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

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ МОНИТОРИНГА ВЫБРОСОВ ВРЕДНЫХ ВЕЩЕСТВ В АТМОСФЕРУ АВТОТРАНСПОРТОМ

Выпускная квалификационная работа

обучающегося по направлению подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения,
группы 07001402
Самхарадзе Кобы Кобаевича

Научный руководитель
ст.пр., аспирант Ерошенко Я.Б.

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современное развитие и повсеместное внедрение информационных технологий способствуют появлению все новых методов исследования явлений в организации каких-либо процессов для принятия в дальнейшем наиболее эффективных управленческих решений. Процесс управления, в данном случае, строится на комплексном мониторинге факторных показателей, их анализе с учетом влияния на поставленную цель и, в конечном итоге, на оценке альтернативных вариантов решения при выборе наиболее оптимальных вариантов.

Использование современных информационных технологий позволяет сформировать такие модели системы управления, которые будут учитывать не только значения входных данных, но и взаимодействия этих данных с другими элементами объекта управления в конкретном процессе, в том числе при исследовании выбросов вредных веществ в атмосферу автотранспортом.

Проблема загрязнения воздушного бассейна автотранспортом в России сохраняет свою актуальность на протяжении длительного времени, так как использование транспорта связано с выбросами таких продуктов сгорания топлива, как оксида азота, диоксида серы, оксида углерода, сажи, неорганической пыли, сероводорода, предельных углеродов и др. Эти вещества попадают в организм человека через систему дыхания и в дальнейшем являются причинами усталости, головной боли, раздражения. Более того, отрицательное влияние загрязняющих веществ (ЗВ) в атмосферном воздухе, приводит к развитию таких заболеваний, как пневмония, бронхиальная астма, онкология, бесплодие у женщин и многих других заболеваний. В связи с этим необходимо тщательное исследование загрязняющих атмосферу веществ, содержащихся в выхлопных газах автомобилей.

Наиболее актуальным решением в проведении качественного мониторинга является использование специализированной информационной системы, которая способна реализовать такие преимущества, как возможность вариантного моделирования процесса с учетом взаимосвязей элементов объекта, а также

способность к оперативному уточнению сформированной модели управляемого процесса при дополнении его необходимой информацией.

Предварительные наблюдения и исследования экологической обстановки в стране показывают, что уровень антропогенной нагрузки на окружающую среду повышается, число случаев высокого загрязнения атмосферного воздуха растет из года в год, количество зарегистрированных пациентов болезнями органов дыхания увеличивается. К тому же в настоящее время планируется увеличить объем обеспеченности населения автомобилями путем расширения автокредитования. Учитывая, что в рейтинге стран по выбросам углекислого газа Россия находится на четвертом месте, а по источникам загрязнения атмосферного воздуха в стране автотранспорт занимает третье место после объектов химической и целлюлозно-бумажной промышленности, то влияние транспорта на окружающую среду вызывает беспокойство и, если не предпринимать никаких мер по защите атмосферы, дальнейшее ухудшение ее состояния будет критичным. Для снижения общей антропогенной нагрузки на окружающую среду разработана государственная программа на 2012-2020 гг., предусматривающая повышение экологической эффективности экономики.

Данные исследования послужили выбору темы выпускной квалификационной работы (ВКР) «Разработка информационной системы прогнозирования выбросов вредных веществ в атмосферу автотранспортом».

Практическая ценность работы состоит в том, что разработанная в рамках данного проекта информационная система (ИС) будет способствовать проведению инвентаризации выбросов вредных веществ от автотранспортных средств (АТС), в том числе и от дорожно-строительных машин (ДСМ). Это позволяет организовать более эффективную и безопасную работу при строительстве и реконструкции автомобильных дорог, а также производить мероприятия по исследованию выбросов в атмосферу от АТС и/или ДСМ для проведения контроля и предвидения процесса превышения их предельно-допустимых норм.

В связи с этим целью выпускной квалификационной работы является разработка информационной системы мониторинга выбросов вредных веществ в атмосферу автотранспортом.

Для достижения данной цели необходимо выполнить следующие задачи:

- дать общую характеристику объекта исследования;

- обосновать необходимость разработки информационной системы;

- спроектировать и разработать структуру базы данных (БД);

- реализовать windows-приложение для мониторинга выбросов вредных веществ в атмосферу автотранспортом;

- провести тестирование windows-приложения на реальных данных.

Объектом исследования является предприятие ООО «Россошанское дорожное ремонтно-строительное управление №1», расположенное в г. Россошь Воронежской области.

Предметом исследования послужила система мониторинга выбросов вредных веществ автотранспортом на исследуемом предприятии.

При разработке информационной системы использовались такие методы и инструментальные средства как системный и сравнительный анализ, математический аппарат статистического анализа, метод восходящего проектирования БД, метод объектно-ориентированного визуального проектирования.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка использованной литературы и приложений. Введение отражает актуальность выбранной темы, цель и задачи. В первой главе дана общая характеристика объекта исследования, обозначена необходимость разработки специализированной ИС и анализ существующих программных продуктов. Во второй главе описаны основные технологии и средства разработки предмета исследования, методы проектирования структуры БД и осуществлена программная реализация разработанной ИС. Третья глава содержит описание работы спроектированной информационной системы и результаты ее тестирования. В заключении подведены итоги и сделаны соответствующие выводы. В приложениях размещены карта участка автодороги и разработанные коды.

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ ДЛЯ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

1.1 Технико-экономическая характеристика предметной области

Предметная область информационной системы – это совокупность реальных процессов и объектов, обладающих определенным набором свойств (атрибутов), представляющих интерес для пользователей системы [13, с. 9].

Объектом исследования выпускной квалификационной работы является Общество с ограниченной ответственностью «Россошанское дорожное ремонтно-строительное управление №1» (ООО «Россошанское ДРСУ №1»).

Предприятие зарегистрировано 10 апреля 2000 г. Администрацией Россошанского района Воронежской области и присвоен ИНН 3627016580, 15 октября 2002 г. Межрайонной инспекцией Федеральной налоговой службы №12 по Воронежской области присвоен ОГРН 1023601232533. Юридический адрес компании: 396659, Воронежская область, Россошанский район, город Россошь, Октябрьская площадь, 137-б.

ООО «Россошанское ДРСУ №1» действует на основании Устава. Уставный капитал предприятия составляет 1 726 026 (один миллион семьсот двадцать шесть тысяч двадцать шесть) рублей. Учредителями общества являются физические лица в количестве 3-х человек. Руководителем предприятия является директор Ярошев Валерий Анатольевич.

Основным видом деятельности общества является «Строительство автомобильных дорог и автомагистралей», также зарегистрировано 6 дополнительных видов деятельности.

ООО «Россошанское ДРСУ №1» является успешным предприятием малого и среднего бизнеса и уже более 15 лет осуществляет строительство автомобильных дорог и тоннелей, возведение мостов, занимается их ремонтными работами и содержанием. В компании трудится коллектив единомышленников и

высококвалифицированных специалистов, способных решать производственные задачи любой сложности.

Организационная структура компании состоит из нескольких отделов под руководством руководителя предприятия (см. рис. 1.1).

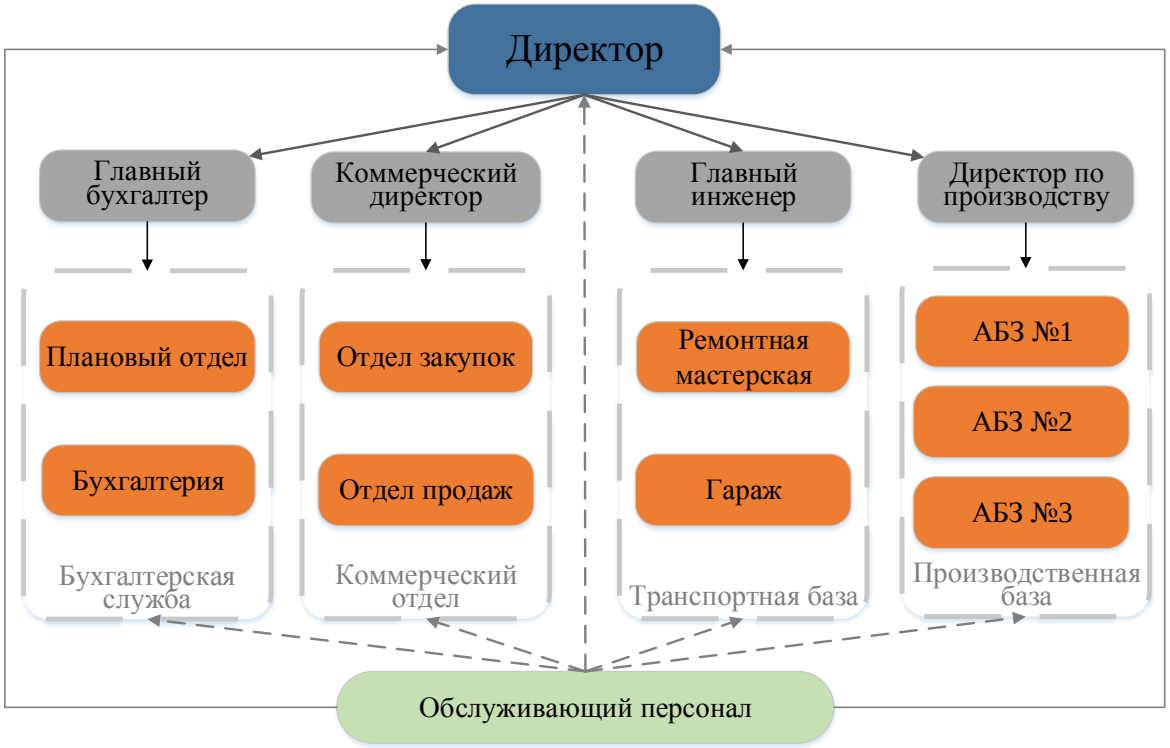


Рис. 1.1. Организационная структура ООО «Россошанское ДРСУ №1»

Бухгалтерская служба включает в себя плановый отдел, занимающийся составлением сметной документации по проектам строительства и реконструкции дорог, и бухгалтерии, осуществляющей подготовку и сдачу бухгалтерской отчетности.

Коммерческий отдел создан для реализации проектов по дополнительным видам деятельности и включает в себя два подразделения отдел закупок и отдел продаж.

Производственная база включает три завода по производству асфальтобетонной смеси, рассчитанных на выпуск 1000 тонн асфальтобетона в сутки.

Транспортная база является основным инструментом деятельности ООО «Россошанское ДРСУ №1» и состоит из производственных и ремонтных поме-

щений, парка современной дорожно-строительной техники и грузовых автомобилей, находящихся в собственности компании.

Информационная база предприятия состоит из 10 единиц компьютерной техники, связанной между собой в локальную сеть. В компании используется лицензионное программное обеспечение: операционная система Windows 7, программы из пакета Microsoft Office 2007 (Word, Excel и др.), программа для автоматизации бухгалтерского и налогового учета «1С Бухгалтерия 8».

Услугами предприятия пользуются не только в Россошанском районе, но и в других районах Воронежской области. Все услуги выполняются в соответствии с утвержденными нормами, и за период работы на объектах не было выявлено случаев нарушения технологии производства [40].

1.2 Обоснование необходимости разработки ИС

Мониторинг, по своей сути, представляет собой систему повторных наблюдений за состоянием объектов окружающей среды в пространстве и во времени в соответствии с заранее подготовленной программой [20, с. 43].

Мониторинг выбросов на практике состоит из наблюдения за состоянием атмосферы, оценки ее фактического состояния и уровня загрязнения, а также прогноза состояния атмосферного воздуха в результате возможных загрязнений в процессе производственного цикла.

Основная причина осуществления мониторинга выбросов загрязняющих веществ в атмосферу заключается в том, что главным загрязнителем атмосферного воздуха является автомобильный транспорт. В отличие от промышленных предприятий, которые оказывают отрицательное воздействие на атмосферу локально, транспортные выбросы распространяются на территории всей страны по автомагистралям и дорогам местного назначения [27, с. 18].

Проведенные исследования свидетельствуют, что с развитием техники и промышленности экологическая обстановка в стране ухудшается, растет число зарегистрированных случаев высокого загрязнения атмосферного воздуха.

При изучении данного вопроса было выявлено, что за первое полугодие 2017 года было зафиксировано 22 таких случаев, а за 9 месяцев 2017 года уже 27 случаев. Но в то же время затраты на охрану атмосферного воздуха сокращаются, тогда как по всем остальным направлениям природоохранной деятельности эти затраты увеличиваются [42].

Объем выбросов веществ, загрязняющих атмосферу от передвижных источников, увеличивается из года в год (см. рис. 1.2).

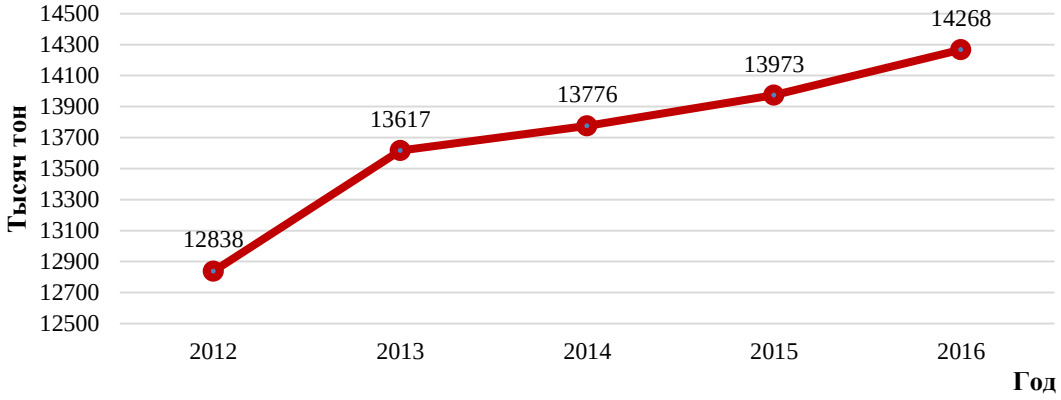


Рис. 1.2. Выбросы вредных веществ в атмосферу передвижными источниками

Больше всего в воздух выбрасывается оксида углерода, что наиболее характерно для автотранспорта (см. рис. 1.3). По данным Росстата в 2016 году количество углекислого газа в расчете на единицу площади страны составляет 925 кг/км^2, а на душу населения – 108 кг [42].

Доля автомобильного транспорта, как источника негативного воздействия на окружающую среду и здоровье населения, составляет более половины процентов среди всех источников загрязнения. Более 300 видов загрязняющих веществ (бензол, бенз(а)пирен, формальдегид, ацетальдегид и др.) содержатся в выхлопных газах автотранспорта [18, с. 5].

Для ДСМ характерны выбросы таких вредных веществ, как оксида углерода, оксидов азота, диоксида серы, диоксида углерода, метана, твёрдых частиц (PM) и др. [35, с. 7-11].

Количество транспортных единиц у граждан России увеличивается с каждым годом. По наличию автомобилей (легковых и грузовых) Центральный Федеральный округ находится на первом месте среди всех округов Российской Федерации. Воронежская область, где находится исследуемое предприятие, занимает вторую позицию с показателем 913 774 единиц после Москвы и Московской области [41].

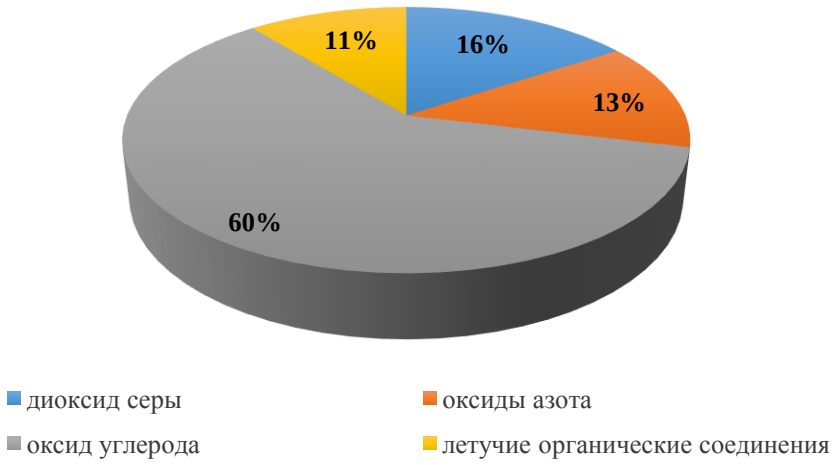


Рис. 1.3. Структура выбросов загрязняющих веществ в атмосферу

На сегодняшний день в Российской Федерации деятельность по охране атмосферного воздуха регулируется Федеральным законом от 04.05.1999 №96-ФЗ «Об охране атмосферного воздуха», согласно которому все юридические лица, имеющие источники загрязнения, должны осуществлять производственный контроль над охраной атмосферного воздуха (статья 25) [2].

Предельно допустимое содержание загрязняющих веществ в атмосферном воздухе населенных мест регламентируются гигиеническими нормативами ГН 2.1.6.1338-03, где указано, что максимальная концентрация, например, оксида азота не должна превышать 0,4 мг/м^3, диоксида серы – 0,5 мг/м^3, оксида углерода – 5 мг/м^3 и т. д. [3].

Поэтому на всех этапах строительства предприятию необходимо выполнять действующие требования по охране окружающей среды, так как на участ-

ке строительства дороги всегда образуется загазованность выхлопными газами дорожно-строительной техники, имеющими высокий класс опасности [24, с. 7].

В связи с этим необходимо осуществлять не только регулярный контроль над техническим состоянием парка машин, но и своевременно производить проверку содержания вредных веществ в выхлопных газах автомобилей. Тем более что строительство и реконструкция автомобильных дорог сопряжено с составлением проектной документацией, которая характеризуется некоторыми технико-экономическими показателями, отвечающими за эффективность и безопасность намечаемых работ [26, с. 421, 440].

Расчет выбросов загрязняющих веществ в атмосферный воздух автотранспортом на стадии проектирования отвечает именно за безопасность строительства и реконструкции дороги.

Для ООО «Россошанское ДРСУ №1» такой расчет производится сторонней организацией на платной основе, что является не выгодным и затягивает процесс составления сметной документации. В связи с этим предприятию требуется собственный программный продукт для осуществления мониторинга выбросов ДСМ в период строительства и реконструкции объекта. Для этих целей предлагается разработать специализированную ИС, которая будет отвечать всем требованиям организационного и производственного процесса строительства. Применение такого рода информационных технологий оптимизирует информационный и документационный процессы, способствует выдаче верной результативной информации, удобной для принятия управленческого и технологического решения, и значительно сокращает как рабочее время специалиста, так и материальную составляющую всего предприятия [11, с. 10].

1.3 Анализ существующих программных средств

Разработка программных средств по охране окружающей среды в России началась с 1990 года, но они не были открытыми и ориентировались на конкретного пользователя-заказчика.

С введением в действие Международных стандартов финансовой отчетности (МСФО) в России в 2011 году стал активно развиваться процесс внедрения различных автоматизированных систем управления бизнесом. Вместе с этим и росла автоматизация процессов по охране окружающей среды, тем более что с 2012 года начала работать Государственная программа РФ по повышению уровня экологической безопасности и сохранения природных систем, требующая новых экологически эффективных инновационных технологий [4].

На сегодняшний день существует несколько информационных решений в данной области таких, как ПК «Кедр», «ЭКО-Эксперт», ООО «ЭкоМастер» и др. Однако данные программные комплексы направлены на выполнение расчетов по экологическим платежам и на учет отходов производства.

Для контроля загрязнений, непосредственно, атмосферного воздуха представлены коммерческие информационные системы группы разработчиков компании ООО «Экоцентр» [46] и фирмы «Интеграл» [47].

Это масштабные комплексные информационные решения предназначены для экологов-проектировщиков, экологических надзорных служб и крупных предприятий, и, соответственно, при использовании требуют наличие профессиональных знаний пользователей.

Конфигурация данных программных решений состоит из программных средств с использованием высокопроизводительного SQL-сервера с открытым кодом FireBird и дополнительных модулей к ним, при этом необходимо приобретать еще и электронные ключи.

Бесплатные программные продукты, в основном, представлены для использования отдельными регионами, например, программы по расчету выбросов компании ООО «Интертрэйд» [43], рассчитанных для Ханты-Мансийского АО, Ямало-Ненецкого АО и Уральского Федерального округа, где специальные коэффициенты учитывают региональные особенности только данных территорий. К тому же данные бесплатные приложения не своевременно учитывают изменения соответствующих Методик.

Информационная система, разрабатываемая в рамках данной ВКР, во-первых, соответствует всем имеющимся стандартам и действующему на сегодняшний день законодательству РФ.

Во-вторых, является узкоспециализированным программным продуктом с простым, понятным интерфейсом и технически выгодным решением, предназначенным для мелкого и среднего бизнеса, не требующего высоких профессиональных знаний, специальной регистрации и покупки электронных ключей, а также присутствия администратора в период обновления.

Особенностью разрабатываемой ИС является возможность осуществлять расчет выбросов загрязняющих веществ в атмосферу в период строительства и реконструкции автомобильных дорог не только от дорожно-строительной техники, но и учитывать при этом поток движущегося городского автотранспорта по специально выделенной полосе участка строительства путем подключения дополнительного модуля, в то время, как в представленных выше программных продуктах, этого сделать было невозможно.

1.4 Основные принципы разрабатываемой ИС предприятия

В широком смысле информационной системой называется «комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей» [15, с. 13].

В узком смысле информационная система – это система сбора, хранения, накопления, поиска и передачи информации, применяемая в процессе управления или принятия решений, включающая базы данных, СУБД и специализированные прикладные программы [32, с. 6].

Для того чтобы определить какой должна быть разрабатываемая ИС, необходимо обратиться к общей классификации информационных систем (см. рис. 1.4) [22, с. 28].

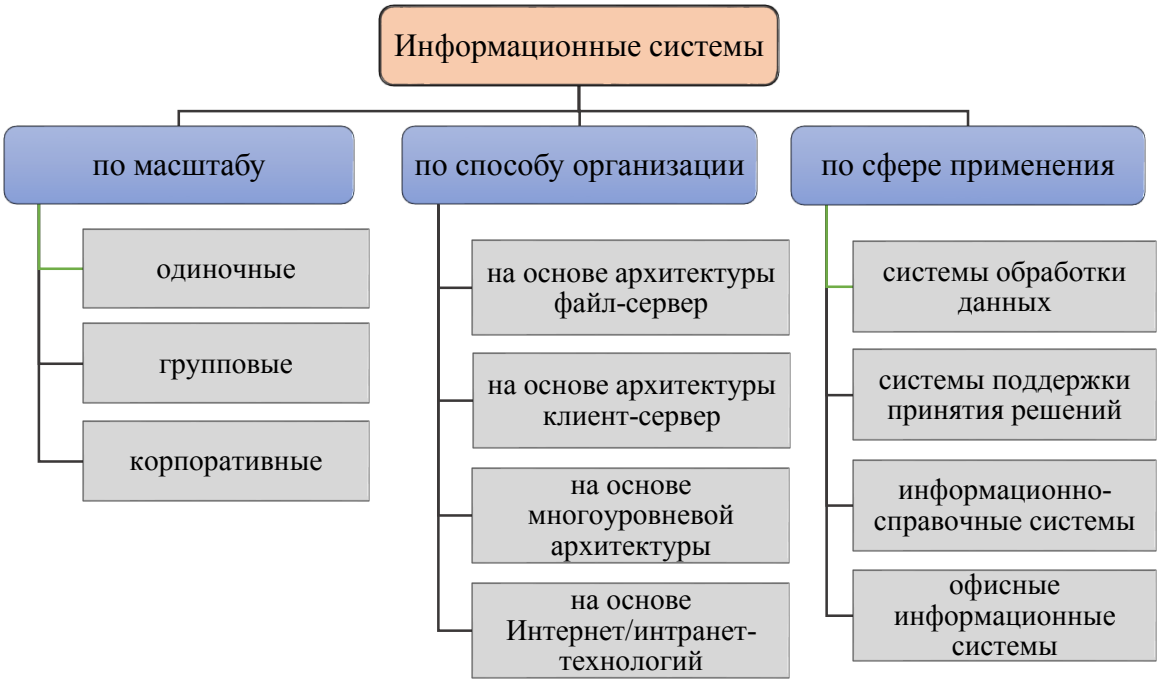


Рис. 1.4. Общая классификация ИС

Исходя из общей классификации, разрабатываемая в рамках данной работы ИС должна быть:

- по масштабу – одиночной, то есть системой, которая будет реализована без использования сети на автономном персональном компьютере (ПК) и будет рассчитана на одно рабочее место с одним или несколькими пользователями, работающими в различные моменты времени;

- по способу организации – на основе архитектуры клиент-сервер, то есть система будет использовать выделенные серверы баз данных (БД), способных выполнять поиск, сортировку и преобразование детализированной информации в так называемые «пакеты» данных (агрегирование информации), причем, данная ИС будет иметь двухуровневую модель, когда приложение работает у клиента, а СУБД — на сервере;

- по способу применения – системой обработки данных, то есть способной накапливать, сохранять, обновлять данные, а также осуществлять поиск и выдачу информации.

