МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Прикладная информатика в социальной сфере
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Автоматизация выдачи разрешений на перевозку тяжелых грузов по территории г. о. Тольятти

Студент Н. Н. Ширяев

Руководитель Н. Н. Казаченок

Допустить к защите
Заведующий кафедрой к.тех.н, доцент, А.В. Очеповский

«_____» _____ 2019г.

Тольятти 2019

АННОТАЦИЯ

С. 56, рис. 55, лит. 20 источников

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА, ЛОГИСТИКА, IDEF0, МОДЕЛЬ, UML, БАЗА ДАННЫХ

Целью выпускной бакалаврской работы является разработка информационной системы выдачи разрешений на перевозку тяжелого и крупногабаритного груза по территории г.о. Тольятти.

Объект исследования – деятельность, направленная на выдачу разрешений заявителям, а именно прием и регистрация заявлений, выдача разрешений и сопроводительной документации.

В рамках работы был проведен анализ предметной области и существующих систем-аналогов. Программная реализация модуля учета заявлений выполнена в среде разработки «1С: Предприятие 8.3» на встроенном языке программирования и функционирует под управлением операционной системы Windows 7 и выше. Программная реализация картографического модуля выполнена в среде разработки MS Visual Studio на языках JavaScript и C#, с использованием технологий Ajax. JQuery и библиотек ESRI JS API. При желании можно работать в ОС Linux Ubuntu 18.04 LTS и выше.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Один из специалистов Департамента дорожного хозяйства и транспорта г.о. Тольятти занимается приемом заявлений и выдачей разрешений (согласований) на перевоз тяжелых и крупногабаритных грузов по территории г.о. Тольятти. Регламент подразумевает ведение базы данных выданных разрешений (согласований), печать ряда документов и расчет возмещения вреда, причиняемого транспортным средством дорожному полотну.

На момент получения ТЗ расчет выполнялся в Excel, база данных велась внутри приложения, написанного на Assembler, а печать документов подразумевала ручное заполнение каждого отдельного документа из блока. Задача осложнялась еще тем, что документы для каждого заявителя заполнялись в разное время. Все это приводило к большим затратам времени на обработку каждого заявления.

Актуальность выпускной квалификационной работы обусловлена необходимостью автоматизации процесса выдачи разрешений на перевозку тяжелых грузов по территории г.о. Тольятти.

Объектом исследования является процесс выдачи разрешений на перевозку тяжелых грузов.

Предметом исследования является автоматизация процесса выдачи разрешений на перевозку тяжелых грузов.

Цель выпускной бакалаврской работы – разработка информационной системы для автоматизации процесса выдачи разрешений на перевозку тяжелых грузов по территории г.о. Тольятти.

Задачи:

- Проанализировать существующую систему выдачи разрешений;
- Изучить документы, регламентирующие процесс выдачи разрешений;
- Автоматизировать процесс приема заявления;
- Автоматизировать процесс формирования документов;
- Обеспечить возможность запуска на любой машине.

В состав выпускной квалификационной работы входит введение, две главы, заключение и список используемой литературы.

В первой главе проводится анализ процесса выдачи разрешений.

Во второй главе идет описание требований к разрабатываемой автоматизированной информационной системе, ее логическое проектирование, описывается техническое задание для разработки.

В третьей главе проводится описание процессов по разработке и реализации автоматизированной информационной системы.

В заключении подводятся итоги, формируются выводы и описывается результат проведенной работы.

Итогом бакалаврской работы является успешно разработанная и внедренная система автоматизации выдачи разрешений на перевозку тяжелых грузов по территории г.о. Тольятти.

ГЛАВА 1 АНАЛИЗ ВЫДАЧИ РАЗРЕШЕНИЙ НА ПЕРЕВОЗКУ ТЯЖЕЛЫХ ГРУЗОВ

1.1 Описание процесса выдачи разрешений

24 июля 2012 г. Министерства транспорта Российской Федерации (Минтранс России) выпустило Приказ N 258 "Об утверждении Порядка выдачи специального разрешения на движение по автомобильным дорогам транспортного средства, осуществляющего перевозки тяжеловесных и (или) крупногабаритных грузов". Он обязывает всех граждан, осуществляющих перевозку тяжелых и/или крупногабаритных грузов по дорогам общего пользования, подавать заявление на получение разрешения (согласования). Приемом заявлений и выдачей разрешений/согласований занимается специалист департамента дорожного хозяйства городского округа.

Система, имевшаяся в распоряжении Департамента дорожного хозяйства и транспорта г.о. Тольятти, имела несколько недостатков:

- Разрозненность модулей;
- Невозможность использования на других машинах;
- Необходимость заполнять периодические заявления с нуля;
- Необходимость заполнять и печатать документы руками.

Как итог – имевшаяся система была неудобна в использовании, запускалась только на одной конкретной машине и требовала большого количества действий от специалиста, что приводило к большим затратам времени на обработку одного заявления.

1.2 Построение концептуальной модели

1.2.1 Выбор технологии концептуального моделирования

IDEF0 - методология функционального моделирования. С помощью наглядного графического языка IDEF0, изучаемая система предстает перед разработчиками и аналитиками в виде набора взаимосвязанных функций (функциональных блоков - в терминах IDEF0). Как правило, моделирование средствами IDEF0 является первым этапом изучения любой системы;

Графический язык IDEF0 удивительно прост и гармоничен. В основе методологии лежат четыре основных понятия.

Первым из них является понятие функционального блока (Activity Box). Функциональный блок графически изображается в виде и олицетворяет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, “производить услуги”, а не “производство услуг”).

Каждая из четырех сторон функционального блока имеет своё определенное значение (роль), при этом:

- Верхняя сторона имеет значение “Управление” (Control);
- Левая сторона имеет значение “Вход” (Input);
- Правая сторона имеет значение “Выход” (Output);
- Нижняя сторона имеет значение “Механизм” (Mechanism).

Вторым “китом” методологии IDEF0 является понятие интерфейсной дуги (Arrow). Также интерфейсные дуги часто называют потоками или стрелками. Интерфейсная дуга отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на функцию, отображенную им.

Обязательное наличие управляющих интерфейсных дуг является одним из главных отличий стандарта IDEF0 от других методологий классов DFD (Data Flow Diagram) и WFD (Work Flow Diagram).

Третьим основным понятием стандарта IDEF0 является декомпозиция (Decomposition). Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

Последним из понятий IDEF0 является глоссарий (Glossary). Для каждого из элементов IDEF0 существующий стандарт подразумевает создание и поддержание набора соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект,

отображенный данным элементом. Этот набор называется глоссарием и является описанием сущности данного элемента. Например, для управляющей интерфейсной дуги “распоряжение об оплате” глоссарий может содержать перечень полей соответствующего дуге документа, необходимый набор виз и т.д. Глоссарий гармонично дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

1.2.2 Разработка моделей бизнес-процессов выдачи разрешения

Согласно приказа №258, заявитель должен заполнить и подать специалисту департамента заявление по установленной форме. В заявлении указываются ТТХ транспортного средства, вес и габариты груза. На основании полученных данных производится расчет вреда, причиняемого транспортным средством дорожному полотну и размер компенсации. Заявитель оплачивает компенсацию и, в случае согласования, госпошлину в кассу. Чек об оплате подается специалисту Департамента дорожного хозяйства и транспорта г.о. Тольятти, формируется блок документов, разрешающий проезд транспортного средства по указанному маршруту, в соответствии с рисунком 1.1.

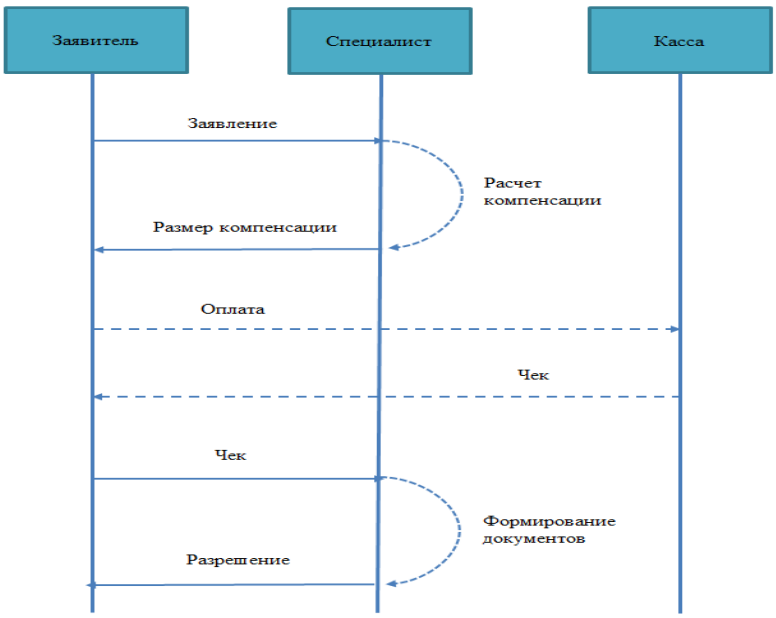

Рисунок 1.1 – Диаграмма последовательности процесса «Получение разрешения на перевозку тяжелых грузов»

Модель бизнес-процесса «КАК ЕСТЬ» описывает принципы организации процесса выдачи разрешений. В соответствии с рисунком 1.2, в существующей модели для выдачи разрешения специалист должен получить заявление и зарегистрировать его в программе (специальном ПО). Далее данные переносятся специалистом в файл Excel, в котором рассчитывается размер компенсации. После этого специалист печатает платежное поручение из документа Word. Заявитель оплачивает компенсацию и приносит чек специалисту. Он, в свою очередь, отмечает заявление в программе как отработанное и печатает блок документов из Word.

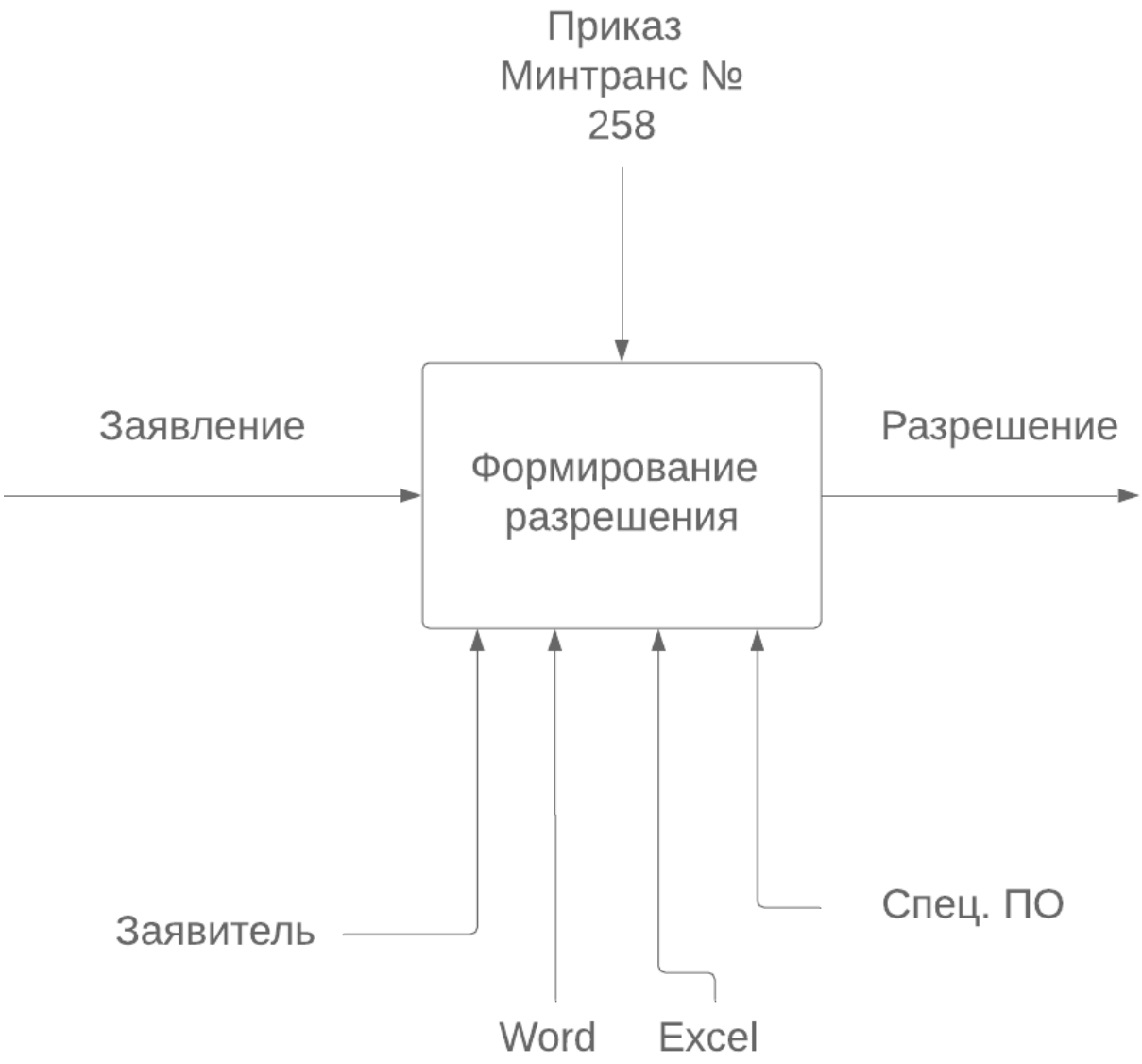

Рисунок 1.2 – диаграмма IDEF0 верхнего уровня «КАК ЕСТЬ» процесса «Формирование разрешения»

Декомпозиция процесса выдачи разрешения представлена на рисунке 1.3.

На диаграммах изображены следующие элементы:

- Выходные данные – разрешение;
- Входные данные – заявление;
- Управляющие воздействия – приказ МинТранс №258;
- Исполнители – Заявитель, Word, Excel, спец. ПО.

Данная модель является основой для анализа и дальнейшего совершенствования процесса выдачи разрешений.

Прием заявления и подготовка разрешения осуществляются специалистом в спец. ПО, с нуля для каждого заявления. Такой способ занимает много времени и сокращает количество обрабатываемых заявлений в день. Также, сопроводительная документация и бланк распоряжения формируются специалистом в Word.

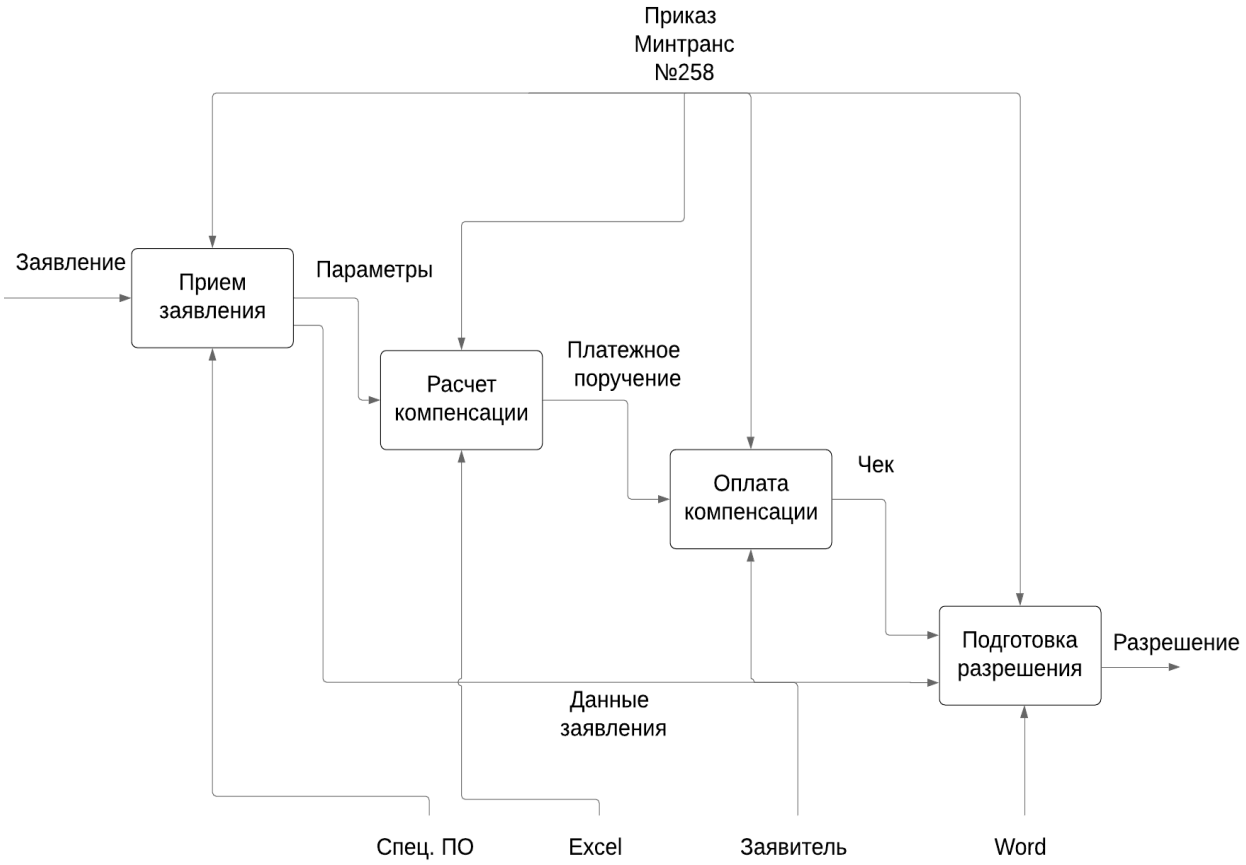

Рисунок 1.3 – модель декомпозиции процесса «Формирование разрешения» «КАК ЕСТЬ»

Анализ модели «Как есть» выявил ряд недостатков существующей системы:

- Невозможность автоматически заполнять периодические заявления;
- Необходимость формирования сопроводительной документации вручную;

Опираясь на вышесказанное, принято решение по улучшению процесса выдачи разрешений путем разработки и внедрения автоматизированной системы регистрации заявлений и выдачи разрешений.

1.3 Выявление требований к автоматизации бизнес-процессов

Модель «КАК ДОЛЖНО БЫТЬ» – это основа для создания технического задания для создания и модернизации автоматизированной системы. Представляет собой концептуальную модель усовершенствованного процесса.

Проанализировав модель «КАК ЕСТЬ» процесса выдачи разрешений, я выявил, что основной недостаток состоит в том, что специалисту необходимо заново регистрировать данные периодических заявлений, а также вручную формировать сопроводительную документацию.

С учетом перечисленных недостатков была разработана модель процесса выдачи разрешений «КАК ДОЛЖНО БЫТЬ», в соответствии с рисунком 1.4.

На рисунке 1.4 представлена IDEF0-диаграмма автоматизированной системы выдачи разрешений «КАК ДОЛЖНО БЫТЬ» (1-й уровень). Благодаря данной диаграмме, наглядно отображено, что автоматизированная система берет на себя функции спец. ПО, Word и Excel.

Улучшение данной системы достигается путем разработки и внедрения автоматизированной системы выдачи разрешений.

Система должна взять на себя формирование документов и ведение регистров. Также должна быть возможность автоматического заполнение формы документа на основе данных из формы авто-заполнения.

Рисунок 1.4 – диаграмма IDEF0 верхнего уровня «КАК ДОЛЖНО БЫТЬ» процесса «Автоматизация процесса формирования разрежения»

На рисунке 1.5 отображена IDEF0-диаграмма декомпозиции процесса выдачи разрешения «КАК ДОЛЖНО БЫТЬ».

На диаграммах изображены следующие элементы:

- Выходные данные – разрешение;
- Входные данные – заявление;
- Управляющие воздействия – приказ МинТранс №258;
- Исполнители – Заявитель, Автоматизированная система.

В ожидаемой модели прием заявления осуществляется в автоматизированной системе, которая, после этого, рассчитывает размер компенсации и формирует сопроводительную документацию.

