

Politecnico di Milano

A.A. 2015-2016

Software Engineering 2: “myTaxiService”

Requirements Analysis and Specifications

Document

Version 1.0

Milica Jovanovic (mat. 835953), Pavle Vidanovic (mat. 854472)

23 October 2015

Contents

[1 Introduction 7](#_Toc433566068)

[1.1 Purpose 7](#_Toc433566069)

[1.2 Scope 7](#_Toc433566070)

[1.3 Glossary 7](#_Toc433566071)

[1.4 Reference Documents 8](#_Toc433566072)

[1.5 Document Overview 8](#_Toc433566073)

[2 Overall Description 9](#_Toc433566074)

[2.1 Product perspective 9](#_Toc433566075)

[2.2 Identifying stakeholders 9](#_Toc433566076)

[2.3 User characteristics 9](#_Toc433566077)

[2.4 Actors identifying 9](#_Toc433566078)

[2.5 Goals 9](#_Toc433566079)

[2.6 Domain properties 10](#_Toc433566080)

[2.7 Constraints 10](#_Toc433566081)

[2.7.1 Regulatory policies 10](#_Toc433566082)

[2.7.2 Hardware limitation 10](#_Toc433566083)

[2.7.3 Interfaces to other applications 10](#_Toc433566084)

[2.7.4 Parallel operation 10](#_Toc433566085)

[2.8 Assumptions 10](#_Toc433566086)

[2.9 Future possible implementation 11](#_Toc433566087)

[3 Specific Requirements 12](#_Toc433566088)

[3.1 External Interface Requirements 12](#_Toc433566089)

[3.1.1 User Interfaces 12](#_Toc433566090)

[3.1.2 API interfaces 16](#_Toc433566091)

[3.1.3 Hardware Interfaces 16](#_Toc433566092)

[3.1.4 Software Interfaces 16](#_Toc433566093)

[3.2 Functional Requirements 17](#_Toc433566094)

[3.3 The world and the machine 19](#_Toc433566095)

[3.4 Scenarios 20](#_Toc433566096)

[3.4.1 Scenario 1 20](#_Toc433566097)

[3.4.2 Scenario 2 20](#_Toc433566098)

[3.4.3 Scenario 3 20](#_Toc433566099)

[3.4.4 Scenario 4 20](#_Toc433566100)

[3.5 UML Models 21](#_Toc433566101)

[3.6 Non Functional Requirements 43](#_Toc433566102)

[4 Appendix 45](#_Toc433566103)

[4.1 Alloy 45](#_Toc433566104)

[4.2 Software and tool used 56](#_Toc433566105)

[4.3 Hours of works 57](#_Toc433566106)

[5 Revision 57](#_Toc433566107)

[5.1 Changed Assumptions 57](#_Toc433566108)

[5.2 Removed goal and functionality 57](#_Toc433566109)

[5.3 Modified Functional Requirements 57](#_Toc433566110)

[5.4 Modified Scenarios and Use Cases 58](#_Toc433566111)

[5.5 Modified Diagrams 59](#_Toc433566112)

* + 1. [G1] Allow a visitor to became a registered user and choose

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | the public or private nature of his/her calendar. . . . . . | 22 |
|  | 3.2.2 | [G2] Allow user to log in to application. . . . . . . . . . . | 22 |
|  | 3.2.3 | [G3] Allow user to create a new event in the calendar and |  |
|  |  | choose the public or private nature. . . . . . . . . . . . . | 22 |
|  | 3.2.4 | [G4] Allow user to modify an existing event of his/her |  |
|  |  | calendar. . . . . . . . . . . . . . . . . . . . . . . . . . . . | 23 |
|  | 3.2.5 | [G5] Allow user to delete an existing event of his/her cal- |  |
|  |  | endar. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 23 |
|  | 3.2.6 | [G6] Allow user to invite/delete other user to a specific |  |
|  |  | event of his/her calendar. . . . . . . . . . . . . . . . . . . | 23 |
|  | 3.2.7 | [G7] Allow user to see the weather forecast of a specific |  |
|  |  | event of his/her calendar. . . . . . . . . . . . . . . . . . . | 23 |
|  | 3.2.8 | [G8] Allow user to see the public event of other registered |  |
|  |  | user. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 24 |
|  | 3.2.9 | [G9] Allow user to see event to which has been invited. . . | 24 |
|  | 3.2.10 | [G10] After login, application will notify only the creator user three days before an event takes place if the weather |  |
|  |  | is not good. . . . . . . . . . . . . . . . . . . . . . . . . . . | 24 |
|  | 3.2.11 | [G11] After login, application will notify invited user one |  |
| days before an event takes place if the weather is not good. 24 | |
| 3.3 | The world and the machine . . . . . . . . . . . . . . . . . . . . . | | 25 |
| 3.4 | Scenarios . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 26 |
|  | 3.4.1 Scenario 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 26 |
|  | 3.4.2 Scenario 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 26 |
|  | 3.4.3 Scenario 3 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 26 |
|  | 3.4.4 Scenario 4 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 27 |
|  | 3.4.5 Scenario 5 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 27 |
|  | 3.4.6 Scenario 6 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 27 |
|  | 3.4.7 Scenario 7 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 27 |
|  | 3.4.8 Scenario 8 . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 28 |
| 3.5 | UML Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 29 |
|  | 3.5.1 Use Case . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 29 |
|  | 3.5.1.1 Vistiros registers to MetoCal . . . . . . . . . . . | | 29 |
|  | 3.5.1.2 login . . . . . . . . . . . . . . . . . . . . . . . . . | | 32 |
|  | 3.5.1.3 Create new event on calendar . . . . . . . . . . . | | 34 |
|  | 3.5.1.4 User modifies an event . . . . . . . . . . . . . . | | 36 |
|  | 3.5.1.5 User deletes an event from calander . . . . . . . | | 38 |
|  | 3.5.1.6 User invites other user to an event . . . . . . . . | | 40 |
|  | 3.5.1.7 Remove user from an event guest list . . . . . .  3.5.1.8 User modify the event date after application has notified him/her that there will be bad weather | | 42 |
|  | and suggest him/her the closest sunny day. . . . | | 44 |
|  | 3.5.1.9 User accept invitation . . . . . . . . . . . . . . .  3.5.1.10 User sees notification for modified event with an- | | 46 |
|  | other invitation . . . . . . . . . . . . . . . . . . | | 48 |

3.5.1.11 User sees details of an event . . . . . . . . . . . 50

* + 1. Class Diagrams . . . . . . . . . . . . . . . . . . . . . . . . 52
    2. State Machine Diagrams . . . . . . . . . . . . . . . . . . . 53

3.6 Non Functional Requirements . . . . . . . . . . . . . . . . . . . . 54

* + 1. Performance Requirements . . . . . . . . . . . . . . . . . 54
    2. Design Constraints . . . . . . . . . . . . . . . . . . . . . . 54
    3. Software System Attributes . . . . . . . . . . . . . . . . . 54
       1. Availability . . . . . . . . . . . . . . . . . . . . . 54
       2. Maintainability . . . . . . . . . . . . . . . . . . . 54
       3. Portability . . . . . . . . . . . . . . . . . . . . . 54

3.6.4 Security . . . . . . . . . . . . . . . . . . . . . . . . . . . . 54

* + - 1. External Interface Side . . . . . . . . . . . . . . 54
      2. Application Side . . . . . . . . . . . . . . . . . . 55
      3. Server Side . . . . . . . . . . . . . . . . . . . . . 55

1. Appendix 56
   1. Alloy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56
      1. Data Type . . . . . . . . . . . . . . . . . . . . . . . . . . 56
      2. Abstract Entity . . . . . . . . . . . . . . . . . . . . . . . . 57
      3. Abstrac Entity Implementation and Signature . . . . . . . 58
      4. Fact . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
      5. Assert . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
      6. Predicates . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
      7. Result . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
      8. Generated world . . . . . . . . . . . . . . . . . . . . . . . 64
   2. Software and tool used . . . . . . . . . . . . . . . . . . . . . . . . 68
   3. Hours of works . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
2. Revision 68
   1. Changed Assumptions . . . . . . . . . . . . . . . . . . . . . . . . 68
   2. Removed goal and functionality . . . . . . . . . . . . . . . . . . . 69
   3. Modified Functional Requirements . . . . . . . . . . . . . . . . . 69
   4. Modified Scenarios and Use Cases . . . . . . . . . . . . . . . . . 70
   5. Modified Diagrams . . . . . . . . . . . . . . . . . . . . . . . . . . 70

# Introduction

## Purpose

The purpose of this document is to provide a comprehensive description of the myTaxiService system. It’s aim is to communicate what the software should do and identify the capabilities and characteristics of the system being developed, as well as the constraints it should respect. This document is meant for everyone included in the production of the software.

## Scope

The aim of this project is to develop and implement myTaxiService, an application similar to Uber, which makes the process of assigning an available taxi vehicle to possible passengers.

