**1. INTRODUCTION**

1. **PURPOSE**
2. **SCOPE**
3. **DEFINITION, ACRONYMS, ABBREVIATIONS**
4. **REVISION HISTORY**
5. **REFERENCE DOCUMENTS**
6. **DOCUMENT STRUCTURE**

**2. OVERALL DESCRIPTION**

1. **PRODUCT PERSPECTIVE**
2. **PRODUCT FUNCTIONS**
3. **USER CHARATERISTIC**

**1. Introduction**

**1.A Purpose**

The aim of RASD document is to analyze the most important features of the web calendar-based application called “**Travlendar+**”. This document is intended to illustrate system goals and functionalities, taking in account interests and needs of the users in a useful way for the relationship between the developer and the consumer and for the following implementations.

The application is finalized to support the user in his daily travels and Meetings at various locations, identifying the best mobility solution. The software must grant the following functionalities in order to satisfy this purpose.

**1.A.1 Goals**

[G1] Allow a Visitor to become a User with a personal account

[G2] Allow a User to manage his Activities

[G3] The User has to be warned if a scheduled activity becomes unreachable in the accounted time.

[G4] Allow a user to choose which means of transport he prefers to reach the Activity

[G5] The application should suggest travel means depending on the nature of appointment.

[G6] Allow a User to reach an activity using the best mobility option and considering user Preferences

[G7] Allow a User to buy public transportation tickets or passes

[G8] Allow a User to use bike-sharing and car-sharing services

[G9] Allow a User to schedule an activity in a flexible way according to his other meetings.

[G10] Allow the User to view a calendar with all his scheduled activities

**1.B Scope**

**B. *Scope*: here we include an analysis of the world and of the shared phenomena**

This system wants to provide the user with personalized and flexible support which can help him in organizing his weekly commitments, from work to leisure. In doing so, it aims to suggest the best means of transport and allow the purchase of tickets or passes for public transport.

**1.C Definitions, Acronyms, Abbreviations**

**1.C.1 Definitions**

● Flexible Activity: a particular event characterized by a fixed duration that can be scheduled in a timeslice

● Meetings: a scheduled appointment characterized by a precise hour, day and destination.

● Route: a scheduled appointment characterized by a precise hour, day and destination.

**1.C.2 Acronyms**

* DDoS attack: Distributed Denial of Service attack
* ETA: Estimated Time of Arrival
* MVC: Pattern Model View Controller Pattern
* JML: Java Modeling Language

**D. Revision history**

**E. Reference Documents**

● “Fundamentals of Software Engineering”, Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli

**F. Document Structure**

**2. Overall Description**

**A. Product perspective**

**The system will be released as both mobile application and web application format, Travlendar+ will have**

**B. Product functions**

[G1] Allow a Visitor to become a User with a personal account

[R1.1] The User must be able to Sign Up to Travlendar+ by inserting their personal informations and email address.

[R1.2] During the Sign Up phase the System must verify that email address is not yet registered on the DBMS

[R1.3] The Visitor must be able to Log In from Travlendar+

[R1.4] During Log In phase the system must verify that the entered username and password are already registered into the DBMS and correct to allow the user to access.

[R 1.5] The System must store User default preferences and general settings on his account

[G2] Allow a User to manage his Activities

[R 2.1] The system must allow the User to create Activities only if the scheduled date and time are after the creation time

[R 2.2] The system must verify that the scheduled Activity is reachable on time and must show a warning if it is not.

[R 2.4] The system must allow the User to delete Activity in any moment

[R 2.5] The system must allow the User to edit Activity preferences and settings in any moment

[D 2.1] The reachability of an Activity is computed considering all the possibilities and selecting the best route from the User’s starting position

[D 2.2] The position of the User is correctly retrieved by GPS

[G3] The User has to be warned if a scheduled activity becomes unreachable in the accounted time.

[R 3.1] The System must verify that the scheduled Activity is reachable on time considering also both Public Transportation Strikes and Weather Conditions

[R.3.2] The System must correctly handle Public Transportation APIs

[R.3.3] The System must correctly handle Weather Information APIs

[R.3.4] The System must show a warning to the User If the scheduled Activity is not reachable on time.

