MeteoCal

Requirements and Specification Analysis Document

Software Engineering 2

A.A. 2014/2015

Andrea Grazioso
Fabrizio Ferrai
Germano Gabbianelli

November 16, 2014

Part I
Introduction

This part is an introduction to this document, the context, the objectives and contraints of the platform that's going to be developed.

1 Meaning of this Document

This document presents the functionality and the high-level requirements of MeteoCal, the weather based online calendar platform. MeteoCal is designed to be used by a general public and offers the possibility to create and modify events on a fully-featured web platform with the possibility to monitor the forecast for the scheduled events.

2 Context of the product

2.1 Objectives

The name of the software that's going to be developed is CalCARE. CalCARE must:

1. allow *visitors (non-registered users)* to register.
2. allow *registered users* to create and modify events –public or private– containing information about when and where they will take place, whether the event will be indoor or outdoor, with the possibility to invite any number of registered users of the service.
3. allow *registered users* to check weather forecast information (if available) and be notified in case of bad weather conditions for outdoor events.
4. allow *registered users* to make their calendar public; it means that is visible to all the other registered users, who will see all the time slots in which the *user* is busy, without seeing the details of the corresponding events, unless they have been defined as public.
5. allow *registered users* to import and export their calendar and be aware –during event creation– of conflicts with existing events.

2.2 Contraints

The first version of CalCARE will present the following contraints:

weather the weather management is not handled by the system, as an external API will be called to provide the forecasts when necessary

sharing the applet will provide links which will be emailed to other registered users in order to invite them to join the event. The invitations will be accessible also in a view of the application.

- **monitored access** there will be an access control mechanism: only the invited users will be able to access to the event invitation.
- **e-mail** the web application will automatically send the email invitations for the corresponding event.

social network the system does not provide any form of social interaction between users –apart from the invitation system– like managing text chat between users or managing user's friend list.

privilege control the system will not implement any form of privilege for the user. On the other hand, non registered users will not be able to use the platform.

2.3 Legacy

As of this writing, MeteoCal does not have an applet or platform, so there is no need to integrate existing databases or different architectures/platforms.

3 Definitions, Acronyms, Abbreviations

3.1 Acronyms

MC from now, MeteoCal, for the sake of brevity.

CC CalCARE, as above.

3.2 Definitions

Visitor non registered user.

User registered user.

Events user-made, generic activity related to a specific time slot in the calendar, with starting/ending time or all-day-long, characterized also by location and forecast (if available) for that spatiotemporal coordinates.

Calendar a chart or a series of pages showing the days, weeks, and months of a particular year, giving also the information about event for a particular day (if any).

4 Reference

- IEEE 830/1998 RASD template from University of St Andrews IEEE requirements book online resources.
- W3C HTML validator http://validator.w3.org/
- W3C CSS validator http://jigsaw.w3.org/css-validator/
- Alloy resources http://alloy.mit.edu/alloy/

5 View of the Document

This document is organized as follow:

- in *Part II* will be described functionality, contraints, limits, assumptions, dependencies and rules of the actors who characterize the whole system.
- in *Part III* will be analized functional requirements and non functional requirements of the system; in particular there will be presented some possible scenarios, use cases and specifications of the model.
- in *Part IV* the model specifications will be formalized in Alloy and will be discussed the coherence of the resulting model.

Part II
General Description

This part describes the context in which CC will operate, i.e.: types of users who interact with the system, resources available for the application, assumptions made to develop the software and hardware/software requirement to use the application.

6 Product Perspective

The product is a self-sufficient platform, except for the weather forecast system provided by some external API. CC does not provide any public API to access private data, nor fisical interfaces for user interactions.

7 Actors of the system

7.1 Analytical Enumeration

A list of types of users who can possibly interact with the system (and for each type are reported all possible functions) follow:

- **User (registered)**
  - Has a personal private area in which is located the calendar.
  - From the calendar can create a new event.
  - Creating a new event she have to insert date, location and the type (which can be public or private) of the event.
  - While creating the event it is possible to send invites to friends that may join the event.
  - Can send invites for each event from the event page (if is the owner).
  - Can modify a previously created event.
  - Can import and export her own calendar.
  - Can modify her personal profile.
  - Can recover a lost password.
- **Visitor**
  - Can register herself to the service.

