POLITECNICO
DI MILANO

SW Requirement Analysis and Specification Document

MeteoCal

November 14, 2014

Manuel Grillo
Francesco Fermi
Jerry M. Jude

0. Index

Table of Contents of RASD v.1.0

(continues)

1. Introduction

1.1. Description of the given problem

The aim of this project is the implementation of a weather-based online calendar called MeteoCal (Meteo Calendar). This system is designed for helping people schedule their personal events, avoiding bad weather conditions when required.

The system will allow the registration of new user with some personal information (e.g. name, surname, permanent address, email, nickname and password). Registered users, as "organizers", will be able to create and manage public or private events, invite other registered users and accept or reject invitations to events. In addition, every user will be able to import or export their calendar. It will also be possible to view the public schedules (i.e. calendars) of other users. Moreover, the system, will avoid the overlap of the events in the calendar.

Users will be alerted with notifications about the events they will attend, but they could also receive other general notifications. As mentioned before, to allow avoiding bad weather conditions, the system will enrich the events with updated weather forecast information. Users will be alerted with notifications about the events they will attend. Notifications will also be generated when adverse change in the weather forecast – for the date of the event – happens. Other possible event-related notifications will be the ones that inform the user about changes in the location or time of the event.

Three days before a weather-compromised event, the system will propose to the organizer the closest day with suitable weather. Likewise, one day before a weather-compromised event, the system will warn all event participants.

Please refer to *Appendix A – Informal Specifications* to identify the points obtained from the specification document.

1.2. Goals

We think that MeteoCal has to provide these main features:

1. Creation of events;
2. Creator of events will always be able to edit or delete them;
3. The possibility to choose whether an event is an outdoor event (i.e. weather-dependent event) or not;
4. Weather forecast services with minimal user's contribution;
5. Correctly schedule events by avoiding overlaps between them (i.e. time consistency management);
6. Sharing proper information (i.e. public and private events and calendars are handled properly).

1.3. Domain properties

We suppose that these conditions hold in the analyzed world:

- A person is aware that weather forecasts can be provided only if a valid location is given;
- When a person wants to go to an event that depends on the weather conditions (and he/she wants to check these conditions), he/she almost never check the temperature (due to the fact that it's usually "perceived" with respect to the period of time), thus he/she only checks the "general" estimated condition (e.g. cloudy, clear, sunny etc.)
- A person is invited to an event only if he/she is actually desired;
- A person can attend an event only if he/she is the creator or he/she has been invited to it;
- A person that create an event is usually going to attend it.

1.4. Definitions

In our documentation, we are going to use some terms that could be confusing or misunderstood. For this reason, here follows an outline of such terms.

Customer
The person (or persons, or organization) who usually decide (in the meaning of demand) the requirements.

User
In general, the person, or persons, who operate or interact directly with the product (the user and the costumer are often not the same entity). Specifically, a person who is registered to the system and uses the MeteoCal services.

Guest
A person who is not registered to the system. He/she can only register to the system, to become a user.

Event
A planned public or private occasion. A name and a time slot identify it.

Time slot
Extend of time that represent, from start to end, a specific event. If a user has a time slot in his/her calendar, it means that he/she is busy in that time span.

Outdoor (event)
An event done, situated or conducted out of doors, it is implicit that it requires specific weather conditions.

Indoor (event)
An event done, situated or conducted within a building or under cover, thus it is not influenced by the weather conditions.

Creator (of an event)
The user that create the event.

Personal (event)
An event that has only its creator as an attendee.

Calendar (abstract)
In general, a chart showing the days, weeks and months of a particular year and showing the collocation (in time) of events. Specifically, an abstract entity with the only property of being private or public (allowing all the users to view the details of the public events and the time slots of the private ones) with respect to all the users but its owner.

Calendar (import/export)
In import or export operations, the calendar is the file that is – respectively – uploaded or downloaded. This file contains the information about all the events.

Owner (of a calendar)
The user that has the ownership of a calendar, i.e. the only one that is able to select its visibility and to export it. If a user is the owner or a calendar, then all the personal events that are in that calendar have that user as creator.

Personal (calendar)
During the import operation, the imported calendar is personal if its owner is the one that is currently importing it.

Original (calendar)
During the import operation, the original calendar is the one that is already in the system, i.e. not the imported one.

Personal (web page)
The web page of a logged user, which displays his/her calendar.

Details (of an event)
Comprehensive particulars of an event, including its name, time slot, eventual location, eventual attendees, eventual weather requirements, eventual description, eventual weather forecast.

Information (notification)
A simple notification with only informative purpose.

Invitation (notification)
An active notification inviting someone to attend an event.

Constraint (weather)
Limitation or restriction on the expected weather condition for a given event.

Here follows some acronyms, in alphabetical order:

API
Application Programming Interface

CAPTCHA
Completely Automated Public Turing test to tell Computers and Humans Apart

DB
Data Base

DD
Design Document

EE
Enterprise Edition

GUI
Graphical User Interface

RASD
(Software) Requirement Analysis and Specification Document

RC
Release Candidate

UML
Unified Modeling Language

1.5. Constraints

It is necessary for the correct functioning of the application that the server of the weather forecast services has a high availability.

1.6. Assumptions

Let us assume some statements about uncertain points of the specification document. If these assumptions are not correct, or any changes to them may occur, the contents of this document (RASD) would then have to change accordingly:

- An "outdoor event" is intended to be a weather dependent event and, vice versa, every weather dependent event is an "outdoor event";
- In case of email stealing, a user is able to get his/her email account back;
- A user can import a calendar that is not owned by him;
- When a user import a calendar, only personal events are imported;
- If importing a calendar two events overlap, by default the event of the original calendar is kept and the one of the imported calendar is discarded (*other personalized behaviors might be realized*);
- When a user import a calendar that is not owned by him, the imported events are renamed (e.g. showing the name of the original owner) and the user himself becomes the new owner of those events;
- The system does not reschedule the weather-compromised event automatically: it simply inform the creator about the issue (*personalized behaviors might be implemented*) and it will propose the closest well-matched day (if any);
- A system administrator (in the meaning of a privileged user) is not required (*a moderator might be implemented*);
- A user can receive invitations for overlapped events, but he/she will not be able to accept it unless he/she is going to refuse the previously confirmed event.

n.b. **the *sentences in italic* refer to advanced functionalities whose actual implementation is not guaranteed. Anyway, their eventual implementation will not change the overall architecture, behavior, or user perception (similar considerations will be also done in *Subsection 3.4.4 – Computer Security considerations*).**

Please refer to *Appendix B – Revision history* for variations about this document.

1.7. Proposed system

We propose a web application that will help users to organize their events with the described services.

Please refer to *Chapter 3 - Requirements* for further information about the services that will be implemented.

Non-registered users (i.e. guests) will only be able to register themselves to the system: the registration (i.e. sign up) is mandatory to use the web application.

Please refer to *Subsection 3.4.1 – User interface* for further information about the interaction with the users.

1.8. Identifying stakeholder

The professor, who gave us the task to build this project, is our costumer (i.e. pseudo-financial stakeholder).

We are going to be both the developers and users of MeteoCal.

Other people interested in this web application, when registered to the system, will also become users.

1.9. Other considerations about the system

As mentioned before (in *Section 1.6 – Assumptions*) some functionalities might or might not be implemented. This is due to time limitations. We think it is important to set some priorities that, if correctly working (along with the whole requirements) in a RC version of the software, will give us the possibility to implement the so-called “advanced functionalities”.

Usability and simplicity

We want the user to be able to use the application easily and quickly. This will also require an appropriate GUI.

Stability and privacy

The application must correctly handle (in the meaning of reliability) at least the fundamental functionalities and all the services about sensitive information and privacy settings.

2. Actors Identifying

We have identified four actors: we have defined two of them (guest and user) as active actors (i.e. that can perform actions, and our system will react), while the other two (weather forecast service and email service) as passive actors (i.e. that will only answer to the requests of our system).

Guest

The guest is a person that has not registered to the system yet. He/she can only register to the system.

User

The user is a person that has already registered to the system.

Weather forecast service

The weather forecast service is the server that will provide the requested information about weather forecast. It only serves the required information through the API.

Email service

The email service is an external email server that will send non-reply-able emails to the users.

3. Requirements

In the scenarios where the domain properties (see *Section 1.3 – Domain properties*) holds, from the written goals (see *Section 1.2 – Goals*) we can arise our requirements:

1. **Creation of events**:
    - The system will allow the user to create a new event (only the name of the event and a time slot are required);
2. **Creator of events will always be able to edit or delete them**:
    - The system will allow the creator of an event to edit every field of it, at any time;
    - The system will allow the creator of an event to delete it, at any time.
3. **The possibility to choose whether an event is a weather-dependent event or not**:
    - The system will allow the user to do not directly specify a weather constraint, in this situation the system will assume the event as a non-weather-dependent event;
    - The system will allow the user to directly specify that the event is an indoor event (i.e. non-weather-dependent);
    - The system will allow the user to directly specify the weather constraint (e.g. requires sun, requires no precipitation, requires snow, etc.).
4. **Weather forecast services with minimal user's contribution**:
    - The system will require the user to put only the location (along with an event name and a time slot) for it to provide a simply informative weather forecast;
    - The system will require the user to user to put the location and a weather constraint, to provide the generation of weather alerts.
5. **Correctly schedule events by avoiding overlaps between them**:
    - The system will not allow the user to create a new event if it overlaps a time slot that is in his/her calendar;
    - The system will not allow the user to accept an invitation for an event if it overlaps a time slot that is in his/her calendar;
    - The system will not allow the creator of an event to edit its time slot if it overlaps another time slot that is in his/her calendar;
    - The system will revoke the event from the calendar of a user and will re-send him an automatic invitation, if the creator of the event has re-scheduled it and it conflicts with another event in the user's calendar.
6. **Sharing proper information**.
    - The system will allow the user to hide his/her calendar by making it private (otherwise it will be public);
    - The system will allow the user to hide the details of his/her event by making it private (only the time slot will be visible, if the calendar is public);
    - The system will allow only the owner of a calendar to export it.

3.1. Functional requirements

Some functional requirements concerning each defined actor are:

- A guest can:
    - Register;
- A user can:
    - Sign in;
    - Sign out;
    - Reset a forgotten password;
    - Recover the user name;
    - Modify the privacy of the calendar;
    - Search for an user of the system (by first name, last name or user name);
    - View the calendar of an user with a public calendar;
    - View the time slots of any event in a public calendar;
    - View the details of any public event;
    - Accept or decline (or ignore) the invitations to events;
    - Import or export his/her calendar;
    - Create an event;
    - Edit or delete his/her events (if he/she is the creator of them);
    - When creating or editing an event, give it a location and a description;
    - When creating or editing an event, invite another user that has no other events in the same time slot;
    - When editing an event, re-invite a user that has previously declined the invitation;
    - When creating or editing an event, mark it as public or private;
    - When creating or editing an event, mark it as an indoor or outdoor event (i.e. weather-dependent event);
    - When creating or editing an event as outdoor, choose if it requires sun, requires no precipitation, requires snow, etc.;
    - Decline the invitation of an event that was previously accepted.

3.2. Nonfunctional requirements

3.4.1. User interface

The GUI of the application is thought to be used via web. We have decided to be quite specific in its description, because we believe that the requirements are strongly related to it.

n.b. **Consider the contents of the showed GUI as concepts, with the only purpose to understand the requirements: during the developing process it may vary, but the basic ideas will remain the same.**

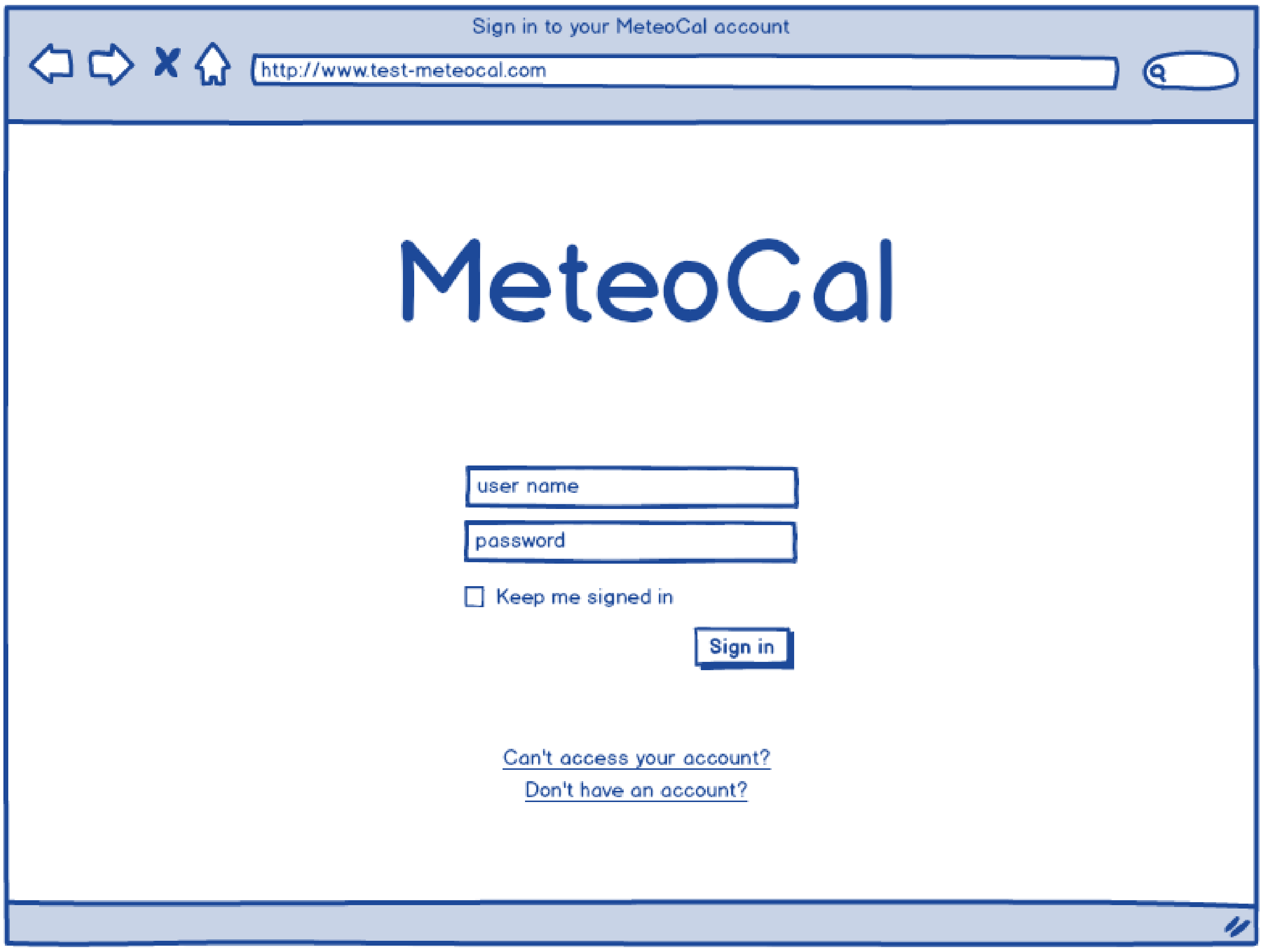

Sign in to your MeteoCal account
http://www.test-meteocal.com
MeteoCal
user name
password
Keep me signed in
Sign in
Can't access your account?
Don't have an account?