Таким образом, разрабатываемой информационной системе будет соответствовать следующая классификационная схема (см. рис. 1.6):

Рис. 1.6. Классификационная схема разрабатываемой ИС

Разработка ИС базируется на принципах эффективности, соответствия назначению и применению, совместимости, а также на использовании нормативно-правовой базы (федеральных законах, ГОСТах и пр.).

Эффективность информационной системы – это «совокупность свойств системы, обусловливающих возможность ее использования для удовлетворения определенных потребностей пользователей в соответствии с ее назначением».

К основным показателям эффективности ИС относятся ее гибкость, надежность, достоверность, безопасность (см. рис. 1.7) [12, с. 38]. Разрабатываемая ИС должна удовлетворять всем основным показателям эффективности.

Информационная система, разрабатываемая в рамках данной ВКР, предназначена для выполнения расчета по выбросам в атмосферу вредных веществ транспортными средствами – ДСМ и городским автотранспортом. Данную ИС можно применять не только в целях осуществления контроля над состоянием атмосферного воздуха в период строительства и реконструкции дорог, но и для осуществления исследований в области загрязнения воздушной среды в городских условиях.

Общий принцип совместимости ИС предполагает наличие информационной, программной совместимости и совместимости программ.

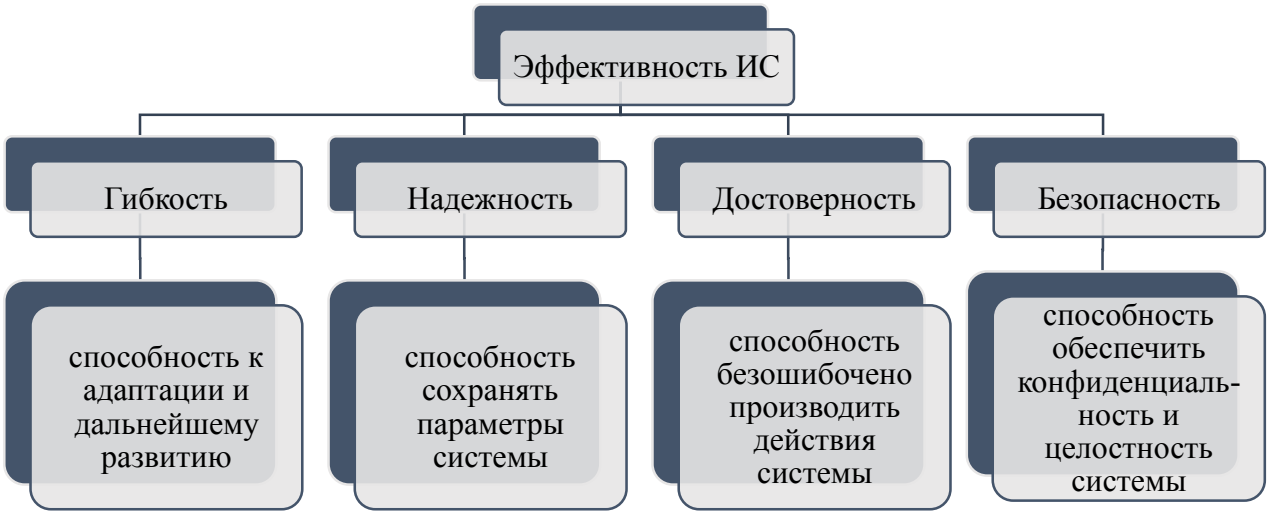


Рис. 1.7. Основные показатели эффективности ИС

Информационная совместимость ИС заключается в способности двух или более компьютеров адекватно воспринимать одинаково представленные данные, в частности, форматов представления данных.

Программная совместимость – это возможность выполнения одних и тех же программ на разных компьютерах с получением одинаковых результатов.

Совместимость программ подразумевает способность программ к взаимодействию друг с другом [38].

Разрабатываемая ИС должна быть совместима с СУБД PostgreSQL и с редактором электронных таблиц Microsoft Excel и адекватно отображать данные на других компьютерах.

Основная нормативно-правовая база разрабатываемой ИС должна:

1) удовлетворять требованиям:

- Федерального закона № 96-ФЗ «Об охране атмосферного воздуха»;

- гигиеническим нормативам ГН 2.1.6.3492-17, утвержденных 22.12.2017 г. Постановлением № 165 Главного государственного санитарного врача РФ;

2) основываться на:

- Расчетной инструкции (методики) по инвентаризации выбросов загрязняющих веществ ДСМ в атмосферный воздух (2008 г.) [6];

- Методе расчета выбросов от автотранспорта при проведении сводных расчетов для городских населенных пунктов (2015 г.) [1];

- Методе расчетов рассеивания выбросов вредных (загрязняющих) веществ в атмосферном воздухе (2017 г.) [5].

1.5 Технические требования к разрабатываемой ИС

Работу ИС обеспечивают процессы, состоящие из определенных блоков, которые в зависимости от предметной области ИС будут отличать одну систему от другой по своим функциям, архитектуре и способам реализации [29, с. 7].

Схема основных процессов разрабатываемой ИС состоит из нескольких этапов (см. рис. 1.8).

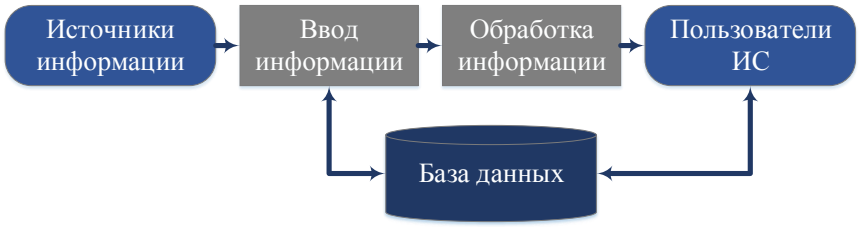


Рис. 1.8. Схема основных процессов разрабатываемой ИС

Специфика предметной области определяет для информационной системы, предъявляемые к ней технические требования.

В связи с этим разрабатываемая ИС должна реализовать следующий состав функций, соответствующих ее предметной области (см. рис. 1.9).

Состав вышеописанных функций требует решить следующие задачи:

1) для реализации ввода данных необходимо:

- организовать процедуру ввода исходных данных (территориальная принадлежность местности, метеорологические условия и параметры участка);

- организовать процедуру регистрации ДСМ, в связи, с чем разработать табличную форму для ввода основных характеристик транспортных единиц;

- организовать процедуру ввода данных об АТС, в связи, с чем разработать табличную форму, содержащую количество транспортных единиц, скорость движения, длину исследуемого участка дороги и общее время остановок на участках;

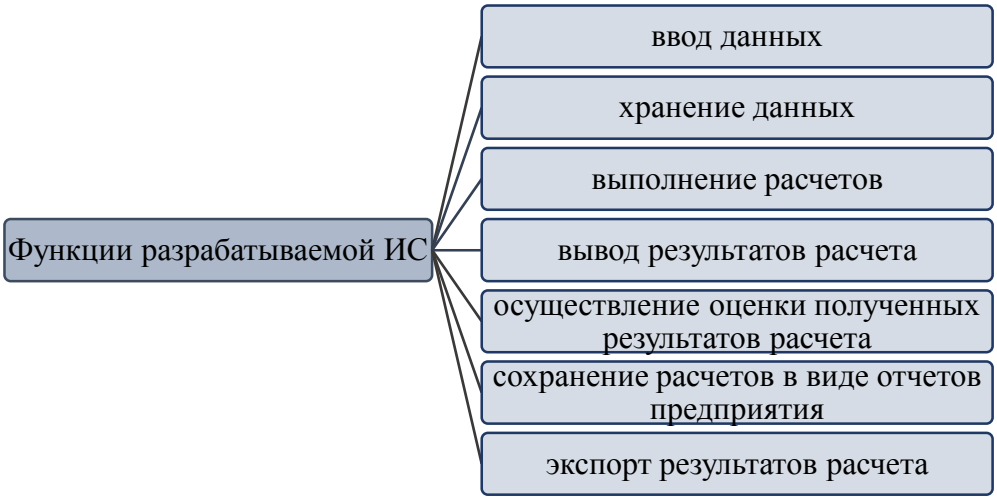


Рис. 1.9. Основные функции разрабатываемой ИС

2) организовать хранение данных, для чего:

- разработать структуру таблицы БД, содержащую данные о выбрасываемых веществах;

- разработать структуру таблицы БД, содержащую данные о ДСМ и АТС;

- разработать структуру БД, содержащую информацию об отчетах исследования;

3) организовать процесс по выполнению математических расчетов:

- по инвентаризации выбросов вредных веществ дорожно-строительной техникой (формула по Расчетной инструкции 2008);

- расчет выбросов вредных веществ городским автотранспортом (формулы по Методу 2015);

- по рассеиванию выбросов вредных веществ дорожно-строительной техникой (формула по Методу 2017);

4) организовать процесс вывода расчетных данных в таблицу:

- по количественным выбросам вредных веществ ДСМ и городским АТС;

- по общей концентрации вредных веществ на исследуемом участке;

5) реализовать оценку расчетных данных с помощью графических схем:

- по количественным выбросам загрязняющих веществ;

- по выбросам загрязняющих веществ от ДСМ и городского АТС;

- по общей концентрации выбросов;

- по рассеиванию загрязняющих веществ в атмосфере;

6) организовать процедуру сохранение расчетов в виде отчетов в БД с возможностью просмотра в ИС;

7) организовать процедуру экспорта расчетов в MS Excel.

Итак, при изучении предметной области актуальность темы исследования нашла свое подтверждение. Автомобильный транспорт, как источник загрязнения атмосферного воздуха, занимает лидирующую позицию, что свидетельствует о необходимости осуществления контроля над выбросами вредных веществ вообще и автотранспортными организациями в частности, к которым и относится исследуемое предприятие.

ООО «Россошанское ДРСУ №1» нуждается в разработке собственной ИС мониторинга выбросов загрязняющих веществ в атмосферу при строительстве и реконструкции автодорог, что значительно уменьшит расходы на подготовку сметной документации.

Основные принципы разрабатываемой ИС должны соответствовать показателям эффективности и нормативно-правовой базы. Предъявляемые к разрабатываемой ИС технические требования зависят от специфики предметной области и складываются из способности системы реализовать те или иные функции, на основании которых определяются основные задачи разработки ИС. В связи с этим в первой главе ВКР дан анализ предметной области, включающий технико-экономическую характеристику объекта исследования, раскрыта сущность задачи и обоснована необходимость разработки ИС мониторинга выбросов автотранспортом предприятия, сформулированы основные принципы программирования и технические требования, предъявляемые к разрабатываемой информационной системе.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Технологии и средства разработки ИС

В первой главе данной ВКР было определено, что разрабатываемая ИС должна быть организована на основе архитектуры «клиент-сервер» – это наиболее прогрессивная технология и стандартная для современных СУБД архитектура, которую можно реализовать с помощью 3-х моделей:

- модель доступа к удаленным данным (Remote Date Access – RDA);
- модель сервера базы данных (DateBase Server – DBS);
- модель сервера приложений (Application Server – AS) [8, с. 37].

Основные характеристика каждой из этих моделей по-разному отвечают основным критериям разработки ИС (табл. 2.1).

Таблица 2.1

Основные характеристики моделей архитектуры «клиент-сервер»

Наименование	RDA-модель	DBS-модель	AS-модель
Сложность разработки приложений	Низкая	Высокая	Высокая
Сложность администрирования	Высокая	Высокая	Высокая
Степень защиты данных	Низкая	Высокая	Высокая
Требования к характеристикам сервера	Низкие	Высокие	Высокие
Трафик, создаваемый в сети	Очень высокий	Низкий	Низкий
Сложность обновления ПО	Высокая	Низкая	Низкая
Требования к характеристикам сети	Очень высокие	Низкие	Низкие
Распределение загрузки	Есть	Есть	Есть
Требования к характеристикам рабочих станций	Очень высокие	Низкие	Низкие
Использование графического интерфейса	+	+	+
Использование символьного интерфейса	+	+	+

Разрабатываемая в рамках данной ВКР информационная система должна исключить такие недостатки существующих коммерческих предложений, как высокая стоимость внедрения и сопровождения ИС. Поэтому, учитывая возможности имеющегося на исследуемом предприятии оборудования и про-

граммного обеспечения, для реализации технологии «клиент-сервер» выбрана модель доступа к удаленным данным (RDA-модель).

Модель RDA подразумевает нахождение компонента представления и прикладного компонента на компьютере-клиенте, а компонент доступа к ресурсам полностью отделен от первых двух компонентов и находится на компьютере-сервере, на который посылаются запросы (см. рис. 2.1) [45].

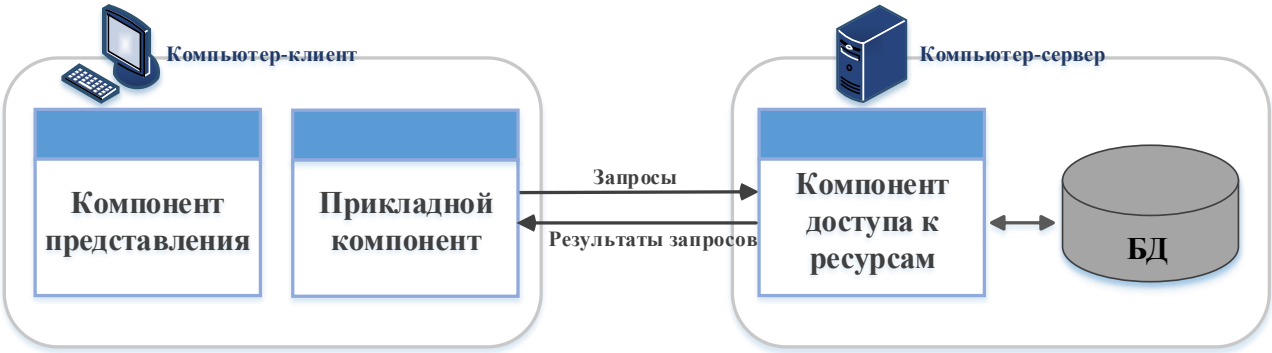


Рис. 2.1. Модель доступа к удаленным данным

Это значит, что на компьютере-клиенте совмещены и исполняются презентационная логика (Presentation Logic) и бизнес-логика (Business Logic), а функции обработка запросов внутри приложения (Database Logic) выполняются на компьютере-сервере, где обеспечивается хранение и управление базами данных. Результаты запросов с компьютер-клиента посылаются на языке SQL и возвращаются на компьютер-клиент в виде набора данных (см. рис. 2.2) [31, с. 102].

Презентационная логика (Presentation Logic) – это функции ввода (экранные формы, с которыми работает пользователь) и функции отображения данных на экране в виде результативной и справочной информации.

Бизнес-логика (Business Logic) – это реализация предметной области с помощью прикладных функций, которые определяют основные алгоритмы решения задач ИС [25, с. 91].

Преимущество RDA-модели заключается в том, что сервер БД существенно разгружается в результате совмещения презентационной и бизнес-логики на компьютере-клиенте и приобретает активную центральную функцию

по обеспечению целостности и безопасности данных. К тому же уменьшается загрузка сети, так как по ней от клиентов к серверу передаются SQL-запросы, объем которых существенно меньше файловых команд.

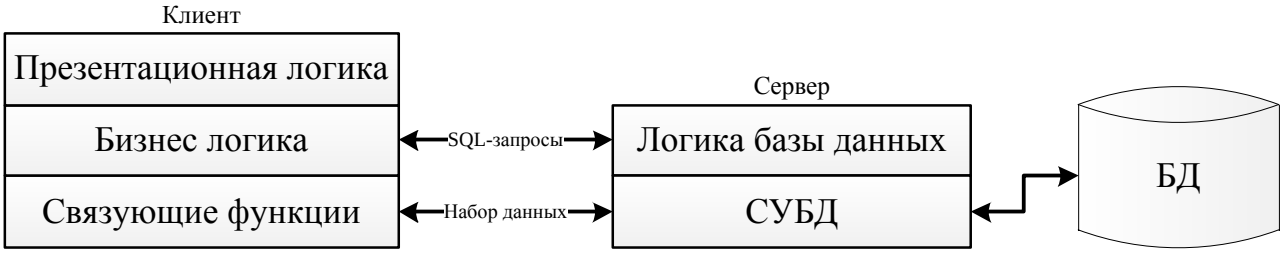


Рис. 2.2. Схема работы модели удаленного доступа

Основным достоинством данной модели является то, что взаимодействие прикладных компонентов ИС с общими данными происходит в стандартной форме: на языке SQL с помощью специального протокола ODBC (Open Database Connectivity), то есть доступ к базам данных является открытым. Это обеспечивает отсутствие каких-либо ограничений во взаимодействии вне зависимости от типа СУБД, что делает информационную систему более гибкой.

Недостатки RDA-модели касаются, в первую очередь, вопроса дублирования кода приложения, отвечающего за бизнес-функции, при наличии нескольких клиентских приложений и высокие требования, предъявляемые к клиентскому оборудованию, так как все бизнес-функции, определяемые спецификой предметной области ИС, выполняются на них.

Разрабатываемая ИС будет использоваться одним пользователем на одном компьютере-клиенте, являющимся высокотехнологичным оборудовании, поэтому загрузка сети и обработка вычислительных функций не повлияет на производительность самой системы.

Использование RDA-модели ввиду широкого набора инструментальных средств разработки обеспечивает быстрое создание одиночной ИС, характерной разрабатываемой в рамках данной ВКР системы, а одиночные ИС проектируются с помощью, так называемых настольных систем управления базами данных.

Настольное (desktop) windows-приложение – это программа, которая требует инсталляции на компьютер пользователя, запускается локально и не требует сети интернет.

Для реализации windows-приложения, проектируемой в рамках данной ВКР, выбран продукт компании Microsoft Visual Studio 2010.

MS Visual Studio – это интегрированная среда разработки (IDE), являющаяся самым эффективным решением программирования информационных систем и приложений и предоставляющая широкий спектр возможностей для разработчиков, обеспечивающих написание максимально качественного кода.

Среда Visual Studio поддерживает несколько самых популярных в мире языков программирования и облегчает создавать базы данных на SQL, а дизайн является удобным и наглядным для выполнения широкого круга операций по работе с кодом [28, с. 5].

Данная среда разработки обеспечивает эффективную и быструю отладку разрабатываемого приложения с возможностью прерывания и приостановки выполнения программы для осуществления проверки кода, а с помощью отладчика можно вычислять значения переменных программы и осуществлять их редактирование, просматривать область памяти, используемой приложением и др. [44].

В первую очередь, использование среды разработки Visual Studio основывается на том, что ее встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения помощью библиотеки классов Windows Forms.

Разработка приложений, в данном случае, осуществляется путем размещения компонентов управления из предлагаемого списка базовых элементов на форму. При этом программный код каждого компонента формируется автоматически, что является удобным и быстрым способом проектирования требуемого разработчику интерфейса. Учитывая, что Windows Forms представляет собой событийно-ориентированное приложение, то разработчику достаточно описать

требуемые действия по наступлению того или иного события в отношении компонентов Windows Forms [9, с. 18].

Windows Forms удобен для программистов еще и тем, что независимо от того, какой язык программирования выбран, используется один и тот же интерфейс программирования приложений (API). Тогда как раньше выбор языка программирования управлял выбором API [49].

В качестве языка программирования разрабатываемой ИС был выбран C++ – это универсальный язык программирования, который является наиболее оптимальным для описания предметной области разрабатываемой ИС.

Во-первых, C++ является расширением C# и поддерживает объектно-ориентированное программирование. Во-вторых, он придает каждому объекту такую характеристику, которая чётко определяет его концептуальные границы, отличая от всех других объектов (принцип абстракции данных). И, в-третьих, с помощью языка C++ данные описываются так, чтобы без изменения самого описания его можно было применить к различным типам данных (принцип обобщенного программирования) [23, с. 59].

Основными свойствами языка C++ являются компактность, быстрота выполнения и переносимость программ, а вышеописанные принципы программирования делают его более мощным, эффективным и наиболее практичным.

В отличии от других языков программирования C++ обладает некоторыми усовершенствованиями в виде параметров функций, имеющих стандартные значения, операций управления свободной памятью, наличием функций-подстановок, возможностью использования символьных констант, одноимённых функций с различными параметрами и переменных, содержащих указатель на данные в другой области памяти (ссылочных типов данных) [21, с. 5].

2.2 Обоснование выбора системы управления базой данных

Система баз данных – это «компьютеризированная система, предназначенная для хранения, переработки и выдачи информации по запросу пользова-

телей», включающая в себя «программное и аппаратное обеспечение, сами данные, а также пользователей» [19, с. 11].

Основным программным обеспечением является система управления базами данных (СУБД), которая обеспечивает взаимодействие с базой данных с помощью языка структурированных запросов (Structured Query Language, SQL), осуществляет операции с информацией непосредственно в базе данных и обеспечивает расшифровку обращенных к ней запросов [17, с. 6].

Положительное качество языка SQL заключается в том, что в интерактивном режиме доступа к БД все возможности языка запросов доступны в прикладном программировании, а в режиме выполнения прикладных программ можно осуществить отладку основных алгоритмов обработки информации для их дальнейшего использования в работающей программе, так как операторы языка SQL легко встраиваются в другие языки, используемые в создании приложений.

Основой любой БД служит модель, которая характеризует структур данных и специфику их обработки. Выделяют 5 типов моделей баз данных:

1) системы управления файлами;

2) иерархические;

3) сетевые;

4) реляционные;

5) объектно-ориентированные.

База данных разрабатываемой ИС относится к реляционным БД, то есть вся «информация представляется в виде прямоугольных таблиц, каждая из которых состоит из строк и столбцов и имеет имя уникальное внутри базы данных» [16, с. 11].

При проектировании реляционной модели данных необходимо выполнить специальные технические требования – правила Кодда [10, с. 39]:

- явное представление данных в отношениях;

- обеспечение гарантированного доступа к данным;

- реализация обработки неопределенных значений;

- описание БД в терминах реляционной модели;
- обеспечена полноты подмножества языка;
- поддержка обновлений представлений;
- использование высокоуровневого языка;
- обеспечение физической независимости данных;
- реализация логической независимости данных;
- обеспечение независимости контроля целостности;
- присутствие дистрибутивной независимости данных;
- обеспечение согласования языковых уровней.

На практике реализовать все правила Кодда затруднительно, они, скорее всего, являются теоретическими требованиями, но главным в реляционной модели данных является требование нормализации отношений – это процесс проектирования данных, который устраняет их избыточность, дублирование и обеспечивает непротиворечивость хранимых данных [30, с. 252].

Все это обеспечивает проектирование наиболее понятной структуры БД, экономит значительные ресурсы памяти компьютера, что делает БД более устойчивой, а это, в свою очередь, повышает эффективность и производительность всей ИС [36, с. 104-112].

Для проектирования СУБД разрабатываемой ИС была выбрана мощная объектно-реляционная система управления базами данных PostgreSQL – это программа с открытым исходным кодом для создания сервера БД, служащая платформой разработки приложений, требующих использования реляционной СУБД.

Преимущества PostgreSQL это:

- надежность, устойчивость и безопасность, так как она позволяет настраивать резервирование, восстановление по точке времени, производить синхронную и асинхронную репликацию, что облегчает работу с критически важными данными и позволяет осуществлять работу с использованием защищенных протоколов;

- высокая производительность, способность работать с данными в параллельном режиме, поддержка расширений пользователя, доступность;

- соответствие ANSI-SQL стандартам и наличие системы управления одновременным доступом (Multi-Version Concurrency Control, (MVCC)), обеспечивающей эффективную изоляцию транзакций («читающие транзакции никогда не блокируют пишущие транзакции, а пишущие – читающих»);

- наличие планировщика для осуществления оптимизации запросов, который умеет запускать задания в определенной последовательности, ограничивать действия окон по расписанию пользователя и др.;

- кроссплатформенность, так как запускается на всех основных платформах (Linux, UNIX, Windows) или переносится на них за счет открытого кода;

- возможность использования различных методов индексирования для ускорения запросов (GiST, SP-GiST, GIN, RUM, BRIN, Bloom);

- наличие широкого набора инструментальных средств разработки (программирование на различных языках, поддержка международных кодировок, обеспечивающих сортировку и полнотекстовый поиск, возможность хранения больших двоичных объектов (BLOB's), включая картинки, звук, или видео, качественная реализация логической составляющей (таблицы, индексы, оконные функции, триггеры, хранимые функции и др.)) [48, с. 7].

2.3 Проектирование структуры базы данных

Первым шагом при проектировании БД является инфологическое проектирование, при котором в разрабатываемой ИС выделяют основные объекты и их свойства. Второй шаг предусматривает проектирование структуры БД. Этот процесс заключается в разработке основных составляющих базы данных и принципов их взаимодействия, включающих в себя два уровня проектирования: логический и физический.

Логическое проектирование полностью зависит от предметной области ИС и заключается в определении структуры таблиц и их количестве, разработке

запросов к БД и алгоритмов их обработки, проектировании оконных форм для ввода и редактирования данных, определении типов отчетных документов, выбора метода проектирования и др.

Физическое проектирование отвечает за эффективное размещение данных и заключается в построении связей логической структурой БД с физической средой хранения, в определении параметров распределения памяти для объектов БД и построении индексов.

Наиболее распространенным методом логического проектирования является метод «сущность-связь» (Entity-Relation, ER-method), с помощью которого можно достаточно адекватно отобразить предметную область ИС [39, с. 149].