7.2 Use Case Diagram

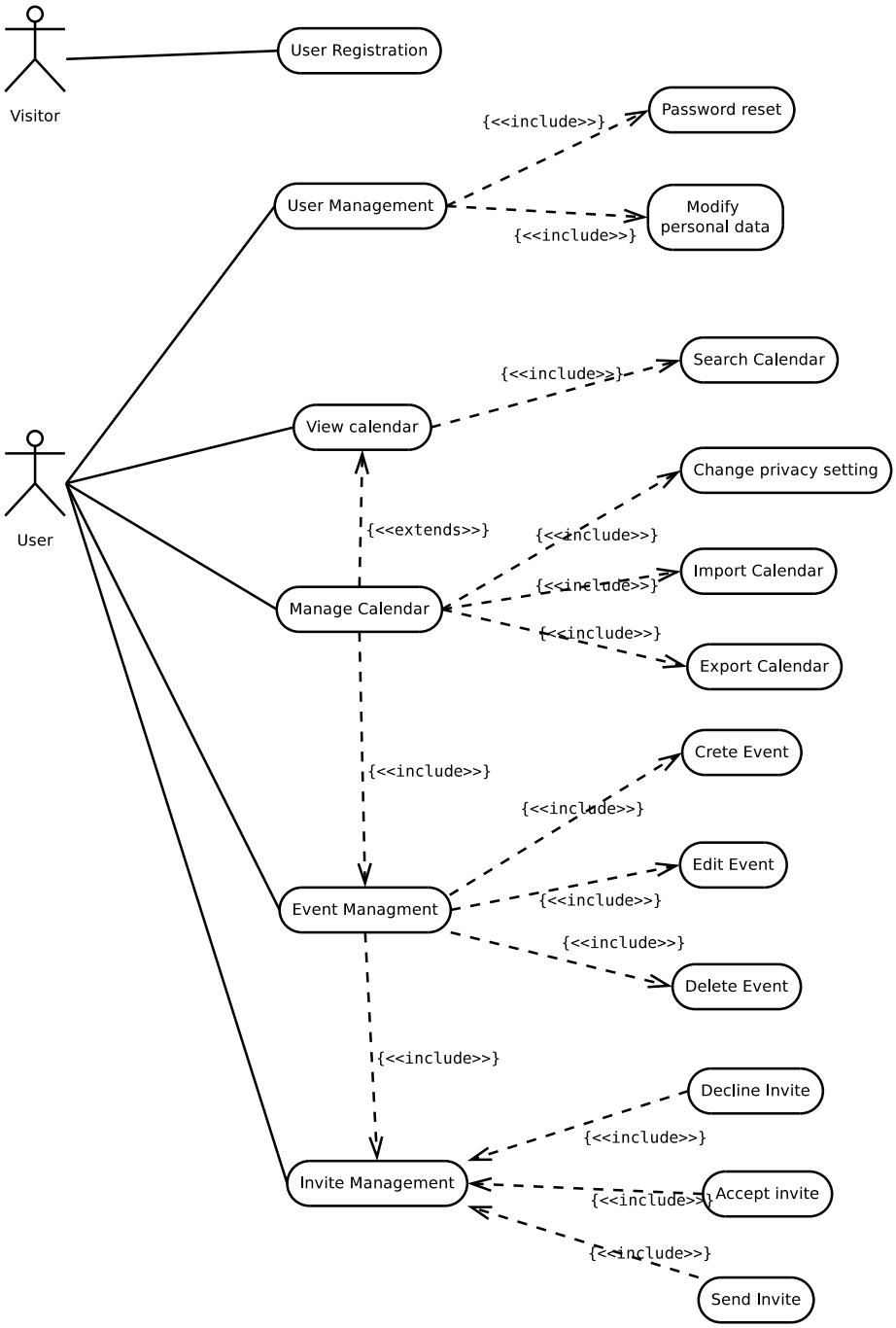

Figure 1: Diagramma use case del sistema

8 Dependencies and system requirements

8.1 Operative System

CalCARE can be executed on every system that supports the Java Virtual Machine and respects the minimum requirements described shortly. The development platform is based on Ubuntu 14.04 (Trusty Tahr), which is recommended for production use.

8.2 Software Stack

8.2.1 Server

In addition to the Java Virtual Machine, CC also depends on the following software components, that are assumed as installed. The term «reference version» is inteded as the version used to actively develop and test the software, so recommended for production.

Table 1: Required software stack

Component Name	Function	Reference version
MySQL	Database Management System	5.5
GlassFish	JEE server & framework	4.1
JRE	Java runtime environment	1.8

8.2.2 Client

The users of the system have to connect to Internet and have a modern browser installed. More non functional requirement with details on supported configuration in the appropriate section.

8.3 Communication interfaces

The communication will use HTTP(S) and TCP/IP, with the following requirements:

Table 2: Tcp/Ip Port used

Port	Protocol	Firewall State
80/tcp	HTTP	allow from everywhere
443/tcp	HTTPS	allow from everywhere
3306/tcp	MySQL	deny from everywhere, allow for given server[1]

8.4 System requirements

Considering the system load expected by the customer and the minimum requirements fixed by the software components of the platform, the minimum system requirements are:

Table 3: Minimum system requirements (server)

Component	Minimum
RAM (primary memory)	2GB
HDD (secondary memory)	4GB
CPU Core	2
CPU Class	Intel Core Duo 2, Xeon or equivalent

9 Assumptions

9.1 System Setup

The hardware, software and network configuration is managed by the system administrator of MeteoCal following the specs and requirements present in this part of the document. It is not a task of the developers platform in object.

9.2 Weather forecast

The weather forecast is handled by an external entity. CC will receive from this external entity the weather information for a given place/time when requested, through a public API, which is detailed in the following sequence diagram:

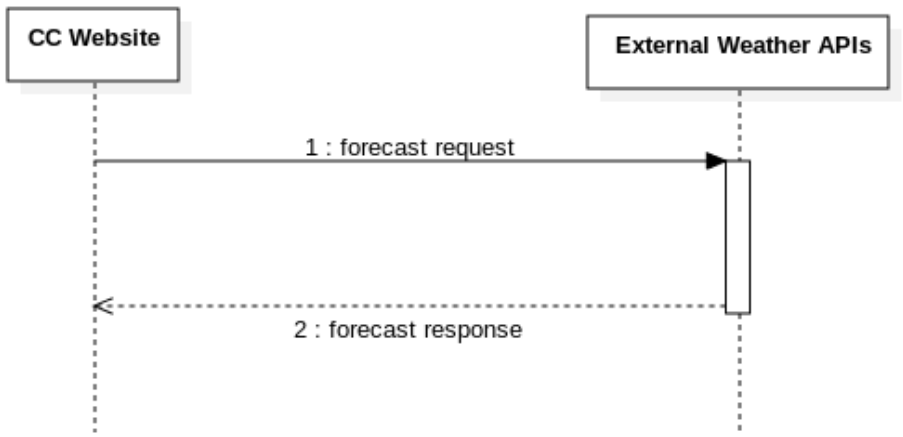

Figure 2: Forecast API process

In case of having to convert the location input to a friendly format for the wheather APIs, it could be necessary to call another external service in order to handle this task. It is however an implementation detail, and will be discussed in the next design documents.

Part III
Specific Requirements

10 Functional Requirements

Functional requirements of the system are divided in 3 macro-areas:

1. Event Management
   (a) add a new event
   (b) modify an existing event
   (c) delete an existing event
   (d) make an event «Public» or «Private»
   (e) specify details for an event: description, time, location
   (f) invite other users of the service to events
   (g) accept other users invitations for their own events

   public: inserted by a user and visible to all the other users
   private: created by a user and visible only in the personal area

2. Users profile management
   (a) registration
   (b) login
   (c) logout
   (d) password reset
   (e) modify personal data

3. Calendar Management
   (a) modify privacy setting of the calendar (it can be public or private)
   (b) import calendar
   (c) export calendar

   A public calendar show to all the other users of the service that a user is busy during a private event - in case of public events all the details are available

10.1 Scenario

10.1.1 Creating and sharing an event

Description	Describe how an user is able to create and share an event with other users
Objective	Allow users to create and share an event with other users
Assumption	–

Alice is planning a pic-nic with other five friends at the «Bosco in città» pic-nic area on Wen Sept 5. Alice opens her browser, and as logins with her mail account and password into her CalCARE account her browser is redirected to her own personal area.
From there she is able to view her calendar which indicates Sept 5 to be a sunny day. So clicking on the curresponding day slot, a button will make possible to create an event. In the event creation modal popup some basic information about the event is required to be inserted:

- Name of the event: Pic-Nic with friends
- Description (optional): Bring guitars and percussions, we're gonna sing all day!
- Date: September 5, 2014
- Place : «Bosco in citta»
- Time: All Day Event
- List of Partecipants: Mario Red, John Timber, Frank Matano, Jonny Muciaccia, Antani.

