REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE MOULOUD MAMMERI DE TIZI OUZOU

FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE DE FIN D'ETUDES

EN VUE DE L'OBTENTION DU DIPLOME DE MASTER EN INFORMATIQUE GENERALE

DOMAINE : ***MATHEMATIQUES ET INFORMATIQUE***

FILIERE : ***INFORMATIQUE***

SPECIALITE : ***CONDUITE DE PROJETS INFORMATIQUES***

THEME :

Développement d'une solution à base de Websocket pour la domotique.

Encadré par : M. DAOUI Mahemmed

Réalisé par : M. BENDOU Nassim

PROMOTION : 2016 / 2017

Remerciements

D'abord, je remercie le bon Dieu pour le courage et la force qu'il m'a donné afin d'accomplir ce travail dans lequel j'ai affronté énormément d'obstacles. Je tiens également à remercier vivement mon encadreur M. DAOUI Mahemmed pour son soutien et sa disponibilité tout au long du projet. Je n'oublie pas de remercier M.DERGUINI Noureddine, attaché de recherche au CDTA, qui m'a proposé ce thème et m'a accueilli chaleureusement au CDTA. Je remercie également ses collègues M. TEBERKAK Allal, enseignant à l'université de Media, et M. BONCEUR Ahcene enseignant à l'université de Brest en France, pour leurs conseils précieux.

Enfin, je remercie tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet.

Dédicaces

Je dédie cet humble travail à mes parents auxquels je dois tout, à mes frères Farid et Mounir, à ma sœur Sarah, au reste de ma famille, et à tous mes amis.

Sommaire

Liste de figures

Introduction générale

L'informatique, de nos jours, est devenue une entité omniprésente dans notre vie quotidienne, et les gens sont devenus de plus en plus connectés aux réseaux informatiques. Cela a fait réfléchir les acteurs du domaine afin de tirer profit de cette connectivité. En effet, ils ont conçu des systèmes offrant au grand public des fonctionnalités très originales, citons par exemple : les réseaux sociaux, le Cloud, l'internet des objets…etc.

Ce projet rentre dans le cadre de l'internet des objets, qui consiste principalement à contrôler des objets physiques via des plateformes logicielles (application mobile, web…etc.).

Nous avons supposé comme cas d'étude « la domotique », qui est une discipline consacrée à l'automatisation des installations de l'habitat, notamment en ce qui concerne le chauffage, les communications et la sécurité.

Notons que le domaine de la domotique est très intéressant et très prometteur. Néanmoins, la conception d'un modèle informatique pour un tel système n'est pas une chose aisée, et elle est contrainte de plusieurs facteurs : l'extensibilité, flexibilité, robustesse. Puisqu'on a affaire à des objets physiques, il faut absolument que le système soit un système temps réel. De ce fait, nous avons recherché les moyens matériels et logiciels les plus adéquats afin de concevoir et réaliser la solution au problème donné que nous allons éclaircir dans le reste de cette rédaction.

Ce mémoire est structuré au tour de trois chapitres, le premier est consacré à introduire les généralités qui se rapportent à la thématique du projet telle que la domotique, l'internet des objets …etc.

Le deuxième chapitre est dédié à l'analyse et la conception, dans lequel nous avons utilisé UML comme langage de modélisation.

Enfin, nous avons terminé avec le troisième chapitre intitulé « Réalisation » dans lequel nous avons cité les outils logiciels et matériels que nous avons utilisés afin de mettre le projet à terme. Nous avons également présenté la solution d'un point de vue d'implémentation, et enfin, nous avons donné une démonstration de l'application développée.

Chapitre 1:

Généralités

Introduction

Afin de mieux assimiler la solution proposée pour la domotique, nous allons d'abord définir les notions générales qui se rapportent à ce domaine, tel que : le Web, l'internet des objets, les systèmes embarqués, et la domotique.

1.1. HTTP et Websocket

1.1.1. Rappel sur l'HTML

HTML (HyperText Mark-Up Language) est un langage interprété, sert à créer des documents Web en offrant des balises permettant d'inclure tous types de contenu, des liens vers d'autres documents Web, formulaires, fichiers. Le HTML est consacré au fond des documents, par conséquent, il est souvent utilisé conjointement avec des feuilles de style en cascade (CSS) pour la mise en forme, et JavaScript afin de modifier dynamiquement le contenu du document du côté client.

Le langage HTML permet également la lecture de documents sur Internet à partir de machines différentes, grâce au protocole HTTP, permettant d'accéder via le réseau à des documents repérés par une adresse unique, appelée URL.

Un site Web est un ensemble cohérent de pages HTML. Par conséquent, le Web est un ensemble de sites web. [1]

1.1.2. L'évolution du Web

Il y'a encore quinze ans le web se restreignait à de simples pages HTML qui offraient du contenu statique et quelques liens pour surfer dans le site. Avec l'avènement des technologies, script CGI, PHP, JEE, ASP et AJAX, les applications web ont considérablement évolué et ont occupé une place plus importante sur Internet, mais aussi, et surtout dans le monde de l'entreprise.

En effet, il est devenu commun au travail d'utiliser des applications qui ne sont pas installées localement sur la machine, mais accessibles à distance via un navigateur.

Actuellement, on parle d'application web riche, RIA (Rich Internet Application), et certaines d'entre elles ont des besoins de communication temps réel, c'est-à-dire sans latence. Quelques exemples:

- Messageries instantanées.
- Réseaux sociaux.

- Applications financières.
- Jeux en ligne multijoueurs.

1.1.3. Inconvénients du HTTP

Une communication temps réel avec HTTP exige des mises à jour fréquentes, cela engendre une prolifération des requêtes et des réponses HTTP. La taille des entêtes HTTP dans certains cas peut être de 1 à 2 kilo-octets, une partie considérable de la bande passante sera alors consacrée aux entêtes. Nous allons insister ces propos par un exemple très simple:

On suppose un polling vers 1000 clients chacune seconde (1s). La taille moyenne des entêtes est de 1 Kilo-octet. Ceci consomme 1000*1000*2 = 2Mo/s de bande passante juste pour les entêtes. On déduit que le protocole HTTP n'est pas adéquat pour ce type de communication.

1.1.4. Le protocole WebSocket

WebSocket a été conçu pour faire face aux inconvénients du protocole HTTP, qui n'est plus adéquat aux applications temps réel.

WebSocket est un protocole full-duplex, ce qui veut dire que les voies montante et descendante peuvent être transmises simultanément.

WebSocket est une couche au-dessus de TCP qui est lui-même bidirectionnel, c'est donc du TCP pour le Web. Le format de l'URL des WebSockets ressemble au HTTP : ws://examples.com ou wss://examples.com.

Enfin même si l'objectif est de se débarrasser du HTTP, particulièrement pour les applications qui utilisent le protocole WebSocket, on a besoin du protocole HTTP pour la phase d'initialisation de la communication, ceci a pour objectif notamment d'assurer une compatibilité avec les infrastructures existantes (proxy, firewall...).

Cette phase d'initialisation se nomme le Handshake, elle est présentée ci-après.

1.1.4.1. Le handshake

Lorsqu'une page Web utilisant les WebSockets est demandée, une méthode HTTP est utilisée pour récupérer la page HTML, le style CSS et le JavaScript. Le JavaScript envoie alors une requête HTTP demandant la mise à niveau (upgrade) pour utiliser le protocole WebSocket, tout en échangeant quelques clés. Suite à un dialogue compliqué garantissant aux deux parties (client et serveur) une adoption complète du protocole WebSocket, le serveur envoie une réponse HTTP renvoyant plus de clés et confirmant l'établissement de la connexion sous le protocole WebSocket. Ce dialogue est géré d'une manière transparente par l'API JavaScript WebSocket, et une fois qu'il est terminé, la puissance totale du protocole est enclenchée. À ce stade (connexion établie sous le protocole WebSocket), il n'y a pas d'en-tête HTTP sur chaque paquet, les couts de protocole ne sont en fait que deux octets par paquet. Bien que l'adoption générale du protocole parmi les fournisseurs de navigateurs modernes ait été satisfaisante, des découvertes récentes ont révélé que le protocole WebSocket est vulnérable aux attaques. Adam Barth a démontré la possibilité d'attaques contre le protocole qui pourrait être utilisé par un attaquant pour empoisonner des caches qui se trouvent entre le navigateur et Internet. En conséquence, les principaux fournisseurs tels que Mozilla et Opera ont désactivé les fonctionnalités WebSockets par défaut pour protéger les utilisateurs jusqu'à ce que le protocole soit corrigé. Toutefois, le protocole a été corrigé. Ci-dessous, la figure1 résume le protocole WebSocket.

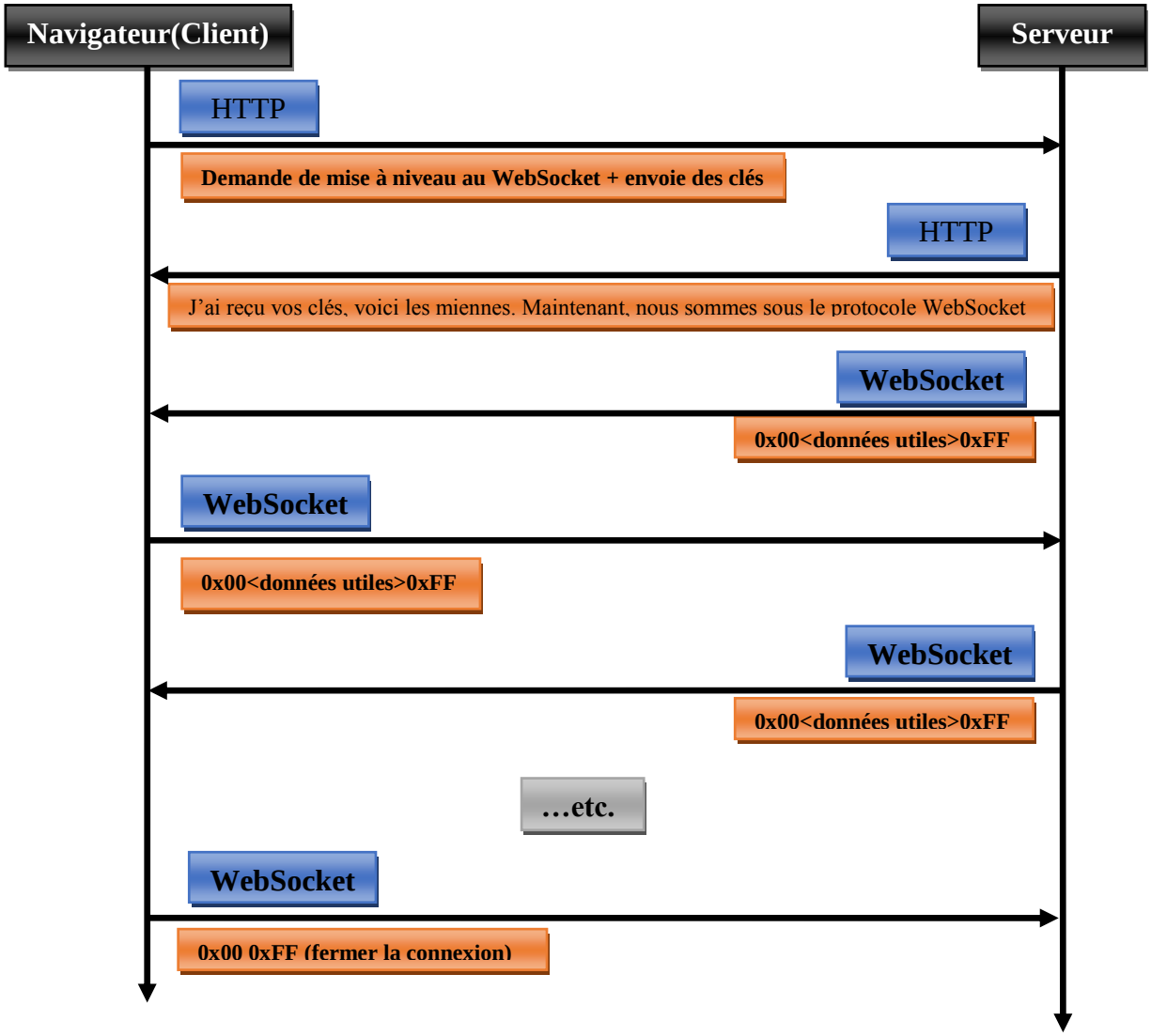


Figure 1 : Le protocole Websocket

1.1.4.2. La classe WebSocket

C'est la W3C qui gère l'API JavaScript WebSocket. La création d'une instance d'un objet WebSocket se fait comme suit :

var wSock = new Websocket(url,[protocol]);

Le paramètre URL représente l'URL de la WebSocket (ws://exemple.com:8080). C'est le serveur avec lequel on désire établir la connexion. [2]

Le paramètre protocole représente le sous-protocole WebSocket.

Une fois l'objet WebSocket est créé, nous aurons accès à ses méthodes. Parmi ces méthodes, il y a les évents handlers:

- **onopen**: lorsque la connexion est ouverte.
- **onmessage**: lorsqu'un message est reçu.
- **onerror**: lorsqu'une erreur est survenue.
- **onclose**: lorsque la connexion est fermée.

Les autres méthodes:

- **send**: envoyer des données.
- **close**: fermer la connexion.

Pour finir sur la description de l'API, voici un exemple d'utilisation:

```
var sock = new WebSocket ('ws://exemple.com:8080');

sock.onopen = function (evt) { alert ('La connexion est ouverte') ; };

sock.onmessage = function (evt) { alert ('Message reçu :' + evt.data) ;};

sock.onerror = function (evt) { alert ('Une erreur est survenue') ;};

sock.onclose = function (evt) {alert ('La connexion est fermée') ;}

sock.send ("Bonjour UMMTO!");

sock.close(); // fermer la connexion
```

Figure 2 : Exemple d'utilisation des méthodes de l'API WebSocket

1.2. Internet des objets

1.2.1. Définition

L'Internet des objets (IdO) est un réseau qui permet, en passant par des systèmes d'identification électronique standardisés et unifiés, et des terminaux mobiles sans fil, d'identifier sans équivoque des entités numériques et des objets physiques et ainsi de pouvoir stocker, transférer et traiter, entre les mondes physiques et virtuels, les données s'y rapportant.

On peut également définir l'IdO comme étant un ensemble d'objets possédant des identités logiques (virtuelles), opérant dans des espaces intelligents et employant des

interfaces intelligentes pour se connecter et échanger de l'information dans des domaines différents.

1.2.2. Historique

L'avènement de l'Internet des objets (IdO) est dû à la forte évolution de la mécanisation et normalisation appliquée au traitement du document et de l'information, sur support matériel puis numérique (au service de la production et recherche documentaire). L'IdO est né aux États-Unis, il s'est rapidement propagé avec la mondialisation, conduisant à connecter des machines à des serveurs aptes à les superviser (ces machines étant principalement des ordinateurs mis en réseau dans ce que certains ont nommé l'« Internet des machines »), peu à peu des objets ont été modifiés (avec des puces RFID par exemple) ou conçus pour utiliser le protocole IP, devenant des « objets connectés », connectés à des serveurs centralisés et/ou capables de communiquer entre eux et/ou avec des réseaux de serveurs et différents acteurs. La figure ci-dessous montre le chronogramme relatif à l'IdO.