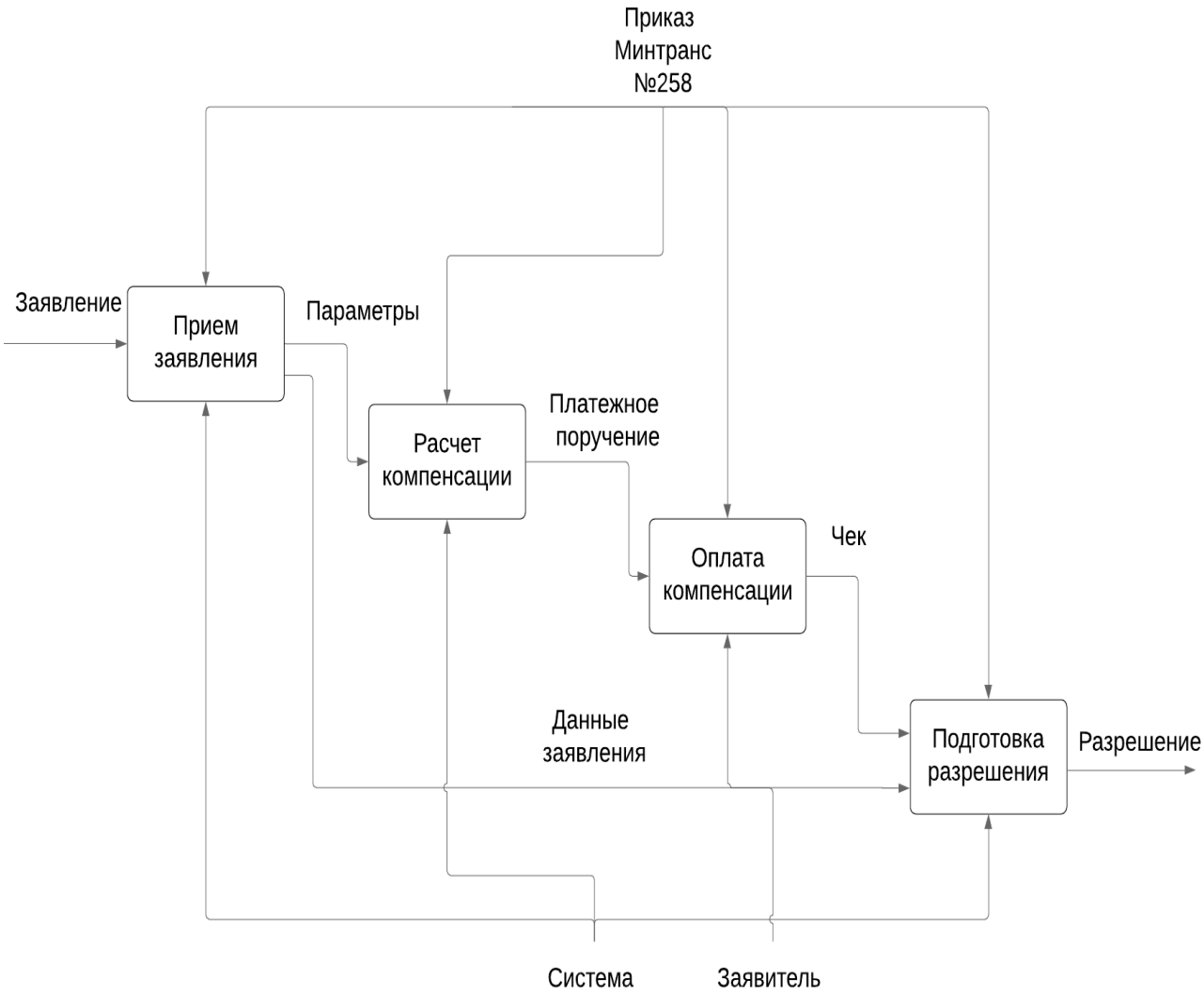

Рисунок 1.5 – модель декомпозиции процесса «Формирование разрешения» «КАК ДОЛЖНО БЫТЬ»

Как видно из схем, улучшение данной системы должно достигаться путем разработки и внедрения автоматизированной системы выдачи разрешений, которая должна объединить в себе функционал уже имеющихся инструментов.

Вывод по главе 1

Проанализировав модель «КАК ЕСТЬ», я выявил, что основной недостаток имеющейся системы заключается в разрозненности инструментов, с помощью которых осуществляется выдача разрешения и принял решение по ее улучшению.

ГЛАВА 2 ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ВЫДАЧИ РАЗРЕШЕНИЙ

2.1 Логическая модель автоматизированной системы и ее описание

В данной диаграмме будут описаны функциональные возможности, которыми должна обладать автоматизированная информационная система.

На рисунке 2.1 представлена диаграмма вариантов использования пользователей автоматизированной системы выдачи разрешений, модель «КАК ДОЛЖНО БЫТЬ».

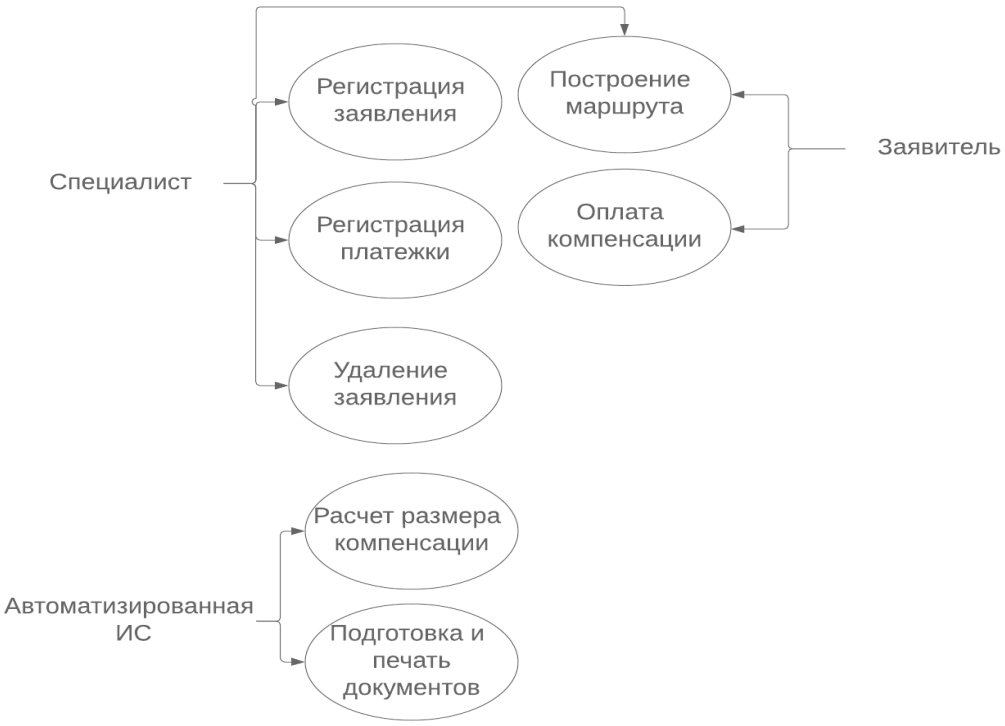

Рисунок 2.1 – Диаграмма вариантов использования, модель «КАК ДОЛЖНО БЫТЬ»

Состав диаграммы.

Актеры:

- Специалист;
- Заявитель

- Автоматизированная система

Заявитель – физическое или юридическое лицо, обращающееся в департамент за разрешением. Доступные ему функции:

- Построение маршрута совместно со специалистом;
- Подтверждение оплаты компенсации.

Специалист – специалист департамента дорожного хозяйства. Имеет функции:

- Регистрация заявления;
- Удаление заявления;
- Регистрация платежки;
- Определение маршрута движения.

Автоматизированная система – система выдачи разрешений. Имеет функции:

- Расчет размера компенсации;
- Подготовка и печать документов на разных этапах процесса выдачи разрешения.

Варианты использования:

Прием заявления – прием заявления в работу и его регистрация в системе.

Расчет компенсации – расчет размера компенсации на основании ттх транспорта и груза, длины пути, категорий дорог и количества поездок. Вывод итоговой суммы на экран.

Подготовка документов – формирование требуемых специалисту документов Word или Excel на основе макетов двоичных данных.

Оплата компенсации – оплата вреда, причиняемого транспортным средством заявителя дорожному полотну.

2.2 Выбор технологии программирования автоматизированной системы выдачи разрешений.

Выбор средств разработки, осуществляется по следующим функциям и критериям:

- Поддержка технологии быстрой разработки приложений RAD (Rapid Application Development), основанной на объектно-ориентированной парадигме программирования;
- возможность полномасштабной поддержки трехзвенной архитектуры «клиент-сервер»;
- простота в изучении и разработке.

Для разработки автоматизированной системы выдачи разрешений будет использоваться среда разработки «1С: Предприятия 8.3», а также язык веб-программирования JavaScript с технологиями Ajax и JQuery. В 1С есть своя база данных, но нам также понадобится выделенный сервер MS SQL.

Система программ «1С: Предприятие 8» включает в себя платформу и прикладные решения, разработанные на ее основе, для автоматизации деятельности организаций и частных лиц. Сама платформа не является программным продуктом для использования конечными пользователями, которые обычно работают с одним из многих прикладных решений (конфигураций), разработанных на данной платформе. Такой подход позволяет автоматизировать различные виды деятельности, используя единую технологическую платформу.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

AJAX — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

Исходные данные:

1. Технические требования:

- количество одновременно работающих пользователей – без ограничений;
- режим работы (9/5/365);
- восстанавливаемость после сбоя;
- требований к территориальному положению нет.

2. Архитектурные требования 1С:

- Двухзвенная клиент-серверная архитектура;
- Требований к сетевым протоколам нет;

3. Архитектурные требования к веб-модулю:

- Языки – JavaScript, C#;
- Использование библиотек ESRI JS API, JQuery;
- Поддержка технологии Ajax;
- Сервер веб-приложений IIS;
- Трехзвенная клиент-серверная архитектура;
- Требований к сетевым протоколам нет.

В качестве средств разработки системы были выбраны «1С: Предприятие» и веб-приложение, как полностью отвечающие нашим требованиям.

2.3 Проектирование базы данных системы выдачи разрешений

На рисунке 2.2 изображена схема базы данных модуля «1С: Предприятие».

Центральная таблица – основной рабочий документ «Заявления». В нем находятся поля со ссылками на справочники и документ «Разрешения», а также текстовые поля параметров.

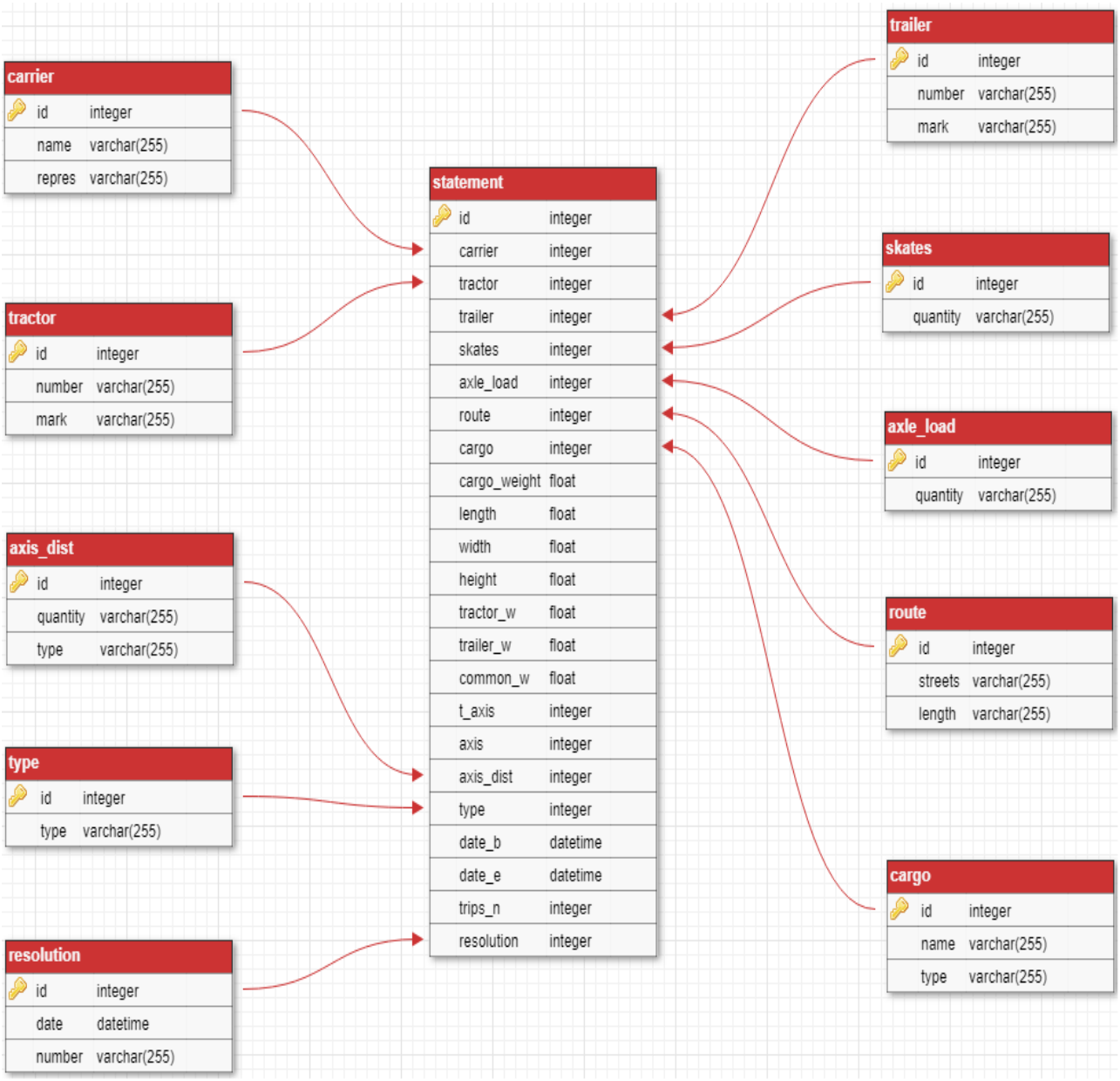

Рисунок 2.2 – база данных 1С

Выводы по главе 2

Во второй главе балы спроектирована и описана логическая модель автоматизированной системы, выбраны технологии для реализации и спроектирована база данных клиентского модуля 1С автоматизированной системы.

ГЛАВА 3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ВЫДАЧИ РАЗРЕШЕНИЙ

3.1 Выбор архитектуры системы выдачи разрешений

Автоматизированная система будет состоять из 2х модулей:

- Клиентский модуль регистрации заявлений «1С: Предприятие»;
- Картографический модуль веб-ГИС.

На рисунке 3.1 изображена схема взаимодействия модулей.

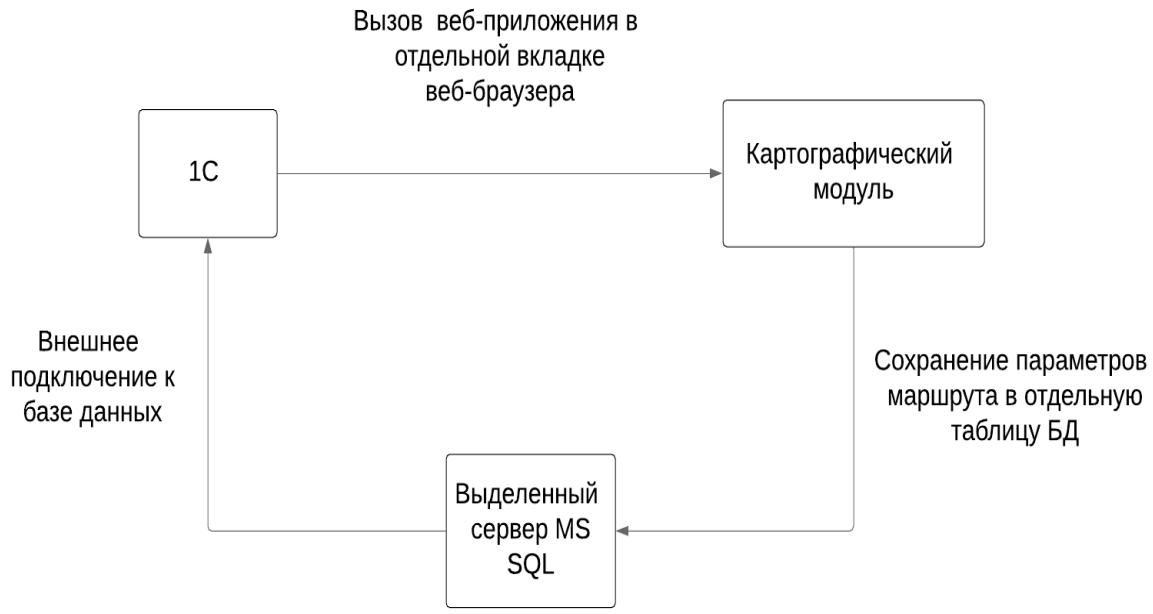

Рисунок 3.1 – схема взаимодействия модулей системы

Согласно ей, специалист нажатием на кнопку «Маршрут» в клиентском модуле 1С вызывает веб-ГИС приложение в отдельном окне браузера. В нем строится маршрут и сохраняется в отдельную таблицу на выделенном сервере БД MS SQL. К этой таблице, с помощью внешнего SQL подключения, подключен модуль 1С. Нажатием на вторую кнопку специалист загружает данные сохраненного маршрута в заявление для расчета размера компенсации. Система также сохраняет загруженный маршрут в справочник «Улицы».

Клиентский модуль регистрации заявлений будет реализован на базе «1С: Предприятие». Он будет иметь двухзвенную архитектуру, как показано на рисунке 3.2.

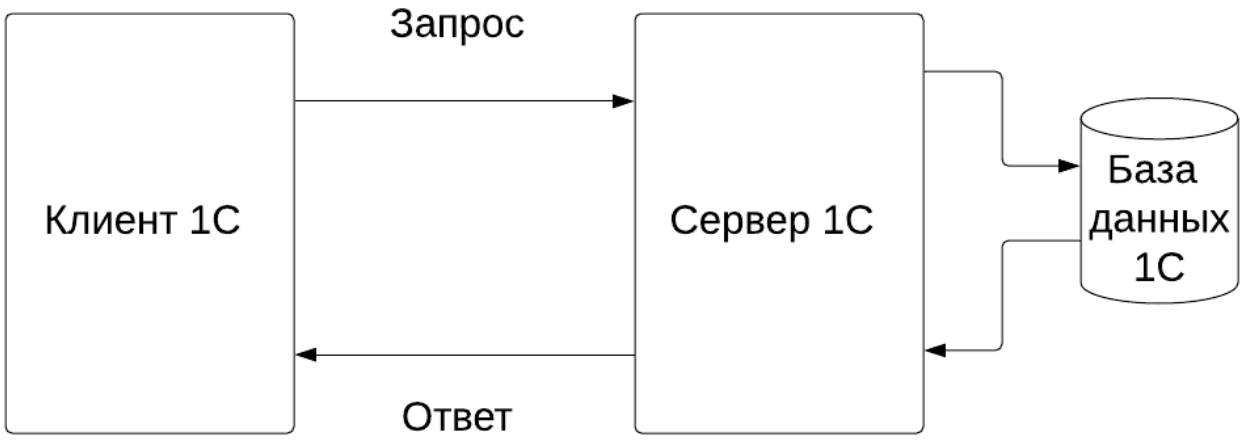

Рисунок 3.2 – двухзвенная архитектура модуля 1С

Картографический модуль маршрутизации будет иметь трехзвенную клиент-серверную архитектуру, как показано на рисунке 3.3.

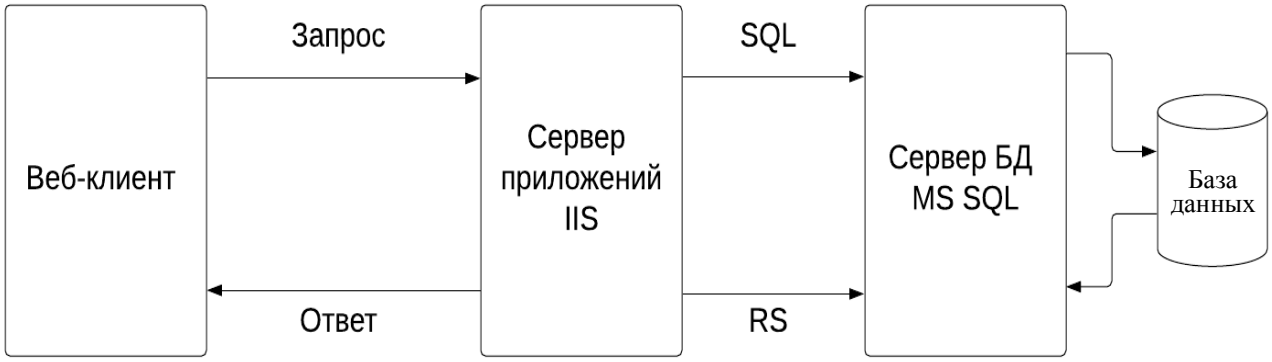

Рисунок 3.3 – двухзвенная архитектура модуля 1С

Таким образом, в системе будет сразу две архитектуры, задействованных в разных модулях. Архитектура картографического модуля позволит получать к нему доступ любому желающему, что открывает широкие возможности по его доработке под разные нужды и пользователей. В свою очередь, архитектура модуля 1С дает гарантии сохранности базы данных, так-как доступ к ней будут иметь только авторизованные пользователи на локальных машинах.

3.2 Выбор системы управления базами данных системы выдачи разрешений

В среду разработки «1С: Предприятие 8.3» интегрирована СУБД собственной разработки. Исходя из этого, осталось выбрать СУБД для картографического модуля.

База данных на выделенном сервере будет использоваться для хранения последнего построенного маршрута, для передачи в модуль 1С. Эта записать будет занимать очень мало места, из чего следует, что стартовый размер БД должен быть маленьким. Однако, в перспективе может быть добавлена вторая таблица, для хранения всех построенных маршрутов и их загрузки как в модуль 1С, там и на клиент картографического модуля. Значит, СУБД должна уметь изменять размер БД динамически.

Важным фактором при выборе СУБД является поддержка проприетарных загрузчика геоданных, просмотрщика и редактора ESRI ArcGIS, т.к. именно это ПО безальтернативно было выбрано для создания картографического сервиса маршрутизации. Также, должна быть поддержка геодезии, для возможных вычислений в сферических координатах.

Требования к СУБД:

- Маленький стартовый размер;
- Возможность расширения;
- Поддержка проприетарных инструментов картографии;
- Поддержка геодезии.

На основе этих требований была выбрана СУБД MS SQL.

3.3 Разработка модуля «1С: Предприятие»

Разработка началась с создания основных структурных единиц.

Был создан основной рабочий документ «Заявления», в соответствии с рисунками 3.4 и 3.5, с реквизитами, соответствующими всем реквизитам, подаваемым заявителем. К ним добавляются 3 табличные части:

- «Маршрут» - для перечисления участков маршрута с разными категориями дорог и длины пути.
- «Платежки – вред» - для перечисления платежек, которыми оплачена компенсация вреда.
- «Платежки – госпошлина» - для перечисления платежек, которыми была оплачена госпошлина.

