UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

HUMAN-IST
RESEARCH CENTER

Gamification and tracking of social interactions at a networking event

Moreno Colombo[1]

Supervisor: Prof. Dr. Denis Lalanne[2]

August, 2015

Department of Informatics - Bachelor Project Report

Département d'Informatique - Departement für Informatik • Université de Fribourg -
Universität Freiburg • Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland

phone +41 (26) 300 84 65 fax +41 (26) 300 97 31 Diuf-secr-pe@unifr.ch http://diuf.unifr.ch

[1] moreno.colombo@unifr.ch, University of Fribourg
[2] denis.lalanne@unifr.ch, University of Fribourg

Acknowledgements

- I wish to express my sincere gratitude to **Prof. Dr. Denis Lalanne** for his precious advices and for guiding me in the choice of a really interesting, challenging and motivating subject for my bachelor project.
- My sincere thanks goes also to the Swiss Integrative Center for Human Health (SICHH)[1], especially **Geneviève Joullié**, **Dr. Jean-Marc Brunner**, **Dr. Jan Kerschgens** and **Dr. Mark Ambühl** for bringing together the required hardware, and for the opportunity of testing a first version of the project in an interesting environment.
- I thank **Thomas Rouvinez** for providing me a good quality first version of ITSI to work on, and for his guidance and advices to get the project started.
- I am also grateful to my friends **Viola** and **Anaïs** for helping me in the revision of this paper.
- Last but not least, I would like to thank my family: my parents and my brother for supporting me throughout the writing of this thesis and all of my studies.

[1] `http://www.cish.ch/en`, August 2015

Abstract

With the goal of increasing social interactions and simplifying the search of people in a room during a medium-sized networking event in mind, this project proposes a way of tracking people and detecting the interactions between them.

The proposed system tracks people by zones, using passive RFID technology, where each person has to wear a non obtrusive RFID tag to be tracked. This allows all of the other users to see, on a big screen and on their own devices with internet connection, the position of all of the participants, so that they can immediately be found.

Moreover, the social interactions between people can be analysed and used for gamification, that is the creation of a leaderboard of the users with the best behaviour for the success of the event and reward them based on the score they get. This encourages users to participate more actively to the networking event, increasing its success.

Another utility of position and social interactions tracking is the fact that some analytics can be done after the event, to help organizers having a detailed feedback on the success of the event.

In this project all of this is successfully implemented, with a good overall performance.

Keywords: Gamification, indoor tracking, RFID, visualization

Contents

List of Figures

1 Introduction

1.1 Goals

Sometimes, the fact of not knowing perfectly what happens around us, limits the efficiency of our actions. For instance, there always is the need of exploring the places we visit, before knowing exactly what to do. This can be done directly (e.g. by walking around) or indirectly (e.g. by using a map). In most of the cases, there are different ways of getting the necessary informations to optimize our behaviour, and all of them have some advantages/disadvantages. For example, by exploring an environment through a map, we can only get a static view of a place, but it is at the same time more efficient than walking around the room from the point of view of time consumption and physical effort, even if this second method gives a more complete idea of what's going on in our surroundings.

The idea of this bachelor project consists in providing to users attending a networking event a method to get the most of informations which can help them to optimize their behaviour. This method tries to be the one with the highest possible ratio between advantages and disadvantages for getting good quality informations, and is easily extensible to events with a different structure (not necessarily networking events), with few or no modifications.

For people attending a networking event, some optimizations can be done in terms of: time, physical effort (avoidable displacements) and quality and number of social interactions.

This project aims to achieve all of these points by providing: a fast way for users of finding people one is interested in speaking with, a method which awards people with the "best" behaviour for the success of the event, some statistics to have a better idea about all of what is happening at the event.

The main technology which has been chosen to be used for achieving these goals is passive RFID (Radio-Frequency IDentification), where each participant of the event has to wear a small chip (RFID tag) of the size of a credit card to be tracked by some antennas. The data collected by the antennas is then elaborated in a way allowing us to get all of the informations we want.

1.2 Context

This project is mainly based on and extends the idea developed by Thomas Rouvinez[2] in his master thesis[1], which is a good reading for anybody who wants to get some complementary informations to what is exposed in this document.

So the work started with the application developed by Thomas Rouvinez as a base, whose functionalities, structure and problems are exposed in this section. From now on, we'll call the base application ITSI 1.0 (Indoor Tracking of Social Interactions), while the new developed one is ITSI 2.0 .

1.2.1 Functionalities

ITSI 1.0 uses the same technologies as ITSI 2.0 , as described in the previous section, so it uses RFID passive tags to track the position of people. Some RFID antennas are placed in a room, and based on the detection of a card at an antenna, one can estimate the position of a person. In ITSI 1.0 , this is used to show on a big screen some real-time informations about the position of all of the participants in the room; in this way, everybody can have direct access to these informations.

Since this application is thought mainly for a big event with a lot of attendants, showing at each instant the position of all of the people on a map could be too complicated to be read. Moreover, what we can detect is only the proximity of a person to a certain antenna; so trying to show the exact position of a person would be incorrect and could lead to confusion.

Because of this, the choice for the live map visualization has been to divide participants in groups based on their interests, and then show them on the map as a slice of pie around the antenna where they have been detected. Graphically, this translates to what can be seen on Figure 1; where the black discs are the antennas, and the slices around them are each a different detected user, with a color corresponding to their interest.

[2] thomas.rouvinez@unifr.ch

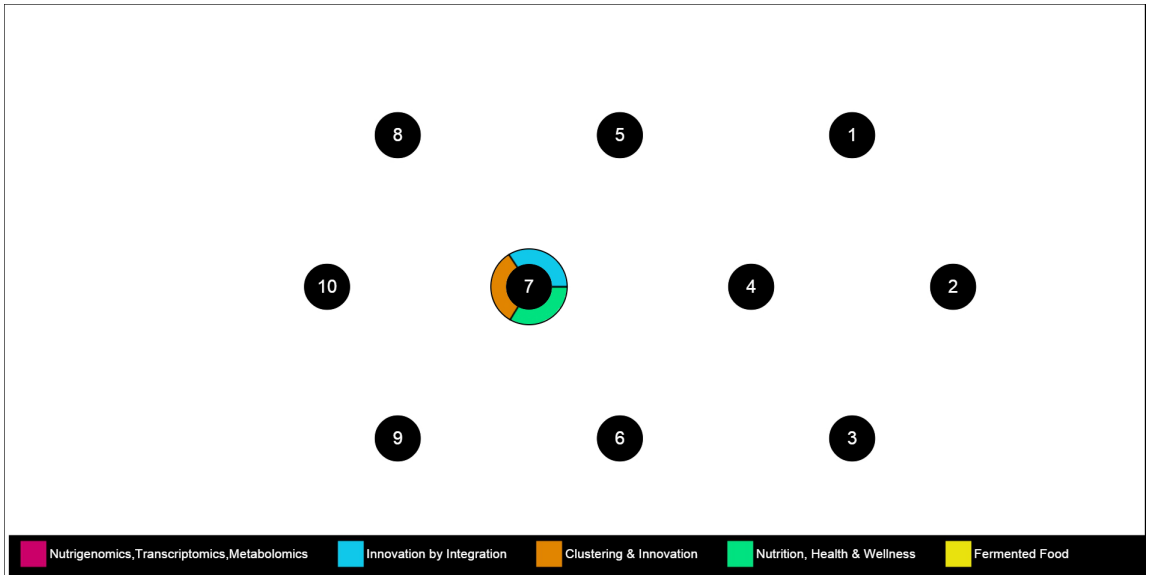


Figure 1: First live visualization. [1]

All the recordings of people are stored in a database, in such a way that the collected data can be used for doing some after-event statistics. Visual analytics can also be done, but it has to be done "manually", since the developed application for this task consists mainly in just returning a list of all of the recordings as text, which have then to be manually introduced in a software for visualizing them in an easy, human-readable and meaningful way.

1.2.2 Structure

The structure of ITSI 1.0 consists in some RFID readers connected to a controller, which sends all of the recorded (and crunched) data to a database via a web application. Other web applications give access to the data saved in the database, allowing live visualization, post-event analytics and direct edition of the database's content. The structure is shown in detail in Figure 2.

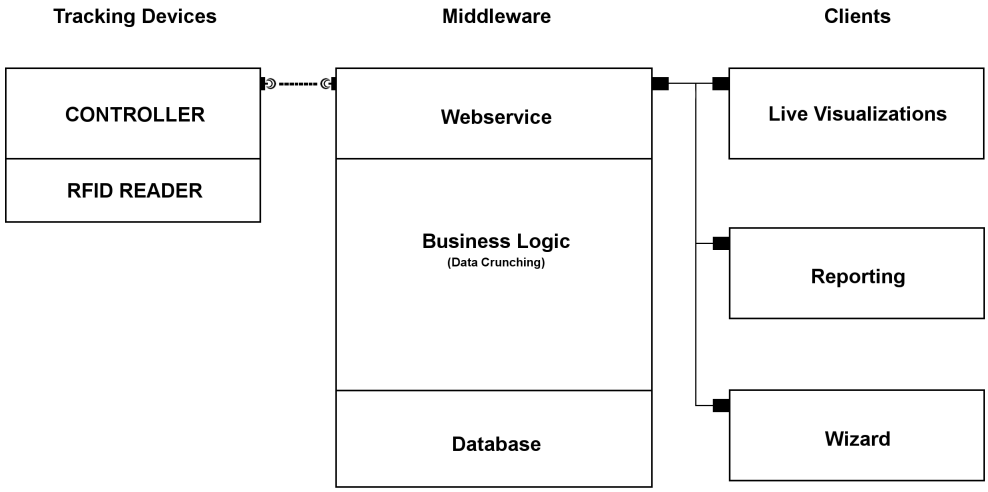


Figure 2: Structure of ITSI 1.0 .[1]

1.2.3 Problems

The application described in Section 1.2.1 has been used during a real event, this has shown some problems reducing slightly its quality. The three main encountered problems are the following:

Detection problems: Due to the orientation of RFID tags, those are sometimes not detected by antennas, even if in the range of detection. So, if during a reading loop a tag is not read, it results that the participant has left the zone where the reader was for a reading cycle. This creates some holes in the recorded data, but also in the live visualization, where this results in a visualization changing really often with tags disappearing and reappearing (in the worst case at each reading loop), and people not shown when they should be.

