Politecnico di Milano
A.A. 2014-2015
Software Engineering 2: "MeteoCal"
Requirements **A**nalysis and **S**pecifications **D**ocument

version 2.0

Federico Migliavacca (mat. 836582), Leonardo Orsello (mat. 837176)

25 January 2015

Contents

1 Introduction

1.1 Purpose

This document represent the Requirement Analysis and Specification Document (RASD). The main goal of this document is to completely describe the system in terms of functional and non-functional requirements, analyse the real need of the customer to modelling the system, show the constraints and the limit of the software and simulate the typical use cases that will occur after the development. This document is intended to all developer and programmer who have to implement the requirements, to system analyst who want to integrate other system with this one, and could be used as a contractual basis between the customer and the developer.

1.2 Actual System

The software house wants to offer a new weather based online calendar. We suppose that until now nothing has been created and we have to create the entair application without using or modify a previous system.

1.3 Scope

The aim of the project is to create a brand new weather based online calendar for helping people scheduling their personal events considering weather forecast. Calendar may be set as public or private. Public calendars can be seen by all registered users, instead private ones can be seen only by owners. An event contains information about when and where the event will take place. The user can check the weather forecast and decide the best schedule for the specific event. User, once registered, should be able to create, delete and update events. User, can also invite other users to partecipate to a specific event. Invited users can only accept or decline the invitation. User can choose if the specific event will be public or private, determine the visibility of the event towards other registered user. In case of bad weather conditions for outdoor events, the system should propose to its creator, three days before the event take place, the closest sunny day. The application should also notify the bad weather to all event participants one day before the event. The user will see the notifications when they log in to the system.

1.4 Actors

- Visitor: all visitor users can only see the login page and complete the registration form to be able to access to all the functionality of the application as registered user.
- Registered user: this type of user, after successful login, is the only that can see the calendar interface and create, delete and update events. They can also invite other user to a specific event during the creation process

of the event itself. After the creation of the event, the application shows the weather forecas of that event to the user.

1.5 Goals

List of the goals of MeteoCal application:

- [G1] Allow a visitor to became a registered user and choose the public or private nature of his/her calendar.
- [G2] Allow user to log in to application.
- [G3] Allow user to create a new event in the calendar and choose the public or private nature.
- [G4] Allow user to modify an existing event of his/her calendar.
- [G5] Allow user to delete an existing event of his/her calendar.
- [G6] Allow user to invite/delete other user to a specific event of his/her calendar.
- [G7] Allow user to see the weather forecast of a specific event of his/her calendar.
- [G8] Allow user to see the public event of other registered user.
- [G9] Allow user to see event to which has been invited.
- [G10] After login, application will notify only the creator user three days before an event takes place if the weather is not good.
- [G11] After login, application will notify invited user one days before an event takes place if the weather is not good.

1.6 Definitions, Acronyms, Abbreviations

1.6.1 Definitions

- Creator user: is a registered user that creates an event and becomes the owner of the event itself.
- Calendar owner: all registered user that administrate his/her calendar.
- Event owner: user that has created an event.
- Invited user: is a registered user who has been invited to an event from other registered user and he/she have accepted.
- Event: anything that takes place or happens in the real world. In our case an event is the only entity that user can create, delete or modify.

- Visibility: relation between two entities (calendar, event and user) that imply property of been visible. If A is visible from B, A will show informations to B. If A is not visible from B, A will hide informations to B.
- Closest sunny day: is the first sunny day encountered during the analysis of a specific period of time.
- Bad weather: climate conditions such as rain, storm, snow.

1.6.2 Acronyms

- RASD: Requirements Analysis and Specification Document.
- DB: DataBase.
- DBMS: DataBase management system.
- API: Application Programming Interface.
- Database Management System (DBMS).
- OS: Operating System.
- JVM: Java Virtual Machine.
- JEE: Java Enterprise Edition.

1.6.3 Abbreviations

- [Gn]: n-goal.
- [Rn]: n-functional requirement.
- [Dn]: n-domain assumption.
- [Mnn]: n-mock up.

1.7 Reference Documents

- Specification Document: MeteoCal Project AA 2014-2015.pdf.
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
- IEEE Std 1016^{tm}-2009 Standard for Information Tecnology-System Design-Software Design Descriptions.

1.8 Document overview

This document is essentially structured in four part:

- Section 1: Introduction, it gives a description of document and some basical information about software.
- Section 2: Overall Description, gives general information about the software product with more focus about constraints and assumptions.
- Section 3: Specific Requirements, this part list requirements, typical scenarios and use cases. To give an easy way to understand all funcionality of this software, this section is filled with UML diagrams.
- Section 4: Appendix, this part contains some information about the attached .als file and some described screenshot of software used to generate it.

2 Overall Description

2.1 Product perspective

The application we will release is a web application which is not integrated with other existing system. That application will not have any internal interface for administration but it will be only user based. The application will not provide any interface or API for integration with future project.

2.2 User characteristics

The user that we expect to use our application is a person who want an easy way to shedule his/her events in relation with weather forecast. This user must be able to use a web browser and have access to internet.

2.3 Constraints

2.3.1 Regulatory policies

MeteoCal doesn't have to meet any regulatory policies.

2.3.2 Hardware limitations

MeteoCal doesn't have to meet any hardware limitations.

2.3.3 Interfaces to other applications

MeteoCal doesn't have to meet any interfaces to other applications.

2.3.4 Parallel operation

MeteoCal must support parallel operations from different users when working with database and with all operation done by the user after connection.

2.3.5 Documents related

- Requirements and Analysis Specification Document (RASD).
- Design Document (DD).
- User's Manual.
- Testing report.

2.4 Assumptions and Dependencies

2.4.1 Assumption

- There isn't an administrator or privileged user. In the text is not specify clearly but we think that is not necessary a hierarchy of users to keep the system safe.
- There isn't any depence between users;
- User have only one calendar.
- For any day can be created illimitate number of events.
- Weather forecast information will be shown only if available, otherwise a message or a notice will inform the user.
- Notification to the owner for bad weather forecast, three days before the event takes place, will propose only the closest sunny day analysing only next seven days after the day of the event is scheduled.
- If weather forecast is not available at the moment of creation of the event, this one will be created anyway and the weather field will be scheduled to be update the next login.
- If an owner of an event delete it, all invited user will be deleted from this event.
- If a user decline the invite to an event could be invited again.
- When any field of an event is modify by the owner all the invited users that have previously accepted the event will be notify of the changes and asked to reaccept the invitation.
- Notification of bad weather condition will be shown.
- The choice to set an event public or private will not be possible to modify later.
- There is not periodical update of weather forecast for event created. Weather will be check at every login only for the event in the next three days.
- Assumption about visibility will be described by the following table:

Visibility	Private Event	Public Event
Private Calendar	Owner, Invited User	Owner, Invited User, Registered user
Public Calendar	Owner, Invited User, Registered user(without detail of event)	Owner, Invited User, Registered user

2.5 Future possible implementation

- Increase of Granularity of visibility policy: MeteoCal wants to provide more possibility of personalization for visibility policy. In order to achive this goal visibility table will be expanded with more fields. Will be available the option, indipendantly form the nature of the calendar or event, to choose more restrictions about who can view our events. For example a user "A" has a Public Calendar and creates a Private Event, according to old visibility table any registred user can see that he is busy that specific date; with new granularity user "A" can set the Private Event to be invisible to non invited user. This new features will be add to satisfy users' request of major control upon privacy.

- Some security policy could be implemented in this application. This will be well descibed in security part under non-funcional requirements section.

- An important function could be E-mail notification. E-mail notification for registered user is a better way to send notification in time.

- Another useful function for user who want to use this web application for the first time is the possibility to import/export his/her calendar.

- Manage time consistency when creating an event by avoiding conflicts with existing events.

- Update weather information associated to events periodically.

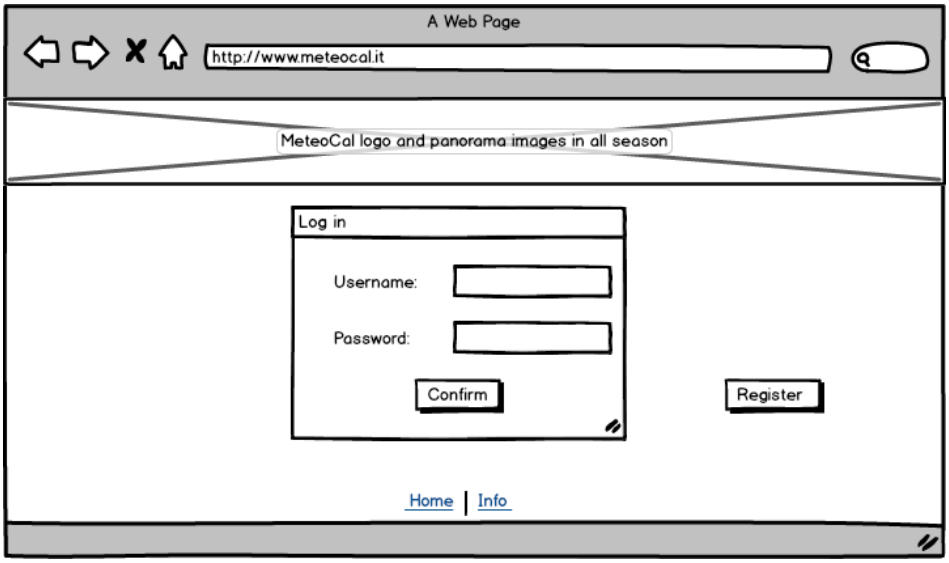
3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

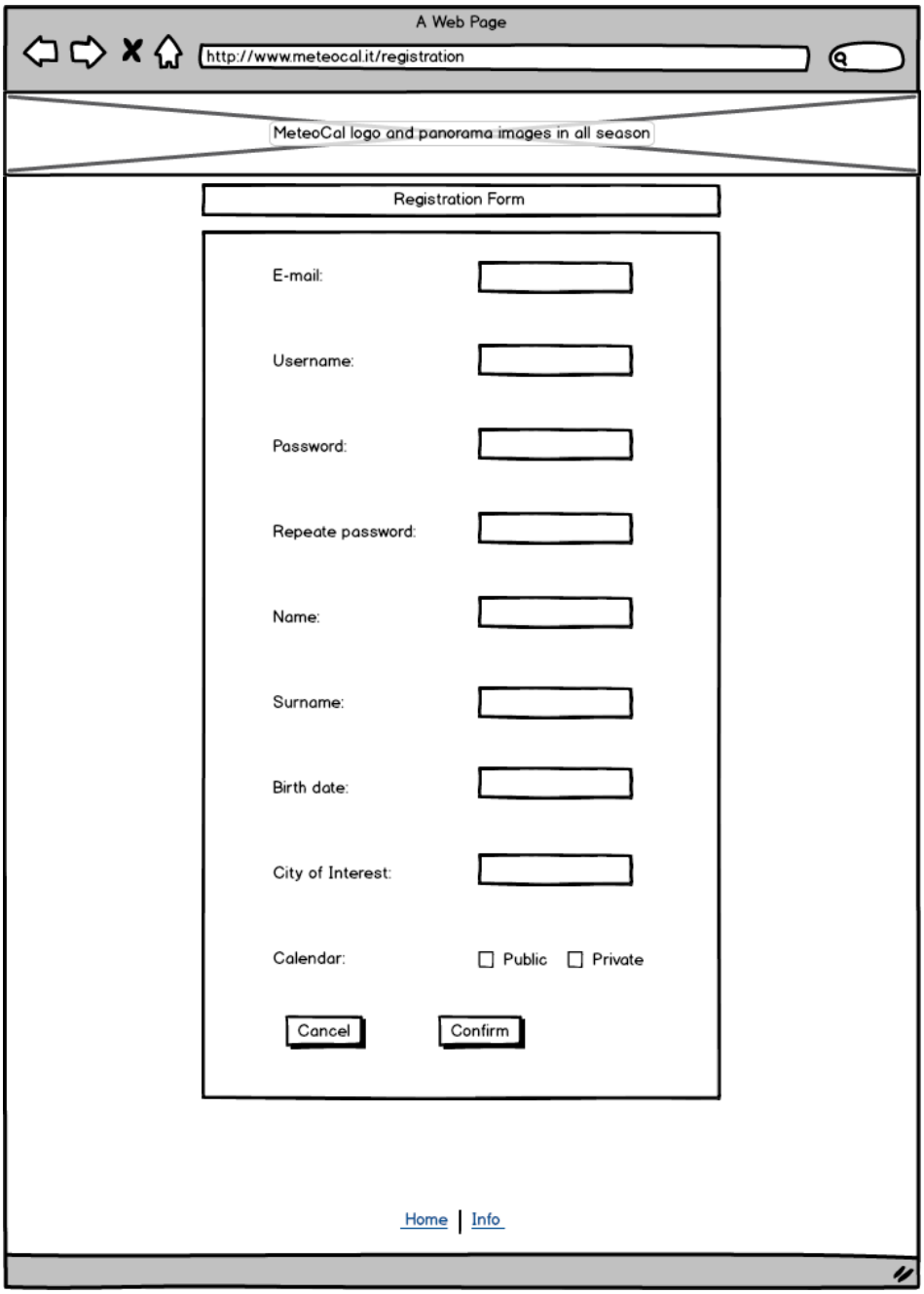
Here are presented some mockup that represent an idea of the structure of the application pages:

3.1.1.1 Login The mockup above shows the home page of MeteoCal. Here users can log in to the application and visitors can access to the regitration form.