Результатом логического проектирования, в данном случае, является концептуальная схема БД, отражающая в себе:

- сведения об объектах предметной области (сущности);

- сведения о свойствах объектов (атрибутов);

- сведения об отношениях между объектами (связи).

Сущность (entity) – это предмет, который может быть идентифицирован некоторым способом, отличающим его от других предметов. В ER-модели сущность представляется в виде прямоугольника, содержащего имя сущности.

Атрибут – это свойство сущности (как, правило, атомарное), служащее для того, чтобы гарантировать отличие одного экземпляра сущности от другого и состоит из его названия, типа данных и размера.

Связь (relationship) – это ассоциация, устанавливаемая между сущностями, в виде ненаправленной линии, соединяющей два типа сущности или ведущей от сущности к ней же самой.

База данных разрабатываемой ИС содержит 11 сущностей (табл. 2.2).

Главные сущности являются наиболее значимыми, поэтому охарактеризуем их:

- сущность «отчеты об исследовании» содержит информацию о территориальной принадлежности местности, метеорологических условиях и типах источников выбросов;

- сущность «вещества» включает данные о ЗВ и их санитарно-эпидемиологических нормах;

- сущность «автомобили» описывает наличие соответствующих типом АТС на участках дороги и их характеристики;

- сущность «дорожно-строительные машины» привязывает к конкретной ДСМ государственный номер;

- сущность «парк дорожно-строительных машин» содержит данные об основных параметрах ДСМ.

Таблица 2.2

Сущности БД разрабатываемой ИС

Название сущности	Значимость
отчеты об исследовании	главные сущности
вещества	
автомобили	
дорожно-строительные машины	
парк дорожно-строительных машин	
вещества в отчете	вспомогательные таблицы
автомобили в отчете	
дорожно-строительные машины в отчете	
типы дорожно-строительных машин	таблицы константных значений
регионы	
коэффициент стратификации	

Используя описание сущностей, построим логическую модель базы данных (см. рис. 2.5).

Для полноценного функционирования и обеспечения работоспособности БД, достаточно произвести нормализацию БД к 3 нормальной форме (НФ) [36, с. 104-112].

Полученная логическая модель уже находится в 1НФ, 2НФ и 3НФ, что обеспечивает:

- атомарность всех атрибутов, отсутствие избыточности информации и однозначную идентификацию сущностей по первичному ключу – 1НФ;

- зависимость каждого не ключевого атрибута от первичного ключа в рамках конкретной сущности – 2НФ;

- независимость каждого не ключевого атрибута от другого не ключевого атрибута в рамках конкретной сущности – 3НФ.

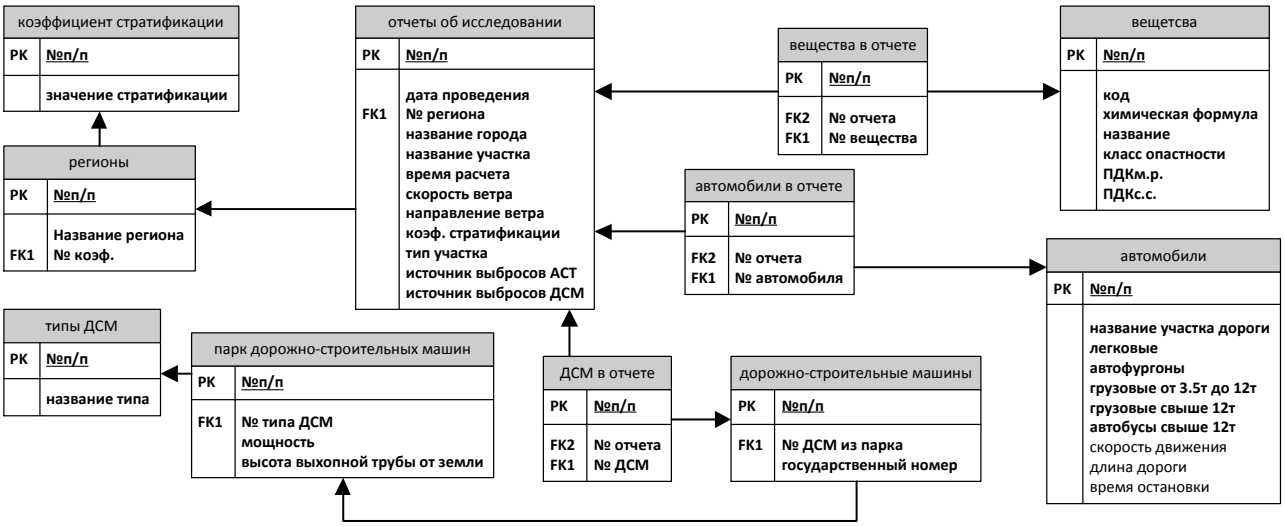


Рис. 2.5. Логическая модель БД

В ходе проектирования логической модель БД были выявлены следующие отношения между сущностями:

- так как, в сущность «отчеты об исследовании» может быть включено несколько сущностей «вещества», то между ними необходимо организовать связующую таблицу «вещества в отчете», которая реализует связи «один-ко-многим». Неявно между сущностями «отчеты об исследовании» и «вещества» реализована связь «многие-ко-многим»;

- аналогичная ситуация наблюдается между сущностями «отчеты об исследовании» – «автомобили» и «отчеты об исследовании» – «дорожно-строительные машины»;

- разрабатываемая ИС предусматривает, что пользователь может выбрать ДСМ из парка машин или указать параметры своей ДСМ, тем самым одни и те же ДСМ могут быть включены в разные отчеты, но при этом иметь разные государственные номера. Это достигается связью «один-ко-многим» между сущностями «парк дорожно-строительных машин» и «дорожно-строительные машины»;

- каждая ДСМ имеет свой тип, чтобы обеспечить масштабирование данных целесообразно хранить данные о типах в отдельной сущности «типы ДСМ», которая имеет связь «один-ко-многим» с сущностью «парк дорожно-строительных машин»;

- для выполнения расчетов необходимо иметь коэффициент стратификации воздуха, который зависит от территориальной принадлежности местности. Для этого сущность «коэффициент стратификации», которая хранит значение коэффициента стратификации воздуха, имеет связь «один-ко-многим» с сущностью «регионы», которая в свою очередь имеет связь «один-ко-многим» с сущностью «отчеты об исследовании».

После установления отношений между сущностями необходимо построить физическую модель БД, в которой каждому атрибуту сущности соответствует определенный тип данных (см. рис. 2.6).

Физическая модель является реальным представлением БД разрабатываемой ИС в СУБД PostgerSQL.

Для организации логики работы БД необходимо описать представления (виды), хранимые процедуры (функции), триггеры (триггерный функции) и генераторы (последовательности перечислений).

Просмотры предназначены для выборки данных из одной или нескольких таблиц, SQL-запросы которых хранятся в базе данных. Преимущества представлений это:

- быстрый доступ к данным;

- для каждого пользователя можно настроить свое представление;

- представления защищают пользователей от изменения структуры БД.

В разрабатываемой ИС будут использоваться 3 представления:

1) get_reports_short – предоставляет краткую информацию обо всех отчетах исследований;

2) get_substances – предоставляет информацию об исследуемых ЗВ;

3) get_trucks_park – предоставляет информацию о парке доступных ДСМ.

Хранимая процедура – это отдельная программа, написанная на процедурном языке и хранящаяся на сервере. Создав хранимую процедуру, ее можно вызвать в любое время из приложения. Хранимая процедура может принимать входные параметры и возвращать значения и наборы данных. Существует два типа хранимых процедур:

1) процедуры выборки в качестве результата своей работы возвращают набор данных либо сообщение об ошибке;

2) выполняемые процедуры осуществляют с базой данных какое-либо действие [14, с. 152].

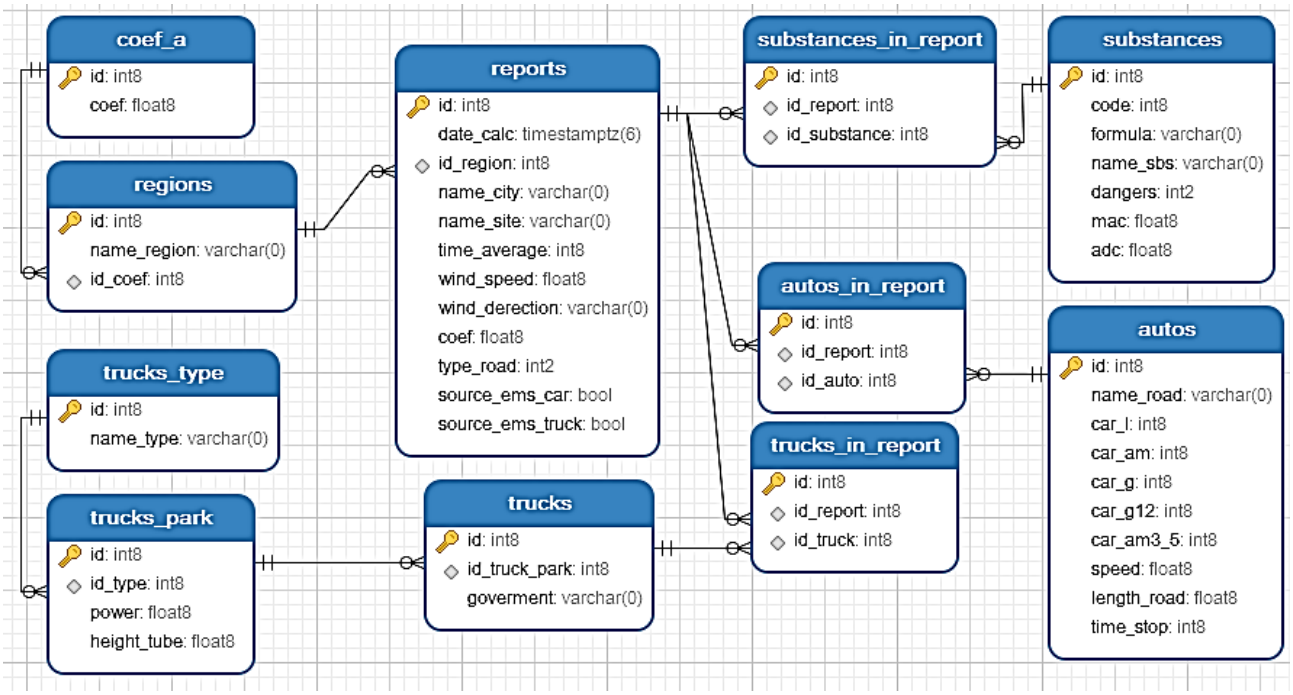


Рис. 2.6. Физическая модель БД

В разрабатываемой ИС используются все типы функций (табл. 2.3).

Если на стороне SQL-сервера перед или после вставки, обновления или удаления данных таблицы требуется выполнить дополнительные действия для этого необходимо использовать триггеры. Триггеры могут определяться как для таблиц, так и для обновляемых представлений.

Для разрабатываемой ИС определим 2 триггера:

1) save_autos_to_report – осуществляет связь сущностей «отчет об исследовании» и «автомобили» в сущности «автомобили в отчете» после вставки новых данных в сущность «автомобили»;

2) save_trucks_to_report – осуществляет связь сущностей «отчет об исследовании» и «дорожно-строительные машины» в сущности «ДСМ в отчете» после вставки новых данных в сущность «дорожно-строительные машины».

Таблица 2.3

Хранимые процедуры (функции) БД разрабатываемой ИС

Название процедуры	Описание
Процедуры выборки	
get_autos_from_report	возвращает список АТС по номеру отчету
get_trucks_from_report	возвращает список ДСМ по номеру отчету
get_substances_from_report	возвращает список исследуемых веществ по номеру отчету
get_coef_region	возвращает значение коэффициента стратификации по номеру региона
Выполняемые процедуры	
save_new_trucks_on_park	осуществляет сохранение списка данных о новых ДСМ в парк, если они есть, и возвращает их номера
save_report	осуществляет сохранение данных об отчете исследования и возвращает номер отчета

При этом вставка новых данных будет осуществляться при вызове хранимой процедуры save_report.

Генератор представляет собой механизм, создающий уникальную последовательность чисел и автоматически заполняющий заданное поле при вставке или обновлении записей. Генераторы, как правило, используются в хранимых процедурах для автоматического заполнения поля (полей), входящих в первичный ключ. Генератор является глобальным объектом для всех прочих объектов базы данных.

В разрабатываемой ИС определены 11 сущностей, но не для каждой нужен генератор, так как только в 7 сущностях будут добавляться новые данные. Генератор будет отвечать за генерацию нового значения первичного ключа. В СУБД PostgreSQL для этого в значение поля первичного ключа по-умолчанию необходимо использовать функцию nextval, которая возвращает следующее значение генератора.

Для более удобной работы с хранимыми процедурами необходимо использовать пользовательский тип данных t_truck, который будет совмещать в себе поля сущностей «дорожно-строительные машины» и «парк дорожно-строительных машин».

Таким образом, для базы данных разрабатываемой ИС определены:

- 11 таблиц (сущностей);
- 3 представления (вида);
- 6 хранимых процедур (функций);
- 2 триггера (триггерные функции);
- 7 генераторов (последовательностей перечислений);
- 1 пользовательский тип данных.

2.4 Реализация структуры базы данных

В соответствии с принципами разрабатываемой ИС необходимо реализовать клиент-серверную архитектуру, которая предполагает разделение всей логики работы СУБД на две части:

1) обслуживание данных – реализуется на стороне SQL-сервера;

2) обслуживание пользователей – реализуется клиентским программным интерфейсом [7, с. 23].

Для программной реализации структуры базы данных была выбрана утилита pgAdmin 3, которая является функциональной платформой для администрирования и разработки в СУБД PostgreSQL [52].

Первым шагом является создание перечислений (см. листинг 2.1).

Листинг 2.1. SQL-запрос создания последовательности перечисления

```
CREATE SEQUENCE count_report
INCREMENT 1
MINVALUE 0
MAXVALUE 9223372036854775807
START 0;
```

Конец листинга

С помощью перечисления будет реализована авто инкрементация ключевых полей таблиц.

Шаблон создания пользовательских типов данных в СУБД PostgreSQL похож на создание структур данных в стиле языка С (см. листинг 2.2).

Листинг 2.2. SQL-запрос создания пользовательского типа данных

```
CREATE TYPE t_truck AS (
    goverment character varying,
    type_truck character varying,
    power double precision,
    height_tube double precision);
```

Конец листинга

Для организации авто инкрементации необходимо при создании таблиц в значении ключевого поля указать следующее значение последовательности перечисления (см. листинг 2.3).

Листинг 2.3. SQL-запрос создания таблицы

```
CREATE TABLE reports (
    id bigint NOT NULL DEFAULT nextval('count_report'::regclass),
    date_calc timestamp with time zone NOT NULL DEFAULT now(),
    id_region bigint NOT NULL,
    name_city character varying NOT NULL,
    name_site character varying NOT NULL,
    time_average bigint NOT NULL DEFAULT 20,
    wind_speed double precision NOT NULL DEFAULT 0.5,
    wind_derection character varying NOT NULL,
    coef double precision NOT NULL DEFAULT 0.0,
    type_road smallint NOT NULL,
    source_ems_car boolean NOT NULL,
    source_ems_truck boolean NOT NULL,
        CONSTRAINT pr_key_report_id PRIMARY KEY (id),
        CONSTRAINT fk_key_reports_id_region FOREIGN KEY (id_region)
        REFERENCES regions (id) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION);
```

Конец листинга

Так же в листинге 2.3 продемонстрировано создание первичного и вторичного ключей.

Таблица содержит данные, которые можно представить в разном формате. Для этого необходимо создать представление, содержание которых выбирается или получается из одной или нескольких таблиц (см. листинг 2.4).

Листинг 2.4. SQL-запрос создания представления

```
CREATE VIEW get_trucks_park AS
SELECT trucks_park.id_type, trucks_type.name_type, trucks_park.power, trucks_park.height_tube
FROM trucks_park, trucks_type
WHERE trucks_park.id_type = trucks_type.id;
```

Конец листинга

В СУБД PostgreSQL для создания функций используется стандартный шаблон программной реализации, но при этом у программиста существует возможность выбора, на каком языке описать тело функции. В СУБД PostgreSQL версии 9.6 функции можно описать с помощью четырех видов языков:

- на языке запросов (SQL);

- на процедурных языках (PL/pgSQL или PL/Tcl);

- использовать внутренние функции;

- на языке C [50].

При реализации функций был использован PL/pgSQL (см. листинг 2.5).

Листинг 2.5. SQL-запрос создания функций

```
REATE OR REPLACE FUNCTION get_substances_from_report(id_rpt bigint) RETURNS SETOF bigint AS
$BODY$
DECLARE id_sbs bigint;
begin
	for id_sbs in select id_substance from substances_in_report where id_report = id_rpt
	loop return next id_sbs; end loop;
	return;
end
$BODY$ LANGUAGE plpgsql;

CREATE FUNCTION save_new_trucks_on_park(list_trucks t_truck[]) RETURNS bigint[] AS
$BODY$
DECLARE id_truck_type bigint; id_truck_park bigint;
ids_trucks_park bigint[]; truck t_truck;
begin
	foreach truck in array list_trucks loop
		SELECT trucks_park.id, trucks_type.id FROM trucks_park, trucks_type
		where trucks_type.name_type = truck.type_truck and
```

Конец листинга

Продолжение листинга 2.5.

```
                trucks_park.power = truck.power and
                trucks_park.height_tube = truck.height_tube
        into id_truck_park, id_truck_type;
        if (id_truck_park is null) then
                select id from trucks_type
                where trucks_type.name_type = truck.type_truck into id_truck_type;
                insert into trucks_park (id_type, power, height_tube)
                values(id_truck_type, truck.power, truck.height_tube);
                id_truck_park = currval('count_trucks_park');
        end if;
        ids_trucks_park=array_append(ids_trucks_park,id_truck_park);
    end loop;
    return ids_trucks_park;
end
$BODY$ LANGUAGE plpgsql
```

Конец листинга

PL/pgSQL позволяет объединить выполнение вычислений и последовательности SQL-запросов на стороне сервера БД. При этом исключается лишнее взаимодействие между клиентом и сервером и не передаются ненужные промежуточные результаты. В результате использование хранимых процедур приводит к увеличению производительности приложения [53].

В СУБД PostgreSQL перед созданием триггера необходимо сначала определить триггерную функцию, а затем привязать ее к определенной сущности. При этом указывается условие и событие, по которому будет выполнена триггерная функция (см. листинг 2.6).

Листинг 2.6. SQL-запрос создания триггера и триггерной функции

```
CREATE FUNCTION save_autos_to_report() RETURNS trigger AS
$BODY$
begin
    insert into autos_in_report(id_report, id_auto)
    values (currval('count_report'), new.id);
    return new;
end;
$BODY$ LANGUAGE plpgsql;
CREATE TRIGGER a_trig_save_autos_to_report AFTER INSERT ON autos
FOR EACH ROW EXECUTE PROCEDURE save_autos_to_report();
```

Конец листинга

При выполнении триггерных функций используются специальные переменные, с помощью которых функция работает с данными таблицы:

- new.* – содержит новые значения полей записи при вставке или обновлении данных, новые данные могут быть изменены внутри триггерной функции;

- old.* – содержит старые значения полей записи при удалении или обновлении базы данных, при этом поля записи OLD изменять нельзя.

Таким образом, в результате проектирования структуры БД были определены и описаны основные сущности, атрибуты и их связи, а также осуществлено приведение структуры БД к 3 НФ, построена физическая и логическая модели БД и выделены и описаны функциональные возможности БД. В результате программной реализации структуры БД в СУБД PostgreSQL были созданы необходимые сущности и связи между ними, посредством первичных и вторичных ключей, осуществлена реализация генераторов, представлений, функций и триггеров с помощью PL/pgSQL и SQL.

2.5 Разработка и реализация структуры windows-приложения

Приложение, управляющее некоторым ресурсом, называется сервером данных, а приложение, которое пользуется услугами сервера – клиентом.

Для windows-приложения необходимо разработать клиентский интерфейс, с помощью которого пользователь будет взаимодействовать с сервером на своем компьютере. При создании клиентского интерфейса будут использованы стандартные компоненты Windows Forms и API для работы с СУБД PostgreSQL и документами Excel.

Первым этапом при разработке windows-приложения является проектирование его структуры. Функции разрабатываемой ИС разделены между собой в отдельных формах:

1) FormMain – главная форма, в которой пользователь вносит исходные данные и осуществляет расчет загрязнений вредными веществами от автотранспорта;

2) FormConnectDB – форма, реализующая в себе подключение к БД;

3) FormReports – форма, предназначенная для отображения сохраненных отчетов исследований и их загрузке в приложение;

4) FormChartEMS_CNT – форма для отображения графических диаграмм на основе полученных расчетов;

5) FormChartCNT3D – форма для отображения трехмерной сетки рассеивания конкретного вредного вещества на основе полученных расчетов.

На основании определения структуры windows-приложения построим схему разрабатываемой ИС (см. рис. 2.7).

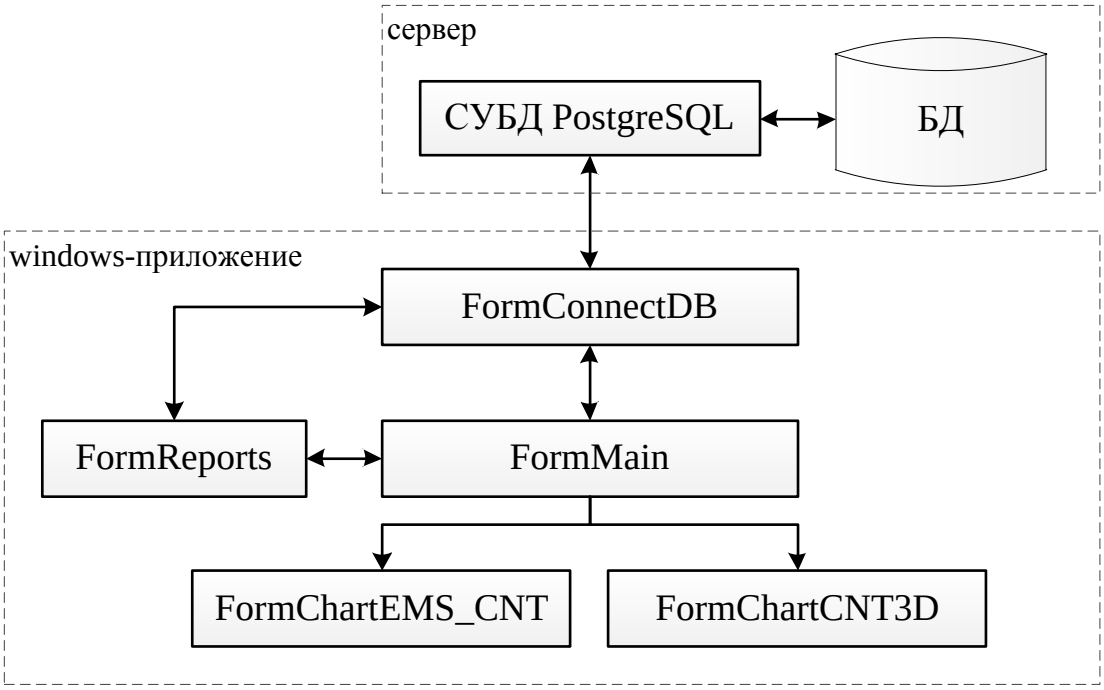


Рис. 2.7. Схема структуры разрабатываемой ИС

Из рисунка 2.7 видно, что обращение к БД осуществляется в единственной форме – FormConnectDB, что является очень удобным.

Следующим этапом является проектирование графического интерфейса windows-приложения.

Ввод исходных данных для расчета загрязнений вредными веществами осуществляется с помощью следующих компонентов (см. рис. 2.8) [33, с. 111-117]:

- ComboBox – выпадающие списки: выбор региона и направления ветра;

- TextBox – текстовые поля: ввод названия города и участка;

- NumericUpDown – счетчики: ввод числовых значений скорости ветра и времени воздействия вредными веществами;

- CheckBox – переключатели: выбор источников загрязнений;

- DataGridView – табличные контейнеры: ввод данных о ДСМ, АТС и исследуемых веществах.