Delay in live visualization: Another problem for live visualization, is the fact that, despite the name, it is not really LIVE. In fact, there is a quite long delay between the moment a tag is detected next to an antenna, and when it is shown on the map. The delay is not even constant but varies randomly between more or less 10 and 25 seconds. This long delay is really significant, since it could possibly keep users from finding each other, since one can see on the big screen someone next to an antenna, when he is already gone from a quite long time.

Cable management: As seen in Figure 2 the readers are each connected to a controller, which sends collected informations to a server. But in praxis, the chosen way to do this has been to connect all of the readers to a central computer emulating all of the controllers. The problem with this solution is that a lot of cables have to cross the room where the event is hosted, which is not an optimal solution for the ease of installation, and probably also for the comfort of participants.

1.3 Fixes

Since there are some problems in ITSI 1.0 , the main part of this bachelor project consists in solving the three principal problems explained in Section 1.2.3. To do this, the first step is trying to find the cause of these errors. After this, one has to ideate a way of solving these problems. In the end the found solution has to be implemented and tested. This leads to some big changes in the structure of the application and to an almost complete reimplementation of the application, due to the major changes.

1.4 Extensions

The second part of this bachelor project consists in doing some interesting and useful extensions to the old application. This lead to the introduction in ITSI 2.0 of gamification, people's positioning search, live individual statistics, and automatized post-event analytics. More details on these extensions can be found in Sections 2.2 and 5.

1.5 Structure of report

In this report there will be at first a section describing the design of ITSI 2.0 , with its hardware and software setup. This is followed by a section describing the chosen tracking system using RFID technology. Then all what happens on the server side, notably database access, data elaboration and gamification, is exposed. Another important part of the application consists in the interfaces through which users access informations, so a section is dedicated to the description of these interfaces. At the end of the report there is an evaluation of the performance of the system, in such a way that one can see the improvements done since ITSI 1.0 and judge the quality of ITSI 2.0 .

2 Design

After having analysed the problems described in Section 1.2.3, notably the delay in live visualization and cable management, one of the ideas to solve this has been to change slightly the design of the system, on both hardware and software sides, even if the base structure (see Figure 2) of ITSI 1.0 has been maintained also in ITSI 2.0 .

2.1 Hardware setup

The used hardware consists physically in some RFID sensors with a quite limited range of detection, a controller for each of the sensors, which sends detected data to a server elaborating the received informations and storing them in a database.

This setup allows us to solve the problem of having a lot of cables crossing the room. In fact, each RFID sensor can be connected to a small controller (usually a microcomputer) with a short USB cable, and they can be both powered by a battery. They can be then put together in a box in such a way that the visual impact of the apparatus is the less obtrusive possible. From a technical point of view, each controller collects data from the sensor, and sends it periodically via WLAN, through a socket, to the server. The server, which has to be connected to the same WLAN as the controllers, listens for the transmitted informations and elaborates them.

Since the chosen way to track people's movements and interactions is by zones (see Section 3.1), we chose to take advantage of the usually already present small tables, around where people regroup to talk during networking events. So we chose to put a sensor on each table, in such a way that everybody can see where the detection zones are (about 1m around each table's center) and that people doesn't have to change their usual behaviour, since usually people tend to go next to a table when talking with someone else.

2.2 Software architecture

Some different programming languages and frameworks have been used in this project for the different tasks. The two main frameworks it has been taken advantage of in ITSI 2.0 are Django and D3.js.

DJANGO[3] is a Python Web[4] framework which helps creating web applications in a fast way and by preserving a clean and secure design. It handles databases, automatically gets a way of accessing and editing them, manages links between pages, helps creating secure applications etc. This framework has been chosen to simplify database access and management, data elaboration and with the scope of speeding up the implementation of the web service.

D3.JS[5] is a Javascript drawing framework. It is a really good framework for data visualization, it in fact transforms really easily collections of data to SVG images displaying the collected data in a significative and easily human-interpretable way. This framework has been chosen to automatize post-event analytics and to visualize live statistics via web applications.

In Figure 3 one can see the software architecture of ITSI 2.0 .

The main difference between ITSI 2.0 and ITSI 1.0 lies in the structure around live visualization. In the first version, in fact, the live visualization part always took the last values registered in the database to show people's positions. Since the database access caused a delay in the visualization, it has been decided to delete this step and to visualize directly the map, by using the data recorded by sensors, after having filtered it a little bit. So, by removing this step, the delay problem for live visualization has been fixed.

[3] `www.djangoproject.com`
[4] `www.python.org`
[5] `www.d3js.org`

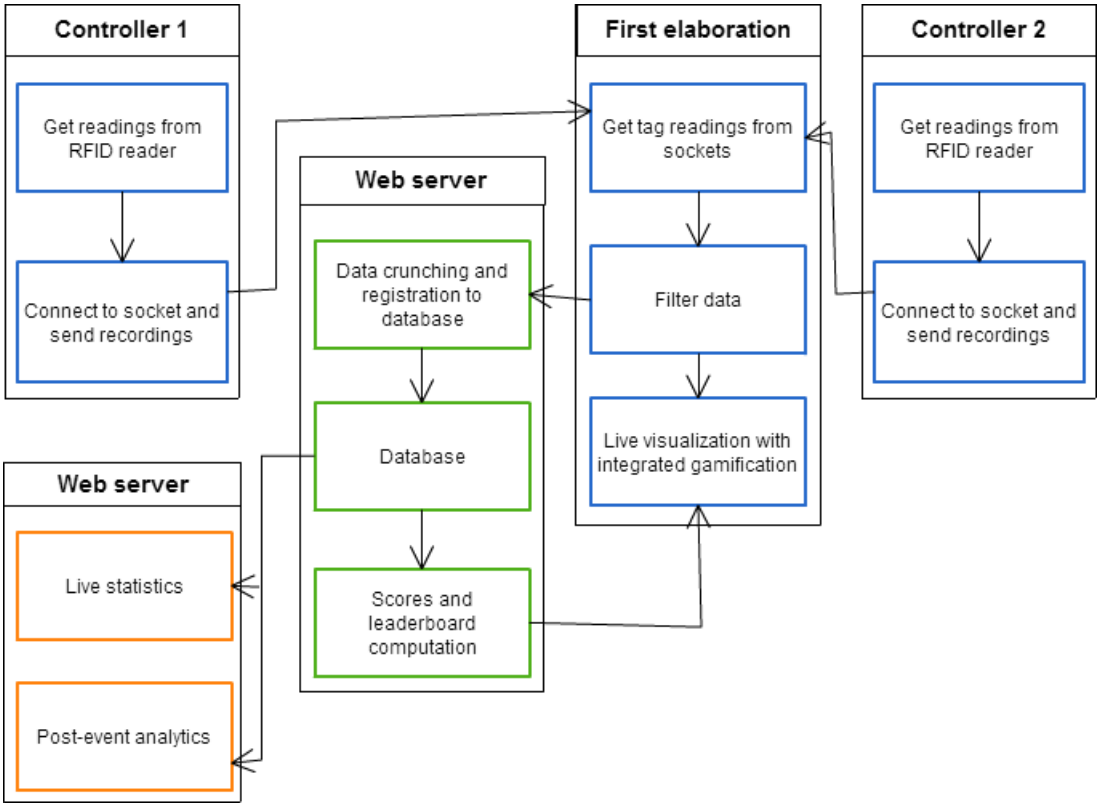


Figure 3: Software architecture of ITSI 2.0

3 RFID-based tracking system

As seen in Section 2, the structure of ITSI 1.0 and ITSI 2.0 are basically the same and divided in 3 sections (see Figure 2). In this section, the focus is on the first tier of that structure, so on the indoor positioning system using RFID readers and the tracking of people.

We first describe the choices, in terms of IPS (Indoor Positioning System) type and hardware, and then how the controller handles and sends the data to the central server.

For a more complete description of RFID technology, see Thomas Rouvinez's master thesis [1].

3.1 IPS choices

RFID technology for IPS can be used mainly in two ways:

Zoning - A lot of RFID sensors with a quite small detection range are placed in a room, and the position of tags read by these sensors is determined on the base of the zone the antenna covers. So this method has the disadvantage that it is not so precise to track the exact position of people, but a good precision is not always needed in some applications.

