbar-light" style="background: #F2F2F3;">

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup"
aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="container">
<div class="collapse navbar-collapse" id="navbarNavAltMarkup">
<div class="navbar-nav">
<a id="nav_topbar" class="nav-item nav-link text-dark" href="/">Маршруты <span
class="sr-only">(current)</span></a>
<a id="nav_topbar" class="nav-item nav-link text-dark" href="./layout/timemndg.html">Расписание
</a>
<a id="nav_topbar" class="nav-item nav-link text-dark" href="#">График</a>
<a id="nav_topbar" class="nav-item nav-link text-dark" href="./layout/vid.html">Вид</a>
<a id="nav_topbar" class="nav-item nav-link text-dark" href="#">Расчет</a>
<a id="nav_topbar" class="nav-item nav-link text-dark" href="#">Справка</a>

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

```
    </nav>
</header>

<div class="container" style="height: 100%;">
    <span class="add"><i class="fas fa-pencil-alt"></i></span>
    <div class="row" style="margin-top: 10px; height: 100%;">

        <div class="col-4">
            <div class="rightSidebar_wrapper ">
                <div class="sidebar_content">
                    <div class="bar_title">
                        <p> Обьекты</p>
                    </div>

                    <ul class="items_lists_">
                        <li class="list_item">
                            <span> <a class="collaps_tem" href="#" data-
toggle="collapse" data-target="#collapseOne"
                                aria-expanded="true" aria-controls="collapseOne">
                                <i class="fas fa-chevron-circle-up"></i></a></span>
Транспорт
                            <div class="accordion" id="accordionExample">
                                <div class="card">

                                    <h2 class="mb-0">


                                    </h2>
                                </div>
```

```
                        <div id="collapseOne" class="collapse" aria-
labelledby="headingOne"
                            data-parent="#accordionExample">
                            <div class="card-body">

                            </div>
                        </div>
                    </div>
                </li>
                <li class="list_item">
                    <span> <a class="collaps_tem" href="#" data-
toggle="collapse" data-target="#collapseTwo"
                            aria-expanded="true" aria-controls="collapseOne">
                            <i class="fas fa-chevron-circle-up"></i></a></span>
Маршруты
                    <div class="accordion" id="accordionExample">
                        <div class="card">

                            <h2 class="mb-0">


                            </h2>
                        </div>

                        <div id="collapseTwo" class="collapse" aria-
labelledby="headingTwo"
                            data-parent="#accordionExample">
                            <div class="body_tabl">
                                <span><i class="add_plus far fa-plus-square"
                                        style="color: black;"></i></span>
```

```
<div class="scroll" style="overflow-y: scroll;">
  <table class="table table-bordered ">
    <thead class="thead-dark">
    <tr>
      <th scope="col-1"></th>
      <th scope="col-1"></th>
      <th scope="col">Название</th>
      <th scope="col-3" style="font-size: 15px;
        ">Тип ТС
      </th>
      <th scope="col-3"></th>

    </tr>
    </thead>
    <tbody>
    <tr>
      <th><input type="checkbox"></th>
      <td></td>
      <td>СтавНИИГиМ</td>
      <td></td>
      <td>
        <button><i class="fas fa-trash-alt"></i></button>
      </td>

    </tr>

    </tbody>
  </table>
```

```
                    </div>
                </div>
            </div>
        </li>
        <li class="list_item">
            <span> <a class="collaps_tem" href="#" data-toggle="collapse"
                    data-target="#collapseThree" aria-expanded="true"
                    aria-controls="collapseThree">
                <i class="fas fa-chevron-circle-up"></i></a></span>
Контрольные точки
            <div class="accordion" id="accordionExample">
                <div class="card">

                    <h2 class="mb-0">


                    </h2>
                </div>

                <div id="collapseThree" class="collapse" aria-labelledby="headingThree"
                    data-parent="#accordionExample">
                    <div class="body_tabl">
                        <span><i class="add_plus far fa-plus-square"
                                style="color: black;"></i></span>
                        <div class="scroll" style="overflow-y: scroll;">
                            <table class="table table-bordered ">
                                <thead class="thead-dark">
                                    <tr>
```

```
                    <th scope="col-2">id</th>
                    <th scope="col">Название</th>
                    <th scope="col-3"></th>

                </tr>
                </thead>
                <tbody>
                <tr>
                    <th scope="row">1</th>
                    <td>СтавНИИГиМ</td>
