

Politecnico di Milano

Facoltà di Ingegneria dell’Informazione

**Progetto di Ingegneria del Software 2**

Parte I: RASD

Prof.ssa Di Nitto

A.A. 2017/18

|  |  |
| --- | --- |
| Autori | Matr. |
| Amadelli Federico |  |
| Artoni Alessandro |  |
| Baccelli Alessio |  |

**R**equirements

**A**nalysis

**S**pecification

**D**ocument

Progetto *Travlendar+*

[**1. Introduction**](#_tvr04desrmvg) **4**

[1.1. Purpose](#_5qnaj7mmisar) 4

[1.2. Scope](#_7poiav197w0y) 4

[1.3. Definitions, Acronyms, Abbreviations](#_wmbswn1ub90h) 5

[1.3.1. Definition](#_3r12h4g0ph7w) 5

[1.3.2. Acronyms](#_sdqk06a99xdn) 5

[1.3.3. Abbreviations](#_ervcjv8loupx) 6

[1.4. Revision history](#_cruvacde83n0) 6

[1.5. Reference Documents](#_2piarg89gwa) 6

[1.6. Document Structure](#_tc3s87judass) 6

[**2. Overall Description**](#_1tn6gshlod8y) **6**

[2.1. Product perspective](#_e1spjdlupi32) 6

[2.2. Product functions](#_fddftzdsqcc7) 6

[2.3. User Characteristics](#_gu3a7i63ftgv) 6

[2.4. Assumptions, dependencies and constraints](#_fiqpc4zgecoo) 7

[**3. Specific Requirements**](#_h0ah6uk65nt5) **7**

[3.1. External Interface Requirements](#_h05cvpuvbcb9) 7

[3.1.1. User Interfaces](#_tif8t0u8v24j) 7

[3.1.2. Hardware Interfaces](#_hewjmbewp83j) 7

[3.1.3. Software Interfaces](#_vo16ias7h15e) 7

[3.1.4. Communication Interfaces](#_17banybb6bs1) 7

[3.2. Functional Requirements](#_i0278imvdf9i) 7

[3.3. Performance Requirements](#_hljo9zy3ppr8) 7

[3.4. Design Constraints](#_6oqsxennspc7) 7

[3.4.1. Standards compliance](#_vbtakg8zlqy7) 7

[3.4.2. Hardware limitations](#_r59nq97nly1w) 7

[3.4.3. Any other constraint](#_ahs376b9eypy) 7

[3.5. Software System Attributes](#_mk681p26x1qe) 7

[3.5.1. Reliability](#_an4efl2tx81y) 8

[3.5.2. Availability](#_4qsbrbrh1cl7) 8

[3.5.3. Security](#_ouedava0cl36) 8

[3.5.4. Maintainability](#_95owgo49hd95) 8

[3.5.5. Portability](#_jwfw9fata8cq) 8

[3.5.6. Usability](#_sq7c844jkdop) 8

[**4. Formal Analysis using alloy**](#_x5ti2fho44gr) **8**

[**5. Effort Spent**](#_kbudkhw2qrw) **8**

[**6. References**](#_cvwdg9fbingp) **8**

Chapter 1

# **1. Introduction**

Travelling has never been so easy. Not only traditional vehicles improved, but new phenomena like car and bike sharing, or Uber appeared. Mobility itself is changing.

People started using new services to move, and when they need to visit a new place often rely completely on Google Maps. The same users need a better manage of their schedules.

Therefore this project wants to help users to schedule their meetings accounting the travel time between each appointment giving the possibility to customize each appointment and each travel path.

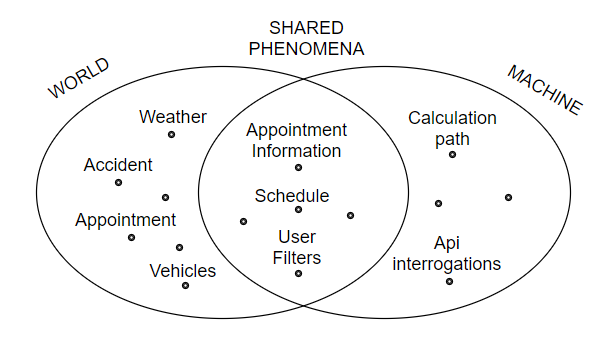
## ***1.1. Purpose***

This project wants to:

* Allow users to build a calendar where they can add appointments;
* Show the distance from one appointment to the next;
* Show all the possible travel solutions to arrive at the next appointment, giving a set of filter (such es lowest ecological footprint, cheaper solution, faster solution..);
* Give the possibility to schedule particular appointment, like lunch, or private ones;
* Allow clients to filter the travel solution to select the one they prefer;
* Support users in their travel, allowing them to buy tickets directly from the platform.

## ***1.2. Scope***

The system, in order to achieve all the aforementioned goals has to interact both with users and other external services (like weather forecasting, ticket prices, etc.) which will help the application in giving a complete and reliable service.

This is an example of how the machine interacts with the environment.

Users insert their appointments and the machine calculates the best path, according to users’ preferences, making interrogations to an appropriate API, taking into account the possibility of accidents and the weather.

