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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Разработка программного обеспечения информационных систем»

ТЕМА: Анализ данных OpenSource репозиториев

Студенты гр. 5382 _____ Разбитнева А.М.
Лисс Н.И.
Черненко А.Б.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2019

ЗАДАНИЕ

Студенты: Разбитнева А.М., Лисс Н.И., Черненко А.Б.

Группа 5382.

Тема ИДЗ: Анализ данных OpenSource репозиториев

Исходные данные: команде необходимо реализовать веб-приложение, которое бы отображало статистику репозитория по некоторым критериям.

Предполагаемый объем:

Не менее 10 страниц (обязательны разделы «Содержание», «Введение», «Заключение», «Список использованных источников»).

Дата выдачи задания:

Дата сдачи реферата: 17.01.2019

Дата защиты реферата: 17.01.2019

Студенты гр. 5382 _____ Разбитнева А.М.
Лисс Н.И.
Черненко А.Б.

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

В данной работе представлены все этапы разработки ПО на тему «Анализ данных OpenSource репозиториев». Разработка осуществлялась на основе работы с нереляционной базой данных MongoDB. В качестве результата было получено пользовательское приложение, получающее данные о репозиториях GitHub с api этого сайта, обрабатывающее эти данные по заданным пользователем критериям и предоставляющее пользователю обработанную информацию в виде графиков столбчатых диаграмм. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2018-os_github_analysis

SUMMARY

This paper presents all the stages of software development on the topic “Data analysis of OpenSource repositories”. The development was carried out on the basis of work with the MongoDB non-interrelated database. As a result, a user application was obtained that received data on GitHub repositories from the api of this site, processes this data according to user-defined criteria and provides the user with the processed information in the form of graphs of bar charts. To find the source code and all the additional information use the link: https://github.com/moevm/nosql2018-os_github_analysis

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

MongoDB — это документо - ориентированная база данных. Т.е. каждая запись – это документ без жестко заданной схемы, который может содержать вложенные документы.

В результате выполнения ИДЗ разработано веб - приложение для анализа данных OpenSource репозиториев. Для обработки данных использовались средства MongoDB. Такого рода обработку удобно делать с помощью средств MongoDB, потому что существует функция db.collection.find(), в качестве аргументов которой можно указать поле документа и значение, по которому требуется отфильтровать документы базы данных.

ПРЕДЛАГАЕМОЕ РЕШЕНИЕ

В результате выполнения данной работы будет разработано пользовательское приложение для анализа данных OpenSource репозиториев с сайта github.com. Анализ может осуществляться только по обработанным данным. А для обработки данных можно использовать средства MongoDB.

Предлагается осуществить обработку данных по количественному значению определённых параметров, таких как: «Количество форков», «Количество звёзд», «Количество строк» и «Скорость написания кода».

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

В рамках курса «Нереляционные базы данных» было выдано задание создать ПО, разработанное на основе базы данных MongoDB. В таком ПО должна производиться обработка данных OpenSource репозиториев с сайта github.com. Параметры обработки данных должны задаваться пользователем, следовательно, должен быть создан пользовательский интерфейс. Далее результат обработки должен быть предоставлен пользователю для дальнейшего анализа. Результат должен быть представлен в виде графиков столбчатых диаграмм, а также должен сохраняться в файле формата JSON. Обработка данных должна выполняться с помощью средств MongoDB, а именно как выборка из исходной базы данных по определённым пользователем значениям определённых полей базы данных.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет веб-приложения