Figure 1 – Non-logged users will only be able to sign in by writing their user name and password. Guests will be able to register by clicking on "Don't have an account?". Log in problems can be solved by clicking on "Can't access your account?".

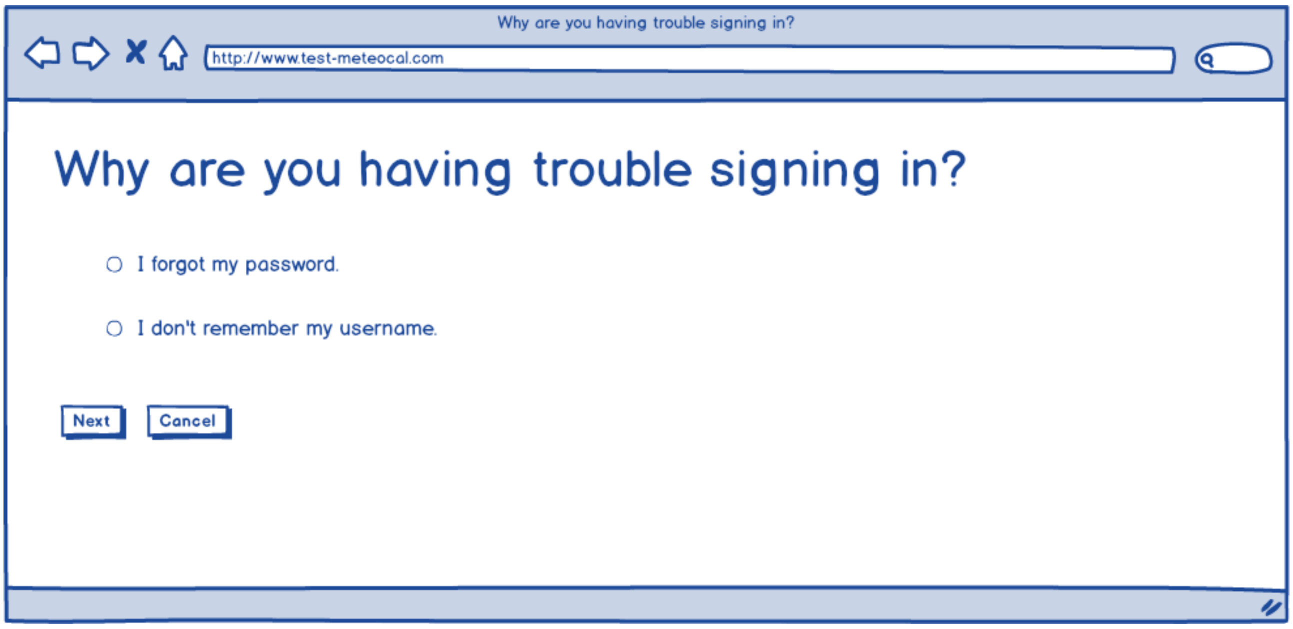

Why are you having trouble signing in?
http://www.test-meteocal.com
Why are you having trouble signing in?
I forgot my password.
I don't remember my username.
Next
Cancel

Figure 2 – When the user clicks on "Can't access your account?" (see Figure 1) to solve his/her log in problems, he/she can start a password reset procedure (its actual implementation is not yet defined) by clicking on "I forgot my password", otherwise he/she ca can ask to send his/her user name (in plaintext) at the registration email, by selecting "I don't remember my user name".

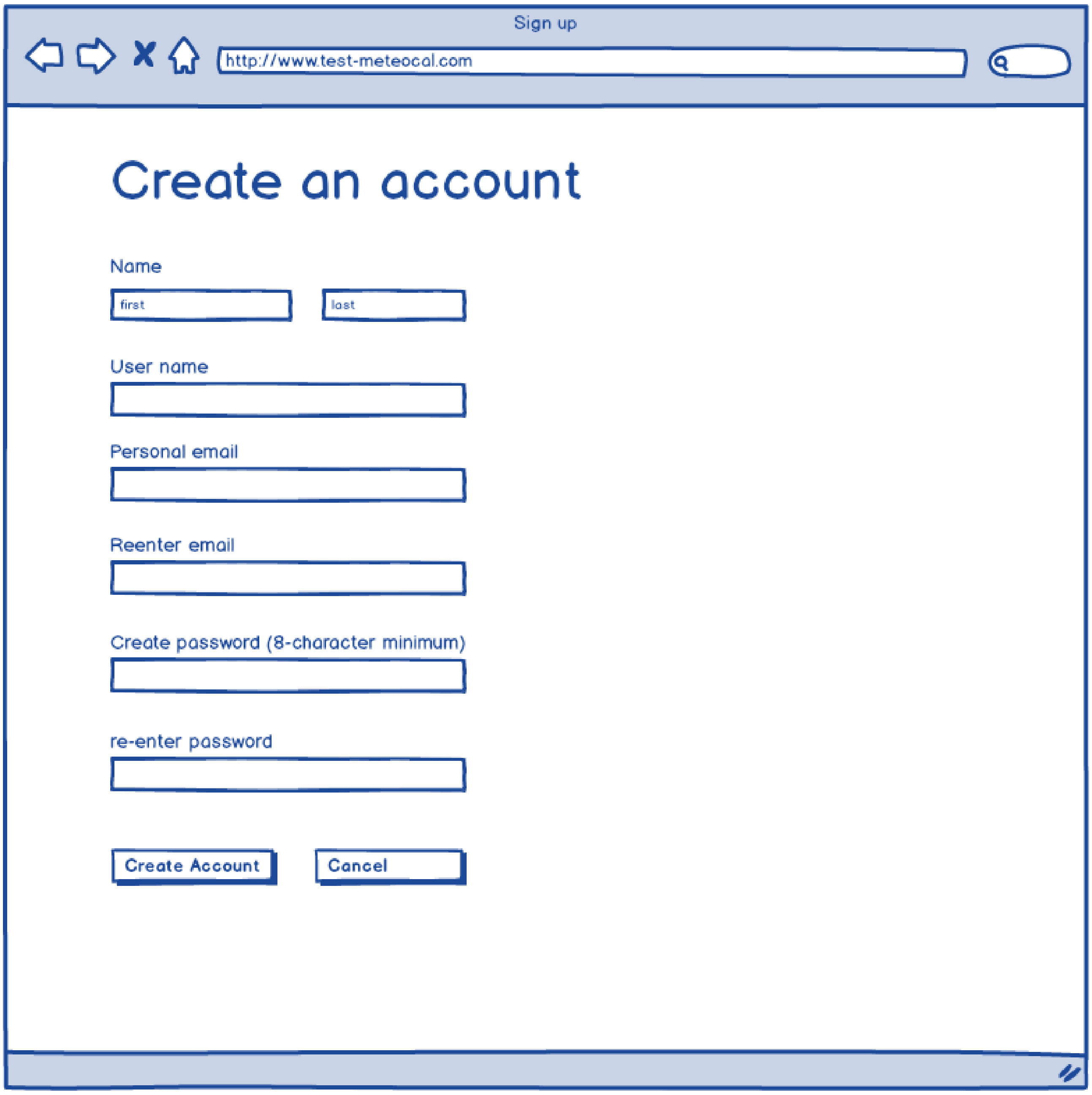


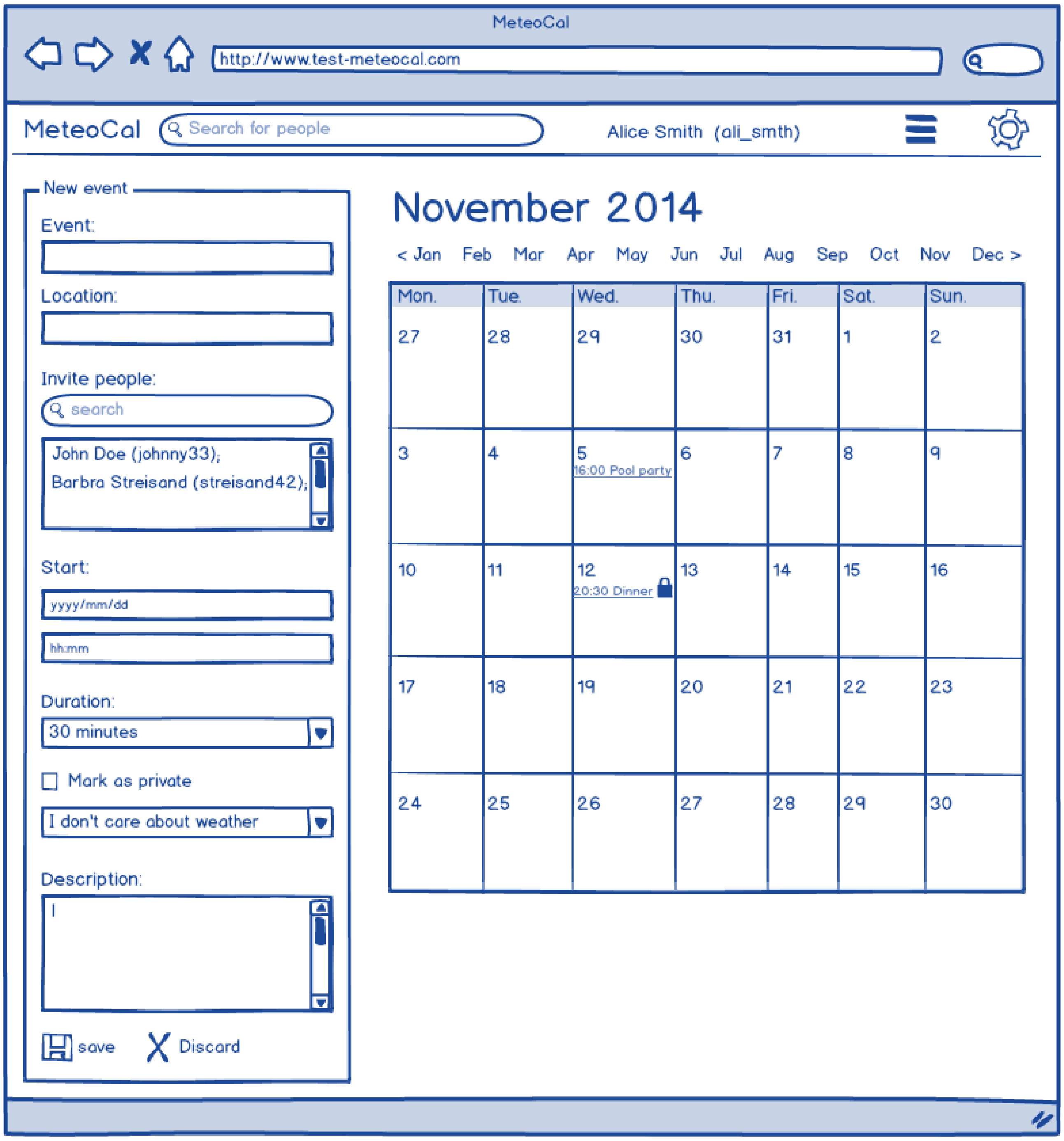
Figure 3 – When a guest clicks on "Don't have an account?" a registration form is displayed. CAPTCHA verification may be showed (please refer to Subsection 3.4.4 – Computer Security considerations for further information about it).

When a user is logged, on the upper side of the screen he/she will always see a bar. This bar will display the name of the application (MeteoCal), a search panel, the name and the user name, an icon for the notifications and an icon for other functionalities.

If the user clicks on "MeteoCal" or his/her name and user name, he/she will return to the first page with the calendar displaying the current month. The search panel will be used to search for other users.

The notifications icon will displays the notifications, which will also include the invitations at the events. From the notifications panel, the user will be able to click on the event name to see its details, accept the invitation or decline it. When the user accepts or declines the invitation, its notification will disappear.

The tools icon will allow to refresh the page, to sign out, to change the state of the calendar (from public to private or, vice versa, from private to public), to export the calendar or to import one. In addition, all the possible (but not ensured) advanced functionalities will be implemented in this menu (please refer to *Section 1.9. – Other considerations about the system* for further information about it).



