Sommario

[1. Introduction 5](#_Toc496446160)

[1.1. Purpose 5](#_Toc496446161)

[1.2. Scope 5](#_Toc496446162)

[1.3. Definition, Acronyms, Abbreviations 5](#_Toc496446163)

[1.4. Revision history 5](#_Toc496446164)

[1.5. Reference documents 5](#_Toc496446165)

[1.6. Document structure 5](#_Toc496446166)

[2. Overall Description 6](#_Toc496446167)

[2.1. Product perspective 6](#_Toc496446168)

[2.2. Product functions 6](#_Toc496446169)

[2.3. User characteristics 6](#_Toc496446170)

[2.4. Domain Assumption and Dependencies 6](#_Toc496446171)

[2.5. Constrains 6](#_Toc496446172)

[3. Specific requirements 7](#_Toc496446173)

[3.1. External interface requirements 7](#_Toc496446174)

[3.1.1. User interfaces 7](#_Toc496446175)

[3.1.2. Software interfaces 14](#_Toc496446176)

[3.2. Functional requirements 14](#_Toc496446177)

[3.2.1. [G1] Allow a Guest to create a registered Travlendar+ account. 14](#_Toc496446178)

[3.2.2. [G2] Allow an User to log in into his Travlendar+ account. 14](#_Toc496446179)

[3.2.3. [G3] Allow an User to create a new appointment in his calendar. 14](#_Toc496446180)

[3.2.4. [G4] Allow an User to delete an existing appointment from his calendar. 14](#_Toc496446181)

[3.2.5. [G5] Allow an User to edit an existing appointment in his calendar. 15](#_Toc496446182)

[3.2.6. [G6] Allow an User to view his appointments. 15](#_Toc496446183)

[3.2.7. [G7] Allow an User to view his Daily Schedule 15](#_Toc496446184)

[3.2.8. [G8] Allow an User to navigate and choose between different travel alternatives. 16](#_Toc496446185)

[3.2.9. [G9] Allow an User to manage alerts for each appointment. 16](#_Toc496446186)

[3.2.10. [G10] Allow an User to manage his travel preferences. 16](#_Toc496446187)

[3.2.11. [G11] Allow an User to buy public transportation tickets. 16](#_Toc496446188)

[3.3. Design constraints 17](#_Toc496446189)

[3.3.1. Standards compliance 17](#_Toc496446190)

[3.3.2. Hardware limitations 17](#_Toc496446191)

[3.4. Software System Attributes 17](#_Toc496446192)

[3.4.1. Reliability 17](#_Toc496446193)

[3.4.2. Availability 17](#_Toc496446194)

[3.4.3. Security 17](#_Toc496446195)

[3.4.4. Maintainability 17](#_Toc496446196)

[3.4.5. Portability 17](#_Toc496446197)

[4. Scenarios 18](#_Toc496446198)

[4.1 Scenario 1 18](#_Toc496446199)

[4.2 Scenario 2 18](#_Toc496446200)

[4.3 Scenario 3 18](#_Toc496446201)

[4.4 Scenario 4 18](#_Toc496446202)

[5. UML modelling 19](#_Toc496446203)

[5.1. Use case diagram 19](#_Toc496446204)

[5.1.1. Sign up 19](#_Toc496446205)

[5.1.2. Log in 19](#_Toc496446206)

[5.1.3. Manage preferences 20](#_Toc496446207)

[5.1.4. View calendar 20](#_Toc496446208)

[5.1.5. Change view calendar 21](#_Toc496446209)

[5.1.6. View daily schedule 21](#_Toc496446210)

[5.1.7. View travel details 21](#_Toc496446211)

[5.1.8. View travel alternatives 22](#_Toc496446212)

[5.1.9. Choose travel alternative 22](#_Toc496446213)

[5.1.10. View movement details 22](#_Toc496446214)

[5.1.11. View movement alternatives 23](#_Toc496446215)

[5.1.12. Choose movement alternative 23](#_Toc496446216)

[5.1.13. Buy travel ticket 24](#_Toc496446217)

[5.1.14. Delete appointment 24](#_Toc496446218)

[5.1.15. Create appointment 24](#_Toc496446219)

[5.1.16. Edit appointment 25](#_Toc496446220)

[5.1.17. Create flexible appointment 25](#_Toc496446221)

[5.1.18. Create repeatable appointment 26](#_Toc496446222)

[5.1.19. Create alert 26](#_Toc496446223)

[5.1.20. Delete alert 27](#_Toc496446224)

[5.1.21. Edit alert 27](#_Toc496446225)

[5.2. Class diagram 28](#_Toc496446226)

[5.3. Sequence diagrams 28](#_Toc496446227)

[5.4. Activity diagram 28](#_Toc496446228)

[6. Formal analysis using alloy 29](#_Toc496446229)

[7. Effort spent 30](#_Toc496446230)

[8. References 31](#_Toc496446231)