Multilateration - Using 3 or more RFID sensors with a big detection range (a tag always has to be detected by at least 3 sensors at the same time), one can determine the exact position of a tag in the room by using the power of the signal received by the sensors. This can be a really precise IPS, but it is a little bit more complicated than zoning.

In this project we chose to use passive RFID technology, since it is less obtrusive for participants (we want them to wear the less technology possible) and because passive RFID tags don't require batteries, which could empty in the middle of the event.

Mainly because of this choice, the selected IPS is by zoning. In fact, after having tried both of the proposed IPS, it has been confirmed that multilateration using passive RFID tags is not

precise at all, because of the easily variable strength of the signal emitted by the tags and the quite low range of this technology.

Moreover, this IPS lets people be aware that they can be detected only next to stand-up tables. So when someone doesn't want to be tracked, he only has to go far from a table.

3.2 Hardware choices

For ITSI 2.0 , the chosen RFID readers have to support passive RFID technology, have a range of detection of about 1m, a good number of readings per second, have the possibility to connect to any standard computer without the need of adapters, and should be as small as possible.

So the choice has been the THINGMAGIC USB PLUS+ RFID Reader (See Figure 4).

Figure 4: THINGMAGIC USB PLUS+ RFID Reader

This reader reads UHF Class 1 Gen 2 (ISO 18000-6C) tags. The read range is slightly variable, but it reaches up to 1 meter (sometimes also a little more) and reads up to 200 tags per second, which is enough for our case. For the connectivity, it uses an emulated COM port interfaced with USB 2.0. This allows to connect to any computer with an USB port. Moreover, the reader is small (97 x 61 x 25 mm), so it shouldn't disturb participants, when put in key locations to track their position.

On the other side, controllers must have at least 2 USB ports, have a WiFi connection, support JAVA applications, have the possibility to be powered by a battery, and have the smallest possible form factor. For these reasons battery powered Raspberry Pi (see Figure 5) has been chosen as controller.

Figure 5: Raspberry Pi

Raspberry Pi is a credit-card sized computer with enough USB ports for one RFID reader and to host an USB WiFi dongle. Linux can be installed on this computer, so it also supports JAVA.

3.3 Readers controllers

As already said in the previous section, the reader controllers are small computers with a battery and internet access. They are supposed to initialize sensors, continuously read data received from the sensors, send this data to a server and to correctly stop tag reading at the end of a recording session.

The work of controllers is really simple: after having connected themselves to the RFID readers and to a socket, the controllers start a loop. During this loop, the readers register all of the RFID tags in their range of detection in an array, the controllers transform it in a JSON object which is sent to the socket, in such a way that it is accessible by the server. If a termination signal is received from the server, the controllers disconnect from the socket and terminate tag readings in a secure way.

4 Server-side middleware

On the second tier of our structure (Figure 2) there is everything what happens on the server-side. All of this is described in this section.

The focus is posed at first on how data is received by the server from controllers and how it is elaborated. On the second part of this section, the structure of the database is exposed. In the third and last part, there are some considerations about gamification in general and in our particular case.

4.1 Data retrieving and filtering

As said in the previous section, each controller sends periodically informations about the tags who have been recorded in the last reading loop to a socket. So the server has to get those informations from the socket, this means it has to connect to a number of different sockets corresponding to the number of readers. To avoid problems, there is the same number of threads running on the server as the number of readers. So each thread can retrieve, filter and elaborate the recordings of a single reader.

Getting informations from controller for the server just consists in accessing the right socket, getting informations as JSON, and finally convert them to JAVA objects.

As seen in Section 1.2.3, sometimes in ITSI 1.0 some not real recordings were shown on the live map, and sometimes people next to a table were only shown once every 2-3 seconds. It is because of this that some filterings have to be done on the raw data received from the controllers. For this first elaboration of data, JAVA is used because of some performance advantages over Python.

The basic idea consists in finding false detections and delete them from the array of all of the readings. To do this, two criteria have been taken into account: the number of tag reads during the last reading loop has to be smaller than 2 (experimental value), and, after this, the number of detections on the last 5 loops has to be of at least 3. This removes from the recordings all of the detections of tags at the wrong sensor (these usually are read at most 2 times per reading loop in the wrong place, or occur only for a single reading loop from time to time), and people detected while just passing by the sensors. Moreover, the second criteria of filtering ensures more smoothness to the live visualization, since if for some exceptional phenomena (actually this happens really often) a tag is not read when it should, this doesn't change the visualization, except if it is not read for at least 3 consecutive reading cycles, which is more likely to mean that the participant has effectively left the table.

4.2 Database

All the recorded and filtered data is both directly used for the live visualization and stored in a database to be used later. We see in this section the structure of the database.

The JAVA objects, created and filtered on the server side, are transformed once again to JSON objects and sent to a Django application, which ensures that the same recording is registered only once per second and then manages the storage to the database.

For the database, SQLite has been chosen because of its light weight, as-is integration in Django applications and simplicity to get started. Despite being a really light database system, it is more than complete and enough efficient for our application.

The schema of this database can be seen in Figure 6.

We now see in detail all of the tables of the database:

Sensor - This table contains all of the registered sensors, with their position inside the room, in such a way that they can easily be represented in the different types of visualizations with a spatial dimension. Each sensor also has an identifier and a description (for example the number of the table they are on). Each sensor also has the possibility to be hidden in the live visualization, so that it can be used just for analytics purposes.

InterestTag - This table contains all of the possible interests choices, with their description and color to be used to display them in the different visualizations. This is used to regroup participants in different categories.

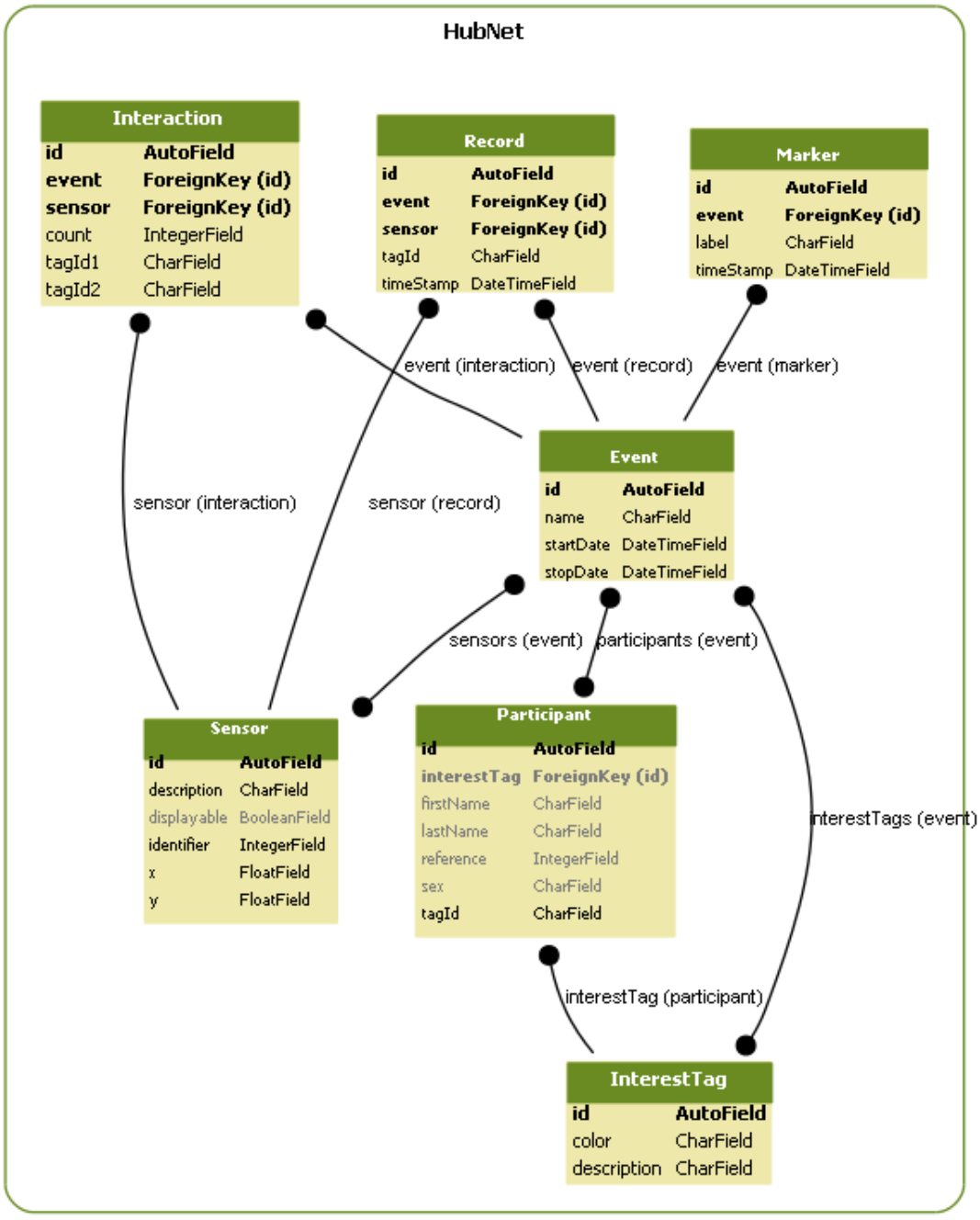


Figure 6: Database schema of the application

Participant - It's the table containing all of the participants registered to any of the events. Each of them has a name, a tag id associated to them (the unique identifier of the RFID tag they are wearing) and a specification about sex, which could be used for statistical purposes.

