République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Blida 1
Faculté des Sciences
Département de l'informatique

Mémoire de fin d'études en vue de l'obtention du diplôme de
MASTER EN INFORMATIQUE
Option : Ingénierie des Logiciels

Thème

Courtage dans le Cloud Computing pour l'interopérabilité sémantique entre Clouds de type Software as a Service

Réalisé par:
- RIF Fella Rym
- SAIDI Soumia

Devant le jury :
- M. BENYAHIA Mohamed (Président)
- Mme. AROUSSI Sana (Examinatrice)
- Mme. MANCER Yasmine (Promotrice)

Promotion : 2019/2020

Remerciement

Nous voulons d'abord remercier Allah, qui nous a donné la force et la capacité pour réaliser ce travail.

Nous remercions également notre promotrice, Mme Mancer, qui nous a guidé et aidé tout au long de notre travail.

Nous remercions sincèrement nos chers parents, nos familles et tous nos amis.

Enfin, nous tenons à remercier les membres du jury qui ont accepté d'évaluer notre travail.

Résumé

Malgré son apparition relativement récente, le Cloud Computing a connu une monté rapide en popularité. Le rapport qualité/prix des services de Cloud Computing a séduit beaucoups de consommateurs qui désirent avoir les performances de certaines ressources sans devoir en procurer de manière définitive. Cependant, les défis auxquels le Cloud Computing doit faire face sont également en constante augmentation et l'interopérabilité en est un des plus complexes.

L'interopérabilité est un problème épineux qui empêche une adoption plus large des services de Cloud Computing à cause du verrouillage des fournisseurs d'un côté et du manque de standardisation de l'autre, empêchant ainsi non seulement la migration vers les solutions de Cloud Computing mais également la collaboration entre les différents services. Ce défis est particulièrement difficile à surmonter dans les services de Cloud Computing de type Software as a Service (SaaS).

Notre but est de proposer une solution à ce défi à base de broker, qui est un système de courtage, qui sert d'intermédiaire et de gérant entre les services de Cloud Computing de type SaaS et assure la collaboration entre eux. Notre solution est destinée aux fournisseurs qui souhaitent mettre à disposition leur services, mais aussi aux clients qui veulent bénéficier de ces services.

Dans notre solution, le courtier s'occupe du traitement des opérations de description, de découverte, de sélection et de composition des services. Son travail repose sur la description sémantique des aspects essentiels d'un SaaS publié par un fournisseur et le choix des meilleurs services en fonction de leur qualité de service (QoS) pour répondre aux sollicitations des clients.

Mots Clés: Cloud Computing, interopérabilité, broker, Software as a Service (SaaS), qualité de service (QoS).

Abstract

Despite its relatively recent appearance, Cloud Computing has experienced a rapid rise in popularity. The price-quality ratio of Cloud Computing services has attracted many consumers who want the performance of certain resources without having to provide them permanently. However, the challenges facing Cloud Computing are also constantly increasing and interoperability is one of the most complex.

Interoperability is a thorny issue that prevents wider adoption of Cloud Computing services because of vendor lock-in on the one hand and lack of standardization on the other, preventing not only migration to Cloud Computing solutions but also collaboration between different services. This challenge is particularly difficult to overcome in Software as a Service (SaaS) Cloud Computing services.

Our goal is to offer a solution to this challenge based on a brokerage system, which acts as an intermediary and manager between Cloud Computing services of the SaaS type and ensures collaboration between them. Our solution is intended for suppliers who wish to make their services available, but also for customers who want to benefit from these services.

In our solution, the broker handles the processing of the description, discovery, selection and composition of services. Its work is based on the semantic description of the essential aspects of a SaaS published by a provider and the selection of the best services according to their quality of service (QoS) to meet customer requests.

Keywords: Cloud Computing, interoperability, broker, Software as a Service (SaaS), quality of service (QoS).

ملخص

على الرغم من ظهورها مؤخرًا نسبيًا، فقد شهدت الحوسبة السحابية ارتفاعًا سريعًا في شعبيتها. اجتذب مفهوم تناسبية النوعية مع السعر لخدمات الحوسبة السحابية العديد من المستهلكين الذين يرغبون في اكتساب أداء موارد معينة دون الحاجة إلى اقتناءها بشكل دائم. ومع ذلك، فإن التحديات التي تواجه الحوسبة السحابية تتزايد باستمرار وتعد إمكانية التشغيل البيني واحدة من أكثر التحديات تعقيدًا.

تعد إمكانية التشغيل البيني مشكلة شائكة تمنع التبني الأوسع لخدمات الحوسبة السحابية بسبب إغلاق الموردين من ناحية ونقص التوحيد القياسي من ناحية أخرى، وبالتالي هذا المشكل لا يمنع فقط الانتقال إلى حلول الحوسبة السحابية، ولكن يحول أيضًا دون التعاون بين الخدمات المختلفة. من الصعب التغلب على هذا التحدي في خدمات الحوسبة السحابية.

هدفنا هو توفير حل لهذا التحدي بحيث يكون حلنا قائم على الوسيط، و يعمل هذا الوسيط كمدير بين خدمات الحوسبة السحابية من نوع خدمات البرمجيات أوSaaS ويضمن التعاون فيما بينها. حلنا مخصص للموردين الذين يرغبون في إتاحة خدماتهم، ولكن أيضًا للعملاء الذين يرغبون في الاستفادة من هذه الخدمات.

في حلنا، يعتني الوسيط بمعالجة الوصف والاكتشاف والاختيار وعمليات التركيب. يعتمد عمله على الوصف الدلالي للجوانب الأساسية للخدمات من نوع SaaS التي ينشرها المورد ويقوم باختيار أفضل الخدمات وفقًا لجودة الخدمة (QoS) لتلبية متطلبات العملاء.

الكلمات الرئيسية: الحوسبة السحابية، إمكانية التشغيل البيني، الوسيط، خدمات البرمجيات (SaaS)، جودة الخدمة (QoS).

Sommaire

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

API : Application Programming Interface

CaaS : Containers as a Service

DAML-S : Semantic Markup for Web services

DaaS : Desktop as a Service

HaaS : Hardware as a Service

IBM : International Business Machines Corporation

IaaS : Infrastructure as a Service

IHM : Interface Homme Machine

JSON : JavaScript Object Notation

NaaS : Network as a Service

NIST : National Institute of Standards and Technology

OWL : Web Ontology Language

OWL-S : Semantic Markup for Web Services

PaaS : Platform as a Service

RDF : Resource Description Framework

SaaS : Software as a Service

SLA : Service Level Agreement

SQL : Structured Query Language

SPARQL : Protocol and RDF Query Language

STaaS : Storage as a Service

USDL : Unified Service Description Language

VM : Virtual Machine

VPNaaS : VPN Virtual Private Network as a Service

WSDL : Web Services Description Language

XML : Extensible Markup Language

XSD : XML Schema Definition

Introduction générale

Malgré le fort engouement que suscite le Cloud Computing, son adoption et sa vulgarisation se heurtent à plusieurs obstacles. Le manque de standardisation a fait émerger beaucoup de problèmes relatifs au nombre grandissant de fournisseurs de services Cloud d'un côté, et d'un autre côté, à l'hétérogénéité des technologies et plateformes utilisées. Delà, un réel besoin de collaboration, et donc d'interaction, est ressenti par les différents intervenants dans le Cloud.

Une des solutions pour faire face à ces problèmes est d'aller vers des Clouds ouverts. Dans ce type de Cloud, l'utilisateur a la liberté de choisir entre les différents fournisseurs et les services Cloud, selon son besoin qui peut changer au cours du temps. Cependant, pour parvenir à l'ouverture du Cloud, il faut d'abord régler les problèmes d'interopérabilité.

1.Problématique

La plupart des solutions et des modèles existants mettent l'accent sur l'infrastructure en tant que service et plate-forme en tant que service. La recherche de l'interopérabilité pour les services Cloud de type logiciel en tant que service est encore très immature. En effet, les solutions et les modèles disponibles conçus pour l'interopérabilité des services Cloud de type logiciels en tant que service ne couvrent pas toutes les exigences de l'interopérabilité.

La problématique de ce travail consiste à définir un modèle de broker délivré comme un service Cloud pour l'interopérabilité au niveau sémantique dans des environnements Cloud de type SaaS.

2.Objectifs

Les objectifs visés à travers ce travail sont les suivants:

1. Etude de l'interopérabilité dans le Cloud Computing et dans les services SaaS.
2. Etude comparative des solutions existantes pour l'interopérabilité au niveau SaaS.
3. Définition des exigences d'interopérabilité dans les environnements Cloud Computing de type SaaS.
4. Définir un modèle de broker dans des environnements Cloud de type SaaS. Le modèle proposé doit garantir les objectifs suivants:

a. Définition des scénarios d'interopérabilité.

b. Description des règles d'interaction.

c. Description des interfaces de communication des SaaS.

d. Définition d'un format SLA interopérable.

5. Validation de la solution proposée par une expérimentation à travers une application Java EE.

3.Organisation du mémoire

Le travail décrit dans ce mémoire est organisé en cinq chapitres.

Le premier chapitre présente les notions de base du Cloud Computing telles que la définition du Cloud Computing, les modèles de livraison et de déploiement des services de Cloud Computing, les acteurs, les avantages, les inconvénients et les défis rencontrés dans ce domaine, mais également les concepts liés au Cloud computing tels que la virtualisation, les accords de niveau de service et l'inter-Cloud.

Le deuxième chapitre présente l'interopérabilité dans le Cloud Computing et les concepts qui y sont liés.

Le troisième chapitre contient les résultats de notre travail de recherche sur la problématique de l'interopérabilité. Nous présentons d'abord une étude comparative des solutions existantes pour l'interopérabilité des services de Cloud Computing. Puis, nous consacrons le reste du chapitre à une étude comparative des solutions existantes pour l'interopérabilité au niveau des SaaS.

Le quatrième chapitre est dédié à la conception de notre solution, nous commençons par une analyse des besoins à travers laquelle nous traçons les règles d'interactions des acteurs avec le système, puis nous présentons la solution proposée en utilisant des exemples illustratifs pour montrer la logique de notre application.

Dans le cinquième chapitre, nous présentons la mise en œuvre de notre solution en détaillant les interfaces de communication des utilisateurs avec notre application JavaEE.

Enfin, nous terminerons par une conclusion générale et quelques perspectives pour les travaux futurs.

Chapitre I

Introduction au Cloud Computing

I.1 Introduction

Traditionnellement, lorsqu'on avait besoin d'une capacité de stockage ou de calcul supérieure, on n'avait pas d'autres choix que de changer de matériel en achetant de nouvelles machines qui répondraient aux besoins. De même, lorsqu'on avait besoin de certaines applications logicielles, on achetait leurs licences et on les installait. Bien évidemment, cela était couteux en termes de temps et d'argent. Aujourd'hui, nous avons la chance d'avoir des alternatives plus économiques et plus rapidement accessible sous forme de services Cloud.

Dans ce chapitre, nous allons introduire ce type de services et les notions de base qui y sont reliées.

Nous commencerons par donner un petit aperçu sur l'histoire du Cloud Computing, suivie de sa définition. Ensuite, nous passerons aux caractéristiques essentielles des services Cloud pour donner une idée sur les critères que les pionniers du domaine utilisent pour classer un service dans cette catégorie. Puis, nous présenterons les acteurs qui entrent en jeu dans le Cloud Computing, suivis des différents modèles de livraison de services Cloud ainsi que leurs modèles de déploiement.

Nous allons également consacrer une partie de ce chapitre au contrat SLA (Accord de Niveau de Service) pour donner quelques détails sur son rôle dans la mise en disposition des services Cloud. Cela sera enchainé par la virtualisation qui est un facteur très important du Cloud Computing bien qu'elle eût apparu longtemps avant ce dernier et pour contraster cela, nous allons invoquer des concepts dérivés du Cloud Computing qui sont apparu après l'évolution de ce domaine pour l'étendre et lui permettre de prendre plus d'ampleur, il s'agit de l'interCloud, du MultiCloud et des fédérations de Cloud.

