Hotel Manager

Analysis and Design Document

Student: Tothăzan Radu

**Group: 30236**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 05.05.2016 | 1.0 |  | Radu Tothăzan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 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

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

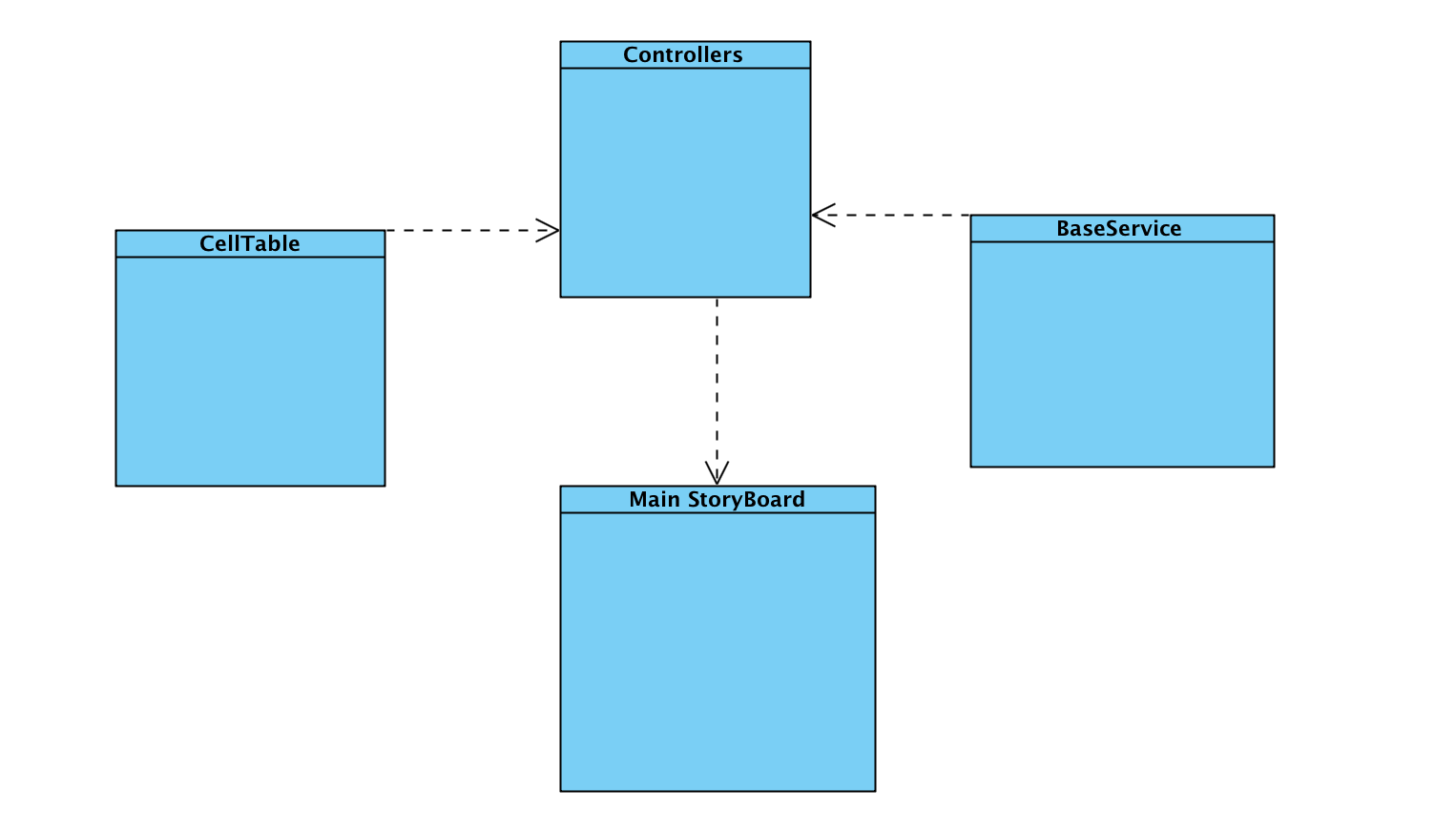
# Project Specification

Design and implement a client-server application for the employees of a hotel. The application should have two types of users (a regular user represented by the receptionist, and an administrator user) which have to provide a username and a password in order to use the application. The regular user can perform the following operations:

Calculate room rates. If a client has reserved a room several times in the past automatically apply a reduction for loyalty, Book rooms for clients, Cancel room bookings, Issue invoices. The administrator can perform the following operations: CRUD operations on user accounts, Issue reports in different formats (xml, txt) with the room bookings from a chosen date interval, the clients that have the largest number of reservations, etc.