## ***1.3. Definitions, Acronyms, Abbreviations***

### ***1.3.1. Definition***

* *Vehicles****.*** All the means of transport (bike, bus, car...) that the user can use to reach the point of interest.
* *Appointment***.** An arrangement to meet someone at a particular time and place, it comprends working appointments, dates, all the events planned by the user.
* *Schedules.* A plan for carrying out a lists of intended events and times.
* *Travel**path.* Is the roadway chosen by the system based on the point the user has to go.
* *Accident.* All the event that can happen to delay the user during the travel path.
* *User**filters.* The user’s preferences on the travel path, like elements to avoid, or elements to take.
* *Id*: identification of the user to access the application
* *Password*: a secret string used by the user to verify his own account
* *Urgency level:* for each appointment is possible to set up a different level, called urgency level, it gives the appointment a certain level of importance, higher is the lever, more important is the appointment, lower is the delay margin.
* *Actor*: an entity that interacts with the system

### ***1.3.2. Acronyms***

* API: Application Programming Interface.
* DBMS: Database Management System.
* UI: User Interface.
* RASD: Requirement Analysis and Specification Document.
* ETA: Estimated Time of Arrival.
* MTBF: Mean time between failures.
* MTTR: Mean time to repair.
* UL: urgency level

### ***1.3.3. Abbreviations***

* Req. as for Requirement.
* WebApp as for Web Application.

## ***1.4. Revision history***

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Authors* | *Summary* |
| 1.0 | / | Amadelli & Artoni & Baccelli | Release |

## ***1.5. Reference Documents***

## ***1.6. Document Structure***

# **2. Overall Description**

## ***2.1. Product perspective***

The product is supposed to be an open source, under the GNU general Public License.

It is a web based system implementing client-server model.

*Travlendar+* provides simple mechanism for users to schedule appointments and organize travels betweens the formers.

It will be a stand-alone product relying on different external information.

## ***2.2. Product functions***

*Travlendar+* is created to solve all schedule problem a person can meet.

It helps creating a calendar, managing meetings and appointments either these are job-meeting or something else like lunch or normal activities.

In order to do that, it will help users selecting their best travel option, either it will be the lowest footprint, or the fastest option. More details on the functionality of the product will be discussed in section [3.2].

## ***2.3. User Characteristics***

It is considered that the user do have the basic knowledge of operating on calendars and to have access to internet.

Our target are all of these people that require a basic management of their schedules and can’t afford to hire a secretary. Nevertheless, the system can help secretaries to manage their boss meetings.

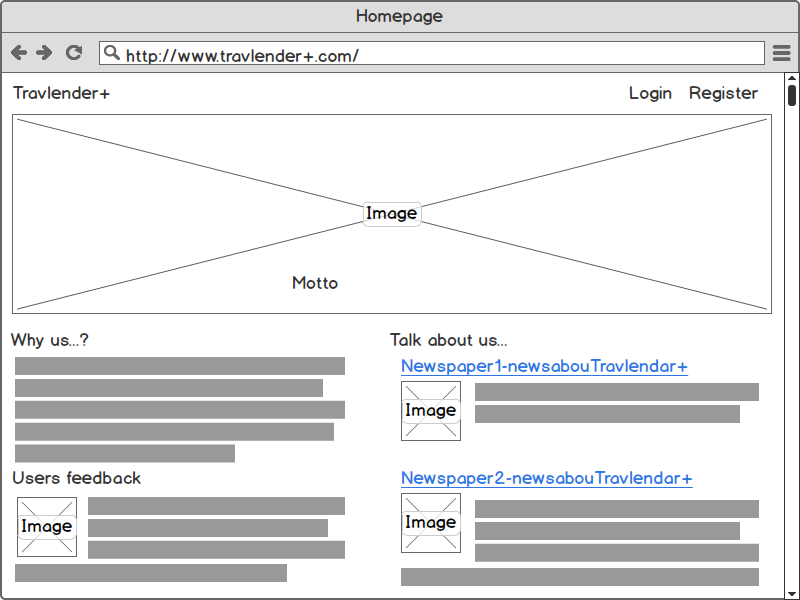
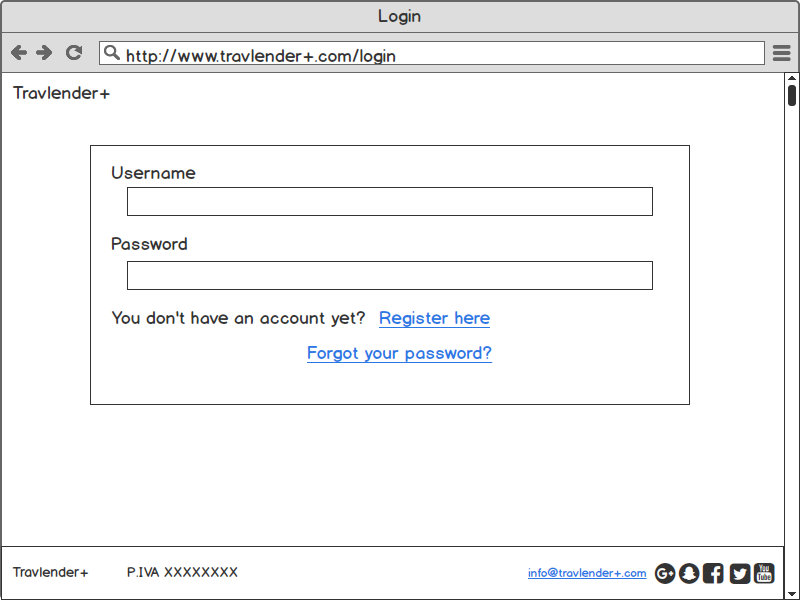
## ***2.4. Assumptions, dependencies and constraints***

* The possibility to schedule appointments and see the existing agenda is under the responsibility of the service and it is strictly correlated with his running. However, the system will use external API’s, in order to compute the shortest path for example. So the continuation of the service cannot be ensured if errors or problems occur to these external platforms.
* The punctuality of all the public transportation cannot be guaranteed as some unpredictable accidents may happen. The time table used by the system will be the official one, provided by the managing institution of the public transportation.
* All the information about traffic and weather are in real time and represents, as much as possible, the reality.
* If the user decides to buy tickets through the platform he cannot ask for a refund in any time later
* We assume for the moment that a certain user will use the Application only in Milan.
* Since the user is in Milan, when he will use public transportation only one ticket will be sufficient to travel all around the city.
* When using multiple types of public transport (example bus and metro) the application will show a generical icon for public transport. Further details will be written next to it.

