MongoDB

Emergency Room App

Gary O Brien – T00181299

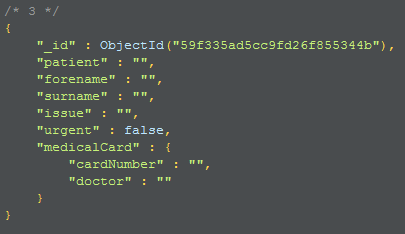
##### INTRO TO mongoDB

INTRO

MongoDB is an easy to use document based database with emphasis on the ability to scale horizontally. In the CAP theorem it has data consistency and partition tolerance. One of the core ideas behind MongoDB is the ability to access data quickly and reliably. MongoDB achieves this with the use of sharding and replica sets. For this project I have chosen to do an application based on the waiting room of a hospital. The software will demonstrate CRUD operations through the manipulation of a patient’s information.

JSON

For the sake of this project I have modelled my JSON object based on essential patient details. This looks like the following.



Patient – The patients ID

Forename & Surname – Name of the patient in question

Issue – The reason the patient is in the emergency room

Urgent – Boolean field denoting whether the patient is in urgent need of attention or not

MedicalCard – A nested JSON object holding fields related to a medical card

CardNumber – The ID of the medical card if available

Doctor – The name of the doctor labelled on the medical card

WHY IM USING mongoDB

I have chosen to use MongoDB for this project as when dealing with patients in a hospital, large volumes of data needs to be gathered and stored. MongoDB specialises in the controlling of a large dataset. This project is not a big data project, but all the elements of MongoDB explored here would carry over into a big data project.

##### CRUD Using RObo 3t

robo 3t summary

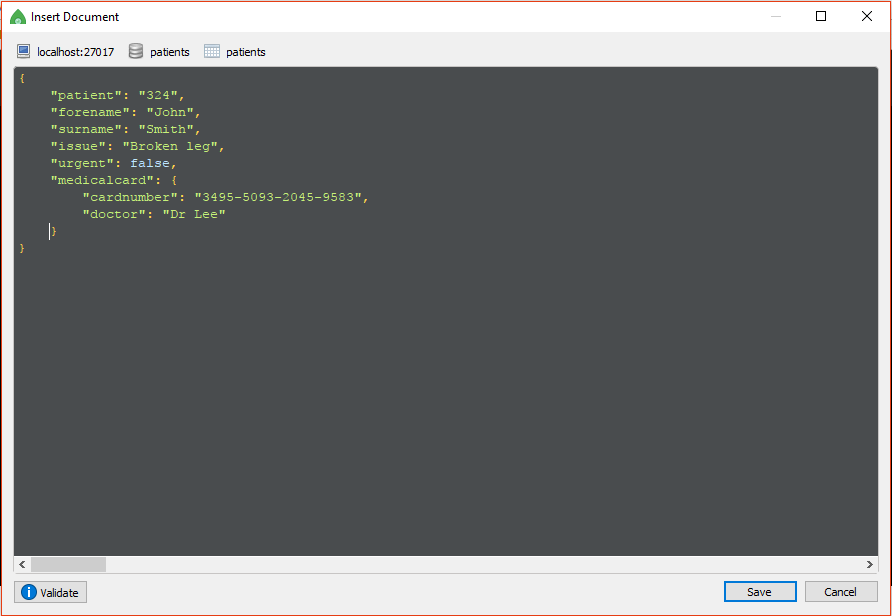
After researching many applications that abstract CRUD functionalities for MongoDB, I found Robo 3T. This GUI lays out collections in the Mongo database as if it were a table in a relational database. Right clicking on the collection allows you to perform basic CRUD operations. The software comes with a feature that allows you to view your documents in the form of a table or as raw JSON. With Robo 3T, you can choose to run command from the built-in command line interface. Using the find() command is especially useful for this if you wish to return specific documents. This command line also has a predictive text to finish your input. I chose to use Robo 3T as most of the other MongoDB abstractions did not have the full features of this GUI and most of the other one’s worth looking at required a payment.

Prerequisites

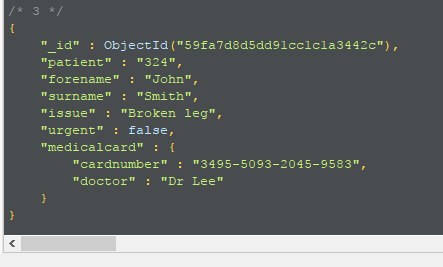
Before setting up a connection to the MongoDB server, you must create a server on the command line. I did this by navigating to the mongodb folder and running the command “mongod”. Then from a separate command line I navigated to the same folder and entered the command “mongo patients”. For this project, my main database and collection will be patients.

