Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

ОТЧЕТ

по лабораторной работе №8

по предмету «Системы и технологии интеллектуальной обработки данных»

Запросы MongoDB с использованием MapReduce, Веб-сервис конвертации данных

|  |  |
| --- | --- |
| Выполнил: | Студент группы 772303  Виноград О.О. |
| Проверил:    Минск 2020 | Ассистент кафедры ЭИ Кунцевич А.А. |

**Указания к заданию**

1. Разработайте консольную утилиту для преобразования лога веб-сервера в формате CSV (Comma Separated Values), в формат JSON. Лог должен содержать поля со следующими названиями: URL, IP, timeStamp, timeSpent.

2. Разработайте запросы для загрузки полученных данных в формате JSON в СУБД MongoDB.

**Разработка запросов в СУБД MongoDB**

1. Разработайте следующие запросы, используя встроенные в СУБД MongoDB средствами выборки:

1) Выдать упорядоченный список URL ресурсов.

2) Выдать упорядоченный список IP-адресов пользователей, посетивших ресурс с заданным URL.

3) Выдать упорядоченный список URL ресурсов, посещенных в заданный временной период.

4) Выдать упорядоченный список URL ресурсов, посещенных пользовате-лем с заданным IP-адресом.

2. Разработайте следующие запросы, используя встроенные в СУБД MongoDB средства программирования на основе парадигмы MapReduce:

1) Выдать список URL ресурсов с указанием суммарной длительности по-сещения каждого ресурса, упорядоченный по убыванию.

2) Выдать список URL ресурсов с указанием суммарного количества посе-щений каждого ресурса, упорядоченный по убыванию.

**Результат**

Для выполнения заданий 1-2 я использовал сервер NodeJS.

• Исходные данные в формате CSV (Рисунок 1).

• Результат преобразования в формат JSON (Рисунок 2) с помощью эндпоинта сервера NodeJS.

• Результат загрузки данных в MongoDB (Рисунок 3)

• Выдать упорядоченный список URL ресурсов. (Рисунок 4)

• Выдать упорядоченный список IP-адресов пользователей, посетивших ресурс с заданным URL. (Рисунок 5)

• Выдать упорядоченный список URL ресурсов, посещенных в заданный временной период (Рисунок 6)

• Выдать упорядоченный список URL ресурсов, посещенных пользовате-лем с заданным IP-адресом (Рисунок 7)

• Выдать список URL ресурсов с указанием суммарной длительности посещения каждого ресурса, упорядоченный по убыванию (Рисунок 8)

• Выдать список URL ресурсов с указанием суммарного количества посе-щений каждого ресурса, упорядоченный по убыванию (Рисунок 9)

