VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY THE INTERNATIONAL UNIVERSITY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

HOTEL BOOKING AND RESERVATION SYSTEM USING MICROSERVICES TECHNOLOGY WITH MEAN STACK AND SPRING MVC HIBERNATE

By
DO HUNG CUONG

A thesis submitted to the School of Computer Science and Engineering in partial fulfillment of the requirements for the degree of Bachelor of Computer Science

Ho Chi Minh City, Vietnam
Feb, 2018

HOTEL BOOKING AND RESERVATION SYSTEM USING MICROSERVICES TECHNOLOGY WITH MEAN STACK AND SPRING MVC HIBERNATE

APPROVED BY SUPERVISOR

_____

Tran Thanh Tung, Ph.D.

APPROVED BY: COMMITTEE

_____

Assoc. Prof. Chair Tran Manh Ha, Ph.D.

APPROVED BY SUPERVISOR

_____

Assoc. Prof. Huynh Kim Lam, Ph.D.

_____

_____

_____

_____

Thesis committee

ABSTRACT

Hotel business is a highly profitable industry but requires huge investment as well as having to meet the customer's demand. So how will your hotel be if one day the number of customers visit your hotel significantly reduce, and how do you know why it happens to your hotel?

You will never know why because you do not have any ability to know the customer’s feeling. Therefore, my solution is to implement a Hotel Booking system that supports all hotel business features with potential to track user’s behavior. Based on that, I can know what customer had done on my websites. I can see which pages customers clicked on, how long customers stayed at each page, which rooms, which services that customers had searched, booked, ordered or send the feedback. So far, I can improve my hotel system to catch up the wishes of customers

The customers will use a website to interact with my hotel, so that I have to make my system powerful. My very first plan is to experiment successfully some popular, modern technologies such as Single Page application, Microservice, MEAN, and Spring MVC to implement my Hotel Booking & Reservation System with more than 60 primary features for both customers and administrator.

List of Tables

List of Figures

Terms and Definitions

Term	Descriptions
Guest	User doesn't have an account who can use some unnecessary-login features
Customer	User has an account who can login to use some additional features
Administrator	User who can manage the system includes users, hotel items, request of customers and statistics.
HTML	Hypertext Markup Language, is used for browsers to render the content of the web page
CSS	Cascading Style Sheets, is used to decorate the elements of HTML
Javascript	JavaScript is the programming language of HTML and the Web allows you to implement complex things on web pages. Nowadays, Javascript can also run on server side thank to Node.js (see more in **section 1.3.4**)
jQuery	jQuery is open source Javascript library that supports the shorter syntax for Javascript coding that makes JavaScript easier to use for website developer.
Boostrap	Open source front-end framework includes HTML, CSS, Javascript, Jqeury, Image, Icon which are used to design templates and support ability to easily create responsive designs.
EJS	Effective JavaScript Templating, is simply used as a templating language which supports rendering HTML markup with plain JavaScript.
JSP	Java Server Page, is used as a technology for building the content of a web page using the Java programming language

Table 1: Terms and Descriptions

Table of Contents

Chapter I. INTRODUCTION

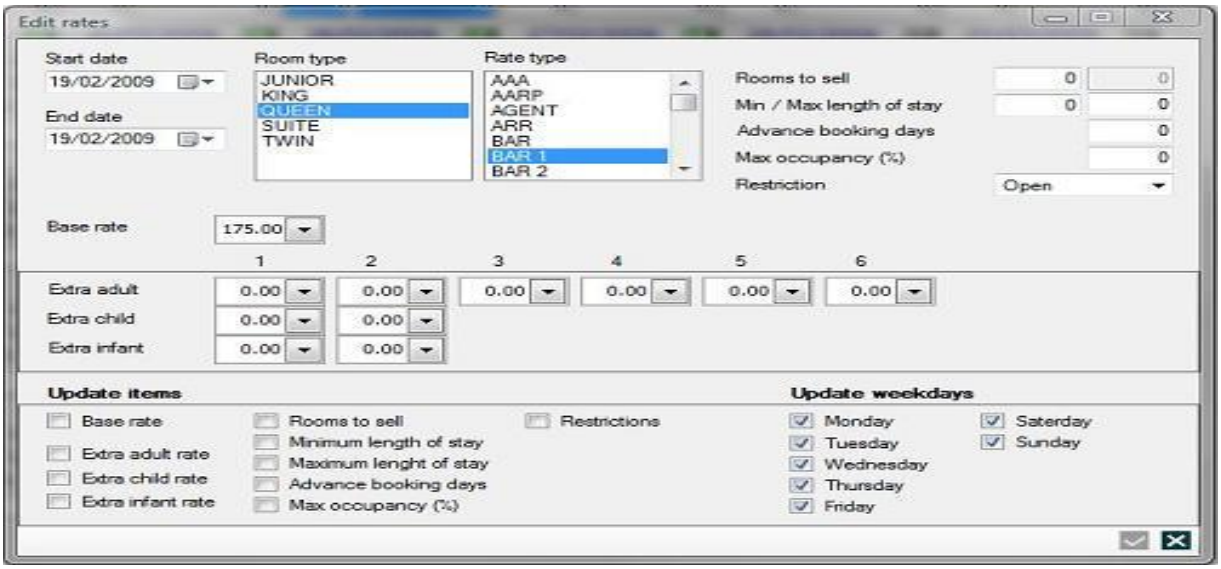
1.1. Problem without modern hotel booking system

Many years ago, in hotel business, the guest who really wanted to book a room had to come directly to the hotel and the reservation process had to be done before booking the desired room completely. In fact, when many guests come to hotel for booking at the same time, this hotel would be crowded. Then the customers had to wait for another booking, made a queue and wait for own turn. It wastes a lot of time and is very uncomfortable. (**Figure 1 – 2**)

Figure 1 – 2: Customers made a queue and wait for another booking at hotel.

Moreover, in many enormous hotels, management was very difficult where booking based on pen and paper was not convenient for both the customers and the receptionists. Besides, the reservations might cause a lot of risk such as invalid information of customers, wrong information of rooms and booking.

Therefore, many applications were released with lots of features to support hotel booking. However, at that time, performance of these systems might be very bad. They could overload or run extremely slow when a huge number of users was accessing at the same time. Moreover, the look and feel of these software user interfaces might not be well designed (**Figure 3 - 4**). It leads to situation that many customers or even the administrators who manage the system were not pleased to use those systems.



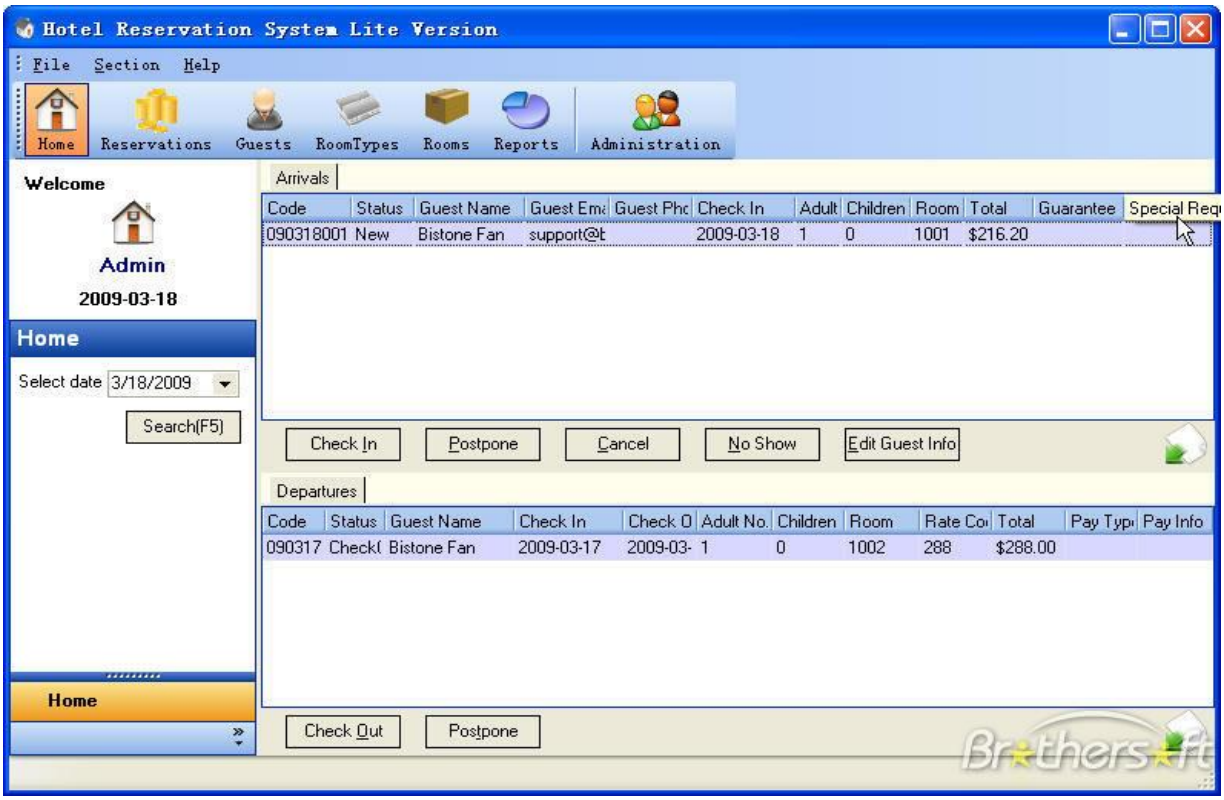


Figure 3 – 4: User interfaces of some old hotel booking systems. [1-2]

[1] https://www.planet-source-code.com/vb/scripts/ShowCode.asp?lngWId=10&txtCodeId=7186

[2] http://www.bistonesoft.com/litereservationstatus.htm

1.2. Modern Hotel Booking System

To solve these problems above, many deluxe hotels or five stars hotels in the world such as Marriott International [1], Hilton Worldwide [2] or InterContinental Hotels Group [3] already have their own hotel booking systems. You can see the modern Hotel booking website of Marriott International Hotel in **Figure 5.**

In developing technology industry, their systems were improved so much with friendly user interface, high performance and especially the ability to track the behavior of customers. With this tracking customers feature, the administrators, the managers or hotel owners could know what customer had done on their websites. They would know which pages customers clicked on, how long customers stayed at each page, which rooms, which services that customers had searched, booked, ordered or send the feedbacks.

Based on the data collection, the systems will automatically suggest what customers may like, recommend which rooms customers should book. Moreover, the hotel owners can improve their hotel business based on the information collected by their systems.

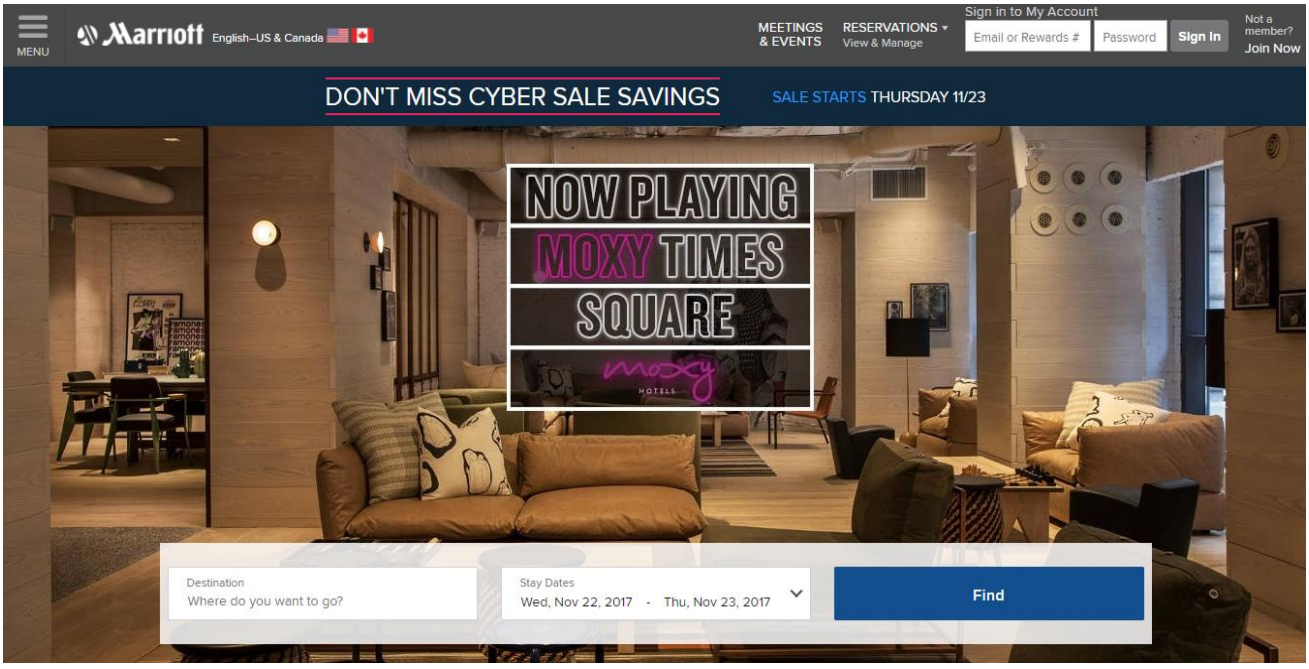


Figure 5: Marriott International's Hotel Booking System. [3]

[3] http://www.marriott.com/

1.3. Technology

Nowadays, there are many technologies for building a hotel booking system based on web application. Today, one of the most popular architecture is Single Page Application [4]. It is a new technology and it supports building web application loading a single HTML page and dynamically updating this page.

Following that trend, there are a lot of powerful frameworks or languages can be used to implement a Single Page Application such as Angular with data binding, Express framework with Node.js or Spring framework that can build RESTful Web Services to provide API for client side and so much more. In this section, I will list those technologies as well as the definitions and features.

1.3.1. Single Page Application

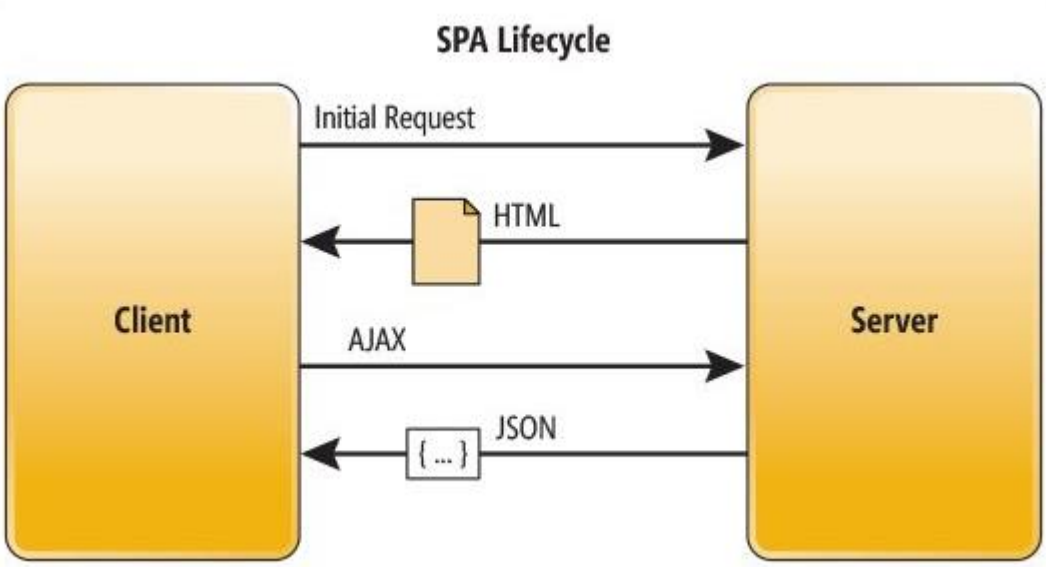


Figure 6: Single Page Application lifecycle. [4]

Single Page Application is a web application that on only one single web page or only one index page contains dynamic actions which we do not need to refresh the page. Single Page Application interactions can be handled without reaching server. [5]. The single page web App‘s lifecycle is shown is **figure 6** above.

[4] www.c-sharpcorner.com/uploadfile/rahul4_saxena/single-page-application-spa-using-angularjs-web-api-and-m/

Single Page Application can improve performance in many cases such as loading time, using AJAX, easy to navigate pages etc.

AJAX [6] is a dream of developer, because the following reason:

- It supports changing the content of a web page without reloading it.
- After the page has loaded, we can still send request or receive data from the server
- In the background, we can also send data to a server

That is the reason makes the end users feel more comfortable when using Single Page Application.

Recently, many frameworks, platforms or techniques were released to support building a Single Page Application. Angular is one of the most popular Single Page Application framework.

1.3.2. Angular

Over the last 4 years, Angular has turned into the main open source JavaScript application framework. A huge number of developers all over the world begin to learn Angular and the Angular version 1.x has been widely used and became well-known for website developers to build the complex applications. Then, Angular 2.x has been released to improve and overcome the weaknesses of its previous version. [7]

In the version 2 of Angular, almost the concepts in version 1 (AngularJS) are completely changed. Therefore, they look like two different frameworks instead of two versions of the same framework.

1.3.2.1. Angular 1 or AngularJS

Figure 7: AngularJS by Google. [5]

Recently, many frameworks, platforms or techniques were developed to support building a Single Page Application. AngularJS (**Figure 7**) is one of the most popular Single Page Application frameworks. It allows us to build a single page application easily because of the following reason:

- Angular makes the HTML more expressive by support some features such as if-else condition, switch-case, loop and local variable.

- Angular has powerful data binding. Thank to data binding, we can easily display variables from the data model such as component, track changes, and process updates from the user.

- Angular promotes modularity by design. Every Angular application is a set of building blocks and that is easier to create and reuse content.

- Angular has built-in support for communication with a back-end service. In Angular application, it is easy for the front-end to integrate with a backend server to get and post data or execute server-side business logic. [8]

With so many developers already using Angular 1 or AngularJS, why do we need Angular 2?

[5] https://commons.wikimedia.org/wiki/File:AngularJS_logo.svg/

1.3.2.2. Angular 2

Figure 8: Angular2 with TypeScript. [6]

Angular 2 (**Figure 8**) is faster than Angular 1. Angular 2 is built for speed, so that the initial load is faster, the change detection and improved rendering times are also faster than Angular 1. [9]

The fewer concepts of Angular 2 make it easier to understand than Angular 1. The code structure is simplifier than the previous version of Angular. Therefore, it is easier for the developer to learn and use Angular 2.

In Angular 2, we can use Typescript which is a super set of Javascript. Typescript is a form of JavaScript, in Typescript we can know types and classes. We can compile Typescript into JavaScript. TypeScript is an open source that contains many aspects of object orientation such as interfaces and inheritance. The TypeScript's syntax is cleaner than Javascript and similar to C# or java. Because of using TypeScript, so we can use all its libraries and the functionality of TypeScript itself in Angular 2. [10]

Angular 2 is mainly focused on mobile apps. Angular 1.x was not built with mobile support in mind, where Angular 2 is mobile oriented.

Angular is simply a front-end framework for building applications. It is not the right determinant for what backend you should use for your application.

[6] https://www.cubettech.com/blog/angular-2-and-typescript-a-high-level-overview /

1.3.3. RESTful Web Service

Figure 9: RESTful Web Service. [7]

There are many ways to connect Angular to backend server. RESTful Web Service (**Figure 9**) which is essentially REST Architecture based Web Services is one of the architectural style that helps Angular and backend server communicate with each other. [11]

In REST Architecture, everything is a resource. RESTful web services are light-weight services so the developers usually use RESTful web services to make APIs for web-based applications. [11]

In some case, RESTful Web Service help us write a software application in various programming languages and we can run them on various platforms. For example, we can write a backend server in Java using RESTful web service and connect to Angular on client side by HTTP method.

REST is a web standard based architecture which was first presented by Roy Fielding in 2000. The word 'REST' means Representational State Transfer. REST uses HTTP Protocol for data communication. It spins around resources where each component is a resource and a resource accessed by a typical interface utilizing HTTP standard methods. [11]

RESTful Web Services are Web services based on REST Architecture. They use HTTP methods to implement the concept of REST architecture. URI is usually a service which a RESTful Web Service provides resource such as Text, JSON and XML. [11]

There are many frameworks written in many programming language support build a RESTful Web Service such as Node.js REST API or Spring REST API for java.

[7] http://codedestine.com/introduction-restful-web-services/

1.3.4. Node.js & Express framework

Figure 10: Node.js and Express framework. [8]

Node.js (**Figure 10**) is a JavaScript runtime based on Chrome's V8 JavaScript engine. Node.js utilizes an non-blocking I/O, event driven which makes it lightweight and productive. [12]

The package npm of Node.js is the biggest of open source libraries on the planet. Node.js application is written in Javascript so it is very easy to interact with Angular to build a Single page application easily. [12]

Express is a minimal and flexible web application framework for Node.js. The advantage of Node.js combine with Express framework is listed below:

- APIs

With many HTTP useful, it is very quick and easy to create a powerful API especially RESTful Web Service with API as JSON-like format. [13]

- Performance

An Express application always has higher performance than a lot of other frameworks for another language. Node.js is built to be a real-time language which means the average handle speed is very fast and it is calculated by millisecond. [13]

[8] https://hasura.io/hub/projects/hasura/hello-nodejs-express

1.3.5. Java & Spring framework

Figure 11: Java and Spring framework. [9]

Java (right hand side of **Figure 11**) is a cross-platform programming language, it is one of the most popular programming language which powerfully supports OOP (Object-oriented programming). Object-oriented programming is programming language model, the reason it called "Object-oriented" is that it focuses on organization around objects rather than function and data rather than logic.

Spring MVC (left hand side of **Figure 11**) is the most powerful J2EE (Java Platform, Enterprise Edition) framework to build Java web application. It is an open source Java platform that provides comprehensive infrastructure support for developing robust Java based Web applications very easily and very rapidly.

Spring MVC provides a model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements. [14]

We can also build a Spring RESTful Web Service with some supported library such as jackson and gson.

[9] http://www.interviewquestionandanswers.com/javaspring.html

1.3.6. SQL & Hibernate

Figure 12: SQL and Hibernate framework. [10 - 11]

SQL (left hand side of **Figure 12**) is a standard language for storing, manipulating and retrieving data in databases. Up to now, there are many kinds of SQL: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres and other database systems. [15]

Hibernate ORM (Hibernate in short, right hand side of **Figure 12**) is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. The Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions. [16]

The primary feature of Hibernate is mapping from Java classes to database tables, and mapping from Java data types to SQL data types. Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from the manual handling and object conversion of the result set. [16]

[10] http://hibernate.org/

[11] http://blog.stoneriverelearning.com/6-reasons-why-you-should-learn-sql/

1.3.7. MongoDB

Figure 13: NoSQL and MongoDB. [12-13]

MongoDB, an open-source database, is one kind of NoSQL database system because the entity relationship is not mandatory. (**Figure 13**)

In MongoDB, we have a mechanism for retrieval of data or storage in JSON-like documents that can vary in structure. Therefore, we can store related information together and with MongoDB query language, we can execute faster than SQL database system. [17]

In MongoDB, we use dynamic schemas so we can create records without first defining the structure. We can also simply add delete fields or add new fields to change the structure of records (documents)

MongoDB is availability and scalability; it supports ability to store a lot of complex structures such as objects, arrays easily. [18]

[12] https://www.mongodb.com/

[13] https://intellipaat.com/blog/nosql-database-tutorial/

1.3.8. MEAN stack

Figure 14: MEAN stack technology. [14]

To build a Web Application we need at least one type of database system to store all the data.

For example, to build hotel booking system, we need to store the information and so on for the servers to retrieve and display on the client user interface or to update the data which users saved to the database.

MongoDB stores data in JSON-like documents so that it is a good type of database system to combine with Node.js, Express framework and Angular to build the best Single page application. It called MEAN stack which means MongoDB, Express, Angular and Node.js. (**Figure 14**)

MEAN is a user-friendly full-stack JavaScript framework ideal for building dynamic websites and applications. It is a free and open-source stack designed to supply developers with a quick and organized method for creating rapid prototypes of MEAN-based web applications.

One of the main benefits of the MEAN stack is that a single language, JavaScript (in Angular 2, Typescript will be compiled into JavaScript) runs on every level of the application, making it an efficient and modern approach to web development. [19]

[14] https://medium.com/@alibhatti99/how-to-make-a-front-end-application-using-the-mean-stack-535e1622860c

1.3.9. Microservice

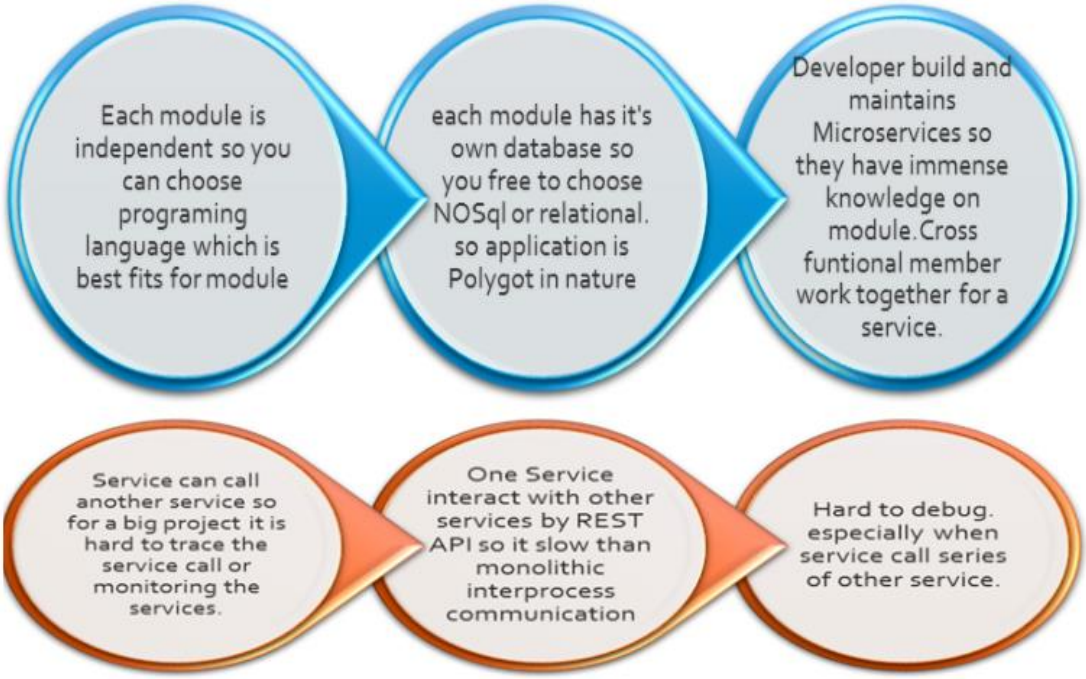


Figure 15: Benefits of Microservice [15]

Microservice is an architectural style where its application is structured as a collection of loosely coupled services and each service implements its own business capabilities. [20]

It is simply known as a combination of many small services called module that is most popular used in many companies to implement large-scale projects.

The Microservice architecture allows us to continuous deliver and deploy the large and complex applications. Based on its architecture, the organizations can evolve its technology stack for their own products.

The **figure 15** above clearly shows the benefits of Mircoservice.

[15] https://dzone.com/articles/microservices-basics

1.4. Goals and Scope

To keep up with current trends in hotel business industry, I will build a Hotel Booking and Reservations system where includes some features that similar to those five-stars hotel booking systems. It means experiment successfully all technology above especially MEAN and Spring MVC to implement Hotel Booking & Reservation System with more than 60 primary features likes a modern hotel booking system.

The main architecture of my system is using MEAN stack technology and J2EE with Spring MVC framework. Applied MEAN stack technology, my system becomes an online single page application with high performance and dynamically loading thank to Angular and RESTful web service which is built by Node.js and express framework.

Furthermore, the administrator's system is built with Spring MVC, the most powerful java frameworks so it becomes a cross-platform system and runs well with all operating system. With dynamic webpages and friendly user interfaces, customers will be very comfortable when booking rooms on my website and the administrators can manage the whole system easily.

Besides, my application supports almost features for hotel bookings as well as reservations management with ability to track user's behavior and it will provide data collection for applying AI machine learning in the future.

Chapter II. Background

With all the definition of the technologies in **section 1.3** above, you might have some knowledge about them. Therefore, in this section, I want to show the architecture of those technologies which necessary for my experiment to implement my hotel booking system.

In this chapter, I will focus on the architecture of each technology I have mentioned in **section 1.3**.

As you can see, MVC and REST are general architectures for the whole of my system.

2.1. Model – View – Controller

MVC stands for Model – View –Controller (**Figure 16**)

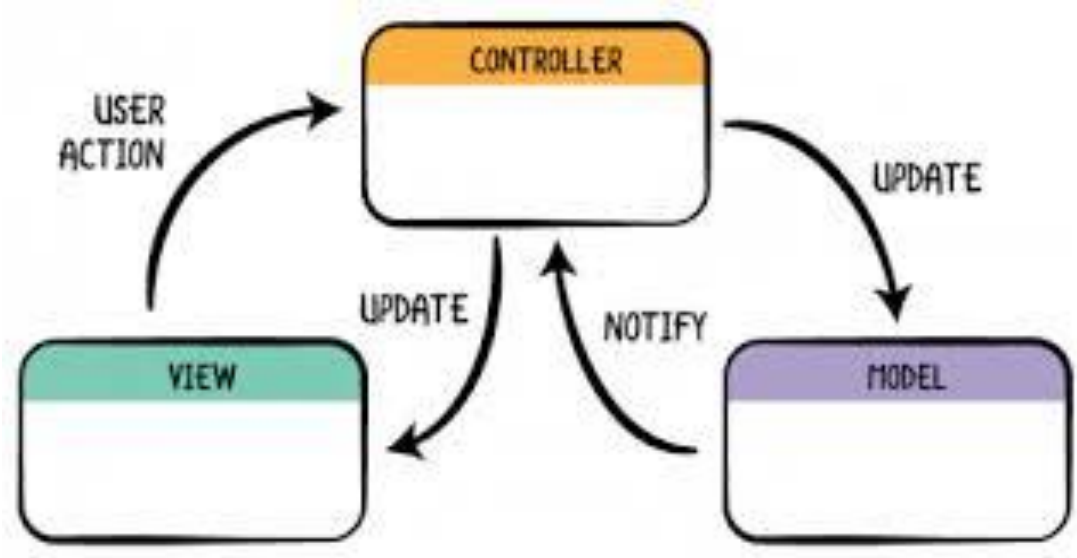


Figure 16: MVC pattern. [16]

[16] https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach

Model

The Model represents the data-type which corresponds to all the data-related logic we need to implement the system. It also supports data transferring between the View and Controller. [21]

View

View is used to display for users. It contains all UI logic of the application. [21]

Controller

Controller performs the actions between Model and View that executes all the business logic and requests and responses. Controller uses the Model to produce data then interact with the View to render the final output. [21]

A MVC framework is a combination of these above Model-View-Controller that supports building a web application easily. It creates a mechanism for communication between view and controller that can use Model as data-type or database interaction.

2.2. Spring MVC

Spring MVC is one of the most popular frameworks for Java web developer and its architecture is based on MVC framework.

Because it follows MVC pattern, the architecture must be separate into three main components: Model, View and Controller. Each component is similar to MVC framework that I have mentioned.

However, in Spring MVC's architecture (**Figure 17**), there is a concept called Front-Controller. It is an initial level of contract point for handling a request. It provides a centralized entry point for that controls and manages web request handling. It also reduces java code and business logic by promoting code reuse ability across the requests. The front controller separates view handling from controller. [22]

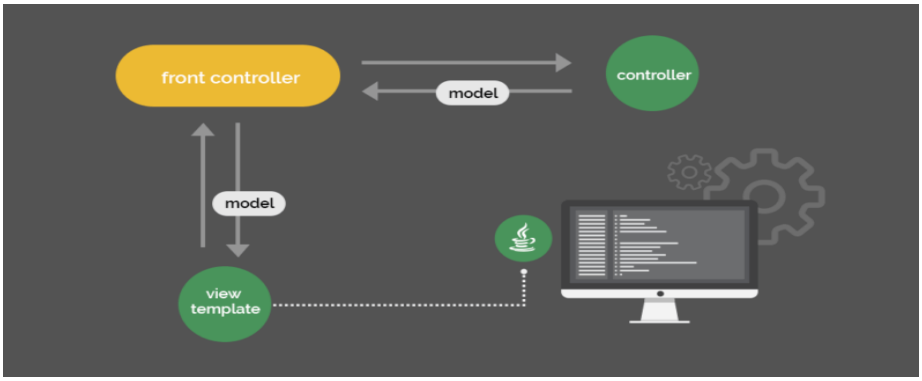


Figure 17: Spring MVC framework architecture [17]

The Controller just needs to handle business logic, connect database, retrieve data or update. It delegates view mapping, annotation configuration and resource mapping to front controller.

On other hand, with RESTController, you can build a RESTful API using Spring framework, which can interact with client and other server side by HTTP methods. Nowadays, many huge systems apply Spring MVC with three layers architecture. (**Figure 18**),

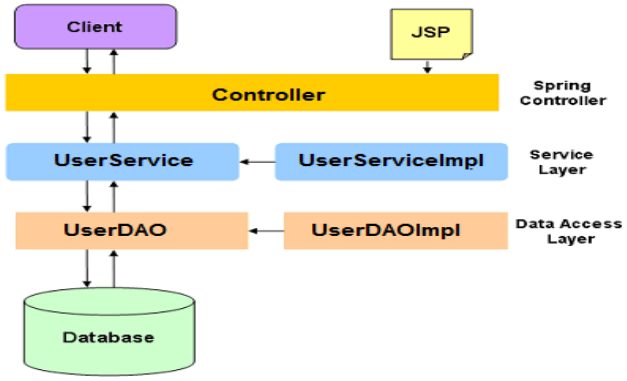


Figure 18: Spring MVC with Three Layer architecture [18]

The reason is to separate the Controller from interact directly with Database or other Data resource. Therefore, we can change the database system or the business logic easily which reduces a lot of risks. Because instead of changing all the code in Controller, we just need to fix the logic in Service or change the way to interact data resource in Data Access Object (DAO).

[17] https://stackify.com/spring-mvc/

[18] https://www.pinterest.com/pin/65724475790180115/

2.3. Hibernate

Hibernate has a layered architecture which helps the user to operate without having to know the underlying APIs. Hibernate makes use of the database and configuration data to provide persistence services (and persistent objects) to the application. [23]

The process of an application using Hibernate framework has to follow the step in **figure 19** below.

Hibernate configuration contains database information which helps you to change the database system dynamically.

Session Factory configures Hibernate for the application using the supplied configuration file and allows a Session object to be instantiated.

A Session is used to connect directly to the database system. Each unit of work with the database is represented by a Transaction. Each time you need an interaction with the database, the session object is instantiated then helps you update and retrieving data from database system.

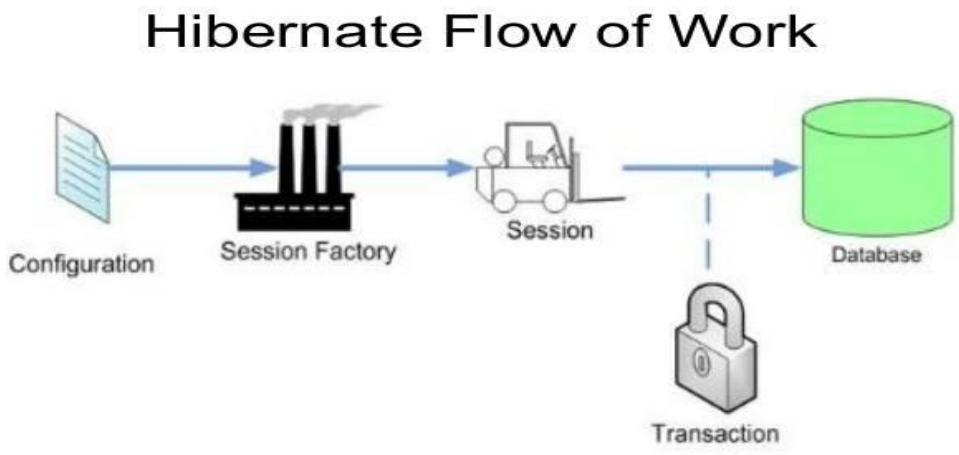


Figure 19: Hibernate architecture [19]

[19] https://www.dineshonjava.com/hibernate/hibernate-architecture/

2.4. Angular

Angular is one of the most popular Javascript frontend frameworks. There are 8 main building blocks of an Angular application in this architecture diagram: Modules, Components, Templates, Metadata, Data binding, Directives, Services and Dependency injection. The way they communicate is clearly shown in **figure 20** from Angular home page.

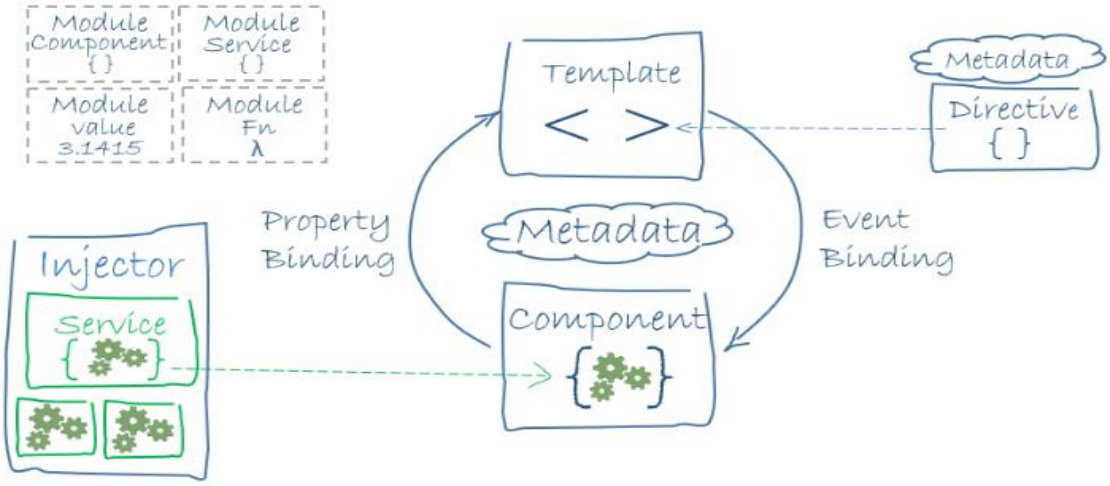


Figure 20: Angular 2 architecture [20]

In each Angular application, there must be one or more Angular module class. The root module is always available in every Angular app. [7]

A component controls the view. We define the application logic in class of the component. That can be fields or functions which support the view. The class and the view interact with each other through an API of properties and methods. We define the view of a component by the template. We can write html code or put html code in the html file to build the template that tells Angular how to render the component Metadata tells Angular how to process a class. Metadata make the class become a component. [7]

Angular supports data binding for coordinating parts of a component with parts of a template. Data binding is added to the template HTML that tell Angular how to connect both sides [7]

[20] https://angular.io/guide/architecture

The directive is also a class, in directive, we will attach the metadata by the @Directive decorator. [7]

Service is a class, we use service as a category contains any feature, function, value or what we need to use in our application. [7]

In Angular, we provide new components with the services they need by using dependency injection. Therefore, Angular can tell the components that the types of their constructor parameters may include the services they need. [7]

2.5. RESTful Web Service

Figure 21 show that RESTful Web Service architecture is based on HTTP methods which the REST Web Service server provides for clients.

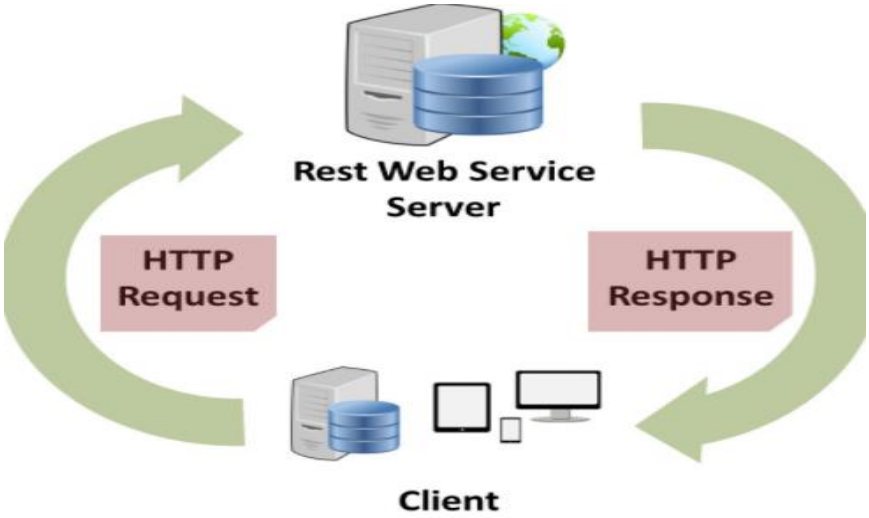


Figure 21: RESTful Web Service architecture [21]

RESTful Web Services are Web services based on REST Architecture. They use HTTP methods to implement the concept of REST architecture. URI is usually a service that a RESTful Web Service provides resource such as Text, JSON and XML. RESTFul Web service provides API as HTTP methods: GET, POST, PUT, PATCH, DELETE which allows other party to call these methods to retrieve data (as JSON, xml or other formats), create, update, modify or delete the data store in database system. [24]

[21] http://www.cousinsinfotech.com/restful-services/

There are five most commonly used HTTP methods in a REST based architecture is shown in **Table 2** below:

HTTP Verb	CRUD	Entire Collection (e.g. /rooms)	Specific Item (e.g. /rooms/{id})
POST	Create	201 (Created), 'Location' header with link to /rooms/{id} containing new ID.	404 (Not Found), 409 (Conflict) if resource already exists.
GET	Read	200 (OK), list of rooms. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single room. 404 (Not Found), if ID not found or invalid.
PUT	Update/ Replace	405 (Method Not Allowed), unless you want to update/replace every resource in the entire collection.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
PATCH	Update/ Modify	405 (Method Not Allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method Not Allowed), unless you want to delete the whole collection—not often desirable.	200 (OK). 404 (Not Found), if ID not found or invalid.

