Travel Agency

Analysis and Design Document

Student: Debre Lóránd - Sándor

**Group: 30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 24.04.2019 | 1.0 | Project specification + Elaboration 1.1 | Debre Lóránd - Sándor |
| 08.05.2019 | 1.1 | Elaboration – Iteration 1.2 | Debre Lóránd - Sándor |
| 29.05.2019 | 2.0 | Final project documentation | Debre Lóránd - Sándor |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

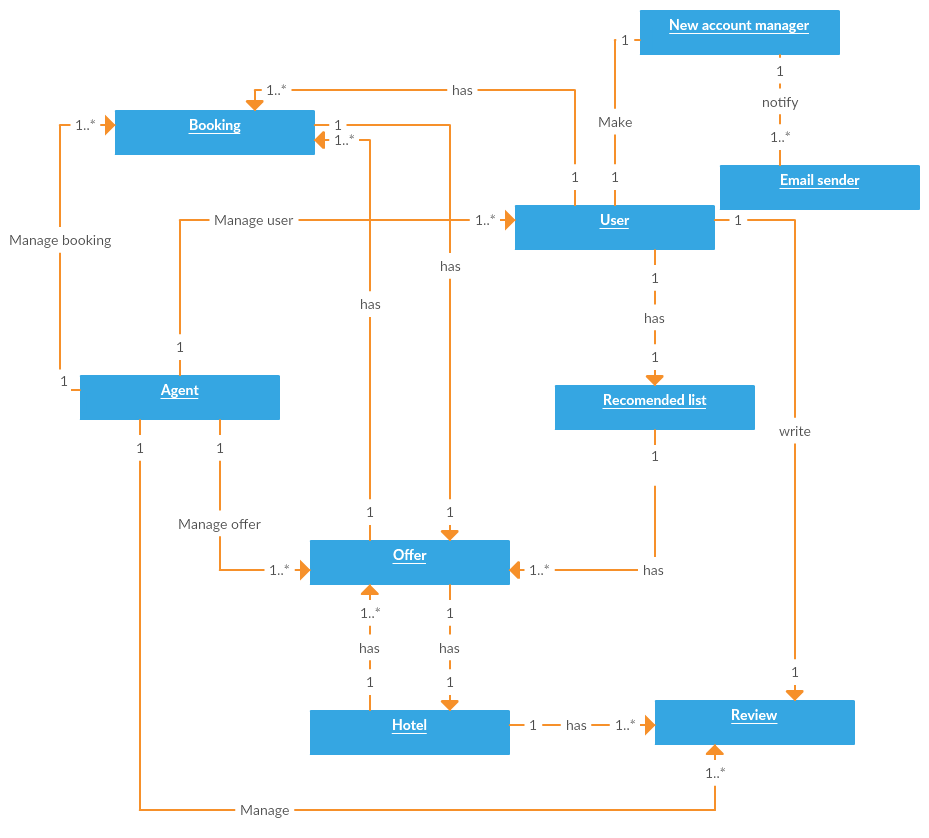
VI. Bibliography 5

# Project Specification

The application is an online web spring application that helps the users to choose the best holiday cottage. The main purpose of the application is to reduce as much as possible the search and booking time. The system also allows the user to see the description of each package, to read the reviews if exists and of course the user has the possibility to add packages to the chart and book them.

# Elaboration – Iteration 1.1

# Domain Model



(**Fig.1** Domain model diagram)

# Architectural Design

## Conceptual Architecture

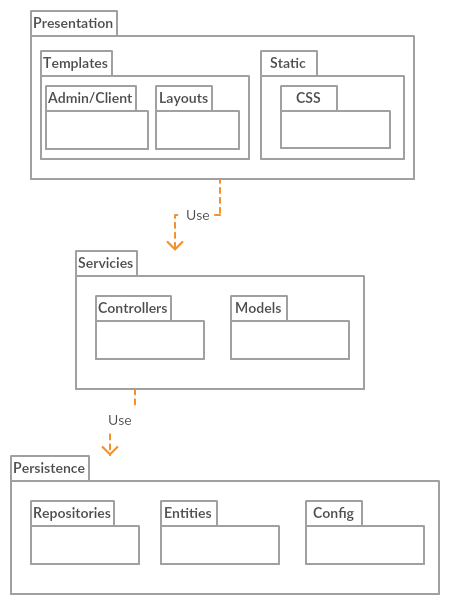
Client server architecture is used in this architecture, to separate the server-side application (data and logic) from the client-side (UI). This way the client-side and the server side may evolve independently of each other. The separation can be observed, where the HTTP calls are made over the network.

