Politecnico di Milano

AA 2017-2018

POLITECNICO
MILANO 1863

Computer Science and Engineering

Software Engineering 2

RASD

Requirement Analysis and Specification Document

version 1.0 – 26.10.2017

Table of Contents

1. Introduction

1.1.Purpose

The purpose of this document is to provide a detailed description of the Travlendar+ system. This will be done by a detailed presentation of the proposed solution and its purpose, listing its goals, and the requirements and assumptions through which they will be achieved. The document is meant to be used by the clients, users and also by the parties designated with the task of creating the specified system, mainly the system and requirements analysts, the project managers, software developers and testers.

The Travlendar+ system is designed as a software application used for making the users' schedules, be it daily, monthly or even yearly ones, and then provide support in their realization. Scheduling your time can be hard, especially if you are a busy person, living in a big city and have to go to many different locations in a single day. The Travlendar+ is designed to be a calendar-based application that also serves as a route planner, making it easy to see and choose the best route and transport means suited for the user. While obliging to the users' preferences, the application is tasked with finding the best means of transport suited to the type of the meeting scheduled and the current situation in the city of residence. Perhaps most importantly, the application is designed to be able to check and validate the given schedule, providing the client with the information on what is the optimal way of completing it. On the other hand, the application should warn the users if their schedules are too optimistic and not able to be completed without any tardiness. Given the fact that the goal of the system is to make life a bit easier in general for its users, a feature for lunch, and break in general, scheduling is also added, making it possible to specify the preferred time for the activity so that the schedules can be tweaked and checked so it is included in the daily events. Also, the application is designed to be able to provide the user with the chance to purchase tickets of the desired public transport means or the location of the nearest service, offering the selected means of travel. In a sense, the Travlendar+ system is envisioned as an all-in-one service for all meeting scheduling and getting to meeting, whether they are formal or the most informal ones.

1.2.Scope

1.2.1. Description of the given problem

As already mentioned, the Travlendar+ system is designed to provide the users with an overview of their schedules and to compute routes using the available transport means. Among the computed routes, some should be marked as more appropriate than others. This should be done in accordance with the information given on the meetings, because some transport means are more suited for specific types of meetings and locations. The users' preferences should be taken into account when suggesting routes, as well as specific conditions related to the current state of the area in which the route is located. This is to say that a bicycle should not be suggested if there are signs of bad weather or a line in the public transport should not be considered if the route is perhaps changed for a certain period due to road works, for an example. The users' preferences can be divided into two groups: long term preferences and short term ones. The former group, as the name suggests, contains the preferences that should not

be changed as often, specifying a single user's desires on all trips and route selections (for an example, the preference of one means of travel as opposed to another, the removal of a certain travel means etc.). The latter group is composed of those designed to be able to accommodate the user's desire on the way to complete a specific trip (fastest option, shortest option etc.).

While calculating the optimal route, the application (or the external service responsible for the task) will consider average times calculated from the available data. It will not be able to guarantee a specific time of arrival, only estimation, if the suggested route is followed. In order to provide the information on tickets, service locations and other transport related subjects, the system will interact with available public transport services, used in the place of residence of the user.

The point of the addition for break scheduling is to make it easier for the user to include a time off during the day in the free slots of time in between the already scheduled meetings and appointments. The most common usage is anticipated to be lunch scheduling, but other options will be included while also making it possible for the user to create specific beak types for them. For every daily break scheduled, with its time period and duration specified, Travlendar+ will try to find room for it in the stated schedule by the user. If that cannot be done, the application will warn the user that the currently specified schedule is not taking into an account the user’s anticipated rest time. In that way, the user is relieved of manually trying to take into consideration the time needed for a break during the day.

1.2.2. Goals

[G1] - The user can be recognized by providing a form of identification.

[G2] - Allow the user to specify meetings in their schedule.

[G3] - The user can see the schedule with the time periods for the meetings and the estimated time needed to get to them, while being notified if some meetings can't fit into the schedule or are not reachable in time.

[G4] - The user is provided with multiple routes to reach every meeting.

[G5] - The application allows the user to specify preferences that should be taken into account while calculating and suggesting the routes.

[G6] - The routes provided to the user take into account external factors that may affect them during the travel time.

[G7] - Allow the user to specify breaks that should be considered in the schedule, in between meetings, in a certain time period.

[G8] - The user can buy public transport tickets through the application for trips that include these means of transport.

1.3.Definitions, Acronyms, Abbreviations

1.3.1. Definitions

- **Meeting**: an event scheduled by the user by providing the necessary information and added to the calendar.
- **Route**: a path that is suggested to the user, considering different transport means in correspondence with the user preferences
- **Transport mean**: a transport service that is available in the city where the application is used; it can include the public transport, car sharing services, bike sharing services etc.
- **Break**: time slot in the day reserved for a break, not allowed to overlap with meetings.
- **Long term preferences:** Preferences set by the user about the preferred means of transport (generally and for each meeting type), the maximum and minimum distance for each mean of transport, information about owning a monthly/annual public transport card, and avoiding a specific mean od transport in a specific time.
- **Short term preferences:** Possibility to choose among different routes offered by the system such as: shortest route, fastest route, a route that minimized carbon footprint etc.
- **Obstacle**: an unplanned event that can cause delays in the trip, such as: strikes, roadwork etc.
- **Payment**: A service provided to the user that allows him to buy tickets or pay for other kinds of transport services.
- **Warning:** A message shown to the user when the input does not correspond with the system requirements.

1.3.2. Acronyms

- RASD – Requirement Analysis and Specification Document
- ETA – Estimated time of arrival
- API - Application Programming Interface
- GPS - Global Positioning System

1.3.3. Abbreviations

- [Gn]: n-th goal
- [Dn]: n-th domain assumption
- [Rn]: n-th functional requirement

1.4.Document Structure

Chapter 1 gives an introduction to the problem and describes the purpose of the application Travlendar+. The scope of the application is defined by stating the goals and description of the problem.

Chapter 2 presents the overall description of the project. The product perspective includes details on the shared phenomena and the domain models. The class diagram describes the domain model used, and the state diagram analyzes the process of arranging a meeting and reaching it in time. Here the majority of functions of the system are more precisely specified, with respect to the already mentioned goals of the system. In the user characteristics section the types of actors that can use the application are described.

Chapter 3 contains the external interface requirements, including: user interfaces, hardware interfaces, software interfaces and communication interfaces. Few scenarios describing specific situations are listed here. Furthermore, the functional requirements are defined by using use case and sequence diagram. The non-functional requirements are defined through performance requirements, design constraints and software system attributes.

Chapter 4 includes the alloy model and the discussion of its purpose. Also, a world generated by it is shown.

Chapter 5 shows the effort spent by each group member while working on this project.

Chapter 6 includes the reference documents.

2. Overall description

2.1.Product perspective

The Travlendar+ system is designed to be a completely new software application. It is designed to use already proven and tested services for its critical tasks. The system is intended to be used as a mobile application. In order to provide the newest and most reliable information for the public transport aspect, official APIs of the public transport providers will be used as much as possible. A more in detail analysis will be provided in later chapters.

Regarding the Travlendar+ system itself, to better describe the domain model used, the following diagram is provided:

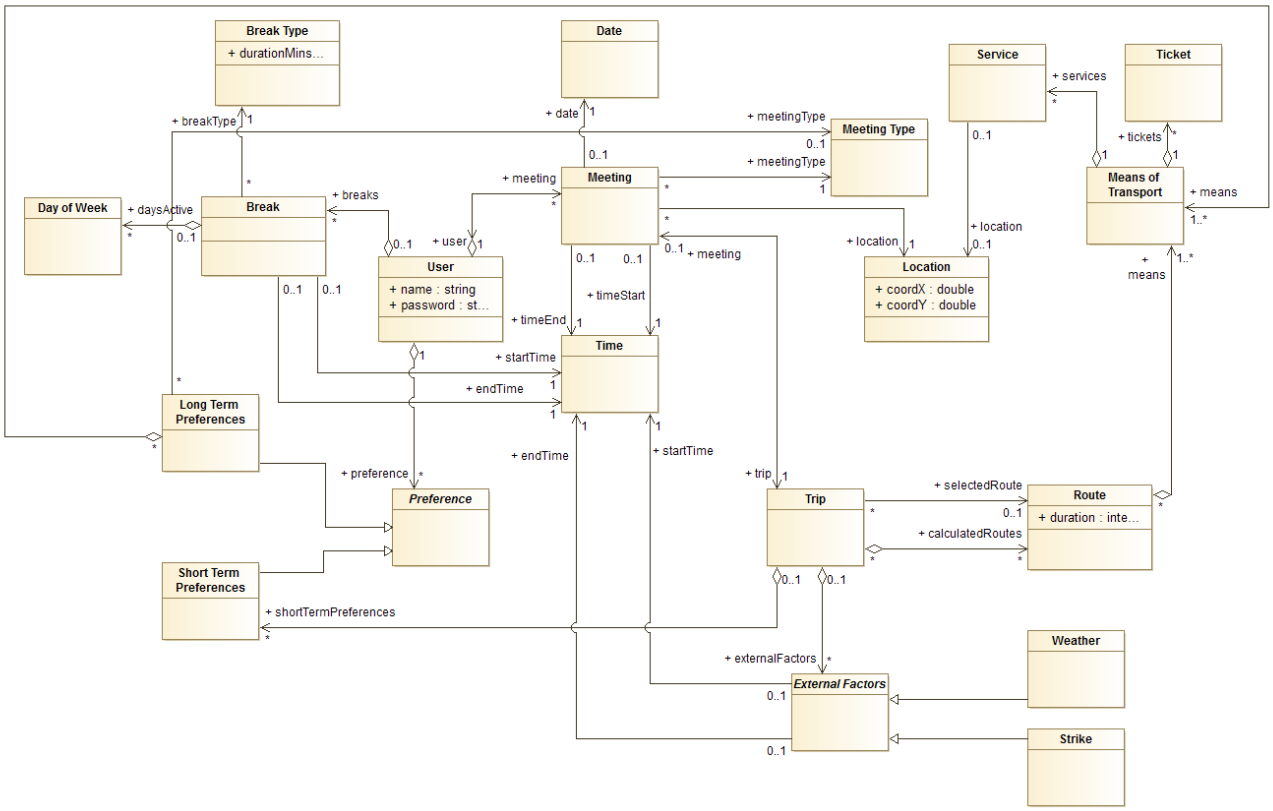

Figure 1 Class diagram

As we can see, the focal point of the system is the Meeting. A large portion is also taken by the Trip and Preference sections, while trying to be as general as possible to provide an opportunity to specialize the application to every single user and his/her needs and desires.

