

# Travlendar+

## Requirement Analysis Specification Document

Calzavara Filippo, Filaferro Giovanni, Benedetto Maria Nespoli

Version 1.0.0



# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.2.1	Goals . . . . .	4
1.3	Definitions, Acronyms, Abbreviations . . . . .	6
1.3.1	Definitions . . . . .	6
1.3.2	Acronyms . . . . .	6
1.3.3	Abbreviations . . . . .	6
1.4	Revision history . . . . .	6
1.5	Reference Documents . . . . .	6
1.6	Document Structure . . . . .	6
<b>2</b>	<b>Overall Description</b>	<b>7</b>
2.1	Product perspective . . . . .	7
2.1.1	Class Diagram . . . . .	8
2.1.2	Statechart . . . . .	8
2.2	Product functions . . . . .	10
2.2.1	Allow people to purchase and view the transit ticket . . .	10
2.3	User characteristics . . . . .	10
2.4	Assumptions, dependencies and constraints . . . . .	10
2.4.1	Interfaces to other applications . . . . .	11
2.4.2	Public transit pass . . . . .	11
2.4.3	Domain assumption . . . . .	11
<b>3</b>	<b>Specific Requirements</b>	<b>12</b>
3.1	External Interface Requirements . . . . .	12
3.1.1	User Interfaces . . . . .	12
3.1.2	Hardware Interfaces . . . . .	29
3.1.3	Software Interfaces . . . . .	29
3.1.4	Communication Interfaces . . . . .	29
3.2	Functional Requirements . . . . .	30
3.2.1	[G0] Allow people to use the app without a login function	30
3.2.2	[G1] Allow people to view the daily schedule with coming up events at the top . . . . .	30
3.2.3	[G2] Allow people to view previous events . . . . .	30
3.2.4	[G3] Allow people to view the detail of each event . . . .	30
3.2.5	[G4] Allow the users to create an event . . . . .	31
3.2.6	[G5] Allow people to see daily events on a map . . . . .	31
3.2.7	[G6] Allow the user to set preferences . . . . .	32
3.2.8	[G7] Allow the users to create calendars . . . . .	32
3.2.9	[G8] Notify the user that it's time to leave for the appoint- ment . . . . .	32
3.3	Definition of use cases . . . . .	33
3.3.1	CreateCalendar . . . . .	33

3.3.2	AddStandardEvent . . . . .	34
3.3.3	AddFlexibleEvent . . . . .	35
3.3.4	EditPreferences . . . . .	35
3.3.5	BuyTickets . . . . .	37
3.3.6	NotifyUser . . . . .	38
3.4	Use Case Diagram . . . . .	39
3.5	Performance Requirements . . . . .	39
3.6	Design Constraints . . . . .	40
3.6.1	Standards compliance . . . . .	40
3.6.2	Hardware limitations . . . . .	40
3.6.3	Any other constraint . . . . .	40
3.7	Software System Attributes . . . . .	40
3.7.1	Reliability . . . . .	40
3.7.2	Availability . . . . .	40
3.7.3	Security . . . . .	40
3.7.4	Maintainability . . . . .	41
3.7.5	Portability . . . . .	41
<b>4</b>	<b>Formal analysis using Alloy</b>	<b>41</b>
<b>5</b>	<b>Effort spent</b>	<b>41</b>
<b>6</b>	<b>References</b>	<b>41</b>

# 1 Introduction

## 1.1 Purpose

This document is the Requirements Analysis Specification Document (RASD) for Travlendar+, the system assigned to be developed for the mandatory project. The document describes the characteristics of the system, both functional and non-functional requirements for the software, design constraints and system interfaces. It is addressed to the developers that will implement the final system.

## 1.2 Scope

The aim of this project is to create a smart calendar-based application that people can use to reduce delays at the appointments and supporting the the users by identifying the best mobility option. Moreover, peoples need to be able to create meetings that notify the user in case he/she will not be on time for it. The system needs to take into account weather forecast as well as strike days. Furthermore, the users should be able to adapt settings based on their preferences, such as walking-time constraints or a more eco-friendly way of transportation, and the possibility to define flexible appointments that given a window of time and a duration an arrangement of the schedule is proposed to the user. Finally, the system should provide the user a ticket or locate the closest car/bike sharing vehicle if it is the mean of transportation chosen.

### 1.2.1 Goals

- [G0] Allow people to use the app without a login function
- [G1] Allow people to view the daily schedule with coming up events at the top
- [G2] Allow people to view previous events
- [G3] Allow people to view the details of each event
  - [G3.1] Allow people to get travel information for a particular event
    - \* [G3.1.1] Allow people to get the ETA to the appointment
  - [G3.2] Allow people to explore a specific route
    - \* [G3.2.1] Allow people to purchase the transit ticket
    - \* [G3.2.2] Allow people to open the sharing services/ride provider's app and call the vehicle
  - [G3.3] Allow people to eventually view the purchased transit ticket
- [G4] Allow the users to create an event
  - [G4.1] Allow the user to define a name/description of the appointment
  - [G4.2] Allow the user to define the location

- [G4.3] Allow users to set a flexible timing
  - \* [G4.3.1] Allow the user to select a time window in which the event will occur
  - \* [G4.3.2] Allow the user to select how long the event will last
- [G4.4] Allow the user to set a starting time and ending time on non-flexible events
- [G4.5] Allow the user to repeat the event regularly throughout the week
- [G4.6] Allow the user to assign the event to a specific calendar
- [G4.7] Allow the user to define the mean of transportations that he/she wants to use for the event
- [G5] Allow people to see daily events on a map
  - [G5.1] Allow the user to have information about the status of the journey to make in order to reach the appointment's location
  - [G5.2] Allow the user to see previous daily events on a map
- [G6] Allow the user to set preferences
  - [G6.1] Allow the user to set the eco-friendly mode
  - [G6.2] Allow the user to set the walkable distance limit
  - [G6.3] Allow the user to set the maximum biking range
  - [G6.4] Allow the users to set the owned bike and/or bike sharing providers
  - [G6.5] Allow the users to set a time range in which they can take public transits
  - [G6.6] Allow the users to set if they own a car and/or their favorite car sharing providers
  - [G6.7] Allow the users to set the preferred ride sharing service
- [G7] Allow the users to create calendars
  - [G7.1] Allow the users to color calendar
  - [G7.2] Allow the users to change calendar's color and name
- [G8] Notify the user that it's time to leave for the appointment

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

Definition	Explanation
Appointment	A period of time in which something take place at a certain time
Travlendar+	The name of the platform to develop
Event	A synonym for appointment
System	A synonym of Travlendar+
User	A potential utilizer of this project
Ride	A service performed by a ride-sharing company and Cabs

### 1.3.2 Acronyms

Acronym	Explanation
GUI	Graphic User Interface
ETA	Estimated time of arrival
API	Application programming interface
RASD	Requirement Analysis and Specification Document
PNR	Passenger name record
QR	QR Code

### 1.3.3 Abbreviations

Abbreviation	Explanation
App	A synonym of Travlendar+
[Gn]	N-goal
[Dn]	N-domain assumption
[Rn]	N-functional requirement

## 1.4 Revision history

- 1.0.0 - Initial Version (16/10/2017)