# 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.

Ecc…

## 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

## 1.4. 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

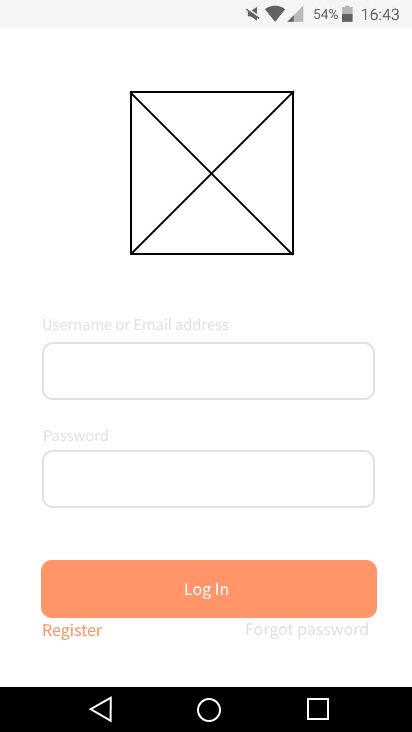
• 30 Mb(?) of storage memory available on own devise to be installed

# 3. Specific requirements

## 3.1. External interface requirements

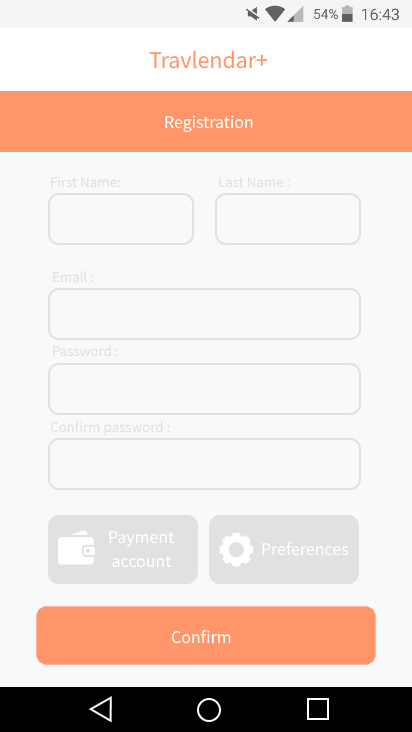
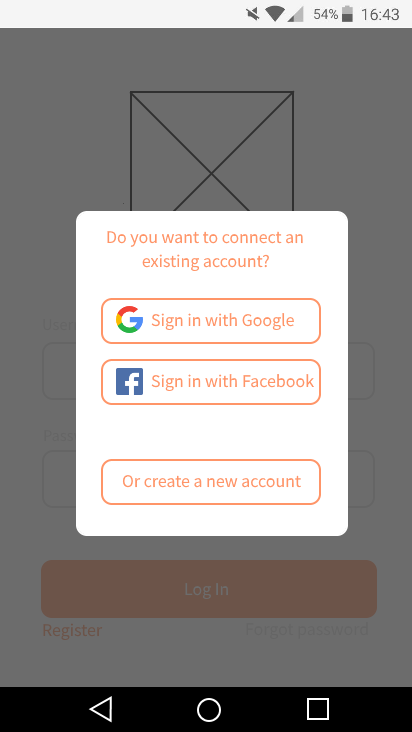
### 3.1.1. User interfaces

