POLITECNICO
MILANO 1863

Travlendar+ project Andrea Mafessoni,
Andrea Mazzeo, Daniele Moltisanti

Requirement Analysis and Specification Document

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

Contents

List of Figures

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

- **RASD**: Requirement Analysis and Specification Document.
- **API**: Application Programming Interface.

1.3.3 abbreviations

1.4 Revision history

1.5 Reference documents

1.6 Document structure

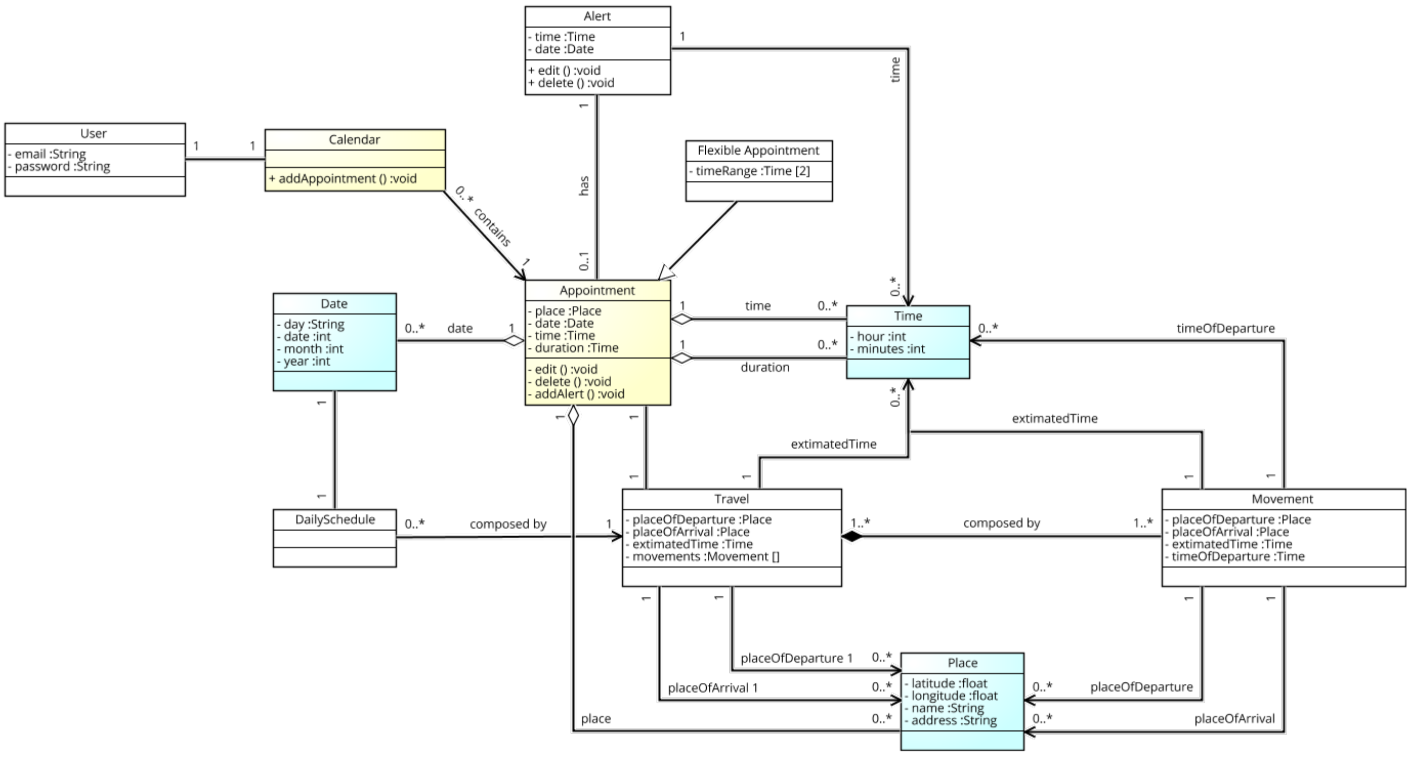
This RASD is composed by 5 parts and an appendix:

1. The first part of RASD document is an introduction to the problem. The base information needed to understand the project scope are given in this section.
2. The second part consists of an overall description of the system. Are described the characteristics that reguard the user, and all the application boundaries.
3. The third part is composed by the specific requirements identified, both functional and non functional.
4. In the fourth part a list of eight scenarios is provided; each of them describes a particular situation with the system might have to cope with.
5. The fifth part is entirely composed by the UML diagrams that model the system in details.
6. Sixth part is embodied with the Alloy model of the system and includes all the relevant details; a proof of consistency and an example of the generated world are also provided.

2 Overall Description

2.1 Product perspective

The product we will develope is a software application for smartphone implementing Android as operative system. The app will require the user to be registered in order to access to all the funtionalities. It will also need a stable internet connection in order to connect to different external system providing information about weather and transport means. The application runs locally on the smartphone but users data are also stored in the the system database.



2.2 Product functions

This application aims to provide a smart and user-friendly appointment-manager system, which allows the user to create and keep track of several appointment on a personal virtual calendar. Plus, the app is able to schedule the travels to reach the different appointment destinations on time in the fastest and most confortable way, according to the preferences expressed by the user and the different weather and public transport condition. Finally, the application offers the possibility of easily buy tickets for the different trips, and to view them when needed.

2.3 User characteristics

We recommend this application to a everyone who wants to easily manage his appointments without any loss of time. The app is designed to be as user friendly as possible, so the user is not forced to have a specific knowledge of the system to be able to benefit of the service. It can be used by a business man with a calendar full of work meetings or just by an ordinary student to schedule his study breaks. Furthermore, the possibility of buying tickets suits even simple travelers who need to manage their trips.

2.4 Dependencies

- The application requires a stable internet connection, by Wi-Fi or mobile network.
- The application makes use of GPS localization to access to the user position.
- Based on the user choice, the application may require a connection to an existing Facebook or Google account.
- The application makes use of external APIs to achieve information about maps, public transport and weather condition.
- The application depends entirely on the ATM and Trenord public transport Payment Service for everything concerning the tickets purchase.

2.5 Constrains

- The application requires the user to be registred and logged into the system to work properly.
- The application works only on smartphones that implement Android v. 4.0.3 or later.
- The application is in alpha version and works only inside the urban area of Milan.
- 30 Mb(?) of storage memory must be available on the device in order to correctly install the application.

