Đỗ Hùng Cường

ITITIU13170

Thesis report

Abstraction: summary of all thesis – viết sau

List of table

List of figure

Definition

|  |  |
| --- | --- |
| Term | Descriptions |
| Guest |  |
| Customer |  |
| Administrator |  |
| HTML |  |
| CSS |  |
| Boostrap |  |
| Jquery |  |
| IDE |  |
| UML |  |
| Linux |  |

Table of content

Introduction

1/ Situation

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, if the huge number of guests came 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.



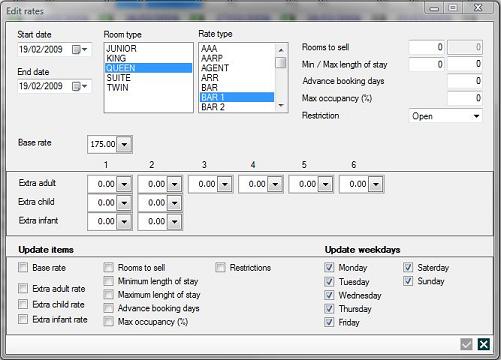
Moreover, in many enormous hotels, management was very difficult and booking based on pen and paper was not convenient for both the customers and the receptionists. Besides, the reservations might cause a lot of risk such as invalid information of customers, wrong information of rooms and booking.

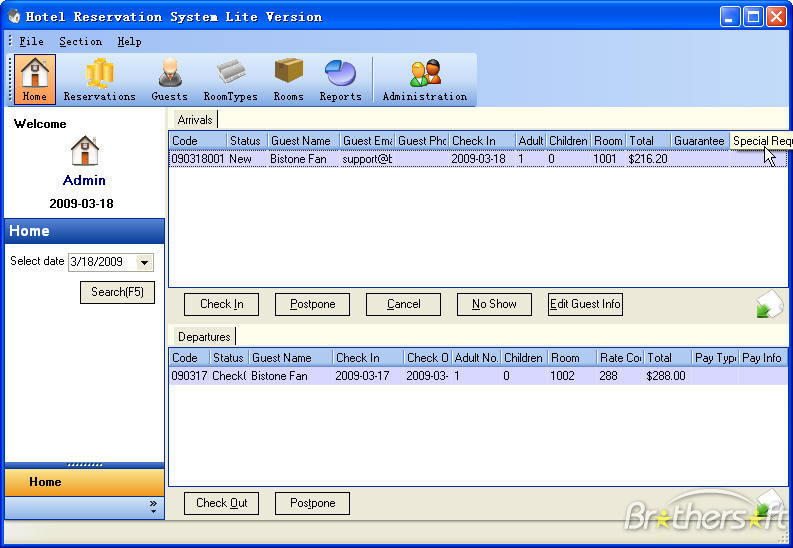
Therefore, many applications had been released with lots of features to support hotel booking. However, at that time, performance of these systems might be very bad. They could overload or run extremely slow when a huge number of users was accessing at the same time. Moreover, the look and feed of these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

Ex: <https://www.planet-source-code.com/vb/scripts/ShowCode.asp?lngWId=10&txtCodeId=7186>

<http://www.bistonesoft.com/litereservationstatus.htm>

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



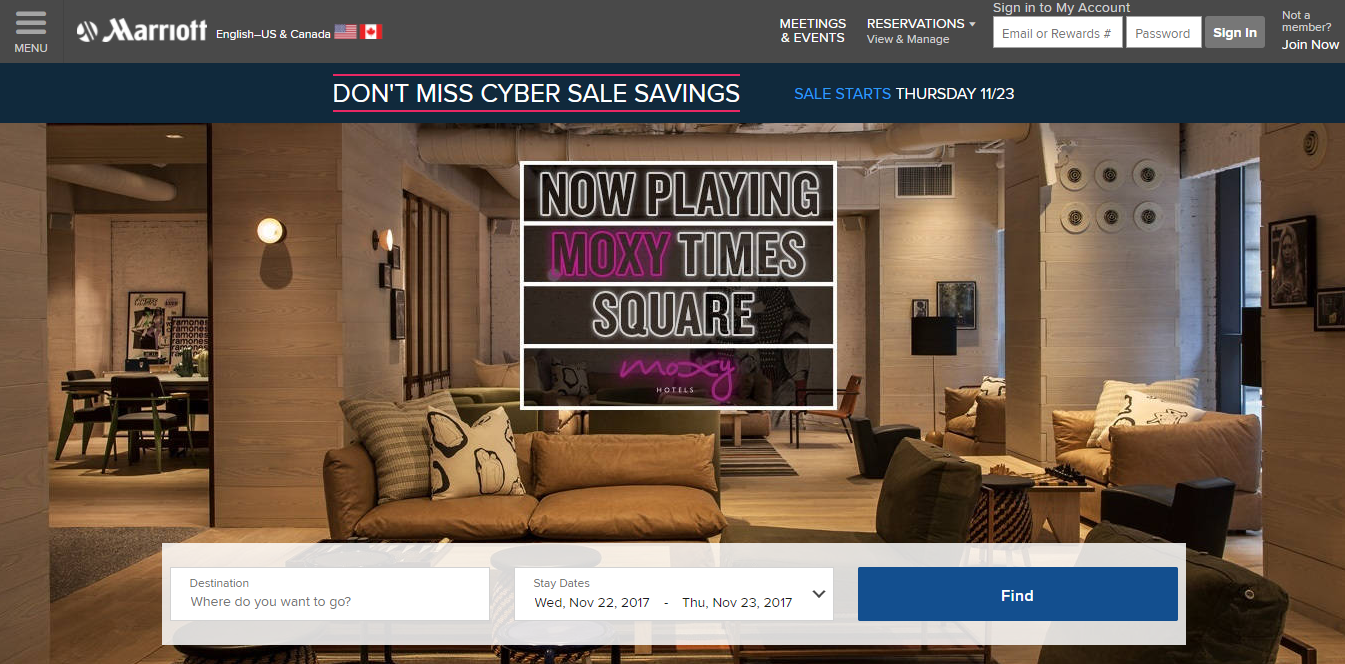


2/ Modern Hotel Booking System

To solve these problems above, many deluxe hotels or five-stars hotels in the world such as Marriott International, Hilton Worldwide or InterContinental Hotels Group already have their own hotel booking systems.

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.

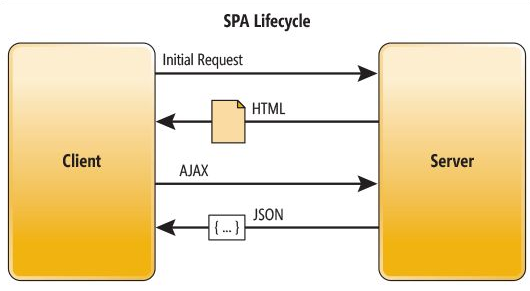
Based on the data collection, the systems will automatically suggest what customers may like, recommend which rooms customers should book. Moreover, the hotel owners can improve their hotel business based on the information collected by their systems.



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.

3.1/ Single Page Application



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 handle without reaching server.

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

AJAX is a developer's dream, because you can:

* Update a web page without reloading the page
* Request data from a server - after the page has loaded
* Receive data from a server - after the page has loaded
* Send data to a server - in the background

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.

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.

3.2.1/ AngularJS:



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. 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.

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

3.2.2/ Angular 2:

* Angular 2 is faster than Angular 1. Angular 2 is built for speed so that the initial loads is faster, the change detection and improved rendering times are also faster than Angular 1.
* The fewer concepts of Angular 2 make it easier to understand than Angular 1. The code structure is very simplified 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. Typescript can be compiled to 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.



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.