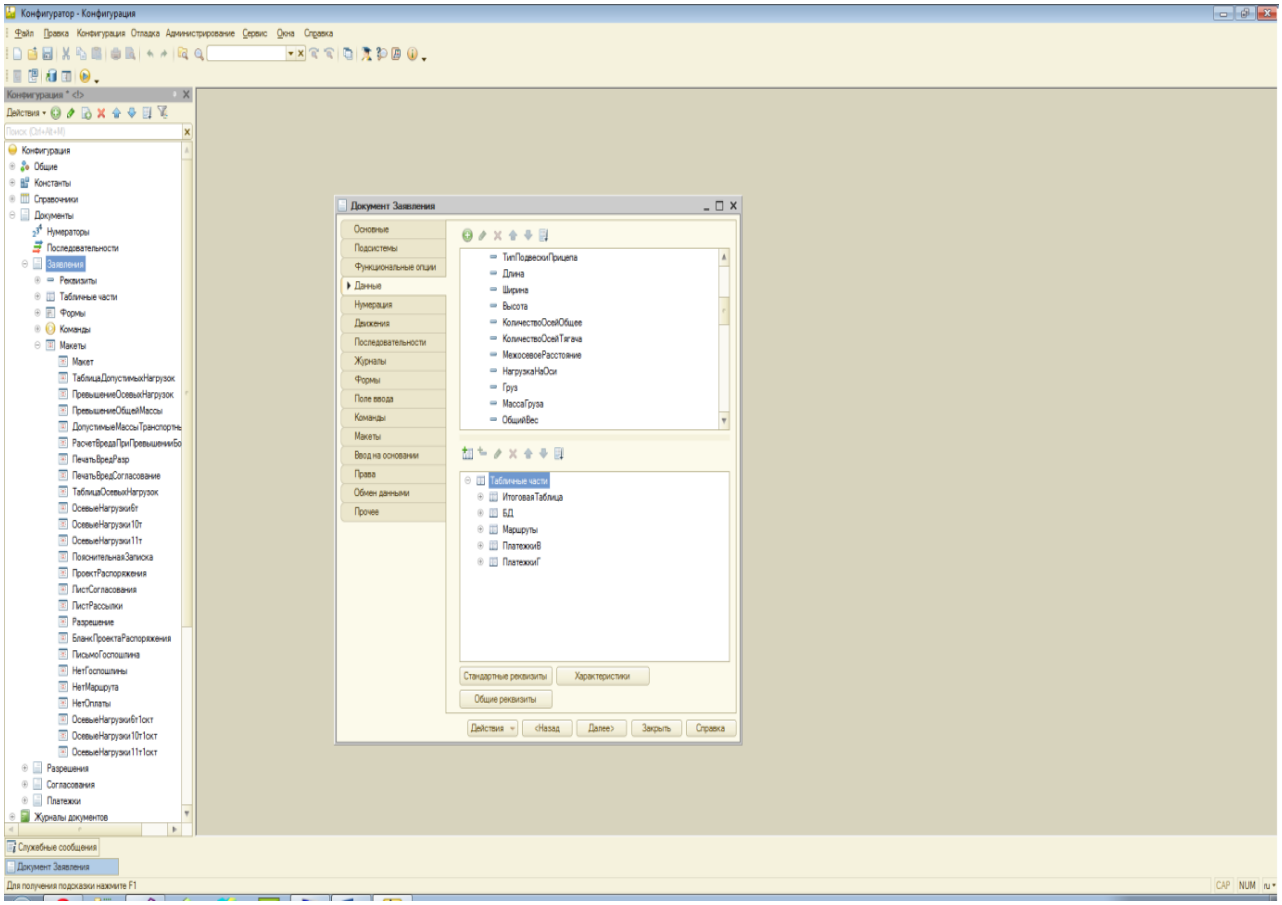

Рисунок 3.4 – Создание документа «Заявления»

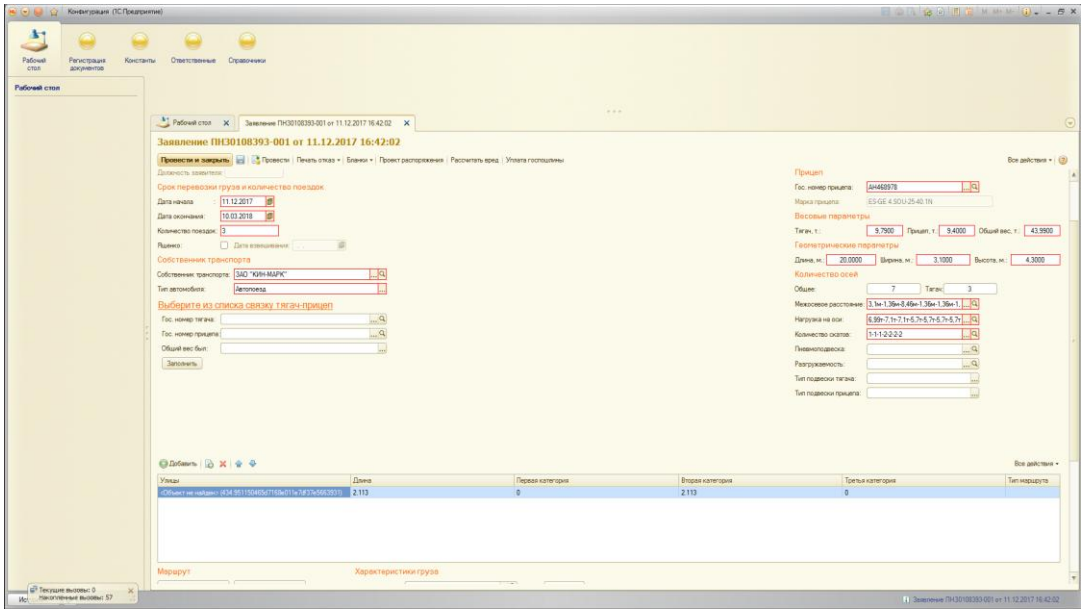

Рисунок 3.5 – Документ «Заявление»

Тем же способом созданы документы «Платежки», «Разрешения» и «Согласования».

Далее были созданы справочники, отвечающие за хранение всей периодической информации, необходимой для подачи заявления, в соответствии с рисунками 3.6 и 3.7.

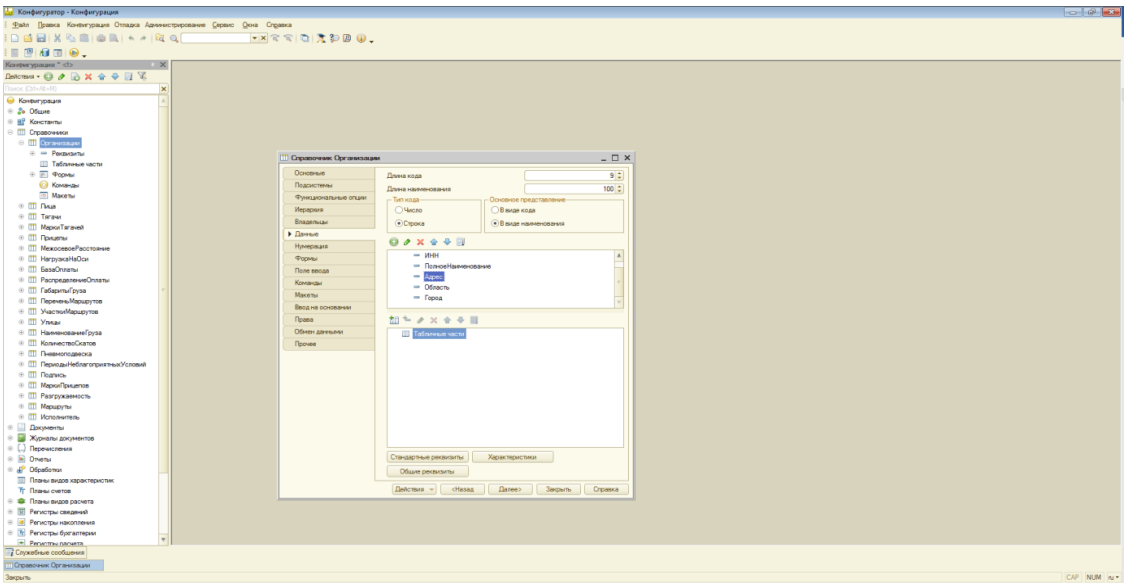

Рисунок 3.6 – Создание справочника «Организации»

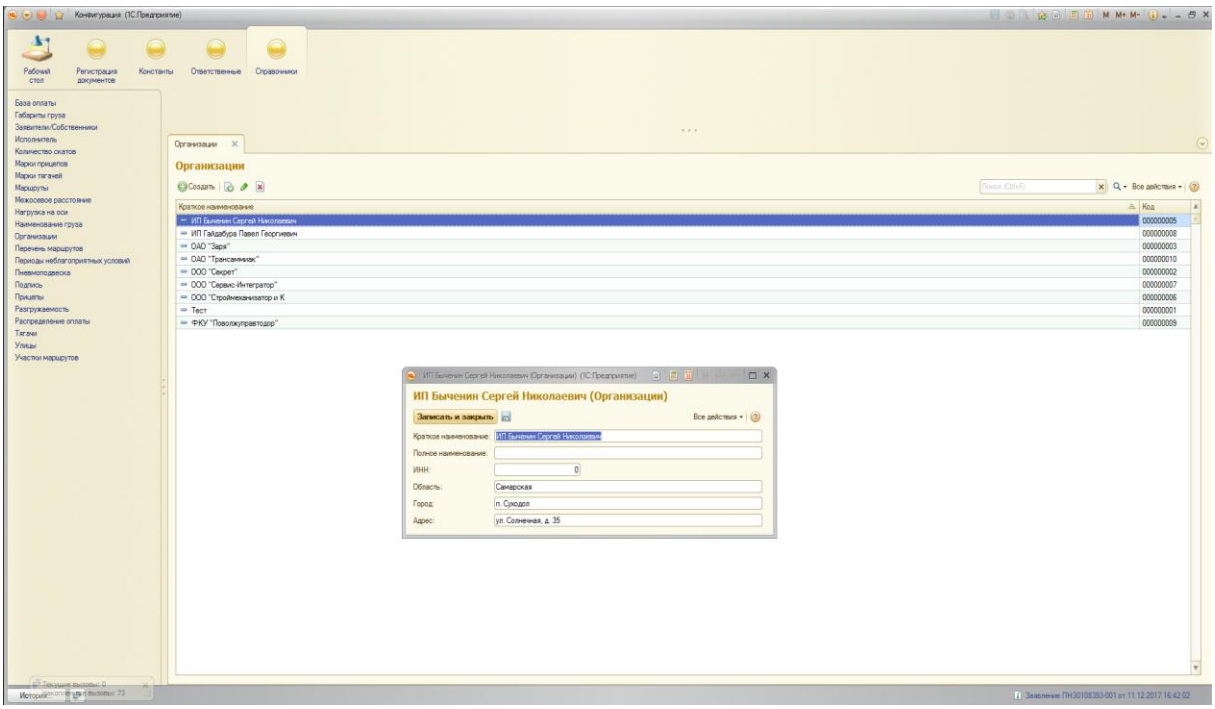

Рисунок 3.7 – Справочник «Организации»

В документ «Заявления» добавляется произвольная форма, на которую нанесены необходимые для получения разрешения реквизиты, со ссылками на возможные документы и справочники, в соответствии с рисунком 3.8.

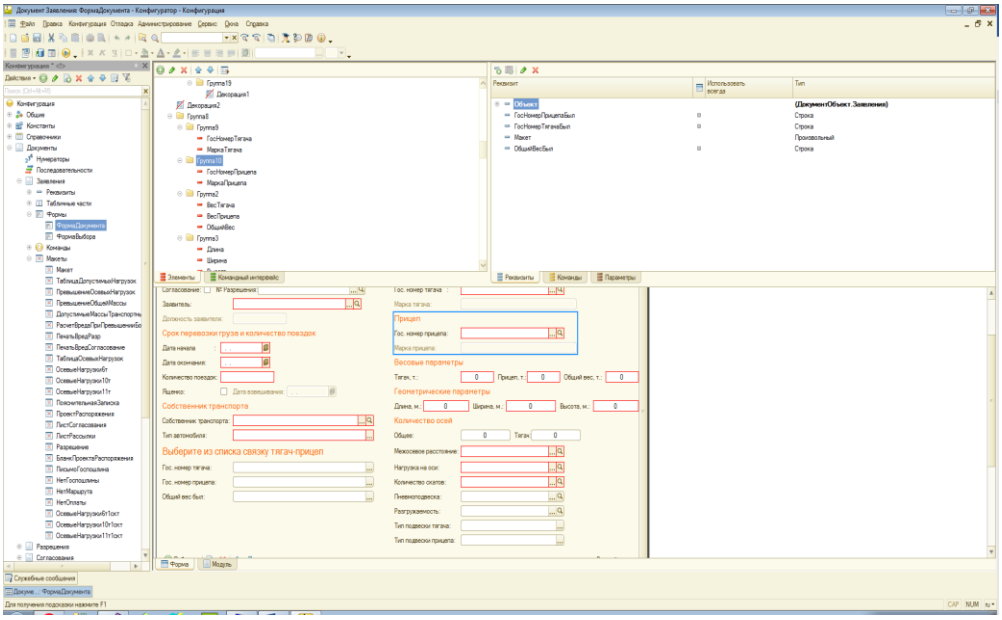

Рисунок 3.8 – Создание формы документа «Заявления»

Также созданы элементы управления формы (кнопки) и соответствующие им команды, в соответствии с рисунками 3.9 и 3.10.

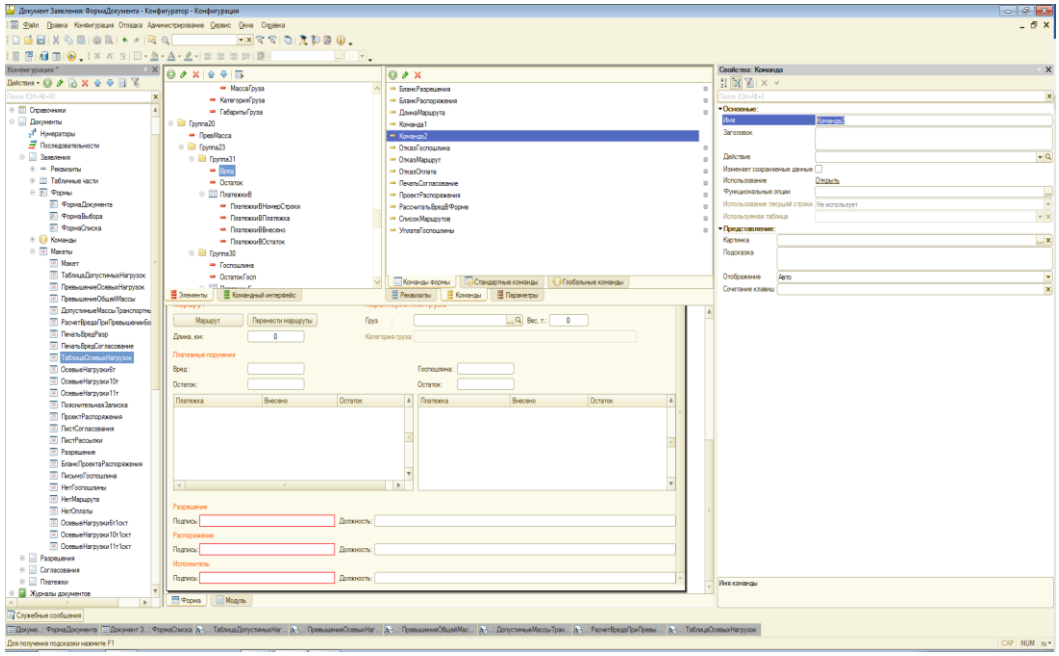

Рисунок 3.9 – Создание команд

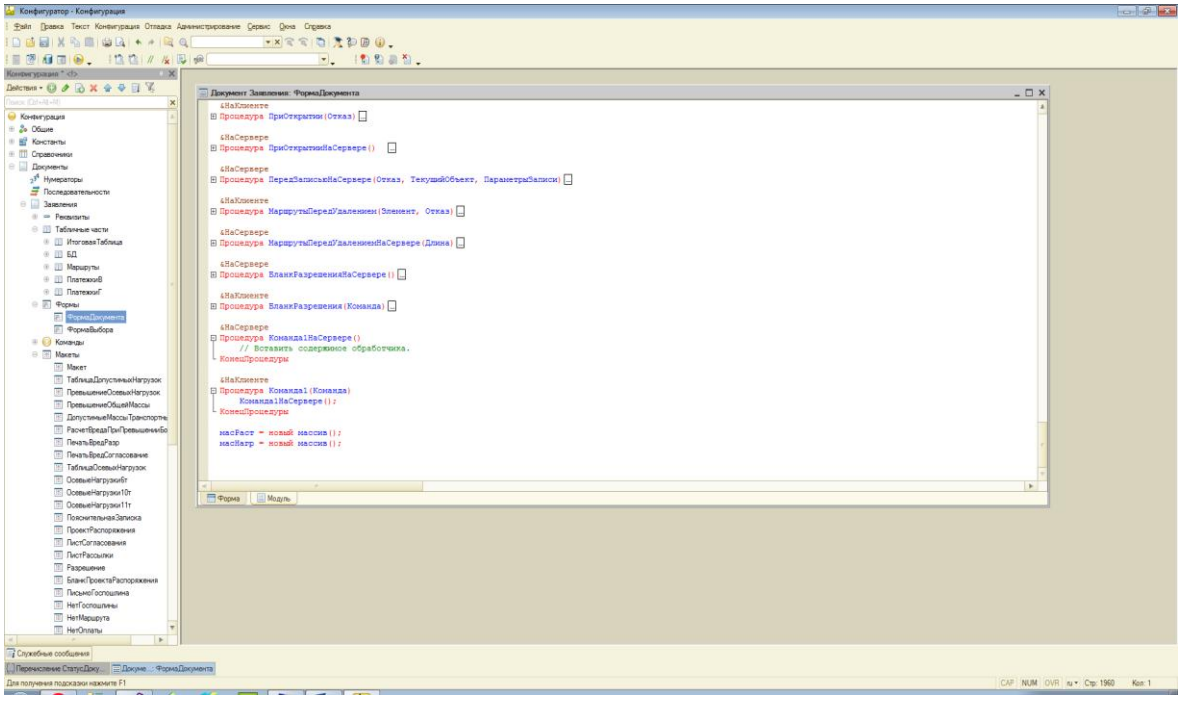

Рисунок 3.10 – Код команд

Команды отвечают за печать документов. Сами документы будут печататься из шаблонов двоичных данных. Один документ – сводная таблица расчетов вреда, выводится в формате .xls из макета табличного документа, в соответствии с рисунками 3.11 и 3.12.

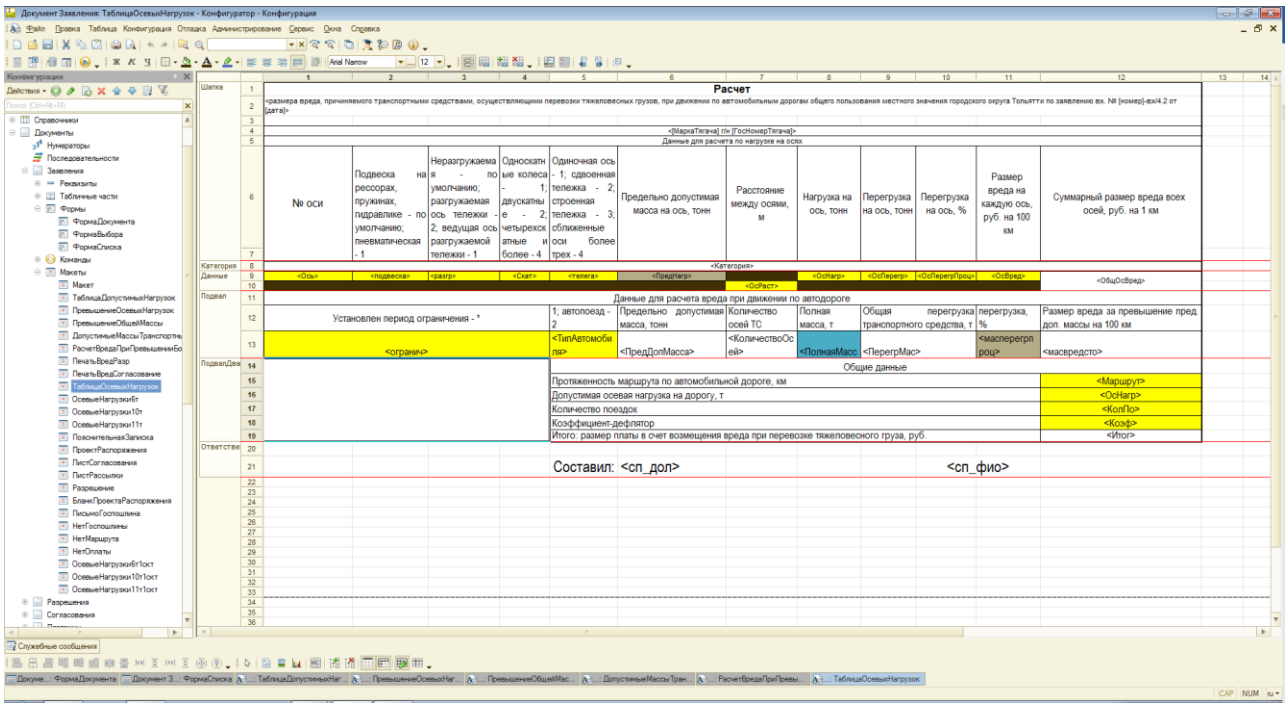

Рисунок 3.11 – Создание табличного документа

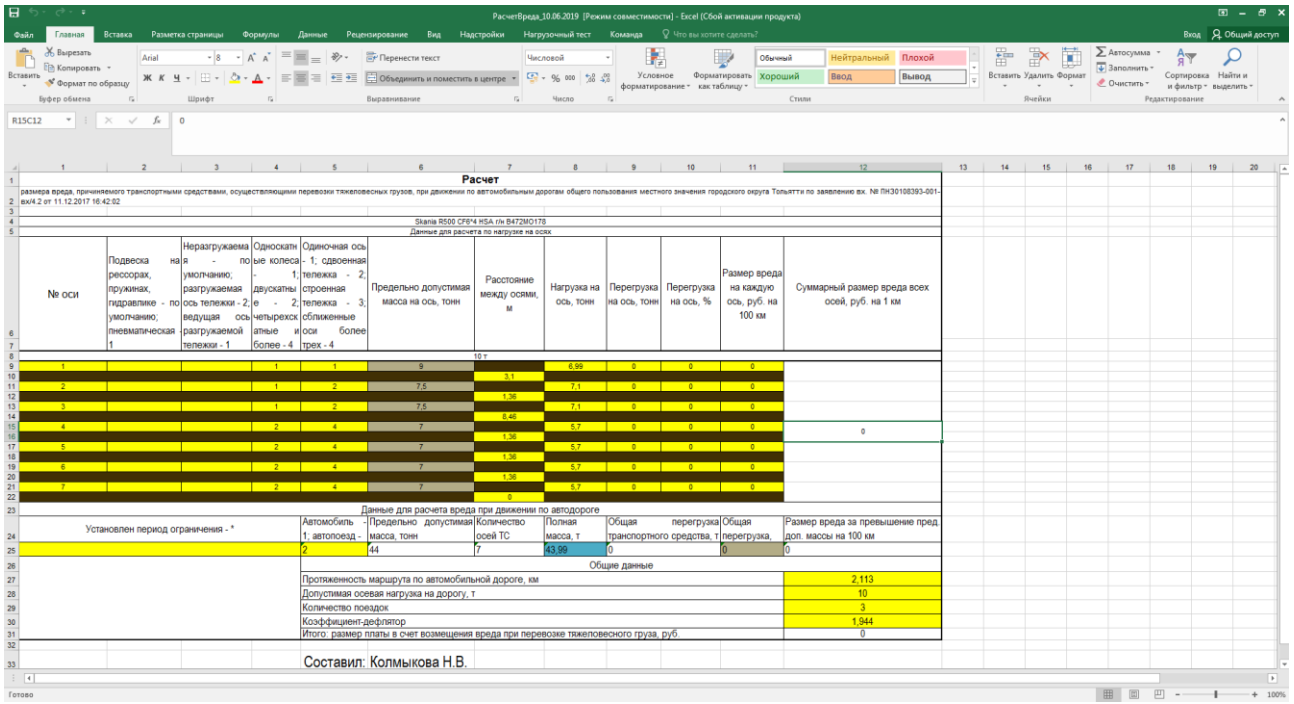

Рисунок 3.12 – Сводная таблица

В модуле объекта формы документа «Заявления» написан код взаимодействия реквизитов и расчета компенсации вреда, в соответствии с рисунком 3.13.