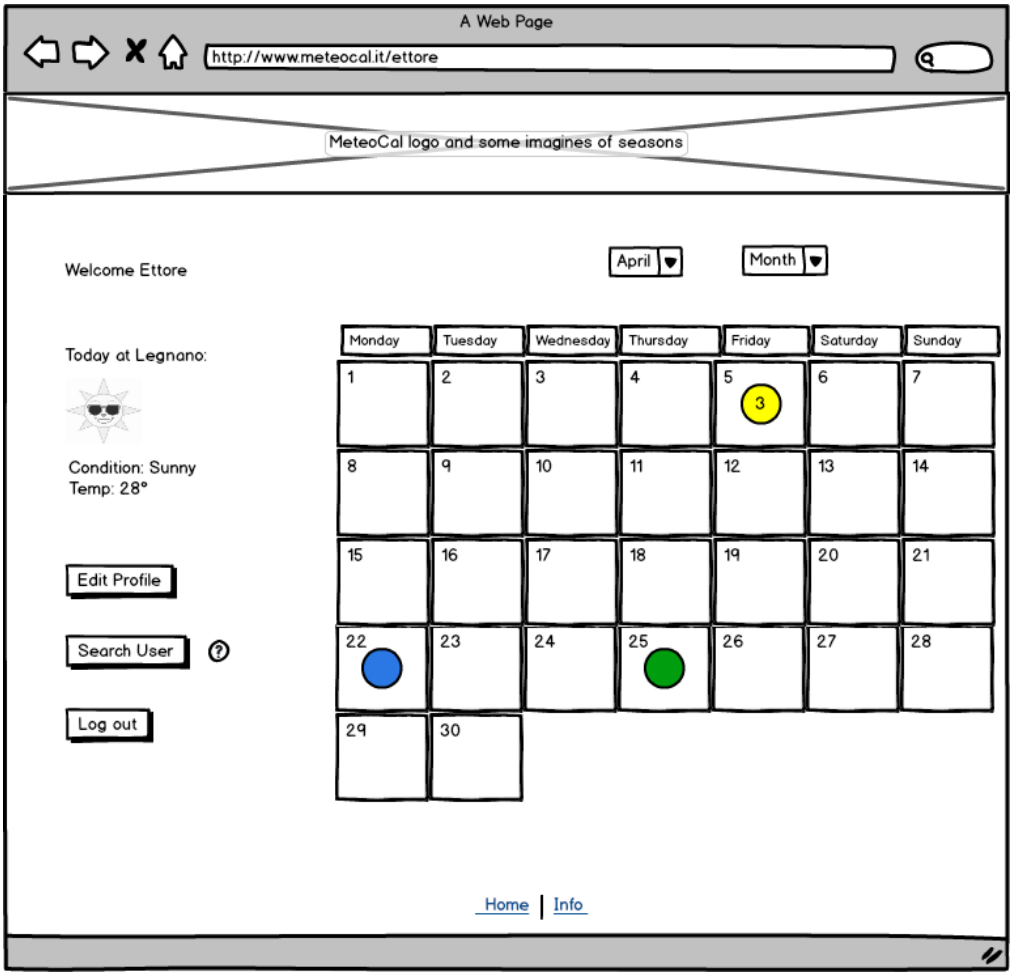


3.1.1.2 Registration form

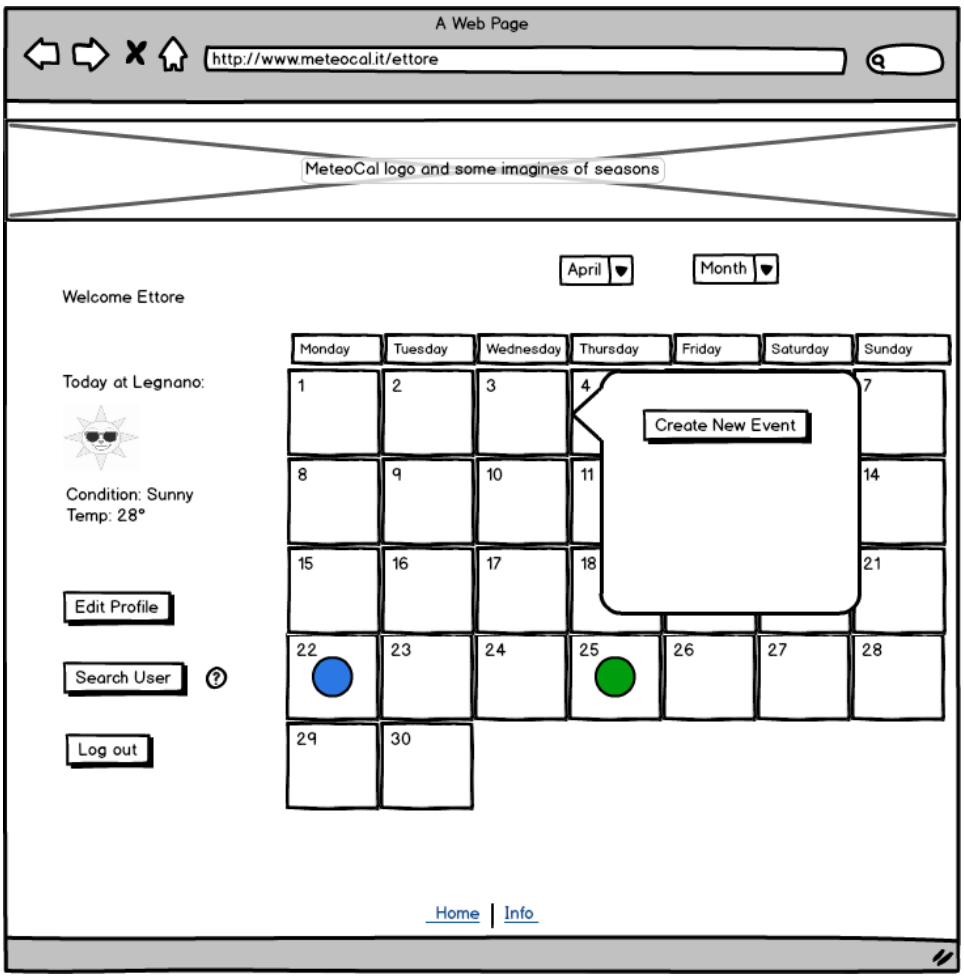
This mock shows the registration form page.



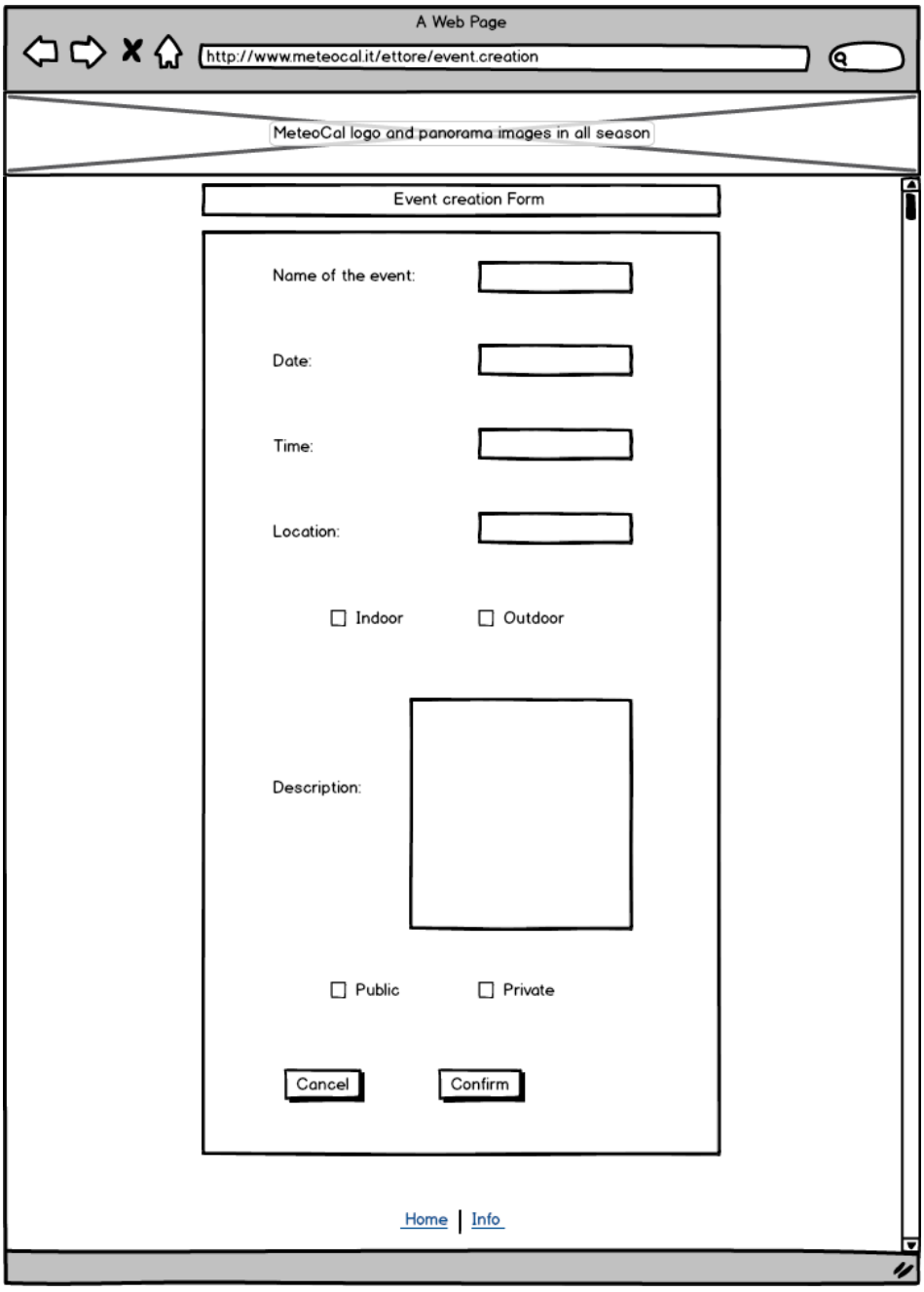
3.1.1.3 Home page This mockup shows the home page of an hipotetic user Ettore. The circles rapresent events, the colours if they are private or public, or both (only if in specific date I have at least two events, one private and the other public, for example on friday the 5th.



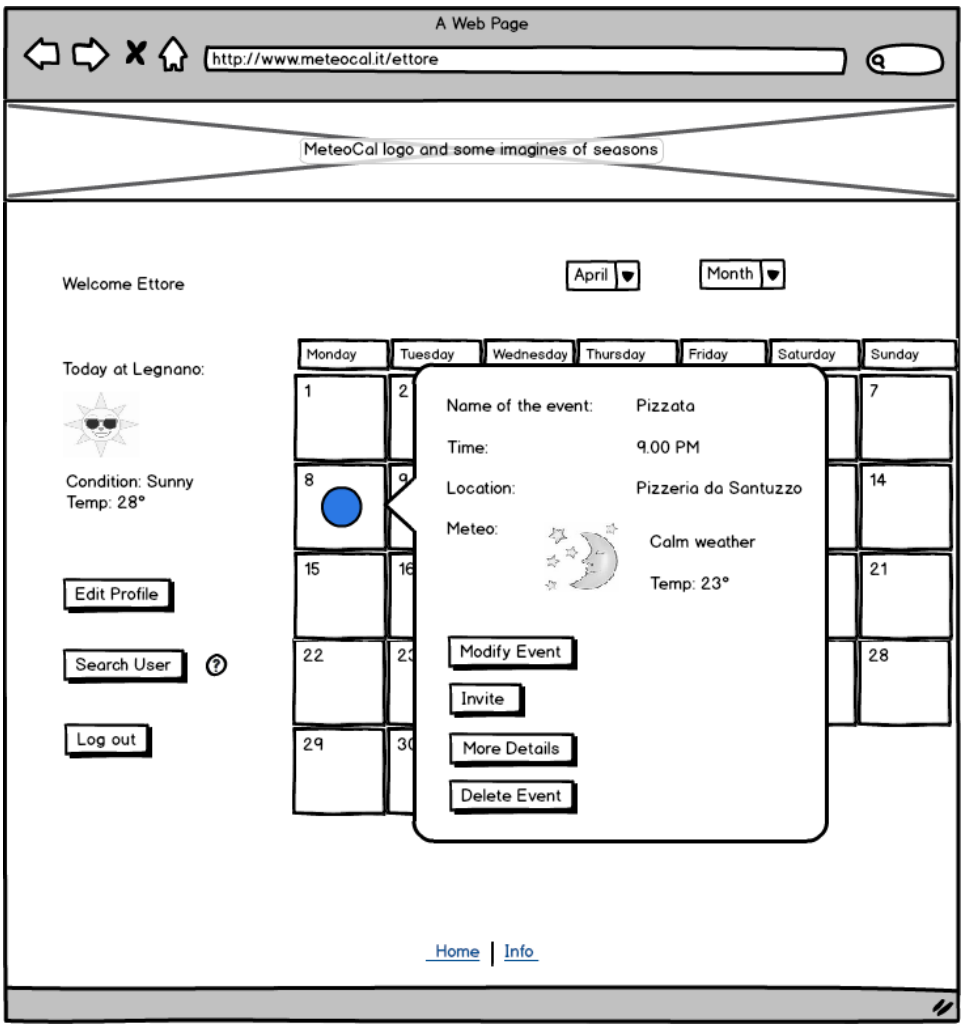
3.1.1.4 Click pop-up This mock up shows what happens when a user clicks on a blank date. A pop-up message will ask to the user if he/she wants to create a new event.



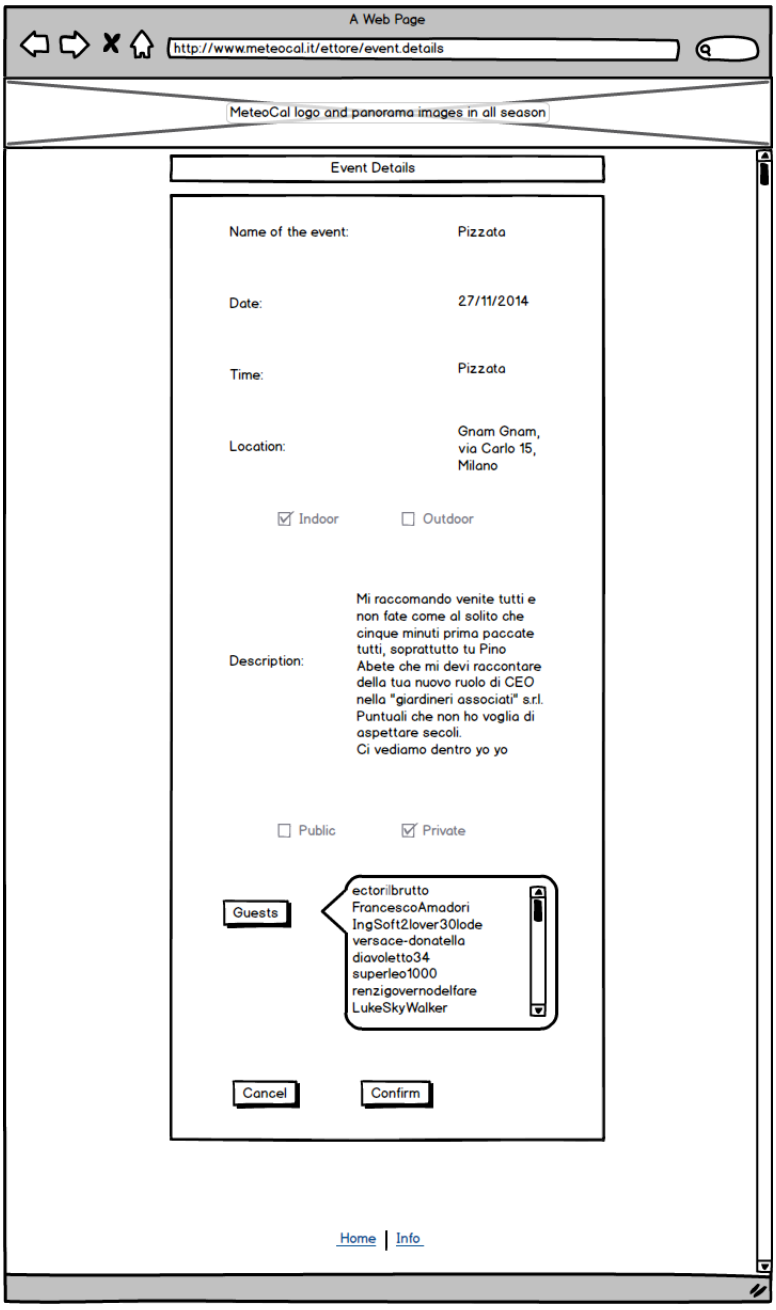
3.1.1.5 Event creation This mock up shows the event creation form.



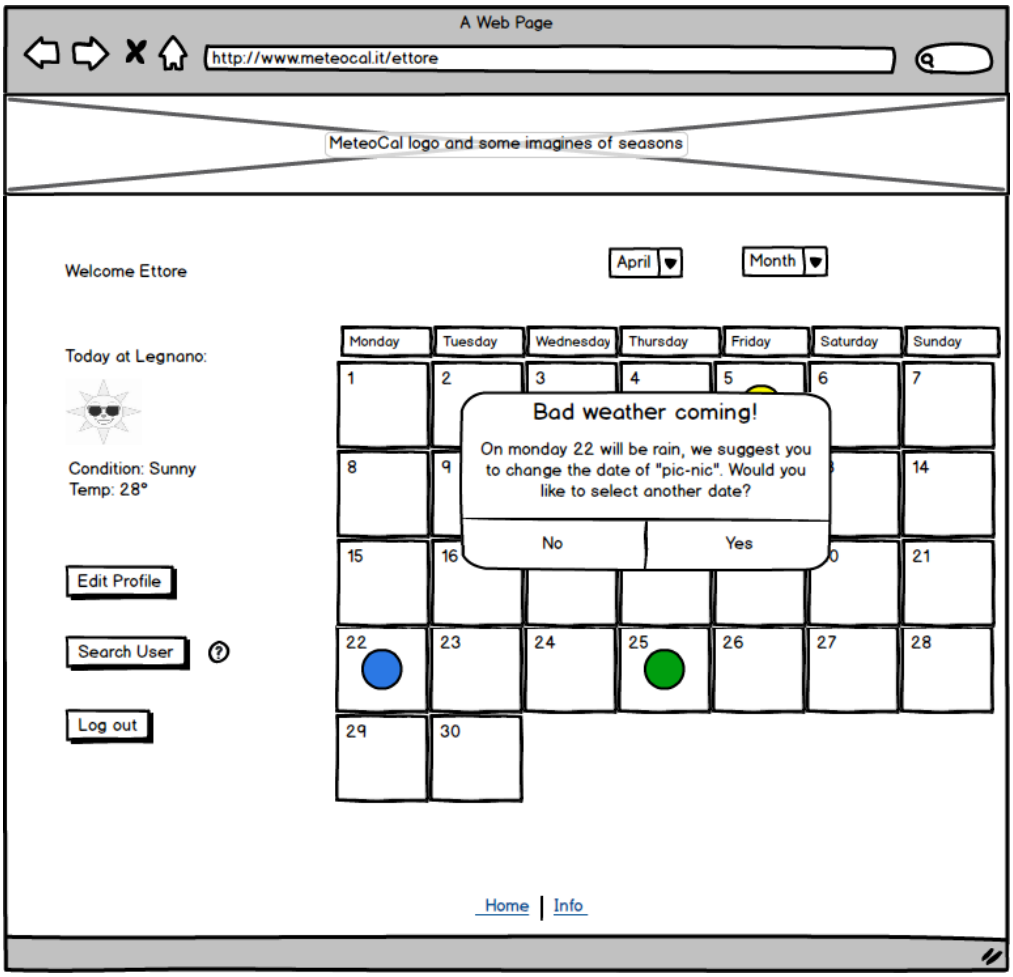
3.1.1.6 Click event created This mock up shows what happens when the user clicks on an event he created (event represented by a colored circle). A pop-up will show some informat and give some functionality options.