The developed system should allow new users to register. Users, once logged in, should be able to:

* request a taxi
* reserve a taxi
* cancel a ride
* check taxi availability around him
* receive a confirmation with information about the assigned vehicle and ETA once taxi is requested
* create/maintain user profile

The developed system should allow new taxi drivers to register. Drivers, once logged in, should be able to:

* inform the system about their availability
* confirm that they are going to take care of a certain call
* create/maintain taxi driver profile

The system should keep information about new arrived requests, as well as the confirmed rides. A ride should have and id number, information about the passenger that requested the ride, as well as the code of the assigned vehicle and ETA. System should also keep information about taxi queues connected to particular zone of the city and ensure fair management of the queues. Developed system should keep information about the list of reservations made by passengers, such as id number of the reservation, information about the passenger that made the reservation and the time of reservation and time of the ride.

## Glossary

The following are the definitions of some commonly used phrases throughout the document:

|  |  |
| --- | --- |
| *ETA* | Estimated Time of Arrival, approximated time of arrival of taxi vehicle to destination |
| *Reservation* | Passenger request for a vehicle at least 2 hours before the ride |
| *Request* | Passenger filled form for immediate ride |
| *Reservation* *Conformation* | Notification sent to the user about the confirmed reservation |
| *Ride Conformation* | Notification sent to the user about the confirmed ride with information of the ride |
| *User* | A person already registered and logged into the system |
| *Guest* | A person accessing a system that has either never registered or hasn't logged in yet. Guest has only two available options, to log in or to register for the first time |
| *Taxi* *driver* | A person already register and logged into the system as a driver |
| *GPS* | Global Positioning System |
| *API* | *:* Application Programming Interface*.* |

Podsetnik

Acronyms

* RASD: Requirements Analysis and Specification Document.
* DB: DataBase.
* DBMS: DataBase management system.
* Database Management System ( DBMS ).
* OS: Operating System.

## Reference Documents

* IEEE Std 830-1998 Recommender Practice for Software Requirements Specifications
* Specification Document: myTaxiService Project AA 2015-2016.pdf

## Document Overview

The document is essentially structured in four parts:

* Chapter 1: Introduction, gives description of document and some basic information about the software
* Chapter 2: Overall Description, gives an overview of the main functionalities of the software to be with constraints, as well as the hardware limitations.
* Chapter 3: Specific Requirements, body of the document that describes in more detail functionalities, possible scenarios and use cases.
* Chapter 4: Appendices, Alloy simulation of the previous UML diagrams.

# Overall Description

## Product perspective

*myTaxiService* is mobile web application that provides user with services described in section 1.2. System will consist of two applications and server between them. The software will be developed using a client-server model. The server side contains the application logic and is used to interact with permanent storage, serve pages to the client and process user input. The web client consists of dynamic web pages which provides user friendly graphical interface and the web browser through which they are accessed. Considering *myTaxyService* is a mobile web application, it is platform independent. The only requirement is users having a web browser installed on a device of their choice.

## Identifying stakeholders

There are two distinct interest groups of people regarding this project:

* *Company* that provided project specification and expect it to be delivered in a way that satisfies given specification while respecting the set deadline and budget
* *Developer group,* in this case group of two people
* *Taxi driver,* person working for the company that ordered the software product
* Passenger, person who need a ride to specific location

## User characteristics

myTaxiService is expected to have users across a wide range of demographics, meaning users of any age, gender and educational background. Still, given of the ubiquitous nature of internet and social media, it is assumed that people using our software do have the basic web browsing skills.

## Actors identifying

Three possible actors interacting with our system are the following:

* *Guest* person accessing a system that has either never registered of hasn't logged in yet. Guest can only access the initial page from where he has only two available options, to log in or to sign up for the first time
* *User* a person already registered and logged into the system. User can use all of the features offered by the passenger application
* *Taxi**driver*a person already registered and logged into the system. User can use all of the features offered by the taxi driver application
* *Admin*- a person responsible for handling reports on users of my taxi service. Admin can ban a user or a driver from the system

## Goals

Having possible users in mind, myTaxyService should have these features:

* [G1] registering new user
* [G2] login to existing user’s account
* [G3] managing user's profile
* [G4] requesting a taxi
* [G5] reserving a taxi
* [G6] canceling a ride
* [G7] checking taxi availability around user
* [G8] receiving notifications about the reservation or request confirmation
* [G9] confirming/declining a ride(taxi driver)

## Domain properties

It is supposed that these conditions hold in the analyzed world:

* the passenger needs a ride to specific location
* the details of the ride provided by the passenger are accurate
* money exchange between the passenger and the taxi driver is made independently from the myTaxiService system
* distinction between the zones are clearly defined

## Constraints

### Regulatory policies

myTaxiService application will not take advantage of users personal information and will respect the privacy policy. User will be notified about it.

### Hardware limitation

User must have access to Internet and own a device with a web browser and GPS service.

### Interfaces to other applications

myTaxiService application is integrated with Google Maps API to access their maps, Google Places API to suggest street addresses to the user and email service in order to make authentication.

### Parallel operation

myTaxiService will support parallel access to the applications database in a transparent way.

## Assumptions

Considering that there were some ambiguities in the specification document, the following facts are assumed:

* users register with their email, password, name, surname and phone number
* users can change their email and password
* user have only one account
* user provides accurate information
* if users location is not available, the application will show a screen with an option of typing your current address
* there is a Terms & Conditions section to indicate clearly the usage of the application, which if not followed will result in account deactivation
* we assume that Google Maps service will calculate location used by myTaxiService accurately
* we assume that taxi driver will respect the ETA, otherwise they could be banned from the system
* if a taxi driver has an unexpected issue, the user will be automatically notified by the system and a new vehicle will be assigned to him with new ETA

## Future possible implementation

* + - another way of payment will be added as an options for users to pay online
    - an option for users to rate the drivers
    - Facebook authentication could be added as a way of registering
    - application could be updated so it works more efficiently
    - taxi sharing options could be added to the system

# Specific Requirements

## External Interface Requirements

### User Interfaces

In this section mockups of myTaxiService application will be enclosed which will mimic the general idea of the software system:

In figure 1. Initial page is shown, which offers two possibilities to the guest user. Guest user could create an account if he does not own one or he could sign in with his credentials.

In figure 2. Sign Up page is shown, which provides user with input fields to add his account information. After the fields are filled user can confirm by clicking the Ok button.

|  |  |
| --- | --- |
| C:\Users\milica13629\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1 Guest screen.png |  |
| Figure 1. Initial page | Figure 2. Sign Up |

In figure 3. Sign In page is shown, which provides user with input fields to add credentials. After the fields are filled user can sign in by clicking the button.

In figure 4. Passenger Home page is shown, which shows user’s location and a drop down menu with different options. From this page user can choose option to request a taxi, reserve a taxi, change his profile, go to Help section or simply log out from his account.

|  |  |
| --- | --- |
| C:\Users\milica13629\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3 Sign In.png | C:\Users\milica13629\AppData\Local\Microsoft\Windows\INetCache\Content.Word\4 Home screen.png |
| Figure 3. Sign In | Figure 4. Passenger Home page |

In figure 5. Request page is shown, which provides user with possibility to request a vehicle from his current location showed on the map.

In figure 6. Reservation page is shown, which provides user with possibility to reserve a vehicle from a certain source address to certain destination address. User also has to provide time of the ride.

|  |  |
| --- | --- |
| C:\Users\milica13629\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5 Request.png |  |
| Figure 5. Request page | Figure 6. Reservation page |

In figure 7. Driver Home page is shown, which provides driver with possibility to state his availability, manage his profile and check information about his current ride.

In figure 8. Pop up message is shown, it informs a driver that a certain passenger has requested his service. Driver has the possiblility to accept or decline the ride.

|  |  |
| --- | --- |
|  |  |
| Figure 7. Driver Home page | Figure 8. Pop up message |

In figure 9. Navigation page is shown, which provides the driver with the route from his location to the source address of the requested ride.

In figure 10. Taxisavailable screen is shown, which shows user taxi vehicles available in his zone.

|  |  |
| --- | --- |
|  |  |
| Figure 9. Navigation page | Figure 10. Taxis available screen |

In figure 11. Notification screen is shown, which shows list with all drivers requests ad reservations. It both shows requests that are completed or canceled and the requests that are active.

In figure 12. Reservation details screen is presented, which shows all informations about the ride and enables quick access to user’s profile.

