MeteoCal Project

Requirements Analysis and Specification Document

Authors:
Claudio Sanna
Walter Samà

October 2014 - January 2015

Contents

Chapter 1

Introduction

1.1 Description of the problem

We will develop a weather based on-line calendar for helping people scheduling their personal events avoiding bad weather conditions in case of outdoor activities.

Users, once registered, should be able to create, delete and update events. An event should contain information about when and where the event will take place, the number of invited people and whether the event will be indoor or outdoor. During event creation, any number of registered users can be invited. Only the organiser will be able to update or delete the event. Invited users can only accept or decline the invitation.

Whenever an event is saved, the system should enrich the event with weather forecast information (if available). Moreover, it should notify all event participants one day before the event in case of bad weather conditions for outdoor events; in addition to that, if possible, the system will inform the event creator about bad weather conditions three days before the event, and should propose the closest (in time) sunny or more suitable day for the event (if any).

Notifications are received by the users when they log into the system. It is also possible to see other users events and calender if they are respectively set to public.

1.2 Some considerations

We think that the description of the problem brings some (not clear enough) points that needs to be explained. First of all the idea of 'bad weather' is not unique because some activities may need different specifications, because for someone 'bad weather' can mean an hard rain and for someone else just a bit cloudy weather; so our intention is to define some labels that indicate the preferred weather to be noticed of; in addition to that some advanced settings for notifications will be available. In any case in order to keep the application as simply as possible all this function will be optional and will be explain in detail later in this document. The second unclear think is that the idea of calendar public for everyone may not satisfied every user; so we decided to introduce a so called "shared calendar" where only some users have the permission to see it; still, public calendar as specified in the description will also be implemented. Finally even if not explicit mentioned in the description, some basic function of a common calendar application will be introduce (e.g. repetition of events and other features).

1.3 Definitions and nomenclature

Here we present some terms that will be used in this document in order to not create misunderstandings:

Guest person not registered in the system.

Users person already registered into the system and has logged in.

Organiser user that create an event and he is the only one that can modify it.

Public calendar a calendar that is accessible by all the users.

Shared calendar a calendar that is accessible by some user, indicated by the calendar owner.

Shared user an user that has been added to a list of users that have access to a specific shared calendar

Private calendar a calendar that is not accessible by any user different from its owner;

Bad weather notification with this term we intend the function of the system to advise the event organiser and all the invited users in case of bad weather (both for three and one day before). It can be activated or not for each event.

Bad weather layer this term indicates the type of label selected by the event organiser to be informed of by the system if "bad weather notification" is activated.

Weather advanced settings optional settings that can personalise the type of weather to be informed of when bad weather notification is activated.

Calendar privacy with this term we indicate the "visibility" of a calendar to other users, in other terms this field indicate if the calendar is private, public or shared.

Event privacy with this term we indicate the "visibility" of an event to other users, just like the calendar privacy this field indicate if the event is private or public.

Month view the aspect of the calendar when the month view is selected for the calendar.

Week view the aspect of the calendar when the week view is selected for the calendar.

Day view the aspect of the calendar when the three days view is selected for the calendar.

1.4 Goals

Due to the requested specifications the features provided by the system are:

- Registration of a person into the system.
- Creation of events and possibility to invite registered users into it.
- Updating and removal of an event.
- Possibility to accept or decline an invitation to events.
- Possibility to see other user's calendar and events if those are respectively set to public or shared.
- Possibility to set the calendar public, shared or private.
- Possibility to set the events public or private.
- Get informed about weather's information for events created.
- Receiving notifications if the weather forecast isn't suitable for an event and eventually date suggestions in which move an event.
- Receiving notifications about events modifications and invitation.
- Possibility to set a personalised type of bad weather notifications.

1.5 Domain Properties

The proprieties that we think the world should have are here presented:

- The event organiser should participate in his event.

- Users may accept or decline an invitation if they intends to go or not.
- An user can attend two or more events in the same day but not at the same time.
- Weather forecast is as correct as possible but not totally precise.
- When suggesting a new date, there will always be a day that satisfied the event organiser weather needs, but it is not assured to be close to the previous event date.
- Due to weather variability during the day, will be provide 3 hours weather forecast data for 5 days forecast, and only dayly data for more than 5 days.
- The weather forecast is linked to a city.

1.6 Assumptions

In order to clarify better some points of the description we introduce some 'assumptions':

- an event is added to the user's calendar if he accepts the invitation. The event is set to private for default; users can still change it by updating the event.
- a public calendar is visible for all the meteocal registered users.
- people that have not accepted an invite are not informed of the bad weather notifications
- in a public or shared calendar, events still remain private unless indicated otherwise by the event organiser (at the time of the creation or after).
- private events will show only the time of the event, no other details, instead public events will provide all the information that the event organiser can see.
- bad weather notification will check the weather in the hours in which the event occur.
- the system in case of bad weather can suggest the same date but different time for an event.

1.7 Proposed System

For this application we will use a web platform in order to permit to people to interact between them with invitations and we grant the possibility to see other users' calendar.

1.8 Stakeholders

The stakeholder of this project are our professor who wants us to learn and improve our ability to create a software, from the requirements and design specifications to the implementation and the testing with the chosen software. So our goal is to show our understanding about the topics of Software Engineering.

Chapter 2

Actors

2.1 Unregistered People

Person that is not registered in the system: the only action that con be performed is registering

2.2 Registered People

Person that has already registered into the system. He can use all the features provided by the application after the log in.

Chapter 3

Requirements

We will define how the system will reach all the goals and the other functions:

- **Registration of a person into the system:** the system has to provide a log in feature and must have a database to store users' information.

- **Creation of events and possibility to invite registered users to it:** the system will provide a form to create a new event in which, optionally, the event organiser can invite other users.
  When a city and time is inserted, the system will provide the weather forecast for that city in the specified time.

- **Updating and deleting of an event:** the system will provide the possibility to modify or delete an event.

- **Possibility to accept or decline an invitation to an event:** the system will inform through notifications the invitation to the event.
  the system will also update the events when the users accept or decline the invite and will add the event to the users, calendar that has accepted the invitation.

- **Possibility to see other user's calendar and events if they are respectively set to public:** the system will provide a search engine to find others' calendar by the insertion of some information.
  the system will inform the searching user if he has the permission to see the calendar and event.

- **Possibility to set the calendar public, shared or private:** the system will allow the user to change his calendar privacy in the settings page;
  if a shared calendar is set with some shared users, they will be informed through notification by the system.
  The system will grant them the permission to see the shared calendar.
- **Get informed about weather's information for events created:** the system will provide weather forecast for each day, if available, thanks to information gathered from a weather forecast site.

- **Receiving notifications if the weather forecast isn't suitable for an event:** the system will inform the user one day before an event if the weather due to the information inserted by the organiser is unsuitable for the event.

- **Receiving date suggestion in which move an event if the weather is bad:** the system, 3 days before an event, in case of not suitable weather, will provide a date where the weather is suitable, as close as possible to the original date.

- **Receiving notifications about events modifications:** the system will send modifications information to all participants to an event when the administrator update and modify the event.

3.1 Functional Requirements

For *not registered* people:

1. Register.

For *registered* people:

1. Log In.
2. Add an event.
3. Update an event.
4. Delete an event.
5. Check Notification.
6. Accept or decline an invitation request.
7. Go to a date in the calendar.
8. Change personal settings.
9. Log Out.
10. Search for other user's calendar.
11. Change calendar view.
12. change calendar privacy.

3.2 Non Functional Requirements

3.2.1 Interoperability

The system will be connected with an external weather application in order to get weather information. This information will be provided to the events. If problems occurs (weather site not operating) an error will be displayed.

3.2.2 Security

The system will provide a private and secure access to the server by using an user name and a password. In this way every user can access only to its personal page.

3.2.3 User Interface

Here we present an idea of the possible user interface that the system will provide; it is not the final and definite UI as our intention is to present the main functionality that every page present.

Figure 3.1: This is the main page that the system will show right after the log in procedure; it presents the user calendar with all the events. Finally there are the buttons to go to the main section of the user private section as Home (to return to the main page), change calendar span time (month, day week), notifications, settings and log out; the notifications button will provide a number to show if there are notifications and how many. By clicking on a selected day the day span time will be presented.

