Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Filière Systèmes industriels

Orientation Infotronics

Travail de bachelor
Diplôme 2021

Lenny Favre

Web of Things (WoT) pour bâtiments intelligents

Professeur
Dominique Gabioud

Expert
Christoph Brönnimann

Date de la remise du rapport
20.08.2021, 12 :00

Ce rapport est l'original remis par l'étudiant.
Il n'a pas été corrigé et peut donc contenir des inexactitudes ou des erreurs

HES-SO Valais-Wallis • Campus Energypolis, rue de l'Industrie 23 • C.P. • 1950 Sion
+41 27 606 85 16 • info.synd@hevs.ch • www.hevs.ch/synd

HES-SO Valais

SYND	ETE	TEVI
X	X	X

Données du travail de diplôme
Daten der Diplomarbeit

FO 1.2.02.07.CB
haj/11/01/2016

Filière / Studiengang **SYND**	Année académique / *Studienjahr* **2020/21**	No TD / *Nr. DA* **IT/2021/48**
Mandant / *Auftraggeber* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Lenny Favre** Professeur / *Dozent* **Dominique Gabioud**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja [1] ☒ non / nein	Expert / *Experte* **Christoph Brönnimann** Vereinung SmartGridready, Falkenplatz 11, Postfach, 3001 Bern christoph.broennimann@smartgridready.ch	

Titre / *Titel*

Web of Things (WoT) pour bâtiments intelligents

Description / *Beschreibung*

Les technologies IoT (*Internet of Things*) sont aujourd'hui matures, si bien qu'il est possible de déployer des systèmes IoT avec des coûts d'investissement et d'exploitation quasi nuls. On voit ainsi se multiplier des solutions intégrées verticalement (depuis une *thing* jusqu'à son interface web) de type « silos » (à chaque type de *thing* sa solution distincte). Toutefois, la multiplication de systèmes parallèles dégrade l'expérience pour les utilisateurs : besoin de gérer des interfaces multiples, difficulté pratique d'intégrer plusieurs *things* dans une même solution.
Pour augmenter l'interopérabilité dans l'IoT, le W3C (*Word Wide Web Consortium*) définit une série de standards sous la dénomination WoT (*Web of Things*).

Le projet consiste à développer un démonstrateur WoT selon le scénario suivant : plusieurs systèmes IoT sont installés dans un bâtiment, chaque système est rendu compatible avec l'architecture WoT, une plateforme intermédiaire fournit à des applications un accès uniforme à l'ensemble des systèmes IoT.

Objectifs / *Ziele*

- Prendre connaissance de l'architecture WoT et expérimenter son implémentation de référence THINGWEB (http://www.thingweb.io/).
- Définir un environnement de test « bâtiment » avec plusieurs systèmes IoT et rendre ces systèmes individuellement compatibles avec WoT.
- Développer une plateforme offrant un accès uniforme aux systèmes IoT du bâtiment selon le modèle « *Intermediary* » de l'architecture WoT.
- Développer une application de démonstration.

Signature ou visa / *Unterschrift oder Visum*	Délais / *Termine*
Responsable de l'orientation / filière *Leiter der Vertiefungsrichtung / Studiengang:* .. [1] Etudiant / *Student* :	Attribution du thème / *Ausgabe des Auftrags:* **10.05.2021** Présentation intermédiaire / *Zwischenpräsentation* **07 – 08.06.2021** Remise du rapport / *Abgabe des Schlussberichts:* **20.08.2021, 12:00** Exposition / *Ausstellung der Diplomarbeiten:* **25 – 27.08.2021** (si autorisé / *falls genehmigt*) Défense orale / *Mündliche Verfechtung:* **30.08 – 09.09.2021**

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.*
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

Web of Things (WoT) pour bâtiments intelligents

Diplômant/e Lenny Favre

Objectif du projet

Le travail consiste à développer un composant intermédiaire entre les appareils intelligents dans les bâtiments et les applications de gestion du bâtiment. Ce composant offre aux applications un accès uniforme aux appareils intelligents, même si ceux-ci ont des interfaces différentes. Il est basé sur le standard WoT du W3C, qui vise à promouvoir l'interopérabilité dans le monde IoT.

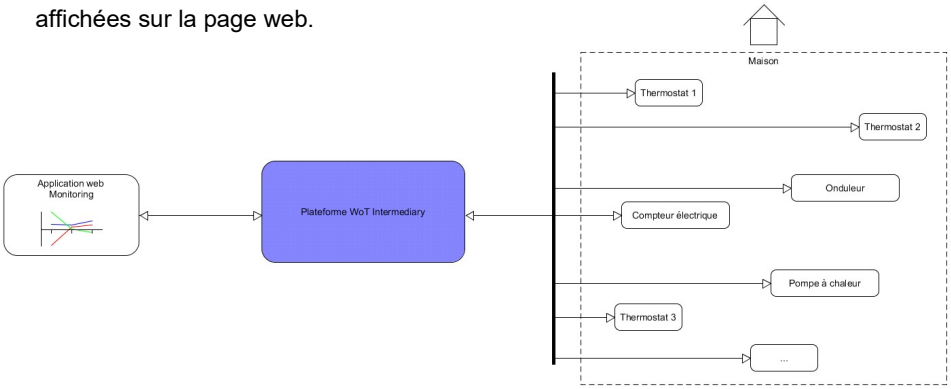
Méthodes | Expériences | Résultats

La plateforme est écrite en Javascript à l'aide de la libraire node-wot de thingweb. Elle agit comme un proxy et comme une passerelle réseau pour les appareils connectés.

Les accès aux différentes interactions avec les systèmes sont limités en fonction des droits de l'application. Les messages retournés par la plateforme intermédiaire peuvent être standardisés si nécessaire.

Un environnement de bâtiment IoT a été développé pour tester la plateforme développée. Un script teste toutes les fonctions de celle-ci en faisant des requêtes valides ou erronées. Ce script n'a retourné aucune erreur.

Un système de démonstration a été développé pour facilement présenter le projet et ces fonctionnalités. Il s'agit d'une page web qui communique avec des Things réelles d'une maison à travers la plateforme. On appelle Things, les appareils connectés comme, dans cette maison, des compteurs électriques, des thermostats, etc. Les différentes valeurs des capteurs sont monitorées et affichées sur la page web.



Une maison est peuplée d'appareils connectés. La page web de monitoring communique avec ces appareils à travers la plateforme et affiche les valeurs mesurées.

Source: Web of Things, Documentation, W3C

Travail de diplôme
| édition 2021 |

Filière
Système industriel

Domaine d'application
Infotronics

Professeur responsable
Dominique Gabioud
dominique.gabioud@hevs.ch

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Web of Things (WoT) für intelligente Gebäude

Diplomand/in Lenny Favre

Ziel des Projekts

Die Arbeit besteht in der Entwicklung einer Zwischenkomponente zwischen den intelligenten Geräten in Gebäuden und den Gebäudemanagementanwendungen. Diese Komponente ermöglicht Anwendungen einen einheitlichen Zugang zu intelligenten Geräten, auch wenn diese unterschiedliche Schnittstellen haben. Es basiert auf dem W3C WoT-Standard, der die Interoperabilität in der IoT-Welt fördern soll.

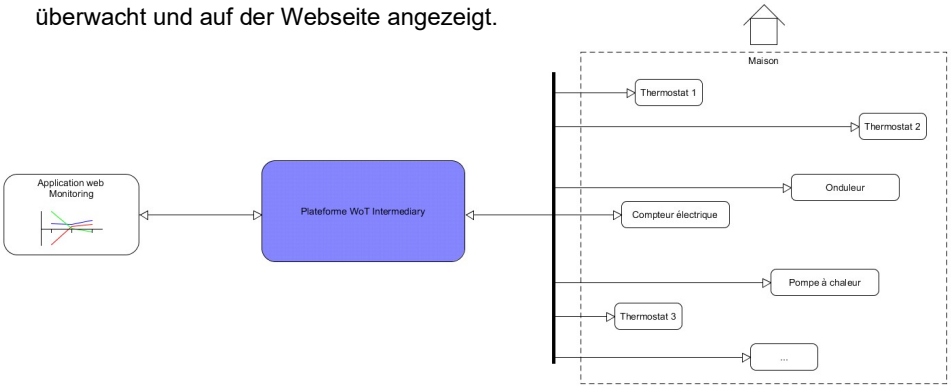
Methoden | Experimente | Resultate

Die Plattform ist in Javascript geschrieben und verwendet die node-wot-Bibliothek von thingweb. Er fungiert als Proxy und als Netzwerk-Gateway für die angeschlossenen Geräte.

Der Zugang zu den verschiedenen Interaktionen mit den Systemen ist entsprechend den Rechten der Applikation eingeschränkt. Die von der Zwischenplattform zurückgegebenen Nachrichten können bei Bedarf standardisiert werden.

Es wurde eine IoT-Gebäudeumgebung entwickelt, um die entwickelte Plattform zu testen. Ein Skript testet alle Funktionen der Plattform, indem es gültige oder fehlerhafte Anfragen stellt. Dieses Skript hat keine Fehler geliefert.

Es wurde ein Demonstrationssystem entwickelt, um das Projekt und seine Funktionalitäten auf einfache Weise zu präsentieren. Es handelt sich um eine Webseite, die über die Plattform mit realen Dingen in einem Haus kommuniziert. Wir nennen die Dinge, die angeschlossenen Geräte, wie in diesem Haus, Stromzähler, Thermostate, usw. Die verschiedenen Werte der Sensoren werden überwacht und auf der Webseite angezeigt.



Ein Haus ist mit angeschlossenen Geräten bevölkert. Die Überwachungswebseite kommuniziert mit diesen Geräten über die Plattform und zeigt die gemessenen Werte an.

Quelle: Web of Things, Documentation, W3C

Diplomarbeit
| 2021 |

Studiengang
Systemtechnik

Anwendungsbereich
Infotronics

Verantwortliche/r Dozent/in
Dominique Gabioud
dominique.gabioud@hevs.ch

Informations sur ce rapport

Contact

Auteur : Lenny Favre
Etudiant en Bachelor
HES-SO//Valais Wallis
Suisse
Courriel : favre.le@hotmail.com

Déclaration d'honneur

Je, soussigné, Lenny Favre, déclare que le travail rendu est le résultat d'un travail personnel. Je certifie que je n'ai pas eu recours au plagiat ou autres formes de fraudes. Toutes les sources d'informations utilisées et les auteurs des citations sont clairement mentionnés.

