Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Базовая кафедра геоинформационных систем

УТВЕРЖДАЮ
Заведующий кафедрой

_____ В.И. Харук
подпись
« _____» _____ 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 - Информационные системы и технологии

Информационно-аналитическое обеспечение задач оценки уровня загрязнения в городе Красноярск

Руководитель _____доцент кафедры Б-ГИС, к.ф.м.н. О.Э. Якубайлик
подпись, дата

Выпускник _____ А.В. Пелих
подпись, дата

Нормоконтролер _____ Е.В. Федотова
подпись, дата

Красноярск 2018

РЕФЕРАТ

Выпускная квалификационная работа по теме «Информационно-аналитическое обеспечение задач оценки уровня загрязнения в городе Красноярск» содержит х страниц текстового документа, 5 приложений, 7 использованных источников.

HTTP, CSS, JAVASCPRIPT, VUE.JS, MYSQL, PHP, КРАСНОЯРСК, PM2.5, ЗАГРЯЗНЕНИЕ

В данной работе было создано интерактивное веб-приложение, предназначенное для визуального анализа и оценки уровня концентрации загрязняющих мелкодисперсных частиц в атмосфере города Красноярск.

Цель работы: создать интерактивное веб-приложение, позволяющее оценить и проанализировать данные о концентрации загрязняющих веществ.

Задачи:

- подготовить данные о концентрации загрязняющего вещества PM 2,5, полученные в атмосфере города Красноярск;

- создать базу данных для хранения подготовленных данных;

- создать веб-приложение, обеспечивающее возможность визуализации полученных данных по выбранным пользователем критериям;

- создать back-end инфраструктуру сайта, обеспечивающую возможность взаимодействия веб-приложения с базой данных;

- создать компонент для обработки, полученной из базы данных информации, и вывода обработанной информации в виде таблицы.

На основе современных инструментов веб-разработки было создано интерактивное веб приложение, позволяющее обрабатывать и анализировать данные о концентрации мелкодисперсной пыли PM2.5 в воздухе, хранящиеся в базе данных.

СОДЕРЖАНИЕ

Введение 5
1 Обзор экологической ситуации в мире 6
1.1 Экологическая ситуация в России 7
1.2 Экологическая ситуация в Красноярске 8
1.3 Постановка задач 9
2 Описание используемого программного обеспечения 10
2.1 Программное обеспечение QGIS 10
2.2 Программное обеспечение Open Server 11
2.3 Программное обеспечение Excel 12
2.4 Программное обеспечение Word 12
2.5 Язык гипертекстовой разметки HTML 13
2.6 Язык описания стилей CSS 13
2.7 Язык программирования JavaScript 14
2.8 Серверный язык программирования PHP 14
2.9 Система управления базами данных MySQL 15
2.10 Фреймворк Vue.js 15
3 Подготовка исходных данных 16
3.1 Получение исходных данных 16
3.1.1 Создание базы данных 18
3.1.2 Создание скрипта для загрузки данных в базу данных 21
3.1.3 Подключение к базе данных 22
3.1.4 Разработка класса для работы с базой данных 23
3.2 Разработка HTML структуры страницы 24
3.2.1 HTML структура компонента построения таблиц 25
3.2.2 HTML структура компонента для построения таблиц без SQL 26
запроса 26
3.3 Создание основной логики для веб-приложения 27
3.3.1 Создание логики для компонента вывода таблиц 30
3.3.2 Результаты работы созданной системы 33
4 Направление дальнейших разработок 36

Заключение 37
Список использованных источников 38
Приложение А HTML структура приложения 39
Приложение Б Программный код класса для обработки запросов «Query» 42
Приложение В Программный код основной логики веб-приложения 44
Приложение Г Вспомогательные функции для веб-приложения 48
Приложение Д Описание CSS стилей приложения 49

ВВЕДЕНИЕ

На текущий момент, экологическая ситуация в России является одним из самых острых вопросов во внутренней государственной политике. Данный тезис подтверждается цитатой президента Российской Федерации В.В. Путина: «Снижение негативного влияния человека на природу является одной из приоритетных общенациональных задач и от её решения прямо зависит устойчивое, гармоничное развитие страны, здоровье людей, сохранность редких видов флоры и фауны».

В связи с этим, указом президента Российской Федерации от 05.01.2016 г. № 7 «О проведении в Российской Федерации Года экологии», 2017 год был объявлен годом экологии в котором был подготовлен ряд постановлений, направленных на привлечение внимания общества к вопросам экологического развития Российской Федерации, сохранения биологического разнообразия и обеспечения экологической безопасности.

В целом, обсуждение проблем экологии вызывает широкий резонанс в обществе, что также привело к различным инициативам со стороны активистов.

Целью работы является создание информационно-аналитического интерактивного веб-сервиса для анализа и оценки текущей экологической ситуации в городе Красноярск.

Задачи:

– подготовка исходных данных о концентрации загрязняющего вещества PM2.5 в атмосфере города Красноярск;

– создание программной инфраструктуры в виде базы данных, обеспечивающей возможность долговременного хранения данных;

– подготовка и создание собственного программного обеспечения, необходимого для проведения наглядного сравнительного анализа данных, полученных из различных источников.

1 Обзор экологической ситуации в мире

Со времён промышленной революции, начавшейся в Европе в XVIII веке, человек начал оказывать значительно большее влияние на биосферу, чем до этого. Причиной тому послужили добыча и использование углеводородного топлива – угля, нефти, сланцев, газа. Далее последовала добыча металлов и других полезных ископаемых в очень больших объёмах. В биосфере появились вещества, запасённые давно вышедшими из кругооборота веществ биосферами. Это вещества, находившиеся до этого в осадочных породах. Появление этих веществ вызвало загрязнение воды, воздуха, почвы. В связи с быстро нарастающей интенсивностью процесса загрязнения начали значительно меняться условия обитания.

В связи с этими процессами перед людьми встала проблема изучения влияния данных процессов на здоровье и условия жизни, из чего последовало создание различных дисциплин в экологии, например, создание таких технологий, которые в наименьшей степени влияют на окружающую среду. Технологии подобного рода принято называть экологичными. Инженерные и научные дисциплины, занимающиеся принципами создания таких технологий, имеют общее название – инженерная или промышленная экология.

На сегодняшний день можно перечислить несколько основных источников негативного антропогенного влияния на окружающую среду, к ним относятся: транспортные средства, работающие на углеводородном топливе, различные заводы химической промышленности, предприятия, обслуживающие топливно-энергетический комплекс.

Перечисленные источники загрязнения приводят к появлению в атмосфере тяжёлых металлов, например, свинца или ртути.

Также стоит отметить, что особо опасными являются токсичные газы, вырабатываемые и выбрасываемые в атмосферу предприятиями химической промышленности, такие отходы как оксид серы или азота, часто являются

причиной кислотных дождей, особенно в окрестностях городов с развитой химической промышленностью. Данные загрязняющие вещества способны вступать в реакции с компонентами биосферы, что влечёт за собой образование других опасных производных веществ.

1.1 Экологическая ситуация в России

На сегодняшний день Россия входит в список стран с высоким уровнем загрязнения окружающей среды. Тяжёлая экологическая ситуация обусловлена целым рядом способствующих тому причин, к которым можно отнести:

- низкий уровень контроля над количеством выбрасываемых в окружающую среду вредных веществ различными предприятиями
- недостаточный надзор за незаконной вырубкой леса
- высокая интенсивность урбанизации населения, в результате которой за последнее время значительно увеличилось количество транспорта в городах
- низкий уровень модернизации на предприятиях, что приводит к использованию устаревших технологий очистки отходов

Для определения степени загрязнения атмосферного воздуха принято использовать следующие нормативы предельно допустимой концентрации (ПДК):

- ПДКраз - предельно допустимая разовая измеренная концентрация загрязняющего вещества
- ПДКср - оценка усреднённой за некоторый период времени (от суток до года) концентрации загрязнителя

В России существует нецентрализованная система контроля загрязнения атмосферы, которая включает в себя более тысячи станций наблюдений. По данным, получаемым с данных станций, показатели концентрации некоторых загрязняющих веществ, таких как: пыль, аммиак,

сероводород оксид углерода иногда могут превышать установленное значение максимальной разовой концентрации ПДКраз многократно.

Несмотря на принимаемые меры по снижению негативного воздействия на окружающую среду, подобные тенденции сохраняются и влекут за собой напряжённую экологическую обстановку в большом количестве городов с высоким уровнем загрязнения. Экологическая обстановка в некоторых городах оценивается как опасная.

1.2 Экологическая ситуация в Красноярске

Красноярск является административным центром Красноярского края, население которого, по данным за 2017 год, составляет 1 082 933 человека.

Город расположен вдоль глубокой долины вдоль обоих берегов реки Енисей. Его протяженность составляет приблизительно 20км вдоль берегов, глубина застройки в городе колеблется в районе 4км. Площадь города составляет 353,9 км^2. В состав города входит 7 районов: Железнодорожный, Октябрьский, Центральный, Ленинский, Советский, Свердловский и Кировский.

Официальный контроль за уровнем загрязнения на территории Красноярска осуществляется министерством экологии и рационального природопользования Красноярского края. Для предоставления данных об экологической обстановке в Красноярске всем гражданам, был создан информационный портал «krasecology.ru».

Данный портал предоставляет в открытом доступе большой перечень жизненно важных параметров экологической обстановки в Красноярске, в список таких параметров входят:

- состояние атмосферного воздуха, в том числе данные оперативного мониторинга;
- данные о радиационной обстановке;
- данные о уровне загрязнения почвы;

- данные о уровне загрязнения водоёмов в Красноярске и его окрестностях.

На текущий момент, наиболее широкий резонанс получило обсуждение такого показателя, как концентрация в воздухе мелкодисперсных частиц PM2,5, так как по оценкам различных экспертов, именно это загрязняющее вещество является наиболее частой причиной неблагоприятной экологической обстановки в период неблагоприятных метеоусловий в городе Красноярск.

Однако, оба ресурса предоставляют доступ лишь к текущим показаниям датчиков, что не позволяет каким-либо образом проанализировать изменения уровня концентрации взвешенных частиц на долгом отрезке времени.

Специалистами Института Вычислительного Моделирования СО РАН был разработан веб-портал gis.krasn.ru, в котором собираются и хранятся данные обоих источников. Данный портал предоставляет возможность свободно скачать эти данные и обрабатывать их по своему усмотрению.

1.3 Постановка задач

Выполнение данной работы заключается в выполнении следующих задач: получение и подготовка исходных данных, полученных благодаря веб-порталу ИВМ СО РАН gis.krasn.ru к загрузке в собственную базу данных. Создание базы данных для хранения этих данных. Загрузга подготовленных исходных данных в созданную базу данных. Создание собственного интерактивного веб-приложения на основе современных веб-технологий, предназначенного для обработки и обеспечивающего возможность анализа, хранящихся в БД данных.

2 Описание используемого программного обеспечения

2.1 Программное обеспечение QGIS

Quantum GIS (QGIS) — свободная кроссплатформенная геоинформационная система. Работа над QGIS была начата в мае 2002 года, а в июне того же года — создан проект на площадке SourceForge. Целью создания QGIS было сделать использование геоинформационных систем легким и понятным для пользователя, чего создатели QGIS отчасти добились: интерфейс Quantum GIS намного понятнее для неискушенного пользователя, а в некоторых аспектах даже превосходит широко распространенные ГИС.