Nous finirons par donner quelques avantages et inconvénients du Cloud Computing ainsi que les défis de recherches auxquels ce domaine est confronté.

I.2 Histoirique

En 1955 John McCarthy [1] a introduit le concept de partage d'une seule machine via plusieurs stations individuelles grâce à sa théorie du "Time sharing", ou "Partage de temps de calcul".

En 1960, J.C.R. Licklider [2] eut l'idée d'un système d'ordinateurs interconnectés. Il imagina aussi un monde où tout est connecté et où on peut accéder à des données et des programmes sans avoir à se soucier de leur emplacement de stockage. Ceci fut le début du Cloud Computing et du réseau internet.

En 1969, Bob Taylor [3] et Larry Roberts [4] se basèrent sur les idées de Licklider et développèrent ARPANET, un système considéré aujourd'hui comme le prédécesseur du réseau Internet.

En 1972, IBM [5], le géant informatique, introduit le VM Operating système, Système d'exploitation qui ouvrit les portes vers le monde des machines virtuelles.

Le concept du Cloud Computing tel qu'on le connaît aujourd'hui fut initié par Amazon [6] en 2006, lorsque ce dernier lança AWS [7] ou Amazon Web Services, service en ligne qui fournit aux utilisateurs des infrastructures informatiques, des plateformes de developement et des applications à la demande et accessible via un simple navigateur web.

I.3 Définition du Cloud Computing

Il existe plusieurs définitions du Cloud Computing dans la littérature. En voici quelques unes:

Selon NIST [1], «Le Cloud Computing est un modèle qui permet un accès omniprésent, pratique et à la demande via le réseau à un ensemble partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, stockage, applications et services) qui peuvent être rapidement approvisionnées et libérées avec un minimum d'effort de gestion ou d'interaction avec les fournisseurs de services.»

Pour Buyya et al.[2] «Un Cloud est un type de système parallèle et distribué consistant en un ensemble d'ordinateurs interconnectés et virtualisés qui sont dynamiquement approvisionnés et présentés comme une ou plusieurs ressources informatiques unifiées basées sur des accords de niveau de service établis par la négociation

[1] John McCarthy, informaticien et mathématicien américain, considéré comme le fondateur de l'intelligence artificielle (AI) ; ses principales recherches portent sur la formalisation des connaissances. https://www.britannica.com/biography/John-McCarthy

[2] Joseph Carl Robnett Licklider, docteur en psychoacoustique et chercheur au laboratoire de psychoacoustique de l'Université de Harvard. Intéressé par les technologies de l'information, son but était de créer une machine plus simple à utiliser. https://www.britannica.com/biography/Joseph-Carl-Robnett-Licklider

[3] Robert "Bob" William Taylor, mathématicien et informaticien. Dans les années 1960, Taylor a proposé son idée au département de défense des USA Pentagone pour lancer un projet réseau informatique appelé ARPANET. https://www.ithistory.org/honor-roll/mr-robert-bob-william-taylor

[4] Lawrence "Larry" Gilman Roberts, ingénieur américain en informatique. Larry Roberts est considéré comme l'un des pionniers de l'Internet. https://www.nytimes.com/2018/12/30/obituaries/lawrence-g-roberts-dies-at-81.html

[5] https://www.ibm.com/

[6] https://www.amazon.com/

[7] https://aws.amazon.com/

entre le fournisseur de services et les consommateurs.»

Selon Amazon Web Services [3], «Le cloud computing est la mise à disposition de ressources informatiques à la demande via Internet, avec une tarification en fonction de votre utilisation. Au lieu d'acheter, de posséder et de gérer des serveurs et des centres de données physiques, vous pouvez accéder à votre guise aux services technologiques, tels que la puissance de calcul, le stockage et les bases de données, d'un fournisseur cloud tel qu'Amazon Web Services (AWS).»

Selon IBM [4], «Le Cloud computing, souvent appelé simplement "le Cloud", est la livraison de ressources informatiques à la demande - variant des applications aux centres de données - à travers l'Internet sur la base d'un paiement à l'utilisation.»

N. Grevet [5] dit du Cloud Computing que «C'est une manière d'utiliser l'informatique dans laquelle tout est dynamiquement couplé et évolutif et dans laquelle les ressources sont fournies sous la forme de services au travers d'Internet. Les utilisateurs n'ont ainsi besoin d'aucune connaissance ni expérience en rapport avec la technologie derrière les services proposés.»

En nous basant sur les définitions précédentes, nous avons reformulé notre propre définition du Cloud Computing qui représente une synthèse de tout ce que nous en avons compris:
Le Cloud Computing, terme Anglais aux équivalents francophones : "Informatiques en Cloud", "Informatique dématérialisée" ou "Informatique virtuelle" est un concept informatique basé sur le contrat de service qui permet l'utilisation à distance des ressources informatiques matérielles et logicielles en tant que services .
Ces ressources sont disponibles aux utilisateurs selon leur demande et ces derniers y accèdent à travers le réseau internet.

Les utilisateurs n'ont pas besoin d'avoir une connaissance préalable du mode de fonctionnement des applications du Cloud Computing ni de l'emplacement des données.La facturation des services de Cloud Computing se fait selon l'utilisation du service. Autrement dit, l'utilisateur ne doit payer que pour les ressources utilisées durant la période de son acquisition du service.

I.4 Les caractéristiques du Cloud Computing

Pour pouvoir dire qu'un service est un service de Cloud Computing, celui-ci doit posséder certaines caractéristiques. On en compte cinq ayant été définies par NIST [8] [1] et que la majorité des prestataires considèrent comme caractéristiques essentielles :

[8]NIST: National Institute of Standards and Technology

- **Service mesurable**

  Pour que la facturation du service se fasse au fur et à mesure, l'utilisation du service par le client est mise sous surveillance automatique afin de pouvoir quantifier la consommation et facturer le client convenablement. Dans le Cloud Computing, la transparence dans l'utilisation des ressources est cruciale pour les fournisseurs et les utilisateurs des services [6].

- **Accès large-bande au réseau**

  Les services de Cloud Computing doivent avoir une très bonne connectivité pour garantir une bonne qualité de service (QoS) et un accès rapide à travers les navigateurs web sur tous types de supports (PC, Smartphone, etc.)

- **Service disponible à la demande**

  Cela signifie qu'un utilisateur aura accès aux services de Cloud Computing dans l'immédiat, sans interaction avec le fournisseur et sans avoir besoin d'attendre jusqu'à ce qu'un service soit livré ou installé. La ressource désirée est instantanément accessible. Cela est aussi valable pour les organisations qui fournissent des applications accessibles sur le Cloud Computing qui peuvent mettre leurs produits à disposition sans devoir communiquer avec le fournisseur du service.

- **Mise en commun des ressources**

  Cette caractéristique permet le partage de la même ressource par plusieurs clients à travers un modèle multitenant (multi-locataire), en se basant sur des outils de virtualisation qui permettent le partage de serveurs entre plusieurs utilisateurs.

  Les ressources sont allouées dynamiquement selon les besoins des clients et ces derniers ne savent pas où se trouvent les ressources et par conséquent, ils ne savent pas non plus où se trouvent leurs données. Cependant, ils peuvent spécifier l'emplacement à un niveau d'abstraction plus élevé, tel qu'un pays [6].

- **Service flexible**

  Ceci signifie qu'il y a une élasticité de ressources sur les services de Cloud Computing. L'utilisateur peut avoir autant de ressources qu'il désire selon ses besoins. S'il y a une monté soudaine de charge, l'utilisateur peut facilement être alloué des ressources supplémentaires pour couvrir ce besoin. De la même manière, si certaines ressources ne sont plus nécessaires, elles seront allouées à d'autres utilisateurs.

I.5 Les acteurs du Cloud Computing

Le fonctionnement des services de Cloud Computing implique l'intervention de plusieurs entités qui manipulent ces services et interagissent avec eux. NIST [7] définit cinq acteurs principaux du Cloud Computing:

- **Le consommateur**

  « Une personne ou une organisation qui entretient une relation d'affaires avec le fournisseur de Cloud et utilise le service fourni. »

  Autrement dit, le consommateur accède au catalogue de services auprès du fournisseur de services de Cloud Computing, demande le service souhaité, établit un contrat de niveau service avec le fournisseur de services de Cloud Computing, puis utilise le service [8].

- **Le fournisseur**

  « Une personne, une organisation qui est l'entité responsable de mettre à disposition des services en Cloud aux tiers intéressés. »

  Le fournisseur doit avoir l'infrastructure nécessaire pour fournir les services et il doit également s'occuper de la livraison de ces services aux clients à travers le réseau. Le fournisseur crée des services (applications, plateformes, infrastructures), gère et fournit ces services, et garantit leur sécurité [8].

- **L'auditeur**

  « Un tiers qui peut procéder à une évaluation indépendante des services du Cloud, des opérations des systèmes d'information, de la performance et de la sécurité de l'implémentation du Cloud. »

  Les auditeurs communiquent une opinion objective après avoir fait leur examen et confirment si tout est conforme aux critères. Les auditeurs sont des entités chargées d'évaluer les performances et la confidentialité des services de Cloud Computing [8].

- **Le courtier (broker)**

  « Une entité qui gère l'utilisation, la performance et la livraison de services Cloud et négocie également les relations entre les fournisseurs et les consommateurs du Cloud. »

  Un courtier (Broker) est un intermédiaire entre le fournisseur de services de Cloud Computing et le consommateur. Un courtier aide donc les consommateurs en gérant les intégrations complexes des services de Cloud Computing et en communiquant avec les fournisseurs à leur place. Les courtiers peuvent être utilisés pour augmenter la capacité des services, fusionner et intégrer plusieurs services ou sélectionner le meilleur service parmi un ensemble de services existant [8].

- **Le transporteur (carrier)**

  « L'intermédiaire qui fournit la connectivité et le transport des services Cloud du fournisseur du Cloud vers le consommateur. »

  Le transporteur est donc responsable d'assurer l'accès aux services de Cloud Computing à travers le réseau pour les consommateurs [8].

I.6 Modèles de livraison

Dans le Cloud Computing, un modèle de livraison représente le niveau d'abstraction sur lequel le service est basé. Ces niveaux sont reliés entre eux de manière à ce que les niveaux les plus bas servent de base sur laquelle les autre niveaux sont fondés.

Dans [1] et [9], les services de Cloud Computing ont été classés en trois catégories selon leur modèle de livraison IaaS, PaaS, SaaS comme le montre la figure suivante:

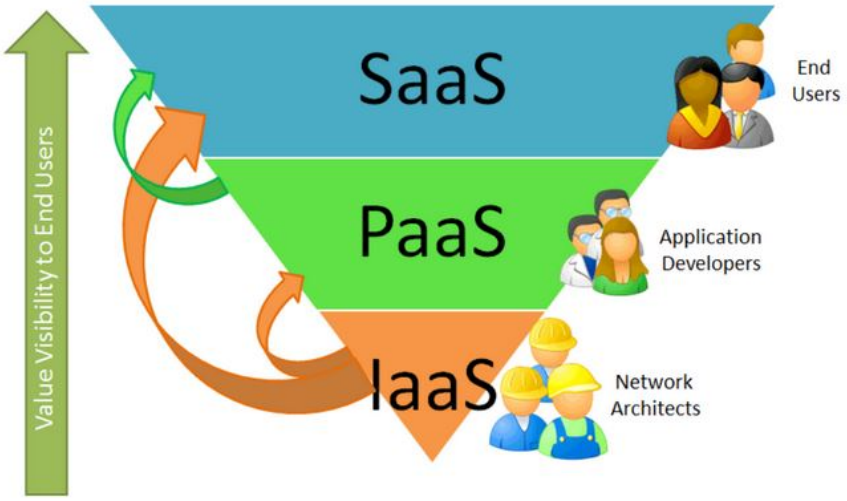


Figure I.1: Modèles de livraison des services de Cloud Computing [5].