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interface

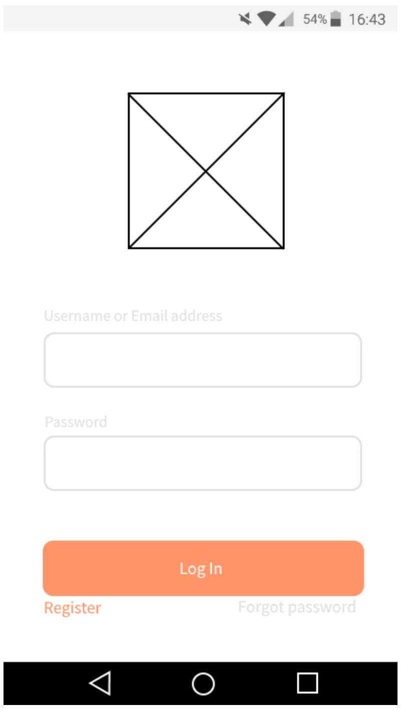


Figure 1: Login page

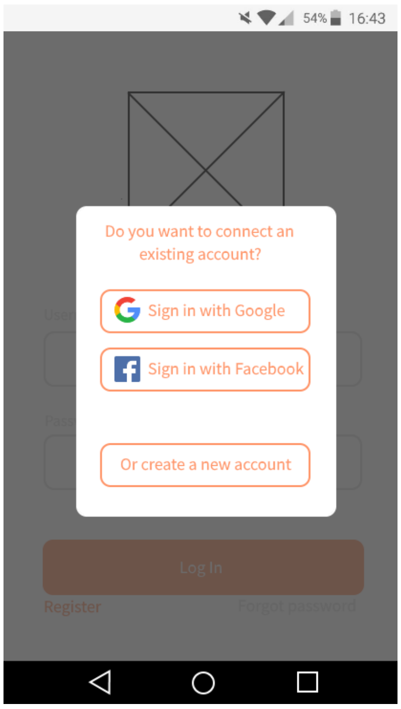


Figure 2: Connection of an existent account

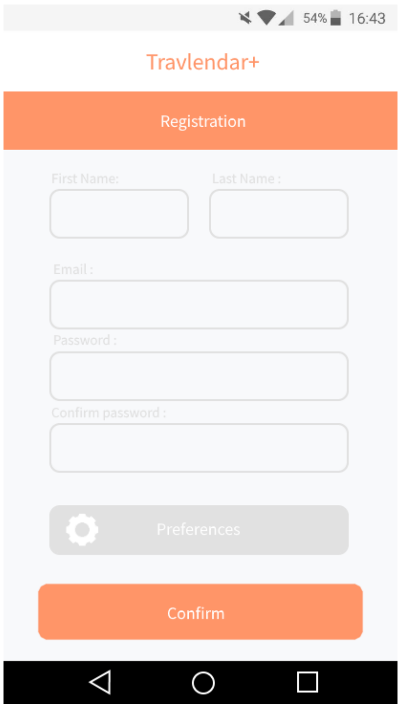


Figure 3: Registration form

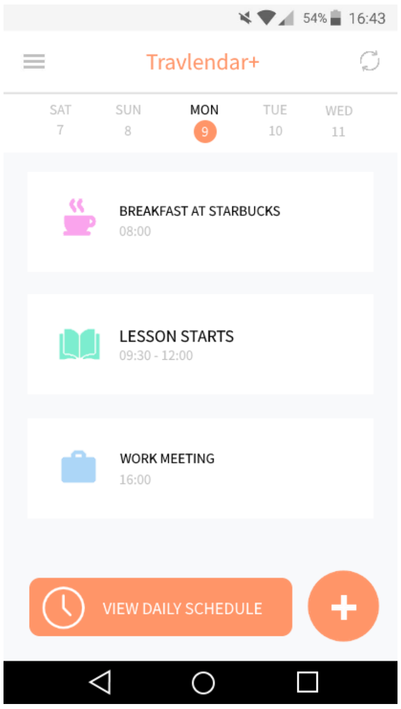


Figure 4: Daily view of calendar

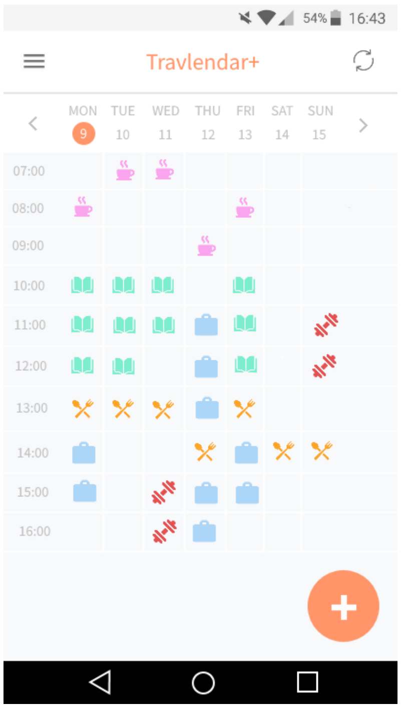


Figure 5: Weekly view of calendar

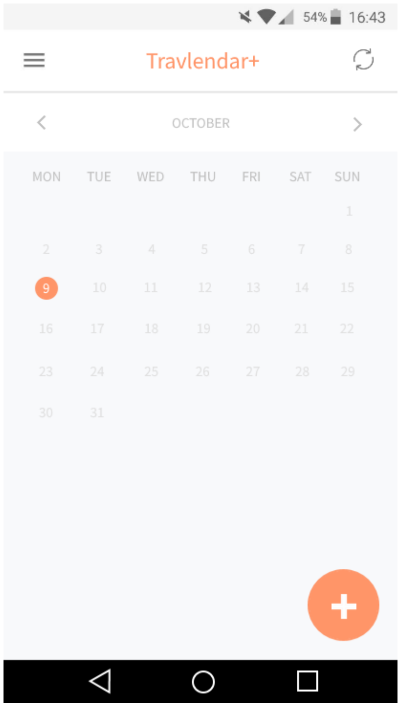


Figure 6: Monthly view of calendar

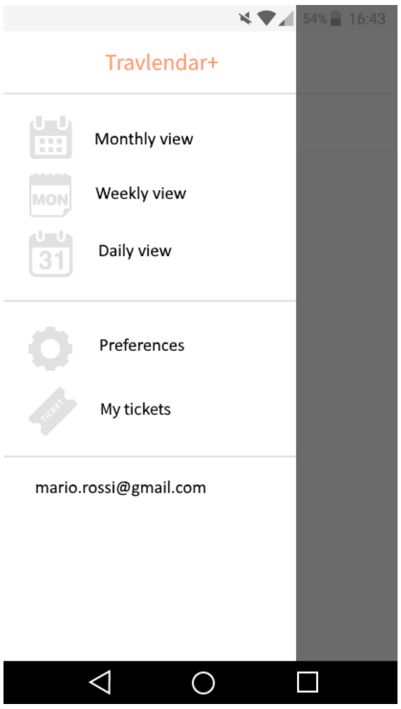


Figure 7: Menu

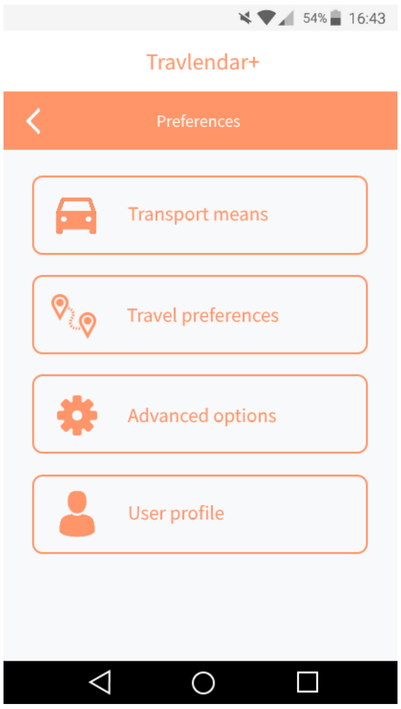
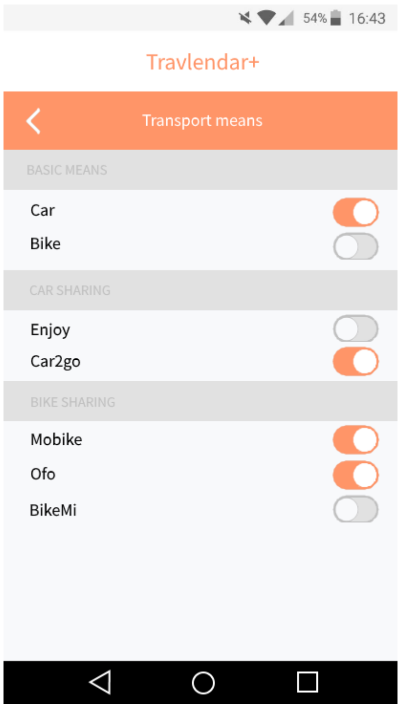


Figure 8: Preferences menu


54% 16:43

Travlendar+

Transport means

BASIC MEANS

- Car [✓]
- Bike [ ]

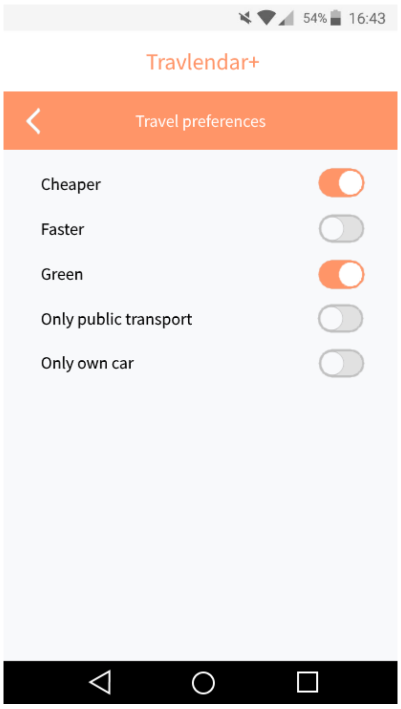
CAR SHARING

- Enjoy [ ]
- Car2go [✓]

BIKE SHARING

- Mobike [✓]
- Ofo [✓]
- BikeMi [ ]

