# SOFTWARE ENGINEERING 2: TRAVLENDAR+

## REQUIRMENTS ANALYSIS AND SPECIFICATION DOCUMENT

Agostini Andrea, Ciampiconi Lorenzo, Es-skidri Rachid

OCTOBER 29, 2017

# CONTENTS

# INTRODUCTION

## PURPOSE

This document is the Requirement Analysis and Specification Document for the Travlendar+ application. Goals of this document are to completely describe the system, its components, functional and non-functional requirements, constraints, and relationships with the external world, analyze the real needs of the customer in order to model the system, and to provide typical use cases and scenarios. This document is written for project managers, developers, testers and Quality Assurance. It may be useful also to users. It may be used in a contractual requirement.

## SCOPE

### Analysis of the world phenomena

Nowadays, time management is one of the most important things in the context of today's society, especially in big cities, where the variety of means of transport is so great to allow a better optimization of a person's events organization. Travlendar+ borns as  a digital calendar that allows the user not only to display his events, but also to provide the user the best way to reach the events in the best possible way, according to the criteria chosen by him.

### Analysis of the shared phenomena

There are two different kinds of shared phenomena, the first one  includes phenomena that are controlled by the world and observed by the machine, such as the GPS position of the user, the traffic, the weather and for example, something quite abstract such as the time schedule of a bus or a tube. The second one contains all those phenomena controlled by the machine and observed by the world (in according to the domain) such as : the user follows Travlendar+  indications, the user inserts the true duration time events, the user uses the system inside the correct geographic area etc.

## GOAL OF THE PROJECT

Travlendar+ is a project that has the goal to define a new idea of calendar.

This system is mean to be a real assistant, because it helps persons to organize their days suggesting solution to move from each appointment and event. The system has to be able to register new user with their information. Once a user is registered to the application, he should be able to manage his calendar by adding new events or modifying and deleting existing ones. The peculiar feature of the system is to suggest travel means by appointment (including every kind of transport service like Bike-Car Sharing, taxi, public transport etc.) and Allow user to buy a ticket or to reserve a mean of transport for the desired solution . The system will generate a warning in case an event is unreachable in the desired time and if it overlaps with another appointment. The application will be user customizable (e.g., the user will be able to select preferred (or non-preferred) means of transportation). The system will offer to the users also the possibility to insert a flexible breaks to keep in mind during the planning of the day. A complete list of the goals of the project is in the functional requirements sections.

To achieve these goals we will develop a client-server solutions, where the client side consist in an Android application (in the first release).

## DEFINITIONS

- **User**: any individual subscribed to the service.
- **Visitor**: an individual not subscribed to the service.
- **Event**: an appointment that could be registered in the calendar.
- **Free Time Interval**: with this term we refer to the interval of time that user could registered in the calendar to indicate where the flexible break must be spent.
- **Overlapping events**: when two events A and B overlaps it means that they share a time interval. More formally, when A starts before the start of B and A ends after the end of B, A overlaps with B.
- **System**: the whole software system to be developed, comprehensive of all its parts and modules.

## ACRONYMS

- **RASD**: Requirements Analysis and Specification Document (this document).
- **API**: Application Programming Interface.
- **UI**: User Interface.
- **DB**: Data Base.

## ABBREVIATIONS

- **[Gn]**: nth goal.
- **[Dn]**: nth domain assumption.
- **[Rn]**: nth functional requirement.

## REFERENCE DOCUMENTS

• This document refers to the specification document: Mandatory Project Assignments.pdf - Assignements AA 2017-2018

• This document follows the IEEE Standard 830-1998 for the format of Software Requirements specifications.

## DOCUMENT STRUCTURE

This document is structured in four parts:

**Chapter 1**: *Introduction*. It provides an overall description of the system scope and purpose, together with some information on this document.

**Chapter 2**: *Overall description*. Provides a broad perspective over the principal system features, constraints, and assumptions about the users and the environment.

**Chapter 3**: *Specific requirements*. Goes into detail about functional and nonfunctional requirements. This chapter is arranged by feature.

**Chapter 4**: *Formal Analysis using Alloy*.

# OVERALL DESCRIPTION

## PRODUCT PERSPECTIVE

The system will be developed from the ground up. In order to achieve the goals listed above, the system will make use with external services to retrieve data and information about the world phenomena, such as Car-sharing or Bike-sharing availability, traffic conditions, public transport options, etc. Our System will be able to manage the information about the user, calculate the best solution for him on the basis of his preferences and possession. Moreover it will have to take into account the trip arrangement.

A diagram that shows an high level vision of the system is provided in Figure 1.

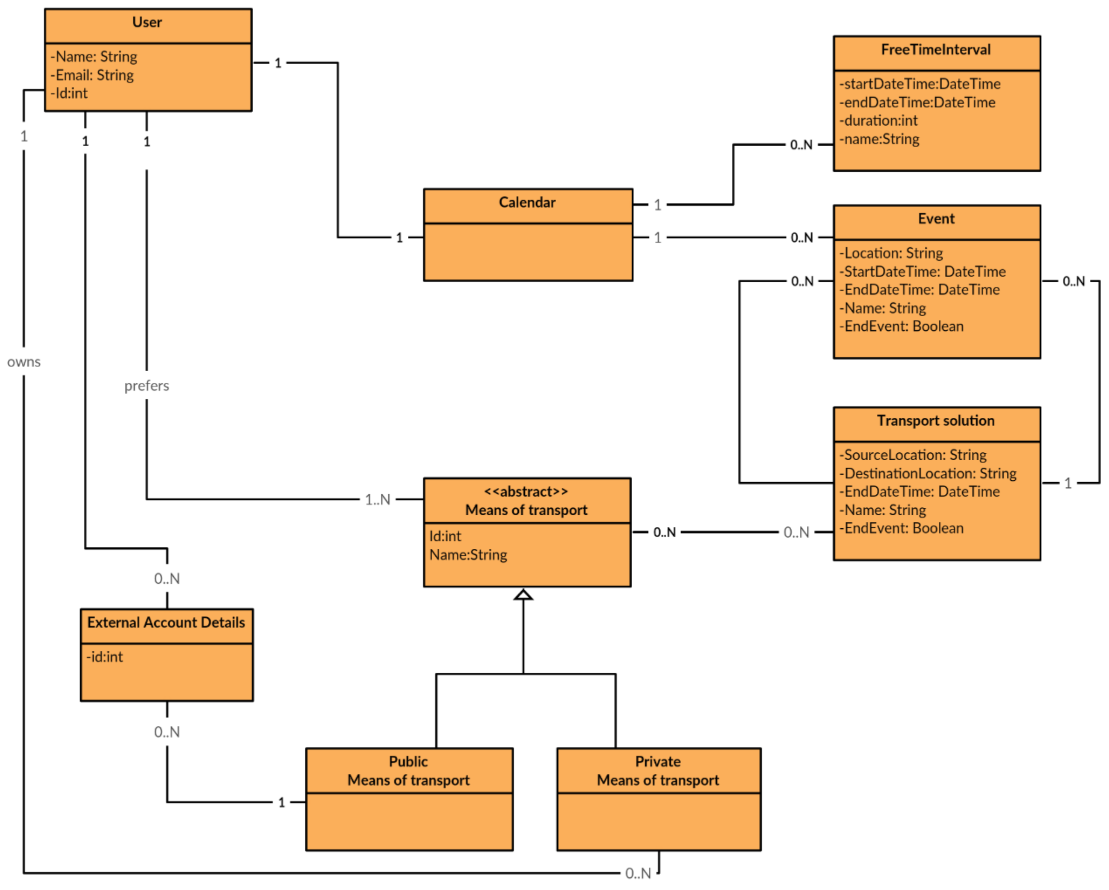


*Figure 1: UML Class Diagram*

## PRODUCT FUNCTIONS

In the direction of defining unambiguous boundaries for the system in development, a specific list of the functions that the product will offer is provided and precisely explained hereunder.

- The system allow a *visitor* to became a *user* after the registration process.
- All *user* can:

  1. Insert and manage events in the calendar, included the possibility of modifying and deleting them.
  2. Configure the profile settings, entering means of transport that he owns and his preferences about the travel arrangements.
  3. Define flexible breaks.
  4. Identify the best mobility option for a transfer calculated on his preference, select it and let the system suggest about reservations and purchases of tickets.
  5. Receive notification about the events overlapping and when an event are going to be unreachable (the transport solutions are running out).

## USER CHARACTERISTICS

- Visitor: a person using Travlendar+ without being registered. The only thing he can do is proceeding with registration.
- Registered User: a person that has completed a successful registration process and now he is able to use all the Travlendar+ services. He/she can login to the system and, after that, use all the functionalities.

## ASSUMPTIONS AND DEPENDENCIES

In the specification document  we found some points that lacked in precision and would have led to some ambiguities. In order to better clarify those situations we decided to introduce the following assumptions.

We assume that:

- *Credentials* that a visitor had to provide to become a registered user are only e-mail and password.

- Every *transport solution* shall be calculated between two subsequent events or from home to an events. To grant usage flexibility, we provide the user to manage his actual position, referred to an exact instant, dynamically during the experience and in this case system will consider that position as a *start position*.

- The *flow of events* always starts from home, that is a location provided by the user during the configuration of his profile. We identify start and ending of a *flow* with a flag on a event that tell the system that this is the last event before coming home, the subsequent event will start from home. If home has not been defined system will tell the user to redefine his position with the *specific function*.

- User will receive a *notification for an unreachable* event if there's no solution to reach it or solution founded does not grant to reach the event in time.

- User will be notified in case he insert two or more *overlapped events*.

- A *flexible break* is intended like a special event defined by a time interval and a minimum duration. For example an user could insert his breakfast like a flexible event with a minimum duration of ten minutes and choose to spend that in the interval from 9 a.m. and 10 a.m..

- The *priority of the system is to manage events*. In case a set of events prevents to spend the flexible break, the user will be notified, anyhow the events will be added to the calendar.

- To identify *the mobility options* the system computes a list of available means of transport on the basis of the user preferences and settings and their availability, then makes up the solutions combining them. If the user confirms the choice of a specific solution, the system suggests the purchases of ticket/reservations to be made to purse it.

- The user must specify his *mobility preferences* (carbon footprint, travel time, maximum distance on walking, etc.), specify his private means if he wants the system to consider them and his subscription to public and sharing services.

- About *trip arrangement*:

    - The system will redirects the user on the external service in case that the specific service does not provide a complete API to accomplish a purchase ( public transport tickets) or a reservation (bike/car sharing services)

    - The user must provide every needed information to complete a purchase operation, the system just delegate a third party the operation concerning payment and mean reservation.

### DOMAIN ASSUMPTIONS

- **[D1]** All user have access to a stable Internet connection with their smartphone.
- **[D2]** The mobile phones of the users are provided of a GPS module.
- **[D3]** The user specifies the correct location, if its GPS is not available or mean to be used.
- **[D4]** Payment information and process is operated and guaranteed by third party.
- **[D5]** External API's are available and provide consistent information.
- **[D6]** The events exist.
- **[D7]** The user will attend events that he inserts in his calendar or inform the system coherently with his real behavior.
- **[D8]** The user will respect the selected solution previously proposed by the system, otherwise the user will notify the system inserting his real position.

## FUTURE EXTENSION

The system will be implemented foreseeing the possibility of further extensions, for example:

- Develop of a web app to extend target area and give no-android users the possibility to use this service.
- Extend preferences to make the user experience closer to reality for example the introduction of a companion or a family travel.
- Introduce the possibility to analyze user behavior and model it to suggest future events and improve user experience.
- Provide the system of a new functionality concerning the displacement of flexible events (e.g. shopping) to a better optimization of the event flow. For instance, if an user has to do a commission located near an event previously added on the calendar, the system should suggest to relocate the commission in a time closer to the fixed appointment.

