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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Нереляционные базы данных»
Тема: Каталог ДТП в США

Студентка гр. 7382 _____ Дерябина П.С.

Студентка гр. 7382 _____ Головина Е.С.

Студент гр. 7382 _____ Чигалейчик А.С.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2020

ЗАДАНИЕ

НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студенты

Дерябина П.С.

Головина Е.С.

Чигалейчик А.С.

группа 7382

Тема работы: каталог ДТП в США

Исходные данные:

Создание приложения с возможностью импорта всего датасета, отображения статистики, пространственной аналитики

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 25 страниц.

Дата выдачи задания: 18.09.2020

Дата сдачи реферата:

Дата защиты реферата:

Студент гр. 7382	_____	Дерябина П.С.
Студент гр. 7382	_____	Головина Е.С.
Студент гр. 7382	_____	Чигалейчик А.С.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения с возможностью в удобной форме просматривать содержимое каталога ДТП в США

Найти исходный код и дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h20-accidents-usa

SUMMARY

Within the framework of this course, it was supposed to develop an application in a team on one of the set topics. The topic of creating an application was chosen with the ability to view the contents of the accident catalog in the USA in a convenient form

You can find the source code and additional information at the link: https://github.com/moevm/nosql2h20-accidents-usa

СОДЕРЖАНИЕ

I. ВВЕДЕНИЕ

Целью работы является создание приложения, в функциональность которого входят добавление новых аварий, оценка различных параметров аварий, фильтрация, сопоставление данных. Выбранный нами стек технологий включает в себя Node.js, Pug[1], CSS, Express.js[2], Mongo[3].

II. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение со следующими возможностями: страница со списком каталогов ДТП, содержащая записи всего датасета, статистика ДТП, фильтрация ДТП, добавление ДТП, импорт и экспорт данных БД. В качестве системы управления базами данных использовать MongoDB [3].

III. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1 Макеты UI

1. Просмотр списка ДТП с возможностью фильтрации.