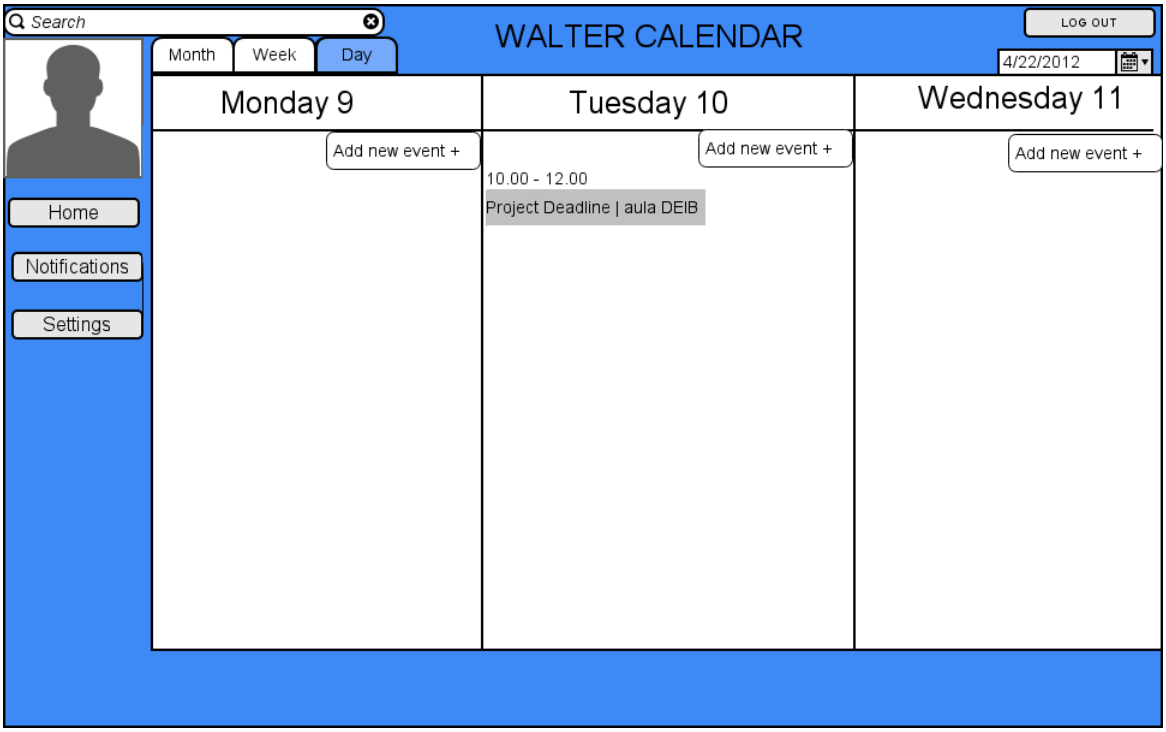

Figure 3.2: With this screen a three day calendar will be shown. The user can add new event by the '+' button and manage the existing event (updating, deleting, view event) by right clicking on them (or other procedure for example a touch system); The event will be displayed on the day they occur with place, time and an event colour.

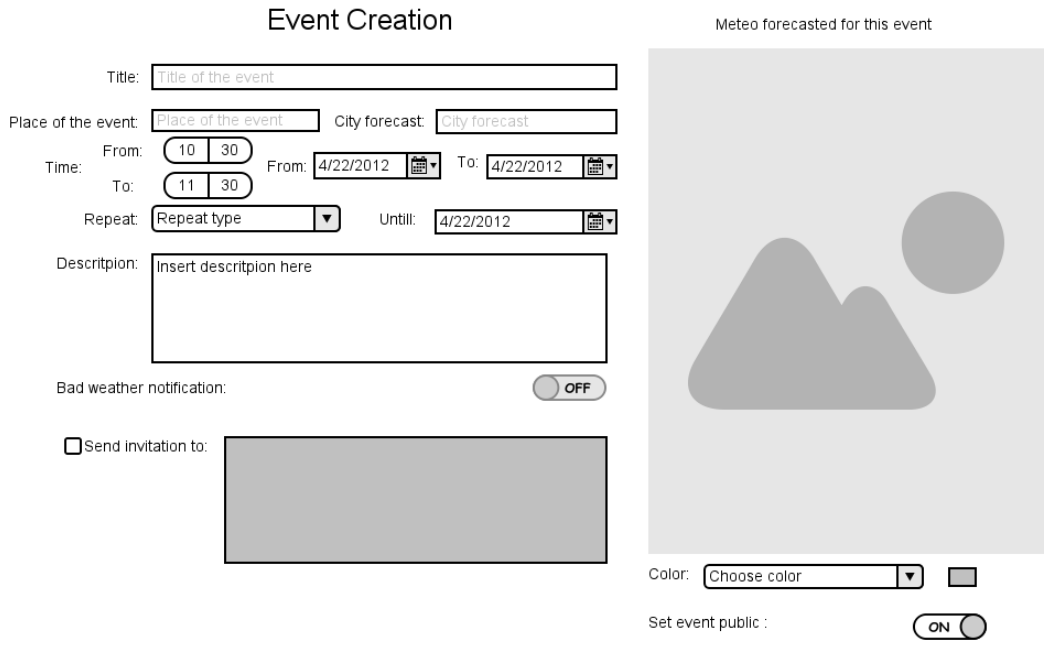

Figure 3.3: Event creation form with no bad weather notification and invited users.

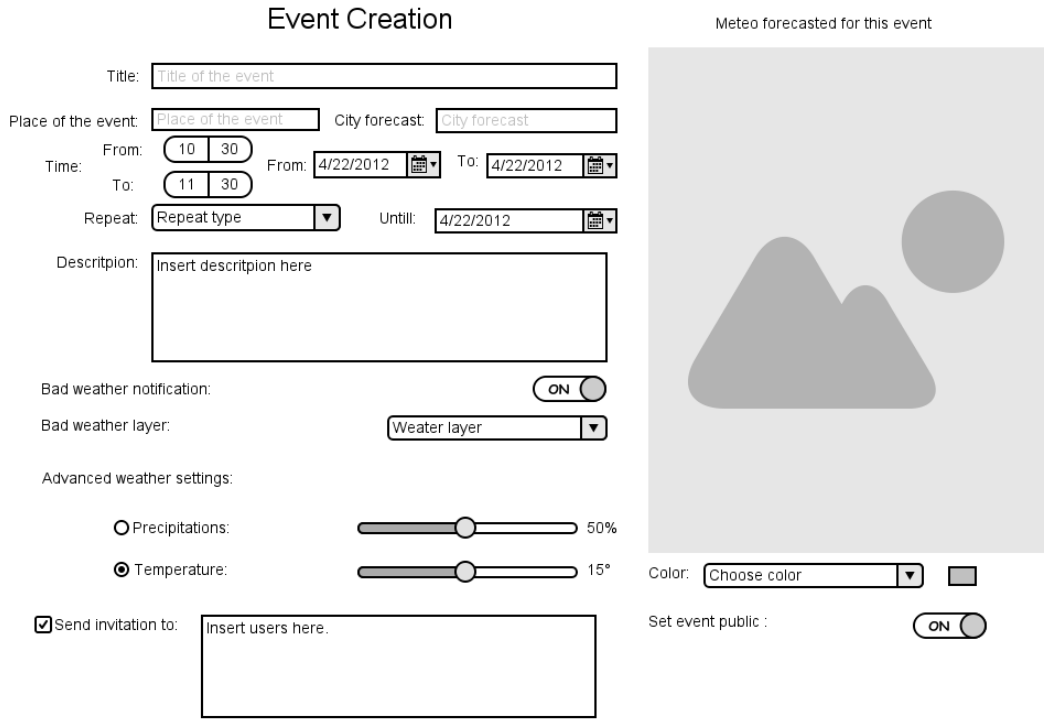

Figure 3.4: Event creation form with bad weather notification and invited users.

Figure 3.3/3.4: These screens show the form to create new event; two form type are presented to show the difference between an event with all the functionality, and an event that does not contains bad weather notification and user invitation. The description of all the functionality will be presented in the specifications section.

3.2.4 Documentation

We will provide a detailed documentation of this application using various document:

- RASD: Requirement Analysis and Specification Document (this document), to give an overall view of the problem the given problem and to analyse in a detailed way which are our goals and how to reach them defining requirements and specification.
- DD: Design Document, to define the real structure of our web application and its tiers.
- JavaDoc comments in the source code: to make anyone that wants to develop the platform or do maintenance on it understand the code.
- Installation Manual: a guide to install Meteocal.
- User Manual: a guide to start using Meteocal..
- Testing Document: a report about another meteocal application that we will test.

3.2.5 Architectural Considerations

We will use JEE platform with a database in which we will store all users' information, events and notifications; we will add more precise data about the architecture when we will do the Design Document and the implementation. Obviously the users need to have an Internet connection to use the application.

Chapter 4

Specifications

In this section we specify in detail all of the functional requirements introduced:

- Registration is made on the home page. Each guest must insert an unique user name, a password, an email and chose if set the calendar private, shared or public (private is the default setting).

- To log in is request the user name and the connected password.

- Adding an event is possible by clicking on the add button on the selected day on the calendar. Each event must provide: a title, a place, a city nearby for the weather forecast, starting and ending time, a brief description (not obligatory but advised) and the event privacy; will also be provided the possibility to repeat the event for more day and a colour to assign to the event; optionally can be added the invitation list, the bad weather notification, the bad weather advanced settings.

- Because weather forecast is linked to a city, is necessary to indicate a city nearby the place of the event to obtain weather forecast.

- Standard bad weather notifications will provide three type of label: cloudy, rainy, snowy; this three label has an order(cloudy¿rainy¿snowy); this mean that cloudy label will inform the organiser even if it is rainy or snowy, rainy label will inform the organiser also when is snowy and the snowy label will inform the organiser only if it snow (and will not inform him if it is rainy or just cloudy).

- Each update or delete is made by clicking the event on the calendar via the day view. The people invited will be notified that the event has been update or delete.

- When an users accept an invitation to an event, that event is added to its calendar and the user can access to all information about it; however is not possible for him or her to modify it. It is only possible for the user to set the event privacy for himself.

- When using the searching tool, a message will inform the searching user if he has not the permission to see the searched users' calendar. Otherwise the main page of the searched users will be shown.