Lieu, date : Sion, le 20 août 2021

Signature : _____

Validation

Accepté par la HES-SO//Valais Wallis (Suisse, Sion) sur formulaire :

Prof. Dominique Gabioud, Professeur responsable
Christoph Brönnimann, Expert

Lieu, date : _____

Prof. Dominique Gabioud

Responsable HEI-VS//Systèmes industriels

Remerciements

Je souhaite remercier mon professeur responsable, Monsieur Gabioud Dominique ainsi que Monsieur Meyer Martin pour leur soutien et suivi tout au long de ce projet. Je tiens aussi à remercier les personnes travaillant sur le projet domOS s'étant intéressées à ce projet, notamment Monsieur Dongo Junior. Je voudrais encore remercier ma famille et mes amis qui m'ont soutenu pendant ce projet et toute la durée de mes études à la HES-SO//Valais Wallis.

Table des matières

Table des illustrations

1 Introduction

1.1 Contexte

Depuis plusieurs années, les nouveaux appareils électroniques sortant sur le marché ont une tendance à être connectés. La multiplication de ces appareils interconnectés via l'Internet donne naissance à l'Internet of Things (IoT). Ces appareils, que l'on nomme Things, se sont diversifiés et complexifiés au point que chacun de ces appareils a une manière différente de communiquer des informations, ce qui n'est pas interopérable. Cette fragmentation rend difficile l'intégration ou la réunion de nouveaux systèmes. L'ajout d'un nouvel appareil connecté dans un système IoT devient alors coûteux en ressources.

Pour résoudre ce problème, des normes et standards peuvent être créer pour les plateformes IoT. L'organisation W3C s'efforce à développer des standards Web. Le Web of Things (WoT) du W3C cherche à éviter la fragmentation des systèmes IoT[1]. Ce standard permet la découverte des Things via les technologies Web et l'interopérabilité de ces systèmes. Encore naissant, il n'a pas été appliqué dans beaucoup de systèmes.

Cependant, chaque Thing n'utilise généralement qu'1 seul protocole de communication. Par exemple, dans un bâtiment, il existe plusieurs Things utilisant des protocoles différents. Si une application veut communiquer avec plusieurs de ces Things, elle doit donc connaître tous les protocoles différents. Elle doit donc pouvoir récupérer et comprendre les données de chaque protocole. Pour ce faire, une application devra investir beaucoup de ressources pour pouvoir atteindre ces buts. Pour chaque Thing différente, l'application devra créer un module de communication et de traduction.

1.2 Objectifs

La partie Architecture du standard WoT décrit l'architecture abstraite et les profils d'interopérabilité pour le WoT[2]. Le but de ce projet est de créer une plateforme de type Intermediary défini dans l'architecture WoT. Elle agira comme le point d'accès central pour les différents systèmes IoT présents dans un bâtiment. Similairement à une passerelle réseau, cette plateforme devra pouvoir gérer la communication entre application et Thing. Quel que soit le protocole de communication et les formes des requêtes des Things, une application devra pouvoir accéder uniformément aux Things à travers la plateforme.

[1] W3C, W3C Web of Things [en ligne], 2021, https://www.w3.org/WoT/ (consulté en août 2021)
[2] W3C, W3C WoT – Architecture Task Force [en ligne], 2021, https://www.w3.org/WoT/activities/tf-architecture/ (consulté en août 2021)

2 Situation initiale

Pour pouvoir comprendre ce rapport dans son ensemble, il est nécessaire de comprendre les standards utilisés. Cette partie explique en quelques mots le standard Web of Things. De plus, elle présente les travaux précédemment réalisés qui ont aidé au développement de ce projet.

2.1 Standard Web of Things

L'organisation W3C développe des standards Web. Ce projet est réalisé conformément aux standards Web of Things du W3C. Ils permettent une intégration facile aux plateformes IoT et aux domaines d'applications[3].

2.1.1 Thing

Une Thing est un objet physique composé de capteurs, actionneur ou programmes pouvant échanger des données sur le réseau internet. La multiplication de ces Things donne naissance à l'internet of Things (IoT). Dans le standard WoT, une Thing est décrite par un document JSON, la Thing Description.

2.1.2 Thing Description

La WoT Thing Description (TD) est le bloc central dans le W3C Web of Things[4]. La TD est le point d'entrée à une Thing, comme le index.html d'un site web. Il prend la forme d'un fichier JSON. Il est comporté de 4 principaux composants : des métadonnées à propos de la Thing, les interactions qui indiquent comment on peut utiliser cette Thing, des schémas de données échangées par la Thing et les liens Web pour exprimer des relations avec d'autres Things/documents.

[3] W3C, W3C Web of Things [en ligne], 2021, https://www.w3.org/WoT/ (consulté en août 2021)
[4] W3C, Web of Things (WoT) Thing Description [en ligne], 2021, https://www.w3.org/TR/wot-thing-description/ (consulté en août 2021)

```
{
    "title": "myCounter",
    "description": "TD for counter",
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:counter",
    "securityDefinitions": { "nosec_sc": { "scheme": "nosec" }},
    "security": "nosec_sc",
    "base": "http://localhost:8081/mycounter/",
    "properties": {
        "count": {
            "type": "integer",
            "readOnly": true,
            "description" : "Displays the value of the counter",
            "forms": [{ "href": "count",
                        "op": "readproperty",
                        "htv:methodName": "GET",
                        "response": {"contentType": "application/json"}}]
        }
    },
    "actions": {
        "increment" : {
            "forms": [{ "href": "increment",
                        "htv:methodName": "GET"}],
            "idempotent":false,
            "safe": false,
            "description":"Increments the counter"
        },
        "decrement" : {
            "forms": [{ "href": "decrement",
                        "htv:methodName": "GET"}],
            "idempotent":false,
            "safe": false,
            "description":"Decrement counter"
        },
        "reset" : {
            "forms": [{ "href": "reset",
                        "htv:methodName": "GET",
                        "response": {"contentType": "application/json"}}],
            "idempotent":true,
            "safe": false,
            "description":"Resets the counter"
        }
    },
    "events": {
        "changed": {
            "description" : "Counter value changed",
            "data": {
                "type": "integer"
            },
            "forms": [{
                "op": "subscribeevent",
                "href": "changed",
                "htv:methodName": "POST",
                "contentType": "application/json"
            },{
                "op": "unsubscribeevent",
                "href": "changed",
                "htv:methodName": "DELETE",
                "contentType": "application/json"
            }]
        },
        "reset": {
            "description" : "Counter value has been reset",
            "data": {
                "type": "null"
            },
            "forms": [{
                "op": "subscribeevent",
                "href": "reset",
                "htv:methodName": "POST",
                "contentType": "application/json"
            },{
                "op": "unsubscribeevent",
                "href": "reset",
                "htv:methodName": "DELETE",
                "contentType": "application/json"
            }]
        }
    }
}
```


Figure 1: Thing Description for a counter

La figure 1 présente un exemple de Thing Description. Il s'agit d'un compteur communiquant en HTTP sans modèle de sécurité. Il possède une propriété count. Ses actions sont incrémenter, décrémenter et remettre à zéro. Il renvoie des évènements lorsque que la valeur de count est changée ou remise à zéro.
Ce fichier contient des métadonnées comme l'ID de la Thing, un titre, une description textuelle de la Thing, etc. Il possède aussi une définition du modèle de sécurité. Les identifiants, clés API, token d'accès ne sont cependant pas listés dans ce document

puisqu'il est public. Sous les clés properties, actions et events, sont décrites les interactions avec cette Thing. Chaque interaction contient un ou plusieurs objets sous la clé « forms » qui décrivent la requête et la réponse pour effectuer cette interaction.

2.1.3 WoT Architecture

Le but de ce projet est de construire une plateforme Intermediary selon l'architecture WoT. Un Intermediary agit comme un proxy pour les Things. Il consomme une ou plusieurs Things. Lorsqu' un Intermediary est accédé par des consommateurs, il leur montre sa Thing Description. Il est donc reconnu comme une Thing.

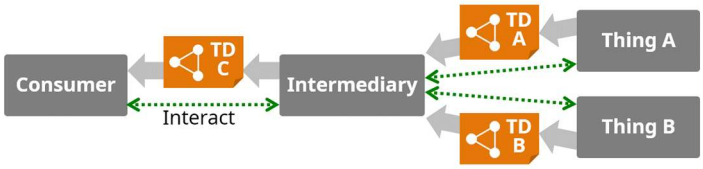


Figure 2: WoT Intermediary
Source: Intermediary, Web of Things (WoT) Architecture 1.1, W3C[5]

Les consommateurs accèdent à l'Intermediary avec seulement 1 protocole de communication avec 1 modèle de sécurité. Ceux-ci sont indépendant des Things connectées[6].

Lorsqu'un consommateur veut accéder à une ressource d'une Thing, il effectue une requête à l'Intermediary. Celui-ci traduira ensuite la requête et son contenu dans le protocole de communication propre de la Thing. Il utilisera le modèle de sécurité correspondant. La réponse de la Thing est ensuite retraduite par l'Intermediary et transmise au consommateur.

2.2 Librairie thingweb.node-wot

Thingweb est une implémentation du modèle WoT discuté dans le groupe de travail W3C WoT[7]. Node-wot est une librairie simple avec des applications de démonstrations permettant de créer et d'expérimenter avec des applications WoT[8]. L'API de scripting WoT consiste en trois parties principales :

- L'objet WoT, qui est le point d'entrée pour découvrir, consommer ou exposer des Things
- L'interface ConsumedThing qui est une instance d'une Thing consommée.
- L'interface ExposedThing qui est une instance d'une Thing exposée sur le réseau

Cette librairie en Typescript a comme fonction principal de communiquer des informations avec les Things selon leur protocole de communication et modèle de sécurité. Elle est encore en développement et peut être trouvée sur le GitHub de Eclipse[9].

---

[5] W3C, Web of Things (WoT) Architecture 1.1, Intermediaries [en ligne], 2021, https://w3c.github.io/wot-architecture/#intermediaries (consulté en août 2021)
[6] W3C, Web of Things (WoT) Architecture 1.1, Intermediaries [en ligne], 2021, https://w3c.github.io/wot-architecture/#intermediaries (consulté en août 2021)
[7] W3C, W3C WoT – Working Group [en ligne], 2021, https://www.w3.org/WoT/wg/ (consulté en août 2021)
[8] Thingweb, Page d'acceuil [en ligne], 2021, http://www.thingweb.io/ (consulté en août 2021)
[9] Eclipse, GitHub repo thingweb.node-wot [en ligne], 2021, https://github.com/eclipse/thingweb.node-wot (consulté en août 2021)

2.3 Projet similaire

Pendant le développement de ce projet, un travail similaire à celui-ci a été trouvé. Il s'agit du travail de master « Universal Explorer for the Web of Things » de Linus Schwab de l'université de Berne[10]. Il avait pour but de créer une passerelle pour les Things HTTP utilisant les TD du W3C ou de Mozilla. Avec les TD reçues, l'Universal Explorer peut reconnaître les Things. Il fournit une API RESTful HTTP avec une description OpenAPI et une API WebSocket pour la gestion des évènements. La structure des composants et le rôle de différentes classes développés ici ont été inspirés par ce travail de master.

