Internet of Things (IoT)
/
Internet des Objets (IdO)
/
Application

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

November 26, 2025: V 1.0

1 Architecture d'une application IoT

Une infrastructure d'IOT repose généralement sur au moins 3 acteurs :

① Un parc d'objets connectés fixes ou mobiles, répartis géographiquement,

② Un réseau telecom (filaire ou pas ou un peu ou beaucoup) qui va permettre de connecter les objets en transmettant des messages.

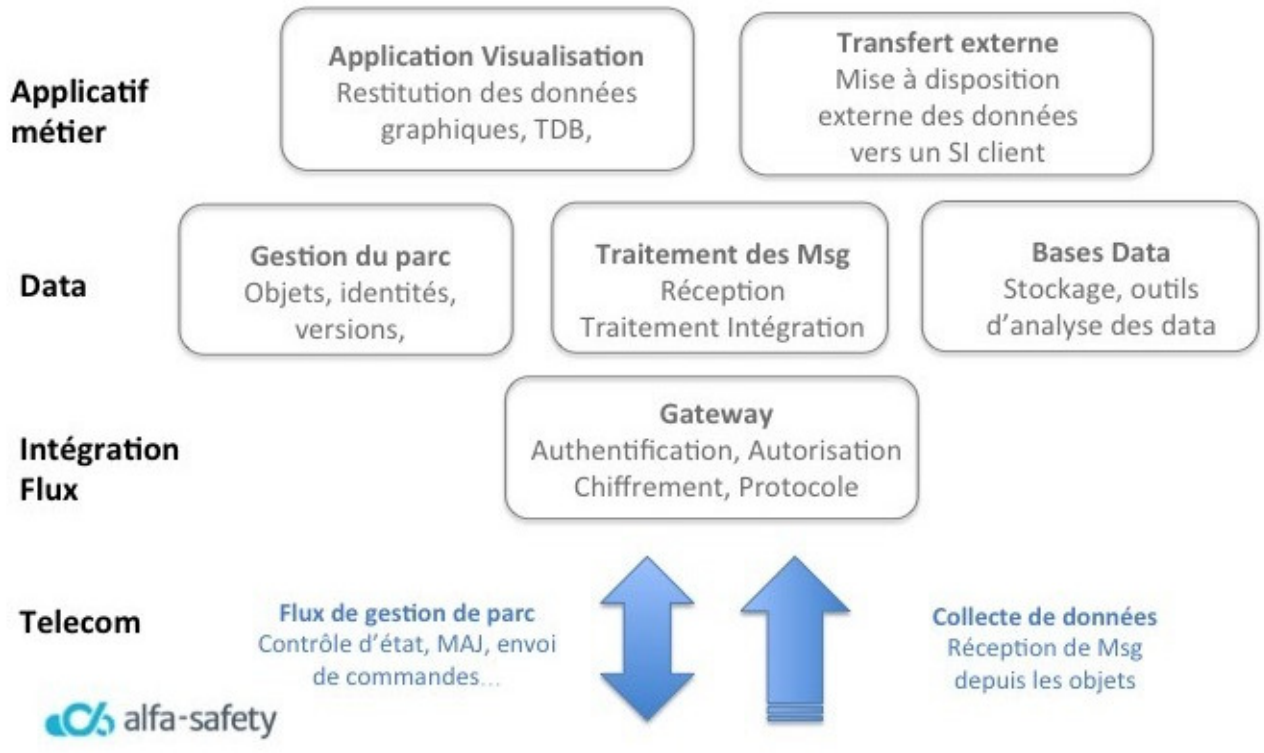
③ Une application qui collecte les "data" du réseau d'objets pour fournir une information agrégée plus ou moins "intelligente" ou pour contrôler un système (un parc éolien, ou un parc de piscines . . . par exemple).

L'application est aussi en charge de la gestion du parc d'objets

1.1 Fonctionnalités selon les acteurs

Une telle application comporte des fonctionnalités "classiques" réparties sur différents plans :

- métier,
- data,
- gestion des flux,
- télécommunications.



1.2 Gateway : intégration des flux

La fonction Gateway est la "porte d'échange" des messages entre l'application et le parc des objets, **elle s'interface avec le réseau de télécommunications** (via le support de protocoles "application" de type MQTT / HTTP / ...) et se charge tout particulièrement de la sécurité :

- ✔ Authentifier et autoriser les objets à dialoguer avec l'application.
- ✔ Contrôler l'intégrité des données et donc empêcher que l'on puisse injecter des jeux de données fictives à partir d'objets illégitimes (c'est le minimum !)
- ✔ Préserver la confidentialité des échanges par exemple par un chiffrement.

L'implémentation de la sécurité dépendra du protocole et du mode de dialogue retenu. Bien souvent cette sécurité viendra "alourdir" les coûts de traitements et de transmissions et par conséquent le coût énergétique.

- ✔ Il y a là des arbitrages à faire selon la nature de l'application.