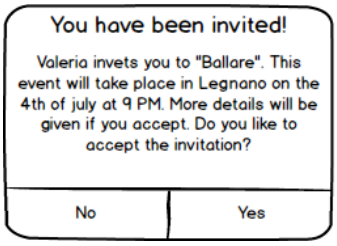


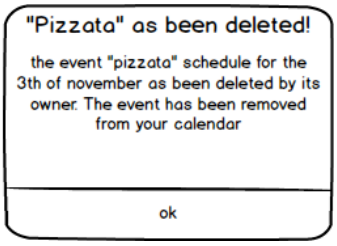
3.1.1.7 More detail on event This mock up shows what user sees when he click on "more details".



3.1.1.8 Alert This mock up shows examples of pop-up alert.







3.1.2 API interfaces

For the weather forecast part of the MeteoCal application we use the OpenWeatherMap API. As well described on the website this API give access to current weather data for any location on Earth including over 200,000 cities. Current weather is frequently updated basing on global models and data from more than 40,000 weather stations. Also Data is available in JSON, XML, or HTML format. For the forecast information indeed both 5 day forecasts and 16 day forecasts are available at any location or city. 5 day forecasts include weather data every 3 hours and 16 day forecasts include daily weather. For more information see the OpenWeatherMap website(http://openweathermap.org/api).

3.1.3 Hardware Interfaces

This project does not support any hardware interfaces.

3.1.4 Software Interfaces

- Database Management System (DBMS):
  - Name: MySQL.
  - Version: 5.6.21
  - Source: http://www.mysql.it/
- Java Virtual Machine (JVM).
  - Name: JEE
  - Version: 7
  - Source: http://www.oracle.com/technetwork/java/javaee/tech/index.html
- Application server:
  - Name: Glassfish.
  - Version: 4.1.
  - Source: https://glassfish.java.net/
- Operating System (OS).
  - Application must be able to run on any SO which supports JVM and DBMS specified before.

3.1.5 Communication Interfaces

Protocol	Application	Port
TCP	HTTPS	443
TCP	HTTP	80
TCP	DBMS	3306 (default)

3.1.6 Memory

The minimum memory requirements are:

- Primary Memory: 2GB+
- Secondary Memory: 32GB+

3.2 Functional Requirements

3.2.1 [G1] Allow a visitor to became a registered user and choose the public or private nature of his/her calendar.

- [R1] Visitor must not be already registred to perform registration process.
- [R2] Visitor must choose a username not already used by another user.
- [R3] User can not sign up twice but only once for session.
- [R4] Visitor can just see login page.
- [R5] Visitor can only access to registration form.
- [D1] Email address used for registration must be formally correct.

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

- [R1] User must be already regitred to success login process.
- [R2] User must know his username and password used during registration to success login.
- [R3] Username and password insert during login process must be correct.
- [R4] Wrong credentials will not grant access to user to the calendar.
- [R5] Visitor can't access to calendar page before registration.
- [R6] Application will not implement retrive password mechanism.

3.2.3 [G3] Allow user to create a new event in the calendar and choose the public or private nature.

- [R1] User must be already registred and logged in the application.
- [R2] User must complete event creation form and confirm the creation.
- [R3] User must complete mandatory fields to complete the event creation process.
- [D1] Time must be included between 00.00 and 23.59.

3.2.4 [G4] Allow user to modify an existing event of his/her calendar.

- [R1] User must be already registred and logged in the application.
- [R2] The event user wants to modify must have been already created and saved.
- [R3] User must confirm updating process.
- [R4] Updating process is not reversible, modified data will be lost.

3.2.5 [G5] Allow user to delete an existing event of his/her calendar.

- [R1] User must be already registred and logged in the application.
- [R2] User must be the owner of the event he wants to delete.
- [R3] User must confirm deleting process.
- [R4] Deleting process is not reversible, all event data will be lost.

3.2.6 [G6] Allow user to invite/delete other user to a specific event of his/her calendar.

- [R1] User must be already registred and logged in the application.
- [R2] User must be the owner of the event to invite or delete other users.
- [R3] User can not invite or delete itself.
- [R4] User can only invite registered users who have not been already invited.
- [R5] To invite a user, the owner of the event, must know email or name or surname or part of them of that specific user. A search bar is implemented to recognise the correct user to invite.
- [R6] User can not delete registered users who have been already deleted.
- [R7] User that have been delete from an event can be invited again.

3.2.7 [G7] Allow user to see the weather forecast of a specific event of his/her calendar.

- [R1] User must be already registred and logged in the application.
- [R2] User can see weather forecast information of a specific event only if they are allowed by visibility property.
- [D1] Weather forecast of a specific event can be seen only if it was available during creation process or scheduled updating.

3.2.8 [G8] Allow user to see the public event of other registered user.

- [R1] User must be already registred and logged in the application.
- [R2] Event must exist and be public.

3.2.9 [G9] Allow user to see event to which has been invited.

- [R1] User must be already registred and logged in the application.
- [R2] Event must exist, owner of the event has correctly created it.
- [R3] User must have been invited to the event from the owner.

3.2.10 [G10] After login, application will notify only the creator user three days before an event takes place if the weather is not good.

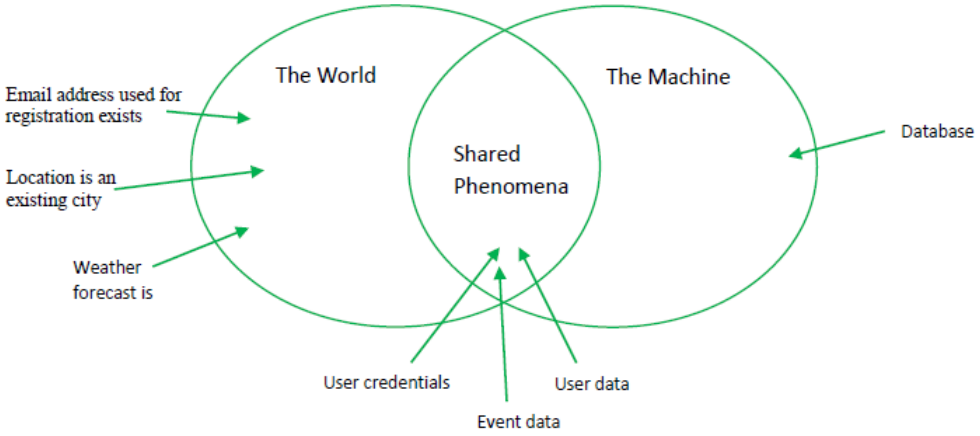
- [R1] User must be already registred and logged in the application.
- [R2] Event must exist, owner of the event has correctly created it.
- [R3] Application will notify the owner of the event only when he/she performs login.
- [R4] Notification will propouse to its creator the closest sunny day if available.

3.2.11 [G11] After login, application will notify invited user one days before an event takes place if the weather is not good.

- [R1] User must be already registred and logged in the application.
- [R2] Event must exists, owner of the event has correctly created it.
- [R3] Application will notify invited users of the event only when they perform login.

3.3 The world and the machine

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



3.4 Scenarios

3.4.1 Scenario 1

Ottaviano's birthdays is in five days, he has already booked a room in "mega super awesome", the most exclusive local in his home town. Ottaviano wants to invite to his party ten of his closest friends, but not Turing. Everyone hates Turing, he is so petulant and arrogant. Ottaviano and his closest friends are already registred users of MeteoCal so he log in to application and starts the event creation process. Otaviano compliles all mandatory fields and carefully set the event as private, and finally invites his ten friends. Now they can easily keep in mind the party's date and Turing will never know he has been excluded, because event has been set has private and he can not see event's details.

3.4.2 Scenario 2

Dante wants to drop a big party somewhere hot. Dante log in to MeteoCal application, starts the event creation process and fills the "loction" field with "Hell". When he tries to end the event creation process and click on "confirm", the event is correctly created and saved in the database. Dante checks the weather forecast information and sees that is not available, but notices that "Hell" actually is not a valid location. He clicks on the "modify event" button and changes Hell with Palermo and saves. The event is updated with a valid location and meteo is now available.

3.4.3 Scenario 3

Leopardi wants to invite his beloved Silvia to a pic-nic in country side for declaring his love. He had check weather forecast in a meteorologic site and knows that next week there will be sunny days. Leopardi logs in into MeteoCal and starts creating event process. Leopardi imputs are valid and event is created for the 20th of july. He invites Silvia and one day later she accept the invitation. Days passes and unfortunaly the meteo forecast wasn't so accurate and meteo has change, probably it will rain. Is the 17th of july and Leopardi logs in into MeteoCal and sees a notification that informs him of bad weather prevision. The application suggest to change the date of the event to 25th of july. Leopardi decides to change the date in which the event is scheduled and set the date suggested by the application. On 19th of july Silvia logs in into MeteoCal to check if she has any new notification and sees the one about the invitation of Leopardi's event. Unfortunally she has a visit to the doctor the same date and decline the invitation. On 23th of july Leopardi logs in to MeteoCal and checks the informations of his event with Silvia and sees that he is the only participant and understands that Silvia has declined the invitation. With brocken heart Leopardi delete the event and decides to never leave his room again and dedicate his life to poetry.

3.4.4 Scenario 4

Randy is an athlete and wants to challenge Cena in a race to prove once for all who is the best. Randy logs in into MeteoCal and create a new event for the 9 of july. Soon after the creation of the event has been confirmed, the application alerts Randy that due to some technical problems the meteo forecast is not available at the moment, and it will be given as soon as possible. Randy decides to not change the date of the event but to wait for meteo forecast to be updated. Next day Randys checks his calendar and sees that weather forecast has been update and ther will be a sunny and warm day. He proceeds to invite Cena that immediately accepts. The day of the race comes and Randy wins.

3.4.5 Scenario 5

MeteoCal has been suggested to Attila as a valid application for scheduling events with meteo forecast informations. Attila wants to give a try to this new application and decides to start the regisration process. He chose "flagello" as username and then fill all the other mandatory fields. When he clicks the "confrim" button the registration process fails and the application alerts him that "flagello" is already used as username by another user, and he has to choose another username. Attila restarts registration process and this time writes "latinlover" in username field, then he clicks the "confirm" button. Registration process ends correctly and the new profile is created. Application notifies Attila that he successfully registrated to MeteoCal, and starting from now he can enjoy all the features.

3.4.6 Scenario 6

Bruce is very popular, has lots of friends and every weekend host a big party in his mansion. He makes all guests pay ten euros at the entry of the party and does not issues any recepit. Bruce is using MeteoCal for creating events and inviting people, and there are been no problems so far, then for the 23th of july he creates his 34th event of the year and invites three undred users. Bruce starts building a fortune and calls the attention of the "guardia di finanza" that starts to investigate on the 21th of july. Bruce decides that time to stop has come and to avoid any problem with the law he deletes his account of MeteoCal. Unfortunately MeteoCal dosn't provide this features so Bruce's account can not be deleted. When "guardia di finanza" asks for information about Bruce, MeteoCal developers are happy to collaborate with justice. Bruce will be captured and will spend the next 10 years in prison hoping in clemency of the President of the Republic.

3.4.7 Scenario 7

Zlatan is an outgoing guy and wants to let his friends know all the event in which he will partecipate so he set his calendar as public. Silvia falls in love with Zlatan and starts cheking daily his calendar to find where he would be, so

that she can try to catch up with him. It has been almost a month since when Silvia has start chasing Zlatan that really does not appreciate her attention. Silvia is impolite and arrogant. He is fed up with this situation and can not take it anymore. Zlatan thinks of the easiest way to avoid Silvia and decides to change visibility policy of his calendar and turn it private. Unfortunately MeteoCal dosn't provide this features. Next day Silvia logs into MeteoCal and continue to see all Zlatan events. Silvia continuos to follow Zlatan everywhere he goes. Zlatan start to appreciate her attention and efforts and suddendly, out of nowhere, fall in love with her and they lived happily ever after.

3.4.8 Scenario 8

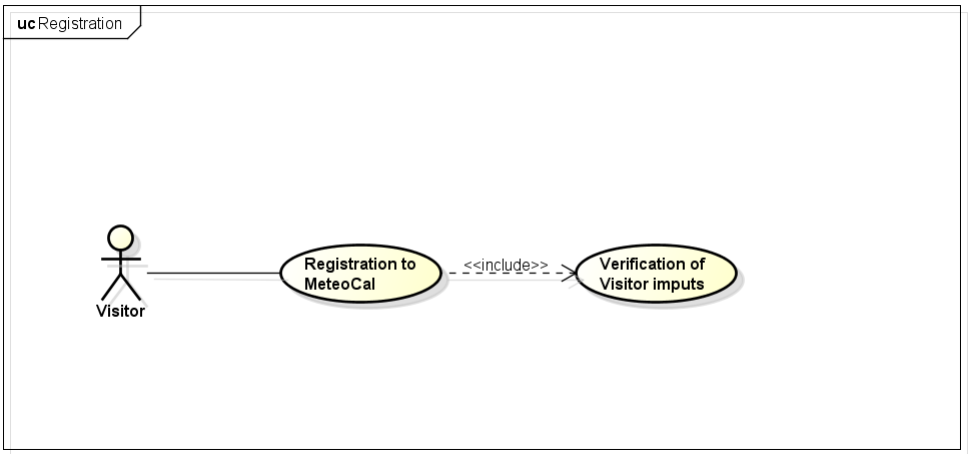
Dante and Virgilio are best friends and spend most of the time togheter, they would follow each other even in hell. Next monday it will be Dante's birthday and he is organizing a party so logs in into MeteoCal and creates a new event, then invites some friends and of course his best friend Virgilio. Four days before the birthday party, Dante and Virgilio have a big fight for silly reasons and now they are mad with each other. Dante decides that he does not want Virgilio to be at his party anymore and delete Virgilio invitation. One day before the birthday party Dante meets Virgilio, they explain themself and make it up. Dante changes his mind one last time, logs in into MeteoCal and invites again Virgilio to his party.