                    <td>
                        <button><i class="fas fa-trash-alt"></i></button>
                    </td>

                </tr>
                <tr>
                    <th scope="row">2</th>
                    <td>СКФУ</td>
                    <td>
                        <button><i class="fas fa-trash-alt"></i></button>
                    </td>

                </tr>
                <tr>
                    <th scope="row">3</th>
                    <td>Универсам</td>
                    <td>
```

<button></button>
</td>

</tr>
</tbody>
</table>

<li class="list_item">
 <a class="collaps_tem" href="#" data-toggle="collapse"
data-target="#collapseFour" aria-expanded="true" aria-controls="collapseFour">
 Водители

<h2 class="mb-0">

</h2>

<div id="collapseFour" class="collapse" aria-labelledby="headingFour"

```
                                data-parent="#accordionExample">
                                <div class="body_tabl">

                                </div>
                            </div>
                        </div>
                    </li>
                    <li class="list_item">
                        <span> <a class="collaps_tem" href="#" data-
toggle="collapse"
                                data-target="#collapseFive" aria-expanded="true"
aria-controls="collapseFive">
                                <i class="fas fa-chevron-circle-up"></i></a></span>
Расписание
                        <div class="accordion" id="accordionExample">
                            <div class="card">


                            </div>

                            <div id="collapseFive" class="collapse" aria-
labelledby="headingFive"
                                data-parent="#accordionExample">
                                <div class="body_tabl">
                                    <span><i class="add_plus far fa-plus-square"
                                            style="color: black;"></i></span>
                                    <div class="scroll" style="overflow-y: scroll;">
                                        <table class="table table-bordered "
style="overflow-y: scroll;">
                                            <thead class="thead-dark">
```

```
                                    <tr>
                                        <th scope="col-2">id</th>
                                        <th scope="col">Название</th>
                                        <th scope="col-3"></th>

                                    </tr>
                                    </thead>
                                    <tbody>
                                    <tr>
                                        <th scope="row">1</th>
                                        <td>12:00</td>
                                        <td>
                                            <button><i class="fas fa-trash-alt"></i></button>
                                        </td>

                                    </tr>
                                    </tbody>
                                </table>
                            </div>
                        </div>
                    </div>
                </div>
            </li>
        </ul>
    </div>

</div>
</div>
<div class="col-8">
```

```
<div class="content_wrapper">
    <div class="bar_title">
        <p> Маршрут</p>
    </div>

    <div class="selector">
        <form class="wrapper_select_top">

            <ul>


                <li>
                    Наименование: <input class="select_item col-9" type="text">
                </li>
                <li>
                    Вид транспорта: <select class="select_item col-9" name="" id="">
                    <option value="">Автобус</option>
                    <option value="">Такси</option>
                    <option value="">Троллейбус</option>
                </select>
                </li>
                <li>
                    Эксплутационная скорость: <input class="select_item col-9" type="number">
                </li>
                <li>
                    Цвет Маршрута: <input class="select_item col-9" type="color" value="">
```

```
            </li>
        </ul>
    </form>
</div>

<div class="selector">
    <form action="" class="wrapper_select_bot">
        <div class="wrapper_select_bot_in">

            <div class="controll_set_wrapper">
                <input type="text" class="controll_set ">
                <button class="button_itemm ">
                    После выбранной
                </button>
                <button class=" button_itemm ">В конец</button>


            </div>

            <nav id="navbar-example2" class="navbar navbar-light bg-light">

            </nav>
            <ul class="tabs">
                <li>
                    <input type="radio" name="tabs" id="tab-1" checked>
                    <label for="tab-1">Прямое направление</label>
                    <div class="tab-content">
```

```
<div>

    <table class="table table-bordered ">
        <thead class="thead-dark">
        <tr>
            <th scope="col-1" class="tude">id
            </th>
            <th class="tude col-3" scope="">Name</th>