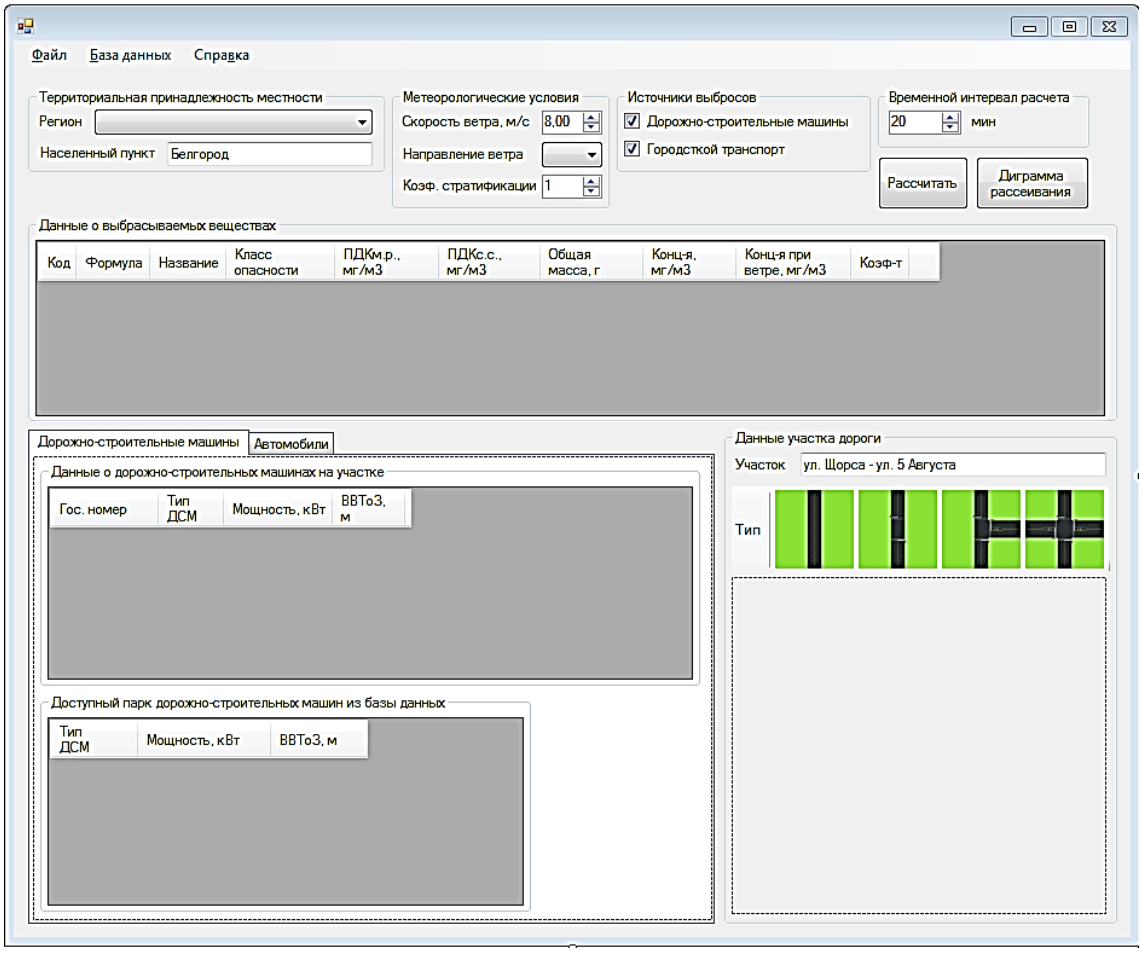


Рис. 2.8. Проектирование формы FormMain

Выполнение расчетов осуществляется по нажатию кнопки «Рассчитать».

Основные расчеты осуществляются на основании методик [1, 5, 6] по соответствующим формулам.

Для расчета массы выбрасываемых вредных веществ от ДСМ применяется формула 2.1.

$$M_{ij}=NeM_j \cdot K_{uj} \cdot \left(g_{ij}+g_{ij} \cdot \frac{K_c}{100}\right) \cdot K_d \qquad (2.1)$$

где K_{uj} – коэффициент использования мощности двигателя;

NeM_j – значение мощности двигателя ДСМ j-го типа, кВт;

g_{ij} – среднее значение выброса i-го ЗВ на единицу мощности двигателя ДСМ j-го типа, г/(кВт·ч);

K_c – коэффициент старения (коэффициент учета возраста машины);

K_d – коэффициент, учитывающий вид рабочего процесса двигателя.

Общая масса выбросов от АТС определяется из суммы расчетных масс выбросов при движении АТС каждого типа по участку дороги, определяемых по формуле 2.2, и расчетных масс выбросов АТС каждого типа в районе остановок перед светофором – по формуле 2.3 за 20 минут [34, с. 57-63].

$$M_{L_i}=\frac{L}{1200} \cdot \sum_{1}^{k}\left(M_{k,i}^{L} \cdot G_k\right) \cdot r_{V_{k,i}} \qquad (2.2)$$

$$M_{П_i}^{З}=\frac{P_ц}{60} \cdot \sum_{1}^{N_ц}\sum_{1}^{k}\left(M'_{П_{i,k}} \cdot G_k\right) \qquad (2.3)$$

где $M_{k,\,i}^{L}$ – удельный пробеговый выброс i-го загрязняющего вещества автомобилями k-й группы, г/км;

G_k – число АТС каждой из групп, проезжающих по участку автодороги в единицу времени в обоих направлениях по всем полосам движения;

$r_{V_{k,\,i}}$ – поправочный коэффициент, учитывающий среднюю скорость движения потока автотранспортных средств $V_{k,i}$ на выбранной автодороге;

$P_ц$ – продолжительность действия запрещающего сигнала светофора, сек;

$N_ц$ – число циклов действия запрещающего сигнала светофора;

$M'_{П_{i,k}}$ – удельный выброс i-го ЗВ автомобилями k-й группы, находящихся в очереди у запрещающего сигнала светофора, г/мин.

Расчет концентрации выбрасываемых ЗВ осуществляется на основании полученных рассчитанных масс по одной из формул 2.4 – 2.6.

$$c_м = \frac{A \cdot M \cdot F \cdot m \cdot n \cdot \eta}{H^2 \cdot \sqrt[3]{V_1 \cdot \Delta T}} \quad (2.4)$$

$$c_м = \frac{A \cdot M \cdot F \cdot n \cdot \eta \cdot D}{H^{4/3} \cdot 8 \cdot V_1} \quad (2.5)$$

$$c_м = \frac{A \cdot M \cdot F \cdot m' \cdot \eta}{H^{7/3}} \quad (2.6)$$

где A – коэффициент стратификации атмосферы;

M – мощность выброса, г/с;

F – коэффициент, учитывающий скорость оседания ЗВ в воздухе;

m и n – коэффициенты, учитывающие условия выброса;

η – коэффициент, учитывающий влияние рельефа местности;

H – высота источника выброса, м;

V_1 – расход ГВС, м3/с;

D – диаметр устья источника выброса, м;

ΔT – разность между температурой выбрасываемой ГВС $T_г$ и температурой атмосферного воздуха $T_в$, °C.

При этом последовательно вычисляются массы выбрасываемых веществ от ДСМ и/или АТС и общая концентрация (см. листинг 2.7).

Листинг 2.7. Выполнение расчетов масс и концентраций ЗВ

```
if (CheckBoxTruck->Checked) { ...
        for (int i = 0; i < trucks->Count - 1; i++) { ...
                for (int j = 0; j < substances->Count; j++) {
                        if (substances[j]->doCalc) {
                                trucks[i]->substance[j]->calcEMSRBMof_17_11_2006(
                                trucks[i]->getIdTypeCar(), trucks[i]->getPower(), timeAverage);  ...
                        }
                }
        }
}
if (CheckBoxAuto->Checked) {
        for (int i = 0; i < autosAtIntersection->Count; i++) { ...
                for (int j = 0; j < substances->Count; j++) {
                        if (substances[j]->doCalc) {   ...
                                autosAtIntersection[i]->calcEMSAUTOof_01_07_2015(j,
```

Конец листинга

Продолжение листинга 2.7.

```
                                        timeAverage, nameDriveSite, nameStopSite); ...
                    }
                } ...
        }
} ...
for (int i = 0; i < substances->Count; i++) {
        if (substances[i]->doCalc) { ...
                substances[i]->calcCNTof_06_06_2017(A, F, w0, H, D, dT, U); ...
        } ...
}
```

Конец листинга

Согласно RDA-модели взаимодействие с БД должно осуществляться в прикладном компоненте windows-приложения, для этого целесообразно осуществить проектировании отдельной независимой формы (см. рис. 2.9).

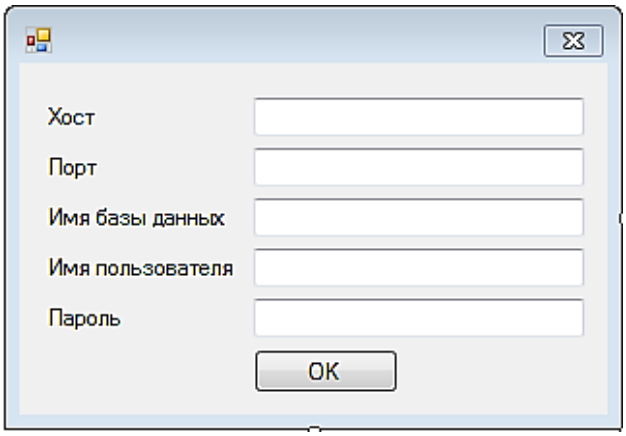


Рис. 2.9. Проектирование формы FormConnectDB

Перед запуском всей ИС необходимо сначала проверить соединение с БД, в случае успеха в любой точке программы можно использовать метод выполнения SQL-запросов, иначе работа ИС останавливается. Для взаимодействия приложений, разработанных на платформе .NET, и СУБД PostgreSQL существует Npgsql-провайдер (см. листинг 2.8) [51].

Листинг 2.8. Взаимодействие с БД

```
public: List<Array^> ^ExecuteSQL(String ^sql) {
        List<Array^> ^response = gcnew List<Array^>;
        NpgsqlConnection ^connentPG;
        try {
                connentPG = gcnew NpgsqlConnection(connectPGStr);
                connentPG->Open();
```

Конец листинга

Продолжение листинга 2.7.

```
            NpgsqlCommand ^command = gcnew NpgsqlCommand(sql,connentPG);
            NpgsqlDataReader ^reader = command->ExecuteReader();
            while(reader->Read()) {
                Array ^row = Array::CreateInstance(Object::typeid, reader->FieldCount);
                for (int i = 0; i < reader->FieldCount; i++)
                    row->SetValue((Object^)reader[i], i);
                response->Add(row);
            }
            connectDBStatus = true;
        }
        catch (Exception ^msg) {
            MessageBox::Show(msg->ToString(), "Ошибка соединения с базой данных",
MessageBoxButtons::OK, MessageBoxIcon::Warning);
            connectDBStatus = false;
            throw false;
        }
        finally { connentPG->Close(); }
        return response;
}
```

Конец листинга

Конструкция try catch finally позволяет в случае неудачного выполнения SQL-запроса выдать сообщение об ошибке.

Для просмотра и загрузки отчетов из БД так же используется отдельная форма, в которой данные отображаются в табличном виде с помощью компонента DataGridView (см. рис. 2.10).

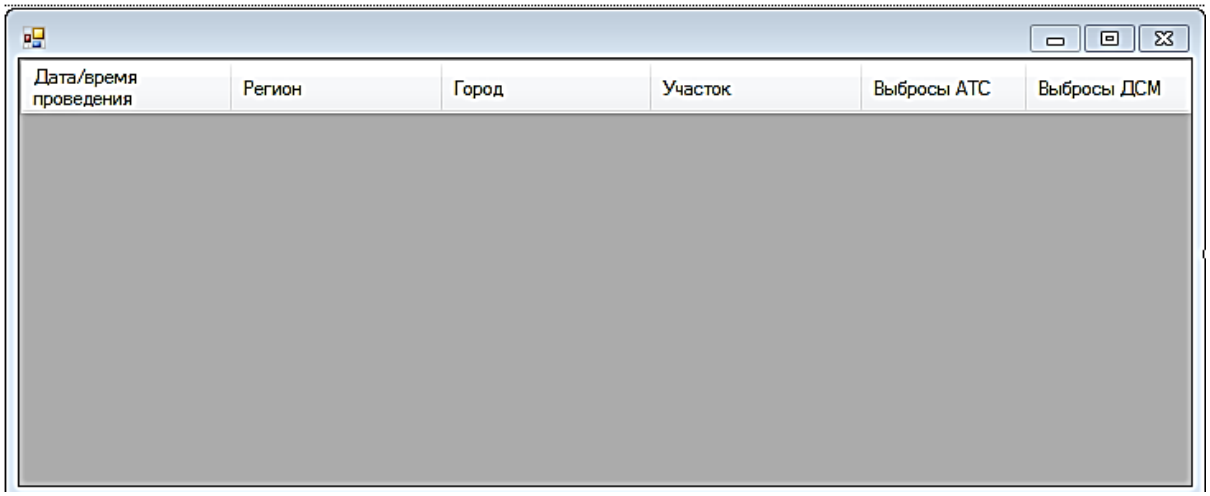


Рис. 2.10. Проектирование формы FormReports

При этом достаточно вызвать метод ExecuteSQL для получения данных об отчетах исследований (см. листинг 2.9).

Листинг 2.9. Загрузка данных об отчетах исследований из БД

```
response = formConnectDB->ExecuteSQL("SELECT * FROM get_reports_short");
DataReports->Rows->Clear();
for (int i = 0; i < response->Count; i++) {
	DataReports->Rows->Add(
		response[i]->GetValue(0)->ToString(),
		response[i]->GetValue(1)->ToString(),
		response[i]->GetValue(2)->ToString(),
		response[i]->GetValue(3)->ToString(),
		response[i]->GetValue(4)->ToString());
	if ((bool)response[i]->GetValue(5))
		DataReports["SourceAuto", DataReports->RowCount -1]->Value = true;
	if ((bool)response[i]->GetValue(6))
		DataReports["SourceTruck", DataReports->RowCount - 1]->Value = true;
}
```

Конец листинга

Построение графиков и гистограмм осуществляется в форме FormChartEMS_CNT (см. рис. 2.11).

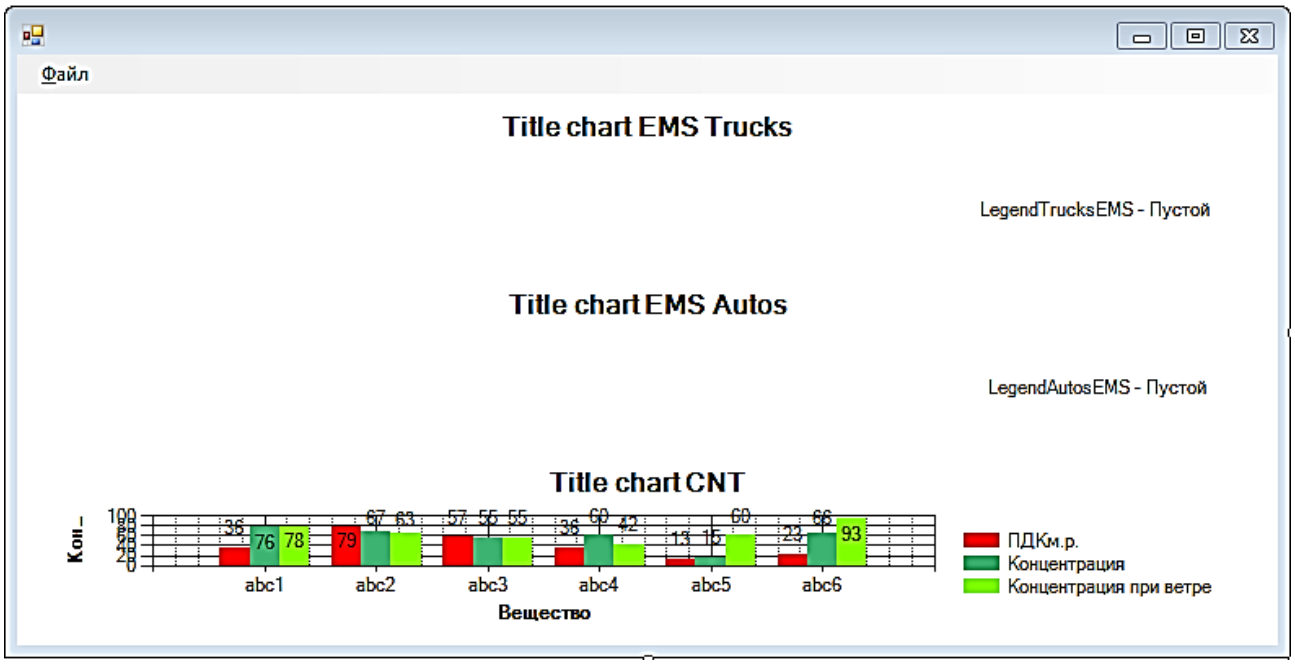


Рис. 2.11. Проектирование формы FormChartEMS_CNT

Сначала необходимо осуществить расчет масс и концентраций выбрасываемых ЗВ в форме FormMain и отправить полученные данные в форму FormChartEMS_CNT.

Так же в FormChartEMS_CNT осуществляется формирование Excel-документа – печатного варианта отчета об исследовании (см. листинг 2.10).

Листинг 2.10. Формирование Excel-документа

```
Excel::Application ^excelApp = gcnew Excel::Application();
Workbook ^excelWb = excelApp->Workbooks->Add(Type::Missing);
Worksheet ^excelWs = safe_cast<Worksheet^>(excelApp->ActiveSheet);
excelWs->Name = "Отчет об исследовании"; …
mergeExcelCells(excelWs, "B3:H3", "Предмет исследования - участок дороги"); …
mergeExcelCells(excelWs, "B4:C4", "Местоположение");
mergeExcelCells(excelWs, "D4:H4", dataReport[1]->ToString());
mergeExcelCells(excelWs, "B5:C5", "Участок");
mergeExcelCells(excelWs, "D5:H5", dataReport[2]->ToString());
mergeExcelCells(excelWs, "B6:C6", "Перекресток");
mergeExcelCells(excelWs, "D6:H6", dataReport[3]->ToString());
setFullBorderArea(excelWs, "B4:H6"); …
Excel::Chart ^chart = MakeBarChart(excelWs, startRow, startColumn, cellArea,
        DataChart->Titles["TitleChartCNT"]->Text,
        DataChart->ChartAreas["ChartAreaCNT"]->AxisX->Title,
        DataChart->ChartAreas["ChartAreaCNT"]->AxisY->Title,
        DataChart->Font->Name, DataChart->Font->Size); ...
```

Конец листинга

Для просмотра распространения рассеивания загрязняющего вещества на определенный участок местности в разрабатываемой ИС определена форма FormChartCNT3D (см. рис. 2.12).

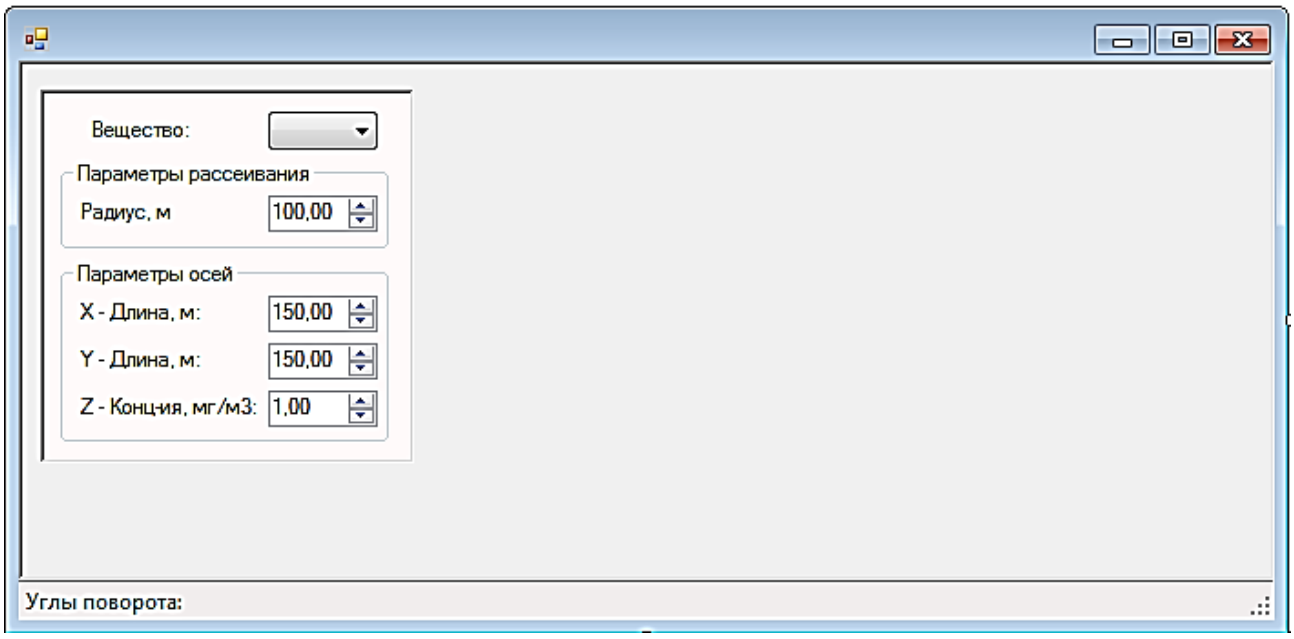


Рис. 2.12. Проектирование формы FormChartCNT3D

Для построения трехмерной сетки рассеивания загрязняющего вещества используется библиотека OpenGL, в которой описаны функции и переменные для работы с трехмерной графикой (см. листинг 2.11) [37].

Листинг 2.11. Построение сетки рассеивания с помощью OpenGL

```
glClearColor(1.0, 1.0, 1.0, 1.0);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glRotated(angleX, 1.0, 0.0, 0.0);
glRotated(angleY, 0.0, 1.0, 0.0);
glRotated(angleZ, 0.0, 0.0, 1.0); ...
glScaled(pow(zoomX, -1.0), pow(zoomY, -1.0), pow(zoomZ, -1.0)); ...
List<List<Point3D^>^> ^data = substances[indexSubstance]->getCxy(); ...
for (int i = 0; i < data->Count; i++) {
        glDrawLineStrip(data[i], substances[indexSbc]->getColor(), 1.0);
        for (int j = 0; j < data[i]->Count; j++) data[i][j]->Y *= -1.0;
        glDrawLineStrip(data[i], substances[indexSbc]->getColor(), 1.0);
} ...
double Xm = substances[indexSbc]->getXm();
double Xmu = substances[indexSbc]->getXmu();
glDrawEllips(gcnew Point3D(0.0, 0.0, 0.0), Xm, Xm, Color::Red, 2);
glDrawEllips(gcnew Point3D(0.0, 0.0, 0.0), Xmu, Xmu, Color::Green, 2);
glDrawEllips(gcnew Point3D(0.0, 0.0, 0.0), zoomX, zoomY, Color::Black, 2); ...
SwapBuffers(hDC);
```

Конец листинга

Таким образом, при разработке windows-приложения спроектирована схема разрабатываемой ИС и осуществлено графическое проектирование оконных форм с помощью Windows Forms, включающее описанием их основных компонентов, программно реализованы все функции ИС: выполняющие расчеты по формулам, построение графиков, гистограмм и трехмерной сетки рассеивания, отвечающие за взаимодействие windows-приложения с базой данных.

Итак, разрабатываемая ИС описывает технологию "клиент-сервер" с моделью доступа к удаленным данным (RDA-модель), что обеспечиваете быстрое создание одиночной ИС, выбранная СУБД PostgreSQL отвечает всем необходимым требованиям, предъявляемым к ИС, проектирование и реализация структуры БД произведено в полной мере, разработка windows-приложения осуществлена в среде разработки Microsoft Visual Studio 2010 с помощью графических форм Windows Forms на языке C++ в соответствии с описанными требованиями, взаимодействие windows-приложения и БД реализовано с использованием API Npgsql-провайдера.

ГЛАВА 3. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Описание работы windows-приложения

Работа разработанной ИС начинается с загрузки windows-приложения, которая сопровождается следующими действиями:

- осуществляется тестовое подключение к БД, если соединение не установлено, загрузка windows-приложения прекращается;

- выгрузка из БД сведений о регионах, исследуемых веществах, парке и типах ДСМ;

- отображение главной формы FormMain.

Для проведения расчетов в главной форме сначала необходимо указать регистрационные данные участка и окружающей среды в соответствующих полях:

1) «Территориальной принадлежность местности»: «Регион» – выбрать из предложенного списка, «Населенный пункт» – ввести вручную;

2) «Метеорические условия»: «Скорость ветра» – ввести вручную, «Направление ветра» – выбрать из предложенного списка и «Коэф. стратификации» – определяется относительно выбранного региона, но в некоторых случаях необходимо указывать вручную;

3) «Источники выбросов»: «Дорожно-строительные машины» и «Городской транспорт» – отметить галочкой один или несколько источников;

4) «Данные участка»: «Участок» – ввести вручную название участка, «Тип» – выбрать тип перекрестка.

Далее в таблице «Данные о выбрасываемых веществах» необходимо выбрать одно или несколько веществ, отметив его галочкой в последнем столбце.

В зависимости от выбранных источников выброса пользователю открывается соответствующая страница для ввода данных об источниках:

- источник «Дорожно-строительные машины»: в таблице «Данные о дорожно-строительных машинах на участке» пользователю необходимо указать

«Гос. номер», «Тип ДСМ» и «Мощность» каждой ДСМ или выбрать подходящую ДСМ двойным щелчком мыши из таблицы «Доступный парк дорожно-строительных машин»;

- источник «Городской транспорт»: в таблице «Данные об автомобилях на участках» пользователю необходимо указать количество соответствующих единиц городского транспорта на соответствующих участках перекрестка. Дополнительно в таблице «Данные на участках движения» требуется указать среднюю скорость транспортного потока и длину участка движения, а в таблице «Данные на участках остановок» – общее время остановок на соответствующих участках.

При этом в таблицах «Данные о дорожно-строительных машинах на участке» и «Данные об автомобилях на участках» будут добавлены столбцы соответствующие выбросам выбранных ЗВ, то есть общий расчет выбросов будет складываться из выбросов каждого отдельного источника.

В графе «Временной интервал расчета» необходимо указать время, в течение которого будет производиться исследование.

