Travlendar+
Requirement Analysis and Specification Document
Version 1.1

Leonardo Bisica
Alessandro Castellani
Michele Cataldo

Contents

1 Introduction

1.1 Purpose

This document is the *Requirement Analysis and Specification Document* (from now on RASD) for the information system *Travelandar+*. This application aims to help users in their everyday life by organizing their appointments and optimizing their travel means. No previous versions of this application have been developed. We'll start by describing Stakeholders' aims (Goals), from which we obtain, subsequently, functional and nonfunctional requirements (*Requirements* Section) useful to describe the system. In addition we will need two other sections to have a complete overview of the model: *Constraints* Section is going to describe constraints about the system, while *Domain Properties* Section is going to underline the limits forced by the world on our software. We'll analyze this first chapter to subsequently delve into scenarios and use cases for the application.

1.1.1 Goals

1. *[G1]* System allows guest user to register with an username and a password; to complete the procedure user should confirm by e-mail.
2. *[G2]* System Login.
3. *[G3]* The application integrates a calendar and a timetable.
4. *[G4]* Registered User can create appointments.
5. *[G5]* Registered Users can edit appointments.
6. *[G6]* The application can automatically compute a personalized selection of travel times between appointments to choose from.
7. *[G7]* User can choose a solution among the scheduled ones.
8. *[G8]* The application warns the user if locations are unreachable in the allotted time.
9. *[G9]* Allow users to put constraints on different travel means and limit carbon footprints.
10. *[G10]* The application features additional user's privileged time spans.
11. *[G11]* The application allows to buy tickets for public services.
12. *[G12]* The application allows the nearest shared vehicle to be found and reserved.
13. *[G13]* The application integrates a map system.
14. *[G14]* The User can submit additional preferences

1.2 Scope

The aim of this project is to develop *Travlendar+*, a calendar-based mobile application. The main functionality of the system is helping people in scheduling their appointments by taking into account useful, external information regarding traffic, public transportation, weather, and the like. Appointments could be scheduled through the entire region of Lombardy (Italy), and the main Italian cities connected via railway system. There could be several types of work meetings and personal appointments. Furthermore the system can help the user by allowing the purchase of public transportation tickets via the mobile application itself, or by reserving a car or a bike of a sharing system (whenever this is possible). In case of bad weather the system should find alternative moving solutions in order to replace walking paths, same goes with strikes and other relevant kinds of information. Naturally, the system will also allow the registration of new users; as for the registration, the system requests both personal and payment information. After the registration succeeds, the user could immediately start scheduling his meetings.

1.3 Glossary

User We will refer to all people who are registered to the system as 'Users'. All users have personal profiles which contain the following information:

- First name;
- Last name;
- Email;
- Username;
- Password;
- Payment information; this in particular includes:
  - Credit card owner;
  - Credit card number;
  - Credit card expiration date;
  - CVV number.

Guest We name 'guests' all the people who are using the interface of the system without being registered or logged in. Guests can't access any functionality of *Travlendar+* except for the registration process and the log in.

Operative Zone We name Operative Zone the area within we can place the location of an event. For the time being the Operative Zone coincide with all the cities and places within italian peninsula that can be reached by simply consulting the Google APIs. Naturally, such an area may be expanded in the future.

Influence Zone We name Influence Zone the area within whose borders the mobile application can not only give travel time by car and on foot (the minimum standard given to us by Google APIs) but also where *Travlendar+* can rely at least on a single car and bike sharing service. For starting, the Influence Zone will coincide with the city of Milan.

Registered User A registered User is a former guest who inserted his/her own credentials in the system. After previous login, a Registered User can then create events, work on the timetable and ultimately is the end-user of *Travlendar+*.

Calendar Calendar reflects the intuitive English meaning of the word. We mean by Calendar a month by month view within our mobile application and its implementation that has to be easy to export and modify.

Timetable Calendar reflects the intuitive English meaning of the word. We mean by Timetable a day by day view within our mobile application and its implementation that has to be easy to export and modify.

Vehicle Sharing services/systems and a Shared Vehicle By vehicle sharing we do not inted referring to a generic 'car-pooling' service. The usage of a shared vehicle may be either one of two types, car or bike sharing. A vehicle within the system operates only within the boundaries and parking zones imposed by its service; it can be picked up by any user registered to its corresponding system, used for the required amount of time (even though a maximum time is always fixed) and then parked in an allowed zone, ready to be picked by up again by another user. Each sharing system possesses an individual API.

Break Time/Break By break we refer to a privileged time span : it is a time span in which events can't be scheduled. It must have a minimum duration specified by the user and can be encapsulated in a bigger time frame.

Mobile Application By mobile application we refer to a program conceived for Android and iOS operative systems, based on touch interfaces and able to run on portable devices. The logic of the mobile application is the system. We often abbreviate 'mobile application' into the more colloquial 'app'.

Appointment/Event An appointment is an event well delimited both in time and space, requiring the presence and the direct investment, in our case, of the user who creates it. Appointments fall in two categories : work appointments (that are often referred also as meetings) and personal appointments (the broader set encapsulating all other kinds of appointments, mainly regarding personal and family life). It is also possible to encounter the word 'Event' meaning the same exact thing.

Warning A warning is a notification given by *Travlendar+* mobile application to the operating System it is hosted by. It behaves as a standard system notification.

Travel Logic By travel logic we refer to the logic that processes the distances and the transportation time within our operative and influcence zones. In the case at hand, in this first implementation, we're going to adopt as Travel Logic the Google Maps APIs.

***Travlendar+* Server** *Travlendar* server is the simplified terminology we adopt to define the remote storage system indexed by user credentials. In such a server we can store users' perfonal preferences and timetables

Travel means We name travel means all possible travel solutions considered by our system : car, byke, public transportation, walking. All kinds of transporation on air and on sea are excluded.

1.4 Revision History

No revision where done since first edition of this document.

1.5 Reference Document

- Specification Document: Mandatory Project Assignments.pdf
- IEEE International Standard ISO/IEC/IEEE 29148 2011-12-01

1.6 Document Structure

This document is composed by 5 chapter:

Introduction: this section gives a brief explanation of the system to be. It helps the reader with regards to the terminology and glossary and it gives him information about the documents that were used as references.

Overall Description: in this section it will be provided a solid background for the requirements so that they will became easier to understand. The reader will see also some pieces of hardware and software used to create the model.

Specific Requirements: this section provides more details on the aspects of the prevoius section. In particular is presented the difference between functional and non-functional requirementes in addition to their perferomance and the explanation of external interface requirements.

Formal Analysis using Alloy: in this chapter the reader will see Alloy model of the system and some outputs obtained by running the model.

Effort Spent: is a section dedicated to shows the number of hours of each authors spent for create the model.

Reference: addiotional material usuful for the completness of the RASD.

2 Overall Description

2.1 Product perspective

Travlendar+ is a mobile application that's going to be built from scratch and heavily reliant on external APIs. We'll start with open, official immediately accessible APIs : one that contitues the core of Travel Logic, and one as a reference for the implementation of car-sharing rental. *Travlendar+* has to implement a modular approach, so that later on it will be able to implement and expand upon iterative additions of external services as its operative zone grows and new partnerships arise. In its firts version, *Travlendar+* will make use of the following APIs:

Google Maps to track actual distances and expected times for travels on foot, by car and with public transportation (that shall include trains and inter-city travels too) Google Maps.

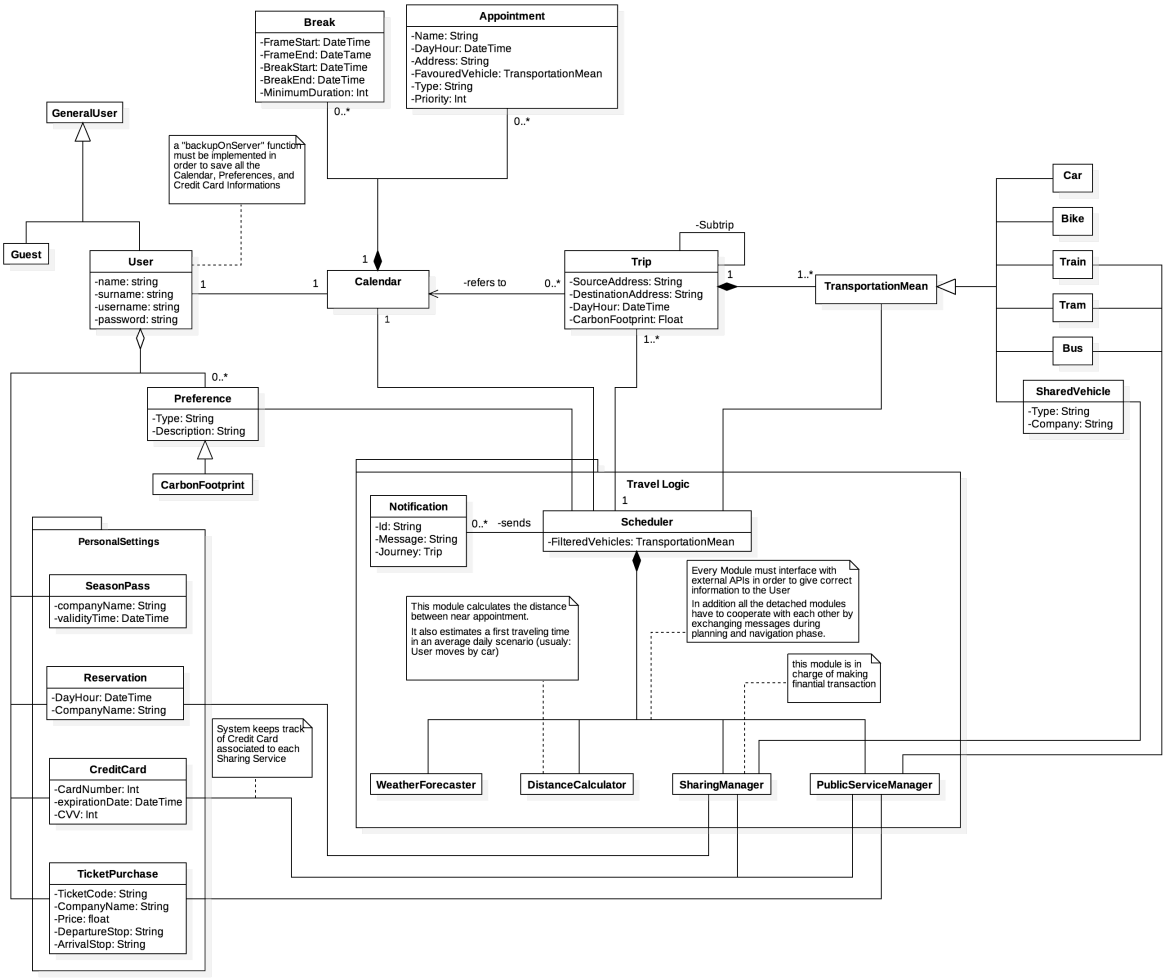
Car2Go to recover information about availability, parkings and to rent a ride with one of the biggest provider of car-sharing services in the world, Car2Go. Car2Go.

Mobike to recover information about availability and to rent a bike through Mobike service by using the official APIs, following the instructions detailed here. Mobike.

OpenWeatherMap to obtain information about weather in order to properly schedule trips. OpenWeatherMap.