# SPECIFIC REQUIRMENTS

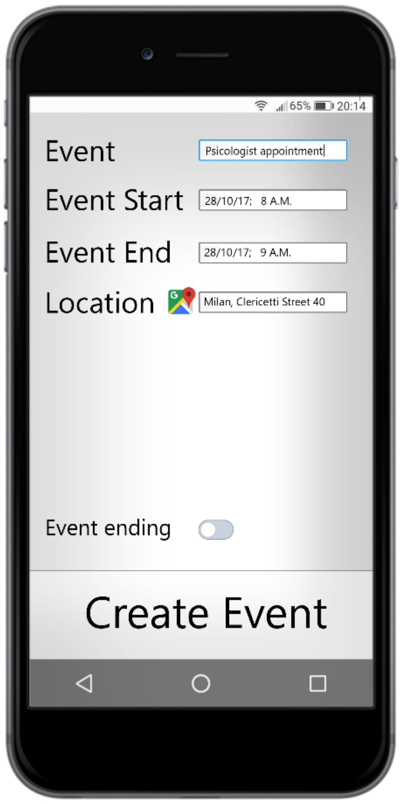
## EXTERNAL INTERFACE REQUIREMENTS

### USER INTERFACES

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

The core functionalities of the application is based on the events inserted by the user that will be managed inside the calendar. In *Figure 2* is represented the adding on a new event: the user inserts the name of the appointment, the start date-time and end date-time information, provides a valid location and chooses whether mark the event as “ending-event”. After that it could press the Create Event Button to confirm the creation.

One possible scenario after that operation is shown in *Figure 3,* if the mandatory fields were correctly filled the user receives a notification concerning the successful creation.



65% 20:14

Event: Psicologist appointment

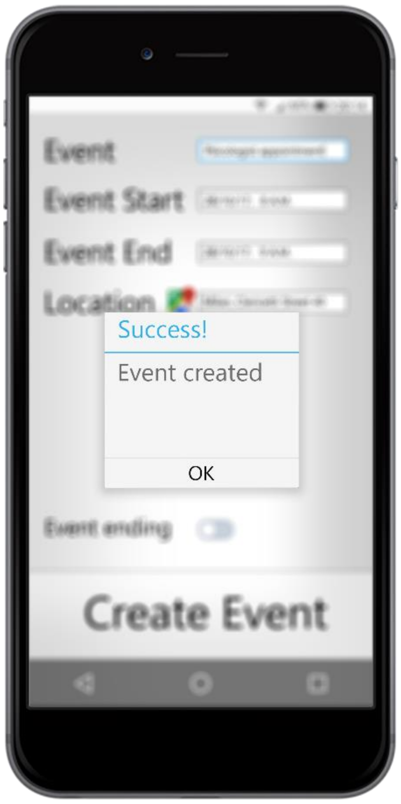
Event Start: 28/10/17; 8 A.M.

Event End: 28/10/17; 9 A.M.

Location: Milan, Clericetti Street 40

Event ending

*Figure 2: Creation of an event*



Success!

Event created

*Figure 3: Notification about successful creation*

Another possible notification that user could receive is relative to the events overlapping, it is shown in *Figure 4*. The user can consults the details of the overlap situation or cancel directly the last event added.

In *Figure 5* is shown a daily view of the calendar, there are two events inserted. The user could delete the events pressing the "X button" or be redirect to the modification form pressing the "pencil button". If The user click the "Bus button" the transport solution are shown.

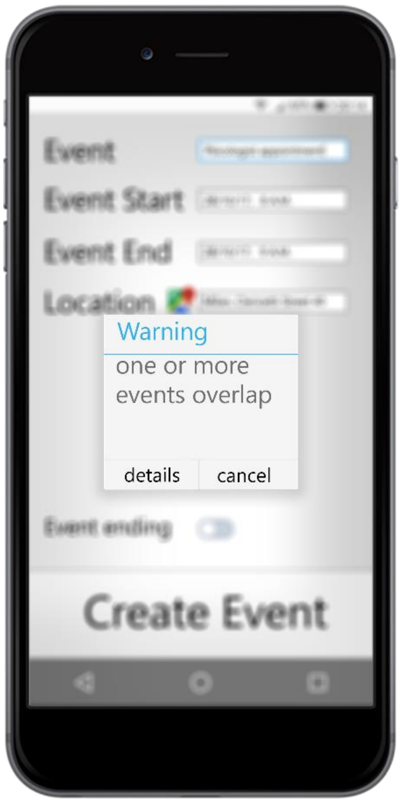


*Figure 4: Notification about successful creation*

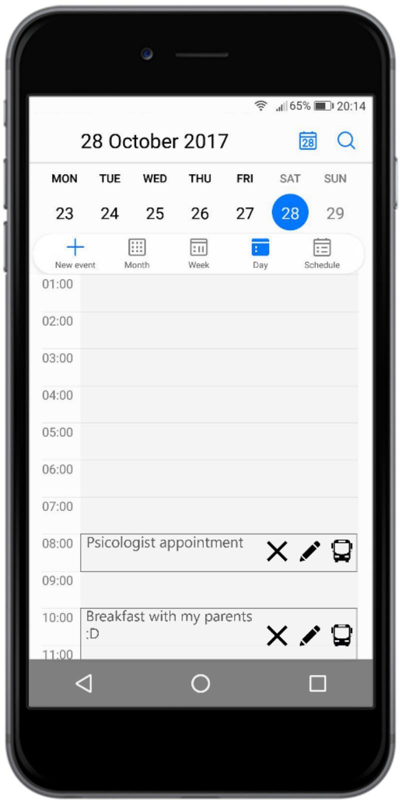


*Figure 5: Daily view*

In the *Figure 6* are shown the possible transport solutions between the two events calculated by the system personalized for the logged user.
Under the black arrow we can  identify the best solution offered. In the example the best solution is the public transport, which means that the logged user previously added this options like the preferred rather than a "green solutions" or "faster solutions".

In the *Figure 7* are shown the selected solution details, the user could purchase the ticket pressing on the ATM Logo.

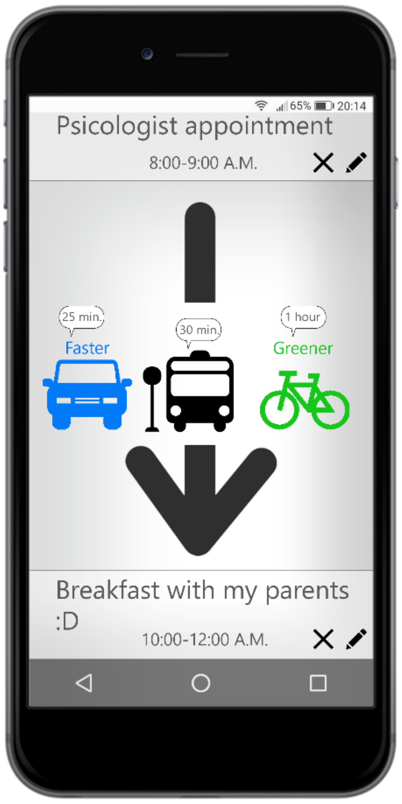


*Figure 6: Transport solutions*

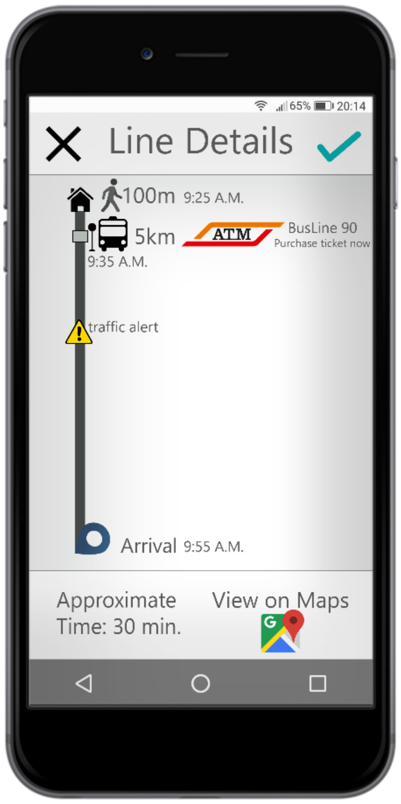


*Figure 7: Transport solution details*

## HARDWARE INTERFACES

- The client app must be able to access the GPS data of the user's phone. It has to deal with security dialogs, authorization and platform-specific standards.

- Mobile Data module such as (2G,3G,4G).

## SOFTWARE INTERFACES

The mobile phone must support Android.

In the first release we don't provide any type of API to allow external service to interact with our system.

## COMMUNICATION INTERFACES

The clients communicate with the server via HTTPS requests.

## FUNCTIONAL REQUIREMENTS

In order to fulfill the goals listed before, the following requirements can be derived. The requirements are grouped under each goal from which it is derived.

**[G1]** Allow a visitor to became a registered user.

- [R1] The visitor must be able to provide an email address and a password.
- [R2] If the visitor insert an email already used the system must be able to reject the registration.
- [R3] If the user insert an email address formally incorrect the system must be able to reject the registration.
- [R4] If the user insert an email address already used the system must be able to reject the registration.

**[G2]** Allow user to login to application.

- [R1] The user must be able to register.
- [R2] The system must be able to reject invalid user credentials.

**[G3]** Allow user to create a new event in the calendar.

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to complete mandatory fields of the creation form:
  - -Location (Address);
  - -Time interval (DateTime of beginning and DateTime of ending of event);
- [R3] The system must be able to reject invalid or inconsistent information.

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

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to create events.
- [R3] The system must reject the modification if the mandatory fields are not consistent or not filled.

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

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to create events.

**[G6]** Allow user to consult the transport solutions between events in the calendar proposed by the system.

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to create and modifying events.
- [R3] The system must have access to personal settings of the user.
- [R4] The user must be able to modify his personal settings.
- [R5] The system computes the solution on the basis of user preferences and the data retrieved from the external services.

**[G7]** Allow user to re-define dynamically his instant position to re-plan the transport solution.

- [R1] The user must be able to log-in to the application.
- [R2] The system must check the validity and consistency of the address provided.
- [R3] The system must ask the user how to insert the position.
- [R4] The system must be able to retrieve data from GPS module.

**[G8]** Allow user to set free time (break) during the day schedule.

- [R1] The user must be able to log-in to the application.
- [R2] The User must be able to complete mandatory fields :
    - DateTime interval.
    - Minimum time duration.
- [R3] The system must be able to reject invalid or inconsistent information.

**[G9]** Notifying events overlapping.

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to create event.
- [R3] The system must recognize when two or more events overlap.
- [R4] The system must be able to send notification.

**[G10]** Notifying the time-unreachable event.

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to create event.
- [R3] The system must recognize if there aren't transport solution between two events.
- [R4] The system must be able to send notification.

**[G11]** Allow user to configure transport preferences and external services to be used

- [R1] The user must be able to log-in to the application.
- [R2] The User must be able to provide account of external service.
- [R3] The User must be able to specify his preferences.
- [R4] The User must be able to register his personal means of transport.
- [R5] The User must be able to register his personal subscriptions to external service.

**[G12**] Allow user to set an event as ending event.

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to create event.
- [R2] The user must be able to mark an event like an "ending event".

**[G13]** Allow user to buy a ticket or to reserve a mean of transport of a suggested solution.

- [R1] The user must be able to log-in to the application.
- [R2] The user must be able to consult a suggested solution.
- [R3] The user must be able to provide valid payment method.

## SCENARIOS

Here some possible scenarios of usage of this application.

**Scenario 1**
Walter is very busy and he needs something that could help him to plan and optimize his days, so he wants to became a new user of Travlendar+. He downloads the App, and open it. He inserts his data in the registration form shown, then he submits the request to the system pressing the confirm button. If the registration is successful Walter will be redirected to the main activity, otherwise an error message will be displayed.