In the next diagram, the process of arranging a meeting and reaching it in time is analyzed through its states.

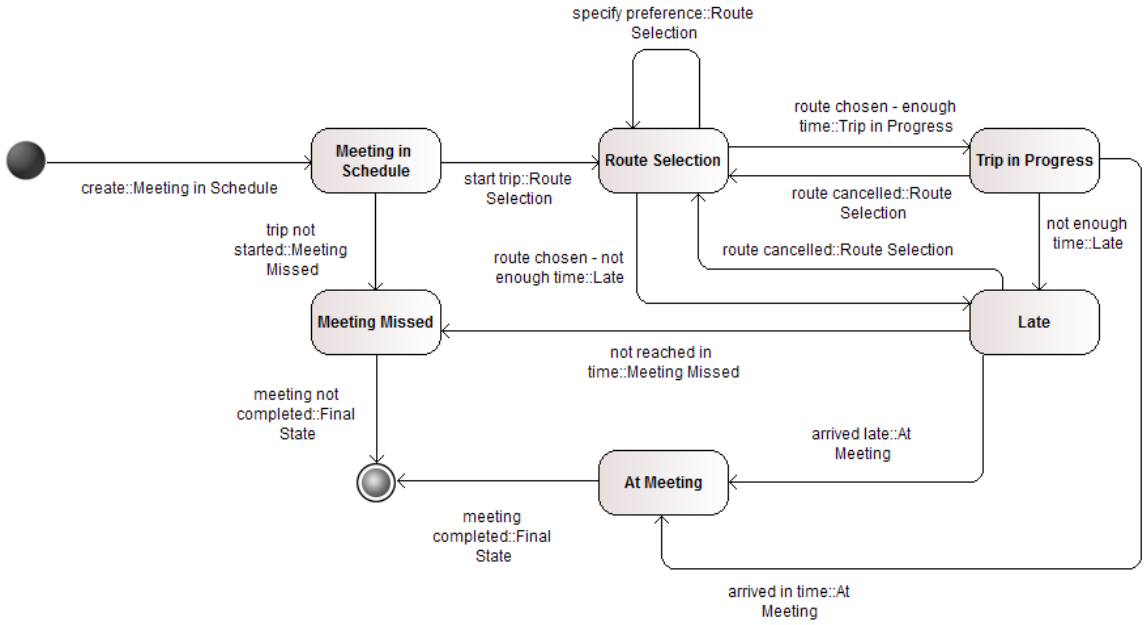

Figure 2 State diagram

After scheduling a meeting (it is assumed that there is no overlapping and the user is able to create a generic meeting in this example, as to be more generic), we can see all of the possible outcomes that are the repercussions of the user's actions. It is worth mentioning again that the time calculations and ETAs are expected to be accurate only if the user follows the provided and selected route, using the specified means of transport.

2.2.Product functions

Considering all of the presented goals of the Travlendar+ system, the majority of functions of the system can be divided into 4 groups. In the following section, they are listed and more precisely specified, with respect to the already mentioned goals of the system.

2.2.1. Meeting management

This requirement is the essence of the whole system. The user will be able to create a meeting, specifying the period during which the arranged meeting will take place, its location and type. All of the users meetings will be represented in a calendar view, for ease of use and a better and more functional visualization of the schedule. Whenever the user will add an appointment to the schedule, Travlendar+ should check the rest of the meetings scheduled for that day and calculate whether the appointment is reachable or not, using the fastest route possible. The user will be warned if the appointment is not

keepable or if the system has checked that the specified schedule is hard to accomplish because only a low number of routes will be able to take the user to the desired meetings in time.

2.2.2. Route and preference management

In order to get directions to the meeting location, the user will have to choose a route. The system will offer a wide range of possible routes, using the transport means at disposal. The user can specify in what way he/she wants to reach the destination, in order for the system to tweak the offered routes and suggest which one will fit in best with the stated desires.

The users will be able to specify global preferences (long term ones) in which way they would like to reach their destination. Some examples include specifying which transport means are not to be used and which ones are preferable. They can define even more specific ones like - no walking distance larger than a specified value. The other type of preferences are the ones specific to a single meeting (short term ones), defined only once, during the route selection. They could be used for a single case or to filter the suggested possible routes. Some examples include shortest route, fastest route, stating that the suggested route should be the one that minimizes the carbon footprint etc.

While displaying the possible routes to the user, the system should also take into account external factors that may affect certain routes and means of transport. One example could be bad weather, in which case routes using bicycles or longer walks would not be considered. Another one would be public transport strikes.

2.2.3. Trip arrangement

While suggesting the best route to reach a destination to the users in accordance to their preferences, if the selected route uses a public transport means, Travlendar+ should offer the user the chance to buy the required ticket online. Because a possibility exists that the user may already possess a pass, the system will let the user to specify it. In this way, it can take it into account so that the ticket buying is not offered and that transport means for which the pass is valid are more used in the suggested routes.

If the selected route contains the usage of a vehicle sharing service, the application will be able to provide the user with the nearest location that offers the stated service.

If the route is scheduled few days in advance and is changed by the time the user should take it, he/she will be notified.

2.2.4. Flexible breaks

One of the additional features that Travlendar+ offers is the possibility to specify a time period during the day for a break activity, its duration and the days in a week during which it will be considered. In that way, the application will try to fit it in the daily schedule and will warn the user if that is not possible. The users will be able to choose predefined types of breaks, most commonly "lunch break".

2.3.User characteristics

The following actors are the users of this application:

- User:

A person that is successfully registered to Travlendar+ and is able to use all the Travlendar+ services to plan his/her events.

- Administrator:

An employee of Travlendar+ that is maintaining and updating the system. The administrator does not have to register, since he is added during system's installation process.

2.4.Assumptions, dependencies and constraints

2.4.1. Domain assumptions

[D1] - The usernames used in the system are unique to every user.

[D2] - The device the user is running Travlendar+ on can provide an accurate enough current location.

[D3] - The data provided by external services regarding the time needed to complete a route is assumed to be correct.

[D4] - The user follows the specified directions on the selected route, without deviations or unexpected stops.

[D5] - The user will use one of the supported travel means in the application.

[D6] - The application has access to the information on the transport means in the city of use.

[D7] - There are external services of the public transport companies of the city of usage that offer the possibility to buy tickets online.

[D8] - There is an external service that will be in charge of the payment information validity and the secure payment transactions.

[D9] - The confirmation of payment (in any form) sent to the user's device can be used to access public transport means in a same way a basic ticket can.