## 1.5 Reference Documents

- Specifications document “Mandatory Project Assignments.pdf”
- ISO/IEC/IEEE 29148 - IEEE Standard on Requirements Engineering
- Software Abstractions, MIT Press

## 1.6 Document Structure

The paper includes six areas. The first one, is composed by the introductory information provided in order to give an orientative view of what this document is about. The second one, provides an overall description of the project including

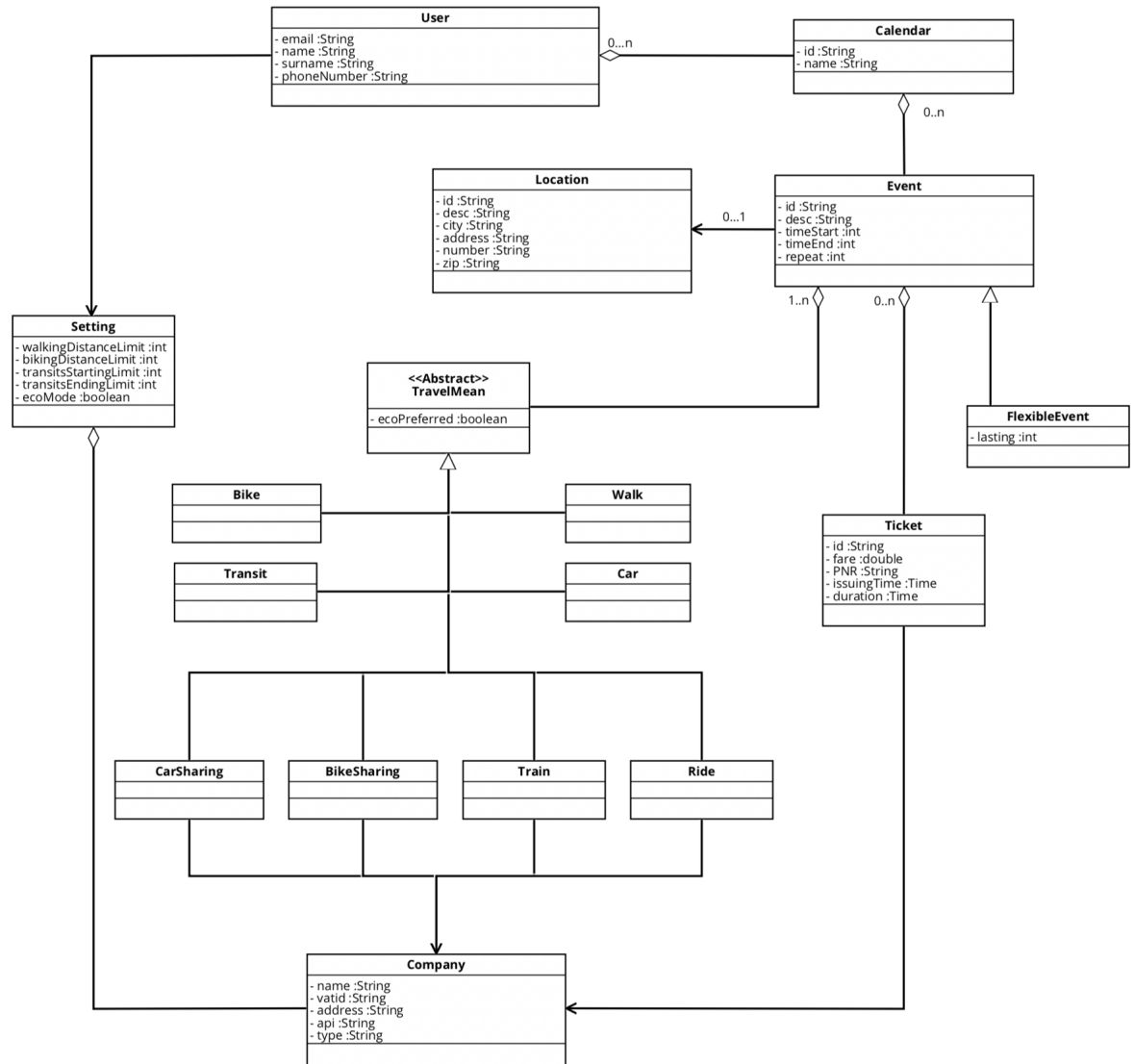
the features, the characteristics and the constraints that its development implies. The main body of this paper is contained in the third section, which assert the requirements and the design constraints. The paper is equipped with a fourth section that contains a formal analysis using Alloy. Finally, a fifth and a sixth section allows the reader to learn the effort spent on this project by each team member and the references made in the whole paper.

## **2 Overall Description**

### **2.1 Product perspective**

Travlendar+ is a standalone system that provides the functionality described in the Product Function Section 2.2. It would implement external services in order to offer weather forecast, maps services, sharing systems information, transits and strike information.

