Conference Manager

Analysis and Design Document

Student: Tothazan Radu

**Group: 30236**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

# Create an application that helps managing scheduling and ticketing for conferences.

You are free to develop one (or a mix) of desktop/web/mobile application .

The data will be stored in a relational database.

Use the Layers architectural pattern organise your application.

Use a domain logic pattern (transaction script or domain model) / a data source hybrid pattern (table module, active record) and a data source pure pattern (table data gateway, row data gateway, data mapper) most suitable for the application

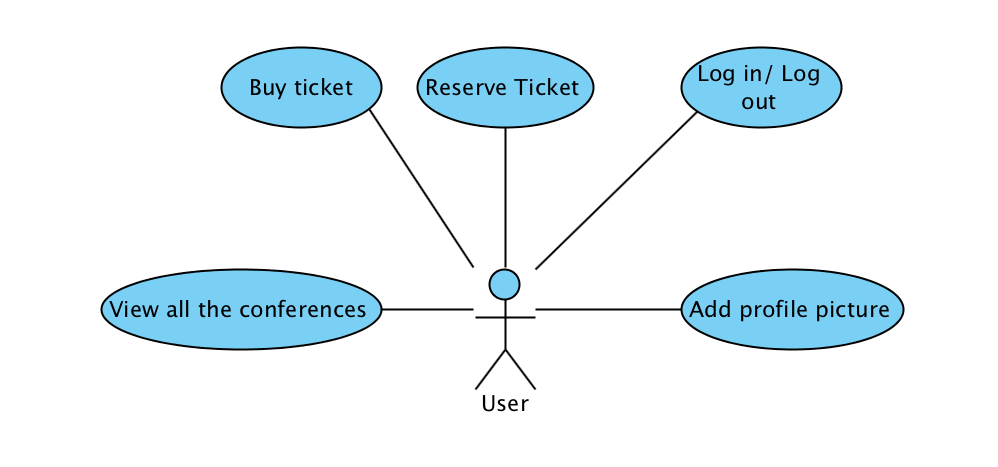
# Non-functional Requirements

# This assignment gave us the choice to choose between a desktop/mobile or a web application.

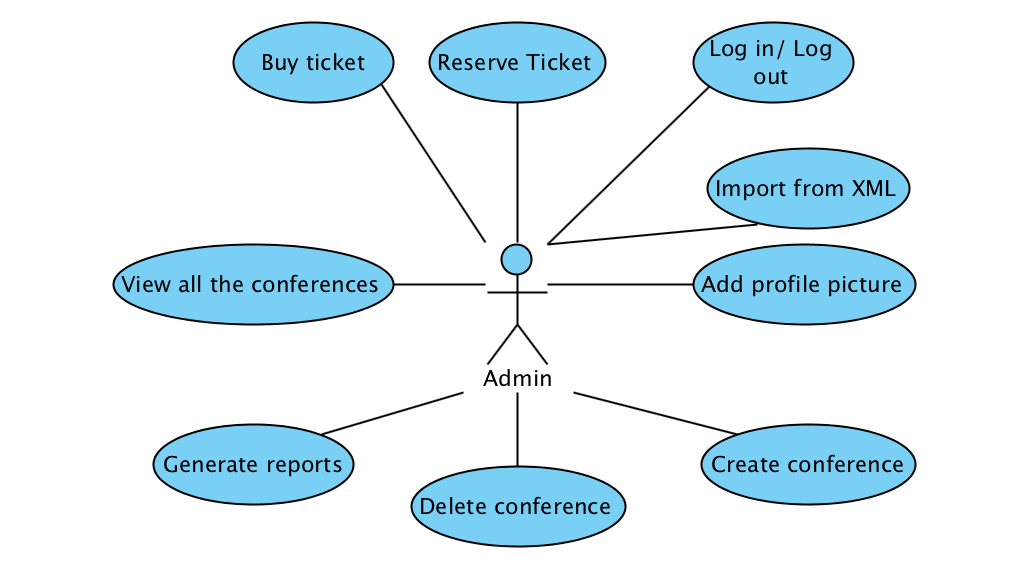
I chose to develop an iOS application, because I want to learn swift language.

2. Use-Case Model

If you are an unregistered user, you can view the information about every conference and create an account.