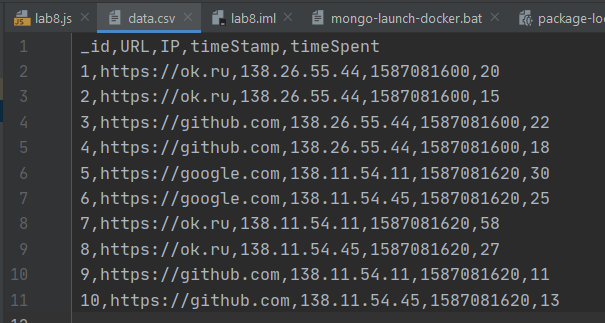
****

Рисунок 1 – Данные в формате CSV

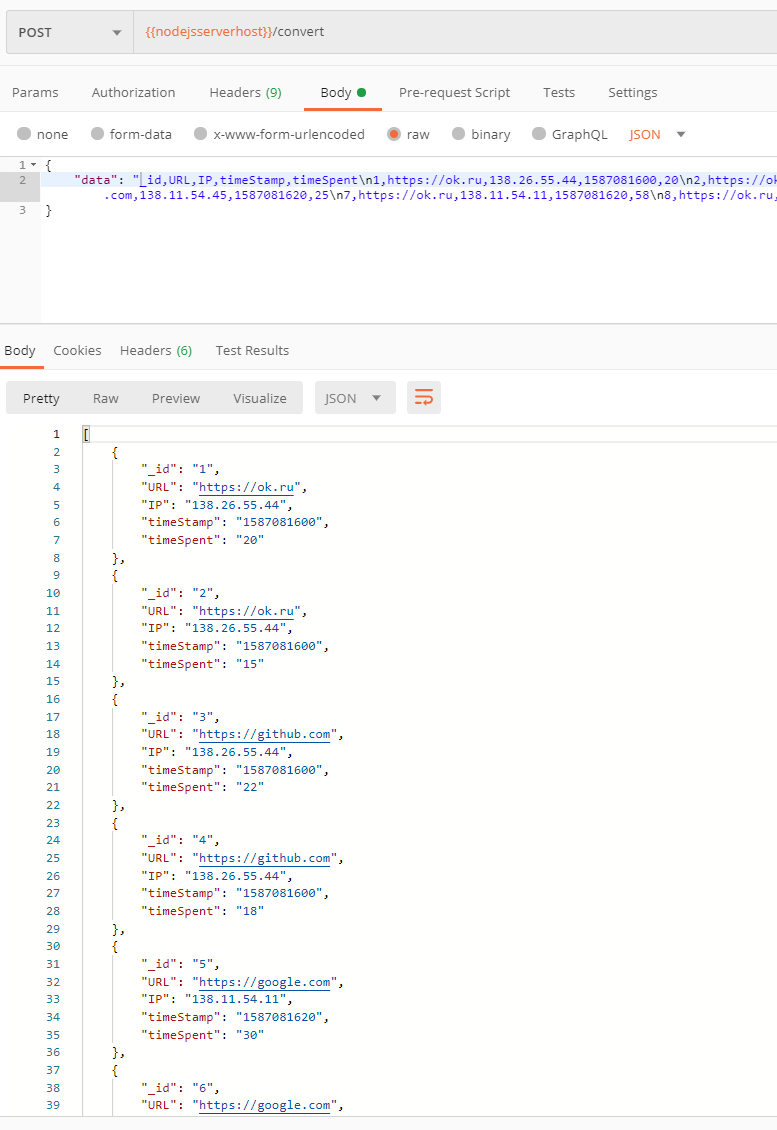


Рисунок 2 – Результат конвертации CSV в JSON

