

Portfolio Part 2: MongoDB

Aoife Sayers

Advanced Database Programming

Due 10th November 2017

Contents

[Introduction 3](#_Toc496920204)

[1. Implement your own MongoDB database for a possible big data application (showing examples of CRUD) (40%) (JSON document) 4](#_Toc496920205)

[RESTful Interface CRUD - Create collection & insert 4](#_Toc496920206)

[RESTful Interface CRUD – Read 5](#_Toc496920207)

[RESTful Interface CRUD – Update 7](#_Toc496920208)

[RESTful Interface CRUD – Delete 8](#_Toc496920209)

[2. Evaluate and Download a driver for a programming language of your choice. Build a UIF in Java/C# etc and show the CRUD operations working through the UIF. Show a call to a mapreduce operation from within the programming language. Comment your code and show screen pictures of it working. (60%) 9](#_Toc496920210)

[NodeJS 9](#_Toc496920211)

[Create UIF 14](#_Toc496920212)

[Read UIF 16](#_Toc496920213)

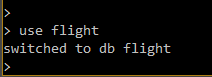
# Introduction

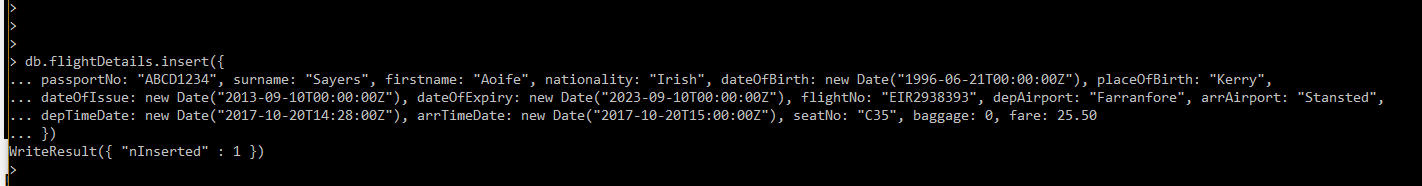
MongoDB is free and open-source NoSQL database particularly good for dealing with Big data. MongoDB stores data in JSON documents (technically BSON), meaning fields can vary from document to document and data structure can be changed over time. MongoDB is a very readable and comprehensible database to learn for a programmer as it’s functions are in a JavaScript format. The flexible schema-less structure allows or faster, iterative development with regards scope creep. MongoDB allows Ad hoc querying and indexing for analyzing data. MongoDB is high availability, has horizontal scaling, and geographic distribution using GridFS are built in and these functions are easy to use. (MongoDB, 2017)

Explain vision behind database design and explain why couchdb/mongo a good fit

# 1. Implement your own MongoDB database for a possible big data application (showing examples of CRUD) (40%) (JSON document)

### RESTful Interface CRUD - Create collection & insert

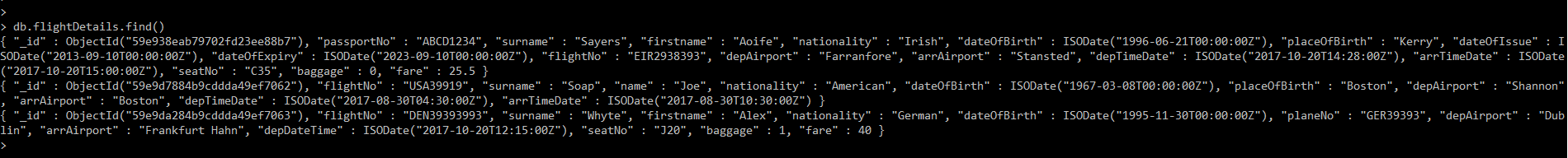


I created a collection called flightDetails and inserted one document via the command line interface. I started the MongoDB server on the path C:\Program Files\MongoDB\Server\3.4\bin>mongod. I opened another CMD prompt on the path C:\Program Files\MongoDB\Server\3.4\bin>mongo to access the CLI. I then issued the command in the figure below to create the database and insert a document to the flightDetails collection. I chose to model a flight booking systems as….