2.1.1 Class Diagram

We now include a sketch of the structure of the mobile application system:



2.2 Product functions

We've already detailed the main goals of our mobile application in the first section of the current document; naturally, the aforementioned goals will be functions of *Travlendar+*. Here though we submit an higher-level description of major functions to ease the understanding of our program to all interested parties.

Appointments Manager Our mobile application will let any registered user to view, create, modify and delete events, denoted by their type, their date and location.

Travel Scheduling When a new appointment is inserted in the application timetable an immediate check is performed whether the location of the event can be reached in the allotted time on foot, by car, or by public transportation (referencing, in lack of additional information, to their default behaviour according to the date and time of the day). Day by day, events and allotted times are periodically checked with

live informations to ensure they're feasible: a list of results is prompted to the user whenever he asks for it and he can express preferences and filter them.

Payment Manager for Public Transportation Our mobile application will redirect registered users to payment portals through secure channels and it'll be able to verify whether transaction succeeded or not by interfacing with external payment services. Unfulfilled payment will result in an impossibility to go forward and will prompt the possibility to submit the data and restart the transaction.

Car and Bike Sharing Integration The mobile app will be able to integrate external info about the availability and location of shared cars and bikes so that it will also allow their reservation. Everything past this mark clearly outgrows the scope of our system and it shall be handled by the company that grants the service (same goes for eventual malfunctions or erratic behaviours). Travlendar+ will send send reservation requests to the selected company and will be able to record a rental acceptance notification (i.e., everything's gone right in the provider's rental system).

Free Time spans Our system will grant the possibility to reserve free time spans: such breaks won't be affected by scheduling and by the need to travel and registered users will be able to insert theme directly in the calendar.

2.3 User characteristics

We list the actors involved in our system:

Guest A guest is a potential user, an unregistered or not-yet-logged person who opens the app. He can't access all its functionalities and, until it's logged in, his only choice is logging.

Registered User/User The user is the final and only customer of *Travlendar+*. He has identifying credentials, can personalize his timetable by setting up and configuring events, can specify favourite transportation means, buy tickets, view expected arrival times, reserve shared vehicles and get notified by the system.

In addition to that, *Travlendar+* also interacts with different service providers:

Payment Service Providers that cover transactions prompted by the mobile application.

Localization Service The provider that manages localization, maps and standard travel times.

Sharing Service Service providers of car and bike sharing services that manage position and rental of vehicles.

Public Transportation Service Public Transportation information systems that show and sell tickets through *Travlendar+*.

2.4 Assumptions, dependencies and constraints

We've already given a formal and methodical definition of our problem, yet there are still some ambiguities which still need to be addresed.

1. Users do not create events outside the "operative zone".
2. The devices *Travlendar+* is installed on possess a well-functioning GPS for geo-localization.
3. If a registered User is willing to use the vehicle of a sharing network, we assume him to have downloaded the corresponding sharing-network app.
4. There's no kind of dependency among the users of our system.
5. Any information coming from sharing services regarding position, rental, and payment won't be double-checked by our system.
6. Information coming from external payment sites won't be double checked by our system.
7. Buying public transportation tickets rely on the aforementioned external payment procedures.
8. System assumes the season passes submitted by the user only aid for filtering results and are not checked nor have any legal value.
9. System assumes that any car rental request is made by a person who's allowed to make one. Only sharing services check the validity of documents; in addition to that we always assume the driver is the user who requests the rental.
10. The application doesn't act as a navigator, and isn't capable of giving live informations about the travel besides the ones that can arrive through notifications.
11. Personal vehicles consider as their starting position the one of the geo-localized device.
12. Modified and deleted events cannot be restored in any way.
13. In order to use a vehicle sharing service its corresponding mobile application must be installed on the user device.

3 Specific Requirements

3.1 External Interface Requirements

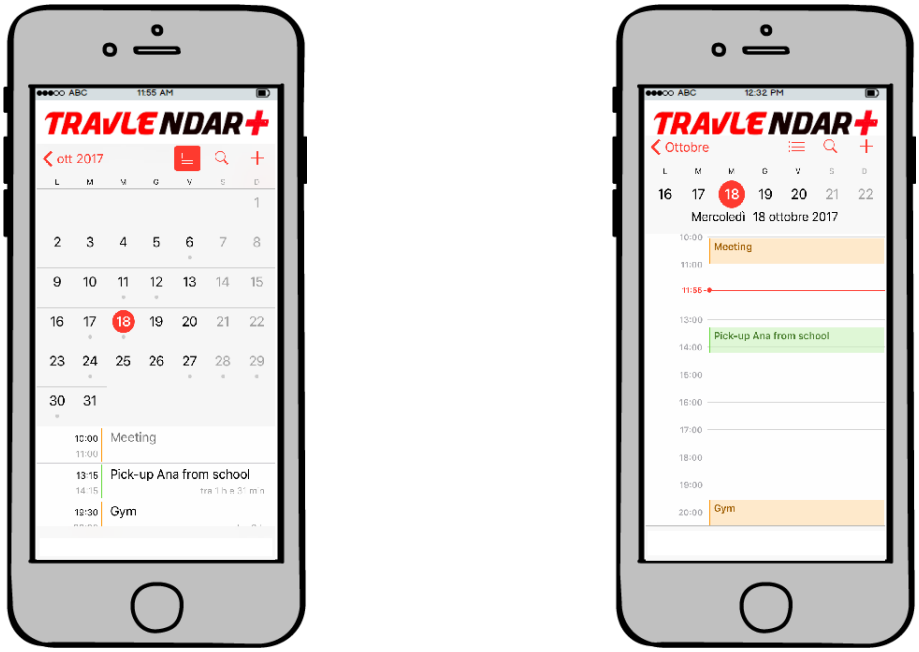
3.1.1 User Interfaces

The interface of our System is to be used via a mobile app because all the functionality make sense only in a *movable* context (meaning that users can exploit them anywhere they have an internet connection).

We will now list some of the user interfaces thought for *Travlendar+*:

Figure 1: Login Page.

As mentioned before, a *Guest* or a *non logged User* will first encounter the page showed in figure 1, and he won't be able to access the app until he completes the Registration or the Login procedures (see sections 3.2.1.1 and 3.2.1.2).



(a) Home Page in Monthly view.

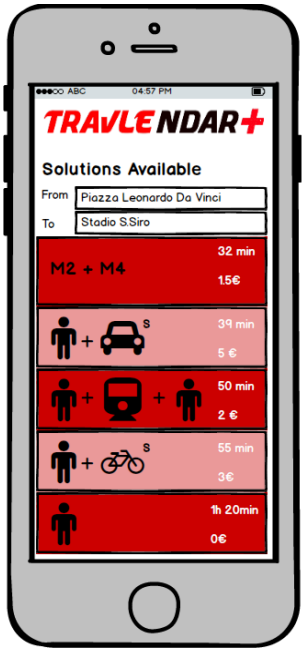
(b) Home Page in Daily view.

Figure 2: Samples of view events.

The user can use two different views to manage events in his homepage. On the left (figure 2a) the application shows monthly view, on the right (figure 2b) the daily view. Thus user will modify the scheduling in a simple way by clicking on the events.

Figure 3: Create an Appointment page.

When the user wants to creates an event, he has to use this interface of the application. In fact figure 3 shows what preferences the user can choose regarding the event.



(a) Solutions found for an Event view.

(b) Buy a public transportation ticket view.

(c) Reserve a Sharing service resource view.

Figure 4: Samples of the Travel logic module.

The application computes the possible solutions to reach a certain event based on the starting position. These are displayed on the screen of the user's smartphone as shown in figure 4a. The user must choose the preferred solution to see the details. It may happen that the solution contains 'buy tickets' (figure 4b) or 'hire cars' (figure 4c). Clicking on the corresponding buttons will open the corresponding externarl application.

3.1.2 Hardware Interfaces

The system allows to interact with all the possible processors that are in the devices of the two companies Android and Apple. Among the most important processors there are: Qualcomm, ARM and Exynos.

3.1.3 Software Interfaces

1. Android
   - Name: Lollipop
   - Version: 5.0+
   - Source: https://www.android.com/intl/it_it/versions/lollipop-5-0/
2. Apple
   - Name: iOS
   - Version: 8.0+
   - Source: https://www.apple.com
3. Google Maps APIs
   - Name: Google Maps APIs
   - Source: https://developers.google.com/maps/

3.1.4 Communication Interfaces

The system also interface with the applications for the use of sharing and payment services

Protocol	Application	Port
TCP	HTTP	443
TCP	HTTP	80

3.2 Functional Requirements

We now adopt a goal-based approach to determine the requirements associated with each one of the goals we have elaborated in Chapter 1.

We'll start numbering and exploring the goals we submitted.

- *[G1]* System allows guest user to register with an username and a password; to complete the procedure user should confirm by e-mail.

R.1.1 System lets registering user choose an username and password.

R.1.2 Every username corresponds to a single user.

R.1.3 Duplicate usernames aren't allowed.

R.1.4 Registering user can't be already registered.

R.1.5 An unregistered user is locked out the application and can only see registration page.

R.1.6 User has to confirm by mail his registration.

R.1.8 New user registration is successful only after data is stored on Travlendar+ Server and a confirmation is received by the system.

R.1.10 Each modification made to a user account must be saved into Travlendar+ Server to be made effective.

- *[G2]* System Login.

R.2.1 User must be already registered to perform correct login.

R.2.2 User must remember username and password to login.

R.2.3 Only a correct combination of username and password will grant access.

R.2.4 Application will implement a password retrieval mechanism.

- *[G3]* The application integrates a calendar and a timetable.

R.3.1 The calendar and the timetable must have two different interfaces.

R.3.2 Calendar must give to the user granularity regarding both months and days.

R.3.3 Calendar and Timetable can be modified only by the user inserting events. No one else is allowed to either see or modify the information they contain.

R.3.4 Calendar and Timetable for each user are remotely copied on Travlendar+ Server every time a user creates/modifies/deletes an event.

- *[G4]* Registered User can create appointments.

R.4.1 User has to be registered and logged in the system in order to create an appointment.

R.4.2 Appointments can be divided into work appointments (or meetings) and personal appointments

R.4.3 Appointments require a location, a starting time and an end time

R.4.4 Appointments location must be within the boundaries of the operative zone

R.4.5 There cannot be appointments with the same name, location and time

R.4.6 System must check suitability of created new entries based on already existing appointments

R.4.7 Appointment start time can't precede the actual system time at the moment of inserting it

R.4.8 User can select favourite travel means and priority for each appointment

R.4.9 Each appointment must be associated to a level priority

R.4.10 The creation of an appointment must be remotely saved on Travendlar+ server in order to be successful and complete

- *[G5]* Registered Users can edit appointments.

R.5.1 A modified meeting must respect all the constraints imposed during the creation of a new meeting, as the requirements in *[G4]*.

R.5.2 A meeting can be modified up until its end time.

R.5.3 If the meeting is modified, the system behaves as if such an event was inserted for the first time, calculating all possibile conflicts with pre-existing events.

R.5.4 No limit actually exists on the amount of times an event can be modified within the aforementioned constraints.

R.5.5 A modification must be correctly saved on the remote *Travlendar+* server in order to be succesful and completed.

R.5.6 Deleting an appointments must belong to the set of modifications.

- *[G6]* The application can automatically compute a personalized selection of travel times between appointments to choose from.

