SmartCityAdvisor
Requirements Analysis and Specifications Document

Mirko Basilico, Federico Dazzan

June 23, 2016

Contents

1 Introduction **1**
- 1.1 Purpose . . . 1
- 1.2 Scope . . . 1
- 1.3 Actors . . . 2
- 1.4 Goals . . . 3
- 1.5 Definitions . . . 3
- 1.6 Stakeholders . . . 4
- 1.7 Reference Documents . . . 4
- 1.8 Document overview . . . 4

2 Overall Description **5**
- 2.1 Product perspective . . . 5
- 2.2 User characteristics . . . 5
- 2.3 Constraints . . . 5
  - 2.3.1 Regulatory policies . . . 5
  - 2.3.2 Hardware limitation . . . 5
  - 2.3.3 Documents related . . . 5
- 2.4 Assumptions . . . 5

3 Specific Requirements **8**
- 3.1 User interfaces . . . 8
  - 3.1.1 Login . . . 8
  - 3.1.2 Registration . . . 9
  - 3.1.3 Homepage . . . 10
  - 3.1.4 Find emergency room . . . 11
  - 3.1.5 Find Parking . . . 12
  - 3.1.6 Find public transport . . . 13
  - 3.1.7 Traffic limitation notification . . . 14
- 3.2 Functional requirements . . . 15
- 3.3 Scenarios . . . 17
  - 3.3.1 Scenario 1 . . . 17
  - 3.3.2 Scenario 2 . . . 17
  - 3.3.3 Scenario 3 . . . 17
  - 3.3.4 Scenario 4 . . . 18
  - 3.3.5 Scenario 5 . . . 18
  - 3.3.6 Scenario 6 . . . 18
  - 3.3.7 Scenario 7 . . . 18
- 3.4 UML Models . . . 20
  - 3.4.1 Use cases . . . 20
  - 3.4.2 Class diagram . . . 30
- 3.5 Non functional requirements . . . 30
  - 3.5.1 Performance Requirements . . . 30
  - 3.5.2 Design Constraints . . . 30
  - 3.5.3 Software System Attributes . . . 31

3.5.4 Security 31

4 Alloy **32**
4.1 Modeling 32
4.2 Analyzer 38
4.3 Worlds generated 40

5 Appendix **42**
5.1 Software and tools used 42
5.2 Hours of work 42

1 Introduction

1.1 Purpose

This document represent the Requirement Analysis and Specification Document (RASD). The main goal of this document is to completely describe the system in terms of functional and non-functional requirements, analyze the real need of the customer to model the system, show the constraints and the limits of the software and simulate the typical use cases that will occur after the development. This document is intended to all developers and programmers who have to implement the requirements, to system analysts who want to integrate other system with this one, and could be used as a contractual basis between the customer and the developers.

1.2 Scope

We will project and implement SmartCityAdvisor, which will be a software system that, using a web/mobile app, will provide the following features to the city of Milan and to its citizens:

1. If the level of CO2 is too high or in case of any special situation managed by the government of the city (e.g., the arrival of some VIP, an accident, ...), the system limits the traffic that enters in the city center by diverting it through paths that avoid the center. This is done both by controlling the traffic lights and by alerting the citizens through the app and through some large displays that are installed at the main entrances of the city.

2. When a citizen signals through the app that he/she has to go to an emergency room, provide suggestions on which hospital to choose based on: i) the condition of the citizen and the specializations available in the hospitals, ii) the status of queues in the various emergency rooms, iii) the location of the citizen and iv) the situation of traffic.

3. When a citizen wants to go to the city center by car, he/she can signal through the app the address of the place he/she wants to go, and the system will return the address of the closest available parking with corresponding availability.

4. When a citizen wants to reach a place in Milan using the public transport system, he/she can signal through the app the starting and the arrival address and the kind of public transport he/she wants to use (underground, tram or bus). After the citizen has put this data into the app, the system will show: i) name and position of the closest station where to get the desired public transport, ii) name and position of the arrival station, iii) the time when the public transport will arrive at the starting station, iv) the estimated time it will take to arrive to destination.

To provide this features, SmartCityAdvisor will use information provided by the Data Control Center of the city, that collects data from various sensors that have been installed in the territory. This sensors acquire data about:

- level of CO2 in the air;
- number of cars that enter in the city center;
- availability of parking places in all areas of the city center.

Besides the sensors, the city has also installed proper actuators that control the traffic lights at the main intersections in the city. The actuators are controlled by the Traffic Control Center of the city, which also deals with the management of the displays installed at the main entrances of the city.

Moreover, the city has established agreements with the main hospitals in the territory to know in real-time the length of the queues in their emergency rooms, and with ATM (the public transport company) to know in real-time the position and current schedule of all public transports in the city. Both of them send their data to the Data Control Center of the city, which will provide the data to our system.

1.3 Actors

The actors of our system are:

Citizen: a citizen of Milan who hasn't signed up into the app yet. This kind of user can only see the app's login page and access to the registration form.

Registered citizen: a citizen of Milan who has signed up into the app. This kind of user can also use all the services provided by our system through the app.

Data Control Center (DCC): an hardware/software system that collects some data useful for the city. This will be the actor that provides the data to our system, as indicated at the previous point of this document. The communications among DCC and our system occur automatically, without human participation, using a specific private API.

Traffic Control Center (TCC): an hardware/software system that, through proper actuators, controls the traffic light at the main intersection in the city and the displays that shows information about the situation of the traffic in the center. It can be operated manually or by an automatic system. The communications among TCC and our system occur automatically, without human participation, using a specific private API. This API allows our system to require to TCC some information that its system can obtain, e.g. the time to get from one point to another in Milan.

Government of Milan: in case of any special situation in the center (e.g., the arrival of some VIP, an accident, ...), an employee of the government will indicate to the system that the access to the city center must be limited, using a web application accessible from any browser.

1.4 Goals

List of the goals of SmartCityAdvisor:

- [G1] Citizen should be able to register to the service;
- [G2] Citizen should be able to login to the service;
- [G3] If the level of CO2 in the air is over a specific threshold, access to the center should not be allowed to cars and the citizens should be alerted of this (by app and by displays installed at the main entrances of the city);
- [G4] In case of any special situation in the center (e.g., the arrival of some VIP, an accident, ...), and if the government of Milan think it is necessary, access to the center should not be allowed to cars and the citizens should be alerted of this (by app and by displays installed at the main entrances of the city);
- [G5] A registered citizen with a specific health problem, should receive suggestions on which hospital to go to receive assistance for his condition in the shortest possible time (considering the time to go to the hospital and the time to wait in queue);
- [G6] A registered citizen that want to go to a specific address in the center by car, should receive information about the closest to this address (as the crow flies) available parking and the corresponding availability;
- [G7] A registered citizen that want to go to a specific address in Milan, using a specific public transport (underground, tram or bus), should receive information about the closest (as the crow flies) station where to get the selected public transport, the arrival station, the time when the public transport will arrive at the starting station, and the estimated time it will take to arrive to destination.

1.5 Definitions

RASD: Requirement Analysis and Specification Document (this document).

DCC: Data Control Center.

TCC: Traffic Control Center.

System: The system that we have to implement (SmartCityAdvisor).

City: The city of Milan.

User: A citizen that uses our system.

1.6 Stakeholders

Our principal stakeholder is the government of the city, the institution that commissioned us to develop this software. It expects to receive a working software that fulfills the requests, the documentation about requirements analysis, design, development, testing, project reporting and a final presentation. The purpose of this stakeholder is to provide some features to its citizen with the aim of improve their quality of life.

Other secondary stakeholders are:

- the citizens of the Milan;
- the public transport company (ATM);
- the main hospitals of Milan.

1.7 Reference Documents

- New Project April 2016.pdf
- IEEE Std 8301998 IEEE Recommended Practice for Software Requirements Specifications.

1.8 Document overview

This document is essentially structured in three parts:

- Section 1: Introduction, it gives a description of document and some basic information about software.
- Section 2: Overall Description, gives general information about the software product with more focus about constraints and assumptions.
- Section 3: Specific Requirements, this part list requirements, typical scenarios and use cases. To give an easy way to understand all functionality of this software, this section is filled with UML diagrams.

2 Overall Description

2.1 Product perspective

SmartCityAdvisor will be interfaced with Data Control Center's and Traffic Control Center's system through the provided API.

Moreover our system will be interfaced with his users through a web and a mobile application.

2.2 User characteristics

The user that we expect to use our applications is a person who want an easy way to access to the service we offer. The only ability that they need are to be able to use a web browser and to download a mobile application.

2.3 Constraints

2.3.1 Regulatory policies

SmartCityAdvisor has to comply with all the applicable privacy laws to store the sensible user data.

2.3.2 Hardware limitation

The only hardware requirements that our product needs to be used by a user are a device with a web browser, to access the web application, or smartphone running iOS/Android/WinPhone to access the mobile app.

2.3.3 Documents related

- Requirements and Analysis Specification Document (RASD).
- Design Document (DD).
- User's Manual.
- Testing report.

2.4 Assumptions

We assume that:

- All citizens own a valid email address.
- Two different citizens can't have the same email address.
- It's defined, by a specific organization, a threshold of the level of CO2 admitted in the city of Milan.

