POLITECNICO
MILANO 1863

Software Engineering II

Travlendar+

Requirements Analysis
and
Specifications Document

Authors:

Edoardo D'Amico
Gabbolini Giovanni
Parroni Federico

1st October 2017

Indice

Chapter 1

Introduction

1.1 Purpose

Our team will develop Travlendar+, a calendar-based application that aims to provide a schedule of user appointments, giving a plan to organize his daily life. The main goals the app must fulfill are:

G1 The system should offer the possibility to create a new account;

G2 The system should be able to handle a login phase;

G3 The system should give to the signed user the possibility to recover his password.

G4 Allow the user to insert an appointment according to his necessities and his preferences;

G5 The system S.P.W. to modify an inserted appointment;

G6 The system S.P.W. to create a valid schedule of the user appointments when requested (**fare ref alla def**) and display the scheduling result **(fare riferimento alle definizioni)**;

G7 The system should let the user create valid multiple schedules and decide which one is chosen for the current day;

G8 The system S.B.A to book the travel means involved in the current schedule under user approval;

G9 The system S.B.A. to display in real time user position and the directions to be followed in order to arrive to the next appointment on a dinamically updated map;

1.2 Scope

Here we provide a brief description of the aspects of the reality of interest which the application is going to interact with.

User can receive an appointment on a certain date, time and location (over a region), that can be reached using different available travel means. The appointment can be held either at a specific time or in a time interval and lasts for a certain amount of time. An appointment can be recurrent, in other words, it repeats regularly over time (e.g., lunch, training, etc.). User can travel with someone else and can pick up or leave off these people during the day.

User can have his own travel means and a pass for public transportation. The travel means considered in this scenario can be grouped in three categories: public, shared or private.

- Public travel means: these include trains, buses, underground, taxis, trams. They have to be taken in their designated stops. User must have a valid ticket in order to get on a public travel means (except for taxis, that pick up the user wherever he wants upon a call and do not require any ticket);
- Shared travel means: these include car and bike. They are located in specific places and require a reservation in order to be used by the user;
- Private travel means: vehicles owned by the user. They can be cars, bikes, motorbikes. Also walking is considered to be a (very special) private travel mean.

Weather conditions can change during the day affecting travel means choice. At the beginning of the day, or on demand, user can request a schedule of his daily appointments, following some criteria evaluated according to their assigned priority and satisfying some constraints imposed by the user. When a new appointment is received, user creates a new item in the application and saves it in the appointment list. User can request the application to reschedule the appointments because of unexpected changes of his plan (e.g. a cancelled appointment).

1.2.1 World Phenomena

- User receives a new appointment;
- User picks up a person;
- User owns private travel means and/or passes for public transportation;
- User wakes up;
- User pass expires;
- Exists various travel means.

1.2.2 Shared Phenomena

- Shared travel mean moves;
- Shared travel mean is not available anymore;
- Wheather condition changes;
- Public travel means reach a stop-place;
- Public travel means are late;
- Public travel means are not available due to a strike day;
- User requests a schedule to the machine;
- User inserts a new appointment into the application;
- User requests to book rides;
- User moves.

1.3 Definitions, Acronyms, Synonims

1.3.1 Synonims

Are synonims:

- Appointment and meeting

- Schedule and Scheduler (**?????????**)
- System and Application
- Preference and Constraint

1.3.2 Definitions

Definition 1.3.1. A preference is a constraint on appointment or a schedule.

la toglierei questa in quanto abbiamo definito i optimization criteria nella sezione schedule model

Definition 1.3.2. An Optimization Criteria is a criteria followed by the scheduler in order to optimize.

Definition 1.3.3. A Travel Pass is ...

Definition 1.3.4. A Travel Option is a combination of travel path and travel means that allow to reach one spot from another.

Definition 1.3.5. The Travel Option Data are additional information about a travel option:

- Cost;
- Traveling time;
- Carbon emission;
- distance (KM);
- Graphical representation of the path.

Definition 1.3.6. A Schedule is a set of time-ordered and not overlapping appointments where their starting times are fixed and they're linked to each other by a path covered with a specific transportation mean.

Definition 1.3.7. A Valid Schedule is a Schedule which:

- is optimized according to the criteria chosen by the user;
- ensures that the user will be on time for all his appointments;

- respects the constraints imposed by the user.

Definition 1.3.8. A Relative Path is a portion of a path travelled by the same travel means.

Definition 1.3.9. A Scheduling Result is the set:

- Graphical representation of the path that will be travelled by the user
- money spent for each relative path
- total money spent
- Length of the path expressed in KM
- Length of relative path
- Carbon footprint emission
- Estimated travel duration of each relative path
- Total estimated travel time

Definition 1.3.10. The Current Appointment is an appointment which has startingTime >= currentTime and data=currentDate, where currentTime and currentDate are the actual time reference of the system.

1.3.3 Acronyms

Acronyms used in the text:

- GPS: Global Positioning System
- GUI: Graphical User Interface
- ETA: Estimated Time of Arrival
- S.P.W.: should provide a way
- S.B.A.: should be able
- API: Application Programming Interface
- CRUD: create/read/update/delete
- URL: Uniform Resource Locator

1.4 Revision history

1.5 Reference documents

1.6 Document structure

Chapter 2

Overall Description

2.1 Product Perspective

2.1.1 User Model

A user is represented within the application by his password and his e-mail. Some important informations about him are held by the following parameters:

- *travelPass*: indicates if the user has a pass for public transportation;
- *hasBike*: indicates if the user has his own car;
- *hasCar*: indicates if the user has his own bicycle.

2.1.2 Appointment Model

An appointment is represented within the application by a set of parameters:

- *duration*: the time extension of the appointment
- *date*: the day in which the appointment is held;
- *startingTime* or *timeInterval*: the first should be given if the starting hour is well-known (deterministic), otherwise a time interval in which the appointment will be held it's provided.
- *location*: identifies the coordinates of the place where the appointment will be held;
- *recurrent*: specifies if the appointment will be repeated over a fixed period of time;

- *peopleTravelling*: represents a variation occourring when the user picks up or leaves off someone.

The life cycle of an appointment can be represented by the following statechart:

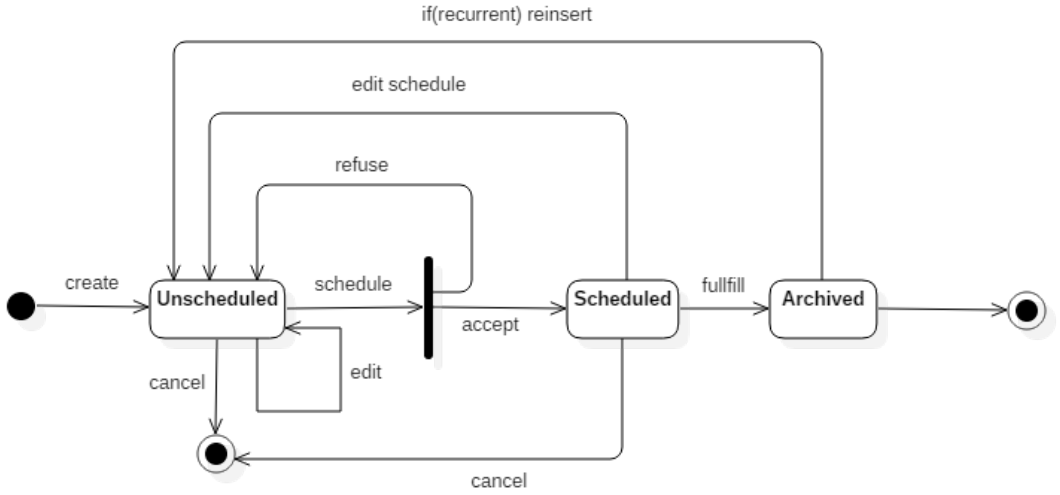


Figure 2.1: Appointment statechart

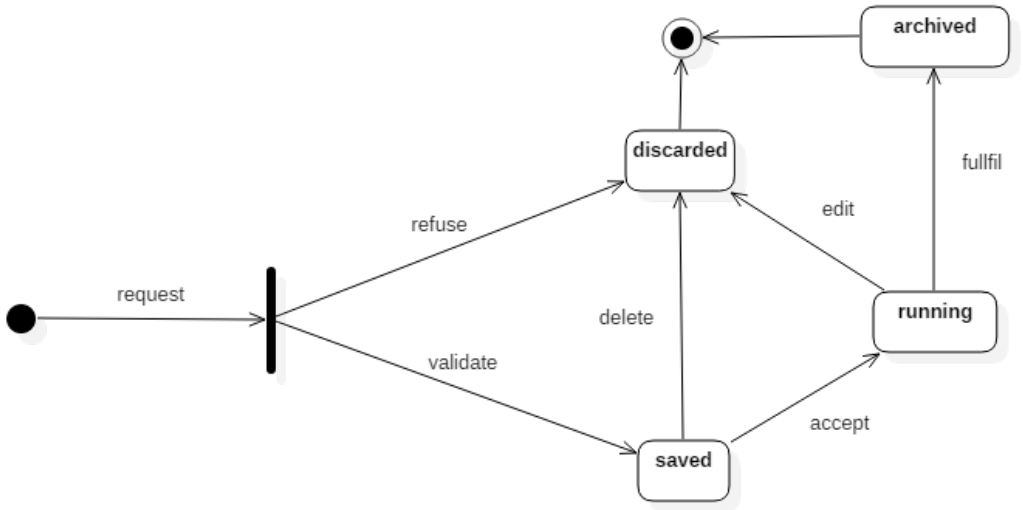
A newly-created appointment is **Unscheduled**. It could remain **Unscheduled** either when edited or there isn't a possible arrangement when a schedule is performed. Otherwise it becomes **Scheduled** if there's a feasible way to arrange it. When a scheduled appointment is edited all the appointments in that schedule return to be **Unscheduled**, because they can possibly cause a different schedule. When a scheduled appointment is fullfilled it becomes **Archived** and stored in the schedule history. If this last one is a recurrent appointment it must be reinserted in the list of unscheduled appointments so it will become **Unscheduled** again. The user can cancel an appointment in every moment.

2.1.3 Schedule Model

A schedule is a subset of Appointments of a given day, ordered by the scheduler following the criteria described below. A schedule is characterized by the following variables:

- *date*;

- *startingPosition*: is the starting location of the user (e.g. user's home);
- *startingNumberOfPeople*: the number of people that must reach the first appointment.
- *wakeUpTime*: it is the starting time from which the schedule should start arranging appointments.



When a schedule is requested by the user it can be either validated or refused by the scheduler according to the definition of valid schedule **fare riferimento alla def**. In the first case the schedule is **saved** in the second one the schedule is **discarded** and a warning to the user is sent. In order to start a schedule the user must accept one of them from the saved ones, after that the schedule is **running**.If the user edits one of the appointments belonging to the running schedule, this one is not more valid and become **discarded**.when a schedile is fullfilled by the user this last is **archived** by the system.

The optimization criteria

Prioritizing criteria for the schedule optimization that can be chosen by the user are the following:

- *Minimize carbon footprint*: the scheduler will try to minimize the amount of kilometers travelled in polluting means;
- *Minimize money spent*: the scheduler will try to avoid expensive means and to exploit the public ones (especially if the user has a pass) or going by bike or on

foot;

- *Minimize travelling time*: the scheduler will compute the quickest possible path reaching all the appointments locations.

2.1.4 Constraints

Constraints are impositions on some parameters managed by the system during the process of scheduling the appointments. We can distinguish between constraints on schedule and contraints on the single appointment. These can be selected by the user when he inserts an appointment or when he requests a schedule, otherwise the constraints are initialized to default values.

Constraints on schedule

- *Maximum travelling distance with a specific travel mean*: the user can set a maximum amount of km to travel with a specific travel mean;
- *Travel means time slots*: user can specify a time interval in which a travel mean can be used;
- *User can deactivate a particular travel mean*;
- *User can select which travel mean he uses under certain weather condition.*

Constraints on appointment

- *User can deactivate a particular travel mean.*

2.1.5 Class Diagram

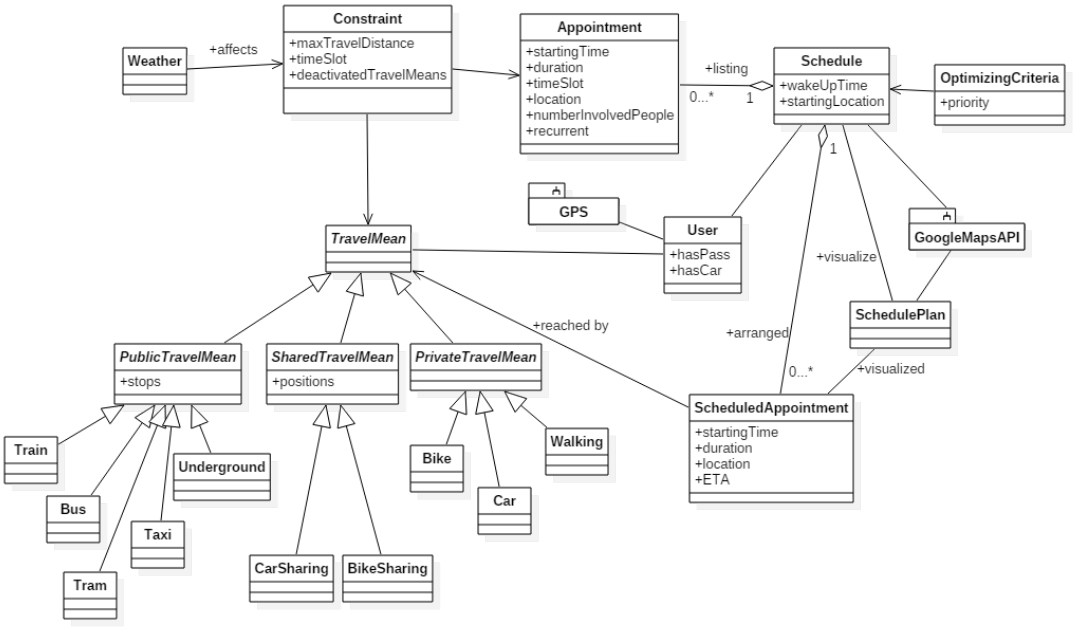


Figure 2.2: System Class Diagram

2.2 Product Functions

The following requirements are derived in order to fullfill the specified goals.

Requirements for **G4**:

1. The system should offer a GUI that handles All the interactions with the user.

Requirements for **G5**:

R2 The system S.B.A. to retrieve information from the user about his appointments;

R3 The system S.B.A to store an appointment in his memory;

Requirements for **G6**:

R4 The system should let the user change the parameters and the constraints of an inserted appointment;

R5 The system S.B.A to rewrite the appointment in his memory with his new parameters;

Requirements for **G7**:

R6 Allow the user to set the constraints of the schedule (**riferimento ai constraint dello schedule**), or to accept the default values;

R7 Allow the user to set the optimization criteria (2.1.3) for the schedule;

R8 Allow the user to set the variables for the schedule;

R9 The system S.B.A. to gather information from external APIs about:

- travel options with related travel option data;
- weather forecast;
- strike days;

R10 The system S.B.A. to select the best travel option according to the optimization criteria taking into account:

- user constraint
- user parameters **(aggiungere riferimento a user model)**
- travel option data
- weather forecast
- information about strike day

R11 The system S.B.A to store valid schedules requested by the user

Requirments for **G1**:

R12 The system should let the user accept a schedule from the saved ones

Requirements for **G2**:

R13 The system S.B.A to handle a registration phase in which the user will provide an e-mail and a password

R14 The system S.B.A. to verify the e-mail given by the user by sending a confirmation e-mail to his address

Requirements for **G??**:

R15 The system S.B.A to recognize a registered user given an e-mail and a password

R16 The system S.B.A. to retrieve information from the user about his registration informations, i.e. his e-mail and password (**da verificare se serve o meno**);

Requirements for **G??**:

R17 The system should offer to the user a way to link all his travel service accounts into the Travlendar+ account;

R18 The system S.B.A to book a travel mean through external API offered by third party application in which the user is signed

Requirements for **G??**:

R19 The system S.B.A to retrive the graphical representation of a path from an external API

R20 The stystem S.B.A to retrive the travel option from an external API **(questa che già si è detta ma serve anche per questo goal come facciamo?)**

R21 The S.B.A to retrieve the length of a path from an external API

Requirements for **G??**:

R22 The system S.B.A to retrive the position of the user from his GPS

R23 The system S.B.A. to retrive from an external API the directions to give to the user for reach the next appointment;

Requirements for **G??**:

R24 The system should be able to recover an e-mail address from the user;

R25 The system should be able to send the password of a user to his e-mail given the e-mail.