После нажатия на кнопку «Рассчитать» будет осуществлен расчет выбросов вредных веществ по каждому источнику выбросов, а так же расчет общей концентрации при нормальных и неблагоприятных метеорологических условиях вредных веществ на всем участке. При этом в отдельной форме FormChartEMS_CNT будут построены соответствующие гистограммы выбросов и общей концентрации вредных веществ.

Для построения сетки рассеивания вредных веществ необходимо нажать на кнопку «Диаграмма рассеивания» после чего откроется отдельная форма FormChartCNT3D, на панели управления которой требуется:

- выбрать ЗВ из списка, рассеивание какого необходимо построить;

- указать радиус рассеивания;

- указать масштаб рассеивания по осям Ox, Oy в метрах и Oz в мг/м^3.

Чтобы сохранить отчет об исследовании в БД необходимо в главной форме в меню «Файл» выбрать пункт «Сохранить отчет». При этом программа

проверяет все данные указанные пользователем и в случае недостатка данных сообщает об этом.

Пользователь так же может загрузить свой или чужой отчет об исследовании из БД, выбрав пункт меню «Файл»/«Загрузить отчет», после нажатия на который, открывается форма FormReports со списком сохраненных отчетов. Двойным нажатием кнопки мыши на нужном отчете будет осуществлена загрузка отчета об исследовании в программу. При этом пользователь не может перезаписать загруженный отчет, а только посмотреть результаты.

Согласно схеме структуры ИС (см. рис. 2.7) взаимодействие windows-приложения и БД осуществляется в форме FormConnectDB. Пользователь может изменить расположение БД, вызвав форму FormConnectDB в меню «База данных»/«Доступ». При этом структура сторонней БД должна соответствовать структуре БД, описанной в пункте 2.3 второй главы.

3.2 Результаты тестирования

Тестирование разработанной ИС производилось в рамках формирования сметы и оценки негативного воздействия ЗВ в атмосфере при реконструкции участка автомобильной дороги (прил. 1) предприятием ООО «Россошанское ДРСУ №1».

Реконструируемый участок автодороги имеет длину 500 м, на котором организовано двустороннее движение. Так же к участку примыкают жилые, образовательные и природные секторы.

Требуется произвести ряд исследований, по результатам которых необходимо принять соответствующие меры по снижению негативного воздействия выбрасываемых ЗВ в атмосферу, если существуют превышения.

Итак, для начала проведения исследования запустим windows-приложение и осуществим ввод данных об участке, метеорологических условиях и исследуемых ЗВ (см. рис. 3.1):

1) из выпадающего списка выберем регион – Воронежская область;

2) введем название населенного пункта – Россошь;

3) отметим источники выбросов – ДСМ и АТС;

4) введем название участка – Октябрьская ул., 138-152;

5) выберем тип дороги – Сплошная дорога;

6) отметим исследуемые ЗВ – оксид углерода, твердые частицы;

7) укажем скорость и направление ветра – 6.8 м/с, С (северный);

8) введем временной интервал расчета – 20 мин.

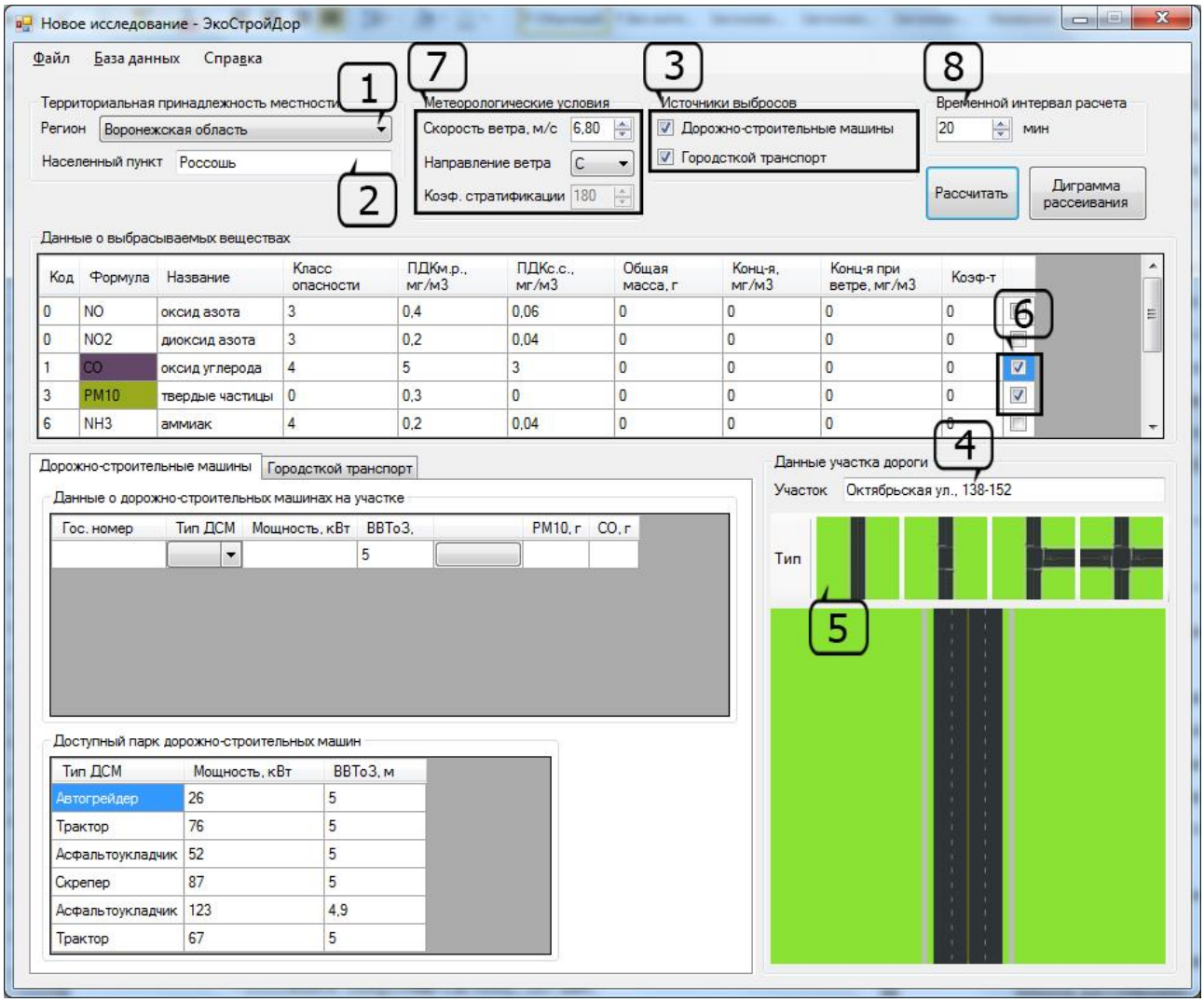


Рис. 3.1. Ввод данных об участке и метеорологических условиях

При реконструкции автодороги будут задействованы следующие ДСМ:

- 1111КР36, Скрепер Cat 621H, 304 кВт;
- 2222АМ36, Погрузчик Cat 966L, 207 кВт;
- 3333ТЕ36, Камаз 65115, 215 кВт.

Данные об используемых ДСМ необходимо ввести во вкладке «Дорожно-строительные машины». При этом для каждой ДСМ, помимо технических параметров, обязательно указывается ее государственный номер (см. рис. 3.2 (а)).

При проведении первого исследования предполагается движение потока городского автотранспорта по участку автодороги в период реконструкции, в числе которых:

- 12 легковых автомобилей (из них 4 осуществляют остановку);
- 3 автофургона (из них 1 осуществляет остановку);
- 2 автобуса (из них 1 осуществляет остановку).

Для этого во вкладке «Городской транспорт» укажем количество единиц соответствующих типов АТС, длину участка автодороги – 500 м, среднюю скорость движения транспортного потока – в 35 км/ч и общее время остановки АТС – в 600 с (см. рис. 3.2 (б)).

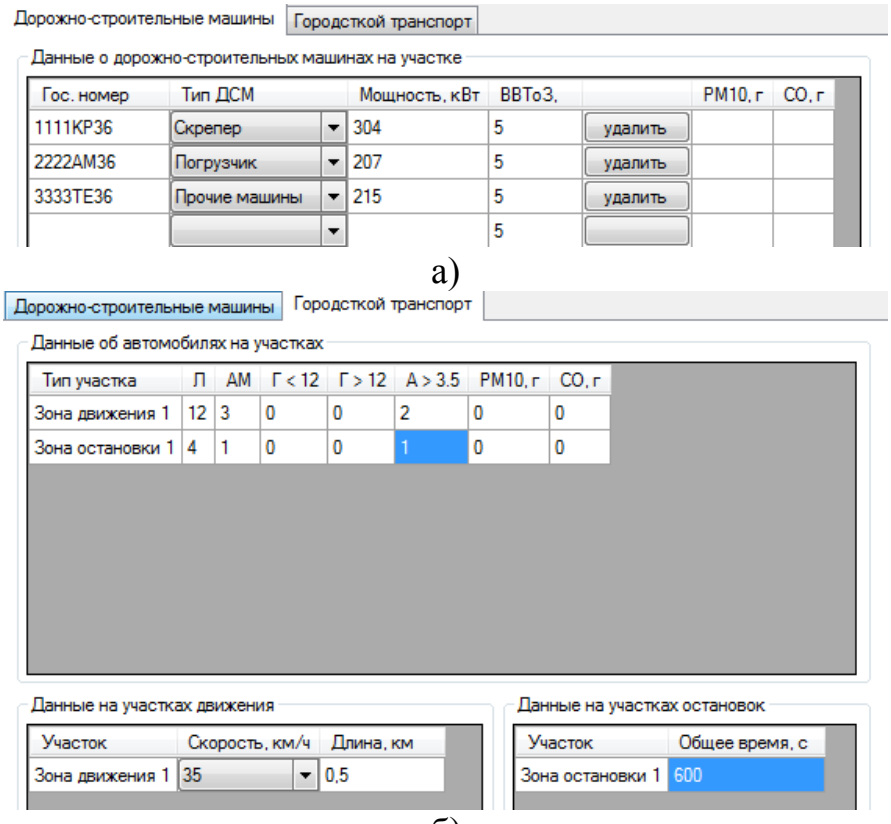


Рис. 3.2. Ввод данных об а) ДСМ; б) АТС на реконструируемом участке

Далее после нажатия на кнопку «Рассчитать» будет осуществлен расчет выбросов ЗВ от каждой ДСМ, АТС, общее значение выбросов и общая концен-

трация по исследуемым ЗВ (см. рис. 3.3(а)). На основании полученных расчетных данных будут построены соответствующие гистограммы выбросов от каждой ДСМ и АТС и общей концентрации исследуемых ЗВ (см. рис. 3.3(б)).

Данные о выбрасываемых веществах

Код	Формула	Название	Класс опасности	ПДКм.р., мг/м3	ПДКс.с., мг/м3	Общая масса, г	Конц-я, мг/м3	Конц-я при ветре, мг/м3	Коэф-т	
0	NO	оксид азота	3	0,4	0,06	0	0	0	0	☐
0	NO2	диоксид азота	3	0,2	0,04	0	0	0	0	☐
1	CO	оксид углерода	4	5	3	1659,654	24,858	2,83	4,97	☑
3	PM10	твердые частицы	0	0,3	0	327,136	4,9	0,558	16,33	☑
6	NH3	аммиак	4	0,2	0,04	0	0	0	0	☐

а)

б)

Рис. 3.3. Расчетные данные по а) выбросам и концентрации ЗВ;

б) построение гистограмм

Для демонстрации рассеивания выбранных ЗВ нажмем кнопку «Диаграмма рассеивания» и выберем вещество «PM» (см. рис. 3.4).

На диаграмме красным кругом обозначена зона, внутри которой концентрация выбранного ЗВ достигает своего максимального значения, то есть нахождение в этой зоне оказывает существенное негативное воздействие на человека и атмосферу. Зеленым кругом обозначена зона, внутри которой наблю-

дается незначительное влияние ЗВ, вне этой зоны уровень концентрации снижается по мере удаления от источника выбросов под действием ветра.

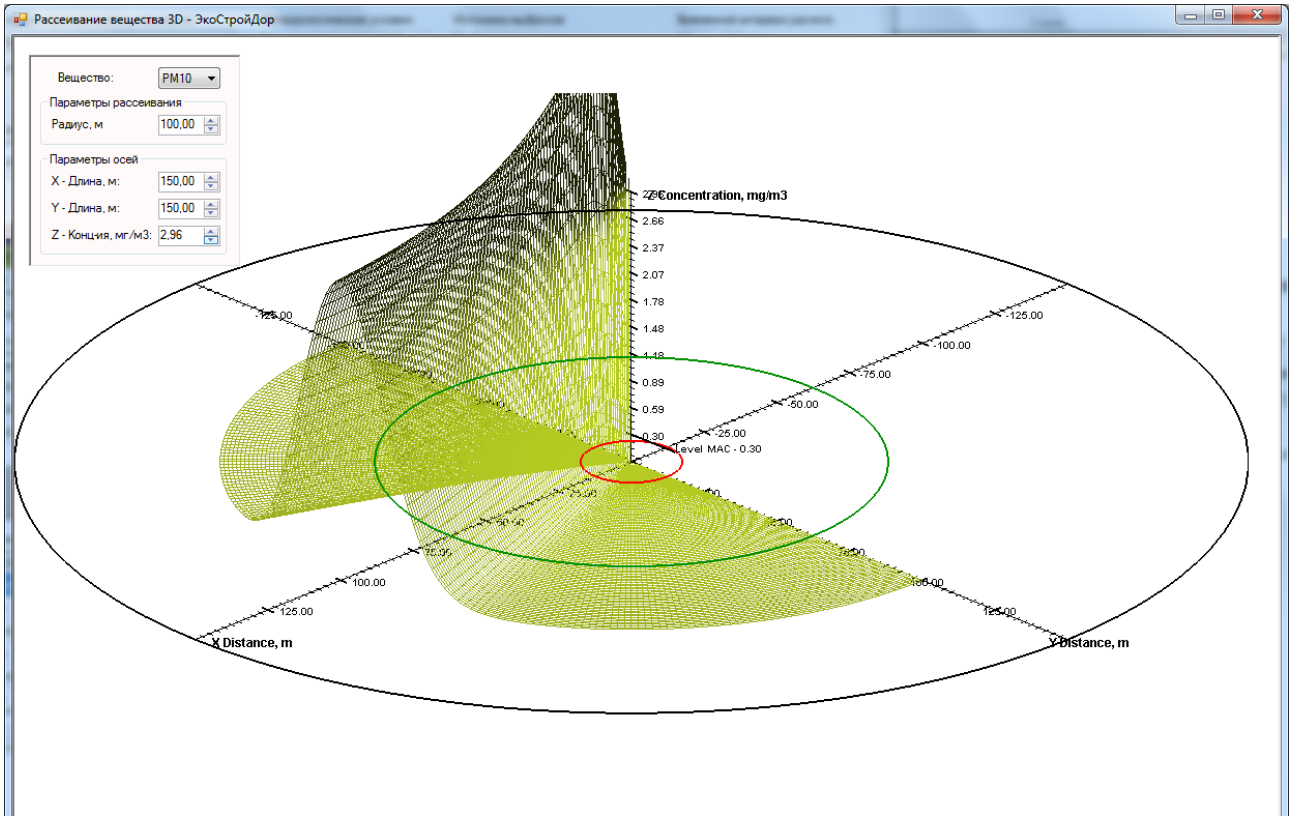


Рис. 3.4. Построение сетки рассеивания ЗВ

Из расчетов, построенных гистограмм и диаграммы рассеивания видно, что концентрация твердых частиц превышает их предельно-допустимую норму, поэтому осуществим второе исследование, но при условии, что на реконструируемом участке автодороги будут работать только ДСМ (см. рис. 3.5).

Данные о выбрасываемых веществах

Код	Формула	Название	Класс опасности	ПДКм.р., мг/м3	ПДКс.с., мг/м3	Общая масса, г	Конц-я, мг/м3	Конц-я при ветре, мг/м3	Коэф-т	
0	NO	оксид азота	3	0,4	0,06	0	0	0	0	☐
0	NO2	диоксид азота	3	0,2	0,04	0	0	0	0	☐
1	CO	оксид углерода	4	5	3	442,794	6,632	0,755	1,33	☑
3	PM10	твердые частицы	0	0,3	0	164,757	2,468	0,281	3,23	☑
6	NH3	аммиак	4	0,2	0,04	0	0	0	0	☐

Рис. 3.5. Получение расчетных данных о выбросах ЗВ от ДСМ

Из рисунка 3.5. и полученных расчетных данных можно сказать, что превышение предельно-допустимых норм ЗВ при скорости ветра 6.8 м/с не наблю-

дается. Можно сделать вывод, при реконструкции рассматриваемого участка автодороги необходимо осуществлять полное перекрытие движение городского транспорта, а так же проводить дополнительные мероприятия по снижению уровня концентрации выбрасываемых ЗВ.

Осуществим формирование печатной версии отчета об исследовании. Для этого в форме построения гистограмм в меню «Файл» выберем пункт «Экспорт в Excel» (см. рис. 3.6).

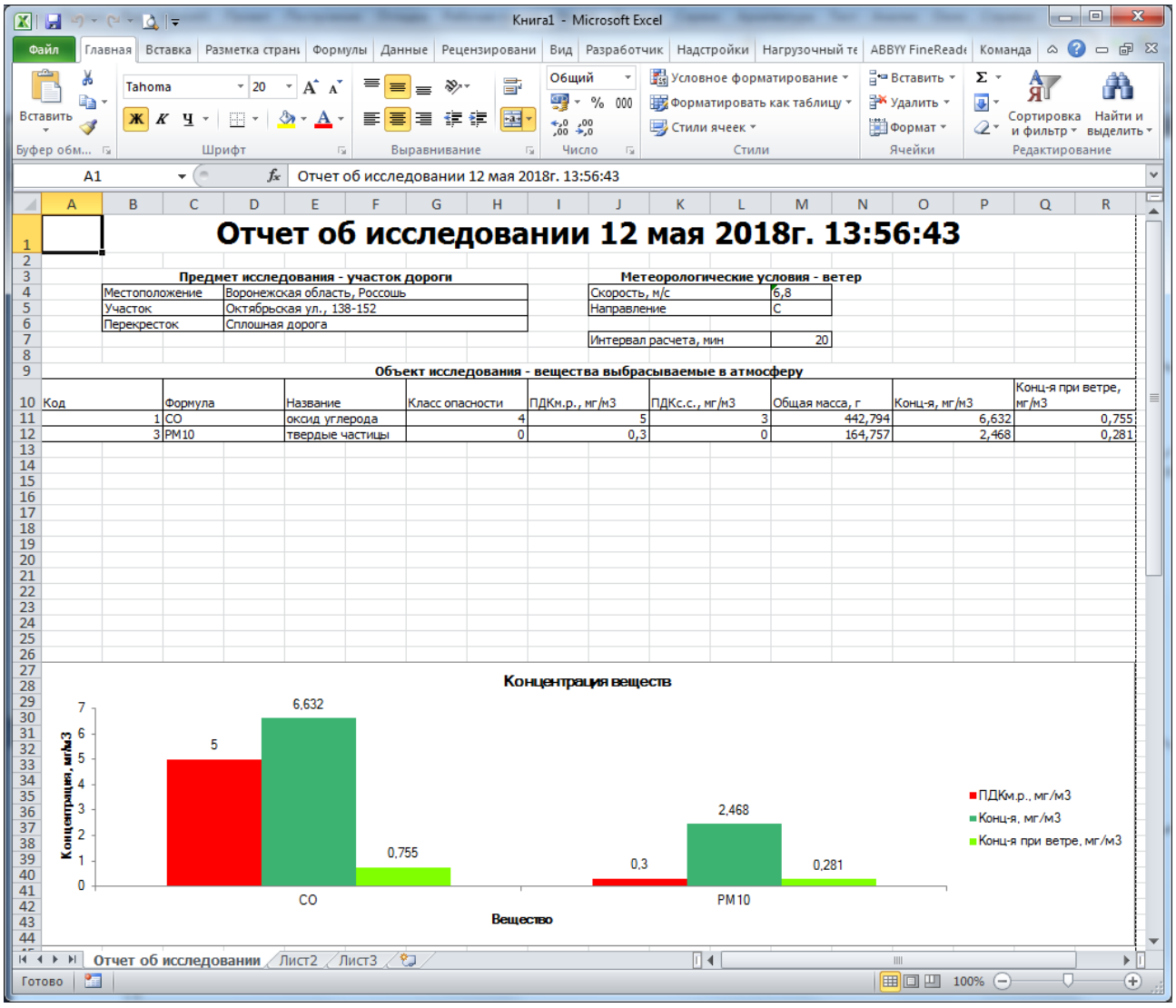
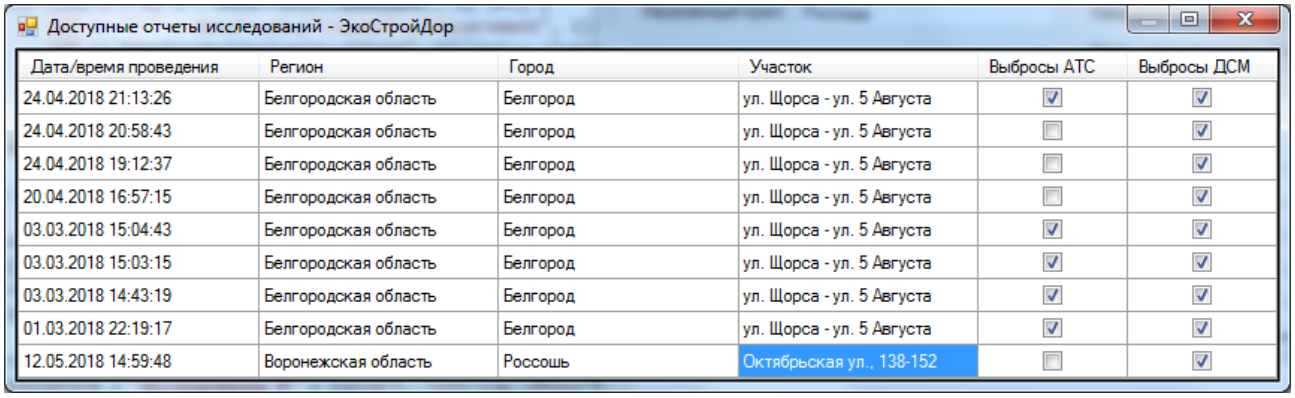


Рис. 3.6. Формирование печатной версии отчета об исследовании

Сохраним отчет об исследовании в БД, выбрав пункт «Файл»/«Сохранить отчет» на главной форме. Затем осуществим просмотр списка сохраненных от-

четов об исследовании, вызвав соответствующую форму из пункта меню «Файл»/«Загрузить отчет» (см. рис. 3.7).



Дата/время проведения	Регион	Город	Участок	Выбросы АТС	Выбросы ДСМ
24.04.2018 21:13:26	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☑	☑
24.04.2018 20:58:43	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☐	☑
24.04.2018 19:12:37	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☐	☑
20.04.2018 16:57:15	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☐	☑
03.03.2018 15:04:43	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☑	☑
03.03.2018 15:03:15	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☑	☑
03.03.2018 14:43:19	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☑	☑
01.03.2018 22:19:17	Белгородская область	Белгород	ул. Щорса - ул. 5 Августа	☑	☑
12.05.2018 14:59:48	Воронежская область	Россошь	Октябрьская ул., 138-152	☐	☑

Рис. 3.7. Просмотр сохраненных отчетов об исследовании

Таким образом, подводя итоги третьей главы, можно сказать, что она содержит подробно описание работы реализованного windows-приложения. Ключевые моменты заключаются в раскрытии определенных действий на этапе запуска приложения, в осуществлении ввода необходимых данных в определенные поля, в выборе исследуемых видов ЗВ и источников выбросов, дано подробное описание для вывода на экран пользователя гистограмм показателей выбросов и диаграммы рассеивания в процессе моделирования ситуации загрязнения атмосферы. Описаны процессы сохранения отчетов об исследовании в БД и их загрузки в windows-приложения. Тестирование ИС проводилось при разработке сметы на реконструкцию участка автомобильной дороги объектом исследования и подсчета степени негативного воздействия ЗВ на окружающую среду. Подробно описаны этапы проведения исследований: при работе только ДСМ и при условии движения автомобилей в момент реконструкции автодороги. По результатам исследований было определено, что необходимо осуществить полное перекрытие участка реконструкции автомобильной дороги. Если отсутствует возможность перекрытия участка реконструкции автодороги в городских условиях, то полученный отчет исследований используется для разработки дополнительных мероприятий для снижения уровня концентрации вредных веществ в атмосфере рабочей зоны.

ЗАКЛЮЧЕНИЕ

Итак, в процессе выполнения выпускной квалификационной работы «Разработка информационной системы мониторинга выбросов вредных веществ в атмосферу автотранспортом» на примере ООО «Россошанское ДРСУ №1» была собрана и систематизирована информация по экологической ситуации относительно загрязнения атмосферного воздуха выбросами, содержащимися в выхлопных газах автотранспорта и предложена ИС мониторинга данных выбросов для предприятия при осуществления им основного вида деятельности: строительство и реконструкция автомобильных дорог.