## 

(**Fig. 2** Conceptual architecture diagram)

## Package Design

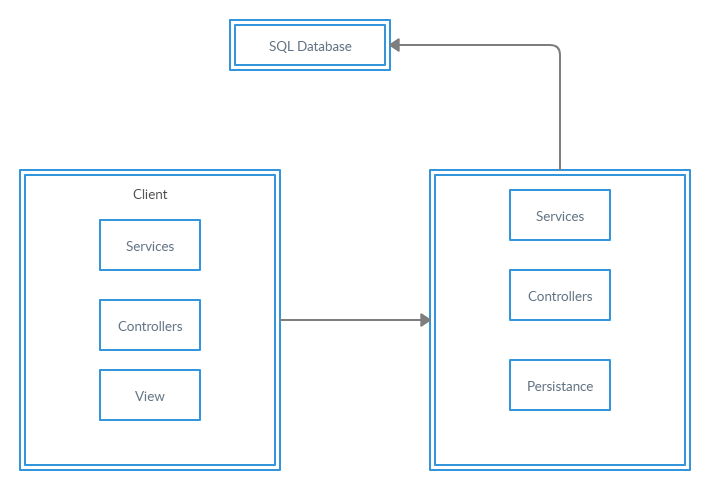
The package diagram of the system represents the layered structure of the application.



(**Fig. 3** Package design)

## Component and Deployment Diagrams

# 



(**Fig. 4** Component and deployment diagram)

# Elaboration – Iteration 1.2

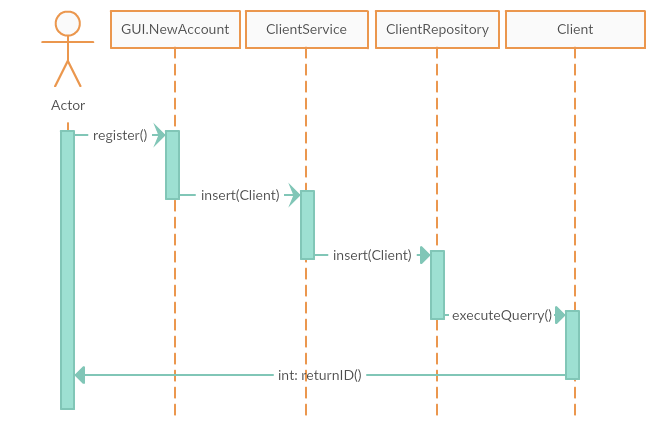
# Design Model

## Dynamic Behavior

Two relevant scenario are the following:

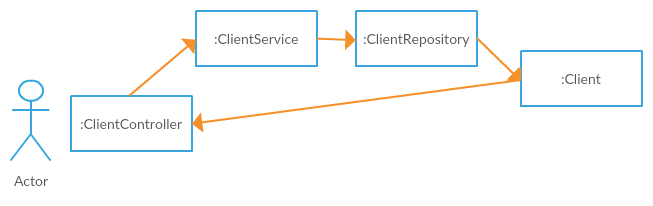
* The viewer wants to make a new account
* The client wants to make a reservation