I.6.1 IaaS

IaaS signifie « Infrastructure as a Service ». Dans ce type de service, le consommateur peut acquérir de la puissance de calcul, des capacités de stockage, des réseaux et d'autres ressources informatiques qu'il peut exploiter pour déployer et exécuter ses logiciels, y compris des systèmes d'exploitation et des applications. Bien qu'il ait la liberté de contrôler les ressources mises à sa disposition, il n'a pas à s'occuper de la gestion ni du contrôle de l'infrastructure sous-jacente.

Les développeurs peuvent créer la plate-forme requise pour leur travail de développement logiciel. Ce type de service permet aux clients de créer simplement des instances des machines virtuelles dont ils ont besoin. Ces instances virtuelles sont installées a priori dans des pools d'équipements nommés Datacenter.

Les services du modèle IaaS présentent les caractéristiques suivantes:

- Possibilité de fournir aux clients un environnement personnalisé.
- Disponibilité de la dernière version.

- Réduction des coûts de maintenance des équipements.
- Assurance de la sécurité des données stockées.
- Création des instances virtuelles en fonction des besoins du client; ces instances peuvent être utilisées comme services d'externalisation.

I.6.2 PaaS

Le modèle «Platform as a Service » offre aux consommateurs des outils (des langages de programmation, des bibliothèques, etc.) qui leur permettent de concevoir et de déployer des applications. Le consommateur ne s'occupe ni de la gestion ni du contrôle de l'infrastructure sous-jacente et cela comprend le réseau, les serveurs, les systèmes d'exploitation et le stockage. Cependant, il peut contrôler les applications qu'il déploie et il lui est possible de configurer les paramètres pour l'environnement d'hébergement des applications.

Avec l'arrivée du modèle PaaS, le problème de l'incompatibilité de l'environnement logiciel sur la machine a été résolue. Ceci a été fait de sorte que toute personne disposant d'une connexion Internet peut immédiatement développer des applications puissantes sans se soucier des problèmes d'infrastructure et de coût.

Le modèle PaaS présente les caractéristiques suivantes:

- Les clients n'ont pas à se soucier du coût de l'infrastructure.
- Les utilisateurs peuvent créer et déployer des applications faciles et rapides à exécuter.
- La plate-forme fournie sera mise à jour régulièrement et les applications pourront être développées sur la base de la meilleure technologie.
- Le temps de développement est minimisé.

I.6.3 SaaS

Cet acronyme signifie « Software as a Service » qui, traduit au Français veut dire « application en tant que service ». Les fournisseurs de ce type de service mettent en disposition des solutions applicatives. Ces applications sont hébergées sur des infrastructure de Cloud Computing et ont l'avantage d'être accessibles via des interfaces clients tel que les navigateurs web ou des interfaces de programmes. Le consommateur n'a pas à gérer ou contrôler l'infrastructure sous-jacente mais peut avoir la possibilité de configurer les paramètres de l'application.

Les solutions SaaS comprennent des logiciels de gestion de la relation client (CRM [9]), la planification des ressources d'entreprise (ERP [10]), la comptabilité et d'autres logiciels utiles aux entreprises, dont la plupart sont

[9] Customer Relationship Management
[10] Enterprise Resource Planning

des logiciels de compétences générales. Actuellement, la plupart des entreprises ont adopté une solution SaaS qui n'exige pas que les employés connaissent les détails de l'infrastructure et de la plate-forme pour exécuter l'application.

Le modèle SaaS présente les caractéristiques suivantes:

- Temps de développement réduit, les applications sont déjà disponibles à l'échelle mondiale.
- Assurer la cohérence et la compatibilité des données.
- Les logiciels fournis sont flexibles et extensibles.
- Les fournisseurs de services offrent les versions les plus récentes du logiciel SaaS (gestion des mises à jour). Le Tableau I.1 represente quelques services de Cloud Computing disponibles sur le marché. On compte Amazon, Google[11], Microsoft [12], Salesforce [13], et Oracle [14] parmi les préstataires les plus populaires qui offrent ces services.

Prestataire	Service	Modèle de livraison
Google	Google Compute Engine	IaaS
	Google App Engine	PaaS
	G Suite	SaaS
Microsoft	Windows Azure Virtual Machines	IaaS
	Windows Azure PaaS	PaaS
Amazon	Amazon Elastic Cloud Compute (EC2)	IaaS
	Amazon Simple Storage Service	IaaS
Salesforce	Salesforce CRM	SaaS
Oracle	Oracle CRM	SaaS

Tableau I.1: Exemples de services de Cloud Computing disponibles sur le marché.

I.6.4 XaaS

Le "X" de l'acronyme XaaS veut dire « anything » et on en conclu alors que le XaaS signifie « Anything as a Service » ou bien « n'importe quoi en tant que service » et cette appellation insinue que n'importe quel concept peut être adapté en service de Cloud Computing. D'ailleurs le XaaS a été définie comme « N'importe quelle technologie livrée par internet qui était livré sur place auparavant. » [10]

Le XaaS explique pourquoi on trouve des acronymes autres que le IaaS, PaaS et SaaS. En voici quelques exemples :

[11] https://cloud.google.com/
[12] https://azure.microsoft.com
[13] https://www.salesforce.com/fr/
[14] https://www.oracle.com

-**DaaS** : Desktop as a Service, ou Station de travail en tant que service.

-**STaaS**: Storage as a Service, cela veut dire Stockage en tant que service.

-**VPNaaS**: VPN [15] as a Service, VPN en tant que service.

-**HaaS**: Hardware as a Service. En français, cela se traduit en Matériel en tant que service.

-**NaaS**: Network as a Service, Réseau en tant que service.

-**CaaS**: Containers as a Service, Centeneur en tant que service.

La Figure I.2 représente un exemple de service XaaS, le Blockchain as a Service [16].

Figure I.2: Exemples de fournisseurs de Blockchain as a Service[11].

[15] Virtual Private Network

[16] La blockchain est une technologie qui permet le stockage et la transmission d'informations de manière sécurisée.

I.7 Modèles de déploiement

Les modèles de déploiement représentent la restriction d'accès aux services de Cloud Computing. Certains services sont accessibles au public, tandis que d'autres sont privés ou restreints à une communauté spécifique. NIST [1] a pris le soin de définir les modèles de déploiement de services Cloud Computing comme suit :

I.7.1 Cloud privé

Celui-ci est destiné exclusivement à certains consommateurs ou organisations. L'accès aux services du Cloud privé n'est pas public, mais se fait via un réseau privé. Pour créer un Cloud privé, il existe deux types de solutions: les solutions internes et les solutions externes.

Une solution interne signifie que l'organisation héberge directement des services du Cloud privé dans sa propre infrastructure, ce qui est illustré dans la Figure I.3.

Une solution externe signifie que l'organisation choisit d'héberger ses services chez un fournisseur de services de Cloud Computing et que chaque utilisateur accède à ces services via un réseau sécurisé [12]. Ceci est bien illustré dans la Figure I.4.

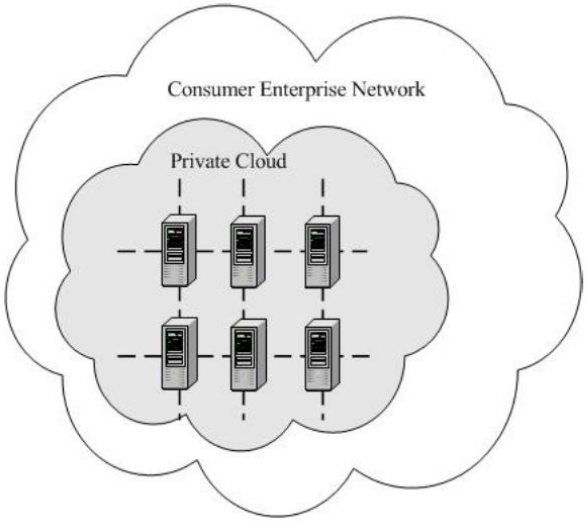


Figure I.3: Cloud privé interne[12].

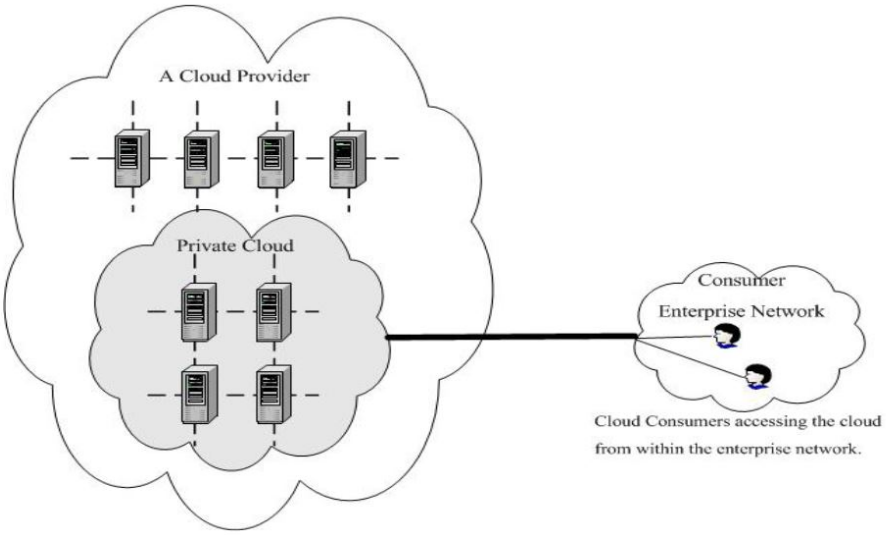


Figure I.4: Cloud privé externe[12].

I.7.2 Cloud Public

Dans ce modèle, le fournisseur de service met ses ressources (des machines virtuelles, des applications, des capacités de stockage, etc.) à la disposition du grand public qui les utilise librement via le réseau public. Ces ressources peuvent être gratuites ou vendues à la demande. Elles peuvent peut être possédées, gérées et opérées par un commerce, une entité académique, une organisation gouvernementale, ou une combinaison de toutes ces entités.

I.7.3 Cloud Communautaire

Ce modèle est exclusivement partagé par plusieurs utilisateurs venant d'organisations ayant des intérêts en communs ou qui partagent les mêmes exigences de confidentialité et de sécurité et qui forment une communauté. Ce modèle peut être mis en œuvre sur site (en interne) ou sous-traité à une société d'hébergement. Il peut être possédé, géré et opéré par une ou plusieurs organisations de la communauté, un tiers, ou une combinaison des deux.

I.7.4 Cloud Hybride

Dans ce modèle, on note l'utilisation d'une combinaison de deux ou plusieurs infrastructures de Cloud Computing distinctes (Privé, communautaire ou public) qui demeurent des entités uniques mais qui sont reliés entre eux à travers des technologies standardisés ou propriétaires qui permettent la portabilité des données et applications.

I.8 Service Level Agreement (SLA)

Un Service Level Agreement ou en français un « Accord de niveau de service » ou « contrat de niveau de service » est un accord entre un fournisseur de service et un consommateur auquel ces deux derniers arrivent après des

négociations concernant la qualité de service attendue par le consommateur [13], la régulation des ressources et le contrôle des coûts ainsi que tout autre critère auquel le fournisseur s'engage à répondre.

