REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Echahid Hamma Lakhdar – El Oued

Faculté des Sciences Exactes
Département d'informatique

N°d'ordre :.....
Série :.....

Mémoire

Présenté en vue de l'obtention du diplôme de Master en Informatique
Option : **Intelligence Artificielle et Systèmes Distribués**

Middleware Orienté Microservice pour les Applications Internet des Objets

Par :

GUEDIRI Mossaab
HAMIED Ali

Soutenu le : 12/10/2017

Devant le jury :

Mr. ABBAS Messaoud	Maître Assistant A	Université d'El Oued	Président
Mr. BALI Mouadh	Maître Assistant B	Université d'El Oued	Examinateur
Mr. BEGGAS Mounir	Maître de Conférences B	Université d'El Oued	Encadreur
Mr. OTHMANI Samir	Maître Assistant A	Université d'El Oued	Co-Encadreur

Middleware Orienté Microservice pour les Applications Internet des Objets

GUEDIRI Mossaab
HAMIED Ali

27 septembre 2017

Remerciements

Tout d'abord, nous tenons à remercier Dieu le tout puissant, nous rendons grâce pour nous avoir donné la volonté, la santé et la patience pour élaborer ce travail.

En suite, nous remercions notre encadreur qui nous a soutenus durant toute notre mémoire.On aime lui adresser toute notre profonde gratitude pour toutes ses suggestions et conseils.

On veut aussi témoigner l'ensemble de personnel du département informatique de l'université d'EL Oued et sur tout son chef département, pour avoir nous bien accueillir et faciliter toutes les procédures administratives.

Enfin, un grand remerciement pour tous ceux qui nous ont aidé, soutenu et encouragé pour achever ce travail.

Resumé

Dans L'Internet des objets (IoT) un nombre grandissant d'objets capables d'acquérir des données depuis leur environnement et d'agir sur celui-ci à travers le réseau internet. Généralement, les objets sont dotés de capacités de calcul et de communication. Ces derniers peuvent être des senseurs ou des capteurs accessibles par un grand nombre d'utilisateurs et d'objets connectés qui sont trop occupées pour y demander l'information. Cela nécessite de trouver un environnement qui gère toutes les communications à l'objet sans aucun problème.

Ce projet propose d'étudier et de résoudre les problèmes susmentionnés en adaptant un middleware orienté microservice pour que objets connectés puissent être présentés comme des services et de fait, réduire le couplage entre ces services et leurs hôtes de façon à abstraire leur nature hétérogène. La contribution principale de ce projet porte sur la conception d'un conteneur qui empêche les communications directes avec les hardwares, et assure de répondre à toute les communications.

Le conteneur ou le middleware a été mise en œuvre au sein de Framework OSGi qui a des avantages de la programmation modulaire en java ; la gestion transparente de dépendances entre composants et l'architecture orientées services.

Nous avons implémenté un middleware et analysé ses performances au travers de nombreuses expérimentations qui démontrent que les solutions que nous proposons sont viables et pertinentes.

Abstract

In the Internet of Things (IoT), a growing number of things able to acquire data from their environment and act on it through the internet. Generally, objects have computing and communication capabilities. These can be sensors are connected with a large number of users and connected objects that are too busy to request information. This requires finding an environment that handles all communications to the object without any problems.

This project proposes to study and solve the above problems by adapting the microservice oriented middleware for the sensors and the objects can be presented as services and, in fact, reduce the coupling between these services and their hosts in order to abstract their heterogeneous nature. The main contribution of this project is the design of a container that prevents direct communication with the hardwares, and ensures that any communication is answered.

Container or middleware has been implemented within OSGi Framework which has advantages of modular programming in java, transparent management of component dependencies and service-oriented architecture.

We have implemented a middleware and analyzed its performance through numerous experiments that demonstrate that the solutions we propose are viable and relevant.

الملخص

في إنترنت الأشياء أو الكائنات ، عدد متزايد منها قادرة على الحصول على البيانات من بيئتها والتعامل معها من خلال شبكة الانترنت. عموما، الكائنات لديها قدرات الحساب والاتصال ، هذه الأخيرة يمكن أن تكون أجهزة الاستشعار متصلة مع عدد كبير من المستخدمين وأجسام أخرى متصلة ، و التي هي منشغلة جدا من جراء الطلب الكبير المعلومات. وهذا يتطلب إيجاد بيئة تتعامل مع جميع الاتصالات مع الكائن دون أي مشاكل.

يقترح هذا المشروع دراسة لحل المشاكل المذكورة أعلاه عن طريق تكييف وسيط الموجه لأجهزة الاستشعار والكائنات والتي يمكن تقديمها كخدمات، وفي الأساس، الحد من الاشتباك بين هذه الخدمات ومضيفيهم عن طريق تجريد طبيعتها غير المتجانسة. المساهمة الرئيسية لهذا المشروع هو تصميم حاوية تمنع التواصل المباشر مع الاجهزه، وتضمن أن يتم الرد على أي اتصال. وقد تم تنفيذ الحاوية أو الوسيط ضمن إطارOSGi والتي لديها مزايا البرمجة بالوحدات في جافا، وإدارة شفافة للتبعيات بين الوحدات المكونة والهندسة الخدماتية.

قمنا بتنفيذ الوسيط وتحليل أدائه من خلال العديد من التجارب التي تثبت أن الحلول التي نقترحها قابلة للتطبيق وذات صلة.

Sommaire

Liste des figures

Introduction Générale

Dans l'internet des objets, les objets hardwares sont connectés à internet et peuvent se communiquer entre eux et avec des entités softwares à travers le réseau internet. Généralement, les objets sont trop sollicités pour y demander l'information, ce qui pose un ensemble de problémes liés à la gestion de l'information, la gestion des accés, la securité et autres.

L'objectif de ce travail est de proposer un modèle de middleware orienté microservice pour jouer le rôle de conteneur de service s'occupant de la communication avec l'objet et toute communication à cet objet se fait à travers ce middleware et de prendre en compte les limites presentés ci-dessus.

Le chapitre I présentera de manière générale le concept de l'internet des Objets ; la définition, le fonctionnement, l'évolution technolologique , les defis et enfin,on étudiera les différents impacts que l'Internet des Objets aura de manière générale sur les professionnels.

Dans le deuxième chapitre on présentera les microservices qui sont une approche très importante de l'informatique distribuée. On va voir leur définition, leurs caractéristiques, avantages et terminera par son architecture.

Dans le reste du mémoire on occurrence le troisième et le quatrième chapitre, on présentera la plate forme OSGi, en suite nous allons citer les outils techniques utilisés pour réaliser notre proposition et les resultats obtenus.Enfin, on terminera par une conclusion générale et un ensemble des perspectives envisagées.

Chapitre I

Etat de l'Art sur IoT

I.1 Introduction

Nous assistons progressivement à une transformation de notre environnement technologique : de plus en plus d'objets physiques sont connectés à l'internet. C'est ce que l'on appelle « l'internet des objets », une révolution en marché depuis plus d'une dizaine d'année.

A l'heure actuelle, l'internet des objets est dans une phase de croissance rapide. Le nombre d'objets connectés a triplé depuis cinq ans et il est estimé à 24 milliards pour 2020. [1]

Les entreprises s'attendent en effet à ce que l'internet des oďjets devienne une grande source de revenus ; elles ont donc prévu d'investir 06 trillions de dollars dans les cinq prochaines années et en attendent un retour d'investissement pas moins de 13 trillions de dollars. [2]

Dans cette premiere partie du mémoire, nous suivrons la logique suivante :

- Dans un premier temps, nous analyserons les informations pertinentes concernant l'internet des objets, concepts que nous définirons et dont nous rendrons compte de l'ampleur, des bénéfices attendus et des challenges liés.

- Dans un second temps, nous analyserons l'impact de ces éléments sur les prestataires de services.

I.2 Internet des Objets

Dans ce chapitre, nous allons d'abord analyser le concept de l'internet des oḑjets, en procédant en quatre étapes : quelle est la définition, quel est le fonctionnement technique, quels sont les prochains développements attendus et quels sont les challenges liés. Ensuite nous allons décrire les principaux impacts de ces développements sur les acteurs professionnels.

I.2.1 Concept

I.2.1.1 Definition

Une des inventions les plus significatives du monde moderne, l'internet, est une réalité qui est disponible pour l'utilisation publique depuis environ 25 ans. Un de ses aspects les plus importants est ses nombreuses perspectives d'utilisations. En effet, de nouvelles applications et technologies dépendantes de l'internet se développent tous les jours.

A ses débuts, internet était principalement un protocole utilisé pour transmettre des messages d'un point à l'autre au moyen d'ordinateurs. Aujourd'hui, on est passé d'un internet connectant les gens et les ordinateurs vers un internet connectant les « choses » et les « objets ». Ce phénomène est appelé « Internet des Objets », pour lequel nous utiliserons son abréviation anglaise :*IoT (Internet of Things).*

Van Kranenburg en 2008 définit ce phénomène comme suit :*« une infrastructure de réseau global dynamique avec des capacités d'auto-configuration basé sur des protocoles de communications interopérables où les « choses » virtuelles et physiques ont une identité, des attributs physiques, une personnalité virtuelle et qui utilisent des interfaces intelligentes intégrées dans un réseau d'information ».*[3]

De manière plus simplifiée, Coetzee et Eksteen en 2011 définissent ce terme comme étant : *« une vision où les objets deviennent une partie de l'Internet : où tous les objets sont uniquement identifiés et accessible par le réseau, où leurs positions et leurs statuts sont connus, où la notion de services et d'intelligences est ajoutée à l'Internet fusionnant ainsi le monde digital et physique, impactant tous les professionnels, les personnes et l'environnement social ».*[4]

En effet, depuis son apparition, l'IoT prend de plus en plus d'ampleur avec le développement des technologies. De plus en plus de domaines commencent à percevoir les nouvelles opportunités. Des chercheurs Italians Luigi Atzori, Antonio Iera et Giacomo Morabito développent dans leur étude "The Internet of Things : A survey" , pourquoi ce concept semble si confus.

Tout d'abord à leur avis l'une des difficultés provient des différentes visions que les parties prenantes (le business, les chercheurs, les organes de standardisation, ... etc) ont de ce que l'IoT signifie : chacun de ces acteurs adopte une vision différente de l'IoT dépendant de leurs intérêts, de la finalité et du contexte dans laquelle le terme est utilisé. Selon eux, les chercheurs de ce domaine l'ont défini selon deux axes : une perspective orientée « Internet » ou une orientée « Objets ».

Malheureusement, aucune de ces définitions ne semble définir l'ensemble du phénomène. Ils argumentent que les deux perspectives doivent être traitées ensemble et que de plus, elles doivent inclure la « sémantique », c'est-à-dire toutes les technologies qui permettent aujourd'hui de traiter de façon unique tous les « Objets » et d'exécuter des communications entre eux. Le paradigme de l'Internet des objets est alors le résultat de la convergence entre ces trois axes : Internet, Objet et Sémantique. [5]

Un autre problème réside dans la compréhension de ce que sont ces « Objets ». Selon Coetzee et Eksteen, pour pouvoir être défini comme un « Objet » de l'Internet des objets, il faut que ce dernier soit connecté à d'autres au moyen de Radio Frequency Identification (RFID) ou de tout autre type de communication sans fils. [4]

Les « Objets » dits intelligents ont le pouvoir de collecter des données, de produire et recevoir de l'information, et d'agir en influençant chacun leur environnement. La puissance de ces objets intelligents est d'autant plus effective s'ils sont couplés ensemble vers un but commun. Si cet énorme montant de données peut être géré et exploité intelligemment au travers de solutions intelligentes, de nouvelles opportunités de « data mining » pourront voir le jour. [6]

La dernière difficulté majeure à l'établissement d'une définition stable réside dans la difficulté du développement de normes communes à tous. En effet, afin de pouvoir définir de manière complète et commune un phénomène il est important que des normes communes soient développées. Ces dernières permettront d'établir son champ d'application, ses limites et surtout, sa vision. Or le développement de normes communes au différents pays du monde n'est pas chose aisée. [7]

I.2.1.2 Fonctionnement

Comment s'opère cette interopérabilité entre tous les différents objets ? Grâce au concept de « Service oriented Architecture » (SoA), tel que décrit par Xu, He & Li en 2014 [8]. Il s'agit d'un modèle d'architecture qui organise l'Internet des objets en quatre différentes couches (Layers) : Sensing Layer, Networking Layer, Service Layer et Interface Layer :

1. Premièrement, la couche « Sensing » regroupe tous les « hardwares » physiques nécessaires à la collection de données.
2. La seconde couche, la couche « Networking », a pour rôle de connecter entre eux tous les équipements.
3. Ensuite, la couche « Service » repose essentiellement sur les technologies de « middleware » permettant de réunir les « hardwares » et « softwares » sur une même plateforme.
4. Finalement, la dernière couche dite « Interface » s'occupe quant à elle de la présentation de l'application aux utilisateurs finaux, du design et de l'agrégation des différentes sources. Cette couche met en place des protocoles d'accès communs permettant ainsi l'accès aux différentes applications.

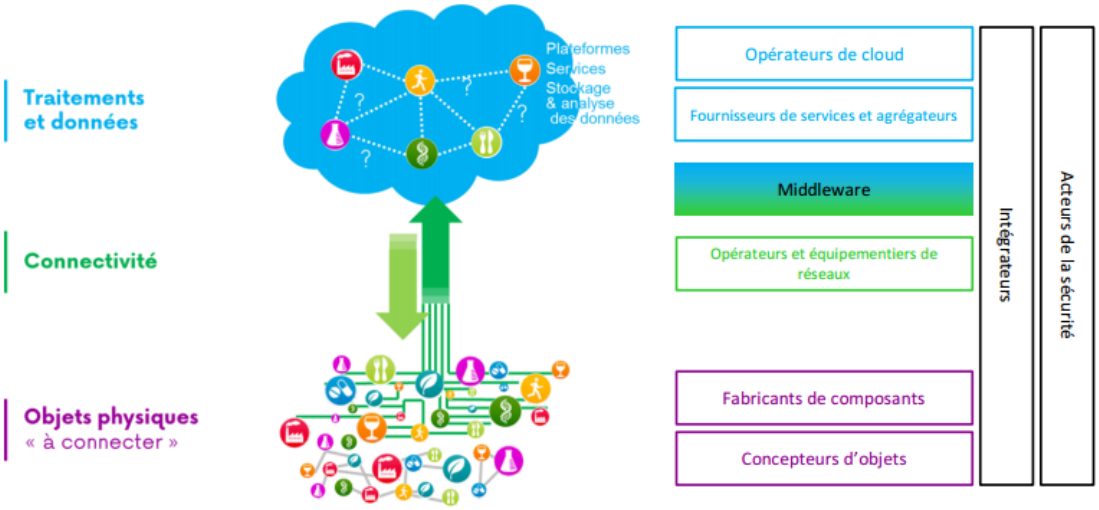

FIGURE 1.1: Differents Volets à considérer dans l'IoT

I.2.1.3 Evolution Technologiques Attendues

Le développement de l'Internet des Objets est rendu possible grâce aux avancées technologiques : les technologies en œuvre deviennent de plus en plus accessibles et intégrables partout. Il est utile d'en recenser les principales dimensions et dans quelle direction elles progressent.

Premièrement, l'explosion de la connectivité. Aujourd'hui, on recense environ 2 milliards de connections en 3G et 900 millions en 4G ; pour 2020 elles sont estimées respectivement à 3,6 et 2,3 milliards. Cet accroissement du taux de pénétration de la 4G va permettre aux Objets de se connecter entre eux et d'intéragir les uns avec les autres, mais d'autres formes de connectivité viennent renforcer cette explosion, à savoir le Wifi, le Bluetooth, le LoRa, et encore d'autres. De plus à l'heure actuelle des géants comme Google et Facebook œuvrent ensemble pour rendre l'Internet accessible partout et pour tous. [9]

Deuxièmement, le cout moyen d'un capteur est de environ 0,5 Dollars aujourd'hui soit moitié moins qu'il y a 10 ans, c'est à dire que le prix des capteurs ne cesse de décroître et il est prévu que ce prix continuer à diminuer jusqu'à 0,3 Dollars d'ici 2020. [10]

Troisièmement, la puissance des processeurs continue d'augmenter, et les prévisions sont que cette augmentation continuera, celle-ci postulant que la puissance double tous les 18 mois. Ceci permet bien entendu un plus large éventail de fonctionnalités aux futurs Objets.

Quatrièmement, les avancées dans la miniaturisation. A l'heure actuelle, on crée des processeurs de 1x1x1 mm, ces derniers pouvant être pourvus de différents équipements tels qu'une cellule solaire, de la mémoire, des capteurs de pression ou un appareil photo. [10]

Finalement, le développement du « Cloud » (stockage de données à distance) permet une capacité de stockage sans limite et dès lors favorise l'intégration et l'analyse des données. [10]

Ces avancées techniques vont permettre la progression de l'IoT telle que nous l'avons chiffrée dans l'introduction de ce mémoire : le nombre d'Objets connectés a triplé depuis cinq ans et il est estimé à 24 milliards pour 2020.