Event - Each event is a new entry in this table, and has a name to identify it, a start date and time, and date and time of the end of the event.

Marker - It is possible to insert markers in the events, in such a way to split events based on what's happening in each moment. In this way one can analyse and compare different moments and situations of one event. This is associated to an event, and has a description of what's

happening, as well as the time and date at which this "sub-event" started.

Record - At each time one RFID tag is detected by a sensor and passed the filter described in Section 6.3, it is registered in this table, where informations about the event and sensor at which the tag has been detected are stored. Moreover, each record has informations about the tag who has been registered, and the time of detection.

Interaction - To increase the efficiency of interactions analysis (for example to compute scores for gamification), there is, in addition to the Record table, another table where recordings are registered. But in this case, recordings are stored in couples. That is that each time that two tags are detected at the same time at the same sensor, an interaction between them occurred, so an interaction between these two tags is registered in this table. In this case there are fields containing informations about the event and sensor where the interactions occurred and about the two interacting tags.

4.3 Gamification

An important part of this project consists in the introduction of a gamification method for networking events. So this section is about what gamification is and how it is introduced in a networking event.

In our case, gamification is introduced to try to increase the success of a networking event, by stimulating users to interact more, and to interact also with people with interests different from their own ones. So the goal of gamification is clear, but what is it?

> *Gamification is the concept of applying game mechanics and game design techniques to engage and motivate people to achieve their goals. Gamification taps into the basic desires and needs of the users impulses which revolve around the idea of Status and Achievement.*[2]

So, in other words, gamification consists in stimulating people to behave in a better way for the success of the event by rewarding them.

The idea, for this project, is to find a way of integrating gamification in the basic functionalities of the application, without having to change its goals.

Since there already is a live visualization part, where one can locate all of the other participants, and which is visible by everybody in the room; we found that this was the best part of the application to integrate a gamification method.

The initial idea for the live visualization was, for space reasons, to show only the initials of each participant. But gamification requires a way of rewarding people, so the idea has been to write the full names of the 3 participants with the highest scores, to increase their visibility. Moreover, the higher score one has, the more easily it is to be seen in the map. What has been done for having this behaviour will be seen more in details in Section 5.2.

For having the three participants with the highest score be much more visible than the others, and to increase visibility of participants with their score; it is clear we need a way of computing scores. So the formula for computing the scores is the following:

$$\mathrm{P}_{\mathrm{total}} = \#interactions + bonus$$

Where *bonus* is $\frac{1}{2}$ times the number of interactions had with participants of a group (interest) different from the one of the user of who the score is currently being computed.

The scores associated to each participants are then sorted, in such a way that we also have an ordered leaderboard to be used as a base for displaying informations to participants.

5 Client-side interfaces

To complete the application, there must be a way for simple users to visualize the collected data in different ways; since by just retrieving data from the database and returning it as text, it is difficult for human eyes to analyse directly the informations contained in the collected data. Moreover, an event organizer has to have the possibility to access the database to create events, register users, sensors and interests. So a wizard is introduced for administration purposes, and for showing informations to users there are: a live visualization interface (also including gamification), an interface for searching people, and the possibility of having live and post-event statistics.

5.1 Wizard

It is necessary, for administration purposes, to have an interface which allows to easily create, read, edit, and delete events, participants and sensors; and to associate them with each other.

This is easily achieved by just using the default database administration interface given by Django. This is one additional reason for choosing to use Django in this project. In fact, with a tenth of lines of code, we can create a complete yet easy to use administration interface protected by password. An example of the available tables to be edited and of the edition page of an event can be seen in Figure 7. The default Django administration panel shows all of the application's databases and offers all of the basic database operations.

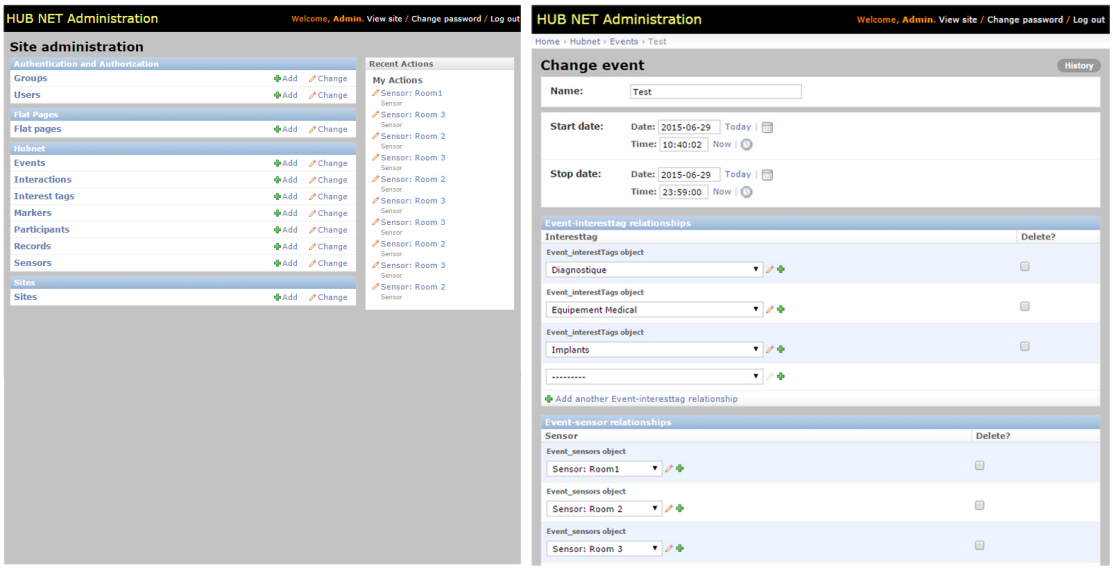


Figure 7: Example pages of the wizard.

5.2 Live visualization

For showing live informations to all of the users about the position of everybody and about the gamification leaderboard, the choice has been to put a big screen in the room where the event is. In this way each participant can see at each time those informations, by just having to look to a visually easily accessible screen.

At the beginning the idea was to just have the exact same visualization as in ITSI 1.0 (see [1]), and to add to it a leaderboard showing the score of all of the different groups of interest (see Figure 8).

The main difference between this and the version of ITSI 1.0 , consists in the used technologies. In fact in ITSI 1.0 the live visualization part has been done using Processing.js, a Javascript framework used for drawings. As we saw in Section 1.2.3, this approach generated a big delay

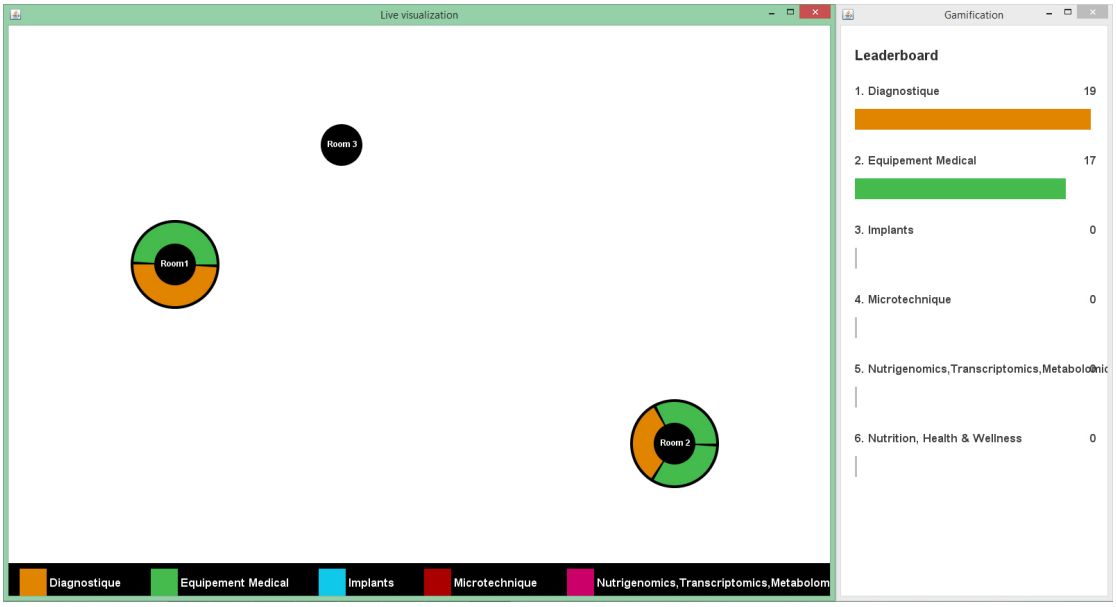


Figure 8: First live visualization with gamification

between registration and visualization of users. So the choice has been to prevent the tag readings to be registered in the database before being retrieved and visualized. This means that right after the data elaboration in JAVA, the retrieved data is directly used by another JAVA application for creating the map of the room showing the tags detected in the last reading loop. Moreover, here has been added a simple gamification interface, where one can see the leaderboard for groups of interests. But this doesn't directly reward people with a good score.

About this old visualization with a simple gamification, we can say that it is quite ineffective, since it isn't personal (gamification speaks about the groups' behaviour, not the individuals'; and visualization doesn't show any name nor identifier for people).

