|  |
| --- |
|  |

Portfolio Assignment 1

An investigation and implementation of couchdb

Eógan Sullivan | T00166087 | 20/10/17

Table of Contents

[Introduction 2](#_Toc496179866)

[Question 1 2](#_Toc496179867)

[Why I Chose This Design 2](#_Toc496179868)

[Example of JSON Document 3](#_Toc496179869)

[CRUD With Rest Interface 3](#_Toc496179870)

[Create 3](#_Toc496179871)

[4](#_Toc496179872)

[Read 4](#_Toc496179873)

[4](#_Toc496179874)

[Update 4](#_Toc496179875)

[Delete 5](#_Toc496179876)

[Question 2 7](#_Toc496179877)

[API Evaluation 7](#_Toc496179878)

[My Couch Installation 7](#_Toc496179879)

[CRUD With Driver And GUI 9](#_Toc496179880)

[Create 9](#_Toc496179881)

[Read 11](#_Toc496179882)

[Update 13](#_Toc496179883)

[Delete 14](#_Toc496179884)

[Map Reduce 15](#_Toc496179885)

[Advantages and disadvantages of the api 17](#_Toc496179886)

[Conclusion 18](#_Toc496179887)

# Introduction

For this project, I plan on developing a weather application in which a user can input and display temperature and weather status on a desktop computer or android device. Usually the input would be provided by different sensors for calculating atmospheric pressure and temperature however for the purpose of this project I will ask the user for input as I do not have access to such apparatus. My first thought where to build my application using JavaScript and Node.js. The reason why I thought this is because CouchDb works well with web applications and JavaScript is ideal for web based applications. However, I did not feel I would have sufficient time to learn JavaScript and create an application with it and this factor lead me to develop an application with C# and to find a suitable driver for C# and visual studio.

# Question 1

# Why I Chose This Design

I chose to design a Weather Application style database, in which an online user could view the current temperature, chance of precipitation and overview of the current cloud level. An offline user could view the weather when they are connected and when they are disconnected the database would save on their desktop/ phone and be able to display the information of the last time they connected, similar to how weather applications on phones currently work. The major difference would be that most weather apps display a n/a where the information should be when disconnected from the internet, whereas one with CouchDb could replicate the data when offline and sync with new data when a network connection is reestablished. I think CouchDb is a good fit for this is because CouchDb was designed for web application and the information would be distributed over the web. Couch Db is able to scale up and down, down for minimal information and locations or up for further implementations with more information and locations. CouchDb will allow for the application not to be connected to a network if the device has no access to one.

# Example of JSON Document



Here I am creating a document called Tralee with the temperature, precipitation and skies fields.

# CRUD With Rest Interface

## Create

I used Curl and CLI to create a document with the \_id of Killarney, below are screenshots of the cli and the prompts I used.

 This Command posts a JSON object called Killarney to my weatherApp database.