R.6.1 The application must refer to Travel Logic for the expected travel time.

R.6.2 The application must be able to suggest a combination of various means to reach the desired destination.

R.6.3 In case the trip expects more than one travel mean, the journey must be divided into sub-problems whose expected travel time has to be calculated. Same goes with public means stop and shared vehicles.

R.6.4 Starting location for travel can be inserted manually, retrieved by the previous event or calculated through geo-localization.

R.6.5 The application must rank the suggestions according to their priority, presence of prefered travel means and time required.

R.6.6 The registered user must be able to choose to filter out specific travel means.

R.6.7 Favourite travel means associated to an appointment must always show up.

R.6.8 In case two or more appointments overlap, an appointment with higher priority is considered automatically chosen and all the remaining ones are arranged according to their priority. Warnings must follow as expected.

R.6.9 The route can include intermediate destinations before the final, target one.

R.6.10 When a shared vehicle is suggested the parking zone nearest to the destination must be always inserted among the intermediate destinations.

R.6.11 The sytem must grant to know daily scheduled times for public transportation through its APIs.

R.6.12 When the starting time of a trip associated to an event is only one hour away the system must notify the user with an updated list of travel time so he can choose.

R.6.13 According to real world data, each travel must have associated to itself the carbon footprints.

R.6.14 Travels that do not satisfy all User's contraints must be excluded.

- *[G7]* User can choose a solution among the scheduled ones.

R.7.1 Selecting a solution that is not a personal vehicle must show both intermediate and final destinations.

R.7.2 The application must arrange a navigable interface of feasible solutions.

R.7.3 Choosing a solution that includes a public transportation mean must show the user the possibility to buy a ticket.

R.7.3 Choosing a solution that includes a shared vehicle must show the user the possibility to locate and rent such a vehicle.

R.7.4 Choosing a solution must not be definitive.

R.7.5 System must recognize by itself through geolocalization that a user reached destination; also, User must always be able to stop the trip.

- *[G8]* The application warns the user if locations are unreachable in the allotted time.

R.8.1 The application must realize if the alloted time is sufficient from either the last event, current location or manually inserted location.

R.8.2 The application must use as a reference the time to cover distance between the starting place and the destination one, using the futured scheduled time for public transportation if necessary.

R.8.3 Warning must arrive also while on the road if the travel mean is no longer suitable, or the best solution: in that case the system is going to prompt a new eventual choice of travel means.

R.8.4 When user reaches destination warnings must stop automatically.

R.8.5 Warnings can be disabled on the road by the user.

- *[G9]* Allow users to put constraints on different travel means and limit carbon footprints.

R.9.1 User must be able to rule out vehicles from search result returned by the system scheduler.

R.9.2 When the option of limiting carbon footprints gets enabled the associated CO2 consumed by each travel must be taken into account in travels scheduling.

R.9.3 User must be able to put a constraint on the number of travel means adopted for a single travel.

R.9.4 User must allow at least a single travel mean.

R.9.5 User cannot remove "walking" from travel mean preferences.

- *[G10]* The application features additional User's breaks.

R.10.1 Each Break is characterized by a duration, the time of the day they start in and by the time frame within are allowed.

R.10.2 Breaks can be periodic.

R.10.2 System reserves a minimum quantity of time which is not shorter than the break duration.

R.10.4 Breaks must be completely encapsulated within the time frames the break is allowed in .

- *[G11]* The application allows to buy tickets for public services.

R.11.1 Buying a ticket must reroute the user to the corresponding payment service.

R.11.2 User must be able to choose among the different purchase options offered by the public transportation service provider.

- *[G12]* The application allows the nearest shared vehicle to be found and reserved.

R.12.1 A shared vehicle must necessarily belong to a bike-sharing service or a car-sharing service.

R.12.2 All services linked to shared vehicles must be automatically disabled if the location of an appointment out of the boundaries of the influence zone.

R.12.3 All sharing services have their own API which is used by the system to locate and reserve the vehicles.

R.12.4 To find or reserve a vehicle it's required that the user logins into the external corresponding service.

R.12.5 The external service can communicate with our mobile application. In case of reservation *Travlendar+* checks if the mobile app corresponding to the desired services is installed on the system. All the following steps take place within such an environment, until control is returned to *Travlendar+*.

R.12.6 The location of all the vehicles must be shown in the same interface, merging data from different APIs.

R.12.7 Only shared vehicles that are free and available must be displayed and possibly reserved.

- *[G13]* The application integrates a map system.

R.13.1 The map must be submitted by an external service.

R.13.2 User must be able navigate through the map indipendently from its current location.

R.13.3 User must be able search for a specific location.

R.13.4 The mobile device must be able to track its current position through geo-localization.

R.13.5 Positions of out the operative zone can't be accepted by the system and won't be displayed.

- *[G14]* The User can submit additional preferences

R.14.1 User must be able to forbid travel means within time spans, also periodical ones.

R.14.2 User must be able to put a constraint on the maximum amount of space and time he can give to each travel mean.

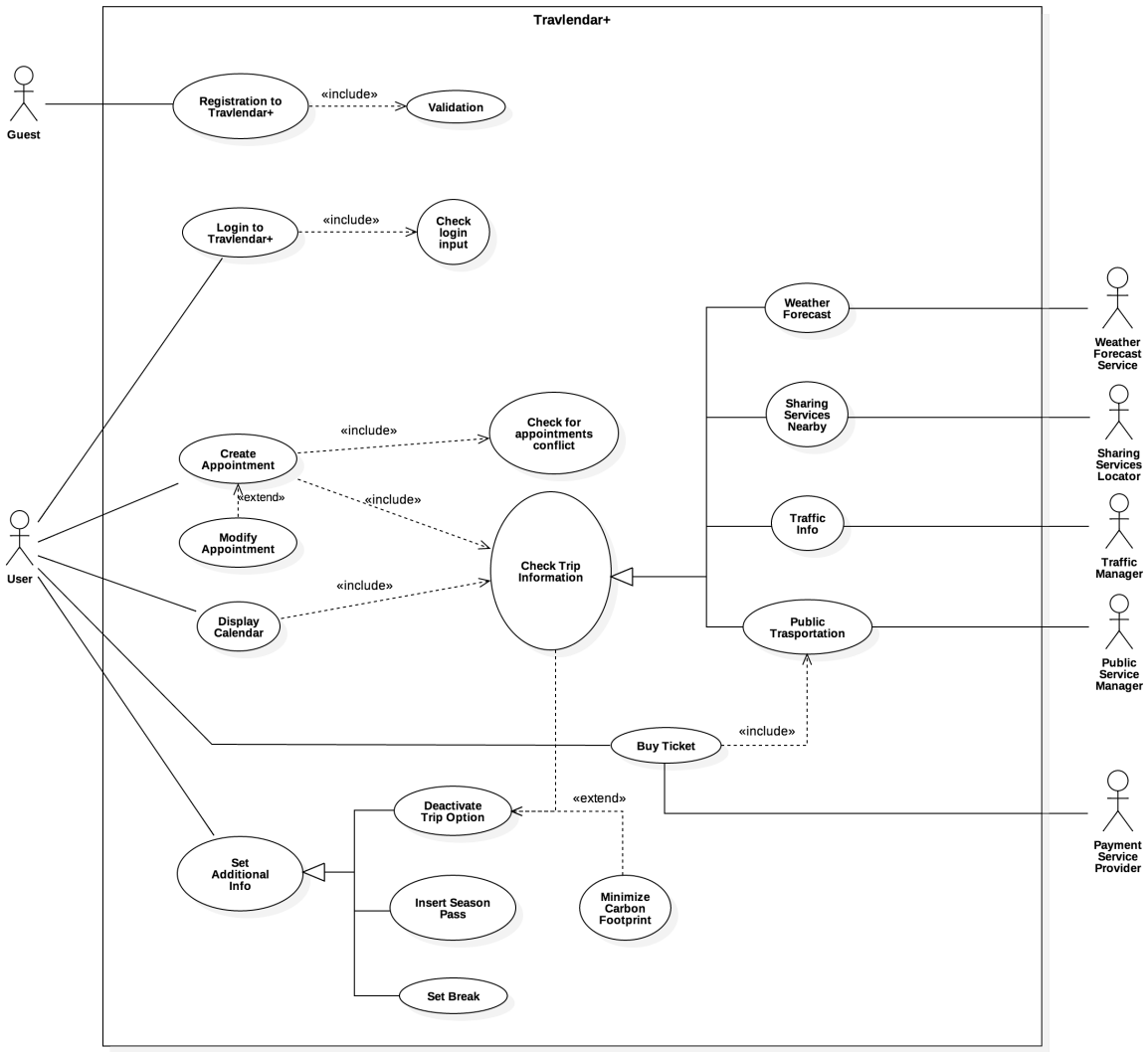
R.14.2 User must be able to link one or more season passes with his account.

R.14.3 User must be able to link one or more credit cards to his account.

R.14.4 Each modification apported by the User to its additional preferences is only made effective when synced on *Travlendar+* Server.

3.2.1 Use Case Diagrams

A global picture of the system interaction with actors is provided here by means of use case diagrams. Following, an analysis of the most interesting use case situations derived from scenarios is presented.



3.2.1.1 Guest registers to *Travlendar+*

Actor	Guest.
Input Condition	NULL.
Event Flow	1. Guest clicks on "Sign Up" button. 2. Guest fills in at least all mandatory fields. 3. Guest reads and accepts privacy policies and agreements from the company. 4. Guest clicks on "Confirm" button. 5. System sends Guest a confirmation link to the provided e-mail. 6. Guest clicks on the confirmation link. 7. System saves the data in the DB.
Output Condition	Guest succesfully ends registration process and become a User. From now on he/she can log in to the application using his/her credential and start using *Tralvendar+*.
Exception	- Guest is already a User. - One or more mandatory fields are not valid. - Choosen username is already in use. - Email choosen is already associated to another user. All exception are handle alerting the visitor of the problem and application goes back to point 2 of Event Flow

3.2.1.2 Guest logins into *Travlendar+*

Actor	Guest, User.
Input Condition	Guest is registered to *Travlendar+*.
Event Flow	1. Guest fills login mandatory fields. 2. Guest clicks on "Log In" button. 3. System verifies login credentials.
Output Condition	Guest is promoted to User and is shown is Calendar home page.
Exception	Login credentials are incorrect and Guest is shows again Login page

3.2.1.3 Create a New Appointment

Actor	User.
Input Condition	User is already logged in into *Travlendar+*.
Event Flow	1. User clicks on "Create Appointment". 2. User sets name, day, time and position of the Appointment. 3. System checks if the new appointment overlaps with already existing appointments or break period. 4. System calculates, ranks and shows multiple solutions depending on user travelling preferences. 5. User selects one of the proposed solutions as preferend one.
Output Condition	*Tralvendar+* shows calendar main page with the new appointment.
Exception	- Created appointment overlaps with already existing appointments. - There are no feasible solution. - Inserted location isn't in the *Operative Zone*.

3.2.1.4 Modify appointment

Actor	User.
Input Condition	- User is already logged in into *Travlendar+*. - Appointment already exists.
Event Flow	1. User clicks on "Appointment". 2. User starts modifying process. 3. System checks if the new appointment overlap with already existing appointments or break period. 4. System calculates, ranks and shows multiple solution depending on user travelling preferences. 5. User selects one of the proposed solutions as preferend one.
Output Condition	*Tralvendar+* shows calendar main page, with the updated appointment.
Exception	- Modified appointment overlaps with already existing appointments. - There are no more feasible solutions. - Modified location is no more in the *Operative Zone*.