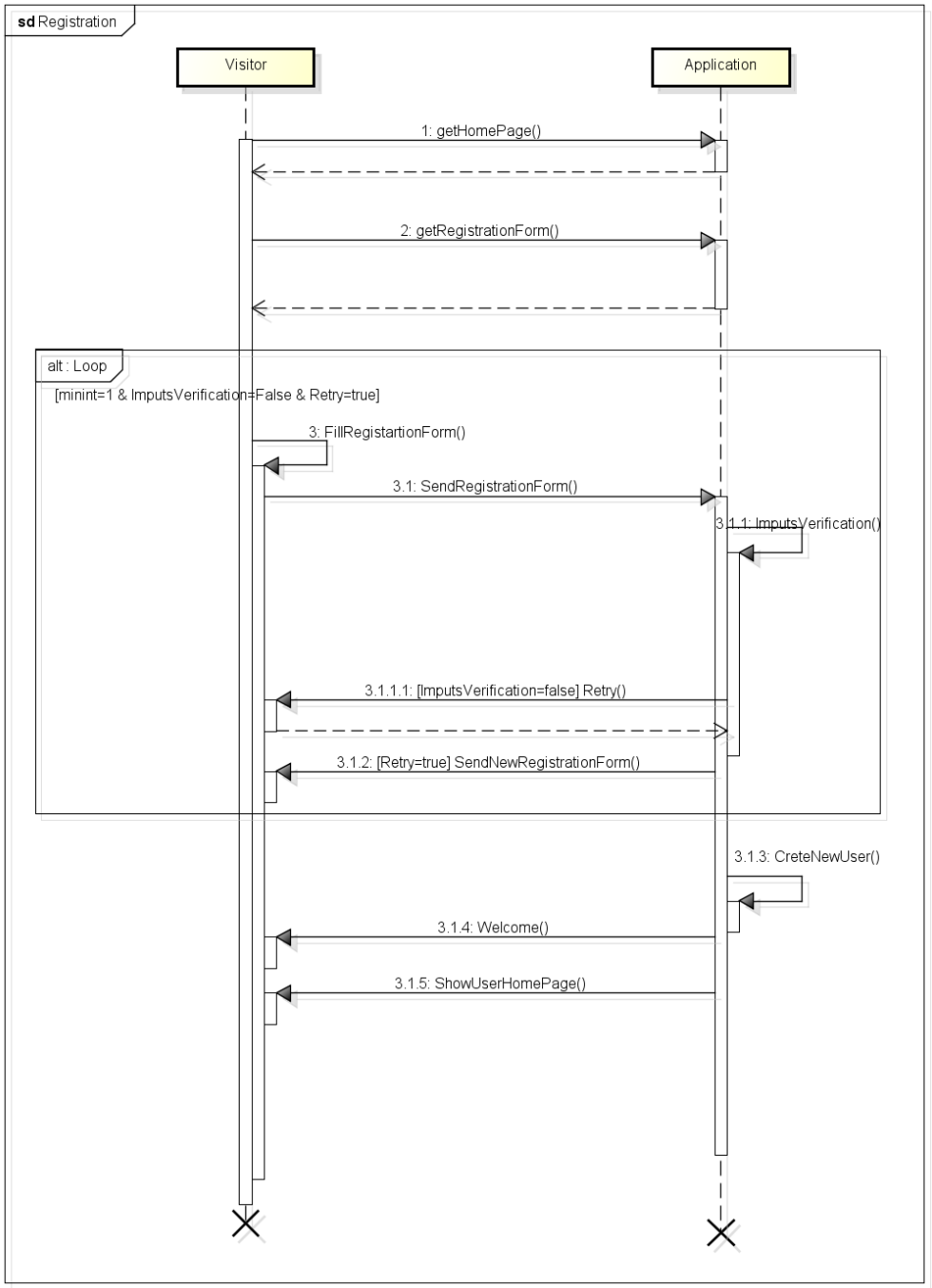
3.5 UML Models

3.5.1 Use Case

3.5.1.1 Vistiros registers to MetoCal .

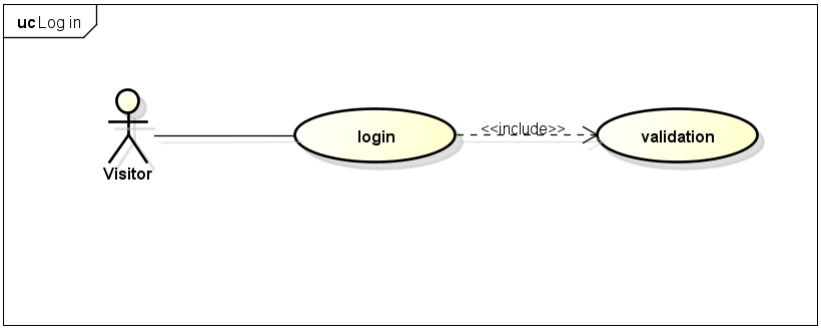
Actor	Visitor
Goal	[G1]
Imput Condition	NULL
Event Flow	1. Visitors on the home page clicks on “register”. button to start the registration process. 2. Visitor fills in at least all mandatory fields. 3. Visitors clicks on “confirm” button. 4. The application will save the date in the DB.
Otput Condition	Visitor succesfully ends registration process and become a User. From now on he/she can log in to the application using his/her credential and start using MeteoCal.
Exception	1. The visitor is already a user. 2. One or more mandatory fields are not valid. 3. Username choosen is already used by another user. 4. Email choosen is already associated to another user. All exception are handle alerting the visitor of the problem and application goes back to point 2 of EventFlow

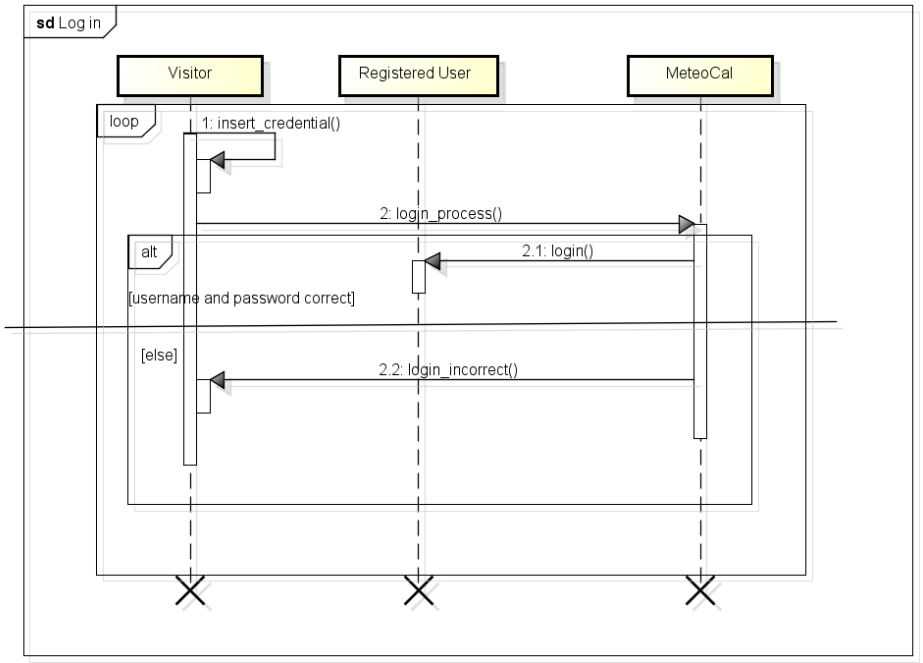

uc Registration
Visitor
Registration to MeteoCal
<<include>>
Verification of Visitor imputs


sd Registration
Visitor
Application
1: getHomePage()
2: getRegistrationForm()
alt : Loop
[minint=1 & ImputsVerification=False & Retry=true]
3: FillRegistartionForm()
3.1: SendRegistrationForm()
3.1.1: ImputsVerification()
3.1.1.1: [ImputsVerification=false] Retry()
3.1.2: [Retry=true] SendNewRegistrationForm()
3.1.3: CreteNewUser()
3.1.4: Welcome()
3.1.5: ShowUserHomePage()

3.5.1.2 login .

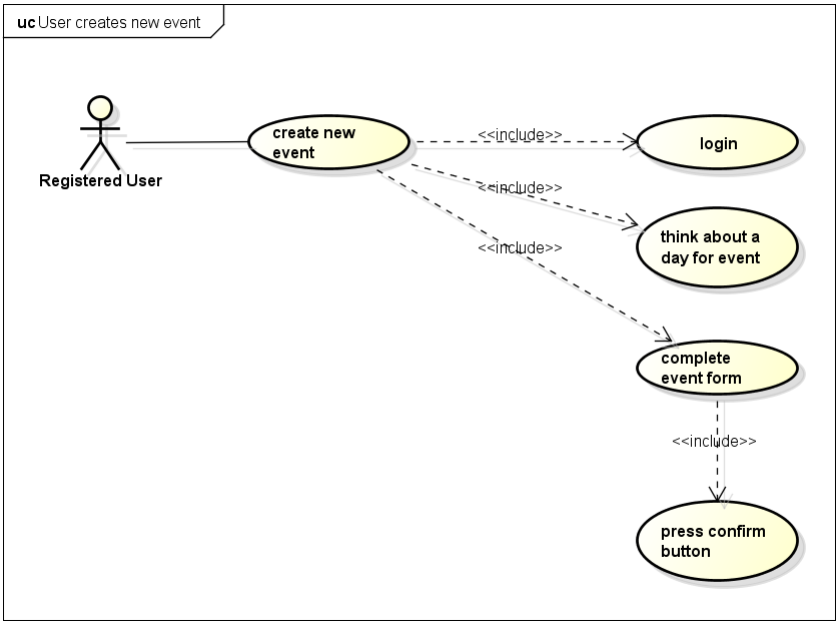
Actor	Visitor, Registered User
Goal	[G3]
Imput Condition	Visitor is registered into the system.
Event Flow	1. MeteoCal show the login page. 2. Visitor complete the form inserting correct username and password.
Otput Condition	1. MeteoCal verify the credential of visitor and if correct show the main page with calendar. 2. Visitor are promote to a registered user.
Exception	If username and/or password are incorrect MeteoCal notify that to visitor and show the login page.

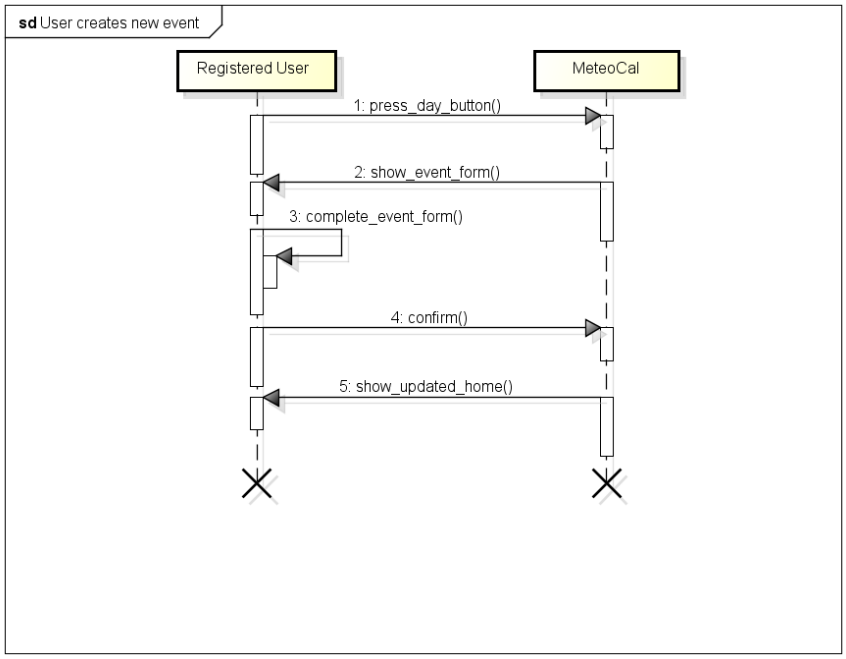



sd Log in
Visitor
Registered User
MeteoCal
loop
1: insert_credential()
2: login_process()
alt
2.1: login()
[username and password correct]
[else]
2.2: login_incorrect()

3.5.1.3 Create new event on calendar .

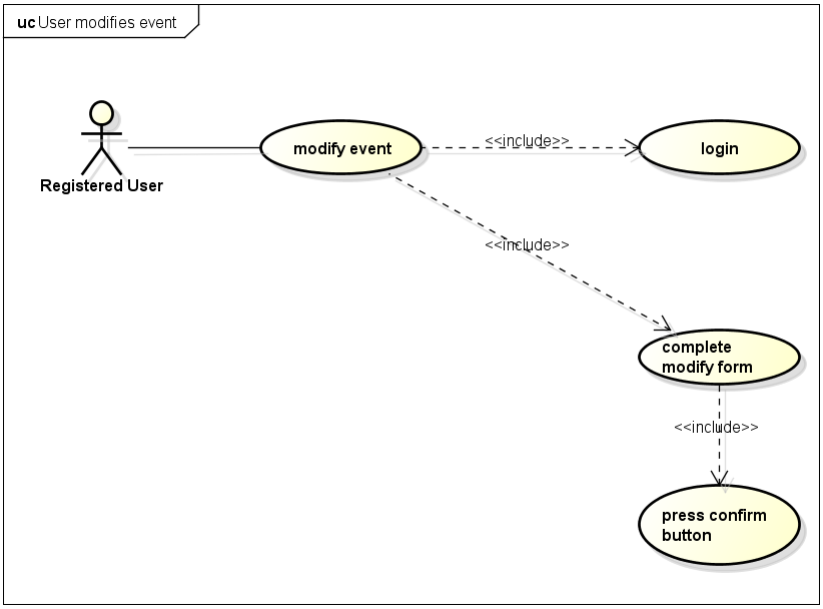
Actor	Registered User
Goal	[G5]
Imput Condition	Registered User is already logged in into MeteoCal.
Event Flow	1. Registered User click on a day of calendar. 2. Next click on "create event" and start the creating process. 3. MeteoCal show a new page with form. User complete form.
Output Condition	MeteoCal show the calendar main page with the new event.
Exception	User want to cancel the creating process. A cancel button is aviable.

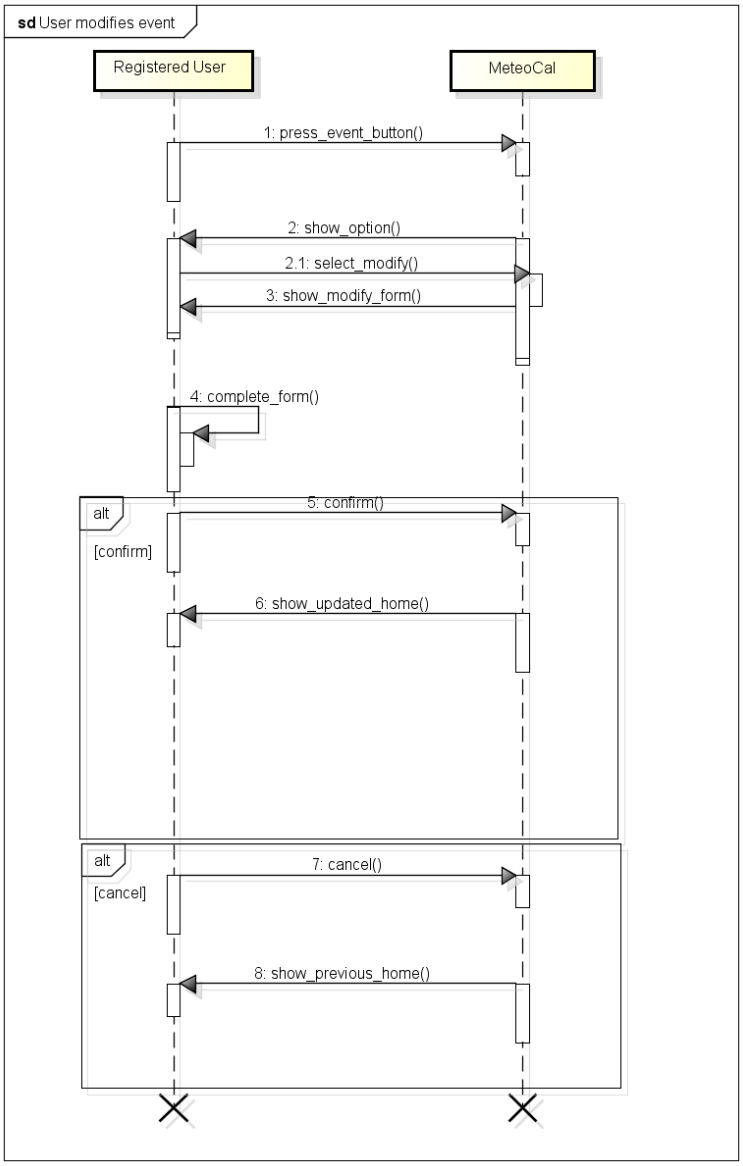



sd User creates new event
Registered User
MeteoCal
1: press_day_button()
2: show_event_form()
3: complete_event_form()
4: confirm()
5: show_updated_home()

3.5.1.4 User modifies an event .

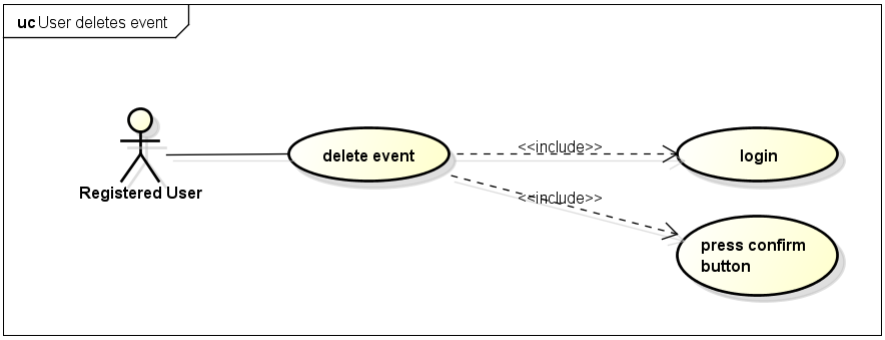
Actor	Registered User.
Goal	[G6]
Imput Condition	Registered User is already logged in into MeteoCal.
Event Flow	1. Registered User push on an existing event of calendar. 2. Next to "modify event" and start the modifyng process. 3. MeteoCal show a new page with form filled with existing information of event. 4. User modify form.
Output Condition	MeteoCal show the calendar main page with the modified event updated.
Exception	User want to cancel the modifyng process. A cancel button is aviable.

