République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD DAHLEB - Blida 1
Faculté des Sciences
Département informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de master en informatique

Option : Ingénierie de logiciel

Thème

Conception et réalisation d'une application permettant la collaboration dans l'Internet des objets

Réalisé par :

Hasnaoui Abdelrahim **Kahil Mohamed Amine**

Devant le jury composé de :

Présidente : Mme. Aroussi Sana	**Maitre-assistant A/USDB**
Examinatrice : Mme. Mezzi Melyara	**Maitre de Conférence B/USDB**
Promotrice : Mme. Abed Hafida	**Professeur/USDB**

Année universitaire 2021/2022

Remerciements

Nous voudrons tout d'abord adresser toute notre reconnaissance à toutes les personnes qui ont contribué au succès de notre projet de fin d'étude et qui nous ont aidé lors de la rédaction de ce mémoire, à notre promotrice de ce mémoire, pour sa patience et surtout ses judicieux conseils, qui ont contribué à alimenter notre réflexion.

Nous exprimons aussi notre reconnaissance envers les amis et collègues qui nous ont apporté leur soutien moral et intellectuel tout au long de nos études, nos parents, pour leur soutien constant et leurs encouragements.

Dédicaces

Nous dédions ce modeste travail :

À nos chers parents, pour tous leurs sacrifices, leur soutien et leurs prières tout au long de nos études

À toutes nos familles pour leur soutien tout au long de notre parcours universitaire, que ce travail soit l'accomplissement et le fuit de vos soutiens infaillibles

Résumé

L'Internet des objets (ou IOT) consiste à connecter plusieurs objets entre eux en utilisant des protocoles de communications prédéfinis. Ces objets peuvent être soit des capteurs ou des actionneurs, qui permettent de mesurer l'environnement d'une manière optimale. Les objets connectés interagissent en ajoutant les notions du web dans l'internet des objets, ce que nous appelons le web des objets. Ce défi a posé plusieurs problèmes, notamment du fait de la collaboration entre les objets physiques, et de la nature dynamique et de l'hétérogénéité des données et des systèmes qui composent le web des objets (appareils puissants/peu puissants, fixes/mobiles, batteries/alimentations continues, etc.).

Ces caractéristiques nécessitent des outils et des méthodes compatibles pour la réalisation des applications capables d'extraire des informations utiles depuis de nombreuses sources de données disponibles et d'interagir aussi bien avec l'environnement, au moyen des actionneurs et d'interfaces dédiées pour une exploitation par l'utilisateur.

Le travail réalisé, objet du présent document, consiste à réaliser une application qui permet la collaboration dans l'internet des objets. L'objectif principal de ce travail est d'utiliser un web service comme un moyen de communication et d'échange de données entre les objets sans se soucier de la manière où les objets sont connectés au réseau. Afin de valider et de concrétiser notre solution, nous avons réalisé un prototype dans le domaine de la domotique. Pour cela nous avons assemblé des capteurs et actionneurs, nous avons implémenté les serveurs et enfin implémenté l'application client.

Mots clés : Internet des objet, Web service, Web des objets, Domotique.

الملخص

تهدف انترنت الأشياء الى ربط العديد من الأشياء ببعضها البعض عن طريق استخدام بروتوكولات خاصة بمجال الاتصالات. هذه الأشياء لها صنفين اما ان تكون مستقبلات او اما ان تكون مشغلات ميكانيكية (تقوم بتنفيذ الأوامر)، التي بدورها تجعل الوسط سلسا. تقوم الأشياء بالتواصل مع بعضها البعض عن طريق إضافة قواعد خاصة بالويب في انترنت الأشياء، و التي تسمى بويب الأشياء. هذا التحدي طرح العديد من المشاكل، على وجه الخصوص نذكر منها مشكلة التعاون و تبادل المعلومات بين الأشياء الملموسة، الطبيعة الديناميكية و مشكلة عدم تطابق البيانات و الأنظمة التي يتكون منها الويب(أجهزة قوية\ أجهزة ضعيفة، الهاتف الثابت\ المحمول، بطارية\سلك موصول، الخ).

هذه المواصفات تتطلب ادواة و وسائل ملائمة لانشاء تطبيقات قادرة على استخراج معلومات مفيدة من مصادر عديدة حسب البيانات المتاحة و لكي تتفاعل أيضا مع الوسط، بواسطة المشغلات الميكانيكية بالإضافة الى واجهات خاصة مخصصة للاستعمال من طرف المستخدم.

العمل المنجز, يقوم على انشاء تطبيق يسمح بتعاون ما بين الأشياء في انترنت الأشياء. الهدف الرئيسي من هذا العمل هو استخدام تكنولوجيا خدمات الويب كوسيلة اتصال و لتبادل البيانات بغض النظر عن كيفية توصيل هذه الأشياء بالشبكة. لتحقيق العمل المطلوب، قمنا بإنجاز مجسم تقريبي للمنزل. لتجسيد ذلك قمنا بربط المستقبلات مع المشغلات الميكانيكية، أيضا قمنا ببرمجة السيرفرات لكي نقوم في الأخير بانشاء تطبيق العميل.

الكلمات المفتاحية : انترنت الأشياء، خدمات الويب، ويب الأشياء، التدبير المنزلي.

Abstract

The Internet of Things (or IOT) is about connecting multiple objects together using predefined communication protocols. These objects can be either sensors or actuators, which allow to measure the environment in an optimal way. Connected objects interact by adding web concepts to the Internet of Things, which we call the Web of Things. This challenge has posed various problems, especially due to the collaboration among physical objects, and the dynamic nature and heterogeneity of the data and systems that are making up the web of things (powerful/small devices, fixed/mobile, batteries/continuous power supplies, etc.).

These requirements need compatible tools and methods for the realization of applications able to extract the useful information from the various data sources available and to interact with the environment, through actuators and with dedicated interfaces for user exploitation.

The work carried out, which is the subject of this document, consists of the realization of an application that allows collaboration in the Internet of Things. The main objective of this work is to use a web service as a means of ècommunication and data exchange between objects without worrying about how the objects are connected to the network. In order to validate and concretize our solution, we realized a prototype in the field of home automation. For this we assembled sensors and actuators, we implemented the servers and finally implemented the client application.

Keywords: Internet of Things, Service web, Web of things, Automation

Table de matière

Liste des figures

Liste des tableaux

Liste d’abréviations

A

AJAX : Asynchronous JavaScript and XML

API : Application Protocol Interface

AMQP : Advanced Message Queuing Protocol

C

COAP : Certified Open Adoption Practitioner

D

DHCP : Dynamic Host Configuration Protocol

G

GSM : Global System for Mobile Communication

GPRS : General Packet Radio Service

GCC : GNU Compiler Collection

H

HTTP : Hypertext Transfer Protocol

HIPAA : Health Insurance Portability and Accountability Act

I

IOT : Internet of Things

IP : Internet Protocol

IDE : Integrated Development Environment

J

JSON : JavaScript Objet Notation

L

LED : Light Emitting Diode

LTE : Long-Term Evolution

M

MQTT : Message Queuing Telemetry Transport

M2M : Machine To Machine

MVVM : Model View ViewModel

N

NFC : Near Field Communication

P

PIB : Produit Intérieur Brut

PSTN : Public Switched Telephone Network

Q

QR code : Quick Response

R

RFID : Radio Frequency IDentification

REST : Representational State Transfer

RPC : Remote procedure call

S

SGBD : Système de Gestion de Base de Données

SOAP : Simple Object Access Protocol

SDK : Software Development Kit

T

TCP : Transmission Control Protocol

U

UIT : Union internationale des télécommunications

UDP : User Datagram Protocol

UDDI : Universal Description Discovery and Integration

URL : Uniform Resource Locator

URI : Uniform Ressource Identifier

UML : Unified Modeling Language

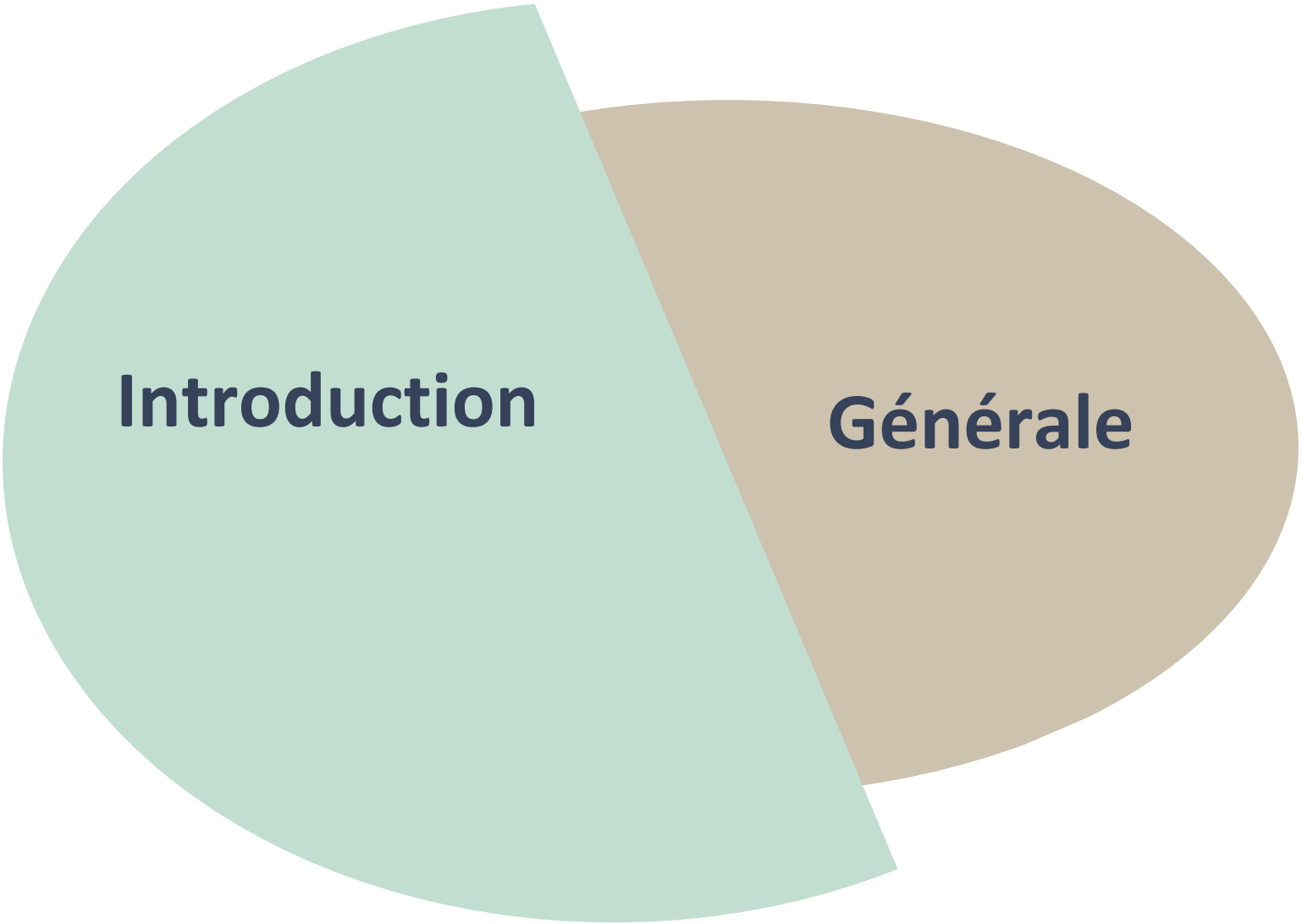
W

WOT : web of Things

WSDL : Web Services Description Language

X

XML : Extensible Markup Language


Introduction
Générale

Introduction générale

L'internet a changé notre vie professionnelle et privée au cours des dernières années et continue de le faire. Le Web 2.0, les réseaux sociaux et l'accès à l'internet mobile ne sont que quelques-uns des développements actuels dans ce contexte. Actuellement, les utilisateurs peuvent se connecter depuis presque n'importe quel endroit et accéder aux réseaux à tout moment grâce à une connectivité permanente (large bande avec ou sans fil). La révolution technologique a connecté les objets inanimés et les choses (Things) aux réseaux de communication à tout moment, en tout lieu, par n'importe qui et n'importe quel objet. Les produits de consommation pourraient être suivis à l'aide de minuscules émetteurs radio ou étiquetés avec des hyperliens et des capteurs intégrés. Comme l'illustre la figure 1.

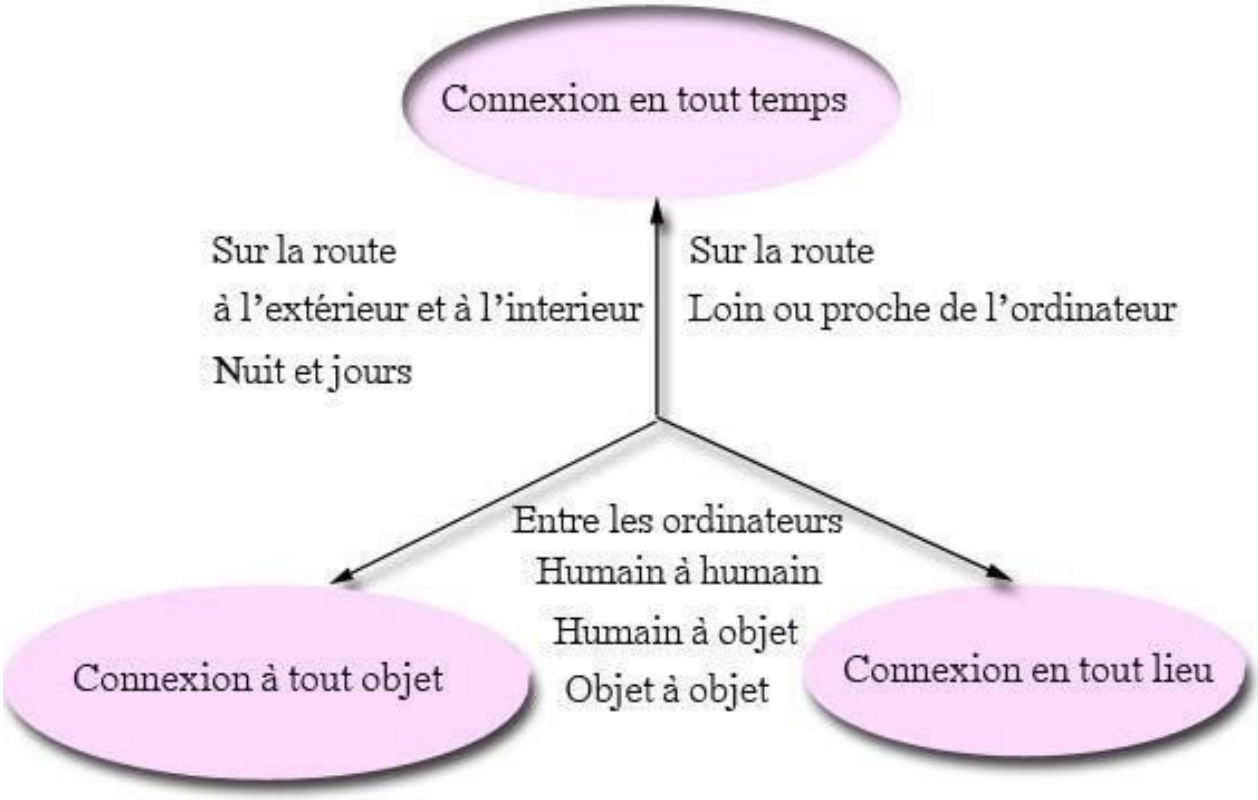


FIGURE 1 : LES TROIS DIMENSIONS PRINCIPALES D'IOT [1]

La connectivité a pris une toute nouvelle dimension, Le réseau mondial d'aujourd'hui est composé non seulement d'êtres humains et d'appareils électroniques, mais aussi de toutes sortes d'objets inanimés. Ces objets sont capables de communiquer avec d'autres objets, par exemple les réfrigérateurs avec les épiceries, les machines à laver avec les vêtements, les étiquettes implantées avec les équipements médicaux et les véhicules avec les objets fixes et mobiles. [1]

Les systèmes domotiques connaissent une croissance rapide. Pour apporter confort, commodité, qualité de vie et sécurité aux habitants, aux résidents. Aujourd'hui, la plupart des systèmes domotiques ne facilitent pas la vie que pour les personnes âgées et handicapées, ils permettent aussi de réduire le travail humain dans la production de services et de biens. Le système domotique peut être conçu et développé en utilisant un seul contrôleur qui a la capacité de contrôler et de surveiller différents appareils interconnectés tels que les prises de courant, les lumières, les capteurs de température et d'humidité, des détecteurs de fumée, de gaz et d'incendie, ainsi que des systèmes d'urgence et de sécurité. L'un des plus grands avantages du système domotique est qu'il peut être contrôlé et géré facilement à partir d'un large éventail de dispositifs tels que les smartphones, les tablettes, les ordinateurs de bureau et les ordinateurs portables. La croissance rapide des technologies sans fil nous incite à utiliser des smartphones pour contrôler et surveiller à distance les appareils domestiques dans le monde entier. Plusieurs systèmes domotiques utilisent des smartphones pour communiquer avec des microcontrôleurs en utilisant diverses techniques de communication sans fil telles que Bluetooth, GSM, ZigBee, et Wi-Fi.

Des applications Smartphone sont utilisées pour se connecter au réseau afin que les utilisateurs autorisés puissent ajuster les paramètres du système sur leurs appareils personnels. Les différents types de systèmes domotiques offrent un large éventail de fonctions et de services, dont certaines caractéristiques communes sont le contrôle des appareils, le contrôle du thermostat, le contrôle à distance de l'éclairage, la vidéo surveillance en direct, le contrôle des caméras de sécurité, les alertes textuelles en temps réel. [2]

1. Problématique

Le rôle principal de l'Internet des objets (IoT) est de connecter des objets physiques à l'Internet. Le Web des objets (WoT) représente une spécification d'internet des objets en ajoutant des normes et des technologies du web. L'arrivé de l'internet des objets et du web des objets a offert une grande capacité pour le développement de nouveaux services et applications connectant le monde physique au monde virtuel. Il existe de multiples plateformes et applications pour l'internet des objets. Cependant, peu de travaux se sont intéressés à la collaboration des dispositifs de l'internet des objets.

Le domaine d'application étant le domaine de la domotique. Dans le cas le plus simple, un objet connecté (comme un capteur) ayant récolté des informations sur l'environnement va chercher à les transmettre. Pour cela l'objet connecté doit transmettre ces données vers Internet via le routeur de la maison, elles seront par la suite retransmises aux différents objets concernés.

Afin d'assurer la collaboration entre les objets, il faut faire face à des différents défis, tels que :

- L'hétérogénéité des objets à savoir les objets utilisent des protocoles de communication différents (MQQT, http, COAP…), différents supports de communication (wifi, bluetouth, Zigbee, SIGfox, lora), différents langages de programmation des objets (c++, microPython).
- Il faut résoudre les problèmes de transmission des données au temps réel.
- Il faut avoir une solution pour synchroniser les objets afin qu'ils puissent collaborer sur une tâche précise.

Comment assurer la connectivité et la synchronisation entre tous ces objets et applications qui sont à la fois hétérogènes et diffuser l'information dans de nombreux format afin qu'ils puissent collaborer ?

2. Objectif

L'objectif général de ce travail est la réalisation d'une application permettant la collaboration dans l'Internet des objets en utilisant les web services. En fait, le service web (ou service de la toile) est un protocole d'interface informatique de la famille des technologies web permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Nous proposons donc d'utiliser les services web pour faire collaborer des éléments ou objets de l'internet des objets.

Chapitre 1 :

État de l’art

Chapitre 1 : l'état d'art

1.1 Internet des objets

Le terme IoT peut être attribué à Kevin Ashton en 1997, lors de ses travaux chez Proctor and Gamble, où il utilisait des étiquettes RFID[1] pour gérer les chaînes d'approvisionnement [21].

La RFID fonctionne en transmettant par ondes radio des informations codées sur une étiquette RFID à un lecteur, puis à une base de données informatique à partir de laquelle les informations peuvent être utilisées pour prendre des décisions opportunes, voire automatisées [21].

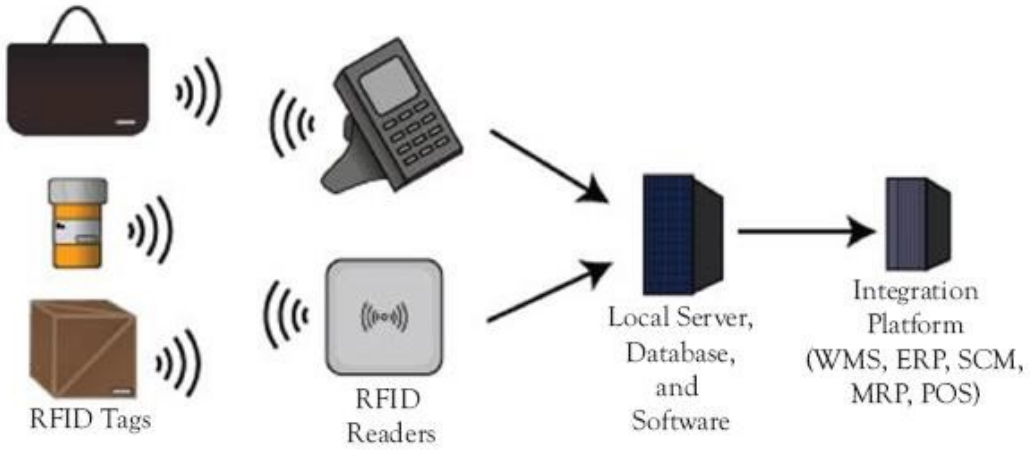


FIGURE 2 : FONCTIONNEMENT D'UN SYSTÈME RFID [21]

Un système RFID se compose généralement d'un ordinateur, d'une base de données, d'un logiciel RFID, de lecteurs RFID, d'un intergiciel RFID, d'antennes RFID et d'étiquettes RFID (transpondeurs) et il est fréquemment intégré à d'autres systèmes de l'organisation (figure 2) [21]. Depuis, L'IOT est passé de simples étiquettes RFID à un écosystème et une industrie qui en 2020, cannibalisera[2], créera ou déplacera cinq mille milliards de dollars sur les cent mille milliards du PIB mondial, soit 6 % du PIB mondial [22].

[1] La radio-identification, le plus souvent désignée par l'acronyme RFID, est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes »

[2] Dans la stratégie de marketing, la cannibalisation fait référence à une réduction du volume des ventes, du chiffre d'affaires ou de la part de marché d'un produit suite à l'introduction d'un nouveau produit par le même producteur.

Jusqu'en 2012, le concept d'objets connectés à l'internet concernait principalement les smartphones, les tablettes, les PC et les ordinateurs portables. Essentiellement, des choses qui ont d'abord fonctionné à tous égards comme un ordinateur [22].

L'internet des objets (Iot) permet à l'homme de relever efficacement divers défis de la société moderne en collectant et en traitant des informations détaillées sur les événements et les environnements grâce à des milliards d'objets connectés, rendant ainsi la vie humaine plus sûre, plus saine, plus productive et plus confortable. En même temps, cela créerait des opportunités commerciales substantielles pour diverses industries verticales et secteurs sociaux tels que l'automobile, l'énergie et les services publics, le transport, la logistique, les villes intelligentes, les soins de santé, les sports et la sécurité publique. [19]

1.1.1 Définition

Il n'existe pas de consensus clair sur ce qu'est l'IoT. Différentes organisations ont proposé des définitions légèrement différentes. Selon l'Union internationale des télécommunications (UIT), c'est « Une infrastructure mondiale pour la société de l'information, permettant de fournir des services avancés en interconnectant des objets (physiques et virtuels) sur la base des technologies de l'information et de la communication interopérables existantes et évolutives. » [20]

- **Dispositif :** dans l'Internet des objets, équipement doté obligatoirement de capacités de communication et éventuellement de capacités de détection, d'actionnement, de saisie de données, de stockage de données et de traitement de données. [23]
- **Objet** : dans l'Internet des objets, objet du monde physique (objet physique) ou du monde de l'information (objet virtuel), pouvant être identifié et intégré dans des réseaux de communication. [23]

1.1.2 L'impact de l'IoT

a) La santé

Le secteur de la santé sera en concurrence avec l'industrie et la logistique pour la première place en termes de revenus et d'impact sur l'IoT. Tous les systèmes qui améliorent la

qualité de vie et réduisent les coûts de santé sont une préoccupation majeure dans presque tous les pays développés. L'IoT est sur le point de permettre une surveillance à distance et flexible des patients, où qu'ils se trouvent. Des outils d'analyse avancée et d'apprentissage automatique observeront les patients afin de diagnostiquer les maladies et de prescrire des traitements. Ces systèmes seront également les gardes malades en cas de soins vitaux nécessaires. Actuellement, on compte environ 500 millions de moniteurs de santé portables, avec une croissance en double du chiffre dans les années à venir.

Les contraintes qui pèsent sur les systèmes de santé sont importantes. De la conformité à la loi HIPAA à la sécurité des données, les systèmes IoT doivent agir comme des outils et des équipements de qualité hospitalière. Les systèmes sur le terrain doivent communiquer avec les centres de soins 24 heures sur 24, 7 jours sur 7, de manière fiable et sans aucun temps d'arrêt si le patient est surveillé à domicile. Les systèmes peuvent avoir besoin d'être sur un réseau hospitalier tout en surveillant un patient dans un véhicule d'urgence [22].

- **Cas d'utilisation de l'IoT dans le secteur de la santé**

  Voici quelques-uns des cas d'utilisation de l'IoT dans le domaine de la santé [22] :

  - Soins aux patients à domicile
  - Modèles d'apprentissage de soins de santé prédictifs et préventifs
  - Soins et suivi des personnes âgées et des personnes atteintes de démence
  - Suivi des équipements hospitaliers et des biens d'approvisionnement
  - Suivi et sécurité des produits pharmaceutiques
  - Médecine de terrain à distance
  - Recherche de médicaments
  - Indicateurs de chute des patients

b) Transport

IoT dans les cas d'utilisation dans le domaine des transports connaît une croissance rapide. Elle permet de réaliser des gains en termes d'efficacité opérationnelle, de réduction des coûts, de sécurité et de mobilité. Les cas d'utilisation impliquent le suivi de l'actif sur les appareils livrés, transportés ou expédiés, que ce soit sur un camion, un train, un avion ou un bateau. C'est aussi le domaine des véhicules connectés qui

communiquent pour offrir une assistance au conducteur, ou une maintenance préventive des véhicules [22].

- **Cas d'utilisation de l'IoT dans le secteur du transport**

  Voici quelques-uns des cas d'utilisation de l'IoT dans le domaine du transport et de la logistique [22] :

  - Suivi et localisation des cargos
  - Identification et suivi des wagons
  - Suivi des biens et des colis au sein des cargos
  - Entretien préventif des véhicules sur la route

c) L'énergie

De nombreuses installations de production d'énergie se trouvent dans des environnements éloignés ou hostiles, comme des régions désertiques pour les panneaux solaires, des collines escarpées pour les parcs éoliens et des installations dangereuses pour les réacteurs nucléaires. En outre, les données peuvent nécessiter une réponse en temps réel ou quasi réel pour les systèmes de contrôle de la production d'énergie [22].

- **Cas d'utilisation de l'IoT énergétique**

  Voici quelques-uns des cas d'utilisation de l'IoT dans énergétique :

  - Analyse de la plate-forme pétrolière à partir de milliers de capteurs et de points de données pour des gains d'efficacité.
  - Surveillance et maintenance à distance des panneaux solaires.
  - Analyse des risques des installations nucléaires.
  - Compteurs électriques intelligents dans le cadre d'un déploiement à l'échelle de la ville pour surveiller la consommation et la demande d'énergie.
  - Ajustement en temps réel des pales d'hélice en fonction de la météo sur de longue distantes.

d) L'industrie et fabrication

Le rôle de l'iot dans industrie est la maintenance prédictive de milliers de machines d'usine et de production pour fournir une quantité de données sans précédent aux infrastructures de cloud privé et public [22].

- **Cas d'utilisation de l'IoT dans l'industrie et la fabrication**

Voici les cas d'utilisation de l'IoT dans l'industrie et la fabrication et leur impact [22] :

- Maintenance préventive sur les machines d'usine nouvelles et préexistantes.
- Augmentation du débit grâce à la demande en temps réel
- Économies d'énergie
- Systèmes de sécurité tels que la détection thermique, la détection de la pression et les fuites de gaz.
- Systèmes experts d'usine

e) Les villes intelligentes

Les objectifs fondamentaux d'une ville intelligente sont, d'une part, l'optimisation de son organisation et de son fonctionnement, et, d'autre part, l'amélioration de son impact sur l'environnement et la qualité de vie de ses habitants [22].

- **Cas d'utilisation de l'IoT dans les villes intelligentes**

Les cas d'utilisation de l'IoT dans les villes intelligentes sont les suivants [22] :

- Contrôle de la pollution et analyse réglementaire grâce à la détection environnementale.
- Prévisions météorologiques microclimatiques à l'aide de réseaux de capteurs à l'échelle de la ville.
- Gains d'efficacité et réduction des coûts grâce à un service de gestion des déchets à la demande.
- Amélioration de la fluidité du trafic et des économies de carburant grâce à un contrôle intelligent des feux de circulation.
- Efficacité énergétique de l'éclairage urbain à la demande.
- Déneigement intelligent basé sur la demande routière en temps réel, les conditions météorologiques.
- Arrosage intelligent des parcs et des espaces publics, en fonction de la météo et de l'utilisation actuelle
- Des caméras intelligentes pour surveiller les crimes et les alertes automatisées en temps réel.
- Des parkings intelligents pour trouver automatiquement les meilleures places de stationnement à la demande.

1.1.3 L'écosystème d'IoT

1.1.3.1 Coté Hardware et Software

Les industries s'appuieront sur le matériel, les logiciels et les services fournis par l'essentiel de l'industrie informatique. Presque toutes les grandes entreprises de technologie investissent ou ont investi massivement dans l'espace IoT. De nouveaux marchés et de nouvelles technologies se sont déjà formés. Nous allons aborder presque tous les segments des technologies de l'information, car ils ont tous un rôle à jouer dans l'IOT [53] :

- **Capteurs** : Systèmes embarqués, systèmes d'exploitation en temps réel, sources d'énergie, systèmes micro-électromécaniques (MEM). D'énergie, systèmes micro-électro-mécaniques (MEM).
- **Systèmes de communication par capteurs** : Les réseaux personnels sans fil s'étendent de 0 cm à 100 m. Les canaux de communication à faible vitesse et à faible puissance, souvent non basés sur le protocole IP, ont leur place dans la communication entre capteurs. Ont leur place dans la communication par capteurs
- **Les réseaux locaux** : Il s'agit généralement de systèmes de communication basés sur le protocole IP, tels que le Wi-Fi 802.11, utilisés pour les communications radio rapides, souvent dans des topologies en étoile ou de pair à pair.
- **Agrégateurs, routeurs, passerelles** : Fournisseurs de Systèmes embarqués, vendeurs les moins chers (processeurs, DRAM et stockage), vendeurs de modules, fabricants de composants passifs, fabricants de clients légers, fabricants de radios cellulaires et sans fil, fournisseurs d'intergiciels, fournisseurs de cadres de brouillard, progiciels d'analyse périphérique, fournisseurs de sécurité périphérique, systèmes de gestion de certificats.
- **WAN** : fournisseurs de réseaux cellulaires, fournisseurs de réseaux satellitaires, fournisseurs de réseaux étendus à faible puissance (LPWAN). Utilise généralement des protocoles de transport Internet ciblés sur l'IOT et les dispositifs à contraintes, comme MQTT, CoAP, et même HTTP.
- **Cloud** : Fournisseur d'infrastructure en tant que service, fournisseur de plateforme en tant que service, fabricants de bases de données, progiciels d'analyse de données, fournisseur de logiciels en tant que service, fournisseurs de lacs de données, fournisseurs de réseaux définis par logiciel/périmètre, et services d'apprentissage automatique.
- **L'analyse des données** : analyse des données et techniques d'apprentissage automatique.
- **La sécurité** : La sécurité touchera chaque composant, des capteurs physiques à l'unité centrale et au matériel numérique, en passant par les systèmes de communication radio et les protocoles de communication. Chaque niveau doit garantir la sécurité, l'authenticité et l'intégrité. Il ne peut y avoir de maillon faible dans une chaîne, car l'IOT constituera la plus grande surface d'attaque au monde.

1.1.3.2 Coté réseau

- **Architecture IOT**

Une architecture IoT, englobera de nombreuses technologies, les complexités et les relations de l'IOT sont nettement plus complexes que les technologies traditionnelles en raison de l'échelle mais aussi des types d'architecture hétérogènes. De plus nous devons faire des choix de protocoles Internet tels que MQTT par rapport à CoAP et AMQP, et comment cela fonctionnera s'il décide de migrer vers un autre fournisseur de cloud. Les choix doivent également être pris en considération en ce qui concerne l'endroit où le traitement doit résider. Il existe désormais plus de 1,5 million de combinaisons différentes d'architectures possibles comme le montre la Figure 3 [51].