Cependant cette définition n'est pas absolue et peut différer d'un domaine à un autre. Voici quelques définitions données par des chercheurs dans leurs contextes respectifs [14] :

Internet: « SLA construit la fondation légale de la livraison de service. Toutes les parties impliquées sont des utilisateurs du SLA. Les consommateurs de services utilisent le SLA comme une description juridiquement contraignante de ce que le fournisseur promet de fournir. Le fournisseur de service l'utilise pour avoir un dossier définit et contraignant de ce qui est destiné à être livré. »[14]

Gestion de centre de données: « Le SLA est un accord formel à promettre ce qui est possible de fournir et de fournir ce qui a été promis. » [15]

Services Web : « Le SLA est un accord utilisé pour garantir la livraison des services Web. Il définit la compréhension et les attentes du fournisseur de service et du consommateur de service » [16]

I.9 La Virtualisation

La virtualisation est une technologie clé qui entre en jeu dans le Cloud Computing et plus précisément pour la mise en commun d'infrastructures, de plateformes et d'applications . En pratique, elle permet aussi « l'affectation ou la réaffectation dynamique des ressources virtuelles ou des applications. » comme le précisent les auteurs dans[17]. En réalité, la virtualisation entre en jeu dans plusieurs domaines informatiques et elle consiste à exécuter une machine virtuelle sur une machine physique (machine hôte) [18].

En d'autres termes, N. Grevet dans [5] définit la virtualisation comme étant « l'ensemble des techniques matérielles et/ou logicielles qui permettent de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation et/ou plusieurs applications, séparément les uns des autres, comme s'ils fonctionnaient sur des machines physiques distinctes. »

La virtualisation permet une exploitation maximale de la machine et permet également de minimiser les coûts grâce à la possibilité de mettre en marche plusieurs machines virtuelles sur une seule machine physique sans avoir besoin d'en acheter d'autres.[18]

I.9.1 L'hyperviseur

Élément principale du Cloud Computing, il s'agit d'un logiciel dont la fonction est la virtualisation du matériel. Un hyperviseur rend possible le fonctionnement de plusieurs systèmes d'exploitation sur une seule et même machine physique[19]. Ainsi, chaque système d'exploitation fonctionne sur une instance de la machine physique que l'on

appelle machine virtuelle. L'hyperviseur a aussi pour tâche la gouvernance des machines virtuelles et la gestion des ressources qui leur sont affectées.

I.9.2 La virtualisation basée sur les conteneurs

Un conteneur est un logiciel qui s'occupe de l'emballage du code et des dépendances d'une application pour que celle-ci puisse être exécutée de manière rapide et fiable dans n'importe quel environnement. Il est possible d'exécuter plusieurs conteneurs à la fois sur la même machine. Ceux-ci se partagent les ressources du système d'exploitation mais se comportent comme des processus isolés [20].

La virtualisation basée sur des conteneurs, également connue sous le nom de "virtualisation au niveau du système d'exploitation", ne nécessite pas de machines virtuelles complètes, elle n'utilise qu'un seul système d'exploitation pour gérer et accéder au matériel ; en d'autres termes, elle " utilise un seul noyau pour exécuter plusieurs instances d'un système d'exploitation ". Cela signifie que les performances sont optimisées car il n'y a pas d'appels multiples au matériel depuis plusieurs systèmes d'exploitation qui provoqueraient des surcharges d'un coté, et de l'autre, moins de ressources sont utilisées puisqu'il n'y a qu'un seul système d'exploitation.

Les instances de conteneurs sont plus petites et plus rapides à créer et à migrer, ce qui permet de faire fonctionner un grand nombre de conteneurs sur un seul système, par rapport à un nombre relativement moins important de VM si l'on opte pour l'autre option de virtualisation.

Cependant, il y a quelques inconvénients à travailler avec la virtualisation basée sur des conteneurs, comme le fait d'être limité dans le choix du système d'exploitation et le problème du point de défaillance unique auquel peuvent être confrontés les conteneurs qui fonctionnent sur le même système d'exploitation.
La virtualisation basée sur les conteneurs est intéressante pour les fournisseurs de cloud Computing en raison de la possibilité de déployer un grand nombre de conteneurs sur le même matériel [21].

I.10 L'Inter-cloud, les fédérations de cloud et le Multi-cloud

L'Inter-cloud, les fédérations et le Multi-cloud sont des concepts qui sont apparus peu après que le Cloud Computing n'eût gagné en popularité. Ces concepts sont apparus pour étendre ce domaine et sont une partie très importante dans son évolution.

I.10.1 Ambiguïté et confusion

Bien que les concepts d'Inter-cloud, de Multi-cloud et de fédération ont prit de l'élan dans le monde informatique après la grande popularité du Cloud Computing, ils restent peu clairement définis par les chercheurs du domaine. D'ailleurs, la définition de ces concepts est caractérisée par une certaine ambiguïté.
Cette ambiguïté se manifeste clairement dans la divergence que nous avons noté dans les travaux [22], [23] et [24].

En effet, il existe des travaux dans la littérature comme certains qui sont cités dans [22] qui considèrent que l'Inter-cloud et le Multi-cloud comme étant des termes synonymes et les utilisent en alternance. Pour cette catégorie, « Le terme "Multi-cloud" est semblable aux termes "interclouds" ou "cloud-de-clouds"… »
M. AlZain et al. [22] cite aussi une autre catégorie de chercheurs ayant une opinion différente, ceux-ci estiment qu'il existe « ...deux couches dans l'environnement Multicloud : la couche inférieure est l'inner-cloud, tandis que la deuxième couche est l'Inter-cloud.»

Par contre, H. Kurdi et al. dans [23] considèrent l'Inter-cloud comme une des approches implémentant le concept de Cloud multiples qui fait référence aux services de Cloud Computing utilisés à travers des fournisseurs multiples. Elles ont aussi donné les termes Multi-cloud et fédération comme des synonymes faisant référence à ce concept (L'intercloud).

Mais, N. Groznev et R. Buyaa dans [24] avaient vu les choses autrement et avaient fait la distinction entre les fédérations et les Multi-cloud et les avaient considérés comme des types de l'Inter-cloud.

On trouve aussi des chercheurs tels que B. Kezia Rani et al. [25] et D. Petcu [26] qui sont d'accord avec N. Grozev et R. Buyya [24] et considèrent le Multi-cloud et les fédérations comme des types de l'Inter-cloud.
Dans notre travail, nous adoptons la même vision que celle de ces chercheurs et nous définissons ces concepts selon cette vision.

I.10.2 Inter-cloud

Pour définir l'Inter-cloud, nous avons choisi de nous appuyer sur le contenu des travaux [24] et [26] .

Dans [24], les auteurs définissent l'Inter-cloud comme « une démarche qui implique d'interconnecter les infrastructures de plusieurs fournisseurs de services Cloud.» Ils précisent aussi que « les fédérations et les Multi-clouds sont des types d'Inter-Clouds.»

D. Petcu [26] a une vision semblable et voit l'Inter-cloud comme étant « un cloud fédéré ou un Multi-cloud ayant au moins un broker et offrant une mise à disposition de service dynamique.»

Ce qui nous amène à conclure, d'après ces définitions, qu'un Inter-cloud est un ensemble de serveurs de Cloud Computing interconnectés provenant de plusieurs fournisseurs. Cet ensemble de serveurs de Cloud Computing peut être une fédération ou un Multi-cloud.

La Figure I.5 représente les formes de l'Inter-cloud tel comme elles sont présentées dans [24] ainsi que leurs exemples.

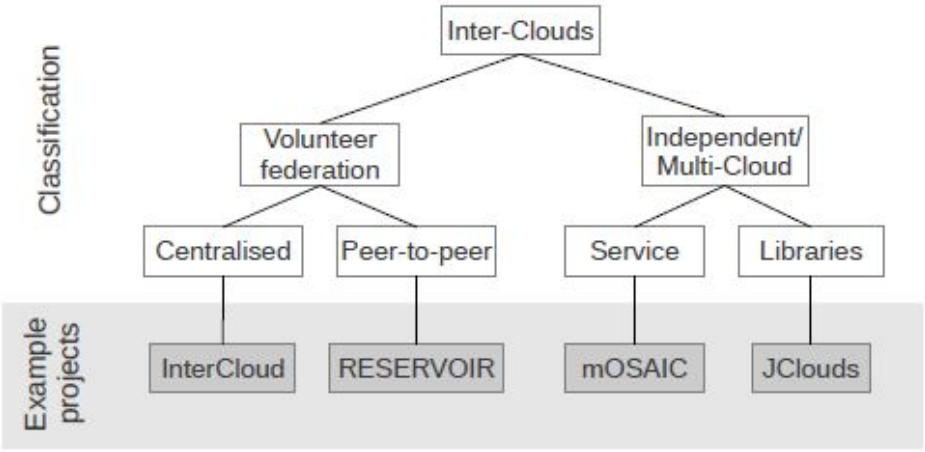


Figure I.5: Classification architecturale des Inter-clouds[24].

I.10.3 Multi-cloud

Pour définir le Multi-cloud, nous avons aussi décidé de nous limiter aux définitions données dans [24] et [26] .

Pour Buyaa. et Grozev [24], «Le terme "Multi-Cloud" désigne l'utilisation de plusieurs clouds indépendants par un client ou un service.» .Un Multi-cloud est formé lorsque « des cloud multiples sont utilisés en agrégation par une application ou son broker... »

Ces chercheurs précisent que l'approche du Multi-cloud est «... essentiellement indépendante du fournisseur de cloud. » et que les clients ou leurs représentants sont responsables de l'approvisionnement et de la planification. » Ils indiquent aussi que l'interconnection et le partage de ressources ne se passent pas de façon volontaire de la part des fournisseurs dans les Multi-cloud .

Selon eux, il y a deux types de Multi-clouds:

Les services : «... le provisionnement des applications est effectué par un service qui peut être hébergé soit en externe, soit en interne par les clients du cloud. La plupart de ces services comprennent des composants de broker en eux-mêmes.»

Les librairies : «… Il faut souvent des brokers d'applications personnalisés qui s'occupent directement de l'approvisionnement et de la planification des composants d'applications à travers les Clouds. Généralement, ces approches font appel à des librairies Inter-clouds qui facilitent l'utilisation de plusieurs clouds de manière uniforme..»

D. Petcu [26] ajoute un détail concernant le degré de collaboration entre les services de Cloud Computing dans le Multi-cloud , elle cite un travail qui précise qu'il n'y a pas d'accord entre les fournisseurs pour ce qui est du partage

de leurs ressources.

D'après les définitions mentionnées, un Multi-cloud désigne l'utilisation de plusieurs services de Cloud indépendants par un client ou un service qui implique un broker qui réalise l'agrégation des Clouds en question. Seuls les clients sont responsables de cette utilisation qui est indépendante du fournisseur. D'ailleurs, l'interconnection et le partage de ressources qui a lieu dans le Multi-cloud sont réalisés sans l'accord du fournisseur et n'est pas fait de manière volontaire. Les Multi-cloud sont présents sous deux formes: les services ou les librairies.

I.10.4 Fédérations de Clouds

Nous allons également nous appuyer sur les travaux [24] et [26] pour définir les fédérations.

Selon [24] «Une fédération est réalisée lorsqu'un ensemble de fournisseurs de services en nuage interconnectent volontairement leurs infrastructures afin de permettre le partage des ressources entre eux.»

De plus, Buyya et Grozev [24] communiquent qu'il existe deux types de fédération:

Les fédérations Peer to Peer : où «... les Clouds communiquent et négocient directement entre eux, sans médiateur.»

Les fédérations centralisées : celles-ci sont caractérisées par «... une entité centrale qui effectue ou facilite l'allocation des ressources. En général, cette entité centrale fait office de dépôt où sont enregistrées les ressources disponibles dans le nuage, mais elle peut aussi avoir d'autres responsabilités, comme celle de servir de marché pour les ressources.»

Dans [26] , Dana Petcu a expliqué que dans les fédérations, «... il y a un accord entre les fournisseurs de partager leurs ressources.» Elle a aussi dit que dans les fédérations, le client interagit avec un seul service de Cloud Computing et n'a aucune connaissance du fait que ce service consomme des ressources et des services provenant d'autres services de Cloud Computing.