En conclusion de cette partie sur le fonctionnement technique et les évolutions prévues, retenons que la mise en place opérationnelle de l'IoT est un investissement dans plusieurs technologies complémentaires qu'il est complexe de mettre en œuvre.

I.2.1.4 Défis

Bien que l'internet des objets soit aujourd'hui déjà une réalité planétaire, certains défis sont encore à résoudre ; certains obstacles ralentissent son développement. Nous pensons notamment à des aspects de gouvernance (la mise en place de normes communes, la confidentialité), des aspects technologiques (l'interopérabilité des données, la sécurité, l'alimentation des capteurs) et enfin l'attitude des consommateurs.

Défis liés à la gouvernance : Tout d'abord, comme énoncé précédemment, l'internet des objets étant un phénomène international, le souci de développer des normes communes aux différents pays du monde est crucial. Qui régulera ce phénomène ? Comment développer différents standards communs ? Au vu de l'ampleur qu'aura l'IoT, une structure de gouvernance doit être mise en place. Le but est de développer des règles et principes qui permettront l'accroissement de l'efficacité. Il ne faudra cependant pas développer l'internet des objets seulement dans une logique de profit et de compétition, mais plus dans une optique favorisant le bien-être de la population et de la croissance économique internationale. [7]

Deuxièmement, depuis l'apparition de l'internet, les citoyens et souvent les états se sont continuellement battus pour le respect de la vie privée. Bien qu'aujourd'hui beaucoup de compromis aient été trouvés et que certaines solutions aient été apportées, cette question gagne en importance. Selon le « TRUSTe Internet of Things Privacy Index », seulement 22 % des utilisateurs d'Internet s'accordent pour dire que les bénéfices des « objets » intelligents surpassent le problème du respect de la vie privée [11]. Il est alors important que le détenteur de données ne puisse pas exploiter les données récoltées sans autorisation.

Défis liés à la technologie : Comme de plus en plus d'objets seront interconnectés, un volome important de données va être créé. A l'heure actuelle l'architecture des centres de données n'est pas encore préparée à interagir avec différentes sources hétérogènes. Une solution pourrait résider dans le développement de plateformes d'échanges (points de rencontre entre les différents acteurs) ; la nécessité de développer des plateformes communes est cruciale puisque ces dernières permettront l'échange de données et le rassemblement des différents acteurs au sein d'une même communauté d'objets connectés. Dès lors il est important que ces plateformes soient développées dans le but de favoriser l'interopérabilité entre les différents acteurs de cette révolution (fournisseurs, utilisateurs, etc.) aussi bien pour faciliter leurs interactions au sein de cette plateforme, que pour réguler l'échange de données. [12]

L'internet des objets pose un problème connexe qui se doit d'être réglé et normalisé à l'échelle internationale : la sécurité du réseau. 70 % des appareils IoT ont un manque crucial de sécurité. En effet, l'internet des objets est vulnérable à de nombreuses attaques pirates pour différentes raisons. Tout d'abord, à cause de la multitude de composantes, ces dernières ne pouvant être constamment sous surveillance et étant donc facilement sujettes à des attaques. Ensuite, les Objets de l'IoT sont constamment connectés à l'internet au moyen de communications sans fil : ces dernières sont sécurisées dans l'Internet au moyen de cryptages, c'est à dire de clés qui permettent de crypter les informations transmises par les Objets. Le problème est qu'actuellement les appareils de l'IoT ne sont pas encore assez puissants pour supporter ces chiffrements. Il est dès lors important

de développer de nouveaux algorithmes plus efficients et requérant moins de puissance. [13]

Le souci d'identification unique de tous les Objets connectés pose aussi problème. En effet, un des composants principaux à la sécurité d'un réseau est le principe d'identification unique des appareils. L'identification est l'action d'attribuer un numéro unique, appelée IP, à un objet pour empêcher toute confusion entre eux. Malheureusement, depuis le 03 février 2011, toutes les adresses IPv4 (4-bit) sont épuisées. Or cela représente un énorme frein au développement de l'internet des objets car ce dernier met justement en relation une multitude d'objet nécessitant une identification. Heureusement, des solutions sont déjà en cours de développement, telles qu'une nouvelle méthode d'adressage : l'IPv6. [7]

Ceci accentue un autre souci : la consommation énergétique.L'IPv6 est un ensemble de protocoles complexes coûteux en mémoire, et qui dès lors requiert une plus grande consommation électrique ; or l'alimentation électrique n'est à l'heure actuelle pas encore suffisante pour permettre au composants de l'IoT de supporter de tels schémas de sécurité. Il faudrait donc rendre les capteurs autosuffisants ; sans cela, le plein potentiel de l'internet des objets ne pourrait être exploité car il faudrait continuellement changer les « piles » de milliards de capteurs. Différentes recherches sont continuellement menées dans le but de trouver les technologies adéquates en terme d'autosuffisance et de prix. [14]

Défis liés à l'attitude des consommateurs : L'Internet des Objets fait aussi face à certains risques d'ordre psychologique : l'adoption de la part des utilisateurs. « Acquity Group » a réalisé une étude pour établir un état sur le comportement, à court terme et à long terme, des consommateurs quant à l'adoption des technologies de l'IoT [15]. Comme le démontre la Figure ci-dessous, l'adoption de ces technologies par les consommateurs va se généraliser.

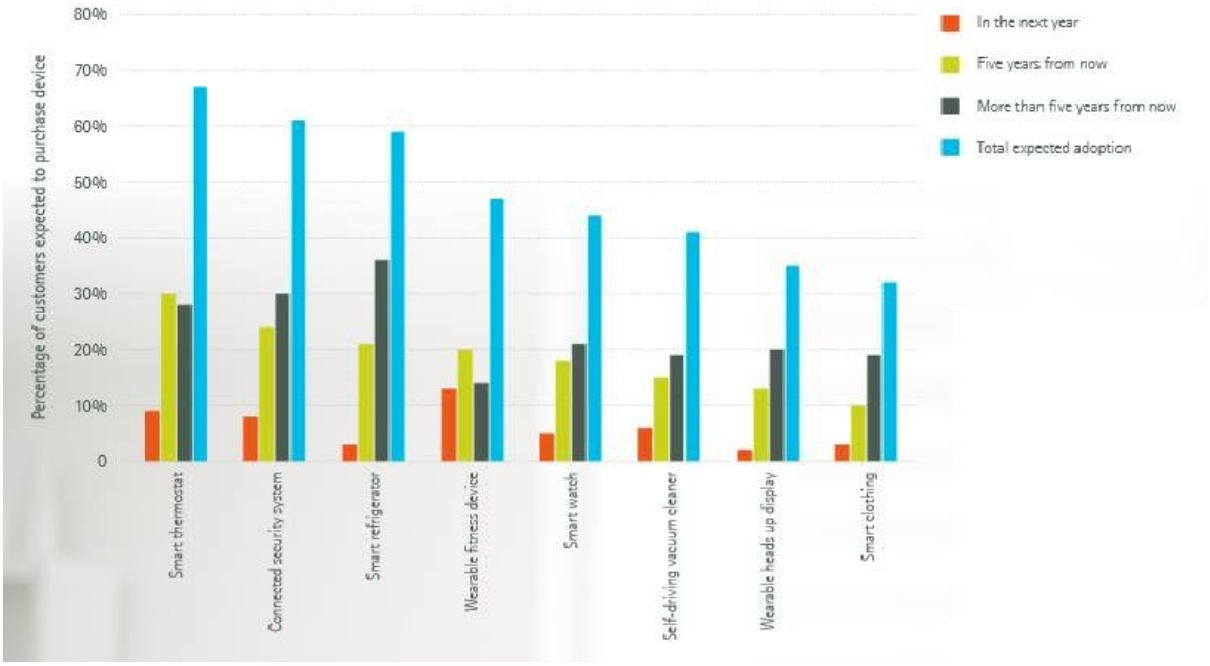

FIGURE 1.2: Adoption des technologies de l'IoT

Selon cette étude, il existerait quatre freins principaux limitant son taux d'adoption à court terme. Tout d'abord, l'enquête démontre que 87 % des répondants n'ont jamais entendu parlé du terme : Internet des Objets. Il y donc un souci de notoriété du terme. Ensuite, 30 % des répondants ne perçoivent pas encore la valeur ajoutée de l'IoT. Comme précisé précédemment, les problèmes liés à la sécurité et au respect de la vie privée freinent aussi les consommateurs (21 %) dans leurs achats. Finalement, 22 % des répondants indique que le prix de ces nouvelles technologies peut les dissuader de les acheter.

Bien que ces freins soient encore présents à court terme, les perspectives de croissance à long terme sont énormes. 65 % des consommateurs prévoient d'adopter une telle technologie à un moment dans le futur. En lien avec le « Technology Adoption Life-Cycle (TALC) » que nous utiliserons plus tard, l'étude indique que 92 % des consommateurs de masse utiliseront ces technologies d'ici 2019 et que 75 % des « later adopters » en achèteront dans les 5 prochaines années. [15]

Conclusion de la section « Défis de l'IoT » : Même si une définition universellement acceptée n'existe pas encore, cela ne saurait tarder compte tenu des challenges liés à la gouvernance. La définition que nous avons proposée est certainement un commun dénominateur simple et pertinent dans le cadre de ce mémoire. De manière plus générale, il ne fait aucun doute que les challenges seront rencontrés, comme en témoignent avec une confiance unanime les projections chiffrées concernant le développement de l'IoT. Enfin nous soulignons l'aspect complexe de la mise en œuvre de l'IoT du point de vue des acteurs professionnels, qui doivent nouer des partenariats pour la mettre en œuvre.

I.2.2 Impacts

Cette section étudiera les différents impacts que l'Internet des Objets aura de manière générale sur les professionnels. Ces différents impacts seront regroupés autour de 5 impacts majeurs, à savoir : les frontières entre les industries deviennent poreuses, les écosystèmes intégrateurs vont capter la valeur, accélération et renforcement des services, l'efficacité opérationnelle comme moyen de différentiation, et la création de nouveaux équilibres de la relation au client.

1. **Les frontières entre les industries deviennent poreuses :**
   Tout d'abord, l'apparition de l'Internet des Objets dans le monde industriel va déplacer et transformer les frontières entre secteurs et industries. En effet, les nouvelles possibilités en termes de récolte et d'exploitation de données va permettre aux acteurs de sortir de leur positionnement habituel. Par exemple, le constructeur d'automobiles va pouvoir étendre son activité en supportant le métier d'assureur : il va pouvoir collecter et analyser les différentes données de conduite et dès lors proposer des plans individualisés. Parallèlement, les petites entreprises vont pouvoir commencer à concurrencer les géants en place. En effet, les énormes volumes de données ne vont plus constituer une barrière à l'entrée puisque ceux-ci deviennent disponibles pour tous. La concurrence se relance. [10]

2. **Les écosystèmes intégrateurs vont capter la valeur :**
   Selon Gartner 2014, 30 % des données exploitables par les professionnels en 2017 proviendront de prestataires de services spécialisés qui chercheront à les monétiser. Bien que les données soient rarement gratuites, la clé pour les valoriser est la capacité à les relier à un écosystème d'agrégation. Or à l'heure actuelle seulement 5 percent des données sont exploitées par les entreprises . Dès lors un fossé va se créer entre les acteurs capables d'exploiter et d'analyser de manière intelligente et efficace les données, et les autres. La valeur a migrer vers ceux qui ont l'expertise requise pour exploiter leurs données. Par exemple, Amazon exploite toutes les données clients disponibles, alors que les distributeurs les exploitent peu. [12]

   Le « White Paper » de Bosch (2015) soutient l'idée que la pénurie en « data scientists » va faire apparaître des plateformes d'agrégations des données où les différents acteurs se rencontrent dans le but de mettre en commun leurs diverses expertises (outils d'analyse, gestion de flux de données, exploitation de données) en vue de produire des données pertinentes. On va donc vers plus d'interopérabilité car dans un monde connecté, les partenariats entre entreprises sont plus importants que jamais : aucune entreprise à elle seule ne peut affronter les challenges complexes induits par l'IoT. Et en effet, aujourd'hui de grands écosystèmes Google / Apple ou Samsung sont déjà en train de se développer. [10]

3. **Accélération et renforcement des services :**
   Selon Wyman (2015), l'Internet des Objets va accélérer les services. Bien que le déplacement des industries vers les services ait déjà démarré, l'IoT va radicalement accélérer le phénomène. O. Wyman (2015) propose une classification des nouvelles

directions pour les services favorisés par l'IoT :

- Certaines entreprises telles que Babolat et SEB tentent d'améliorer la qualité de l'expérience client aux moyens des technologies de l'IoT, on parle du « renforcement de la valeur du produit existant ».

- Les entreprises vont pouvoir offrir à leurs clients de nouveaux services révolutionnaires. Les nouvelles technologies de l'IoT vont faire apparaître de nouveaux services tels que les maisons connectées « Homelive » d'Orange. Les entreprises vont pouvoir aussi évoluer du produit vers de nouveaux services et de nouvelles solutions : la maintenance prédictive. Des entreprises telles que « Thyssen-Krupp » vont, en plus d'offrir leur « produit initial », fournir des services de maintenances prédictives permettant ainsi la diminution du risque opérationnel.

- L'IoT risque aussi de transformer la manière dont les entreprises créées de la valeur en permettant le développement de nouveaux concept comme le « Product As A Service » (PAAS). Les entreprises, telles que Hilti, vont pouvoir permettre aux clients de payer pour un produit en fonction de l'usage qu'ils en font plutôt que de l'acheter, c'est le PAAS.

- Au vu des différents challenges de standardisation des communications, de nouvelles entreprises vont émerger en proposant des « services sans objets ». En effet, les soucis d'interopérabilité entre les différentes marques créent de nouvelles opportunités pour les entreprises, à savoir le développement de plateformes permettant cette interopérabilité. Par exemple la plateforme « Wink » permet aux clients de connecter ensemble tous les appareils d'une maison.

- L'intégration des capteurs dans l'environnement physique va permettre de personnaliser un service à l'infini. En effet, l'IoT renforce le service « If This Then That » (IFTTT), c'est à dire un service qui permet de lier à une action une réaction, ouvrant ainsi la porte à des millions d'utilisations.

4. **L'éfficacité opérationnelle comme moyen de différentiation :**
   Le développement de l'Internet des Objets va permettre aux entreprises de se différencier au moyen de l'efficacité opérationnelle, c'est à dire en augmentant leur productivité tout en diminuant les coûts [5]. En effet, pour les industries cette rencontre entre le digital et le physique va permettre aux Objets d'interagir entre eux de manière autonome, sans interventions humaines. Les réseaux vont donc pouvoir être automatisés, la distribution va pouvoir être optimisée et la logistique va pouvoir suivre en temps réel ses différents produits.

5. **La création de nouveaux équilibres de la relation au client :**
   L'Internet des Objets va révolutionner la manière dans s'effectue la relation entreprise-client. Olivier Wyman (2016) souligne dans son étude deux changements qui vont modifier cette relation ainsi qu'approfondir la connaissance du client : [16]

- L'IoT va renforcer la connaissance du client ainsi que la relation que l'entreprise entretient avec celui-ci en le « voyant » plus souvent, et cela, à travers de plus nombreux points de rencontre. En effet, les nouvelles technologies de l'IoT vont permettre aux entreprises de capter l'expérience client de manière plus récurrente et dès lors, elles vont pouvoir adapter leur offre en temps réel. On passe du transactionnel au relationnel. Par exemple, Carrefour a mis en place un système de géolocalisation du client à l'intérieur du magasin, permettant ainsi de révolutionner l'expérience du client en magasin en lui proposant un service interactif en temps réel. [10]

- Suite au renforcement de cette relation, la chaine de valeur va se transformer. On se déplace progressivement d'un modèle B2B (Business-to-Business) vers un modèle B2B2C (Business-to-Business-to-Consumer) équilibré, on parle de ré-intermédiation partielle. Selon Wyman (2015), le rapport entre les trois acteurs de la chaine va s'équilibrer suite à cette nouvelle relation précisée précédemment. Bien qu'on aurait pu penser à un déplacement vers du pur B2C (Business-to-Consumer), il n'est pas nécessaire pour les fabricants de vendre et distribuer leur produit directement aux clients. En effet, en « B2B2C » ces derniers captureront de la valeur en augmentant la satisfaction et la fidélisation du client. [10]

I.2.2.1 Conclusion de la section « Impacts de l'IoT »

Nous constatons bien que l'IoT a un impact croissant sur les professionnels et qu'aucun d'entre eux ne peut ignorer cette évolution : dans le meilleur des cas ils ratent des opportunités considérables, dans le pire des cas leur propre business va être gravement attaqué. A terme, cette évolution menace donc la survie des acteurs impliqués qui ne considéreraient pas sérieusement comment se positionner face à l'émergence de l'IoT.