### 2.1.1 Class Diagram

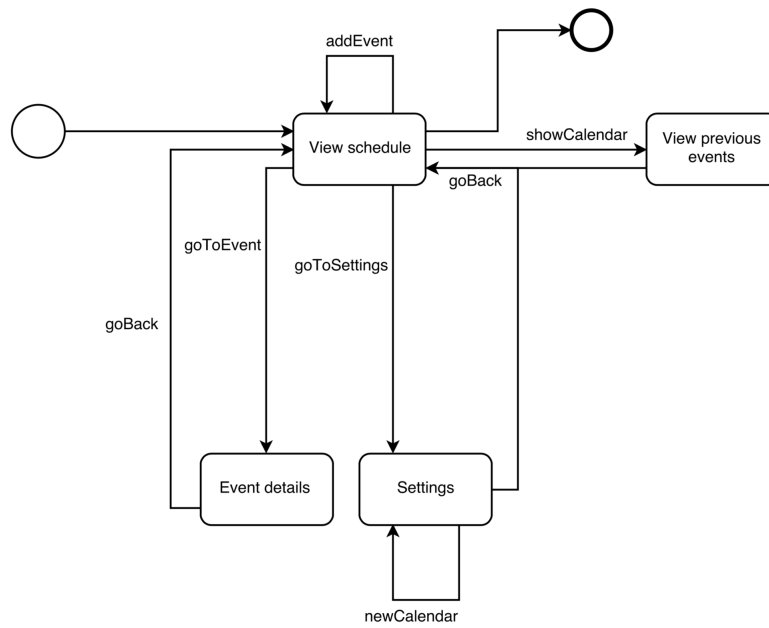


### 2.1.2 Statechart

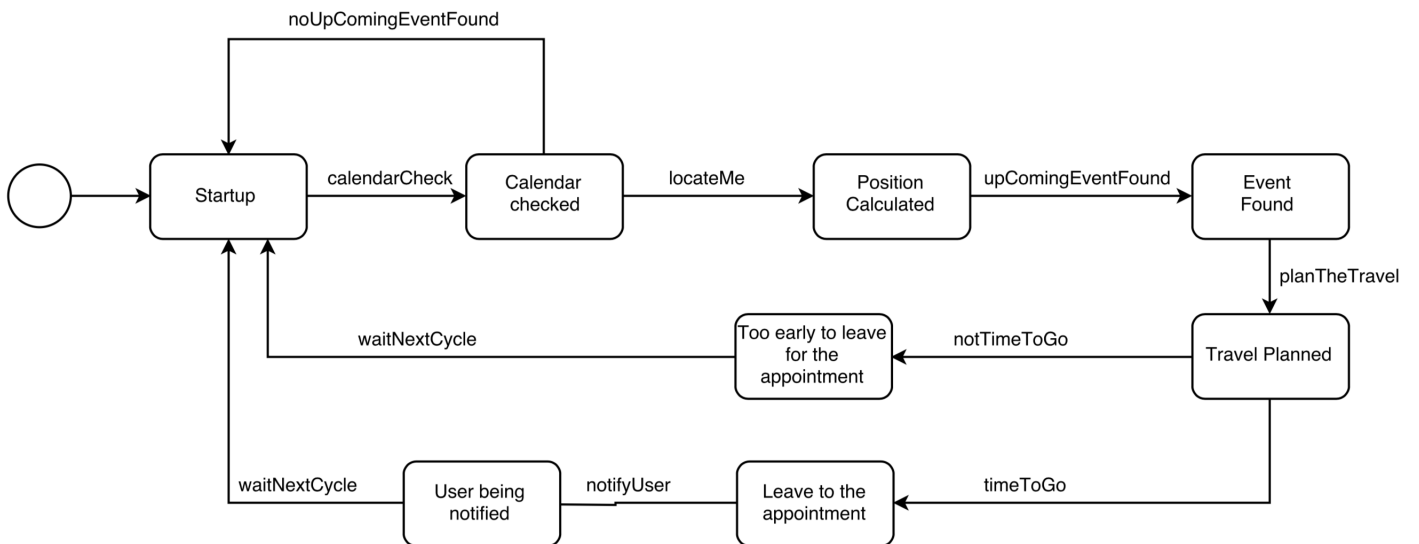
The system is based on two different modes: The active function and the passive function. The first is in place when the person is actively using the application and its functionality. The latter is a background service that runs in order to keep track of the position and other information and it attract user's attention when a notification is risen.



### Active mode:



### Passive mode:



## **2.2 Product functions**

Each goal described in Section 1.2.1 is function that will be implemented in the system. We report below the functionalities that require a further explanation and the boundaries between the system to be developed and third party system.

### **2.2.1 Allow people to purchase and view the transit ticket**

The ticket buying system will use regular text messages to send a purchasing request at the local transit company which reply to our app with the PNR code of the ticket. This choice of designing was made to ensure compatibility among every transit provider and to allow users to pay it by charging the cost of the ticket on their cell phone credit/bill, letting the carrier provider handling the payment.

When the purchasing is completed, a QR code representing the ticket's PNR will be displayed on the user's screen.

## **2.3 User characteristics**

This application is designed to be used by any person that would like to have a more organized schedule

## **2.4 Assumptions, dependencies and constraints**

We assume that:

1. The client is using the app through a new generation smartphone
2. The mobile device has a GPS system
3. The mobile device has a regular 3G internet connection available
4. The software is designed to be utilized in Italy, precisely in the region of Lombardy
5. Public transits companies provide a sms-based purchasing system
6. The user has already logged in to his/her iCloud account (which is compulsory on all iOS and macOS devices) so that we can identify him/her by using the unique iCloud Keychain identifier
7. Vehicle-sharing/ride-sharing companies' will provide APIs to locate the nearest vehicle available
8. Weather forecast are provided from an external service through API
9. While calculating the shortest path, follows a time-based criteria instead of a cost-based one

#### **2.4.1 Interfaces to other applications**

At the first stages of the project, Travlendar+ will not release API interfaces to the public

#### **2.4.2 Public transit pass**

The system will not take into consideration whether the person owns a transit's pass, however it always allows to buy a transit ticket via text-message if asked by the user

#### **2.4.3 Domain assumption**

- [D1] App data are stored remotely and retrieved by multiple devices
- [D2] Up coming events' section needs to contain appointments planned to happen 2h before
- [D3] Transportation information are not shown for the past events
- [D4] Indications on the ETA, calculated mean of transportation and time-to-leave are calculated on the current approximative position
- [D5] User's position is retrieved using GPS system
- [D6] Position is re-calculated every 2 hours, or when a significant change in GPS position is discovered
- [D7] The map can be browsed by the user
- [D8] The localization service is available and functionally working
- [D9] The map service is reliable
- [D10] The button to choose the calendar is, by default, the last calendar chosen, or the first calendar if the user has never added an event
- [D11] While creating an event, the system must take into account whether the eco mode has been turned on
- [D12] The map can be browsed by the user
- [D13] Every preference agrees with the actual record saved in the database
- [D14] Allowed services have been preinstalled
- [D15] The user has always at least one calendar
-

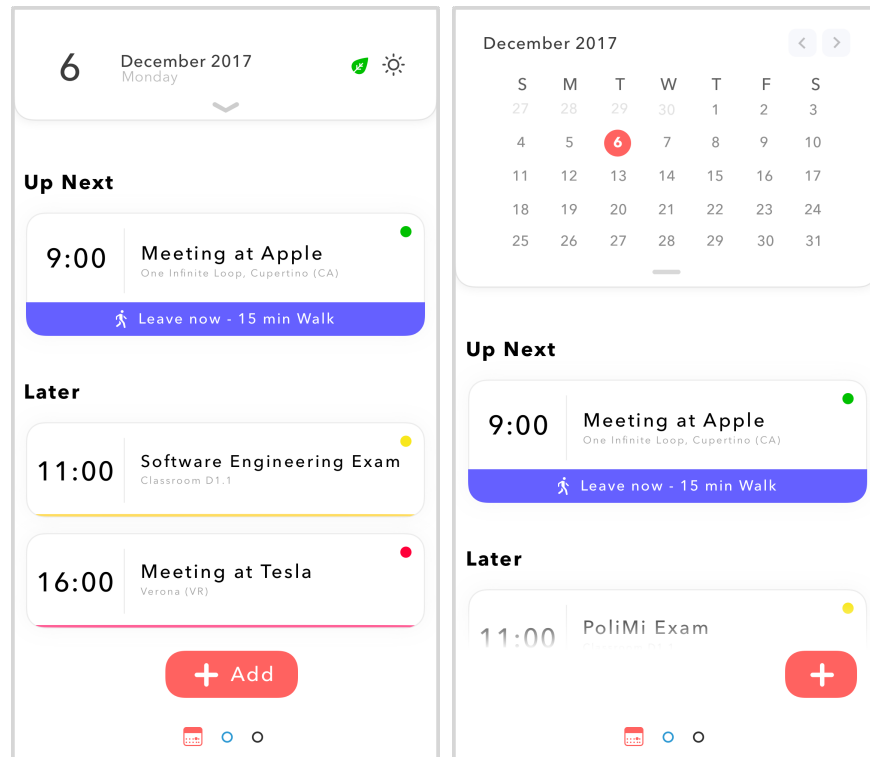
## **3 Specific Requirements**

### **3.1 External Interface Requirements**

#### **3.1.1 User Interfaces**

Continues on the next page.

## Home Screen



That's the main page that will appear once opened the app. People can now see directly their upcoming events of the day and just by pulling down the top view (where 16 December 2017 is written) they can access the calendar view (Right Picture) and simply by tapping on a particular day the UI re-adapts to that day and shows its regarding events.