If you are an registered user you can log in or log out from the application. After logging in, you can see the conferences and their information, you can add a profile picture, reserve a ticket for a specific conference and later you can purchase the ticket from your cart. After buying the ticket, the application sends to the buyer an email with the billing information.

If you are an admin, you have all the above features and you can create a conference, you can delete a conference, you can generate reports, you can import from an xml talks.



3. System Architectural Design

**3.1 Architectural Pattern Description**

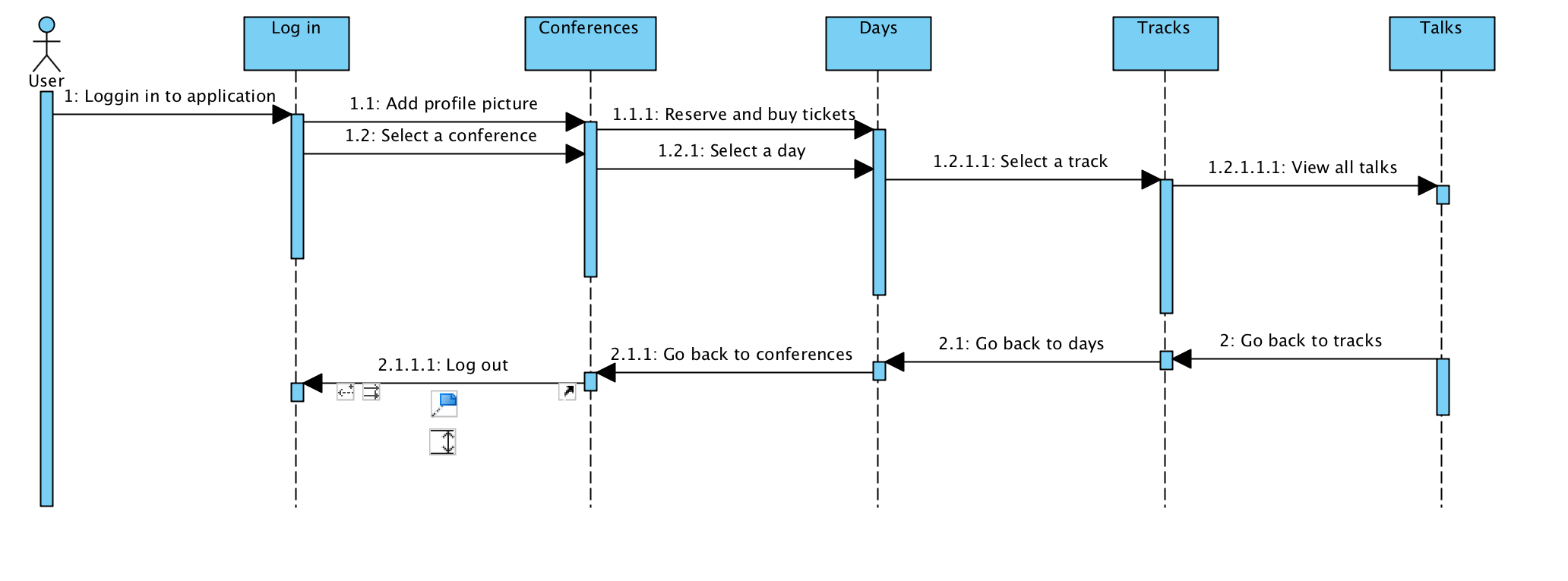
The architectural pattern used in this web application is the MVC pattern. MVC stands for Model, View, Controller. This pattern requires a specific type of relation between this three entities.

As we can see, The controller communicates with both Model and View. And the view has a dependence of the Model, but the Model never uses information from the View.

In the class diagram displayed below, can be seen the similarity between the classes from the application, and the structure of MVN architectural pattern.