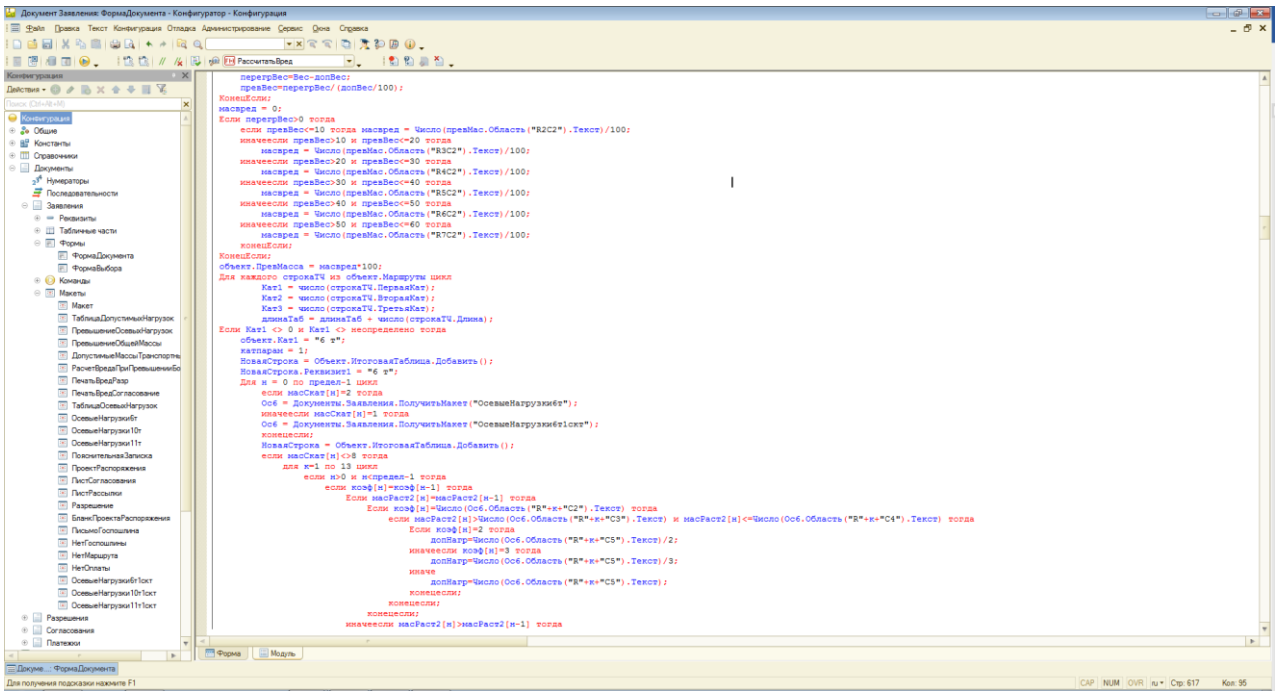

Рисунок 3.13 – Программный код расчета вреда

Для удобства взаимодействия пользователя и программы, добавлены перечисления и константы, в соответствии с рисунками 3.14 и 3.15.

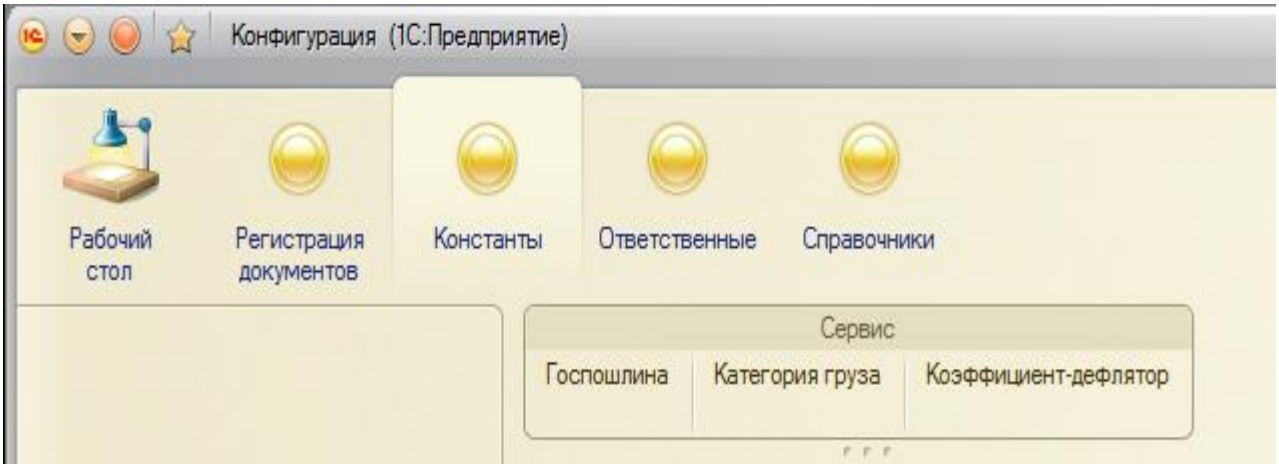

Рисунок 3.14 – Меню констант

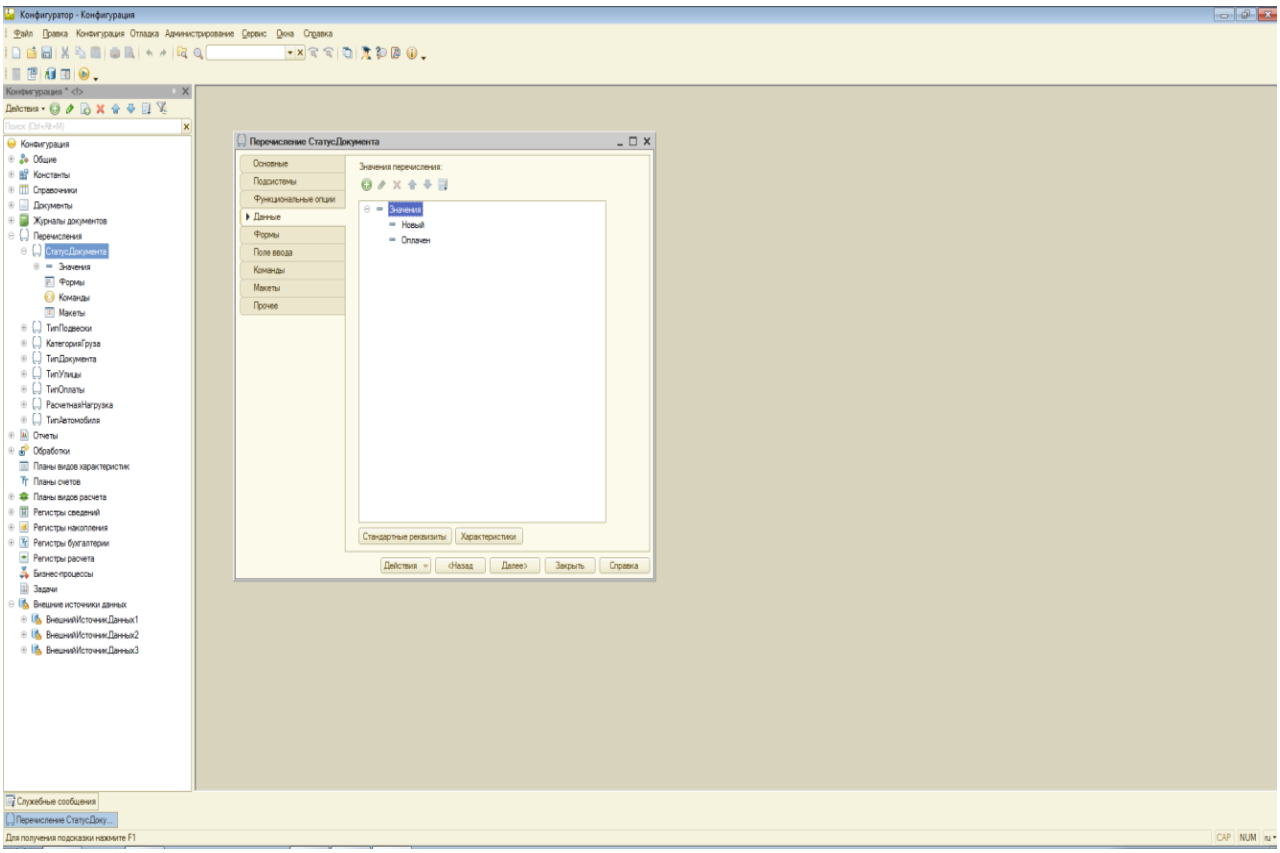

Рисунок 3.15 – Создание перечисления «Статус документа»

Для размещения структурных элементов созданы подсистемы, в соответствии с рисунками 3.16 и 3.17:

- Выдача разрешений
- Справочники
- Ответственные
- Константы

В первой подсистеме расположены документы, регистры, отчет и обработки.

Во второй – все справочники.

В третьей – константы ответственных и специалиста.

В четвертой – константы госпошлины, категории груза и коэффициента-дефлятора.

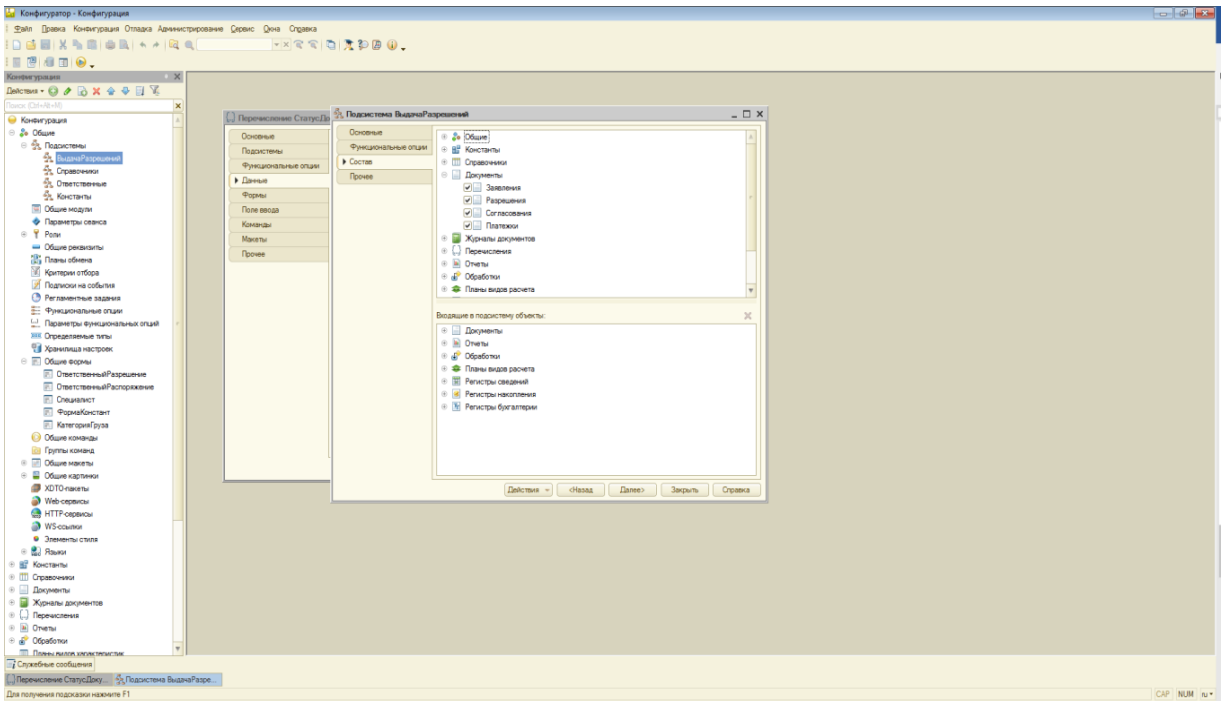

Рисунок 3.16 – Создание подсистемы «Выдача разрешений»

Для проведения оплаты создана обработка «Оплата». На форму обработки добавлены реквизиты поиска заявлений, табличная часть для вывода результатов поиска, окна выбора платежек по типам. Написан программный код поиска заявлений по входным параметрам, вывода результатов в таблицу, сохранения платежных поручений и их привязки к заявлению, в соответствии с рисунками 3.18 и 3.19.

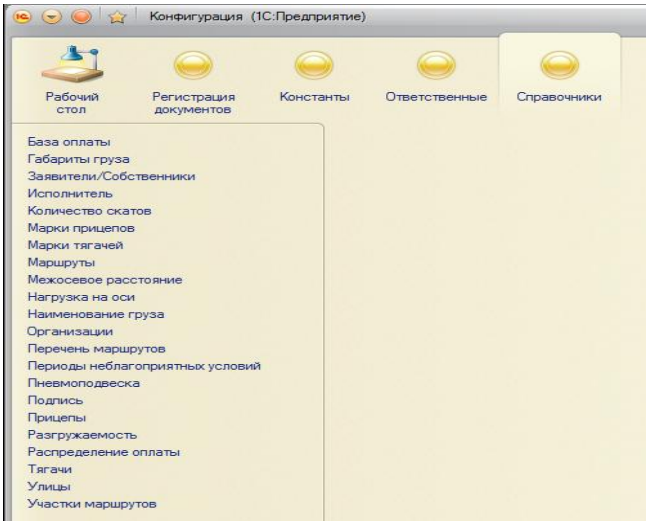

Рисунок 3.17 – Подсистемы конфигурации

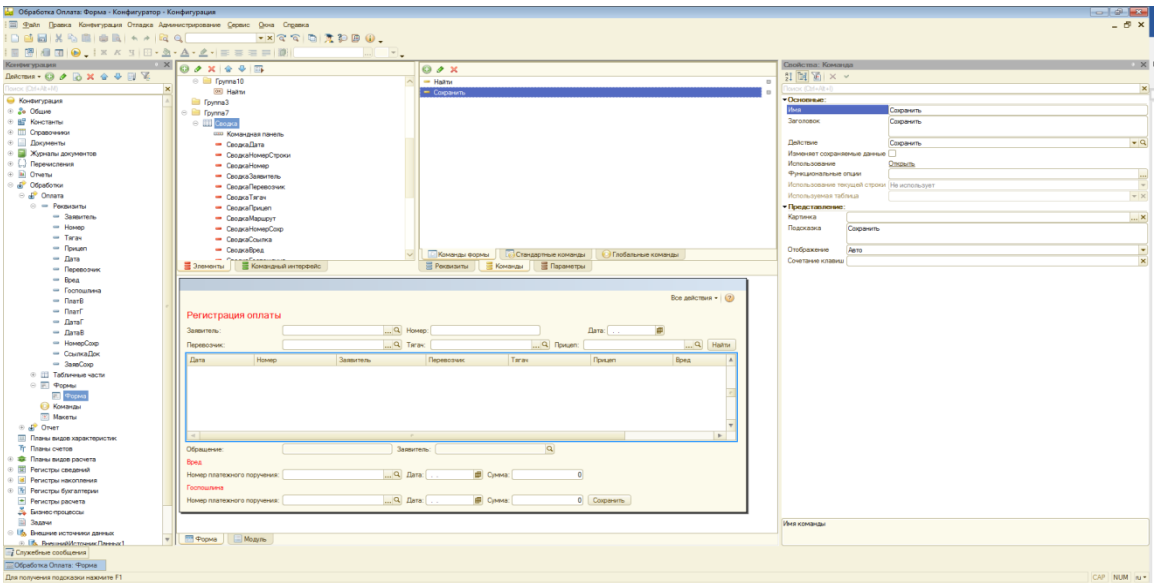

Рисунок 3.18 – Создание обработки «Оплата»

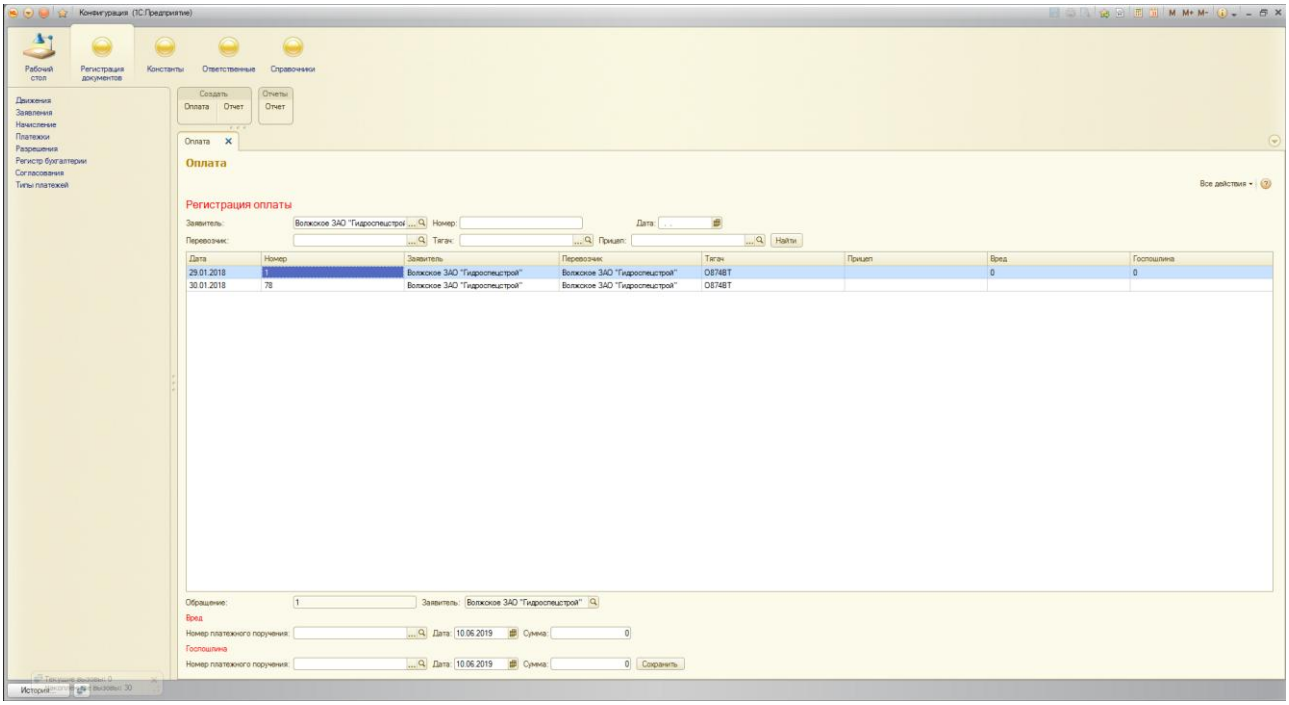

Рисунок 3.19 – Обработка «Оплата»

Для создания отчета по форме заказчика добавлена обработка «Отчет». В ней создана форма обработки, на нее добавлены 2 табличные части для вывода результатов, переключатель типа выдаваемого документа, поля выбора дат начала и окончания выборки, кнопки поиска и печати отчета, в соответствии с рисунками 3.20 и 3.21.

