Advanced topics in NoSQL databases :

Final Project

Team members :

Jérémy MIARA

Marine MEZIN

Elodie TOROSSIAN

Database :

DBLP_clean (Difficulty 2)

What NoSQL Database did we choose ?

We chose MongoDb because it’s a document oriented NoSQL database and it was very easy to import our dataset into the database.

What application handler did we choose ?

We chose NodeJS to make the link between our database and our website made in HTML and CSS. It displays queries results and modifies them. By default you are considered as a simple user. You have to connect to the administrator mode to add queries.

How to install our application ?

1. Go into our application folder and enter this two command lines :
   npm install
   npm install express
   npm install mongodb

These commands will install all necessary packages needed to start our application.

2. In a new terminal, go into the folder “bin” of MongoDB. Your path should look like this : *“C:\Program Files\MongoDB\Server\3.6\bin”*

3. Import data into MongoDb : *mongoimport --db dblp --collection publis “PATH”\DBLP_clean.json*

How to start our application ?

1. In a new terminal, go into the folder “bin” of MongoDB. Your path should look like this : *“C:\Program Files\MongoDB\Server\3.6\bin”*
2. Enter : mongod.exe

3. In a new terminal, go into the folder of our project

4. Enter : node app.js

Now your application just started !

5. Go in your browser and write localhost:3000 (if you didn’t change the port)

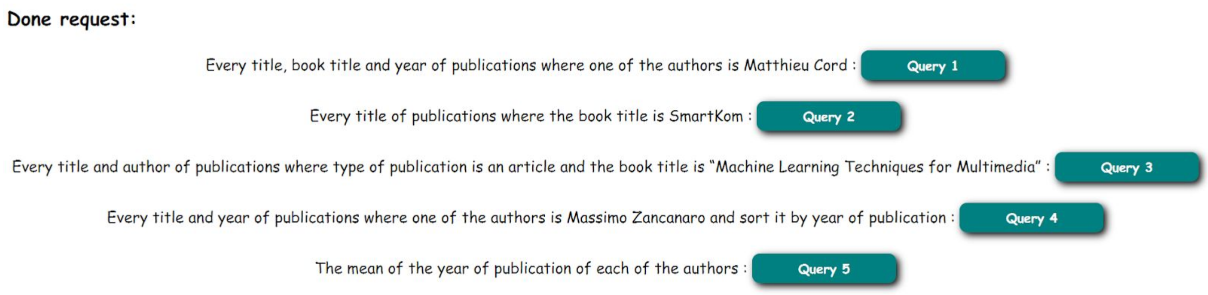
Dataset importation

As said in our “How to install our application ?” part, we used “mongoimport” to import our dataset into the MongoDb database.

Command line : *mongoimport --db dblp --collection publis “PATH”\DBLP_clean.json*

Queries available for a simple user

- Pre-registered queries:



Done request:

Every title, book title and year of publications where one of the authors is Matthieu Cord :

Every title of publications where the book title is SmartKom :

Every title and author of publications where type of publication is an article and the book title is “Machine Learning Techniques for Multimedia” :

Every title and year of publications where one of the authors is Massimo Zancanaro and sort it by year of publication :

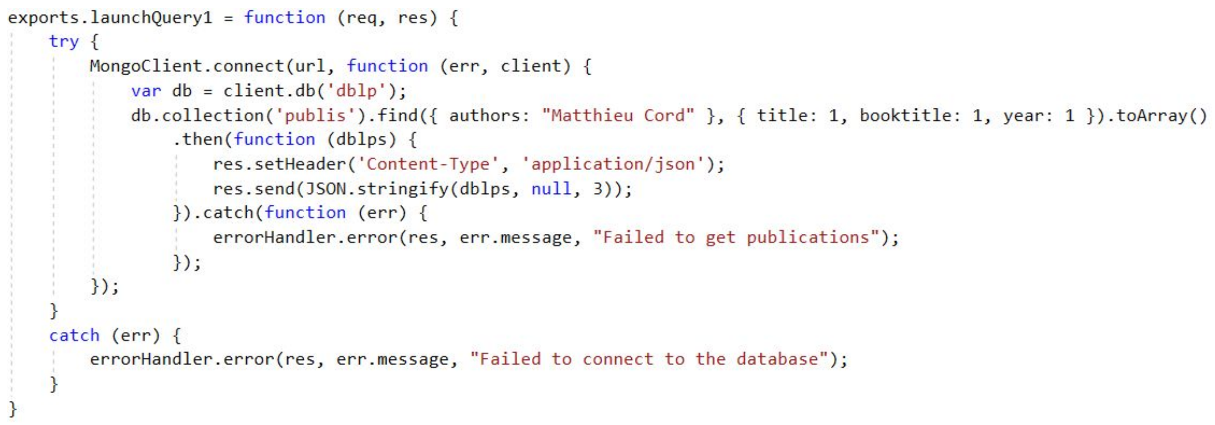
The mean of the year of publication of each of the authors :

