Michael Edgar

Institute of Technology tralee | michael.edgar@students.ittralee.ie

CouchDB in Java

Using the lightcouch driver to perform CRUD operations on a hotel room management database

# Contents

[1. Contents 1](#_Toc23862934)

[2. Introduction 2](#_Toc23862935)

[3. Why CouchDB Suits this Application 2](#_Toc23862936)

[4. Java & IntelliJ 2](#_Toc23862937)

[5. LightCouch 3](#_Toc23862938)

[6. Postman REST Interface 5](#_Toc23862939)

[7. Hotel Rooms Application 7](#_Toc23862940)

[8. Conclusion 14](#_Toc23862941)

[9. Bibliography 14](#_Toc23862942)

# Introduction

In this report, I will outline one possible use of a CouchDB database, along with how it can be practically be implemented within a Java application. I will explain why CouchDB suits this application. After this, I will outline how CouchDB can be implemented into a Java application to allow CRUD operations to be performed from a user interface on the database. I will evaluate the CouchDB driver I chose and how it suits my chosen programming language. I will then outline how CouchDB CRUD operations can be performed on this database within the Postman REST interface. Finally, I will explain the code of my application, giving examples of the code working.

# Why CouchDB Suits this Application

The application I chose was a simple hotel room tracking system. It allows the user to perform CRUD operations and basic analysis on a hypothetical hotel’s rooms. In other words, the user can create, retrieve, update and delete both rooms and room types. A room in relation to this application contains only the room number and the room type (i.e. room 101 which is a double room), while a room type provides information on the contents of a room and the price of said room. A room type contains the name, capacity, beds and price of that room type. This implementation allows for multiple rooms of the same type, as is the case in many real-world hotels. For example, one room type named king can occupy 2 people at one time, contains one king-sized bed and costs 140.00 to stay in. The benefit of having a CouchDB database for this system is its ability to scale without losing performance. While my example hotel contains ten rooms, a real-world hotel may contain many hundreds of rooms, which could cause performance drops in a traditional SQL database. The same argument applies for the room type, as hotels could have any number of different configurations of rooms. For this reason, CouchDB provides a suitable database implementation for this system.

# Java & IntelliJ

I chose to create the client-side application using the Java programming language and the IntelliJ IDE. I decided to use Java and IntelliJ as I have been using Java to create applications since 2016, giving me the experience necessary to correctly identify and troubleshoot any issues that may arise. Another reason I chose to use Java is its wide use in industry; a recent study by TIOBE found that Java remains the top programming language at the beginning of November 2019 (TIOBE Software BV, 2019). The figure below outlines the ten most popular programming languages, giving a percentage rating of how popular each language is over A map with text

Description automatically generatedtime.

Figure 4.1: TIOBE Programming Index Graph

This graph shows that Java has been one of the top three rated programming languages since at least 2001. Java’s popularity was a significant factor for my choice in programming language, as using it gives me greater experience in a programming language that is seen as one of the most widely used in industry.

# LightCouch

After deciding to use Java to create the user interface for my application, I then had to find a suitable CouchDB driver to implement my required functionality. I decided to use the LightCouch API as a driver. I made this decision based upon its ease of use and simplicity. Implementation of this API required me to create a Java Maven project and import LightCouch as a dependency in my Pom.XML file (shown below)

A screenshot of a cell phone

Description automatically generated

Figure 5.1: Pom.XML containing lightcouch dependency

Once I had specified this dependency, I was able to connect to my chosen database by instantiating a CouchDBClient object and specifying its properties (i.e. target database) using the CouchDBProperties Class. This configuration is shown in the figure below.

A screenshot of a cell phone

Description automatically generated

Figure 5.2: CouchDBClient Configuration

The CouchDBClient contains separate methods to perform each of the CRUD operations. To create a document in the database, A JSON object must be created and is saved using the save method, which accepts the JSON object as a parameter. This method returns the response given from CouchDB (i.e. error message if it failed or id and revision number if it succeeded). The read operation can be performed using the find or view methods of the CouchDBClient; the find method accepts the class of the object you are attempting to read and the document id as parameters, returning the document it finds. The view method accepts the viewID as a parameter and returns the view from CouchDB. The view object that is returned has additional methods that allows for extra processing of the view. For example, the includeDocs method determines if the document for the view should be returned in the result. The update method updates the given document in CouchDB, requiring the document to contain the \_id and \_rev of the document. Finally, the delete method removes a given document from the database, once again requiring the \_id and \_rev of the document.