By hitting the button «create event» the popup will close and now she can observe in her calendar the event just created. In the meantime Alice's friends will receive the invite and they have to decide if accept or decline. If they accept the event appears also on their calendar.
In the slot corresponding to the selected day in the calendar Alice is able to see the forecast for that day (if available) and more precisely for the exact time of the event.

10.1.2 A visitor registers into the system

Description	Describe how a visitor can register himself to the CalCARE system
Objective	Allow a visitor to register in the system
Assumptions	–

Luke is planning to make a party in his house; a friend adviced him to try using the CalCARE platform because all of his friends are already on board. But, in order to create an event on CalCARE Luke has to be registered, so he wakes his pc and points his browser to the CalCARE website, and then hits the 'Register' button.
He's then redirected to a page with a form to be filled with personal information: email, password, name and surname. By clicking on the «Submit» button the system will receive the request, adds a pending user in the system and an email is sent to his email account (in order to check that it actually exists). The email will contain a sincere welcome message from the CalCARE team and a confirmation link. Once Luke accesses to his mailbox he reads the mail, clicks on the link, and redirected to a successful registration page on the CalCARE website. From here he can access his personal area and he can start planning his marvelous house party.

10.1.3 A user decides to modify an event

Description	Describe how an user can edit an already created event
Objective	Allow users to modify previously created events.
Assumptions	–

It's Wednesday, and Carl has just scheduled a meeting with some of his colleagues, to talk about their next Software Engineering project for university, for this weekend, on Sunday, from 3:00 PM to 6:00 PM. Of course he has created an event on the CalCARE platform, and, since they are users of the service too, he invited them to the event.
Then, on Thursday, he remembers that his mom has her birthday on Sunday. Disaster! He must move the meeting with his colleagues. And there the CalCARE system comes in help: he logs in to the system, and once in the calendar view, he selects the cell for next Sunday, and then the «Edit» button for the meeting event.
A modal popup comes out, and he can edit all the information he entered about the event: title, date, time, indoor flag, description, invited people. Since the event is indoor the forecast for the new day is not fundamental. After agreeing with his colleagues on a new time and date he edits the corresponding fields and hits the «Save» button. The event then is moved to the new location on the calendar page (like Saturday from 5:00pm to 8:00pm), and the other users invited to the event are notified of the change.

10.1.4 Bad weather for an already scheduled outdoor event

Description	Describe how in case of bad weather the system will warn the event author three day before, allowing him to move it
Objective	Allow users to reschedule an event in case of bad weather
Assumptions	–
Alice created on day August 29 a pic-nic event with 5 friends at «Bosco in città» pic-nic area, scheduled for Mer, 5, September. On 2 September the weather forecast predicts rain for Sept 5; at this point the system will generate a notification (which is also sent to her through an email) for Alice suggesting her to move the scheduled day from Sept 5 to Sept 8, which is the next sunny day, according to weather forecast. Then –on the next login into the system, or while checking her mailbox – she reads the notification and accepts the suggestion to move the event to Sept 8. At this point all the partecipants will be notified about the change, via a notification into the system and an email.	

10.2 Use cases

10.2.1 Create Event

Name	CreateEvent
Description	An user creates a new event in her calendar
Actors	User
Pre conditions	The user is authenticated
Post conditions	A new event is created and correctly added to the user's calendar. The invited users received an email notification as required.
Event Flow	

1. The user selects the "Create new event" function
2. The server shows a page containing a form to create the event, that includes the following fields:
   (a) Event name
   (b) Description (optional)
   (c) Location
   (d) Outdoor event (checkbox, default false)
   (e) Start date and time
   (f) End date and time
   (g) Mark as private (checkbox, default is True)
3. The page also contains a widget that allows to invite other users
4. The user fills the form, with all the required informations, and selects some users to invite to his event.
5. The user clicks the "Create event" button and submits the data.
6. The server validates the data, creates the new event and sends the required email invitations.

Exceptions

- If the form data submitted by the user is incorrect (e.g. the date is not a valid date) or some required fields are missing, the form is displayed again, notifying the user about the errors, and the event is not created until the form is submitted with valid data.
- If the event overlaps with an other event in the calendar of the user, an error will be shown and the user will be asked to change the date of the event.

10.2.2 Edit Event

Name	EditEvent
Description	An user edits an event already present in her calendar
Actors	User
Pre conditions	The user is authenticated and is the *organizer* of the event
Post conditions	The event is correctly modified according to the user preferences.
Event Flow	

1. The user selects an event, of which she is an organizer, from his calendar
2. The server shows the page of the event
3. The user selects the "Edit event" function
4. The server shows a page containing a form to edit the event, that includes the following fields, prefilled with the current values:
   (a) Event name
   (b) Description (optional)
   (c) Location

(d) Outdoor event (checkbox)
(e) Start date and time
(f) End date and time
(g) Mark as private (checkbox)

5. The users edits the form changing the desired fields.
6. The user clicks the "Save event" button and submits the data.
7. The server validates the data and saves the changes to the event.

Exceptions

- If the form data submitted by the user is incorrect (e.g. the date is not a valid date) or some required fields are missing, the form is displayed again, notifying the user about the errors, and the event is not edited until the form is submitted with valid data.
- If the event overlaps with an other event in the calendar of the user, an error will be shown and the user will be asked to change the date of the event.

10.2.3 Delete Event

Name	DeleteEvent
Description	An user deletes an event already present in her calendar
Actors	User
Pre conditions	The user is authenticated and is the *organizer* of the event
Post conditions	All the invited users are removed from the event, and thus all the pending invitations are invalidated. The event is permanently deleted from the system.

Event Flow

1. The user selects an event, of which she is an organizer, from his calendar
2. The server shows the page of the event
3. The user selects the "Delete event" function
4. The server shows a confirmation dialog, alerting the user that the action is permanent and non reversible.
5. The users confirms the deletion of the event.
6. The server removes all invited users from the event, invalidating all pending invitations, and then removed permanently the event from the system.

Exceptions

None

10.2.4 Search Calendar

Name	SearchCalendar
Description	An user searches the Calendar of an other user
Actors	User
Pre conditions	The user is authenticated
Post conditions	The user is viewing the desired Calendar

Event Flow

1. The user selects the «Search Calendar» function.
2. The server shows a form with a single input named «Email».
3. The user enters the email of the user of whom she desires to view the calendar.
4. The server finds the User associated with the submitted email address and then shows the page representing her Calendar.

Exceptions

- If the form data submitted by the user is incorrect (e.g. the email address is not valid, or does not correspond to any registered user) or some required fields are missing, the form is displayed again, notifying the user about the errors, and the desired calendar is not shown until the form is submitted with valid data.
- If the desired Calendar is not public it will not be shown, and an error will be displayed to the user.