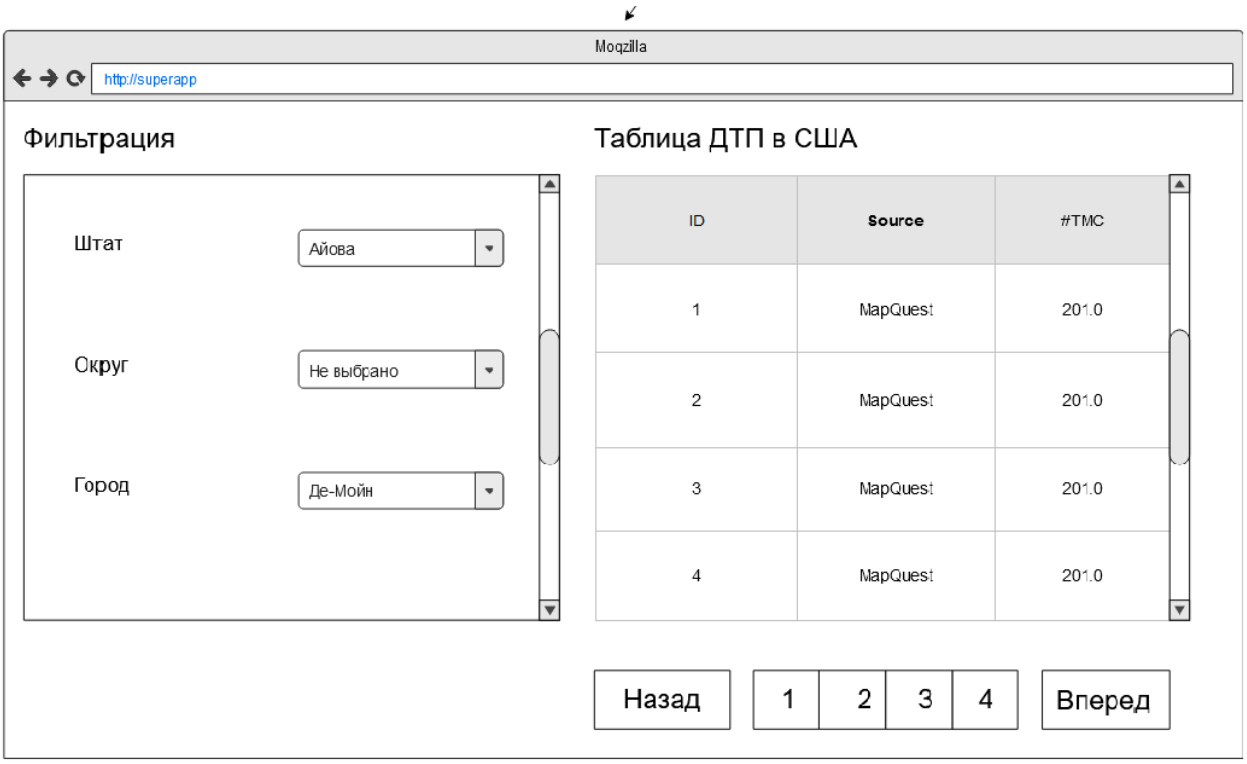


Рисунок 3.1. — Просмотр списка ДТП.

2. Добавление данных пользователя.

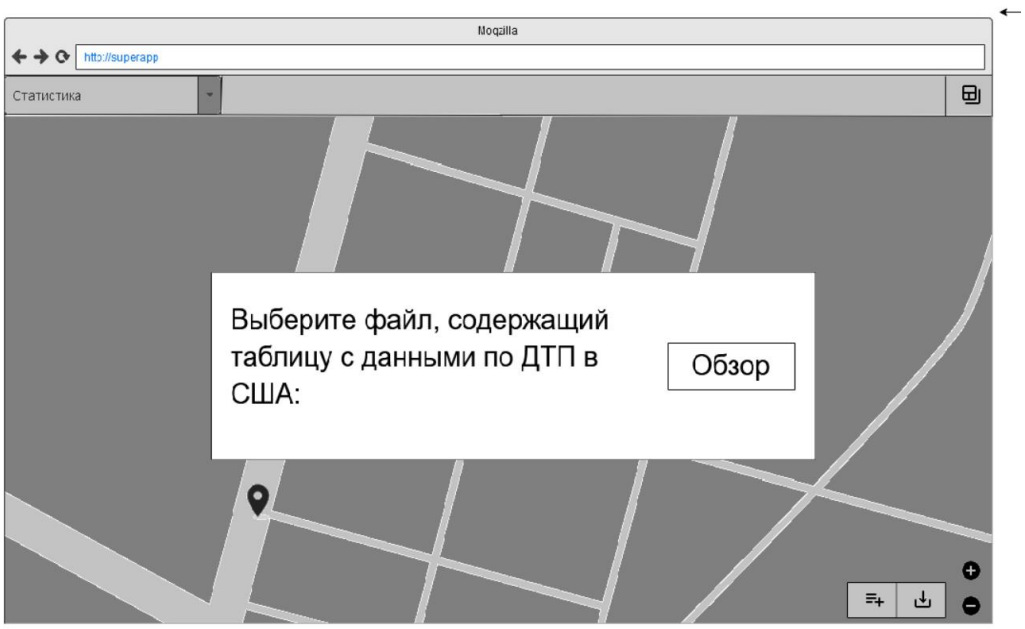


Рисунок 3.2. — Добавление данных пользователя.

3. Просмотр ДТП.

