Master degree in Computer Engineering

Large Scale and Multi Structured Databases

UNIVERSITÀ DI PISA

TASK 3

Aprile 2020

Federico Cappellini, Andrea Lelli, Alberto Lunghi, Lorenzo Susini

1 - Introduction

The purpose of this project is double: we want both to manage information about users, in particular about their positiveness or not to a given deseas (Covid-19 in our case-study), and also to obtain information about some points of interest where commonly users spend their time.
What we are going to do is defining these just mentioned points of interest, and to discover relations with the users subscribed to the service. The relations will be found through GPS technologies: when an user, who needs to have his GPS receiver activated, is detected in the nearby of the area defined for a given place, the relation between him and the place is established and it will be persistent for a time (to be established, for example, by some administrator). This relation will also have associated a timestamp.
Specifically we are going to deploy a database system, based on a graph database, that will store traces of users simulated through some simulation tool.
In the case of the Covid-19 we would like to have a relation persistent for at least 2 weeks (the incubation period), because in this way, if the user gets positive to the deseas all the other users that have been in the same place of the infected will be notified by the application. Another useful feature is the possibility to compute some aggregate analytics, such as the estimation of the "risk of infection" of a person or for a given place.
Of course with this type of service we can make some other useful computations, for example what is the level of frequentation of a given place, to know its acceptance index. This last case can be useful, for instance, to know which are the best restaurants nearby or if it is likely to find long queues in a supermarket at a given time.

2 - Actors, Functionalities, Use-Case Diagram and Requirements

2.1 Functional requirements

The actors of this application are intended to be the physical users subscribed to the service and a delegated person certifying users' and places' risks of infection.

A **User** can:

- subscribe to the system.
- given a number of hops, check the number of infected users within that number of hops.
- find its risk infection risk index.
- find the infection risk index of a specified place.
- see the most critical places, that he frequented, w.r.t. the risk of infection.

The **Delegated person** can:

- find the most critical places w.r.t. the infection risk index.

- update the status of an user from “infected” to “not infected” or vice versa.
- add new places to the database, from a textual file.

2.2 Non-functional requirements

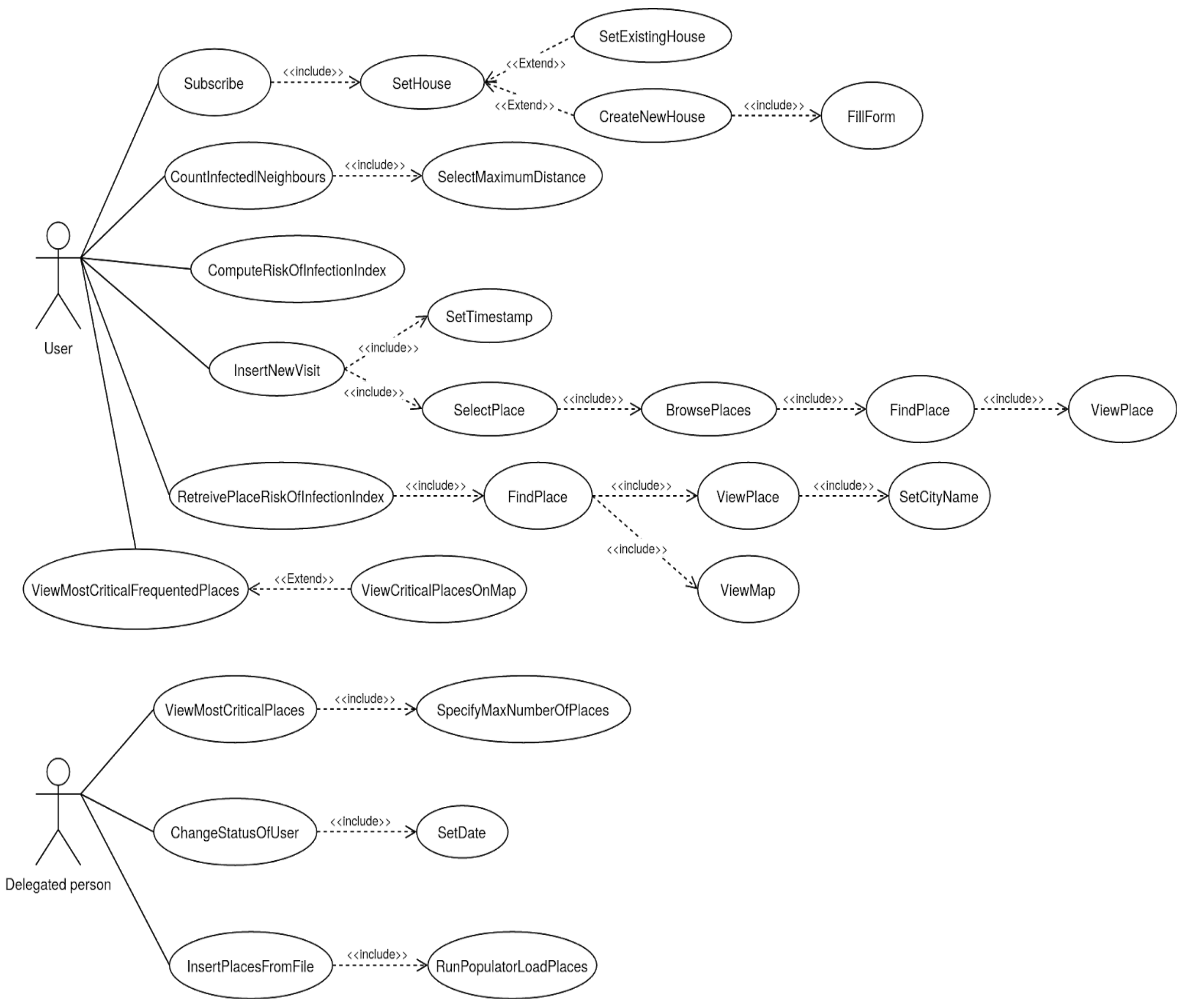
The **non-functional requirement** is:

- **Portability**: the application should be portable among different architectures, exploiting the «Write once, run everywhere» motto.

2.3 Use case diagram

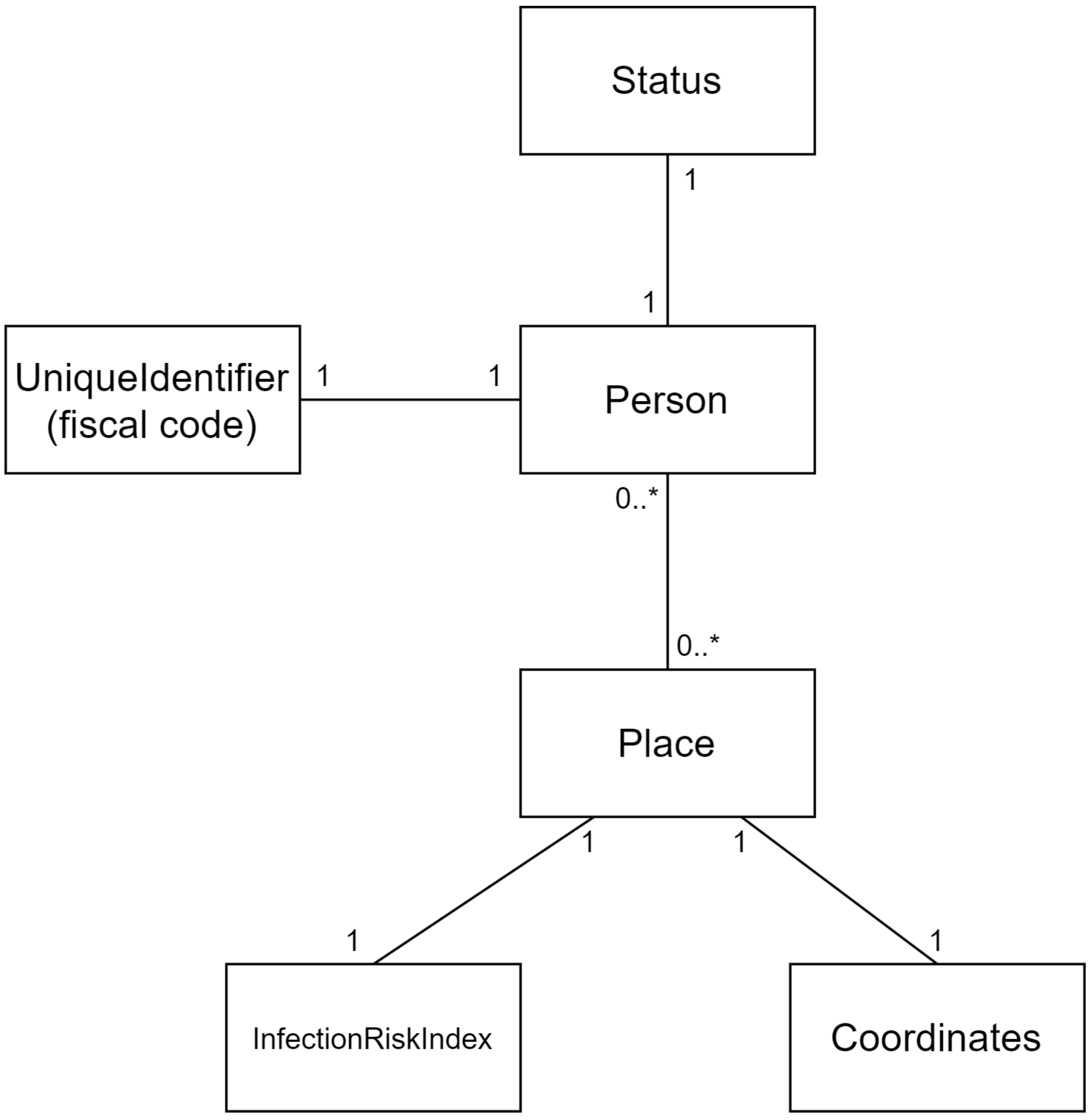
The following is the **use case diagram** for 2 different actors:

- The Delegated person
- User



3 - Data modelling

The following is the UML diagram of the data used in the application:



4 - Queries

Before defining which are the queries supported by the systems, it is necessary to identify some parameters:

- **VALIDITY_PERIOD**. Not all relations has the same importance for the computation of the results. In particular older relations has less importance respect to others more recent. In particular, all relations older than VALIDITY_PERIOD are omitted. Normal values for this parameter are in the order of weeks, by default VALIDITY_PERIOD is equal to 2 weeks since it is the same amount of time of the incubation period of CoVid-19, but it can be increased considering the capacity of this virus and other pathogens to persist of physical surfaces.