- If a special event occurs in the center of Milan (e.g., the arrival of some VIP, an accident, ...), it will be notified at the government of the city.
- TCC is able to limit the traffic that enters in the city center by diverting it through paths that avoid the center. This is done by controlling the traffic lights using proper actuators.
- When TCC receives a request to limit the traffic in the city center by our system, it will limit the traffic within 5 minutes.
- TCC is able to write message that will be shown on the display at the main entrances of the city.
- When the TCC limits the traffic that enters in the city center, it will also write a message on the displays to inform the citizens of this.
- Our system knows a list of possible generic health problems.
- Our system knows, for each hospital that has established agreements with the city of Milan, the specializations available in the hospital, selected from the list of possible health problems, and the coordinates of its location.
- There is at least one hospital for each possible health problem in the list.
- The health problem of a user who is using our system, is always one which is included in the list of possible health problems.
- Each hospital has an unique queue for every possible problem.
- All devices (PC and mobile) are provided with a GPS.
- Our system is able, given the coordinates of two points, to calculate the distance between this points (as the crow flies).
- The coordinates of the current position of each user are known through GPS.
- TCC is able, given a specific address in Milan, to calculate the coordinate of this address.
- TCC is able, given the coordinates of two points in Milan, to calculate an estimated time to go from the first to the second point by car, considering the situation of the traffic at the moment.
- DCC knows in real-time the length of the queues in the emergency rooms, with an associated estimated time to wait, of all the hospitals that has established agreements with the city of Milan.
- Our system knows all the parking areas in the center of Milan with their coordinates.

- DCC knows in real-time the availability of all the parking areas in the city center.
- There are only three kind of public transports in Milan: underground, tram and bus.
- Our system knows the coordinates and the address of all the station of Milan's public transports.
- DCC knows in real-time the position and current schedule of all public transports in the city.
- DCC and TCC provide to our system two specific private API that allow our system to have access to all the necessary services.

3 Specific Requirements

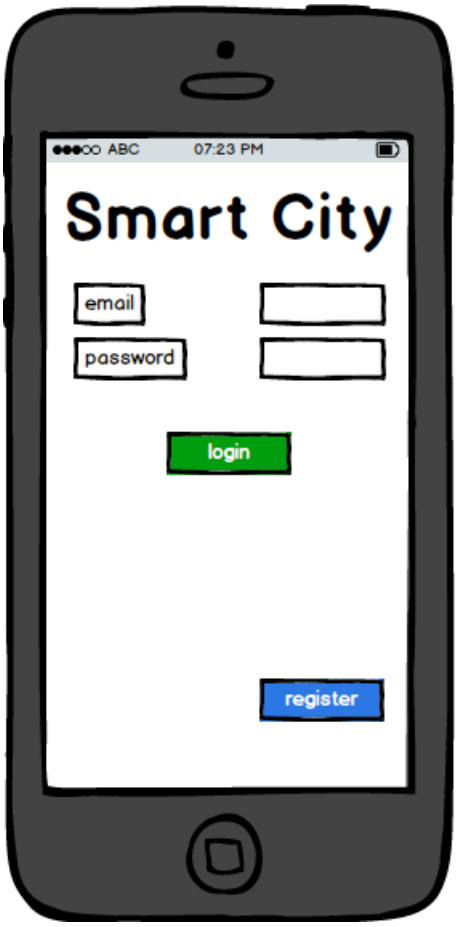
3.1 User interfaces

Here are presented some mock-up that represent an idea of the structure of the mobile application screen.

The screen that will be shown when a user use a web browser are similar to those for the mobile application.

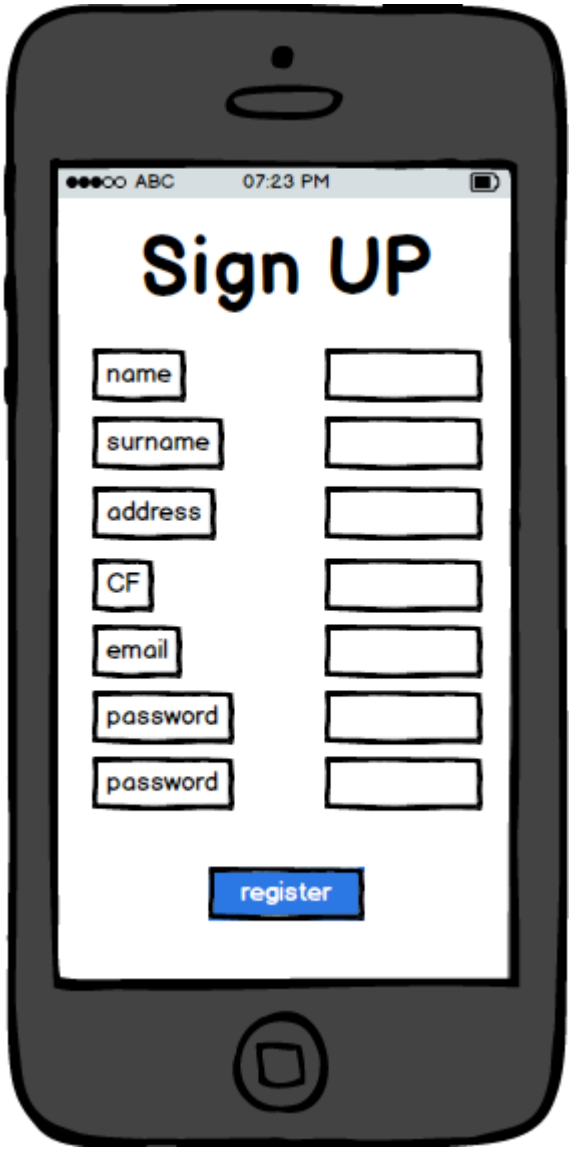
3.1.1 Login

The mock-up shows the first screen of SmartCityAdvisor. Here users can log in to the service and guests can access the registration form.



3.1.2 Registration

The mock-up shows the registration screen of SmartCityAdvisor. Here users can register to the service with their personal data.



3.1.3 Homepage

The mock-up shows the main page of the application. Here a user can choose one of the three features provided by the app.

3.1.4 Find emergency room

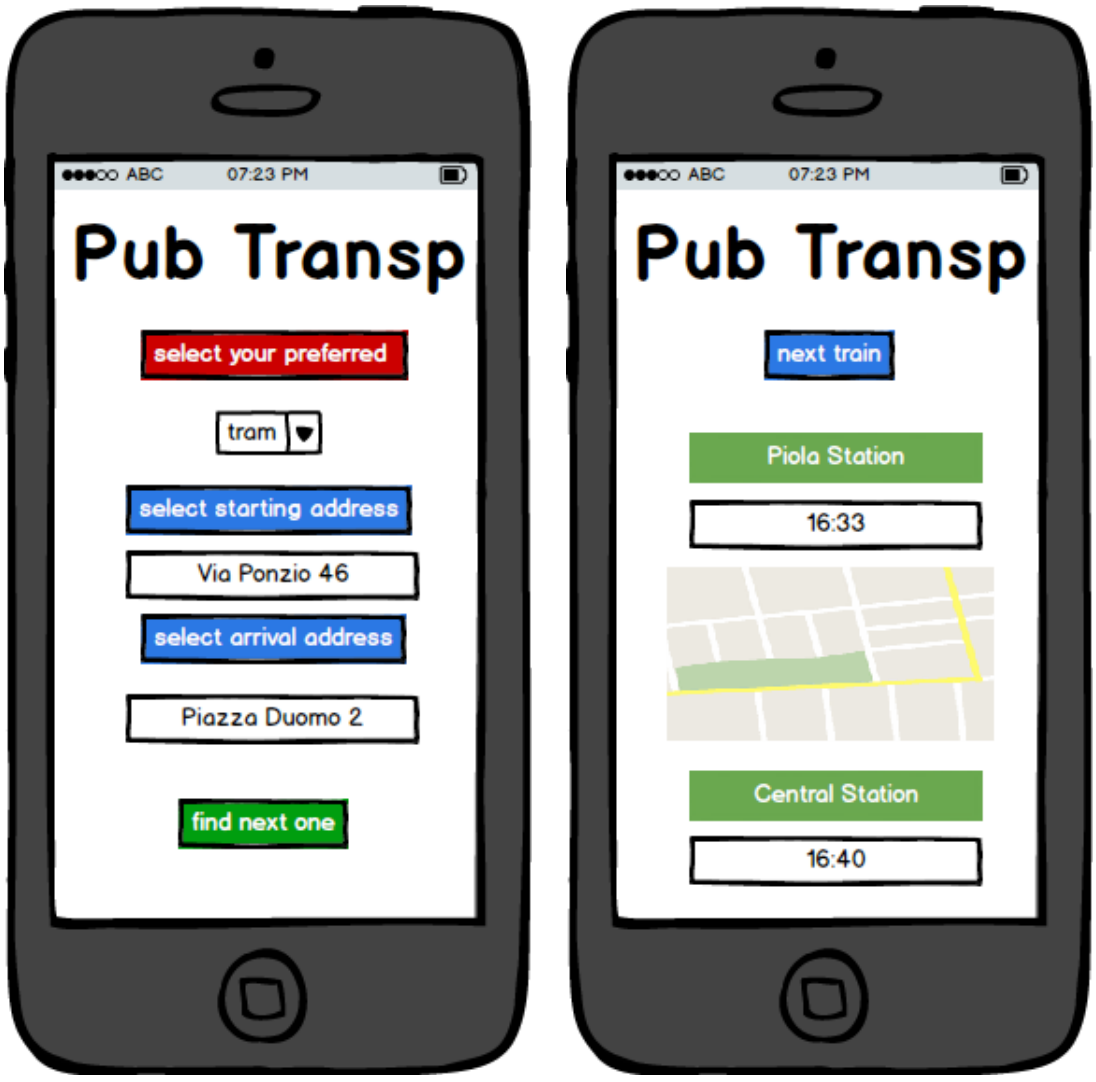
These are the mock-ups related to the "find emergency room" service. The first is the screen where a user can select his health problem and his position, and asks to the system which are the best hospital to go. In the second screen the system shows the information about the best hospital found.