В этой связи основное назначение данной ИС заключается в осуществлении расчетов по инвентаризации и рассеиванию выбросов вредных веществ автотранспортом, в том числе и дорожно-строительными машинами, и подлежит использованию автотранспортными организациями, в том числе и предприятиями дорожного строительства, и специалистами, профессионально занятых в сфере экологической безопасности.

При разработке данной ИС была полностью изучена предметная область и намечены основные принципы и технические требования, предъявляемые к ней в первой главе ВКР, и которые реализованы во второй ее главе, а именно:

- описаны основные технологии и средства разработки ИС;
- спроектирована и реализована структура БД;
- разработана и реализована структура windows-приложения.

Тестирование ИС показало, что система способна осуществлять моделирование ситуации загрязнения атмосферы при реконструкции автодорог.

Разработанный программный продукт полностью соответствует российскому законодательству и отвечает все требованиям, предъявляемым к информационным системам.

Эффективность представленной ИС заключается в том, что она способна:

1) учитывать различные изменения явлений и процессов, причем некоторые изменения может осуществлять сам пользователь системы;

2) обеспечить передачу заданного количества информации наиболее экономичным способом (в виде таблиц, графических схем, облегчающих принятия в дальнейшем соответствующих мер);

3) достоверно, опираясь на действующие нормативные документы, и безошибочно быстро осуществлять все необходимые математические расчеты;

4) работать на любом персональном компьютере и не требовать подключения к сети интернет;

5) обеспечить максимально удобное взаимодействие информационной системы с ее пользователем;

Основной критерий эффективности данной ИС для предприятия – это значительное сокращения расходов на подготовку проектной документации. Инвестиции, вложенные в разработку ИС, повысят конкурентоспособность компании, так как она способствует предоставлению нужной информации, в нужное время и в нужном месте.

По теме данной ВКР были написаны и опубликованы научные статьи:

- Компьютерный анализ рассеивания выбросов в атмосферный воздух дорожно-строительной техникой;

- Компьютерная оценка воздействия выбросов в атмосферный воздух городским автотранспортом;

- Мониторинг загрязнения воздушного бассейна строительной техникой;

- Применение OpenGL в построении сетки рассеивания выбросов в атмосферу дорожно-строительной техникой.

Таким образом, цель, поставленная в начале выпускной квалификационной работе, достигнута. Намеченные в связи с ней задачи решены. Актуальность разработки ИС мониторинга выбросов вредных веществ в атмосферу автотранспортом подтверждена проведенными исследованиями экологической обстановки в стране, в связи с чем разработан программный продукт, позволяющий контролировать состояние воздушной среды при работе дорожно-строительной техники, учитывая движение городского автотранспорта по участку строительства.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Метод расчета выбросов от автотранспорта при проведении сводных расчетов для городских населенных пунктов: ГОСТ Р 56162-2014 Выбросы загрязняющих веществ в атмосферу [Электронный ресурс]. – URL: http://docs.cntd.ru/document/1200113823 (дата обращения 10.10.2017)

2. Об охране атмосферного воздуха: Федеральный закон № 96-ФЗ от 04.05.1999 г. [Электронный ресурс]. – URL: http://www.consultant.ru/document/cons_doc_LAW_22971/ (дата обращения 24.01.2018)

3. Об утверждении гигиенических нормативов ГН 2.1.6.3492-17 «Предельно допустимые концентрации (ПДК) загрязняющих веществ в атмосферном воздухе городских и сельских поселений»: Постановление Главного государственного санитарного врача РФ № 165 от 22.12.2017 г. – [Электронный ресурс]. – URL: http://docs.cntd.ru/document/556185926 (дата обращения 04.04.2018)

4. Об утверждении государственной программы РФ «Охрана окружающей среды» на 2012-2020 годы (с изменениями на 30 марта 2018 года): Постановление правительства РФ № 326 от 15.04.2014 г. [Электронный ресурс]. – URL: http://docs.cntd.ru/document/499091755

5. Об утверждении методов расчетов рассеивания выбросов вредных (загрязняющих) веществ в атмосферном воздухе: Приказ Министерства природных ресурсов и экологии РФ № 273 от 06.06.2017 г. [Электронный ресурс]. – URL: http://docs.cntd.ru/document/456074826 (дата обращения: 12.09.2017)

6. Расчетная инструкция (методика) по инвентаризации выбросов загрязняющих веществ дорожно-строительными машинами в атмосферный воздух [Электронный ресурс]. – URL: http://snipov.net/c_4654_snip_56630.html (дата обращения 12.09.2017)

7. Бураков, П.В. Введение в системы баз данных [Текст]: учеб. пособие / П.В. Бураков, В.Ю. Петров. – СПб.: Питер, 2010. – 130 с.

8. Зафиевский, А. В. Базы данных [Текст]: учебное пособие / А.В. Зафиевский, А.А. Короткин, А.Н. Лататуев. – Ярославль: ЯрГУ, 2012. – 164 с.

9. Зиборов, В.В. MS Visual C++ 2010 в среде .NET. Библиотека программиста [Текст] / В.В. Зиборов. – СПб.: Питер, 2012. – 320 с.

10. Зрюмов, Е. А. Базы данных для инженеров [Текст]: учебное пособие / Е.А. Зрюмов, А.Г. Зрюмова. – Барнаул: Изд-во АлтГТУ, 2010. – 131 с.

11. Информационные системы [Текст]: учебное пособие / Е.В. Бурцева, И.П. Рак, А.В. Селезнев и др. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2015. – 128 с.

12. Информационные системы [Текст]: учебник для вузов / Ю.С. Избачков, В.Н. Петров, А.А. Васильев, И.С. Телина. – СПб.: Питер, 2015. – 544 с.

13. Карпов, И.П. Базы данных [Текст]: учебное пособие / И.П. Карпов. – М.: Изд-во МГЭИМ, 2014. – 131 с.

14. Карпова, Т.С. Базы данных: модели, разработка, реализация [Текст]: учеб. пособие / Т.С. Карпова. – СПб.: Питер, 2015. – 304 с.

15. Когаловский, М.Р. Перспективные технологии информационных систем / М.Р. Когаловский. – М.: ДМК Пресс; М., 2013. – 288 с.

16. Кузин, А.В. Базы данных [Текст]: учебное пособие. – 5-е изд., испр. / А.В. Кузин. – М.: Издательский центр «Академия», 2012. – 320 с.

17. Кузнецов, М. В. MySQL 5 [Текст] / М.В. Кузнецов, И.В. Симдянов. – СПб.: БХВ-Петербург, 2010. – 1024 с.

18. Майорова, Л.П., Защита атмосферы [Текст]: учебно-практическое пособие / Л.П. Майорова, В.П. Тищенко, А.А. Черенцова. – Хабаровск: Изд-во Тихоокеан. гос. ун-та, 2014. – 115 с.

19. Моргунов, Е.П. Язык SQL. Базовый курс [Текст]: учебно-практ. пособие / Е.П. Моргунов. – М.: Postgres Professional, 2017. – 257 с.

20. Муравей, Л.А. Безопасность жизнедеятельности [Текст]: учебное пособие для вузов – 2-е изд., перераб. и доп. / Л.А. Муравей. – М.: ЮНИТИ-ДАНА, 2010. – 431 с.

21. Надейкина, Л.А. Программирование на языке высокого уровня. Часть I [Текст]: учебное пособие / Л.А. Надейкина. – М.: МГТУ ГА, 2012. – 84 с.

22. Петров, В.Н. Информационные системы [Текст]: учебник для вузов / В.Н. Петров. – СПб.: Питер, 2003. – 688 с.

23. Страуструп, Б. Язык программирования C++. Специальное издание. Пер. с англ. / Б. Страуструп. – М.: Издательство Бином, 2011 г. – 1136 с.

24. Толстых, А.В. Рассеивание загрязняющих веществ в атмосфере [Текст]: методические указания к практическим занятиям / А.В. Толстых. – Томск: Изд-во Том. гос. архит.-строит. ун-та, 2014. – 34 с.

25. Управление данными [Текст]: учебник / Ю.Ю. Громов, О.Г. Иванова, А.В. Яковлев, В.Г. Однолько. – Тамбов: ФГБОУ ВПО «ТГТУ», 2015. – 192 с.

26. Ушаков, В.В. Строительство автомобильных дорог [Текст]: учебник / В.В. Ушаков, В.М. Ольховиков. – М.: Кнорус, 2013. – 576 с.

27. Ушаков, С.А. Экологическое состояние территории России [Текст]: учебное пособие / С.А. Ушаков, Я.Г. Кац – М.: ИЦ «Академия», 2012. – 128 с.

28. Федоров, А.Г. Microsoft Visual Studio 2010: первое знакомство [Текст] / А.Г. Федоров. – Microsoft, 2009. – 42 с.

29. Федорова, Г.Н. Информационные системы [Текст]: учебник. – 3-е изд., стер. / Г.Н. Федорова. – М.: Издательский центр «Академия», 2013. – 208 с.

30. Фуфаев, Э. В. Базы данных [Текст]: учебное пособие. – 7-е изд., стер. / Э.В. Фуфаев, Д.Э. Фуфаев. – М.: Издательский центр «Академия», 2012. – 320 с.

31. Чистов, Д.В., Проектирование информационных систем [Текст]: учебник и практикум для академического бакалавриата / Д.В. Чистов, Ю.А. Воронцов, А.И. Уринцов. – М.: Издательство Юрайт, 2017. – 258 с.

32. Чудинов, И.Л., Информационные системы и технологии [Текст]: учебное пособие / И.Л. Чудинов, В.В. Осипова. – Томск: Изд-во Томского политехнического университета, 2013. – 145 с.

33. Ерошенко, Я.Б. Компьютерный анализ рассеивания выбросов в атмосферный воздух дорожно-строительной техникой / Я.Б. Ерошенко, К.К. Самхарадзе // Научные ведомости БелГу. Серия «Экономика информатика». – 2018 г. – Том 45 № 1. – с. 111-117

34. Ерошенко, Я.Б. Компьютерная оценка воздействия выбросов в атмосферный воздух городским автотранспортом / Я.Б. Ерошенко, К.К. Самхарадзе // Научно-технический журнал «Информационные системы и технологии». – 2018 г. – №3 (107). – с. 57-63

35. Ерошенко, Я.Б. Мониторинг загрязнения воздушного бассейна строительной техникой / Я. Б. Ерошенко, К. К. Самхарадзе // Научный журнал «Инновации в науке». – 2017 г. – №8 (69). – с. 7-11

36. Ерошенко, Я.Б. Повышение эффективности информационных систем путем нормализации баз данных / Я.Б. Ерошенко, К.К. Самхарадзе // Научный аспект. – 2017. – №3. – с. 104-112

37. Самхарадзе, К.К. Применение Opengl в построении сетки рассеивания выбросов в атмосферу дорожно-строительной техникой / К.К. Самхарадзе, Я.Б. Ерошенко // Научный результат. Серия «Информационные технологии». – 2018. – Том 3. Выпуск № 2. – с.

38. Аппаратная совместимость. – Режим доступа: http://helpiks.org/7-45982.html (дата обращения 01.03.2018)

39. База данных. Вводный курс [Электронный ресурс] / С. Кузнецов. – URL: http://citforum.ru/database/advanced_intro/27.shtml

40. Дорожник – работа на все времена. Союз дорожных организаций Воронежской области. – Режим доступа: http://sdo36.ru/news/ooo-россошанское-дрсу-№1-дорожник-р/ (дата обращения 29.01.2018)

41. Наличие автотранспортных средств по субъектам РФ [Электронный ресурс]. – URL: http://www.gks.ru/free_doc/new_site/business/trans-sv/trans_gaz.htm (дата обращения 24.01.2018)

42. Окружающая среда. Федеральная служба государственной статистики [Электронный ресурс]. – URL:

http://www.gks.ru/wps/wcm/connect/rosstat_main/rosstat/ru/statistics/environment/# (дата обращения 24.01.2018)

43. ООО «Интертрэйд». Бесплатное ПО. – Режим доступа: http://intertreid2.okis.ru/PO.html (дата обращения 15.03.2018)

44. Отладка в Visual Studio. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/sc65sadd.aspx (дата обращения 28.02.2018)

45. Различные модели технологии «Клиент – сервер». – Режим доступа: https://lektsia.com/3x547f.html (дата обращения 05.03.2018)

46. УПРЗА «ЭКОцентр – Стандарт». – Режим доступа: http://soft.eco-c.ru/products/emission (дата обращения 15.03.2018)

47. Фирма «Интеграл». Программные продукты. – Режим доступа: https://integral.ru/shop/index.html (дата обращения 15.03.2018)

48. PostgreSQL. Для начинающих [Электронный ресурс] / П. Лузанов, Е. Рогов, И. Лёвшин. – URL: https://postgrespro.ru/media/2018/01/10/introbook_v4.pdf (дата обращения 06.03.2018)

49. Windows Forms: Современная модель программирования для создания GUI приложений. – Режим доступа: http://www.codenet.ru/progr/cpp/WinForms.php (дата обращения 10.03.2018)

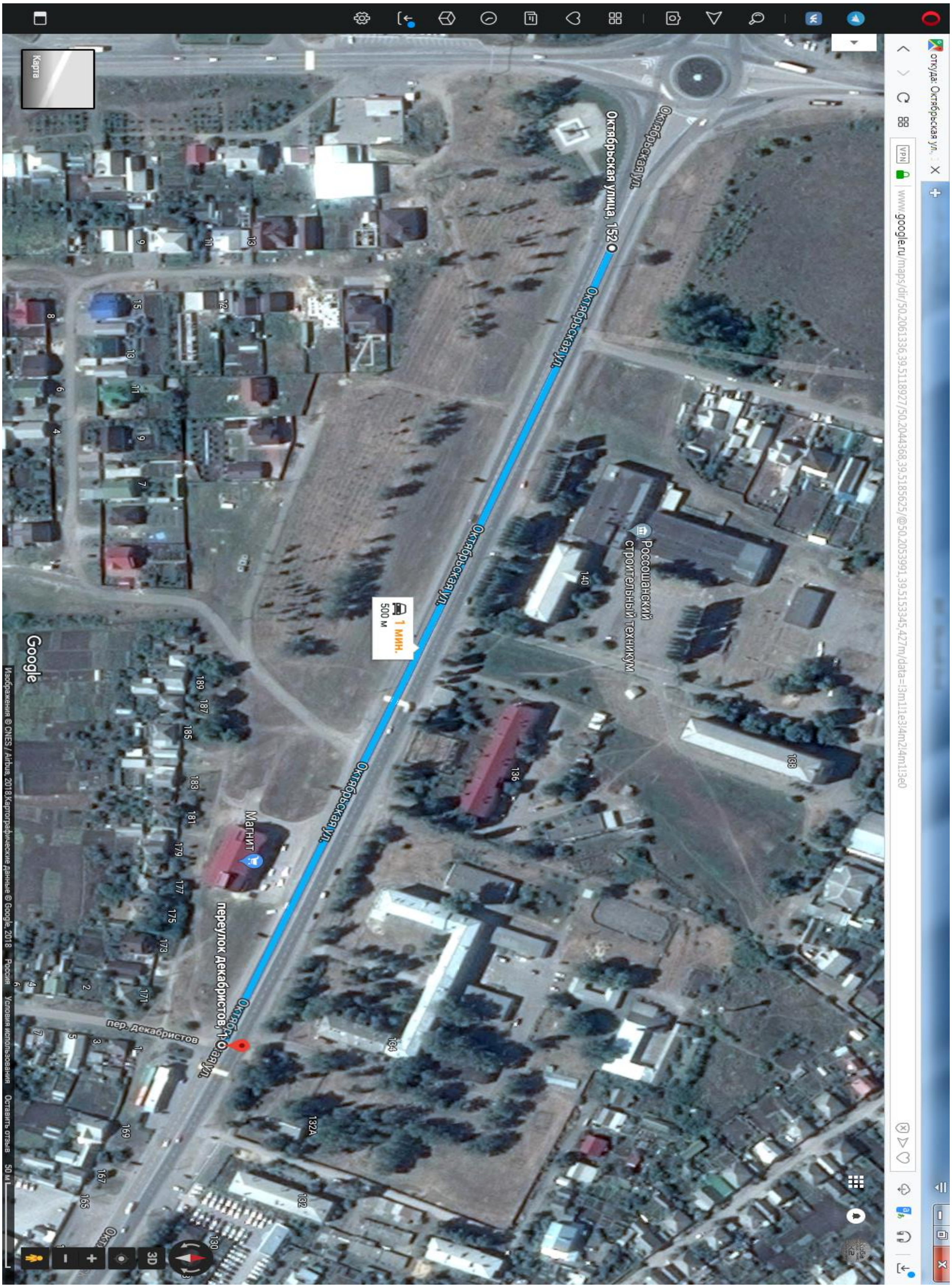
50. Extending SQL. – Режим доступа: https://postgrespro.ru/docs/postgresql/9.6/xfunc# (дата обращения 15.04.2018)

51. Npgsql - .NET Access to PostgreSQL. – Режим доступа: http://www.npgsql.org (дата обращения 10.04.2018)

52. pgAdmin 3. – Режим доступа: https://www.pgadmin.org/download/ (дата обращения 12.04.2018)

53. PL/pgSQL – SQL Procedural Language. – Режим доступа: https://postgrespro.ru/docs/postgresql/9.6/plpgsql-overview# (дата обращения 10.04.2018)