[10] SCHWAB, Linus. Universal Explorer for the Web of Things [en ligne], 2018, https://www.unifr.ch/inf/softeng/en/assets/public/files/research/students_projects/master/Master_Schwab_Linus.pdf (consulté en août 2021)

3 Analyse

Avec la donnée de base, nous pouvons lister les notions importantes que la plateforme devra utiliser. Ce chapitre liste les objectifs qu'elle doit remplir.

3.1 Multiple Things

La plateforme doit pouvoir communiquer plusieurs Things. Les consommateurs doivent pouvoir interagir avec chaque Thing via celle-ci de manière simple et uniforme.

3.2 Protocole abstraction

La plateforme doit se connecter avec chaque Thing via le protocole de communication spécifique de la Thing. Cependant, les consommateur de la plateforme peuvent y accéder avec un seul protocole de communication. Les consommateur n'ont pas besoin d'informations sur le protocole de communication de la Thing pour pouvoir interagir avec celle-ci. La figure 3 prends un exemple de deux Things communiquant en MQTT et en Modbus. L'application peut communiquer avec ceux-ci à travers la plateforme grâce au HTTPS.

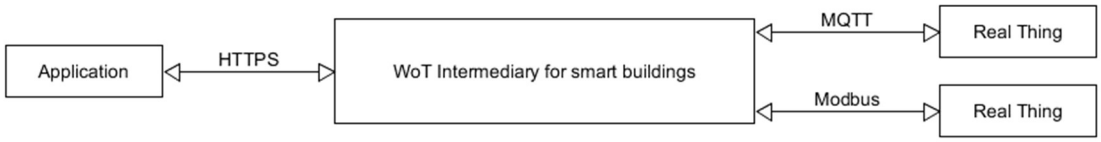


Figure 3: Exemple d'abstraction de protocole

3.3 Security management

Tout comme avec les protocoles de communication, chaque Thing doit être connectée avec son propre modèle de sécurité. La plateforme fait correspondre le modèle de sécurité pour chacun des Things sans affecter le consommateur de la plateforme. Tous les consommateurs utilisent tous le même modèle de sécurité.

3.4 Privacy control

Les utilisateurs auront des droits différents. Si un utilisateur n'a pas accès à une certaine Thing, celle-ci ne sera pas visible, pour lui, via la plateforme. Les interactions avec une Thing ne pourront être effectuées seulement par les utilisateurs y ayant accès.

3.5 Standard Object

La technologie IoT est aujourd'hui assez développée pour avoir plusieurs objets connectés différents ayant la même fonction. La plateforme doit avoir une fonctionnalité pour standardiser la valeur retournée par une Thing. Cela peut être extirper une valeur d'un groupe de valeur et/ou modifier la forme de cette valeur.

4 Design

Pour remplir les objectifs demandés, on peut en déduire une liste d'actions que les clients doivent pouvoir effectuer sur la plateforme. Ce chapitre contient une description théorique des requêtes nécessaires à la plateforme.

4.1 Récupérer la Thing Description de la plateforme

Une application utilisant la librairie node-wot doit pouvoir voir la plateforme comme une Thing. Il est donc nécessaire d'avoir une requête pouvant récupérer la TD de la plateforme.

4.2 Ajouter une Thing

Si une Thing est ajoutée dans le bâtiment, un administrateur doit notifier cet ajout. La Thing Description à ajouter doit être envoyée à la plateforme. La nouvelle Thing sera alors accessible par les applications via la plateforme.

4.3 Supprimer une Thing

De même, si une Thing est retirée du bâtiment, la plateforme doit être notifiée de ce changement. Un administrateur doit être capable de supprimer la connexion entre une Thing et la plateforme.

4.4 Lister les Things connues de la plateforme

La Thing Description de la plateforme ne contient aucune information sur les Things présentes dans le bâtiment. Pourtant les applications ont besoin de ces informations pour y accéder. La plateforme doit alors lister les IDs des Things connus connectés dans le bâtiment.

4.5 Récupérer la Thing Description d'une Thing

Comme pour le point 4.4 (4.4 Lister les Things connues de la plateforme), une requête doit pouvoir être faite pour récupérer la Thing Description d'une Thing. Cette Thing Description est une modification de celle passée à l'ajout. Les interactions y décrites passent par la plateforme intermédiaire pour communiquer avec la Thing. Une application verra la Thing comme un objet indépendant de l'intermédiaire même si les requêtes à celle-ci passent par la plateforme.

4.6 Interagir avec une Thing

Les 4 interactions possibles effectuées sur une instance d'une Thing passent par l'intermédiaire. L'intermédiaire envoie la requête à la Thing réelle demandée et retourne le résultat à l'application.

4.7 Standardiser une interaction

Le message retourné par une Thing réelle doit pouvoir être modifié avant d'être retourné à l'application. Par exemple, la lecture du statuts des températures sur un capteur retourne ceci :

```
{
   "measured": 47.18,
   "compensation": 25.80,
   "compensated": 21.38
}
```

L'intermédiaire doit pouvoir récupérer la valeur « 21.38 » et transformer ces unités pour la retourner à l'application.

4.8 Modifier l'accès à une interaction

Chaque interaction contient une liste de droits. Si une application n'a pas le droit d'accéder à une ressource, la plateforme doit bloquer cet accès. Un administrateur doit pouvoir modifier la liste des droits pour chacune des interactions.

5 Implémentation

Avec la listes des requêtes précédentes, il est possible de créer la plateforme. Ce chapitre décrit les caractéristiques du travail réalisé. Il explique sa structure, ses différents composants ainsi que chaque requête possible.

5.1 Plateforme Intermediary WoT

La plateforme est écrite en Javascript avec Node.js. Elle est utilisée à l'aide de la librairie node-wot qui donne les bases pour interagir avec des Things réelles. Elle peut être installée avec le gestionnaire de paquet de Node.js « npm ». Avec cet outil, il est possible de déployer l'application, de tester les fonctionnalités de la plateforme ou de faire démarrer un programme d'exemple utilisant celle-ci.

5.2 Diagramme de composants

Le protocole de communication entre les clients et la plateforme Intermediary choisi est le HTTPS puisque c'est le protocole le plus simple et commun qui supporte toutes les interactions. Le modèle de sécurité OAuth2.0 permet une gestion des droits et donc de l'accès aux ressources. Il nécessite d'être effectué sur une couche TLS. C'est pour cela que le protocole utilisé est le HTTPS et non le HTTP.

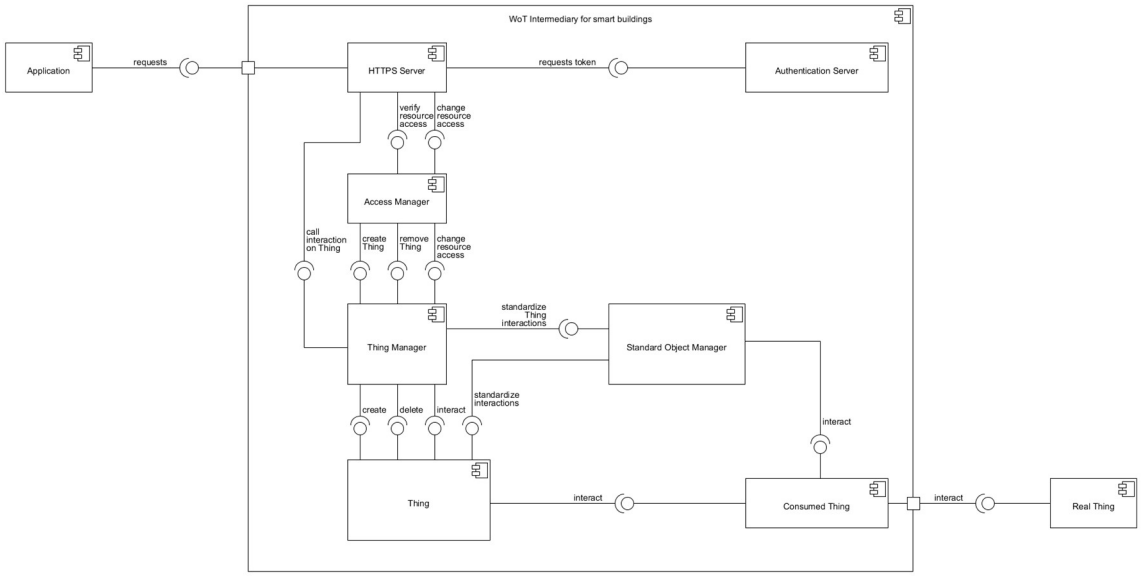


Figure 4: UML Diagramme de composants

.

La figure 4 présente les relations entre les différents composants de la plateforme. Pour pouvoir accéder aux ressources des Things, les consommateurs doit se faire autoriser à la plateforme par l'Authentication Manager avec le protocole OAuth2.0.
Le composant Access Manager autorise ou refuse l'accès aux ressources en fonction des droits de l'utilisateur.
La communication avec les Things réelles est traitée par des instances « Consumed Thing » de la librairie node-wot.
Le bloc Thing Manager permet de créer, supprimer et accéder aux Things demandées. Les valeurs de retour des requêtes sur les Things pourront être modifiée par le Standard Object Manager si besoin.

Figure 5: UML Diagramme de classe du Thing Manager

Le Thing Manager contient une liste des Things ajoutées, un schéma de validation JSON pour vérifier la qualité des TDs à l'ajout d'une Thing et une instance Servient qui permets de créer les instances node-wot des Things. Il contient aussi toutes les fonctions nécessaire pour manipuler les Things ainsi que pour les récupérer.

5.3 Serveur HTTPS avec API Rest

Les clients peuvent communiquer avec la plateforme grâce à une API RESTful via HTTPS. Les fonctions de la plateforme sont les mêmes que celles expliquées dans le chapitre 4. (4. Design) De plus, il existe une URI pour récupérer un token OAuth2.0 pour les futures requêtes (5.4 Modèle de sécurité OAuth2.0). Une définition plus détaillée des requêtes possibles est disponible au sous-chapitre 5.8 (5.8 Liste des requêtes).
La classe HttpsServer crée le serveur et gère les requêtes HTTPS de celui-ci. Elle utilise le module HTTPS pour créer un serveur qui nous permets de communiquer avec les clients.