**Scenario 2**
Jesse, after being suggested by Walter to organize their cooking schedule, have already sign up to Travlendar+, have the App installed in his smartphone and want log in to use the system. After he is connected to the Internet, He opens the application and chooses "Access", enters the correct credentials and confirms. After the first Login the main activity Page loaded but he wants to configure his preferences and travel services so he touches the settings button.
He sets his home address, registers his traveler service account such as Enjoy and MoBike, his personal car and bicycle. Now he's ready to use the full functionality of the App!

**Scenario 3**
It's Monday and Skyler has been invited by her sister Marie for a dinner.
She wants to schedule her appointment in her Calendar. She logs in the application and clicks on the create event Button then She inserts the information about the event (when,where ecc..), ticks the "ending event option", because is the last event of the day, and presses the confirm button. The system rejects the new event and shows to her the message that the address is incorrect. Skyler reinserts the address correctly and presses the button to confirm. The system saves his event in his calendar.

**Scenario 4**
Junior has been noticed by his driving school that Tuesday lesson is canceled so he opens Travelendar's app and log in, then on the main activity he selects next Tuesday and erases the event about his driving lesson by choosing "Delete" and "Confirm".

**Scenario 5**
Detective Schrader is notified that next Monday meeting of his task force is not in Leonardo but in Bovisa, so he opens the Travelendar+ app and logs in, then on the main activity selects next Monday and opens the meeting event and choose "Modify". On the modification form he finds the "location" field and he deletes "Milano, Piazza Leonardo", types "Milano, Bovisa" and confirms.

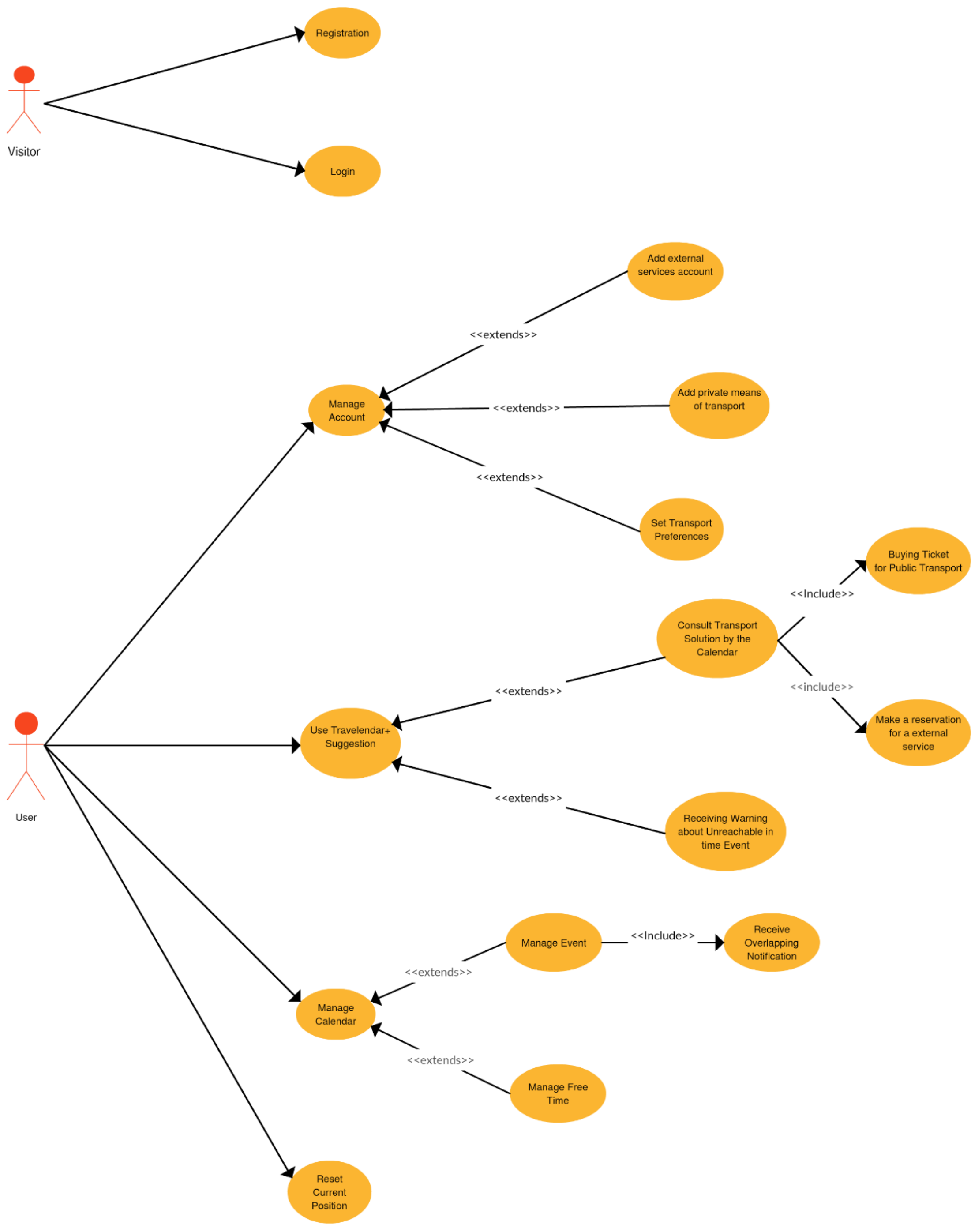
**Scenario 6**
Walter got confident with Travlendar+ and he filled his calendar with a lot of appointments this week. In Monday, in the morning, he has to teach a Chemistry lesson in the High school, so he goes to work with the public transport. At 12 o'clock he finishes the lesson and is preparing to eat his meal. In the days before Walter, that is a fan of fried chicken, made a reservation at 13 o'clock into a new restaurant (not quite close to the school) called "Los pollos Hermanos" and added that event in his calendar.

Quickly at 12.15 he logs in Travlendar+ , expands the daily planning, clicks on the "show transport solution" button and choose the better solution to reach the restaurant that fit with his preferences. During the Lunch Walter meet his brother-in-law who invites him to his home for tasting a new beer during the next saturday afternoon. Walter accepts this great invitation, suddenly he adds this fantastic event in his calendar but the system promptly notifies him that this event overlaps with another appointment. Walter saves the event anyway, but informs his brother-in-law that he might have a more relevant appointment.

**Scenario 7**
Jesse is now quitting his daily job to have lunch and then go to the next appointment with Walt. Travlendar+ is planning his trip without considering lunch as a break between his busy hours so Jesse decides to use the free time functionality to set his break. He open his app and select current day and choose "add free time", tick the "recurrent" option and choose his other working days on a week, he selects the time interval and the minimum duration,then confirm. After lunch Jesse noticed that the app doesn't know his current position so he decides to insert his position manually to permit Travlendar+ to help him to reach next appointment in time. He opens the app, chooses "insert current position" and the app asks him if he wants to use a localization service or wants to write down manually the address. He chooses the second and insert the address. After this operation Jesse receive a Push Notification that warns him that he will be late and there's no way to reach in time the destination, Jesse touches the notification and the App suggests him the fastest way to reach the appointment Jesse decides to inform Walt about his late and quickly follows the app's instructions.

## USE CASE DIAGRAM

The Figure 8 gives an overview on the main actors involved in the system and the functionalities that they are able to execute.



Visitor
Registration
Login
User
Manage Account
Add external services account
Add private means of transport
Set Transport Preferences
<<extends>>
<<extends>>
<<extends>>
Use Travlendar+ Suggestion
Consult Transport Solution by the Calendar
Buying Ticket for Public Transport
Make a reservation for a external service
<<Include>>
<<include>>
<<extends>>
Receiving Warning about Unreachable in time Event
<<extends>>
Manage Calendar
Manage Event
<<Include>>
Receive Overlapping Notification
<<extends>>
Manage Free Time
<<extends>>
Reset Current Position

Figure 8: Use Case Diagram for Travlendar+

## USE CASE DESCRIPTION

We describe in a detailed way the use cases that we derived from the scenarios. It is important to understand that all the references to "activity", "buttons" or "forms" are only hypothesis to make the situation as clear as possible and to help the reader to draw a visual picture in his mind of what we plan to do, but the real structures will be well defined in the Design Document.

### *1-Registration of a new user*

| | |
|---|---|
| Actor | Visitor (Unregistered user) |
| Goal | G1 |
| Input Conditions | none |
| Event Flow | 1. The visitor compiles the registration form with the email address and password.<br><br>2. Visitors clicks on "confirm" button.<br><br>3. The application will save the date in the DB.<br><br>4. The application shows the main activity. |
| Output conditions | Visitor successfully ends registration process and become a User. From now on he/she can log in to the application using his/her credential and start using Travlendar+. |
| Exceptions | 1. The email address is already used.<br><br>2. The email address is formally incorrect.<br><br>3. The email address doesn't exist.<br><br>4. The password doesn't respect the standard.<br><br>All exceptions are handle alerting the visitor of the problem and application goes back to point 1 of event flow. |

*2-Login of an user*

| | |
|---|---|
| Actor | Registered user |
| Goal | G2 |
| Input Conditions | There is no instance of user logged in.<br><br>The user, already registered, choose to login. |
| Event Flow | 1. The user selects "login".<br><br>2. The login form is shown.<br><br>3. The user inserts username and password.<br><br>4. The user confirms. |
| Output conditions | The system tells the user that he has been successfully logged in and load the main activity. |
| Exceptions | In case of the credentials are not valid the user is notified by the system and the login page is shown: login has failed. |

*3-Creation of a new event*

| | |
|---|---|
| Actor | User |
| Goal | G3 |
| Input Conditions | The User is logged in |
| Event Flow | 1. The User arrives at main activity of the mobile application.<br>2. The User clicks on the Button "Add Event".<br>3. The User compiles the fields of the creation form.<br>4.The user confirms the creations.<br>5.The application inserts the event in the calendar. |

| | |
|---|---|
| Output conditions | The system tells the user that he has been successfully created a new event and reload the main activity.<br>If the event created overlap with another event (previously added in the calendar) the application notifies the user about this situation. |
| Exceptions | 1. The mandatory fields aren't filled.<br>2. The Address inserted doesn't exist.<br>3. DateTime of ending is before of the DateTime of beginning.<br><br>This exceptions (1-2-3) are handled alerting the visitor of the problem and application goes back to point 3 of event flow resetting the indicted field.<br><br>4. The user chooses to discard the system will ask to confirm and then return to the main activity. |

### *4-Modification of an event*

| | |
|---|---|
| Actor | User |
| Goal | G4 |
| Input Conditions | The user is logged in |
| Event Flow | 1. User selects a day from the main activity.<br><br>2. User selects an event from the day that has been chosen.<br><br>3. User chooses "modify".<br><br>4. User modifies the fields in the form.<br><br>5. User confirms the changes. |
| Output conditions | System notifies that the changes have been applied and re-calculate the transport suggestion. |

| | |
|---|---|
| Exceptions | 1. The user chooses to discard, the system will ask to confirm and then returns to the main activity<br><br>2. If a mandatory field is empty user cannot confirm the changes but only discard it, system will notify it.<br><br>3. If a mandatory field is filled incorrectly the system will notify it. |

### *5-Deletion of an event*

| | |
|---|---|
| Actor | User |
| Goal | G5 |
| Input Conditions | The user is logged in |
| Event Flow | 1. User selects a day from the main activity.<br><br>2. User selects an event from the day that has been chosen.<br><br>3. User presses the Button "Delete event".<br><br>4. User confirms the deletion. |
| Output conditions | System notifies that the deletion has been applied, re-calculate the transport suggestion. |
| Exceptions | - If user chooses to discard the system will ask to confirm and then return to the main activity |

*6-Consultation of the transport solutions*

| | |
|---|---|
| Actor | User |
| Goal | G6 |
| Input Conditions | The user is logged in |
| Event Flow | 1. The user selects a day from the main activity.<br><br>2. The app shows a list of events of the selected day.<br><br>3. User selects "show transport options"<br><br>4. The user could select a particular solution proposed. |
| Output conditions | Between the various events a list of possible transport option, calculated on the base of the user preferences, is shown.<br><br>If the user select a particular solution, then the application suggests the purchases of ticket/reservations to be made to purse the solution. |
| Exceptions | none |