3.2.1.5 Insert Payment Method

Actor	User.
Input Condition	- User is already logged in into *Travlendar+*. - Credit Card isn't already inserted on the system.
Event Flow	1. User clicks on "Preferences/Payment Methods". 2. User sets all the credit cards info. 3. System checks and validate provided informations.
Output Condition	*Tralvendar+* returns to "Payment Methods" page showing added card as a valid payment method.
Exception	Credit card given informations are invalid.

3.2.1.6 Buy Public Transportation Ticket

Actor	User.
Input Condition	- User is already in "Solutions" page. - A payment method is already available.
Event Flow	1. User clicks on "desired solution". 2. User clicks on "Buy Ticket" button. 3. System shows available public trasportation tickets. 4. User selects a ticket. 5. System starts purchase transaction.
Output Condition	Based on public transportation service, User receives a valid ticket.
Exception	*External Transaction Service* doesn't worked as expected.

3.2.1.7 Reserve a *Sharing Service* resource

Actor	User.
Input Condition	- User is already in "Solutions" page. - A "Sharing mean" is already available. - A "payment method" is already available. - User is in the *influence zone*
Event Flow	1. System connects to available "Sharing Service" resources. 2. System ranks per distance all the freasible recources and shows them on a map centered on User position. 3. User chooses one on the possible solutions. 4. User clicks on "Reserve it" button. 5. System reconnects to selected *Sharing Service*, and starts the reserving procedure.
Output Condition	User redirected to the *Reservation Service* app.
Exception	*Reservation Service* doesn't worked as expected.

3.2.1.8 Set Trip Preferences

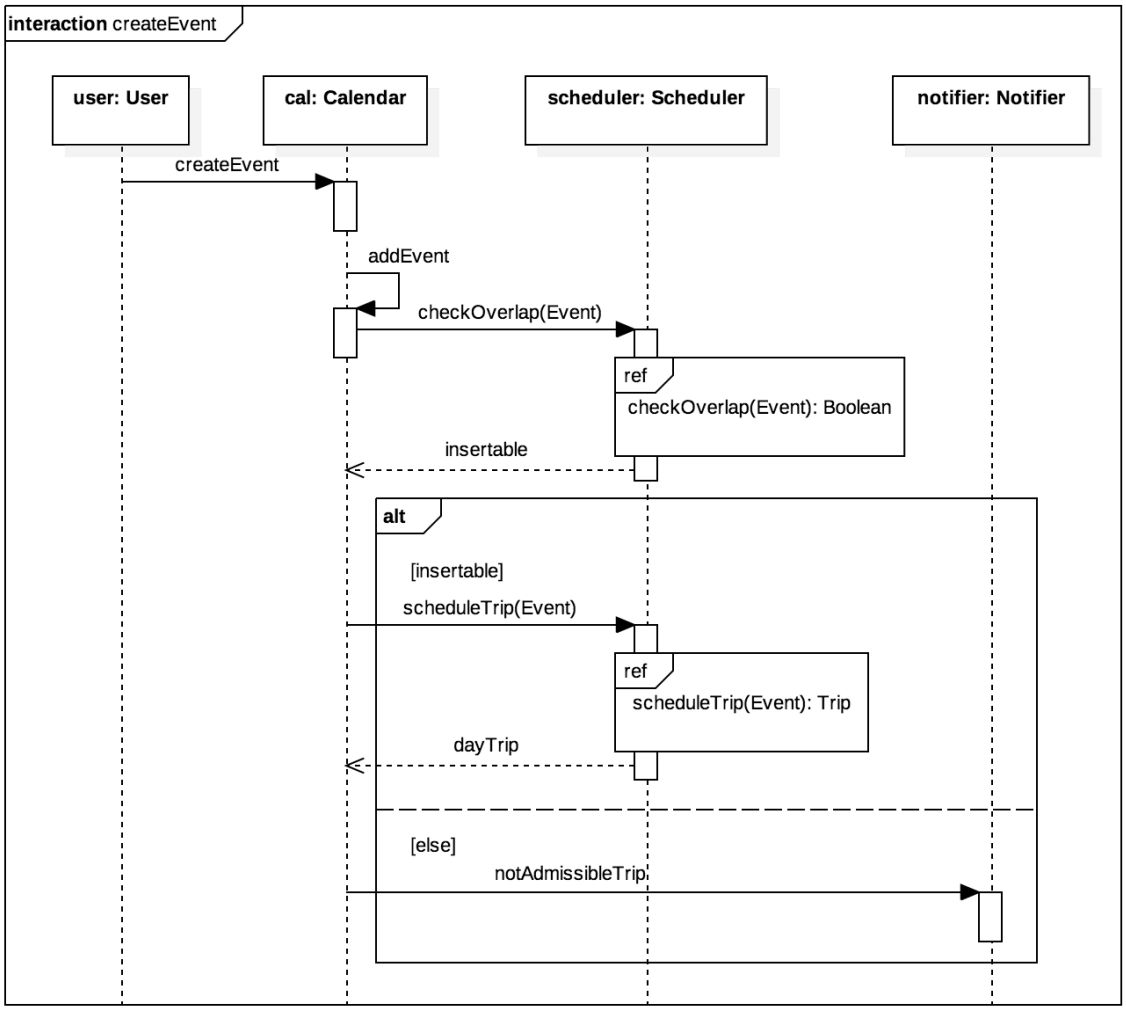
Actor	User.
Input Condition	User is already logged in into *Travlendar+*.
Event Flow	1. User click on "Preferences/Trip". 2. System shows all the possible preferences. 3. User pins prefered options.
Output Condition	- User returns to Caledar page. - System recalculate all future trip solution according to User preferences. - User is informed of particular problem
Exception	User unpins all the possible travel means.

3.2.1.9 Set Break Period

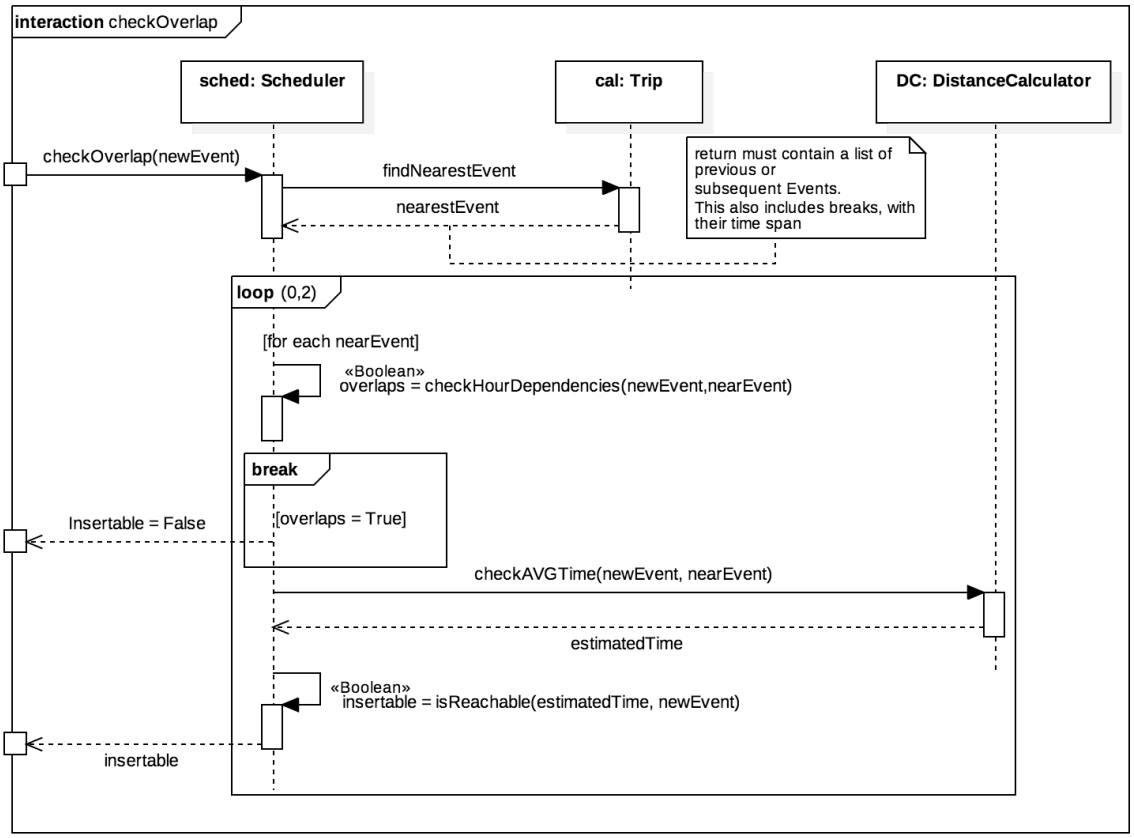
Actor	User.
Input Condition	User is already logged in into *Travlendar+*.
Event Flow	1. User clicks on "Preferences/Breaks". 2. User selects an interval and a minimum lenght for his break.
Output Condition	System adds those hours as a special meeting every day in the calendar.