3.3/ RESTful webservice:



There are many ways to connect angular 2 to your backend server. RESTful Web Service which is essentially REST Architecture based Web Services is one of the architectural style that helps Angular and your backend server communicate with each other.

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.

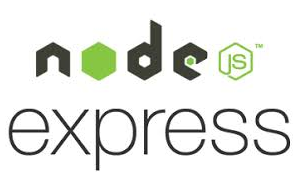
In some case, RESTful Web Service help us write a software application in various programming languages and we can run them on various platforms. For example, we can write a backend server in Java using RESTful web service and connect to fontend in Angular 2 using Typescript.

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.

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.

There are many framework written in many programming language support build a RESTful webservice such as Express framework for Nodejs and Spring MVC for java.

3.4/ Nodejs & Express framework



Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

Node.js package ecosystem, npm, is the largest ecosystem of open source libraries in the world. Node.js application is written in Javascript so it is very easy to interact with Angular to build a Single page application easily

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. The advantage of Nodejs combine with Express framework is listed below:

* APIs

With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

* Performance

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

3.5/ Java & Spring MVC: 

Spring MVC is the most powerful J2EE 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.

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 users, rooms, admin 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 Nodejs Express framework and Angular to build the best Single page application. It called MEAN stack technology which means MongoDB, Express, Angular and Nodejs.

3.6/ SQL & Hibernate:



SQL is a standard language for storing, manipulating and retrieving data in databases.

Some kind of SQL: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.

Hibernate ORM (Hibernate in short) 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. Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Hibernate is free software that is distributed under the GNU Lesser General Public License 2.1.

Hibernate's primary feature 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.

3.7/ MongoDB:

MongoDB is an open-source database. In MongoDB, entity relationship is not mandatory so we can call it NoSQL – No SQL which provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

MongoDB stores data in JSON-like documents that can vary in structure. Related information is stored together for fast query access through the MongoDB query language.

MongoDB uses dynamic schemas, meaning that you can create records without first defining the structure, such as the fields or the types of their values. You can change the structure of records (which we call documents) simply by adding new fields or deleting existing ones.

This data model give you the ability to represent hierarchical relationships, to store arrays, and other more complex structures easily.

Documents in a collection need not have an identical set of fields and denormalization of data is common. MongoDB was also designed with high availability and scalability in mind, and includes out-of-the-box replication and auto-sharding.

No entity relationship => don’t need to define the structure

- stores data in JSON-like documents => easily work with javascript, Angular, nodejs

- redundant data but high performance (Related information is stored together for fast query) => Optimize Query Performance

3.8/ MEAN stack:



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.

4/ Goals and Scope

To keep up with current trends in hotel business industry, I will build a Hotel Booking and Reservations System which 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 120 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 2 and RESTFULL web service which is built by Nodejs and express framework.

Furthermore, the administrator’s system is built by Spring MVC, the most powerful java framework 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 feature.

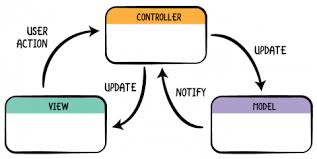
II/ Background

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.

Firstly, MVC is general architecture of the whole of my system

1/ MVC:

MVC stands for (Model – View -Controller)



Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View

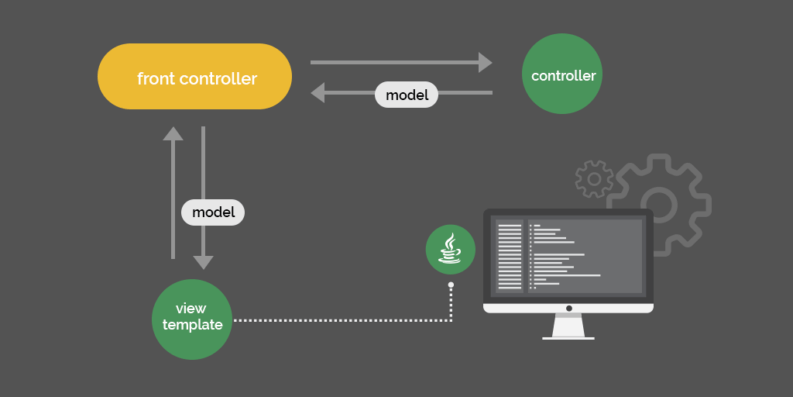
The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

A MVC framework on the other hand makes it easier to build a web application. It may provide data type and database interaction for controllers and a mechanism for the communication between view and controller.

2/ Spring MVC



Spring MVC is one of the most popular framework for java and also based on MVC framework.

Because it follows MVC pattern, the architecture must be separate into 3 main components: Model, View and Controller and each component work basically similar to MVC framework that I mentioned above. However, in Spring MVC architecture, 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.

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

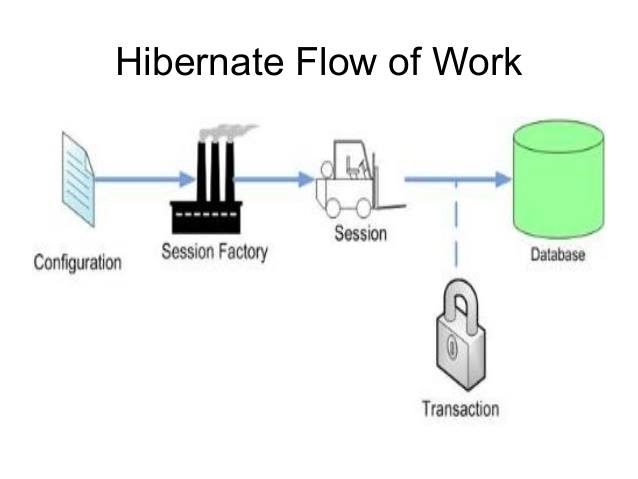
With RESTController, you can build a RESTful API with Spring framework, which can interact with client and other server side by HTTP methods.

3/ Hibernate

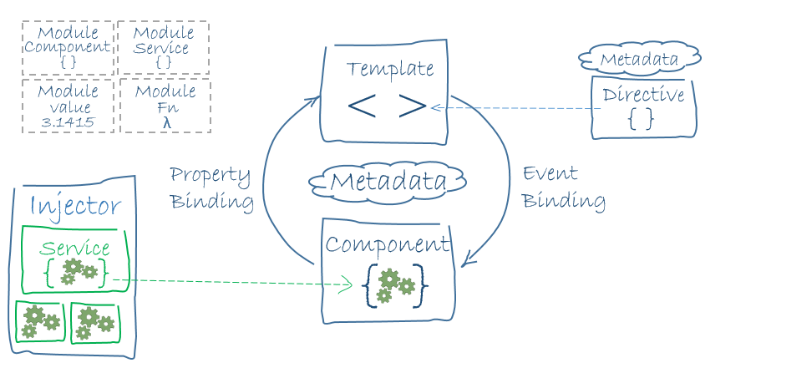
Hibernate has a layered architecture which helps the user to operate without having to know the underlying APIs. Hibernate makes use of the database and configuration data to provide persistence services (and persistent objects) to the application.

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 for 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 which help you update and retrieving data from database system.



3/ Angular



Angular is one of the most popular Javascript frontend framework. 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. Their definitions and the way they communicate is describe in brief below:

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

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 interacts 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.

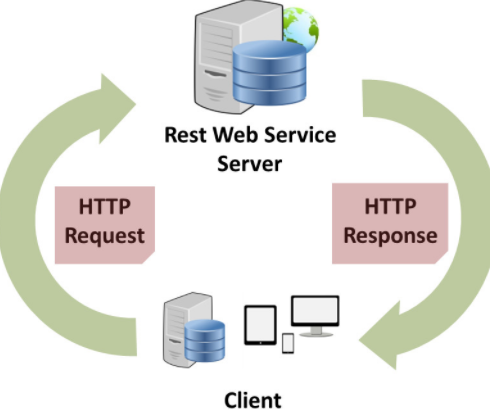
Angular supports data binding for coordinating parts of a component with parts of a template. Data binding is added to the template HTML that tell Angular how to connect both sides

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

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

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.