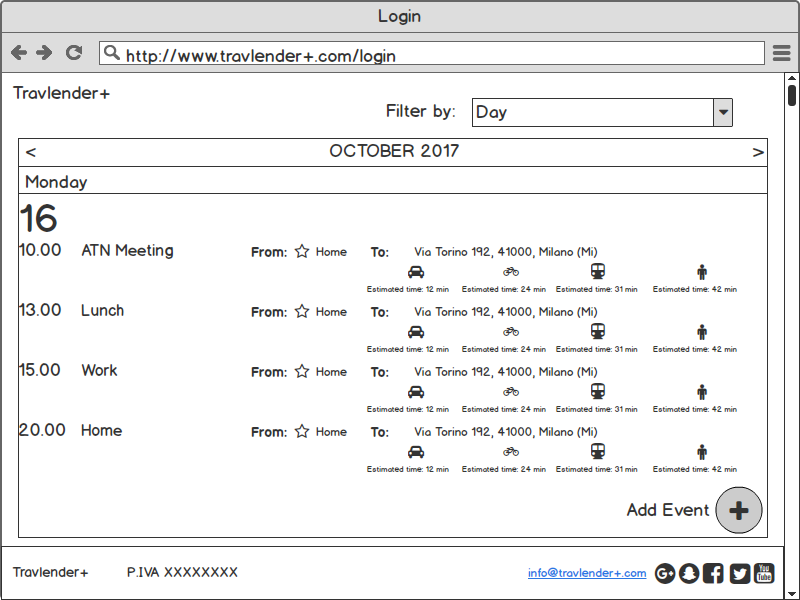
# **3. Specific Requirements**

## ***3.1. External Interface Requirements***

### ***3.1.1. User Interfaces***

The user interface will be developed

### 



### Here vision from a mobile phone:

TODO: INSERT IMAGES FROM WORD.

### ***3.1.2. Hardware Interfaces***

There won’t be any Hardware interfaces. Users will be able to connect to Travlender+ using the application or the website.

### ***3.1.3. Software Interfaces***

### ***3.1.4. Communication Interfaces***

## ***3.2. Functional Requirements***

## ***3.2.1. Major Function of the Login and Registration System***

1. Authenticate and login user to the webapp;
2. enable new users to register to the system, inserting username, password and email;
3. enable a registered user to change his password and his username;
4. enable a registered user to request his password if he forgets the former.

## ***3.2.2. Major functions of the Calendar***

1. enable the user to insert new appointments;
2. inform the user if the programmed appointment is too far to arrive in time;
3. enable the user to set up different UL for the appointment;

## ***3.2.3. Major functions of the Travelling System***

1. enable a register user to see different paths to go to the next destination;
2. enable a register user to set up favorite transportation means or paths to avoid;
3. recommend the user a path, based on the ETA, weather, cost;
4. inform the user if during the day there are strikes or problems of the transportation means.

## ***3.2.4. Major functions of the Ticketing System***

1. enable a registered user to buy a ticket online;
2. enable a registered user to insert and save his payment card for future uses.

## ***3.2.5. Major functions of the Lunch System***

1. enable a registered user to book a restaurant for a certain number of people;
2. enable a registered user to see the menu of the restaurant;
3. enable a registered user to buy coupon for the chosen restaurant, using a payment card.

## ***3.3. Performance Requirements***

The system needs to be extremely rapid (maximum 10s of milliseconds).

Users not only want to create an event, but usually get live information about the traffic or when they will arrive in a certain location. Users also may rely on the application when they’re in difficulty and they don’t want to spend time looking themselves at informations.

## ***3.4. Design Constraints***

### ***3.4.1. Standards compliance***

### ***3.4.2. Hardware limitations***

### ***3.4.3. Any other constraint***

## ***3.5. Software System Attributes***

## ***3.5.1. Reliability***

The system *Travlendar+* has to guarantee the persistency of data inserted by the user, like personal data or appointments.

*Travlendar+* will guarantee the maximum precise travel time, considering the most number of variables possible. It has to contains the lowest amount of bugs, the MTBF has to be extremely low, and the MTTR has to be relatively fast, of course the graver the error faster must be the correction.

### ***3.5.2. Availability***

*Travlendar+* has to be available 7 days per week, 24 hours per day considering the constraints mentioned before.

Since the system has to help users during all their travels, at every hour of the day,

it will be completely automatic.

### ***3.5.3. Security***

The system has to be safe, allowing users to enter a personal space, usingtheir own *id* and *password,* so it has to avoid the possibility of stealing the profile of others users.

Furthermore *Travlendar+* offers the service to buy tickets online, using bancomat or other cards prepared for the online payment, therefore those personal data will be protected appropriately.

### ***3.5.4. Maintainability***

With new technologies invented everyday, it's really hard to estimate an application lifecycle time because it will depend on how market will react.