In this page people can have a look if they chose to enable ECO-Friendly mode (Green Leaf on top) and the current weather.

Regarding Events:

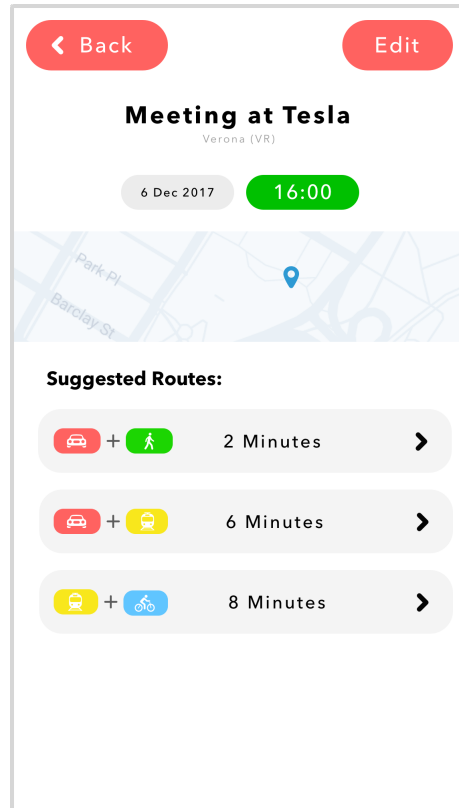
The first event of the day shows the suggested route informations and the ETA.

There is also a little dot at the upper right side of the cell in which people can see if they'll arrive early, late or on time at the event:

- Green dot means on time / early.
- Yellow dot means that the user will be late if he/she doesn't hurry up.
- Red dot means he'll be late or that there is a problem in that route.

In each cell there is a small bar on the bottom which color depends on the calendar color where the event is saved to.

## Event Detail

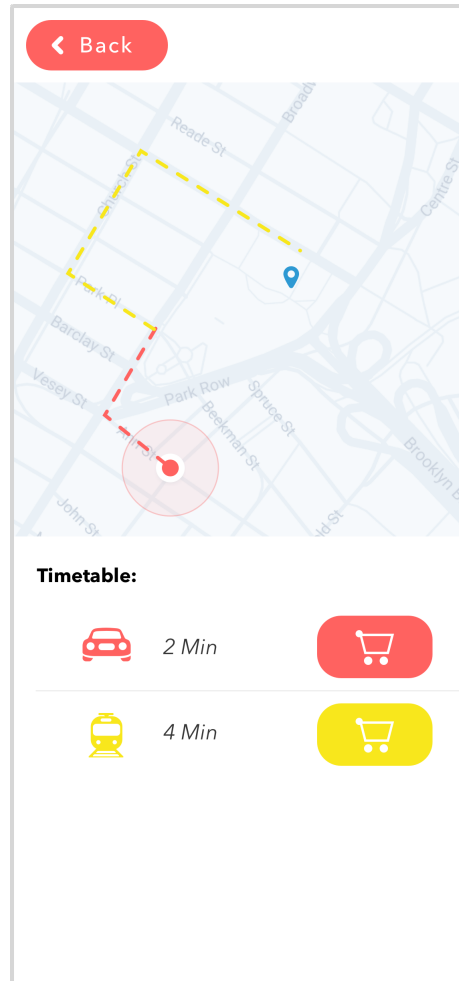


This is the place where, once tapped on an event in the Home Screen, people can have a fast look at its more interesting details:

- Basic details of the event such as day, time, position and name.
- There will be a map view where the location of the event will be shown.
- Suggested routes (compound ones too) to take based on the user preferences

From this view the user is allowed also to edit the event by tapping the “Edit” button.

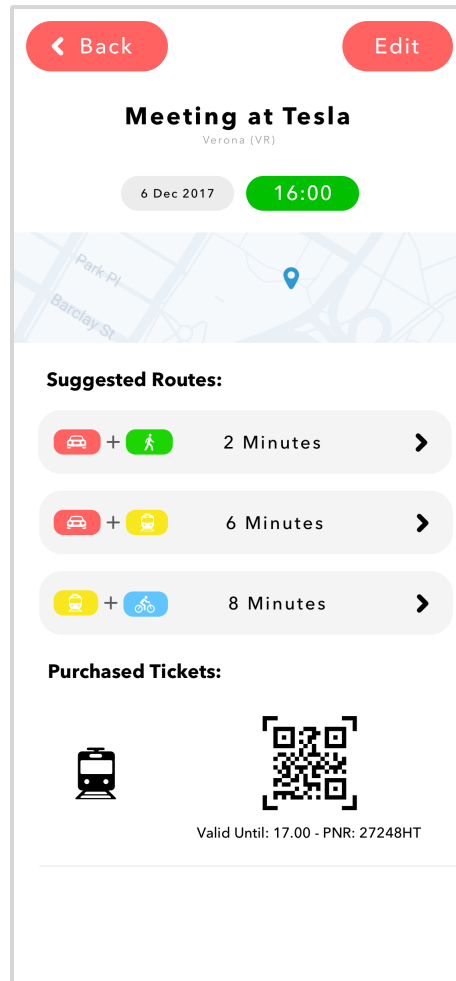
## Event Detail / Route Detail



Just by tapping to one suggested route from “Event Detail” view, the app will present to the user this view in which he / she are allowed to have a look at the routes to reach the event.

In particular all the parts composing the route are shown on map and just by tapping the button on the right of each cell the app will prompt the user to buy a ticket.

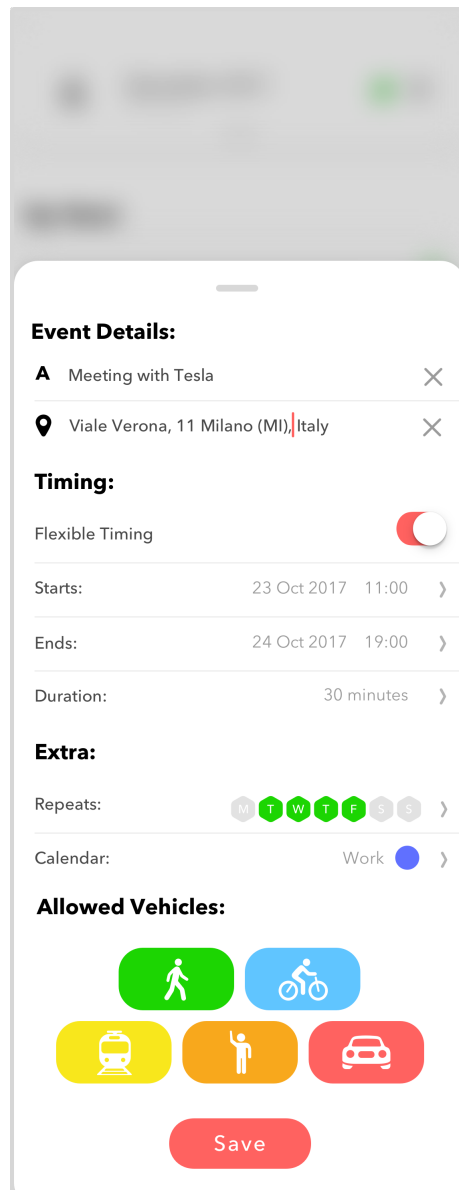
## Event Detail / Ticket Bought



If the user has bought a ticket it will appear then in this view which is the main “Event Detail” view so that he can just show the ticket or have a fast look to the QR code.