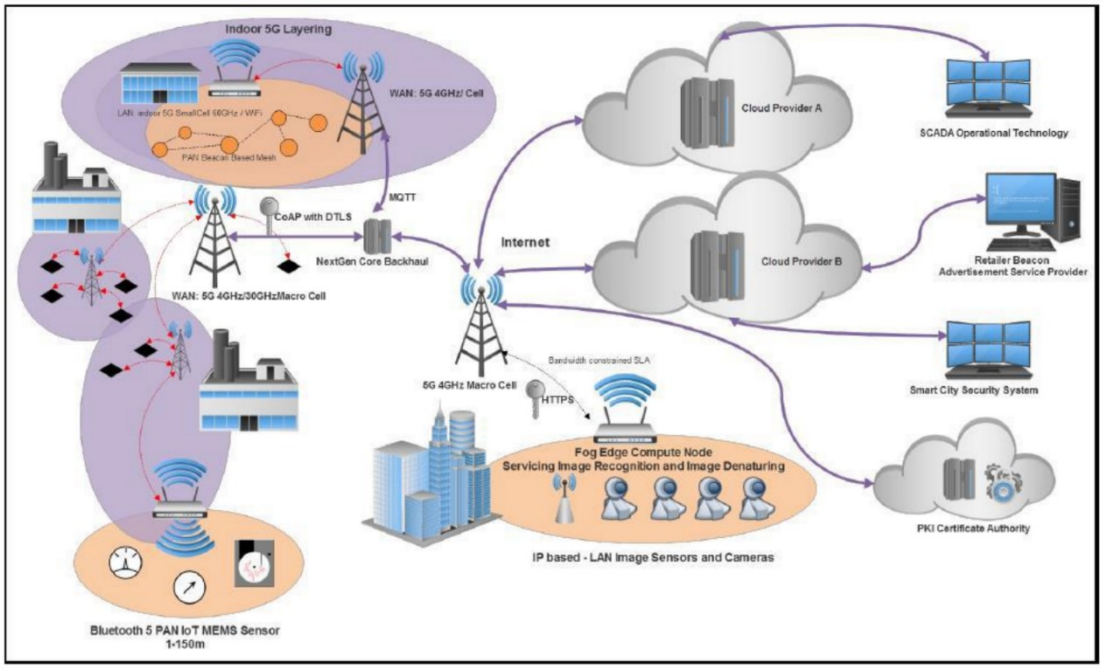


FIGURE 3 : LE SPECTRE COMPLET DES DIFFERENTS NIVEAUX D'ARCHITECTURE IOT, DU CAPTEUR AU CLOUD ET INVERSEMENT [51]

L'IoT et les appareils portables, d'autre part, peuvent être très limités en termes de puissance et de bande passante, ce qui déclenche le besoin de protocoles plus efficaces, sécurisés et évolutifs pour gérer différents types d'appareils communiquant via différentes topologies de réseau. La couche application sert d'interface entre l'appareil et l'utilisateur final. Vous trouverez ci-dessous quelques protocoles de couche application largement utilisés [51] :

- **Message Queuing Telemetry Transport (MQTT)**

MQTT est Probablement la norme la plus largement adoptée dans le domaine de l'IoT et des appareils portables. MQTT est un protocole de messagerie léger destiné aux appareils connectés fonctionnant sur batterie, il a été spécialement conçu pour les réseaux de communication peu fiables afin de desservir le nombre croissant d'objets intelligents à faible consommation d'énergie.

MQTT est basé sur un modèle d'abonné et courtier. Le capteur peut être configuré pour être un éditeur, l'application peut être configurée en tant qu'abonné, et un système intermédiaire est configuré pour agir en tant que courtier pour relayer les informations entre l'éditeur et l'abonné.

- **Constrained Application Protocol (CoAP)**

CoAP est un protocole de la couche application conçu pour être utilisé dans des appareils Internet à ressources limitées. Bien que l'infrastructure Internet existante soit disponible gratuitement et puisse être utilisée par n'importe quel appareil connecté, elle est souvent trop lourde et consommatrice d'énergie pour la plupart des applications de l'IoT et de la technologie portable. Conçu pour simplifier l'intégration des systèmes IoT basés sur http avec le Web, CoAP s'appuie sur le protocole UDP (User Datagram Protocol) pour assurer une communication sécurisée et à faible surcharge entre les nœuds du réseau.

- **AMQP (Advanced Message Queuing Protocol)**

AMPQ est un protocole de couche d'application ouvert conçu pour permettre l'interopérabilité dans des environnements middleware orientés messages

Les solutions IoT s'appuient sur différents protocoles et appareils radio. En général, les protocoles radio doivent couvrir de longues distances (voire des kilomètres) et être économes en énergie. Les appareils doivent également être économes en énergie, car ils sont déployés pendant des mois, voire des années, dans des environnements difficiles. Un cycle de sommeil est un mécanisme qu'ils utilisent pour économiser de l'énergie, généralement coordonné par les « loggers/gateways » ou préprogrammé. Les « loggers » sont déployés à proximité des capteurs et disposent de plus d'espace de stockage. Ils servent de tampons entre les capteurs et les services supérieurs. Souvent, les « loggers » et les capteurs sont intégrés dans la même carte, sinon, ils sont connectés à l'aide de câbles ou de protocoles radio de proximité. D'autre part, les « gateways » servent de point de collecte des données. Ces dispositifs sont beaucoup plus performants et consomment généralement plus d'énergie. Dans certains scénarios de déploiement, les passerelles hébergent un système d'exploitation complet avec plusieurs logiciels installés. Sinon, les passerelles servent uniquement de relais des données envoyées par les enregistreurs et les capteurs aux services en cloud et vice-versa. Les services en cloud peuvent être partiellement hébergés dans des serveurs périphériques afin de préserver la confidentialité des données et la réactivité de l'ensemble de la solution IoT [52].

La topologie complète de déploiement d'une solution IoT est décrite dans le schéma ci-dessous figure 4 :

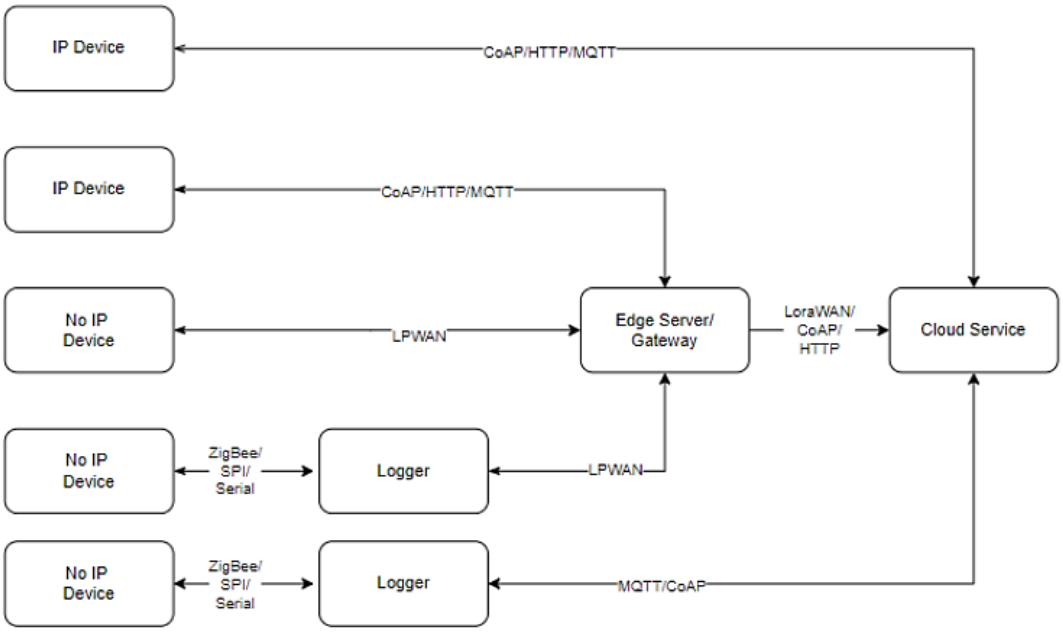


FIGURE 4 LA TOPOLOGIE COMPLÈTE DE DÉPLOIEMENT D'UNE SOLUTION IOT [52]

1.1.4 Les principales exigences de l'IoT

L'infrastructure de l'IoT doit satisfaire à un certain nombre d'exigences pour pouvoir être déployée économiquement et technologiquement pour des services et des applications utiles. La recommandation ITU-T Y.2066 énumère plusieurs conditions exigences communes de l'IoT.

Ces exigences ont été classées en deux groupes : non fonctionnelles et fonctionnelles. Les exigences non fonctionnelles sont liées à la mise en œuvre et au fonctionnement, tandis que les exigences fonctionnelles sont liées aux applications, aux services, à la communication, aux dispositifs, à la gestion des données et à la sécurité

Nous présentons ci-dessous plusieurs autres exigences importantes liées aux réseaux de communication de l'IoT. Telles que la confiance, la fiabilité, la possibilité de partage, la prise en charge de différents modes de communication (par exemple, la prise en compte des services, la prise en compte des données, la prise en compte de l'utilisateur), la communication basée sur l'identification indépendante de l'emplacement, la communication hétérogène, l'attribution de noms, la numérotation et l'identification, les interfaces de programmation d'applications ouvertes et les systèmes configurables à distance [19-24].

a) Infrastructure partageable

La mise en réseau centrée sur l'utilisateur permet aux fournisseurs de services IoT et aux utilisateurs d'avoir un contrôle total de leurs appareils ou des données qu'ils génèrent. Des

outils permettant de fournir différents niveaux de sécurité et de confidentialité devraient exister pour permettre aux utilisateurs de choisir le niveau le plus approprié qui répond de manière optimale à leurs besoins.

b) Nommage et identification évolutifs

Selon la définition de l'IoT, tout ce qui existe dans le monde physique et dans le monde de l'information doit pouvoir être identifié et intégré dans des réseaux de communication. Compte tenu du volume et de la diversité des usages et des capacités des objets, il doit exister des systèmes de dénomination et d'identification évolutifs capables de gérer des espaces de noms hétérogènes.

c) Communication hétérogène indépendante de la localisation

L'IoT doit pouvoir intégrer des types hétérogènes de protocoles de couche réseau tels que IPv4, IPv6, les réseaux cellulaires (2G, 3G, 4G, 5G) et le PSTN, ainsi que des technologies sans fil telles que ZigBee, Bluetooth et Wifi. Les appareils doivent pouvoir se déplacer sans heurts d'un endroit à l'autre dans des réseaux hétérogènes. En outre, les dispositifs situés dans un type de réseau doivent être en mesure de communiquer avec des dispositifs situés dans d'autres types de réseaux.

d) Auto configurable et contrôlable à distance

Étant donné que la plupart des dispositifs IoT effectuent des communications M2M (machine to machine), ils doivent être auto configurables et contrôlables à distance. L'auto configuration permet aux appareils de se connecter dynamiquement à un réseau (ou de le former), tandis que le contrôle et la gestion à distance permettent de surveiller leur état, de mettre à jour leur logiciel et de les reconfigurer.

e) Interfaces de programmation d'applications ouvertes

Pour permettre la mise à disposition de différents types d'applications et de services sur une infrastructure IoT partagée, il devrait exister des interfaces de programmation d'applications (API) ouvertes et communes. Elles permettraient de partager l'infrastructure de l'IoT entre de nombreuses applications de différents domaines.

1.2 Web service

Le web est devenu populaire vu que les gens ordinaires peuvent l'utiliser pour faire des choses vraiment utiles avec un entraînement minimal. Mais en coulisses, le web est aussi une plateforme puissante pour l'informatique distribuée. [6]

Les bases de données, les sites web et les applications métier doivent échanger des données. Pour ce faire, des formats de données standard sont définies tels que XML ou JSON, ainsi que des protocoles de transfert ou des services Web tels que SOAP ou REST. Les développeurs doivent souvent concevoir leurs propres interfaces de programmation d'applications (API) pour que les applications fonctionnent tout en intégrant une logique métier spécifique autour des systèmes d'exploitation ou des serveurs. [5]

1.2.1 Définition

1.2.1.1 Calcul distribué ou système distribué

Un système distribué est un système qui s'exécute sur un ensemble de machines sans mémoire partagée. Pour l'utilisateur final du système, la machine apparaîtra comme un système non distribué. [7]

En une autre reformulation, Un système distribué est un ensemble de machines autonomes connectées par un réseau, et équipées d'un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ses ressources. [8]

Il sert à résoudre des problèmes d'une façon optimale et flexible en utilisant des applications distribuées en respectant aussi des technologies précises. Le transport des messages est assuré par des protocoles internet tels que TCP/IP et UDP (figure 5).

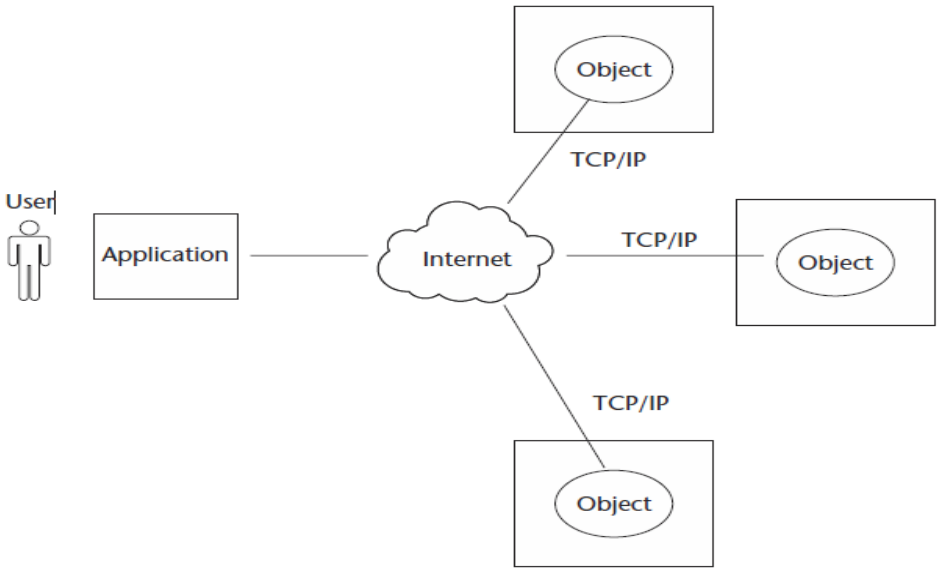


FIGURE 5 : SCHÉMA REPRÉSENTE UN MODÈLE DE SYSTÈME DISTRIBUÉ [36]

1.2.1.2 Web service

Les web services sont des « applications » modulaires et autonomes ou une logique d'application développée selon un ensemble de normes ouvertes. Ils ne sont généralement pas destinés à être des applications à part entière et riche en fonctionnalités. [1]

Les services web peuvent fournir des fonctions, qui peuvent aller de l'information, par exemple un service de prévisions météorologiques, à des demandes simples ou des activités complexes ayant un effet dans le monde réel. [3]

Selon w3c [34] :

Un Web service est un composant logiciel identifié par une URI, dont les interfaces publiques sont définies et appelées en XML ou JSON. Sa définition peut être découverte par d'autres systèmes logiciels. Les services Web peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML ou JSON portés par les protocoles Internet.

1.2.2 Les protocoles de Web service

Les Web services permettent d'intégrer des systèmes disparates et d'exposer des fonctions professionnelles réutilisables via HTTP. Ils exploitent HTTP comme un simple moyen de transport des données (par exemple, les services SOAP/WSDL) ou l'utilisent comme un protocole d'application complet qui définit la sémantique du comportement du service (par exemple, les services RESTful) [35] comme le montre la figure 6.

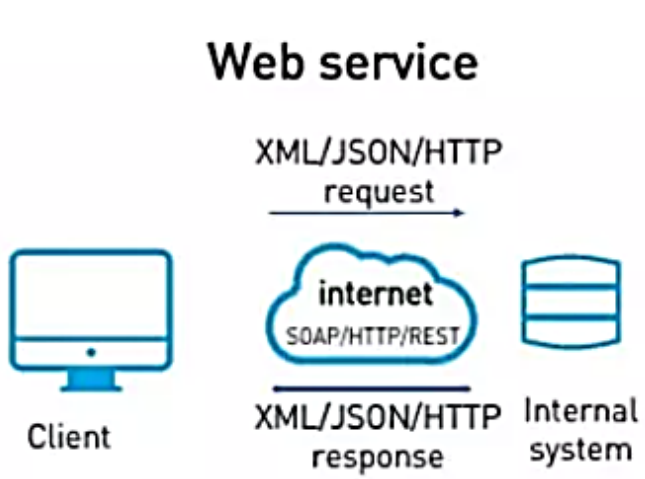


FIGURE 6 : ARCHITECTURE DE WEB SERVICE [30]

Il existe deux web service SOAP et REST.

1.2.2.1 Web service basé sur SOAP

a) L'utilisation de protocole SOAP a plusieurs caractéristiques :

- La sécurité
- Omniprésent
- Robustesse du langage XML

- Avoir des données fortement typées
- Complexité
- Surcharge XML
- Les réponses ne peuvent pas être mises en cache

b) SOAP :

SOAP est actuellement un bloc de construction fondamental et préalable pour les services web XML. SOAP est utilisé pour envoyer et recevoir des sorties de services web XML conventionnels. Ainsi, c'est le mécanisme de communication sous-jacent pour les services web. Étant donné qu'un service web nécessite des paramètres d'entrée pour être activé, SOAP fournit les paramètres d'entrée nécessaire pour un service web. [17]

Il Permet la transmission de messages entre les objets distants pour assurer la communication, par le moyen d'échanges de message, entre le client et le fournisseur de service. ***« SOAP est un protocole de la famille XML servant à l'échange d'informations dans un environnement distribué et décentralisé. Il est considéré comme la technologie la plus importante des Web Services. » [4]***

Il utilise deux styles de communication :

- RPC (Remote Procedure Call) : pour l'appel des procédures distantes.
- DOC (document) : pour l'échange des messages conformes à des schémas arbitraires.

c) WSDL :

WSDL est un langage permettant de décrire et publier le format et les protocoles d'un Web Service de manière homogène par l'utilisation de XML. Cela permettra au requérant et à l'émetteur d'un service de comprendre les données qui seront échangées. [4]

d) UDDI :

UDDI définit les mécanismes permettant de répertorier des Web Services. Ce standard régit donc l'information relative à la publication, la découverte et l'utilisation d'un Web Service. En fait, UDDI définit un registre des Web Services sous un format XML. Ce registre peut être public, privé ou partagé. [4]

e) XML :

XML est un langage de balisage textuel qui est standard pour l'échange de données sur le Web [5]. Il Permet de transformer Internet d'un univers d'information et de présentation de sites web statiques à un univers web programmable et dynamique, centré sur les données. Il est largement utilisé par les entreprises et supporté par les manufacturiers informatiques. Il est aussi indépendant des plates-formes informatiques. [4] Son rôle est de permettre le développement flexible entre les types de document définis par l'utilisateur, cela signifie qu'il fournit un format de dossier

persistant, robuste, non propriétaire et vérifiable qui peut être utilisé pour l'entreposage et la transmission de données sur le Web. [5]

SOAP, WSDL et UDDI se basent sur le langage XML. Les figure 7,8 représentent les fonctionnalités de chaque protocole cité avant.

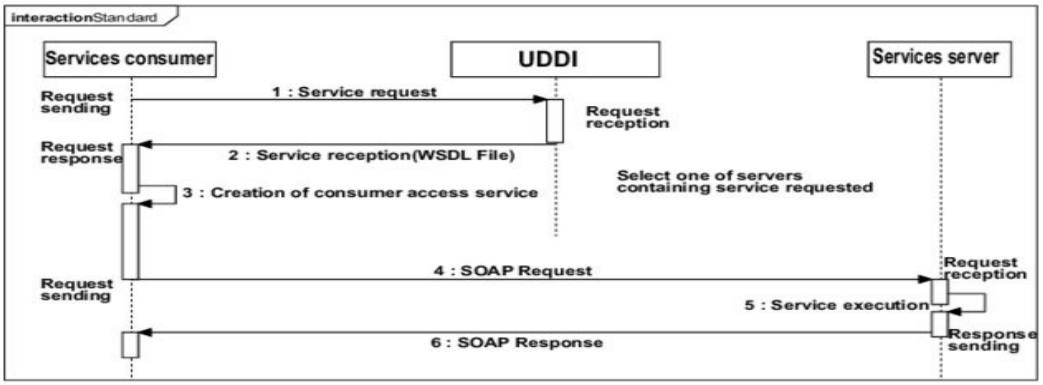


FIGURE 7 : DIAGRAMME DE SÉQUENCE SUR LE WEB SERVICE BASÉ SUR LE PROTOCOLE SOAP [37]

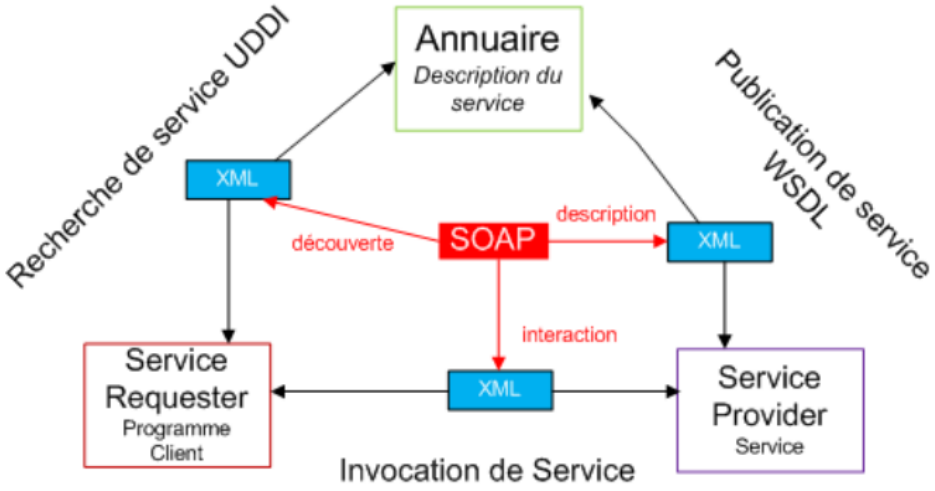


FIGURE 8 : SCHÉMA REPRÉSENTE LE WEB SERVICE BASÉ SUR LE PROTOCOLE SOAP [38]

1.2.2.2 Web service basé sur REST

a) Que signifie REST ?

REST est l'acronyme de Representational State Transfer. Il décrit un style d'architecture logicielle permettant de construire une application devant fonctionner sur des systèmes distribués, typiquement internet. [10]

Donc, ce n'est pas une technologie, ni un protocole à proprement parler mais plus un style d'architecture pour les systèmes distribués. Fondées par Roy Fielding. Les architectures REST sont particulièrement adaptées au web même si elles n'en dépendent pas et peuvent parfaitement s'adapter à d'autres protocoles que le HTTP. L'idée principale dans le paradigme REST est que les architectures sont soumises à un ensemble de contraintes

appliquées aux éléments de l'architecture. En examinant l'impact de chaque contrainte ajoutée au modèle, il est facile d'identifier les propriétés induites par les contraintes du web. D'autres contraintes peuvent alors être appliquées pour former un nouveau style architectural qui reflète mieux les propriétés souhaitées pour une architecture web plus moderne. [12] la figure 9 montre l'architecture REST.

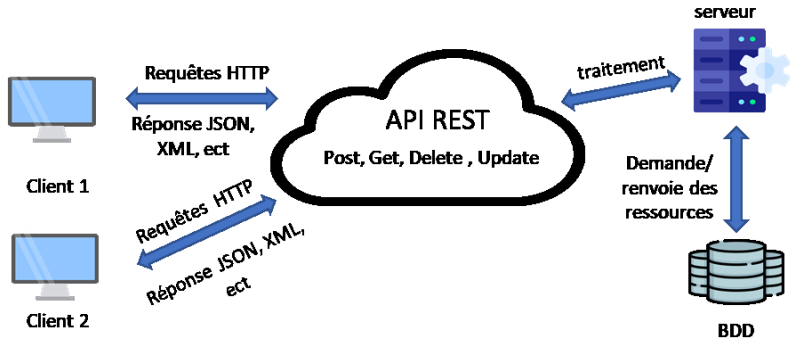


FIGURE 9: L'ARCHITECTURE REST

b) Les contraintes de REST

REST doit respecter 7 contraintes :

- Le modèle vide : il s'agit du système souhaité sans aucune distinction entre les composants constituant ce système. Le système est vu comme un seul bloc non encore segmenté. [12]
- Client-Serveur : première contrainte introduite au modèle REST. L'objectif est ici de séparer les couches métiers et présentation (interface). Cela améliore la portabilité de l'interface et facilite la scalabilité coté serveur en simplifiant les composants du serveur. [12]
- Indépendance de l'état du serveur : cette contrainte est ajoutée pour les communications client-serveur. Elle implique notamment que toute requête formulée par le client doit contenir l'intégralité des informations nécessaires à son exécution par le serveur et ne doit pas présupposer un quelconque état de celui-ci ou l'exploitation d'un quelconque contexte stocké coté serveur. [12]
- Cache : les réponses aux requêtes sont classées comme pouvant être mises en cache ou non selon la nature de la requête [12] dans le but d'améliorer la performance de réseau.
- Interface uniforme : l'une des principales caractéristiques qui distinguent REST des autres styles d'architectures logicielles est le fait que l'interface entre les composants est toujours présente peu importe les composants qui interagissent. [12]
- Système en couches : l'un des objectifs est de restreindre le comportement des composants de sorte qu'ils "voient" seulement la couche adjacente immédiate avec laquelle ils interagissent. Cela favorise une certaine indépendance entre les différentes couches et permet d'améliorer, grâce aux couches intermédiaires, la scalabilité et la répartition des charges (Load Balancing) engendrées par les services à travers différents réseaux et processeurs. [12]

- Code à la demande (optionnelle) : REST autorise l'extensibilité des fonctionnalités du client en téléchargeant et en exécutant du code sous forme d'applets ou de scripts. En particulier, ceci simplifie les clients en réduisant le nombre de fonctionnalités nécessaires à un client pré-implémenté. [12]

c) L'utilité de REST

Les systèmes distribués qui utilisent l'architecture REST apportent des améliorations dans les domaines suivants :

- Performance : le mode de communication proposé par REST est se veut efficace et simple, permettant des gains de performances sur les systèmes qui l'utilisent. [5]
- Évolutivité des interactions des composants : tout système distribué devrait bien gérer cet aspect, et simple Les interactions proposées par REST le permettent largement. [5]
- Interface simple : une interface simple permet des interactions entre les systèmes, qui à son tour peut apporter des avantages comme mentionné précédemment. [5]
- Modifiable des composants et des systèmes, et la séparation des préoccupations que REST propose, permet de modifier les composants indépendants les uns des autres avec un coût et un risque minime. [5]
- Portabilité : REST est indépendant de la technologie et du langage, ce qui signifie Il peut être mis en œuvre et utilisé par tout type de la technologie. [5]
- Fiabilité : les contraintes sans état proposées par REST permettent au système de se rétablir plus facilement après une panne. [5]

d) API

Les API ne sont pas nouvelles. Ils ont servi d'interfaces qui permettent aux applications de communiquer entre elles depuis des décennies. Mais le rôle des API a radicalement changé au cours des dernières années. Les entreprises innovantes ont découvert que les API peuvent être utilisées comme interface avec l'entreprise, ce qui leur permet de monétiser les actifs numériques, d'étendre leur proposition de valeur avec des capacités fournies par les partenaires et de se connecter aux clients sur tous les canaux et appareils. [5]

e) JSON

JSON ou JavaScript Object Notation est une norme ouverte légère basée sur du texte conçu pour l'échange de données lisibles par l'homme. JSON utilise des conventions bien connues des programmeurs, y compris ceux qui connaissent C, C++, Java, Python, Perl, etc. [5]

Travailler avec JSON est une expérience différente et supérieure. Premièrement, une syntaxe plus simple peut aider à éviter de choisir entre les nombreuses façons différentes de représenter les données, Il n'y a généralement qu'une seule façon directe d'exprimer quelque chose :

```
1 {"nom" : "Mohamed",
2 "Nom de famille" : "Cherfi"} [5]
```

Le format JSON est utilisé pour sérialiser et transmettre des données structurées via une connexion réseau. Il est principalement utilisé pour transférer des données entre des serveurs et des applications Web. [5]

f) HTTP

Le protocole HTTP (HyperText Transfer Protocol) est le protocole utilisé par les programmes pour communiquer sur le World Wide Web. Des nombreuses applications utilisent le protocole HTTP, mais HTTP est souvent célèbre pour les conversations bidirectionnelles entre les navigateurs Web et les serveurs Web. [14]

HTTP utilise des protocoles de transmission de données fiables, il garantit que les données ne seront pas endommagées ou brouillées en transit. [14]

Les messages HTTP sont les blocs de données envoyés entre les applications HTTP. Les blocs de données commencent par des méta-informations textuelles décrivant le contenu du message et la signification, suivie de données facultatives. Ces messages circulent entre les clients, serveurs et proxys. Les termes « entrant (Inbound) », « sortant (Outbound) », « en amont (Upstream) » et « en aval (Downstream) » décrivent l'orientation du message (figure 10,11). [14]

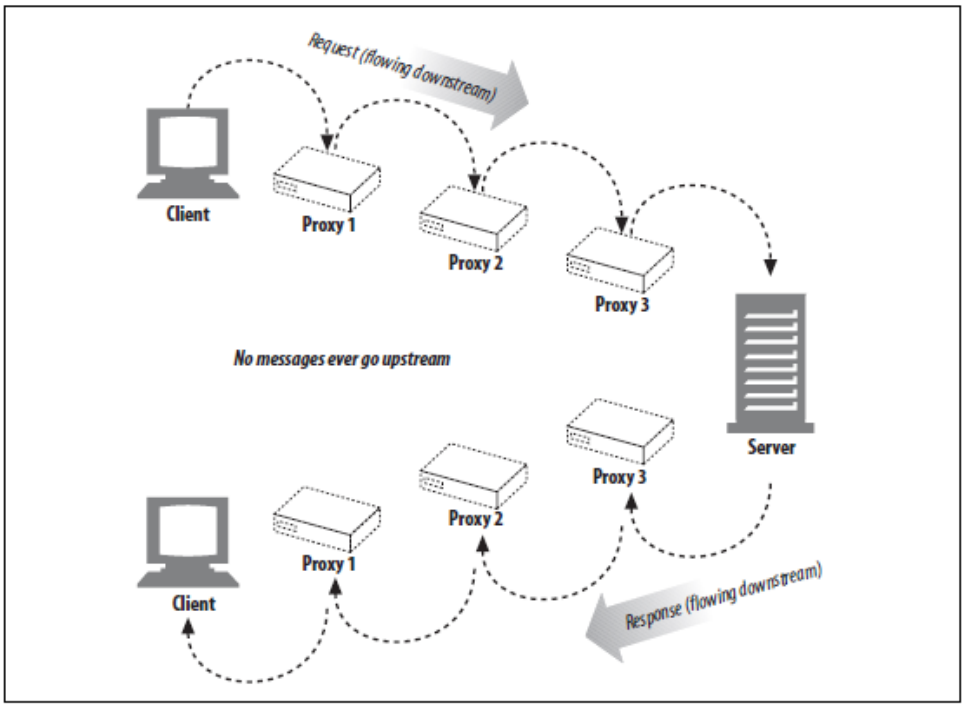


FIGURE 10: LE CYCLE DE VIE D'UN MESSAGE HTTP [14]

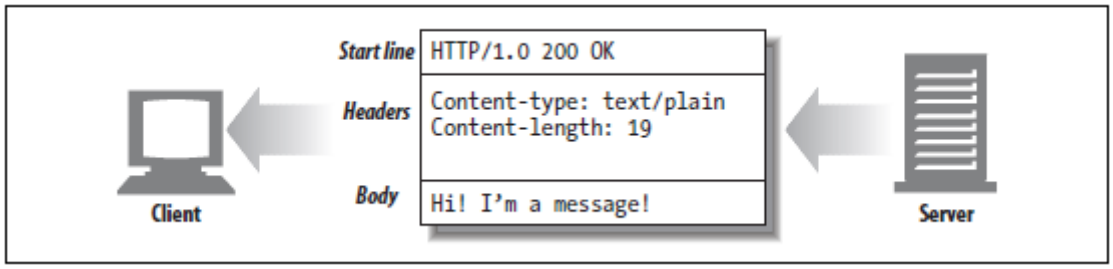


FIGURE 11 : LES COMPOSANTS D'UN MESSAGE HTTP [14]

g) Ressource

Une ressource est tout ce qui est suffisamment important pour être référencé. Habituellement, une ressource est quelque chose qui peut être stocké sur un ordinateur et représenté sous la forme d'un flux de bits : un document, une ligne dans une base de données ou le résultat de l'exécution d'un algorithme. [15]

Une ressource RESTful est tout ce qui peut être traité sur le Web. Adressable fait référence aux ressources accessibles et transportables entre le client et le serveur. Ensuite, les ressources sont des mappages logiques et temporels aux concepts du domaine du problème pour lequel nous mettons en œuvre des solutions. [16]

h) URL

L'URL (Uniform Resource Locator) est la forme la plus courante d'identificateur de ressource. Les URL décrivent l'emplacement spécifique d'une ressource sur un serveur particulier. Ils vous disent exactement comment récupérer une ressource à partir d'un emplacement précis et fixe (figure 12). [14]

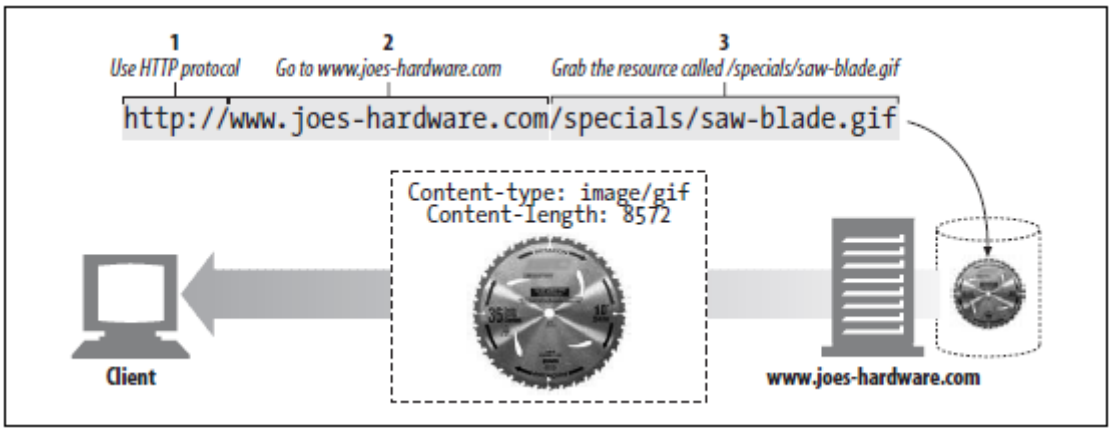


FIGURE 12 : LA PRÉSENTATION D'UNE RESSOURCE URL [14]

i) URI

Chaque ressource de serveur Web a un nom, de sorte que les clients peuvent indiquer les ressources qu'ils les intéressent. Le nom de la ressource serveur est appelé identificateur de ressource uniforme, ou URI. Les URI sont comme les adresses postales d'Internet, identifiant de manière unique et localisant les ressources d'information dans le monde entier (figure 13). [14]

Un URI est une chaîne de caractères utilisée pour identifier une ressource sur le Web. En termes simples, l'URI d'un service Web RESTful est un lien hypertexte vers une ressource, et c'est le seul moyen pour les clients et les serveurs d'échanger des représentations. [16]

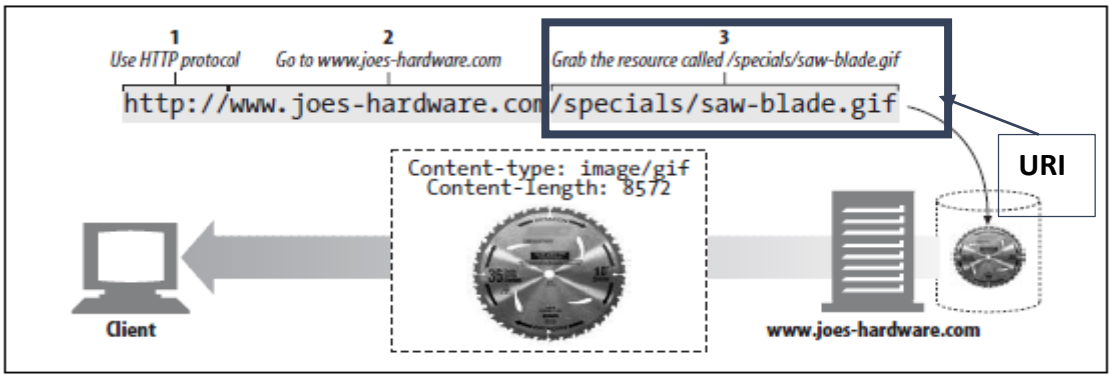


FIGURE 13 : LA PRÉSENTATION D'UNE RESSOURCE URI [14]

REST demande aux développeurs d'utiliser des méthodes HTTP explicitement et d'une manière conforme à la définition du protocole. Chaque ressource est identifiée par une URL. Chaque ressource doit prendre en charge les opérations communes HTTP, et REST permet à cette ressource d'avoir différentes représentations, par exemple, texte, xml, JSON, etc. [5]

j) Les méthodes REST HTTP

Contrairement à SOAP, REST ne nécessite pas vraiment un nouveau format de message. Il utilise les basses de http CRUD (Create, Read, Update et Delete), nous pouvons facilement mapper nos actions CRUD sur les ressources aux méthodes HTTP appropriées telles que POST, GET, PUT et DELETE. Ceci est indiqué dans le tableau (tableau 1) suivant :

Action de donnée	HTTP méthode
CREATE	POST or PUT
READ	GET
UPDATE	PUT or PATCH
DELETE	DELETE

TABLEAU 1: LES MÉTHODES CRUD EN HTTTP [16]

1.2.3 Comparaison entre web service SOAP et REST

Nous notons des points de différence entre SOAP et REST dans le Tableau 2 et la figure 14 :

SOAP	RESTFUL
Un protocole de message basé sur XML	Un protocole de style architecture
Il ne peut pas utiliser le protocole REST	Il peut utiliser le protocole SOAP avec HTTP

Utilise WSDL pour la communication entre client et le serveur	Utilise XML et JSON (javascript Object Notation) pour envoyer et recevoir les donnés
Le transfert se fait sur HTTP mais pas nécessairement, il peut utiliser autre protocole comme FTP, SFTP, ect	Le transfert se fait uniquement via HTTP
SOAP est moins performant que REST	La performance est beaucoup mieux par rapport à SOAP - moins de CPU intensif, plus de code
SOAP définisse leur propre sécurité	REST hérite des mesures de sécurité de transport sous-jacent
SOAP nécessite plus de bande passante et de ressources	REST nécessite moins de bande passante et de ressources.
SOAP autorise uniquement le format XML	REST autorise plusieurs formats : XML, jSON, HTML, ect

TABLEAU 2 : COMPARAISON ENTRE SOAP ET REST [5]

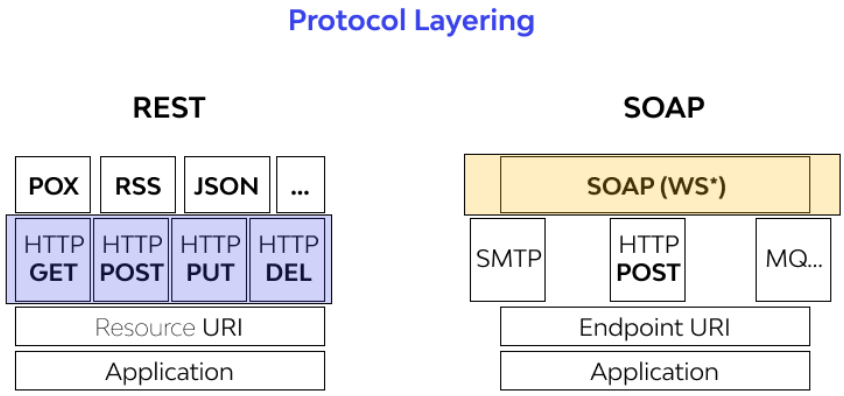


FIGURE 14 : SOAP VS REST [39]

Afin que la communication puisse s'effectuer sur diverses plateformes, dans notre cas client/serveur, nous avons choisi le web service REST. Car les comparaisons entre REST et SOAP [9-25] ont montré que les services de style architecture REST sont plus adaptés à des d'environnement IOT en termes d'évolutivité et de performances élevées.