2.2.1 User characteristics

Users can use our system when they want something that allows them to schedule their meetings according to their necessities and constraints. Necessary conditions for the users in order to use the system are:

- He must have an **(specificare meglio in seguito in quale dispositivo occorre farla girare)** connected to the internet in which the application runs

This is the only requirement that is needed. Anyway additional characteristics of the user lead to the exploitment of all the system features. In fact some of them are guaranteed only after having submitted some information to the application. In this sense, welcomed user's characteristics are:

- The ownership of some travel means
- The ownership of travel passes
- The registration to sharing services

Beside these, an obvious tacit assumption is that the user has a valid age to move where he wants with autonomy.

2.2.2 Assumptions, dependencies and constraints

- The system should be able to retrieve information about public travel means. In particular:
  - information about delays;
  - information on possible strike days.
- The system should be able to retrieve information about shared means. In particular:
  - position of the available ones;
  - prices per time unit;
- There exist external APIs that allow to:
  - retrieve all travel options and travel option data
  - signed user to book and pay for all travel services
  - retrieve information about weather forecast
  - retrieve a graphical representation of a path
  - retrieve the lenght of a path

- The device on which the application runs is connected to the internet;
- The user has a GPS active in every moment;
- Every user has at least one personal e-mail;

Chapter 3

Specific requirements

3.1 External Interface Requirements

The application shows its best potential when running in a mobile device, for instance a smartphone or a tablet. This permits to extend the features and the automatic tasks of the application, thanks to the built-in device functionalities. However, a computer client version of the application can be installed, too.

3.1.1 User interfaces

The user can interact with the application through several graphical interfaces:

1. **Registration/login interface**: allows the user to insert credentials in order to registering or logging into the system;

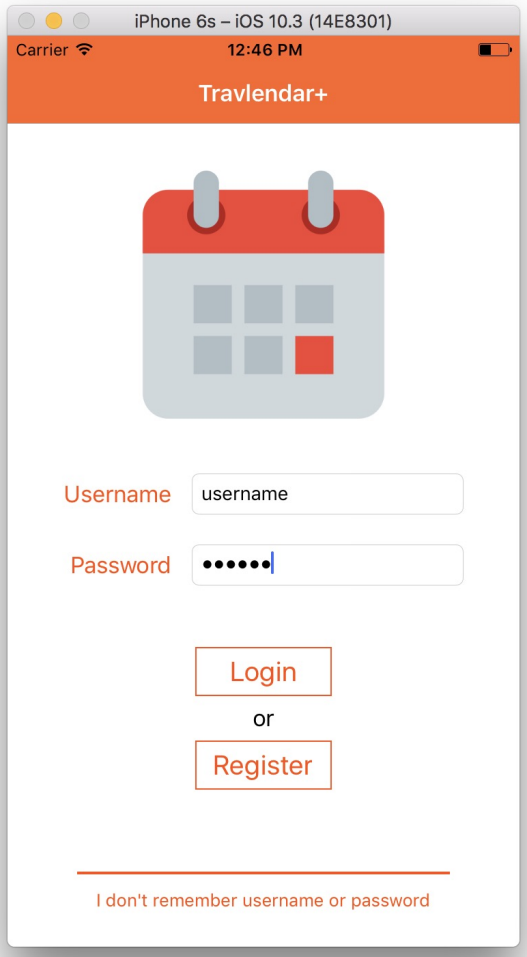


Figure 3.1: Registration/login interface

2. **User account interface**: user can specify his profile characteristics, such as his passes, car and/or bike ownership;

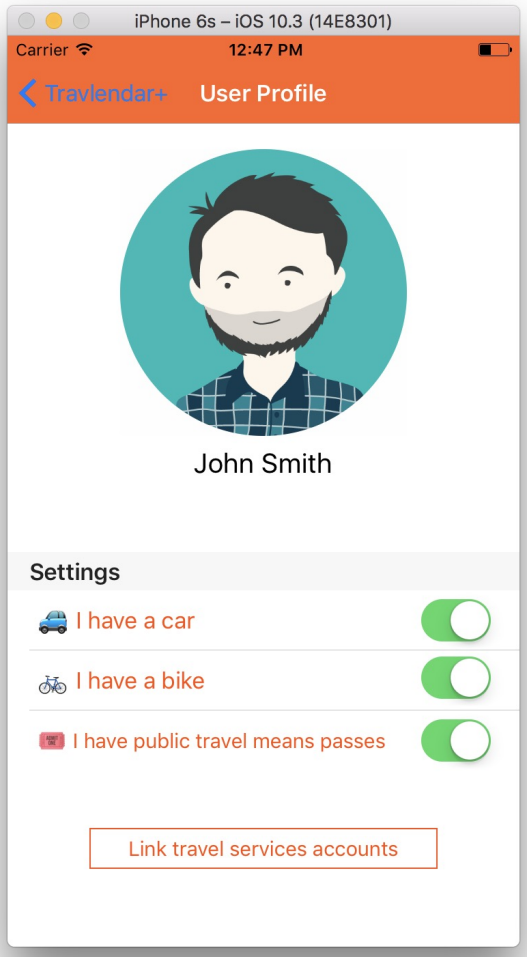


Figure 3.2: User account interface

3. **Home interface**: shows currently running schedule and displays some navigation links to other interfaces;

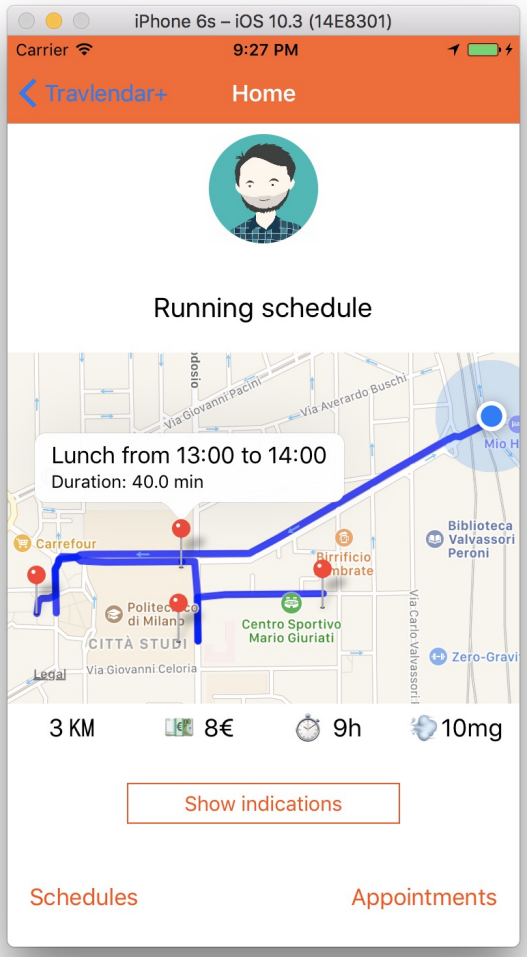


Figure 3.3: Home interface

4. **Appointment CRUD interface**: allows creating, showing and editing appointment parameters and related constraints;

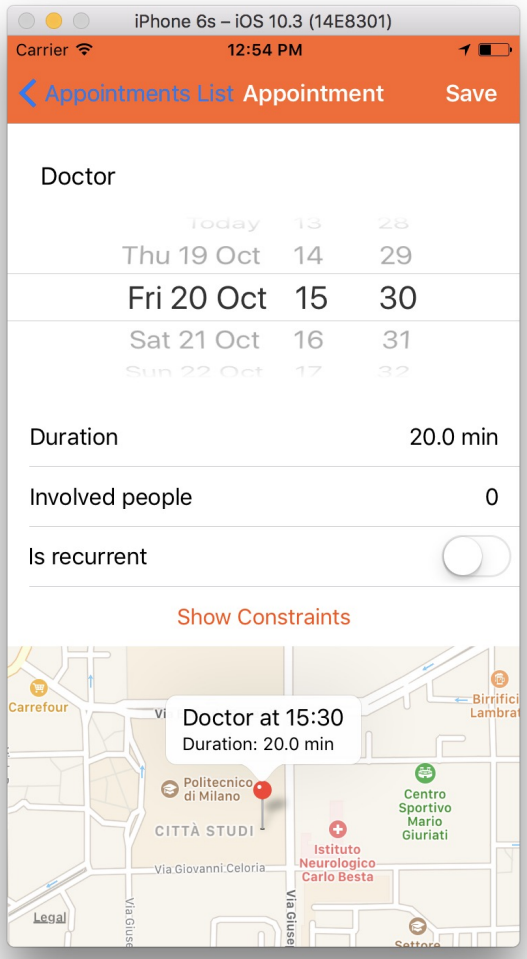


Figure 3.4: Appointment CRUD interface

5. **Appointments list interface**: provides a list of all inserted appointments, with the possibility to filter between non-scheduled/scheduled ones (includes the possibility to delete an item of the list);