3. Specific requirements

3.1.External interface requirements

3.1.1. User interfaces

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

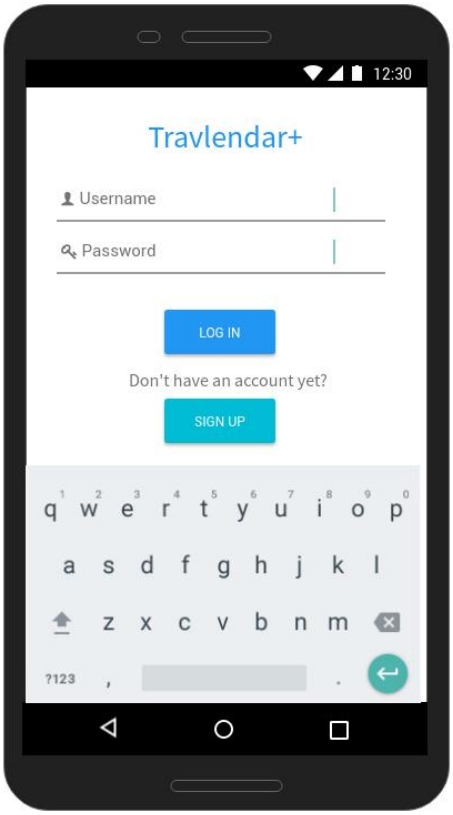

Figure 3 Mock up – Log in/Sign up screen

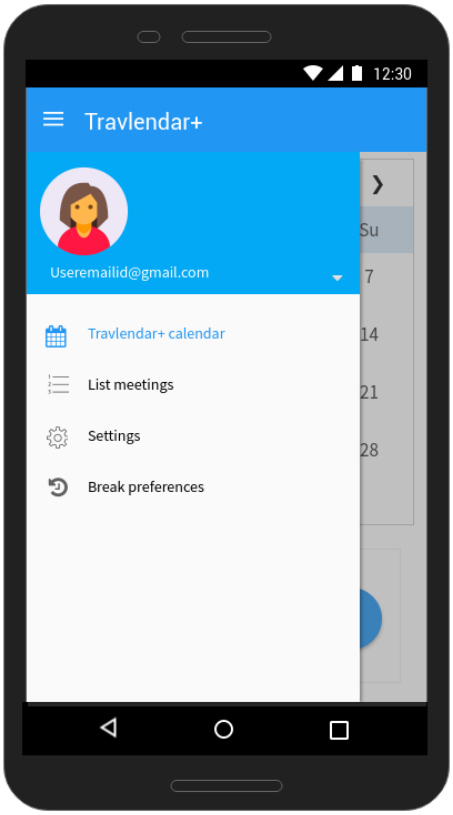

Figure 4 Mock up - Menu

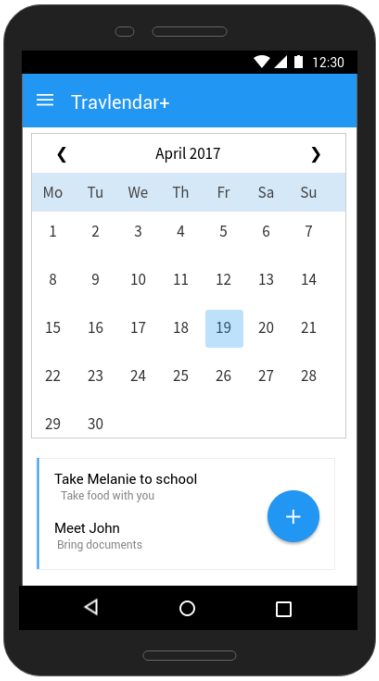

Figure 5 Mock up – Calendar view

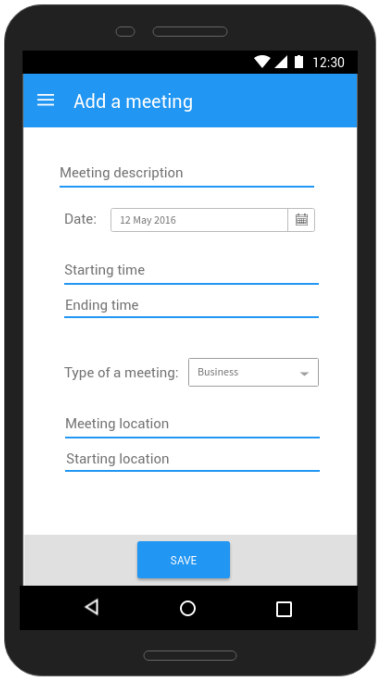

Figure 6 Mock up - Add a meeting

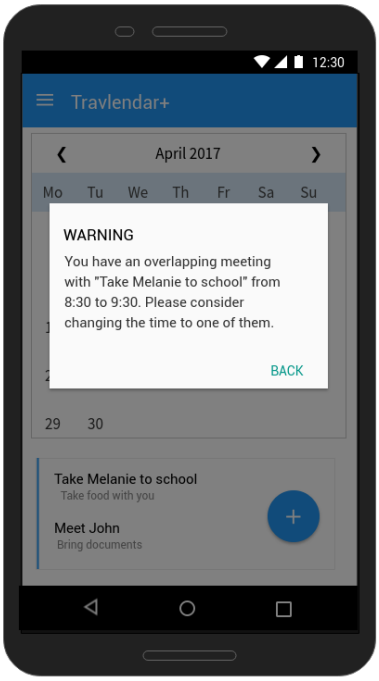

Figure 7 Mock up - Warning message

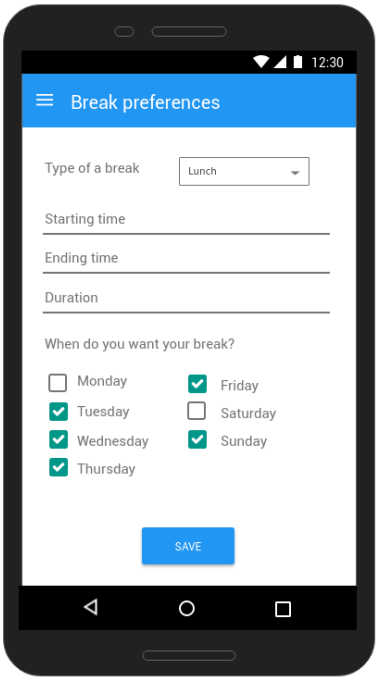

Figure 8 Mock up - Break preferences

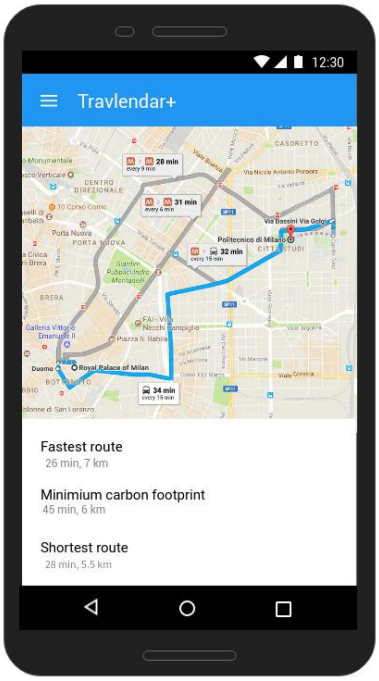

Figure 9 Mock up – Map of the route

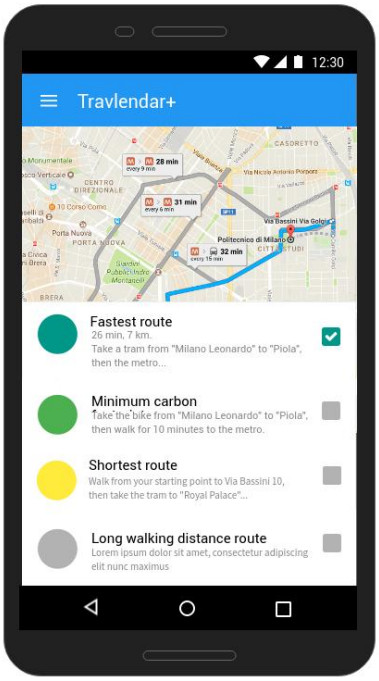

Figure 11 Mock up – Route selection

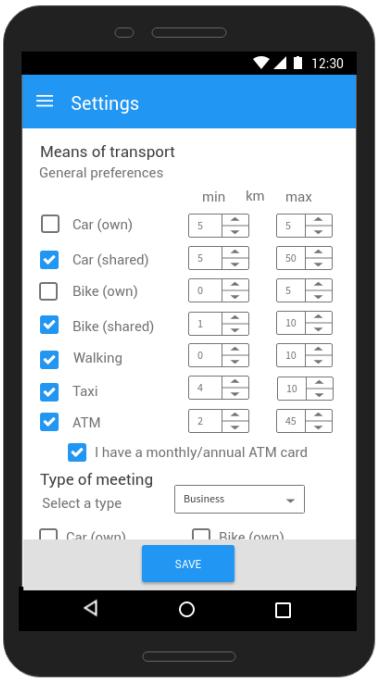

Figure 12 Mock up - Settings

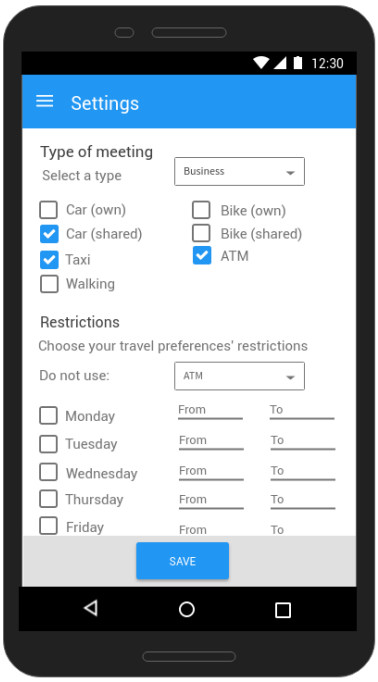

Figure 10 Mock up – Settings

3.1.2. Hardware Interfaces

This is a software application whose main function is to plan daily meetings and a way of transportation to reach them. It is not using any hardware interfaces; however, it requires using a smartphone or a web browser that can use GPS services. Bluetooth might also be required for some of the functionalities (for example, using an external service as Mobike requires Bluetooth to unlock a bicycle).

3.1.3. Software Interfaces

The application is using external services that are suitable for embedding in order to simplify the overall architecture and make it more lightweight.

- Calendar

Since the idea of this application is calculating and planning the daily trips and meetings of the user, it requires a calendar. There are several solutions to this requirement. For example, the Google Calendar API lets you display, create and modify calendar events as well as work with many other calendar-related objects, such as calendars or access controls. An optional solution is using a library as UI-Calendar.

- City maps

  The application requires a usage of map of the city it is planned for. One option here is using Google Maps, as the data gives accurate real-time information for mapping, navigation and places. It also provides the users with the reliable location information they need throughout the world. Using this API will help in building better user experiences with data based on real-time mapping and traffic signals. An alternative here would be using another map, depending on the city the application is being developed for.

- Car-sharing services

The car sharing services are a good possibility for getting around any city that offers them. It is a good option for rainy days, as well as during strikes. This service also varies from one city to another, but in Milan there are few good options that can be considered. One of them is Car2go, which is a point to point car sharing service with flexible credits and rental rates. Enjoy is another car sharing system that offers vehicles with one-way point-to-point rentals charged by the minute. Cars can be rented wherever parked.

- Bike-sharing services

Bike-sharing services are usually cheap and very convenient. If the city has this option as a transport mean, it can be included in the application, and it will also help with finding more routes that minimize carbon-footprint. In Milan there are few options. Mobike is a bicycle sharing service that allows the user to find the nearest bike using GPS, scan the code and ride it until it is manually locked. It is a convenient service to be integrated in this application as it can be used from one point to any other. Another service

to be considered is BikeMi. This bicycle needs to be returned to one of the marked BikeMi stations, so the application has to consider this also. Ofo is another bike sharing service in Milan. The user can scan the code of the nearest bike and ride it to the desired destination.

