

**Travlendar+ project Andrea Mafessoni,  
Andrea Mazzeo, Daniele Moltisanti**



**POLITECNICO**  
MILANO 1863

# **Requirement Analysis and Specification Document**

---

<b>Deliverable:</b>	RASD
<b>Title:</b>	Requirement Analysis and Verification Document
<b>Authors:</b>	Andrea Mafessoni, Andrea Mazzeo, Daniele Moltisanti
<b>Version:</b>	1.0
<b>Date:</b>	29-October-2017
<b>Download page:</b>	<a href="https://github.com/AndreaMazzeo289/MafessoniMazzeoMoltisanti.git">https://github.com/AndreaMazzeo289/MafessoniMazzeoMoltisanti.git</a>
<b>Copyright:</b>	Copyright © 2017, Andrea Mafessoni, Andrea Mazzeo, Daniele Moltisanti – All rights reserved

---

## Contents

<b>Table of Contents</b>	<b>3</b>
<b>List of Figures</b>	<b>5</b>
<b>List of Tables</b>	<b>5</b>
<b>1 Introduction</b>	<b>6</b>
1.1 Purpose	6
1.2 Scope	6
1.3 Definition, Acronyms, Abbreviations	6
1.3.1 definition	6
1.3.2 acronyms	7
1.3.3 abbreviations	7
1.4 Revision history	7
1.5 Reference documents	7
1.6 Document structure	7
<b>2 Overall Description</b>	<b>8</b>
2.1 Product perspective	8
2.2 Product functions	9
2.3 User characteristics	9
2.4 Domain Assumption and Dependencies	9
2.5 Constrains	9
<b>3 Specific Requirements</b>	<b>10</b>
3.1 External interface requirements	10
3.1.1 User interface	10
3.1.2 Software interface	20
3.2 Functional requirements	20
3.2.1 [G1] Allow a Guest to create a registered Travlendar+ account.	20
3.2.2 [G2] Allow an User to log in into his Travlendar+ account.	20
3.2.3 [G3] Allow an User to create a new appointment in his calendar.	20
3.2.4 [G4] Allow an User to delete an existing appointment from his calendar.	20
3.2.5 [G5] Allow an User to edit an existing appointment in his calendar.	21
3.2.6 [G6] Allow an User to view his appointments.	21
3.2.7 [G7] Allow an User to view his Daily Schedule	21
3.2.8 [G8] Allow an User to navigate and choose between different travel alternatives.	22
3.2.9 [G9] Allow an User to manage alerts for each appointment.	22
3.2.10 [G10] Allow an User to manage his travel preferences.	22
3.2.11 [G11] Allow an User to buy public transportation tickets.	23
3.3 Design constraints	23
3.3.1 Standards compliance	23
3.3.2 Hardware limitations	23
3.4 Software system attributes	23
3.4.1 Reliability	23
3.4.2 Availability	23
3.4.3 Security	23
3.4.4 Maintainability	24
3.4.5 Portability	24

<b>4</b>	<b>Scenarios</b>	<b>25</b>
4.1	Scenario 1	25
4.2	Scenario 2	25
4.3	Scenario 3	25
4.4	Scenario 4	25
<b>5</b>	<b>UML modelling</b>	<b>26</b>
5.1	Use case	26
5.1.1	Sign up	27
5.1.2	Log in	28
5.1.3	Manage preferences	28
5.1.4	View daily schedule	29
5.1.5	View travel details	30
5.1.6	Edit travel	30
5.1.7	View specific movement details	31
5.1.8	Choose prior appointment	31
5.1.9	Buy ticket	32
5.1.10	Delete appointment	33
5.1.11	Create appointment	34
5.1.12	Edit appointment	35
5.1.13	Create flexible appointment	36
5.1.14	Create repeatable appointment	36
5.1.15	Create alert	37
5.1.16	Edit alert	38
5.1.17	Delete alert	39
5.1.18	Check appointments on calendar	39
5.1.19	Change calendar view	40
5.2	Sequence diagram	41
5.2.1	Guest registration	41
5.2.2	User Login	42
5.2.3	Appointment creation	43
5.2.4	Appointment editing	44
5.2.5	Appointment deletion	45
5.2.6	Alert editing	46
5.2.7	Alert deletion	47
5.2.8	Scheduling and view travels	48
5.2.9	View and edit movements	49
5.3	Activity diagram	50
5.3.1	Create appointment	50
5.3.2	Edit appointment	51
5.3.3	Delete appointment	52
5.3.4	Create alert	52
5.3.5	Edit travel	53
5.3.6	Movement alternative	54
5.3.7	Buy ticket	55
<b>6</b>	<b>Formal Analysis Using Alloy</b>	<b>56</b>
<b>7</b>	<b>Effort Spent</b>	<b>60</b>
	<b>References</b>	<b>61</b>

## List of Figures

1	Login page	10
2	Connection of an existent account	10
3	Registration form	11
4	Daily view of calendar	11
5	Weekly view of calendar	12
6	Monthly view of calendar	12
7	Menu	13
8	Preferences menu	13
9	Transport means settings	14
10	Travel preferences	14
11	Daily schedule	15
12	New appointment	15
13	New alert	16
14	appointment options	16
15	Edit or delete appointment	17
16	Edit appointment	17
17	Edit or delete alert	18
18	Travel details	18
19	Movement details	19

## List of Tables

# 1 Introduction

## 1.1 Purpose

This document represents the Requirement Analysis and Specification Document (RASD). The goal of this document is to describe the software application and focus on all its features. Furthermore, it's interested to describe the functional and non-functional requirements of the system. Show the constraint, imposed by stakeholders and application environment, the limits of the software. This document is intended to all people that are interested to the project, such as stakeholders, investors and all developer and programmer that have to implement the application.

## 1.2 Scope

The application to develop is a mobile application that is called Travlendar+. This software is intended to help people with many commitments to manage the calendar on their smartphone. The only action that the user has to do is insert his daily appointments. The application should be able to organize the whole user's day, providing advice and reminding all inserted appointment. The application aims to be an advanced calendar management system, since it isn't a simple appointments reminder but it has a lot functionality that allow to the user to be always well organized. Lot are the functionality that the application provides, such as the complete transport management, that allow to compute the travel time and to identify the better travel solution basing on user's preferences and environment information, such as weather conditions. The user can choose if travel with own car or walk. He can decide to travel also in public transport and the application provides to the user the transport schedules and which transport choose. The system allows also the functionality to buy a ticket in-app. Furthermore, the application is able to find the car sharing or bike sharing points nearest to the user. It has an advices system when the appointment and the travel times overlaps. Daily the application can set a little time window (at most half an hour) reserved for the lunch. As this functionality, the user can schedule little break that the application set in day autonomously.

## 1.3 Definition, Acronyms, Abbreviations

### 1.3.1 definition

- **Daily view:**
- **Monthly view:**
- **Weekly view:**
- **Daily schedule:**
- **Appointment:**
- **Alert:**
- **Travel:**
- **Movement:**
- **Unreachability:**
- **Overlapping:**
- **Green:**
- **Cheaper:**

- **Faster:**
- **Repeatable:**
- **Flexible:**

### **1.3.2 acronyms**

### **1.3.3 abbreviations**

## **1.4 Revision history**

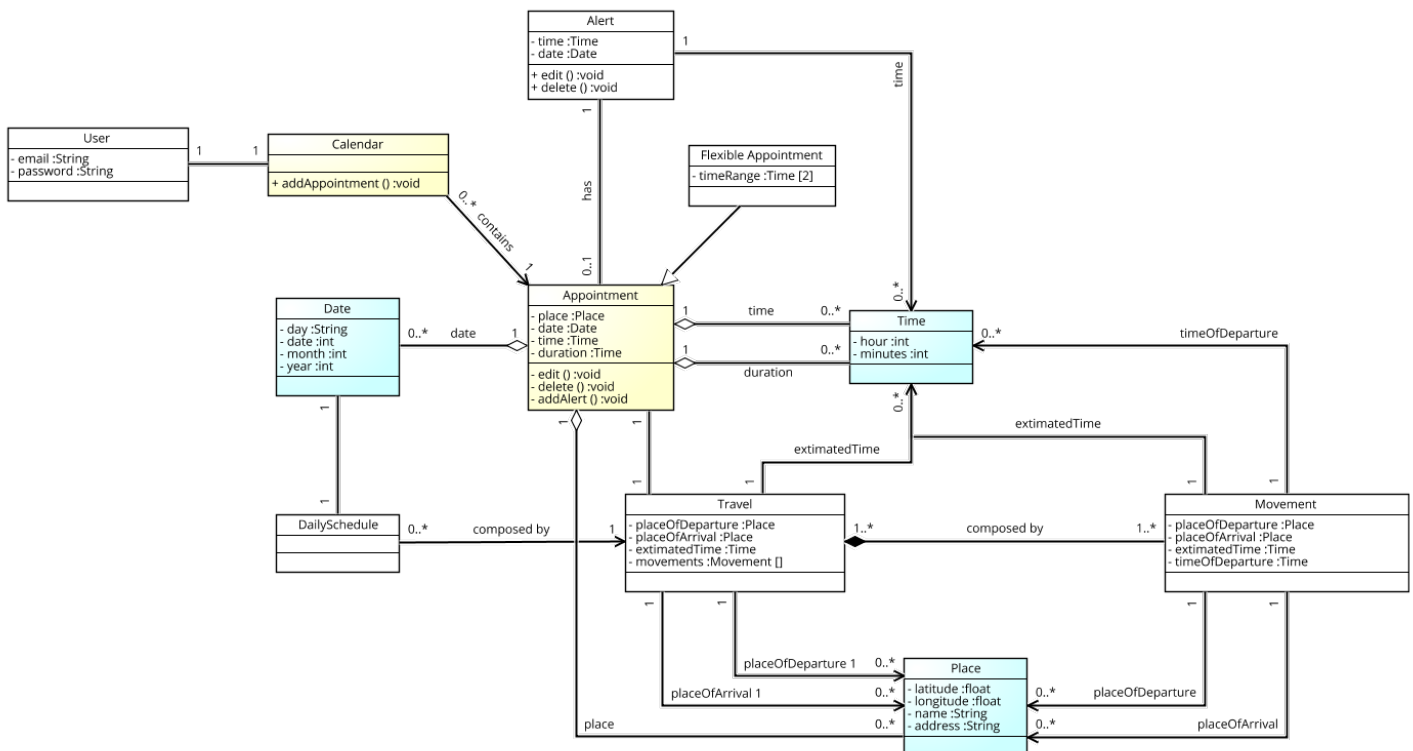
## **1.5 Reference documents**

## **1.6 Document structure**

## 2 Overall Description

## 2.1 Product perspective

The product we will provide is an application distributed for any kind of device that supports Android as operative system. This application will immediately be useble as soon as you install it on a device. It will not have any internal interface for administration but it will be only user based.



(UML e stateCharts)

## 2.2 Product functions

This application aims to provide a smart calendar, which schedules the best organization, taking account of your personal appointments, which you inserted in the calendar. The computed schedule depends on some preferences that you filled out and you can modify them when you want.

### 2.3 User characteristics

We recommend the application to a person who wants to organize easily his time in the best way. He will be able to benefit from this service in a very simple way because Travlendar+ requires only basic knowledge of a simple calendar. After registering an account, the application is ready to handle his commitments, so scheduling the best organization.