## 

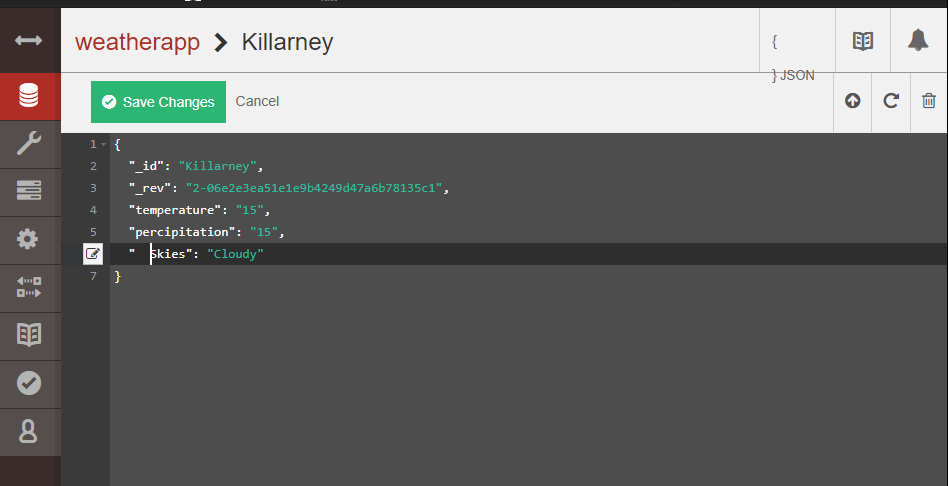
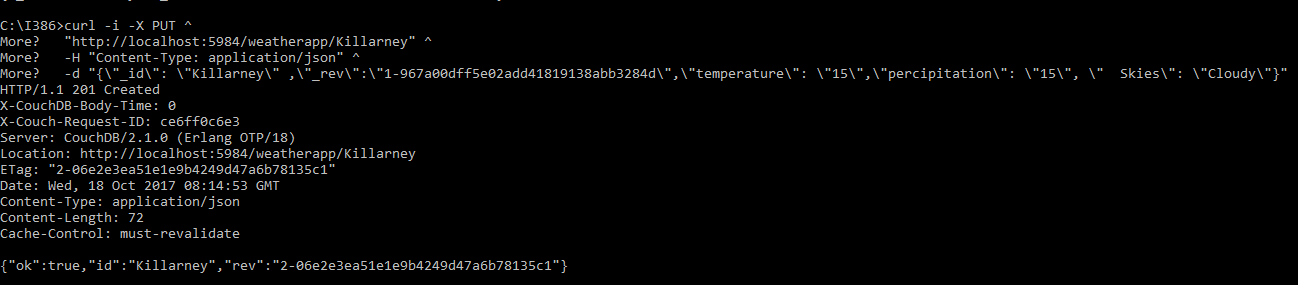
## Read

The curl command above calls the Killarney document written in the create operation and displays the result below.

## 

## Update

I used curls put command to update the Killarney JSON object with temperature, precipitation and Skies fields.

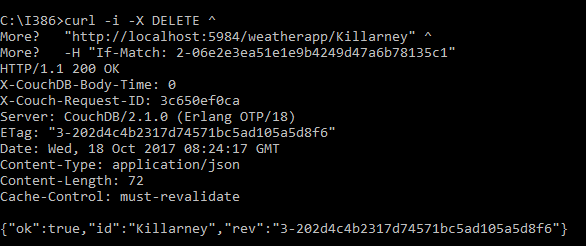
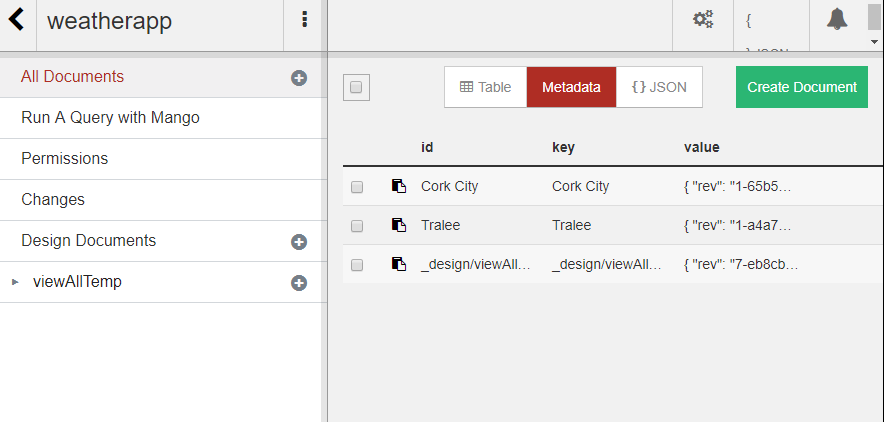


## 

## Delete

The delete command in curl checks the weatherApp database for the id and rev of the document and removes it from the database, the information is still there as can be seen by the rev number increasing if the document with the same id is created again.



