

Portfolio Part 1: CouchDB

Aoife Sayers

Advanced Database Programming

Due 20th October 2017

Contents

[Introduction 3](#_Toc496115024)

[1. Implement your own CouchDB database (something requiring both a desktop/mobile app or mobile game implementation) and use the REST interface to show examples of CRUD. 40% 4](#_Toc496115025)

[CouchDB Database 4](#_Toc496115026)

[RESTful Interface CRUD – READ 5](#_Toc496115027)

[RESTful Interface CRUD – CREATE 6](#_Toc496115028)

[RESTful Interface CRUD – UPDATE 7](#_Toc496115029)

[RESTful Interface CRUD – DELETE 8](#_Toc496115030)

[2. Download a driver for a programming language of your choice (Android would be nice!). 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%) 1](#_Toc496115031)

[Ektorp 1](#_Toc496115032)

[Java UIF CRUD - CREATE 2](#_Toc496115033)

[Java UIF CRUD - READ 4](#_Toc496115034)

[Java UIF CRUD – UPDATE 6](#_Toc496115035)

[Java UIF CRUD – DELETE 9](#_Toc496115036)

[Conclusion 11](#_Toc496115037)

[References 11](#_Toc496115038)

# Introduction

Developed by Apache in 2005, CouchDB is a scalable, NoSQL document oriented database storing key value maps in a JSON format. CouchDB can be implemented for a range of projects across many environments locally or over clusters of servers on mobile phones to web browsers on inconsistent networks with it’s offline first data sync. CouchDB is durable with regards data storage. Unlike MongoDB and Relational databases SQL, CouchDB does not permit ad hoc querying; instead CouchDB utilizes indexed views with MapReduce to find your documents. CouchDB can be accessed via the Fauxton web interface or using RESTful tools like CURL or Google Chrome’s web application Postman.

Also CouchDB has no major learning curve for those who have previously worked with Web applications. (Warner Onstine, 2012). Document data stores such as CouchDB and MongoDB have the least learning curve when migrating from relational databases to NoSQL databases. There is a much less rigid schema as well as avoiding having to join tables with CouchDB. (Whitehouse, 2009)

 CouchDB’s main features include:

Easy replication of a database across multiple server instances using the Couch Replication Protocol

Fast indexing and retrieval

HTTP REST based API for document insertion, updates, retrieval and deletion

JSON-based document format (easily translatable across different languages)

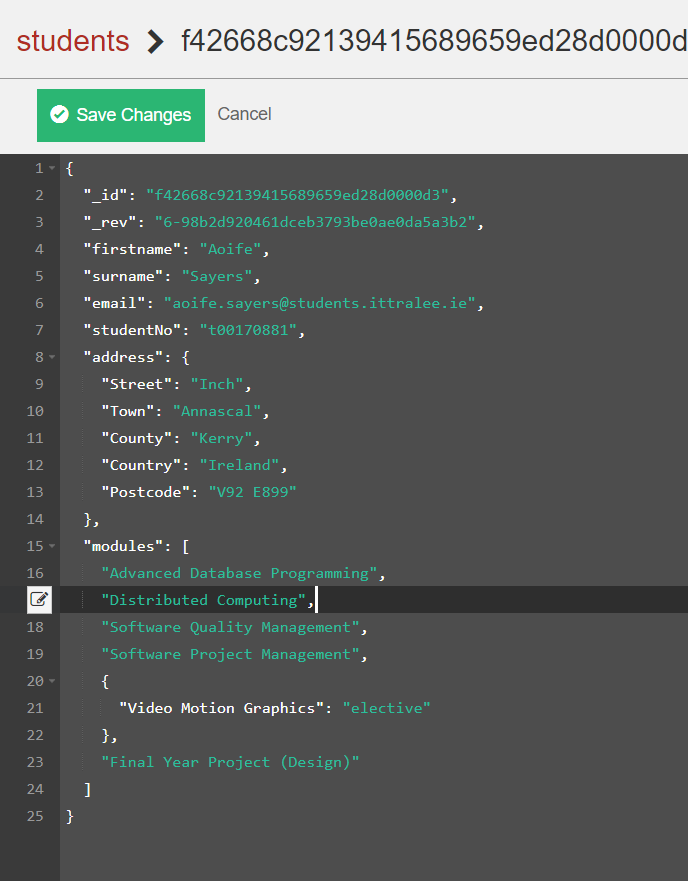
Multiple libraries for your language of your choice

MapReduce for simple, efficient, and comprehensive data retrieval. (Wasington, n.d.)