## 2.4 Domain Assumption and Dependencies

- For any day user can create unlimited number of events.
- User has only one calendar.
- There isn't any dependence between users.



- User can choose among some alternative travel proposals.
- If an event is overlapping another one, the user must select a choice from the choices proposed.
- User can delete an event.
- User can modify an event already created.
- User can change the scheduling proposed.
- User can select in which preferences the scheduling based on.
- Notification of best proposal will be shown.
- Notification of any problem that occurs will be shown.

## **2.5 Constrains**

Travlender+ requires:

- Internet connection enabled on own device
- GPS available on own device
- Login during the first access
- Initially registration with an account
- Android device
- Milano as the default city
- 30 Mb(?) of storage memory available on own devise to be installed

### 3 Specific Requirements

#### 3.1 External interface requirements

##### 3.1.1 User interface

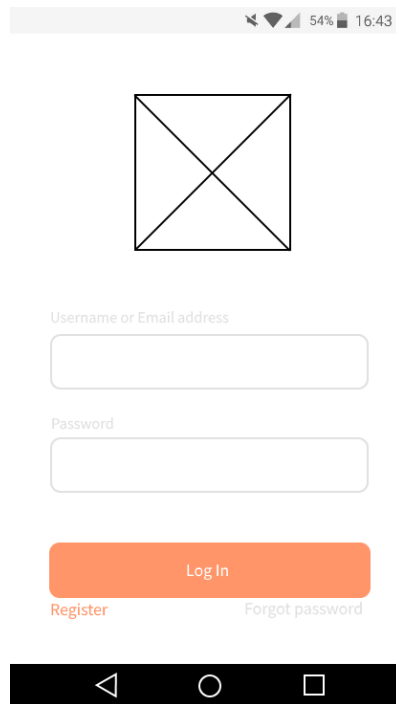


Figure 1: Login page

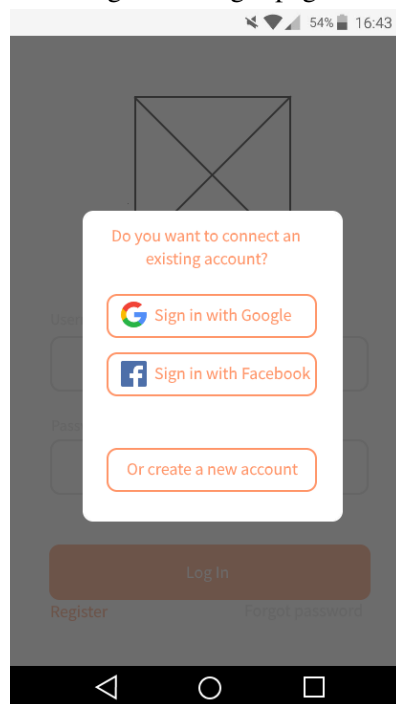
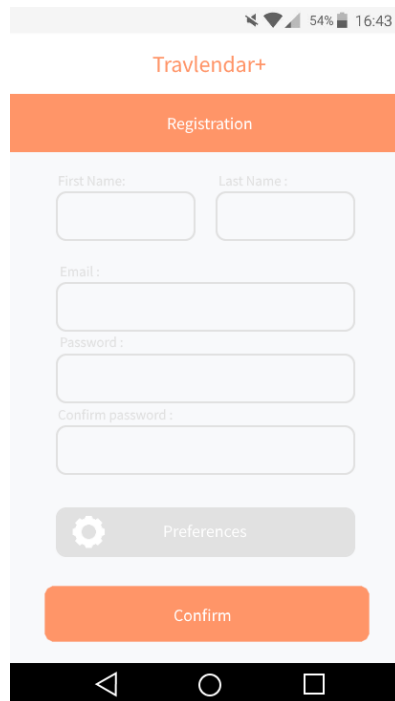


Figure 2: Connection of an existent account



The image shows a mobile app interface for the 'Travlendar+' registration form. At the top, there's a status bar with signal, Wi-Fi, 54% battery, and the time 16:43. Below it is the app's header 'Travlendar+' in orange. The main section is titled 'Registration' in white text on an orange background. The form contains five input fields: 'First Name:', 'Last Name:', 'Email:', 'Password:', and 'Confirm password:'. Below these fields is a grey button with a gear icon and the text 'Preferences'. At the bottom of the form is a large orange button labeled 'Confirm'. The Android navigation bar is visible at the very bottom.

