МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные системы управления базами данных»

Тема: Система экологического мониторинга

Студент гр. 9304 _____ Силкин В.А.

Студент гр. 9304 _____ Боблаков Д.С.

Студент гр. 9304 _____ Атаманов С.Д.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2023

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ (КУРСОВОЙ ПРОЕКТ)

Студенты

Силкин В.А.

Боблаков Д.С.

Атаманов С.Д.

Группа 9304

Тема работы: Система экологического мониторинга

Исходные данные:

Необходимо реализовать приложение по мониторингу экологических катастроф.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Заключение»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 05.09.2022

Дата сдачи реферата: 17.01.2023

Дата защиты реферата: 17.01.2023

Студент гр. 9304 _____ Силкин В.А.

Студент гр. 9304 _____ Боблаков Д.С.

Студент гр. 9304 _____ Атаманов С.Д.

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

Была разработана система экологического мониторинга. При разработке использовались neo4j и react для создания веб-страницы. Создано приложение, которое позволяет просматривать информацию об экологических катастрофах в разных местах, а также включает в себя фильтрацию данных по полю name и их изменение. Также реализована выгрузка и загрузка данных из приложения. Запуск приложения реализован при помощи docker-compose build --no-cache.

SUMMARY

An environmental monitoring system has been developed. The development used neo4j and react to create the web page. An application has been created that allows you to view information about environmental disasters in different places, and also includes filtering data by the name field and changing it. Uploading and loading data from the application is also implemented. Application launch is implemented using docker-compose build --no-cache.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы — создать веб приложение для экологического мониторинга.

Задачи:

1. Сформулировать основные сценарии использования и составить макет
2. Разработать модель данных
3. Разработать прототип «Хранение и представление»
4. Разработать прототип «Анализ»
5. Подготовить последнюю версию приложения «App is ready»

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Текущие требования к решению:

1. Релизовать страницу логина и добавить хотя бы по одному пользователю на каждую из ролей

2. Просмотр базы данных с помощью таблицы

3. Добавление новых маркеров экологических катастроф и изменение информации о существующих

4. Фильтрация данных

5. Приложение разворачивается через docker-compose build --no-cache на ubuntu

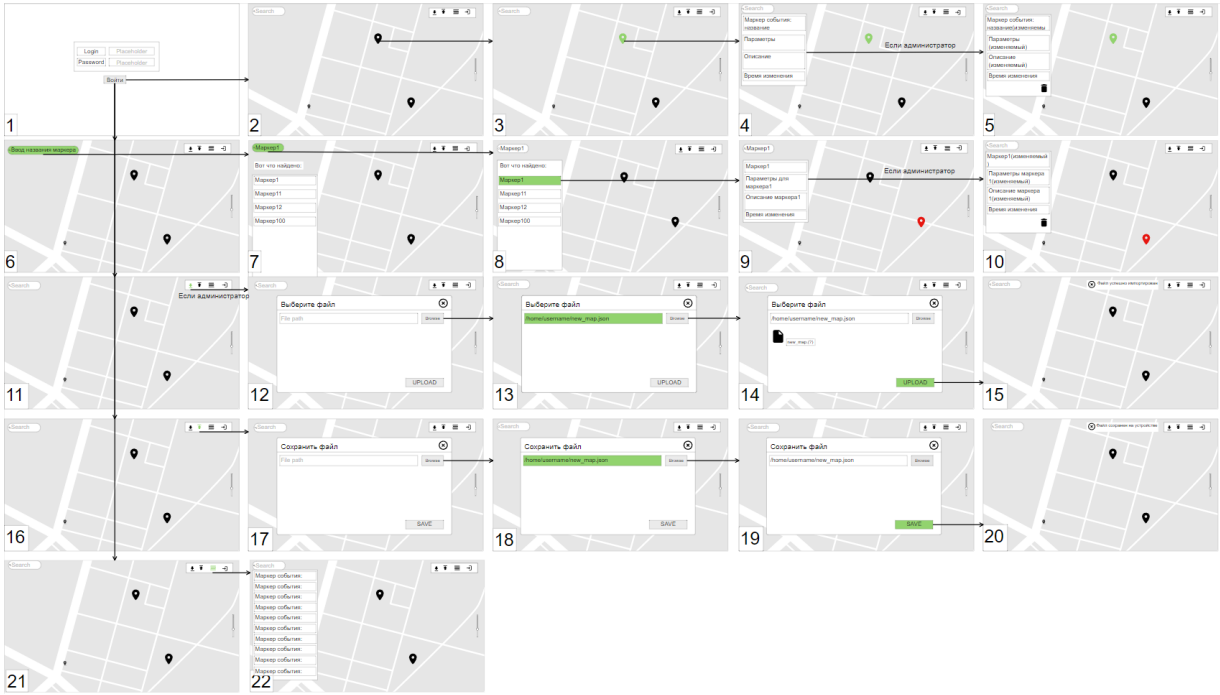
6. Импорт и экспорт данных в формате json