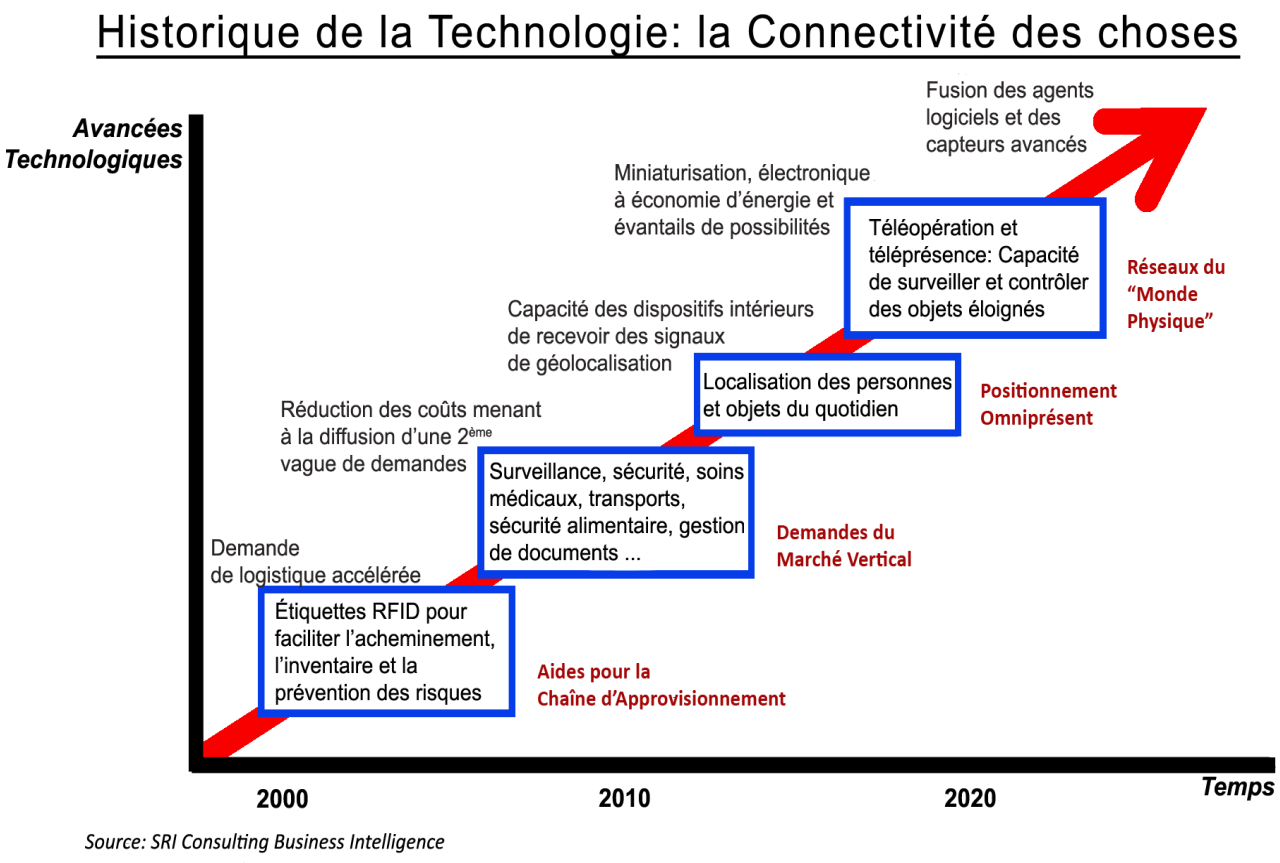


Figure 3: Historique de l'internet des objets

1.2.3. Domaines d'applications de l'IdO

Les domaines d'applications sont divers, nous citons ci-après quelques uns :

1.2.3.1. Transport et logistique

Les routes peuvent être dotées de capteurs qui transmettent des informations sur la circulation aux stations de contrôles et également aux voyageurs pour mieux gérer le trafic, améliorer la sécurité routière et guider les touristes.

1.2.3.2. Santé

Les objets connectés permettent d'identifier, en temps réel, les équipements et médicaments, avoir de telles informations instantanément peut souvent être décisif.

1.2.3.3. Environnements intelligents

Capteurs et actionneurs distribués dans plusieurs maisons et bureaux augmentent le confort : le chauffage peut s'adapter à la météo, l'éclairage suivre l'horaire et la position du soleil ; des incidents domestiques peuvent être évités avec des alarmes et beaucoup d'énergie pourrait être économisée.

La ville intelligente est un exemple d'environnement intelligent. Le quartier d'affaires de Songdo en Corée du Sud est la première ville intelligente fonctionnelle.

1.2.3.4. Signalements

Dans ce domaine, on trouve des systèmes permettant aux utilisateurs de signaler un dysfonctionnement, par exemple, à la sortie des toilettes publiques, on offre à l'utilisateur la possibilité de signaler un manque de savon, de papier, ou un problème d'odeur, rien qu'en appuyant sur un bouton.

1.2.4. Critiques et controverses

1.2.4.1. Fragmentation de la plateforme

L'Internet des objets souffre de la répartition des plateformes qui complique le développement des applications, imaginons qu'on est en train de développer une application de prévention d'incendies dans les forêts dont nous avons une multitude de capteurs connectés

à un serveur, et ce dernier, lui-même connecté aux actionneurs (alarmes par exemple) via Internet, cette fragmentation peut engendrer un travail fastidieux lors du déploiement et de la maintenance du système à cause notamment de l'éloignement géographique.

L'Internet des objets souffre également du manque de standards techniques tant au niveau matériel que logiciel, par conséquent, les objets connectés se voient dépendre de divers logiciels et protocoles parfois propriétaires présentant des complexités de personnalisation et de communication entre ces objets.

1.2.4.2. Sécurité

De grands sites Internet ont été piratés et bloqués pendant plusieurs heures tels que EBay, PayPal, Spotify, Twitter. De ce fait, les gens pensent que la sécurité est leur principale préoccupation au moment d'adopter l'Internet des objets, et ils ont totalement raison, car dans ce cas, on a pas seulement affaire à des menaces virtuelles, mais également physiques.

Joseph Steinberg, chroniqueur en cybersécurité, a révélé que plusieurs appareils connectés à Internet peuvent « espionner les gens dans leur propre maison », à savoir les téléviseurs, les caméras...etc.

Le 21 octobre 2016, un nombre important de ces appareils ont été détournés de leur fonction et piratés afin de voler des informations confidentielles.

1.2.4.3. Contrainte sur la conception

L'étape cruciale dans le développement d'un système informatique est sa conception qui accumule un bon nombre de facteurs de qualités fonctionnels et non fonctionnels (réutilisation, sécurité, adaptabilité, convivialité, passage à l'échelle …etc.). Dans le cas d'une conception destinée à un système relatif à l'Internet des objets, il y a un facteur de qualité qui peut vraiment causer de graves soucis à la conception, c'est l'adaptabilité flexible, qui veut dire que le système doit être extensible avec de faibles impacts, car, là, il ne s'agit pas seulement de créer une plateforme logicielle adaptable, mais également une plateforme physique, par exemple, si on gère les éclairages publics par Internet, le nombre des lampes peut augmenter, donc, il faudrait prendre en considération cette évolution sur le niveau logiciel aussi bien que sur le niveau matériel. [3] [4] [5]

1.3. La domotique

1.3.1. Définition

La domotique est la discipline qui étudie et réalise l'automatisation de différents systèmes de la maison et l'entreprise, notamment ce qui concerne la sécurité, la gestion de l'énergie, le contrôle des appareils électroménagers, et la communication (commandes à distance, signaux visuels ou sonores... etc.)

1.3.2. Techniques de domotique

On peut diviser les techniques de domotique en deux parties, la partie « serveur », qui est la centrale de commande, et la partie « client », qui est l'interface de contrôle.

La centrale de commande : est le cerveau du système de domotique, elle est programmable et l'intermédiaire entre les capteurs (de température, d'humidité, de mouvement… etc.), actionneurs (volets roulants, servomoteurs, alarme… etc.), et les commandes de l'utilisateur.

En outre, on peut programmer des scénarios, en l'occurrence, déclencher automatiquement l'arrosage à une certaine heure de la journée, allumer la lumière à l'entrée d'une personne s'il fait noir… etc.

L'interface de contrôle : elle est figée dans le domicile, et/ou accessible via Internet avec un smartphone ou un ordinateur, c'est par le biais de cette dernière que l'utilisateur peut piloter son système de domotique, où, par exemple, il peut ajuster la température de sa maison, de son lieu de travaille, et en y arrivant, il la trouve à la température voulue.

3.1.1. Exemples de scénarios de domotique

- En partant au travail, un simple clic sur un bouton installé dans l'entrée enclenche le scénario « départ au travail ». L'éclairage s'éteint, la porte de garage s'ouvre, le chauffage se met en veille au bout de 10 minutes, les volets et le garage se ferment après 20 minutes ;
- En quittant le travail pour rentrer chez soi, on active le scénario de retour à l'aide d'un smartphone ou depuis un ordinateur : les volets s'ouvrent et le chauffage s'allume ;

- Quand on est fatigué, on agit sur la télécommande de la maison afin d'enclencher le scénario « relaxation », les lumières se tamisent, un fond sonore apaisant se propage dans la pièce.

Les scénarios peuvent également se déclencher automatiquement :

- Le vent se lève et souffle puissamment. Le store de terrasse se relève afin d'éviter que celui-ci soit arraché;
- En été, un détecteur d'ensoleillement informe le système domotique de baisser les stores vénitiens ou les brise-soleil orientables afin de maintenir la maison au frais. En hiver, ce même détecteur peut, contrairement, ouvrir les fermetures de la maison afin de faire entrer la chaleur naturelle et économiser ainsi de l'énergie ;

3.1.2. Appareils de domotique

Il existe plusieurs appareils de domotique, citons quelques exemples :

3.1.2.1. La sécurité

Concernant la sécurité, de nombreux systèmes sont envisageables :

- Un ensemble de caméras de surveillance qui enregistrent en cas d'intrusion.
- Des détecteurs de présence qui, à l'approche d'un intrus, vont mettre en mouvement les fermetures de la maison et/ou simuler une présence (en allumant une lumière, par exemple) et/ou appeler automatiquement les services de sécurité.
- Ouverture automatique de la porte par le biais d'une carte magnétique ou d'un dispositif de reconnaissance vocale ;
- Un système de sécurité assuré par des contacts normalement fermés qui se déconnectent lors d'une ouverture. L'ouverture du circuit déclenche l'alarme.

3.1.2.2. Cinéma maison

Pour davantage de confort, la domotique propose également des systèmes qui permettent de contrôler facilement les médias (vidéo et son) dans la maison.

- La diffusion sonore permet d'envoyer de la musique dans la maison depuis un téléphone, une chaîne Hi-fi ou un PC.

- La diffusion de l'image permet de connecter toutes les sources vidéo (DVD, démodulateur satellite, décodeur, PC, bibliothèque multimédia, interphone, caméra…) et de les envoyer sur n'importe quel récepteur de l'habitat (poste TV principal ou secondaire, écran tactile…) ;

Les différentes applications décrites plus haut sont, pour la plupart d'entre elles, des prototypes à l'étude qui nécessitent encore de nombreuses mises au point pour être suffisamment fiables.

3.1.2.3. Éclairage et appareils électriques

La domotique permet à partir d'une seule commande de mettre en route ou éteindre les éclairages et le chauffage, de gérer l'ouverture ou la fermeture partielle ou totale des volets roulants, d'activer ou désactiver les systèmes d'alarme, mais aussi des appareils électroménagers tels que les machines à laver.

3.1.2.4. Loisir et jardin

La domotique peut permettre la gestion et le contrôle d'arrosage via Internet : un programmateur d'arrosage intégrant un pico serveur de pages web dédiées permet de gérer et de contrôler l'arrosage à distance via Internet.

3.1.2.5. Assistance à la vie quotidienne

Des robots d'assistance à la vie quotidienne ou à des malades ou handicapés sont en cours de mise au point dans de nombreux laboratoires de recherche. Ces robots domestiques prennent parfois forme humaine et peuvent se déplacer en évitant tous les obstacles. Ces machines sont actuellement limitées par des exigences de sécurité. Des robots disposent de dispositifs de sécurité leur ôtant toute force mécanique dès qu'ils entrent en contact avec une personne par exemple, mais ce type de dispositif ne suffirait pas, par exemple, pour la sécurité d'un enfant de moins de 3 ans. [6] [7] [8] [9]

Conclusion

Ce chapitre nous a permis de se familiariser avec le Web, l'internet des objets, les systèmes embarqués, et la domotique, cela, va forcément nous faciliter l'appréhension de la solution informatique que nous avons développée et que nous allons exposer dans les chapitres à suivre.

Chapitre 2 : Analyse

et conception

Introduction

L'analyse d'un projet informatique sert à déterminer si le projet est opportun et faisable, si cela est affirmé, on s'engage à spécifier les acteurs interagissant avec le système à concevoir, ainsi que les besoins attendus de celui-ci, une fois l'analyse est effectuée, on entame la conception du système qui est la tâche la plus cruciale dans n'importe quel projet informatique, car c'est sur elle que repose le travail des développeurs, donc, la conception prend la majeure responsabilité du projet.

Pour bien mener à terme l'analyse et la conception du système, nous avons choisi UML comme langage de modélisation, notons que l'UML est parmi les langages de modélisation les plus utilisés notamment pour la conception orientée objet. Il contient quatorze diagrammes normalisés, avec lesquels on peut pratiquement modéliser n'importe quel système informatique tout en offrant une grande lisibilité et précision à la conception.

3.1. Analyse

3.1.1. Diagramme de contexte

Ce diagramme UML consiste à définir les utilisateurs interagissant avec le système. Dans notre cas, c'est seulement le propriétaire du domicile qui interagie avec ce dernier.

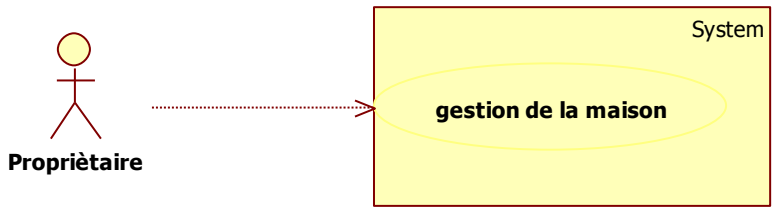


Figure 4 : Diagramme de contexte

3.1.2. Les cas d'utilisation

Le diagramme de cas d'utilisation sert à définir les besoins attendus du système à concevoir, il contient trois notions principales : acteur, cas d'utilisation, et relation. Exemple : si nous avons un acteur « client », un cas d'utilisation « acheter un produit », et une relation « association » entre ces deux derniers, cette représentation sera lue comme suit : le système conçu donne au client la possibilité d'acheter un produit.