Приложение 1



Приложение 2

Разработанный код формы FormMain.h (основные обработчики событий):

```
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
	this->MaximumSize = this->Size;
	this->MinimumSize = this->Size;
	formConnectDB = gcnew FormConnectDB(titlePostfixForm);
	formConnectDB->ViewConnectDB_Load(sender, e);
	try {
		response = formConnectDB->ExecuteSQL("SELECT name_region FROM regions");
		Region->Items->Clear();
		for (int i = 0; i < response->Count; i++)
			Region->Items->Add(response[i]->GetValue(0)->ToString());

		response = formConnectDB->ExecuteSQL("SELECT * FROM get_substances");
		substances->Clear();
		for (int i = 0; i < response->Count; i++)
			substances->Add(gcnew ECOSubstance(
			Convert::ToInt32(response[i]->GetValue(0)),
			response[i]->GetValue(1)->ToString(),
			response[i]->GetValue(2)->ToString(),
			Convert::ToInt32(response[i]->GetValue(3)->ToString()),
			Convert::ToDouble(response[i]->GetValue(4)->ToString()),
			Convert::ToDouble(response[i]->GetValue(5)->ToString())));
		DataSubstances->Rows->Clear();
		for (unsigned int i = 0; i < substances->Count; i++)
			DataSubstances->Rows->Add(
			substances[i]->getIdSubstance(),
			substances[i]->getFormula(),
			substances[i]->getName(),
			substances[i]->getDangers(),
			substances[i]->getMAC(),
			substances[i]->getADC());

		response = formConnectDB->ExecuteSQL("SELECT name_type FROM trucks_type");
		for (int i = 0; i < response->Count; i++)
			ColumnTruckType->Items->Add(response[i]->GetValue(0)->ToString());

		response = formConnectDB->ExecuteSQL("SELECT * FROM get_trucks_park");
		DataTrucksPark->Rows->Clear();
		for (int i = 0; i < response->Count; i++)
			DataTrucksPark->Rows->Add(
			response[i]->GetValue(1)->ToString(),
			response[i]->GetValue(2)->ToString(),
			response[i]->GetValue(3)->ToString());
	}
	catch(bool error) { ToolStripMenuConnectDB_Click(sender, e); }
}
private: System::Void DoCalculation_Click(System::Object^ sender, System::EventArgs^ e) {
	double timeAverage = Convert::ToDouble(TimeAverage->Value) * 60;
	for (unsigned int i = 0; i < substances->Count; i++)
		substances[i]->setEmission(0.0, timeAverage);
	if (CheckBoxTruck->Checked) {
		DataTrucks->CurrentCell->Selected = false;
		for (unsigned int i = 0; i < trucks->Count - 1; i++) {
			DataTrucks_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(0, i));
			DataTrucks_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(1, i));
			DataTrucks_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(2, i));
			DataTrucks_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(3, i));
			trucks[i]->setSubstances(gcnew List<ECOSubstance^>(substances));
```

```
                        for (unsigned int j = 0; j < substances->Count; j++) {
                                if (substances[j]->doCalc) {
                                        trucks[i]->substance[j]->calcEMSRBMof_17_11_2006(
                                                trucks[i]->getIdTypeCar(), trucks[i]->getPower(), timeAverage);
                                        double ems = trucks[i]->substance[j]->getEmission();
                                        String ^substanceFormula = substances[j]->getFormula();
                                        DataTrucks[prefixNameColumn + substanceFormula, i]->Value = Con-
vert::ToString(roundTo(ems, 3));
                                        ems += substances[j]->getEmission();
                                        substances[j]->setEmission(ems, timeAverage);
                                }
                        }
                }
        }
        if (CheckBoxAuto->Checked)
                for (int i = 0, iDrive = 0, iStop = 0; i < autosAtIntersection->Count; i++) {
                        DataAutos_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(1, i));
                        DataAutos_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(2, i));
                        DataAutos_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(3, i));
                        DataAutos_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(4, i));
                        DataAutos_CellEndEdit(sender, gcnew DataGridViewCellEventArgs(5, i));
                        autosAtIntersection[i]->setSubstances(substances);
                        for (int j = 0; j < substances->Count; j++) {
                                if (substances[j]->doCalc) {
                                        if (autosAtIntersection[i]->getName()->Contains(nameDriveSite)) {
                                                autosAtIntersection[i]-
>setSpeed(Convert::ToInt32(DataAutosL["ColumnAutosLSpeed", iDrive]->Value));
                                                autosAtIntersection[i]-
>setDistance(Convert::ToDouble(DataAutosL["ColumnAutosLDistance", iDrive]->Value));
                                        }
                                        if (autosAtIntersection[i]->getName()->Contains(nameStopSite)) {
                                                autosAtIntersection[i]-
>setTimeStop(Convert::ToInt32(DataAutosP["ColumnTimeStopP", iStop]->Value));
                                        }
                                        autosAtIntersection[i]->calcEMSAUTOof_01_07_2015(j, timeAverage,
nameDriveSite, nameStopSite);
                                        double ems = autosAtIntersection[i]->substance[j]->getEmission();
                                        String ^substanceFormula = substances[j]->getFormula();
                                        DataAutos[prefixNameColumn + substanceFormula, i]->Value = Con-
vert::ToString(roundTo(ems, 3));

                                        ems += substances[j]->getEmission();
                                        substances[j]->setEmission(ems, timeAverage);
                                }
                        }
                        if (autosAtIntersection[i]->getName()->Contains(nameDriveSite)) iDrive++;
                        if (autosAtIntersection[i]->getName()->Contains(nameStopSite)) iStop++;
                }

        }
        for (unsigned int i = 0; i < substances->Count; i++)
                if (substances[i]->doCalc) {
                        List<double> ^heightsTube = gcnew List<double>();
                        double A, w0, D, dT, U; A = Convert::ToDouble(CoefA->Value);
                        F = 1.0; w0 = H = D = 0.0;              dT = 0.0000000001;
                        if (CheckBoxTruck->Checked && trucks->Count - 1 > 0) {
                                heightsTube->Add(0.0);
                                for (unsigned int j = 0; j < trucks->Count - 1; j++)
                                        heightsTube[heightsTube->Count - 1] += trucks[j]->getHeightTube();
                                heightsTube[heightsTube->Count - 1] /= trucks->Count - 1;
                        }
```

```
                        if (CheckBoxAuto->Checked && autosAtIntersection->Count > 0)
        heightsTube->Add(0.0);
                                for (int j = 0; j < autosAtIntersection->Count; j++)
                                        heightsTube[heightsTube->Count - 1] += autosAtIntersection[j]-
>getHeightTube();
                                heightsTube[heightsTube->Count - 1] /= autosAtIntersection->Count;
                        }
                        for (int i = 0; i < heightsTube->Count; i++)  H += heightsTube[i];
                        H /= (heightsTube->Count - 1) ? heightsTube->Count : 1.0;
                        U = Convert::ToDouble(WindSpeed->Value);
                        substances[i]->calcCNTof_06_06_2017(A, F, w0, H, D, dT, U);
                        double ems = substances[i]->getEmission();
                        double C = substances[i]->getConcentration();
                        double Cu = substances[i]->getConcentrationWithWind();
                        DataSubstances["ColumnSBSEMS", i]->Value = Convert::ToString(roundTo(ems, 3));
                        DataSubstances["ColumnSBSConcentration", i]->Value = Convert::ToString(roundTo(C,
3));
                        DataSubstances["ColumnSBSCNTWind", i]->Value = Convert::ToString(roundTo(Cu, 3));
                        DataSubstances["ColumnSBSGrowth", i]->Value = Convert::ToString(roundTo(C / sub-
stances[i]->getMAC(), 2));
                }
                else {
                        DataSubstances["ColumnSBSEMS", i]->Value = "0";
                        DataSubstances["ColumnSBSConcentration", i]->Value = "0";
                        DataSubstances["ColumnSBSCNTWind", i]->Value = "0";
                        DataSubstances["ColumnSBSGrowth", i]->Value = "0";
                }
        }
        List<Object^> ^dataReport = gcnew List<Object^>;
        dataReport->Add(dateTimeReport);
        dataReport->Add(Region->Text + ", " + City->Text);
        dataReport->Add(Site->Text);
        dataReport->Add(PictureIntersection->Text);
        dataReport->Add(WindSpeed->Value);
        dataReport->Add(WindDirection->Text);
        dataReport->Add(TimeAverage->Text);
        if (formChartEMS_CNT == nullptr || !formChartEMS_CNT->Visible)
                formChartEMS_CNT = gcnew FormChartEMS_CNT(titlePostfixForm);
        formChartEMS_CNT->RebuildChartAreas(DataSubstances);
        formChartEMS_CNT->dataReport = dataReport;
        if (CheckBoxTruck->Checked) formChartEMS_CNT->BuildChartsOfTrucks(DataTrucks);
        if (CheckBoxAuto->Checked) formChartEMS_CNT->BuildChartsOfAutos(DataAutos);
        formChartEMS_CNT->Show();
}

private: System::Void DoCNT3D_Click(System::Object^  sender, System::EventArgs^  e) {
         if (formChart3D == nullptr || !formChart3D->Visible)
                  formChart3D = gcnew FormChartCNT3D(titlePostfixForm);
         formChart3D->substances = gcnew List<ECOSubstance^>(substances);
         formChart3D->F = F;
         formChart3D->U = Convert::ToDouble(WindSpeed->Value);
         formChart3D->H = H;
         formChart3D->PrepareChartCNT3D();
         formChart3D->Show();
 }

private: System::Void Region_SelectedIndexChanged(System::Object^  sender, System::EventArgs^  e) {
        try {
                response = formConnectDB->ExecuteSQL("SELECT * FROM get_coef_region(" + Region-
>SelectedIndex.ToString() + ")");
                String ^coef = Convert::ToString(response[0]->GetValue(0));
                if (coef->Equals("")) {
```

```
                        MessageBox::Show(
                                "Для дальнейших расчетов\nнеобходимо учтонить коэффициент стратифика-
ции атмосферы.",  "Внимание!!!",          MessageBoxButtons::OK,
                                MessageBoxIcon::Information);
                        CoefA->Enabled = true;    CoefA->Value = 1;
                }
                else {
                        CoefA->Enabled = false;CoefA->Value = Convert::ToDecimal(coef);
                }
        }
        catch(bool error) { ToolStripMenuConnectDB_Click(sender, e); }
}
private: System::Void ToolStripMenuSaveReport_Click(System::Object^  sender, System::EventArgs^  e) {
        try {
        if (City->Text->Length == 0) throw "Необходимо указать '" + LabelCity->Text + "'.";
        if (Site->Text->Length == 0) throw "Необходимо указать '" + LabelSite->Text + "'.";
        if (!DoCalculation->Enabled) throw "Необходимо выбрать хотя бы одно исследуемое вещество.";
        if (!CheckBoxAuto->Checked && !CheckBoxTruck->Checked) throw "Необходимо выбрать '" + Group-
BoxCalcMethods->Text + "'.";
        }
        catch(String ^title) {
                MessageBox::Show(title, "Недостаточно данных!", MessageBoxButtons::OK, MessageBoxI-
con::Information);          return;
        }
        DataSubstances->CurrentCell->Selected = false;
        if (CheckBoxAuto->Checked) {
                DataAutos->CurrentCell->Selected = false;
                DataAutosL->CurrentCell->Selected = false;
                DataAutosP->CurrentCell->Selected = false;
        }
        if (CheckBoxTruck->Checked) {
                DataTrucks->CurrentCell->Selected = false;
                DataTrucksPark->CurrentCell->Selected = false;
        }
        DateTime ^dateTime = DateTime::Now;
        String ^dateTimeToDB = DateTime::Now.ToString("o");
        sql = "SELECT save_report(";
        sql +="(0, ";
        sql += "'" + dateTimeToDB + "'::timestamp with time zone, ";
        sql += Region->SelectedIndex.ToString() + ", '";
        sql += City->Text + "', '" + Site->Text + "', ";
        sql += TimeAverage->Value.ToString() + ", ";
        sql += WindSpeed->Value.ToString()->Replace(',', '.') + ", '" + WindDirection->Text + "', ";
        sql += CoefA->Value.ToString() + ", ";
        sql += DataAutosL->RowCount.ToString() + ", ";
        sql += CheckBoxAuto->Checked.ToString() + ", " + CheckBoxTruck->Checked.ToString();
        sql += ")::reports, ";
        sql += "array[";
        for (int i = 0; i < substances->Count; i++)
                if (substances[i]->doCalc) sql += Convert::ToString(i) + ", ";
        sql = sql->Remove(sql->LastIndexOf(", "), 2) + "], ";
        sql += "array[";
        if (CheckBoxAuto->Checked) {
                for (int i = 0; i < DataAutos->RowCount; i++) {
                        sql += "(0, ";
                        sql += "'" + DataAutos["ColumnTypeRoad", i]->Value->ToString() + "', ";
                        sql += DataAutos["ColumnAuto1", i]->Value->ToString() + ", ";
                        sql += DataAutos["ColumnAuto2", i]->Value->ToString() + ", ";
                        sql += DataAutos["ColumnAuto3", i]->Value->ToString() + ", ";
                        sql += DataAutos["ColumnAuto4", i]->Value->ToString() + ", ";
                        sql += DataAutos["ColumnAuto5", i]->Value->ToString() + ", ";
                        int indexDataAutosLP = i % (DataAutos->RowCount / 2);
```

```
                        if (i < DataAutos->RowCount / 2) {
                                String ^distance = DataAutosL["ColumnAutosLDistance", indexDataAutosLP]-
>Value->ToString();
                                distance = distance->Replace(",", ".");
                                sql += DataAutosL["ColumnAutosLSpeed", indexDataAutosLP]->Value-
>ToString() + ", ";
                                sql += distance + ", ";                 sql += "NULL";
                        }
                        else {
                                sql += "NULL, ";sql += "NULL, ";
                                sql += DataAutosP["ColumnTimeStopP", indexDataAutosLP]->Value->ToString();
                        }
                        sql += ")::autos, ";
                }
                sql = sql->Remove(sql->LastIndexOf(", "), 2) + "], ";
        }
        else sql += "(0, ", 0, 0, 0, 0, 0, NULL, NULL, NULL)::autos], ";
        sql += "array[";
        if (CheckBoxTruck->Checked) {
                for (int i = 0; i < DataTrucks->RowCount - 1; i++) {
                sql += "('" + DataTrucks["ColumnTruckGovernment", i]->Value->ToString() + "', '";
                        sql += DataTrucks["ColumnTruckType", i]->Value->ToString() + "', ";
                        sql += DataTrucks["ColumnTruckPower", i]->Value->ToString() + ", ";
                        sql += DataTrucks["ColumnTruckHeightTube", i]->Value->ToString() + ")::t_truck, ";
                }
                sql = sql->Remove(sql->LastIndexOf(", "), 2) + "]";
        }
        else sql += "(NULL, NULL, NULL, NULL)::t_truck]";
        sql += ")";
        try {
                response = formConnectDB->ExecuteSQL(sql);
                idReport = Convert::ToInt32(response[0]->GetValue(0));
                MessageBox::Show("Отчет успешно сохранен.", "Выполнено.", MessageBoxButtons::OK, Mes-
sageBoxIcon::Information);
                titleForm = "Исследование №" + Convert::ToString(idReport) + " от " + dateTime->ToString() + " "
+ Region->Text + ", " + City->Text;
                this->Text = titleForm + titlePostfixForm;
        }
        catch(bool error) { ToolStripMenuConnectDB_Click(sender, e); }
}

private: System::Void ToolStripMenuConnectDB_Click(System::Object^ sender, System::EventArgs^ e) {

        formConnectDB->ShowDialog();
        if (!formConnectDB->connectDBStatus)  this->Close();
}

private: System::Void ToolStripMenuOpenReport_Click(System::Object^ sender, System::EventArgs^ e) {
        DataSubstances->CurrentCell->Selected = false;
        CheckBoxAuto->Checked = false;
        CheckBoxTruck->Checked = false;
        FormReports ^viewReports = gcnew FormReports(titlePostfixForm);
        viewReports->formConnectDB = formConnectDB;
        viewReports->ShowDialog();
        idReport = viewReports->idReport;
        if (idReport > -1) {
                try {
                        response = formConnectDB->ExecuteSQL("SELECT * FROM reports WHERE id = " +
Convert::ToString(idReport));
                        Array ^report = response[0];
                        dateTimeReport = DateTime::Parse(report->GetValue(1)->ToString());
                Region->SelectedIndex = Convert::ToInt32(report->GetValue(2)->ToString());
```

```
			City->Text = report->GetValue(3)->ToString();
			Site->Text = report->GetValue(4)->ToString();
		TimeAverage->Value = Convert::ToDecimal(report->GetValue(5)->ToString());
		WindSpeed->Value = Convert::ToDecimal(report->GetValue(6)->ToString());
			WindDirection->Text = report->GetValue(7)->ToString();
			CoefA->Value = Convert::ToDecimal(report->GetValue(8)->ToString());
			ToolStripAuto_ItemClicked(sender, gcnew ToolStripItemClickedEventArgs(
				ToolStripTypeSite->Items[1 + Convert::ToInt16(
					report->GetValue(9)->ToString())]));

			CheckBoxTruck->Checked = Convert::ToBoolean(report->GetValue(11)->ToString());
			CheckBoxAuto->Checked = Convert::ToBoolean(report->GetValue(10)->ToString());
			titleForm = "Исследование №" + Convert::ToString(idReport) + " от " + dateTimeReport-
>ToString() + " " + Region->Text + ", " + City->Text;
			this->Text = titleForm + titlePostfixForm;
			for (int i = 0; i < DataSubstances->RowCount; i++) {
				if ((bool)DataSubstances["ColumnSBSDoCalc", i]->EditedFormattedValue) {
					DataSubstances["ColumnSBSDoCalc", i]->Value = false;
					DataSubstances_CellContentClick(sender, gcnew DataGridViewCellEven-
tArgs(10, i));
				}
			}
			response = formConnectDB->ExecuteSQL("SELECT * FROM
get_substances_from_report(" + Convert::ToString(idReport) + ")");
			for (int i = 0; i < response->Count; i++) {
				int rowIndex = Convert::ToInt32(response[i]->GetValue(0)->ToString());
				DataSubstances["ColumnSBSDoCalc", rowIndex]->Value = true;
				DataSubstances_CellContentClick(sender, gcnew DataGridViewCellEventArgs(10,
rowIndex));
			}
			if (CheckBoxAuto->Checked) {
				response = formConnectDB->ExecuteSQL("SELECT get_autos_from_report(" +
Convert::ToString(idReport) + ")");
				DataAutos->Rows->Clear();	DataAutosL->Rows->Clear();
				DataAutosP->Rows->Clear();
				for (int i = 0; i < response->Count; i++) {
					Array ^dataAuto = response[i]->GetValue(0)->ToString()
						->Replace("(", "")->Replace(")", "")->Replace("\"", "")
						->Split(',');
					DataAutos->Rows->Add(
						dataAuto->GetValue(1)->ToString(),
						dataAuto->GetValue(2)->ToString(),
						dataAuto->GetValue(3)->ToString(),
						dataAuto->GetValue(4)->ToString(),
						dataAuto->GetValue(5)->ToString(),
						dataAuto->GetValue(6)->ToString());

					if (dataAuto->GetValue(1)->ToString()->Contains(nameDriveSite))
						DataAutosL->Rows->Add(
						dataAuto->GetValue(1)->ToString(),
						dataAuto->GetValue(7)->ToString(),
					dataAuto->GetValue(8)->ToString()->Replace(".", ","));

					if (dataAuto->GetValue(1)->ToString()->Contains(nameStopSite))
						DataAutosP->Rows->Add(
						dataAuto->GetValue(1)->ToString(),
						dataAuto->GetValue(9)->ToString());
				}
			}
			if (CheckBoxTruck->Checked) {
				response = formConnectDB->ExecuteSQL("SELECT get_trucks_from_report(" +
Convert::ToString(idReport) + ")");
```

```
                                        DataTrucks->Rows->Clear();
                                        DataTrucks->AllowUserToAddRows = false; trucks->Clear();
                                        DataTrucks->AllowUserToAddRows = true;
                                        for (int i = 0; i < response->Count; i++) {
                                                Array ^dataTruck = response[i]->GetValue(0)->ToString()
                                                        ->Replace("(", "")->Replace(")", "")->Replace("\"", "")
                                                        ->Split(',');
                                                DataTrucks->Rows->Add(
                                                        dataTruck->GetValue(0)->ToString(),
                                                        dataTruck->GetValue(1)->ToString(),
                                                        dataTruck->GetValue(2)->ToString(),
                                                        dataTruck->GetValue(3)->ToString());
                                        }
                                }
                                MessageBox::Show("Отчет успешно загружен.", "Выполнено.", MessageBoxBut-
tons::OK, MessageBoxIcon::Information);
                                ToolStripMenuSaveReport->Enabled = false;
                        }
                        catch(bool error) { ToolStripMenuConnectDB_Click(sender, e); }
                }
}
```

Разработанный код формы FormConnectDB.h (основные обработчики событий):

```
public: System::Void ViewConnectDB_Load(System::Object^  sender, System::EventArgs^  e) {
        this->MaximumSize = this->Size;  this->MinimumSize = this->Size;
        StreamReader ^file = File::OpenText(fileNameConnectDB); String ^str;          int i = 0;
        while((str = file->ReadLine()) != nullptr) {
                switch(i++) {
                case 0: { Host->Text = str; break; }
                case 1: { Port->Text = str; break; }
                case 2: { DBName->Text = str; break; }
                case 3: { UserName->Text = str; break; }
                case 4: { Password->Text = str; break; }
                default:;
                }
        }
        file->Close();
        connectPGStr = "Server=" + Host->Text + ";" + "Port=" + Port->Text + ";" +
                "User Id=" + UserName->Text + ";" +"Password=" + Password->Text + ";" +
                "Database=" + DBName->Text + ";";
}
public: System::Void Connect_Click(System::Object^  sender, System::EventArgs^  e) {
        ViewConnectDB_Load(sender, e);
        try {
                List<Array^> ^response = ExecuteSQL("SELECT now()");
                StreamWriter ^file = gcnew StreamWriter(fileNameConnectDB);
                file->WriteLine(Host->Text); file->WriteLine(Port->Text);
                file->WriteLine(DBName->Text); file->WriteLine(UserName->Text);
                file->WriteLine(Password->Text);
                file->Close();      this->Close();
        }
        catch (bool error) { }
}
public: List<Array^> ^ExecuteSQL(String ^sql) {
        List<Array^> ^response = gcnew List<Array^>;
        NpgsqlConnection ^connentPG;
        try {
                connentPG = gcnew NpgsqlConnection(connectPGStr);
                connentPG->Open();
                NpgsqlCommand ^command = gcnew NpgsqlCommand(sql, connentPG);
                NpgsqlDataReader ^reader = command->ExecuteReader();
```

```
                while(reader->Read()) {
                Array ^row = Array::CreateInstance(Object::typeid, reader->FieldCount);
                for (int i = 0; i < reader->FieldCount; i++) row->SetValue((Object^)reader[i], i);
                response->Add(row);
                }
                connectDBStatus = true;
        } catch (Exception ^msg) {
                MessageBox::Show(msg->ToString(), "Ошибка соединения с базой данных",
MessageBoxButtons::OK, MessageBoxIcon::Warning);        connectDBStatus = false;  throw false;
        } finally { connentPG->Close(); }
        return response;
}
```

Разработанный код формы FormReports.h (основные обработчики событий):

```
private: System::Void ViewReports_Load(System::Object^  sender, System::EventArgs^  e) {
        response = formConnectDB->ExecuteSQL("SELECT * FROM get_reports_short");
        DataReports->Rows->Clear();
        for (int i = 0; i < response->Count; i++) {
                DataReports->Rows->Add(
                response[i]->GetValue(0)->ToString(), response[i]->GetValue(1)->ToString(),
                response[i]->GetValue(2)->ToString(), response[i]->GetValue(3)->ToString(),
                response[i]->GetValue(4)->ToString());
                if ((bool)response[i]->GetValue(5))
                        DataReports["ColumnReportSourceAuto", DataReports->RowCount - 1]->Value = true;
                if ((bool)response[i]->GetValue(6))
                        DataReports["ColumnReportSourceTruck", DataReports->RowCount - 1]->Value = true;
        }
}
private: System::Void DataReports_CellDoubleClick(System::Object^  sender, Sys-
tem::Windows::Forms::DataGridViewCellEventArgs^  e) {
        int indexRow = e->RowIndex;       int indexColumn = e->ColumnIndex;
        if (indexRow > -1 && indexColumn > -1) {
                idReport = Convert::ToInt32(DataReports["ColumnReportId", indexRow]->Value);
                this->Close();
        }
}
```