Chapitre II

Etat de l'art sur les Microservices

II.1 Introduction

Les microservices représentent une approche radicalement différente de la conception, du développement, du déploiement et de la gestion des applications monolithiques classiques.

L'informatique distribuée ne cesse d'évoluer depuis le milieu des années 1990, où on a vu l'essor de la technologie des composants avec Corba, DCOM et J2EE. Un composant était défini comme une unité de code réutilisable, aux interfaces immuables et partageables entre des applications distribuées.

L'architecture de composants était en rupture totale avec le mode traditionnel de développement d'applications, notamment au moyen de bibliothèques de liens dynamiques (DLL).

Toutefois, chaque technologie de composants utilisait un protocole de communication qui lui était propre (RMI pour Java, IIOB pour Corba et RPC pour DCOM) : l'interopérabilité et l'intégration des applications construites sur des plateformes aux langages différents n'en étaient que plus complexes.

II.2 Les Microservices

II.2.1 Definition

En informatique, les microservices sont un style d'architecture logicielle à partir duquel un ensemble complexe d'applications est décomposé en plusieurs processus indépendants et faiblement couplés, souvent spécialisés dans une seule tâche. Les processus indépendants communiquent les uns avec les autres en utilisant des APIs. Des API REST sont souvent employées pour relier chaque microservice aux autres. Un avantage avancé est que lors d'un besoin critique en une ressource, seul le microservice lié sera augmenté, contrairement à la totalité de l'application dans une architecture classique, par exemple une architecture trois tiers. Cependant, le coût de mise en place, en raison des compétences requises, est parfois plus élevé.[17]

Adrian Cockford, le concepteur de l'architecture de Netflix, l'une des architectures informatiques en microservices les plus avancées actuellement, a qualifié ce type d'architecture d'une SOA "à grain fin" car, dans un tel environnement, ce sont des centaines, des milliers de services qui sont instanciés chaque jour pour faire face aux requêtes des utilisateurs. Et chacune d'elles peut mobiliser des centaines de microservices pour délivrer la réponse demandée.

C'est, pour les fervents militants de cette nouvelle architecture, le seul moyen d'encaisser les montées en charge vertigineuses qu'un service Internet ou mobile peut connaitre lorsqu'il rencontre le succès à l'échelle mondiale. C'est un moyen de monter en charge de manière théoriquement infinie si on s'appuie sur le cloud, mais aussi un moyen d'assurer une haute disponibilité avec une répartition des données et des tâches sur plusieurs milliers de serveurs.

Puissante mais complexe, l'architecture en microservices demande de maitriser des solutions nouvelles sur lesquelles les compétences de haut niveau sont rares. Elle implique aussi de revoir la façon de concevoir, développer et déployer les applications.

Jérôme Mainaud, architecte logiciel chez Ippon Technologies, livre sa définition des microservices : *"c'est un style d'architecture logicielle dont les différents composants sont développés sous la forme de services autonomes qui s'exécutent avec leur propre processus. Ce modèle s'oppose ainsi aux architectures monolithiques où les composants écrits sous la forme de bibliothèque, modules ou classes sont tous déployés dans le même processus."* [18]

II.2.2 Evolution

Plus qu'une nouvelle révolution dans les architectures informatiques, les microservices sont une évolution de l'architecture SOA. Une évolution logique qui résulte d'un long retour d'expérience dans la mise en place d'architecture SOA, et qui vient gommer certains de ses défauts.

Il est des architectures informatiques comme de l'habillement : une mode chasse la précédente. Sans remonter à l'époque de Cobol et des mainframes, le client serveur a cédé la place aux architectures web qui a donné naissance au SOA. Aujourd'hui, l'architecture informatique en vogue, c'est *les microservices*.

Avec l'adoption des protocoles standard XML et HTTP pour la communication entre plateformes, l'architecture orientée services (SOA) a tenté de définir un ensemble de normes d'interopérabilité. Ainsi, les normes applicables aux services Web qui trouvaient leur origine dans le protocole SOAP (Simple Object Access Protocol) ont été confiées à un comité baptisé Oasis. Il est notamment composé de représentants de Computer Associates, Fujitsu, IBM, Novell et Sun. A travers cet effort , l'Oasis prévoyait simplifier les déploiements en environnements hétérogènes.

Des éditeurs comme IBM, Tibco, Microsoft et Oracle ont commencé à proposer des produits d'intégration applicative pour l'entreprise fondés sur les principes de l'architecture orientée services. Malgré le succès remporté par ces produits auprès des entreprises, les jeunes sociétés Web 2.0 (qui désigne le deuxième âge d'internet et du Web) ont commencé à se tourner vers le protocole REST (REpresentational State Transfer) pour l'informatique distribuée. Alors que JavaScript gagnait du terrain, JSON (JavaScript Object Notation) et REST sont rapidement devenus des normes de fait pour le Web.[19]

II.2.3 Caractéristiques des microservices

Les microservices sont des unités fonctionnelles de grande précision. Ils sont conçus pour réaliser parfaitement une seule chose. Chaque microservice comprend exactement un point d'entrée connu. Cela rappelle l'attribut d'un composant mais avec une différence majeure : l'assemblage.

Les microservices ne sont pas de simples bibliothèques ou modules de code : chacun constitue une unité fonctionnelle qui prend la forme d'un assemblage complet avec système d'exploitation, plateforme, framework, environnement d'exécution et dépendances. Chaque microservice est un processus autonome et indépendant qui ne dépend pas d'autres microservices. Il ne connaît pas ni ne reconnaît l'existence d'autres microservices.

Les microservices communiquent entre eux par le biais d'interfaces de programmation (API) indépendantes des plateformes et des langages. Ces API sont habituellement exposées sous la forme de terminaisons REST ou peuvent être appelées via des protocoles légers de messagerie. Leur modularité permet d'éviter autant que possible les appels synchrones et bloquants. [19]

II.2.4 Transition vers les microservices

Les applications actuelles s'appuient sur l'intégration et le déploiement continus pour obtenir une itération rapide. Pour tirer parti de cette approche, l'application est subdivisée en petites unités fonctionnelles indépendantes. Chaque unité est attribuée à une équipe qui est chargée de l'améliorer. Grâce à ce fonctionnement, les équipes peuvent livrer rapidement de nouvelles versions des microservices sans perturber d'autres parties de l'application.

L'évolution de l'Internet des objets (IoT) et de la communication machine-à-machine (M2M) exige de structurer différemment les modules d'application. Chaque module doit s'occuper d'une seule tâche qui intervient dans le workflow global. Pour chacune des parties d'une application, les développeurs choisissent les langages, les frameworks et les outils les plus appropriés.

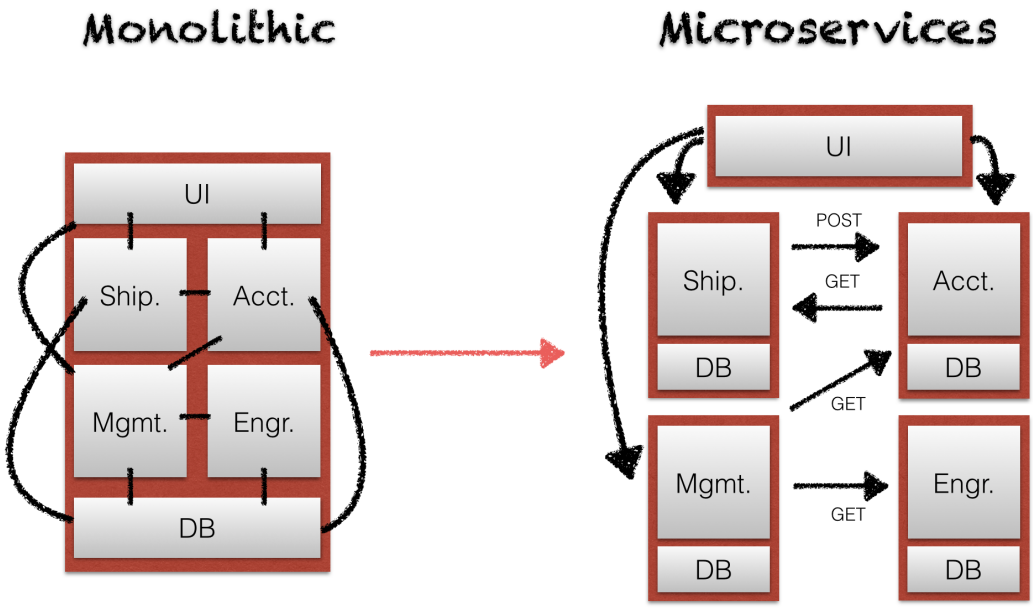

FIGURE 2.1: Transition vers les microservices

La technologie des conteneurs, permet de porter le code d'un environnement à l'autre. Les développeurs peuvent transférer en toute transparence le code écrit sur leurs machines de développement vers des machines virtuelles, ou vers le Cloud. Chaque conteneur qui s'exécute est un ensemble complet, du système d'exploitation au code d'exécution de la tâche.

L'infrastructure sous forme de code est un concept puissant : il permet aux développeurs de gérer l'infrastructure sous-jacente via un programme. On peut ainsi provisionner, configurer et orchestrer de façon dynamique quelques centaines de serveurs virtuels. Combinée aux conteneurs, cette fonctionnalité offre de puissants outils comme Kubernetes pour le déploiement dynamique de clusters exécutant des microservices.

Pour chacune des parties d'une application, les développeurs choisissent les langages, les frameworks et les outils les plus appropriés. Ainsi, une grande application peut se composer de microservices écrits en Node.js, Ruby on Rails, Python, R et Java. Chaque microservice est écrit dans le langage le plus adapté à la tâche.

C'est également le cas de la couche de stockage permanent. Les applications Web ont toujours plus besoin de stockage objet, de stockage de données semi-structurées et structurées, et de cache en mémoire pour garantir la persistance. Les microservices facilitent l'adoption d'une stratégie polyglotte en termes de langages de codage et de bases de données.[19]

II.2.5 Avantages des microservices

Avec les microservices, développeurs et opérateurs peuvent développer et déployer des applications auto-adaptatives. Chaque microservice étant autonome et indépendant, il est facile de détecter et de remplacer un service défectueux sans perturber d'autres services.

La technologie des microservices permet aux entreprises d'investir dans des composants modulables et réutilisables. A l'inverse des applications monolithiques, celles à base de microservices peuvent évoluer de manière sélective.

Au lieu de lancer plusieurs instances du serveur d'applications, on peut faire monter en charge un microservice donné à la demande. Quand la charge se déplace vers d'autres parties de l'application, un microservice utilisé antérieurement subit une baisse de charge parallèlement à la montée en charge d'un autre. On valorise mieux l'infrastructure sous-jacente : inutile désormais de provisionner de nouvelles machines virtuelles, il suffit de provisionner de nouvelles instances de microservices sur les machines existantes.

Les développeurs et les administrateurs pourront choisir les technologies de pointe qui fonctionnent le mieux avec tel ou tel microservice. Ils pourront combiner divers systèmes d'exploitation, langages, frameworks, environnements d'exécution, bases de données et outils de contrôle.

Enfin, la technologie des microservices permet aux entreprises d'investir dans des composants modulables et réutilisables. Chaque microservice fonctionne comme une brique de Lego qui peut être enfichée dans une pile d'applications. Si les entreprises investissent dans un jeu de microservices élémentaires, elles pourront ensuite construire par assemblage des applications répondant à diverses utilisations.[19]

II.2.6 Architecture Orientée Microservices

Les logiciels d'entreprise ont toujours été très volumineux. On attendait d'une application unique qu'elle gère les pics d'activités d'une entreprise toute entière. Toutes les fonctions requises par cette application devaient être effectuées par l'application elle-même. Les services ne représentaient que des fonctionnalités d'une structure monolithique. La base de données est un bon exemple de structure monolithique. Une entreprise peut en effet reposer sur une seule base de données.

Toujours est-il que plus les demandes informatiques se complexifient, plus la maintenance d'une telle structure est importante. Même les plus petits changements peuvent relever du défi, car modifier une partie d'une infrastructure monolithique implique de modifier l'infrastructure dans son ensemble. C'est de ce constat qu'est née une nouvelle approche de l'architecture que l'on nomme architecture orientée services (SOA). Dans une architecture SOA, il n'est plus question d'une application unique. Certaines fonctions peuvent être effectuées par différentes applications associées librement selon un schéma d'intégration, comme dans une architecture ESB (Enterprise Service Bus).

L'architecture SOA accroît la complexité des systèmes de l'environnement global. Certaines opérations ont certes été simplifiées dans les architectures SOA (comme l'introduction de nouveaux composants ou les mises à jour), mais si les interactions entre les composants ne sont pas parfaitement comprises, un changement risque d'avoir des répercussions sur l'ensemble de l'environnement.[20]

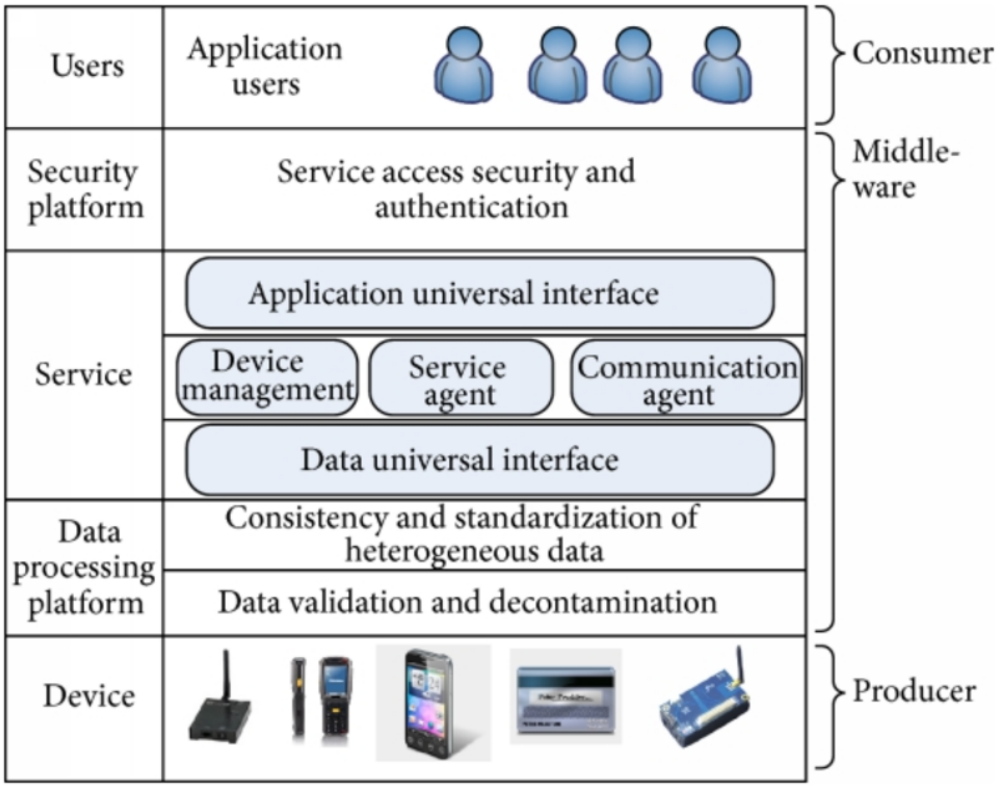

FIGURE 2.2: Une architecture IoT-middleware basée SOA

Au final, même si l'architecture SOA présente des avantages indéniables, elle ne peut rivaliser avec un modèle plus mature qui a petit à petit vu le jour : l'architecture orientée microservices. À l'instar des architectures SOA, les architectures orientées microservices disposent de services spécialisés en association libre, mais ces microservices ont été davantage décomposés. Dans une architecture orientée microservices, un service est clairement défini par les caractéristiques suivantes :

- un objectif unique et clair ;
- des paramètres bien définis ;
- une mise en œuvre « polyglotte ».

Les services de l'architecture utilisent une infrastructure de messagerie commune (comme les API REST) pour communiquer entre elles sans conversion de données ni couches d'intégrations supplémentaires. Le réseau de messagerie permet, et favorise même, l'accélération de la livraison des nouvelles fonctions et mises à jour. Chaque service est distinct. Il est possible de remplacer, améliorer ou supprimer un service sans affecter les autres services de l'architecture. Cette architecture légère aide à optimiser les ressources distribuées ou cloud et prend en charge l'évolutivité dynamique des services individuels.[20]

II.3 Conclusion

Les microservices ne sont pas la réponse à tous nos problèmes. Rien ne l'est. Ils ne sont pas la façon dont toutes les applications devraient être créées. Il n'y a aucune solution qui adresse tous les cas de figures.

Les microservices existent depuis longtemps et ont connu un regain de popularité depuis quelques années. Il y a plusieurs facteurs responsables de cette tendance, le dimensionnement étant probablement le plus important d'entre eux.

L'émergence de nouveaux outils et en particulier de Docker nous permettent de voir les microservices sous un nouveau jour et suppriment une partie des problèmes que leur développement et leur déploiement avaient crées.