En exploitation, on veillera à mettre en oeuvre une supervision applicative adaptée qui permettra de détecter toute anomalie des flux de messages : coupures de transmission, valeurs erratiques, ...

1.3 Le traitement des messages

Dans le contexte des traitements opérés sur l'information qui remonte des objets, **la question de la volumétrie est critique**.

- ✔ Même si à un instant donnée cette fonctionnalité semble correctement dimensionnée, elle (cette fonction) doit pouvoir absorber (réceptionner, traiter et intégrer) **un volume de messages potentiellement fluctuant**.

  "Fluctuant" parce les objets transmettent l'état du monde réel/physique et que si cet état évolue, le volume de données à de grande chance d'évoluer car les techniques d'échantillonnages sont souvent (très) réactives à des évenements ou à des variations.

Sur une échelle de temps "plus maitrisable" l'augmentation du parc d'objets engendre aussi une fluctuation de la volumétrie.

- ✔ Cette évolution peut remettre en cause l'architecture initiale : cf "le besoin MQTT" !

1.4 La gestion du parc d'objets

Un parc d'objets est en constante évolution : augmentation des capacités techniques, générations de matériels et de logiciels ...

- ✔ La gestion du parc est donc vital : inventaire, status, mise à jour ...

La taille du parc influe sur la complexité de la tâche et peut nécessiter une plate-forme de gestion.

Nous voyons un certain nombre de ces plates-formes IoT proposées par des fabricants de matériel qui les introduisent en complément ou dans le cadre de leur offre de matériel.

1.4.1 A "petite" échelle ...

De telles applications permettent des fonctionnalités IoT "limitées" :

- ➢ connecter des dispositifs et des capteurs au nuage/cloud,
- ➢ surveiller et collecter des données,
- ➢ et fournir une visualisation du projet IoT.

La domotique rentre typiquement dans cette catégorie d'applications :

- peu d'objets,
- peu de diversités d'objets,
- peu de réseaux,
- peu de protocoles,
- des distances réduites, ...

et en résumé, une complexité et une diversité restreinte.

Si on devait faire une analogie avec le monde des machines, **la domotique est l'équivalent d'une salle machines/PC telle que celle de TP** et on **mesure bien la complexité réelle MAIS limitée** d'une plateforme/application logicielle permettant la gestion d'une telle salle.

1.4.2 A "grande" échelle

Un tel scénario/configuration peut rapidement évoluer et se compliquer !

Si il y a plusieurs salles machines, sur des sites différents avec des réseaux hétérogènes, avec des machines de natures (Hardware/OS) différentes, avec des fonctions différentes (serveurs, gateway, ...) ... la gestion se complexifie !

Pour les scénarios IoT à grande échelle (Smartcity, Farming,...) on retrouve des problématiques (amplifiées) :

La maintenance du parc nécessite une véritable **plateforme de gestion** des objets.

- ➢ Cette plateforme permet la connexion/adhésion, la gestion, la mise à jour et l'intégration de milliers de périphériques et de capteurs.

1.5 Les bases de données

Le parc d'objets va alimenter l'application d'un flux données qui doit être analyser. Or, cette analyse est d'autant plus pertinente qu'elle s'appuie sur un horizon temporel adapté.

- ➢ Pour générer cet horizon, il faut "stocker" ... dans des Bases de Données.

1.6 Le serveur d'application

L'objectif d'une application d'IOT est de traiter/analyser puis de présenter les connaissances aux utilisateurs.

- ➢ Cette "connaissance" peut consister en des données brutes ou en des résultats d'analyses poussées.
- ➢ Souvent une présentation graphique synthétise cela graphiquement sur un "Dashboard".
- ➢ Le serveur d'application peut être accéder depuis le Web ou depuis un smartphone ou encore un outil dédié.

L'accès à ce serveur peut être limité si l'audience est ciblée, il peut devenir important sur une audience grand public. Là encore, attention au dimensionnement.

1.7 Les transferts externes de data

Un autre mode d'usage des données est de les **transférer vers le SI d'un client qui va à son tour les intégrer pour les exploiter dans son cas d'usage particulier** :

- ➢ Un opérateur de service IOT propose à ses clients de s'abonner à un service d'information basé sur son réseau d'objets, les données sont remontées dans le SI du client final qui va s'en servir pour produire ses propres services, comme une société de surveillance ou maintenance qui transmet certaines alarmes à un sous-traitant qui se charge d'intervenir sur site.

Enfin, on veillera ici encore à mettre en oeuvre une supervision applicative efficace des flux et notamment sur le plan des droit d'accès.

2 Les tentatives de standardisation

On a évoqué en cours la problématique de solutions IoT développées en silos . . . avec autant de "standards".

La complexité de la tâche de standardisation est grande notamment parce que le problème a plusieurs dimensions :

① Il y a les **"domaines d'applications"** représentés verticalement : villes, santé, énergie, . . .

Ils sont certainement anciens, éventuellement avec des contraintes spécifiques.