Ci-dessous, nous représentons le diagramme de cas d'utilisation global.

Note : *pour des raisons de lisibilité, nous avons détaillé les cas d'utilisation dans des diagrammes séparés du diagramme global.*

3.1.2.1. Diagramme de cas d'utilisation global

Figure 5 : Diagramme de cas d'utilisation global.

3.1.2.2. Diagramme de cas d'utilisation « gestion des modes »

Le cas d'utilisation « gestion des modes » contient des sous cas d'utilisation qui permettront à l'utilisateur de créer, ajuster, sélectionner des modes, par exemple, en sortant de son domicile, il sélectionne le mode « hors maison » afin d'éteindre toutes les lumières, et activer le système d'alarme. Ci-dessous le diagramme UML qui lui correspond :

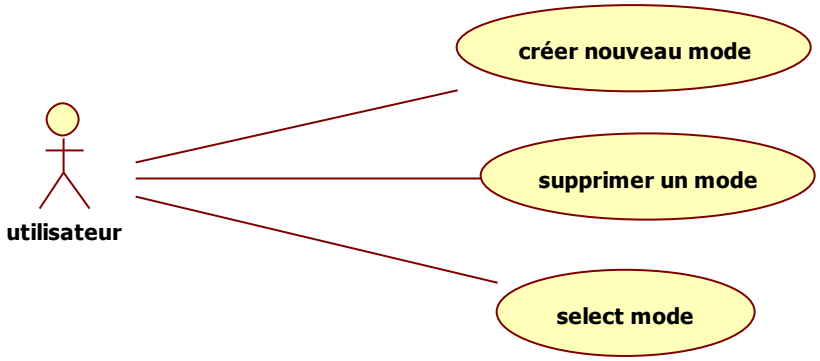


Figure 6 : Diagramme de cas d'utilisation "gestion des modes"

3.1.2.3. Diagramme de cas d'utilisation « gestion de la lumière »

La gestion de la lumière consiste à fournir à l'utilisateur une interface avec laquelle il va pouvoir contrôler les lumières de la maison, soit une à une, ou toutes à la fois.

Ci-dessous, le diagramme de cas d'utilisation « gestion de la lumière ».

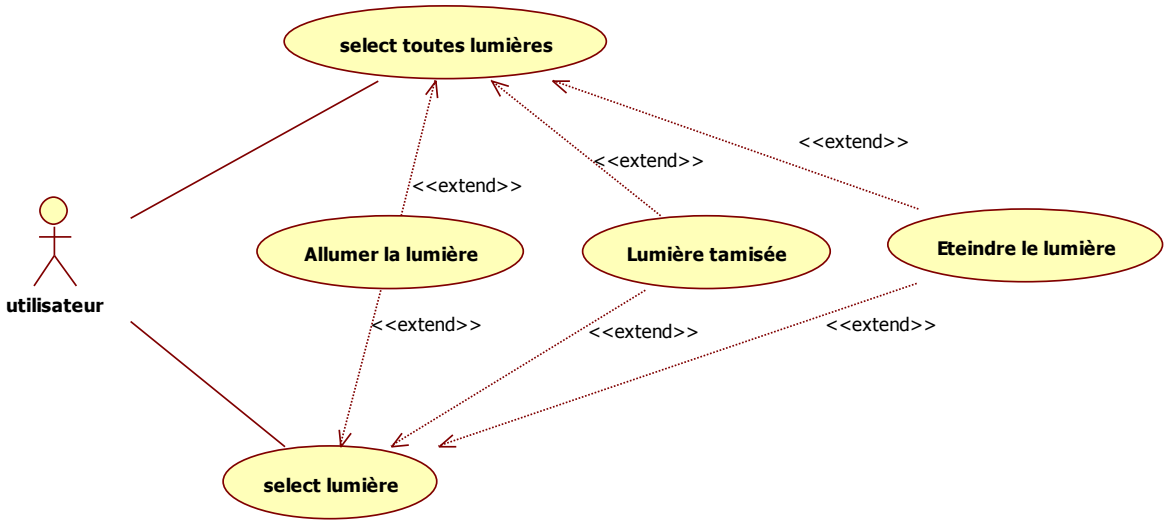


Figure 7 : diagramme de cas d'utilisation "gestion de la lumière"

3.1.2.4. Diagramme de cas d'utilisation « Gestion de la température »

La gestion de la température permet à l'utilisateur de prendre conscience de la température ambiante, ainsi que gérer la climatisation et le chauffage, et tout cela, il pourra le contrôler à distance.

Ci-après, le diagramme correspondant à ce besoin.

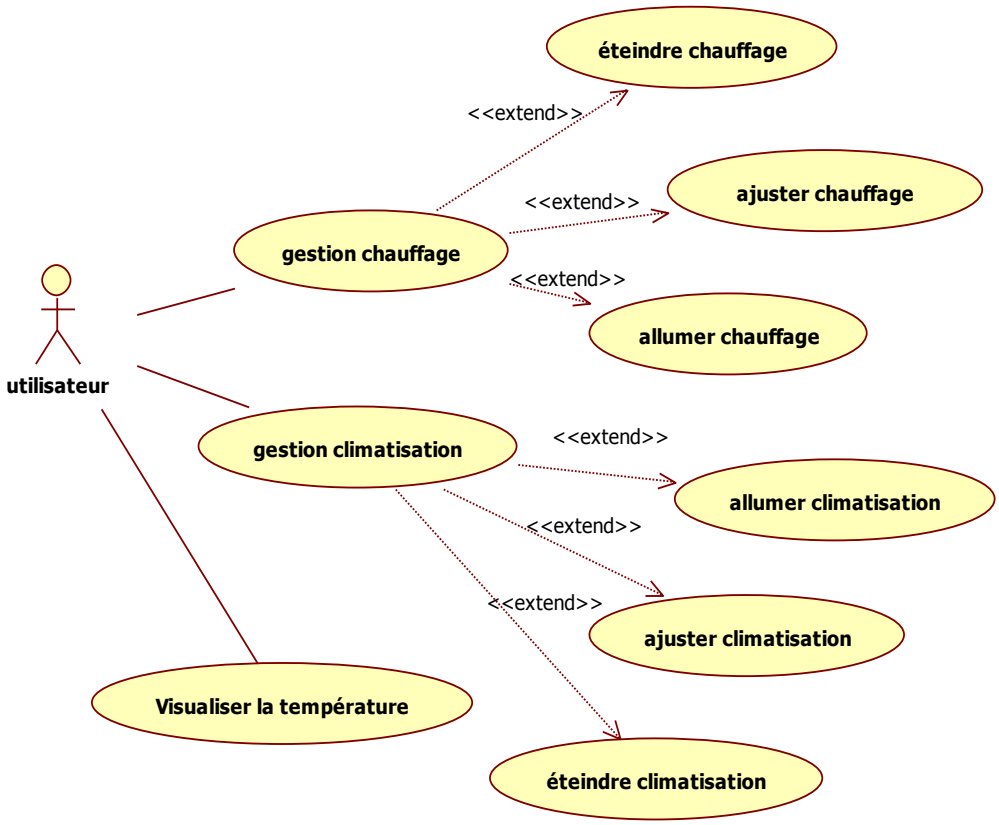


Figure 8 : Diagramme de cas d'utilisation «gestion de la température »

3.1.2.5. Diagramme de cas d'utilisation « gestion de l'acoustique »

Aujourd'hui, on sait créer des ambiances acoustiques dans les salles de concert où des microphones et des enceintes sont reliés par un système électronique et commandés par ordinateur. C'est ce qu'on projette sur la domotique, dont on offre au propriétaire de la maison la possibilité de gérer l'ambiance musicale, par exemple, en entrant de son travail épuisé, il peut ajuster l'ambiance acoustique à une ambiance musicale relaxante rien qu'avec son smart phone, voire automatiquement.

Ci-dessous, le diagramme de cas d'utilisation correspondant à ce dernier.

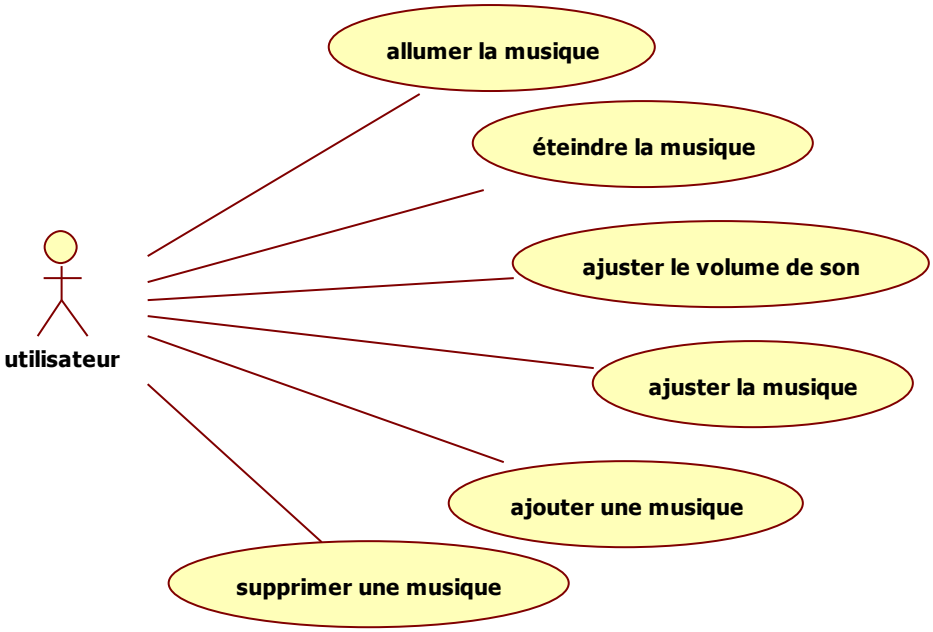


Figure 9 : Diagramme de cas d'utilisation "gestion de l'acoustique"

3.1.2.6. Diagramme de cas d'utilisation « gestion de la sécurité »

On ne peut pas travailler sur la domotique sans parler de la sécurité. Néanmoins, la sécurité est vaste, elle peut comprendre : la détection de voleurs, de gaz toxiques, d'incendies, la surveillance. De ce fait, il est difficile d'atteindre l'exhaustivité, quant à nous, nous avons prévu une conception qui prévient l'utilisateur s'il y a lieu d'une intrusion (si le système d'alarme est activé), et qui offre à l'utilisateur un système de surveillance.

Ci-dessous, le diagramme de cas d'utilisation « gestion de la sécurité ».

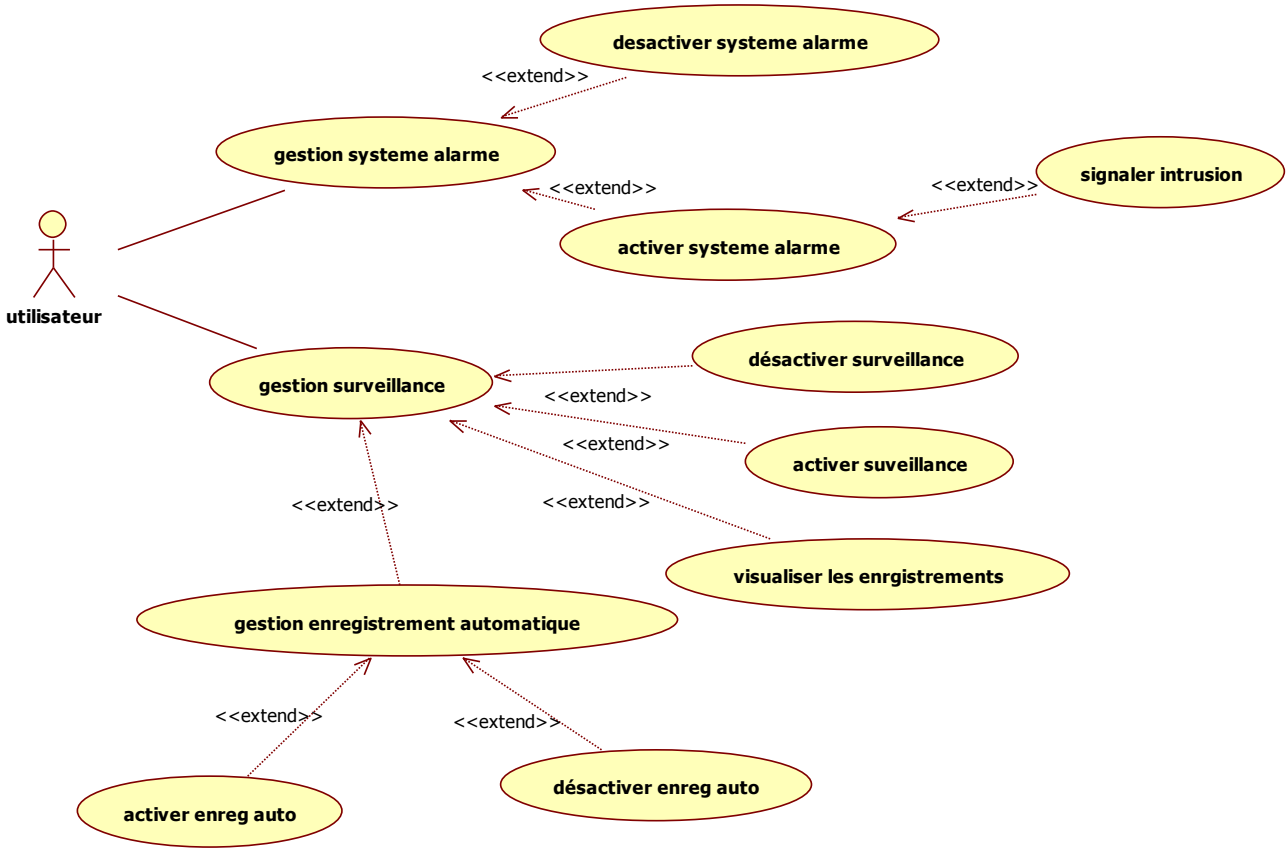


Figure 10 : Diagramme de cas d'utilisation «gestion de la sécurité"

3.1.2.7. Diagramme de cas d'utilisation « gestion de cinéma »

Le cinéma est une chose indispensable dans nos maisons, donc, il sera judicieux de satisfaire ce besoin.

Ci-dessous, le diagramme de cas d'utilisation correspondant à ce besoin.