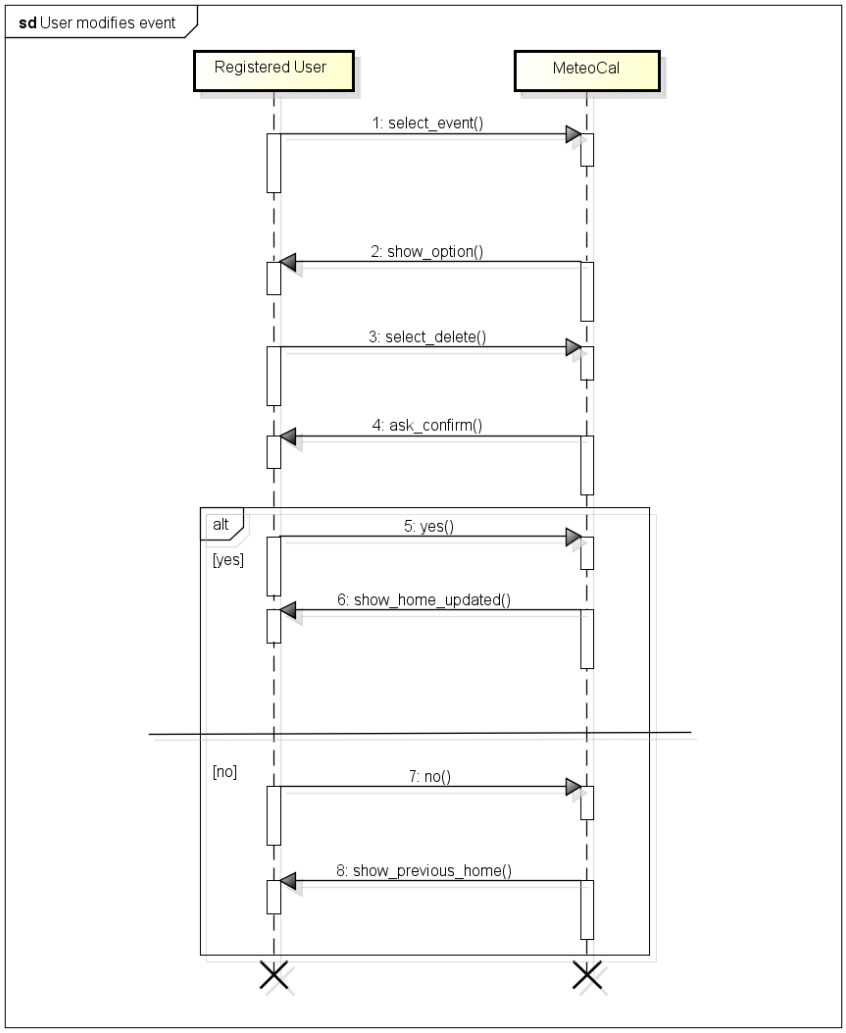



sd User modifies event
Registered User
MeteoCal
1: press_event_button()
2: show_option()
2.1: select_modify()
3: show_modify_form()
4: complete_form()
alt
[confirm]
5: confirm()
6: show_updated_home()
alt
[cancel]
7: cancel()
8: show_previous_home()

3.5.1.5 User deletes an event from calander .

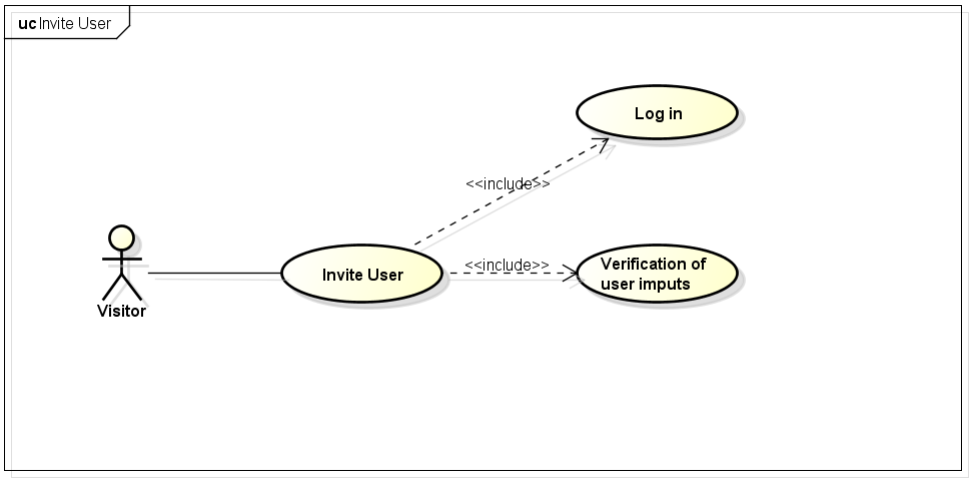
Actor	Registered User.
Goal	[G7]
Imput Condition	Registered User is already logged in into MeteoCal.
Event Flow	1. Registered User push on an existing event of calendar. 2. Next on “delete event”.
Output Condition	MeteoCal show the calendar main page without the event that user choose to delete.
Exception	NULL

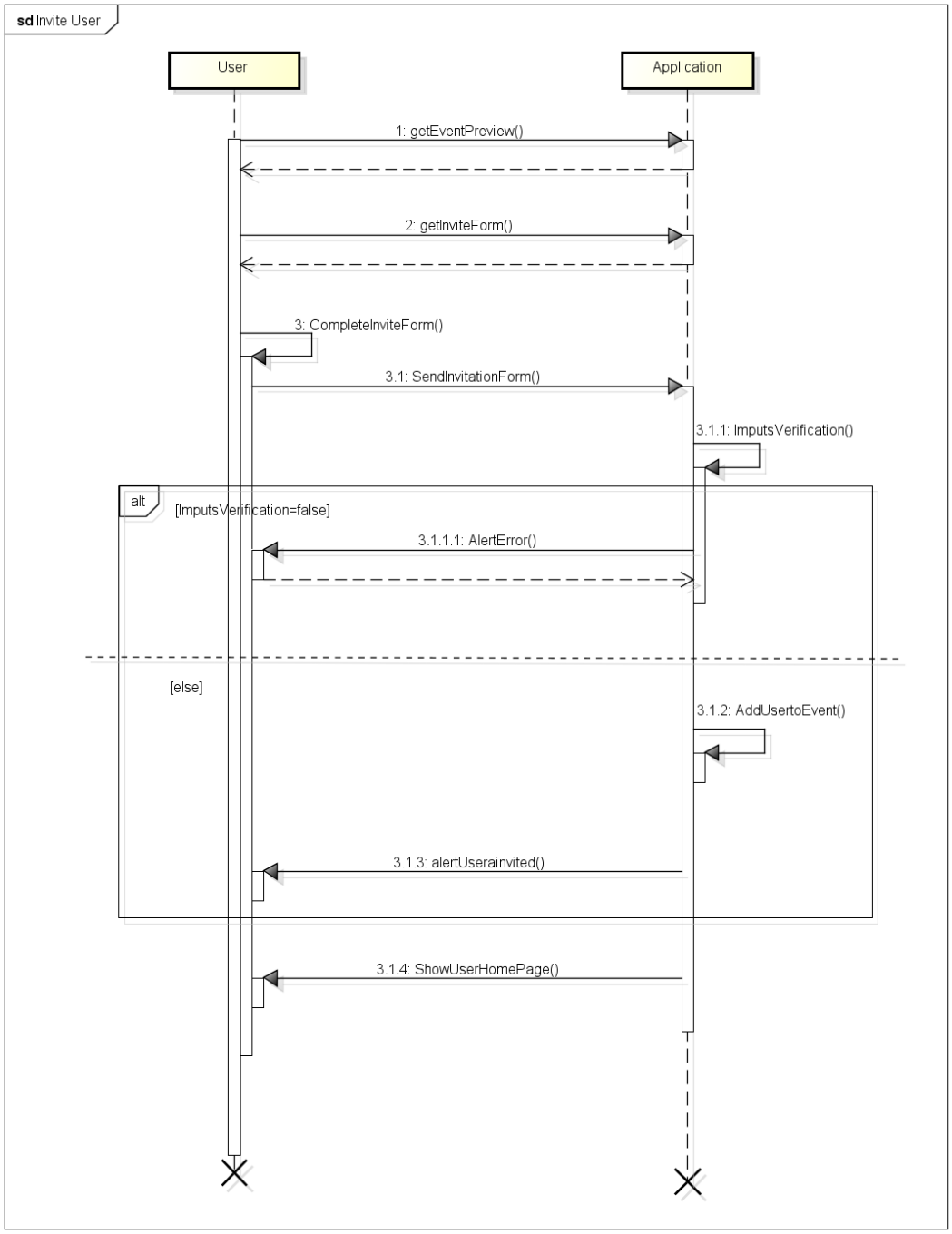



sd User modifies event
Registered User
MeteoCal
1: select_event()
2: show_option()
3: select_delete()
4: ask_confirm()
alt
[yes]
5: yes()
6: show_home_updated()
[no]
7: no()
8: show_previous_home()

3.5.1.6 User invites other user to an event .

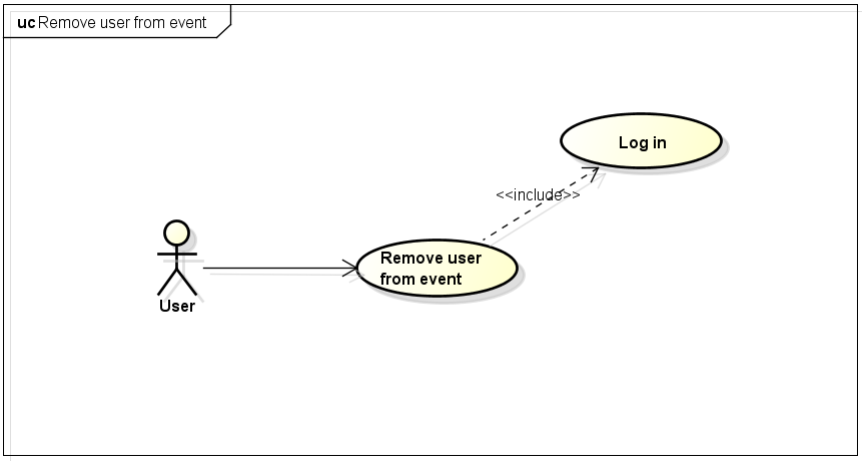
Actor	User
Goal	[G8]
Imput Condition	1. User must be registred. 2. User must be logged in into application. 3. User is in the calendar home page.
Event Flow	1. User clicks on the event colored circle. 2. User clicks on “invite” button. 3. User searchs for users who wants to invite. 4. User confirm invitation.
Output Condition	The invited user receive an invitation alert.
Exception	1. Appllication was not able to invite the user due to some internal issues. All exception are handle alerting the user of the problem and application goes back to calendar home page.

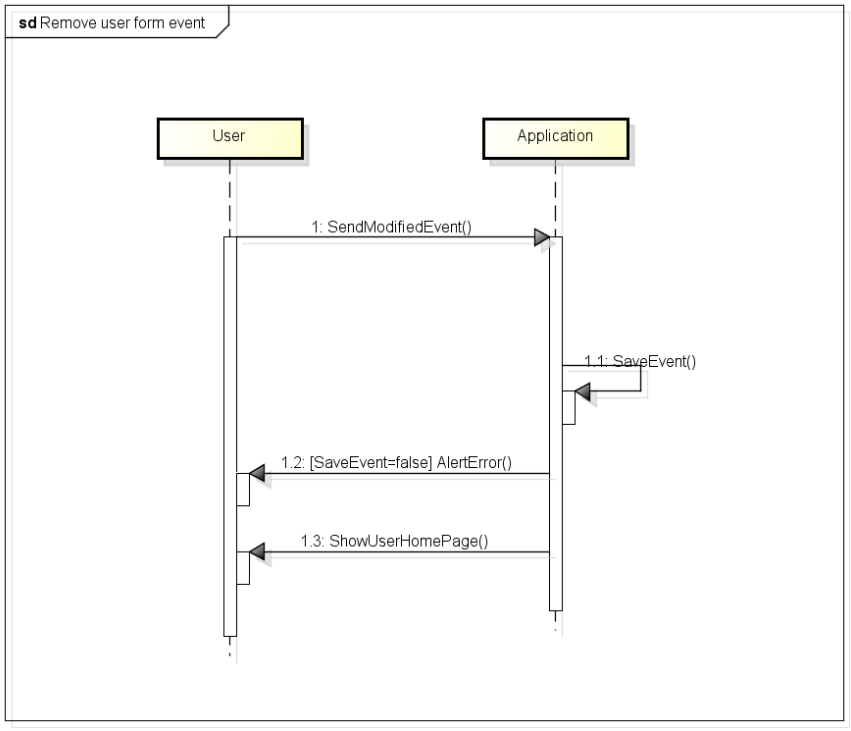



sd Invite User
User
Application
1: getEventPreview()
2: getInviteForm()
3: CompleteInviteForm()
3.1: SendInvitationForm()
3.1.1: ImputsVerification()
alt
[ImputsVerification=false]
3.1.1.1: AlertError()
[else]
3.1.2: AddUsertoEvent()
3.1.3: alertUserainvited()
3.1.4: ShowUserHomePage()

3.5.1.7 Remove user from an event guest list .

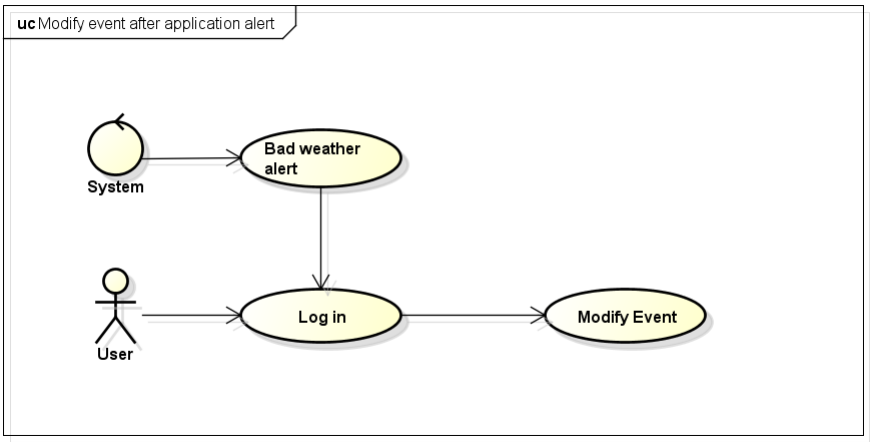
Actor	User
Goal	[G8]
Imput Condition	1. User must be registred. 2. User must be logged in into application. 3. User is in the modify event page.
Event Flow	1. User "uncheck"the usernames of the users he/she wants to remove from the guest list. 2. User confirms.
Output Condition	Users that have been unchecked are removed from the guest list and the event is updated.
Exception	1. Appllication was not able to complete remove operation the due to some internal issues. All exception are handle alerting the user of the problem and application goes back to calendar home page.