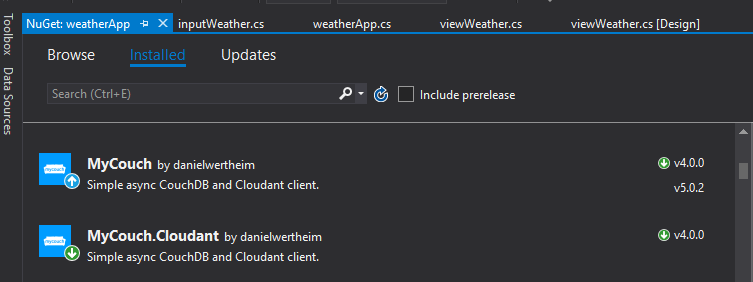


# Question 2

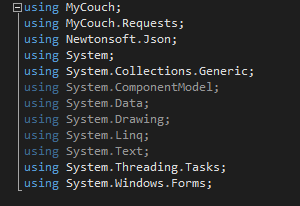
# API Evaluation

At first, I found it difficult to find an appropriate API for CouchDb, I initially tried to write my project in JavaScript but felt as I had never done JavaScript before and thought that I wouldn’t be able to finish the project in time so I tried to work in JAVA. In JAVA I used Eclipse as my IDE and installed LightCouch through Maven. From researching the API, I thought it would useful as the Database connection was made through the initialization of the CouchDb object. However, LightCouch was last updated in April of 2015, as it was two years old the version of JAVA that I was using was not compatible with a majority of the methods. I decided to further research different Java APIs. Many users of CouchDb, who wrote projects in Java used Ektorp. Ektorp is still being updated so logically I decided to try use Ektorp as a replacement for LightCouch. I again used Maven to install Ektorp, unfortunately after I installed Ektorp I had problems using it as it used apaches HttpClient which I couldn’t get working within my own code. Frustrated with JAVA and the APIs for CouchDb for the language I decided to research C# and its CouchDb APIs. I first built a GUI in C# then installed MyCouch through Nu get Package manager in visual studio. MyCouch was well documented on GitHub which made it the easiest to install and use. The documentation eliminated the confusion I had with errors in JAVA that I didn’t understand because it taught me how to form the connection and use the CRUD methods. The methods for working with CouchDb are asynchronous so it allows for multiple connections to exist and won’t cause errors as they close before a new connection is made. I felt MyCouch was the nicest of the three I tried and ultimately used it in my completed project.

# My Couch Installation

I first installed MyCouch and MyCouch.Cloudant, both need to be version 4.0.0 to work.

After these are installed I imported the necessary packages for the project



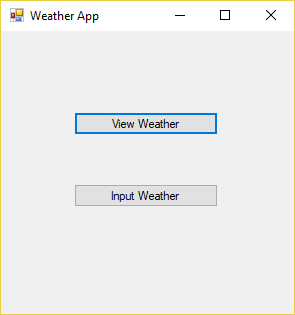
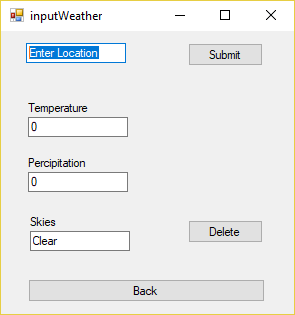
After these where imported, I began to connect my GUI to the database. The using function allowed for the connection and disconnection of the database once entered and exited respectively. This function is used for every call to the database with respect to every operation on task performed on it



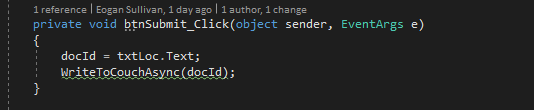
The MyCouchClient method takes in two arguments, the first a string which in this context is dbConnect which is “http://localhost:5984/ and a string representing the name of the database in couch that the application is to connect to. In this case it is the database for weatherApp, this could be user defined if the application I was building was a database management system.

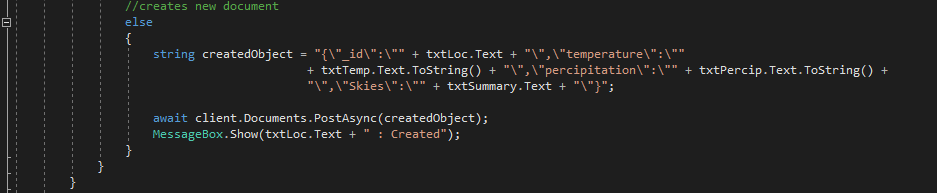
# CRUD With Driver And GUI

## Create



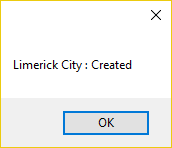
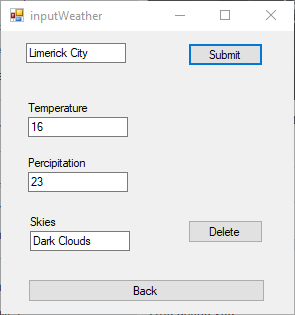
In Weather App, select Input Weather, this will take you to the Input Weather Screen, Enter the details in the text boxes, when satisfied hit submit.