Данная система позволяет просматривать и накладывать друг на друга векторные и растровые данные в различных форматах и проекциях без преобразования во внутренний или общий формат. Поддерживаются следующие основные форматы:

- пространственные таблицы PostgreSQL с использованием PostGIS, векторные форматы, поддерживаемые установленной библиотекой OGR, включая shape-файлы ESRI, MapInfo, SDTS (Spatial Data Transfer Standard) и GML (Geography Markup Language) и др.;
- форматы растров и графики, поддерживаемые библиотекой GDAL (Geospatial Data Abstraction Library), такие, как GeoTIFF, Erdas IMG, ArcInfo ASCII Grid, JPEG, PNG и др.;
- базы данных SpatiaLite;
- растровый и векторный форматы GRASS (область/набор данных).

Имеется возможность анализировать векторные пространственные данные в PostgreSQL/PostGIS и других форматах, поддерживаемых OGR, используя модуль fTools, написанный на языке программирования Python. В настоящее время QGIS предоставляет возможность использовать

инструменты анализа, выборки, геопроцессинга, управления геометрией и базами данных.

QGIS может быть адаптирован к особым потребностям с помощью расширяемой архитектуры модулей. QGIS предоставляет библиотеки, которые могут использоваться для создания модулей. Можно создавать отдельные приложения, используя языки программирования C++ или Python.

2.2 Программное обеспечение Open Server

Open Server является профессиональным инструмент для веб-разработки. Он представляет собой портативный локальный WAMP/WNMP сервер, имеющий многофункциональную управляющую программу и большой выбор подключаемых компонентов.

Для отладки скриптов в различном окружении Open Server предлагает на выбор сразу два вида HTTP серверов, различные версии PHP и СУБД модулей, а также возможность быстрого переключения между ними.

Среди особых возможностей Open Server можно перечислить:

- подробный просмотр логов всех компонентов в реальном времени;
- выбор HTTP, СУБД и PHP модулей в любом сочетании;
- поддержка SSL и кириллических доменов «из коробки»;
- поддержка алиасов или по-другому доменных указателей, а также удобная форма их настройки;
- создание локального поддомена без потери видимости основного домена в сети Интернет;
- доступ к доменам и быстрый доступ к шаблонам конфигурации модулей;
- мультиязычный интерфейс (Русский, Украинский, Белорусский, Английский).

2.3 Программное обеспечение Excel

Excel - это широко распространенная компьютерная программа. Нужна она для проведения расчетов, составления таблиц и диаграмм, вычисления простых и сложных функций. Она входит в состав пакета Microsoft Office.

Excel представляет собой большую таблицу, в которую можно вносить данные. Также, используя функции этой программы, можно производить с цифрами разные манипуляции: складывать, вычитать, умножать, делить и многое другое.

Если требуется не только расчертить таблицу со словами и цифрами, но еще и произвести с цифрами какие-либо действия (сложить, умножить, вычислить процент и т.д), тогда Вам нужно работать в Microsoft Excel.

2.4 Программное обеспечение Word

Microsoft Word - это программа для печати текста и составления документов.

В ней можно набрать любой тип текста: статью, документ, реферат, курсовую, диплом и даже книгу. Также в этой программе можно красиво оформить текст - добавить в него картинку или фото, выделить его части разными цветами, изменить шрифт, размер букв и многое другое. А еще в Microsoft Word можно составить таблицу, напечатать объявление или сделать плакат. Плюс ко всему напечатанное можно вывести на бумагу, то есть распечатать на принтере.

Программа Word представляет собой белый лист бумаги, на котором, используя клавиатуру компьютера, сразу же можно печатать. Причем, это не один лист бумаги: если Вам нужно напечатать много текста, и на один лист он не поместится, то программа автоматически добавит еще листы. Также напечатанный текст можно отредактировать: изменить размер букв, шрифт, начертание и многое другое.

2.5 Язык гипертекстовой разметки HTML

Язык гипертекстовой разметки HTML, как и его, на данный момент, последняя версия HTML5, которая использовалась в данной работе, предназначен для описания структуры веб страницы при помощи различных тегов.

Тег – в языке HTML является структурированной текстовой записью, определяющей тип и, соответственно, поведение HTML элемента.

На данный момент является самым распространённым инструментом для разработки структуры HTML документа.

2.6 Язык описания стилей CSS

Язык описания стиле CSS используется для описания внешнего вида HTML элементов на странице.

Данный язык применяется к уже существующим элементам веб страницы, следовательно, используется исключительно в связке с языком гипертекстовой разметки HTML.

Так как разметка структуры веб-страницы при помощи языка HTML представляет из себя лишь базовую структуру для размещения элементов, находящихся на веб-странице, существует необходимость правильно оформить эти элементы.

Язык описания стилей CSS позволяет настроить такие параметры элемента, как цвет, цвет фона, границы, размер, расположение по отношению к другим элементам веб-страницы и многие другие.

В данной работе используется последняя версия инструмента – CSS3.

2.7 Язык программирования JavaScript

Язык программирования JavaScript является высокоуровневым языком программирования. Инструментарий данного языка является одним из наиболее продвинутых на сегодняшний день и позволяет реализовывать для приложений логику любой сложности на стороне клиента.

Данный язык является мультипарадигменным, так как поддерживает императивный, функциональный и объектно-ориентированный стили программирования.

Основные архитектурные черты языка: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

Также, данный язык программирования располагает огромном количеством различных сторонних библиотек и фреймворков, что позволяет быстро находить готовые эффективные решения для самых разных задач.

2.8 Серверный язык программирования PHP

Язык программирования PHP является скриптовым языком программирования. Он используется на большинстве тех сайтов, которым необходимо взаимодействие с сервером для работы.

Данный язык работает по следующему принципу – в тексте основного HTML документа, находящегося на сервере, при помощи специального тега «<?PHP ?>» располагают текст скрипта написанного на PHP. Перед тем, как страница отправится в браузер клиента, все скрипты обрабатываются сервером, формируется конечная HTML разметка и клиенту отправляется файл с готовой HTML разметкой.

Таким образом, в значительной степени, обеспечивается высокий уровень безопасности той части приложения, которая основана на работе с сервером, так как клиент не имеет доступа к PHP скриптам.

2.9 Система управления базами данных MySQL

СУБД MySQL предоставляет возможности для хранения большого количества различных типов данных, объединённых в таблицы.

Так как MySQL является совершенно бесплатным продуктом с открытым исходным кодом – это даёт возможность использовать для приложений почти любого масштаба, а также идеально подходит для разработки данного веб-сервиса.

Простота использования и поддержки баз данных, разработанных при помощи данной СУБД, позволяет разворачивать базы данных в крайне короткие сроки, что также является преимуществом данной технологии.

2.10 Фреймворк Vue.js

Фреймворк Vue.js является очень мощным современным инструментом для разработки, в основном, клиентской части веб приложений.

Данный фреймворк был разработан совсем недавно, в 2015 году, и представляет собой, своего рода – эволюцию, либо некое переосмысление более старых фреймворков. В основном, он вобрал в себя лучшие качества таких фреймворков, как React.js и Angular, которые, тем не менее, до сих пор являются лидерами на рынке.

По данным различных синтетических тестов, именно Vue.js показывает наилучшие показатели оптимизации, а простота его изучения и относительно низкий порог входа ценятся большим количеством опытных программистов в области front-end разработки.

3 Подготовка исходных данных

3.1 Получение исходных данных

Для получения исходных данных, в данной работе использовался веб-портал gis.krasn.ru. Данный геопортал разработан сотрудниками института вычислительного моделирования ИВМ СО РАН.

Геопортал – это, по определению, располагающемуся на самом сайте, специализированный картографический веб-сайт, предоставляющий удаленный доступ к географическим пространственным данным (картографической информации) и связанные с ним сервисы.

Данный геопортал предоставляет пользователям большой спектр картографических данных, в том числе, напрямую связанных с экологией.

В разделе «Оперативный мониторинг» данного геопортала были получены данные о концентрации загрязняющего вещества PM2,5 (мелкодисперсная пыль). Полученные данные можно разделить на две части: данные, предоставленные официальным информационно-аналитическим ресурсом министерства экологии и рационального природопользования Красноярского края krasecology.ru, а также данные, предоставляемые открытым картографическим веб-ресурсом krasnoyarsknebo.ru, на котором публикуются данные о уровне концентрации в воздухе мелкодисперсных частиц в 11 точках города Красноярск.

Исходные данные были получены в формате таблиц XLSX. Содержание таблицы состоит из следующих столбцов: дата, пункт наблюдения, lon (долгота), lat (широта), раздел, показатель, значение. Пример исходных данных представлен на рисунке 1.

Дата	Пункт наблюдения	Lon	Lat	Раздел	Показатель	Значение
2017-12-01 00:00:00	Свердловская	92,877846	55,981876	Атмосферный воздух	Взвешенные частицы PM2.5	0,011
2017-12-01 00:00:00	Комсомольский	92,930832	56,062969	Атмосферный воздух	Взвешенные частицы PM2.5	0,014
2017-12-01 00:00:00	Копылова	92,817154	56,013161	Атмосферный воздух	Взвешенные частицы PM2.5	0,026
2017-12-01 00:00:00	Юности	92,983276	56,017902	Атмосферный воздух	Взвешенные частицы PM2.5	0,027
2017-12-01 00:00:00	Павлова	92,947685	55,993896	Атмосферный воздух	Взвешенные частицы PM2.5	0,019
2017-12-01 00:00:00	Ады Лебедевой	92,871704	56,015892	Атмосферный воздух	Взвешенные частицы PM2.5	0,02
2017-12-01 01:00:00	Свердловская	92,877846	55,981876	Атмосферный воздух	Взвешенные частицы PM2.5	0,01
2017-12-01 01:00:00	Комсомольский	92,930832	56,062969	Атмосферный воздух	Взвешенные частицы PM2.5	0,017

Рисунок 1 – Пример исходных данных

Исходные данные необходимо было обработать для загрузки в базу данных. Для этого, исходные данные, при помощи программного обеспечения Excel, были сохранены в формате CSV. За ненадобностью были убраны столбцы «раздел» и «показатель», так как заранее известно, какой вид загрязнителя будет хранится в базе данных.

Формат CSV позволяет хранить табличные данные в текстовом формате, отделяя каждую строку таблицы знаком переноса строки. Пример обработанных исходных данных представлен на рисунке 2.

```
Date;Sensor_Name;Lon;Lat;Value
2017-12-01 00:00:00;Свердловская;92.877846;55.981876;0.011
2017-12-01 00:00:00;Комсомольский;92.930832;56.062969;0.014
2017-12-01 00:00:00;Копылова;92.817154;56.013161;0.026
2017-12-01 00:00:00;Юности;92.983276;56.017902;0.027
2017-12-01 00:00:00;Павлова;92.947685;55.993896;0.019
2017-12-01 00:00:00;Ады Лебедевой;92.871704;56.015892;0.02
2017-12-01 01:00:00;Свердловская;92.877846;55.981876;0.01
2017-12-01 01:00:00;Комсомольский;92.930832;56.062969;0.017
```

Рисунок 2 – Пример обработанных данных

Как видно на примере обработанных данных, значения названий столбцов таблицы находятся на первой строке файла, разделяясь между собой точкой с запятой. А все значения ячеек таблицы располагаются на строках ниже.

Данный формат часто используется для экспорта и импорта табличных данных, так как занимает небольшой объем памяти и относительно легко обрабатывается.

3.1.1 Создание базы данных

Далее, для хранения исходных данных и их обработки была создана база данных на основе СУБД MySQL.

Для создания базы данных используется утилита PhpMyAdmin, включенная в дистрибутив бесплатного программного обеспечения OpenServer. Данная утилита предоставляет возможность управлять всеми созданными внутри её интерфейса базами данных, при этом не лишая возможности использовать для каждой базы данных свою технологию хранения данных. Интерфейс утилиты phpMyAdmin представлен на рисунке 3.