10.2.5 Import Calendar

Name	ImportCalendar
Description	An user imports her Calendar
Actors	User
Pre conditions	The user is authenticated
Post conditions	The user successfully imported a copy of his calendar.
Event Flow	

1. The user selects the «Calendar Settings» function.
2. The server shows the Calendar Settings page.
3. The user selects the «Import Calendar» function.
4. The server converts the uploaded calendar into a valid internal representation, creating all the required events.

Exceptions
If the uploaded calendar contains events that overlap with some existing user events, the import may fail.

10.2.6 Export Calendar

Name	ExportCalendar
Description	An user exports her Calendar
Actors	User
Pre conditions	The user is authenticated
Post conditions	The user successfully downloaded a copy of his calendar.

Event Flow

1. The user selects the «Calendar Settings» function.
2. The server shows the Calendar Settings page.
3. The user selects the «Export Calendar» function.
4. The server converts the internal representation of the user's Calendar into an usable format and lets the user download it

Exceptions

None

10.2.7 Change privacy settings of Calendar

Name	CalendarPrivacy
Description	An user changes the privacy settings of her calendar
Actors	User
Pre conditions	The user is authenticated
Post conditions	The user's calendar is set as either public or private. Setting the calendar as public will let other users see the date and time of the private events, but not their details.

Event Flow

1. The user selects the «Calendar Settings» function.
2. The server shows the Calendar Settings page.
3. The user checks (or unchecks) the «Make calendar public» option.
4. The user clicks the «Save settings» button and submits the data.
5. The server sets the privacy of the calendar according to the user preferences.

Exceptions

None

10.2.8 Respond to Event Invitation

Name	RespondToInvitation
Description	An user accepts or declines an invitation to an event
Actors	User
Pre conditions	The user received an invitation to an event
Post conditions	The status of the invitation is correctly changed to accepted or declined according to the user choice.

Event Flow

1. The user opens the invitation email.
2. The user clicks on the provided link.
3. The server shows a page containing «Accept» and «Decline» options.
4. The selects one of the two options.
5. The server updates the invitation status and sets it to Accepted or Declined.

Exceptions

The event could have been deleted before the user actually reads the email notification. In that case the system will show an error page showing that the event does not exist anymore.

10.2.9 User registration

Name	Registration
Description	A visitor registers to the system an becomes an User
Actors	Visitor
Pre conditions	None
Post conditions	A new User is registered on the system.
Event Flow	

1. The visitor opens the MeteoCal website.
2. The visitor selects the «Register» function.
3. The server shows a page containing a registration form, that includes the following required fields:
   (a) Given Name
   (b) Family Name
   (c) Email
   (d) Password
4. The visitor fills the form, with all the required informations.
5. The visitor clicks the «Register» button and submits the data.
6. The server validates the data and creates the new User
7. The server sends a confirmation email to the new User, to verify that the provided email actually exists.
8. The visitor opens her email client and clicks on the link in the confirmation mail.
9. The server sets the User as active, allowing her to log in.
10. The server shows a page informing the Visitor that the registration was successful.

11. The Visitor is now an User and can log in the system.

Exceptions
If the form data submitted by the user is not valid (e.g. the email is already in the system, or the password is too short) or some required fields are missing, the form is displayed again, notifying the user about the errors, and the new User is not created until the form is submitted with valid data.

10.2.10 Reset Password

Name	ResetPassword
Description	An User lost her password and wants to reset it
Actors	User
Pre conditions	The User is not authenticated
Post conditions	The User set a new password

Event Flow

1. The visitor opens the MeteoCal website.
2. The visitor selects the «Reset Password» function.
3. The server shows a page containing a form to ask the User the email address corresponding to her account.
4. The User inserts the email address with which she registered.
5. The server generate a reset link for the User.
6. The server sends the reset link to the given email address.
7. The User opens her email client and clicks on the link in the reset email.
8. The server shows a page with a form to reset the User's password.
9. The User enters the new password and submits the form.
10. The system sets the new password for the User.

Exceptions

If the form data submitted by the user is not valid (i.e. the email does not correspond to any user) or some required fields are missing, the form is displayed again, notifying the user about the errors.