Figure 3: Registration form

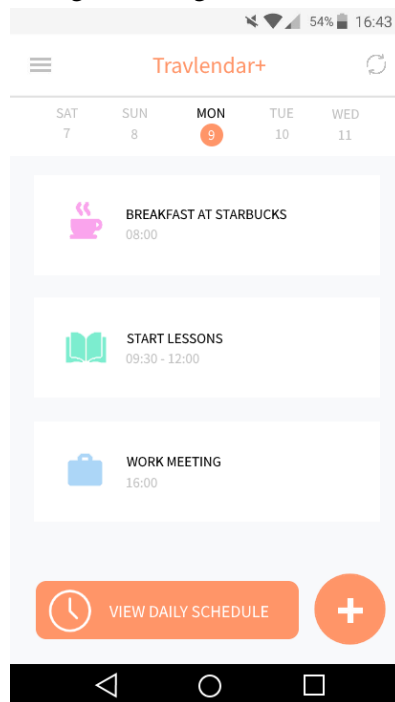


Figure 4: Daily view of calendar

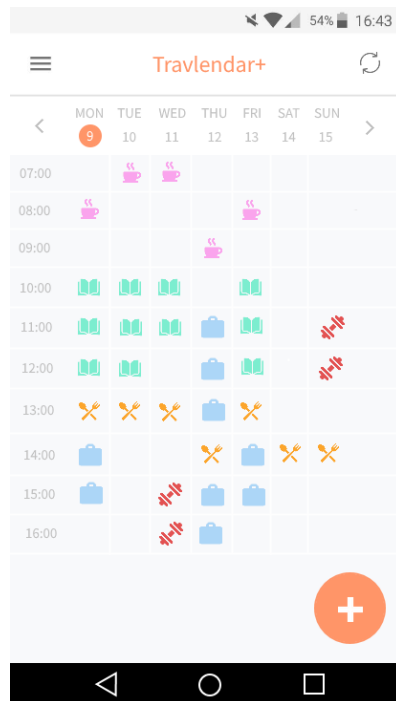


Figure 5: Weekly view of calendar

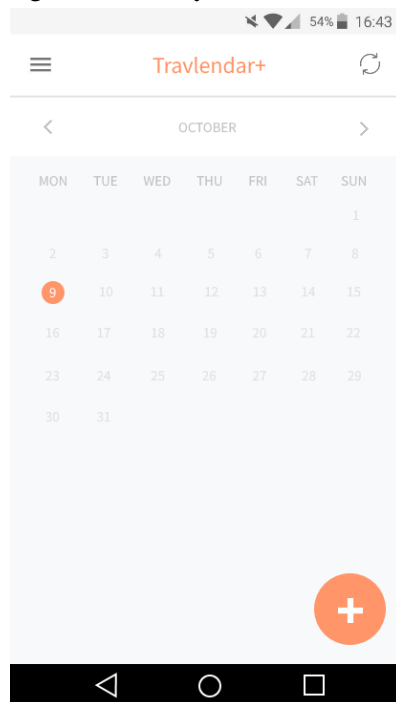


Figure 6: Monthly view of calendar

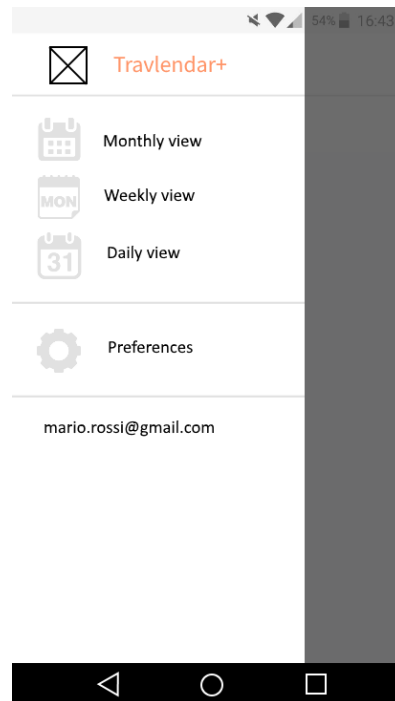


Figure 7: Menu

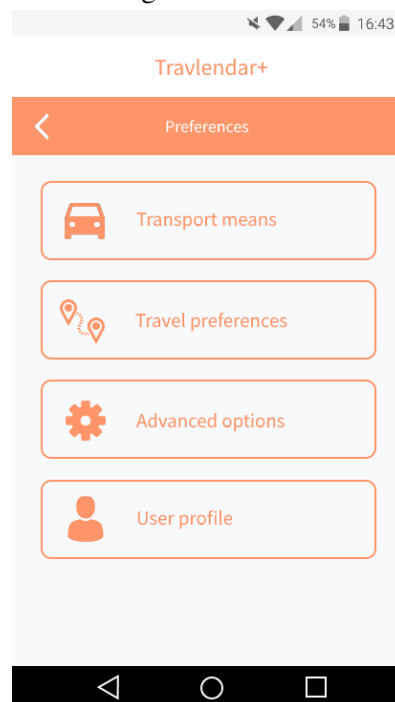


Figure 8: Preferences menu

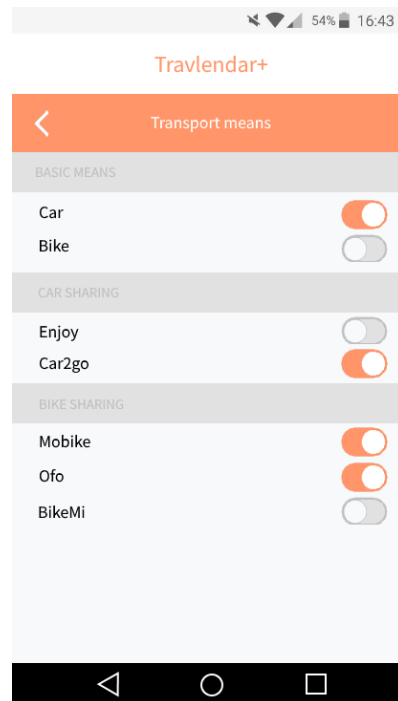


Figure 9: Transport means settings

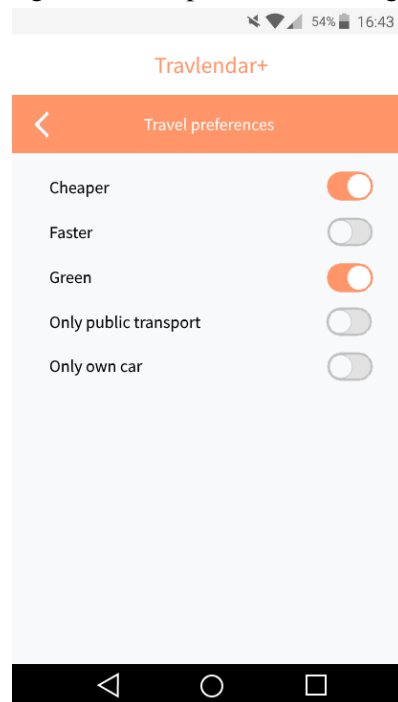


Figure 10: Travel preferences

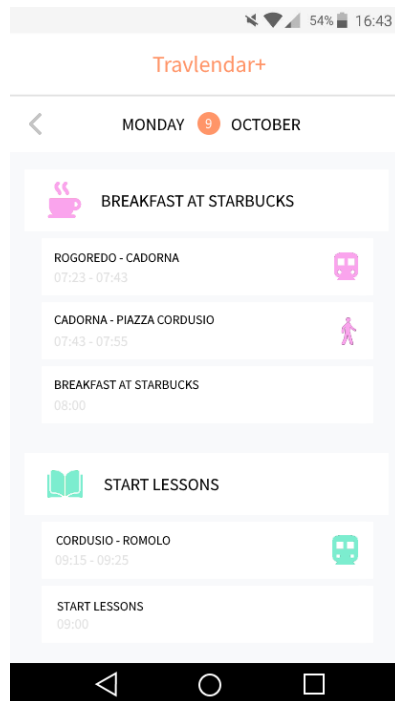


Figure 11: Daily schedule

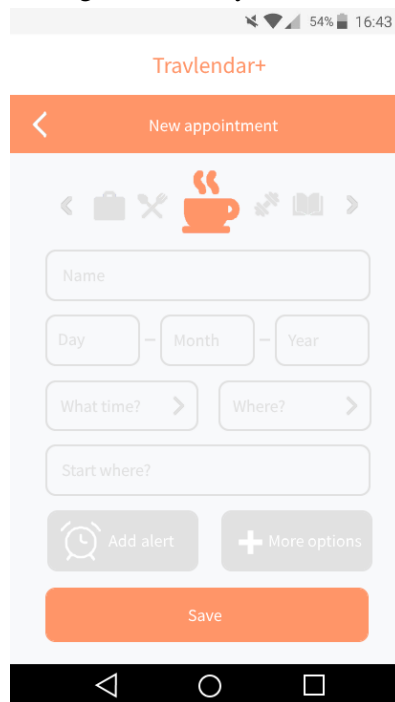


Figure 12: New appointment

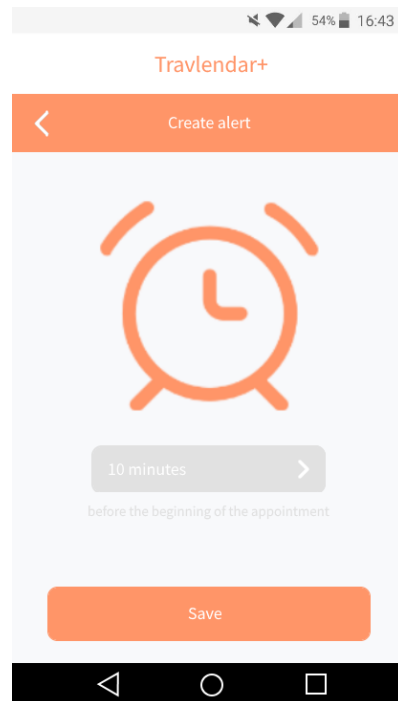


Figure 13: New alert

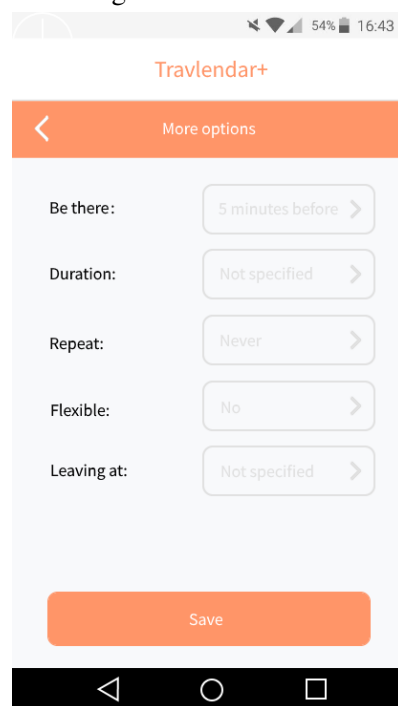


Figure 14: appointment options



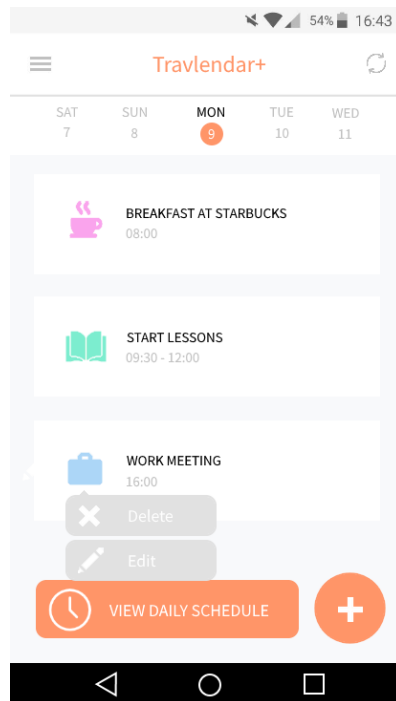


Figure 15: Edit or delete appointment

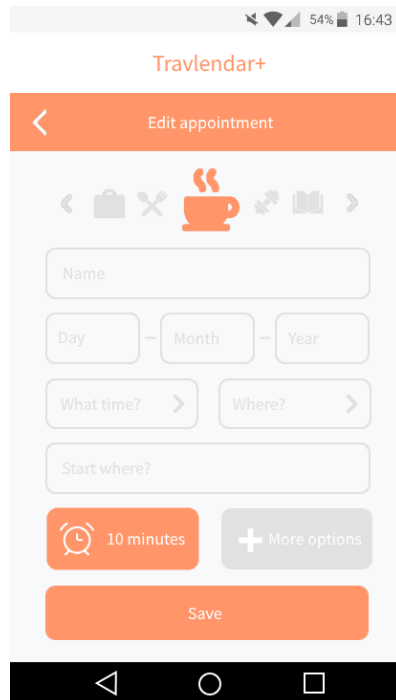


Figure 16: Edit appointment

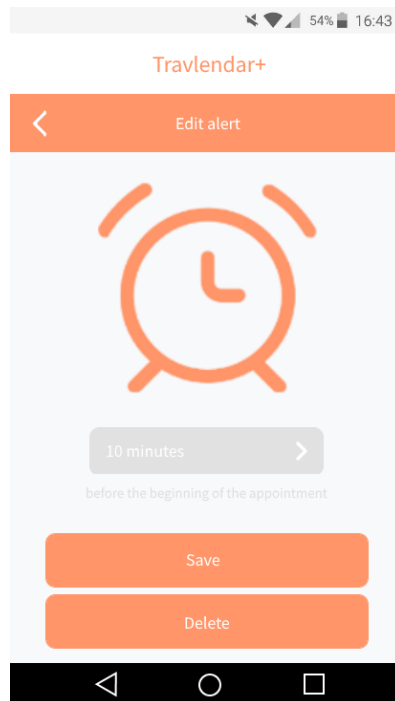


Figure 17: Edit or delete alert

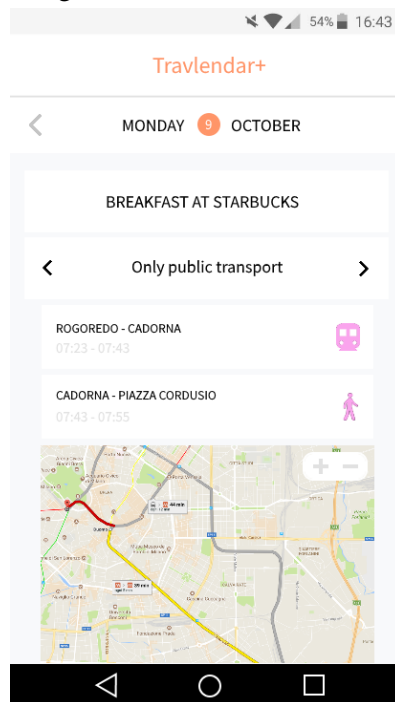


Figure 18: Travel details

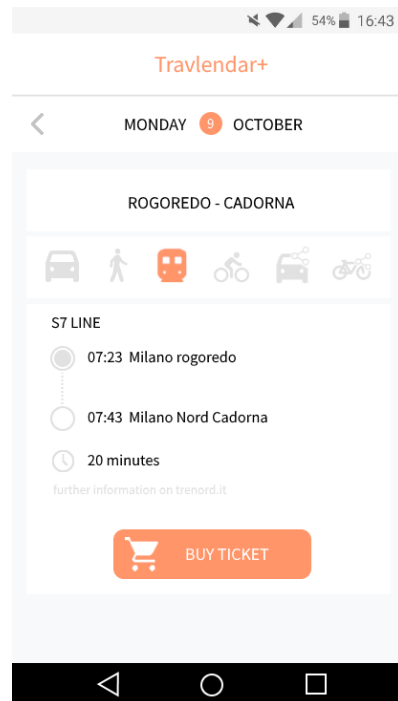


Figure 19: Movement details

### 3.1.2 Software interface

Inside application are used different API (Application programming interface):

- Weather API: <https://openweathermap.org/api>
- Google Maps API: <https://developers.google.com/maps/>
- Trenord API: <https://github.com/bluviolin/TrainMonitor/wiki/API-del-sistema-Viaggiatreno>
- Car2go API: <https://github.com/car2go/openAPI>
- Enjoy API: <https://github.com/mattiaongit/enjoy/blob/master/enjoy.py>
- BikeMi API: <https://github.com/pierlauro/bikemi-unofficial-api>
- MoBike API: <https://github.com/ubahnverleih/WoBike>

## 3.2 Functional requirements

**3.2.1 [G1] Allow a Guest to create a registered Travlendar+ account.**

**3.2.2 [G2] Allow an User to log in into his Travlendar+ account.**

**3.2.3 [G3] Allow an User to create a new appointment in his calendar.**

- [R9] The user must be logged into the system to access application features.
- [R10] The system must be able to provide the user with an overview of his calendar and the user must be able to view all appointments fixed in a certain period.
- [R11] The user must be able to pick a chosen day from the overview of his calendar.
- [R12] The user must be able to choose the option of creating a new appointment.
- [R13] The system must ask the user to provide all information needed for the creation of a new appointment, such as place and time of start and overall duration.
- [R14] The system must check if an appointment overlaps with other events and must eventually notify it to the user.
- [R15] The user must confirm the creation of the new appointment.
- [R16] The system must save the user modifications in memory and the calendar must be updated.
- [D2] All information provided by the user in the process of appointment creation or modification must be formally corrected.

**3.2.4 [G4] Allow an User to delete an existing appointment from his calendar.**

- [R17] The appointment intended to be modified must have been previously successfully created and not already deleted.
- [R9] The user must be logged into the system to access application features.
- [R10] The system must be able to provide the user with an overview of his calendar and the user must be able to view all appointments fixed in a certain period.
- [R11] The user must be able to pick a chosen day from the overview of his calendar.