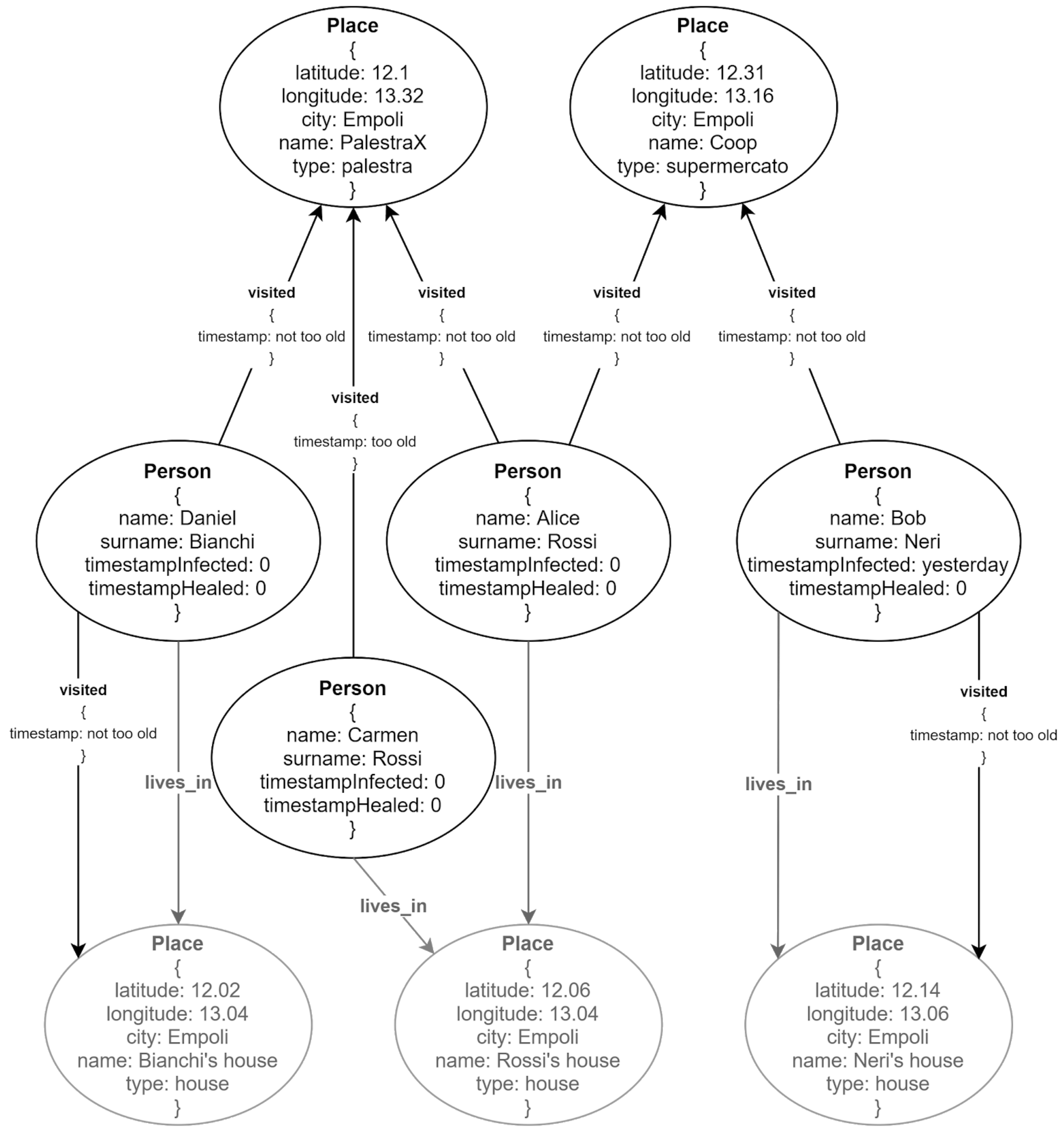
The following are the analytical queries on the data:

4.1 - Query: infected in a given social distance

This query is meant to be requested by the normal user.

Domain specific	**Graphic centric**
Find the infected users at a given distance *n* around the user	Given the number of hops *n* and a node of type “Person”, find all the nodes of type “Person” that are connected with the specified node within a number of hops equal to *n* and whose relations have the field “timestamp” greater than (current_timestamp - VALIDITY_PERIOD). Of those, return the number of nodes of type “Person” which have the property (*CURRENT_TIMESTAMP -* VALIDITY_PERIOD> *infectionTimestamp* AND *CURRENT_TIMESTAMP* < *healedTimestamp*)

Example: considering the following situation,



Place { latitude: 12.1 longitude: 13.32 city: Empoli name: PalestraX type: palestra }
Place { latitude: 12.31 longitude: 13.16 city: Empoli name: Coop type: supermercato }
visited { timestamp: not too old }
visited { timestamp: not too old }
visited { timestamp: not too old }
visited { timestamp: not too old }
visited { timestamp: too old }
Person { name: Daniel surname: Bianchi timestampInfected: 0 timestampHealed: 0 }
Person { name: Alice surname: Rossi timestampInfected: 0 timestampHealed: 0 }
Person { name: Bob surname: Neri timestampInfected: yesterday timestampHealed: 0 }
visited { timestamp: not too old }
Person { name: Carmen surname: Rossi timestampInfected: 0 timestampHealed: 0 }
visited { timestamp: not too old }
lives_in
lives_in
lives_in
lives_in
Place { latitude: 12.02 longitude: 13.04 city: Empoli name: Bianchi's house type: house }
Place { latitude: 12.06 longitude: 13.04 city: Empoli name: Rossi's house type: house }
Place { latitude: 12.14 longitude: 13.06 city: Empoli name: Neri's house type: house }

if Daniel decides to run this query with a very short distance (n = 2), the result will be 0 (only 1 negative person - Alice - is found within 2 hops). But, if he decides to run the same query with a larger distance (in this example n = 4), the result will be 1 (Bob is indeed infected).

4.2 - Query: user's risk of infection

This query is meant to be requested by the normal user.

Domain specific	**Graphic centric**
For the current user, find the infection risk	Given a node of type "Person", find the

index, that is related to the distance between him and another user who is infected	minimum number of hops needed to reach another node of type “Person”, with *CURRENT_TIMESTAMP* > *infectionTimestamp* AND *CURRENT_TIMESTAMP* < *healedTimestamp*, whose relations have the field “timestamp” smaller than (*CURRENT_TIMESTAMP* - *VALIDITY_PERIOD*)

Example: considering the same situation of the example before, if Daniel runs this query, the result will be 4 because from Daniel to Bob (who is the closest infected person) there are 4 hops.

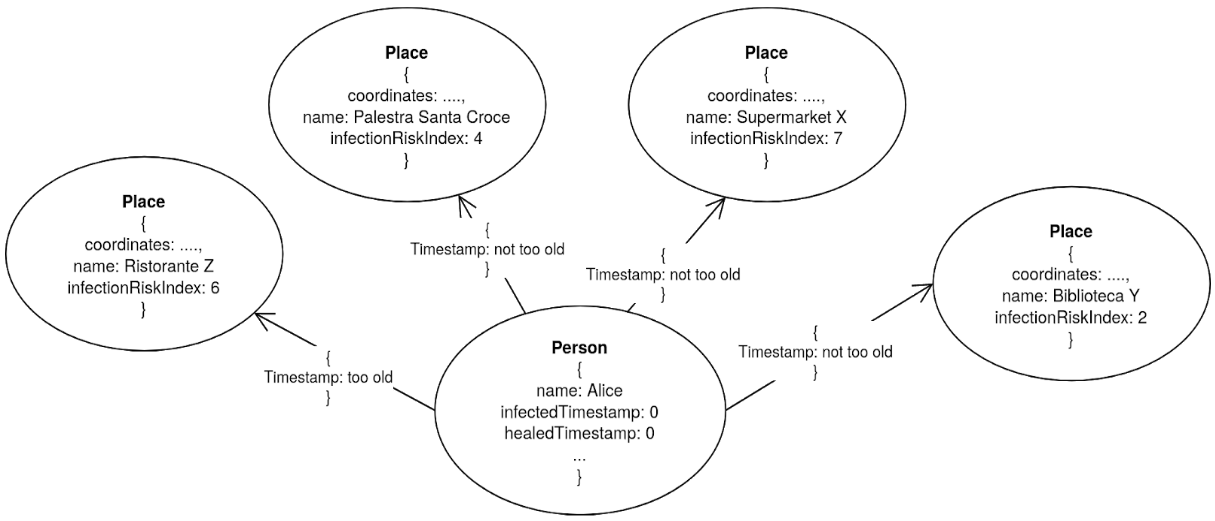
4.3 - Query: user’s most riskful places

Also this query is meant to be requested by the normal user.
Before launching this query, one parameter has to be configured:

- **NODE_NUMBER** whom meaning can be inferred from the following description

Domain specific	**Graphic centric**
Considering the current user, retrieve a list containing the most riskful places that he frequents	For a given node of type “Person”, retrieve the nodes of type “Place” that have at least one incoming edge outgoing from the selected “Person” node. Order them by the property *infectionRiskIndex* and return the first *NODE_NUMBER*

Example: considering the graph below, if Alice runs the query specifying NODE_NUMBER=3, the result will be the list composed by “Supermarket X”, “Ristorante Z”, “Palestra Santa Croce”, in this exact order (descending order of the infection risk index). Note that even if the timestamp of “Ristorante Z” is too old, it is considered by the query, this is because it must take into account all the relations between the user and the visited places until they are removed from the database (e.g. if Alice went to the restaurant 1 month ago, she may wants to return to the restaurant to eat there again, so it is useful to know if it is too riskfull).


Place { coordinates: ...., name: Palestra Santa Croce infectionRiskIndex: 4 }
Place { coordinates: ...., name: Supermarket X infectionRiskIndex: 7 }
Place { coordinates: ...., name: Ristorante Z infectionRiskIndex: 6 }
Place { coordinates: ...., name: Biblioteca Y infectionRiskIndex: 2 }
Person { name: Alice infectedTimestamp: 0 healedTimestamp: 0 ... }
{ Timestamp: too old }
{ Timestamp: not too old }
{ Timestamp: not too old }
{ Timestamp: not too old }

4.4 - Query: risk of infection index

This query is meant to be requested by the delegated person only.

Domain specific	**Graphic centric**
Given a place, retrieve its infection risk index	For a given node of type “Place”, retrieve its property *infectionRiskIndex*

The infection risk index is a mathematical real index that express how high the risk of infection is for a given place. You can find more information and an example about how the index is computed in paragraph 4.8 - Computing the infection risk index.