So the idea has been to better integrate the gamification part in the live visualization, by also adding the possibility of rewarding people with a better score with more visibility. In fact, the choice has been to use the computed scores and leaderboard (Section 4.3) to set the size of the "slices of pie" representing people next to the tables. The more an user has a score next to the highest one, the bigger the slice representing him/her. Moreover, initials are shown next to each slice representing a person, and, if the person is in the first three positions of the leaderboard, their first name is shown, increasing their visibility and prestige. There is also a small red triangle in the visualization, which can help people orientate themselves, if the organizers of the event put a similar sign on the corresponding wall. This ameliorated version of live visualization can be seen in Figure 9.

This live visualization is implemented in the same way as the first try of ITSI 2.0 (Figure 8). This means that it is done entirely in JAVA and directly uses tags readings, without having to store them in the database before. At the same time, the scores for gamification are taken from the databases as explained in Section 4.3; this could lead to a small delay, but for gamification's scoring, a small delay is acceptable.

5.3 People search

On the client side, there also is the possibility of having access to some functionalities via a web interface by using any device with internet connection (smartphone, tablet, PC,...). This is really useful for personal searches or personalized informations access on the users' side.

In this subsection, we analyze the case when one is looking for a specific person. In this case, it is not easy to find a specific person on the live visualization screen, because there are a lot of participants and their name are not completely shown, but only their initials. So the idea has

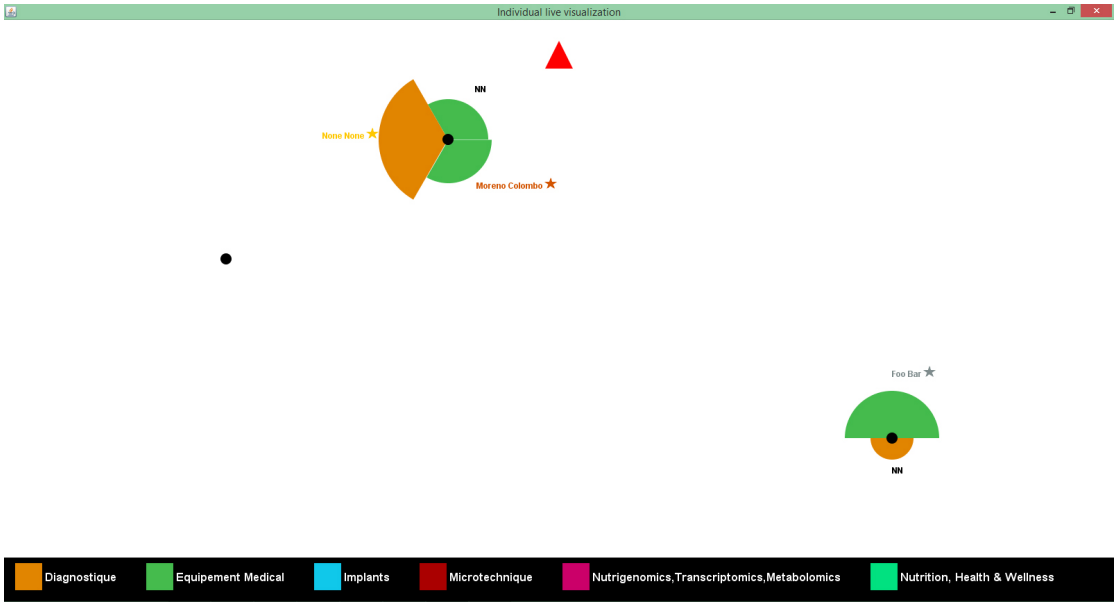
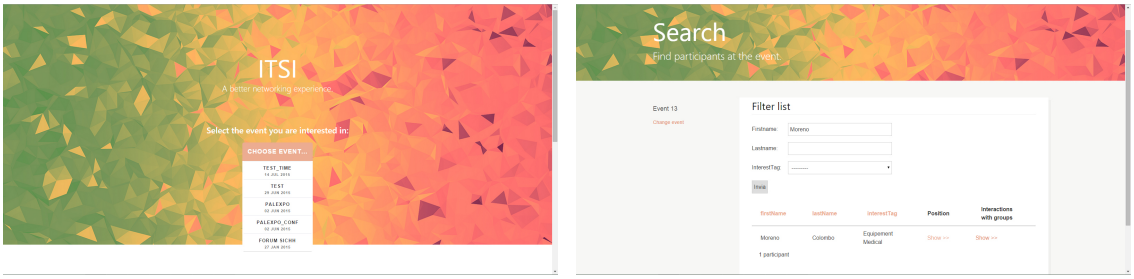


Figure 9: Live visualization with included gamification

been to give people the possibility of looking for participants on the event's web application. For searching people, one has to go to the home page, select the current event (Figure 10a), and then access the "Search people" menu. This gives the possibility to search for an attendant of the event in the database, see his/her details, and both find his/her position in the room or visualize some statistics about him/her (Figure 10b).



(a) Home page of the web service

(b) Person's search

Figure 10: Web interface for searching people

If we want to know the position of someone, a new page opens and shows a map like the one of the live visualization, where the sensor currently (in the last 10 seconds) detecting the searched participant is highlighted (Figure 11). This view is generated by using D3.js, as well as all of the following ones.

5.4 Live statistics

Under the same menu as for the people's search, in each person's row there also in the possibility to add some specific live statistics regarding the selected person. In this way, one can have some informations about his/her own behaviour at the event. The only statistic of this type is for the moment the possibility of seeing visually the ratio between all of the interactions had by the current user with other groups of interests. This view is called a force directed graph, and the length and thickness of the links between dots are proportional to the number of interactions had by the users (or groups) represented by the linked dots (Figure 12).

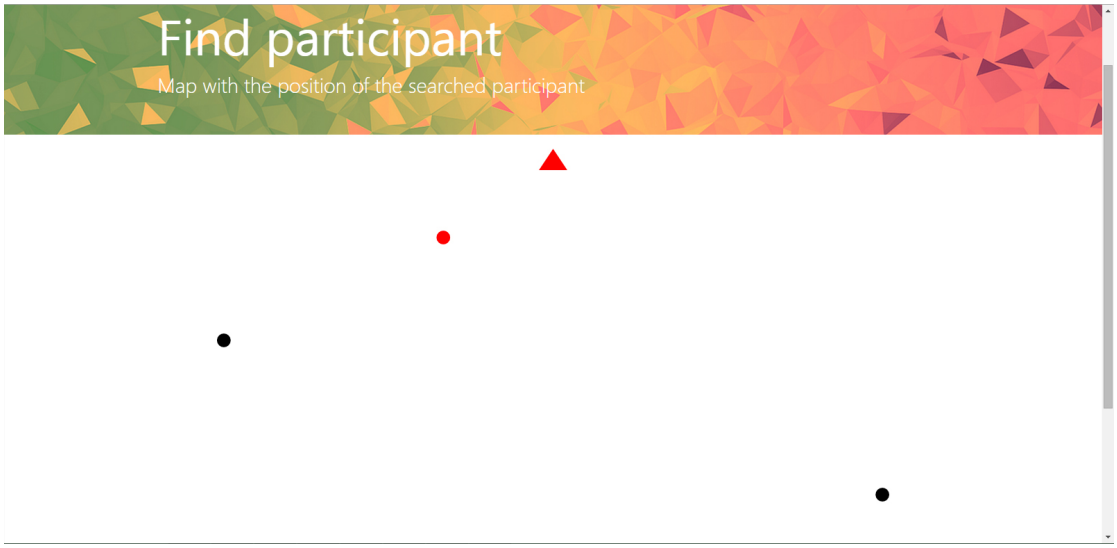


Figure 11: Highlighted table where the searched person is

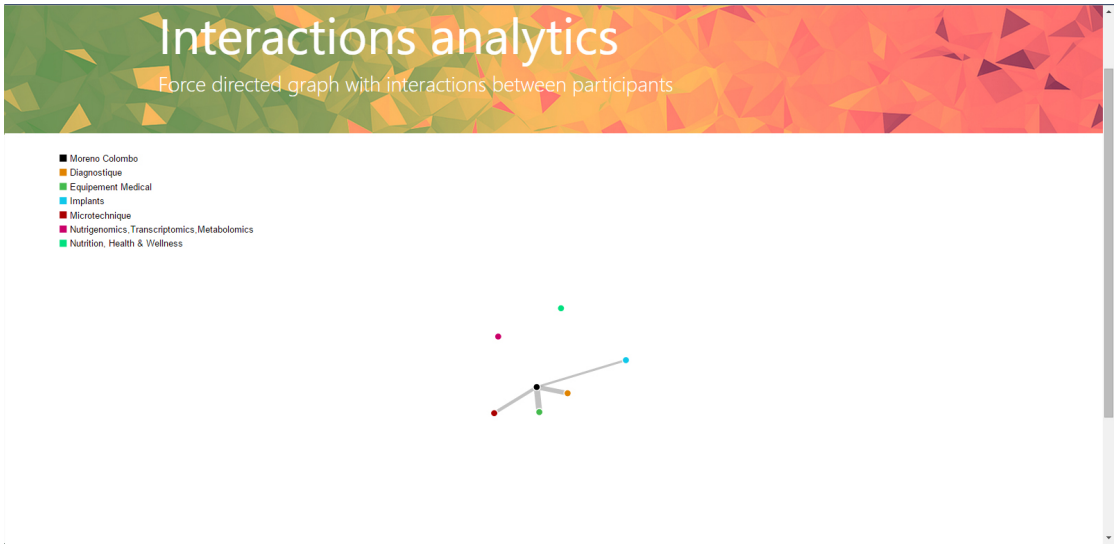


Figure 12: Force-directed graph for interactions between user and other groups

Another interesting information one can get as live statistics, which could be useful to help him/her increasing its own score, is the leaderboard with some details. This visualization shows the distribution of points for each participant, showing the percentage got from interactions, and the one got from the bonus given when one interacts with someone from a different group of interest. This can be seen in Figure 13.