#### 3.1.1.1. Login



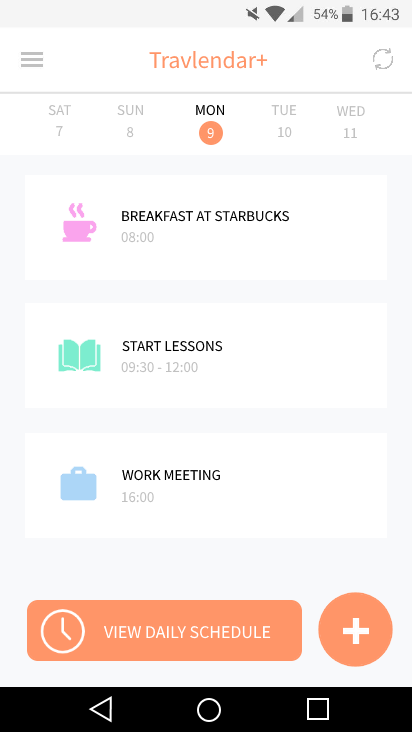
#### 3.1.1.2. Registration

#### 

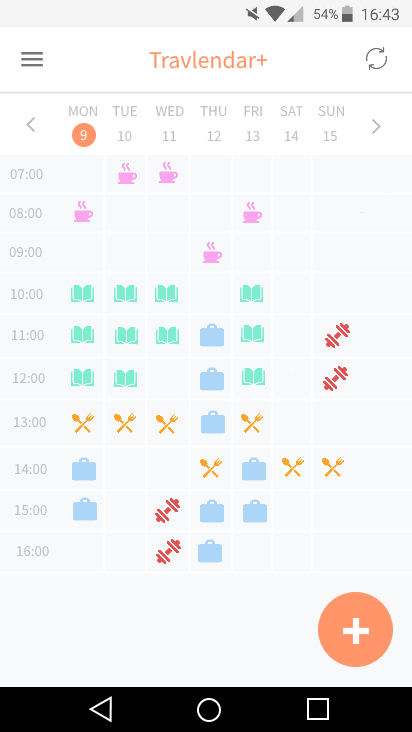


#### 3.1.1.3. Homepage

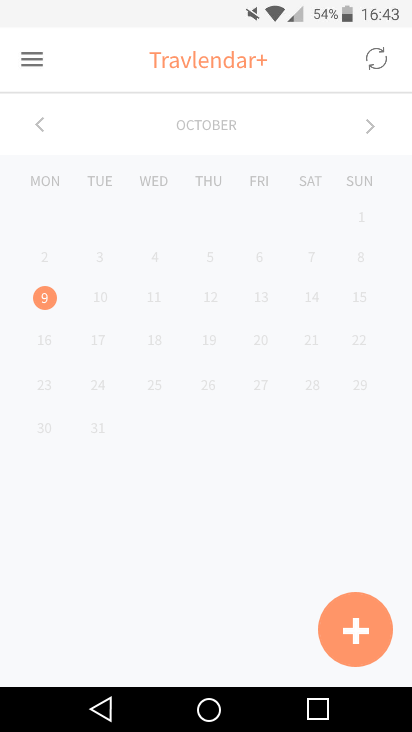
##### 3.1.1.3.1. Daily view



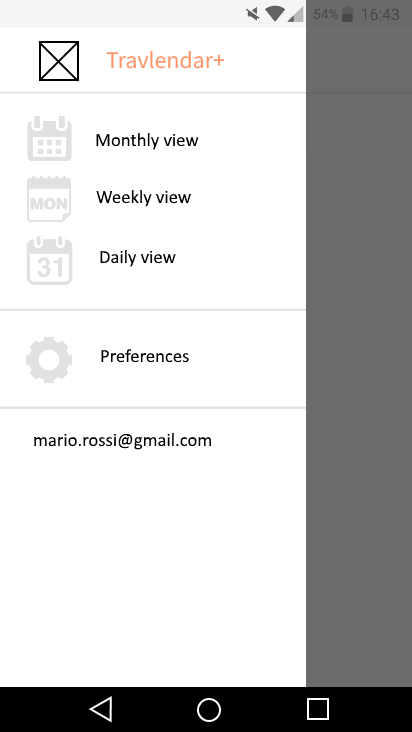
##### 3.1.1.3.1. Weekly view



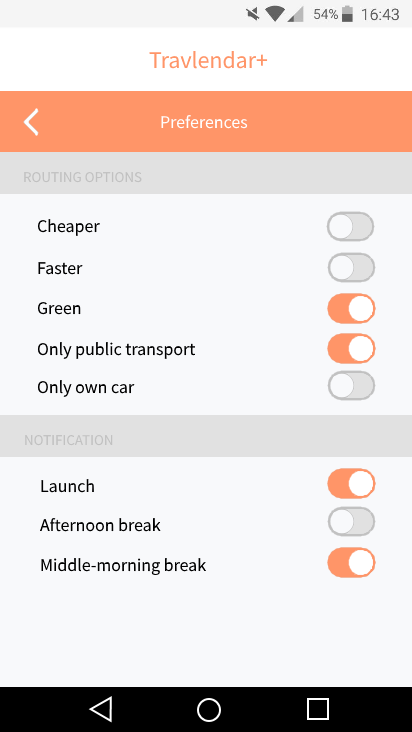
##### 3.1.1.3.1. Monthly view



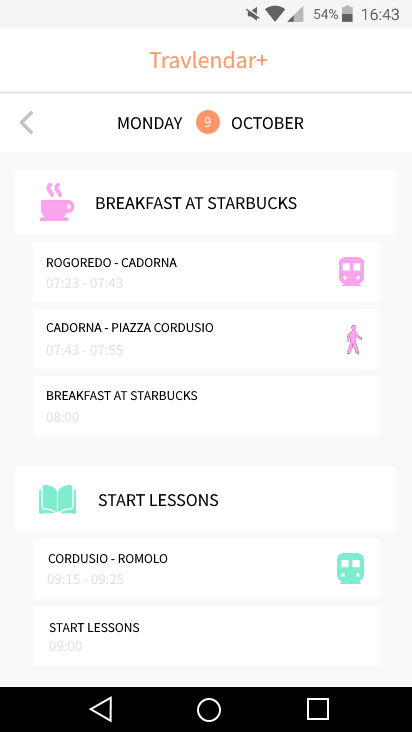
#### 3.1.1.4. Menu



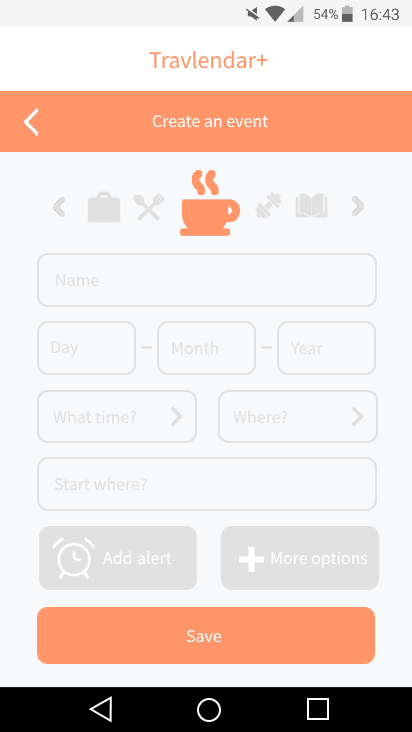
#### 3.1.1.5. Preferences



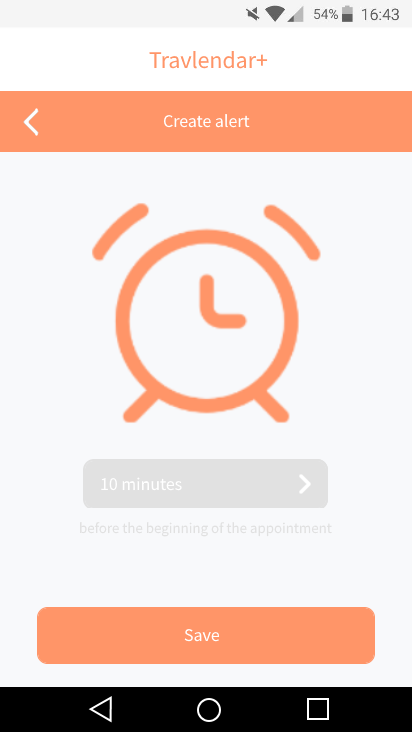