## 1. Implement your own CouchDB database (something requiring both a desktop/mobile app or mobile game implementation) and use the REST interface to show examples of CRUD. 40%

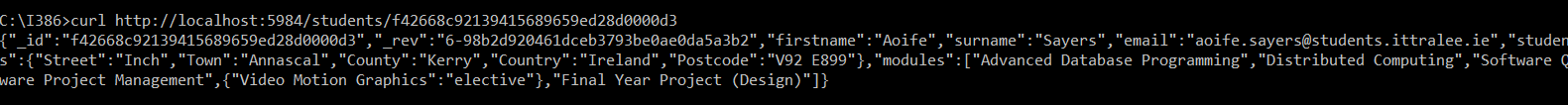
### CouchDB Database

* I created a document in the student database



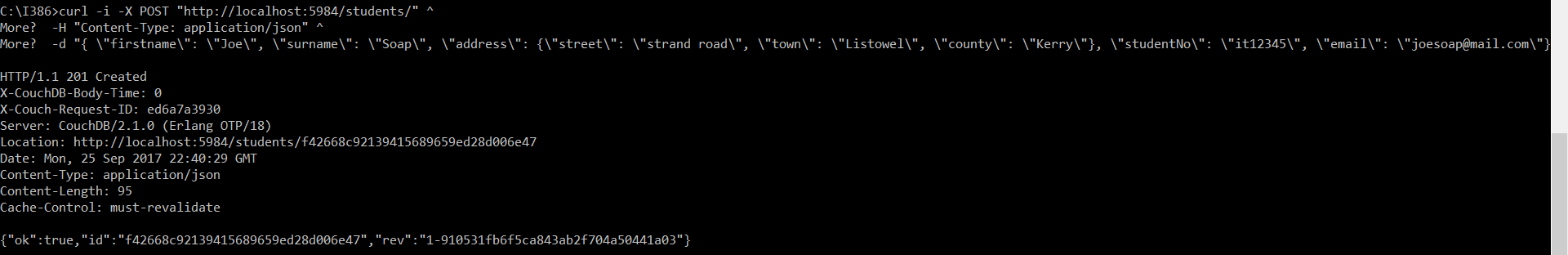
*Figure 1- Creating a document in the student database*

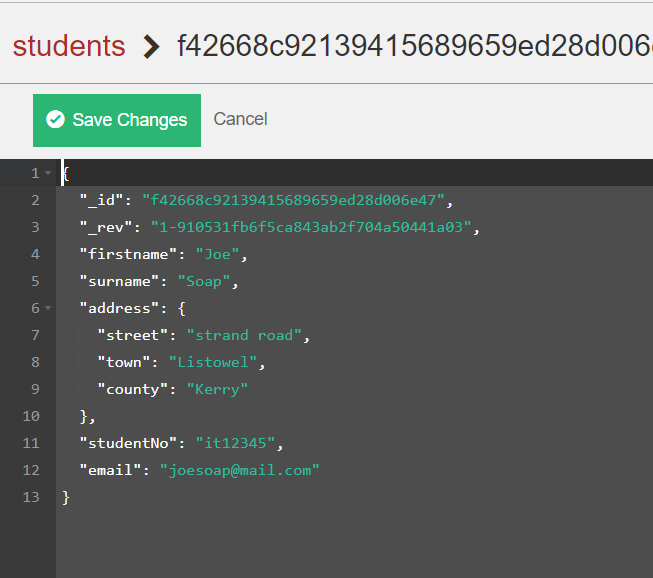
### RESTful Interface CRUD – READ

Issued a Curl get request to the student database using the ID as seen in the image above on the command line interface.

*Figure 2 – Reading/ Getting from database*

### RESTful Interface CRUD – CREATE

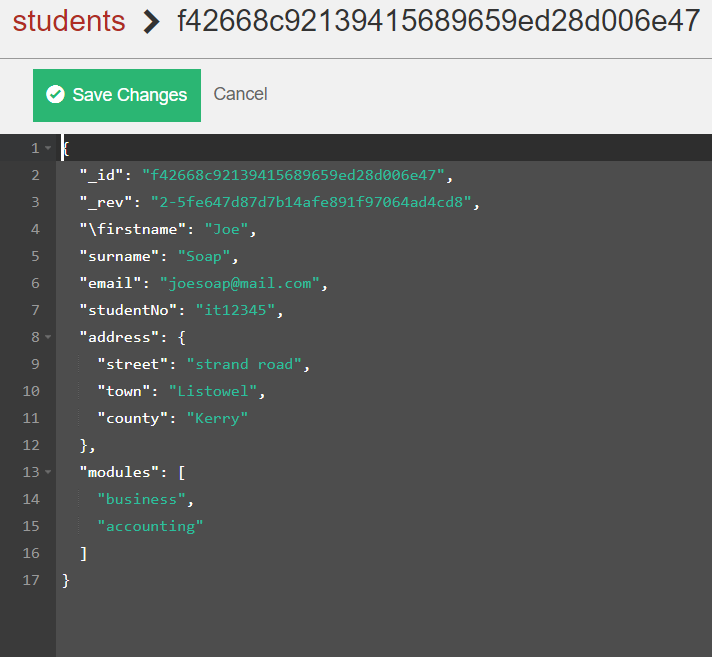
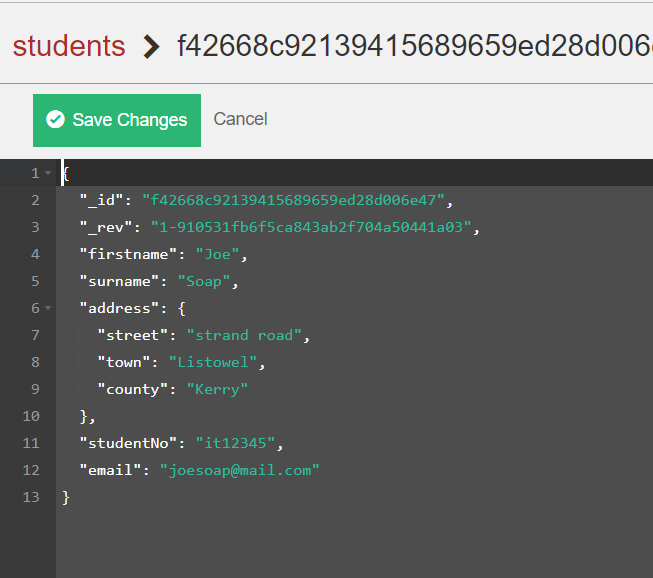
Created a new document to the student database via Curl. I used the post command along with the database URL, document of type JSON and each of the fields. The result below is o the insert on the Fauxton web interface.