- **Comparaison entre XML et JSON**

L'utilisation du protocole REST augmente chaque année grâce à ses caractéristiques, parmi ces caractéristiques, nous citons le langage de communication JSON. La comparaison entre les deux langages de communication XML et JSON montré dans le tableau suivant d'où nous avons choisi le langage JSON (tableau3) :

Propriété	XML	JSON
Simplicité	XML simple et facile à lire	JSON est beaucoup plus simple que XML

Traitement	Facile à utiliser	JSON est plus facile à gérer Parce que sa structure est plus simple.
Performance	Non optimisé pour les performances en raison des balises	Plus rapide que XML en raison de la taille
Ouverture	XML est ouvert	JSON est ouvert que XML, peut-être plus parce qu'il n'est pas au centre d'une lutte de normalisation corporative / politique
Orienté objet	XML est orienté document	JSON est orienté données. JSON peut être mappé plus facilement aux systèmes orientés objet
Extensibilité	XML est extensible	JSON n'est pas extensible. Il n'est pas un Document Markup Language, donc pas besoin de définir de nouvelles balises ou propriétés pour représenter les données qu'il contient

TABLEAU 3 : COMPARAISON ENTRE XML ET JSON [5]

1.3 Service web et l'Internet des objets

L'IoT ne sera pas considéré comme des systèmes individuels, mais comme une infrastructure critique et intégrée sur laquelle de nombreuses applications et services peuvent fonctionner. Certaines applications seront personnalisées, comme la numérisation des activités de la vie quotidienne, d'autres seront à l'échelle d'une ville, comme des transports efficaces et sans retard, et d'autres encore seront mondiales, comme des systèmes de livraison mondiaux. [24]

Comme des billions d'objets sont connectées à l'internet, il est nécessaire de disposer d'une architecture adéquate qui permet une connectivité, un contrôle et des communications aisés, ainsi que des applications utiles. Comment ces objets interagiront-ils dans et entre les applications ?

Dans l'Internet des objets, des centaines de protocoles incompatibles coexistent aujourd'hui. Cela rend l'intégration des données et des services de divers appareils extrêmement complexe et coûteuse. Dans le Web des objets, n'importe quel appareil est accessible à l'aide de protocoles Web standard. La connexion d'appareils hétérogènes au Web simplifie considérablement l'intégration entre les systèmes et les applications.

Ne serait-il pas merveilleux si n'importe quel appareil pouvait être facilement intégré et consommé par n'importe quelle application, quels que soient les protocoles ou les normes de réseau qu'ils utilisent ? C'est exactement ce que permet le Web des objets, comme l'illustre la figure 15.

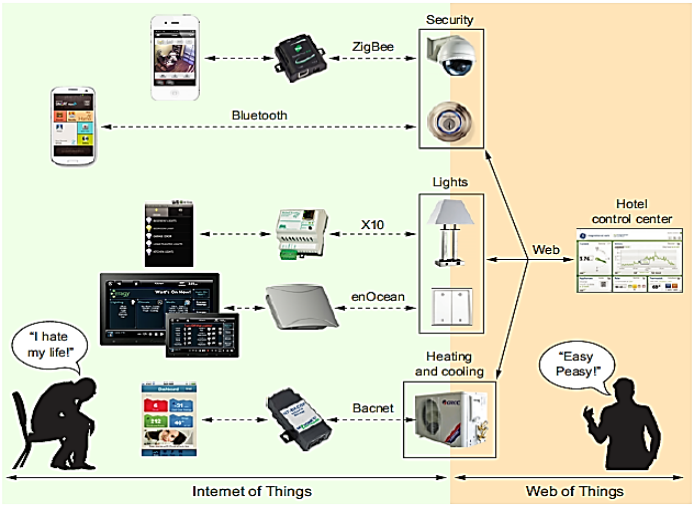


FIGURE 15 : IOT ET WOT [27]

1.3.1 Le web des objets

Ces dernières années, les progrès des technologies dans le matériel et le réseau créent un ensemble de conditions positives pour permettre à la vision de devenir enfin une réalité. Tout d'abord, avec la diminution des coûts de production du matériel, le volume d'appareils dotés de capacités réseau augmente considérablement et devient une partie intégrante de la plupart des vies quotidiennes. En d'autres points, les technologies de communication qui permettent de connecter des objets à Internet se développent, du GPRS au LTE, du WiFi au Bluetooth, ZigBee, etc. Ces technologies augmentent la vitesse de connexion, varient en bande passante, fournissent diverses échelles de communication et réduisent la consommation d'énergies. Les technologies permettent de connecter de nouveaux objets, ou des objets réformés, à l'Internet existant, formant ainsi « l'Internet des objets ». [26]

D'autre part, les technologies liées à Internet permettent de connecter divers appareils électroniques. Selon ce concept, les objets du monde réel qui sont des appareils grand public peuvent être mappés sur des ressources fournies par le Web, formant ce que l'on appelle « l'Internet des objets ». [26]

Dans le Web des objets, les appareils et leurs services sont entièrement intégrés au Web car ils utilisent les mêmes normes et techniques que les sites Web traditionnels. Cela signifie que nous pouvons écrire des applications qui interagissent avec les appareils embarqués exactement de la même manière que nous interagissons avec tout autre service Web utilisant des API Web, en particulier les architectures RESTful.

Donc le Web of things ou le Web de choses (WoT) cherche à répondre à la fragmentation de l'IoT en utilisant et en étendant des technologies de Web existantes, standardisées, en fournissant des métadonnées standardisées et d'autres unités élémentaires de structure technologiques réutilisables. [16]

1.3.2 Le rôle de Web des objets

FIGURE 16 : LE WEB DES OBJETS ET LES NORMES WEB MODERNES [27]

Comme le montre la figure 16, le Web des objets est la possibilité d'utiliser les normes Web modernes sur des appareils embarqués. En utilisant toutes ces normes pour les scénarios de l'Internet des objets, nous permettons à la fois de créer de nouveaux types d'applications interactives et nous nous assurons que les appareils peuvent être intégrés aux applications et services Web modernes avec un minimum d'effort.

Le web des objets permet aussi aux objets connectés de fournir et de partager des informations sur eux même, et de permettre la supervision des objets à distance. De plus, le web des objets fourni des meilleures solutions à la demande d'utilisateur. [26]

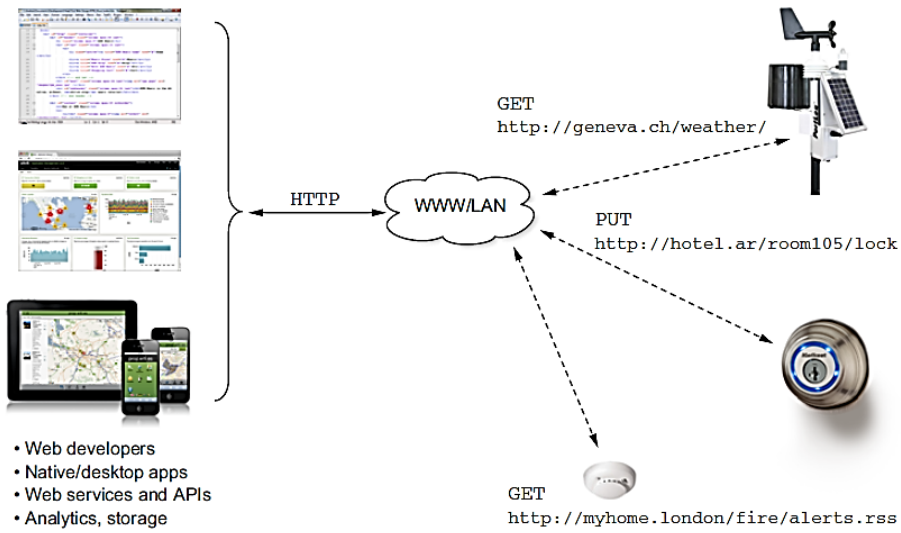


FIGURE 17: LES APPLICATIONS ÉCHANGENT DES DONNÉES AVEC OBJETS PHYSIQUES [27]

Comme le montre la figure 17, le Web des objets permet aux développeurs et aux applications d'échanger des données avec n'importe quel objet ou appareil physique à l'aide de requêtes HTTP standard, quel que soit le mode de connexion de l'appareil en utilisant des commandes « GET », « POST », « PUT », « DELETE ».

1.3.3 Le fonctionnement du Web des objets

Les données doivent être collectées à partir des capteurs en utilisant différents types de capteurs. Ces données doivent être converties en un format de données universellement accepté pour leur traitement sur le Web. Il doit être emballé avec des destinations appropriées, puis placé sur le Web. Toutes les données doivent être dans le cloud pour leur accès immédiat. Il doit y avoir des programmes pour accéder facilement à ces données et pour répondre au client. Si le programme est dans le cloud, nous pouvons avoir l'accessibilité chronométrée. Après le traitement, le résultat sera envoyé à l'utilisateur (figure 18). [28]

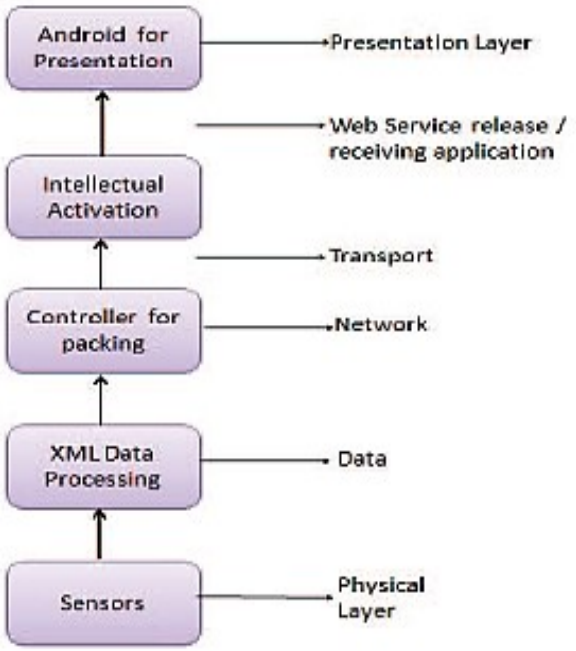


FIGURE 18: LE FONCTIONNEMENT DE WEB DES OBJETS [28]

Il existe deux grandes catégories d'objets physiques sur le Web des objets : objets balisés et objets connectés.

La première catégorie comprend diverses technologies d'étiquetage attachées à un produit, telles que les codes à barres, les codes QR, les étiquettes NFC ou RFID, etc. Dans ce cas, les objets ne sont pas connectés directement au Web, mais uniquement passivement, car un autre appareil ou une autre application est nécessaire pour interagir avec le produit. Les objets connectés sont directement connectés au Web des objets et sont le monde des systèmes

embarqués[3] et des appareils embarqués, qui sont essentiellement de petits ordinateurs relativement peu coûteux, de faible puissance avec des ressources et des capacités limitées.

Littéralement des milliers de types de plates-formes embarquées sont disponibles, allant de petites séries de production de nœuds de capteurs à usage général construits pour les chercheurs ou les pirates informatiques à des circuits bon marché et produits en série spécialement pour les détecteurs de fumée, les fours à micro-ondes et les réveils. Ce qu'il faut retenir, c'est qu'il existe deux grandes ligues d'appareils embarqués : ceux destinés aux amateurs (moins spécifiques et optimisés mais plus réutilisables et flexibles) comme Arduino, Raspberry et ceux destinés à être intégrés dans des produits industriels du monde réel (plus optimisés pour des cas d'utilisation spécifiques, donc plus difficiles à étendre et à utiliser dans d'autres contextes).

1.3.4 Le domaine de Web des objets

Le Web des objets consiste essentiellement dans le développement de concepts, d'outils et de systèmes pour la création et l'exploitation de réseaux d'objets associés à des ressources embarquées (puces RFID, capteurs et actionneurs, installations informatiques complexes) accessibles par des services web [30] (figure 19) :

- Le Web social : partage des objets, des données ou des fonctionnalités vers une utilisation participative et collaborative.
- Le Web physique : applications de géolocalisation.
- Le Web sémantique : en-tête de métadonnées analysées et indexées par des moteurs de recherche pour permettre à des agents logiciels de partager, de réutiliser ou de combiner ces informations.
- Le Web temps réel : informations en temps réel livrées en temps opportun.
- Le Web programmable : accès à des données brutes avec une interaction avec les objets physiques par le biais d'API ouvertes.

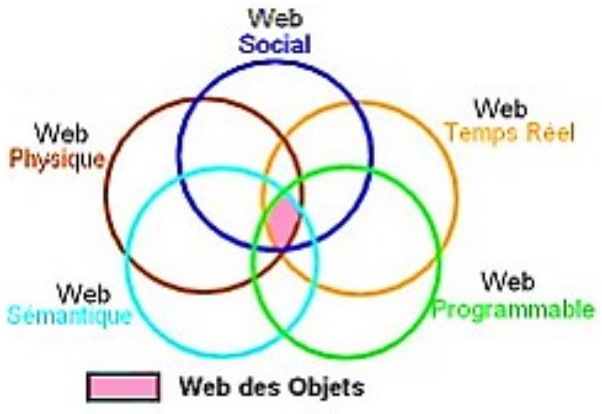


FIGURE 19 : LE DOMAINE DE WEB DES OBJETS [27]

[3] Les systèmes embarqués peuvent être définis comme des systèmes électroniques et informatiques autonomes qui sont de plus en plus utilisés pour contrôler des systèmes complexes, ils se trouvent dans les équipements médicaux, les guichets automatiques, etc. [29]

1.4 La domotique

L'évolution de l'habitat dans le temps montre que le souci de confort ou de sécurité, que ce soit dans l'antiquité, au moyen âge ou aux périodes suivantes, a toujours été présent. Ceux qui avaient les moyens faisaient « gérer » les fonctions, selon les époques, par des esclaves, du personnel libre, puis plus tard par des domestiques. La gestion et l'anticipation des besoins/désirs des occupants, que ce soit pour le chauffage, le rafraîchissement, la sécurité ou d'autres aspects, étaient ainsi assurées, mais pour une frange de la population seulement. La démocratisation de ce rêve, le XXe siècle l'a partiellement permise grâce aux progrès techniques, à commencer par l'électricité. La première étape a été l'introduction d'équipements électroménagers qui ont allégé les charges des utilisateurs (surtout des femmes). Ces équipements se sont sophistiqués et ont offert de plus en plus de possibilités grâce à l'électronique. D'autres fonctions de l'habitat ont été aussi couvertes (communication, multimédia, sécurité, éclairage ...). L'informatique a ensuite permis une évolution vers une automatisation de ces équipements et une interface utilisateur plus intuitive grâce en particulier à l'intégration des smartphones et tablettes dans l'environnement domestique. [31]

1.4.1 Les concepts de base

1.4.1.1 L'intelligence ambiante

L'intelligence ambiante tient aux capacités de mobilité et d'intégration des systèmes numériques dans le milieu physique au point de s'y confondre, et ceci de manière spontanée, à de multiples échelles, de la planète au micro-, voire nano-objet. Cette mobilité et cette fusion rendues possibles par la miniaturisation et la puissance des composants électroniques, par l'omniprésence des réseaux sans fil, et par la chute des coûts de production, permettent, à leur tour, d'entrevoir la composition opportuniste de dispositifs et de services de toutes sortes au-dessus d'une infrastructure à granularité et géométrie variables, dotés de facultés de capture, d'action, de traitement, de communication et d'interaction. La finalité de l'intelligence ambiante est l'amélioration, Autrement dit, il s'agit de créer des services et des dispositifs intelligents capables de répondre à des besoins individuels, collectifs et sociétaux. [13]

L'un de ces caractéristiques nous disons l'ubiquité. Elle est la caractéristique d'un système qui permet à l'utilisateur une interaction à partir de n'importe quel point de l'environnement qu'il contrôle. Les progrès en traitement de l'Information qui ont permis le développement de cette discipline concernent surtout les réseaux de capteurs, les environnements perceptifs et l'intelligence artificielle. La représentation de connaissances, un problème très abordé dans l'intelligence artificielle, présente une importance majeure pour l'intelligence ambiante (figure 20). [33]

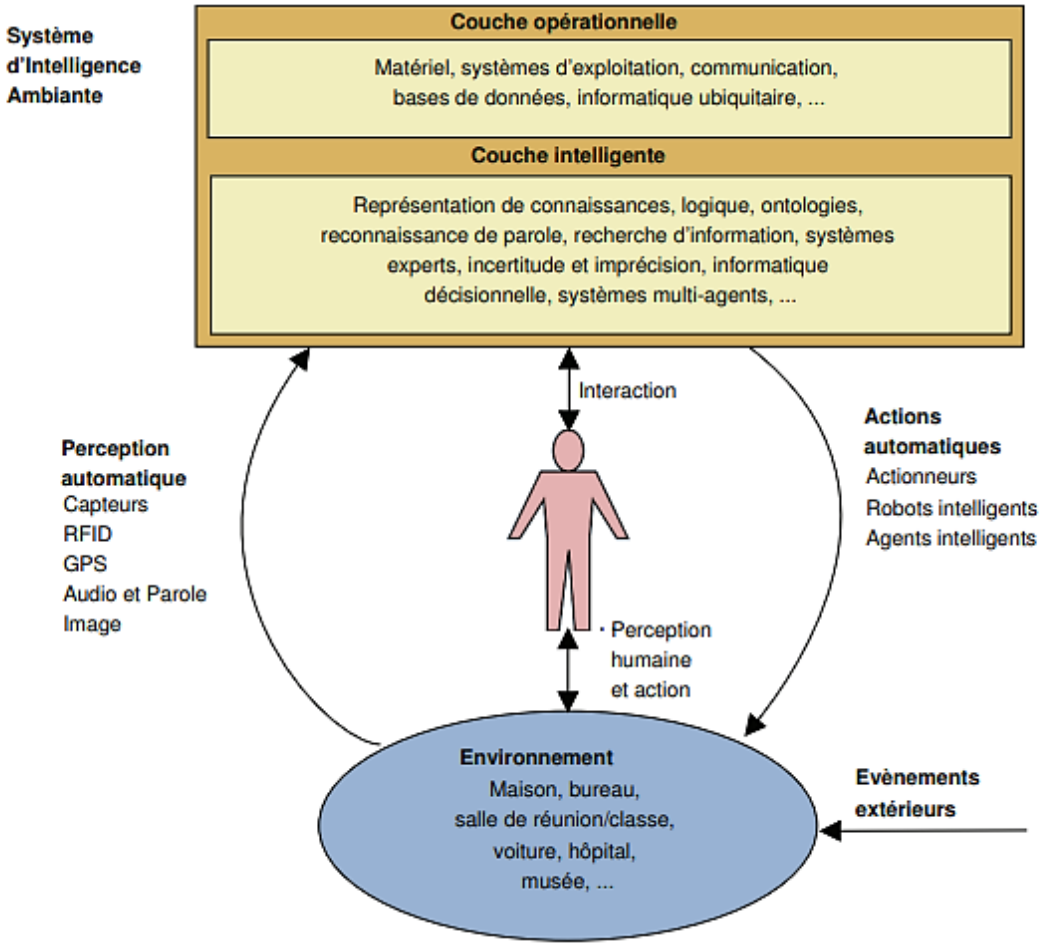


FIGURE 20 : RELATION ENTRE L'INTELLIGENCE AMBIANTE ET L'INTELLIGENCE ARTIFICIELLE [33]

1.4.1.2 Un environnement intelligent

Un environnement intelligent consiste en l'implémentation d'un espace physique dynamique et adaptable qui optimise les services aux utilisateurs en utilisant des systèmes sensibles au contexte et des technologies ubiquitaires. L'accès ubiquitaire est rendu possible par la présence de multiples systèmes hétérogènes connectés par un réseau et incorporés dans le milieu (meubles, vêtements, murs) et qui sont capables de détecter l'apparition de phénomènes physiques et éventuellement de les caractériser. [33]

Le terme domotique vient du latin « *dom us* » (maison) référence à une technologie qui englobe les équipements en permettant d'automatiser et/ou de gérer certaines fonctions domestiques. Le terme a été créé en France milieu des années 1980. [31]

La domotique regroupe l'ensemble des technologies permettant l'automatisation des équipements d'un habitat. Ce terme vient du mot latin domus qui désigne une demeure patricienne. La domotique vise à apporter des fonctions de confort (commandes à distance, gestion d'énergie, optimisation de l'éclairage et du chauffage, etc.), de sécurité (alarme) et de communication (contacts et discussion avec des personnes extérieures) [33]

1.4.1.3 Smart home ou la maison intelligente

Le smart home qui englobe une approche globale de l'habitat et des fonctions associées peut se différencier par le type de technologies domestiques installées ou leur niveau d'interaction. [31]

Une maison intelligente est une combinaison de plusieurs technologies avancées. Elle consiste à incorporer de très petites puces, qui possèdent des capacités de communication sans-fil, de perception et de traitement de l'information, dans les articles à usage quotidien. Le but étant de créer un environnement informatique transparent pour l'habitant. Pour être en mesure de fournir des services, le système doit être capable d'acquérir, de traiter, et de transmettre l'information à tout moment. Il doit également être capable de comprendre les besoins de l'usager et de contrôler les différents équipements de façon intelligente, afin de rendre l'environnement plus confortable. De plus il doit permettre de réduire la consommation d'énergie sans influencer les habitudes de l'habitant. [32]

Smart home est l'aboutissement du concept d'habitat au service des occupants, car il utilise les possibilités offertes par les technologies domestiques (qui proposent différentes solutions) (figure 21). Un équipement domestique assure généralement une fonction spécifique comme l'éclairage, la fermeture des volets roulants ou la régulation de température. Il peut cependant être relié à d'autres éléments, comme par exemple un capteur de présence, de luminosité ou de température ou une station météo, pour interagir et étendre les possibilités de l'équipement. [31]

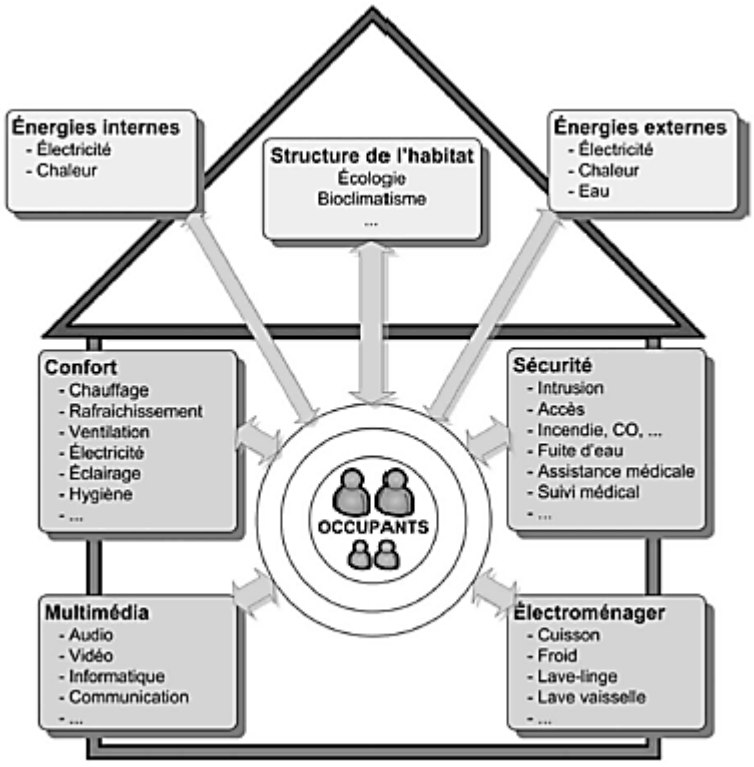


FIGURE 21 : FONCTIONS COUVERTES PAR LE SMART HOME [31]

1.4.1.4 Caractéristiques de base du smart home

La maison intelligente est caractérisée par : [31]

- Connectivité interne : un réseau interne à l'habitat avec la possibilité de contrôler toutes les fonctions à partir de boutons poussoirs, d'une télécommande et éventuellement d'un smartphone ou d'une tablette.
- Utilisation d'un même langage de communication (protocole) pour tous les composants.
- Couverture de tous les services ou fonctions de base par le même système, c'est-à-dire sans passer par une autre interface.
- Compatibilité descendante, où les nouveaux composants ou les nouvelles versions des logiciels restent compatibles avec les équipements installés.
- Extension des fonctions ou des services sans modification du système.
- Connectivité externe comme option (passerelle vers Wi-Fi, Internet ou autres protocoles).

1.4.1.5 Exemple d'objets connecté dans la maison intelligent

- Thermostat, la plus célèbre NEST pour régler la température de toute la maison (figure 22)

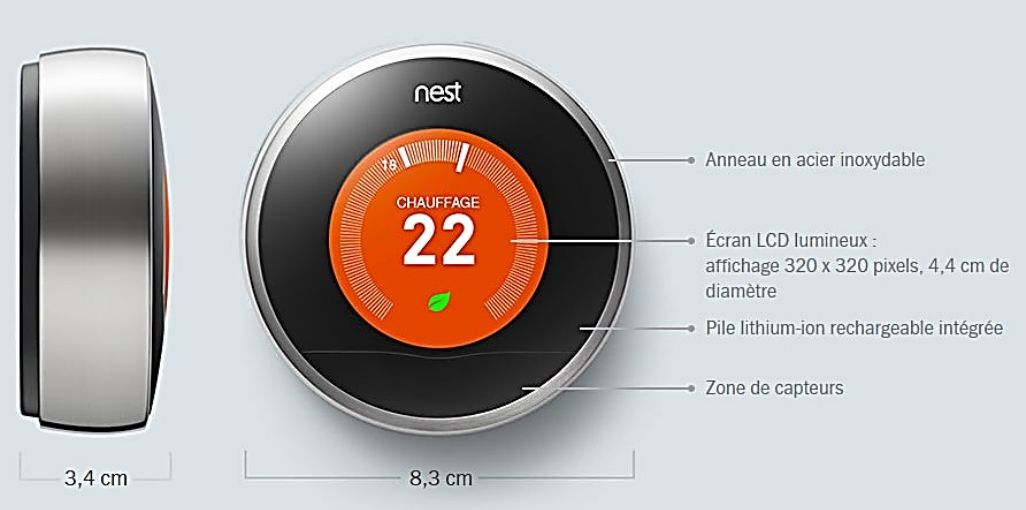


FIGURE 22 : THERMOSTAT NEST

- Une prise esthétique pour piloter l'appareil branché et de voir sa consommation (figure 23)

FIGURE 23 : WALL PLUG PRISE

- Des lampes réduisent la consommation d'énergie avec un régulateur de couleur (figure 24)

FIGURE 24 : LAMPE INTELLIGENTE DE PHILIPS

- Contrôle de toute la maison par un seul objet qui fait le rôle d'humain dans le smart home (piloter le chauffage, l'éclairage, ouverture des portes et des fenêtres, etc.) (Figure 25)

FIGURE 25 : LE BOX DOMOTIQUE TAHOMA

- Le détecteur de fuite d'eau qui détecte les petites fuites d'eau et leurs dommages et il peut aussi couper l'eau s'il y a des risques dans les tuyaux de conduite (figure 26)

FIGURE 26 : LE DÉTECTEUR DE FUITES

Conclusion

Ce chapitre a été découpé en quatre parties, la première partie a été consacrée à la présentation de l'internet des objets, ses domaines d'application ainsi que ses exigences. Dans la deuxième partie, nous avons mis l'accent sur quelques concepts liés au web service comme JSON, API, ect. Nous avons fait une comparaison entre les deux architectures du web service. La troisième partie était consacré au rôle du web dans l’internet des objets avec des exemples sur la collaboration via les web service. Le chapitre a été clôturé par les définitions des termes liés au domotique avec quelques exemples sur les nouveaux objets domestiques. Le chapitre suivant sera consacré à faire une conception globale du système.

Chapitre 2 : Conception

Chapitre 2 : Conception

2.1 Introduction

Le but de ce mémoire est de développer un système permettant la collaboration d'éléments de l'IoT afin d'améliorer l'échange des données facilitant la réalisation des services. Cette collaboration se concrétise dans une tâche individuelle entre deux objets ou dans une tâche commune entre plusieurs objets dans le but de produire un service quelconque. En effet, Les architectes logiciels migrent aujourd'hui peu à peu vers les architectures centrées service. Les applications sont maintenant construites comme des compositions de services intégrant de plus en plus de fonctionnalités.

Ce chapitre consiste à faire une modélisation et une conception de l'application qui permet de faire la collaboration entre les objets. Étant donné notre domaine d'application est la domotique, Où, nous allons faire une conception qui permet aux objets connectés de collaborer via un web service afin de synchroniser et échanger les données.

