Internet of Things (IoT)
/
Internet of Things (IoT)
/
Application

Gilles Menez
University Nice Côte d'Azur
email: menez@unice.fr
www: www.i3s.unice.fr/~menez

January 8, 2024: V 1.0

1 Architecture of an IoT application

An IOT infrastructure generally relies on 3 actors:

① A fleet of fixed or mobile connected objects, distributed geographically,

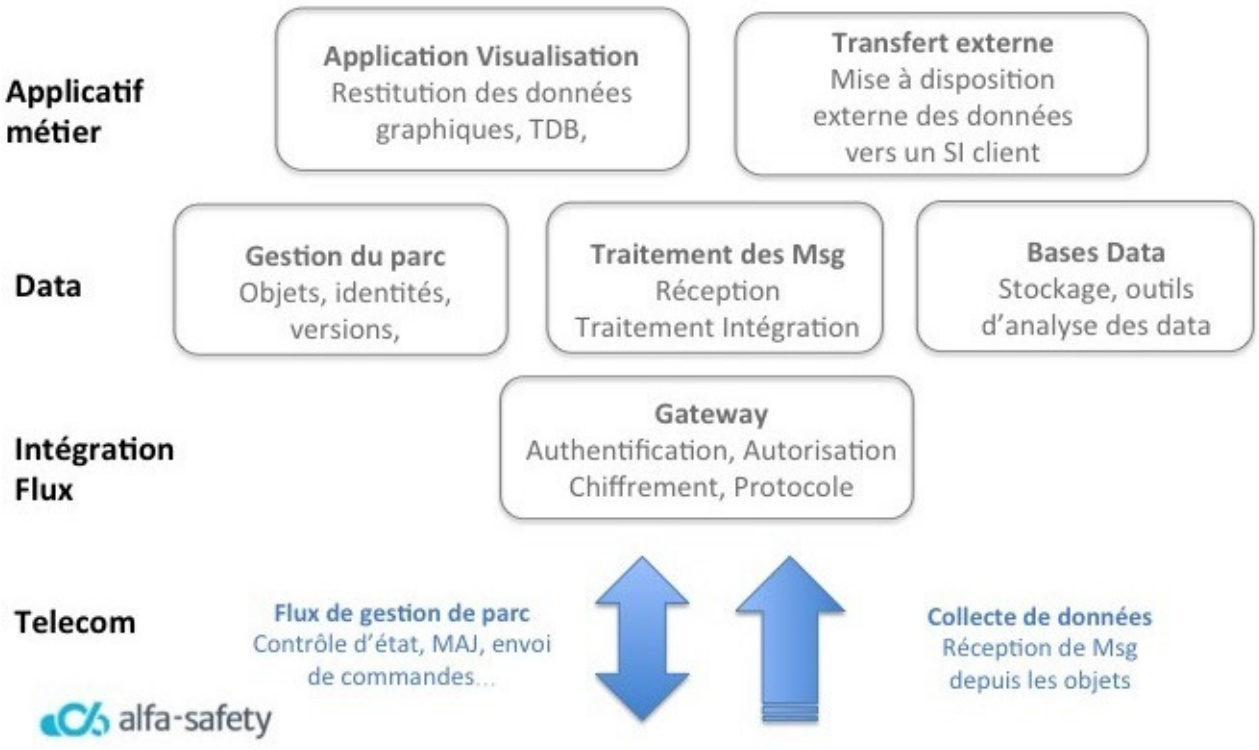
② A telecom network (wired or not or a little or a lot) which will allow objects to be connected by transmitting messages.

③ An application which collects "data" from the network of objects to provide more or less "intelligent" aggregated information or to control a system (a wind farm, or a swimming pool park, etc. for example).

The application is also in charge of managing the fleet of objects

Such an application includes "classic" functionalities distributed over different levels:

- job,
- data,
- flow management,
- telecommunications.



