1.Introduction

1.1 Description of the problem

I will project and implement an online calendar for helping people scheduling events avoiding bad weather(MeteoCal).

The system will allow to sign up with standard informations (Name,Surname,Date of birth,age etc.), create events with their informations (where, indoor or outdoor,when etc.), update or delete them, invite other users(via searching).
Only the organizer have full controll of events who he created. No one else can delete,update or invite persons for others events.
It also enrichs events with meteo forecast and alerts users in case of bad weather in advance, one day before, for outdoor events.
All notifications will be show up on user's personal page: if one person accepted invitation or is invited, successful registratation,bad wheater,event deleted by creator.

1.2 Goals

the main features which MeteoCal has to provide are the following:

- Registration of a person to the system.
- Possibility to create events.
- Possibility to delete events.
- Possibility to udate events.
- Possibility to invite users.
- Ensure a meteo forecast for events if availble.
- Notifications.

1.3 Domain Properties

I suppose these conditions for modelling the reality:

- A person can invite any numbers of users.
- Events can't occur in the same time on the same date or partial overlap.
- User's Calendar may be empy.
- Events are private.
- Events are legal respect to the law.

1.4 Glossary

Defination of some words that i will use:

- **User**: a person registered in the system.They can use all functionality.
- **Event**: an istance created by a user.It could be shared with other users, indoor outdoor. Also it have a start and end time in one specifi date.It have a description and location.
- **Calendar Page**: the personal page of user where he can do anything within his powers.
- **Home page**: the principal page of application. Here a guest may sign up.
- **Guest:** a person who visits the home page of application. He can only sign up.
- **Calendar**: an user's page where he can controll his events.
- **Policy**: rules of the system built.

1.5 Assumptions

It is not all clear , so I will do some assumptions:

- A user could create an event but overlap someting else. In these cases, the system will abort the creation and notify him.
- There isn't a special user , system administrator. It's not necessary for this reality but there will be contacts to have assistance or give suggestions.
- No event could be indoor and outdoor.
- Update can add informations or change settings. In the second case, the system will check again policy: for example if there is a change from indoor and outdoor, system will add prevision(if available).
- If a event is deleted, a notification will be add to all user invited.

1.6 Proposed system

The chosen system will be a web application.
Users will able to access their page, add events through a calendar and modify as well.
Server will generate different pages for each user, according to the

informations put in by himself. It also provides a valide meteo support for events.

1.7 Identifying Stakeholders

The stakeholder is the X sofware house, direct by my professor.
The target is to create a sofware in all its phases: to prove that i understood the right approch in shaping reality, make a functional software, know how to use tools in a proper ways and overcome any kind of problem that maybe i will meet.

1.9 Other considerations

I think MeteoCal needs to have some properties that i will try to implement:

- **Simple**: user must find web application easy to use. Not twisted or difficult, without cryptic pages.
- **Stable:** every aspect of system must work(notifications, registration etc.).

2. Actors Identifying

The actors of my system are:

- **Guest**: a person not registred, can only sign up in home page.
- **User**:person that has registered and has provided his personal information.

3. Requirements

Thinking to all I wrote before i can derive requirements:

1) **Registration of a person to the system:** the system will provide a sign up simple and fuctional.
2) **Add an event into the calendar.**
3) **Modify a event:** change informations about an event.
4) **Invite other user to an event:** the system will provide a research fuction to invite users.
5) **Notifications.**
6) **Delete event**.

3.1 Functional Requirements

Here i will define what actors can do:

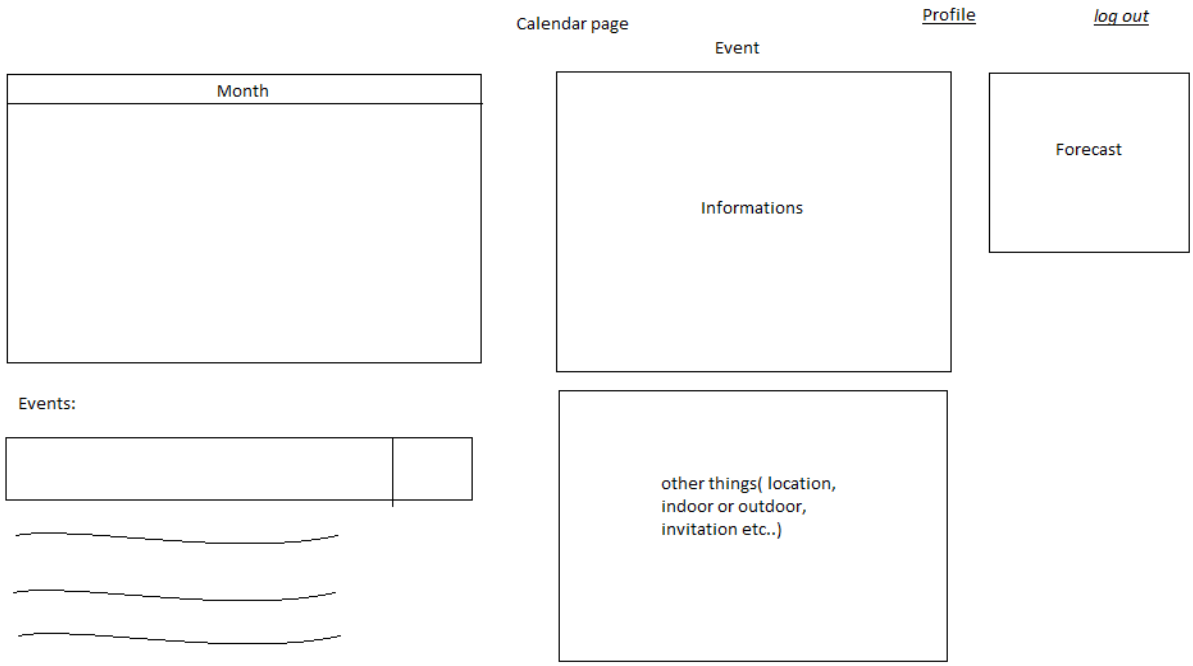
- Guest: he can only sign up.
- User:
    - Log in.
    - Modify his profile .
    - Create events.
    - Delete events.
    - Update events.
    - Invite other users.

3.2 Non Functional Requirements

3.2.1 User interface.

The interface of my web application will be easy to understand.The calendar page will simple and compact so that users have all things in his view range.