3.1.5 Find Parking

These are the mock-ups related to the "find parking" service. The first is the screen where a user can select the position in the city center he/she wants to go, and asks to the system which are the closest available parking. In the second screen the system shows the information about the found parking.

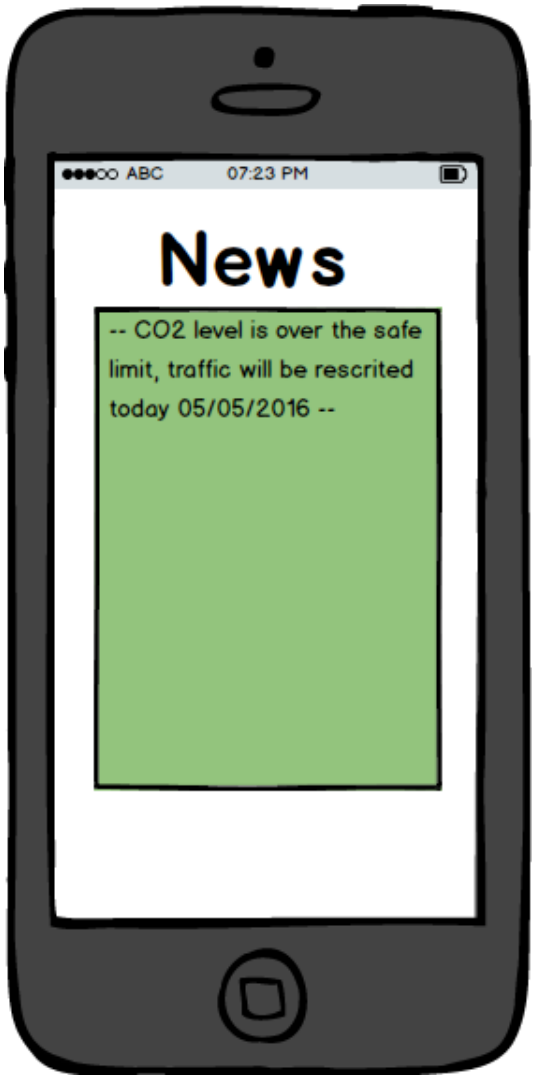
3.1.6 Find public transport

These are the mock-ups related to the "find public transport" service. The first is the screen where a user can select the starting and the arrival positions, the public transport he/she want to use. In the second screen the system shows the information about the the first train/tram/bus that will arrive in the selected starting station.



3.1.7 Traffic limitation notification

This mock-up shows the screen that is displayed when the system notifies to all the registered citizens that the traffic in the city center is closed.



3.2 Functional requirements

Here the functional requirements identified for each goal:

- [G1] Citizen should be able to register to the service:
  - The system must allow all citizen to compile the registration form, entering name, surname, address, CF, email and password.
- [G2] Citizen should be able to login to the service:
  - The system has to provide a login function, with the possibility of recovering a forgotten password.
- [G3] If the level of CO2 in the air is over a specific threshold, access to the center should not be allowed to cars, and the citizens should be alerted of this (by app and by displays installed at the main entrances of the city);
  - When DCC reveals that the level of CO2 in the air of Milan is over the indicated threshold, it will advise our system through a specific signal.
  - When the system receives the signal from DCC, it must advise TCC to limit the traffic in the city center.
  - TCC must confirm to the system that the traffic in the center is going to be limited.
  - After TCC has confirmed to the system that the traffic in the center is going to be limited, the system must advice all the registered citizens of that, through an app notification.
- [G4] In case of any special situation in the center (e.g., the arrival of some VIP, an accident, ...), and if the government of Milan think it is necessary, access to the center should not be allowed to cars and the citizens should be alerted of this (by app and by displays installed at the main entrances of the city):
  - When the government of Milan knows that a special situation is going to occur in the center, it will advise our system of that.
  - After the system has received the information from the government, it must advise TCC to limit the traffic in the city center.
  - TCC must confirm to the system that the traffic in the center is going to be limited.
  - After TCC has confirmed to the system that the traffic in the center is going to be limited, the system must advice all the registered citizens of that, through an app notification.

- [G5] A registered citizen with a specific health problem, should receive suggestions on which hospital to go to receive assistance for his condition in the shortest possible time (considering the time to go to the hospital and the time to wait in queue):
  - The system must allow a registered citizen to indicate his specific health problem and his address.
  - The system must allow the user to indicate his position also using GPS.
  - The system must provide to TCC the coordinates of each hospital that is specialized for the indicated health problem, and the coordinates or the address where the user is. TCC must return to the system an estimated arrival time for each hospital that has been communicated.
  - The system must provide to DCC the list of the hospitals that are specialized for the indicated health problem. DCC must return to the system an estimated time to wait in the queue for each hospital that has been communicated.
  - After the system has found the hospitals that can assist the user in the shortest possible time (considering the time to go to the hospital and the time to wait in queue), among those that are specialized for the indicated health problem, it must provide to the user information about this hospital.
- [G6] A registered citizen that want to go to a specific address in the center by car, should receive information about the available parking closest to the given address (as the crow flies) and the corresponding availability:
  - The system must allow a registered citizen to provide an address in the city center where he/she want to go.
  - The system must allow the user to indicate the address also using GPS.
  - If the user inserts a textual address, the system must communicate this address to TCC. TCC must return to the system the coordinates associated to the address.
  - The system must communicate to DCC that it needs information about the availability of all the parking areas in the center, and DCC must provide this information.
  - The system must provide to the user information about the available parking area closest to the given address and the corresponding availability.
- [G7] A registered citizen that want to go to a specific address in Milan, using a specific public transport (underground, tram or bus), should receive information about the closest (as the crow flies) station where to get

the selected public transport, the arrival station, the time when the public transport will arrive at the starting station, and the estimated time it will take to arrive to destination:

- The system must allow a registered citizen to indicate the starting and the arrival address of the trip, and the public transport he/she want to use.
- The system must allow the user to indicate the addresses also using GPS.
- If the user inserts one or two textual addresses, the system must communicate this addresses to TCC. TCC must return to the system the coordinates associated to them.
- The system must provide to DCC the two stations, of the selected public transport, closest to the starting and to the arrival addresses and the selected public transport. DCC must return to the system the schedule of the first train/tram/bus that will pass through the starting station and that is directed toward the arrival station.
- The system must provide to the user information about the closest (as the crow flies) station where to get the selected public transport, the arrival station, the time when the public transport will arrive at the starting station, and the estimated time it will take to arrive to destination.

3.3 Scenarios

3.3.1 Scenario 1

Mario just saw an ad on the local newspaper about this new SmartCityApp, he pulls out his phone and downloads it. He then opens the app, compiles the registration form and the system provides him with a confirmation.

3.3.2 Scenario 2

Luigi just registered to our system, now he is trying to login to see what he can actually do with the app. So he enters the username and password to login. The app forwards this info to our system that sends back a confirmation, so the app display the home screen of SmartcityAdvisor.

3.3.3 Scenario 3

Caterina is a tourist enjoying a vacation in Milan, she finds herself in a medical emergency, she was walking around the city and accidentally fell down braking her left arm... She immediately pulls off the phone and opens the SmartCityAdvisor app. She is already logged in, so she selects the "find emergency room" option. In the next screen she selects her condition from the drop down menu, she then confirms her position obtained through GPS and finally she taps

on "find the best hospital". The system sends to TCC the coordinates of all the hospitals that satisfies the criteria and obtain an ETA for each hospital. It also queries the DCC to obtain the queue time of the same hospitals. After a short wait the system elaborates the data coming from the TCC/DCC and provides the name and address of the hospital best suited for Caterina.

3.3.4 Scenario 4

Claudio is a student at PoliMI, he just decided he wants to go shopping for shoes in the city center, after a quick search on the internet he found a couple of places that are selling some cool looking shoes. Currently there is a strike of the public transport system so he decides to use his personal car, he opens the SmartCityAdvisor app and selects the "find parking" option. He now enters the destination address and selects the "found nearest parking" option, the system sends the address to the TCC and gets the coordinates in return. The system requires information about the availability of the parking areas in the city center to the DCC and gets back the required data. After a short wait the app displays the nearest parking and the number of available spots.