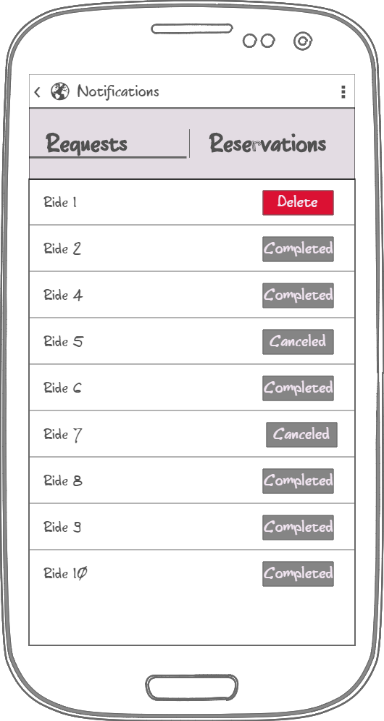
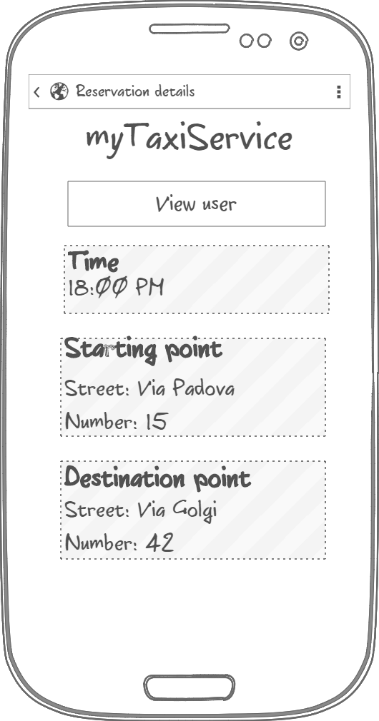


Figure 11. Notification page

Figure 12. Reservation details

### API interfaces

In myTaxiService application Google Maps API (https://developers.google.com/maps) is used to represent the map around user’s current location, as well as to provide routes between two given addresses. Google Places API (https://developers.google.com/places) is used to provide the user with autocompleting addresses and to suggest addresses. Email service is used to allow the user to authenticate himself with his email address, a link will be sent to given email address and by clicking the link user confirms his identity. This service is used as a security measure.

### Hardware Interfaces

This project does not support any hardware interfaces.

### Software Interfaces

* Database Management System (DBMS): MySQL
* Programming technologies: HTML5, CSS3, jQuery, PHP
* Application server: Apache
* Operating System ( OS ): Cross platform

## Functional Requirements

3.2.1 [G1] Allow a guest to became a registered user and choose his role in the myTaxiService system

* [R1] Guest must not be already registered to perform sign up process.
* [R2] Guest must choose a username or in our case email address that is not already used by another user.
* [R3] User cannot sign up twice but only once for session.
* [R4] Visitor can just see Guest screen.
* [R5] Guest can only access to registration form in signup screen.
* [D1] Email address used for signing up must be formally correct.

3.2.2 [G2] Allow user to sign in to application.

* [R1] User must be already registered to success to sign in process.
* [R2] User must know his email address and password used during registration to success login.
* [R3] Email address and password insert during to sign in process must be correct.
* [R4] Wrong credentials will not grant access to user.
* [R5] Guest can’t access to calendar page before registration.
* [R6] Application will not implement retrieve password mechanism.

3.2.3 [G3] Allow user to manage his profile

* [R1] User must be already registered and signed in the application.
* [R2] User must confirm updating process.

3.2.4 [G4] Allow user to request a taxi

* [R1] User must be already registered and logged in the application.
* [R2] User must specify starting location
* [R3] User must confirm request.

3.2.5 [G5] Allow user to make a reservation of a taxi

* [R1] User must be already registered and signed in the application.
* [R2] User must specify origin of the ride
* [R3] User must specify destination of the ride
* [R4] User must specify meeting time at least 2 hours before the ride

3.2.6 [G6] Allow user to cancel a ride

* [R1] User must be already registered and signed in the application.
* [R2] User can view list of all his rides in notification area
* [R3] User must select an active ride that he wants to cancel and confirm.

3.2.7 [G7] Allow user to see available taxi vehicles around him

* [R1] User must be already registered and signed in the application.
* [R2] User must have available location settings

3.2.8 [G8] Allow user to receive notifications and request confirmations

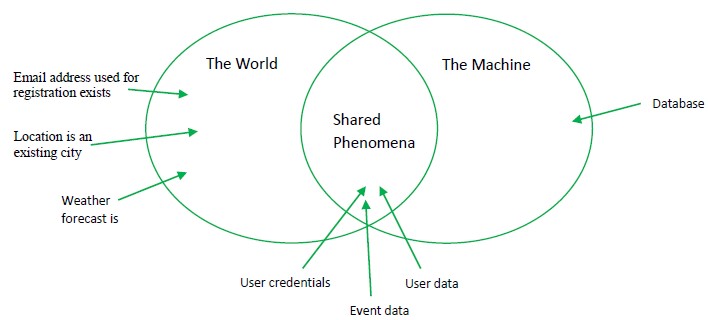
* [R1] Taxi driver must be already registered and signed in the application.
* [R2] Taxi driver must set his availability option

3.2.9 [G9] Allow driver the option to confirm or decline request for a ride

* [R1] Taxi driver must be already registered and signed in the application.
* [R2] Taxi driver must set his availability option in order to receive requests
* [R3] Taxi driver can confirm or decline request that he get

## The world and the machine

For a first domain analysis of MeteoCal application we use “The World & Macchine” model by M. Jackson & P. Zave. This approach let us identify the entities inside the domain that interact with the application (“The World”), entities to be developed (“The Machine”) and the intersection (“Shared Phenomena”) between the world and the application, that are all world informations known or managed direclty by the application.



## Scenarios

### Scenario 1

Emily is running late to a Saturday evening house party. The metro is about to close, so she needs to find a ride quickly. She has heard that there is new well-functioning application called myTaxiService and she wanted to give it a try. She picked up her smart phone and typed in the web address of the app. A login page has appeared and she had to register in order to use it. Emily quickly typed in her personal information and authenticated by email. After that she requested a ride from her current location, and instantaneously received a notification with ETA and code of the incoming taxi. Taxi arrived in few minutes and her dinner hasn’t got cold.

### Scenario 2

Dan and his friends are planning to go play football in an indoor pitch court. So Dan searched and found free court today from 16 to 17 o’clock so he booked that court. His friends live close to the court or near the metro station. Dan isn’t that lucky so he must find a ride that will be secure and that will respect his time schedule. His friends told him about an application myTaxiService that can solve his problems. He easily created an account and found an option of reserving a taxi for the time of the match. He made a reservation 3 hours before the meeting point and received a confirmation of the reservation. 10 minutes before the ride he received a message containing the information about the assigned vehicle. That night his team won the game.

### Scenario 3

Economist Jack has lost his job because of the world economic crisis and he found a part-time job at city taxi service. The taxi service uses well-functional cross platform web application. So jack registered as a driver and started to work hardly. He was assigned to work in zone 1, close to the central train station. Being assigned at that area he had a lot of calls from passengers. Around noon when is the rush hour he was really busy so he had to reject some rides when his vehicle was occupied. As soon as he finishes a ride he change his status to available so he could get a contacted by the taxi system about other rides.

### Scenario 4

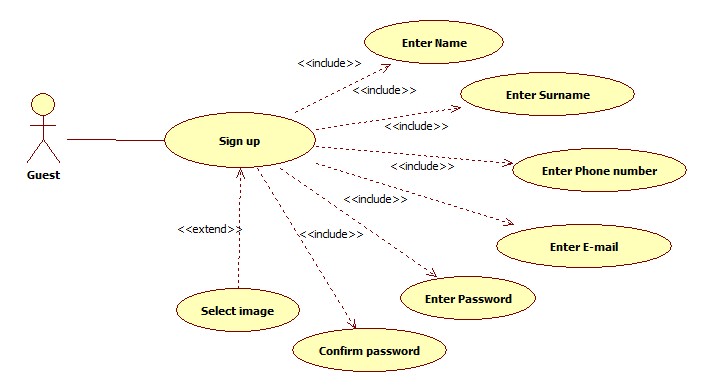
Catherine as a well experienced F1 driver got hurt in a car accident and after the recovery her doctor told her that she has to give up that extreme sport and find a nice, less invasive job. She thought about it intensively and she figured out that she couldn’t live her life without driving a car so with her resume she managed to get a job easily as a taxi driver. After a few days working as a taxi driver she still couldn’t remember all the street addresses, but luckily myTaxiService application when accepting a request for a ride offers an navigation option from her current location to the passenger location and after that as well as the route from starting to destination point. When Catherine found out about this convenience she felt so relieved that she didn’t have to remember all the routes around the city.

## UML Models

### Use Case

After stating all the desired features, goals and requirements, and describing possible scenarios we can identify some use cases. The diagrams are shown and described below.