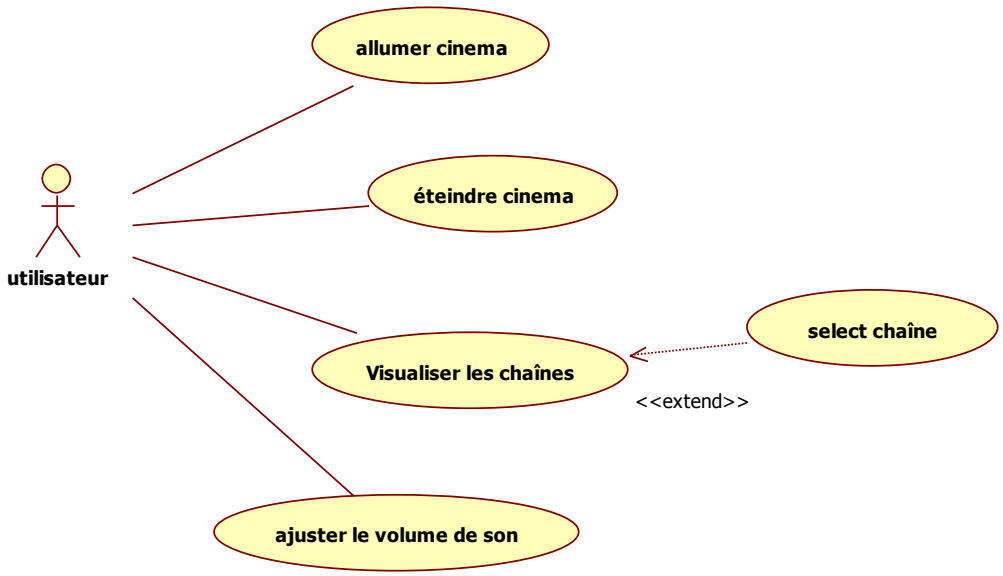


Figure 11 : Diagramme de cas d'utilisation «gestion de cinéma »

3.1.2.8. Diagramme de cas d'utilisation « gestion des portes »

Le propriétaire de la maison aimerait bien en entrant chez soi, que la porte de garage ou le portail s'ouvre automatiquement , et avoir d'autres fonctionnalités concernant les portes de son domicile. Ci-dessous, le diagramme de cas d'utilisation « gestion des portes ».

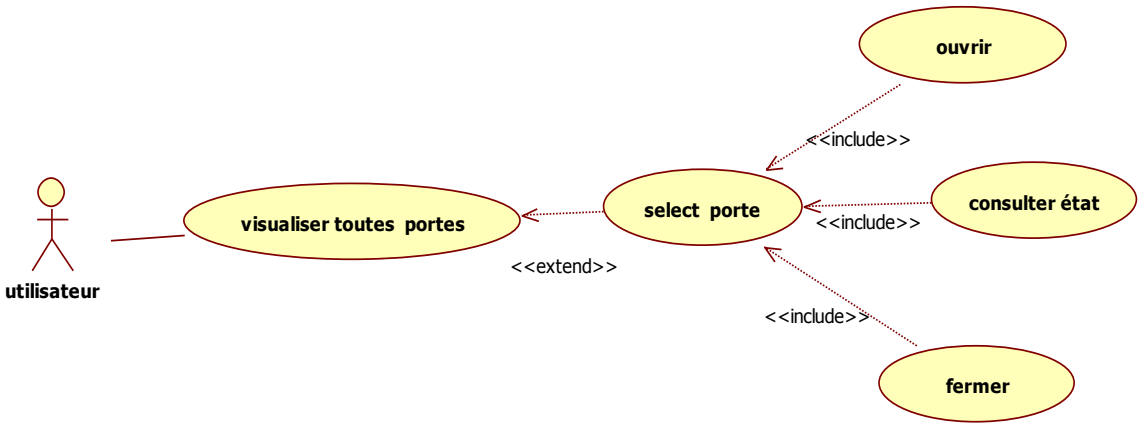


Figure 12 : Diagramme de cas d'utilisation « gestion des portes »

3.1.2.9. Diagramme de cas d'utilisation « gestion du jardin »

Ce qu'on peut offrir également au client, c'est la gestion du jardin, qui va lui permettre de contrôler l'arrosage à distance, voire la programmation de celui-ci.

Ci-dessous, le diagramme de cas d'utilisation « gestion du jardin ».

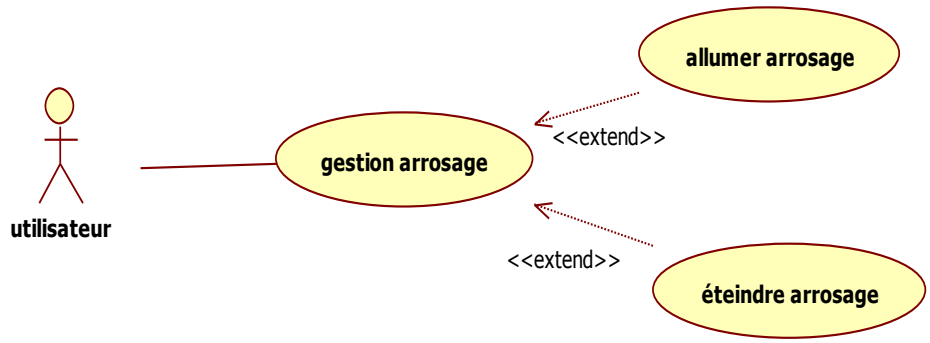


Figure 13 : Diagramme de cas d'utilisation "gestion du jardin"

3.1.2.10. Diagramme de cas d'utilisation « gestion des volets »

Un besoin aussi important que les autres, c'est la gestion des volets, donc, on offre ici au client la possibilité de contrôler à distance via pc ou Smartphone les volets de son domicile. Ci-après, le diagramme de cas d'utilisation « gestion des volets ».

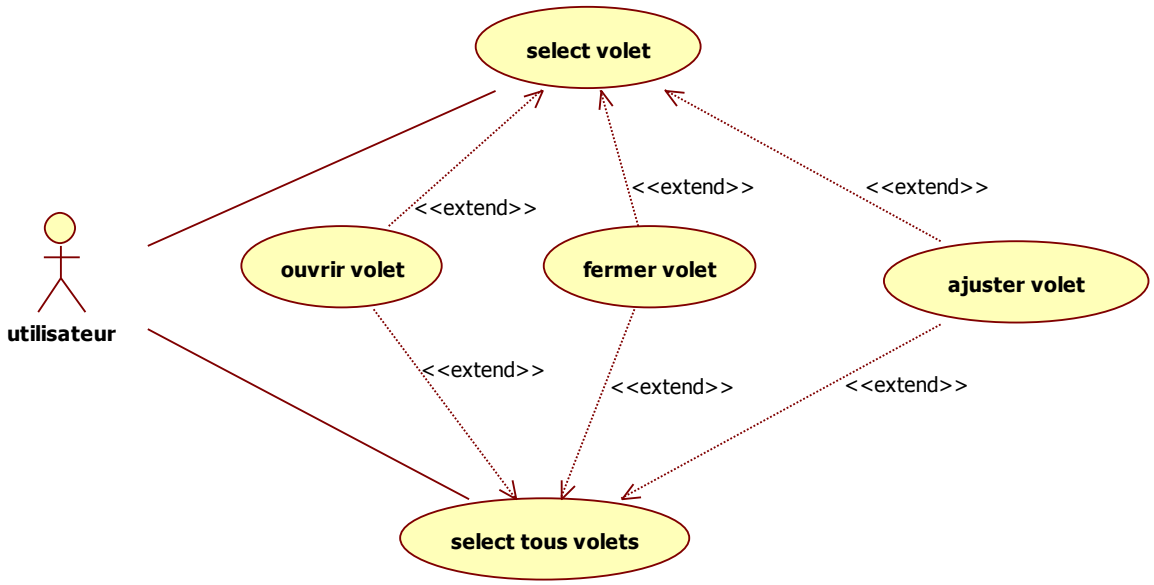


Figure 14 : Diagramme de cas d'utilisation "gestion des volets"

3.2. Conception

Afin de concevoir le système décrit au-dessus, nous avons utilisé la conception orientée objet, car, elle comporte des concepts avancés tels que : les interfaces, l'héritage…etc. Ces derniers, simplifieront la création d'une conception tout en respectant les principes de SOLID, et simplifieront également l'intégration des patrons de Gang of four, ces derniers à leur tour, contribuent énormément à concevoir un système qui répond aux principes de l'architecture logicielle.

Dans ce qui suit, nous allons présenter quelques diagrammes de séquences, et quelques diagrammes de classes que nous jugeons plus significatifs, et cela, en raison du nombre important des diagrammes, en outre, le principe de la solution reste toujours le même pour le reste des diagrammes.

3.2.1. Diagrammes de séquences

Diagramme de séquences d'une tâche définit notamment les objets intervenant à la réalisation de cette dernière, ainsi que la succession des appels entre ses objets. Cela, aide énormément le développeur à comprendre comment programmer la tâche, et c'est le but principal de n'importe quelle conception. Ci-dessous, nous allons présenter quelques diagrammes de séquences.

3.2.1.1. Diagramme de séquence « Allumer la lumière »

Ci-dessous, le diagramme de séquence correspondant à la tâche « Allumer la lumière ».

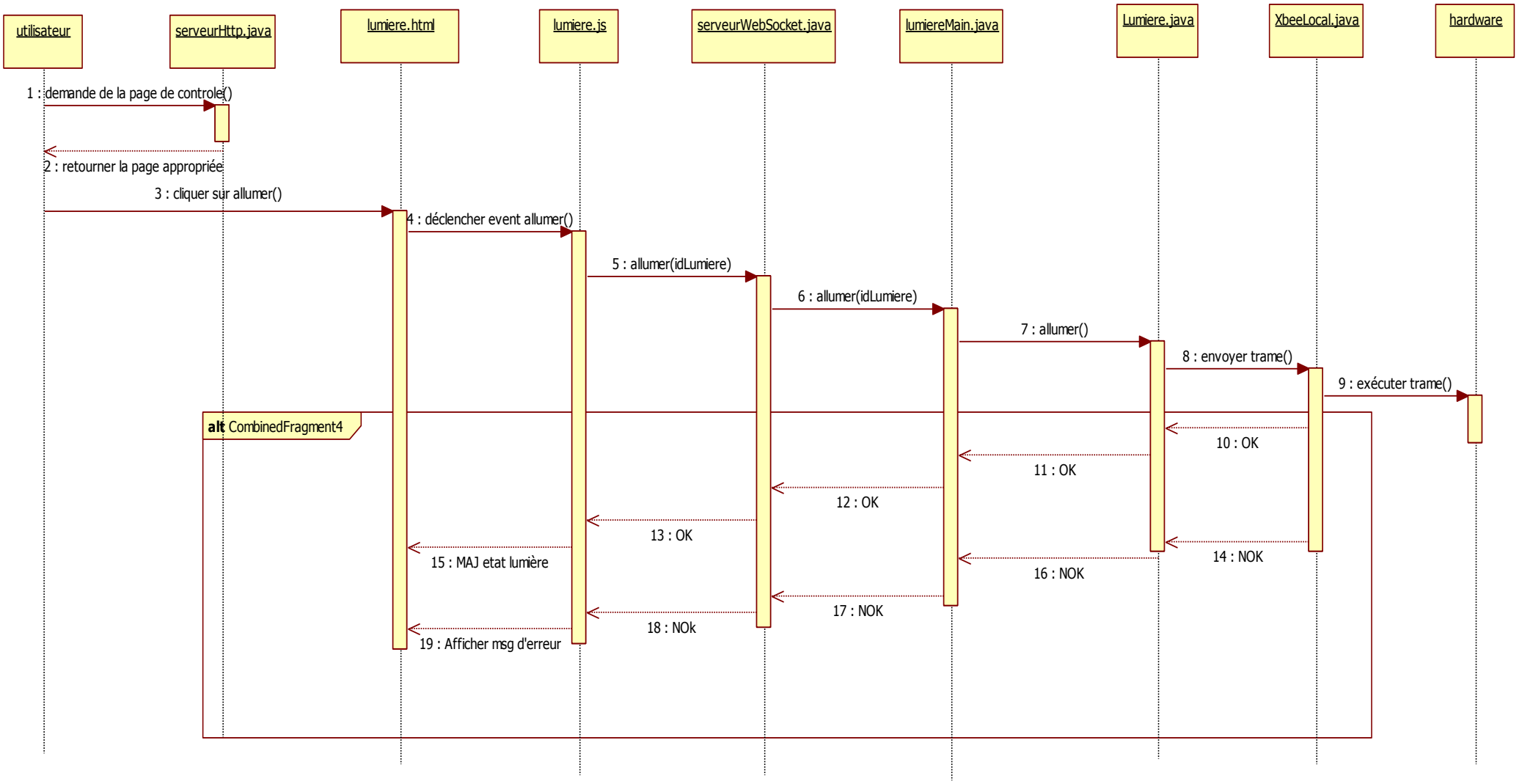


Figure 15 : Diagramme de séquence "Allumer la lumière"

3.2.1.2. Diagramme de séquences « Visualiser la température »

Ci-dessous, le diagramme de séquences correspondant à la tâche « visualiser la température ».

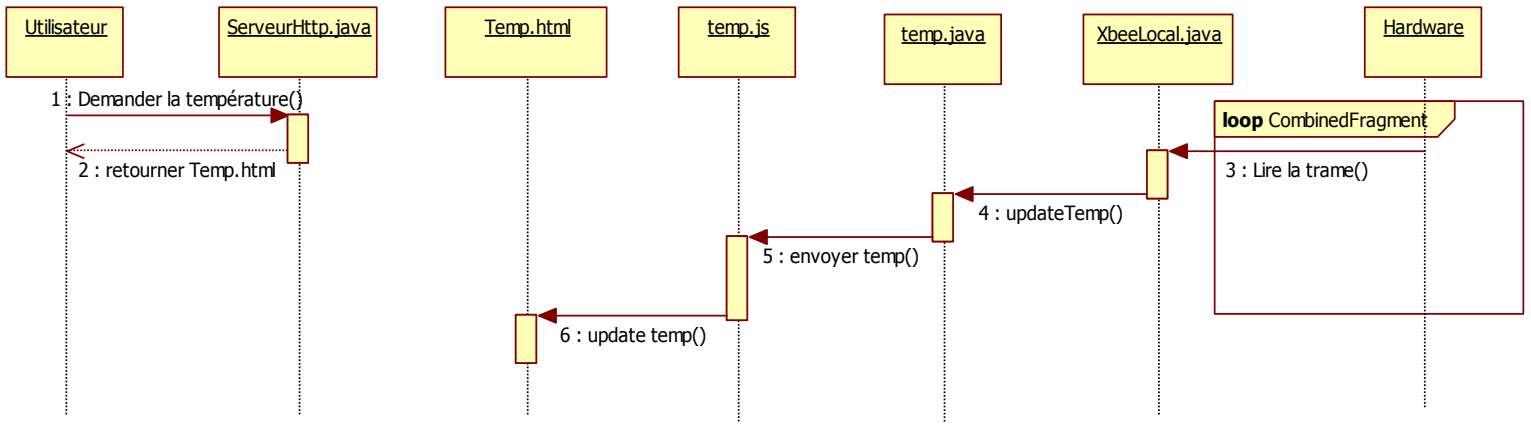


Figure 16 : Diagramme de séquences "Visualiser la température"

3.2.1.3. Diagramme de séquence « Signaler une intrusion »

Ci-dessous, le diagramme de séquences correspondant à la tâche « signaler une intrusion ».