In our code, we used the five queries we created in our MongoDb report during the semester. When the user click on a button he will receive the result of the query defined in the little sentence next to the button.

Result of the first query:

```
[
   {
      "_id": "series/cogtech/CunninghamCD08",
      "type": "Article",
      "year": 2008,
      "title": "Supervised Learning.",
      "authors": [
         "P?draig Cunningham",
         "Matthieu Cord",
         "Sarah Jane Delany"
      ],
      "pages": {
         "start": 21,
         "end": 49
      },
      "booktitle": "Machine Learning Techniques for Multimedia",
      "journal": {
         "series": null,
         "editor": null,
         "volume": null,
         "isbn": []
      },
      "url": "db/series/cogtech/354075170.html#CunninghamCD08",
      "cites": []
   },
   {
      "_id": "series/cogtech/CordG08",
      "type": "Article",
      "year": 2008,
      "title": "Online Content-Based Image Retrieval Using Active Learning.",
      "authors": [
         "Matthieu Cord",
         "Philippe Henri Gosselin"
      ],
      "pages": {
         "start": 115,
         "end": 138
      },
      "booktitle": "Machine Learning Techniques for Multimedia",
```

Every of our queries were defined in the dblpController.js file. For example the first query looks like this.



```
exports.launchQuery1 = function (req, res) {
    try {
        MongoClient.connect(url, function (err, client) {
            var db = client.db('dblp');
            db.collection('publis').find({ authors: "Matthieu Cord" }, { title: 1, booktitle: 1, year: 1 }).toArray()
                .then(function (dblps) {
                    res.setHeader('Content-Type', 'application/json');
                    res.send(JSON.stringify(dblps, null, 3));
                }).catch(function (err) {
                    errorHandler.error(res, err.message, "Failed to get publications");
                });
        });
    }
    catch (err) {
        errorHandler.error(res, err.message, "Failed to connect to the database");
    }
}
```

First we make a connection with our database. Then, we tell our program which database to use. After, we send our query into the collection we wanted. If everything went well, we return the result ; if not we return an error message.

Our four other queries were handled like the one above. Below, we will only show you the part were we wrote the query for the database.

Query 2: *db.collection('publis').find({ booktitle: "SmartKom" }, { title: 1 })*

Query 3: *db.collection('publis').find({ type: "Article", booktitle: "Machine Learning Techniques for Multimedia" }, { title: 1, authors: 1 })*

Query 4: *db.collection('publis').aggregate([*

{ $match: { authors: "Massimo Zancanaro" } },

{ $project: { title: 1, year: 1 } },

{ $sort: { year: 1 } }])

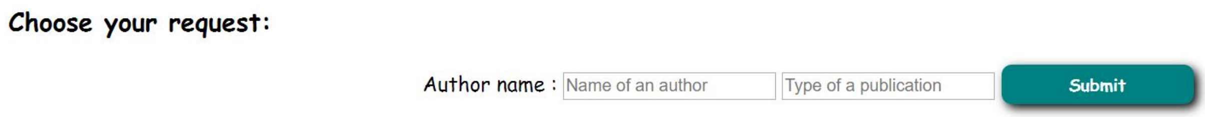
Query 5: *db.collection('publis').aggregate([*