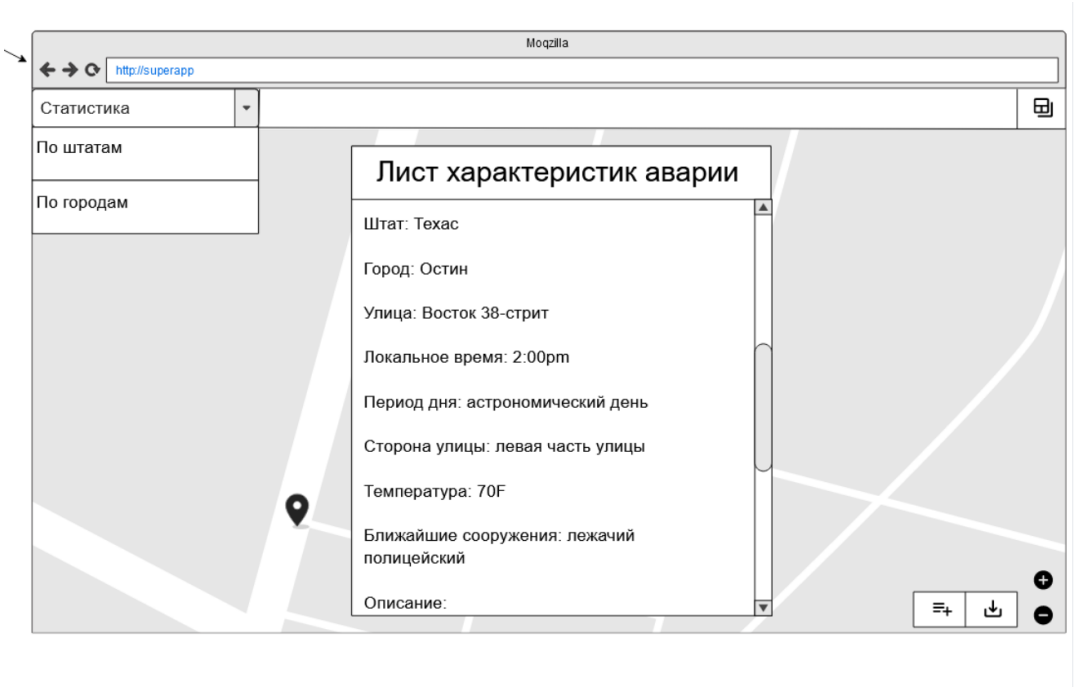


Рисунок 3.3. — Просмотр ДТП.

4. Просмотр статистки ДТП.

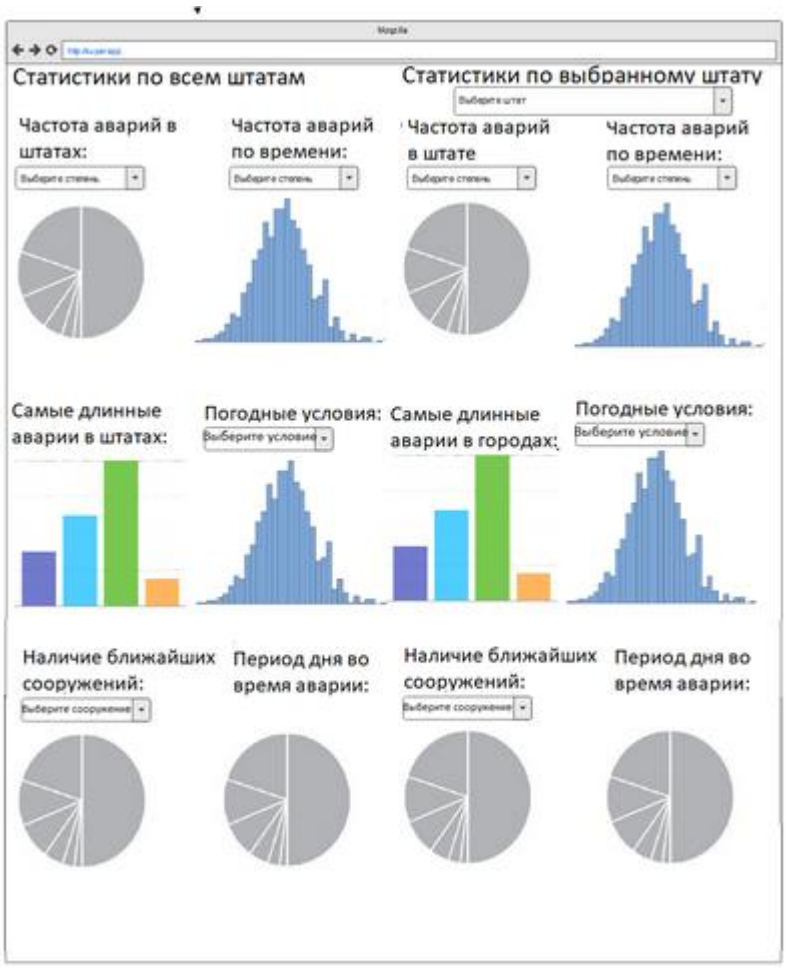


Рисунок 3.4 — Просмотр статистки ДТП по штатам.

