Java Programming  
Report #2: Java IO/Databases  
ARM   
(Application for Restaurant Management)

**Class : 18CLC2-KTPM**

|  |  |
| --- | --- |
| **Group 07**: | **Dinh Hoang Duong – 18127084**  **Duong Tran Man Duy – 18127087**  **Huynh Duc Lee – 18127126** |

**Github:**

[tarzanchemgio/Java-Group07-ARM: Application for Restaurant Management (github.com)](https://github.com/tarzanchemgio/Java-Group07-ARM)

**Table of content**

[Revision History 3](#_Toc58078303)

[Introduction 4](#_Toc58078304)

[Analysis and design 5](#_Toc58078305)

[Implementation 7](#_Toc58078306)

[Sample data 9](#_Toc58078307)

[Result 10](#_Toc58078308)

[Plan 12](#_Toc58078309)

[References 14](#_Toc58078310)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 21/11/2020 | 0.0.1 | Login and Sign-up protoype | Hoàng Dương |
| 5/12/2020 | 1.0.0 | Complete report | Hoàng Dương  Đức Lê  Mẫn Duy |

# Introduction

Our application is tended to be run on multiple machines so we need some ways to transfer data around as needed.

E.g.

- When administrator create new dish, all machines should see it and update to Menu.

- There are many orders from many machines, bills need to be stored so manager can see them later on

The list go on and on.

We decide to choose MongoDB for the sake of easy to design schema and implementation.

Data are stored as JSON format in one server which implement RestAPI for CRUD operation. Other machines would connect to and make request as needed

# Analysis and design

Stored information:

-         User account of managers and employees

-         List of dishes

-         List of all bills

-         List of customer

JSON schema:

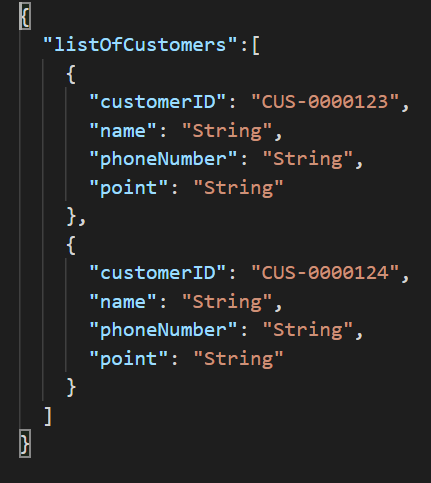
* User: (user’s password has been hashed by sha256)



* Item:



* Customer (Customer does not have DiscountPolicy yet):



# Implementation

The data is stored on a cluster in MongoDB, 2 drivers needed to be able to connect, load, save data: mongodb-driver-legacy and mongodb-driver-async.

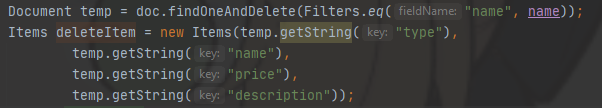
First of all, to connect to the database, we will access the database through an account. MongoDB will provide a connection string which is used to connect to a database in Java. After connecting, depends on the purpose that a specific collection in the database will be chosen to read. Below is the example way to connect to our database.