- If an event in a public (or shared) calendar is public, every user (or shared user) can access to

it and see all the information; otherwise only starting and end time and event title are available and the event will be coloured in dark-grey.

- An user can check his notifications as invitation to an event, modifying of an event,if someone has accepted or declined an invitation and maybe information from the system. A possible 'go to event' button can be provided.

- In the notification zone the users acknowledge of an invitation, and go to a page where the event is displayed and he or her can accept or decline the invitation.

- Navigation through days and months are made by previous and next button on the main page of a users.

- On the calendar will be displayed the weather forecast to organise better outdoor activities.

Chapter 5

Scenarios

Here are presented some basic scenarios that can occur:

1. Alexander need a new calendar to organise both his personal and working activities. He finds out the existence of the MeteoCal application and decide to try it; first of all he go the the main page and start the registration phase. He choose a user name, an E-mail and a password and insert it; unfortunately his user name has already been chosen so he decided for another user name that result to be available; then he choose to set his calendar private. After that he performs the log in into the system by inserting his username and password. The first thing he see is his calendar in which he decide to add a new event to try the functionality of the system; he add the conference he have to attend on the weekend so he click on the chosen day and a form is opened. He insert basic information (title, place, time,a brief description), set no invitations and decide to not set th bad weather notification because it is an indoor event e does not required any type of weather.

2. Alexander want to organise his birthday at the park so he decided to use his MeteoCal to achieve this; he suggested it to most of his friend so he thinks it suits perfectly to manage the invitations. So he goes to his main page (after log in), where the calendar is, and create a new event on the date of his birthday and invite his friend by adding the emails of each of them; because this time the event created is an outdoor activity he active the bad weather notifications setting the bad weather layer to rainy; in addition to that decide to try the advanced settings and set 30% of % precipitations and a minimum of 14 degrees for the temperature. The next day when he log in into the system he notice that there are two new notifications : his friends Walter and John have accepted the invitation. Three days before the event Alexander see a notification that indicates % of rain greater that the one he selected and a suggestion of a date 3 days after his birthday where the rain % is very low; so he goes to the event date and select it in order to change the date of the event. All the invited users are informed of the date notification. The day before the new date a new notification states that for the selected date new weather forecast indicates rain, so he decide to delete the event. The invited users are again informed of the deleting of the event.

3. Alexander has to make a new project with his colleague John so he needs to schedule his calendar in order to make meetings with him. So he enter his MeteoCal applications and goes to setting where he set his calendar to shared adding the user jhonFulz (john's user name in MeteoCal) to the list of shared users. Now john can access to Alexander's calendar and see when he has free time to make a meeting; then after speaking with John they decide to make a meeting on the next Friday so Alexander goes to his MeteoCal application and set the calendar privacy to shared adding john to the shared users list; after that he create a new meeting event,

where he invites his friend, so john can see the details of the event (even if private, john, as an invited users, can see it as public). The next day his friend Roberto send an invitation to a party to Alexander, who decides to accept it and the event is automatically added to his calendar; however Alexander wants it to remain private and so he go to the day of the event, select it and sees that it is already set to private.

4. Alexander want to organise a surprise for his friend Jessica. He wants to know the days in which she is free without letting her know; he knows that she has a MeteoCal account so he decided to search for her. From the home page he insert her email address in the search bar but a message advise him that she has set his calendar private. With an excuse he asks to Jessica to set her calendar shared and add in to the shared users list so that he could see it; the next day he tries again to access to Jessica's calendar and this time he is successful. He goes through the days and see dark grey marked events (private even), in which are shown only time but he can't click on it, and on the 25th January he sees an event titled 'special event' and sees that is not coloured in dark grey (public event). Because he was interested, he decide to open it and he find out that the event was set to public so he can see all the details (date, time, place, participants). So he goes on the calendar and sees that on the 27Th January she is free and decided that he will invite her on that day.

5. Alexander was assigned by his boss to create and manage a set of events for people of the society in which he is working and for people who do not belong to it; he suggest him to use the MeteoCal application because all his colleagues has MeteoCal.So he create a new user and set the new calendar created to public. Thanks to this all his colleagues and other people can see the calendar and see when the events are set.

Chapter 6

UML Models

6.1 USE CASE Diagram

In this section will be present some use case diagrams to show the possible action that can be taken by each type of users. A brief description will then explain in a more detailed way for each use case.

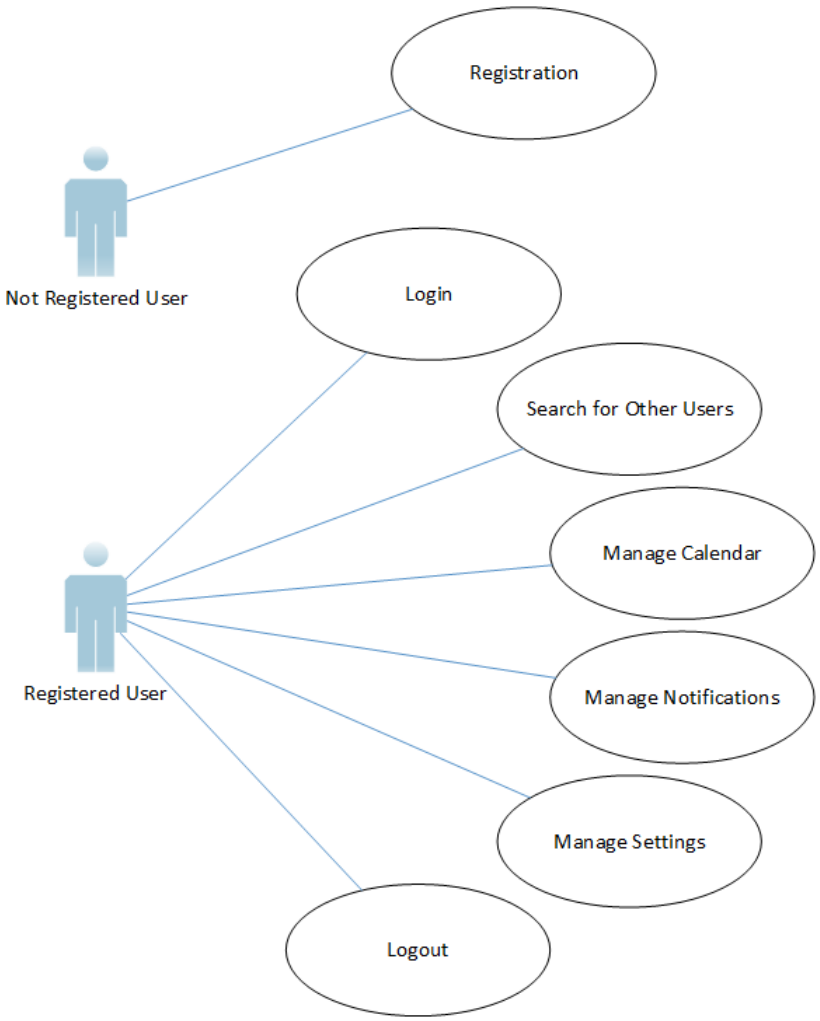

Figure 6.1: Main Use Case

Here are presented two use case: log in and registration; other use cases will be explained in the next sections:

Name	Registration
Actors	Not registered users
Entry condition	no entry condition
Event flow	1. The guest open the MeteoCal application. 2. The guest click on 'Sign in' link. 3. A new window is opened. The guest is requested to insert a user name, an email and a password; then the users will be requested if he intends to set his calendar public or private.
Exit Condition	The system will add the new user to the database and grant him access to the application.
Exceptions	• The user name inserted is already in use. • Password and email inserted wrongly.

Name	Log in
Actors	Registered users
Entry condition	no entry condition
Event flow	1. The guest open the MeteoCal application. 2. The guest insert his user name and password and click on 'log in'. 3. The system accepted the data inserted, and the main page is shown to the user.
Exit Condition	No exit conditions.
Exceptions	• If the user name or password inserted doesn't exist in the database an error message will be shown.

6.1.1 Manage Calendar and Search Users

Here we will present in detail two of the main Use Case and the interaction between each other.