1.1 Gateway: flow integration

The Gateway function is the “exchange gateway” of messages between the application and the object park, **it interfaces with the telecommunications network**(via support for MQTT/HTTP/ type “application” protocols. . .) and is particularly responsible for security:

- ✔ Authenticate and authorize objects to communicate with the application.
- ✔ Control the integrity of the data and therefore prevent the possibility of injecting fictitious data sets from illegitimate objects (this is the minimum!)
- ✔ Preserve the confidentiality of exchanges, for example by encryption.

The implementation of security will depend on the protocol and the dialogue mode chosen. Very often this security will "increase" the processing and transmission costs and consequently the energy cost.

- ✔ There are trade-offs to be made depending on the nature of the application.

In operation, care will be taken to implement appropriate application supervision which will make it possible to detect any anomaly in message flows: transmission outages, erratic values, etc. . .

1.2 Message processing

In the context of processing carried out on the information that comes from objects,**the question of volumetry is critical**.

- ✔ Even if at a given moment this functionality seems correctly dimensioned, it (this function) must be able to absorb (receive, process and integrate)**a potentially fluctuating message volume**.

  "Fluctuating" because the objects transmit the state of the real/physical world and if this state evolves, the volume of data has a high chance of evolving because sampling techniques are often (very) reactive to events or changes. variations.

On a "more controllable" time scale, the increase in the number of objects also generates a fluctuation in volume.

- ✔ This development may call into question the initial architecture: see “the MQTT need”!

1.3 Park management

A fleet of objects is constantly evolving: increase in technical capabilities, generations of hardware and software. . .

- ✔ Park management is therefore vital: inventory, status, updating. . .

The size of the fleet influences the complexity of the task and may require a management platform.

We are seeing a number of these IoT platforms offered by hardware manufacturers introducing them as an add-on or as part of their hardware offering.

1.3.1 On a “small” scale. . .

Such applications enable "limited" IoT functionality:

- ➢ connect devices and sensors to the cloud/cloud,
- ➢ monitor and collect data,
- ➢ and provide visualization of the IoT project.

Home automation typically falls into this category of applications:

- • few objects,
- • little diversity of objects,
- • few networks,
- • few protocols,
- • reduced distances, . . .

in short, complexity and limited diversity.

If we were to make an analogy with the world of machines,**home automation is the equivalent of a machine/PC room such as that of TP**and we**measures the real complexity BUT limited**a software platform/application allowing the management of such a room.

1.3.2 On a “large” scale

Such a scenario/configuration can quickly evolve and become complicated!

If there are several machine rooms, on different sites with heterogeneous networks, with machines of different types (Hardware/OS), with different functions (servers, gateway, etc.). . . management is becoming more complex!

For large-scale IoT scenarios (Smartcity, Farming, etc.) we find (amplified) problems:

Park maintenance requires real**management platform**Objects.

- ➢ This platform allows connection/membership, management, updating and integration of thousands of devices and sensors.

1.4 Databases

The collection of objects will feed the application with a data flow which must be analyzed. However, this analysis is all the more relevant as it is based on a suitable time horizon.

- ➢ To generate this horizon, it is necessary to “store”. . . in Databases.

1.5 The application server

The objective of an IOT application is to process/analyze and then present the knowledge to users.

- ➢ This “knowledge” can consist of raw data or the results of advanced analyses.
- ➢ Often a graphic presentation summarizes this graphically on a “Dashboard”.
- ➢ The application server can be accessed from the Web or from a smartphone or even a dedicated tool.

Access to this server may be limited if the audience is targeted, it may become important for a general public audience. Again, pay attention to sizing.

1.6 External data transfers

Another way of using data is to**transfer to the IS of a client who will in turn integrate them to exploit them in their particular use case**:

- ➢ An IOT service operator offers its customers to subscribe to an information service based on its network of objects, the data is sent to the IS of the end customer who will use it to produce its own services, such as a monitoring or maintenance company which transmits certain alarms to a subcontractor who is responsible for intervening on site.

Finally, we will again take care to implement effective application supervision of flows and particularly in terms of access rights.