5.5 Reporting

As a tool thought mainly for organizers, an interface allowing automatized post-event analytics has been created. This is accessible on the web application in the same way as the others client-side interfaces described in this Section. To access this part of the website, one has to select "Post-event" from the navigation menu. After having done this, a sub-menu allowing to choose the visualization one is interested in is shown. This post-event analytics tool works in the same way as the other interfaces described in this section, with a Django view accessing the database and elaborating data, which are sent to a Django template, where they are transformed to images by

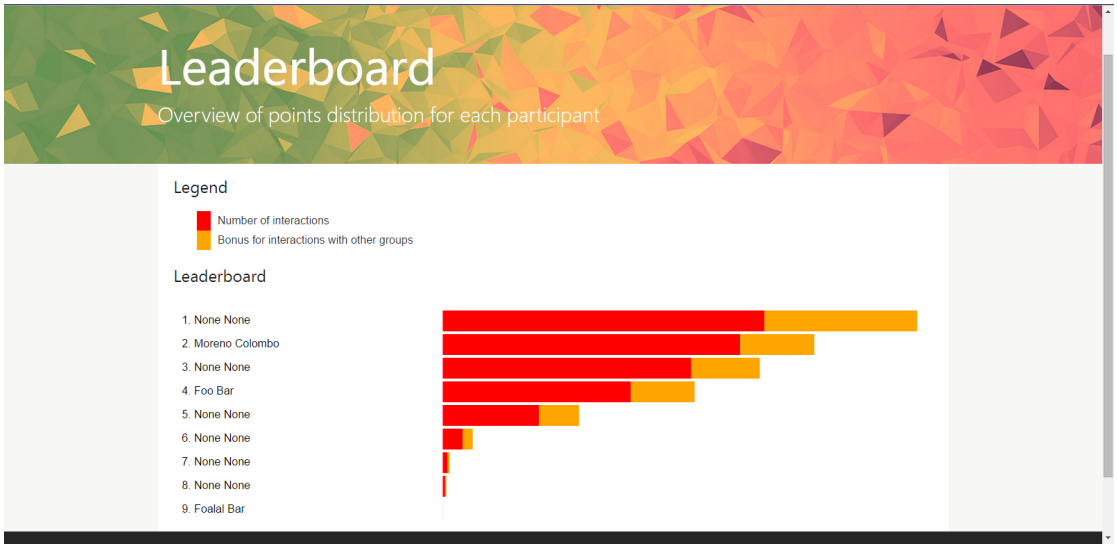
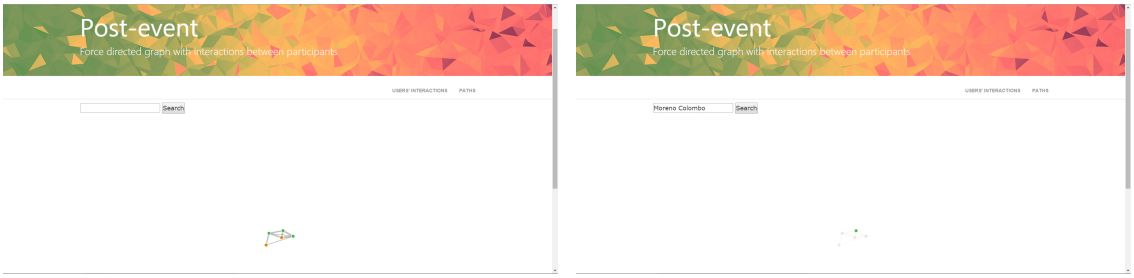


Figure 13: Detailed leaderboard

using D3.js.

Two reporting views have been created for the moment. The first one describes interactions between all of the users, by using once more a force-directed graph (Figure 14a). In this view, there is also a search box giving the possibility of searching a specific person. In fact, when one search for someone in the search box, if this person is in the graph, then all of the rest of the graph except the dot representing the searched person, becomes invisible for a couple of seconds. In this way one can clearly see where the searched person is (Figure 14b).



(a) Interactions between all participants

(b) Search of a person in the graph

Figure 14: Visualization of interactions between participants

The second reporting method, tries to find the most recurrent patterns in the walks of people. It in fact analyses the movement of people between the tables, and based on the number of persons who went directly from one table to another, a map of the tables with paths of different width connecting them is drawn. An example can be seen in Figure 15.

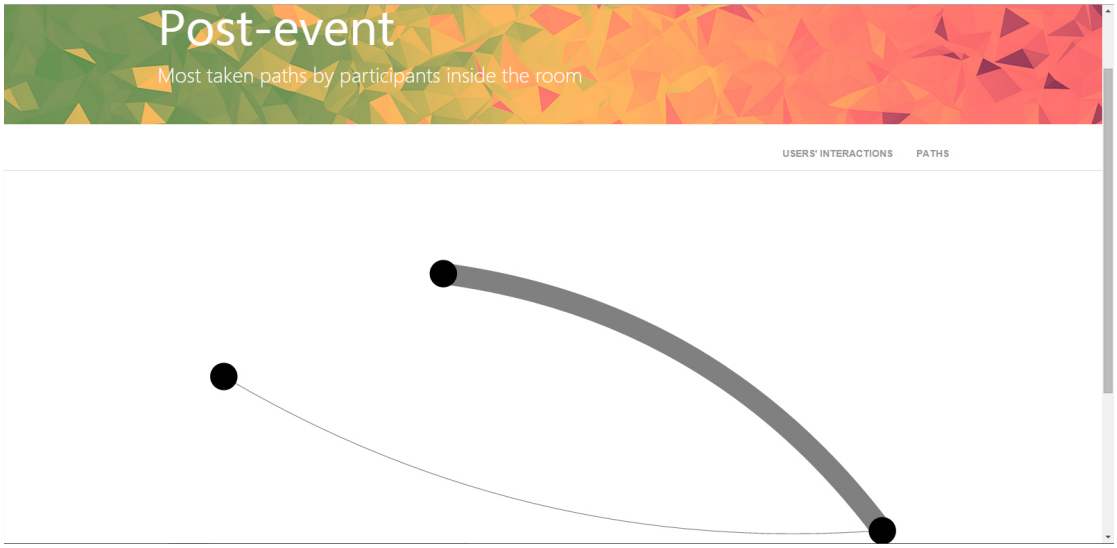


Figure 15: Most recurrent patterns in the movement between tables

6 System performance

One important step for evaluating the quality of the system, consists in testing its overall performance. For doing this, it has been chosen to evaluate each step of the system, in such a way that, if there are some performance issues, it is easy to find the exact step where the bottleneck lies, and therefore correct the performance issues.

The first step consists in analysing the performance of the readers themselves, then the communication between readers and server and the elaboration of recorded data. An analysis of database performance for reading and writing follows, and the last step consists in evaluating the efficiency of live visualization.

6.1 Readers

For evaluating the readers' performance, the most important step we take, is to measure how many tag readings per second each reader is able to do. In the manufacturer's specifications for THINGMAGIC USB PLUS+ RFID Reader[6], it is said that the antennas are able to detect up to 200 tags per second. We want to verify experimentally this value. So we set up an experiment, where a reader is connected to a computer with a JAVA application running on it, which manages the reader and counts at each second the number of tag readings. This is repeated for a long enough time, and obviously with some tags placed next to the RFID reader. Then in the end one can compute the mean number of detections per second, and see if the readings number is more or less always the same or if it varies a lot.

The experience has been executed for 2000 seconds, to have enough records to do some statistics, with 5 tags placed in circle around the antennas (only 5 tags have been used because it is the maximum number of tags available at the time of the tests).