*Figure 3 – Creating/Inserting to the database*

Note the difference in the schema to the figure 1 database

*Figure 3.1 – Result on Fauxton interface for Creating/Inserting to the database*

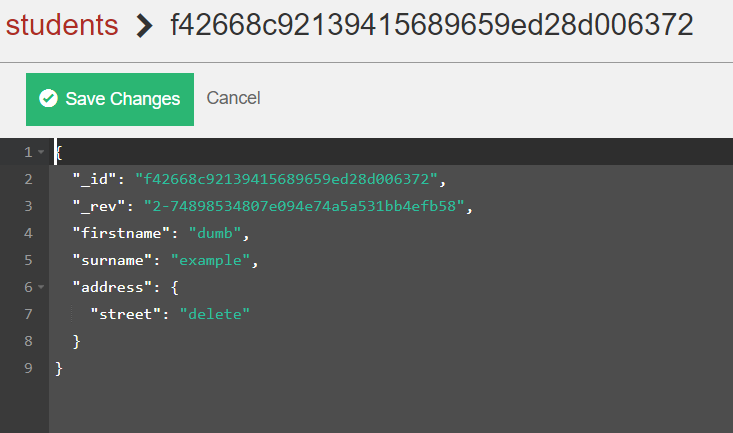
### RESTful Interface CRUD – UPDATE

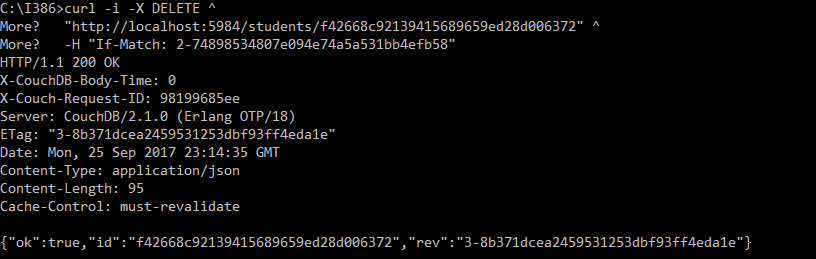
*Figure 4 - Updating documents in the database*

*Figure 4.1 – Updated document in the database Figure 4.2 – Document before it was updated*

### RESTful Interface CRUD – DELETE

*Figure 5 – Document to be deleted*



*Figure 5 – Deleting document via CURL CLI*

## 2. Download a driver for a programming language of your choice (Android would be nice!). 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%)

### Ektorp

I downloaded the Ektorp driver for Java. Ektorp is a persistence API that uses CouchDB as storage engine. (helun, n.d.). Ektorp uses the Java class Jackson for mapping Objects to JSON. Ektorp gives the programmer a choice of abstraction level from full object document mapping to raw streams. . (helun, n.d.) It is also very easy to do CRUD due to it’s simple API, detailed documentation, and active growing community. Ektorp has an optional spring support module. Ektorp is set up in a Java Maven project specifying the Ektorp dependencies.

|  |  |
| --- | --- |
| Advantages of Ektorp  Object Mapping  Ektorp gives the programmer a choice of abstraction level from full object document mapping to raw streams. . (helun, n.d.)  Easy to do CRUD  Sensical methods  Good Documentation  Spring Support  Good community support  JSON annotations for mapping | Disadvantages of Ektorp  As CouchDB is web oriented, Java may not be the best choice of language for the web  Difficult to map nested objects and lists in Ektorp for JSON  Java isn’t the most popular choice for working with CouchDB. I personally thing NodeJS or PHP would have been more suitable. In hindsight, this was the first project I programmed with a NoSQL database and chose Java as it was in my comfort zone…! |

### Java UIF CRUD - CREATE

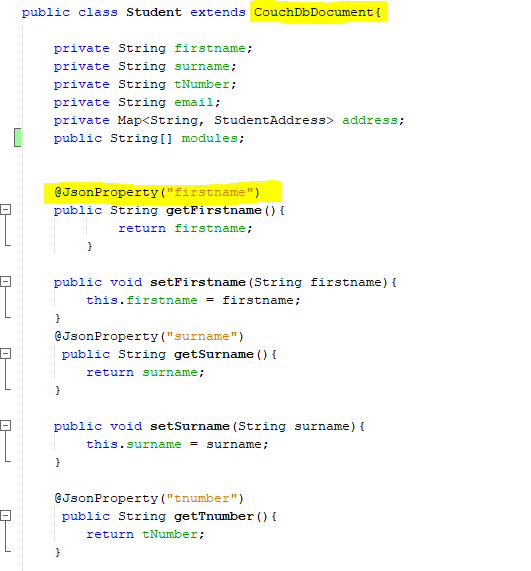
I completed the UI on Netbeans Swing JFrames