L'utilisation des microservices par les « grands » comme Amazon, Netflix, eBay et d'autres, donne assez de confiance pour que ce style d'architecture soit prêt à être évalué (si ce n'est utilisé) par les développeurs d'applications professionnelles.

Chapitre III

La Technologie OSGi

III.1 Introduction

Cette introduction a pour objectif de fournir une definition à la technologie *OSGi (Open Service Gateway Initiative)*, technologie qui après avoir été longtemps utilisée dans le monde de l'embarqué est de plus en plus mise en oeuvre dans les applications classiques et serveurs.

Nous verrons dans un premier temps les différents concepts relatifs à OSGi et détaillerons à quelles problématiques cette technologie s'adresse tout en soulignant ses apports en terme d'architecture et de structuration des applications.

Nous rentrerons par la suite dans les aspects techniques d'OSGi afin de détailler ses caractéristiques et de voir concrètement comment mettre en oeuvre cette technologie au sein d'applications Java.

III.2 OSGi

Avant d'entrer dans le coeur de la technologie, nous allons détailler les caractéristiques de la programmation orientée composant et des architectures orientée service afin d'identifier les problématiques qu'elles visent à résoudre. Nous verrons alors que la technologie OSGi adresse ces deux types de technologies.

Tout d'abord, la programmation orientée composant vise à adresser les limitations de la programmation orientée objet. Cette dernière offre d'intéressants mécanismes afin de modulariser les traitements et les rendre réutilisables mais n'apporte aucun support afin de mettre en relation les classes. De plus, plus les traitements de l'application augmentent et se complexifient, plus ces limitations sont visibles et pénalisantes au niveau de la maintenabilité et de l'évolutivité de l'application.

Ainsi, cet aspect peut devenir très problématique lors de la mise en oeuvre de réseaux complexes d'objets. En effet, cela se traduit très souvent par des couplages forts entre objets, ce point nuisant énormément à la réutilisabilité des traitements. Chaque développeur a alors le choix des outils afin d'adresser au mieux au sein de ses applications ces aspects. L'approche la plus adaptée consiste en la mise en oeuvre du patron de conception injection de dépendance par l'intermédiaire de frameworks tels que Spring utilisés conjointement avec la programmation par interface.

En parallèle de la programmation orientée composant, les architectures orientées service peuvent être mises en oeuvre afin de mettre à disposition des traitements tout en diminuant les couplages entre les briques techniques. La SOA (Service Oriented Architecture) n'implique pas nécessairement l'utilisation des services Web avec les technologies SOAP et WSDL et peut être mise en oeuvre au sein d'un même processus Java notamment avec des conteneurs légers (entités mettant en oeuvre l'injection de dépendances).

Dans ce contexte, nous parlons de fournisseur de services et de consommateurs de services, les fournisseurs mettant à disposition des composants. La programmation orientée composant et les architectures orientées service peuvent donc être mises en oeuvre de manière complémentaire afin de bénéficier des avantages des deux types de technologies.[21]

La figure suivante illustre différents composants exposant des services par l'intermédiaire d'interfaces et reliés entre eux par des services :

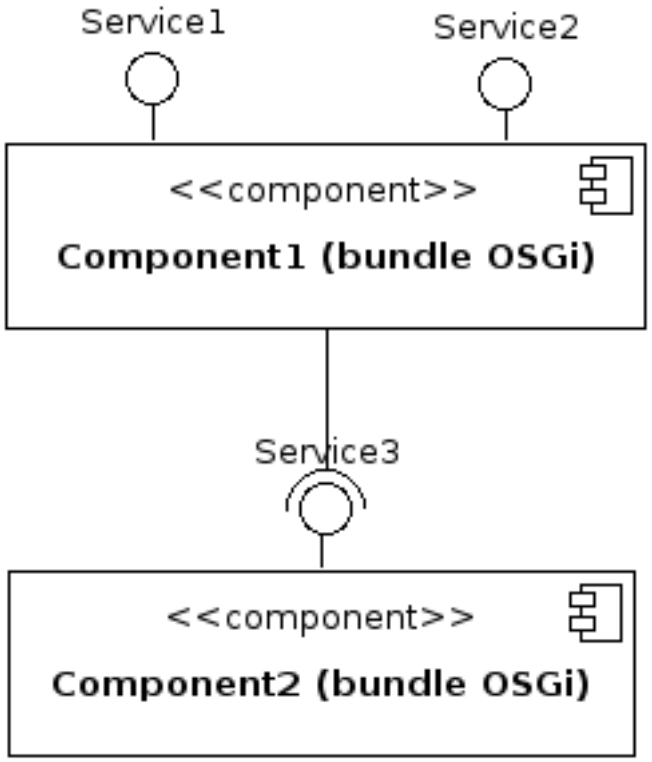

FIGURE 3.1: Différents composants exposant des services

Il est à noter que des composants OSGi peuvent être également reliés entre eux par l'intermédiaire des packages, packages qu'ils exportent et importent. Dans ce cas, les services ne sont donc pas l'unique manière de les relier. Dans ce contexte, la technologie OSGi vise à adresser les différentes problématiques vues précédemment, problématiques récapitulées ci-dessous :

- Adresser les limitations de la programmation orientée objet
- Permettre la gestion des applications complexes et de taille importante
- Améliorer la qualité de service des applications en permettant une administration à chaud
- Permettre la mise en oeuvre d'architectures orientées service légères

Une des autres caractéristiques de la technologie OSGi est sa portabilité puisqu'elle peut être mise en oeuvre aussi bien dans des terminaux de manière embarquée que dans des applications classiques ou serveurs. Le premier aspect a été à la base d'OSGi, les seconds types d'application s'étant développés ces dernières années par l'intermédiaire d'outils tels qu'Eclipse. Le fait qu'OSGi repose sur la technologie Java pour son exécution a été également un important facteur de portabilité.

III.2.1 Architecture OSGi

Dans cette section, nous allons décrire les caractéristiques des conteneurs et composants OSGi. Le composant correspond à l'entité centrale de la technologie et désignée par le terme bundle avec la terminologie de cette dernière. Nous verrons également les spécificités du cycle de vie des bundles, la manière de les gérer au niveau du conteneur ainsi que la façon de les configurer par l'intermédiaire de leur descripteur de déploiement.

La figure suivante décrit les différentes briques de l'architecture de la technologie OSGi, briques que nous allons détaillées tout au long de cette section :

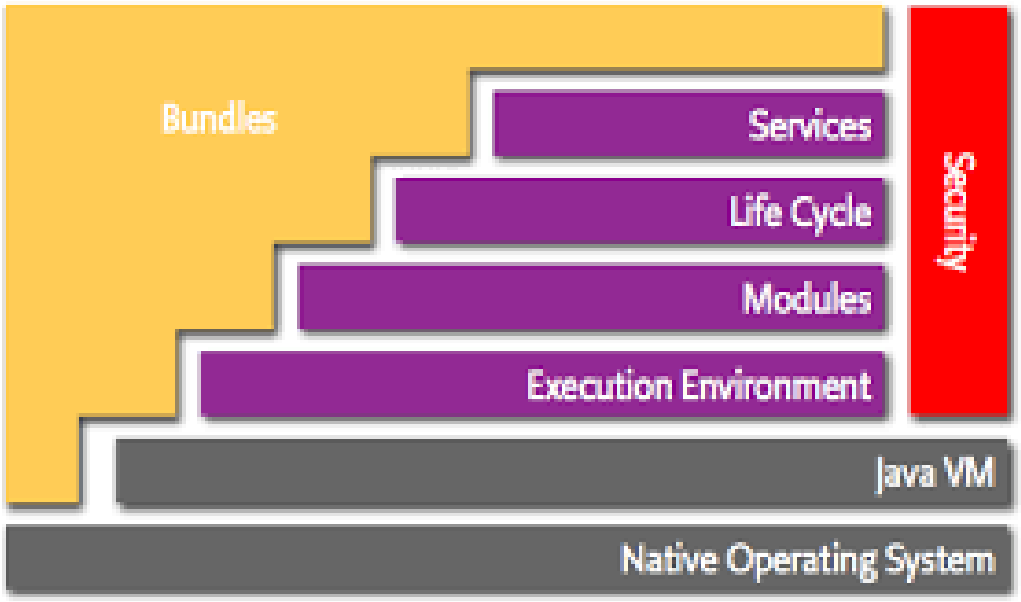

FIGURE 3.2: Architecture OSGI

La couche Module adresse la gestion des bundles, aussi bien au niveau du chargement des classes (classloading) que de la gestion de leurs visibilités et de leurs versions. La couche Cycle de vie (life cycle) prend quant à elle en charge les états des bundles supportés par le conteneur ainsi que les API correspondantes. Pour finir, la couche Services offre la possibilité de mettre à disposition des services au sein d'une même machine virtuelle tout en masquant leurs implémentations à leurs utilisateurs. Les applications OSGi peuvent se fonder et tirer parti de ces différentes couches afin de mettre en oeuvre des composants et des services.[22]

III.2.1.1 Gestion des composants OSGi

La couche module permet la mise en oeuvre des bundles dans le conteneur. Elle a la responsabilité de la gestion des différents chargeurs de classes et des versions des dépendances.

En effet, une des caractéristiques de la technologie consiste en l'isolation des classloaders des composants. En effet, chaque composant ou bundle OSGi dispose d'un classloader

indépendant pour ses traitements. Cet aspect permet par exemple la mise en oeuvre dans deux bundles différents d'une même bibliothèque avec deux versions différentes et incompatibles.

Cet aspect offre la possibilité à OSGi de complètement maîtriser les classes et packages du composant visibles depuis l'extérieur et visibles par les autres composants. Par défaut, la stratégie est la plus restrictive possible, à savoir que rien n'est visible. Par contre, lors de la mise à disposition d'un package d'un composant, un numéro de version spécifique peut être spécifié. Ainsi, deux versions de packages peuvent coexister dans le conteneur OSGi sans aucun problème.

III.2.1.2 Cycle de vie des bundles

Comme dans tout conteneur, les éléments contenus dans ce dernier possèdent un cycle de vie bien particulier puisque le conteneur OSGi a la responsabilité de les gérer. Nous pouvons distinguer deux grandes parties dans les différents états supportés :

1. **Non opérationnel :** Les états correspondent à la présence du bundle dans le conteneur, mais ce dernier n'est pas utilisable par les autres bundles pour leurs traitements ;
2. **Opérationnel :** Les états de ce type correspondent aux moments où le bundle est utilisable ou en phase de l'être ou de ne plus l'être.

La figure suivante illustre les différents états de gestion des bundles par le conteneur ainsi que leurs enchaînements possibles :

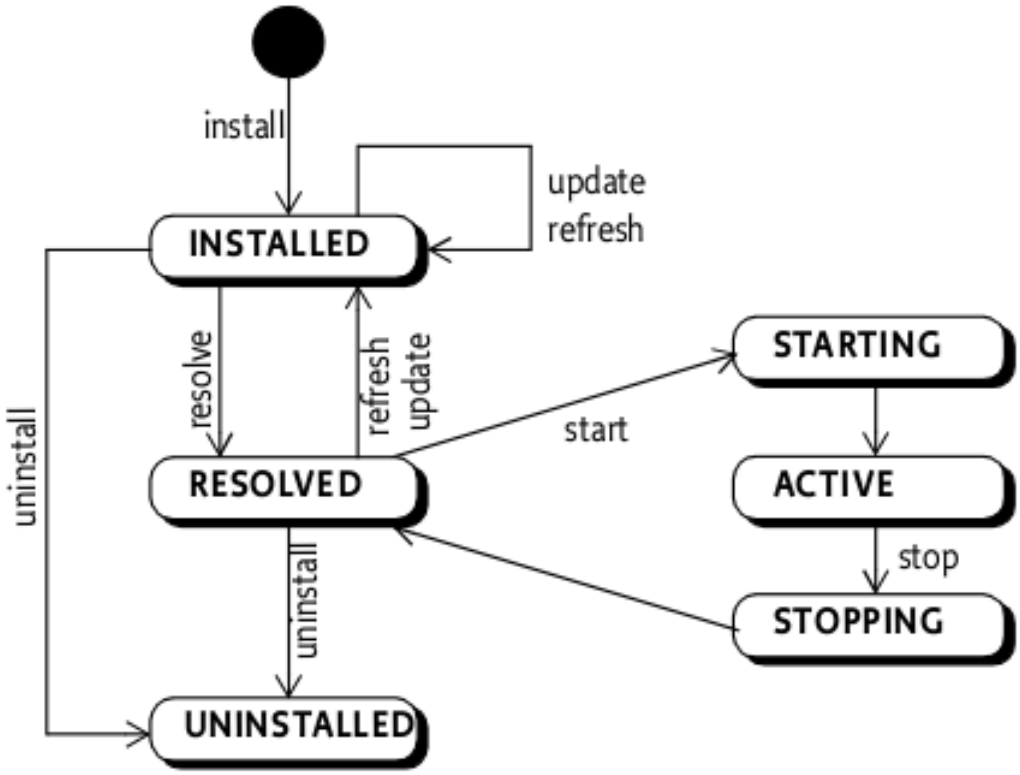

FIGURE 3.3: Différents états possibles des bundles

III.2.1.3 Services

La couche Services offre la possibilité de mettre à disposition certains traitements des composants par l'intermédiaire de services. Nous désignons par le terme service des traitements structurés de la manière suivante. Il s'agit d'une entité dont le contrat est clairement défini par l'intermédiaire d'une interface Java. Seule cette interface est connue par les consommateurs du service. La ou les implémentations du service doivent rester internes aux composants et implémenter la précédente interface.

Le mécanisme de gestion des services est complètement dynamique. En effet, dès qu'un service est enregistré, il est automatiquement utilisable par tous les composants disponibles dans le conteneur OSGi. Comme nous le verrons par la suite, une API est disponible afin d'enregistrer des implémentations de services. De plus, par convention, *il convient de faire correspondre le nom du service au nom de l'interface implémentée.*

Dans le cadre de la technologie OSGi, l'architecture orientée services a la caractéristique d'être très "légère" et de pouvoir fonctionner de manière autonome dans un seul processus Java. Elle correspond à un modèle de services dans une unique machine virtuelle (in-VM).

Nous verrons comment mettre en oeuvre concrètement des services dans le cadre de la technologie OSGi dans la section III.2.5 à la page 36

III.2.1.4 Conclusion - Architecture OSGi

L'architecture de la technologie OSGi permet de mettre en oeuvre des composants et de mettre à disposition des services applicatifs. Elle offre également un cycle de déploiement de services très rapides par l'intermédiaire du cycle de vie des bundles tout en offrant une meilleure structuration des fondations des applicatifs.

Nous pouvons remarquer que la sécurité est prévue en standard puisqu'elle fait partie intégrante de la spécification OSGi. Elle offre la possibilité de spécifier des permissions d'accès aux entités gérées par le conteneur. Une de ses principales caractéristiques consiste dans le fait que sa configuration est complètement dynamique et ne nécessite aucun redémarrage après une modification.

Rentrons maintenant dans le détail technique de la technologie OSGi et voyons concrètement comment la mettre en oeuvre.

III.2.2 Bundle OSGI

Dans cette section, nous allons détailler les différentes caractéristiques de l'entité centrale d'OSGi, le composant ou bundle. Nous verrons comment le mettre en oeuvre par l'intermédiaire d'un simple fichier JAR de Java.

III.2.2.1 Caractéristiques d'un bundle

Avec la technologie OSGi, le concept de composant est mis en oeuvre par l'intermédiaire des bundles, un bundle correspondant à un composant. Ces derniers permettent de mettre en oeuvre les différents concepts des composants et ce sont eux qui sont déployés dans les conteneurs OSGi.

Avec OSGi, un composant (bundle) est simplement stocké dans un fichier JAR de Java. Les informations de déploiement sont spécifiées par l'intermédiaire du fichier standard MANIFEST.MF présent dans le répertoire META-INF. En effet, OSGi reprend ce fichier en y ajoutant différents en-têtes afin de configurer le composant.

Il n'est pas nécessaire d'ajouter d'autres éléments au fichier JAR si ce n'est les classes bien sûr et les éventuelles bibliothèques utilisées, uniquement les bibliothèques n'étant pas définies en tant que dépendances. Nous pouvons constater que cet aspect favorise de manière importante la simplicité de mise en oeuvre de composants dans des conteneurs de ce type. Les conteneurs OSGi sont de ce fait très légers. N'oublions pas qu'OSGi était à l'origine utilisé dans des systèmes embarqués donc très soucieux de la consommation mémoire.

Les conteneurs possèdent diverses caractéristiques quant à la gestion des composants. Tout d'abord, ils ont la particularité de permettre la gestion des dépendances entre composants. En effet, comme nous l'avons vu précédemment, un composant s'appuie la plupart du temps sur d'autres afin de réaliser ses traitements et de fonctionner correctement. Pour ce faire, OSGi offre la possibilité de rendre visible dans le conteneur les packages de composants de manière très fine. La gestion des versions des dépendances est également supportée.