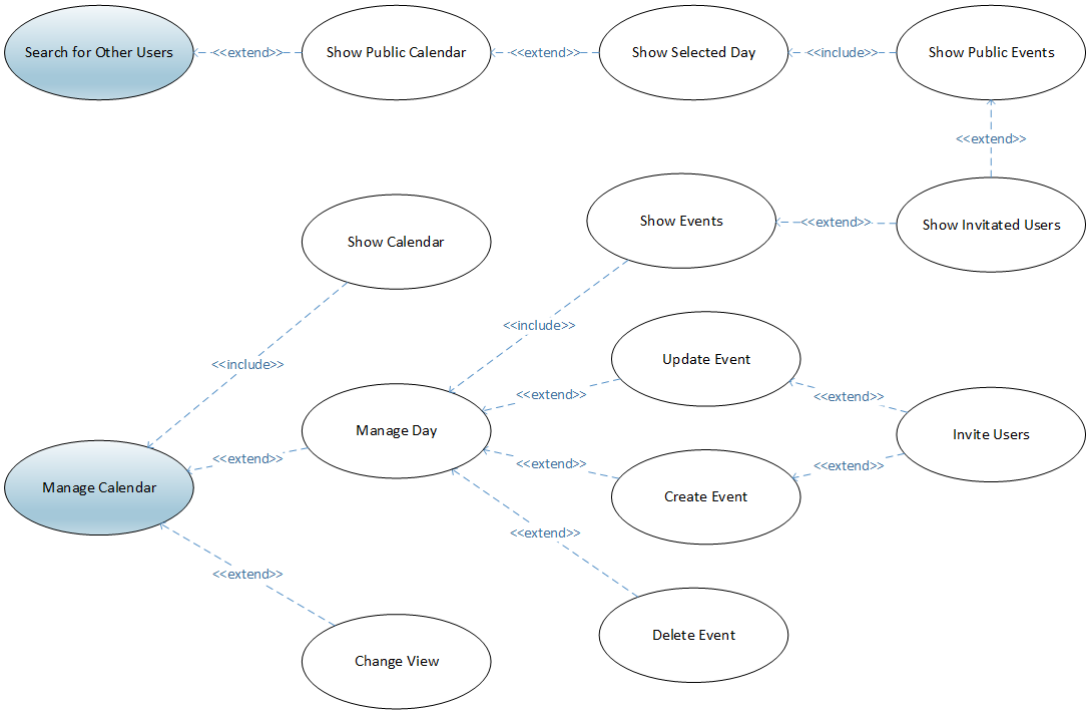

Figure 6.2: Manage Calendar and Search Users

Name	Change view
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user from the main page click on month or day view butto. 2. The month or day view will be shown
Exit Condition	No exit conditions.
Exceptions	No Exceptions.

Name	Search for other users
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user insert in the searching bar the user's name in order to see its calendar. 2. If the user has permission to see the calendar (if it is public or shared to the searching user) the main page of the searched user is shown.
Exit Condition	No exit conditions.
Exceptions	• If the user doesn't exist in the database an error message will be shown. • If the user that made the searching do not have the permission to see the searched user's calendar he will be informed of this with a message in a window.

Name	Show public events
Actors	Registered users
Entry condition	Log in successful. User searched has a public calendar.
Event flow	1. The user insert in the searching bar the user's name in order to see its public calendar. 2. The searched user calendar will be shown. 3. The user navigate though the calendar. 4. The user click on the day in which he is interested in. 5. The day view of the calendar will be shown. 6. The user click on the event in which he is interested in. 7. All the information about the event will be shown.
Exit Condition	No exit conditions.
Exceptions	• If the user doesn't exist in the database an error message will be shown.

Name	Manage day
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user select a day in his calendar. 2. A new page will be shown with a three day's calendar.
Exit Condition	No exit conditions.
Exceptions	No exception

Name	Create event
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user select a day in his calendar. 2. A new page will be shown with a three day's calendar. 3. The user will click on a '+' on the day in which adding the new event. 4. A new page will be shown with the create event form; the user is requested to enter event data (title, place and time), can add some optional information (description, set advanced weather setting, activate bad weather notification) and can invite other people by inserting in a input text box the emails of every user that intend to invite separate by ';'.
Exit Condition	The new event will be created and inserted in the user's calendar. If invitations are made, notifications will be sent to all the invited users
Exceptions	• If one of the invited user doesn't exist in the database an error message will be shown indicating "not existing users". • If the weather is not available, an error message will be shown instead of the weather's view.

Name	Update event
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user click on a selected day in his calendar. 2. A new page will be shown with a three day's calendar. 3. The user will right click on a event on the event he intends to update. 4. A new page equals to the event's creation one will be shown, with all the field already filled; the user can insert new event data (title, place and time), can add some optional information (description, set advanced weather setting, activate bad weather notification) and can invite other people by inserting in a input text box the emails of every user that intend to invite.
Exit Condition	The event will be updated. If new invitations are made, notifications will be sent to all the invited users
Exceptions	• If one of the invited user doesn't exist in the database an error message will be shown indicating the not existing users. • If the weather is not available, an error message will be shown instead of the weather view.

Name	Delete event
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user click on a selected day in his calendar. 2. A new page will be shown with a three day's calendar. 3. The user will right click on a event on the event he intends to delete. 4. A message will shown the successful of the operation.
Exit Condition	The event will be deleted from user calendar. Notifications will be sent to all the invited user to inform of the event deleting
Exceptions	No Exception.

6.1.2 Manage Notifications

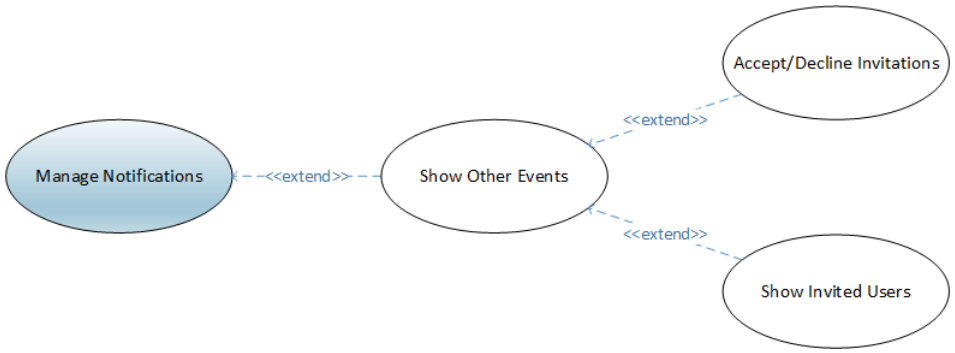

Figure 6.3: Use Case Notifications

Name	Show notification
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user click on "notifications" on his main page. 2. The notifications page will be shown.
Exit Condition	No exit conditions.
Exceptions	No exceptions.

Name	Show Other events
Actors	Registered users
Entry condition	Log in successful. The user have received a notification of an event(update).
Event flow	1. The user click on "notifications" on his main page. 2. The notifications page will be shown. 3. From the notifications list the user will click on a notification of an event update. 4. A new page will be opened that shows the event details.
Exit Condition	No exit conditions.
Exceptions	No exceptions.

Name	Accept/decline invitations
Actors	Registered users
Entry condition	Log in successful. The user have received an invite to an event(update).
Event flow	1. The user click on "notifications" on his main page. 2. The notifications page will be shown. 3. From the notifications list the user will click on a notification of an event update. 4. A new page will be opened that shows the event details. 5. On that page the users select accept or decline invite.
Exit Condition	The users will be added to the users list that have accepted the invitation or to the list of users that have declined the invitation if respectively he accepts or declines the invitation. A notification will be sent to the event organiser.
Exceptions	No exceptions.

6.1.3 Manage Settings

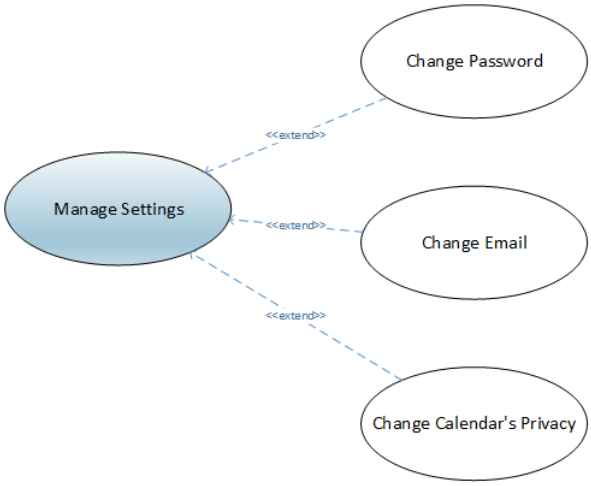

Figure 6.4: Use Case Settings

Name	Change password
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user click on "setting" on his main page. 2. The user select "change password". 3. The user insert the old password and the new password. 4. The new password will be set.
Exit Condition	The new password will be set as the user current password
Exceptions	If a non valid password is inserted an error message will be shown.

Name	Change email
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user click on "setting" on his main page. 2. The user select "change email". 3. The user insert the password and the new email. 4. The new email will be set.
Exit Condition	The new email will be set as the user current email
Exceptions	If a non valid email is inserted an error message will be shown.

Name	Change calendar's privacy
Actors	Registered users
Entry condition	Log in successful
Event flow	1. The user click on "settings" on his main page. 2. The user select a new privacy (public, private, shared). 3. If the calendar is set to shared, a text input box will be shown and the user will insert the user name of the people he wants to show the calendar to.
Exit Condition	The new privacy will be set. A notification will be sent to every user inserted for the shared calendar
Exceptions	No exception

6.2 CLASS Diagram

Here we present a first and simple Class Diagram in which we added only the main attributes and associations; we don't add the name and the multiplicities because we are not sure how to implement the application, here we just want to have an idea of the main entities. Maybe after the first implement decision is better to talk with the stakeholders to really understand if our decision is the right one.