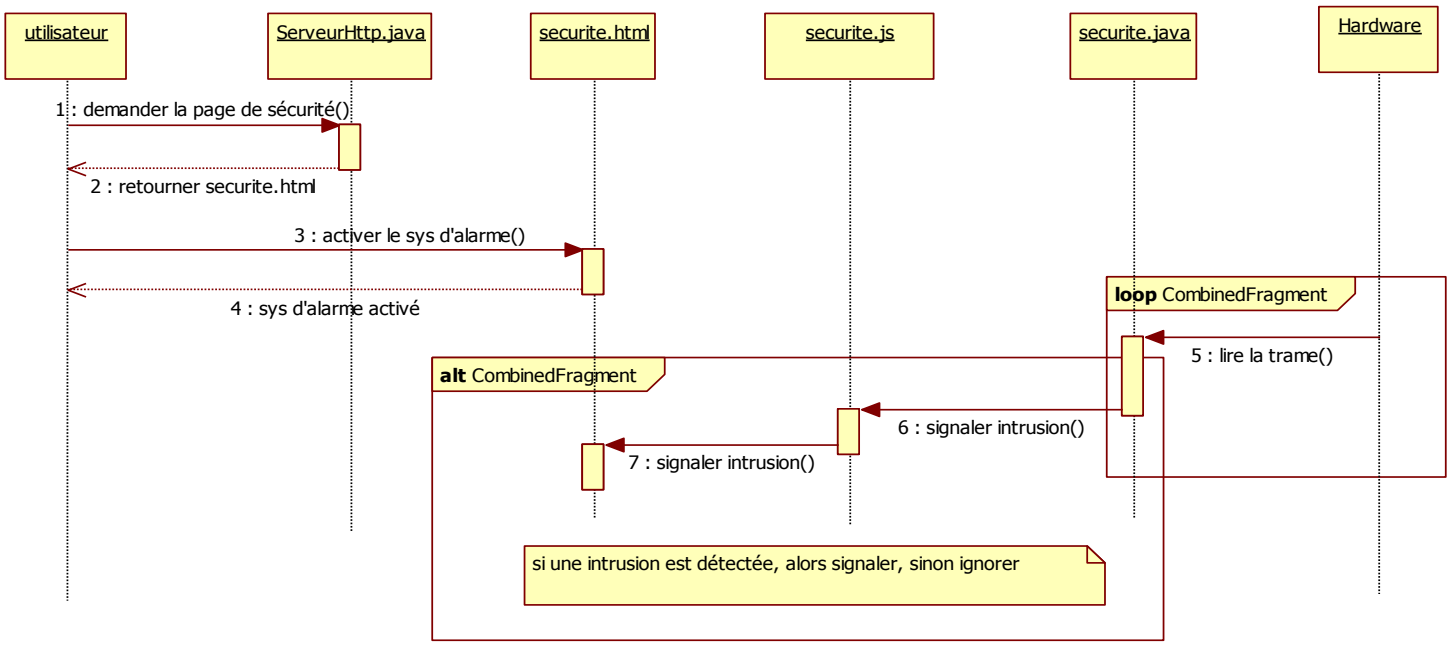


Figure 17 : Diagramme de séquences "signaler une intrusion"

3.2.2. Diagrammes de classes

Le diagramme de classes global définit la structure statique des données du système sous forme de classes, interfaces, et relations entre ces dernières. Et le diagramme de classe détaillé de chaque classe définit les attributs et les méthodes constituant cette classe. En effet, nous allons nous limiter au diagramme de classes global, car le nombre de classes est relativement grand pour qu'on présente chaque classe en détail. Néanmoins, le diagramme de classes global est largement suffisant pour comprendre l'architecture du système.

3.2.2.1. Diagramme de classes global

Ci-dessous, nous présentons le diagramme de classes global correspondant au système.

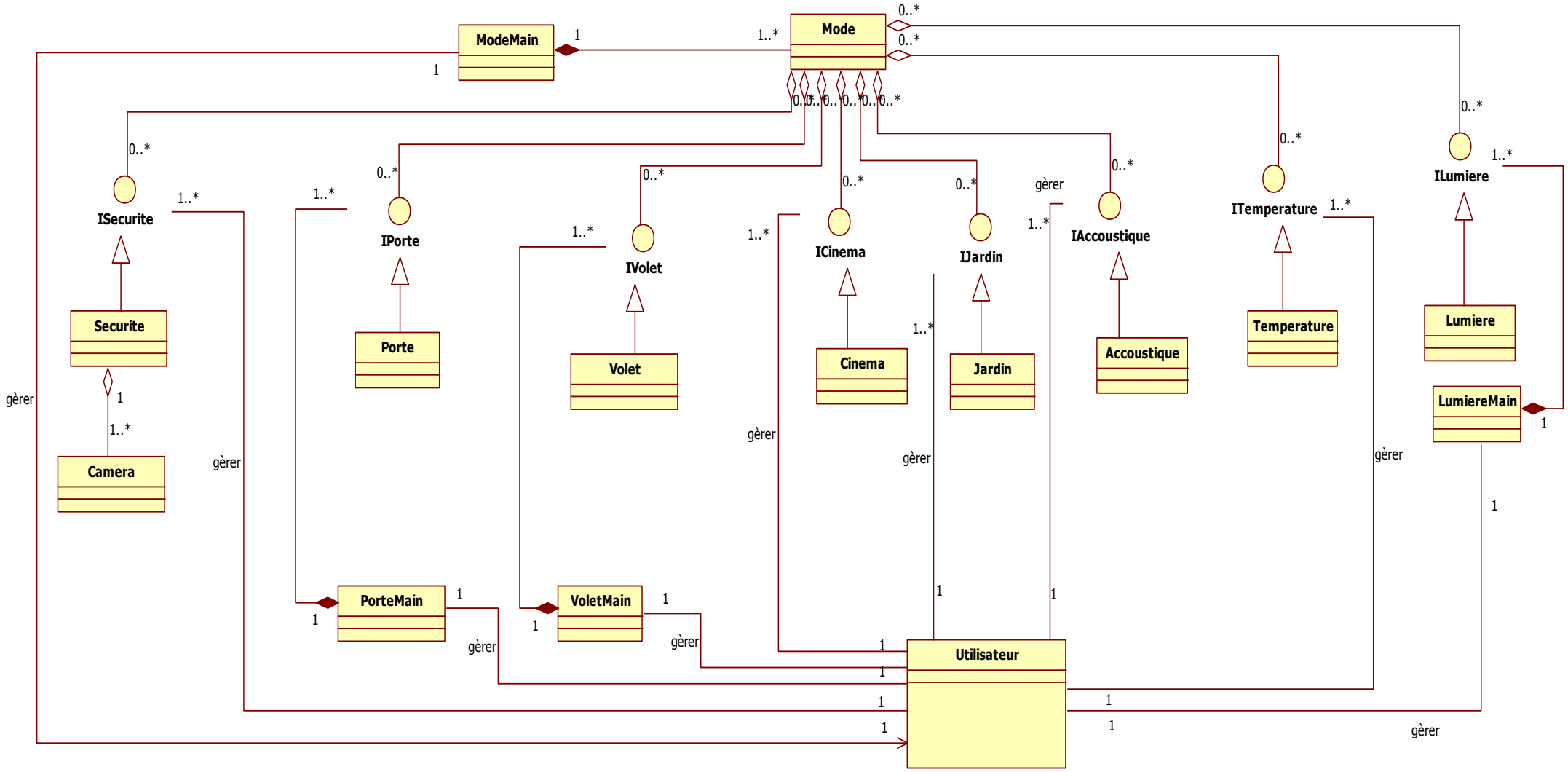


Figure 18 : Diagramme de classes global

3.2.2.1.1. Quelques éclaircissements sur le diagramme de classes

L'utilisation des interfaces : on remarque par exemple que la classe « utilisateur » ne gère pas directement la classe « Temperature ».

Raison : Cela est afin de respecter le 2ème principe de SOLID, qui est Open/Closed principle, qui veut dire que le système doit être ouvert aux extensions et fermé aux modifications. Et c'est exactement ce qu'on atteint avec cette conception. Exemple, si nous avons implémenté la classe « Temperature » avec une certaine API, et puis nous voulons l'implémenter avec une autre, nous n'avons qu'à implémenter l'interface «ITemperature ».

L'utilisation de la composition : on remarque que nous avons utilisé la relation de composition, entre, par exemple, la classe «Lumiere » et la classe « LumiereMain ».

Raison : Dans ce cas, nous aurions pu utiliser seulement une classe « Lumiere », sans créer la classe « LumiereMain », mais, lorsqu'il s'agit de contrôler toutes les lumières à la fois, c'est plus simple avec la classe « LumiereMain », car elle contient la collection des « Lumiere », et si on veut par exemple allumer toutes les lumières, on n'a qu'à parcourir la collection et appeler la méthode « allumer » de chaque objet « Lumière ».

En outre, l'accès à une certaine lumière est plus simple et plus dynamique avec la classe « LumiereMain », car elle contient une map dont la clé est l'identificateur de la lumière, et la valeur est l'objet « Lumière », donc, si on veut accéder à une lumière, on a seulement besoin de l'identificateur, et ça nous dispense d'utiliser des « if...else »

L'utilisation de l'agrégation : on remarque que nous avons employé l'agrégation notamment entre la classe « Mode » et les autres classes. Nous rappelons que le mode désigne un certain état des fonctionnalités, par exemple, on suppose un mode « hors maison », dont toutes les lumières sont éteintes, les portes sont fermées…etc.

Raison : Un mode peut ignorer quelques fonctionnalités, et c'est la raison pour laquelle nous avons utilisé l'agrégation, contrairement à la composition qui imposerait la présence de toutes les fonctionnalités.

Remarque : les classes « LumiereMain », « Temperature », « Acoustique », « Jardin », « PorteMain », « Securite », « VoletMain », « ModeMain », « Cinema » devront être des classes singleton, ce qui veut dire qu'elles n'auront qu'une seule instance, car, ça serait incohérent de créer plusieurs instances pour ces dernières alors qu'elles ne sont liées qu'à une

seule configuration matérielle. Cela, est facile à concevoir grâce au pattern Singleton des patterns de gang of four.

Note : pour plus d'éclaircissement sur les principes de SOLID et les patterns de gang of four, nous en avons parlés en annexe de cette rédaction.

Conclusion

Grâce à UML, nous avons pu modéliser notre conception d'une manière claire et précise. À ce stade, nous pouvons entamer l'implémentation du système conçu en toute tranquillité puisque nous avons en main une conception détaillée et bien étudiée. C'est ce que nous allons aborder au prochain chapitre.

Chapitre 3 :

Réalisation

Introduction

Nous allons consacrer ce chapitre à citer les technologies logicielles et matérielles que nous avons utilisées afin de mettre au point ce projet, ainsi que présenter la solution pour laquelle nous avons opté tout en décrivant les contraintes qui nous ont emmenées à nos choix.

3.3. Outils de développement

3.3.1. Outils matériels

➢ Carte Raspberry Pi

Le Raspberry Pi est un nano-ordinateur monocarte à processeur ARM, avec un système d'exploitation Raspbian, qui est une variante de Debian adaptée au processeur ARM. Puisque nous allons travailler sur le Web, c'est sur cette carte qu'on va installer le programme serveur. Ainsi, elle va être l'intermédiaire entre les capteurs, les actionneurs, et les commandes du client Web. [10]

Caractéristiques de la carte Raspberry pi que nous avons utilisée :

- Nom : Raspberry Pi Model B+.
- Processeur : ARM V6.
- RAM : 512 Mb
- 40 broches.
- Port HDMI.
- 4 ports USB.
- Port audio.
- Port d'alimentation 5v, de 600 à 1800mA du courant électrique.
- Port SD carte.
- Port RJ45.
- Port pour caméra
- Port pour afficheur.

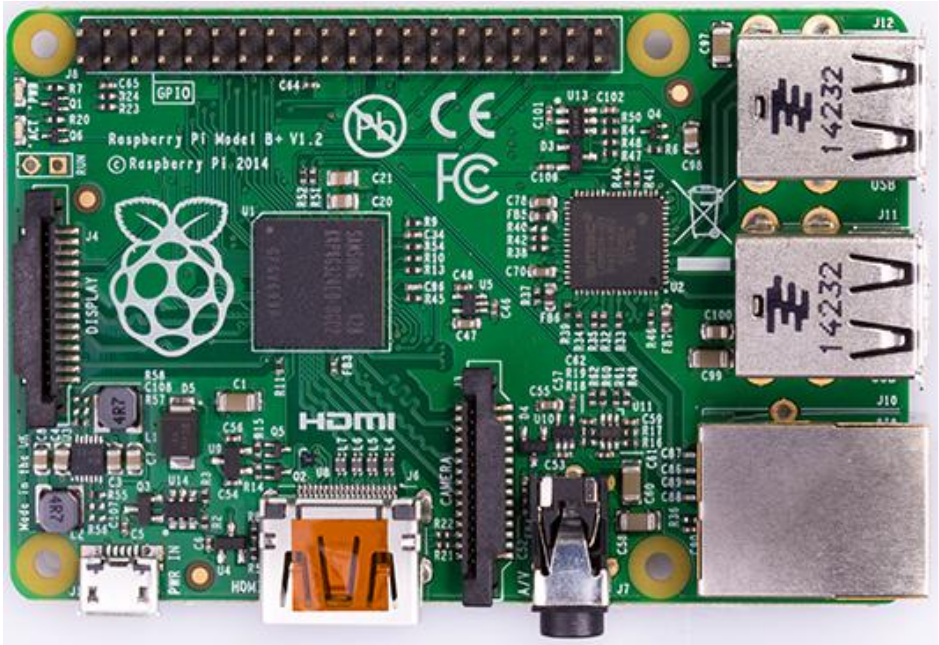


Figure 19 : carte Raspberry Pi B+

➢ Xbee

L'Xbee est un module émetteur et récepteur d'ondes radios, il comporte un CAN, et un port série asynchrone, des entrées analogiques et des entrées/sorties digitales. Il nous sert à la transmission sans fil entre les capteurs, actionneurs, et la carte Raspberry, et il joue aussi le rôle d'un CAN. [11] [12] [13] [14]

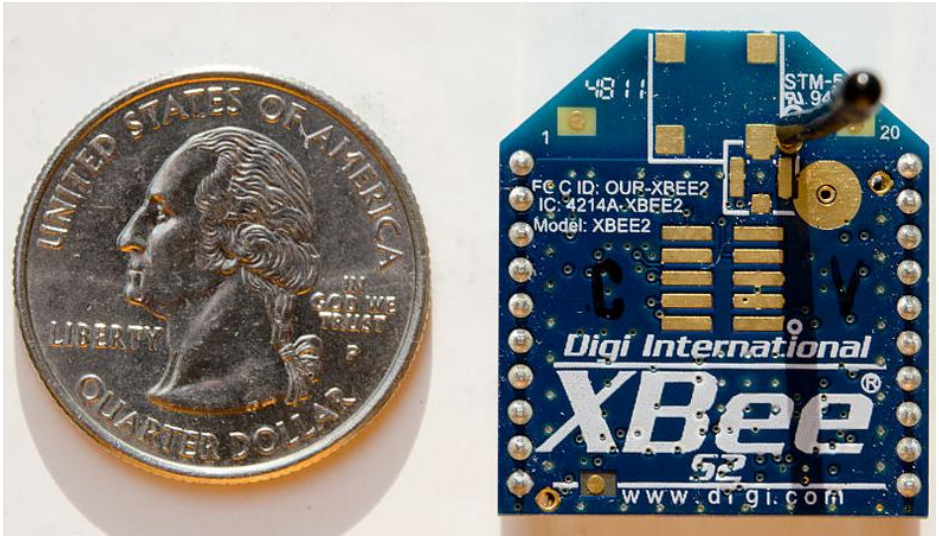


Figure 20 : Un XBee comparé à une pièce de monnaie

➢ Capteur de température LM35

Ci-dessous l'image correspondant au capteur de température LM35, une broche pour le GND, une broche pour 5V, et une dernière pour la sortie analogique dont chaque 10 MV correspondent à 1 °C.