*7-Setting an event as ending event*

| | |
|---|---|
| Actor | User |
| Goal | G3, G4, G12 |
| Input Conditions | The user is logged in.<br><br>The user is creating or modifying an event. |
| Event Flow | 1. The user ticks "ending event"<br><br>4. The user confirms. |
| Output conditions | The events is signed like the last of the series, it means that the trip to the subsequent event will be calculated from the default location (home) instead of the position of the event. |
| Exceptions | none |

*8-Setting a new flexible break*

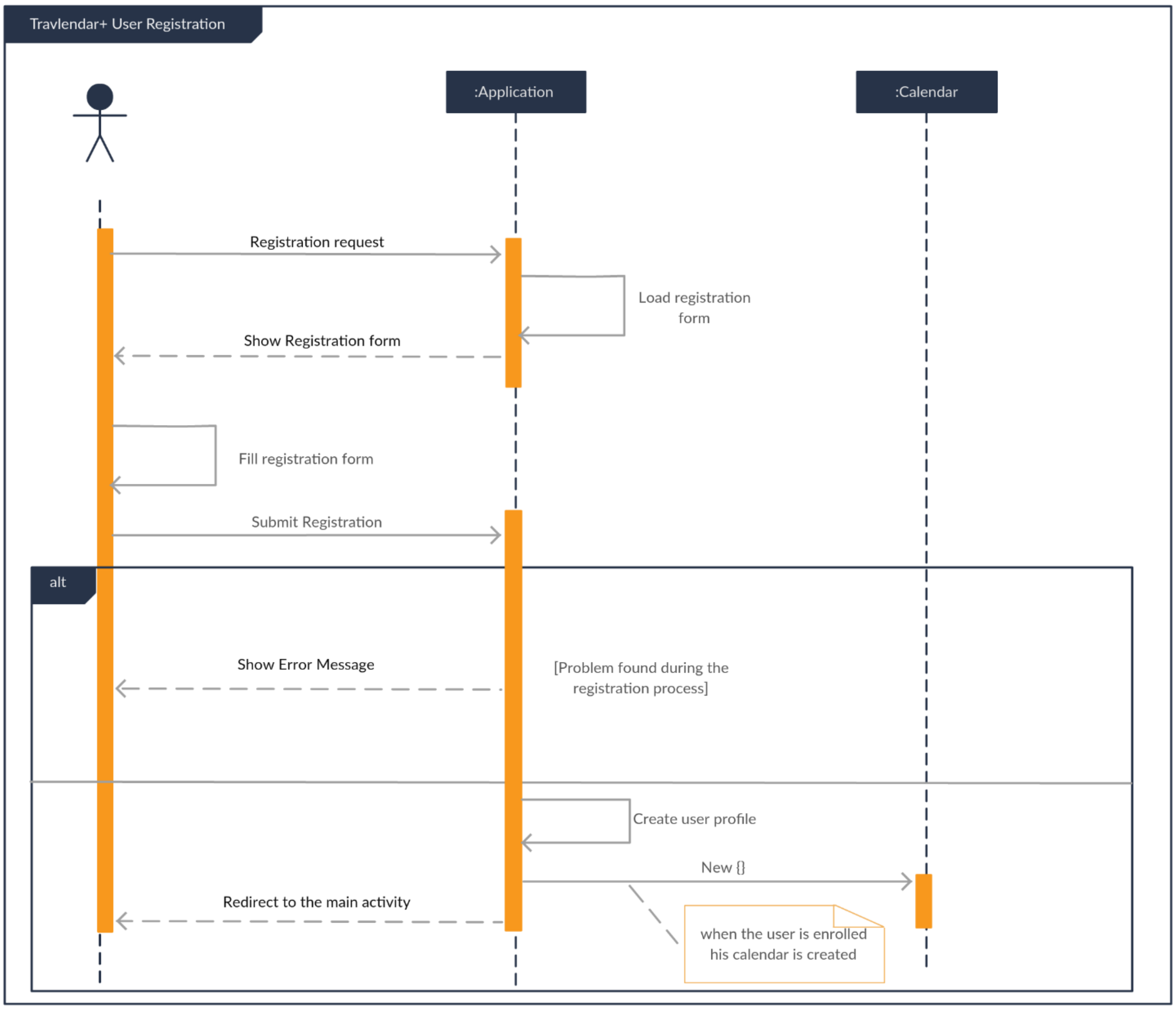
| | |
|---|---|
| Actor | User |
| Goal | G8 |
| Input Conditions | The user is logged in. |
| Event Flow | 1. The user select a day from the main activity.<br>2. The app shows a list of events of the selected day.<br>3. The user select "add free time"<br>4. The app load a creation form for the free time<br>5. The user fills the fields of the loaded form.<br>6. The user confirms. |
| Output conditions | The system tells the user that he has been successfully created a free time interval and reload the main activity. |
| Exceptions | If, at the moment of the creation, the flexible break cannot be spent due to events that overlap with the entire free time interval, the user is notified in the confirmation message. |

*9-Purchase tickets form external service.*

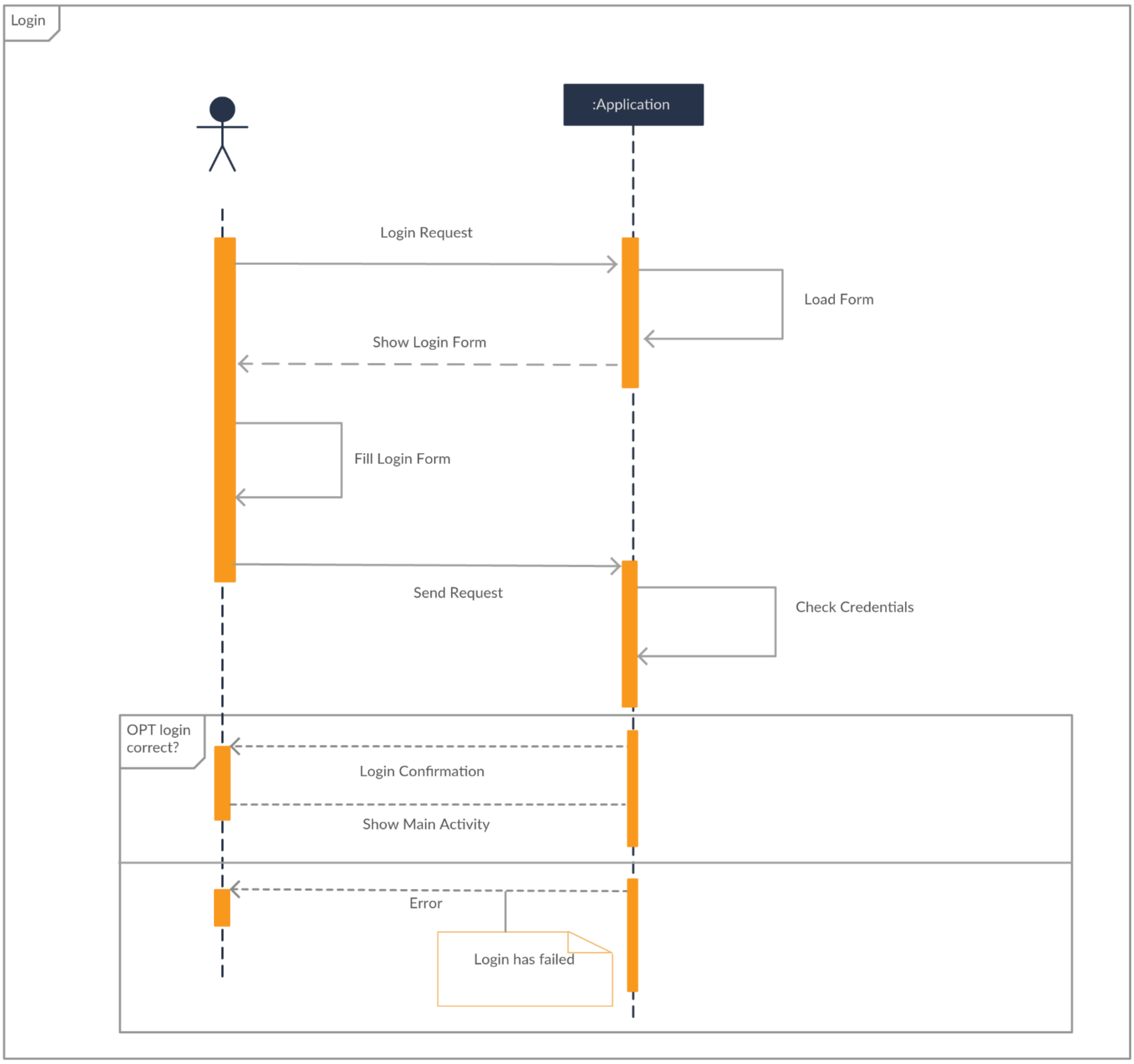
| | |
|---|---|
| Actor | User |
| Goal | G6 |
| Input Conditions | The user is logged in.<br><br>The user is consulting the transport solutions. |
| Event Flow | 1. The user select a solution with a services to be reserved/bought.<br>2. System shows the details of the solution chosen.<br>2. The user select "purchase this solution".<br>3. System redirects the user in the correct platform to complete the operation. |
| Output conditions | System sends a confirmation message. |
| Exceptions | If payment through external service fail the system notifies the user. |