Create

To create a new document in Robo 3T, you have to right click your collection from the hierarchy on the left side. After selecting “insert document… ” you are prompted with a screen where you have to enter your JSON. Below I have entered the template JSON I used for my GUI application.

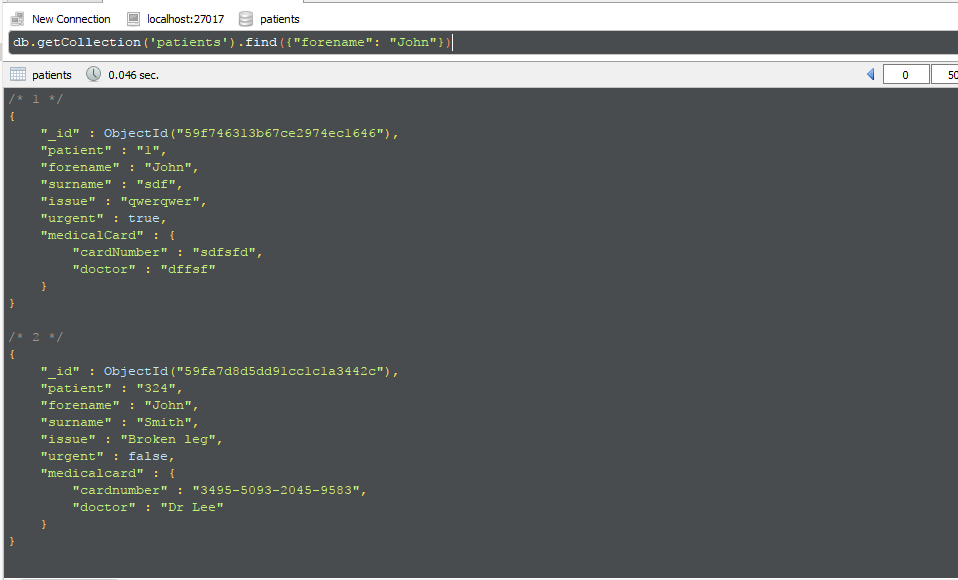


After pressing save, the document can be viewed under the “patients” collection. An Object id has been generated for the new document.



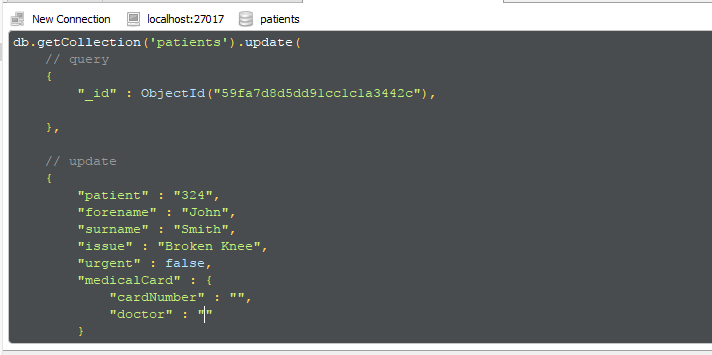
READ

Reading from Robo 3T is as simple as clicking on the collection under the hierarchy. From here we can see all the document inside the collection. The command line interface can be used to search for specific documents if any detail related to the JSON is known. As shown below I have used the find() function to return all document with the patient forename of “John”.

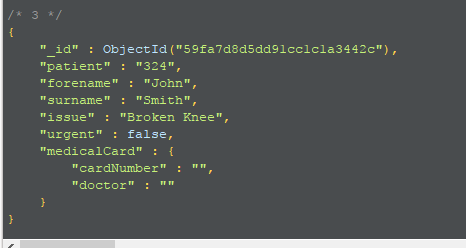


UPDATE

To update a document you must select the “Update document… ”. This brings you to a separate window where it asks you to enter the Id of the document you wish to update and the JSON that will replace the document. Below this you can also enable whether you wish to update many and, if the document doesn’t exist already, create a new one.

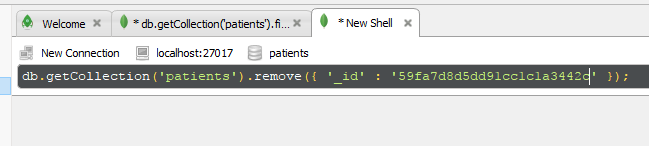
****

In the full collection view.



DELETE

Deleting a document is a simple procedure with Robo 3T. You can perform the command-line operation “delete()” or select the “Delete document.. ” function from the collections right click menu. Also with this you can delete all the documents in the current collection.