John Doe (johnny33);
Barbra Streisand (streisand42);

Duration: 30 minutes

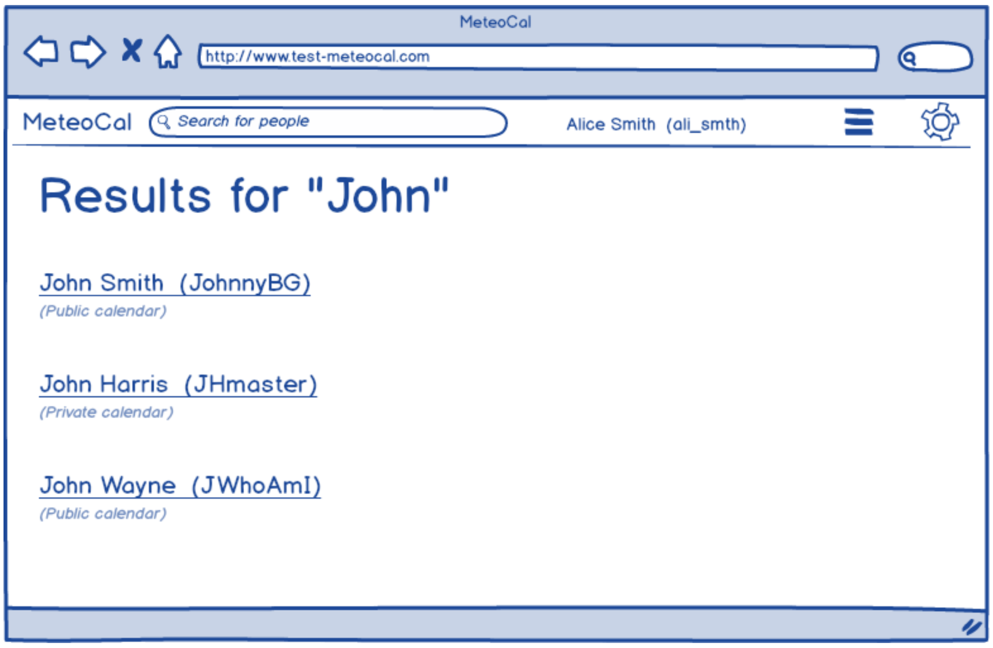
☐ Mark as private

I don't care about weather

Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
27	28	29	30	31	1	2
3	4	5 16:00 Pool party	6	7	8	9
10	11	12 20:30 Dinner 🔒	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Figure 4 – The main page seen by logged user displays the calendar with the events. A fixed maximum number of events will be displayed for each day. By clicking on the number of the day, all the events for that day will be displayed, this overview will also allow to move to the previous or the next day. Events can be created using the left-side panel.

The main page seen by logged user displays the calendar with the events. A fixed maximum number of events will be displayed for each day. By clicking on the number of the day, all the events for that day will be displayed, this overview will also allow to move to the previous or the next day. Events can be created using the left-side panel.


MeteoCal

http://www.test-meteocal.com

MeteoCal | Search for people | Alice Smith (ali_smth)

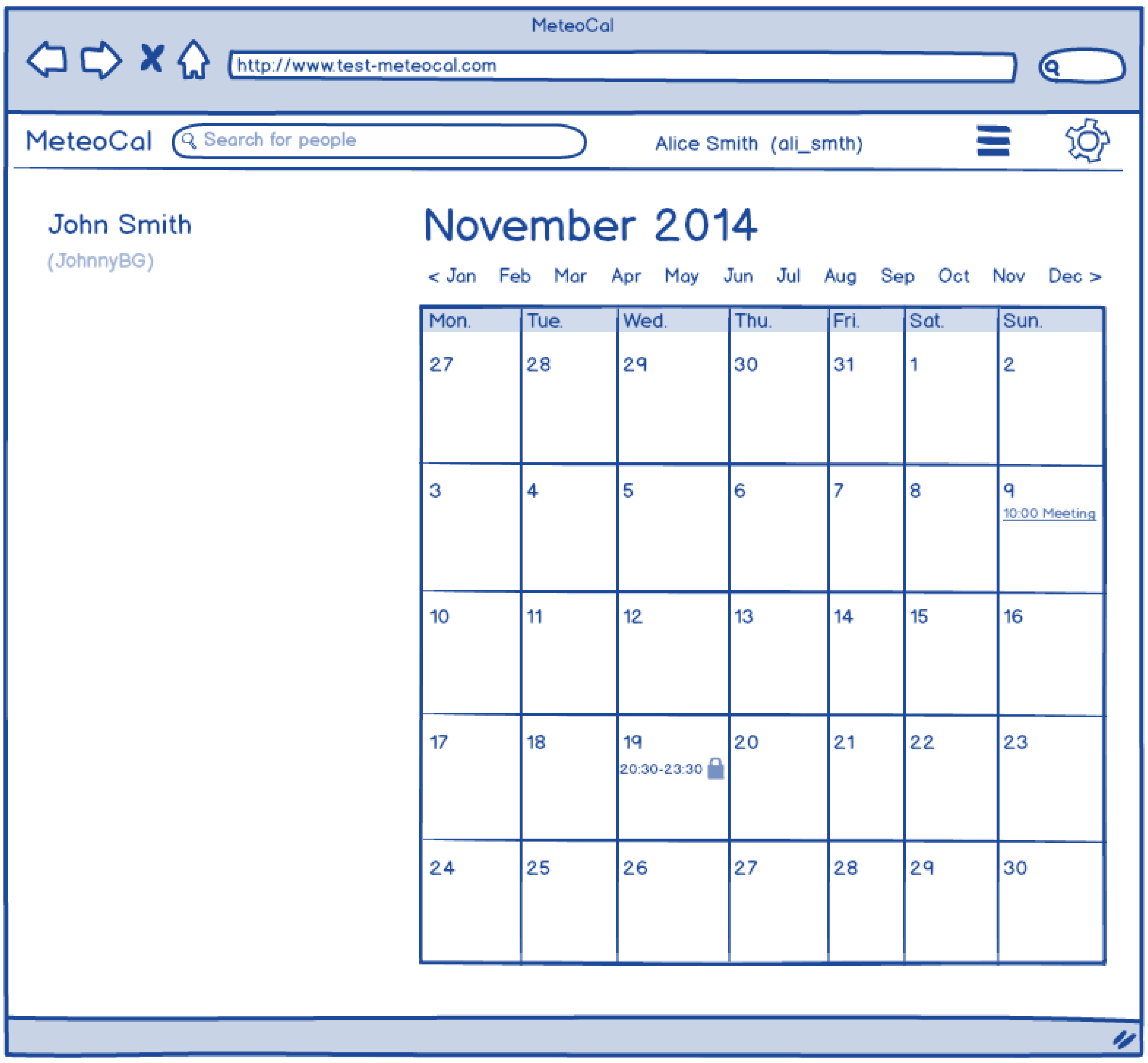
Results for "John"

John Smith (JohnnyBG)
(Public calendar)

John Harris (JHmaster)
(Private calendar)

John Wayne (JWhoAmI)
(Public calendar)

Figure 5 – The users can search for any other user information by writing his/her name or user name in the search panel.


MeteoCal

http://www.test-meteocal.com

MeteoCal | Search for people | Alice Smith (ali_smth)

John Smith
(JohnnyBG)

November 2014

< Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec >

Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
27	28	29	30	31	1	2
3	4	5	6	7	8	9 10:00 Meeting
10	11	12	13	14	15	16
17	18	19 20:30-23:30 🔒	20	21	22	23
24	25	26	27	28	29	30

Figure 6 – When a user opens the profile of another user who has his/her calendar public, he/she will see all his/her public events, but the private events will only be displayed as a time slot with a closed-lock icon.

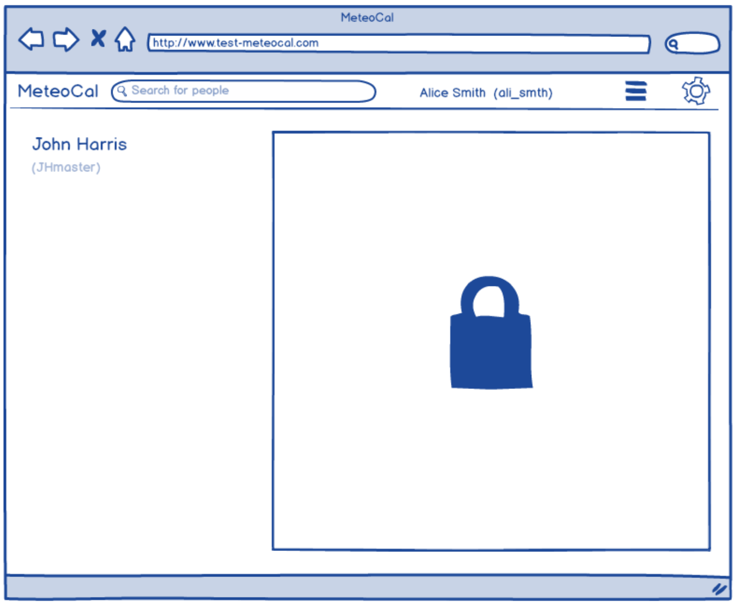
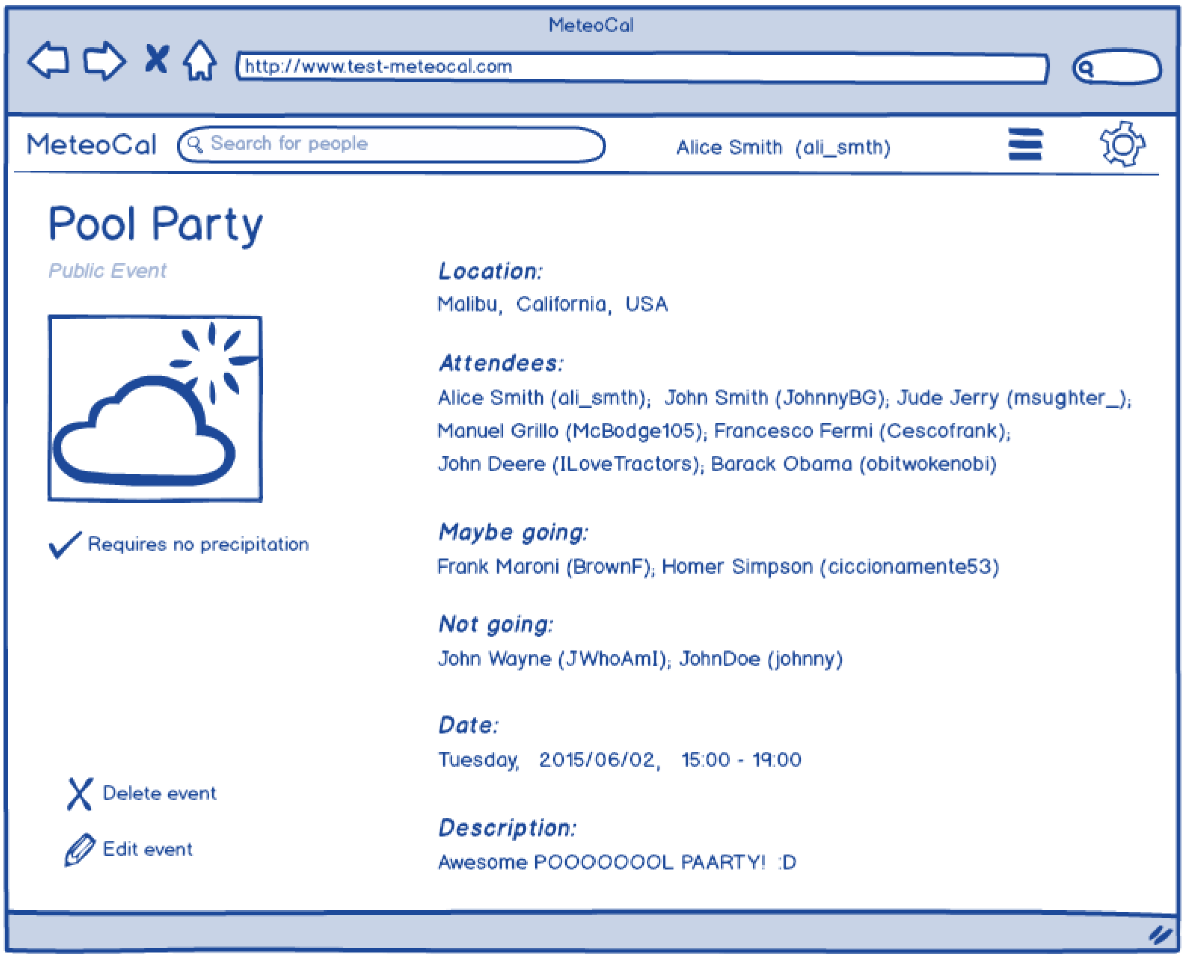


Figure 7 – When a user opens the profile of another user who has his/her calendar private, no information will be available.



Pool Party

Public Event

✓ Requires no precipitation

X Delete event

Edit event

Location:
Malibu, California, USA

Attendees:
Alice Smith (ali_smth); John Smith (JohnnyBG); Jude Jerry (msughter_); Manuel Grillo (McBodge105); Francesco Fermi (Cescofrank); John Deere (ILoveTractors); Barack Obama (obitwokenobi)

Maybe going:
Frank Maroni (BrownF); Homer Simpson (ciccionamente53)

Not going:
John Wayne (JWhoAmI); JohnDoe (johnny)

Date:
Tuesday, 2015/06/02, 15:00 - 19:00

Description:
Awesome POOOOOOOL PAARTY! :D

Figure 8 – When the creator clicks on the name of the event on the calendar (or on the daily overview page), all the information will be displayed. Users who has accepted the invitation will be displayed under "Attendees", users that have not answered will be under "Maybe going" and user that have declined will be under "Not going". In addition, the creator will be able to cancel the event or modify it. If he/she clicks on "Edit event", to modify it, the first page will be displayed, with every field filled. The only removed information will be the not going people (notifications will only be sent to new attendees or re-invited attendees, who have previously declined the invitation). If a user who is not the creator browse the page, he/she will not see the "Not going" field, the "Delete event" and "Edit event" buttons, but he/she will see the creator's name and, if the viewer has previously received and accepted the invitation, a "Decline invitation" button.