5.4 Modèle de sécurité OAuth2.0

OAuth2.0 est un protocole d'industrie standard pour l'autorisation[11]. Notre serveur d'authentification utilisera le flot d'autorisation « client credentials ». Une application donne son ID de client et son mot secret dans les headers de la requête au serveur d'authentification. Celui-ci renvoie au client un token d'accès permettant d'accéder aux ressources qui lui sont permises (cf. Figure 6).

[11] OAuth, OAuth 2.0 [en ligne], 2021, https://oauth.net/2/ (consulté en août 2021)

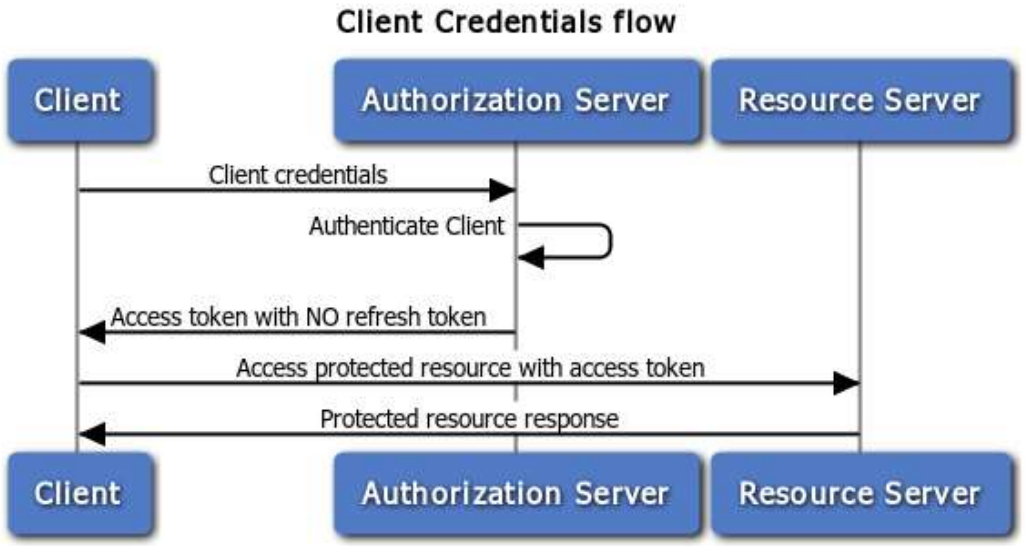


Figure 6: Flow client credentials du protocole OAuth2.0
Source: Client Credentials Grant Flow, API Gateway OAuth 2.0 Authentication Flows, Oracle[12]

Les scopes sont un mécanisme du OAuth 2.0 pour limiter l'accès d'une application aux ressources. A chaque ressource appartient une liste de scopes et à chaque utilisateur différent appartient une liste de scopes différentes. Si au moins une des valeurs de la liste de l'interaction voulue et une des valeurs de la liste de l'utilisateur correspondent, l'utilisateur a l'accès à cette ressource. Si ce n'est pas le cas, le serveur refuse l'accès à cette ressource. La liste des scopes de l'utilisateur est encodée dans le token reçu. Le token contient aussi l'ID de l'utilisateur et ses dates de création et d'expiration. Le token est un JSON Web Token (5.4.1 JSON Web Token (JWT)).

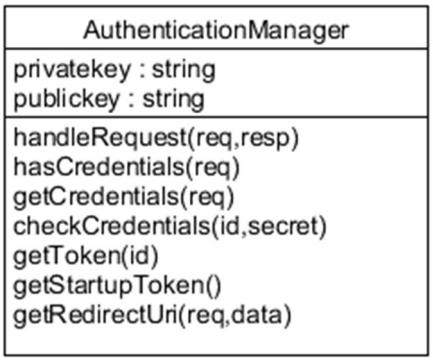


Figure 7: UML Diagramme de classe de l'Authentication Manager

L'Authentication Manager contient une clé privée et publique pour encoder et signer le token en RSA SHA256 grâce à la librairie « jsonwebtoken ». Les requêtes sur l'URI « /oauth/token » sont gérée dans la méthode handleRequest. La classe AuthenticationManager contient toutes les fonctions pour vérifier les identifiants de la requête et pour créer ou récupérer le token nécessaire.

[12] Oracle, API Gateway OAuth 2.0 Authentication Flows [en ligne], 2021, https://docs.oracle.com/cd/E50612_01/doc.11122/oauth_guide/content/oauth_flows.html (consulté en août 2021)

5.4.1 JSON Web Token (JWT)

JSON Web Token (JWT) est un moyen compacte et URL-safe pour représenter des jetons à échanger entre plusieurs parties. Il a la particularité d'être d'une part auto-explicatif et d'autre part d'être protégé contre toute tentative de modification grâce à une signature digitale. Le jeton est composé de 3 parties séparée par des points :

- Le header qui décrit la méthode de signature utilisée.
- Le contenu qui contient les données telles que les scopes, les dates de créations et d'expiration.
- La signature créée à partir du header et du contenu avec une clé privée.

Les deux premières parties sont encodée en base 64. Puis, la signature est un hash calculé à partir des deux premières parties. Les trois parties séparées par des points et concaténées forment le jeton complet. Le jeton distribué peut ensuite être décodé et vérifié grâce à une clé publique[13].

5.5 Accès limité pour utilisateurs

Chaque utilisateur aura un accès limité aux ressources de la plateforme. Un administrateur ayant tous les droits pourra gérer les droits des autres utilisateurs sur chaque interaction précise. La classe AccessManager gère l'accès aux ressources des Things en fonction de l'utilisateur. Ce contrôle d'accès est effectué grâce au token renvoyé par le server d'authentification OAuth2.0.

Figure 8: UML Diagramme de classe de l'Access Manager

L'objet accessTree est une arborescence des Things et de leurs interactions. Sous chaque interaction est listée les scopes ayant droit à celle-ci. La clé public sert à vérifier le token reçu de l'utilisateur. La classe AccessManager contient toutes les méthodes nécessaire à la vérification de token, à la récupération des droits d'accès et au changement de ceux-ci.

[13] Internet Engineering Task Force, JSON Web Token (JWT) [en ligne], RFC 7519, 2015, https://datatracker.ietf.org/doc/html/rfc7519 (consulté en août 2021)

5.6 Things

Lorsqu'un utilisateur ajoute une nouvelle TD, elle est retranscrite et ajoutée à la liste des Things de la plateforme. La nouvelle Thing Description utilisant la plateforme sera distribuée par l'Intermediary. Une instance ConsumedThing de la librairie node-wot est créée. C'est avec cette instance que les prochaines interactions avec la Thing réelle seront effectuées (cf. Figure 9).

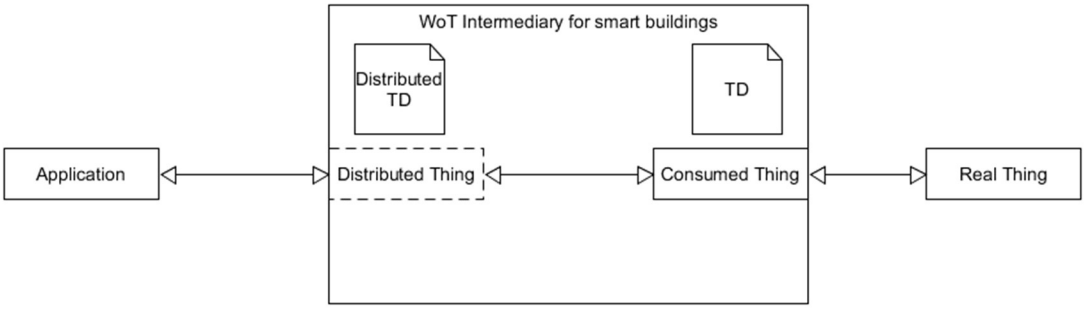


Figure 9: UML Chemin d'interaction

La plateforme doit pouvoir interagir avec plusieurs Things, la classe ThingManager s'occupe de créer, supprimer et accéder aux Things demandées.

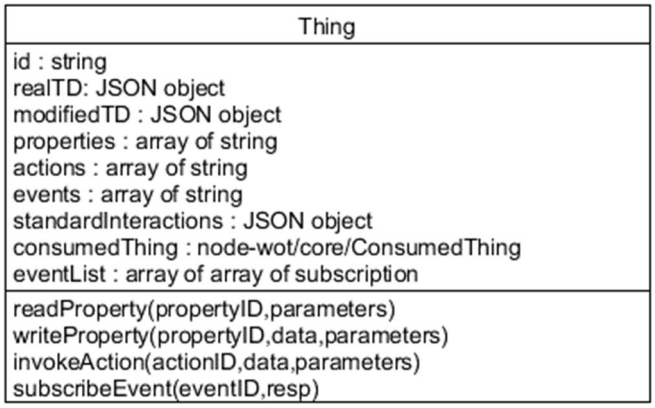


Figure 10: UML Diagramme de classe d'une Thing

Chaque Thing locale a un ID unique, sa TD réelle et la TD modifiée distribuée aux utilisateurs de la plateforme. Elle contient aussi des listes pour les propriétés, actions et évènements. Les interactions standardisées sont enregistrées dans l'objet « standardInteractions ». Les méthodes appellent directement l'instance Consumed Thing sauf s'il s'agit d'un interaction standardisée. Dans ce cas, le Standard Object Manager s'occupe de la requête.

5.7 Standard Object

Plusieurs Things remplissant les mêmes fonctions mais avec des versions ou producteurs différents peuvent être accéder de façon standard.
Par exemple, la valeur de la température d'un capteur pourra être lue via la ressource « temp » qui retournera le message JSON suivant :

```
{
  "measured": 47.18,
  "compensation": 25.80,
  "compensated": 21.38
}
```

L'accès à la température via la ressource « temperature » doit retourner « 21.38 ».
Pour effectuer cela, il faudra ajouter une clé « standardObject » avec comme valeur le nom de la nouvelle ressource sous « compensated » de la propriété « temp » de la Thing Description avant de faire une requête pour l'ajouter. La figure 11 montre la nouvelle propriété « temp » standardisée.

```
"temp": {
  "type":"object",
  "description":"Displays the temperature datas",
  "readOnly":true,
  "properties": {
    "measured": {
      "type":"number",
      "unit":"°C",
      "readOnly":true
    },
    "compensation": {
      "type":"number",
      "unit":"°C",
      "readOnly":true
    },
    "compensated": {
      "type":"number",
      "unit":"°C",
      "readOnly":true,
      "standardObject":"temperature",
      "operation": "x+273.15"
    }
  }
}
```

Figure 11: Exemple d'une interaction standardisée

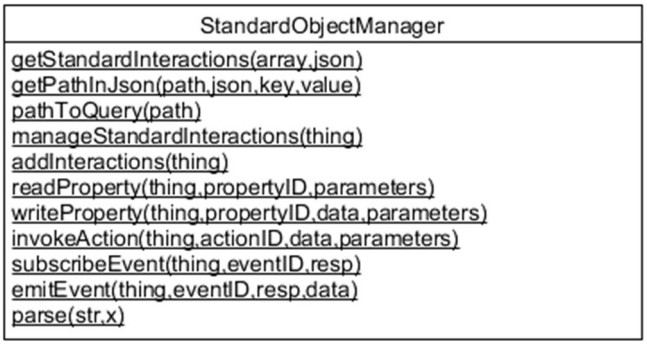


Figure 12: UML Diagramme de classe du Standard Object Manager

Le Standard Object Manager contient des méthodes pour récupérer l'interaction d'origine de l'interaction standardisée et l'opération à effectuer sur la valeur retournée. Ces méthodes servent aussi à créer cette nouvelle interaction. Lors d'une requête sur une interaction standardisée, il utilise directement l'instance Consumed Thing de la Thing passée en paramètre pour accéder à l'interaction d'origine. La librairie « jmespath » va chercher la valeur voulue dans le payload. La méthode parse est responsable de l'opération. Elle est utile pour des raisons de sécurité.