#### Accessing the application



|  |  |  |
| --- | --- | --- |
| **Name** | **SignUp** | |
| **Actors** | | Guest | |
| **Entry conditions** | | The guest has never registered in the system | |
| **Event Flow** | | * Guest navigates to the myTaxiService homepage containing a form asking him to enter basic information, his email, password, his phone number and image which is optional. * Upon the form submission system checks if the information entered is valid i.e. if all the fields are filled and email as username is not already in use. * If the entered information is valid, the user will get confirmation email with which he will be automatically logged in to the system and redirected to his personal home page | |
| **Exit Conditions** | | The information about the new user is correctly stored, a welcome email is sent and the user can use all functions of myTaxiService | |
| **Exceptions** | | if the information the user entered is not valid, an appropriate message is displayed and the guest is asked to enter his email and password again | |

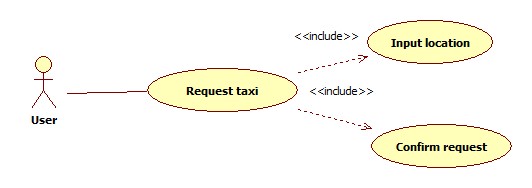


|  |  |  |
| --- | --- | --- |
| **Name** | **SignIn** | |
| **Actors** | | Guest | |
| **Entry conditions** | | The guest has already registered in the system and knows his email and password. | |
| **Event Flow** | | * After navigating to myTaxiService homepage or the first screen if he uses the cross platform application containing a form asking him to enter his email as username and password. * Upon the form submission system checks if the username and password match an existing user * If the entered information is valid, the user is signed in to the system and redirected to his personal page | |
| **Exit Conditions** | | The user is granted access to all of the myTaxiService's functionalities | |
| **Exceptions** | | If he information the user entered is not valid, an appropriate message is displayed and the guest is asked to enter his username and password again | |

#### Managing user’s profile

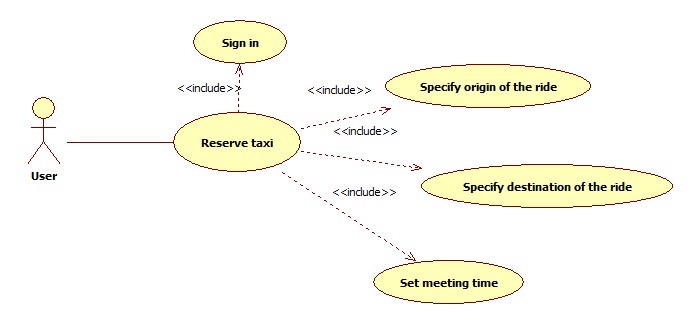
|  |  |  |
| --- | --- | --- |
| **Name** | **Manage Profile** | |
| **Actors** | | User | |
| **Entry conditions** | | The user is logged in | |
| **Event Flow** | | * User chooses the Manage Profile option from the drop down menu in home screen * User is redirected to a Manage profile screen where he/she can change the following information:   + name   + surname   + password   + phone number   + image (optional) * User submits the filled out form | |
| **Exit Conditions** | | The new user’s data are correctly stored and the account is successfully edited. | |
| **Exceptions** | | If the user submits the form leaving any field empty, an appropriate message will be displayed and the user will be asked to fill out the missing information | |

#### Requesting a taxi



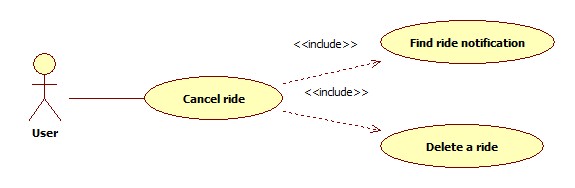
|  |  |  |
| --- | --- | --- |
| **Name** | **Request taxi** | |
| **Actors** | | User | |
| **Entry conditions** | | The user is logged in | |
| **Event Flow** | | * User chooses the *Request taxi* option from drop down menu of the home screen * User is redirected to a Request screen where he/she can select desired location on the map where to catch a taxi. * After selecting a location user submits request simply by clicking request button. | |
| **Exit Conditions** | | The request is send to myTaxiService system. And confirmation with code of incoming taxi and ETA. | |
| **Exceptions** | | If the user doesn’t select location on the map, an appropriate message will be displayed and the user will be asked to choose starting point location on the map. | |

#### Reserving a taxi



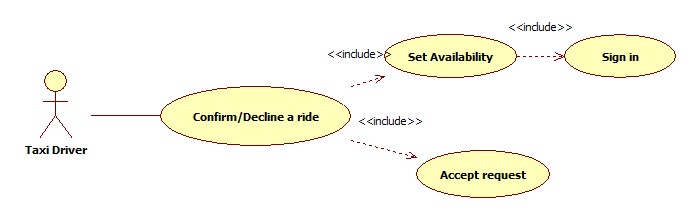
|  |  |  |
| --- | --- | --- |
| **Name** | **Reserve taxi** | |
| **Actors** | | User | |
| **Entry conditions** | | The user is signed in | |
| **Event Flow** | | * User chooses the *Reserve taxi* option from drop down menu of the home screen * User is redirected to a Reserve screen where he/she must:   + Specify Origin of the ride   + Specify destination of the ride   + Set meeting time * After selecting a location user submits request simply by clicking Ok button. | |
| **Exit Conditions** | | The request of reservation is send to myTaxiService system. User receives confirmation of the reservation immediately. And 10 minutes before the ride he/she receives message with the information of the assigned vehicle. | |
| **Exceptions** | | * If user doesn’t fill all the fields the warning message will be displayed and the user will be asked to fill the missing information. * If the doesn’t reserve a taxi at least 2 hours before the ride. The appropriate message will be displayed and the user will be asked to set new time. | |

#### Canceling a taxi



|  |  |  |
| --- | --- | --- |
| **Name** | **Cancel ride** | |
| **Actors** | | User | |
| **Entry conditions** | | The user is signed in | |
| **Event Flow** | | * User chooses the *Ride notifications* option from drop down menu of the home screen * User is redirected to a Notifications screen where he/she can delete the active requests or reservations for a ride. * Cancel request must be send at most 10 minutes after the request or maximal 15 minutes before the reservation time. | |
| **Exit Conditions** | | The cancel request is sent to myTaxiService, and after that system respond with confirmation of cancel request. | |
| **Exceptions** | | * If user cancel request later than 10 minutes after request he will need to pay compensation for the delay. * If user cancel request later than 10 minutes before the reservation he will need to pay compensation for the delay. | |

#### Confirm/Decline a ride



|  |  |  |
| --- | --- | --- |
| **Name** | **Confirm/Decline a ride** | |
| **Actors** | | Taxi Driver | |
| **Entry conditions** | | The driver is signed in | |
| **Event Flow** | | * Taxi Driver control availability by switching YES/NO button on his home page that indicates his status. * Taxi Driver switches his status button to YES. * Taxi Driver accept or decline a request by clicking confirm or decline option on a pop-up request. | |
| **Exit Conditions** | | If taxi driver confirmed a ride, he can use navigation to get to destination point. He does that by clicking *Confirm ride* button in his home page, and after that he is redirected to navigation screen. | |
| **Exceptions** | | * If driver have some problem with the car and can’t reach destination point in the agreed time he/she can cancel the ride. | |

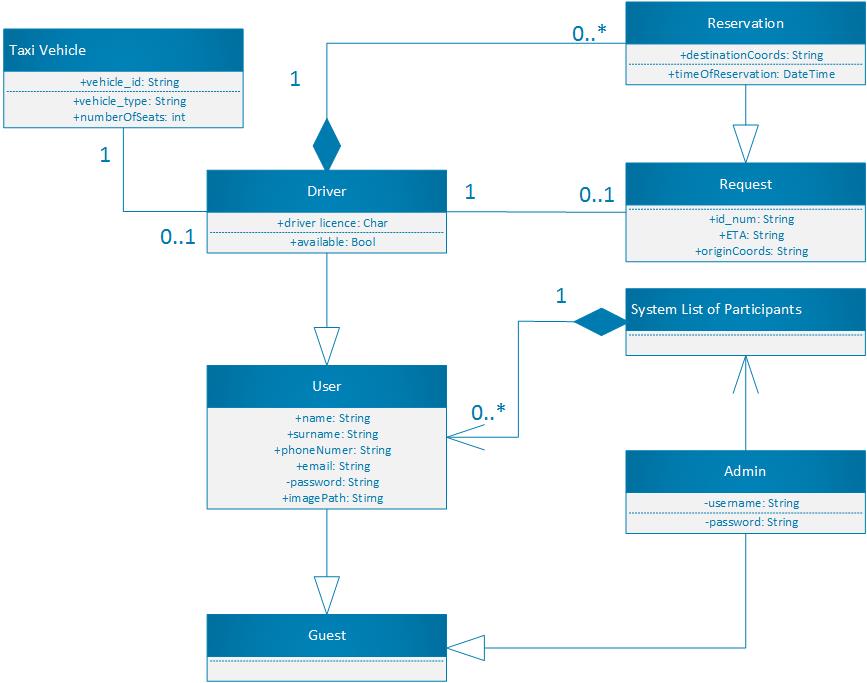
#### Check taxis available



|  |  |  |
| --- | --- | --- |
| **Name** | **Check Taxis Available** | |
| **Actors** | | User | |
| **Entry conditions** | | The driver is signed in | |
| **Event Flow** | | * Taxi Driver control availability by switching YES/NO button on his home page that indicates his status. * Taxi Driver switches his status button to YES. * Taxi Driver accept or decline a request by clicking confirm or decline option on a pop-up request. | |
| **Exit Conditions** | | If taxi driver confirmed a ride, he can use navigation to get to destination point. He does that by clicking *Confirm ride* button in his home page, and after that he is redirected to navigation screen. | |
| **Exceptions** | | * If driver have some problem with the car and can’t reach destination point in the agreed time he/she can cancel the ride. | |