3.3.5 Scenario 5

Brenno is a sales clerk working in a big electronics chain, he commutes everyday using the underground, last night he slept over at his new girlfriend place, he just woke up and he wants to know how to reach his working place, he opens the SmartCityAdvisor app and he selects the "find public transport" option. He inputs both the starting and arrival addresses, he then selects the underground as the preferred option and taps on "take me there". The system queries the TCC to get the addresses respective coordinates. Then the system calculates the two underground stations closest respectively to the starting and to the arrival coordinates and sends those to the DCC, including the preferred public transport. The DCC returns the schedule of the next train. The app provides Brenno with a map, displaying the closest underground station, the time of the next train and the estimated time of arrival.

3.3.6 Scenario 6

It hasn't been raining for a while in Milan and the CO2 level has soared over the safe threshold set by the city government. The DCC has automatically sent this info to our system. The system in accordance to the internal rules instructs the TCC to limit the traffic. The TCC than confirms the traffic limitation to our system, as soon as this confirmation arrives, the system sends a notification to all the registered users.

3.3.7 Scenario 7

The first lady of the USA is visiting Milan, the city government has decided to limit traffic in the city center so it instructs our system to do exactly that. Our

system then proceeds to contact the TCC and get confirmation of the successful limiting. As soon as the confirmation is received by the system, a notification is sent to all the registered users.

3.4 UML Models

3.4.1 Use cases

Here are shown the system's use cases with the relative sequence diagrams.

User registration

Name	User registration
Actors	Citizen
Entry condition	Citizen isn't registered to SmartCityAdvisor yet.
Flow of events	1. the citizen opens the web application or the mobile app of SmartCityAdvisor; 2. the citizen clicks on the registration button; 3. the citizen inserts into the registration the request data and a password; 4. the citizen confirms the registration; 5. the system confirms the successful registration.
Exit condition	Registration successfully done
Exceptions	● The citizen is already registered; ● Email inserted is already associated to another user; ● One or more mandatory fields are not valid; All this exception are handled alerting the visitor of the problem and going back to point 2 of Event Flow.

User login

Name	User login
Actors	Registered citizen
Entry condition	Registered citizen is already registered to the system but he isn't logged in yet.
Flow of events	1. the citizen opens the web application or the mobile app of SmartCityAdvisor; 2. the registered citizen inserts his email address and password in the appropriate fields; 3. the registered citizen clicks on "login" button; 4. the system confirms the successful login.
Exit condition	The system shows the main page of the web application/mobile app.
Exceptions	If username and/or password are incorrect the system notifies it to registered passenger and doesn't allow the login.

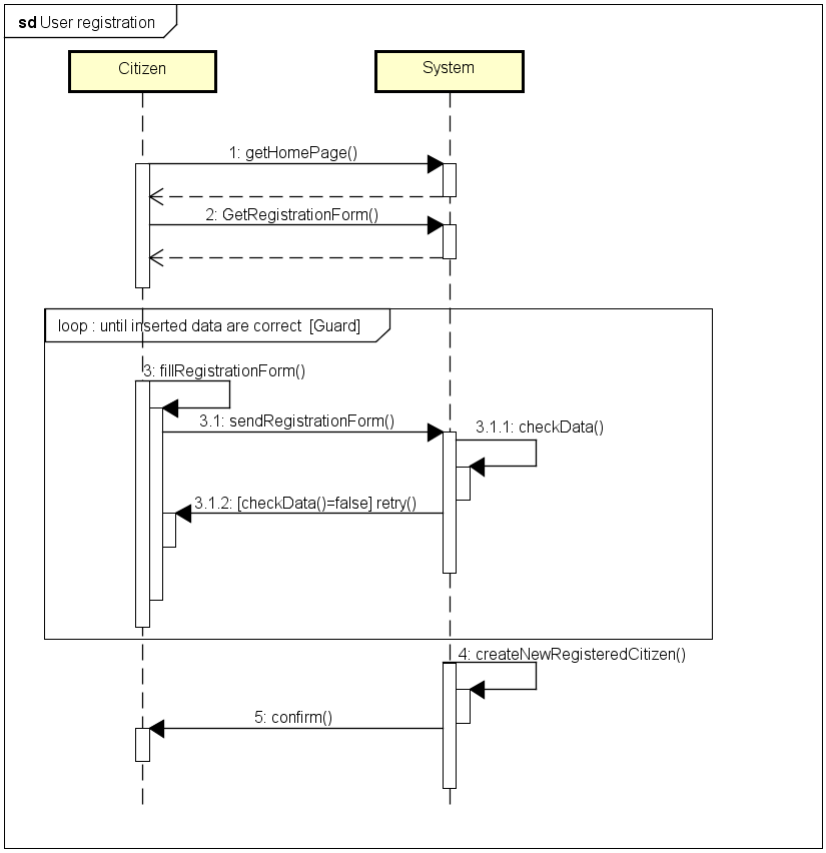


Figure 1: User registration sequence diagram

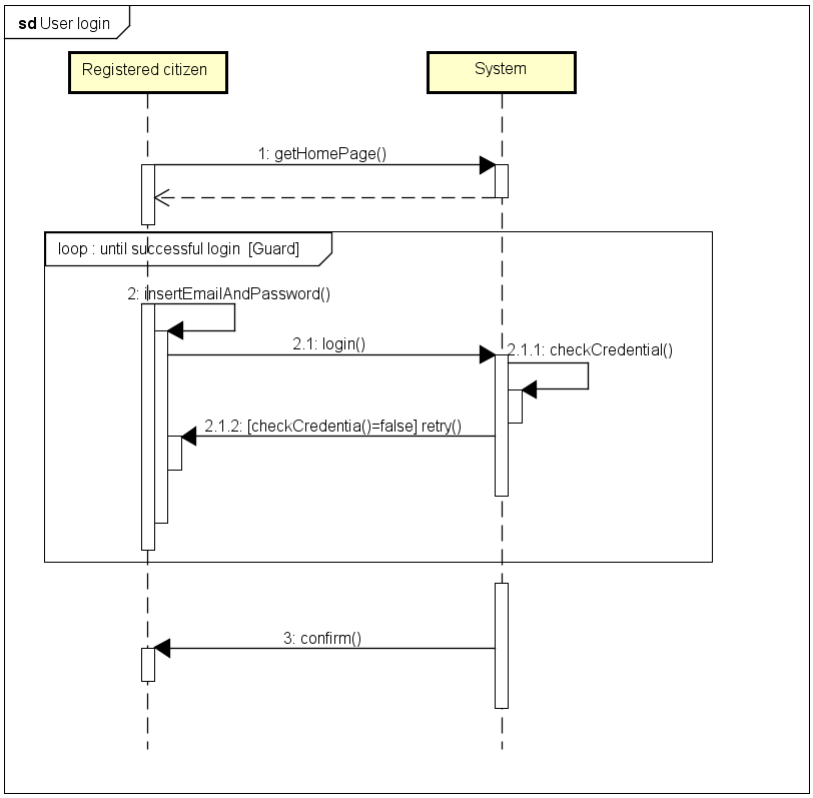


Figure 2: User login sequence diagram

Traffic limitation

Name	Traffic limitation
Actors	● Data control center (DCC) ● Government of Milan ● Traffic control center (TCC) ● Registered citizen
Entry condition	DCC or the government of Milan notifies the system that the traffic in the city center must be limited.
Flow of events	1. The system tells TCC to limit the traffic in the city center; 2. TCC confirms to the system that the traffic in the center is going to be limited; 3. the system sends a notification of traffic limitation to all registered citizens.
Exit condition	The traffic in the city center is closed.
Exceptions	

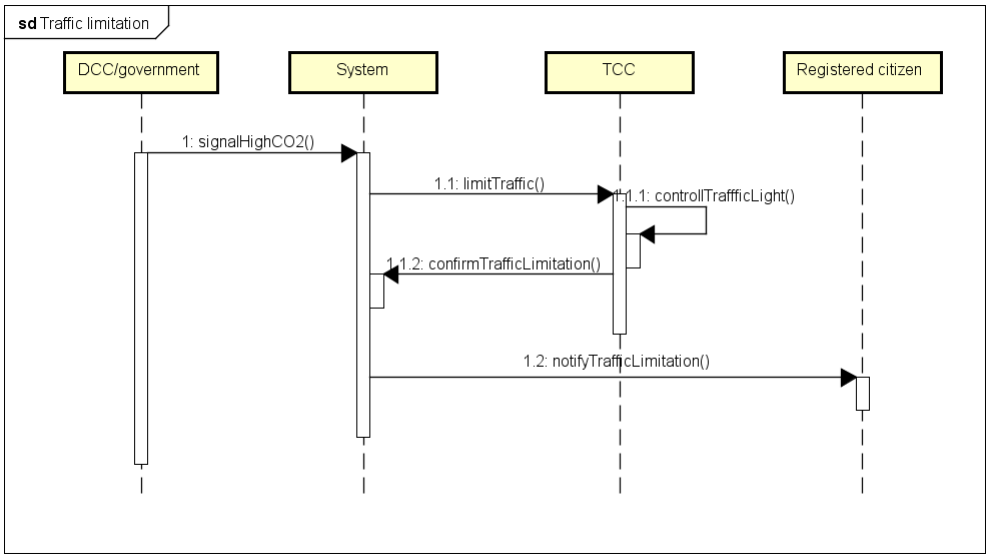


Figure 3: Traffic limitation sequence diagram

Find emergency room

Name	Find emergency room
Actors	● Data control center (DCC) ● Traffic control center (TCC) ● Registered citizen
Entry condition	A registered citizen selects the "find emergency room" option in the main menu of the web/mobile application.
Flow of events	1. The registered citizen inserts his position manually or selecting his current GPS position; 2. The registered citizen selects an health problem; 3. The system provides to TCC the coordinates of each hospital that is specialized for the indicated health problem; 4. the system provides to TCC the coordinates or the address where the user is; 5. TCC returns to the system an ETA for each hospital that has been communicated; 6. The system provides to DCC the list of the hospitals that are specialized for the indicated health problem; 7. DCC returns to the system an estimated time to wait in the queue for each hospital that has been communicated;
Exit condition	The system provides to the user information about the best hospital found.
Exceptions	The inserted address is not valid or is not contained in the city. In this case TCC notifies that to the system and it goes back to point 1 of Event Flow.