4/ RESTfull webservice



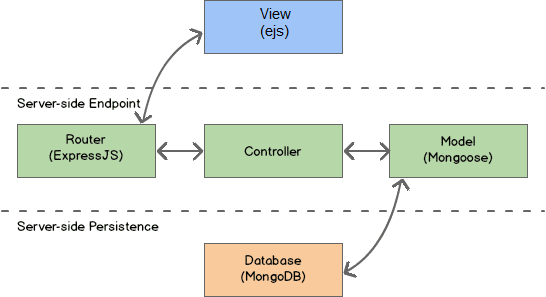
RESTful Web Services are Web services based on REST Architecture. They use HTTP methods to implement the concept of REST architecture. URI is usually a service which a RESTful Web Service provides resource such as Text, JSON and XML.

There are 5 HTTP methods are most commonly used in a REST based architecture.

| **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. |

RESTFul Web service provides API as HTTP methods: GET, POST, PUT, PATCH, DELETE which allows other party to calls these methods to retrieve data (as json, xml or format), create, update, modify or delete the data store in database system.

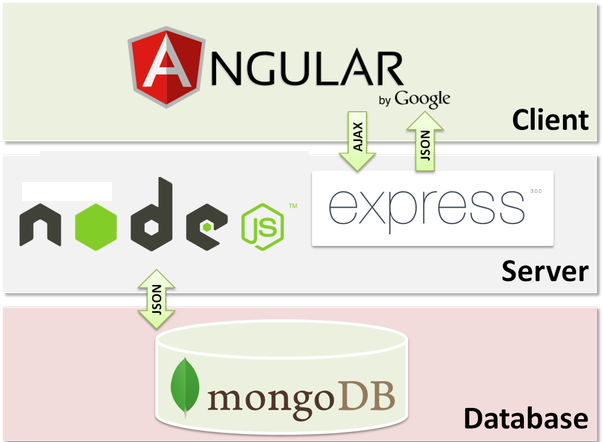
5/ Express framework + Nodejs



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

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 Nodejs can retrieve data or update to mongodb or other 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 webservice

6/ MEAN stack

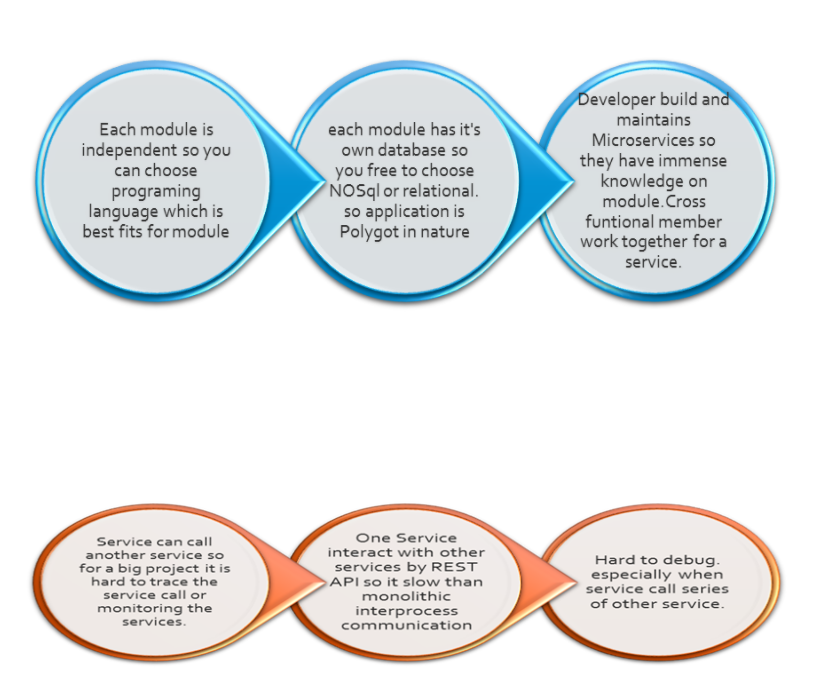


The term MEAN stack refers to a collection of JavaScript based technologies used to develop web applications. MEAN is an acronym for MongoDB, ExpressJS, Angular and Node.js. From client to server to database, MEAN is full stack JavaScript. Nodejs and Express framework are on server side, Angular is on client side and MongoDB is a database system that stores data in JSON (JavaScript Object Notation ). Angular on client side gets data from Nodejs & express framework on server side in JSON format which may retrieve from mongodb to represent the view. And in contrast, Nodejs & express framework receive AJAX request from Angular to execute business logic and update to mongodb.

III/ Software Requirement

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 & Nodejs 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 Nodejs 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 which 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 42 primary feature and hundreds of small features.

2/ Feature

The users of my system were separated into 3 primary roles: Guest, Customer and Administrator. There are many features that my system support for each role

a/ 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.

b/ Customer

The customers can do anything which the guests can do. 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.

c/ 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, Admin |  |
| 2 | Login | Guest, Admin |  |
| 3 | Logout | Customer, Admin |  |
| 4 | View Rooms | Guest, Customer, Admin |  |
| 5 | View Restaurant | Guest, Customer, Admin |  |
| 6 | Search for Room | Guest, Customer, Admin |  |
| 7 | Search for Food, Drink | Guest, Customer, Admin |  |
| 8 | View gallery of hotel | Guest, Customer |  |
| 9 | View introduction of hotel | Guest, Customer |  |
| 10 | Filer rooms | Guest, Customer, Admin |  |
| 11 | Filer food or drink | Guest, Customer, Admin |  |
| 12 | Send contact | Guest, Customer |  |
| 13 | Send reservation form | Guest, Customer |  |
| 14 | Book room | Customer |  |
| 15 | Cancel room | Customer |  |
| 16 | View profile | Customer, Admin |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Feature** | **User** | **Description** |
| 17 | Edit profile | Customer, Admin |  |
| 18 | Change password | Customer, Admin |  |
| 19 | View activity | Customer |  |
| 20 | Send feedback | Customer |  |
| 21 | Dashboard management | Admin |  |
| 22 | Receive notification | Admin |  |
| 23 | Send message | Admin |  |
| 24 | View users | Admin |  |
| 25 | Manage users | Admin |  |
| 26 | Ban users | Admin |  |
| 27 | Add new room | Admin |  |
| 28 | Delete room | Admin |  |
| 29 | Update room | Admin |  |
| 30 | Add food or drink | Admin |  |
| 31 | Remove food or drink | Admin |  |
| 32 | Update food or drink | Admin |  |
| 33 | Update profile image | Admin |  |
| 34 | Follow users | Admin |  |
| 35 | Send feedback & rate hotel | Customer |  |
| 36 | Send feedback & rate room | Customer |  |
| 37 | View customer activity | Admin |  |
| 38 | View statistic of visit times | Admin |  |
| 39 | View recommendation room | Guest, Customer |  |
| 40 | View related room | Admin |  |
| 41 | View top of rooms | Guest, Customer, Admin |  |
| 42 | Email template | Admin |  |

3/ Use case:

Based on the feature above, I have a use case diagram below with 3 actors using the system: guest, customer and administrator



4/ User story:

According to the Use Case diagram, it will drop down into 25 user story:

* 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 which 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

Kẻ thêm 1 số table use case quan trọng, ko quan trong đem xuống appendix

IV/ Methodology

1/ All Technology used:

To build the above system I need to use and apply a lot of technology.