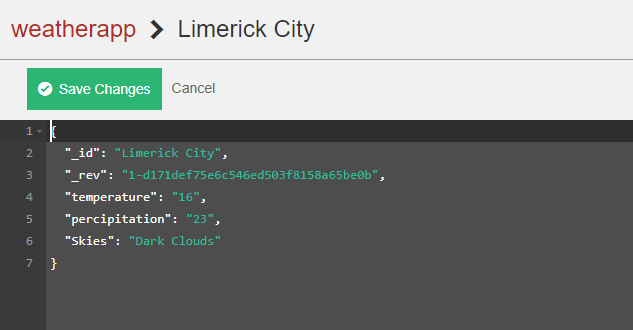


 The button makes a call to the WriteToCouchAsync method, the method takes in the text in the location textbox as an argument. The WriteToCouchAsync method checks to see if the document is created and if not creates a new document. Below is the code to create a new document.

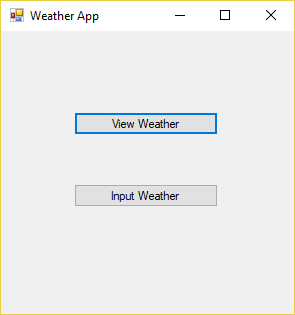
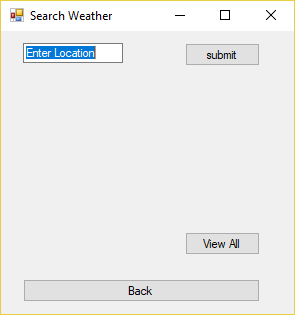
The createdObject string is saving the information passed in to the text boxes into the format of A JSON file. The client variable in the context of the project is the connection to the database, to createdObject string is sent to the database with Post commands built into the MyCouch API

Below is a Screenshot of the Inputs I entered into my GUI and the Document it created in my database.

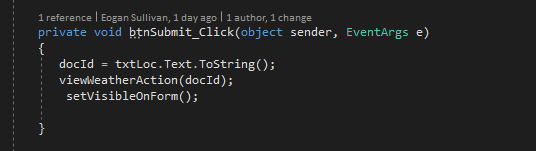




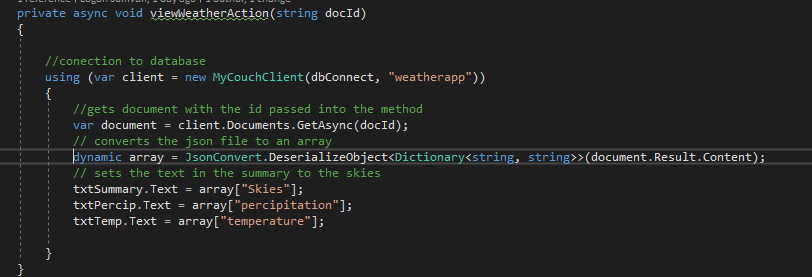
## Read



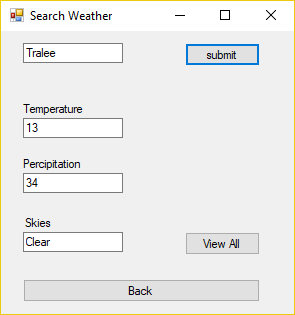
In Weather App, select View Weather, this will take you to the Input Weather Screen, Enter the details in the text boxes, when satisfied hit submit.



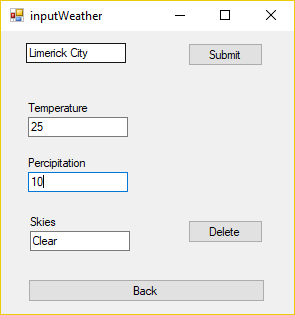
The Submit Button sets docId to the string in the location Text box, then calls the viewWeather Action method with the location as an argument after this is completed the setVisibleOn Form method sets all the text boxes on the form to visible so they can be seen.