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет пользовательского интерфейса

Условные обозначения для макета

- Зелёный цвет - выделено пользователем
- Красный цвет - выделено программой
- "Если администратор" - функция, доступная только пользователям со статусом "администратор"



2.2. Сценарии использования

1. Авторизация

- При вводе данных - логина и пароля, пользователю присваевается один из двух статусов: "пользователь" или "администратор". Логин не может быть одинаковым для любых двух аккаунтов, даже если они имеют разные статусы.

2. Выделение маркера вручную

- Пользователь авторизируется.
- Пользователь нажимает на какой-либо маркер мышкой.
- Если пользователь имеет статус "пользователь": Всплывает окно с данными о маркере - название, описание, параметры маркера - координаты, время последнего изменения, все данные нередактируемые.
- Если пользователь имеет статус "администратор": Всплывает окно с данными о маркере - название (редактируемое), описание (редактируемое), параметры маркера (редактируемое), время последнего изменения, а также появляется кнопка удаления маркера.

3. Поиск маркера по названию

- Пользователь авторизуется.
- Пользователь вводит в поле поиска название маркера.
- Появляется окно, которое содержит в себе все маркеры, где введённое название является подстрокой названия самих маркеров (К примеру, по поиску "Маркер1" будет найден не только "Маркер1", но и "Маркер101"), его можно пролистывать колёсиком мыши.
- Пользователь нажимает на один из найденных маркеров.
- Программа выделяет маркер на карте и выводит информацию о нём (как и в прошлом случае, информация редактируемая, если пользователь имеет статус "администратор").

4. Массовая загрузка маркеров на карту

- Массовая загрузка доступна только администраторам. Если "пользователь" нажмёт на кнопку, всплывёт окно с информацией о том, что функция доступна только администраторам.
- Администратор авторизуется.

- Администратор нажимает на кнопку массового импорта.
- Появляется окно с выбором файла.
- Пользователь выбирает локальный файл формата json.
- Если файл имеет верный формат данных, то при нажатии на кнопку upload, на карту добавляются те маркеры из файла, которых ещё нет на карте.
- Если файл имеет неверный формат данных, администратору выдаётся ошибка обработки данных.
- Окно выбора файла можно закрыть, нажав на кнопку сверху справа окна.

5. Массовое сохранение маркеров

- Пользователь авторизуется.
- Пользователь нажимает на кнопку массового экспорта.
- Появляется окно с выбором места для сохранения файла.
- После указания пути сохранения, при нажатии на кнопку save, все маркеры с карты сохраняются в файл формата json.

6. Каталог маркеров

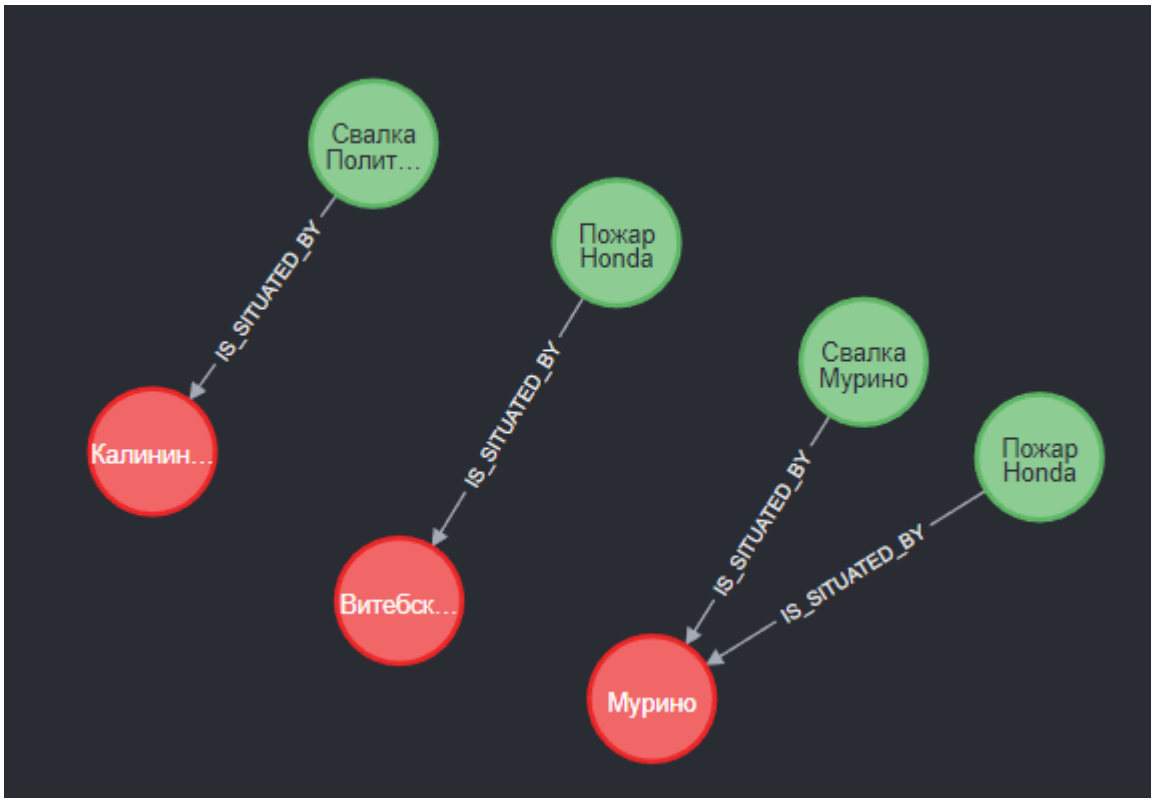
- Пользователь авторизуется
- Пользователь нажимает на кнопку каталога
- Появляется окно, схожее с окном поиска, оно содержит в себе все маркеры, при нажатии на любой маркер, будет выведена информация о нём как в случае с выделением маркера вручную.