*Travlendar+* has to guarantee an updated system, so should be relatively easy to add new function and allowing to use the new means of transport.

### ***3.5.5. Portability***

Travlendar+ has to be supported by all the devices (smartphones, computers, tablets), focussing mostly on smartphone and mobile devices, because they’re more comfortable and easier to use while travelling.

### ***3.5.6. Usability***

*Travlendar+* should be extremely easy to use, the user has only to know the destination and the system will work on the travel.

For more complicated functions, like deciding the favorite paths or the ones to avoid, the user has to try a few times before the app will become easier to understand.

# **4. Formal Analysis using alloy**

## ***4.1. Actors***

We can define 2 different actors that interacts with the system:the user and the external information.

The 2 different actors have different role in our system, following we give a brief description:

* *User:* he is the person to which the system is addressed,he has an account and uses Travlendar+, he addes appointments, buies tickets…
* *External informations:* it has a central role in our system during the path creation, it shares APIs with the system, allowing the system to give to the user the best path for his needs.

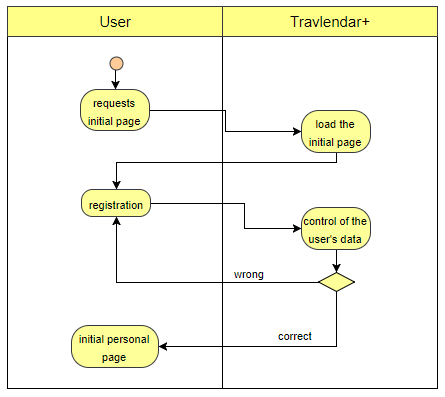
These are only some of the characteristics of the interaction between the system and the actors, following we describes some of the scenario of the system, giving a deeper analysis of the potentiality of our system.

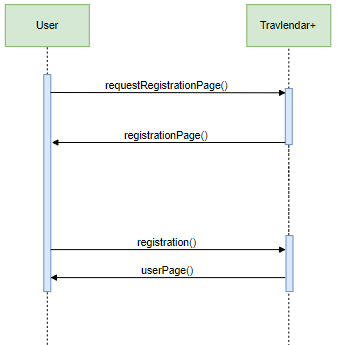
## ***4.2. Scenarios Identification***

In this section, some possible scenarios are identified.

### ***4.2.1 Registration***

Luke is a veterinarian. He just started his job. For the moment, he’s doing a home-service visiting patients directly at their home. Luck hears about Travlender+ and decides to download the application. He registers, and using his credentials he logs in the system controls if the credentials are already used on another account, not finding errors the registration is done successfully.





### ***4.2.2 Scenario 2***

Mark works in a big city, and has 3 appointments this day, one of them in the city centre. Since the levels of pollution in towns are pretty high, the town city center has been declared off-limits in some days, but he doesn’t know which one.Travlendar+ suggests Mark the optimal means of transport knowing the situation of the city.

### ***4.2.2 Insertion of an appointment***

Mark is the head of a big industry, and today a company had called to create an appoinment with him

### ***4.2.3 Scenario 3***

Jennifer this morning has a really important appointment. She scheduled that months ago, and she absolutely can’t be late. After she got dressed, her father is going with her.

Sadly during the way there’s a car accident, and the road won’t be access for some hours.

Since Jennifer’s father doesn’t usually drive, he only knows that road!

At this point, Jennifer uses her Travlender+ application. The application understands there’s an accident and manages to find another way to arrive at her appointment.

Even if Jennifer is usually a latecomer, she will arrive in time at her wedding!

### ***4.2.4 Scenario 4***

Met is coming back in Milan after a long time. He has just arrived at the central station but realizes it’s Sunday and that the ticket offices are closed. So he goes to the automatic ticketing service and since “bad luck never travels alone”, none of them is working.

Met then decides to rely on his application, and buy a Metro ticket online.

### ***4.2.5 Scenario 5***

Alice he’s a busy person. She has always something to do and somewhere to go except for the weekend. On sunday she likes to run around the city for a couple of hours. Since she’s already using Travlender+ to manage her appointments, she can also uses the application to know how many kilometers she ran.

## ***4.3. User use cases***

Here there will be a list of relevant use cases for the users. The actors in this use cases will always be the user.