Donc, nous faisons la synthèse des définitions précédentes et nous en gardons qu'une fédération est une interconnection des ressources de plusieurs fournisseurs faite de manière volontaire et avec l'accord des fournisseurs en question ayant pour but le partage de leurs ressources entre eux. La collaboration est transparente au client qui n'interagit qu'avec un seul service de Cloud Computing. Il y a deux types de fédération: les fédérations Peer to peer, et les fédérations centralisées

I.11 Avantages du Cloud Computing

Il y a de bonnes raisons pour lesquelles le Cloud Computing est devenu une tendance informatique aujourd'hui. En effet, ce domaine offre beaucoup d'avantages parmi lesquels on compte [27]:

- **Capacité de stockage illimitée** : « Nous travaillons dans un nuage, nos données sont stockées dans une centaine de pétaoctets de stockage qui a été disponible dans un nuage. Sans aucune limitation, les données peuvent être stockées et consultées. »
  La particularité du Cloud Computing est le stockage illimité, ou plutôt l'illusion d'avoir un stockage illimité grâce à la prise en charge par de puissants serveurs. Ceci est particulièrement utile quand nous avons un travail qui demande une grande capacité de stockage et notre ordinateur ne dispose pas d'une mémoire suffisante. C'est le Cloud Computing qui assure cette capacité.

- **Accès universel aux documents** : «Si vous chargez vos données dans le cloud Il n'y aura pas de problème. Lorsqu'il y a une disponibilité de données dans le cloud ces données sont alors accessibles de n'importe où, il n'est pas nécessaire de transporter le document où que vous soyez. Avec la connexion Internet constante, tous vos documents sont accessibles à tout moment. »
  Le Cloud Computing nous permet de télécharger nos documents sans les transporter et nous pouvons donc y accéder à tout moment et de n'importe où.

- **La disponibilité de la dernière version** : « Les utilisateurs du cloud n'ont pas besoin de mettre à jour le logiciel ou l'application qu'ils utilisent constamment, le cloud proposera toujours la version mise à jour. »
  Cela évite de devoir penser à effectuer des mises à jours manuellement et a devoir s'assurer qu'on a accès à la même version de l'application partout où l'on est.

- **Collaboration en groupe plus facile** : « Cela semble être l'un des avantages les plus importants et uniques du cloud computing, où la collaboration des documents est facile dans le cloud. Ici le cloud offre la possibilité à plusieurs utilisateurs de collaborer dans un seul document. »
  Avant, quand le travail se passait au sein d'un projet collectif, les personnes communiquaient à l'aide des courriels; mais après l'apparition du Cloud Computing, la collaboration en groupe est devenue plus facile car on peut consulter les modifications faites par un autre membre du groupe.

- **Réduction des coûts** : « Pour exécuter une application Web ou un logiciel de cloud computing, l'utilisateur n'a pas besoin de maintenir les ordinateurs de grande puissance. L'application Web que nous utilisons est en cours d'exécution sur un serveur géant en nuage et non pas sur notre propre ordinateur. Ainsi, l'utilisateur du cloud a simplement besoin des ordinateurs bon marché dotés d'un disque dur plus petit, ayant moins de

mémoire. »

Dans le cas où nous voulons exécuter un service de Cloud Computing à travers un navigateur web, il n'est pas nécessaire d'avoir un ordinateur puissant du moment que l'ordinateur est en bonne état, il fera l'affaire.

I.12 Inconvénients du Cloud Computing

Bien évidemment, comme toute autre technologie, le Cloud Computing n'a pas que des avantages, il possède aussi des inconvénients comme [27] :

- **La nécessité d'une connexion Internet constante** : « Le cloud computing ne peut rien faire pour vous si vous n'avez pas une connexion Internet parce que les documents sur lesquels vous travaillez ont été enregistrés dans le cloud. Si vous n'avez pas de connexion Internet, vous ne pourrez pas accéder à vos propres documents. Pour travailler dans un cloud, vous devez disposer d'une connexion Internet haut débit. » .

  L'un des inconvénients de l'utilisation des services de Cloud Computing est la nécessité d'une connexion Internet de haut débit. Cela est nécéssaire à cause du fait que le Cloud Computing ne fonctionne pas avec une connexion Internet à faible débit. Donc, sans connexion ou avec une connexion médiocre les services de Cloud Computing devient inutile.

- **Les données stockées peuvent ne pas être sécurisées** : « Les données chargées sur un serveur seront répliquées sur tous les serveurs, ce qui modifiera la disponibilité des données en double. En raison de cela, la sécurité présente un risque élevé. Les données en cloud peuvent être consultées par n'importe qui, de n'importe où. » .

  La sécurité des données confidentielles n'est pas assurée par le Cloud Computing. Quand un utilisateur charge des données dans un nuage, celles-ci seront vulnérables et à la portée des cybercriminels.

- **Peut-être lent** : « Le cloud étant utilisé dans le monde entier, des centaines de personnes utilisent la même application au même moment. » .

  Dans ce cas, l'utilisateur ne peut pas accéder à la ressource qu'il sollicite malgré la disponibilité d'une connexion de haut débit à cause de la surcharge.

- **Comparaison des coûts** : « Il est vrai que le coût est élevé pour le logiciel que vous installez chez vous par rapport au logiciel cloud. Mais lorsque vous comparez cette fonctionnalité, le logiciel que vous installez dans votre ordinateur sera, peut être, meilleur que le logiciel cloud. Il y a des chances que le logiciel du cloud ait une fonctionnalité médiocre et vous serez facturé en plus pour l'utilisation de quelques fonctionnalités. »

  Dans ce cas, on doit faire des compromis après avoir bien penser aux avantages et inconvénients pour déterminer si l'utilisation des services de Cloud Computing, malgré son coût, est le meilleur choix.

I.13 Défis du Cloud Computing

Les avantages et les inconvénients mis à part, comme toute autre technologie le Cloud Computing est en constante évolution et doit faire face à de nombreux défis. A. Verma et al. dans [28] ont énuméré quelques uns de ces défis comme suite :

- **L'interopérabilité:** Ceci veut dire la capacité d'une application de Cloud Computing opérant sur une plateforme d'intégrer des services opérant sur d'autres plateformes ou de collaborer avec eux. C'est à dire, la capacité que possède une application à fonctionner avec d'autres applications.
- **La sécurité:** Ceci est peut-être le plus grand défi du Cloud Computing et c'est le fait de pouvoir garder les données sur le Cloud privées et sécurisées.
- **La portabilité:** Autrement dit, la capacité de transférer une application de Cloud Computing de la plateforme d'un fournisseur à celle d'un autre. Ceci est difficile car les fournisseurs de services Cloud Computing n'utilisent pas tous les mêmes modèles et technologies.
- **La capacité de calcul:** Les applications qui manipulent beaucoup de données requièrent une haute connectivité au réseau, ce qui augmente les frais. Une basse capacité de transfert de données ne couvre pas la performance attendue des applications de Cloud Computing.
- **La fiabilité et disponibilité:** La fondation du service de Cloud Computing doit être solide et fiable pour un service constamment disponible à la demande.

I.14 Conclusion

Bien que le Cloud Computing soit une technologie qui ait fait surface il y a plus de 14 ans maintenant, le domaine reste encore relativement jeune et ouvert à l'exploration. C'est aussi un domaine de l'informatique qui a du potentiel à évoluer avec l'évolution de l'internet, et qui est déjà en croissance accélérée.

Dans ce chapitre, une définition du Cloud Computing a été fournie ainsi que les caractéristiques, les acteurs, le SLA, les différents modèles de livraison, les modèles de déploiement, et d'autres informations supplémentaires tels que la virtualisation et l'inter-cloud. Nous avons aussi parlé des avantages, inconvénients et défis pour mettre en perspective le domaine du Cloud Computing dans le monde de l'informatique.

Dans le chapitre suivant, nous allons explorer l'un des défis auxquels le Cloud Computing est confronté et qui demeure sans solution concrète. Il s'agit de l'interopérabilité.

Chapitre II

L’interopérabilité dans le Cloud Computing

II.1 Introduction

Avec le temps, les progrès du domaine informatique ont augmenté de manière exponentielle, ce qui a impliqué une augmentation des nouvelles technologies et systèmes, tous différents les uns des autres. Cette différence a poussé les chercheurs à trouver un moyen pour permettre à ces systèmes de travailler et interagir ensemble; il s'agit de l'interopérabilité.

Dans ce chapitre, nous allons définir la problématique de l'interopérabilité, et plus spécifiquement dans le Cloud Computing. Nous présenterons également des concepts tels que la portabilité et la migration. Nous allons également parler des classifications relatives aux types et niveaux de l'interopérabilité. Puis, nous allons dédier une partie du chapitre à certains concepts qui sont liés à l'interopérabilité, notamment la description, la découverte, l'invocation et la composition des services. Ensuite, nous allons aborder les problèmes et exigences de l'interopérabilité avant de parler des façons de réalisation de l'interopérabilité.

II.2 Définition de l'interopérabilité

Il existe plusieurs définitions de l'interopérabilité dans la littérature, nous en citons quelques unes que nous avons jugé pertinentes :

La définition de [19] précise que l'interopérabilité est « la capacité de différents systèmes hétérogènes à être capables de fonctionner/interagir ensemble. »

Le travail [29] offre plusieurs définitions recueillies de plusieurs différentes sources:

«...l'interopérabilité peut être définie comme mesure du degré auquel divers systèmes, organisations, et/ou individus sont en mesure de travailler ensemble pour réaliser un objectif commun.»

« La capacité de deux ou plusieurs systèmes ou composants à échanger des informations et à utiliser les informations qui ont été échangées. »

«Pouvoir réaliser des applications destinées aux utilisateurs finaux en utilisant différents types de systèmes informatiques, de systèmes d'exploitation, et de logiciels d'application, interconnectés par de différents types de réseaux locaux et étendus. »

Les chercheurs dans [29] ont également formulé leur propre définition qui indique que l'interopérabilité est « La capacité d'interaction d'un service avec d'autres services homogènes ou hétérogènes pour améliorer son service, qui

pourrait être mis en œuvre sous un ou sous différents domaines. »

II.3 L'interopérabilité dans le cadre du Cloud Computing

Pour l'interopérabilité dans le Cloud Computing, il existe également plusieurs définitions dans la littérature. Nous en avons choisis quelques unes que nous considérons adéquates:

Dans le Cloud Computing, le travail [19] considère l'interopérabilité comme « ... la capacité à comprendre les formats d'application des autres, les modèles d'accord de niveau de service (SLA), les formats de jetons d'authentification et d'autorisation et les données d'attributs.»

Tandis que [30] estime que c'est «... la capacité des clouds publics, des clouds privés et d'autres systèmes divers au sein de l'entreprise à comprendre les interfaces d'application et de service, la configuration, les formes d'authentification et d'autorisation, les formats de données, etc. les uns des autres afin de coopérer et d'interagir les uns avec les autres. »

Le travail [31] la définit comme «... l'exigence selon laquelle les composants d'un système de traitement doivent fonctionner ensemble pour atteindre leur résultat prévu. Les composants doivent pouvoir être remplacés par de nouveaux composants ou différents composants provenant de différents fournisseurs et doivent continuer à fonctionner. »

D'après [32], la définition de l'interopérabilité dans le Cloud Computing est la suivante: «... capacité des applications fonctionnant dans différents clouds à partager des données, application à transférer vers une autre solution de cloud ou ayant les mêmes fonctionnalités et options dans différentes plateformes ou solutions de cloud. La portabilité, la gestion et la migration des données et des images entre différentes solutions de cloud ne sont pas non plus exclues.»

Les auteurs de [33] donnent une définition différente des précédentes. Pour eux, l'interopérabilité est «La capacité d'écrire un code qui fonctionne avec plus d'un fournisseur de services dans le Cloud Computing simultanément, indépendamment des différences entre les prestataires.»

Selon [34], «L'interopérabilité des clouds fait référence à la facilité de migration et d'intégration des applications et des données entre les clouds de différents fournisseurs.»