{ $unwind: "$authors" },

{ $group: { _id: "$authors", mean_year_pub: { $avg: "$year" } } },

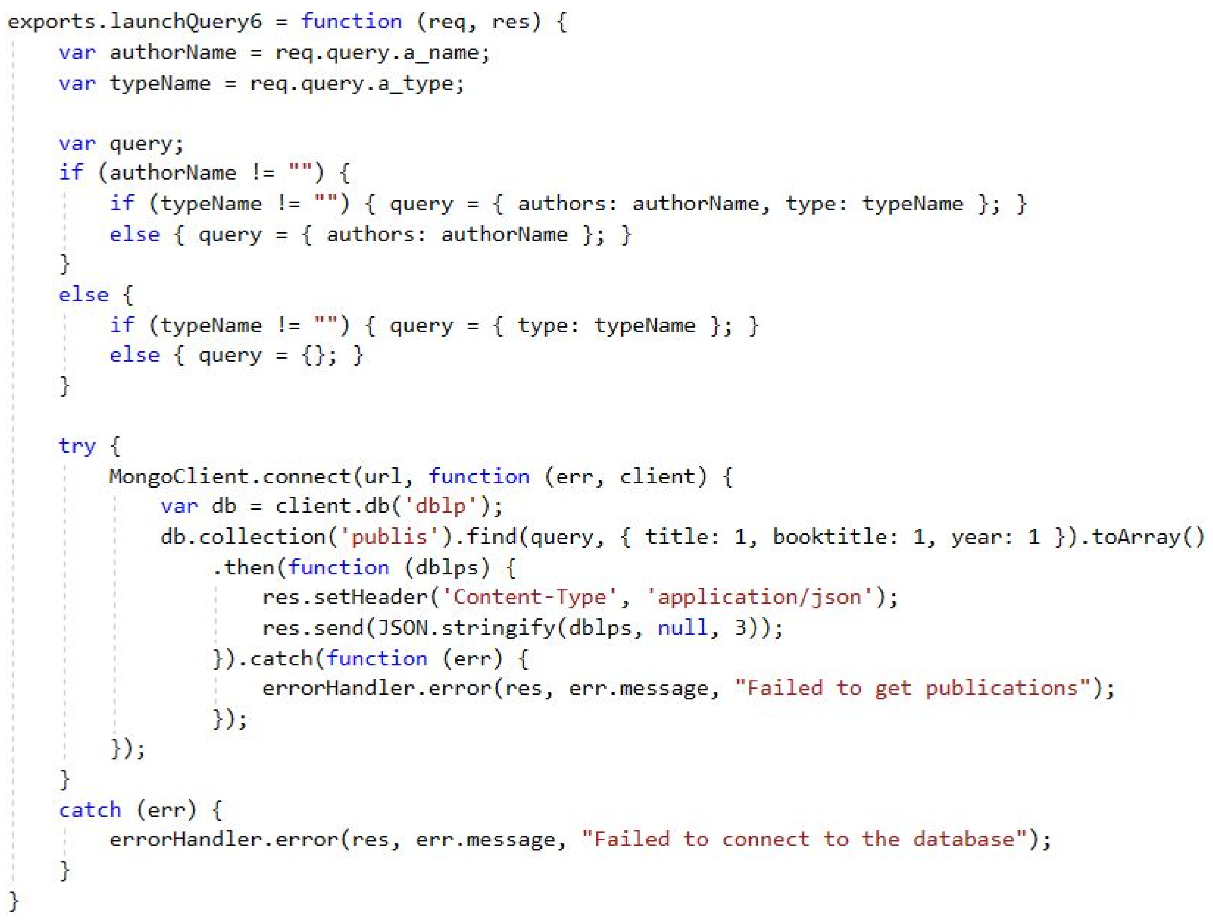
{ $sort: { mean_year_pub: 1 } }])

- Query with a modifiable parameter:



The user also have the possibility to personalize a query. Here he can enter an author name for example “Massimo Zancanaro”, or a type of publication for example “Article” or both of them. The user can choose between enter one information between those two or use the two areas.

Here it’s the code part where we handle this:

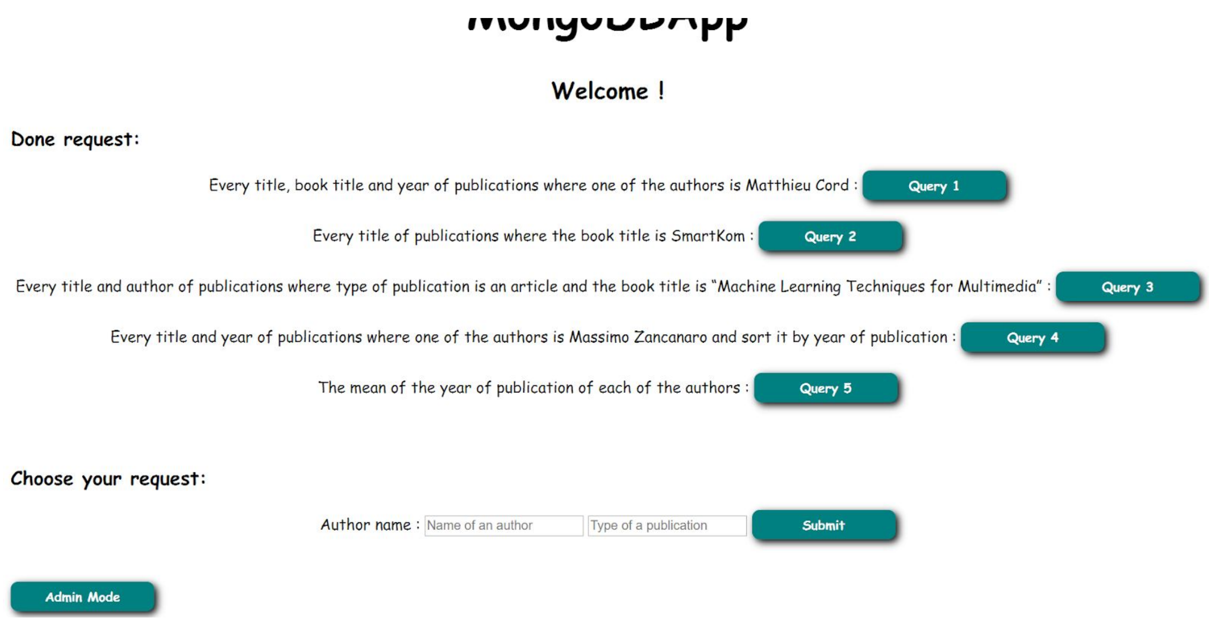


```
exports.launchQuery6 = function (req, res) {
    var authorName = req.query.a_name;
    var typeName = req.query.a_type;

    var query;
    if (authorName != "") {
        if (typeName != "") { query = { authors: authorName, type: typeName }; }
        else { query = { authors: authorName }; }
    }
    else {
        if (typeName != "") { query = { type: typeName }; }
        else { query = {}; }
    }

    try {
        MongoClient.connect(url, function (err, client) {
            var db = client.db('dblp');
            db.collection('publis').find(query, { title: 1, booktitle: 1, year: 1 }).toArray()
                .then(function (dblps) {
                    res.setHeader('Content-Type', 'application/json');
                    res.send(JSON.stringify(dblps, null, 3));
                }).catch(function (err) {
                    errorHandler.error(res, err.message, "Failed to get publications");
                });
        });
    }
    catch (err) {
        errorHandler.error(res, err.message, "Failed to connect to the database");
    }
}
```

How to login to go into the administrator mode ?

To log into the administrator mode the first thing you have to do is to click on the "admin mode" button, at the bottom of the home page.

Home page:



MongoDBApp

Welcome !

Done request:

Every title, book title and year of publications where one of the authors is Matthieu Cord : Query 1

Every title of publications where the book title is SmartKom : Query 2

Every title and author of publications where type of publication is an article and the book title is "Machine Learning Techniques for Multimedia" : Query 3

Every title and year of publications where one of the authors is Massimo Zancanaro and sort it by year of publication : Query 4

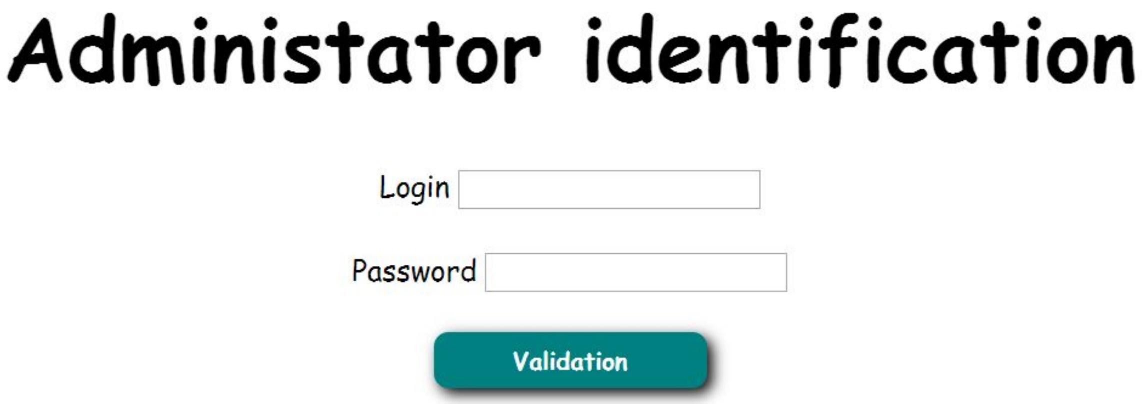
The mean of the year of publication of each of the authors : Query 5