### ***4.3.1 User registration***

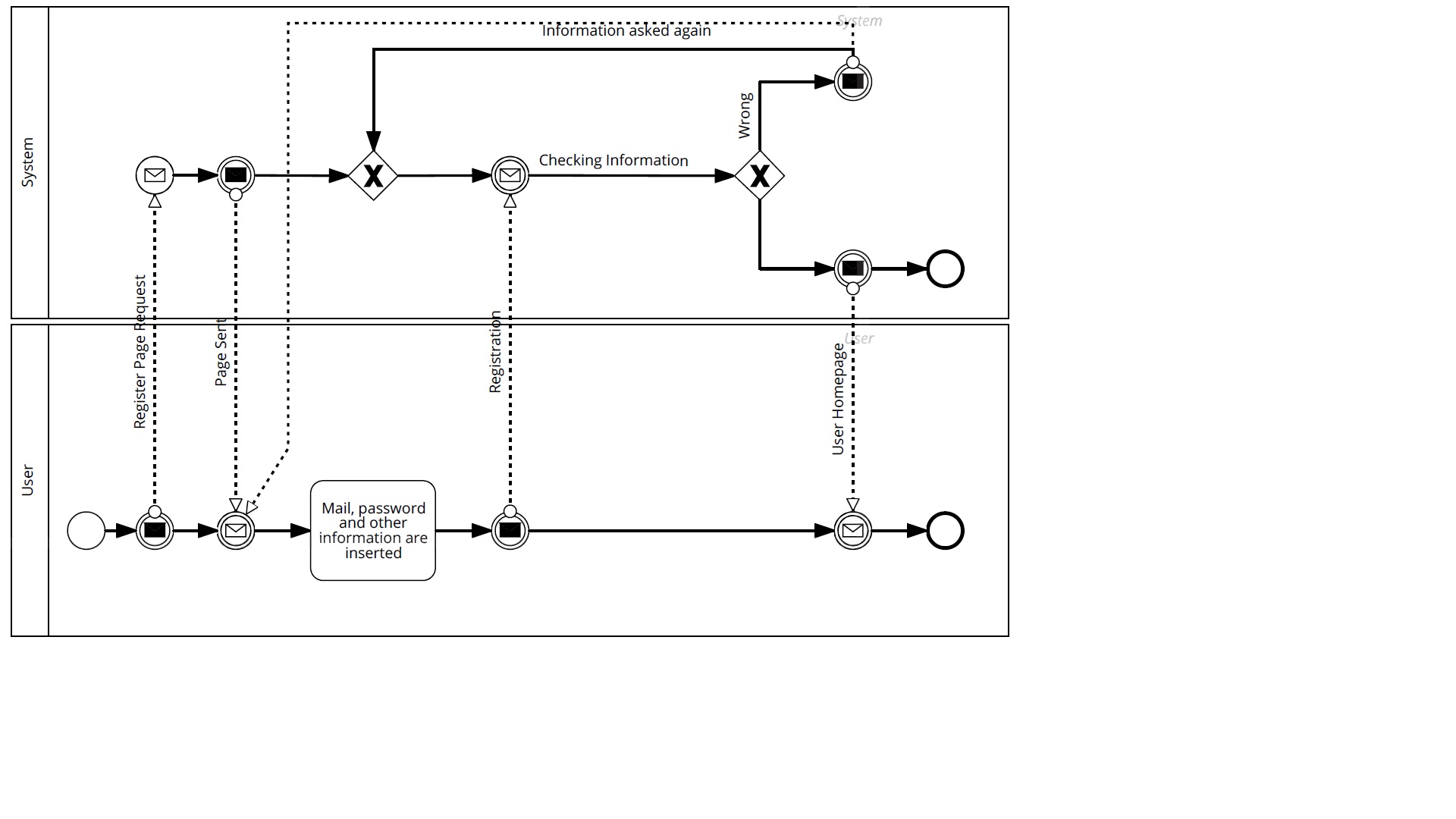
**Precondition:** The user has already downloaded the application, or she/he is connected to the web portal.

**Flow of events:**

* The user provides a mail and password.
* The user gives information about himself (Name, Surname, date of birth, home address, sex)
* The user clicks the button to sign up, and the registration is submitted to the system.
* The system verifies if the email is valid or if it is already used.
* The system verifies if the password is strong enough
* The Systems tells the user that the registration was successful and sends a “welcome e-mail” where it confirms the registration.

**Exit condition:** the user is registered and logged into the system.

**Exception:** if the user submitted data that are invalid, or the username is already used the system will ask them again.



### ***4.3.2 User login***

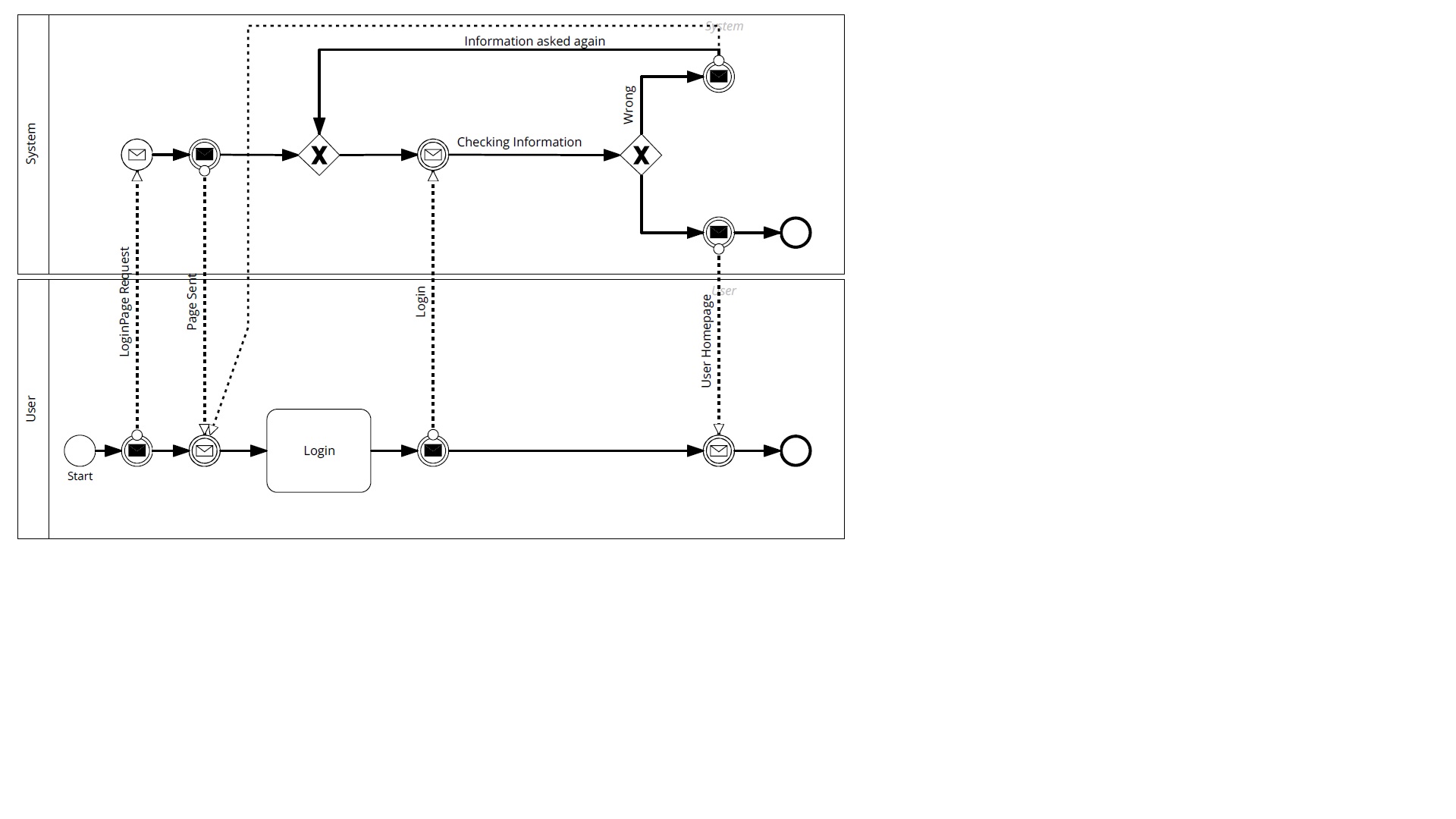
**Precondition:** The user has downloaded the application, or she/he is connected to the web portal. The user is also registered. We expect him to see something similar to fig cellulare