**Use case 1**: The viewer wants to make a new account



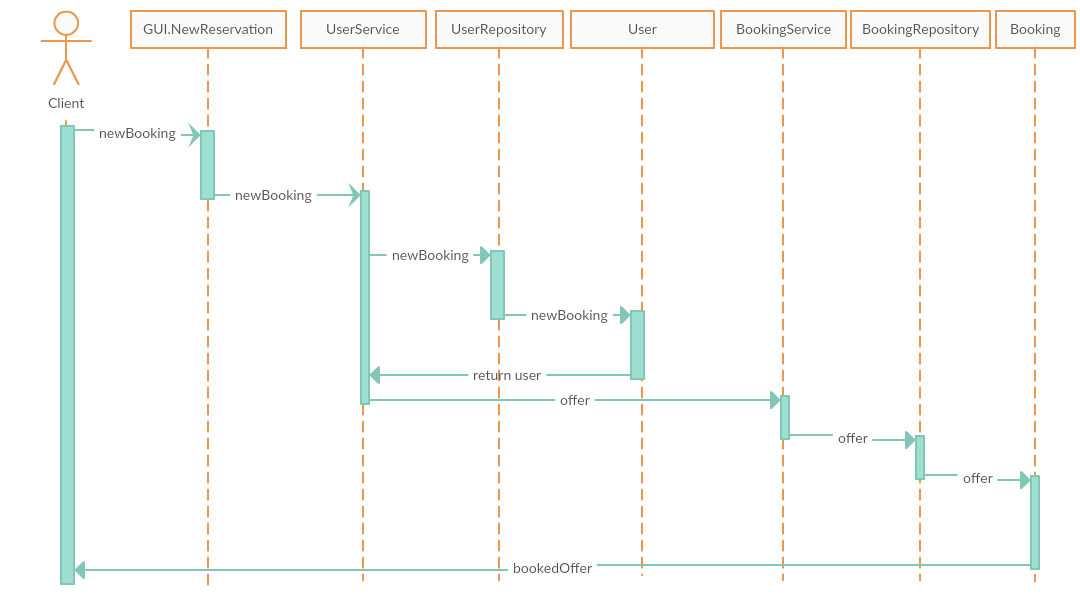
(**Fig. 5** Use case 1)

**Communication diagram 1**: The viewer wants to make a new account



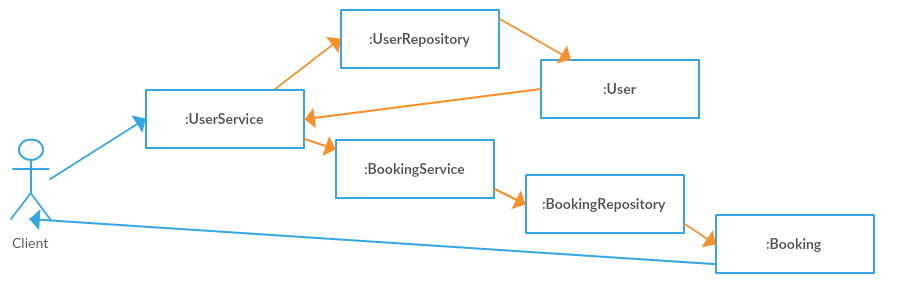
(**Fig. 6** Communication diagram 1)

**Use case 2:** The client wants to make a reservation



(**Fig. 7** Use case 2)

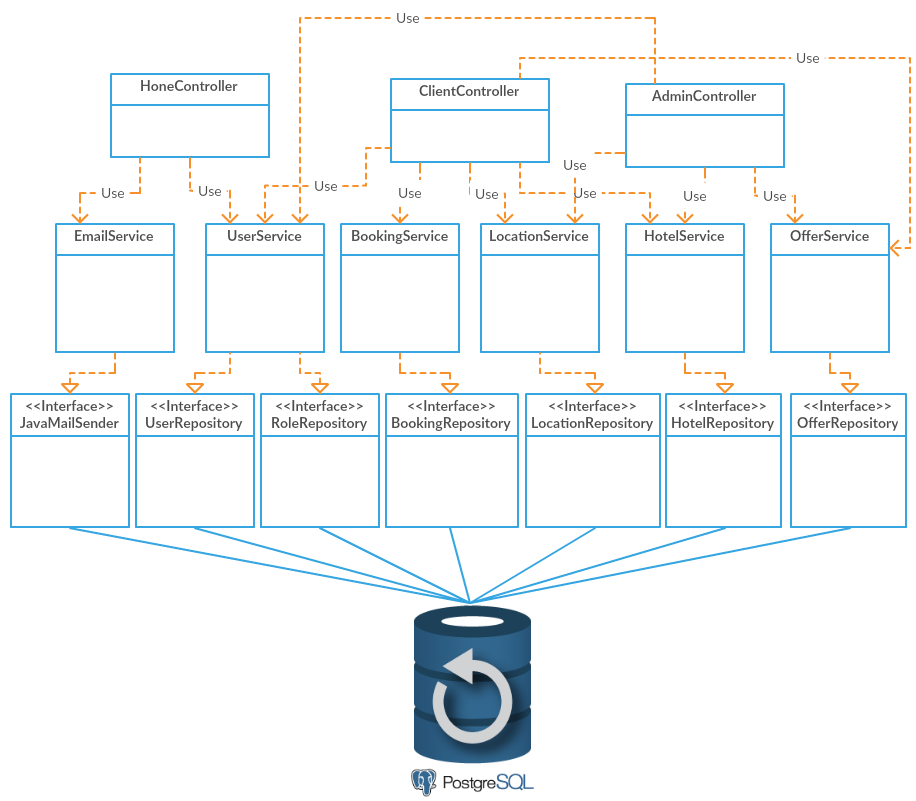
**Communication diagram 2**: The client wants to make a reservation