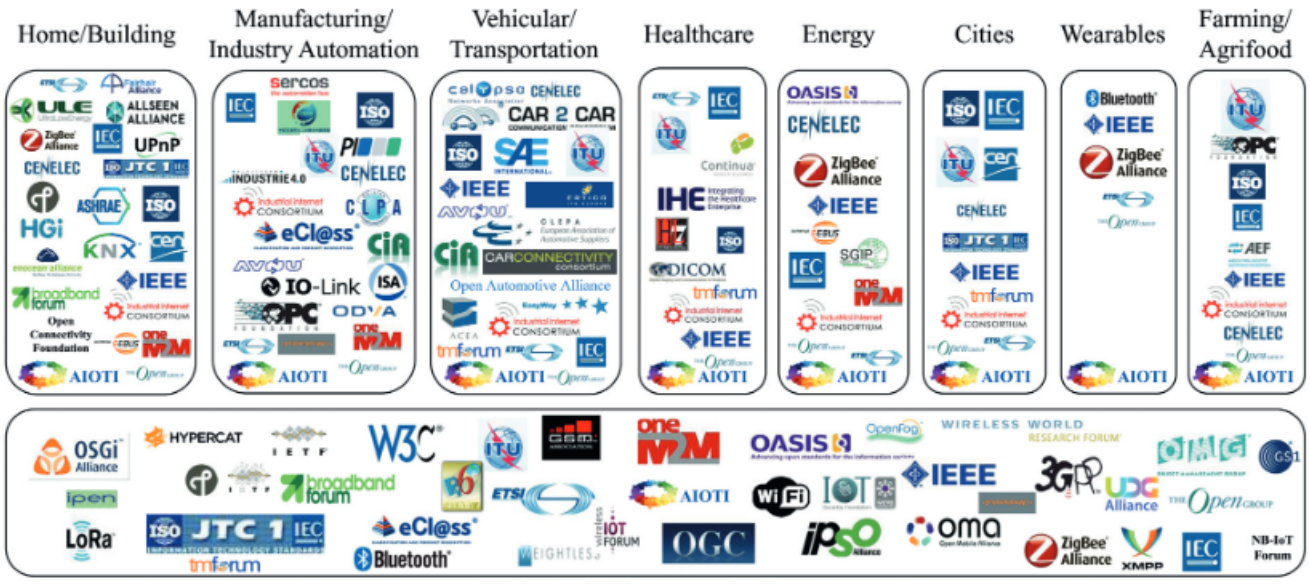
② Horizontalement, les infrastructures de télécommunications qui cherchent à répondre aux besoins et à capter ces flux.

Jusqu'au niveau où Internet met tout le monde d'accord, **les technologies et les protocoles sont multiples et potentiellement concurrentes**.

Leurs contraintes sont l'étendue, la consommation, le débit, le média, . . .

③ Et enfin il y a le "business", dimension sous jacente à tout "marché" qui fait par exemple que l'utilisateur se retrouve avec des dizaines de standards différents de prises USB ;-)

La figure qui suit liste quelques acteurs/intervenants dans ces domaines.



Deux types d'acteurs :

➢ **Standards Developing Organization** (SDOs) qui développent et proposent des standards (plutôt technologiques) de l'IoT,

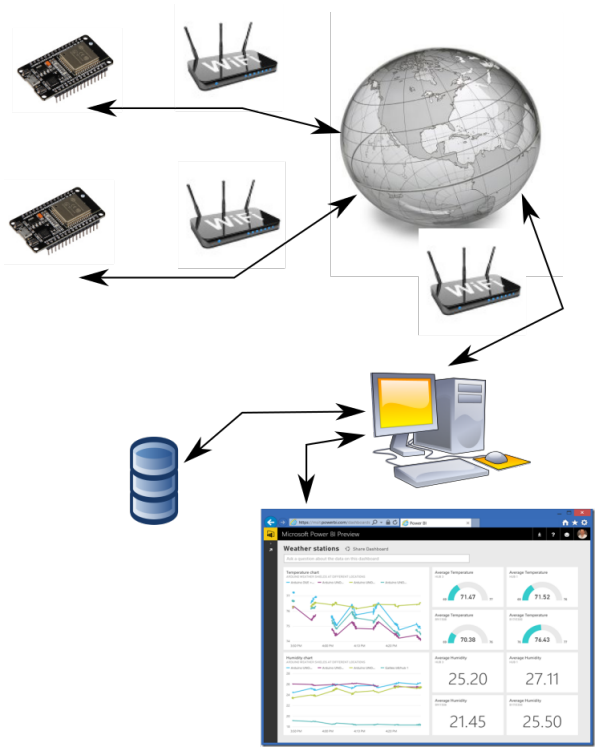
➢ et des **"alliances"** dont certaines ont bien compris qu'un standard pouvait aussi être "de fait" et qui peuvent servir (lorsqu'elles agglomèrent des industriels) des objectifs plutôt commerciaux, marketing, promotionnels, . . .

3 Une application "générique"

Pour aborder les problématiques informatiques du domaine de l'IoT pourquoi ne pas essayer de programmer une application un peu générique ?