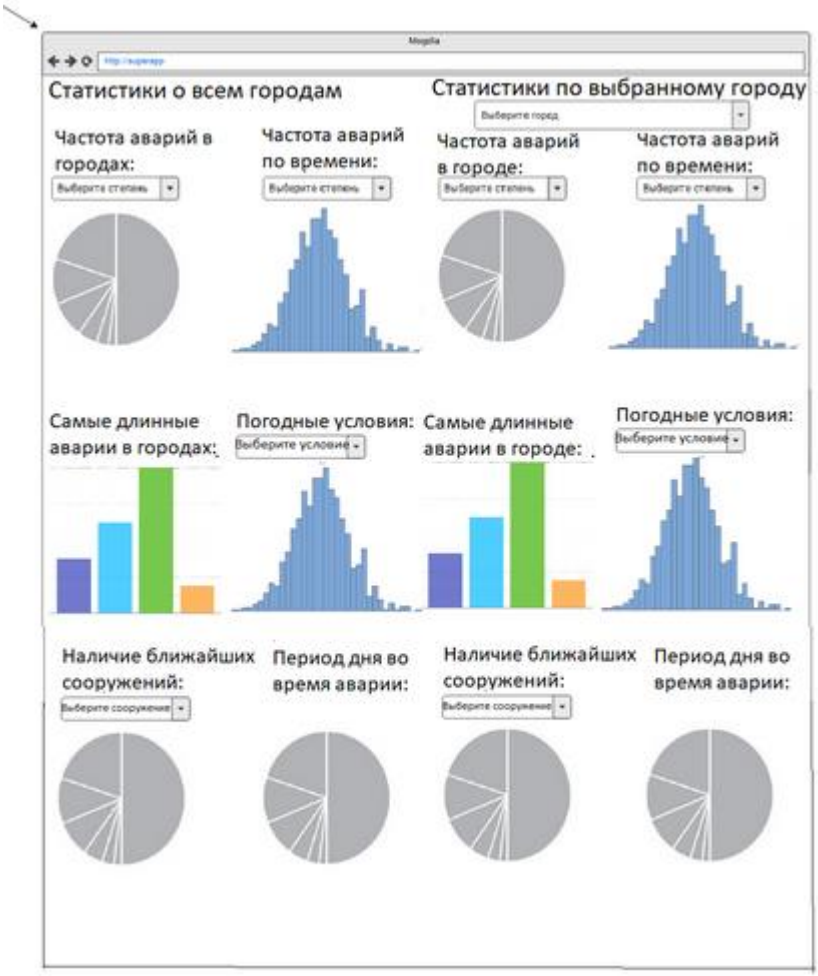


Рисунок 3.5 – Просмотр статистки ДТП по городам.

5. Экспорт БД.

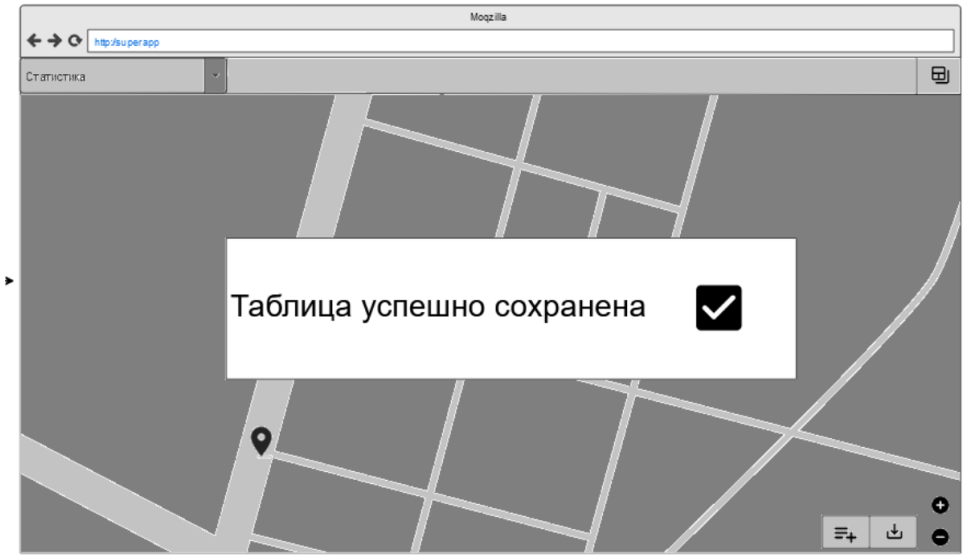


Рисунок 3.6 — Экспорт БД.