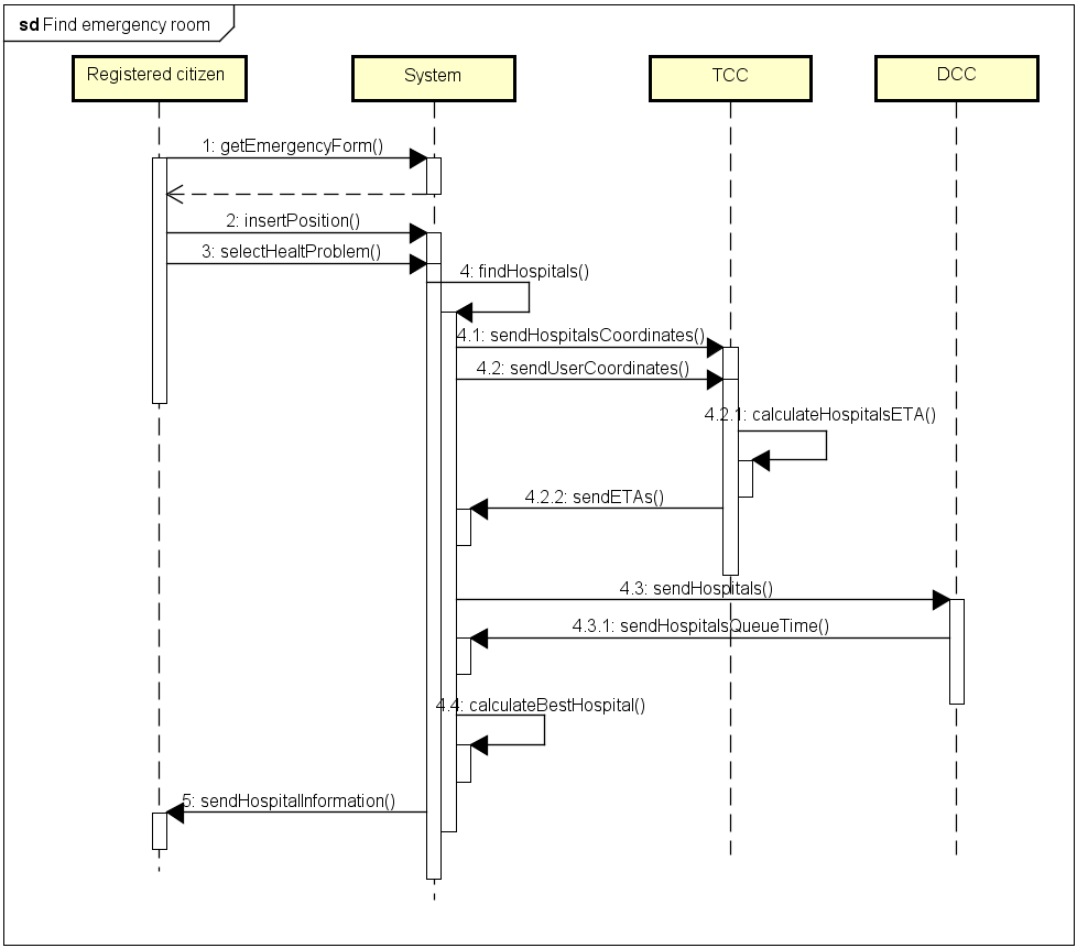


Figure 4: Find emergency room sequence diagram

Find parking area

Name	Find parking area
Actors	● Data control center (DCC) ● Traffic control center (TCC) ● Registered citizen
Entry condition	A registered citizen selects the "find parking" option in the main menu of the web/mobile application.
Flow of events	1. The registered citizen inserts the position he/she wants to go in the city center, manually or selecting his current GPS position; 2. If the user inserts a textual address, the system communicates this address to TCC; 3. Depending on previous point, TCC returns to the system the coordinates associated to the indicated address; 4. The system communicates to DCC that it needs information about the availability of all the parking areas in the center; 5. DCC provides to the system the required information;
Exit condition	The system provides to the user information about the available parking area closest to the given address and the corresponding availability.
Exceptions	The inserted address is not valid or is not contained in the city center. In this case TCC notifies that to the system and it goes back to point 1 of Event Flow.

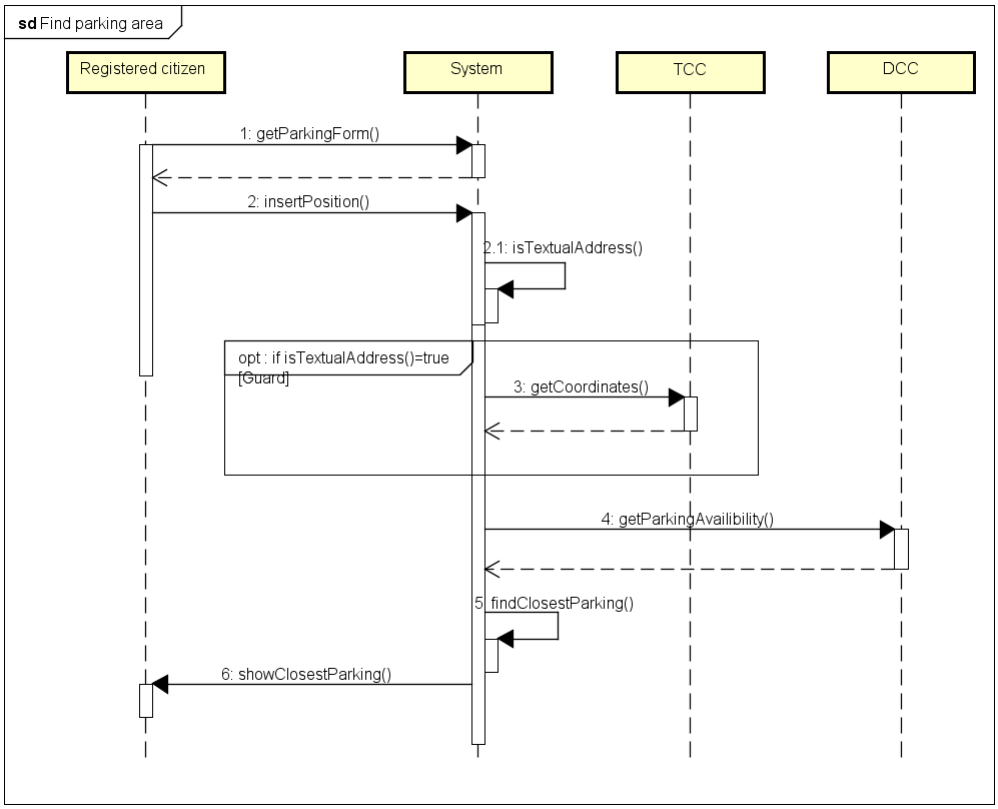


Figure 5: Find parking area sequence diagram

Find public transport

Name	Find public transport
Actors	● Data control center (DCC) ● Traffic control center (TCC) ● Registered citizen
Entry condition	A registered citizen selects the "find public transport" option in the main menu of the web/mobile application.
Flow of events	1. The registered citizen inserts the position he/she wants to start and the which one he/she wants to arrive, manually or selecting his current GPS position; 2, The registered citizen selects the public transport he/she want to use; 3. If the user inserts one or two textual addresses, the system communicates this addresses to TCC; 4. Depending on previous point, TCC returns to the system the coordinates associated to the indicated addresses; 5. The system provides to DCC the two stations, of the selected public transport, closest to the starting and to the arrival addresses; 6. The system communicates to DCC the public transport that the user has selected; 7. DCC returns to the system the schedule of the first selected public transport that will pass through the starting station and that is directed toward the arrival station;
Exit condition	The system provides to the user information about: the closest (as the crow flies) station where to get the selected public transport, the arrival station, the time when the public transport will arrive at the starting station, and the estimated time it will take to arrive to destination.
Exceptions	One of the inserted addresses is not valid or is not contained in the city . In this case TCC notifies that to the system and it goes back to point 1 of Event Flow.