[D 3.1] The reachability of a Activity is computed considering all the possibilities and selecting the best route from the User’s starting position

[D 3.2] The position of the User is correctly retrieved by GPS

[D 3.3] The functionality of retrieve weather conditions and forecast is correctly provided by an external service with its APIs.

[D 3.4] The functionality of retrieve information about public transportation strikes is correctly provided by an external service with its APIs.

[G4] Allow a user to choose which means of transport he prefers to reach the Activity

[R 4.1] The User must be able to select which means of transport enable for reaching that particular activity

[R 4.2] The preferences has to be correctly stored into the System and available in any moment

[G5] The application should suggest travel means depending on the nature of appointment.

[R 5.1] The User must be able to create Activity Templates in which he can choose what travel mean he want to use for each different kind of Activity.

[R 5.2] The system must suggest to the User the travel mean basing on created templates

[G6] Allow a User to reach an activity using the best mobility option and considering user Preferences

[R 6.1] The System must compute the best route by analyzing all possible paths considering traffic conditions, eventual accidents, weather conditions or Public transportation strikes.

[R 6.2] The System must take in account preferences expressed by the User to reach the Activity

[D 6.1] The System computes the best route in any condition

[D 6.2] The position of the User is correctly retrieved by GPS

[G7] Allow a User to buy public transportation tickets or passes

[R 7.1] The System must correctly handle the Public Transportation Company APIs

[D 7.1] The effective functionality is provided by an external service with its APIs.

[G8] Allow a User to use bike-sharing and car-sharing services

[R 8.1] The System must correctly handle the Bike Sharing Service APIs

[R 8.2] The System must correctly handle the Car Sharing Service APIs

[D 8.1] The position of bikes, cars and bike stations shown on the map really indicates the location

[D 8.2] The bike-sharing functionality is correctly provided by an external service with its APIs

[D 8.3] The car-sharing functionality is correctly provided by an external service with its APIs

[G9] Allow a User to schedule an activity in a flexible way according to his other meetings.

[R 9.1] The User must be able to specify a time interval in which will be done a Flexible Activity

[R 9.2] The system must compute the best time in interval specified by the User to place the Flexible Activity, according to the other meetings during the day

[G10] Allow the User to view a calendar with all his scheduled activities

[R 10.1] The System must show the User a Calendar from which he can visualize his daily meetings.

**C. User characteristics (Da finire)**

The user must have a personal computer or a smartphone to run the application of Travlendar+.

If the user decide to insert the possibility of use a car in the route track then that user must be in the age of majority and have the driving license. The application can not control whether the user is in a driving condition and therefore this is a problem solely of the user, who declares he is not in a state of being drunk.

The user agrees to these conditions during the registration to the system.

**C.1 Actor (FACOLTATIVO)**

**D. Assumptions, dependencies and constraints**

Here we include further specifications in order to avoid any kind of ambiguity in the interpretation of the document.

**D.1 Assumptions**

**Text Assumptions**

* The internet connection works correctly
* The Car sharing functionality is provided by an external service with its APIs
* Credentials that a visitor has to provide to become a registered user are: name, surname, address, email, telephone number and username.
* Weather Forecast are provided and constantly updated by an external service
* The required functionalities provided are ambiguous about how the software suggest the travel mean “depending” on the appointment: we assume that the User can set his preferences for each kind of appointment.
* When a destination address is inserted, the map with the route will be available in most 30 seconds.
* The Route computed by the software is the best Route according to ETA

**Domain Assumptions**

* The User must be identified uniquely by the username.
* User location is correctly retrieved by GPS.
* When the system shows a car or a bike in a certain position it means that it's actually there

**3. Specific Requirements**

**3.A External Interface Requirements**

**3.A.1 User Interfaces**

The following mockups represent a basic idea of what the mobile app will look like in the first release.

**3.A.2 Hardware Interfaces**

The system does not have any hardware interfaces except for accessing of GPS data.