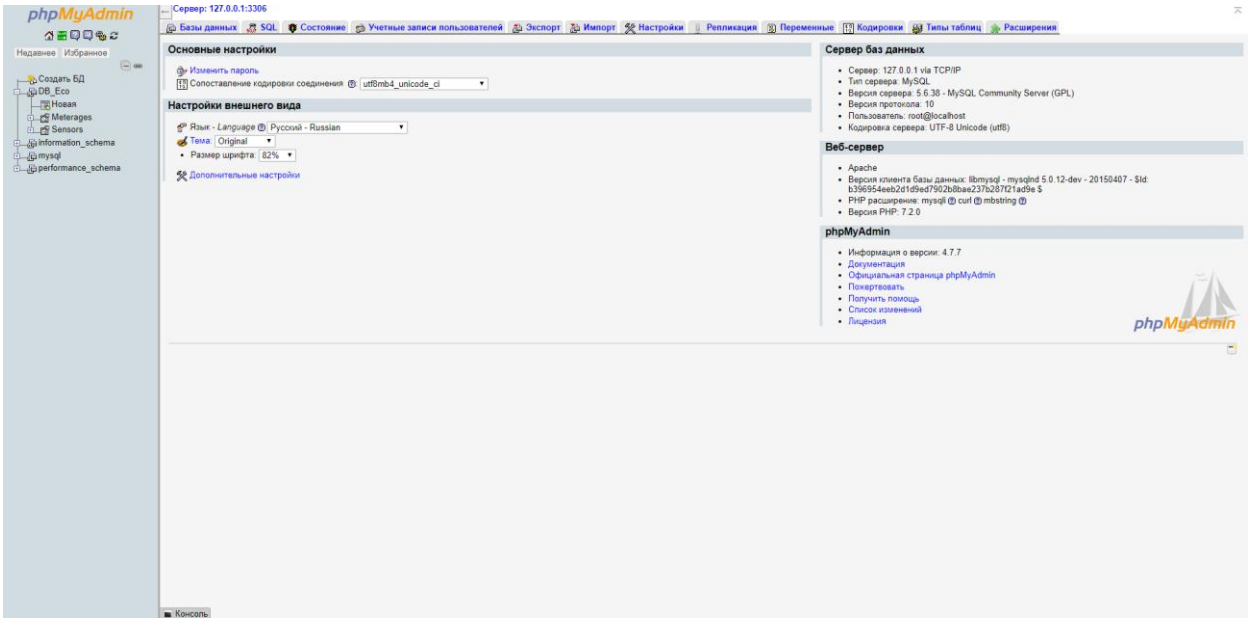


Рисунок 3 – Окно утилиты phpMyAdmin

База данных в данной работе создана на основе технологии MySQL, с использованием подсистемы низкого уровня InnoDB.

Структура созданной базы данных соответствует структуре входных данных и представлена в таблице 1.

Таблица 1 – Структура базы данных

Название столбца	Тип данных	Содержание
ID	Int(11)	Идентификатор замера
Sensor_Name	Varchar(32)	Название измеряющего датчика
Source	Varchar(32)	Источник данных
Lon	Float(10,8)	Долгота измеряющего датчика
Lat	Float(10,8)	Широта измеряющего датчика
Date	Datetime	Дата и время произведения замера
Value	Float(5,3)	Показания измеряющего датчика

В созданную базу данных были загружены данные о измерениях уровня концентрации загрязняющего вещества PM2,5 за период времени с 12.01.2017 по 01.05.2018. Количество полученных в результате заполнения базы данных: 44386 строк. Для того, чтобы получить этот результат, в консоли SQL запроса был отправлен запрос на получение всех строк в текущей базе данных, результат запроса представлен на рисунке 4.

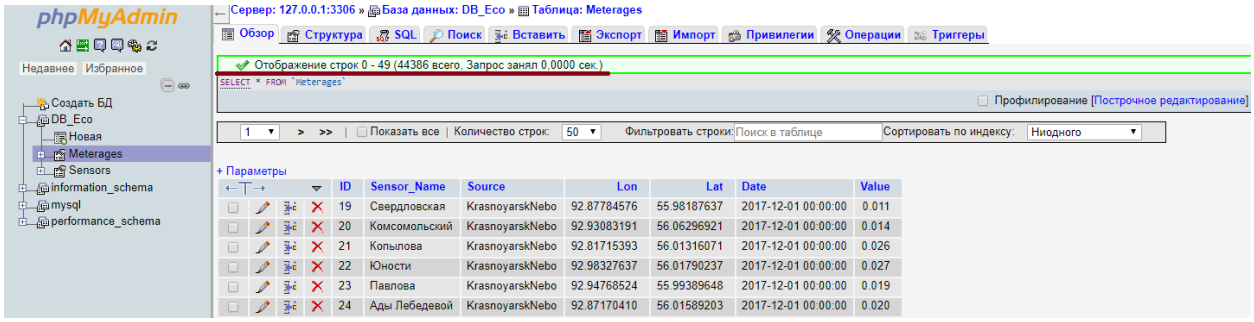


Рисунок 4 – Окно утилиты phpMyAdmin, демонстрирующее количество всех записей в базе данных

Из общего количества записей, записи, полученные измерителями КВИАС (Краевая ведомственная информационно-аналитическая система) – 19387. Результат запроса представлен на рисунке 5.

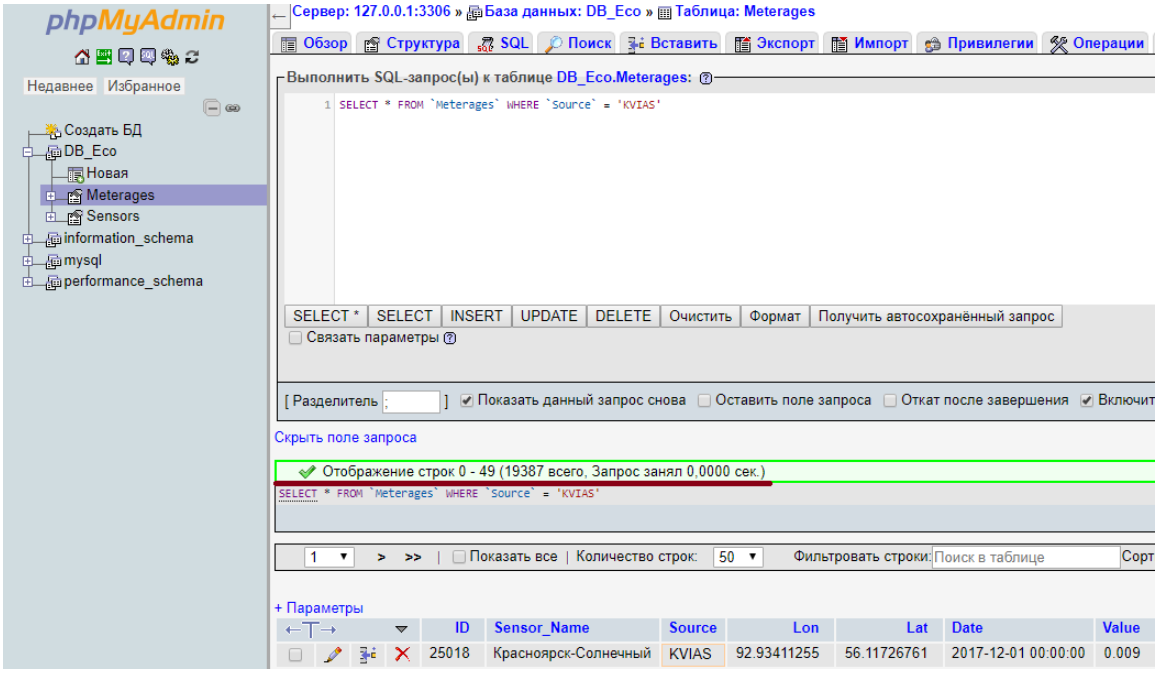


Рисунок 5 – Окно утилиты phpMyAdmin

Количество записей, полученных измерителями экологического проекта Krasnoyarsknebo.ru – 24999. Результат запроса представлен на рисунке 6.

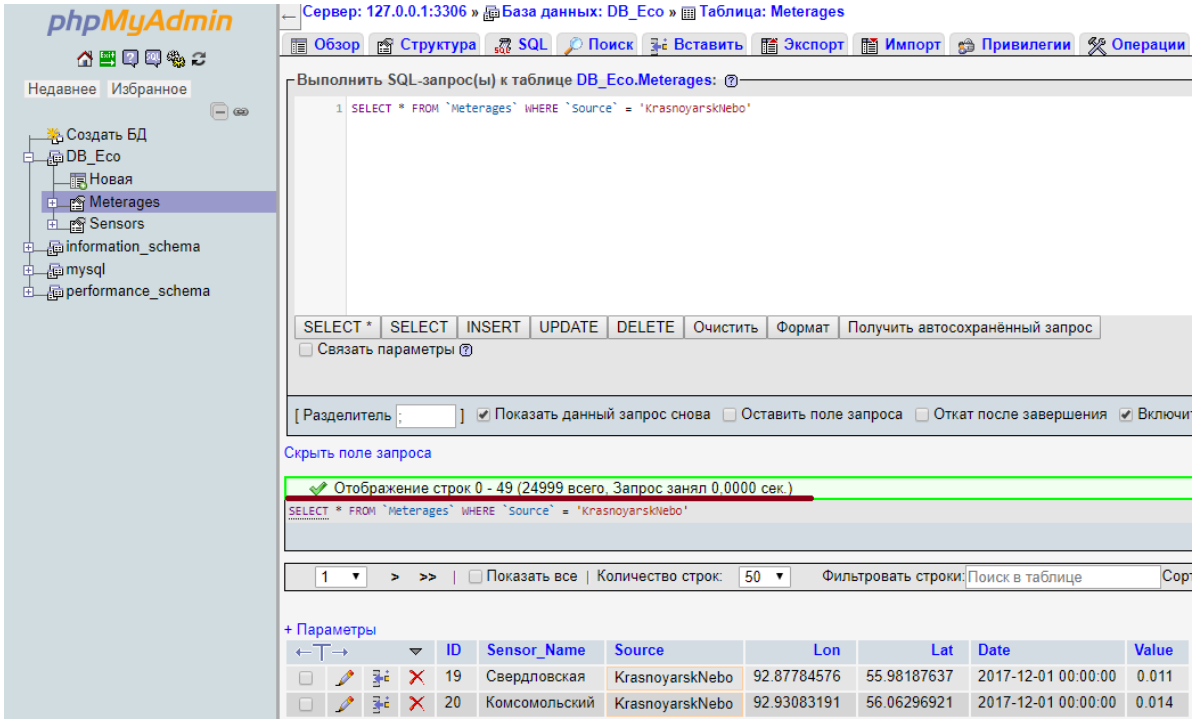


Рисунок 6 – Окно утилиты phpMyAdmin

3.1.2 Создание скрипта для загрузки данных в базу данных

Скрипт для загрузки подготовленных исходных данных в созданную базу данных был создан при помощи серверного языка программирования PHP.

Для того, чтобы исходные данные были успешно загружены в базу данных, необходимо разобрать файл в формате CSV построчно, пропустив первую строку с названиями столбцов, так как в базе данных эти столбцы заранее подготовлены.

Для работы с данными в формате CSV в списке стандартных функций PHP присутствует функция fgetcsv(). Данная функция поочерёдно считывает строки файла, возвращая массив, состоящий из массивов, чьи поля соответствуют значению столбца. Далее необходимо последовательно загрузить полученные строки в базу данных при помощи отправки sql запроса к подключённой базе данных.