3.2.2 Sequence Diagrams

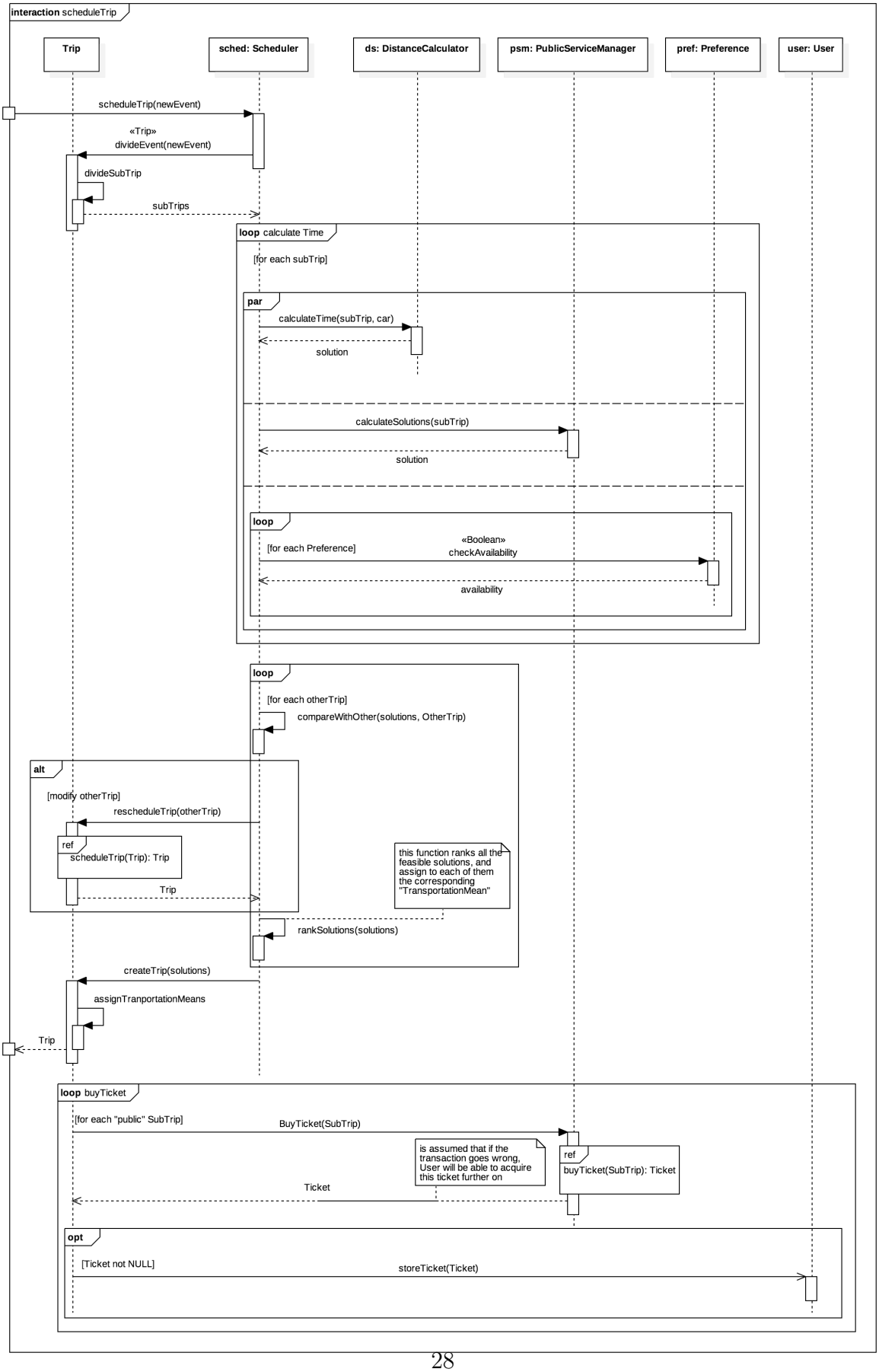
3.2.2.1 Create Event



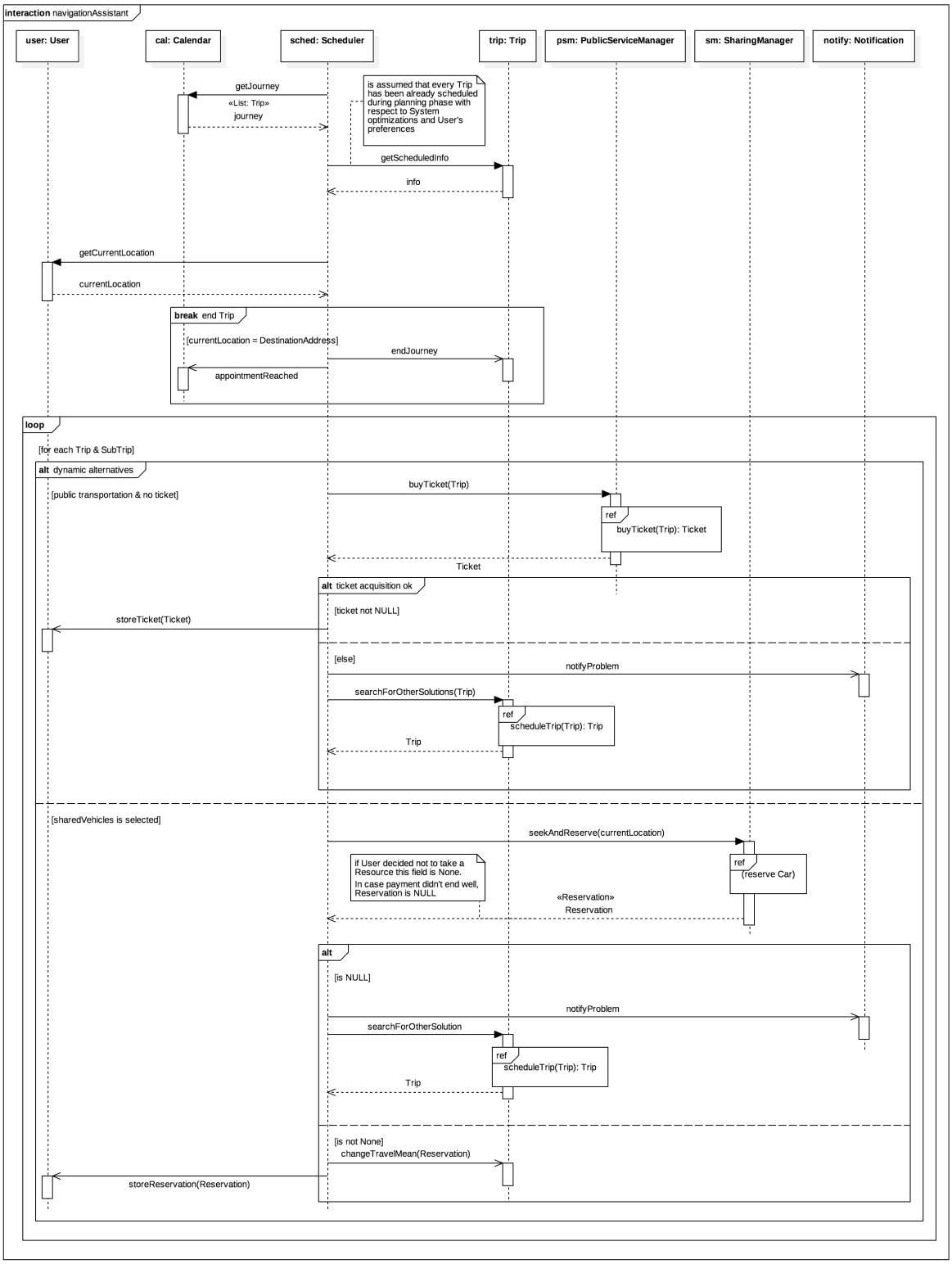
3.2.2.2 Subsquence: Check for Overlapping event