By computing the average of all of the measures, we get 104.929 recordings/second. The standard deviation on the 2000 records is 0.467, this means that the number of recordings in each second has always remained more or less the same for all of the 2000 seconds of the experience (we can see this also from Figure 16). This means that the maximum number of readings never decreases for unknown reasons, so the readers are very stable. By comparing the average of 104.929 recordings/second with the value given on the manufacturer's specification (up to 200 recordings/second), we can see a quite big difference between the two values. But this is probably given by the fact that for our experiment only 5 tags have been used, which decreases the number of detections because the frequency with which each tag can send signals isn't high enough for

[6] `http://www.thingmagic.com/index.php/usb-rfid-reader`, August 2015

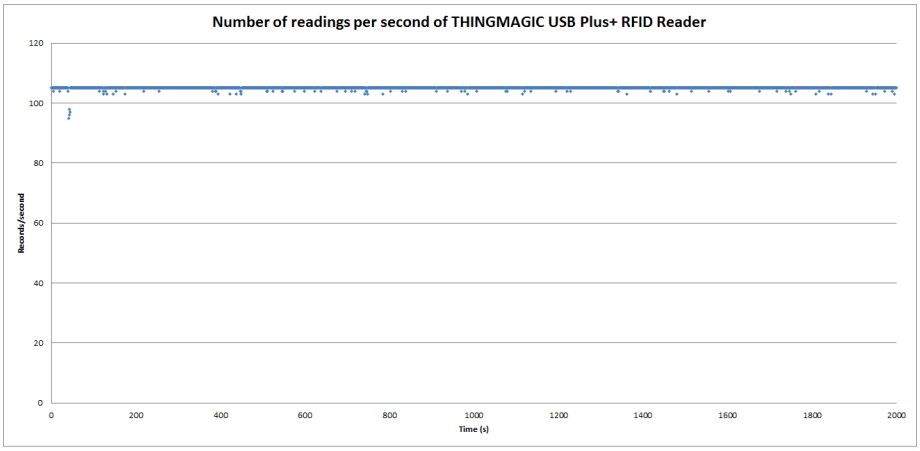


Figure 16: Test of reading frequency

having more detections. To prove this, we repeated the experience with 1,2,3 and 4 tags and we can see in Figure 17 that the number of detections strictly depends on the number of tags used. Moreover, even if the maximum number of detections per second is only 105, it would be more than enough, since in the reality there will never be more than 10-15 persons in the range of detection of the same antenna (each sensor has a detection radius of 1 meter, which is partially occupied by a table).

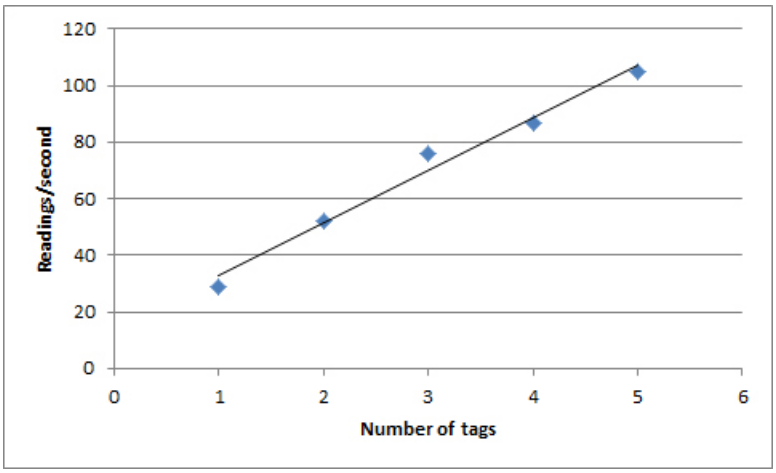


Figure 17: Average number of readings per second over number of tags

6.2 Reader-server communication

The second step to analyse in terms of performance, is the communication between readers and server. From this point of view, two things have to be checked: the transmission rate, and the average time for data transmission.

The transmission rate is easily computed as the number of tag reads received by the server over the number of effectively registered tags by the client. To experimentally find this value, we send a lot of messages of randomly generated detected tags from the client to the server, and we put a counter on the server, counting the number of received tags via socket on the WLAN. At the same time, when the server receives some data, it sends it back again to the client. In this way the client can find the elapsed time from the moment when the data has been sent and when it has been received back again, so that the transmission time can be estimated as $\frac{1}{2}$ of the total elapsed time.

We specify that data is exchanged between clients and server via socket by using the TCP protocol, which is a little slower than UDP, but at the same time less error-prone.

On a sample of 5000 message sent, 5000 were received from the server, sent back to the client and received by the client again. This experience has been repeated 3 times on 3 different WLANs, always with a transmission rate of 100% and the average transmission time has been of 4.13 ms. The value of the transmission time can vary depending on the quality of the LAN, but it should never be high enough to be noticed by users (more of 1 second delay). In fact, even by testing transmission speed on a LAN created using an Android smartphone as hotspot (the slowest solution in our test), the average transmission time was 6.24 ms, which is only 0.6% of a significant delay.

6.3 Filtering

The following step creating some small delays on the path between registration and visualization is the filtering stage. In fact, by keeping only the tags which have been registered at least three times in the last 5 reading loops, it is obvious that sometimes there are some small delays, when there is a change in the position of someone. In fact, when a person gets in the range of detection of a sensor, it has to be detected at least three times to be kept as an actual recording. This means that in this case there is a delay of about 2 or 3 seconds. The same happens when someone leaves the range of detection.

Even though this delay on position change is quite significant, it is not a negative point. In fact this delay is the result of a compromise between the quality of the system and the responsiveness of the same. It has been chosen that it is much more important to provide to users a good quality live visualization (not continuously changing because of people just walking next to a table), than to have a really reactive system. Moreover, the range of detection of the readers is not exact, so when users get next to a table, it takes about 3 seconds to register them, but the delay can't be seen, since it is impossible for them to determine the exact moment when they should have been detected for the first time by the reader.

6.4 Database

Database access speed is now analysed, this measures can help us to understand if the database can be kept in SQLite or if it is necessary to try speeding it up a little, by changing database type. At first, reading speed is analysed, and then it's the turn of writing speed. To do this, we retrieve a lot of recordings from the database and we compute the time between start and end time. Then we do the same, but by writing a lot of entries to the database.

For the reading speed, all of the recordings in the database have been retrieved, and the time necessary for doing this is computed. We do this 3 times, and the average time taken for retrieving 379743 elements is 18.138 seconds. So we can find the average time for retrieving a single element from the database

$$\frac{18.138 \text{ s}}{379743 \text{ elements}} = 0.048 \text{ ms / element}$$

We do the same again for getting the writing speed. The experience is once again repeated 3 times. Writing 5000 elements to the database, takes an average of 48.460 seconds. So we compute the average time for writing a single element to the database

$$\frac{48.460 \text{ s}}{5000 \text{ elements}} = 9.629 \text{ ms / element}$$

This means that writing to the database takes about 200 times more than reading from it. Please note that these values are tightly hard disk-bound[7], so for servers with different disks, the database speed can be really different.

By leaving everything in this way, the writing speed on the database could be a problem, if at each second more than 103 tags are registered and put on the database. In fact, this would create a delay in the registration, which would increase at each reading loop. This could produce some

[7]These measurements have been done with a Lenovo ThinkPad X1 Carbon, i5-3427U @ 1.80GHz, 8 GB RAM, 256 GB Solid State Drive.

quite big delays in the live statistics, while for the live visualization there would be no problem, since the database step is not needed.

Fortunately, from Django v1.6 there is the possibility of saving a lot of entries in the database at the same time with a single disk access. This reduces a lot the writing time, in fact the same sample of 5000 elements is now registered in 0.584 seconds, which means

$$\frac{0.584 \text{ s}}{5000 \text{ elements}} = 0.117 \text{ ms / element}$$

Which is a much better speed, since it allows to write up to 8547 tags per second.

In the real application, even with a single database access for each update, at each reading loop two tables are updated: Record and Interaction. So we test the time taken by the program to update those tables at each interaction. We now have a small problem, since the time taken by the operation of updating Interaction doesn't increase linearly with the number of registered tags. In fact for doing this, all of the possible combinations of 2 numbers of the registered tags are saved in the database. The chosen test method consists in measuring the time taken for the update of both of the tables relatively to an increasing number of tag reads. We fix the maximum number of readings to be registered to 20, since it should be impossible to have more than 20 persons around the same small table. The time is registered 1000 times and the average time is taken as result of the test. Since in this project the goal is to provide a system working for an event with at most 300 persons, it can be more interesting to see how much time is required to register 300 participants at the same time, with 1 to 20 detections per table. This gives an idea about the efficiency of the database write access, and about the choices in terms of sensors number.

The results of this test can be seen in Figure 18.

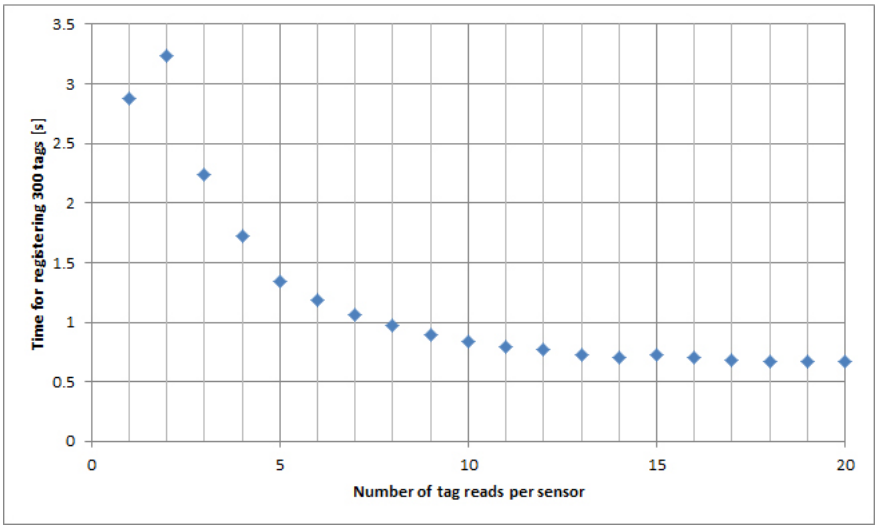


Figure 18: Average time to write 300 tags to the database

From the results of the test, we can clearly see that when more than 8 tags are registered in a single sensor, the time taken to register 300 participants is less than 1 second, which is the time between a recording and the next one. This means that if in the room there are $\lfloor \frac{300}{8} \rfloor = 37$ sensors or less there are never delays caused by database access for writing, even when all of the 300 participants are registered at the same time.