Choose your request:

Author name : Name of an author | Type of a publication | Submit

Admin Mode

After that, you will be redirected on a login page. You will have to enter your login and your password to access to the admin page.



Administator identification

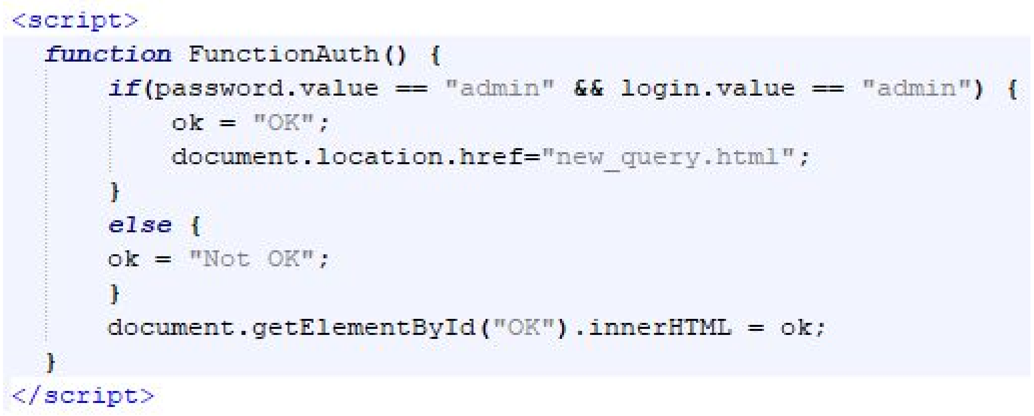
Login

Password

Validation

For the moment, the login is “admin” and the password is “admin” too.

Here the javascript code related with the authentication:

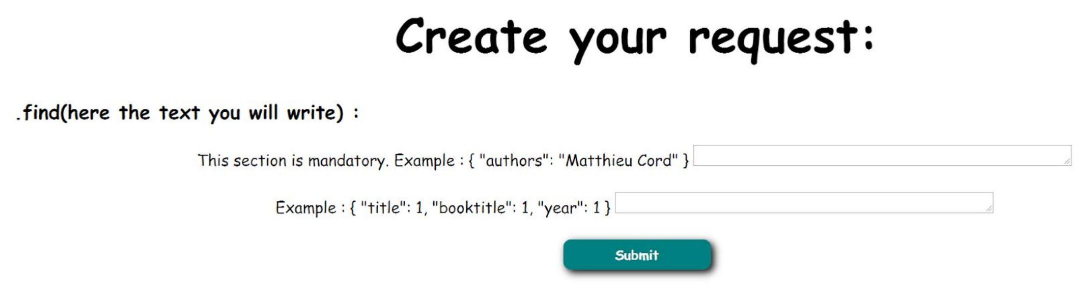


```
<script>
  function FunctionAuth() {
      if(password.value == "admin" && login.value == "admin") {
          ok = "OK";
          document.location.href="new_query.html";
      }
      else {
      ok = "Not OK";
      }
      document.getElementById("OK").innerHTML = ok;
  }
</script>
```

Finally, you will go into a page were you will have access to all the functionality reserved for the administrator.

Queries available for an administrator

An administrator have the possibility to create a query and submit it to obtain a result with our database.