III.2.2.2 En-Têtes OSGi du fichier MANIFEST.MF

Comme nous l'avons précisé dans la précédente section, le fichier MANIFEST.MF correspond au descripteur de déploiement du bundle. Ce fichier est lu par le conteneur OSGi afin de configurer le bundle.

La spécification OSGi définit un ensemble d'en-têtes standards pour un composant, en-têtes utilisables dans le fichier MANIFEST.MF et dont la liste des principales est récapitulée dans le tableau suivant :

En-tête	Descriptif
Bundle-ManifestVersion	Correspond à la version de fichier MANIFEST du bundle
Bundle-SymbolicName	Spécifie l'identifiant symbolique du bundle
Bundle-Name	Spécifie le nom du bundle
Bundle-Version	Spécifie la version du bundle
Bundle-DocURL	Permet de préciser l'adresse de la documentation du bundle.
Bundle-Category	Spécifie la catégorie du bundle.
Import-Package	Spécifie les noms et les versions des packages utilisés par le bundle
Export-Package	Spécifie les noms et les versions des packages mis à disposition par le bundle
Require-Bundle	Spécifie les identifiants symboliques des bundles nécessaires au bon fonctionnement du bundle
Bundle-Activator	Spécifie le nom de la classe dont les traitements sont exécutés lors du démarrage et de l'arrêt du bundle
Bundle-Classpath	Spécifie le classpath du bundle

III.2.3 Interactions entre bundles

Comme nous l'évoquions dans la section précédente, un conteneur OSGi offre la possibilité de gérer la visibilité des ressources en composant. Par défaut, un composant est complètement opaque depuis l'extérieur et rien n'est accessible. Cet aspect implique de préciser toutes les dépendances utilisées par un composant et tous les packages mis à disposition.

La spécification OSGi offre la possibilité de jouer sur cet aspect par l'intermédiaire d'en-têtes dans le fichier MANIFEST.MF, fichier contenant les données relatives au déploiement et décrit dans la précédente section. Détaillons maintenant les spécificités de ces en-têtes et la manière de les mettre en oeuvre.[23]

Notons qu'il existe une manière complémentaire de faire interagir les bundles entre eux par l'intermédiaire des services. Nous décrirons cet aspect dans une prochaine section.

III.2.3.1 Mise à disposition de packages

Afin de mettre à des dispositions des packages (ou exporter) d'un bundle, l'en-tête Export-Package doit être mise en oeuvre dans le fichier MANIFEST.MF du composant. Cet en-tête contiendra la liste des packages pour lesquels l'accès est possible.

Il est à noter que la spécification d'un package dans ce cadre autorise l'accès à toutes les classes du package tout en gardant inaccessible les classes contenues dans les sous packages du package.

III.2.3.2 Importation de packages

La technologie OSGi offre la possibilité de spécifier les packages utilisés afin qu'ils soient visibles dans le bundle. Une fois spécifiées, les classes sont utilisables dans les classes

implémentées par le bundle. Cet aspect se configure dans le fichier MANIFEST.MF par l'intermédiaire des en-têtes Import-Package.

III.2.4 Principales API OSGi

Par cette section, nous allons introduire deux interfaces des APIs de la technologie OSGi relatives respectivement à l'entité d'activation et au contexte des bundles. Elles permettent notamment d'initialiser et de finaliser les ressources des bundles et d'interagir avec le conteneur OSGi. [23]

L'entité relative au contexte OSGi nous sera utile dans la section relative aux services puisque cette interface met à disposition des méthodes afin de manipuler et d'avoir accès aux services présents dans le conteneur OSGi.

III.2.4.1 BundleActivator - Entité d'activation

La technologie OSGi offre la possibilité de spécifier une entité appelée lors de l'activation (état Démarrage) et de la désactivation (état En cours d'arrêt) d'un composant, c'est-à-dire lorsque ce dernier est démarré ou arrêté. La classe implémentant cette entité doit nécessairement posséder l'interface BundleActivator et être configurée par l'intermédiaire de l'en-tête Bundle-Activator dans le fichier MANIFEST.MF.

L'interface BundleActivator définit la structure d'une classe d'activation en définissant les signatures des méthodes appelées lors des différents événements précédemment cités :

- **Méthode start** appelée lors du démarrage du composant (état Démarrage).
- **Méthode stop** appelée lors de l'arrêt du composant (état En cours d'arrêt).

Ces deux méthodes prennent en paramètre le contexte OSGi du composant par l'intermédiaire de l'interface BundleContext, interface permettant d'interagir aussi bien avec le conteneur qu'avec le composant. Le contenu de l'interface BundleActivator est décrit ci-dessous :

```
public interface BundleActivator {
          void start(BundleContext context);
          void stop(BundleContext context);
 }
```

Le code suivant illustre la mise en oeuvre d'une entité d'activation pour un bundle par l'intermédiaire d'une implémentation simple appelée ici SimpleActivator :

```
public class SimpleActivator implements BundleActivator {

       private ServiceRegistration serviceRegistration;

       public void start(BundleContext context) {
               // Enregistrement d'un service
               SimpleService service  = new SimpleServiceImpl();
               this.serviceRegistration = context.registerService(
               SimpleService.class.getName(), service, null);
       }
       public void stop(BundleContext context) {
               // Désenregistrement d'un service
               if( this.serviceRegistration!=null ) {
               this.serviceRegistration.unregister
               }
           }
 }
```

III.2.4.2 BundleContext - Contexte de composant

Nous avons vu dans la précédente section que l'entité d'activation s'appuie sur le contexte du composant. Détaillons maintenant les spécificités de l'interface BundleContext, interface relative au contexte du composant. Cette dernière permet d'interagir aussi bien au niveau du conteneur OSGi lui-même que du bundle dans lequel il est utilisé. Elle offre la possibilité de réaliser les différentes opérations suivantes :

Opération	**Descriptif**
Manipulation des bundles	Permet de récupérer les instances de bundles du conteneur et d'installer de nouvelles instances
Manipulation des services	Permet de récupérer les instances de services du conteneur et d'installer de nouveaux

Le contenu de l'interface BundleContext est décrit dans le code ci-dessous :

```
public interface BundleContext {

  void addBundleListener(BundleListener listener);
  void addFrameworkListener(FrameworkListener listener);
  void addServiceListener(ServiceListener listener);
  void addServiceListener(ServiceListener listener,
                          Java.lang.String filter);
  Filter createFilter(Java.lang.String filter);
  ServiceReference[] getAllServiceReferences(Java.lang.String clazz,
                                             Java.lang.String filter);
  Bundle getBundle();
  Bundle getBundle(long id);
  Bundle[] getBundles();
  Java.io.File getDataFile(Java.lang.String filename);
  Java.lang.String getProperty(Java.lang.String key);
  Java.lang.Object getService(ServiceReference reference);
  ServiceReference getServiceReference(Java.lang.String clazz);
  ServiceReference[] getServiceReferences(Java.lang.String clazz,
                                          Java.lang.String filter);
  Bundle installBundle(Java.lang.String location);
  Bundle installBundle(Java.lang.String location,
                       Java.io.InputStream input);
  ServiceRegistration registerService(Java.lang.String[] clazzes,
                                      Java.lang.Object service,
                                      Java.util.Dictionary properties);
  ServiceRegistration registerService(Java.lang.String clazz,
                                      Java.lang.Object service,
                                      Java.util.Dictionary properties);
  void removeBundleListener(BundleListener listener);
  void removeFrameworkListener(FrameworkListener listener);
  void removeServiceListener(ServiceListener listener);
  boolean ungetService(ServiceReference reference);
}
```

Détaillons maintenant les spécificités relatives aux bundles et aux observateurs. Cette interface offre tout d'abord la possibilité de gérer et récupérer les instances des bundles présents dans le conteneur. Tout d'abord, l'installation de bundles se réalise par l'intermédiaire des méthodes installBundle, méthode prenant en paramètre le chemin du fichier JAR du bundle. Les méthodes getBundle et getBundles retournent quant à elle une ou plusieurs instances de bundles. Sont supportées les récupérations du bundle courant, d'un bundle en se fondant sur son identifiant dans le conteneur ou de tous les bundles du conteneur.[23]

Le code suivant illustre la mise en oeuvre de ces méthodes dans une entité ayant accès au contexte OSGi :

```
// Installation d'un nouveau bundle
   Bundle bundleInstalle = contexte.installBundle(
       "file://com/master/OSGi/simplebundle.jar");
   long idBundle = bundleInstalle.getBundleId();

// Récupération d'une instance d'un bundle présent dans le conteneur
   Bundle bundle = contexte.getBundle(idBundle);
   int etatBundle = bundle.getState();
```

Comme vous pouvez le constater, le code précédent repose sur l'interface Bundle correspondant à l'entité relative à un composant OSGi. Cette dernière permet de récupérer des informations sur le bundle mais également de gérer son état. Le code suivant décrit le contenu de l'interface Bundle :

```
 public interface Bundle {
      public static final int UNINSTALLED = 0x00000001;
      public static final int INSTALLED = 0x00000002;
      public static final int RESOLVED = 0x00000004;
      public static final int STARTING = 0x00000008;
      public static final int STOPPING = 0x00000010;
      public static final int ACTIVE = 0x00000020;

      Enumeration findEntries(String path,String filePattern,
                                    boolean recurse);
      BundleContext getBundleContext();
      long getBundleId();
      URL getEntry(String path);
      Enumeration getEntryPaths(String path);
      Dictionary getHeaders();
      Dictionary getHeaders(String locale);
      long getLastModified();
      String getLocation();
      ServiceReference[] getRegisteredServices();
      URL getResource(String name);
      Enumeration getResources(String name);
      ServiceReference[] getServicesInUse();
      int getState();
      String getSymbolicName();
      boolean hasPermission(Object permission);
      Class loadClass(String name);
      void start() ;
      void start(int options);
      void stop();
      void stop(int options);
      void uninstall();
      void update();
      void update(InputStream in);
}
```

Il est ainsi possible par la programmation, après l'installation d'un bundle, de réaliser le démarrage du bundle et de récupérer les différents services mis à disposition par le bundle puis de l'arrêter, comme l'illustre le code suivant :

```
// Installation d'un nouveau bundle
Bundle bundleInstalle = contexte.installBundle(
    "file:///com/master/OSGi/simplebundle.jar");

// Démarrage du bundle (état installed vers active)
   bundle.start();

// Récupération des services mis à disposition
   ServiceReference[] serviceReferences = bundle.getRegisteredServices();

// Arrêt du bundle (état active vers resolved)
   bundle.stop();
```

Pour finir, le contexte offre la possibilité de spécifier des observateurs d'événements sur le conteneur OSGi lui-même, sur les bundles et sur les services. Cet aspect est couramment utilisé dans les bonnes pratiques de mise en oeuvre de la technologie OSGi, notamment par l'intermédiaire du patron de conception Extender au niveau des bundles.

Le principe consiste en la mise en oeuvre d'un observateur utilisant les données des bundles afin de réaliser des traitements. Ce patron permet de mieux gérer les états des bundles lors de traitements généraux, traitements pouvant être réalisés de manière paresseuse. Cette approche est mise en oeuvre dans des outils tels que Declarative Services, iPOJO, Spring Dynamic Modules.

Un observateur de bundles peut être mis en oeuvre par l'intermédiaire des interfaces BundleListener et SynchronousBundleListener, respectivement déclenchées de manière asynchrone et synchrone. Les deux interfaces mettent en oeuvre la méthode bundleChanged afin de notifier un changement pour un bundle. Cette méthode prend en paramètre un objet de type BundleEvent, objet permettant d'avoir accès au bundle impacté et à l'événement déclencheur dans le cycle vie. Le code suivant illustre le contenu de l'interface BundleListener :

```
public interface BundleListener extends EventListener {
  void bundleChanged(BundleEvent event);
       }
```

La mise en oeuvre d'un observateur de bundles se réalise de la manière suivante :

```
// Enregistrement de l'observateur
   BundleListener observateur = new BundleListener()  {

         public void bundleChanged(BundleEvent evenement) {
             // Récupération du bundle impacté \newline
             Bundle bundle = evenement.getBundle();\newline
             int etat = evenement.getType();\newline
             (...)\newline
          }
      };
    context.addBundleListener(observateur);
```

III.2.5 Mise en oeuvre de services

Comme nous l'avons évoqué précédemment, la technologie OSGi offre la possibilité de mettre des services au niveau des composants. Elle permet de bien séparer le contrat du service (interface) de la ou des implémentations fournies par les composants. Les consommateurs du service n'ont connaissance que de son interface. De plus, les services mis en oeuvre avec OSGi sont des simples POJOs.

La technologie OSGi, par l'intermédiaire de l'interface BundleContext, offre la possibilité à un composant de mettre à disposition des services par l'intermédiaire de la méthode registerService. Le premier paramètre de cette méthode correspond au nom du service, nom visible au niveau du conteneur. Un usage courant consiste à spécifier le nom de l'interface du service comme nom du service.

Le code suivant décrit la manière d'enregistrer un service dans un conteneur OSGi par l'intermédiaire du contexte OSGi :

```
SimpleService service  = new SimpleServiceImpl();
ServiceRegistration serviceRegistration = context.registerService(
SimpleService.class.getName(), service, null);
```

Le désenregistrement d'un service OSGi se réalise par l'intermédiaire de la méthode *unregister* de l'instance de *ServiceRegistration* renvoyée précédemment par la méthode *register*. Une bonne pratique consiste donc à le garder en variable d'instance (de l'activateur par exemple). Le code suivant illustre la mise en oeuvre d'un désenregistrement de service :

```
ServiceRegistration serviceRegistration = (...)
serviceRegistration.unregister();
```

Différentes méthodes sont également fournies afin d'avoir accès aux services afin de les utiliser. La récupération d'une instance de service se réalise en deux étapes. La

première consiste en la récupération d'une instance de l'interface ServiceReference pour le service par l'intermédiaire de la méthode getServiceReference ou getServiceReferences. L'utilisation de la méthode getService avec en paramètre l'instance précédente permet d'avoir accès à l'instance du service et de l'utiliser. [23]

Le code suivant décrit la manière d'avoir accès à une instance d'un service à partir de son nom et par l'intermédiaire du contexte OSGi :

```
// Récupération de la référence du service
ServiceReference reference = context.getServiceReference(
                            SimpleService.class.getName());

// Récupération de l'instance du service
SimpleService service = context.getService(reference);
  (...)
```

Nous pouvons également noter la présence d'une méthode ungetService au niveau du contexte afin de libérer l'instance du service référencée par l'instance de ServiceReference correspondante.

III.3 Conclusion

Dans ce chapitre, nous avons introduit la technologie OSGi et les différentes problématiques qu'elle vise à adresser. Nous avons également décrit les caractéristiques de la technologie et comment mettre en oeuvre des composants OSGi, composants désignés par le terme ***bundle*** dans la terminologie d'OSGi. Nous avons ainsi abordés les différentes couches structurant les conteneurs de ce type, à savoir les couches *Module, Cycle de vie et Service*.

Chapitre IV

Implementation

IV.1 Introduction

Dans ce chapitre, nous nous intéresserons à l'aspect de validation de notre approche proposée, qu'il s'agit d'expliquer l'environnement matériel et logiciel dans lequel notre système a été développé. On commence tout d'abord par citer les outils et langages de programmations utilisés, puis ensuite, nous aborderons le choix de la plate-forme. Aussi, nous monterons les techniques et les moyens avec les idées adoptées, et la manière avec laquelle nous avons implémenté chaque composant dans ce système. Enfin, Les résultats obtenus seront présentés à la fin de ce chapitre.

IV.2 Outils matériel

Le travail est fait sur un PC avec un système d'exploitation Windows 8.1, processeur I5 2.5 GHZ et 4 GO de RAM.

IV.3 Outils et environnements de programmation

IV.3.1 Eclipce

Pour notre choix de l'environnement de développement Java, nous avons opté pour l'utilisation de Eclipse (jee-mars-2-win32-x86_64), est un IDE (Integrated Development Environment) Open Source conçu pour concevoir, déployer et tester des applications développées sous différents langages (Java, PHP, Ruby, C/C++, etc.).

IV.3.2 JAVA

JAVA est un langage de programmation développé par Sun Microsystème (les premières versions datent de 1995). On note que nous avons utilisé la version : jdk-8u144-windows-x64.