*Figure 1: Creating a database called flightDetails and inserting a document to the flightDetails collection*

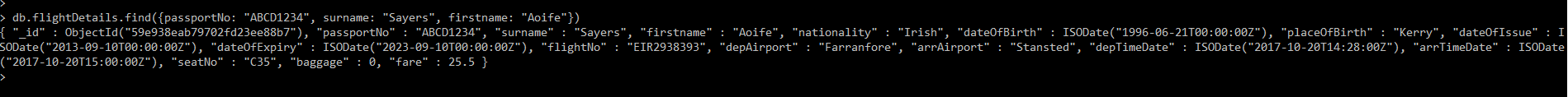
### RESTful Interface CRUD – Read

Finding all documents in the collection using db.flightDetails.find()

I currently have 3 documents in flightDetails

*Figure 2.0: Reading all documents from the flightDetails collection with the find() method with no parameters*

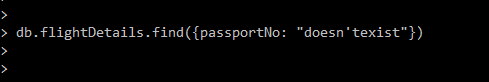
Finding a specific document *db.flightDetails.find({key, value})*

I want to find the document with specific details of passportNo ABCD1234, surname Sayers and first name Aoife

*Figure 2.1: Reading specific documents from the flightDetails collection with the find() method with specified key and value parameters*

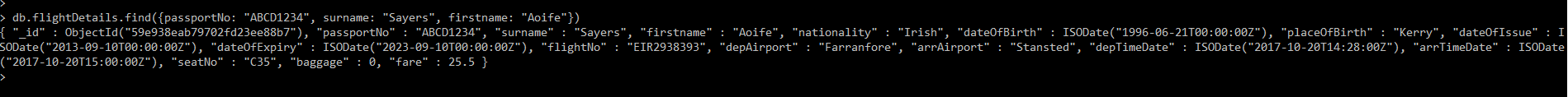
Finding a document with no relevant fields in the collection

As expected it returns no document



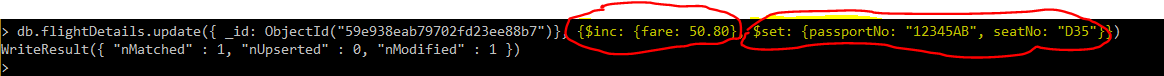
*Figure 2.1: Reading a document with specified key value fields that does not exist in the flightDetails collection*

### RESTful Interface CRUD – Update

Updating details from this

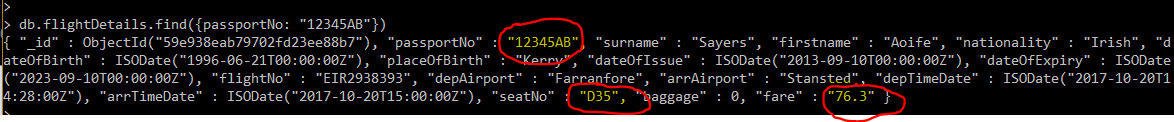
*Figure 3.0: Original details not updated*

Updating using $set & $inc

I updated the details using the db.*collectionName.*update() method passing the ObjectID and $set and $inc fields as the fields to be updated. The $set updates the specified field names of the document and the $inc adds the updated value onto the existing value in the database.

*Figure 3.1: Updating fields using $set and $inc using the specified ObjectID*

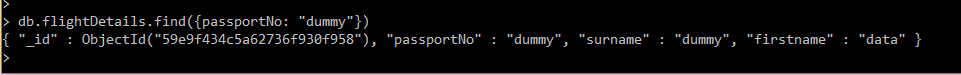
Updated details



*Figure 3.2 Result of updating details*

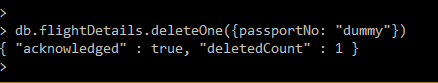
### RESTful Interface CRUD – Delete

Deleting this data



*Figure 4.0: document to be deleted*

Deleting with passportNo dummy



*Figure 4.1: Deleting a document with a passportNo of “dummy”*

Proof

Proof the document with passportNo dummy is deleted



*Figure 4.2: Document with a passportNo of “dummy” no longer exists within the flightDetails collection*

# 2. Evaluate and Download a driver for a programming language of your choice. Build a UIF in Java/C# etc and show the CRUD operations working through the UIF. Show a call to a mapreduce operation from within the programming language. Comment your code and show screen pictures of it working. (60%)

# NodeJS

The driver I started out originally was NodeJS.