4.5 - Query: most critical places

Also this query is meant to be requested by the delegated person only.
Before launching this query, one parameter has to be configured:

- **NODE_NUMBER** whom meaning can be inferred from the following description

Domain specific	**Graphic centric**
Find the *NODE_NUMBER* most critical places w.r.t. the risk of infection, i.e. the infection risk index of the place	Retrieve the first *NODE_NUMBER* nodes of type “Place” ascendingly ordered by their property *infectionRiskIndex*

Example: considering the previous graph, if the delegated person runs this query specifying NODE_NUMBER=2, the result will be the list composed by “Supermarket X”, “Ristorante Z”, that are ordered from the higher index to the lower.

4.6 - Query: update the status of a user

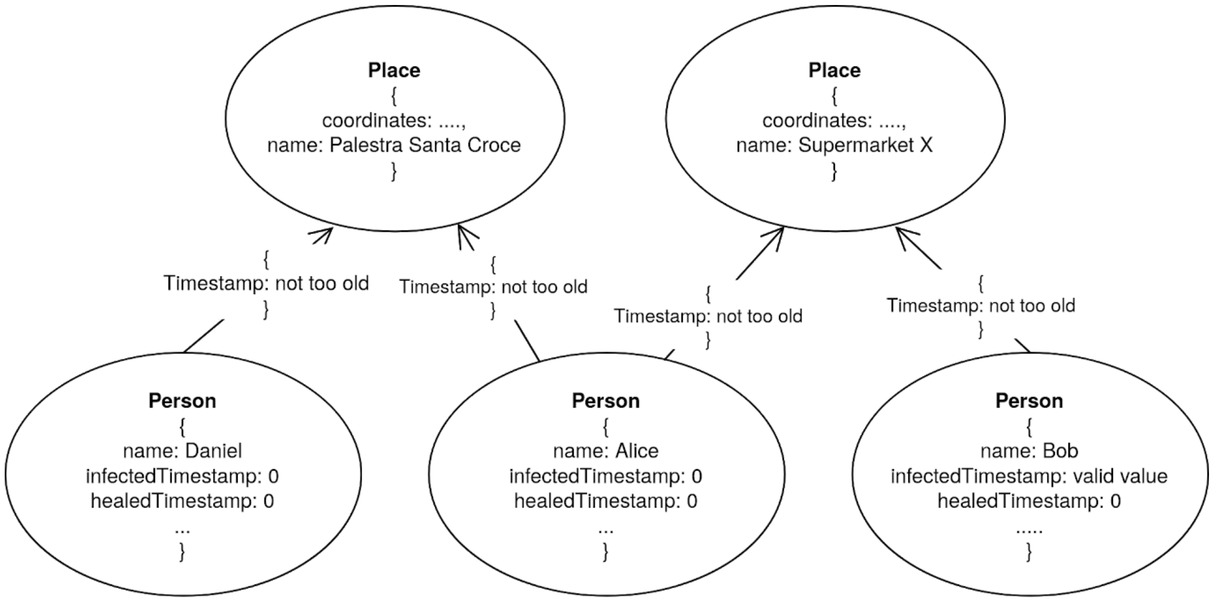
Also this query is meant to be requested by the delegated person only.

Domain specific	**Graphic centric**
Update the status of a user	Given a node of type "Person", update its properties *infectedTimestamp and healedTimestamp*

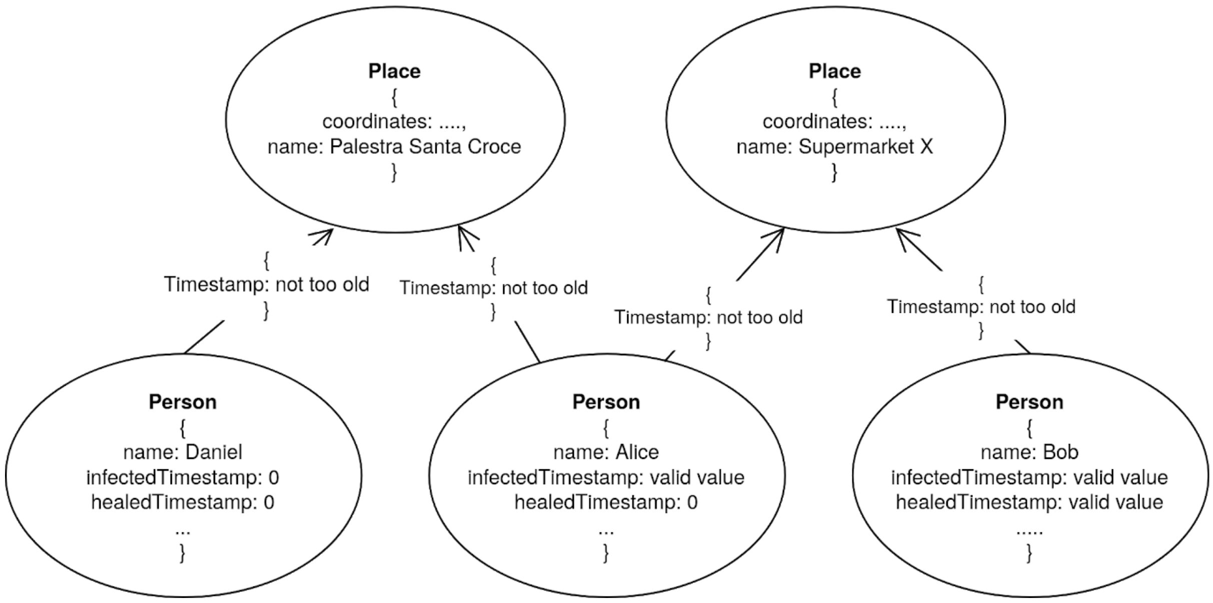
When a person is detected as infected (or healed), only the delegated person can update the status in the database since he is the only authorized actor.
The status is represented by two timestamps (*infectedTimestamp* and *healedTimestamp*):

- if the person has never been infected his node has the two timestamps equal to zero;
- if the person is infected his node has the property *infectedTimestamp* with a valid timestamp value (or also has the property *healedTimestamp* but it is equal to zero);
- if the person was infected and now he is healed his node has both the timestamps with valid timestamp values.

Example:


Place { coordinates: ...., name: Palestra Santa Croce }
Place { coordinates: ...., name: Supermarket X }
{ Timestamp: not too old }
{ Timestamp: not too old }
{ Timestamp: not too old }
{ Timestamp: not too old }
Person { name: Daniel infectedTimestamp: 0 healedTimestamp: 0 ... }
Person { name: Alice infectedTimestamp: 0 healedTimestamp: 0 ... }
Person { name: Bob infectedTimestamp: valid value healedTimestamp: 0 ..... }

Supposing that the delegated person updates the status of Bob because he received the news regarding his successful cure, and also the status of Alice, who had became infected. Then the graph will change to the following:


Place { coordinates: ...., name: Palestra Santa Croce }
Place { coordinates: ...., name: Supermarket X }
{ Timestamp: not too old }
{ Timestamp: not too old }
{ Timestamp: not too old }
{ Timestamp: not too old }
Person { name: Daniel infectedTimestamp: 0 healedTimestamp: 0 ... }
Person { name: Alice infectedTimestamp: valid value healedTimestamp: 0 ... }
Person { name: Bob infectedTimestamp: valid value healedTimestamp: valid value ..... }

Of course the value of the property “healedTimestamp” of Bob corresponds to a timestamp that is more recent than the one of the “infectedTimestamp” property.

4.7 - Other user’s queries

The following are the remaining queries that a user or the delegated person can execute.

Domain specific	**Graphic centric**
Subscribe a user	Add a node of type “Person” in the graph
Register a new detection of a user who went into a specific place	Add an edge in the graph outgoing form a node of type “Person” and incoming to a node of type “Place”, assigning it the *timestamp* field value
Insert a place	Add a node of type “Place” in the graph

4.8 - Computing the infection risk index

Both the queries *infection risk index* and *most critical places* rely on the capacity of the application to compute the homonym **infection risk index** that is stored as a property of the nodes of type “Place”.
This index is a real value that express how high the risk of infection is for a given place considering the following factors:

- how many infected and not infected people have visited the place
- when these people visited the place w.r.t. the current date and time
- the area of the place (to take into account the possible closer interactions between users)

Before describing how this index is computed, some fundamental parameters have to be considered:

- $I_1, ..., I_{N_i}$ are timestamps of the visits that the place in exam received from infected people
- $NI_1, ..., NI_{N_{ni}}$ are the timestamps of the visits that the place in exam received from non infected people
- N_{NI} is the number of visits from non-infected people
- N_I is the number of visits from infected people
- P_{fn} is the so called false-negative probability. It is the probability that a person is recognized from the system as non infected while, in reality, it is. This probability takes into account also that people who are infected but they don’t show any symptoms and so they do not know to be infected.

The final formula is:

$$infectionRiskIndex = \left(\sum_{j=1}^{N_I} \beta(I_j) + \sum_{j=1}^{N_{NI}} P_{fn} * \beta(NI_j) \right) * \alpha(AREA)$$

where β and α are two functions described later and *AREA* is a variable that represent the area of the building in exam.

As it is possible to infer, the *infectionRiskIndex* depends linearly from the number of visits by infected people. For this reason, for each visit the index grows of a specific factor determined by β.
Considering the probability to be infected without knowing it (false negative probability) as an independent bernoullian random variable, it is possible to affirm that, on average, on a pool of N_{NI} people recognized by the system as healthy, $N_{NI} * P_{fn}$ of them are infected. This can be proved because the sum of independent bernoullian random variables is a binomial random variable whose expectation is exactly the one described before.
For this reason, every visit from a healthy person has to be considered scaled by a factor P_{fn}.

VISIT TEMPORAL WEIGHT β

β is a multi-linear real function that takes as input a timestamp and returns the weight of the visit to which the timestamp is referred to.
The multi-linearity nature of the function borns to take into account two different scenarios:

- a recent visit to a place by an infected person is dangerous because it is unlikely that the responsible for the place had found time to clean or sanitize it,
- a non recent visit is, instead, less dangerous: the place could have been already cleaned or sanitized.

To discern between these two limits, 3 configurable parameters have been introduced:

- **RECENT_TIMESTAMP_WINDOW**, that is the threshold described before. By default it is set to 3 days.
- **RECENT_TIMESTAMP_WEIGHT**, that is the maximum value in case of a recent timestamp. By default this value is 4.0.
- **NON_RECENT_TIMESTAMP_WEIGHT**, that is the maximum value in case of a non recent timestamp. By default this value is 2.0.

