**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 the User to have a personal account

[G2] Allow a User to create Meetings

[G2.1] If the scheduled Meeting is unreachable at the accounted time the User is warned

[G2.2] Allow a User to choose preferences about the Meeting

[G2.3] Allow a User to create Meetings templates that can be reused for new Meetings.

[G3] Allow a User to delete Meetings

[G4] Allow a User to modify Meetings

[G5] Allow a User see all the informations about scheduled Meetings in any time

[G.5.1] If the system detects any critical changes about the journey conditions the User is warned

[G6] Allow a User to define preferences about the journey at any time

[G6.1] The User can avoid particular mean of transport

[G6.2] The User can avoid particular mean of transport in a given time interval

[G6.3] The User can express a limit for walking distances

[G6.4] The User can choose to minimize carbon footprint

[G7] Allow a User to reach a Meeting using the best mobility option

[G8] Allow a User to buy public transportation tickets or passes

[G9] Allow a User to use bike-sharing services

[G10] Allow a User to use car-sharing services  
[G11] Allow a User to schedule Flexible Activities

[G12] The application should also suggest travel means depending on the appointment

**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 after providing credentials

[R1.1] The system must verify that the email address is not yet registered on the DBMS

[R1.2] The system allows Guests to register themselves, by inserting their personal informations.

[G2] The Visitor must be able to Log In to the service

[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.

[G3] The User must be able to Log Out to the service

[R.3.1] The system must correctly Log Out the User after his request

[G4] Allow a User to create Meetings

[R4.1] The system must allow the User to create Meetings only if the scheduled date and time are after the creation time

[R4.2] The system must verify that the scheduled meeting is reachable on time.

[R.4.3] The system must show a warning to the User If the scheduled Meeting is not reachable on time.

[D4.1] The reachability of a Meeting is computed considering all the possibilities and selecting the best route from the User’s starting position

[D4.2] The position of the User is correctly retrieved by GPS

[G5] The User has to be warned If a scheduled Meeting becomes unreachable in the accounted time.

[R 5.1] The System must verify that the scheduled meeting is reachable on time.

[R.5.2] The System must show a warning to the User If the scheduled Meeting is not reachable on time.

[D5.1] The reachability of a Meeting is computed considering all the possibilities and selecting the best route from the User’s starting position

[D5.2] The position of the User is correctly retrieved by GPS

[G6] Allow a User to choose preferences about the Meeting

[R 6.1] The Preferences has to be correctly stored into the System and available in any moment

[G7] Allow a User to store his own default preferences for a new generic Meeting

[R 7.1] The System must save User default preferences on his account

[G8] Allow a User to create Meetings templates

[R 8.1] The User must be able to create a personalized template that could be reused for new specific kind of Meetings.

[G9] Allow a User to reach a Meeting using the best mobility option

[R 9.1] The System must compute the best route by analyzing all possible paths considering traffic conditions, eventual accidents, weather conditions or strikes.

[D 9.1] The reachability of a Meeting is computed considering all the possibilities and selecting the best route from the User’s starting position, according to ETA

[D 9.2] The position of the User is correctly retrieved by GPS

[G10] Allow a User to buy public transportation tickets or passes

[D 10.1] The effective functionality is provided by an external service with its APIs.

[R 10.1] The System must allow the User to select his purchase option

[R 10.2] The System must verify that the payment method inserted is valid

[G11] Allow a User to use bike-sharing services

[D 11.1] The position of bikes or bike stations shown on the map really indicates the location

[D 11.2] The bike-sharing functionality is provided by an external service with its APIs

[R 11.2] The system must verify that the payment method inserted is valid

[G12] Allow a User to use car-sharing services

[D 12.1] The position of cars shown on the map really indicates their location

[D 12.2] The car-sharing functionality is provided by an external service with its APIs

[R 12.1] The System must verify that the payment method inserted is valid

[G13] Allow a User to schedule Flexible Activities

[R 13.1] The User must be able to specify a time interval in which will be done a Flexible Activity

[R 13.2] 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

[G14] The application should suggest travel means depending on the appointment.

**C. User characteristics**

**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**

**3.A.2 Hardware Interfaces**

**3.A.3 Software Interfaces**

**3.A.4 Communication Interfaces**

**3.B Functional Requirements**

**3.B.1 Scenarios**

3.B.1.1 Sign up in the System [G1] [G7]

3.B.1.2 Login and Log Out [G2] [G3]

3.B.1.3 Event creation [G4] [G8] [G9]

3.B.1.4 Event modification [G5] [G6] [G13]

3.B.1.5 Cars or bike sharing [G11] [G12]

3.B.1.6 Purchase of public transport tickets or passes [G10]

**QUI VANNO I CASI D’USO, GLI UML ETC.**

**3.B.2 Activity Diagram**

**VISITOR**

|  |  |
| --- | --- |
| **Sign up in the system [UC1]** | |
|  |  |
| **Goal** | [G1] Allow a Visitor to become a User after providing credentials |
| **Requirement** | [R1.1] The system must verify that the email address is not yet registered on the DBMS  [R1.2] The system allows Guests to register themselves, by inserting their personal informations. |
| **Assumption** | Visitor is not registered yet to the system |
| **Actors** | Visitor |
| **Entry Condition** | The Visitor must have the app downloaded on the smartphone |
| **Exit Condition** | All fields are correctly filled in by the Visitor, also the box in which he agrees to the treatment of his personal informations. |
| **Flow of Events** |  |
| **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.