En nous basant sur les définitions ci-dessus, nous définissons nous mêmes l'interopérabilité comme étant la capacité des services Cloud à comprendre les différents éléments qui composent d'autres services provenant de

différents fournisseurs afin d'interagir avec eux et d'établir un échange de données et d'application pour atteindre un objectif commun. Elle est également relative à la facilité de migration, de portabilité et d'intégration des données et applications au sein d'un environnement de Cloud Computing.

II.4 La compatibilité des Clouds

L'interopérabilité est étroitement liée au concept de la compatibilité entre les services du Cloud Computing. D'ailleurs, ces deux concepts sont souvent confondus [35]. Nous avons donc jugé que ce concept mérite d'être mentionné dans notre travail.

Le travail [31] la définit comme étant « ... la capacité de l'application et des données à fonctionner de la même manière quelque soit le modèle de service (IaaS, PaaS, SaaS) ou les modèles de déploiement (privé, public et hybride) et le lieu (interne ou externe à l'entreprise). »

Le papier [36] présente une définition plus brève qui considère que « ... La compatibilité dans le Cloud signifie que l'application et les données peuvent fonctionner de la même manière quelque soit le fournisseur du Cloud. »

En prenant en compte ces deux définitions, nous définissons nous mêmes la compatibilité comme étant:
La capacité des applications et des données de fonctionner d'une manière identique sans devoir prendre en compte les modèles de service, les modèles de déploiement, l'emplacement et le fournisseur de services.

II.5 L'intégration entre les applications

L'intégration des applications est un élément clé dans l'interopérabilité [34]. En réalité, Elle consiste à réaliser l'interaction entre les applications et à rendre le partage des données plus facile [37].

L'intégration des applications est classée en trois types [30]:

L'intégration des processus (ou de contrôle): Dans ce cas, une application invoque une autre application pour exécuter un flux de travail.

L'intégration des données: C'est lorsque les applications partagent des données communes, ou lorsqu'une application utilise les données de l'autre, comme dans le cas où ce que produit une application en sortie est utilisé comme l'entrée d'une autre application.

L'intégration de présentation: C'est le cas où les résultats du traitement de plusieurs applications est présenté à l'utilisateur de manière simultanée par le devis d'un tableau de bord.

II.6 La portabilité

La définition de l'interopérabilité dans le Cloud Computing dans [32] a également inclus le concept de la portabilité.

D'après [30], « La portabilité concerne la capacité de déplacer une entité d'un système à un autre afin qu'elle soit utilisable sur le système cible.»

Les auteurs dans [31] présentent une définition plus détaillée : « Le concept de portabilité tel qu'il s'applique au cloud prévoit pour les composants d'application et de données de continuer à fonctionner de la même manière lorsqu'ils sont déplacés d'un environnement de cloud à un autre sans avoir à être modifiés ...» Ce travail précise aussi que, dans le Cloud Computing, «...Un composant portable peut être déplacé facilement et réutilisé, quelque soit le fournisseur, la plate-forme, le système d'exploitation, l'emplacement, le stockage ou d'autres éléments de l'environnement environnant. »
C'est sur cette deuxième définition que nous nous basons dans notre travail.

Il existe deux types de portabilité :

La portabilité des applications : celle-ci signifie que les applications exécutées dans un environnement de Cloud Computing peuvent être déplacées vers un autre et exécutées, en conséquence, sans avoir à les traiter au préalable [31].

La portabilité des données: c'est la possibilité de déplacer simplement des données d'un service de Cloud Computing vers un autre sans avoir à ressaisir les données de nouveau, les modifier, ou modifier l'application qui est censée les utiliser [31] . Pour les services source et cible, la syntaxe et la sémantique des données doivent généralement être les mêmes, sinon un mapping des données doit être fait en utilisant les outils disponibles [30].

II.7 La migration

Le travail [32] considère aussi que la migration fait partie de l'interopérabilité et les auteurs du travail [34] partagent cet avis.
La migration est le processus de déplacement d'entités telles que des données, des applications ou des charges de

travail vers un environnement de Cloud Computing.

Il existe divers types de migration dans le contexte du Cloud Computing [38] :

Cloud-to-Cloud: ce type consiste à passer d'une plate-forme de Cloud Computing à une autre. Par exemple, les utilisateurs trouvent qu'un service d'un certain fournisseur a plus d'options (sécurité, coût, etc.) que le service qu'ils utilisent déjà et décident de migrer vers ce nouveau service.

Local-to-Cloud: c'est la migration d'un environnement sur site (local) vers un environnement de Cloud Computing. De nombreuses organisations envisagent de migrer leurs anciens systèmes vers un environnement de Cloud Computing afin de profiter des avantages de ce dernier, tels que la grande élasticité, la disponibilité des services à la demande et le paiement à l'utilisation.

Cloud-Exit: il s'agit du déplacement des données ou des application qui sont sur un environnement de Cloud Computing vers un site ou centre de données local.

II.8 Les types d'interopérabilité

Il existe divers types d'interopérabilité et les chercheurs du domaine sont arrivés à plusieurs classifications. Dans notre travail, nous suivons l'exemple de [39] et nous citons les types classés selon un nombre de critères.

- Par rapport au niveau de déploiement:

  -**L'interopérabilité horizontale**: ce type d'interopérabilité se fait entre des services de Cloud Computing équivalents qui sont sur le même niveau d'abstraction (IaaS, PaaS ou SaaS) [39] [40].
  L'interopérabilité horizontale peut aussi désigner l'interopérabilité entre des services de Cloud Computing appartenant à de différents fournisseurs [41].

  -**L'interopérabilité verticale**: ce type d'interopérabilité désigne le cas de la construction de services de Cloud Computing, opérant sur un certain niveau d'abstraction, sur d'autres services qui opèrent sur des niveaux d'abstraction différents [39] (un SaaS construit sur un PaaS ou un PaaS sur un IaaS) [40].
  Ce terme peut aussi désigner l'interopérabilité entre les services de Cloud Computing du même fournisseur mais auxquels l'utilisateur accède depuis de différents appareils [41].

- Par rapport à l'échange de données [42]:

  -**Interopérabilité syntaxique**: se réfère à la possibilité de communication et de transmission de données entre les systèmes en utilisant des formats de données et des protocoles de communication adéquats.

  -**Interopérabilité sémantique**: désigne l'interprétation automatique et significative des données transmises en utilisant un modèle commun dans le processus de communication dans le but de produire des résultats efficaces.

- Par rapport au mode d'adoption [42]:

  -**Interopérabilité par la conception**: dans ce cas, il existe un document standard pour le développement de services. Les services produits sont donc équivalents puisqu'ils sont tous conformes aux standards en question.

  -**Interopérabilité post-facto**: c'est le cas où une certaine technologie domine le marché. Dans ce cas, les autres entités qui doivent communiquer avec l'entité opérant avec cette technologie, elles doivent utiliser le même format que celui qu'elle utilise.

II.9 Niveaux de l'interopérabilité

La complexité du concept de l'interopérabilité ne se limite pas aux différents types. En effet, l'interopérabilité dans le Cloud Computing prend place selon différents niveaux.

Selon [39], les niveaux de l'interopérabilité dans le Cloud Computing sont les suivants :

-**Niveau commercial**: ce niveau est relatif aux différentes stratégies commerciales imposées au service, à la réglementation du service et au mode d'utilisation du service. L'interopérabilité au niveau commercial se réalise lorsque les conflits concernant ces points sont résolus.

-**Niveau sémantique**: ce niveau concerne le traitement sémantique des requêtes des consommateurs et des réponses à ces requêtes. L'interopérabilité à ce niveau nécessite également la prise en charge des appels aux opérations et du contenu des messages.

-**Niveau d'application / service**: la configuration de l'application doit pouvoir s'adapter à chaque changement de configuration dans l'environnement du Cloud Computing et l'application ne doit nécessiter aucune modification autre

que la recompilation. L'interopérabilité sur ce niveau peut être réalisée à travers l'adaptation d'une programmation indépendante de l'infrastructure.

-**Niveau de gestion**: ce niveau concerne le cas où une application de gestion organise et contrôle les composants appartenant à différents services de Cloud Computing. L'interaction entre les acteurs intervenant dans la gestion est primordiale. On parle ici de gestion d'infrastructure, de VMs et d'application par rapport aux processus de déploiement et de migration.

-**Niveau technique / infrastructure**: ce niveau d'interopérabilité implique la sélection du protocole de communication ou le middleware, le langage et la plate-forme de l'environnement pour une interaction harmonieuse. L'interopérabilité sur ce niveau nécessite également un consensus sur l'encodage des requêtes et réponses.

-**Images / données**: sur ce niveau, il est nécessaire de vérifier les images de machines virtuelles, les applications ou les bases de données et de trouver les bonnes méthodes pour les déployer sur d'autres environnements, sans modifications additionnelles, autant qu'applications empaquetées sous formes de VMs prêtes à l'emploie .

-**Le réseau**: ce niveau implique la prise en charge de l'accès unifié aux ressources et aux applications même au sein des ensembles de systèmes de Cloud Computing. Cela nécessite des normes de contrôle d'accès et d'authentification.

II.10 Les avantages de l'interopérabilité

L'interopérabilité dans les environnements de Cloud Computing permet l'interconnexion et la portabilité, et ces dernières peuvent être bénéfiques aux utilisateurs. Elle présente de nombreux avantages, notamment [31] :

- **Élimination du verrouillage des fournisseurs**: le verrouillage des fournisseurs est l'un des plus gros problèmes du Cloud Computing, en raison de l'utilisation de technologies et de ressources propriétaires par les fournisseurs. La réalisation de l'interopérabilité et de la portabilité peut éliminer le problème de verrouillage des fournisseurs et leur accorder plus de liberté dans leurs choix.
- **Abstraction de l'infrastructure**: le matériel sous-jacent sur lequel l'application s'exécute n'est plus une préoccupation pour les développeurs et gestionnaires d'applications. L'hyperviseur fournit une abstraction de l'infrastructure pour éliminer les barrières de compatibilité.
- **Abstraction entre l'application et les données**: Cette abstraction qu'offre l'interopérabilité assure la portabilité, la modularité et le couplage faible des applications. Il n'est plus nécessaire pour les données et les applications de résider au même endroit.

- **Adaptation et particularisation du Cloud**: l'interopérabilité permet aux entreprises d'adapter et de personnaliser l'environnement de Cloud Computing pour correspondre à leur vision et répondre à leurs besoins.
- **La transparence**: la transparence que l'interopérabilité fourni assure la continuité des activités des utilisateurs en cas de changement de fournisseur de service de Cloud Computing.

II.11 Les concepts liés à l'interopérabilité

L'interopérabilité est liée à un nombre de concepts qui ne lui sont pas forcément éxclusifs. Nous dédions cette section à la présentation de ces concepts.

II.11.1 La découverte de services

La découverte est le processus de collecte de services publiés sur Internet qui correspondent aux besoins des clients [43]. La découverte des services est basée sur la description des services disponibles. La découverte est effectuée en exécutant une analyse détaillée des aspects fonctionnels et non fonctionnels des services. Par conséquent, le meilleur service (le service qui répond aux besoins de l'utilisateur) doit être sélectionné [44].

Il existe deux mécanismes de découverte :

- **La découverte syntaxique:** est généralement basée sur une comparaison entre les mots-clés extraits des demandes des utilisateurs et les mots-clés extraits des descriptions de services (tels que WSDL). Le principal inconvénient de la découverte syntaxique est qu'elle ne prend pas en compte les aspects sémantiques de la requête.

- **La découverte sémantique:** réside dans l'utilisation de différents concepts trouvés dans l'ontologie pour calculer le degré de correspondance entre les mots-clés de la requête et les mots-clés de la description de services [43].

II.11.2 La description de services

Comme la section précédente le précise, la description de services est nécessaire pour pouvoir effectuer leur découverte.

La description de service saisit les caractéristiques fonctionnelles et non fonctionnelles du service dans un format interprétable par la machine.
Les modèles de description de services sont divisés en deux catégories: la description syntaxique et la description sémantique [45].