            <th class="tude col-2" scope="">Latitude</th>
            <th class="tude col-2 "
scope="">Longitude</th>
            <th class="tude col-2" scope="">Radius</th>
            <th class="tude  col" scope=""></th>

        </tr>
        </thead>
        <tbody>
        <tr>
            <th scope="row">1</th>
            <td>СтавНИИГиМ</td>

            <td>45.040501</td>
            <td>41.905497</td>
            <td>0</td>
            <td></td>

        </tr>
        <tr>
            <th scope="row">1</th>
            <td>СтавНИИГиМ</td>
```

```
    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>
```

```
    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>
```

```
    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>СтавНИИГиМ</td>
```

```
                    <td>45.040501</td>
                    <td>41.905497</td>
                    <td>0</td>
                    <td></td>

                </tr>
                <tr>
                    <th scope="row">1</th>
                    <td>СтавНИИГиМ</td>

                    <td>45.040501</td>
                    <td>41.905497</td>
                    <td>0</td>
                    <td></td>

                </tr>
                </tbody>
            </table>
        </div>
    </div>
</li>
<li>
    <input type="radio" name="tabs" id="tab-3">
    <label for="tab-3">Обратное направление</label>
    <div class="tab-content" style="overflow-y: scroll;">
        <table class="table table-bordered ">
            <thead class="thead-dark">
            <tr>
                <th scope="col-1" class="tude">id
```

```
    </th>
    <th class="tude col-3" scope="">Name</th>
    <th class="tude col-2" scope="">Latitude</th>
    <th class="tude col-2 " scope="">Longitude</th>
    <th class="tude col-2" scope="">Radius</th>
    <th class="tude  col" scope=""></th>

</tr>
</thead>
<tbody>
<tr>
    <th scope="row">1</th>
    <td>Скфуу</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
<tr>
    <th scope="row">1</th>
    <td>Универсам</td>

    <td>45.040501</td>
    <td>41.905497</td>
    <td>0</td>
    <td></td>

</tr>
```

```
                                    </tr>
                                    </tbody>
                                </table>
                            </div>


                        </li>
                    </ul>


                </div>

            </form>
        </div>
        <div class="footer_confirm_buttom">
            <button> Сохранить</button>
            <button> Отмена</button>

        </div>


    </div>
</div>

</div>
```

```
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script                                                    src="
https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
        integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0
W1" crossorigin="anonymous">
    </script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
        integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous">
    </script>

    <script src="./js/main.js">

    </script>
    </body>

    </html>
    <?php
    session_start();
    if (!isset($_SESSION['username'])) {
       header('location: index.php');
    }
    $title = "Авторизация в системе";
```

```
require_once("forms/LoginForm.php");
require_once('DBA.php');
require_once('includes/Password.php');
require_once('includes/Session.php');
require_once('Dbsettings.php');

$msg = '';

try {
    $db = new DBA($host, $user, $password, $db_name, $port);
} catch (Exception $e) {
}
$form = new LoginForm($_POST);

if ($_POST) {
    if ($form->validate()) {
        $username = $db->escape($form->getUsername());
        $password = new Password($db->escape($form->getPassword()));
        try {
            $res = $db->query("SELECT * FROM auth_user WHERE username =
'{$username}' AND password = '{$password}' LIMIT 1");
        } catch (Exception $e) {
        }
        if (!$res) {
            $msg = 'Пользователя с таким учетными данными нет';
        } else {
            $username = $res[0]['username'];
            $staff = $res[0]['is_staff'];
            Session::set('username', $username);
            Session::set('staff', $staff);
```

```
                header('location: lessons.php?msg=Вы авторизированы на сайте');
            }
        } else {
            $msg = 'Пожалуйста, заполните все поля';
        }
    }
    include_once('includes/header.php');
    ?>
        <div class="container mt-2">
            <div class="row">
                <div class="col-12">
                    <div class="jumbotron">
                        <h1 class="text-center">Учебные курсы </h1>
                        <p class="text-center">Обучение английскому языку</p>
                        <img src="images/LearningResearch.jpg" alt="" class="img-
fluid">
                    </div>
                </div>
            </div>
        </div>
        <div class="container mb-5">
        <div class="row">
            <div class="text-center col-md-12 col-12">
                <div class="container page-middle">
                    <div class="row">
                        <div class="col-md-6">
                            <h2 class="fat-title"></h2>
                            <span class="h2-description">