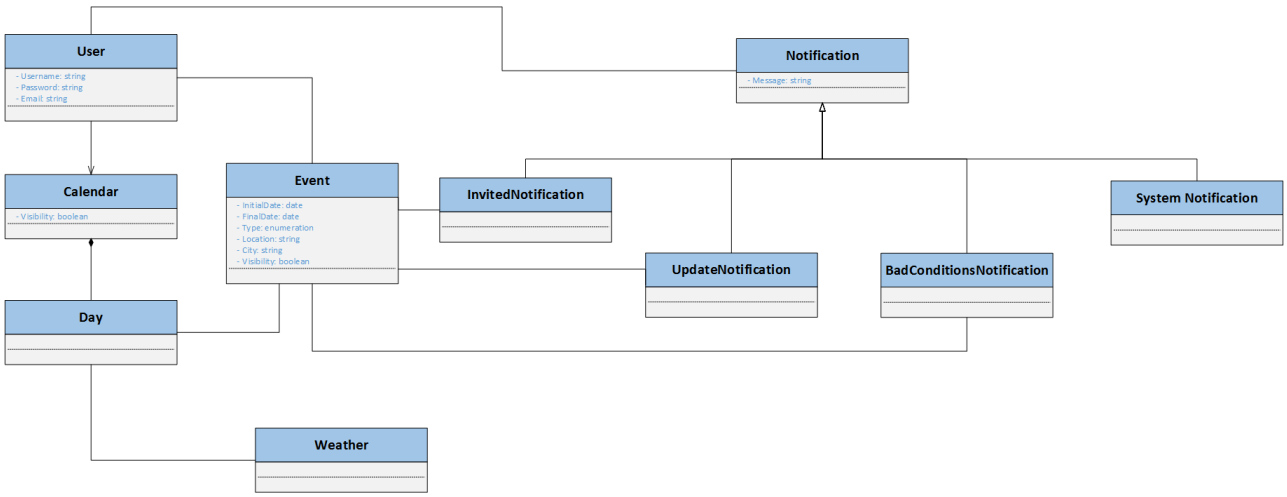

Figure 6.5: Class Diagram

6.3 SEQUENCE Diagram

Here we present some of the main cases that we consider important to understand how the application will work.

6.3.1 Register

A new user want to register himself in the MeteoCal application.

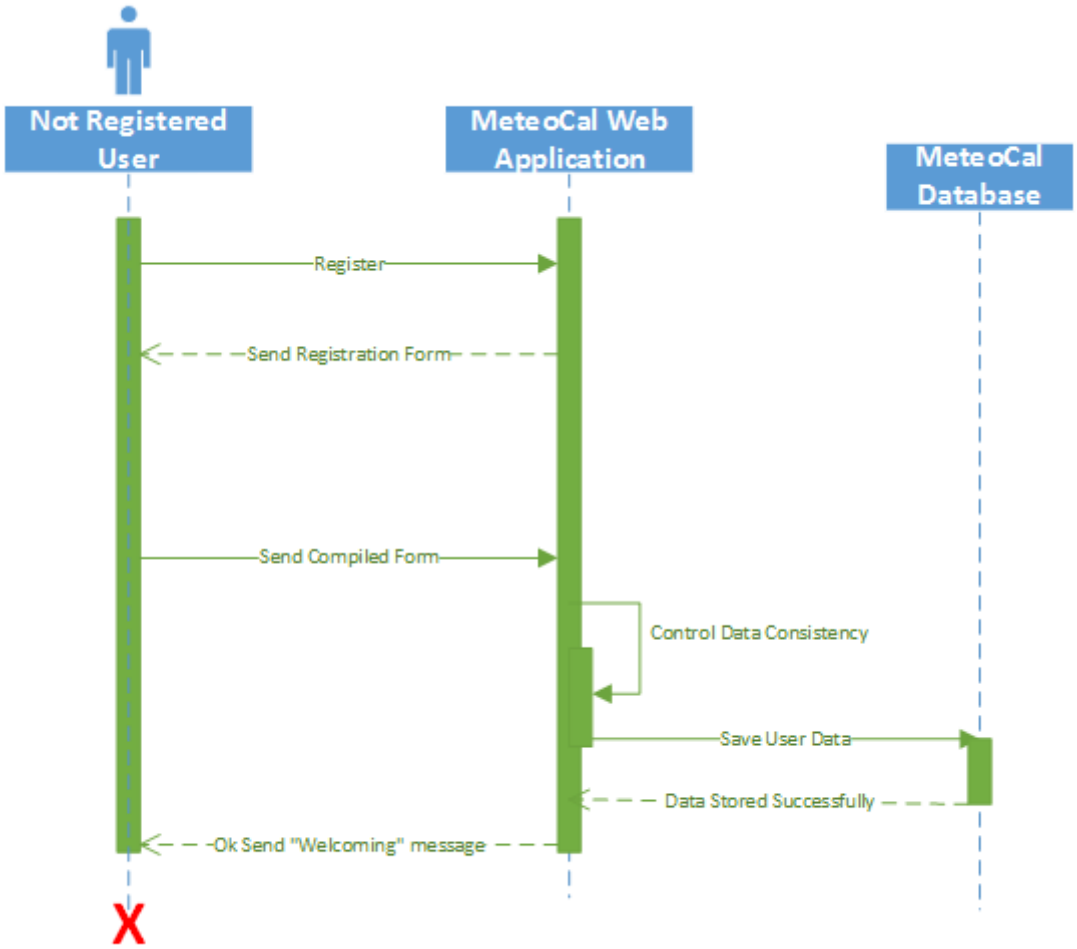

Figure 6.6: Sequence Diagram Register

6.3.2 Login

A registered User want to Log in the application.

NB: in this Diagram we will show how the control of the data, that the user send to the system, will work. This control named :"Data Control Consistency" and the corresponding Loop and Alternative Fragment, will be avoided in the further diagrams only to increase clarity.

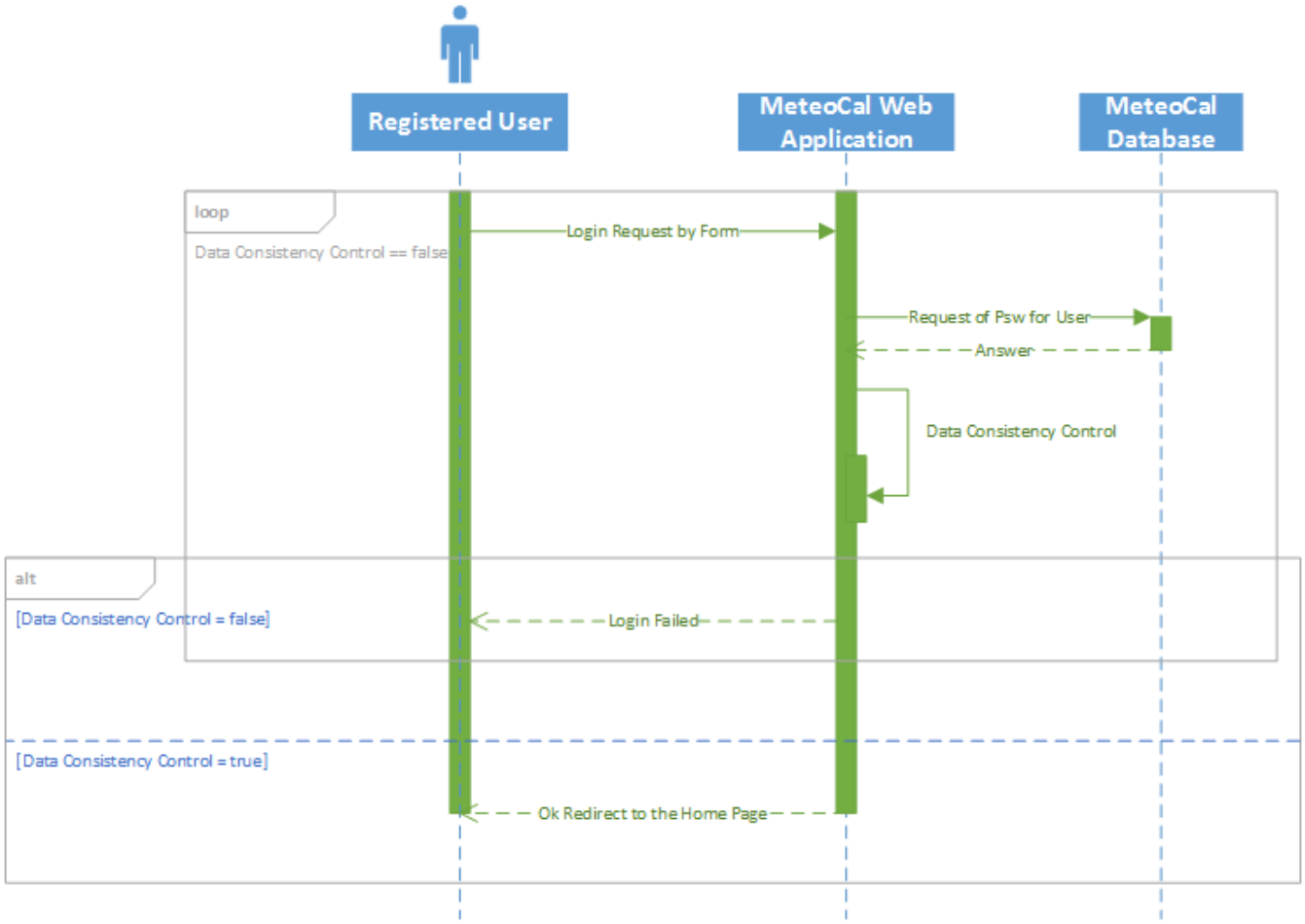

Figure 6.7: Sequence Diagram Login

6.3.3 Create Event

A registered and logged user want to create an event and save it on the calendar.