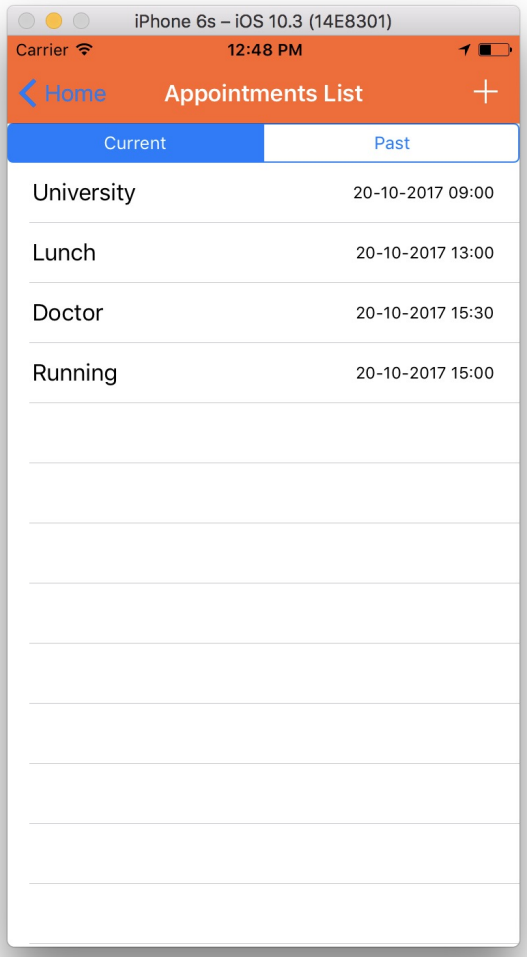


Figure 3.5: Appointments list interface

6. **Schedules list interface**: display a list of the created schedules

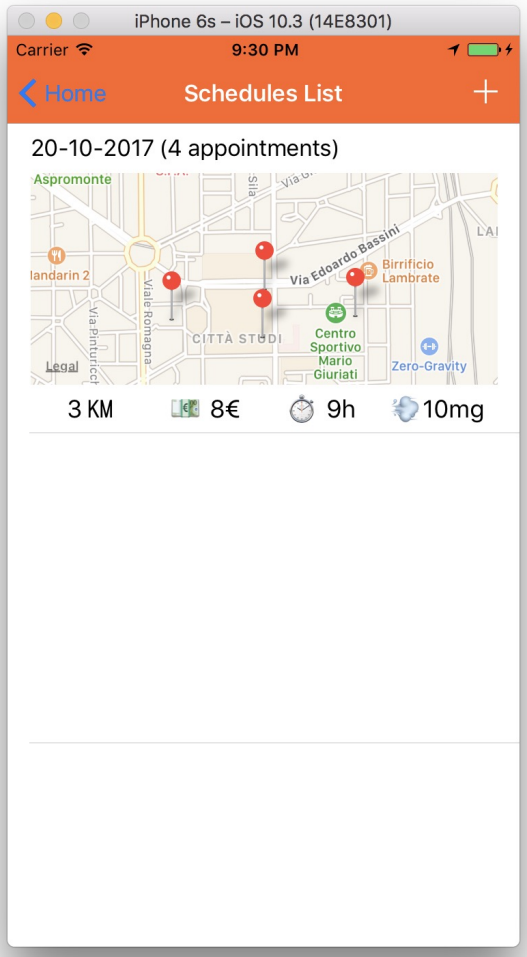


Figure 3.6: Schedules list interface

7. **Schedule interface**: user can set parameters, contraints, optimization criteria and request a schedule creation for a given date;

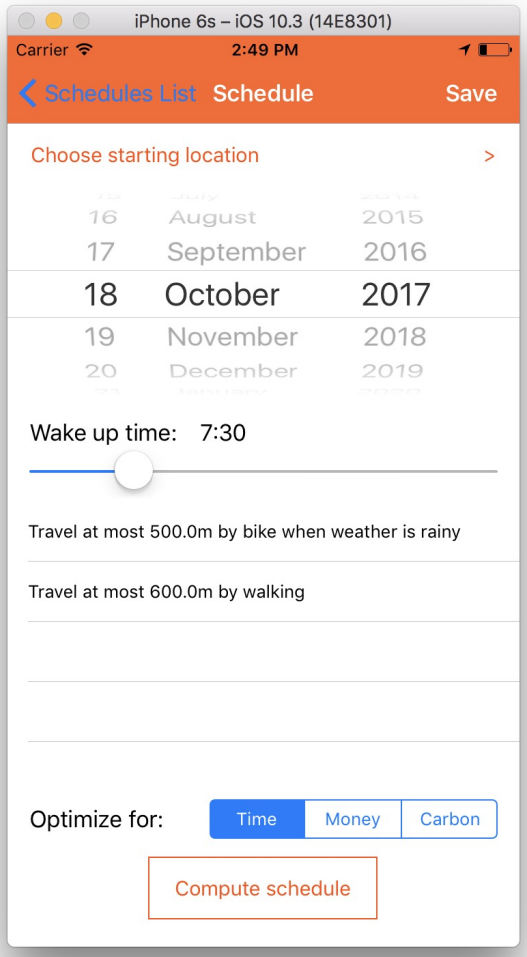


Figure 3.7: Schedule interface

8. **Schedules result interface**: shows the computation of the requested schedules for a given date and asks the user to select one, then waits for confirmation for that;

9. **Schedule progress interface**: permits to keep track of the completeness percentage, indicating the directions to be followed by the user in a map, in order to arrive to the next appointments;

10. **Tickets/rides reservation interface**: allows user to buy tickets for public travel means and/or reserve a ride for the shared travel means;

3.1.2 Hardware interfaces

Hardware interfaces are physical linking across which two or more separate components of a system exchange information. A hardware interface is described by the mechanical and electrical signals at the interface and the protocol for sequencing them. There are no interesting hardware interfaces in our scope.

3.1.3 Software interfaces

Software interfaces are logical linking across which two or more separate applications running on a system exchange information. The most relevant software interface in our system is API. APIs are sets of subroutine definitions, protocols and clearly defined methods of communication, allowing data exchanging and service requests. There are several kinds of these:

- **Operating System APIs**: specify interface between applications and OS, permitting to access low level routines calls (for instance, to communicate with memory or with an internal device)**(!)**
- **Remote APIs**: DBMS expose a set of standards that the API user can adopt in order to manage the database data. SQL is the standard language for storing, manipulating and retrieving data in this context;
- **Web API**: information can be exchanged through the internet by encapsulating it in HTTP request/response. Weather forecast, travel services, mapping systems offer this typology of API.

3.1.4 Communications interfaces

Communication interfaces allows two different architectures of the system to exchange information through communication channel. These non-homogeneous components of the system can communicate thanks to the following software interfaces and protocols:

- **Cellular connectivity**: mobile devices can connect to the internet thanks to LTE standard;
- **GPS**: cellular can retrieve his coordinates position through NMEA protocol;

- **QRCode**: associates a matrix of bits to an URL. QRCodes are present in most of the shared means, semplifying the booking of that.

3.2 Functional requirements

3.2.1 Scenarios

Here are some scenarios that describe the usage of the system.

Scenario 1

Giovanni will start the fourth year of his Master's degree. Surfing the internet, he finds out that his lesson schedule for the first semester it has been published. Giovanni decides to fill in the application with his new appointments related to lessons attendance. In fact he knows where to go, at which time and day and for which amount of time. Since he knows that these events will going to happen for 3 months, he sets them as recurrent.