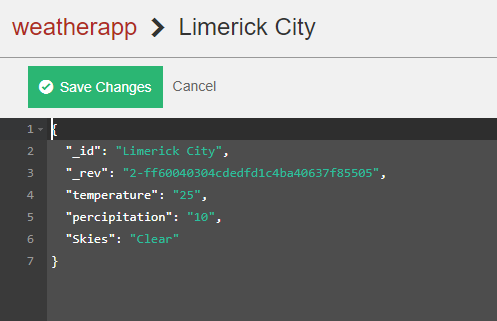
Like in the Create operation the client variable is the connection to the my CouchDb. The document variable the return of the method in the MyCouch apo GetAsync with the argument of the documents \_id. The dynamic variable array is then used to store the objects in the JSON file that is returned by the GetAsync method after it has been deserialised my Newtonsofts. JSON package. The last three lines of code set the text fields with the values in the document with the \_id that was passed in to the method. Below is the result of a search for Tralee.



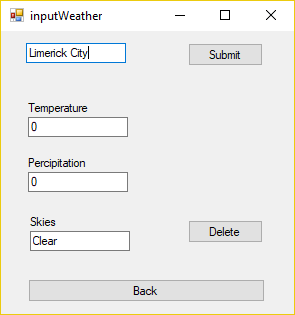
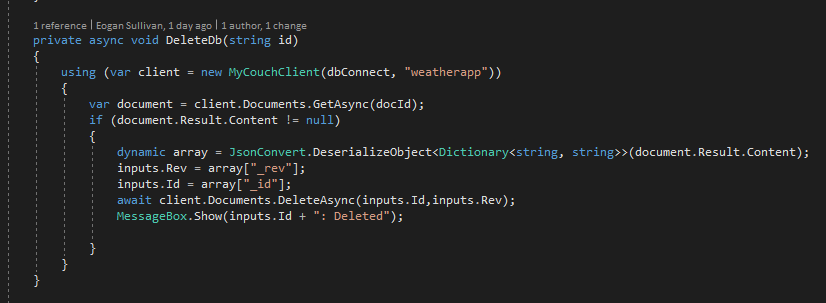
## Update

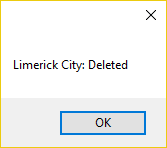


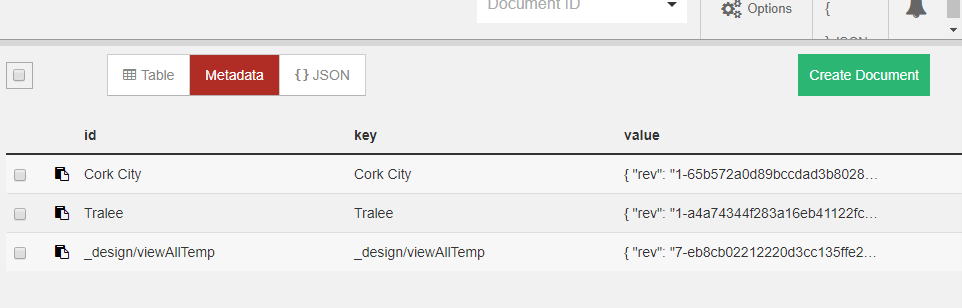
I wrote the create and update operations in the same method as to me it made sense in the context of my applications, if an object exists amend it, if not create it. In this part of the method, I find the document with the Id in the location text box, if the GetAsync method returns null it does not exist and defaults to the else statement in which the Create Operation is written. If the document returns some value then the document exists in the Database and we can amend it. I read in the document the same way as in the READ operation. I then store the \_rev and \_id into an object called inputs with a rev field and id field. I used the clients. Documents PutAsync method to update the database in the document. The method, PutAsync works the same as curls put function. The id and rev number are passed in as arguments first then the rest of the document is written the same in curl except represented as a string for the method. To confirm this method works I added a Message box with an updated message. Below is a screenshot of the updated database with the figures in the top left corner. Note I have updated the same document I created in the previous step as shown by the rev number update



## Delete

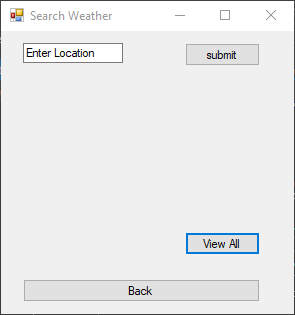
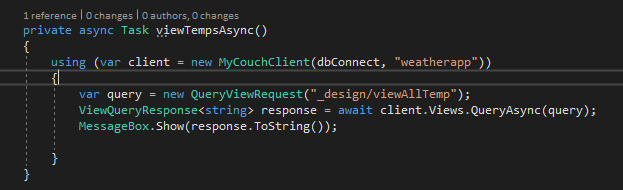