# Postman REST Interface

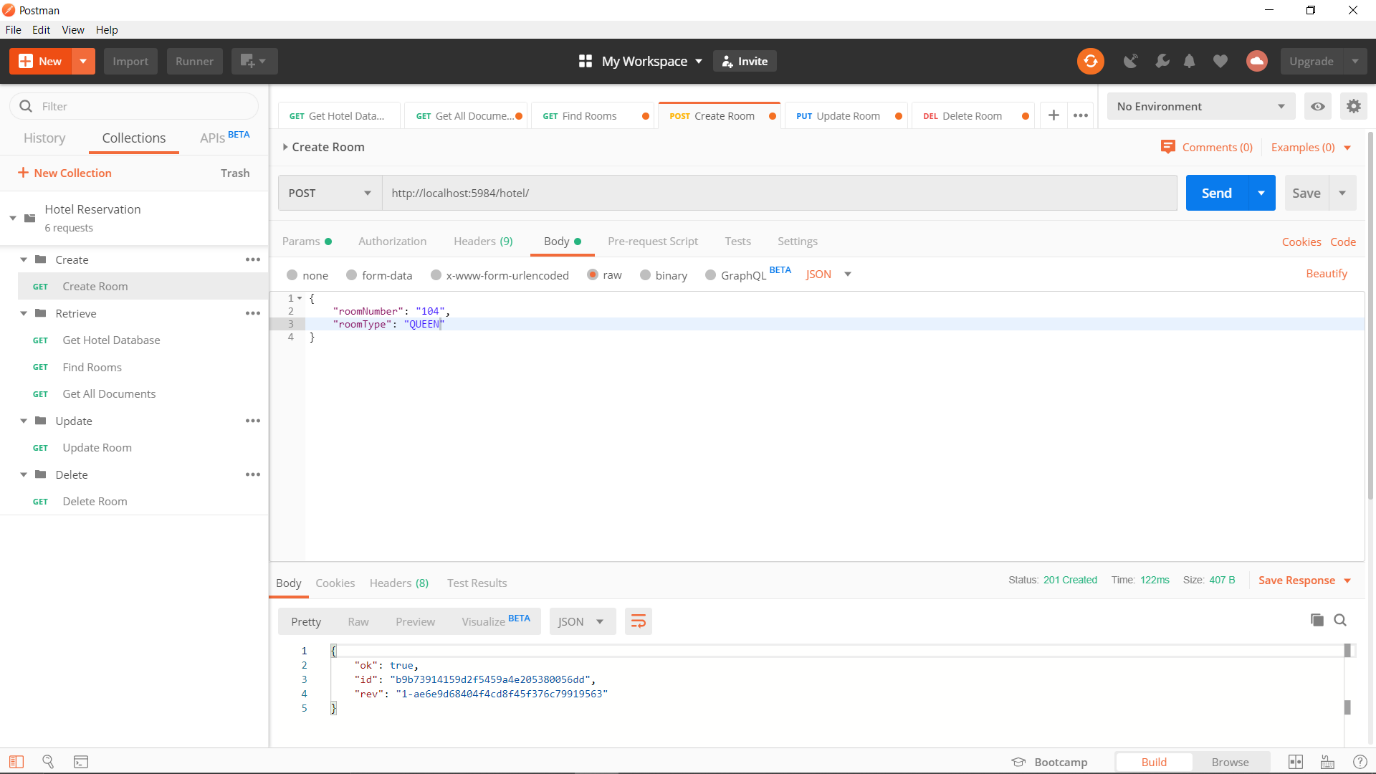


Figure 6.1: Postman Create

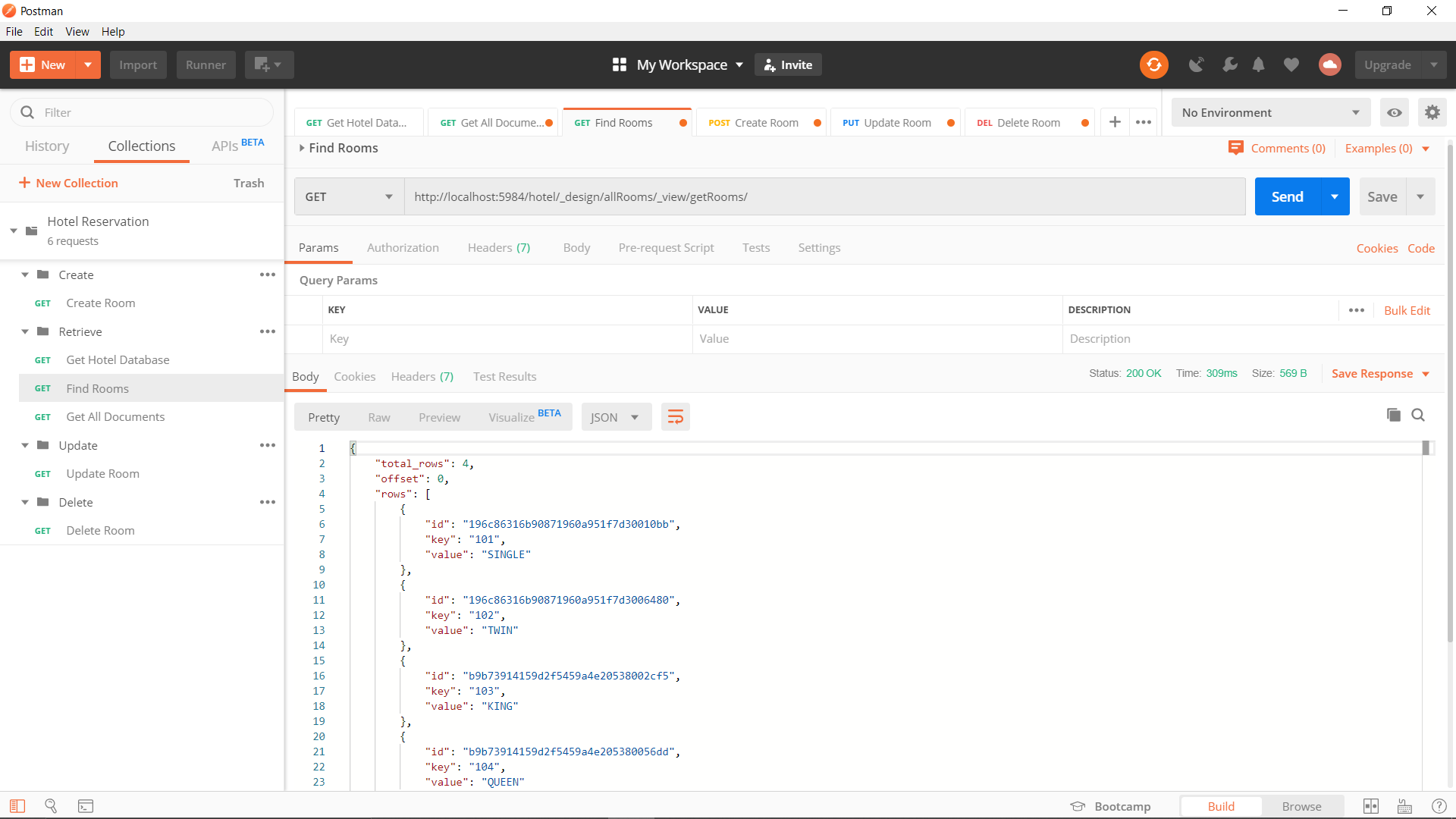


Figure 6.2: Postman Read

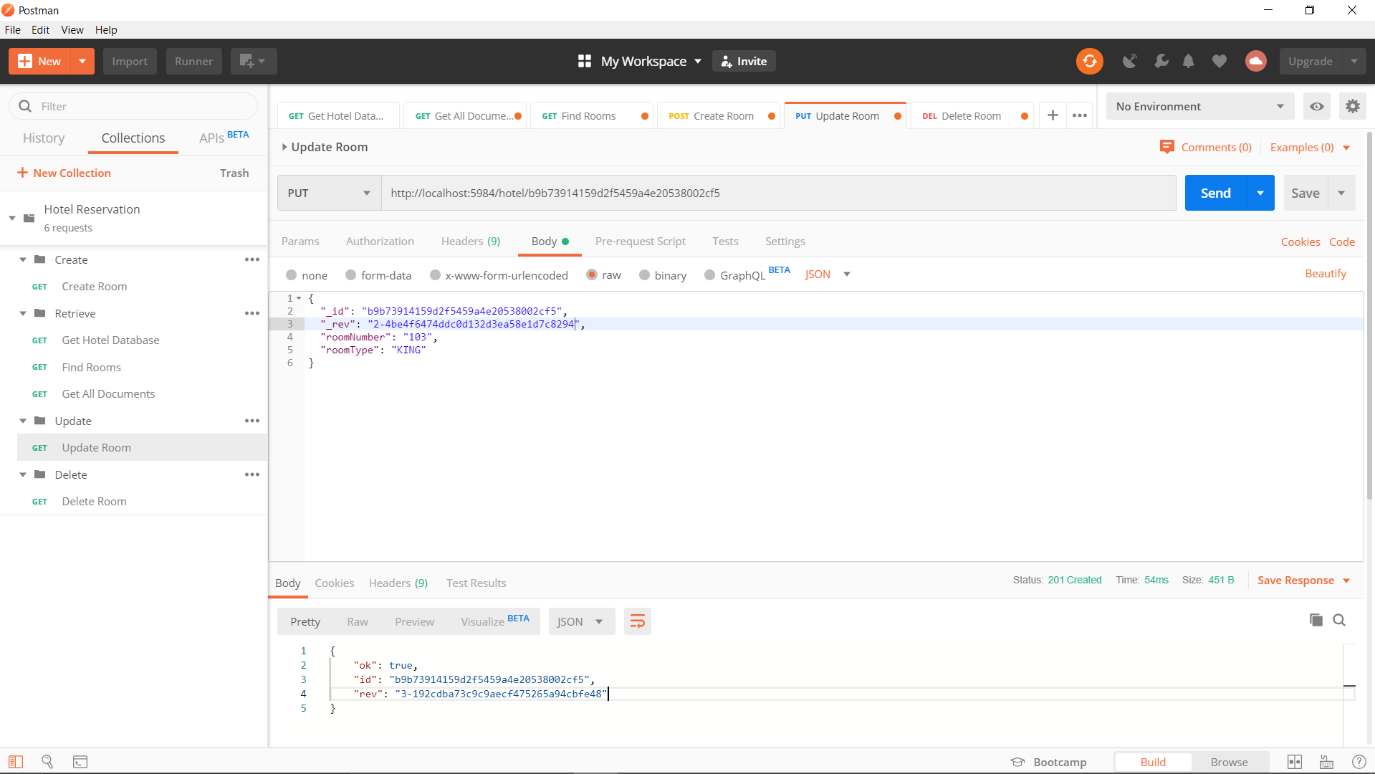


Figure 6.3: Postman Update

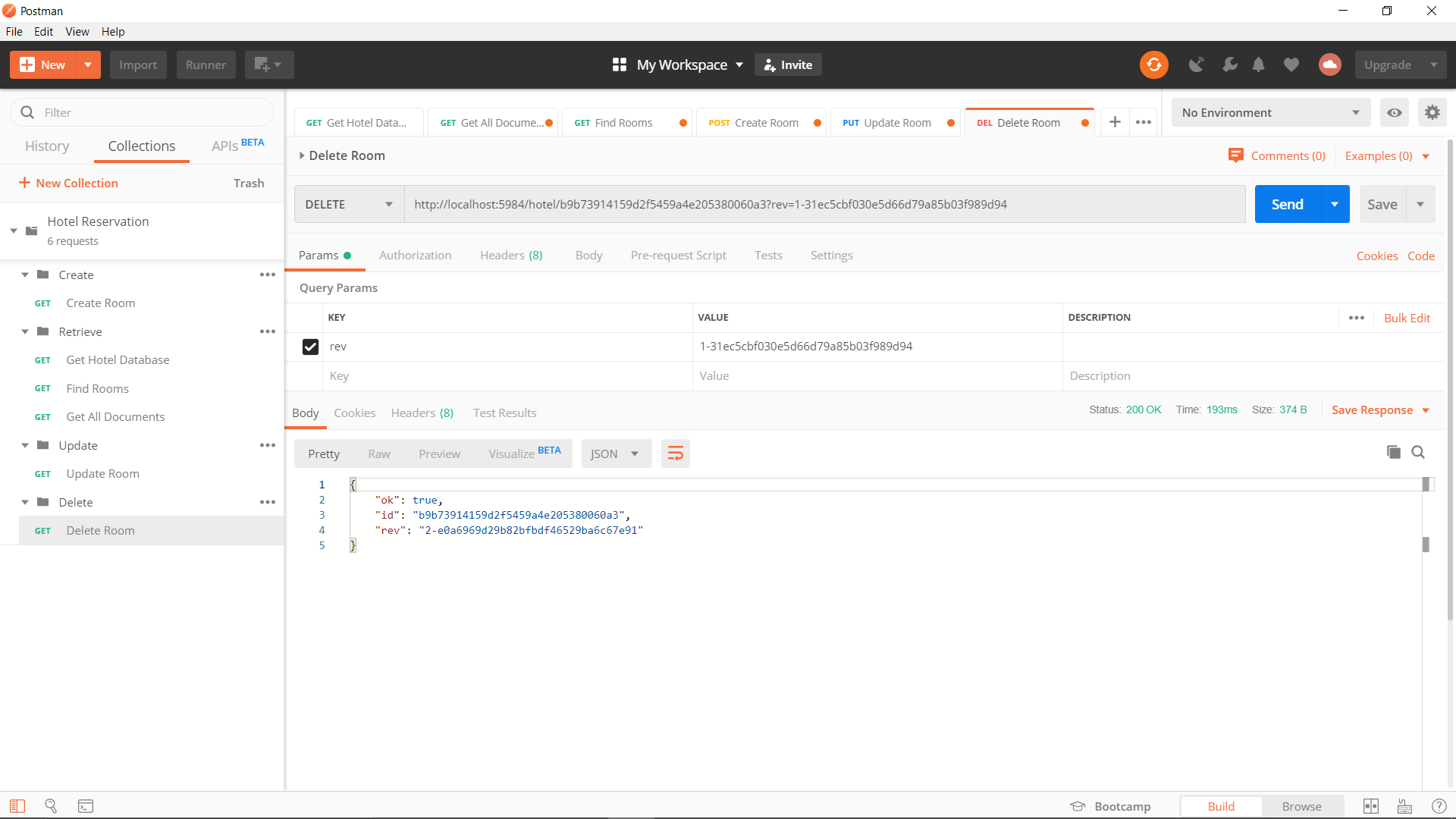


Figure 3.4: Postman Delete

# Hotel Rooms Application

A screenshot of a cell phone

Description automatically generated

Figure 7.1: Hotel App

A screenshot of a cell phone

Description automatically generated

Figure 7.2: Create Room in App

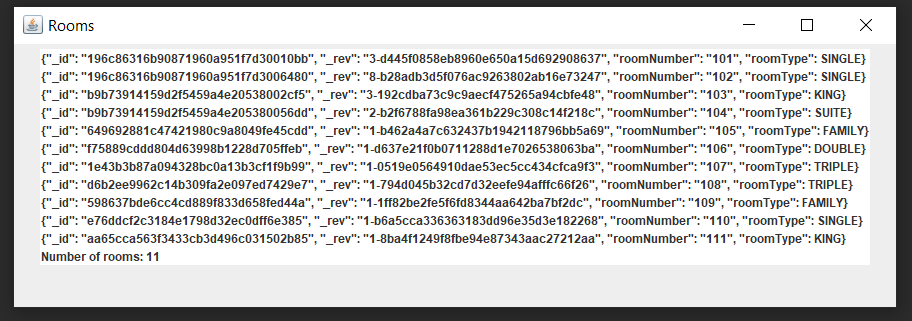


Figure 7.3: Read Room in App

A screenshot of a social media post

Description automatically generated

Figure 4.4: Update Room in App

A screenshot of a cell phone

Description automatically generated

Figure 7.5: Delete Room in App