The mathematical expression is:

$$\beta(t) = \begin{cases} 0, & t + VALIDITY\_PERIOD < now \\ NB\left(\left[1 - \dfrac{now - t}{VALIDITY\_PERIOD}\right], 1, RTW\right), & t + RECENT\_TIMESTAMP\_WINDOW \geq now \\ NB\left(\left[1 - \dfrac{now - t}{VALIDITY\_PERIOD}\right], 1, NRTW\right), & t + RECENT\_TIMESTAMP\_WINDOW < now \\ RTW, & t = now \end{cases}$$

where

- *RTW* is simply the *RECENT_TIMESTAMP_WEIGHT* parameter
- *NRTW* is simply the *NON_RECENT_TIMESTAMP_WEIGHT* parameter
- *NB(f,a,b)* is a function that returns the value of the function *f* normalized between *a* and *b*.

AREA WEIGHT α

During an epidemic it is easy to infer that being in a large building, like a big supermarket, together with other 10 people is really less dangerous that being but in a 10 square meters town shop with the same number of people.

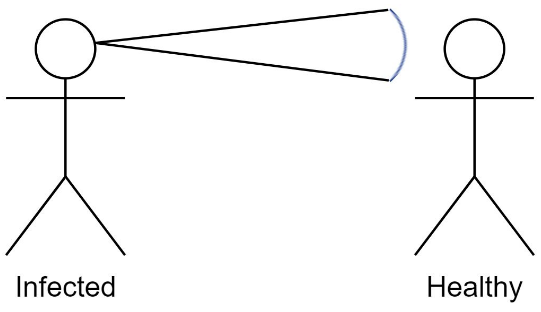
For this reason, to compute the infection risk index, another weight that depends on the area of the place in exam has to be considered: α is indeed a real function that returns this weight.
Its mathematical formula is:

$$\alpha(A) = \begin{cases} MAX\_AREA\_WEIGHT, & if \left(\sqrt{\frac{A}{\pi}}\right) < RADIUS\_THRESHOLD[0] \\ NB\left(\alpha'\left(\sqrt{\frac{A}{\pi}}\right), 1, MAX\_AREA\_WEIGHT\right), & otherwise \\ 1, & if \left(\sqrt{\frac{A}{\pi}}\right) > RADIUS\_THRESHOLD[3] \end{cases}$$

where

- as before, *NB(f,a,b)* is a function that returns the value of the function *f* normalized between *a* and *b*.
- **MAX_AREA_WEIGHT** is a configurable parameter that represent the maximum weight this function can return and so the importance of the size of the building to compute the infection risk index. By default this parameter is set to 4.0.
- *RADIUS_THRESHOLD* is an array of “thresholds” that is described later.

The core of this function is the function α'. This function takes as input an estimation of the average distance between an infected and an healthy person and return a number that is higher with the probability that an ill person can infect another.
In this case, this estimation is done using the square root of *A* divided by *π.* This is because this value is the radius in case the area of the building is a circle and, to simplify the whole application, all the areas are considered as circles. Indeed, considering two points taken at random in a circle, their average distance is directly proportional to the radius.
How it is possible to retrieve the value of α'?
Considering air and micro particles of saliva as the transmission medium of the disease, an approximation can be done: the quantity of “infected” air an healthy person receives from an interaction with an infected one could be seen as inversely proportional to the external round surface of the cone shown in the figure below.



This is because the healthy person enters in contact only with a fraction of this surface of the cone. The fraction depends, obviously, on the distance (that is the radius) and since this surface grows with the square of it, the fraction decreases with it.

Anyway, this is a rough approximation and to better computer α' it is necessary to use a more complex expression where the exponent is not a constant of value 2, but a function:

$$\alpha'(r) = \frac{k}{r^{p(r)}}$$

where

- k is an apparently meaningless constant. However, it was introduced to mitigate an approximation problem that could occurs with floating point numbers (Java's double) very close to 0 and to better visualize the curve as shown later. By default $k = 10000$.
- p is the function mentioned before:

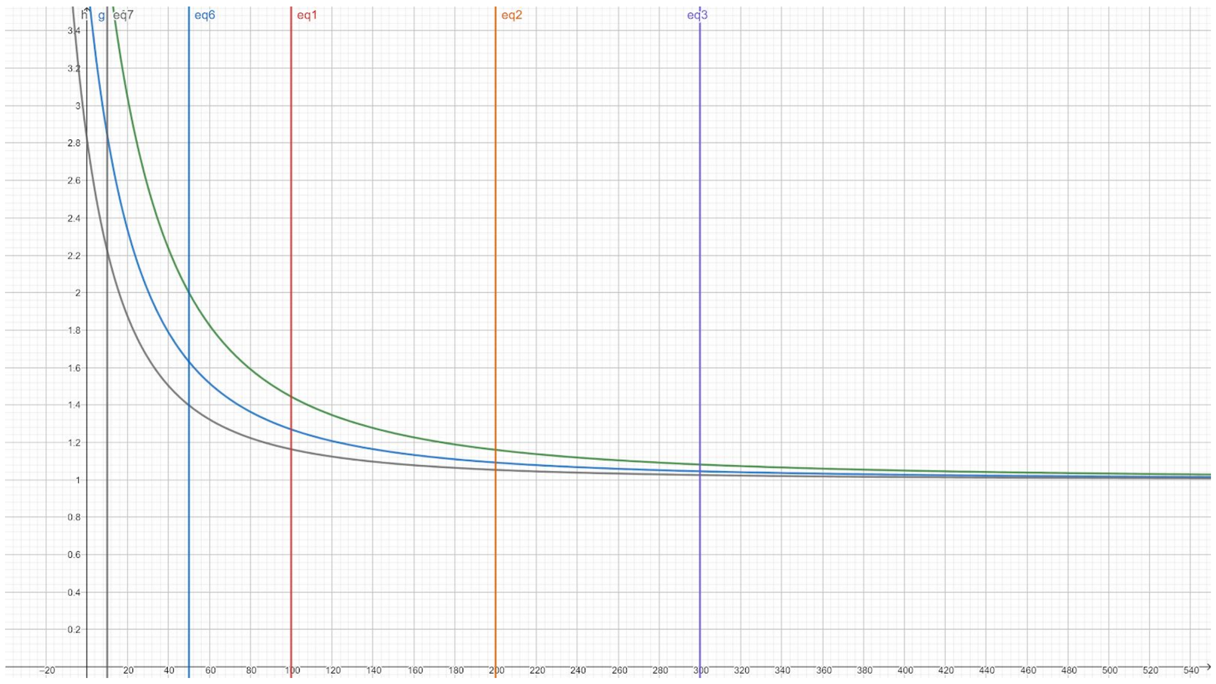
A configurable parameter has now to be described:

- **RADIUS_THRESHOLD** is an array that is used as a parameter. By default:
  - *RADIUS_THRESHOLD*[0] = 10
  - *RADIUS_THRESHOLD*[1] = 50
  - *RADIUS_THRESHOLD*[2] = 100
  - *RADIUS_THRESHOLD*[3] = 300

The function $p(r)$ makes $\alpha'(r)$ aware of the fact that the model described before is too "ideal" for a real world scenario. In particular, the higher the distance the higher is the number of external factors that can interfere with the propagation of the pathogen such as air movements, obstacles, etc.
$p(r)$ has the following expression:

$$p(r) = \begin{cases} 2.0, & r < RADIUS\_THRESHOLD[1] \\ 2.1, & r < RADIUS\_THRESHOLD[2] \\ 2.2, & r < RADIUS\_THRESHOLD[3] \end{cases}$$

To a better understanding, $\alpha'(r)$ is a composition of 3 (bounded) hyperbolas. These 3 hyperbolas are drawn in the graph below.



5 - Dataset

Another problem we have encountered is how to populate the graph database. The following are the steps that are taken to achieve this purpose.

5.1 - Users

For the generation of the users, two large datasets are downloaded for free from the site “data.world”. One dataset contains Italian names, both males and females, and the other contains Italian surnames.
Then, through a Python script, from the names’ dataset 100 names are selected at random with the respective surnames, again at random, from the other dataset. Surnames are extracted and assigned to the respective names in such a way that families are generated: 20 families composed by 2 people and 20 families composed by 3 people.
For a given user a 3-digits identifier is assigned (in a real scenario an ID have to be the fiscal code of the user), and after that all the users are written in a file called “people_dataset.txt”. The lines of this file are composed in this way: “**ID,LAST_NAME,FIRST_NAME**”.

5.2 - Places

The selection of the places is done manually choosing them from the city of Empoli(FI), in such a way that the most common points of interest are considered.
Also the places are written in a textual file, called “places.txt”, and for each place the respective coordinates are found, through the usage of Google Maps. Based on the “practical” knowledge of the selected places, an approximate area, or better an

approximation of the ray of the circular area covered by the place, is assigned to each of them.
Of course each family has an home, so in the aforesaid file the homes of the 40 families have to be inserted. The coordinates of those are generated randomly selecting random positions in Google Maps, and for each one a standard ray of 10 meters is specified.
The lines of this file are composed in this way:
"**NAME,TYPE,LATITUDE,LONGITUDE,RAY,CITY**".
Types can be "supermercato", "struttura sportiva", "sanitario", "house", etc.
If the delegated person wants to add some place(s), he has to use a textual file with the same format of the "places.txt" file used to populate the database. The path of the file containing the new places has to be specified as second argument when starting the application with the "--addplaces" option.

5.3 - Movements

The most challenging part is the generation of a set of random movements for the users of the dataset. These are generated considering a time interval of one month, March.
Consider that, because of the quarantine, people are invited to stay at home, so for each family we can assume that only one component per day will move (work, commissions, medical reasons).
Furthermore, a person who leaves his home will go randomly to one or two different places, before returning home. We assume different "movement laws" with the respect to the number of places to visit:

- 1 place: the first movement will be between 8am and 10am, and the homecoming will be between 1pm and 7pm
- 2 places: the first movement will be between 8am and 10am, the second movement will be between 1pm and 6pm, and the homecoming will be between an hour after the last movement and 7pm.

For each user a textual file is generated, with the name of the user's ID, where the generated random movements are written line by line.
The lines of these files are composed in this way: "**DATE TIME,PLACE_NAME**", where the date has the format "dd/MM/YY" and the time has the format "HH:mm".
All these steps are implemented again with a Python script, using the functions "randint()" and "randrange()" from the "random" library, to guarantee the needed randomicity mentioned before.

