LO53 Lab report

Tao Sauvage
Stéphane Parunakian
Thomas Gagneret

Contents

Chapter 1

Positioning server

In this chapter, we will speak about the positioning server part of our project.

1.1 Tools and versions

Before seeing the server itself, lets see the different tools used for its creation.

1.1.1 Java

This server is coded in Java 7, the programming language of Oracle. So the different servlets must use TomCat 7.

1.1.2 Simulation scripts

In order to test the server, without having to setup all the elements of the project (Access points, mobile device), three Python scripts have been written:

AccessPoint Simulate the response of an access point.

MobileDevice (Calibration) Simulate the calibration request of a mobile device.

MobileDevice (Localisation) Simulate the localisation request of a mobile device.

1.2 Programm

Now we will look at the server.

1.2.1 DAO

A major element of the server code is the Data Access Object (DAO). It is the link between the database and the code. It allows the programm to manipulate data as it was objects, without having to care about the database. The DAO creates objects from information in the database, and updates information in the database from objects.

The major advantage of this concept is the possibility to change the way to store data easily. Only the DAO interface needs to be modified in case we want, for example, to use a NoSQL database.

1.2.2 Calibration

The Calibration servlet is called by mobile devices to put calibration information in the server database.

How the servlet works:

1. A mobile device send a calibration request, with a map id and coordinates.
2. The servlet get a list of all routers present in database.
3. If there is enough routers, it goes on, else it stops.
4. It sends RSSI requests to router and wait no more than 500 ms for the response.
5. If there is enough useful responses, it stores them in the database (RSSI table), else it stops.
6. It sends a response to the mobile device.

1.2.3 Localisation

The Localisation servlet is called by mobile devices who want their localisation.

How the servlet works:

1. A mobile device send a localisation request, with no parameters.
2. The servlet get a list of all routers present in the database.
3. If there is enough routers, it goes on, else it stops.
4. It sends RSSI requests to router and wait no more than 500 ms for the response.

5. If there is enough useful responses, it stores them in the database (TempRSSI table), else it stops.

6. It gets every location calibrated from the database and calculate the distance with the data in the TempRSSI table, for each location.

7. It returns the closest location to the mobile device (map id, coordinates).

1.3 Database

The database of the server is shared between the servlets. Its structure is almost the same as in the subject. There is only one addition: an "ip" column in the AccessPoints table (it is more reliable than using the id for the last part of the ip).

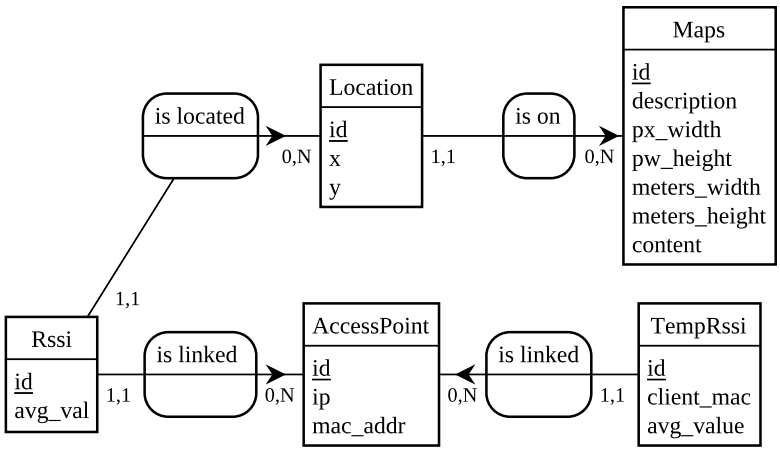

Figure 1.1: Database

1.3.1 Creation script

In order to create the database, a postgresql script has been created. It creates a user "lo53" with a password "lo53" and a database "lo53_rssi".

1.3.2 Filling

The filing of the database has to be handmade, except for the Maps table. As this table contains binary data, it is quite difficult to do it manually. So we wrote a

filling script to do this. This script put maps pictures in the database, and get informations about maps from text files like this one :

```
[picture]
description: This is the description for the H_RDC picture
meters_width: 2.5
meters_heigt: 3.5
```

The size in pixel of the picture is automatically determined, thank to the PIL library. The addition in the database uses the psycopg2 connector.