- [R18] The user must be able to choose the option of deleting the appointment.
- [R19] The user must confirm the deletion.
- [R20] The system must remove a deleted appointment from the memory and cancel every alert related to it.
- [R16] The system must save the user modifications in memory and the calendar must be updated.
- [R21] Deleting process is not reversible.

### **3.2.5 [G5] Allow an User to edit an existing appointment in his calendar.**

- [R17] The appointment intended to be modified must have been previously successfully created and not already deleted.
- [R9] The user must be logged into the system to access application features.
- [R10] The system must be able to provide the user with an overview of his calendar and the user must be able to view all appointments fixed in a certain period.
- [R22] The user must be able to select a specific appointment in his calendar.
- [R23] The system must give the user access to all details of a selected appointment and the user must be allowed to edit the information needed.
- [R14] The system must check if an appointment overlaps with other events and must eventually notify it to the user.
- [R24] The user must confirm the modifications.
- [R16] The system must save the user modifications in memory and the calendar must be updated.
- [D2] All information provided by the user in the process of appointment creation or modification must be formally corrected.

### **3.2.6 [G6] Allow an User to view his appointments.**

- [R9] The user must be logged into the system to access application features.
- [R10] The system must be able to provide the user with an overview of his calendar and the user must be able to view all appointments fixed in a certain period.
- [R25] The user must be able to switch between different possible calendar, such as daily calendar, weekly calendar and monthly calendar.

### **3.2.7 [G7] Allow an User to view his Daily Schedule**

- [R9] The user must be logged into the system to access application features.
- [R26] The user must be able to select a specific day from his calendar.
- [R27] The system must be able to provide detailed information about the scheduled travels for a chosen day, showing the trace route and the estimated time required from each movement.
- [R28] The system must be able to choose a route between the possible travel alternatives according to the preferences expressed in the user profile settings and the information about external weather.

- [D3] Travel data are provided by an external agent.
- [D4] Information about weather are provided by an external agent.
- [D5] If the system displays a travel alternative, it means that it's actually possible to successfully perform that travel in the way and in the time displayed.

### **3.2.8 [G8] Allow an User to navigate and choose between different travel alternatives.**

- [R9] The user must be logged into the system to access application features.
- [R26] The user must be able to select a specific day from his calendar.
- [R27] The user must be able to select a specific travel in the chosen day.
- [R28] The system must be able to provide the user with an overview of the possible travel alternatives for the chosen travel, specifying all details for each one.
- [R29] The user must be able to filter the travel alternatives furnished by the system according to defined parameters, such as time of travelling or overall cost.
- [R30] The user must be able to choose a favourite travel option different from the displayed default one.
- [R31] The system must update the daily schedule according to the travel option chosen by the user and the user must be able to see the new updated schedule.
- [D5] If the system displays a travel alternative, it means that it's actually possible to successfully perform the travel in the way and in the time displayed.

### **3.2.9 [G9] Allow an User to manage alerts for each appointment.**

- [R32] The system must give the user the possibility of adding an alert to an appointment while it is being created or modified.
- [R33] The user must be able to choose a desired interval of time for the warning alert.
- [R34] The user must confirm the alert creation and the system must save the insertion in the memory.
- [R35] The user must be able to modify or remove the inserted alert when needed.
- [R36] In case of any alert modification made by the user, the user must confirm the modification and the system must save all changes.
- [D6] If a user creates a new alert, he must receive the notification after the specified amount of time.

### **3.2.10 [G10] Allow an User to manage his travel preferences.**

- [R9] The user must be logged into the system to access application features.
- [R39] The user must be able to access the preferences panel of his account.
- [R40] The system must give the user the possibility of setting various preferences, such as owned and preferred travel means, address of Home and other general travel preferences.
- [R41] The user must be able to edit the provided preferences when needed.

### **3.2.11 [G11] Allow an User to buy public transportation tickets.**

- [R9] The user must be already logged into the system to access his calendar.
- [R26] The user must be able to select a specific day from his calendar.
- [R27] The user must be able to select a specific travel in the chosen day.
- [R37] The system must give to the user the possibility of buying the ticket for the selected travel.
- [R38] The system must save a copy of the bought tickets and the user must be able to view them when needed.
- [D7] The payment process and ticket acquisition is made by an external public transport service.

## **3.3 Design constraints**

### **3.3.1 Standards compliance**

The application must require to the user different permissions:

- Accesso to the calendar;
- Get hit position with GPS;
- Access to device storage.

### **3.3.2 Hardware limitations**

The application, at the moment, runs only on Android 4.0.3 version or newer.

The device needs:

- Internet connection;
- GPS;
- Space for save application in memory.

Actual devices on the market satisfy all these requirements.

## **3.4 Software system attributes**

### **3.4.1 Reliability**

The system must guarantee a 24/7 service.

### **3.4.2 Availability**

The system requires a GPS service and internet connection in order to work properly. When the connection is down the system works with the last updated information available in the device memory.

### **3.4.3 Security**

The application must provide secure storage for all sensitive data inserted by the user. One way to achieve it is the use of cryptographical techniques.

#### **3.4.4 Maintainability**

The application now is in beta version, this means that can present some bugs. Certainly, the application will be periodically upgraded and each release allow to remove known bugs and add more functionality. Periodically all information that are stored inside the application must be backed up, in order to reduce danger of lost information in case of malfunction of the application.

#### **3.4.5 Portability**

Now the application has been developed only for android device (more specifically only for android version 4.0.3 Ice Cream Sandwich or newer versions). Further developments will lead this application also in iOS devices. Another possible development is the creation of a web site that is synchronized with application, and allow to the user to control their appointment also on desktop pc and laptop.



## 4 Scenarios

### 4.1 Scenario 1

Andrea has just booked a last minute flight from Milan Orio al Serio airport to Prague, in Czech Republic, but being not a frequent flyer, he's pretty worried about the idea of losing his plane. Furthermore, he lives far away from the airport and he needs to reach it by public transportation. Two days before his departure, he decides to download Travlendar+ app. He creates a new account by connecting his Facebook profile and fills out the essential account settings, giving his basic preferences (he doesn't have any car or bike and he prefers cheaper travels). Then he creates a new appointment called "Prague" in his calendar, inserting the date of Saturday 11/11/17, the time 13:00 and the location of the airport. He chooses the option "I want to be there..." and select "2 hours before". Then he creates the new appointment. His calendar now shows the "Prague" event, and the app easily displays how to reach the airport in time, by leaving home at 9:32, walking until the nearest metro station, taking the metro until Stazione Centrale and then taking a public bus to the airport at 10:10. Now Andrea feels much more confident!

### 4.2 Scenario 2

Serena has been using the Travlendar+ app since a couple of weeks. She has just bought a new car that allows her to move through the city in a easier way, and wants the app to consider that when it display the optimal travel solution. She opens the app and moves to the preferences panel of her account. She then select "Travel means owned" and puts a tick in the "car" option. Then, she open her calendar and check her daily schedules for the next three days. Some of the travels suggested by the app are now changed and replaced with faster and more comfortable movements with car.

### 4.3 Scenario 3

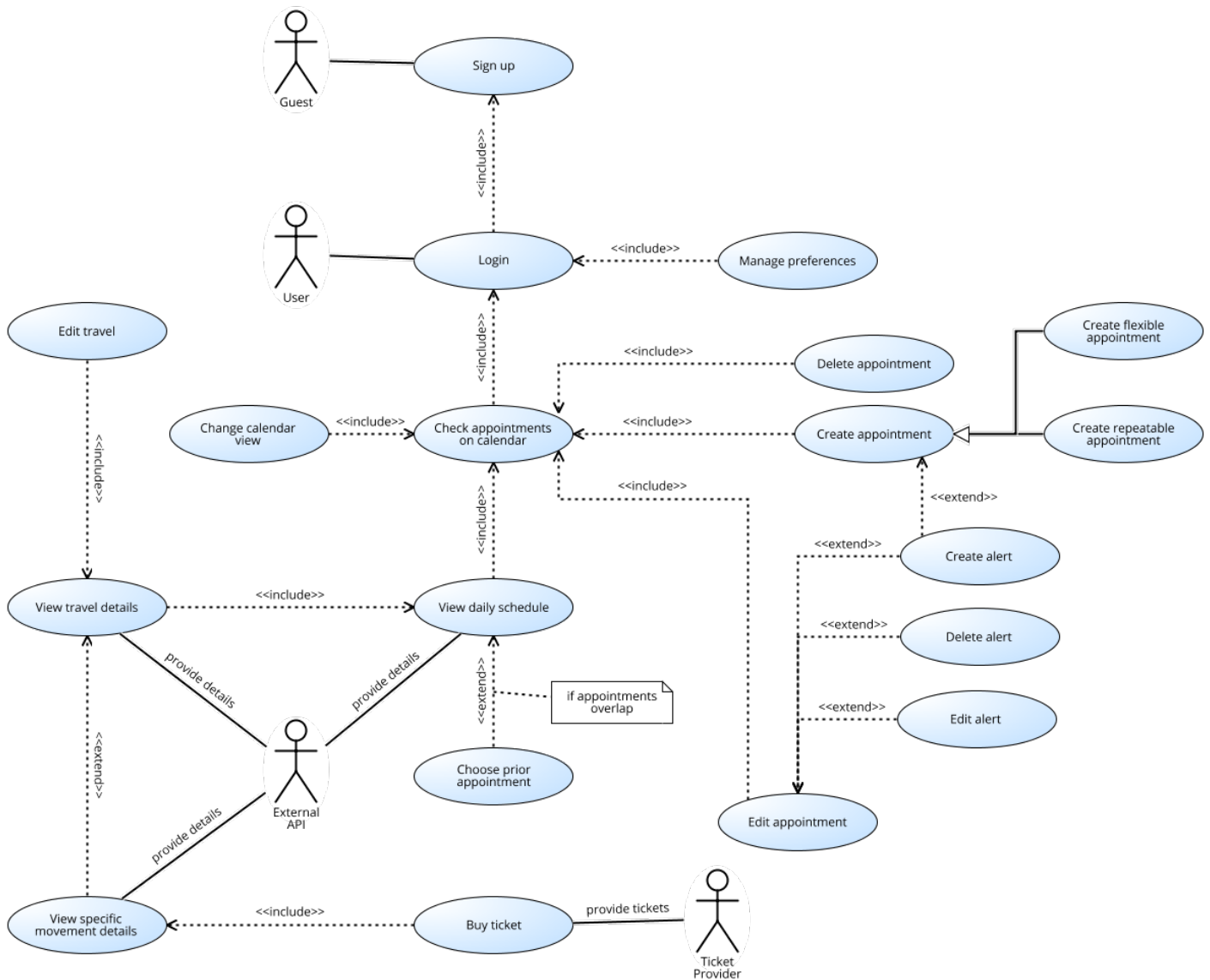
Marco has a scheduled appointment in program for the following day, and the app used to suggest a 7 minutes-movement by bike until the train station. But now the weather widget seems to announce a rainy day, and the app switched to a warmer travel by metro. But Marco is not afraid of rain and likes walking, so he opens the daily schedule, selects the metro movement and checks the possible travel alternatives. He then chooses the option "by walk". The app nows shows the updated schedule, and the system will remember the choice for the future.