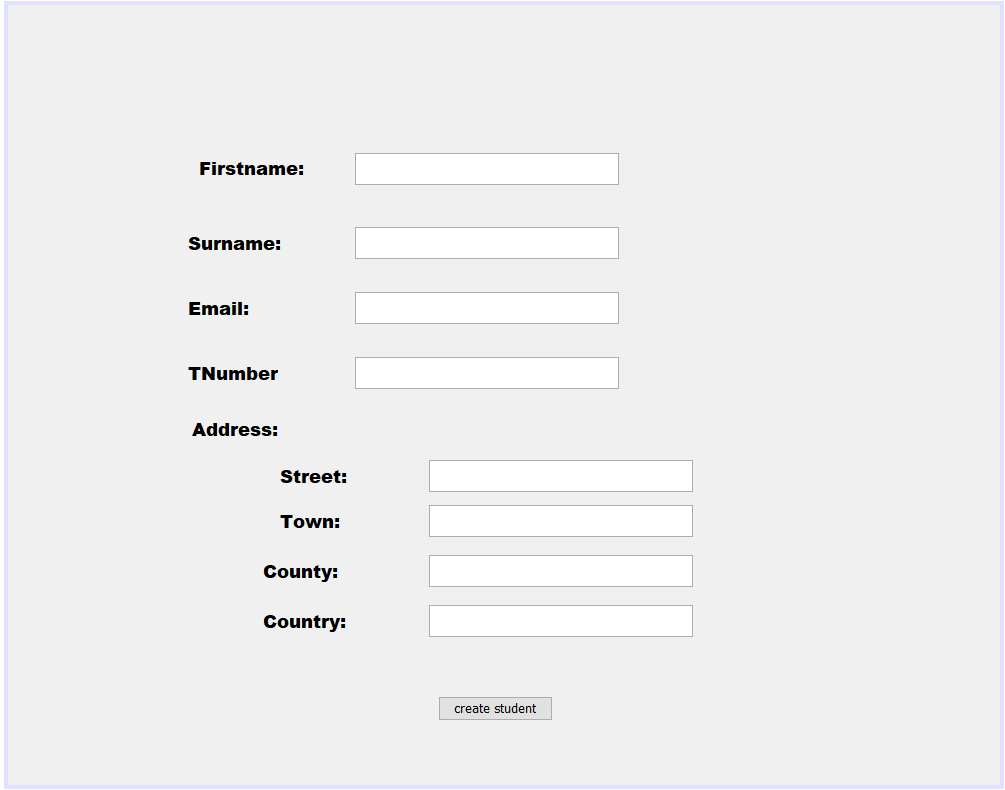
Link to full project on GitHub: <https://github.com/AoifeNicAntSaoir/EktorpJavaCouchDB>

**Student Object**

See the Jacksons Annotations for mapping to JSON *- @JsonProperty(“myProperty”)*

The StudentAddress Map attribute is for a nested JSON object e.g. {“firstname”: ”Aoife”, “surname”: “Sayers”, “tnumber”: “t00170881”, “address” : { “street”: “Strand street”, “town”: “Tralee”, “county”: “Kerry” }}





#### CREATE Code Snippet

//When the button is clicked

**private** **void** **btncreateActionPerformed**(java.awt.event.ActionEvent evt) {

HttpClient httpClient;

**try** {

//Connection details to localhost

httpClient = **new** StdHttpClient.Builder()

.url("http://localhost:5984")

.build();

//Connection to CouchDB server

CouchDbInstance dbInstance = **new** StdCouchDbInstance(httpClient);

//Connector to person database - creates database if it doesn't exist

CouchDbConnector db = dbInstance.createConnector("person", **true**);

//Creating a student object & setting the properties

Student s = **new** Student();

s.setFirstname(txtname.getText());

s.setSurname(txtsname.getText());

s.setEmail(txtemail.getText());

s.settNumber(txttnum.getText());

//For the nested Address Object

Map<String,StudentAddress> addr = **new** HashMap();

addr.put("address", **new** StudentAddress(txtStreet.getText(),txtTown.getText(), txtCounty.getText(), txtCountry.getText()));

s.setAddress(addr);

**try** {

//Creating/Inserting

db.create((s));

JOptionPane.showMessageDialog(**null**, "Student Created in the system", "Student created", JOptionPane.PLAIN\_MESSAGE);

//Clearing the UIF after insert successful

txtname.setText("");

txtsname.setText("");

txtemail.setText("");

txttnum.setText("");

txtStreet.setText("");

txtTown.setText("");

txtCounty.setText("");

txtCountry.setText("");

}

**catch**(Exception ex)

{

String exMessage = ex.getMessage();

JOptionPane.showMessageDialog(**null**, exMessage, "Error occured", JOptionPane.WARNING\_MESSAGE);