## Add Event



A mobile app form for adding an event. The form is white with rounded corners and a shadow, set against a blurred background. It contains several sections: 'Event Details' with fields for title and location; 'Timing' with a toggle for 'Flexible Timing' and date/time pickers for start, end, and duration; 'Extra' with a 'Repeats' section showing a weekly calendar and a 'Calendar' section with a 'Work' toggle; and 'Allowed Vehicles' with five color-coded buttons for different transport modes. A red 'Save' button is at the bottom.

**Event Details:**

**A** Meeting with Tesla ✕

**📍** Viale Verona, 11 Milano (MI), Italy ✕

**Timing:**

Flexible Timing ☒

Starts: 23 Oct 2017 11:00 ›

Ends: 24 Oct 2017 19:00 ›

Duration: 30 minutes ›

**Extra:**

Repeats: M T W T F S S ›

Calendar: Work ● ›

**Allowed Vehicles:**

🚶 🚲

🚆 🚶 🚗

Save

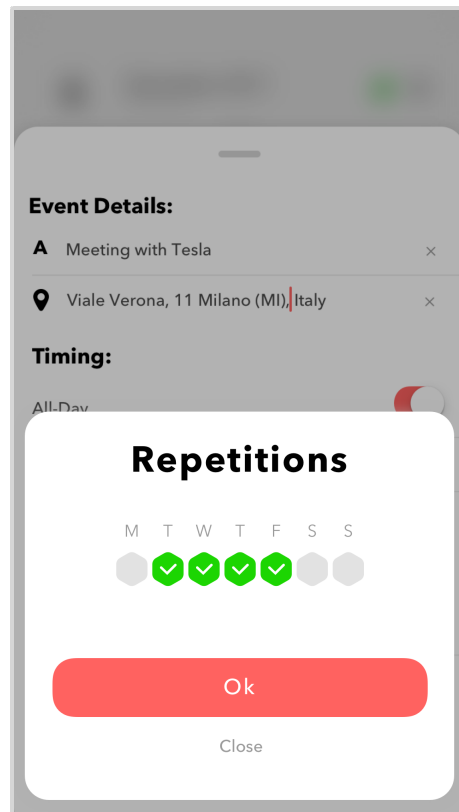
Now let's go back to the main view.

Once the user taps the add button this is the view that the app will present to him.

In here he can set:

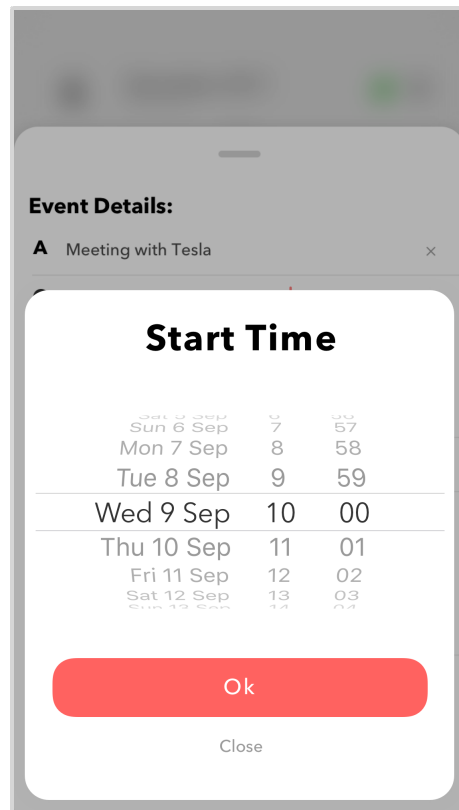
- Name of the event.
- Location of the event.
- Start time and end time.
- If he / she toggles the “Flexible Timing” switch a new cell called “Duration” will appear so that he can just say that “From start time to end time this event will last for Duration minutes”. This is useful if the user will set breaks during the day.
- An Extra section where the user can set repeating days and choose a calendar where to save the event.
- There is also an additional section where the user can choose the vehicles and travel means so that the app can compute the best route based on them.
- Tapping an save will store the event locally and synchronize it on the server.

## Add Event / Repetitions



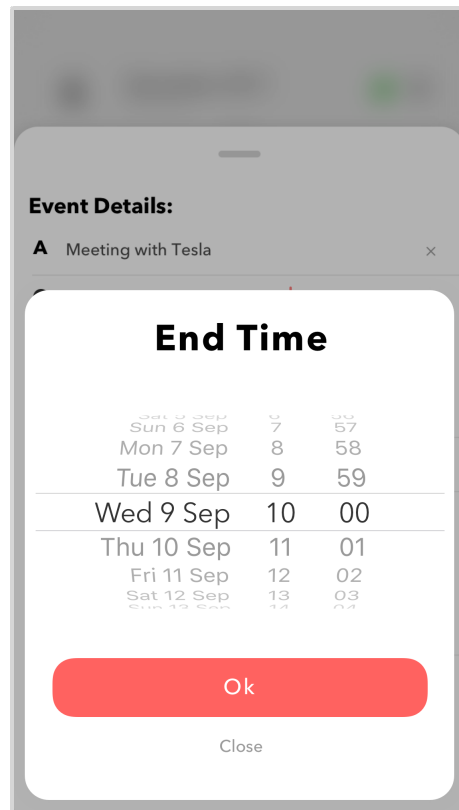
This is the pop-up that will appear once the user taps on the “Repeats” cell from the “Add Event” screen.

## Add Event / Start Date Time



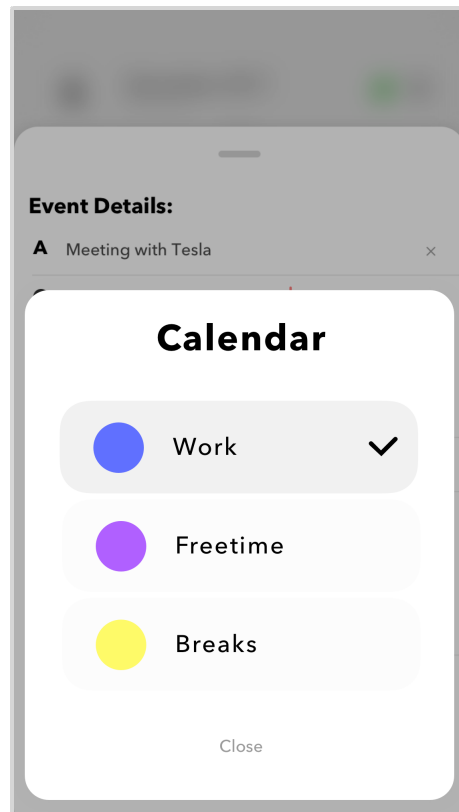
This is the pop-up that will appear once the user taps on the “Starts” cell from the “Add Event” screen.