### 4.4 Scenario 4

Riccardo is a pretty absent-minded man, and has a morning full of appointments in schedule for the next day. He has added all the meetings in his calendar, but must absolutely not forget them, so he decides to add an alarm to each of them, in order to remember his commitments. He opens the app and select one by one all his appointments of Tuesday. For each one, he taps on "add an alert to this appointment" and chooses to be warned 15 minutes before the start.

## 5 UML modelling

### 5.1 Use case



### 5.1.1 Sign up

<b>Name</b>	<b>Sign up</b>
<b>Actors</b>	Guest
<b>Entry conditions</b>	The guest is on the log in page of the application and clicks on “Register” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. A pop-up shows up asking to the guest if he wants to connect an existing account, such as Google or Facebook, or if he want to create a new account. <ol style="list-style-type: none"> <li>(a) If the guest connects his account, the system accepts the request and creates a new Travlendar+ account based on provided account.</li> <li>(b) If the guest chooses to create a new account, he is redirected to the registration page which contains all the fields to be filled.</li> </ol> </li> <li>2. The guest fills out all the mandatory fields.</li> <li>3. (Optional) The guest adjusts the preferences settings.</li> <li>4. The guest clicks on button “Confirm”.</li> <li>5. The system checks data provided and eventually creates and registers the user account.</li> </ol>
<b>Exit conditions</b>	The guest has successfully created a new account and he can log into the system with his credentials.
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>• Email provided is already in use. The system does not proceed in the registration process and the account is not created. It is possible to repeat the procedure.</li> <li>• Data provided are incorrect. The system highlights the incorrect fields and asks the user to repeat the procedure.</li> </ul>

### 5.1.2 Log in

<b>Name</b>	<b>Log in</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user launches the application and clicks on the “Log In” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user inserts his email.</li> <li>2. The user inserts his password.</li> <li>3. The user clicks on the “Log In” button.</li> </ol>
<b>Exit conditions</b>	The login procedure is successfully completed. The user is logged into the system and is able to access to all the functionalities.
<b>Exceptions</b>	The credentials provided are not associated to any existing account. The login procedure is rejected and the guest is brought back to the login page. It is possible to repeat the procedure.

### 5.1.3 Manage preferences

<b>Name</b>	<b>Manage preferences</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user clicks on “Preferences” from the side menu on the homepage.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The system shows the preferences settings, that include transport means owned, favorite kind of travel and other advanced options.</li> <li>2. The user accesses to the desired settings and adjusts the preferences as wanted.</li> <li>3. The user saves the changes.</li> </ol>
<b>Exit conditions</b>	The changes are saved and remembered from the system. The preferences have been updated.
<b>Exceptions</b>	The user clicks on “back” without having confirmed the creation. The new preferences are not saved and the application returns to the homepage. It is possible to repeat the procedure.

#### 5.1.4 View daily schedule

<b>Name</b>	<b>View daily schedule</b>
<b>Actors</b>	User, External APIs
<b>Entry conditions</b>	The user clicks on the “View daily schedule” button while checking an appointment on his calendar.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The system receives data from different external APIs about routes, traffic, weather and available transport means and computes the best travel option according to the user preferences.</li> <li>2. The system provides a graphic overview of all the travels scheduled for the day, with the relative movements.</li> <li>3. The user checks all the information needed and eventually clicks on a specific travel or movement to get further information.</li> </ol>
<b>Exit conditions</b>	User has obtained all the information needed and has clicked “back” to return to the homepage or has selected a specific travel/movement to get further information.
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>• Some of the appointments overlap. The system is not able to compute travels between the appointments and forces the user to choose a prior appointment.</li> <li>• Some of the appointments are not reachable in the allotted time. The system doesn’t show the travel and signals the problem to the user with a warning.</li> <li>• Internet connection is not available. The system fails to compute the best travel option and displays a warning to the user.</li> </ul>

### 5.1.5 View travel details

<b>Name</b>	<b>View travel details</b>
<b>Actors</b>	User, External APIs
<b>Entry conditions</b>	The user is checking his daily schedule and clicks on a specific travel.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The system receives data from different external APIs and collects detailed information about the travel, such as the specific itinerary on the map and the weather conditions.</li> <li>2. The user is redirected to a new page that displays all the information about the selected travel.</li> <li>3. The user checks all the information needed and eventually clicks on a specific movement to get further information or buy a ticket.</li> </ol>
<b>Exit conditions</b>	User has obtained all the information needed and has clicked “back” to return to the daily schedule or has selected a specific movement to get further information.
<b>Exceptions</b>	Internet connection is not available. The system fails to obtain the needed information and displays a warning to the user.

### 5.1.6 Edit travel

<b>Name</b>	<b>Edit travel</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is checking a specific travel.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user provides modifications to the travel as followed: <ol style="list-style-type: none"> <li>(a) Clicking on the travel preferences and switching to a different alternative.</li> <li>(b) Clicking on a specific movement and changing the transport mean.</li> </ol> </li> <li>2. The user saves the changes.</li> </ol>
<b>Exit conditions</b>	The changes are saved and the system displays the new modified travel.
<b>Exceptions</b>	There are no possible alternatives for the selected travel. The system displays a warning to notify the user.

### 5.1.7 View specific movement details

<b>Name</b>	<b>View specific movement details</b>
<b>Actors</b>	User
<b>Entry conditions</b>	User is checking a specific travel and select a specific movement.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The system receives data from different external APIs and collects detailed information about the movement, such as the specific itinerary on the map, the weather conditions and the eventual availability of tickets.</li> <li>2. The user is redirected to a new page that displays all the information about the selected movement.</li> <li>3. The user checks all the information needed and eventually proceeds in buying a ticket.</li> </ol>
<b>Exit conditions</b>	User has obtained all the information needed and has clicked “back” to return to the travel page or has proceeded in buying a ticket.
<b>Exceptions</b>	Internet connection is not available. The system fails to obtain the needed information and displays a warning to the user.

### 5.1.8 Choose prior appointment

<b>Name</b>	<b>Choose prior appointment</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is checking the schedule and two appointments overlap.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The system signals the user the exactly time overlapping of the appointments.</li> <li>2. The user clicks on the preferred appointment.</li> <li>3. The system re-computes the daily schedule giving priority to the chosen appointment.</li> </ol>
<b>Exit conditions</b>	No more appointments overlap. The system shows the daily schedule without errors.
<b>Exceptions</b>	No exceptions expected

### 5.1.9 Buy ticket

<b>Name</b>	<b>Buy ticket</b>
<b>Actors</b>	User, Ticket provider
<b>Entry conditions</b>	User is checking a specific movement and clicks on “Buy ticket”
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The system connects trough APIs to the external system of the ticket provider.</li> <li>2. The system shows the user a form to fill with all the payment information.</li> <li>3. The user fills out all the fields and confirm the payment.</li> <li>4. The system sends information to the ticket provider and waits for confirmation and ticket data.</li> </ol>
<b>Exit conditions</b>	Payment is successfully fulfilled and bought tickets are available for the view. The user is brought back to the movement page.
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>• Payment is rejected. The system notifies the user with a warning. It is possible to repeat the procedure.</li> <li>• Internet connection is not available. The system fails to connect to the ticket provider and notifies the user with a warning. The application returns to the movement page.</li> </ul>



### 5.1.10 Delete appointment

<b>Name</b>	<b>Delete appointment</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user selects a specific appointment in his calendar (through daily or weekly view) and clicks on “Delete” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. A pop-up shows up, asking the user to confirm the deletion.</li> <li>2. The user confirms the deletion.</li> <li>3. The system removes all the appointment information from the memory, alert included.</li> </ol>
<b>Exit conditions</b>	The appointment has been deleted and removed from the system. The user is redirected to his calendar page, which has been updated with the removal of the appointment.
<b>Exceptions</b>	The user does not confirm the deletion. The appointment has not been deleted and the user is redirected to his calendar page. It is possible to repeat the procedure.

### 5.1.11 Create appointment

<b>Name</b>	<b>Create appointment</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is checking his calendar and clicks on “Create appointment” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user is redirected to the appointment creation page, which contains all the fields required to perform the creation.</li> <li>2. The user fills out all mandatory fields. The following steps aren’t mandatory: <ol style="list-style-type: none"> <li>(a) User chooses among the available icons.</li> <li>(b) User adds an alert to remember the appointment.</li> <li>(c) User clicks on “More options” and provides more detailed options.</li> </ol> </li> <li>3. User confirms the creation.</li> <li>4. The system saves the appointment information.</li> </ol>
<b>Exit conditions</b>	The appointment is created and inserted in the system. The user is redirected to his calendar page, which has been updated with the new appointment.
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>• The user has provided incorrect information: the appointment is not created and the user must repeat the procedure.</li> <li>• The user clicks on “back” without having confirmed the creation. The appointment is not created and the application returns to the calendar page. It is possible to repeat the procedure.</li> <li>• The appointment overlaps with other previously created appointments. The appointment is created and inserted anyway, but the user receives a notification of the overlapping.</li> </ul>

### 5.1.12 Edit appointment

<b>Name</b>	<b>Edit appointment</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user selects a specific appointment in his calendar (through daily or weekly view) and clicks on “Edit” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user is redirected to the appointment editing page which contains all the previously inserted information.</li> <li>2. The user can perform the following actions: <ol style="list-style-type: none"> <li>(a) Changing the appointment icon.</li> <li>(b) Editing the information.</li> <li>(c) Adding an alert to the appointment.</li> <li>(d) Clicking on “More options” and providing more detailed options.</li> </ol> </li> <li>3. The user saves the changes.</li> <li>4. The system saves the updated appointment information.</li> </ol>
<b>Exit conditions</b>	The changes are saved and the appointment has been modified. The user is redirected to his calendar page, which has been updated with the new information provided.
<b>Exceptions</b>	<ul style="list-style-type: none"> <li>• The user has provided incorrect information: the changes are not saved and the user must repeat the procedure.</li> <li>• The user clicks on “back” without having saved the changes. The appointment has not been modified and the application returns to the calendar page. It is possible to repeat the procedure.</li> <li>• The appointment now overlaps with other previously created appointments. The changes are saved anyway, but the user receives a notification of the overlapping.</li> </ul>

### 5.1.13 Create flexible appointment

<b>Name</b>	<b>Create flexible appointment</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is creating/editing an appointment, clicks on “More options” and clicks on the “Flexible” field.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. A pop-up shows up, containing the fields to be filled.</li> <li>2. The user fills out the fields, specifying the time range of the appointment.</li> <li>3. The user confirms the choice.</li> </ol>
<b>Exit conditions</b>	The choice is saved for later and the user is back on the More option page. At the end of the creation process, the appointment will be created at a time compatible with the time interval provided.
<b>Exceptions</b>	The user clicks on “back” without having confirmed the choice. The appointment has not been modified and the application returns to the More options page. It is possible to repeat the procedure.

### 5.1.14 Create repeatable appointment

<b>Name</b>	<b>Create repeatable appointment</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is creating/editing an appointment, clicks on “More options” and clicks on the “Flexible” field.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. A pop-up shows up, containing the fields to be filled.</li> <li>2. The user fills out the fields, specifying the days in which the appointment is wanted to be created.</li> <li>3. The user confirms the choice.</li> </ol>
<b>Exit conditions</b>	The choice is saved for later and the user is back on the More option page. The appointment is now a repeatable appointment.
<b>Exceptions</b>	The user clicks on “back” without having confirmed the choice. The appointment has not been modified and the application returns to the More options page. It is possible to repeat the procedure.