# Elaboration – Iteration 1.1

# Domain Model

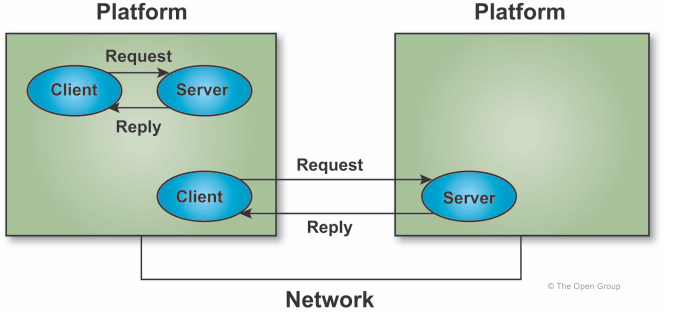


# Architectural Design

## Conceptual Architecture

Modelul client-server este o structură sau arhitectură aplicație distribuită care partajează procesarea între furnizorii de servicii numiți servere și elementele care solicită servicii, numite clienți. Clienții și serverele comunică printr-o rețea de calculatoare, de obicei prin Internet, având suporturi hardware diferite, dar pot rula și pe același sistem fizic. Un server (fizic) rulează unul sau mai multe programe server, care partajează resursele existente cu clienții. Clientul nu partajează niciuna dintre resursele proprii, ci apelează la resursele serverului prin funcțiile server. Clienții inițiază comunicația cu serverele și așteaptă mesajele acestora. Pentru menținerea legăturii între cei doi, indiferent de pauzele care intervin, se folosește conceptul de sesiune, care de obicei este limitată în timp.

Aplicatia principala este compusa din doua aplicatii secundare ,una care face partea de client si una care face partea de server.Clientul comunica cu serverul prin ConnectionUtils,care face legatura dintre Controllerul din Client si cel din Server.