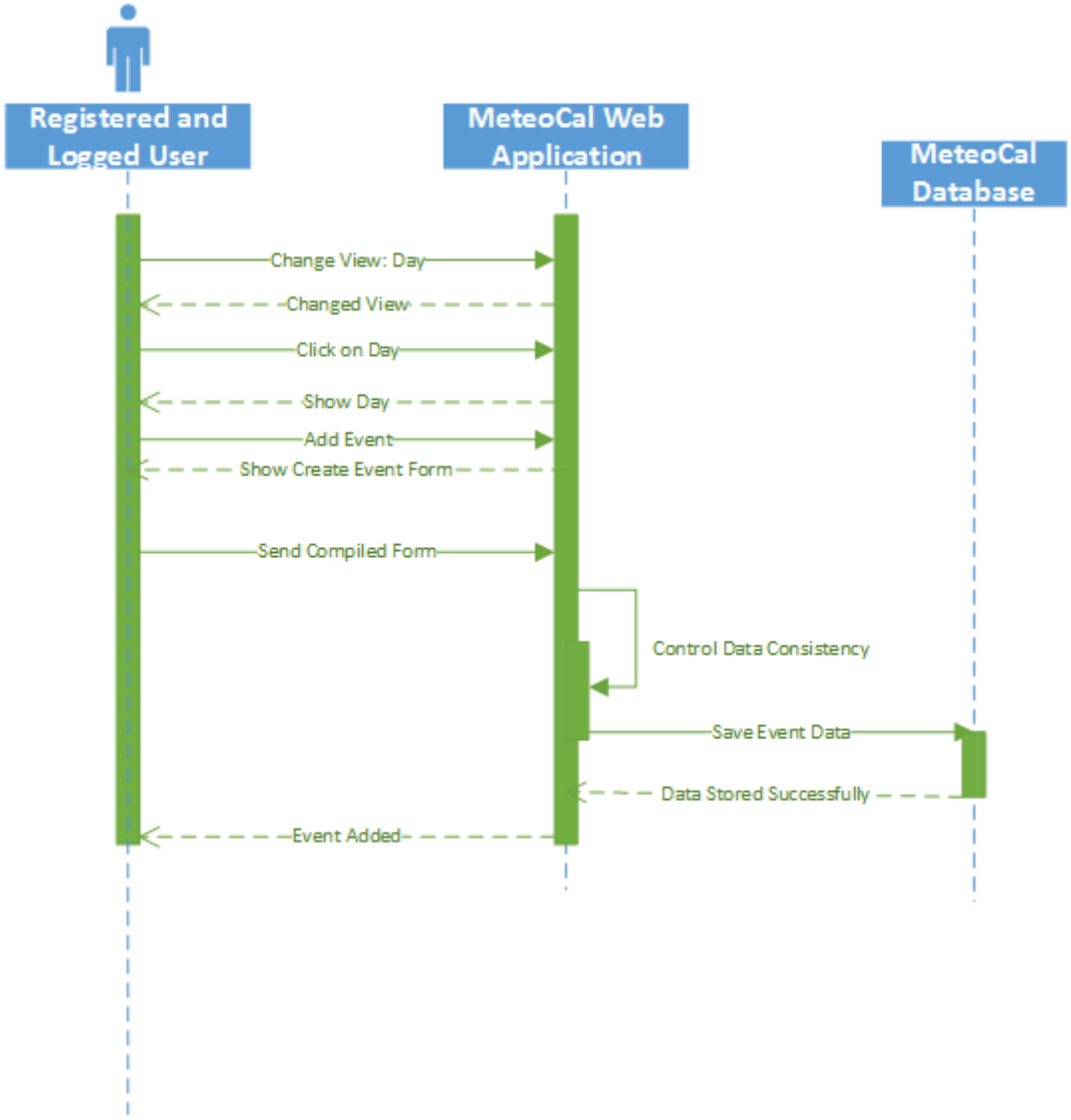

Figure 6.8: Sequence Diagram Create Event

6.3.4 Notification, Update Events and Invitations

In this big Sequence Diagram we want to acknowledge you about how the Update, Notification and Invitation feature will work.

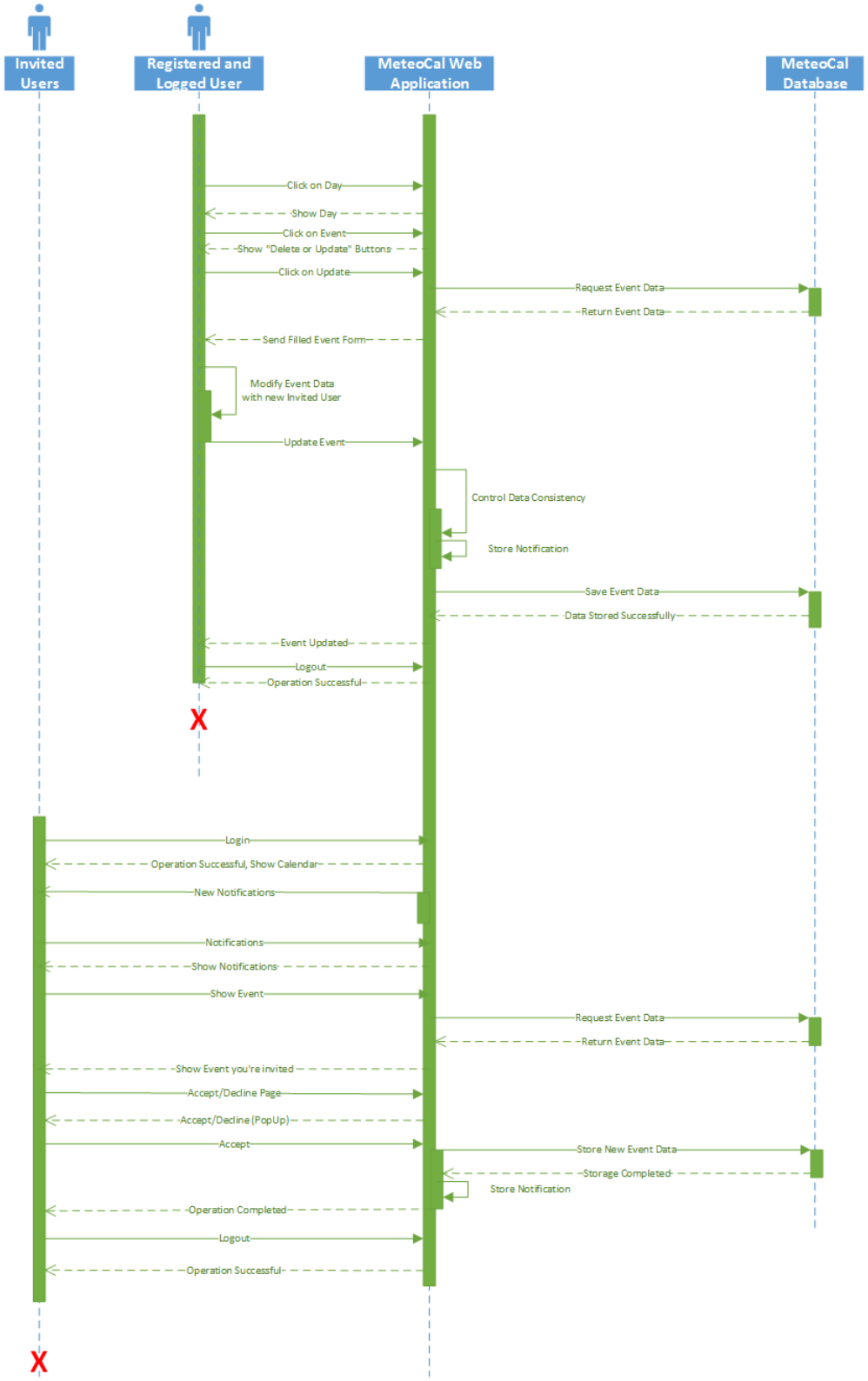

Figure 6.9: Sequence Diagram Update Notification and Invitations

6.4 State Chart, Activity Diagram

Here we will explain in detail how the interaction between the system and the users will work using these diagram.