#### 3.1.1.6. Daily schedule



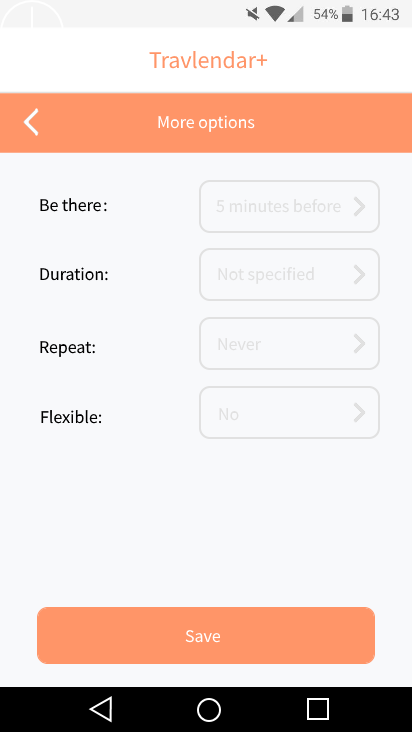
#### 3.1.1.7. Event creation



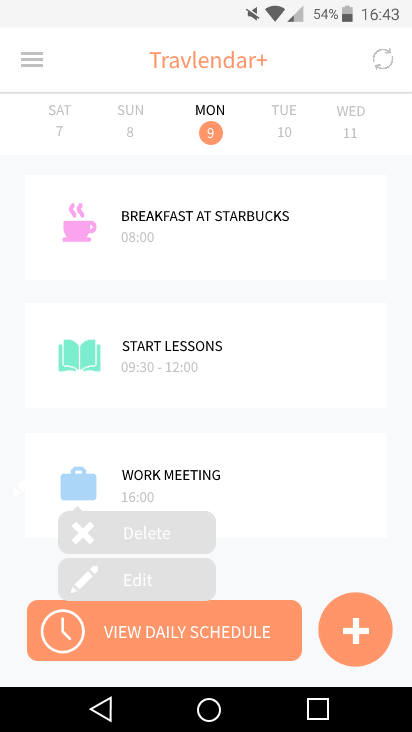
#### 3.1.1.8. Alarm creation



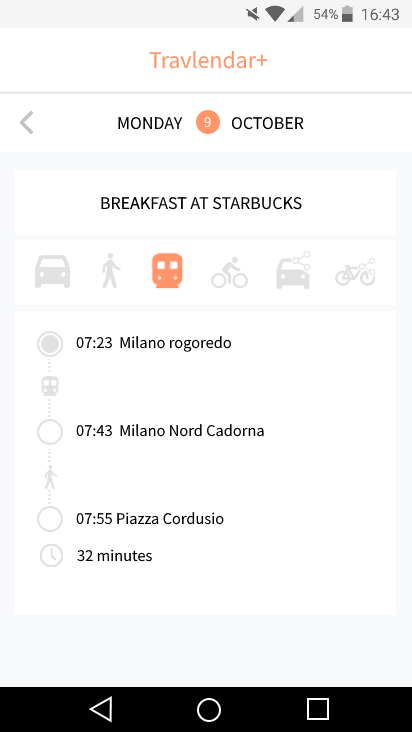
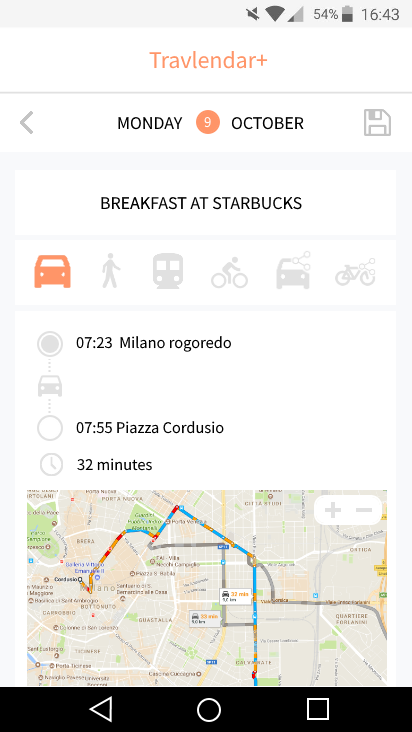
##### 3.1.1.9. Option panel



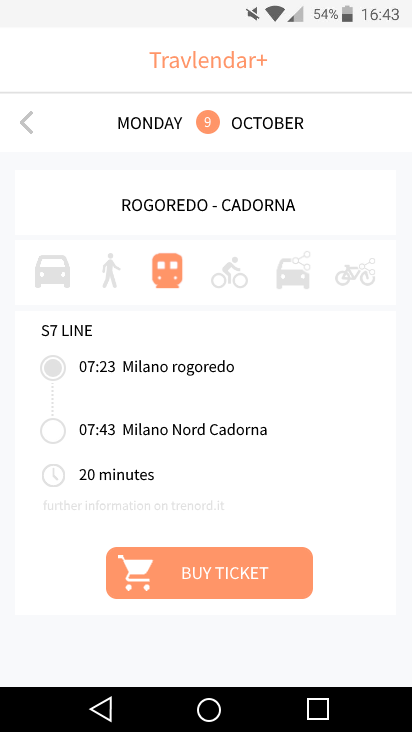
#### 3.1.1.10. Delete/Edit panel



#### 3.1.1.11. Travel details



#### 3.1.1.12. Movement details



### 3.1.2. Software interfaces

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.