2.2 Présentation du système domotique

2.2.1 Le défi du système domotique

Plusieurs constructeurs dans le monde ont proposé plusieurs solutions dans la domotique que ce soit dans le domaine de la sécurité, l'électroménager, le refroidissement et le réchauffage. Chacun de ces objets est contrôlé par sa propre application de contrôle via une passerelle et un protocole de communication prédéfinie par le constructeur.
Si un jour nous voulons développer une application qui permet de collaborer ces objets, nous devrons faire face au divers défis tel que l'hétérogénéité des protocoles de communication, l'hétérogénéité des formats de données. La solution que nous proposons est d'utiliser les normes du web plus précisément les technologies du web service de type REST pour échanger les données avec l'application toute en gardant les protocoles de communications prédéfinies par le constructeur.
Nous considérons le web service REST comme une couche d'abstraction pour la couche hardware de l'objet, il sera ainsi facile d'interagir avec les objets de la même manière qu'avec un web service et de créer des solutions de collaboration entre les objets hétérogènes. (Figure 27)

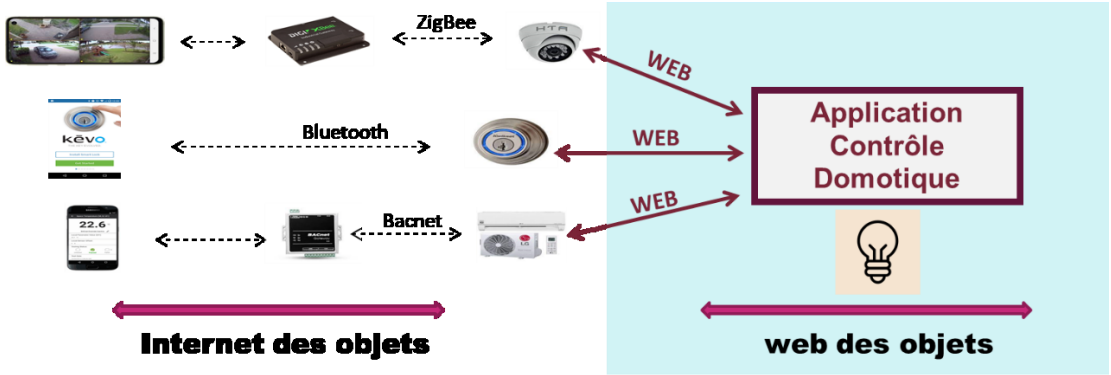


FIGURE 27 : SYSTÈME DOMOTIQUE

2.2.2 Architecture générale du système

L'architecture du système est représentée par la figure 28.

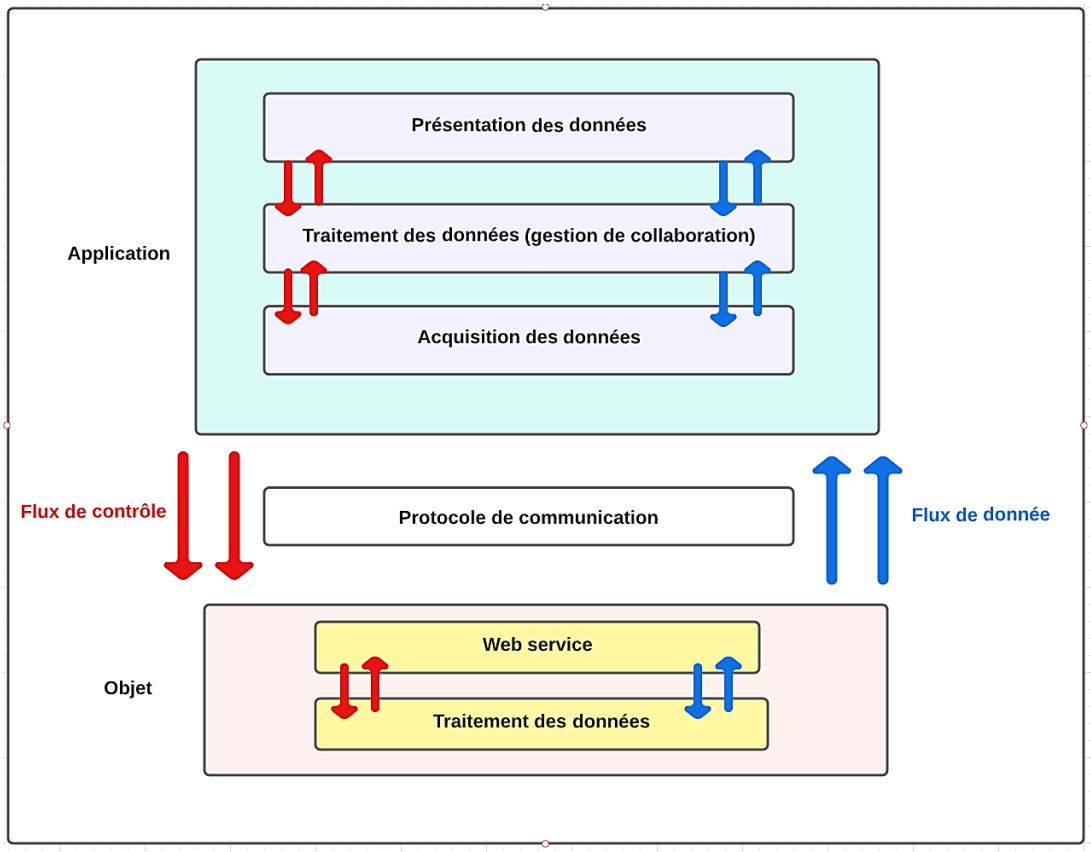


FIGURE 28 : ARCHITECTURE GÉNÉRALE DU SYSTÈME

NOTE : l'objet représente un objet connecté qui peut être soit un actionneur ou un capteur ou bien les deux.

- Application : l'application est modélisée en suivant le pattern MVVM (Model, View, ViewModel), pour une meilleure séparation entre le traitement logique des données et la couche présentation des données et les données, offrant ainsi un moyen facile pour l'extensibilité, la maintenance ou des éventuels changements de l'application.
- Protocole de communication : assurer la communication entre l'objet et l'application en utilisant les standards du web à savoir les différentes technologies et protocoles utilisés sur le Web et en particulier ceux définis par le W3C tel que AJAX, HTTP, web socket.
- Objet : l'objet connecté contient des informations mais ne gèrent pas le comportement et n'influence pas la façon dont les informations apparaissent à l'utilisateur final.

2.3 Model-View-ViewModel (MVVM)

Le modèle MVVM (Model-View-ViewModel) est utilisé pour assurer une séparation nette des préoccupations entre l'interface contrôle utilisateur (user interface controls) et leur logique sous-jacente. Le patron MVVM se compose de trois éléments principaux : le modèle, la vue et

le modèle de vue, chacun d'entre eux jouant un rôle distinct et séparé, comme le montre le diagramme ci-dessous Figure 29.

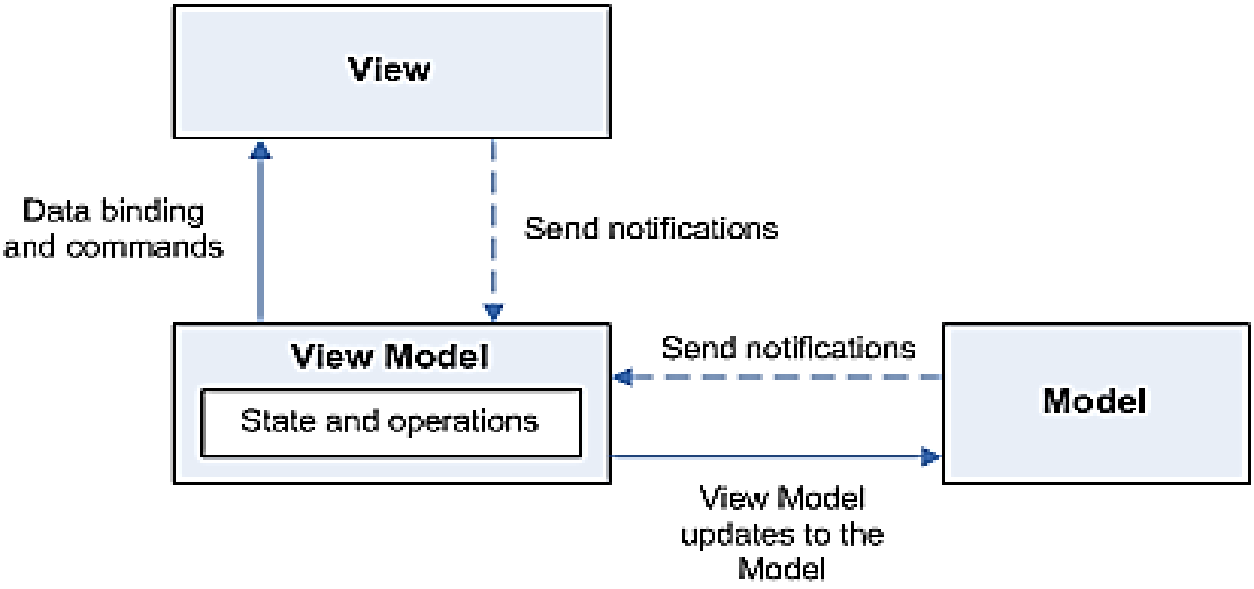


FIGURE 29 : MVVM

2.3.1 Le modèle (Model)

Le modèle est une représentation du modèle de domaine de l'application. Les modèles contiennent des informations mais ne gèrent pas le comportement et n'influencent pas la façon dont les informations apparaissent à l'utilisateur final.

2.3.2 La vue (View)

La vue contrôle la structure, la disposition et l'apparence de ce que l'utilisateur final voit. La vue est responsable de la représentation de l'état du modèle de vue. La vue est construite à partir de documents HTML qui contiennent des liaisons déclaratives qui la relient au modèle de vue.

2.3.3 Le modèle de vue (View Model)

Le modèle de vue sert d'intermédiaire entre la vue et le modèle, Il est responsable de la gestion de la logique de la vue. Le modèle de vue récupère les données du modèle et les met à la disposition de la vue par le biais de liaisons de données. Dans le cadre de ce processus, le modèle de vue peut reformater les données de manière à faciliter leur utilisation par la vue. Le modèle de vue est également responsable de la transmission des commandes de la vue au modèle, agissant essentiellement comme un intermédiaire entre les deux.

2.3.4 Architecture utilisée

La figure 30 est un patron de conception pour développer une interface utilisateur pour notre application. Ce dernier est composé de Modèle (Model) où l'application sauvegarde les données indépendamment de l'interface graphique. Nous allons souvent utiliser des requêtes vers le serveur (plus précisément le web service), dans notre cas ça sera l'objet pour lire et écrire des données. La réponse à notre requête va être sous un format de données textuelles.

Comme le montre la figure 30, le model est séparé du View et du ViewModel, car le model est un programme serveur.

Le ViewModel est une représentation pure de la logique de données et des opérations sur l'interface graphique, il faut noter que le ViewModel n'est pas l'interface graphique utilisateur, il ne contient pas de concept de bouton ou d'affichage graphique, il gère les données non sauvegardées avec lesquelles l'utilisateur travaille.

L'objectif principal du ViewModel est de lire (GET) les données du serveur (Model) puis appliquer des opérations sur ces données dans le but de les appliquer sur notre interface graphique à l'aide de la liaison de données à fin que la View puise afficher ces derniers.

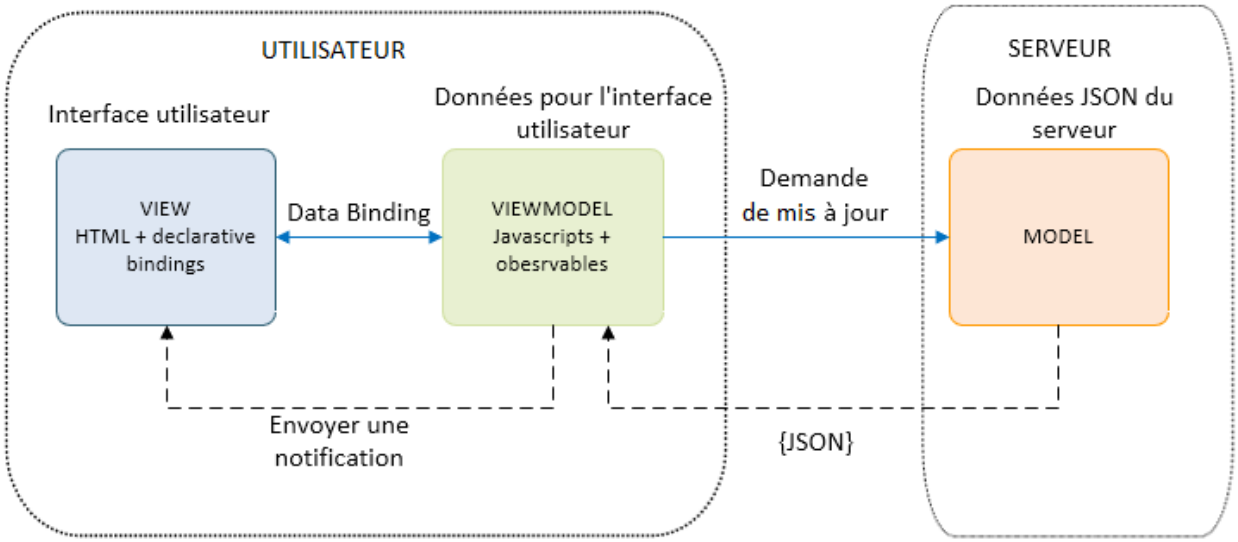


FIGURE 30 : ARCHITECTURE MVVM UTILISÉ

2.3 La vue statique du système

2.3.1 Digramme de cas d'utilisation

- **Identification des acteurs :**
  - Utilisateur : son rôle est de consulter les données et de créer des règles de collaboration entre les objets.
  - Objet : permettre de recueillir des informations depuis le monde physique et de les transmettre en temps réel, et permettre au système d'agir sur le monde physique en modifiant son état.
  - SGBD : représente le gisement cible au les données vont être sauvegardées.
- **Description textuelle**
  - **Cas d'utilisation** : Acquisition des données

Cas d'utilisation	Acquisition des données
Acteurs principaux	Utilisateur
Acteurs secondaires	Objet connecté
Objectif	Après que les objets ayant récolté les données, ils vont chercher à les transmettre au système.
Prés conditions	▪ Les objets sont allumés et opérationnels. ▪ L'accès au réseau internet au local. ▪ Le serveur de l'application est en écoute.
Poste conditions	/
Scénario nominale	1. L'utilisateur allume l'objet connecté 2. L'objet connecté capte les données et les publie sur son service web. 3. L'application lit les données depuis le service web de l'objet
Scénario alternative	Si le service web de l'objet n'est pas disponible l'application affiche un message d'erreur.

TABLEAU 4 : CAS D'UTILISATION ACQUISITION DONNÉES

La figure 31 représente le diagramme de cas d'utilisation acquisition de données :

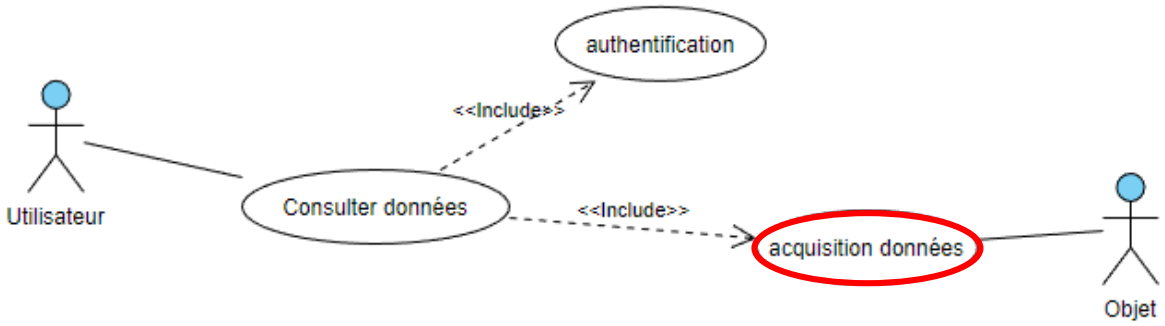


FIGURE 31 : CAS D'UTILISATION ACQUISITION DONNÉES

- **Cas d'utilisation** : Consulter données.

Cas d'utilisation	Consulter les données
Acteurs principaux	Utilisateur
Objectif	Utilisateur peut utiliser l'application web pour consulter les données suivantes : ▪ État de capteurs ▪ État des actionneurs ▪ Configuration des actionneurs ▪ Les données générer depuis les objets en temps réel
Prés conditions	▪ L'objet est allumé et opérationnel ▪ Connexion wifi établie avec le Gateway ▪ Authentification de l'utilisateur
Poste conditions	/
Scénario nominale	▪ L'utilisateur lance l'application ▪ L'utilisateur s'authentifie au système ▪ L'utilisateur consulte les données.
Scénario alternative	Si l'authentification n'est pas valide : ▪ L'application affiche les informations incorrectes.

TABLEAU 5 : CAS D'UTILISATION CONSULTER DONNÉES

La figure 32 représente le diagramme de cas d'utilisation consulter données :

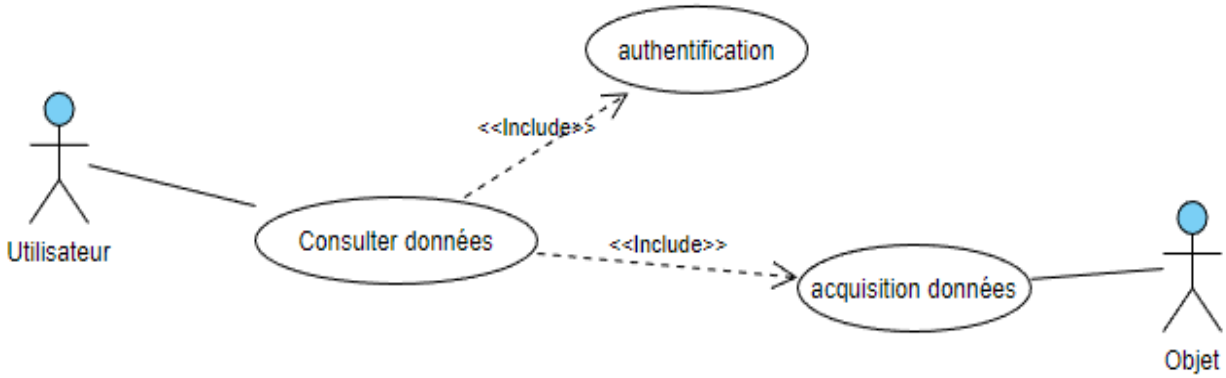


FIGURE 32 : CAS D'UTILISATION CONSULTER DONNÉES

- **Cas d'utilisation** : Gestion d'objet

Cas d'utilisation	Gestion d'objet
Acteurs principaux	Utilisateur
Objectif	L'utilisateur peut : - Ajouter objet - Modifier objet - Supprimer objet En précisant les fonctionnalités de l'objet et la localisation de l'objet
Prés conditions	▪ L'objet est allumé et opérationnel ▪ Connexion wifi établie avec le Gateway ▪ Authentification de l'utilisateur
Poste conditions	/
Scénario nominale	▪ L'utilisateur lance l'application ▪ L'utilisateur s'authentifie au système ▪ L'utilisateur Ajout l'objet connecté.
Scénario alternative	Si l'authentification n'est pas valide : ▪ L'application affiche les informations incorrectes.

TABLEAU 6 : CAS D'UTILISATION GESTION OBJET

La figure 33 représente le diagramme de cas d'utilisation gestion d'objet :

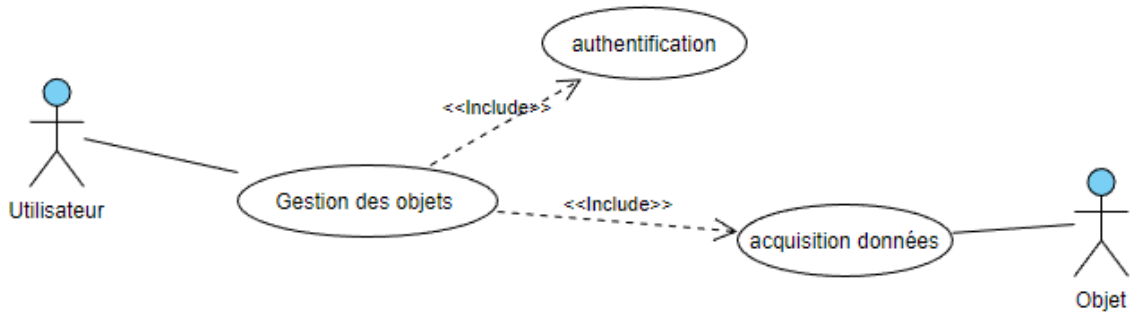


FIGURE 33 : CAS D'UTILISATION GESTION OBJETS

- **Cas d'utilisation** : Contrôler objet

Cas d'utilisation	Contrôler objet
Acteurs principaux	Utilisateur
Objectif	L'utilisateur peut contrôler l'objet manuellement selon les fonctionnalités de l'objet
Prés conditions	▪ L'objet est allumé et opérationnel ▪ Connexion wifi établie avec le Gateway(routeur) ▪ Authentification de l'utilisateur
Poste conditions	/
Scénario nominale	▪ L'utilisateur lance l'application ▪ L'utilisateur s'authentifie au système ▪ L'utilisateur sélection l'objet et envoi des requêtes de commande.
Scénario alternative	Si l'authentification n'est pas valide : ▪ L'application affiche les informations incorrectes. Si l'objet sélectionné n'est connecté ▪ L'application affiche un message d'erreur.

TABLEAU 7 : CAS D'UTILISATION CONTRÔLER OBJET

La figure 34 représente le diagramme de cas d'utilisation contrôler objet :

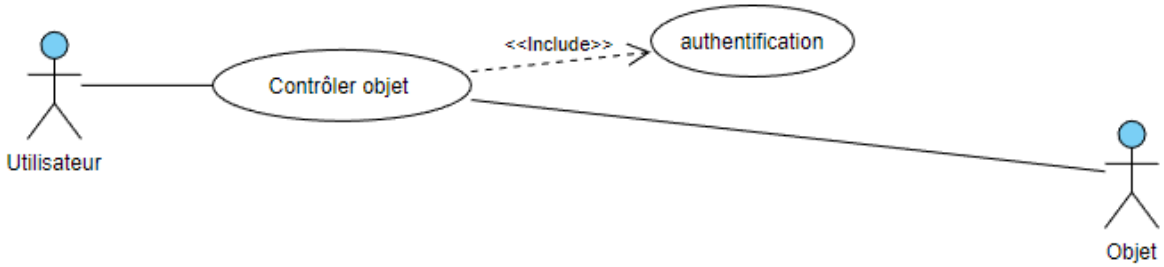


FIGURE 34 : CAS D'UTILISATION CONTRÔLER OBJET

- **Cas d'utilisation** : Gestion collaboration.

Cas d'utilisation	Gestion collaboration
Acteurs principaux	Utilisateur
Objectif	▪ Pour avoir une collaboration entre les objets connectés i faut analyser les données émis de façon continue par les objets connectés via un web service : ▪ L'utilisateur peut ajouter des scénarios de collaboration selon ces besoins. ▪ L'utilisateur peut modifier des scénarios de collaboration. ▪ L'utilisateur peut supprimer des scénarios de collaboration.
Prés conditions	▪ Les objets sont allumés et opérationnels. ▪ L'accès au réseau internet au local. ▪ Le serveur de l'application est en écoute. ▪ L'existence d'un compte dans l'application. ▪ Lancer l'application client. ▪ Authentification d'utilisateur.
Poste conditions	/
Scénario nominale	▪ L'utilisateur lance l'application. ▪ L'utilisateur s'authentifier au système. ▪ L'application afficher le menu. 1. L'utilisateur click sur l'onglet ajouter collaboration 2. L'application affiche le formulaire pour l'ajout d'une collaboration 3. L'utilisateur remplie le formulaire selon ces besoins.

TABLEAU 8 : CAS D'UTILISATION GESTION COLLABORATION

La figure 35 représente le diagramme de cas d'utilisation gestion collaboration :

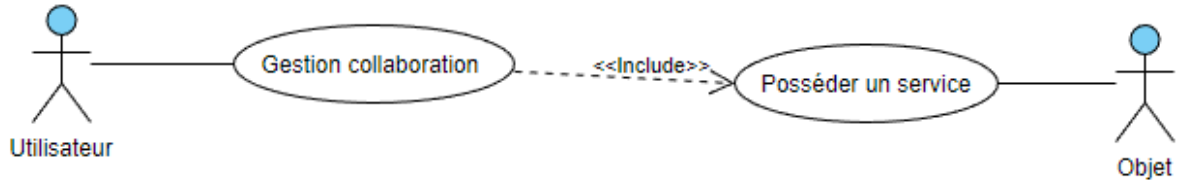


FIGURE 35 : CAS D'UTILISATION GESTION COLLABORATION

La figure 36 représente le diagramme de cas d'utilisation générale de l'application qui permet la collaboration dans l'internet des objets :

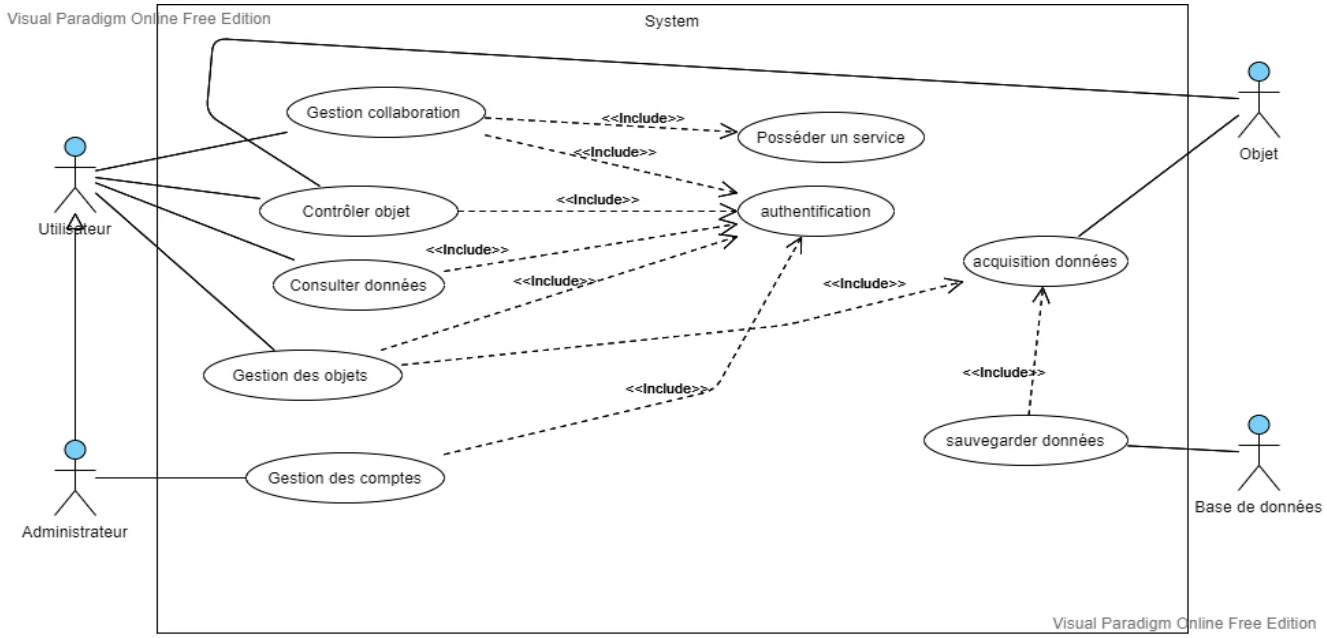


FIGURE 36 DIAGRAMME DE CAS D'UTILISATION GÉNÉRALE

La figure 37 représente le diagramme de cas d'utilisation détaillé du l'application :

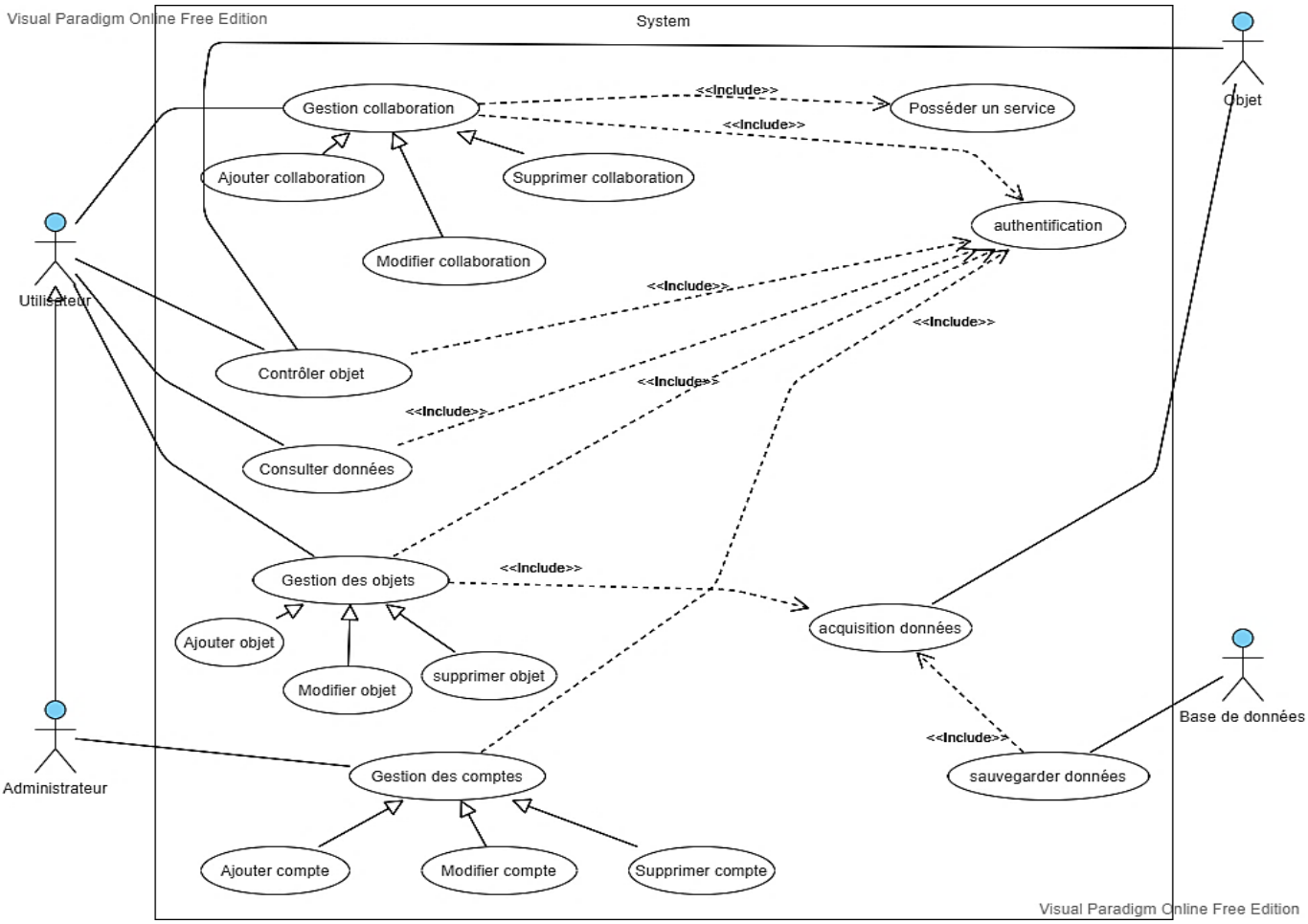


FIGURE 37 : DIAGRAMME DE CAS D'UTILISATION DÉTAILLÉ

2.3.2 Diagramme de classes

Dans ce type d'application le plus important c'est d'avoir une base de données représentant la zone où les objets vont collaborer, cette base de données doit contenir ce que l'objet connecté peut faire et comment l'utiliser peu importe la localisation de l'objet.
Par la suite, l'utilisateur peut créer des scénarios s'adaptant aux informations émises par l'objet en y ajoutant un comportement.

En utilisant le langage UML, nous pouvons représenter un objet par une classe comme le montre la figure 38 :

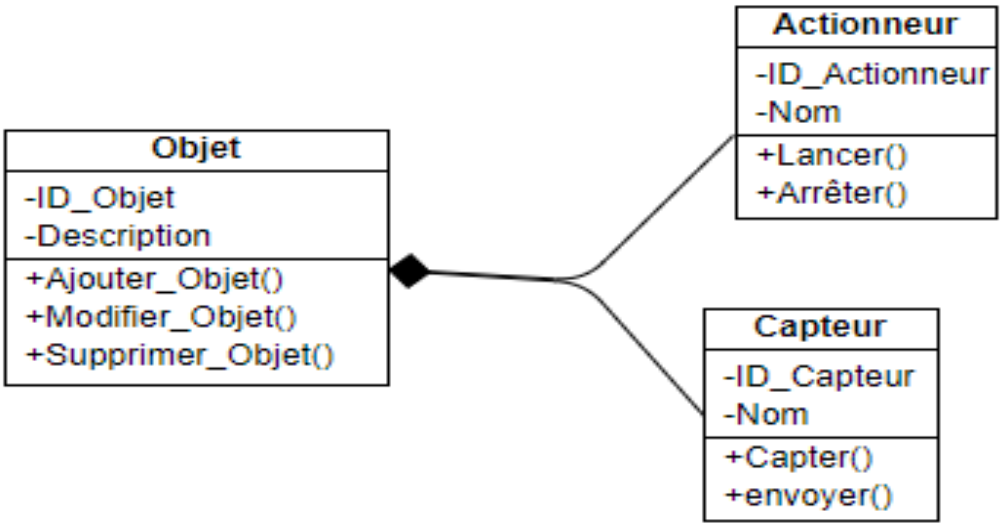


FIGURE 38 DIAGRAMME DE CLASSE REPRÉSENTANT UN OBJET CONNECTÉ

Ce diagramme de classe (figure 38) montre que l'objet connecté peut être composé d'actionneurs ou/et des capteurs.

- Capteur : permet de recueillir des informations depuis le monde physique en temps réel. Le capteur dispose de deux méthodes : capter () et envoyer ().
  - Capter () : une méthode qui permet de recueillir des informations du monde physique.
  - Envoyer () : permet d'envoyer des informations vers un gisement cible.
- Actionneur : permet au système d'agir sur le monde physique en modifiant son état. L'actionneur dispose de deux méthodes : arrêter (), lancer ().
  - Arrêter () : pour arrêter un comportement sur le monde physique.
  - Lancer () : pour lancer un comportement sur le monde physique.

Cette représentation est dédiée à un domaine d'application bien précis vu que nous avons précisé les composants de l'objets et ses fonctionnalités.