3.4.2. Documentation

Here follows the documentation that will be provided during the development of the project:

- **RASD** (Requirement Analysis and Specification Document): to understand the given problem and to analyze in a detailed way what are our goals. Defining requirements and specifications, we describe our proposal to achieve the identified goals.
- **DD** (Design Document): to define the actual structure of the web application that is going to be implemented and its tiers.
- **JavaDoc and comments** in the source code: with the purpose of making the source code easier to understand (for those who wants to develop the platform or do maintenance on it). *The comprehensiveness of the details of the JavaDoc and annotations cannot be known at this point of the project*.
- **Installation manual:** a guide to install the server application;
- **User manual:** a guide to use the application, via the web browser.

3.4.3. Architectural considerations

We will use the Java EE 7 platform with a database, where all the information of the system will be stored (only the recurrently used information will be kept at "run-time").

An internet connection and a recent web browser (Internet Explorer 11 and Google Chrome 38 will be tested) are required to access and use the service and features of MeteoCal.

3.4.4. Computer Security considerations

Alongside what mentioned in the previous considerations about privacy (see Section 1.9. – Other considerations about the system) and the goal of sharing proper information (see Chapter 2 – Requirements), we would like to implement some other features whose implementation cannot be guaranteed (because it is not considered a priority):

- Filtering of user input (to prevent SQL-injections);
- Use of CAPTCHA challenge-response tests in seldom used forms;
- Not keep the users' passwords in plaintext in the DB;
- Use an appropriate password-recovery procedure.

4. Specification

The following are some specifications for the proposed web application (please refer to *Appendix A – Informal specification* for the informal specification, directly derived from the costumer request):

- To use the application, a guess will need to register;
- Every user will be able to change the privacy settings of his/her calendar;
- Users will be able to create and manage events on their calendars;
- Events can be edited or deleted by a user if and only if the user is the creator of the event;
- Events will not be allowed to overlap;
- User can invite other users to attend events they have created;
- Invited users may accept or refuse invitation to and event;
- Every invitation will have a particular state:
    - Pending: the invited user has not accepted nor refused the invitation;
    - Accepted: the user has accepted the invitation;
    - Refused: the user has refused the invitation.
- Once a user refuses an invitation, the user will not receive any notification from that event;
- A user who accepts an invitation will receive notifications about the event.
- Events can be either public or private:
    - Public: Details of such event will be available to the whole users;
    - Private: Details will be only available to invited users.
- When creating an outdoor event, the user can specify a weather constraint.
- Notifications will be sent locally or via email in the following situations:
    - When a user is invited to an event, the invited users will be notified;
    - When users accept or refuse invitations, the creator of the event will be notified;
    - When an event is modified or deleted all the concerned users will be duly notified;
    - When the weather condition of the event changes;
    - When a user has been uninvited to an event;
    - When a user accepts an invitation that will cause an overlap, the user is alerted.
- Users can export their calendar to other users, only the public events will be importable by the other users;
- Weather information associated to the events will be updated periodically.

5. Scenarios Identifying

Here are some possible scenarios of MeteoCal:

- A person finds our system and decides to register and create an event;
- A person uses the import/export features of MeteoCal;
- A person invites other people to his/her event;
- A person quickly create a personal event.

Please also refer to *Subsection 3.2.1. – User interface* to better understand the user perception of the web application and the details of the actions that the actors in the scenarios do.

A person finds MeteoCal and decide to register and create an event

Alice is exhausted of having to waste time to manage hiking events. She search the internet for a solution and she find the MeteoCal website that offers a digital calendar which can add to the events updated weather forecast information and assist if the weather is not favorable.

Alice decides to register by filling out the registration form (see *Figure 3*) and creates her first event (see *Figure 4*) for the weekend and the system adds information about the weather – Alice is satisfied.

One day before the event, the system informs Alice that it is likely that it will rain, but the day after will be sunny. Consequently, Alice decides to move to move the hiking one day forward.

A person uses the import/export features of MeteoCal

A company has just hired Bob. Carol works in his same department, but she has been hired long time before. She has filled her calendar with business appointments (personal events, private because of the policies of the company) that also Bob has to attend. Carol wants to help Bob, but since it would be a waste of time to invite him to every single event, she decides to:

- Export her calendar;
- Delete a very confidential event;
- Export the new calendar giving it to Bob;
- Re-import the first calendar getting her confidential event back.

Bob import the calendar received from Carol and now has everything he needs to attend the working commitment. Bob is also able to discriminate some of Carol's event not useful for him, since her name is provided in the name of the event.

A person invites other people to his/her event

Eve is a new MeteoCal user and she has used the web application only for personal purpose. She would like to organize a party, thus she decides to try to create a public event and invite other people who already are on MeteoCal. Frank receive the invitation and, having no other engagements for that day, he accepts to attend the party.

A person quickly create a personal event

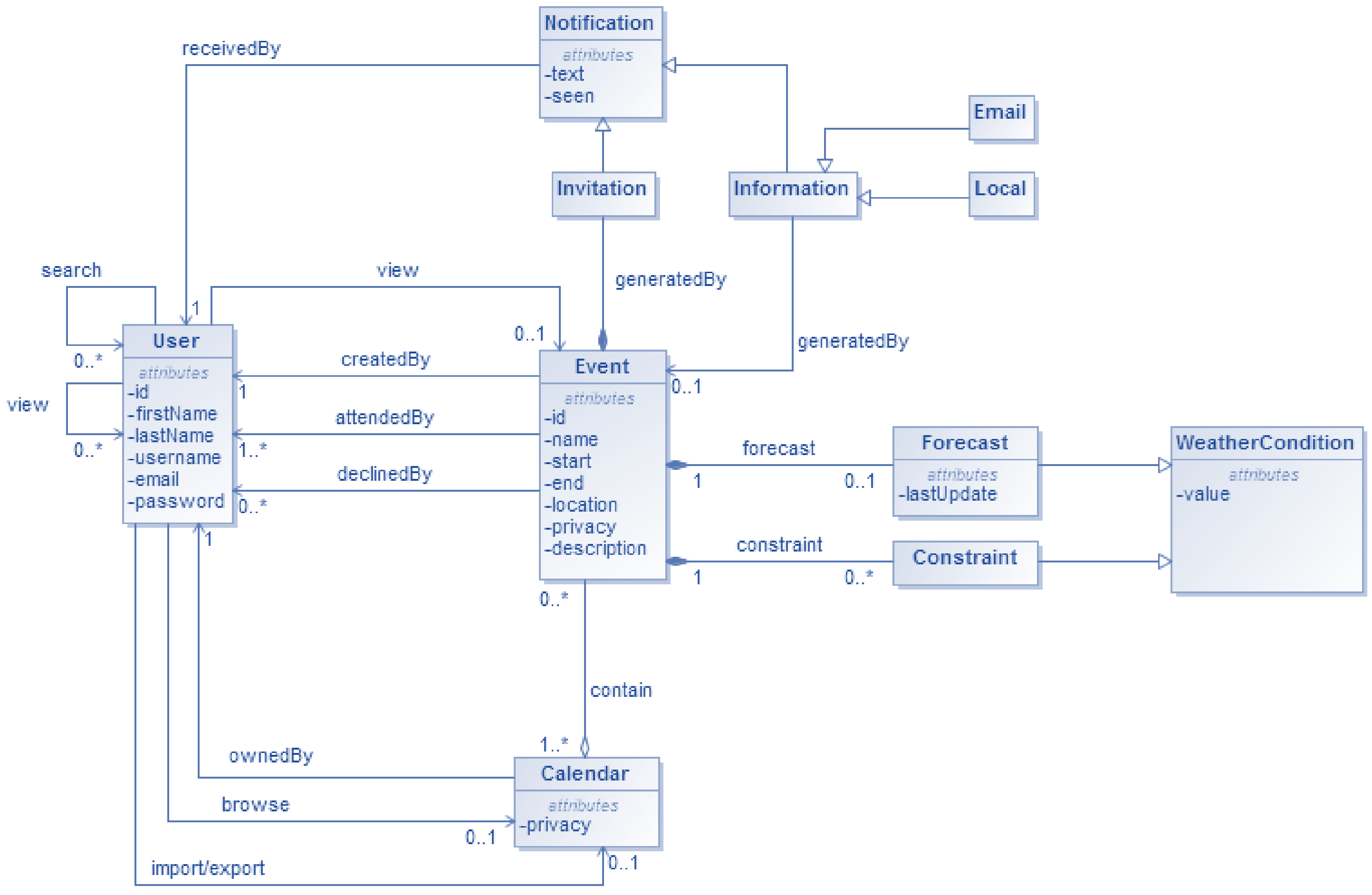
Giovanni, an important academic full of commitments, has to hurriedly remind himself that the next day has to go to his/her grandma. He create the event providing this information:

- Event name: Dinner at grandma's house;
- Time slot: 2014/11/8 19:00-22:00.

He saves the event in his/her calendar. Done.

6. UML models

6.1. Class diagram



6.2. Use case diagrams

To provide functional modeling, we provide the subsequent use case diagrams:

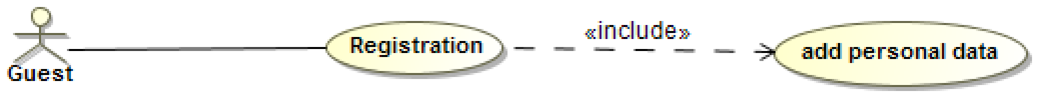


Figure 10 – This trivial use diagram shows the only action that a visitor of the web site (i.e. guest) can perform (sign up).

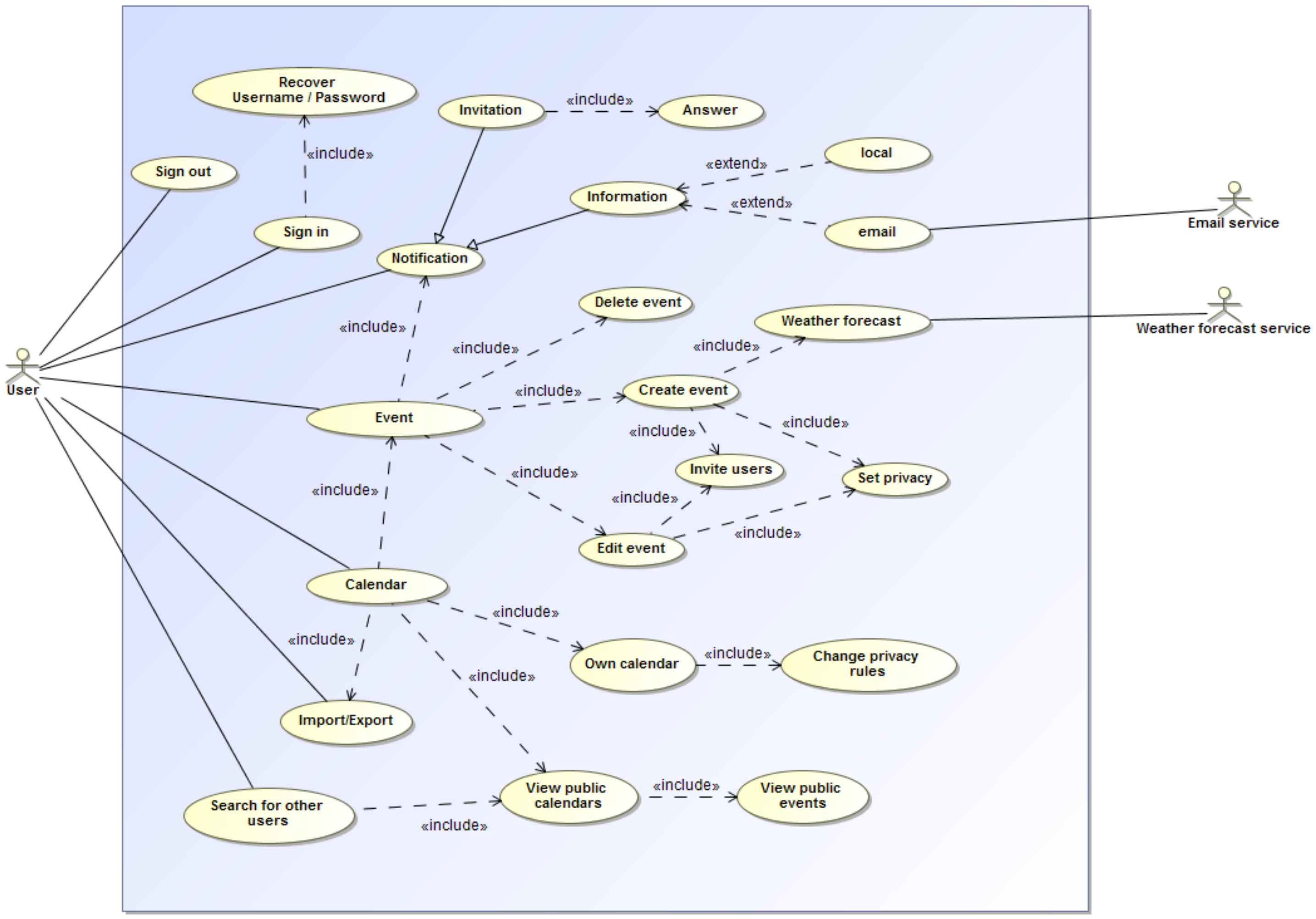


Figure 11 – A user has already registered an account to the system, so he/she can sign in, sign out, see notifications, answer an invitation, and manage events. In addition, he/she can looks for other users, sees users' public calendars and events, and import/export his/her calendar. In the use case, the system involves two services: the first one provide weather forecast information, while the second one is used to send the emails to the interested users.