10.2.11 Login

10.3 Sequence Diagrams

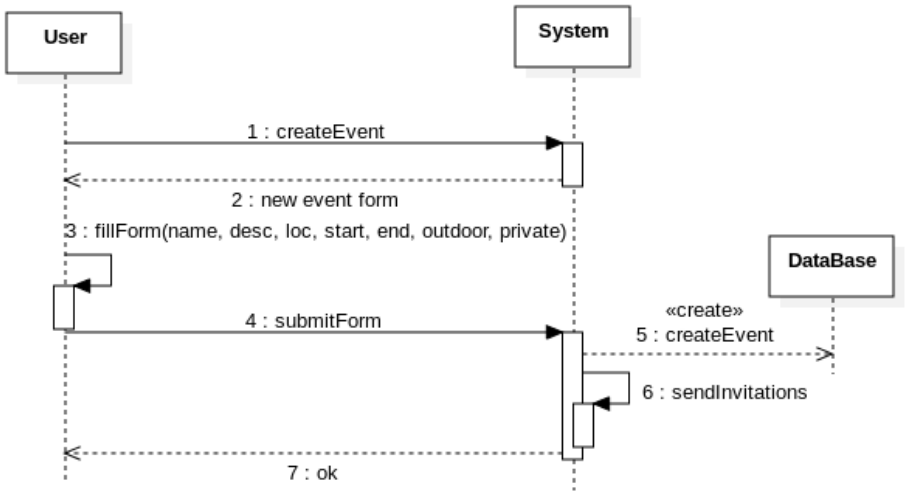

Figure 3: CreateEvent

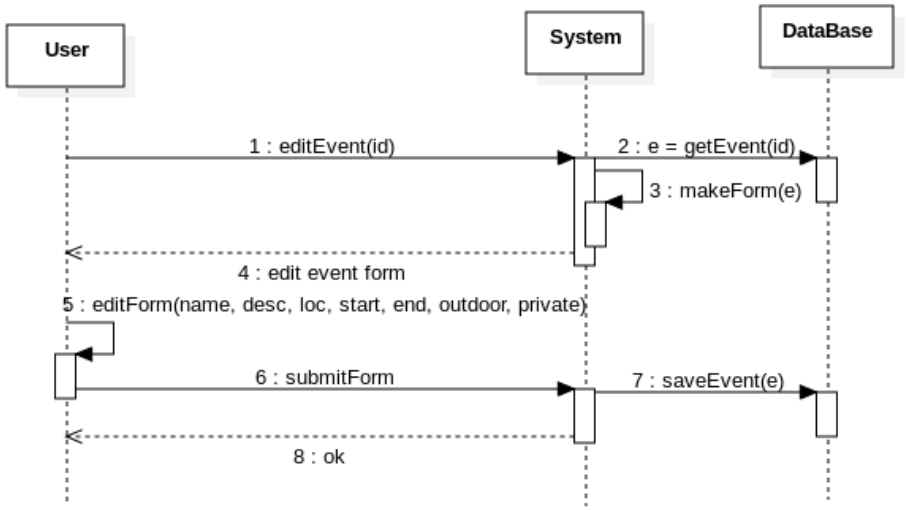

Figure 4: Edit Event

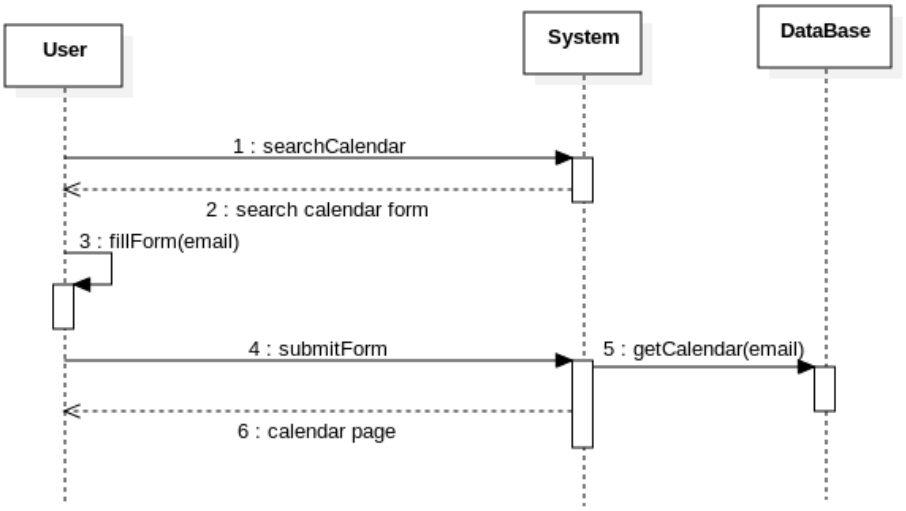

Figure 5: Search Calendar

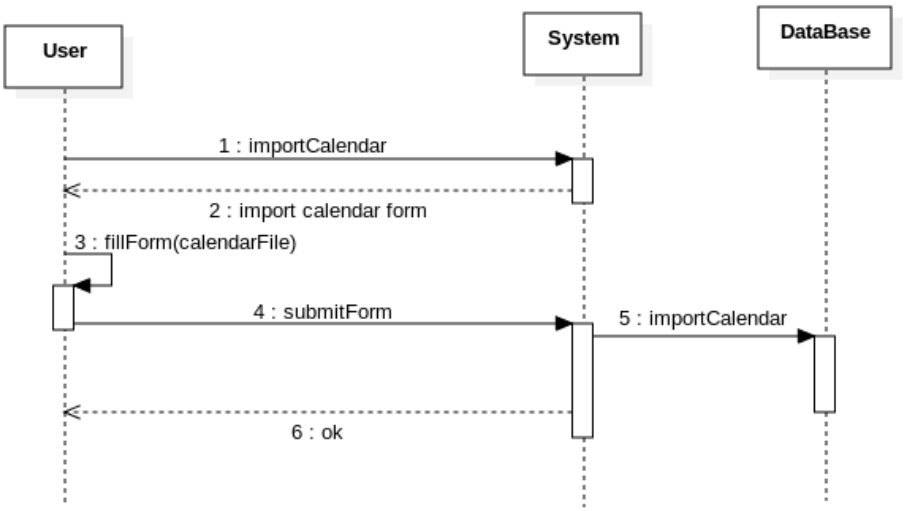

Figure 6: Import Calendar

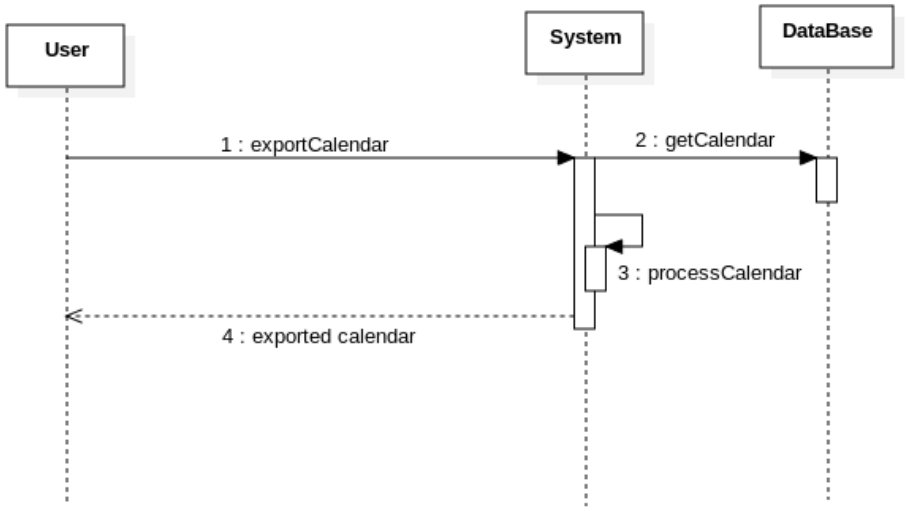

Figure 7: Export Calendar

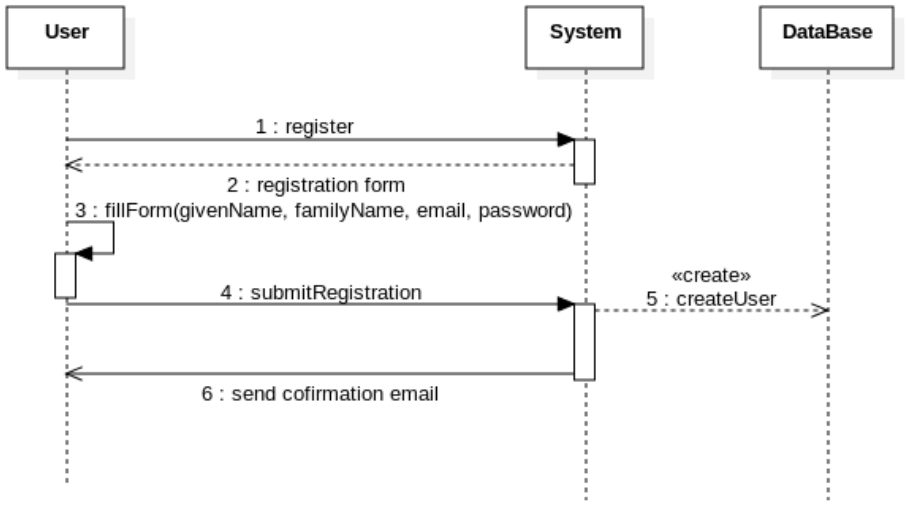