## SEQUENCE DIAGRAMS

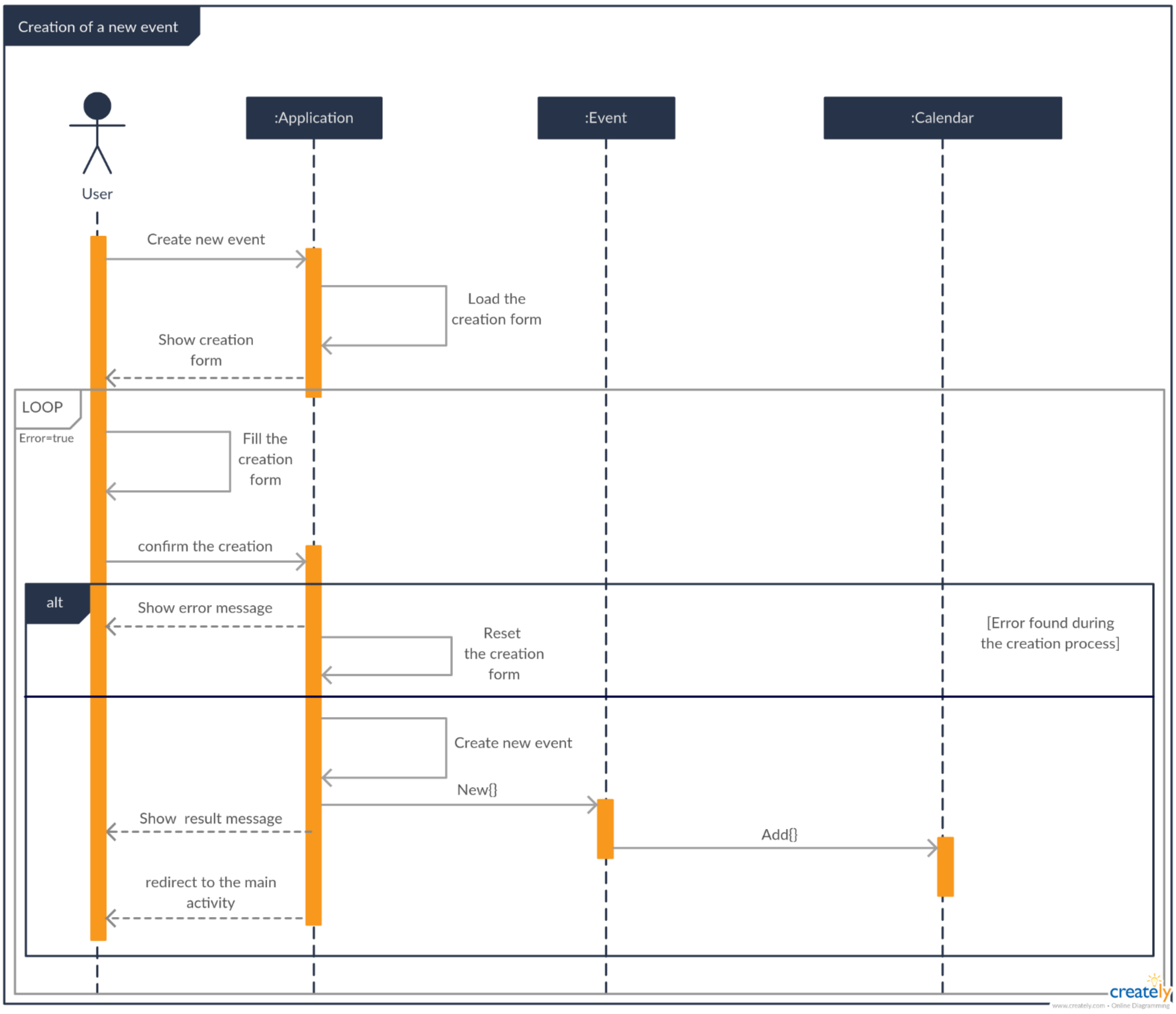
### *1-Registration of a new user*



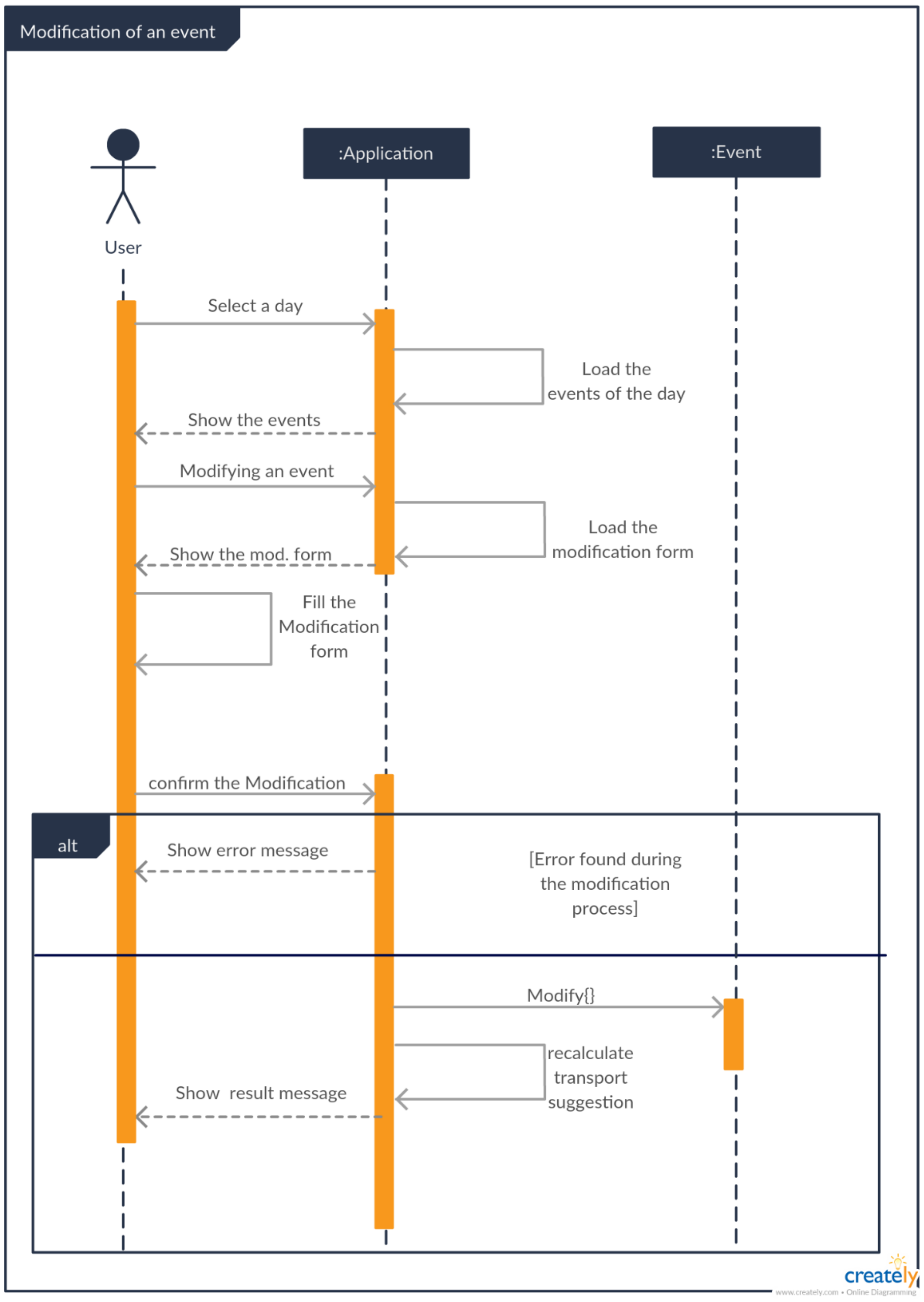
*2-Login of  an user*



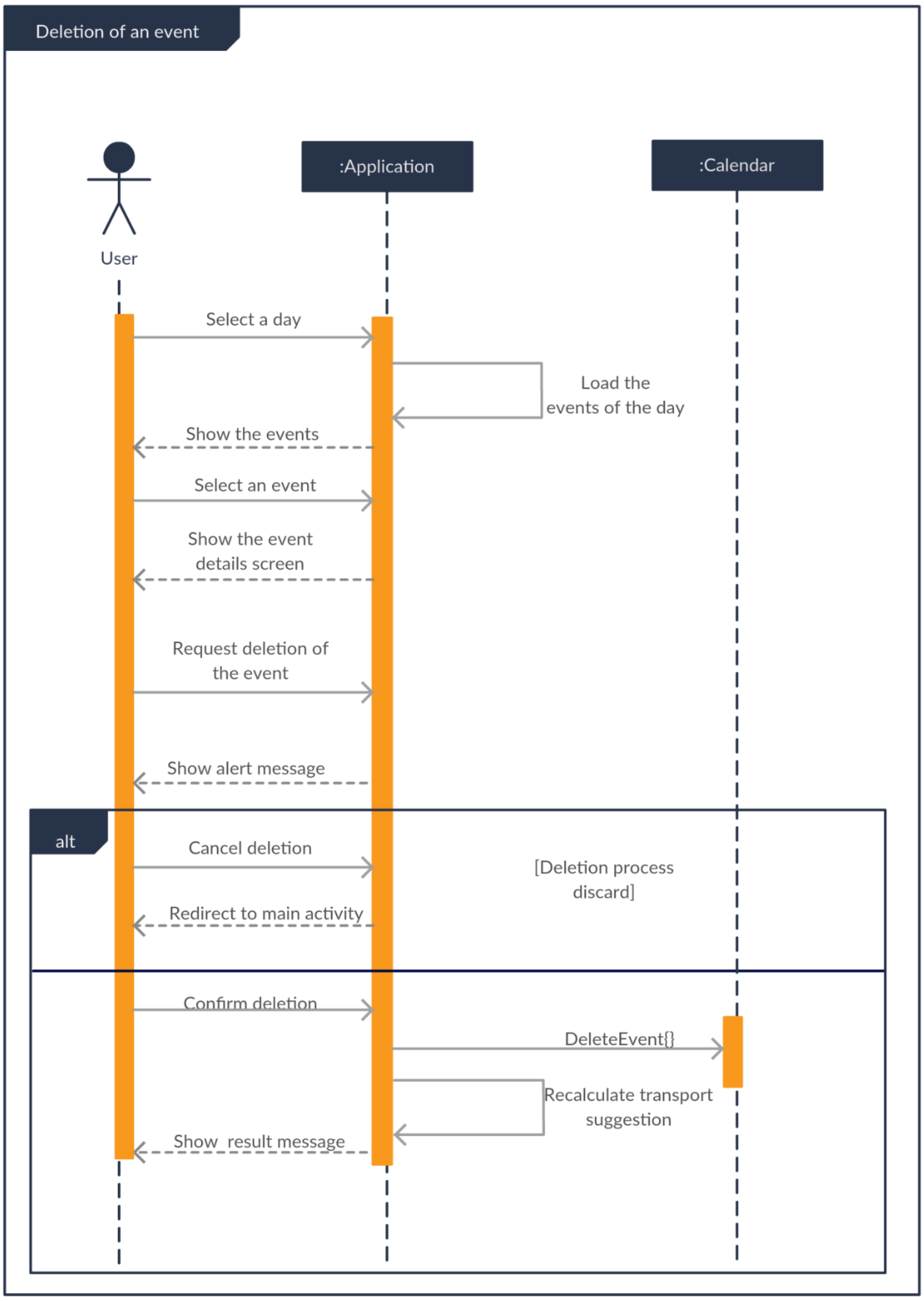
*3-Creation of a new event*



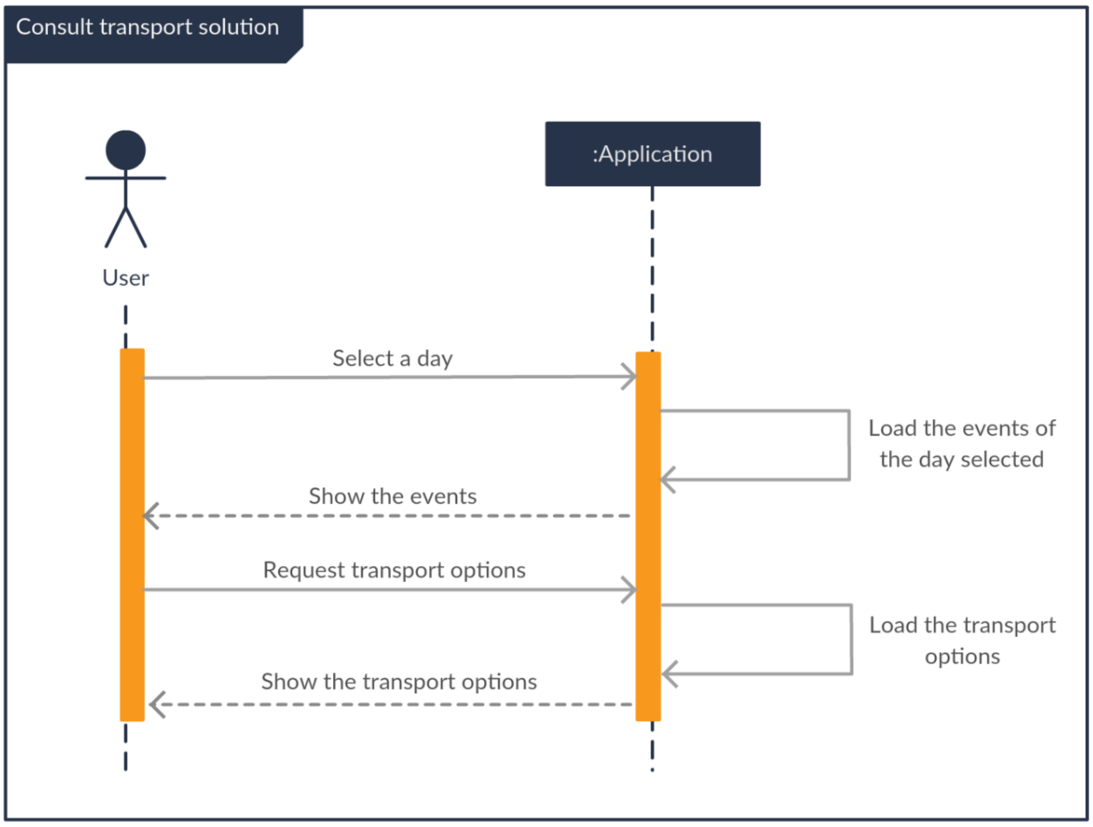
*4-Modification of  a new event*



*5-Deletion of an event*

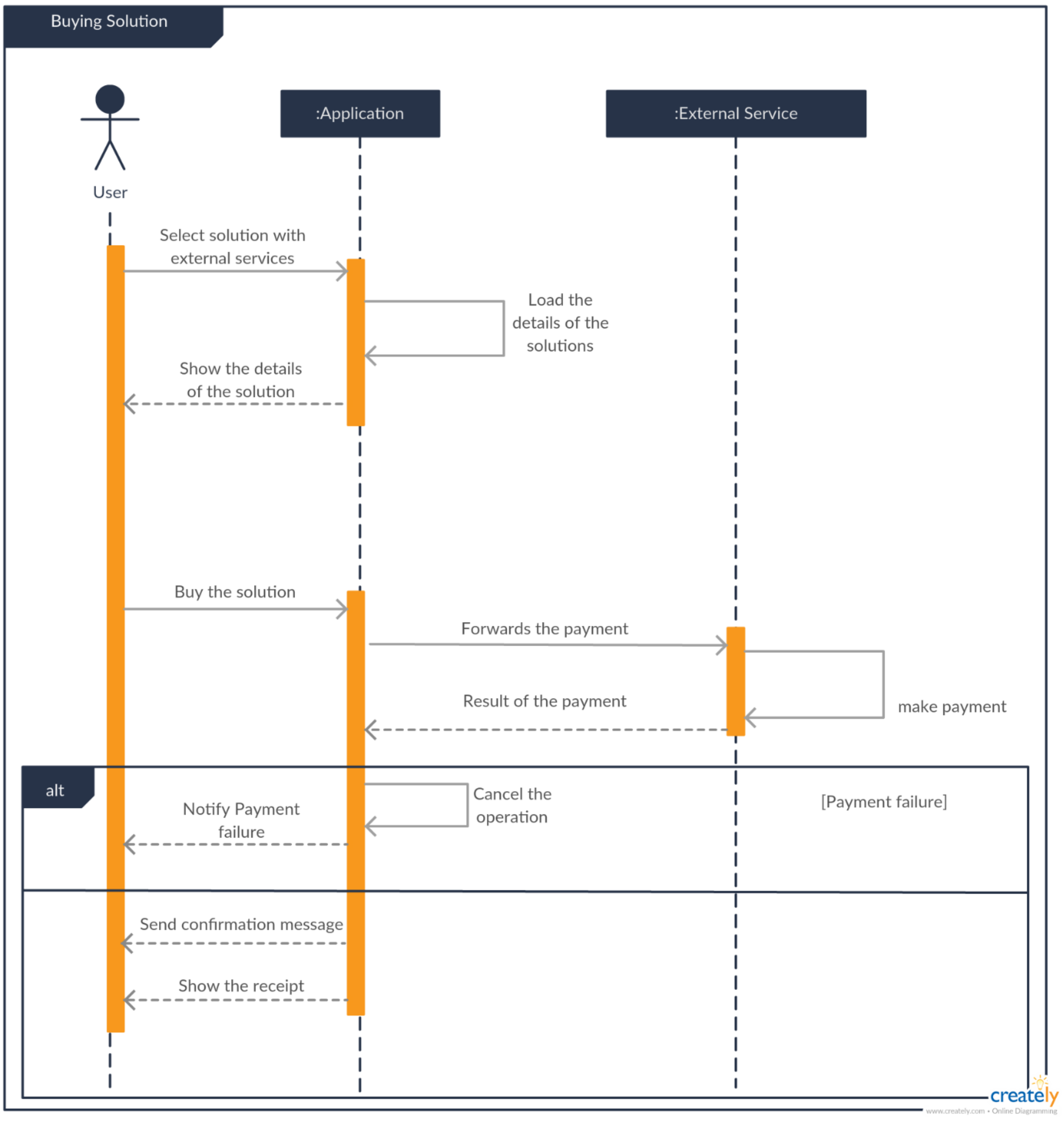


## *6-Consultatiton of  a transport solution*

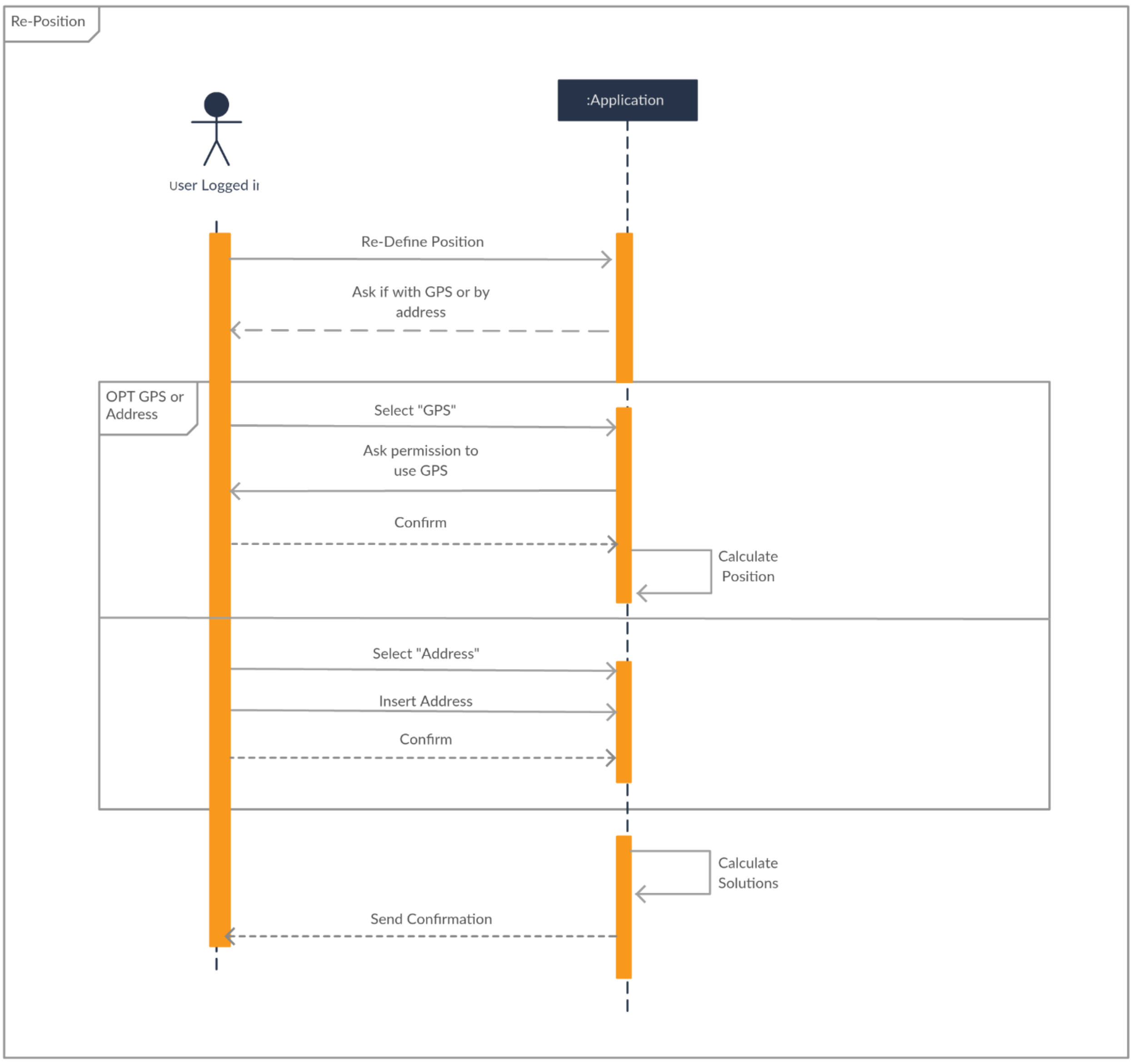

Consult transport solution
User
:Application
Select a day
Load the events of the day selected
Show the events
Request transport options
Load the transport options
Show the transport options

*for simplicity we indicate only the arrow "load the solution", obviously the process of retrieval information require the interaction with external systems as indicated above*

*7-Purchase of a transport solution*



*8- Reset the personal position and force the re-planning of  solutions*



## PERFORMANCE REQUIREMENT

To supply suitable service, the system has to be reactive and able to answer to a high number of also simultaneous requests. In particular the system has to be able to process the 95% of request in less than 5s and 99% of request in less than 10s.
There is no limit on the total number of registered users.

## DESIGN CONSTRAINTS

### STANDARDS COMPLIANCE

The applications follows the official Android's design guidelines.

### HARDWARE LIMITATIONS

The system has to run under the following worst-case conditions:

• App: – 2G connection – 100 MB of free space of internal storage.

## SOFTWARE SYSTEM ATTRIBUTES

### RELIABILITY

A mobile application with poor stability quickly leads to abandonment. Mobile application crashes and freezes are the leading types of problems encountered by users who experienced a problem with their mobile app. There are few popular approaches to avoid development crashes, such as surrounding your code with a Try/Catch blocks and implementing an exception handling strategy. Both methods possess potential temporary solutions but fail to eliminate the exception on a long-term basis.