****

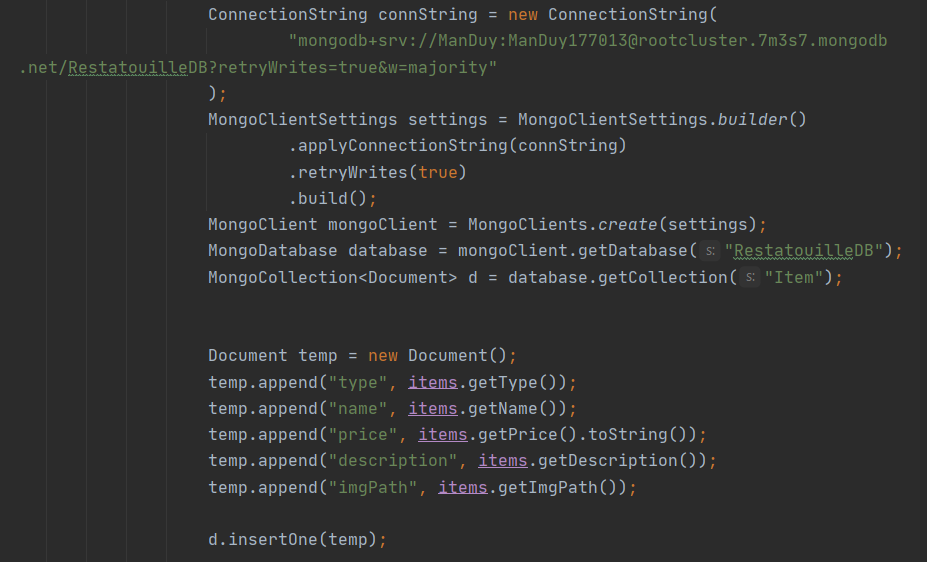
And connect to each collection in the database (here is Item)



To find and delete a specific items from database



The data will be splitted into collections. Currently, we have: Customer, Item and User

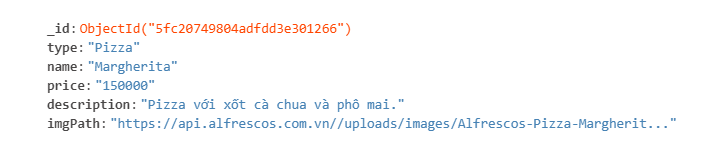


First is connecting to the database. After that is connected to the needed collection, for example is “Item”. Create a new Document object, add necessary information like the picture, insert the object into the collection, and that is finished. The way they add User’s data and Customer’s data is similar to Item’s data.

# Sample data

All the data stored on MongoDB Cloud

Item data sample:



User (Employee) data sample:



Customer data will be imported later.

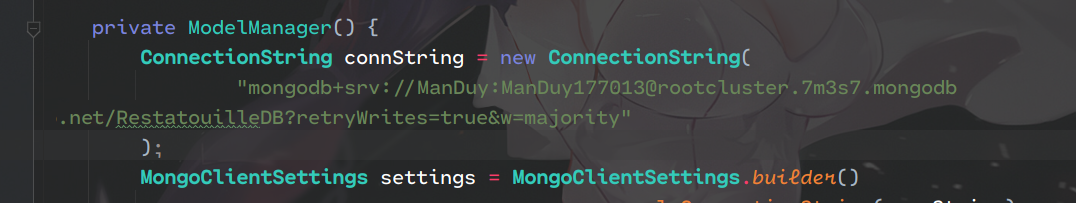
# Result

At this point, we have completed Login and Sign-Up function for both source code and UI design.