Figure 9: Transport means settings

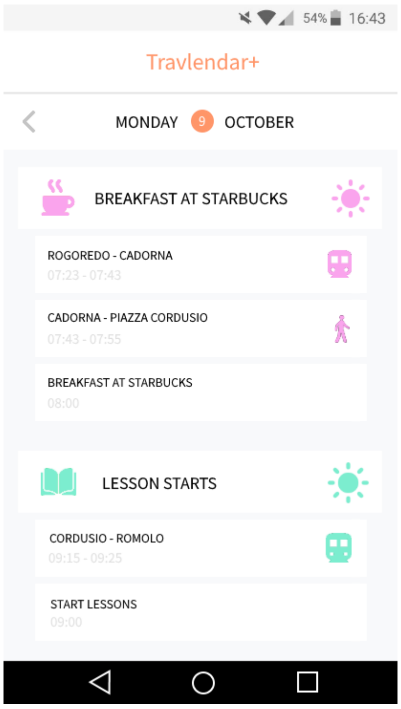

54% 16:43

Travlendar+

Travel preferences

- Cheaper [✓]
- Faster [ ]
- Green [✓]
- Only public transport [ ]
- Only own car [ ]

Figure 10: Travel preferences



54% 16:43

Travlendar+

MONDAY 9 OCTOBER

BREAKFAST AT STARBUCKS

ROGOREDO - CADORNA
07:23 - 07:43

CADORNA - PIAZZA CORDUSIO
07:43 - 07:55

BREAKFAST AT STARBUCKS
08:00

LESSON STARTS

CORDUSIO - ROMOLO
09:15 - 09:25

START LESSONS
09:00

Figure 11: Daily schedule

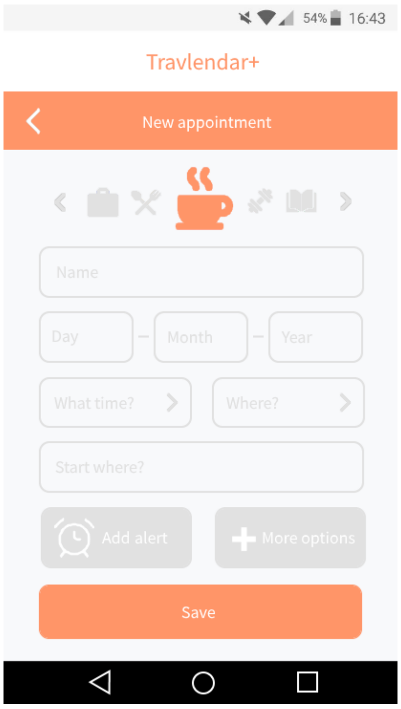


Figure 12: New appointment

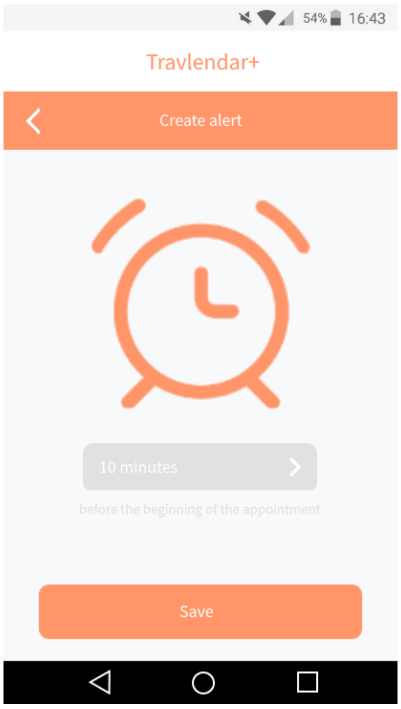
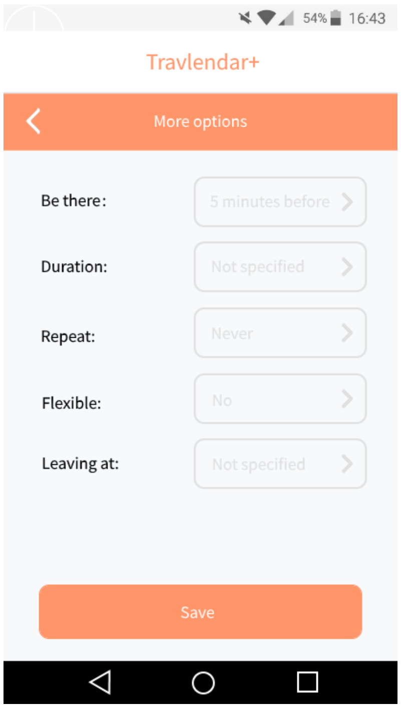


Figure 13: New alert



Be there: 5 minutes before
Duration: Not specified
Repeat: Never
Flexible: No
Leaving at: Not specified

Figure 14: appointment options

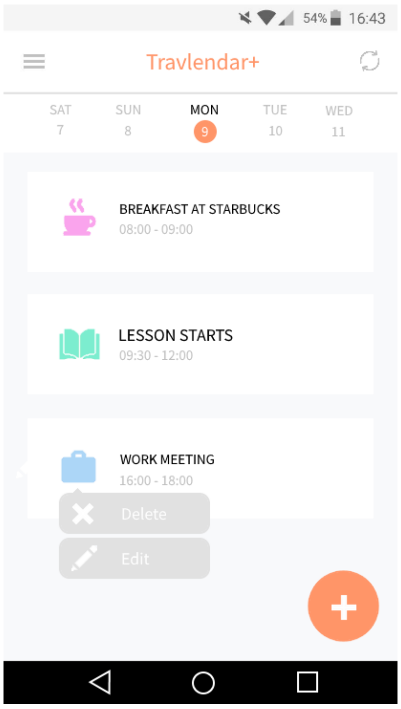


Figure 15: Edit or delete appointment

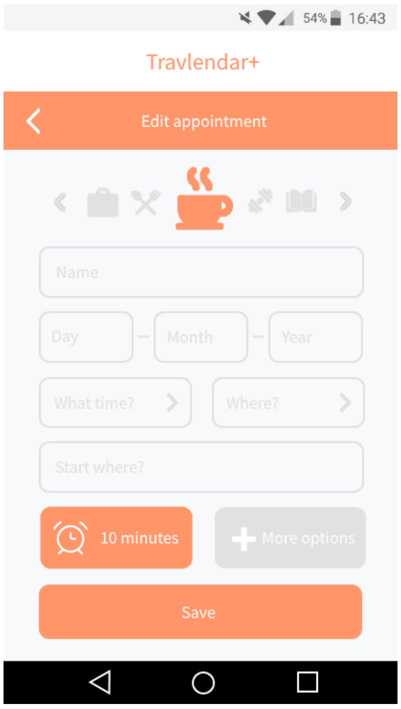


Figure 16: Edit appointment

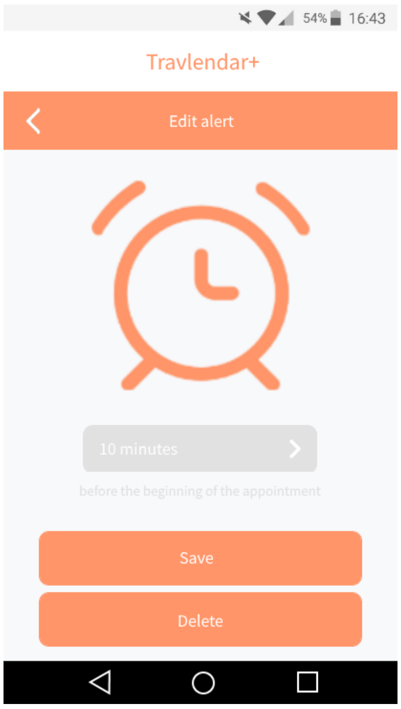


Figure 17: Edit or delete alert

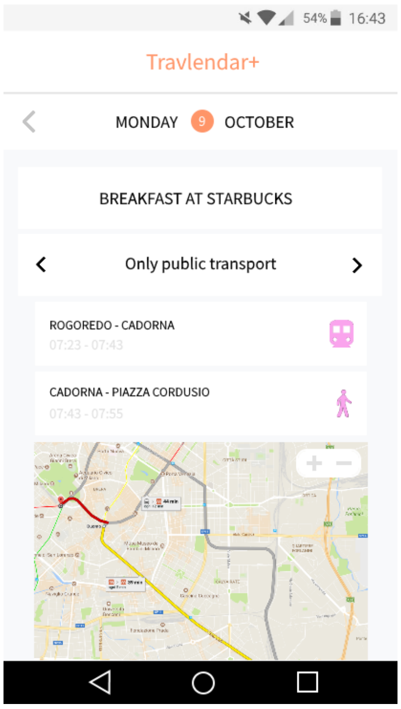


Figure 18: Travel details

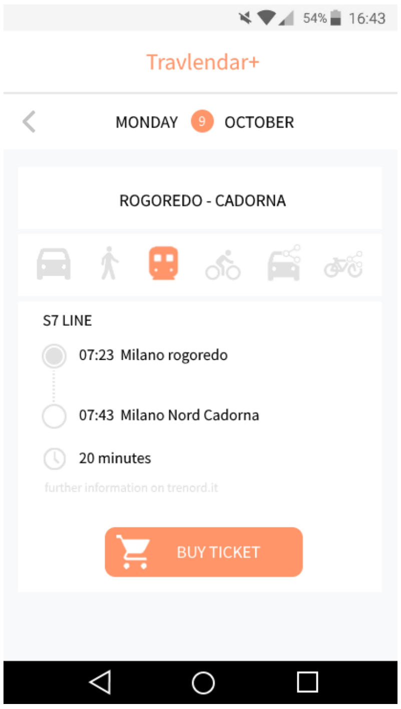


Figure 19: Movement details

3.1.2 Software interface

The application makes uses of the following APIs:

- 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 connetting 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 trough 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 confortable 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

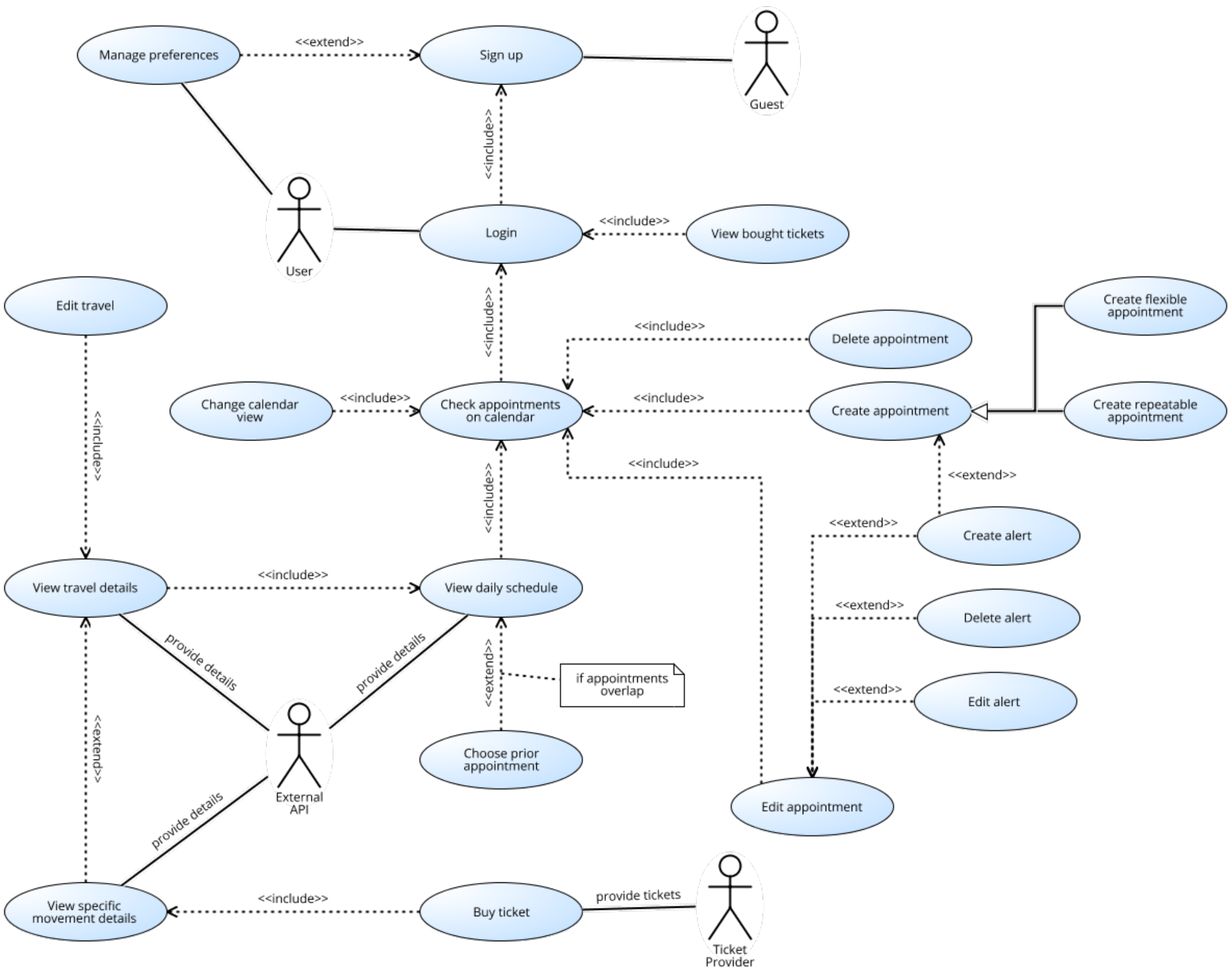


Figure 20: Use case diagram

5.1.1 Sign up

Name	**Sign up**
Actors	Guest
Entry conditions	The guest is on the log in page of the application and clicks on "Register" button.
Flow of events	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. (a) If the guest connects his account, the system accepts the request and creates a new Travlendar+ account based on provided account. (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. 2. The guest fills out all the mandatory fields. 3. (Optional) The guest adjusts the preferences settings. 4. The guest clicks on button "Confirm". 5. The system checks data provided and eventually creates and registers the user account.
Exit conditions	The guest has successfully created a new account and he can log into the system with his credentials.
Exceptions	• 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. • Data provided are incorrect. The system highlights the incorrect fields and asks the user to repeat the procedure.

5.1.2 Log in

Name	**Log in**
Actors	User
Entry conditions	The user launches the application and clicks on the “Log In” button.
Flow of events	1. The user inserts his email. 2. The user inserts his password. 3. The user clicks on the “Log In” button.
Exit conditions	The login procedure is successfully completed. The user is logged into the system and is able to access to all the functionalities.
Exceptions	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

Name	**Manage preferences**
Actors	User
Entry conditions	The user clicks on “Preferences” from the side menu on the homepage.
Flow of events	1. The system shows the preferences settings, that include transport means owned, favorite kind of travel and other advanced options. 2. The user accesses to the desired settings and adjusts the preferences as wanted. 3. The user saves the changes.
Exit conditions	The changes are saved and remembered from the system. The preferences have been updated.
Exceptions	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

Name	**View daily schedule**
Actors	User, External APIs
Entry conditions	The user clicks on the “View daily schedule” button while checking an appointment on his calendar.
Flow of events	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. 2. The system provides a graphic overview of all the travels scheduled for the day, with the relative movements. 3. The user checks all the information needed and eventually clicks on a specific travel or movement to get further information.
Exit conditions	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.
Exceptions	• 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. • 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.

5.1.5 View travel details

Name	**View travel details**
Actors	User, External APIs
Entry conditions	The user is checking his daily schedule and clicks on a specific travel.
Flow of events	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. 2. The user is redirected to a new page that displays all the information about the selected travel. 3. The user checks all the information needed and eventually clicks on a specific movement to get further information or buy a ticket.
Exit conditions	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.
Exceptions	No exceptions expected

5.1.6 Edit travel

Name	**Edit travel**
Actors	User
Entry conditions	The user is checking a specific travel.
Flow of events	1. The user provides modifications to the travel as followed: (a) Clicking on the travel preferences and switching to a different alternative. (b) Clicking on a specific movement and changing the transport mean. 2. The user saves the changes.
Exit conditions	The changes are saved and the system displays the new modified travel.
Exceptions	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

Name	**View specific movement details**
Actors	User
Entry conditions	User is checking a specific travel and select a specific movement.
Flow of events	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. 2. The user is redirected to a new page that displays all the information about the selected movement. 3. The user checks all the information needed and eventually proceeds in buying a ticket.
Exit conditions	User has obtained all the information needed and has clicked “back” to return to the travel page or has proceeded in buying a ticket.
Exceptions	No exceptions expected

5.1.8 Choose prior appointment

Name	**Choose prior appointment**
Actors	User
Entry conditions	The user is checking the schedule and two appointments overlap.
Flow of events	1. The system signals the user the exactly time overlapping of the appointments. 2. The user clicks on the preferred appointment. 3. The system re-computes the daily schedule giving priority to the chosen appointment.
Exit conditions	No more appointments overlap. The system shows the daily schedule without errors.
Exceptions	No exceptions expected

5.1.9 Buy ticket

Name	**Buy ticket**
Actors	User, Ticket provider
Entry conditions	User is checking a specific movement and clicks on “Buy ticket”
Flow of events	1. The system connects trough APIs to the external system of the ticket provider. 2. The system shows the user a form to fill with all the payment information. 3. The user fills out all the fields and confirm the payment. 4. The system sends information to the ticket provider and waits for confirmation and ticket data.
Exit conditions	Payment is successfully fulfilled and bought tickets are available for the view. The user is brought back to the movement page.
Exceptions	Payment is rejected. The system notifies the user with a warning. It is possible to repeat the procedure.