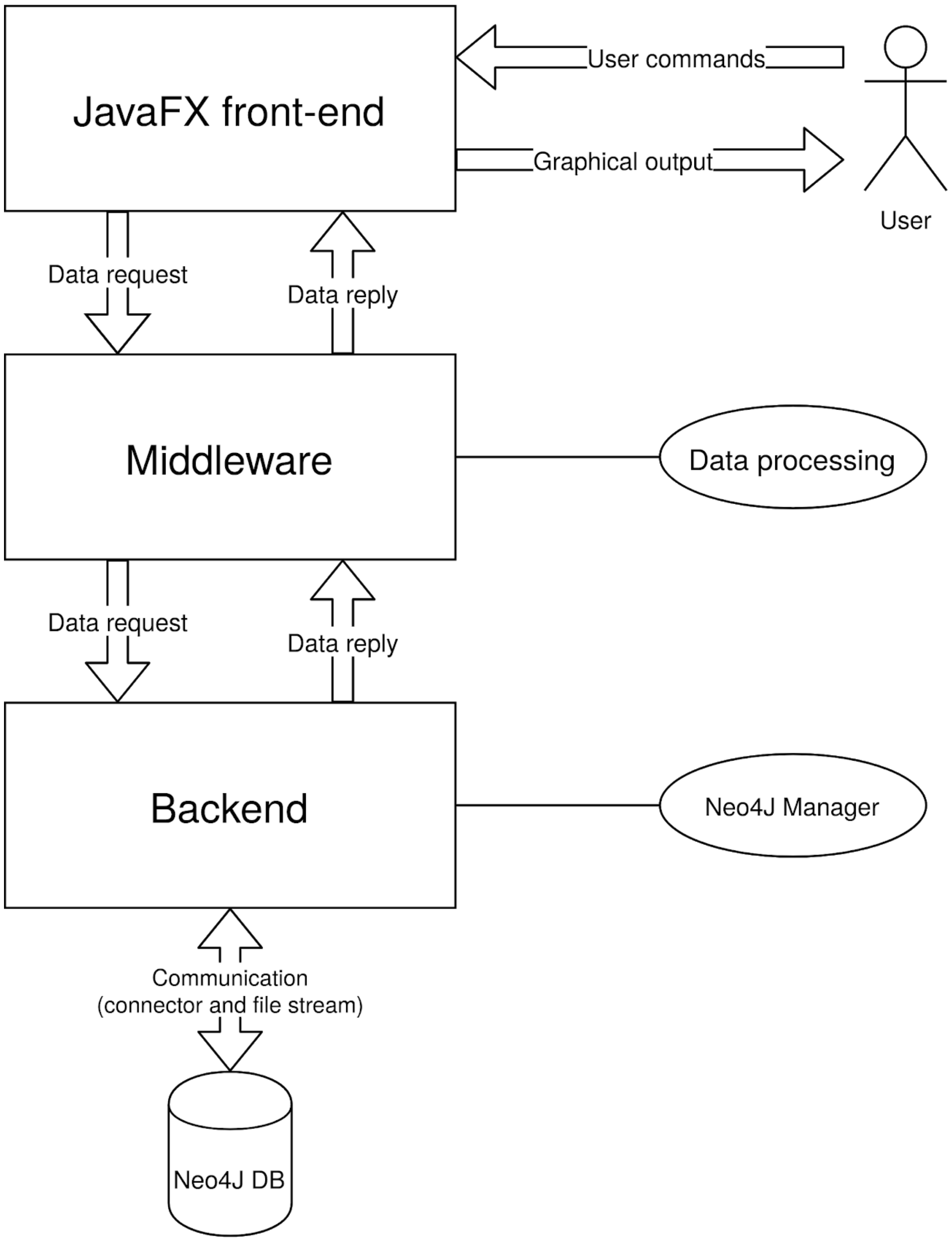
5.4 - Data removal

The data, related to the visits of the users, in the case of the Covid-19 pandemic, become useless after 14 days, the incubation period. So we decided to implement a background job that runs on the graph database every day, that removes all the visits in the DB that are older than 14 days. This last period can be settled by modifying it directly in the code related to the background job, that is shown here:

```
CALL apoc.periodic.repeat('removeOldVisit', "MATCH (a:Person) -
[r:visited] -> (b:Place) WHERE r.timestamp <
apoc.date.currentTimestamp() - 14*24*60*60*1000 DELETE r",
60*60*24)
```

6 - Software architecture and Data Flow

The following is a graphical representation of the overall architecture:



7 - Indexes study

Indexes help Neo4J to speed up the executions of certain queries, specifically, it may be useful to define indexes on those fields used by the most commonly performed queries or those queries that are more computationally weighted to let the database to better organize the data ready to be processed.
Since that the architecture and logic of a graph database fits almost perfectly the domain of our problem, our solution would not need any indexes.
Anyway, for convenience reasons, we think that would be appropriate to insert

- an index for the field “City” of the Place nodes. This is because in several portions of the GUI of our application, we let the user retrieve specific places by city. Without this index, when, for example, the user asks for the most critical places in a given city, the DBMS goes through the whole graph to search for all the nodes of type Place with the correct City field, wasting lots of time and computational resources. With the index, instead, the DBMS immediately retrieves all the places belonging to the specified city directly, almost without any searching task.
  To create this index we can process the following query:
  `CREATE INDEX city_index FOR (a:Place) ON (a.city)`
- a constraint of type “UNIQUE” on the place name. This will also automatically create an index on that property. In this way we both ensure uniqueness and performances in all the queries that search for a specific place by its name.
  To create this index we can process the following query:
  `CREATE CONSTRAINT nameIsUnique ON (n:Place) ASSERT n.name IS UNIQUE`

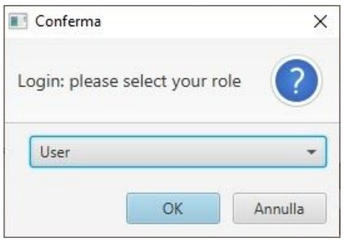
Indexes, as we know, are going to decrease performances during heavy write operations, but in this case it is not a big deal, because the insertion of new place(s) is sporadic.
Of course, due to lack of data in our dataset, we can not make a “real” and documented study for this indexes, but as we already explained, there are mandatory reasons that make these indexes very useful for this application.

8 - User Manual

8.1 - Login for normal user and administrator one (delegated person)

In the application a user can log as a:

- *"User"* : normal user which can compute and visualize statistical information
- *"Administrator (delegated person)"* : the user which can add new information in the database or in general control the system.

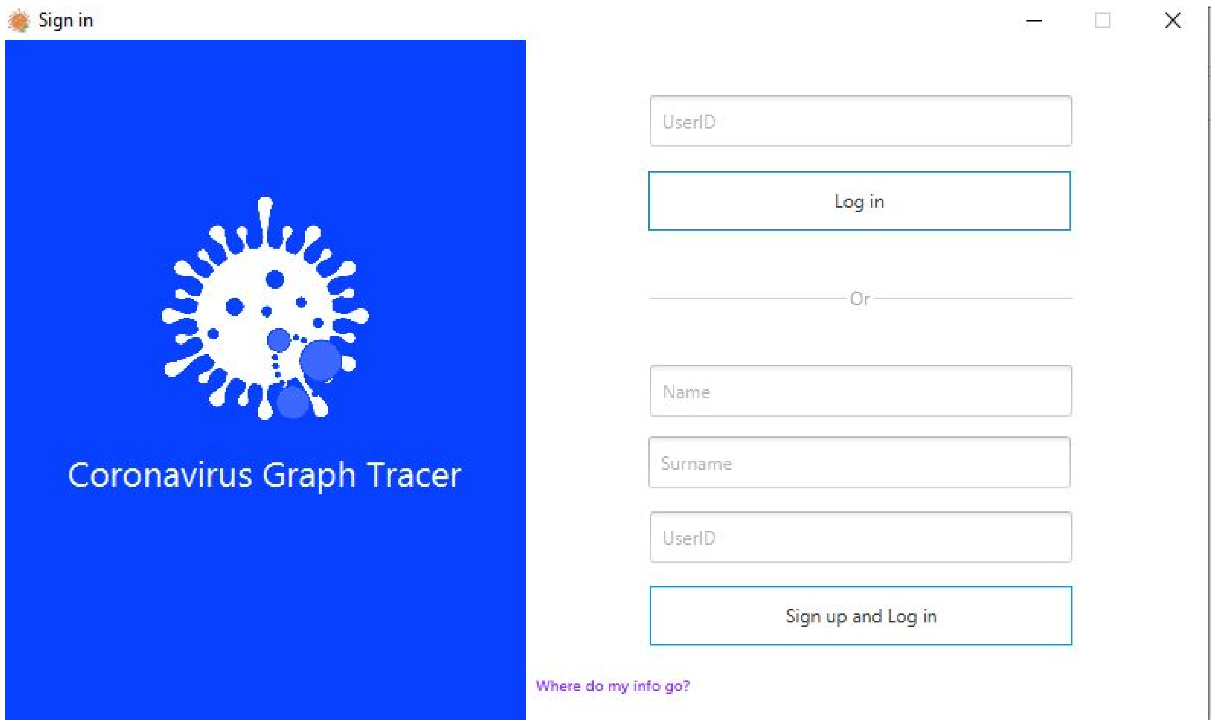


The login window

In order to login, a user has to select between *"User"* or *"Administrator"* with the dropdown menu, and then press the button *"OK".*

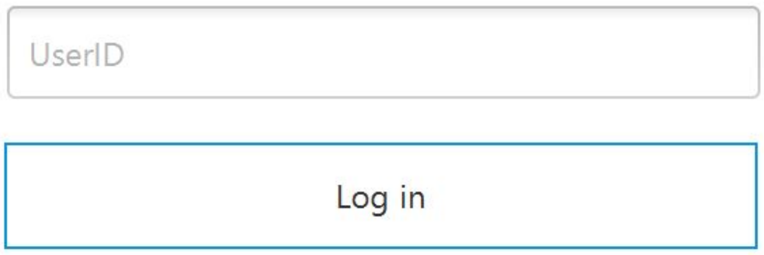
8.2 - User

After the previous window, a new one is displayed.



"The user panel"

8.2.1 - User login



"The login part of the login and register window"

In order to login, the user has to insert his *UserID* in the form and then press to *Log in* button. After that the user panel window will appear and the user is now logged in the service.