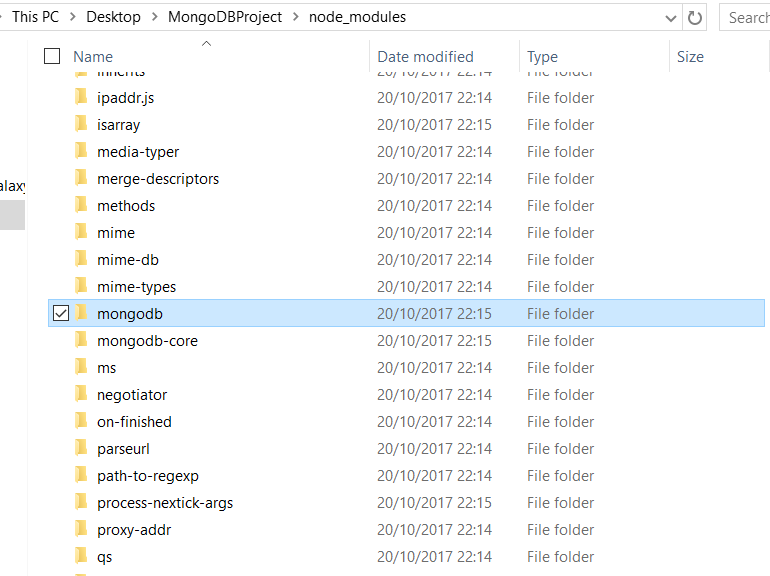
Node.js is an open source server framework that allows you to run JavaScript on the server. I downloaded the MongoDB driver for NodeJS via npm (Node Package Manager).

To start I created a *MongoDBProject* directory. On the command line I changed directory to the *MongoDBProject* directory. I issued the command npm init to create the package.json file. It serves as documentation for what packages your project depends on – what drivers and their versions are used etc. Then I installed the mongodb driver. The mongodb driver is saved into the node\_modules folder in the root of your project.

cd MongoDBProject

npm init

npm install mongodb --save



To learn more about Node I completed Create or Insert without a UI

*//Insert*

**var** MongoClient = require('mongodb').MongoClient;

*// Connect to the db*

MongoClient.connect("mongodb://localhost:27017/exampleDb", **function**(err, db) {

**if**(err) { **return** console.dir(err); }

**var** collection = db.collection('test');

**var** doc1 = {'hello':'doc1'};

**var** doc2 = {'hello':'doc2'};

**var** lotsOfDocs = [{'hello':'doc3'}, {'hello':'doc4'}];

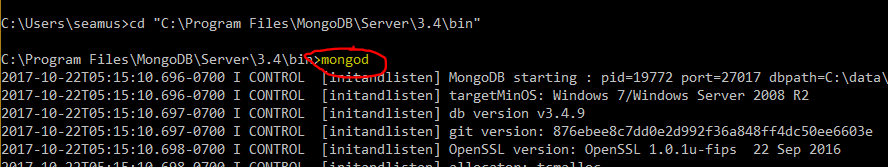
collection.insert(doc1);

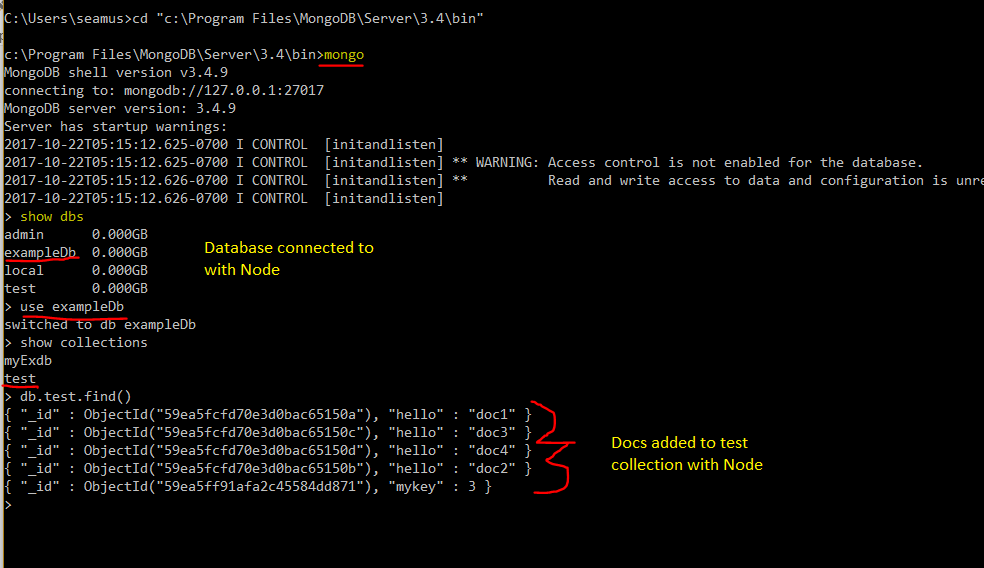
collection.insert(doc2, {w:1}, **function**(err, result) {});