6.3. Use cases description

Here follow the lists of steps defining the interactions between some actors and the components of the system. We have specified, in the box about the exceptions, the transition (between two events) wherein the exception is raised.

6.3.1. Registration

Name	Registration (i.e. sign up)
Actors	Guest
Entry conditions	- No user has to be logged; - The person has not to be registered.
Flow of events	1. The user opens the homepage of the web application; 2. The system load the login page; 3. The user clicks on the "Don't you have an account?" link; 4. The system load the page that contains the registration form; 5. The user fill the form and clicks on "Create account"; 6. The system check the data; 7. The user is now registered.
Exit conditions	The system adds the user's information to the database.
Exceptions	- The user does not provide the required information *(5-6)* – an error message is shown; - User's email or user name are already in the system *(6-7)* – an error message is shown.

See *Figure 12*.

6.3.2. Log in and log out

Name	Log in (i.e. sign in)
Actors	User
Entry conditions	- No user has to be logged; - The user already own an account on the platform.
Flow of events	1. The user opens the homepage of the web application; 2. The system load the login page; 3. The user provides username and password; 4. The user clicks on the "Sign in" button; 5. The system checks the user's credential; 6. The system presents the personal page.
Exit conditions	No exit condition.
Exceptions	The provided credentials are wrong (4-5) – an error message is shown.

See *Figure 13*.

Name	Log out (i.e. sign out)
Actors	User
Entry conditions	The user has to be logged.
Flow of events	1. The user clicks on the tools button; 2. The user clicks on the sign out button; 3. The system signs out the user; 4. The system load the homepage.
Exit conditions	No exit condition.
Exceptions	The provided credentials are wrong (4-5) – an error message is shown.

See *Figure 14*.

6.3.3. Event creation and event deletion

Name	Event creation
Actors	User, weather forecast service
Entry conditions	- The user has to be logged - The user already own an account on the platform.
Flow of events	1. The user fills the form in the left panel; 2. The user clicks on the "Save" button; 3. The system check the filled fields; 4. The system create the event; 5. The system get the weather forecast for the event; 6. The weather forecast server give the information to the server; 7. The system add the event to the user's calendar.
Exit conditions	- The system add the event to the user's calendar; - The system refresh the page.
Exceptions	- Not all the required information are provided (2-3) – an error message is shown; - The event overlaps with an other one (2-3) – an error message is shown; - The weather forecast server is not available (5-6) – the system skips the callback.

See *Figure 15*.

Name	Event deletion
Actors	User
Entry conditions	- The user has to be logged; - The user has created at least one event; - The user has to be on his/her personal page.
Flow of events	1. The user clicks on the number of the day, shown in the calendar; 2. The system load the list of the events; 3. The user clicks on an event created by him; 4. The system load the page with the details of the event; 5. The user clicks on the "Delete" button; 6. The system delete the event, notify all the former attendee and remove the active invitations.
Exit conditions	The event is deleted, all the former attendee are notified and the active notifications are removed.
Exceptions	No exceptions.

See *Figure 16*.

6.3.4. Answer an invitation

Name	Answer an invitation (accept/decline/ignore)
Actors	User
Entry conditions	- The user has to be logged; - The user has been invited to an event; - The user realize that there is a notification.
Flow of events	1. The user clicks on the notification button; 2. The system loads the list of the notifications; 3. The user answers the invitation by clicking on the button “Accept” or “Decline”; 4. The system adds the user to the “not going” list (case: decline); 5. The system adds the user to the “attendee” list (case: accept); 6. The system checks if there is overlapping (case: accept); 7. The system adds the event to the calendar (case: accept).
Exit conditions	- The event is added to the calendar (case: accept). - The invitation is removed (case: declined).
Exceptions	There is overlapping (5-6) – an error message is shown.

See *Figure 17*.

6.3.5. Reschedule for bad weather condition

Name	An outdoor event has to be re-scheduled due to bad weather condition
Actors	User, weather forecast service, email service
Entry conditions	- The user is the creator of a weather-dependent event; - The user has received an email warning about the issue.
Flow of events	1. The user log in; 2. The system locally notifies the user about the bad weather condition; 3. The user reads the notification; 4. The user clicks on the notification; 5. The system loads the page with the details of the event; 6. The user clicks on “Edit”; 7. The user modifies the date of the event; 8. The user saves the modification; 9. The system check the modification; 10. The system update the calendar.
Exit conditions	Event shifted or deleted.
Exceptions	There is overlapping (8-9) – an error message is shown.

See *Figure 18*.

6.3.6. Import or export of a calendar

Name	Export a calendar
Actors	User
Entry conditions	- The user is signed in; - The user has at least one event in his/her calendar.
Flow of events	1. The user clicks on the tools button; 2. The user clicks on the "Export calendar" label; 3. The system generates the file that has to be exported 4. The system sends the file to the user's browser; 5. The user downloads the file
Exit conditions	No exit conditions.
Exceptions	There are no personal events (2-3) – an error message is shown.

See *Figure 19*.

Name	Import a calendar
Actors	User
Entry conditions	The user is signed in.
Flow of events	1. The user clicks on the tools button; 2. The user clicks on the "Import calendar" label; 3. The system asks for the file that has to be imported; 4. The user selects the file; 5. The user's browser allows the upload of the file; 6. The system receives the file; 7. The system reads the file; 8. The system merges all the personal events in the file with the ones in the calendar (if two events overlap, it keeps the one of the current calendar).
Exit conditions	No exit conditions.
Exceptions	The file is not recognized by the system (6-7) – an error message is shown.

See *Figure 20*.

6.4. Sequence diagrams

6.4.1. Registration

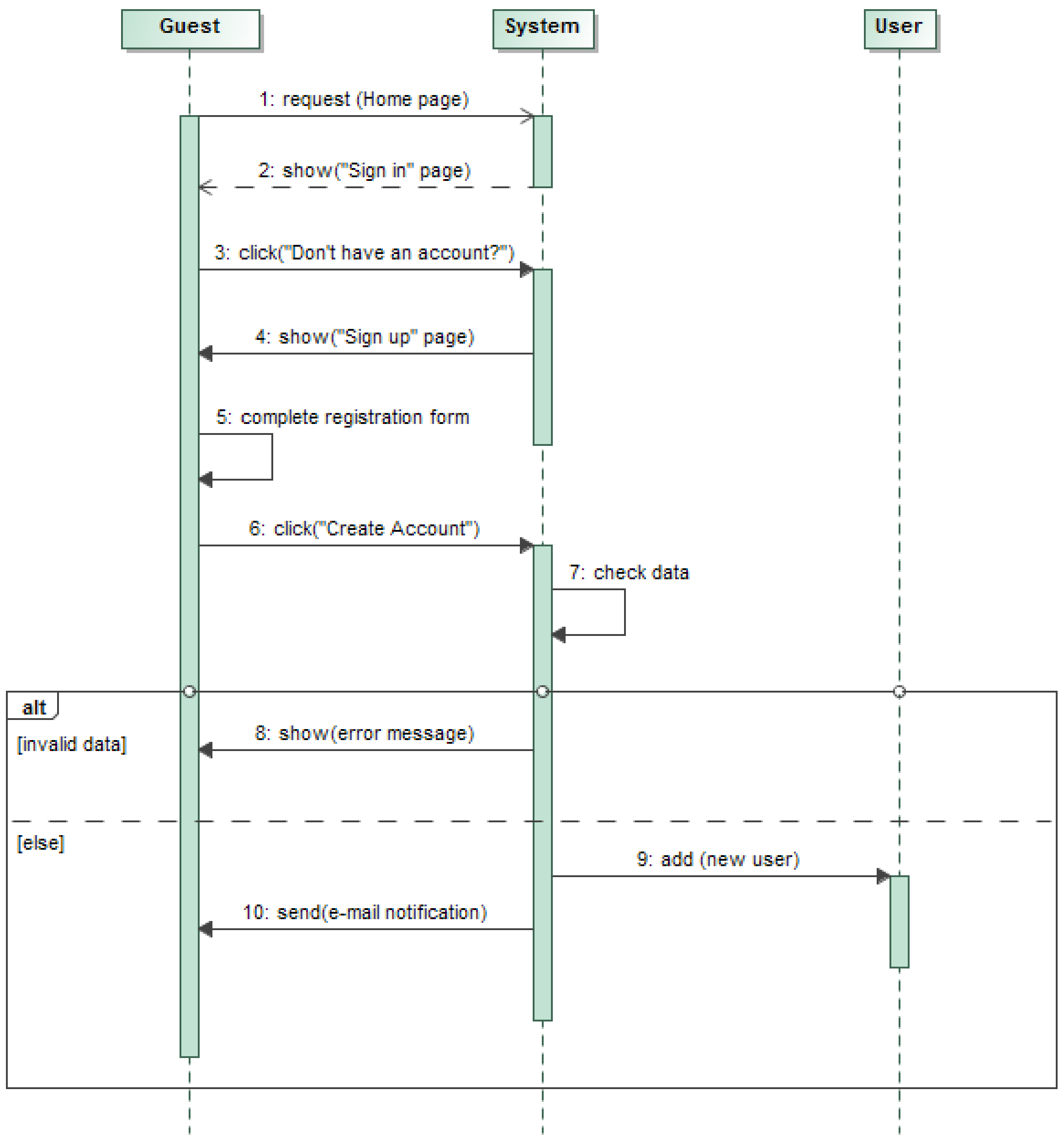


Figure 12 – Sequence diagram of the interactions involved in the registration process. The system will also check the data accordingly to the adopted security policies (e.g. to prevent SQL injections – please refer to Subsection 3.4.4. – Computer Security considerations for considerations about it).

6.4.2. Log in and log out

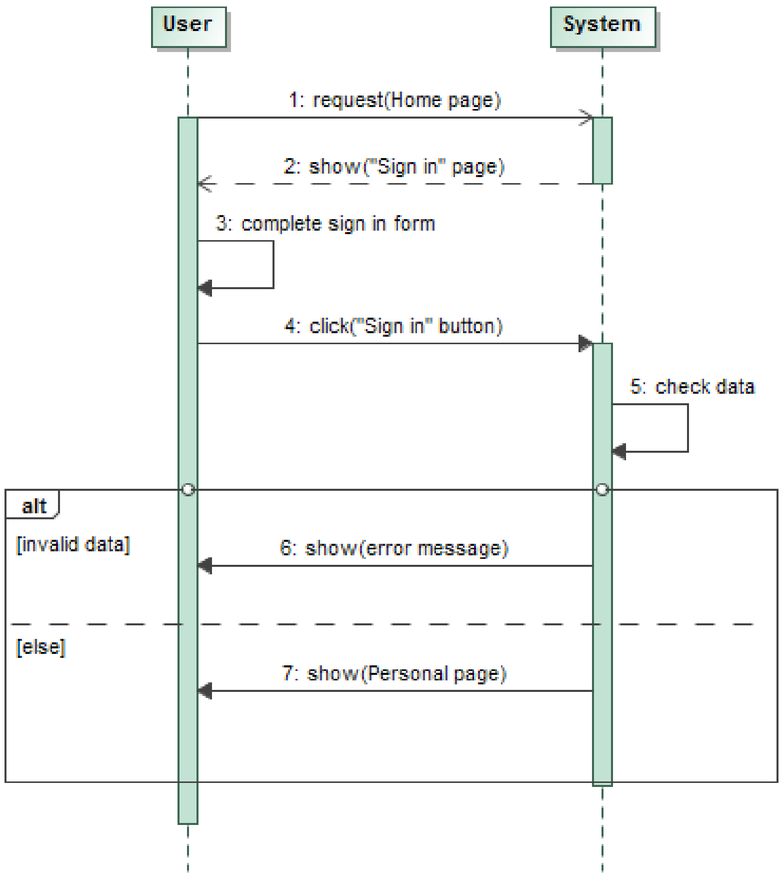


Figure 13 – Sequence diagram of the interactions involved in the sign in process. The system will also check the data accordingly to the adopted security policies (e.g. to prevent SQL injections – please refer to Subsection 3.4.4. – Computer Security considerations for considerations about it).

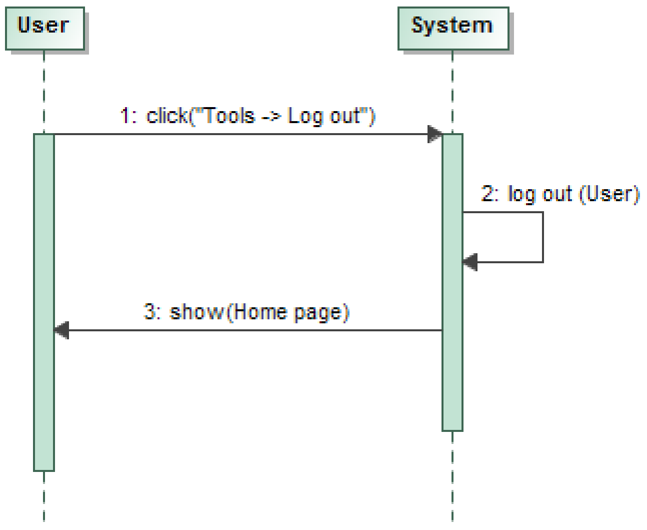


Figure 14 – Sequence diagram of the interactions involved in the sign out process.