Dans le cas général, nous ne pouvons pas savoir quel type d'objet va être manipulé ni comment l'utiliser, car le domaine de l'IOT est très varié (le domaine de la santé diffère du domaine de fabrication). La solution a développé doit s'adapter à tous les domaines d'application, nous laisserons le choix à l'utilisateur finale d'ajouter l'objet en précisant les

composants et les fonctionnalités afin d'interagir avec l'objet et de changer le comportement du monde physique. Comme le montre la figure 39 :

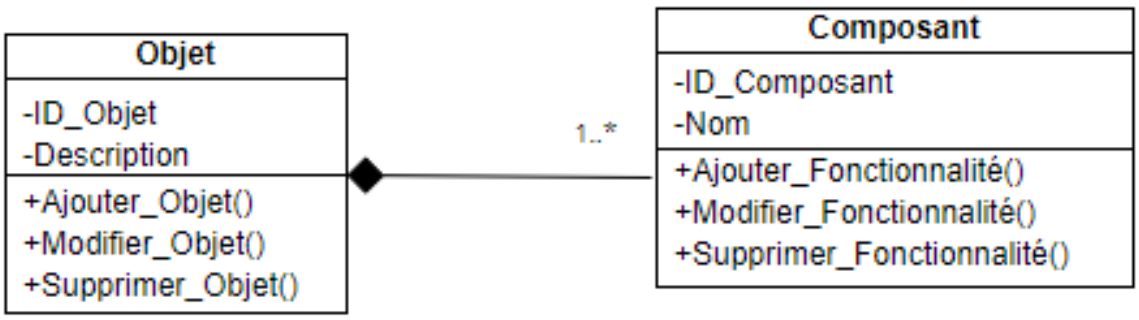


FIGURE 39 DIAGRAMME DE CLASSE OBJET CONNECTÉ

Chaque objet connecté offre un ou plusieurs services, un service peut être composé d'un ou plusieurs services comme le montre la figure 40 :

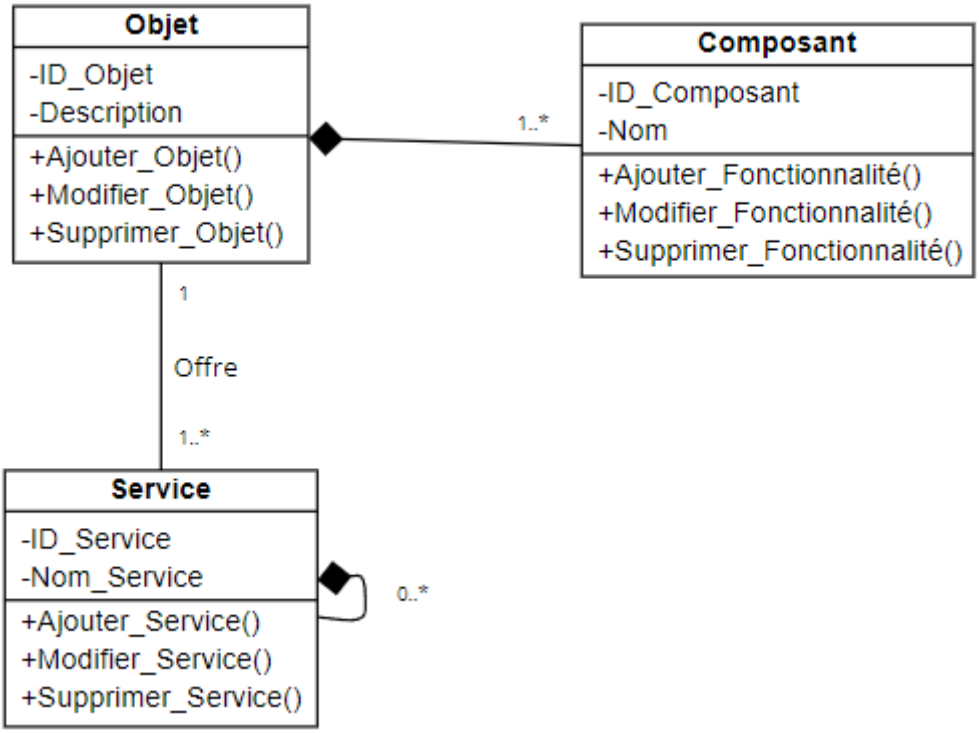


FIGURE 40 DIAGRAMME DE CLASSE OBJET CONNECTÉ ET SERVICE

Les objets connectés vont générer des données sur l'environnement de différents types comme le montre la figure 41 :

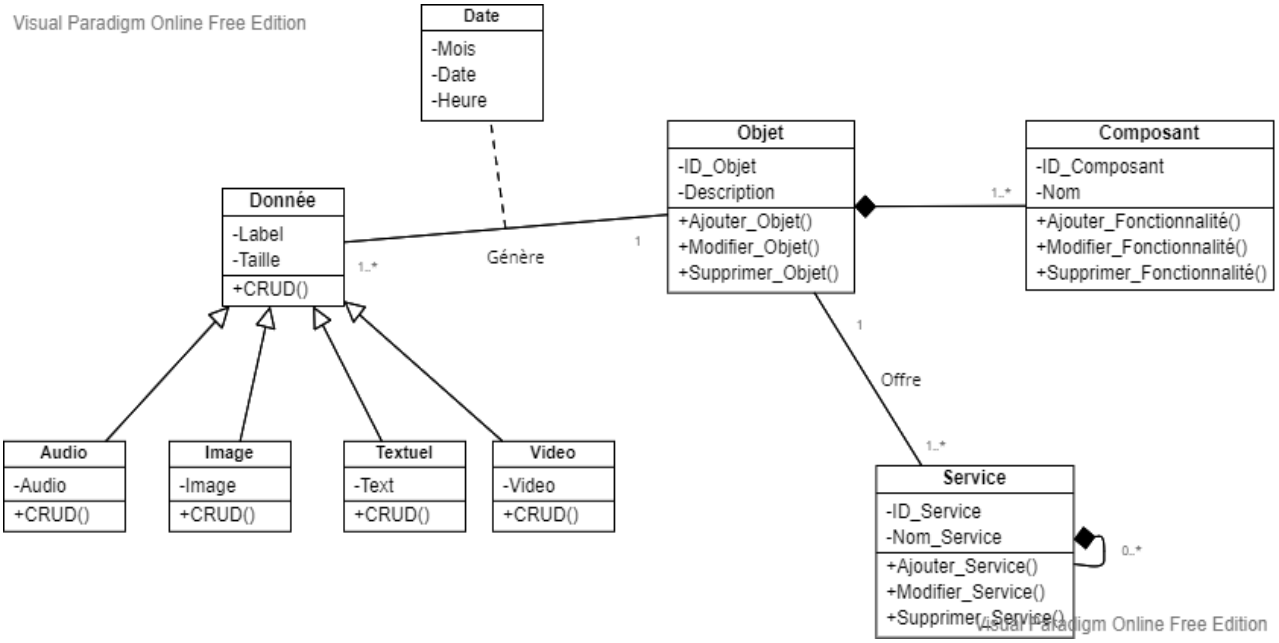


FIGURE 41 DIAGRAMME DE CLASSE OBJET DONNÉE

Après que les objets connectés y ont récolté des données sur l'environnement, l'utilisateur va créer des scénarios en s'adaptant à ces informations récoltées en y ajoutant un comportement sur le monde physique comme la montre la figure 42 :

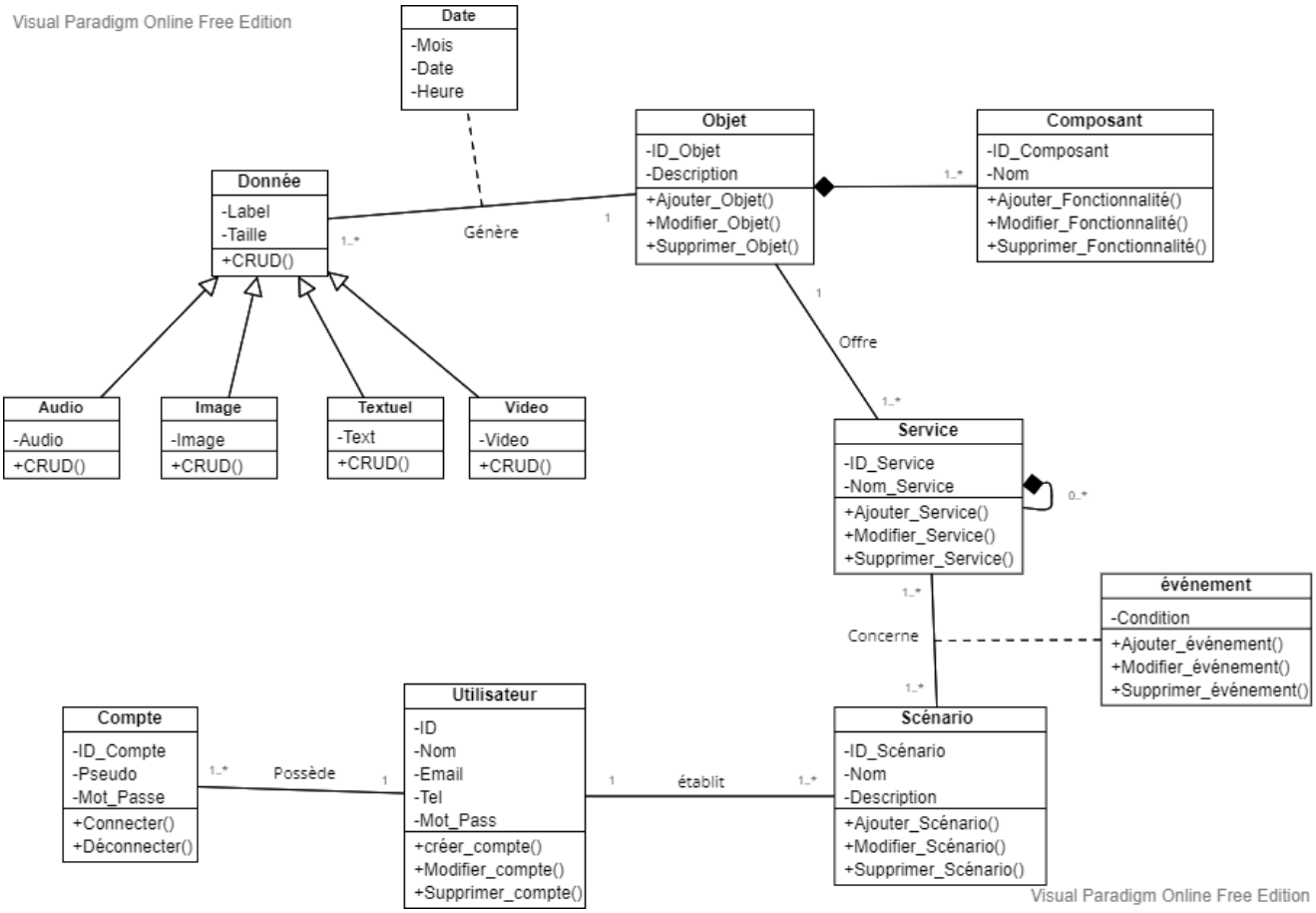


FIGURE 42 DIAGRAMME DE COLLABORATION OBJET SERVICE

La figure 43 représente le diagramme de classes détaillé du système :

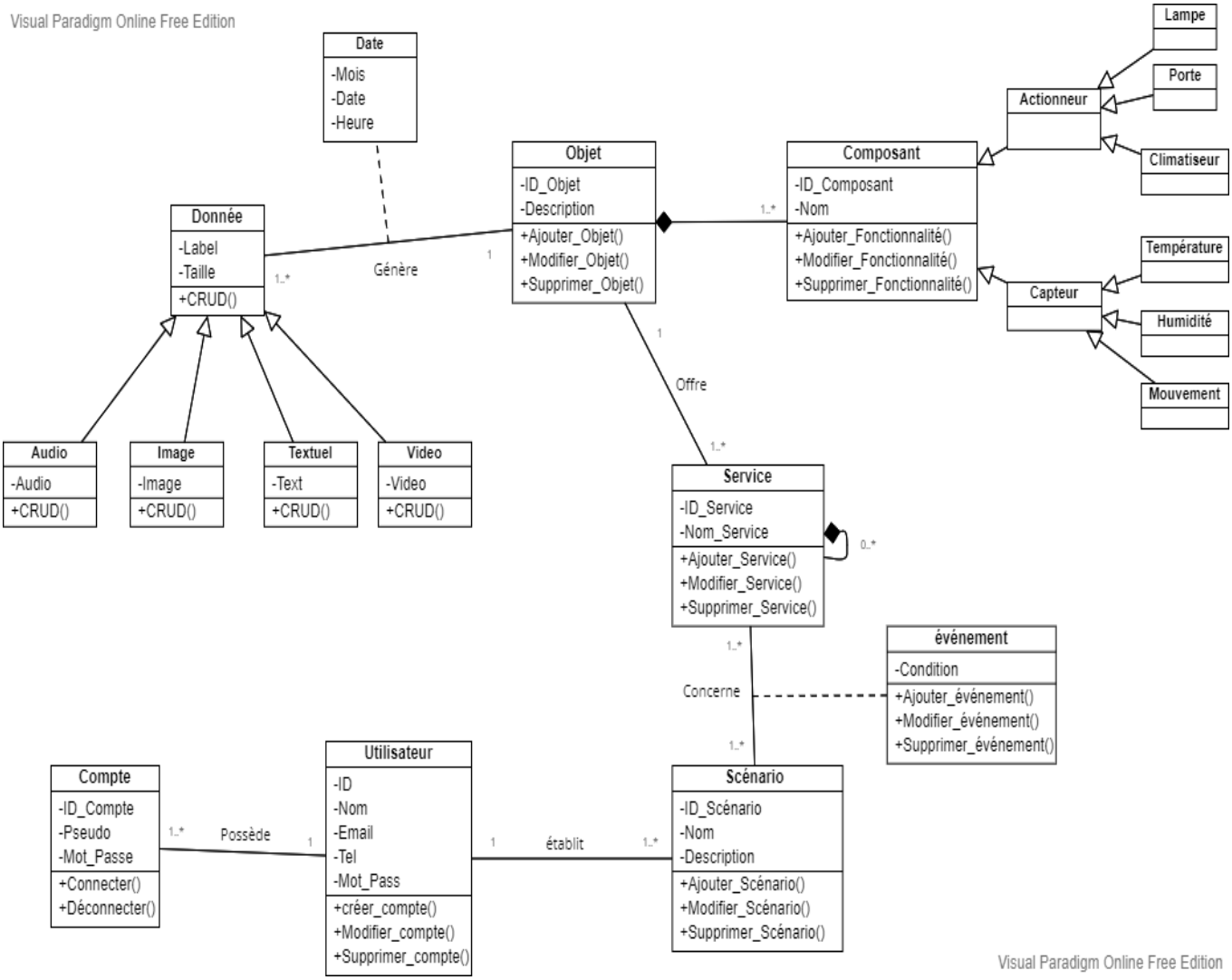


FIGURE 43 : DIAGRAMME DE CLASSE DÉTAILLÉ

2.3.3 Diagramme de séquence

2.3.3.1 Diagramme de séquence d'authentification

L'utilisateur doit saisir le mot de passe et l'identifiant dans le formulaire d'authentification. Une fois l'opération d'authentification validée, la page d'accueil s'affichera. Sinon, un message d'erreur s'affichera, comme le montre la figure 44.

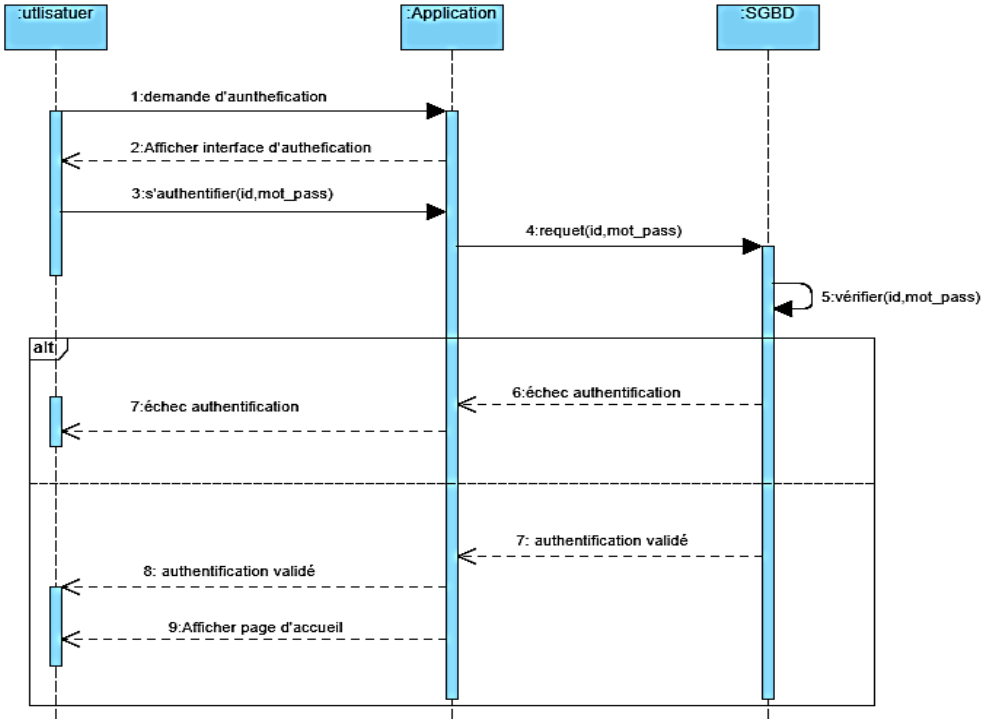


FIGURE 44 DIAGRAMME DE SÉQUENCE D'AUTHENTIFICATION

2.3.3.2 Diagramme de séquence consulter donnée

une fois l'utilisateur s'authentifie, il pourra soit consulter les données en temps réel ou il pourra consulter les logs (données stocké dans SGBD),comme le montre la figure 45.

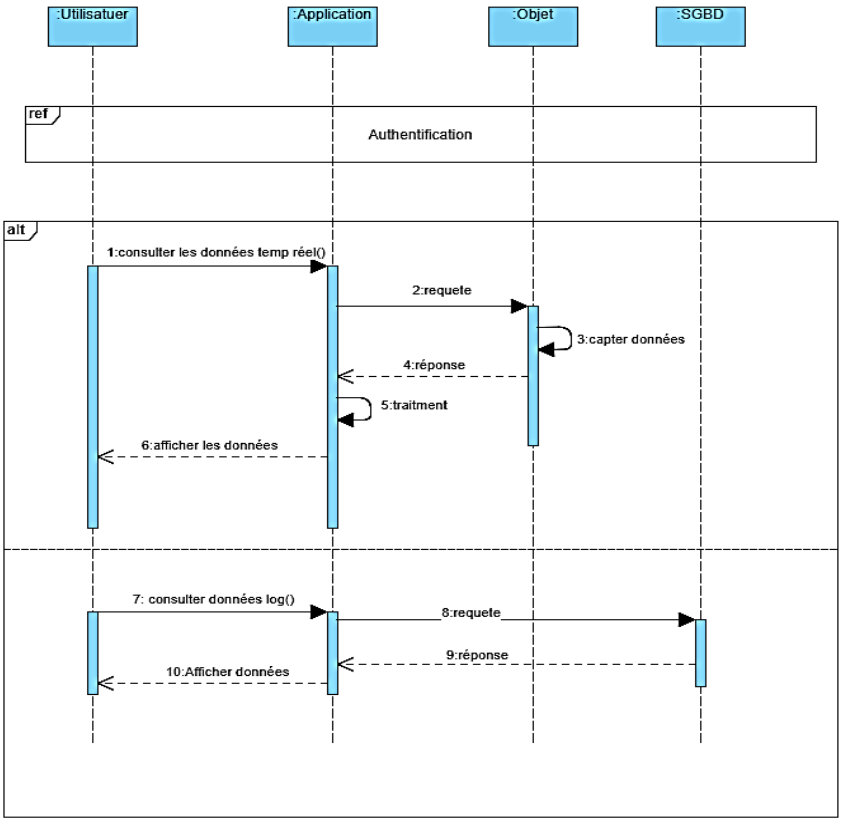


FIGURE 45 DIAGRAMME DE SÉQUENCE CONSULTER DONNÉE

2.3.3.3 Diagramme de séquence de stockage de données

Après que les objets connectés ayant récolté des données sur l'environnement, ils seront par la suite stockés dans un gisement cible (SGBD) comme le montre la figure 46.

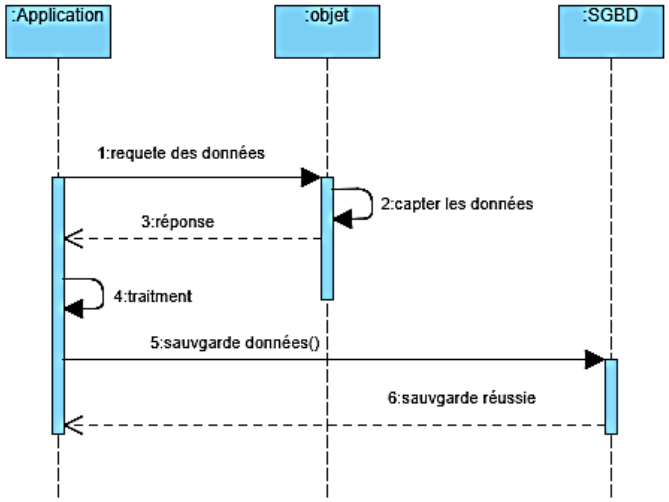


FIGURE 46 DIAGRAMME DE SÉQUENCE STOCKAGE DES DONNÉES

2.3.3.4 Diagramme de séquence de gestion de collaboration

Après que l'utilisateur s'authentifie, il pourra ajouter, supprimer, modifier et afficher ses collaborations (scénarios) selon ses besoins, comme le montre la figure 47.

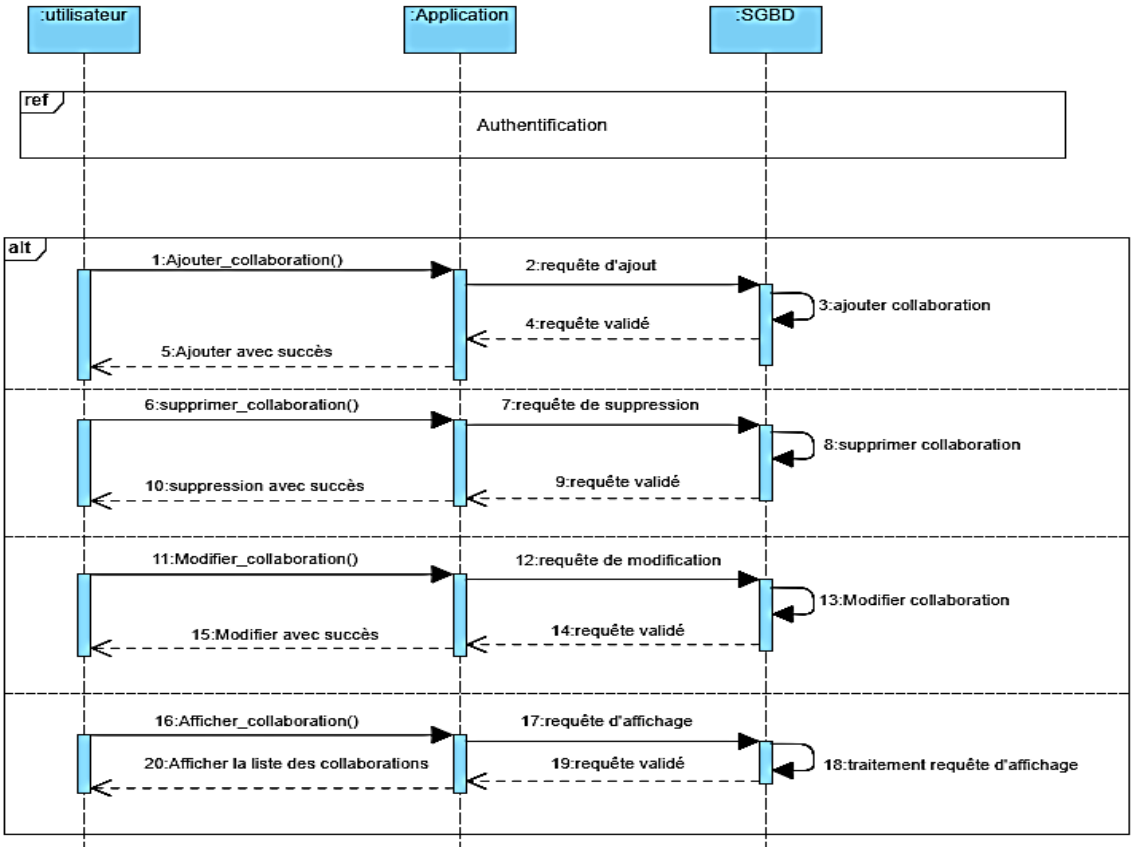


FIGURE 47 DIAGRAMME DE SÉQUENCE DE GESTION DE COLLABORATION

2.3.4 Diagramme d'activité

Ceci est une introduction avant de commencer la partie conception du Frameware qui est un programme intégré dans un matériel informatique (objet).

2.3.4.1 Modélisation d'objet actionneur

Il y a trois principaux besoins fonctionnels que doit effectuer l'objet actionneur :

- Servir l'interface graphique.
- Contrôler les objets liés avec l'actionneur dépendamment de l'interaction de l'utilisateur vial'interface graphique.
- Établir la connexion avec le routeur via le point d'accès wifi pour d'accéder à l'application à partir de n'importe quel endroit au monde.

Afin de satisfaire les besoins fonctionnels voici le diagramme d'activité que nous proposons figure 48 :

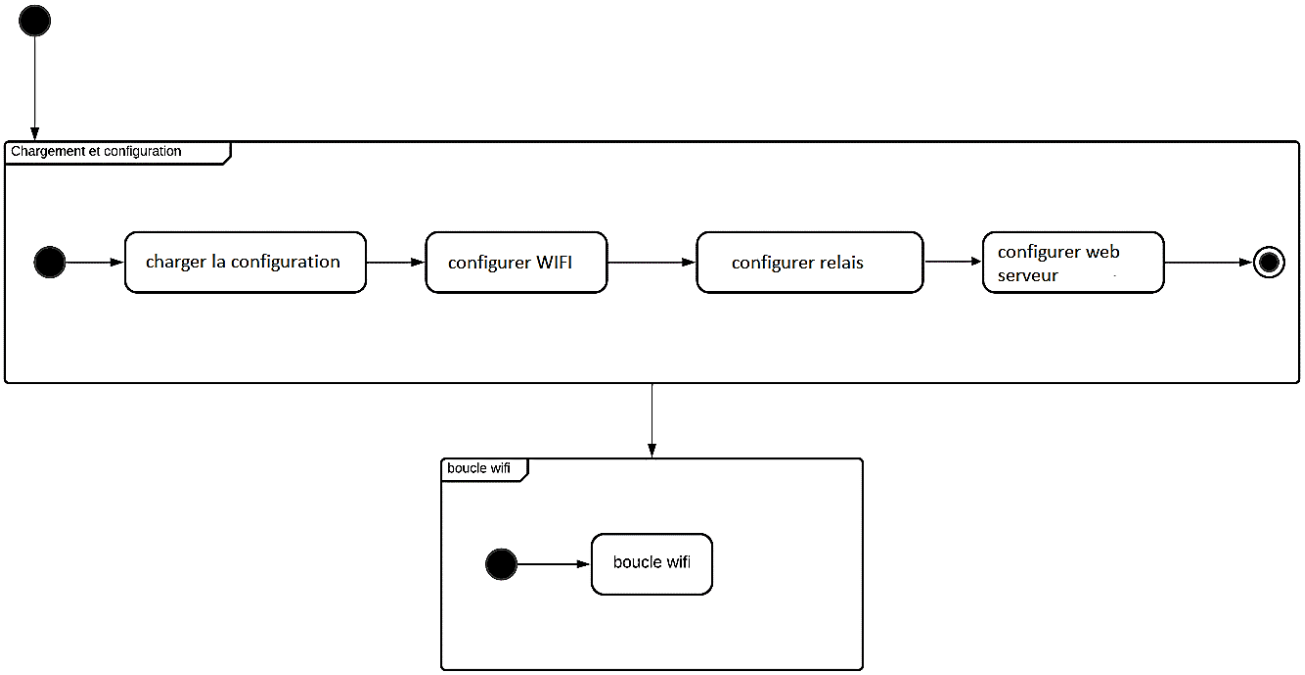


FIGURE 48 DIAGRAMME D'ACTIVITÉ GLOBALE SERVEUR ACTIONNEUR

a) Chargement et configuration

Après l'allumage de l'objet une série de tâches va s'exécuter :

- Chargement de la configuration :

Le serveur doit à tout prix charger la configuration initiale des états de relais sauvegardés à partir du fonctionnement précédent de l'appareil.

- Configuration Wi-Fi :

  Dans cette partie objet connecté doit se connecter au point d'accès Wi-Fi, si la connexion est établie la tache GET IP INFO s'exécutera afin de récupérer l'adresse IP donnée par le serveur DHCP du router, sinon l'appareil devra redémarrer jusqu'à connexion et attribution de l'adresse IP, Comme le montre la figure 49.

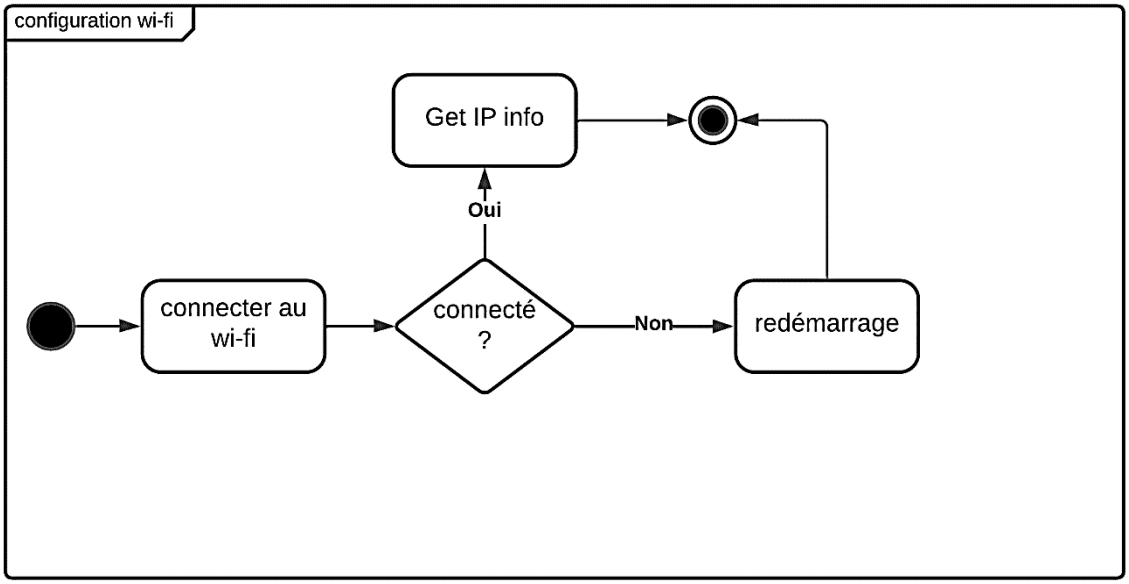


FIGURE 49 DIAGRAMME D'ACTIVITÉ CONFIGURATION WI-FI

- Configuration des relais

  Cette partie consiste à déclarer les pins de sortie (output) et initialiser les états des relais (figure 50).

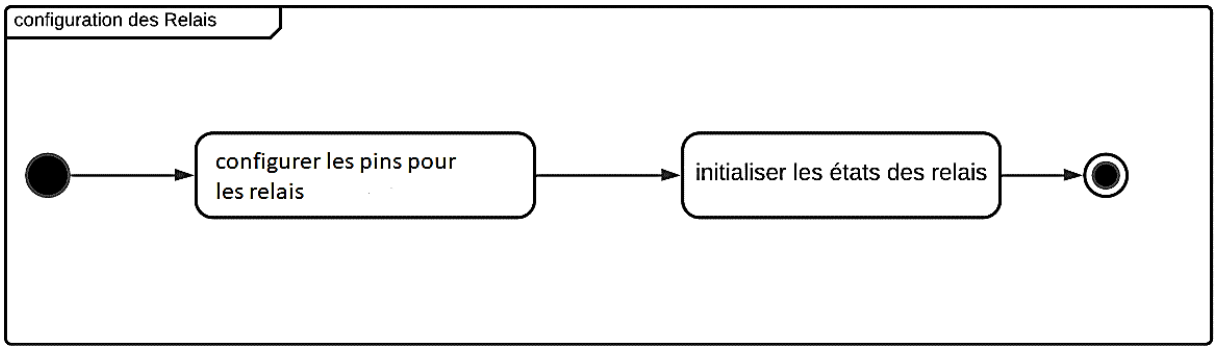


FIGURE 50 DIAGRAMME D'ACTIVITÉ CONFIGURATION CAPTEUR

Le serveur utilise le web socket ce qui fait que l'ajout du web socket sur le serveur est nécessaire. Pour la partie de démarrage du serveur nous allons définir des gestionnaires pour chaque requête reçue à partir de l'application web vers un web service, des opérations bien spécifiques vont être exécuté pour répondre à des requêtes bien définis. À l'envoie des

réponses au client, le serveur peut gérer d'autres connexion pendant qu'il envoie des réponses en arrière-plan. (Figure 51)

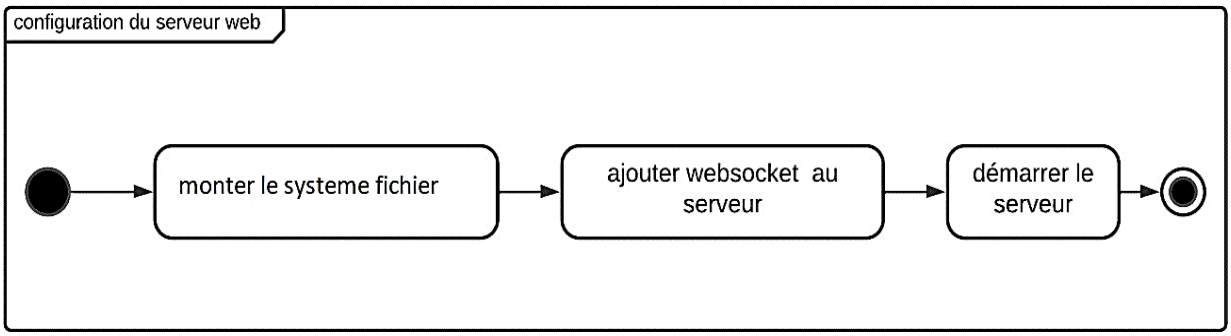


FIGURE 51 DIAGRAMME D'ACTIVITÉ CONFIGURATION SERVEUR WEB

b) Boucle wifi

Comme le montre la figure 52 la boucle wifi est utilisé pour faire clignoter les LED à fin d'indiquer que la connexion wifi est établie.