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: NoSQL (MongoDB) and SQL (Microsoft SQL Server, MySQL)

Besides, I used UMLET and Edraw to design UML and draw diagram. For coding Nodejs, 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 I used maven with tomcat or glassfish for java server and node\_modules with npm for nodejs server

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

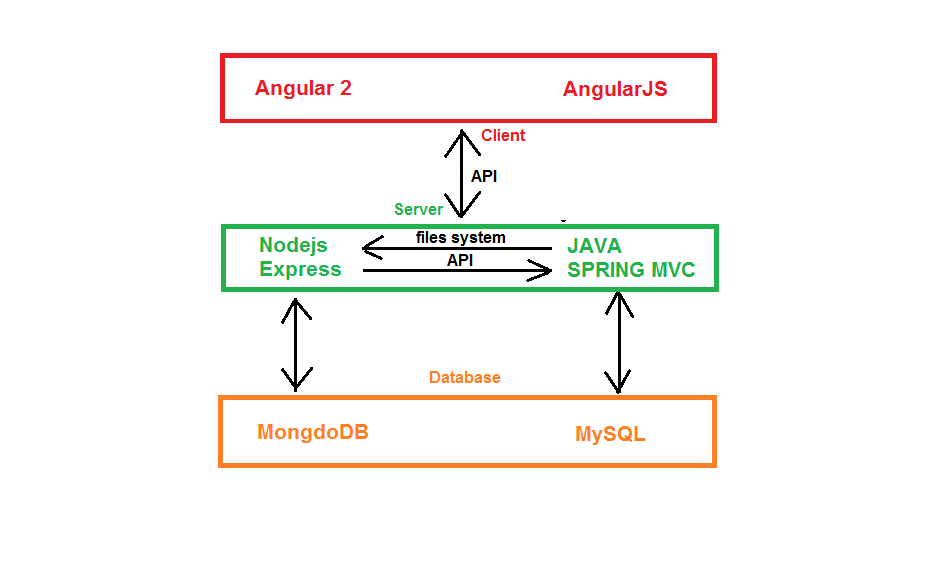
Some Library: gson, geoip

2/ System Architecture

My system includes 2 main architectures: one for Customer which applied MEAN stack technology and one for Administrator built in AngularJS with Spring MVC Hibernate.

They are built and deployed in 2 different servers and they communicate through API and file system

The picture below shows the very first background architecture of my system



The system architechture includes MVC architecture and RESTful Webservice with Angular Architecture

MVC architecture

Nodejs + Express MVC

Request of Customer from view EJS to Route which call the function from Controller to delegate Model 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 app-const provide constant variable for the whole system.

Spring + Hibernate MVC

Request of Administrator from JSP to FrontController which handle the view mapping before send to Controller. After that, the Controller will call method from Service which use the methods from DAO to connect database. The Model provide datatype for Controller, Service DAO and mapping with Hibernate 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.

Spring + API MVC

This architecture is almost the same as Spring + Hibernate MVC. However, the difference is that instead of retrieving or update data to database system, the DAO communicate with the RESTful Webservice to get the data or request for update data to REST API provider.

RESTful Webservice architecture

Nodejs + Express REST

Based on Nodejs + Express MVC architecture above, the Nodejs + Express RESTful webservice has the same structure but it is the webservice and it has no view. Therefore, the route became the API route which provide API as HTTP method for Angular on client side. (I use Angular for my system but you can use other Javascript library or framework)

Spring + Hibernate REST

Follow the Spring MVC + Hibernate structure and similar to the REST architecture above, this is a webservice provide API for Angular on client side by RESTController.

Spring + API REST

Another REST architecture which is the combination of Spring RESTful webservice and ability to interact with other API provider. In this case, the DAO communicate with Nodejs + Express RESTful webservice in section 3.5.5 to provide data as well as update to MongoDB database system

Angular on client side architecture:

In my system, Angular can communicate directly with the RESTful Webservices above through API

The diagram below show how Angular work in my system

AngularJS app

Request of Admin from JSP file which 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 Template and Controller which have function to interact with RESTful Webservice. Moreover, the AngularJS filters provide …

And AppConst provides const variable for the whole client side app

Angular 2 app

Full system architecture:

Based on the system architecture above, you can easily see that the Clients which include Angular communicate with the Servers through API. The first server built by Nodejs and Express framework connect to MongoDB and the second server build by Spring MVC connect to MySQL to store or provide data for client through API.

With 3 actors using the system showed in use case (section 2) and based on 2 main server for Guest, Customer and Admin. I separate the Database system into 2 primary part: MongoDB and SQL

MongoDB collection & document

with 4 collection: customer, activity, tracking and suggestion-data

Collection Customer contains Customer personal information, username & password

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

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

SQL ERD

MongoDB + SQL ERD

Database:

A Customer has id, name, username, address

A customer can book rooms, each room will contains booked\_by field which stored the username of the customer

A administrator

- Show all collections and describe => give image

- Some related field in collections:

- customer has activity store in activity collection

- customer books rooms store in rooms collection

- customer is follow by username store in follow-users collection

- user ip address of users in follow-users collection is the user ip address store in ip-suggest collection

- ip-suggest collection contains data of rooms which help determine the recommendation room

- rooms and restaurant items are created by admin which stored in admin collection

Based on these system architecture and ERD above, I created the class diagram

Because the system is too big and I cannot draw the class diagram in one page, I separated it into smaller class diagrams.

Mean stack technology class diagram:

Spring MVC hibernate diagram

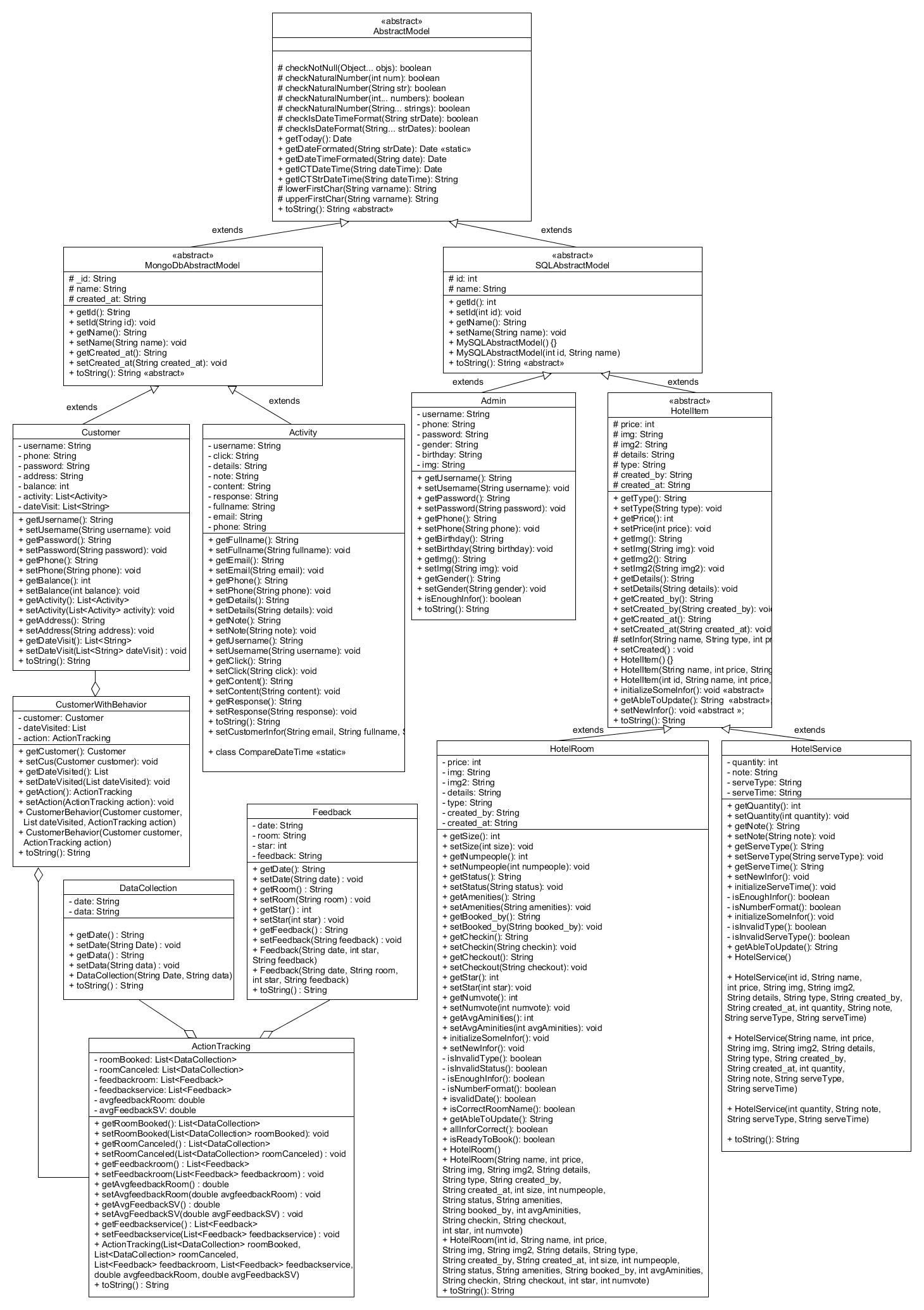
Model class diagram:

3 layers class diagram

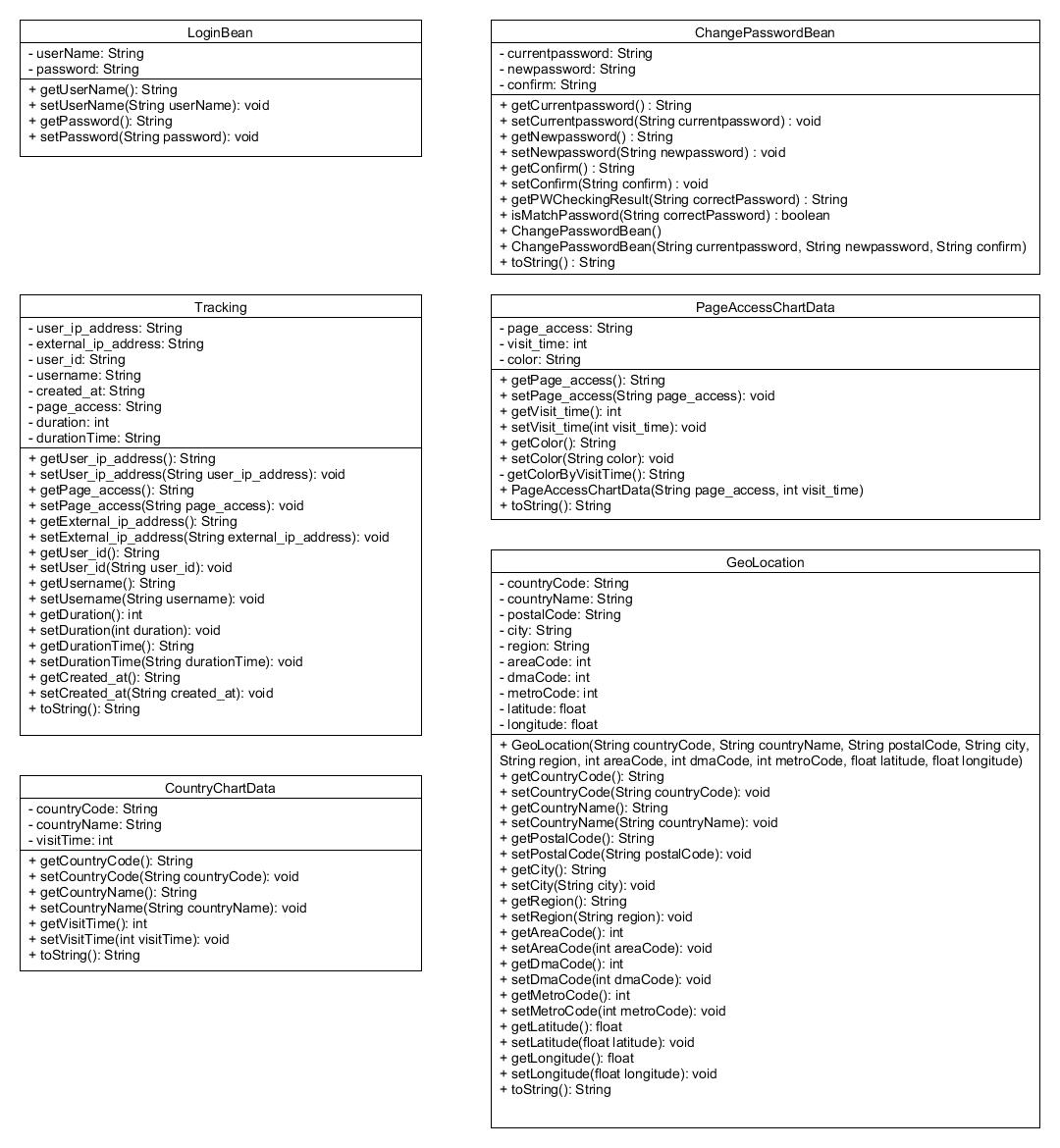
Spring MVC class diagram

According to the class diagram above, you can easily see that AdminDAO uses Admin model,

Class diagram – Relationship Model



Class diagram – PoJo, Bean- No Relationship Model



public abstract class AbstractModel {

This is the abstract class that provide some methods for sub class. The sub classes extend this class will inherit all method of this class that includes:

# checkNotNull(Object... objs): boolean : check many objects are null or not

# checkNaturalNumber(int num): boolean : check an integer is natural number or not

# checkNaturalNumber(String str): boolean : check a String is natural number or not

# checkNaturalNumber(int... numbers): boolean : check many integers are natural numbers or not

# checkNaturalNumber(String... strings): boolean : check many Strings are natural numbers or not

# checkIsDateTimeFormat(String strDate): boolean : check a String is correct date-time format or not

# checkIsDateFormat(String... strDates): boolean : check many Strings is correct date format or not

+ getToday(): Date : get Date of today

+ getDateFormated(String strDate): Date <<static>> : get Date by a String follow the exist format

+ getDateTimeFormated(String date): Date : get Date with Time by a String follow the exist format

+ getICTDateTime(String dateTime): Date : get Date with Time by a String follow ICT format

+ getICTStrDateTime(String dateTime): String: : get a String with ICT Date Time format by a String