* [R1] The system must ask the user to provide credentials to perform registration process.
* [R2] The system must require a valid email address and password or a link with an existing Facebook / Google account for the registration.
* [R3] The system must check that the provided email/account for the registration is not already in use from a different Travlendar+ account.
* [D1] Data used for the registration must be formally correct.

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

* [R5] The user must be already registered to perform login process.
* [R6] The user must be able to insert his credentials to log into the system.
* [R7] The system must be able to check the correctness of the credentials provided by user for the login.
* [R8] The system must let the user log in only if data provided are correct.

### 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:

* Access to the calendar;
* Get his 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 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 popup is showed and ask to the guest if he want to connect an existing account, such as Google or Facebook, or if he want to create a new account.    1. If guest select to connect an existing account the system accepts the request and creates a new Travlendar+ account based on selected account and procedure ends.    2. If guest select to create a new account the system initializes the registration procedure. 2. The guest fills all the mandatory fields (first name, last name, email address and password), can insert information to manage a payment account and set the preferences (these two steps aren’t mandatory). 3. The guest clicks on button “Confirm”. 4. The system controls received data and saves it. 5. The guest must confirm registration through a link sent to email address previous inserted. |
| **Exit conditions** | * Guest has selected to connect an existing account to Travlendar+ and the system creates the account. * Guest after has confirmed the registration become new user.   From now he can log in the application ad start to use the system with all functionality. |
| **Exceptions** | * Connection with existing account is not possible. * The guest is already registered. * The information inserted inside mandatory fields are incorrect (ex. Email not valid, Password that not respect all requisites). * Username chosen by guest is already used. * Email inserted by guest is associated to another user.   For the first exception an error message shows to the user that is impossible to connect with the existing account and advices to create it without connecting on existing one.  When the other exceptions occur, an error message appears and displays the reason that caused the failure in the registration procedure. Then the procedure restart from the second steps. |

### 5.1.2. Log in

|  |  |
| --- | --- |
| **Name** | **Log in** |
| **Actors** | User |
| **Entry conditions** | The User is on the log in page of the application. |
| **Flow of events** | 1. The user inserts his credentials (username and password) inside appropriate fields. 2. The user clicks on “Log In” button. 3. The system receives the user’s request and allow him to view his calendar. |
| **Exit conditions** | The user is successfully redirected to view his calendar. |
| **Exceptions** | * The User inserts not valid Username. * The User inserts not valid Password.   All exceptions are handled with an error message that notify to the user the occurred issue and the procedure restarts to the first step. |

### 5.1.3. Manage preferences

|  |  |
| --- | --- |
| **Name** | **Manage preferences** |
| **Actors** | User, guest |
| **Entry conditions** | * The user, with a right slide on the screen, open the menu that contains a preferences button and clicks on it. * The guest, during registration procedure, clicks on preferences button. |
| **Flow of events** | 1. User/Guest actives the options that he need. 2. User/Guest clicks on back arrow and return to the previous screen (lateral menu for the user or registration form for the guest). 3. The system receives the user/guest choice and replaces/creates existing/new preferences. |
| **Exit conditions** | Preferences are set. |
| **Exceptions** | No exceptions provided. |

### 5.1.4. View calendar

|  |  |
| --- | --- |
| **Name** | **View calendar** |
| **Actors** | User |
| **Entry conditions** | The user must be logged in. |
| **Flow of events** | 1. The user on the homepage of the application can see the state of his calendar. |
| **Exit conditions** | The user controls all his appointments. |
| **Exceptions** | No exceptions provided. |

### 5.1.5. Change view calendar

|  |  |
| --- | --- |
| **Name** | **Change view calendar** |
| **Actors** | User |
| **Entry conditions** | The user is on the application homepage. |
| **Flow of events** | 1. The user, with a right slide on the screen, open the menu that contains all possible views. 2. User clicks on one of them. 3. The system receives the request to change the view and closes the menu. |
| **Exit conditions** | * User open the lateral menu but closes it through left slide without change the view. * System has changed the view and has closed the menu. |
| **Exceptions** | No exceptions provided. |

### 5.1.6. View daily schedule

|  |  |
| --- | --- |
| **Name** | **View daily schedule** |
| **Actors** | User |
| **Entry conditions** | The user selects daily view, and clicks on “view daily schedule” button. |
| **Flow of events** | 1. User visualizes in sequence all appointments scheduled for the selected day and the relative travel to reach them, with the estimated times. |
| **Exit conditions** | User clicks on back arrow and return to homepage (daily view). |
| **Exceptions** | * Appointments inserted are overlapped. * The travel to reach one appointment is too long to reach it without overlaps with another appointment.   When user views daily schedule, this is computed by exploiting weather condition or the live situation about the street. When the system run into two appointments that are overlapped (first exception) or travel too long (second exception) a message is displayed to the user and commit him to select which is/are the appointment/s to keep. |