Table 2: RESTful Web Service HTTP methods

2.6. Express framework and Node.js

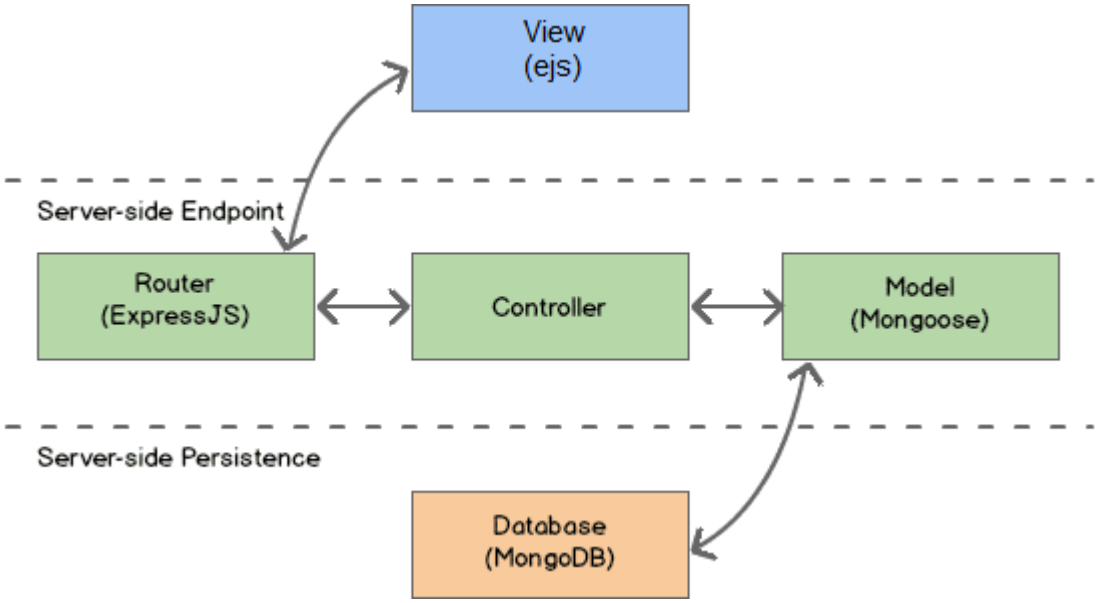


Figure 22: Express MVC architecture [22]

Node.js is an open source framework uses JavaScript on the server side and Express is the most popular framework for Node.js. The architecture of an express Node.js application should contains 4 main building blocks and based on MVC partten: Model, View, Controller and last but not least the Router. (**Figure 22**)

ExpressJS support Routers to separate the controller and the view. The Routers interact with the view and handle all request mapping and response from the view, they also support building a RESTful API which also provide API to communicate with client side or other party server. Besides, with Mongoose library, the Model in

Node.js can retrieve data or update to MongoDB or another library to interact with other database system. In express framework, Controller uses Model as the data types and interaction with the database as well. Controller also provides data and functions for the routers to render the view (EJS). Router receive request from view and call controller's functions to update database. On the other hand, router provides API for View (Client side) by building a RESTful Web Service

[22] https://dzone.com/articles/using-mongodb-and-mongoose

2.7. MEAN stack technology

The term MEAN stack refers to a collection of JavaScript based technologies used to develop web applications. MEAN is an acronym for MongoDB, ExpressJS, Angular and Node.js. From client to server to database, MEAN is full stack JavaScript.

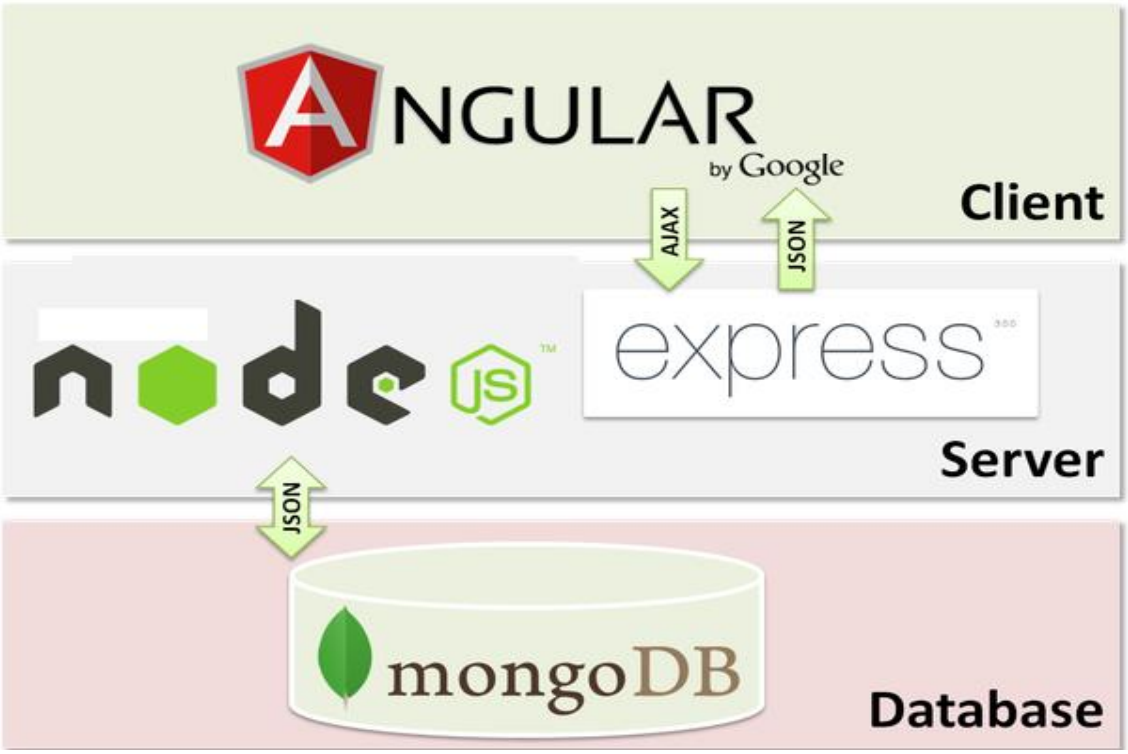


Figure 23: MEAN stack technology architecture [23]

Node.js and Express framework are on server side, Angular is on client side and MongoDB is a database system that stores data in JSON (JavaScript Object Notation). Angular on client side gets data from Node.js & express framework on server side in JSON format which may retrieve from MongoDB to represent the view. In contrast, Node.js & express framework receive AJAX request from Angular to execute business logic and update to MongoDB. (**Figure 23**)

[23] https://www.quora.com/What-are-the-advantages-of-developing-with-the-MEAN-stack-mongoDB-Express-js-Angular-js-Node-js

2.8. Microservice

The architecture of Mircoservice is the interaction of many independent services. Each service has the own architecture which can be implemented in different programming language or use different database system. (**figure 24**)

In Microservice, a service provides API or uses API from other services to execute its own business logic. A service can also be a client with variety of difference from other services including desktop browsers, mobile browsers and native mobile applications. [25]

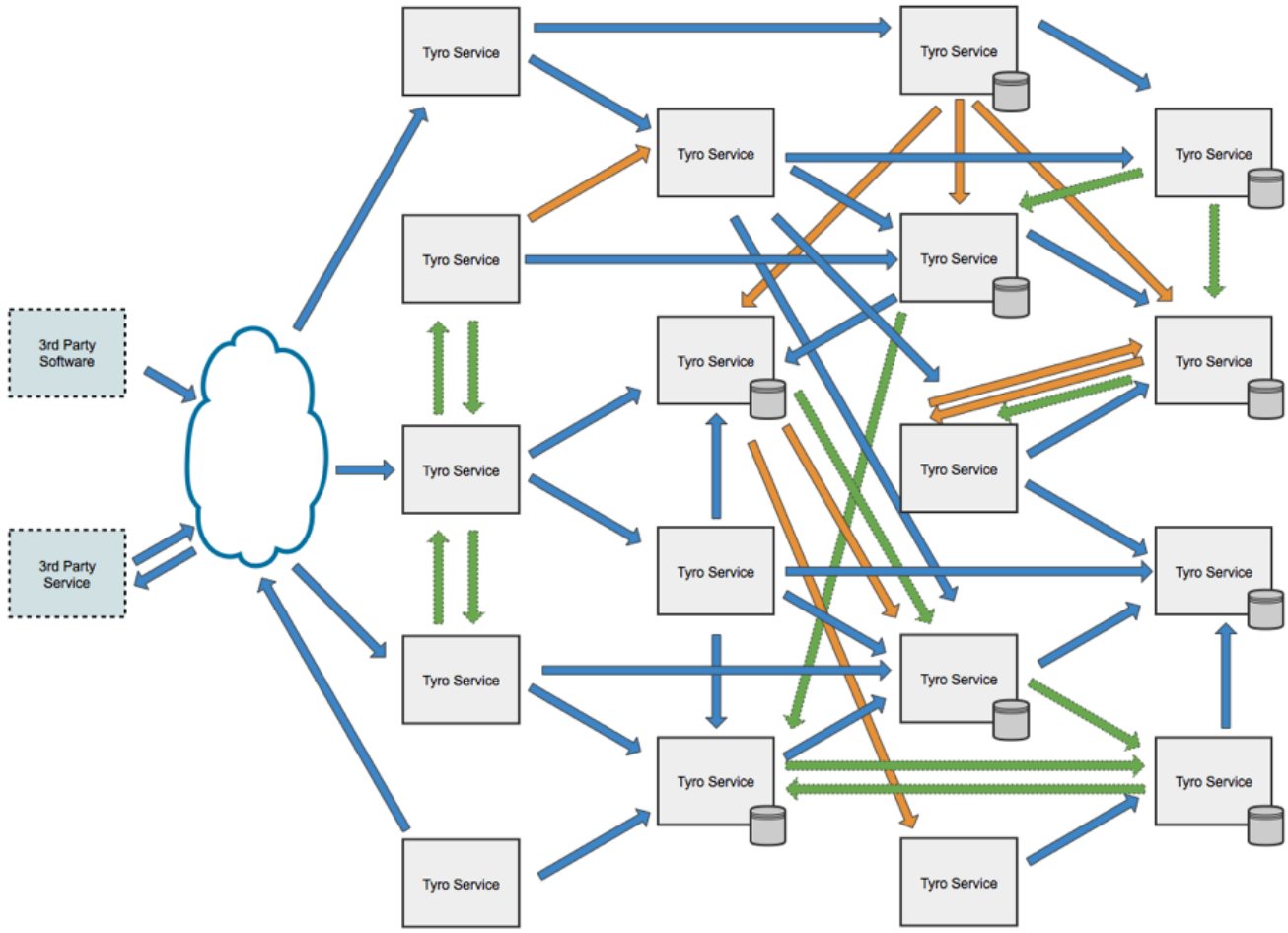


Figure 24: Mircoservice architecture [24]

[24] https://softwareengineering.stackexchange.com/questions/339817/should-services-talk-directly-to-each-other-in-a-microservice-architecture

Chapter III. Software Requirement

3.1. System Overview

My Hotel Bookings & Reservations System is a web application running on 2 servers. The purpose of running 2 servers at the same time is that each server doesn't have to do a lot of job and they communicate through API. Because of API, these 2 servers can work independently which means I can apply Microservice architecture to my system.

With Microservice architecture, I can separate my system into many different modules. And each module is independent which means I can choose programing language that best fits for my module and I can also pick which kind of database system that fits with my module, It is the reason why I used MongoDB with Angular, Express & Node.js and SQL with Spring MVC Hibernate.

Therefore, the system architecture is separate into 2 main architecture includes MEAN stack technology and Spring MVC Hibernate. With MEAN stack technology for Customer's website, it becomes an online single page application with high performance running on server built by Node.js and express framework which focus on RESTFULL web service architecture and Angular 2 on client side. This website support dynamically loading with a lot of features for customer as well as provides tracking users feature for Administrator system.

The second main system architecture run on server which was built and deployed by Java and Spring MVC Hibernate. It is a cross-platform flexible and loosely coupled web application runs well on all operating system.

The client side is the combination of HTML5, CSS3, Bootstrap and Angular that provides the very friendly user interface make it easy to use. Hence, the users will feel convenient and comfortable when using this application with about 64 primary feature and hundreds of small features.

3.2. Feature

As I have mentioned above, my application supports almost features for hotel bookings as well as reservations management with ability to track user's behavior and it will provide data collection for applying AI machine learning in the future.

Based on my system's requirement, I divide it into 50 main features with 3 primary roles: Guest, Customer and Administrator.

The decentralization of each role will be clarified in the sections below:

3.2.1. Guests

The guest can view introduction and gallery of the hotel, send reservation form, contact with administrators, view, search the rooms or the items in the restaurant which they would like to see more details or register an account to become a customer.

3.2.2. Customer

The customers can do what the guests can do. Moreover, they can login to the system to book room or cancel it, rate the room, send feedback, check profile, view transaction history. With data collection feature, customers were tracked; therefore, the system can suggest the recommendation rooms for the customers.

3.2.3. Administrator

The admin can login to the website and go to their dashboard to manage the hotel, check his profile, add, update and delete rooms or other services in the restaurant, receive the request of customers and reply them with several available email templates. Admin can also manage the users, view information and activity of users or ban them if they did something unacceptably.

Thank to follow-users feature, administrator is able to see which page customers clicked, how long they stayed in each page, which keyword they used to search, which image they used to click on. Admin can view the chart with the statistics of visitor from country as well.

No.	Feature	User	Description
1	Register	Guest	The guests can create new account
2	Login	Customer, Admin	Customers, Admins can login to use more feature
3	Logout	Customer, Admin	Customers, Admins can logout
4	View Rooms	Guest, Customer, Admin	Customers, Admins can view list of rooms in hotel
5	View Restaurant	Guest, Customer, Admin	Customers, Admins can view list of restaurant items in hotel
6	Search for Room	Guest, Customer, Admin	Customers, Admins can input to search for rooms
7	Search for Restaurant Item	Guest, Customer, Admin	Customers, Admins can input to search for restaurant items
8	View gallery of hotel	Guest, Customer	Guests, Customers can view image gallery of hotel
9	View introduction of hotel	Guest, Customer	Guests, Customers can view introduction of hotel
10	Filer rooms	Guest, Customer, Admin	Guests, Customers, Admins can filter the list of rooms by their attribute
11	Filer food or drink	Guest, Customer, Admin	Guests, Customers, Admins can filter the list of restaurant items by their attribute
12	Send contact	Guest, Customer	Guests, Customers can send contact form to administrator
13	Send reservation form	Guest, Customer	Guests, Customers can send reservation of booking form to administrator
14	Book room	Customer	Customers can book room using account balance
15	Cancel room	Customer	Customers can cancel rooms they booked
16	View profile	Customer, Admin	Customers, Admins can view personal information in profile page

Table 3: List of features - 1

No.	Feature	User	Description
17	Edit profile	Customer, Admin	Customers, Admins can edit personal information in profile page
18	Change password	Customer, Admin	Customers, Admins can change password
19	View activity	Customer	Customers can view their transaction history with all of their activities
20	Send feedback	Customer	Customers can send feedback with rating score and comment
21	Dashboard management	Admin	Admins can view dashboard page with chart and notification
22	Receive notification	Admin	Admins can receive notification from customer and guest
23	Send message	Admin	Admins can send message via email for customer
24	View users	Admin	Admins can view user information
25	Manage users	Admin	Admins can manage what user have done on the website
26	Ban users	Admin	Admins block the users
27	Add new room	Admin	Admins can add new room to database system
28	Delete room	Admin	Admins can delete room from database system
29	Update room	Admin	Admins can edit room
30	Add food or drink	Admin	Admins can add new restaurant item to database system
31	Remove food or drink	Admin	Admins can add remove restaurant item from database system
32	Update food or drink	Admin	Admins can change restaurant item information
33	Update profile image	Admin	Admins can change profile image
34	Follow users	Admin	Admins can view the tracked data of user
35	Send feedback & rate hotel	Customer	Customers can send feedback with rating and comment about hotel service
36	Send feedback & rate room	Customer	Customers can send feedback with rating and comment about room
37	View customer Data collection	Admin	Admins can view customer transaction history as statistics data
38	View recommendation room	Guest, Customer	Guests and Customers can view the room they should book which is suggest automatically by system
39	View related rooms	Admin	Admins can view related rooms when they stay on view details, or edit room page

Table 4: List of features - 2

No.	Feature	User	Description
40	View related restaurant items	Admin	Admins can view related restaurant items when they stay on view details, or edit restaurant item page
41	View top of rooms	Guest, Customer	Guests, Customers can view list of rooms which automatically sort by the highest rating score
42	Email template	Admin	Admins have some email templates to send an email easily and quickly.
43	View User manual file	Admin	Admins can view user manual file as PDF file on website
44	Upload User manual file	Admin	Admins can upload user manual file as PDF file to web server
45	Download User manual file	Customer	Admins can download user manual file as PDF file from web server
46	Download Customer's Data Tracking file	Customer	Admins can download all tracking data information include IP address, page access, the time customer stay in each page, date visited … as PDF file from web server
47	Filter Table Management	Admin	Admins can filter the management tables by their columns.
48	Forget Password	Customer, Admin	Admins, Customers can input email for receiving an email to change the password
49	Notification Email	Customer	Customers can receive an email every time they have any transaction on website.
50	Upload Image of rooms	Admin	Admins can upload 2 images for each room
51	Upload Image of restaurant items	Admin	Admins can upload 2 images for each restaurant items
52	View pie chart of country and visit time	Admin	Admins can view pie chart with statistics from all the countries of uses and the visited time
53	View column chart of page access and visit time	Admin	Admins can view column chart with statistics from the page that users access and the visited time
54	View column chart of user IP address and visit time	Admin	Admins can view column chart with statistics from the user's IP address and the visited time

Table 5: List of features – 3

No.	Feature	User	Description
55	View column chart of username and visit time	Admin	Admins can view column chart with statistics from the username and the visited time
56	View statistics and chart of all data tracking by the system	Admin	Admins can view statistics and column chart with all user tracking data
57	Download country chart	Admin	Admins download country chart as JPG, PNG, SGV and PDF
58	Download page access chart	Admin	Admins download page access chart as JPG, PNG, SGV and PDF
59	Export page access chart	Admin	Admins can export page access chart data as CSV, XLSX, JSON
60	Export country chart	Admin	Admins can export country chart data as CSV, XLSX, JSON
61	Annotate, Draw in country chart	Admin	Admins annotate and draw on country chart to print or download a new one.
62	Annotate, Draw in page access chart	Admin	Admins annotate and draw on page access chart to print or download a new one.
63	Print Chart	Admin	Admins can print the charts
64	View geo location based on Google map	Admin	Admins can view the locations of users which contain country, city, portal code, region code, latitude, longitude … of users who used or are using the system.

Table 6: List of features – 4

Recently, my system has already maintained 64 primary features. As you can see **from table 3 to table 6**, these table have shown the list of primary features in my system for each role and their description, each role can use many features.

3.3. Use Case

Based on the feature above, I have a use case diagram as **figure 25** below with three actors using the system: guest, customer and administrator

Figure 25: Use Case Diagram

3.4. User Story

According to the Use case diagram, it will drop down into 25 user stories as below:

- As a guest, I can register a new account so that I can login to the system

- As a guest, I can view the rooms so that I can see the details of the rooms, watch the image of the rooms.

- As a guest, I can view the food or drink in the restaurant of the hotel so that I can see the details, watch the images of each item in the restaurant.

- As a guest, I can view introduction and gallery page so that I can see the information of the hotels and watch the image gallery of the hotel.

- As a guest, I can send contact to the administrator so that I can write what I want to communicate with him and wait for his response.
- As a guest, I can view the recommendation rooms so that I can see which room that the system automatically suggests me book.

- As a customer, I can login to the system or logout so that I can use more features.

- As a customer, I can edit my profile so that I can change my personal information.

- As a customer, I can book room so that when I come to the hotel, this room belongs to me,

- As a customer, I can send a feedback about a room or about the whole hotel services so that I can rate the star of service and comment or complaint my opinion.

- As a customer, I can view my activity so that I can see the transaction history, what I have done, what I interacted with the hotel.

- As an administrator, I can login to the system or logout so that I can use admin features.

- As an administrator, I can edit my profile so that I can change my personal information.

- As an administrator, I can manage the rooms so that I can view the rooms, add a new room, edit a room or delete it

- As an administrator, I can manage the items in restaurant so that I can view the items, add a new item, edit an item or delete it.

- As an administrator, I can manage users so that I can view user information, view what they interacted with hotel or delete a user from database.

- As an administrator, I can view my messages and notifications that the guests or customers send to me so that I can interact with them and reply their message.

- As an administrator, I can follow user's behavior so that I can see what they clicked, what they searched, what they did on the website.

- As an administrator, I can view the visitor chart from country so that I can easily compare which is the most visited country, which is the less visited country and another.

- As an administrator, I can view the page access chart based on all IP address or single IP address so that I can easily compare which is the most visited page, which is the less visited page and another.

- As an administrator, I can receive the message, the booking request, cancel room request and feedback of the customers so that I can view the information that they send to me and reply them by myself or using some available email templates

These user stories above have clarified the use case and some primary features of my system. During the whole time I built and implemented those features, I need to use and apply many technologies. In next chapter, I will talk more about technology and the architecture of my hotel booking system.

Chapter VI. Methodology

4.1. All Technologies used

As I mentioned above, for Server side, I used Java web J2EE with Spring MVC and Hibernate Framework for Administrator sites. On another hand, I used Node.js & Express Framework to develop server for Customer sites. I also used HTML5, CSS3, Javascript, JQuery, Boostrap with AngularJS & Angular 2 Framework for client side.

Moreover, I used 2 kind of database system: MongoDB (NoSQL) and SQL. Because of Hibernate, I can also test running the system on 2 different SQL database system: Mircosoft SQL Server and MySQL without changing the query.

Besides, I used UMLET and Edraw to design UML and draw diagram. For coding Node.js, Java and Angular, I some need IDE tool such as VSCode, Eclipse, Netbeans.

To design and code front-end with HTML, CSS and JS, I used Adobe Dreamweaver CS6.

To build and deploy the server on local host, I used maven with tomcat or glassfish for java server and node_modules with npm for Node.js server

For running server online, I used heroku to deploy Node.js and Java application; Mlab for MongoDB and freesqldatabase for SQL database system.

So far, I would like to list some other tool, software, framework or library that supported me to build my system:

Code review and analysis: Sonar Lint

Version Control: Git hub

Project management: Trello

Some Library: gson, geoip, mail service, jackson, external IP and so much more.

4.2. System Architecture

My system includes two main architectures: one for Customer applied MEAN stack technology and one for Administrator built in AngularJS with Spring MVC Hibernate.

They are built and deployed in two different servers and communicate through API.

The system architecture includes MVC architecture and RESTful Web Service with Angular Architecture

4.2.1. MVC architecture

4.2.1.1. Node.js & Express framework MVC

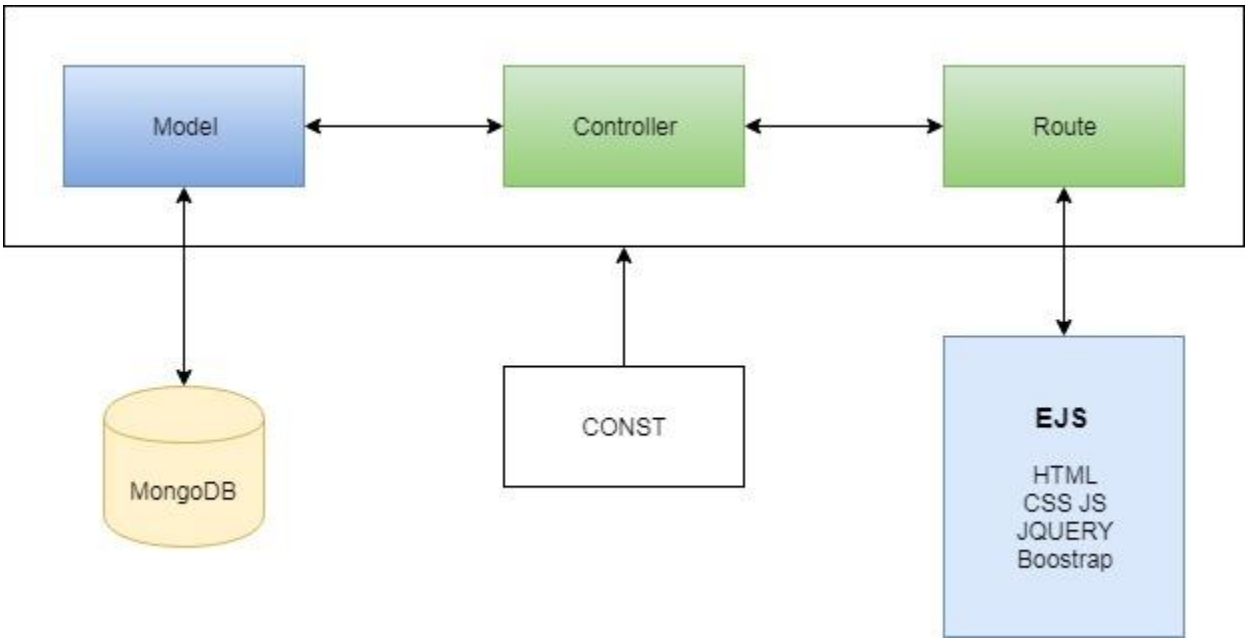


Figure 26: Node.js and Express MVC framework system architecture

This architecture (**figure 26**) observes the MVC architecture's rule with Node.js & Express framework on server side. The request of customer from view EJS to Routes which call the functions from Controller to delegate Models interact with MongoDB to retrieve or update the database and then send response backward from Model to Controller to Route and display by EJS for Customer. The CONST provide constant variables for the whole system. [26]

4.2.1.2. Spring MVC Hibernate

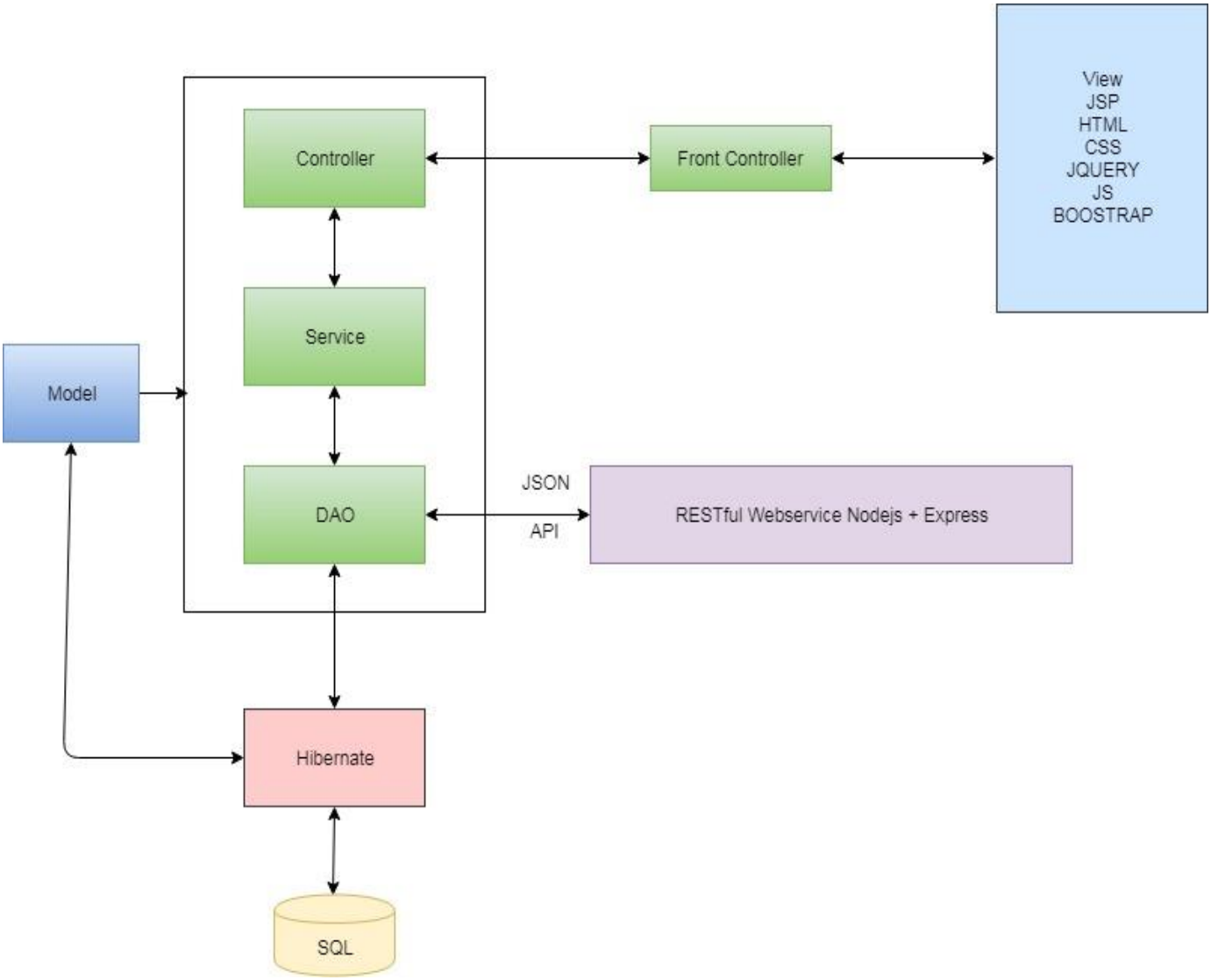


Figure 27: Spring MVC Hibernate system architecture

This is another MVC architecture build by Spring MVC and Hibernate framework on server side. (**figure 27**)

The request of Administrator from JSP to FrontController which handles the view mapping before sending to Controller. After that, the Controller will call method from Services where use the methods from DAOs to connect database. The Models provide data-type for Controller, Services, DAOs (see **section 2.2** for more information about controller, service and DAO). [27]

Hibernate maps Models and DAOs to persist SQL database system.

The Helper class provides some additional functions; the AppConst and APIConst provide constant application data and constant API URL link as public static final variables for the whole system. Similar to request process, the response process will execute in opposite direction.

4.2.2. RESTful Web Service architecture

4.2.2.1. Node.js and Express framework RESTful Web Service

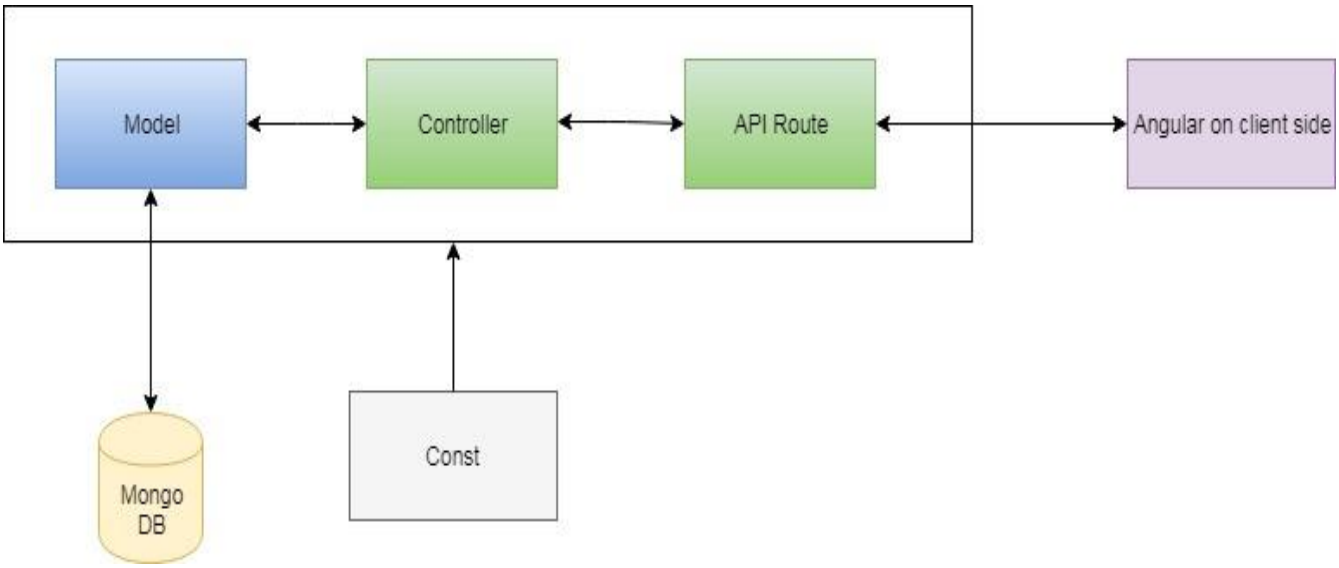


Figure 28: Node.js and Express framework RESTful Web Service system architecture

Based on Node.js & Express MVC architecture above, this architecture (**figure 28**) has the same structure but it is the web service and it has no view. Therefore, the routes became the API routes where provide API as HTTP method for Angular on client side or other application instead. [28]

4.2.2.2. Spring Hibernate RESTful Web Service

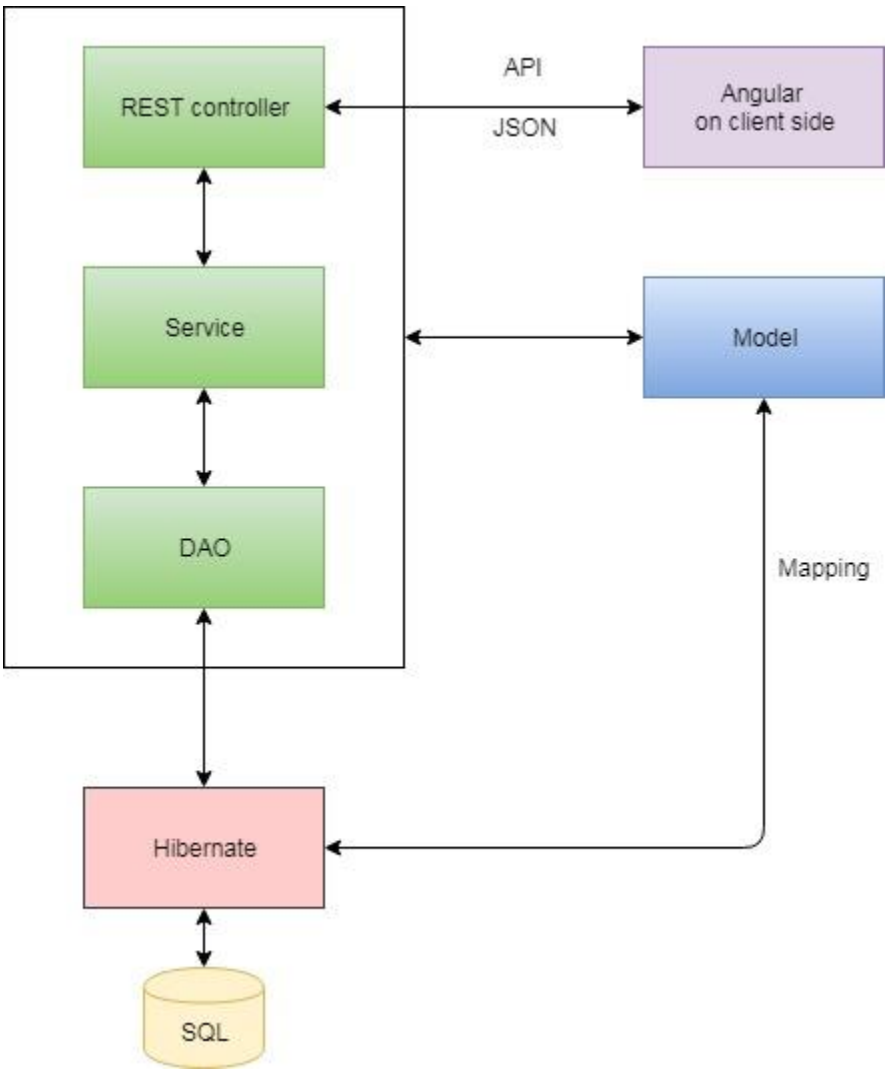


Figure 29: Spring Hibernate RESTful Web Service system architecture

This architecture (**figure 29**) follow the Spring MVC Hibernate structure and similar to the REST architecture above. It is a web service that provides API using RESTController. [29]

4.2.2.3. Spring RESTful Web Service interact other RESTful Web Service

The architecture in **figure 30** below is RESTful Web Service architecture which is the combination of Spring RESTful Web Service and ability to interact with other API provider. In this case, the DAO communicate with Node.js & Express RESTful Web Service in **section 4.2.1.2** to provide data as well as update to MongoDB database system. [30]

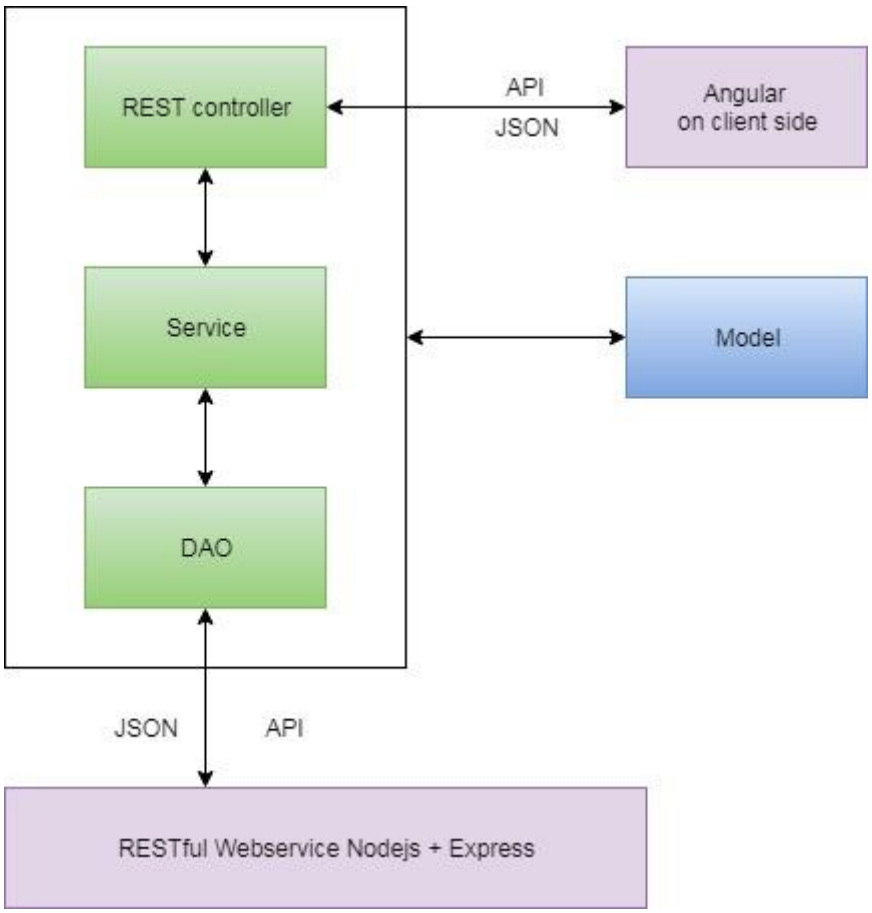


Figure 30: Spring Hibernate RESTful Web Service interacts other REST API system architecture

4.2.3. Angular on client side architecture

In my system, Angular can communicate directly with the RESTful Web Services above through API.

4.2.3.1. AngularJS architecture

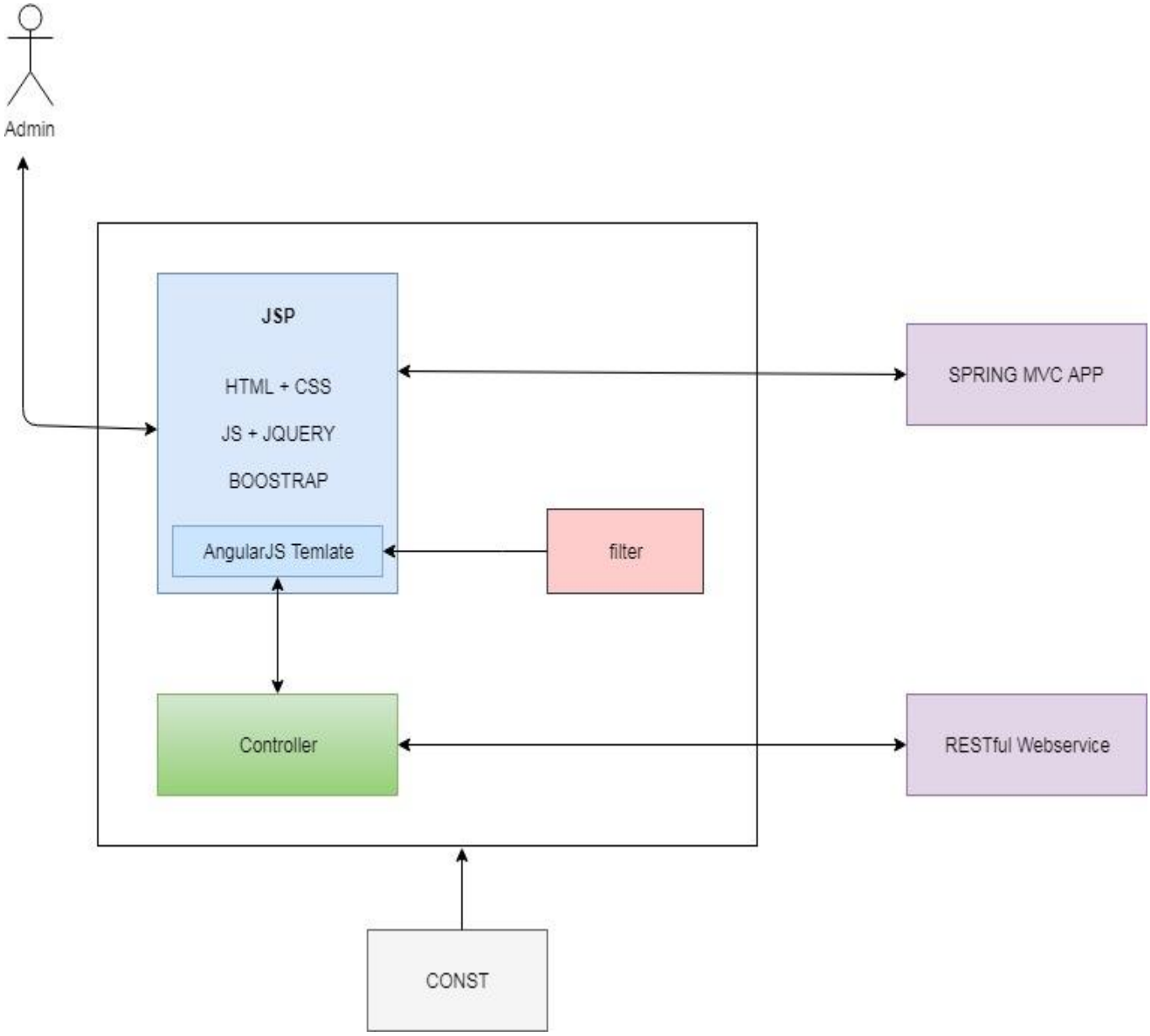


Figure 31: AngularJS system architecture

According to **figure 31**, request of Admin from JSP files where includes HTML, CSS, JS, JQuery, Bootstrap and AngularJS is controlled by Spring MVC App.