6.4.3. Event creation and event deletion

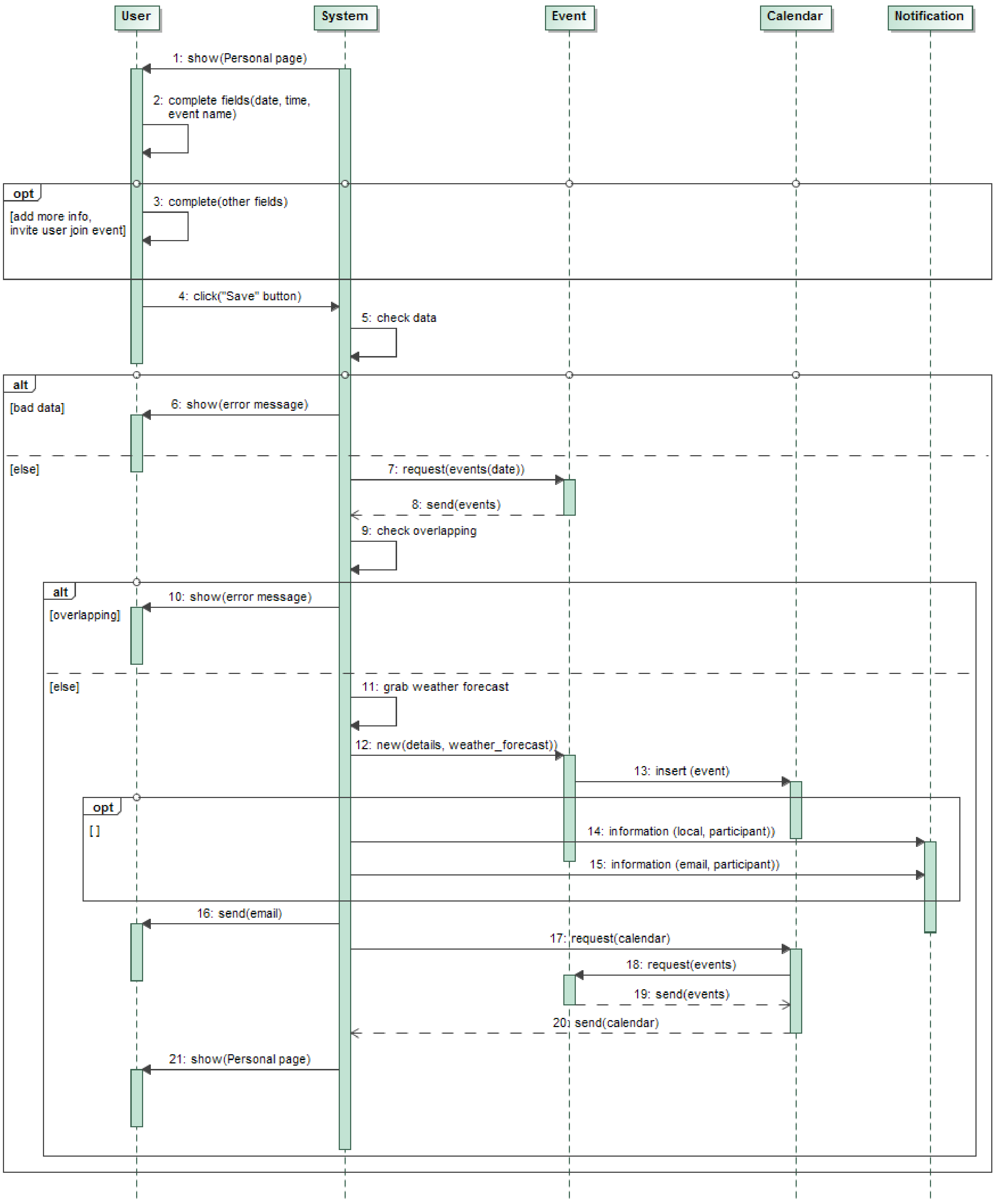


Figure 16 – Sequence diagram of the interactions involved in the event deletion process. The "Event details" page is loaded in both cases, nonetheless, only the creator will be able to see the edit and delete buttons (see Figure 8).

6.4.4. Answer an invitation

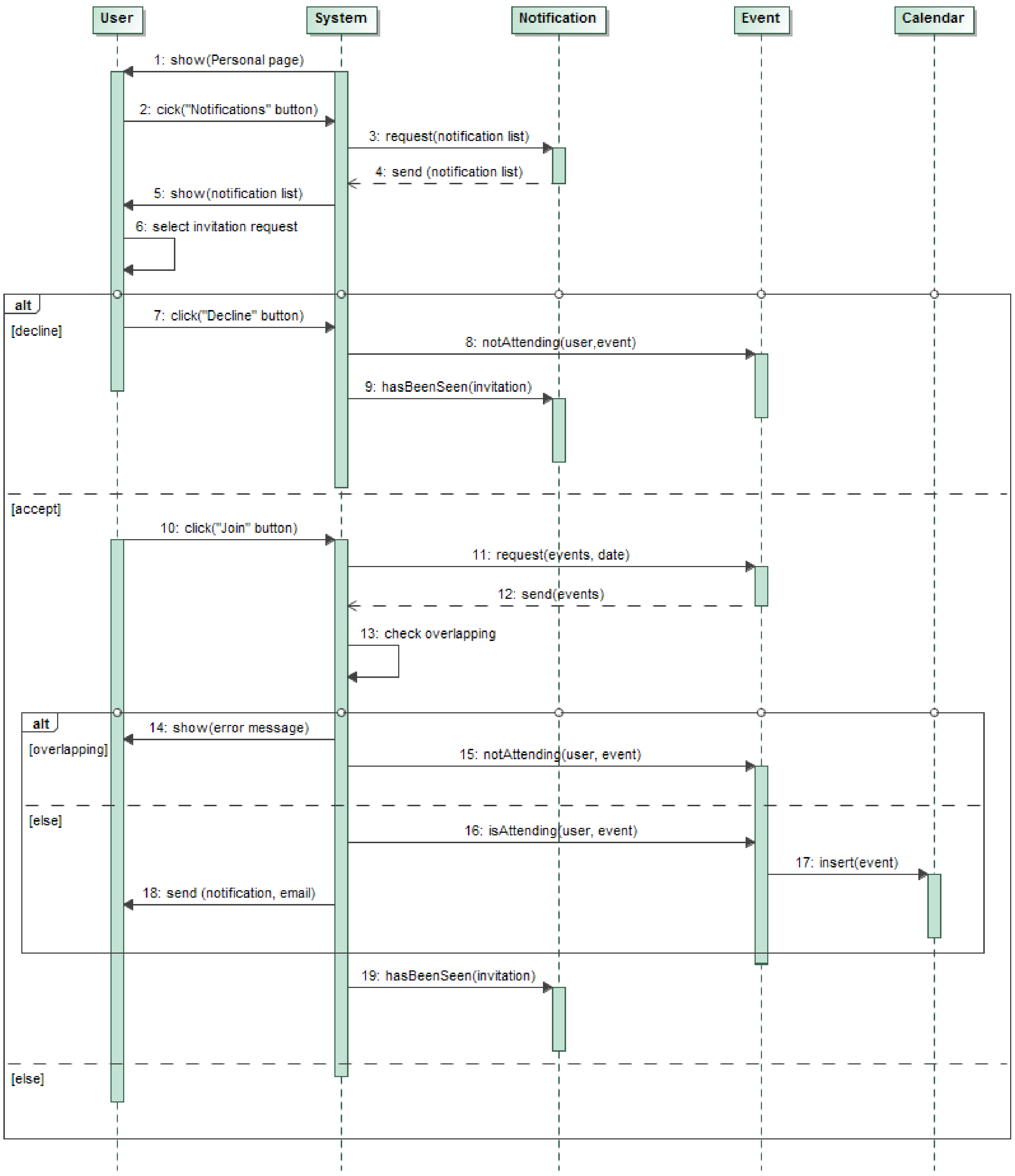

User
System
Notification
Event
Calendar
1: show(Personal page)
2: cick("Notifications" button)
3: request(notification list)
4: send (notification list)
5: show(notification list)
6: select invitation request
alt
[decline]
7: click("Decline" button)
8: notAttending(user,event)
9: hasBeenSeen(invitation)
[accept]
10: click("Join" button)
11: request(events, date)
12: send(events)
13: check overlapping
alt
[overlapping]
14: show(error message)
15: notAttending(user, event)
[else]
16: isAttending(user, event)
17: insert(event)
18: send (notification, email)
19: hasBeenSeen(invitation)
[else]

Figure 18 – Sequence diagram of the interactions involved in the re-scheduling process. If a rescheduling is accomplished, all the participants will be notified.

6.4.6. Import or export of a calendar

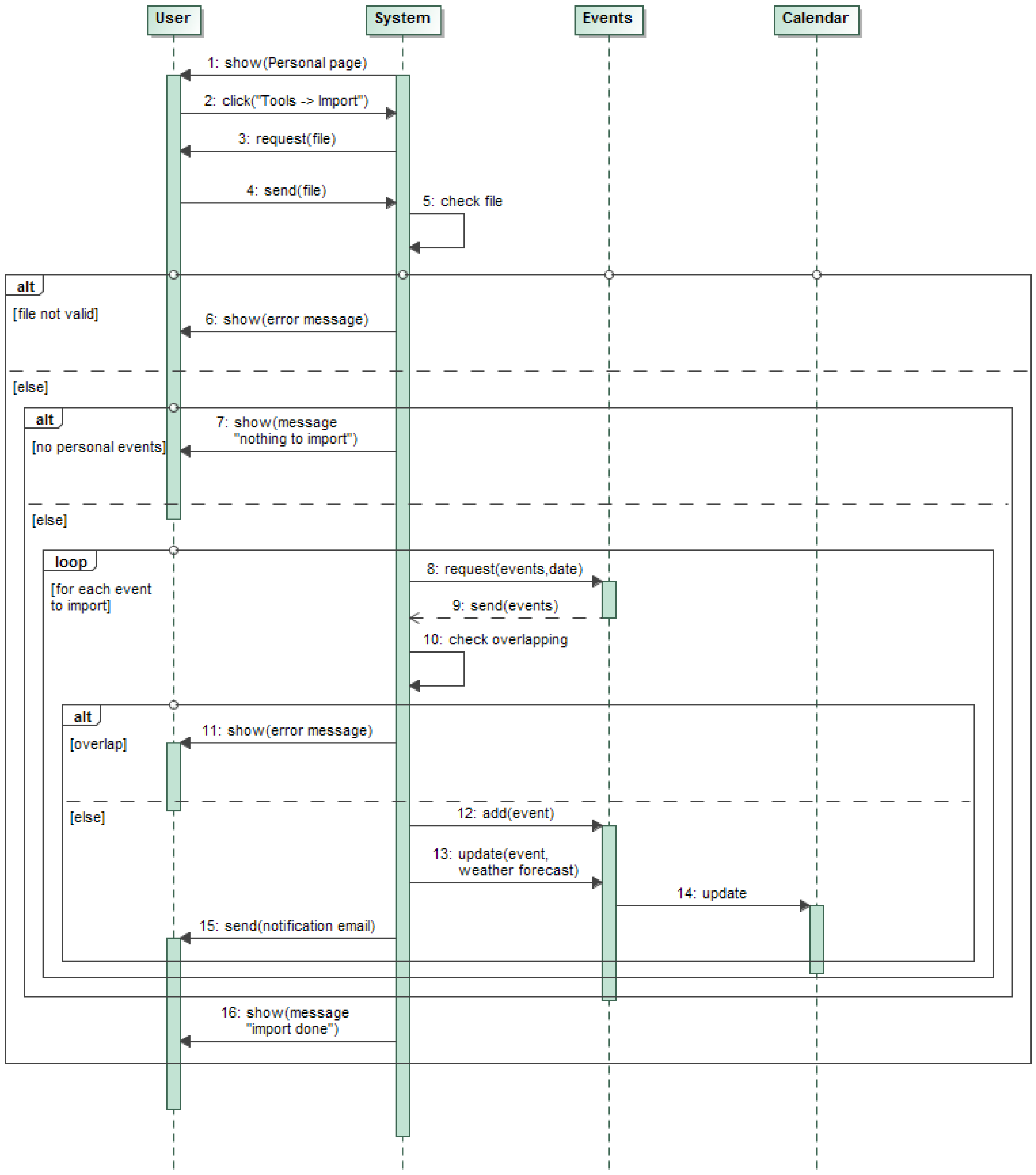


Figure 19 – Sequence diagram of the interactions involved in the process of importing a calendar.

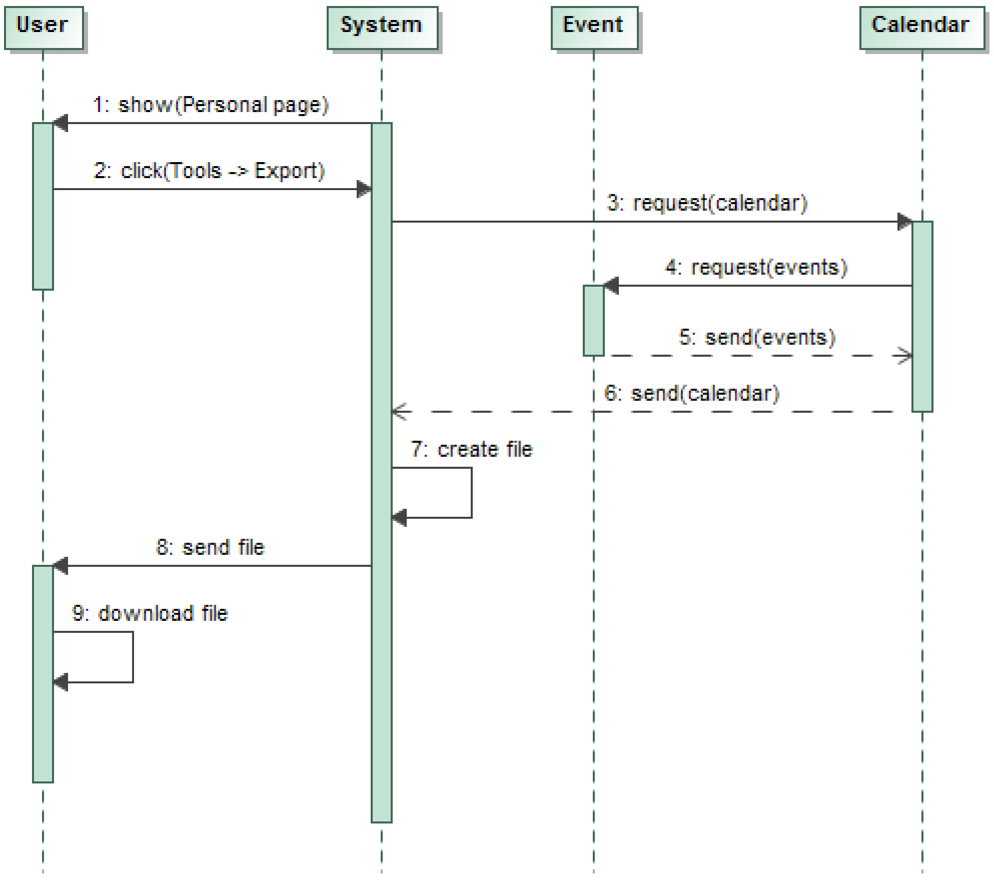


Figure 20 – Sequence diagram of the interactions involved in the process of exporting a calendar.