There are several digital libraries for reporting application crashes. We use the Application Crash Report for Android (ACRA) that is a popular crash report library among Android users. It allows the application to catch and report an error and its details moments before the application crashes, even if a user doesn't proactively report the error. (https://github.com/ACRA/acra)

### AVAILABILITY

The calendar service has to be always accessible, so the system has to provide always an access to at least the manage profile/manage event in normal condition.The system must guarantee an availability of 98%.

## SECURITY

-Server side
The database security strategy focuses on proactively protecting data from internal and external attacks, curtailing data exposure to privileged and authorized IT users, and safeguarding all databases.
Our key to building our  successful database security strategy encompasses:

-Understanding what type of data needs to be protected, such as personal identification information (PII), credit card numbers, customer data, Social Security Numbers, and health information, etc.
-Understanding applicable regulatory compliance requirements, such as payment card industry (PCI)
-Performing an inventory of all databases
-Discovering and classifying databases based on the sensitivity of data.
-Establishing security policies for all databases in the environment, segregated across zones.
-Having actions for each category of policies and deploying them across databases, segregated across zones.
-Taking suitable security measures, such as data masking (redaction), monitoring, auditing,  encryption, and access control.
-Looking for an all-inclusive database security solution that can implement robust database  security at a low cost.
-Client side

In this side it's enough to follow the best practice suggested by Android developer guide, there are no important stored data into the physical device. All data is on the server side.

## MAINTAINABILITY

The code should be well documented using JavaDoc in order to enable other developers to easily understand and edit it. The development of the software will follow the object-oriented ModelView-Controller pattern and the principles of separation of concerns. The build from source code in the Version Control System will be completely automated, as well as unit testing execution.

## PORTABILITY

The target user is limited to Android device, that covers a great percentage in the market.