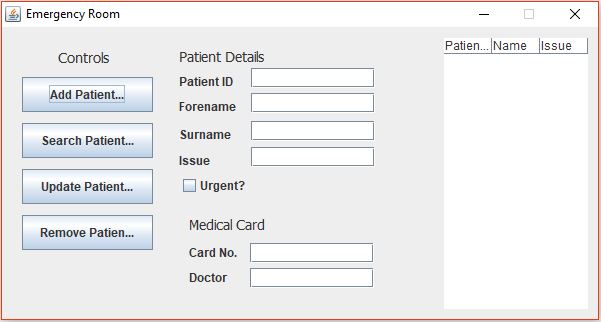
##### **CRUD USING Java driver**

INTRO TO Java Driver

The first driver that I researched for use in this project was the official .net MongoDB driver. The documentation was vast enough to merit use of the driver but the last update was five months ago. I left this as a last option if nothing else was worth looking into. Web searches brought up many Python and JavaScript drivers, which are of no use to me as I do not know these languages. For this reason I decided to investigate Java drivers. The first search result brought up a link to the official site of MongoDB describing a supported Java driver. I found this to be a good fit over the other choices I had as it was constantly updated with support from the MongoDB developers. A web search for Stack overflow questions related to this driver brought back a full page of results. I decided I would use the Java driver due to the heavy documentation/support and, on a more personal level, I want to diversify my Github projects so I don’t only use the .net framework for projects.

driver GUI

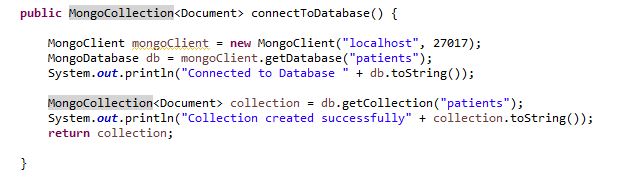
For this project I have chosen to model a system based on an emergency room, with a constant influx of patient data needing to be recorded. There are three main components to my GUI. A controls section, a section holding specific patient data and a table view of all the patients in the database.



For this application to work the commands to set up a MongoDB server and Collection called “patients” must be established just as above in the case of Robo 3T. Once this is set up it will automatically fill the JTable with all the patients in the local MongoDB database. Now you can perform the CRUD functions on the left side of the application. Every time an action is performed a call to a “fillTable()” method in the code will reset the contents of the table without needing to reload.

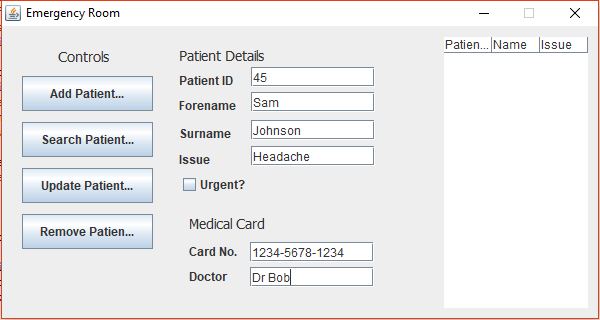


The code for “connectToDatabase()” is called on every function. The code for this method is seen below.

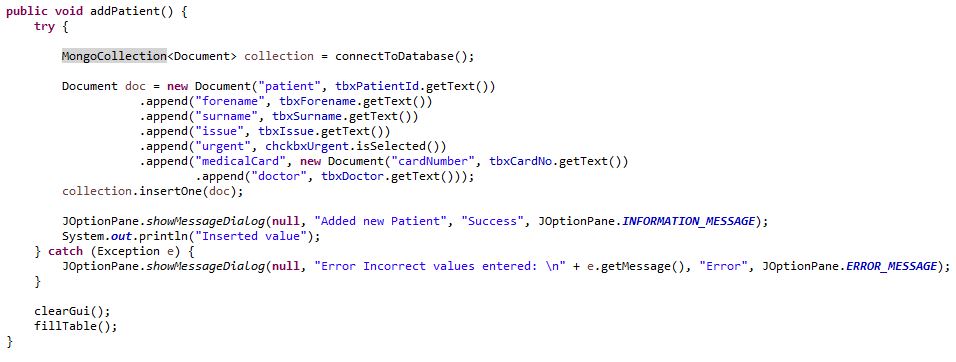


CREATE

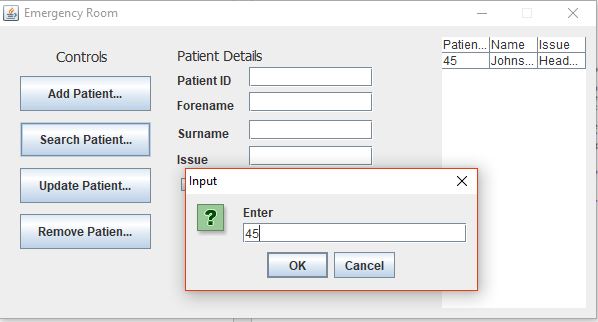
Selecting the “Add Patient… ” button under the controls section of my GUI will add the current details in the textboxes into the JSON object which is detailed at the beginning of this report.