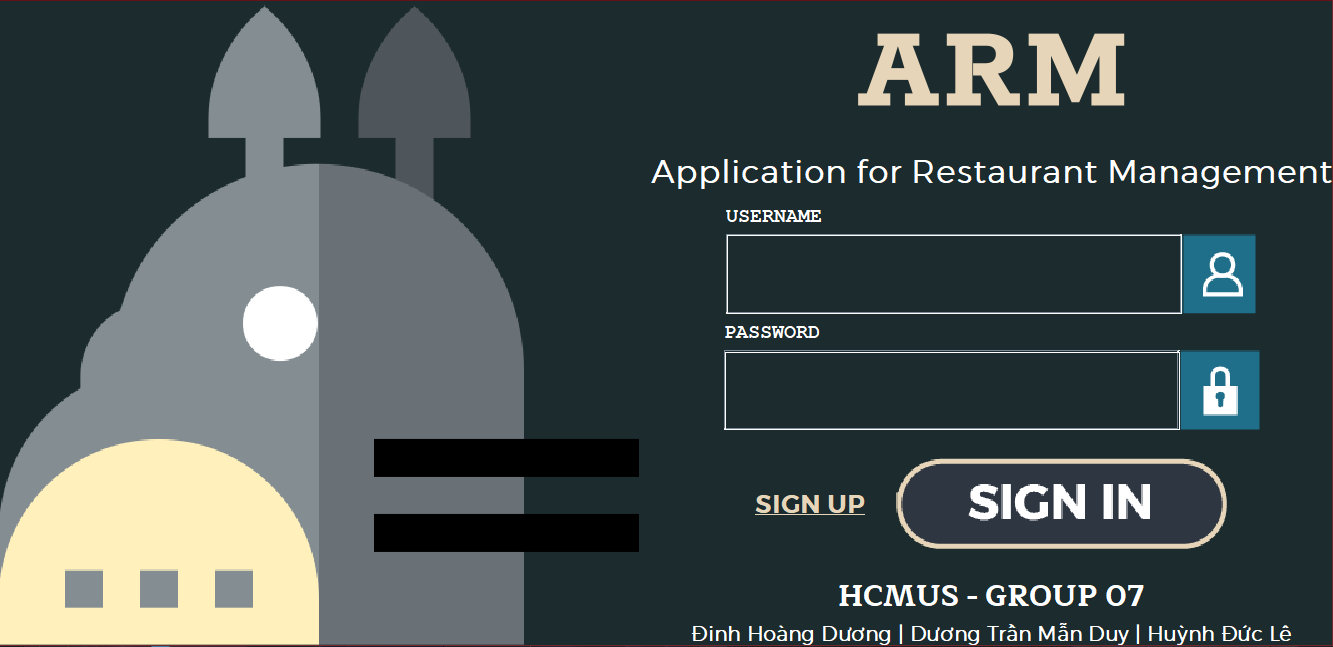
Also, a database server is set up and ready to use. This would force users to have an internet connection but this requirement is normal.

In source code, to avoid rapidly creating/destroying connection to the database, we made our database instance become singleton.

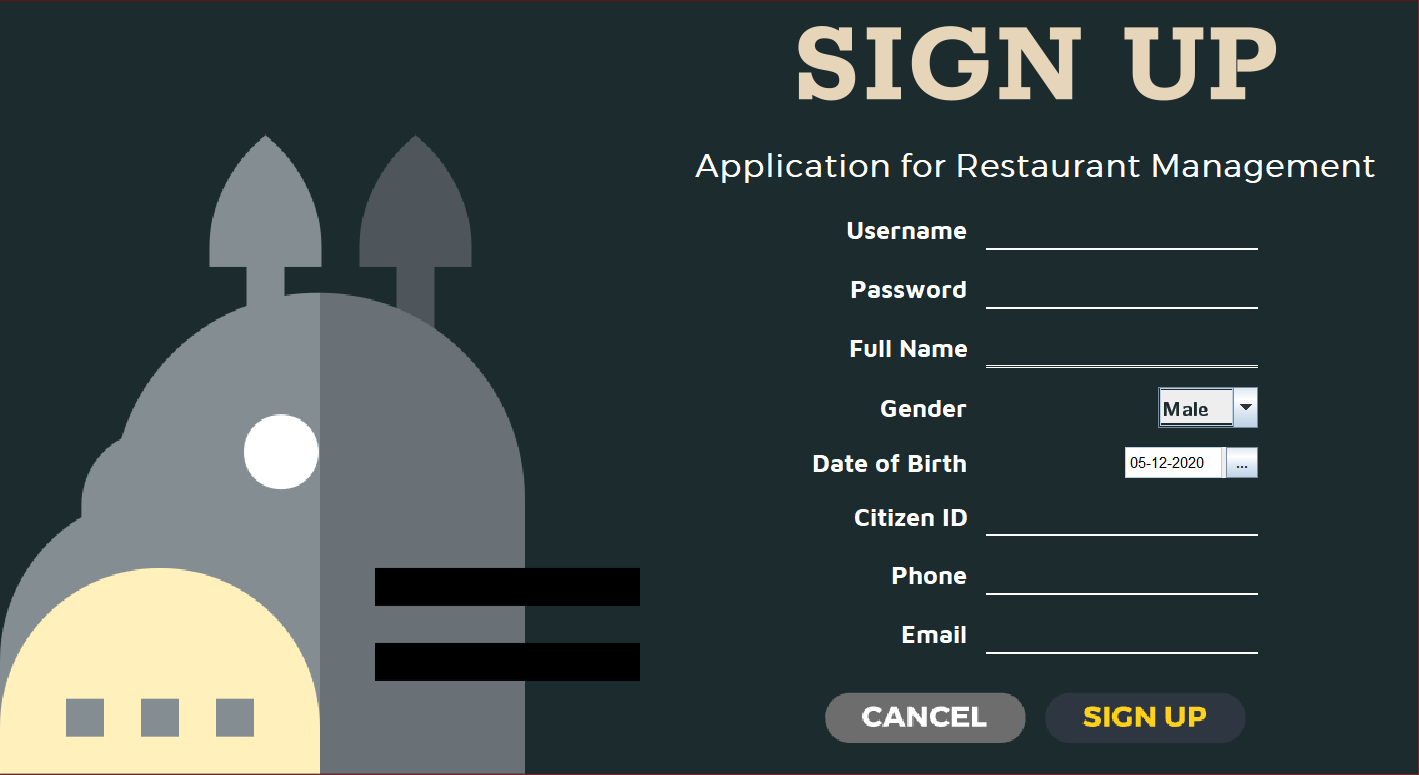
For database connection, we ship a database account, whose permission is under only the database owner, together with source code. This would cause serious security issues if byte-code were de-compile.