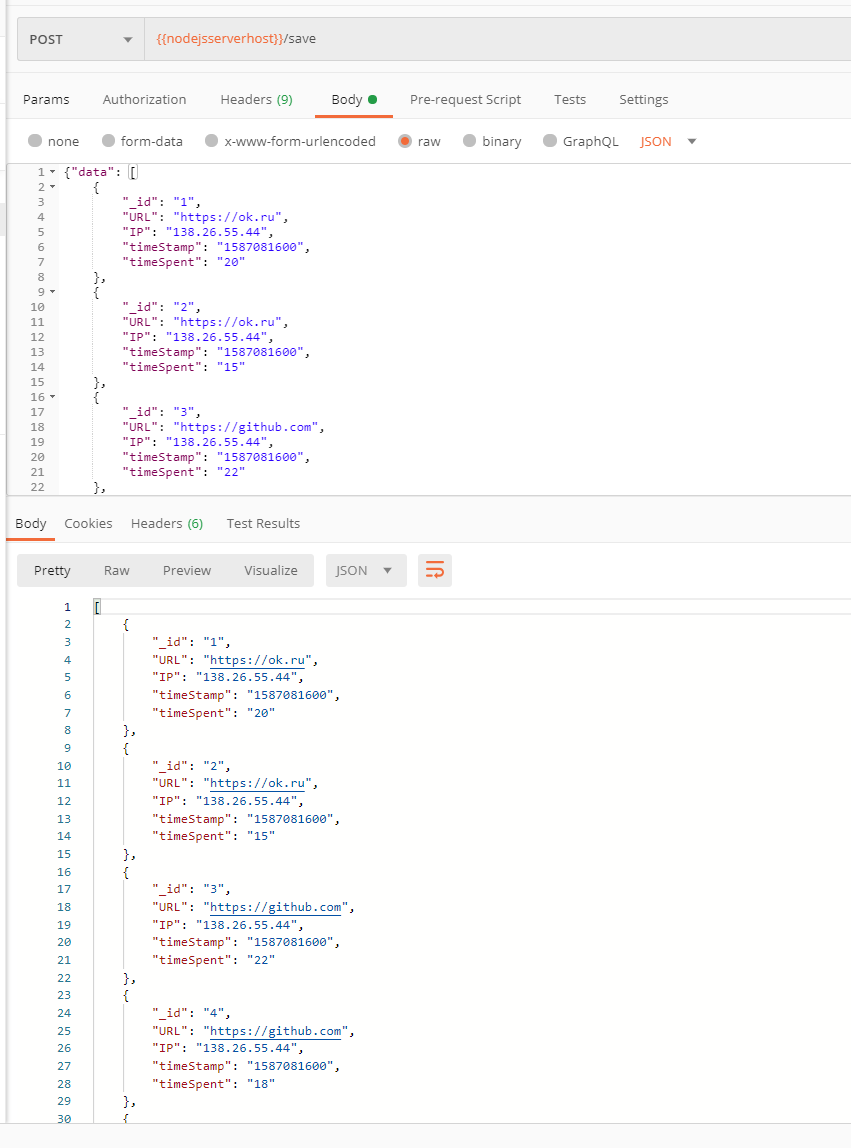


Рисунок 3 – Результат загрузки данных в MongoDB

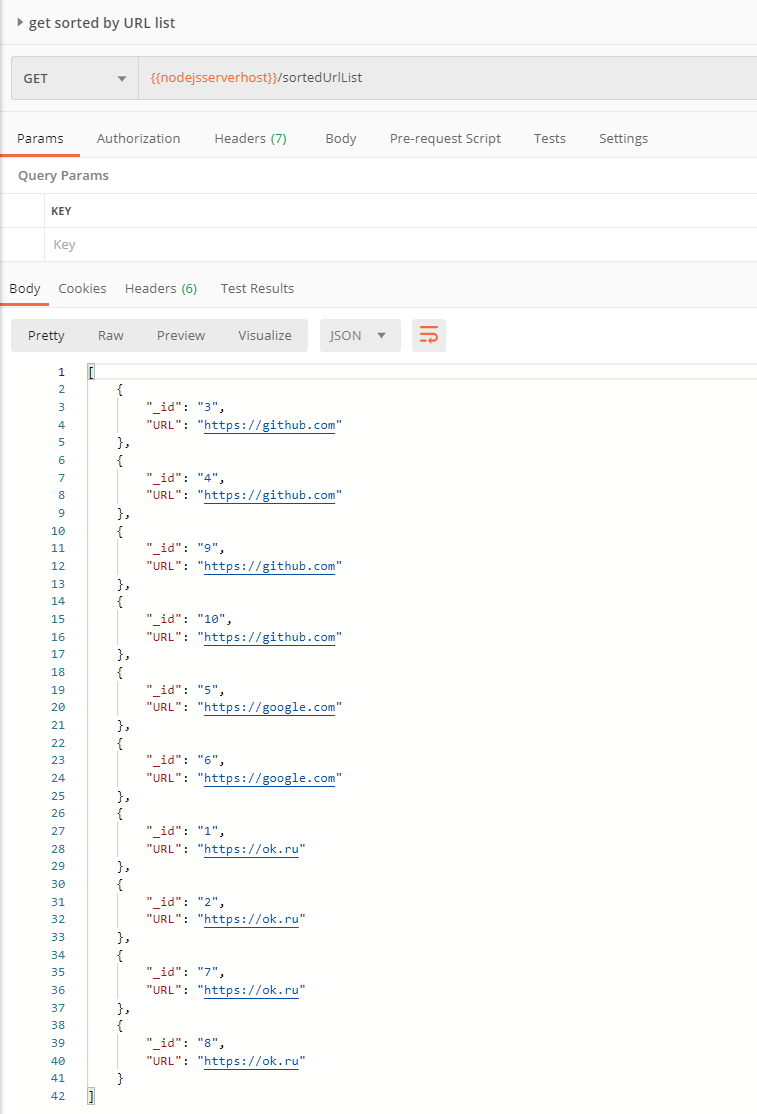