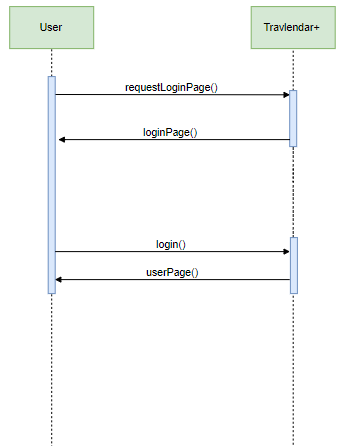
**Flow of events:**

* The user insert username and password.
* The user clicks the Login button.
* System verifies the credential.

**Exit condition:** the user is logged into the system and system shows figure

**Exception:** if the user inserted wrong credential, the system will ask them again.





### ***4.3.3 User add an event to his calendar***

### **Precondition:** The user has just logged in.

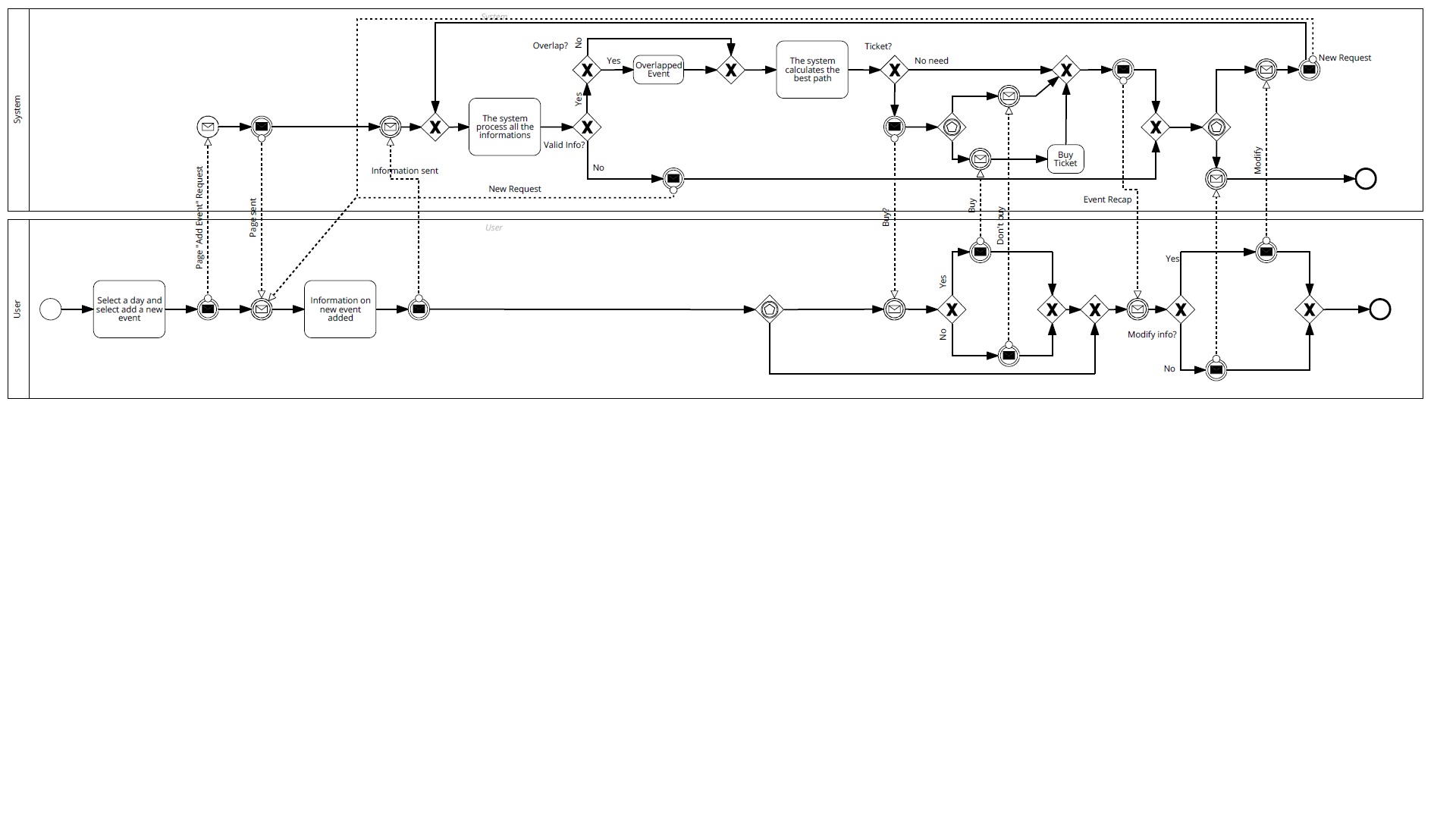
**Flow of events:**

* User selects a day.
* The system shows all the events in that particular day. figure
* The user clicks “add an event” button.
* The system displays the add event window figure
* The user fills all the information needed.
* The system checks if all information are semantically corrected. If no “change start” information is delivered, he will use the information of user’s last appointment.