## Add Event / End Date Time



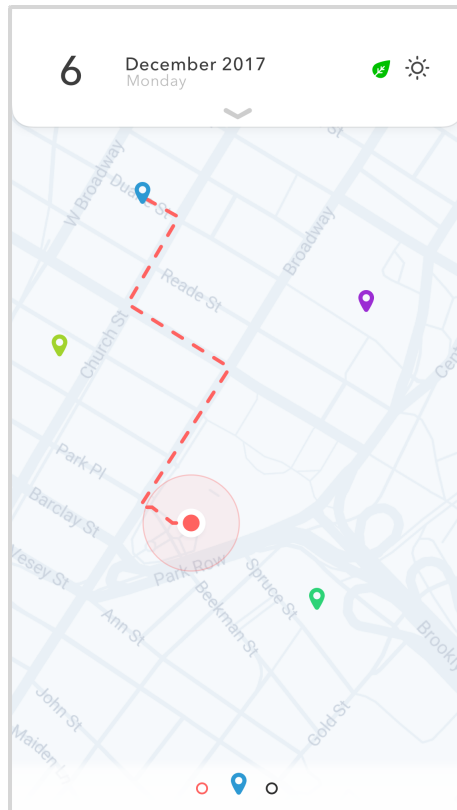
This is the pop-up that will appear once the user taps on the “Ends” cell from the “Add Event” screen.

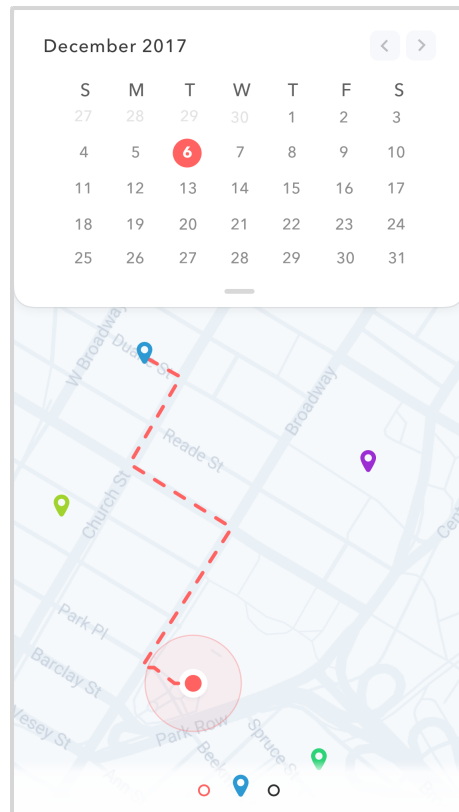
## Add Event / Calendar Pick



This is the pop-up that will appear once the user taps on the “Calendar” cell from the “Add Event” screen.

## Map





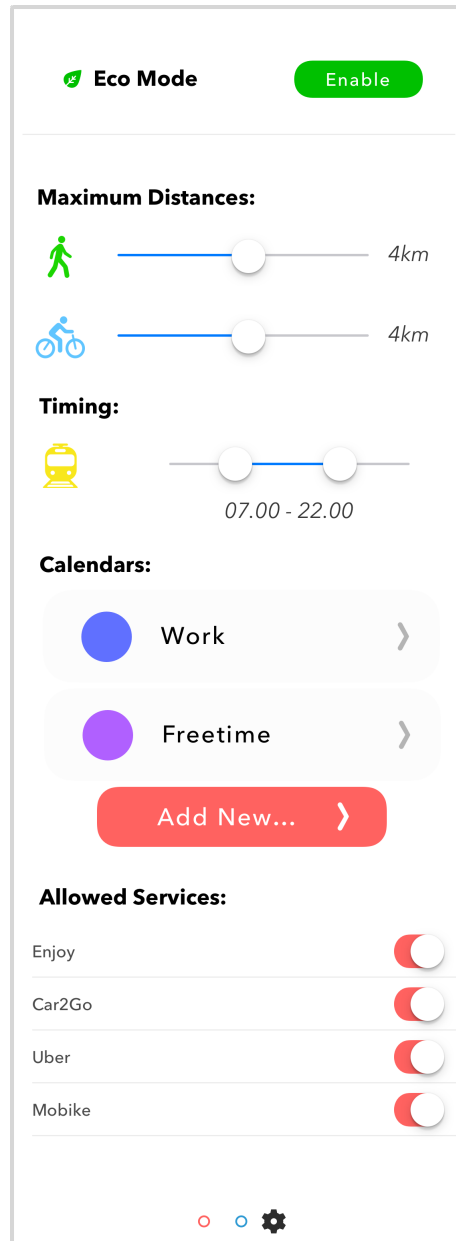
Swiping right from the main view or by tapping on the bottom icon the app will present this view.

In here the user is allowed to have a look at his / her daily events on a map and it shows also the suggested route from her / his position to the first event location.

Pulling down the calendar the user can change day and have a look at that specific events.



## Settings



Swiping right from the map view or by tapping on the bottom icon the app will present this view.

In this view we can find 5 sections:

**Eco Mode** By enabling it the app will minimize routes powered by cars or vehicles that produce carbon.

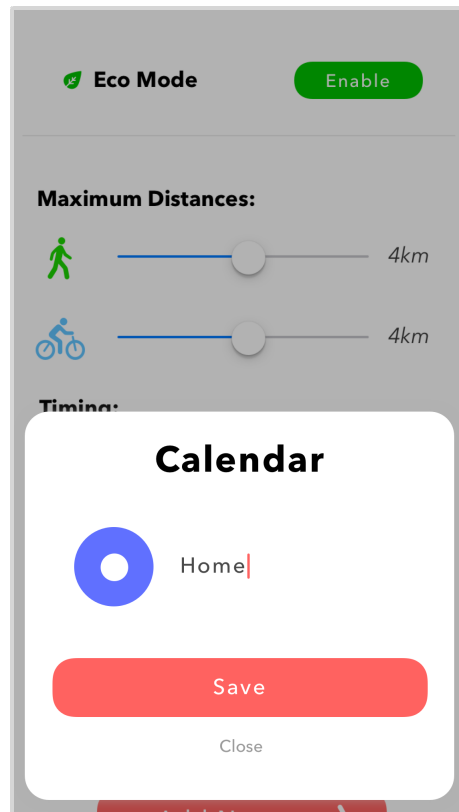
**Maximum Distances** Here the user is allowed to set a maximum amount of distance walking or biking so that the app will re arrange its calculations based on this new values.

**Timing** Here the user can just set the protected times where he / she wants the app to propose him transit vehicles.

**Calendars** In this section the user can edit his existing calendars or create a new one by tapping “Add New...” button.

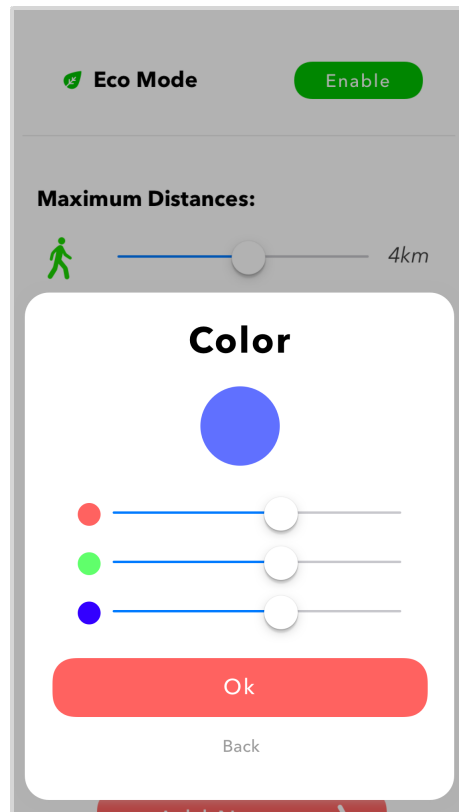
**Allowed Services** Here the user can just activate or de-activate services (“Enjoy”, “Mobike”, ...) so that he can just disable them and the app won’t consider them.