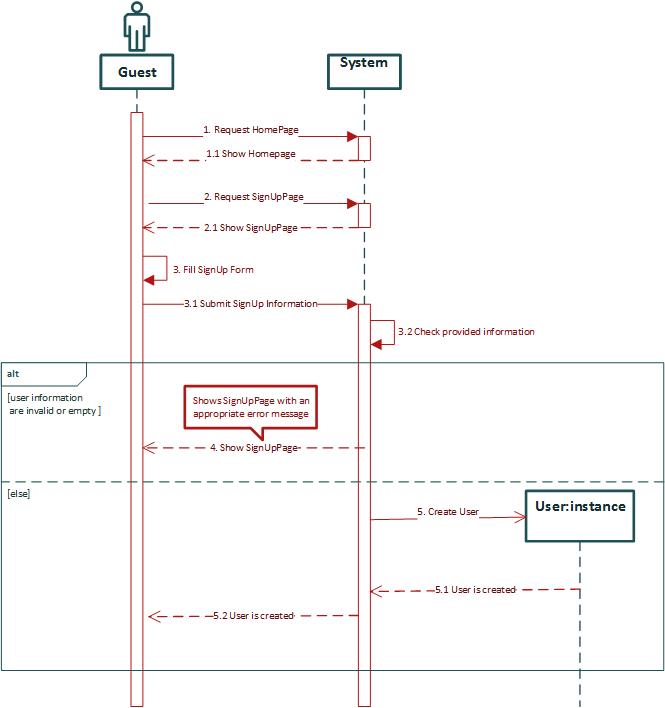
### Class Diagram

Here is presented the class diagram. This diagram will be updated during the developing process.

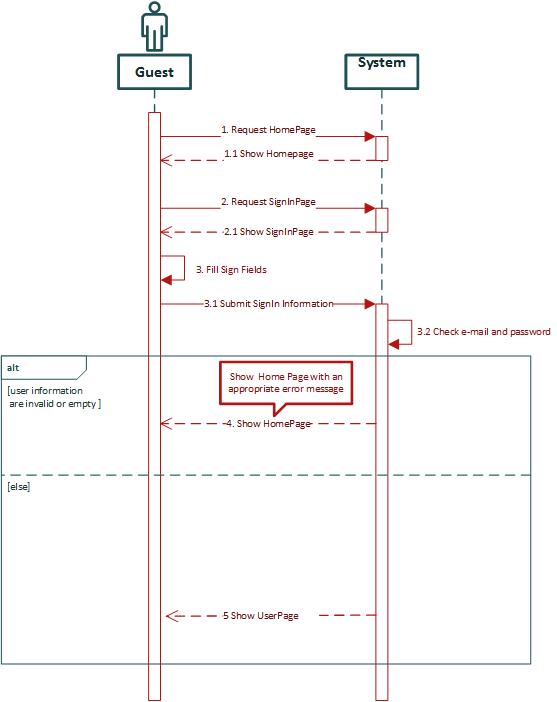


### Sequence Diagram

#### SignUp

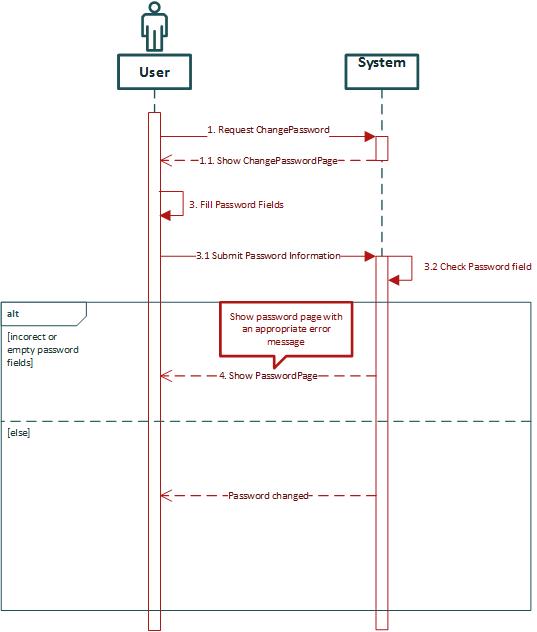


#### SignIn

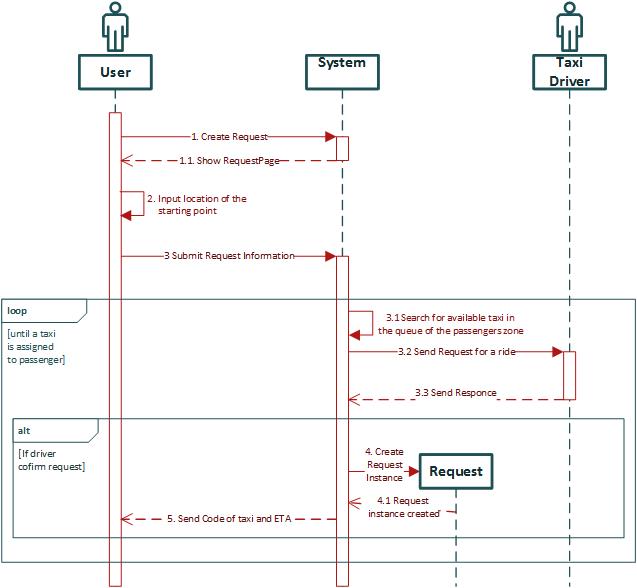


#### Manage profile

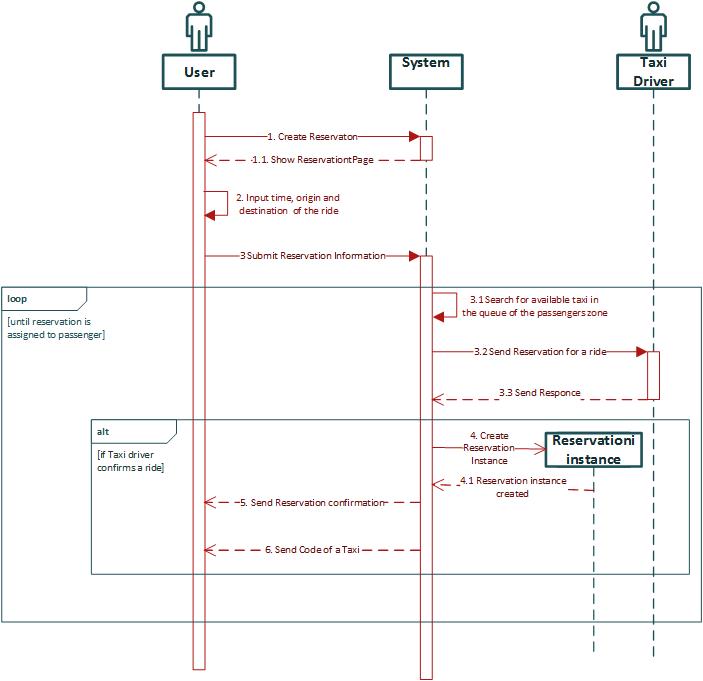
In this sequence diagram as example of managing user profile, Change password sequence of actions are shown.