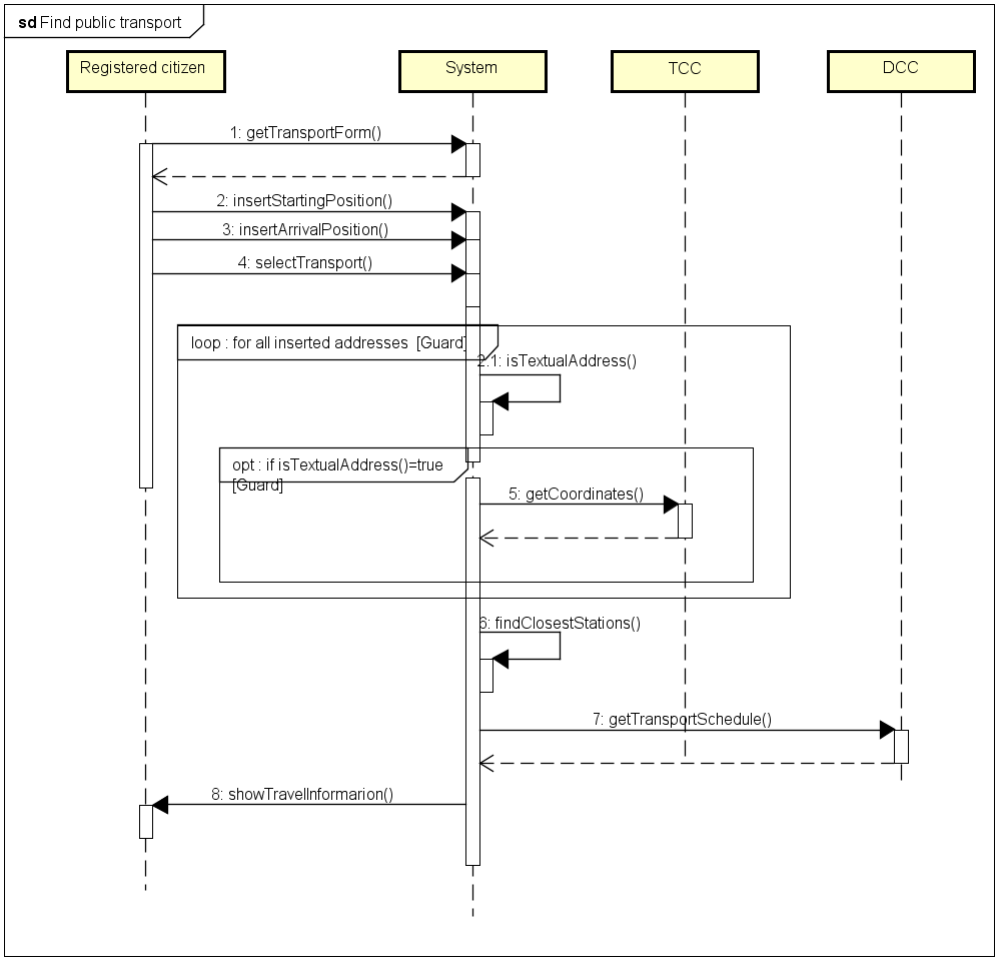


Figure 6: Find public transport sequence diagram

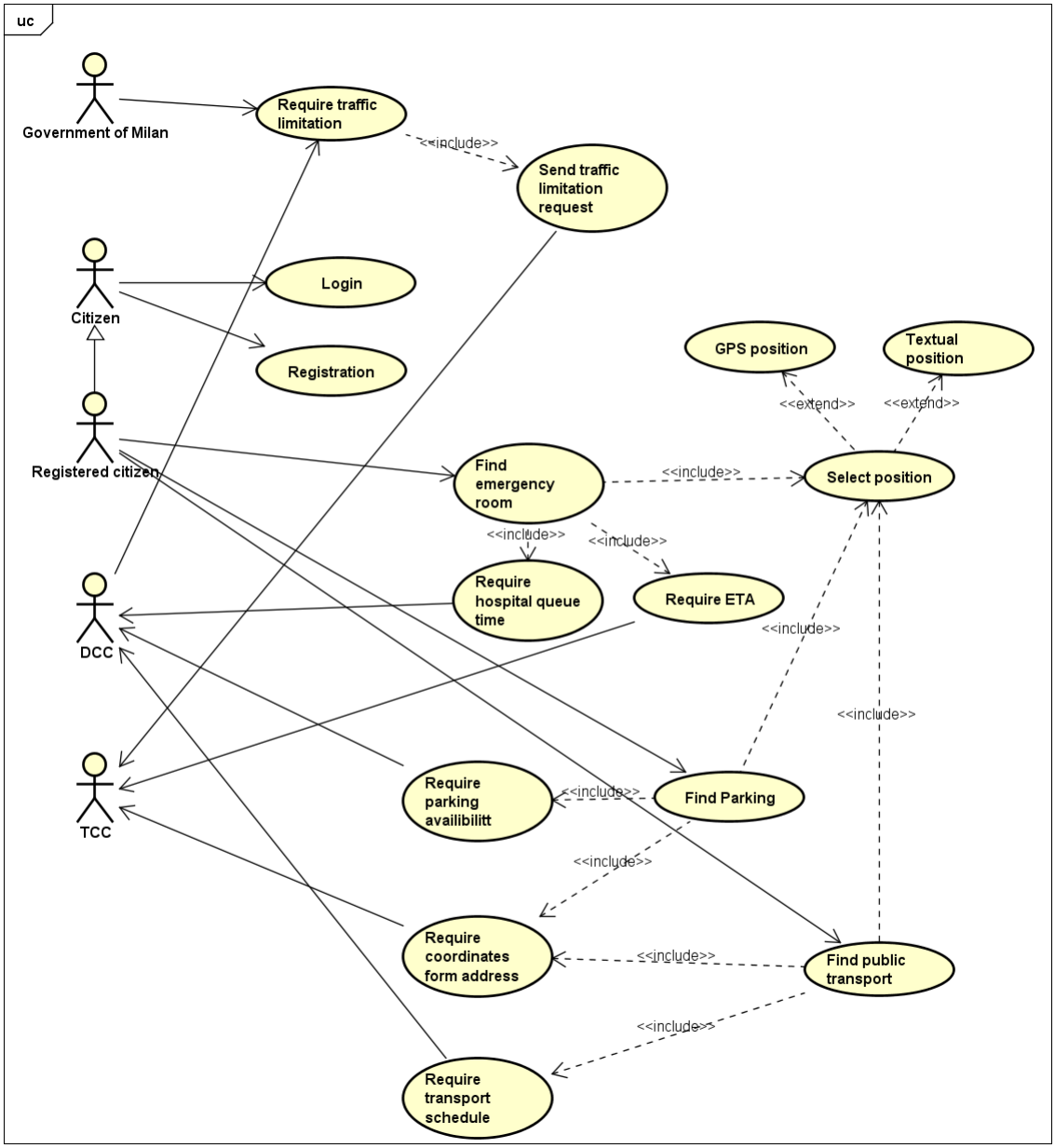


Figure 7: System use case diagram

3.4.2 Class diagram

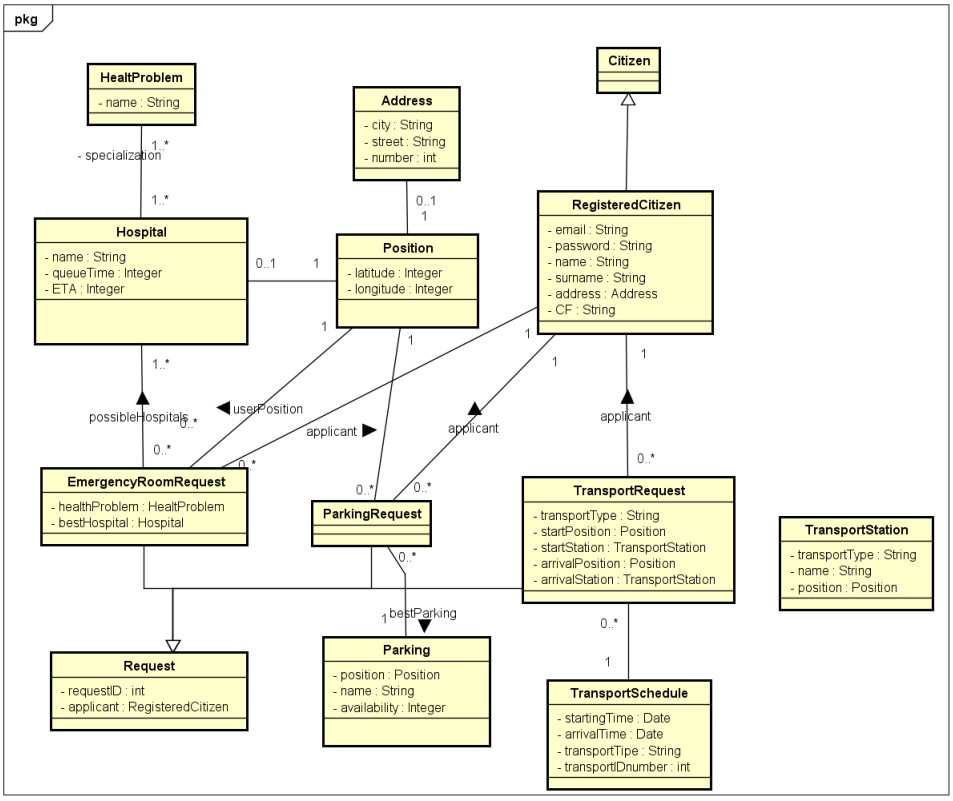


Figure 8: Class diagram of SmartCityAdvisor

3.5 Non functional requirements

3.5.1 Performance Requirements

Performance must be acceptable to grantee a good grade of usability. We assume the response time of the system is close to zero, so the performance are essentially bounded by users internet connection.