Рисунок 4 – Упорядоченный список URL

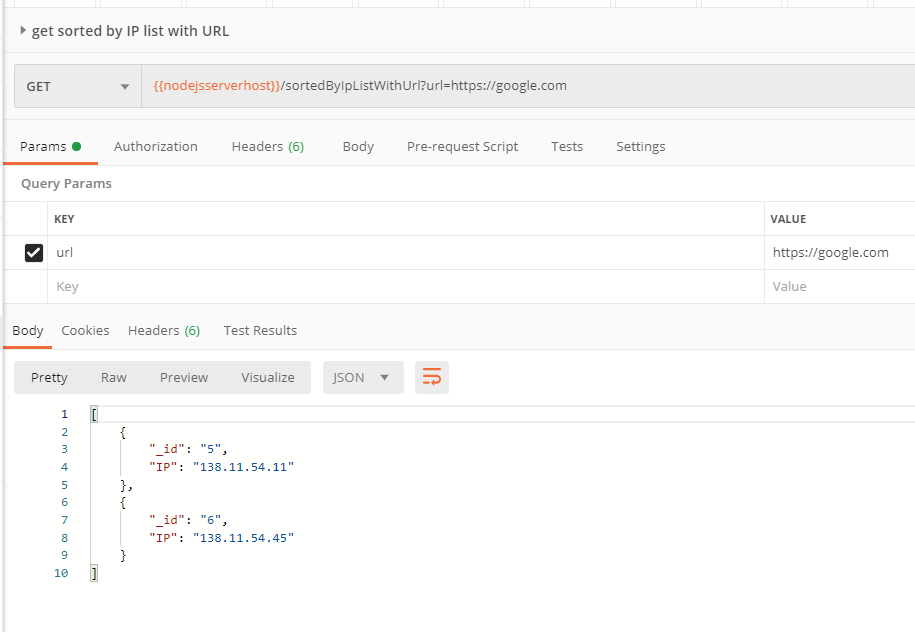


Рисунок 5 – упорядоченный список IP-адресов пользователей, посетивших ресурс с заданным URL