Scenario 2

da rileggere scritta veloce Giovanni want to start training but he doesn't know what are the best hours in which he can run in accord to his appointments, he know only that he can run between 5 and 7 pm, for 45 minutes. he can insert this last appointment in the application whitout specify the exactly starting hour and the system will give him the best hours in which he can run

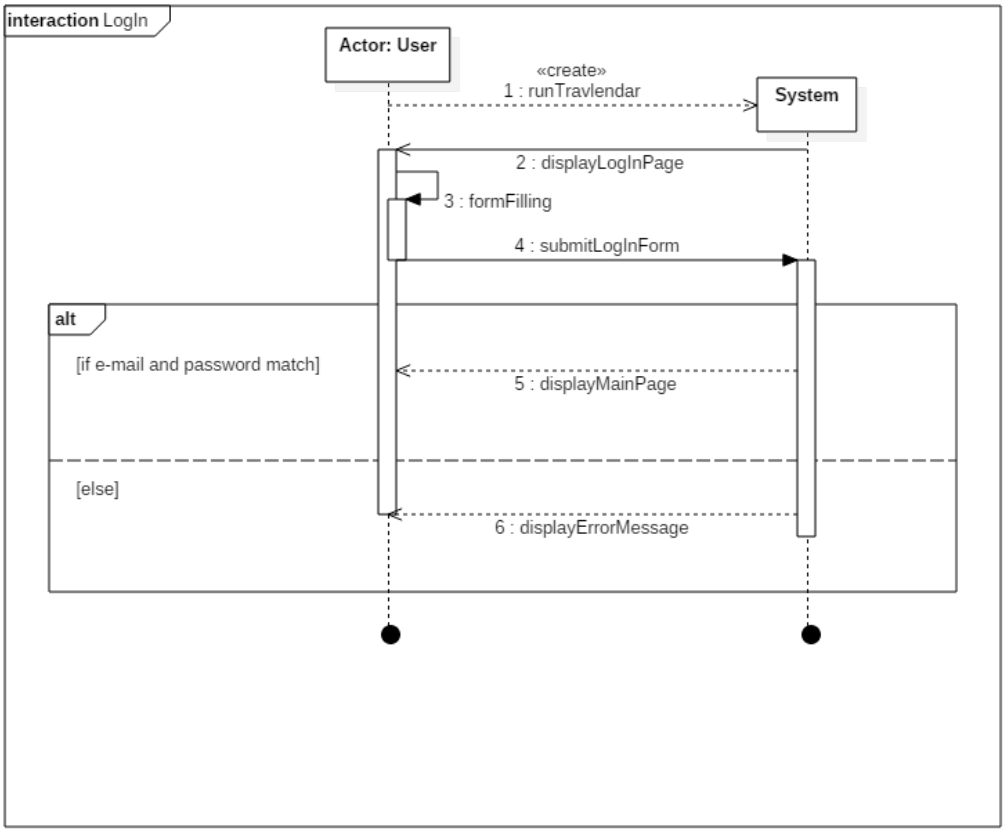
Scenario 3

da rileggere scritta veloce Giovanni has scheduled his appointments but at lunch time his son called him because he needed a ride for go back to home. Giovanni decided to help his son and so he brought him home. now the current running schedule is not more valid so he request to the system a reschedule of his appointment according to his position and the hour of day in which he is.

3.2.2 Use cases

User log-in

Name: User log-in
Actors: User
Goals: (**Goal del login**
Input Condition: The user is registered to the system
Event Flow: 1. The user needs to log-in the application, so he runs it; 2. The system provides to the user a form to fill; 3. The user fills up the form with the his e-mail and his password (as said in 2.1.1) 4. The user submits the form to the system; 5. The system checks the user identity and provides to the user the main application page (**reference to the main application page**)
Output Condition: The user is logged-in to the system.
Exceptions: The user submits the form after having filled it with a wrong email or password.
Mapping on requirements: • Events from 3 through 5 granted by (**requirement che può recuperare informazioni dell'utente riguardo i dati della registrazione**); • Event 6 grandet by (**requirement che il sistema può controllare se l'identità di un utente è giusta**;



Appointment creation

Name: Appointment creation
Actors: User
Goals: G4
Input Condition: • The user is registered to the system • The user is logged in to the systems

Event Flow:

1. The user wants to add a new appointment to his schedule;
2. The user requests the appointments page
3. The system provides the appointments page
4. The user requests the creation of a new appointment to the application;
5. The system provides to the user a form to fill;
6. The user fills up the form with the parameters (specified in 2.1.2) and constraints (specified in 2.1.4 about the new appointment;
7. The user submit the form to the system;
8. The system allocates the new appointment as Unscheduled (referring to statechart in figure;)
9. The system sends a confirmation to the user.

Output Condition: The user has created a new appointment;

Exceptions:

1. Some fields of the form referring to parameters are left blank;
2. The *location* field doesn't belong to the domain area of the application (**riferimento alla domain assumption della regione**)

Mapping on Requirements:

- Events 4 through 7 are granted by (**requirement che il sistema può recuperare informazioni riguardanti un appointment**)
- Event 7 is granted by (**requirement che il sistema è in grado di memorizzare un appointment**

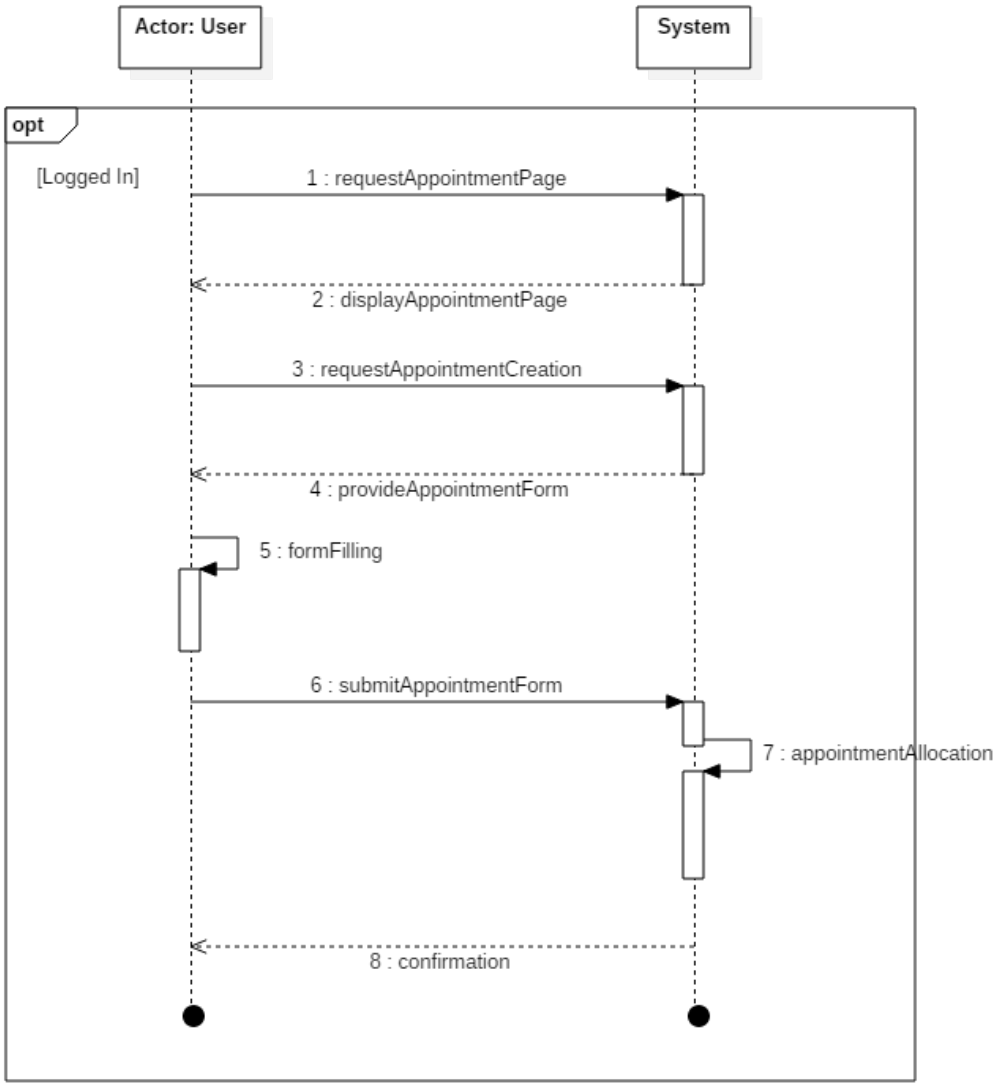


Figure 3.8: Appointment creation sequence diagram

In the sequence diagram there's the assumption that the log-in proceeds successfully. The log-in procedure referenced is the one explained in (**use case del log-in**.)