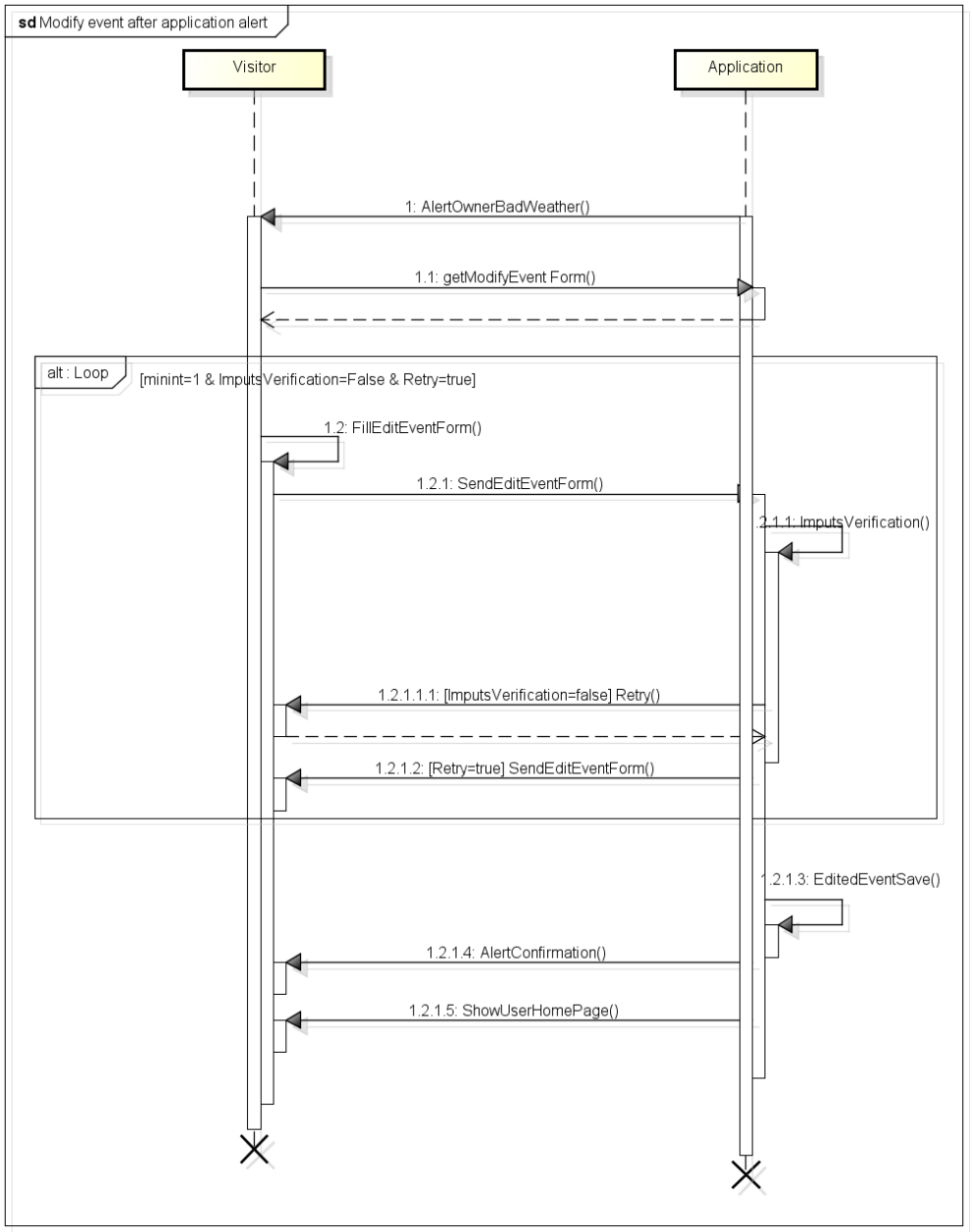



sd Remove user form event
User
Application
1: SendModifiedEvent()
1.1: SaveEvent()
1.2: [SaveEvent=false] AlertError()
1.3: ShowUserHomePage()

3.5.1.8 User modify the event date after application has notified him/her that there will be bad weather and suggest him/her the closest sunny day. .

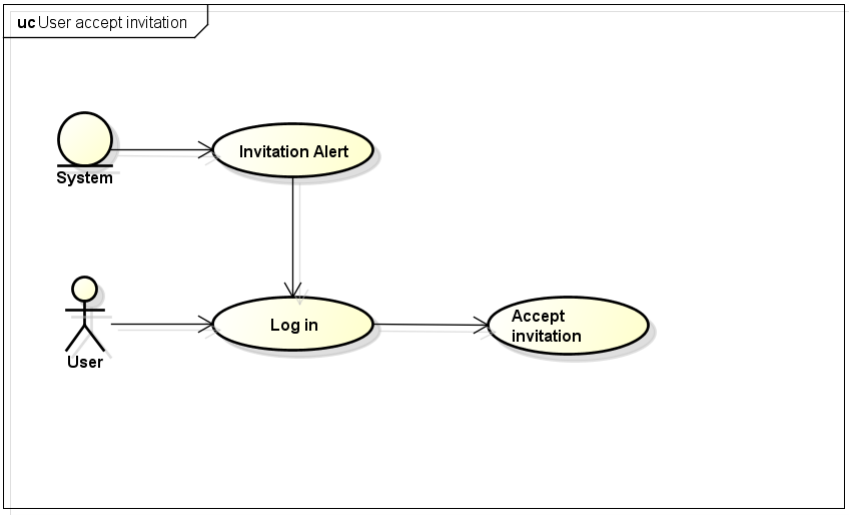
Actor	User
Goal	Modify an event date thanks to the suggestion of the application
Imput Condition	1. User must be registred. 2. User must be logged in into application.
Event Flow	1. Application notifies three days before the event will take place that there will be weather. 2. Application suggest the closest sunny day. 3. User accepts to modify the event date.
Output Condition	Event has been modified.
Exception	1. One or more mandatory fields are not valid. 2. Appllication was not able to complete event update operation due to some internal issues. All exception are handle alerting the user of the problem and application goes back to point 3 of Event Flow

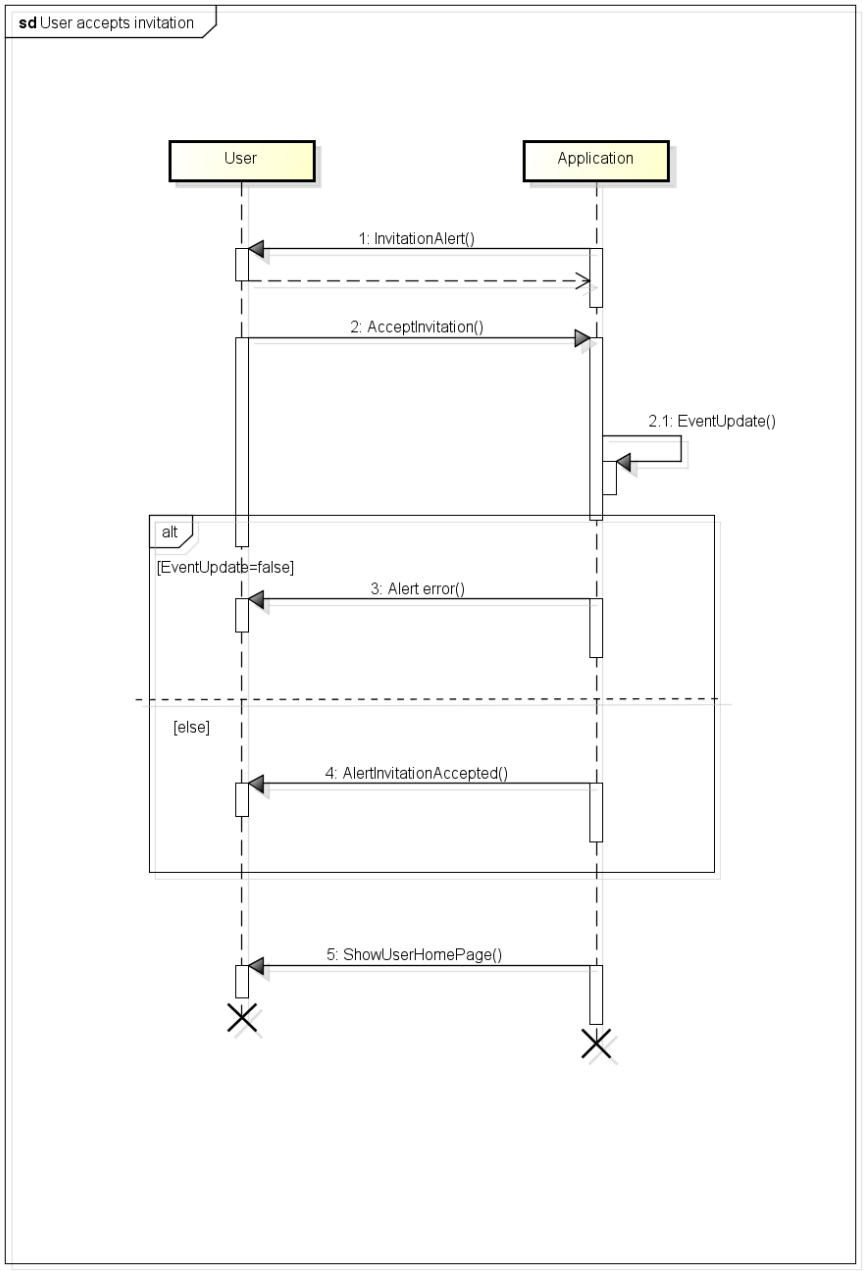



sd Modify event after application alert
Visitor
Application
1: AlertOwnerBadWeather()
1.1: getModifyEvent Form()
alt : Loop
[minint=1 & ImputsVerification=False & Retry=true]
1.2: FillEditEventForm()
1.2.1: SendEditEventForm()
.2.1.1: ImputsVerification()
1.2.1.1.1: [ImputsVerification=false] Retry()
1.2.1.2: [Retry=true] SendEditEventForm()
.2.1.3: EditedEventSave()
1.2.1.4: AlertConfirmation()
1.2.1.5: ShowUserHomePage()

3.5.1.9 User accept invitation .

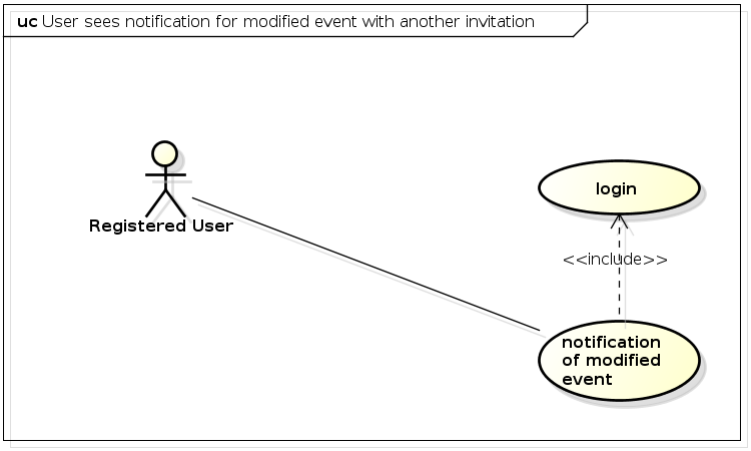
Actor	User
Goal	User accept an invitation for an event he/she has been invited.
Imput Condition	1. User must be registred. 2. User must be logged in into application.
Event Flow	1. Application notifies the user the he/she has been invited to an event and asks to accept. 2. User accepts the invitation for the event.
Output Condition	User is now a guest of the event, he/she can see the event in their calendar home page.
Exception	1. Appllication was not able to complete event update operation due to some internal issues. All exception are handle alerting the user of the problem and application goes back to user's calendar home page.

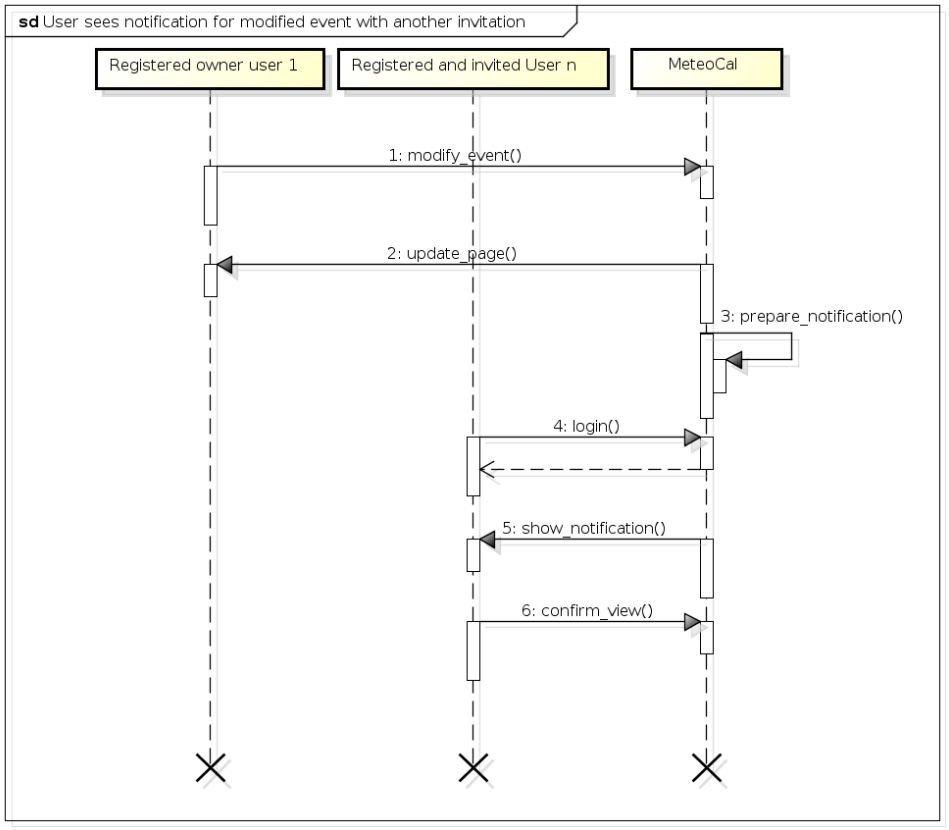



sd User accepts invitation
User
Application
1: InvitationAlert()
2: AcceptInvitation()
2.1: EventUpdate()
alt
[EventUpdate=false]
3: Alert error()
[else]
4: AlertInvitationAccepted()
5: ShowUserHomePage()

3.5.1.10 User sees notification for modified event with another invitation .

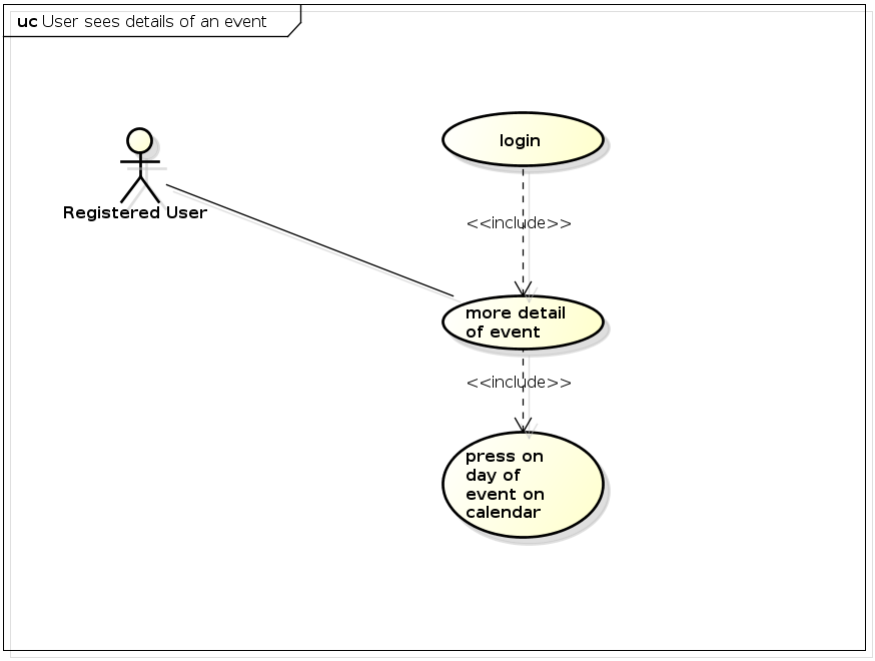
Actor	Registered Users
Goal	User sees notification for modified event with another invitation.
Imput Condition	Registered Users log in into MeteoCal.
Event Flow	1. MeteoCal show a notification to user who have been invited to an event when the event is modified by the owner asking for a new confirm. 2. User press “ok” button to notify.
Output Condition	MeteoCal show the main calendar page.
Exception	NULL