2 Attempts at standardization

We discussed in class the problem of IoT solutions developed in silos. . . with so many "standards".

The complexity of the standardization task is great, particularly because the problem has several dimensions:

① There are the**"areas of application"**represented vertically: cities, health, energy,. . . They are certainly old, possibly with specific constraints.

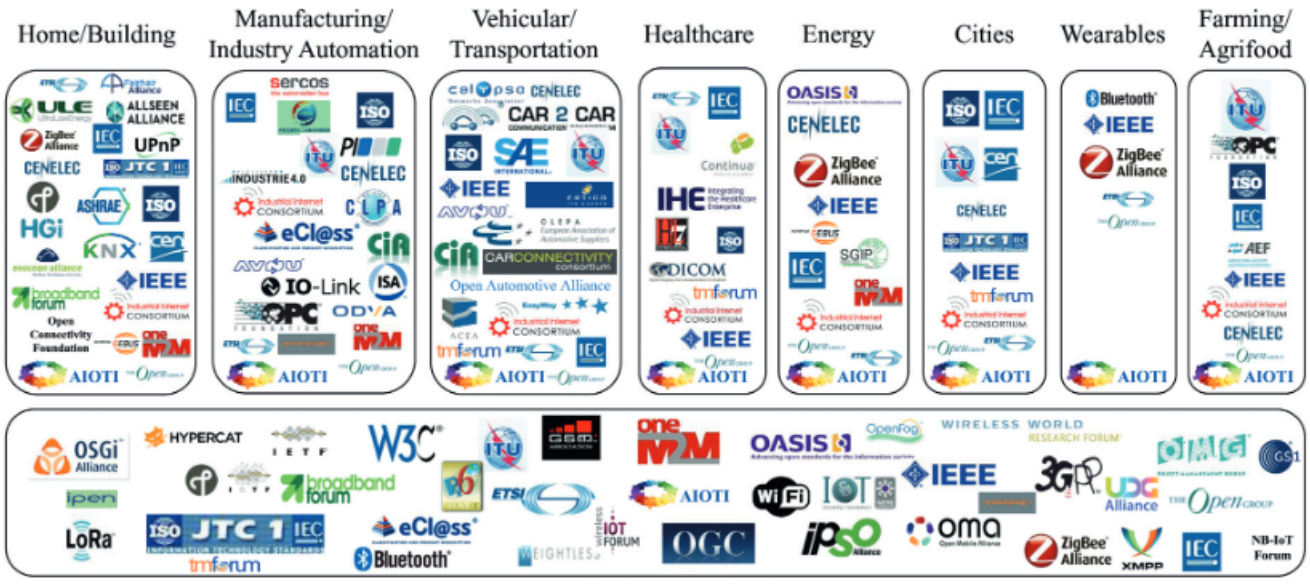
② Horizontally, telecommunications infrastructures which seek to meet needs and capture these flows.

To the level where the Internet makes everyone agree,**technologies and protocols are multiple and potentially competing**.

Their constraints are the extent, the consumption, the flow, the media, . . .

③ And finally there is the "business", a dimension underlying any "market" which means, for example, that the user ends up with dozens of different standards of USB sockets ;-)

The following figure lists some actors/stakeholders in these areas.



Two types of actors:

➢ **Standards Developing Organization**(SDOs) which develop and propose (rather technological) IoT standards,