}

} **catch** (MalformedURLException ex) {

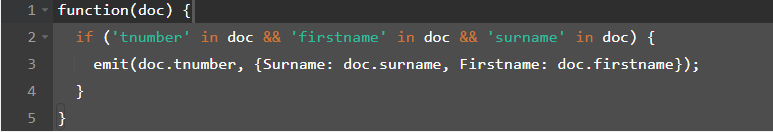
Logger.getLogger(CreateStudentGUI.class.getName()).log(Level.SEVERE, **null**, ex);

}

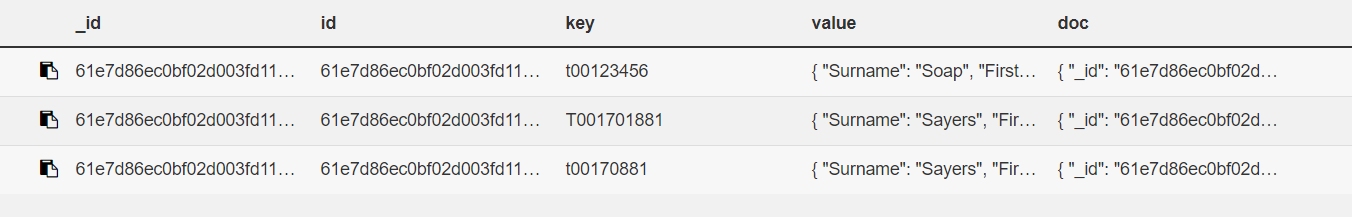
}

### Java UIF CRUD - READ

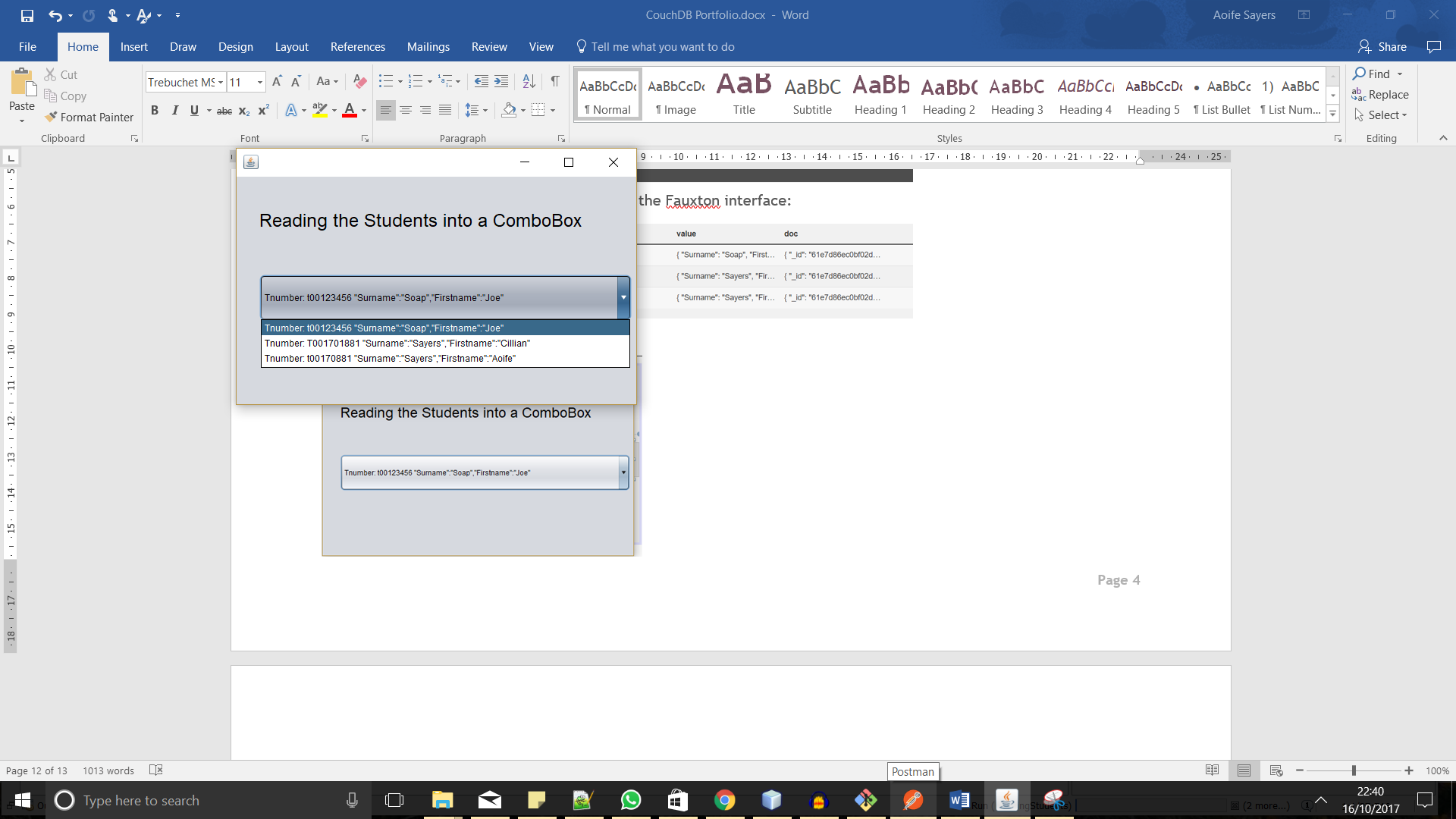
#### READ

I created a mapreduce view to display the Tnumber as a key and 2 values in a nested object- the first name and surname as a value.