2.3. Преобладающие операции.

Преобладающая операция – чтение, экологические катастрофы появляются относительно нечасто, но информация о них нужна большому числу людей

3. МОДЕЛЬ ДАННЫХ

3.1. Нереляционная модель данных



Сущности модели

MARKER - описание маркеров:

- id - Поле идентификации маркера внутри СУБД - 1 байт
- name - Название маркера - str(20) 20*2 байт
- desc - Краткое описание маркера - str(50) - 50*2 байт
- lat - Географическая ширина расположения маркера - double(10,8) - 8 байт
- lon - Географическая долгота расположения маркера - double(10,8) - 8 байт
- time - Время изменения маркера - long - 8 байт
- danger_level - Уровень опасности для окружающей среды по 5 бальной шкале - 1 байт

- pol_type - Тип загрязнения - str(10) - 10*2 байт
- area - Площадь загрязнения - int - 4 байта Итого: 190 байт

DISTRICT - район, в котором происходит экологическая катастрофа

- id - Поле идентификации района внутри СУБД - 1 байт
- name - Название района - str(20) - 20*2 байт Итого: 41 байт

Расчёт избыточности модели

Если количество маркеров принять за A, количество районов за B, общий объём данных равен 190*A + 41*B. На один район в среднем приходится 10 экологических катастроф, поэтому зависимость будет B=10A, тогда избыточность модели равна 600*A/589*A = 1.018 по памяти, поскольку id - избыточное поле в обоих сущностях. Прирост памяти линейный в зависимости от количества элементов.

Запросы к модели

Поиск маркера по частичному имени:

- MATCH (marker:MARKER) WHERE marker.name =~ '.*NAME.*' RETURN marker;

Добавить маркер:

- CREATE (marker:MARKER {name:NAME, desc:DESC, lat:LAT, lon:LON});

Редактировать маркер:

- MATCH (marker:MARKER) WHERE marker.name = NAME SET marker.name = NAME_1, marker.desc = DESC, marker.lat = LAT, marker.lon = LON;

Убрать маркер:

- MATCH (marker:MARKER) WHERE marker.name = NAME DETACH DELETE marker;