                </span>
                        </div>
```

```
                    <div class="col-md-6">
                        <form class="custom-form" method="post">
                            <h4>Войти</h4>
                            <label>
                                <input type='text' placeholder='Login' class='form-control custom-input' name="username"
                                       value="<?= $form->getUsername(); ?>">
                            </label>
                            <label>
                                <input type="password" placeholder="Password" class="form-control custom-input"
                                       name="password">
                            </label>
                            <input type="submit" value="Войти" class="btn custom-btn">
                            <div class="bottom-link">
                                <a href="#">У меня нет аккаунта!</a>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    <?php include_once('includes/footer.php'); ?>
    <!doctype html>
    <html lang="en">

    <head>
        <title>Title</title>
```

```
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.1.1/css/all.css"
        integrity="sha384-O8whS3fhG2OnA5Kas0Y9l3cfpmYjapjI0E4theH4iuMD+pLhbf6JI0jIMfYcK3yZ"
    crossorigin="anonymous">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">
    <link rel="stylesheet" href="./../css/main.css">
    <link rel="stylesheet" href="./js/main.js">
</head>

<body>


<header>
    <nav class="navbar navbar-expand-lg navbar-light" style="background: #F2F2F3;">

        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup"
```

```
                aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="container">
                <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
                    <div class="navbar-nav">
                        <a class="nav-item nav-link text-dark" href="/">Маршруты<span
                                class="sr-only">(current)</span></a>
                        <a class="nav-item nav-link text-dark" href="#">Расписание </a>
                        <a class="nav-item nav-link text-dark" href="#">График</a>
                        <a class="nav-item nav-link text-dark" href="./vid.html">Вид</a>
                        <a class="nav-item nav-link text-dark" href="#">Расчет</a>
                        <a class="nav-item nav-link text-dark" href="#">Справка</a>

                    </div>
                </div>
            </div>
        </nav>
    </header>

    <div class="container" style="height: 100%;">
        <span class="add"><i class="fas fa-pencil-alt"></i></span>
        <div class="row" style="margin-top: 10px; height: 100%;">

            <div class="col-4">
                <div class="rightSidebar_wrapper ">
                    <div class="sidebar_content">
                        <div class="bar_title">
                            <p> Обьекты</p>
```

```
                </div>

                <ul class="items_lists_">
                    <li class="list_item">
                        <span> <a class="collaps_tem" href="#" data-toggle="collapse" data-target="#collapseOne"
                                aria-expanded="true" aria-controls="collapseOne">
                            <i class="fas fa-chevron-circle-up"></i></a></span> Транспорт
                        <div class="accordion" id="accordionExample">
                            <div class="card">

                                <h2 class="mb-0">


                                </h2>
                            </div>

                            <div id="collapseOne" class="collapse" aria-labelledby="headingOne"
                                 data-parent="#accordionExample">
                                <div class="card-body">

                                </div>
                            </div>
                        </div>
                    </li>
                    <li class="list_item">
                        <span> <a class="collaps_tem" href="#" data-toggle="collapse" data-target="#collapseTwo"
```

```
                                aria-expanded="true" aria-controls="collapseOne">
                               <i  class="fas  fa-chevron-circle-up"></i></a></span>
Маршруты
                           <div class="accordion" id="accordionExample">
                              <div class="card">

                                 <h2 class="mb-0">



                                 </h2>
                              </div>

                              <div       id="collapseTwo"       class="collapse"       aria-
labelledby="headingTwo"
                                data-parent="#accordionExample">
                               <div class="body_tabl">
                                    <span><i class="add_plus far fa-plus-square"
                                         style="color: black;"></i></span>
                                  <div class="scroll" style="overflow-y: scroll;">
                                    <table class="table table-bordered ">
                                       <thead class="thead-dark">
                                       <tr>