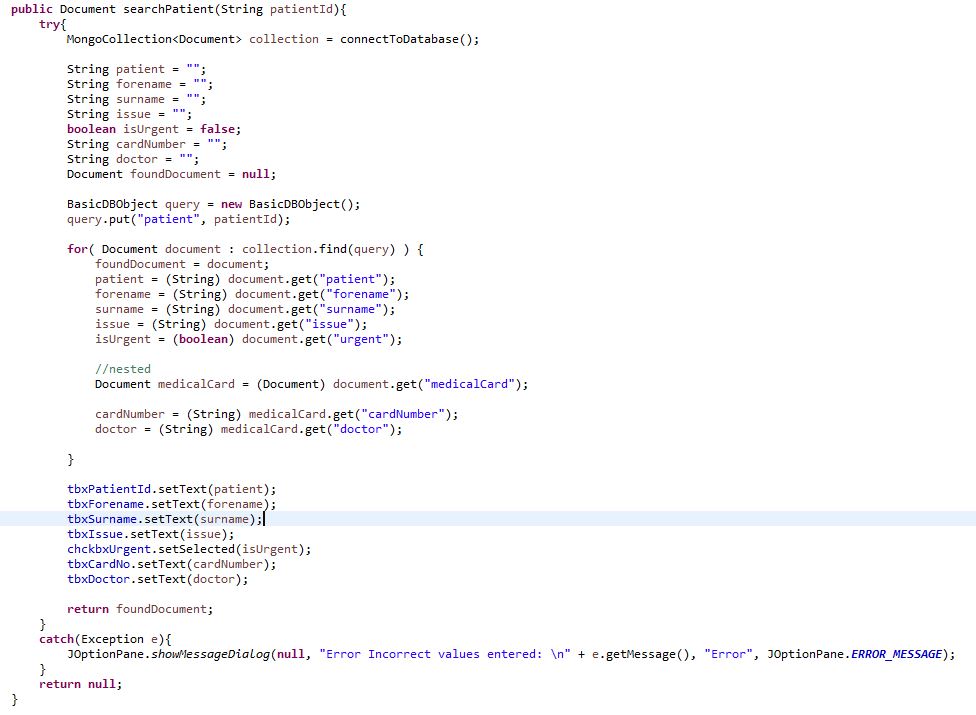
The details are passed into a Document object which is specific to the MongoDB driver. The methods for calling CRUD functions are very similar to the actual commands that would be called from the command line. In this instance the line of code that is called is “collection.insertOne()”, with the Document as a parameter. When calling methods related to MongoDB, they must be wrapped in a try-catch.

****

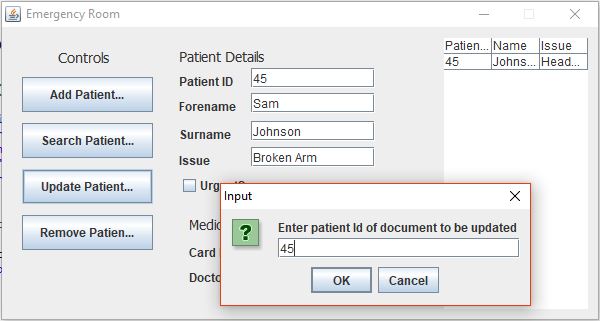
READ

To perform a read on my GUI application you must select “Search Patient… ”. This will that bring up a prompt to enter the related patients details. 