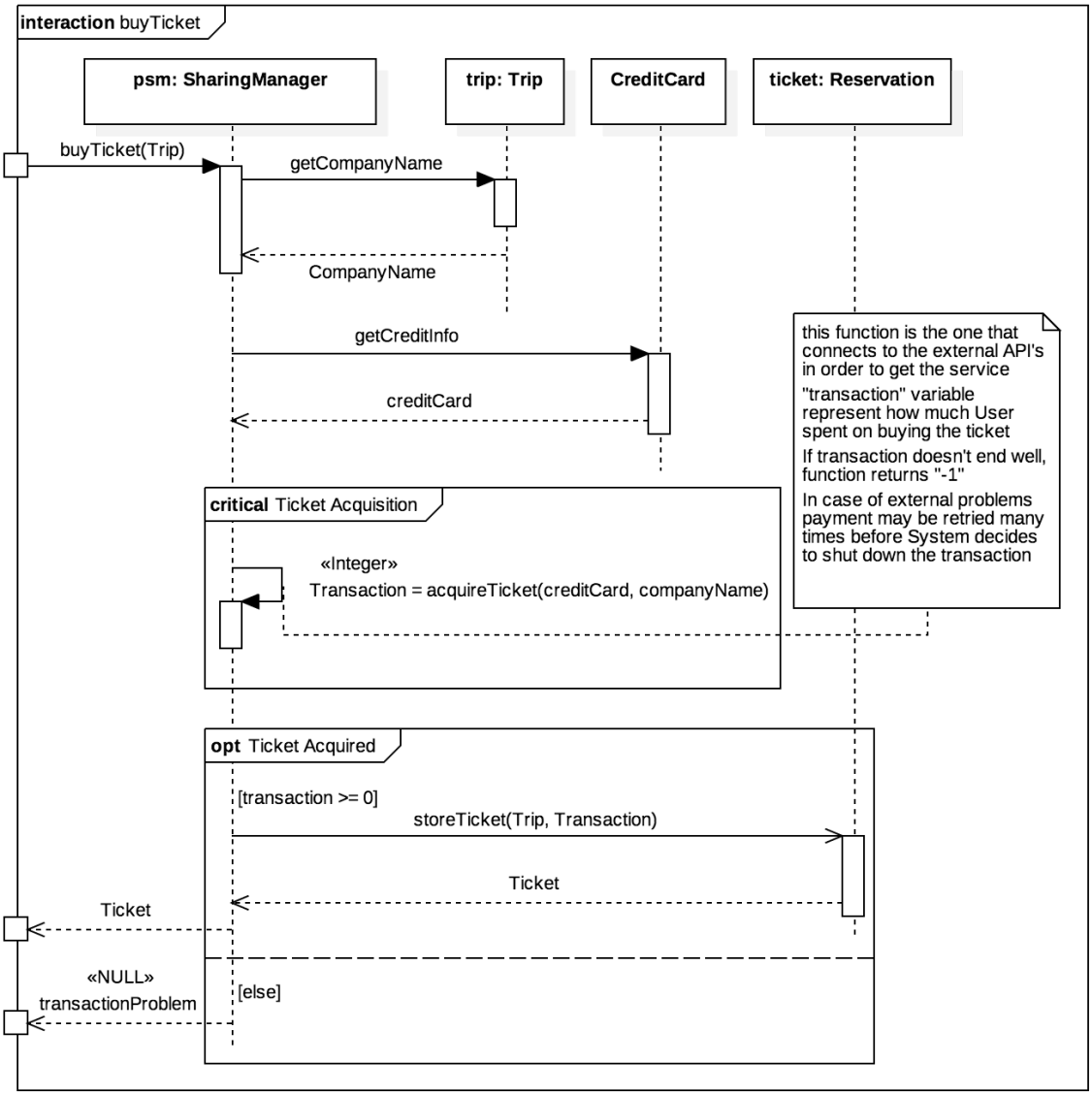
3.2.2.3 Subsquence: Schedule a Trip



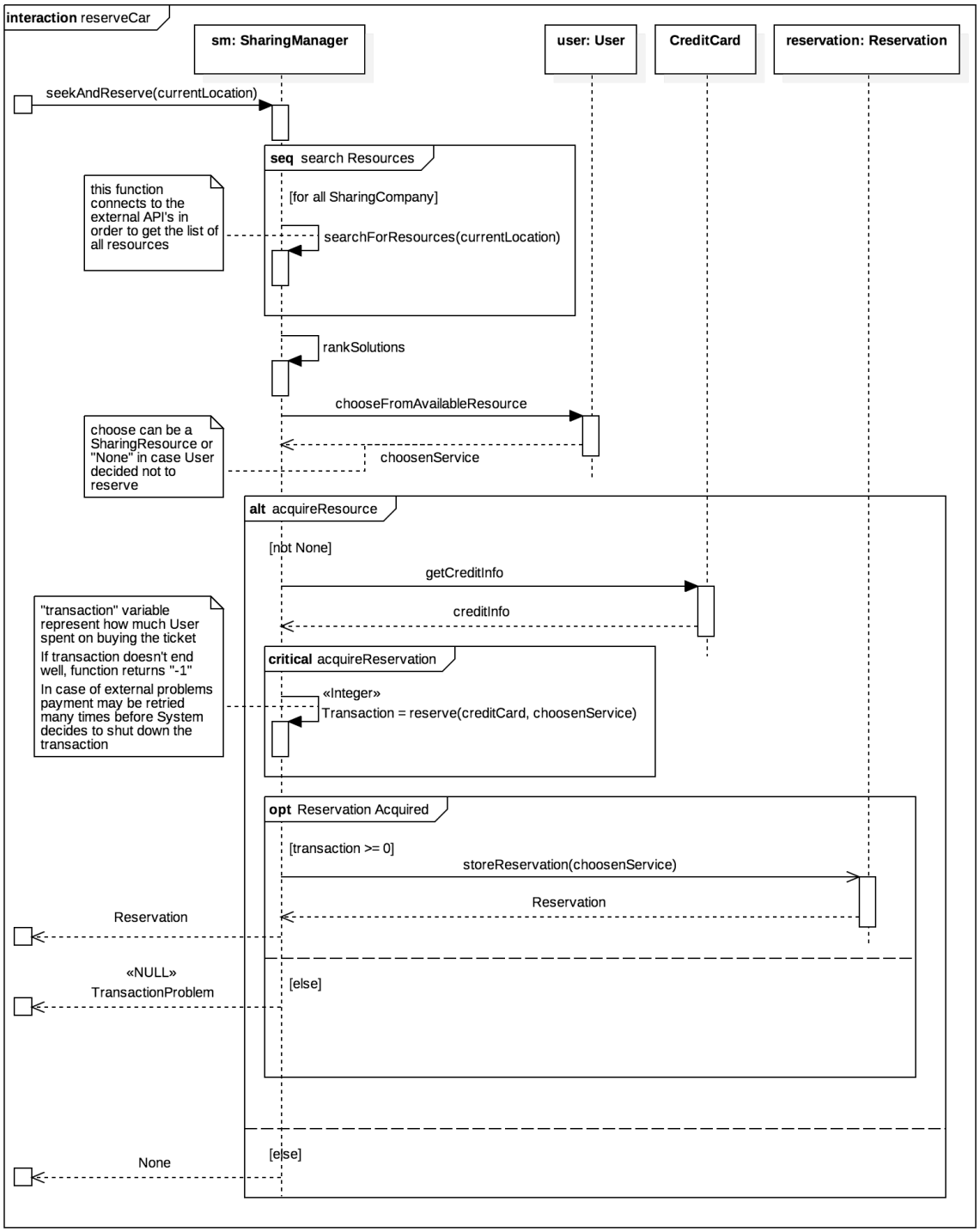
3.2.2.4 Dynamic Navigation



3.2.2.5 Buy Public Tranportation Ticket



3.2.2.6 Reserve Sharing Service Resource



3.3 Performance Requirements

This section is useful to see how the application manages statical and dynamical non-functional requirements in terms of quantity and quality. In particular:

Static Non-Functional Requirements

1. Application was developed to handle until 50.000 users simultaneously.
2. Mobile application must run on two most important operative system: iOS and Android.

Dynamic Non-Functional Requirements

1. 98% of the requests shall be processed in less than 2 second.
2. The system should be available 99.8% of the time in one year.

3.4 Design Constraints

Standards compliance The system must require User's location to work, so the application asks for it according to privacy laws the first time the user runs the application. The payment system is guaranteed by the corresponding external application, so *Travlendar+* doesn't ask users for their credit card number.

Hardware limitations Mobile app:

- Space for app package.
- 3G connection or better.
- GPS.

Any other constraint No further constraints are imposed.

3.5 Software System Attributes

Sofware System Attributes (also called *Non Functional Requirements*) describe how the system works and allow us to judge system's doing as it receives an high quantity of requests and its performance gets tested under such circumstances.

3.5.1 Reliability

The system has a 99.9/100 percent reliability since the only problems that could create system anomalies are those related to server management; payments and eventual interactions with sharing services are managed by their corresponding companies.

3.5.2 Availability

The system must be active 24/7 and servers are allowed to be out of service exclusively for system updates and the likes, possibly after a warning to the users.

3.5.3 Security

Users' credentials are stored and managed in according to privacy laws, so the user is safe from malicious agents.

3.5.4 Maintainability

In order to provide a perfect service, constant maintenance on the system is required. The application is updated every 14 days (except for urgent cases) with an average out-of-service time of 8 minutes thanks to the work of a senior engineer and three junior engineers.

3.5.5 Portability

The scope of this software is confined to the environment of mobile applications. Preferences transfer is allowed through the insert of user credentials so data like events, locations and personalized timetables can be exported and saved on the *Travlendar+* server.

4 Scenarios

Scenario 1

Matt's quietly working in his office when he receives a mail from his boss noticing him there's going to be an important meeting at 14:15 in another building of the company. Matt takes his smartphone and inserts the new appointment in his *Travlendar+* calendar, marking it as a 'working appointment'. Travlendar+ warns Matt that he can't take part in this meeting : he's already registered another event, 'take your daugher at school' in the time frame that spans from 13:30 to 14. Matt phones his wife : she's willing to substitute him, so he can delete the 'take your daughter at school'. *Travlendar+* has calculated the time required to get to the meeting and it gives Matt the best solution.

Scenario 2

Tim's boss contacts him in the evening noticing him his presentation has been delayed to the following day; the location itself has changed, too, forcing Tim to get to the renowned Samsara Beach. Consequently, Tim opens up his *Travlendar+* mobile app and reaches for the event 'presentation to the investors' in order to modify both time and place. *Travlendar+* calculates the distance and the possible travel means, it then informs Tim that the place (while widely known) can't be reached with public transportation and it isn't covered by any kind of Car Sharing service (it is out of the influence zone of *Travlendar+*). The app suggests Tim to go by car. Of course Tim agrees and for the rest of the day, *Travlendar+* will consider the car a favourite transport mean.

Scenario 3

David is 17 years old and has recently joined the soccer team of his city. His coach has fixed that training sessions will be held every Monday, Wednesday and Friday at 18:00. Then David opens *Travelandar+* and puts his commitments until the end of the season. The application suggests that the fastest way to go to the field training is the subway, so David decides to buy a season pass so that he can safely go to training sessions without having to ask his parents. After buying season pass he registered it on the application.

Scenario 4

Elizabeth loves dedicating the right amount of time to her appointments, from work to family to her hobbies. Recently though she's having a hard time conciling all of her commitments. That's why her friend Alex recommends her to use the *Travelandar+* application: Elizabeth follows his advice and downloads the application on her smartphone. It's only half an hour and Elisabetta is very satisfied, especially because she could set up an "Break Period" that allows her to devote the right amount of time to her lunch, denying the opportunity to add appointments around the 30 minutes dedicated to lunch.

Scenario 5

John had an hard day, he's just put the finishing touches on his project: he had to work even during this weekend, locked at home. Just when he's done with his assignment he receives an invitation to go out and see a movie with Jane and the rest of his friends. He's in a rush and he hasn't previously registered such an appointment in his calendar; to make things worse, he doesn't own a car, and public transportation is rather slow in the weekend. Because of this he rules out both car and the public transportation as travel means, and when he inserts location and time of the unexpected appointment only car sharing pops up as a viable and fast option. John obviously accepts and rents the car through *Travlendar+*.

Scenario 6

Luca is a nature-loving person, very passionate about environment and its well-being. He decided to download *Travlendar+* because it offers the possibility to minimize the carbon footprints and the usage of its own car. Since he lives in Milan Luca's usually lucky enough to rent and move with a bike of a shared-system; he's already chosen to go on foot or by bike as preferred travel means but unfortunately there's no way to predict whether a bike will be available or not at a given time. When the sun's up and the new day starts he checks on his phone and verifies that the nearest bike is definitely not worth the trip : he decides to go to work on foot.

Scenario 7

Tom is a bank employee working in Bologna. He decided to return at home in Milan next Friday to celebrate his father's birthday together with his family. Because of this he opens *Travlendar+* on his smartphone and creates a "Dad's party" event for Friday night. Tom's job does not allow him to leave Bologna before 18.00. Fortunately *Travlendar+* also allows him to find travel solutions by cross-region trains as well as by car. In fact, it all comes down to Tom's choice. He proceeds to buy train tickets: to him Friday isn't coming fast enough!

5 Alloy modeling

5.1 The Model

The following model concerns the most characterizing features of the system. We avoided to burden the model with trivial and non-significant details.

Data Types

```
//Alloy Model fo Travlendar+

//_____________________________________________
//_____________________DATATYPE_____________________
//_____________________________________________


//Importing Time-realated DataTypes
open util/time

//Datatype representing alphanumeric strings
sig Strings{}

//Datatype representing integer numbers
sig Integer {}

//Datatype representing boolean numbers
sig Bool {}


//Datatype representing floats
sig Floats{}
```

Signatures

```
//_____________________________________________
//_____________SIGNATURE_____________________
//_____________________________________________

//Calendar : multiple Calendars are allowed for predicates handling

 sig Calendar {
  appointments : set Appointment,
  breaks : seq Break,
  trips : seq Trip
} {
  not breaks.hasDups
  not trips.hasDups
}

//Break

sig Break {
  frameStart : one Time,
  frameEnd : one Time,
  minimumDuration : one Time,
  breakStart : one Time,
  breakEnd : one Time
}

//Appointment

sig Appointment {
  name : one Strings,
  date : one Time,
  time : one Time,
  address : one Strings,
  favouredVehicle : one Strings,
  type : one Strings
}


//Trip

sig Trip {
  departureAddress : one Strings,
  destinationAddress : one Strings,
  transportationMean : some TransportationMean,
  startTime : one Time,
  arrivalTime : one Time,
  calendar : one Calendar,
  carbonFootprints : lone Integer,
  eventId : one Strings
}


//Transportation Means and its related subclasses

abstract sig TransportationMean {}

sig SharedVehicle extends TransportationMean {
  type : one Strings,
  company : one Strings,
  sharing : one SharingManager,
}

sig Train extends TransportationMean {
  public : one PublicServiceManager
}
sig Tram extends TransportationMean {
  public : one PublicServiceManager
}
sig Bus extends TransportationMean {
  public : one PublicServiceManager
}
sig Car extends TransportationMean {
  distance : one DistanceManager
}
sig Bike extends TransportationMean {
  distance : one DistanceManager
}
sig Walking extends TransportationMean {
  distance : one DistanceManager
}
```

```
//User Management

abstract sig GeneralUser {}
sig Guest extends GeneralUser {}

sig User extends GeneralUser {
  name : one Strings,
  surname : one Strings,
  username : one Strings,
  password : one Strings,
  calendar : some Calendar,
  creditCard : set CreditCard,
  seasonPass : set SeasonPass,
  preference : set Preference,
  tickets : set Ticket
}

//User settings

sig CreditCard {
  cardNumber : one Integer,
  expirationDate : one Time,
  cvv : one Integer
}

sig SeasonPass {
  companyName : one Strings,
  validityTime : one Integer,
}

sig Ticket {
  companyName : one Strings,
  type : one Strings,
  date : one Time,
  name : lone Strings,
  reservedSeat : lone Strings
}

//Preference Management

sig Preference {
  type : one Strings,
  description : one Strings,
  selected : one Bool,
  scheduler : one Scheduler
}

lone sig CarbonPreference extends Preference {
  quantity : one Integer
}


//Travel Scheduling and Warning notifications

one sig Scheduler {
  notify : set Notification,
  trips : set Trip,
  weatherForecaster : one WeatherForecaster,
  sharingManager : one SharingManager,
  publicServiceManager : one PublicServiceManager,
  distanceManager: one DistanceManager,
  excludedVehicles : set TransportationMean,
  calendar : some Calendar
}

sig Notification {
  id : one Strings,
  message : one Strings
}


//External Modules

one sig WeatherForecaster {  scheduler : one Scheduler}
one sig SharingManager {scheduler : one Scheduler}
one sig PublicServiceManager {scheduler : one Scheduler}
one sig DistanceManager {scheduler : one Scheduler}


//Reservation of Shared Vehicles

sig Reservation {
  date : one Time,
  cCard : one CreditCard,
  sharedVehicle : lone SharedVehicle
}
```