Create your request:

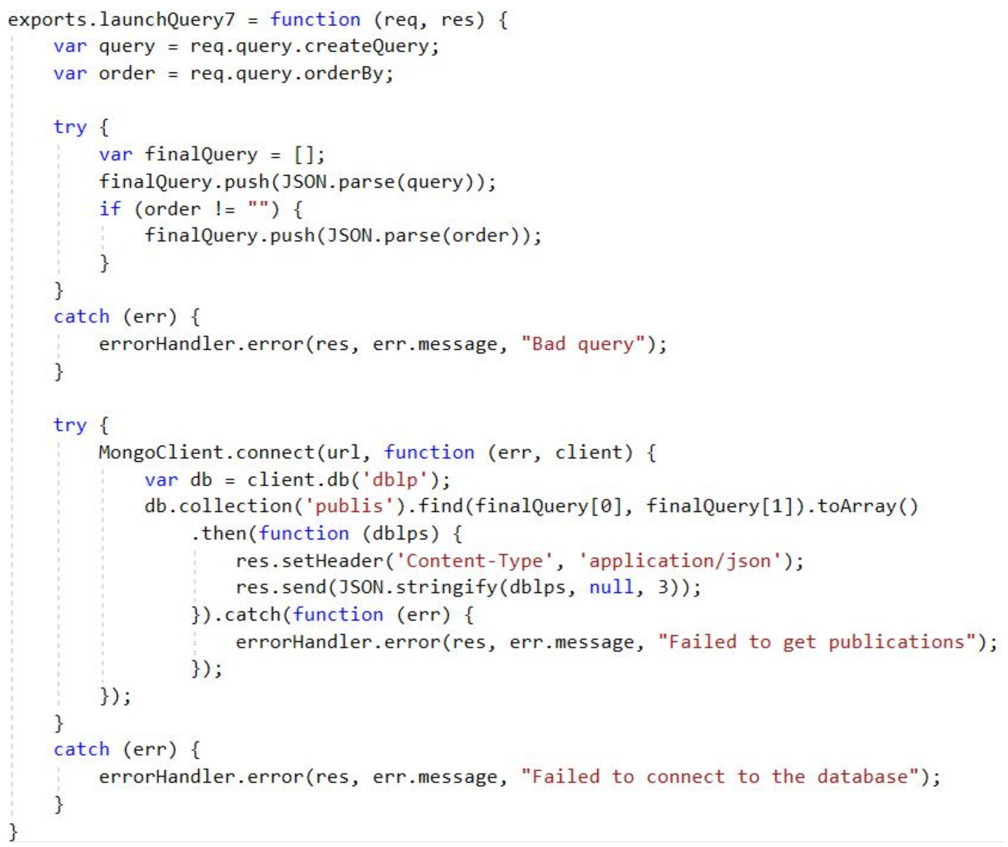
.find(here the text you will write) :

This section is mandatory. Example : { "authors": "Matthieu Cord" }

Example : { "title": 1, "booktitle": 1, "year": 1 }

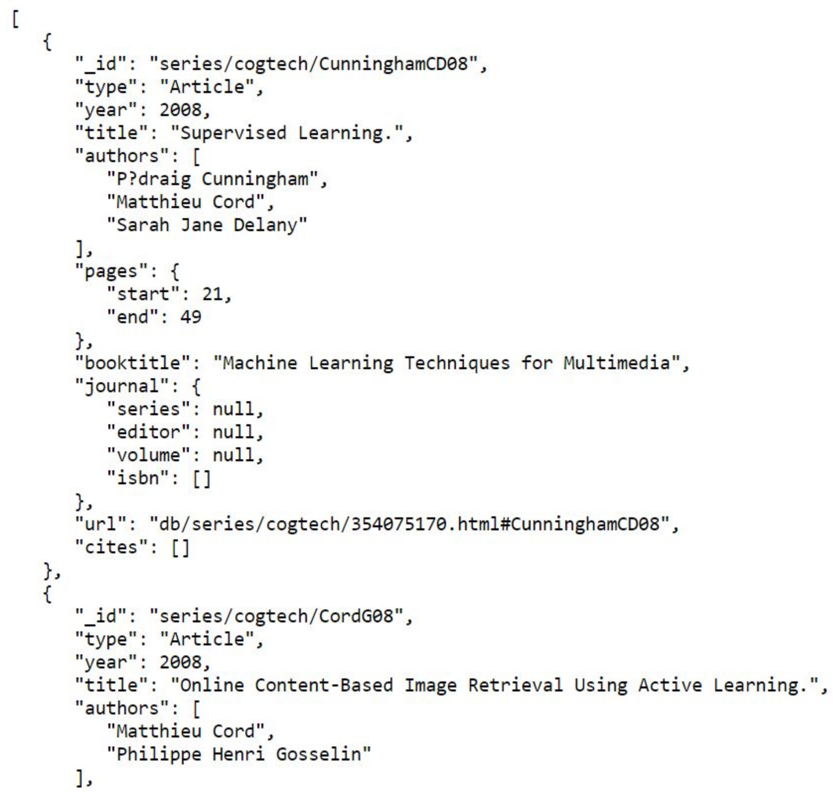
In the first text area, the administrator have to precise the first part of the query ; for example the json part with the element he wants to make a research on. In the second text area, he can precise in with order he wants his results to be displayed.