**3.A.3 Software Interfaces**

The system operates using software to store and save data, as well as databases and DBMSs. The following software are used: MySQL version 5.7.19, Microsoft Access version 2016 in order to manage data and queries. **The system needs also to use external APIs and to expose interfaces in order to interact with other systems.**

**The mobile application must support Android, iOS and Windows Phone. The web application works on any web server that supports Java.  
The back-end stores its data in a RDBMS and can run on every platform that supports the JVM. The back-end must offer APIs for user interfaces and external modules, like:  
 taxi reservation  
 car/bike sharing  
 online payments**

**3.A.4 Communication Interfaces**

**3.B Functional Requirements**

**3.B.1 Scenarios**

**3.B.1.1 Sign up in the System and Log In**

Francesco works as a trading agent for large company in Milan and he is in need of organizing a great deal of appointments in different areas of the city during the day. He finds the new service **Travlendar+** and decides to download it from the Play Store on his Android device and sign up to better organize his commitments. For the registration, the System provides a form with different fields: username, name,  
surname, birth information, email address and optional personal information to complete the profile. Then Francesco fills in all the fields and pushes the registration button. The system approves the registration and sends to him a confirmation email with a secure password that allows him to login into the system from there. Now Francesco can better organize his activities during the day using Travlendar+. Some days after the registration, Francesco decided to log in the system with his email address and password, which will be stored into the system whenever he decides to relog to the application. If he would decide to close or uninstall the app, he could run a Log Out to disconnect from the application.

**3.B.1.2 Creation of an Activity and Purchase of public transport tickets**

It's half past four of a rainy afternoon and Francesco has to be in a sales outlet at 6pm, then he decides to create an Activity. He inserts the scheduled time and the exact address but he have not any constraint about the mean of transport, and then he click on Create the Activity. The application calculates the best route, suggesting the metro as best mean, to be taken at 5.36 some stops away from the arrival station; after that he will have to walk for 6 minutes to reach the exact destination. The system preferred the metro to the bicycle not to let Francesco get wet to the place, and avoid the car after detecting heavy traffic it that zone; then he decided to buy the ticket using the app, to not to waste time going to the metro ticket office.

**3.B.1.3 Activity modification**

Some minutes later, the store manager calls Francesco by advising him of moving the appointment two hours later. Francesco then clicks on "Edit Activity" on the activity previously scheduled in his homepage, and enter the new scheduled hour for the appointment. The system recalculates the route, but now the app suggests to the user a car as means of transport since the road is tendentially much less busy for that hour; the bike is still avoided because rain is expected until 10 pm.

**3.B.1.4 Bike or Car Sharing service utilization**

It’s half past ten in the following morning and Francesco has to reach a client almost 2 miles from his position at 11.15, then he decides again to create an Activity in Travlendar. It's fortunately a sunny day and Francesco would prefer to avoid taking the car to enjoy the good weather, so he selects the "Avoid Machine" option and creates the Meeting. The app suggests a bike sharing service: the closest free bike is 300 meters away and he books it, then he starts to walk to reach the bike.

**3.B.1.5 Creation of Activity Template**

However, after that activity Francesco realizes that using a bike to get to job appointments, especially near the summer season, can be overly tiring. Noticing the Create to Template functionality then he decides to use it to create a specific Template suitable for work appointments, which excludes the bike from the means of transport. At the same time, going to sports appointments may require to arrive already warmed up, so bike could be the right means of transport: Francesco creates another template.

**3.B.1.6 Creation of Flexible Activity**

Lunch time is approaching and Francesco needs to optimize the available time available due to many appointments scheduled from 2.30 pm onwards. He decided to use the Flexible Activity functionality, which offers him the possibility to specify a range in which positionate a task of a particular duration, choosing the best best time according to his next activities. Francesco therefore chooses a half-hour lunch to be entered between 12.30 - 14.30; the application chooses 13.15 to 13.45 to

to leave Francesco the time needed to reach the destination of first appointment.

To reach Activities scheduled for the first afternoon, the app does not consider the subway option according to informations received from the system about an ATM strike which will last until 6 o'clock in the afternoon.