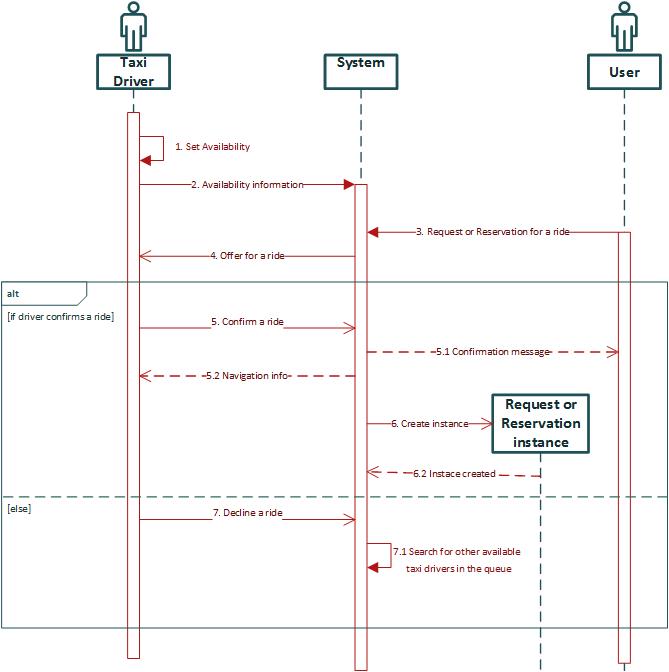
#### Request a ride



#### Reserve a ride

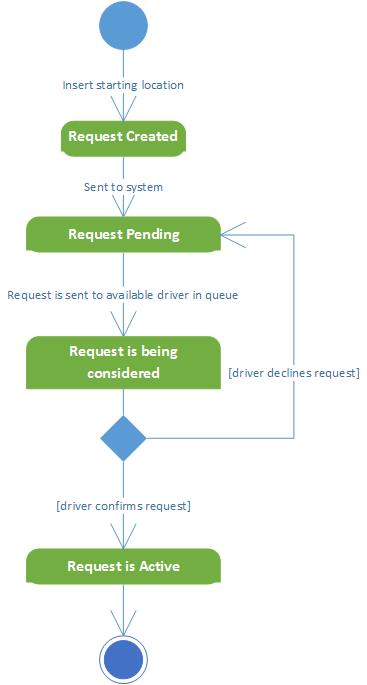


#### Confirm/Decline a ride

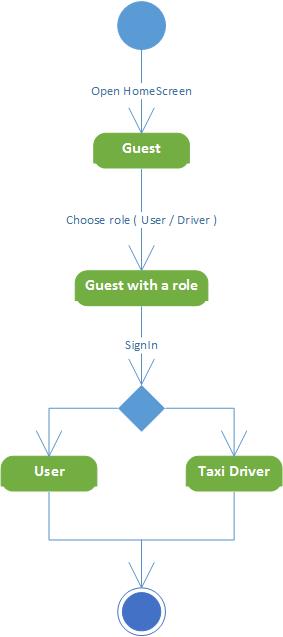


### State Chart diagrams

#### Request lifecycle



#### Visitor lifecycle



## Non Functional Requirements

3.6.1 Performance Requirements

Performance must be acceptable to guaranteed a good grade of usability. We assume the response time of the system is close to zero, so the performance are essentially bounded by user’s internet connection.

3.6.2 Design Constraints

The application will be developed with Java EE so it will inherit all language’s constraints.

3.6.3 Software System Attributes

3.6.3.1 Availability The application will be accessible online anytime. To achieve this goal could be necessary to use a dedicated server but to guarantee more availability, all system could be hosted into cloud platform like Amazon EC2. This solution give more scalability to performance required by the system and could reduce the cost for dedicated server, mantaining an high level of performance especially in case of full load with a lot of connected users.

3.6.3.2 Maintainability The aplication does not provide any specific API, but the whole application code will be documented to well inform future developers of how application works and how it has been developed.

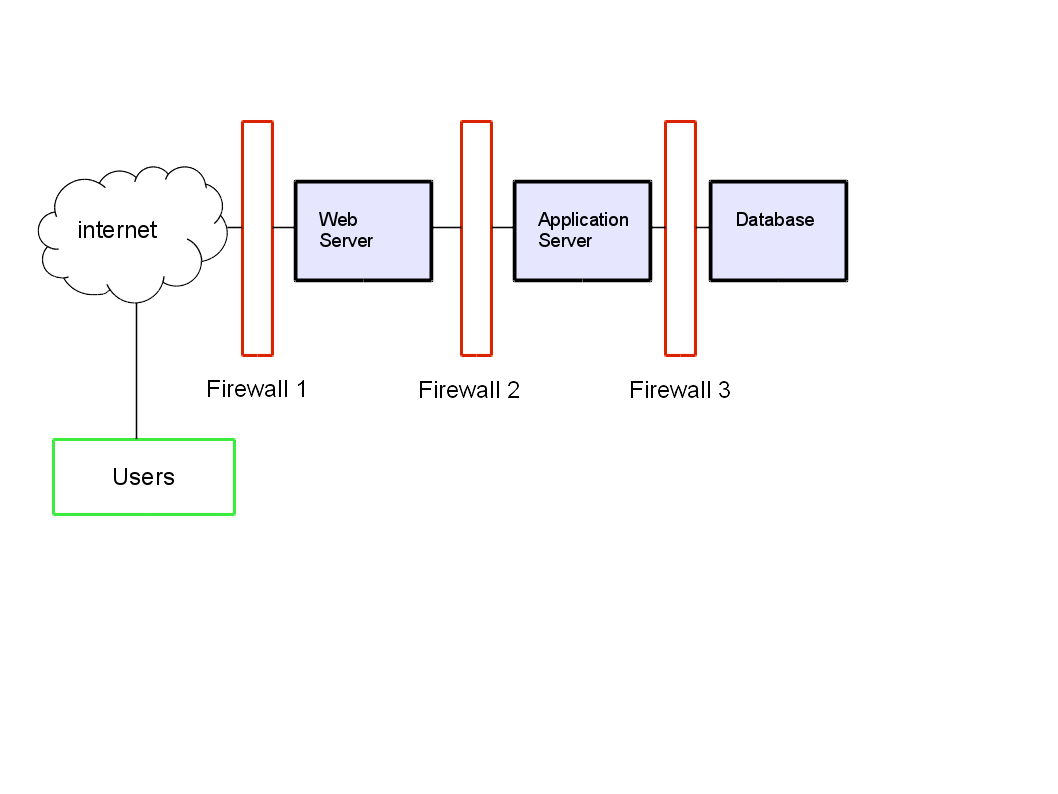
3.6.3.3 Portability The application could be used on any SO which supports JVM and DBMS.

3.6.4 Security

3.6.4.1 External Interface Side MeteoCal application implements a login authentication to protect the information of users. Password of user is saved using hashing mechanism but could not be enough. This system do not actually require anything about the strongness of the password so could be developed a system that require an 8 character password with number, letter and special character. Password also is static, user is not involved in password changing so system will ask to user to change frequently the password for example every 6 month or less. Another useful thing could be some advice to users about how to build a safe password. A more secure system will implement a login system with captcha test to prevent from botnet attack; could also be implemented a multifactor authentication system with a mixing of at least one of this technologies:

* Two-factor authentication whith a code sent by email or sms to the user
* Smart-cart authentication, a card with a chip used to authenticate the user through a smart card reader.
* Dyno tecnologies or one-time password, composed by a card static password list owned by user or a dyamic embedded password generator. The system ask for random code on that card or an entair code generated.
* Biometric: fingerprint, retina scan, voice analysis

3.6.4.2 Application Side On the application side could be implemented a filtering system to all form. Malicious user indeed can fill the form with sql code (sql injection) to have access to information who normally can not have access. Another important secure method is to implement the https connection instead of http to guarantee communication confidentiality and integrity and also mutual authentication. SSL is resistent to man in the middle attack(MITM) but need a server certificate signed by a Certification Authority(CA).

3.6.4.3 Server Side The server side architecture could be implemented dividing strongly the data from application. An idea of possible secure infrastructure is well represented by the following picture. Application Server is separated from database and from the web server. All zone are divided by firewall. Access to this zone is restricted and forbidden to not authorized user.

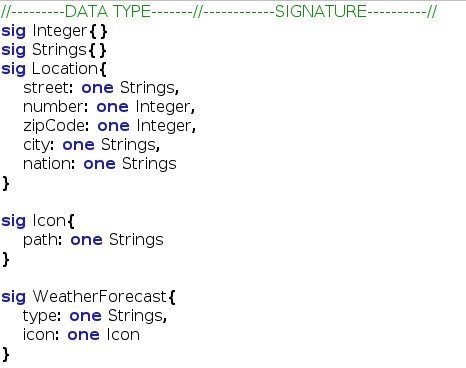
# Appendix

## Alloy

The complete alloy file (.als) could be find on our google code repository. The following alloy model presented is created using the class diagram. We try to divide the code in part dividing signature from fact, assert and predicates. In the last part there are the generated word.

4.1.1 Data Type

This is the definition of data type.