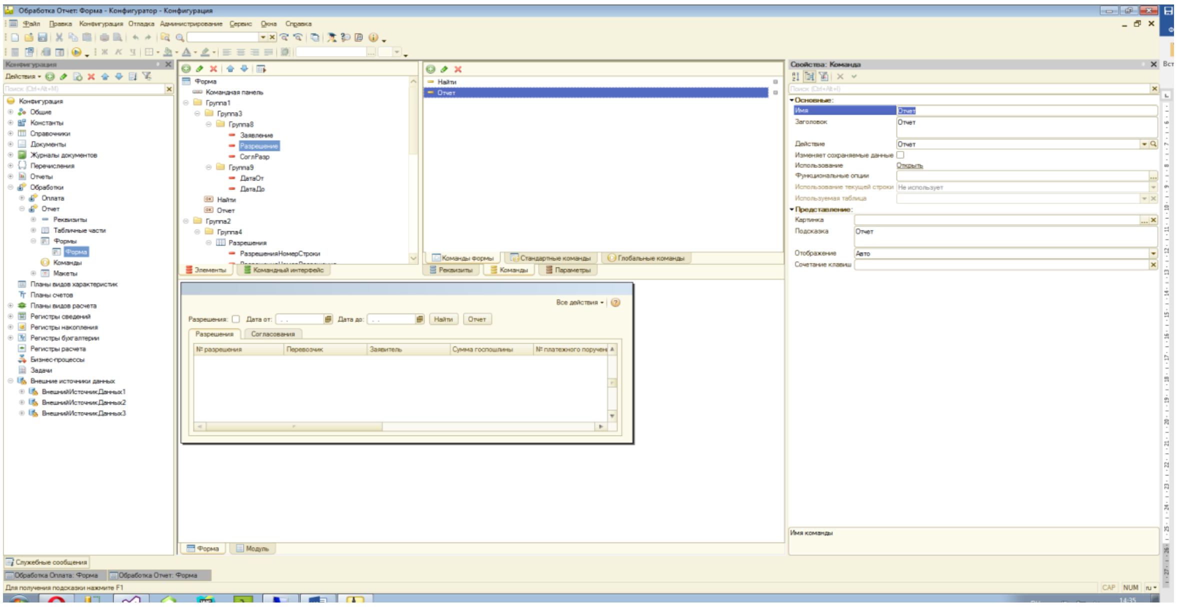

Рисунок 3.20 – Создание обработки «Отчет»

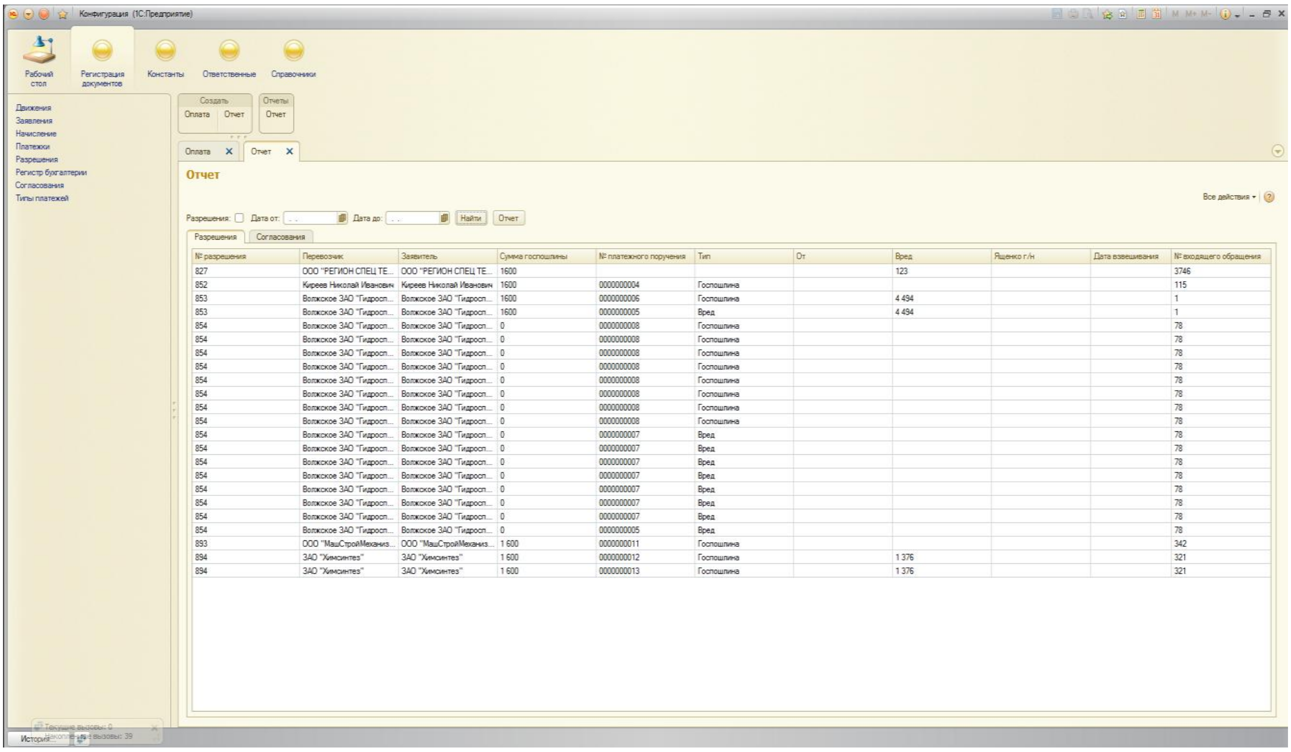

Рисунок 3.21 – Обработка «Отчет»

Для отслеживания поступления заявлений, принятых к исполнению, создан регистр сведений. Для вывода регистра на рабочий стол добавляется форма регистра, в соответствии с рисунками 3.22 и 3.23.

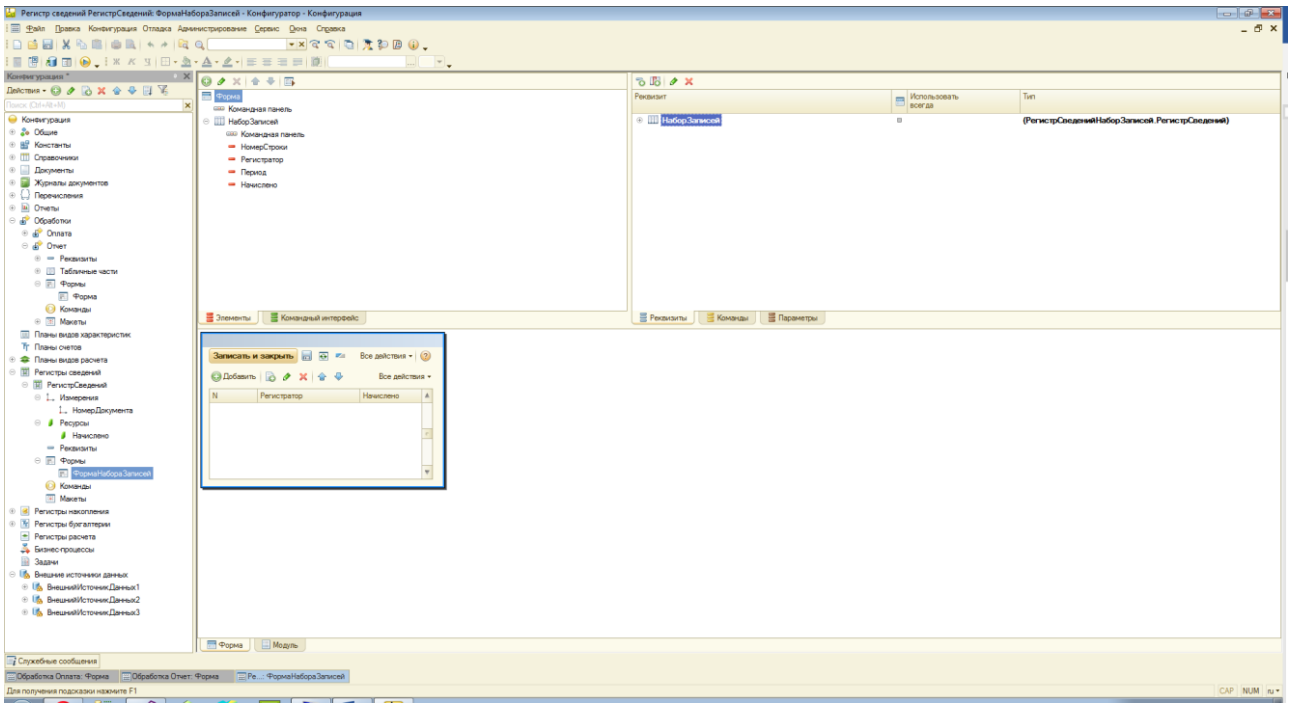

Рисунок 3.22 – Создание регистра сведений

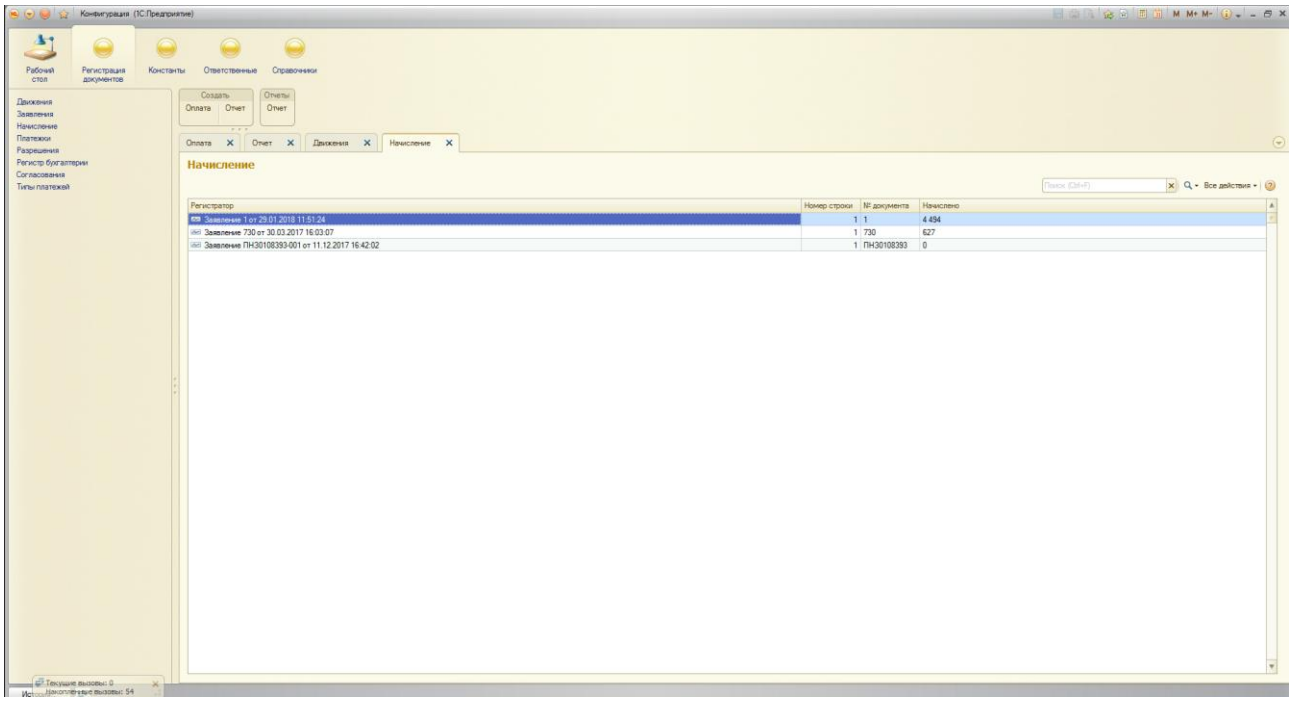

Рисунок 3.23 – Регистр сведений «Начисления»

Также, для отслеживания оплат заявлений, принятых к исполнению, создан регистр накоплений, в соответствии с рисунками 3.24 и 3.25.

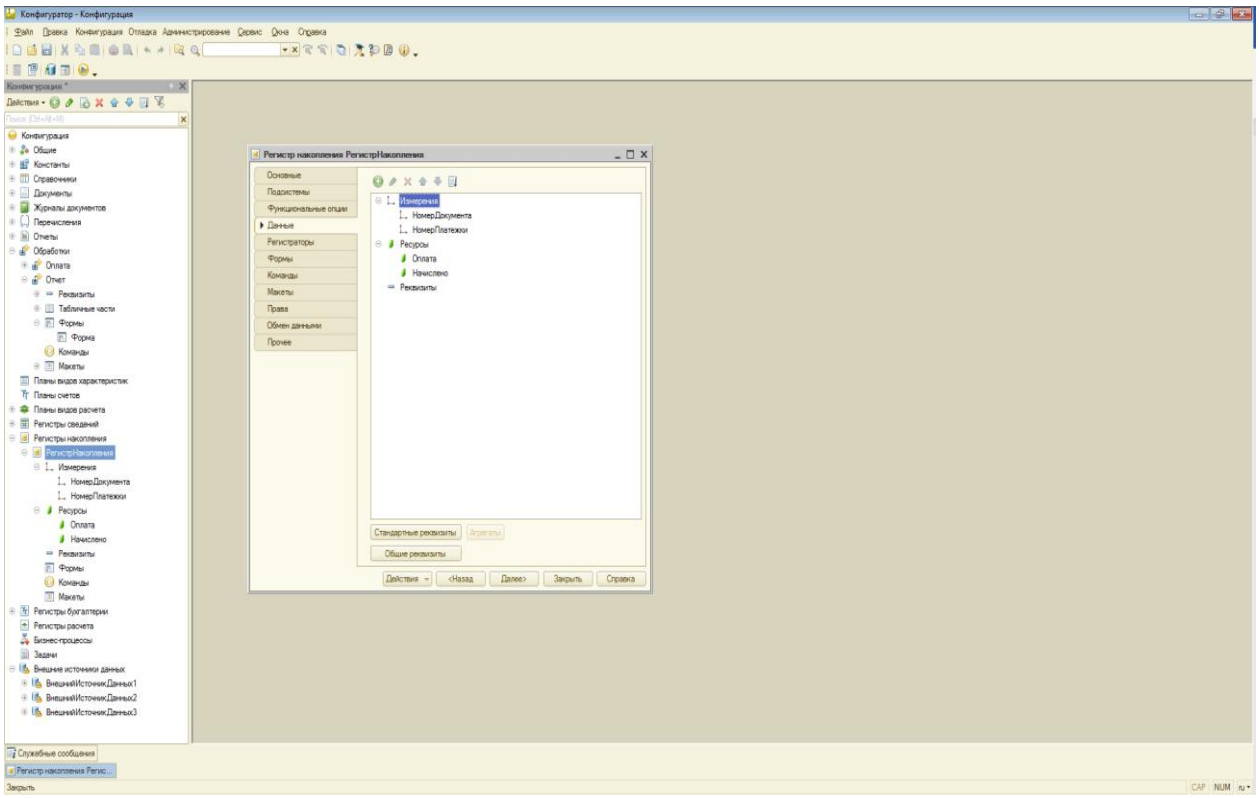

Рисунок 3.24 – Создание регистра накоплений

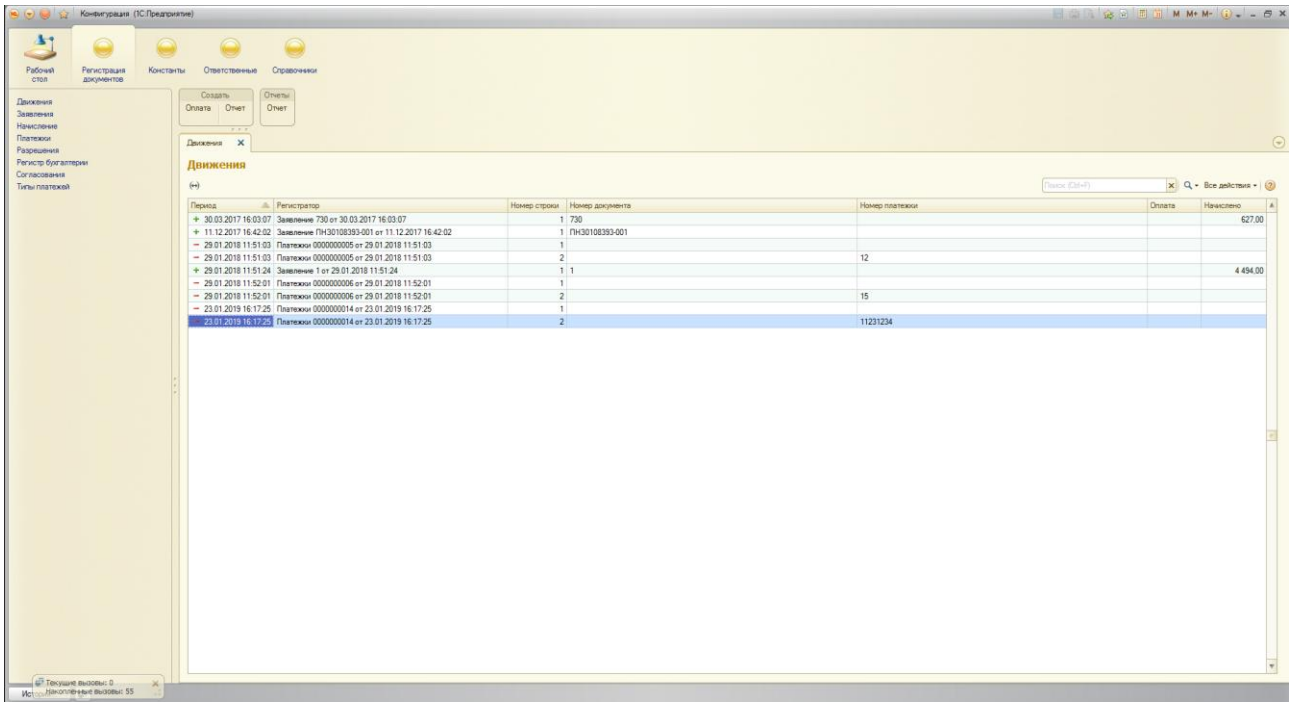

Рисунок 3.25 – Регистр накоплений «Движения»

Также создан структурный элемент «Отчет», в соответствии с рисунками 3.26 и 3.27.

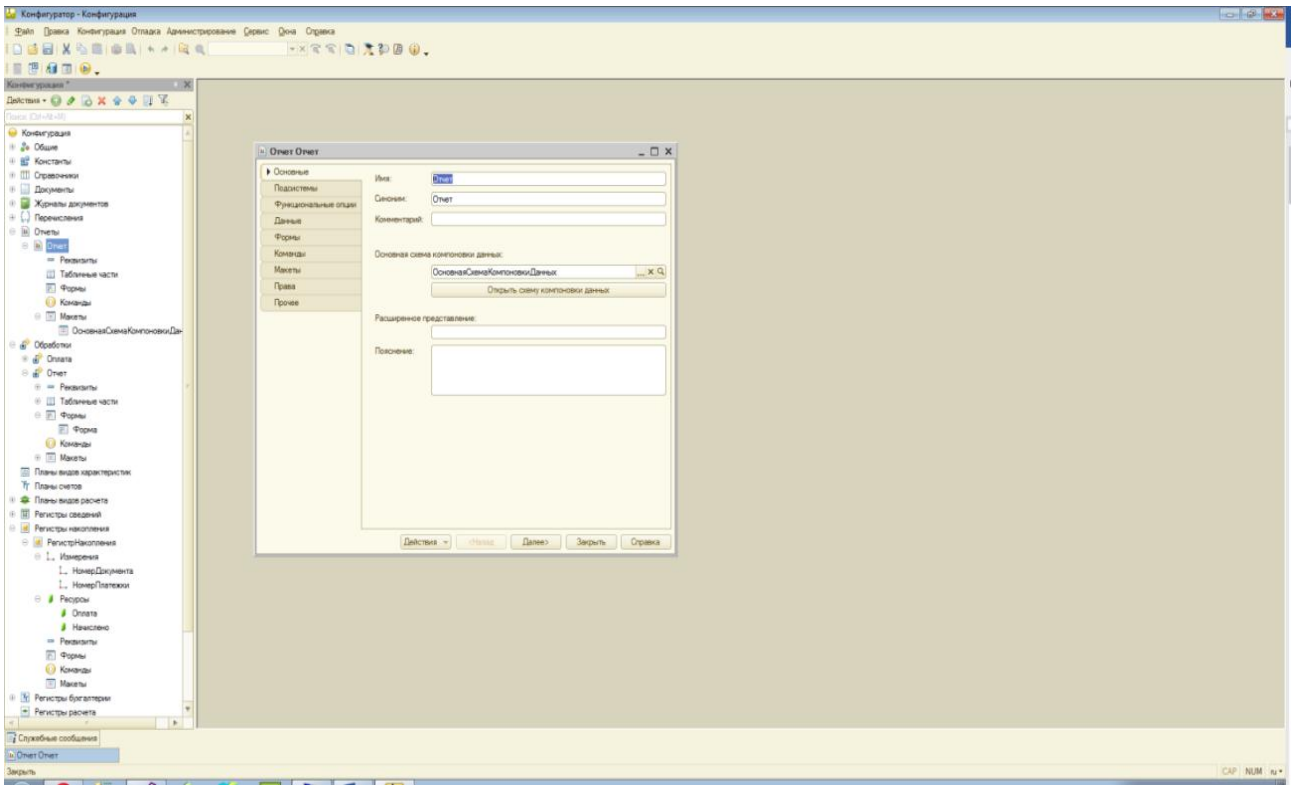

Рисунок 3.26 – Создание отчета

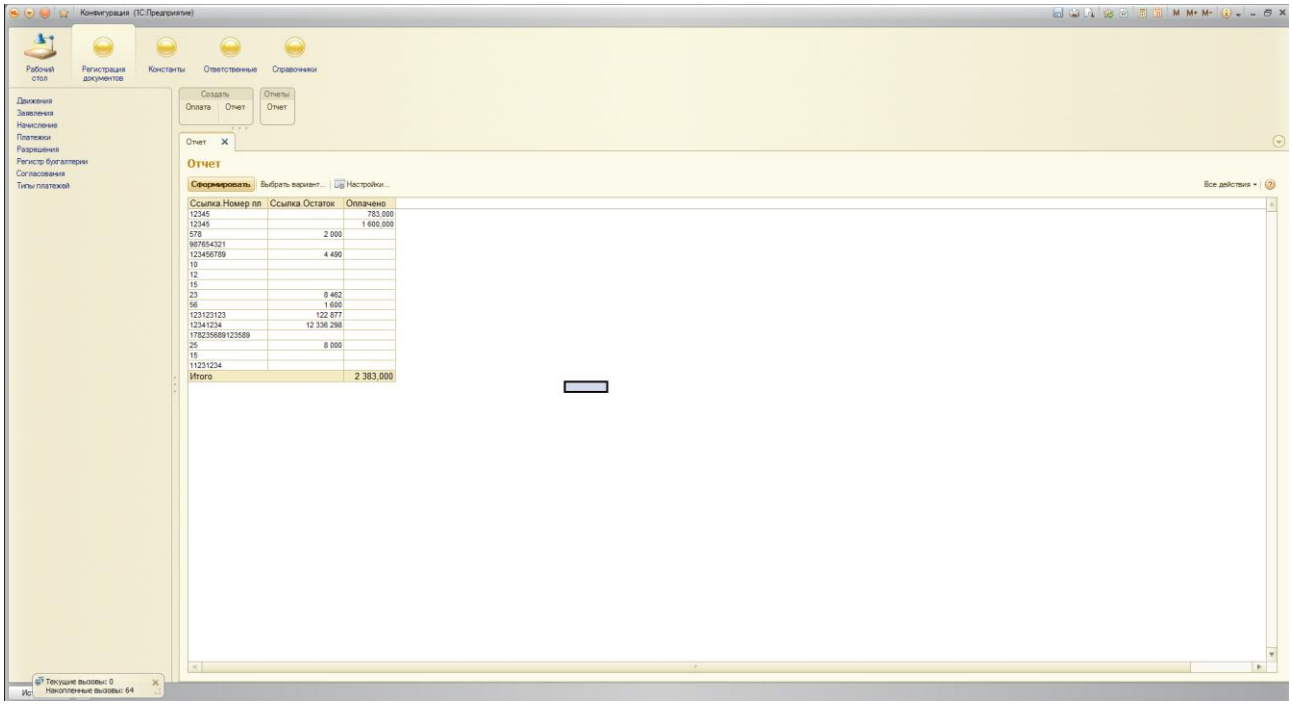

Рисунок 3.27 – Отчет

В отчете создан макет «Основная схема компоновки данных», в соответствии с рисунком 3.28.