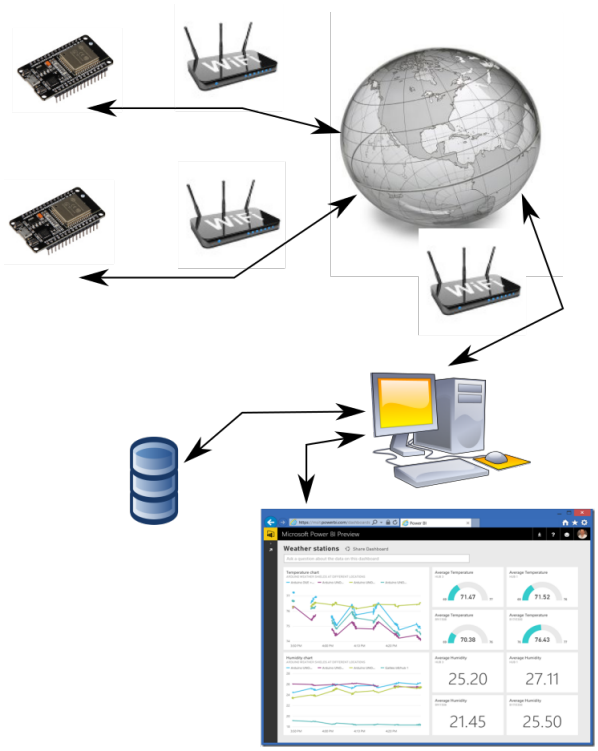
➢ and**"alliances"**some of which have understood that a standard could also be "de facto" and which can serve (when they bring together industrialists) rather commercial, marketing, promotional objectives. . .

3 A “generic” application

To tackle IT issues in the IoT field, why not try programming a somewhat generic application?

We “remain” in an architecture of the type:

"objects (ESP32)<->1 network<->Service/Monitoring station (PC)".



Compared to previous achievements, this new application/architecture will, in addition, focus on the

① Persistence :

We see in the figure a database without which it is not possible to consolidate/construct “knowledge” (reference to the top of the “knowledge pyramid” in the course).

② Deploying a service in the “cloud”:

Your application must now exist on the resources made available in the Internet space.

Its accessibility must be universal and permanent (24 hours a day)!

4 “Water BnB”

For this subject, you are the owner of a beautiful villa with a “connected” swimming pool and a large gate.

But you end up using it very little because you are often on your yacht!

You decide to create a service that allows you to rent/use a swimming pool while its owners are away.

- There are quite a few difficulties to resolve to obtain a "product" but you will focus on managing "access to the pool".

This is the subject of the TP!

The use case is as follows:

1. A user of the WaterBnB service has a dashboard showing surrounding pools.

   Behind each swimming pool, there is an ESP who publishes his status on the topic "**uca/iot/pool**" and gives its "name" to the pool (based on the "ident" field of the Json schema).

2. He wants to rent one of them for a little swim.

   The rental request is made by clicking on the swimming pool selected on the dashboard.

3. Access is authorized (or not) by the (paid) WaterBnB service that you will/must develop in the cloud (render.com)

   He is the one who opens the portal and invoices for the service!

An advice :

- Read the entire topic carefully before starting. . . to have a global view!