Behind the Delete button there is a call to the DeleteDB method, in hindsight this should have been renamed to DeleteDoc. It takes the document Id as an argument. I first find the document by id, if the document exists then it will enter the if statement. I then pass the id and rev numbers into the inputs.rev and inputs.id. I next invoke the DeleteAsync method and pass in the id and rev number of the document that was found by the client. This method removes the document from the database and once finished a message box that confirms the deletion. Above I entered the id of the document created and updated in the earlier operation and pressed the submit button, which deletes the document from the database. Below is the message box after I pressed the submit and the result in the database.

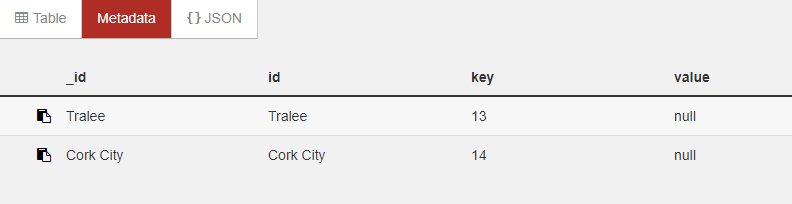


## Map Reduce

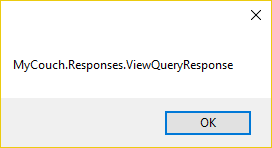
For my map reduce function I want to map all the locations in the database and reduce them to one temperature for each location. I did this by adding a button to the view weather window, if a user was to press the button then the GUI would display a window with the result of the map reduce function.



The application connects to the database, the QueryViewRequest gets the document in which the map reduce function was written. The VuewQueryResponse is the response of the Map Reduce. The Views.QueryAysnc method returns the query by running the document and saves the response in the response object. The message Box then displays the information.



Above is the view in the Fauxton Browser, below is the output of the code, for some reason the method does not work as it is intended to.



The text below shows the JavaScript code for the Map Reduce functions.





## Advantages and disadvantages of the api

Advantages

The API was straightforward to use, the fact that it was well documented was a bonus to figuring out how the methods worked and how to apply it to my project. The functionality of the driver made sense in the context of the database and the project. The installation of the API into visual studio was straightforward and it was fully integrated with C# with no need to install extra packages before use. MyCouch has methods that cover every aspect of couch dB’s functionality and I would recommend it for future projects.

Disadvantages

The latest version for some reason would not work with Visual studio and I had to roll back to a version before the latest. This could also be the reason the map reduce function isn’t working correctly. MyCouch like most CouchDb drivers haven’t been updated in years which would give the impression that CouchDb isn’t used that much so it was hard to find information on how to get methods working when they wouldn’t work and a lot of experimentation was done. If it had been an API used by industry or a better supported API I would imagine that it would be easier to use and better functioning methods. Another disadvantage I found was the naming conventions of methods, view query request and view query response logically a request would return a response but passing in a query to return a query felt like an unnecessary first step. If the method could find the map function by id it should run it and return the response to a string or appropriate data type.

# Conclusion

In conclusion I found CouchDb to be an outdated document database, from researching the databases online Couchbase lite seems to be more equipped with APIS and drivers for and development. However, there are lots of databases built on CouchDb such as Couchbase or pouch DB so a project built on one of them might be more relevant. I found CouchDb a good fit for this time of project as a relation database would have been too cumbersome in working out information and manipulating data where as a document store such as CouchDb was far easier to work with. My reasoning for developing such simplistic JSON object was that I felt that a weather app for viewing weather at the specific time would not need arrays nor nested objects as they would just need to return the specific temperate, precipitation and sky forecast for a given moment. The addition of an array or nested object would have added an unnecessary layer of complication to the project. If I wanted to add functionality like checking weather at a specific time of specific areas of a town I would have included them. Overall the use of a document stores for these type of applications is clear and are superior to relational databases for such web applications or offline storage databases.