* The system then calculates the user’s best path with the information the user has given to him and automatically chooses the vehicle. The system displays a recap of the event.
* The user can then modify any option or finish the operation.

**Exit condition:** User added a full event to the calendar

**Exceptions:**

* If there is an overlap of events, the system will notify the user who will still be able to continue the “add event” operation and add an “overlapped event”.
* If the user wants to use a public vehicle, the system will check if the user has already a subscription. If user hasn’t, the system will show a warning message telling the user that he needs to buy a ticket for the travel. In the message, there are also 2 buttons where the user can decide to buy the ticket himself, or to let the system buy it for him.



### ***4.3.4 User modifies an event from his calendar***

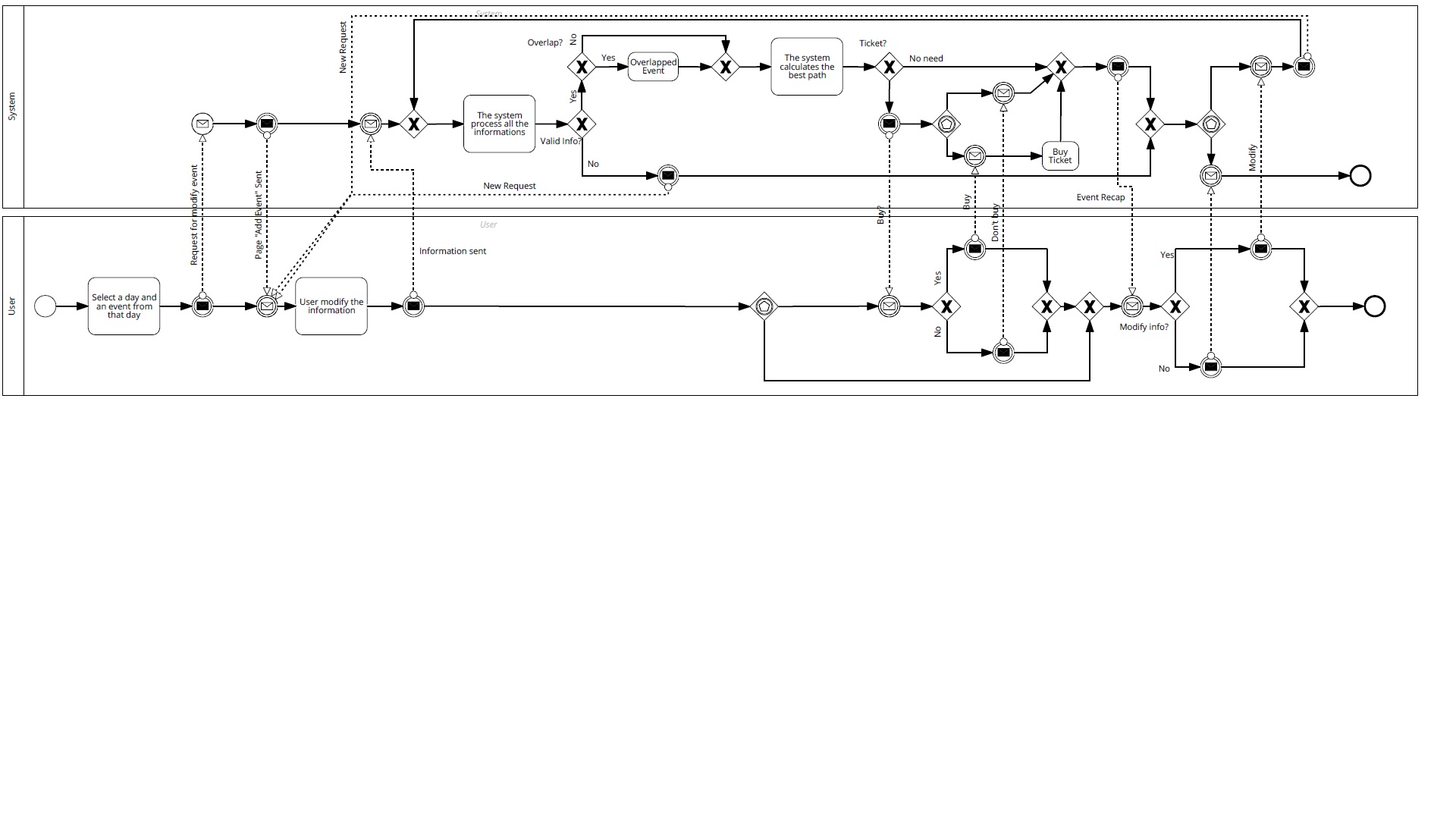
### **Precondition:** The user is logged in and added ad event to his calendar.