Facts

```
//_____________________________________________
//_____________________FACTS_____________________
//_____________________________________________

//All User's preferences can't exist without the corresponding user

fact creditCardsDependency {
  all c : CreditCard | some u : User | c in u.creditCard
}

fact seasonPassDependency {
  all s : SeasonPass | some u: User | s in u.seasonPass
}

fact preferenceDependences {
  all p : Preference | some u : User | p in u.preference
}

fact ticketDependency {
  all t : Ticket | some u: User | t in u.tickets
}


//All the breaks and the appointments can't exist without a Calendar to refer to

fact appointmentsDependency {
  all a : Appointment | some c : Calendar | a in c.appointments
}

fact breaksDependency {
  all b : Break | some c : Calendar | b in univ.(c.breaks)
}


//All notifications must refer to a Scheduler

fact notificationDependency {
  all n : Notification | some s : Scheduler |
    n in s.notify
}


//All transportation means must refer to a trip

fact tripRequired {
  all t : TransportationMean | some tr : Trip | t in tr.transportationMean
}


//User must allow at least a transportation mean when looking for travel solutions

fact oneTravelMean {
  all s : Scheduler | some t : TransportationMean | t not in s.excludedVehicles
}


//Two notifications with the same id can't possibily cohexist

fact noIdenticalNotify {
  no disjoint n1,n2 : Notification | n1.id = n2.id
}


// Trip's got to be directly related to the Scheduler

fact allTripsAreLinked {
  all t : Trip | some s : Scheduler | t in s.trips
}


//There can't be two identical excluded vehicles in the Scheduler

fact noTwoIdenticalTransportationMeans {
  all s : Scheduler | all disjoint t1,t2 : s.excludedVehicles |
   t1 != t2
}

//User must have always "walking" active in his travel mean preferences

fact walkingActive{
    all t : Trip, s : Scheduler | all id :  Trip.eventId | some w : Walking | w in t.transportationMean and t.eventId = id and not (w in s.excludedVehicles)
}
```

```
//Trips are not allowed during break time

fact noTripDuringBreak {
  no t : Trip | some b : Break | (gte[t.arrivalTime, b.breakStart] and lte[t.arrivalTime, b.breakStart + b.minimumDuration]) or
                                 (gte[t.startTime, b.breakStart] and lte[t.startTime, b.breakStart + b.minimumDuration])
}


//I trips devono essere inclusi nel frame time

fact withinFrame {
  no b : Break | lt [b.breakStart, b.frameStart] or gt[b.breakEnd, b.frameEnd]
}


//Every trip has to show carbon footprints if I expressed a preference regarding carbon footprints

fact showCarbonFootprints  {
  all t: Trip | some CarbonPreference implies #t.carbonFootprints > 0 else #t.carbonFootprints = 0
}
```

Predicates

```
//_____________________________________
//_____________Predicates_____________
//_____________________________________

// Inserting a new appointment into the calendar

pred insertAppointment [a : Appointment, c : Calendar , c' : Calendar, u : User] {
  //preconditions
        a not in c.appointments
  //postconditions
  c'.appointments = c.appointments + a
  c'.breaks = c.breaks
  c'.trips = c.trips
        u.calendar = (c + c')
}


// Reserving a Car

pred reserving [ s : SharedVehicle, r' : Reservation, r : Reservation] {
  //no preconditions
  #r.sharedVehicle = 0
  //postconditions
  r'.date = r.date
  r'.cCard = r.cCard
  r'.sharedVehicle = r.sharedVehicle + s

}


// Ticket purchasing

pred  ticketPurchase [ t : Ticket, u1 : User, u2 : User ] {
  //preconditions
  not t in u1.tickets
  //postconditions
  u2.name = u1.name
  u2.surname = u1.surname
  u2.username = u1.username
  u2.password = u1.password
  u2.calendar = u1.calendar
  u2.creditCard = u1.creditCard
  u2.seasonPass = u1.seasonPass
  u2.preference = u1.preference
  u2.tickets = u1.tickets + t
}


// Adding SeasonPasses and CreditCards to User's content

pred addSeasonPassAndCreditCard [ s : SeasonPass, c: CreditCard, u1: User, u2: User]{
  //preconditions
  not s in u1.seasonPass
  not c in u1.creditCard
  //postconditions
  u2.name = u1.name
  u2.surname = u1.surname
  u2.username = u1.username
  u2.password = u1.password
  u2.calendar = u1.calendar
  u2.creditCard = u1.creditCard +c
  u2.seasonPass = u1.seasonPass + s
  u2.preference = u1.preference
  u2.tickets = u1.tickets
}
```

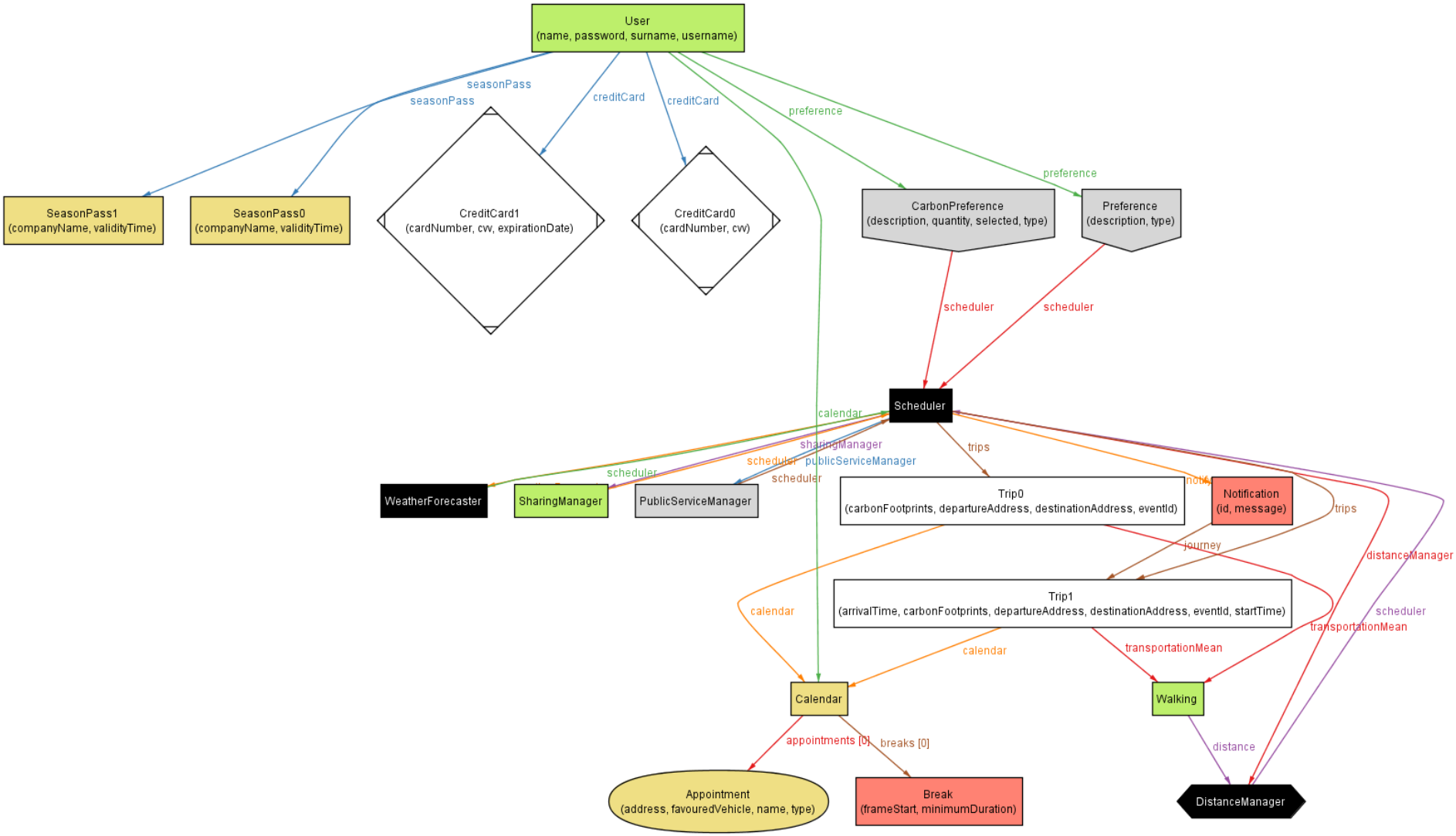
5.2 Results

```
Executing "Run insertAppointment"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   7175 vars. 722 primary vars. 12003 clauses. 53ms.
   Instance found. Predicate is consistent. 55ms.

Executing "Run reserving"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   6977 vars. 719 primary vars. 11425 clauses. 40ms.
   Instance found. Predicate is consistent. 65ms.

Executing "Run ticketPurchase"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   7154 vars. 719 primary vars. 11932 clauses. 33ms.
   Instance found. Predicate is consistent. 51ms.

Executing "Run addSeasonPassAndCreditCard"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   7206 vars. 722 primary vars. 12023 clauses. 38ms.
   Instance found. Predicate is consistent. 49ms.
```