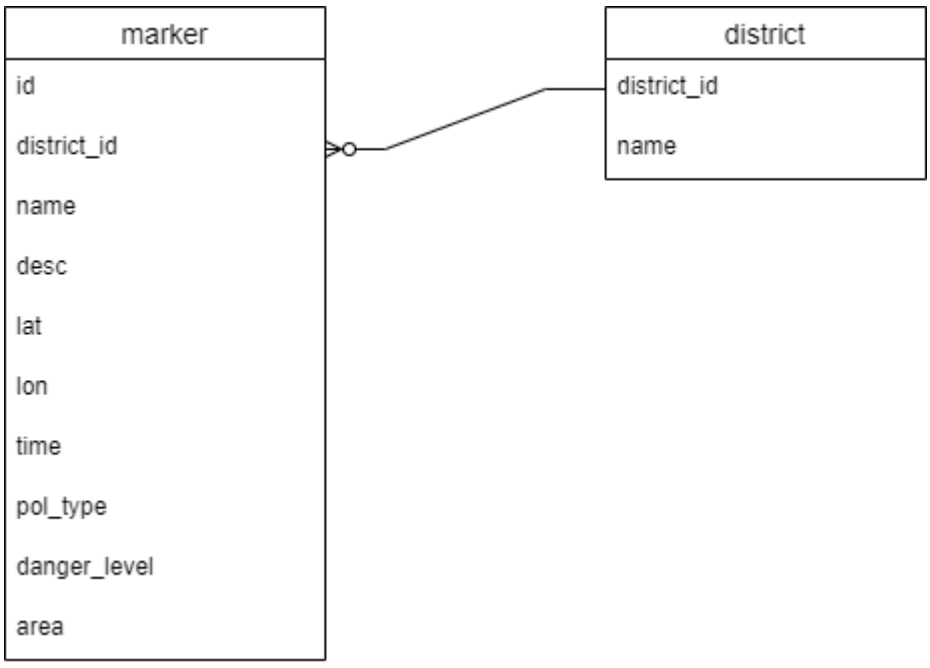
Вывести все маркеры:

- MATCH (marker:MARKER) RETURN marker;

Количество запросов

При выполнении сценариев использования, количество запросов в случае массового импорта и экспорта линейно зависит от числа элементов, в остальных случаях не зависит от числа элементов.

3.2. Аналог модели данных для SQL СУБД



MARKER - описание маркеров:

- id - Поле идентификации маркера внутри СУБД - INT(7) - 4 байта
- district_id - id района, в котором находится экологическая катастрофа - INT(7) - 4 байта
- name - Название маркера - CHAR(20) 20*2 байт

- desc - Краткое описание маркера - CHAR(50) - 50*2 байт
- lat - Географическая ширина расположения маркера - DOUBLE(10,8) - 8 байт
- lon - Географическая долгота расположения маркера - DOUBLE(10,8) - 8 байт
- time - Время изменения маркера - BIGINT(10) - 8 байт
- pol_type - Тип загрязнения - CHAR(10) - 10*2 байт
- danger_level - Уровень опасности для окружающей среды по 5 бальной шкале - TINYINT(3) - 1 байт
- area - Площадь загрязнения - INT(10) - 4 байта Итого: 197 байт

  DISTRICT - район, в котором происходит экологическая катастрофа

- district_id - Поле идентификации района внутри СУБД - INT(7) - 4 байта
- name - Название района - CHAR(20) - 20*2 байт Итого: 44 байта

Расчёт избыточности модели

Если количество маркеров принять за A, количество районов за B, общий объём данных равен 197*A + 44*B. На один район в среднем приходится 10 экологических катастроф, поэтому зависимость будет B=10A, тогда избыточность модели равна 637*A/589*A = 1.081 по памяти, поскольку id - избыточное поле в обоих сущностях. Прирост памяти линейный в зависимости от количества элементов.

Запросы к модели

Поиск маркера по частичному имени:

- SELECT * FROM markers WHERE name LIKE '%NAME%';

Добавить маркер:

- INSERT INTO markers VALUES (NAME, DESC, LAT, LON, TIME);

Редактировать маркер:

- UPDATE markers SET name=NAME, desc=DESC, lat=LAT, lon=LON, time=TIME WHERE id=ID;

Убрать маркер:

- DELETE FROM markers WHERE id=ID;

Вывести все маркеры:

- SELECT * FROM markers;

Количество запросов

При выполнении сценариев использования, количество запросов в случае массового импорта и экспорта линейно зависит от числа элементов, в остальных случаях не зависит от числа элементов.

3.3. Вывод

Сравнивая между собой реляционную и нереляционную СУБД, вторая незначительно выигрывает по избыточности занимаемой памяти, число запросов в обоих случаях одинаковое.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

При входе пользователь получает определенные права в зависимости от того является ли он администратором.

Для обычных пользователей доступен лишь просмотр маркеров и выгрузка всех маркеров.

Для администратора доступно то же, что для пользователя, но помимо этого он может редактировать маркеры экологических катастроф.