Here a very simple sketch of calendar page:



there is other importants pages which are profile and add event.
The first is just contains all data who user inserts at the registratation. Instead add event will have forms to compile and a serch box to invite users.

3.2.2 Proposed architecture

I will use JEE platform with a database in which all system's information will be stored.

Also an architecture client-server is the best way for this project, but it will discuss it in other documentations.
An Internet connection is needed to use MeteoCal and also a browser.

4. Specification

I will write here some specifications to reach the goals in mind:

- An user can't revoke is invitations.
- An user can accept or refuse a invitation, if the event passed or deleted the system will warn the user.
- Events are private.
- Invited users can see informations about an event but nothing more. No changes or its annullation.
- Invited users can't create an event in the same zone of calendar in which the shared event occurs.
- Notification are always in sight. User will decide when delete them.
- Users' profile are private.
- If an event passed, user can't do any changes.

5. Scenarios Identifying

here some cases of MeteoCal:

- Eleonora is a busy girl. She want to have a method to sort out her events but she is annoyed by the wheater.,which too often is against her.
  Hence, she decided to google in order to find a site who it will do it. And after a while, she found MeteCal.
  In home page there is a description of it. It seems easy. She maybe thinks that is what she was looking for and decide to sign up: name,surname, telephone number,hobby,adresss,photo.
  The form to compile is simple and the registratation is successful.
- Kavin an user of MeteoCal love the beach. He likes to surf and have some fun with friends,meteo is crucial for him, so through the calendar he want to set the next event. Into calendar page, with bottons, he can slide months. Here the date who wants, there is no other events on it. He click and a new page appears before him: time, event's name, general informations, location, outdoor/indoor, and search bar to invite other user.

When Kavin have finished , he commit and in the calendar it show up.He can click the day, and the list of events will see under the calendar. He can click an event

- Eleonora who has just registred, want to change her profile. In the calendar page there is a link to his profile. From there she can change her photo,which was bad, and add some informations.When she finished, click and all is upadated.
- Kevin log in. He found 1 notifications(there is an arrow close to "notifications") and decides to check. He click the word notfications and finds that tomorrow it will be a rainy day. Sadly he decides to delete the event: he goes in calendar page, and after select the date, events show up under the calendar and with the delete box, he erases it.
- Eleonora has just found that the chosen restaurant have a problem for that day, therefore she have to change the location. After some time, she found it.
  She goes in her calendar page, clicks on the day then on event below and clicks the buttor for the modifications. A new page appears with all the informations and she can easly change location.
  She commit and everything works.The calendar is upadated.

6. UML Models

6.1 Use Case Diagram

I can derive some use cases from what i wrote above:

- Search for an user and add it;
- Sign up;
- log in ;
- Add event;
- Delete event;
- Update event;
- Modify own profile;

6.2 Use Cases description

I describe here in details the use cases who i wrote:

Name	**Log in**
Actors	**User**
Entry conditions	The user has successfully signed up to the system.
Flows of events	• The user opens the home page. • The system shows him the page; User use his name and password in input form provided. • The user click the button “log in”. • The system show him the calendar page.
Exit conditions	There are no exit conditions.
Exceptions	The information inserted in the form is wrong, a message of error will appear.

Name	**Sign up**
Actors	**Guest**
Entry conditions	The user is on the web application.
Flows of events	• The user clicks sign up in home page. • The system shows up the form to compile. • The user compiles the input boxs. • The user click “ok button” to commite. • The system shows the modification page. • The user types in the search box the user who want to add and clicks ok. • The system will find him if he exists and adds to event. • The user will click the “ ok button” to confirm the update. • The system will update the database.
Exit conditions	Click on cancel button in modification page.
Exceptions	If user doesn't exist a message will appear in modification box.

Name	**Search for an user and add it.**
Actors	**User**
Entry conditions	The user has successfully logged in to the system and has to be on calendar page.
Flows of events	• The user clicks the day in calendar page. • The system shows up the events. • The user clicks the update button. • The system shows the modification page. • The user types in the search box the user who want to add and clicks ok. • The system will find him if he exists and adds to event. • The user will click the " ok button" to confirm the update. • The system will update the event.
Exit conditions	Click on cancel button in modification page.
Exceptions	If user doesn't exist a message will appear in modification box.

Name	**Add event**
Actors	**User**
Entry conditions	The user has successfully logged in to the system and has to be on calendar page.
Flows of events	• The user clicks the day who wants in calendar page. • The system shows up the events if they exist. • The user clicks the add button. • The system shows the modification page(empy). • The user adds informations and also invites other users. • The user commit the page. • The system will add the event to the database. It will add forecast if available. • The system will refresh the calendar page.
Exit conditions	Click on cancel button in modification page.
Exceptions	If user the created event overlap other events, the system will notify the user.