- Public transport

The most often used transport means are the ones that come from the public transport. In Milan for example, there are more options to be considered. The user can use the metro as it is usually faster than the other ground public services. The bus/tram/train is also another possibility.

When it comes to using the application for these choices, there will be two options for using public transport: either the user will have an annual or monthly subscription to these transportation services, or the application will allow the user to buy a ticket by using the ATM application API.

3.1.4. Communication Interfaces

This network-connected Android app uses HTTP to send and receive data. The Android platform includes the HttpsURLConnection client, which supports TLS, streaming uploads and downloads, configurable timeouts, IPv6, and connection pooling.

In order to make a connection to one of the Google APIs provided in the Google Play services library (such as Google Maps), an instance of GoogleApiClient ("Google API Client") needs to be created. The Google API Client provides a common entry point to all the Google Play services and manages the network connection between the user's device and each Google service.

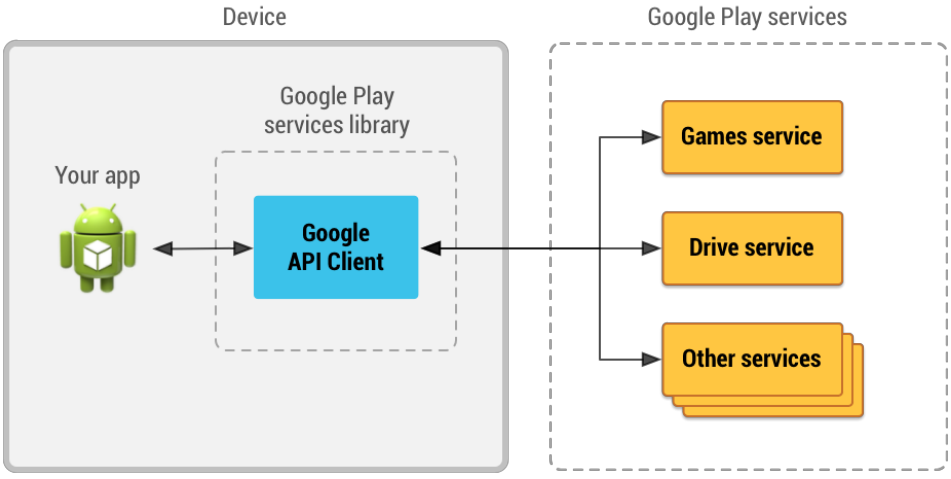

Figure 13 Communication interfaces - APIs

This illustration shows how the Google API Client provides an interface for connecting and making calls to any of the available Google Play services such as Google Play Games and Google Drive.

3.2. Scenarios

3.2.1. Scenario 1

Professor Bianchi needs to schedule a meeting with Martin, his former student, in order to discuss with him the opportunity for a start-up. Martin preferred Thursday evening as an appropriate time to meet, so professor Bianchi launched Travlendar+ on his Android and created a meeting on Thursday evening, at 6:00 p.m., at Nabucco restaurant. Once he added the meeting, Travlendar+ created a warning that he is not going to be able to reach Nabucco at 6:00 p.m. since he is having classes until 5:45 p.m.. Therefore, professor Bianchi and Martin postponed the meeting and agreed to meet at 7:00 p.m.

3.2.2. Scenario 2

Sarah and Jessica are colleagues and they arranged to meet today at 8:00 p.m. at Piazza del Duomo and then to go somewhere for a dinner. Sarah was 20 minutes late and she complained that she always chooses the wrong way of transport which takes her more time to reach a place. Jessica recommended her Travlendar+ application which can help her choose the most appropriate type of transport service to use. Sarah immediately downloaded the application, registered to Travlendar+ and added some details in her user preferences section:

- she unchecked the option "I have a driver's license" because she is not a driver, so driving is deactivated for her
- she checked the option "I have annual urban travel card", so she will not be offered to buy a ticket
- Sarah started actively using Travlendar+ and next time they met, she arrived even earlier than scheduled.

3.2.3. Scenario 3

Mario is a programmer, working remotely from home. He desperately wants to start doing some sport activities because of his sitting work. Since he is very busy with his work and he is currently working on additional project, he really doesn't have time to travel to the closest gym which is 22km far from his village. Instead, he decided running every day except Monday and Thursday, because in those days he has meetings with his partners. He wants to run in the interval between 7:30 p.m. and 10:30 p.m. for at least 40 minutes through his village. He added this plan in his "Travlendar+" as a break preference, meaning a break from work which should repeat as a regular habit. "Travlendar+" started reminding Mario for his "break" from the next day on and so he started running.

3.2.4. Scenario 4

Alex wants to visit his friend Samath who lives on the other side of Milan. He adds his meeting to Travlendar+, scheduling it for Saturday morning at 11 o'clock. After Travlendar+ suggests him bunch of routes, he chooses the one which is shortest. He chooses a route which first suggests him to take a bike in order to get to a metro station of the violet line and then to take a metro. After choosing the route,

Alex is asked if he wants to buy a public transport ticket and if he wants to top up for the bike sharing. Since Alex doesn't have monthly ATM card and he has spent the credit on his 'Mobike' account, he immediately buys ATM ticket and puts money on his 'Mobike' account. Now, Alex is fully prepared to visit Samath.

3.2.5. Scenario 5

Mike goes to gym every Wednesday from 7:30 p.m. to 9:00 p.m. When registered to Travlendar+, he selected combinations of transportation means that minimize carbon footprint as a preferred option. Because of this, every time when it is possible, Travlendar+ suggests him to ride a bicycle. However, this Wednesday when he opened Travlendar+, the suggested option for him was to catch a tram from work in order to arrive at the gym as soon as possible. This suggestion was offered due to the weather forecast which was stating that it is going to rain between 7:00 p.m. and 8:00 p.m. o'clock in the evening.

3.3.Functional Requirements

[G1] - The user can be recognized by providing a form of identification.

[D1] - The usernames used in the system are unique to every user.

[R1] - The user can create an account for the usage of Travlendar+, by selecting a username and a password.

[R2] - The user can log in to the application by providing the combination of a username and a password that match an account.

[D7] - There are external services of the public transport companies of the city of usage that offer the possibility to buy tickets online.

[D8] - There is an external service that will be in charge of the payment information validity and the secure payment transactions.

[D9] - The confirmation of payment (in any form) sent to the user's device can be used to access public transport means in a same way a basic ticket can.

[R21] - If the selected travel route includes public transport means, the user is offered to buy the tickets online, through the application.

[G2] - Allow the user to specify meetings in their schedule.

[R3] - The user can create a meeting in their schedule by stating the meeting name (to distinguish it), its type, the starting and ending time and the location of the meeting.

[R4] - The user may provide a starting position for the trip to the meeting location when specifying the meeting, in order to calculate the time needed to reach the location.

[R4.1] - If the starting point is not specified, the system will use the location of the meeting previous to the newly specified one (if such exists).

[R5] - The schedules with the meetings can always be seen by the user.

[G3] - The user can see the schedule with the time periods for the meetings and the estimated time needed to get to them, while being notified if some meetings can't fit into the schedule or are not reachable in time.

[D2] - The device the user is running Travlendar+ on can provide an accurate enough current location.

[D3] - The data provided by external services regarding the time needed to complete a route is assumed to be correct.

[D4] - The user follows the specified directions on the selected route, without deviations or unexpected stops.

[R6] - For any new meeting added to the schedule, the system checks if there is any overlapping with other appointments.

[R6.1] - If an overlap is detected, a warning is sent and the meeting is not created.

[R7] - After creating a meeting, the user can select which route they would prefer to take to the meeting (as an initial route).

[R7.1] - If the user doesn't specify the route, the fastest route is chosen as the default one.

[R8] - Using the information about the selected initial route, the system checks whether the meetings in the schedule can be reached without being late.

[R8.1] - If it is calculated that a meeting can't be reached in time, a warning is sent.

[G4] - The user is provided with multiple routes to reach every meeting.

[D5] - The user will use one of the supported travel means in the application.

[D6] - The application has access to the information on the transport means in the city of use.

[R9] - The system calculates multiple routes, for every transport means.

[R10] - The user is shown the possible routes, with the distances and the ETAs.

[R11] - The user is allowed to choose a route out of the provided ones to be the selected travel route.

[R12] - The user can, at any time, see the possible routes and choose a different selected travel route.

[R12.1] - If the user has started the trip to the location of the meeting, the routes provided use the current location as the starting point.

[G5] - The application allows the user to specify preferences that should be taken into account while calculating and suggesting the routes.

[R13] - The user can specify preferences specific to the users account, which can be long term (for every trip) or short term (specific to a single trip).

[R14] - The user can state the preferred travel means, globally and/or to specific meeting types.

[R15] - The user can specify constraints to be taken into account while calculating the routes.

[R16] - The routes that are displayed are the ones that also satisfy the long term preferences stated by the user.

[R17] - While the routes are displayed, the user can specify some short term preferences as conditions to be used to filter and sort the calculated routes.

[G6] - The routes provided to the user take into account external factors that may affect them during the travel time.

[R18] - Depending on the external factor, a route affected by it is not considered, is recalculated and/or removed from the suggested routes.

[G7] - Allow the user to specify breaks that should be considered in the schedule, in between meetings, in a certain time period.

[R19] - The user can create a break, stating its type, the duration, the time period during which the break should take place and the days of the week in which the break should be taken into account.

[R19.1] - One of the predefined break types is "Lunch".

[R20] - The system will check each schedule to find if the specified breaks are possible in the given arrangement of meetings, including the time travel between meetings.

[R20.1] - If the system discovers that a specified break can't fit into a schedule, the user will be warned.

[G8] - The user can buy public transport tickets through the application for trips that include these means of transport.