collection.insert(lotsOfDocs, {w:1}, **function**(err, result) {});

});

Starting MongoDB



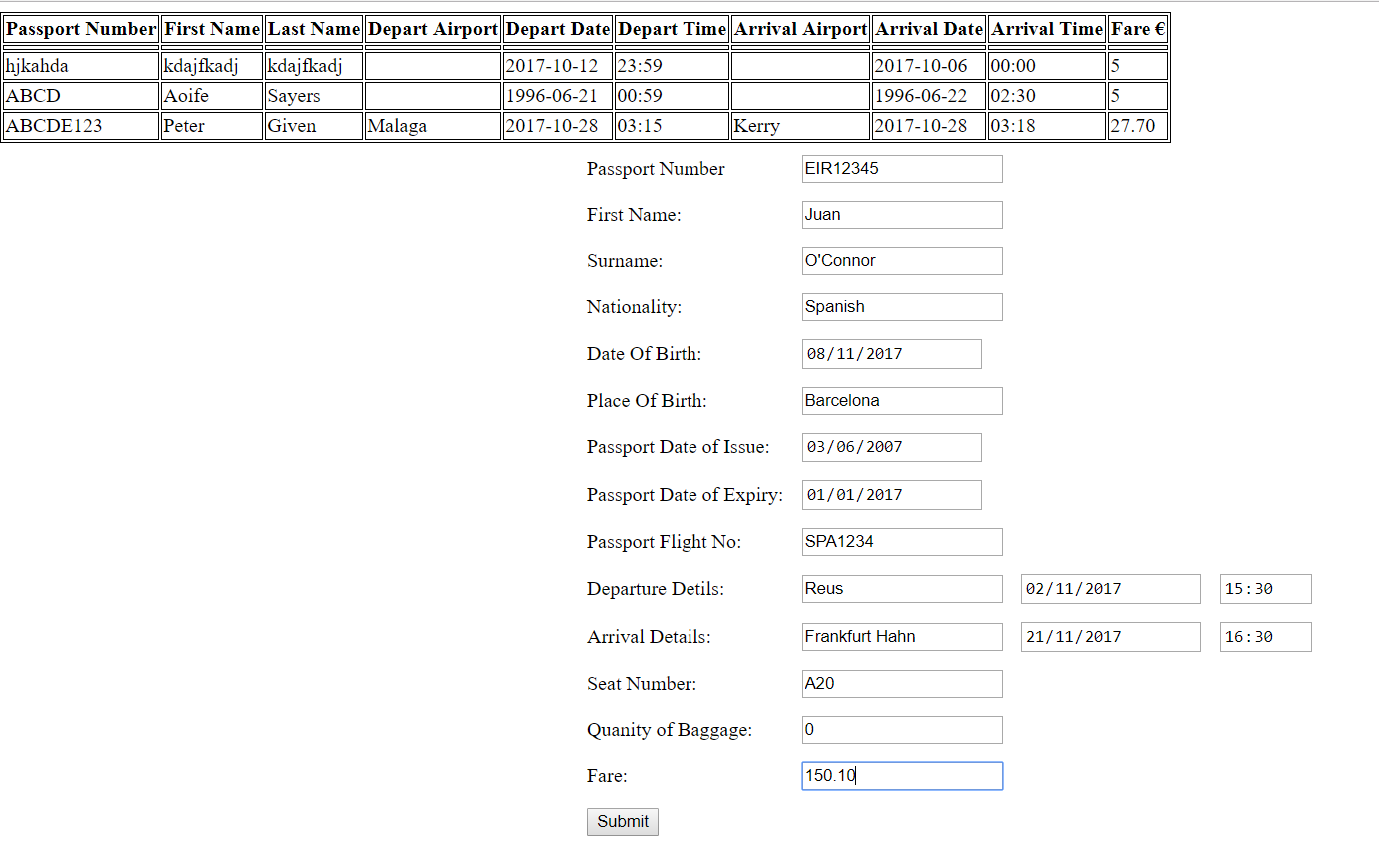


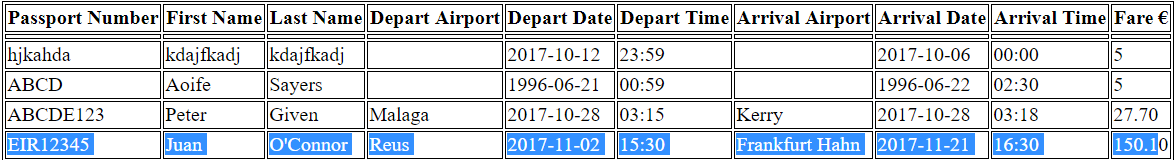
Node Modules

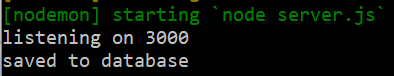
Equivalent to the packages in Java

* Mongodb
* Express
* Body-parser

## Create UIF









Server.js code snippet

res.render('index.ejs', {flightDetails: result}) – takes the name value from the html form in index.ejs.

For Example:

<input type="text" name="passportNo" required> - value in json is passportNo