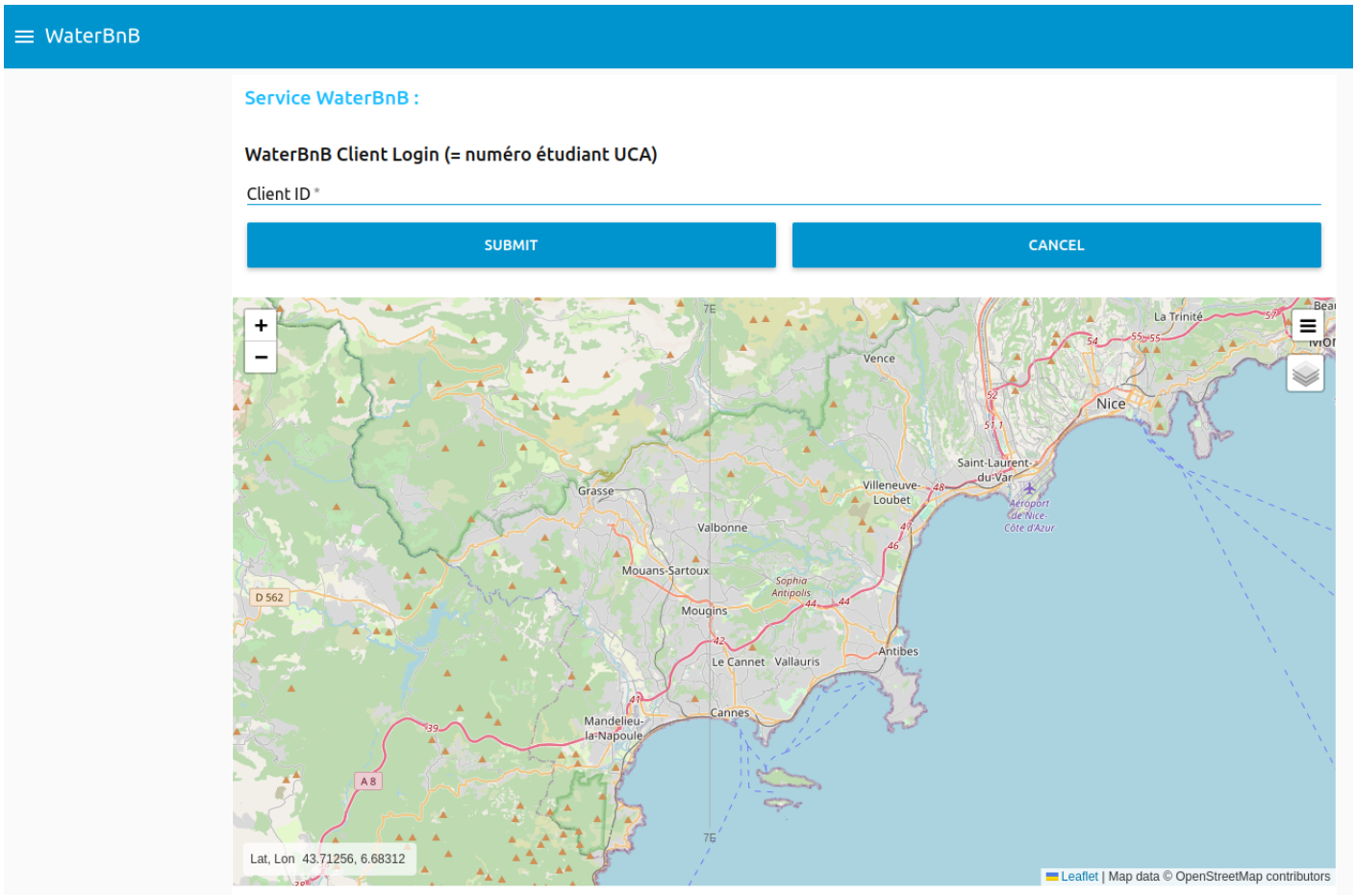
4.1 Service access interface

Since the end of Step3.7, you have a dashboard where you see the neighboring pools with a "hotspot" indicator (in red or blue depending on the calculation result).

You will make two changes so that this dashboard becomes a (graphical) interface for accessing the WaterBnb service:

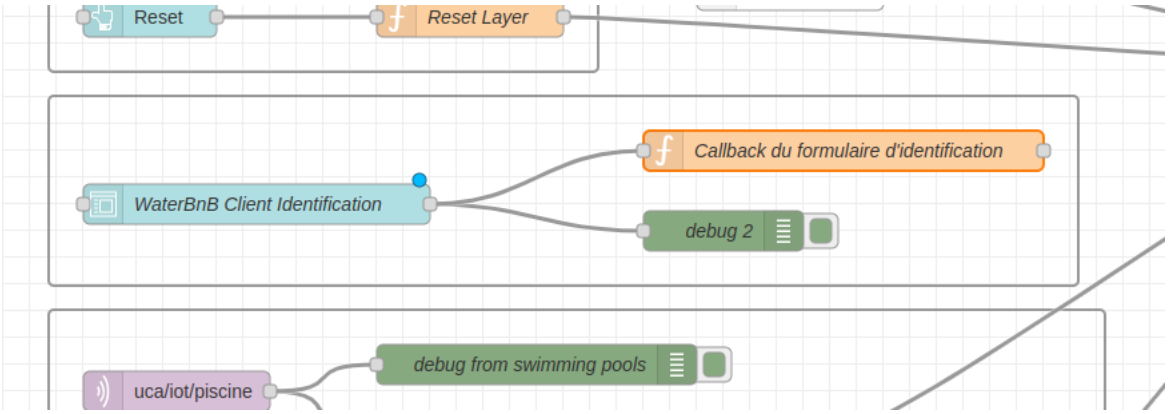
4.1.1 Customer identification

The first development consists of being able to define an identification of the person who uses this dashboard.