A screenshot of a computer

Description automatically generated

Figure 7.6: Create Room Class

A screenshot of a cell phone

Description automatically generated

Figure 7.7: Create Room Logs

A screenshot of a cell phone

Description automatically generated

Figure 7.8: Read Room Class

A close up of text on a black background

Description automatically generated

Figure 7.9: Read Room Logs

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Figure 7.10: Update Room Class

A screenshot of a cell phone

Description automatically generated

Figure 7.11: Update Room Logs

A screenshot of a cell phone

Description automatically generated

Figure 7.12: Map Reduce Function App

A close up of a logo

Description automatically generated

Figure 7.13: Map Reduce Function Logs

A screen shot of a computer

Description automatically generated

Figure 7.14: Documents View CouchDB

A screenshot of a cell phone

Description automatically generatedA close up of a logo

Description automatically generated

Figure 7.15: Map Reduce Function to Count Rooms

A screenshot of a cell phone

Description automatically generated

Figure 7.16: Function to Get All Rooms

A screenshot of a cell phone

Description automatically generated

Figure 7.17: View of all rooms

A screenshot of a cell phone

Description automatically generated

Figure 7.18: Function to Get All Room Types

A screenshot of a cell phone

Description automatically generated

Figure 7.19: View of All Room Types

A screenshot of a cell phone

Description automatically generated

Figure 7.20: Example Room

A screenshot of a cell phone

Description automatically generated

Figure 7.21: Example Room Type

A screenshot of a cell phone

Description automatically generatedA close up of a logo

Description automatically generated

Figure 5: Map Reduce Function to Count All SINGLE Rooms

# Conclusion

As I have shown, CouchDB is a viable option as a database driver for the given hotel room system. Using LightCouch API, it can be integrated within a Java application using a Maven Dependency. LightCouch can perform the basic CRUD operations within a user interface, allowing full interaction with the CouchDB database.

# Bibliography

TIOBE Software BV, 2019. *TIOBE Index for November 2019.* [Online]   
Available at: https://www.tiobe.com/tiobe-index/  
[Accessed 5 11 2019].