### 5.1.7. View travel details

|  |  |
| --- | --- |
| **Name** | **View travel details** |
| **Actors** | User |
| **Entry conditions** | 1. The user is on daily schedule page and this is created without errors. 2. User clicks on one travel that are schedule for the day. |
| **Flow of events** | 1. User visualizes all information about the selected travel, such as the vehicle to reach the appointments, the trace route and the travel estimated time. |
| **Exit conditions** | User clicks on back arrow and return to daily schedule. |
| **Exceptions** | No exceptions expected. |

### 5.1.8. View travel alternatives

|  |  |
| --- | --- |
| **Name** | **View travel alternatives** |
| **Actors** | User |
| **Entry conditions** | 1. The user is on daily schedule page and this is created without errors. 2. User clicks on one travel that are schedule for the day. |
| **Flow of events** | 1. The user visualizes little bar that shows different transports. 2. User, clicking on one of these transports icon, changes the travel way. 3. The system receives the user request and re-route the travel on the user choice. 4. The system replaces the old travel screen with new compute travel and shows to the user all details regarding it. |
| **Exit conditions** | The system shows the new travel way on the user choice. |
| **Exceptions** | Only exception provided is the absence of travel alternatives. This is handled with a message that report to the user the lack of alternatives. |

### 5.1.9. Choose travel alternative

|  |  |
| --- | --- |
| **Name** | **Choose travel alternative** |
| **Actors** | User |
| **Entry conditions** | User have been selected one among the travel alternative and it exists. |
| **Flow of events** | 1. The system shows travel details to the user. 2. User, clicking on button which have “floppy disk” shape, overwrite the old travel with the new. 3. Application stays on the same page, thus to allow to the user to visualize travel details and wait a user action. |
| **Exit conditions** | User clicks on back arrow and return to daily schedule. Now the old travel has been overwritten with the new choice. |
| **Exceptions** | The only exception provided is expected during the procedure to saving data into the memory. This error can be caused by internet connection absence and it involve that is impossible to update the changes on the account.  The system shows message error and ask to the user to retry to save the changes. |

### 5.1.10. View movement details

|  |  |
| --- | --- |
| **Name** | **View movement details** |
| **Actors** | User |
| **Entry conditions** | 1. User is on the page that shows travel details. 2. User clicks on the icon that figure the vehicle related to the movement. |
| **Flow of events** | 1. User visualizes all information about the selected movements, such as the vehicle to reach the appointments, the trace route and the travel estimated time. |
| **Exit conditions** | User clicks on back arrow and return to travel details. |
| **Exceptions** | No exceptions expected. |

### 5.1.11. View movement alternatives

|  |  |
| --- | --- |
| **Name** | **View movement alternatives** |
| **Actors** | User |
| **Entry conditions** | 1. User is on the page that shows travel details. 2. User clicks on the icon that figure the vehicle related to the movement. |
| **Flow of events** | 1. The user visualizes little bar that shows different transports. 2. User, clicking on one of these transports icon, changes the movement way. 3. The system receives the user request and re-route the movement on the user choice. 4. The system replaces the old movement screen with new movement and shows to the user all details regarding it. |
| **Exit conditions** | The system shows the new movement way on the user choice. |
| **Exceptions** | Only exception provided is the absence of movement alternatives. This is handled with a message that report to the user the lack of alternatives. |

### 5.1.12. Choose movement alternative

|  |  |
| --- | --- |
| **Name** | **Choose movement alternative** |
| **Actors** | User |
| **Entry conditions** | User have been selected one among the movement alternative and it exists. |
| **Flow of events** | 1. The system shows movement details to the user. 2. User, clicking on button which have “floppy disk” shape, overwrite the old travel with the new. 3. Application stays on the same page, thus to allow to the user to visualize movement details and wait a user action. |
| **Exit conditions** | User clicks on back arrow and return to daily schedule. Now the old movement has been overwritten with the new choice. |
| **Exceptions** | The only exception provided is expected during the procedure to saving data into the memory. This error can be caused by internet connection absence and it involve that is impossible to update the changes on the account.  The system shows message error and ask to the user to retry to save the changes. |

### 5.1.13. Buy travel ticket

|  |  |
| --- | --- |
| **Name** | **Buy travel ticket** |
| **Actors** | User, Ticket manager |
| **Entry conditions** | User have selected one movement to see it details and clicks on “buy ticket” button. |
| **Flow of events** | 1. The system receives the user request and takes the payment information which this it can complete the transition. 2. The system sends the payment information to the ticket manager that complete the operation “buy ticket”. 3. The system shows, on the same page where are contained the movement details, the ticket bought by user. |
| **Exit conditions** | The system shows bought ticket to the user. |
| **Exceptions** | * Not sufficient credit to buy the ticket. * Payment account not valid. * Internet connection error.   All these exceptions are handled with a message error that advise the user to repeat the operation. |

### 5.1.14. Delete appointment

|  |  |
| --- | --- |
| **Name** | **Delete appointment** |
| **Actors** | User |
| **Entry conditions** | 1. User selects a specific appointment in his calendar (through daily or weekly view). 2. User selects “Delete” command present in an option panel that contains all possible commands. |
| **Flow of events** | 1. The user confirms the deletion. 2. The system receives the request and remove the appointment from the calendar. 3. The system removes also the travel that are associated to deleted appointment. 4. The system removes all alerts that are associated to deleted appointment. |
| **Exit conditions** | The appointment selected by user has been cancelled and the system redirects the user to the previous screen displayed by him before delete the appointment. |
| **Exceptions** | No exceptions provided. |