Il existe plusieurs langages pour décrire les services, notamment le langage WSDL, le langage USDL, le langage OWL-S, etc. Nous allons présenter quelques uns en plus de détails dans ce qui suit.

USDL

Le langage USDL (Unified Service Description Language) permet de décrire les services d'un point de vue opérationnel, technique et commercial.

En ce qui concerne la perspective commerciale, l'USDL permet d'exprimer des caractéristiques définies par l'organisation pour permettre aux consommateurs d'utiliser des services commerciaux.

L'USDL se compose d'un ensemble de modules, tels que le module fonctionnel, le module participants et le module juridique, etc. et chacun de ces modules traite un aspect particulier de la description du service [46].

L'importance du langage USDL se manifeste dans son applicabilité sur la découverte, la comparaison, l'évaluation et la gestion des services. Ceci permet aux demandeurs de services une prise de décision plus éclairée lors de la sélection.
Cependant, l'inconvénient du langage USDL est la médiocrité de sa prise en charge de l'aspect sémantique [47].

WSDL

Web Service Description Language (WSDL) est un document basé sur la syntaxe XML recommandée par le W3C[1] pour décrire les services Web.

Le fichier WSDL comprend les éléments suivants [48] :

Type: cet élément est utilisé pour définir les types de données utilisées.
Message: l'élément Message représente les données échangées.
Opération: cet élément représente la description d'une action , une fonctionnalité, fournie par le service.
PortTypes: représente une description des opérations pouvant être exécutées et des messages impliqués.
Binding: représente une description concrète décrivant le protocole et le format de données utilisés pour les opérations et les messages.
Port: représente un point d'accès, c'est une combinaison d'une adresse réseau et d'une liaison (binding).
Service: représente un ensemble de ports (points d'accès) pour permettre l'accès.

[1] https://www.w3.org/

RDF

Le Resource Description Framework (RDF) est un modèle de représentation et d'échange de données développé par W3C [49].Il fournit une structure simple pour décrire les ressources Web, ou chaque ressource est définie par un URI (Uniform Resource Identifier).

Un graph RDF contient des triplets (Sujet, prédicat, objet) , également appelés (ressource, propriété, valeur) tel que [50] [49]:
Sujet: toute ressource décrite par des expressions RDF est appelée sujet tel qu'une page Web.
Prédicat: représente une propriété ou une relation spécifique utilisée pour décrire une ressource.
Objet: représente la valeur d'une propriété attribuée à une ressource particulière.

La syntaxe d'un document RDF est basée sur le langage XML.

OWL

L'un des langages les plus populaires utilisés pour les ontologies est le langage OWL.
OWL est l'abréviation de Ontology Web Language. C'est en fait une extension de RDFS (RDF Schema), ce qui le rend plus puissant pour exprimer la sémantique.

Le langage OWL se manifeste en trois versions [51]:

OWL Lite : cette version de Owl «...supporte une hiérarchie de classification et des fonctionnalités de contraintes simples comme les contraintes de cardinalité.» C'est la version la plus facile à comprendre et à mettre en œuvre, mais aussi celle qui présente la plus grande restriction d'expressivité.
OWL DL : inclut des restrictions supplémentaires comme la séparation des types. Cette version permet un support de raisonnement très efficace mais, par contre, la compatibilité totale avec RDF n'est plus supportée.
OWL Full : c'est la version la plus expressive d'OWL et offre également une compatibilité syntaxique et sémantique complète avec RDF. Toutefois, elle perd de son efficacité lorsqu'il s'agit du raisonnement dans certains cas.

OWL-S

OWL-S est une ontologie de services qui permet de décrire différents aspects des services Web. Cette ontologie était précédemment connue comme l'ontologie de service DAML (DAML-S) [2].

OWL-S est basée sur le langage d'ontologies OWL qui permet la description des services à travers des classes et des propriétés de manière structurée et compatible avec le Web.

[2]DARPA Agent Markup Language for Services

Le but d'OWL-S est de réaliser la découverte, l'invocation et la composition automatiques de services Web. Un fichier OWL-S est caractérisé par une structure particulière due à la nécessité de définir trois types de connaissances sur les services. Cette structure inclut les éléments suivants [52]:

Service Profile: indique ce que fait le service, les limites de son applicabilité et sa qualité. Cet élément spécifie également les exigences auxquelles le demandeur doit répondre pour pouvoir utiliser le service.

Service Model: permet au client de savoir comment utiliser le service. Cet élément détaille la manière dont un service est sollicité et ce qui se passe lorsqu'il est exécuté. Il est aussi possible d'exploiter cet élément pour des fins d'analyse, de coordination et de surveillance de services.

Service Grounding: cet élement détail la manière d'accéder à un service. En d'autres termes, il indique le protocole de communication, le format du message et d'autres détails tels que le numéro de port utilisé qui permet de contacter le service. Cet élément spécifie aussi des techniques utilisées pour l'échange non-ambiguë de données entre chaque type d'entrée ou de sortie.

XML

Le langage de balisage extensible (Extensible Markup Language) ou simplement XML est un format de texte développé à l'origine par un groupe de travail appelé SGML Editorial Review Board en 1996. Il est caractérisé par sa simplicité et sa flexibilité et est particulièrement apprécié pour son efficacité dans l'échange de données sur le web [53].

Le langage XML a été conçu dans le but d'en faire un langage facilement traitable par les programmes et également lisible par les humains [54].

XSD

XSD, également connu sous le nom de XML Schema Definition, est un langage basé sur XML et dédié à l'expression des contraintes sur les documents XML définis par le W3C. Il peut être utilisé pour une variété de choses comme l'association de types avec des valeurs et pour mettre des restrictions sur les éléments. XSD est également utilisé pour contrôler la qualité des documents dans ce qui est connu sous le nom de validation XSD [55] .

XML Schema permet la définition de nouveaux types de données par l'extension et la restriction de ceux qui existent déjà (integer, string, date, etc) [56].

II.11.3 Les ontologies

Les ontologies servent de vocabulaire commun lorsqu'il est nécessaire de partager des informations dans un domaine.

Dans les ontologies, les concepts d'un domaine et les relations entre eux sont définis de manière à être interopérables avec les machines, c'est-à-dire qu'ils peuvent être lus et compris par la machine.

Le fait que des connaissances communes et une compréhension commune de ces connaissances soient partagées entre les systèmes permet l'extraction et l'agrégation d'informations à partir de ces systèmes et, par conséquent, l'utilisation des données extraites et agrégées pour répondre aux demandes des utilisateurs ou comme entrée dans des applications.

Les ontologies permettent la réutilisation des connaissances du domaine. Certains chercheurs peuvent simplement réutiliser des ontologies existantes pour décrire des concepts qui sont impliqués dans leur domaine au lieu de les créer eux-mêmes.

De même, plusieurs ontologies peuvent être intégrées dans la création d'une nouvelle ontologie plus large, ce qui simplifie la tâche des chercheurs qui n'auront plus qu'à décrire les concepts manquants et leurs relations.

L'objectif de la création d'une ontologie n'est pas seulement la représentation des connaissances du domaine. Les ontologies sont souvent conçues pour fournir des ensembles de données cohérents et bien structurés qui alimenteraient des programmes et des applications [57].

Il existe différentes catégories d'utilisation des ontologies [58] :

- Permettre la communication entre les personnes et les organisations.
- Interopérabilité entre différents systèmes.
- Ingénierie des systèmes.
- Gestion des connaissances.
- Traitement du langage naturel.
- Applications du web sémantique.

II.11.4 L'annotation sémantique

En général, le concept d'annotation fait référence à l'attachement de données à une autre donnée.
Les annotations sémantiques sont divisées en trois catégories:

Les annotations informelles: elles n'utilisent pas de langages formels, ce qui signifie que les annotations ne sont pas visibles par les ordinateurs.

Les annotations formelles: elles sont compréhensibles par la machine. Elles sont basées sur des langages formels.

Les annotations basées sur l'ontologie: elles sont faites tel que les données sont associées aux concepts de l'ontologie [59].

L'un des exemples de l'annotation sémantique est l'annotation des services Web. Ces derniers sont annotés avec des ontologies et ce processus est effectué en liant les éléments de fichier WSDL aux concepts de l'ontologie utilisée [60].

II.11.5 Les critères de qualité de service (QoS)

Les critères de qualité de service (QoS) représentent les aspects non fonctionnels des services. Ce sont des mécanismes d'évaluation pour les services.

Ils peuvent aider les consommateurs des services à choisir des services qui leur convient.
De plus, les critères de qualité de service permettent aux prestataires de services de comparer leurs services avec d'autres services fournis par d'autres prestataires. Les critères de qualité de service sont également utilisés pour détecter les violations des SLA [61].

Il existe plusieurs critères de qualité de service. Nous en mentionnons quelques uns [62] [63]:

Coût: Ce sont les coûts d'accès et d'utilisation des services que les consommateurs doivent payer. Ils dépendent également du nombre de tâches que l'utilisateur du service doit effectuer.

Fiabilité: Elle correspond à la capacité du service à assurer un fonctionnement correct et cohérent tout en respectant les conditions établies dans le SLA.

Disponibilité: Elle représente la quantification de l'accessibilité et l'utilisabilité du service aux utilisateurs. Les

prestataires de services dans le Cloud Computing utilisent souvent des valeurs de pourcentages pour exprimer la disponibilité de leurs services .

Temps de réponse: Correspond au temps écoulé entre l'envoi d'une demande au service et la réception d'une réponse. Il est exprimé en millisecondes ou en secondes.

Réputation: La réputation correspond à la valeur minimum de confiance que l'utilisateur requiert pour accepter un service. Cette valeur est mesurée en fonction des expériences des utilisateurs passées qui avaient fournis une évaluation du service.

II.11.6 Mesures de similarité

La similarité sémantique fait référence à l'évaluation de la correspondance entre deux concepts provenant d'une base de connaissances [64].

Le but de l'application de mesures de similarité sémantique est de déterminer le degré de correspondance entre les termes, qui sont similaires en sens mais pas forcément similaires en terme de syntaxe [65].

On trouve deux catégories de mesures de similarité sémantique basées sur l'ontologie [66]:

Mesures basées sur les chemins :

Dans le cadre de cette méthode, la similarité est calculée en fonction de la distance entre les concepts et leurs positions dans la taxonomie. Elle est totalement basée sur la représentation graphique de l'ontologie.

Mesures basées sur le contenu informationnel (IC) :

L'objectif principal de cette méthode est de surmonter les limites des mesures basées sur les chemins [66]. Elle se base sur la notion du contenu informationnel (IC). La similarité sémantique est calculée en fonction de la quantité d'informations partagées par les concepts concernés [67].

II.11.7 L'invocation des services

L'invocation consiste à établir une communication entre le demandeur et le fournisseur de services [68]. L'invocation du service comprend l'adresse du réseau de service et le mécanisme de communication entre le client et le service sollicité [69]. L'invocation du service est ce qui permet son utilisation.

II.11.8 La composition des services

La composition de services est la capacité de combiner deux ou plusieurs services existants pour créer de nouveaux services en combinant les fonctions des services candidats pour répondre aux besoins spécifiques des utilisateurs [70].

La composition de services est divisée en plusieurs approches, celles-ci sont classées par rapport à la perspective que l'on veut adopter.

- Par rapport à l'approche de sélection [71] :

  Sélection par optimisation locale : elle permet de sélectionner le meilleur service pour chaque tâche spécifique du service composite. Cette approche n'assure pas nécessairement la meilleure qualité de service globale car elle ignore les contraintes de QoS qui concernent plusieurs tâches.

  Sélection par planification globale : elle prend en compte les contraintes de qualité de service affectées aux services combinés autant qu'ensemble (plutôt qu'aux tâches individuelles). Cette approche garantit que le service global (composite) avec la meilleure qualité de service est sélectionné.