On "reste" dans une architecture de type :

"des objets (ESP32) <-> 1 réseau <-> Service/Station de monitoring (PC)".



Par rapport aux réalisations précédentes, cette nouvelle application/architecture va, en plus, se focaliser sur la

① La persistance :

On voit apparaître sur la figure une base de données sans laquelle il n'est pas possible de consolider/construire un "savoir" (référence au sommet de la "pyramide des connaissances" dans le cours).

② Le déploiement d'un service dans le "cloud" :

Votre application doit désormais exister sur les ressources mises à disposition dans l'espace Internet.

Son accessibilité devra être universelle et permanente (24H/24)!

4 "Water BnB"

Pour ce sujet, vous voilà propriétaire d'une belle villa avec piscine "connectée" **et un grand portail**.

Mais vous l'utilisez au final assez peu car vous êtes souvent sur votre yacht !

Vous décidez de créer un service qui permette de louer/utiliser une piscine pendant l'absence de ses propriétaires.

- Il y a pas mal de difficultés à résoudre pour obtenir un "produit" mais vous allez vous focaliser sur la gestion de l'"accès à la piscine".

C'est le sujet du TP !

Le cas d'utilisation (use case) est le suivant :

1. Un utilisateur du service WaterBnB dispose d'un tableau de bord (NodeRed) montrant des piscines environnantes.

   Derrière chaque piscine, il y a un ESP qui publie son status sur le topic "**uca/iot/piscine**" et donne son "name" à la piscine (sur la base du champs "ident" du schéma Json).

2. Il souhaite louer une d'elles pour faire une petite baignade.

   La demande de location se fait un cliquant sur la piscine sélectionnée sur le tableau de bord.

3. L'accès est autorisé (ou pas) par le service (payant) WaterBnB que vous allez/devez développer dans le cloud (`render.com`)

   C'est lui qui ouvre le portail et facture la prestation !

Un conseil :

- Lisez bien tout le sujet avant de commencer . . . pour avoir une vue globale !