### 5.1.15 Create alert

<b>Name</b>	<b>Create alert</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is on the appointment creation/editing page and clicks on the “Add alert” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user is redirected to the alert creation/editing page.</li> <li>2. The user fills the field specifying the time of the alert.</li> <li>3. The user confirms the creation.</li> <li>4. The system saves the alert information.</li> </ol>
<b>Exit conditions</b>	The alert has been created and inserted in the system. The application returns to the appointment creation/editing page and it is now possible to edit the alert.
<b>Exceptions</b>	The user clicks on “back” without having confirmed the creation. The alert is not created and the application returns to the event creation page. It is possible to repeat the procedure.

### 5.1.16 Edit alert

<b>Name</b>	<b>Edit alert</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is on the appointment creation/editing page and clicks on the “Edit alert” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user is redirected to the alert creation/editing page.</li> <li>2. The user replaces needed information with updated ones.</li> <li>3. The user saves the changes.</li> <li>4. The system saves the alert information.</li> </ol>
<b>Exit conditions</b>	All the changes have been saved and inserted in the system. The application returns to the appointment creation/editing page and it is possible to repeat the procedure.
<b>Exceptions</b>	The user clicks on “back” without having saved the changes. The changes have not been saved and the application returns to the appointment creation/modification page. It is possible to repeat the procedure.

### 5.1.17 Delete alert

<b>Name</b>	<b>Delete alert</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is on the appointment creation/editing page and clicks on the “Edit alert” button.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user is redirected to the alert creation/editing page.</li> <li>2. The user clicks on “Delete”.</li> <li>3. The user confirms the deletion.</li> <li>4. The system removes all the alert information from the memory.</li> </ol>
<b>Exit conditions</b>	The alert has been deleted and removed from the system. The application returns to the appointment creation/editing page and it is now possible to add a new alert.
<b>Exceptions</b>	The user does not confirm the deletion. The alert has not been deleted and the application returns to the alert editing page. It is possible to repeat the procedure.

### 5.1.18 Check appointments on calendar

<b>Name</b>	<b>Check appointments on calendar</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is logged in and he is on the homepage of the application.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The system shows an overview of the existing appointments.</li> <li>2. The user moves between the appointments and collects the needed information.</li> </ol>
<b>Exit conditions</b>	The user has collected the needed information and moves to a different page.
<b>Exceptions</b>	No exception expected.

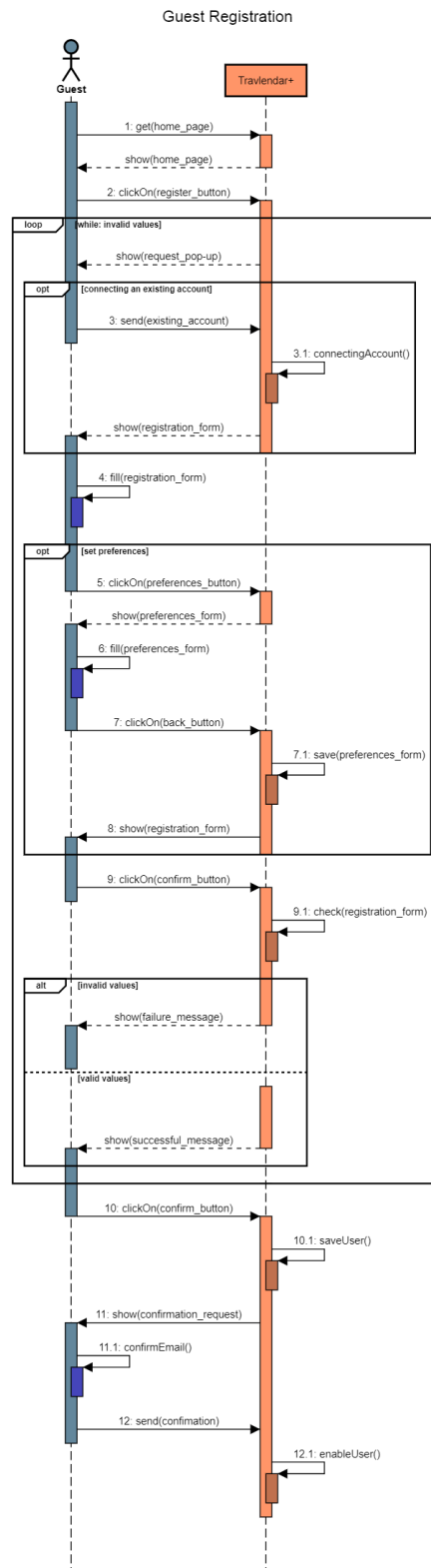
### 5.1.19 Change calendar view

<b>Name</b>	<b>Change calendar view</b>
<b>Actors</b>	User
<b>Entry conditions</b>	The user is on the homepage of the application and he is checking his appointments.
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. The user opens the lateral menu.</li> <li>2. The user clicks on one of available view (daily, weekly or monthly).</li> <li>3. The system changes the layout according to the user's choice.</li> </ol>
<b>Exit conditions</b>	The layout is changed and the user is back on the calendar view.
<b>Exceptions</b>	No exception expected.