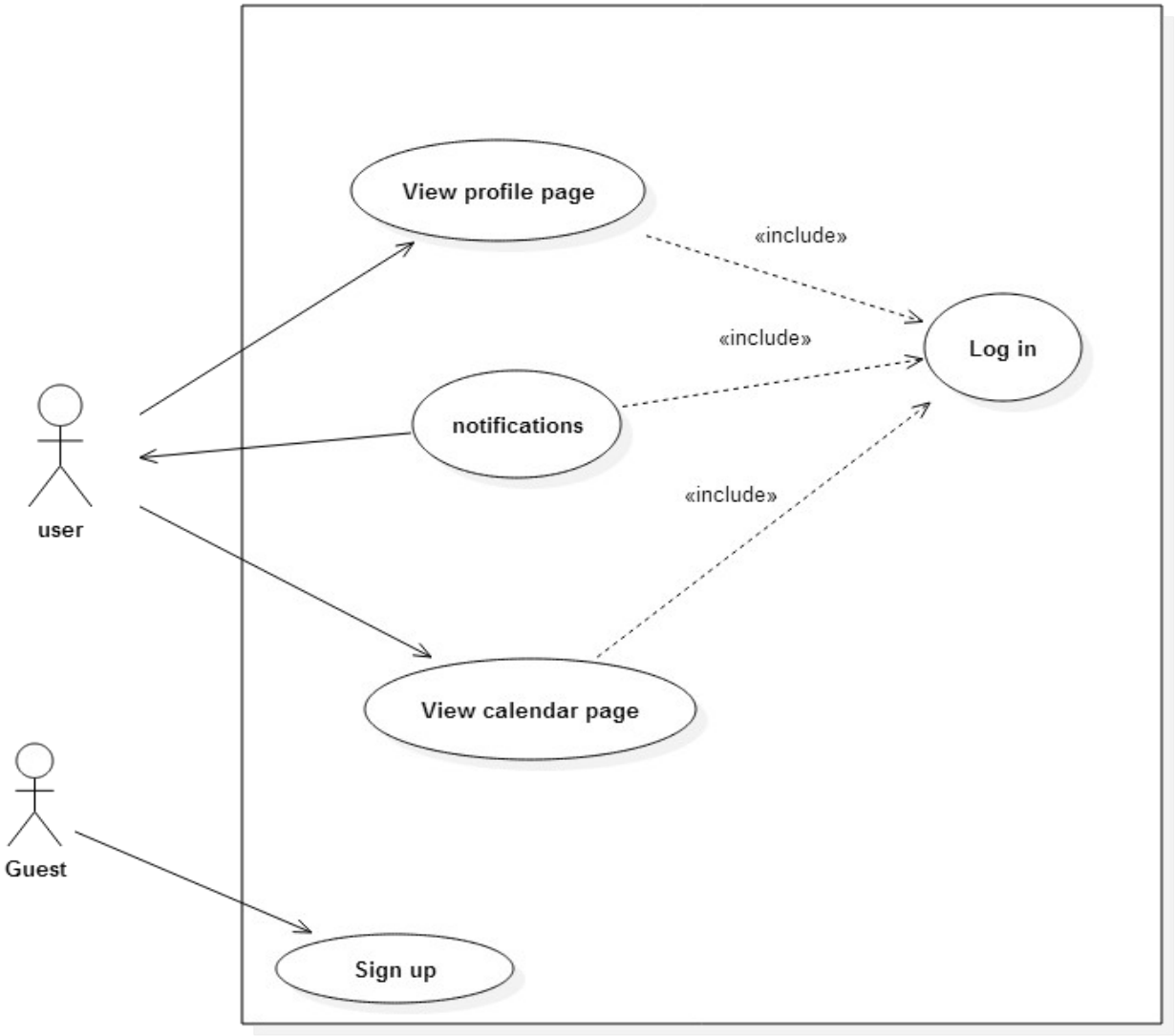
Name	**Delete event**
Actors	**User**
Entry conditions	The user has successfully logged in to the system and has to be on calendar page. Event must exist.
Flows of events	• The user clicks the day in where there is the event, in calendar page. • The system shows up the events. • The user clicks the delete button. • The system will delete the event. • The system will send a notifcation to all participants. • The system will refresh the calendar page.
Exit conditions	There are no exit conditions.
Exceptions	There are no exceptions.

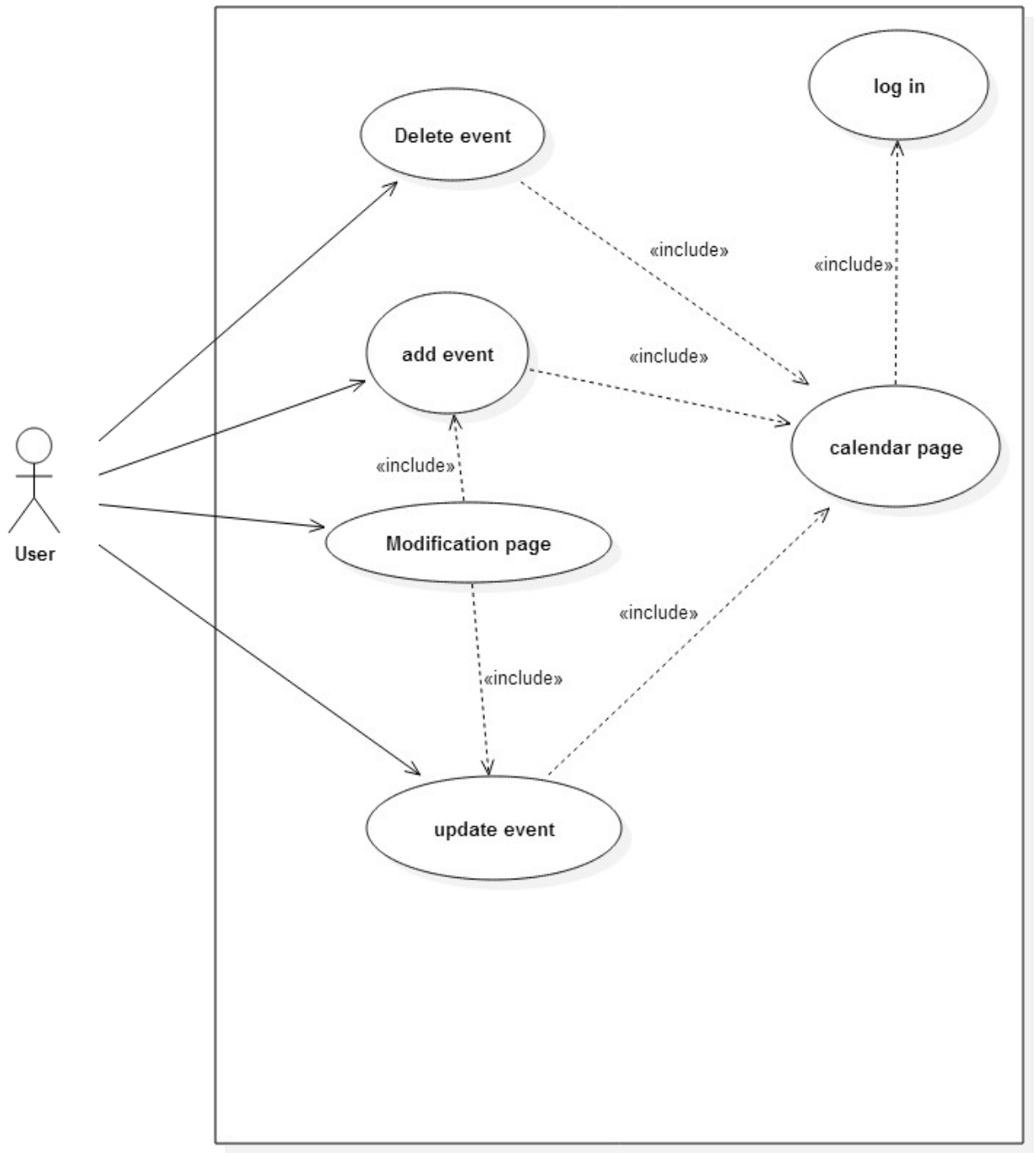
Name	**Update event**
Actors	**User**
Entry conditions	The user has successfully logged into the system and has to be on calendar page. Event must exist.
Flows of events	• The user clicks the day who wants in calendar page. • The system shows up the events . • The user clicks the update button on the right event. • The system shows the modification page. • The user changes informations and/or invites other users. • The user commit the page. The system will add the event to the database. It will add forecast if available. • The system will refresh the calendar page.
Exit conditions	User cancel the upadate with the correct button in modification page.
Exceptions	If user 's created event overlap other events, the system will interrupt and notify the user.