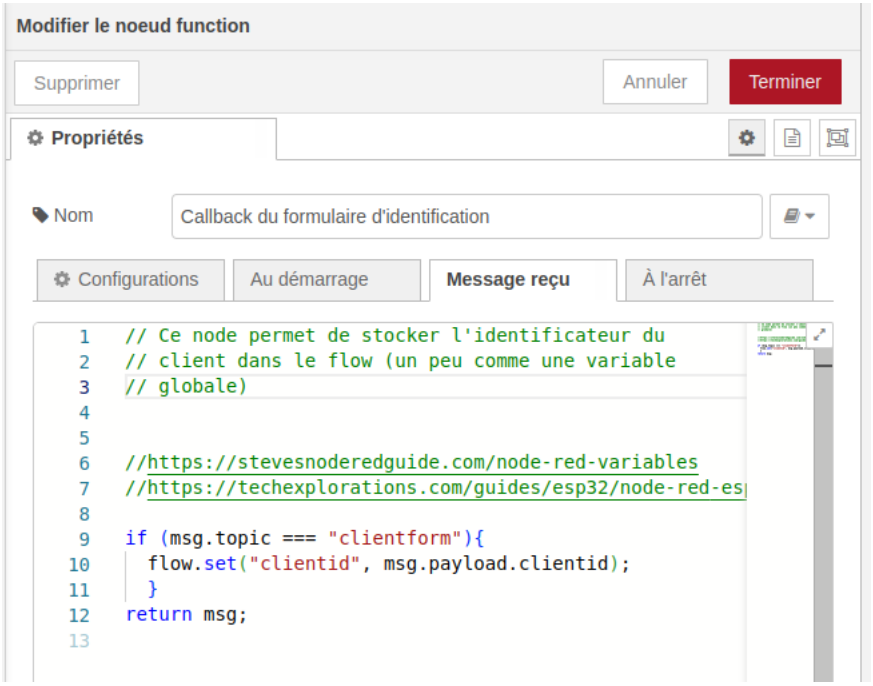


This identification corresponds to their “customer name/login” for the service.

1. To obtain the input line (above the card), I added a form node in the NodeRed flow which allows you to specify an identifier at the dashboard level.



2. The function node that follows it defines a callback called by submitting the form.



In this function, the function "**set**" applied to flow**allows you to define a global variable (in the NodeRed flow)**.