Figure 8: Registration

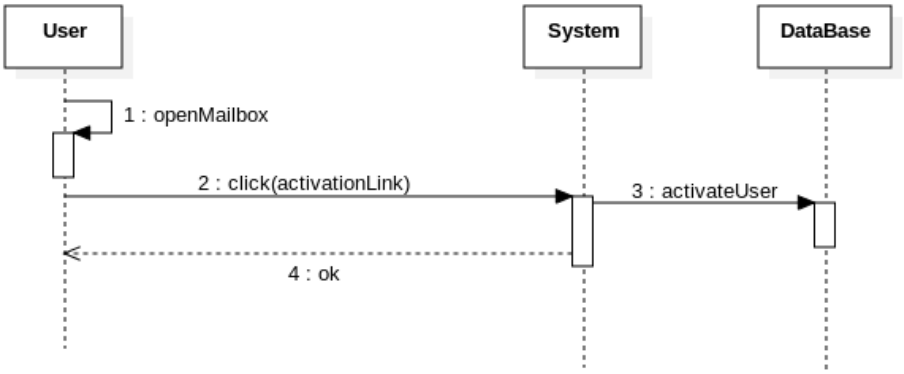

Figure 9: Email Confirmation

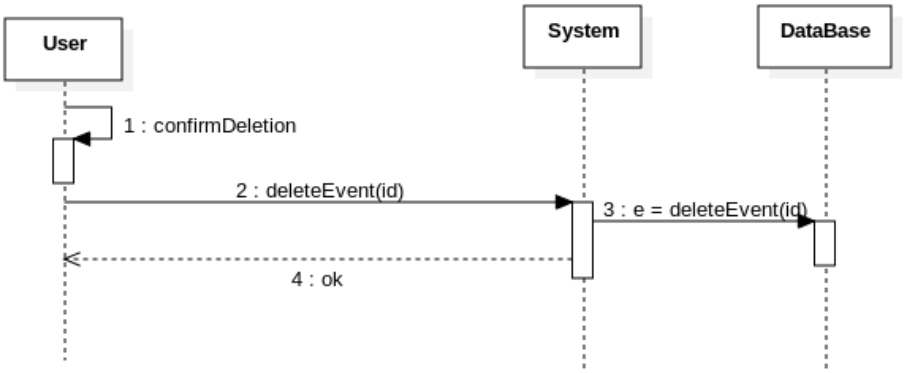

Figure 10: Delete Event

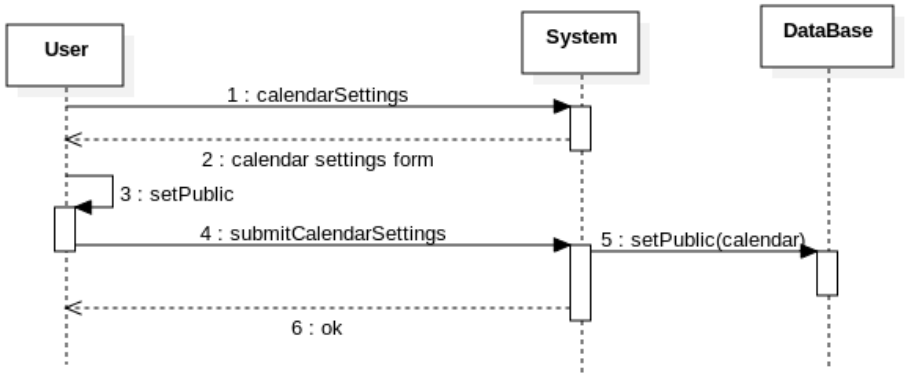

Figure 11: Calendar Privacy Settings

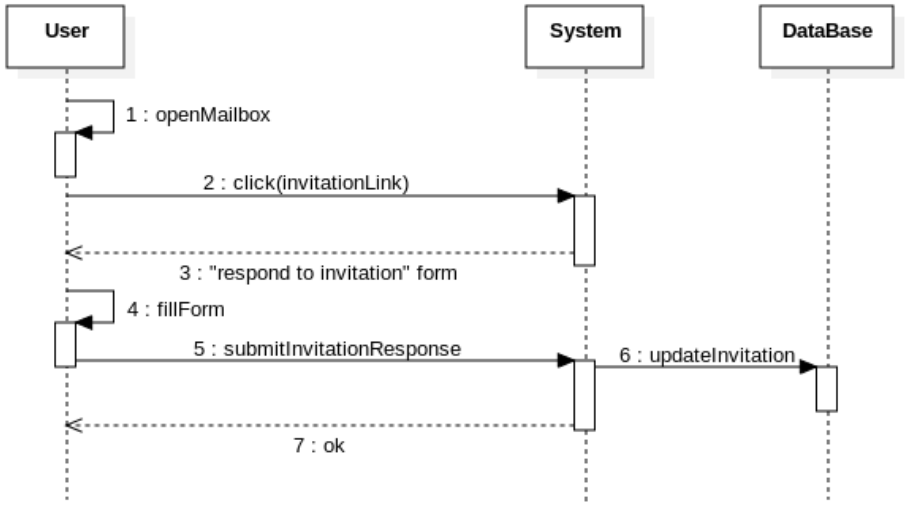

Figure 12: Respond to Event Invitation

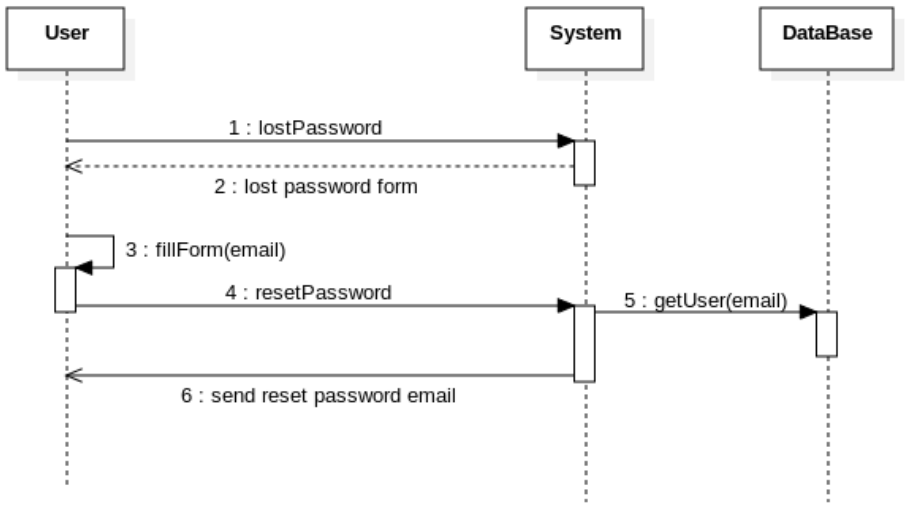

Figure 13: Reset Password (first step)