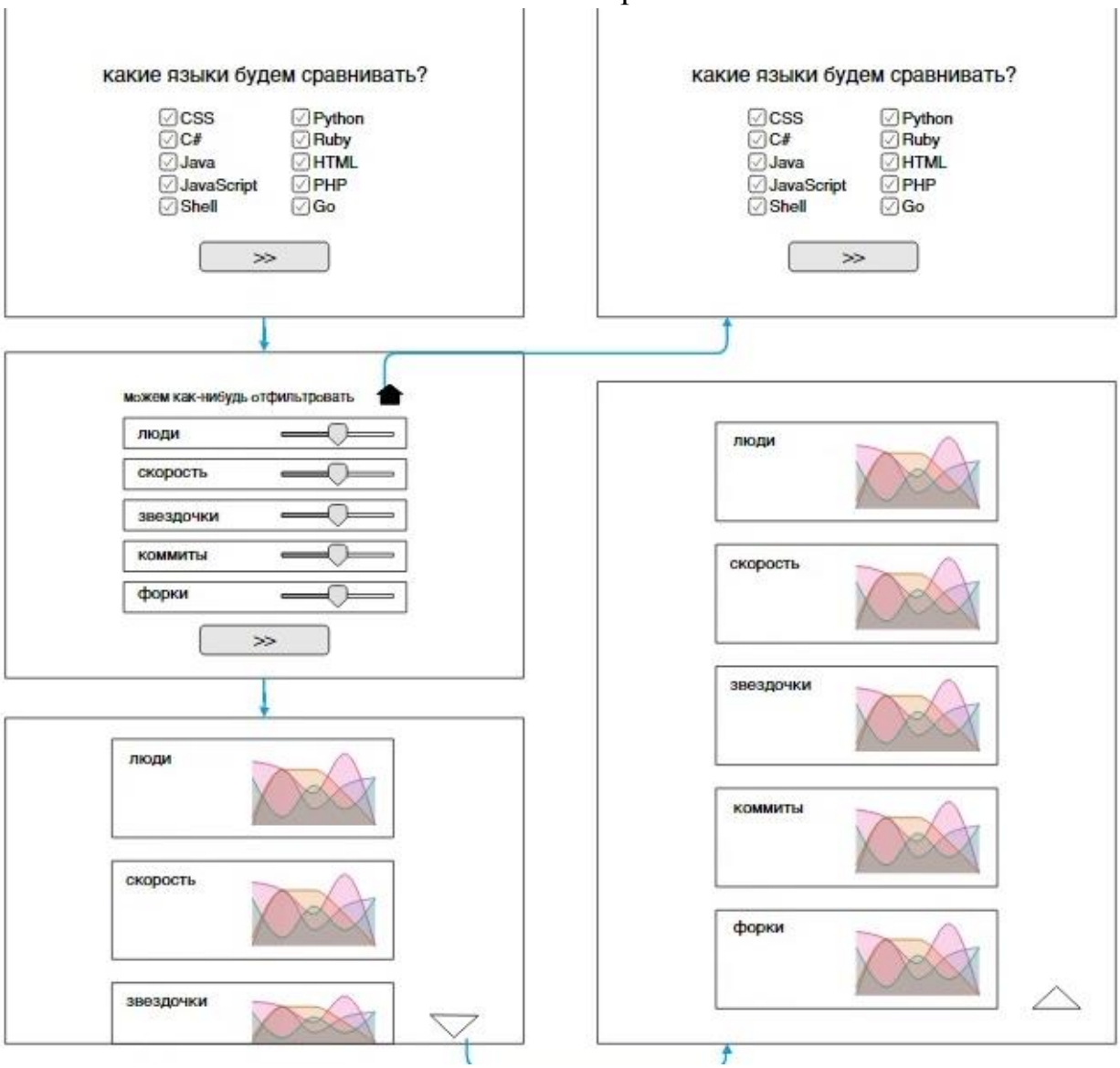

Рис. 1. Макет UI

2.1. Описание сценария использования для различных задач

1 экран: пользователю предлагается выбрать языки, которые далее будем сравнивать. На экране представлена одна кнопка перехода к экрану 2.

2 экран: на нём представлен список параметров, по которым будут анализироваться выбранные языки. Пользователь может к каждому из параметров применить фильтр. Также страница содержит результаты работы программы. Каждый блок представляет тот или иной параметр и содержит столбчатую диаграмму.

3. НЕРЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

3.1. Графическое представление

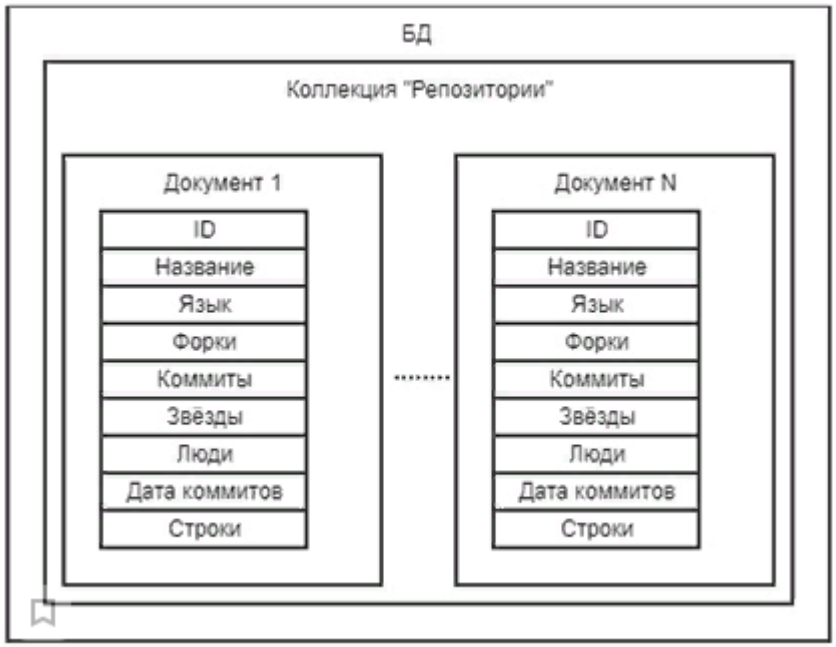

Рис. 2. Модель данных

Мы будем работать с характеристиками репозиториев, выбирать из них нужную нам информацию и отображать статистики. Для этого нам нужно работать с БД, в которой одна коллекция – это репозитории.

3.2. Описание назначений коллекций, типов данных и сущностей

В БД только одна коллекция – это репозитории. В коллекции хранятся документы разных репозиториев, а в документах имеются такие поля как:

- ID (репозитория): ObjectID
- Название (репозитория): array(string)
- Язык (на котором написан код): string
- Форки (Количество форков): int
- Коммиты (Количество коммитов): int
- Звёзды (Количество звёзд): int
- Люди (Комичество людей): int
- ПервыйКоммит (День, когда был сделан первый коммит): Date

- ПоследнийКоммит (День, когда был сделан последний коммит): Date
- Строки (Количество строк): int
- Скорость (Количество строк, делённое на промежуток времени): float

3.3. Оценка удельного объема информации, хранимой в модели

Рассчитаем, сколько места будет занимать каждый документ:

- ID: ObjectId – занимает 4 байта
- Название: string – пусть в среднем название репозитория состоит из 18 символов, тогда занимает 18 байтов
- Язык: array(string) – пусть в среднем название языка состоит из 5 символов – занимает 5 байтов
- Форки: int – занимает 4 байта
- Количество коммитов: int – занимает 4 байта
- Звёзды: int – занимает 4 байта
- Люди: int – занимает 4 байта
- ПервыйКоммит(День, когда был сделан первый коммит): Date - 8 байт
- ПоследнийКоммит (День, когда был сделан последний коммит): Date - 8 байт
- Строки: int – занимает 4 байта
- Скорость: float – занимает 4 байта

Каждый документ, занимает 62 байта, если считать без языка. Тогда, если N - это размер выборки (для какого количества репозиториев будем выводить статистику), а L - среднее количество используемых языков (в программе примем за величину 2), тогда формула для вычисления удельного объёма будет выглядеть так: $V = (62 + L * 5) * N$ (байт)

3.4. Запросы к модели, с помощью которых реализуются сценарии использования

1. Запрос на «создание» БД с нужными полями. Фактически это только скачивание нужной нам информации с GitHub.

   Форма запроса: GET /repos/:owner/:repo/

2. Запросы на добавление поля "Скорость":

db.collection.aggregate([{$project:{"id":1, "speed": {$devide[{$subtract: ["first_com", "last_com"]},"size"]}}}{$out:{speedfield}}])

Полученные значения из запроса aggregate передаём в update поля "speed"

db.speedfield.find({}).forEach(doc) db.collection.update($set:{"speed":sp})

В итоге получаем N+1 запрос.

3. Запрос на поиск репозиториев с определёнными пользователем параметрами:

db.collection.find({“language”: “lang”, “forks_count”: fork_num, “commits”:com_num, “star”:stars_num, “contributors”: contr_num, "first_com":"date_of_fc", "last_com":"date_of_lc", "size": size, "speed":sp})

4. АНАЛОГ МОДЕЛИ ДАННЫХ ДЛЯ SQL СУБД

4.1. Графическое представление

Рис. 3. Модель данных для SQL СУБД

4.2. Описание назначений коллекций, типов данных и сущностей

Таблица "Репозитории":

- ID (репозитория): длинное целое(ключ)
- Название (репозитория): varchar(20)
- Язык (на котором написан код): varchar(20)
- Форки: длинное целое
- Количество коммитов: длинное целое
- Звёзды: длинное целое
- Люди: длинное целое
- ПервыйКоммит: Дата/Время
- ПоследнийКоммит: Дата/Время
- Строки: длинное целое
- Скорость: float(20,5)

Таблица "Язык":

- ID языка: длинное целое (ключ)
- Язык (на котором написан код): varchar(20)

4.3. Оценка удельного объема информации, хранимой в модели

Таблица "Репозитории":