5.1.10 Delete appointment

Name	**Delete appointment**
Actors	User
Entry conditions	The user selects a specific appointment in his calendar (through daily or weekly view) and clicks on "Delete" button.
Flow of events	1. A pop-up shows up, asking the user to confirm the deletion. 2. The user confirms the deletion. 3. The system removes all the appointment information from the memory, alert included.
Exit conditions	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.
Exceptions	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

Name	**Create appointment**
Actors	User
Entry conditions	The user is checking his calendar and clicks on "Create appointment" button.
Flow of events	1. The user is redirected to the appointment creation page, which contains all the fields required to perform the creation. 2. The user fills out all mandatory fields. The following steps aren't mandatory: (a) User chooses among the available icons. (b) User adds an alert to remember the appointment. (c) User clicks on "More options" and provides more detailed options. 3. User confirms the creation. 4. The system saves the appointment information.
Exit conditions	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.
Exceptions	• The user has provided incorrect information: the appointment is not created and the user must repeat the procedure. • 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. • The appointment overlaps with other previously created appointments. The appointment is created and inserted anyway, but the user receives a notification of the overlapping.

5.1.12 Edit appointment

Name	**Edit appointment**
Actors	User
Entry conditions	The user selects a specific appointment in his calendar (through daily or weekly view) and clicks on “Edit” button.
Flow of events	1. The user is redirected to the appointment editing page which contains all the previously inserted information. 2. The user can perform the following actions: (a) Changing the appointment icon. (b) Editing the information. (c) Adding an alert to the appointment. (d) Clicking on “More options” and providing more detailed options. 3. The user saves the changes. 4. The system saves the updated appointment information.
Exit conditions	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.
Exceptions	• The user has provided incorrect information: the changes are not saved and the user must repeat the procedure. • 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. • The appointment now overlaps with other previously created appointments. The changes as saved anyway, but the user receives a notification of the overlapping.

5.1.13 Create flexible appointment

Name	**Create flexible appointment**
Actors	User
Entry conditions	The user is creating/editing an appointment, clicks on "More options" and clicks on the "Flexible" field.
Flow of events	1. A pop-up shows up, containing the fields to be filled. 2. The user fills out the fields, specifying the time range of the appointment. 3. The user confirms the choice.
Exit conditions	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.
Exceptions	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

Name	**Create repeatable appointment**
Actors	User
Entry conditions	The user is creating/editing an appointment, clicks on "More options" and clicks on the "Flexible" field.
Flow of events	1. A pop-up shows up, containing the fields to be filled. 2. The user fills out the fields, specifying the days in which the appointment is wanted to be created. 3. The user confirms the choice.
Exit conditions	The choice is saved for later and the user is back on the More option page. The appointment is now a repeatable appointment.
Exceptions	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

Name	**Create alert**
Actors	User
Entry conditions	The user is on the appointment creation/editing page and clicks on the “Add alert” button.
Flow of events	1. The user is redirected to the alert creation/editing page. 2. The user fills the field specifying the time of the alert. 3. The user confirms the creation. 4. The system saves the alert information.
Exit conditions	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.
Exceptions	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

Name	**Edit alert**
Actors	User
Entry conditions	The user is on the appointment creation/editing page and clicks on the “Edit alert” button.
Flow of events	1. The user is redirected to the alert creation/editing page. 2. The user replaces needed information with updated ones. 3. The user saves the changes. 4. The system saves the alert information.
Exit conditions	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.
Exceptions	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

Name	**Delete alert**
Actors	User
Entry conditions	The user is on the appointment creation/editing page and clicks on the "Edit alert" button.
Flow of events	1. The user is redirected to the alert creation/editing page. 2. The user clicks on "Delete". 3. The user confirms the deletion. 4. The system removes all the alert information from the memory.
Exit conditions	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.
Exceptions	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

Name	**Check appointments on calendar**
Actors	User
Entry conditions	The user is logged in and he is on the homepage of the application.
Flow of events	1. The system shows an overview of the existing appointments. 2. The user moves between the appointments and collects the needed infomation.
Exit conditions	The user has collected the needed information and moves to a different page.
Exceptions	No exception expected.

5.1.19 Change calendar view

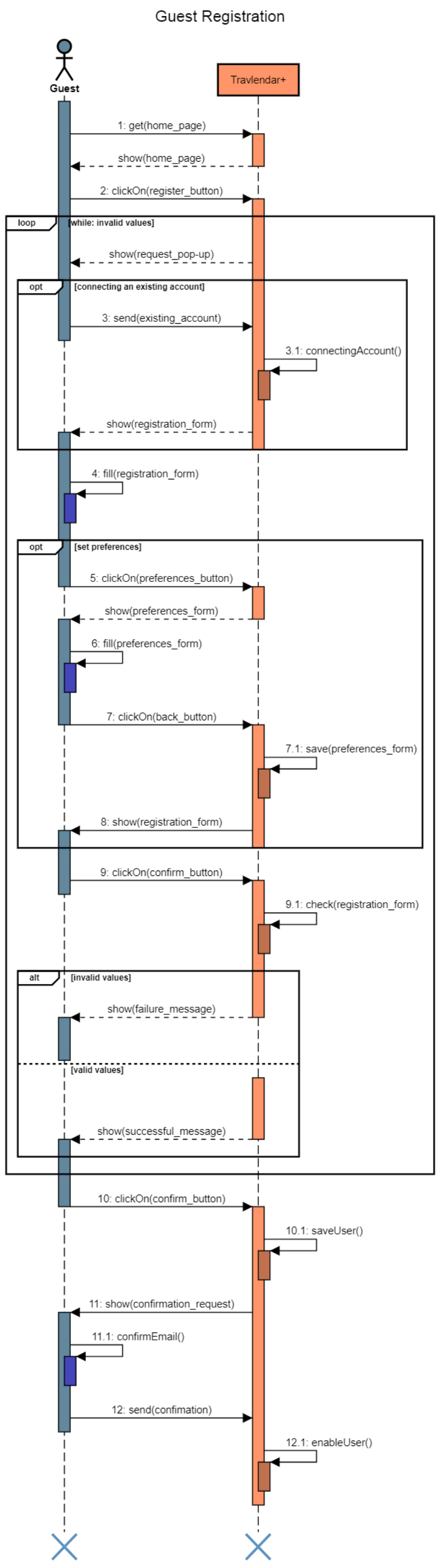
Name	**Change calendar view**
Actors	User
Entry conditions	The user is on the homepage of the application and he is checking his appointments.
Flow of events	1. The user opens the lateral menu. 2. The user clicks on one of available view (daily, weekly or monthly). 3. The system changes the layout according to the user's choice.
Exit conditions	The layout is changed and the user is back on the calendar view.
Exceptions	No exception expected.

5.1.20 View bought tickets

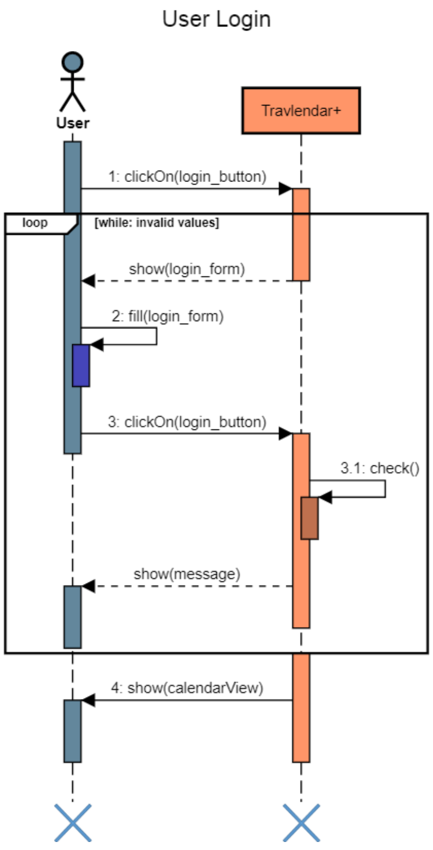
Name	**View bought tickets**
Actors	User
Entry conditions	The user is logged into the system and clicks on "My tickets" button on the side menu.
Flow of events	1. The system loads the tickets saved in memory and displays the list to the user. 2. The user chooses a specific ticket and clicks on it. 3. The system displays the full screen ticket.
Exit conditions	The user has checked the needed tickets and clicks "back" to return to the homepage.
Exceptions	No tickets saved in memory. The system notifies the absence of saved tickets to the user.