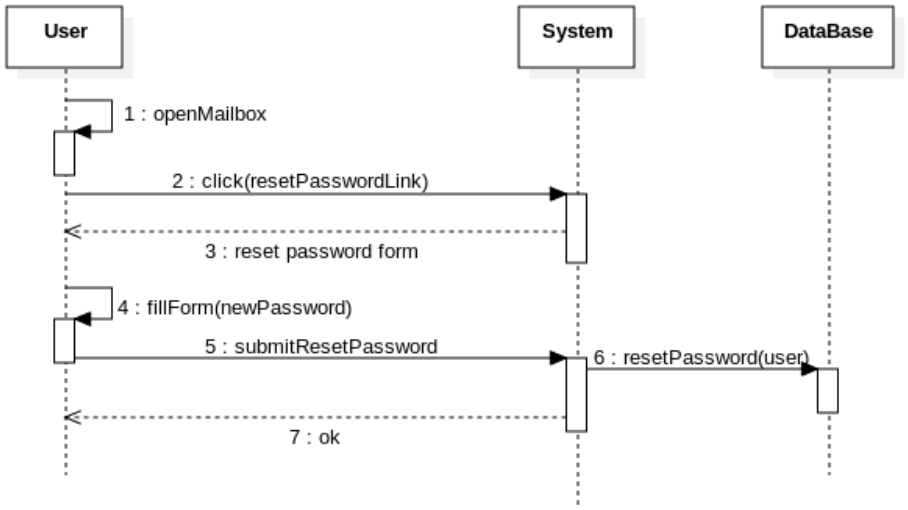

Figure 14: Reset Password (second step)

11 Non functional requirements

In the following paragraph will be explained the not functional requirements of the system. Expressions such as «is desiderable» or «is convenient» mean requirements that are not necessary, and which are not essential for the client, or for which missing implementation (for every possible reason) will not be considered as a missing functionality and could be implemented in future releases of the software.

11.1 Infrastructure

11.1.1 Platform

The system will be developed on the JEE platform, in particular using EJB for the application logic.

11.1.2 Reliability

The system will guarantee that all the events created and modified by users will be correctly inserted into the system's database and eventually notified to friends. In particular, is absolutely fundamental for the system to keep track of the updated weather forecast from the API, to promptly notify weather changes (in case of transitions from good to bad weather) to the user, such as every time there will be bad forecast weather for an outdoor event three days before no notification will be missed.

11.1.3 Availability

The system must not reduce its availability below the one granted by the server class on which it will be installed.

In particular it must be designed for a 365 days/year, 24h/day availability, and provide a fast reboot method after any version update, or in case of server reboot.

11.1.4 Security

1. Passwords will be stored only after having hashed them criptographycally (ex. >10 iterations of SHA512).
2. The system must not allow any form of intentional backdoor.
3. As far as possible developers should use any known tecnique in their knowledge to prevent any form of malicious attack, such as:
   (a) mitigate SQL-injections
   (b) avoid XSRF attack

(c) escape web pages content, sanitize any form of user input, and prevent XSS attacks in any possible way.

(d) discourage password brute-forcing through exponential timer back-off between a failed authentication attempt and the next possible attempt.

11.1.5 Privacy

With considerations of sensible data stored in the database (Name, Surname), the system will provide all necessary security method to make data accessible only to allowed agents.

11.1.6 Integrity

Check on integrity property on Database is demanded to Database Managment System.

11.2 Interface Requirement

11.2.1 Interactions

User interactions will be through a web application. Any form of off-line client or desktop/mobile client in not planned.

11.2.2 Validity of Markup

All web pages that are part of the interface, public and private have to be in valid HTML5. CSS3 code used to present the contents has to be valid. The validation will be made throught instruments in the apposite section Reference of this document.

11.2.3 Accessibility

The web interface will have to

1. present a semantic consistently markup.
2. present a font dimention and a contrast between backgound and text sufficient to allow the reading of the contents without getting eye tired.

11.2.4 Supported Browser

The web interface have to be usable al least on the latest stable version on the following browser:

Table 4: Instance Browser

Browser	Testing versiomn
Google Chrome/Chromium	31
Firefox/Iceweasel	17esr, 24esr, 25
Opera	17
Safari	7.0
Internet Explorer	(8, 9), 10, 11

Considering the very restricted part of users that still user bloated browser (for instance IE<10) and the implementative cost to support CalCARE on such browsers, is not planned any form of official support for them. Users of bloated browser will see a popup message suggesting them to update their browser and explaining why the site does not allow them to navigate.

11.2.5 Mobile system support

The web interface have to remain still usable for mobile system, in particular have to provide users necessary information within a brief space, without obliging him to scroll for long distances in the page. A *responsive* interface is suggested, but not strictly required.

11.2.6 Ease of use

Users have to be able to use the platform without any need of additional knowledge. In particular, user interface will provide:

1. for varius actions there have to be easy to find mechanism of interactions or button with text and suggestion in English
2. Reasonably understandable error messages, that should refer text understandable by everyone and not strictly related to hardware/software errors (such as «Cannot create event now, please try later» insted of «Connection to DB failed»)

Part IV
Appendix

12 Model in natural language

12.1 Contraints modelled in Alloy

- There are 0 or more users
- There are not two users with the same email
- Every event must have a name, starting date, ending date, starting time, ending time, location
- An event can have an unlimited number of invited user
- An event can have a description
- The creator of an event is also the owner
- Every event has one and only one owner
- Every event has a unique identifier
- Every event has a weather forecast associated to its location and time interval if it's outdoor
- If an invitation is accepted by a user the event is also on his calendar
- An event may be modified or deleted only by its owner
- An invitation sent for an event can be only accepted, rejected or pending
- There not exist any event without an owner
- Every user can have one and only one calendar
- Users cannot create events referring to a time interval in the past