## Settings / Add Calendar



This is the view that will be presented when the user wants to add a new calendar.

Settings / Add Calendar / Color



This is the view where the user is allowed to edit the color of an existing or new calendar just by adjusting the RGB Sliders.

### **3.1.2 Hardware Interfaces**

The system will require, in order to work properly, one or more dedicated servers in which the server management software (API) will run.

It also requires the user to have a smartphone in order to download and use the app.

### **3.1.3 Software Interfaces**

The system will need:

- A server database where it will store user data like calendars, events and preferences.
- A client database where it will store user data on device so that the app will work offline too.
- The App that will be installed on the users's smartphones.

### **3.1.4 Communication Interfaces**

In order to establish a communication between the app and the server, there will be needed to build an API that lets the app to store and / or retrieve data from the database.

## **3.2 Functional Requirements**

### **3.2.1 [G0] Allow people to use the app without a login function**

- When the app is launched for the first time, the system recognizes the new users and create a record of their data to be locally updated on the phone
- [D1] App data are stored remotely and retrieved by multiple devices

### **3.2.2 [G1] Allow people to view the daily schedule with coming up events at the top**

- The system must be able to provide the list of daily events scheduled by the user
- The events must be divided between “Up coming” and “Later on”
- Information of event’s description, calendar saved and time must always be shown, while the location of the event is optional
- Button add event must always be visible
- [D2] Up coming events’ section needs to contain appointments planned to happen 2h before

### **3.2.3 [G2] Allow people to view previous events**

- A calendar screen must be shown
- Information of event’s description, calendar saved and time must always be shown, while the location of the event is optional
- Button add event must always be visible
- [D3] Transportation information are not shown for the past events

### **3.2.4 [G3] Allow people to view the detail of each event**

- ETA and trip information must be provided
- Tickets services must be shown
- Different suggested routes can be explored, if available
- The system must provide a way to buy tickets or book a sharing service vehicle
- The system must show the purchased ticket
- The paths color in the map changes accordingly to the means of transportation

- [D4] Indications on the ETA, calculated mean of transportation and time-to-leave are calculated on the current approximative position
- [D5] User's position is retrieved using GPS system
- [D6] Position is re-calculated every 2 hours, or when a significant change in GPS position is discovered
- [D7] The map can be browsed by the user
- [D8] The localization service is available and functionally working
- [D9] The map service is reliable

### **3.2.5 [G4] Allow the users to create an event**

- Information of event's description, calendar saved and time must always be provided, while the location of the event is optional
- The system must ask for the description and the location of the appointment
- The system must ask whether the event should be time-flexible. In this case it must require a time window and how long the appointment will last
- Button for repeating an event weekly must be shown
- Button for choosing the calendar in which the event will be saved must be shown
- A multiple choice list of available means of transportation must be included
- [D10] The button to choose the calendar is, by default, the last calendar chosen, or the first calendar if the user has never added an event
- [D11] While creating an event, the system must take into account whether the eco mode has been turned on

### **3.2.6 [G5] Allow people to see daily events on a map**

- All the events must be shown on a interactive map
- The map must show a path to the next event
- [D12] The map can be browsed by the user
- [D5] User's position is retrieved using GPS system
- [D9] The map service is reliable

### **3.2.7 [G6] Allow the user to set preferences**

- The system must provide of a way to change the maximum distance that could be traveled by bike or walk
- The system must provide a time window in which public transits can be taken
- All the available calendars must be shown
- The system must allow to toggle the available external sharing services
- [D13] Every preference agrees with the actual record saved in the database
- [D14] Allowed services have been preinstalled

### **3.2.8 [G7] Allow the users to create calendars**

- The app should allow the user to add a new calendar
- The system allows to change the name of a calendar and eventually its color
- [D15] The user has always at least one calendar

### **3.2.9 [G8] Notify the user that it's time to leave for the appointment**

- The app must send a notification when it's time to leave for the appointment.
- The system must take into account the means of transportation that the user has selected
- [D5] User's position is retrieved using GPS system



### 3.3 Definition of use cases

#### 3.3.1 CreateCalendar

Allow the user to create a new calendar with a name and a color chosen by the user

Name	CreateCalendar
Actors	User
Goals	[G7]
Input conditions	None
Flow of Events	<ol style="list-style-type: none"><li>1. The user selects “Create Calendar” from the “settings” view</li><li>2. A pop-up is shown with a name input</li><li>3. The user chooses the calendar name and its color</li><li>4. The system records the new calendar in the database</li></ol>
Output conditions	The calendar is saved in the database
Exceptions	None

### 3.3.2 AddStandardEvent

Allow the user to insert an event in the calendar

Name	AddStandardEvent
Actors	User
Goals	[G4]
Input conditions	None
Flow of Events	<ol style="list-style-type: none"><li>1. The user selects “Add event” from the “schedule” view</li><li>2. A pop-up is shown with a form to fill out</li><li>3. The user chooses the location and a description</li><li>4. The user selects the starting and ending date-time</li><li>5. Eventual repetitions during the week and type of calendar are chosen</li><li>6. Preferred means of transportation are selected</li><li>7. The system checks if the appointment can be reached in time with the user’s preferred vehicles</li></ol>
Output conditions	<ul style="list-style-type: none"><li>• The event is added to the calendar</li><li>• Flexible events are rearranged to fit the schedule</li></ul>
Exceptions	Users are notified immediately with a warning if their event will not be reached on time

### 3.3.3 AddFlexibleEvent

Allow the user to insert a flexible event in the calendar

Name	AddFlexibleEvent
Actors	User
Goals	[G4], [G4.3]
Input conditions	None
Flow of Events	<ol style="list-style-type: none"><li>1. The user selects “Add event” from the “schedule” view</li><li>2. A pop-up is shown with a form to fill out</li><li>3. The user choose the location and a description</li><li>4. The user toggle the switch for a Flexible Event</li><li>5. The user selects the starting and ending date-time window</li><li>6. The user choose the duration of the event</li><li>7. Eventual repetitions during the week and type of calendar are chosen</li><li>8. Preferred means of transportation are selected</li><li>9. The system checks if the appointment can be reached in time with the user’s preferred vehicles</li><li>10. The system rearrange this and other flexible event in order to fit the schedule</li></ol>
Output conditions	<ul style="list-style-type: none"><li>• The event is added to the calendar</li><li>• Flexible events are rearranged to fit the schedule</li></ul>
Exceptions	Users are notified immediately with a warning if their event will not be reached on time

### 3.3.4 EditPreferences

Allow the users to edit their settings: preferred vehicles (cars, bikes, public transportation, trains, sharing services), depending on the weather, if help low-

ering carbon emission or not, and specify a preferred lunch time

Name	EditPreferences
Actors	User
Goals	[G6]
Input conditions	None
Flow of Events	<ol style="list-style-type: none"><li>1. The user selects the “Settings” tab</li><li>2. A window with all the available settings is displayed</li><li>3. The user chooses preferences</li><li>4. The settings are automatically saved</li></ol>
Output conditions	The preferences are saved into user’s profile
Exceptions	None

### 3.3.5 BuyTickets

Allow the user to buy public transportation tickets directly from the application.