ScheduleSelection

Name: Multiple Schedules creation
Actors: User
Goals: aggiungere rif al goal

Input Condition:

- The user is registered to the system
- The user is logged in to the systems

Event Flow:

1. The user wants to compare multiple schedules;
2. The user requests the schedules page;
3. The system provides the schedules page;
4. The user selects a schedule to be run;
5. The system display the mainpage with the schedule results (fare riferimento alla definizione)

Output Condition: The user selects a schedule to be run

Exceptions:

Mapping on Requirements:

- Events are granted by the requirment R10 **(mettere riferimenti)**

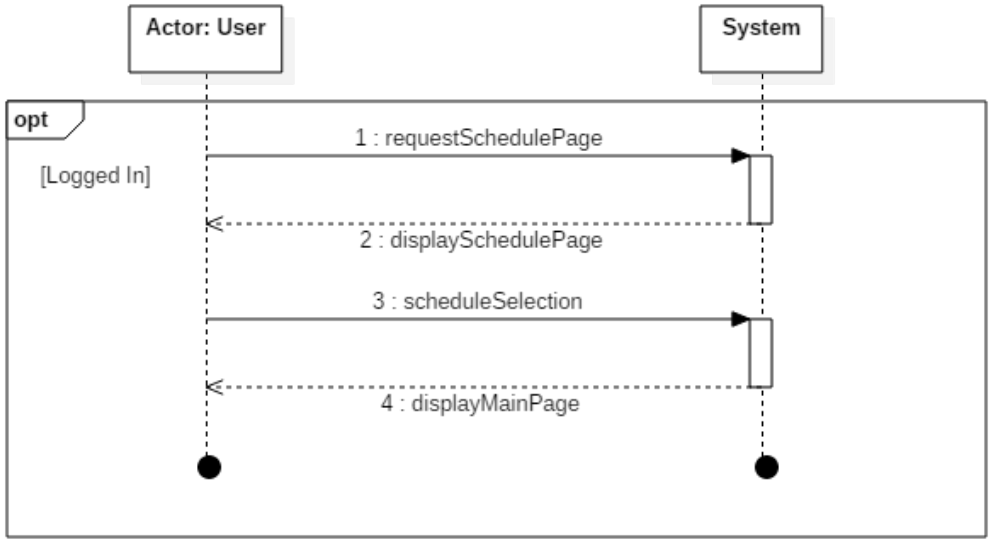


Figure 3.9: Appointment creation sequence diagram

Appointment editing

Name: Appointment editing
Actors: User
Goals: (**goal che l'utente può modicare un appointment**)
Input Condition: The user is logged-in to the system

Event Flow:

1. The user wants to modify an appointment of his schedule;
2. The user selects the appointment to modify;
3. The system provides to the user the appointment form with all the parameters and constraints (with reference to and that were specified yet by the user;
4. The user edits the fields of the form;
5. The user submit the form to the system;
6. The system set the appointment as Unscheduled with the new parameters (referring to statechart in figure ..);)
7. The system sends a confirmation to the user.

Output Condition: The user has modified an appointment;

Exceptions:

1. Some fields of the form referring to parameters are left blank;
2. The *location* field doesn't belong to the domain area of the application (**riferimento alla domain assumption della regione**)

Mapping on Requirements:

- Events 3 through 5 are granted by **(the requirement that says that the system should let the user change the parameters and the constraints of an inserted appointment)**
- Event 6 is granted by (**requirement che il sistema è in grado di memorizzare un appointment modificato**

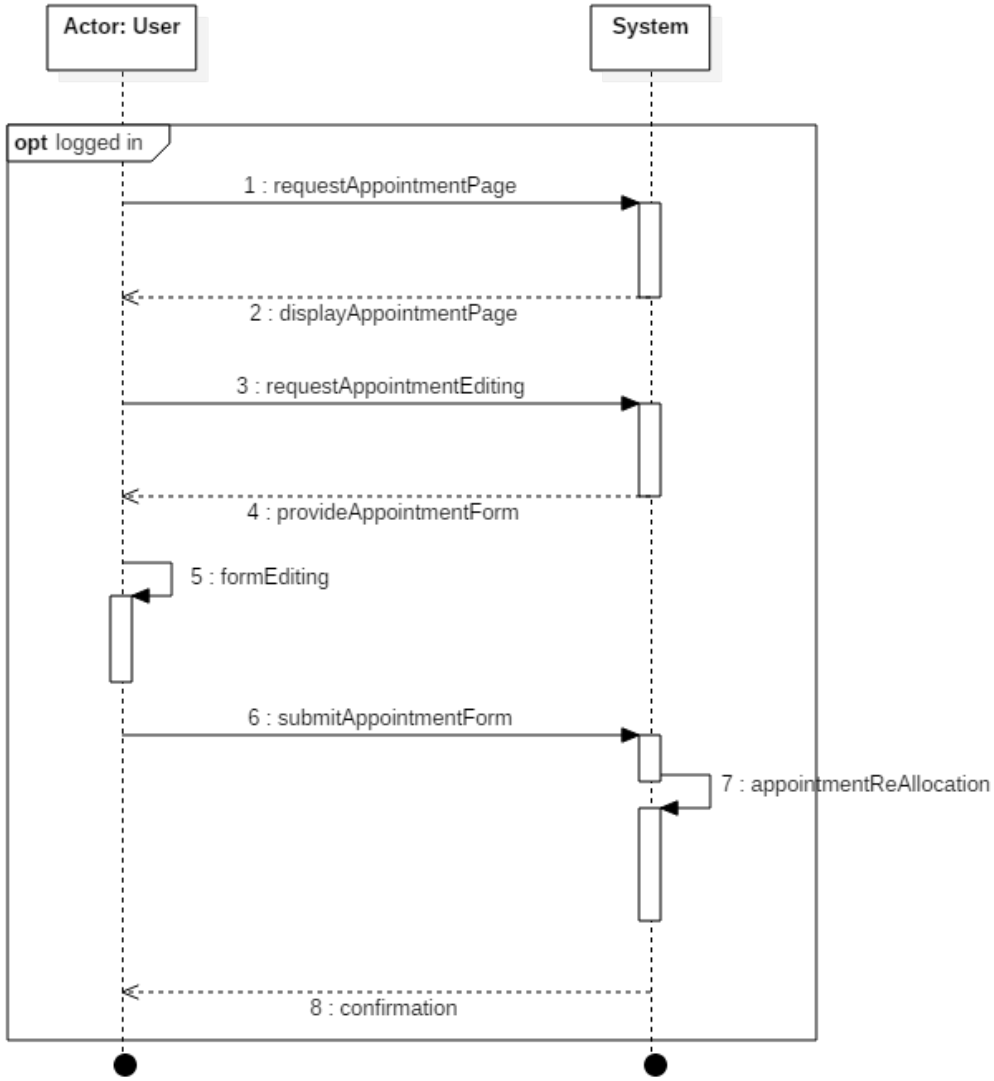


Figure 3.10: Appointment editing sequence diagram

Schedule appointments

Name: Schedule appointments
Actors: User, External API
Goals: (**goal che permette di fare uno scheduling degli appuntamenti dell'utente**)
Input Condition: The user is logged-in to the system (in futuro mettere che l'actor è un logged user e togliere questo, se ci piace di più

Event Flow:

1. The user wants to schedule his appointments;
2. The user requests a schedule by mean of selecting the appointments to schedule of the current day;
3. The system provides to the user the schedule form with all the parameters, optimization criteria (with reference to (**parametri dello schedule** and **opt criteria dello schedule**) and the constraints (**riferimento ai constraints sullo schedule**.
4. The user fills up the fields of the form;
5. The user submits the form to the system;
6. The system retrieves information from external APIs about; travel options and related travel option data, weather forecast and strike days;
7. The system retrieves about the Travel Option Data of the newly created Schedule
8. The system stores the Schedule, together with his travel option data (**link alla definizione**) and stores it as Saved (**fare riferimento allo state chart**) with the appointment selected by the user.