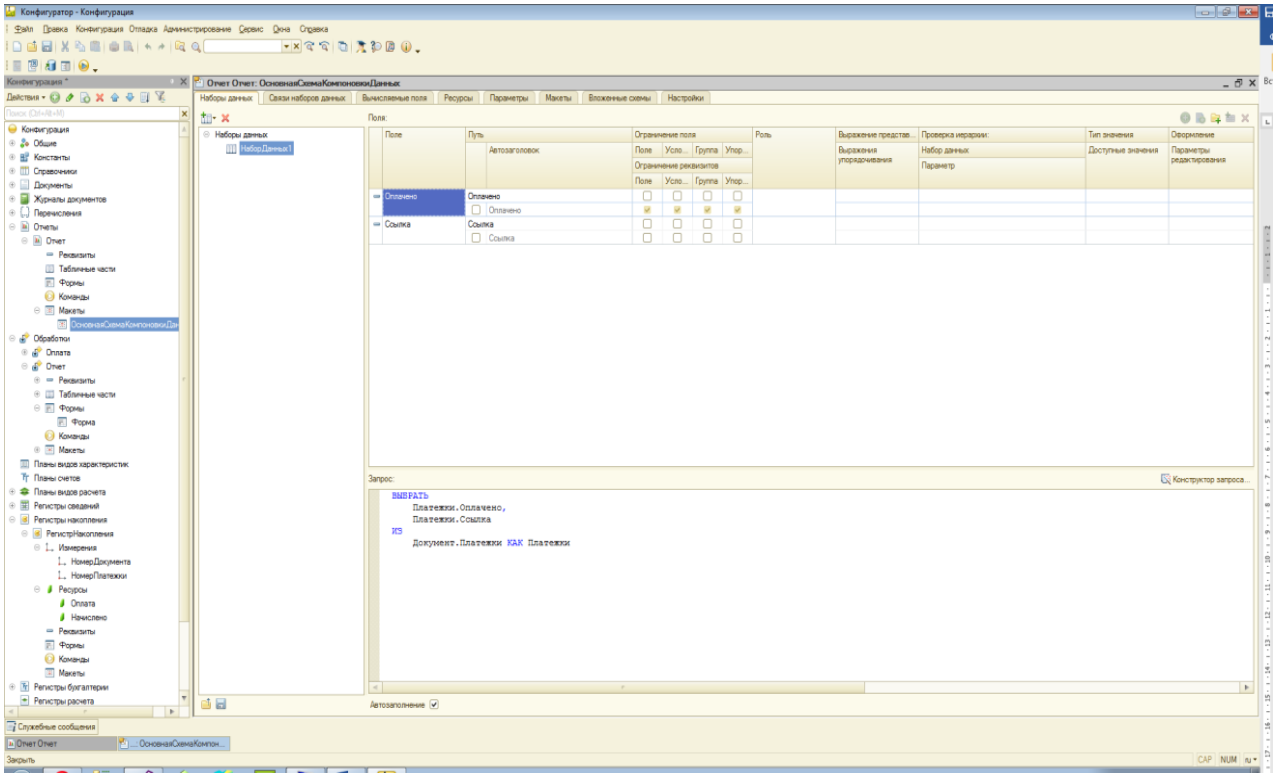

Рисунок 3.28 – Создание макета

Через «Конструктор запроса» добавлены поля, на основе которых будет формироваться отчет, в соответствии с рисунком 3.29.

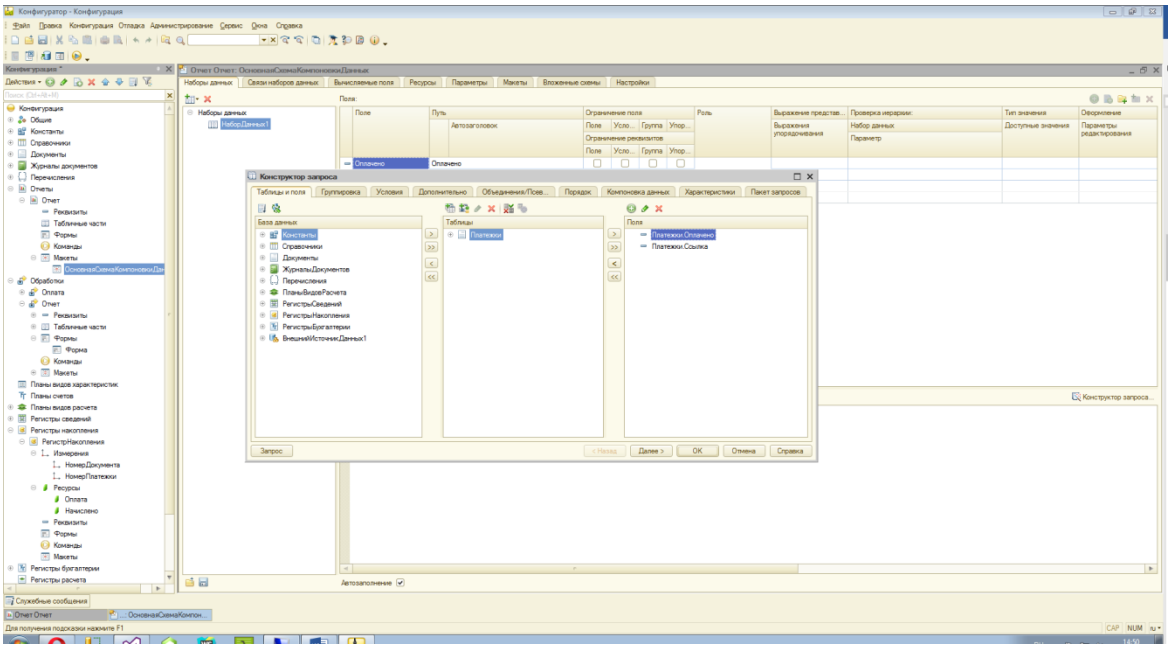

Рисунок 3.29 – Добавление полей

Во вкладке «Ресурсы» выбраны поля, по которым будут проводиться расчеты, в соответствии с рисунком 3.30.

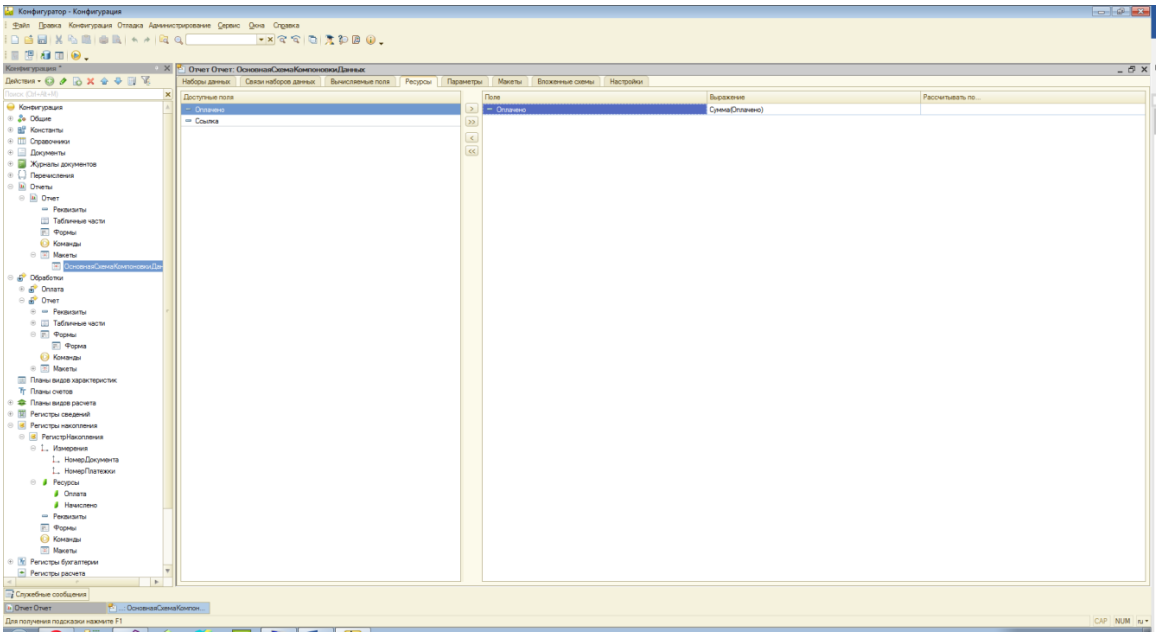

Рисунок 3.30 – Добавление ресурсов

На вкладке «Настройки» выбраны поля, которые будут выводиться в отчет, в соответствии с рисунком 3.31:

- Номер платежного поручения;
- Остаток средств на платежном поручении;
- Средства, списанные в оплату.

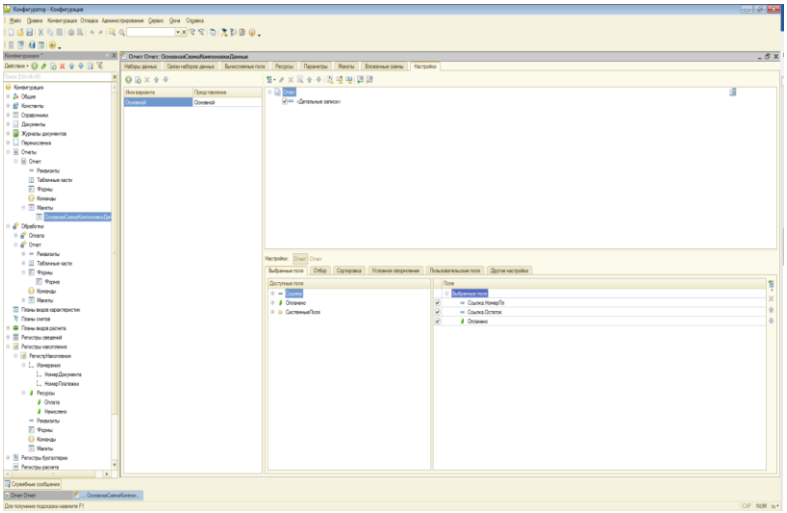

Рисунок 3.31 – Выбор итоговых полей

Также в программе имеется внешнее соединение с базой данных MS SQL, в которой хранятся данные построенных маршрутов, в соответствии с рисунком 3.32.

Для возможности работы разных специалистов добавлена система авторизации с ролями, в соответствии с рисунком 3.33 и созданы 2 пользователя: администратор и специалист, в соответствии с рисунком 3.34. Окно авторизации на рисунке 3.35.