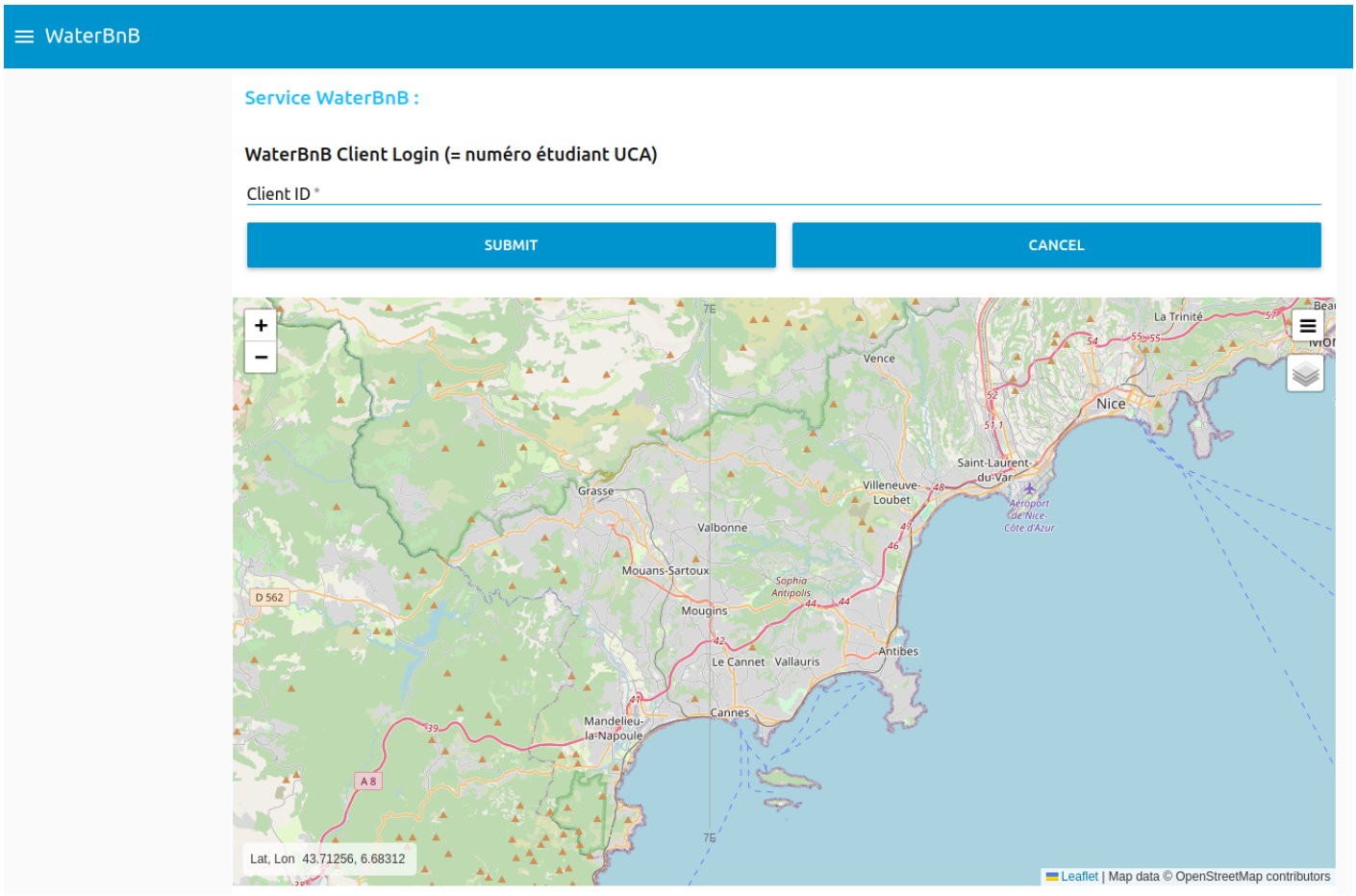
4.1 Interface d'accés au service

Depuis la fin du Step4.2, vous disposez d'un dashboard où vous voyez les piscines avoisinantes avec un indicateur de "hotspot" (en rouge ou en bleu selon le résultat du calcul).

Vous allez réaliser deux évolutions pour que ce tableau de bord deviennent une interface (graphique) d'accès au service WaterBnB :

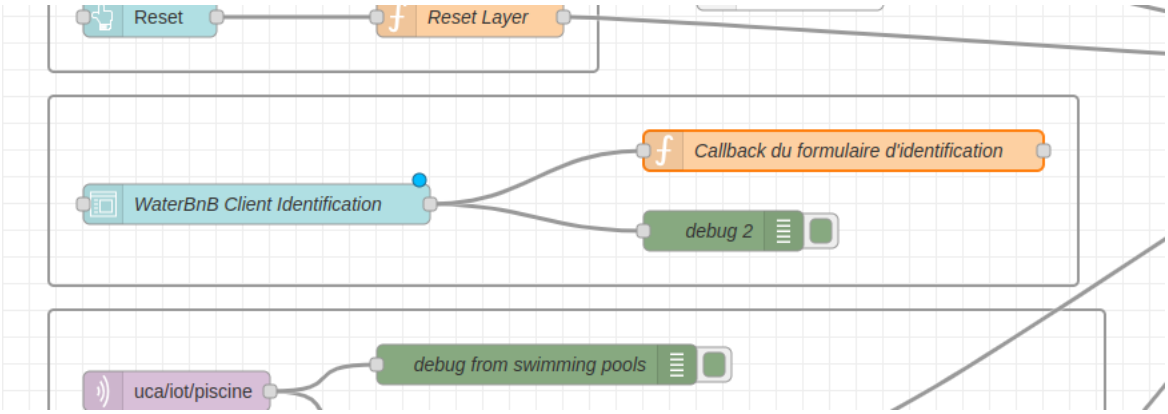
4.1.1 Identification du client

La première évolution consiste à pouvoir définir une identification de la personne qui utilise ce tableau de bord.

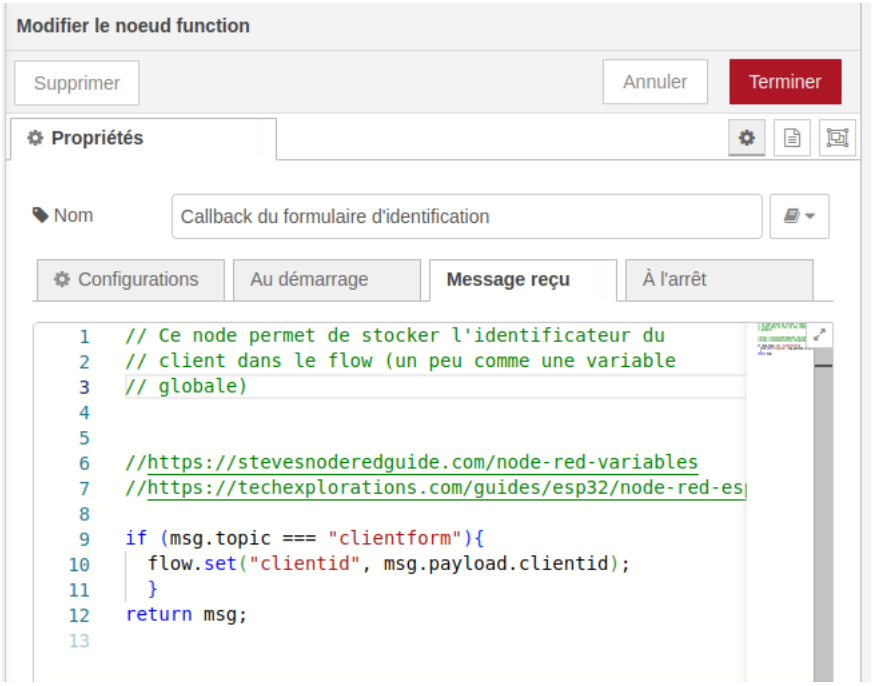


Cette identification correspond à son "nom/login de client" pour le service.