### 5.1.15. Create appointment

|  |  |
| --- | --- |
| **Name** | **Create appointment** |
| **Actors** | User |
| **Entry conditions** | The user clicks on “Plus” button laid on homepage. |
| **Flow of events** | 1. The system loads a new page that contains all fields required to create new event. 2. The user fills all mandatory fields (name, date, time and location).   The following steps aren’t mandatory:   * 1. User can choose among the available icons.   2. User can add alert related to the new event.   3. User can create a more complex event with “more option”.  1. User clicks on “Save” button. 2. The system controls if the received data are correct. 3. System collects all information inserted by the user and creates a new event, saves it and updates the calendar. |
| **Exit conditions** | * User clicks on “Confirm” button and the event created is added to the calendar and application return to calendar view. * User clicks on back arrow and the procedure to create a new event is interrupted without save. |
| **Exceptions** | The only exception is occurred when user inserts wrong information inside the mandatory fields. It is handled notifying the error to the user and restart the execution from second steps. |

### 5.1.16. Edit appointment

|  |  |
| --- | --- |
| **Name** | **Edit appointment** |
| **Actors** | User |
| **Entry conditions** | 1. User selects a specific appointment in his calendar (through daily or weekly view). 2. User selects “Edit” command present in an option panel that contains all possible commands. |
| **Flow of events** | 1. System receives the request and loads all information related to the selected appointment and these are presented to the user. 2. User edit the information needed. 3. User clicks on “Confirm” button. 4. System check if the information inserted by the user violate any constraints. 5. The system saves the changes and update the calendar. |
| **Exit conditions** | he appointment selected by user has been updated and the system redirects the user to the previous screen displayed by him before edit the appointment. |
| **Exceptions** | The only exception is occurred when user inserts wrong information inside the mandatory fields. It is handled notifying the error to the user and restart the execution from second steps. |

### 5.1.17. Create flexible appointment

|  |  |
| --- | --- |
| **Name** | **Create flexible appointment** |
| **Actors** | User |
| **Entry conditions** | 1. The user initializes the procedure of event creation. 2. User clicks on “more options” button and a panel that contains all the advanced options is opened. |
| **Flow of events** | 1. User clicks on dropdown menu related to flexible setting. 2. User select the preferred option. 3. User clicks on “Save” button. 4. The system receives the request and saves the changes. |
| **Exit conditions** | * User clicks on back arrow before to save the changes and the system without considering the user changes return to event creation page. * User clicking on “save” button, actives the saving procedure. After this step the system return to event creation page. |
| **Exceptions** | No exception provided. |

### 5.1.18. Create repeatable appointment

|  |  |
| --- | --- |
| **Name** | **Create repeatable appointment** |
| **Actors** | User |
| **Entry conditions** | 1. The user initializes the procedure of event creation. 2. User clicks on “more options” button and a panel that contains all the advanced options is opened. |
| **Flow of events** | 1. User clicks on dropdown menu related to repeatable setting. 2. User select the preferred option. 3. User clicks on “Save” button. 4. The system receives the request and save the changes into the memory. |
| **Exit conditions** | * User clicks on back arrow before to save the changes and the system without considering the user changes return to event creation page. * User clicking on “Save” button, actives the saving procedure. After this step the system return to event creation page. |
| **Exceptions** | No exception provided. |

### 5.1.19. Create alert

|  |  |
| --- | --- |
| **Name** | **Create alert** |
| **Actors** | User |
| **Entry conditions** | * User creates a new appointment. * User edit an existing appointment. |
| **Flow of events** | 1. User clicks on “add alert” button. 2. System provide to open new page that contains all fields to create a new alarm. 3. User fills all fields and clicks “Confirm” button. 4. System saves the new alarm. |
| **Exit conditions** | The alarm has been created and the application return to the event creation/modification page, and it is possible to see the created alert. |
| **Exceptions** | No exceptions provided. |

### 5.1.20. Delete alert

|  |  |
| --- | --- |
| **Name** | **Delete alert** |
| **Actors** | User |
| **Entry conditions** | 1. The user selects a specific alarm related to a specific appointment previous selected. 2. User clicks on “Delete” command. |
| **Flow of events** | 1. The user confirms the deletion. 2. System delete the alarm from the memory. |
| **Exit conditions** | The selected alarm has been deleted and the application returns to appointment edit page. |
| **Exceptions** | No exceptions provided. |

### 5.1.21. Edit alert

|  |  |
| --- | --- |
| **Name** | **Edit alert** |
| **Actors** | User |
| **Entry conditions** | 1. The user selects a specific alarm related to a specific appointment previous selected. 2. User clicks on “Edit” command. |
| **Flow of events** | 1. The user edits the alarm settings needed. 2. The user clicks on “Confirm” button. 3. The system receives the request and update the alarm and save all changes. |
| **Exit conditions** | The selected alarm has been updated and the application returns to appointment edit page. |
| **Exceptions** | No exceptions provided. |