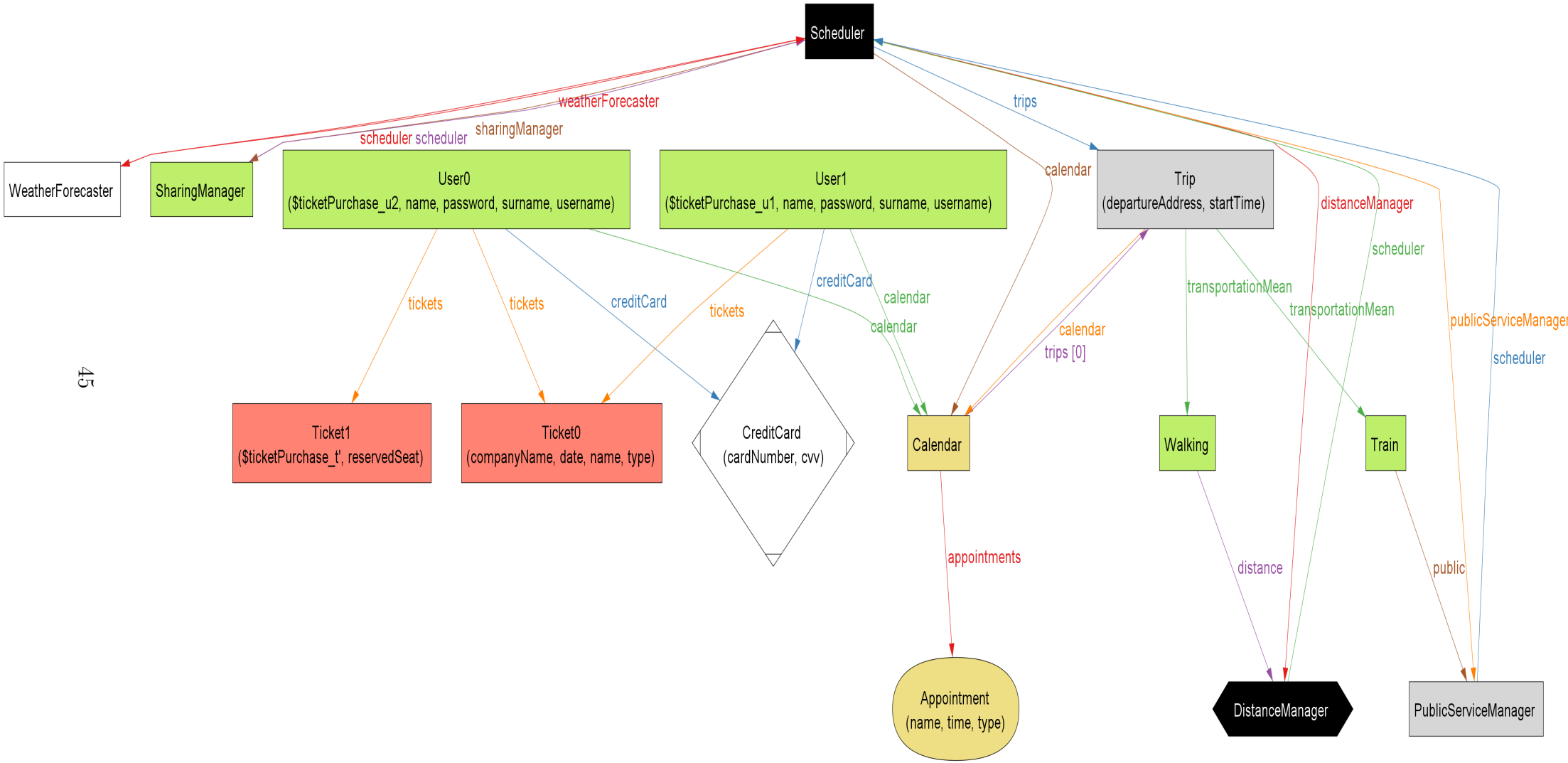
Figure 5: Result of the model analysis.

5.3 Generated World

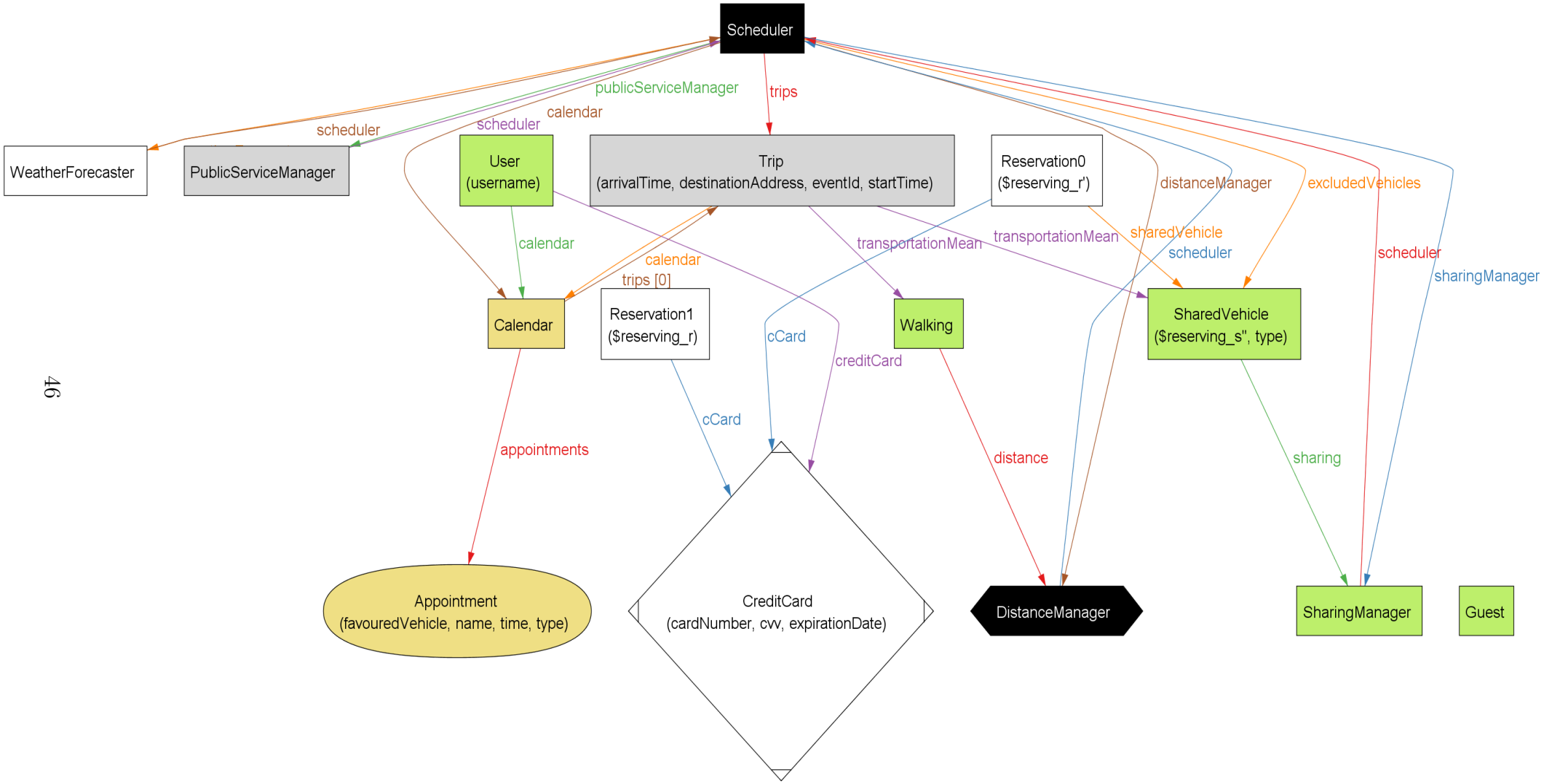
5.3.1 World Generated



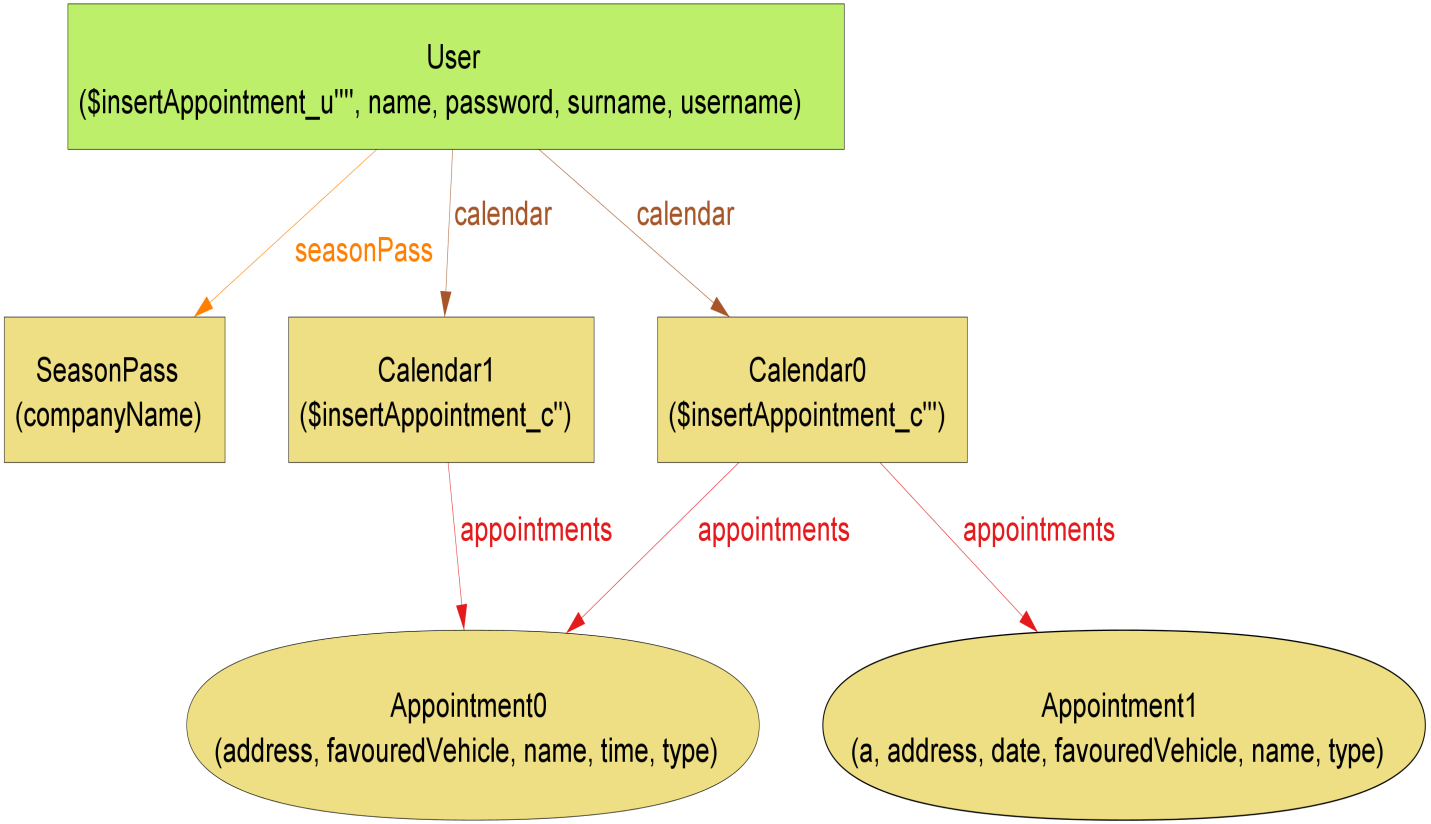
5.3.2 Ticket Purchase



5.3.3 Renting


Scheduler
WeatherForecaster
PublicServiceManager
User
(username)
Trip
(arrivalTime, destinationAddress, eventId, startTime)
Reservation0
($reserving_r')
Calendar
Reservation1
($reserving_r)
Walking
SharedVehicle
($reserving_s'', type)
Appointment
(favouredVehicle, name, time, type)
CreditCard
(cardNumber, cvv, expirationDate)
DistanceManager
SharingManager
Guest
scheduler
publicServiceManager
calendar
scheduler
trips
distanceManager
excludedVehicles
sharedVehicle
scheduler
scheduler
sharingManager
calendar
calendar
trips [0]
transportationMean
transportationMean
cCard
creditCard
cCard
appointments
distance
sharing

5.3.4 Insert Appointment



6 Appendix

List of Figures

List of Tables

6.1 Used tools

For this assignment, we used the following tools:

Alloy We used the alloy tool to write the code and check the models for the specification.

LaTeX The group used LaTeX to structure the final document and to help with versioning.

Github We leaned on Github for versioning and coordinating synchronized work.

StarUML We used StarUML to make Use Case, Class and Sequence Diagrams. StarUML.

Balsamiq Mockups We used Balsamiq Mockups to create mockups of our mobile application interface. Balsamiq Mockups

6.2 Hours of work

Bisica, Leonardo around 66 hours of work;

Castellani, Alessandro around 65 hours of work;

Cataldo, Michele around 63 hours of work.