Output Condition: The user has created a valid schedule of his appointments;

Exceptions:

1. Some fields of the form referring to schedule variables and optimization criteria are left blank. The parameters of schedule constraints could also be left blank since they will assume default values;
2. It's not possible to list the appointments as a Valid Schedule, so the schedule is Discarded **(riferimento allo statechart)**

Mapping on Requirements:

- Events 2 through 5 are granted by **(the requirement that says that the user can set the opt criteria, constraints, variables and of the schedules)**;
- Event 6 and 7 is granted by (**the requirement that says that we can retrieve info from APIs**);
- Event 8 is granted by (**the requirement that says that is possible to store a schedule**, (**requirement che il sistema è in grado di ottimizzare in base a quanto gli dico**)

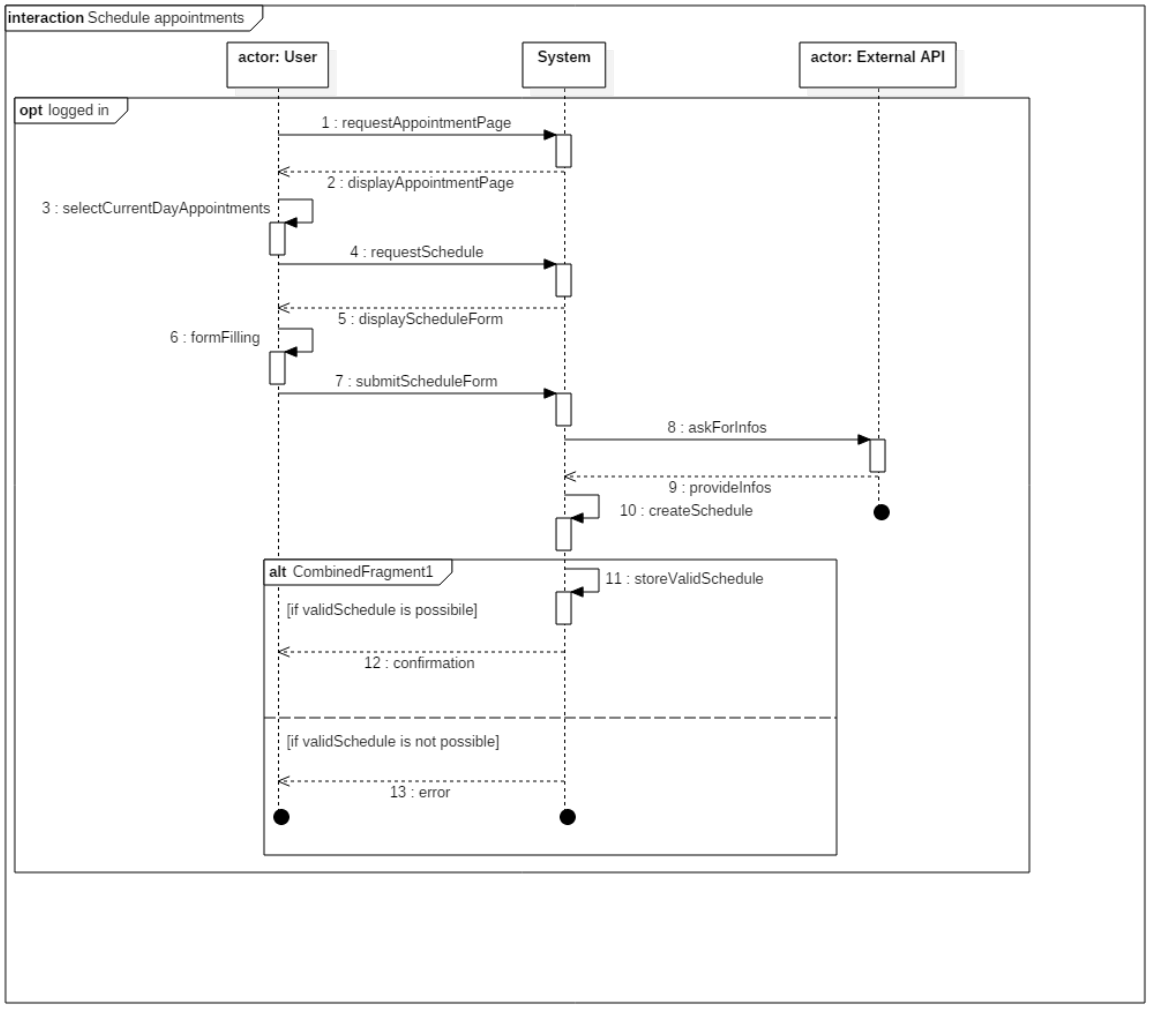


Figure 3.11: Schedule appointents sequence diagram

User registration

Name: User registration
Actors: User, external e-mail service
Goals: G1
Input Condition:

Event Flow:

1. The user wants to register to the system, so he runs the application;
2. The system display the login/registration page
3. The user fills the form (the form is present on the first page that the application display after the startup)
4. The user submit the filled form to the system
5. The system send a confirmation e-mail to the user;
6. The system display a message in which the user is informed that he will recieve a confirmation e-mail;
7. The user confirm the registration by clicking a link in the received e-mail;
8. A confirmation message is sent to the application;
9. A registration completed message is sent to the user by the system.

Output Condition: The registration is confirmed to the system;

Exceptions:

1. The e-mail given by the user is fake
2. The user makes a typo during the insertion of his e-mail

Mapping on Requirements:

- Events 1 through 3 are granted by R12;(AGGIUNGERE I RIFERIMENTI)
- Evens 4 through 6 are granted by R13. (AGGIUNGERE I RIFERIMENTI)

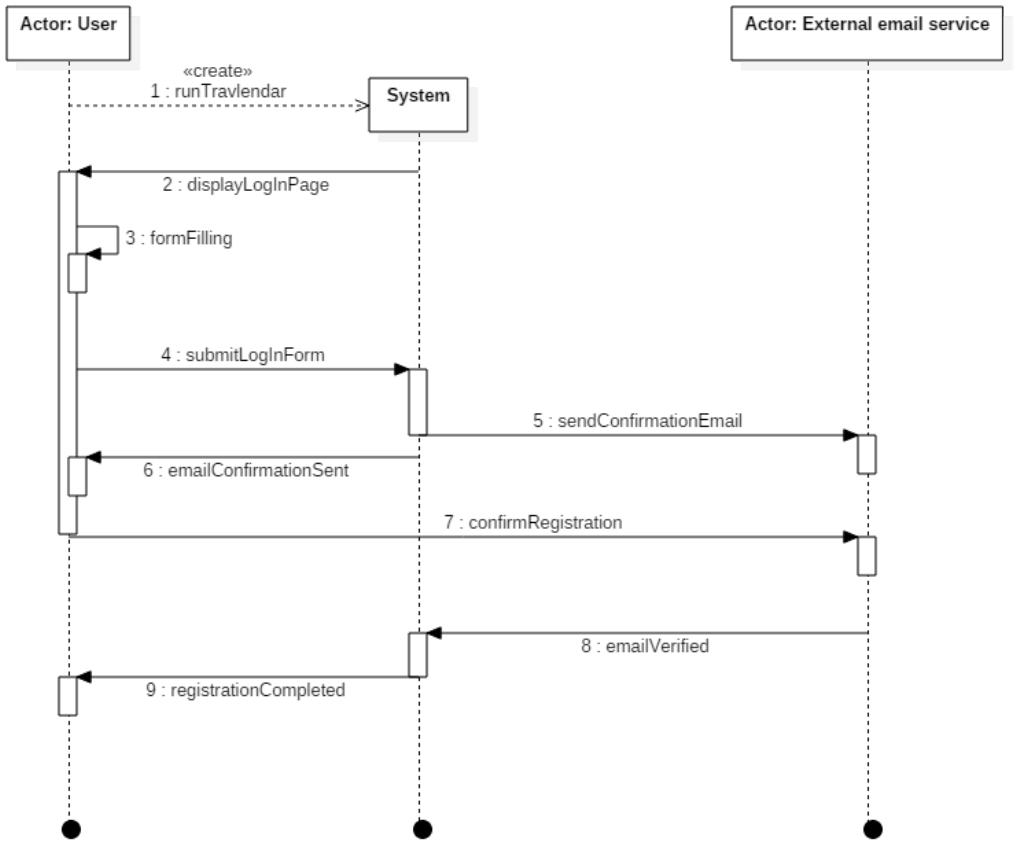


Figure 3.12: Registration sequence diagram

Booking phase

Name: Booking phase
Actors: User, external apis
Goals: FARE RIFERIMENTO AL GOAL CORRETTO

Input Condition:

- The user must be logged in to the system;
- The user must have selected a schedule to be run;
- The user must have linked to the system his external accounts;(FARE RIFERIMENTO ALLA SEZIONE DOVE SI PARLA DI QUESTA COSA)
- The user would like to buy the tickets for the travel means involved in the running schedule.

Event Flow:

1. The System, after the user have selected a schedule, asks to the user if he want to buy the ticket for the running schedule;
2. The User confirm to the system his intention;
3. The System perform a call to the travel means APIs for buying the ticket;
4. The APIs send back a confirmation message of the purchase;
5. The system send a confirmation message to the user.