Besides, with AngularJS on client side, the data binding mechanical can be used between AngularJS Templates and Controllers where contain functions to interact with RESTful Web Service.

Moreover, the AngularJS filters provide the methods to transform data in the template and AppConst provides constant variables for the whole client-side with AngularJS application.

4.2.3.2. Angular 2 architecture

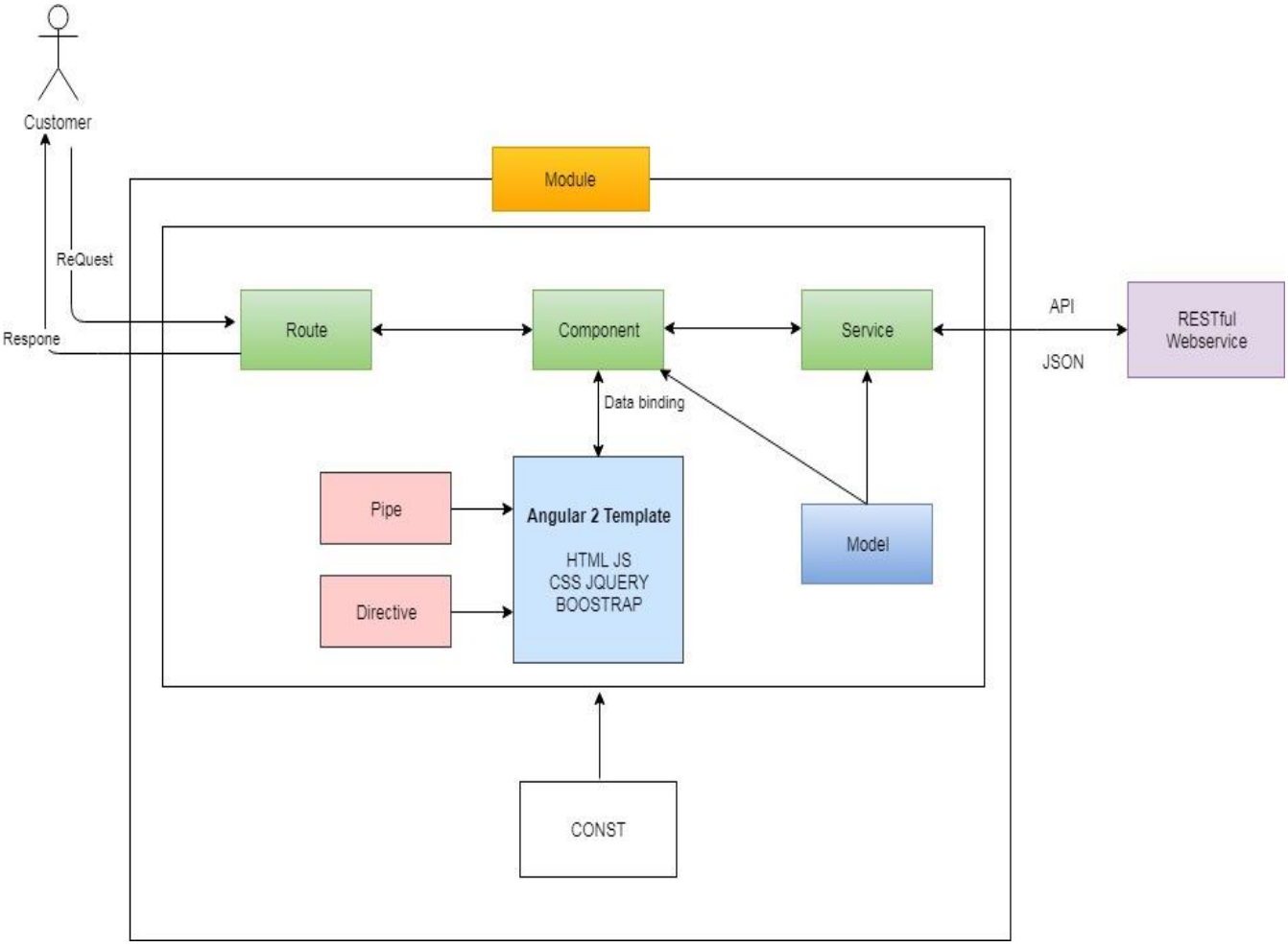


Figure 32: Angular2 system architecture

Arcoding to the **figure 32**, when the customer sends request to Angular 2 app, it first handles the view mapping by Routes then they will send exactly this request to component.

Similar to AngularJS but the controllers are removed and changed to Component, the data binding mechanical now occurs between Component and Template. [31]

Services take responsibility for communication with RESTful Web Service through HTTP methods. Directive supports change the structure of the view, it also help change the style of elements and so much more. [31]

The CONST and Pipes work the same as CONST and Filters in AngularJS architecture in **section 4.2.3.1**.

4.2.4. The architecture of the whole system

In this section, I will show you the combination of all architectures from **section 4.2.1.1 to 4.2.3.2**

Based on those system architectures above, you can easily see that on the client sides, Angular communicate with the Servers through API.

The server built by Node.js and Express framework can connect to MongoDB and the second Spring MVC server connect to SQL to store or provide data for client through API. [32]

Moreover, two servers can also interact with each other by their own provided API. That is one of the most popular architecture in many big companies today called Mircoservice. [30]

The **figure 33** below will make clear how these services communicate with each other.

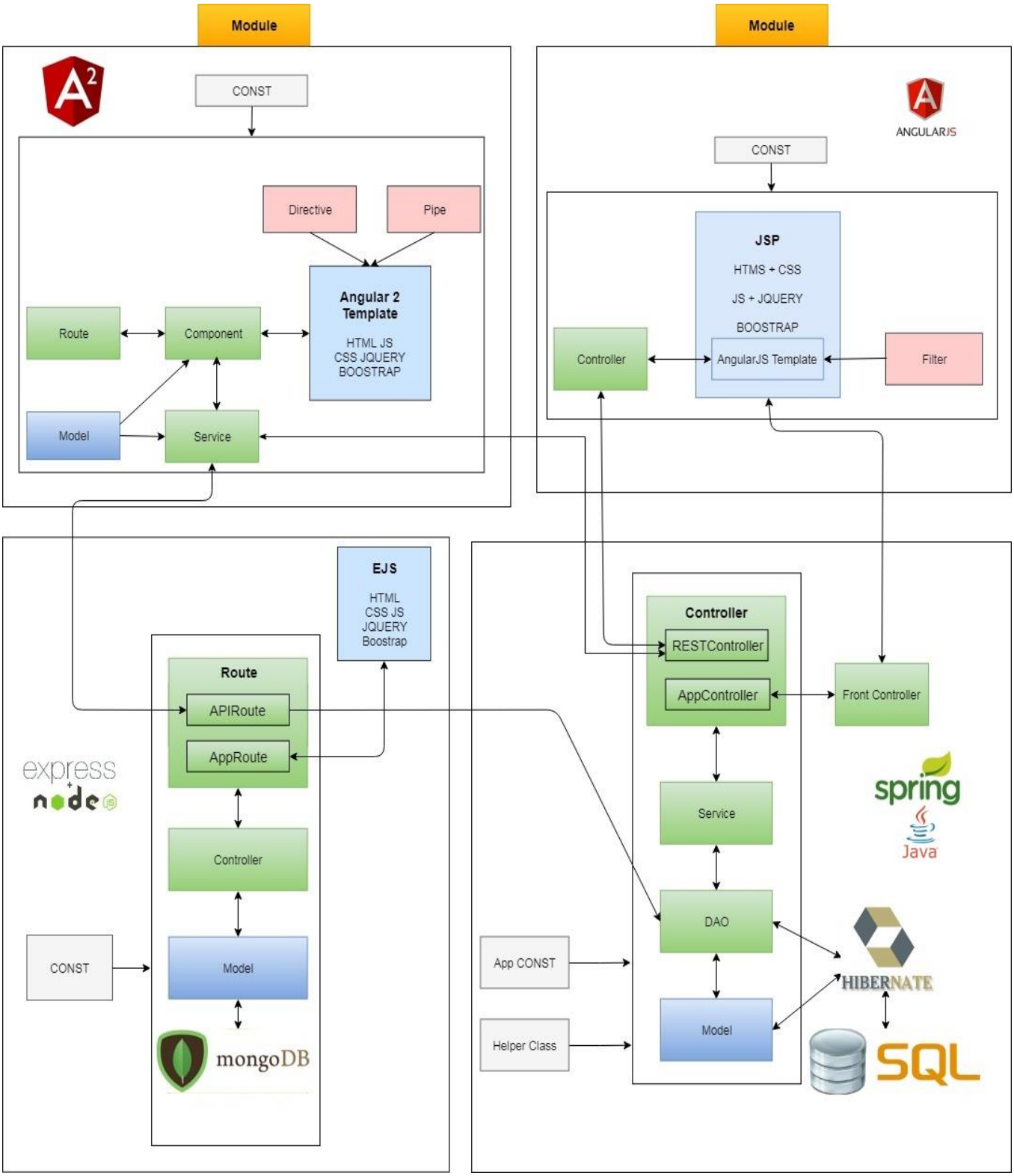


Figure 33: The architecture of the whole system

4.3. Database system

With three actors who use the system showed in use case (see in **section 3.3**) and based on two main servers for Guest, Customer and Admin. I separate my database into two different systems: MongoDB and SQL.

4.3.1. MongoDB collection & document

For MongoDB database, there are 4 collections include customer, activity, tracking and suggestion-data. (**figure 34**)

Collection Customer contains Customer personal information, username & password

Collection Activity contains some activity of customer. It is also the transaction history where includes some transactions such as: create new account, book room, cancel room, send contact, send reservation form.

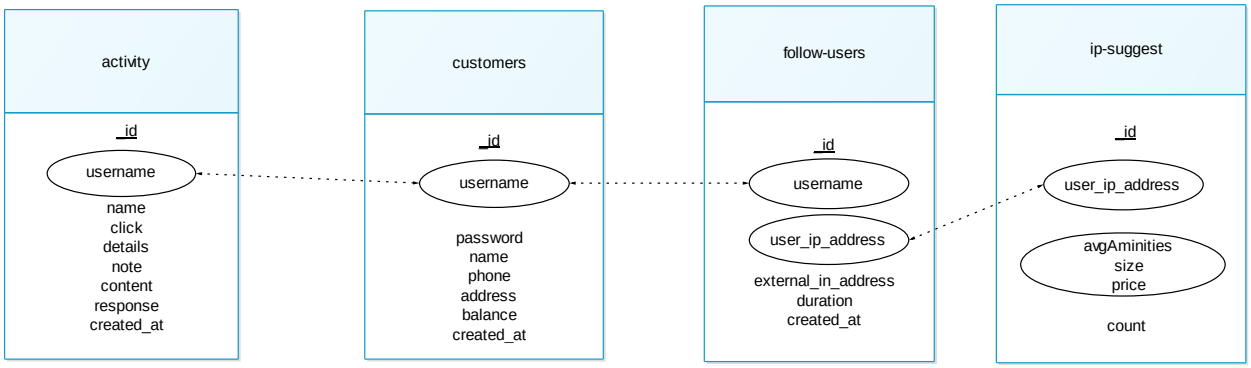


Figure 34: MongoDB collections

Collection Tracking contains tracking data of user such as the IP address, the external IP, the page user clicked and the duration that user stays in this page.

Collection suggestion-data contains the data to recommend room for user. The data include the average size, average price, average amenity of the rooms that customer view, click, book or search.

4.3.2. SQL ERD (Entity Relationship Diagram)

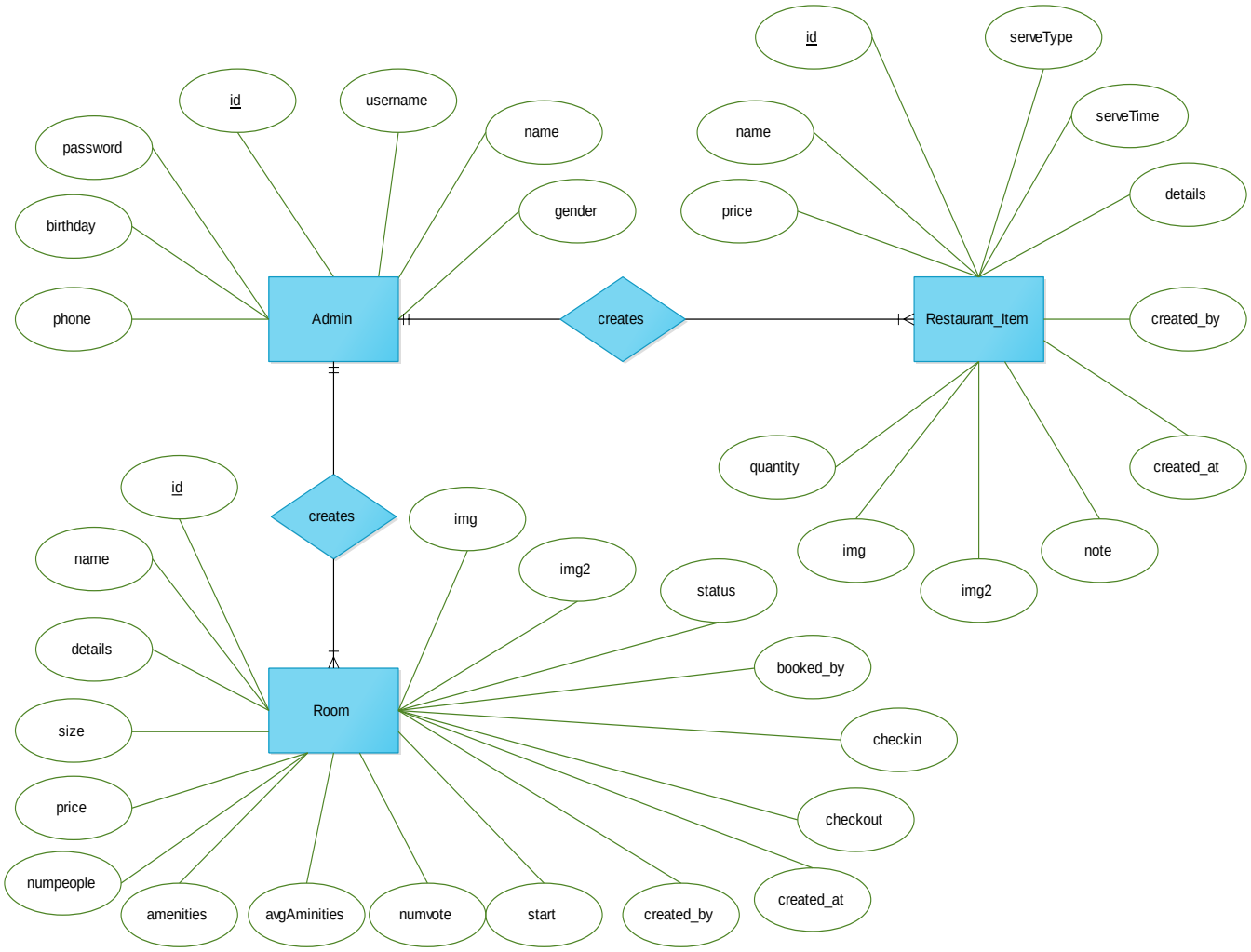


Figure 35: SQL Entity Relationship diagram

According to the entity relationship diagram in **figure 35**, we can easily see all the fields of each table and how they work together.

Table admin contains all administrator information, username and password.

Table Room contains all information of room which booking details includes who booked the room, check in date, checkout date.

Table RestaurtantItem contains all information of the items in restaurant includes food, drink, fruit or ice-cream.

The relationship is that one Admin cans create a lot of Room or lots of Restaurant Items so that in there is a field username in these 2 tables that maps with the username of an Administrator

4.3.3. MongoDB interact with SQL ERD

After combined the collection of MongoDB and table of SQL database system, we have this relationship in **figure 36:**

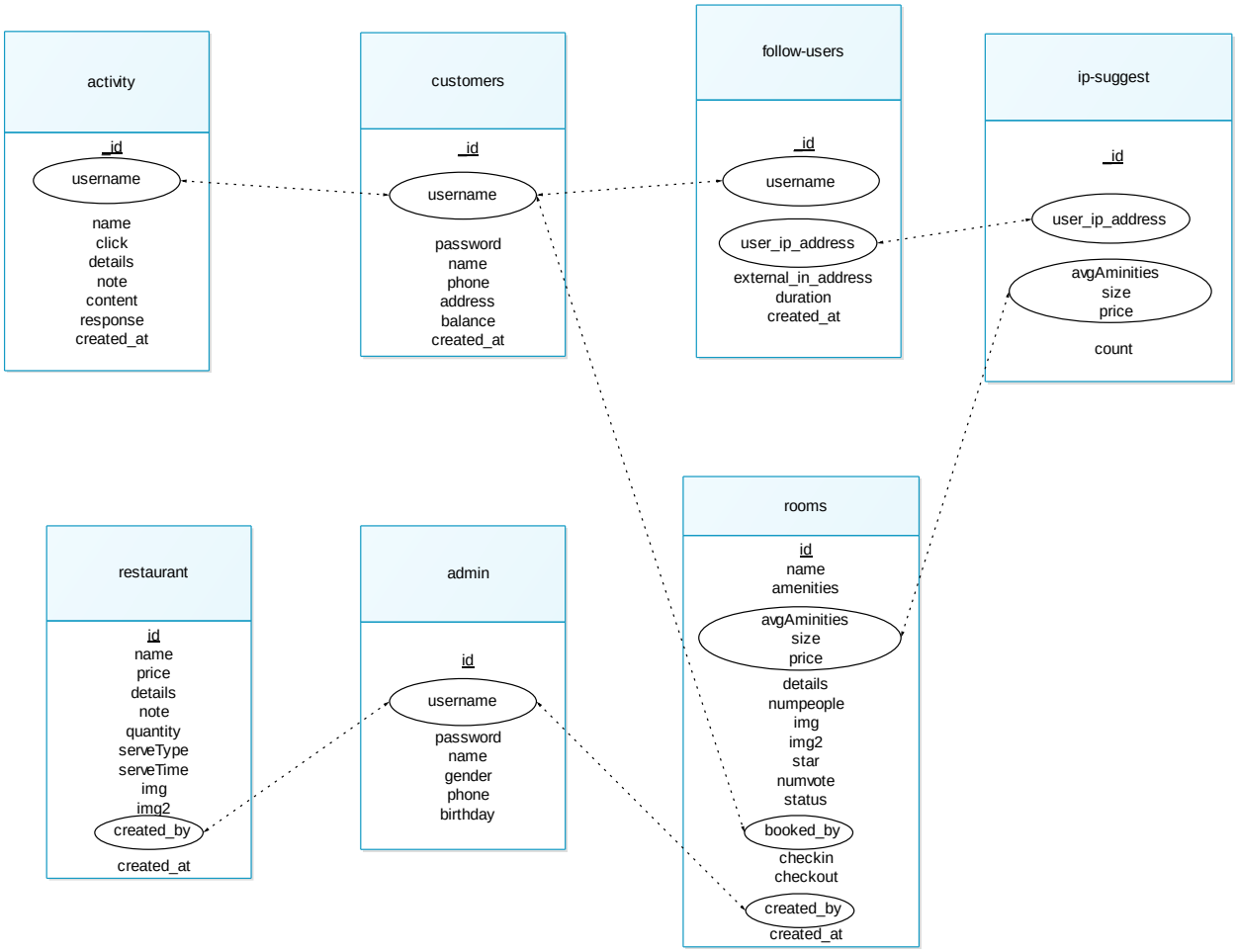


Figure 36: MongoDB collections and SQL Entity Relationship diagram

4.4. Class diagram

Based on these system architectures and database structures above, with Mircoservice architecture and two database systems, it is not easy to draw the class diagram in one page because the system is too big. Therefore, I separated it into smaller class diagrams in **section 4.4** below:

4.4.1. Spring Model class diagram

4.4.1.1. Spring relationship Model class diagram

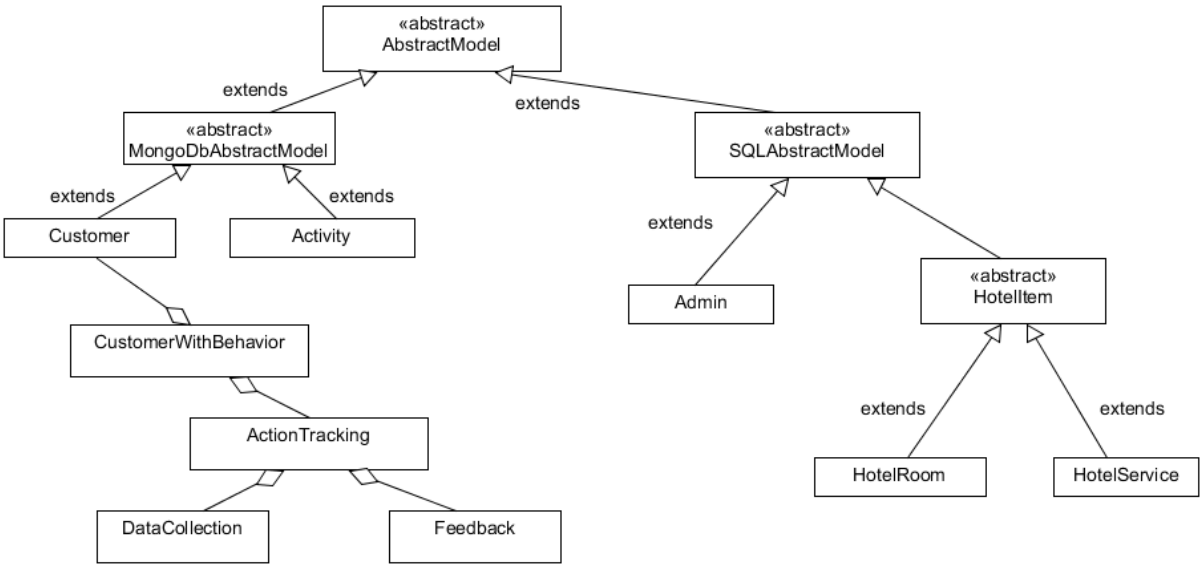


Figure 37: Spring Model class diagram

The class diagrams in **figure 37** represent clearly the models with their relationship. You can see all variables, methods of each class in **Appendix A** section. Then, I just want to describe in short how they work:

The AbstractModel is an abstract class contains some checking function, get date and time in several format and some addition functions for its subclass.

The MongoDBAbtractModel and SQLAbtractModel are two abstracts class extends AbstractModel which represent for 2 kind of database system MongoDB & SQL.

With each database system, I created several corresponding models. For MongoDB, we have Customer, Activity extends MongoDBAbtractModel and some related class includes CustomerWithBehavior, ActionTracking, DataCollection, Feedback.

Class Customer contains all information of customer with username and password.

Class Activity contains all customer's activity includes book room, cancel room, send feedback, rating room, send contact, send reservation form.

Class CustomerWithBehavior owns the Customer class and the list of date customer visited the hotel website. Moreover, this class also contains some addition behavior of customer that store in ActionTracking class which includes average rating point for room and hotel service. the action that tracked by system such as list room booked, list room canceled, list of feedback or list room feedback declared in Feedback class with feedback time, which room, rating and comment.

On another hand, for SQL database system, there are 2 primary main models called Admin and HotelItem. The HotelItem class has 2 subclasses HotelRoom and HotelService. All 4 models are mapped by Hibernate with corresponding tables includes admin, room, restaurant in SQL database.

Class Admin contains all information of admin with username and password.

HotelItem is an abstract class contains some variables that HotelRoom and HotelService should use such as name, size, price, type… and some abstract methods for its subclass to override such as initialize some information, set new information or return the ability to update new data…

HotelRoom class contains all information of room such as amenity, average amenity score, rating score, image, details with some checking method like check valid type, valid status, enough information of room, room is ready to book or not…

HotelService class contains all information of hotel service item. In my case, the hotel service items are only the items sold in hotel restaurant. The information include quantity, note, service type, service time and function for compute the

service time, checking whether a service item has valid type, enough information, correct server type or not.

4.4.1.2. Spring Bean and POJO class diagram

Besides, there are some POJO models without any relationship with other models. They just provide data-type for the Spring application. (**figure 38**)

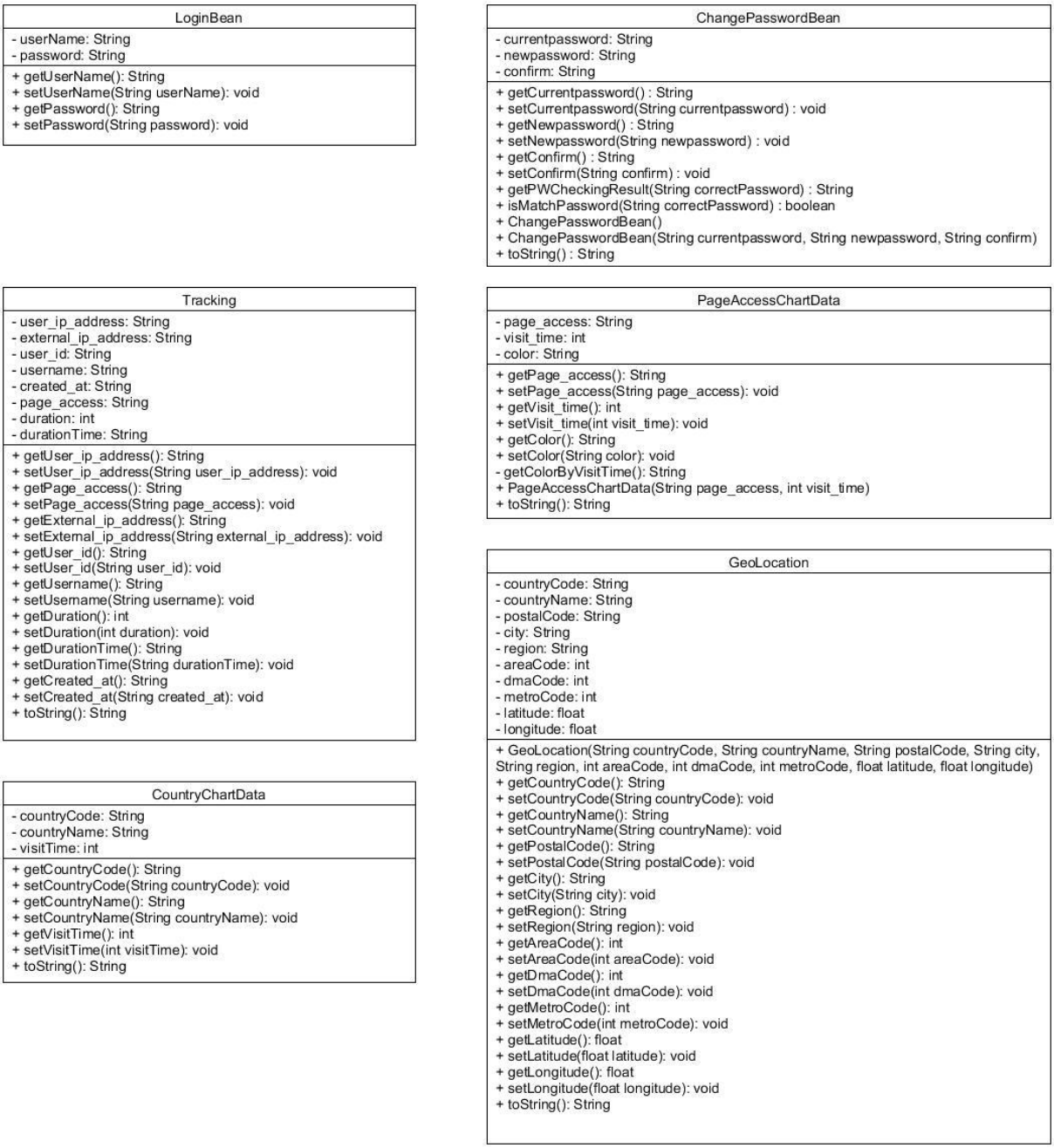


Figure 38: Spring Bean and POJO class diagram

The LoginBean and ChangePasswordBean are two java beans contains data such as username, password, new password, confirm password, current password to support login and change password feature

The Tracking class contains information tracked such as ip address, external ip, username, the page which user accessed and the duration in millisecond format and hh/MM/ss: SSS format

The CountryChartData and PageAccessChartData declare the necessary data to draw chart a country chart and page access chart based on visit time

The GeoLocation class contains tracked information based on geo location which includes country, city, region, postal code, area code, DMA code, metro code and the coordinate of user IP address.

4.4.2. Spring DAO (Data Access Object) class diagram

There are 2 kinds of DAO named Hibernate DAO and API DAO which have the following meaning:

4.4.2.1. Spring Hibernate DAO class diagram

Firstly, Hibernate DAOs are the interfaces that inject the session factory follow Hibernate and Spring framework to interact with SQL database system

The main purpose of them is working with Hibernate and SQL Model (see in **section 4.4.1.1**) to interact with SQL for retrieving data or update then provide methods for Services.

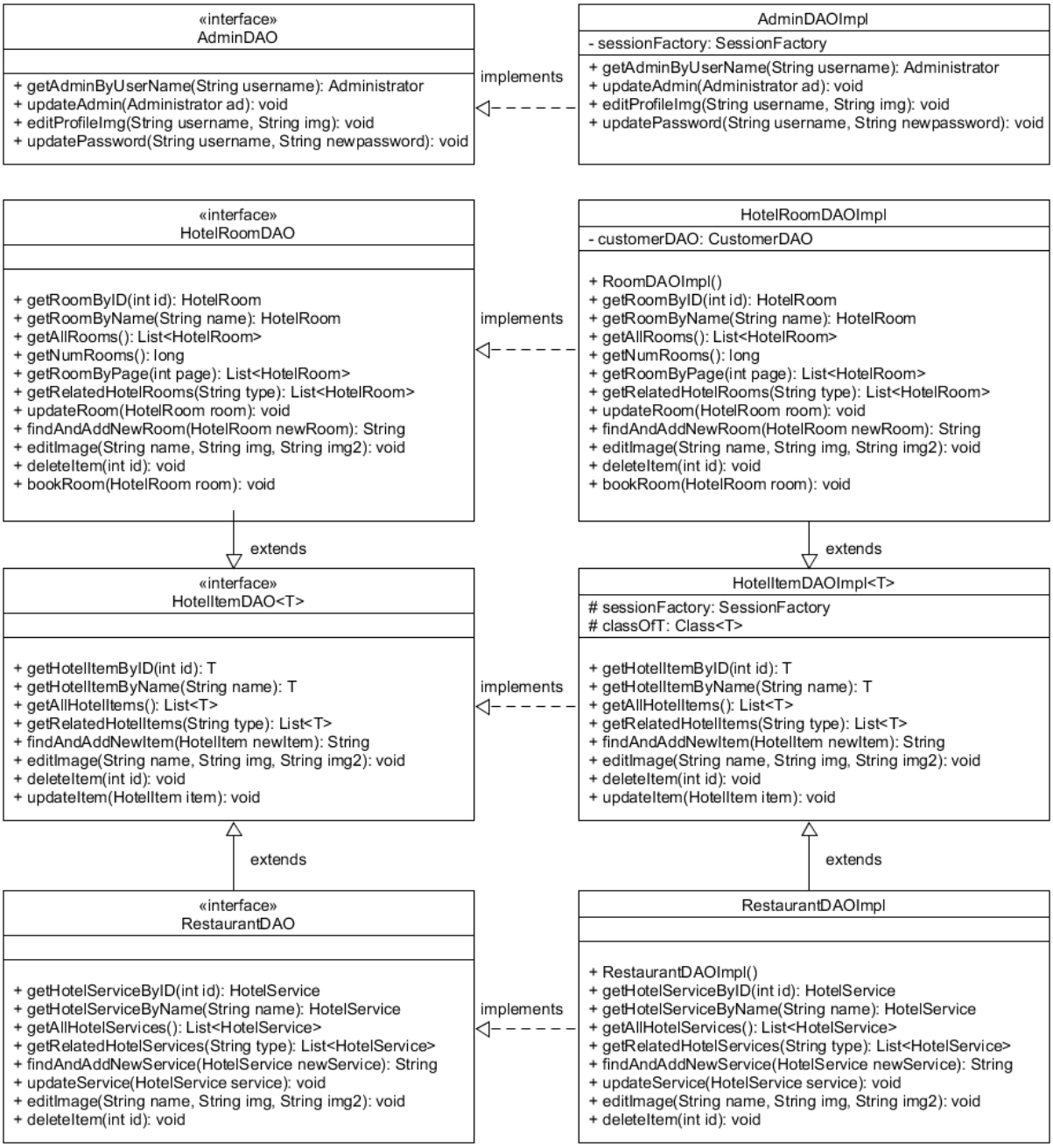


Figure 39: Spring Hibernate DAO class diagram

Base on **figure 39**, interface AdminDAO use Admin model mapping with table admin in SQL provides some methods such as: get Admin by username, edit admin, edit admin profile image, change admin password…

Interface HotelItemDAO use HotelItem model provides some methods relate to room and restaurant such as: get Hotel Item by id, by name, by the same type as

another item; get all Hotel Item as a list, insert, update or delete hotel item; change image of item…

Interface RoomDAO use HotelRoom model and they are mapped with table room in SQL. RoomDAO extends HotelItemDAO so it can use the super methods from HotelItemDAO to provides some methods such as get Room by id, by name, by the same type as another room; get list all rooms; get list of some rooms for each page and also edit image, insert; update; delete room…

Interface RestaurantDAO use HotelService model and maps with table restaurant in SQL. It also extends HotelItemDAO and have the similar method to RoomDAO but relate to restaurant instead of room…

4.4.2.2. Spring API DAO class diagram

Secondly, API DAOs are also the interfaces but instead of expectation SQL for data, they interact with web service as data storage.

According to **figure 40,** CusomterDAO, ActivtyDAO and TrackingDAO use the corresponding MongoDB models includes Customer, Activity, ActionTracking, CustomerWithBehavior, DataCollection, PageAccessChartData and CountryChartData.

Interface CustomerDAO contains some methods such as get customer by username, get all customers, get customer behavior…

Interface ActivityDAO contains some methods relate to Activity, Notification and Message

Interface TrackingDAO contains some methods relate to Tracking Data, Statistic and Chart

All these DAO Interfaces have their implementation that are AdminDAOImp, HotelItemDAOImp, RoomDAOImpl, RestaurantDAOImpl, APIDAOImpl, CusomterDAOImpl, ActivtyDAOImpl and TrackingDAOImpl, These classes also

extends APIDAO to use a function which support retrieving data and clarify the methods that I just mentioned above.

You can have a look at the APIDAO class diagram in **figure 40** below:

Figure 40: Spring API DAO class diagram

4.4.3. Spring Service class diagram

The Services use the corresponding DAOs methods to build functions for Controller. Their methods are process directly from DAOs to Services to Controllers through Dependence Injection of Spring framework where also use the models as data-type.

Figure 41 will show the relationships, variables and methods of these services.

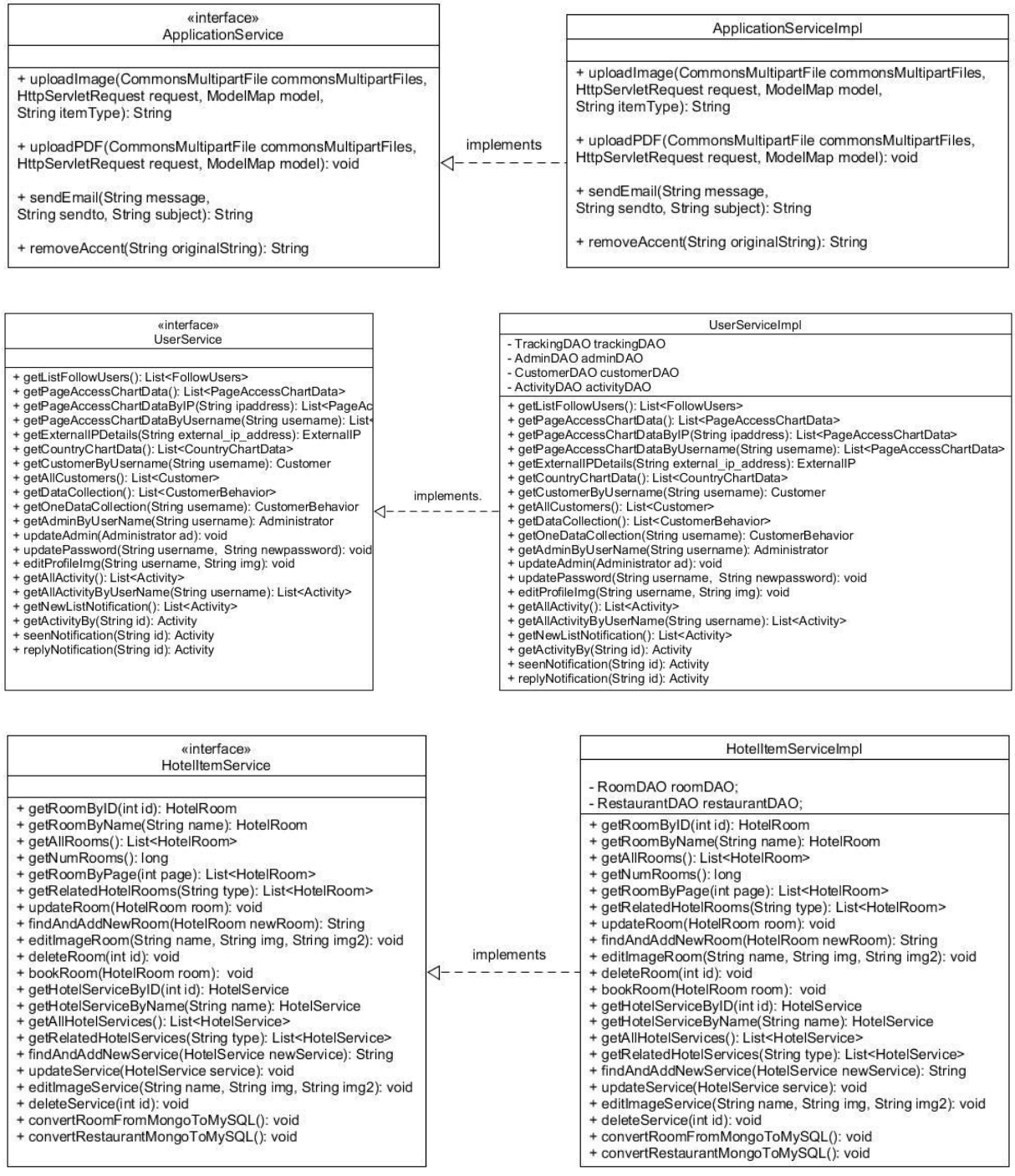


Figure 41: Spring Service class diagram

4.4.4. Spring Controller class diagram

The Controllers interact directly with Services to use their methods for doing the business logic then return data for JSP to display or be a RESTful Web Service provides API for another server or client side.

There are two kind of Controllers supported in my Spring framework application. There are RESTController and AppController with handle their own jobs in my system. Each controller maps with an URL so that user can access this URL on browser.

4.4.4.1. Spring REST Controller class diagram

Figure 42: Spring REST Controller class diagram

In **figure 42,** we have The RESTController class diagram. It contains methods for building RESTful Web Service. All the methods are mapping as URL: /api/{api-endpoint} which HTTP method: GET, PUT, POST, DELETE … api-endpoint is the exactly end point mapping with any API which RESTController provides.

There are several methods that this RESTAPI provides data for other app by HTTP GET method such as get all rooms, get list rooms by page, get room by name, get restaurant item, get page acess chart data …

It also has other HTTP methods such as book room method using HTTP method: PUT. This method allows other application to update booking information to table room without communicate with SQL database server.

4.4.4.2. Spring App Controller, Front Controller and JSP class diagram

On another hand, the AppController represents all mapping URLs for admin website that interacts with JSP files declared mapping in FrontController.

The AppController also handles the business logic when admin sends a request from client side (JSP) and response.

I put the JSP file name similar to AppController methods for easily working. You can see the diagram about AppCountroller and JSP in **figure 43** to understand how they work. If you want to see all variables and methods in class AppController, you can refer **Appendix B** section.