The following shows the results of the view in the Fauxton interface:



#### READ UI



#### READ Code Snippet

**public** **class** **ReadingStudents** **extends** javax.swing.JFrame {

**public** **ReadingStudents**() **throws** IOException {

initComponents();

//Iterates through list of tnumbers and adds to combo box

**for** (String **s:** getdetails()) {

cboReadStudents.addItem(s);

}

}

//Calls the allTnumbers view query & places them in a list

**public** List < String > getdetails() **throws** MalformedURLException, IOException {

List < String > queryListDetails = **new** ArrayList < > ();

HttpClient httpClient = **new** StdHttpClient.Builder()

.url("http://localhost:5984")

.build();

CouchDbInstance dbInstance = **new** StdCouchDbInstance(httpClient);

CouchDbConnector db = dbInstance.createConnector("person", **true**);

ViewQuery query = **new** ViewQuery()

.designDocId("\_design/allTNumbers")

.viewName("allTnumbers");

ViewResult result = db.queryView(query);

**for** (ViewResult.Row **row:** result.getRows()) {

String keyValue = row.getKey();

String stringValue = row.getValue();

//formatting Value to remove JSON format

stringValue = stringValue.replace("{", " ");

stringValue = stringValue.replace("}", " ");

String queryDetails = "Tnumber: " + row.getKey() + stringValue;

queryListDetails.add(queryDetails);

}

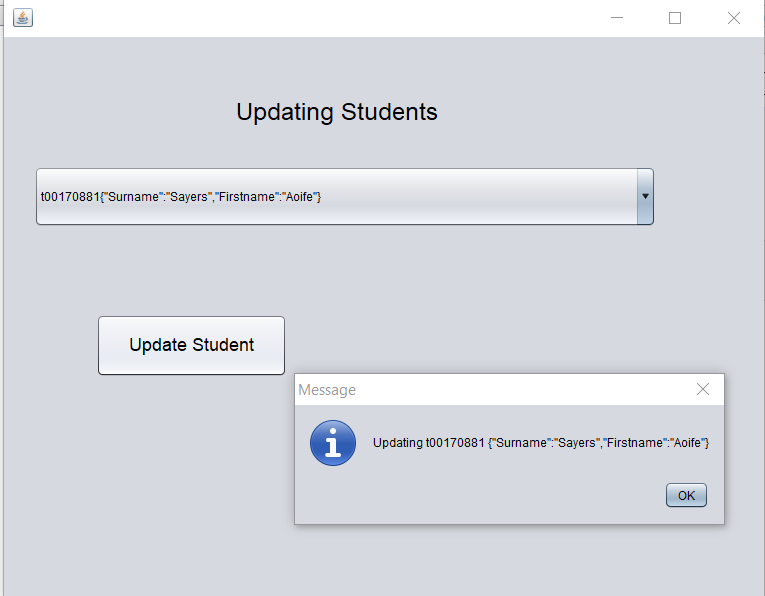
**return** queryListDetails;

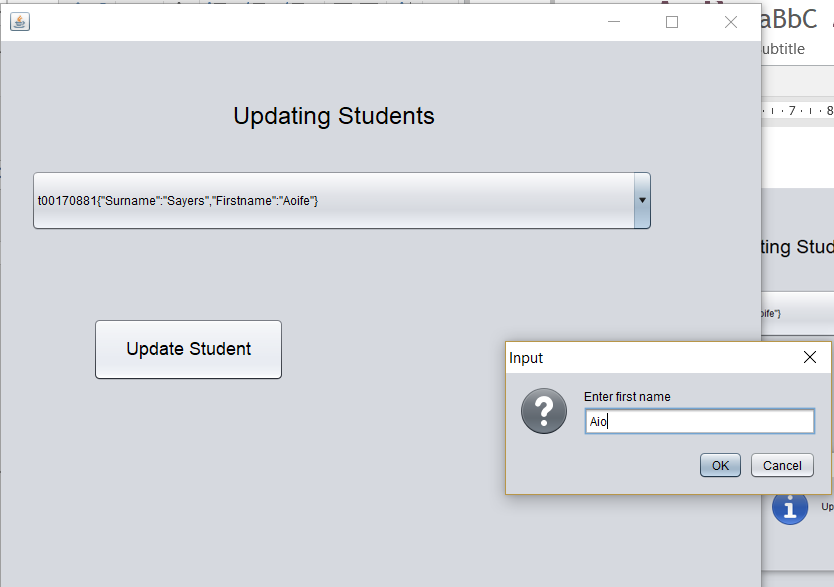
}

### Java UIF CRUD – UPDATE

#### UPDATE UI

The user selects a student from the combo box that they want to update and then clicks the update student button. On click of the button, the student id is found using map reduce queries. Input dialogs appear and prompt the user to update entries. Each entry is set and updates the selected document id.