5.2 Sequence diagram

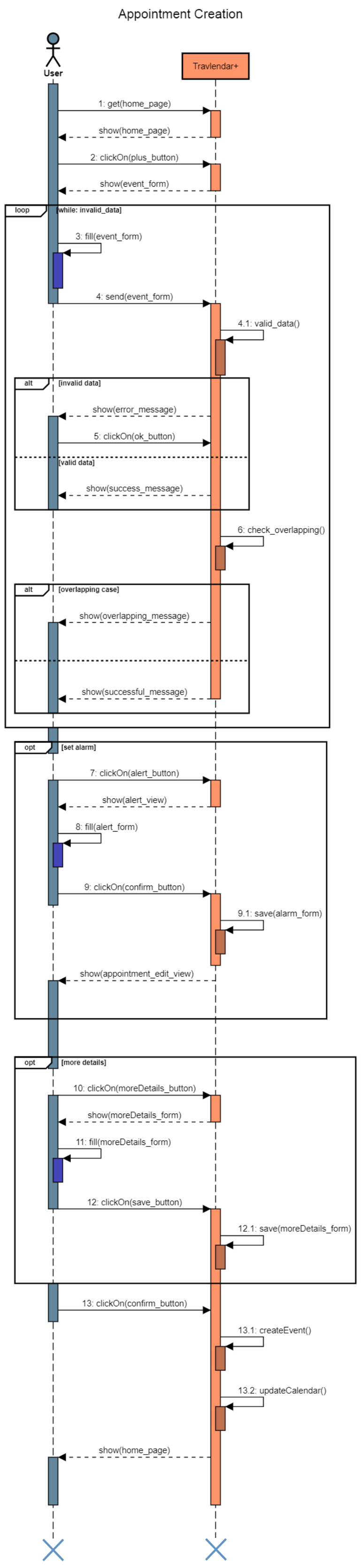
5.2.1 Guest registration



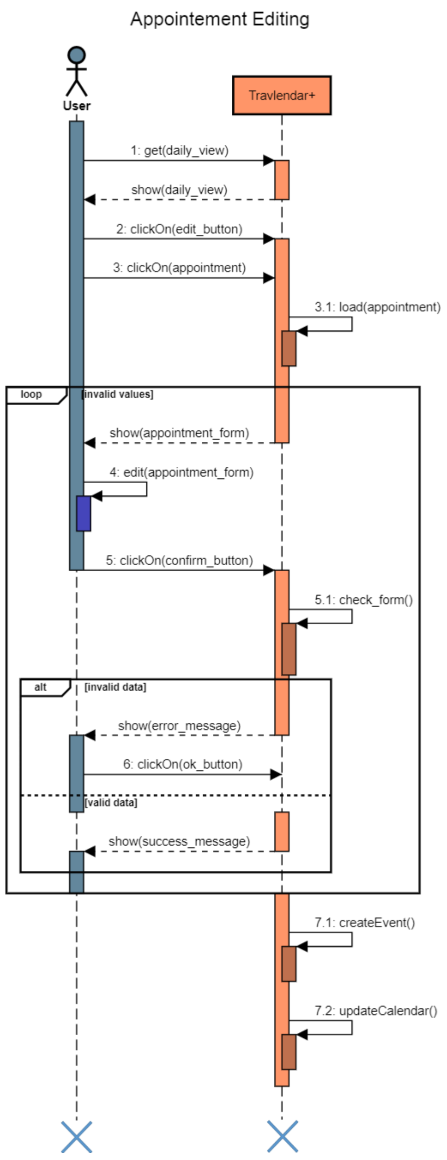
5.2.2 User Login



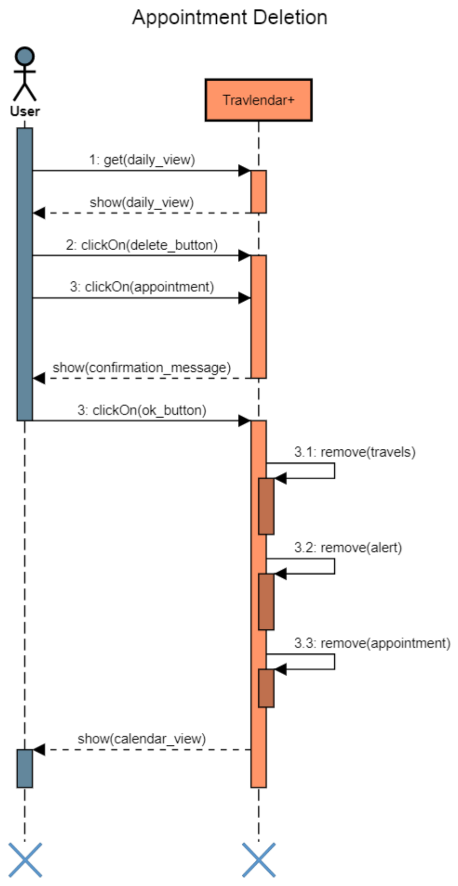
5.2.3 Appointment creation



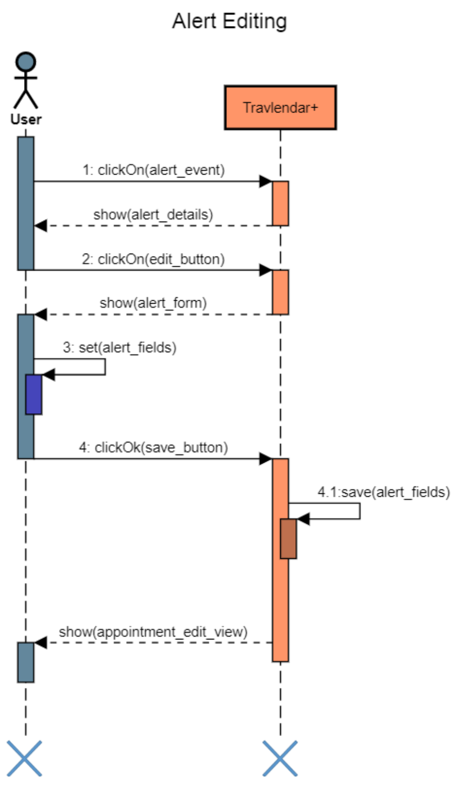
5.2.4 Appointment editing



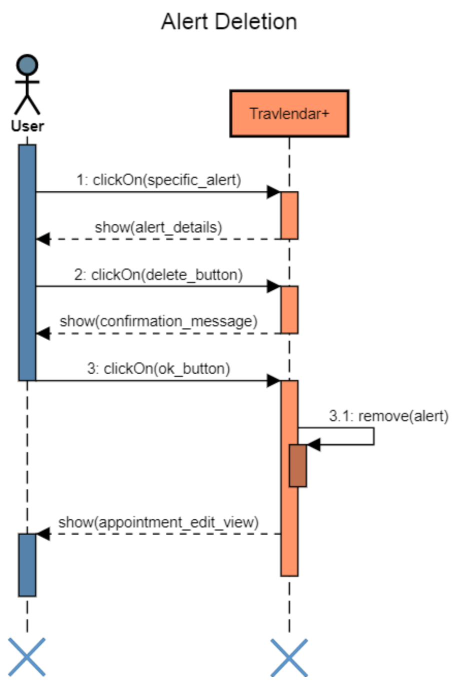
5.2.5 Appointment deletion



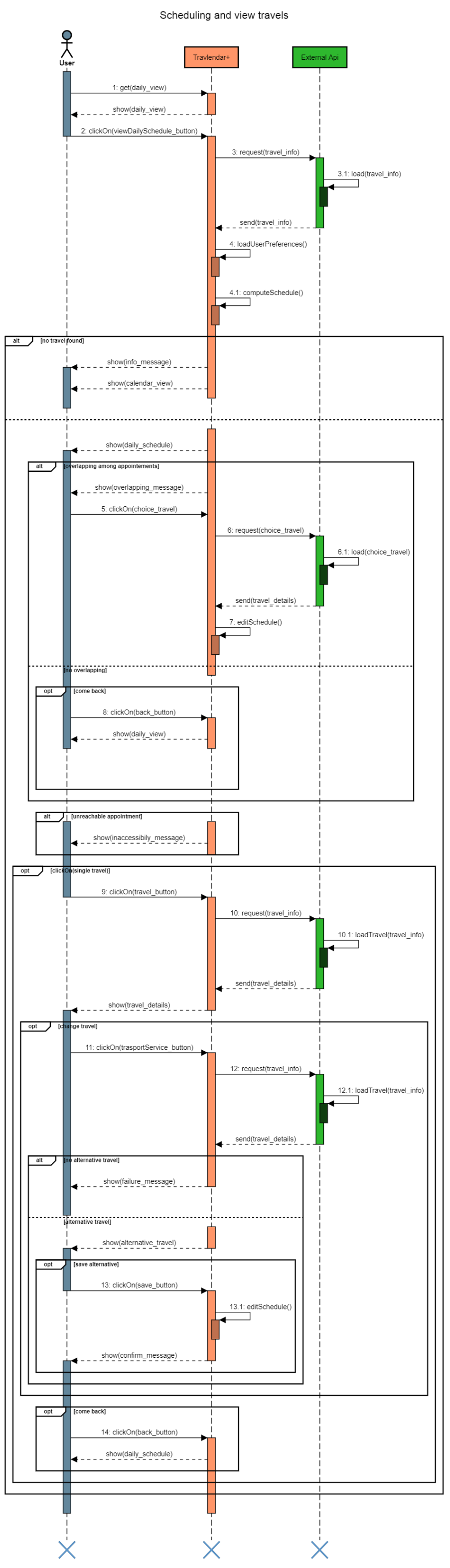
5.2.6 Alert editing



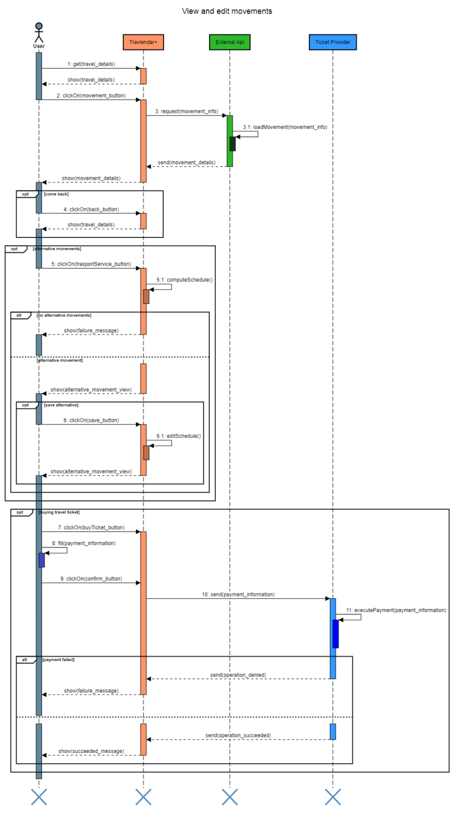
5.2.7 Alert deletion


Alert Deletion
User
Travlendar+
1: clickOn(specific_alert)
show(alert_details)
2: clickOn(delete_button)
show(confirmation_message)
3: clickOn(ok_button)
3.1: remove(alert)
show(appointment_edit_view)