(**Fig. 8** Communication diagram 1)

## Class Design

.



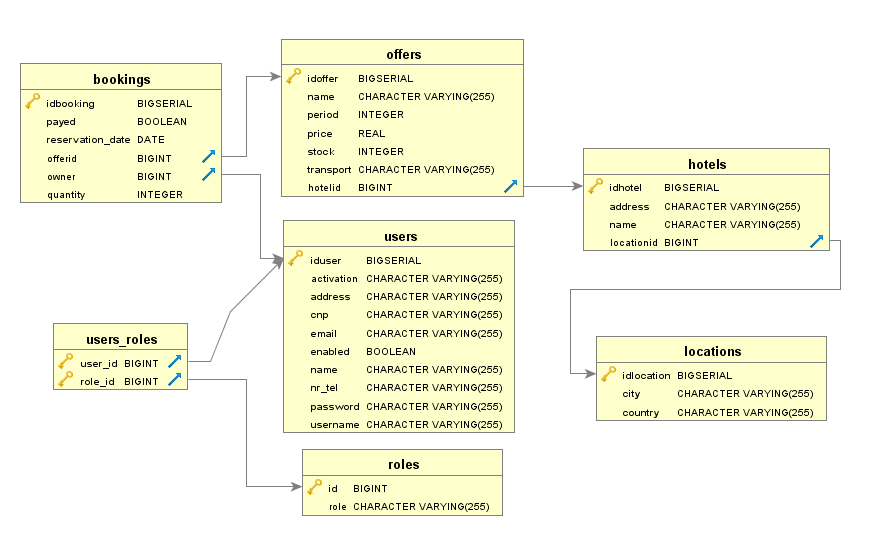
(**Fig. 9** Class diagram)

The used design patterns: dependency injection, builder design patter and repository pattern.

# Data Model

The persistance data will be stored in a PostgreSQL database. The data model closely resembles the domain model. In order to create the data model, first I designed domain classes and let the framework create the necessary tables. Bellow you can see the resulting tables of this process:

*.*



(**Fig. 10** PostgreeSQL data model)

# Unit Testing

I will use Junit/Mokito to test my application. An object under test may have dependencies on another objects. Unit test are performed on the repositories and services, both in sequential and concurrent conditions, to make sure they work perfect.

# Construction and Transition

# System Testing

The system has been tested using the client-side application. The user, which can be an admin or a regular user interacts directly with the application through the friendly user interface implemented using CSS and HTML5 description language.

The following usual and particular cases have been tested:

* The user log in, the username and password validation.
* The distinction between an user account and an admin account using springboot security package.
* The creation of a new user account and user account validation.
* The system response when the provided data for an user account is not valid.
* The reservation placing.
* The operations performed on the admin panel – add new locations, hotels, packages

# Future improvements

Multiple features could be added to the system, such as a new type of package, allowing users to rent a car. Each of these possible improvements have been accounted for in the design phase and the application should be easily extendable. More tests could be written to make sure that the application is properly functioning.

# Bibliography

* <https://www.mkyong.com/tutorials/spring-boot-tutorials/>
* <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658109(v=pandp.10)>
* <https://www.tutorialspoint.com/uml/uml_deployment_diagram.htm>