                                          <th scope="col-1"></th>
                                          <th scope="col-1"></th>
                                          <th scope="col">Название</th>
                                          <th scope="col-3" style="font-size: 15px;
                                             ">Тип ТС
                                          </th>
                                          <th scope="col-3"></th>
```

```
                                    </tr>
                                    </thead>
                                    <tbody>
                                    <tr>
                                        <th><input type="checkbox"></th>
                                        <td></td>
                                        <td>СтавНИИГиМ</td>
                                        <td></td>
                                        <td>
                                            <button><i class="fas fa-trash-alt"></i></button>
                                        </td>

                                    </tr>

                                    </tbody>
                                </table>

                            </div>
                        </div>
                    </div>
                </li>
                <li class="list_item">
                    <span> <a class="collaps_tem" href="#" data-toggle="collapse"
                              data-target="#collapseThree" aria-expanded="true"
                              aria-controls="collapseThree">
                            <i class="fas fa-chevron-circle-up"></i></a></span> Контрольные точки
                    <div class="accordion" id="accordionExample">
```

```
<div class="card">

    <h2 class="mb-0">


    </h2>
</div>

<div id="collapseThree" class="collapse" aria-labelledby="headingThree"
     data-parent="#accordionExample">
    <div class="body_tabl">
        <span><i class="add_plus far fa-plus-square"
                 style="color: black;"></i></span>
       <div class="scroll" style="overflow-y: scroll;">
        <table class="table table-bordered ">
            <thead class="thead-dark">
            <tr>
                <th scope="col-2">id</th>
                <th scope="col">Название</th>
                <th scope="col-3"></th>

            </tr>
            </thead>
            <tbody>
            <tr>
                <th scope="row">1</th>
                <td>СтавНИИГиМ</td>
                <td>
```

```
                                    <button><i class="fas fa-trash-alt"></i></button>
                                </td>

                            </tr>
                            <tr>
                                <th scope="row">2</th>
                                <td>СКФУ</td>
                                <td>
                                    <button><i class="fas fa-trash-alt"></i></button>
                                </td>

                            </tr>
                            <tr>
                                <th scope="row">3</th>
                                <td>Универсам</td>
                                <td>
                                    <button><i class="fas fa-trash-alt"></i></button>
                                </td>

                            </tr>
                            </tbody>
                        </table>

                    </div>
                </div>
            </div>
        </li>
```

```
                    <li class="list_item">
                        <span>  <a  class="collaps_tem"  href="#"  data-
toggle="collapse"
                            data-target="#collapseFour"  aria-expanded="true"
aria-controls="collapseFour">
                           <i  class="fas  fa-chevron-circle-up"></i></a></span>
Водители
                       <div class="accordion" id="accordionExample">
                          <div class="card">

                            <h2 class="mb-0">



                           </h2>
                          </div>

                          <div  id="collapseFour"  class="collapse"  aria-
labelledby="headingFour"
                            data-parent="#accordionExample">
                           <div class="body_tabl">

                           </div>
                          </div>
                       </div>
                    </li>
                    <li class="list_item">
                        <span>  <a  class="collaps_tem"  href="#"  data-
toggle="collapse"
                            data-target="#collapseFive"  aria-expanded="true"
aria-controls="collapseFive">
```

```
                        <i  class="fas  fa-chevron-circle-up"></i></a></span>
Расписание
                    <div class="accordion" id="accordionExample">
                        <div class="card">



                        </div>

                        <div      id="collapseFive"      class="collapse"      aria-
labelledby="headingFive"
                             data-parent="#accordionExample">
                            <div class="body_tabl">
                                <span><i class="add_plus far fa-plus-square"
                                         style="color: black;"></i></span>