Figure 21 : capteur de température LM35

➢ Détecteur de mouvement PIR

Il est facile à utiliser, son principe est de donner une impulsion de 5V de durée de quatre secondes à peu près, une fois un mouvement est détecté.

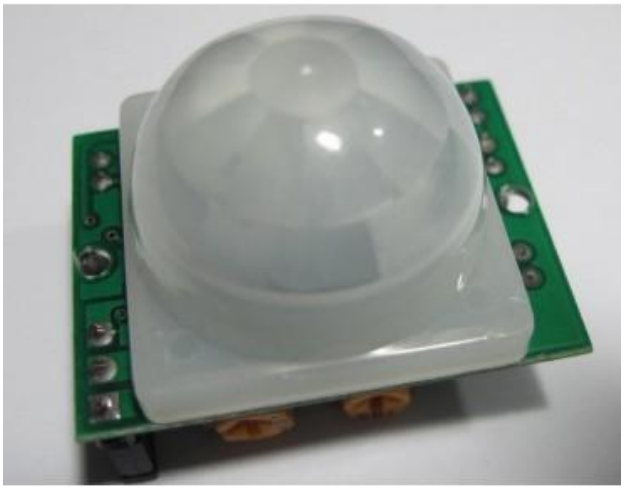


Figure 22 : détecteur de mouvement PIR

➢ Moteur pas-à-pas

Nous avons utilisé un moteur pas à pas afin d'interfacer un volet roulant, la figure suivante le représente.

Figure 23 : moteur pas-à-pas

➢ Autres outils

- Un PC, pour pouvoir communiquer avec la Rosebery via SSH.
- Des lampes, plaque d'essai, et câbles.

3.3.2. Outils logiciels

➢ Langages de programmation

Nous avons utilisé le langage de programmation **Java** en raison de son architecture orientée objet qui est une architecture moderne et conseillée par les grands informaticiens, ainsi que, la disponibilité des API pour pouvoir utiliser les broches de la Raspberry et les Xbee.

Nous avons également utilisé les langages **HTML** et **CSS** pour les interfaces graphiques, le langage **JavaScript** pour l'interaction entre la WebSocket du côté client et la WebSocket du côté serveur, et le langage **JSON** pour formater les messages échangés entre les clients et le serveur.

➢ IDE

Nous avons employé comme IDE Netbeans, qui est conçu par Oracle, la propriétaire de Java. [12]

➢ Système d'exploitation

Raspbian est le système d'exploitation que nous avons installé sur la Raspberry, et il est le mieux adapté, en effet, il n'a été conçu rien que pour cette dernière.

Et pour pouvoir interagir avec la Raspberry avec SSH, nous avons eu besoin d'un PC. Celui que nous avons utilisé, il a comme système d'exploitation Windows 8.1.

➢ API externes

Pour pouvoir utiliser les Xbee, nous avons eu besoin d'une API externe appelée « XBee Java library », et une autre pour pouvoir utiliser les broches de la Raspberry appelée « PI4j ».

3.4. Principe général de la solution

Puisque nous devons contrôler des objets physiques à distance, il faudrait que ça soit en temps réel, nous rappelons qu'un système temps réel est un système qui dépend des contraintes temporelles, exemple tiré de notre cas : la consultation de la température de la maison dépend du temps, car la température change de temps à autre, alors le système doit afficher la température avant qu'elle change de valeur. Alors, notre système a une contrainte majeure qui est les performances.

Comment faire face à cette contrainte ?

- **Créer un serveur web natif**

les serveurs d'applications java sont riches en modules, ce qui fait qu'ils sont lents, et pour remédier à cet inconvénient, nous étions obligés de créer un serveur HTTP natif qui sera léger et performant.

- **Utiliser les websockets**

Le websockets et un protocole qui crée un canal de communication full duplex entre le serveur et le client. l'avantage majeur offert par les websockets, est que le serveur peut envoyer un message au client sans qu'il y ait une requête par ce dernier, ce qui est impossible

avec le protocole HTTP où c'est toujours le client qui initie la communication avec une requête HTTP, et le serveur lui répond par une réponse HTTP.

En outre, nous avons développé un code avec un minimum de modules afin de réduire les appels entre ces derniers.

- **Utiliser le langage JSON**

JSON est un langage statique de description de données, nous l'avons utilisé pour formater les messages échangés entre le client et le serveur. Par exemple, pour dire au serveur « allumer la lampe 1 » avec JSON :

Message :{ action : « lumiere »,

Cmd : « allumer »,

Para : 1}

C'est un langage très simple, mais son importance s'avère lorsque le nombre de messages est important, et avec ce langage, on peut facilement créer une hiérarchie de messages, c'est ce que nous avons créé afin de réduire le nombre de tests qu'on est censé faire sur les messages reçus. Une petite illustration : si nous avons deux types d'actions, et chaque action possède deux types de commandes, et chaque commande peut avoir deux paramètres, alors le nombre de tests qu'on doit faire sans hiérarchie est huit tests, mais avec hiérarchie est trois. Cela n'est qu'une petite illustration, et notre système est plus vaste que ça, et il comporte un nombre important de messages échangés dans les deux sens entre le client et le serveur.

3.5. Démonstration de la solution

La création d'une solution intégrale pour toute une maison requiert énormément de moyens, et par manque de ces derniers, notre solution est réduite à quelques fonctionnalités, mais on peut toujours projeter la solution sur les autres moyens que n'avons pas utilisés, en effet, si on allume une lumière ou un chauffage, le principe reste toujours le même.

Ci-dessous, nous présentons une vue globale du côté hardware du système ainsi que quelques fonctionnalités développées.

3.5.1. Vue globale du matériel

Ci-dessous, nous présentons une vue globale sur le côté matériel du système.

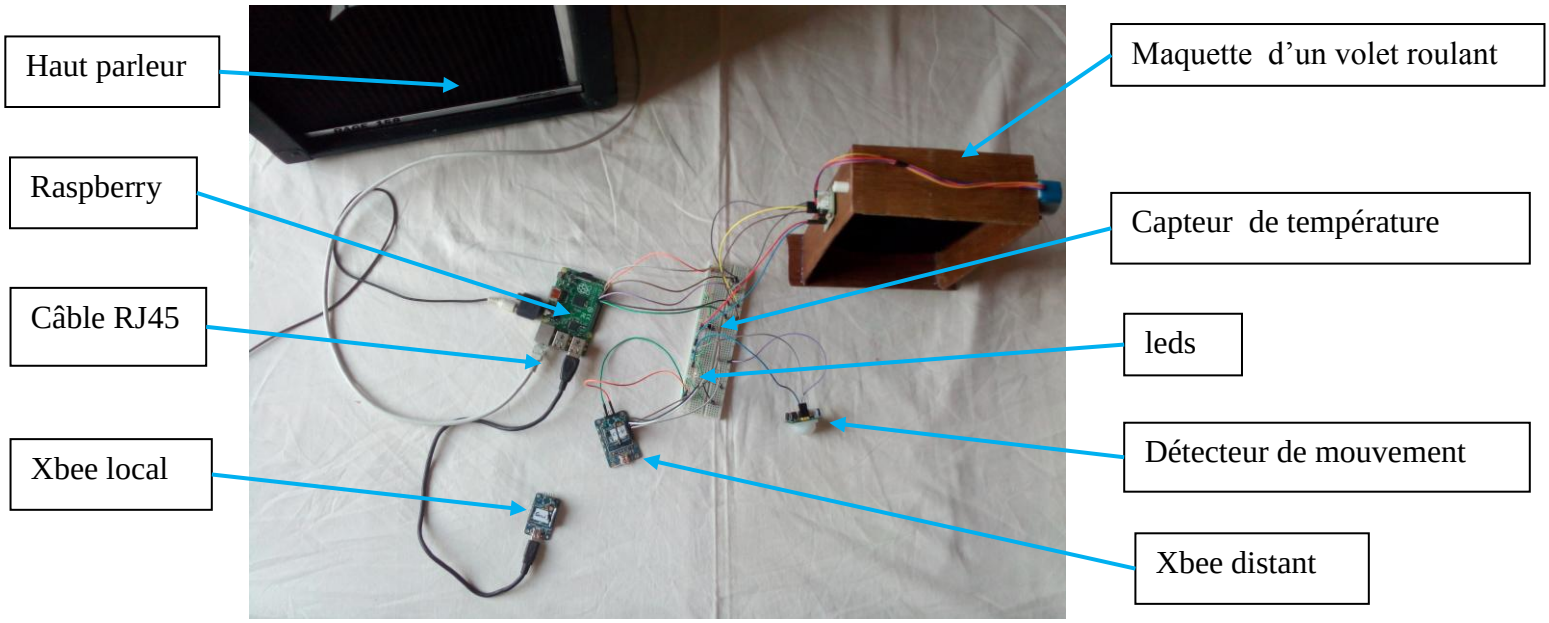


Figure 24 : Vue global du matériel utilisé

3.5.2. Tâche d'authentification

Pour des mesures de sécurité, l'utilisateur doit s'authentifier avant de pouvoir prendre le contrôle de la maison via la plateforme. Ci-dessous l'interface graphique qui permet l'authentification de l'utilisateur.

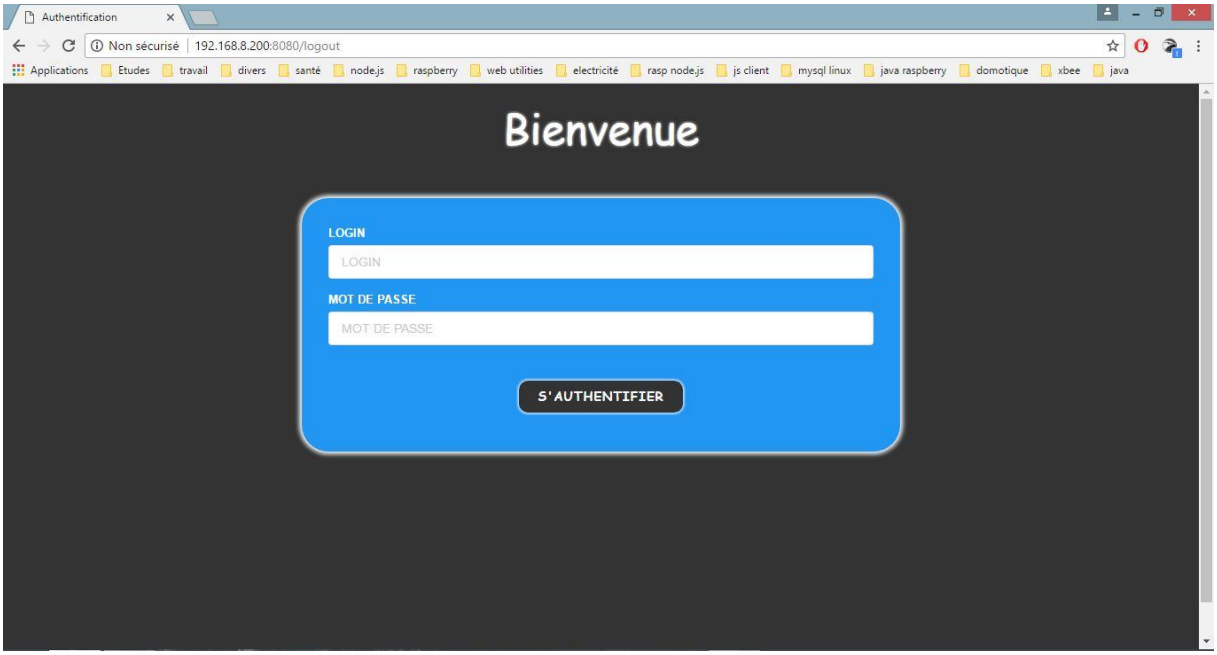


Figure 25 : Vue d'authentification

3.5.3. Contrôle de la lumière

Ci-dessous, l'interface graphique qui permet le contrôle de deux lumières soit une à une, soit toutes à la fois.

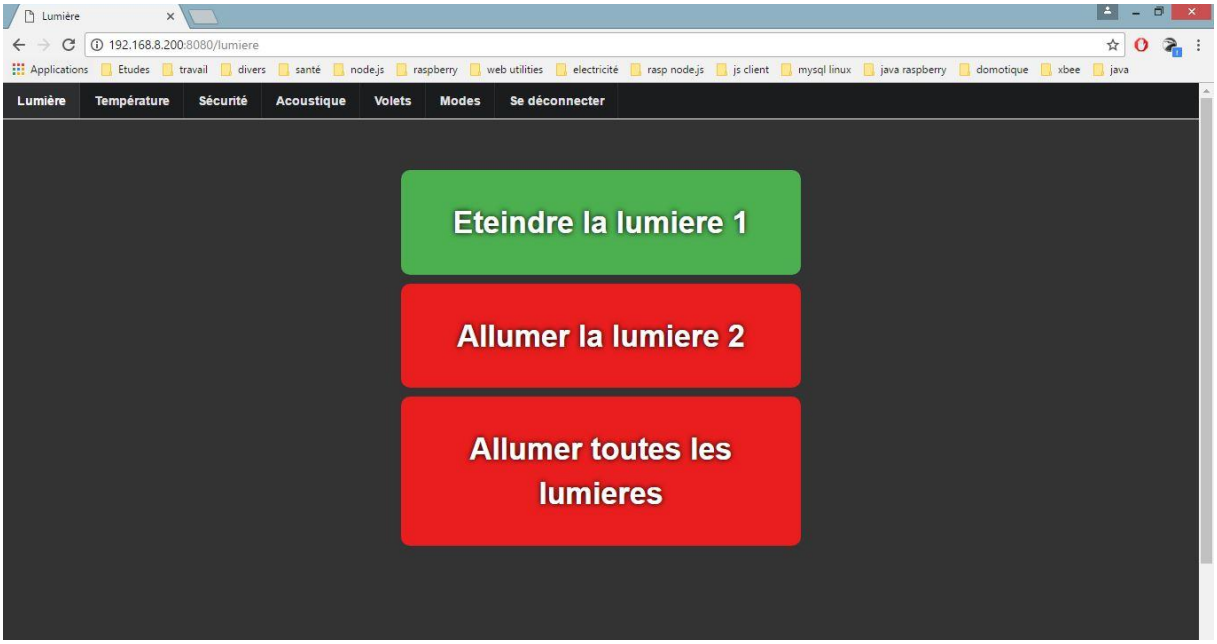


Figure 26 : Vue de gestion des lumières

3.5.4. Consulter la température

3.5.4.1. Le calcul de la température

Puisque nous avons utilisé le capteur de température LM35 qui donne en sortie une valeur analogique qui est une grandeur électrique, il a fallu utiliser un convertisseur analogique numérique, et c'est une bonne chose que les XBee que nous avons utilisés comportent des CAN intégrés donnant des grandeurs numériques, à ce stade, il ne reste que calculer la grandeur physique à partir de ces dernières. La figure suivante présente la chaine d'acquisition de la température.