But this application was tended to be used by internal employees of a restaurant so we will live with that.



*(Sign In screen)*



*(Sign Up screen)*

# Plan

|  |  |  |
| --- | --- | --- |
| **Duration** | **Task** | **Author** |
| 11/11/2020 - 5/12/2020 | Design Login & Sign Up screen | Dương |
| Validate user input |
| Implement ViewModel for Login screen | Duy |
| Write methods to communicate with database |
| Implement ViewModel for  Sign Up screen | Lê |
| Write new User after sign up to database |
| 6/12/2020 - 12/12/2020 | Design Menu screen | Dương |
| Implement Bill view and confirm order |
| Implement ViewModel for Menu screen | Duy, Lê |
| Implement ViewModel for Bill |
| 13/12/2020 - 19/20/2020 | Design User Info screen | Dương |
| Implement ViewModel for User Info screen | Duy, Lê |
| 20/12/2020 - 26/12/2020 | Design Statistic screen | Dương |
| Implement ViewModel for Statistic  screen | Duy, Lê |
| 27/12/2020 - 2/1/2021 | Design Manage User screen for manager | Dương |
| Implement ViewModel for Manage User screen | Duy, Lê |
| 3/1/2021 - 9/1/2021 | Design Membership screen | Dương |
| Implement ViewModel for Membership  screen | Duy, Lê |

# References

[Stack Overflow - Where Developers Learn, Share, & Build Careers](https://stackoverflow.com/) - Solving all kinds of problem

<https://commons.apache.org/proper/commons-codec/download_codec.cgi> - For encryption

[CompletableFuture (Java Platform SE 8 )](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html) - Asynchronous Code

<https://www.flaticon.com/> - For UI elements

[Free Vectors, Stock Photos & PSD Downloads | Freepik](https://www.freepik.com/) - For UI elements

<https://github.com/LGoodDatePicker/LGoodDatePicker.git> - For date picker UI

[Maven Repository: com.google.code.gson » gson » 2.8.6 (mvnrepository.com)](https://mvnrepository.com/artifact/com.google.code.gson/gson/2.8.6) - For working with JSON

[https://www.mongodb.com](https://www.mongodb.com/) - For database