Код разработанного цикла для обхода CSV файла и загрузки данных в базу данных продемонстрирован на рисунке 7.

```
while(( $data = fgetcsv ($file, 1000, ';' )) !== FALSE ) && ($counter < $strings_amount)) {

    if ($data[0] != 'Date') {

        echo "<pre>";
        print_r($data);
        echo "</pre>";

        try {

            $sql = "INSERT INTO `Meterages`(`Sensor_Name`, `Source`, `Lon`, `Lat`, `Date`, `Value`) VALUES ('$data[1]'
                , '$dataSource', '$data[2]', '$data[3]', '$data[0]', '$data[4]')";

            $query = $db->prepare($sql);
            $query->execute();

            $info=$query->errorInfo();

            if($info[0] != PDO::ERR_NONE){
                print_r("$info[2]<br>");
            }
        } catch (PDOException $e) {

            print_r("$e<br>");

        }

    }

    $counter++;
}
```

Рисунок 7 – Программный код скрипта для загрузки CSV файла в БД на языке PHP

Полный текст программного кода разработанного скрипта содержится в Приложении А.

3.1.3 Подключение к базе данных

Для того, чтобы подключиться к базе данных со стороны серверной части приложения был разработан скрипт на языке программирования PHP.

Для подключения к любой, созданной на сервере, базе данных, в PHP предусмотрен ряд стандартных классов, с набором собственных методов. На сегодняшний день наиболее универсальным является объект PDO.

PDO (PHP Data Objects) – это прослойка, которая предлагает универсальный способ работы с несколькими базами данных

Код подключения к базе данных продемонстрирован на рисунке 8.

Рисунок 8 – Программный код скрипта для подключения к базе данных на языке PHP

Оператором new создаётся экземпляр объекта класса PDO, в круглых скобках указывается: тип базы данных, имя базы данных, имя пользователя и пароль. Далее производится установка кодировки UTF-8 для того, чтобы в случае несовпадения кодировок между базой данных и отправляемыми данными, кодировка на лету была приведена к указанной в функции.

3.1.4 Разработка класса для работы с базой данных

После написания скрипта для подключения к базе данных, был разработан класс, обрабатывающий запросы, поступающие от клиента и формирующий соответствующий ответ. Данный класс также разработан на языке PHP.

В структуру класса входит конструктор, принимающий в качестве параметров SQL запрос в виде строки и экземпляр созданного подключения к базе данных. Также, в структуру класса входит метод, который отправляет сформированный SQL запрос к базе данных, принимает массив с ответом от базы данных и формирует из него HTML разметку с обработанной таблицей. Программный код данного метода демонстрируется на рисунке 9.

```php
if ($info[2] !== PDO::ERR_NONE && $mod == NULL) {
    $count = 0;
    echo "<table>";
    for($i = 0; $i < count($res); $i++) {
        echo "<tr>";
        foreach ($res[$i] as $key => $value) {
            if (!is_numeric($key)){
                if (is_numeric($value) && $value < 1 && $value >= 0.35) {
                    $indicator = '#EC665F';
                    echo "<td bgcolor=\"$indicator\">$value</td>";
                } elseif(is_numeric($value) && $value < 1 && $value >= 0.16) {
                    $indicator = '#FCD68C';
                    echo "<td bgcolor=\"$indicator\">$value</td>";
                } elseif (is_numeric($value) && $value < 0.16) {
                    $indicator = '#98F5CD';
                    echo "<td bgcolor=\"$indicator\">$value</td>";
                } else {
                    echo "<td bgcolor=\"#EAEAEA\">$value</td>";
                }
            }
        }
        $count++;
        echo "</tr>";
    }
```

Рисунок 9 – Цикл обработки данных, полученных в ответе от базы данных на языке PHP

Обработка исходных данных, полученных в ответе от базы данных, производится по следующему принципу: если в ответе от сервера содержатся поля, со значением, полученным датчиком при измерении концентрации загрязнения, тогда данное значение сравнивается со значением ПДКраз и, в зависимости от результата, ячейка со значением оформляется различным образом.

Полное содержание программного кода разработанного класса содержится в приложении Б.

3.2 Разработка HTML структуры страницы

Задача разработки HTML структуры веб-приложения в данной работе состояла в том, чтобы предложить пользователю такой интерфейс, который он мог бы использовать, словно он работает в обычном настольном приложении. Для каждого функционального элемента веб-приложения было выделено заранее определённое место на странице, так, например, меню – располагается в левой части страницы. Демонстрация внешнего вида веб-приложения представлена на рисунке 10.

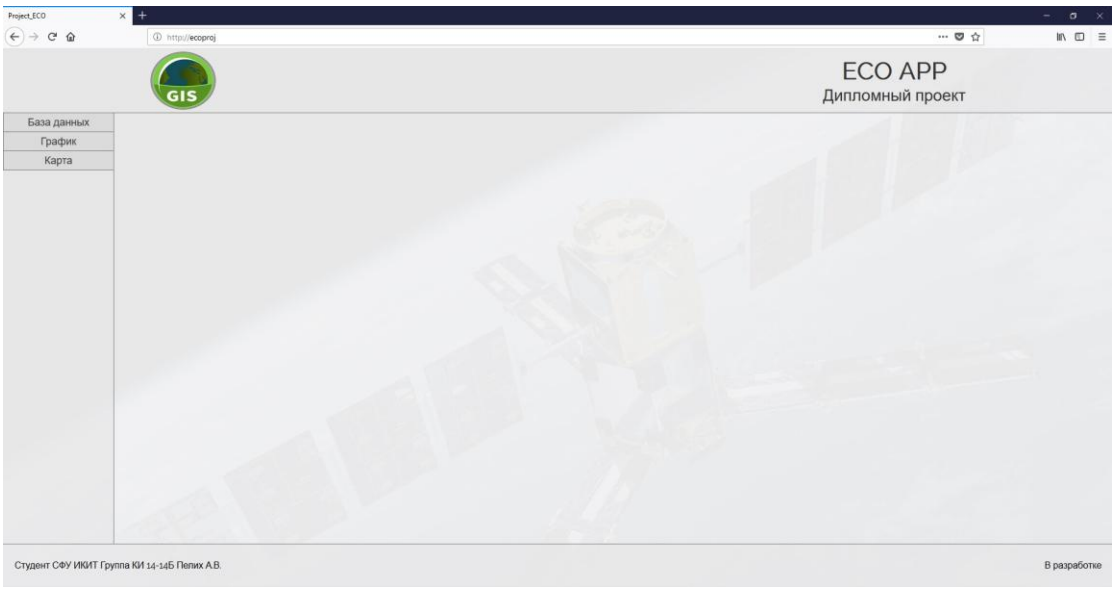


Рисунок 10 – Интерфейс разработанного веб-приложения, открытый в окне браузера

Содержание всех функциональных модулей отображается в центральной части веб-страницы. Нижняя часть веб-страницы отведена для вывода информационных сообщений и контактных данных.

Для каждого функционального компонента разработанного веб-приложения также разработана уникальная HTML структура. Пользователь, по клику левой кнопкой мыши на выбранном пункте меню, выбирает какой компонент он хочет увидеть.

На текущий момент, в веб-приложении реализованы три функциональных компонента. Наведя указатель мыши на первый пункт меню «Базы данных», пользователь увидит два подпункта на выбор: «Построить таблицу» и «Построить таблицу без SQL». Структура выпадающего меню показана на рисунке 11.

Рисунок 11 – Структура выпадающего меню

Полное содержание HTML структуры веб-приложения представлено в приложении В. Далее будут рассмотрены функциональные компоненты по-отдельности.

3.2.1 HTML структура компонента построения таблиц

Разработка HTML структуры компонента построения таблицы заключалась в создании удобного и понятного интерфейса для ввода параметров, которые будут переданы базе данных и вывода в удобном виде

построенной PHP скриптом таблицы. Внешний вид компонента представлен на рисунке 12.

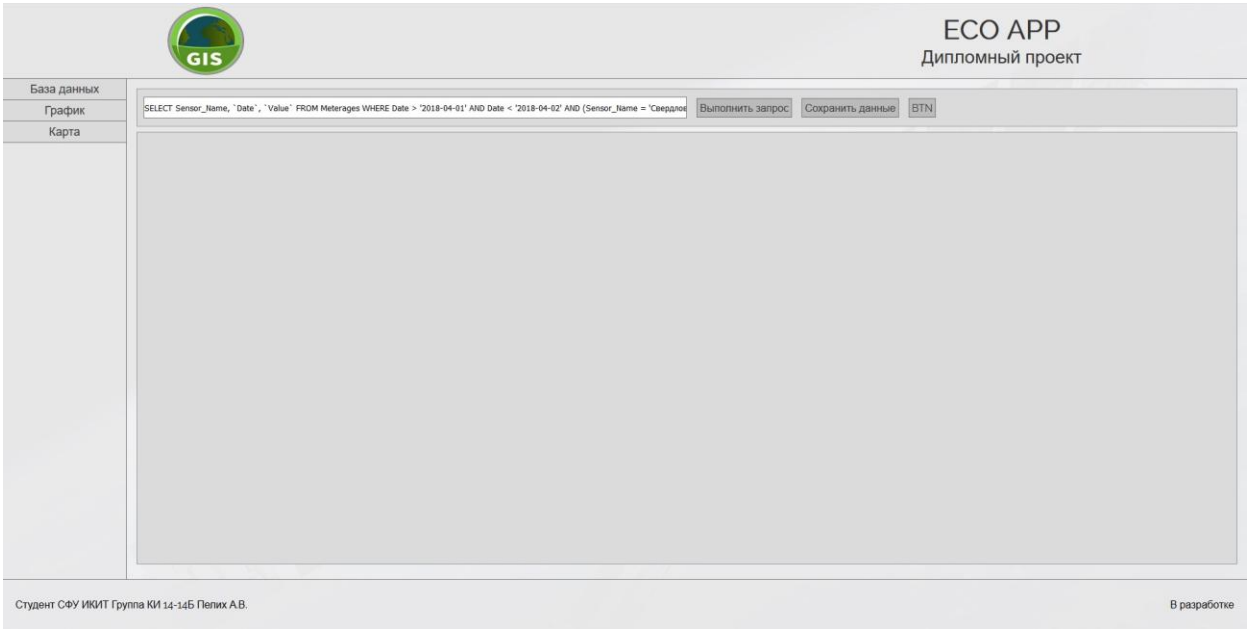


Рисунок 12 – Внешний вид компонента вывода таблицы.

Элементы управления, среди которых: поле для ввода параметров, кнопка для вывода сформированной таблицы, кнопка для сохранения выведенной информации в память, расположены в верхней части компонента. Поле с выводом сформированной таблицы занимает большую часть компонента и находится ниже элементов управления.

CSS код описания стилей компонента представлен в приложении Д.

3.2.2 HTML структура компонента для построения таблиц без SQL запроса

Следующий компонент предоставляет возможность пользователю выбрать параметры для вывода таблицы без использования SQL запроса. В компонент добавлено поле для ввода названия интересующих станций контроля, а также поля для выбора даты первого измерения и даты последнего измерения. Внешний вид компонента представлен на рисунке 13.