**Flow of events:**

* The User selects a day with an event from the calendar. figure
* The System shows all the events in that day. figure
* The user double clicks (PC) or taps (Mobile) into an event name.
* The system shows the details of that event
* The user modifies the information of the event.
* The system checks if all information are semantically corrected and acts as if a new event has been submitted doing similar operations to [4.2.3]

**Exit condition:** User added modifies an event from the calendar

**Exceptions:** A user can’t try to modify an event that doesn’t exist. So all the exceptions here come from [4.2.3]. Therefore they are the same.



### ***4.3.5 User adds a pass to his profile***

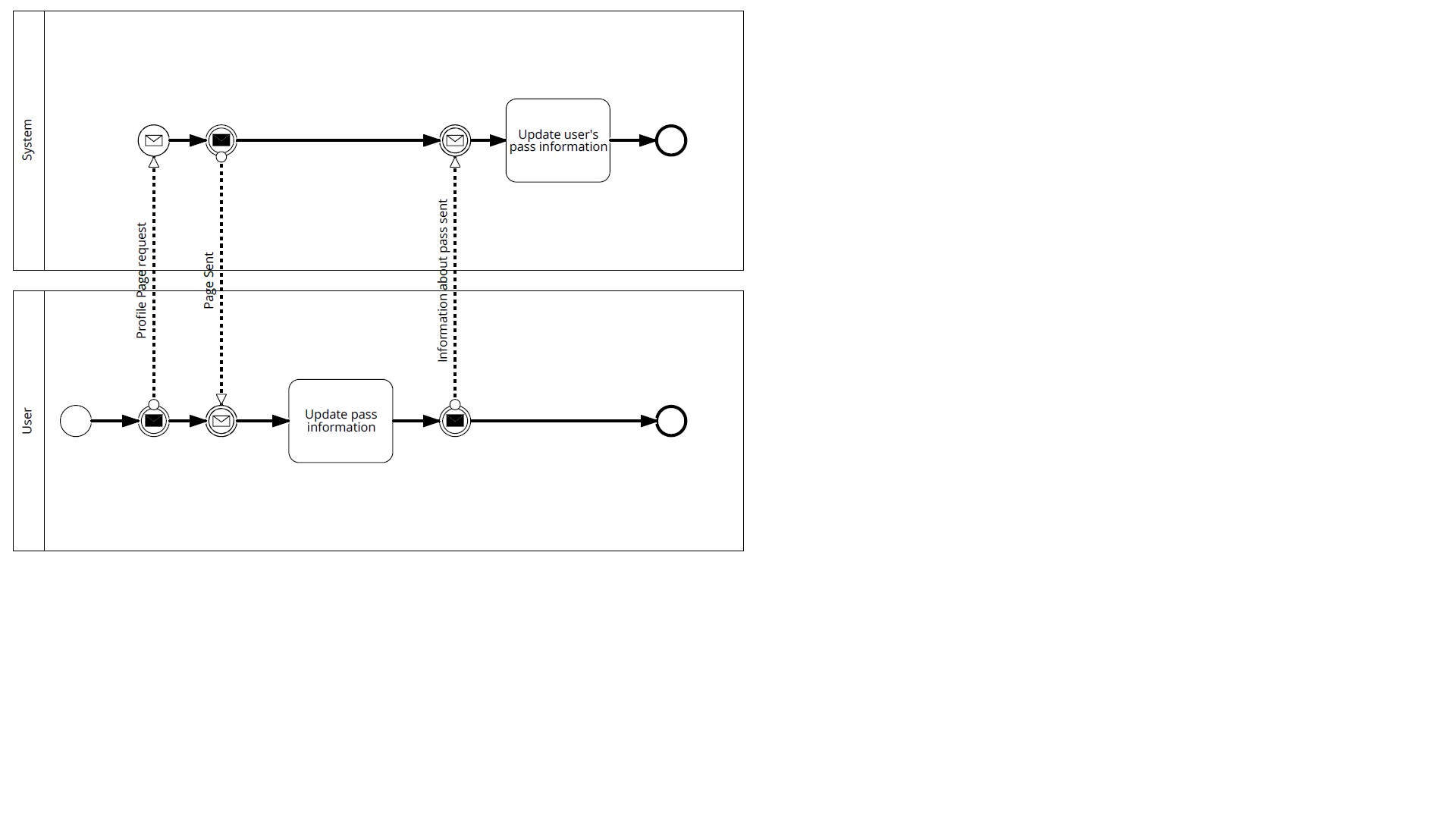
### **Precondition:** The user is logged in.

**Flow of events:**

* The User selects the profile button . figure
* The System displays the “profile window”.
* The user clicks the “Add pass…” button.
* The system display a window where it asks what type of pass the user has.
* The user fills all the information.
* The System checks if the information are semantically correct and accepts them.

**Exit condition:** User added a pass to his profile.

**Exceptions:** If informations aren’t correct the system asks the user to insert them again.



# **5. Effort Spent**

Federico Amadelli 10.5 hrs

Alessandro Artoni 15.0 hrs

Alessio Baccelli 6 hrs

# **6. References**