6.4.1 Register

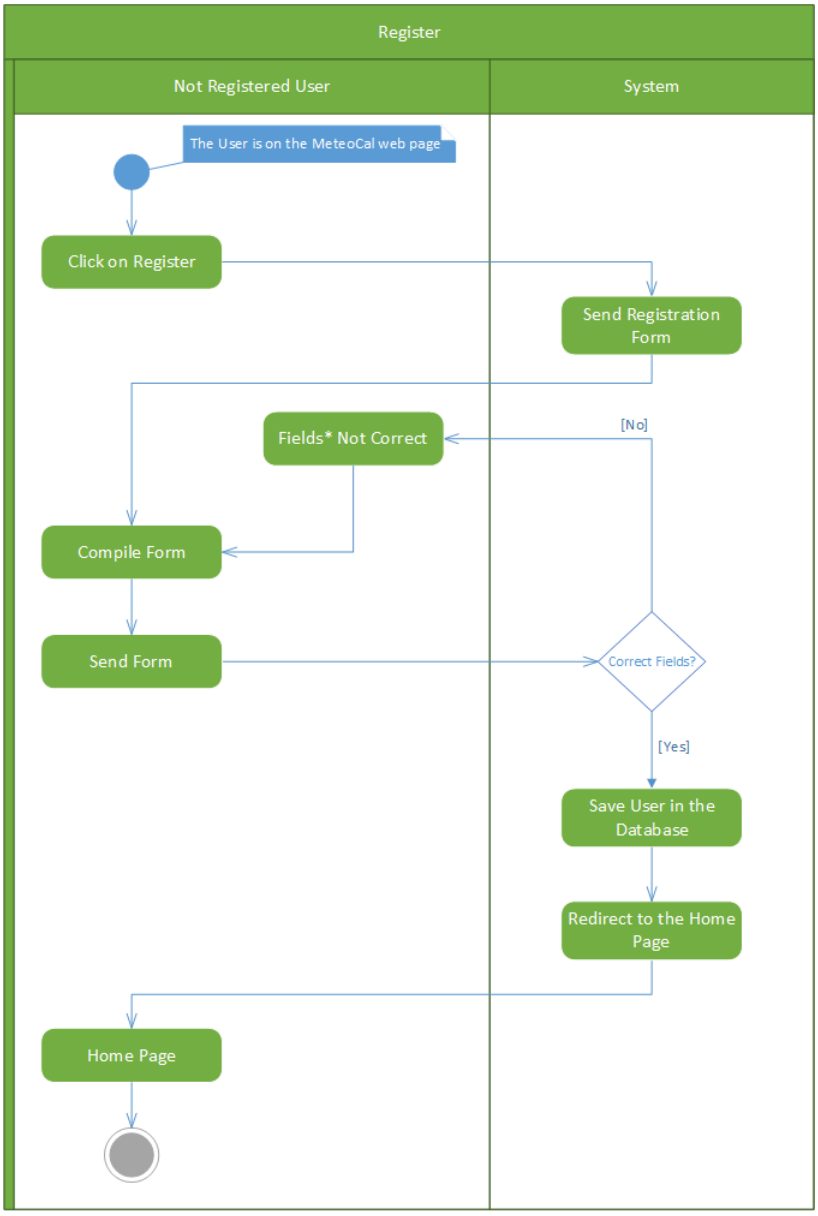

Figure 6.10: Activity Diagram for the User Registration

6.4.2 Login

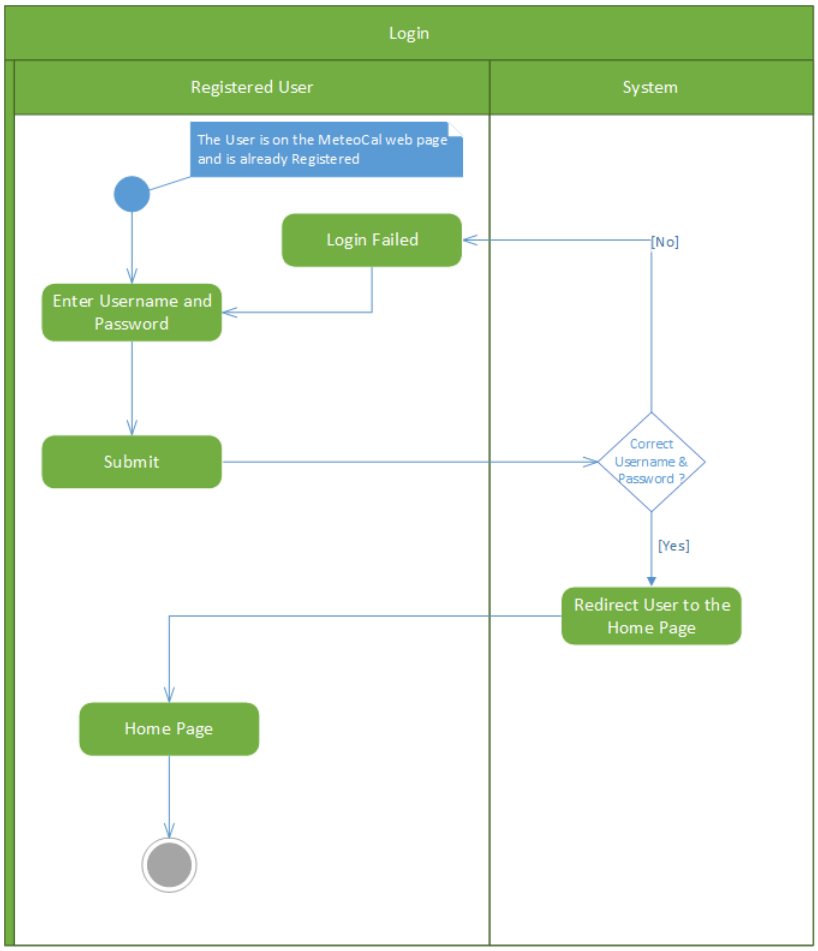

Figure 6.11: Activity Diagram for the User Login

6.4.3 Create Event and Notifications

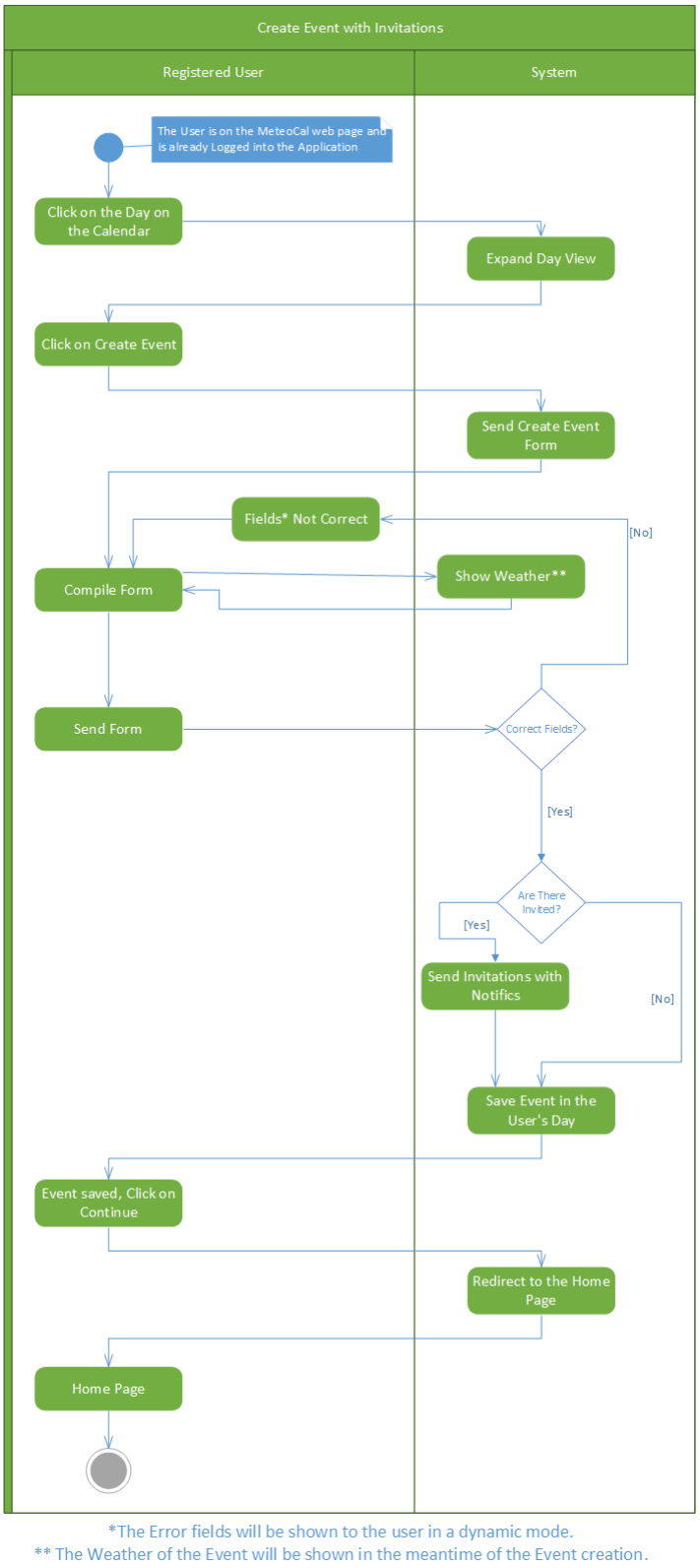

Figure 6.12: Activity Diagram for the Event Creation with Invited Users

Chapter 7

Alloy

In this paragraph we try to verify if all the assumption an specifications presented in this document are consistent using Alloy Analyzer. We report below the code used and the result obtained:

```
sig User{
   calendar: one Calendar,
   notification: set Notification
}

sig Day {
   events: set Event,
   weather: lone Weather
}

sig Calendar{
   days: set Day
} {#days>0
}

sig Event{
   eventDays: set Day,
   acceptedInvitation : set User,
   refusedInvitation : set User,
   organiser: one User
}{
   #eventDays>0
}

sig Weather{
   referredDay : one Day
}

abstract sig Notification{}

sig  InvitedNotification extends Notification {
   invitedForEvent: one Event
}

sig BadConditionsNotification extends Notification {
   weatherEvent: one Event
}

sig UpdateNotification extends Notification {
   updatedEvent: one Event
}
```

```
sig SystemNotification extends Notification {}

sig AcceptDeclineNotification extends Notification {
   event : one Event
}
fact allCalendarHasAnUser{
   all c : Calendar | some u : User   | u.calendar = c
}

fact noSameCalendar {
   all v', v : User  | v.calendar = v'.calendar implies v = v'
}

fact dayBelongToCalendar {
   all d : Day | some c : Calendar   | d in c.days
}

fact dayBelongToCalendar {
   all d : Day | all c ,c' : Calendar   | (d in c.days and d in c'.days) implies c = c'
}

fact eventDayConsistency {
   all d : Day | all e : Event |  e in d.events iff d in e.eventDays
}

fact eventExclusionContraint {
   all e : Event |  #(e.organiser & e.acceptedInvitation) = 0
               and  #(e.organiser & e.refusedInvitation) = 0
               and  #(e.acceptedInvitation & e.refusedInvitation) = 0
}

fact acceptedInvitationUserHasEvent {
   all u : User | all e : Event | u in e.acceptedInvitation implies e in u.calendar.days.events
}

fact  refusedInvitationUserHasNotEvent {
   all u : User | all e : Event | u in e.refusedInvitation implies not (e in u.calendar.days.events)
}
```

```
fact organiserHasEvent {
   all u : User | all e : Event | u in e.organiser implies e in u.calendar.days.events
}

fact notificationHasAnUser {
   all n : Notification | some u : User | n in u.notification
}

fact noSharedNotification {
   all n : Notification | all u, u' : User | n in u.notification and n in u'.notification implies u = u'
}

fact invitationToInvitated {
   all n : InvitedNotification  | all u : User |  (n in u.notification) implies  (u in n.invitedForEvent.acceptedInvitation or u in n.invitedForEvent.refusedInvitation)
}

fact noInvitationToSameEvent {
   all u : User | all n, n' : InvitedNotification | (n in u.notification and n' in u.notification) and (n.invitedForEvent = n'.invitedForEvent) implies n = n'
}

fact updateNotificationsToAcceptedInvitations {
   all n : UpdateNotification | all u : User | n in u.notification implies  u in n.updatedEvent.acceptedInvitation
}

fact badWeatherNotificationsToParecipants {