6. Просмотр ДТП на карте.

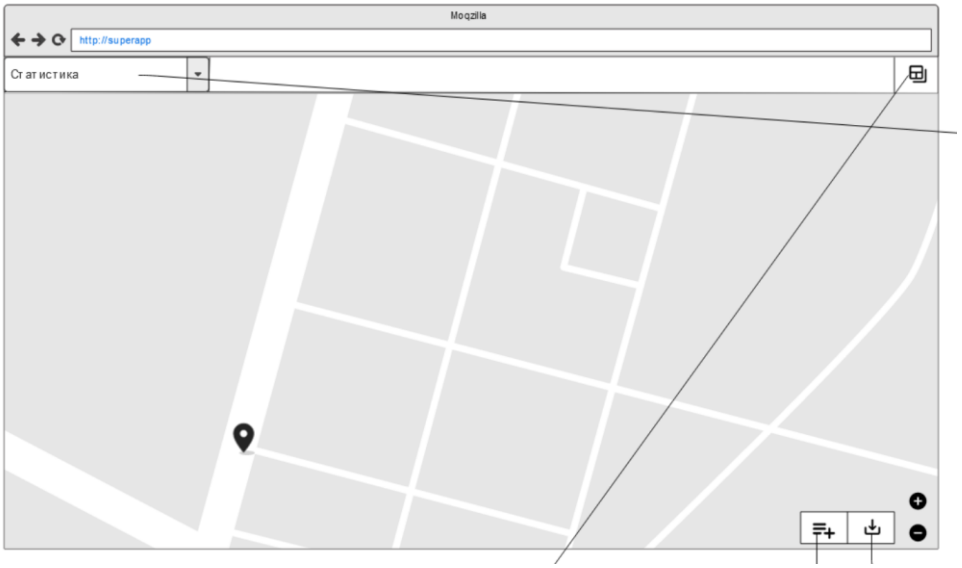


Рисунок 3.7 – Просмотр ДТР на карте.

3.2 Сценарии использования.

Сценарий использования просмотра аварий на карте

1. На карте пользователь нажимает на интересующую его аварию, отмеченную значком.
2. Открывается окно с краткой информацией об аварии, пользователь может это прочитать.
3. В открывшемся окне можно нажать на значок скачивания всей информации о выбранной аварии.
4. В открывшемся окне можно нажать на значок просмотра всей информации о выбранной аварии, после чего всплывет окно со всей информацией, которую можно будет просмотреть.

Альтернативный вариант:

После нажатия на значок скачивания полной информации, может произойти сбой соединения.

Сценарий использования «Просмотр статистики»

1. На главном окне пользователь может нажать на выпадающий список и выбрать просмотр статистики по штатам или городам.
2. При просмотре по штатам открывается окно, в левой части которого расположены 6 разных статистик с информацией по всем штатам, для некоторых статистик можно задать фильтрацию по соответствующим значениям. В правой части можно во выпадающему списку выбрать конкретный штат и такие же 6 характеристик появятся для всех городов выбранного штата.
3. При просмотре по городам открывается окно, в левой части которого расположены 6 разных статистик с информацией по всем городам, для некоторых статистик можно задать фильтрацию по соответствующим значениям. В правой части можно во выпадающему списку выбрать конкретный город и такие же 6 характеристик появятся для всех улиц выбранного города.

Сценарий использования «Просмотр таблицы»

1. В верхнем правом углу окна пользователь может нажать на значок таблицы, после чего откроется окно, в правой части которого находится ограниченное число колонок таблицы.
2. Внизу таблицы, нажимая на кнопки, соответствующие страницам таблицы, можно переходить на следующие колонки.
3. В левой части окна последовательно расположены все поля таблицы, по которым пользователь может настроить фильтрацию: при выборе конкретного значения для поля будет происходить фильтрация таблицы.

Сценарий использования «Экспорт»

1. В правом нижем углу находится значок скачивания, при нажатии на который пользователь сможет скачать всю таблицу на свой девайс.
2. После удачного скачивания появится окно об успешном завершении операции.