**

#### UPDATE Code Snippet

Filling the combo box

//Calls the allTnumbers view query & places them in a list

**public** List<String> **getdetails**() **throws** MalformedURLException, IOException{

List<String> queryListDetails = **new** ArrayList<>();

HttpClient httpClient = **new** StdHttpClient.Builder()

.url("http://localhost:5984")

.build();

CouchDbInstance dbInstance = **new** StdCouchDbInstance(httpClient);

CouchDbConnector db = dbInstance.createConnector("person", **true**);

ViewQuery query = **new** ViewQuery()

.designDocId("\_design/allTNumbers")

.viewName("allTnumbers");

ViewResult result = db.queryView(query);

**for** (ViewResult.Row row : result.getRows()) {

keyValue = row.getKey();

stringValue = row.getValue();

String queryDetails = row.getKey() + stringValue;

queryListDetails.add(queryDetails);

}

**return** queryListDetails;

}

**for**(String **s:** getdetails())

{

cboReadStudents.addItem(s);

}

Updating the selected item in the combo box

**private** **void** **jButton1ActionPerformed**(java.awt.event.ActionEvent evt) {

JOptionPane.showMessageDialog(**null**,"Updating " + keyValue + " " + stringValue);

**try** {

HttpClient httpClient = **new** StdHttpClient.Builder()

.url("http://localhost:5984")

.build();

CouchDbInstance dbInstance = **new** StdCouchDbInstance(httpClient);

CouchDbConnector db = dbInstance.createConnector("person", **true**);

ViewQuery query = **new** ViewQuery()

.designDocId("\_design/allTNumbers")

.viewName("getDocID");

ViewResult result = db.queryView(query);

String id ="";

**for** (ViewResult.Row row : result.getRows()) {

id = row.getKey();

}

**try**{

JOptionPane.showMessageDialog(**null**, "Document with ID: " + id + "will be updated");

// new update details

Student s = **new** Student();

s.setFirstname(JOptionPane.showInputDialog("Enter first name"));

s.setSurname(JOptionPane.showInputDialog("Enter surname"));

s.setTnumber(JOptionPane.showInputDialog("Enter T-Number"));

s.setEmail(JOptionPane.showInputDialog("Enter Email"));

Map<String,StudentAddress> addr = **new** HashMap();

String street = JOptionPane.showInputDialog("Address 1\n Enter Street");

String town = JOptionPane.showInputDialog("Address 2\n Enter Town");

String county = JOptionPane.showInputDialog("Address 3\n Enter County");

String country = JOptionPane.showInputDialog("Address 1\n Enter country");

addr.put("address", **new** StudentAddress(street, town, county, country));

s.setAddress(addr);

JOptionPane.showMessageDialog(**null**,s.toString());

db.update(id);

}

**catch**(Exception ex){

JOptionPane.showMessageDialog(**null**, "Error occured");

}

} **catch** (MalformedURLException ex) {

Logger.getLogger(DeleteStudents.class.getName()).log(Level.SEVERE, **null**, ex);