**3.B.2 Activity Diagram**

**3.B.3 Use Case Model**

We can deduce some use cases from the scenarios previously described. The main actors are:

* **Visitor**: a guest who is not yet authenticated in the system
* **User:** an app visitor who has already signed up in the system and has an account

**3.B.3.1 Visitor Use Cases**

**Sign up to the system [UC1]**

|  |  |
| --- | --- |
| **Goal** |  |
| **Requirement** |  |
| **Assumption** | Visitor is not registered yet to the system |
| **Actors** | Visitor |
| **Input Condition** | The Visitor must have the app downloaded on the smartphone |
| **Output Condition** | All fields are correctly filled in by the Visitor, also the box in which he agrees to the treatment of his personal informations. |
| **Event Flow** |  |
| **Exceptions** |  |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Log In to the System [UC2]**

|  |  |
| --- | --- |
| **Goal** | **[G2]** The Visitor must be able to Log In to the service |
| **Requirement** | **[R2.1]** The system must verify that the entered username and password are already registered into the DBMS.  **[R2.2]** If username and password are correct, the system allows the User to log in. |
| **Assumption** | Visitor is registered. |
| **Actors** | Visitor, User |
| **Input Condition** | The Visitor must have the app downloaded on the smartphone and have opened it. |
| **Output Condition** | The username and the password are correct. |
| **Event Flow** | 1. The user opens the "Travlendar +" app.   2) The User enters his Username and Password in their respective fields.  3) The User clicks on “Log In” button.  4) The System checks the correctness of the informations inserted.  5) The Systems shows on the display the App homepage |
| **Exceptions** | Username or password are not correct. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**3.B.3.2. User Use Cases**

**Create an Activity [UC3]**

|  |  |
| --- | --- |
| **Goal** | **[G4]** Allow a User to create Activities |
| **Requirement** | **[R4.1]** The System must allow the User to create Activities only if the scheduled date and time are after the creation time  **[R4.2]** The System must verify that the scheduled Activity is reachable on time.  **[R4.3]** The System must show a warning to the User If the scheduled Activity is not reachable on time. |
| **Assumption** | User is registered, has logged in and navigates on the homepage. |
| **Actors** | User |
| **Input Condition** | The User navigates in his Homepage. |
| **Output Condition** | The User has scheduled an Activity. |
| **Event Flow** | 1. The User selects the “Create an Activity” option.  2. The User enters Activity date and hour.  3. The User chooses his preferences about the Activity. 4. The system divides the trip into Subroutes, suggesting the best means of transport for each one. 5. If the destination is not reachable at the scheduled time, the system notifies the user with a warning |
| **Exceptions** | The User deletes the operation. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Edit an Activity [UC4]**

|  |  |
| --- | --- |
| **Goal** | **[G5]** Allow a User to edit Activities |
| **Requirement** | **[R 5.1]** The system must allow the User to edit Activity preferences and settings in any moment |
| **Assumption** | User is registered, has logged in and has already created at least one Activity. |
| **Actors** | User |
| **Input Condition** | The User navigates in his Homepage. |
| **Output Condition** | All the modified informations about the Activity are saved. |
| **Event Flow** | 1. The User selects the “Edit” option on a Activity in his homepage. 2. The System shows all the preferences option about the Activity. 3. The User changes one or more Settings or Preferences about the Activity. 4. The System computes again the best route based on new preferences.  5. If the destination is not reachable at the scheduled time according to the new Settings or Preferences, the system notifies the user with a warning.  6. The changes about the Activity are stored. |
| **Exceptions** | The User does not change anything. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Delete an Activity [UC5]**

|  |  |
| --- | --- |
| **Goal** | **[G6]** Allow a User to delete Activities |
| **Requirement** | **[R 6.1]** The system must allow the User to delete Activities in any moment |
| **Assumption** | User is registered, has logged in and has already created at least one Activity. |
| **Actors** | User |
| **Input Condition** | The User navigates in his Homepage. |
| **Output Condition** | The User has deleted an Activity. |
| **Event Flow** | 1. The User selects the “Delete” option on an Activity in his Homepage. 2. The system asks the User a confirmation about his will to delete the selected Activity  3. The System deletes the Activity.  4. Eventual Flexible Activities are computed again according to new disposition of Activities |
| **Exceptions** | The User deletes the operation. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Create an Activity Template [UC6]**

|  |  |
| --- | --- |
| **Goal** | **[G10]** Allow a User to create Activities templates |
| **Requirement** | **[R 10.1]** The User must be able to create a personalized template that could be reused for new specific kind of Activities. |
| **Assumption** | User is registered and has logged in. |
| **Actors** | User |
| **Input Condition** | The User navigates in his Homepage. |
| **Output Condition** | The User has created a new Template. |
| **Event Flow** | 1. The User selects the “Create a Template” option in his Homepage.  2. The User enters Template general settings and preferences.  3. The new Template is stored in User account. |
| **Exceptions** | The User deletes the operation. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Create a Flexible Activity [UC7]**