5.8 Liste des requêtes

Sous ce point sont expliquées chacune des requêtes que la plateforme peut effectuer. Les détails techniques (Headers, body requis, payload, …) de ces requêtes peuvent être retrouvés soit dans le README.md à la racine de la plateforme soit avec la spécification Swagger de l'API de la plateforme.

5.8.1 Spécification Swagger de l'API

Une simple requête GET à l'URI « /swagger » permet de récupérer la spécification Swagger de l'API de la plateforme sous forme d'un fichier JSON. La spécification peut

être visualisée sous forme graphique à l'URL https://petstore.swagger.io/ en entrant l'URI https://<host>/swagger dans le champ du sommet et en cliquant sur « Explore ».

5.8.2 Récupérer la Thing Description de la plateforme

La TD de la plateforme peut être récupérée à l'URI « / ». Elle ne change jamais. Certaines interactions ne peuvent être effectuées seulement par des administrateurs.

5.8.3 Récupérer un token d'accès

Les prochaines requêtes ne peuvent être effectuées seulement avec un token d'accès de la plateforme. Un utilisateur peut demander un token à la plateforme à l'URI « /oauth/token » à l'aide de ses identifiants. La requête est passée à l'Authentication Manager qui répondra avec un token en fonction des identifiants (cf. Figure 13). Le token reçu dure 1 heure après laquelle l'utilisateur devra redemander un nouveau token.

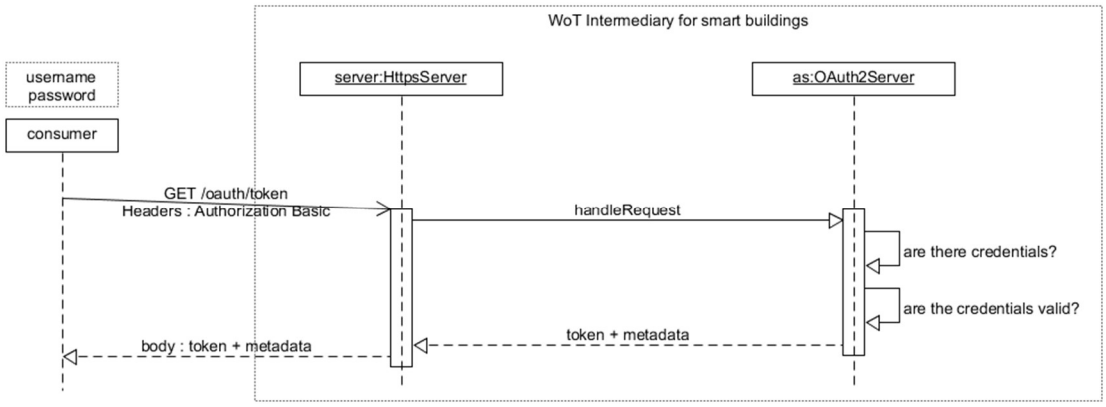


Figure 13: UML Diagramme de séquence, récupération du token

Le body de la réponse est un fichier JSON contenant le token, son type et le nombre de secondes restantes durant lesquels le token est encore valide.

5.8.4 Ajouter une Thing

Un administrateur peut ajouter une Thing en envoyant sa Thing Description à la plateforme. L'Access Manager vérifie que l'utilisateur a bien les droits d'administrateurs puis le Thing Manager crée une nouvelle instance de Consumed Thing et une copie de la TD modifiée utilisant la plateforme (cf. Figure 14).

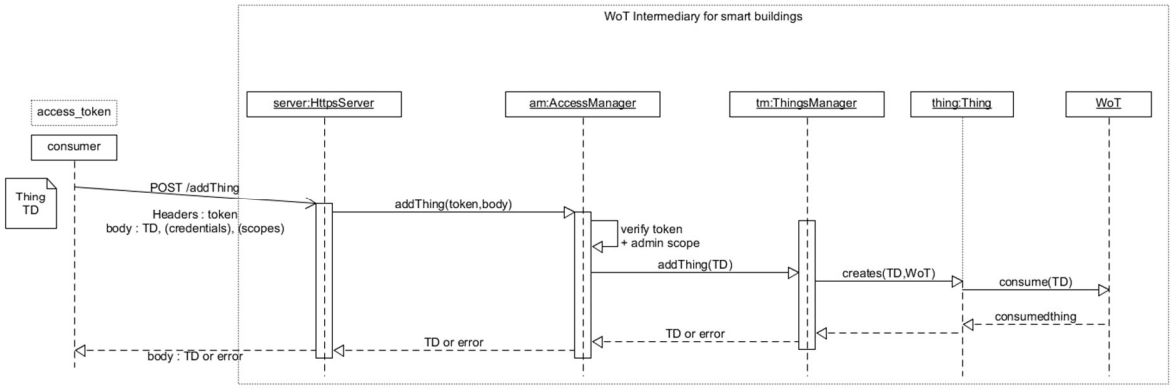


Figure 14: UML Diagramme de séquence, ajouter une Thing

La requête d'ajout valable d'une Thing est enregistrée sur le disque dur local de la plateforme. Cela permet au serveur de se souvenir des Things présentes à l'arrêt du serveur lorsqu'il est redémarré.

5.8.5 Standardiser une interaction

En reprenant l'exemple du point 4.7 (4.7 Standardiser une interaction), la propriété « temp » de la Thing en question est présentée en figure 15.

```
"temp": {
  "type": "object",
  "description" : "Displays the temperature datas",
  "properties": {
    "measured" : {
      "type": "number",
      "unit": "°C",
      "readOnly": true
    },
    "compensation" : {
      "type": "number",
      "unit": "°C",
      "readOnly": true
    },
    "compensated" : {
      "type": "number",
      "unit": "°C",
      "readOnly": true
    }
  }
}
```

Figure 15: Description de la property "temp"

Le client peut créer une nouvelle propriété nommée « temperature » qui retourne le contenu « compensated » du payload report. Cela peut être fait en ajoutant une clé standardObject sous la propriété voulue du payload dans la Thing Description de la requête d'ajout. Cette manipulation peut être faite sur toutes les interactions. Si le type du payload de l'interaction voulue n'est pas un objet, la plateforme créera seulement une copie de l'interaction avec le nom donné. Si une manipulation sur la valeur retournée doit être effectuée, l'utilisateur peut ajouter une chaîne de caractères sous la clé « operation » qui représentera l'opération que la plateforme doit effectuer sur la valeur. Pour cet exemple, si les unités doivent être converties des degré Celsius en Kelvin, la chaîne de caractères sera « x+273.15 ». La figure 16 affiche la nouvelle propriété standardisée.

```
"temp": {
  "type":"object",
  "description":"Displays the temperature datas",
  "readOnly":true,
  "properties": {
    "measured": {
      "type":"number",
      "unit":"°C",
      "readOnly":true
    },
    "compensation": {
      "type":"number",
      "unit":"°C",
      "readOnly":true
    },
    "compensated": {
      "type":"number",
      "unit":"°C",
      "readOnly":true,
      "standardObject":"temperature",
      "operation": "x+273.15"
    }
  }
}
```

Figure 16: Description standardisée de la property "temp"

Le Standard Object Manager s'occupe au moment de l'ajout de la Thing de créer cette nouvelle interaction dans la TD modifiée. Il s'occupe aussi, lors de chaque accès à une interaction standardisée, de chercher l'interaction voulue existante et de transformer le payload reçu.

5.8.6 Supprimer une Thing

Un administrateur peut aussi supprimer une Thing en spécifiant son ID avec une requête DELETE à l'URI « /removeThing/{thingID} ». L'Access Manager vérifie que l'utilisateur a bien les droits d'administrateurs puis le Thing Manager supprime toute relation avec la Thing demandée (cf. Figure 17). La requête d'ajout stockée localement liée à cette Thing est aussi supprimée.

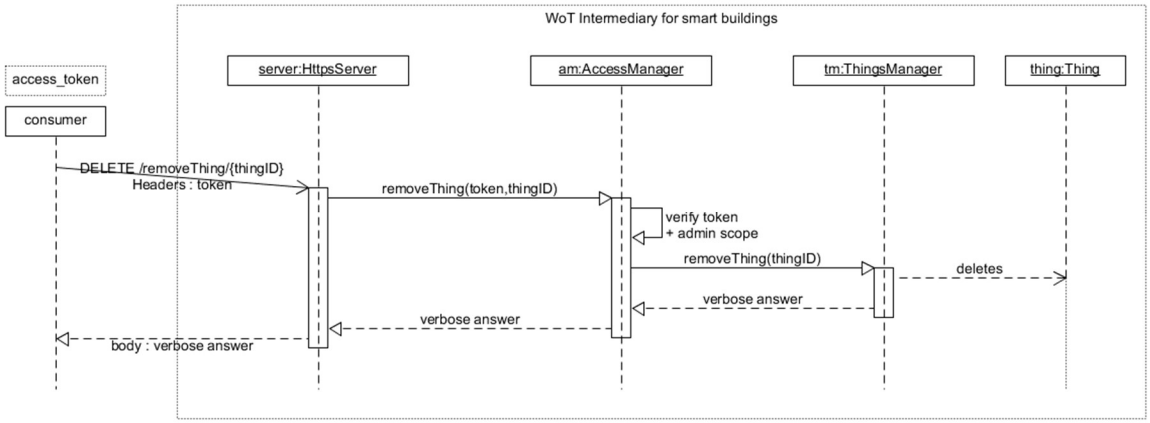


Figure 17: UML Diagramme de séquence, supprimer une Thing

5.8.7 Lister les Things ajoutées

Une liste des IDs des Things connues de la plateforme peuvent être trouvée à l'URI « /things ». L'Access Manager vérifie que l'utilisateur a accès à au moins une interaction d'une Thing pour l'ajouter dans la liste. Si ce n'est pas le cas, l'utilisateur ne voit pas la Thing dans la liste.