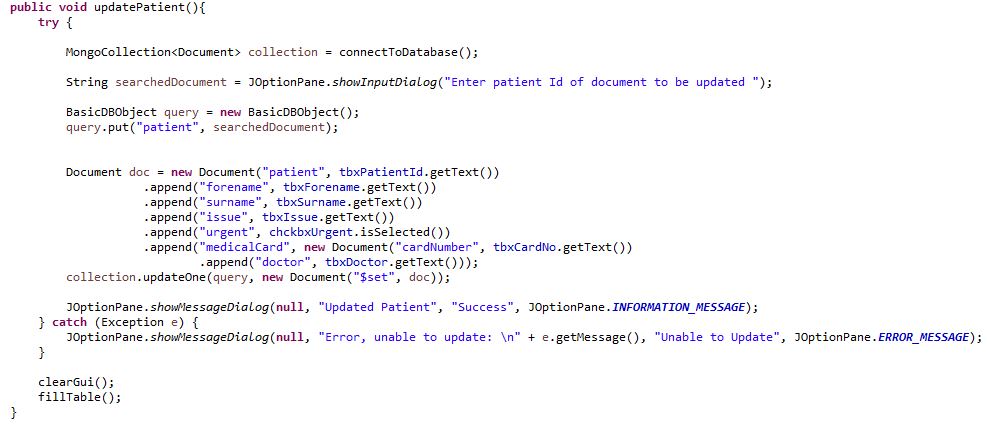
My “searchPatient()” method takes one String parameter, which is the Id of the patient you want to return. After connecting to the database a DBObject is created. This is used to pass in a query to the database. After the query to return a specific value is returned it the cycles through the answers using a foreach loop. This makes sure that the most recent value entered for that specific Id is returned. After returning all the values they are set to the text in all the textboxes. Just as before the code is wrapped in a try-catch to prevent runtime errors from occurring.



UPDATE

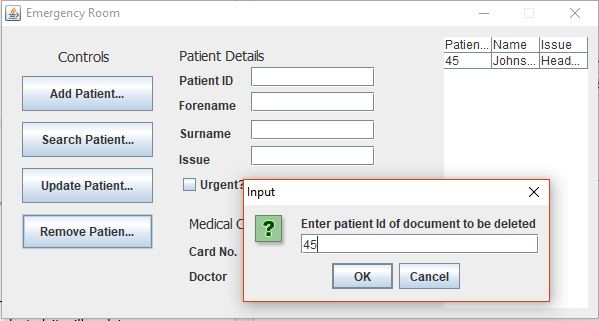
To update a document from the GUI, select the “Update Patient…” control. This will prompt the user for the Id of the document to be updated. Once you input the patient id and select ok it will update that specific patient with the details that are in the text boxes. 

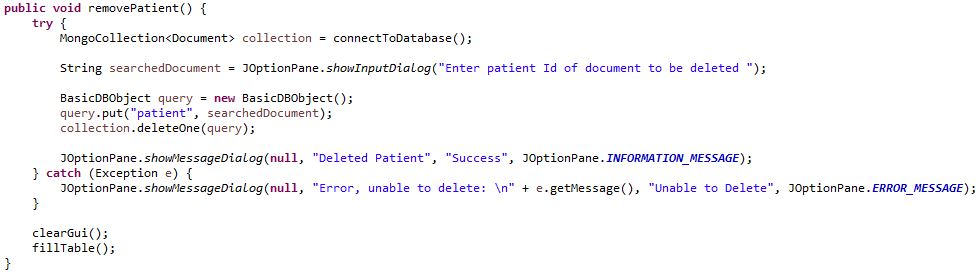
The code for updating a patient is very similar to the create function. With the update instead of a call to “insertOne()”, there is a built in method in the MongoDB driver called “updateOne()”.



DELETE

The final functionality the MongoDB GUI has is the ability to delete a patient once their details are no longer need or if the data was inputted incorrectly. To delete a document you must input the Id of the document you wish to delete. This then removes them from the MongoDB collection



As before, we use the DBObject to perform a query on a specific database entry. We pass the query to the method “collection.deleteOne(query)”. 

##### **CONCLUSION**

**ADVANTAGES**

* As it is supported by the official MongoDB developers, there is a lot of documentation. This makes the learning process a lot easier with this driver.
* The structure of the methods makes it very easy to predict what the command would be to do a specific action. All the commands follow the same layout as the commands that would be used on a command prompt.
* Java dependencies are very easy to install, as simple as adding a .jar file to the project structure.

**DISADVANTAGES**

* Writing map-reduce functions require you to type a JavaScript function as a String. This can cause errors which won’t be picked up by the compiler.

CONCLUSION

In conclusion, I found MongoDB as a whole very easy to work with. This is due to the fact that MongoDB is one of the more popular databases, especially document based databases. Robo 3T is a very useful tool for visualising your data. Sometimes when working from the command line it can be difficult to see all of your JSON documents with all the clutter. I found the Java driver for MongoDB very easy to use. I found all of the information I needed from the official driver documentation so very little external research needed to be conducted. I had issues trying to implement a Map-reduce as there was not too much complexity to my system with regards big data. Every time I tried to run my map-reduce it wold return a null document, even if conducted from the command line. Overall I would not change much from what I conducted for this project and have furthered my understanding of the background of MongoDB.