3.5.2 Design Constraints

The application will be developed with Java EE so it will inherit all language's constraints.

3.5.3 Software System Attributes

Availability: The application will be accessible online anytime. To achieve this goal could be necessary to use a dedicated server.

Maintainability: The application does not provide any specific API, but the whole application code will be documented to well inform future developers of how application works and how it has been developed.

Portability: The application could be used on any SO which supports JVM.

3.5.4 Security

Web and mobile application: SmartCityAdvisor application implements a login authentication to protect the information of users. Password of user is saved using hashing mechanism but could not be enough. This system do not actually require anything about the strongness of the password so could be developed a system that require an 8 character password with number, letter and special character. Password also is static, user is not involved in password changing so, system will ask to user to change frequently the password for example every 6 month or less. Another useful thing could be some advice to users about how to build a safe password.

Government web app: Before our system is put into operation, access credentials will be provided to the goverment of Milan. When an employee of the government wants to communicate to the system that the traffic in the center must be limited, he has to login using the given credentials. The system changes automatically this credential every 6 month or in case of security violation, and it communicates the new credentials to the govermnent per email.

TCC and DCC API: Our system use some private API provided by DCC and TCC to access the functions provided by them, i.e. to require parking availibility or to ask TCC to limit the traffic in the center.

Server Side: The server side architecture could be implemented dividing strongly the data from application. Application Server is separated from database and from the web server. All zone are divided by firewall. Access to this zone is restricted and forbidden to not authorized user.

4 Alloy

4.1 Modeling

```
open util/integer as integer

//SIGNATURE//
abstract sig Citizen{}
sig RegisteredCitizen extends Citizen{
email: String,
password: String,
name:String,
surname:String,
address:Address,
CF:String,
}

sig Float{}

sig Position{
latitude: Float,
longitude: Float,
address: String,
}

sig HealthProblem{
name: String,
}

abstract sig Request{
requestID: Int,
applicant: RegisteredCitizen,
}
```

```
sig EmergencyRoomRequest extends Request{
healthProblem: HealthProblem,
userPosition: Position,
possibleHospitals: some Hospital,
bestHospital: Hospital,
}{bestHospital in possibleHospitals}

sig Address{
//we assume that the city is always Milan
street: String,
number: Int,
}

sig Hospital{
name:String,
position: Position,
queueTime: Int ,//number of minutes
ETA: Int, //number of minutes
specializations: some HealthProblem,
}

sig ParkingRequest extends Request{
position: Position,
bestParking: Parking,
}

sig Parking{
name: String,
position: Position,
availibility: Int,
}
```

```
sig TransportRequest extends Request{
transportType: String,
startPosition: Position,
arrivalPosition: Position,
startStation: TransportStation,
arrivalStation: TransportStation,
transportSchedule: TransportSchedule,
}


sig TransportStation{
name: String,
transportType: String,
position: Position,
}
sig Time{}

sig TransportSchedule{
startingTime: Time,
arrivalTime: Time,
transportType: String,
transportIDNumber: Int,
}

//facts

//Two different citizens can't have the same email address.
fact noDifferentCitizensWithSameEmail{
no disj c1,c2: RegisteredCitizen | c1.email =c2.email}

//Two different citizens can't have the same CF.
fact noDifferentCitizensWithSameCF{
no disj c1,c2: RegisteredCitizen | c1.CF =c2.CF}

//Two Position can't have the same latitude and longitude
fact Location{
no disj l1,l2:Position | l1.latitude=l2.latitude and l1.longitude=l2.longitude
}
```

```
//Two different addresses can't have the same street and number.
fact noDifferentAddressesWithSameStreetAndNumber{
no disj a1,a2: Address | a1.street =a2.street and a1.number=a2.number}

//Two different health problems can't have the same name.
fact noDifferentHealthproblemsWithSameName{
no disj h1,h2: HealthProblem | h1.name =h2.name}

//Two different hospitals can't have the same name.
fact noDifferentHospitalsWithSameName{
no disj h1,h2: Hospital| h1.name =h2.name}

//Two different parking can't have the same name.
fact noDifferentParkingWithSameName{
no disj p1,p2: Parking| p1.name =p2.name}

//two different station of the same transport type can't have the same name
fact noDifferentStationOfTheSameTransportTypeWithTheSameName{
no disj s1,s2:TransportStation | s1.transportType=s2.transportType
   and s1.name=s2.name}

//two different station of the same transport type can't have the same position
fact noDifferentStationOfTheSameTransportTypeWithTheSamePosition{
no disj s1,s2:TransportStation | s1.transportType=s2.transportType
   and s1.position=s2.position}

//Two different request can't have the same requestID.
fact noDifferentRequestsWithSameRequestID{
no disj r1,r2: Request| r1.requestID =r2.requestID}

//the possible hospitals in a hospital request must be specialized in the same health problem
//indicated in the request
fact hospitalsSpecialization{
all err: EmergencyRoomRequest
   {all h: Hospital|(h in err.possibleHospitals) implies (err.healthProblem in h.specializations)}
}
```

```
//There is at least one hospital for each possible health problem in the list.
fact atLeastOneHospitalForHeachHealthProblem{
all hp:HealthProblem{some h:Hospital | hp in h.specializations}
}

//in a transport request the starting and the arrival station must be different
fact differentStartAndArrivalStation{
no tr: TransportRequest | tr.startStation=tr.arrivalStation}

//The type of the start and the arrival station in a transport request must be the same type
//indicated in the request
fact stationsWithTheCorrectTransportType{
all tr: TransportRequest
   {all s1,s2:TransportStation| (s1=tr.startStation and s2=tr.arrivalStation) implies
      (s1.transportType = s2.transportType and s1.transportType = tr.transportType)}
}


//ASSERT
//the possible hospitals in a hospital request must be specialized in the same health problem
//indicated in the request
assert hospitalsSpecialization{
all err: EmergencyRoomRequest
   {all h: Hospital|(h in err.possibleHospitals) implies (err.healthProblem in h.specializations)}
}
check hospitalsSpecialization for 5

//There is at least one hospital for each possible health problem in the list.
assert atLeastOneHospitalForHeachHealthProblem{
all hp:HealthProblem{some h:Hospital | hp in h.specializations}
}
check atLeastOneHospitalForHeachHealthProblem for 3

//in a transport request the starting and the arrival station must be different
assert differentStartAndArrivalStation{
no tr: TransportRequest | tr.startStation=tr.arrivalStation}
check differentStartAndArrivalStation for 5
```