## C:\Users\andre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ClassDiagram.png5.2. Class diagram

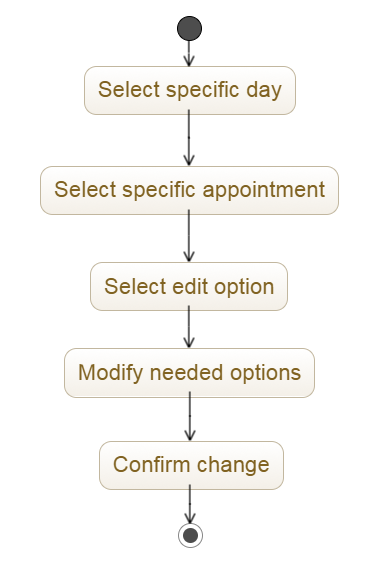
## 5.3. Sequence diagrams

## 5.4. Activity diagram

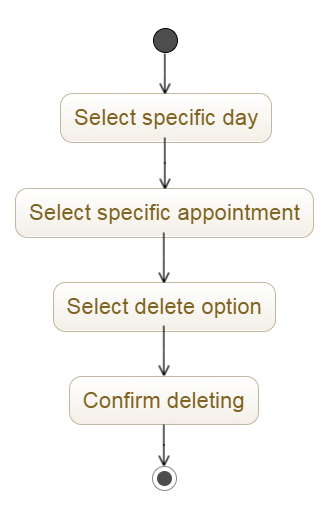
### 5.4.1. Create new appointment

### C:\Users\andre\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CreateEvent.png

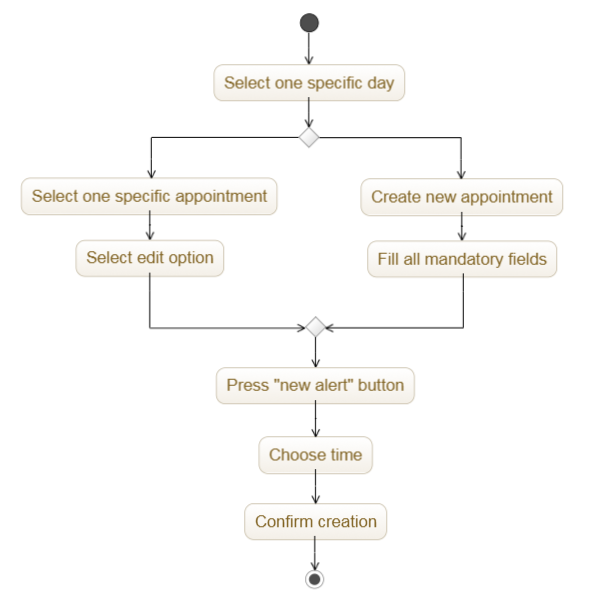
### 5.4.2. Edit existing appointment



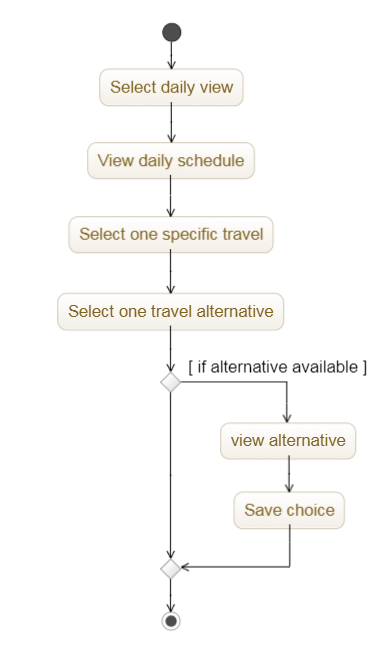
### 5.4.3. Delete existing appointment



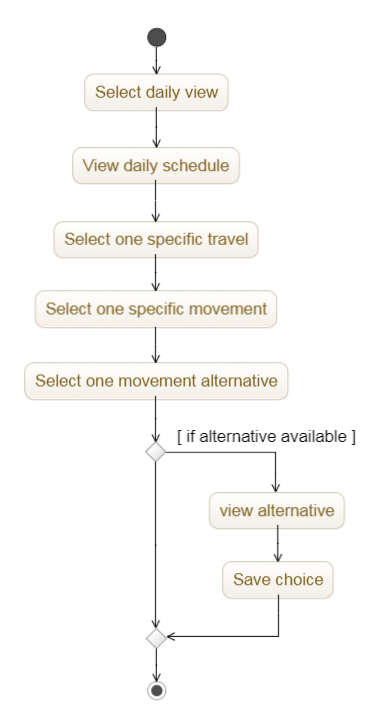
### 5.4.4. Create new alert



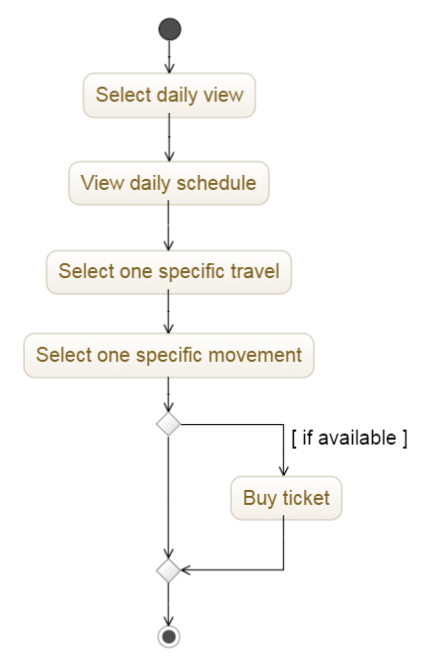
### 5.4.5. Choose travel alternative



### 5.4.6. Choose movement alternative



### 5.4.7. Buy ticket



# 

# 6. Formal analysis using alloy

# 7. Effort spent

# 8. References