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

**3.D Design constraints**

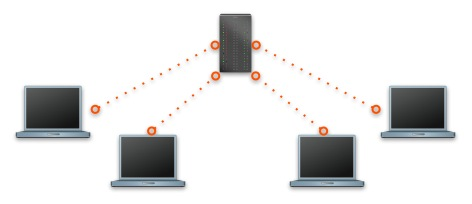
**3.D.1 Standards compliance**

**3.D.2 Hardware limitations**

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

**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**

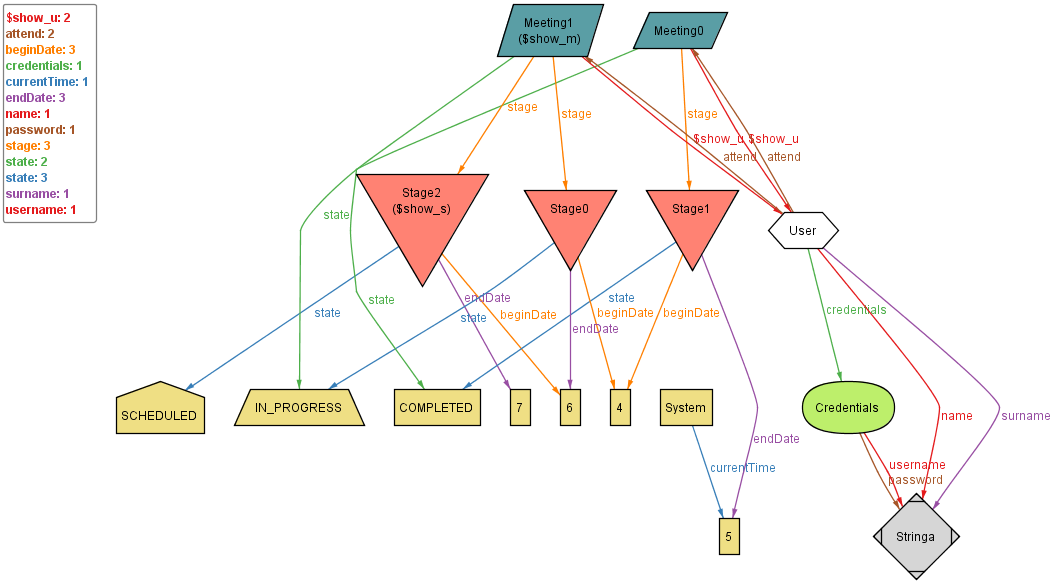
The mobile application must be supported by the last 3 major versions of Android, iOS and Windows Phone. There isn’t a web application for this software application. The system must also guarantee an high level of scalability.  
**4. Formal analysis using Alloy**

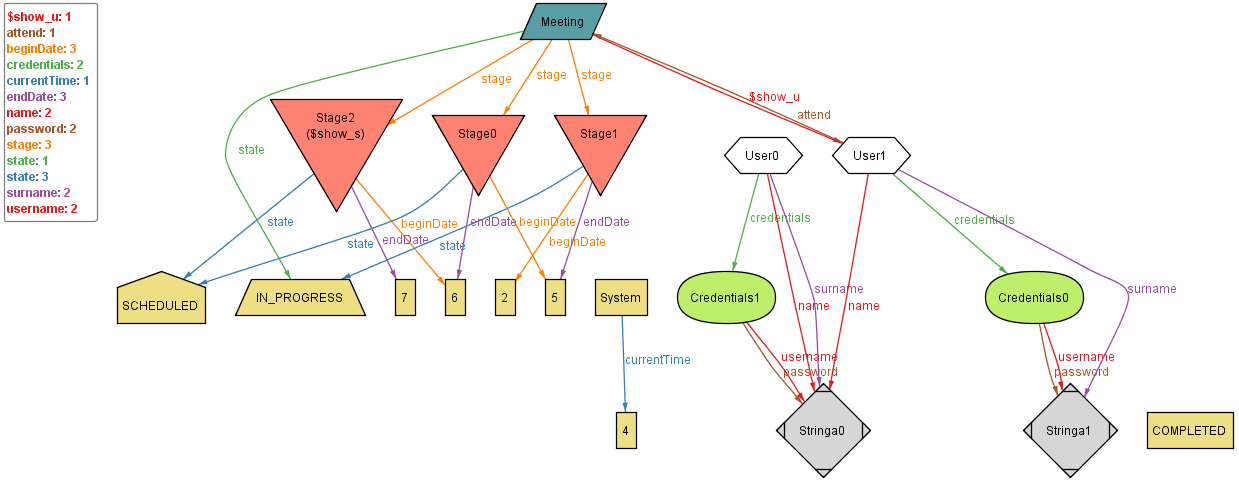
**VERSIONE NON DEFINITIVA (E CON GRAFICO NON DEFINITIVO)**

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 (da aggiornare gli ultimi giorni)**

5/10/2017 (Insieme): 2hrs

7/10/2017 (Insieme): 2hrs

10/10/2017 (Insieme): 4hrs

12/10/2017 (Insieme): 2hrs

**6. References**

**Cose da chiarire:**

* **Come trattiamo le API nello Use Case Diagram ?**