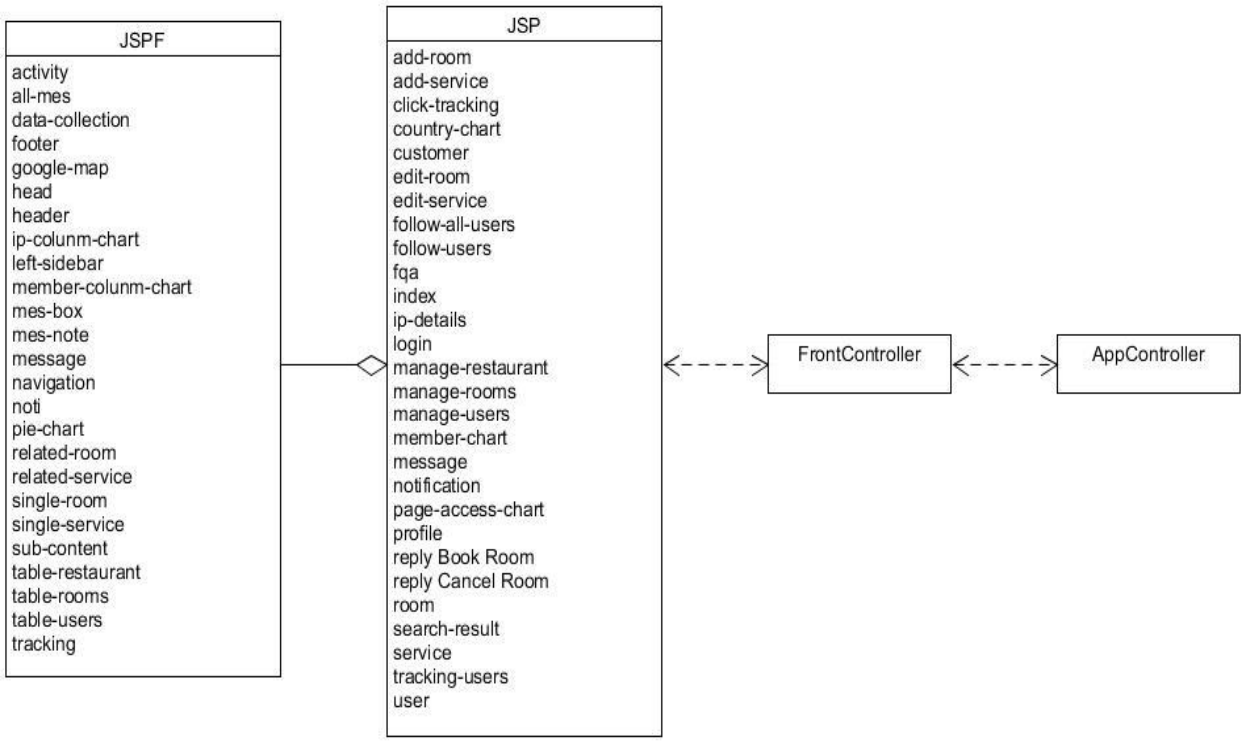


Figure 43: Spring JSP class diagram

The JSP files which include JSPF files for resuse code in JSPF files. The JSPs communicate with AppController by mapping in FrontController.

The FrontController handle view mapping before do bussiness logic in AppController. In my case, I use this bean to mapping:

```
<bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping" />
```

This bean help FrontController mapping with the URL available in AppControler by Anotation ResquestMapping with Request Method is GET or POST.

When ever admin view a page, click another page without any involve anything, like storing or updating data, or upload file, or sending E-mail. For example, admin go to index page, view manage room page, view notification … The AppController will performs the corresponding RequestMapping method with Request Method GET to interact with FrontController for display the JSP file on browser.

For Request Method POST such as login, send email, edit profile, change password, add room, upload fqa file, … The AppController get the parameter from JSP then do bussiness logic and return a next JSP page for admin.

The Front Controller also declare a lot of mapping or anotation config for J2EE web application like mapping package, resource, view resolver, hibernate …

You can have a look at FrontController in **figure 44** below:

dispatcher-servlet.xml

```
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd
        http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-4.0.xsd
        http://www.springframework.org/schema/lang http://www.springframework.org/schema/lang/spring-lang-4.0.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
        http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd
        http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd ">

    <tx:annotation-driven />
    <mvc:annotation-driven
        content-negotiation-manager="contentNegotiationManager" />
    <bean id="contentNegotiationManager"
        class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">
        <!-- Turn off working out content type based on URL file extension, should
            fall back to looking at the Accept headers -->
        <property name="favorPathExtension" value="false" />
    </bean>

    <context:annotation-config />
    <context:component-scan base-package="controller" />
    <context:component-scan base-package="daos" />
    <context:component-scan base-package="api" />
    <context:component-scan base-package="services" />

    <bean id="multipartResolver"
        class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="maxUploadSize">
            <value>104857600</value>
        </property>
        <property name="maxInMemorySize">
            <value>4096</value>
        </property>
    </bean>

    <bean
        class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping" />

    <bean id="viewResolver"
        class="org.springframework.web.servlet.view.InternalResourceViewResolver"
        p:prefix="/WEB-INF/view/" p:suffix=".jsp" />

    <mvc:resources mapping="/resources/**" location="/resources/" />

    <!-- Database Configuration -->
    <import resource="DataSource.xml" />
    <!-- Hibernate 4 SessionFactory Bean definition -->
    <import resource="hibernate.cfg.xml" />

    <bean id="transactionManager"
        class="org.springframework.orm.hibernate4.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
</beans>
```

Figure 44: Spring Front Controller

4.4.5. Spring CONST and Helper class digram

While implement the system, I need some classes that support me calling directly the methods as well as the variables without following the process from DAO to Service to Controller.

The CONST classes and Helper Classes helped me pass my requirement successfully.

4.4.5.1. Spring CONST class diagram

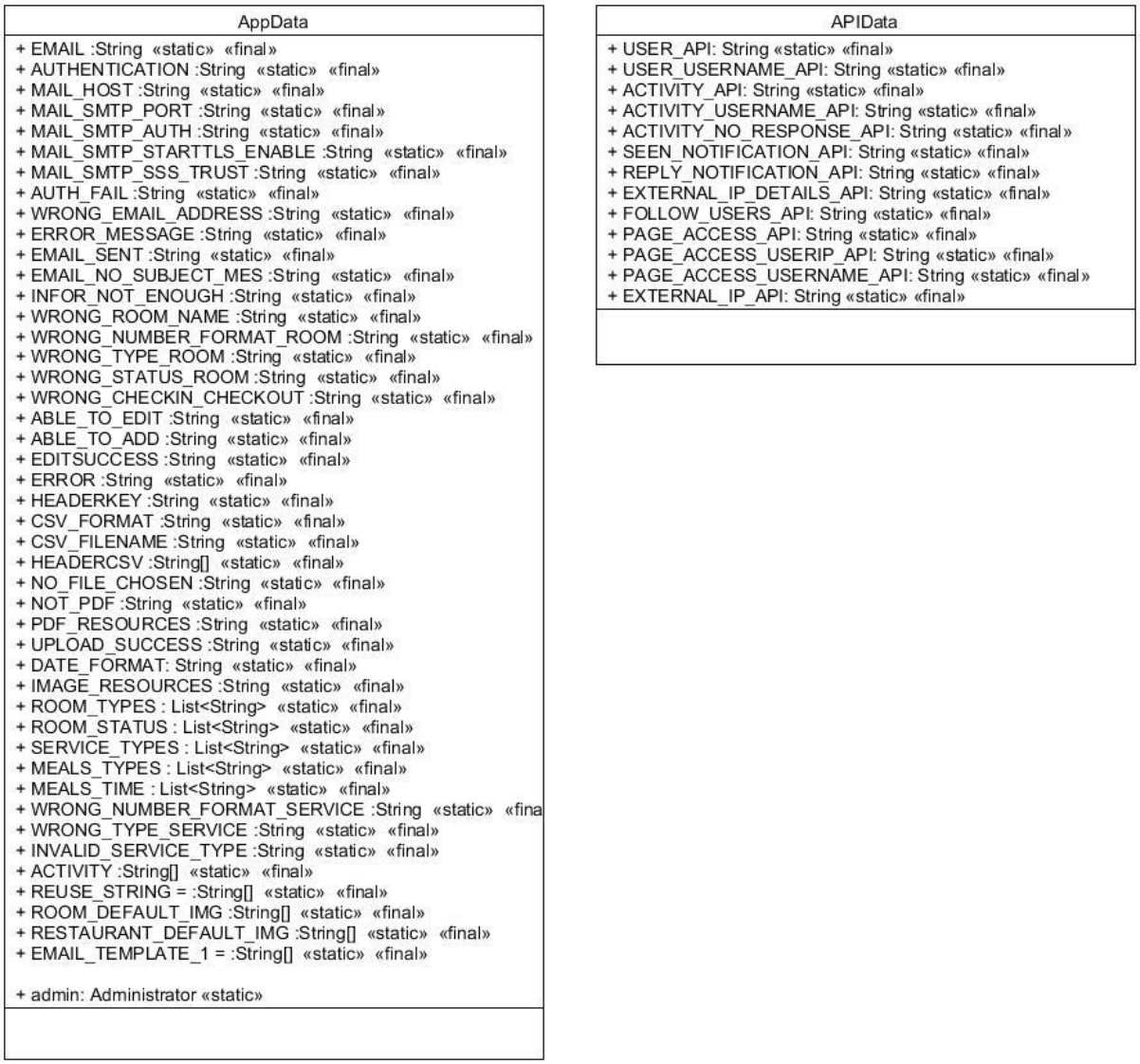


Figure 45: Spring CONST class diagram

These classes in **figure 45** contain lots of Constant variables for the whole Spring application use. The different between them is that the AppData includes variables for configuration, error message, information message, constant variable for email sending, CSV uploading. Besides, the APIData only contains Constant variable

related to API. These 2 classes help refactor code easily and solve the hard coding problem.

4.4.5.2. Spring Helper class diagram

Figure 46: Spring Helper class diagram

These classes in **figure 46** contain some methods for the whole system which do not really need any business model. Their methods work independently but they are very necessary. In my system, I need to format date and time so much because I use two kinds of database system MongoDB and SQL; their date and time format are different.

Moreover, to work with String and Number is not very easy, MathCalculator and StringUtils help me to solve this problem. My system also support email sending, file uploading and geo location lookup feature so that I need three classes EmailSender, FileUploader and GeoLookup.

4.4.6. The Class Diagram for the whole Spring application

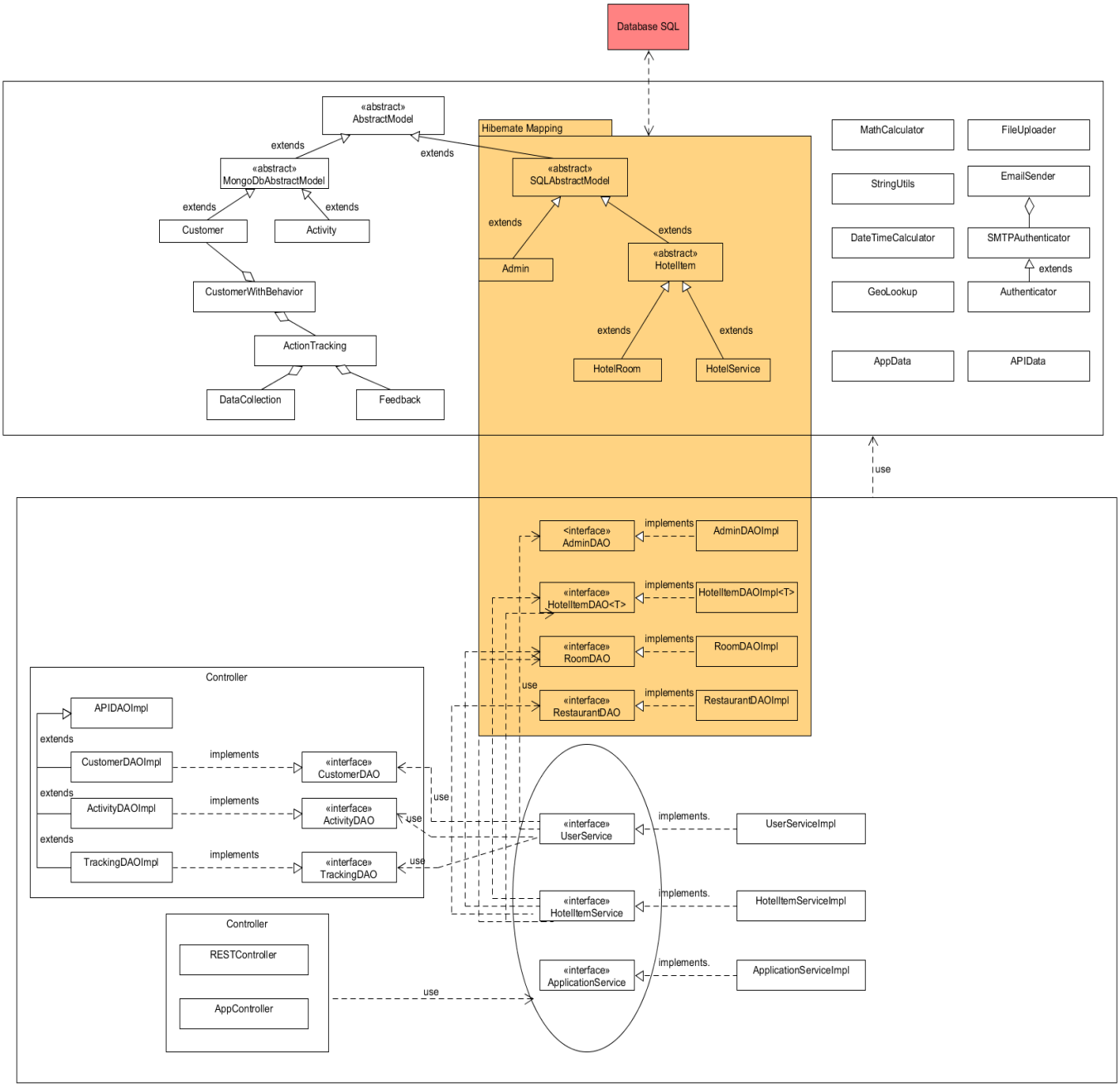


Figure 47: Class diagram for the whole Spring application

This class diagram is combined by all the class diagrams from **section 4.4.1 to 4.4.5**

You can see that in **figure 47**, the UserService uses methods from CustomerDAO, ActivityDAO, TrackingDAO and AdminDAO. The HotelItemService use methods from HotelItemDAO, RoomDAO and RestaurantDAO. The ApplicationService uses almost methods from Helper classes. These Services provide methods for Controller and RESTController.

All the process goes though Dependence injection supports by Spring IoC container.

The Models provide data-type for application, the CONST and Helper classes provides constant variables and methods for the whole system.

4.4.7. Express Model class diagram

Express models communicate directly with MongoDB by mongoose library. They also provide data-type for Node.js Application.

Model Customer contains customer personal information, username & password with some functions get a user by id, get a user by username, get list of users and edit a user.

Model Activity contains some activity of customer with some functions such as get an activity by id or a list of activities of an user by username, get the list of all activities, list activities that haven't been replied yet, list notifications to display for administrator, add new activity and delete an activity.

Model Tracking contains tracking data of user such as the IP address, the external IP, the page user clicked and the duration that user stays in this page with all functions to return the data of users tracked by the system.

These data will be sorted, searched or filtered to provide other functions for reasonable purposes such as sort a list of tracking data by IP address, view tracking

data by page, view statistics of page access and username or IP address, view country chart by location of user by IP address …

Figure 48 will clarify all the variables and functions of those models.

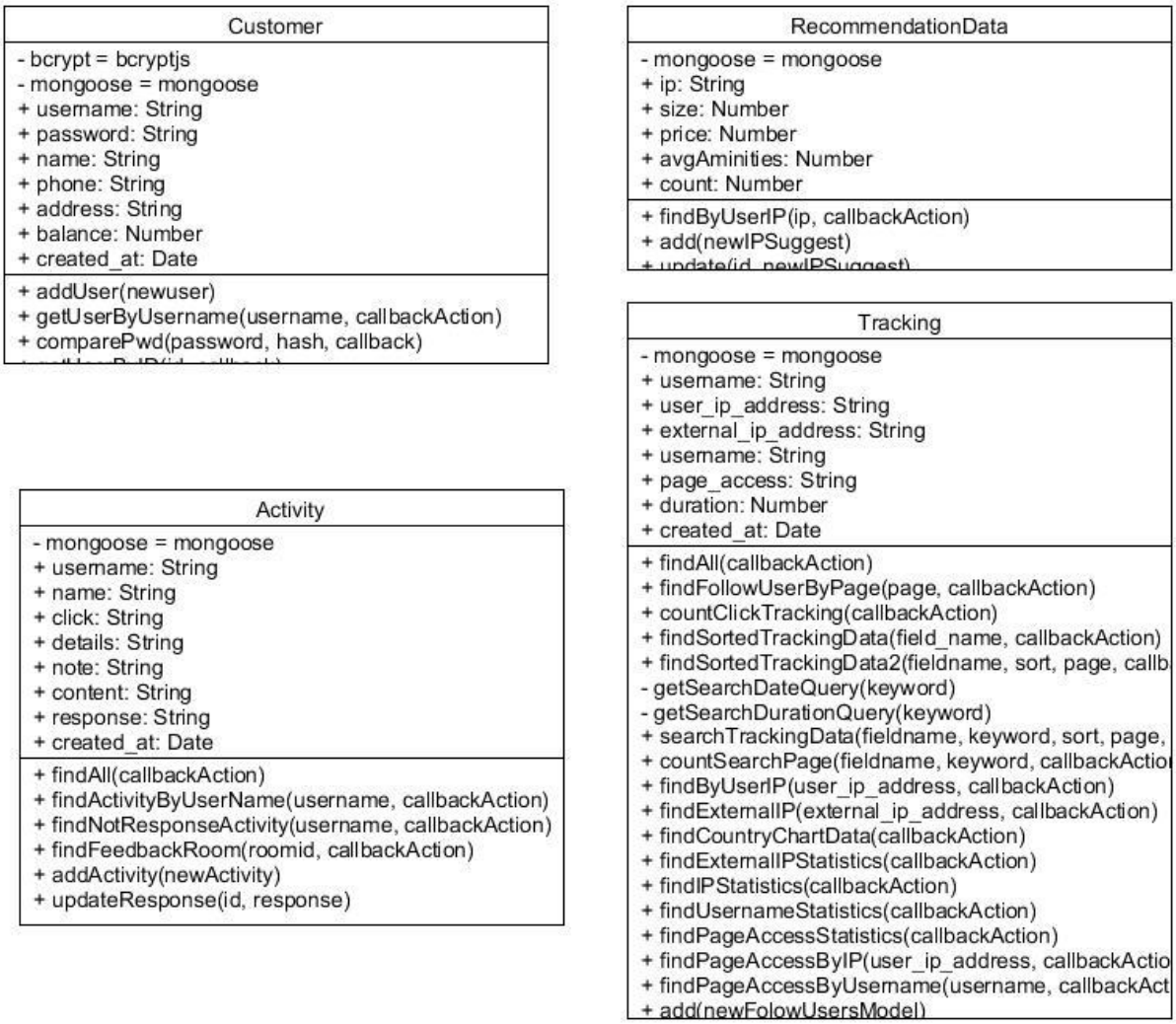


Figure 48: Express Model class diagram

4.4.8. Express Controller class diagram

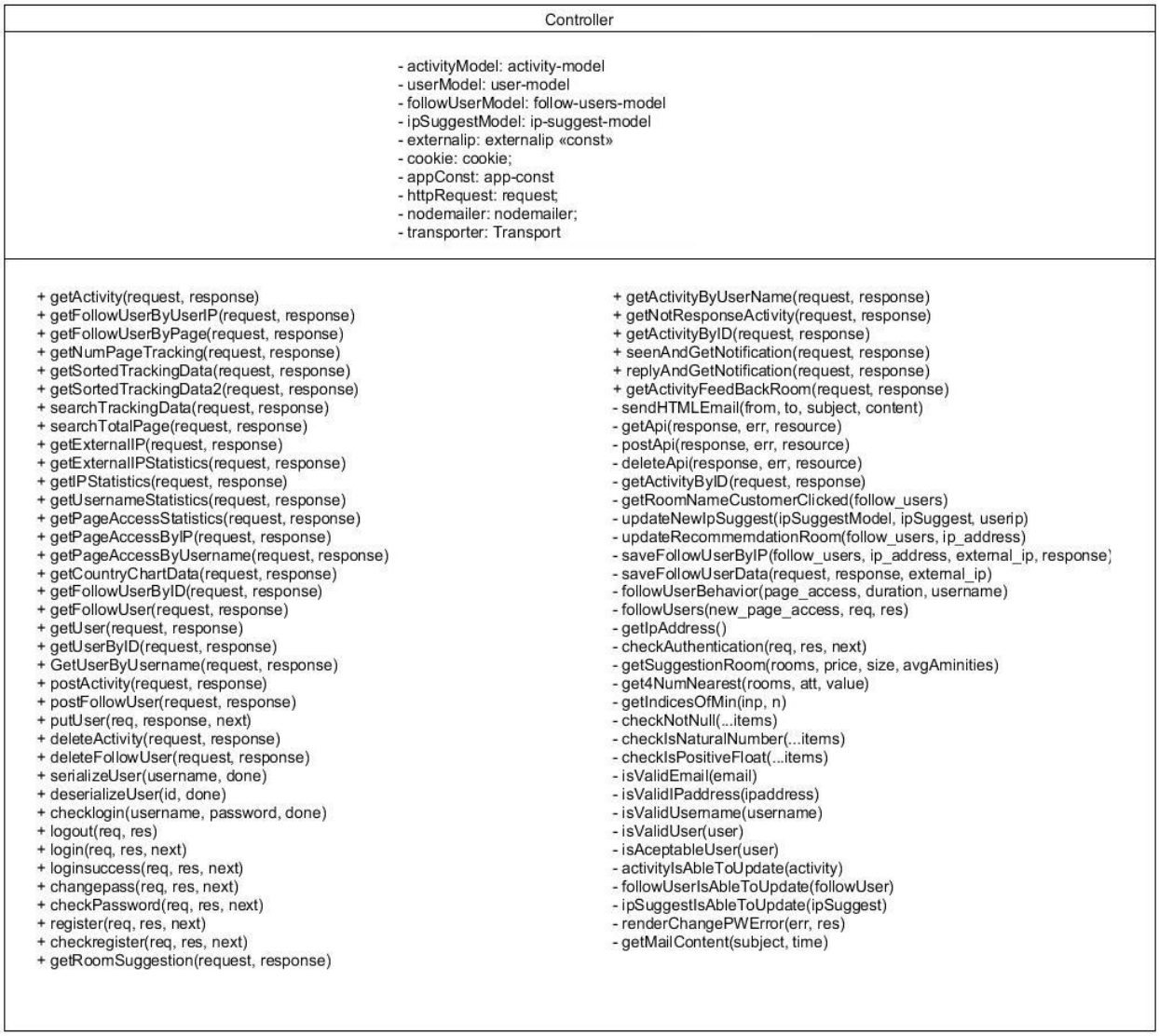


Figure 49: Express Controller class diagram

The Controller has a lot of checking functions such as: check valid email, check valid IP address, check valid username, check user is acceptable, check activity is able to edit and so much more.(**figure 49**) It also uses the Models as data-type and their functions that interact with MongoDB to create lots of suitable functions for Routes.

You can have a closer look at the variables and functions of this class in **Appendix C** section.

4.4.9. Express Route class diagram

The routes are supported by Node.js route library and Express framework that handle responses and requests of users from client side or from other applications.

4.4.9.1. Express AppRoute with EJS class diagram

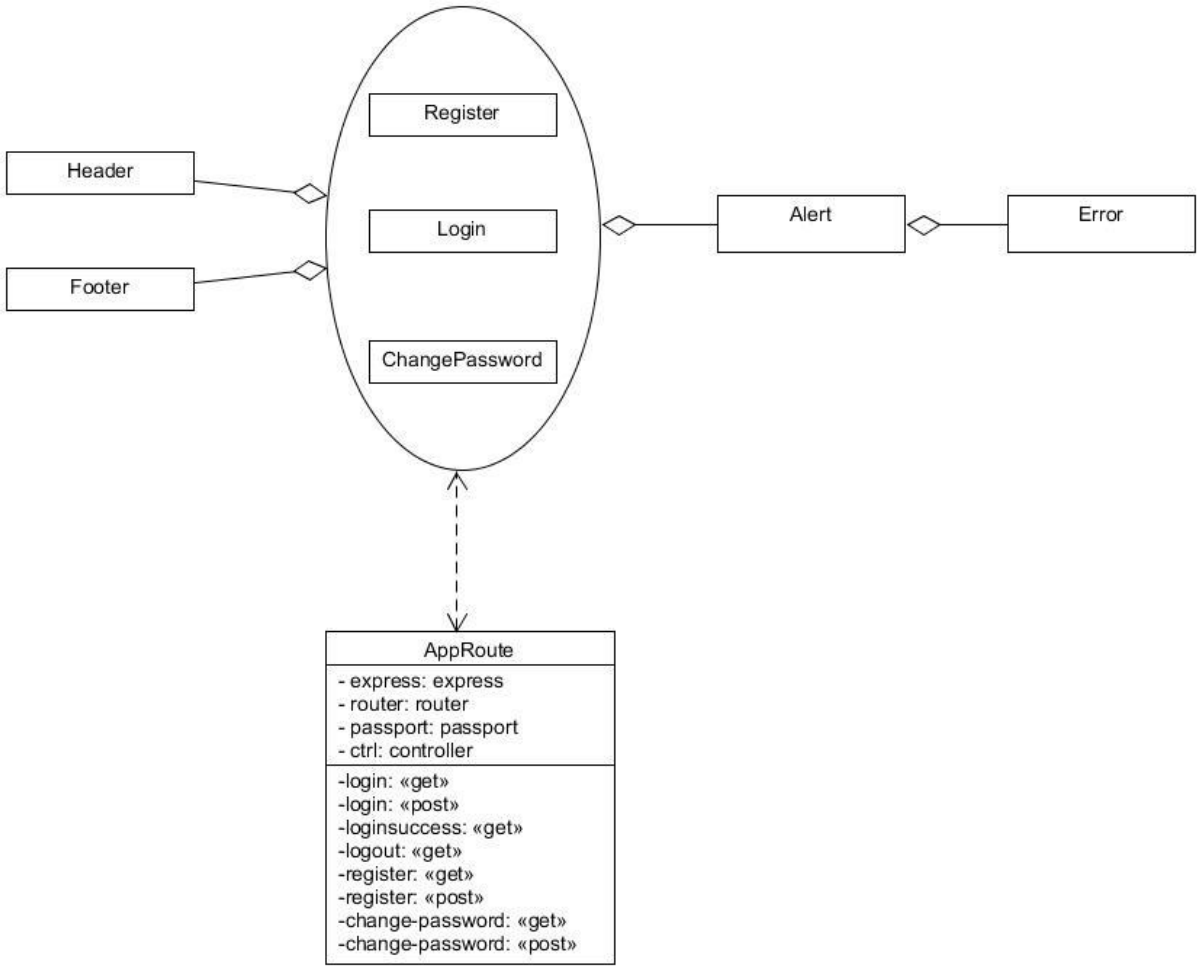


Figure 50: Express App Route and EJS class diagram

This AppRoute in **figure 50** interacts directly with the EJS files to render the view. It supports function for login, logout, register or change password.

When users click on login, register or change password page, the load page requests will go to corresponding functions in AppRoute with request mapping is

GET. The logic will be executed by these functions and render directly the page that user requested.

When users input username, password or other information to login, register or change password. The requests will go to corresponding functions in AppRoute with request mapping is POST. Then the AppRoute will check the information and return response display for users.

4.4.9.2. Express APIRoute class diagram

Besides, there is no view rendered by APIRoutes, these Router's mission is to provides a REST API for other application.

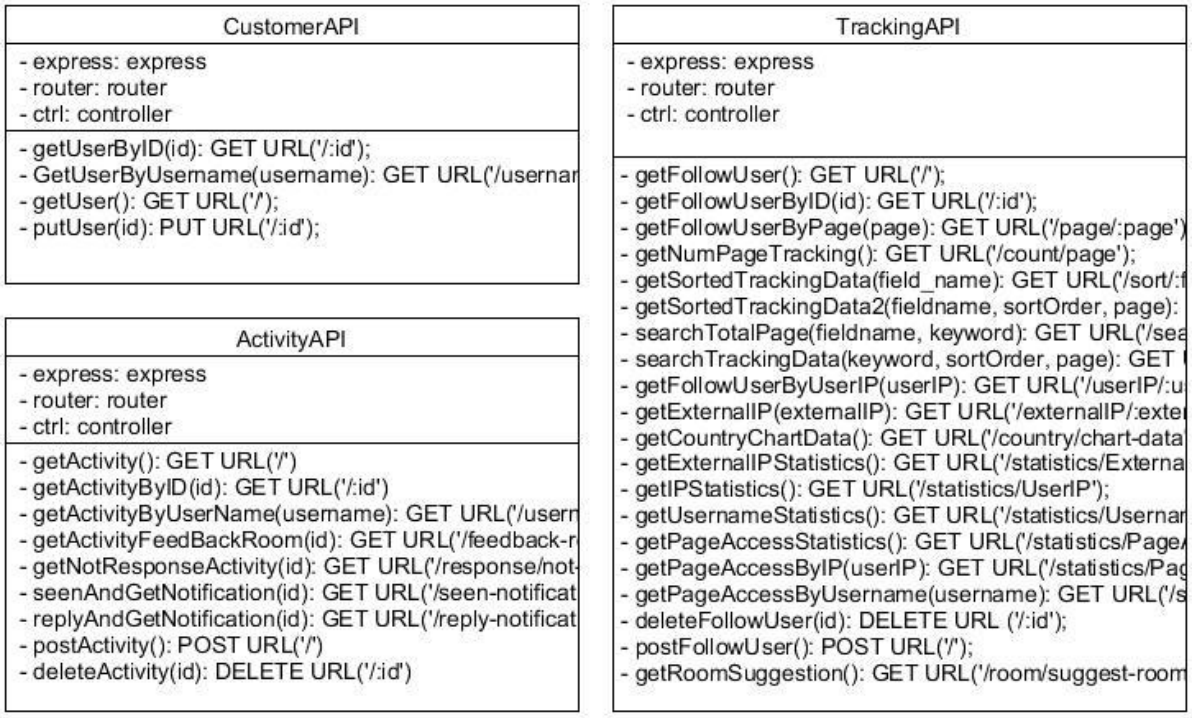


Figure 51: Express API Route class diagram

These RouteAPIs (**figure 51**) provide HTTP methods such as GET, PUT, POST, DELETE. They are main components for building RESTful Web Service by Node.js and Express framework.

The CustomerAPI Route provides some methods that relates to Customer. Other application want to access the data from MongoDB need to ask this Route.

With ActivityAPI Route, all the activities of Customers can be retrieved, updated by another application without any connection to MongoDB.

For which want to use the tracking features have to contact the Tracking API Route for data because it provides a huge amount of tracking data as JSON thought HTTP methods.

4.4.10. The class diagram for the whole Express Application

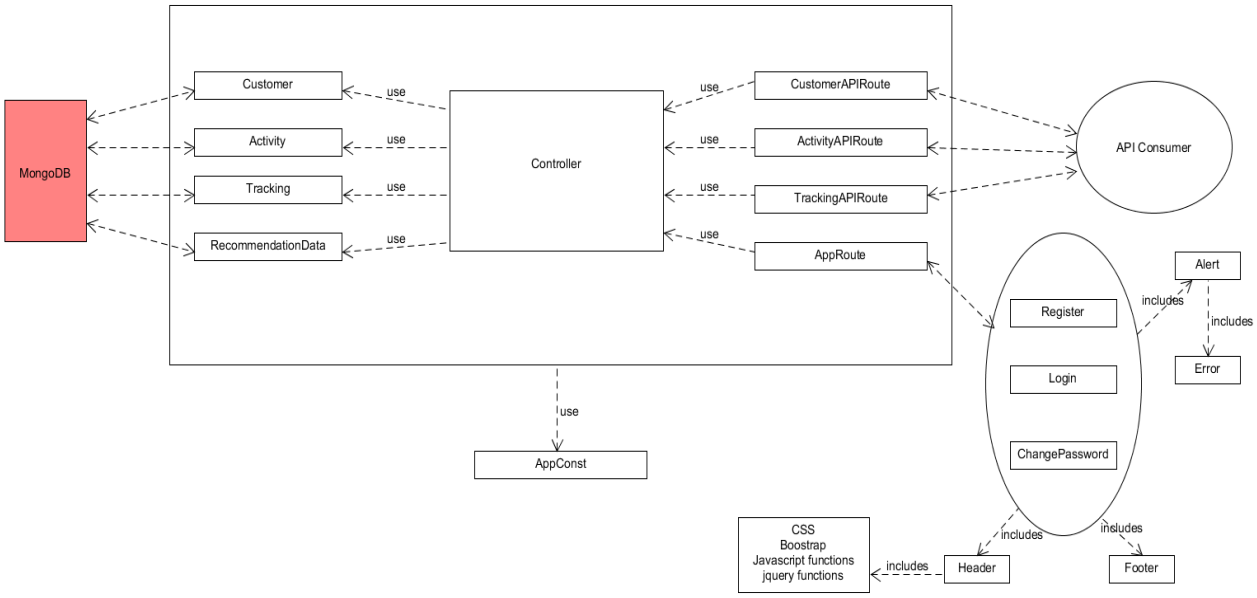


Figure 52: Class diagram for the whole Express application

Figure 52 showed the general Node.js Express framework class diagram. As you can see, the Model includes Customer, Activity, Tracking and RecomendationData that interact directly to MongoDB then provide functions and data for Controller to execute business logic.

The final step is from the AppRoute to view or from CustomerAPI, ActivityAPI and TrackingAPI Routes to build RESTful APIs provide data for other application.

The AppConst is used as the CONST data storage where contains all the necessary constant variables.

4.4.11. Angular2 Model class diagram

To build a MEAN stack application, we have to merge MongoDB, Express framework with Node.js on server side to Angular on client side. Angular play as the role 'View' to display data from Node.js & Express RESTful Web Service on server retrieved from MongoDB as JSON-like format. [33]

Moreover, Angular can interact with other web service not necessarily MEAN stack technology as well.

In this thesis report, the class diagrams of my system not only focus on the server-side systems but also coverage the client-side applications. **From this section to 4.4.16**, I will show the class diagrams on front-end sides where include Angular2, AngularJS and so much more.

Firstly, this section will clarify the class diagram for Angular2 models. Follow the rules of server's API, the Angular2 models need to have the variables based on the JSON that RESTful web services provide, and therefore, they are similar to the variables of Spring and Express models in **section 4.4.1.1 and 4.4.7**.

My Angular2 models only contain variables and constructors because in my Angular2 application, the validate functions are not declared in models.

Angular2 models just provide data-type for Services and Components. You can recognize that when looking at **figure 53** below.

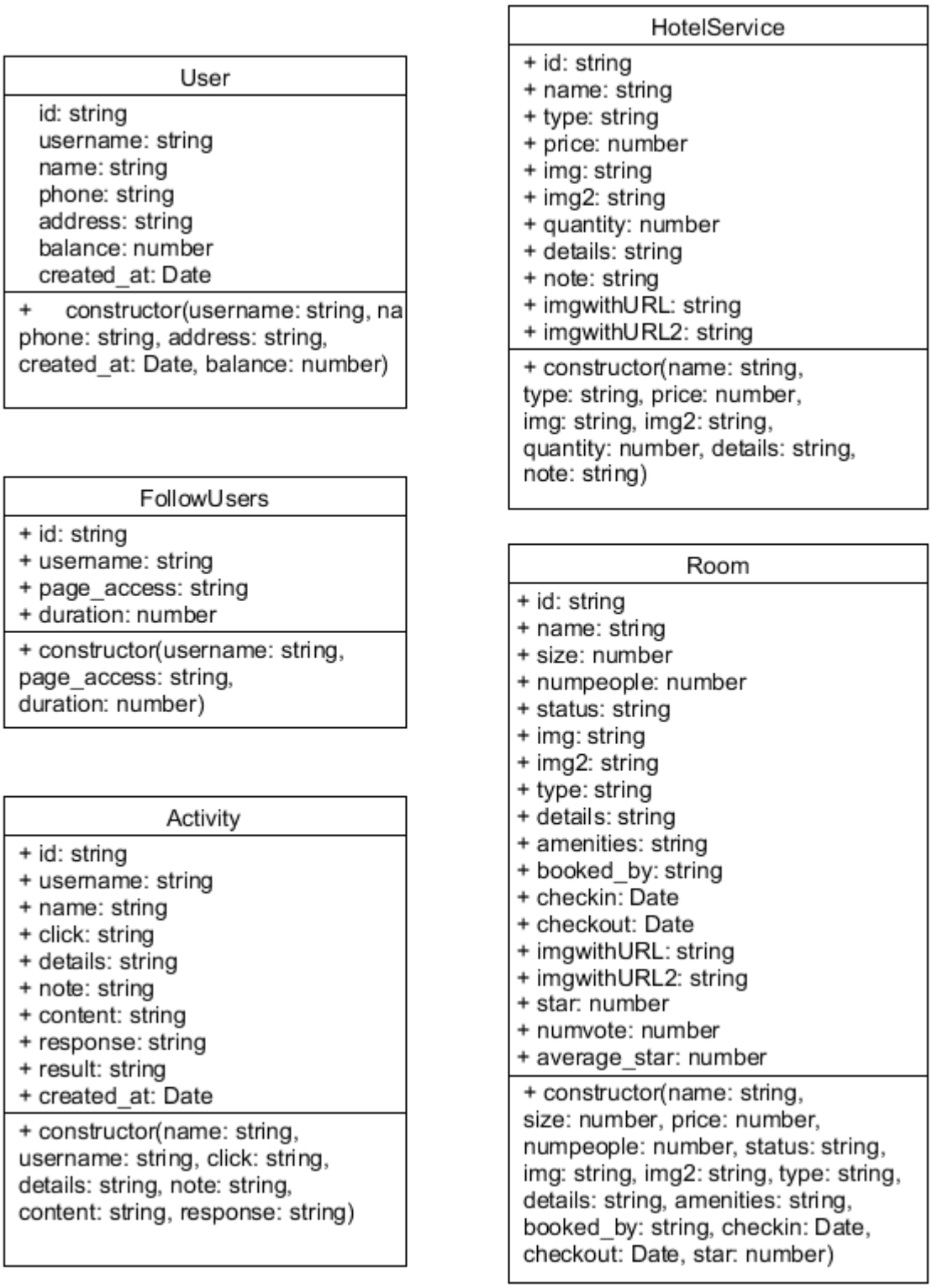


Figure 53: Angular2 model class diagram

4.4.12. Angular 2 Component and Template Class Diagram

Components and Templates are the main reasons that make Angular2 powerful. No Angular2 application can run without them.

As you knew in **section 2.4**, Components control the views called templates. In my Angular 2 app, each component has a corresponding template for data-binding mechanism.

In **figure 54**, you can easily see that the AppComponent includes HeaderComponent, FooterComponent and the content of each page. The HeaderComponent and FooterComponent control the header and footer of the page and it is permanent in different page. However, the content is initialized as home page written in HomeTemplate and then it will be changed to another content in another component by Angular2 Router whenever users click on other link.

On other hand, HomeComponent and RoomDetailComponent use ReservationComponent. The HomeComponent controls HomeTemplate which is home page and RoomDetailComponent controls RoomDetailTemplate.

In the case above, when customers click on home page or room-details page, in both sides they can see the reservation form to input. This reservation from is display by ReservationComponent with all validate functions as well as submit function. That is similar to the case of RoomTariffComponent and RoomSuggestionComponent.

With the data-binding mechanism between Components and Templates, everything display on Angular 2 application pages can be changed without loading page. That will improve the performance as well as reduce loading page and rendering page of servers.

If you are looking for more details about the variables and functions of my Angular2 Components, please pull down to **Appendix D** section.

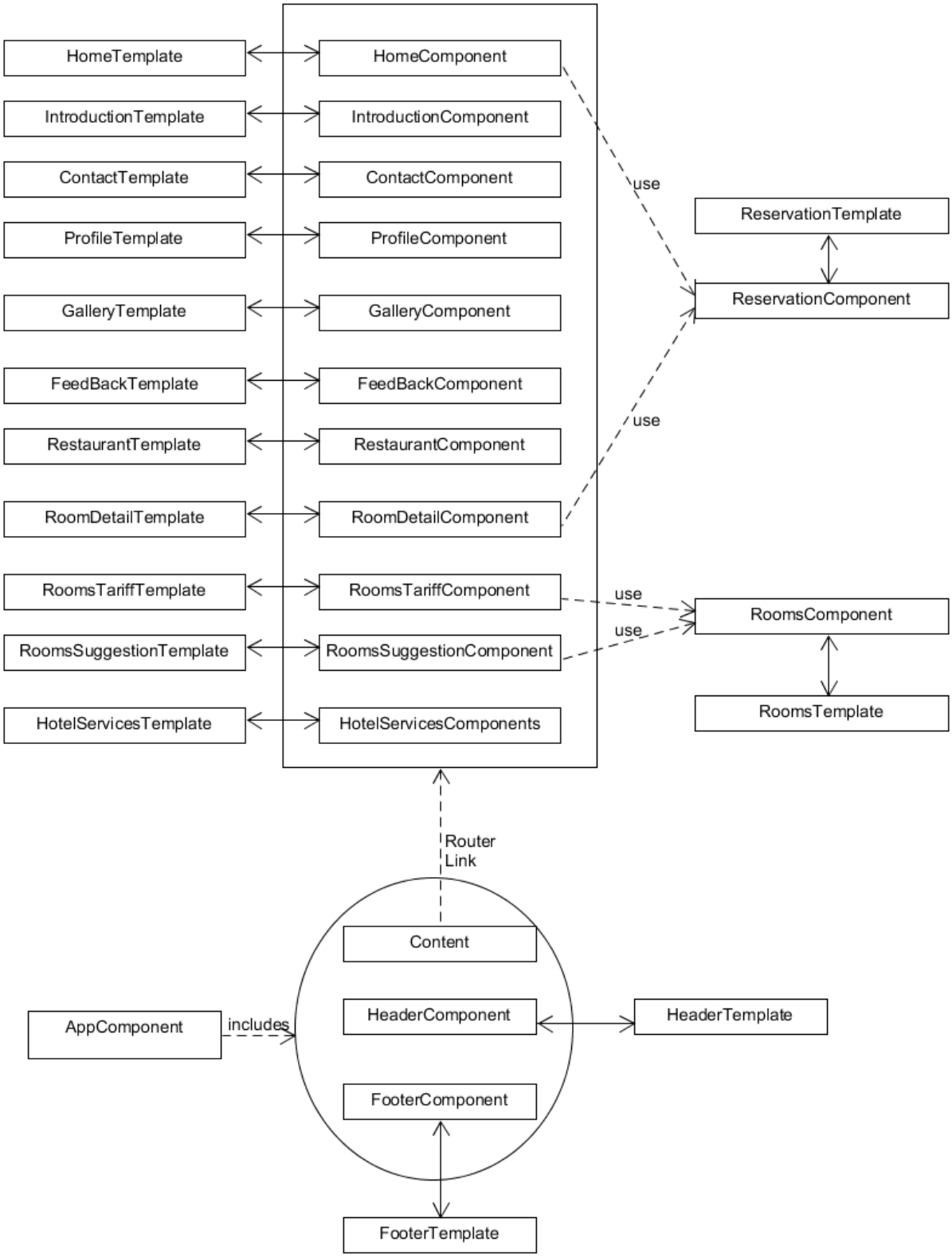


Figure 54: Angular2 Component and Template class diagram.