   all n : BadConditionsNotification | all u : User | n in u.notification implies  u in n.weatherEvent.acceptedInvitation or  u in n.weatherEvent.organiser
}

fact noMoreThanTwoBadWeatherNotification {
   all e : Event | (sum n : BadConditionsNotification | #((n->n.weatherEvent).e)) < 3
}

fact AcceptDeclineNotificationToOganiser {
   all n : AcceptDeclineNotification | all u : User | n in u.notification implies u = n.event.organiser
}

fact invitatedEqualsToNotification {
   all e : Event | (sum n : AcceptDeclineNotification | #(n->n.event).e) = (#e.acceptedInvitation + #e.refusedInvitation)
}


fact differentWeatherPerDay {
   all d,d' : Day | d.weather = d'.weather implies d = d'
}
assert noInvitationToOrganiser {
   all n :  InvitedNotification | all u : User | n in u.notification implies not n.invitedForEvent.organiser = u
}

check noInvitationToOrganiser

assert noEventUpdateToOrganiserOrRefusedInvitation {
   all n :  UpdateNotification | all u : User | n in u.notification implies not (n.updatedEvent.organiser = u or n.updatedEvent.refusedInvitation = u)
}
check noEventUpdateToOrganiserOrRefusedInvitation

assert noWeatherNotificationTorefusedInvitation {
      all n :  BadConditionsNotification | all u : User | n in u.notification implies not (n.weatherEvent.refusedInvitation = u)
}

check  noWeatherNotificationTorefusedInvitation

assert noAcceptDeclineNotificationToInvited {

}

assert noBothAcceptedRefusedInvitation {
   all u : User | all e : Event | u in e.acceptedInvitation implies (not u in e.refusedInvitation)
}

check  noBothAcceptedRefusedInvitation
```

```
assert creatorCalendar {
   all e : Event | all u : User | e.organiser = u implies e in u.calendar.days.events
}

check creatorCalendar


assert eventBelongToCalendar {
    all e : Event | some c : Calendar | e in  c.days.events
}
check eventBelongToCalendar

assert invitationEqualsToInvitated {
   all e : Event | (sum n : InvitedNotification | #(n->n.invitedForEvent).e) = (#e.acceptedInvitation + #e.refusedInvitation)
}

check  invitationEqualsToInvitated

pred show {#Event>0 #User>0  #BadConditionsNotification>0 #SystemNotification>0 #UpdateNotification>0 #InvitedNotification>0 #AcceptDeclineNotification>0
run show for 5 but 10 Notification
```

Here we present the output:

```
7 commands were executed. The results are:
  #1: No counterexample found. noInvitationToOrganiser may be valid.
  #2: No counterexample found. noEventUpdateToOrganiserOrRefusedInvitation may be valid.
  #3: No counterexample found. noWeatherNotificationTorefusedInvitation may be valid.
  #4: No counterexample found. noBothAcceptedRefusedInvitation may be valid.
  #5: No counterexample found. creatorCalendar may be valid.
  #6: No counterexample found. eventBelongToCalendar may be valid.
  #7: Instance found. show is consistent.
```

Figure 7.1: Alloy output

7.1 World Generated

Here we present the world generated with Alloy :

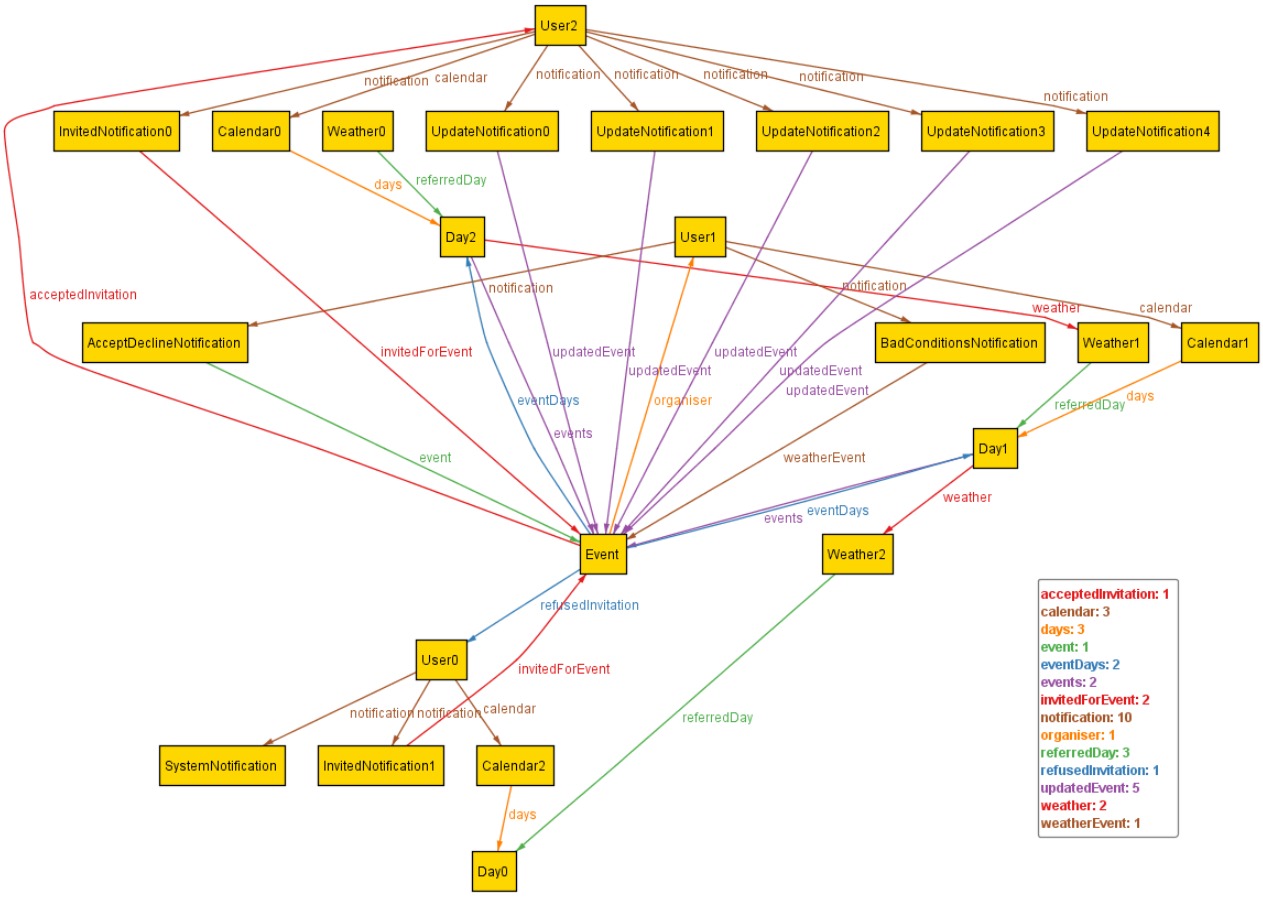

Figure 7.2: Alloy instance

Chapter 8

Used Tools

These are the software used to create this RASD:

- **Share Latex Web Platform**: used to redact and format this document.
- **Moqups Web Platform**: used to create the User Interface sketches;
- **Alloy 4.2**: used to verify our world consistency and generate the alloy diagram.
- **Microsoft Visio 2013**: used to create all the Uml Diagrams;

Bibliography

[1] Rafaela Mirandola. Requirements slides, 2014.