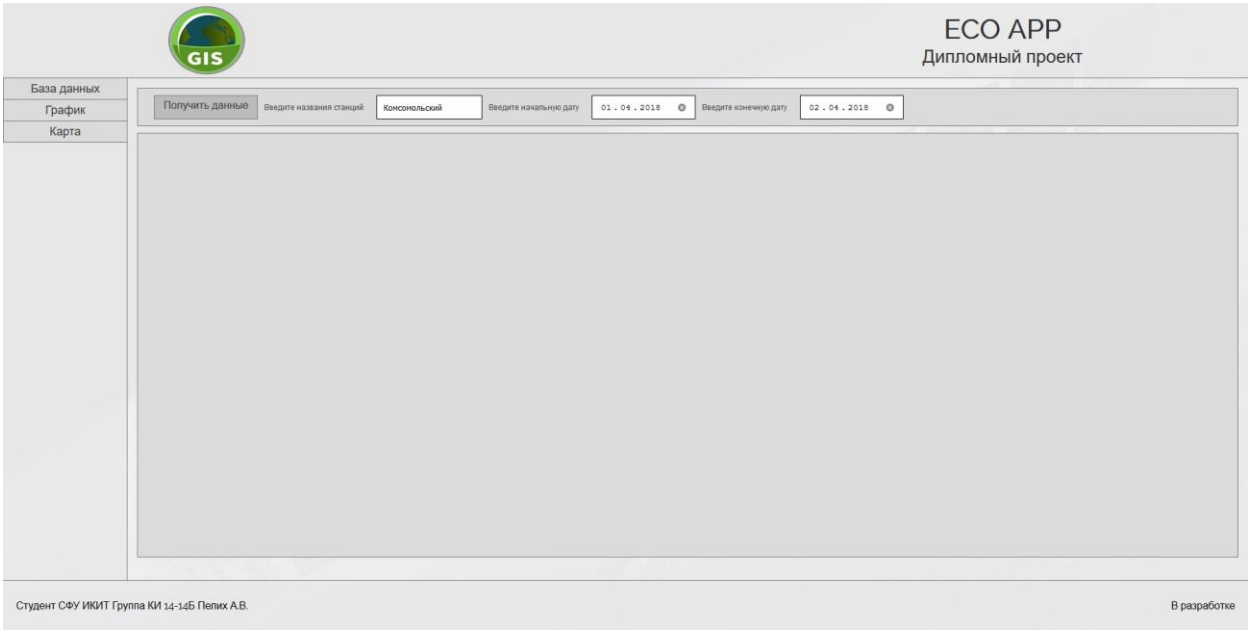


Рисунок 13 – Внешний вид компонента для построения таблиц без использования SQL запроса

Полный файл описания CSS стилей компонента представлен в Приложении Д.

3.3 Создание основной логики для веб-приложения

Для создания логики работы веб-приложения использовались два основных инструмента: JavaScript и Vue.js. Использование подобных современных инструментов позволяет создавать веб-приложения, способные обеспечить такой-же уровень реактивности и интерактивности веб-приложения, как и при работе с приложениями, разработанными для настольных компьютеров.

Используемый в данной работе фреймворк Vue.js позволяет создавать сложные приложения, в которых содержится большое количество взаимосвязанных компонентов, за достаточно короткое время.

Для того, чтобы начать использовать Vue.js, в проекте нужно выбрать основной компонент, в данном случае, таким является всё приложение, за исключением «заголовка» сайта и его «подвала».

На рисунке 14 показан программный код инициализации компонента на языке JavaScript с использованием фреймворка Vue.js.

```
vueApp = new Vue({

    el: '#vue-app',

    data:{

        queryValue: "SELECT Sensor_Name, `Date`, `Value` FROM Meterages WHERE Date > '2018-04-01' AND Date < '2018-04-02'
        "AND (Sensor_Name = 'Свердловская' OR Sensor_Name = 'Копылова')",

        showTable: false,

        showChart: false,

        showMap: false,

        mapBoxAccessKey: 'pk.eyJ1IjoiaW5zeW4iLCJhIjoiY2ppYm9qcTRiMDdoeTNxcDc0YnV3MWNtdCJ9.ilVIytsG9p7hY8pob1c9ig',

        tableData: {},

        chartNames: 'Комсомольский',

        chartSDate:'2018-04-01',

        chartEDate:'2018-04-02',

        chartValues:[],

        sql: '',

        showChartG:false

    },
```

Рисунок 14 – Программный код инициализации объекта Vue.js

На рисунке 14 видно, что при инициализации объекта Vue.js, создаются поля, которые являются объектами. Так, например, объект «el» - содержит ссылку на HTML структуру, внутри которой находится описываемое приложение. А объект «data» содержит в себе переменные, которые будут использоваться приложением в ходе работы.

Для того, чтобы сделать элементы HTML структуры реактивными, в их HTML структуру встраиваются так называемые директивы.

Для примера использования директив в данной работе, можно привести принцип взаимодействия между элементами меню веб-приложения и теми компонентами, которые мы хотим открыть.

На рисунке 15 приведён код HTML разметки элемента меню «Построить таблицу».

Рисунок 15 – Код HTML структуры элемента меню «Построить таблицу»

На рисунке 15 красными линиями подчёркнуты: сам HTML элемент и директива «@click» (сокращенная форма записи v-on:click). Данная директива устанавливает для элемента следующий паттерн поведения – в момент, когда на элементе сработает событие «click» (нажатие левой кнопкой мыши), будет вызвана функция, указанная далее в скобочках, в данном случае – «showTable()».

На рисунке 16 продемонстрирован программный код функции «showTable()».

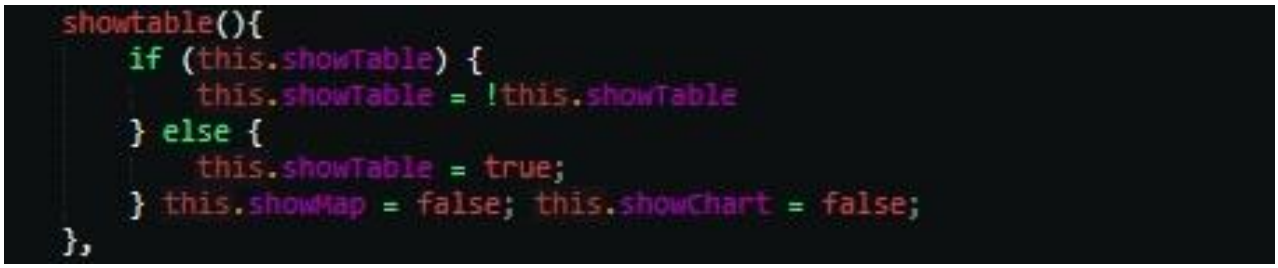


Рисунок 16 – Программный код функции «showTable()» на языке JavaScript

Изображённая на рисунке 16 функция отвечает за то, чтобы в случае нажатия пользователем на пункт меню «Построить таблицу» был открыт только компонент для вывода таблицы, а все другие скрыты.

3.3.1 Создание логики для компонента вывода таблиц

Компонент вывода таблиц с информацией о концентрации загрязняющего вещества в атмосфере работает следующим образом: пользователь пишет в текстовом поле ввода SQL запрос с параметрами, данный SQL запрос сохраняется в переменную после каждого ввода символа, затем пользователь нажимает на элемент «Выполнить запрос». Далее срабатывает функция «createResponse()», написанная на языке JavaScript, её программный код представлен на рисунке 17.

Рисунок 17 – Программный код функции «createResponse()»

Данная функция отправляет на сервер ajax запрос, по адресу, указанному в переменной «preURL», с переданной в качестве GET параметра строкой SQL запроса, и функцией коллбэком – данная функция сработает после того, как придёт ответ от сервера.

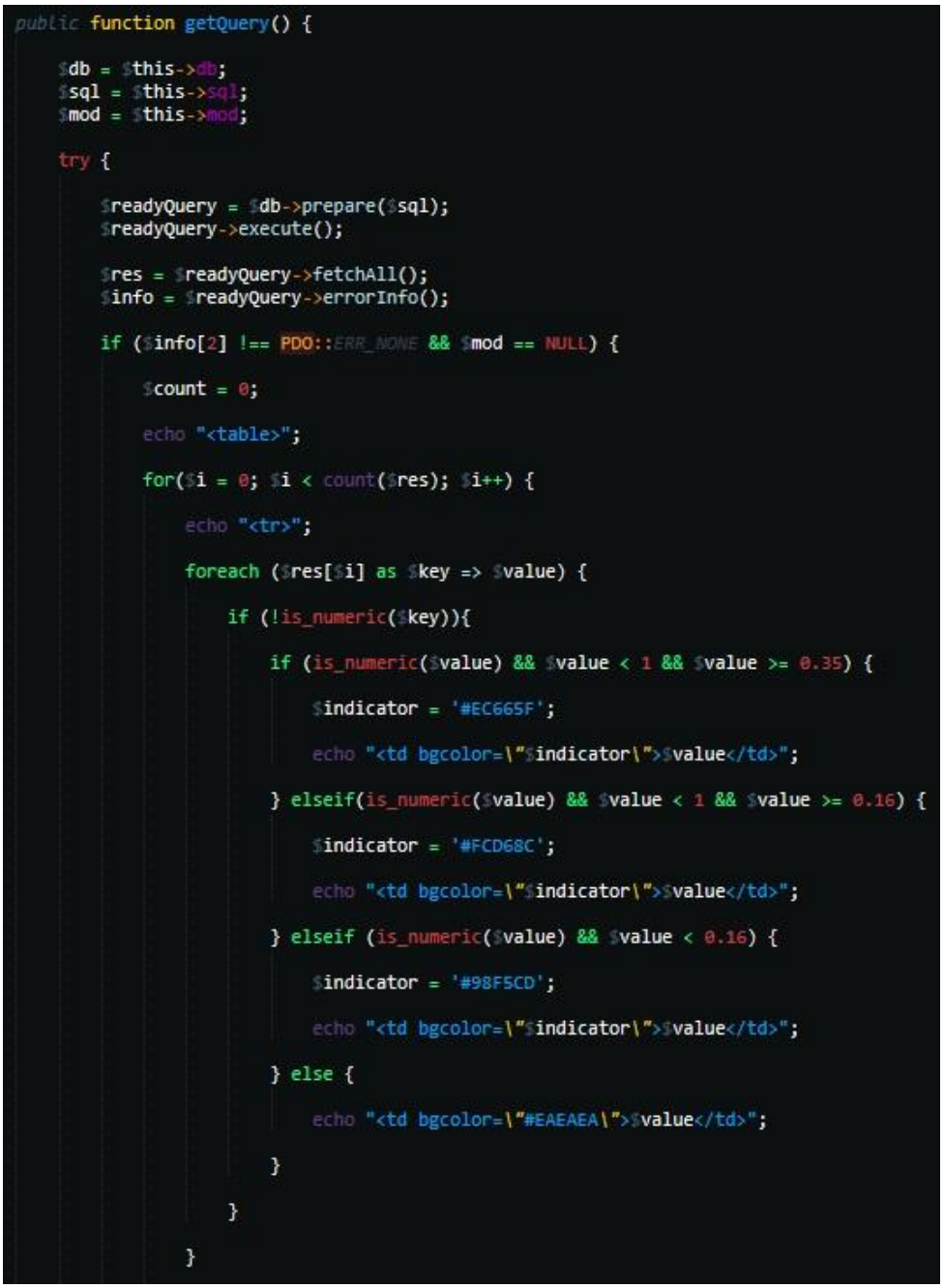
Далее, на стороне сервера вызывается скрипт, указанный в переменной «preURL». Программный код данного скрипта демонстрируется на рисунке 18.

Рисунок 18 – Программный код функции обработки ajax запроса для вывода таблицы

Данная функция, вызванная ajax запросом, проверяет массив GET параметров на наличие в ней строки с SQL запросом, после чего зоздается экземпляр класса «Query» и вызывается метод данного класса «getQuery».

Класс «Query» был создан для данной работы с целью получить универсальный инструмент для обработки запросов от клиентской части веб-приложения, направленных на получение данных из базы данных.