Name	**Modify own profile**
Actors	**User**
Entry conditions	The user has successfully logged in to the system and has to be on calendar page.
Flows of events	• The user clicks on profile link in calendar page. • The system shows up his profile. • The user change what he wants through the forms. • The user clicks the right button to send modifications. • The system update its data an profile page.
Exit conditions	User changes page.
Exceptions	There are no exceptions.

6.2.1 Use cases diagrams:

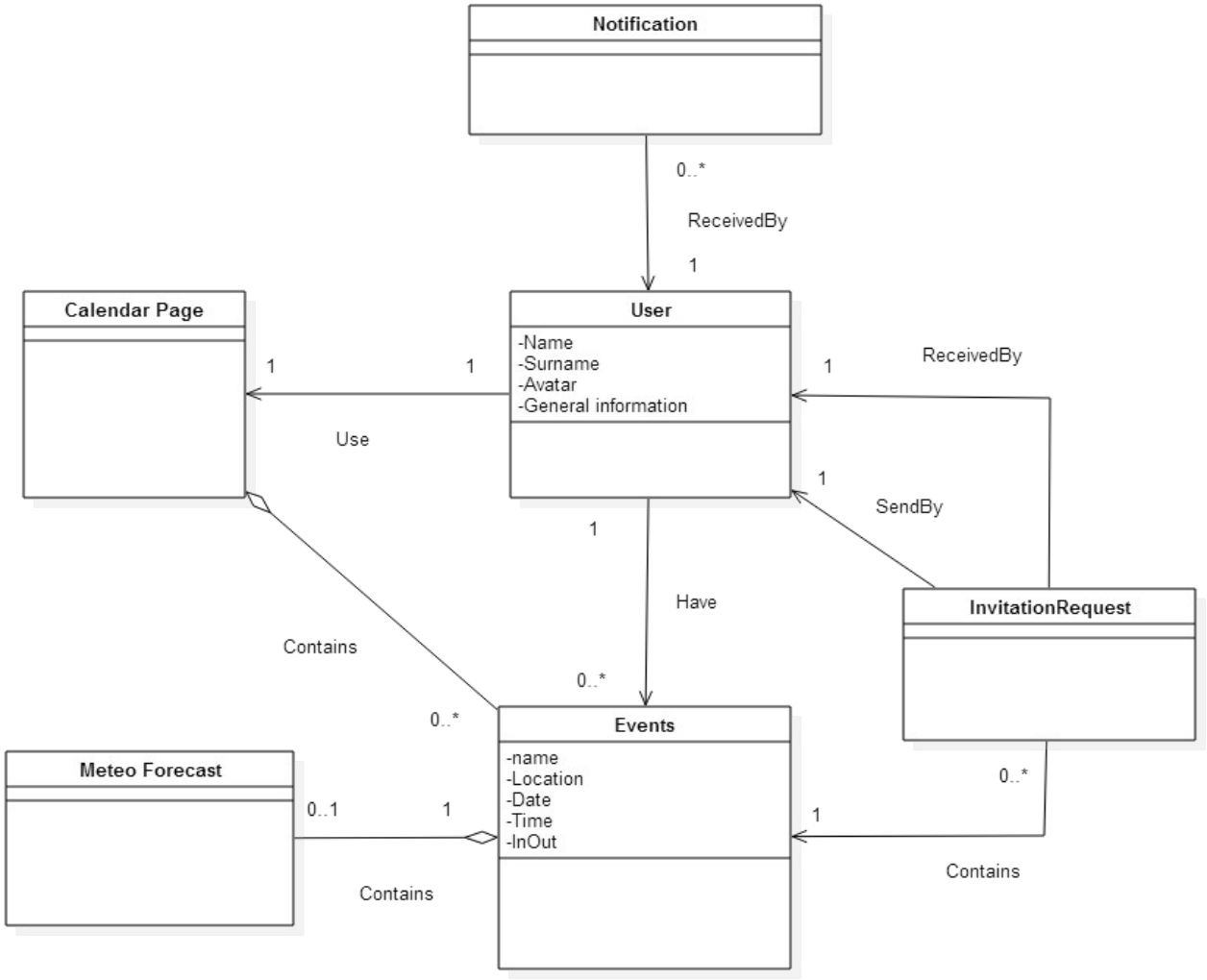
here there is use cases that i have splitted for a better comprension:



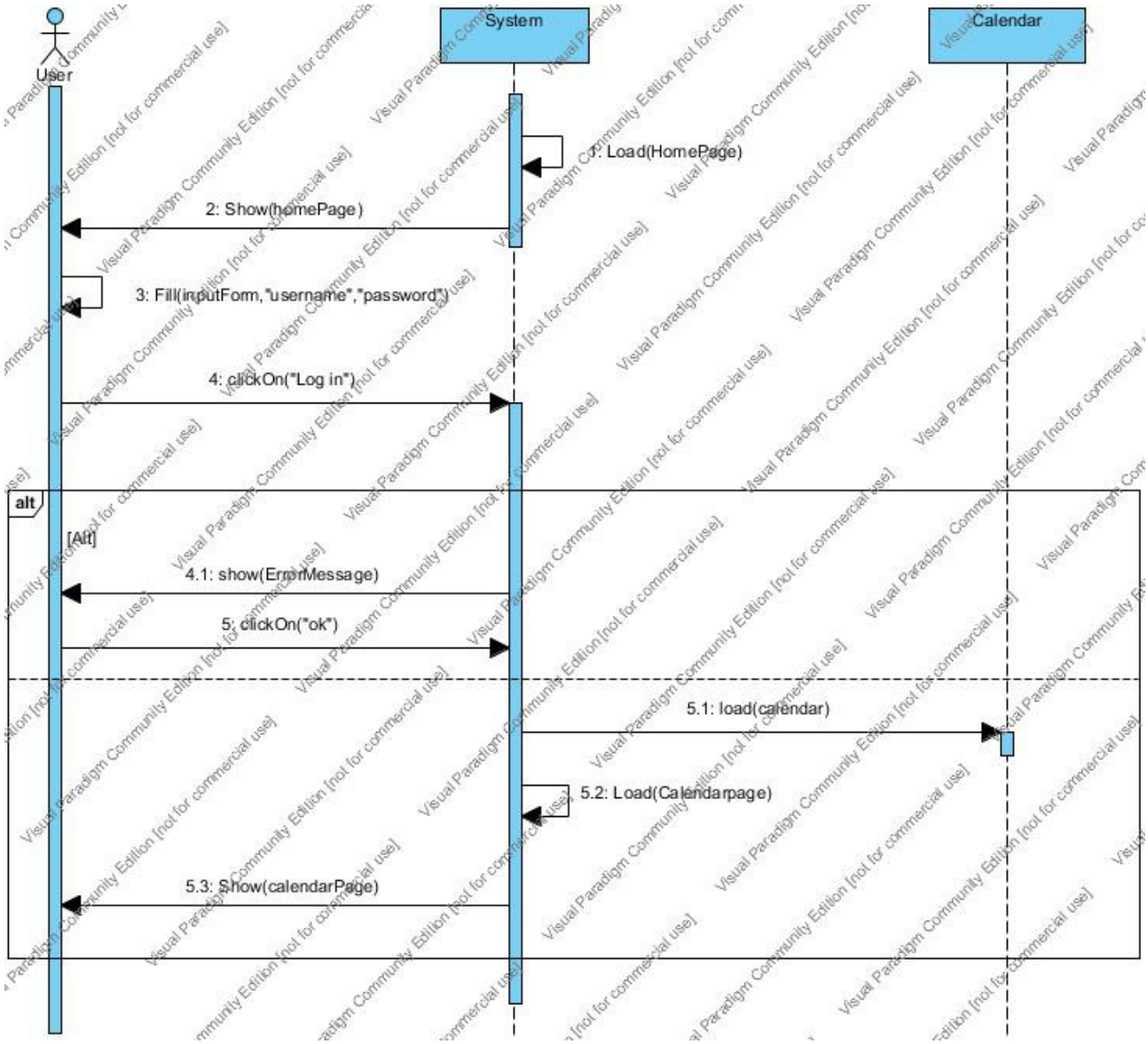

log in
Delete event
«include»
«include»
add event
«include»
calendar page
«include»
User
Modification page
«include»
«include»
update event

6.3 Class Diagram

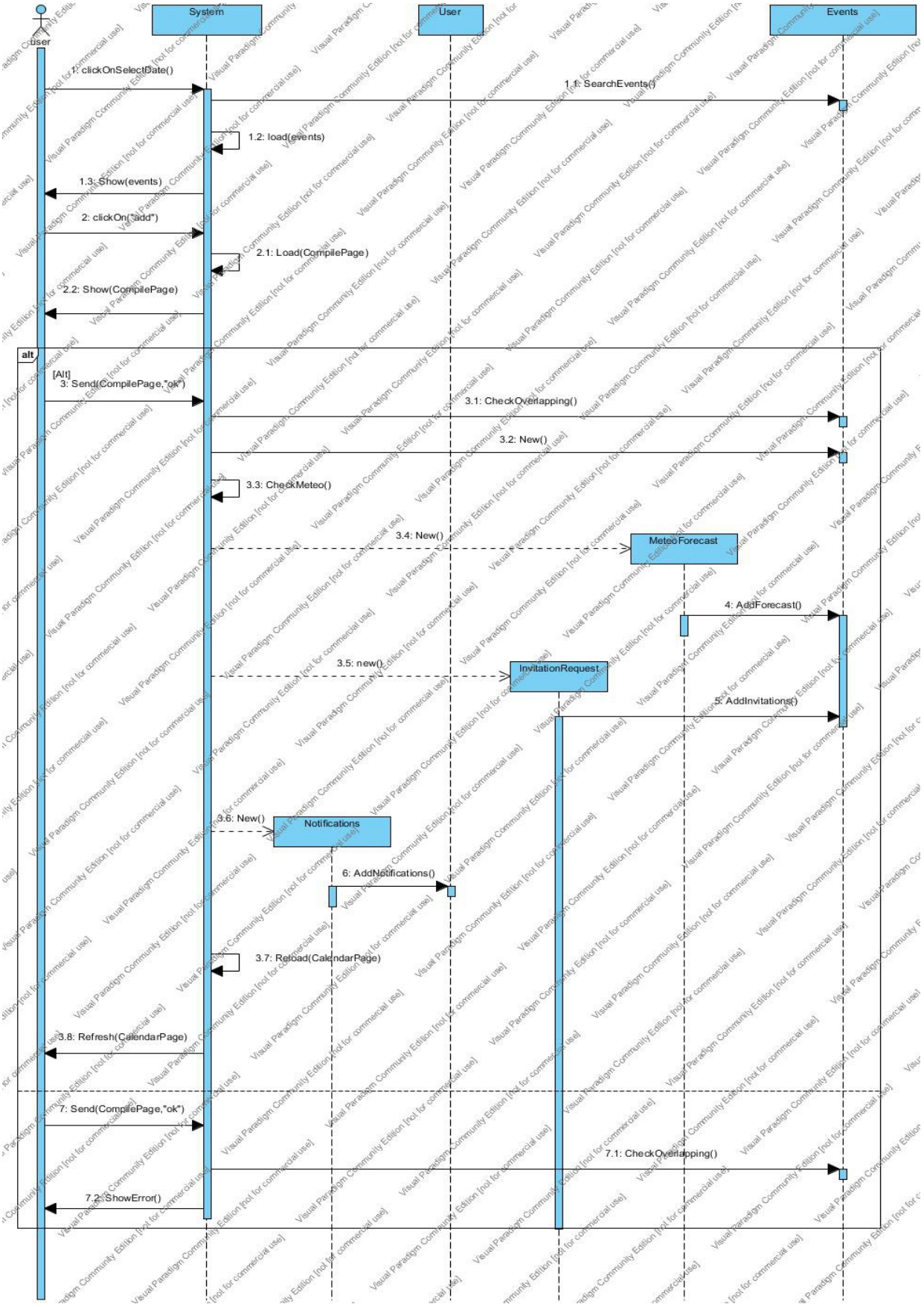
After the use cases I can drow a class diagram:



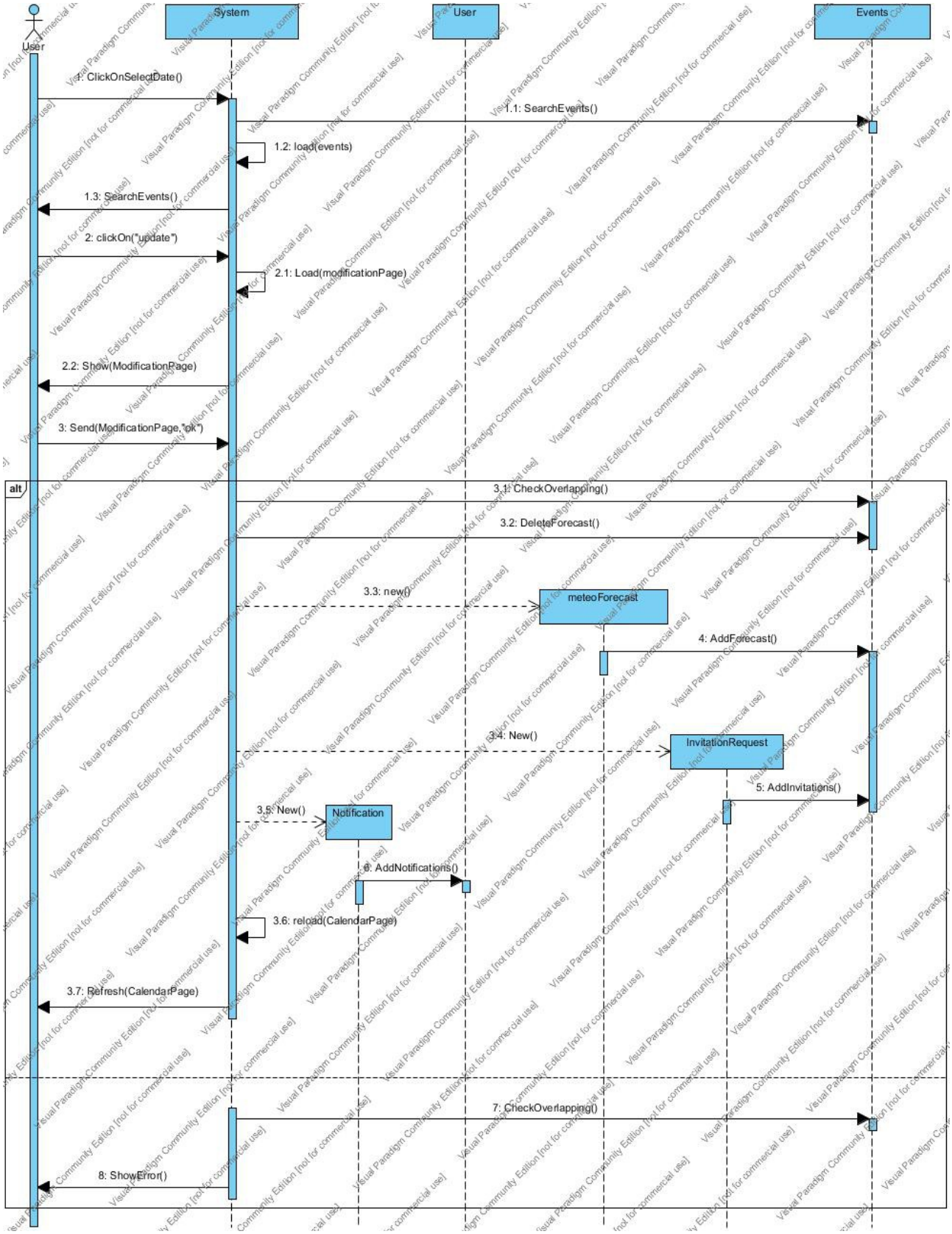
6.4 Sequence Diagrams



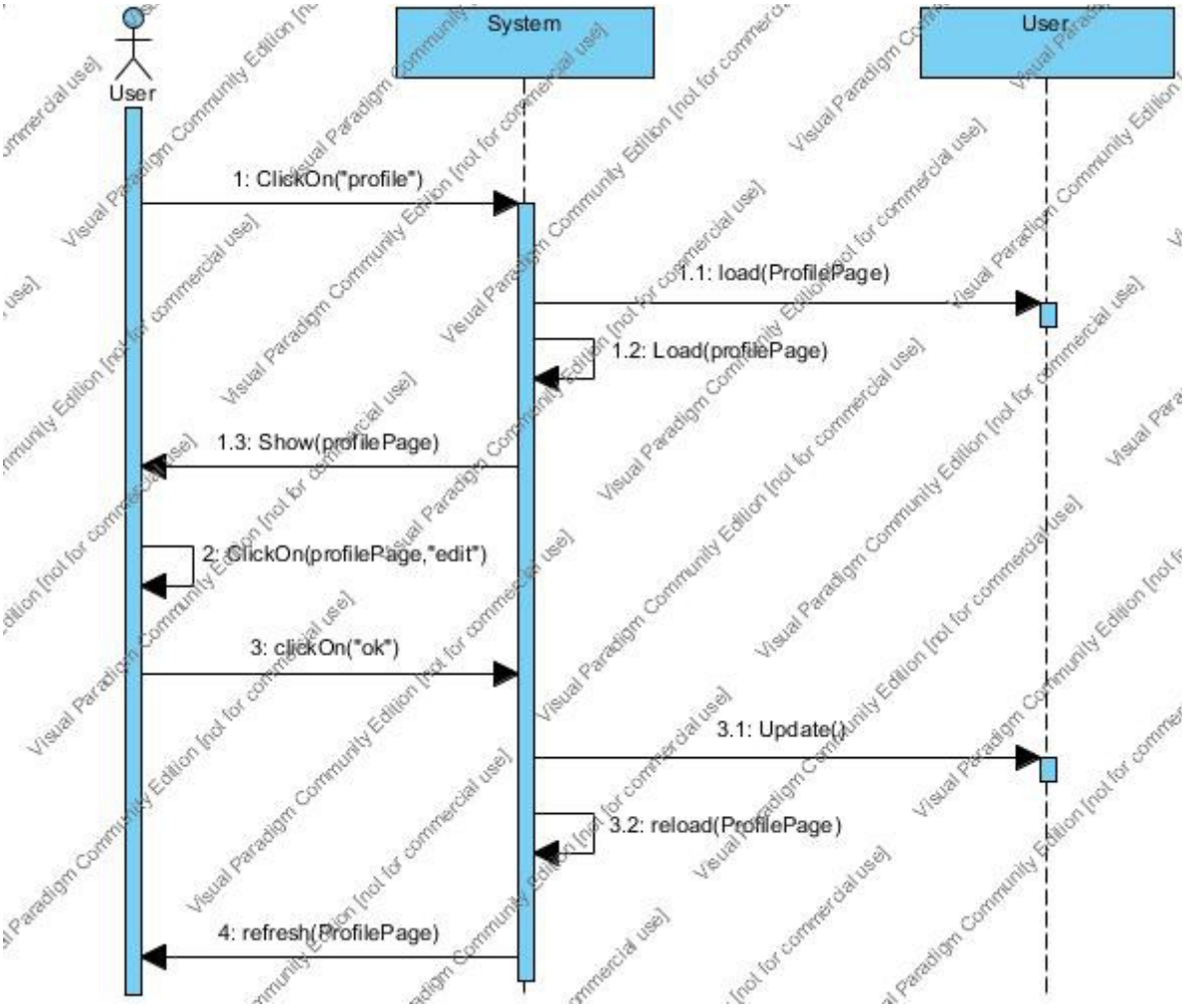
6.4.2 Add event



6.4.3 Modify event



6.4.4 Modify profile information



7.Alloy Modeling

```
module   MeteoCal
// SIGNATURES

sig System{}
sig Date{}
sig Time{}
sig Out{}

sig Forecast{
date: one Date,
time:  one Time
}

sig Event{
      owner: one User,
      mforecast:lone Forecast,
      date:one Date,
      time: one Time,
      invitedUsers: set User,
      invitedRequest: set InviteRequest,
      out: lone Out

}
sig User{
   events: set Event,
   notifications:set Notification
}

sig Notification{
      sender: one System,
      receiver: one User
}
sig InviteRequest{
      sender: one User,
      receiver:one User,
}
```

```
// FACTS

fact InvitationRequestProperty{
   // an user can't invite himself
      no  req: InviteRequest | req.sender=req.receiver
   // all  req1:InviteRequest ,  ev1:Event | req1.sender= ev1.owner implies  req1.receiver!=ev1.owner
   // no multiple invitations
      no  disj req1,req2: InviteRequest | req1.sender=req2.sender && req1.receiver=req2.receiver
   // an invitation is refered only to an event
      all rev: InviteRequest |  one ev:Event|  rev in ev.invitedRequest
   // for all  invitations the sender is the owner
      all   ev:Event, inv:  InviteRequest |  ev.owner=inv.sender implies   inv in ev.invitedRequest


}
```

```
fact UserProperty{
   // if  a user is the owner, than event must be into user's events