- ID: длинное целое(ключ) – занимает 4 байта
- Название репозитория: varchar(20) – 20 байт
- Форки: длинное целое – занимает 4 байта
- Количество коммитов: длинное целое – занимает 4 байта
- Звёзды: длинное целое – занимает 4 байта
- Люди: длинное целое – занимает 4 байта
- ПервыйКоммит: Дата/Время - 10 байт
- ПоследнийКоммит: Дата/Время - 10 байт
- Строки: длинное целое – занимает 4 байта
- Скорость: float(20,5) – занимает 20 байт

Таблица "Язык":

- ID языка: длинное целое(ключ)– 4 байт
- Язык (на котором написан код): varchar(20) – 20 байт

Каждый документ, занимает 84 байта, если считать без языка. Тогда, если N - это размер выборки (для какого количества репозиториев будем выводить статистику), а L - среднее количество используемых языков (в программе примем за величину 2), тогда формула для вычисления удельного объёма будет выглядеть так: $V = (84 + L * 24) * N$ (байт)

4.4. Запросы к модели, с помощью которых реализуются сценарии использования

1. 2 запроса на «создание» БД с нужными полями.

2. Запрос на добавление поля "Скорость".

   ALTER TABLE repository ADD speed float(20, 5); UPDATE repository SET speed= (lastcom- firstcom)/size;

3. 2 запроса на поиск репозиториев с определёнными пользователем параметрами (т.к. из двух таблиц).

SELECT * FROM repository WHERE “language”= “lang” AND “forks_count”= fork_num AND “commits”= com_num AND “star”=stars_num AND “contributors”= contr_num AND "first_com"="date_of_fc" AND "last_commit"="date_of_lc" AND "size"= size AND "speed":sp

4. 2 запроса на получение информации о каждом параметре (т.к. из двух таблиц)

5. СРАВНЕНИЕ МОДЕЛЕЙ

Данные занимают больший объём в реляционных базах данных. Это видно по формулам: Для SQL: V = (84 + L * 24) * N (байт) Для NoSQL: V = (62 + L * 5) * N (байт). Запросов в NoSQL больше по количеству, обусловлено тем, что в SQL реализация запросов на добавление поля и вычисление значений поля устроена удобнее. Также можно заметить, что в MongoDb в нашей БД вся информация, все поля лежат в одной коллекции, когда для SQL пришлось бы разделять данные по таблицам, из-за того, что иначе база будет занимать много места и возникнет угроза появления противоречивых данных.

6. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

6.1. Описание

Back-end представляет из себя python-приложение.

1. Получение данных из базы данных сайта github.com осуществлено с помощью запроса GET к api.github.com
2. Получение данных введённых пользователем:

1) @app.route("/init")

Request: На него приходит массив languages[].

Пример:

GET init?languages[]=Ruby&labguages[]=PHP

Do:

Метод делает запрос к гитхабу, применяя фильтр по языкам и добавляет полученные данные в базу данных mongo.

Response:

После своей работы метод возвращает ответ в формате json {"success": true.

return Response({"success": true}, status=200, mimetype='application/json')

2) @app.route("/list")

Request:

Все параметры фильтрации от – до:

speedFrom - speedTo

starFrom - starTo

sizeFrom - sizeTo

forkFrom - forkTo

Содержат в себе числа. Пример - list? startFrom=5&starTo=10 и тд.

Do:

Метод выполняет поиск по mongo, применяя данные фильтры. Если какой-то фильтр не пришел – он учитывается как от 0 до 9999999.

Response:

После работы метод возвращает ответ в формате json с информацией о полученной выборке.

Front-end – это web-приложение, которое использует API back-end приложения и отображает данные удобным образом для пользователя.

6.2. Скриншоты работы приложения

На данной странице можно выбрать репозитории с каким языком будем обрабатывать:

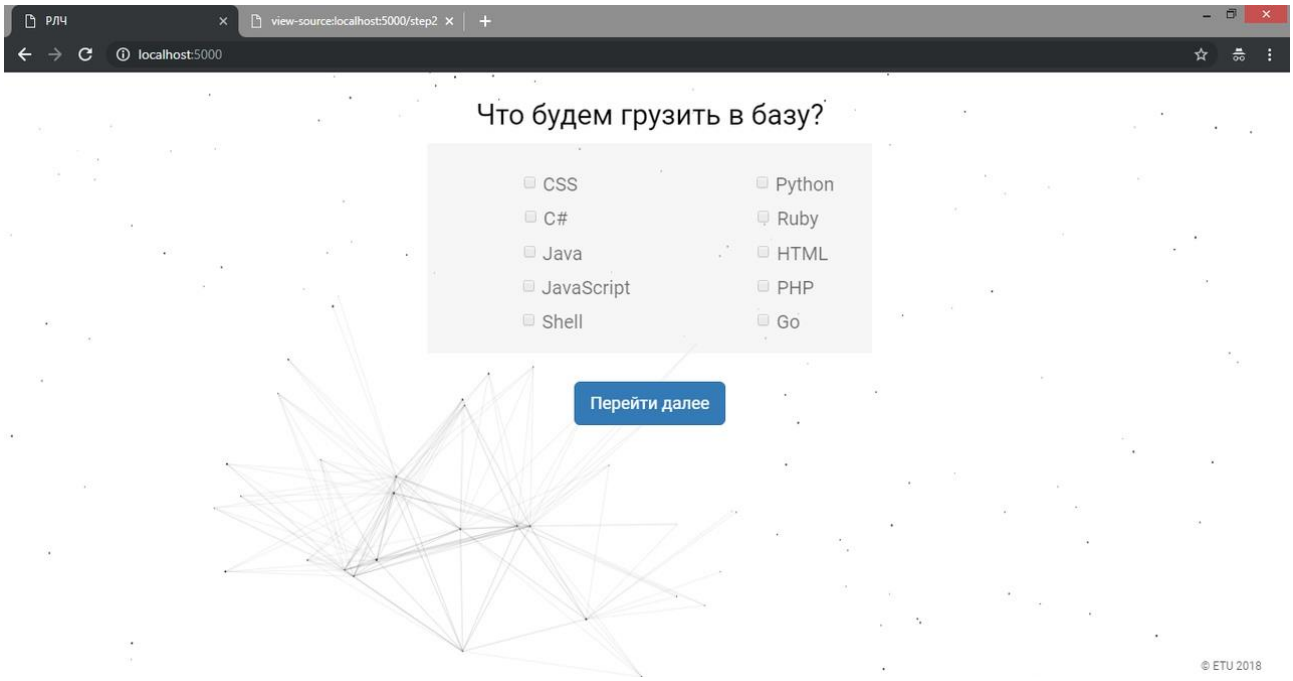

Рис. 4.. Экран с выбором языка, для дальнейшей работы

После нажатия на кнопку «перейти далее», открывается вторая страница:

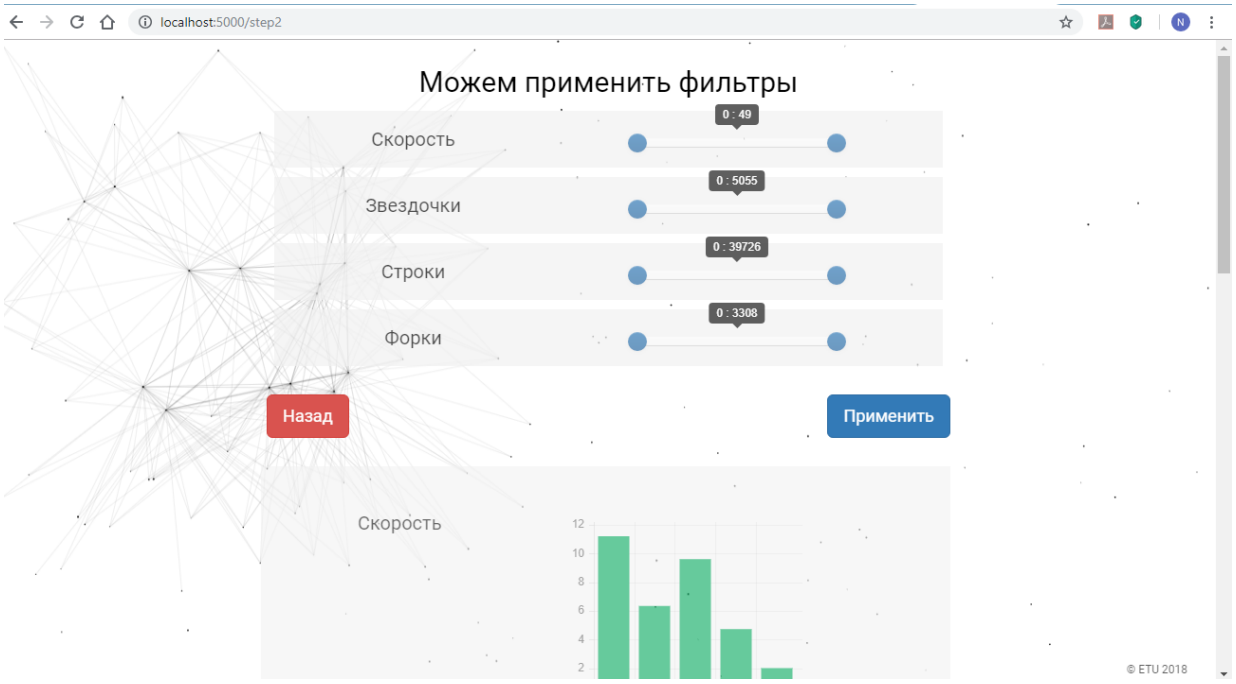

Рис. 5. Экран для применения фильтра

Здесь мы можем изменить значения параметров фильтрации:

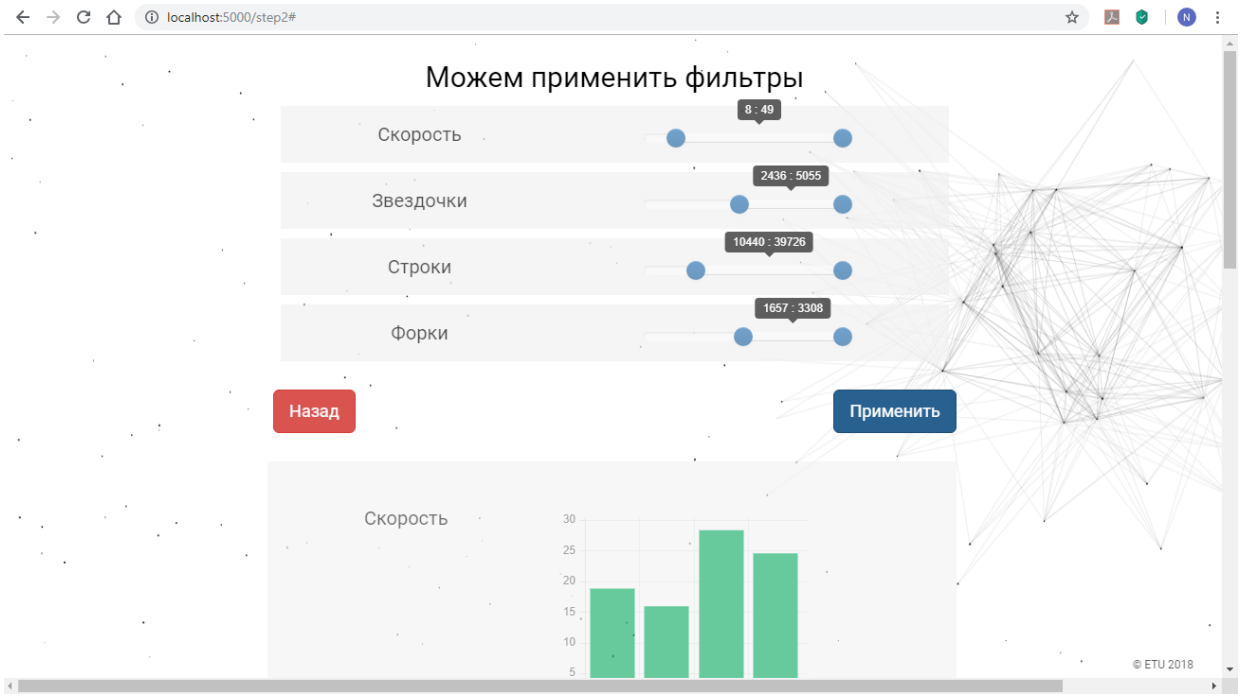

Рис. 6. Изменение значения параметров фильтрации

Ниже расположено четыре графика, которые можно увидеть, если прокрутить страницу вниз. В данном примере на первой странице было выбрано пять языков, поэтому их мы видим на графике:

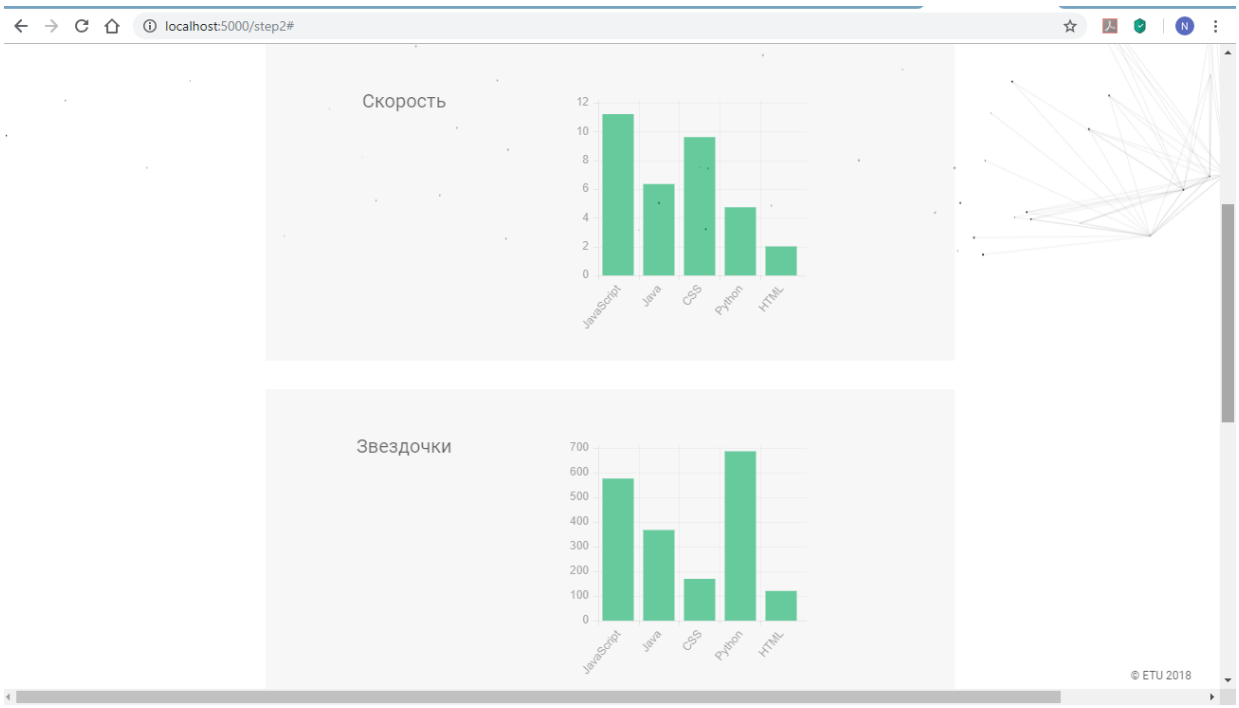

Рис. 7. Результат работы фильтрации (столбчатые диаграммы)

Применим фильтр по скорости и посмотрим на результат:

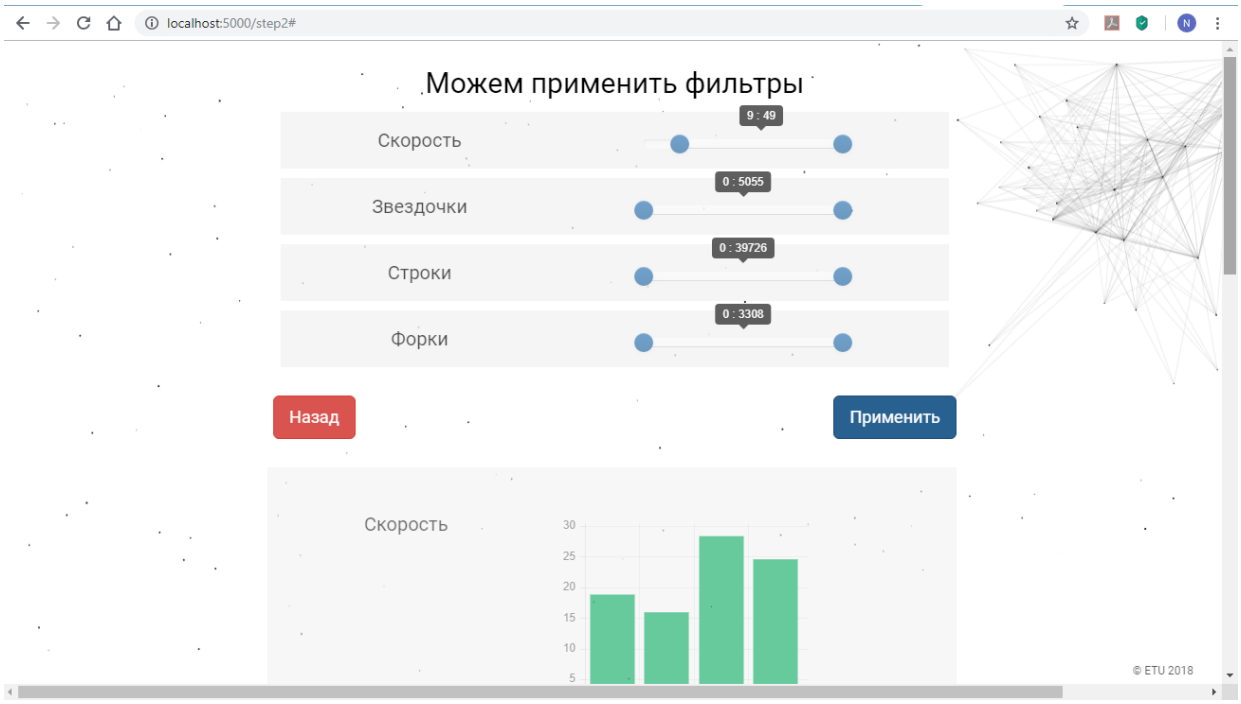

Рис. 8. Результат работы фильтрации (столбчатые диаграммы)

Видим, что количество языков изменилось, так как для одного из языков не существует ни одного репозитория с заданным пользователем значением скорости написания кода:

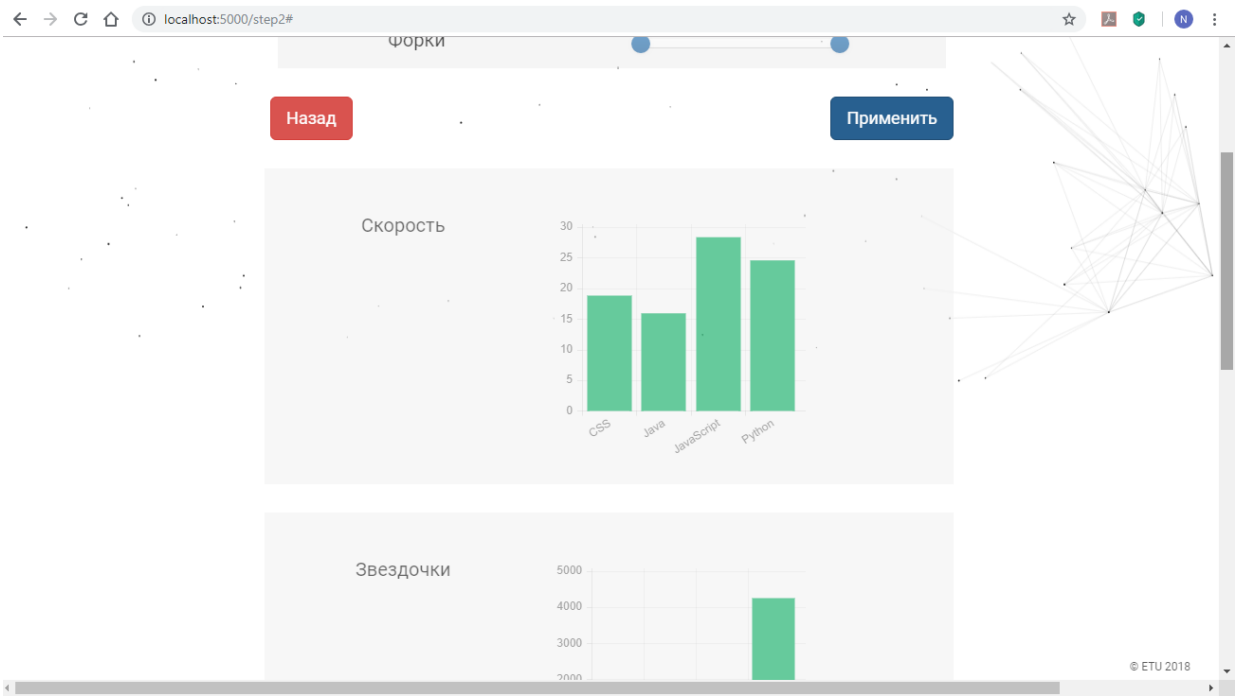

Рис. 9. Результат работы фильтрации (столбчатые диаграммы)

ЗАКЛЮЧЕНИЕ

В результате выполнения индивидуального домашнего задания было разработано пользовательское приложение для анализа данных OpenSource репозиториев. Веб-приложение было разработано с помощью средств MongoDB. Применение средств MongoDB хорошо подошло под выполняемую задачу и было успешным. Приложение корректно функционирует и удобно в использовании.

Ссылки на приложение:

Ссылка на github: https://github.com/moevm/nosql2018-os_github_analysis

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ

1. Скачать PyCharm, flask, pymongo, requests;
2. Скачать проект из репозитория (предыдущий пункт);
3. Запустить на компьютере файл server.py;
4. Открыть в браузере страницу http://localhost:5000/

ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА

Документация MongoDB: https://docs.mongodb.com/manual/