Экземпляр данного класса принимает в качестве параметра строку SQL запроса и экземпляр подключения к базе данных. Вызыаемый метод «getQuery()» обрабатывает строку SQL запроса методом объекта PDO «prepare» таким образом, чтобы были экранированы все потенциально-опасные символы и отправляет запрос к базе данных методом объекта PDO «execute». Программный код метода «getQuery()» представлен на рисунке 19.



```
public function getQuery() {

    $db = $this->db;
    $sql = $this->sql;
    $mod = $this->mod;

    try {

        $readyQuery = $db->prepare($sql);
        $readyQuery->execute();

        $res = $readyQuery->fetchAll();
        $info = $readyQuery->errorInfo();

        if ($info[2] !== PDO::ERR_NONE && $mod == NULL) {

            $count = 0;

            echo "<table>";

            for($i = 0; $i < count($res); $i++) {

                echo "<tr>";

                foreach ($res[$i] as $key => $value) {

                    if (!is_numeric($key)){

                        if (is_numeric($value) && $value < 1 && $value >= 0.35) {

                            $indicator = '#EC665F';

                            echo "<td bgcolor=\"$indicator\">$value</td>";

                        } elseif(is_numeric($value) && $value < 1 && $value >= 0.16) {

                            $indicator = '#FCD68C';

                            echo "<td bgcolor=\"$indicator\">$value</td>";

                        } elseif (is_numeric($value) && $value < 0.16) {

                            $indicator = '#98F5CD';

                            echo "<td bgcolor=\"$indicator\">$value</td>";

                        } else {

                            echo "<td bgcolor=\"#EAEAEA\">$value</td>";

                        }

                    }

                }
```

Рисунок 19 – Программный код метода «getQuery»

После подготовки и отправки запроса к базе данных, метод getQuery преобразовывает полученный от базы данных в новый объект при помощи функции объекта PDO «prepare()». Созданный объект содержит в себе строки, полученные от базы данных, в виде объектов, в качестве полей у которых – названия столбцов, а значения – значение соответсвующей ячейки в базе данных.

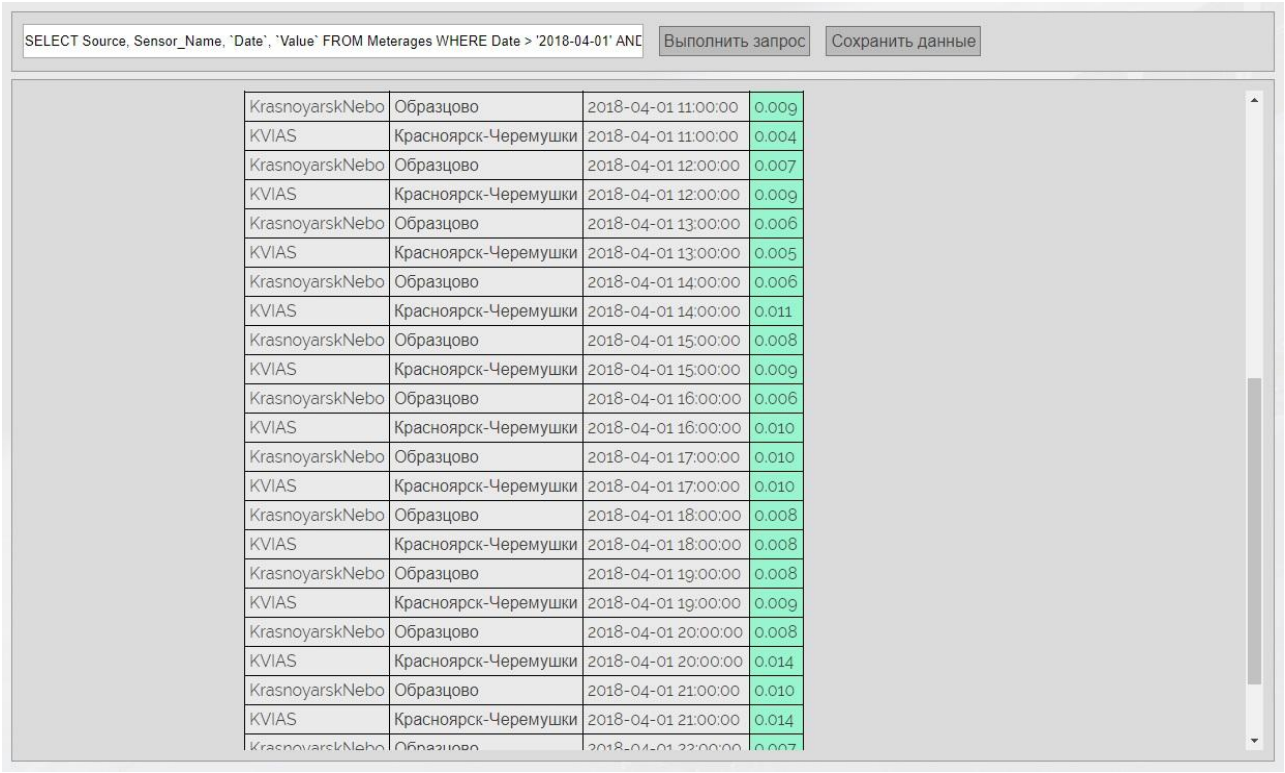
Далее, после получения ответа от базы данных, для каждого объекта-строки запускается цикл, формирующий HTML структуру таблицы, которая будет отправлена на сорону клиента в виде HTML документа. В данном цикле осуществляется проверка значений концентрации загрязняющего вещества и, в случае превышения ПДКсс или ПДКраз ячейке с соответствующим значением будет задан CSS стиль, оформляющий цвет заднего фона зелёным, жёлтым, или красным цветом соответственно.

Полученная от сервера HTML структура подготовленной скриптом таблицы обрабатывается колбэк функцией, которая была указана при вызове ajax запроса, в данном случае срабатывет функция, продемонстрированная на рисунке 17. Данная функция просто распечатывает полученную от сервера таблицу во внутреннее содержимое компнонента вывода таблицы.

3.3.2 Результаты работы созданной системы

Полученный от сервера HTTL документ встраивается в структуру HTML структуру веб-приложения и к ней автоматически применяются CSS стили, описанные в отдельном файле для оформления таблиц.

Результат работы системы продемонстрирован на рисунке 20.



KrasnoyarskNebo	Образцово	2018-04-01 11:00:00	0.009
KVIAS	Красноярск-Черемушки	2018-04-01 11:00:00	0.004
KrasnoyarskNebo	Образцово	2018-04-01 12:00:00	0.007
KVIAS	Красноярск-Черемушки	2018-04-01 12:00:00	0.009
KrasnoyarskNebo	Образцово	2018-04-01 13:00:00	0.006
KVIAS	Красноярск-Черемушки	2018-04-01 13:00:00	0.005
KrasnoyarskNebo	Образцово	2018-04-01 14:00:00	0.006
KVIAS	Красноярск-Черемушки	2018-04-01 14:00:00	0.011
KrasnoyarskNebo	Образцово	2018-04-01 15:00:00	0.008
KVIAS	Красноярск-Черемушки	2018-04-01 15:00:00	0.009
KrasnoyarskNebo	Образцово	2018-04-01 16:00:00	0.006
KVIAS	Красноярск-Черемушки	2018-04-01 16:00:00	0.010
KrasnoyarskNebo	Образцово	2018-04-01 17:00:00	0.010
KVIAS	Красноярск-Черемушки	2018-04-01 17:00:00	0.010
KrasnoyarskNebo	Образцово	2018-04-01 18:00:00	0.008
KVIAS	Красноярск-Черемушки	2018-04-01 18:00:00	0.008
KrasnoyarskNebo	Образцово	2018-04-01 19:00:00	0.008
KVIAS	Красноярск-Черемушки	2018-04-01 19:00:00	0.009
KrasnoyarskNebo	Образцово	2018-04-01 20:00:00	0.008
KVIAS	Красноярск-Черемушки	2018-04-01 20:00:00	0.014
KrasnoyarskNebo	Образцово	2018-04-01 21:00:00	0.010
KVIAS	Красноярск-Черемушки	2018-04-01 21:00:00	0.014
KrasnoyarskNebo	Образцово	2018-04-01 22:00:00	0.007

Рисунок 20 – Результат работы компонента вывода таблицы в окне браузера

На данном изображении показан результат работы компонента вывода таблицы после отправки запроса на сервер. В данном случае, в качестве вывода получена таблица, на которой показаns сводные данные полученные от двух разных источников: датчика находящегося в районе Черемушки, принадлежащего министерству экологического развития и рационального природопользования Красноярского края, и датчика находящегося в районе Образцово, принадлежащего организации krasnoyarsk.nebo.

На таблице продемонстрированы данные за первое апреля 2018 года, в период с 11:00 до 21:00. Так как выбранные датчики находятся на расстояни не более 200 метров друг от друга, можно сравнивать их показания, делая вывод о том, на сколько показания несертифицированного датчика, принадлежащего организации krasnoyarsk.nebo отличаются от показаний официального сертифицированного датчика.

Для получения данной таблицы использовался SQL запрос следующего содержания: «SELECT Source, Sensor_Name, `Date`, `Value` FROM Meterages WHERE Date > '2018-04-01' AND Date < '2018-04-02' AND (Sensor_Name = 'Красноярск-Черемушки' OR Sensor_Name = 'Образцово') order by `Date` Desc».

Для проверки суммарного количества измерений, показавших превышение уровня ПДКраз. За период с 01.01.2018 по 01.05.2018 был составлен другой запрос, со следующим содержанием: «SELECT `Source`, `Sensor_Name`, `Date`, `Value` FROM Meterages WHERE Date > '2018-01-01' AND Date < '2018-05-01' AND `Value` > 0.35 order by `Date`, `Sensor_Name` Desc ». Результат представлен на рисунке 21.

KrasnoyarskNebo	Павлова	2018-01-26 21:00:00	0.362
KrasnoyarskNebo	Павлова	2018-01-26 22:00:00	0.357
KrasnoyarskNebo	Павлова	2018-02-02 22:00:00	0.361
KrasnoyarskNebo	Павлова	2018-02-02 23:00:00	0.396
KrasnoyarskNebo	Перенсона	2018-02-03 21:00:00	0.354
KrasnoyarskNebo	Павлова	2018-02-03 21:00:00	0.454
KrasnoyarskNebo	Копылова	2018-02-03 21:00:00	0.360
KrasnoyarskNebo	Перенсона	2018-02-03 22:00:00	0.360
KrasnoyarskNebo	Павлова	2018-02-03 22:00:00	0.441
KrasnoyarskNebo	Павлова	2018-02-03 23:00:00	0.504
KrasnoyarskNebo	Павлова	2018-02-04 00:00:00	0.465
KrasnoyarskNebo	Павлова	2018-02-04 01:00:00	0.390
KrasnoyarskNebo	Павлова	2018-02-04 02:00:00	0.406
KrasnoyarskNebo	Копылова	2018-02-04 19:00:00	0.378
KrasnoyarskNebo	Ады Лебедевой	2018-03-17 20:00:00	0.396
KVIAS	Красноярск-Березовка	2018-03-27 12:00:00	0.386

Рисунок 21 – Результат работы компонента вывода таблицы

На полученном изображении видно, что общее количество превышений уровня ПДКраз. По данных обоих источников равняется 16. Также можно обратить внимание на то, что 15 из 16 превышений показывают датчики, установленные организацией krasnoyarsk.nebo.