4.4.13. Angular 2 Service Class Diagram

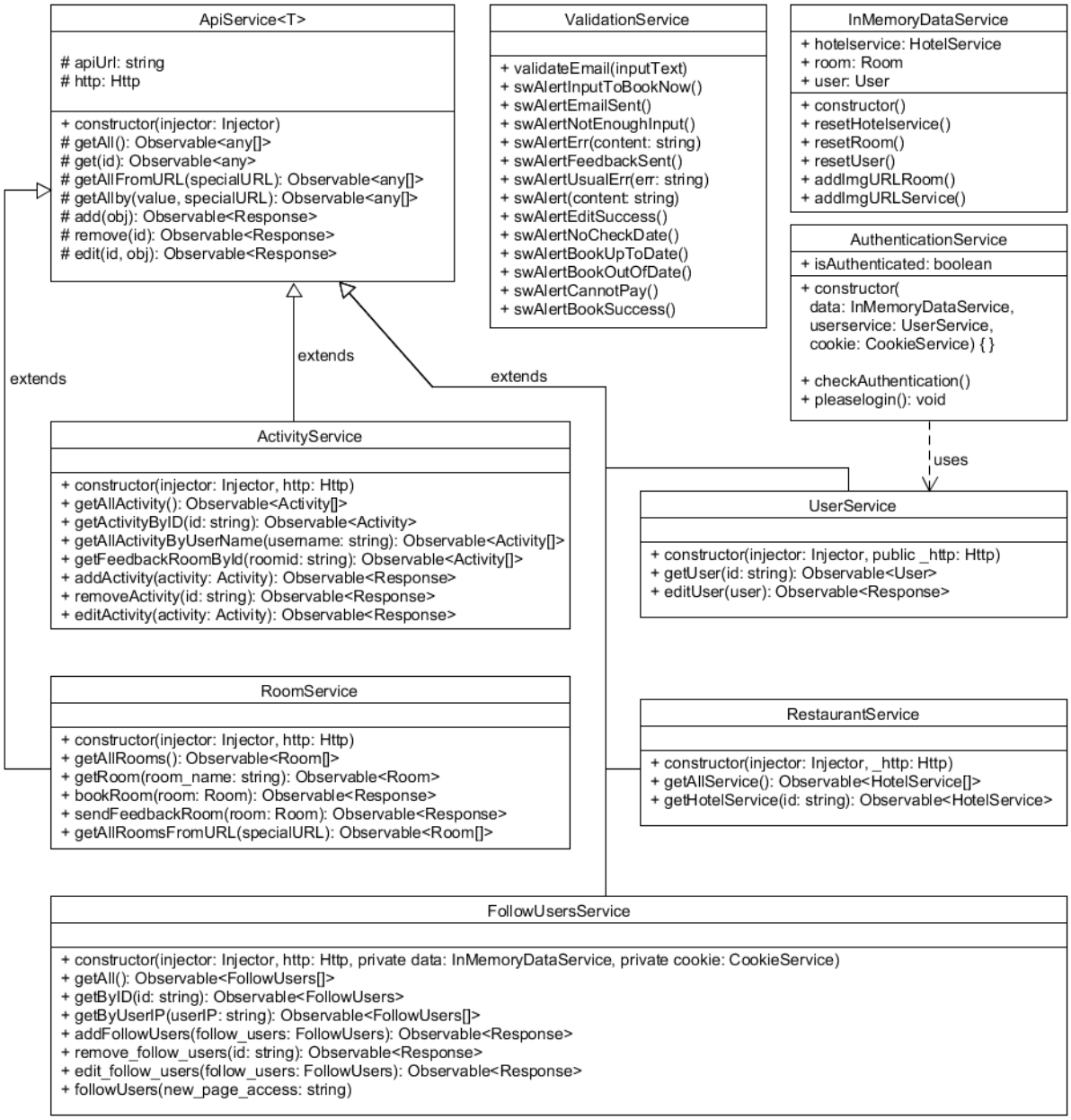


Figure 55: Angular2 Component and Template class diagram

Services provide functions and data for components. One of the main purposes of them is reusing code. In my Angular2 application, services stand as the role 'The API interactor'. They support Component communication with RESTful web service, parse JSON get from API into object and send HTTP methods: POST, PUT, DELETE to web service. Besides, other jobs of my Angular2 services are validate, alert information or error and check authentication. (**Figure 55**)

4.4.14. Angular 2 Const Class Diagram

With the same meaning and purpose of other CONST classes in my system, this Angular2 CONST also provides constant variables for my Angular 2 application. (**Figure 56**)

Figure 56: Angular2 CONST class diagram.

4.4.15. The Class Diagram of the whole Angular 2 Application

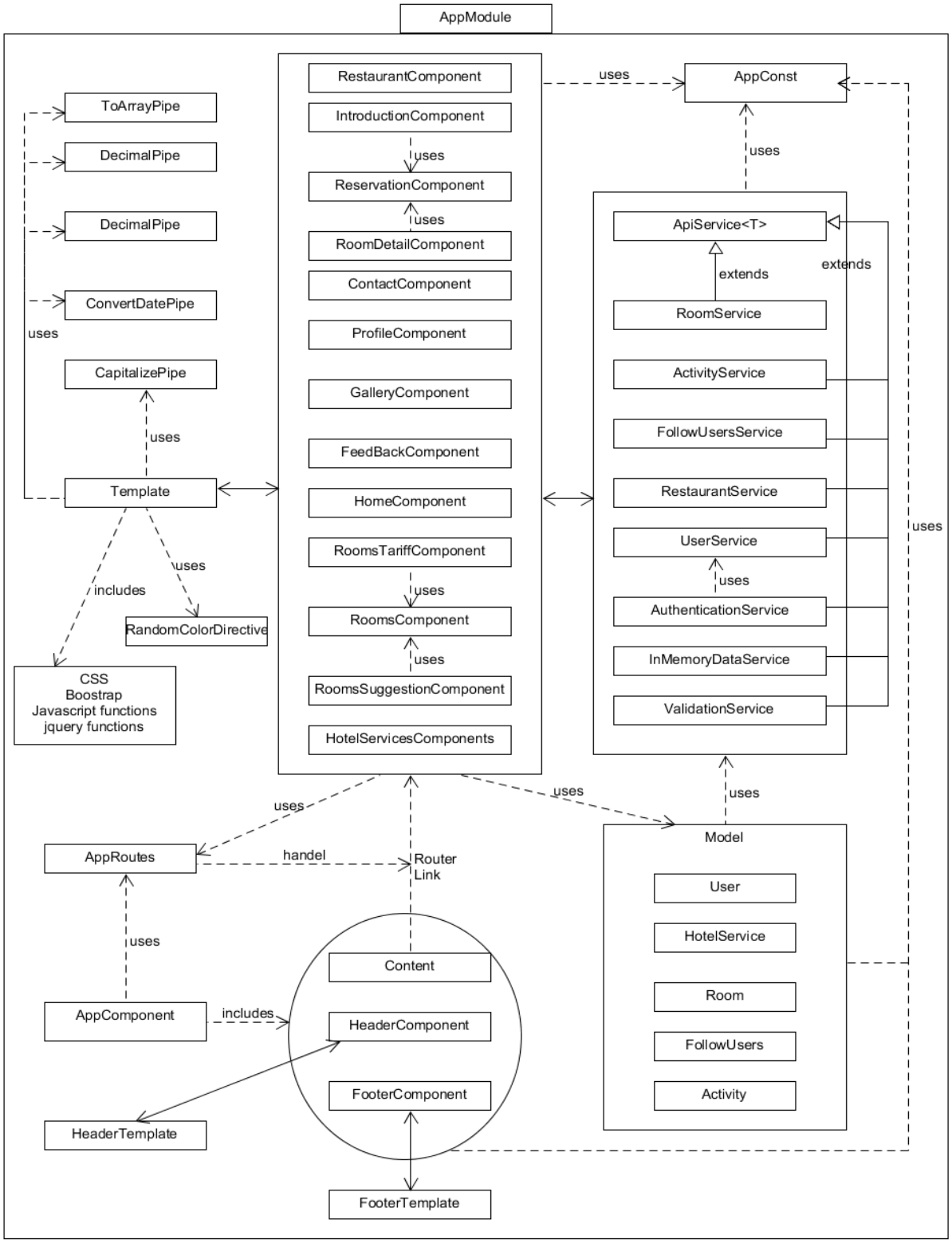


Figure 57: The class diagram of the whole Angular2 Application

The class diagram in **figure 57** is the combination of all class diagrams from **section 4.4.11 to 4.4.14.** These sections have already described how Components, Templates, Services, Models and AppCONST work in my Angular2 application.

Furthermore, my Angular2 app also contains Pipes, Directive, Router and Module.

The ToArrayPipe creates a mechanism to convert a number into an Array in my Angular2 template.

CapitalizePipe helps me to capitalize the first letter of a string.

DecimalPipe supports converting all formats into decimal number.

ConvertDatePipe provides the feature for my Angular2 template to convert a string into date.

Finally, I use OrderByPipe to sort all the data in arrays.

RandomColorDirective creates a random-color element in my Angular2 templates. It supports changing a color every one second. It also changes the color whenever user click or hover on a link embedded this directive.

My Angular2 AppRouter handles the view mapping as well as makes decision which component is used to display and which is hidden.

Components control Templates and some reused data and in Services. Components also authorize API interaction for Services and both of them use data-type from Models.

AppModule controls all Components, Templates, Services, Pipes, Directives, AppRouter and AppConst.

4.4.16. The Class Diagram of AngularJS embedded in JSP

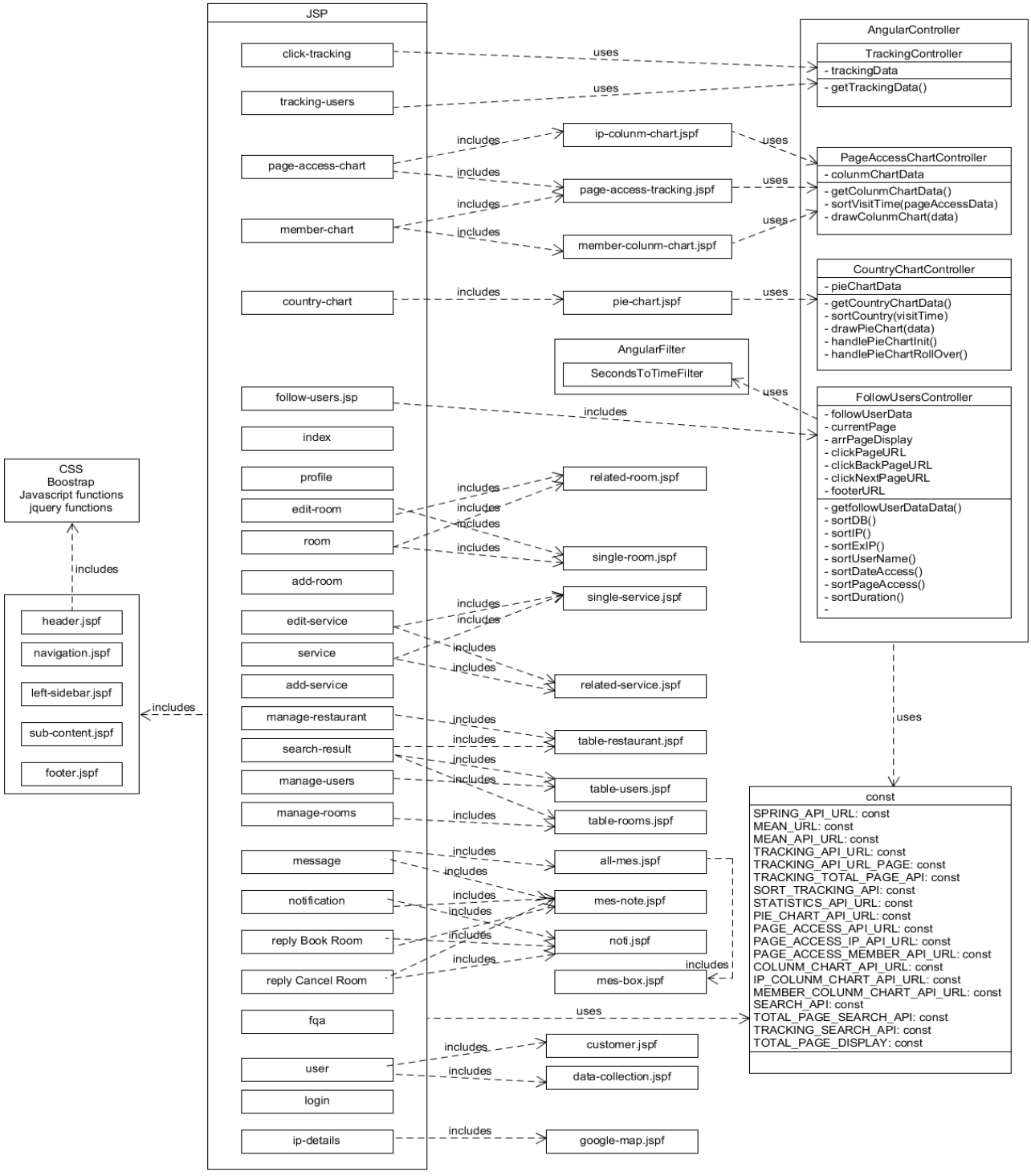


Figure 58: The class diagram of AngularJS embedded in JSP sides

The class diagram in **figure 58** clarified the JSP class diagram in **Figure 43** and **section 4.4.4.2** with the appearance of AngularJS framework on client side.

The SecondsToTimeFilter creates a mechanism for AngularJS template to convert a number in millisecond unit into HH:mm:ss.SSS format.

Some JSP files or JSPF files (JSP fragments, is used to be statically included in another JSP file) use AngularJS controllers include TrackingController, FollowUsersController, PageAccessChartController, CountryController to retrieve data from RESTful web services by some functions such as: get tracking data, get follow-users data, get page access and country chart data to draw column chart and pie char.

These AngularJS controllers will interact with a part of view in JSP or JSPF files by data-binding mechanism to reduce loading page in my Spring MVC app.

This class diagram also uses CONST as constant data storage and everything is embedded in JSP files controlled by Spring AppController. (review in **section 4.4.4.2**)

4.5. Implementation

According to the class diagrams above, in this section, I will describe in brief how I implemented some several highlight features of my system. The **figures from 59 to 88** will be located under each features which I am going to mention below:

4.5.1. Customer feature

As you knew, customer needs an account to login into system to book room, send feedback, view profile with transaction history. The login process first checks whether user have logged in or not by passport library of Node.js. If they have signed into system successfully, the system allows users to use these features. When customers logout, the session and cookie are removed so that they cannot use customer's features anymore.

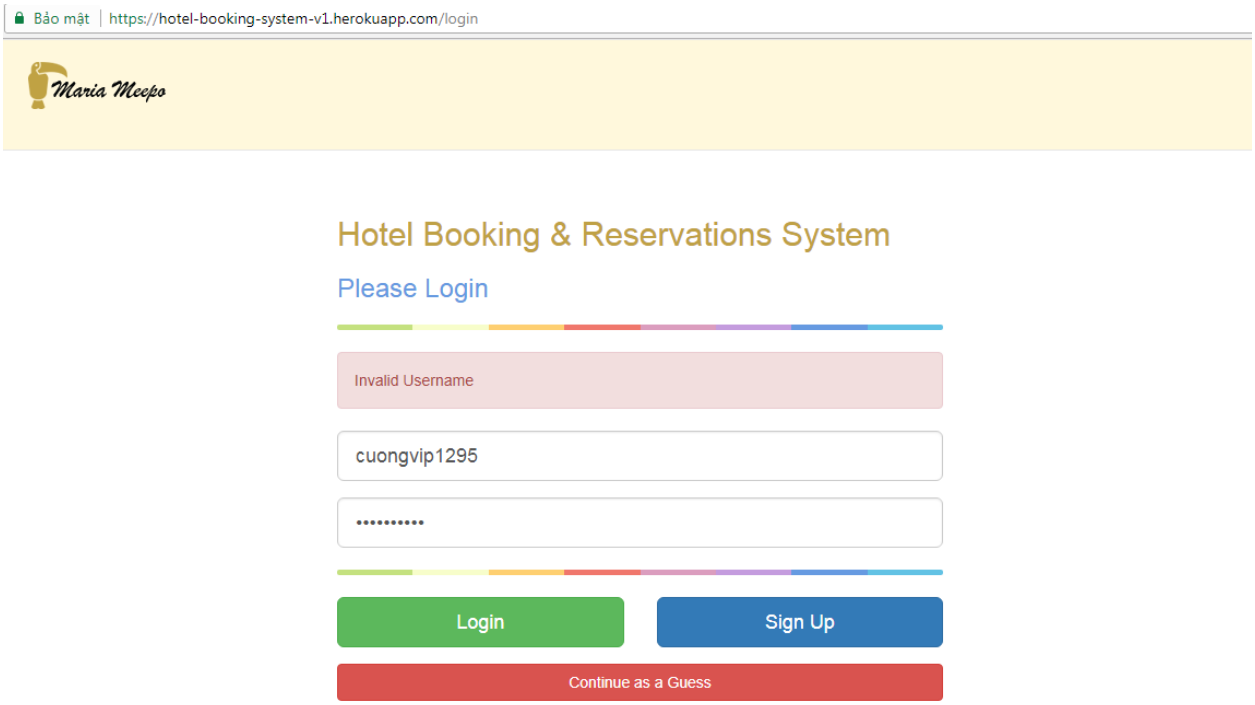


Figure 59: Customer login feature

The register feature is actually the function that add new user into customer collection of MongoDB. The password must be encrypted by bcryptjs before the customer data are inserted into database collection. This function also works for change password feature which compare the hash of encrypt password with new password and confirm password; if new password is different from current password and the confirm password is correct, the system will update the password of customer to MongoDB.

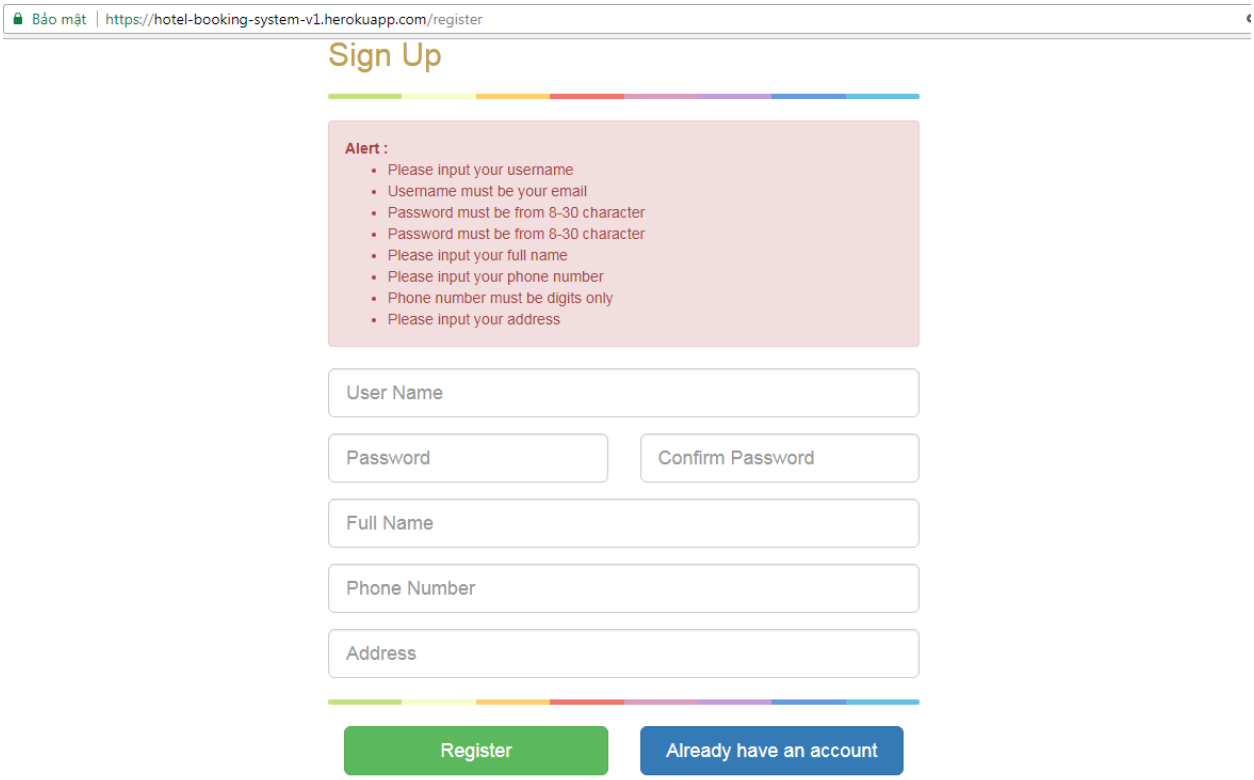


Figure 60: Customer register feature

With EJS powered by Express framework and Node.js, all the input data of customer to register, login, change password are validated by 'request.checkBody' function.

Node.js & Express RESTful Web Service is the transportation connect to MongoDB for providing Customer data as JSON though API. Every feature relates to customer have to contact with this web service for retrieving data or update to MongoDB.

For example, to manage Customer, the CustomerDAO has to get the customer information from REST API to display for Administrator

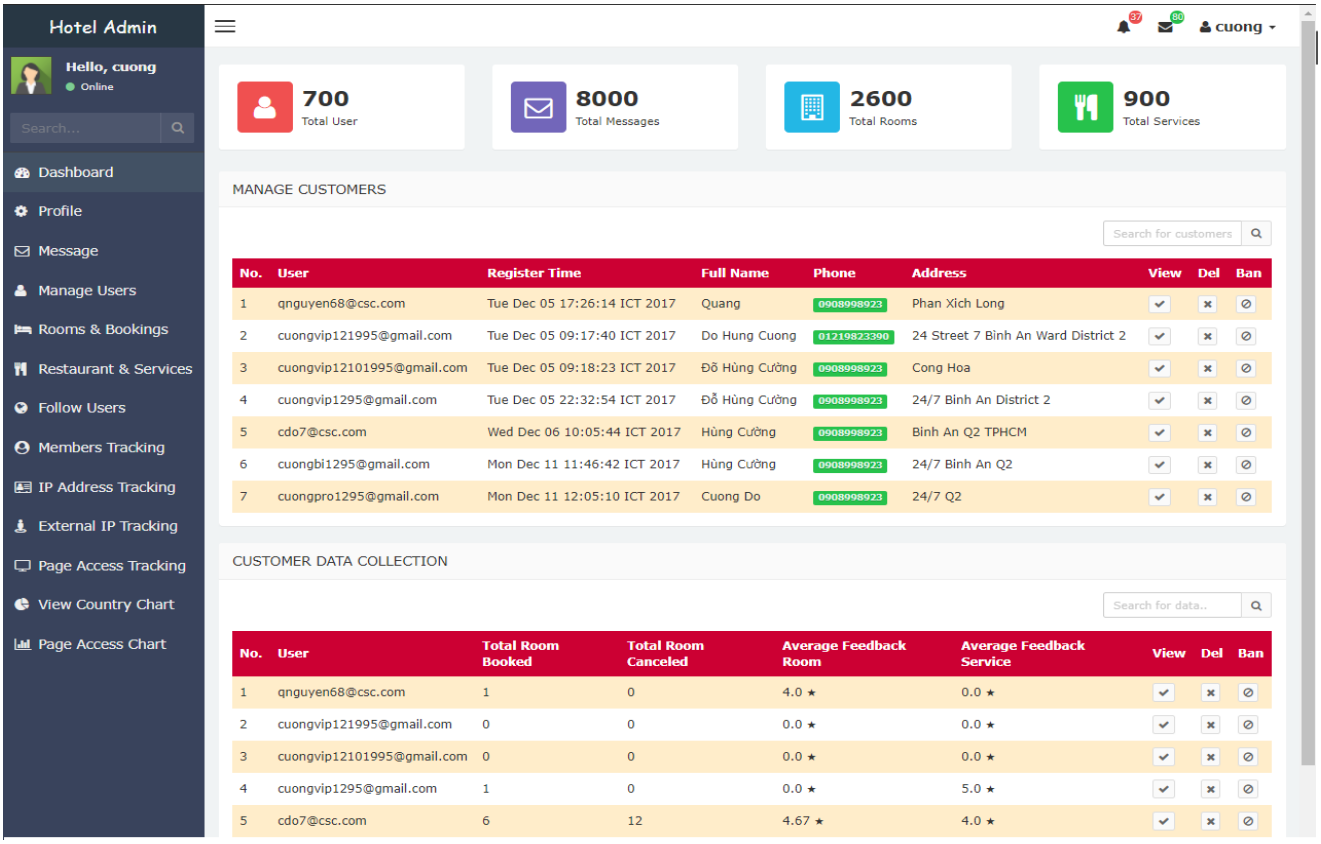


Figure 61: Manage Customer feature

4.5.2, Activity feature

All activities of customers are stored in activity collection. Whenever a customer books a room, cancels a room, sends feedback, rates a room, sends contact or sends reservation form, Angular 2 send request with parameters (if necessary) by HTTP POST method to Node.js for checking these parameters whether they are acceptable or not. If all the parameters are valid, Node.js then Activity Node.js Model inserts new activity into MongoDB.

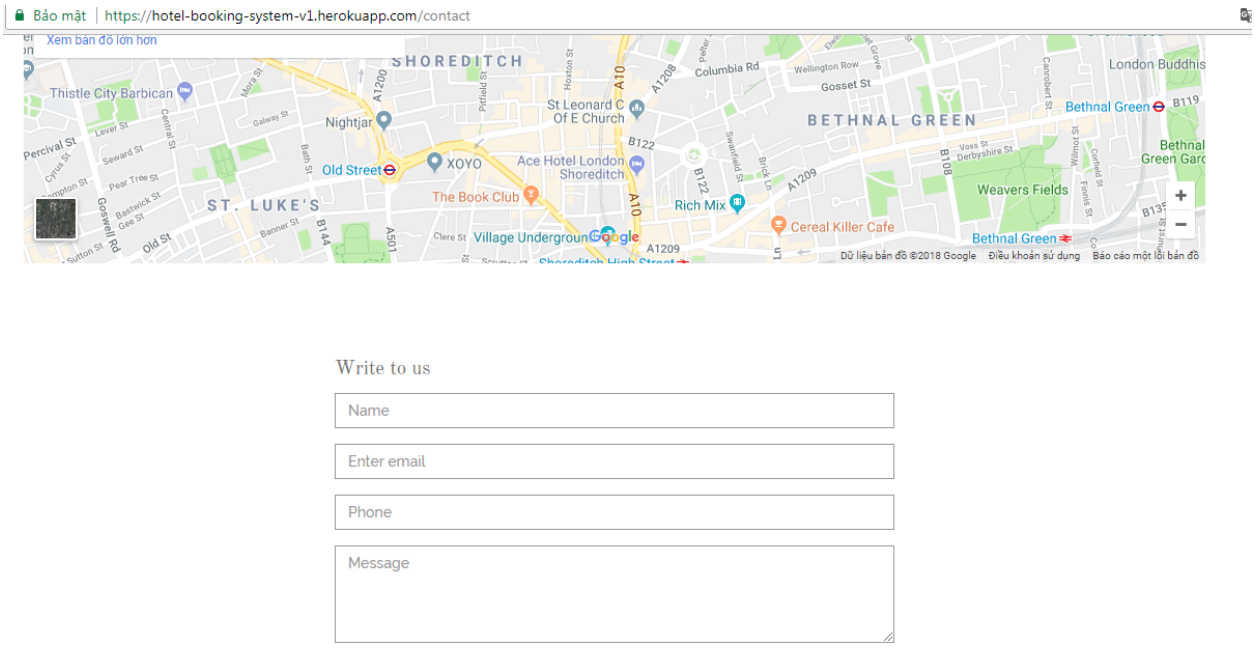


Figure 62: Send Contact Feature

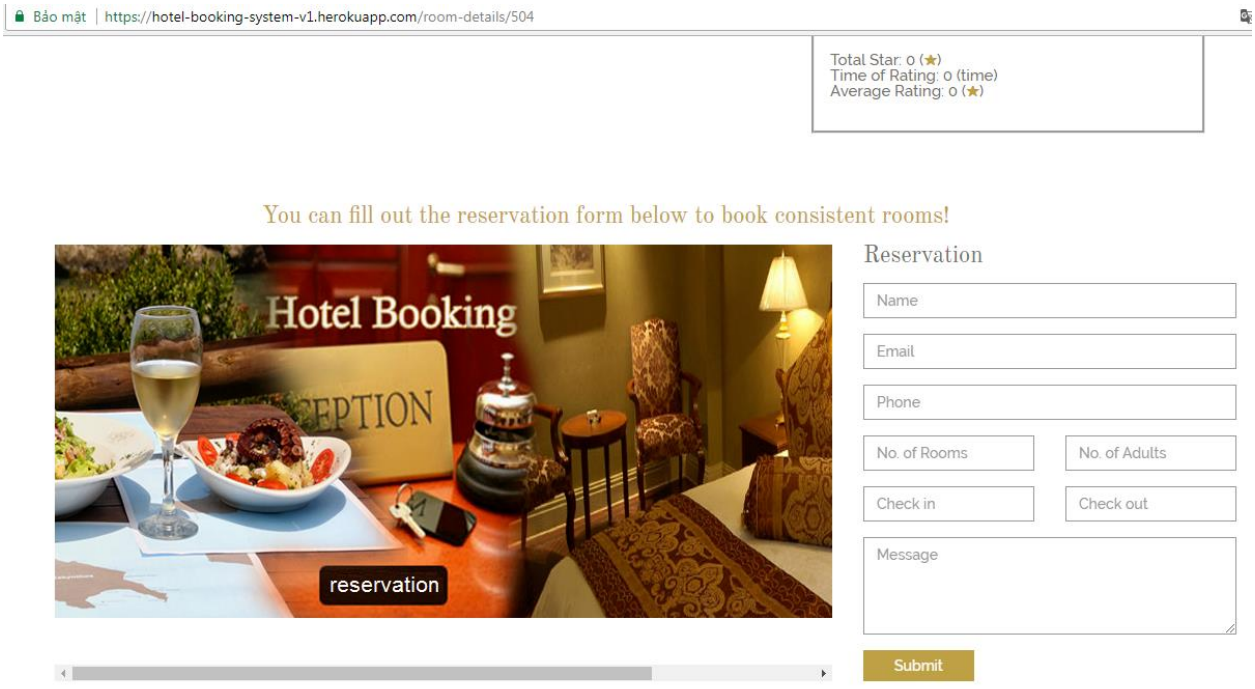


Figure 63: Send Reservation Form feature

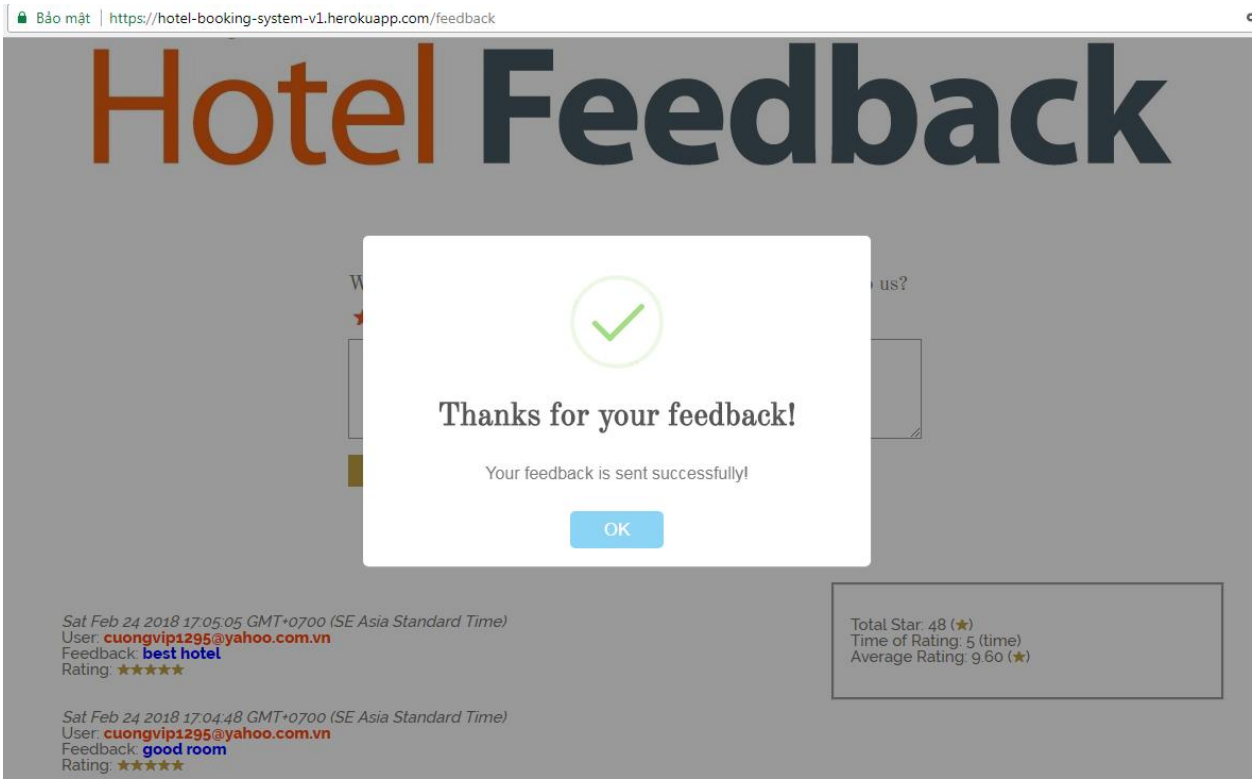


Figure 64: Send Feedback feature

Customer can review his activities where is shown like a transaction history.

Your Activity

Time	Activity	Details	Notes	Response	Content
Thu Dec 21 2017 17:16:27 GMT+0700 (SE Asia Standard Time)	Feedback	Feedback sent successfully!	Rating with 3 ★	Not Yet	View content
Tue Dec 19 2017 18:14:28 GMT+0700 (SE Asia Standard Time)	**Send Contact**	Message sent successfully.	**Waiting for response!**	**Seen**	View content
Tue Dec 19 2017 17:13:35 GMT+0700 (SE Asia Standard Time)	Reservation	Message sent successfully.	Waiting for response!	Seen	View content
Tue Dec 19 2017 17:05:25 GMT+0700 (SE Asia Standard Time)	Reservation	Message sent successfully.	Waiting for response!	Seen	View content
Tue Dec 19 2017 13:30:49 GMT+0700 (SE Asia Standard Time)	Cancel Room	Clicked Cancel Room 101	Waiting for response!	Seen	View content
Tue Dec 19 2017 13:29:39 GMT+0700 (SE Asia Standard Time)	Cancel Room	Clicked Cancel Room 101	Waiting for response!	Seen	View content
Tue Dec 19 2017 13:28:33 GMT+0700 (SE Asia Standard Time)	Cancel Room	Clicked Cancel Room 101	Waiting for response!	Seen	View content
Tue Dec 19 2017 13:27:33 GMT+0700 (SE Asia Standard Time)	Cancel Room	Clicked Cancel Room 205	Waiting for response!	Not Yet	View content
Tue Dec 19 2017 13:23:33 GMT+0700 (SE Asia Standard Time)	Book Room	Booked Room 205	Check in: 2017-12-30 & Check out: 2017-12-31	Not Yet	View content
Tue Dec 19 2017 13:23:19 GMT+0700 (SE Asia Standard Time)	Cancel Room	Clicked Cancel Room 101	Waiting for response!	Not Yet	View content
Tue Dec 19 2017 13:11:25 GMT+0700 (SE Asia Standard Time)	Cancel Room	Clicked Cancel Room 102	Waiting for response!	Seen	View content
Tue Dec 19 2017 13:11:19 GMT+0700 (SE Asia Standard Time)	Cancel Room	Clicked Cancel Room 101	Waiting for response!	Not Yet	View content

Figure 65: Review Activity feature

Administrator cans review or manage all the activities of customers. When admin needs to access the activity collection, the system has to interact with Node.js RESTful API to return activity data for Admin to view as well as update the response status of administrator when the notification or message about this activity is reply.

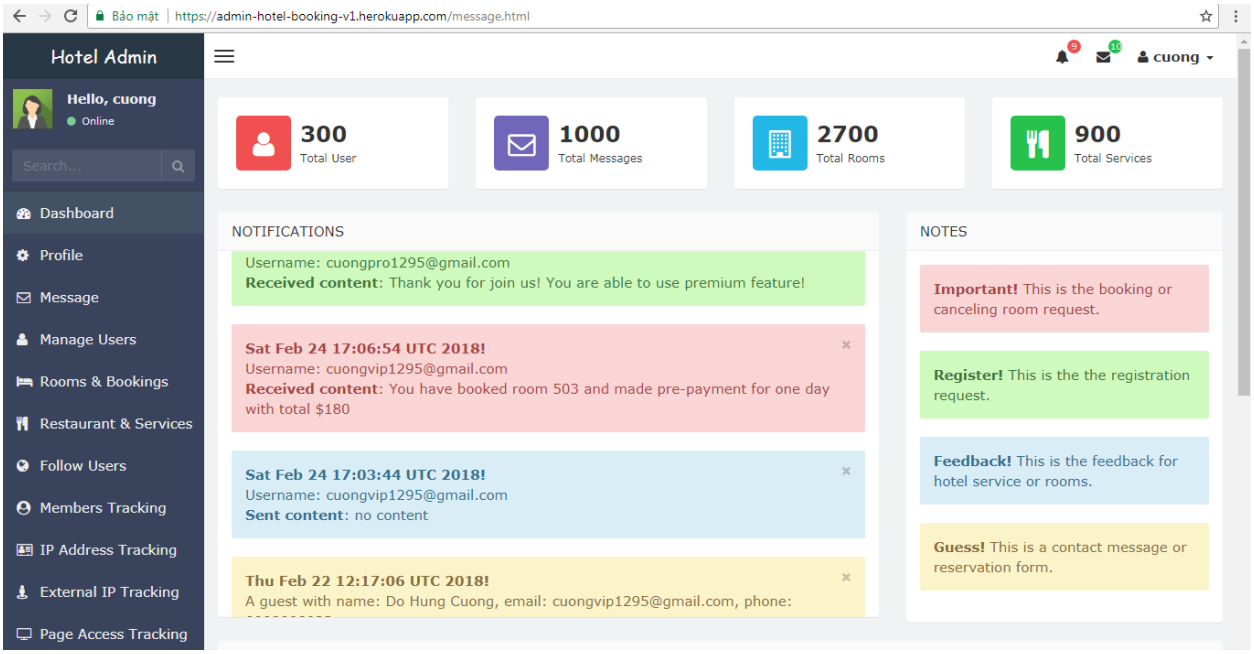


Figure 66: Manage Customer's Activity feature

Besides, when Customer executes an activity, an email is sent to his email using nodemailer library and administrator is able to reply the email using javax.mail library with some available email template that analyzed simply based on information of each activity.

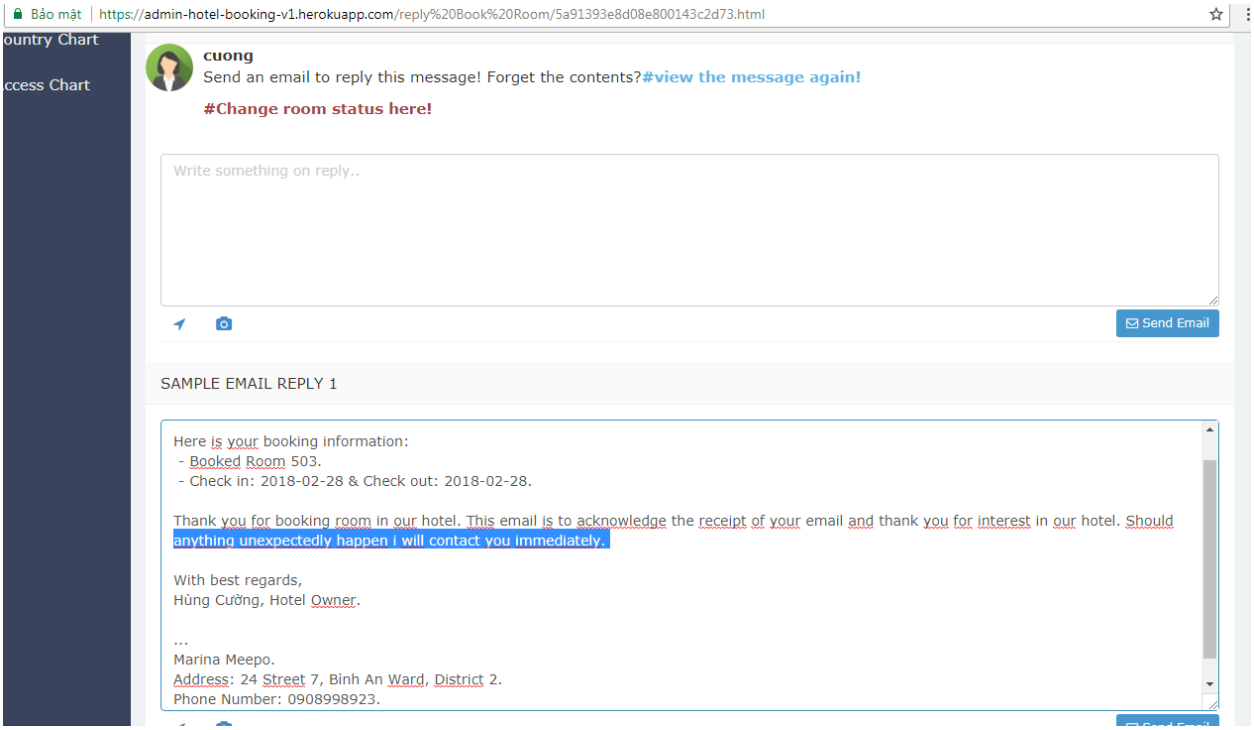


Figure 67: Reply Email feature

These activity and customer data can be used to calculate the customer data collection includes list of rooms booked, rooms canceled, rooms feedback and rating scores with comment, the average rating score and all personal information of customers which will be displayed for administrator.