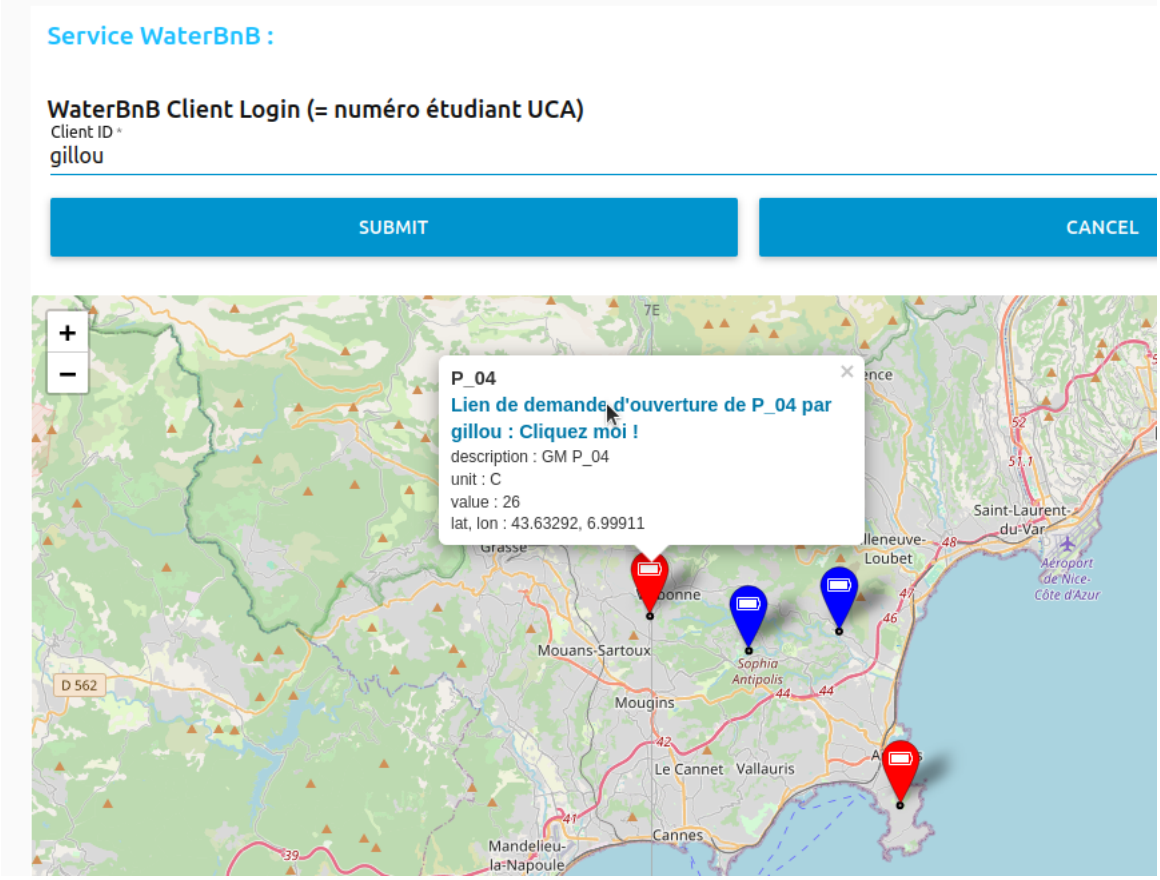
- The identifier of this variable is "clientid"!

This variable contains the name of the service user.

4.1.2 Identified access request

At the marker of each pool on the map, I added (via the "json adapt" node) a hyperlink in the menu.

The menu and the hyperlink can be viewed by clicking on the swimming pool icon:



For now, this hyperlink points to the URL "http://localhost:1880/open" which corresponds to the nodered "open" node defined further down in the flow... this allows you to debug!

On the other hand, by completing it with the "good" arguments, here is a way to request with a GET request

- to a server in the cloud. . . (which will take the place of localhost),
- to leave the clientid (contained in the NodeRed flow variable)
- enter the pool identified by mouse click.

This will be the “access request” to the swimming pool!

5 TODO: Access authorization

5.1 Finishing the pool marker

So your job is to improve the URL of this hyperlink so that it makes the HTTP GET request with the "/open" route to the WaterBnB server with two parameters:

- “idu” which has the value of the service client (therefore the value of clientid)

• "idswp" which has as value the name of the owner of the pool (therefore the info.ident field of the JSON).

We recall that**the swimming pools are identified by "P_numeroetudiant"**.

5.2 A server. . . must answer!

The WaterBnB server must be hosted in the cloud to be accessible 24 hours a day.

In what follows, I choose the solution of a server "in Python/Flask running onhttps://render. com and connected to a database hosted on Mongo Atlas (the mongoDB cloud:https://account.mongodb.com/account/login)"

Why Flask rather than Django? => faster skill development:

https://www.ionos.fr/digitalguide/sites-internet/developpement-web/flask-vs-django/

If that doesn't suit you and you prefer to do JS or . . . on another cloud: why not!? but the service must be accessible 24 hours a day!

5.2.1 render.com

"https://render.com" is one of the clouds that allows you to build and deploy an application... an application server for example!