6.5. State chart diagram

The following is the only relevant state chart diagram that we can generate and it is about an invitation.

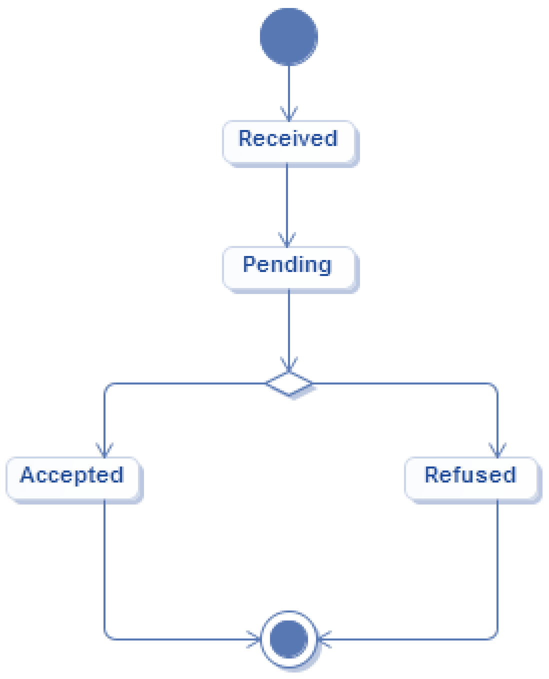


Figure 21 – The state chart diagram of an invitation sent to a user who has been invited to an event (by another user). In the exit state the notification will be marked as “seen”.

7. Alloy modeling

Here is the Alloy code used to verify the consistency of the class diagram (see Section 6.1 – Class diagram):

```
module MeteoCal


//SIGNATURES-------------------------------------------------------------

sig WeatherCondition{}

sig User{}

sig Event{
      creator: one User,
      attended: some User,
      declined: set User,
      constraints: set WeatherCondition,
      forecast: lone WeatherCondition
}{
      //Embedded facts of Event
      attended not in declined //Who is attending cannot have declined
      declined not in attended //Viceversa
      creator in attended      //The creator must attend the event
}

sig Notification{
      receiver: one User
}

sig Calendar{
      owner: one User,
      displayed: set Event
}

sig Invitation extends Notification{
      generator: one Event
}

sig Information extends Notification{
      generator: lone Event
}
```

```
//FACTS-------------------------------------------------------------------

fact eventProperties{

      //Users that have declined cannot also have accepted (wrt the
            invitation)
      no u: User | one i: Invitation | one e: Event | u = i.receiver &&
            u in e.attended && e = i.generator

      //The creator has the event in his calendar
      no e: Event | one c: Calendar | e.creator = c.owner && not (e in
            c.displayed)

      //Attendees must see the event in their calendar
      no u: User | one c: Calendar | some e: Event | u = c.owner && u in
            e.attended && e not in c.displayed

      //Not going people mustn't see the event in the calendar
      no u: User | one c: Calendar | some e: Event | u = c.owner && u in
            e.declined && e in c.displayed

      //All other people cannot see the event in the calendar
      no u: User | one c: Calendar | some e: Event | u = c.owner && not
            (u in e.attended || u in e.declined) && e in c.displayed

      //Attendees no longer have invitations for that event
      no i: Invitation | one e: Event | e.attended = i.receiver &&
            e = i.generator

}

fact calendarProperties{
      //Two different calendars  cannot have the same owner
      no disj c1, c2: Calendar | c1.owner = c2.owner
}

fact userProperties{
      //Every user has one calendar
      all u: User | one c: Calendar | c.owner=u
}

fact invitationProperties{

      //If an user has an *invitation* for an event, he don't see the
            event in the *calendar*
      no u: User | some i: Invitation | one c: Calendar | u = i.receiver
            && u = c.owner && i.generator in c.displayed

      //There cannot be multiple invitations directed to the same user,
            for the same event
      no disj i1, i2: Invitation | i1.receiver = i2.receiver &&
            i1.generator = i2.generator

}

```

```
//ASSERTIONS------------------------------------------------------------

assert NoSelfInvitation{
      no e: Event | some i: Invitation | e.creator =  i.receiver &&
            e = i.generator
}

assert NoRefusingCreators{
      no u: User | some e: Event | u = e.creator && u in e.declined
}

assert NoUnconsistentPeople{
      no u: User | some e: Event | u in e.declined && u in e.attended
}

assert NoCreatorsWithoutTheirEventsInCalendar{
      no e: Event | some c: Calendar | e.creator = c.owner && not (e in
            c.displayed)
}

assert NoDisplayForNotGoing{
      no u: User | some c: Calendar | some e: Event | u = c.owner && u
            in e.declined && e in c.displayed
}

assert NoGhostCreations{
      no e: Event | some c: Calendar | e.creator = c.owner && not (e in
                  c.displayed)
}

//Checks of the assertions
check NoSelfInvitation
check NoRefusingCreators
check NoUnconsistentPeople
check NoCreatorsWithoutTheirEventsInCalendar
check NoGhostCreations


//PREDICATES------------------------------------------------------------

pred showEventCreation(){
      #Event<5
}

run  showEventCreation  for  7  but  4  User,  4  Event,  2  Notification,  0
      WeatherCondition

pred show(){}

run show for 4
```

As well, here follows the result of the execution in Alloy Analyzer:

```
7 commands were executed. The results are:

   #1: No counterexample found. NoSelfInvitation may be valid.
   #2: No counterexample found. NoRefusingCreators may be valid.
   #3: No counterexample found. NoUnconsistentPeople may be valid.
   #4: No counterexample found. NoCreatorsWithoutTheirEventsInCalendar may
be valid.
   #5: No counterexample found. NoGhostCreations may be valid.
   #6: Instance found. showEventCreation is consistent.
   #7: Instance found. show is consistent.
```

Please refer to *Chapter 8 – Generated worlds* for the instances of world generated by Alloy Analyzer.

8. Generated worlds

In this chapter, we show some worlds that have been generated by Alloy Analizer – to illustrate the consistency of our model.

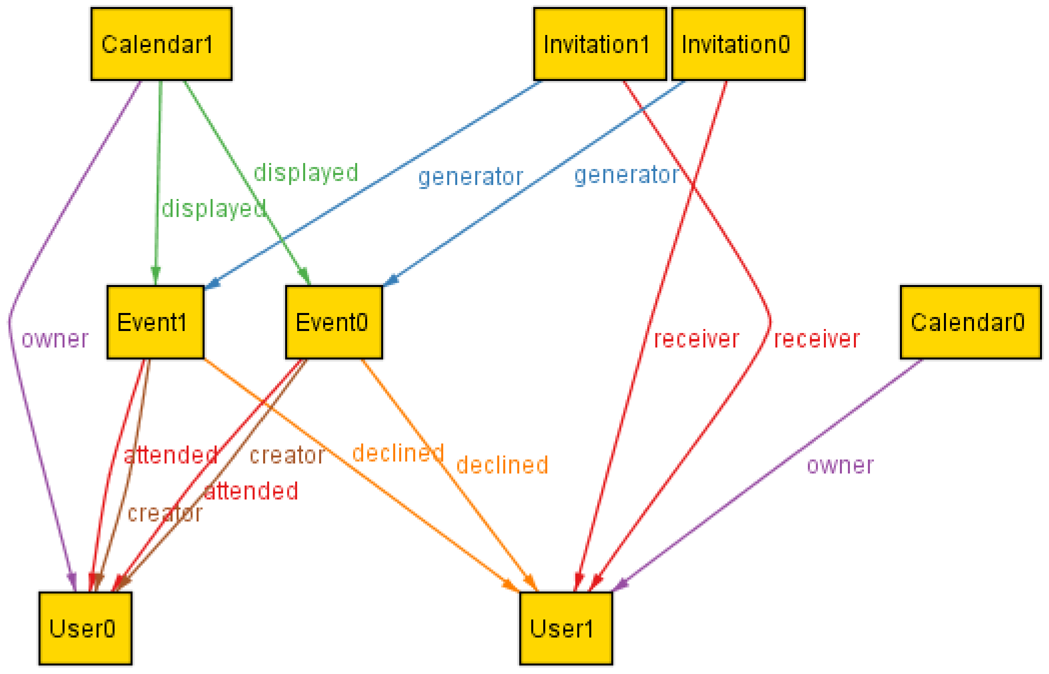


Figure 22 – In this simple scenario, a user creates two events, inviting another user, who decline the invitation, but then he/she is invited again. As we can see, the events are displayed in the creator's calendar but not in the calendar of the second user.

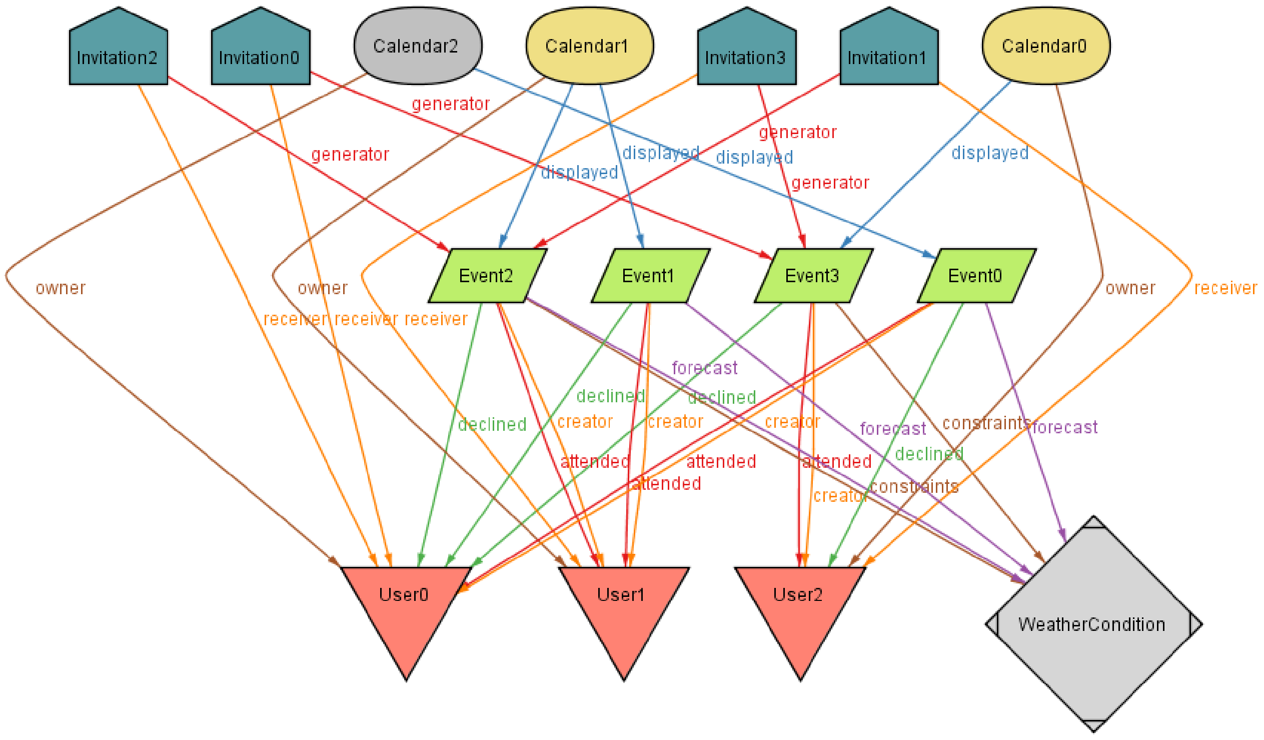


Figure 23 – This world is a little bit messy, so we are going to split it in different perspective.