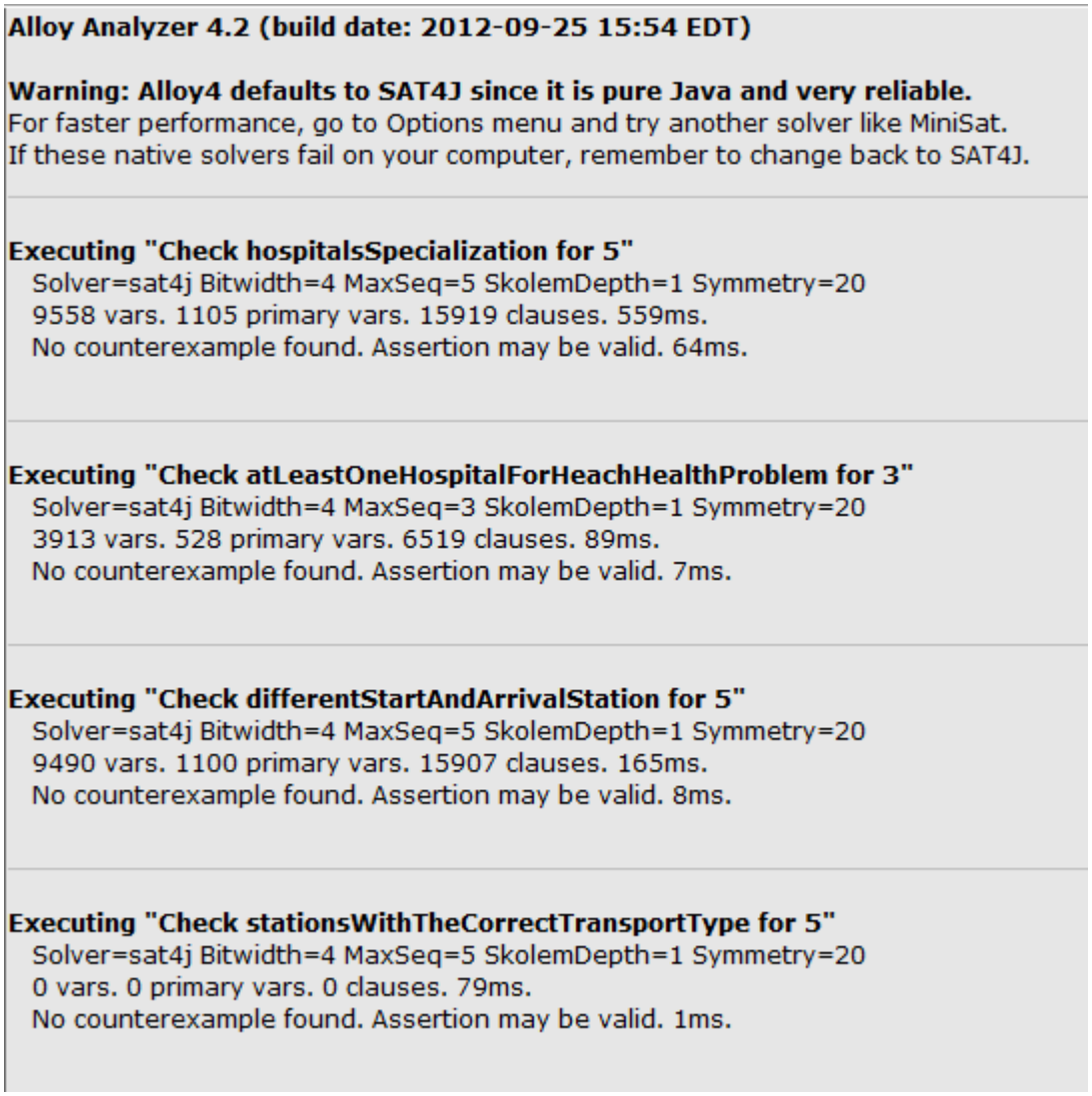
```
//The type of the start and the arrival station in a transport request must be the same type
//indicated in the request
assert stationsWithTheCorrectTransportType{
all tr: TransportRequest
   {all s1,s2:TransportStation| (s1=tr.startStation and s2=tr.arrivalStation) implies
      (s1.transportType = s2.transportType and s1.transportType = tr.transportType)}
}
check stationsWithTheCorrectTransportType for 5

pred EmergencyRoomRequestWorld(){
#RegisteredCitizen>0
#EmergencyRoomRequest=1
#ParkingRequest=0
#TransportRequest=0
#Parking=0
#TransportStation=0
#TransportSchedule=0
}
run EmergencyRoomRequestWorld for 2 but exactly 2 String

pred ParkingRequestWorld(){
#RegisteredCitizen>0
#EmergencyRoomRequest=0
#ParkingRequest=1
#TransportRequest=0
#Hospital=0
#HealthProblem=0
#TransportStation=0
#TransportSchedule=0
}
run ParkingRequestWorld for 2 but exactly 2 String

pred TransportRequestWorld(){
#RegisteredCitizen>0
#EmergencyRoomRequest=0
#ParkingRequest=0
#TransportRequest=1
#TransportSchedule=1
#Hospital=0
#HealthProblem=0
#Parking=0
}
run TransportRequestWorld for 2 but exactly 2 String
```

4.2 Analyzer


Alloy Analyzer 4.2 (build date: 2012-09-25 15:54 EDT)
Warning: Alloy4 defaults to SAT4J since it is pure Java and very reliable.
For faster performance, go to Options menu and try another solver like MiniSat.
If these native solvers fail on your computer, remember to change back to SAT4J.
Executing "Check hospitalsSpecialization for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
9558 vars. 1105 primary vars. 15919 clauses. 559ms.
No counterexample found. Assertion may be valid. 64ms.
Executing "Check atLeastOneHospitalForHeachHealthProblem for 3"
Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
3913 vars. 528 primary vars. 6519 clauses. 89ms.
No counterexample found. Assertion may be valid. 7ms.
Executing "Check differentStartAndArrivalStation for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
9490 vars. 1100 primary vars. 15907 clauses. 165ms.
No counterexample found. Assertion may be valid. 8ms.
Executing "Check stationsWithTheCorrectTransportType for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
0 vars. 0 primary vars. 0 clauses. 79ms.
No counterexample found. Assertion may be valid. 1ms.

Executing "Run EmergencyRoomRequestWorld for 2 but exactly 2 String"
Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
2408 vars. 362 primary vars. 3991 clauses. 58ms.
Instance found. Predicate is consistent. 74ms.

Executing "Run ParkingRequestWorld for 2 but exactly 2 String"
Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
2411 vars. 362 primary vars. 3998 clauses. 52ms.
Instance found. Predicate is consistent. 41ms.

Executing "Run TransportRequestWorld for 2 but exactly 2 String"
Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
2411 vars. 362 primary vars. 3998 clauses. 43ms.
Instance found. Predicate is consistent. 37ms.

4.3 Worlds generated

Here the worlds associated to the three kind of request that the system can receive:

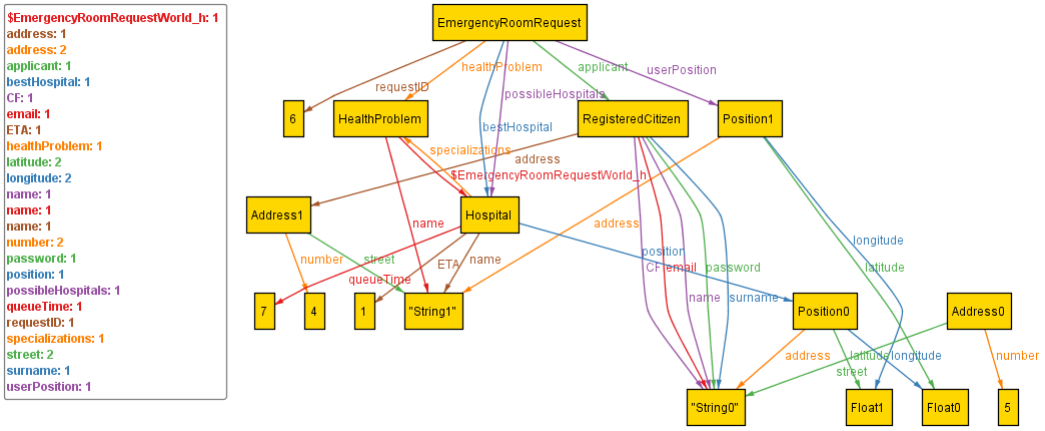


Figure 9: World associated to the emergency room request

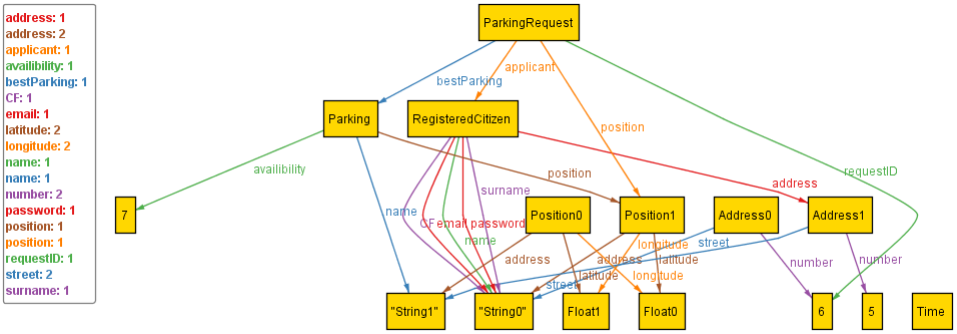


Figure 10: World associated to the parking request

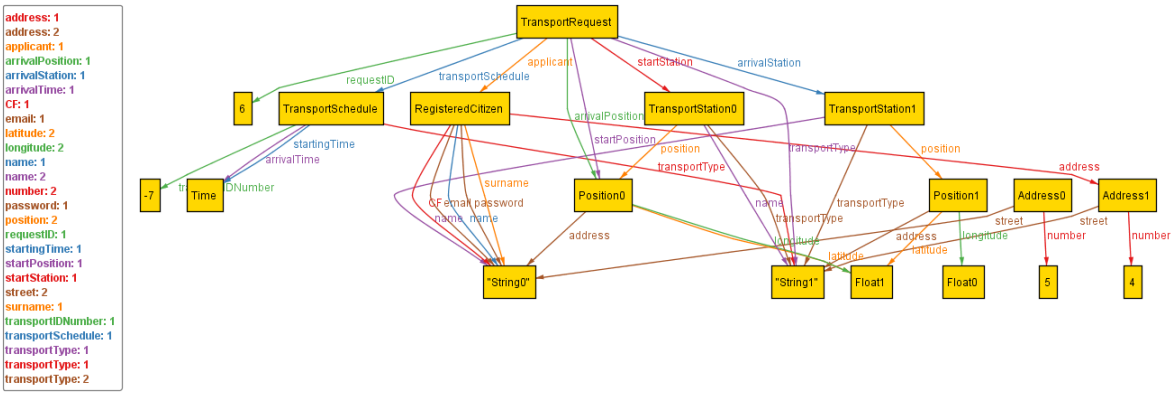


Figure 11: World associated to the public transport request

5 Appendix

5.1 Software and tools used

- LyX: to redact and to format this document.
- Astah Professional (http://astah.net/editions/professional): to create Use Cases Diagrams, Sequence Diagrams, Class Diagrams and State Machine Diagrams
- Alloy Analyzer(http://alloy.mit.edu/alloy/): to prove the consistency of our model.
- Balsamiq Mockups(http://balsamiq.com/products/mock-ups/): to create mock-ups.

5.2 Hours of work

- Mirko Basilico: ~28 hours;
- Federico Dazzan: ~28 hours.