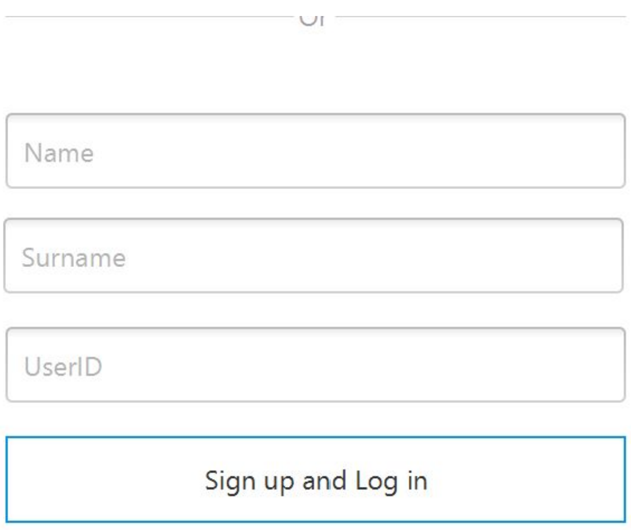
8.2.2 - User registration and login

The registration procedure is composed by two parts:

1. registration of the new user.
2. creation of a new house in the system or selection of an existing one.

8.2.2.1 - Registration of a new user

In order to register itself, the user has to complete the registration form and then press to the "*Sign up and Log in"* button.

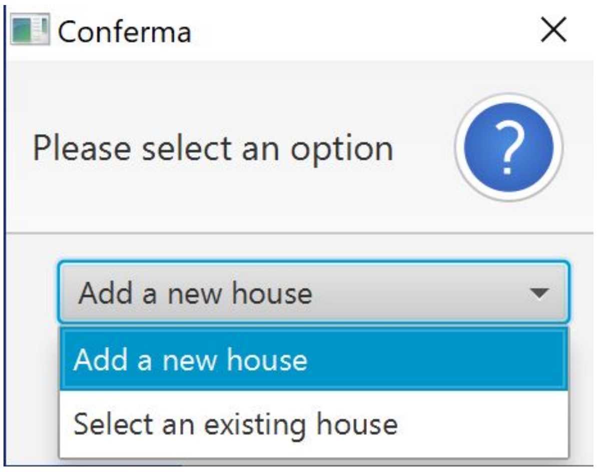


"The registration part of the login and registration window"

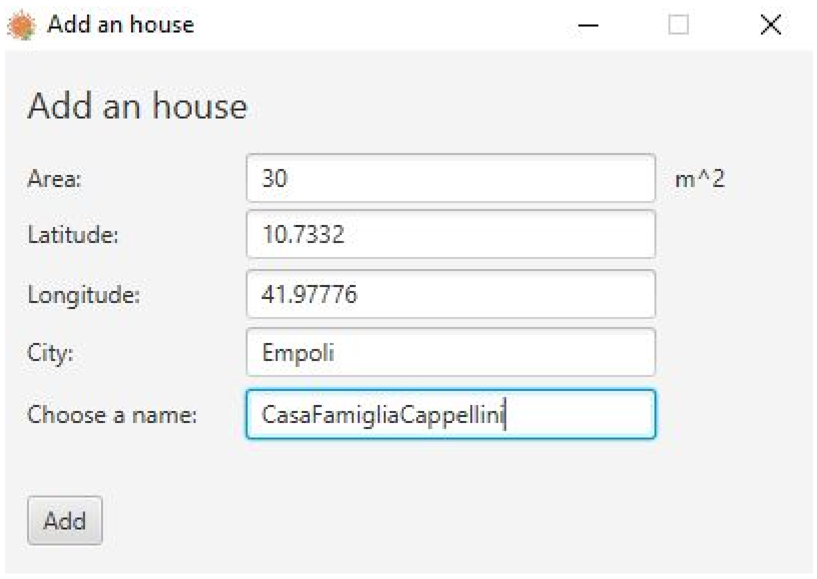
After clicking to the button a new window will appear. This window asks to the new user if want to create a new house or selected an already existing one in the system.

8.2.2.2 - Creation of a new house

If the new user want to create a new house he has to select "Add a new house" and then press to the OK button.
In the new window the user has to complete the form with the house information and then press to the "Add" button.



"Selection window"


Add an house

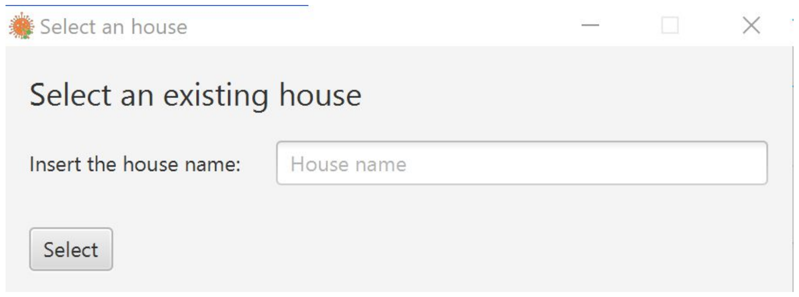
Add an house

Area:	30	m^2
Latitude:	10.7332	
Longitude:	41.97776	
City:	Empoli	
Choose a name:	CasaFamigliaCappellini	

"Add new house window"

8.2.2.3 - Selection of an existing house

If the new user want to select an existing house in the system he has to select "Select an existing house" and then press to the OK button.
In the new window the user has to insert the house name and then press to "Select".


Select an house

Select an existing house

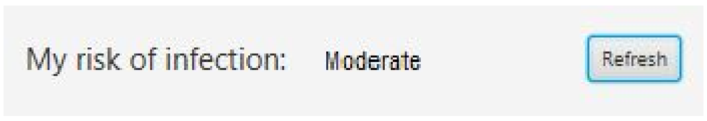
Insert the house name: House name

"Selection of a existing house window"

After the registration, the success window will appear. Pressing the OK button the user panel window will appear and the user is now logged in the service.

8.2.2 - User risk of infection

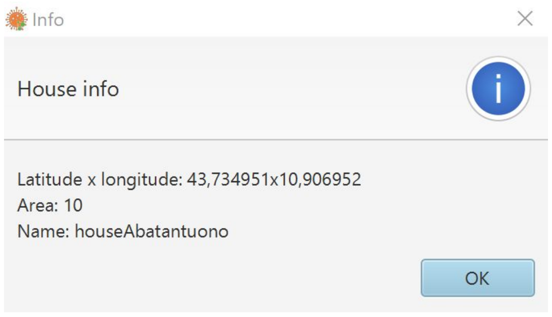
In order to see his own risk of infection, the user has to login and in the user panel window he has to press to the button "Refresh". After that his risk of infection will be shown at the left of the bottom.


My risk of infection: Moderate

"The user's risk of infection"

8.2.3 - User house information

In order to see his house information, the user has to login and in the user panel window he has to press to the button “Show house info”. After that a new window will appear with the house information.

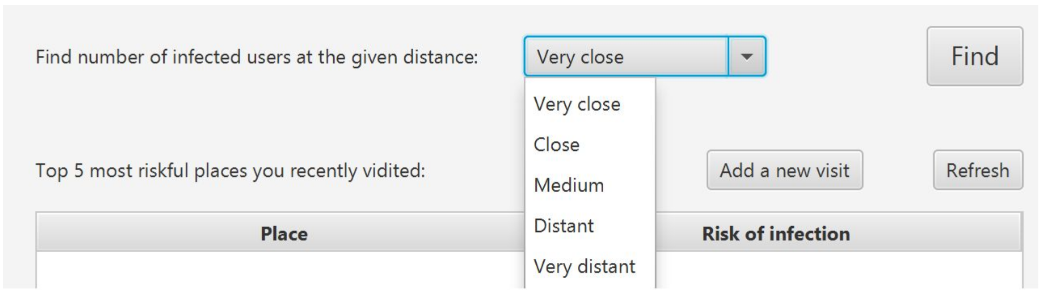


Latitude x longitude: 43,734951x10,906952
Area: 10
Name: houseAbatantuono

“House information window”

8.2.4 - Find the number of infected people at the given distance

In order to see the number of infected people within a certain distance, the user has to login to the service. Then In the user panel window he has to select the distance between the options and then press to the “Find” button.



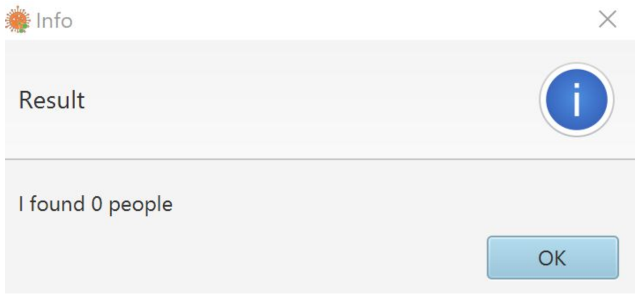
Find number of infected users at the given distance: Very close

Top 5 most riskful places you recently vidited:

Place	Risk of infection

“Select the distance window”

After that a new window will appear with the number of infected user within the selected distance.

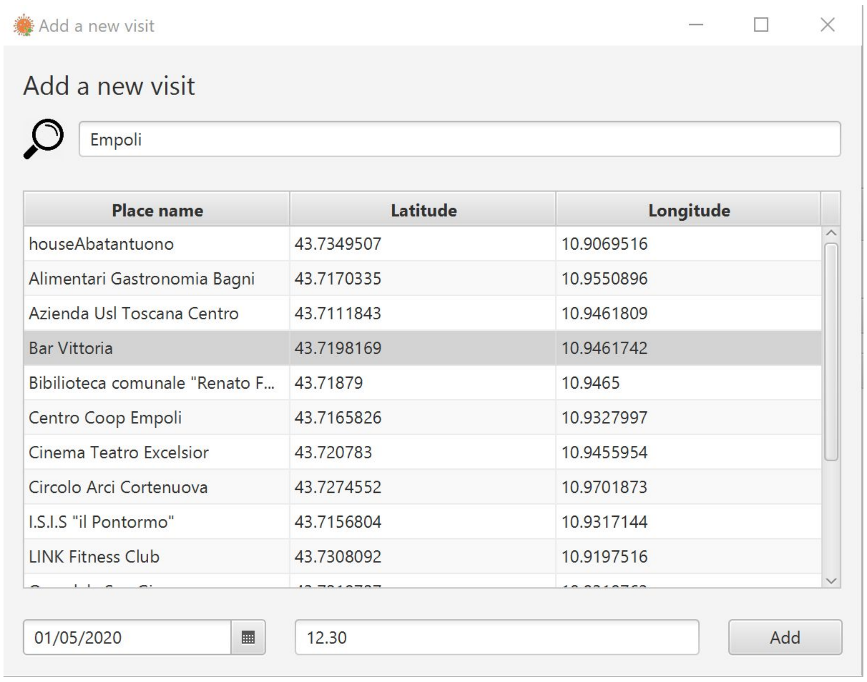


I found 0 people

“The result window”

8.2.5 - Add a new visit

In order to see the infection risk in a specified city, the user has to login to the service. After that he has to press the button “Add a new visit” and then a new window will appear.
In the new window he has to insert the name of the city and press “Enter” on the keyboard. The table will be updated with all the places of the specified city . In order to insert a new visit in a specified place, the user has to press on this place in the table, and then insert the date and the time of the visit.