IV.4 Création des Bundles

On commence tout d'abord par la création du bundle (plug-in) qui est l'entité centrale d'OSGi, c'est la première chose effectuée dans ce projet.Un bundle peut être généré par Eclipse via : File -> New-> Other... -> Plug-In Development -> (Entrée du menu : Plug-In Project), comme l'indique la figure suivante :

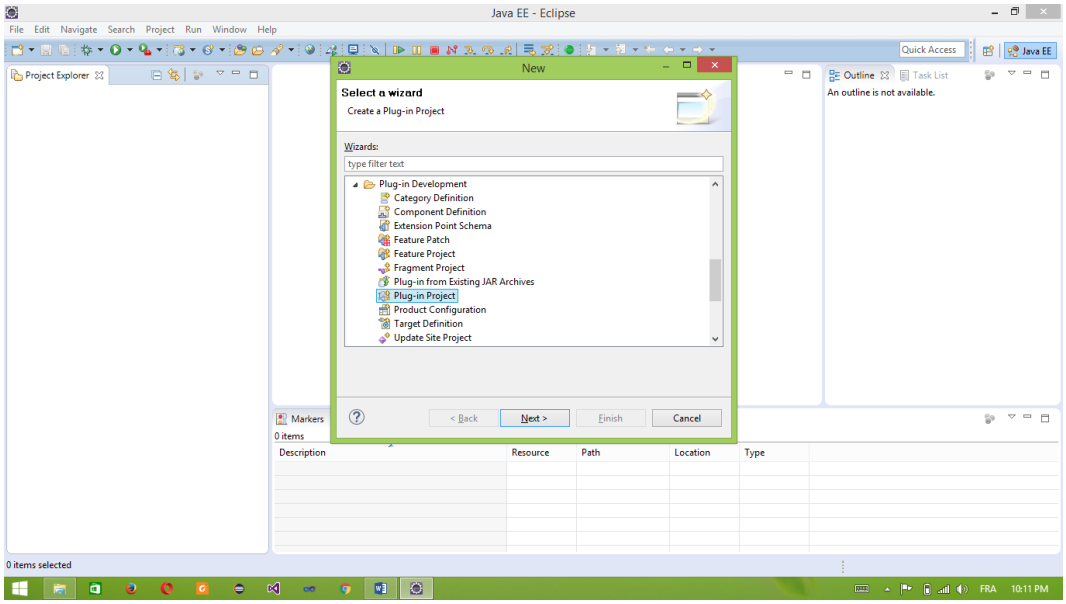

FIGURE 4.1: Génération du Bundle (plug-in) via Eclipse

Les bundles sont exécutés sur la plateforme *OSGi Equinox* qui a besoin des bibliothèques externes requises :

- org.eclipse.equinox.console
- org.apache.felix.gogo.command
- org.apache.felix.gogo.runtime
- org.apache.felix.gogo.shell
- Org.eclipse.osgi

IV.4.1 Création de deux services temperature et humidity

Ces deux bundles servent à fournir le degré de la température et le degré de l'humidité à un moment donné. On a creé dans le package : *com.java.temperatureservice* une interface *Gettemperature* et la classe *GettemperatureImpl* qui implémente cette interface.Au ssi, On a créé la classe *gettemperatureActivator* dans le package : *com.java.temperature.**.

Dans le deuxieme bundle : *com.java.humidity*, on a créé dans le package : *com.java.temperatureservice* et une Intrerface : *Gethumidity* et la Class *GethumidityImpl* qui implemente cette interface. Aussi, On a créé une Classe *gethumidityActivator* dans le package : *com.java.humidity*.

```
package com.java.temperature;

import org.osgi.framework.BundleActivator;

public class gettemperatureActivator implements BundleActivator {

    ServiceRegistration tempRegistration;

    public void start(BundleContext context) throws Exception {
        System.out.println("Temperature Bundle Working");

        Gettemperature te = new GettemperatureImpl();
        tempRegistration =context.registerService(Gettemperature.class.getName(), te, null);
    }

    public void stop(BundleContext context) throws Exception {
        tempRegistration.unregister();
    }

}
```

FIGURE 4.2: pseudo Code de Classe gettemperatureActivator

Le conteneur appellera la méthode *start()* de la classe Activator pour démarrer le bundle. Le bundle peut prendre cette opportunité pour effectuer l'initialisation des ressources, comme l'obtention d'une connexion à la base de données pour une utilisation future. La méthode *start()* prend un argument, l'objet *BundleContext*. Cet objet permet aux le bundles d'interagir avec le framework en fournissant l'accès aux informations liées au conteneur OSGi. Si une exception est lancée pour un bundle particulier, le conteneur marquera le blocage de ce bundle et ne le mettra pas en service.

A l'intérieur de la méthode *start()*, On utilise *BundleContext.registerService()* pour exporter le service de température *Gettemperature* ; qui est une Classe mise dans le registre de service de OSGi. Pour permettre a d'autres composants d'exploitera le service temperature, l'exportation de son package qui contient l'Interface *Gettemperature* est requis. Ça se fait par un double clic sur le fichier **MANIFEST.MF** dans le dossier META-INF -> Runtime -> Add et Sélectionner le package *com.java.temperatureservice*.

Le conteneur appellera la méthode *stop()* de la classe Activator pour indiquer l'arrêt du bundle. On peut utiliser cette opportunité pour effectuer des tâches de nettoyage telles que la libération de la connexion à la base de données.

La Classe Gettemperature : La Classe appelle la méthode *gettemperature()* prend un argument : time et retourne une valeur entier qui indique le degré de la température. Cette valeur se trouve dans un tableau de base de donné qui s'appelle : temperature ; le champ de ce tableau a deux cases : un pour le temps (time), et l'autre pour le degré de température de ce temps.

On a besoin de télécharger le driver : mysql-connector-java-5.1.42-bin pour connecter à la base de données mysql. On a creé un dossier "lib" et copier le driver qui est un fichier .jar dans le dossier "lib", un click droit sur le pluginProperties -> Java Build Path -> Add JARs et sélectionner le driver jar à l'intérieur de "lib".

```
public class GettemperatureImpl implements Gettemperature {

    public int gettemperature(String time){

        int te = 0;
        Connection MyConn = null;

        try{
            //1.
            MyConn = Connect(MyConn);
            //2.
            Statement myStmt = MyConn.createStatement();
            ResultSet myRes = myStmt.executeQuery("Select tmpr From temperature WHERE dt = '"+time+"'");

            if(myRes.next())      te=myRes.getInt(1);
            Disconnect(MyConn);
        }

        catch(Exception exc){
            System.out.println("Can not connect to database");
            exc.printStackTrace();
            System.exit(1);
        }
        return te;
    }
```

FIGURE 4.3: pseudo code de la classe GettemperatureImpl

IV.5 L'Exportation d'un bundle

L'Exportation d'un bundle permet de l'installer lors de l'exécution d'OSGi. On Sélectionne les deux bundles et on choisi : File -> Export-> Plug-in Development -> Deployable plug-ins and fragment. Choisir un répertoire et clicker finish, les deux fichiers jars sont créés comme le montre la figure ci-dessous :

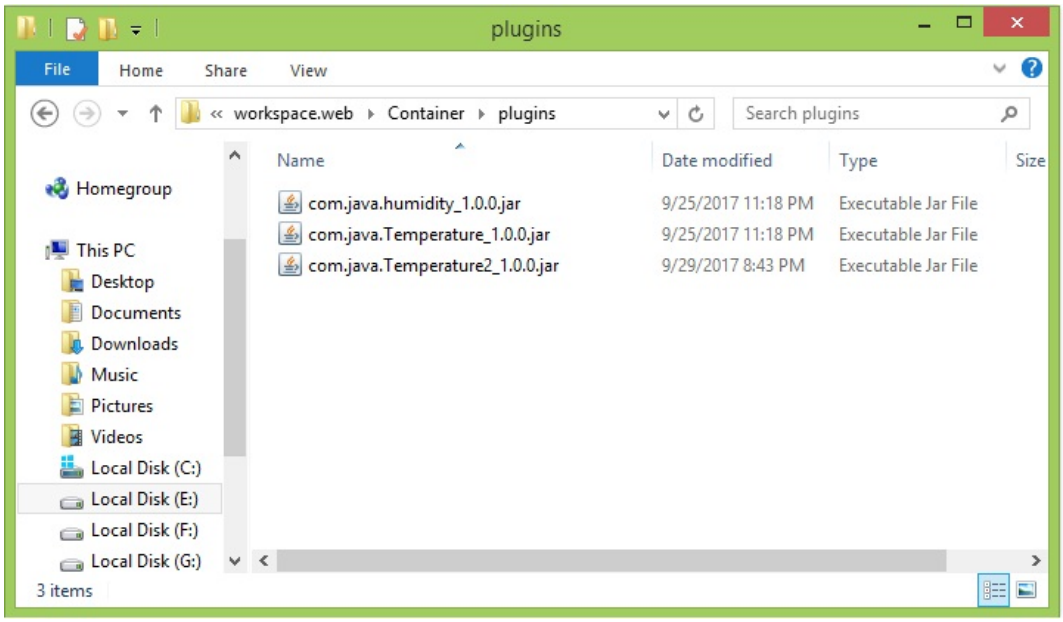

FIGURE 4.4: Les bundles installés dans le conteneur

IV.6 Le Projet Primaire (Serveur)

Le serveur est une classe de nom Container.

```
@Path("/OSGi"):
public class Container {
}\\
```

- @Path("/OSGi") Définit le chemin d'accès qui est égal à : l'URL de base + /OSGi.

Méthodes du Serveur :
La Classe Container ayant la liste de méthodes suivantes :

1. **get_Temperature :**
   @GET
   @Path("/temperature/{time}")
   @Produces(MediaType.TEXT_PLAIN)
   public String get_Temperature(@PathParam("time")
   String time) throws Exception {....}

   - La méthode @GET s'inscrit comme une ressource get via l'annotation @GET.
   - @Path(...) est disponible sous l'URL de base +/OSGi+ /temperature/temps.
   - "time" est un paramètre de format hh :mm.
   - @Produces(MediaType.TEXT_PLAIN) définit qu'elle délivre le texte.

2. **get_humidity :**
   @GET
   @Path("/humidity/{time}")
   @Produces(MediaType.TEXT_PLAIN)
   public String get_Humidity(@PathParam("time")
   String time) throws Exception {....}

3. **adjust_time :**
   public String adjust_time (String) {....}

   - ajuste le temps sous le format : hh :00

4. **Trait_Demand :**
   public int Trait_Demand(Demand , String , String , String) throws Exception {....}

   - Elle est appelée par l'un de deux méthodes : get_Temperature et get_humidity.
   - Tester si la demande (requête de client) est traitée avant moindre d'une heure.
   - Enregistrer une nouvelle demande.
   - Faire l'appelle de la méthode run_service.

- Enregistrer le résultat de la demande.
- Retourner le résultat de la demande qui est le degré de la température ou de l'humidité.

5. **run_service :**
   public int run_service(String , String) throws Exception{....}

   - Faire l'appel de la méthode install.
   - Faire un autre appel de l'un de deux méthodes : Call_Service_Temperature et Call_Service_humidity.
   - Retourner une valeur entier qui représente le degré de la température ou de l'humidité.

6. **install :**
   public BundleContext install(String) throws Exception {....}

   - Sert à installer le fichier jar soit : com.java.temperature ou com.java.humidity.
   - Elle lance le service de température ou de l'humidité dans le registre de service de la plateforme OSGi.
   - Retourne une valeur de type BundleContext qui à utiliser par l'un de deux méthodes : Call_Service_Temperature oubien Call_Service_humidity.

7. **Call_Service_Temperature :**
   public int Call_Service_Temperture(BundleContext , String) throws Exception {....}

   - Sert à consommer le service de température qui est dans le registre de service de : OSGi.
   - Retourne une valeur de type entier qui représente le degré de la température.

8. **Call_Service_humidity :**
   public int Call_Service_humidity(BundleContext , String) throws Exception {....}

   - Sert à consommer le service de l'humidité qui est dans le registre de service de OSGi.
   - Retourne une valeur de type entier qui représente le degré de l'humidité.

IV.7 Séquence de l'exécution

1. Le Client : lance une requête http via un serveur Apache Tomcat, cette requette est comme suit :
   http ://localhost :8080/Container/OSGi/temperature/hh :mm
   (hh : 00->23, mm : 00->59)

   ou bien

   http ://localhost :8080/Container/OSGi/humidity/hh :mm
   (hh : 00->23, mm : 00->59)

2. Si la requête contient : /temperature/hh :mm, la méthode get_temperature se lance

   —> Adjust_time —> Trait_Demand —> Run_service —>
   install —> Call_Service_Temperature.

3. Si la requête contient : /humidity/hh :mm, la méthode get_humidity se lance

   —> Adjust_time —> Trait_Demand —> Run_service
   —> install —> Call_Service_humidity.

IV.8 Architecture Globale

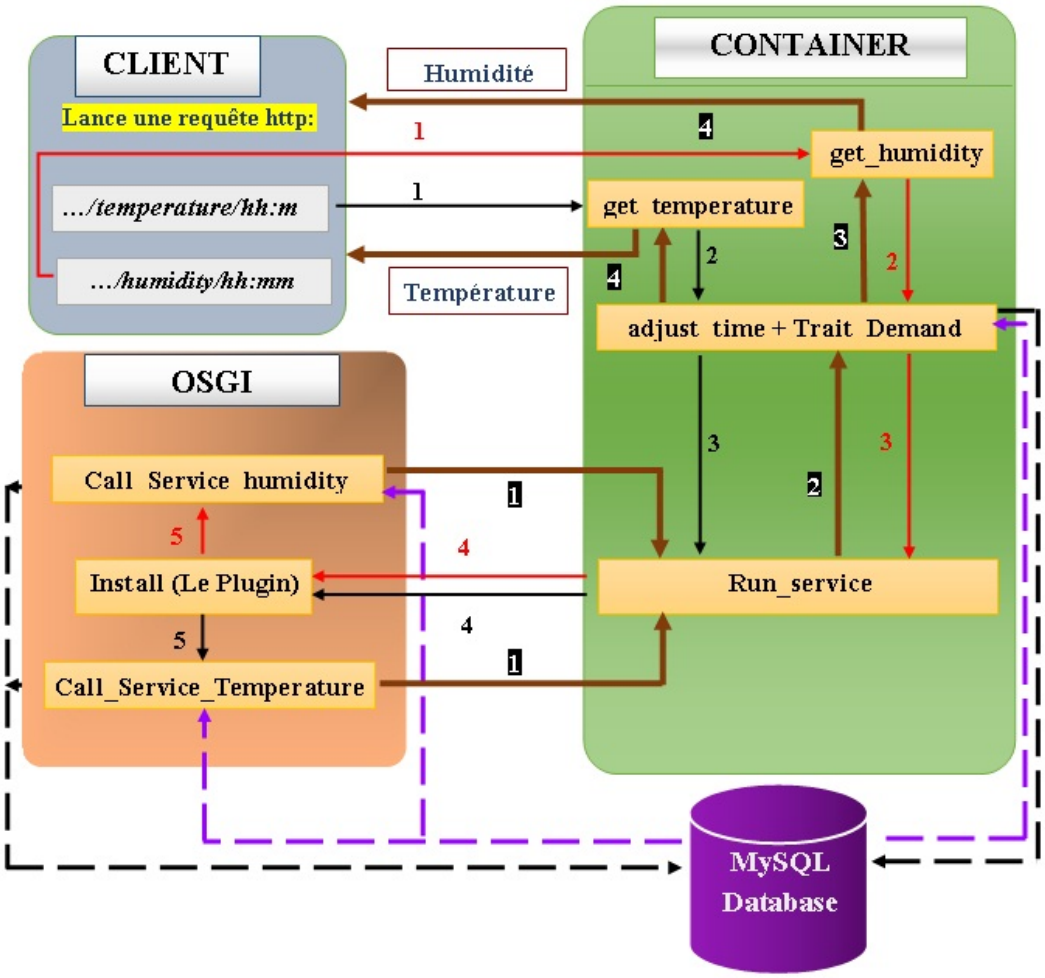

FIGURE 4.5: Architecture Glabale du Projet

IV.9 Les résultats obtenus

Les résultats sont illustrés par les expérimentations suivantes montrées dans les figures ci-dessous :