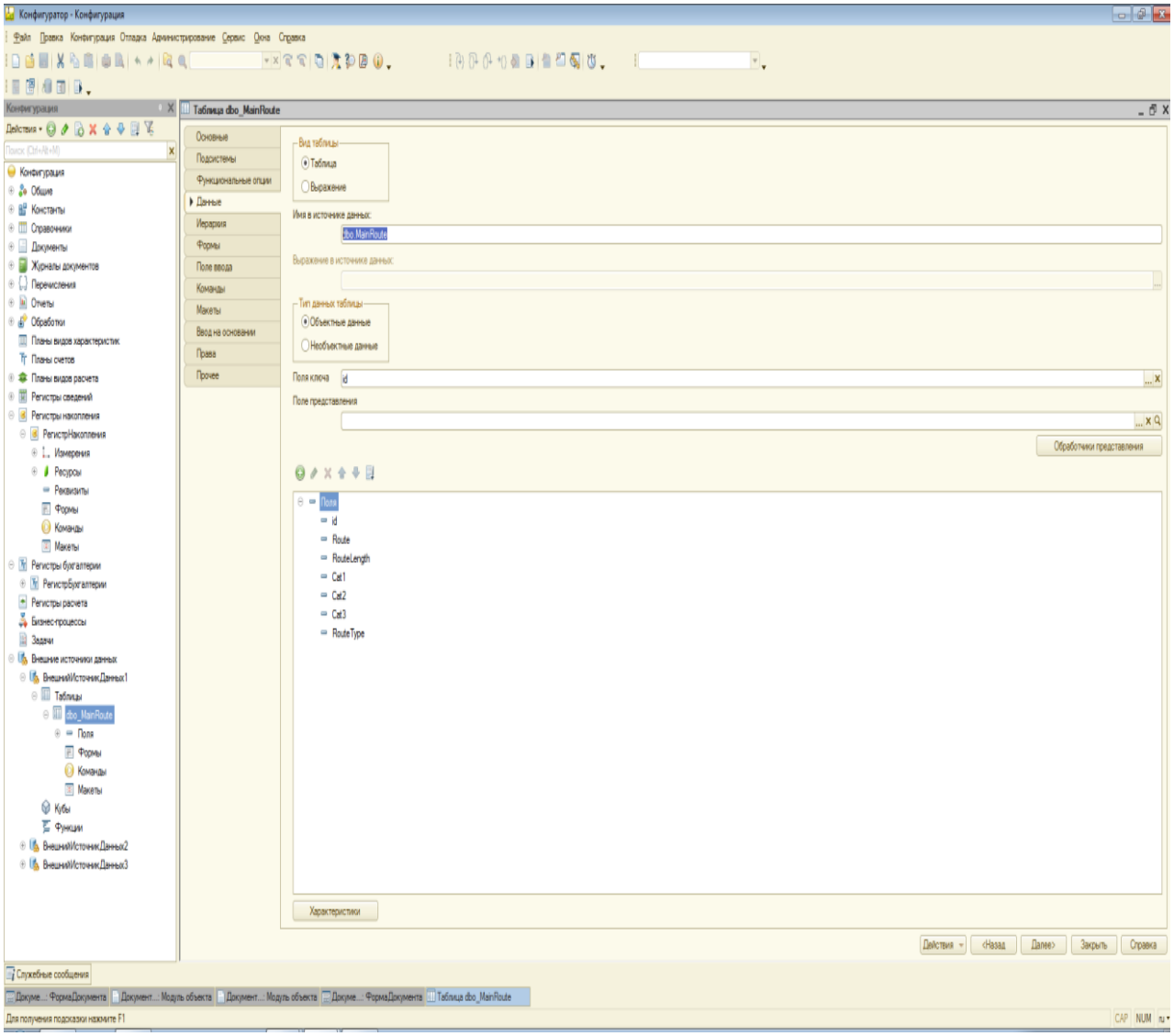

Рисунок 3.32 – Создание внешнего подключения

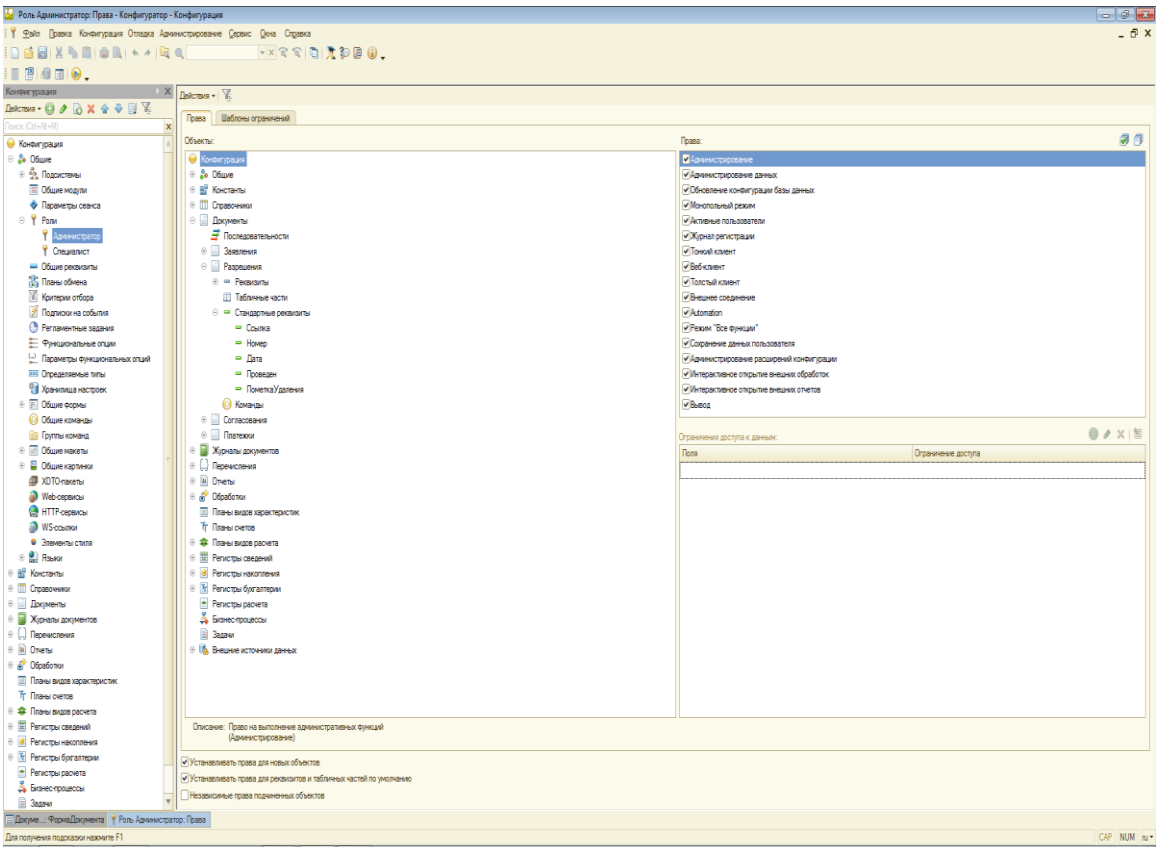

Рисунок 3.33 – Создание ролей

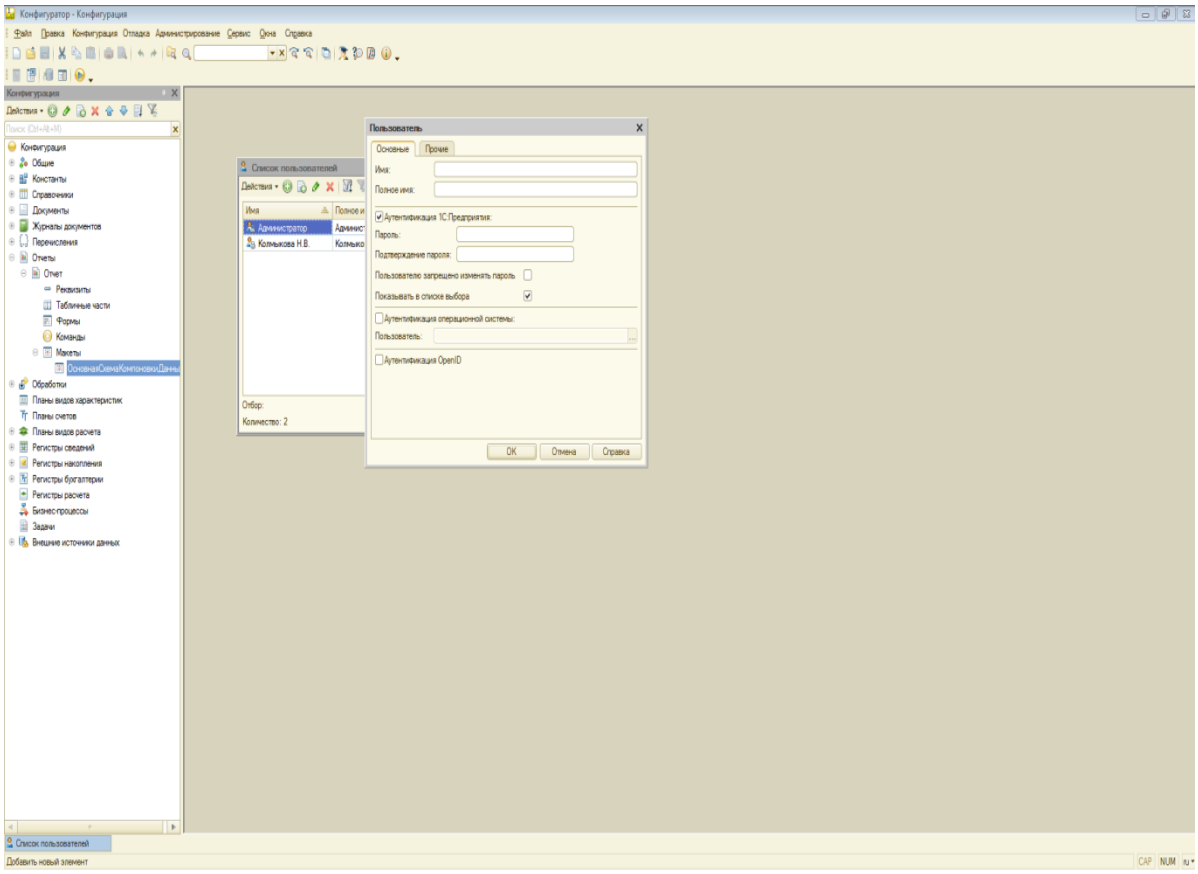

Рисунок 3.34 – Создание пользователей

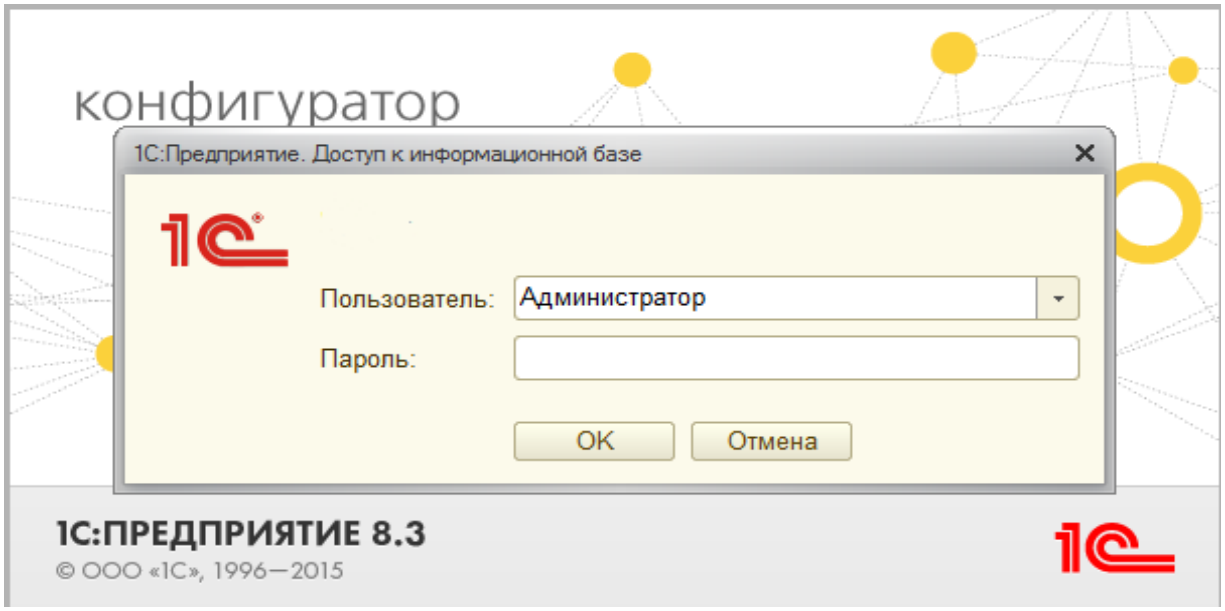

Рисунок 3.35 – Окно авторизации

Реализация модуля системы учета разрешений проведена успешно и соответствует поставленным задачам.

3.4 Программная реализация картографического модуля

В первую очередь был создан слой возможных путей. Сначала был создан простой слой объектов. Тип геометрии – поли-линия. Для каждого объекта были заданы параметры, необходимые для расчета пути, как то:

- Скорость движения;
- Время в пути при заданной скорости;
- Длина объекта;
- Название улицы.

Так же, каждому объекту был присвоен тип дорожного покрытия - поле Category, в соответствии с рисунком 3.36.

Далее, из слоя объектов был создан набор сетевых данных, из которого, с помощью набора инструментов Network Analyst, был создан и опубликован слой маршрутизации, в соответствии с рисунком 3.37 Таким образом был получен картографический сервис.

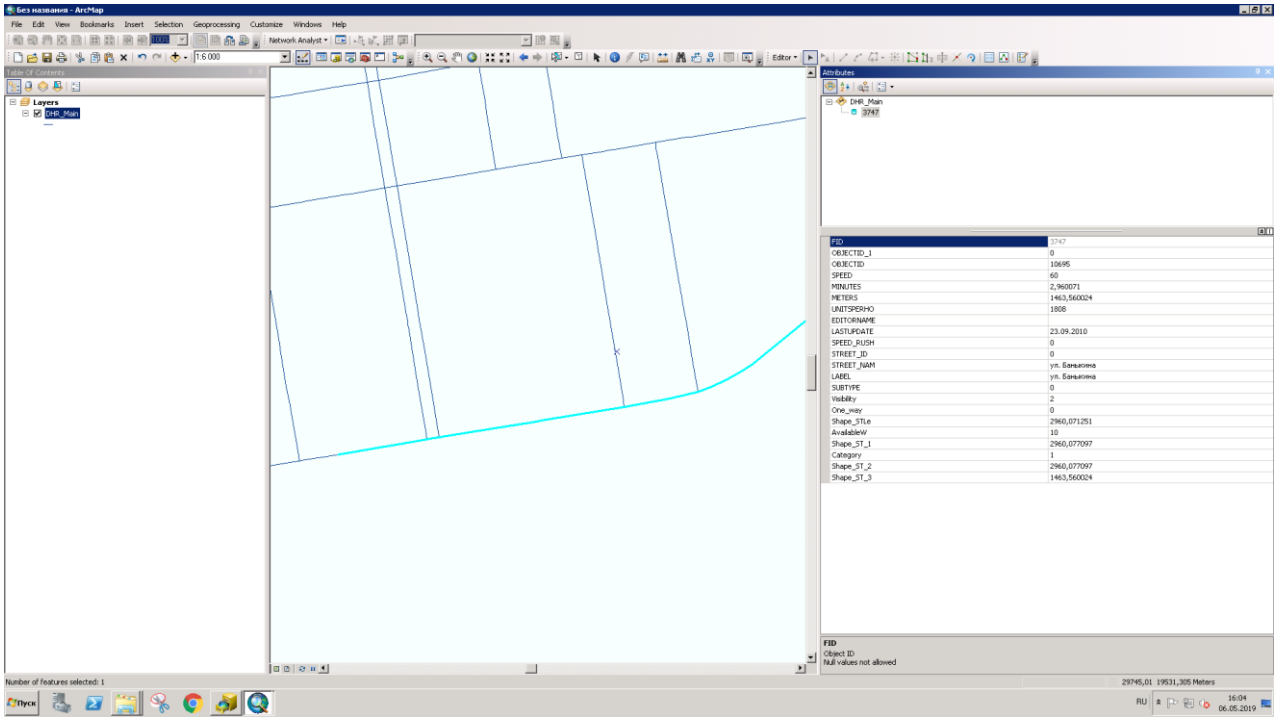

Рисунок 3.36 – Создание слоя возможных путей

Именно к этому сервису веб-приложение будет обращаться посредством библиотек API для построения маршрута и получения данных для 1С.

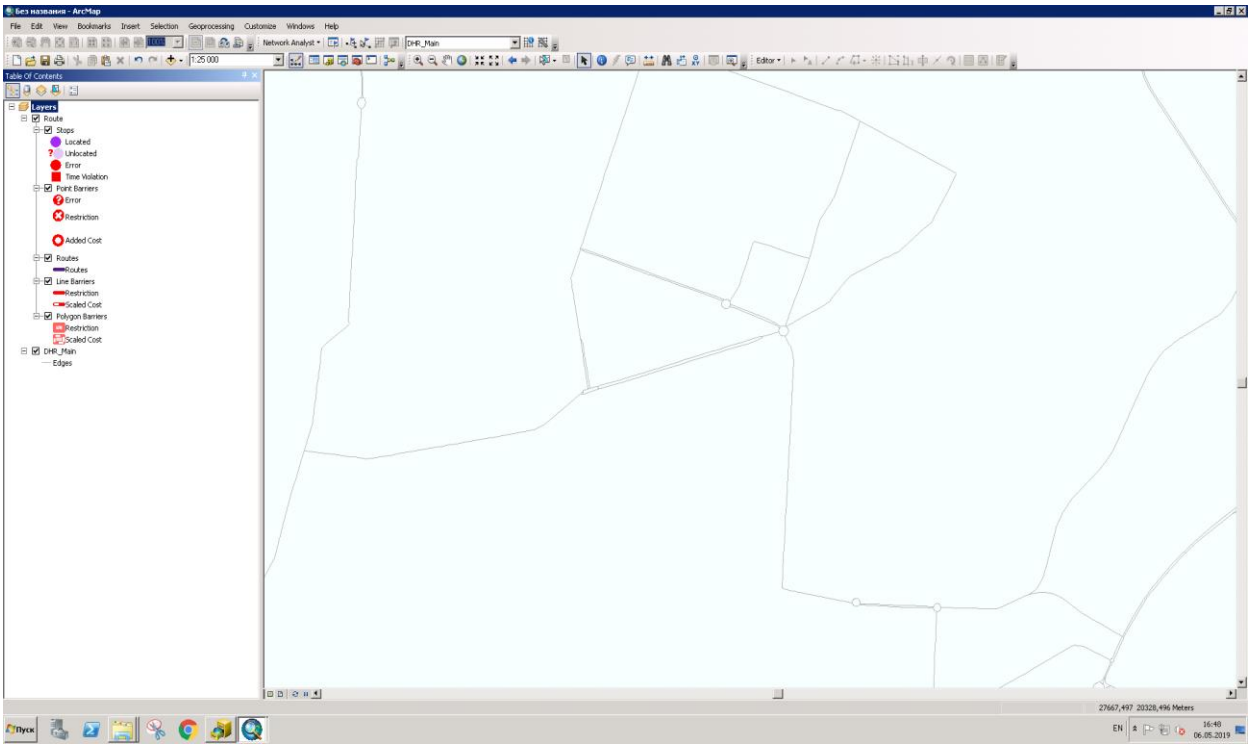

Рисунок 3.37 – Создание слоя маршрутизации

Следующим шагом сверстал рабочую область, в соответствии с рисунком 3.38.

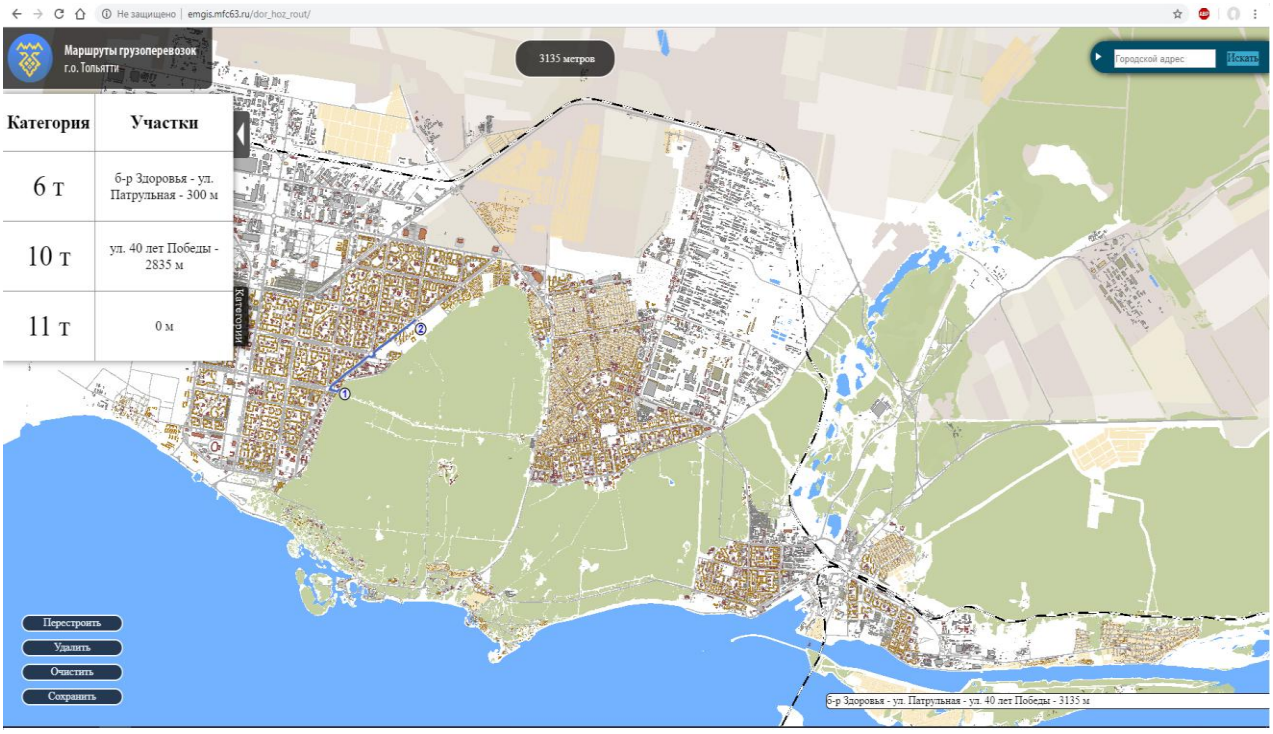

Рисунок 3.38 – Рабочая область веб-приложения

Она состоит из:

- Строки поиска дома по городскому адресу;
- Кнопок управления инструментом маршрутизации;
- Кнопки сохранения результатов в БД;
- Итоговой таблицы с длиной пути и пройденными улицами по каждой категории;
- Общей длины маршрута;
- Общего списка улиц на маршруте.

Следующий этап – написание инструментария.

Первым делом я подключил к проекту библиотеки API, в соответствии с рисунком 3.39.

```
dojo.require("dijit.form.Button");
dojo.require("dijit.layout.AccordionContainer");
dojo.require("dijit.layout.BorderContainer");
dojo.require("dijit.layout.ContentPane");
dojo.require("dojo._base.connect");
dojo.require("dojo.data.ItemFileReadStore");
dojo.require("dojo.dnd.Moveable");
dojo.require("dojo.domReady!");
dojo.require("dojo.parser");
dojo.require("dojox.grid.DataGrid");
dojo.require("esri.arcgis.utils");
dojo.require("esri.dijit.Popup");
dojo.require("esri.dijit.Legend");
dojo.require("esri.dijit.Directions");
dojo.require("esri.geometry.Geometry");
dojo.require("esri.geometry.geodesicUtils");
dojo.require("esri.geometry.jsonUtils");
dojo.require("esri.geometry.mathUtils");
dojo.require("esri.geometry.Point");
dojo.require("esri.geometry.Polyline");
dojo.require("esri.graphic");
dojo.require("esri.IdentityManager");
dojo.require("esri.InfoTemplate");
dojo.require("esri.layers.ArcGISDynamicMapServiceLayer");
dojo.require("esri.layers.FeatureLayer");
dojo.require("esri.layers.LabelClass");
dojo.require("esri.map");
dojo.require("esri.SpatialReference");
dojo.require("esri.symbols.TextSymbol");
dojo.require("esri.symbols.Font");
dojo.require("esri.tasks.DirectionsFeatureSet");
dojo.require("esri.tasks.find");
dojo.require("esri.tasks.FeatureSet");
dojo.require("esri.tasks.geometry");
dojo.require("esri.tasks.GeometryService");
dojo.require("esri.tasks.identify");
dojo.require("esri.tasks.LengthsParameters");
dojo.require("esri.tasks.QueryTask");
dojo.require("esri.tasks.query");
dojo.require("esri.tasks.RouteTask");
dojo.require("esri.tasks.RouteParameters");
dojo.require("esri.toolbars.edit");
dojo.require("esri.units");
```

Рисунок 3.39 – Подключение библиотек API

Для работы карты и инструментов необходимо загрузить их на клиент. В процедуре Init() определяется map extent (здесь и далее, некоторые определения будут приводиться в их англоязычной версии, в связи с тем, что при переводе теряется точный смысл), окно вывода информации PopUp, основные триггеры и загружаются картографические сервисы (basemap, searchTask).

Map extent – блок параметров, определяющих область отображения карты на экране. Состоит из:

- Координат границ области
  - Левая – Xmin;
  - Верхняя – Ymax;
  - Правая – Xmax;
  - Нижняя – Ymin.

- Пространственной привязки, определяющей систему координат, в соответствии с рисунком 3.40.

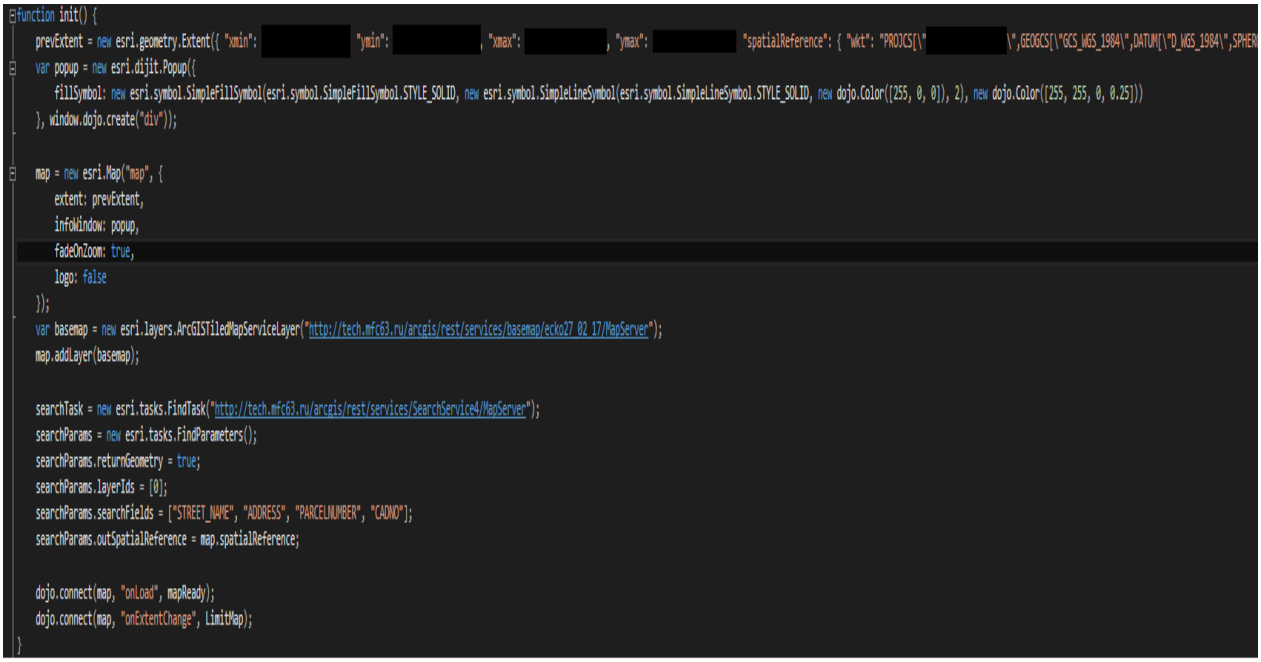

Рисунок 3.40 – Процедура Init()

Первый триггер, определенный в Init() – Ready. Он срабатывает при загрузке extent и сервисов (далее - слоев) на клиент. К нему привязана процедура MapReady(), в соответствии с рисунком 3.41. Первая ее строка отключает кнопки приближения-отдаления. Далее определяется триггер клика по карте – onClick, к нему привязывается процедура executeIdentifyTask(). Следом загружается слой маршрутизации и определяются его основные параметры routeParams:

- stops – массив поворотных точек, определяющих маршрут;
- returnDirections – определяет, вернет-ли инструмент список направлений маршрута в виде пакета данных JSON;
- directionsLengthUnits – единицы измерения пути.