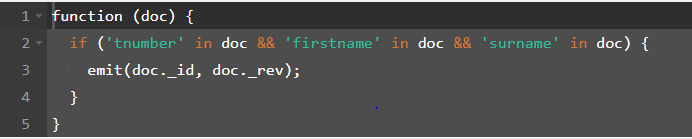
}

}

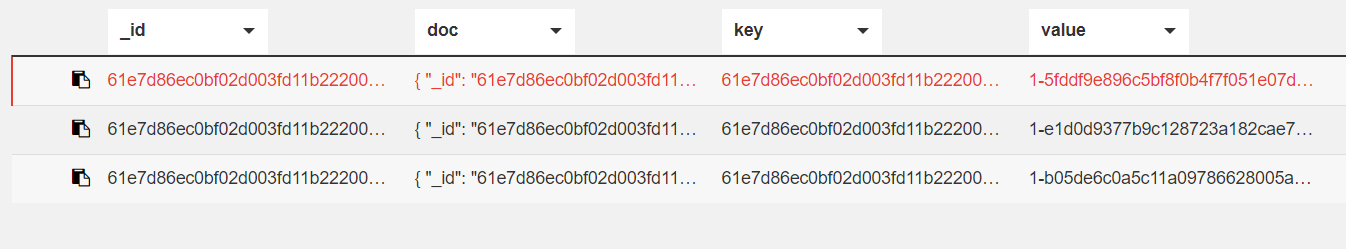
### Java UIF CRUD – DELETE

#### DELETE

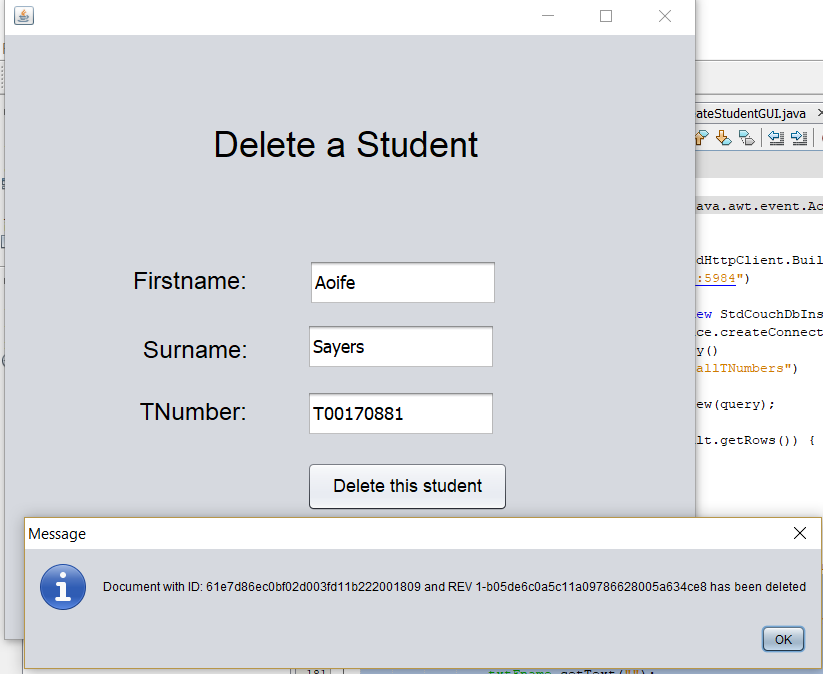
I created another call to a map reduce view to get the document id and revision number of a student with a valid first name, surname and tnumber for deletion.



The following image shows the results of all documents of the function as seen above



#### DELETING a student via the UI



#### DELETE – Code Snippet

**private** **void** **jButton1ActionPerformed**(java.awt.event.ActionEvent evt) {

**try** {

HttpClient httpClient = **new** StdHttpClient.Builder()

.url("http://localhost:5984")

.build();

CouchDbInstance dbInstance = **new** StdCouchDbInstance(httpClient);

CouchDbConnector db = dbInstance.createConnector("person", **true**);

ViewQuery query = **new** ViewQuery()

.designDocId("\_design/allTNumbers")

.viewName("getDocID");

ViewResult result = db.queryView(query);

String id ="", rev ="";

**for** (ViewResult.Row row : result.getRows()) {

id = row.getKey();

rev = row.getValue();

}

**try**{

db.delete(id,rev);

JOptionPane.showMessageDialog(**null**, "Document with ID: " + id + " and REV " + rev + " has been deleted");

}

**catch**(Exception ex){

JOptionPane.showMessageDialog(**null**, "Error occured - make sure the student exists");

}

**finally**{

txtFname.setText("");

txtSname.setText("");

txtTnumber.setText("");

}

} **catch** (MalformedURLException ex) {

Logger.getLogger(DeleteStudents.class.getName()).log(Level.SEVERE, **null**, ex);

}

}

# 3. Conclusion

To conclude, CouchDB is a very powerful open source NoSQL database. CouchDB’s native JSON document store trumps the traditional key value SQL store for less rigid data models. CouchDB adopts a semi-structured data model, based on the JSON (JavaScript Object Notation) format; JSON offers a lightweight alternative to XML; (Anon., n.d.). CouchDB’s data is durable like Oracle SQL’s rdbms. However, CouchDB is eventually consistent over time whereas Oracle SQL is automatically consistent. There is less of a learning curve for programmer new to CouchDB in that most MapReduce functions are written with JavaScript. To do the same functions with Oracle, a programmer would need to learn to do PLSQL and learn about procedures, triggers etc. CouchDB’s highly scalable NoSQL database favours the web. CouchDB’s master slave replication is also very for enabling users to distribute data across several nodes or datacenters, but also to move data more closely to clients. (Apache, 2017)

I decided to program with Ektorp – a Java persistence API that uses CouchDB as storage engine. (helun, n.d.). Ektorp uses the Java class Jackson for mapping Objects to JSON. Ektorp gives the programmer a choice of abstraction level from full object document mapping to raw streams. If I was to program with CouchDB I wouldn’t use Ektorp. I feel it’s not fully developed yet as some calls to update and bind to an object threw a lot of error. Checking on forums like Stack Overflow, I saw that I was not alone. Personally I think I would use NodeJS the next time as it seems more stable. I found that the update method and nesting of objects was not very efficient.

I also could have included more error handling with my project. My main focus was solely on carrying out the CRUD operations.

# References

Anon., n.d. *Putting into Practice: COUCHDB, a JSON Semi-Structured Database.* [Online]   
Available at: http://webdam.inria.fr/Jorge/html/wdmch21.html

Apache, 2017. *Replication Introduction Couch DB.* [Online]   
Available at: http://docs.couchdb.org/en/2.0.0/replication/intro.html  
[Accessed 17 October 2017].

helun, n.d. *helun/Ektorp.* [Online]   
Available at: https://github.com/helun/Ektorp

Warner Onstine, L. P., 2012. *infoq.com.* [Online]   
Available at: https://www.infoq.com/articles/warner-couchdb

Wasington, U. o., n.d. *Coursera.* [Online]   
Available at: https://www.coursera.org/learn/data-manipulation/lecture/qtEKw/couchdb-overview

Whitehouse, A., 2009. *When to use CouchDB vs RDBMS.* [Online]   
Available at: https://stackoverflow.com/questions/1307100/when-to-use-couchdb-vs-rdbms