   all ev:Event, u:User|  ev.owner=u implies ev in u.events


}
fact NotificationProperty{

      //  notification from system  must be assign to an user

      all n: Notification, u:User | n.receiver=u  implies n in u.notifications

}




fact EventsProperty{
   // an event can't overlap  another

      all disj ev1, ev2: Event |  ev1.owner=ev2.owner  implies ev1.time!=ev2.time || ev1.date!=ev2.date

   // an event must have a  right  forecast

      all ev:Event| lone met: Forecast |   (#ev.out=1 && #ev.mforecast=1) implies  met.date=ev.date && met.time=ev.time

   //  the owener is not  an invited user

      all ev:Event | ev.owner not in ev.invitedUsers


   // if a user receive an invitation, then he can't be a partecipiant

   all  even: Event,inv:InviteRequest| inv.receiver not in even.invitedUsers


}
```

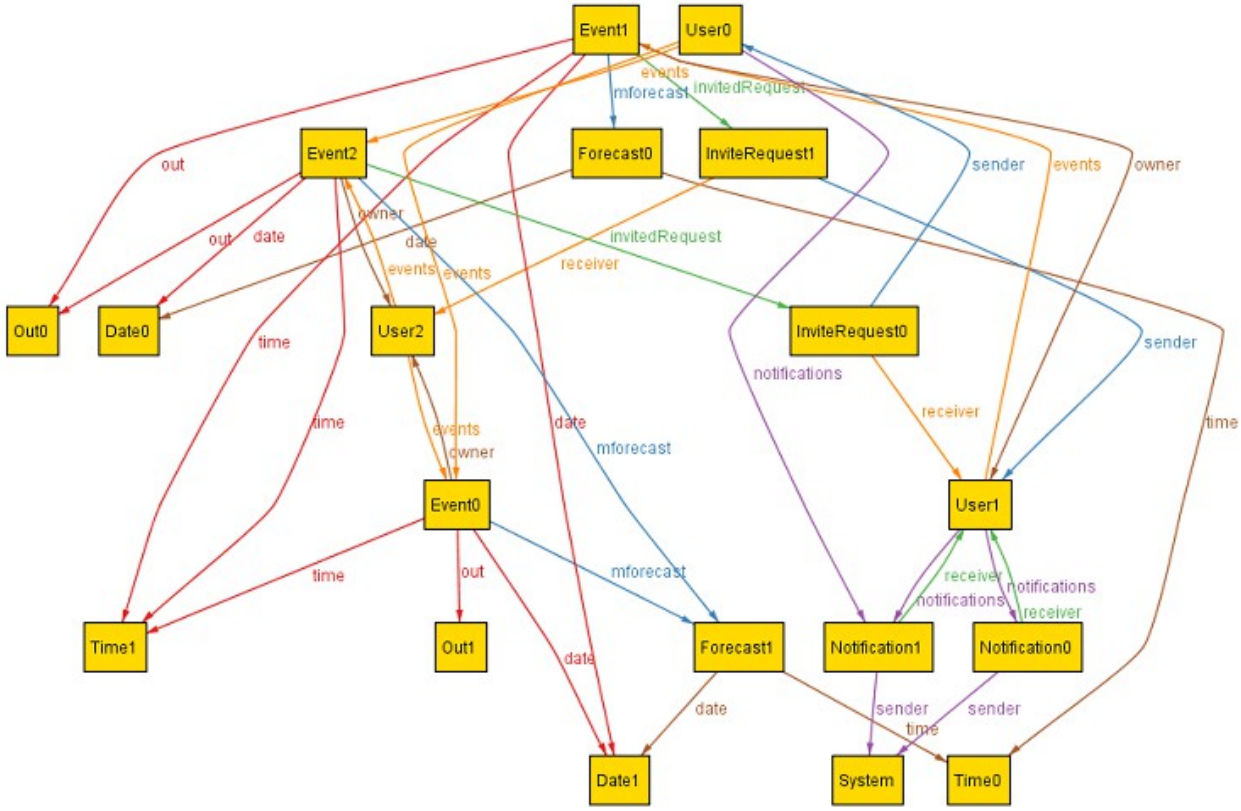
```
// ASSERTATIONS

assert SelfInvitation{
      // an user can invite himself

         no  u: User| some inv:InviteRequest | inv.sender= u && inv.receiver=u

}
check SelfInvitation


assert NosameEvent{

// events can't be equal


   all disj ev1, ev2: Event |  ev1.owner=ev2.owner  implies ev1.time!=ev2.time || ev1.date!=ev2.date

}

check NosameEvent

//PREDICATES

pred  showEvents(){}

run showEvents for 3 but 2 Event, 0 Forecast,  0 InviteRequest,2 User, 0 Notification

pred showUser(){}

run showUser for 3 but  2 Event, 3 Notification, 3 User, 0 Forecast,0 InviteRequest

pred show{}
```