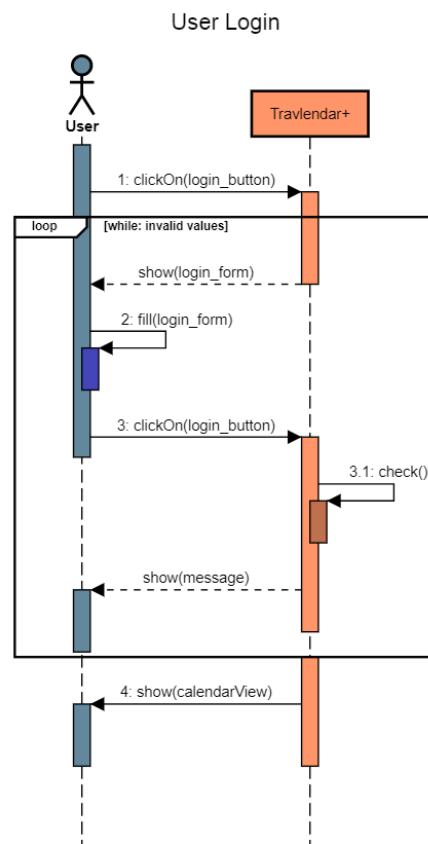


## 5.2 Sequence diagram

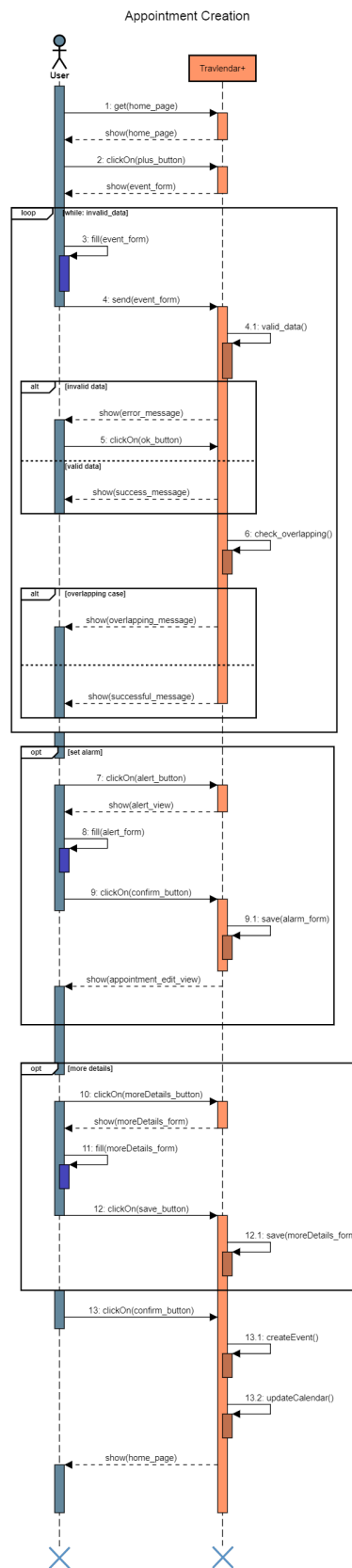
### 5.2.1 Guest registration



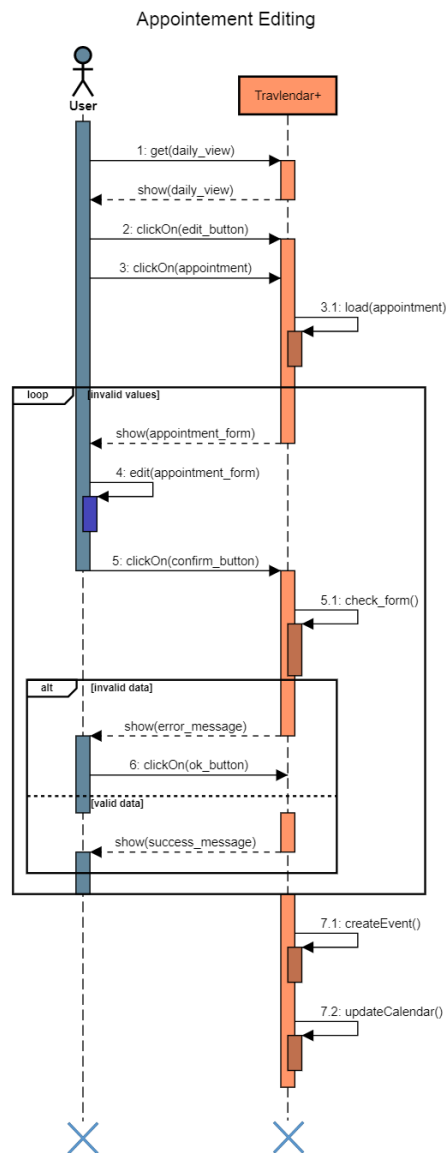
## 5.2.2 User Login



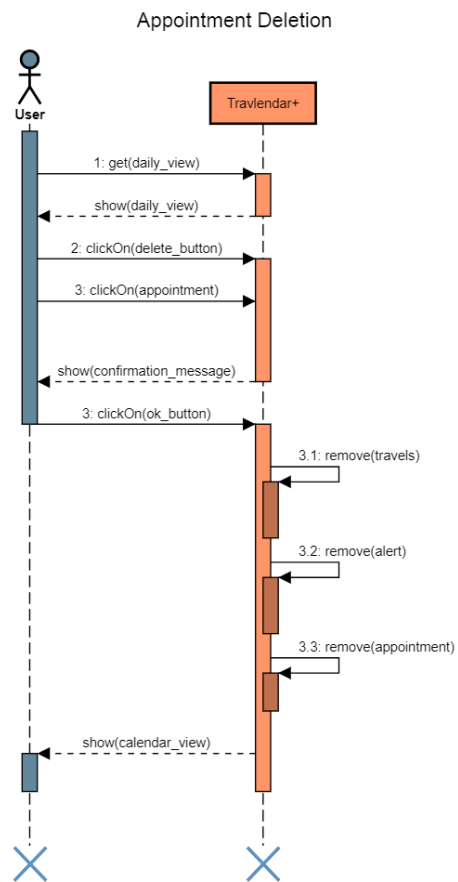
### 5.2.3 Appointment creation



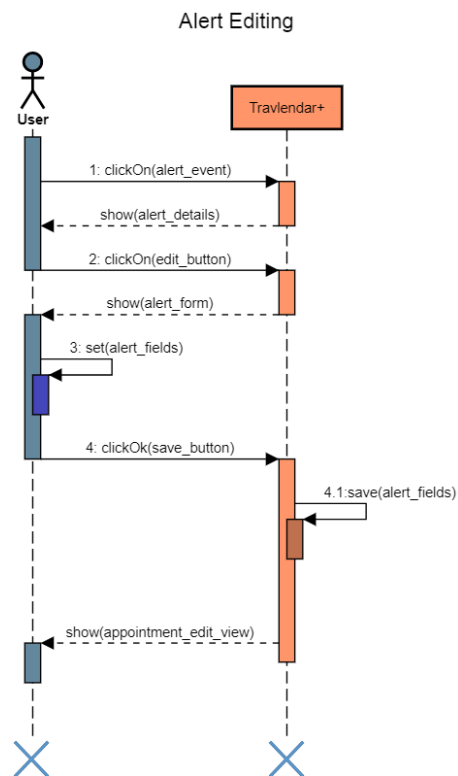
## 5.2.4 Appointment editing



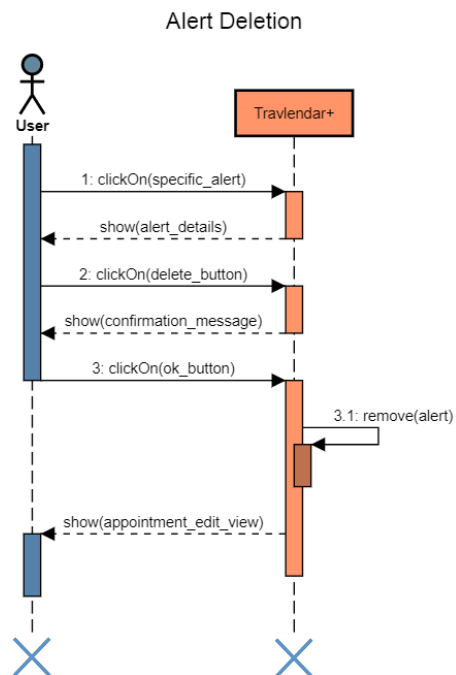
### 5.2.5 Appointment deletion



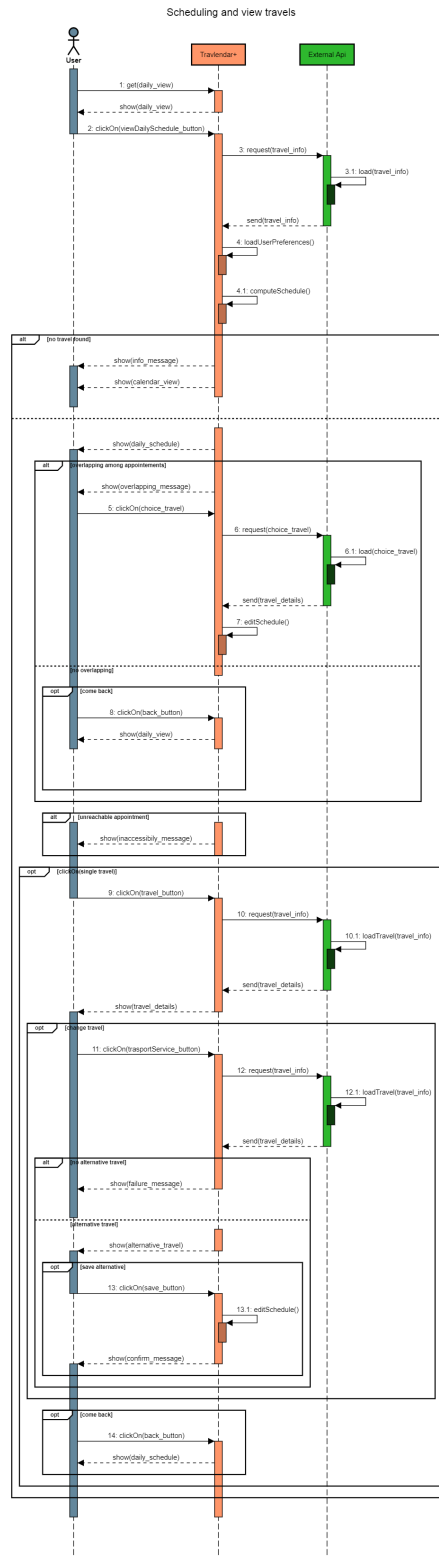
## 5.2.6 Alert editing



### 5.2.7 Alert deletion

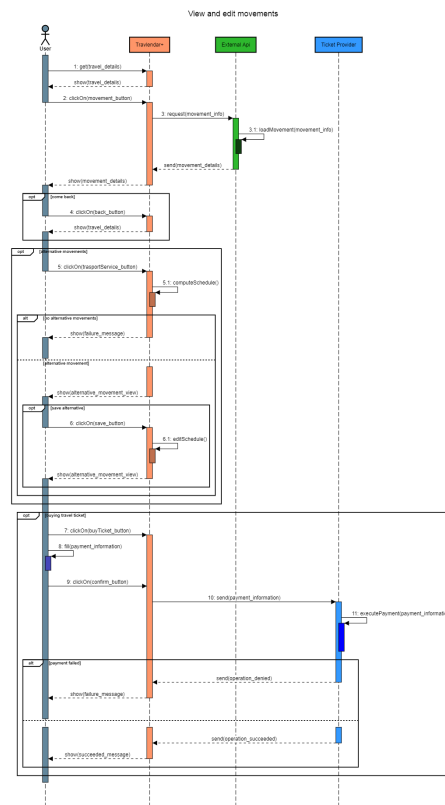


## 5.2.8 Scheduling and view travels



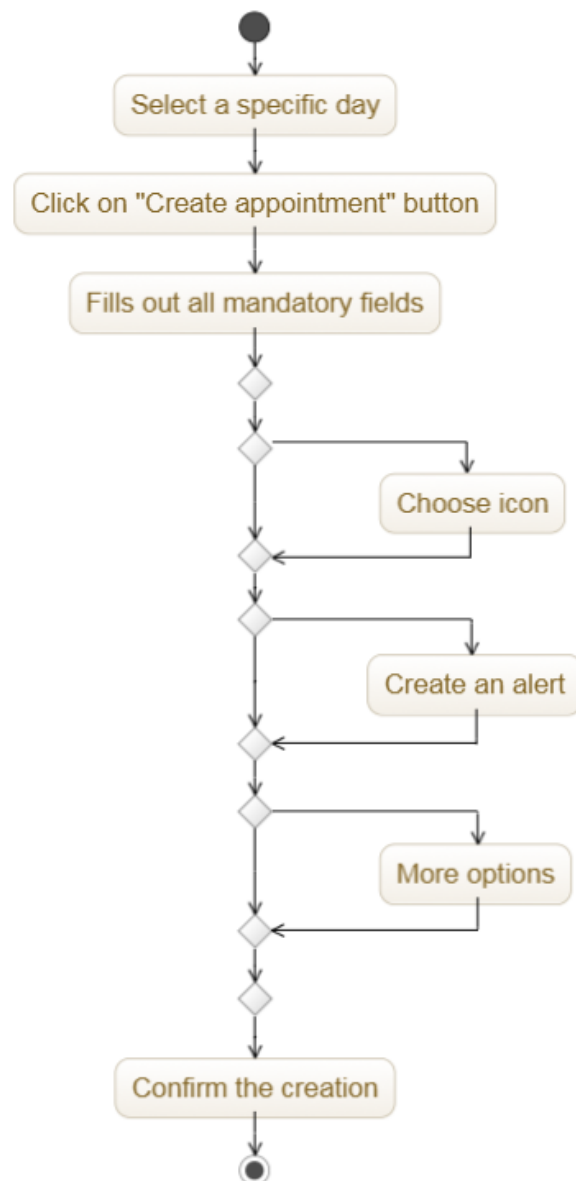


## 5.2.9 View and edit movements

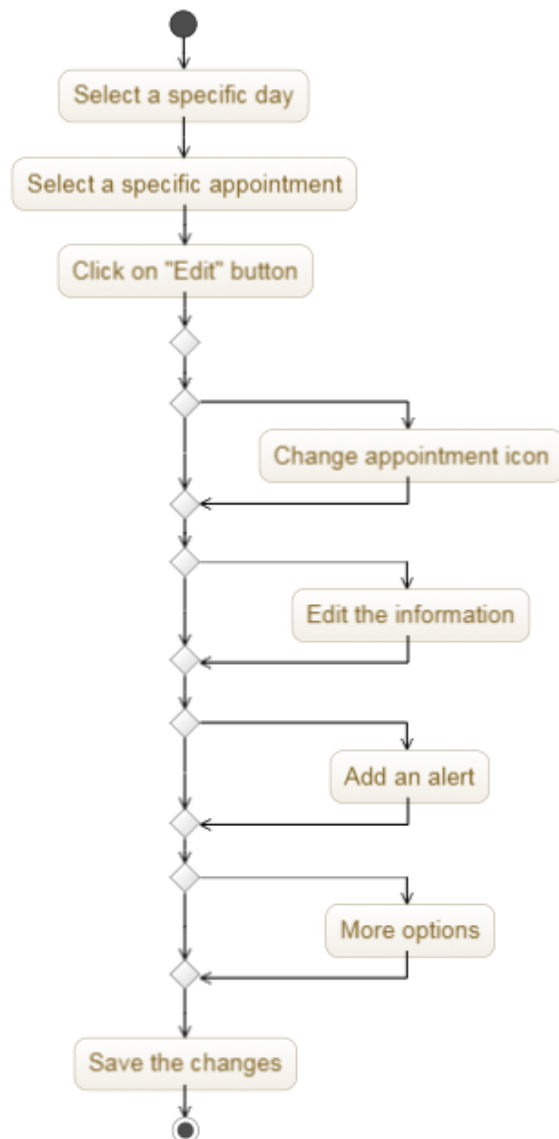


## 5.3 Activity diagram

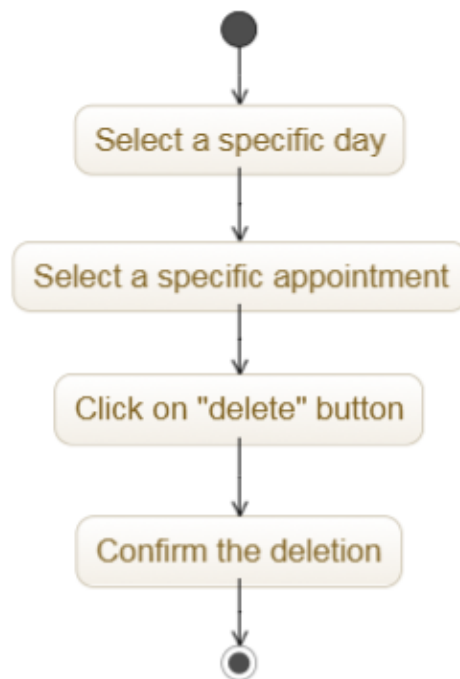
### 5.3.1 Create appointment



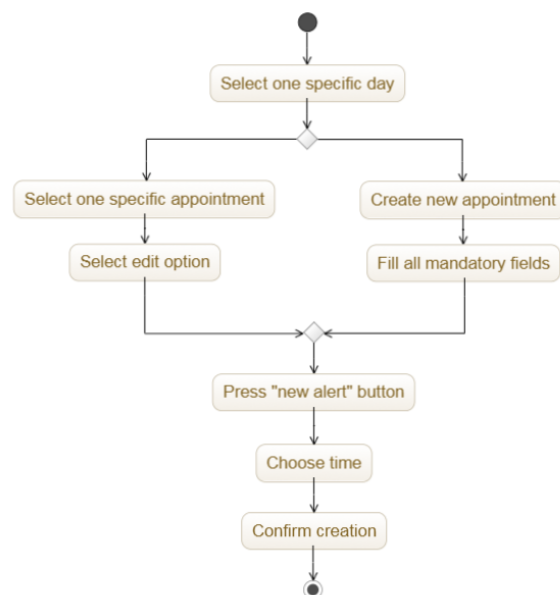
### 5.3.2 Edit appointment



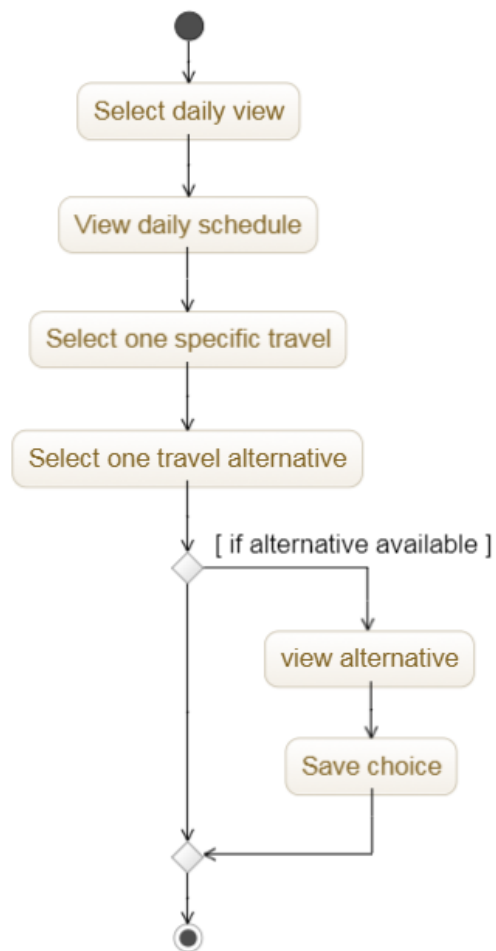
### 5.3.3 Delete appointment



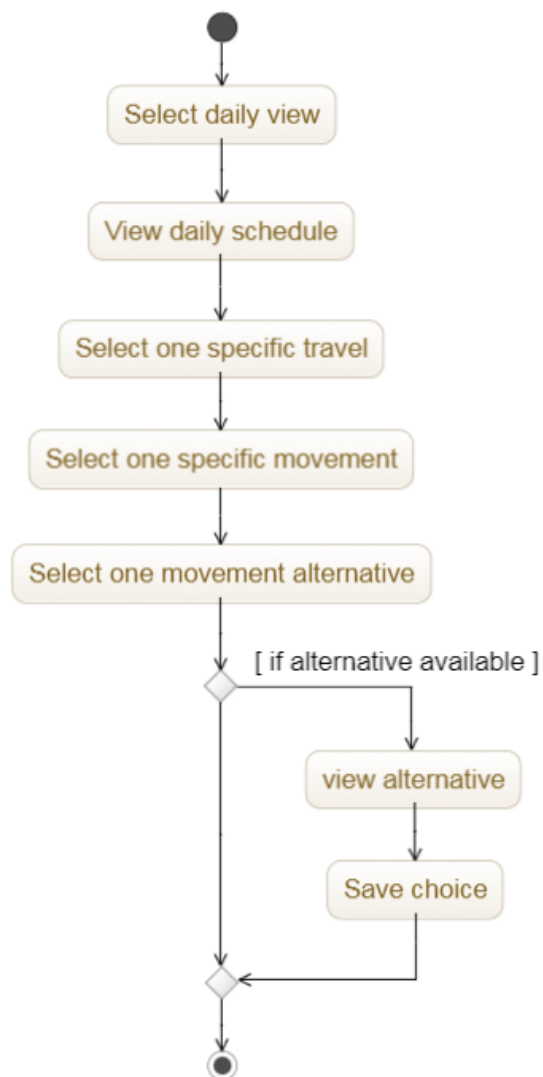
### 5.3.4 Create alert



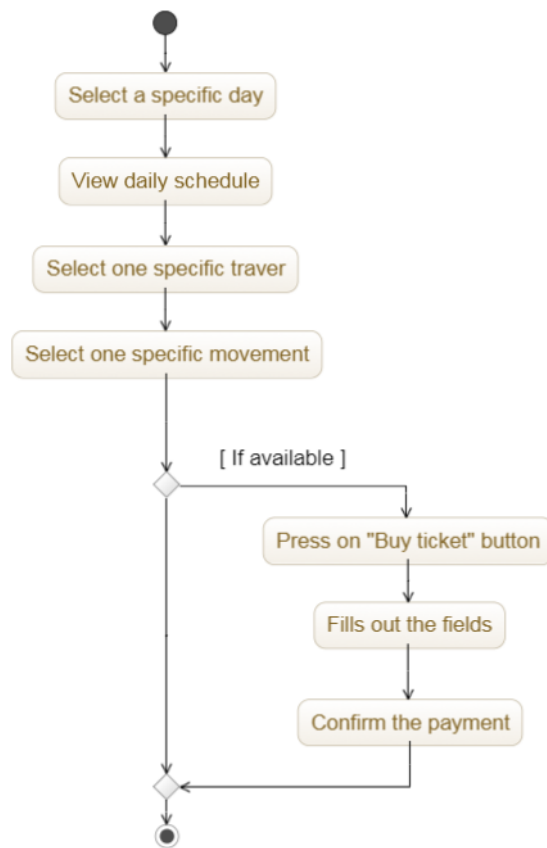
### 5.3.5 Edit travel



### 5.3.6 Movement alternative



### 5.3.7 Buy ticket



## 6 Formal Analysis Using Alloy

```
open util/integer

//-----MODEL:-----/

sig Name {}

sig Place {
  latitude: one Int,
  longitude: one Int,
  name: one Name,
  address: one Name
} {
  latitude > 0
  longitude > 0
}

sig Appointment {
  place: one Place,
  date: one Int,
  time: Int,
  duration: lone Int,
  alert: lone Alert,
  travel: one Travel
} {
  date > 0
  time > 0
  alert.appointment = this
  travel.placeOfArrival = place
}

sig Alert {
  appointment: one Appointment,
  date: one Int,
  time: one Int
} {
  date > 0
  time > 0
  appointment.alert = this
}

sig Calendar {
  appointm
  ents: some Appointment
}

sig User {
  calendar: one Calendar,
  email: one Name,
```



```
password: one Name
}
```

```
sig Movement {
  placeOfDeparture: one Place,
  placeOfArrival: one Place,
  extimatedTime: one Int,
} {
  placeOfDeparture != placeOfArrival
  extimatedTime > 0
}
```

```
sig Travel {
  placeOfDeparture: one Place,
  placeOfArrival: one Place,
  extimatedTime: one Int,
  movements: some Movement,
  alternatives: set Travel
} {
  placeOfDeparture != placeOfArrival
  extimatedTime > 0
}
```

```
//-----FACTS:-----//
```

```
//different users have different email addresses
fact mailUnique {
  no disjoint u1, u2: User | u1.email = u2.email
}
```

```
//different users must have different calendars
fact calendarUnique {
  no disjoint u1, u2: User | u1.calendar = u2.calendar
}
```

```
//no different places with same name, address or coordinates
fact placesAreDifferent {
  no disjoint p1, p2: Place | p1.name = p2.name or
  p1.address = p2.address or
  (p1.latitude = p2.latitude and p1.longitude = p2.longitude)
}
```

```