Here the code related to this query (in the dblpController.js file):



```
exports.launchQuery7 = function (req, res) {
    var query = req.query.createQuery;
    var order = req.query.orderBy;

    try {
        var finalQuery = [];
        finalQuery.push(JSON.parse(query));
        if (order != "") {
            finalQuery.push(JSON.parse(order));
        }
    }
    catch (err) {
        errorHandler.error(res, err.message, "Bad query");
    }

    try {
        MongoClient.connect(url, function (err, client) {
            var db = client.db('dblp');
            db.collection('publis').find(finalQuery[0], finalQuery[1]).toArray()
                .then(function (dblps) {
                    res.setHeader('Content-Type', 'application/json');
                    res.send(JSON.stringify(dblps, null, 3));
                }).catch(function (err) {
                    errorHandler.error(res, err.message, "Failed to get publications");
                });
        });
    }
    catch (err) {
        errorHandler.error(res, err.message, "Failed to connect to the database");
    }
}
```

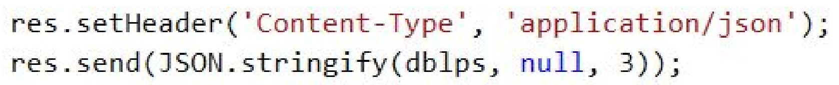
And here it’s what we obtained after entered a query (for this example we used the examples showed on the website):



```
[
   {
      "_id": "series/cogtech/CunninghamCD08",
      "type": "Article",
      "year": 2008,
      "title": "Supervised Learning.",
      "authors": [
         "P?draig Cunningham",
         "Matthieu Cord",
         "Sarah Jane Delany"
      ],
      "pages": {
         "start": 21,
         "end": 49
      },
      "booktitle": "Machine Learning Techniques for Multimedia",
      "journal": {
         "series": null,
         "editor": null,
         "volume": null,
         "isbn": []
      },
      "url": "db/series/cogtech/354075170.html#CunninghamCD08",
      "cites": []
   },
   {
      "_id": "series/cogtech/CordG08",
      "type": "Article",
      "year": 2008,
      "title": "Online Content-Based Image Retrieval Using Active Learning.",
      "authors": [
         "Matthieu Cord",
         "Philippe Henri Gosselin"
      ],
```

How we displayed our queries results

To displayed our result in json, we used these lines to prettyfing our result.



```
res.setHeader('Content-Type', 'application/json');
res.send(JSON.stringify(dblps, null, 3));
```

Difficulties

- At first we used a model with mongoose. We implemented a whole model in a file named “dblpModel.js”. We figured out after that this wasn’t useful for our application because the computer understood that we wanted to use a independant database with this new model and we couldn’t find our data. So we used MongoClient to recover data in our database.

- We had some trouble using MongoClient. At first, we had a problem with the name of the database. Before it was “DBLP” with capital letters and it didn’t want to connect us to the database. We changed to “dblp” and then our problem was solved.

- The second problem we had with MongoClient was a problem a version. Before it was : *MongoClient.connect(url, function (err, db) {...}*. Nowadays, with the latest update it’s : *MongoClient.connect(url, function (err, client) {...}*. Without this changement we can’t access to our database.

- We had to think about the organisation of our application. We didn’t want to put our whole code in a unique file. We found pretty fast the solution we wanted which is to have in separate files our routes, our methods, etc.

- For the login and the password, at first we wanted to use a database with encrypted passwords. We changed our mind and we used Javascript. This method isn’t at all a safe method but for our project we thought that was acceptable for now. As an improvement, the first thing we will do will definitely be to secure our application with a safer authentication.