1. Pour obtenir la ligne de saisie (au dessus de la carte), j'ai rajouté dans le flow NodeRed un noeud de formulaire qui permet de spécifier un identifiant au niveau du tableau de bord.



2. Le noeud de fonction qui le suit définit un callback appelé par la soumission du formulaire.



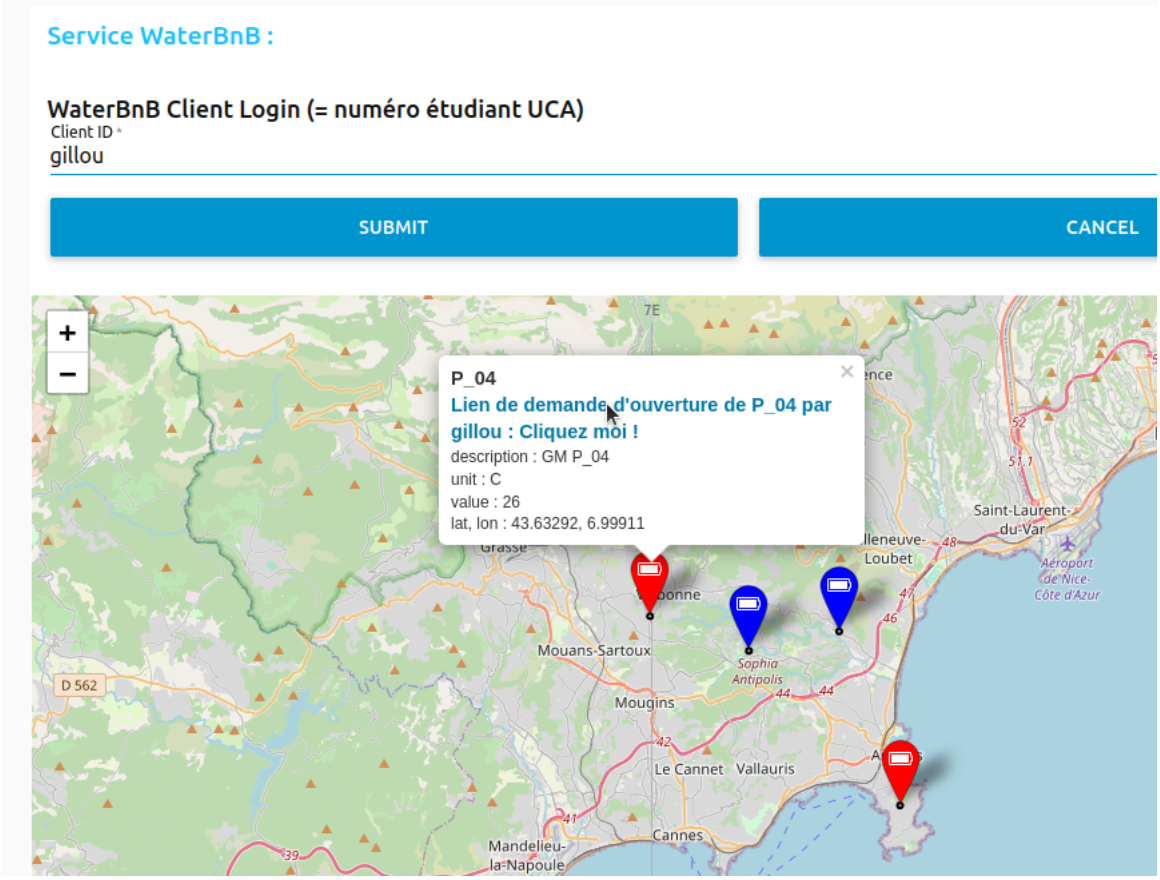
Dans cette fonction, la fonction "**set**" appliquée au flow **permet de définir une variable globale (au flot NodeRed)**.

- L'identificateur de cette variable est "cliendid" !

Cette variable contient le nom de l'utilisateur du service.

4.1.2 Demande d'accés identifié

Au niveau du marqueur de chaque piscine sur la carte, j'ai rajouté (via la noeud "json adapt") un hyperlien dans le menu.
Le menu et l'hyperlien sont visualisables en cliquant sur l'icône d'une piscine :



Pour l'instant, cet hyperlien pointe vers l'URL "`http://localhost:1880/open`" qui correspond au noeud nodered "open" définit plus bas dans le flot . . . cela permet de deboguer !

Par contre, en la complétant avec les "bons" arguments, voilà un moyen de demander par une requête GET

- à un serveur dans le cloud . . . (qui prendra la place de localhost),
- de laisser le clientid (contenu dans la variable du flot NodeRed)
- entrer dans la piscine identifiée par le clic souris.

Ceci sera "la demande d'accés" à la piscine !

5 TODO 1 : Autorisation d'accés

5.1 Finir le marqueur de la piscine

Votre travail consiste donc à améliorer l'URL de cet hyperlien pour qu'il effectue la requête HTTP GET avec la route "/open" au **serveur WaterBnB** avec deux paramètres :

- "idu" qui a comme valeur le client du service (donc la valeur de clientid)
- "idswp" qui a comme valeur le nom du propriétaire de la piscine (donc le champ info.ident du JSON).