                                <div class="scroll" style="overflow-y: scroll;">
                                    <table      class="table      table-bordered      "
style="overflow-y: scroll;">
                                        <thead class="thead-dark">
                                        <tr>
                                            <th scope="col-2">id</th>
                                            <th scope="col">Название</th>
                                            <th scope="col-3"></th>

                                        </tr>
                                        </thead>
                                        <tbody>
                                        <tr>
                                            <th scope="row">1</th>
                                            <td>12:00</td>
                                            <td>
```

```
                                        <button><i class="fas fa-trash-alt"></i></button>
                                    </td>

                                </tr>
                                </tbody>
                            </table>
                        </div>
                    </div>
                </div>
            </li>
        </ul>
    </div>

</div>
<div class="col-8">
    <div class="content_wrapper">
        <div class="timemanadge_wrapper">
            <!-- Tab links -->
            <div class="tab">
                <button class="tablinks col-4" onclick="openCity(event, 'item_1')">Редактор расписания
                    1
                </button>
                <button class="tablinks col-4" onclick="openCity(event, 'item_2')">Редактор расписания
                    2
                </button>
```

```
                    <button class="tablinks col-4" onclick="openCity(event,
'item_3')">Детализация
                        расписания
                    </button>
                    <button class="tablinks bot col-6" onclick="openCity(event,
'item_4')"
                        style="  border-top: 1px solid gray; ">Контроль режима
                        отдыха и
                        труда
                    </button>
                    <button class="tablinks bot col-6"
                        style="  border-top: 1px solid gray;  padding: 14px 0px; "
                        onclick="openCity(event, 'item_5')">Графическая схема
                        расписания
                        расписания
                    </button>
                </div>

        <!—  Tab  content  —>
<div  id="item_1"  class="tabcontent">
<h3>Контент  №1</h3>
<p>Редактор  расписания  1  </p>
</div>

<div  id="item_2"  class="tabcontent">
<h3>Контент  №2</h3>
<p>Редактор  расписания  2  </p>
</div>

<div  id="item_3"  class="tabcontent">
```

```
<h3>Контент №3</h3>
<p>Детализация расписания </p>
</div>
<div id="item_4" class="tabcontent">
<h3>Контент №4</h3>
<p>Контроль режима отдыха и труда </p>
</div>
<div id="item_5" class="tabcontent">
<h3>Контент №5</h3>
<p>Графическая схема расписания </p>
</div>
</div>
</div>
</div>

        <!-- Optional JavaScript -->
        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
        <script src="
https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
            integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0
W1"
            crossorigin="anonymous">
        </script>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
            integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
```

```
            crossorigin="anonymous">
        </script>

        <script src="./../js/main.js">



        </script>
    </body>

    </html>

    <!doctype html>
    <html lang="en">

    <head>
        <title>Title</title>
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
        <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.1.1/css/all.css"
            integrity="sha384-O8whS3fhG2OnA5Kas0Y9l3cfpmYjapjI0E4theH4iuMD+pLhbf6JI0jIMfYcK3yZ"
        crossorigin="anonymous">

        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
```

```
            integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj
7Sk" crossorigin="anonymous">
        <link rel="stylesheet" href="./../css/main.css">
        <link rel="stylesheet" href="./js/main.js">
    </head>

    <body>



    <header>
        <nav class="navbar navbar-expand-lg navbar-light" style="background:
#F2F2F3;">

            <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNavAltMarkup"
                aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="container">
                <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
                    <div class="navbar-nav">
                        <a class="nav-item nav-link text-dark" href="/">Маршруты<span
                            class="sr-only">(current)</span></a>
                        <a class="nav-item nav-link text-dark"
href="./timemndg.html">Расписание </a>
                        <a class="nav-item nav-link text-dark" href="#">График</a>
                        <a class="nav-item nav-link text-dark" href="#">Вид</a>
                        <a class="nav-item nav-link text-dark" href="#">Расчет</a>