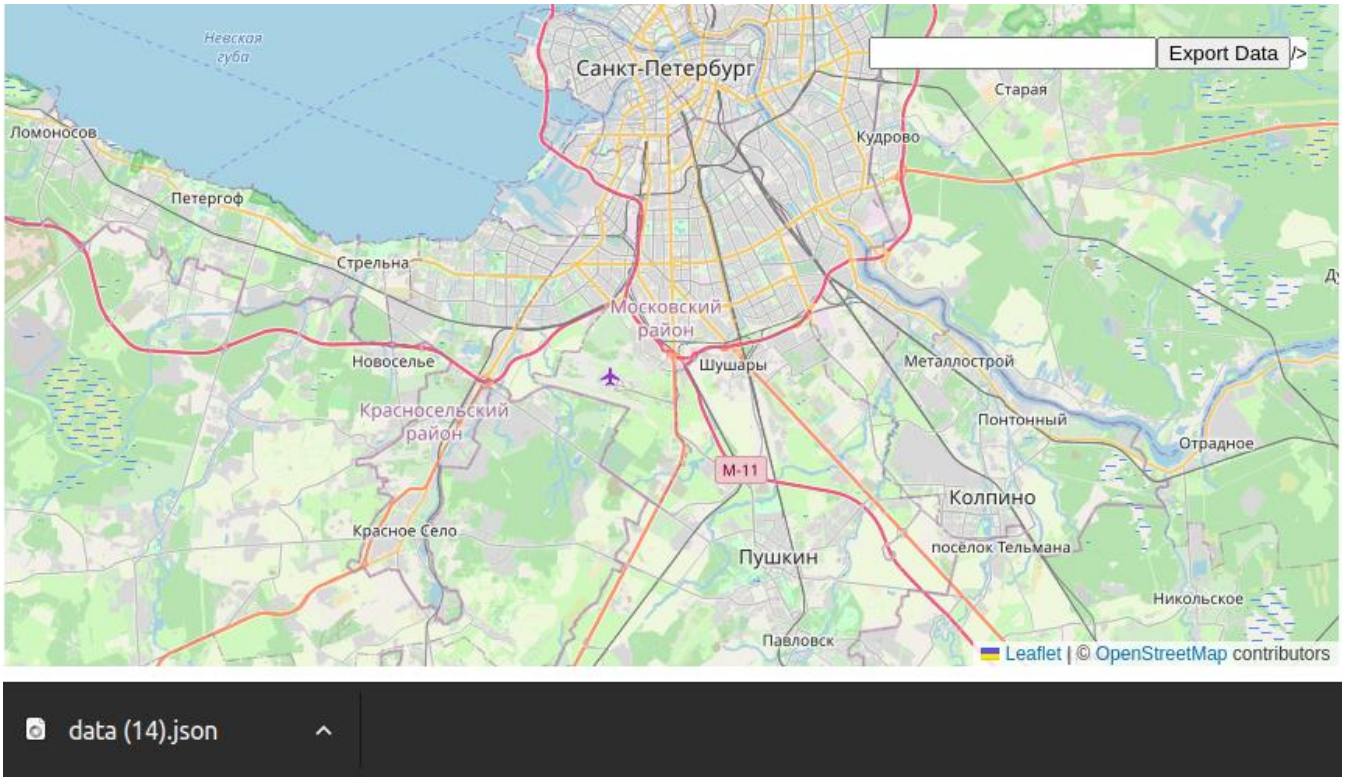
4.2. Схема экранов приложения

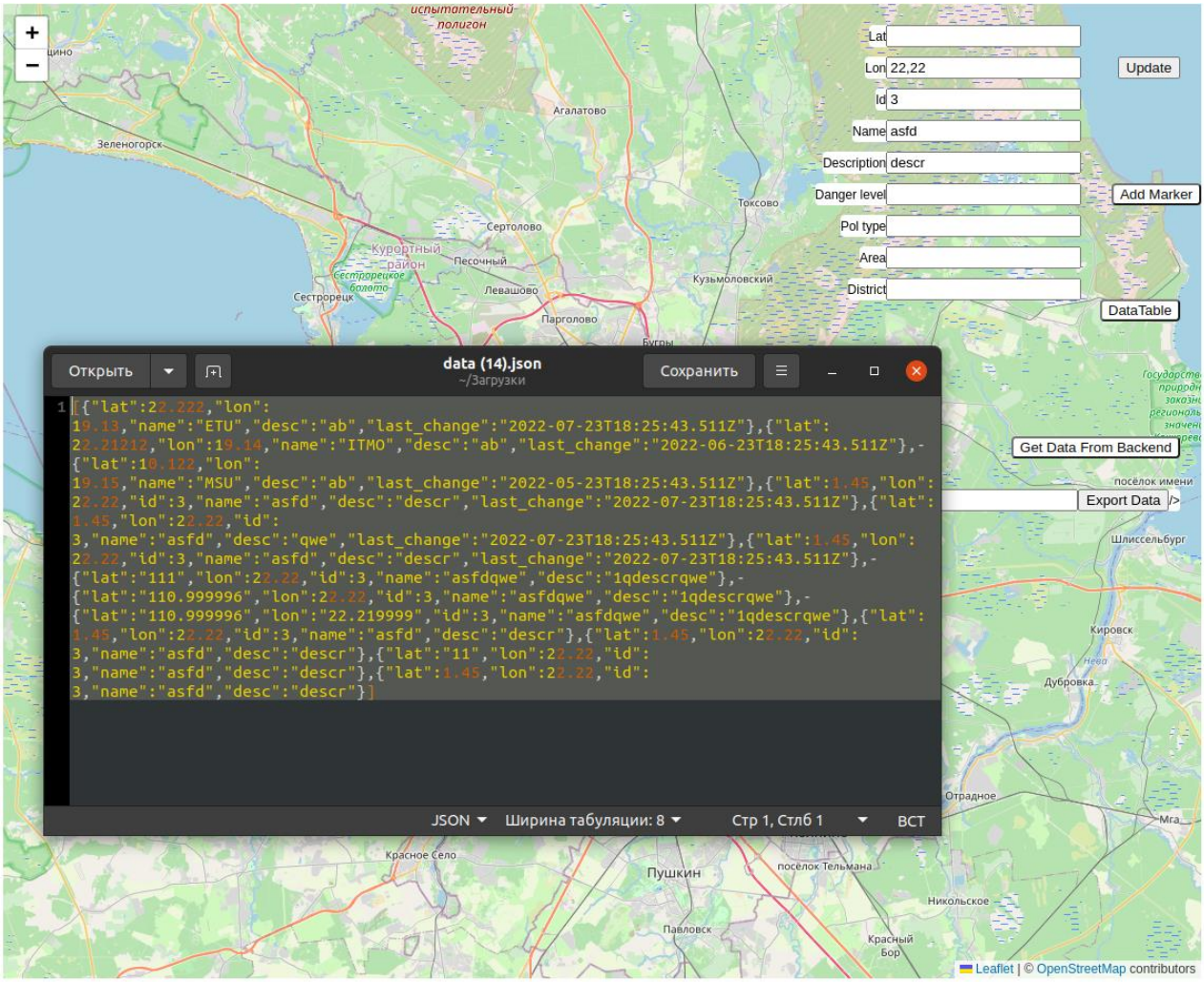
Вид карты от лица администратора

Search by Name:

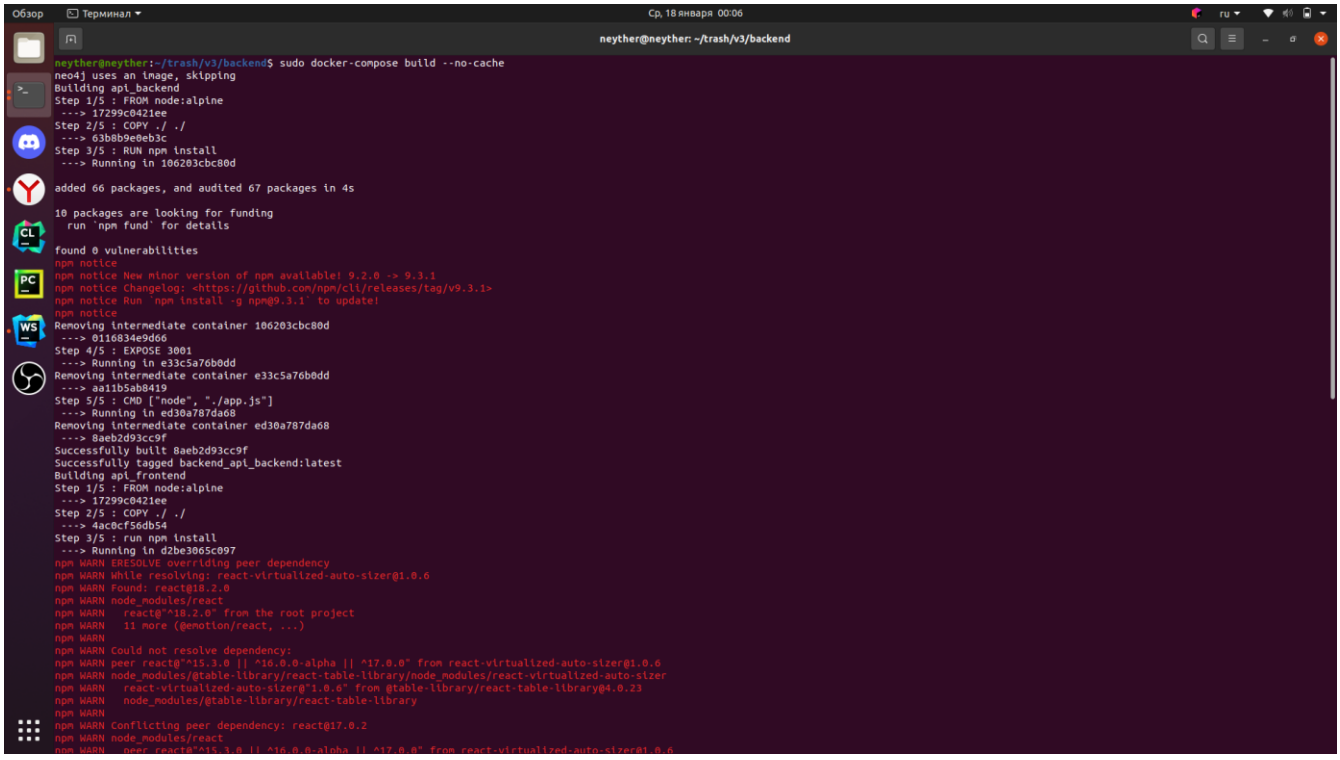
Name	Latitude	Longitude	Description	Last Time Change
ETU	22.222	19.13	ab	2022-07-23T18:25:4…
ITMO	22.21212	19.14	ab	2022-06-23T18:25:4…
MSU	10.122	19.15	ab	2022-05-23T18:25:4…
asfd	1.45	22.22	descr	2022-07-23T18:25:4…
asfd	1.45	22.22	qwe	2022-07-23T18:25:4…
asfd	1.45	22.22	descr	2022-07-23T18:25:4…
asfdqwe	111	22.22	1qdescrqwe	
asfdqwe	110.999996	22.22	1qdescrqwe	
asfdqwe	110.999996	22.219999	1qdescrqwe	
asfd	1.45	22.22	descr	