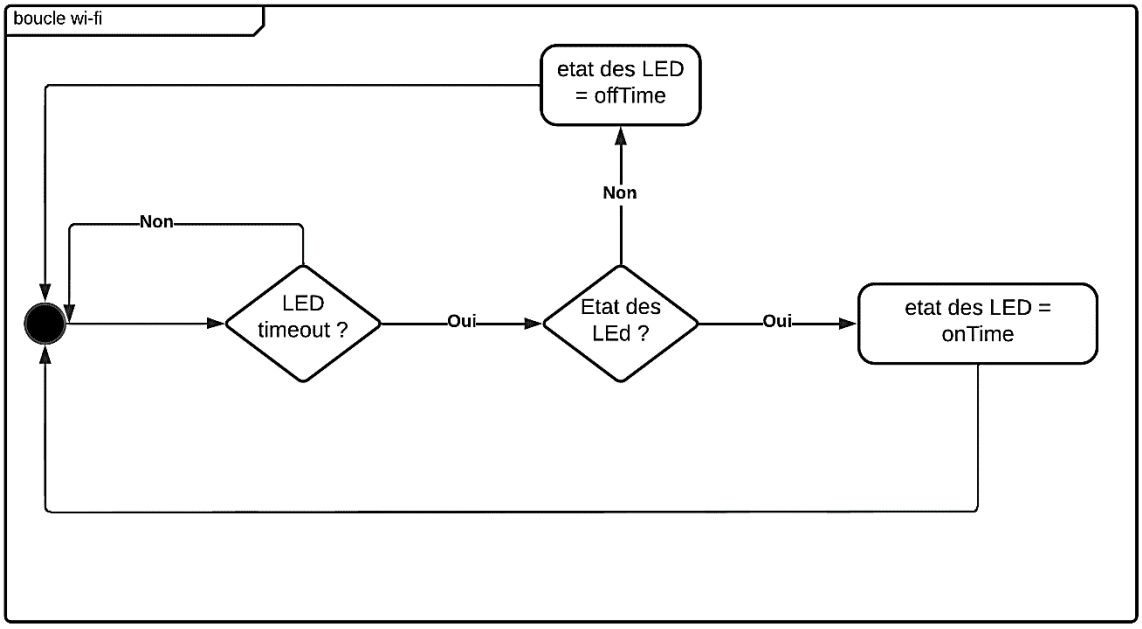


FIGURE 52 DIAGRAMME D'ACTIVITÉ BOUCLE WI-FI

2.3.4.2 Modélisation d'objet capteur

Il y a quatre principaux besoins fonctionnels que doit effectuer l'objet capteur :

- Servir l'interface graphique.
- Lire la température et l'humidité à partir des capteurs d'humidité et température.
- Détection mouvement.
- Établir la connexion avec le routeur via le point d'accès wifi pour d'accéder à

l'application à partir de n'importe quel endroit au monde.

Afin de satisfaire les besoins fonctionnels voici le diagramme d'activité que nous proposons figure 53 :

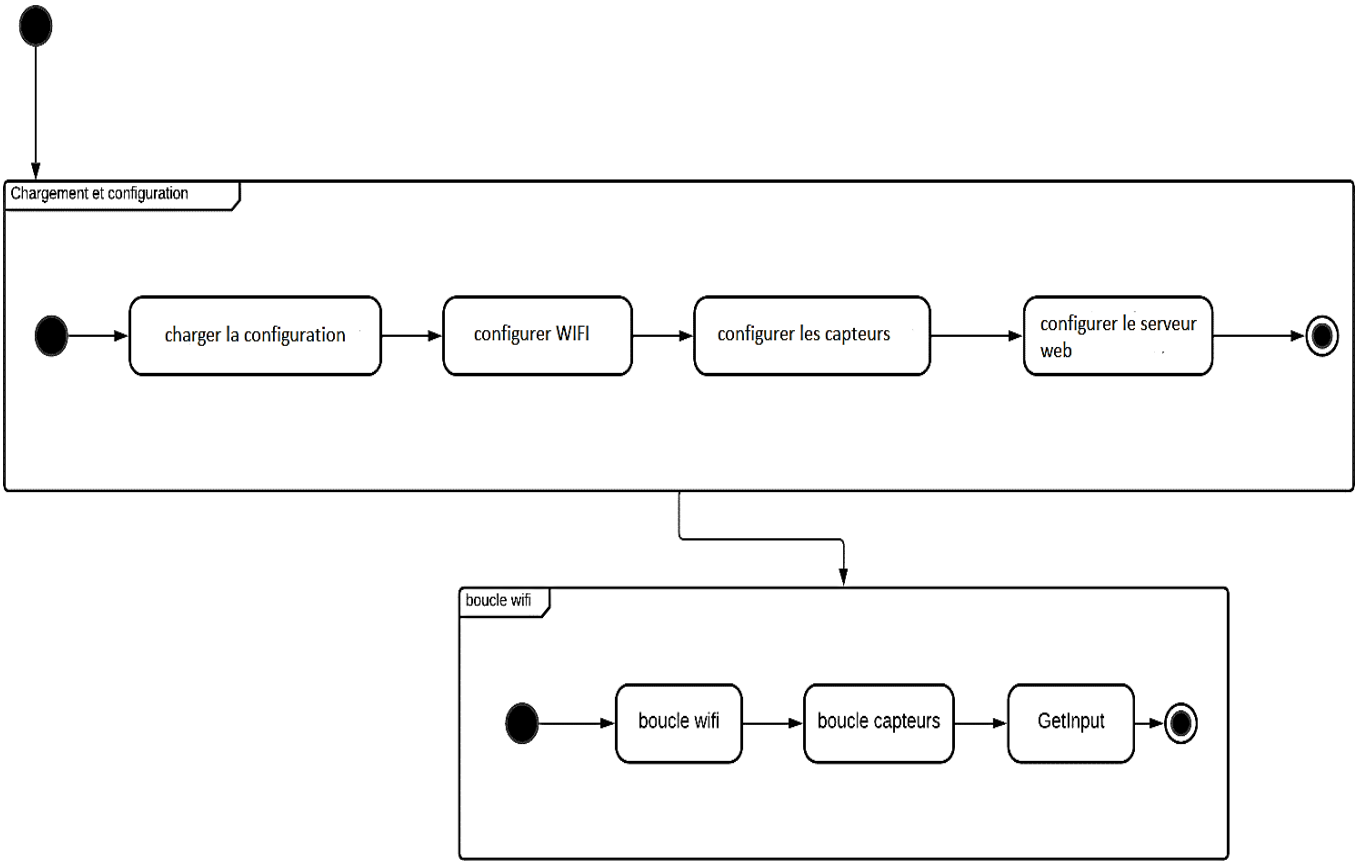


FIGURE 53 DIAGRAMME D'ACTIVITÉ GLOBALE SERVEUR DES CAPTEURS

- **Boucle capteur**

Comme le montre la figure 54, pour chaque intervalle de temps nous allons préparer la variable résultat, puis lire la température et l'humidité, puis nous allons construire la chaine de caractère qui va contenir le résultat, et à la fin nous définirons l'entrée de la chaine de caractère, qui va être utilisé dans la fonction GetInput mentionné précédemment dans la figure 53.

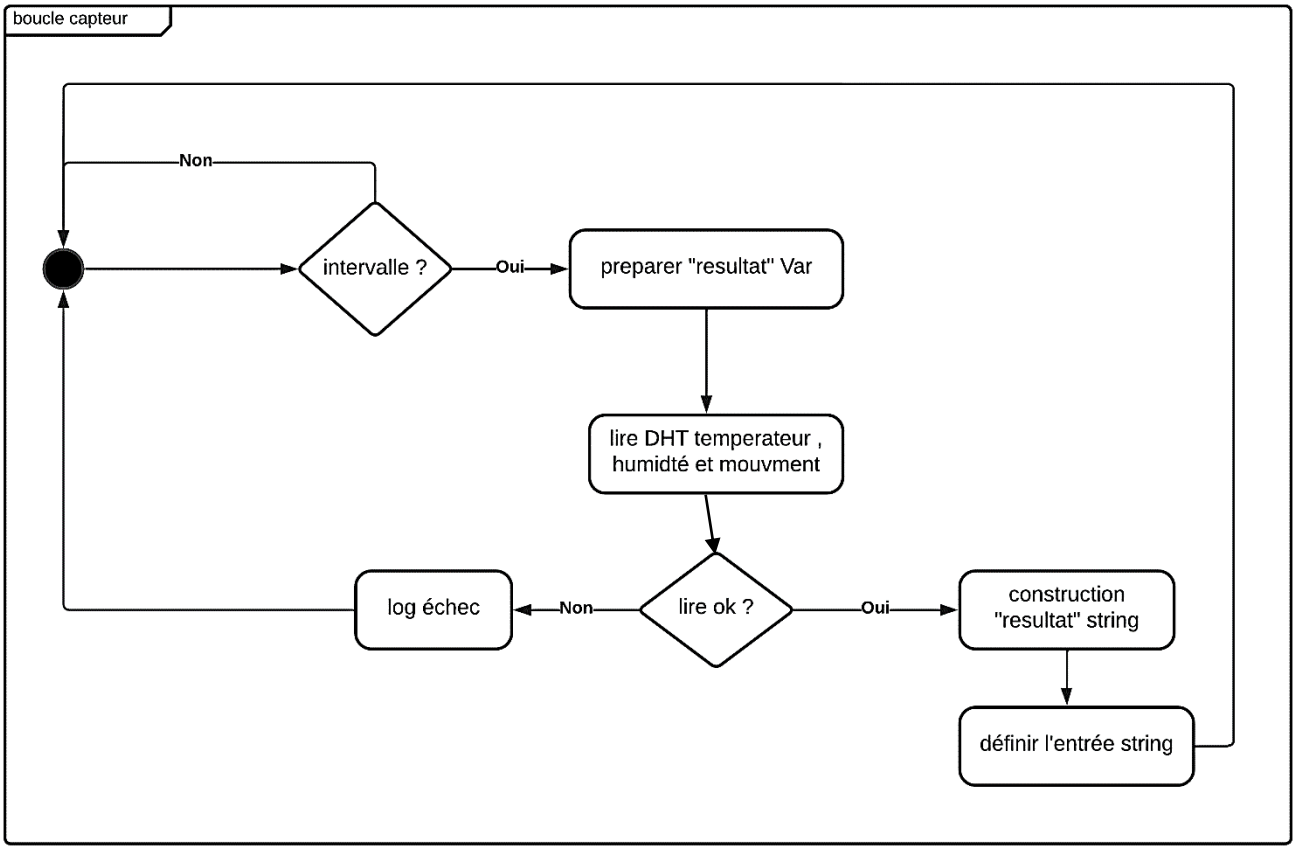


FIGURE 54 DIAGRAMME D'ACTIVITÉ BOUCLE CAPTEUR

Conclusion

La réalité est un peu plus complexe : établir une relation entre des millions d'objets nécessite tout un système qui gère le contrôle d'accès, la sécurité. Dans ce chapitre, nous avons illustré les plans de travail qui concerne : la structure du système, les fonctionnalités du système et les défis que nous allons accomplir. Dans le prochain chapitre nous allons décrire les outils permettant la réalisation de notre système et un aperçu sur les fonctionnalités réalisées.

Chapitre 3 :

Réalisation

Chapitre 3 : Réalisation

3.1 Introduction

Ce chapitre consiste à faire la réalisation de l'application modélisée précédemment. Il sera composé de trois parties :

1. Partie 1 : dédiée à l'assemblage des différents capteurs et actionneurs du prototype IOT. Assemblage réalisé :
    - Branchement des relais avec microcontrôleur
    - Branchement des capteurs mouvement, température et humidité avec microcontrôleur.
2. Partie 2 : dédiée à l'implémentation des serveurs (Model). Implémentation réalisée:
    - Implémentation des fonctions pour la gestion de relais (changer état relai, sauvegarder état relai, charger état relai).
    - Implémentation des fonctions dédiées au capteur (initialisation capteur et démarrer capteur).
    - Implémentation du serveur web (serveur websocket, web service Restfull).
3. Partie 3 : dédiée à l'implémentation de l'application client (View, ModelView). Implémentation réalisée :
    - Implémentation des fonctions acquisition données, présentation des données émises par les objets (sous forme de graphe ou textuelle en temps réel).
    - Implémentation des fonctions de contrôle (client websocket).

Pour notre application, nous avons choisi l'architecture MVVM (Model, View, ViewModel) afin de découpler au maximum l'affichage (View) de la logique (ViewModel) et la logique des données (Model).

3.2 Architecture MVVM et web service REST

La figure 55 montre les déférents ViewModel qui vont être utilisé dans notre code. Du coté serveur nous avons créé trois ressources, quand une requête AJAX est envoyé au serveur pour accéder aux données se trouvant à la ressource « */derniers valeur »* le serveur renvoi les données sous frome JSON *{"R1":0,"R2":0,"R3":1,"R4":1, "T":20.7,"H:50"}.*
Ces données vont être gérés par derniers valeurs VIEWMODEL en appliquant certaines opérations, puis appliquer le data bindings afin que l'HTML puisse afficher ces données sur interface graphique. Pareil pour les ressources « */statue »* et « */config »* mais d'une façon plus différente.
On peut déclarer que la VIEW est une interface utilisateur visible interactive représentant le statu du VIEWMODEL. Elle affiche les informations du VIEWMODEL et envoi des commandes au VIEWMODEL, si l'utilisateur click sur un bouton une mise à jour sera appliqué sur le VIEWMODEL.

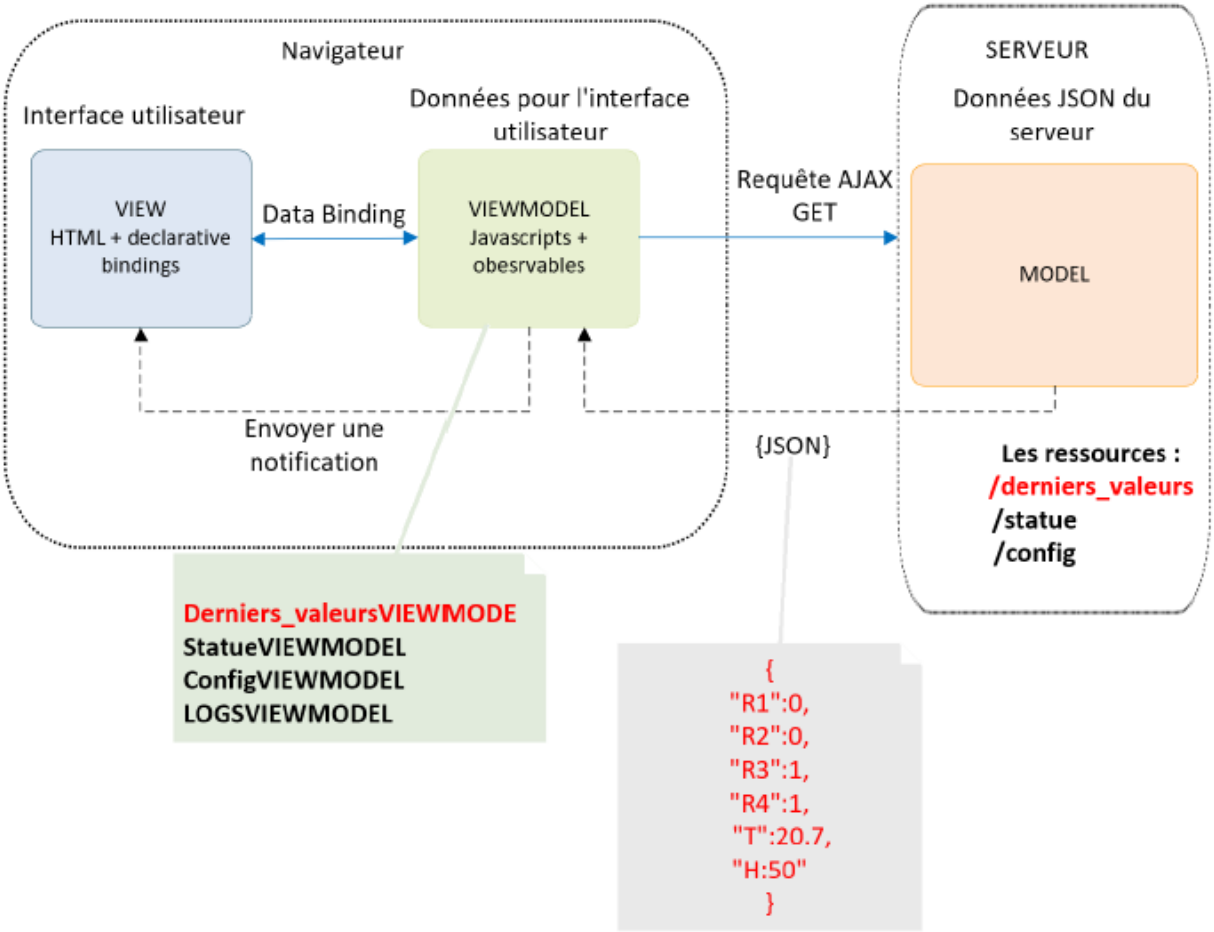


FIGURE 55 : ARCHITECTURE MVVM UTILISÉ

3.3 Les étapes de l'implémentation

3.3.1 les outils utilisés

Plusieurs langages et outils on était utilisé durant le développement de l'application qui permet la collaboration entre les objets connectés, nous présentons dans ce qui suit les langages et outils utilisés :

1. **Les langages de programmation :**

- **PHP** : pour l'implantation de l'application serveur.

PHP (acronyme récursif de PHP : Hypertext Preprocessor) est un langage de script open source à usage général largement utilisé, particulièrement adapté au développement Web et pouvant être intégré au HTML. [40]

FIGURE 56 : LOGO PHP [40]

- **C++ :** pour l'implémentation des serveurs web et implémentation du firmware des objets.

C++ est un langage multiplateforme qui peut être utilisé pour créer des applications hautes performances. [41]

FIGURE 57 : LOGO C++ [41]

- **JavaScript** : pour l'implantation de l'application client.

JavaScript est un langage de script ou de programmation qui permet d'implémenter des fonctionnalités complexes sur des pages Web [42]

FIGURE 58 : LOGO JAVASCRIPT [42]

- **Knockout :** pour l'implantation de l'application client.

Knockout est une bibliothèque JavaScript qu'aide à créer des interfaces utilisateur d'affichage et d'édition riches et réactives avec un modèle de données sous-jacentes [43]

FIGURE 59 : JAVASCRIPT KNOCKOUT [43]

- **MySQL :** pour la sauvegarde des données.

FIGURE 60 : LOGO MYSQL [48]

MySQL, le système de gestion de base de données SQL Open Source le plus populaire, est développé, distribué et pris en charge par Oracle Corporation. [48]

- **Bootstrap :** pour le développement de l'interface graphique.

FIGURE 61 : LOGO BOOTSTRAP [49]

Bootstrap est une boîte à outils frontale puissante et riche en fonctionnalités. [49]

- **HTML** et **CSS :** pour le développement de l'interface graphique.

FIGURE 62 : LOGO HTML ET CSS [50]

HTML (Hypertext Markup Language) et CSS (Cascading Style Sheets) sont deux des technologies de base pour la création de pages Web. [50]

2. Les outils logiciels

- **Arduino (IDE) :** pour la programmation des microcontrôleurs.

FIGURE 63 : LOGO ARDUINO IDE [44]

Le logiciel open source Arduino (IDE) facilite l'écriture de code et son téléchargement sur la carte. Ce logiciel peut être utilisé avec n'importe quelle carte Arduino. [44]

- **Visual Studio Code :** pour l'implantation de l'application serveur/client_Visual Studio.

FIGURE 64 : LOGO VISUAL STUDIO CODE [45]

Visual Studio Code est un éditeur de code redéfini et optimisé pour créer et déboguer des applications Web et cloud modernes. [45]

- **POSTMAN :** pour le test des services web et serveur websockets.

Postman est une plate-forme API pour la création et l'utilisation d'API. Postman simplifie chaque étape du cycle de vie des API et rationalise la collaboration afin que vous puissiez créer de meilleures API plus rapidement. [46]

FIGURE 65 : LOGO PLATEFORME POSTMAN [46]

- **Apache server :** pour l'exécution d'application serveur.

Apache HTTP Server vise à développer et à maintenir un serveur HTTP open source pour les systèmes d'exploitation modernes. [47]

FIGURE 66 : LOGO APACHE SERVEUR [47]

3.3.2 Assemblage du prototype IOT

Cette partie est dédiée à l'assemblage du prototype IOT utilisé durant la réalisation de notre projet. Nous allons voir en détail les composants utilisés et comment l'assemblage a été fait.

➢ **Choix du microcontrôleur :**

Il ne fait aucun doute qu'il y a eu une augmentation massive du développement de cartes de développement IoT à faible coût et à faible consommation dans l'industrie électronique. D'Intel à Arduino open-source, toutes les marques ont intensifié leur jeu pour battre la concurrence dans le secteur de l'IoT en évolution rapide (figure 67).

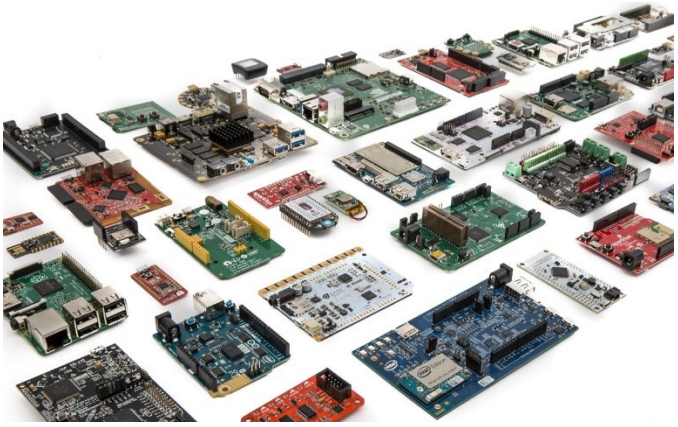


FIGURE 67 : LES MICROCONTRÔLEURS

Il est important de choisir la meilleure carte de développement IoT. Puisqu'il sert de cerveau à notre projet, nous ne pouvons pas prendre cette étape à la légère.

Nous devons choisir une carte qui peut communiquer avec tous les composants électroniques interconnectés avec une certaine facilité. Voici donc les trois étapes clés que nous devons suivre pour vous assurer de prendre la bonne décision comme le montre la figure 68.

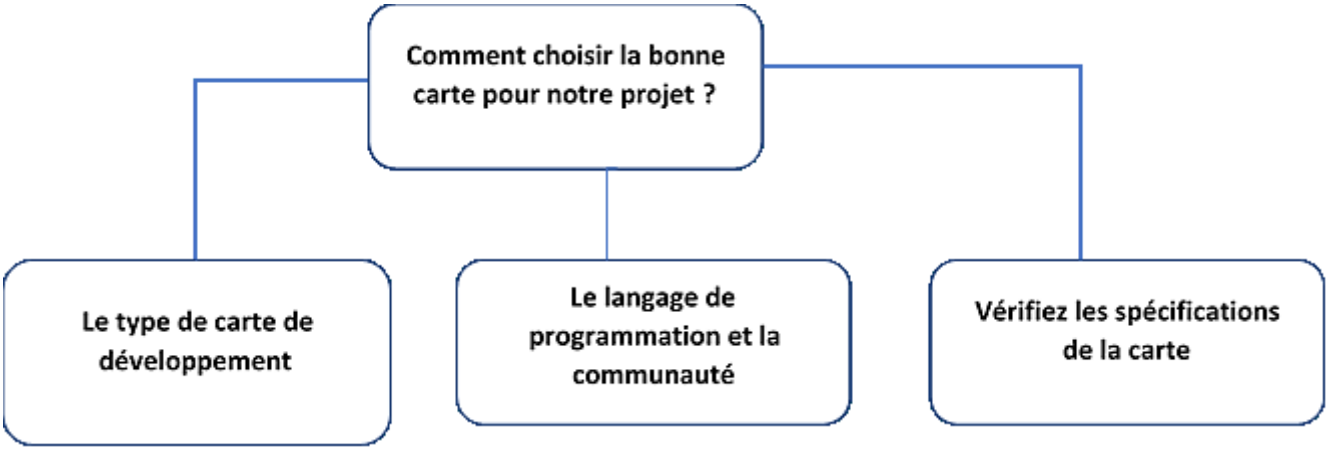


FIGURE 68 : CHOIX MICROCONTRÔLEUR

1. **Le type de carte de développement**

   La carte de développement peut appartenir à plusieurs catégories en fonction de sa conception et de ses spécifications, elle peut s'agir d'une carte à microcontrôleur comme Mega et Arduino Uno ou d'une carte SBC (single-board computers) comme BeagleBone Black et Raspberry Pi.

2. **Le langage de programmation et la communauté**

   Le langage de programmation que nous utiliserons est un facteur important à prendre en compte dans la carte de développement IoT. Par exemple, nous pourrions avoir besoin de la prise en charge de plusieurs langages, systèmes d'exploitation et IDE pour créer une expérience riche et complète.
   C et C++ sont des langages de programmation universels et de nombreuses cartes les prennent en charge.
   Sinon, il sera assez difficile d'obtenir de l'aide si nous rencontrons un obstacle dans notre projet.

3. **Les spécifications de la carte**

   Une carte avec plus de spécifications coûte généralement plus cher et deuxièmement, cela crée parfois plus de problèmes si l'équipe de développement n'est pas expérimentée dans le travail avec des cartes compliquées auparavant.
   Par conséquent, il est préférable de trouver une carte de spécification moyenne. Bien sûr, si nous prévoyons d'ajouter plus de fonctionnalités à la carte plus tard.

Nous avons fini par choisir le microcontrôleur ESP8266 car il permet de créer de nombreux projets IoT et domotiques à faible cout et pour sa popularité dans la communauté IoT.

- **Choix des capteurs et actionneurs :**
  Pour ce qui est des composant capteurs et actionneurs nous avons choisi ce qui a été disponible dans le marché car les choix des composants n'ont pas une grande influence sur notre projet.

➢ **Composants utilisés :**

a) ESP8266 (MCU NODE)

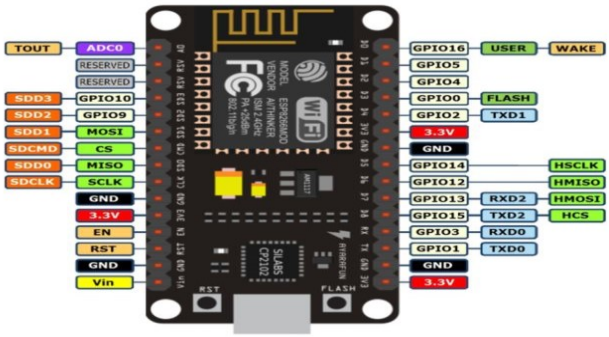


FIGURE 69 : ESP8266 (MCU NODE)

L'ESP8266 est un circuit intégré à microcontrôleur avec connexion Wi-Fi développé par le fabricant chinois Espressif. (Figure 69)

L'ESP8266 peut se programmer de plusieurs façons :

- En Lua, interprétés ou compilés, avec le firmware NodeMCU (langage par défaut).
- En C, avec le SDK esp-open-sdk basé sur la chaîne de compilation GCC ou avec le SDK d'Espressif.
- En C++, avec l'IDE (Environnement de développement) Arduino.
- En Go, avec le framework Gobot.
- En JavaScript, avec le firmware Espruino.
- En MicroBlocks, notamment sur NodeMCU.
- En MicroPython, avec le firmware MicroPython ou CircuitPython.

Le langage de programmation que nous utiliserons pour programmer l'ESP8266 sera le langage C++.

Caractéristiques

Le tableau ci-dessous montre les caractéristiques du Microcontrôleur ESP8266

Caractéristique	Valeur
CPU :	Tensilica 32-bit RISC CPU Xtensa LX106
Tension d'alimentation :	7-12V
Tension login :	3.3V
E/S digitales :	16
Entrées analogiques :	1

Flash:	4000kB
SRAM:	64kB
EEPROM:	NCkB
Fréquence d'horloge :	80 MHz
Wifi:	oui
Bluetooth:	Non
SD card :	Non

TABLEAU 9 : CARACTÉRISTIQUES MICROCONTRÔLEUR

b) Capteur de température et d'humidité DHT11

Le capteur DHT11 est capable de mesurer des températures de 0 à +50°C avec une précision de +/- 2°C et des taux d'humidité relative de 20 à 80% avec une précision de +/- 5%. Une mesure peut être réalisée toutes les secondes. (Figure 70)

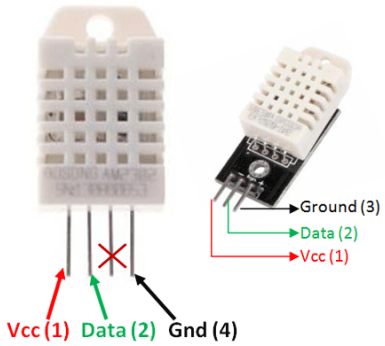


FIGURE 70 : CAPTEUR DE TEMPÉRATURE ET D'HUMIDITÉ DHT11

Caractéristiques

Le tableau ci-dessous montre les caractéristiques du Microcontrôleur ESP8266

Caractéristique	Valeur
Humidité (relative %)	0 ~ 100 %
Précision (humidité)	+/- 2% (+/- 5% aux extrêmes)
Température	-40 ~ +150°C
Précision (température)	+/- 0.5°C
Fréquence mesure max	2Hz (2 mesures par seconde)
Tension d'alimentation	3 ~ 5 volts
Stabilité à long terme	+/- 0.5% par an

TABLEAU 10 : CARACTÉRISTIQUE DHT 11

c) Module relais 5V DC 4 canaux avec optocoupleur :

Ce module est Idéal pour le pilotage de dispositifs externes dont la tension/courant sont élevés. Cette platine intègre 4 relais 5V avec optocoupleurs de protection dont le pilotage s'effectue sur un connecteur de 2,54 mm directement depuis un module Arduino ou compatible. (Figure 71)

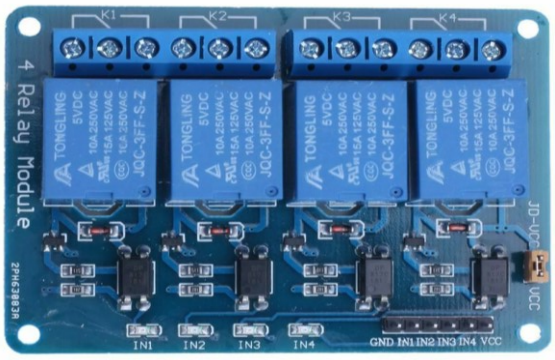


FIGURE 71 : MODULE RELAIS 5V DC 4 CANAUX AVEC OPTOCOUPLEUR

Caractéristiques

- Quatre sorties relais RTC (NO-COM-NF)
- Raccordement sur les relais via bornes à vis
- Pilotable direct depuis Arduino
- Entrées isolées par des optocoupleurs
- Pouvoir de coupure : 30V / 10A max.
- Dimensions : 75 x 55 x 19,3 mm - Poids : 61 g

d) Assemblage :

Les figures 72, 73, 74 représente l'assemblage de notre prototype

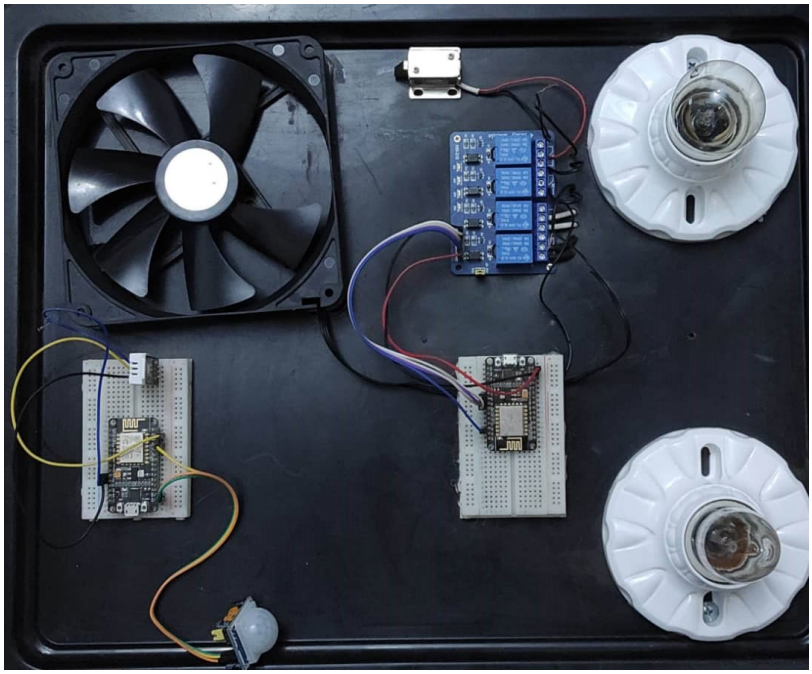


FIGURE 72 : ASSEMBLAGE DE PROTOTYPE

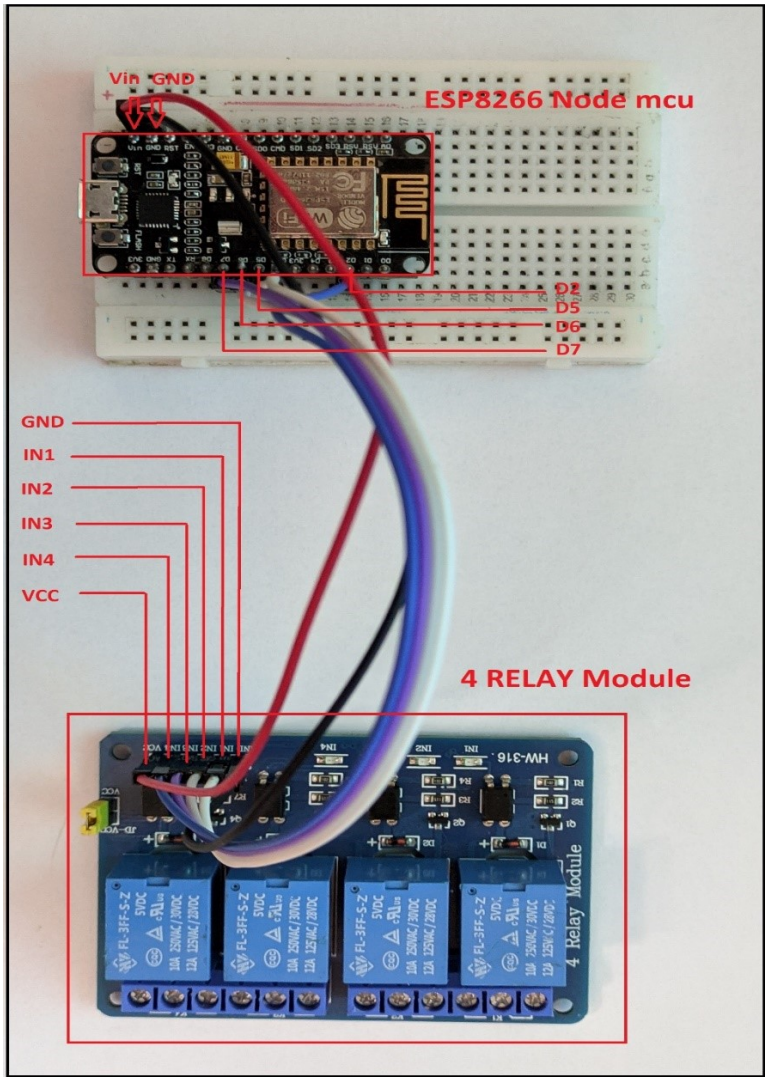


FIGURE 73 : ASSEMBLAGE ESP8266 POUR LES CAPTEURS

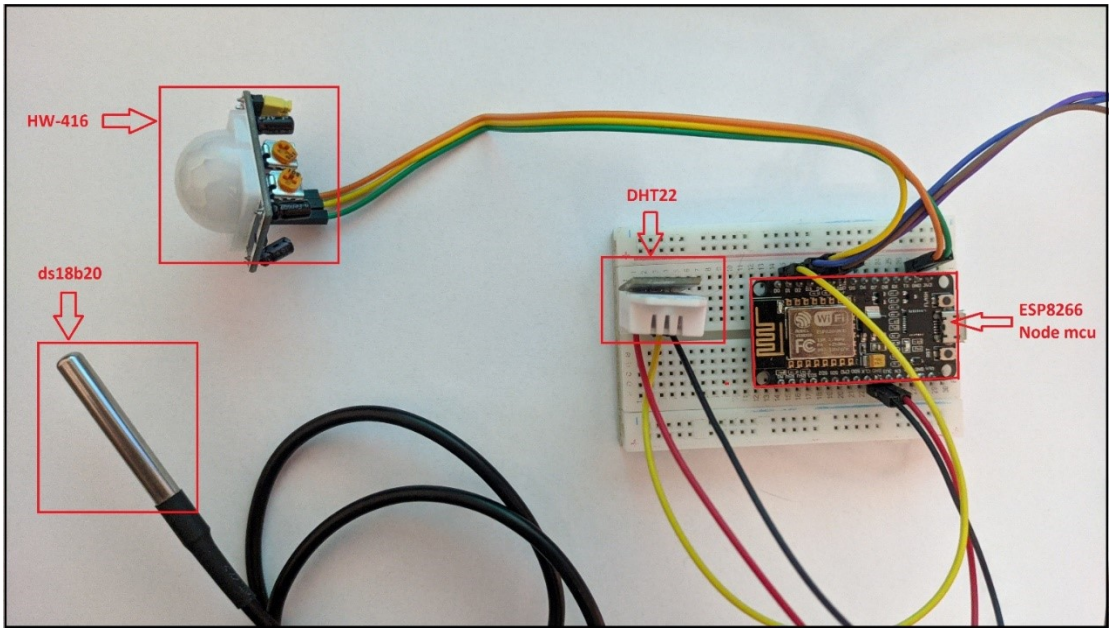


FIGURE 74 : ASSEMBLAGE ESP8266 POUR LES ACTIONNEURS

3.3.3 Implémentation microprogramme objet :

3.3.3.1 L'implémentation du serveur actionneur

a) Le Relay.h

Nous commencerons par inclure la bibliothèque <Arduino.h> pour accéder à toutes les fonctionnalités de base d'Arduino. Cependant, c'est un fichier très important sans lequel nous ne pouvons pas accéder aux fonctionnalités de l'Arduino telles que GPIO IO, Analog IO, etc. Puis nous définirons les pins pour les quatre relais.