From this graph, one can also see that from a performance point of view it is better to have less sensors. But this also implies that if all of the users want to talk with someone at the same time and be detected, it becomes difficult to find a free spot next to a table. So to have a good compromise between these two points, the best choice would be to have about between 15 and 30 RFID reader for an event with 300 participants, depending on how much time people spend next to the tables.

6.5 Live visualization

The last step to analyse to get the overall system performance is the live visualization. More precisely, we want to check how much time it takes for updating the view and then the overall delay between tag registration and visualization.

To check the time taken by JAVA for drawing the view, a timer starting right before the drawing loop and stopping right after it is created. The results over 100 iterations are analysed and we can see that when there are no changes in the visualization, the drawing time is between 0 ms and 1 ms. When there are instead some changes in the visualization, a time between 5 ms and 14 ms per drawing loop is registered. To prevent these small delays to bother users because causing a delay in the live visualization, we change the refresh rate of the visualization from 1 Hz to 10 Hz, this causes delays due to live visualization not being correctly synchronized with recordings updates to be always less than 0.1 s, which in this application are imperceptible from an human point of view.

To find the overall delay between the first tag reading and the live visualization, a timer is started when a tag is first read, and it is stopped via a socket message when it has finished being drawn by the live visualization method. At the end, the average message transmission time is subtracted from the total time registered by the timer.

The timer is started at the absolute first tag read, and is stopped when the first tag has been drawn on the live visualization map. This means that this reproduces the case where there is a change in the detections, so the filtering part creates a delay which should be of about 2 or 3 seconds. The time of execution has been registered 10 times and the average measured delay is of 2769 ms. So we can subtract the average transmission time found in Section 6.2 to this value. We get

$$2769 \text{ ms} - 6 \text{ ms} = 2763 \text{ ms}$$

This delay is almost all given by the filtering part, and as said in Section 6.3 it is a delay created to provide a better looking, smooth and less error-prone live visualization.

7 Conclusions

The main goals of this project were to give people an efficient tool allowing them to optimize their behaviour at a networking event and a way for organizers to analyse what happens during the event. This work has been based on Thomas Rouvinez's ITSI 1.0 , and it mainly aimed to make the first version of live visualization faster, include gamification in it, reduce logistic difficulties by making it wireless, include a function for the search of people and automatize analytics visualization.

The live visualization structure has been totally changed to increase its speed, accuracy and at the same time include gamification. This has been a successful choice, since as one can see in Section 6.5 the time between the registration of a tag and its visualization has passed from a random time between 10 and 25 seconds to an average of 2.763 seconds. Moreover, the continuity and precision of live visualization have been ameliorated and gamification has been integrated directly in the live visualization.

The step after this has been the modification of the base JAVA program and the creation of a new one to be hosted on small computers. Those modifications aimed to manage WLAN message passing between the central server and the readers, in this way the communication between them has become wireless. As seen in Section 6.2, this WLAN communication doesn't create delays in the execution of the program, and at the same time simplifies a lot the installation of the required hardware.

In the end, some ways of visualizing informations have been added to the already existing web application, which has also been completed with an user-friendly way of using it, and made nicer to see and responsive, in such a way to be used even on smartphones without any problem.

To continue this project, something interesting to do would be to add more ways of visualizing (both live and post-event) the registered informations. For example in the few visualizations proposed in the actual project, the time dimension is never taken in account for data analysis, even if it could be really useful for organizers and users. Moreover, a more precise way of detecting interactions could be introduced, since in ITSI 2.0 an interaction is registered when people are next to each other, even if they are not talking and maybe not even looking at each other.

So we can conclude by saying that all of the initial goals have been fully reached, and even if sometimes some complications occurred, doing this project has always been really challenging and motivating.

Appendices

A Source code

This is an open source project, the source code, as well as the application ready to be installed, can be found at `https://github.com/colombmo/itsi-gamification`.

B Installation

The files necessary for the installation can be retrieved at `https://github.com/colombmo/itsi-gamification`.
Moreover, in order to run ITSI 2.0 , the following tools are required:

- Python 2.7.9;
- Xampp 5.6.3;
- Java 8 Standard Edition;
- VCForPython27.msi (Python compiler for Windows).

Installation instructions in Windows (for the central server):

1. Install Python and add *C:\Python27\;C:\Python27\Scripts* at the end of the `PATH` environment variable (My Computer > Properties > Advanced > Environment Variables)
2. Run a shell with administrator privileges in the *Installation* folder and type `python get-pip.py`
3. Run a shell with administrator privileges, execute `pip install Django`
4. Run a shell with administrator privileges, execute `pip install django-extensions`
5. Run a shell with administrator privileges, execute `pip install pyyaml`
6. Run a shell with administrator privileges, execute `pip install django-tables2`
7. Run a shell with administrator privileges, execute `pip install django-filter`
8. Install VCForPython27.msi
9. Install Xampp
10. Place the folder *WebService* in `xampp/htdocs`
11. Adapt `xampp/htdocs/WebService/mod.wsgi` file to fit your *WebService* path
12. Go to `xampp/apache/conf` and replace `httpd.conf` with the one provided in *Installation* (check the paths correspond to your Windows installation)
13. Copy `mod_wsgi.so` in `xampp/apache/modules`
14. Install Java 8 SE
15. Run a shell in `xampp/htdocs/WebService`, execute `manage.py runserver 0.0.0.0:80`
16. Open a browser and test this link: `http://127.0.0.1/hubnet/version`, if this opens a page with the text *v2.0*, then the installation was successful.

For getting the reader controllers ready to work, one has just to install Java 8 SE on them.

C User manual

Each time the application is run, one has to launch a server in order to manage database access and web applications. For doing this, run a shell in `xampp/htdocs/WebService` and execute `manage.py runserver 0.0.0.0:80`.

Then, before running a new event, this has to be set up in the database by accessing the wizard at the following address: `http://127.0.0.1/admin` and logging in using the username *admin* and password *livefeed*. Then:

1. Create a new event object and fill in the name, start and stop dates.
2. From the same screen, add interest tags, sensors and participants (reuse already created ones or create some). More can be added by clicking the "Add another ..." button.
3. Click on the "Save" button.

The database is now ready, since interest tags, sensors and participants are associated to the event.

The following step consists in the configuration of the server and of the reader controllers. In both the server and the clients, copy their corresponding folder (found in `Installation/Java/Program`) somewhere easy to find (for example on the Desktop). In the folders there are a Java application (`.jar`) and a configuration file (`default.cfg`).

In the configuration file of the reader controllers, there are informations about the IP address of the server, the port to communicate to, and the COM port through which the sensor is connected to the controller. For each controller, open the configuration file and write the following:

1. the IP address of the server (server and clients have to be on the same LAN), which can be found by going on the server, opening a shell and typing `ipconfig` in Windows, the IP address is the value next to `IPv4 Address` of the form `192.168.1.115`;
2. the COM port to which the reader is connected, which can be found under *Device manager > Ports (COM and LPT)* in Windows. The RFID sensors are *USB Serial Port (COMx)*, so in the configuration file one has to write `comx`, where `x` is the number found in Device Manager;
3. the port to which the controller will send data (a random number, preferably between 49152 and 65535[3], which has to be different for all of the sensors).

All of these elements are separated by a comma.

In the configuration file of the server application there are the informations used to know to which port listen for data sent by a certain controller, to which sensor in the database is associated that controller, and for which event to record data. So open the configuration file for the server and write the following:

1. a port number (an unique identifier associated to a sensor, preferably between 49152 and 65535) followed by a comma (,);
2. the identifier of the event, so the value in the column *id* of the event which has been created in the database, followed by a comma;
3. the identifier of the sensor, so the value in the column *identifier* of the sensor corresponding to the real sensor connected to the port specified in step 1.;
4. change line and repeat these steps for each sensor associated to the event.

At the end, with three sensors this file should look like this:

```
49152,13,21,
49153,13,22,
 49154,13,20
```

and the configuration file on the controller connected to the sensor 21 on port COM2 will be as follow:

```
129.168.1.115, com2, 49152
```

Once the configuration files are ready, to start reading and showing the live visualization, launch the Java program on the server, click on *Start reading*, and then launch the Java program in all of the reader controllers, too.

On the server there is also the possibility of adding some time markers during the event. To do this, on the control panel, insert an identifier under *Time marker* and click on *Add Marker*.

To end the reading, click on *Stop Reading*, which will close the connection to the sockets and terminate the application running on the controllers.

To access the web application, connect to the same WLAN as the central server and write the server IP address (the same as for Step 1. for the configuration file of the controllers) in the address bar of any web browser. Then choose the current event in the home page and use navigation to explore all what can be done.

References

[1] Thomas Rouvinez, *Real time tracking and visualization of indoor social interactions*, University Of Fribourg, 2015.

[2] *Gamification*, `www.badgeville.com/wiki/Gamification`, June 2015.

[3] *Ephemeral port*, `https://en.wikipedia.org/wiki/Ephemeral_port`, August 2015.