|  |  |
| --- | --- |
| **Goal** | **[G15]** Allow a User to schedule Flexible Activities |
| **Requirement** | **[R 15.1]** The User must be able to specify a time interval in which will be done a Flexible Activity  **[R 15.2]** The system must compute the best time in interval specified by the User to place the Flexible Activity, according to the other meetings during the day |
| **Assumption** | User is registered and has logged in. |
| **Actors** | User |
| **Input Condition** | The User navigates in his Homepage. |
| **Output Condition** | The User has created a new Flexible Activity. |
| **Event Flow** | 1. The User selects the “Create a Flexible Activity” option.  2. The User enters Flexible Activity hour range and duration.  3. The User chooses his preferences about the Flexible Activity.  4. The system computes the best time into the range for the Flexible Activity, according to other Activities during the day. 5. The system divides the trip into Subroutes, suggesting the best means of transport for each one. |
| **Exceptions** | The User deletes the operation. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Create an Activity Template [UC8]**

|  |  |
| --- | --- |
| **Goal** | **[G10]** Allow a User to create Activities templates |
| **Requirement** | **[R 10.1]** The User must be able to create a personalized template that could be reused for new specific kind of Activities. |
| **Assumption** | User is registered and has logged in. |
| **Actors** | User |
| **Input Condition** | The User navigates in his Homepage. |
| **Output Condition** | The User has created a new Template. |
| **Event Flow** | 1. The User selects the “Create a Template” option in his Homepage.  2. The User enters Template general settings and preferences.  3. The new Template is stored in User account. |
| **Exceptions** | The User deletes the operation. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Modify General Settings [UC9]**

|  |  |
| --- | --- |
| **Goal** | **[G9]** Allow a User to store his own default preferences for a new generic Activity |
| **Requirement** |  |
| **Assumption** | User is registered and has logged in. |
| **Actors** | User |
| **Input Condition** | The User navigates in his Account page. |
| **Output Condition** | The User has changed General Settings. |
| **Event Flow** | 1. The User selects the “Edit General Setting”  2. The User modifies some General Settings.  3. The changes are stored in User account. |
| **Exceptions** | The User deletes the operation. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**View Calendar [UC10]**

|  |  |
| --- | --- |
| **Goal** | **[G17]** Allow the User to view a calendar with all his scheduled Activities |
| **Requirement** | **[R 17.1]** all the scheduled activities must be correctly retrieved from DBMS |
| **Assumption** | User is registered, has logged in and has already created at least one Activity. |
| **Actors** | User |
| **Input Condition** | The User in already in his Homepage. |
| **Output Condition** | The User has successfully viewed all his meetings |
| **Event Flow** | 1. The User select the option “View Calendar” in his home page |
| **Exceptions** |  |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**View Activity Details [UC11]**

|  |  |
| --- | --- |
| **Goal** | **[G18]** Allow the User to view all details related to a scheduled Activity |
| **Requirement** | **[R 18.1]** The information about the scheduled Activity are correctly retrieved from DBMS |
| **Assumption** | User is registered, has logged in and has already created at least one Activity.  **[D 18.1]** Weather Information are correctly retrieved from an External API  **[D 18.2]** Traffic Information are correctly retrieved from an External API  **[D 18.3]** Information about Public Transportation Strike are correctly retrieved from an External API |
| **Actors** | User |
| **Input Condition** | The User is in the Calendar View |
| **Output Condition** | The User has successfully received all details for his meeting |
| **Event Flow** | 1. The User selects an Activity in his calendar View  2. The User views all details related to that meeting |
| **Exceptions** | The User deletes the operation. |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Navigate to Activity [UC12]**

|  |  |
| --- | --- |
| **Goal** |  |
| **Requirement** |  |
| **Assumption** |  |
| **Actors** | User |
| **Input Condition** | The User is in “Activity Details” view |
| **Output Condition** | The User has successfully reached the Activity |
| **Event Flow** | 1. The user select “Navigate to Activity” option  2. The User choose his preferred means of transport for this activity  3. The User starts the navigation following the route computed by the System |
| **Exceptions** | 1. Warning is shown if There is a transportation strike and Public Transport is a mean to reach the activity 2. Warning is shown if It will rain when the User is going to use bike 3. The User deletes the operation |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**Buy Public Transportation Ticket [UC13]**