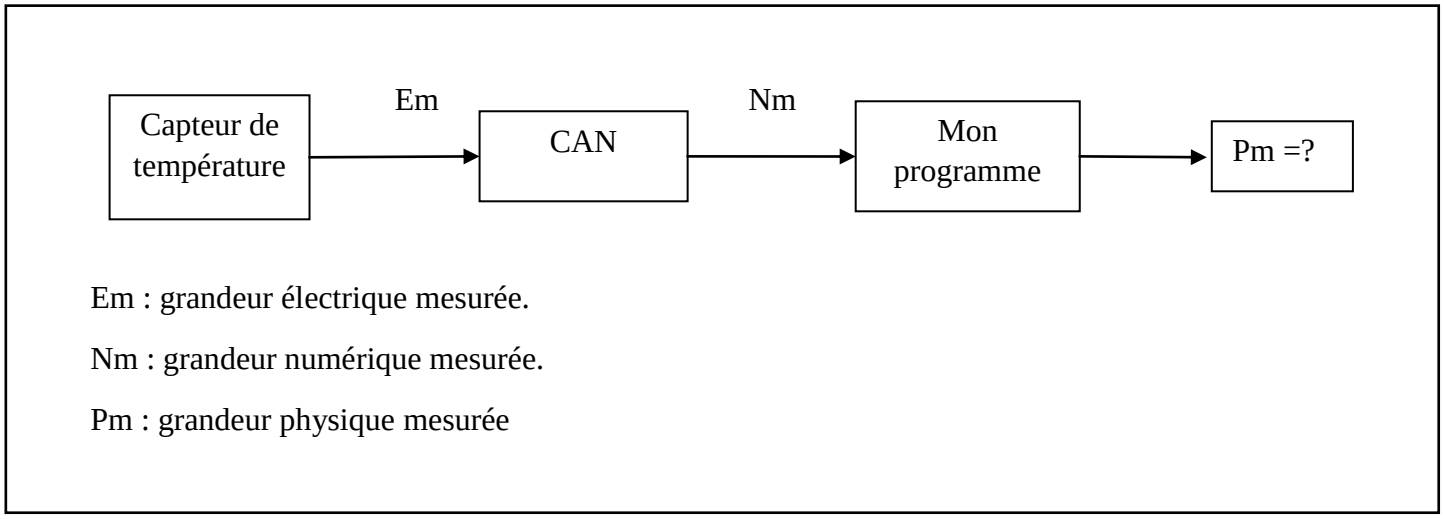
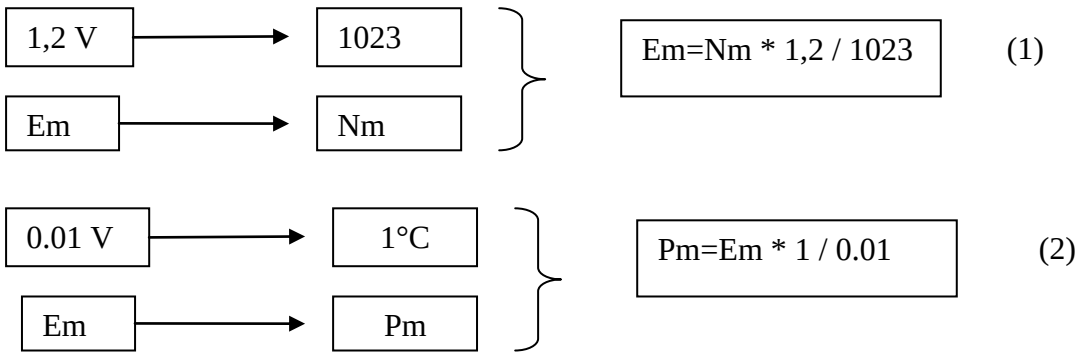


Figure 27 : chaine d'acquisition de la température

Les données que nous avons pour calculer la température mesurée :

- Le CAN donne une valeur numérique comprise entre 0 et 1023 pour une entrée électrique comprise entre 0 et 1,2 V.
- Le capteur de température LM35, pour chaque 1°C, il donne en sortie 10mV.

À partir de ces données, on déduit que :



On remplace (1) dans (2), et on aura :

Pm = Nm * 1,2 / 1023 / 0.01 = Nm * 120 / 1023

Ci-dessous, l'interface graphique qui permet la consultation de la température ambiante de la maison.

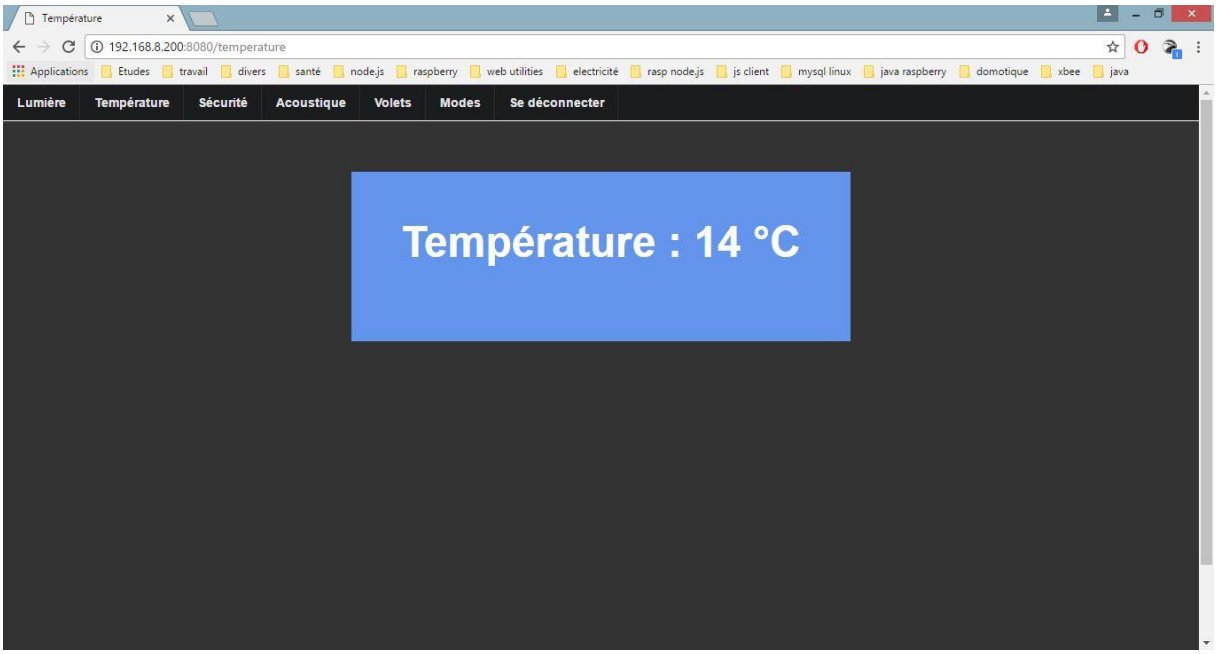


Figure 28 : Vue de gestion de la température

3.5.5. Gestion de la sécurité

Ci-dessous, l'interface graphique qui permet d'activer ou de désactiver le système d'alarme. Si un mouvement est détecté et le système d'alarme était activé, il serait signalé dans l'interface courante, et pour cet effet également, un Email et un SMS seront envoyés à l'utilisateur.

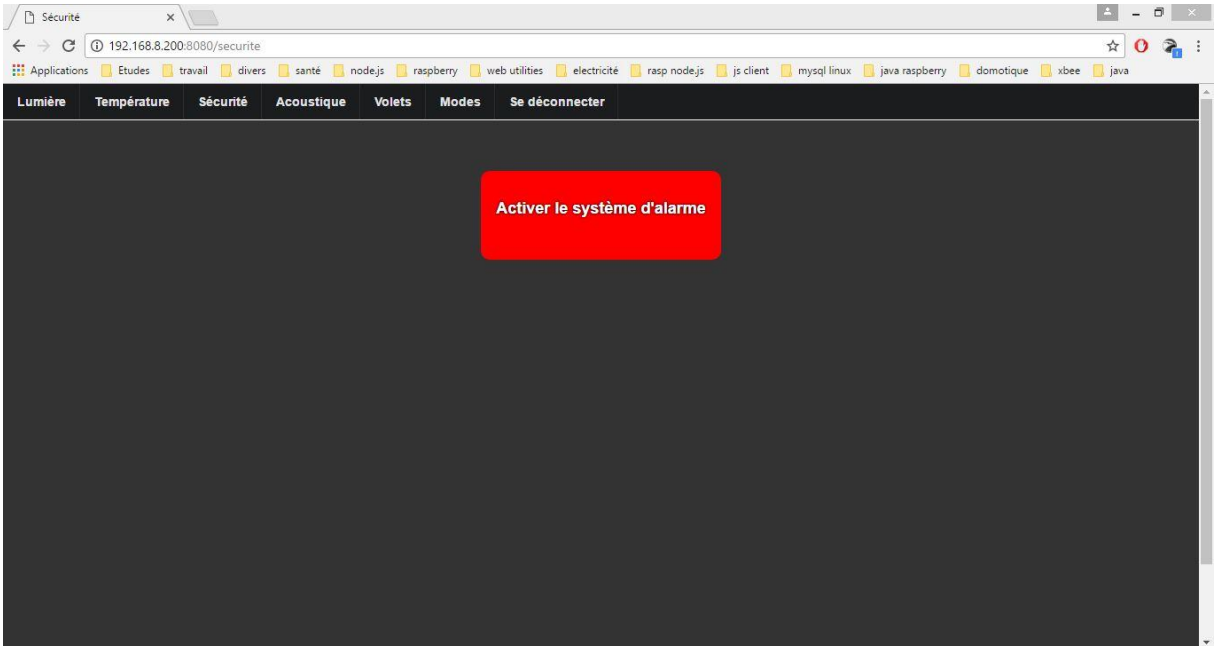


Figure 29 : Vue de gestion de la sécurité « système d'alarme désactivé »

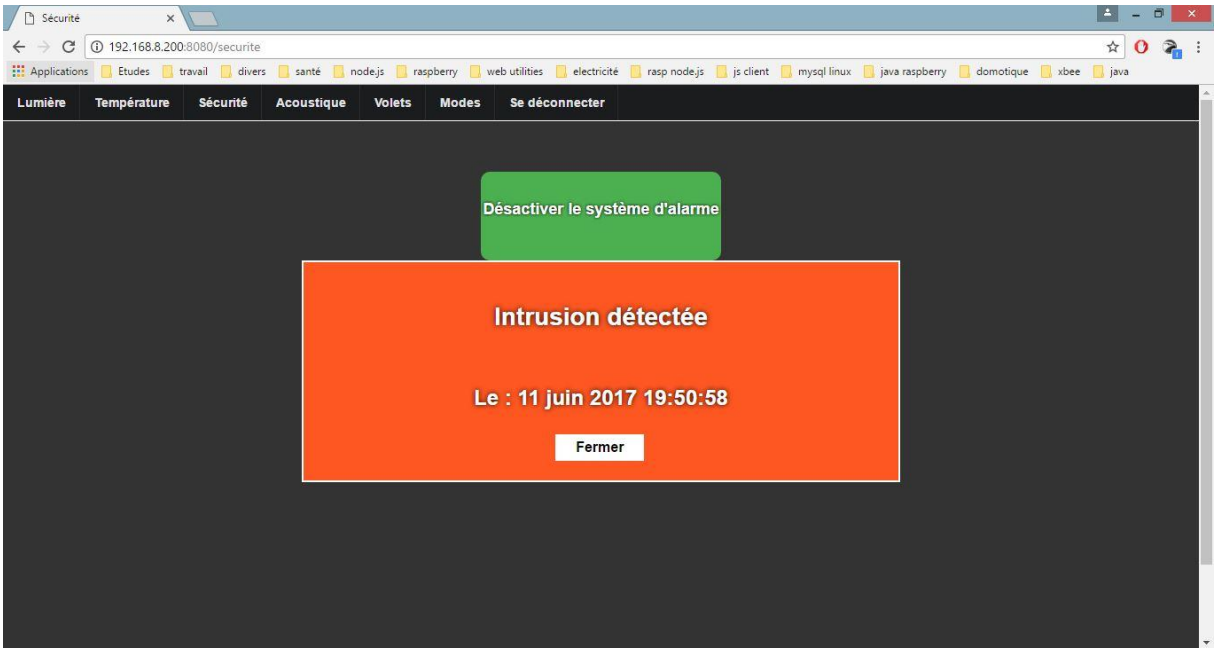


Figure 30 : Vue de gestion de la sécurité « Intrusion détectée »

3.5.6. Gestion des volets

Une fonctionnalité aussi importante que les autres, c'est de pouvoir contrôler les volets de la maison par un smartphone ou un PC. Cette fonctionnalité permet de contrôler les volets soit un à un soit tous à la fois. Notons, que cette fonctionnalité requiert des moteurs, capteurs de distance, capteurs de fin de course. Néanmoins, par manque de moyens, notre implémentation se limite à utiliser seulement un moteur pas à pas, et donc, pour ajuster le volet, nous avons dû utiliser un Timer.

Ci-dessous, la capture d'écran correspondant à l'interface graphique « gestion des volets ».