Some constraints are not easily translatable in Alloy:

- All visitors must register in order to use the service
- If an event is defined as public then all information about it can be seen by all users
- If the calendar is defined as public then all users can see when the owner is busy
- Users who receive an invitation for a private event can see all the details about it

13 Alloy Model

13.1 Static model

```
module CalCARE

// Base types definiton

open util/boolean
open util/ordering[Date]
open util/ordering[Time]

sig Name {}
sig Surname {}
sig Date {}
sig Hash {}
sig Time {}
sig Status {}
sig Location {}
sig ID {}

enum Meteo{
sunny,
rainy,
cloudy,
snowy,
na
}

enum InvitationStatus{
pending,
accepted,
rejected
}

// Definition of entity and implicit constraints of the model

// 1. Events and Calendar

sig Event {
name: one String,
    beginDate: one Date,
    endDate: one Date,
    beginTime: one Time,
    endTime: one Time,
    location: one Location,
owner: one User,
    invited: set User,
meteo: one Meteo,
identifier: one ID,
```

```
outdoor: one Bool,
public: one Bool,
}{
    lt[beginTime, endTime]
!gt[beginDate, endDate]
owner not in invited
}

sig Calendar{
    events: set Event,
    user: one User,
    public: one Bool,
}

sig Invitation{
event: one Event,
sender: one User,
receiver: one User,
status: one InvitationStatus,
}{
sender != receiver
}

// 2. System User

sig Email {}

abstract sig User{
    name: one Name,
    surname: one Surname,
    email: one Email,
    password: one Hash,
calendar: one Calendar,
identifier: one ID,
}

// Facts

fact DifferentEvent {
//two events can't have the same id
no disj e1, e2:Event | e1.identifier=e2.identifier
}

fact DifferentEmailDifferentUsers {
    // An email can't be used for registering 2 accounts
    no disj u1, u2: User | u1.email = u2.email
}

fact DifferentCalendar {
// make sure that two users have different calendars
```

```
no disj u1, u2: User | u1.calendar = u2.calendar
}

fact DifferentID {
// make sure that two users have different calendars
no disj u1, u2: User | u1.identifier = u2.identifier
}

fact CalendarOwner{
//the owner of the calendar have referring to his caledar
all u:User |
all c:Calendar |
u=c.user implies u.calendar=c
}

fact DifferentUser {
// make sure that two calendars have different users
no disj c1, c2: Calendar | c1.user = c2.user
}



fact EventMustHaveOwner {
all e: Event |
one u: User |
u in e.owner
}

fact CalendarMustHaveOwner {
    all c: Calendar |
        one u: User |
c in u.calendar
}

fact OwnerIsCreator{
all e: Event |
all i: Invitation |
i.sender = e.owner
}

// Run

//there are not event which finish before starting

assert Spatiotemporal {
no e:Event |
(lt[e.endDate, e.beginDate]
or e.beginDate=e.endDate
and gt[e.beginTime, e.endTime])
}
```

```
check Spatiotemporal

//there are no owner self-invited to the event
assert Eownership {
no e:Event |
 (e.owner=e.invited)
}

check Eownership

//there are not multiple event with the same id
assert SameEvent {
no disj e1,e2:Event |
e1.identifier = e2.identifier
}

check SameEvent

assert Cownership {
all c:Calendar |
all u:User |
u = c.user implies u.calendar = c
}

check Cownership

assert InvitationLoop{
no i:Invitation | i.sender = i.receiver
}

check InvitationLoop

assert NoEqualUser {
// make sure that two calendars have different users
no disj u1, u2 :User | u1.calendar=u2.calendar or
no disj u1, u2: User | u1.email = u2.email or
no disj u1, u2: User | u1.identifier = u2.identifier
}

check NoEqualUser

assert EventsCreatedInCalendar {
all c: Calendar |
all e: Event |
e in c.events implies
e.owner = c.user or
one i: Invitation |
i.receiver = c.user and i.event = e and i.status = accepted
}
```

```
check EventsCreatedInCalendar for 6

assert InvitesSentOnlyByOwner {
all e:Event |
all u:User |
all i:Invitation|
i.event=e and i.sender=u implies u=e.owner
}

check InvitesSentOnlyByOwner

pred addEvent [c, c': Calendar, e: Event] {
some e:Event |
some c,c':Calendar |
c'.events = c.events + e
}

pred delEvent [c, c': Calendar, e: Event] {
some e:Event |
some c,c':Calendar |
c'.events = c.events - e
}

assert DeletingEvents {
all c, c', c'': Calendar, e: Event |
no c.events and addEvent [c, c', e]
and delEvent [c', c'', e]
implies c.events = c''.events
}

check DeletingEvents for 10

pred show (c: Calendar) {
#c.user = 1
one u: User | u.calendar = c
all e:Event | e in c.events
}

run show for 6
```

13.2 An example world generated from alloy model

NOTE The model was optimally adapted to fit the document format, so it's recommended to execute personally the Alloy Program Listing to take benefit from a fully featured view.

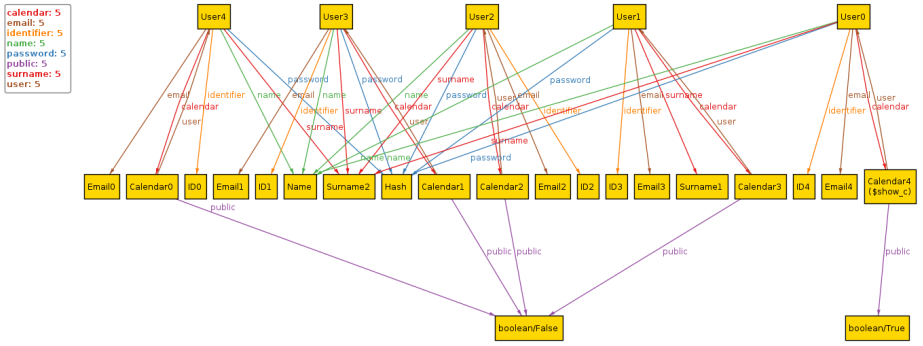

Figure 15: Every user has a different calendar, email, and ID. However, password hashes, names and surnames can be the same. It is also shown that calendars can be public and private.

14 Hours Count

Table 5: Hours of work per task

Task	Ferrai	Gabbianelli	Grazioso
Description of the Model	2	2	3
Introduction	1	0	1
Scenarios	0	1,5	1
Use cases	2	4	0
Sequence Diagrams	0	4	0
Non functional requirements	1,5	0	1,5
Alloy	5	0	5,5
Revision	1	1	0,5
	12,5	12,5	12,5
Total			

Contents