# lowerFirstChar(String varname): String : lowercase first character of string

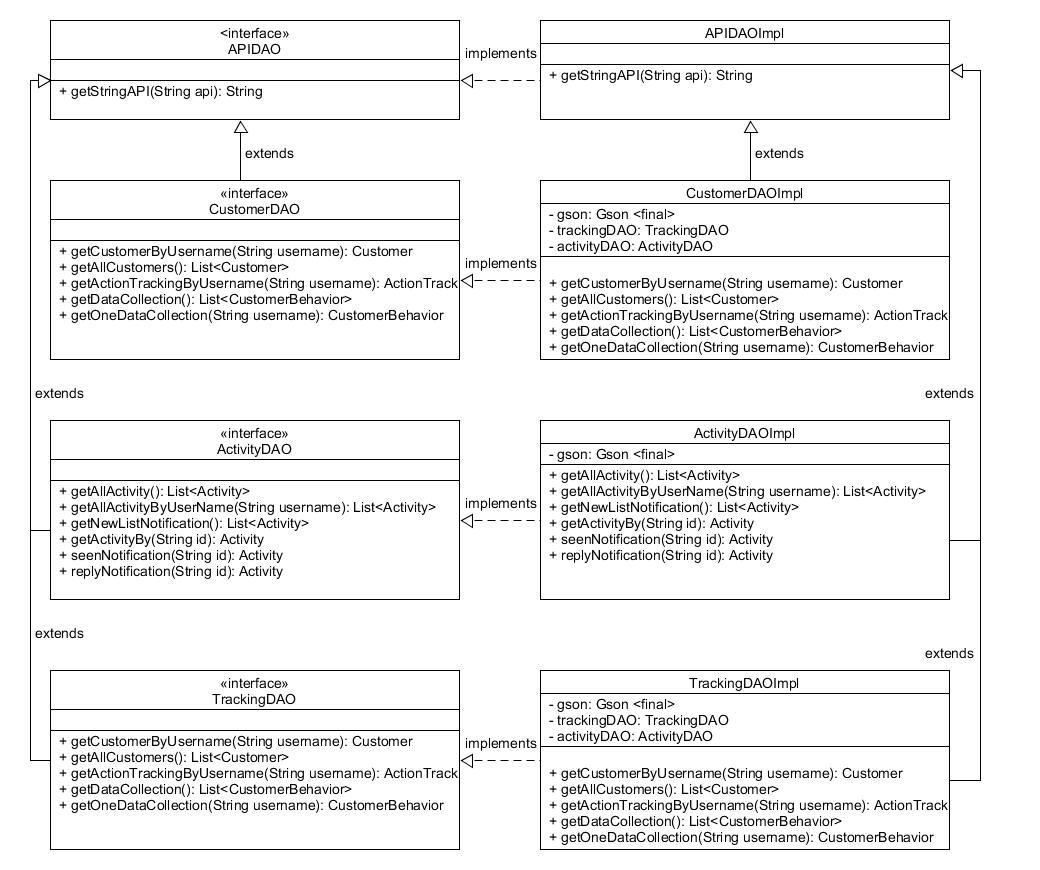
# upperFirstChar(String varname): String: uppercase first character of string