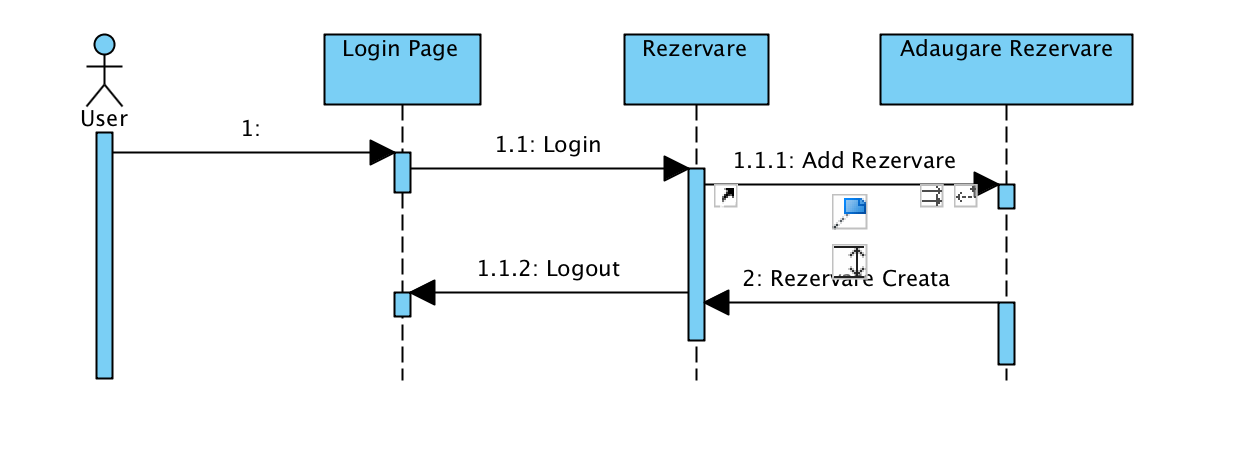
## Component and Deployment Diagrams

# *overview-serverAndContainers*

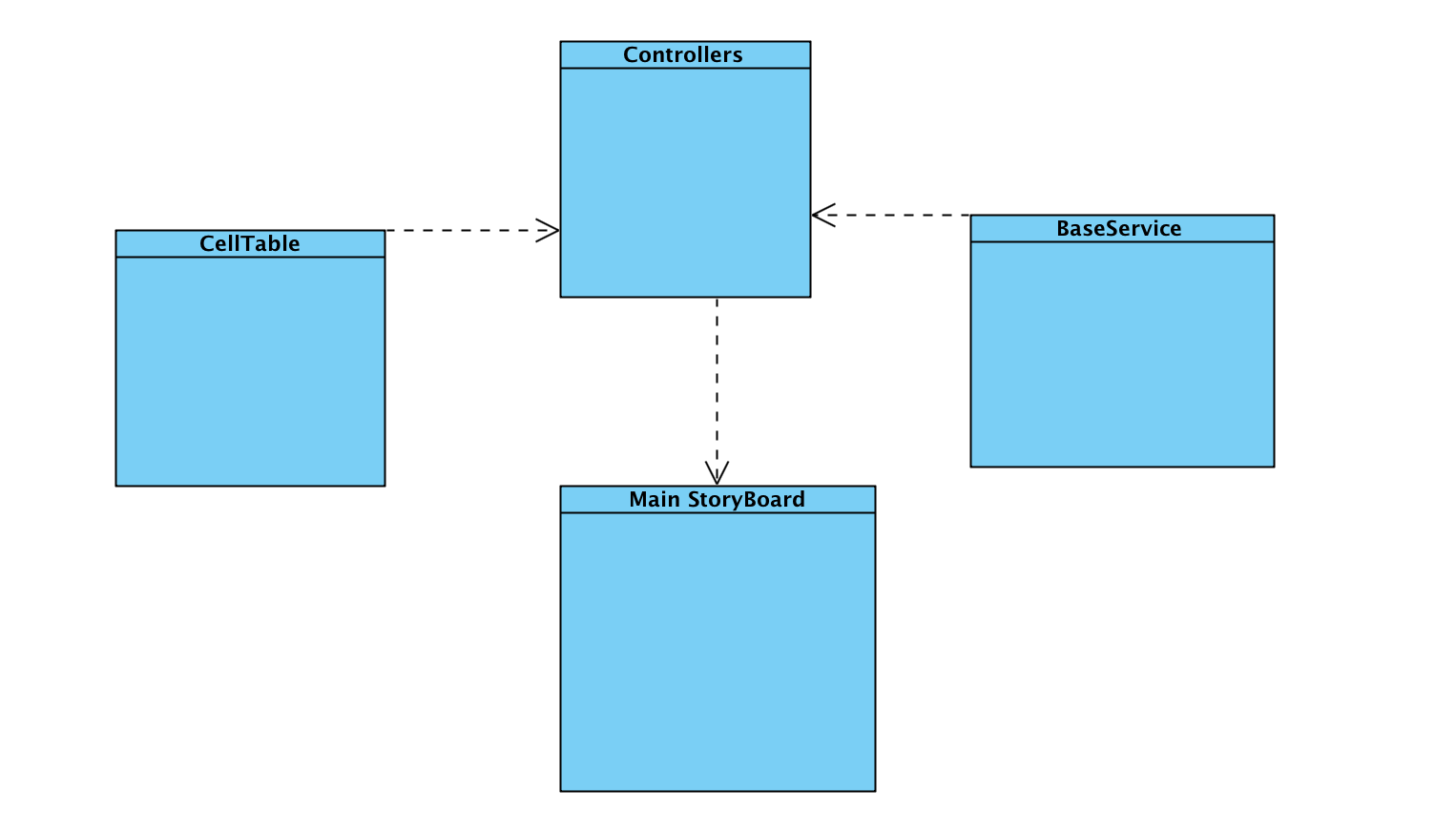
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior



## Class Design



# Data Model

Pentru conectarea la baza de date am folosit Firebase cocapods. Dupa instalarea SDK-ului Firebase, conexiunea arata asa:

var ref : FIRDatabaseReference!

self.ref = FIRDatabase.database().reference()

Aceste 2 linii de cod trebuie introduse in toate clasele unde este nevoie de o conexiune la baza de date.

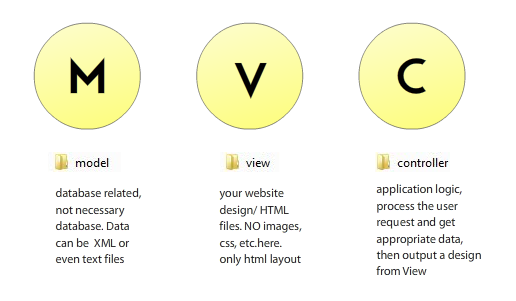
# Elaboration – Iteration 2

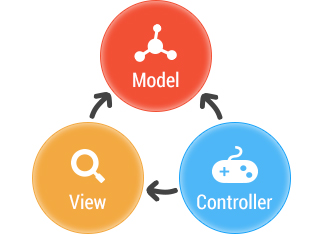
# Architectural Design Refinement

Am folosit Client/Server Architectural Design pentru a face o aplicatie dinamica.In aplicatia client folosim mai multe metode cum ar fi, PUT,GET,DELETE,depinde de cerinta pe care o are clientul la baza de date.

# Design Model Refinement

Arhitectural Pattern folosit in aplicatia web pe care am realizat-o este MVC pattern.Acesta se refera la Model, View si Controller.Acest pattern implica relatia dintre cele 3 entitati.





Controllerul comunica cu ambele entitati, atat cu Model cat si cu View.Astfel din imaginea de mai sus putem observa faptul ca View-ul este dependent de Model, luand informatii din acesta, dar Model nu ia niciodata informatii din View.

# Construction and Transition

# System Testing

All of the testing I've done to this application is manually. I have tested plenty scenarios to be sure the application have no bugs.

# Future improvements

Ulterior ar putea sa fie dezvoltata aplicatia prin optimizarea vitezie si prin imbunatatirea interfetei.

# 

# Bibliography

www.stackoverflow.com

https://firebase.google.com/docs/

And other tutorials from youtube.com and google.com