4.5.3. Tracking User feature

For track users features, the system firstly finds the IP address and external IP of the user by external-ip library for Node.js. Whenever users click on a page, search for item, login, log out or do anything on the website, all actions of users are tracked on client side with Angular 2 and requests are sent to Node.js RESTful API to compute the duration that user stay on each page. The formula to calculate the duration is that the previous information of an action is stored in cookie with the time user click on each page, when the next action occurs, the system will return the duration that equals to the subtraction of time in millisecond of the current page access and one of previous page access. All tracking information is stored in tracking collection of MongoDB.

The user is tracked who is automatically assigned the wished data for room recommendation by the system. Every time user does whatever actions relate to a room such as click room, click image of room, search room, view room's details, book room … The system will update the user's recommendation data include average size of room usually interact, average price, average amenity to MongoDB.

Whenever users accesses recommendation rooms page, Controller then calculates the list of rooms with the best fit with user's recommendation data that means the list of rooms have the closest data compare to the size, price and amenity that customer usually interact. After returned final result, Angular 2 will get it by HTTP Method from REST API provides by Node.js API Route to suggest room for User.

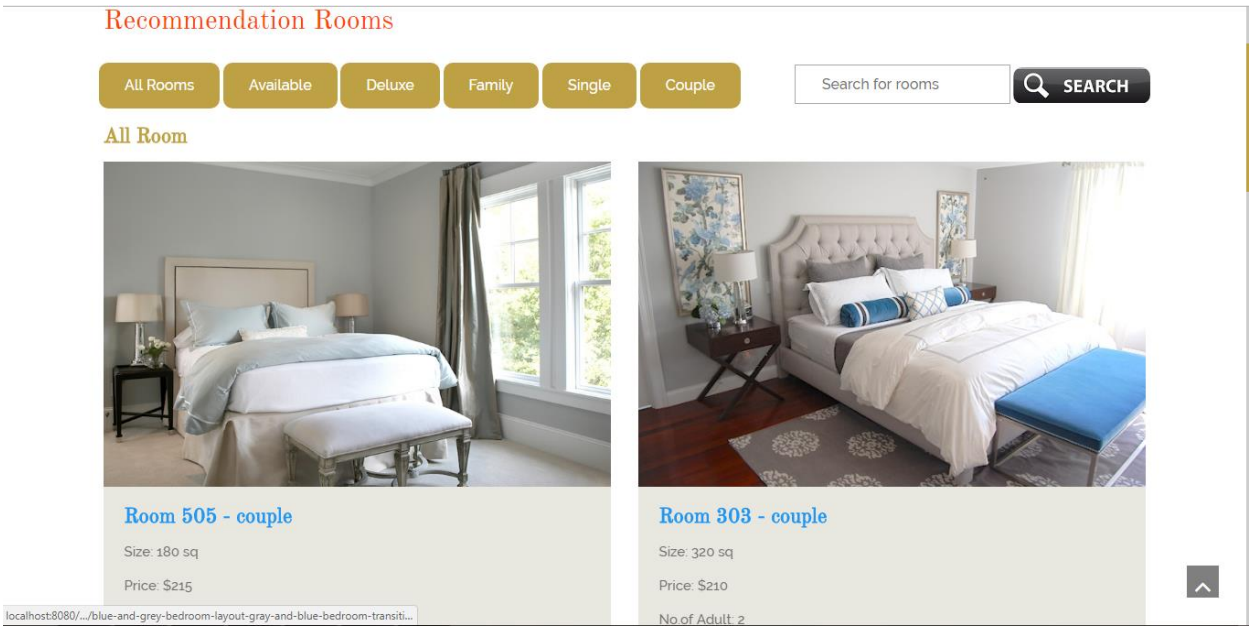


Figure 68: Recommendation Room feature

With the tracked data above, administrator can view tracking information by AngularJS which communicate with the Node.js REST API for retrieving data. On this feature's user interface web page, administrator can view all data at the same time or filter by page, sort data in each field by alphabet, by number or by duration time or search for desired information. These filters will be provides by AngularJS which interact with Node.JS API to retrieve data from MongoDB.

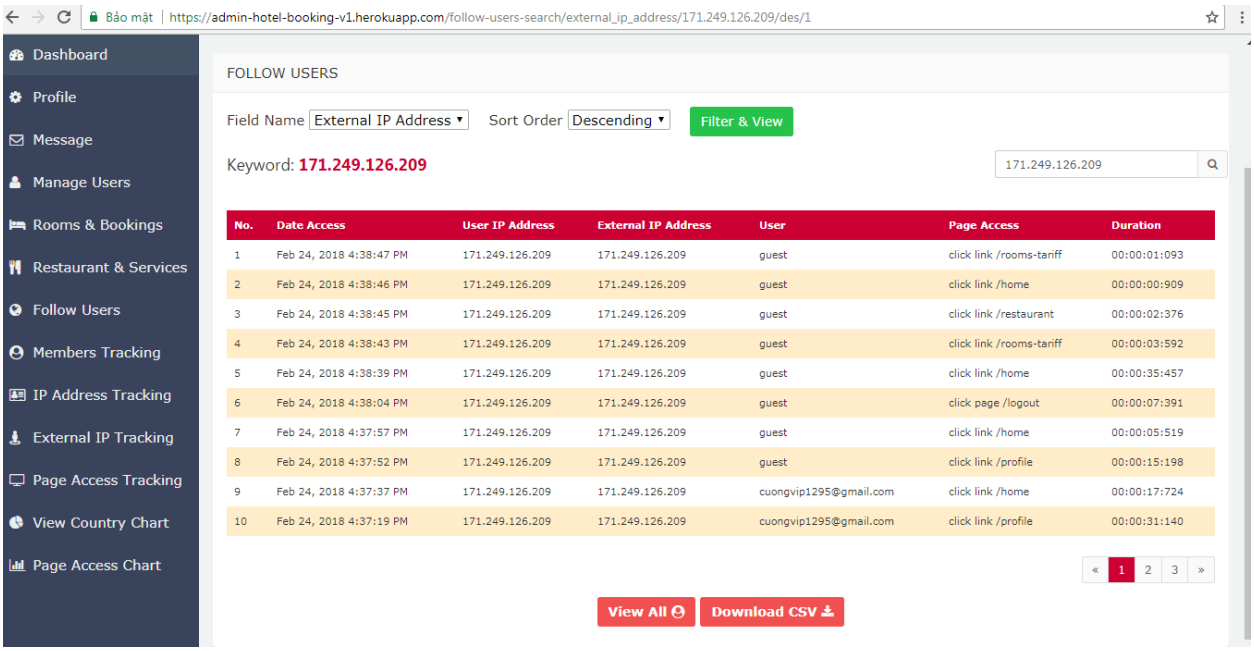


Figure 69: View tracking data feature

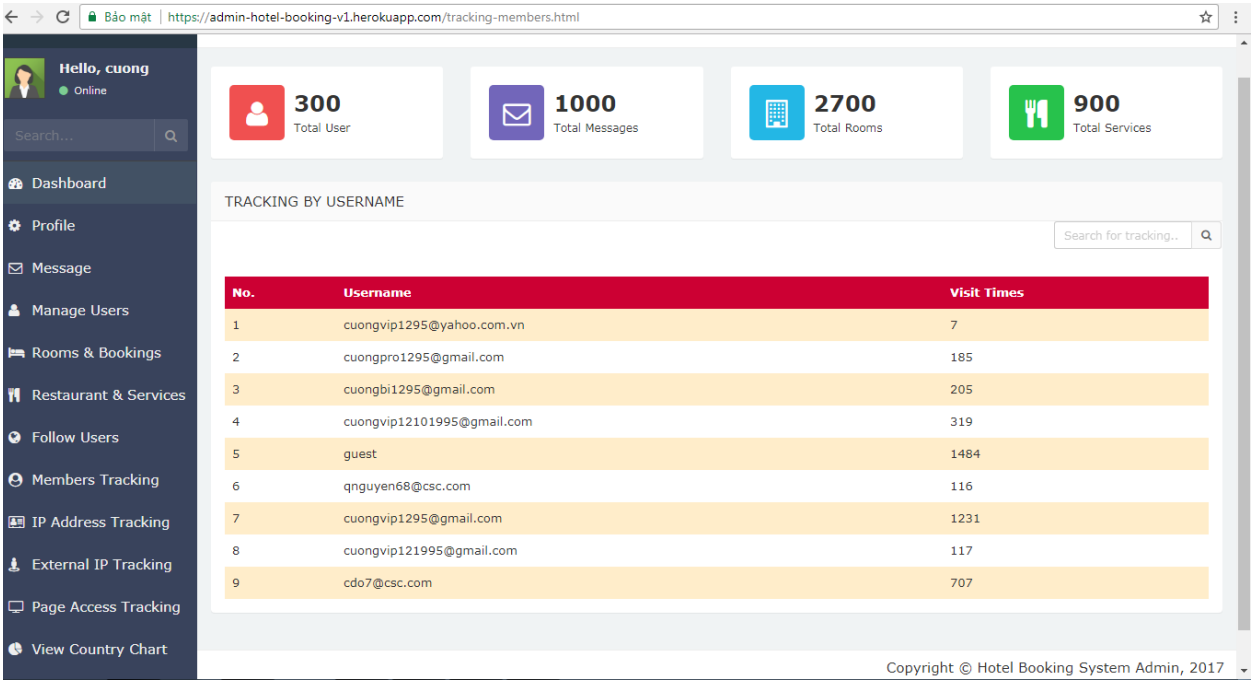


Figure 70: View tracking data feature by username

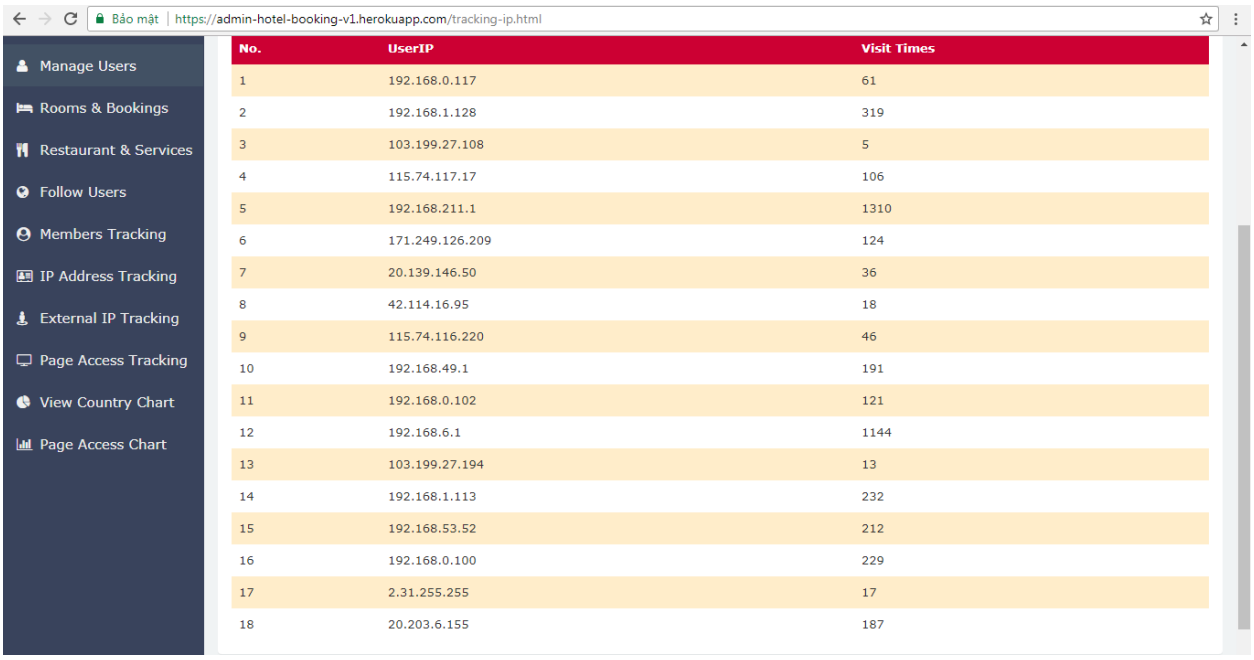


Figure 71: View tracking data feature by IP address

Furthermore, with geoip-api library, Administrator can view location of user with some addition information look up from the IP address of the user.

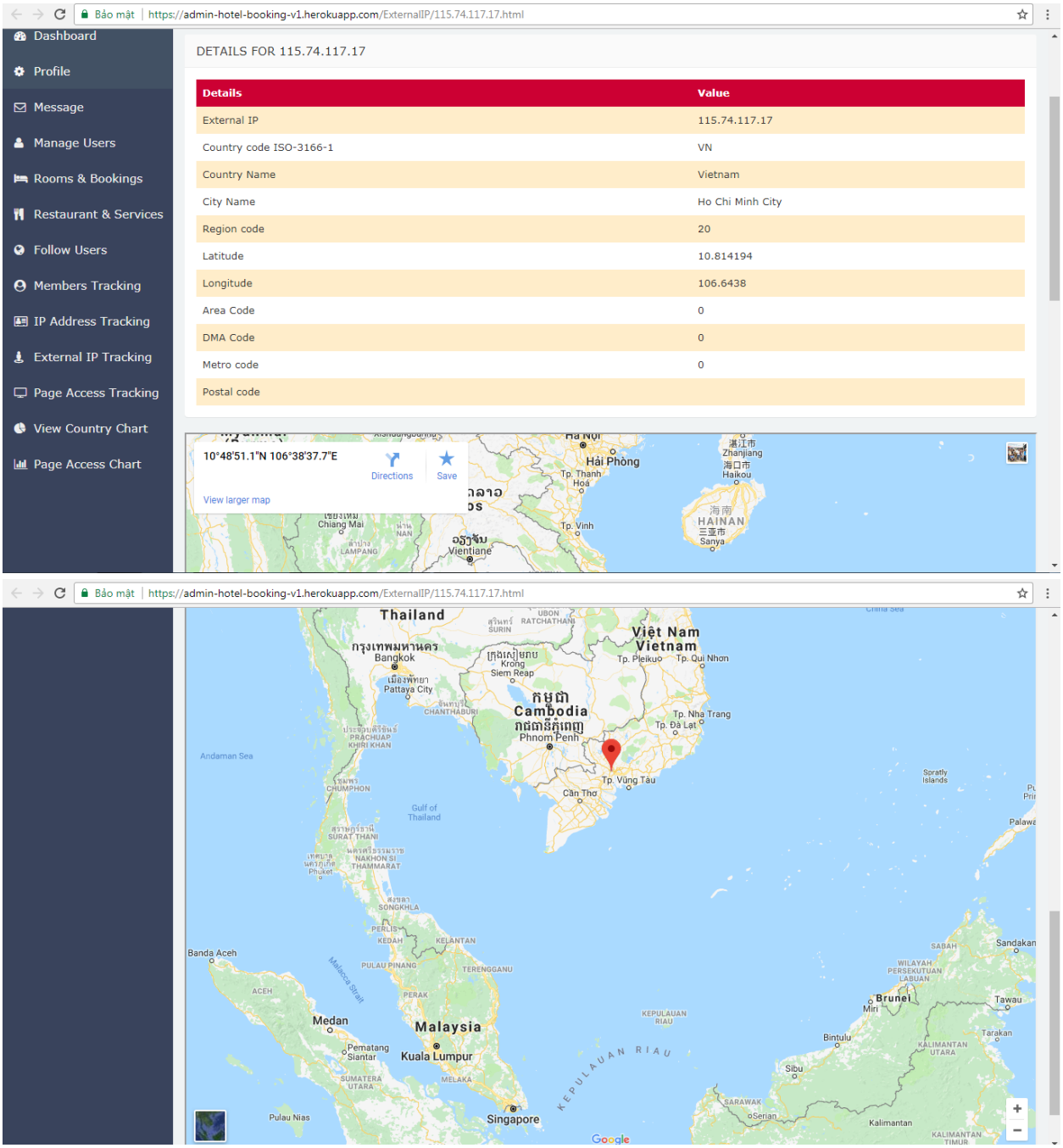


Figure 72-73: Geo Location Lockup Feature

The tracking data can be used to draw chart.

Using AmCharts Javascript library with AngularJS, the system is able to show the pie chart of country and column chart of Tracking Data by username, user IP and page access. This AmCharts Javascript library also support features that allow me to export or download chart image as JPG, PDF, CSV …

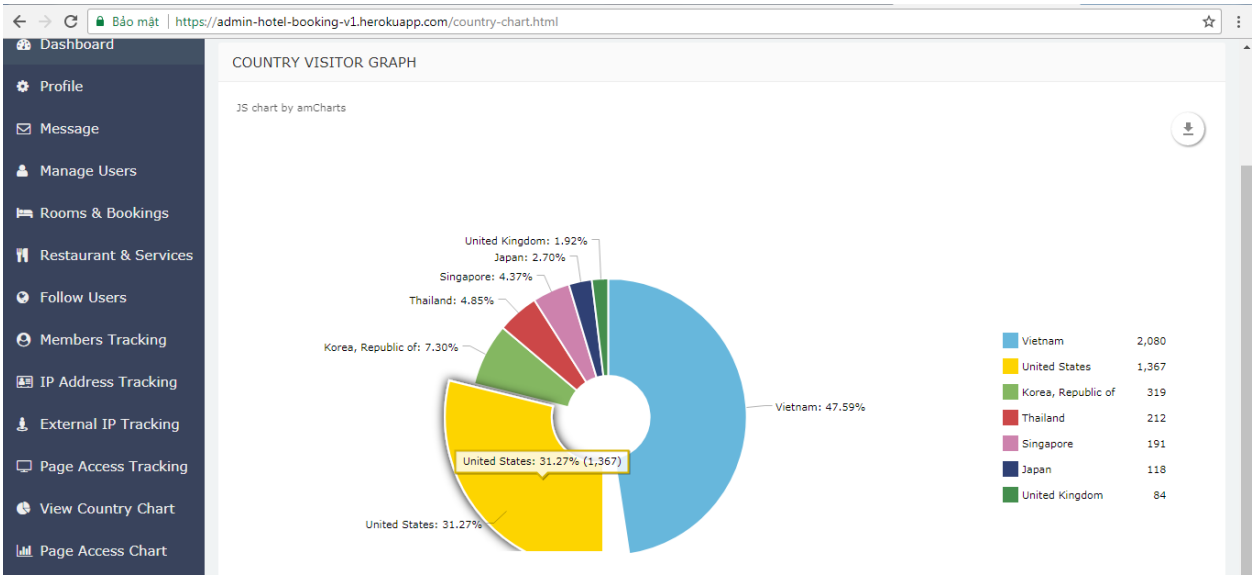


Figure 74: View Pie chart by Country and visit time Feature

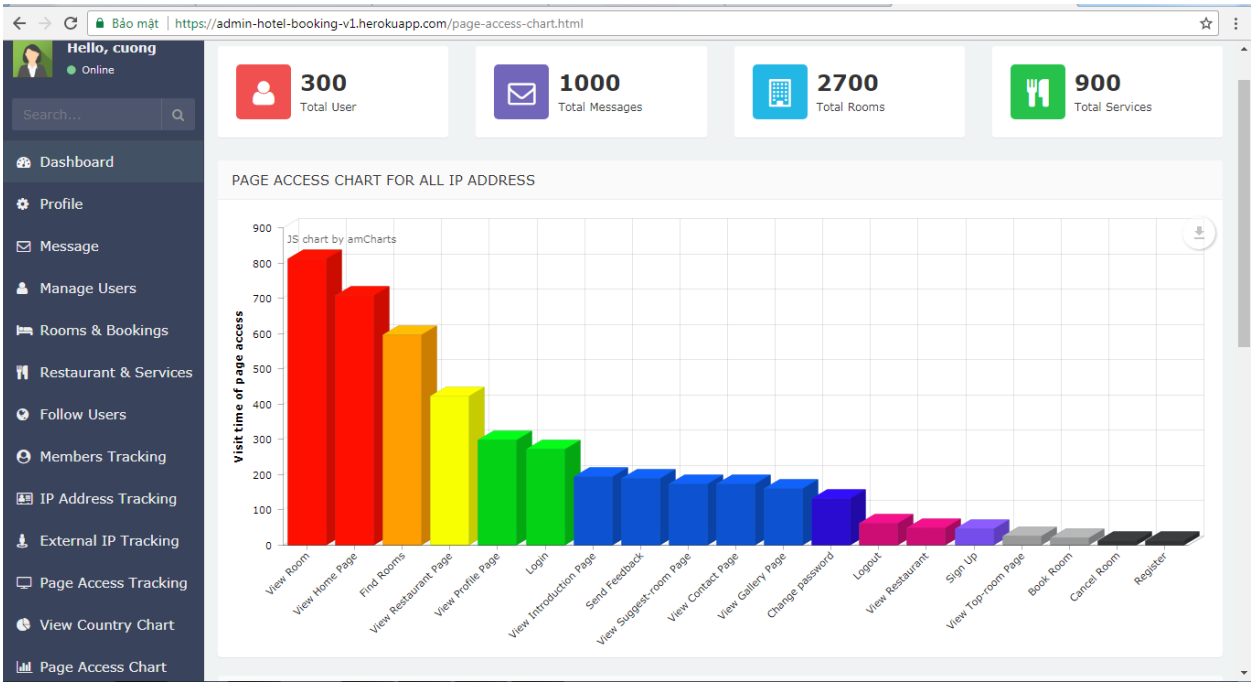


Figure 75: View Column chart by Page Access and visit time Feature

4.5.4. Room feature

Secondly, for SQL database system, we have many features related to room, restaurant and Administrator. The features for room and restaurant are quiet similar but there are more features for room than restaurant.

The administrator can create new room by input information of room such as name, type, size, number of adults, amenities, details then they will become parameters and be sent to Spring AppController which uses method add new room from RoomService.

RoomService calls methods from DAO user HotelRoom Model to check whether input data is acceptable or not. If user input correctly, Hibernate framework will create new session factory to insert into room table of SQL database system.

This process is similar for some features includes edit room, delete room, add new restaurant item, edit restaurant item or delete restaurant item.

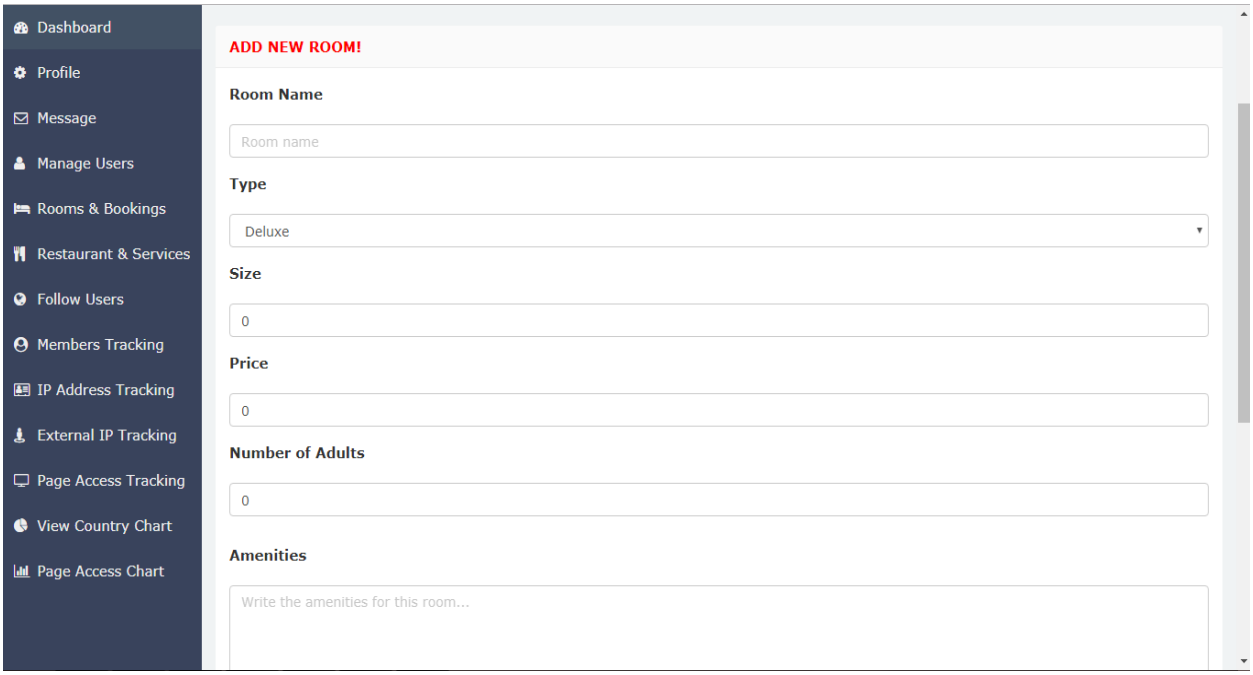


Figure 76: Add new Room Feature

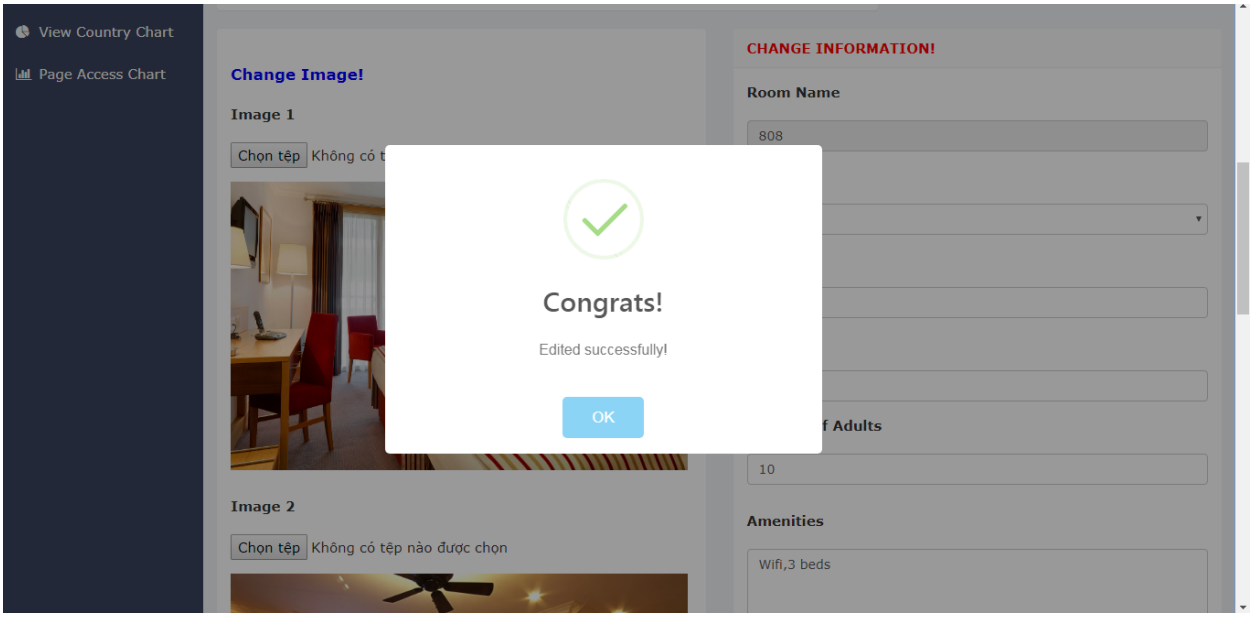


Figure 77: Update Room Feature

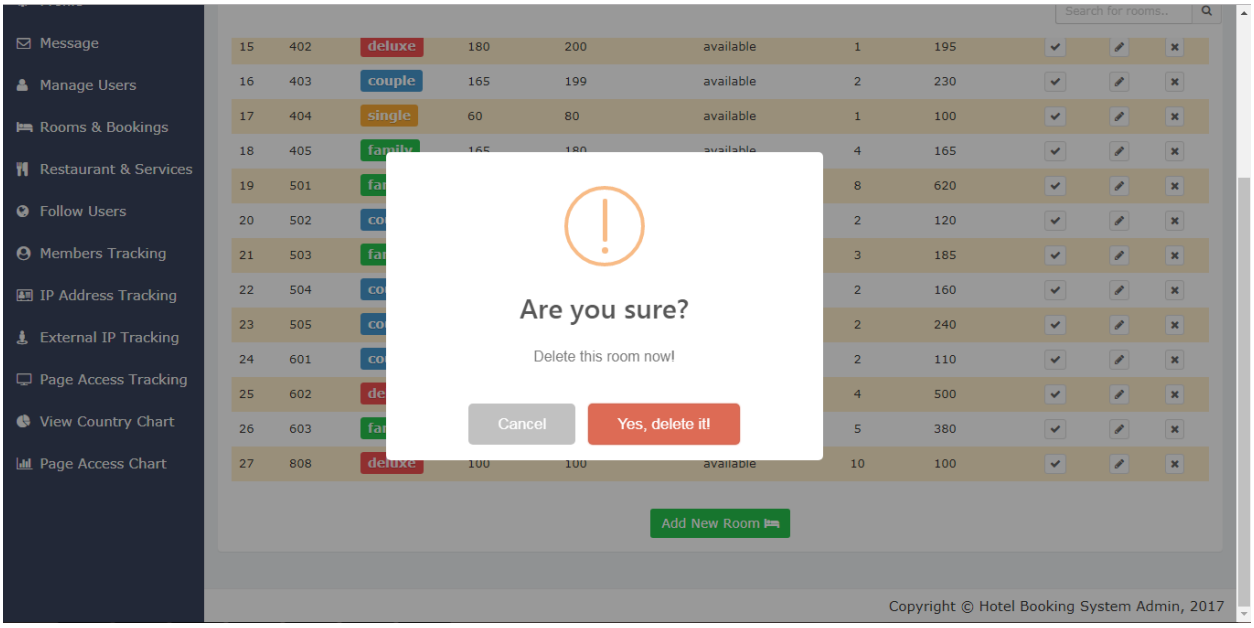


Figure 78: Remove Room Feature

Moreover, administrator can view this list of rooms, view a single room with all more details and information, search or sort the room by keyword or by its attributes. These search and sort functions are written in jQuery that need the list of rooms which was retrieved from SQL through Hibernate, DAO and HotelRoom Model. The process is the same for Restaurant Item.

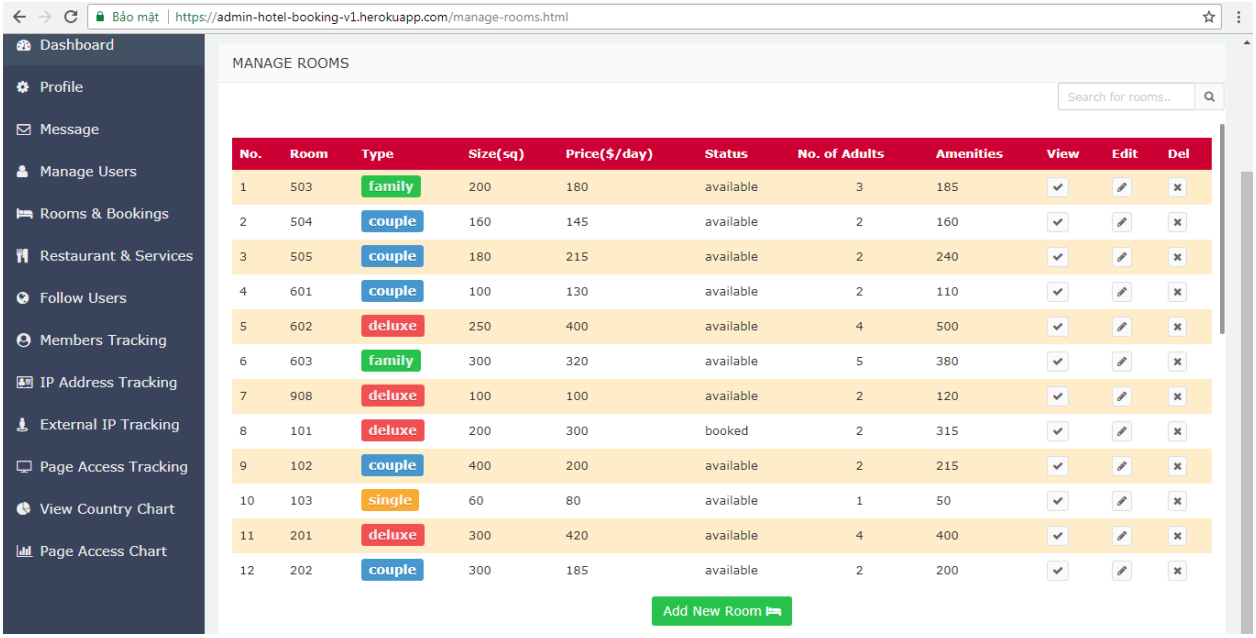


Figure 79: View All Rooms Feature

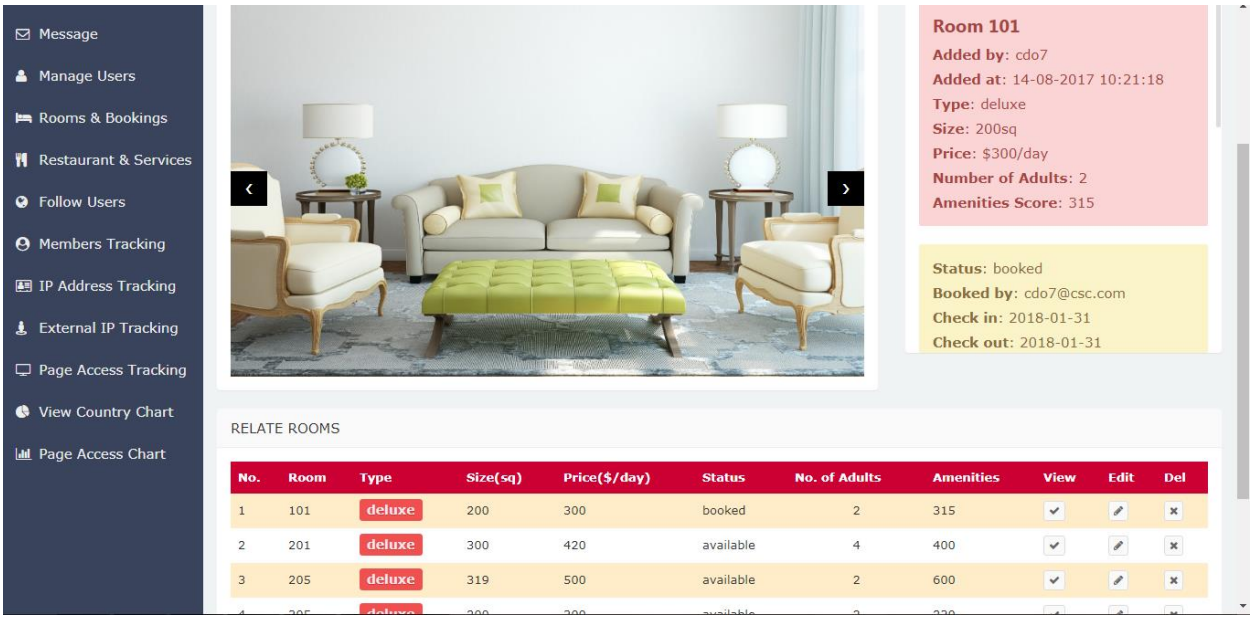


Figure 80: View Single Room with Related Room Feature

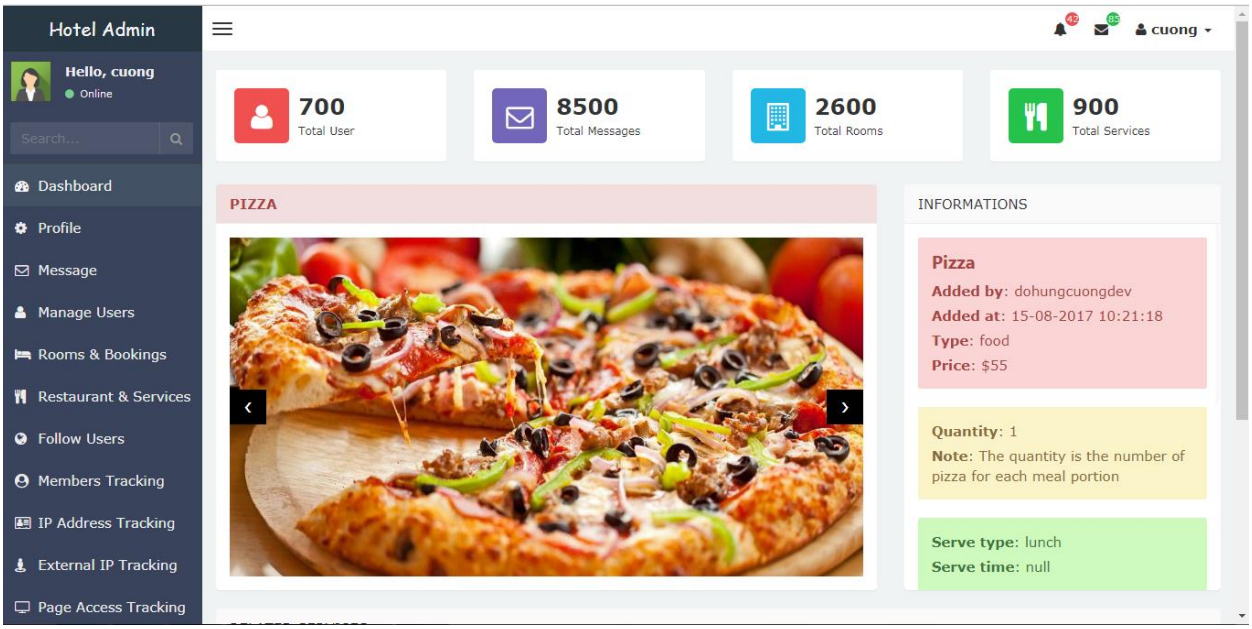


Figure 81: View Single Restaurant Item Feature

Besides, the customer cans view the room and restaurant item on client side with Angular 2 which get data by HTTP method from Spring RESTful Web Service where stored data as JSON. The data is retrieved like the process in section above but return data to REST Controller instead of AppController.

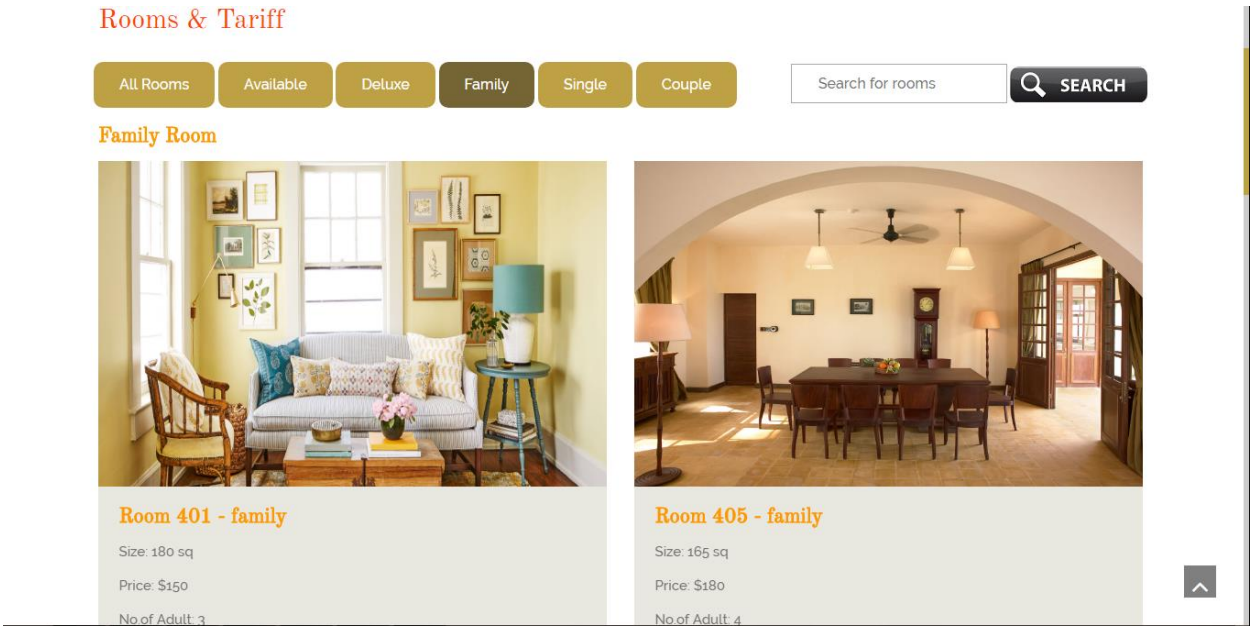


Figure 82: Customer view Rooms Feature.