Output Condition: The User recieve the confirmation message;

Exceptions:

1. The user doesn't have enough money in his card to complete the transaction;
2. there aren't free sits in one of the selected travel means;

Mapping on Requirements:

Events 3 through 5 granted by R17;(AGGIUNGERE I RIFERIMENTI)

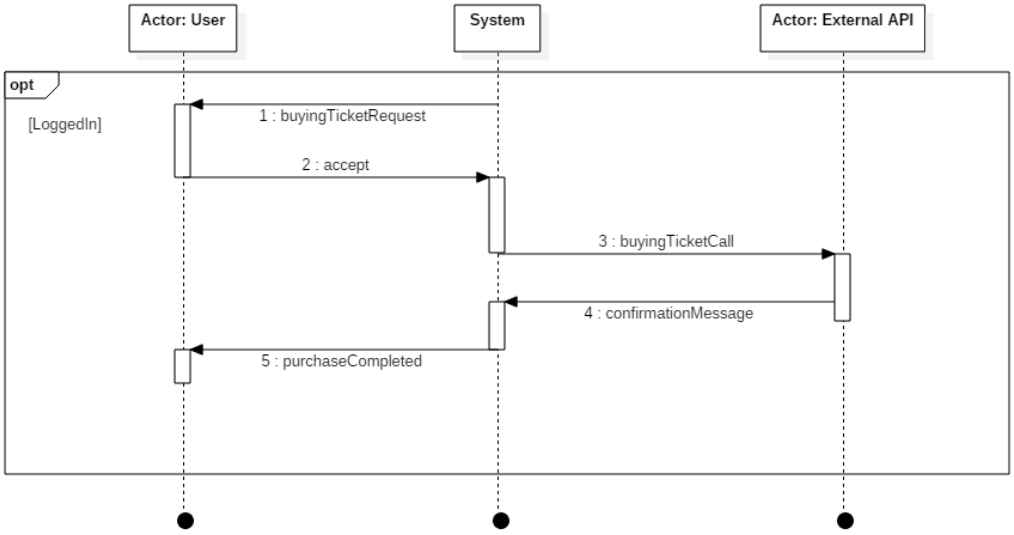


Figure 3.13: Booking phase sequence diagram

Dynamic Directions

Name: Dynamic Directions
Actors: User,external apis,GPS
Goals: FARE RIFERIMENTO AL GOAL CORRETTO
Input Condition: • The user must be logged in to the system; • The user must have a running schedule;

Event Flow:

1. The user requests the Directions for the travel to the system;
2. The system retrieves the user position from his GPS;
3. The system retrives from external APIs the directions to give to the user based on his position;
4. The system display to the user the updated map and the directions that him must follow in order to arrive to the next appointment
5. user doesn't need more directions so he closes the dynamic map.

Output Condition: The User is satisfied with the information gathered until this moment so he decides to close the dynamic map;

Mapping on Requirements:

Events 3 through 5 granted by R17;(AGGIUNGERE I RIFERIMENTI)

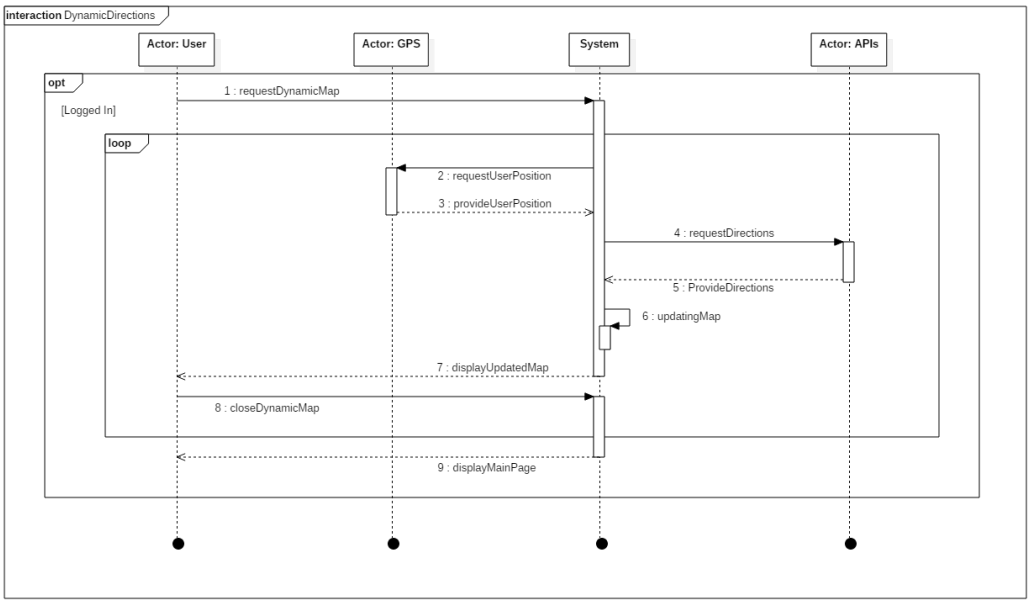


Figure 3.14: Booking phase sequence diagram

Recover credentials

Name: Schedule appointments
Actors: User, External Email Service
Goals: (**goal che permette di recuperare le credenziali dell'utente**)
(in futuro mettere che l'actor è un registered user e togliere questo, se ci piace di più
Event Flow: 1. The user wants to recover his passoword; 2. The user requests to recover his passoword; 3. The system provides to the user the schedule form with his e-mail; 4. The user fills up the field of the form; 5. The user submits the form to the system; 6. The system sends the e-mail to the user with his password.
Output Condition: The user has recovered his password;
Exceptions: 1. The form it's left blank; 2. An invalid address is given;
Mapping on Requirements: • Actions 2 through 4 granted by requirement (**the system should be able to recover email address from the user** • Action 5 granted by the other requirement

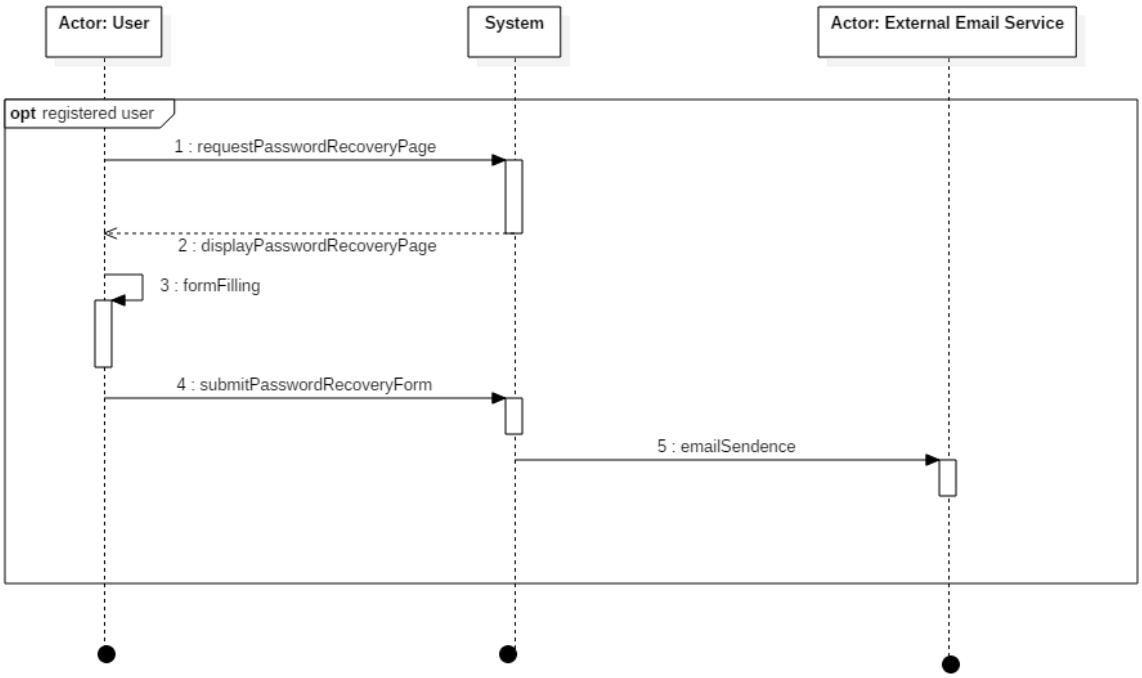


Figure 3.15: Password recovery sequence diagram

3.2.3 Definition of use case diagrams

In these diagrams we assume that the user has ran the application. In particular, if the actor is a *Registered User* or an *Enternal User* then we assume that the user is facing the login page (**riferimento alla login page**. On the other hand, if the actor is a *Logged User* then is assumed that it is on the home page (**riferimento alla home page**.