|  |  |
| --- | --- |
| **Goal** | [G12] Allow a User to buy public transportation tickets or passes |
| **Requirement** | [R 12.1] The System must correctly handle the Public Transportation Company APIs |
| **Assumption** | [D 12.1] The effective functionality is provided by an external service with its APIs. |
| **Actors** | User |
| **Input Condition** | The User is performing Navigation to his Activity |
| **Output Condition** | The User has successfully bought his public transportation Ticket for the desired route |
| **Event Flow** | 1. When a Public Mean of transport is used to reach an Activity the user can select the button “Buy Public Transport Tickets”  2. The System using the APIs redirects the User to the page where he can buy the Ticket for the interested Trip.  3. The User completes other procedures on the external Website |
| **Exceptions** |  |
| **Non functional Requirements** |  |
| **Sequence diagram** |  |

**3.C Performance Requirements**

The system has to have low response time and to held a great number of simultaneous requests in an acceptable time, which means always less than 25 seconds for each one.

When a destination address is inserted, the map with the route will be available in most 15 seconds.

**3.D Design constraints**

**3.D.1 Standards compliance**

The user must ensure that the personal use of the application is in compliance with current regulations and laws of his own country.

The user must give consent to the application to access the GPS before using the application. The application will store and process personal data only for the purpose of properly running the service.

**3.D.2 Hardware limitations**

The mobile application works only in Android, iOS and Windows Phone devices. The client app must be able to access the GPS data of the user's phone. Connection for mobile: 3G or 4G connection. To install and use the app the user must have 120MB of free space in his device.

The web application requires a browser that can identify the user's location and show pop ups. The browser must support also the HTTPS protocol. Display resolution must be 800x600 or more.

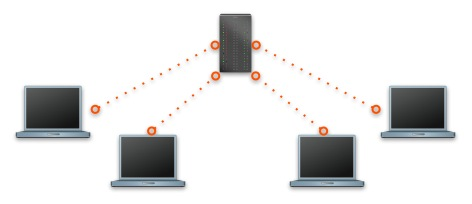
Internet connection: The application must be able to work if the user dispose of a 2Mb/s connection

**3.D.3 Any other constraint**

**DA VEDERE**

**3.E Software System Attributes**

**3.E.1 Reliability**

****

The system is designed for the only centralized environment. The reliability of the system is strictly related to the reliability of the server it runs on. The system must ensure the truthfulness and accurateness of the data shown to the Users.

**3.E.2 Availability**

The system must guarantee an availability of 99.4%. This means approximately 2 days of downtime per year. Normally the system should be accessible 24 hours per day. Within 2 years we want to bring this percentage to 99.6%. Application updates are frequent but only a very small part could stop ordinary operations.

**3.E.3 Security**

Because Meeting scheduling is a strictly personal matter, for privacy issues it must be ensured that User passwords are properly stored in the System. In particular, passwords and also personal data are stored, encrypted and not in text format, in a proper database. The users’ passwords are stored in the database using a proper hashing mechanism.

Every 60 days the application reminds users that there is the possibility to change their password in order to gain more security of their data.

All the communications between clients and server must be protected by strong encryption using the SSL protocol. The server does not allow visitors and users to establish unsafe connections.

The system has a software that detects bots (non-human users) in order to prevent potential external malicious actions. The server is thus able to protect itself against possible DDoS attacks.

**3.E.4 Maintainability**

The entire application code, written mainly in Java, will be documented to facilitate the work of current developers and to well inform future developers of how it has been developed and how application works. A standard for writing code, JavaDoc and JML specifications will be used. Additionally, the MVC Pattern will be used to facilitate code maintenance.