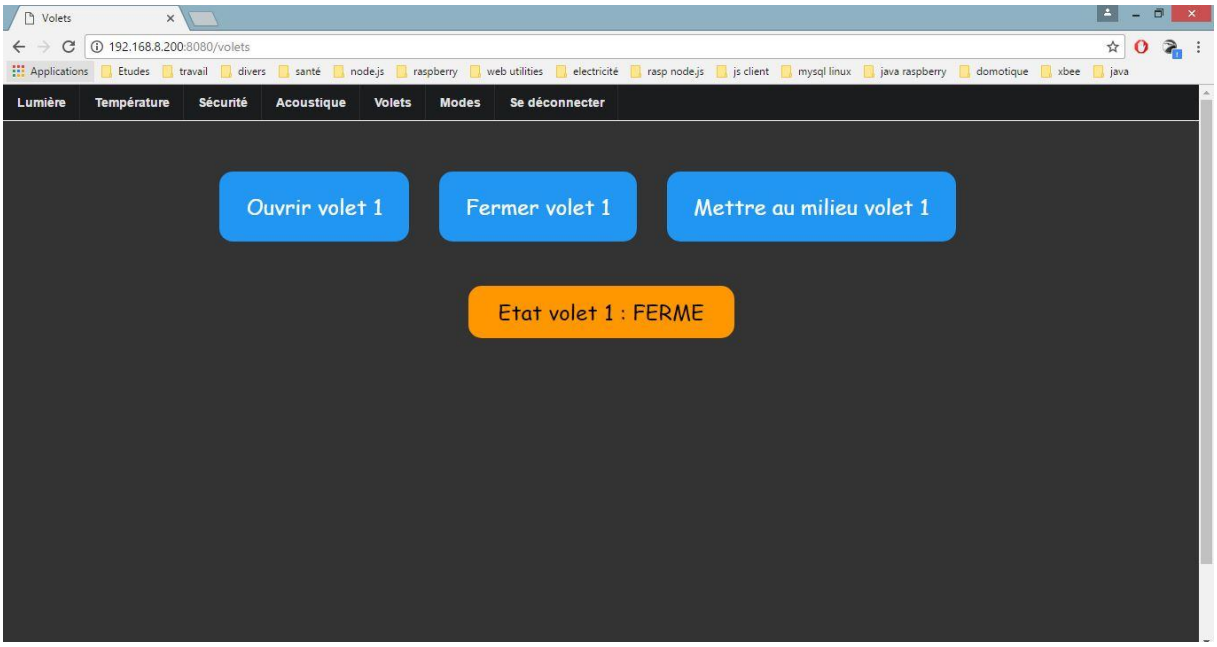


Figure 31 : Vue de gestion de volets

3.5.7. Gestion de l'acoustique

Afin d'apporter davantage de confort, nous proposons cette fonctionnalité qui permet d'ajuster l'ambiance acoustique. En effet, notre implémentation de cette dernière permet à l'utilisateur de contrôler la musique via une page web. La figure ci-après, montre l'interface graphique permettant la gestion de l'acoustique.

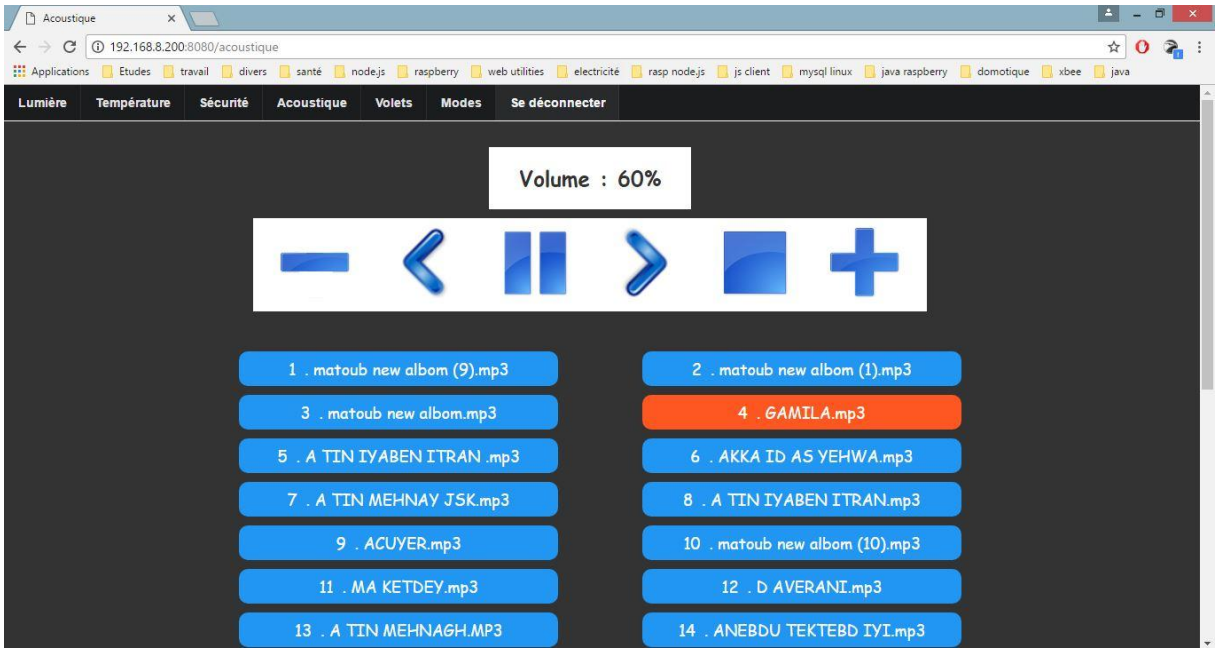


Figure 32 : Vue de gestion de l'acoustique

3.5.8. Gestion des modes

Ce module permet à l'utilisateur de créer des modes qui permettent de sauvegarder une certaine configuration de toutes les autres fonctionnalités, par exemple, il peut créer un mode « hors maison », qui consiste à éteindre toutes les lumières, fermer les volets, activer le système d'alarme …etc. et dès qu'il active ce mode, toute la configuration est exécutée à la fois. Ci-dessous, l'interface graphique correspondant à la gestion des modes.

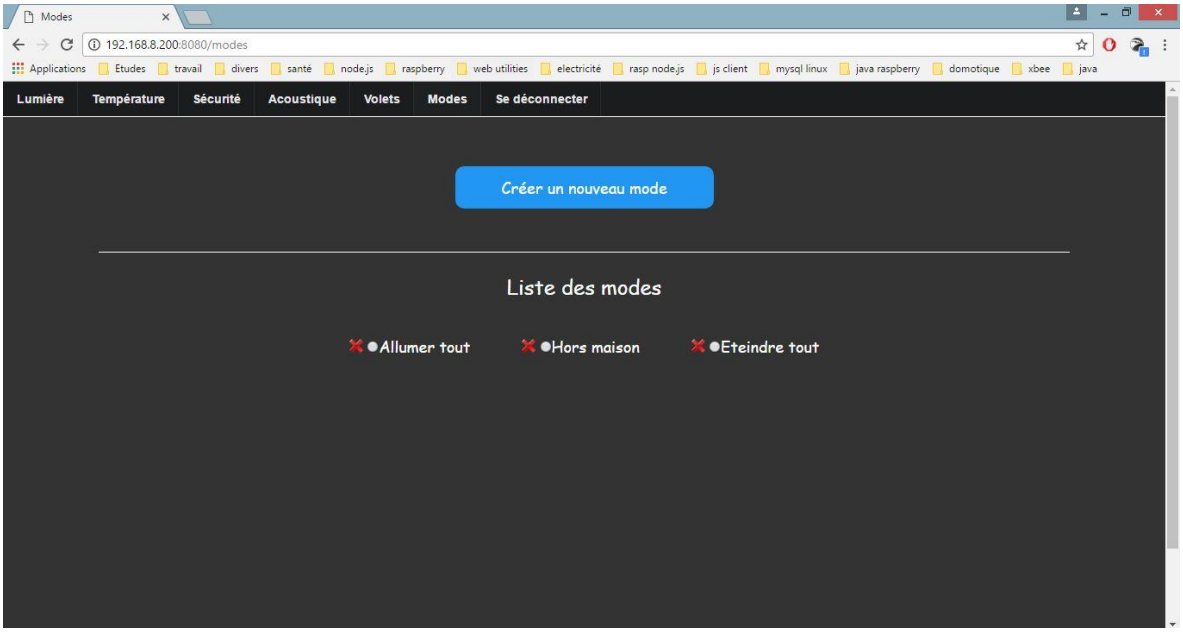


Figure 33 : Vue de gestion des modes

Conclusion

Dans ce chapitre nous avons vu les outils utilisés afin de mettre à terme le projet, ainsi que la contrainte principale du système, et comment nous avons réussi à en remédier. Nous aurions aimé programmer davantage de fonctionnalités, mais la domotique exige beaucoup du matériel, ce qui n'est pas à notre portée malheureusement, mais le principe de la solution reste toujours applicable au reste des fonctionnalités, soit en allume une lumière ou un chauffage, le principe reste toujours le même.

Conclusion générale

Grâce à ce projet, nous avons pu nous familiariser avec des concepts importants et intéressants tels que l'internet des objets, la domotique, websocket…etc. Il nous a aussi permis d'apprendre davantage sur les technologies les plus populaires au monde telles que Java, Raspberry, XBee...etc.

En outre, ce projet nous a poussés à affronter des problèmes complexes qui ont dû consolider nos connaissances théoriques et améliorer notre savoir-faire.

Enfin, nous aurions aimé développer davantage de fonctionnalités, mais la domotique requiert énormément de moyens, ce qui n'est pas à notre portée, néanmoins, cela sera comme une perspective.

Annexe

Annexe

1. Les principes de SOLID

En matière de développement logiciel, on constate aujourd'hui que le design reste principalement une affaire de style personnel et d'expérience. Les principes de base de l'objet que sont l'encapsulation, l'héritage et le polymorphisme ne suffisent pas à guider le design.

Il existe pourtant des principes extrêmement utiles en matière de design. Les principes sont listés ici en trois groupes principaux:

- Gestion des évolutions et des dépendances entre classes,
- Organisation de l'application en modules,
- Gestion de la stabilité de l'application.

Ci-dessous, nous présentons les principes de SOLID qui mènent à une conception qui respecte l'architecture logicielle :

- **Single Responsibility Principle (SRP)**
  - Une classe ne doit avoir qu'une seule responsabilité.
- **Open/Closed Principle (OCP)**
  - Programme ouvert aux extensions, fermé aux modifications.
- **Liskov Substitution Principle (LSP)**
  - Les sous-types doivent être substituables par leurs types de base.
- **Interface Segregation Principle (ISP)**
  - Éviter les interfaces qui contiennent beaucoup de méthodes.
- **Dependency Inversion Principle (DIP) :**
  - Les modules d'un programme doivent être indépendants.
  - Les modules doivent dépendre d'abstractions.

Annexe

1.1. Single Responsibility Principle

- **Principe**
  - Une classe ne doit avoir qu'une seule responsabilité.
- **Signification et objectifs**
  - Une responsabilité est une « **raison de changer** ».
  - Une classe ne doit avoir qu'une seule raison de changer.
- **Avantages :**
  - Diminution de la complexité du code.
  - Amélioration de la lisibilité du code.
  - Les classes ont plus de chance d'être réutilisables.

1.2. Open/Closed Principle

- **Principe :**
  - Programme ouvert pour l'extension, fermé à la modification.
- **Signification**

  Vous devez pouvoir ajouter une nouvelle fonctionnalité, en ajoutant des classes (ouvert pour l'extension), et cela sans modifier le code existant (fermé à la modification).
- **Avantages :**
  - Le code existant n'est pas modifie, ce qui augmente de la fiabilité.
  - Les classes ont plus de chance d'être réutilisables.
  - Simplification de l'ajout de nouvelles fonctionnalités.

Annexe

1.3. Liskov Substitution Principle (LSP)

- **Principe**
  - Les sous-types doivent être substituables par leurs types de base.
- **Signification**
  - Si une classe A étend une classe B (ou implémente une interface B) alors un programme P écrit pour manipuler des instances de type B doit avoir le même comportement s'il manipule des instances de la classe A.
- **Avantages**
  - Meilleure organisation du code.
  - Modification locale lors des évolutions.
  - Augmentation de la fiabilité.
  - Les classes ont plus de chance d'être réutilisables.

1.4. Interface Segregation Principle

- **Principe**
  - Éviter les interfaces qui contiennent beaucoup de méthodes.
- **Signification et objectifs :**
  - Découper les interfaces en responsabilités distinctes (SRP).
  - Quand une interface grossit, se poser la question du rôle de l'interface.
  - Éviter de devoir implémenter des services qui n'ont pas à être proposés par la classe qui implémente l'interface
  - Limiter les modifications lors de la modification de l'interface.

- **Avantages**
  - Le code existant est moins modifié ce qui augmente de la fiabilité.
  - Les classes ont plus de chance d‘ être réutilisables.
  - Simplification de l'ajout de nouvelles fonctionnalités.

1.5. Dependency Inversion Principle (DIP)

- **Principe**
  - Les modules d'un programme doivent être indépendants.
  - Les modules doivent dépendre d'abstractions.
- **Signification et objectifs :**
  - Découpler les différents modules de votre programme.
  - Les lier en utilisant des interfaces.
  - Décrire correctement le comportement de chaque module.
  - Permet de remplacer un module par un autre module plus facilement.
- **Avantages**
  - Les modules sont plus facilement réutilisables.
  - Simplification de l'ajout de nouvelles fonctionnalités.
  - L'intégration est rendue plus facile.

2. Les patrons de conception (design pattern)

Les patrons de conception sont des solutions qui décrivent et résolvent un problème général et qui est adaptable à un contexte particulier. Cette notion a vu le jour en 1994 grâce au livre « Design patterns: elements of reusable object-oriented software » (Gamma, Helm, Johnson et Vlissides, dit le Gang of Four : GoF).

- Un DP est ainsi défini par :
  - **un nom**
  - **une intention** *:* description du problème résolu par le DP

- **une structure** : solution proposée, souvent décrite par un diagramme UML
- **ses propriétés** *:* indiquent comment le DP fonctionne et ce qu'il apporte.

Le GoF présente 23 design pattern organisés en 3 classes :

- **Patrons de création**
  - ils ont pour but de gérer les problèmes de création de nouveaux objets : *Abstract Factory*, *Builder*, *Factory Method*, *Prototype*, *Singleton.*
- **Patrons de structure**
  - ils servent à organiser les informations dans un graphe d'objets : *Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy.*
- **Patrons de comportement**
  - ils servent à maîtriser les interactions entre objets : *Chain of responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor, Callback.*

Bibliographie et Webographie

[1] http://www.commentcamarche.net/contents/498-html-langage.

[2] http://igm.univ-mlv.fr/~dr/XPOSE2010/WebSocket/proto_bref.html.

[3] Livre Blanc des objets communicants et Internet des objets, Association institut Carnot.

[4] Aspects techniques de l'internet des objets. MM. KOFFI FIAWOO ET SLIM TOUHAMI.

[5] L'Internet des objets, Comment l'évolution actuelle d'Internet transforme-t-elle le monde ?, Dave Evans.

[6] « DOMOTIQUE ET CONFORT : UN ÉTAT DES LIEUX », Arthur Gential, école d'architecture de Lyon.

[7] Système domotique, Legrand.

[8] Le guide de la maison et des objets connectés, Cédric Locqueneux, Eyrolles.

[9] Modéliser le concept de confort dans un habitat intelligent : du multisensoriel au comportement, Mathieu Gallissot.

[10] www.raspberrypi.org.

[11] www.digi.com.

[12] www.digi.com, Xbee libray for java, user guide..

[13] www.digi.com, Xbee-Quick-Reference-Guide.

[14] www.digi.com, Xbee-Manual.

[15] www.oracle.com.