Следующие три строки определяют символы поворотных точек:

- RouteSymbol – точка построенного маршрута;
- DefaultSymbol – точка не построенного маршрута;
- HighlightSymbol – изменяемая точка.

```
function mapReady() {
    map.hideZoomSlider();
    dojo.connect(map, "onClick", executeIdentifyTask);
    routeTask = new esri.tasks.RouteTask("http://tech.mfc63.ru/arcgis/rest/services/DorHozRoutes/DHR_Fixed/NAServer/Route");
    routeParams = new esri.tasks.RouteParameters();
    routeParams.stops = new esri.tasks.FeatureSet();
    routeParams.returnDirections = true;
    routeParams.directionsLengthUnits = esri.units.Meters;
    dojo.connect(routeTask, "onSolveComplete", showRoute);
    dojo.connect(routeTask, "onError", errorHandler);

    routeSymbol = new esri.symbol.SimpleLineSymbol().setColor(new dojo.Color([70, 107, 200])).setWidth(3);
    defaultSymbol = new esri.symbol.SimpleMarkerSymbol().setColor(new dojo.Color([240, 240, 240])).setOffset(5, -5);
    highlightSymbol = new esri.symbol.SimpleMarkerSymbol().setColor(new dojo.Color([255, 255, 255]));
}
```

Рисунок 3.41 – Процедура MapReady()

Построение маршрута

В свою очередь, OnClick – основной триггер программы. Привязанная к нему процедура executeIdentifyTask() запускает инструмент маршрутизации, в соответствии с рисунком 3.42.

```
function executeIdentifyTask(evt) {
    textSymbPar++;
    SymbFont = new esri.symbol.Font();
    SymbFont.setSize("11pt");
    SymbFont.setWeight(esri.symbol.Font.WEIGHT_BOLD);
    if (routeparam == 1) {
        var stop2 = map.graphics.add(new esri.Graphic(evt.mapPoint, defaultSymbol));
        var stop = map.graphics.add(new esri.Graphic(evt.mapPoint, new esri.symbol.TextSymbol().setColor(new dojo.Color([0, 0, 255])).setFont(SymbFont).setText(textSymbPar).setOffset(5, -10)));
        routeParams.stops.features.push(stop);
        routeparams2.push(stop2);
        if (routeParams.stops.features.length >= 2) {
            routeTask.solve(routeParams);
        }
    } else if (routeparam == 2) {
    }
}
```

Рисунок 3.42 – Процедура executeIdentifyTask()

Первая строка – счетчик установленных поворотных точек. Следующие три – объявление переменной для стилей текста и ее параметры. Далее – установка на карту точки с порядковым номером в месте, по которому специалист кликнул мышкой. Одна точка – графическая, вторая – текстовая. Так же они добавляются в соответствующие массивы для возможности дальнейшей работы с ними. Если количество графических точек на карте превышает одну, запускается инструмент маршрутизации. Инструмент

рассчитывает маршрут и запускает процедуру showRoute(), в соответствии с рисунками 3.43-3.46.

В ней объявил внутренние переменные. На карте отображается кривая маршрута методом map.graphics.add(…). Задал массив текстовых разделителей streetsArray[], в соответствии с рисунком 3.43.

Результат работы инструмента маршрутизации приходит на клиент в виде пакета данных JSON. Вычленил из него названия улиц и пройденное расстояние по каждой из категорий дорог, в соответствии с рисунками 3.44 и 3.45. Записал результаты в соответствующие переменные, вывел данные на экран, в соответствии с рисунками 3.45 и 3.46.

Удаление поворотной точки

Для удаления крайней поворотной точки написал процедуру Delete(), в соответствии с рисунком 3.47. Она запускается при нажатии кнопки «Удалить», в соответствии с рисунком 3.38.

```
mk1[0] = 0;
mk2[0] = 0;
mk3[0] = 0;
mkall[0] = 0;
mkrout[0] = 0;
routes = [];
routes2 = [];
map.graphics.add(solveResult.routeResults[0].route.setSymbol(routeSymbol));
routes2 = solveResult.routeResults[0];
routes3 = solveResult.routeResults[0];
routes = solveResult.routeResults[0].route.setSymbol(routeSymbol);
geometry = solveResult.routeResults[0].route;
geometrylength = geometry.geometry.paths[0].length;
directions = solveResult.routeResults[0].directions;
oldnewnum = routes.geometry.paths[0].length;
if (editind != 0) {
    oldnewarr[editind] = oldnewnum;
} else {
    oldnewarr[vari-1] = oldnewnum;
}
routescount[0] = 0;
routescount[vari - 1] = geometrylength;
var dirFeat = solveResult.routeResults[0].route.attributes.StopCount;
var streetsArray = ["ул.", "пр-д", "пр-т", "б-р", "шоссе", "площадь", "пер.", "проспект", "пл.", "М5"];
routeLengthAll = directions.totalLength;
var i = 0;
count = "";
var count2 = "";
var immedi = " go";
lengthAll = 0;
var split = "";
var par = "";
var rez = "";
var par2 = 0;
var rez2 = 0;
var rezPar = 0;
var rezPar2 = 0;
var valtext = "";
var opr = "";
var esri = "esriDMT";
var esriEnd = "";
var subtype = 0;
var dlin1 = 0;
var dlin2 = 0;
var dlin3 = 0;
var dlin1Route = "";
var dlin2Route = "";
var dlin3Route = "";
var dlinpar1 = "";
var dlinpar2 = "";
var dlinpar3 = "";
var toward = " toward";
var rezparsimple = "";
var casepar = "";
var im = " on";
var immedi = " and";
var reach = 0;
```

Рисунок 3.43 – Процедура showRoute()

```
    $.each(directions.features, function (index, value) {
        reach++;
        var newval;
        if (value.attributes.text.split("Location") == value.attributes.text) {
            if (value.attributes.text.split(im) != value.attributes.text.split(im)) {
                newval = value.attributes.text.split(im);
                valtext = newval[1];
                if (valtext.split(toward) != valtext.split(toward)) {
                        newval = valtext.split(toward);
                        valtext = newval[0];
                    }
            }
            if (value.attributes.text.split(immedi) != value.attributes.text) {
                newval = value.attributes.text.split(immedi);
                valtext = newval[0];
            }
        for (var t = 1; t <= valtext.length; t++) {
            if (valtext[t - 1] + valtext[t] + valtext[t + 1] == streetsArray[0]) {
                rez = valtext.split(streetsArray[0]);
                if (rez[1] != par) {
                    if (rez[1] != undefined) {
                        if (reach == directions.features.length) {
                            count += streetsArray[0] + " " + rez[1];
                            par = rez[1];
                            rezPar = streetsArray[0] + rez[1];
                        } else {
                            count += streetsArray[0] + " " + rez[1] + " - ";
                            par = rez[1];
                            rezPar = streetsArray[0] + rez[1];
                        }
                    }
                }
            } else if (valtext[t - 1] + valtext[t] + valtext[t + 1] + valtext[t + 2] == streetsArray[1]) {
                rez = valtext.split(streetsArray[1]);
                if (rez[1] != par) {
                    if (rez[1] != undefined) {
                        if (reach == directions.features.length) {
                            count += streetsArray[1] + " " + rez[1];
                            par = rez[1];
                            rezPar = streetsArray[1] + rez[1];
                        } else {
                            count += streetsArray[1] + " " + rez[1] + " - ";
                            par = rez[1];
                            rezPar = streetsArray[1] + rez[1];
                        }
                    }
                }
            } else if (valtext[t - 1] + valtext[t] + valtext[t + 1] + valtext[t + 2] == streetsArray[2]) {
                rez = valtext.split(streetsArray[2]);
                if (rez[1] != par) {
                    if (rez[1] != undefined) {
                        if (reach == directions.features.length) {
                            count += streetsArray[2] + " " + rez[1];
                            par = rez[1];
                            rezPar = streetsArray[2] + rez[1];
                        } else {
                            count += streetsArray[2] + " " + rez[1] + " - ";
```

Рисунок 3.44 – Процедура showRoute()

```
        lengthAll += Math.round(value.attributes.length);

        if (value.attributes.length != 0) {
            if(value.attributes.text)
                var rezParToServer = rezPar.split(" ");
            $.ajax({
                type: "POST",
                url: 'ashx/QueryRequest.ashx',
                dataType: "text",
                async: false,
                data: { rezPar: rezParToServer[rezParToServer.length - 1] },
                success: function (data) {
                    subtype = data;
                },
                error: function(xhr, ajaxOptions, thrownError) {
                    alert(xhr.status);
                    alert(thrownError);
                }
            });
        }
        if (subtype == 4 || subtype == 5) {
            dlin1 += Math.round(value.attributes.length);
            if (dlinpar1 != rezPar) { dlin1Route += rezPar + " - "; }
            dlinpar1 = rezPar;
        } else if ((subtype == 1 || subtype == 2 || subtype == 3) && rezPar != "ул. 40 лет Победы(11т.)" ) {
            dlin2 += Math.round(value.attributes.length);
            if (dlinpar2 != rezPar) { dlin2Route += rezPar + " - "; }
            dlinpar2 = rezPar;
        }
        if (rezPar == "ул. 40 лет Победы(11т.)") {
            dlin3 += Math.round(value.attributes.length);
            if (dlinpar3 != rezPar) { dlin3Route += rezPar + " - "; }
            dlinpar3 = rezPar;
        }
        }
    });
    var ost = lengthAll % 10;
    var ost2 = lengthAll % 100;
    if (ost == 0 || ost == 5 || ost ==6 || ost == 7 || ost == 8 || ost == 9) {
        opr = "метров";
    } else if (ost == 1) {
        opr = "метр";
    } else if (ost == 2 || ost == 3 || ost == 4) {
        opr = "метра";
    }
    if(Math.floor(ost2 / 10) == 1){
        opr = "метров";
    }
    count.substring(0, count.length - 1)
    cat1par = (dlin1 / 1000).toFixed(3);
    cat2par = (dlin2 / 1000).toFixed(3);
    cat3par = (dlin3 / 1000).toFixed(3);
```

Рисунок 3.45 – Процедура showRoute()

```
$("#dlin1Rez").text(dlin1Route + dlin1 + " м");
$("#dlin2Rez").text(dlin2Route + dlin2 + " м");
$("#dlin3Rez").text(dlin3Route + dlin3 + " м");
mk1[vari-1] = dlin1Route + dlin1 + " м";
mk2[vari-1] = dlin2Route + dlin2 + " м";
mk3[vari-1] = dlin3Route + dlin3 + " м";
rezBtnPrm2 = 1;
dlin1Save = dlin1;
dlin2Save = dlin2;
dlin3Save = dlin3;
$("#table").css("width", "350px");
$("#rezBtn2").css("left", "350px");
$("#rezBtn2").css("background-image", "URL('images/Left.png')");
$("#MultipleSlide").css("left", "350px");
$("#MultipleSlide").css("display", "block");
$("#ResultSlide").css("left", "350px");
$("#ResultSlide").css("display", "block");
$("#rezHeader").css("border", "1px solid black");
$("#RezMarshText").html(formatDistance(lengthAll, opr));
$('#RezMarsh').css("display", "block");
$("#allRoute").text(count + lengthAll + " м");
$("#allRoute").animate({ left: "65%" }, "slow", "swing");
mkrout[vari-1] = count + lengthAll + " м";
$("#rezBtn2").css("background-image", "url('images/Left.png')");
hided = 0;
rezBtnPrm = 1;
rezBtnPrm2 = 1;
dlin1 = 0;
dlin2 = 0;
dlin3 = 0;
sohranparam = count;
sohranLength = (lengthAll / 1000).toFixed(3);
```

Рисунок 3.46 – Процедура showRoute()

```
function Delete() {
    textSymbPar--;
    var i = routeParams.stops.features.length - 1;
    var i2 = routes.geometry.paths[0].length;
    var i3 = routes2.directions.features.length - 1;
    var v = "Depart Location " + i;
    var spl = [];
    var start, end;
    var startarr = [];
    var endarr = [];
    if (i > 0) {
        map.graphics.clear();
        routeParams.stops.features.splice(i, 1);
        routeparams2.splice(i, 1);
        if (routeParams.stops.features.length >= 2) {
            map.graphics.clear();
            for (var i = 0; i < routeParams.stops.features.length; i++) {
                map.graphics.add(new esri.Graphic(routeparams2[i].geometry, defaultSymbol));
                map.graphics.add(new esri.Graphic(routeParams.stops.features[i].geometry, new esri.symbol.TextSymbol().setColor(new dojo.Color([0, 0, 255])).setFont(SymbFont).setText(i + 1).setOffset(5, -10)));
            }
            routeTask.solve(routeParams);
            solveParam++;
        } else if (routeParams.stops.features.length == 1) {
            map.graphics.clear();
            map.graphics.add(new esri.Graphic(routeparams2[0].geometry, defaultSymbol));
            map.graphics.add(new esri.Graphic(routeParams.stops.features[0].geometry, new esri.symbol.TextSymbol().setColor(new dojo.Color([0, 0, 255])).setFont(SymbFont).setText(1).setOffset(5, -10)));
            mk1 = [];
            mk2 = [];
            mk3 = [];
            mkall = [];
            mkrout = [];
            $("#dlin1Rez").text("");
            $("#dlin2Rez").text("");
            $("#dlin3Rez").text("");
            $("#RezMarshText").html("0 м");
            $("#allRoute").text("0 м");
            $('#RezMarsh').css("display", "none");
            $("#allRoute").animate({ left: "100%" }, "slow", "swing");
            $("#RezDiv2").animate({ "width": "0px" }, "slow", "swing");
            $("#rezBtn2").css("background-image", "url('images/Right.png')");
            $("#rezBtn2").animate({ "left": "0px" }, "slow", "swing");
            vari = 1;
        }
    }
}
```

Рисунок 3.47 – Процедура Delete()

Здесь программа уменьшает счетчик точек на 1, удаляет крайний элемент из массивов поворотных точек и массива параметров маршрутизации. Если на карте все еще больше одной точки – повторно запускает инструмент расчета маршрута, с отредактированными параметрами и массивами.

Перестроение маршрута

Для редактирования маршрута необходимы 2 процедуры, привязанные к разным триггерам(), в соответствии с рисунком 3.45. Первый – к моменту захвата графики мышкой, второй – к ее освобождению. Процедуры становятся активны после нажатия на кнопку «Перестроить» , в соответствии с рисунком 3.35 и отключаются при повторном нажатии.

Первая процедура запоминает порядковый номер захваченной точки. Вторая – перезаписывает массивы поворотных точек, меняя координаты захваченной точки на новые и запускает инструмент построения маршрута.

```
function holdGraphic(evt) {
    rebuildID = evt.target.textContent - 1;
}
function releaseGraphic(evt) {
    if (routeparam == 2) {
        var Newstop2 = map.graphics.add(new esri.Graphic(evt.mapPoint, defaultSymbol));
        var Newstop = map.graphics.add(new esri.Graphic(evt.mapPoint, new esri.symbol.TextSymbol().setColor(new dojo.Color([0, 0, 255])).setFont(SymbFont).setText(rebuildID + 1).setOffset(5, -10)));
        routeParams.stops.features[rebuildID] = Newstop;
        routeparams2[rebuildID] = Newstop2;
        map.graphics.clear();
        if (routeParams.stops.features.length >= 2) {
            for (var i = 0; i < routeParams.stops.features.length; i++) {
                map.graphics.add(new esri.Graphic(routeparams2[i].geometry, defaultSymbol));
                map.graphics.add(new esri.Graphic(routeParams.stops.features[i].geometry, new esri.symbol.TextSymbol().setColor(new dojo.Color([0, 0, 255])).setFont(SymbFont).setText(i + 1).setOffset(5, -10)));
            }
            routeTask.solve(routeParams);
            solveParam++;
        }
    }
}
```

Рисунок 3.48 – Процедуры перестроения маршрута

Сохранение маршрута

После построения маршрута список улиц и длина пути (общая и по категориям) сохраняются в базу данных для дальнейшего использования в модуле 1С кликом по кнопке «Сохранить», в соответствии с рисунком 3.38.

3.5 Тестирование системы

Автоматизированная система выдачи разрешений тестируется от лица специалиста.

Задачи:

- Авторизоваться;
- Создать новый экземпляр документа «Заявления»;
- Заполнить реквизиты согласно тестового заявления;
- Построить произвольный маршрут;
- Рассчитать размер компенсации;
- Распечатать сводную таблицу и платежное поручение;
- Зарегистрировать новый экземпляр документа «Платежки» и привязать его к тестовому заявлению;
- Распечатать сопроводительную документацию;
- Подтвердить появление записей в регистрах после проведения тестового экземпляра документа «Заявления»;
- Создать новый экземпляр документа «Заявления»;
- Воспользовавшись формой авто-заполнения, заполнить реквизиты на основании предыдущего заявления.

При тестировании, были получены следующие результаты:

- Вход в систему успешен;
- Был создан и заполнен экземпляр документа;
- Произвольный маршрут успешно построен и сохранен;
- Система рассчитала вред и компенсацию, вывела на печать сводную таблицу и платежное поручение;
- Зарегистрирован новый экземпляр документа «Платежки», привязан к тестовому заявлению;
- Система успешно вывела на печать блок документов;
- Записи в регистрах присутствуют;
- Еще один экземпляр документа «Заявления» создан успешно;

- Реквизиты заполнены автоматически через форму авто-заполнения.

Реализация картографического модуля проведена успешно и соответствует поставленным задачам.

Вывод по главе 3

В ходе реализации автоматизированной системы были выбраны и описаны архитектуры разрабатываемых модулей и СУБД картографического модуля.

Были реализованы модуль регистрации заявлений на базе «1С: Предприятие» и картографический сервис маршрутизации на базе ESRI ArcGIS.

В ходе тестирования был реализован алгоритм подачи заявления и выдачи разрешения, а также авто-заполнение заявления на основе уже зарегистрированного.

Система прошла проверку и работает согласно поставленного ТЗ.

ЗАКЛЮЧЕНИЕ

Итогом бакалаврской работы является разработанная система выдачи разрешений на перевозку тяжелых грузов по территории г.о Тольятти.

Бакалаврская работа удовлетворяет основным требованиям, предъявленным в задании, и реализует функционал, необходимый специалисту Департамента дорожного хозяйства.

В данный момент система прошла опытную эксплуатацию, внедрение и находится в производственной эксплуатации.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. Приказ Минтранса России от 24.07.2012 N 258 (ред. от 21.09.2016, с изм. от 16.01.2017) "Об утверждении Порядка выдачи специального разрешения на движение по автомобильным дорогам транспортного средства, осуществляющего перевозки тяжеловесных и (или) крупногабаритных грузов" (Зарегистрировано в Минюсте России 11.10.2012 N 25656). [Электронный ресурс]: Сайт справочной правовой системы: http://www.consultant.ru/document/cons_doc_LAW_136642/

Научная и методическая литература

2. Акперов И. Г. Информационные технологии в менеджменте: Учебник / И. Г. Акперов А. В.Сметанин И. А. Коноплева. — М.: НИЦ ИНФРА–М, 2013. — 400 с.

3. Золотухина Е.Б., Красникова С. А., Вишня А.С. Моделирование бизнес-процессов. Курс лекций. – КУРС, 2017. – 79с

4. Стешин А. И. Информационные системы в организации [Электронный ресурс]: учебное пособие – Вузовское образование, 2019. – 194 с.

5. Золотов С. Ю. Проектирование информационных систем: учеб. пособие. – Томск: Эль Контент, 2013. – 88 с.

Электронные ресурсы

6. Балдин К. В. Информационные системы в экономике. Учеб. пособие – Москва : ИНФРА-М, 2017. [Электронный ресурс]: http://znanium.com/bookread2.php?book=661252

7. Силич В. А. Реинжиниринг бизнес-процессов: Учебное пособие – Томск: ТУСУР, 2014. [Электронный ресурс]: https://edu.tusur.ru/publications/680

8. Бжиская Ю.В. Английский язык: информационные системы и технологии. - Издание 2-е. Учебное пособие – Москва: Феникс, 20139. [Электронный ресурс]: https://elib.pstu.ru/vufind/Record/RUPSTUbooks165993

9. Гарагуля С.И. Английский язык в сфере информационных систем и технологий. - Москва: КНОРУС, 2014. [Электронный ресурс]:

https://aldebaran.ru/author/i_garagulya_s/kniga_angliyiskiyi_yazyik_v_sfere_informacionn/

10. Пирогов В.Ю. Информационные системы и базы данных: организация и проектирование. – Санкт-Петербург: БХВ-Петербург, 2016. - [Электронный ресурс]: https://www.litres.ru/vladislav-pirogov/informacionnye-sistemy-i-bazy-dannyh-organizaciya-i-proektirovanie-uchebnoe-posobie/

11. Лашина М.В., Соловьев Т.Г. Информационные системы и технологии в экономике и маркетинге. – Москва: КНОРУС, 2017. - [Электронный ресурс]: https://www.book.ru/book/922282

12. Избачков Ю., Петров В., И Телина И. Информационные системы. – Санкт-Петербург: ПИТЕР, 2013. - [Электронный ресурс]: https://nashol.com/2016042989179/informacionnie-sistemi-izbachkov-u-s-petrov-v-n-vasilev-a-a-telina-i-s-2011.html

13. Советов Б.Я., Цехановский В.В. Информационные технологии. Теоретические основы. – Интернет: Лань, 2017. - [Электронный ресурс]: https://biblio-online.ru/book/informacionnye-tehnologii-372405

14. Алиев В.С Информационные технологии и системы финансового менеджмента. Учебное пособие – Москва: КНОРУС, 2017. - [Электронный ресурс]: https://market.yandex.ru/product--aliev-vagif-sudeif-ogly-informatsionnye-tekhnologii-i-sistemy-finansovogo-menedzhmenta-uchebnoe-posobie/1467490

Литература на иностранном языке

15. Jane P. Laudon, Kenneth P. Laudon «Essentials of business information systems» – Prentice Hall, 200617. Ralph Stair, George Reynolds «Fundamentals of information systems» - Course Technology, 2015

16. Carlos Coronel, Stevens Morris, Peter Rob «Database systems: design, implementation, and management» - Course Technology, 201219. Kathy Schwalbe «Information technology project management» - Cengage Learning, 2013

17. David Olson «Information systems project management» - Business expert press, 2014

18. Keri E. Pearlson, Carol S. Sanders «Managing and using information systems» - Wiley, 2012

19. Jeffrey Whitten, Lonnie Bentley «Systems analysis and design methods» - McGraw-Hill/Irwin, 2005

20. Efraim Turban, Carol Pollard «Information technology for management» - Wiley, 2013