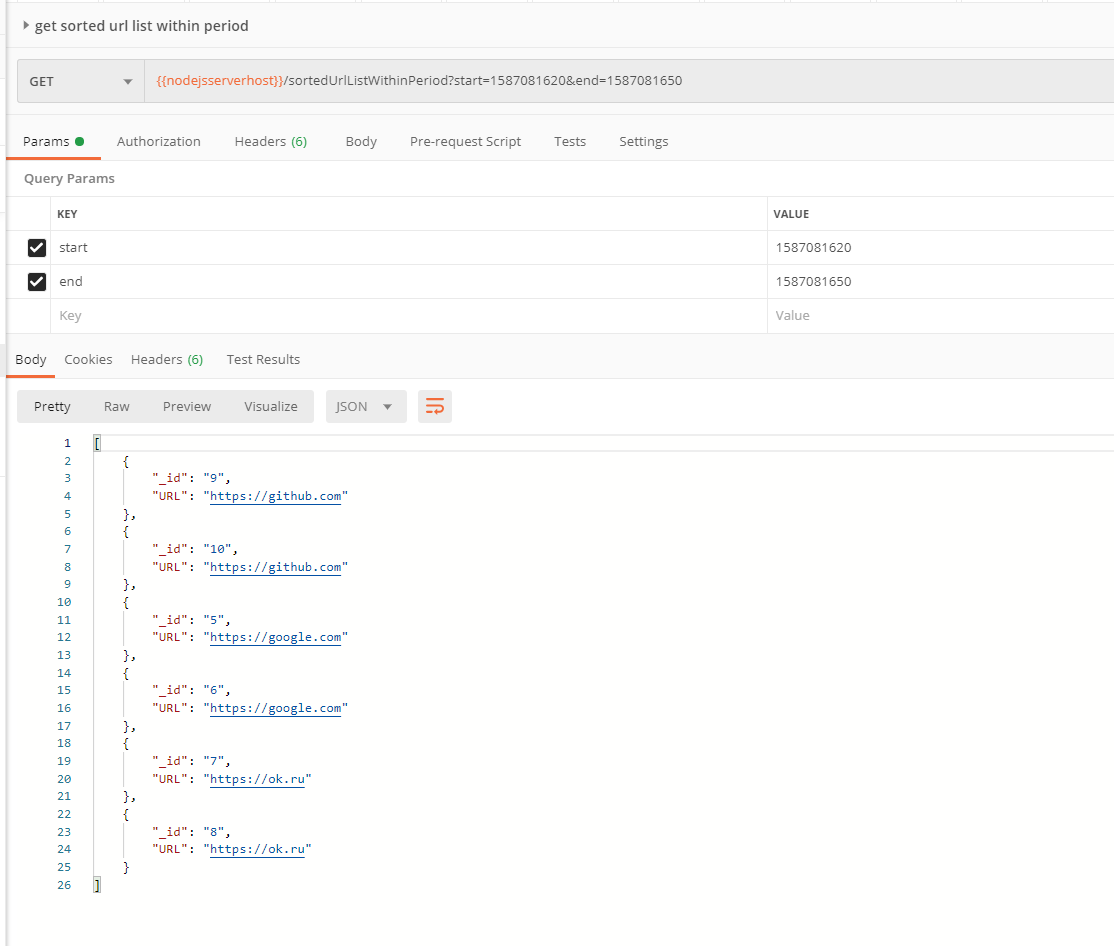


Рисунок 6 – упорядоченный список URL ресурсов, посещенных в заданный временной период

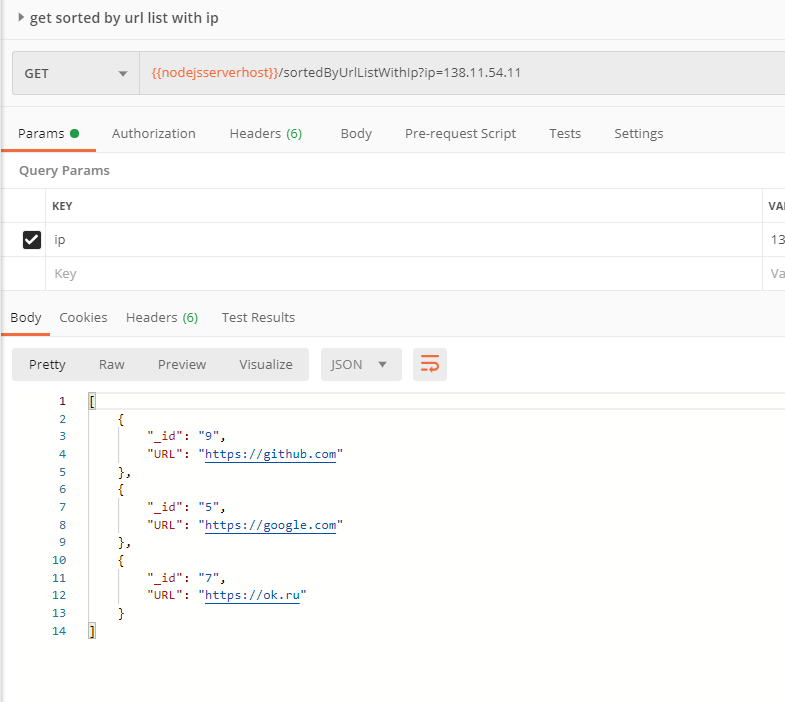


Рисунок 7 – упорядоченный список URL ресурсов, посещенных пользовате-лем с заданным IP-адресом