And here the executions:

5 commands were executed. The results are:
#1: No counterexample found. SelfInvitation may be valid.
#2: No counterexample found. NosameEvent may be valid.
#3: **Instance found.** showEvents is consistent.
#4: **Instance found.** showUser is consistent.
#5: **Instance found.** show is consistent.

8. Worlds Generated

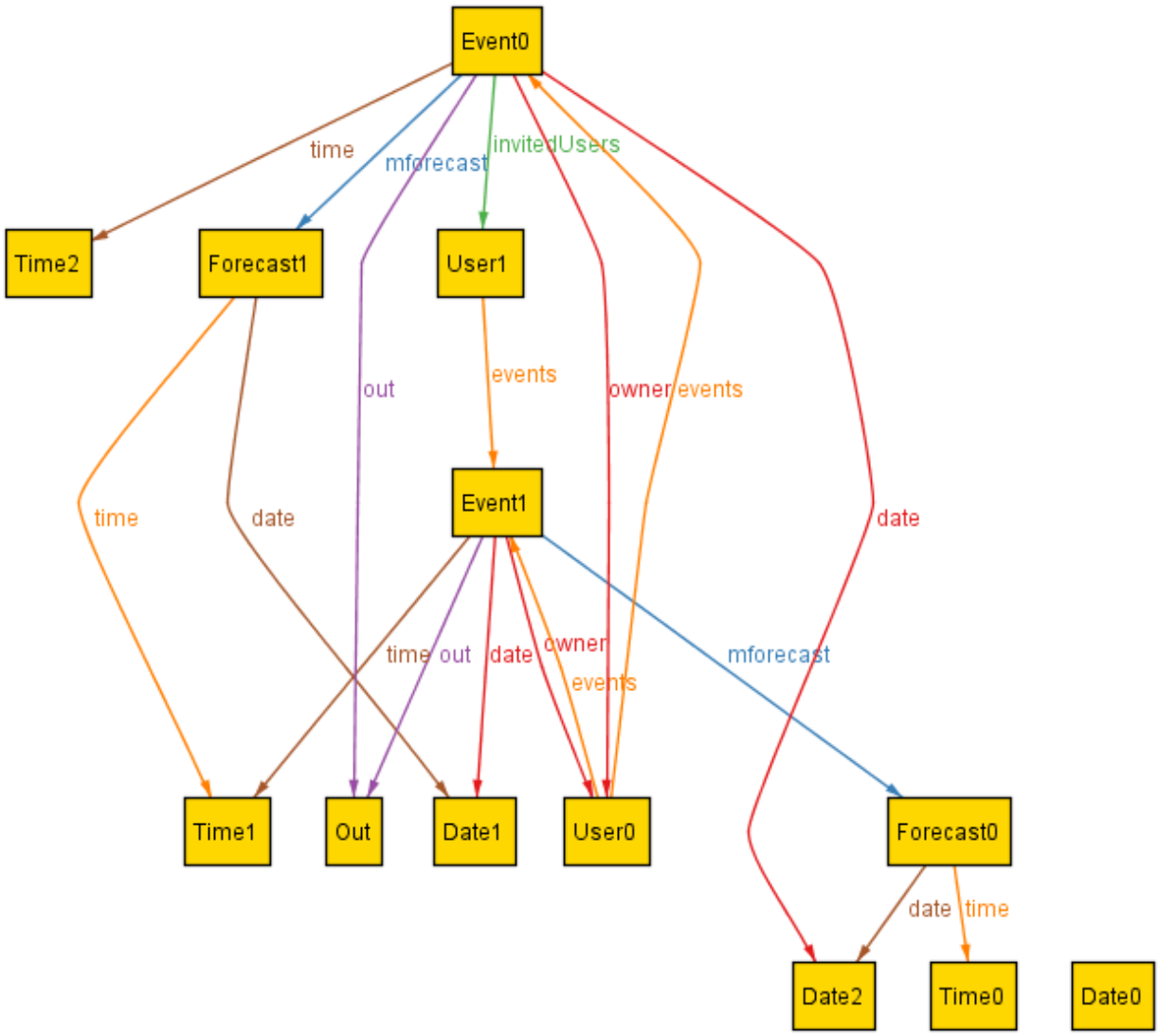
Here a possible world genereted by alloy:



8.1 A case semplified

Here a case without:

-Invitation requests.
-Notifications.



9. Used Tools

The tools I used to create this RASD document are:

- Open Office Writer: to redact and to format this document.
- Paint : to create sketch.
- StartUML: to create Use Cases Diagrams and Class Diagrams;
- Visual Paradigm 11.2 Community Edition: to create Sequence Diagrams .
- Alloy Analyzer 4.2: to prove the consistency of our model.