[R22] - The user can provide payment information through which they will be charged for the purchase of tickets.

[R23] - The user is given a confirmation of payment after the purchase of tickets.

[R24] - The user can specify that they own a public transport pass, while also stating the period of the validity of the pass.

[R25] - If the selected route is using transport means for which an already specified pass can be used, the buying of the ticket is not offered to the user.

[R26] - If the user nonetheless tries to buy the ticket, the system will warn the user that a pass can be used.

3.3.1. Use case diagram

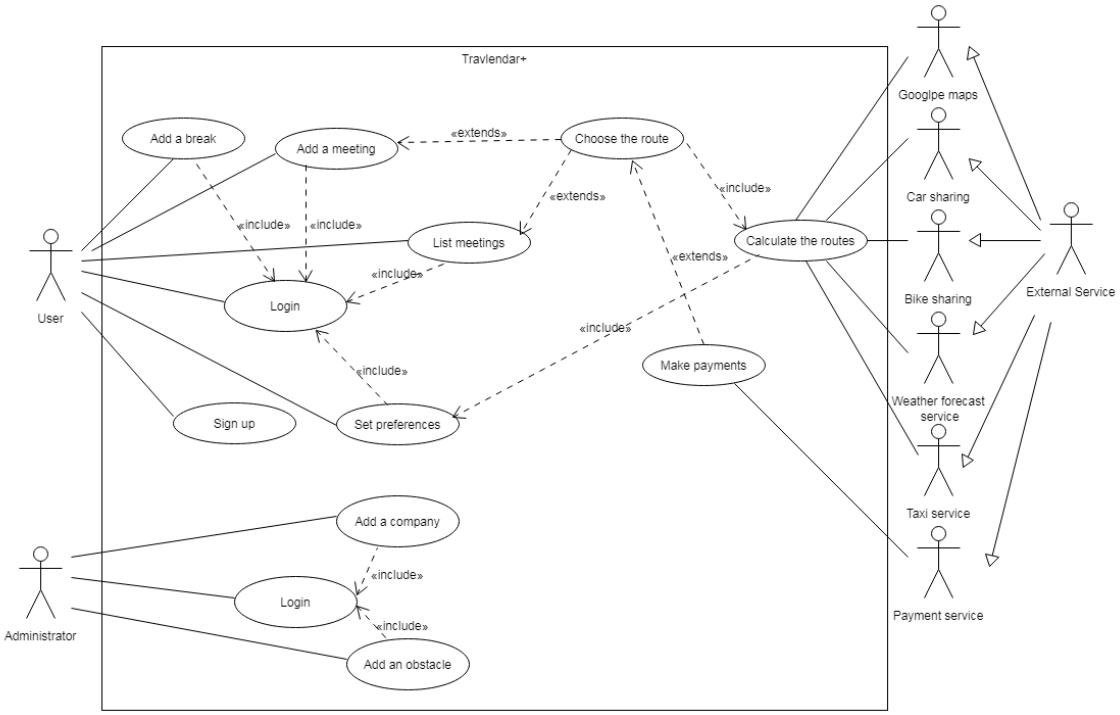

Figure 14 Use case diagram

Name	Sign up
Actor	User
Entry conditions	The user has installed the application on his/her device
Events flow	1. Click on “Sign up” button 2. Fill all the mandatory fields and provide the necessary information 3. Click on “Confirm” button 4. The system saves the data
Exit conditions	The user has successfully registered and now he’s able to use the application.
Exceptions	1. The user is already signed up 2. The user didn’t fill all of the mandatory fields with valid data 3. The username is already taken 4. The e-mail is already registered 5. All the exceptions are handled by notifying the user and taking him back to the sign up activity.

Name	Log in
Actor	User
Entry conditions	The user is previously successfully signed up and has the application installed on his/her device
Events flow	1. The user opens the application on his/device 2. He enters his credentials in the “Username” and “Password” fields of the home page of “Travlendar+” 3. The user clicks on the “Log in” button 4. The user is successfully logged in his/her “Travlendar+” and the system automatically redirects him/her to the calendar view
Exit conditions	The user is successfully redirected to the calendar view
Exceptions	1. The user enters invalid Username 2. The user enters invalid Password 3. All the exceptions are handled by notifying the user and taking him/her back to the login activity

Name	Add a meeting
Actor	User
Entry conditions	The user has already logged in
Events flow	1. The user clicks on the desired date for the appointment 2. The users clicks on the “+” button 3. Fill all the mandatory fields and provide the necessary information 4. Click on “Confirm” button 5. The system saves the data
Exit conditions	The user has successfully added a meeting to his calendar.
Exceptions	1. The user added a meeting in an already busy time slot. All the exceptions are handled by notifying the user and taking him back to the “Add a meeting” activity.

Name	Add a break
Actor	User
Entry conditions	The user has already logged in
Events flow	1. The user chooses the option “Break preferences” 2. The user chooses a type of break from the drop-down menu 3. The user checks the days of the week for which he/she wants to add a specific type of break 4. For the chosen days he specifies the time interval in which he wants to take the break and the preferred duration 5. The user clicks on “Save” button and the system saves the data
Exit conditions	The user has successfully specified break preferences to his “Travlendar+” account
Exceptions	1. The user added a break in an already busy time slot of a specific day in which a meeting is scheduled. All the exceptions are handled by notifying the user and taking him back to the “Break preferences” activity.

Name	Set preferences
Actor	User
Entry conditions	User has successfully signed up
Events flow	1. The user chooses settings option in order to set his/her general preferences 2. The user checks only those types of transport which he/she prefers. This means that only the chosen types of transport are going to be offered to the user. 3. For each type of transport chosen, he/she enters minimum and maximum distance for which that type of transport should be considered. 4. The user specifies if he/she has a monthly/yearly ATM travel card, so Travlendar+ knows if buying tickets should be offered to him/her. 5. The user specifies preferred means of transport for each available type of meeting.

	6. The user specifies days and hours in which a specific type of transport should be avoided. 7. The user clicks the “Save” button.
Exit conditions	User preferences are saved to his/her profile and are considered while calculating and offering routes to the user.
Exceptions	/

Name	Choose the route
Actor	User
Entry conditions	1. A meeting is created, providing all necessary information 2. There are appropriate routes calculated by Travlendar+
Events flow	1. The user chooses preferred route according to his short term preferences 2. After choosing the route, the user is provided with an option to buy public transport tickets or make a payment for transport sharing service if it is needed 3. The system saves the data
Exit conditions	The chosen route is saved in the meeting info
Exceptions	1. “Calculate route” did not execute properly

Name	Calculate route
Actor	Travlendar+
Entry conditions	1. A meeting is created, providing all necessary information 2. There is information available for the optional routes that are calculated by the provided external service for maps 3. There is information available for the weather forecast provided by the weather forecast service 4. There is information available for the available transport services 5. There is information available for any possible obstacles

Events flow	1. “Travlendar+” gathers all the information provided by external services and administrator 2. “Travlendar+” calculates and suggests routes in accordance with the user’s general (long term) preferences 3. The system saves the data
Exit conditions	The calculated routes are presented to the user
Exceptions	1. The user unchecked all transport means while creating his/her profile, so there is an appropriate warning notifying him about changing the settings. 2. The user has chosen a location that cannot be reached using the transport means offered in that particular city. A warning about changing the location is being showed.

Name	Make a payment
Actor	User
Entry conditions	The user has chosen a route that requires a payment
Events flow	1. The user chooses what kind of payment he/she wants to make 2. The user provides the system with all the required information for making a payment
Exit condition	The payment is successful
Exceptions	1. The user did not enter a valid data

Name	List meetings
Actor	User
Entry conditions	There exist meetings that have been created by the user
Events flow	1. After the user has opened his/her Travlendar+ profile, he/she chose to see the options 2. He chose the option “List meetings”

	3. The user can choose one of the meetings in the list and optionally can choose a route for it
Exit conditions	A list of all active meetings of the user appeared.
Exceptions	1. The user does not have any meetings added and he is notified that the list is empty.

Name	Log in
Actor	Administrator
Entry conditions	No entry conditions
Events flow	1. The user opens the application on his/device. 2. In the “Username” and “Password” fields of the home page of “Travlendar+” he/she enters his/her credentials. 3. He/she clicks on the “Log in” button. 4. The administrator is successfully logged in and the system automatically redirects him/her to the options menu.
Exit conditions	The administrator is successfully redirected to the options menu.
Exceptions	1. The administrator enters invalid Username. 2. The administrator enters invalid Password 3. All the exceptions are handled by notifying the administrator and taking him/her back to the login activity

Name	Add an obstacle
Actor	Administrator
Entry conditions	The administrator is successfully logged in.
Events flow	1. The administrator opened the options menu 2. He chose the option “Add an obstacle” 3. He entered the announced strike/roadwork/another obstacle in “Travlendar+” in order the system to be aware of the following

	delays and cancellations of public transport on the announced days in particular time intervals.
Exit conditions	The announced strike/roadwork/another obstacle is added.
Exceptions	1. The administrator has not filled all the mandatory fields of the form. 2. The data that the administrator has entered does not satisfy the conditions (invalid data).

Name	Add a company
Actor	Administrator
Entry conditions	The administrator is successfully logged in.
Events flow	1. The administrator opened the options menu. 2. He chose the option “Add a company”. 3. He added the new car sharing, bike sharing, taxi company, etc.. (which was recently open in Milan) in “Travlendar+” in order the system to consider them when calculating the routes.
Exit conditions	The new companies are successfully added, so “Travlendar+” is aware of their existing.
Exceptions	1. The administrator has not filled all the mandatory fields of the form. 2. The data that the administrator has entered does not satisfy the conditions (invalid data).