**3.E.5 Portability**

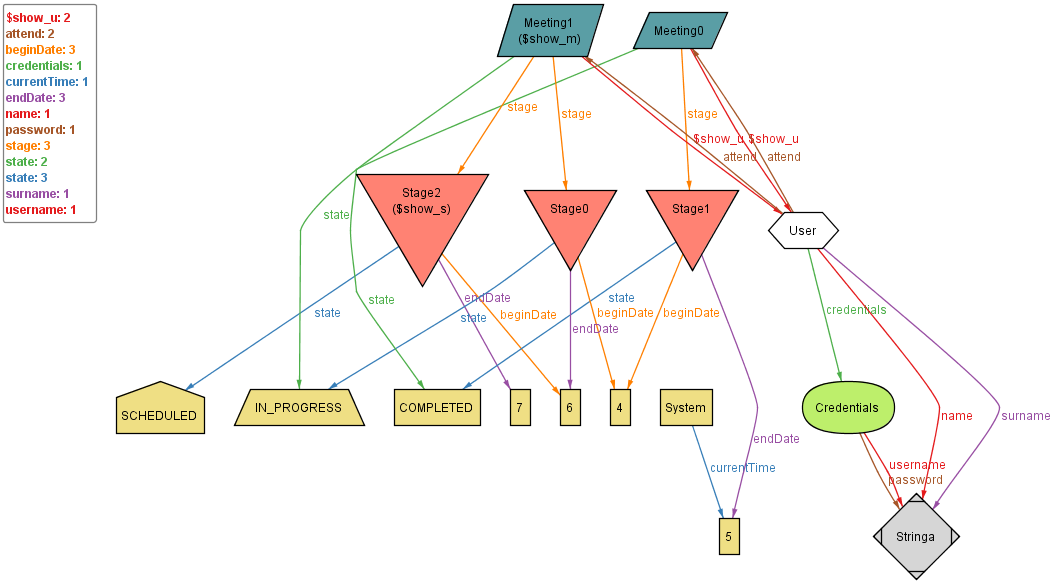
The mobile application must be supported by the last 3 major versions of Android, iOS and Windows Phone. The web application must be supported by the last 2 major versions of Google Chrome, Mozilla Firefox and Safari (browsers). The system must also guarantee an high level of scalability.  
**4. Formal analysis using Alloy**

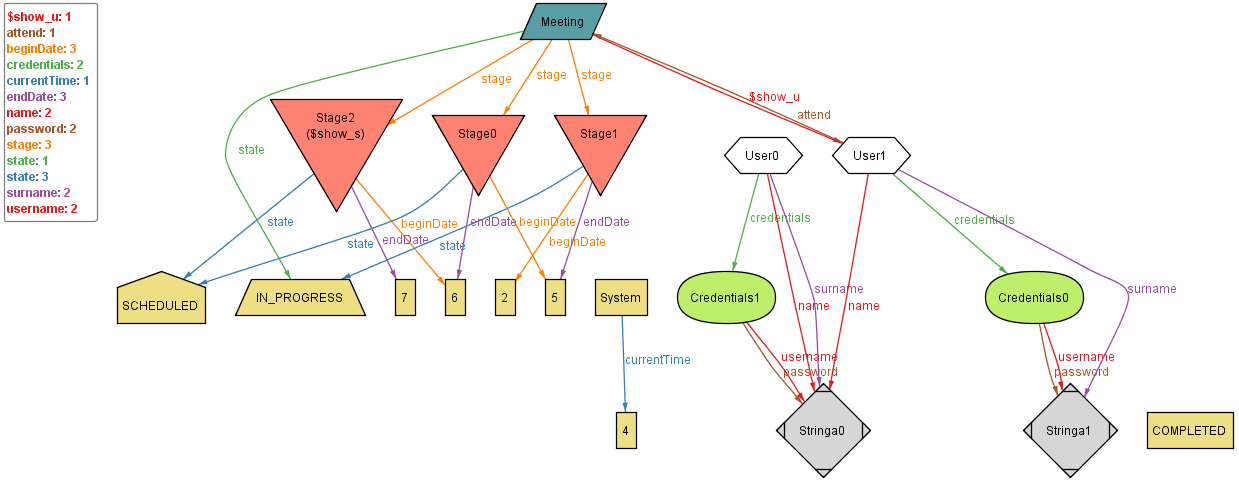
**VERSIONE NON DEFINITIVA (E CON GRAFICO NON DEFINITIVO)**

sig Stringa{}  
  
sig User{  
 name: one Stringa,  
 surname: one Stringa,  
 credentials: one Credentials,  
 attend: set Meeting  
}  
  