Ensuite, nous définirons la fonction relay_setup () qui permettra de configurer les pins spécifiés pour les relais afin qu'ils puissent se comporter comme sortie (OUTPUT).
La deuxième fonction à définir est relay_set_state (uint8_t id, bool state), elle permet de modifier l'état d'un relai en spécifiant l'identifiant du relai et l'état.

Code complet relay.h est comme suit :

```
#ifndef _RELAY_H
#define _RELAY_H
#include <Arduino.h>
extern void relay_setup();
extern void relay_set_state(uint8_t id, bool state);
#endif //_RELAY_H
```

a) Relay.ccp

Ce fichier va contenir les fonctions relay_set_state (uint8_t id, bool state) et Relay_setup ().
La fonction Relay_setup () utilise la fonction relay_set_state pour initialiser les états des relais et la fonction pinmode () est utilisé pour définir les pins en sortie (OUTPUT).

Code complet Relay.ccp est comme suit :

```
#include "relay.h"
#include "debug.h"
void relay_setup() {
    pinMode(RELAY_1_PIN, OUTPUT);
    pinMode(RELAY_2_PIN, OUTPUT);
    pinMode(RELAY_3_PIN, OUTPUT);
    pinMode(RELAY_4_PIN, OUTPUT);
    relay_set_state(RELAY_1, relay_1_state);
    relay_set_state(RELAY_2, relay_2_state);
    relay_set_state(RELAY_3, relay_3_state);
    relay_set_state(RELAY_4, relay_3_state);}
void relay_set_state(uint8_t id, bool state) {
    switch(id) {
        case RELAY_1:
            digitalWrite(RELAY_1_PIN, state);
            relay_1_state = state;
```

```
        break;
        case RELAY_2:
            digitalWrite(RELAY_2_PIN, state);
            relay_2_state = state;
        break;
        case RELAY_3:
            digitalWrite(RELAY_3_PIN, state);
            relay_3_state = state;
        break;
        case RELAY_4:
            digitalWrite(RELAY_4_PIN, state);
            relay_4_state = state;
        break;
        default:break;}}
```

a) Config.h

Dans config.h, nous trouvons trois fonctions :

- Config_load_settings ()
- Config_save_relay_states ()
- Config_load_realy_states ()

Code Config.h est comme suit :

```
#ifndef _CONFIG_H
#define _CONFIG_H
#include <Arduino.h>
extern void config_load_settings();
extern void config_save_relay_states();
extern void config_load_relay_states();
#endif
```

a) Config. ccp

Objectif de config.ccp est de développer les fonctions pour sauvegarder les états des relais. Afin de réaliser ce dernier nous devrons manipuler la mémoire EEPROM de microcontrôleur esp 8266.

Le fichier config.ccp sera composé de deux fonction principale :

Config_load_setting () utilise la fonction config_load_realy_states () pour charger la configuration des relais si les relais sont verrouillés comme le montre le code suivant :

Code Config.ccp est comme suit :

```
void config_load_settings() {
    EEPROM.begin(EEPROM_SIZE);
    if(relay_latch_enabled) {
        config_load_relay_states();}
    EEPROM.end();}
```

- Le code de la fonction config_load_relay_states () est comme suit :

Code Config.ccp est comme suit :

```
void config_load_relay_states() {

   uint8_t state_byte = 0;
   EEPROM.begin(EEPROM_SIZE);
   state_byte = EEPROM.read(EEPROM_RELAY_STATES_START);
   EEPROM.end();
   relay_1_state = state_byte & BITMASK_RELAY_1;
   relay_2_state = state_byte & BITMASK_RELAY_2;
   relay_3_state = state_byte & BITMASK_RELAY_3;
   relay_4_state = state_byte & BITMASK_RELAY_4;}
```

- Le code de la fonction save_relay_states () est comme suit :

Code Config.ccp est comme suit :

```
void config_save_relay_states() {
uint8_t state_byte = 0;
state_byte = relay_1_state | (relay_2_state << 1) | (relay_3_state << 2) |
(relay_4_state << 3);
EEPROM.begin(EEPROM_SIZE);
EEPROM.write(EEPROM_RELAY_STATES_START, state_byte);
EEPROM.end();}
```

Pour sauvegarder les états des relais, nous devrons écrire les états des relais dans la mémoire EEPROM en utilisant la fonction EEPROM.write () avec un décalage d'un bit pour chaque relai. Pour la lecture des états des relais, nous utilisons la fonction EEPROM.read () en spécifiant le bit masque pour chaque relai.

b) Wifi.ccp

Nous allons inclure la bibliothèque <ESP8266WIFI.h> qui va contenir les fonctions pour manipuler les wifi de microcontrôleur eps8266.

Nous allons définir le « esid » et le « epass » de notre routeur au quelle ce microcontrôleur va se connecter.

La fonction get_ip() va afficher l'adresse ip de microcontrôleur dans le moniteur de série et il allume un LED pour indiquer à l'utilisateur que le microcontrôleur est connecté au routeur.

Code Wifi.ccp est comme suit :

```
#include <ESP8266WiFi.h>
String ipadress = "";
String esid = "SS";
String epass = "***********";
#ifdef WIFI_LED
    int wifiLedState = !WIFI_LED_ON_STATE;
    unsigned long wifiLedTimeout = millis();
#endif
void get_ip () {
```

```
    IPAddress myAdress = WiFi.localIP();
    char tmpStr[40];
    sprintf(tmpStr, "%d.%d.%d.%d", myAdress[0], myAdress[1], myAdress[2],
myAdress[3]);
    ipadress = tmpStr;
 DBUG("[WiFi] Connected, IP: ");
    DBUGLN(tmpStr);
    #ifdef WIFI_LED
        wifiLedState = WIFI_LED_ON_STATE;
        digitalWrite(WIFI_LED, wifiLedState);
    #endif}
```

La fonction start_client() permet d'exécuter la procédure de connexion via la fonction wifi.begin(). Au cas d'échec de connexion, un message s'affiche dans le moniteur de série.

Code Wifi.ccp est comme suit :

```
void start_client() {
    WiFi.begin(esid.c_str(), epass.c_str());
    delay(50);
    if(WiFi.waitForConnectResult() != WL_CONNECTED)
    {  DBUGLN("Wifi non connecté redémarage...");
        delay(1000);
        ESP.restart(); }
    get_ip();}
```

c) Web_server.h

Pour notre serveur web nous allons utiliser la librairie <ESP_async_web_server>. Afin d'utiliser cette librairie, nous devons mettre en place un montage de système fichier. Le serveur utilise le web socket ce qui fait que l'ajout du web socket sur le serveur est nécessaire.

Pour la partie de démarrage du serveur nous allons définir les handlers (les gestionnaires) pour chaque requête reçue à partir de l'application web, des opérations bien spécifiques vont être exécuté pour répondre à des requêtes bien définis.

Nous devons aussi mentionner que l'utilisation de la librairie <ESP_async_web_server> fournie un moyen facile de développer un serveur web asynchrone, ce type de serveur a plusieurs avantages, il peut gérer plus d'une connexion. À l'envoie des réponses au client, le serveur peut gérer d'autres connexion pendant qu'il envoie des réponses en arrière-plan.

Pour l'esp 8266, il est nécaissaire d'inclure la bibliothèque <ESPAsyncTCP> pour pouvoir utiliser la bibliothèque <ESP_async_web_server>. <ESPAsyncTCP> s'agit d'une bibliothèque TCP entièrement asynchrone, destinée à permettre un environnement réseau multi-connexion sans problème pour les microcontrôleurs ESP8266.

Les deux bibliothèques citer précédemment sont inclut dans le fichier web_server.h comme suit :

Code Web_server.h est comme suit :

```
#ifndef _WEB_SERVER_H
#define _WEB_SERVER_H
#include <Arduino.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
extern AsyncWebServer server;
extern void web_server_setup();
#endif // _WEB_SERVER_H
```

a) web_server.ccp

L'objet AsyncWebServer sera utilisé pour configurer le serveur web du NodeMCU ESP8266. Nous passerons le port HTTP par défaut, qui est 80, en entrée du constructeur. Ce sera le port sur lequel le serveur écoutera les requêtes. En outre, nous aurons besoin d'un objet de la classe AsyncWebSocket, que nous utiliserons pour configurer notre point de terminaison websocket et la fonction de gestion correspondante. Le constructeur de cette classe reçoit en entrée le endpoint websocket. Nous écouterons sur le champ "/ws".

Code Web_server.ccp est comme suit :

```
   AsyncWebServer server(80);
   AsyncWebSocket ws("/ws");
```

La bibliothèque <ESPAsyncWebServer> nous permet de configurer les routes où le serveur sera à l'écoute des requêtes HTTP entrantes et d'exécuter des fonctions (handleStatus, handleConfig, handleLastValues, handleNotFound) lorsqu'une requête est reçue sur cette route. Pour cela, nous utilisons la méthode ‘‘on’’ sur l'objet serveur comme suit :

Code Web_server.ccp est comme suit :

```
void web_server_setup() {
    server.on("/status", handleStatus);
    server.on("/config", handleConfig);
    server.on("/lastvalues", handleLastValues);
    server.onNotFound(handleNotFound);
    server.begin();
    ws.onEvent(onWsEvent);
    server.addHandler(&ws);}
```

La fonction handleStatus renvoie une réponse sous format textuelle plus précisément JSON contenant les états des relais (true ou false) puis mettre le code de statut de réponse HTTP à 200 pour indiquer la réussite de la requête, comme le montre le code suivant :

Code Web_server.ccp est comme suit :

```
void handleStatus(AsyncWebServerRequest * request) {
    AsyncResponseStream *response;
    if(false == requestPreProcess(request,response)) {
        return;}
String s = "{";
    s += "\"relay_1_state\":" + String (relay_1_state ? "false":"true") + ",";
    s += "\"relay_2_state\":" + String (relay_2_state ? "false":"true") + ",";
    s += "\"relay_3_state\":" + String (relay_3_state ? "false":"true") + ",";
    s += "\"relay_4_state\":" + String (relay_4_state ? "false":"true") + "";
    s += "}";
    response->setCode(200);
    response->print(s);
    request->send(response);}
```

Le même principe pour les fonctions handleLastValue, handleConfig et handleNotFound.

Code Web_server.ccp est comme suit :

```
void onWsEvent
```

Il faut mentionner que la fonction onWsEvent est une fonction qui est directement implémenté à partir de la bibliothèque <ESPAsyncWebServer.h>. Nous avons ajouté notre fonction handleRecievedMessage dans le corps de la fonction onWsEvent.

À l'envoie d'un message websocket au serveur, la fonction handleRelayMessage va faire la désérialisation du message qui est sous format JSON, comme le montre le code suivant :

Code Web_server.ccp est comme suit :

```
void handleRelayMessage(String msg) {
    DynamicJsonDocument doc (RELAY_MESSAGE_MEMORY_POOL);
    DeserializationError error = deserializeJson(doc, msg);
    if(error) {
   DBUG("[web_server] Websocket message deserialization failed with the
following error: ");
        DBUGLN(error.c_str());
        return;}
    uint8_t relay_id = (uint8_t) doc["id"];
    bool relay_state = (bool) doc["state"];
    const char* relay_name = doc["name"];
    relay_set_state(relay_id, relay_state);
    if (relay_latch_enabled) {
        config_save_relay_states();}}
```

a) src.ino

Il représente le programme principal qui doit exécuter par le microcontrôleur esp8266. Le fichier src.ino comporte deux fonctions principales :

- Setup () : consiste à charger la configuration et initialisation des relais, initialisation du wifi et démarrage de serveur web.
- loop () : consiste à faire clignoter les LED du esp 8266 afin d'indiquer la connexion a été établit avec le routeur.

Le code du programme principal est comme suivi :

Code src.ino est comme suit :

```
#include <Arduino.h>
#include "debug.h"
#include "wifi.h"
#include "relay.h"
#include "sensors.h"
#include "web_server.h"
#include "config.h"
void setup() { delay(2000);
    config_load_settings();
    wifi_setup();
    relay_setup();
    web_server_setup();
    delay(100);}
void loop () { wifi_loop(); }
```

3.3.3.2 L'implémentation du serveur récepteurs (capteurs)

L'implémentation du serveur récepteurs est la même l'implémentation du serveur actionneurs, la déférence entre les deux implémentations réside dans l'absence des relais et l'ajout des capteurs.

a) Sensors.ccp

Nous commencerons par inclure les bibliothèques qui nous permettrons de manipuler les capteurs température, humidité et présence en citant la bibliothèque <DallasTemperature.h>, <OneWire.h> et <DHT.h>. Puis définir les pins pour chaque capteur. Pour manipuler le capteur DS1820 nous utilisons la bibliothèque <DallasTemperature.h> et <OneWire.h>. Le capteur DHT22 nécessite la bibliothèque <DHT.h>. Comme le montre le code suivant :

Code Snesors.ccp est comme suit :

```
#include "DHT.h"
#include <OneWire.h>
#include <DallasTemperature.h>

DHT dht(DHTPIN, DHTTYPE);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature dallas(&oneWire);
void sensors_setup() {
```

```
    dht.begin();
    dallas.begin();}
void sensors_loop() {
    if(millis() - sensors_previous_millis >= sensors_poll_interval)
    {   strcpy(result, "");
        dht_temperature = dht.readTemperature();
        dht_humidity = dht.readHumidity();
        rip_motion = digitalRead(RIP_pin);
        dallas.requestTemperatures();
        dallas_temperature =dallas.getTempCByIndex(0);
        strcat(result, "T1:");
        dtostrf(dht_temperature, 2, 2, measurement);
        strcat(result, measurement);
        strcat(result, ",H:");
        dtostrf(dht_humidity, 2, 2, measurement);
        strcat(result, measurement);
        strcat(result, ",T2:");
        dtostrf(dallas_temperature, 2, 2, measurement);
        strcat(result, measurement);
        strcat(result, ",Rip:");
        dtostrf(rip_motion, 2, 2, measurement);
        strcat(result, measurement);
        lastvalues = result;
        sensors_previous_millis = millis();}}
```

a) Web_server.ccp

Lors de l'envois d'une requête type GET au service RESTFULL (192.186.1.7/status), la fonction handleStatus renvoie une réponse sous format textuelle plus précisément JSON contenant les données lues à partir des capteurs (température, humidité et présence) puis mettre le code de statut de réponse HTTP à 200 pour indiquer la réussite de la requête, comme le montre le code suivant :

Code Web_server.ccp est comme suit :

```
void handleStatus(AsyncWebServerRequest * request) {
    AsyncResponseStream *response;
    if(false == requestPreProcess(request,response)) {
        return;
}
    String s = "{";
    s += "\"dht_temperature\":\"" + String(dht_temperature) + "\",";
    s += "\"dht_humidity\":\"" + String(dht_humidity) + "\",";
    s += "\"rip_motion\":\"" + String(rip_motion) + "\",";
    s += "\"db_temperature\":\"" + String(dallas_temperature)+ "\"";
    s += "}";
    response->setCode(200);
```

```
    response->print(s);
    request->send(response);}
void web_server_setup() {
    server.on("/status_sensors", handleStatus);
    server.onNotFound(handleNotFound);
    server.begin();
}
```

3.3.4 L'implémentation de l'application

Dans ce travail, nous utiliserons javascript knockout il a été conçu pour développer des applications web dynamiques, qui offrent des caractéristiques similaires aux logiciels traditionnels installés sur un ordinateur. La dimension interactive et la vitesse d'exécution sont particulièrement soignées dans ces applications Web.

Cela rend très simple la mise en œuvre d'une interface utilisateur complexe, de plus de sa légèreté cela rend pratique pour les systèmes embarqués comme l'ESP8266.

Javascript knockout est dédié pour l'architecture MVVM (Model–View–Viewmodel), il a été conçu de manière à utiliser des Object javascript comme ViewModel.

Javascript knockout est développé sur trois principaux corps (figure 75) :

FIGURE 75 : CONCEPTS DE BASE KNOCKOUT [11]

- Dependency tracking avec Observable : si une propriété est chargée ou modifiée, elle avertit automatiquement l'interface utilisateur de la lier à un ou plusieurs endroits pendant que les modifications ont été apportées. L'interface utilisateur reflétera les modifications et aura également une option indiquant si l'interface utilisateur peut modifier la valeur pour mettre à jour automatiquement l'objet source à nouveau. [11]
- Declarative bindings : Ce concept permet de connecter des parties de l'interface utilisateur au modèle de données de manière simple et pratique. C'est là que les objets sources sont liés aux éléments cibles via le code HTML lui-même. [11]
- Templating : Les structures répétitives d'une page Web, telles que les lignes ou les zones de liste, peuvent être créées à l'aide de modèles. [11]

a) Le modèle de vue

La première étape consiste a créé le principale modèle vue (View Model), puis activer knockout js en utilisant une fonction jQuerry. Nous allons créer une variable nommé « vm » qui va représenter le MainView, puis nous allons appliquer le data bidings en utilisant la fonction ko.applyBindings () et mettre « vm » en argument comme le montre le code suivant **:**

Code est comme suit :

```
function MainViewModel() {
var self = this;}
```

```
$(function() {
  var vm = new MainViewModel();
  ko.applyBindings(vm);
  vm.start();});
```

Après avoir définir la structure de base de l'application js knockout, nous allons définir la fonction BaseViewModel () qui vont être utiliser pour implémenter StatusViewModel () et ConfigViewModel ().
La fonction BaseViewModel () va prendre en argument trois paramètre :

1. Default: variable par defaults.
2. Mapping : créer un mappage à partir de JSON vers un observable ViewModel.
3. RemoteURL : lien de la ressource.

Code est comme suit :

```
function BaseViewModel(defaults, remoteUrl, mappings) {
  if (mappings === undefined) {
    mappings = {};
  }
  var self = this;
  self.remoteUrl = remoteUrl;

  ko.mapping.fromJS(defaults, mappings, self);
  self.fetching = ko.observable(false);
}
```

Si la variable « mappings » en entrée est indéfinie, elle recevra un tableau vide. Puis, nous appliquerons le mapping en utilisant la fonction ko.mapping.fromJS (). À la fin, nous ajouterons la variable « fetching ». Quand nous fessons des requêtes AJAX de type « GET », « fetching » recevra « true » sinon elle recevra la valeur « false ».

En utilisant Prototypes Objet qui est un mécanisme au sein de JavaScript qui permettent aux objets JavaScript d'hériter des propriétés d'autres objets. Nous allons implémenter la méthode update () qui recevra un argument « after » qui est une fonction de rappel (callback) qui s'exécutera après la fonction update (). La fonction update () va exécuter une requête AJAX en utilisant RemoteURL passé en paramètre, puis appliquer le mapping des données reçues. À la fin de la requête AJAX, une fonction de rappel (callback) va s'exécuter pour mettre le « fetching » en « false » et exécuterla fonction after () comme le montre le code suivant :

Code est comme suit :

```
BaseViewModel.prototype.update = function(after) {
  if (after == undefined) {
    after = function () {}; }
  var self = this;
  self.fetching(true);
  $.get(self.remoteUrl, function(data) {
    ko.mapping.fromJS(data, self);
```

```
  }, 'json').always(function(){
    self.fetching(false);
    after(); });};
```

Implémentation de la fonction StatusViewModel () va se faire par l'appel de la fonction BaseVeiwModel () qui fa prendre en paramètre l'URL de la ressource Status qui est un web service RESTfull développer précédemment et les états par défaut des relais. Puis, nous utilisons la méthode object.create () pour créer un nouveau objet de type BaseViewModel comme le montre le code suivant :

Code est comme suit :

```
function StatusViewModel() {
  var self = this;
  BaseViewModel.call(self, {
    "relay_1_state": false,
    "relay_2_state": false,
    "relay_3_state": false,
    "relay_4_state": false
  }, 'http://192.168.1.8' + '/status');}
StatusViewModel.prototype = Object.create(BaseViewModel.prototype);
StatusViewModel.prototype.constructor = StatusViewModel;
```

Nous procèderons de la même manière pour StatusViewModel2 () et ConfigViewModel () comme le montre le code suivant :

Code est comme suit :

```
function StatusViewModel2() {
  var self = this;
  BaseViewModel.call(self, {
    "dht_temperature": "",
    "dht_humidity": "",
    "rip_motion": "",
    "db_temperature": ""
  },"http://192.168.1.7"+ '/status_sensors');

}
StatusViewModel2.prototype = Object.create(BaseViewModel.prototype);
StatusViewModel2.prototype.constructor = StatusViewModel2;
function ConfigViewModel() {
  var self = this;
  BaseViewModel.call(self, {
    "relay_1_name": "DOOR",
    "relay_2_name": "A.C",
    "relay_3_name": "LIGHT 1",
    "relay_4_name": "LIGHT 2"
  }, "http://192.168.1.8" + '/config');
}
ConfigViewModel.prototype = Object.create(BaseViewModel.prototype);
ConfigViewModel.prototype.constructor = ConfigViewModel;
```

Dans la fonction ViewModel développer précédemment, nous allons créer la variable « Status » qui représente StatusViewModel et « Config » qui représente ConfigViewModel comme le montre le code suivant :

Code est comme suit :

```
function MainViewModel() {
  var self = this;
  self.status = new StatusViewModel();
  self.status2 = new StatusViewModel2();
  self.config = new ConfigViewModel();}
```

La fonction start () est la plus importante, elle permet d'initialiser toute l'application. La fonction updating va être mis à « true », pour qu'elle puisse utiliser l'utilité « obsevable » qui va faire la mise à jour a toute l'application.

Les fonctions config, status et status2 utilisent la méthode update () de BaseViewModel () développer précédemment.
La méthode update () comporte un argument after () qui est une fonction de rappel (callback) qui va être exécuter après self.config.update (), puis elle va appeler une nouvelle fonction status.update () et de même pour status2, comme le montre le code suivant :

Code est comme suit :

```
  self.start = function() {
  self.updating(true);
  self.config.update(function() {
    self.status.update(function() {
      self.status2.update(function() {
          self.init(function() {
              self.connect();
              self.initialised(true);
              updateTimer = setTimeout(self.update, updateTime);
              self.updating(false); }); }); });});};
```

La fonction update () permet de faire la mise à jour instantané de l'application chaque second, comme l'indique le code suivant :

Code est comme suit :

```
self.update = function() {
  if (self.updating()) {
      return; }
  self.updating(true);
  if (null !== updateTimer) {
      clearTimeout(updateTimer);
      updateTimer = null;}
  self.status.update(function() {
      self.status2.update(function() {
          self.refreshRelayStates();}); });
```

Le code de la partie de la collaboration des objets sera déployée au niveau de la fonction update () pour assurer que son exécution se fait après la mise à jour de données. Le code consiste à lire les données des capteurs émit depuis le web service Restfull du serveur des capteurs et d'envoyer un message web socket au serveur actionneur selon une condition définit, comme l'indique le code suivant :

Code est comme suit :

```
if(get_Alarme() == "true"){
      if (self.status2.rip_motion() == 1){
          var ws = new WebSocket('ws://192.168.1.8/ws');
          ws.onmessage = function(e) {
       ws.send(JSON.stringify({type:10, id:1, name: "Relay 1",state: false}));
              console.log(e.data);  }; } }
  if (self.status2.dht_temperature() >= get_temperature()){
      var ws = new WebSocket('ws://192.168.1.8/ws');
      ws.onmessage = function(e) {
      ws.send(JSON.stringify({type:10, id:2, name: "Relay 2",state: false}));
          console.log(e.data);  };}};
```

a) La vue (View)

Après avoir fini l'implémentation du ViewModel, nous procéderons maintenant à l'implémentation de la vue (View), qui représente un simple document HTML avec des liaisons déclaratives (Declarative Bindings), comme le montre le code suivant :

Code est comme suit :

```
<p class="sensor">
  <i class="fas fa-thermometer-full" style="color:#f57608"></i>
    <span class="sensor-labels">DHT Temperature</span>
     <span data-bind="text: status2.dht_temperature()"></span>
        <sup class="units">°C</sup>
 </p>
<hr>
 <p class="sensor">
  <i class="fas fa-tint" style="color:#5bc0de"></i>
    <span class="sensor-labels">DHT Humidity</span>
      <span data-bind="text: status2.dht_humidity()"></span>
         <sup class="units">%</sup> </p>
```

3.4 Tests et résultats

3.4.1 L'interface graphique ShareIot

Lorsque l'utilisateur accède à l'application, il sera dirigé vers la page principale. (Figure 76)

FIGURE 76: INTERFACE GRAPHIQUE AVANT AUTHENTIFICATION

L'utilisateur devra créer un compte pour accéder à la plateforme en remplissant le formulaire de création de compte. (Figure 77)

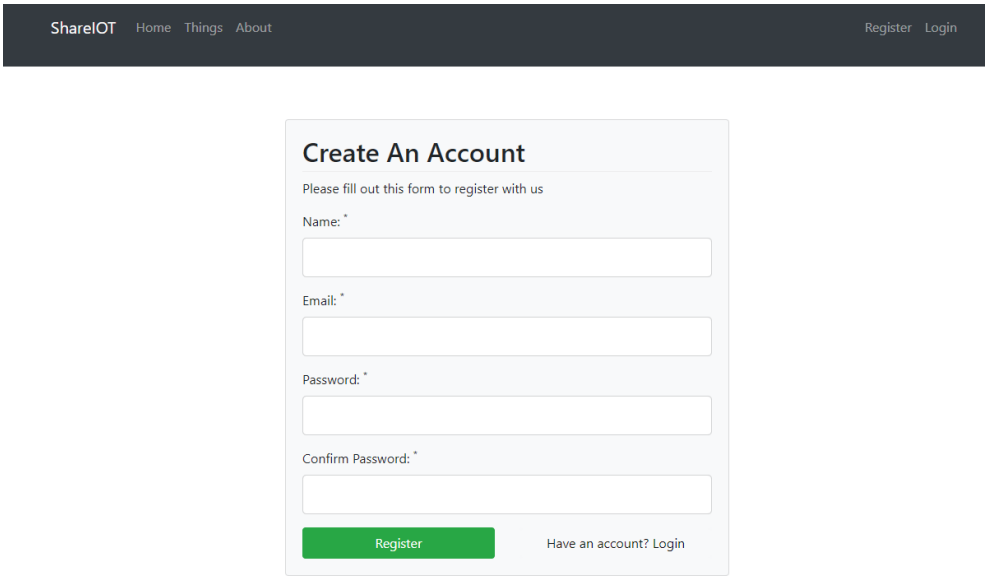


FIGURE 77: FORMULAIRE DE CRÉATION DE COMPTE

L'utilisateur sera par la suite dirigé à la page d'authentification. (Figure 78)

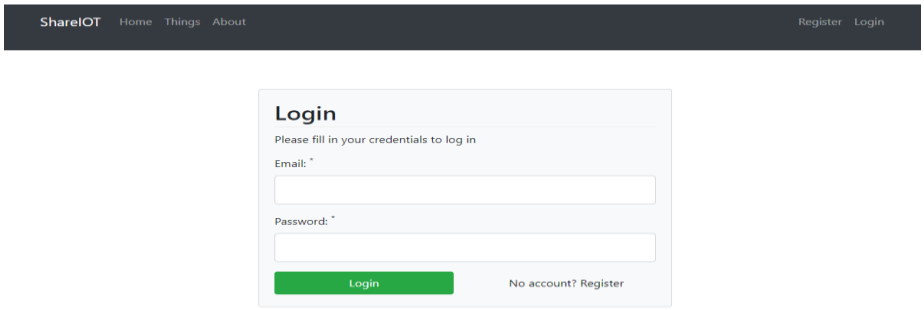


FIGURE 78: PAGE D'AUTHENTIFICATION

Après l'authentification, l'utilisateur peut naviguer jusqu'à l'onglet « Things » pour ajouter les objets en cliquant sur le buttons « add thing ». (Figure 79)

FIGURE 79: INTERFACE THINGS

L'utilisateur devra saisir les informations concernant l'objet. (Figure 80)

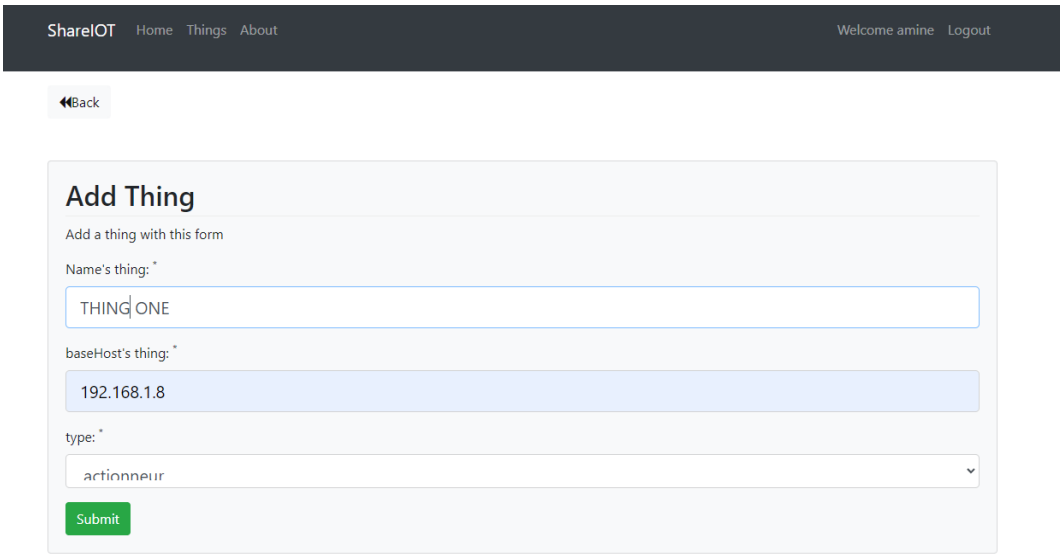


FIGURE 80: FORMULAIRE AJOUTER OBJET

L'objet ajouter sera afficher immédiatement afficher dans l'interface « Things ». (Figure 81)

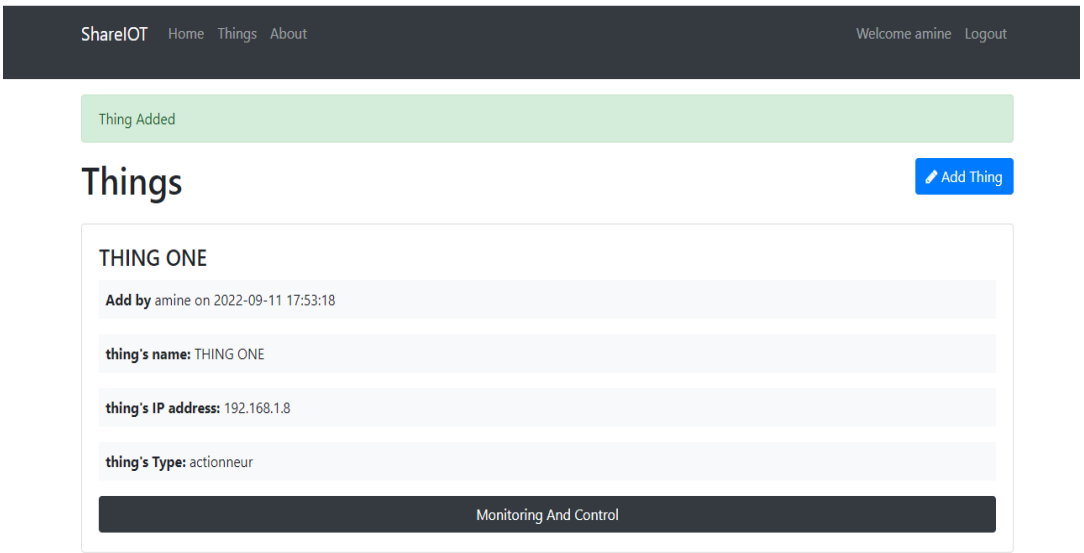


FIGURE 81 : LA LISTE DES OBJETS AJOUTÉS

En cliquant sur « monitoring and control » de l'objet souhaité, l'utilisateur peut alors consulter et contrôler objet. De plus, il aura la possibilité de modifier ou supprimer l'objet. (Figure 82, 83, 84)