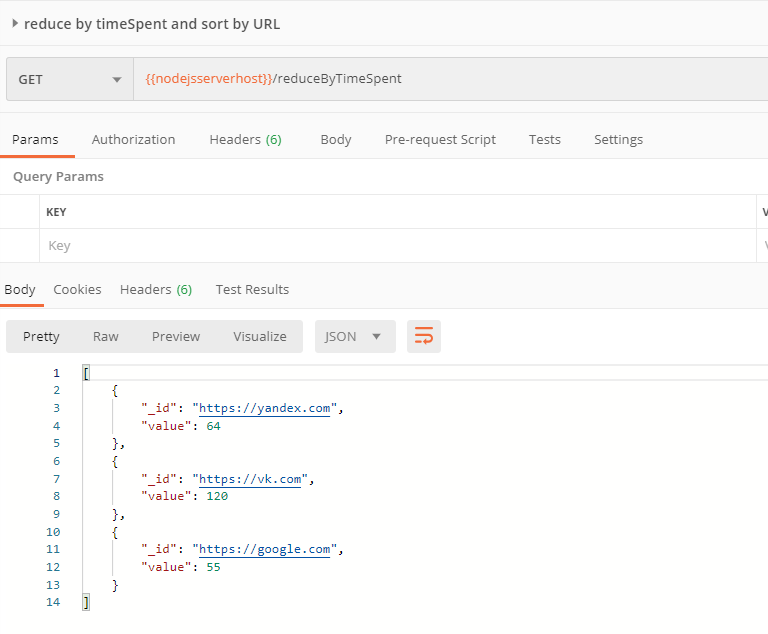


Рисунок 8 – список URL ресурсов с указанием суммарной длительности посещения каждого ресурса, упорядоченный по убыванию

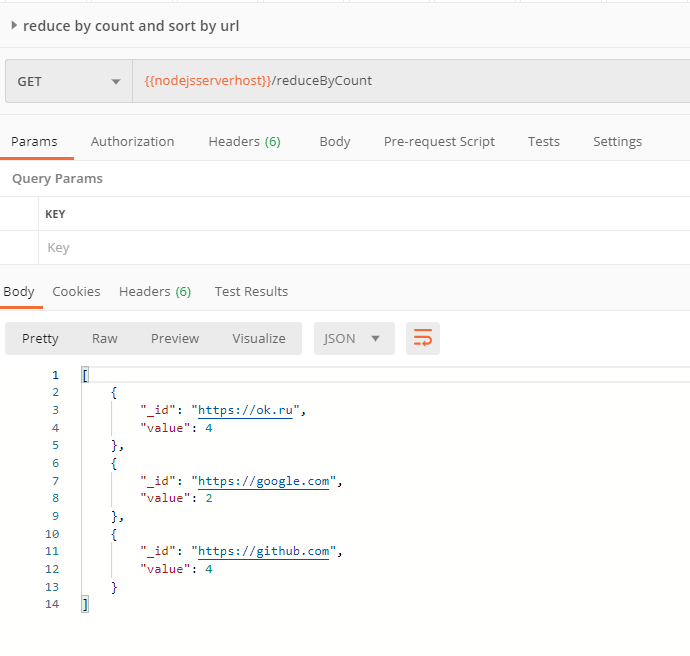


Рисунок 8 – список URL ресурсов с указанием суммарного количества посе-щений каждого ресурса, упорядоченный по убыванию

**Код сервера:**

const mongo = require('mongodb').MongoClient  
const url = "mongodb://lab:lab@localhost"  
const ***csv*** = require('csvtojson')  
const express = require('express')  
const jsonParser = express.json();  
const app = express()  
const port = 3000  
  
app.post('/convert', jsonParser, (request, response) => {  
 ***csv***()  
 .fromString(request.body.data)  
 .then((jsonObj) => {  
 response.send(jsonObj)  
 })  
})  
  
app.post('/save', jsonParser, (request, response) => {  
 let data = request.body.data;  
 executeMongoRequest((err, collection) => {  
 collection.insertMany(data, (err, result) => {  
 if (err) {  
 response.send(err)  
 return ***console***.log(err)  
 }  
 ***console***.log(result.ops)  
 response.send(result.ops)  
 })  
 })  
})  
  
app.get("/sortedUrlList", jsonParser, (request, response) => {  
 executeMongoRequest((error, collection) => {  
 collection.find({}, {fields: {URL: 1}}).sort({URL: 1}).toArray((error, result) => {  
 response.send(result)  
 })  
 })  
})  
  