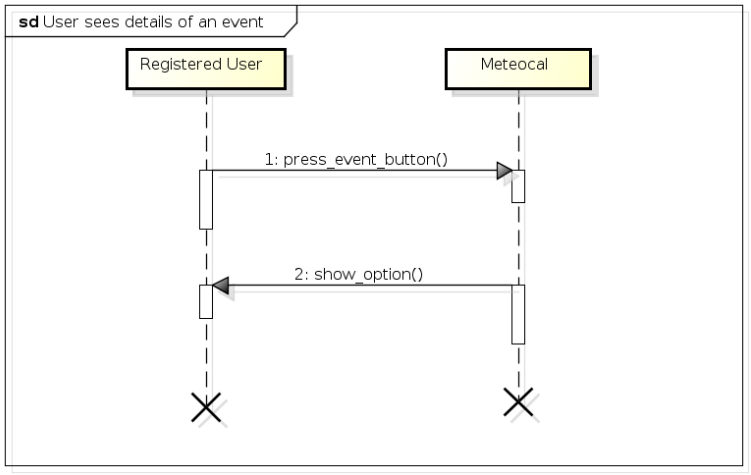



sd User sees notification for modified event with another invitation
Registered owner user 1
Registered and invited User n
MeteoCal
1: modify_event()
2: update_page()
3: prepare_notification()
4: login()
5: show_notification()
6: confirm_view()

3.5.1.11 User sees details of an event .

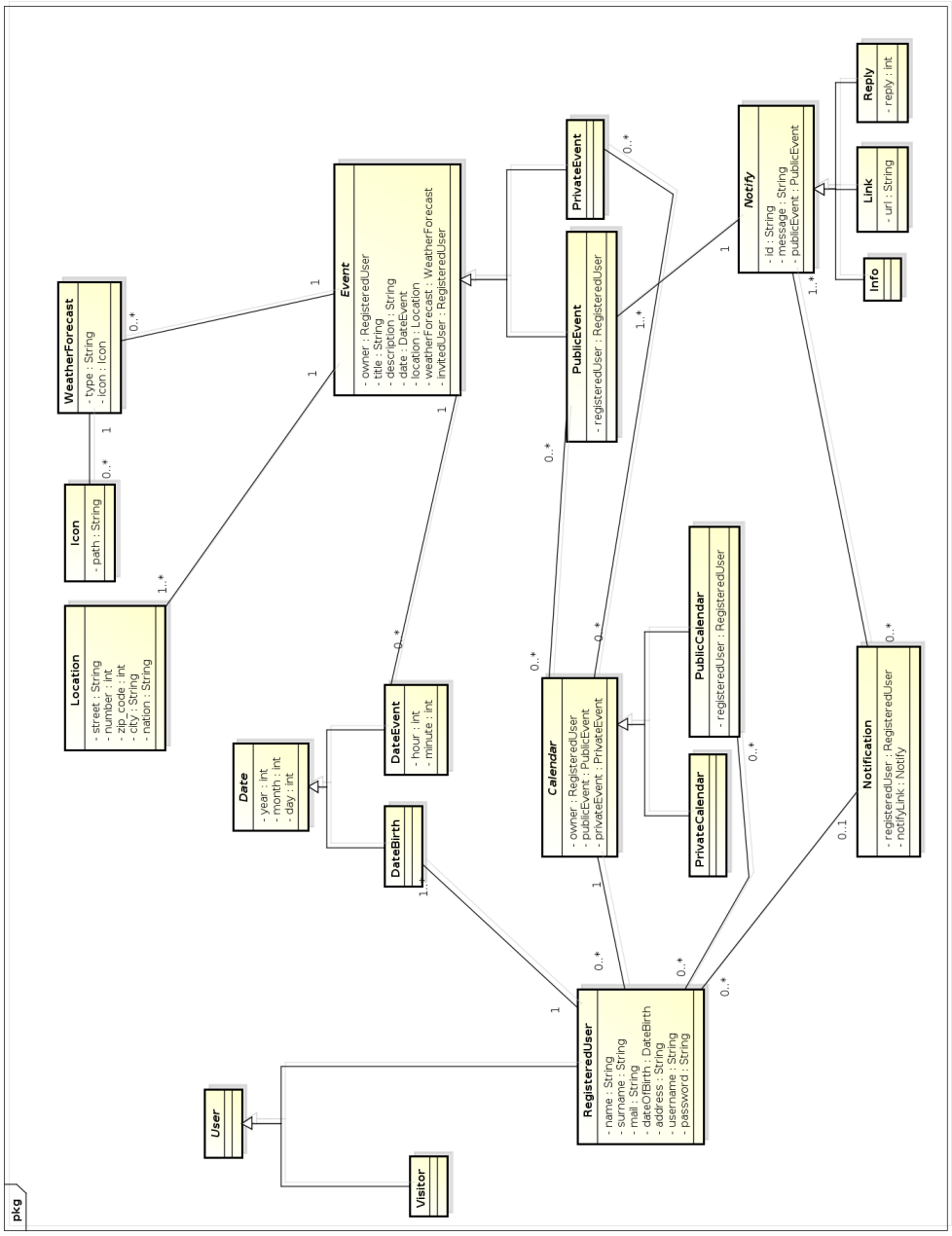
Alert	Registered User.
Goal	User sees "other details" of an event.
Imput Condition	Registered User is already logged in into MeteoCal.
Event Flow	1. Registered User click on an existing event of calendar and see detail if he/she can beacuse of the visibility policy.
Output Condition	MeteoCal show a new page with all detail about the event.
Exception	NULL




sd User sees details of an event
Registered User
Meteocal
1: press_event_button()
2: show_option()

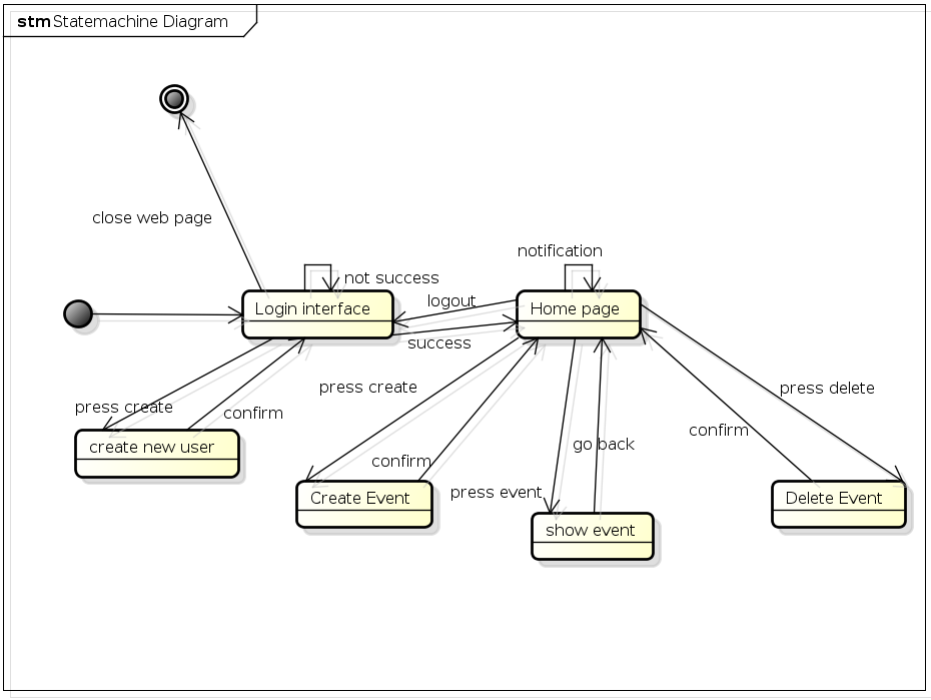
3.5.2 Class Diagrams

Here is presented the class diagram. This diagram will be updated during the developing process expecially adding all method.



3.5.3 State Machine Diagrams

The follow state machine diagram would give a simplify vision of entair application.



3.6 Non Functional Requirements

3.6.1 Performance Requirements

Performance must be acceptable to garantee a good grade of usability. We assume the response time of the system is close to zero, so the performance are essentialy bounded by users internet connection.

3.6.2 Design Constraints

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

3.6.3 Software System Attributes

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

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

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

3.6.4 Security

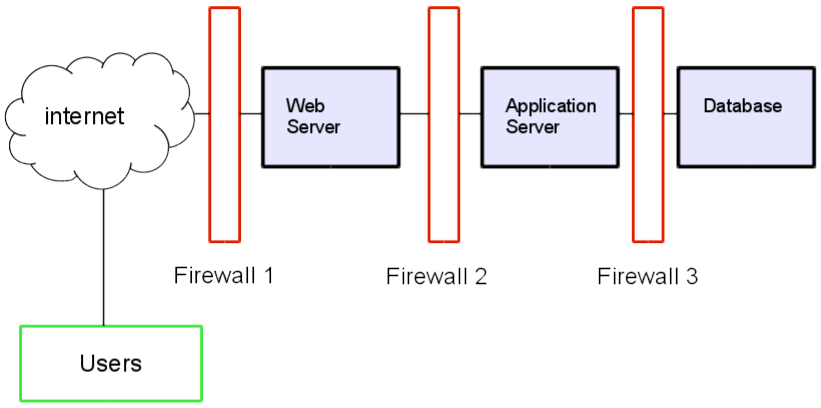
3.6.4.1 External Interface Side MeteoCal application implements a login authentication to protect the information of users. Password of user is saved using hashing mechanism but could not be enough. This system do not actually require anything about the strongness of the password so could be developed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

- Two-factor authentication whith a code sent by email or sms to the user
- Smart-cart authentication, a card with a chip used to authenticate the user through a smart card reader.

- Dyno tecnologies or one-time password, composed by a card static password list owned by user or a dyamic embedded password generator. The system ask for random code on that card or an entair code generated.
- Biometric: fingerprint, retina scan, voice analysis

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

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



4 Appendix

4.1 Alloy

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

4.1.1 Data Type

This is the definition of data type.

```
//---------DATA TYPE-------//------------SIGNATURE----------//
sig Integer{}
sig Strings{}
sig Location{
    street: one Strings,
    number: one Integer,
    zipCode: one Integer,
    city: one Strings,
    nation: one Strings
}

sig Icon{
    path: one Strings
}

sig WeatherForecast{
    type: one Strings,
    icon: one Icon
}
```

4.1.2 Abstract Entity

This is the definition of abstract entity.

```
//---------ABSTRACT ENTITY-------//------------SIGNATURE----------//

//abstract class of calendar
abstract sig Calendar{
    owner: one RegisteredUser,
    publicEvent: set PublicEvent,
    privateEvent: set PrivateEvent
}

//abstract class of user
abstract sig User{}

//abstract class of event
abstract sig Event{
    owner: one RegisteredUser,
    title: one Strings,
    description: one Strings,
    date: one DateEvent,
    place: one Location,
    weatherForecast: lone WeatherForecast,
    invitedUser: some RegisteredUser
}

//abstract (generic date class)
abstract sig Date{
    year: one Integer,
    month: one Integer,
    day: one Integer,
}

//abstract calss of notify
abstract sig Notify{
    id: one Strings,
    message: one Strings
}
```

4.1.3 Abstrac Entity Implementation and Signature

This is the implementation of some abstract entity and other signature.

```
//---------DEFINITION OF ABSTRACT ENTITY-------//------------SIGNATURE----------//

//class notification

sig Notification{
   user: one RegisteredUser,
   notifyLink: some NotifyLink,
   notifyReply: some NotifyReply,
   notifyInfo: some NotifyInfo
}

//visitor and registered user are disjoint
sig Visitor extends User{}

//user after registration
sig RegisteredUser extends User{
   name: one Strings,
   surname: one Strings,
   mail: one Strings,
   dateOfBirth: one DateBirth,
   address: one Strings,
   username: one Strings,
   password: one Strings,
}

//date for birth
sig DateBirth extends Date{}

//date  for the event
sig DateEvent extends Date{
   hour: one Integer,
   minutes: one Integer
}

//public event
sig PublicEvent extends Event{
   registeredUser: some RegisteredUser
}
```

```
//private event
sig PrivateEvent extends Event{}

//private calendar
sig PrivateCalendar extends Calendar{}

//public calendar
sig PublicCalendar extends Calendar{
    user: some RegisteredUser
}

//notify with link
sig NotifyLink extends Notify{
    link: one Strings
}

//notify with reply
sig NotifyReply extends Notify{
 reply: one Integer
}

//notify with information
sig NotifyInfo extends Notify{}
```

4.1.4 Fact

This is the fact part that defines the costraint of the class.

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

//no empity date
fact noEmpityDate{
    all d : Date | (#d.year=1) and (#d.month=1) and (#d.day=1)
}

//no empity date for event
fact noEmpityDateEvent{
    all d : DateEvent | (#d.hour=1) and (#d.minutes=1)
}

//no empity location
fact noEmpityLocation{
    all l : Location | (#l.street=1) and (#l.number=1) and (#l.zipCode=1) and (#l.city=1) and (#l.nation=1)
}

//no empity calendar
fact noEmpityCalendar{
    all c : Calendar | (#c.owner=1)
}

//no empity event
fact noEmpityEvent{
    all e : Event | (#e.owner=1) and (#e.title=1) and (#e.description=1) and (#e.date=1) and (#e.place=1)
}

//no empity notify
fact noEmpityNotify{
    all n: Notify | (#n.message=1)
}

//no duplicate id notify
fact noDuplicateNotify{
    no disj n1, n2: Notify | n1.id=n2.id
}

//no duplicate user, impose no duplicate mail and no duplicate username
fact noDuplicateUser{
     no disj u1, u2: RegisteredUser | (u1.mail=u2.mail) and  (u1.username=u2.username)
}

//limit inviting User one time
fact noDoubleInvite{
    no disj u1,u2:RegisteredUser, e: PublicEvent | e.invitedUser=u1 and e.invitedUser=u2 and u1=u2
}

//impose no self invite to an event
fact noSelfInvite{
    all u: RegisteredUser, e: PublicEvent | not ((e.invitedUser=u) and (e.owner=u))
}
```

4.1.5 Assert

In this last code part is presented the assert used to verify the model.