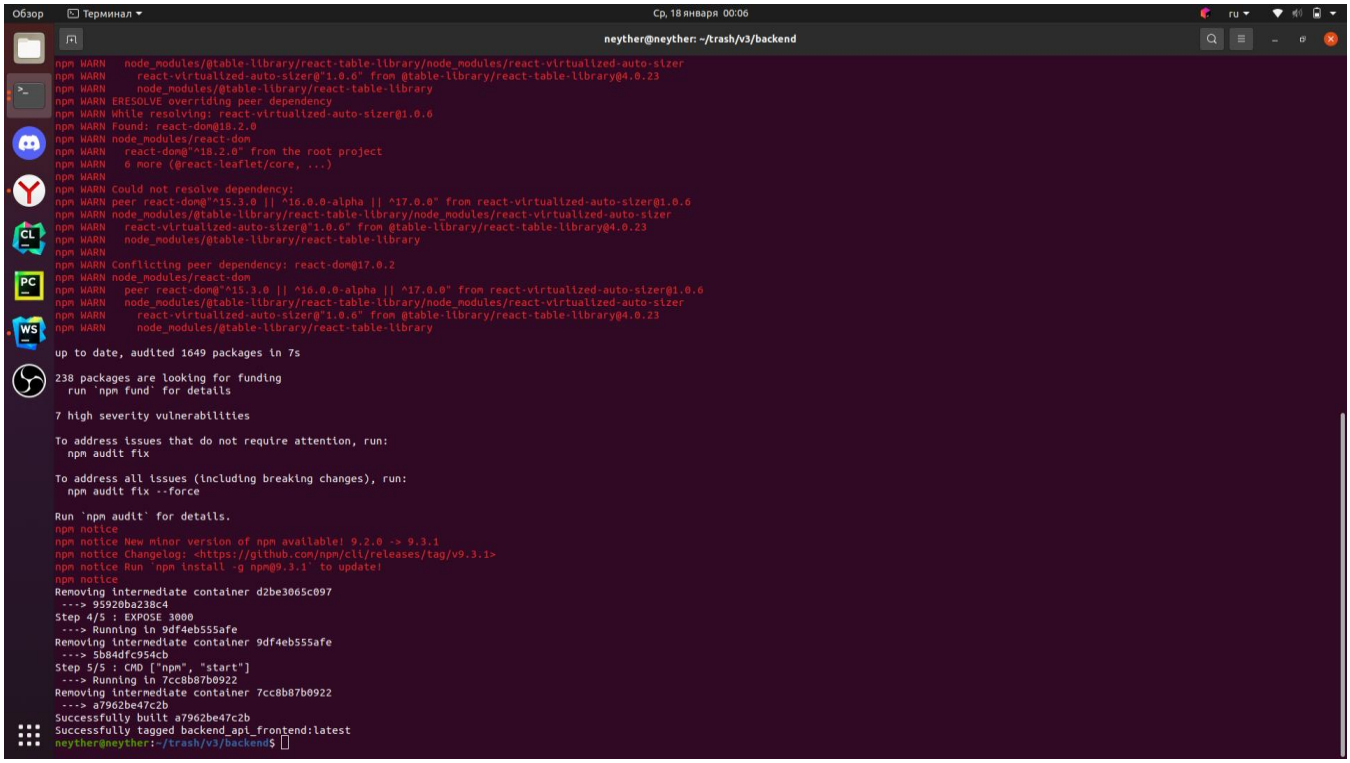
Таблица с перечислением всех активных маркеров, а также поиск по имени