Разработанный код формы FormChartEMS_CNT.h (основные обработчики событий):

```
private: Excel::Chart ^ MakeBarChart(Worksheet ^ws, int startRow, int startColumn, String ^regionRange,
        String ^title, String ^titleAxisX, String ^titleAxisY,   String^ nameFont, Single sizeFont) {
        Range ^range = ws->Range[regionRange, Type::Missing];  double xPos = 0.0;
        for (int i = 1; i < startColumn; i++)
        xPos += Convert::ToDouble(safe_cast<Range^>(ws->Columns[i, Type::Missing])->Width);       double yPos
= 0.0;
        for (int i = 1; i < startRow; i++)
        yPos += Convert::ToDouble(safe_cast<Range^>(ws->Rows[i, Type::Missing])->Height);
        double width = -1.0;
        Range ^range2 = ws->Range[regionRange->Split(';')[1], Type::Missing];
        for (int i = 1; i <= 18; i++)
                width += Convert::ToDouble(safe_cast<Range^>(ws->Columns[i, Type::Missing])->Width);
        double height = 0.0;
        for (int i = startRow; i < startRow + 18; i++)
        height += Convert::ToDouble(safe_cast<Range^>(ws->Rows[i, Type::Missing])->Height);
        ChartObjects ^chartObjs = safe_cast<ChartObjects^>(ws->ChartObjects(Type::Missing));
        ChartObject ^chartObj = chartObjs->Add(xPos, yPos, width, height);
        Excel::Chart ^chart = chartObj->Chart;
```

```
        chart->ChartWizard( range, Constants::xlColumn, Type::Missing, Excel::XlRowCol::xlColumns,
        2, 1, true, title, titleAxisX, titleAxisY, Type::Missing); chart-
>ApplyDataLabels(Excel::XlDataLabelsType::xlDataLabelsShowValue,
                false, false, false, false, false, true, false, false, false);
        chart->ChartArea->Format->TextFrame2->TextRange->Font->Name = nameFont;
        chart->ChartArea->Format->TextFrame2->TextRange->Font->Size = sizeFont;
        return chart;
}
private: System::Void ExportToExcel_Click(System::Object^ sender, System::EventArgs^ e) {
        Excel::Application ^excelApp = gcnew Excel::Application();
        Workbook ^excelWb = excelApp->Workbooks->Add(Type::Missing);
        Worksheet ^excelWs = safe_cast<Worksheet^>(excelApp->ActiveSheet);
        excelWs->Name = "Отчет об исследовании"; excelWs->StandardWidth = 7;
        excelWs->PageSetup->Orientation = Excel::XlPageOrientation::xlLandscape;
        excelWs->PageSetup->TopMargin = excelApp->CentimetersToPoints(1.91);
        excelWs->PageSetup->BottomMargin = excelApp->CentimetersToPoints(1.91);
        excelWs->PageSetup->LeftMargin = excelApp->CentimetersToPoints(0.64);
        excelWs->PageSetup->RightMargin = excelApp->CentimetersToPoints(0.64);

        int startRow = 1; int startColumn = 1;
        safe_cast<Range^>(excelWs->Cells)->Font->Name = "Tahoma"; safe_cast<Range^>(excelWs->Cells)-
>Font->Size = 8;
        String ^cellArea = getExcelAreaAdress(startRow, startColumn, startRow, 18);
        excelWs->PageSetup->PrintTitleRows = "A1:R1";
        DateTime ^dateTimeReport = (dataReport[0] == nullptr) ? DateTime().Now : (DateTime^)dataReport[0];
        mergeExcelCells(excelWs, "A1:R1", "Отчет об исследовании " + dateTimeReport->ToString("dd MMMM
yyyyг. HH:mm:ss"));
        safe_cast<Range^>(excelWs->Range["A1:R1", Type::Missing])->Font->Size = 20;
        safe_cast<Range^>(excelWs->Range["A1:R1", Type::Missing])->Font->Bold = true;
        safe_cast<Range^>(excelWs->Range["A1:R1", Type::Missing])->HorizontalAlignment = Ex-
cel::XlHAlign::xlHAlignCenter;
        safe_cast<Range^>(excelWs->Range["A1:R1", Type::Missing])->VerticalAlignment = Ex-
cel::XlVAlign::xlVAlignCenter;
        mergeExcelCells(excelWs, "B3:H3", "Предмет исследования - участок дороги");
        safe_cast<Range^>(excelWs->Range["B3:H3", Type::Missing])->Font->Bold = true;
        safe_cast<Range^>(excelWs->Range["B3:H3", Type::Missing])->HorizontalAlignment = Ex-
cel::XlHAlign::xlHAlignCenter;
        mergeExcelCells(excelWs, "B4:C4", "Местоположение");
        mergeExcelCells(excelWs, "D4:H4", dataReport[1]->ToString());
        mergeExcelCells(excelWs, "B5:C5", "Участок");
        mergeExcelCells(excelWs, "D5:H5", dataReport[2]->ToString());
        mergeExcelCells(excelWs, "B6:C6", "Перекресток");
        mergeExcelCells(excelWs, "D6:H6", dataReport[3]->ToString());
        setFullBorderArea(excelWs, "B4:H6");
        mergeExcelCells(excelWs, "J3:N3", "Метеорологические условия - ветер");
        safe_cast<Range^>(excelWs->Range["J3:N3", Type::Missing])->Font->Bold = true;
        safe_cast<Range^>(excelWs->Range["J3:N3", Type::Missing])->HorizontalAlignment = Ex-
cel::XlHAlign::xlHAlignCenter;
        mergeExcelCells(excelWs, "J4:L4", "Скорость, м/с"); safe_cast<Range^>(excelWs-
>Range["M4", Type::Missing])->Value2 = dataReport[4]->ToString();
        mergeExcelCells(excelWs, "J5:L5", "Направление");
        safe_cast<Range^>(excelWs->Range["M5", Type::Missing])->Value2 = dataReport[5]->ToString();
        setFullBorderArea(excelWs, "J4:M5");
        mergeExcelCells(excelWs, "J7:L7", "Интервал расчета, мин"); safe_cast<Range^>(excelWs-
>Range["M7", Type::Missing])->Value2 = dataReport[6]->ToString();
        setFullBorderArea(excelWs, "J7:M7");
        mergeExcelCells(excelWs, "A9:R9", "Объект исследования - вещества выбрасываемые в атмосферу");
        safe_cast<Range^>(excelWs->Range["A9:R9", Type::Missing])->Font->Bold = true;
        safe_cast<Range^>(excelWs->Range["A9:R9", Type::Missing])->HorizontalAlignment = Ex-
cel::XlHAlign::xlHAlignCenter;
        safe_cast<Range^>(excelWs->Range["A10:R10", Type::Missing])->WrapText = true;
```

```
	safe_cast<Range^>(excelWs->Range["A10:R10", Type::Missing])->RowHeight = excelWs->StandardHeight
* 2;
	int count = 0;
	startRow = 10; startColumn = 1;
	for (int i = -1; i < dataSubstances->RowCount; i++) {
		bool doCalc;
		if (i != -1) {
		doCalc = (bool)dataSubstances["ColumnSBSDoCalc", i]->EditedFormattedValue;
		count += (doCalc) ? 1 : 0;
		}
		for (int j = 0; j < dataSubstances->ColumnCount - 2; j++) {
			Object ^data;
			if (i == -1) {
				data = dataSubstances->Columns[j]->HeaderText;
				cellArea = getExcelAreaAdress(
					startRow, (startColumn + j) * 2 - 1,
					startRow, (startColumn + j) * 2);
			}
			else if (doCalc) {
				data = dataSubstances[j, i]->Value->ToString();
				cellArea = getExcelAreaAdress(
					startRow + count, (startColumn + j) * 2 - 1,
					startRow + count, (startColumn + j) * 2);
				if (j > 2) data = Convert::ToDouble(data);
			}
			mergeExcelCells(excelWs, cellArea, data);
		}
	}
	cellArea = getExcelAreaAdress(startRow, startColumn, startRow + count, 18);
	setFullBorderArea(excelWs, cellArea);
	startRow = 10; startColumn = 3;
	cellArea = getExcelAreaAdress(startRow, startColumn, startRow + count, startColumn + 1) + ";" +
	getExcelAreaAdress(startRow, startColumn + 6,startRow + count, startColumn + 6 + 1) + ";" + getExcelAre-
aAdress(startRow, startColumn + 12,startRow + count, startColumn + 12 + 1) +";" + getExcelAreaAdress(startRow,
startColumn + 14,startRow + count, startColumn + 14 + 1);
	int zoom = 0;	int listSize = 0;int amountLists = 1;
	switch (count) {
	case 7: { listSize = 48; zoom = 95; break; }
	case 8: { listSize = 50; zoom = 90; break; }
	case 9:
	case 10: { listSize = 54; zoom = 85; break; }
	case 11: { listSize = 57; zoom = 80; break; }
	default: { listSize = 45; zoom = 100; }
	}
	excelWs->PageSetup->Zoom = zoom;startRow = listSize*amountLists-18; startColumn = 1;
	Excel::Chart ^chart = MakeBarChart(excelWs, startRow, startColumn, cellArea,
		DataChart->Titles["TitleChartCNT"]->Text,
		DataChart->ChartAreas["ChartAreaCNT"]->AxisX->Title,
		DataChart->ChartAreas["ChartAreaCNT"]->AxisY->Title,
		DataChart->Font->Name, DataChart->Font->Size); // создание гистограммы
	safe_cast<Excel::Series ^>(chart->SeriesCollection(2))->Delete();
	safe_cast<Excel::Series ^>(chart->SeriesCollection(3))->Delete();
	safe_cast<Excel::Series ^>(chart->SeriesCollection(4))->Delete();
	safe_cast<Excel::Series ^>(chart->SeriesCollection(1))->Interior->Color = DataChart->Series["SeriesMAC"]-
>Color;
	safe_cast<Excel::Series ^>(chart->SeriesCollection(2))->Interior->Color = DataChart->Series["SeriesCNT"]-
>Color;
	safe_cast<Excel::Series ^>(chart->SeriesCollection(3))->Interior->Color = DataChart-
>Series["SeriesCNTWind"]->Color;
	if (dataTruck != nullptr) {
		startRow = listSize * amountLists  + 2;		amountLists++;
	mergeExcelCells(excelWs, startRow, 1, startRow, 8, "Источники выбросов - ДСМ");
```

```
		safe_cast<Range^>(excelWs->Range[getExcelAreaAdress(startRow, 1, startRow, 8),
Type::Missing])->Font->Bold = true;
		safe_cast<Range^>(excelWs->Range[getExcelAreaAdress(startRow, 1, startRow, 8),
Type::Missing])->HorizontalAlignment = Excel::XlHAlign::xlHAlignCenter;
		startRow++; startColumn = 1;
		for (int i = -1; i < dataTruck->RowCount - 1; i++) {
			for (int j = 1; j < dataTruck->ColumnCount; j++) {
				Object ^data;
				if (i == -1) {
					data = dataTruck->Columns[j]->HeaderText;
					if (j == 1) {
						cellArea = getExcelAreaAdress(startRow, startColumn, startRow,
4);
						data = dataTruck->Columns[j - 1]->HeaderText + " - " + data-
Truck->Columns[j]->HeaderText;
					}
					else if (j == 2 || j == 3)
						cellArea = getExcelAreaAdress(
						startRow, (startColumn + j) * 2 - 1,
						startRow, (startColumn + j) * 2);
					else if (j > 4)
						cellArea = getExcelAreaAdress(startRow, 8 + j, startRow, 8 + j);
				}
				else {
					if (j == 1) {
						cellArea = getExcelAreaAdress(
							startRow + i + 1, startColumn,
							startRow + i + 1, 4);
						data = dataTruck[j - 1, i]->Value->ToString() + " - " + dataTruck[j,
i]->Value->ToString();
					}
					else if (j == 2 || j == 3) {
						cellArea = getExcelAreaAdress(
							startRow + i + 1, (startColumn + j) * 2 - 1,
							startRow + i + 1, (startColumn + j) * 2);
						data = Convert::ToDouble(dataTruck[j, i]->Value);
					}
					else if (j > 4) {
						cellArea = getExcelAreaAdress(startRow + i + 1, 8 + j, startRow +
i + 1, 8 + j);
						data = Convert::ToDouble(dataTruck[j, i]->Value);
					}
				}
				if (j != 4)
					mergeExcelCells(excelWs, cellArea, data);
			}
		}
		cellArea = getExcelAreaAdress(
			startRow, startColumn,
			startRow + dataTruck->RowCount - 1, startColumn + 7);
		setFullBorderArea(excelWs, cellArea);
		cellArea = getExcelAreaAdress(
			startRow, startColumn + 12,
			startRow + dataTruck->RowCount - 1, startColumn + 7 + dataTruck->ColumnCount - 1);
		setFullBorderArea(excelWs, cellArea);
		cellArea = getExcelAreaAdress(startRow, startColumn, startRow + dataTruck->RowCount - 1,
startColumn + 3) +		";" + getExcelAreaAdress(startRow, startColumn + 12, startRow + dataTruck->RowCount -
1, startColumn + 7 + dataTruck->ColumnCount - 1);
		startRow = listSize * amountLists - 18;
		chart = MakeBarChart(excelWs, startRow, startColumn, cellArea,
			DataChart->Titles["TitleChartTrucksEMS"]->Text,
			DataChart->ChartAreas["ChartAreaTrucksEMS"]->AxisX->Title,
```

```
                DataChart->ChartAreas["ChartAreaTrucksEMS"]->AxisY->Title,
                DataChart->Font->Name, DataChart->Font->Size);
            safe_cast<Excel::Series ^>(chart->SeriesCollection(1))->Delete();
            safe_cast<Excel::Series ^>(chart->SeriesCollection(1))->Delete();
            for (int i = 0, countColumn = 1; i < dataSubstances->RowCount; i++) {
                bool doCalc = (bool)dataSubstances["ColumnSBSDoCalc", i]->EditedFormattedValue;
                if (doCalc) {
                    safe_cast<Excel::Series ^>(chart->SeriesCollection(countColumn++))->Interior->Color =
                        DataChart->Series[prefixNameSeriesTruckEMS + dataSubstances["ColumnSBSFormula", i]->Value->ToString()]->Color;
                }
            }
        }
        if (dataAutos != nullptr) {
            startRow = listSize * amountLists  + 2;        amountLists++;
            mergeExcelCells(excelWs, startRow, 1, startRow, 12, "Источники выбросов - городской транспорт");
            safe_cast<Range^>(excelWs->Range[getExcelAreaAdress(startRow, 1, startRow, 12), Type::Missing])->Font->Bold = true;
            safe_cast<Range^>(excelWs->Range[getExcelAreaAdress(startRow, 1, startRow, 12), Type::Missing])->HorizontalAlignment = Excel::XlHAlign::xlHAlignCenter;
            startRow++; startColumn = 1;
            for (int i = -1; i < dataAutos->RowCount; i++) {
                for (int j = 0; j < dataAutos->ColumnCount; j++) {
                    Object ^data;
                    if (i == -1) {
                        data = dataAutos->Columns[j]->HeaderText;
                        if (j < 6)
                            cellArea = getExcelAreaAdress(
                            startRow, (startColumn + j) * 2 - 1,
                            startRow, (startColumn + j) * 2);
                        else
                            cellArea = getExcelAreaAdress(
                            startRow, startColumn + 6 + j,
                            startRow, startColumn + 6 + j);
                    }
                    else {
                        if (j < 6)
                            cellArea = getExcelAreaAdress(
                            startRow + i + 1, (startColumn + j) * 2 - 1,
                            startRow + i + 1, (startColumn + j) * 2);
                        else
                            cellArea = getExcelAreaAdress(
                            startRow + i + 1, 7 + j,
                            startRow + i + 1, 7 + j);
                        data = dataAutos[j, i]->Value->ToString();
                        if (j > 0) data = Convert::ToDouble(data);
                    }
                    mergeExcelCells(excelWs, cellArea, data);
                }
            }
            cellArea = getExcelAreaAdress(startRow, startColumn, startRow + dataAutos->RowCount, startColumn + 6 + dataAutos->ColumnCount - 1);
            setFullBorderArea(excelWs, cellArea);
            cellArea = getExcelAreaAdress(startRow, startColumn, startRow + dataAutos->RowCount, startColumn + 1) +  ";" + getExcelAreaAdress( startRow, startColumn + 12, startRow + dataAutos->RowCount, startColumn + 6 + dataAutos->ColumnCount - 1);
            startRow = listSize * amountLists - 18;
            chart = MakeBarChart(excelWs, startRow, startColumn, cellArea,
                DataChart->Titles["TitleChartAutosEMS"]->Text,
                DataChart->ChartAreas["ChartAreaAutosEMS"]->AxisX->Title,
```

```
                        DataChart->ChartAreas["ChartAreaAutosEMS"]->AxisY->Title,
                        DataChart->Font->Name, DataChart->Font->Size);
chart->ChartArea->Format->TextFrame2->TextRange->Font->Name = DataChart->Font->Name;
chart->ChartArea->Format->TextFrame2->TextRange->Font->Size = DataChart->Font->Size;
            for (int i = 0, countColumn = 1; i < dataSubstances->RowCount; i++) {
                        bool doCalc = (bool)dataSubstances["ColumnSBSDoCalc", i]->EditedFormattedValue;
                        if (doCalc) {
                                    safe_cast<Excel::Series ^>(chart->SeriesCollection(countColumn++))->Interior-
>Color =
                                                DataChart->Series[prefixNameSeriesAutoEMS + dataSubstanc-
es["ColumnSBSFormula", i]->Value->ToString()]->Color;
                        }
            }

      }
      excelApp->Visible = true;
}
```

Разработанный код формы FormChartCNT3D.h (основные обработчики событий):

```
private: System::Void Panel_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) { PaintE-
vent = e; glClearColor(1.0, 1.0, 1.0, 1.0);
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glLoadIdentity(); glRotated(angleX, 1.0, 0.0, 0.0);
      glRotated(angleY, 0.0, 1.0, 0.0); glRotated(angleZ, 0.0, 0.0, 1.0);
      double zoomX = Convert::ToDouble(LengthOX->Value);
      double zoomY = Convert::ToDouble(LengthOY->Value);
      double zoomZ = Convert::ToDouble(LengthOZ->Value);
      int indexSubstance = GetSelectedIndexFromListSubstances();
      glScaled(pow(zoomX, -1.0), pow(zoomY, -1.0), pow(zoomZ, -1.0));
      glEnable(GL_LIGHTING);          glEnable(GL_NORMALIZE);
      GLfloat diffuse[] = { 1.0, 1.0, 1.0, 1.0 };  GLfloat position[] = { 0.0, 0.0, zoomZ, 1.0 };
      glEnable(GL_LIGHT0); glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
      glLightfv(GL_LIGHT0, GL_POSITION, position); glEnable(GL_COLOR_MATERIAL);
      glDrawAxis('X', "", -zoomX, 0.0, 6, 10, zoomX, zoomY, zoomZ);
      glDrawAxis('X', "X Distance, m", 0.0, zoomX, 6, 10, zoomX, zoomY, zoomZ);
      glDrawAxis('Y', "", -zoomY, 0.0, 6, 10, zoomX, zoomY, zoomZ);
      glDrawAxis('Y', "Y Distance, m", 0.0, zoomY, 6, 10, zoomX, zoomY, zoomZ);
      glDrawAxis('Z', "Z Concentration, mg/m3", 0.0, zoomZ, 10, 4, zoomX, zoomY, zoomZ);
      List<List<Point3D^>^> ^data = substances[indexSubstance]->getCxy();
      List<List<Point3D^>^> ^data2 = gcnew List<List<Point3D^>^>();
      for (int i = 0; i < data[0]->Count; i++) { List<Point3D^> ^lines = gcnew List<Point3D^>();
            for (int j = 0; j < data->Count - 1; j++)
                  lines->Add(gcnew Point3D(data[j][i]->X, data[j][i]->Y, data[j][i]->Z));
            data2->Add(lines);
      }
      for (int i = 0; i < data->Count; i++) {
            glDrawLineStrip(data[i], substances[indexSubstance]->getColor(), 1.0);
            for (int j = 0; j < data[i]->Count; j++) data[i][j]->Y *= -1.0;
            glDrawLineStrip(data[i], substances[indexSubstance]->getColor(), 1.0);
      }
      for (int i = 0; i < data2->Count; i++) {
            glDrawLineStrip(data2[i], substances[indexSubstance]->getColor(), 1.0);
            for (int j = 0; j < data2[i]->Count; j++) data2[i][j]->Y *= -1.0;
            glDrawLineStrip(data2[i], substances[indexSubstance]->getColor(), 1.0);
      }
      double Xm = substances[indexSubstance]->getXm();
      double Xmu = substances[indexSubstance]->getXmu();
      glDrawEllips(gcnew Point3D(0.0, 0.0, 0.0), Xm, Xm, Color::Red, 2);
      glDrawEllips(gcnew Point3D(0.0, 0.0, 0.0), Xmu, Xmu, Color::Green, 2);
```

```
            glDrawEllips(gcnew Point3D(0.0, 0.0, 0.0), zoomX, zoomY, Color::Black, 2);
            double mac = substances[indexSubstance]->getMAC();
            char buff[30]; sprintf(buff, "Level MAC - %.2f", mac);
            glDrawFontRastr(10, gcnew Point3D(0.0, zoomY / 10, mac), (const char *)buff);
            glLineWidth(2.0); glBegin(GL_LINES);
           glVertex3d(0.0, 0.0, mac); glVertex3d(0.0, zoomY / 10, mac);              glEnd();
            SwapBuffers(hDC);
 }
```

Коды SQL-запросов создания последовательностей:

```
CREATE SEQUENCE count_autos
INCREMENT 1  MINVALUE 0  MAXVALUE 9223372036854775807  START 1  CACHE 1;
CREATE SEQUENCE count_autos_in_report
INCREMENT 1  MINVALUE 0  MAXVALUE 9223372036854775807  START 1  CACHE 1;
CREATE SEQUENCE count_report
INCREMENT 1  MINVALUE 0  MAXVALUE 9223372036854775807  START 1  CACHE 1;
CREATE SEQUENCE count_substances_in_report
INCREMENT 1  MINVALUE 0  MAXVALUE 9223372036854775807  START 33  CACHE 1;
CREATE SEQUENCE count_trucks
INCREMENT 1  MINVALUE 0  MAXVALUE 9223372036854775807  START 1  CACHE 1;
CREATE SEQUENCE count_trucks_in_report
INCREMENT 1  MINVALUE 0  MAXVALUE 9223372036854775807  START 1  CACHE 1;
CREATE SEQUENCE count_trucks_park
INCREMENT 1  MINVALUE 0  MAXVALUE 9223372036854775807  START 1  CACHE 1;
```

Коды SQL-запросов создания пользовательского типа данных:

```
CREATE TYPE t_truck AS (
            goverment character varying, type_truck character varying,
            power double precision, height_tube double precision);
```

Коды SQL-запросов создания таблиц:

```
CREATE TABLE autos(
  id bigint NOT NULL DEFAULT nextval('count_autos'::regclass),
  name_road character varying NOT NULL,
  car_l bigint NOT NULL DEFAULT 0,
  car_am bigint NOT NULL DEFAULT 0,
  car_g bigint NOT NULL DEFAULT 0,
  car_g12 bigint NOT NULL DEFAULT 0,
  car_am3_5 bigint NOT NULL DEFAULT 0,
  speed double precision DEFAULT 0.0,
  length_road double precision DEFAULT 0.0,
  time_stop bigint DEFAULT 0,
  CONSTRAINT pr_key_autos_id PRIMARY KEY (id));

CREATE TABLE autos_in_report(
  id bigint NOT NULL DEFAULT nextval('count_autos_in_report'::regclass),
  id_report bigint NOT NULL,
  id_auto bigint NOT NULL,
  CONSTRAINT pr_key_autos_in_report_id PRIMARY KEY (id),
  CONSTRAINT fr_key_autos_in_report_id_auto FOREIGN KEY (id_auto)
     REFERENCES autos (id) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fr_key_autos_in_report_id_report FOREIGN KEY (id_report)
     REFERENCES reports (id) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE TABLE coef_a(
  id bigint NOT NULL,
```

```
  coef double precision,
  CONSTRAINT pr_key_coef_a_id PRIMARY KEY (id));

CREATE TABLE regions(
  id bigint NOT NULL,
  name_region character varying NOT NULL,
  id_coef bigint NOT NULL,
  CONSTRAINT pr_key_regions_id PRIMARY KEY (id),
  CONSTRAINT fr_key_regions_id_coef FOREIGN KEY (id_coef)
      REFERENCES coef_a (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE TABLE reports(
  id bigint NOT NULL DEFAULT nextval('count_report'::regclass),
  date_calc timestamp with time zone NOT NULL DEFAULT now(),
  id_region bigint NOT NULL,
  name_city character varying NOT NULL,
  name_site character varying NOT NULL,
  time_average bigint NOT NULL DEFAULT 20,
  wind_speed double precision NOT NULL DEFAULT 0.5,
  wind_derection character varying NOT NULL,
  coef double precision NOT NULL DEFAULT 0.0,
  type_road smallint NOT NULL,
  source_ems_car boolean NOT NULL,
  source_ems_truck boolean NOT NULL,
  CONSTRAINT pr_key_report_id PRIMARY KEY (id),
  CONSTRAINT fk_key_reports_id_region FOREIGN KEY (id_region)
      REFERENCES regions (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE TABLE substances(
  id bigint NOT NULL,
  code bigint NOT NULL,
  formula character varying NOT NULL,
  name_sbs character varying NOT NULL,
  dangers smallint NOT NULL,
  mac double precision NOT NULL DEFAULT 0.0,
  adc double precision NOT NULL DEFAULT 0.0,
  CONSTRAINT pr_key_substances_id PRIMARY KEY (id));

CREATE TABLE substances_in_report(
  id bigint NOT NULL DEFAULT nextval('count_substances_in_report'::regclass),
  id_report bigint NOT NULL,
  id_substance bigint NOT NULL,
  CONSTRAINT pr_key_substances_in_report_id PRIMARY KEY (id),
  CONSTRAINT fr_key_substances_in_report_id_report FOREIGN KEY (id_report)
      REFERENCES reports (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fr_key_substances_in_report_id_substance FOREIGN KEY (id_substance)
      REFERENCES substances (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE TABLE trucks(
  id bigint NOT NULL DEFAULT nextval('count_trucks'::regclass),
  id_truck_park bigint NOT NULL,
  goverment character varying NOT NULL,
  CONSTRAINT pr_key_trucks_id PRIMARY KEY (id),
  CONSTRAINT fr_key_trucks_id_truck_park FOREIGN KEY (id_truck_park)
      REFERENCES trucks_park (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE TABLE trucks_in_report(
```

```
  id bigint NOT NULL DEFAULT nextval('count_trucks_in_report'::regclass),
  id_report bigint NOT NULL,
  id_truck bigint NOT NULL,
  CONSTRAINT pr_key_trucks_in_report_id PRIMARY KEY (id),
  CONSTRAINT fr_key_trucks_in_report_id_report FOREIGN KEY (id_report)
      REFERENCES reports (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fr_key_trucks_in_report_id_truck FOREIGN KEY (id_truck)
      REFERENCES trucks (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE TABLE trucks_park(
  id bigint NOT NULL DEFAULT nextval('count_trucks_park'::regclass),
  id_type bigint NOT NULL,
  power double precision NOT NULL DEFAULT 0.0,
  height_tube double precision NOT NULL DEFAULT 0.0,
  CONSTRAINT pr_key_trucks_park_id PRIMARY KEY (id),
  CONSTRAINT fr_key_trucks_park_id_type FOREIGN KEY (id_type)
      REFERENCES trucks_type (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION);

CREATE TABLE trucks_type(
  id bigint NOT NULL,
  name_type character varying NOT NULL,
  CONSTRAINT pr_key_trucks_type_id PRIMARY KEY (id));
```

Коды SQL-запросов создания представлений:

```
CREATE OR REPLACE VIEW public.get_reports_short AS
SELECT reports.id, reports.date_calc, regions.name_region, reports.name_city, reports.name_site, re-
ports.source_ems_car,  reports.source_ems_truck FROM reports, regions
WHERE reports.id_region = regions.id;

CREATE OR REPLACE VIEW public.get_substances AS
SELECT substances.code, substances.formula, substances.name_sbs, substances.dangers,
substances.mac, substances.adc  FROM substances;

CREATE OR REPLACE VIEW public.get_trucks_park AS
SELECT trucks_park.id_type, trucks_type.name_type, trucks_park.power,trucks_park.height_tube
FROM trucks_park, trucks_typeWHERE trucks_park.id_type = trucks_type.id;
```

Коды SQL-запросов создания функций:

```
CREATE FUNCTION get_autos_from_report(id_rpt bigint) RETURNS SETOF autos AS
$BODY$
DECLARE id_at bigint; auto autos;
begin
for auto in
select autos.id,autos.name_road,autos.car_l, autos.car_am, autos.car_g, autos.car_g12, autos.car_am3_5,autos.speed,
autos.length_road, autos.time_stop from autos, autos_in_report
where autos_in_report.id_report = id_rpt and autos_in_report.id_auto = autos.id
loop return next auto; end loop;
return;
end
$BODY$  LANGUAGE plpgsql;

CREATE FUNCTION get_coef_region(id_region bigint) RETURNS double precision AS
$BODY$
DECLARE coef double precision = 0.0;
begin
```

```
SELECT coef_a.coef  FROM regions, coef_a  WHERE regions.id = id_region and coef_a.id = regions.id_coef INTO
coef;
return coef;
end
$BODY$  LANGUAGE plpgsql;

CREATE FUNCTION get_substances_from_report(id_rpt bigint) RETURNS SETOF bigint AS
$BODY$
DECLARE id_sbs bigint;
begin
for id_sbs in select id_substance from substances_in_report where id_report = id_rpt
loop return next id_sbs; end loop;
return;
end
$BODY$  LANGUAGE plpgsql;

CREATE FUNCTION get_trucks_from_report(id_rpt bigint)  RETURNS SETOF t_truck AS
$BODY$
DECLARE id_trk bigint; truck t_truck;
begin
for id_trk in select id_truck from trucks_in_report where id_report = id_rpt
loop
        select trucks.goverment, trucks_type.name_type, trucks_park.power, trucks_park.height_tube
        from trucks, trucks_park, trucks_type
        where trucks.id = id_trk and trucks.id_truck_park = trucks_park.id and trucks_park.id_type = trucks_type.id
into truck;
        return next truck;
end loop;
return;
end
$BODY$  LANGUAGE plpgsql;

CREATE FUNCTION save_new_trucks_on_park(list_trucks t_truck[])  RETURNS bigint[] AS
$BODY$
DECLARE id_truck_type bigint; id_truck_park bigint; ids_trucks_park bigint[]; truck t_truck;
begin
foreach truck in array list_trucks loop
        select trucks_park.id, trucks_type.id           from trucks_park, trucks_type
        where trucks_type.name_type = truck.type_truck and trucks_park.power = truck.power and
trucks_park.height_tube = truck.height_tube
        into id_truck_park, id_truck_type;
        if (id_truck_park is null) then
                select id from trucks_type where trucks_type.name_type = truck.type_truck into id_truck_type;
                insert into trucks_park (id_type, power, height_tube)
                values (id_truck_type, truck.power, truck.height_tube);
                id_truck_park = currval('count_trucks_park');
        end if;
        ids_trucks_park = array_append(ids_trucks_park, id_truck_park);
end loop;
return ids_trucks_park;
end
$BODY$  LANGUAGE plpgsql;

CREATE FUNCTION save_report( report reports, list_ids_substances bigint[], list_autos autos[], list_trucks t_truck[])
RETURNS bigint AS
$BODY$ DECLARE id_object bigint; auto autos; truck t_truck; ids_trucks_park bigint[];
begin
insert into reports(date_calc, id_region, name_city, name_site, time_average, wind_speed, wind_derection, coef,
type_road, source_ems_car, source_ems_truck)
values (report.date_calc,report.id_region,  report.name_city, report.name_site, report.time_average, report.wind_speed,
report.wind_derection, report.coef, report.type_road,           report.source_ems_car, report.source_ems_truck);
report.id = currval('count_report');
```

```
foreach id_object in array list_ids_substances loop
	insert into substances_in_report(id_report, id_substance) values (report.id, id_object);
end loop;
if(report.source_ems_car = true) then
	foreach auto in array list_autos loop
		insert into autos (name_road, car_l, car_am, car_g, car_g12, car_am3_5, speed, length_road, time_stop) values (auto.name_road, auto.car_l, auto.car_am, auto.car_g, auto.car_g12, auto.car_am3_5, auto.speed, auto.length_road, auto.time_stop);
	end loop;
end if;
if(report.source_ems_truck = true) then
	SELECT save_new_trucks_on_park(list_trucks) into ids_trucks_park; id_object = 1;
	foreach truck in array list_trucks loop
		insert into trucks(id_truck_park, goverment) values (ids_trucks_park[id_object], truck.goverment);
id_object = id_object + 1;
	end loop;
end if;
return report.id;
end
$BODY$  LANGUAGE plpgsql;
```

Коды SQL-запросов создания триггерных функций:

```
CREATE FUNCTION save_autos_to_report()  RETURNS trigger AS
$BODY$
begin
	insert into autos_in_report(id_report, id_auto) values (currval('count_report'), new.id);
	return new;
end;
$BODY$  LANGUAGE plpgsql;
CREATE TRIGGER a_trig_save_autos_to_report AFTER INSERT ON autos FOR EACH ROW
EXECUTE PROCEDURE save_autos_to_report();

CREATE FUNCTION save_trucks_to_report()  RETURNS trigger AS
$BODY$
begin
	insert into trucks_in_report(id_report, id_truck) values (currval('count_report'), new.id);
	return new;
end;
$BODY$  LANGUAGE plpgsql;
CREATE TRIGGER a_trig_save_trucks_to_report AFTER INSERT ON trucks FOR EACH ROW
EXECUTE PROCEDURE save_trucks_to_report();
```