  On rappelle que **les piscines sont identifiées en "P_numeroetudiant"**.

Je vous laisse proposer une solution pour choisir l'adresse IP de l'URL. Cela peut aller de la constante dans le noeud NR à . . . ???

5.2 Un serveur . . . doit répondre !

Le serveur WaterBnB doit être hébergé dans le cloud pour être accessible 24H/24.

Je choisis dans ce qui suit la solution d'un serveur "en Python/Flask s'exécutant sur https://render.com et connecté à une base de donnée hébergée sur Mongo Atlas (le cloud mongoDB : https://account.mongodb.com/account/login)"

Pourquoi Flask plutôt que Django ? => montée en compétence plus rapide :

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

Si ca ne vous convient pas et que vous préférez faire du JS ou . . . sur un autre cloud : why not !? mais le service doit être accessible 24H/24 ! . . . et le code de qualité.

5.2.1 render.com

"https://render.com" fait partie des clouds qui permettent de construire et de déployer une application . . . un serveur d'application par exemple !

- Il peut être gratuit (ou pas selon vos contraintes) . . . ce que n'est plus "https://heroku.com" :-(
- Il supporte un grand nombre de langages dont Python.
- Il se connecte bien avec Github, et avec le cloud MongoDB Atlas.
- Au global, je trouve qu'il est plutôt bien fait.

On commence par créer un repository GitHub : "WaterBnB_numéroétudiant"

- Je n'en dis pas plus sur GitHub . . . j'espère que vous savez faire ! (cela sort du périmètre de l'IoT)

Je devrais pouvoir accéder à votre repo avec un lien de la forme :

https://github.com/login_git/WaterBnBF_numeroetudiant

Pour éviter le plagiat, le repo reste privé ET vous m'invitez ! (=> gmenez sous GitHub)

On va connecter ce repository à render dans ce qui suit.

On revient à render :

1. Vous créez un compte (gratuit) sur https://render.com/
2. Vous jetez un oeil aux tutoriaux/documentation dans le menu Docs/ Render QuickStarts :
   - Pour un app python flask : https://render.com/docs/deploy-flask
   - Pour une app js express : https://render.com/docs/deploy-node-express-app
3. Dans votre dashboard, vous créez un "New => Web Service"
4. Qui sera "Build and Deploy from a Git repository" !
   Vous utilisez celui que vous venez de créer !
5. Vous utilisez les paramètres d'une app Python Flask évoqués dans le QuickStart "Python Flask".

Vous devriez vous retrouver avec un service en Python Flask dont l'url ressemblera à :

https://waterbnb_numeroetudiant.onrender.com

Malheureusement avec un cloud gratuit ... il est possible que le service soit swappé et donc éteint ... faut juste le temps de le redémarrer !

Le service peut proposer une page de front (par exemple : https://waterbnbf.onrender.com) **MAIS** il doit aussi et surtout décider de donner, ou pas, l'accès à la piscine en répondant par la positive (ou la négative) à une demande d'ouverture de la porte d'accés.

5.2.2 Flask

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

Je vous donne de quoi démarrer dans : https://github.com/gmenez/WaterBnB

- Il y a des illustrations de l'utilisation de Flask, mais aussi de MQTT et de MongoDB.
- Le login du cluster mongodb est : visitor
- Le mot de passe du cluster mongodb est : doliprane
- Pour les "visitor", la base est read only
- Pour les connexions réseaux, il faut me demander d'inscrire votre IP comme ayant droit.

5.2.3 Mongo Atlas

Il vous faut créer un compte (gratuit) sur le cloud mongo (https://cloud.mongodb.com/ ou https://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/

Dans votre Cluster/Project sur le cloud vous devait créer une base "WaterBnB" et au moins une collection "users".

5.3 Accès à la piscine

5.3.1 Les conditions d'accès

Pour qu'un utilisateur "toto" du service WaterBnB puisse avoir accès à la piscine "P_123456" (sur laquelle il vient de cliquer), il faut plusieurs conditions :

1. "toto" doit être un utilisateur déclaré du service dans le cloud.

   Dans le code que je vous donne, vous avez accès à un fichier qui sert au remplissage de la collection "users" de la base WaterBnB.

2. La piscine doit exister **et** elle doit être "non déjà occupée".

   Ces informations on les a ! . . . grâce au flux MQTT ("uca/iot/piscine") des piscines.

   Dans le code que je vous donne, il y un exemple de manipulations de MQTT.

5.3.2 Les effets de l'accés

Tout se joue sur la Led Strip de l'ESP !

1. Si elle est verte alors la piscine est disponible à la location. C'est la couleur avant la demande d'ouverture.
2. Si elle est jaune alors la piscine est désormais occupée (elle le reste tant que le capteur de lumière le dit)
3. Si elle est rouge alors l'accès est refusé. Elle reste dans cet état pendant 30 secondes puis revient à vert.