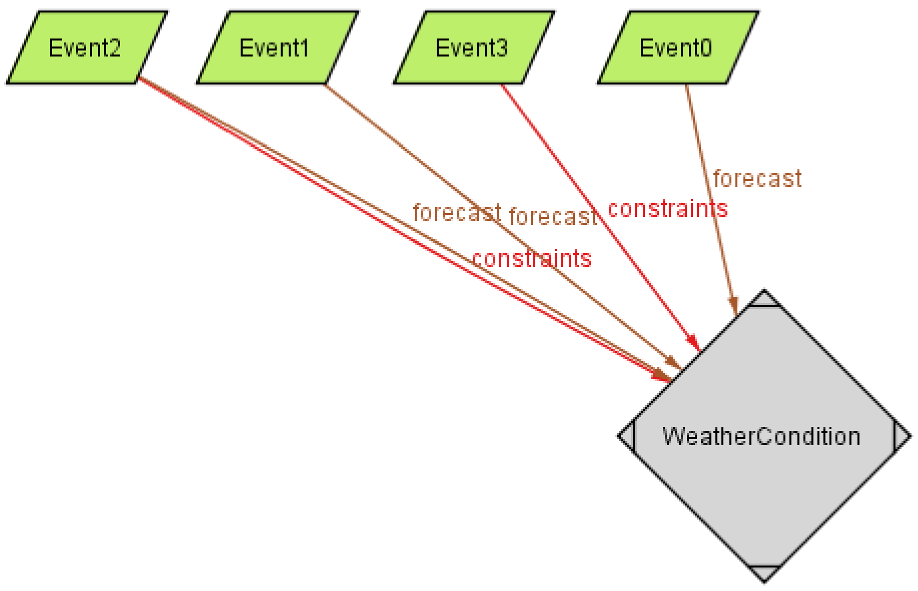


Figure 24 - Here we can see that every combination is possible: Event2 has a weather constraint, Event1 and Event0 have forecast information, but they have no constraints (since they are indoor events with a specified location). On the other hand, Event3 has defined a constraint, but a weather forecast is not yet available.

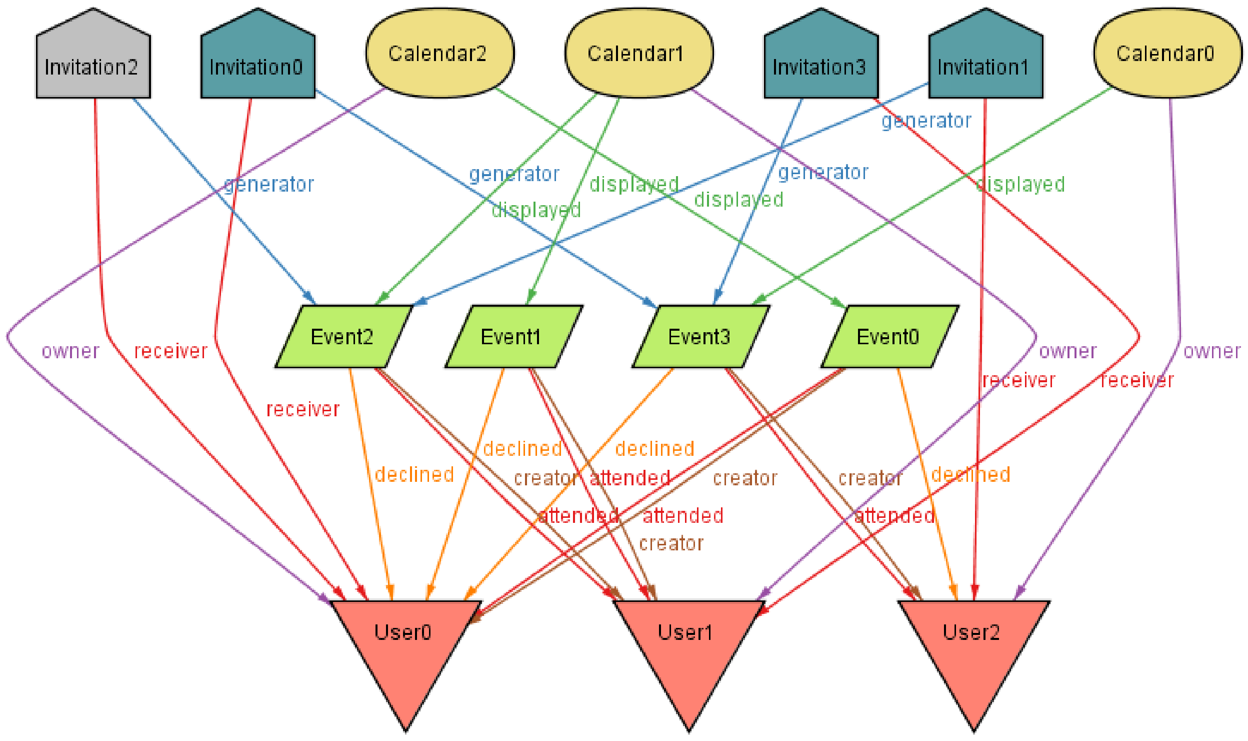


Figure 25 - Here it is possible to see that every event has a creator and the creator is always attending the event. In addition, only the users that attend the events see them in their calendar. Every invitation has only one event (the generator) and one user (the receiver). Please note that user User2 has declined the Event0, but has not received any other invitation about that event (this scenario was not covered in the previous world (see Figure 22).

Here follows some other generated worlds:

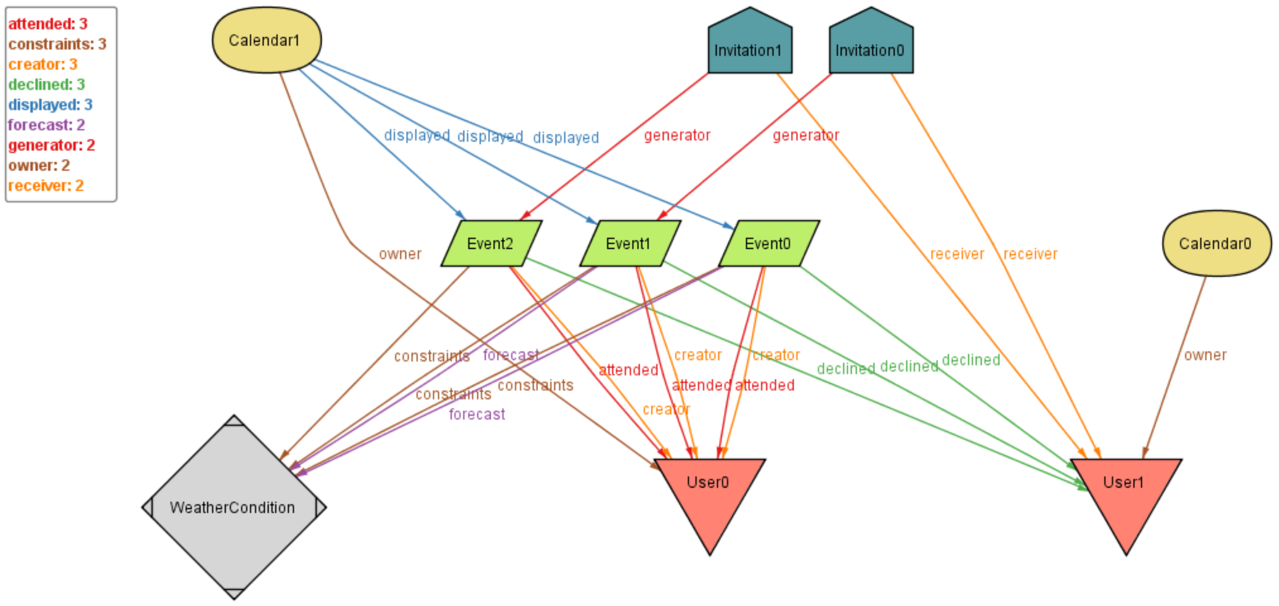


Figure 26 – Another similar world, where a user has no events in his/her calendar.

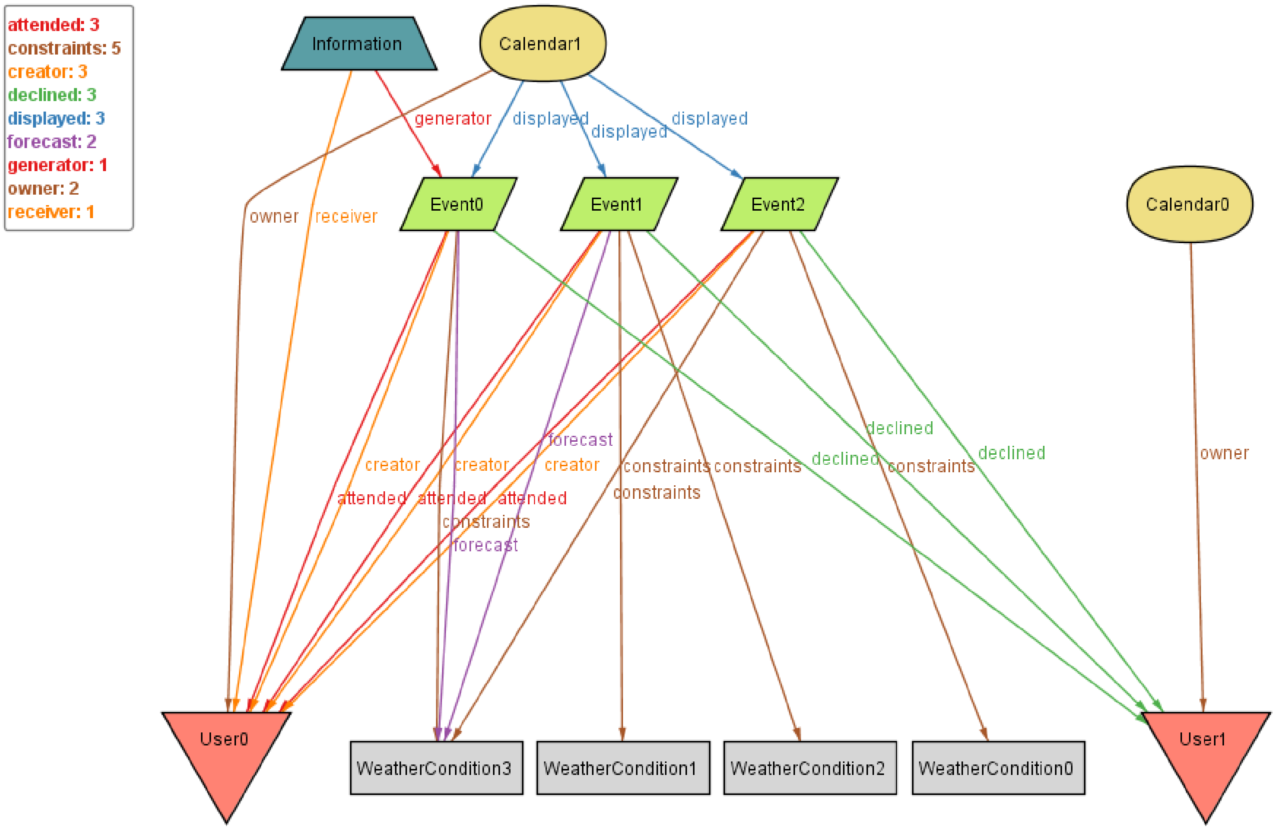


Figure 27 – In this world there is a general Information generated by one event. It is also possible to see that every event can have multiple weather constraints, but only one weather forecast.

9. Used Tools

The tools used to prepare this RASD document are:

Microsoft Word 2013	– http://products.office.com/en-us/word
Microsoft OneNote 2013	– http://www.onenote.com/
OneDrive	– https://onedrive.live.com/
Alloy Analyzer 4.2	– http://alloy.mit.edu/alloy/
Balsamiq Mockups 2.2	– http://balsamiq.com/
MagicDraw 18.0	– http://www.nomagic.com/products/magicdraw.html

Appendix A – Informal specification

Problem description

- The X **software house** wants to offer a new weather based online 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 events;
    - Delete events;
    - Update events.
- An event should **contain** information about **when** and **where** the event will take place whether the event will be indoor or outdoor.
- During event creation, any number of registered users can be invited.
- Only the organizer will be able to update or delete the event.
- Invited users can only:
    - Accept the invitation;
    - Decline the invitation.

- **Whenever** an event is **saved**, the system should enrich the event with weather forecast information (if available).
- The system should notify all event participants one day before the event in case of bad weather conditions for outdoor events.
- *Notifications are received by the users when they log into the system.*

- A user X should be allowed to make his/her calendar public, that is, visible to **all** other **registered** users.
- The registered users will see all the time slot in which X is busy but without seeing the details of the corresponding events, unless they have been defined as public.
- Events can be defined as **public** or **private** by their owners, upon creation.
- If an event is public, all the registered users can see its details, including the corresponding participants.

- In case of bad weather conditions for outdoor events, three days before the event, the system should propose to its creator the closest (in time) *sunny day* (if any).

- Implement email notifications (both for invitation and cloudy outdoor events alerts).
- Allow users to import and export their calendar.
- *Manage time consistency when creating an event by avoiding conflicts with existing events.*
- Update weather information associated to events periodically (e.g. every 12 hours), and, of course, notify outdoor event participants in case the forecast has changed.

The task

- We must develop the system using Java EE platform.
- We will use EJBs to develop the business logic.
- *The user interface can either be a web application or a normal Java application.*
- In both cases (web application, normal Java application) the user interface has to interact with the business logic.

Appendix B – Revision history

Initial release:	v1.0
Current release:	v1.0
Date of the last review:	2014/11/16

v0.01

2014/11/14 – Partial draft (first two chapters, from OneNote).

v0.02

2014/11/14 – Partial draft (first three chapters, added chapter 5, from OneNote, section 1.6 fixed).

v0.10

2014/11/15 – Partial draft (first six chapters, from OneNote, chapter 3 fixed).

v0.20

2014/11/15 – Draft copy (all nine chapters, from OneNote, added appendices A and B, added figures to chapter 8, chapter 4 fixed).

v0.89

2014/11/16 – Draft copy (pre-release, added figures, chapter 3 fixed).

v0.90

2014/11/16 – Draft copy (pre-release, chapter 5 fixed, section 6.4 fixed, added captions).

v0.91

2014/11/16 – Release candidate 1 (juicy version).

v0.92

2014/11/16 – Release candidate 2 (chapter 4 fixed, section 6.5 fixed).

v0.99

2014/11/16 – Release candidate 3 (subsection 3.4.1 fixed).

v1.0

2014/11/16 – Initial document.