3.3.2. Sequence diagram

- **Create a meeting**

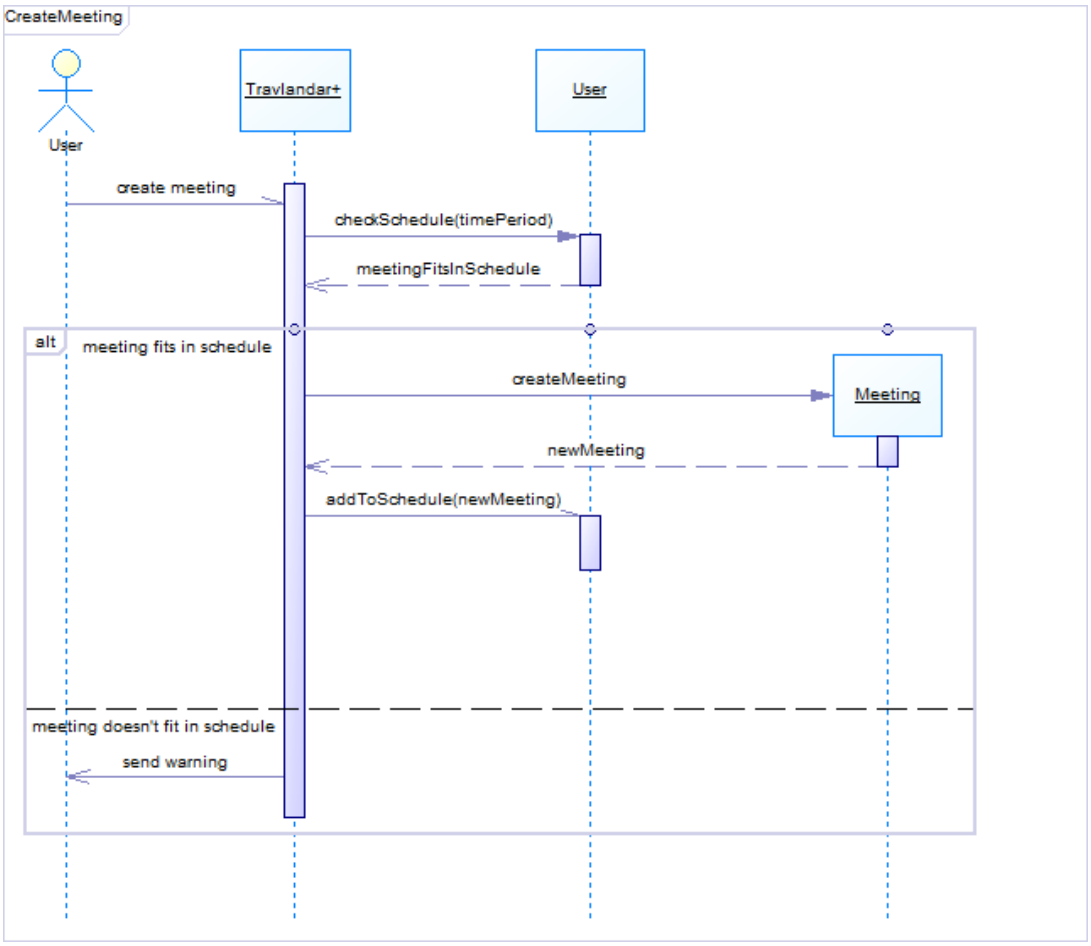

Figure 15 Sequence diagram - Add a meeting

- **Select a route**

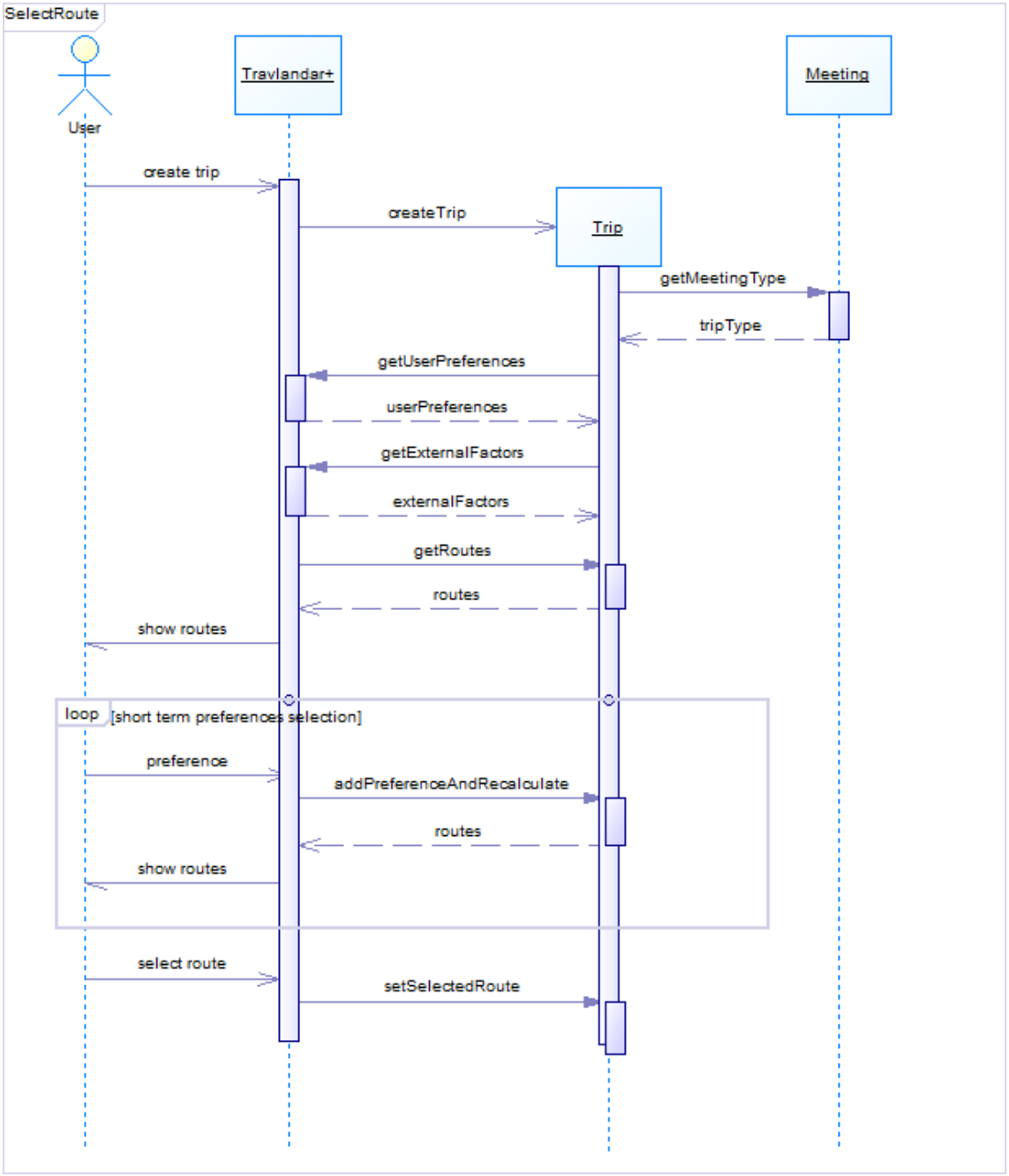

Figure 16 Sequence diagram – select a route

- **Create a break**

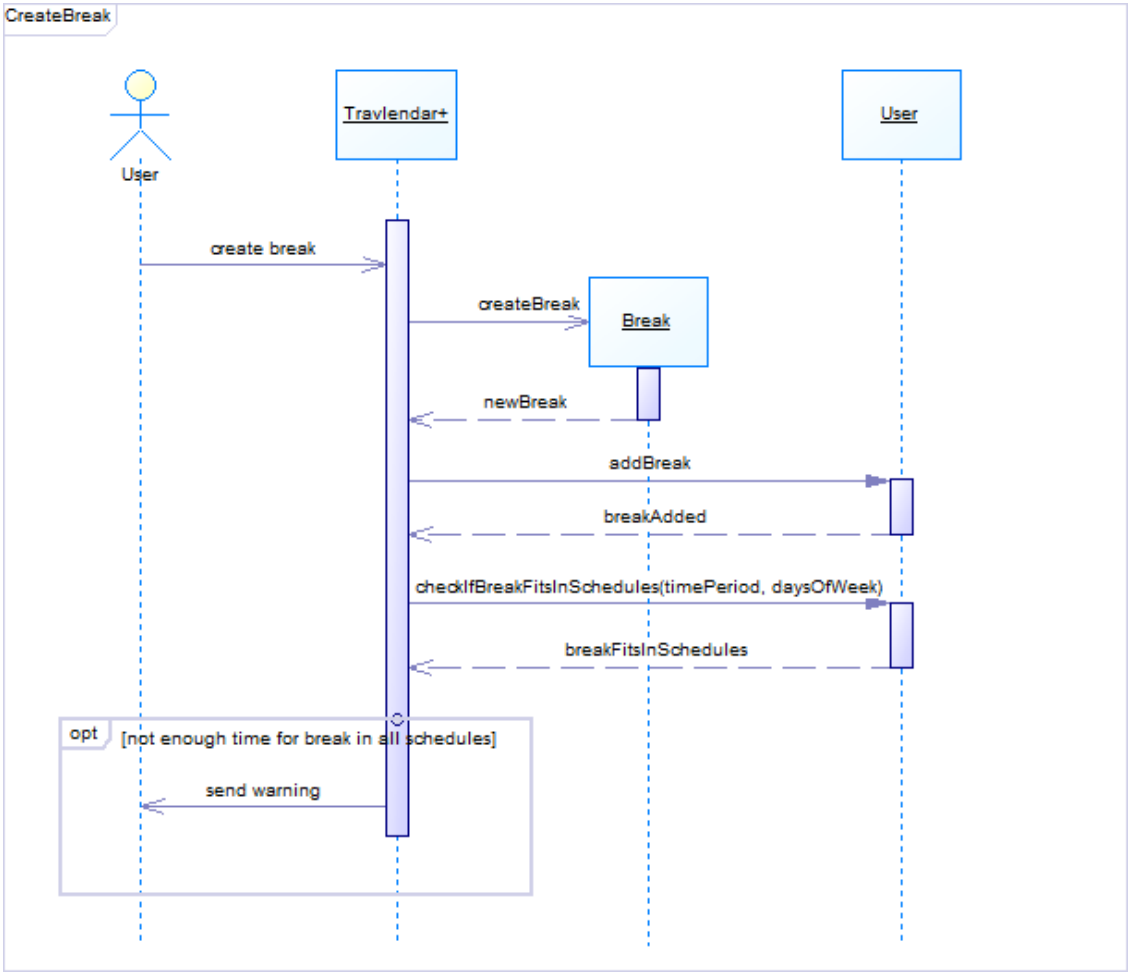

Figure 17 Sequence diagram - create a break

3.4.Performance Requirements

The system is provided to serve fairly great number of users simultaneously. At the beginning the idea is to be able to respond to around 50 000 of users, but it surely should be flexible to adapt depending on the requests and number of servers taken. Users will rely on the application in order to organize their everyday activities and obligations, so we have to guarantee quick, reactive and correct response.