A vous de réaliser ce retour d'information vers l'ESP depuis le serveur d'app !

Rmq : Je pense qu'on peux aussi utiliser un retour du clic sur le dashboard . . .

6 TODO 2 : Exploitation des données

Toute application IoT digne de se nom doit comporter une synthèse des connaissances . . . déduites de ses données.

Mongo Atlas permet de réaliser des tableaux de bords à partir des données stockées :

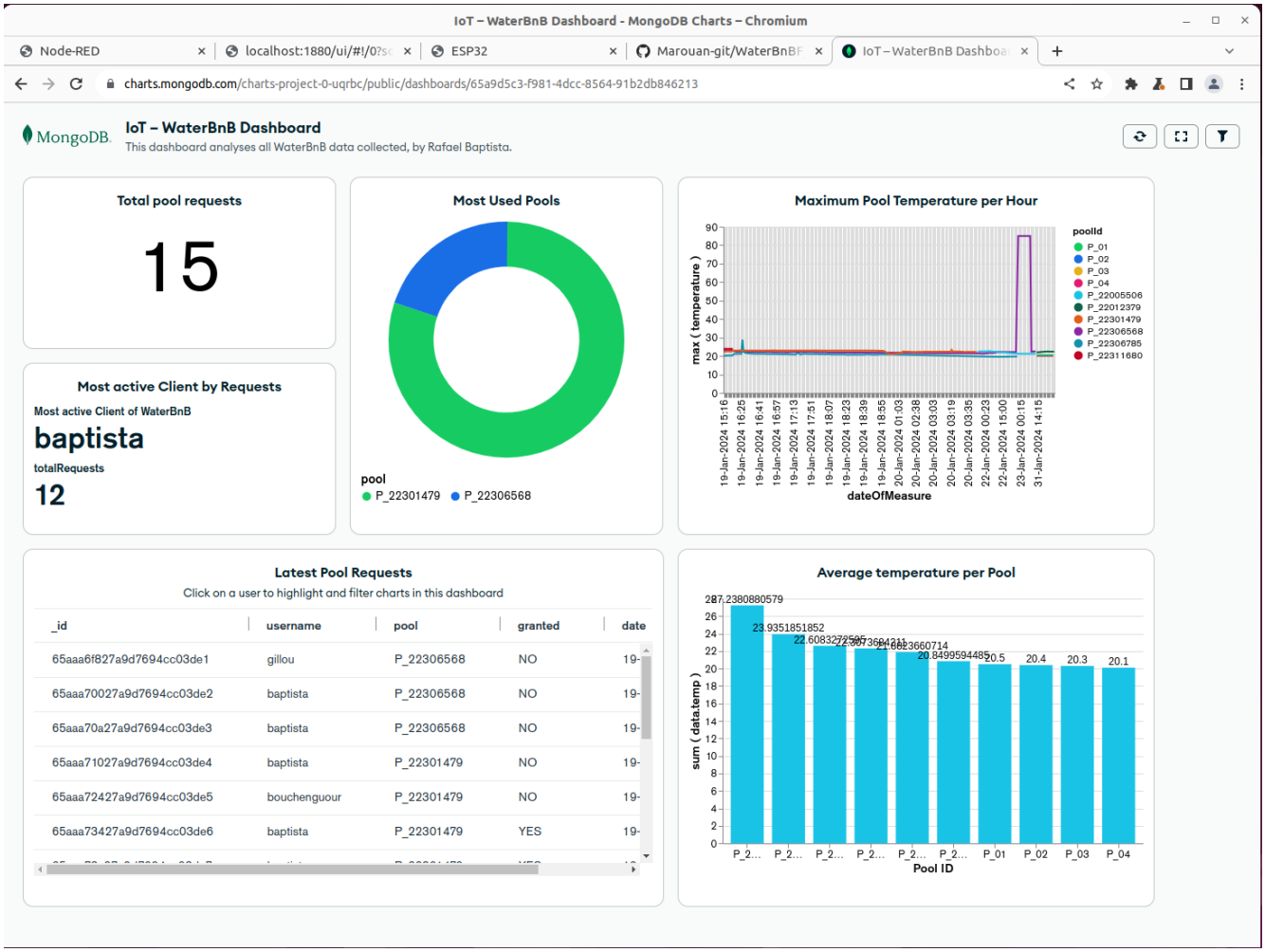
https://charts.mongodb.com/charts-project-0-wdlbq/public/dashboards/6579b0fa-11d0-477b-846d-2ffda2aaba29#

Vous reconnaissez ces numéros ?

J'aimerais en savoir plus . . . par exemple (au minimum), un histogramme des demandes : quels sont les piscines les plus demandées ?

Tout ce qui a un sens m'intéresse . . . mais il est clair que c'est "juste" de la manipulation d'un outil donc pas la peine d'en faire des tonnes . . . mais quand même ;-)

Voila un exemple réalisé par un étudiant :



Il va de soit que la richesse de ce dashboard dépend de ce que vous avez mis en base.