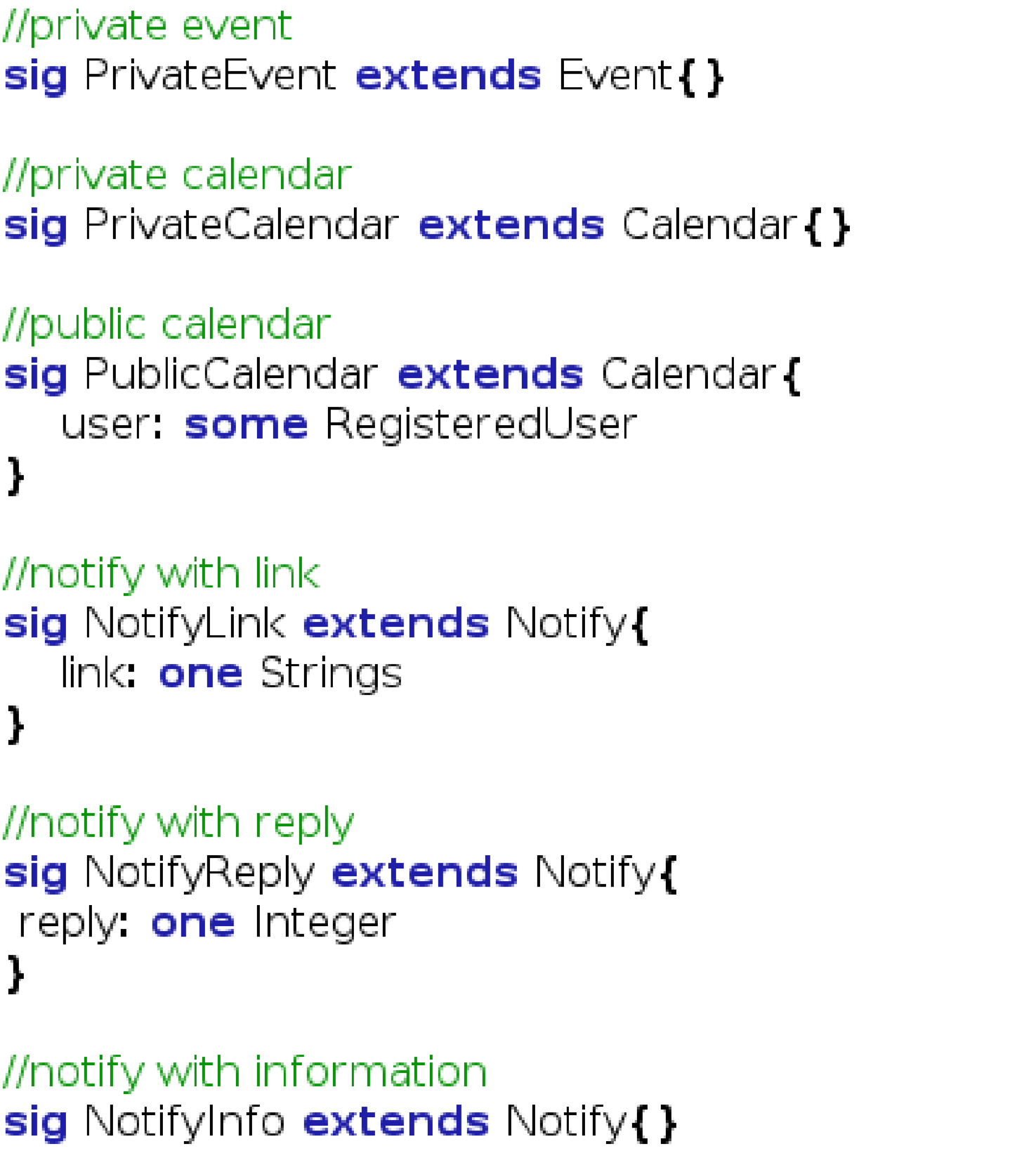
4.1.2 Abstract Entity

This is the definition of abstract entity.



4.1.3 Abstrac Entity Implementation and Signature This is the implementation of some abstract entity and other signature.





4.1.4 Fact

This

is

the

fact

part

that

defines

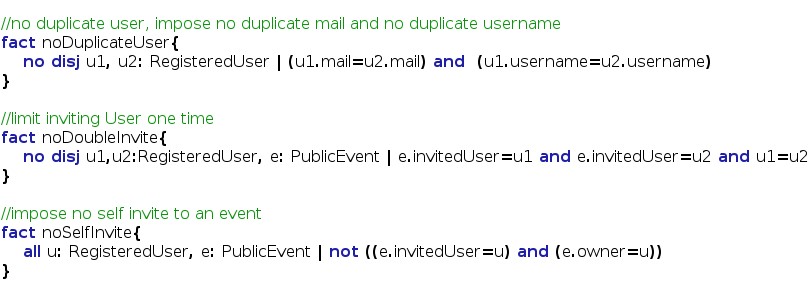
the

costraint

of

the

class.



4.1.5 Assert

In

this

last

code

part

is

presented

the

assert

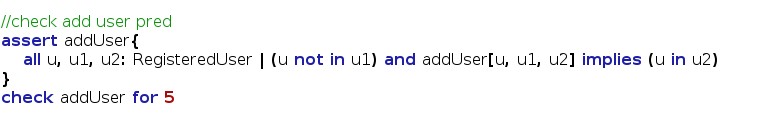
used

to

verify

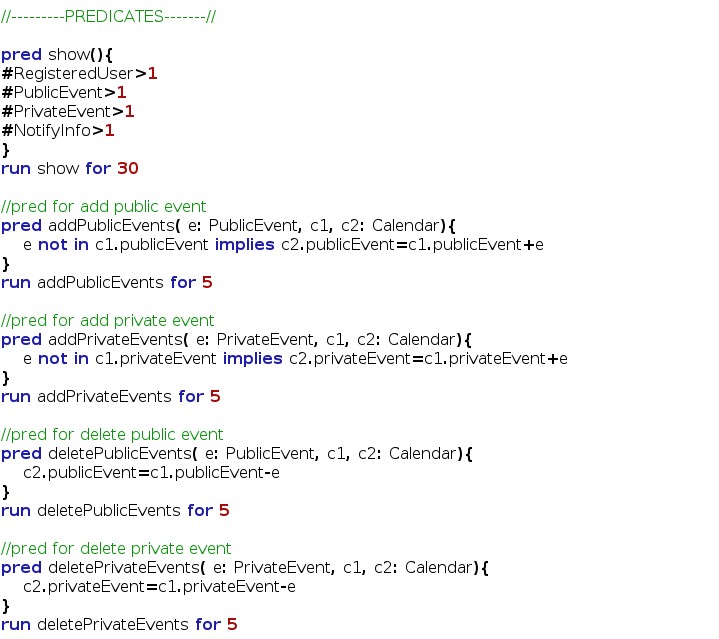
the

model.



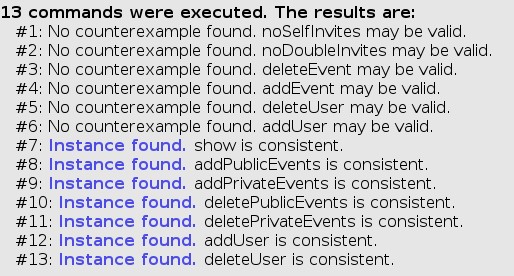
4.1.6 Predicates

This is the predicates used with the previous assert to verify the model.



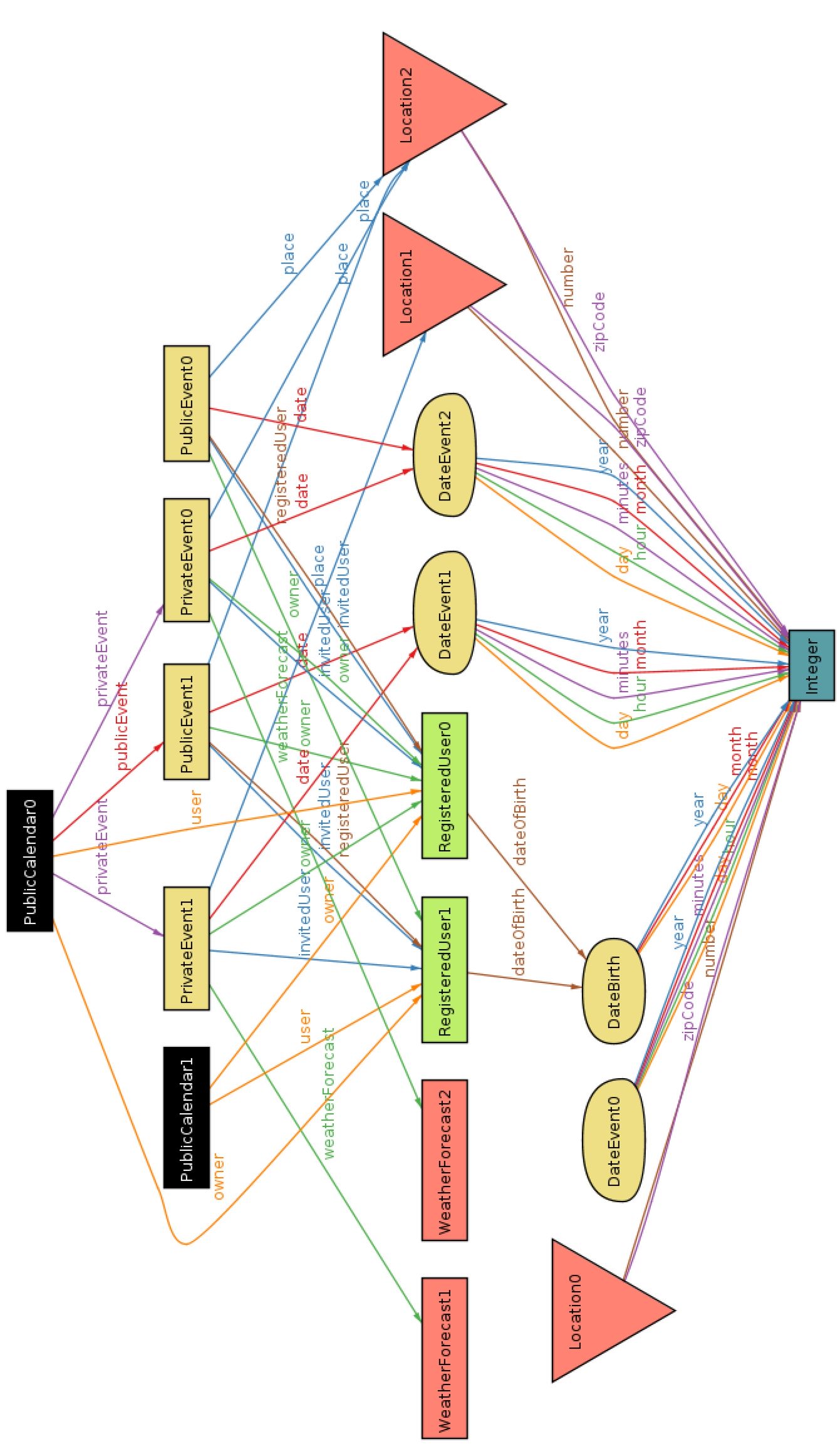
4.1.7 Result

This screenshot of the Alloy Analizer software that shows the consistence of the model in all part.