3.5.Design Constraints

3.5.1. Standards compliance

- The app requests only the absolute minimum permissions that it needs to support core functionality.

- The app functions normally when installed on SD card (if supported by app).
- The app supports both landscape and portrait orientations (if possible). Orientations expose largely the same features and actions and preserve functional parity. Minor changes in content or views are acceptable.
- The app correctly preserves and restores user or app state. The app preserves user or app state when leaving the foreground and prevents accidental data loss due to back-navigation and other state changes. When returning to the foreground, the app must restore the preserved state and any significant stateful transaction that was pending.
- All private data is stored in the app's internal storage.

3.5.2. Hardware limitations

This is a software application with no strict hardware limitations. The only requirement is a smartphone that can use GPS services.

- iOS or Android smartphone
- 2G/3G/4G connection
- GPS

3.5.3. Other constraint

- Regulatory policies

If the user decides that they want to use the application also for payment purposes (such as ATM tickets, or car or bike services), it may ask for users' payment information. They will be used only for fees and rides payments.

Moreover, the application will have to ask for users' position in order to calculate the route. Telephone numbers and email addresses won't be used for commercial uses.

3.6. Software System Attributes

3.6.1. Reliability

The application must be available 24/7. Negligibly small concessions from this requirement might be tolerated.

3.6.2. Availability

In order to guarantee high degree of availability, system of redundant servers may be considered. This way, if possibly one server fails, the other one will be ready to take over. The system is expected to be available 99.99% of the time.

3.6.3. Security

User passwords and sensitive information like payment credentials should be confidentially stored and encrypted with high-security encryption. Security of the data and of the communications user-Travlendar+ and Travlendar+-APIs is of a highest importance.

3.6.4. Maintainability

As previously mentioned, the application is going to be flexible and easy to maintain, i.e capable to facilitate addition of new features and options. For that purpose we will use clear code following the design patterns.

Design patterns provide a standard terminology and are specific to particular scenario. For example, a singleton design pattern signifies use of single object so all developers familiar with single design pattern will make use of single object and they can tell each other that program is following a singleton pattern.

Of course, complete and detailed documentation will also be provided in order to keep the maintainability on the highest level.

3.6.5. Compatibility

This is an android application, so it needs to be compatible with as many devices as possible, while still implementing all of the requirements defined. It should be able to run on a wide variety of devices and circumstances.

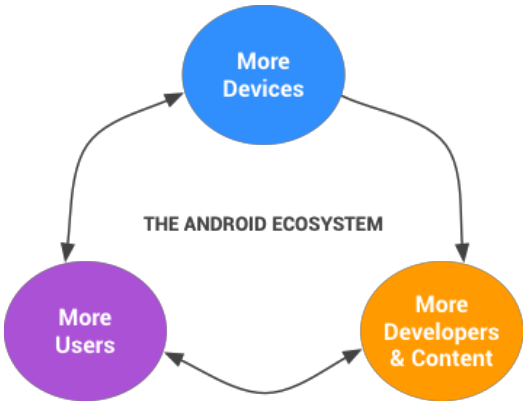

Figure 18 Device compatibility

The Android ecosystem thrives with device compatibility.

4. Formal analysis using Alloy

In this section, the Alloy model is given. Using it, some of the features of the system are specified and explained in more details, with the focus being on the constraints:

- There can be no overlaps of meetings of one user in the same day.
- The suggested routes in a single trip must obey the preferences specified by the user for that specific meeting type. If no preference is found, then the routes must satisfy the general preference (for all meeting types), if one is defined.
- The user must be shown a warning that it is not possible to reach a meeting if it is calculated that the time to reach the location of the meeting using the currently selected route will exceed the time of the meeting start.
- The user must be shown a warning that a break can't be completed in the given schedule if there is no gap big enough to satisfy the specified duration of the break considering meeting starts, ends and the time it takes to reach the next meeting location for meetings that fall into the specified period of the break.

It should be noted that in order to be able to check whether an instance satisfies the mentioned constraints, time is represented in minutes of the day (integers from 0 to 1440). However, in order for the model to be more simple and the execution time to be shorter but without the loss of any meaning, the minutes of the day are specified as integer values from 0 to 6.

4.1.Alloy model

```
open util/integer
open util/boolean

-- Represented as the number of minutes passed in a single day at the specified moment
sig Time {
        minutes: one Int
}
--{ minutes >= 0 and minutes <= 1440 }
{ minutes >= 0 and minutes <= 6 }

sig Date { }

sig Location {
        -- latitude
        coordX: one Int,
```

```
		-- longitude
		coordY: one Int
}
--{ coordX >= -90 and coordX <= 90 and coordY >= -180 and coordY <= 180 }
{ coordX >= -3 and coordX <= 3 and coordY >= -6 and coordY <= 6 }

sig Ticket { }

-- for an example, a bike rental station
sig TransportService {
		locations: lone Location
}

abstract sig Warning { }
one sig NotReachableInTime extends Warning { }
one sig BreakCantFit extends Warning { }

sig TransportMeans {
		publicTransport: one Bool,
		services: set TransportService,
		purchasedTicket: lone Ticket
}

sig MeetingType { }

sig Route {
		means: some TransportMeans,
		duration: one Int
}
--{ duration >= 0 }
{ duration >= 0 and duration <= 6 }

sig ShortTermPreference { }

sig LongTermPreference {
		meetingType : lone MeetingType,
		means : some TransportMeans,
		user: one User
}

abstract sig ExternalFactor { }
sig Strike extends ExternalFactor { }
sig Weather extends ExternalFactor { }
```

```
sig Trip {
	selectedRoute: lone Route,
	routes: set Route,
	meeting: one Meeting,
	factors: set ExternalFactor,
	preferences: set ShortTermPreference
}
{ selectedRoute = none or selectedRoute in routes }

sig BreakType { }

-- for ease, we are considering that breaks are specified for every day
sig Break {
	breakType: one BreakType,
	startTime: one Time,
	endTime: one Time,
	duration: one Int,
	user: one User
}
{
	(sub[endTime.minutes, startTime.minutes] >= duration) and
	(startTime.minutes < endTime.minutes) and
	(duration > 0)
}

sig Meeting {
	meetingType: one MeetingType,
	date: one Date,
	startTime: one Time,
	endTime: one Time,
	location: one Location,
	trip: one Trip,
	user: one User
}
{ startTime.minutes < endTime.minutes }

sig User {
	meetings: set Meeting,
	preferences : set LongTermPreference,
	warnings: set Warning,
	breaks: set Break
}
```

```
{
	-- no two preferences for one user can reference the same meetingType
	some p1, p2 : preferences | p1.meetingType != p2.meetingType
}


-- a meeting cannot exist on its own, or belong to more than one user, and that user must have it in their meetings
fact MeetingUserConnection {
	all u: User | all m: Meeting | (m in u.meetings implies m.user = u) and (m.user = u implies m in u.meetings)
}

fact MeetingTripConnection {
	all m: Meeting | all t: Trip | (m.trip = t implies t.meeting = m) and (t.meeting = m implies m.trip = t)
}

fact UserPreferenceConnection {
	all u: User | all p: LongTermPreference | (p in u.preferences implies p.user = u) and  (p.user = u implies p in u.preferences)
}

fact UserBreakConnection {
	all u: User | all b: Break | (b in u.breaks implies b.user = u) and (b.user = u implies b in u.breaks)
}

fact RoutesSatisfyPreferences {
	all m: Meeting |
	-- there is a preference for the meeting type
	(one p: LongTermPreference | p in m.user.preferences and p.meetingType = m.meetingType and m.trip.routes.means in
p.means ) or
	-- there is a general preference and no preference for a the meeting type
	((no p1: LongTermPreference | p1 in m.user.preferences and p1.meetingType = m.meetingType) and
	 (one p2: LongTermPreference | p2 in m.user.preferences and p2.meetingType = none and m.trip.routes.means in
p2.means)) or
	-- there are preferences but none are general or are concerning the meeting type
	((no p1: LongTermPreference | p1 in m.user.preferences and p1.meetingType = m.meetingType) and
	 (no p2: LongTermPreference | p2 in m.user.preferences and p2.meetingType = none))
}

-- no two meetings of the same user, on the same date that are overlapping (one starts earlier and doesn't end before the second one
starts)
fact NoOverlappingMeetings {
	no disj m1, m2: Meeting |  m1.date = m2.date and m1.user = m2.user and m1.startTime.minutes <= m2.startTime.minutes and
m1.endTime.minutes >= m2.startTime.minutes
}
```

```
-- there can't be a situation when a meeting is not reachable with the currently selected route and a warning is not visible to the user
fact WhenAreNotReachableWarningsSent {
	no disj m1, m2: Meeting | m1.date = m2.date and m1.user = m2.user and m1.startTime.minutes <= m2.startTime.minutes and
	add[m1.endTime.minutes, m2.trip.selectedRoute.duration] > m2.startTime.minutes and  NotReachableInTime not in
m2.user.warnings
}

-- when all meetings are reachable in time with the selected routes, send no warnings (update for break warnings)
fact WhenAreWarningsNotSent {
	all disj m1, m2: Meeting | (m1.date = m2.date and m1.user = m2.user and m1.startTime.minutes <= m2.startTime.minutes and
	add[m1.endTime.minutes, m2.trip.selectedRoute.duration] <= m2.startTime.minutes) implies NotReachableInTime not in
m2.user.warnings
}

fact WhenAreBreakCantFitWarningsSent {
	some disj m1, m2: Meeting | some b: Break | (m1.date = m2.date and m1.user = m2.user and m1.user = b.user and
	-- meeting1 is inside the break period
	((m1.startTime.minutes <= b.startTime.minutes and m1.endTime.minutes >= b.startTime.minutes) or
	 (m1.startTime.minutes >= b.startTime.minutes and m1.endTime.minutes <= b.endTime.minutes) or
	 (m1.startTime.minutes <= b.endTime.minutes and m1.endTime.minutes >= b.endTime.minutes)) and
	-- meeting2 is inside the break period too
	((m2.startTime.minutes <= b.startTime.minutes and m2.endTime.minutes >= b.startTime.minutes) or
	 (m2.startTime.minutes >= b.startTime.minutes and m2.endTime.minutes <= b.endTime.minutes) or
	 (m2.startTime.minutes <= b.endTime.minutes and m2.endTime.minutes >= b.endTime.minutes)) and
	-- one meeting starts and ends before the start of the other one
	(m1.endTime.minutes <= m2.startTime.minutes) and
	-- but, when you take into account the trip duration and the break, there is not enough time
	(add[m1.endTime.minutes, add[m2.trip.selectedRoute.duration, b.duration]] > m2.startTime.minutes))
	implies (BreakCantFit in m1.user.warnings)
	else (BreakCantFit not in m1.user.warnings)
}

pred createAMeeting[u: User, m: Meeting, d: Date, l: Location, t1, t2: Time, tr: Trip, mType: MeetingType] {
	m.date = d
	m.location = l
	m.startTime = t1
	m.endTime = t2
	m.meetingType = mType
	m.trip = tr
	tr.meeting = m
```

```
        m.user = u
        u.meetings = u.meetings + m
}

pred show {
        (some u: User | #u.meetings >= 2 and BreakCantFit in u.warnings) and
        (one m: Meeting | m.startTime.minutes = 0 and m.endTime.minutes = 2) and
        (one m: Meeting | m.startTime.minutes = 3 and m.endTime.minutes = 6) and
        (all t: Trip | t.selectedRoute.duration = 1) and
        (some b: Break | b.duration = 1)
}

assert MeetingStartsBeforeImpliesMeetingEndsBefore {
        all disj m1, m2: Meeting | (m1.user = m2.user and m1.date = m2.date and m1.startTime.minutes <= m2.startTime.minutes)
        implies (m1.endTime.minutes < m2.endTime.minutes)
}

assert MeetingAlwaysNotReachable {
        all disj m1, m2: Meeting | (m1.user = m2.user and m1.date = m2.date and
    m1.startTime.minutes = m2.endTime.minutes and m1.trip.selectedRoute != none)
        implies (NotReachableInTime in m1.user.warnings or m1.trip.selectedRoute.duration = 0)
}

run show for 4 but 1 User, 2 Meeting, 2 MeetingType, 1 Date, 4 Int, 1 Break, 1 ShortTermPreference, 1 Weather, 1 Ticket, 1
TransportService, 1 BreakType
run createAMeeting
check MeetingStartsBeforeImpliesMeetingEndsBefore
check MeetingAlwaysNotReachable
```