Passport number and other entries are passed into the method and inserted into the flightDetails collection. On successful insertion, a console.log is printed out and the res.redirect(‘/’) reloads the page

app.post('/flightDetails', (req, res) => {

db.collection('flightDetails').save(req.body, (err, result) => {

**if** (err) **return** console.log(err)

console.log('saved to database')

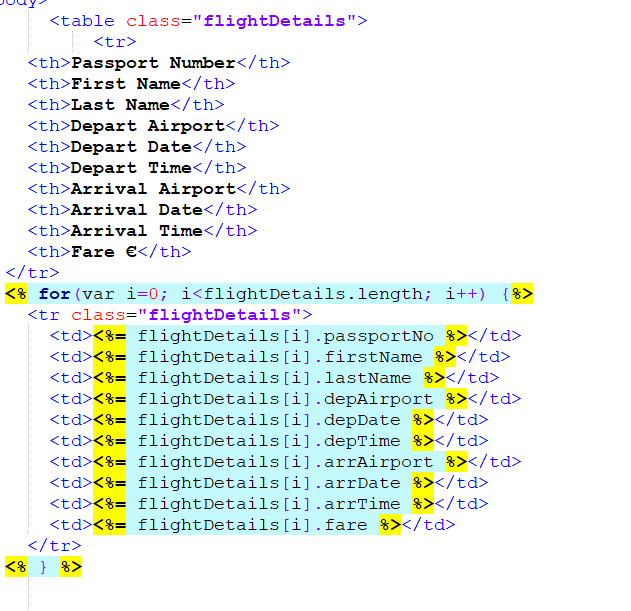
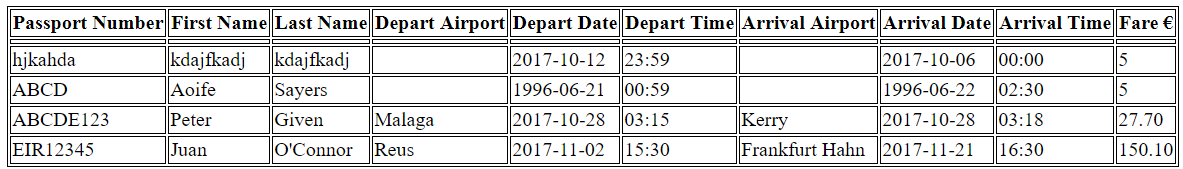
res.redirect('/')

})

})

## Read UIF

Read selected attributes of the JSON document into a HTML table with NodeJS and Express



References

<https://zellwk.com/blog/crud-express-mongodb/>