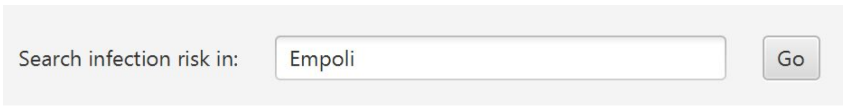


Place name	Latitude	Longitude
houseAbatantuono	43.7349507	10.9069516
Alimentari Gastronomia Bagni	43.7170335	10.9550896
Azienda Usl Toscana Centro	43.7111843	10.9461809
Bar Vittoria	43.7198169	10.9461742
Bibilioteca comunale "Renato F...	43.71879	10.9465
Centro Coop Empoli	43.7165826	10.9327997
Cinema Teatro Excelsior	43.720783	10.9455954
Circolo Arci Cortenuova	43.7274552	10.9701873
I.S.I.S "il Pontormo"	43.7156804	10.9317144
LINK Fitness Club	43.7308092	10.9197516

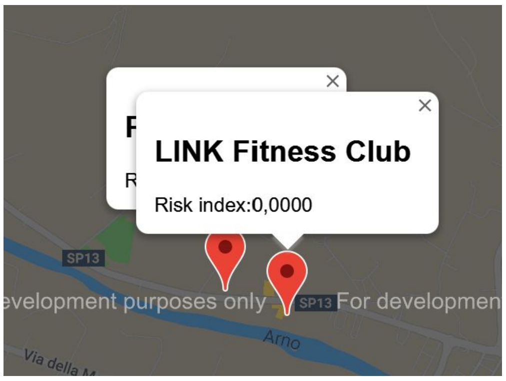
“Add a new visit window”

8.2.6 - Search infection risks in a selected city

In order to see the infection risk in a specified city, the user has to login to the service. Then he has to insert the city name in the form and press the “Go” button. After that in the map window will appear all the places as markers with the infection risk information in their popup.



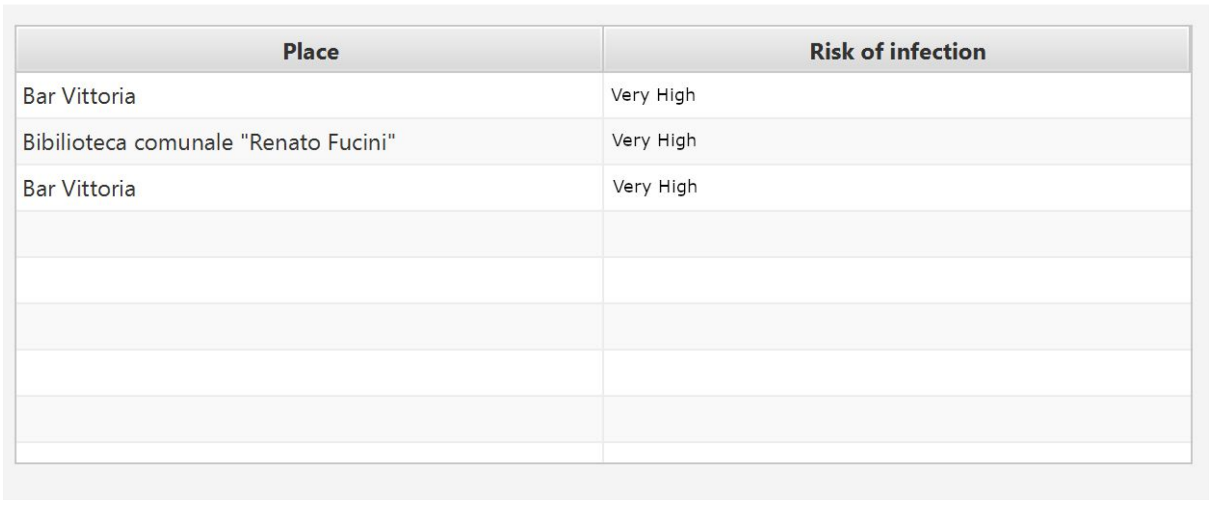
“The form in which insert the city”



“The map with the places marker and the information popup”

8.2.7 - See the top X critical visited places

In order to see the top X critical visited places, the user has to login to the service. The ranking of the most critical places will be displayed to the table.

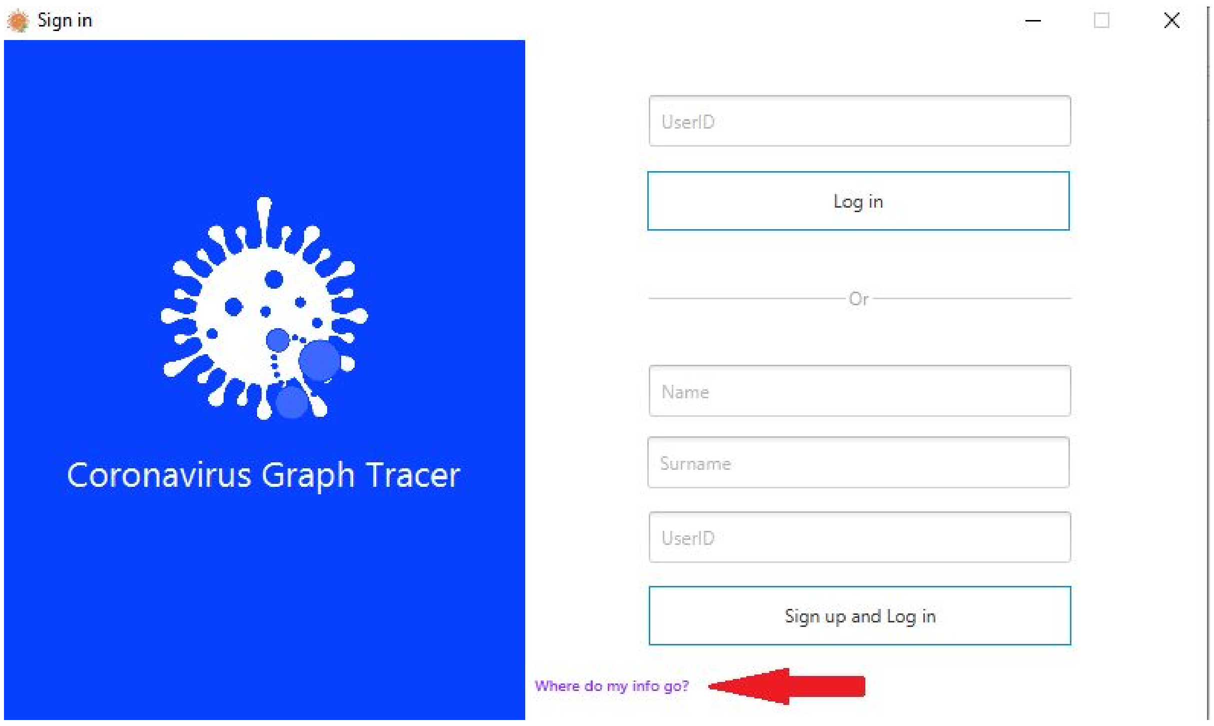


Place	Risk of infection
Bar Vittoria	Very High
Bibilioteca comunale "Renato Fucini"	Very High
Bar Vittoria	Very High

“The list of most critical places with their risk infection”

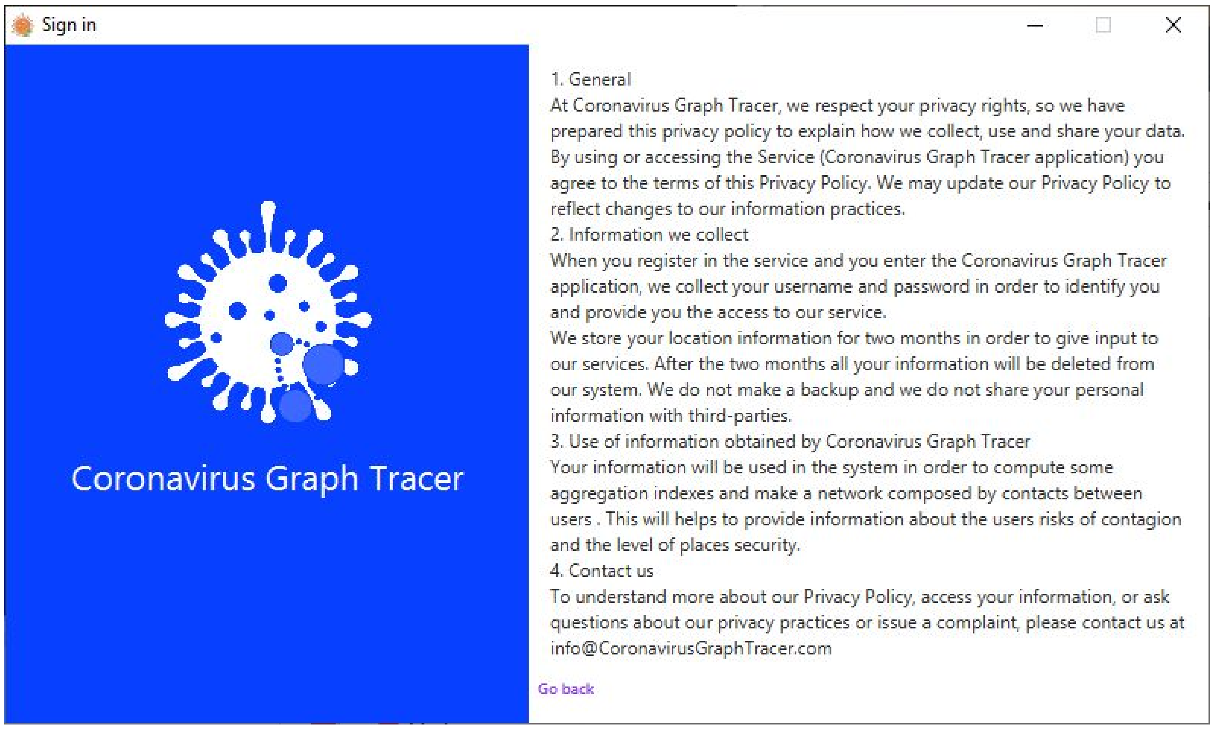
8.2.8 - See the privacy information

In order to see the privacy information the user has to press to “Where do my info go?” button in the first user panel



"The user panel with the privacy button in evidence"

and then a new window will appear with the privacy information.