app.get("/sortedByIpListWithUrl", jsonParser, (request, response) => {  
 executeMongoRequest((error, collection) => {  
 let url = request.query.url  
 collection.find({URL: url}, {fields: {IP: 1}}).sort({IP: 1}).toArray((error, result) => {  
 response.send(result)  
 })  
 })  
})  
  
app.get("/sortedByUrlListWithIp", jsonParser, (request, response) => {  
 executeMongoRequest((error, collection) => {  
 let ip = request.query.ip  
 collection.find({IP: ip}, {fields: {URL: 1}}).sort({URL: 1}).toArray((error, result) => {  
 response.send(result)  
 })  
 })  
})  
  
app.get("/sortedUrlListWithinPeriod", jsonParser, (request, response) => {  
 executeMongoRequest((error, collection) => {  
 let start = request.query.start  
 let end = request.query.end  
 collection.find({timeStamp: {$gte: start, $lte: end}}, {fields: {URL: 1}}).sort({URL: 1})  
 .toArray((error, result) => {  
 response.send(result)  
 })  
 })  
})  
  
function mapReduceByTimeSpent() {  
 emit(this.URL, this.timeSpent);  
}  
  
function reduceReduceByTimeSpent(keyURL, valuesTimeSpents) {  
 return valuesTimeSpents.reduce((a, b) => ***Number***.parseInt(a) + ***Number***.parseInt(b))  
}  
  
app.get("/reduceByTimeSpent", jsonParser, (request, response) => {  
 mongo.connect(url, (err, client) => {  
 client.db("lab").collection("reduceByTimeSpent").drop({});  
 client.close()  
 })  
 executeMongoRequest((error, collection) => {  
 collection.mapReduce(  
 mapReduceByTimeSpent,  
 reduceReduceByTimeSpent,  
 {out: "reduceByTimeSpent"}  
 );  
 })  
 mongo.connect(url, (err, client) => {  
 client.db("lab").collection("reduceByTimeSpent").find({}).sort({\_id: -1}).toArray((err, result) => {  
 if(err) {  
 response.send(err)  
 }  
 response.send(result)  
 })  
 client.close()  
 })  
})  
  
function reduceReduceByCount(keyURL, valuesTimeSpents) {  
 return valuesTimeSpents.length;  
}  
  
app.get("/reduceByCount", jsonParser, (request, response) => {  
 mongo.connect(url, (err, client) => {  
 client.db("lab").collection("reduceByCount").drop({});  
 client.close()  
 })  
 executeMongoRequest((error, collection) => {  
 collection.mapReduce(  
 mapReduceByTimeSpent,  
 reduceReduceByCount,  
 {out: "reduceByCount"}  
 );  
 })  
 mongo.connect(url, (err, client) => {  
 client.db("lab").collection("reduceByCount").find({}).sort({\_id: -1}).toArray((err, result) => {  
 if(err) {  
 response.send(err)  
 }  
 response.send(result)  
 })  
 client.close()  
 })  
})  
  
app.listen(port, (err) => {  
 if (err) {  
 return ***console***.log('something bad happened', err)  
 }  
 ***console***.log(`server is listening on ${port}`)  
})  
  
app.post('/save', jsonParser, (request, response) => {  
 let data = request.body.data;  
 executeMongoRequest((err, collection) => {  
 collection.insertMany(data, (err, result) => {  
 if (err) {  
 response.send(err)  
 return ***console***.log(err)  
 }  
 ***console***.log(result.ops)  
 response.send(result.ops)  
 })  
 })  
})  
  
function executeMongoRequest(request) {  
 mongo.connect(url, (err, client) => {  
 request(err, client.db("lab").collection("lab"))  
 client.close()  
 })  
}  
  
  
// Use it to drop the collection  
/\*  
mongo.connect(url, function (err, client) {  
 executeMongoRequest((error, collection) => {  
 collection.drop({}, (error, result) => console.log(result))  
 collection.find({}).toArray((error, result) => {  
 if (error) throw error;  
 console.log("Collection data successfully removed");  
 });  
 })  
})  
\*/