## FORMAL ANALYSIS USING ALLOY

### Consideration

In this paragraph we will show our model developed with the usage of Alloy language whose consistency has been verified by Alloy Analizer.
We have modelled our world observed in an instant specifying the various constraints that in every instant must be valid to obtain a detailed photograph of interesting scenarios in many combination of event and instance. Our alloy model consider generic uses preferences, events relationship, transport solution, the worst cases of event overlapping, unreachable event and notification generation about these kind of condition.

```
open util/boolean

sig Stringa{}

one sig System{
    currentTime: one Int,
    user: set User,
}{
    currentTime>0
}

sig User {
    id: one Int,
    email: one Stringa,
    name: one Stringa,
    calendar: one Calendar,
    homeAddress: lone Stringa,
    preferences: some MeanOfTransport,
    externalAccounts: set ExternalAccount,
    meansPossessed: set PrivateMeanOfTransport,
    currentPosition: lone PositionUpdate
}{
    id>0
}


sig Calendar {
    user: one User,
    timeOccupation: set TimeOccupation,
}

abstract sig TimeOccupation {
    endDateTime: one Int,
    startDateTime: one Int,
}{
    startDateTime > 0
}


sig FreeTime extends TimeOccupation {
    timeDuration: one Int,
}{
    endDateTime - startDateTime >= timeDuration and timeDuration > 0
}

sig Event extends TimeOccupation {
    location: one Stringa,
    end: one Bool
}
```

*Signatures*

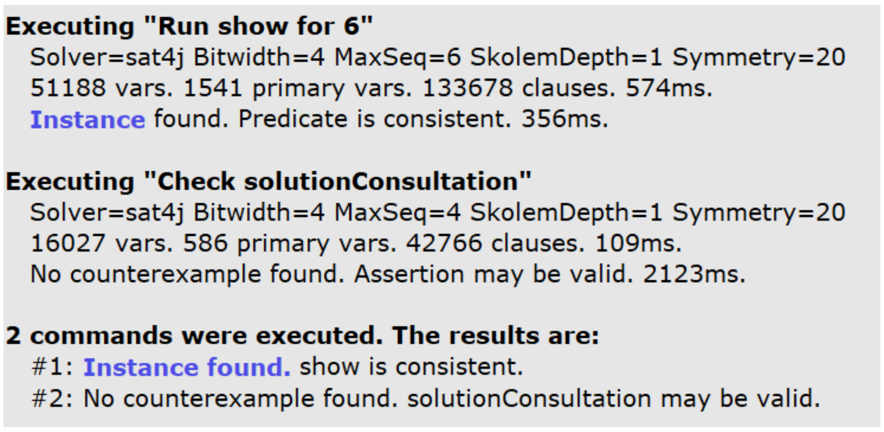
```
sig PositionUpdate {
   position: one Stringa,
   time: one Int
}{
   time > 0
}

sig TransportSolution {
   meansOfTransport: some MeanOfTransport,
   startLocation: one Stringa,
   startDateTime: one Int,
   timeDuration: one Int,
   startEvent: lone Event,
   endEvent: one Event
}{
   startDateTime > 0 and
      startDateTime = endEvent.startDateTime - timeDuration and
         timeDuration > 0
}

sig UnreachableNotification {
   event: one Event,
}

sig OverlappingNotification {
   time1: one Event,
   time2: one Event
   }

sig ExternalAccount {
   user: one User,
   meansOfTransport: one PublicMeanOfTransport,
}

abstract sig MeanOfTransport {}

sig PublicMeanOfTransport extends MeanOfTransport {}

sig PrivateMeanOfTransport extends MeanOfTransport {}



fact SystemKnowledge {
   all u : User, s : System | (u in s.user)
}
```

*Signatures*

```
fact SystemKnowledge {
    all u : User, s : System | (u in s.user)
}

fact uniqueUser {
    all u: User | (u.id >0)
    and
    no u1, u2: User | (u1 != u2 and
        (u1.email = u2.email or u1.id = u2.id))
}

fact uniqueCalendarForUser {
    no c1, c2: Calendar | (c1 != c2 and
        c1.user = c2.user)
}

fact calendarExistance {
    all  c: Calendar, u: User | (c.user = u <=> u.calendar = c )
}

fact externalAccountAndPrivateMeanOfTransportExistence{
    all  eA: ExternalAccount | ( one u : User | (eA in u.externalAccounts))
    all pMOT: PrivateMeanOfTransport | (one u : User| (pMOT in u.meansPossessed))
}

fact timeOccupationValidity {
    all t: TimeOccupation | ( t.startDateTime < t.endDateTime and (one c:Calendar | t in c.timeOccupation))
}

/*** position update is considered valid before the happening of a subsequent event,
of course it makes sense only if it's time of insertion is not future ***/
fact positionUpdateValidity{
    all u : User,  s: System | ( #u.currentPosition > 0 implies ( s.currentTime > u.currentPosition.time and
        (no e: Event | e in u.calendar.timeOccupation and e.startDateTime > u.currentPosition.time and e.startDateTime < s.currentTime)))
    all p : PositionUpdate | (one u : User | (u.currentPosition = p))
}

/***transport solution must be calculated or must be send a notification for an unreachable event***/
fact transportSolutionOrNotification {
    all e: Event | ( !inTimeReachable[e] <=> one n: UnreachableNotification | (n.event = e))
    all n: UnreachableNotification | (no  ts : TransportSolution | (n.event = ts.endEvent) )
}


/***transport solution is valid only if is between two consequent event (with overlapping exception) or from Home
 in case there is overlapping the solution is calculated between the  first event before the overlapping events
 and between the overlapped events***/
fact transportSolutionStarting {
    all ts : TransportSolution, u : User | ( #ts.startEvent > 0 implies
        (ts.startDateTime > ts.startEvent.endDateTime and ts.startEvent.startDateTime < ts.endEvent.startDateTime
            and no t: TimeOccupation | (t.startDateTime < ts.endEvent.startDateTime and t.endDateTime < ts.endEvent.startDateTime )
            and ts.startLocation = ts.startEvent.location and ts.endEvent in nextEvent[ts.startEvent] and ts.startEvent.end = False )
        and #ts.startEvent = 0 implies (
            (#u.currentPosition = 0 implies (ts.startLocation = u.homeAddress)) and
                (#u.currentPosition = 1 implies (ts.startLocation = u.currentPosition.position)) and (one e : Event | (ts.startEvent in nextEvent[e] and e.end = True))))
}

fact transportSolutionMustBeReferredToAUniqueUser {
    all ts : TransportSolution, u : User | ( #ts.startEvent > 0 implies
        (eventOfAUser[u, ts.startEvent] and eventOfAUser[u, ts.endEvent]))
}

fact respectPreferences {
    all ts : TransportSolution, e : Event, u: User | ( (eventOfAUser[u, e] and ts.endEvent = e) implies
        ( ts.meansOfTransport in u.preferences))
}

fact warningNotificationOfOverlapping{
    all disj e1, e2: Event | ( (overlapping[e1, e2]) implies
        (one n: OverlappingNotification | (n.time1 = e1 and n.time2 = e2)))
    no n : OverlappingNotification | !(overlapping[n.time1, n.time2])
}
```

*Facts*

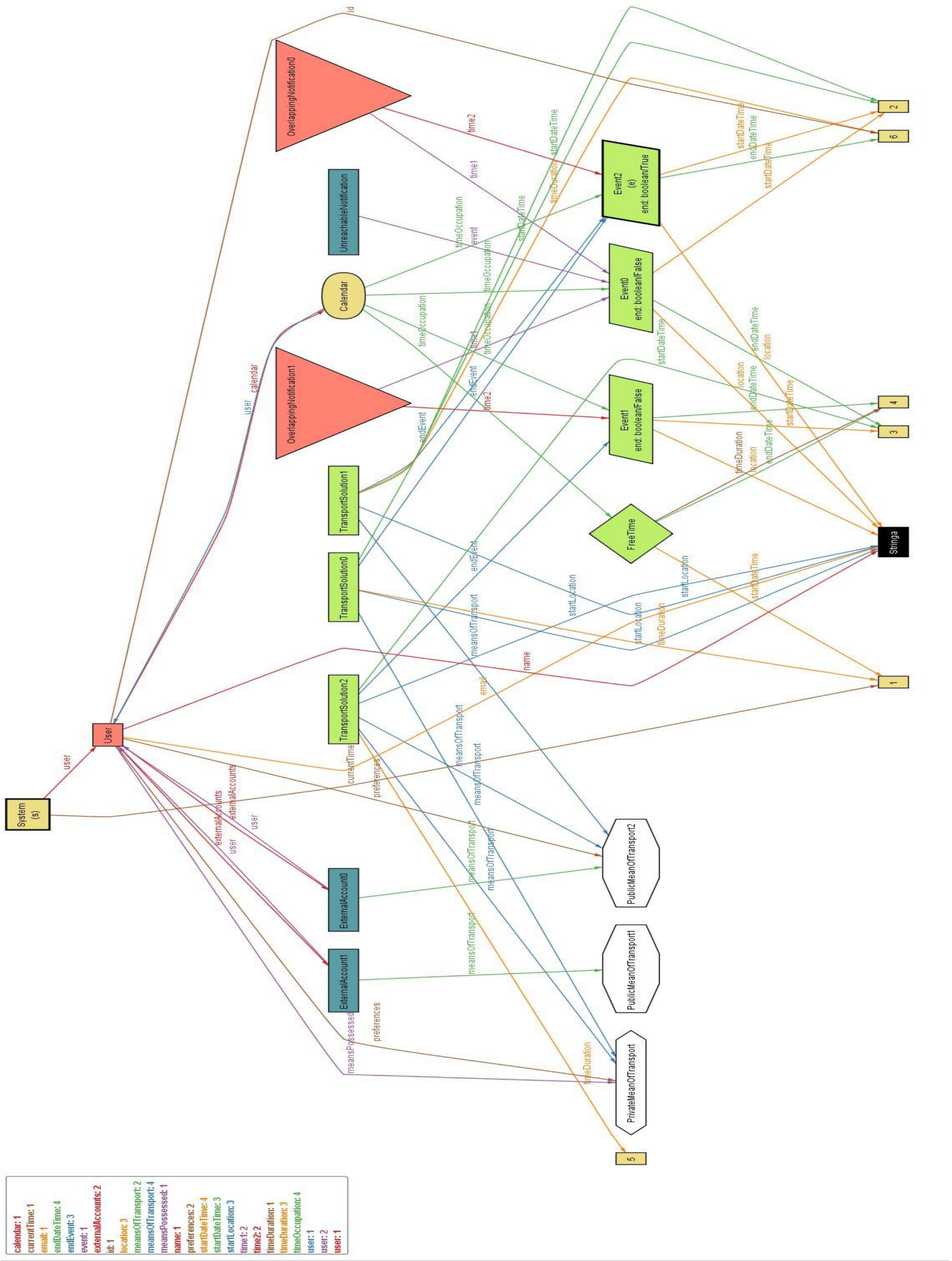
```
/***ASSERTION
solutions proposed to user are consistent
   - if there's an event there's a solution to reach it in time, if the solution is missing user is notified
   - if two events overlap, system will notify it
***/
assert solutionConsultation{
   all e1, e2 : Event | ( (overlapping[e1, e2] <=> one n : OverlappingNotification | (n.time1 = e1 and n.time2 = e2)))
   all e : Event | (  (!inTimeReachable[e] => (one n : UnreachableNotification | n.event = e)))
}

/*** Predicate ***/

pred inTimeReachable[e : Event] {
   some ts: TransportSolution | (ts.endEvent = e and (ts.startDateTime + ts.timeDuration <= ts.endEvent.startDateTime))
}

/***The event e1 is in the calendar c1 of user u1***/
pred eventOfAUser[u1 : User, e1: Event] {
   all c1 : Calendar | ((c1.user = u1) => ( e1 in c1.timeOccupation ))
}

pred overlapping[ t1, t2: TimeOccupation] {
   one c1 : Calendar | (t1 != t2 and (t1 in c1.timeOccupation) and (t2 in c1.timeOccupation)
      and (( t1.startDateTime <= t2.startDateTime and t1.endDateTime >= t2.startDateTime and t1.endDateTime <= t2.endDateTime)))
}

/*** Function ***/

fun nextEvent [e : Event] : set Event {
   {e1 : Event | ((one c : Calendar | (e1 in c.timeOccupation and e in c.timeOccupation)) and e1.startDateTime > e.startDateTime and
      (all e2: Event| ((e2 != e1 and e2.startDateTime > e.startDateTime ) implies (e1.startDateTime <= e2.startDateTime))))}
}


/*** Generation of a world with focus on a specific user ***/
pred show() {
   #User = 1
   #TransportSolution >= #Event
   #FreeTime >= 1
   #Event < 4
   #PrivateMeanOfTransport >= 1
   #PublicMeanOfTransport >= 1
   #UnreachableNotification  = 1
   #OverlappingNotification  = 2

   // for graphic necessities we exclude in this simulation event that start and end at the same time
   all disj e1, e2 : Event | (e1.startDateTime != e2.startDateTime or e1.endDateTime != e2.endDateTime)
   some e : Event, s : System | (e.startDateTime > s.currentTime)
}

run show for 6
```