5.2.8 Scheduling and view travels

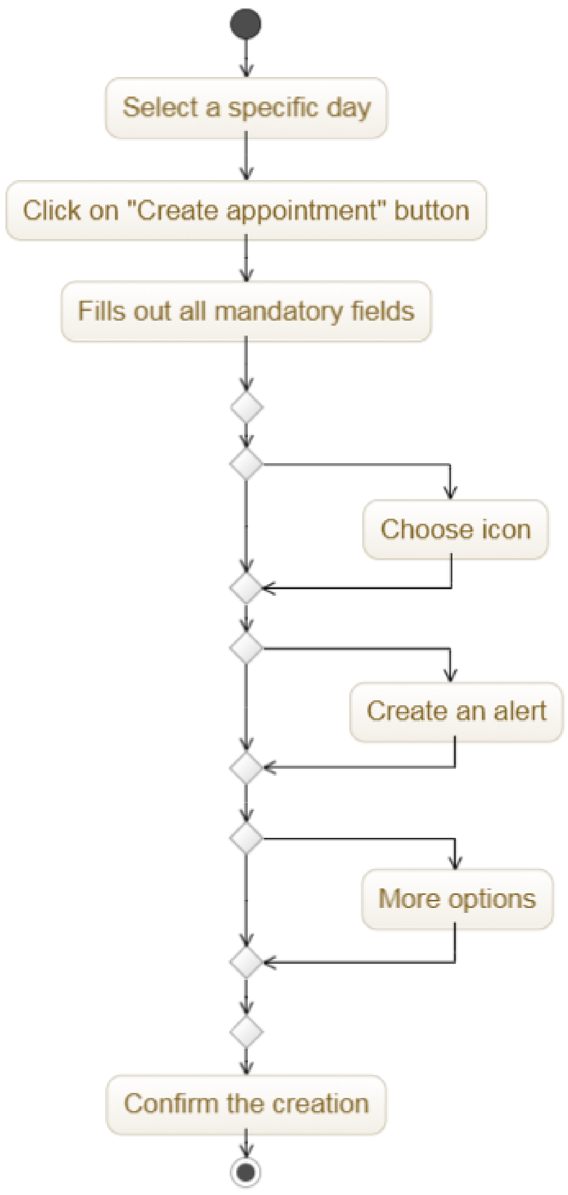


5.2.9 View and edit movements

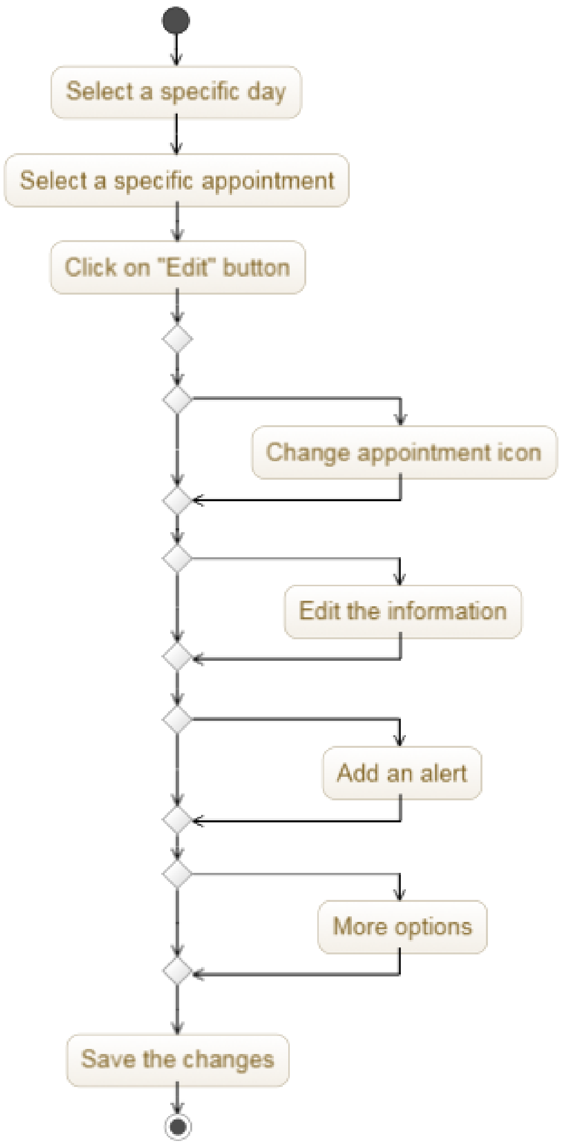


5.3 Activity diagram

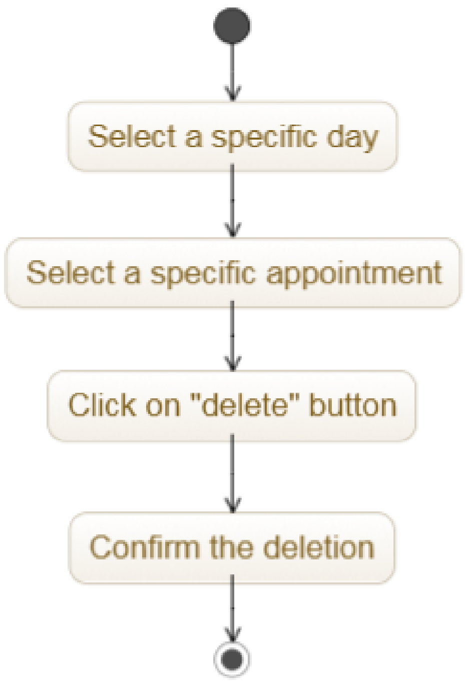
5.3.1 Create appointment



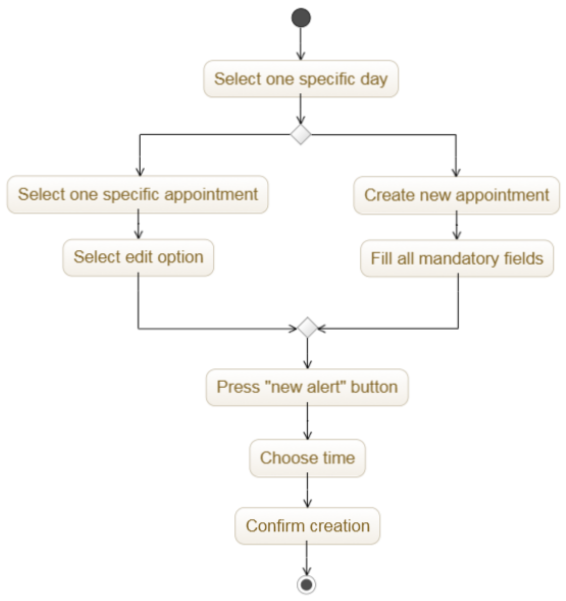
5.3.2 Edit appointment



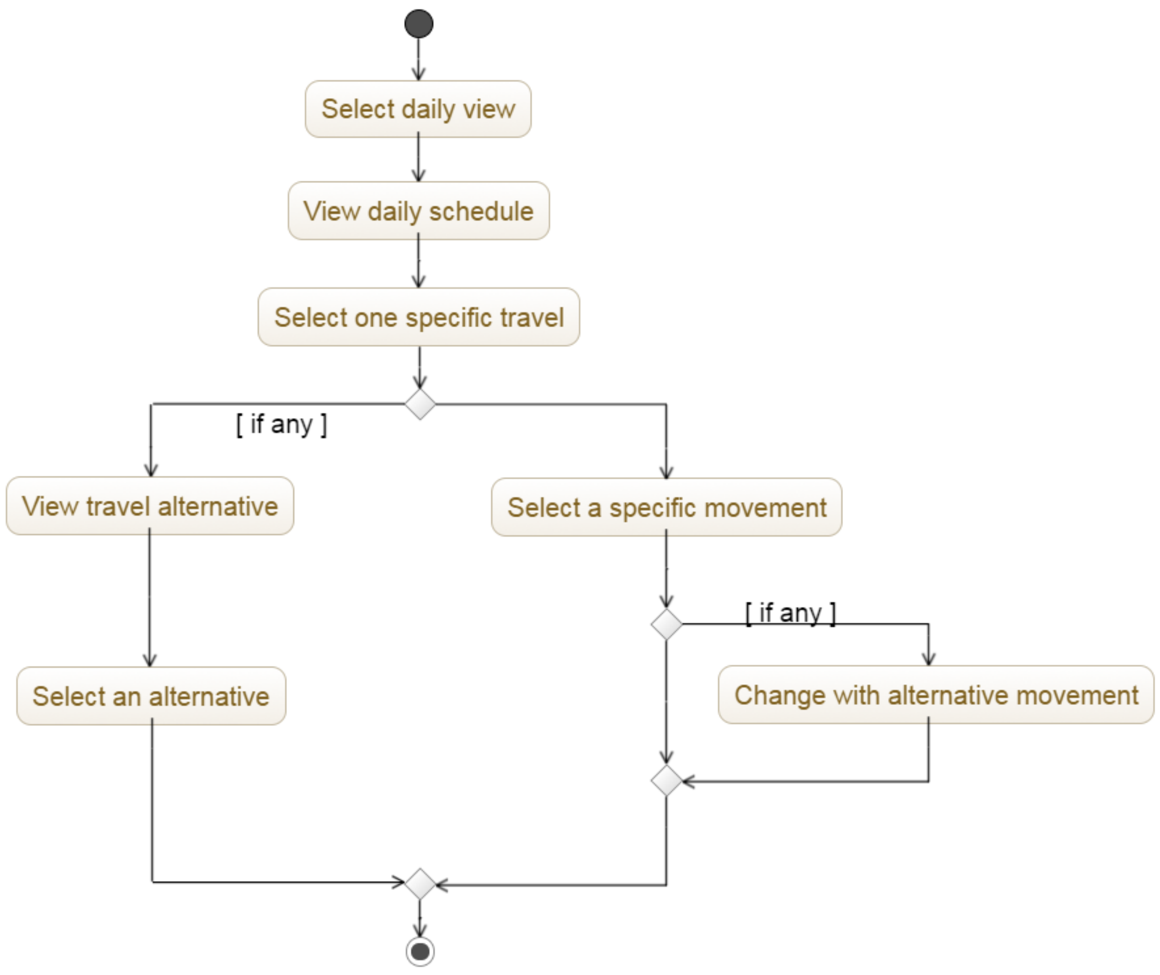
5.3.3 Delete appointment



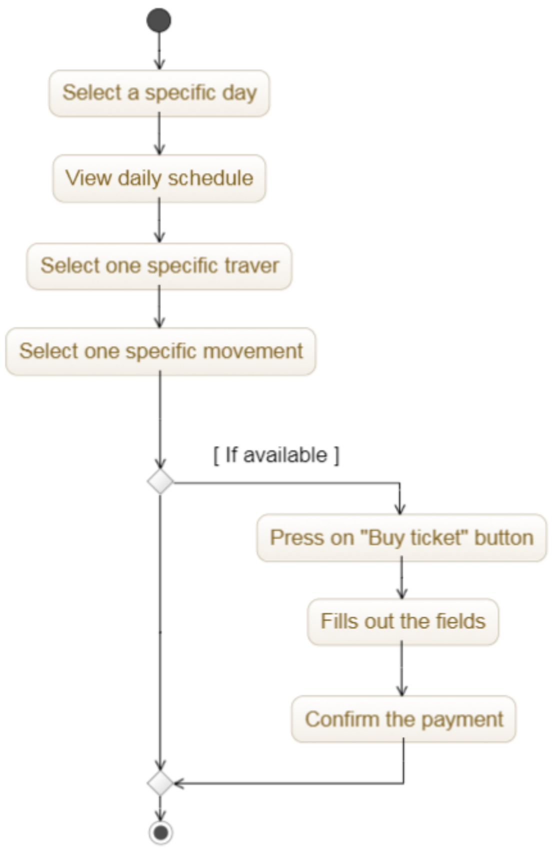
5.3.4 Create alert



5.3.5 Edit travel



5.3.6 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.placeOfDepart
}


//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 trav
fact alternativesAreSymmetrical {
all t: Travel | (all t1 : t.alternatives | t1.alternatives = t.alternatives +t −
}


//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 trave
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 th
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.plac
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.extimatedTime = sum (t.movements.extimatedTime)

}
```

//———————–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

7.1 Hours of work

7.1.1 Andrea Mafessoni

Task name	Start date	End date	Hours(h)	Day
RASD	**02/10/2017**	**28/10/2017**	**34.5**	
Thinking about project ideas	02/10/2017	10/10/2017	5	9
3. Specific Requirements (Functional requirements)	05/10/2017	10/10/2017	5	6