```

```
                    <a class="nav-item nav-link text-dark" href="#">Справка</a>

                </div>
            </div>
        </div>
    </nav>
</header>

<div class="container" style="height: 100%;">
    <span class="add"><i class="fas fa-pencil-alt"></i></span>
    <div class="row" style="margin-top: 10px; height: 100%;">

        <div class="col-4">
            <div class="rightSidebar_wrapper ">
                <div class="sidebar_content">
                    <div class="bar_title">
                        <p> Обьекты</p>
                    </div>

                    <ul class="items_lists_">
                        <li class="list_item">
                            <span> <a class="collaps_tem" href="#" data-
toggle="collapse" data-target="#collapseOne"
                                aria-expanded="true" aria-controls="collapseOne">
                                <i class="fas fa-chevron-circle-up"></i></a></span>
Транспорт
                            <div class="accordion" id="accordionExample">
                                <div class="card">

                                    <h2 class="mb-0">
```

```
                    </h2>
                </div>

                <div id="collapseOne" class="collapse" aria-labelledby="headingOne"
                    data-parent="#accordionExample">
                    <div class="card-body">

                    </div>
                </div>
            </div>
        </li>
        <li class="list_item">
            <span> <a class="collaps_tem" href="#" data-toggle="collapse" data-target="#collapseTwo"
                    aria-expanded="true" aria-controls="collapseOne">
                <i class="fas fa-chevron-circle-up"></i></a></span> Маршруты
            <div class="accordion" id="accordionExample">
                <div class="card">

                    <h2 class="mb-0">

                    </h2>
                </div>
```

```
                    <div id="collapseTwo" class="collapse" aria-
labelledby="headingTwo"
                     data-parent="#accordionExample">
                    <div class="body_tabl">
                        <span><i class="add_plus far fa-plus-square"
                             style="color: black;"></i></span>
                       <div class="scroll" style="overflow-y: scroll;">
                        <table class="table table-bordered ">
                            <thead class="thead-dark">
                            <tr>
                                <th scope="col-1"></th>
                                <th scope="col-1"></th>
                                <th scope="col">Название</th>
                                <th scope="col-3" style="font-size: 15px;
                                   ">Тип ТС
                                </th>
                                <th scope="col-3"></th>

                            </tr>
                            </thead>
                            <tbody>
                            <tr>
                                <th><input type="checkbox"></th>
                                <td></td>
                                <td>СтавНИИГиМ</td>
                                <td></td>
                                <td>
                                    <button><i class="fas fa-trash-
alt"></i></button>
                                </td>
```

```
                                    </tr>

                                    </tbody>
                                </table>

                            </div>
                        </div>
                    </div>
                </li>
                <li class="list_item">
                    <span> <a class="collaps_tem" href="#" data-toggle="collapse"
                            data-target="#collapseThree" aria-expanded="true"
                            aria-controls="collapseThree">
                        <i class="fas fa-chevron-circle-up"></i></a></span> Контрольные точки
                    <div class="accordion" id="accordionExample">
                        <div class="card">

                            <h2 class="mb-0">


                            </h2>
                        </div>

                        <div id="collapseThree" class="collapse" aria-labelledby="headingThree"
                             data-parent="#accordionExample">
                            <div class="body_tabl">
```

```
                    <span><i class="add_plus far fa-plus-square"
                            style="color: black;"></i></span>
                <div class="scroll" style="overflow-y: scroll;">
                    <table class="table table-bordered ">
                        <thead class="thead-dark">
                        <tr>
                            <th scope="col-2">id</th>
                            <th scope="col">Название</th>
                            <th scope="col-3"></th>

                        </tr>
                        </thead>
                        <tbody>
                        <tr>
                            <th scope="row">1</th>
                            <td>СтавНИИГиМ</td>
                            <td>
                                <button><i class="fas fa-trash-