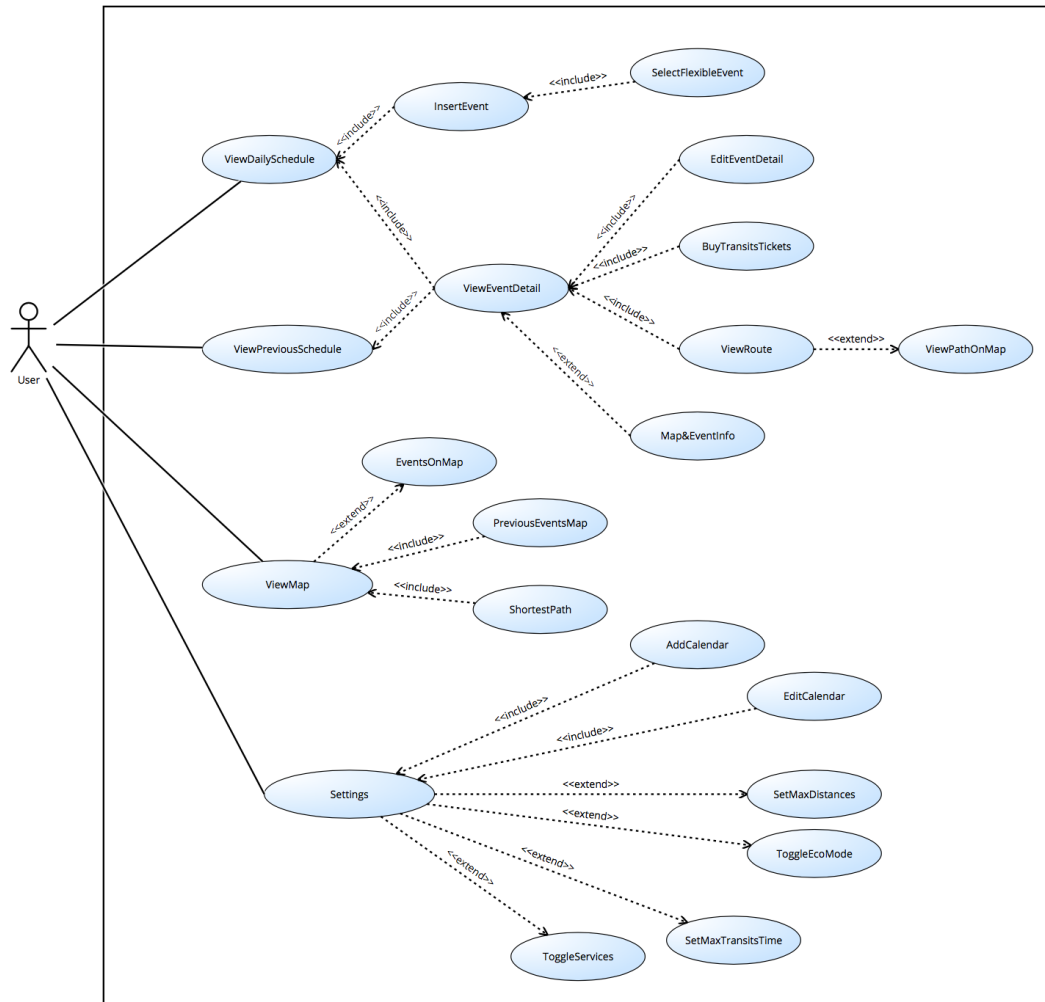
Name	BuyTickets
Actors	User
Goals	[G3.2.1]
Input conditions	<ul style="list-style-type: none"><li>• The event has already been created</li><li>• The user is seeing a suggested route</li><li>• The suggested route has a public transit option</li></ul>
Flow of Events	<ol style="list-style-type: none"><li>1. The user taps on the “Purchase ticket” button</li><li>2. The text message app is displayed in foreground with a predefined text and the local transit company number as receiver</li><li>3. The user sends the text message</li><li>4. The user receives a reply from the local transit company with a PNR code</li><li>5. The text message sent from the local transit company is read by the application</li></ol>
Output conditions	The ticket is recorded in the database
Exceptions	The purchased is aborted if the user doesn’t have enough credit on the phone

### 3.3.6 NotifyUser

Calculate the best lunch time reserving at least 30 minutes a day from the 11.30am to the 2.30pm

Name	CalcLunchTime
Actors	No active actors
Goals	
Input Conditions	
Flow of Events	<ul style="list-style-type: none"><li>• <b>Initiation:</b> When an event is registered, the system checks if there is a period of time in which the user can have lunch.</li><li>• If the system doesn't find it, an exception is thrown.</li><li>• <b>Termination:</b> The system finds (or not) the lunch solution</li></ul>
Output conditions	
Exceptions	If no lunch solution is found

### 3.4 Use Case Diagram



### 3.5 Performance Requirements

The system must be able to fulfill a big number of requests in the shortest time possible. Approximately a request should not last more than a second.

The server database should be fast enough to store and search data in order of milliseconds even if million of records are stored.

The server should be located near the maximum concentration of final users, in this case in Lombardy so that network latency will be minimized.

The application should not present any kind of lag in any situation and should never crash.

## **3.6 Design Constraints**

### **3.6.1 Standards compliance**

The app must comply to the standard “iOS Human Interface Guidelines” and “App Store Review Guidelines” by Apple.

Regarding the interconnections between the app and the server the system will use some APIs which will be REST-Compliant and they should output data using XML or JSON.

### **3.6.2 Hardware limitations**

- Mobile App
  - iOS Smartphone
- Connectivity
  - 2G, 3G or LTE
  - GPS

### **3.6.3 Any other constraint**

The system will ask the user permissions to get her / his current position.

The user won’t require login because it will use the iCloud unique identifier in order to synchronize events and calendars to the server.

## **3.7 Software System Attributes**

### **3.7.1 Reliability**

The system must guarantee a 24 hours per day and 7 days per week service.

The server should never be offline.

Very small deviations will be accepted only if the system will go in maintenance mode.

### **3.7.2 Availability**

### **3.7.3 Security**

The app will use the iCloud unique identifier to identify the user also when he / she uninstalls the app and reinstalls it.

This identifier is kept secret on the iCloud Keychain and won’t be stored on the server for promotional or ambiguous purposes. It is just needed to identify the user.

Security of the user data is the primary concern.

The app will communicate with the server via SSL certificates enabling HTTPS secure requests.



#### **3.7.4 Maintainability**

#### **3.7.5 Portability**

### **4 Formal analysis using Alloy**

### **5 Effort spent**

- Filippo Calzavara: 20.00h
- Giovanni Filaferro: 20.00h
- Benedetto Maria Nespoli: 20.00h

### **6 References**

- iOS Human Interface Guidelines (<https://developer.apple.com/ios/human-interface-guidelines/>)
- App Store Review Guidelines (<https://developer.apple.com/app-store/review/guidelines/>)