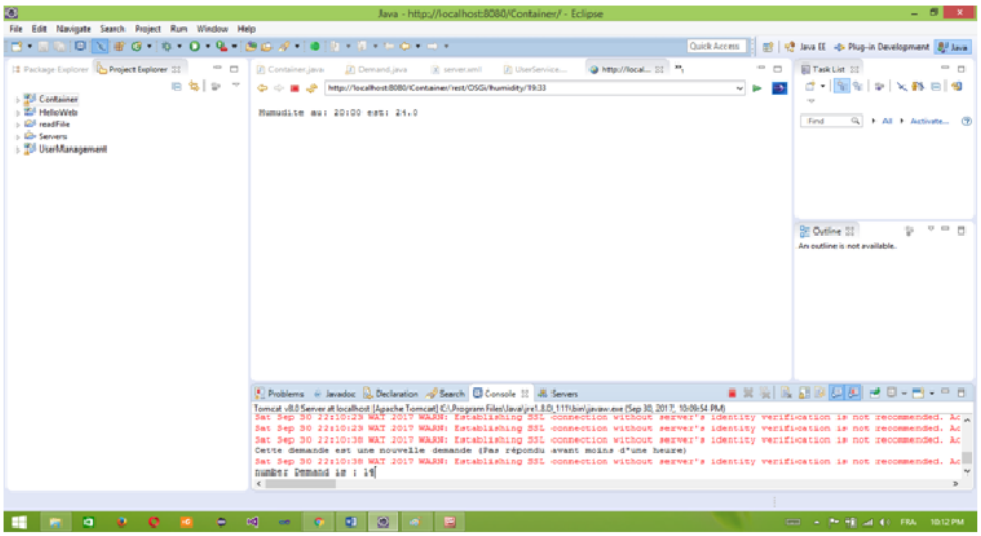

FIGURE 4.6: Résultat de la demande du service Humidité

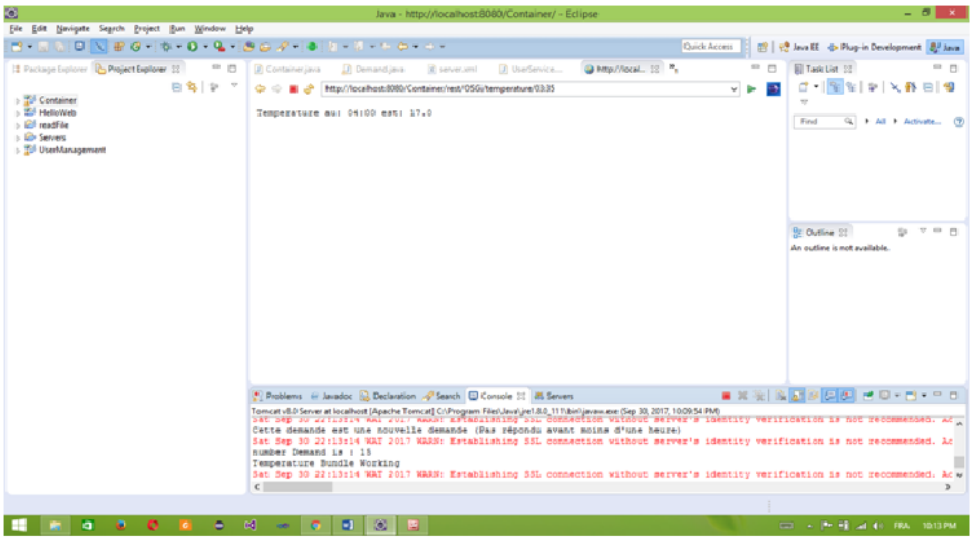

FIGURE 4.7: Résultat de la demande du service temperature

Conclusion Générale

Dans ce dernier chapitre, nous avons élaboré une solution au problème de la communication avec las objets connectés, où cette solution est basée sur le modèle présenté dans le chapitre précédent. On a également présenté, les éléments du système et leurs fonctionnements, ainsi, la fonctionnalité de notre middleware orienté microservice et les mises en oeuvres des bundles.

Par ailleurs, le développement de notre système nous a permis de faire une étude réelle pour améliorer la gestion des services IoT par des plugins OSGi, nous réalisons l'étape d'évaluation des ressources de système par des APIs et les envoyer au serveur.

En effet, dans notre application nous avons constaté une efficacité remarquable de plugin dans l'évaluation des ressources et la communication entre les entités du système.

Annexes

Configuration du serveur

Développement web Java avec Eclipse WTP

1. Installation et configuration d'Eclipse WTP :

Clic : Help —> Eclipse Marketplace, comme indiqué sur la figure ci-dessous :

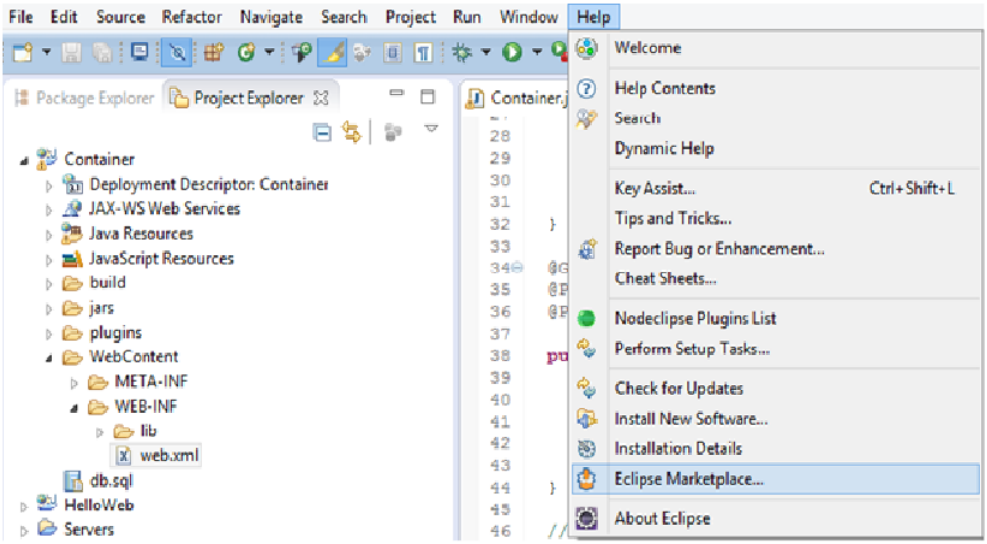

FIGURE 0.8: Configuration du serveur

Et installer les fonctionnalités suivantes :

- Eclipse Java EE Developer Tools
- Eclipse Java Web Developer Tools
- Eclipse Web Developer Tools
- JST Server Adapters

- JST Server Adapters Extensions

2. Création d'un Dynamic Web Project :

On suit : File —> New —> Other... —> Web —> (Entrée du menu : Dynamic Web Project), et on le nomme : **Container**

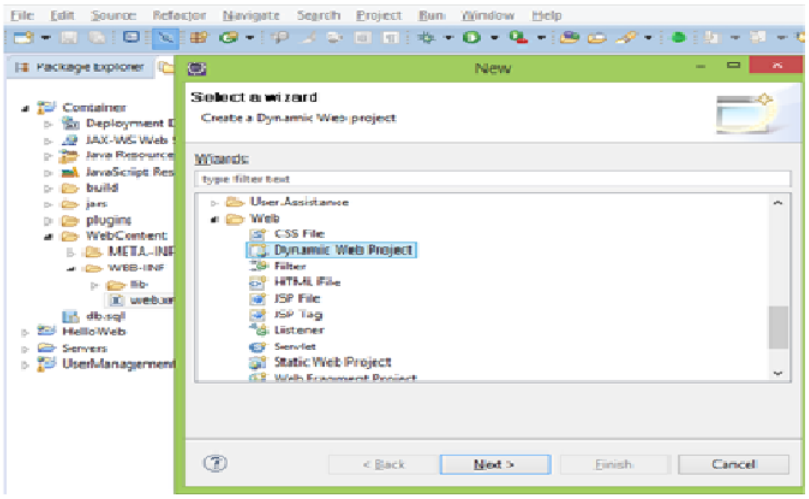

FIGURE 0.9: Création d'un Dynamic Web Project

3.Configuration manuelle des bibliothèques Jersey dans un projet Eclipse :

Téléchargez la distribution de Jersey sous forme de fichier zip à partir du site de téléchargement de Jersey. Le zip contient la JAR de Jersey et ses dépendances principales. Copiez tous les JAR de votre téléchargement Jersey dans le dossier WebContent / WEB-INF / lib.

Dans le dossier jars : on ajoute les fichiers jars suivantes : à partir du fichier de plugins qui est dans le répertoire d'installation Eclipse.

- org.eclipse.equinox.console
- org.apache.felix.gogo.command
- org.apache.felix.gogo.runtime
- org.apache.felix.gogo.shell
- Org.eclipse.osgi

Dans le dossier plugins : on ajoute les 2 fichiers jars exportés en antérieur com.java.Temperature_1.0.0.jar, com.java.humidity_1.0.0.jar et le pilote base de donnée mysql-connector-java-5.1.42.

Un click droit sur le plugin —> Properties —> Java Build Path —> Add JARs et sélectionne les 8 fichiers jar —> Apply —> Ok.

Apache Tomcat

Lancez l'Eclipse, qui contient un module WTP (Web Tool PlatForm) qui va nous servir pour intégrer tomcat dans Eclipse Clic : Menu New —> Other

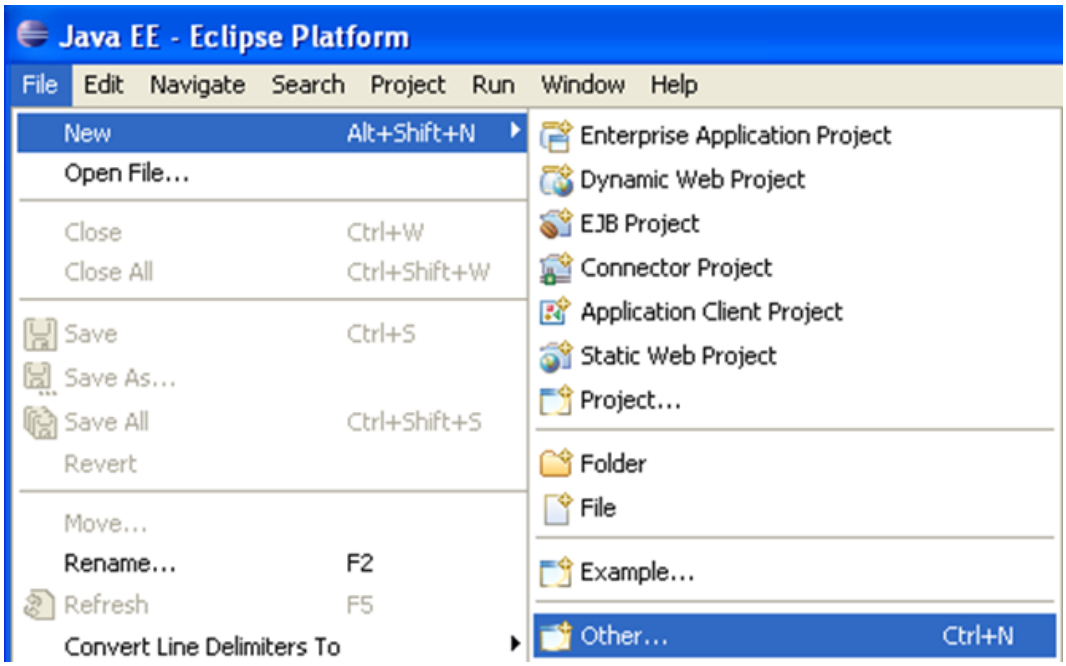

FIGURE 0.10: Configuration du serveur Apache Tomcat étape 1

L'écran suivant apparaît :