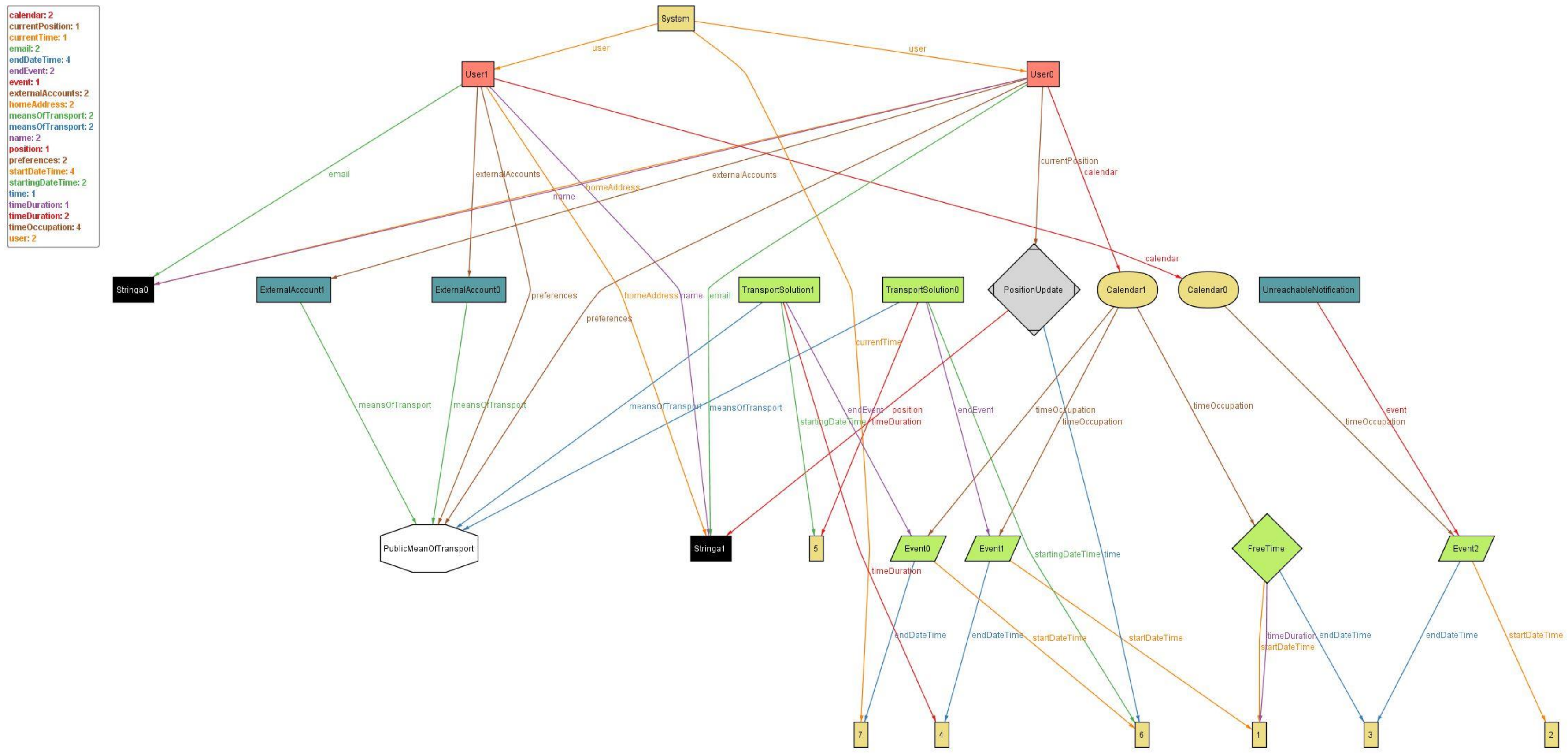


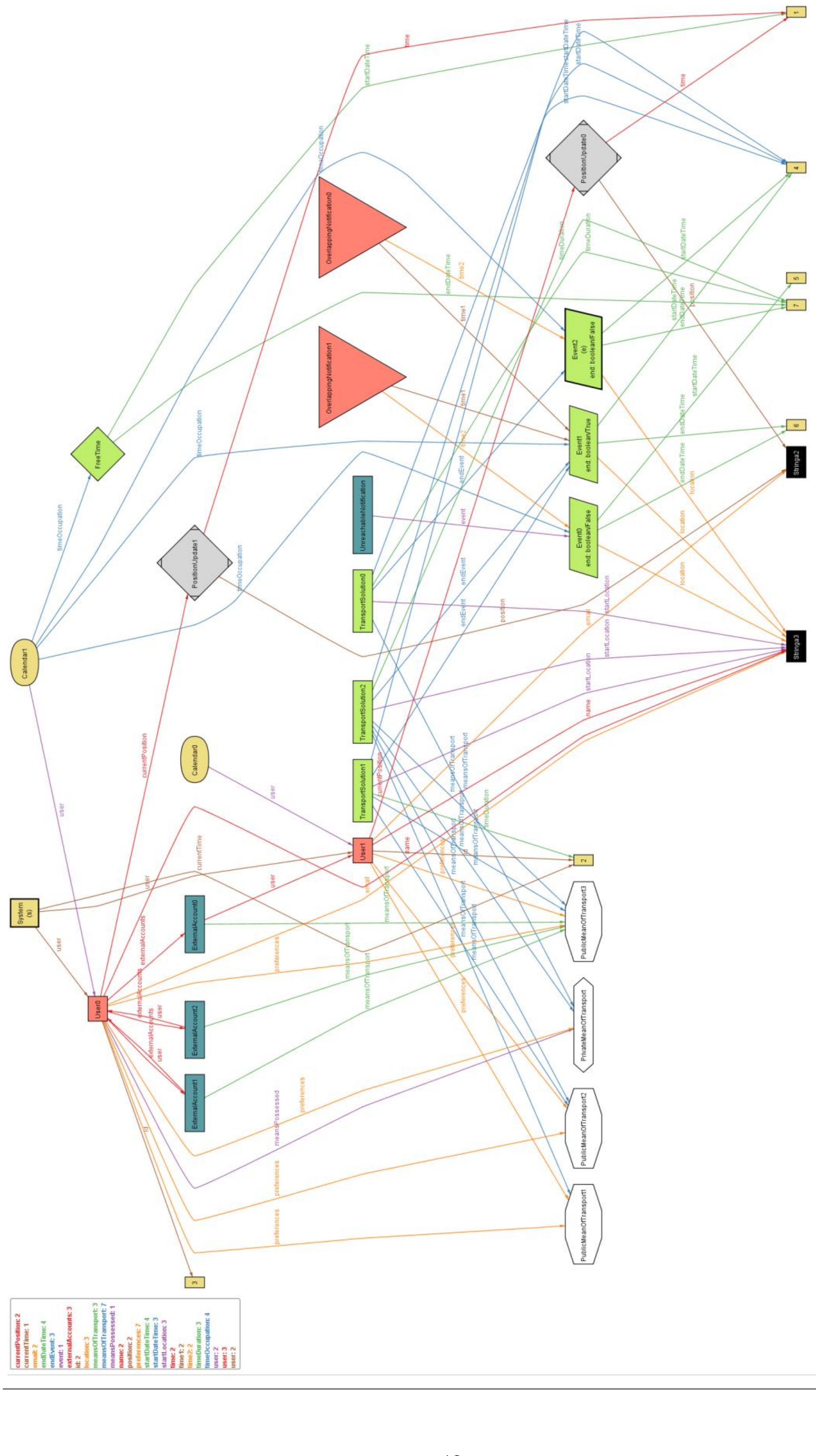
```
Executing "Run show for 6"
   Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20
   51188 vars. 1541 primary vars. 133678 clauses. 574ms.
   Instance found. Predicate is consistent. 356ms.

Executing "Check solutionConsultation"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   16027 vars. 586 primary vars. 42766 clauses. 109ms.
   No counterexample found. Assertion may be valid. 2123ms.

2 commands were executed. The results are:
   #1: Instance found. show is consistent.
   #2: No counterexample found. solutionConsultation may be valid.
```

*Assertion, predicates, function and run*

# GENERATED WORLD




calendar: 2
currentPosition: 1
currentTime: 1
email: 2
endDateTime: 4
endEvent: 2
event: 1
externalAccounts: 2
homeAddress: 2
meansOfTransport: 2
meansOfTransport: 2
name: 2
position: 1
preferences: 2
startDateTime: 4
startingDateTime: 2
time: 1
timeDuration: 1
timeDuration: 2
timeOccupation: 4
user: 2
System
User1
User0
Stringa0
ExternalAccount1
ExternalAccount0
TransportSolution1
TransportSolution0
PositionUpdate
Calendar1
Calendar0
UnreachableNotification
PublicMeanOfTransport
Stringa1
5
Event0
Event1
FreeTime
Event2
7
4
6
1
3
2
user
email
externalAccounts
homeAddress
name
currentPosition
calendar
preferences
meansOfTransport
currentTime
startingDateTime
timeDuration
endEvent
position
timeOccupation
event
startingDateTime
time
endDateTime
startDateTime



System (s)
User0
User1
Calendar0
Calendar1
FreeTime
PositionUpdate0
PositionUpdate1
OverlappingNotification0
OverlappingNotification1
UnreachableNotification
TransportSolution0
TransportSolution1
TransportSolution2
ExternalAccount0
ExternalAccount1
ExternalAccount2
Event0 end: booleanFalse
Event1 end: booleanTrue
Event2 (e) end: booleanFalse
PublicMeanOfTransport1
PublicMeanOfTransport2
PublicMeanOfTransport3
PrivateMeanOfTransport
Stringa2
Stringa3
1
2
3
4
5
6
7
currentPosition: 2
currentTime: 1
email: 2
endDateTime: 4
endEvent: 3
event: 1
externalAccounts: 3
id: 2
location: 3
meansOfTransport: 3
meansOfTransport: 7
meansPossessed: 1
name: 2
position: 2
preferences: 7
startDateTime: 4
startDateTime: 3
startLocation: 3
time: 2
time1: 2
time2: 2
timeDuration: 3
timeOccupation: 4
user: 2
user: 3
user: 2

## EFFORT SPENT

| | | |
|---|---|---|
| Agostini Andrea: | Introduction: | 1h |
| | Overall description: | 8h |
| | Specification requirements: | 10h |
| | Alloy: | 4h |
| | Overall Design: | 4h |
| Ciampiconi Lorenzo: | Introduction: | 2h |
| | Overall description: | 5h |
| | Specification requirements: | 10h |
| | Alloy: | 10h |
| | Overall Design: | 4h |
| Es-skidri Rachid: | Introduction: | 1h |
| | Overall description: | 5h |
| | Specification requirements: | 10h |
| | Alloy: | 2h |
| | Overall Design: | 10h |

# REFERENCES

[1] Google, Android Developers - Design https://developer.android. com/design/index.html

[2] Software Engineering 2 Project, AA 2017/2018 - Project goal, schedule and rules

[3] Software Engineering 2 Project, AA 2017/2018 - Assignments 1 and 2

[4] IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications

[5] MIT, alloy: a language & tool for relational models http://alloy.mit. edu/alloy/

[6] Software Abstractions (Logic, Language and Analysis) – Daniel Jackson

[7] Il linguaggio Alloy nella specifica formale di modelli UML – Tiziano Verone