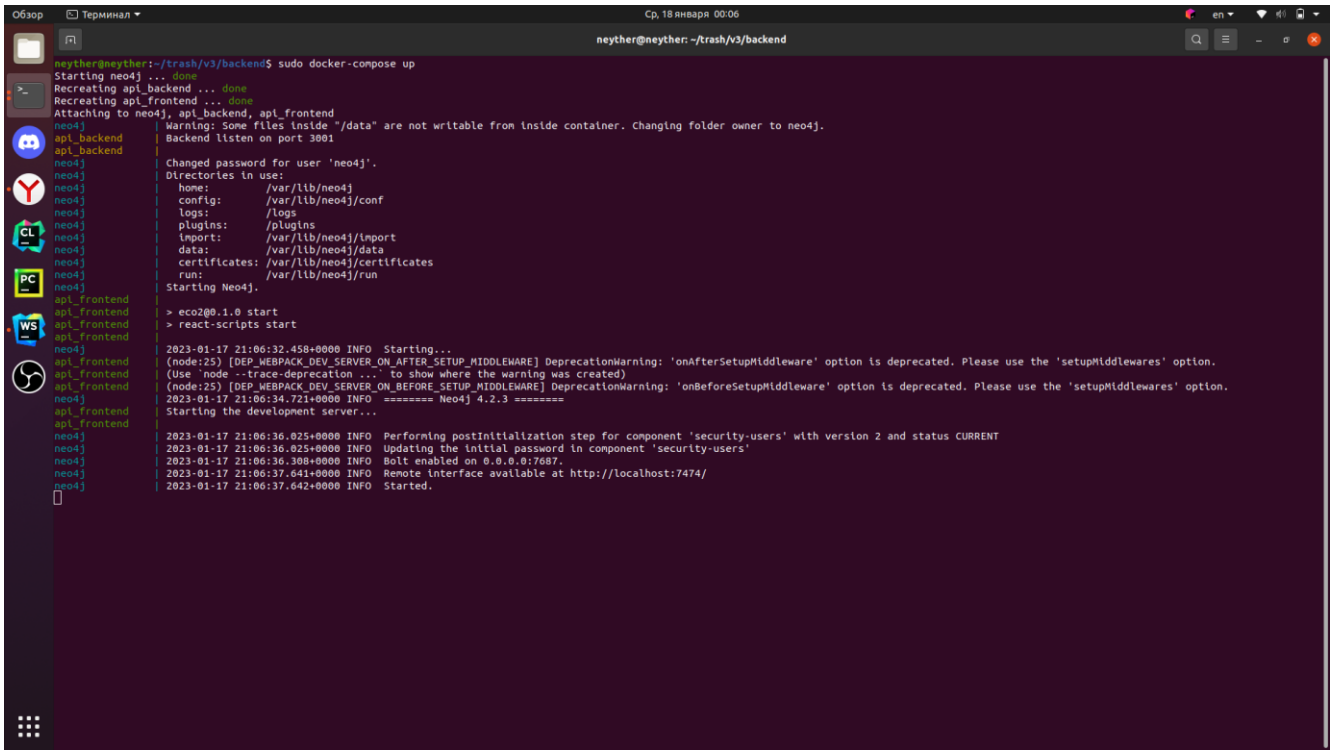


Экспорт данных





Сборка при помощи docker



Запуск при помощи docker

4.3. Использованные технологии

Для хранения данных была использована neo4j, для создания веб-приложения использовался react.

ВЫВОДЫ

Достигнутые результаты:

В ходе работы было разработано веб-приложение для экологического мониторинга при помощи react. Для хранения данных и управления ими использовалась СУБД neo4j.

Будущее развитие решения:

Реализация зависимости экологических катастроф друг от друга, добавление функционала (группировка по различным полям).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация neo4j: https://neo4j.com/
2. Документация react: https://reactjs.org/