sig Credentials{  
 username: one Stringa,  
 password: one Stringa  
}  
  
sig Meeting{  
 state: one MeetingState,  
 stage: some Stage  
}  
  
sig Stage{  
 state: one MeetingState,  
 beginDate: one Int,  
 endDate: one Int  
}{beginDate >= 0 and endDate > beginDate}  
  
abstract sig MeetingState{}  
sig IN\_PROGRESS extends MeetingState{}  
sig SCHEDULED extends MeetingState{}  
sig COMPLETED extends MeetingState{}  
  
sig System{  
 currentTime: one Int  
}{currentTime>=0}  
  
//un solo sistema  
fact f0{  
 #System = 1  
}  
  
//non esistono due persone con lo stesso username  
fact f1{  
 no disj u1, u2: User | u1.credentials.username = u2.credentials.username  
}  
  
//non esistono credenziali senza il corrispettivo proprietario  
fact f2{  
 all c: Credentials | one u : User | u.credentials = c  
}  
  
//se un meeting è completato allora tutti i suoi stages sono completati  
fact f3{  
 all m: Meeting | all s: Stage | s in m.stage and   
 m.state in COMPLETED implies s.state in COMPLETED  
}  
  
//se un meeting è solo programmato, allora tutti i suoi stage sono solo programmati  
fact f4{  
 all m: Meeting | all s: Stage | s in m.stage and   
 m.state in SCHEDULED implies s.state in SCHEDULED  
}  
  
//se un meeting è in corso allora esiste uno e un solo stage in corso relativo al meeting  
fact f5{  
 all m: Meeting | m.state in IN\_PROGRESS implies   
 (one s: Stage | s in m.stage and s.state in IN\_PROGRESS)  
}  
  
//un utente può avere al massimo un solo meeting in corso  
fact f6{  
 all u: User | lone m:Meeting | m in u.attend and m.state in IN\_PROGRESS  
}  
  
//non esistono meeting senza il corrispettivo proprietario  
fact f7{  
 all m: Meeting | some u : User | m in u.attend  
}  
  
//non esistono stage senza il corrispettivo meeting  
fact f8{  
 all s: Stage | one m : Meeting | s in m.stage  
}  
  
//tutti gli stage antecedenti alla data corrente sono completati  
fact f9{  
 all s: Stage | s.state in COMPLETED  
 implies (all sys: System | s.endDate <= sys.currentTime)  
}  
  
//tutti gli stage programmati dopo la data corrente sono nello stato di scheduling  
fact f10{  
 all s: Stage | s.state in SCHEDULED  
 implies (all sys: System | s.beginDate >= sys.currentTime)  
}  
  
//tutti gli stage in corso hanno sys.currentTime compreso tra inizio e fine  
fact f11{  
 all s: Stage | s.state in IN\_PROGRESS  
 implies (all sys: System | s.beginDate < sys.currentTime  
 and sys.currentTime < s.endDate)  
}  
  
//gli stages nei meeting sono ben formati (contiguità e coerenza)  
fact f12{  
 all m: Meeting | #m.stage > 1 implies  
 ((one s: Stage| s in m.stage and no e:Stage|  
 s.beginDate = e.endDate) and  
 (one s: Stage| s in m.stage and no e:Stage|  
 e.beginDate = s.endDate))  
}

//non ci sono più stages che iniziano allo stesso tempo nello stesso meeting  
fact f13{  
 no disj s1,s2: Stage | one m: Meeting |  
 s1 in m.stage and s2 in m.stage and s1.beginDate = s2.beginDate  
}  
  
//non ci sono più stages che finiscono allo stesso tempo nello stesso meeting  
fact f14{  
 no disj s1,s2: Stage | one m: Meeting |  
 s1 in m.stage and s2 in m.stage and s1.endDate = s2.endDate  
}

pred show{  
 #COMPLETED = 1  
 #IN\_PROGRESS = 1  
 #SCHEDULED = 1  
 some m: Meeting | #m.stage > 1  
 some s: Stage | s.state in SCHEDULED  
}  
run show

****

****

**5. Effort Spent**

5/10/2017 (Insieme): 2hrs

7/10/2017 (Insieme): 2hrs

10/10/2017 (Insieme): 4hrs

12/10/2017 (Insieme): 2hrs

17/10/2017 (Insieme): 4hrs

21/10/2017 (Insieme): 8hrs

**6. References**

**Cose da chiarire:**