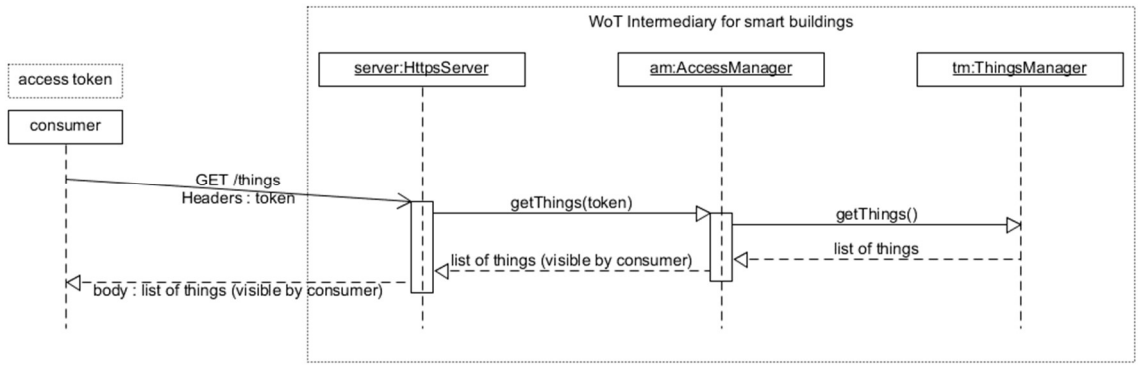


Figure 18: UML Diagramme de séquence, lister les Things

5.8.8 Récupérer la Thing Description d'une Thing

La Thing Description modifiée utilisant la plateforme de chaque Thing peut être récupérée à l'URI « /things/{thingID} ». Le Thing Manager trouve l'objet Thing voulu si existante et y récupère la TD modifiée. L'Access Manager modifie ensuite cette Thing Description pour cacher les interactions dont l'utilisateur n'a pas accès (cf. Figure 19).

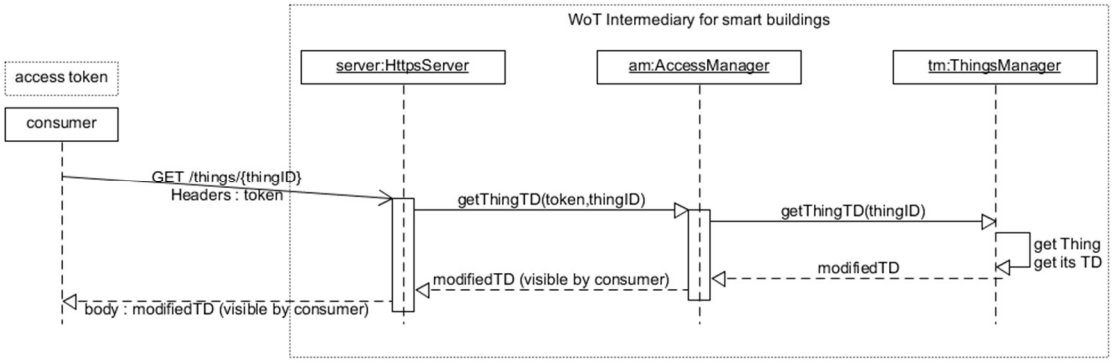


Figure 19: UML Diagramme de séquence, récupérer la TD d'une Thing

5.8.9 Lister les properties/actions/events d'une Thing

Une liste de chaque type d'interaction peut être affichée aux URI :

- « /things/{thingID}/properties »
- « /things/{thingID}/actions »
- « /things/{thingID}/events »

L'Access Manager récupère la liste complète des interactions avec cette Thing et cache les interactions dont l'utilisateur n'a pas accès (cf. Figure 20).

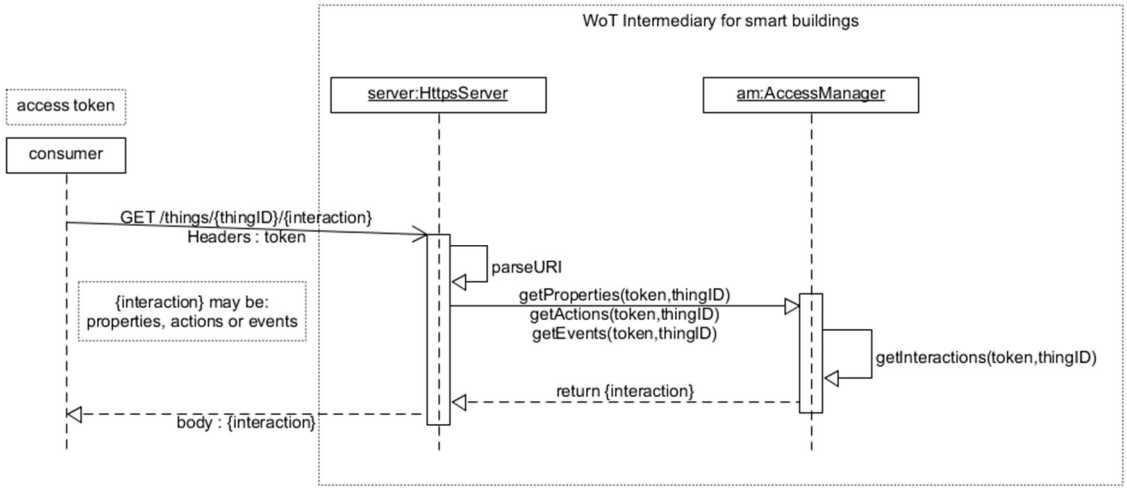


Figure 20: UML Diagramme de séquence, Lister un type d'interaction

5.8.10 Interagir avec une Thing

Les 4 interactions possibles effectuées sur une instance d'une Thing passent par l'intermédiaire. Le Thing Manager vérifie que la Thing et l'interaction existent. L'Access Manager vérifie que l'utilisateur y ait accès puis le Thing Manager trouve l'objet Thing voulu et ordonne l'interaction. Si elle est standardisée, la Thing la passe au Standard Object Manager. Les figures 21, 22 et 23 présente l'accès à une interaction.

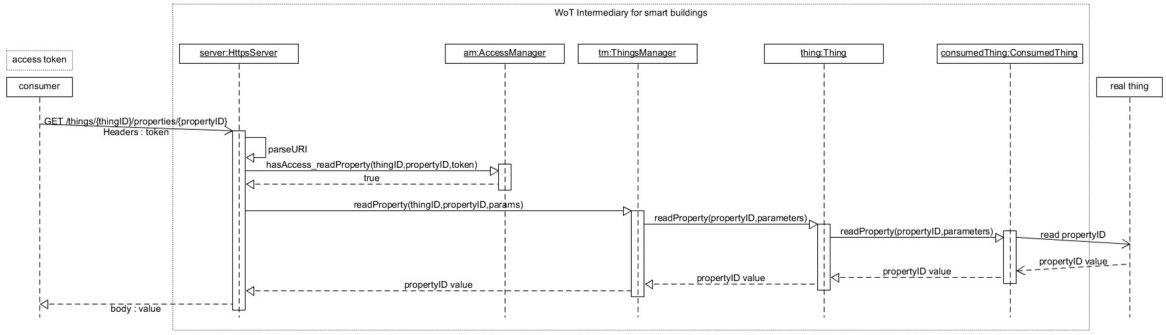


Figure 21: UML Diagramme de séquence, lecture de propriété

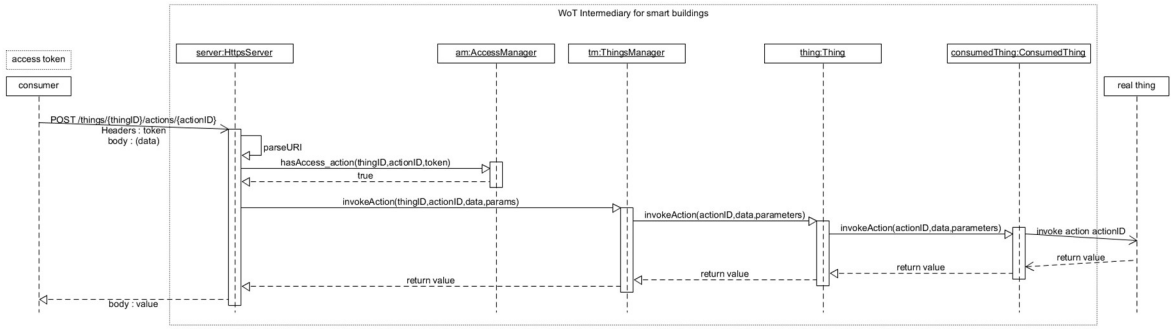


Figure 22: UML Diagramme de séquence, invoquer une action

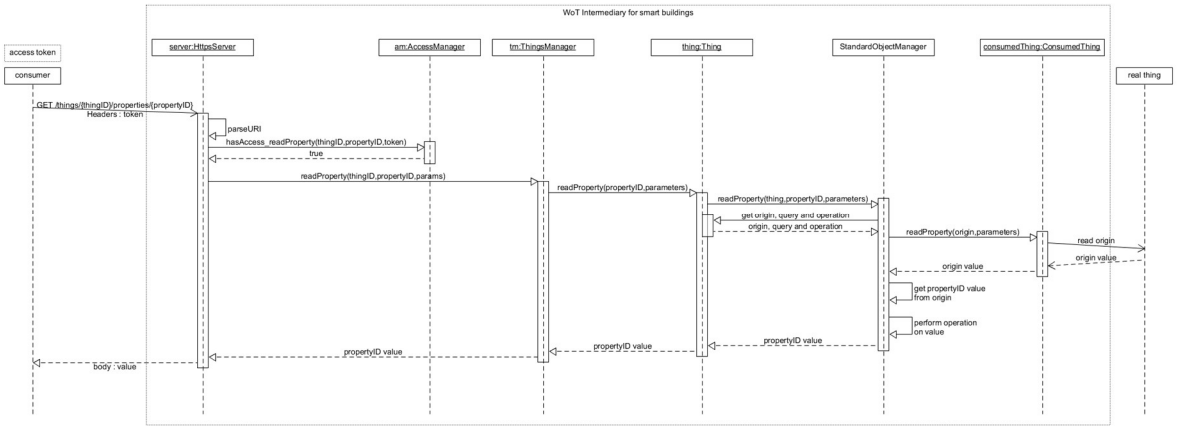


Figure 23: UML Diagramme de séquence, lecture de propriété standardisée

5.8.11 Recevoir les évènements

Les évènements des Things peuvent être reçu grâce à la technique du long poll. Si aucun évènement n'a été reçu dans les 20 secondes après la requêtes, la requête est retournée avec une erreur.