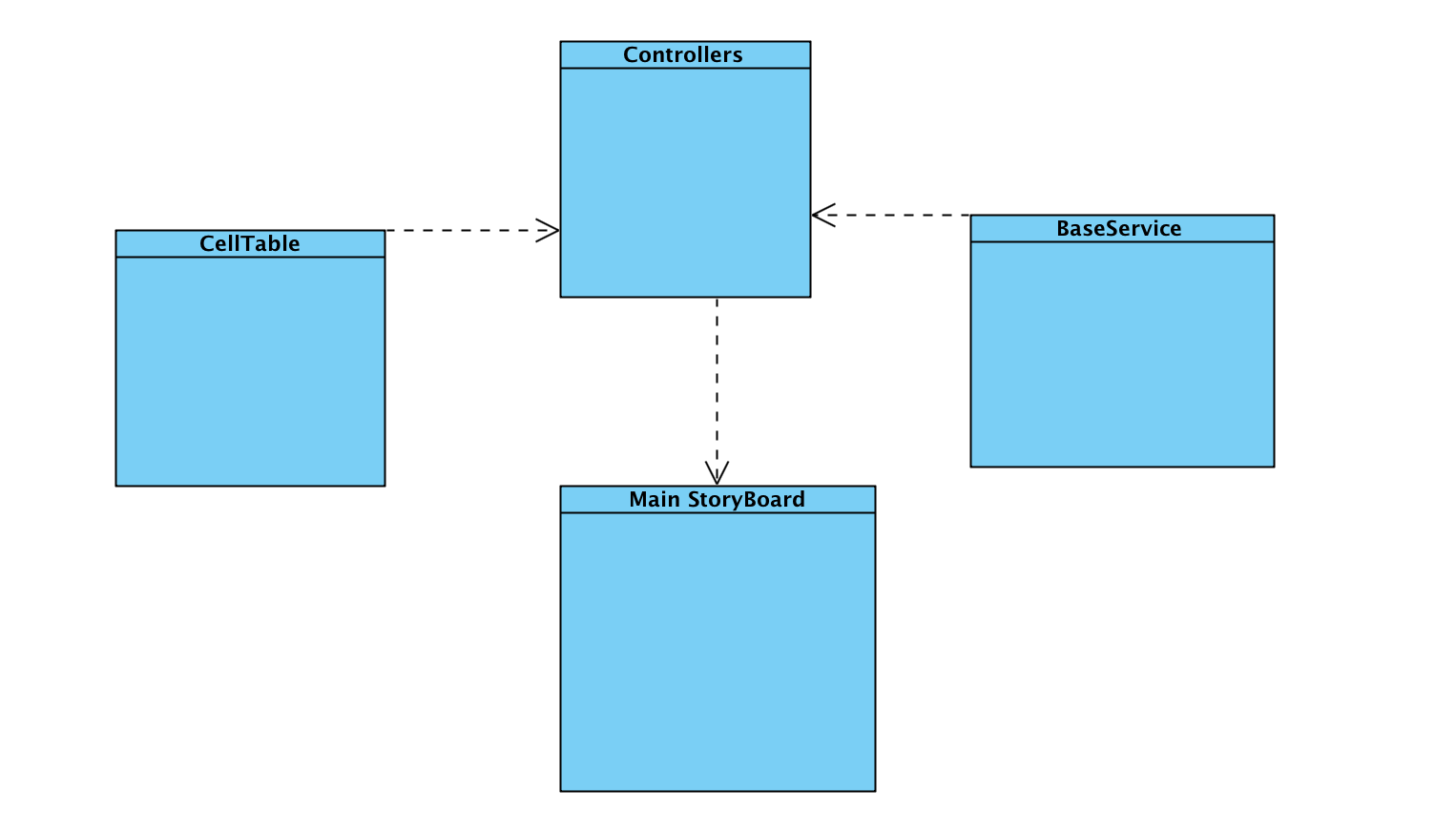
As with assignment 2, the structure of the application remained the same. I used the same architectural pattern, the MVC ( Model View Controller)

4. UML Sequence Diagrams



5. Class Design

**5.1 UML Class Diagram**



This is a package diagram ( class ) . This diagram shows us the relationships between classes, because the relationship between two packages is the same to the relationship between two classes from every package.

6. Data Model

For manipulating the data, the application uses a connection with Firebase. The connection to our server is in BaseService class. I have also made some changes in the Info.plist file for this connection to work. I have used CocoaPods to install the Firebase SDK. My connection to firebase looks like this:

let FIREBASE\_REF = Firebase (url: "https://ps-radu.firebaseio.com"

7. 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.

8. Bibliography

www.stackoverflow.com

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

And other tutorials from youtube.com and google.com