Final Problem Report

By ALGERA Pieter, BAALI Karim
and ALBIZZATI Grégoire

Table des matières

Back-end

The back-end is made using Node.js, express, mongoose, body-parser and mongoDB. These modules allow us to provide endpoints which are used by the client. MongoDB is a database is a free and open-source cross-platform document-oriented database program. It It's widely used for data which are not relational.
The server listens on the port 8081.

There is three main important endpoints :

- /saveQuery, which allow the client to save a new custom query
- /getQueries which return all the saved queries from the server
- /executeFullCustomQuery permits to execute one the queries the user previously saved.

Parser

The parser is the part of the server which takes care of executing any custom queries with a list of parameters. The idea is to replace the parameters with '???' in the query.
Then we send it to the end point with all the parameters separated by a semi-colon.
The parser will split the query on the ??? sign and split the parameters on the semi-colon. So at this point we end up with two arrays. Thus the parser just have to merge the two arrays in the right order. It's going to insert the parameters at the very same place the '???' occupied before.
Then the parser will convert the json-like query into a javascript object (using the JSON.parse() function). This object is hence ready to be used by mongoose as a mongo query.
Please mind that the parser only parses queries to be given to an aggregate function, that is why it starts and ends with brackets.

Database

For this project on MongoDB database, we are still using the dataset named “Companies2.json”. It contains different kind of information about a very large set of company object, such as a name, a website URL or the lists of customers or the number of employees. It allows us to be able to query a lot of different information based on various parameters. For example, a little part of the first object in this dataset :


```
{
   "_id":{
      "$oid":"5a5c533c942d09e481c157e7"
   },
   "name":"Vidyo",
   "permalink":"vidyo",
   "crunchbase_url":"http://www.crunchbase.com/company/vidyo",
   "homepage_url":"http://www.vidyo.com",
   "blog_url":"",
   "blog_feed_url":"",
   "twitter_username":"Vidyo",
   "category_code":"messaging",
```


The 6 predefined queries are :

- [EASY] Get the companies which have more than N products.

```
var nbProducts = 3
db.getCollection('companies').find({ $where: "this.products.length > " + nbProducts })
```

- [EASY] Get the top 3 documents ordered by number of employees.

```
db.getCollection('companies').aggregate([{$match: {}}, { $sort : { "number_of_employees" : -1} }, {$limit: 30}])
```

- [MEDIUM] Get the companies with at least a given number of employees.

```
var nmbrOfEmployees = 400000
db.getCollection('companies').aggregate([
{$match: {number_of_employees: {$gt: nmbrOfEmployees}}},
{$project: {number_of_employees: 1, name: 1}},
{$sort: {number_of_employees: -1}}
]);
```

- [MEDIUM] Get the max number of employees for each category code.

```
db.getCollection('companies').aggregate([
{$sort: {category_code: -1}},
{$project: {number_of_employees: 1, category_code: 1}},
{$group:{_id:"$category_code", employesMax:{$max:"$number_of_employees"}}}
]);
```

- [HARD] Get the list and number of companies grouped by city in which they have an office.

```
db.getCollection('companies').aggregate([
{$unwind: "$offices" },
{$group: {_id : "$offices.city", companies: { $push: "$name" }, count: { $sum: 1 }}}
]);
```

- [HARD] Get the name of companies which were created before a given year and have more than N providers.

```
var year = 2000
db.getCollection('companies').aggregate([
{$match: {founded_year: {$lt: year}}},
{$match: {providerships: {$size:2}}},
{$project: {founded_year: 1, providerships: 1}}
]);
```

Frontend and user interface

A navbar is available to easily navigate into the main page, and to access to the admin page. Three different methods are available to perform queries on the database, allowing the user to have different levels of customization.

The first one is the simplest one : 6 buttons are available on the first section of the website, each one triggering a post request with one of the previous requests. The results are displayed on a popup as JSON objects. To do so, we used a JavaScript library called TreeJsonViewer.

The second alternative allows the user to customize the parameters used in the previous requests. A tab menu is available in the second section of the website to display a form. The user can enter the needed parameters to perform a custom query on the database. The results are also displayed on a popup as JSON objects.

The third option allows the user to select a previously saved query and provide a list of parameters to customize the selected query. A list of available queries is displayed in a dropdown list, and the different parameters need to be separated by “;”. Previously, the user need to use the admin mode to add a pre-formatted query in the database.

Using the site

To start the server. Run a mongoDB instance and while placed inside the project folder. Run the following command: node MongoNode.js
This will launch the server and allow the user to be able to go and use the website from their browser

The website can be located at the following URL: localhost:8081

The website is design in a one page setup for non-admin users. You can select one of the color squares to launch a preset query. You can use a form to change the parameters of a predefined query and lastly, you can choose a custom query that was entered by an Admin.

To go to the Admin page. Click on the button admin in the upper right hand corner. You will be taken to the login page. To login, type “Admin” for the username and “admin” for the password. If the combination is correct, you will be sent to the Admin page. Here you can enter a query that will be parameterized. Following the onscreen instructions to add a new query and pushing the submit button underneath the page to send the query to the database. You can click on the home button to go back to the main page. You can now see your new queries at the bottom of the page.