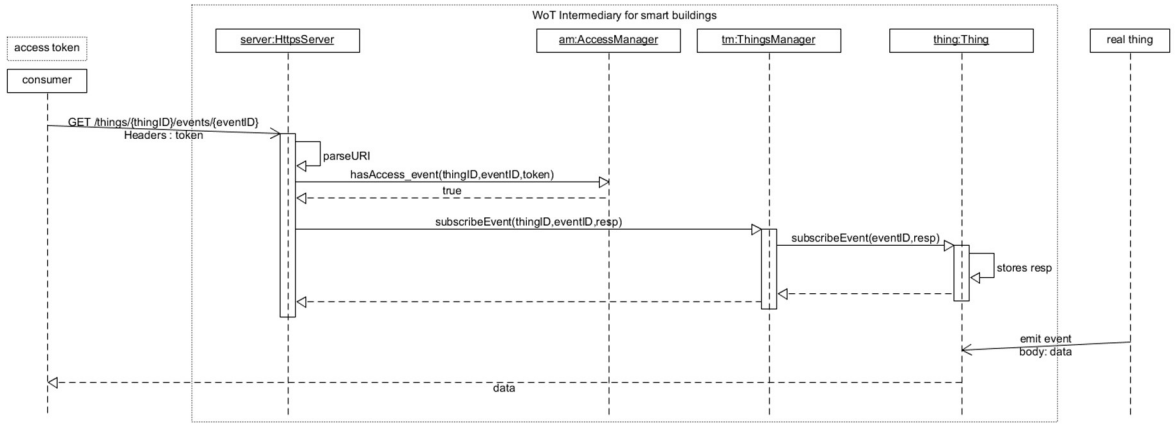


Figure 24: UML Diagramme de séquence, recevoir les évènements

5.8.12 Modifier l'accès à une interaction

Un administrateur peut modifier les scopes associés à chaque interactions. L'Access Manager vérifie que la requête vient d'un administrateur puis modifie les scopes de l'interaction. Le Thing Manager modifie de même les scopes dans le TD modifié de la Thing. Il met aussi à jour la requête d'ajout stockée localement.

6 Déploiement

Plusieurs problèmes ont été rencontrés pour le déploiement du travail développé sur de nouvelles machines. Pour éviter ces problèmes, ce chapitre décrit comment installer, démarrer et utiliser la plateforme développée. Les détails techniques pour cette parties sont contenus dans les fichiers README.md, INSTALLATION.md et HOWTOUSE.md à la racine de la plateforme.

6.1 Composants prérequis

Pour installer la plateforme, il faudra avoir des composants spécifiques installés sur la machine.

6.1.1 Node.js

Le programme étant écrit en Javascript à l'aide de Node.js. Il faudra avoir une instance de Node avec sa version 14 ou plus élevée.

6.1.2 node-gyp

node-gyp est un outil de compilation additionnel à Node.js. Cet outil est requis pour utiliser la librairie node-wot et donc pour la plateforme développée. Il est expliqué dans le fichier INSTALLATION.md comment installer les composants nécessaire pour node-gyp.

6.1.3 Certificat racine

Le serveur HTTPS de la plateforme utilise un certificat issu d'un certificat racine nommé rootCA.crt. Il peut être trouvé dans le dossier « certificate » à la racine de la plateforme. Pour que la connexion à la plateforme soit sécurisée, il est nécessaire d'installer le certificat racine dans les « Trusted Root Certificate Authorities » de la machine.

6.1.4 Version du navigateur

Pour utiliser l'application de démonstration du chapitre 7.2 (7.2 Environnement démonstration de la plateforme), il est nécessaire d'avoir des versions de navigateurs supportant ECMAScript 2015. Comme :

- Microsoft Edge 15 et suivants
- Firefox 54 et suivants
- Chrome 58 et suivants
- Safari 10 et suivants

6.2 Installation des modules

Dès que toutes les étapes précédentes sont validées, les modules utilisés par la plateforme peuvent être installés. Pour cela, il suffit d'entrer la ligne suivante dans l'invité de commande à la racine de la plateforme:

- npm install.

Dès l'installation finie, il devrait être possible de démarrer le serveur comme indiqué dans le README.md.
Un exemple d'utilisation de la plateforme intermédiaire pour être trouvé dans HOWTOUSE.md.

7 Validation

Pour vérifier le bon fonctionnement de la plateforme, deux scripts de tests ont été effectués. L'un utilisant la librairie node-wot et l'autre en utilisant des requêtes HTTPS pures. De plus, un site web en HTML + CSS a été designé pour présenter le projet de manière graphique.

7.1 Plateforme de test pour bâtiment avec IoT

Les tests utilisent des Things qui doivent fonctionner en local sur la machine. Les deux scripts comportent les mêmes tests. Ils vérifient chaque fonctionnalité de la plateforme en faisant des requêtes valides ou avec erreurs. Chaque réponse de la plateforme est comparée avec sa réponse attendue. Un bilan du total de réponses correctes et erronées sont affichées à la fin du test. La figure 25 montre que les tests ne retournent pas d'erreurs.

Figure 25: Screenshot des tests utilisant directement l'API REST

7.2 Environnement démonstration de la plateforme

Cet environnement de démonstration représente une maison à 2 étages comportant différentes Things. Le bâtiment et les Things sont réels. Le site web ajoute ces Things à la plateforme Intermediary et essaie de lire les valeurs utiles des différentes Things. Ces valeurs sont ensuite affichées (cf. Figure 26 et 27). Les Things étant réelles, il est possible qu'une Thing ne réponde pas tout de suite ou renvoie une erreur. Les valeurs affichées contiennent les dernières réponses retournées sans erreurs.

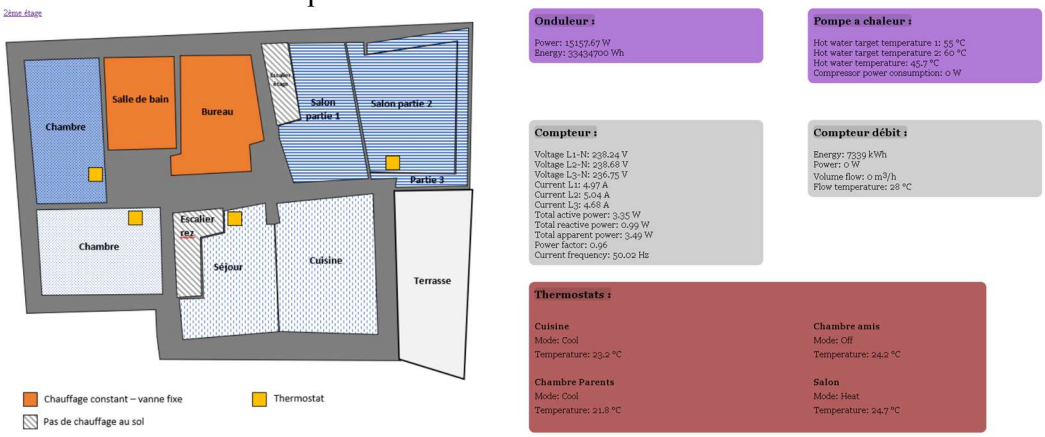


Figure 26: Screenshot site web 1er étage

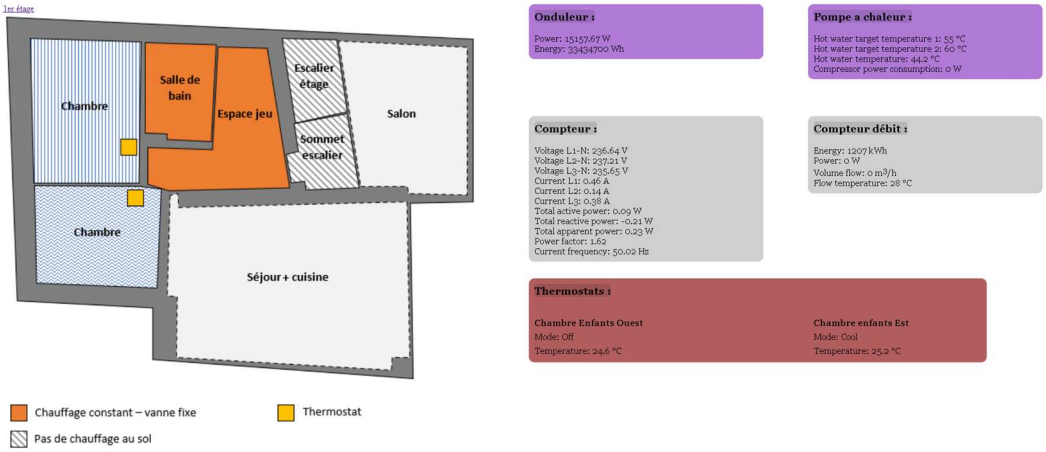


Figure 27: Screenshot site web 2ème étage

7.3 Test de scalabilité

Un programme de test a été écrit pour vérifier le temps de réponse des requêtes dans une situation de stress. Le programme envoie un certain nombre de requêtes (de 1 à 1000 requêtes) en même temps et mesure le temps totale de réponse. Les résultats sont inscrits dans un fichier CSV. Voici un graphe des résultats reçus :

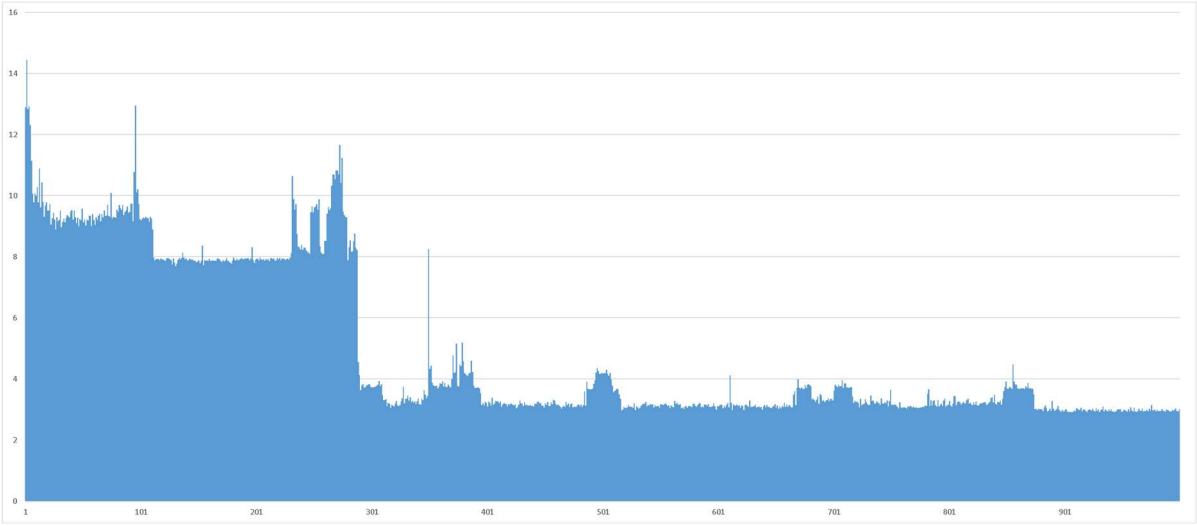


Figure 28: Temps de réponse [ms] en fonction du nombre de requêtes simultanées

Le graphe de la figure 28 démontre que le temps de réponse n'augmente pas en fonction du nombre de requêtes envoyées en même temps. Ce test prend la moyenne du temps de réponse sur 10 fois pour chaque x nombre de requête de 1 à 1000 requêtes simultanées. Le test a été effectué avec une Thing hébergée sur le PC en local, l'Intermediary en local et le programme de test en local. Le temps de réponse moyen tombe de 8[ms] à 3[ms] vers les 300 requêtes simultanées. Le test ayant duré plusieurs heures, il est possible que la charge de travail sur la machine local ait fluctué. Cela peut expliquer cette chute de temps de réponse.