Альтернативный вариант использования:

1. При скачивании таблицы у пользователя может закончиться место на жестком диске.
2. Может произойти сбой соединения.

Сценарий использования «Импорт»

1. В правом нижнем углу также есть значок добавления таблицы, при нажатии на который пользователю открывается окно, которое позволяет выбрать файл со своей таблицей по дтп.
2. После загрузки файла будет отображаться главное окно с новыми данными.

Альтернативный вариант:

1. Пользователь может осуществить загрузку файла с некорректным форматом, что приведет к всплытию окна об ошибке.
2. При загрузке файла может произойти сбой соединения.

IV. МОДЕЛЬ ДАННЫХ

4.1. Схема нереляционной базы данных.

```
{
    "_id": "ObjectId",
    "severity": "Int",
    "time": {
        "start": "Timestamp",
        "end": "Timestamp"
    },
    "start": {
        "lat": "Double",
        "lng": "Double"
    },
    "end" : {
        "lat": "Double",
        "lng": "Double"
    },
    "distance": "Double",
    "description": "String",
    "country": "String",
    "state": "String",
    "county": "String",
    "city": "String",
    "address": {
        "number": "Int",
        "street": "String",
        "side": "String",
    },
    "weather": {
        "temperature": "Double",
        "wind_chill": "Double",
        "humidity": "Int",
        "pressure": "Double",
        "visibility": "Double",
        "wind_direct": "String",
        "wind_speed": "Double",
        "precipation": "Double",
        "condition": "String"
    },
    "near_objects": {
```

```
            "amenity": "Boolean",
            "bump": "Boolean",
            "crossing": "Boolean",
            "give_way": "Boolean",
            "junction": "Boolean",
            "no_exit": "Boolean",
            "railway": "Boolean",
            "roundabout": "Boolean",
            "station": "Boolean",
            "stop": "Boolean",
            "traffic_calming": "Boolean",
            "traffic_signal": "Boolean",
            "turning_loop": "Boolean"
        },
        "day_period": "String"
}
```

4.2. Список сущностей модели

Так как исходные данные представляют собой одну общую таблицу с описанием аварий - в базе данных будет создана одна коллекция - accidents. Каждая авария - это документ коллекции (описывает сущность Accident). При наличии всех данных один документ будет иметь структуру как в п.1.

4.3. Описание назначений коллекций, типов данных и сущностей.

Accident - сущность аварии, содержащая следующие атрибуты:

* _id - уникальный идентификатор аварии (12B)
* severity - целое число от 1 до 4, серьезность аварии (4B)
* time - объект времени аварии (8+8=16B)
  * start - время начала аварии
  * end - время конца аварии
* start - место начала аварии (8+8=16B)
  * lat - широта в координатах GPS
  * lng - долгота в координатах GPS
* end - место завершения аварии (8+8=16B)

* lat - широта в координатах GPS

* lng - долгота в координатах GPS

* distance - протяженность дороги, пострадавшей в рез-те ДТП (8B)

* description - описание аварии (4*200=800B)

* country - страна, где произошла авария (4*2=8B)

* state - штат (4*2=8B)

* county - округ (15*4=60B)

* city - город (20*4=80B)

* address - адрес ближайшего к аварии дома (4+40*4+4=168B)

* number - номер дома

* street - улица

* side - сторона улицы

* weather - погода при которой произошла авария (8+8+4+8+8+(4×5)+8+8+(4×25)=172B)

* temperature - температура воздуха (в фаренгейтах)

* wind_chill - температура ветра (в фаренгейтах)

* humidity - влажность (в процентах)

* pressure - давление воздуха (в дюймах)

* visibility - видимость дороги (в милях)

* wind_direct - направление ветра

* wind_speed - скорость ветра (миль в час)

* precipation - количество осадков (в дюймах), если есть

* condition - погодные условия (дождь/снег/туман/и т.д)

* near_objects - близлежащие объекты дорожной и городской инфраструктуры (2*13=26B)

* amenity - места обслуживания (рестораны/библиотеки/колледжи/и т.д.)