//a movement cannot exist without its travel
fact noMovementWithoutTravel {
  Travel.movements = Movement
}
```

```

/*
//every travel is related to an appointment or is an alternative to a travel rel
fact noTravelWithoutAppointment {
Travel = Travel.alternatives + Appointment.travel
}
*/

//an appointment cannot exist without its calendar
fact noAppointmentWithoutCalendar {
Calendar.appointments = Appointment
}

//a calendar cannot exist without its user
fact noCalendarWithoutUser {
User.calendar = Calendar
}

//the alert of an appointment cannot be scheduled after the beginning of the app
fact alertBeforeAppointment {
all a: Appointment | a.date < a.alert.date or a.time < a.alert.time
}

//travel alternatives depart from and lead to the same place
fact alternativesAreEquivalent {
all t: Travel | (all t1 : t.alternatives | t1.placeOfDeparture = t.placeOfDeparture)
}

//a travel cant be an alternative of itself
fact alternativesDontContainThemselves {
all t: Travel | (t not in t.alternatives)
}

//alternatives of a travel are also alternatives of the alternatives of the travel
fact alternativesAreSymmetrical {
all t: Travel | (all t1 : t.alternatives | t1.alternatives = t.alternatives + t.alternatives)
}

//every travel is composed by a sequence of connected movements
fact travelIsMadeByMovements {

//every travel starts with a movement
all t: Travel | (one m: t.movements | m.placeOfDeparture = t.placeOfDeparture)
//every travel ends with a movement
all t: Travel | (one m: t.movements | m.placeOfArrival = t.placeOfArrival)
}

```

```
//the ending of a movement is the beginning of a new one or the end of the travel
all t: Travel | (all m: t.movements | m.placeOfArrival = t.placeOfArrival
or
(one m1: t.movements | m1!=m and m.placeOfArrival = m1.placeOfDeparture))
//the beginning of a movement is the ending of an old one or the beginning of the travel
all t: Travel | (all m: t.movements | m.placeOfDeparture = t.placeOfDeparture
or
(one m1: t.movements | m1!=m and m.placeOfDeparture = m1.placeOfArrival ))
//different movements cannot start or end at the same position
all t: Travel | (no disjoint m1, m2: t.movements | m1.placeOfDeparture = m2.placeOfDeparture
or
m1.placeOfArrival = m2.placeOfArrival )
//no close path
all t: Travel | (no m: t.movements | m.placeOfArrival = t.placeOfDeparture)

}

fact travelTimeSumOfMovementsTime {
all t: Travel | t.estimatedTime = sum (t.movements.estimatedTime)

}

//-----ASSERTIONS:-----//

assert numberOfMovementsPerTravel {
all t: Travel | #t.movements = #t.movements.placeOfDeparture
}

run {} for 4 but exactly 2 Travel, exactly 0 Appointment, exactly 2 Movement
```

## **7 Effort Spent**

Provide here information about how much effort each group member spent in working at this document. We would appreciate details here.

## References