@Override

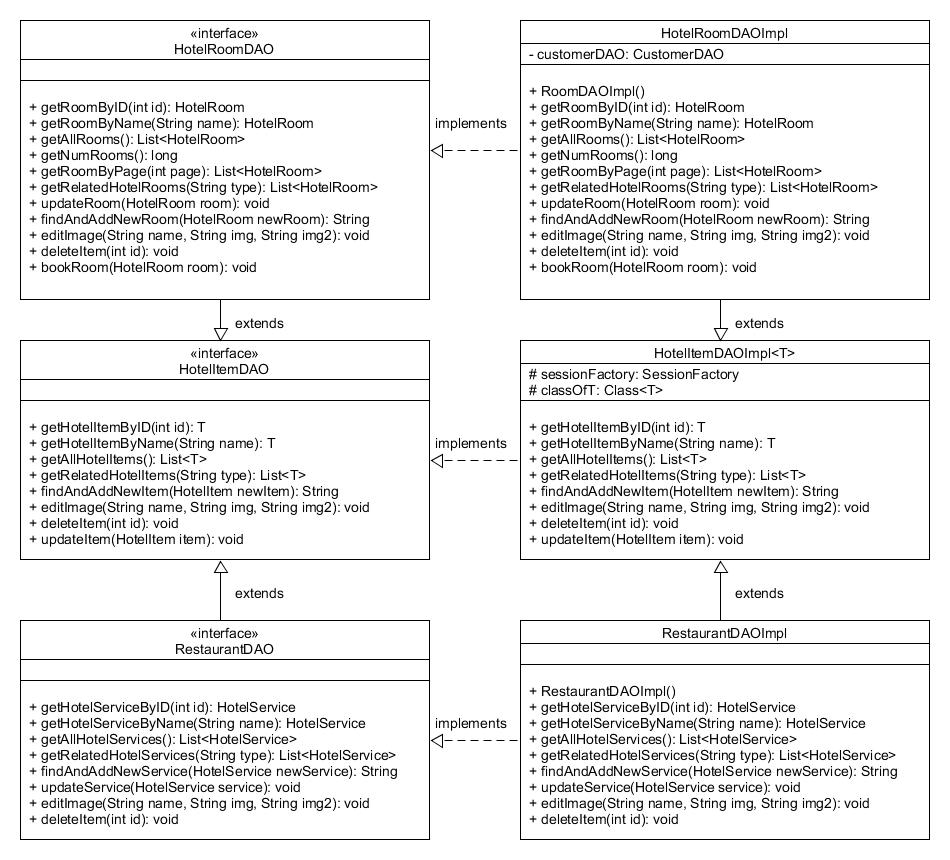
+ toString(): String <<abstract>>

}

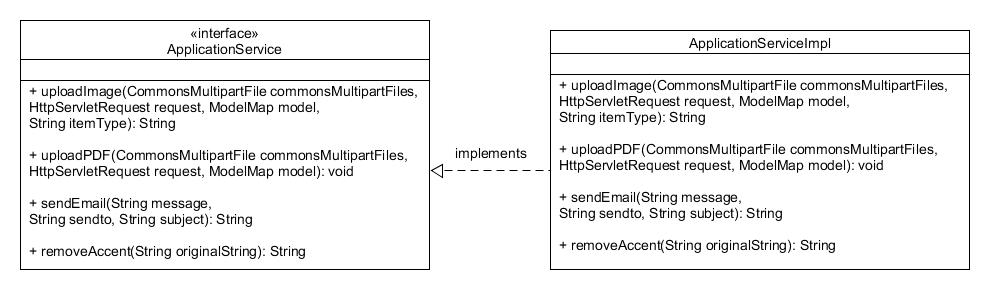
API DAO

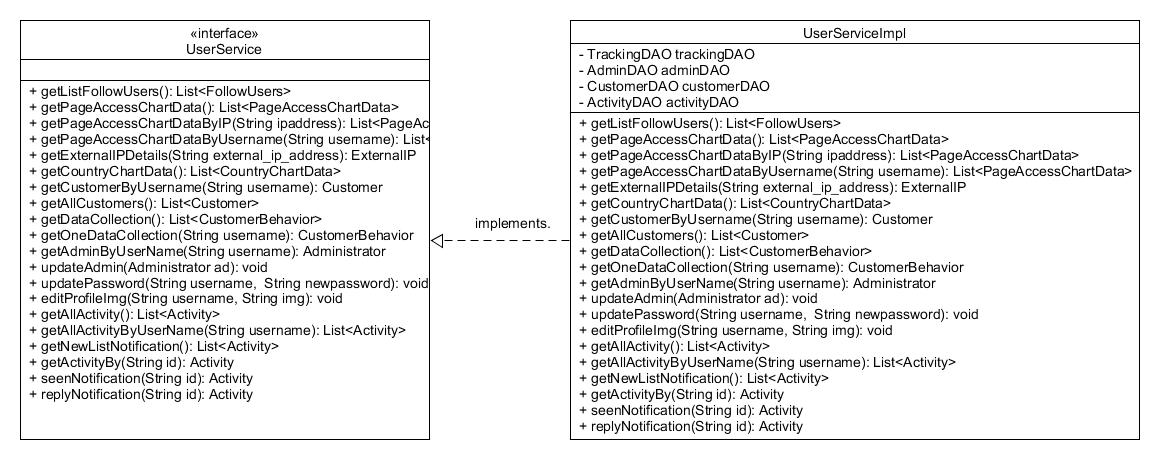


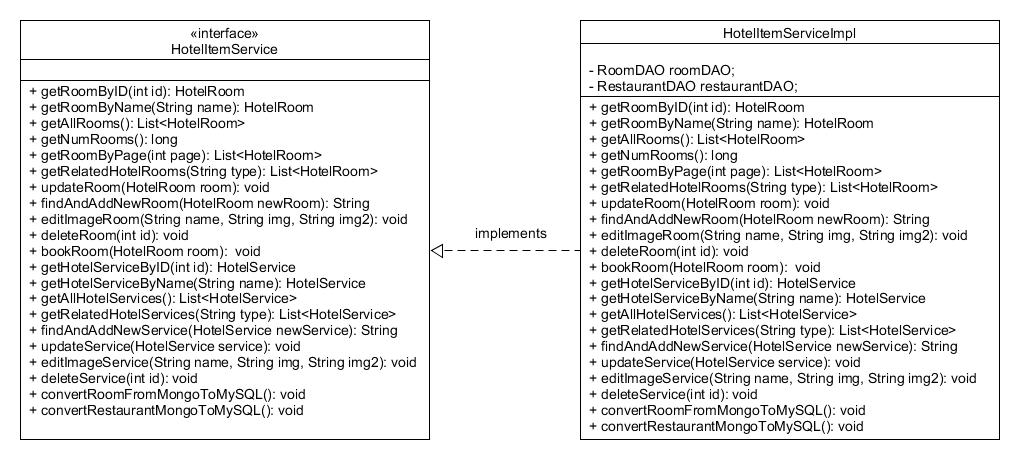
Hibernate DAO



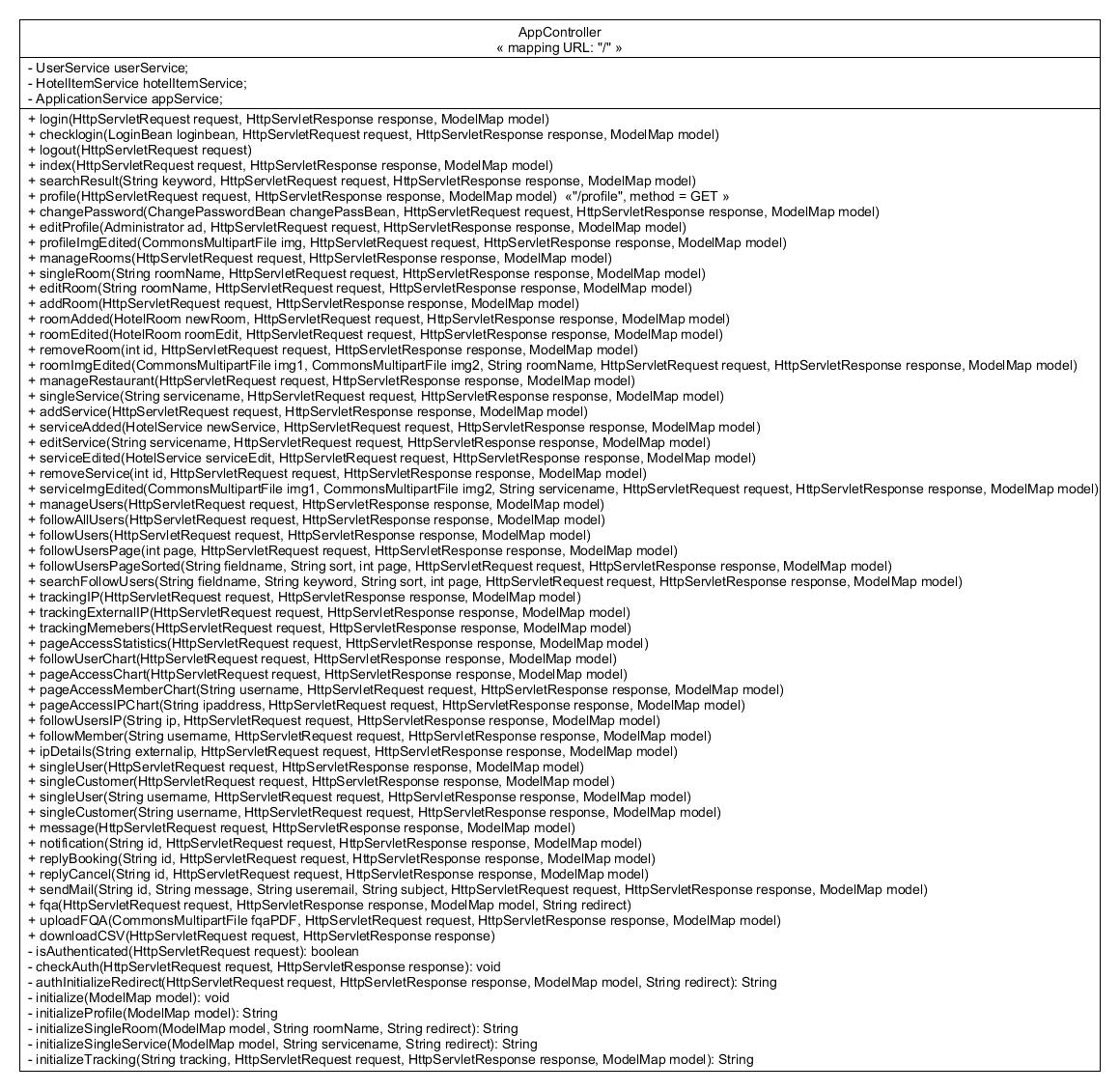
Service



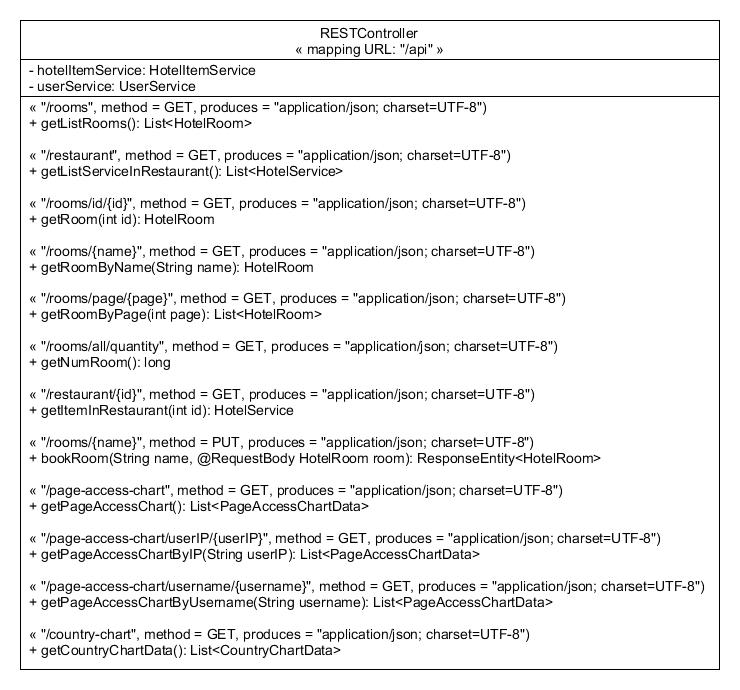




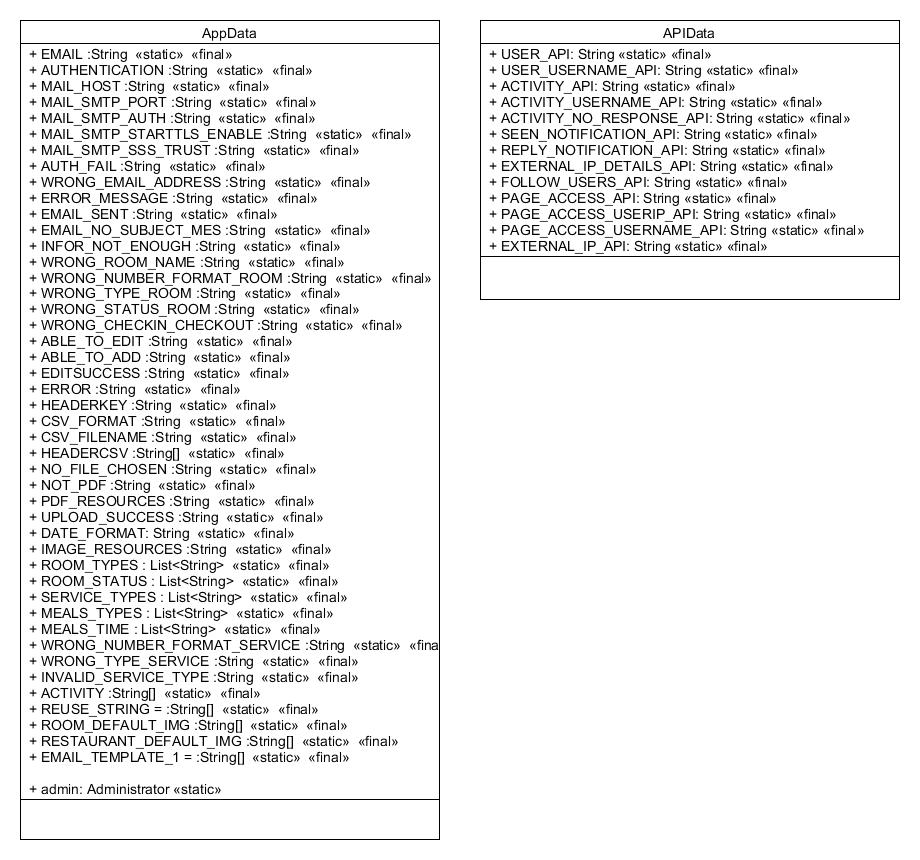
App Controller



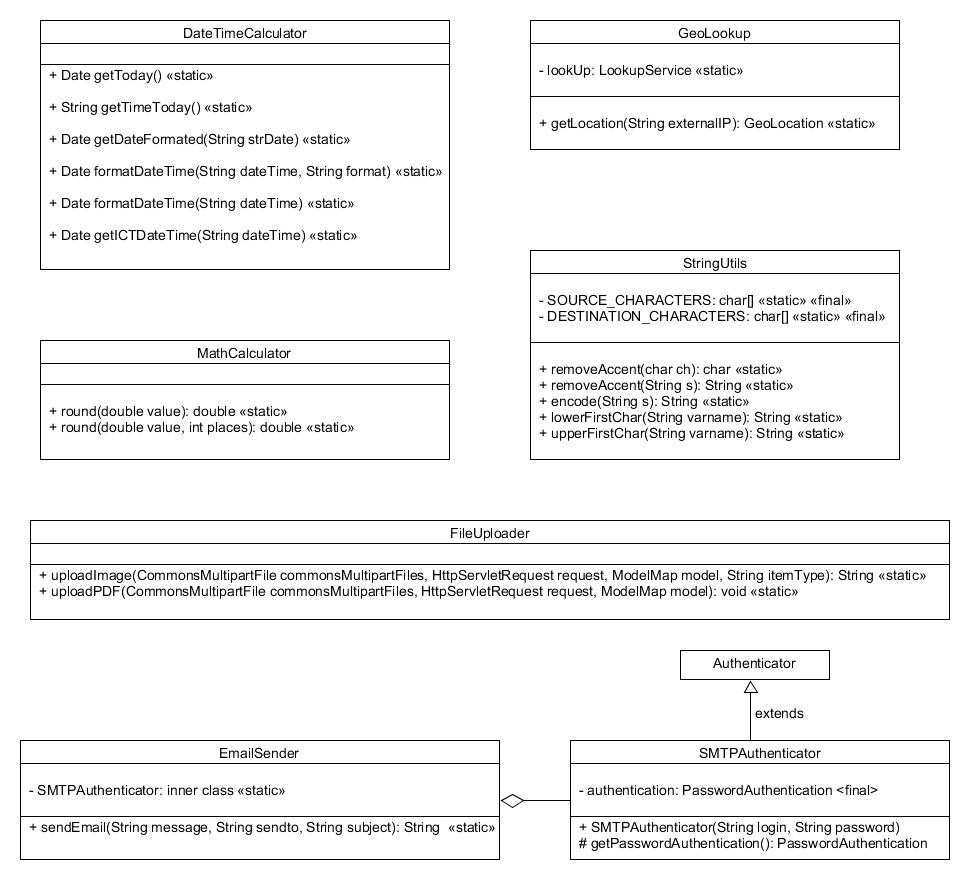
REST Controller



App CONST



Helper Class



Sequense 1:

Angular 2 route => Angular 2 Component => Angular 2 Service (Angular 2 Model) => API Route Nodejs = > Controller Nodejs => Model Nodejs => MongoDB (Const)

Sequense 2:

Angular 2 route => Angular 2 Component => Angular 2 Service (Angular 2 Model) => RestController => Service => DAO => Database (Model + Helper class)

Sequense 3:

JSP => AppController => Service => DAO => Database (Model + Helper class)

Sequense 4:

JSP + AngularJS => AppController + RestController => Service => DAO => Database (Model + Helper class)

Sequense 5:

EJS => Route Nodejs = > Controller Nodejs => Model Nodejs => MongoDB (Const)

User feature: Register, Login, Edit profile, Change Avatar, Change Password, Manage User

Activity feature: View Activity, Send Contact, Send Reservation Form, Feedback Hotel, Reply Message

Room feature: View Room, Book Room, Add Room, Edit Room, Delete Room, Feedback Room

Restaurant feature: View RestaurantItem, Add RestaurantItem, Edit RestaurantItem, Remove RestaurantItem.

Tracking feature: Track Users, View Tracking User Data, View Pie Chart Tracking Data by Country, View column chart Tracking Data by IP address or Username or Page Access, View GeoLocation Tracking data

Track Users feature:

User click page or click I or search or login or logout or ….

* Angular service call followUsers function => Route Nodejs => …

V/ Experiment and Result

1/ Experiments

* System run well on window
* Test on Linux
* Responsive website => test mobile UI
* Encrypt password
* Recommendation room
* Chart

2/ Evaluation

* Online single page application with high performance => speed? Compare with what?
* Dynamically loading? Why? prove
* Cross-platform system runs well with all operating system. => prove? Run well on window + Linux, + mobile?
* Friendly user interfaces, easy to use => show image
* Supports almost features for hotel bookings & reservations management. => show list features
* Ability to track user’s behavior => show chart, image
* Give examples, image to prove

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 with customers ‘s wish.

In the future, I will add more features, apply machine learning with 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, Nodejs, Express Framework. I can work with a lot 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

VII/ Appendix

1/ Use case bổ sung

Kẻ thêm nhưng bảng use case bổ sung cho phần software requirement, trên phần software requirement nói “để hiểu rõ về … xem phần appendix 1.1, 1.2…

2/ Sequence diagram bổ sung

* Draw some sequence diagram that represent for the whole system

3/ Test Case bổ sung

* Test case table

4 User Manual

* How to use + image

VIII/ Reference

<https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html>

http://terasolunaorg.github.io/guideline/1.0.1.RELEASE/en/Overview/SpringMVCOverview.html

http://viralpatel.net/blogs/spring3-mvc-hibernate-maven-tutorial-eclipse-example/