4 Направление дальнейших разработок

В дальнейших планах предполагается расширение и развитие уже существующего веб-приложения. В систему будут добавлены модули, необходимые для более глубокого анализа и оценки существующих данных о загрязнении атмосферы города Красноярск. Среди таких модулей можно перечислить:

- компонент для быстрого построения графиков, с использованием библиотеки D3.js, позволяющий более наглядно оценить соотношение между показаниями датчиков двух разных организаций, либо находящихся в разных местах
- модуль для построения карты города Красноярск, с возможностью накладывать генерируемые на основе данных из модуля построения таблиц слои в формате GeoJSON с использованием библиотеки Leaflet.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной дипломной работы было создано веб-приложение, основанное на наборе современных технологий веб-программирования, среди которых такие как: PHP, JavaScript, Vue.js и другие.

Возможности данного веб-приложение заключаются в динамическом выводе оформленных таблиц, на основе введённых пользователем параметров, с целью оценки и анализа данных о концентрации загрязняющего вещества PM2.5 в атмосфере города Красноярск.

В данной работе создан полноценный каркас, разработанный с учётом современных тенденций в веб-программировании, обеспечивающий возможность дальнейшего насыщения проекта новыми возможностями и модулями. Все элементы веб-приложения работают как части полноценного десктопного приложения, без необходимости перезагружать страницу, или наблюдать зависание браузера в момент ожидания ответа от сервера.

Также, в ходе работы был получен значительный опыт в разработке веб-приложений и усовершенствованы навыки владения целым набором веб-технологий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Информационное агентство России «ТАСС» [Электронный ресурс]: – Режим доступа: http://tass.ru/obschestvo/4804434

2 Сайт, посвящённый году экологии в России [Электронный ресурс]: – Режим доступа: http://ecoyear.ru/documentation/ukaz7/

3 Портал «Наука и жизнь» [Электронный ресурс]: – Режим доступа: https://www.nkj.ru/archive/articles/10376/

4 Портал министерства экологии и рационального природопользования Красноярского края [Электронный ресурс]: – Режим доступа: http://krasecology.ru

5 Геопортал ИВМ СО РАН [Электронный ресурс]: – Режим доступа: http://gis.krasn.ru/blog

6 Справочник для языка программирования JavaScript [Электронный ресурс]: – Режим доступа: http://javascript.ru/manual

7 Документация языка программирования JavaScript [Электронный ресурс]: – Режим доступа: http://developer.mozilla.org/ru/docs/Web/JavaScript

8 Документация серверного языка программирования PHP [Электронный ресурс]: – Режим доступа: http://php.net/manual/ru

9 Руководство пользователя для фреймворка Vue.js [Электронный ресурс]: – Режим доступа: https://ru.vuejs.org/v2/guide/

ПРИЛОЖЕНИЕ А

HTML структура приложения

Index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<title>Project_ECO</title>
	<link rel="shortcut icon" href="">
	<link rel="stylesheet" href="css/Style.css">
	<link rel="stylesheet" href="css/media.css">
	<link rel="stylesheet" href="css/Interactions.css">
	<link rel="stylesheet" href="css/nv.d3.css">

	<link rel="stylesheet" href="css/Tables-window-styles.css">
	<link rel="stylesheet" href="css/Chart-window-styles.css">
	<link rel="stylesheet" href="css/Map-window-styles.css">

	<link rel="stylesheet" href="https://unpkg.com/leaflet@1.3.1/dist/leaflet.css"
  integrity="sha512-Rksm5RenBEKSKFjgI3a41vrjkw4EVPlJ3+OiI65vTjIdo9brlAacEuKOiQ5OFh7cOI1bkDwLqdLw3Zg0cRJAAQ=="
  crossorigin=""/>
	<script src="https://unpkg.com/leaflet@1.3.1/dist/leaflet.js"
  integrity="sha512-/Nsx9X4HebavoBvEBuyp3I7od5tA0UzAxs+j83KgC8PU0kgB4XiK4Lfe4y4cgBtaRJQEIFCW+oC506aPT2L1zw=="
  crossorigin=""></script>

	<link href="https://fonts.googleapis.com/css?family=Raleway" rel="stylesheet">
</head>
<body>
	<header>
		<div class="wrapper">
			<div class="picture-wrapper">
				<a href="http://e.sfu-kras.ru/" target="blank"><img src="img/GIS_Logo1.png" class="picture" alt="">
				</a>

			</div>
			<div class="header-text">
				<h2>ECO APP</h2>
				<p>Дипломный проект</p>
			</div>
		</div>
	</header>
	<div class="main-wrapper" id="vue-app">
		<nav>
			<div class="nav-wrapper">
				<div class="menu-item maps">
					<p>База данных</p>
					<ul class="menu-item-ul">
						<li class="menu-item-li item-1" @click="showtable">Постороить таблицу
						</li>
						<li class="menu-item-li item-1" @click="showchart">Постороить таблицу без
SQL
						</li>
					</ul>
				</div>
				<div class="menu-item maps">
					<p>График</p>
					<ul class="menu-item-ul">
						<li class="menu-item-li item-3"
							@click='chartQuery'>
							Построить график
						</li>
					</ul>
				</div>
				<div class="menu-item maps">
					<p>Карта</p>
					<ul class="menu-item-ul">
						<li class="menu-item-li item-5"
							@click="showmap"
							@click="if(showMap){createMap}">Показать карту
						</li>
```

```
                                        </ul>
                                    </div>
                                </div>
                            </nav>
                            <main>
                                <div class="wrapper">
                                    <div class="tables-window" v-if="showTable">
                                        <div class="tables-window-item input-wrapper">
                                            <input type="text" class="query-input"
                                                   v-model=queryValue>
                                            <div class="query-accept-button table-button"
                                                   v-on:click='createResponse'>
                                                Выполнить запрос
                                            </div>
                                            <div class="save-data-button table-button"
                                                   @click="saveTableData">
                                                Сохранить данные
                                            </div>
                                            <div style="display: none;"
                                                class="table-button"
                                                @click="showRes"> BTN </div>
                                        </div>
                                        <div class="tables-window-item output-wrapper">
                                            <div style="width: 100%; height: 100%; overflow: auto;">
                                            <div class="printed-table">
                                            </div>
                                            </div>
                                        </div>
                                    </div>
                                    <div class="chart-window" v-if="showChart">
                                        <div class="chart-input-section">
                                            <div class="build-chart-button table-button">
                                                   Получить данные
                                            </div>

                                            <div class="build-chart-button table-button"
                                                 @click="createChart"
                                                 style="">
                                                   Построить график
                                            </div>

                                            <span @click = "showChartG = !showChartG">Введите названия станций</span>
                                            <input type="text"
                                                   class="name-input chart-input"
                                                   v-model="chartNames">

                                            <span>Введите начальную дату</span>
                                            <input type="date"
                                                   class="start-date-input chart-input"
                                                   v-model="chartSDate">

                                            <span>Введите конечную дату</span>
                                            <input type="date"
                                                   class="end-date-input chart-input"
                                                   v-model="chartEDate">
                                        </div>
                                        <div class="chart-output-section">
                                            <svg id="chart" class="nv-chart" v-if="showChartG" >
                                            </svg>
                                        </div>
                                    </div>
                                    <div class="map-window"
                                        v-if="showMap">
                                        <div class="map-wrapper">
                                            <div class="show-map-button"
                                                 @click="createMap">
                                            <span>Показать карту</span>
                                            </div>
                                            <div id="leafletMap" ></div>
                                        </div>
                                    </div>
                                </div>
                            </main>
                        </div>
                        <script src="js/vue/vue.js"></script>
                        <script src="js/nv/d3.v3.js"></script>
                        <script src="js/nv/nv.d3.js"></script>
```

```
        <script src="js/functions.js"></script>
        <script src="js/script.js"></script>
</body>
</html>
```

ПРИЛОЖЕНИЕ Б

Программный код класса для обработки запросов «Query»

Queries.php

```
<?php
	class Query {
		private $db;
		private $sql;
		private $mod;

		public function __construct($_db, $_sql, $_mod = NULL) {
			$this->db = $_db;
			$this->sql = $_sql;
			$this->mod = $_mod;
		}

		public function getQuery() {
			$db = $this->db;
			$sql = $this->sql;
			$mod = $this->mod;
			try {
				$readyQuery = $db->prepare($sql);
				$readyQuery->execute();

				$res = $readyQuery->fetchAll();
				$info = $readyQuery->errorInfo();

				if ($info[2] !== PDO::ERR_NONE && $mod == NULL) {
					$count = 0;
					echo "<table>";
					for($i = 0; $i < count($res); $i++) {
						echo "<tr>";
						foreach ($res[$i] as $key => $value) {
							if (!is_numeric($key)){
								if (is_numeric($value) && $value < 1 && $value >= 0.35) {
									$indicator = '#EC665F';
									echo "<td bgcolor=\"$indicator\">$value</td>";

								} elseif(is_numeric($value) && $value < 1 && $value >= 0.16) {
									$indicator = '#FCD68C';
									echo "<td bgcolor=\"$indicator\">$value</td>";

								} elseif (is_numeric($value) && $value < 0.16) {
									$indicator = '#98F5CD';
									echo "<td bgcolor=\"$indicator\">$value</td>";

								} else {
									echo "<td bgcolor=\"#EAEAEA\">$value</td>";
								}
							}
						}
						$count++;
```

```
                                        echo "</tr>";
                                    }
                                    echo "</table>";
                                    echo "<span>Количество измерений $count</span>";

                                    } elseif ($mod == 'c') {

                                        for($i = 0; $i < count($res); $i++) {

                                            foreach ($res[$i] as $key => $value) {

                                                if ($key == 'Date' && !is_numeric($key)){

                                                    $d = explode(' ', $value);

                                                    echo "{x:'$value',";

                                                } elseif ($key == 'Value' && !is_numeric($key)
&& $i < count($res)-1){


                                                    echo "y:$value},";

                                                } elseif ($key == 'Value' && !is_numeric($key))
{
                                                    echo "y:$value}";
                                                }
                                            }
                                        }
                                        //print_r ($res);

                                    }
                                }

                                catch(PDOException $e) {

                                    print_r("$e<br>");

                                }
                            }
                        }

?>
```

ПРИЛОЖЕНИЕ В

Программный код основной логики веб-приложения

script.js

```
window.onload = function(){
	vueApp = new Vue({
		el: '#vue-app',
		data:{
			queryValue: "SELECT Sensor_Name, `Date`, `Value` FROM Meterages WHERE Date > '2018-04-01' AND Date < '2018-04-02'	"
			"AND (Sensor_Name = 'Свердловская' OR Sensor_Name = 'Копылова')",

			showTable: false,
			showChart: false,
			showMap: false,

			mapBoxAccessKey: 'pk.eyJ1IjoiaW5zeW4iLCJhIjoiY2ppYm9qcTRiMDdoeTNxcDc0YnV3MWNtdCJ9.ilVIytsG9p7hY8pob1c9ig',

			tableData: {},
			chartNames: 'Комсомольский',
			chartSDate:'2018-04-01',
			chartEDate:'2018-04-02',
			chartValues:[],
			sql: '',
			showChartG:false
		},

		computed:{
			chartNamesArray(){
				return this.chartNames.split(', ');
			}
		},
		methods:{
			showmap(){
				if (this.showMap) {
					this.showMap = !this.showMap
				} else {
					this.showMap = true;
				} this.showChart = false; this.showTable = false;
			},
			showtable(){
				if (this.showTable) {
					this.showTable = !this.showTable
				} else {
					this.showTable = true;
				} this.showMap = false; this.showChart = false;
			},
			showchart(){
				if (this.showChart) {
					this.showChart = !this.showChart
				} else {
					this.showChart = true;
				} this.showMap = false; this.showTable = false;
			},
			createResponse(){
				let preUrl = '../php/db_queries.php';
				let params = this.queryValue;
```