alt"></i></button>
                            </td>

                        </tr>
                        <tr>
                            <th scope="row">2</th>
                            <td>СКФУ</td>
                            <td>
                                <button><i class="fas fa-trash-
alt"></i></button>
                            </td>
```

```
                                    </tr>
                                    <tr>
                                        <th scope="row">3</th>
                                        <td>Универсам</td>
                                        <td>
                                            <button><i class="fas fa-trash-alt"></i></button>
                                        </td>

                                    </tr>
                                    </tbody>
                                </table>

                            </div>
                        </div>
                    </div>
                </li>
                <li class="list_item">
                    <span> <a class="collaps_tem" href="#" data-toggle="collapse"
                              data-target="#collapseFour" aria-expanded="true" aria-controls="collapseFour">
                            <i class="fas fa-chevron-circle-up"></i></a></span> Водители
                    <div class="accordion" id="accordionExample">
                        <div class="card">

                            <h2 class="mb-0">
```

```
                    </h2>
                </div>

                <div id="collapseFour" class="collapse" aria-labelledby="headingFour"
                    data-parent="#accordionExample">
                    <div class="body_tabl">

                    </div>
                </div>
            </div>
        </li>
        <li class="list_item">
            <span> <a class="collaps_tem" href="#" data-toggle="collapse"
                    data-target="#collapseFive" aria-expanded="true" aria-controls="collapseFive">
                    <i class="fas fa-chevron-circle-up"></i></a></span> Расписание
            <div class="accordion" id="accordionExample">
                <div class="card">

                </div>

                <div id="collapseFive" class="collapse" aria-labelledby="headingFive"
                    data-parent="#accordionExample">
                    <div class="body_tabl">
                        <span><i class="add_plus far fa-plus-square"
```

```
                                    style="color: black;"></i></span>
                                <div class="scroll" style="overflow-y: scroll;">
                                    <table class="table table-bordered "
style="overflow-y: scroll;">
                                        <thead class="thead-dark">
                                        <tr>
                                            <th scope="col-2">id</th>
                                            <th scope="col">Название</th>
                                            <th scope="col-3"></th>

                                        </tr>
                                        </thead>
                                        <tbody>
                                        <tr>
                                            <th scope="row">1</th>
                                            <td>12:00</td>
                                            <td>
                                                <button><i class="fas fa-trash-
alt"></i></button>
                                            </td>

                                        </tr>
                                        </tbody>
                                    </table>
                                </div>
                            </div>
                        </div>
                    </li>
                </ul>
```

```
            </div>

        </div>
    </div>
    <div class="col-8">
        <div class="content_wrapper">
            <div class="vid_wrapper">
                <ul class="model">
                    <li class="model_item">
                        <h1><b> Номер:</b> 1</h1>
                    </li>
                    <li class="model_item">
                        <h1><b> Название:</b> СтавНИИГиМ</h1>
                    </li>
                    <li class="model_item">
                        <h1><b> Широта:</b> 45.040501</h1>
                    </li>
                    <li class="model_item">
                        <h1><b> Долгота:</b> 41.905498</h1>
                    </li>
                </ul>
            </div>
            <div/>
        </div>

        <!-- Optional JavaScript -->
        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
        <script src="
https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
            integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0
W1"
            crossorigin="anonymous">
        </script>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
            integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
            crossorigin="anonymous">
        </script>

        <script src="./../js/main.js">



        </script>
    </body>

    </html>
```