* bump - лежачий полицейский

* crossing - пешеходный переход

* give_way - знак "главная дорога"

* junction - перекресток

* no_exit - тупик

* railway - ж/д пути

* roundabout - круговая развязка

* station - остановка транспорта

* stop - знак "Движение без остановки запрещено"

* traffic_calming - любое устройство для снижения скорости движения

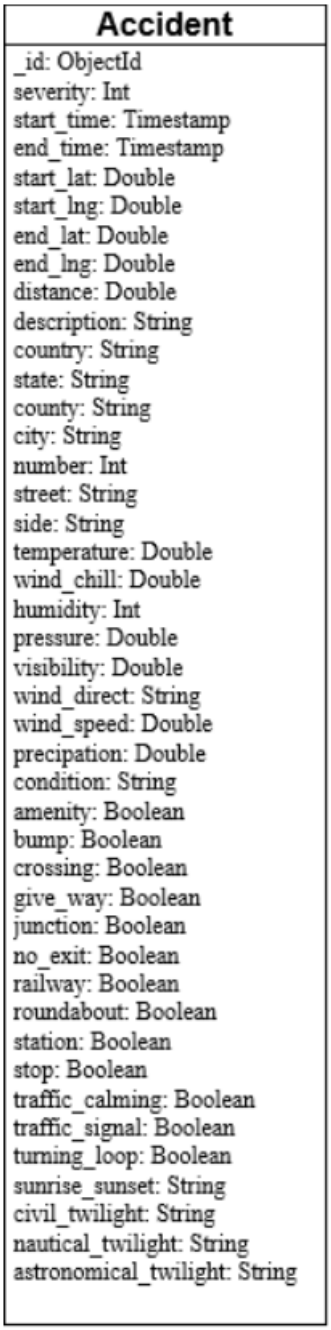
* traffic_signal - светофор

* turning_loop - расширенный участок шоссе с непроходимым островом для поворота

* day_period - время дня (день/ночь) (4B)

4.4. Аналог модели данных для SQL СУБД.

В SQL модели - также единственная сущность Accident, только без группировки по блокам. Поля такие же как для NoSQL модели.



4.5. Оценка удельного объема информации, хранимой в модели.

Известно, что в датасете 3.5 миллиона записей. Для того, чтобы высчитать полный объем памяти, необходимый для хранения данных, умножим максимальный объем одной записи 1398 байт (1.4 кБ) на количество записей в датасете: 1 398 * 3 500 000 = 4.56 гБ

Если учесть, что в NoSQL модели отсутствующие объекты не занимают памяти, то можно понять, что примерная прикидка была очень грубой и

подходит только для реляционной модели, где все поля существуют вне зависимости от их наполненности.

Обратимся к информации о данных, которая представлена на странице с датасетом:

* 70% данных о местоположении (end) в конце аварии отсутствуют (потому что авария не была продолжительной и закончилась там же, где и началась);

* 64% данных о номере дома (number) отсутствуют (потому что рядом с аварией не было домов);

* 53% данных о температуре ветра (wind_chill) отсутствуют;

* 58% данных о количестве осадков (percipation) отсутствуют;

* объектов дорожной и городской инфраструктуры (near_objects) как правило несколько (2).

Зная эту информацию подсчитаем объем памяти:

((1256×3500000)+0.3*(16×3500000)+0.36*(4×3500000)+0.47*(8×3500000)+ 0.42*(8*3500000))/1024^3 = 4.14 гБ

Получим универсальную формулу. Примем 3.5 млн записей за N, а количество памяти (memory) за M, тогда:

Для SQL-модели: M = 1398*N

Для NoSQL- модели: M = 1270*N

Видим, что при увеличении количества записей рост занимаемой памяти линейный в обоих случаях, но в NoSQL-модели одна запись занимает меньше места, поэтому использование этой модели более выгодное.

V. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

5.1. Схема экранов приложения

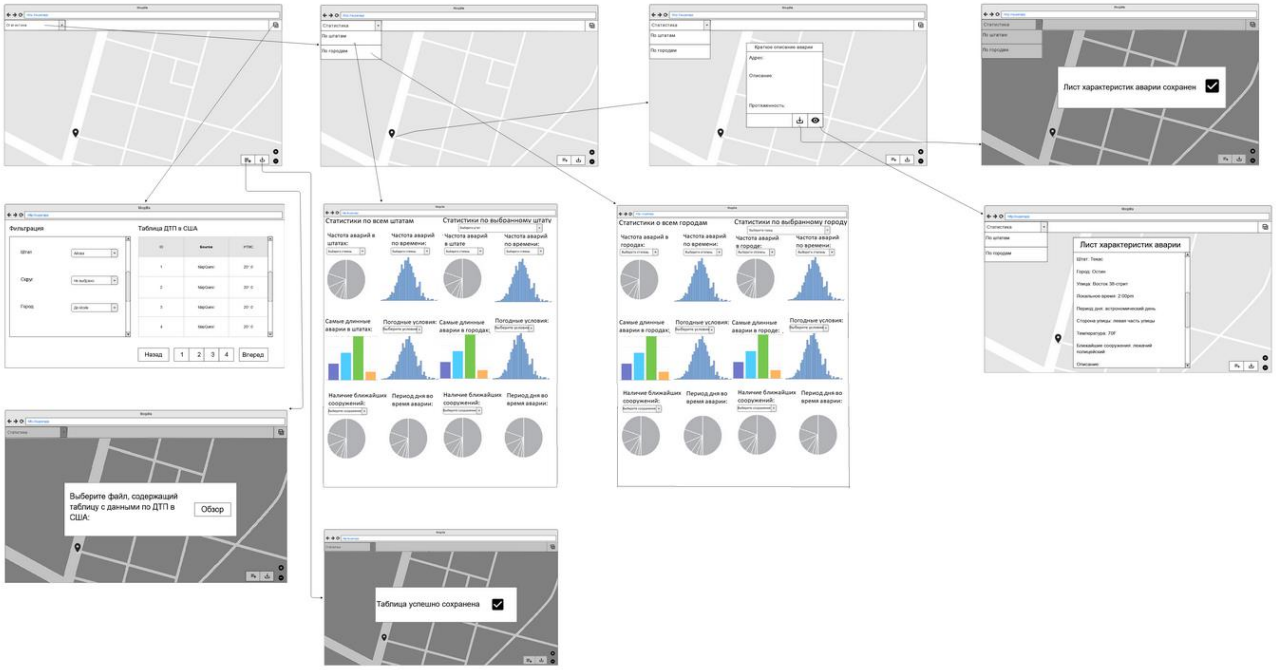


Рисунок 5.1 — Схема экранов приложения.

5.2. Использованные технологии.

5.3. Node.js, Pug, CSS, Express, Mongo.

5.4. Ссылки на приложение.

1. Github: https://github.com/moevm/nosql2h20-accidents-usa

VI. ВЫВОДЫ

6.1. Достигнутые результаты.

Было разработано приложение, в функциональность которого входит: страница со списком морских путешествий, содержащая записи за все время, статистика путешествий, фильтрация и сортировка путешествий, добавление путешествий, импорт и экспорт данных БД. В качестве системы управления базами данных используется MongoDB.

Было разработано приложение, в функциональность которого входят добавление новых аварий, фильтрация данных по полям, пространственное отображение точек ДТП на карте, экспорт и импорт данных БД. В качестве системы управления базами данных используется MongoDB.

6.2. Недостатки и пути для улучшения полученного решения.

К недостаткам текущей реализации можно отнести отсутствие пространственной аналитики, статистики по параметрам, медленный экспорт всех данных, невозможность добавления данных через интерфейс, перенаправление на новую страницу при добавлении новых (пользовательских) данных.

6.3. Будущее развитие решения.

Дальнейшее развитие приложения предполагает предоставления пространственной аналитики, статистики по параметрам, быстрый экспорт, возможность добавления данных через интерфейс приложения, исключение возможности перенаправления на новую страницу при добавлении новых (пользовательских) данных.

VII. ПРИЛОЖЕНИЯ

7.1. Документация по сборке и развертыванию приложения.

Инструкция для Docker.

1. Скачать репозиторий [4].
2. Внутри папки дирекории проекта открыть терминал.
3. Выполнить команду docker-compose up –build.

VIII. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Pug — https://pugjs.org/api/getting-started.html.
2. Документация Express.js — https://expressjs.com/ru/starter/installing.html
3. Документация MongoDB — https://docs.mongodb.com/. Github-репозиторий — https://github.com/moevm/nosql2h20-accidents-usa