```
		let toSendUrl = preUrl+'?sql='+params;

		ajaxGet(toSendUrl, function(data){
			document.querySelector('.printed-table').innerHTML = data;
		})
},
saveTableData(){
		let preUrl = '../php/saveTable.php';
		let params = this.queryValue;
		let toSendUrl = preUrl+'?sql='+params;
		ajaxGet(toSendUrl, (data)=>{
			jResponse = JSON.parse(data);
			for (k in jResponse){
				Vue.set(this.tableData, k, jResponse[k]);
			};
		})
},
showRes(){
			let inner = document.querySelector('.printed-table').innerHTML;
		for (k in this.tableData) {
			inner += Object.entries(this.tableData[k]) + '<br>';
			document.querySelector('.printed-table').innerHTML = inner;
		}
},
createChart(){
		nv.addGraph(function() {
		  var chart = nv.models.lineChart()
			.margin({left: 100})
			.useInteractiveGuideline(true)
			.showLegend(true)
			.showYAxis(true)
			.showXAxis(true)
		  ;

		  chart.xAxis
		    .axisLabel('Time (ms)')
		    .tickFormat(d3.format(',r'));

		  chart.yAxis
		    .axisLabel('Voltage (v)')
		    .tickFormat(d3.format('.02f'));


		  var myData = sinAndCos();

		  d3.select('#chart')
		    .datum(myData)
		    .call(chart);


		  nv.utils.windowResize(function() { chart.update() });
		  return chart;

		});

		function sinAndCos() {
		  var sin = [],sin2 = [],
		      cos = [];


		  for (var i = 0; i < 100; i++) {
		    sin.push({x: i, y: Math.sin(i/10)});
```

```
                sin2.push({x: i, y: Math.sin(i/10) *0.25 + 0.5});
                cos.push({x: i, y: .5 * Math.cos(i/10)});
              }


              return [
                {
                  values: sin,
                  key: 'Sine Wave',
                  color: '#ff7f0e'
                },
                {
                  values: cos,
                  key: 'Cosine Wave',
                  color: '#2ca02c'
                },
                {
                  values: sin2,
                  key: 'Another sine wave',
                  color: '#7777ff',
                  area: true
                }
              ];
            }
        },

        chartQuery(){

            let queryFirst = "SELECT `Source`,`Sensor_Name`,`Date`, `Value` FROM `Meterages` WHERE `Date`>\'"+this.chartSDate+"\' and `Date`<\'"+this.chartEDate+"\' and ";
            let queryLast = "";

            if (this.chartNamesArray.length > 1) {
                chartNamesArrayLenght = this.chartNamesArray.length;
                console.log(chartNamesArrayLenght);
                firstBracket = '(';
                lastBracket = ')';
                queryLast = queryLast + firstBracket;
                SN = ' Sensor_Name =';
                for (let k = 0; k <= chartNamesArrayLenght-1; k++) {
                    if (k == 0) {

                        queryLast = queryLast + SN + ' \" + this.chartNamesArray[k] + '\";

                    }
                    else {

                        queryLast = queryLast + ' or ' + SN + ' \" + this.chartNamesArray[k] + '\";
                    }
                }
                queryLast = queryLast + lastBracket;

            } else {
                queryLast = 'Sensor_Name = \" + this.chartNamesArray + '\";
            }
            this.sql = queryFirst + queryLast;
            let preChartUrl = '../php/db_queries.php';
            let chartParams = this.sql;
            let toSendChartUrl = preChartUrl+'?sql='+chartParams;
            console.log(toSendChartUrl);
```

```
                ajaxGet(toSendChartUrl, (data)=>{
                        console.log(toSendChartUrl);
                        let splitted = data.split(",");
                        /*for (string in splitted) {

                            newStr = string.replace(/["]/g, '');
                            string = newStr;
                        }*/
                        this.chartValues = '['+data+']';
                        console.log(this.chartValues);
                        console.log(typeof(this.chartValues));

                        document.querySelector('.chart-output-section').innerHTML = data;

                });
            },
            createMap(){

                var leafletMap = L.map('leafletMap').setView([55.994339009763166,92.79757545654581], 14);


        L.tileLayer('https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_token={accessToken}', {
                attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, Imagery © <a href="https://www.mapbox.com/">Mapbox</a>',
                    maxZoom: 18,
                    id: 'mapbox.streets',
                    accessToken: this.mapBoxAccessKey
                }).addTo(leafletMap);

                var myStyle = {
                    "color": "#ff7800",
                    "opacity": 0.65
                };
                ajaxGet('../php/readFile.php', function(data){

                        this.savedGJSON = JSON.parse(data);

                        L.geoJSON(this.savedGJSON, {
                            style: myStyle
                        }).addTo(leafletMap);

                        console.log(this.savedGJSON);
                });
                console.log(this.savedGJSON);

            },
            prepareChartValues(){
                console.log(this.tableData);
            }
        }
    });
}
```

ПРИЛОЖЕНИЕ Г

Вспомогательные функции для веб-приложения

Functions.js

```
function getXmlHttpRequest(){

	if(window.XMLHttpRequest){

		return new XMLHttpRequest();
	}
	else if(window.ActiveXObject) {

		try {

			return new ActiveXObject("Msxml12.XMLHTTP");
		} catch(e) {}

		try {

			return new ActiveXObject("Microsoft.XMLHTTP");
		} catch(e) {}
	}
}

function ajaxGet(url, callback){

	f = callback || function(data){};

	xhr = getXmlHttpRequest();

	xhr.onreadystatechange = function(){

		if (xhr.readyState == 4 && xhr.status == 200){

			f(xhr.responseText);

		}
	}

	xhr.open('GET', url);
	xhr.send();
}
```

ПРИЛОЖЕНИЕ Д

Описание CSS стилей приложения

```
*{
	box-sizing: border-box;
	margin: 0;
	padding: 0;
}

a{
	color: black;
	text-decoration: none;
}

html{
	height: 100%;
	min-width: 365px;
}

body{
	font-family: 'Raleway', sans-serif;
	background-color: #777777;
	background-image: url(../img/anywalls.com-30156.jpg);
	background-size: cover;
	background-attachment: local;
	height: 100%;
}

header{
	display: flex;
	justify-content: space-around;
	background-color: #EAEAEAFC;
	max-width: 100%;
	height: 12%;
	border-bottom: 1px solid #777777;
}

header .wrapper{
	display: flex;
	justify-content: space-between;
	align-items: center;
	width: 75%;
	font-size: 28px;
	padding: 0px 15px;
	margin: 5px 0;
}

.wrapper .picture-wrapper{
	height: 100%;
}

.picture-wrapper .picture{
	height: 100%;
	transition: 200ms;
	cursor: default;
	border: 1px solid #777777;
	border-radius: 100%;
	box-shadow: inset 0 0 10px rgba(0,0,0,0.8);
}
```

```
header .wrapper .header-text{
	display: flex;
	color: #333333FF;
	flex-direction: column;
	align-items: center;
	text-align: center;
}

.main-wrapper{
	display: flex;
	flex-wrap: wrap;
	height: 80%;
	justify-content: space-between;
	align-content: stretch;
}

nav{
	display: flex;
	font-size: 18px;
	flex-direction: column;
	height:calc(100% + 2px);
	width: 10%;
	background-color: #EAEAEAFC;
	border-right: 1px solid #777;
	border-collapse: collapse;
}

nav .nav-wrapper{
	margin-top: 0px;
}

.menu-item{
	display: flex;
	color: #3F3F3FFF;
	align-items: center;
	justify-content: center;
	margin: 0px;
	padding: 5px 0px;
	width: 100%;
	background-color: #DBDBDBFC;
	transition: background-color 300ms;
	border: 1px solid #777777;
	border-right: none;
	border-top: none;
	border-collapse: collapse;
}

.menu-item-ul{
	display: none;
}

.menu-item .maps{
	position: relative;
}

main{
	display: flex;
	color: #3F3F3FFF;
	width: 90%;
	height: 100%;
}
```

```
main .wrapper{
	display: flex;
	flex-wrap: wrap;
	background-color: #F2F2F2F5;
	width: 100%;
	height: 100%;
	height: inherit;
	z-index: 1000;
	overflow: auto;
	justify-content: flex-start;
}

footer{
	display: flex;
	align-items: center;
	background-color: #EAEAEAFC;
	height: 8%;
	width: 100%;
	position: fixed;
	bottom: 0;
	border-top: 1px solid #777777;
}

footer .footer-text{
	display: flex;
	width: 100%;
	padding: 20px;
	justify-content: space-between;
}

.closed{
	display: none !important;
}

.data-frame{
	border: 1px solid #777777;
	padding: 5px;
	margin: 10px;
}

.picture:hover{
	padding: 2px 2px;
	opacity: 0.9;
}

.menu-item:hover{
	flex-direction: column;
	justify-content: space-between;
	cursor: pointer;
	background-color: #ACACAC;
	border-right: 2px solid #262626;
}

.menu-item:hover .menu-item-ul{
	display: block;
	background-color: #ACACAC;
	width: 100%;
}

.menu-item:hover .menu-item-li{
	display: block;
	cursor: pointer;
```

```
        width: 100%;
        text-align: center;
        justify-content: center;
}

.menu-item-li:hover{
        background-color: #999999;
}

.menu-item-li:active{
        background-color: #777777;
}

.chart-window{
        padding: 15px;
        width: 100%;
}

.chart-input-section{
        display: flex;
        text-align: center;
        padding: 10px;
        border:1px solid #777777;
        height: 8%;
        margin-bottom: 10px;
        background-color: #DBDBDB;
}

.chart-output-section{
        display: flex;
        justify-content: center;
        overflow: auto;
        border:1px solid #777777;
        height: 88% !important;
        width: 100%;
        padding: 10px;
        background-color: #DBDBDB;
}

.nv-chart{
        height: 100%;
        width: 100%;
}

.build-chart-button{
        display: flex;
        min-width: 160px;
        justify-content: center;
}

.chart-input{
        padding: 10px;
        border: 1px solid black;
        margin-left: 20px;
        min-width: 160px;
}

.chart-input-section span{
        display: flex;
        justify-content: center;
        align-items: center;
        font-size: 12px;
```

```
        margin-left: 10px;

}
.map-window{
        position: relative;
        width: 100%;
        height: 100%;
        padding: 15px;
}

.map-wrapper{
        height: inherit;
        width: inherit;
        border: 1px solid #777777;
        display: flex;
}

#leafletMap{
        height: inherit;
        width: inherit;
}

.show-map-button{
        padding: 5px;
        background-color: #999;
        cursor: pointer;
        position: absolute;
}

table, tr, td{
        border: 1px solid black;
        border-collapse: collapse;
        padding: 4px;
}

.tables-window{
        display: flex;
        flex-wrap: wrap;
        height: inherit;
        width: inherit;
        justify-content: flex-start;
        padding: 15px;
}

.tables-window-item{
        border :1px solid #999999;
        padding: 10px;
        background-color: #DBDBDB;
}

.input-wrapper{
        display: flex;
        align-items: center;
        justify-content: flex-start;
        height: 8%;
        width: 100%;
}

.query-input{
        height: 90%;
        width: 50%;
}
```

```
.table-button{
	display: flex;
	text-align: center;
	margin-left: 15px;
	background-color: #BBBBBB;
	border:1px solid #777777;
	padding: 4px;
	cursor: pointer;
	transition: 0.2s;
}

.table-button:hover{
	background-color: #777777;
}

.output-wrapper{
	height: 90%;
	width: 100%;
	overflow: auto;
}

.printed-table{
	display: flex;
	justify-content: center;
}

.printed-table span{
	transition: 80ms;
	padding: 10px;
	max-height: 40px;
	cursor:default;
}
.printed-table span:hover{
	color: #85D5B2;
}
```