• It may be free (or not depending on your constraints). . . what is no longer "https://heroku.com" :-(

• It supports a large number of languages including Python.

• It connects well with Github, and with the MongoDB Atlas cloud.

• Overall, I think it's pretty well done.

We start by creating a GitHub repository: "WaterBnB_studentnumber"

• I won't say more about GitHub. . . I hope you know how to do it! (this is outside the scope of IoT)

I should be able to access your repo with a link like this:

https://github.com/login_git/WaterBnBF_studentnumber

To avoid plagiarism, the repo remains private AND you invite me! (=>gmen under GitHub) We will connect this repository to render in the following.

We return to render:

1. You create a (free) account onhttps://render.com/
2. You take a look at the tutorials/documentation in the Docs/Render QuickStarts menu:

- For a python flask app:https://render.com/docs/deploy-flask
- For an express js app:https://render.com/docs/deploy-node-express-app

3. In your dashboard, you create a “New =>Web Service"
4. Which will be “Build and Deploy from a Git repository”! You use the one you just created!
5. You are using the settings of a Python Flask app mentioned in the "Python Flask" QuickStart.

You should end up with a Python Flask service whose url will look like:

https://waterbnb_studentnumber.onrender.com

Unfortunately with a free cloud. . . it is possible that the service is swapped and therefore turned off. . . just need time to restart it!

The service may offer a front page (for example:https://waterbnbf.onrender.com)**BUT** Above all, he must decide whether or not to give access to the swimming pool by responding positively (or negatively) to a request to open the access door.

5.2.2 Flask

https://flask-pymongo.readthedocs.io/en/latest/

I'll give you something to start with:https://github.com/gmenez/WaterBnb

- There are illustrations of the use of Flask, but also of MQTT and MongoDB.
- The mongodb cluster login is: visitor
- The mongodb cluster password is: doliprane
- For visitors, the database is read only
- For network connections, you must ask me to register your IP as the right holder.

5.2.3 Mongo Atlas

You need to create a (free) account on the mongo cloud (https://cloud.mongodb.com/Orhttps://www.mongodb.com/fr-fr/cloud/atlas/lp/try4)

- https://www.geeksforgeeks.org/sending-data-from-a-flask-app-to-mongodb-database/
- https://www.digitalocean.com/community/tutorials/how-to-use-mongodb-in-a-flask-application
- https://stackabuse.com/integrating-mongodb-with-flask-using-flask-pymongo/

In your Cluster/Project on the cloud you should create a "WaterBnb" database and at least one "users" collection.

5.2.4 Access conditions

For a “toto” user of the WaterBnb service to have access to the “P_123456” swimming pool (on which he has just clicked), several conditions must be met:

1. “toto” must be a declared user of the service.

   In the code that I give you, you have access to a file which is used to fill the "users" collection of the WaterBnB database.

2. The swimming pool must exist**And**it must be "not already occupied".

   We have this information! . . . thanks to the MQTT flow ("uca/iot/piscine") of the swimming pools.

   In the code that I give you, there is an example of MQTT manipulations.

5.2.5 The effects of access

Everything is played out on the ESP Led Strip!

1. If it is green then the pool is available for rental. This is the color before the opening request.

2. If it is yellow then the pool is now occupied (it remains occupied as long as the light sensor says so)

3. If it is red then access is denied. It stays in this state for a few seconds then returns to green.

It's up to you to provide this feedback to the ESP from the app server!

6 TODO: Data exploitation

Any IoT application worth its name must include knowledge synthesis. . . deduced from its data.

Mongo Atlas allows you to create dashboards from stored data:

https://charts.mongodb.com/charts-project-0-wdlbq/public/dashboards/6579b0fa-11d0-477b-846d-2ffda2aaba29#

Do you recognize these numbers?

I would like to know more. . . for example (at a minimum), a histogram of requests: which pools are most in demand?

Anything that has meaning interests me. . . but it's clear that it's "just" handling a tool so there's no point in overdoing it.