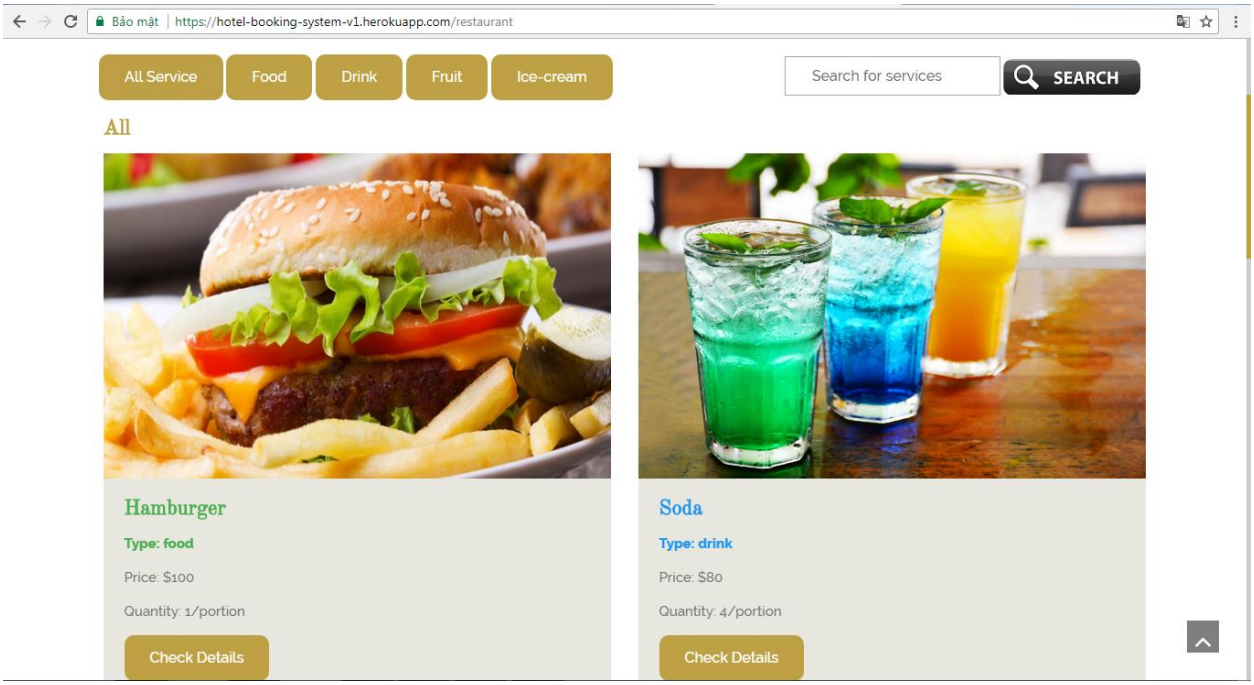


Figure 83: Customer view Hotel Restaurant Feature

Customer can book room, cancel room or send feedback for a room. Of course when they request for this activity, the process update activity will start (see more in **section 4.5.2**). In addition, because these activities relate to room so they need to request Spring RESTful Web Service for retrieving data or update.

If customer books a room, the request is send to Spring REST API and computation is executed. If the account balance is enough to pay for this room, REST API will update the status of the room and subtract the account balance of the user. If not enough money, system will response message to notify the customer.

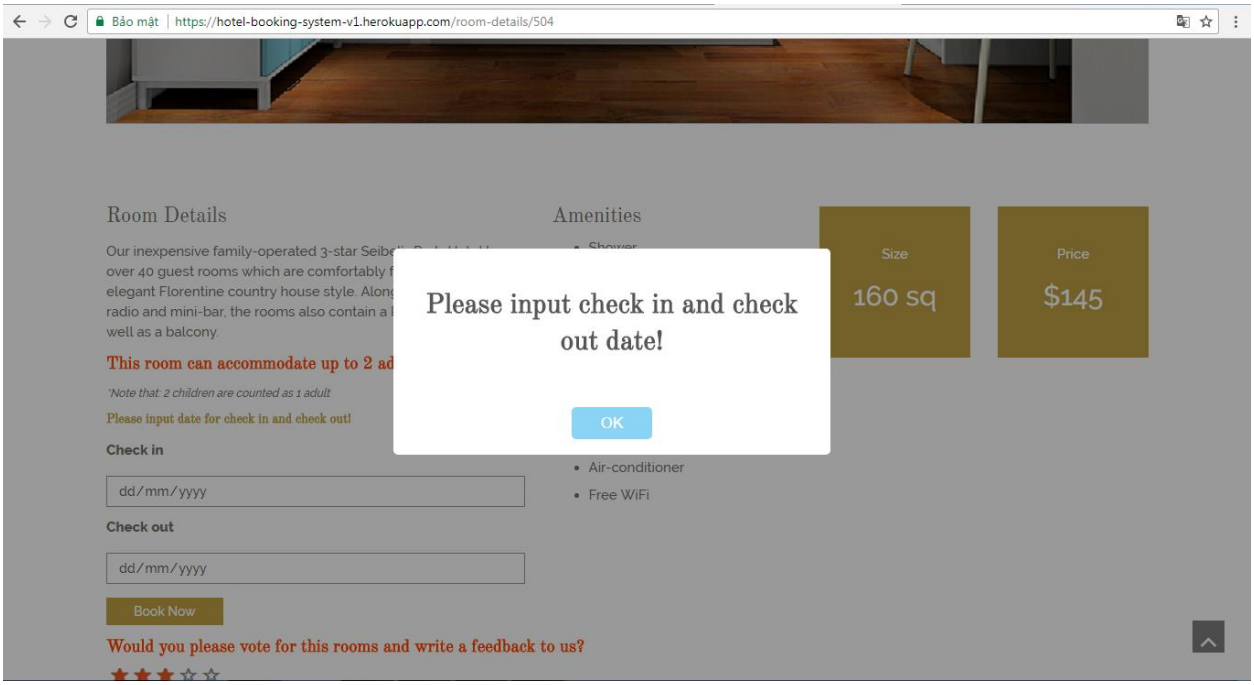


Figure 84: Customer book room feature

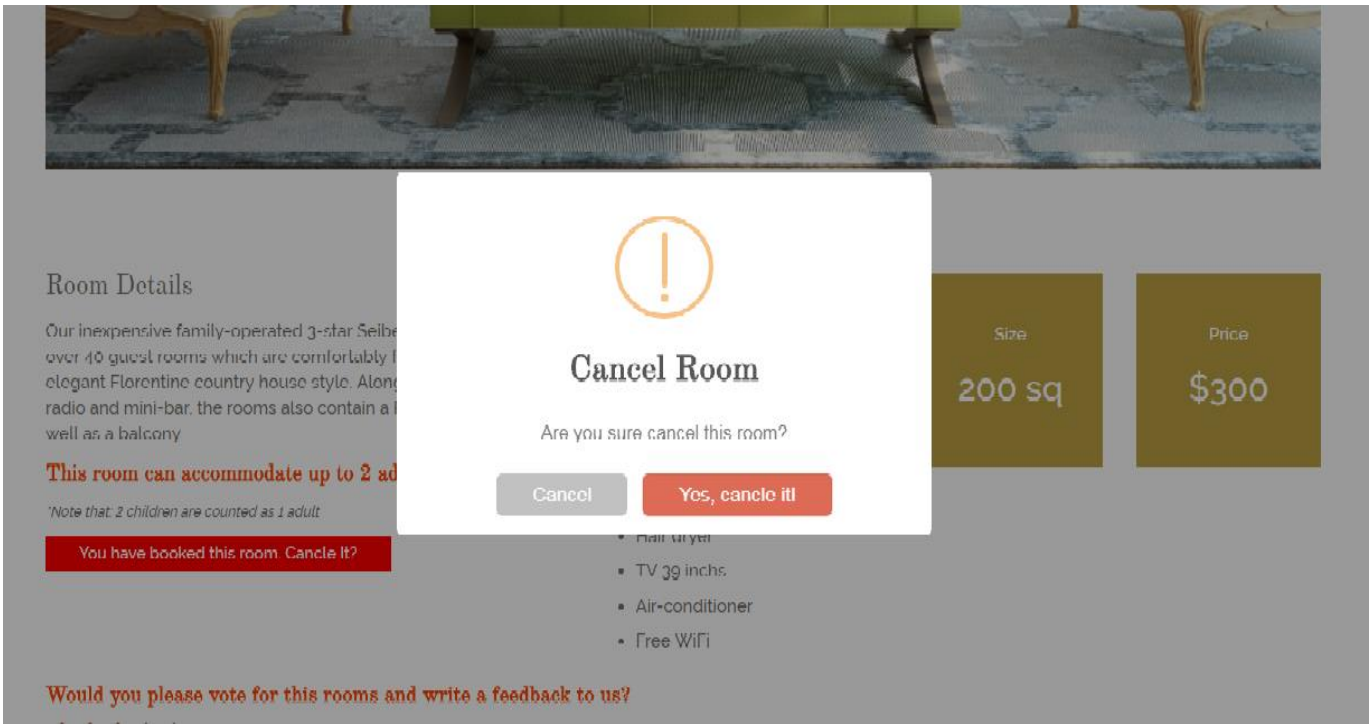


Figure 85: Customer cancel room feature

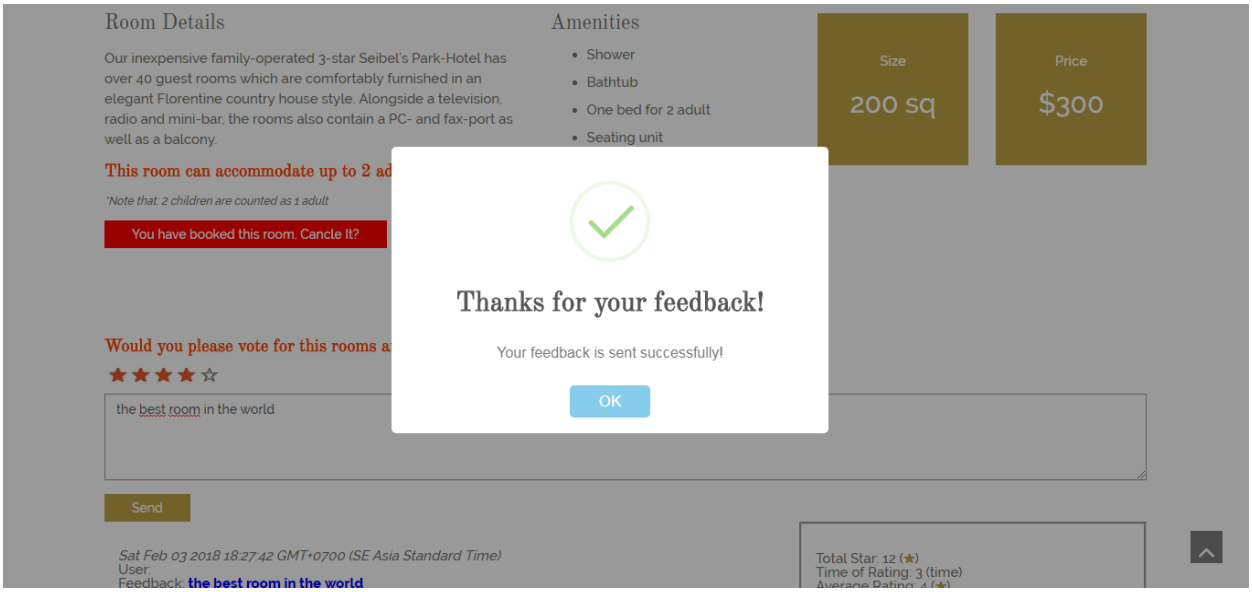


Figure 86: Customer feedback room feature

4.5.5. Admin feature

In my Spring MVC server, the Admin features have no relationship with other application. That means all the features just follow traditional MVC pattern.

After login, Spring framework saved the administrator data as by session, until he logout

Admin can also change password, change personal information, update avatar in profile page.

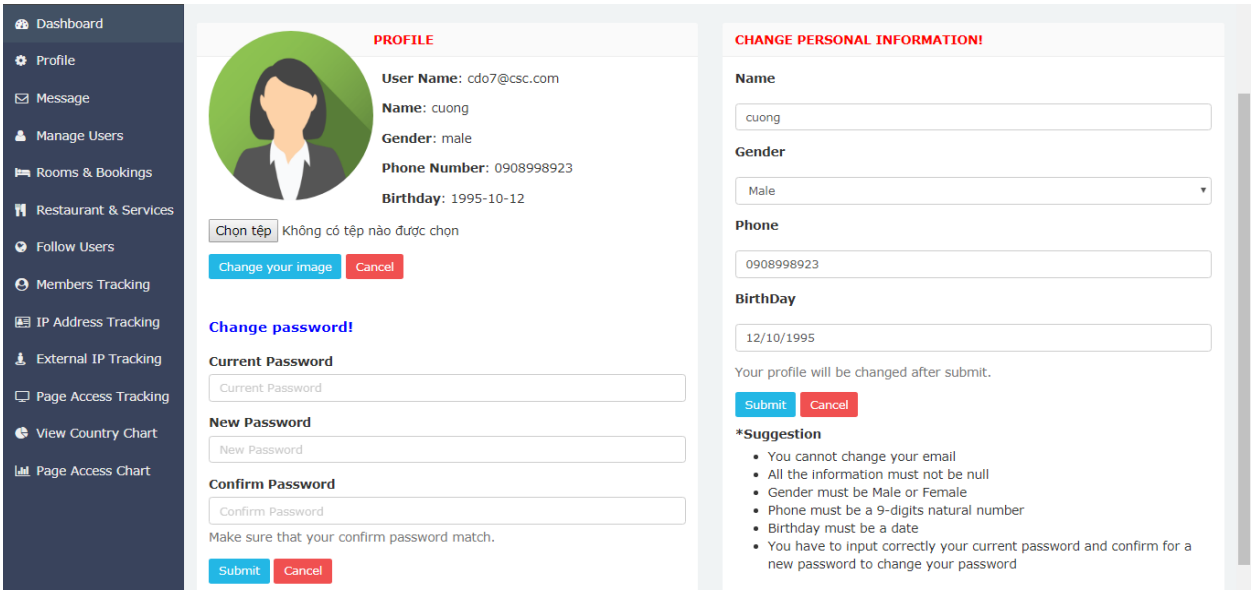


Figure 87: Admin profile feature

4.5.6. Other highlight feature

To easily manage the system, Administrator need a seach feature. This feature helps administrator to input a keyword and search for a lot of things includes message, notification, users, rooms, restaurant items…

This feature is built primarily by jQuery with data retrieved from SQL by Hibernate using 4 models Activity, Customer, HotelRoom, HotelService with their corresponding DAO and Service.

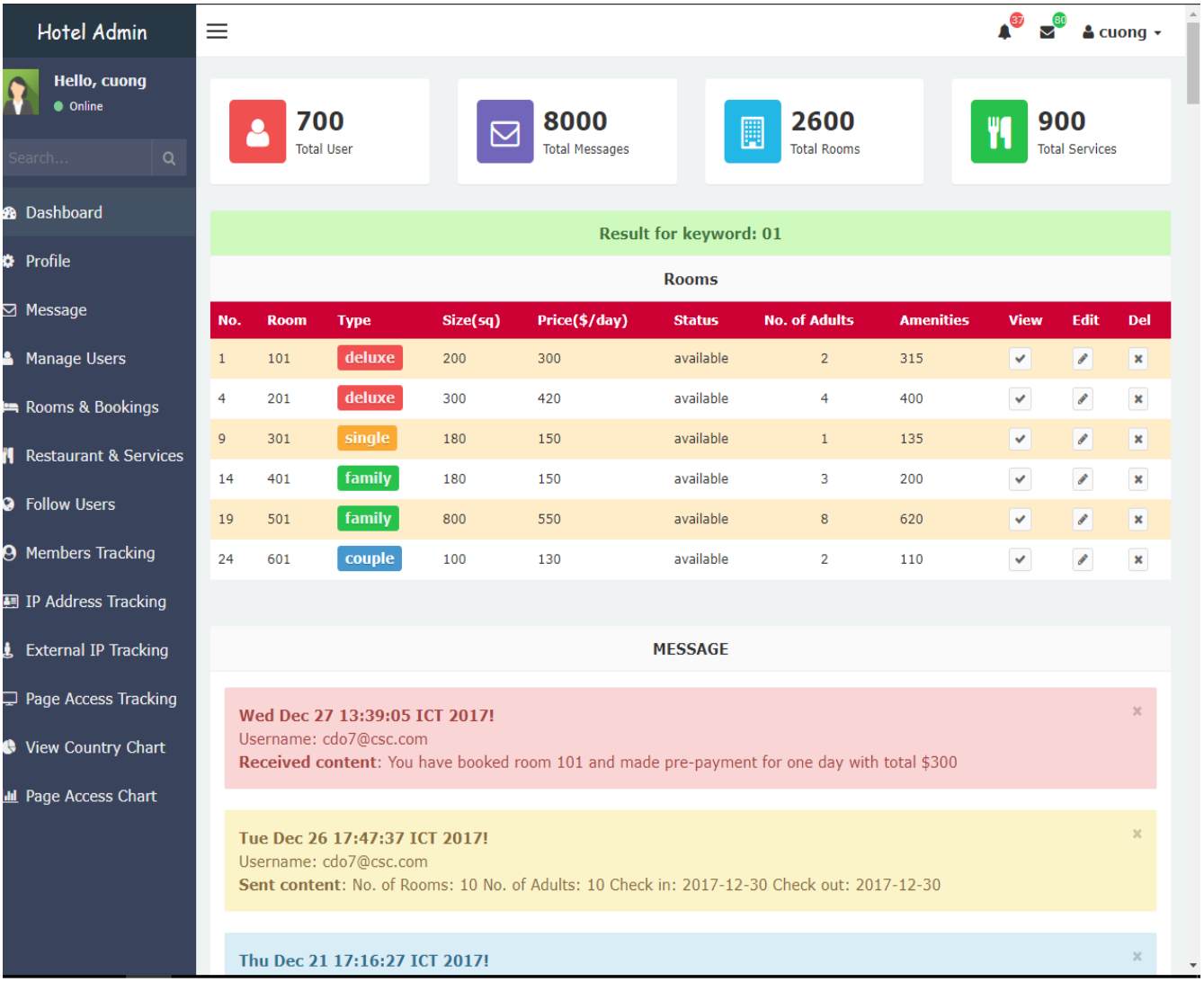


Figure 88: Advance search feature

Chapter V. Experiment and Evaluation

5.1. Experiments

These images below will prove that my system has friendly user interface and work well on many browsers such as IE, CocCoc, Chorme, Firefox, Microsoft Edge. Moreover, it also runs well on many mobile phone operation systems such as android, nokia, ipad. Laptop, desktop.

Run Adminsitrator's page on desktop with Microsoft Edge (**Figure 89**)

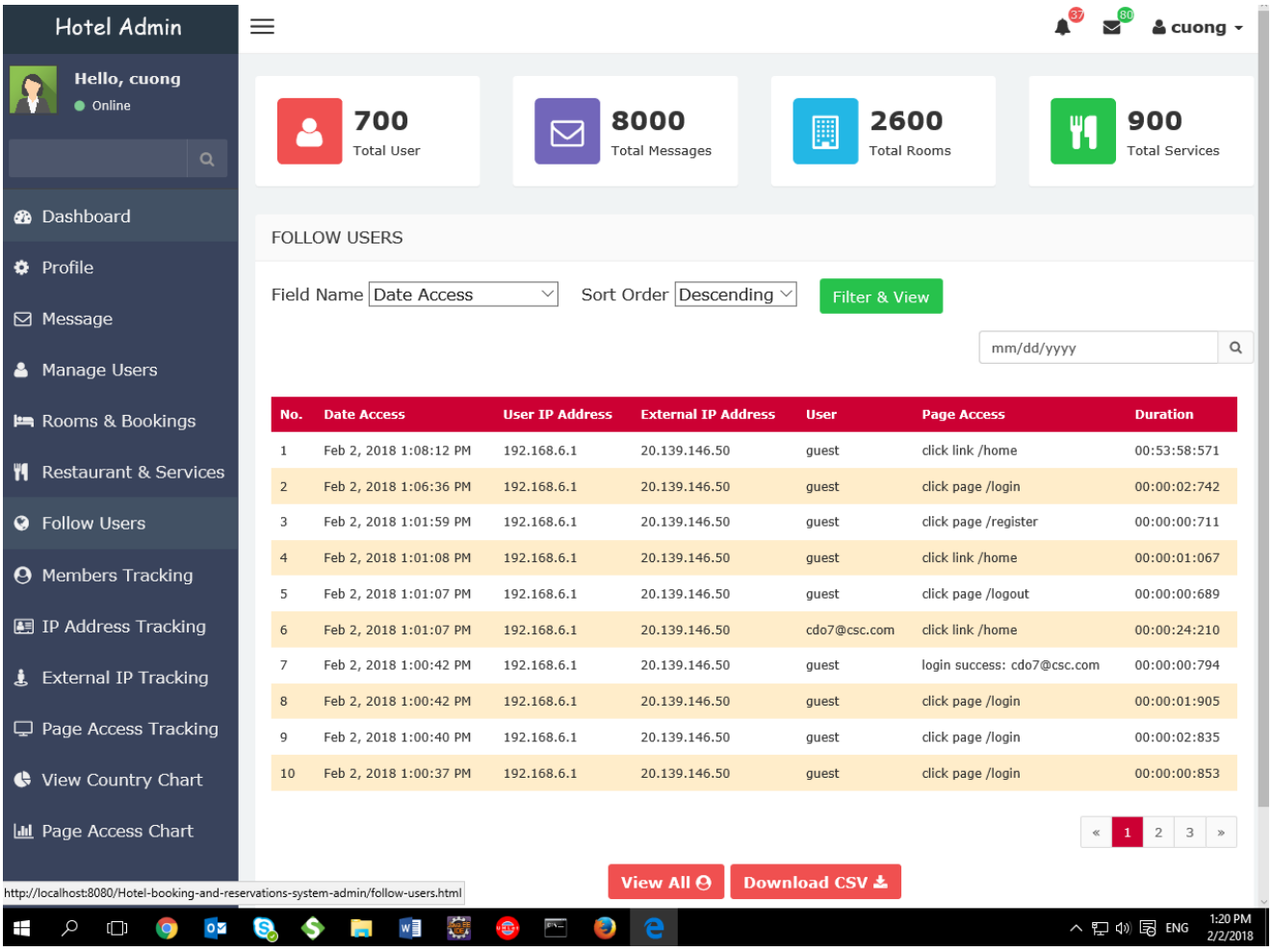


Figure 89: Experiment on desktop with Microsoft Edge

Run Customer's page on laptop with Chome (**Figure 90**)

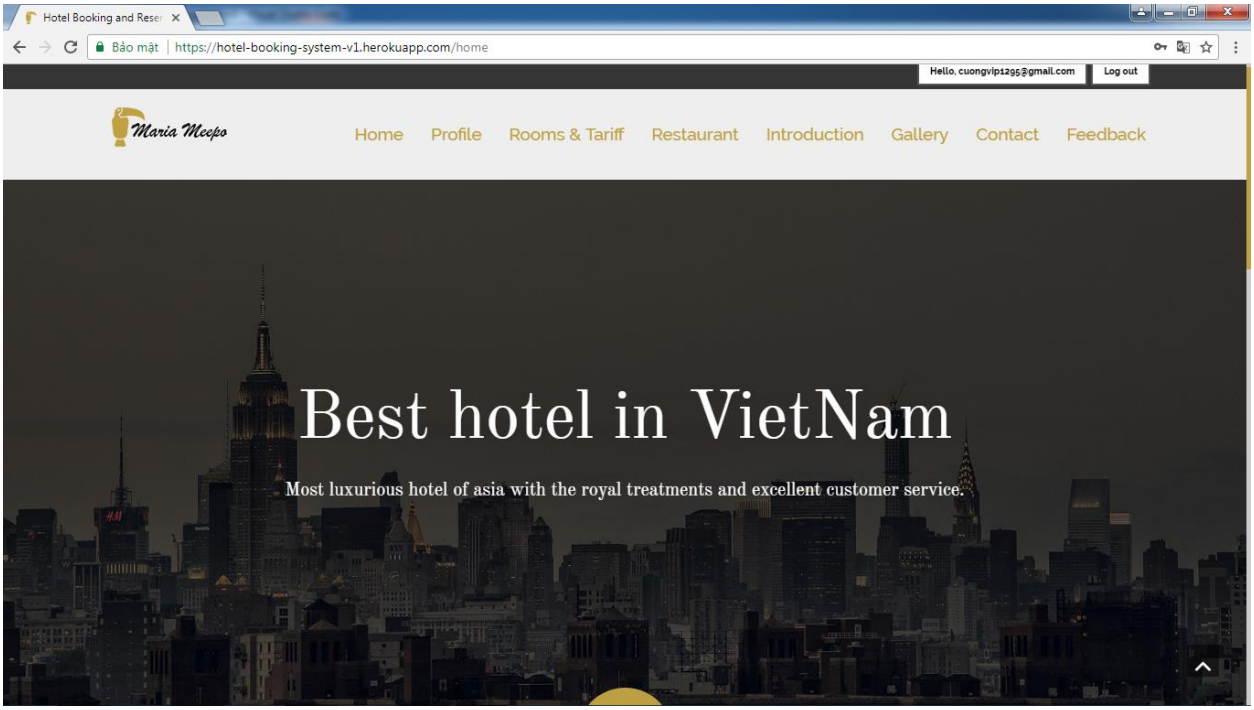


Figure 90: Experiment on laptop with Chome

Running Adminsitrator's page on mobile (Nexus 6P) (**Figure 91**)

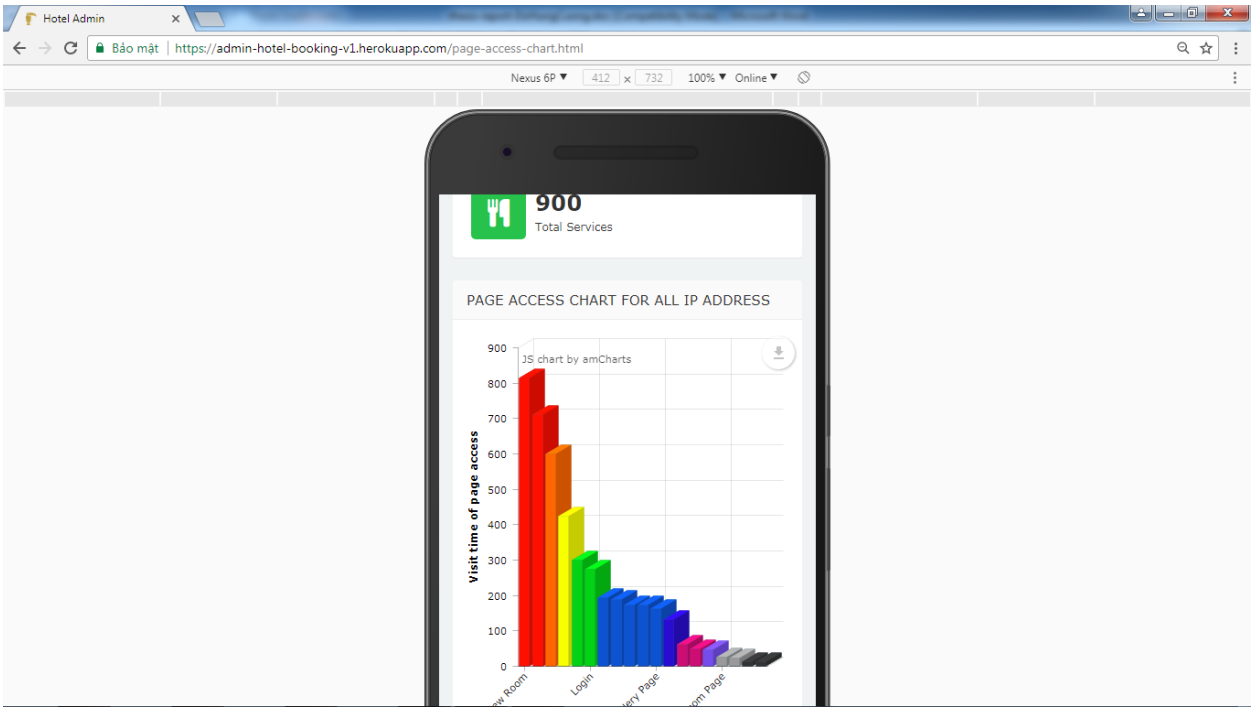


Figure 91: Experiment on Nexus 6P

Running Customer's page on mobile (Iphone 8 plus) (**Figure 92**)

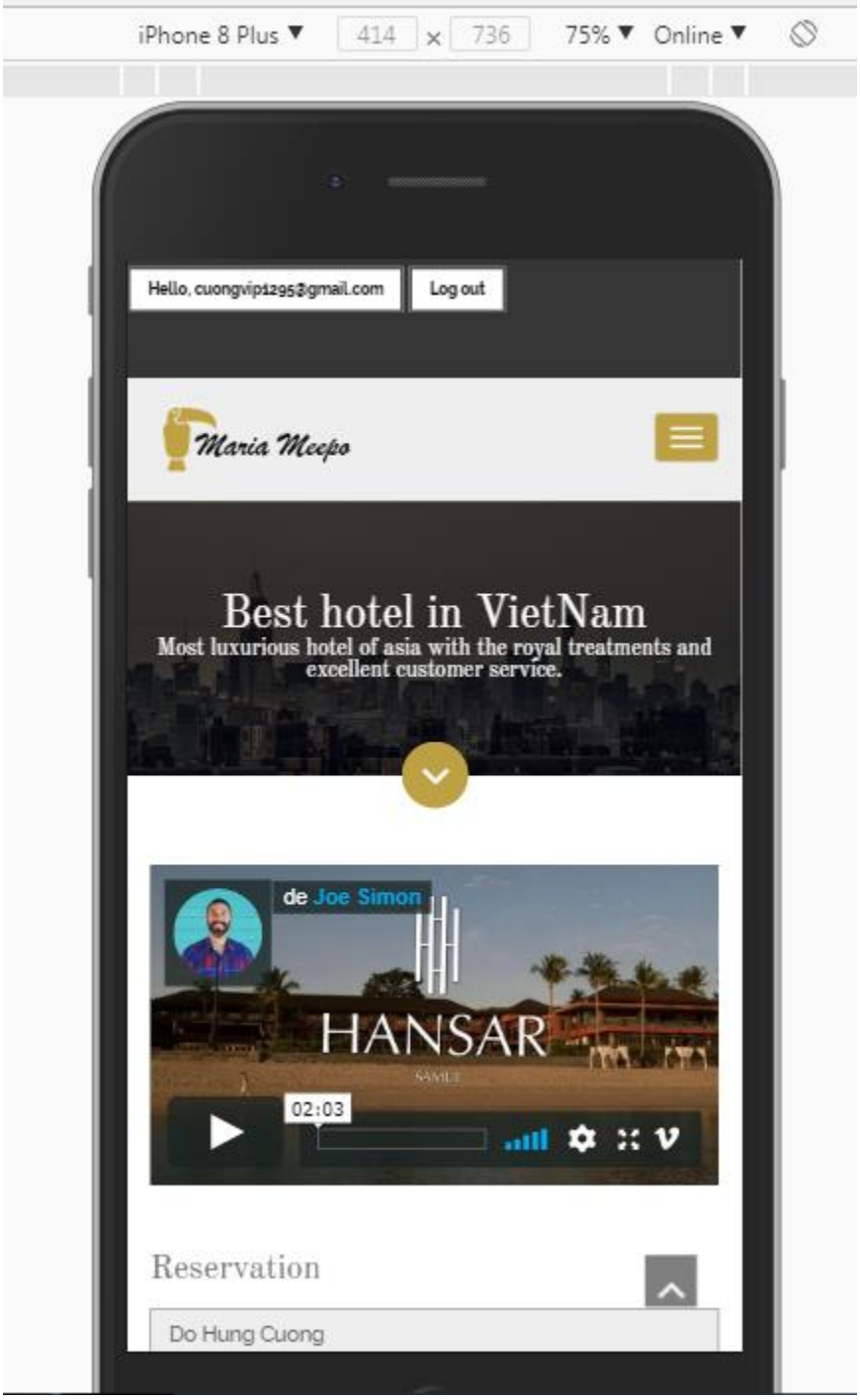


Figure 92: Experiment on Iphone 8 plus

Running Customer's page on Ipad (**Figure 93**)

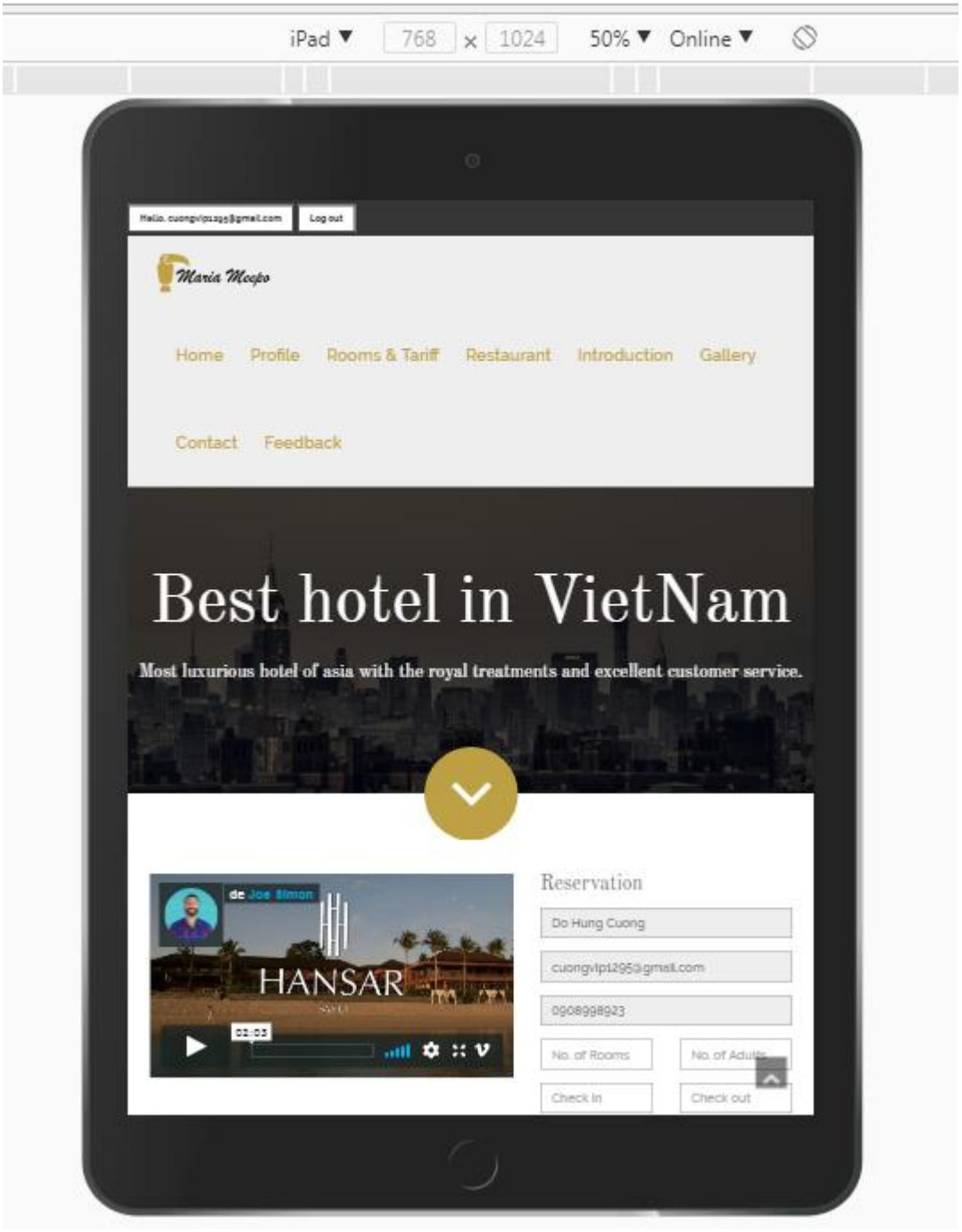


Figure 93: Experiment on iPad

5.2. Evaluation

According to experiment section and all the chapters above, it proves that my system is an online single page web application with high performance and dynamically loading.

A lot of advanced, modern and popular web application technologies are used to implement my system includes Mircoservices, RESTful web service; Java, Spring framework, AngularJS, SQL and Hibernate; Mean stack with MongoDB, Express framework, Angular 2 and Node.js.

My hotel booking websites also have friendly user interface which supports running on many browsers, devices include smart phone, iPad, laptop and desktop.

Moreover, it owns almost features for hotel bookings & reservations management with more than 60 features available (review details in **section 3.2**)

It has ability to track user's behavior so that I can know the pages, the images customers click, total times they visited my websites, the duration they stayed in each page with their username, IP address and location. Even the keyword that customers search or the buttons they click for filter rooms or restaurant services are under controlled.

With all the data collection from customers, my system can automatically recommend rooms for them according to the rooms they had interacted.

So far, the hotel owners can improve their hotel based on customers wish if they apply my system for their hotel business.

Chapter VI. Conclusion

Hotel business is a highly profitable industry but requires huge investment as well as having to meet the customer's demand. However, managing the hotels is not easy, Therefore, hotel management system is really important. The key of hotel business is service which means pleasure the customers. The best management system will bring the highest profit but there are many factors and difficulty to build a good management system.

After applied MEAN stack and Spring MVC to implements Hotel Booking system, I recognize that my system has friendly user interface, high performance with dynamically loading. Moreover, with tracking customer’s behavior feature, you can easily know what customers like and what they don’t in order to improve system day by day to match the customers ‘s wish.

In the future, I will add more features as well as apply machine learning when I have enough data collection.

After thesis, I have learnt a lot of new things, I learnt a lot of technologies such as AngularJS, Angular 2, MongoDB, Spring MVC, Node.js, Express Framework. I can work with lots of frameworks and I have ability to learn new technology. I also had a lot of experience in building single page application. I will improve myself in the future for working in professional environment

Reference

[1] Marriott Hotel, MARRIOTT INTERNATIONAL, INC
Accessed: December 12, 2017
http://www.marriott.com/

[2] Hilton Worldwide Holdings Inc., Hilton Hotels Corporation
Accessed: December 12, 2017
http://www3.hilton.com

[3] IHG Hotel, InterContinental Hotels Group
Accessed: December 12, 2017
https://www.ihgplc.com/

[4] Michael S. Mikowski and Josh C. Powell, Foreword by Gregory D. Benson, Single Page Web Applications JavaScript end-to-end, 2013

[5] Philip Klauzinski, John Moore , Mastering JavaScript Single Page Application Development, 2016

[6] Thomas A. Powell, Ajax: The Complete Reference 1st Edition, 2008

[7] Architecture Overview, Angular.io
Accessed: December 18, 2017
https://angular.io/guide/architecture

[8] What Is AngularJS?, Google
Accessed: December 25, 2017
https://docs.angularjs.org/guide/introduction

[9] By Mehdi, Why Angular? Why Angular 2?, April 11, 2016 3:17 am
Accessed: December 26, 2017
https://blogs.msmvps.com/deborahk/why-angular-why-angular-2/

[10] TypeScript – Overview, Tutorials Point
Accessed: January 03, 2018
https://www.tutorialspoint.com/typescript/typescript_overview.htm

[11] RESTful Web Services Tutorial
Accessed: January 06, 2018
https://www.tutorialspoint.com/restful

[12] Node.js, Node.js Foundation
Accessed: January 06, 2018
https://nodejs.org

[13] Express. Fast, unopinionated, minimalist web framework for Node.js; StrongLoop, IBM, and other expressjs.com contributors
Accessed: January 03, 2018
https://expressjs.com/

[14] Spring - MVC Framework, Tutorials Point
Accessed: January 07, 2018
https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

[15] List of relational database management systems, From Wikipedia, the free encyclopedia
Accessed: January 09, 2018
https://en.wikipedia.org/wiki/List_of_relational_database_management_systems

[16] Fisher, Paul, Murphy, Brian D, Spring Persistence with Hibernate, Second Edition, 2016

[17] MongoDB and MySQL Compared, MongoDB, Inc
Accessed: January 09, 2018
https://www.MongoDB.com/compare/MongoDB-mysql

[18] MongoDB Architecture, MongoDB, Inc.Mongo
Accessed: January 09, 2018
https://www.MongoDB.com/MongoDB-architecture

[19] MEAN stands for, Mean.io contributors
Accessed: January 15, 2018
http://mean.io/

[20] Contributors, Microservices architecture style, 11/28/2017
Accessed: January 16, 2018
https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

[21] 05 May 2008, Understanding Model-View-Controller, Written by Jeff Atwood
Accessed: January 17, 2018
https://blog.codinghorror.com/understanding-model-view-controller/

[22] Spring Framework Reference Documentation;
Authors: Rod Johnson, Juergen Hoeller, Keith Donald, Colin Sampaleanu, Rob Harrop, Thomas Risberg, Alef Arendsen, Darren Davison, Dmitriy Kopylenko, Mark Pollack, Thierry Templier, Erwin Vervaet, Portia Tung, Ben Hale, Adrian Colyer, John Lewis, Costin Leau, Mark Fisher, Sam Brannen, Ramnivas Laddad, Arjen Poutsma, Chris Beams, Tareq Abedrabbo, Andy Clement, Dave Syer, Oliver Gierke, Rossen Stoyanchev, Phillip Webb;
Accessed: January 11, 2018
https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html

[23] Hibernate - ORM Overview, Tutorials Point
Accessed: February 02, 2018
https://www.tutorialspoint.com/hibernate/hibernate_quick_guide.htm

[24] Leonard Richardson, Sam Ruby, Mike Amundsen, RESTful Web APIs Services for a Changing World, Publisher: O'Reilly Media, 2013

[25] What are microservices?, Chris Richardson
Accessed: February 02, 2018
http://microservices.io

[26] Sandro Pasquali, Mastering Node.js, 2013

[27] J Sharma & Ashish Sarin, Getting started with Spring Framework: a hands-on guide to begin developing applications using Spring Framework 3rd Edition, CreateSpace Independent Publishing Platform; 3 edition, 2016

[28] Valentin Bojinov, RESTful Web API Design with Node.js - Second Edition, 2016

[29] Mario Romano, Using Spring and Angular for Web Application, 2017

[30] Sourabh Sharma, Mastering Microservices with Java, Tuesday, 2017

[31] Yakov Fain & Anton Moiseev, Angular 2 Development with TypeScript, Manning Publications Company, 2016

[32]: Joris Hermans, Web Development with Node.js, MongoDB and Express, 2017

[33]: Amos Q. Haviv & Adrian Mejia & Robert Onodi, Web Application Development with MEAN, 2016.