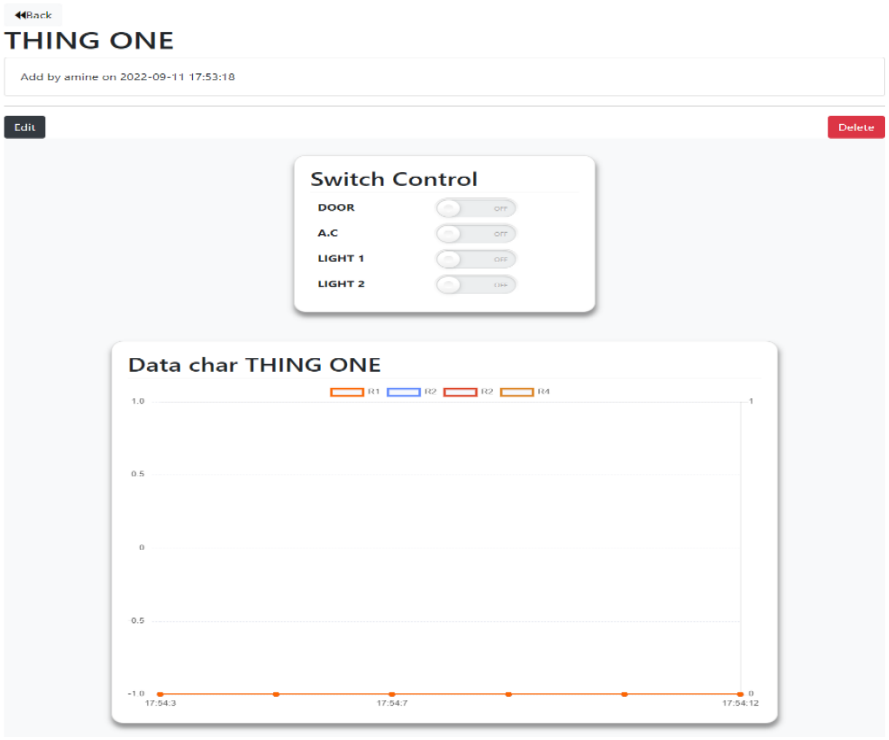


FIGURE 82: CONSULTATION ET CONTRÔLE OBJET ACTIONNEUR

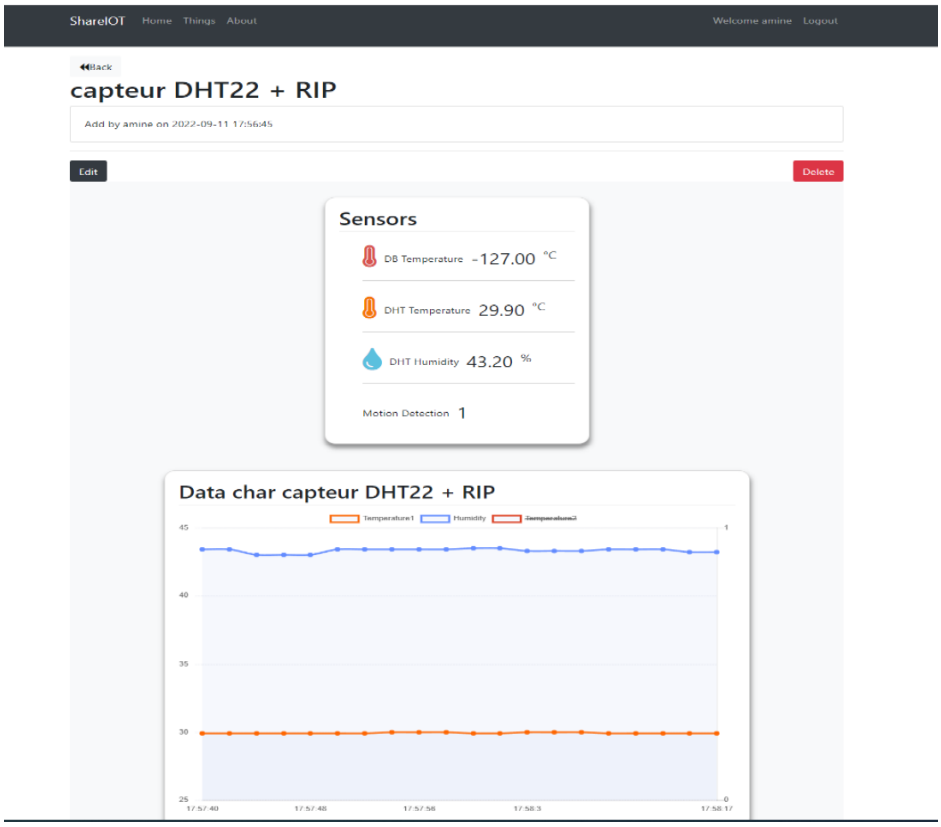


FIGURE 83: CONSULTATION ET CONTRÔLE OBJET CAPTEUR

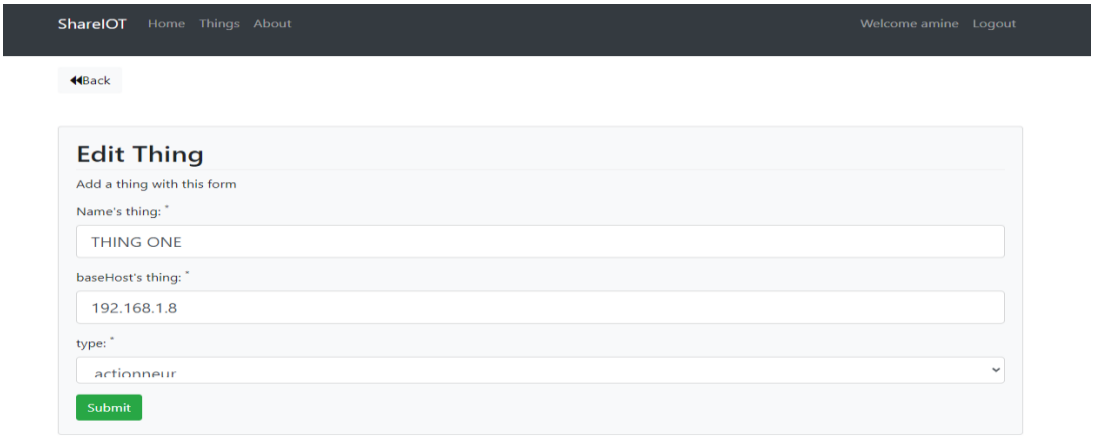


FIGURE 84 : MODIFICATION OBJET

L'utilisateur a la possibilité d'ajouter une collaboration entre deux objets en remplissant le formulaire de la collaboration comme le montre la figure 85.

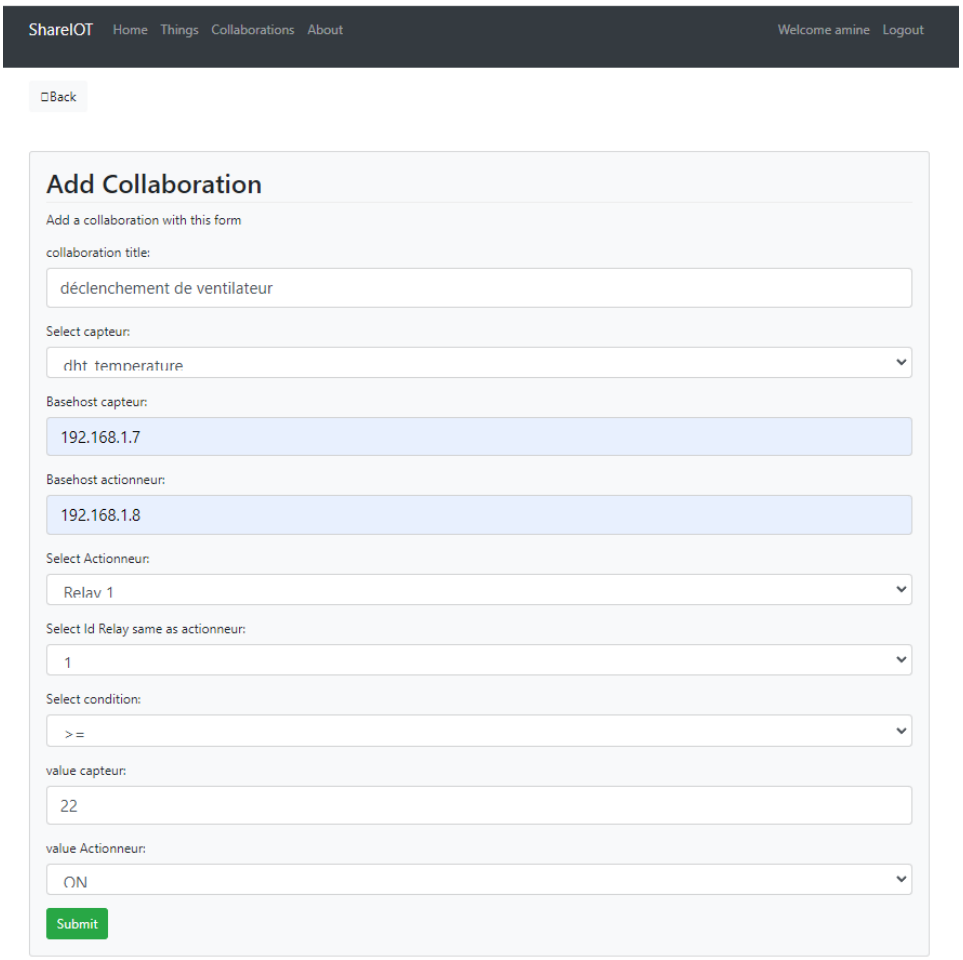


FIGURE 85 : AJOUTER COLLABORATION

L'utilisateur a aussi la possibilité de poster des messages et les problèmes rencontrés concernant les objets sur la plateforme. (Figure 86)

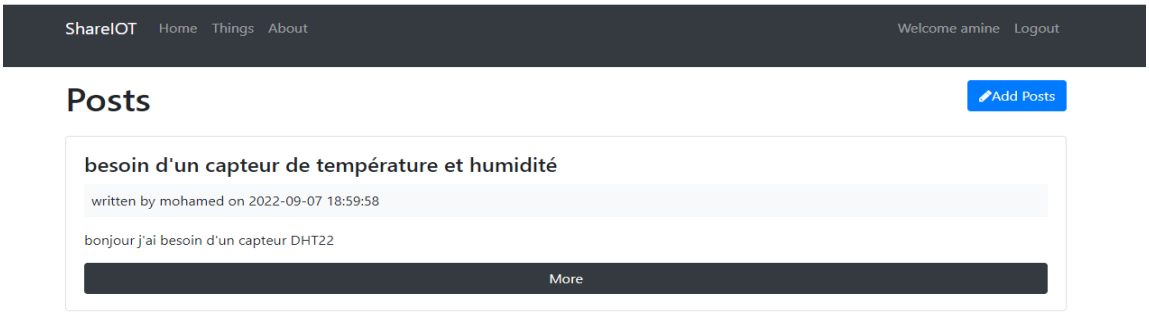


FIGURE 86 :INTERFACE POSTS MESSAGES

3.4.2 Scénario 1 : automatisation du process déclenchement climatisation

Dans le deuxième scénario, nous paramétrons notre température idéale dans le champ AC « set température ». Ce paramètre fait la comparaison entre la température actuelle de l'environnement et la température paramétré, si la température paramétrée est inférieure de la température actuelle, elle impliquera le déclenchement de la climatisation, comme le montre la figure 87 :

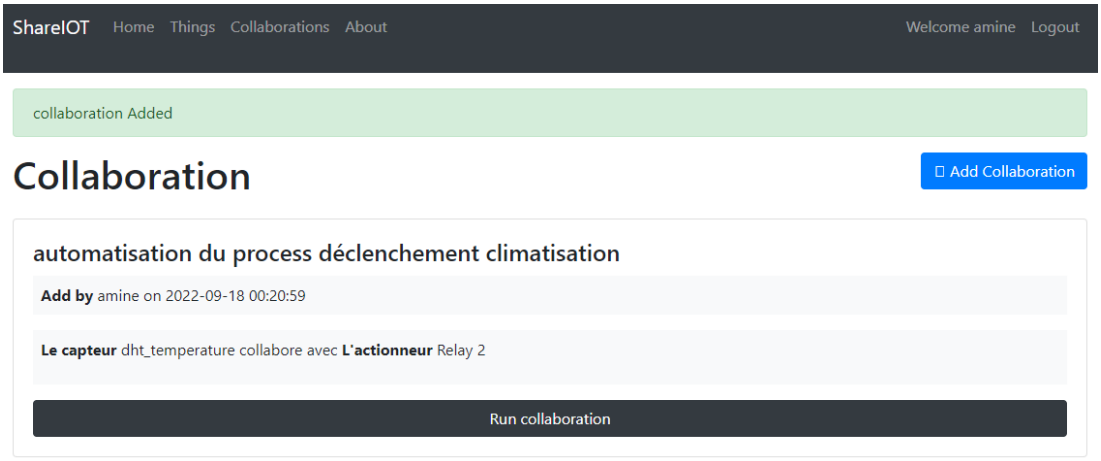


FIGURE 87 : INTERFACE GRAPHIQUE CORRESPONDANT AU SCÉNARIO 1

La figure 88 représente l'interface graphique en montrant la collaboration entre deux objets, elle est composée de deux compartiment à savoir :

- Le compartiment condition de la collaboration dans l'objectif principale est de montrer la condition de l'enclenchement de ventilateur.
- Le compartiment pour indiquer l'état de connectivité de ventilateur.

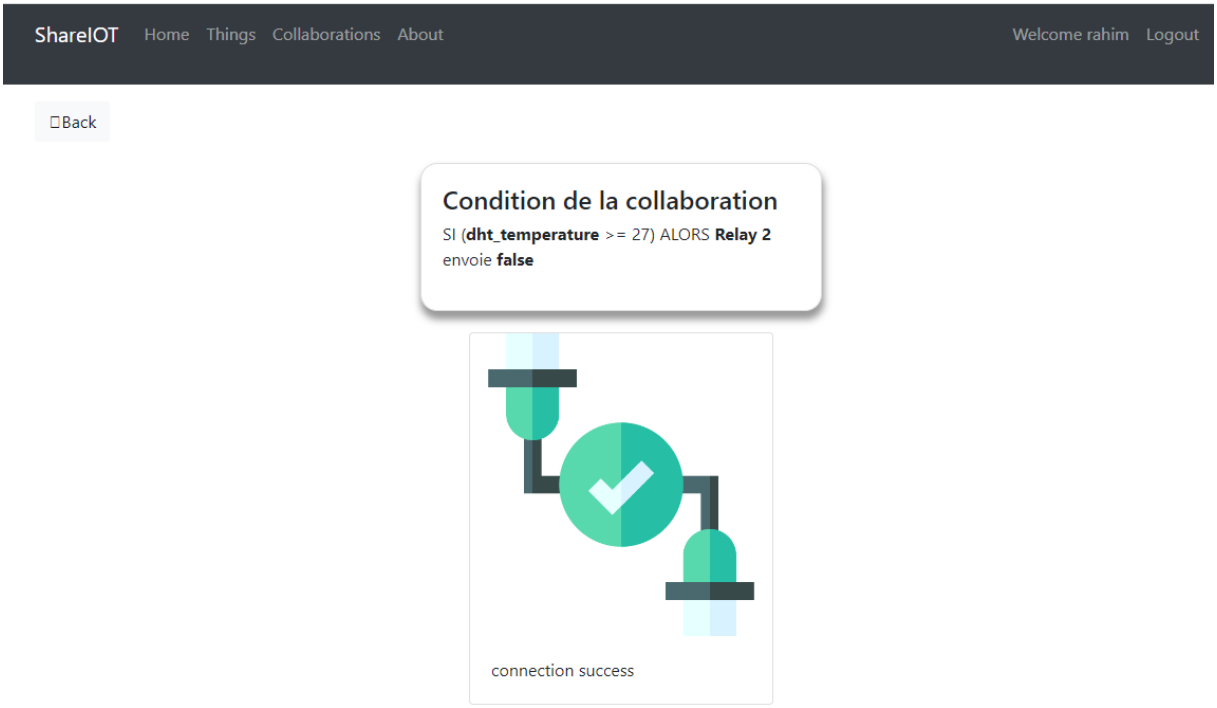


FIGURE 88 : RÉSULTAT DE SCENARIO 1 COTE APPLICATION

3.4.3 Scénario 2 : automatisation du process verrouillage porte

Ce scénario consiste de verrouiller la porte lors de la détection de mouvement, cette fonctionnalité peut être activer/ désactive. La figure 89 montre l'exécution du dernier scénario :

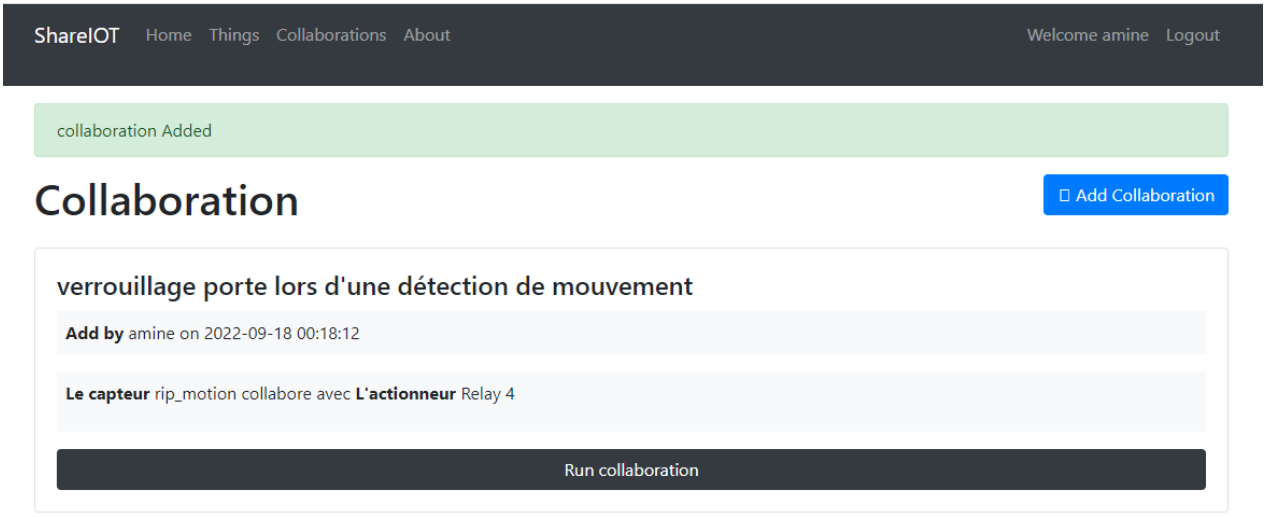


FIGURE 89 : L'INTERFACE GRAPHIQUE DE SCÉNARIO 2

La figure 90 représente l'interface graphique en montrant la collaboration entre deux objets, elle est composée de deux compartiment à savoir :

- Le compartiment condition de la collaboration dans l'objectif principale est de montrer la condition de l'ouverture de la porte.
- Le compartiment pour indiquer l'état de connectivité de la porte.

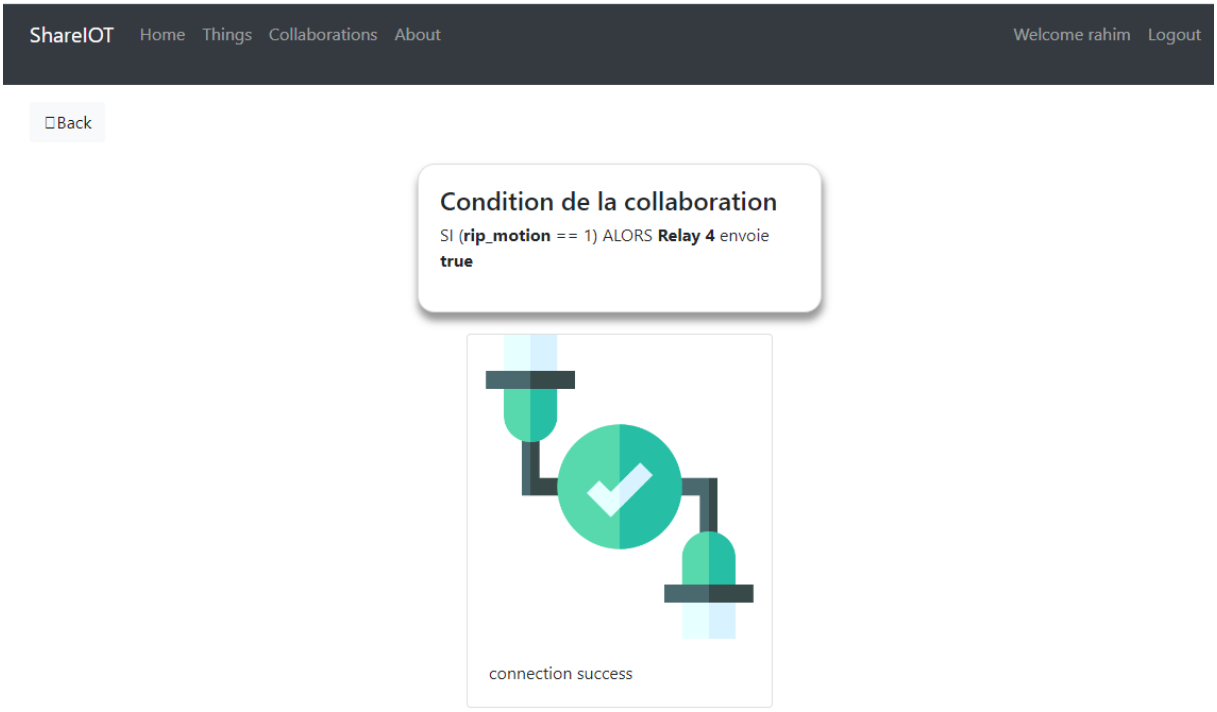


FIGURE 90 : RÉSULTAT DE SCENARIO 2 COTE APPLICATION

Conclusion

Dans ce chapitre, nous avons présenté les notions essentielles relatives à la réalisation de notre projet, les outils et les logiciels nécessaires à la réalisation. Nous avons expliqué les concepts de base pour créer un web service au niveau des objets. Puis, nous avons présenté l'interface graphique de notre application et pour finir, nous avons testé deux scénarios dans le but de faire une collaboration dans l'internet des objets via un web service. Les tests ont abouti à des résultats satisfaisants.

Conclusion générale et perspectives

Dans le monde actuel, que ce soit dans les secteurs socioéconomiques ou les organisations publiques, l'utilisation de l'internet des objets est devenue un accélérateur favorable d'innovation. Il s'agit de connecter les objets entre eux pour satisfaire un besoin spécifique dans un domaine précis. Pour assurer que les objets collaborent et fonctionnent convenablement, il faut respecter des normes du web. Cette problématique devient une question centrale qui incombe aux métiers des développeurs et des producteurs.

Notre travail a pour but de résoudre le problème de la collaboration des objets entre eux dans le domaine de l'internet des objets, en proposant une application qui permet de faire la collaboration dans l'internet des objets en utilisant les web service comme un moyen pour normaliser et de simplifier les développements d'application IOT en suivant le paradigme bien connu et réussit du web.

Après avoir défini ce qu'est l'IOT et le web service, nous avons fait un premier pas vers la collaboration des objets via les web services et la découverte des techniques de mise en œuvre.

Nous avons fait dérouler et réaliser le projet demandé en plusieurs étapes, à savoir :

- **Examiner les cas proposés et identifier les travaux préliminaires à réaliser**
  Dans cette phase, nous avons pris connaissance et étudier les conditions de la réalisation du projet. Il est apparu qu'il fallait :
  - Disposer d'un matériel (hardware) adéquat c'est-à-dire un microcontrôleur adapter, des capteurs, des lampes et des relais.
  - Télécharger les « software » nécessaires à la réalisation du projet, tels que « MySQL », « Apache », « VScode », « Arduino IDE ».
- **Préparer et prendre connaissance des outils de travail**
  Après avoir téléchargé les outils, nous avons pris une grande partie de notre temps à nous familiariser aux fonctions offertes par les outils en consultant les documents techniques ainsi que le blog dédié à ces outils pour trouver des réponses aux difficultés rencontrées.
- **Modélisation de l'application et des objets**
  Une modélisation a été effectué pour l'application et les objets concernant le projet.
- **Examiner les erreurs afin de les résoudre**
  Durant la réalisation du projet, nous avons rencontré plusieurs problèmes à savoir :
  - L'incompatibilité entre les bibliothèques (certaines bibliothèques n'étaient pas mises à jour).
  - Difficulté dans l'assemblage du prototype.
- **Charger et visualiser les résultats**
  Après la réalisation de toutes les étapes du projet, des tests ont été réalisés sur deux scénarios. Ces résultats étaient concluants.

Lors du développement de ce projet, nous avons acquis certaines compétences à savoir la modélisation application avec une architecture de style MVVM, la modélisation des objets connectés et la programmation embarquée.
S'il n'y avait pas la contrainte de temps, il serait intéressant de continuer le projet en développant :

- Le coté administrateur.
- Un stockage de données dans le cloud pour une utilisation en ligne et immédiate.
- Une application qui offre un paramétrage de collaboration plus détaillé.

Nous pouvons dire à la fin que ce travail nous a permis d'appliquer des concepts théoriques sur des cas pratiques.

Références bibliographiques et webographiques

[1] *L Srivastava, T Kelly - International Telecommunication Union, Tech. Rep, 2005 - itu.int pp 3*

[2] Asadullah, M., & Raza, A. (2016). An overview of home automation systems. 2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI). doi:10.1109/icrai.2016.7791223

[3] Fensel, D., & Bussler, C, The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications, 2002, 1(2), 113–137. doi:10.1016/s1567-4223(02)00015-7

[4] Babin, Gilbert & Leblanc, Michel, Les Web Services et leur impact sur le commerce B2B, 2003, CIRANO, CIRANO Burgundy Reports.

[5] S. Patni, Pro RESTful APIs. Berkeley, CA: Apress, 2017. consulté le 11/03/2022. Available : https://doi.org/10.1007/978-1-4842-2665-0

[6] L. Richardson and M. Amundsen, *RESTful Web APIs*. Beijing; Cambridge; Farnham; Köln; Sebastopol; Tokyo: O'reilly, -05-22, 2015.

[7] A. Tanenbaum, Systèmes d'exploitation : systèmes centralisés et systèmes distribués. Interéditions, 1994, Paris.

[8] G. Coulouris, J. Dollimore, and T. Kindberg, Distributed Systems—Concepts and Design, 1994, 2nd Ed. Addison-Wesley Publishers Ltd.

[9] AlShahwan, Feda, and Klaus Moessner. "Providing soap web services and restful web services from mobile hosts. " Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on. IEEE, 2010.

[10] P. Gambarotto and W. Services, "Technologies pour Web Services faciles: REST, JSON Mots clefs." Available: https://2009.jres.org/soumission/papers/render/pdf/92.pdf consulté le 10/04/2022.

[11] V. Kumar, Beginning Windows 8 data development: using C# and JavaScript. Berkeley: Ca, 2013.

[12] M. Chouiten, "Architecture distribuée dédiée aux applications de Réalité Augmentée mobile." Available: https://tel.archives-ouvertes.fr/tel-00903538/document consulté le 09/03/2022.

[13] Coutaz, Joëlle & Crowley, James. Plan « intelligence ambiante » : défis et opportunités Document de réflexion conjoint du comité d'experts « Informatique Ambiante » du département ST2I du CNRS et du Groupe de Travail « Intelligence Ambiante » du Groupe de Concertation Sectoriel (GCS3) du Ministère de l'Enseignement Supérieur et de la Recherche, DGRI A3 Version 1.2 finale - 14 octobre 2008.

[14] D. Gourley and B. Totty, *HTTP: the definitive guide*. Beijing; Sebastopol, Ca: O'reilly, 2002.

[15] L. Richardson and S. Ruby, *RESTful web services: [web services for the real world]*. Beijing: O'reilly, 2007.

[16] Jobinesh Purushothaman, *Restful java web services.* Packt Publishing Limited, 2015.

[17] Anura Gurugé, Web services : theory and practice. Burlington, Ma: Digital Press, 2004.

[18] http://www.w3.org consulté le 11/02/2022.

[19] *Kafle, V. P., Fukushima, Y., & Harai, H. (2016). Internet of things standardization in ITU and prospective networking technologies. IEEE Communications Magazine, 54(9), pp 43–49. doi:10.1109/mcom.2016.7565271*

[20] *ITU-T, "Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks", ITU, 2012, pp. 2-9.*

[21] RFID for the Supply Chain and Operations Professional, 3rd Edition by Dr. Pamela J. Zelbst, Dr. Victor E. Sower pages 7.

[22] Perry Lea. Internet of Things for Architects Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security, pages 11-33.

[23] UIT-T Y.2060 secteur de la normalisation des télécommunications de l'uit (06/2012) série y : infrastructure mondiale de l'information, protocole internet et réseaux de prochaine génération pages 1-2.

[24] Recommendation ITU-T Y.2066 (2014), "Common Requirements of the Internet of Things."

[25] Zur Muehlen, Michael, Jeffrey V. Nickerson, and Keith D. Swenson. "Developing web services choreography standards—the case of REST vs. SOAP." Decision Support Systems 40, no. 1 (2005): 9-29.
[26] W. Xu, "Modeling and exploiting the knowledge base of web of things," *tel.archives-ouvertes.fr*, Jan. 16, 2015. https://tel.archives-ouvertes.fr/tel-01178286.

[27] D. D. Guinard and Vlad Trifa, *Building the web of things: with examples in Node.js and Raspberry Pi*. Shelter Island, Ny: Manning, 2016.

[28] M. Thiyagarajan and C. Raveendra, "Role of web service in Internet of Things," 2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2017, pp. 268-270, doi: 10.1109/ICATCCT.2017.8389146.

[29] N. S. Bouchaïb Radi, "Probabilistic study of an embedded system," *https://www.openscience.fr*, Feb. 09, 2017. http://openscience.fr/Probabilistic-study-of-an-embedded-system.

[30] Mike Stowe, Undisturbed REST, First-look Edition. 77 Geary Street, Suite 400, San Francisco: MuleSoft, 2015 consulté le 10/03/2022. Available: http://www.mulesoft.com/restbook

[31] Méziane Boudellal, *Smart Home : Habitat connecté, installations domotiques et multimédia*. Paris, France: Dunod, 2014.

[32] M. E. A. Sehili, "Reconnaissance des sons de l'environnement dans un contexte domotique," *tel.archives-ouvertes.fr*, Jul. 05, 2013. https://tel.archives-ouvertes.fr/tel-00950751 consulté le 02/05/2022.

[33] P. C. Quispe, "Contrôle intelligent de la domotique à partir d'informations temporelles multi sources imprécises et incertaines," *tel.archives-ouvertes.fr*, Mar. 27, 2013. https://tel.archives-ouvertes.fr/tel-00957941/

[34] "World Wide Web Consortium (W3C)," W3.org, 2019. https://www.w3.org

[35] R. Daigneau, Service design patterns: Fundamental design solutions for SOAP/WSDL and restful web services. Upper Saddle River, NJ: Addison-Wesley, 2012.

[36] R. Nagappan, Developing Java Web Services. New York: John Wiley & Sons, Ltd., 2003.

[37] T. Tiendrebeogo, "A Scalable Architecture for Secured Access to Distributed Services," ICST Transactions on Scalable Information Systems, vol. 4, no. 13, p. 152751, Jun. 2017, doi: 10.4108/eai.28-6-2017.152751.

[38] "Les services Web Par SoftDeath," 2011 consulté le 09/04/2022. Available: http://user.oc-static.com/pdf/203276-les-services-web.pdf

[39] "Differences SOAP vs REST: Comparison of protocols and their security," www.wallarm.com. https://www.wallarm.com/what/differences-soap-vs-rest

[40] PHP: What is PHP? - Manual.PHP: Hypertext Preprocessor. https://www.php.net/manual/en/intro-whatis.php

[41] C++ Introduction. W3Schools Online Web Tutorials. https://www.w3schools.com/cpp/cpp_intro.asp

[42] What is JavaScript? - Learn web development | MDN. (s. d.). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[43] Knockout: Introduction. Knockout: Home. https://knockoutjs.com/documentation/introduction.html

[44] Arduino IDE 2 Tutorials | Arduino Documentation | Arduino Documentation. (S. d.). Arduino Docs | Arduino Documentation | Arduino Documentation. https://docs.arduino.cc/software/

[45] Microsoft. (2021, 3 novembre). Visual Studio Code - Code Editing. Redefined. Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/

[46] Introduction | Postman Learning Center. Postman Learning Center. https://learning.postman.com/docs/getting-started/introduction/

[47] Welcome! - The Apache HTTP Server Project. Welcome! - The Apache HTTP Server Project. https://httpd.apache.org/

[48] MySQL: MySQL 8.0 Reference Manual: 1.2.1 What is MySQL? MySQL: Developer Zone. https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html

[49] Get started with Bootstrap. Bootstrap · The most popular HTML, CSS, and JS library in the world. https://getbootstrap.com/docs/5.2/getting-started/introduction/

[50] HTML &; CSS - W3C. World Wide Web Consortium https://www.w3.org/standards/webdesign/htmlcss

[51] Perry Lea. Internet of Things for Architects Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security, pp 11-33

[52] Web of Things (WoT): Use Cases and Requirements. World Wide Web Consortium (W3C). https://www.w3.org/TR/wot-usecases/#smart-agriculture

[53] Internet of Things for Architects Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security by Perry Lea, pages 25-33.