8 Travail supplémentaire

8.1 Problèmes de la librairie node-wot

La librairie node-wot est actuellement encore en développement. Cela veut dire qu'elle n'est pas parfaite et peut contenir des défauts. Ce projet utilise une grande partie des fonctionnalités de la librairie. C'est en testant ces différentes fonctionnalités que des erreurs/comportements inattendus ont été remarqués. J'ai donc notifié les développeurs de la librairie via le GitHub des quelques erreurs trouvées et de moyens de résoudre ces erreurs. Celles-ci ont été corrigées rapidement par les développeurs.

8.2 Protocole M-Bus pour la librairie node-wot

Une des Things de l'environnement de démonstration de la plateforme (7.2 Environnement démonstration de la plateforme) communique avec le protocole de communication M-Bus over TCP. La librairie ne comportant pas de moyen de communiquer en M-Bus, j'ai développé une partie de la librairie permettant de communiquer en M-Bus over TCP. Cette partie utilise la librairie node-mbus[14] pour établir un client M-Bus. Cette modification a été fournie aux développeurs de la librairie node-wot via le GitHub. Après quelques modifications, l'intégration de cette partie reste encore en traitement au moment de l'écriture de ce document.

[14] Apollon77, GitHub repo node-mbus [en ligne], 2021, https://github.com/Apollon77/node-mbus) (consulté en août 2021)

9 Conclusion

En reprenant l'exemple de l'introduction, dans un bâtiment, il y a plusieurs systèmes IoT et Things. Chacune des Things possèdent un protocole de communication et des formes de données propres à elles-mêmes. Avec cette plateforme, une application peut simplement se connecter à n'importe quelle Thing en utilisant 1 seul protocole de communication avec 1 seul modèle de sécurité. Les données envoyées à/reçues de la plateforme sont uniformes. La plateforme Intermediary permet l'interopérabilité entre les systèmes IoT. L'application peut utiliser la librairie node-wot ou directement l'API RESTful pour interagir avec les Things.
Le projet a été mené correctement à bout et remplit les 5 fonctions présentés dans le chapitre 3 (3 Analyse) :

- **Multiple Things** : La plateforme supporte l'ajout de plusieurs Things
- **Protocole abstraction** : Les applications utilisent le HTTPS pour n'importe quelle Thing, indépendamment de son protocole de communication.
- **Security management** : Les applications utilisent l'OAuth2.0 pour n'importe quelle Thing, indépendamment de son modèle de sécurité.
- **Privacy control** : Les interactions avec les Things ont des droits dont les applications ont besoin d'avoir pour y accéder.
- **Standard object** : Les interactions et leur données peuvent être standardisée.

La plateforme peut être utilisée avec la librairie node-wot ou directement avec son API RESTful. Les tests ont validé les fonctionnalités de la librairie et leur bon fonctionnement.

9.1 Améliorations et orientations futures

Ce projet, comme tout projet, est susceptible à de futures améliorations ou développements. Tout d'abord, comme la librairie node-wot est en développement, ce projet pourra et devra possiblement être mis à jour avec les nouveaux développements de la librairie.
Une autre amélioration pourrait être de faire le management d'accès via une base de données. Pour le moment, la liste des utilisateurs et de leurs scopes sont sauvées en local dans un fichier contenant un objet JSON. Pour pouvoir plus facilement modifier et accéder à cette liste d'utilisateur, il serait très utile de l'implémenter dans une base de données.
Pour le moment, la possibilité d'observer une propriété grâce à la librairie node-wot est encore peu clair. Une amélioration serait d'ajouter cette interaction à la librairie.

10 Glossaire

Body : Données contenus dans le corps de la requête.

Consommateur : Client communiquant avec une Thing.

ConsumedThing : Instance de la librairie node-wot qui permet de communiquer avec une Thing

Description OpenAPI : Aussi nommée « **Swagger** », Spécification pour des fichiers machine-readable de description de service Web RESTful.

Hash : Fonction qui calcule une empreinte numérique servant à identifier rapidement la donnée initiale.

Header : L'entête d'une requête, contient des données et métadonnées sur la requêtes

Interaction : Dans le WoT, les 3 types d'interactions sont les propriétés, les actions et les évènements.

Internet of Things : Réseau d'objets connectés, de Things

Interopérabilité : Capacité que possède un produit ou un système, dont les interfaces sont intégralement connues, à fonctionner avec d'autres produits ou systèmes existants ou futurs et ce sans restriction d'accès ou de mise en œuvre.

Long poll : Mode de communication client-serveur qui consiste en ce que, lorsque le client envoie une requête au serveur, celui-ci ne réponde pas immédiatement mais seulement lorsque la ressource voulue est disponible.

Métadonnée : Donnée servant à définir ou décrire une autre donnée.

OAuth2.0 : Protocole libre qui permet d'autoriser un site web, logiciel ou application à utiliser une API sécurisée d'un autre site web pour un utilisateur. (5.4 Modèle de sécurité OAuth2.0)

Payload : Body d'une requête, message transmis ou reçus.

Proxy : Composant réseau qui se place entre deux hôtes pour faciliter leurs échanges

REST : Style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web. Il permet aux systèmes effectuant des requêtes de manipuler des ressources web via leurs représentations textuelles à travers un ensemble d'opérations uniformes et prédéfinies sans état.

RESTful : Conforme au style d'architecture REST

Scalabilité : capacité à maintenir ses fonctionnalités et ses performances en cas de forte demande.

Scope : Mécanisme de OAuth2.0 qui limite l'accès d'une application au compte de l'utilisateur. (5.4 Modèle de sécurité OAuth2.0)

Servient : Instance de la librairie node-wot qui permet de créer et stocker des instances de ConsumedThing et leur identifiants.

Smart plug : Prise électrique intelligente connectée qui contient en général une fonction pour allumer/éteindre la prise et une fonction pour mesurer la puissance de la prise.

Thing : Un objet physique composé de capteurs, actionneur ou programmes pouvant échanger des données sur le réseau internet. (2.1.1 Thing)

Thing description : Modèle d'information ouvert et libre de droits avec un format de représentation basé sur JSON pour l'Internet des objets. (2.1.2 Thing Description)

Token (jeton) : Identificateur (chaîne de caractère) utilisé pour permettre l'accès à une ressource protégée. (5.4.1 JSON Web Token (JWT))

Web of Things : Un ensemble de standard du W3C pour résoudre les problèmes d'interopérabilité de différentes plateformes IoT et domaines d'application.

Websocket : Protocole réseau visant à créer des canaux de communication sur une connexion TCP.

11 Abréviations

API: Application Programming Interface

CSS: Cascading Style Sheets

CSV: Comma-Separated Values

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

HTTPS: HyperText Transfer Protocol Secure

ID: Identification

IoT: Internet of Things

JSON: JavaScript Object Notation

JWT: JSON Web Token

npm: Node Package Manager

REST: REpresentational State Transfer

RSA SHA256: Rivest–Shamir–Adleman Secure Hash Algorithm 256bits

TCP: Transmission Control Protocol

TD: Thing Description

TLS: Transport Layer Security

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

WoT: Web of Things

W3C: World Wide Web Consortium

12 Bibliographie

[1,3] **W3C, W3C Web of Things** [en ligne], 2021, https://www.w3.org/WoT/ (consulté en août 2021)

[2] **W3C, W3C WoT – Architecture Task Force** [en ligne], 2021, https://www.w3.org/WoT/activities/tf-architecture/ (consulté en août 2021)

[4] **W3C, Web of Things (WoT) Thing Description** [en ligne], 2021, https://www.w3.org/TR/wot-thing-description/ (consulté en août 2021)

[5,6] **W3C, Web of Things (WoT) Architecture 1.1, Intermediaries** [en ligne], 2021, https://w3c.github.io/wot-architecture/#intermediaries (consulté en août 2021)

[7] **W3C, W3C WoT – Working Group** [en ligne], 2021, https://www.w3.org/WoT/wg/ (consulté en août 2021)

[8] **Thingweb, Page d'acceuil** [en ligne], 2021, http://www.thingweb.io/ (consulté en août 2021)

[9] **Eclipse, GitHub repo thingweb.node-wot** [en ligne], 2021, https://github.com/eclipse/thingweb.node-wot (consulté en août 2021)

[10] **SCHWAB, Linus. Universal Explorer for the Web of Things** [en ligne], 2018, https://www.unifr.ch/inf/softeng/en/assets/public/files/research/students_projects/master/Master_Schwab_Linus.pdf (consulté en août 2021)

[11] **OAuth, OAuth 2.0** [en ligne], 2021, https://oauth.net/2/ (consulté en août 2021)

[12] **Oracle, API Gateway OAuth 2.0 Authentication Flows** [en ligne], 2021, https://docs.oracle.com/cd/E50612_01/doc.11122/oauth_guide/content/oauth_flows.html (consulté en août 2021)

[13] **Internet Engineering Task Force, JSON Web Token (JWT)** [en ligne], RFC 7519, 2015, https://datatracker.ietf.org/doc/html/rfc7519 (consulté en août 2021)

[14] **Apollon77, GitHub repo node-mbus** [en ligne], 2021, https://github.com/Apollon77/node-mbus) (consulté en août 2021)

13 Annexes

13.1 README.md

13.2 HOWTOUSE.md

13.3 INSTALLATION.md

13.5 Planning prévu

13.6 Planning réalisé