4.2.World generated

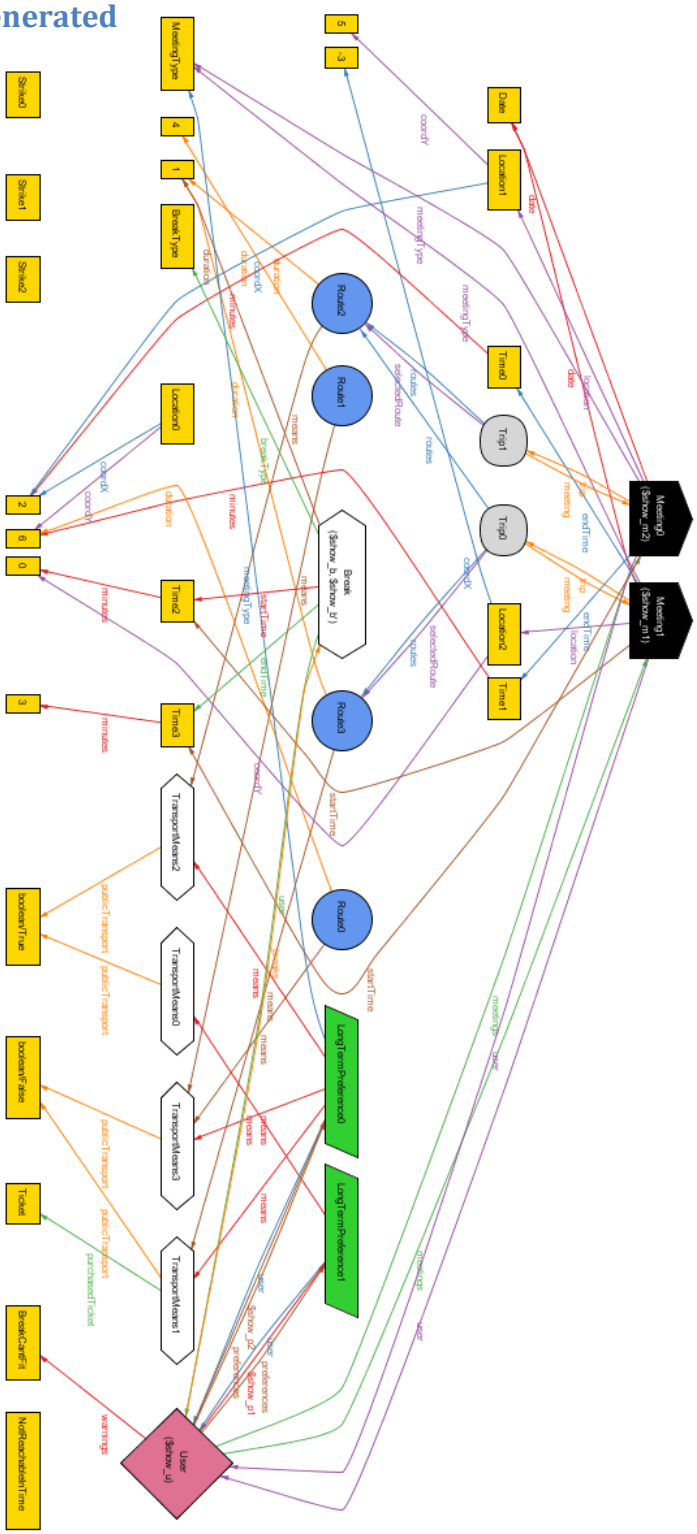

Figure 19 Alloy - World generated

4.3.Alloy results

These are the results produced by the code shown above.

```
Executing "Run show for 4 but 4 int, 1 User, 2 Meeting, 2 MeetingType, 1 Date, 1 Break, 1 ShortTermPreference, 1 Weather, 1 Ticket,
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   11018 vars. 513 primary vars. 31670 clauses. 93ms.
   Instance found. Predicate is consistent. 95ms.

Executing "Run createAMeeting"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13473 vars. 588 primary vars. 37979 clauses. 239ms.
   Instance found. Predicate is consistent. 112ms.

Executing "Check MeetingStartsBeforeImpliesMeetingEndsBefore"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   14058 vars. 570 primary vars. 40565 clauses. 82ms.
   No counterexample found. Assertion may be valid. 1823ms.

Executing "Check MeetingAlwaysNotReachable"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   13472 vars. 570 primary vars. 38148 clauses. 122ms.
   No counterexample found. Assertion may be valid. 563ms.
```

Figure 20 Alloy results

5. Effort spent

5.1

Description of the task	Hours
Purpose, scope, definitions	3
Product perspective	6
Product functions	2
Domain assumptions	4
External interface requirements	1
Functional requirements	6
Non-functional requirements	3
Formal analysis using alloy	8

5.2

Description of the task	Hours
Purpose, scope, definitions	3
Product perspective	3
Product functions	2
Domain assumptions	4
External interface requirements	8
Functional requirements	6
Non-functional requirements	4
Formal analysis using alloy	3

5.3

Description of the task	Hours
Purpose, scope, definitions	3
Product perspective	3
Product functions	2
Domain assumptions	4

Scenarios	4
Functional requirements	8
Non-functional requirements	5
Formal analysis using alloy	4

6. References

- Specification document “AA 2017---2018 Software Engineering 2—Mandatory Project goal, schedule, and rules”
- ISO/IEC/IEEE 29148 - Standard on requirement engineering
- Google API for android- https://developers.google.com/android/guides/api-client
- Alloy documentation - http://alloy.mit.edu/alloy/documentation.html
- Slides – “Usage of alloy in RE.pdf”