Appendix

With some class diagrams contain many relationships as well as the variables, methods or functions in each class are too much and can only be drawn on huge pieces of paper, I decided to put those in this appendix section.

Appendix A. Spring Model class diagram

Figure 94 to 99 below will show the large model class diagram in **section 4.4.1.1** with many relationships, methods and variables

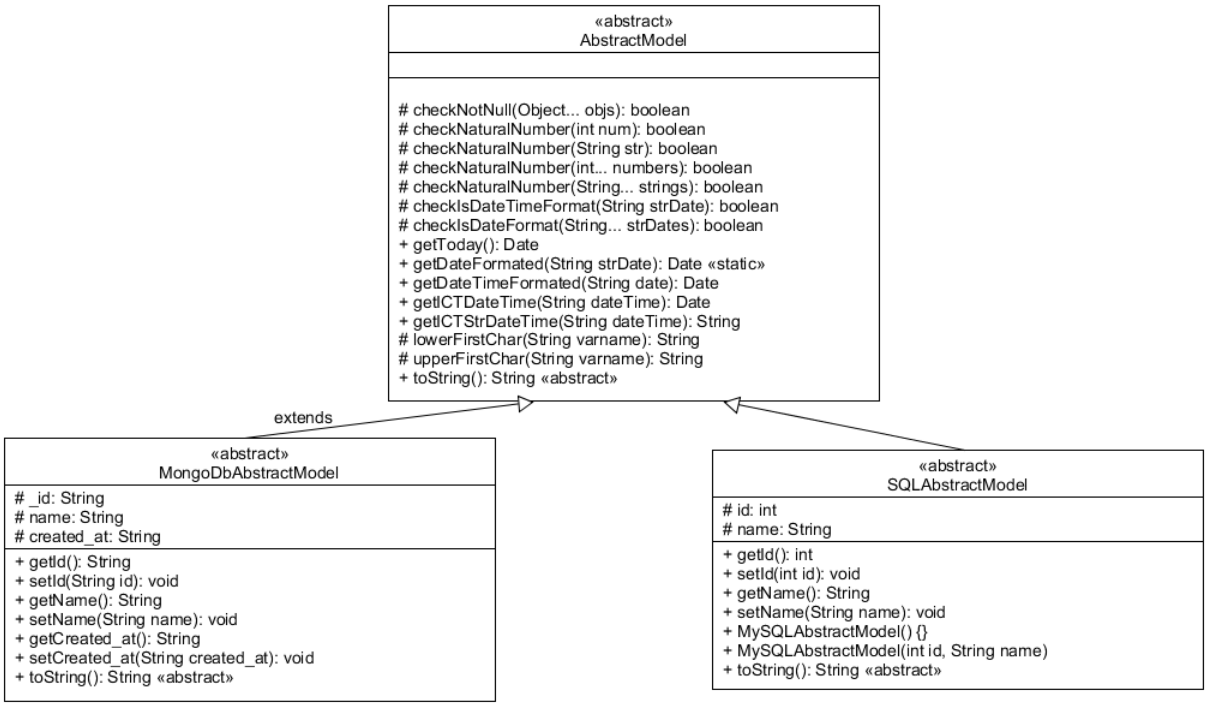


Figure 94: Spring Abstract Model class diagram

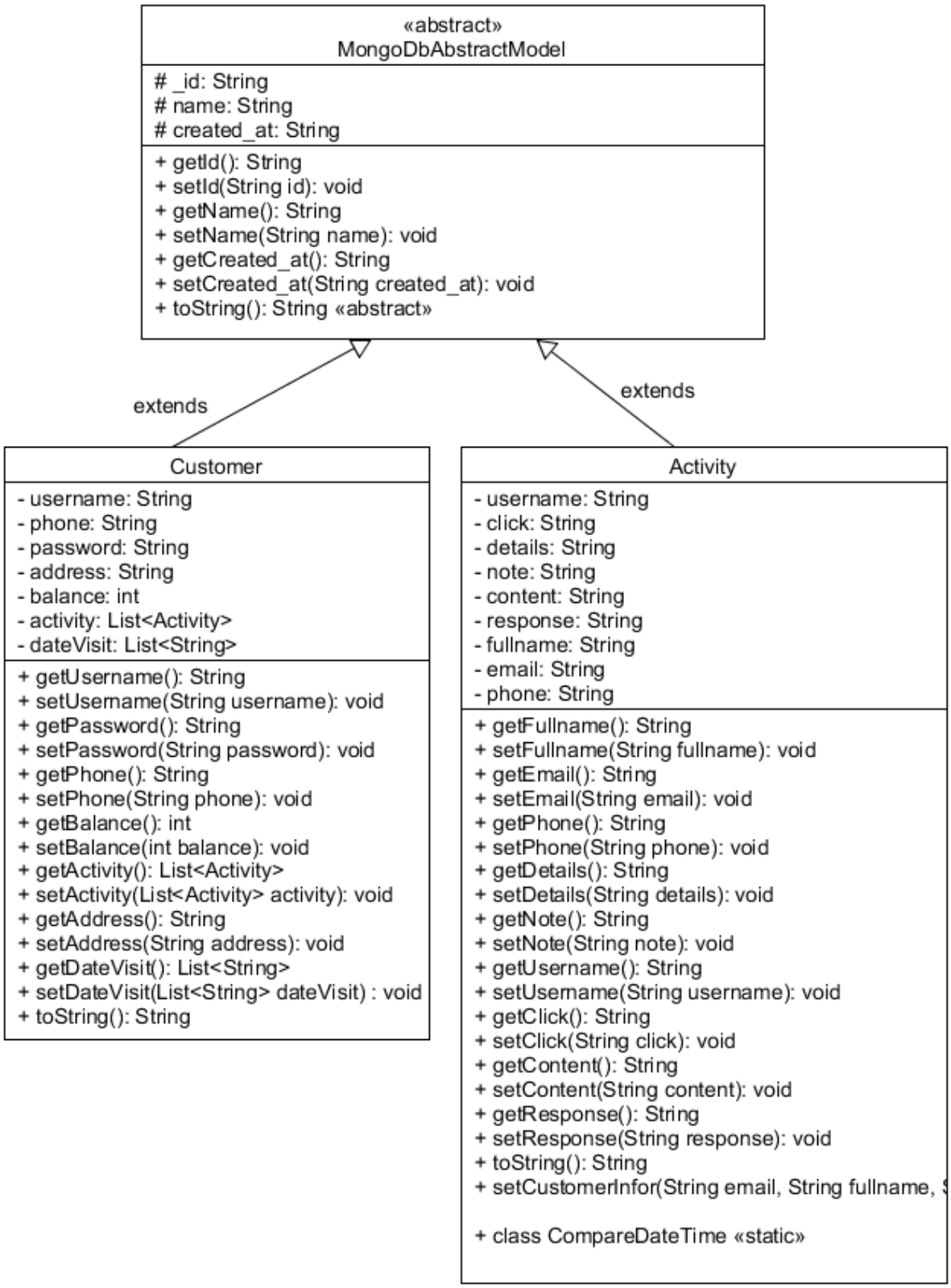


Figure 95: Spring MongoDB Model class diagram

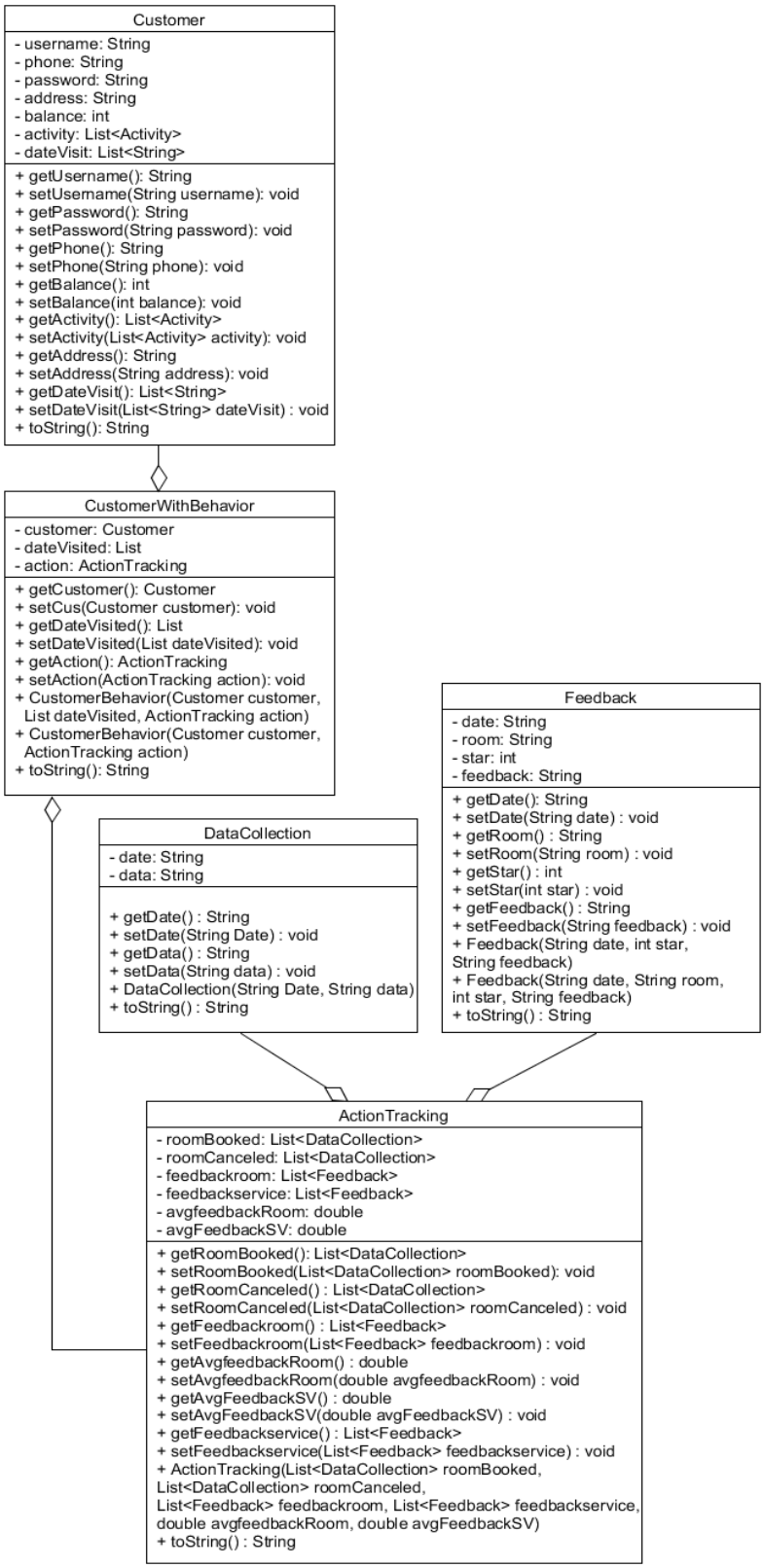


Figure 96: Spring Customer and Activity Model class diagram

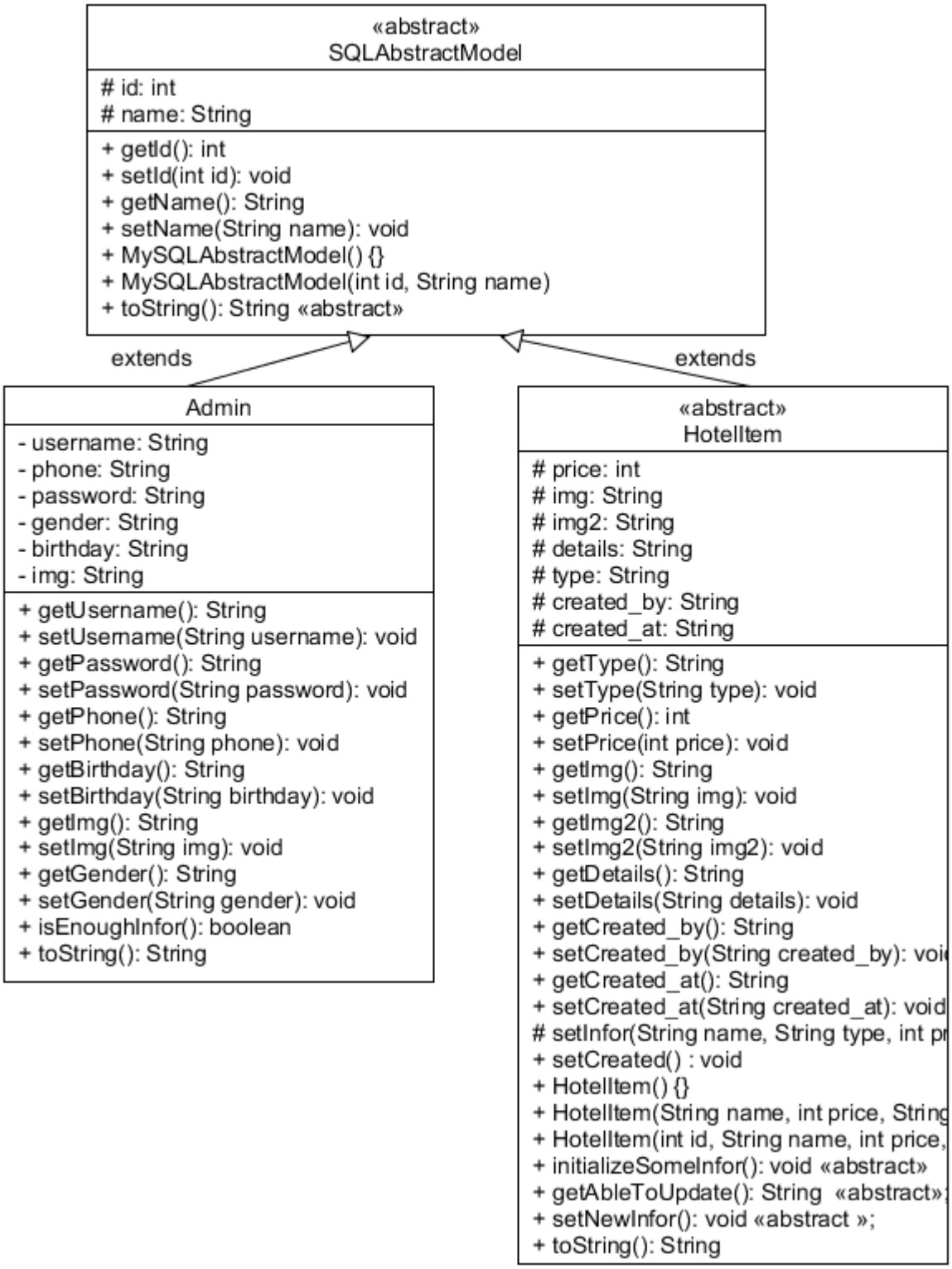


Figure 97: Spring SQL Model class diagram

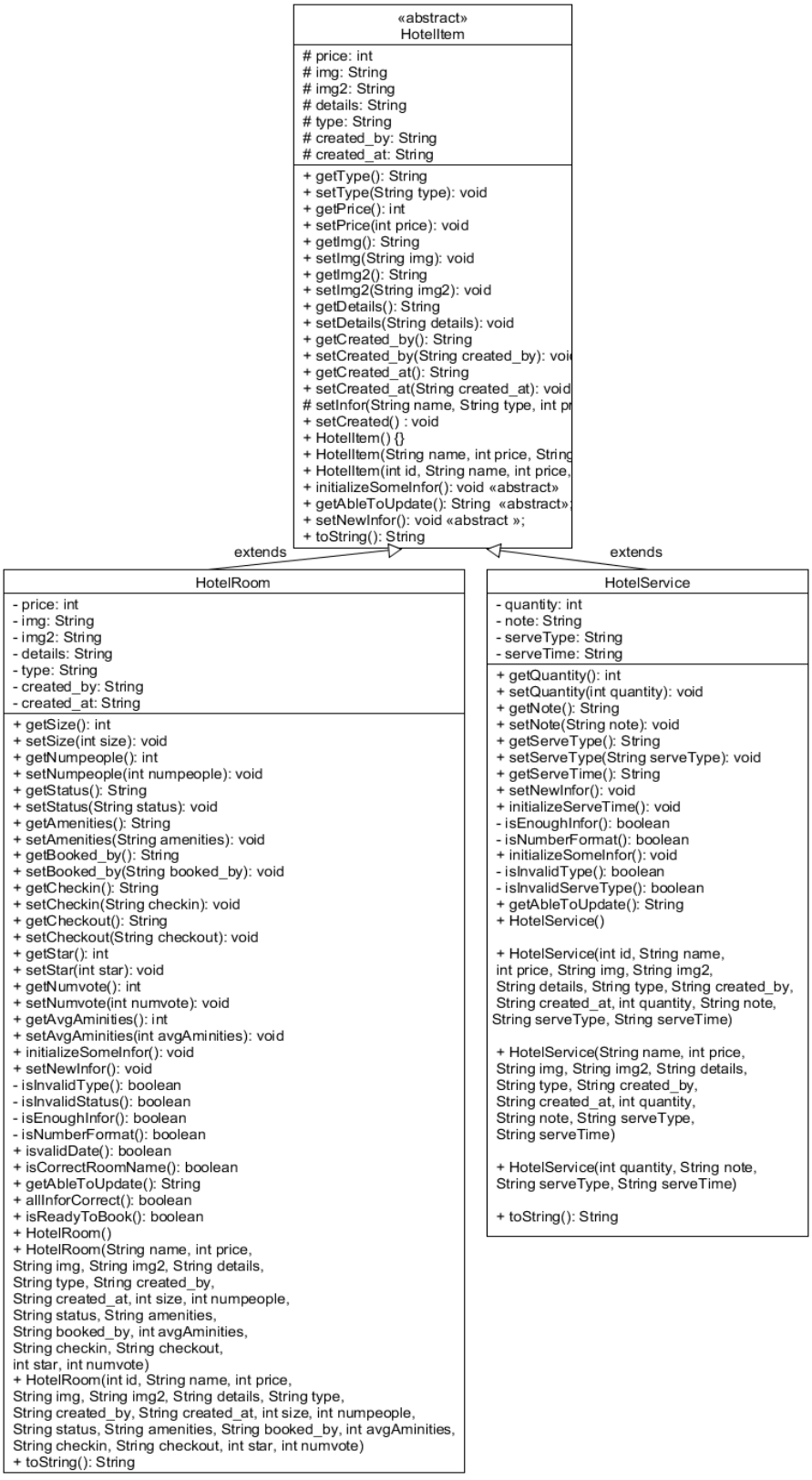


Figure 98: Spring Hotel Item Model class diagram

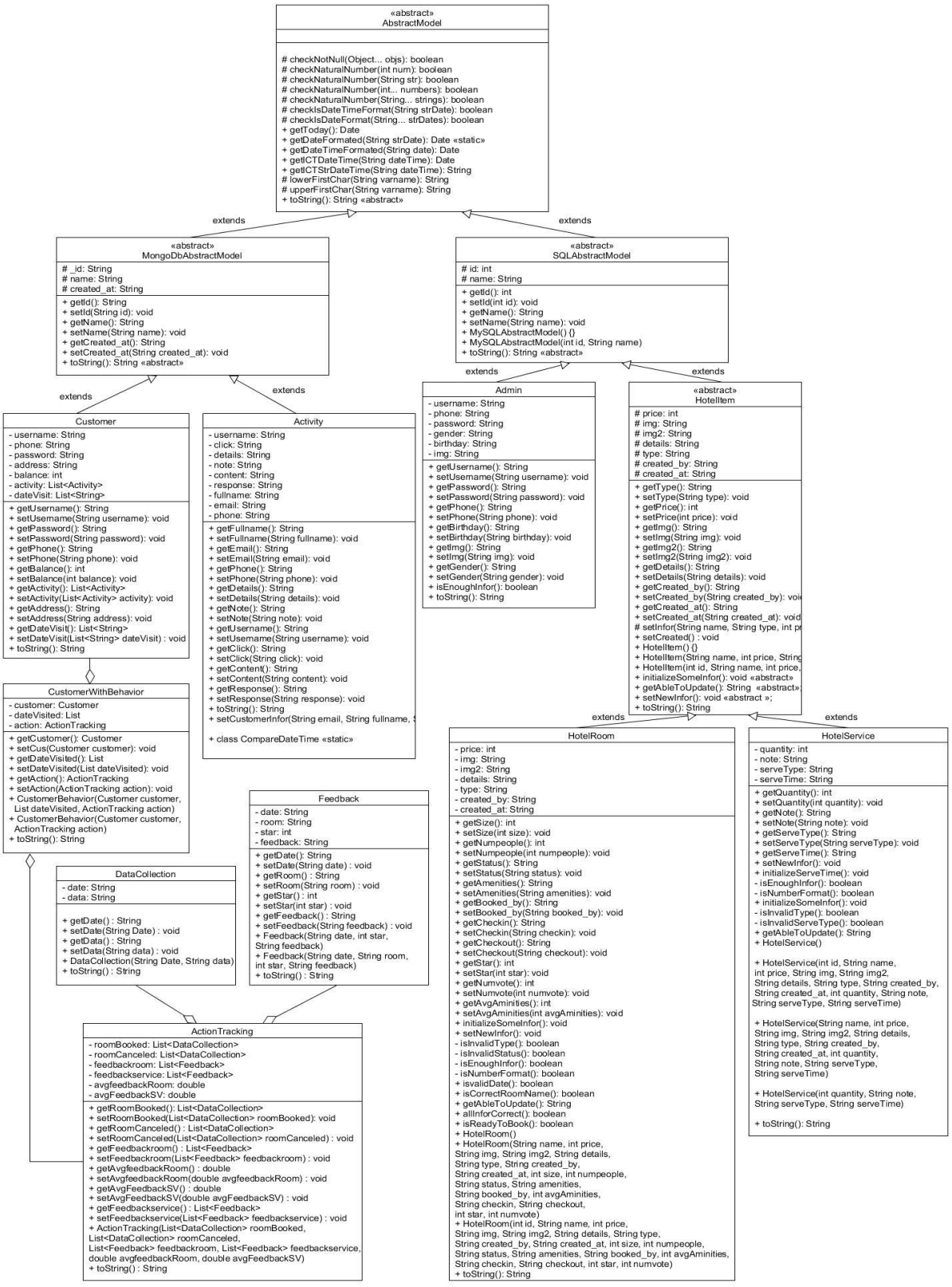


Figure 99: The class diagram for the whole Spring relationship Model

Appendix B. Spring AppController class diagram

In this appendix section, I will show all the variables, methods as well as the mapping URL in the class AppController in **section 4.4.4.2**.

Figure 100: Spring App Controller class diagram

Because the AppController has to handle too much business logic (**figure 100**), it is very difficult to see its variables and methods in one side A4 paper. If you are looking for more details, please refer the code panels below:

AppController

<< mapping URL: "/" >>

- UserService userService;

- HotelItemService hotelItemService;

- ApplicationService appService;

+ login(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ checklogin(LoginBean loginbean, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ logout(HttpServletRequest request)

+ forgetPassword(String email, HttpServletResponse response)

+ index(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ searchResult(String keyword, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ profile(HttpServletRequest request, HttpServletResponse response, ModelMap model) <<**"/profile"**, method = GET >>

+ changePassword(ChangePasswordBean changePassBean, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ editProfile(Administrator ad, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ profileImgEdited(CommonsMultipartFile img, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ manageRooms(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ singleRoom(String roomName, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ editRoom(String roomName, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ addRoom(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ roomAdded(HotelRoom newRoom, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ roomEdited(HotelRoom roomEdit, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ removeRoom(int id, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ roomImgEdited(CommonsMultipartFile img1, CommonsMultipartFile img2, String roomName, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ manageRestaurant(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ singleService(String servicename, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ addService(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ serviceAdded(HotelService newService, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ editService(String servicename, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ serviceEdited(HotelService serviceEdit, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ removeService(int id, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ serviceImgEdited(CommonsMultipartFile img1, CommonsMultipartFile img2, String servicename, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ manageUsers(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ followAllUsers(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ followUsers(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ followUsersPage(int page, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ followUsersPageSorted(String fieldname, String sort, int page, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ searchFollowUsers(String fieldname, String keyword, String sort, int page, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ trackingIP(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ trackingExternalIP(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ trackingMemebers(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ pageAccessStatistics(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ followUserChart(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ pageAccessChart(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ pageAccessMemberChart(String username, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ pageAccessIPChart(String ipaddress, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ followUsersIP(String ip, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ followMember(String username, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ ipDetails(String externalip, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ singleUser(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ singleCustomer(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ singleUser(String username, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ singleCustomer(String username, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ message(HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ notification(String id, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ replyBooking(String id, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ replyCancel(String id, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ sendMail(String id, String message, String useremail, String subject, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ fqa(HttpServletRequest request, HttpServletResponse response, ModelMap model, String redirect)

+ uploadFQA(CommonsMultipartFile fqaPDF, HttpServletRequest request, HttpServletResponse response, ModelMap model)

+ downloadCSV(HttpServletRequest request, HttpServletResponse response)

- isAuthenticated(HttpServletRequest request): boolean

- checkAuth(HttpServletRequest request, HttpServletResponse response): void

- authInitializeRedirect(HttpServletRequest request, HttpServletResponse response, ModelMap model, String redirect): String

- initialize(ModelMap model): void

- initializeProfile(ModelMap model): String

- initializeSingleRoom(ModelMap model, String roomName, String redirect): String

- initializeSingleService(ModelMap model, String servicename, String redirect): String

- initializeTracking(String tracking, HttpServletRequest request, HttpServletResponse response, ModelMap model): String

Appendix C. Express Controller class diagram

In this appendix section, I will clarify the single class Controller in **section 4.4.8** with their variables and functions. They are shown in the code panels below:

Controller

- activityModel: activity-model

- userModel: user-model

- followUserModel: follow-users-model

- ipSuggestModel: ip-suggest-model

- externalip: externalip <<const>>

- cookie: cookie;

- appConst: app-const

- httpRequest: request;

- nodemailer: nodemailer;

- transporter: Transport

- sendHTMLEmail(from, to, subject, content)
- getApi(response, err, resource)
- postApi(response, err, resource)
- deleteApi(response, err, resource)
+ getActivityByUserName(request, response)
+ getNotResponseActivity(request, response)
- getActivityByID(request, response)
+ getActivityByID(request, response)
+ seenAndGetNotification(request, response)
+ replyAndGetNotification(request, response)
+ getActivityFeedBackRoom(request, response)
+ getActivity(request, response)
+ getFollowUserByUserIP(request, response)
+ getFollowUserByPage(request, response)
+ getNumPageTracking(request, response)
+ getSortedTrackingData(request, response)
+ getSortedTrackingData2(request, response)
+ searchTrackingData(request, response)
+ searchTotalPage(request, response)
+ getExternalIP(request, response)
+ getExternalIPStatistics(request, response)
+ getIPStatistics(request, response)
+ getUsernameStatistics(request, response)
+ getPageAccessStatistics(request, response)
+ getPageAccessByIP(request, response)
+ getPageAccessByUsername(request, response)
+ getCountryChartData(request, response)
+ getFollowUserByID(request, response)
+ getFollowUser(request, response)
+ getUser(request, response)
+ getUserByID(request, response)
+ GetUserByUsername(request, response)
+ postActivity(request, response)

- getMailContent(subject, time)
+ postFollowUser(request, response)
+ putUser(req, response, next)
+ deleteActivity(request, response)
+ deleteFollowUser(request, response)
+ serializeUser(username, done)
+ deserializeUser(id, done)
+ checklogin(username, password, done)
+ logout(req, res)
+ login(req, res, next)
+ loginsuccess(req, res, next)
+ changepass(req, res, next)
- renderChangePWError(err, res)
+ checkPassword(req, res, next)
+ register(req, res, next)
+ checkregister(req, res, next)
- getRoomNameCustomerClicked(follow_users)
- updateNewIpSuggest(ipSuggestModel, ipSuggest, userip)
+ getRoomSuggestion(request, response)
- updateRecommemdationRoom(follow_users, ip_address)
- saveFollowUserByIP(follow_users, ip_address, external_ip, response)
- saveFollowUserData(request, response, external_ip)
- followUserBehavior(page_access, duration, username)
- followUsers(new_page_access, req, res)
- getIpAddress()
- checkAuthentication(req, res, next)
- getSuggestionRoom(rooms, price, size, avgAminities)
- get4NumNearest(rooms, att, value)
- getIndicesOfMin(inp, n)
- checkNotNull(...items)
- checkIsNaturalNumber(...items)
- checkIsPositiveFloat(...items)
- isValidEmail(email)
- isValidIPaddress(ipaddress)

```
- isValidUsername(username)
- isValidUser(user)
- isAceptableUser(user)
- activityIsAbleToUpdate(activity)
- followUserIsAbleToUpdate(followUser)
- ipSuggestIsAbleToUpdate(ipSuggest)
```

Appendix D. *Angular2 Components class diagram*

This final appendix part has three **figures includes 101, 102 and 103** which will clearly describe the variables and functions of each component in **section 4.4.12** and **figure 54**

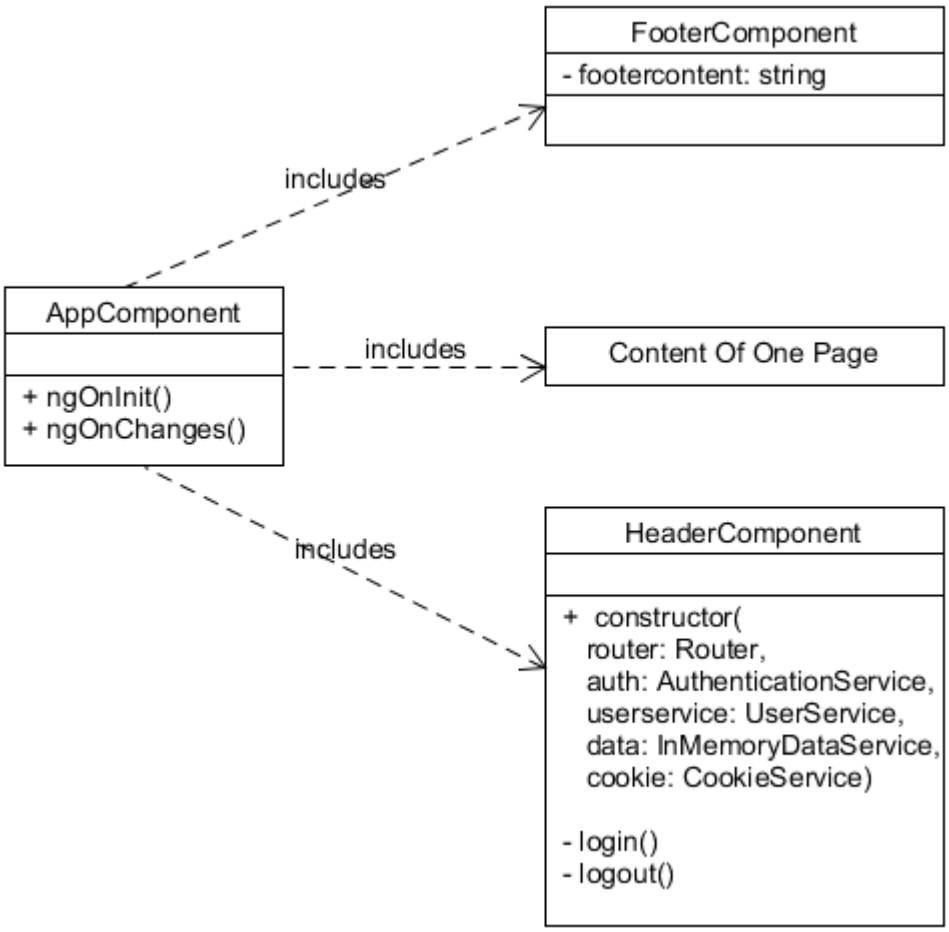


Figure 101: Angular2 AppComponent

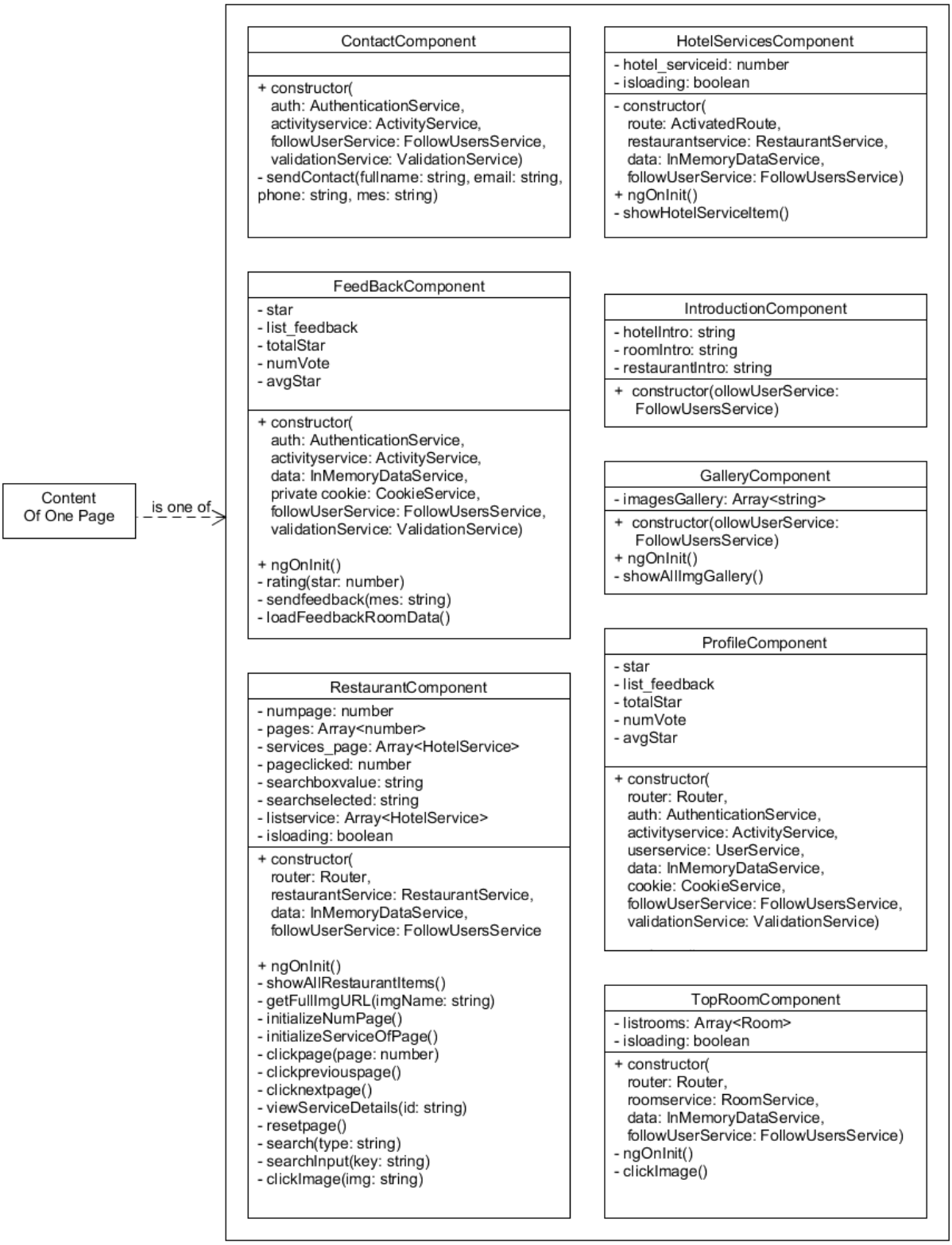

Content
Of One Page
is one of
ContactComponent
+ constructor(
auth: AuthenticationService,
activityservice: ActivityService,
followUserService: FollowUsersService,
validationService: ValidationService)
- sendContact(fullname: string, email: string, phone: string, mes: string)
FeedBackComponent
- star
- list_feedback
- totalStar
- numVote
- avgStar
+ constructor(
auth: AuthenticationService,
activityservice: ActivityService,
data: InMemoryDataService,
private cookie: CookieService,
followUserService: FollowUsersService,
validationService: ValidationService)
+ ngOnInit()
- rating(star: number)
- sendfeedback(mes: string)
- loadFeedbackRoomData()
RestaurantComponent
- numpage: number
- pages: Array<number>
- services_page: Array<HotelService>
- pageclicked: number
- searchboxvalue: string
- searchselected: string
- listservice: Array<HotelService>
- isloading: boolean
+ constructor(
router: Router,
restaurantService: RestaurantService,
data: InMemoryDataService,
followUserService: FollowUsersService
+ ngOnInit()
- showAllRestaurantItems()
- getFullImgURL(imgName: string)
- initializeNumPage()
- initializeServiceOfPage()
- clickpage(page: number)
- clickpreviouspage()
- clicknextpage()
- viewServiceDetails(id: string)
- resetpage()
- search(type: string)
- searchInput(key: string)
- clickImage(img: string)
HotelServicesComponent
- hotel_serviceid: number
- isloading: boolean
- constructor(
route: ActivatedRoute,
restaurantservice: RestaurantService,
data: InMemoryDataService,
followUserService: FollowUsersService)
+ ngOnInit()
- showHotelServiceItem()
IntroductionComponent
- hotelIntro: string
- roomIntro: string
- restaurantIntro: string
+ constructor(ollowUserService: FollowUsersService)
GalleryComponent
- imagesGallery: Array<string>
+ constructor(ollowUserService: FollowUsersService)
+ ngOnInit()
- showAllImgGallery()
ProfileComponent
- star
- list_feedback
- totalStar
- numVote
- avgStar
+ constructor(
router: Router,
auth: AuthenticationService,
activityservice: ActivityService,
userservice: UserService,
data: InMemoryDataService,
cookie: CookieService,
followUserService: FollowUsersService,
validationService: ValidationService)
TopRoomComponent
- listrooms: Array<Room>
- isloading: boolean
+ constructor(
router: Router,
roomservice: RoomService,
data: InMemoryDataService,
followUserService: FollowUsersService)
- ngOnInit()
- clickImage()

Figure 102: Angular2 Content Component 1

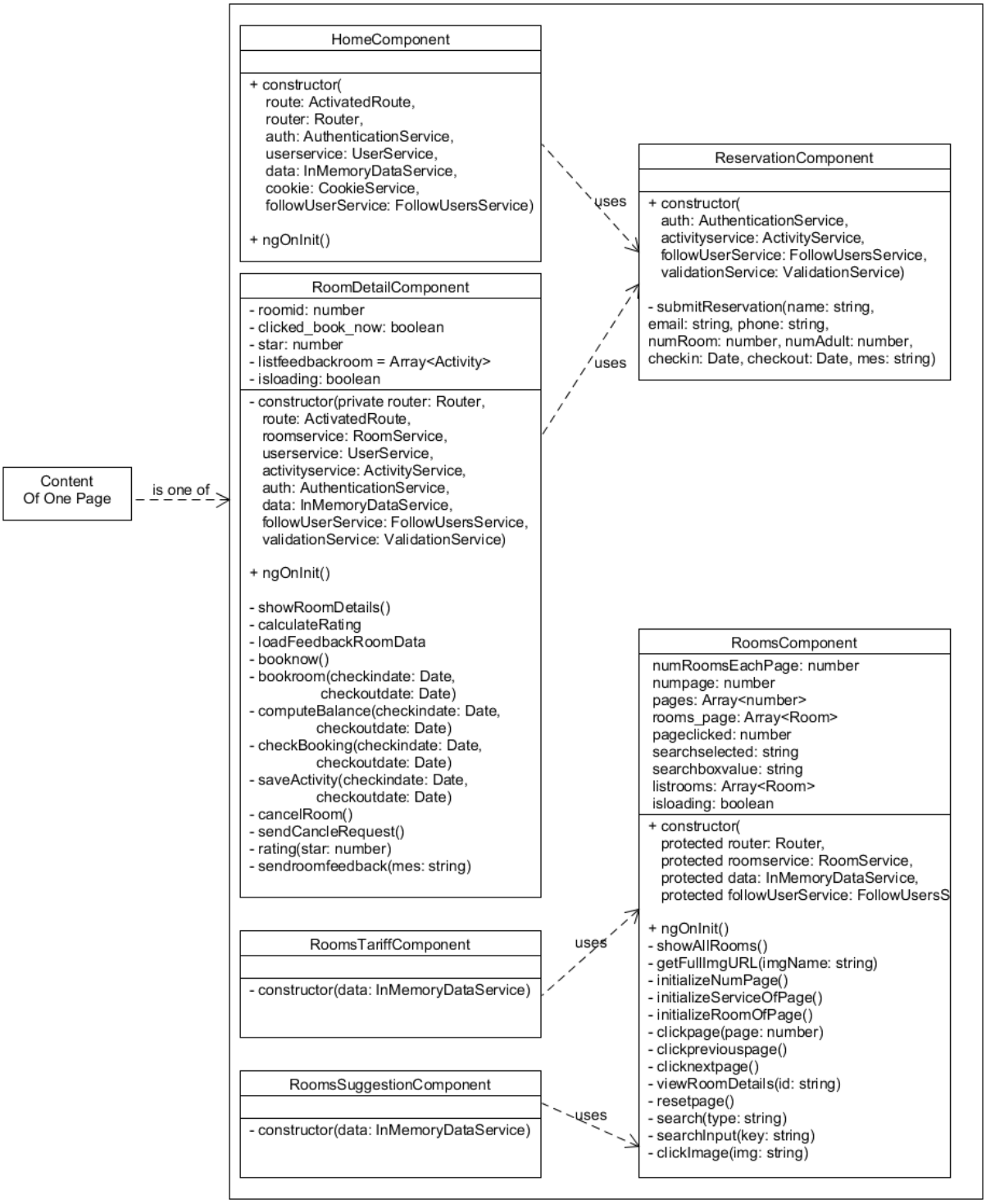


Figure 103: Angular2 Content Component 2