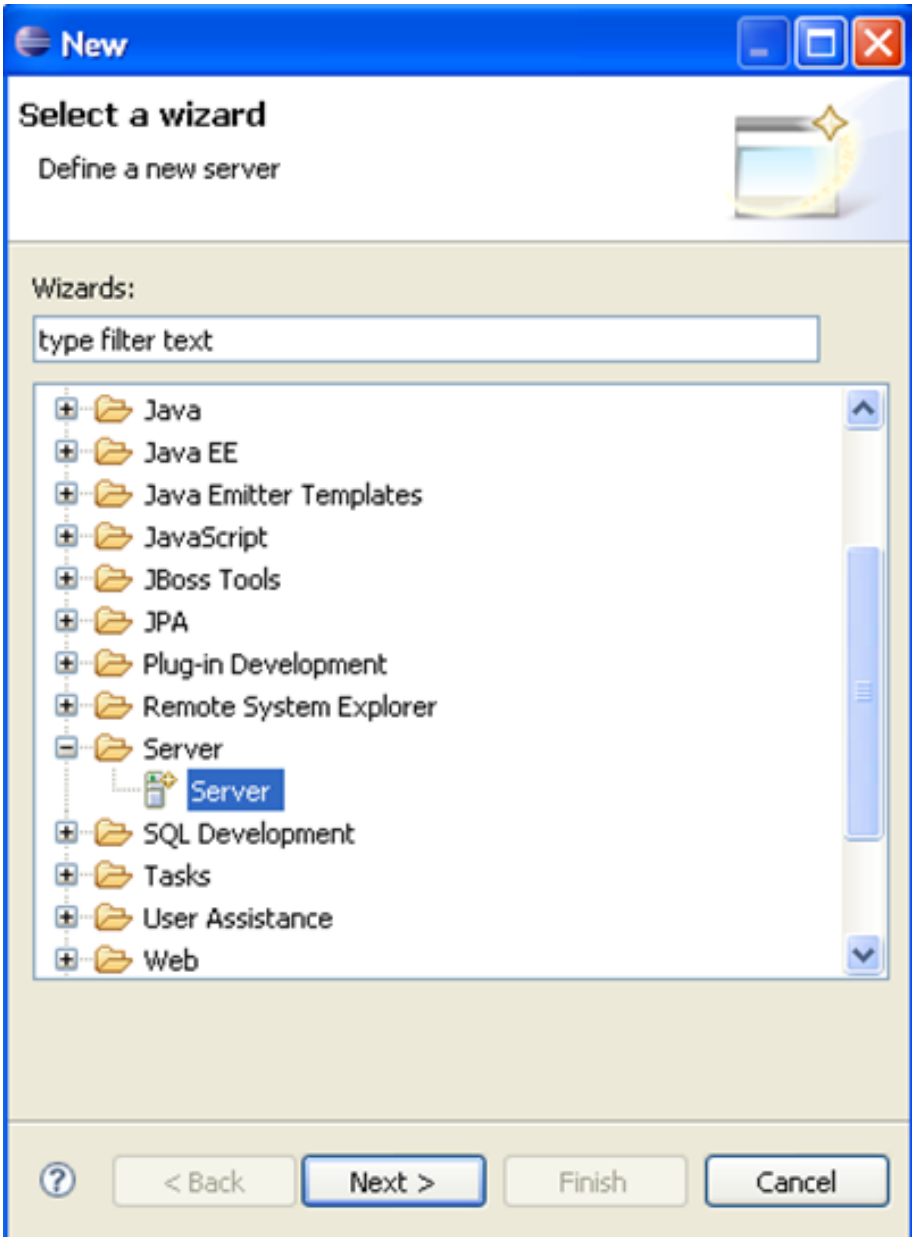

FIGURE 0.11: Configuration du serveur Apache Tomcat étape 2

Choisir Server/Server puis bouton Next

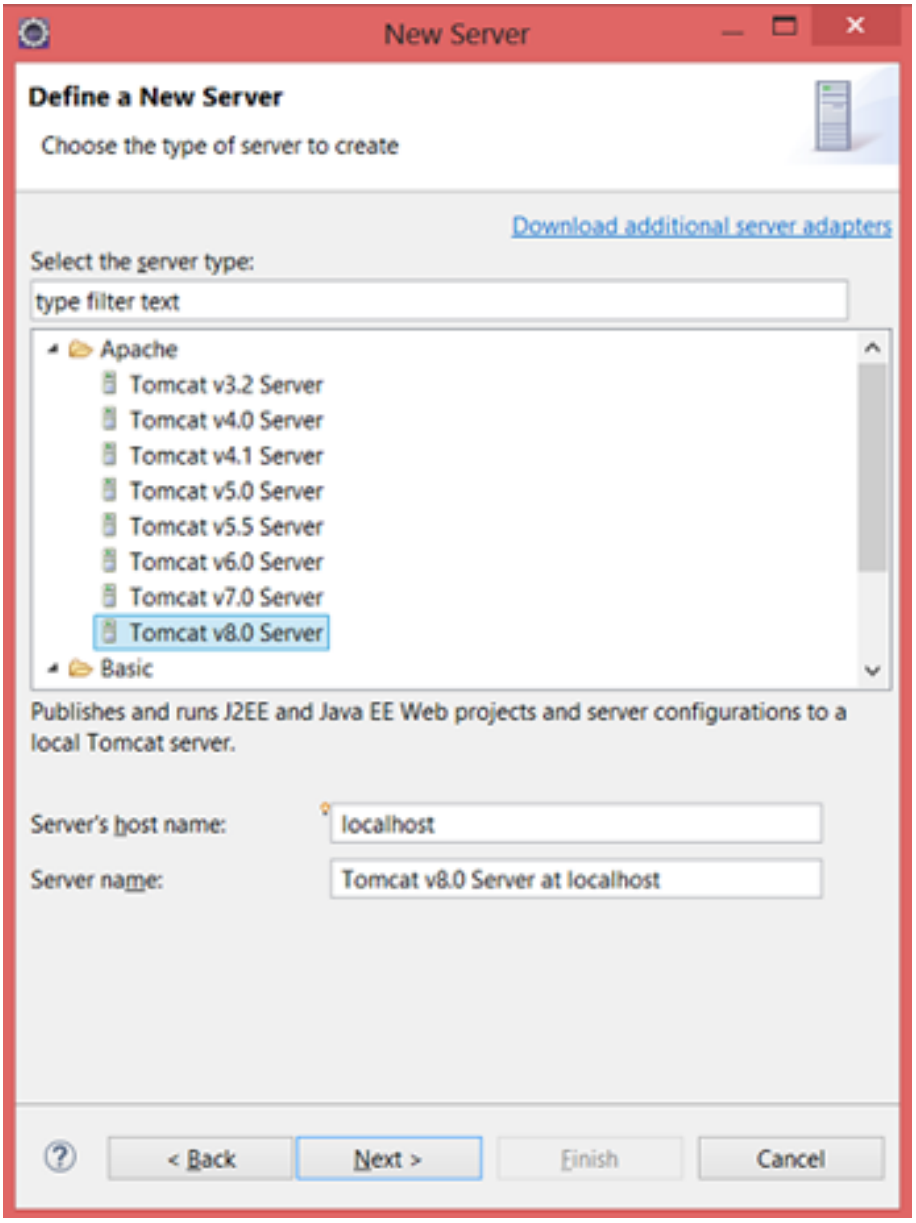

FIGURE 0.12: Configuration du serveur Apache Tomcat étape 3

Choisissez Apache/Tomcat v8.O Server puis bouton Next.
L'écran suivant apparaît

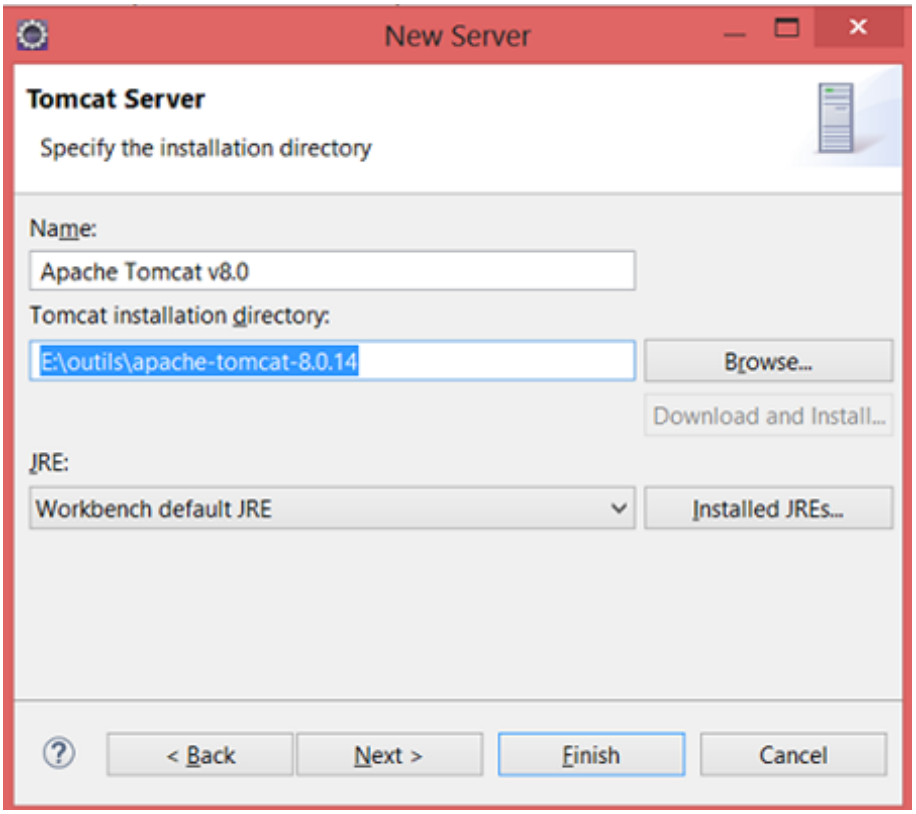

FIGURE 0.13: Configuration du serveur Apache Tomcat étape 4

Cliquez sur 'Browse' puis naviguez jusqu'au répertoire d'installation de tomcat. Cliquez sur le bouton Next

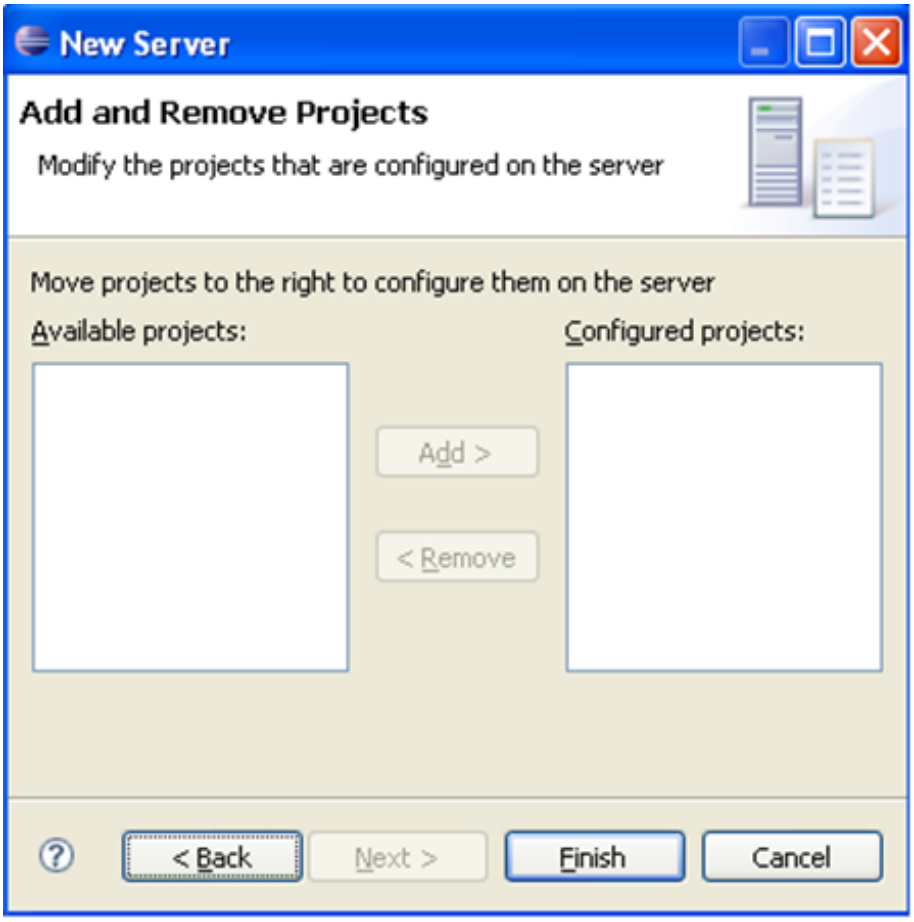

FIGURE 0.14: Configuration du serveur Apache Tomcat étape 5

Comme il n'y a aucun projet web à déployer, Cliquez sur le bouton **Finish** Tomcat génère les informations suivantes dans la vue Explorer.

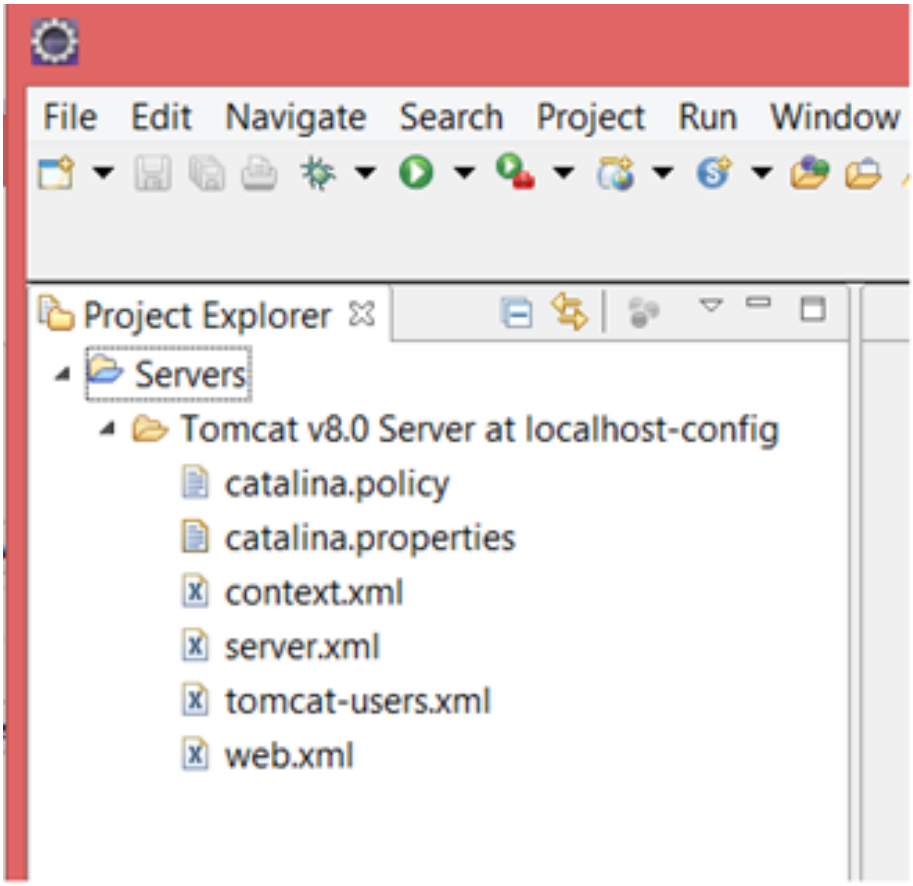

FIGURE 0.15: Configuration du serveur Apache Tomcat étape 6

REMARQUE : Eclipse a fait une copie des fichiers de configuration du tomcat installé dans le workspace. Remarquez qu'une nouvelle vue 'Servers' est apparue, nous informant que Tomcat est bien intégré à Eclipse, avec un statut Arrêté (Stopped).

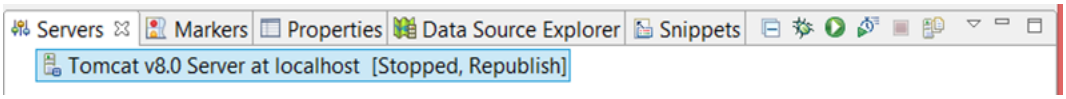

FIGURE 0.16: Configuration du serveur Apache Tomcat étape 7

Démarrage tomcat dans eclipse

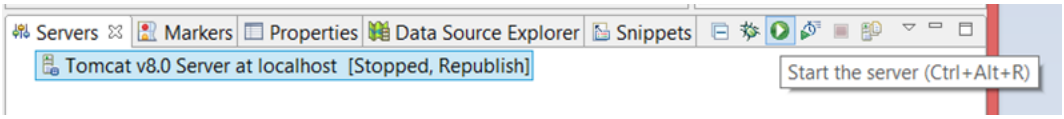

FIGURE 0.17: Configuration du serveur Apache Tomcat étape 8

Eclipse affiche dans la vue console les infos de logs Tomcat habituellement présent dans la console Invite de commande.

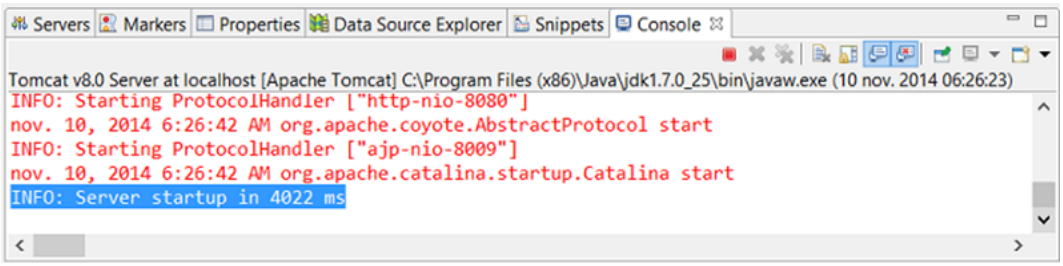

FIGURE 0.18: Configuration du serveur Apache Tomcat étape 9

Remarquez que dans la nouvelle vue 'Servers', Eclipse affiche un statut démarré (Started).

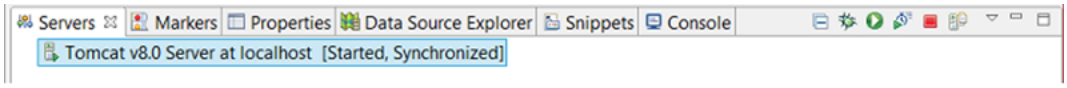

FIGURE 0.19: Configuration du serveur Apache Tomcat étape 10

Arrêt tomcat dans eclipse : Bouton rouge.

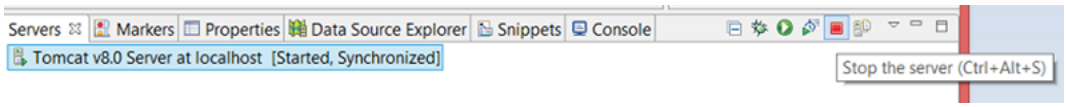

FIGURE 0.20: Configuration du serveur Apache Tomcat étape 11

Servlet Jersey

On a besoin d'enregistrer Jersey comme dispatcher de servlet pour les demandes REST.
Dans le projet Container ouvrir web.xml qui est dans le Fichier WebContent / WEB-INF :

```
4   <display-name>Container</display-name>
5   <servlet>
6     <servlet-name>Jersey REST Service</servlet-name>
7     <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
8      <!-- Register resources and providers under com.vogella.jersey.first package. -->
9     <init-param>
10        <param-name>jersey.config.server.provider.packages</param-name>
11        <param-value>Web.Container</param-value>
12    </init-param>
13    <load-on-startup>1</load-on-startup>
14  </servlet>
15  <servlet-mapping>
16  <servlet-name>Jersey REST Service</servlet-name>
17    <url-pattern>/rest/*</url-pattern>
18  </servlet-mapping>
19 </web-app>
```

FIGURE 0.21: Servlet Jersey

- **Le nom d Container :** le premier mot qui apparaitre après http ://localhost :8080/Container.
- **Jersey REST Service :** C'est le nom de Servlet.
- **Web.Container :** C'est le package qui contient des ressources.

Références bibliographiques

[1] Greenough John and Camhi Jonathan. The internet of things : Examining the iot affecting the world, Novembre 2015. www.scribd.com.

[2] Lee In and Lee Kyoochun. The internet of things (iot) : Applications, investments, and challenges for enterprises., Mars 2015. Business Horizons, 58(4), 431-440 from http://dx.doi.org/10.1016/j.bushor.2015.03.008.

[3] Kranenburg Van. A critique of ambient technology and the all-seeing network of rfid. www.networkcultures.org, 2008.

[4] Louis Coetzee and John Ekstee. The internet of things-promise for the future ? an introduction. Paper presented at the IST-Africa Conference Proceedings, 2011.

[5] Atzori Luigi, Iera Antonio, and Morabito Giacomo. The internet of things : A survey. computer networks. http://dx.doi.org/10.1016/j.comnet.2010.05.010, 2010.

[6] G Kortuem, F Kawsar, V Sundramoorthy, and D Fitton. Smart objects as building blocks fort he internet of things, 2010. IEEE Internet Computing, 14, 44-51.

[7] EuraRFID CITC. Analyse et perspective de l'internet des objets, Octobre 2013. www.citc-eurarfid.com.

[8] Xu Li Da, He Wu, and Shancang Li. Internet of things in industries : A survey, Novembre 2014. https://www.researchgate.net/publication/270742269.

[9] J Liu. Facebook and google vie to bring internet connectivity to all. fodder for young minds. http://www.dogonews.com/2015/8/24/facebook-and-google-vie-to-bring-internet-connectivity-to-all, 2015.

[10] Oliver Wyman. Internet des objets : Les business models remis en cause. http://www.oliverwyman.fr/nos-publications.html, 2015.

[11] US edition. Internet of things privacy index, 2014. www.truste.com.

[12] Inc Gartner. Gartner says the internet of things will transform the data center, 2014. http://www.gartner.com/newsroom/.

[13] Bandyopadhyay and Sen. Internet of things : Applications and challenges in technology and standardization, Mai 2011. https://www.researchgate.net/publication/.

[14] J Holdowsky, M Mahto, M Raynor, and M Cotteleer. Inside the internet of things, 2015. https://dupress.deloitte.com/.

[15] accenture. The internet of things : The future of consumer adoption, 2014. https://www.accenture.com/.

[16] Oliver Wyman. Internet des objets. http://www.oliverwyman.fr/nos-publications.html, 2016.

[17] Les microservices. https://fr.wikipedia.org/wiki/Microservices.

[18] Clapaud Alain. Le concept des microservices, Navombre 2015. www.journaldunet.com.

[19] Janakiram MSV. Microservices : Se preparer a la nouvelle generation d'applications cloud, Juillet 2015. www.lemagit.fr.

[20] Jboss REDHAT. Une plateforme de modernisation des applications java, Juin 2016. www.fr.redhat.com.

[21] Programmation orientée composant. https://fr.wikipedia.org/wiki/Programmation_orientée_composant.

[22] Alliance OSGI. Osgi architecture. https://www.osgi.org/developer/architecture/.

[23] Eclipse. Eclipse articles. http://wiki.eclipse.org/Eclipse_Articles,_Tutorials,_Demos,_Books,_and_More.