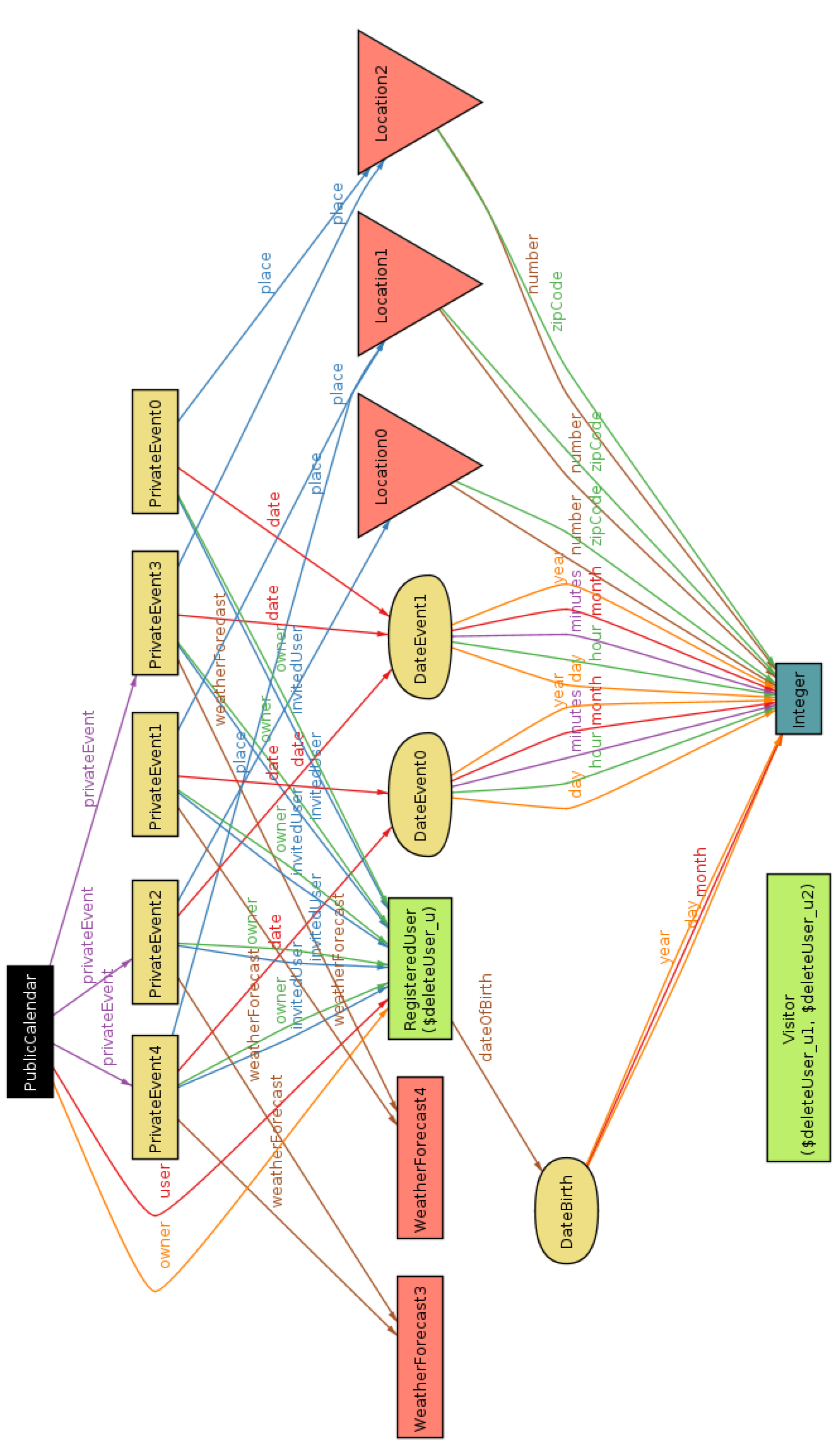


4.1.8 Generated world

Here is presented the generated world using the alloy verification software. First diagram is the predicate show() for 4 case. The second is the predicate show() for 8 case. For more case the model will be more and more complex and difficult to read. The third graph is generated from the deleteUser predicate. We decide to not attach all diagram because they are very complex and would not give other information or help anyone want to read this document.







## Software and tool used

* Lyx [(http://www.lyx.org/)](http://www.lyx.org/): to redact and to format this document.
* Astah Professional [(http://astah.net/editions/professional)](http://astah.net/editions/professional): to create Use Cases Diagrams, Sequence Diagrams, Class Diagrams and State Machine Diagrams
* Alloy Analyzer[(http://alloy.mit.edu/alloy/)](http://alloy.mit.edu/alloy/): to prove the consistency of our model.
* Gimp[(http://www.gimp.org/)](http://www.gimp.org/): to modelling some image.
* Balsamiq Mockups[(http://balsamiq.com/products/mockups/)](http://balsamiq.com/products/mockups/): to create mockups.

## Hours of works

This is the time spent for redact this document:

* Federico Migliavacca: ~37 hours.
* Leonardo Orsello: ~37 hours.

# Revision

This is 2.0 version of RASD that contains update of the document after the entair development of the application. In the follow are listed the difference between the previous version:

## Changed Assumptions

* If there will not be a sunny day, the application will notify the owner this situation.
* If weather forecast is not available at the moment of creation of the event, this one will be created anyway and the weather field will be scheduled to be update the next ~~day untill weather forecast will be available~~ .
* [deleted] User can invites only after creation of the event, not during the creation of the event itself.
* [deleted] If an owner of one or more events delete his/her account all events will be deleted.
* There is not periodical update of weather forecast for event created. ~~Weather will be check only during creation process of the event~~ .
* [deleted] Weather is checked and updated just once three days before the event takes place.
* Notification of bad weather condition will be shown ~~just once~~ .

## Removed goal and functionality

* [G2] Allow user to change public or private nature of his/her calendar
* [G4] Allow user to delete his/her account from database.

## Modified Functional Requirements

* [G1] Allow a visitor to became a registered user and choose the public or private nature of his/her calendar.
  + [D1] Email address used for registration ~~must exist~~ .
* [deleted] [ex G2] Allow user to change public or private nature of his/her calendar
  + [R1] User must be already regitred to success login process.
  + [R2] Changing the global visibility of his/her calendar will affect the visibility of all related events.
* [deleted] [ex G4] Allow user to delete his/her account from database.
  + [R1] User must be already registred and logged in to application.
  + [R2] User must confirm deleting process.
  + [R3] The deleting process is not reversible, all user data will be lost.
* [G3] Allow user to create a new event in the calendar and choose the public or private nature.
  + [R3] User must complete mandatory fields ~~(date, time, location)~~ to complete the event creation process.
  + [deleted] [R4] Event creation process has some discretionary filelds.
  + [deleted] [D1] Location must be an existing city.
* [G6] Allow user to invite/delete other user to a specific event of his/her calendar.
  + [R5] To invite a user, the owner of the event, ~~must know the email of that specific user~~ .
* [G10] After login, application will notify only the creator user three days before an event takes place if the weather is not good.
  + [deleted] [R4] Notification will appear only once.
  + [deleted] [R6] If there will not be a sunny day in the next seven days after the event will take place, application will notify the owner.
  + [deleted] [R7] If weather forecast will not be available in the next seven days after the event will take place, application will notify the owner.
* [G11] After login, application will notify invited user one days before an event takes place if the weather is not good.
  + [R3] Application will notify ~~the owner~~ and invited users of the event only when they perform login.
  + [deleted] [R4] Notification will appear only once.

## Modified Scenarios and Use Cases

Updating some goal and functional requirements has bring some changes also to the scenario. In particular the Scenario 2, 3, 4, 6 and 7. For the use cases the modification are listed in the follow: • [deleted] “Edit Profile” Use Case

* [deleted] “Delete Profile” Use Case
* ~~User sees “event has been modifed" alert~~. This is substituted with: “User sees notification for modified event with another invitation” Use Case
* ~~User sees "other details" of an event~~. This is substituted with: “User sees details of an event” Use Case.

## Modified Diagrams

The diagrams of Use Case quoted above have been modified. An update has also been applied to the State Machine Diagram. The Class Diagram has not been modified since there aren’t structural error but some class were already natively implemented in Java. Some little modification could be necessary for Mock-up but screenshot of real application are available in the user manual.