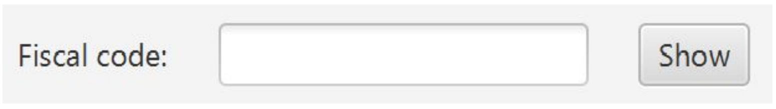
1. General
At Coronavirus Graph Tracer, we respect your privacy rights, so we have prepared this privacy policy to explain how we collect, use and share your data. By using or accessing the Service (Coronavirus Graph Tracer application) you agree to the terms of this Privacy Policy. We may update our Privacy Policy to reflect changes to our information practices.
2. Information we collect
When you register in the service and you enter the Coronavirus Graph Tracer application, we collect your username and password in order to identify you and provide you the access to our service.
We store your location information for two months in order to give input to our services. After the two months all your information will be deleted from our system. We do not make a backup and we do not share your personal information with third-parties.
3. Use of information obtained by Coronavirus Graph Tracer
Your information will be used in the system in order to compute some aggregation indexes and make a network composed by contacts between users . This will helps to provide information about the users risks of contagion and the level of places security.
4. Contact us
To understand more about our Privacy Policy, access your information, or ask questions about our privacy practices or issue a complaint, please contact us at info@CoronavirusGraphTracer.com

"The privacy information window"

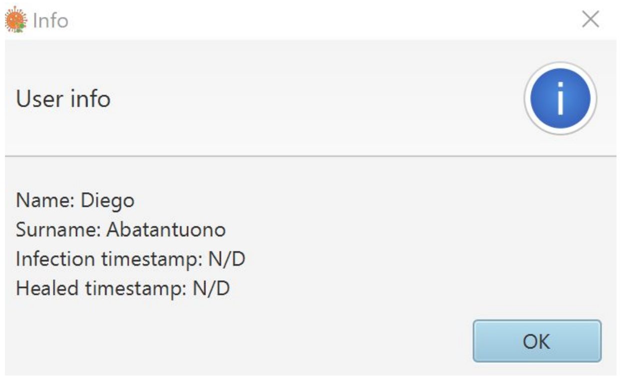
8.3 - Delegated Person

8.3.1 - See a user information

In order to see a user information, the administrator has to login and press to the “Update User” option in the navigation menu.
Then the administrator has to insert the userID in the “Fiscal code” form and press to the “Show” button.



“The Fiscal Code form”

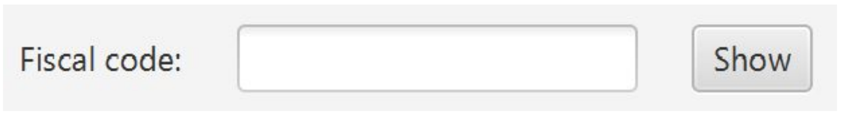


Name: Diego
Surname: Abatantuono
Infection timestamp: N/D
Healed timestamp: N/D

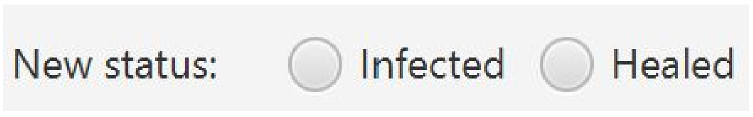
“The user information window”

8.3.2 - Change the user current status

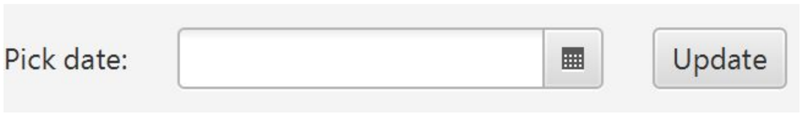
In order to see a user information, the administrator has to login and press to the “Update User” option in the navigation menu.
Then the administrator has to insert the userID in the “Fiscal code” form, choose the correct option between “Infected” and “Healed” in the “New status” menu and pick up a date.
At the end the administrator has to press to the “Update” button.



“The Fiscal Code form”



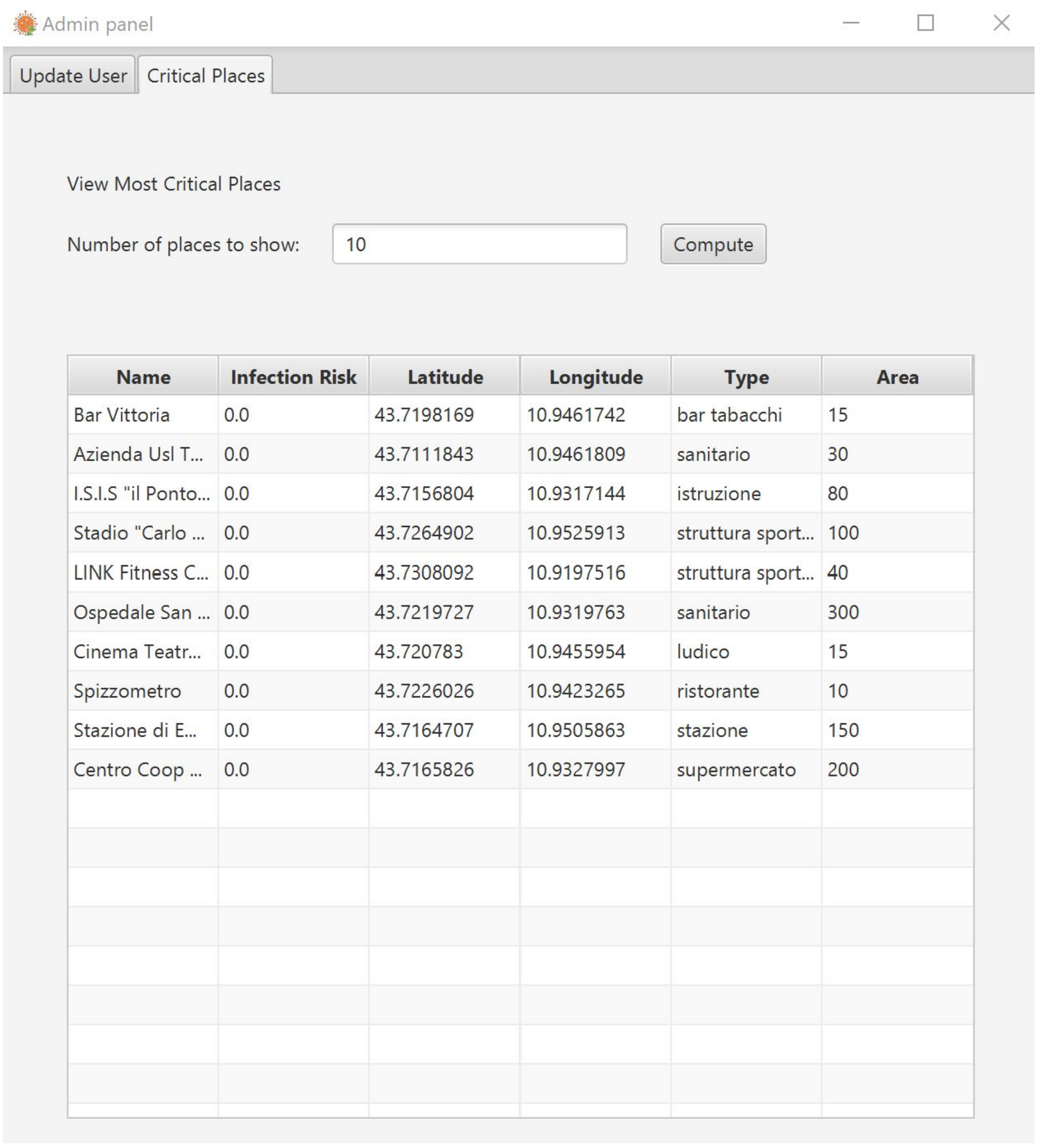
“The New Status menu”



“The pick up date menu”

8.3.3 - View the most critical places in the system

In order to see a user information, the administrator has to login and press to the "Critical Places" option in the navigation menu.
Then he has to insert the max number of critical places the administrator wants to see and press to "Compute". After that the table will update with the list of critical places.



Name	Infection Risk	Latitude	Longitude	Type	Area
Bar Vittoria	0.0	43.7198169	10.9461742	bar tabacchi	15
Azienda Usl T...	0.0	43.7111843	10.9461809	sanitario	30
I.S.I.S "il Ponto...	0.0	43.7156804	10.9317144	istruzione	80
Stadio "Carlo ...	0.0	43.7264902	10.9525913	struttura sport...	100
LINK Fitness C...	0.0	43.7308092	10.9197516	struttura sport...	40
Ospedale San ...	0.0	43.7219727	10.9319763	sanitario	300
Cinema Teatr...	0.0	43.720783	10.9455954	ludico	15
Spizzometro	0.0	43.7226026	10.9423265	ristorante	10
Stazione di E...	0.0	43.7164707	10.9505863	stazione	150
Centro Coop ...	0.0	43.7165826	10.9327997	supermercato	200

"The Critical Places panel"

8.4 - Configuration parameters

The application uses a default configuration that is stored, as a JSON file named “configuration.json”, inside the application JAR, under the folder “otherConfigurations”.
This configuration can be modified by the user, and in particular he can modify the following values:

- VALIDITY_PERIOD: period in milliseconds during which the visit of the user in a place is considered valid for the related computations. By default it is 2 weeks.
- USER_MOST_CRITICAL_PLACES_NUMBER: the number of places that are shown to the user when he asks for the most critical places visited by him. By default it is 5 places.
- DISTANCE_LOOKUP_TABLE: it is the lookup table that translates the strings related to the distances, to the number of hops.
- INFECTION_RISK_LOOKUP_TABLE: it is the lookup table that translates the number of hops to reach an infected person, by the user, to a more interpretable string, that is a human-readable indication of the infection risk of the user.
- SERVER_UPDATER_INTERVAL: it is the number that define the period of time between two consecutive updates of the risk of infection, computed for every node of type Place in the database (except the ones related to the houses of the users).

The default “configuration.json” file content is:

```
{
    "VALIDITY_PERIOD": "1209600000",
    "USER_MOST_CRITICAL_PLACES_NUMBER": "5",
    "DISTANCE_LOOKUP_TABLE": [
        {"key": "Very close",   "value": 2},
        {"key": "Close",        "value": 4},
        {"key": "Medium",       "value": 6},
        {"key": "Distant",      "value": 8},
        {"key": "Very distant", "value": 10}
    ],
    "INFECTION_RISK_LOOKUP_TABLE": [
        {"key": "4",         "value": "Very high"},
        {"key": "8",         "value": "High"},
        {"key": "12",        "value": "Moderate"},
        {"key": "18",        "value": "Low"},
        {"key": "MAX_VALUE", "value": "Very low"}
    ],
    “SERVER_UPDATE_INTERVAL”: “14400000”
}
```