4. Scenarios	11/10/2017	14/10/2017	3.5	4
5.1 Uml modelling (Use case)	08/10/2017	20/10/2017	4	13
5.1 Uml modelling (Class diagram)	10/10/2017	20/10/2017	5	11
6. Alloy	14/10/2017	22/10/2017	7	9
Final revision	25/10/2017	28/10/2017	5	9

7.1.2 Andrea Mazzeo

Task name	Start date	End date	Hours(h)	Day
RASD	**02/10/2017**	**28/10/2017**	**35**	
Thinking about project ideas	02/10/2017	10/10/2017	5	9
1. Introduction	04/10/2017	5/10/2017	2.5	1
3. Specific Requirements (User Interface)	06/10/2017	15/10/2017	10	10
3. Specific Requirements (Functional requirements)	07/10/2017	08/10/2017	2.5	2
3.4 Software system attributes	14/10/2017	14/10/2017	1	1
5.3 Activity diagram	10/10/2017	22/10/2017	2	3
5.1 Uml modelling (Use case)	10/10/2017	10/10/2017	2	1
Latex	23/10/2017	28/10/2017	5	6
Final revision	25/10/2017	28/10/2017	5	9

7.1.3 Daniele Moltisanti

Task name	Start date	End date	Hours(h)	Day
RASD	**02/10/2017**	**28/10/2017**	**33.5**	
Thinking about project ideas	02/10/2017	10/10/2017	5	9
2. Overall Description	04/10/2017	06/10/2017	2	2
3. Specific Requirements (Functional requirements)	7/10/2017	10/10/2017	3.5	1
5.1 Uml modelling (Use case)	08/10/2017	15/10/2017	5	8
5.2 Sequence diagram	16/10/2017	24/10/2017	12	9
3.4 Software system attributes	14/10/2017	15/10/2017	1	2
Final revision	25/10/2017	28/10/2017	5	9