```
//---------ASSERT-------//


//no self invite to an event
assert noSelfInvites{
   no u: RegisteredUser, e: PublicEvent | (e.invitedUser=u) and (e.owner=u)
}
check noSelfInvites for 5

//check no double user invited to same event
assert noDoubleInvites{
   no disj u1,u2: RegisteredUser, e: PublicEvent |  e.invitedUser=u1 and e.invitedUser=u2 and u1=u2
}
check noDoubleInvites for 5

//check delete event pred
assert deleteEvent{
   all e: Event, c1, c2: Calendar | (e in c1.publicEvent) and deletePublicEvents[e,c1,c2] implies (e not in c2.publicEvent)
   all e: Event, c1, c2: Calendar | (e in c1.privateEvent) and deletePrivateEvents[e,c1,c2] implies (e not in c2.privateEvent)
}
check deleteEvent

//check add event pred
assert addEvent{
   all e: Event, c1, c2: Calendar | (e not in c1.publicEvent) and addPublicEvents[e,c1,c2] implies (e in c2.publicEvent)
   all e: Event, c1, c2: Calendar | (e not in c1.privateEvent) and addPrivateEvents[e,c1,c2] implies (e in c2.privateEvent)
}
check addEvent for 5

//check delete user pred
assert deleteUser{
   all u, u1, u2: RegisteredUser | (u in u1) and deleteUser[u, u1, u2] implies (u not in u2)
}
check deleteUser for 5

//check add user pred
assert addUser{
   all u, u1, u2: RegisteredUser | (u not in u1) and addUser[u, u1, u2] implies (u in u2)
}
check addUser for 5
```

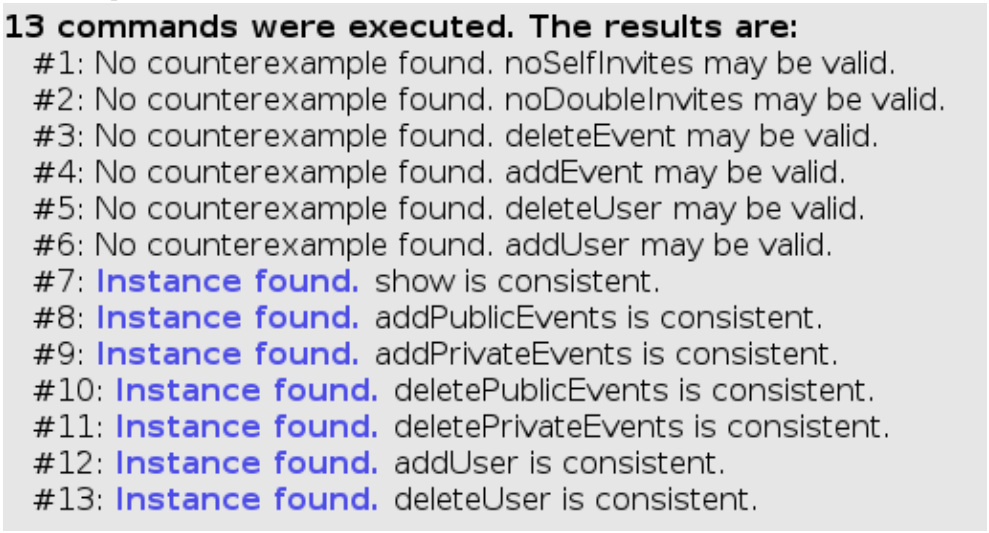
4.1.6 Predicates

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

```
//---------PREDICATES-------//

pred show(){
#RegisteredUser>1
#PublicEvent>1
#PrivateEvent>1
#NotifyInfo>1
}
run show for 30

//pred for add public event
pred addPublicEvents( e: PublicEvent, c1, c2: Calendar){
   e not in c1.publicEvent implies c2.publicEvent=c1.publicEvent+e
}
run addPublicEvents for 5

//pred for add private event
pred addPrivateEvents( e: PrivateEvent, c1, c2: Calendar){
   e not in c1.privateEvent implies c2.privateEvent=c1.privateEvent+e
}
run addPrivateEvents for 5

//pred for delete public event
pred deletePublicEvents( e: PublicEvent, c1, c2: Calendar){
   c2.publicEvent=c1.publicEvent-e
}
run deletePublicEvents for 5

//pred for delete private event
pred deletePrivateEvents( e: PrivateEvent, c1, c2: Calendar){
   c2.privateEvent=c1.privateEvent-e
}
run deletePrivateEvents for 5
```

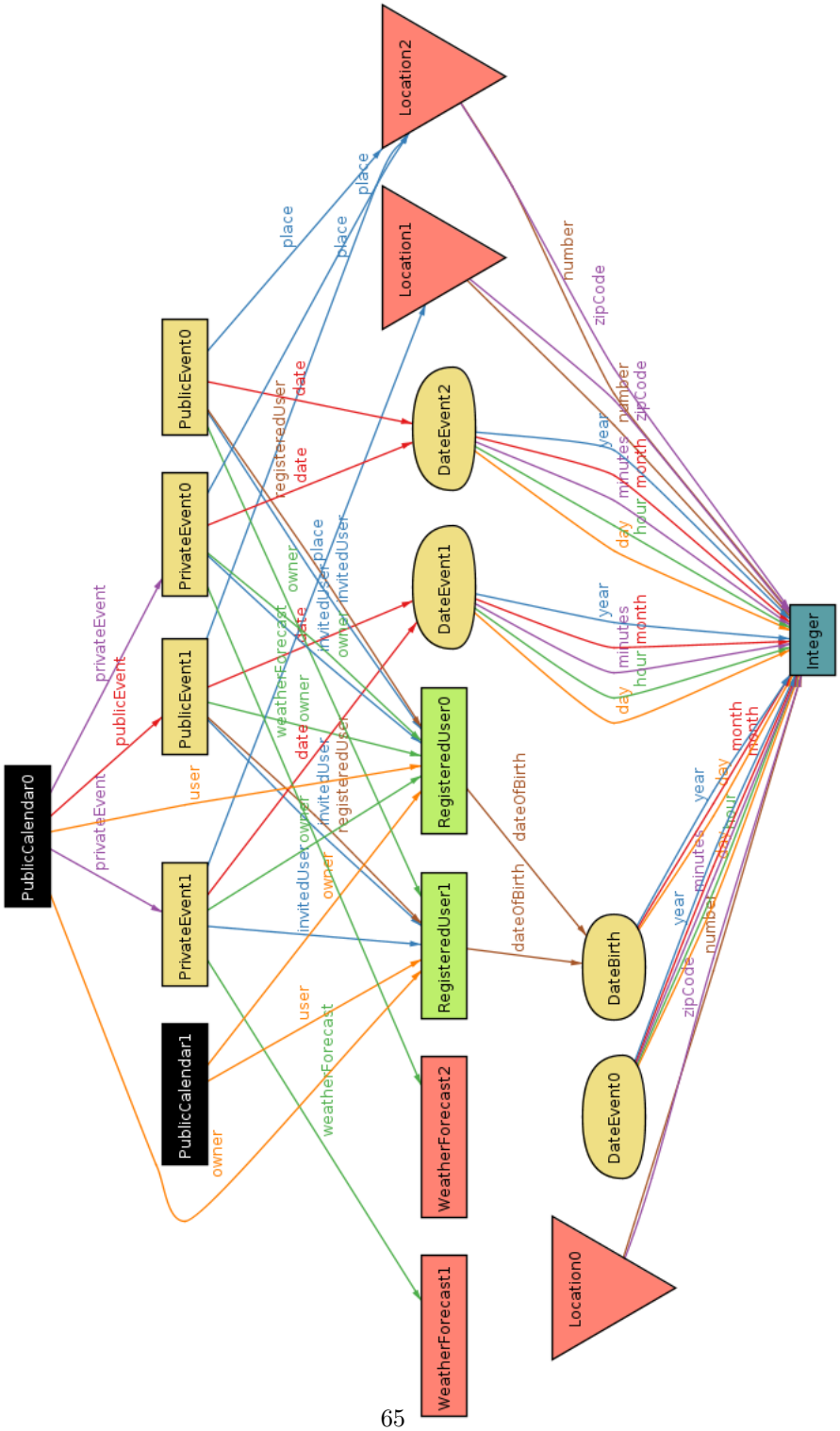
4.1.7 Result

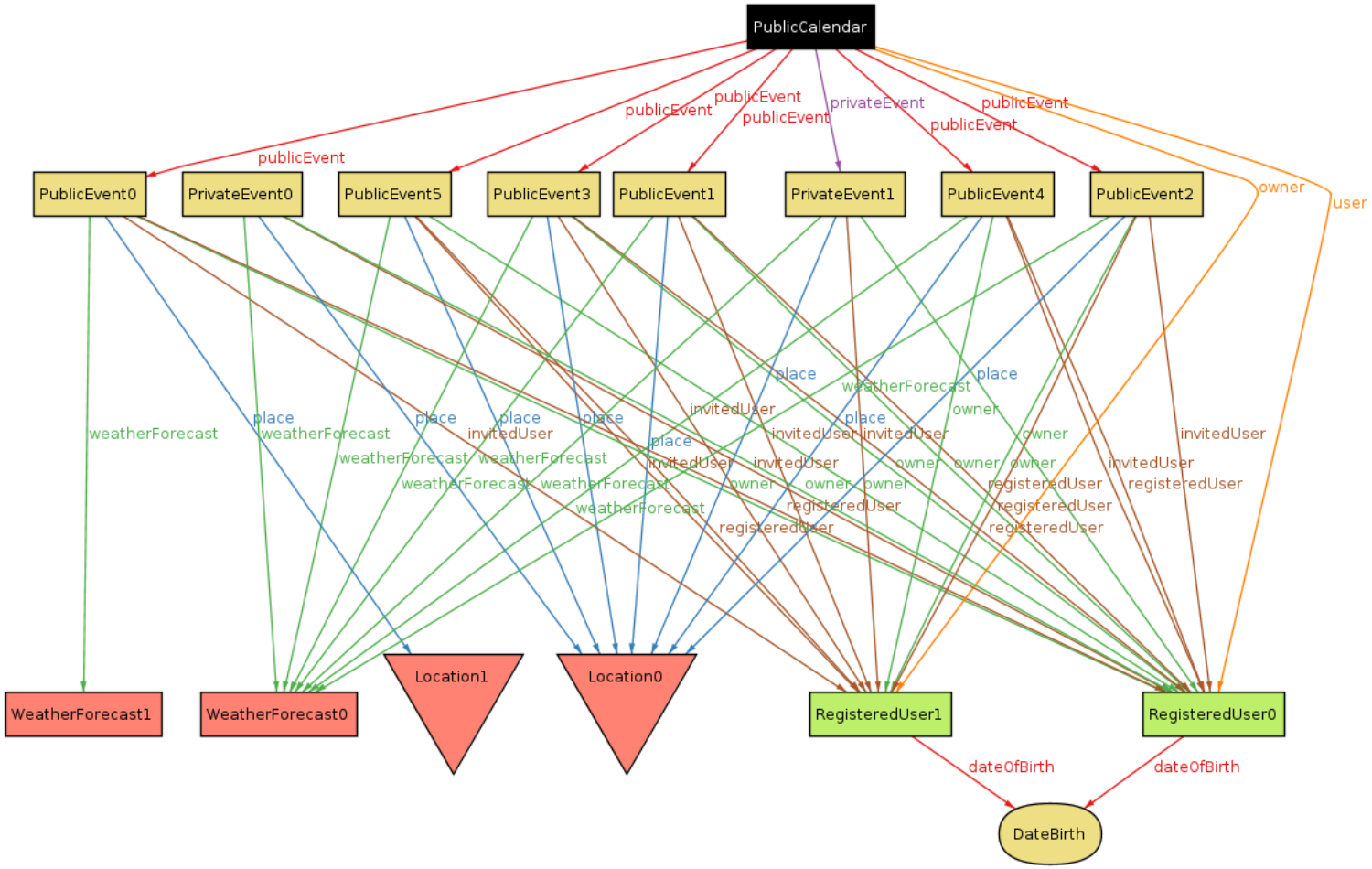
This screenshot of the Alloy Analizer software that shows the consistence of the model in all part.

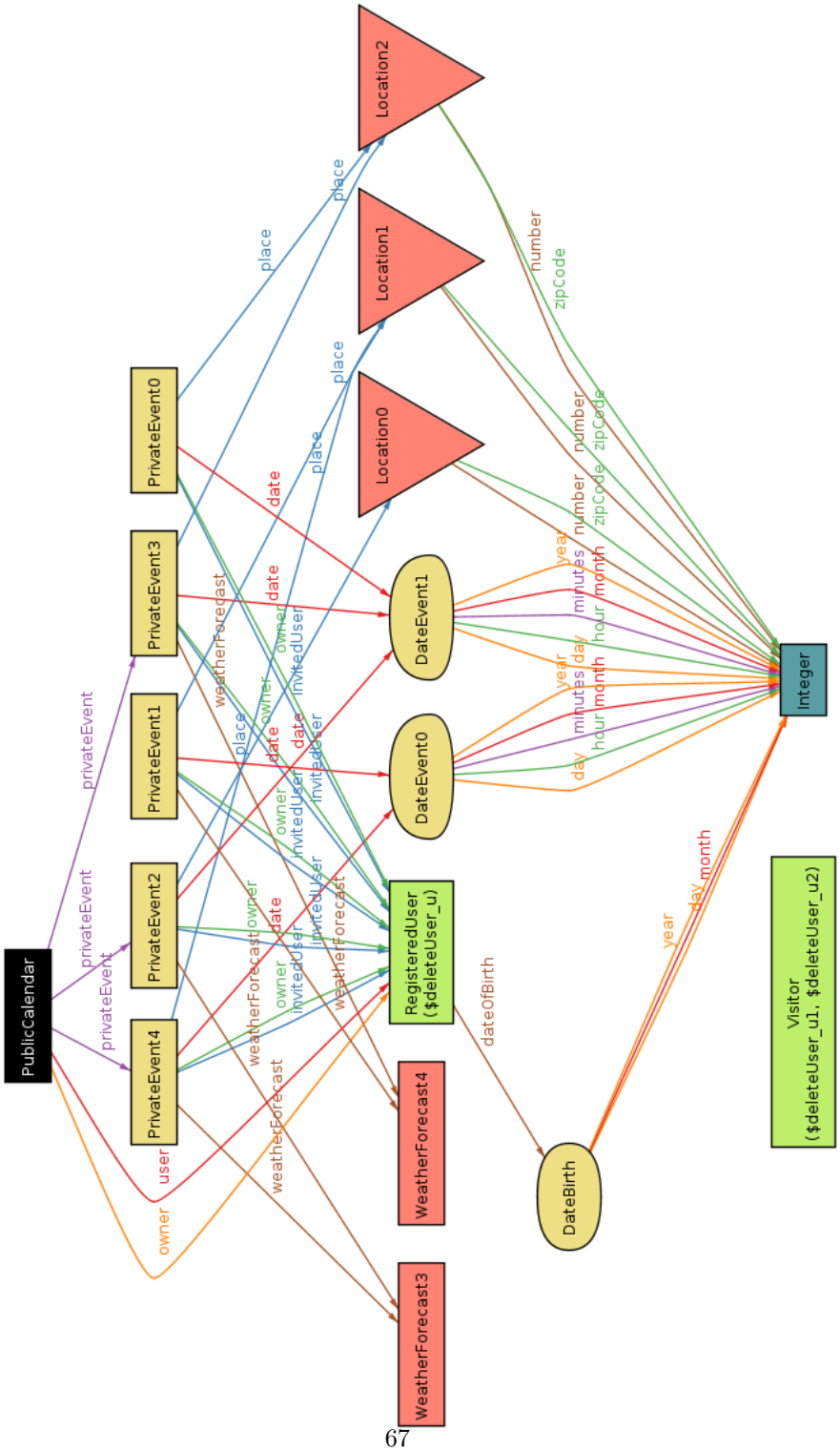


4.1.8 Generated world

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


PublicCalendar0
PublicCalendar1
PrivateEvent0
PrivateEvent1
PublicEvent0
PublicEvent1
RegisteredUser0
RegisteredUser1
WeatherForecast2
WeatherForecast1
DateEvent0
DateEvent1
DateEvent2
DateBirth
Location0
Location1
Location2
Integer
privateEvent
publicEvent
user
owner
place
date
registeredUser
invitedUser
weatherForecast
dateOfBirth
number
zipCode
year
minutes
day
hour
month


PublicCalendar
publicEvent
publicEvent
publicEvent
publicEvent
privateEvent
publicEvent
publicEvent
owner
user
PublicEvent0
PrivateEvent0
PublicEvent5
PublicEvent3
PublicEvent1
PrivateEvent1
PublicEvent4
PublicEvent2
weatherForecast
place
invitedUser
owner
registeredUser
Location1
Location0
WeatherForecast1
WeatherForecast0
RegisteredUser1
RegisteredUser0
dateOfBirth
dateOfBirth
DateBirth


PublicCalendar
PrivateEvent0
PrivateEvent1
PrivateEvent2
PrivateEvent3
PrivateEvent4
Location0
Location1
Location2
DateEvent0
DateEvent1
RegisteredUser
($deleteUser_u)
DateBirth
Visitor
($deleteUser_u1, $deleteUser_u2)
Integer
WeatherForecast3
WeatherForecast4
privateEvent
user
owner
place
date
owner
invitedUser
weatherForecast
dateOfBirth
year
month
day
hour
minutes
number
zipCode

4.2 Software and tool used

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

4.3 Hours of works

This is the time spent for redact this document:

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

5 Revision

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

5.1 Changed Assumptions

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

5.2 Removed goal and functionality

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

5.3 Modified Functional Requirements

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

  - [deleted] [R7] If weather forecast will not be available in the next seven days after the event will take place, application will notify the owner.

- [G11] After login, application will notify invited user one days before an event takes place if the weather is not good.

  - [R3] Application will notify ~~the owner~~ and invited users of the event only when they perform login.
  - [deleted] [R4] Notification will appear only once.

5.4 Modified Scenarios and Use Cases

Updating some goal and functional requirements has bring some changes also to the scenario. In particular the Scenario 2, 3, 4, 6 and 7. For the use cases the modification are listed in the follow:

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

5.5 Modified Diagrams

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