- Par rapport à l'arrivée de la requête de l'utilisateur :

  La composition réactive : la composition réactive est également appelée composition dynamique ou composition en-ligne, ce qui fait référence à la sélection en temps réel et à la combinaison de services en fonction des besoins des utilisateurs.
  Une fois que la demande de l'utilisateur arrive, la sélection des services et la liaison de ces services entre eux sont effectuées pour générer un nouveau service.
  Cette composition peut répondre aux besoins des utilisateurs, mais elle est difficile à réaliser en raison des changements dynamiques des contextes des utilisateurs et des services [72].

  La composition proactive : la composition proactive est également appelée composition statique ou composition hors-ligne. Dans ce type de composition, les services à combiner doivent être présélectionnés et liés de

manière fixe. C'est-à-dire qu'ils ne peuvent pas être modifiés.
La composition proactive ne répond pas nécessairement aux exigences des utilisateurs, puisqu'elle est effectuée avant l'arrivée de leurs requêtes. La gestion du flux de données (entrée/sortie) est prétraitée.
Ce type de composition convient aux environnements qui ne sont pas évolutifs et dont les services sont statiques [72].

- Par rapport à la présence d'un médiateur:

L'orchestration : le NIST définit l'orchestration des services de Cloud Computing comme «l'arrangement, la coordination et la gestion de l'infrastructure cloud pour fournir des services répondant aux besoins informatiques et commerciaux.» Pour effectuer l'orchestration, un courtier (broker) est utilisé pour les services d'intermédiation, d'agrégation et d'arbitrage [34].
Afin de réaliser l'orchestration; des aspects tels que la sécurité, les accords de niveau de service(SLA) et les stratégies de prestation de services sont nécessaires. Par conséquent, le courtier doit mettre en œuvre tous les aspects ci-dessus afin que les clients profitent d'une meilleure coopération entre différents services.
L'orchestration est confrontée à certains problèmes comme la défaillance du point unique en cas de panne du courtier ou la médiation des données [29].

La chorégraphie : La chorégraphie selon [73] est la composition des interactions entre plusieurs services ainsi que les protocoles de communications qui entrent en jeu dans cette composition, ses interfaces, ses séquences et sa logique associée.
Dans la chorégraphie les services s'occupent de la prise en charge de la communication avec d'autres services par eux mêmes. Autrement dit, il n'y a pas d'intermédiaires tels qu'un courtier pour assurer la communication est la composition comme dans le cas de l'orchestration.
Pour assurer cette interopérabilité, chaque fournisseur de services doit fournir les conditions nécessaires pour l'intégration de services provenant d'autres prestataires avec son propre service. Notamment, des conditions pour permettre l'échange de données, la migration des processus et des VM. Il leur faut aussi s'assurer que certains aspects de l'interaction comme la sécurité, la prise en charge de la connexion ainsi que la gestion de politiques sont disponibles.
La réalisation de l'interopérabilité dans le cas d'une chorégraphie est moins aisée car il s'agit d'un travail de collaboration basé sur une connexion directe entre les services de Cloud Computing [29].

II.11.9 Accord de niveau de service (SLA)

Nous avons définit ce qu'est un SLA dans le chapitre I, mais nous tenons à reprendre ce concept pour parler de ses aspects dans le cadre de l'interopérabilité dans le Cloud Computing.

En ce qui concerne l'interopérabilité dans le Cloud Computing, les SLAs doivent couvrir quatres éléments importants: l'architecture, le format de modèle, la surveillance et les objectifs SLA.
Voilà à quoi sert chaque aspect [35]:

- **L'architecture** : Une architecture (du SLA) bien conçue est nécessaire pour mesurer et gérer les exigences des SLA des différents services de Cloud Computing. Cela sert à éviter aux utilisateurs la confusion qui peut surgir lorsqu'ils ont affaire à de différents SLA avec des différentes spécifications.

- **Le format de modèle** :Il s'agit de la représentation électronique du SLA pour la gestion automatisée de ce dernier. Le format de modèle doit être lisible par la machine et doit être unique pour garantir l'interopérabilité.

- **La surveillance** : Celle-ci est aussi importante pour les utilisateurs que pour les fournisseurs. Le fournisseur en a besoin pour s'assurer qu'il fournit les ressources comme il en a fait la promesse dans le SLA et l'utilisateur en a besoin pour s'assurer qu'il reçoit bien toutes les onctionnalités du service qui lui ont été promis dans le SLA.

- **Les objectifs SLA** : aussi connus sous l'abréviation SLO, ce sont des mesures de qualité de service que le fournisseur s'engage à respecter. Ils représentent la partie principale du SLA. L'utilisateur se base sur ces objectifs pour sélectionner un fournisseur de services ou pour changer de fournisseur.

II.12 Problèmes liés à l'interopérabilité du Cloud Computing

La réalisation de l'interopérabilité entre les services de Cloud Computing a toujours été confrontée à des défis et des problèmes. Ces problèmes impliquent différents aspects, notamment:

- **Le verrouillage de fournisseurs**

  Le verrouillage des fournisseurs signifie qu'une fois que les clients choisissent un fournisseur de services de Cloud Computing, ils ne peuvent pas changer de fournisseur et même s'ils le peuvent, ça sera une opération très couteuse [74].

  Le verrouillage des fournisseurs est causé par le fait que chaque fournisseur possède sa propre API, son format de description de service et son propre format de SLA [19].

  En d'autres termes, les fournisseurs de services de Cloud Computing ne permettent pas à leurs utilisateurs d'utiliser des services offerts par d'autres fournisseurs de services, ce qui est un obstacle à la réalisation de l'interopérabilité dans les environnements de Cloud Computing [29].

- **La sécurité**

  La sécurité est l'un des principaux obstacles qui empêchent les clients d'utiliser les services de Cloud Computing [32].

  Les mécanismes de sécurité traditionnels tels que l'authentification et l'autorisation d'accès ne sont plus applicables au Cloud Computing à cause des particularités du domaine car on parle d'un environnement à très grande échelle,virtualisé et distribué [75], ce qui signifie que les différents fournisseurs de services utilisent différentes stratégies pour le contrôle de la sécurité des données et divers mécanismes de contrôle d'accès et de confidentialité. Ces différences ont provoqué des incohérences et des incompatibilités [41].

- **L'authentification**

  Actuellement, chaque fournisseur de services de Cloud Computing a ses propres techniques pour prouver les identités des utilisateurs. Les différences entre ces techniques peuvent entraîner des incohérences et des conflits [34].

  En raison de l'absence de normes de contrôle d'accès, les attributs des utilisateurs ne peuvent pas être directement transférés d'un fournisseur à un autre [19].

- **La portabilité**

  L'un des plus grands défis de l'interopérabilité est la portabilité et l'échange de données.

  L'absence de format portable peut conduire à des modifications imprévues des données à transférer vers un nouveau fournisseur [30].

  De plus, la conversion des données du format du système source au format requis par le système cible peut demander beaucoup d'efforts [31].

- **La migration**

  Il y a un manque de standards auxquels les fournisseurs de services dans le Cloud Computing devraient se conformer pour faciliter la migration des données vers et depuis leurs environnements [31].

  D'autre part, il est impossible de migrer une application en cours d'exécution d'une plateforme de Cloud Computing à une autre [76].

- **Les SLAs**

  Le problème du SLA concerne le manque de diversité des SLA et repose sur des modèles statiques, ceci n'est pas pratique dans le contexte des applications qui ont des exigences de qualité de service strictes ou qui nécessitent une évaluation précise des risques avant d'adopter le service de Cloud Computing [31]

  De plus, Il n'y a pas d'outils permettant d'évaluer les accords de niveau de service [39].

  Pour les SLAs, un autre problème se manifeste. Il s'agit du manque de mécanismes de contrôle du SLA entre le client et le fournisseur pour vérifier si l'accord de niveau de service attendu est respecté [19].

- **Stockage**

  Chaque fournisseur de services de Cloud Computing utilise son propre format de stockage de données. La

gestion et l'organisation du stockage sont des problèmes qui doivent être résolus pour garantir la compatibilité et l'interopérabilité entre les différents services de Cloud Computing [35].

- **Les machines virtuelles (VMs)**
  Un élément clé de l'interopérabilité est la transportabilité des machines virtuelles [77]. Les machines virtuelles ne peuvent pas être migrées d'un hyperviseur à un autre sans les arrêter [78].
  De plus, les différences entre les hyperviseurs représentent un obstacle face à la collaboration entre les fournisseurs [29].

II.13 Exigences de l'interopérabilité

Avant toute chose, pour la réalisation de l'interopérabilité dans les environnements de Cloud Computing, les exigences doivent être clairement tracées. Ces exigences sont relatives aux problèmes de l'interopérabilité mentionnés dans la section précédente.

Selon [39], les exigences principales concernent les aspects suivants :

Concernant la programmation :

- Le passage d'un fournisseur à l'autre ne doit pas nécessiter de changements importants dans l'implémentation.
- Une API standard est nécessaire pour permettre aux développeurs de créer des entités réutilisables.
- Il est aussi nécessaire de travailler avec un ensemble d'outils qui permet le fonctionnement des applications basées sur le Cloud Computing et propres à l'entreprise à la fois. Ces outils doivent prendre en charge les programmes existants et les différents fournisseurs.
- L'utilisations d'une ontologie du Cloud Computing et de nouveaux modèles de modélisations et de programmation est également important.

Concernant les applications :

- La portabilité et la capacité d'échange de données entre les applications doivent être assurées.
- La fonctionnalité doit être assurée sur plusieurs services en nuages.
- Les applications déployées selon des modèles de Clouds privés doivent pouvoir obtenir des ressources à partir de Clouds publics selon les besoins.
- Les applications ne doivent pas dépendre de leur localisation.

- La gestion des flux de travail doit être intégrée.

Concernant la surveillance :

- La surveillance et l'audit des services est un facteur indispensable.
- Un suivis des performances est nécessaire pour assurer le respect du SLA. Ce suivis peut nécessiter des benchmark pour l'évaluation des performances.
- Les services doivent être fournis à la demande à travers des technologies de virtualisation, dans un environnement extensible, tout en minimisant les coûts et en maximisant la qualité de services.

Concernant le déploiement :

- Mettre une plateforme commune pour la navigation entre les services et les applications à la disposition des utilisateurs.
- Assurer la possibilité de communication et d'intégration entre les services des différentes plate-formes.
- Assurer la disponibilité des ressources nécessaires à la découverte, à la réservation et à l'invocation des services.
- Assurer le support des divers technologies d'hyperviseurs ainsi que la gestion des VMs.
- Assurer la prise en charge de l'interfonctionnement des environnements de Cloud Computing avec le réseau et optimiser le routage.

Concernant l'authentification et la sécurité :

- La prise en charge des identités et des mécanismes d'authentification unique des utilisateurs à travers plusieurs domaines, fournisseurs et environnements de Cloud Computing.
- Assurer la sécurité à travers l'audit, les mécanismes de confiances pour les services de Cloud Computing et la gestion des autorisations et de l'authentification.
- Prendre en charge plusieurs technologies pour assurer la sécurité sans devoir obliger un fournisseur de services de Cloud Computing à changer ses politiques et mécanismes de sécurité pour rejoindre une fédération.

Concernant le marché :

- Optimiser la facturation à travers la comptabilité et les techniques basées sur les modèles économiques.

- Se fier aux prix concurrentiels du marché pour la tarification spécifiée dans les SLA et former une fédération chargée de fixer les tarifs de location des services selon les fluctuations du marché.
- Assurer la flexibilité des licences et des rapports services/ressources pour maximiser l'efficacité, la rentabilité et l'utilisation.

II.14 La réalisation de l'interopérabilité

Maintenant que la définition des exigences de l'interopérabilité est effectuée, il est possible de passer à la réalisation de l'interopérabilité.

Il est possible de réaliser l'interopérabilité par deux façons. La première étant la création des services de Cloud Computing selon des standards acceptés par la communauté de manière unanime. Cette façon s'applique à la phase de la création du service. La deuxième façon réalise l'interopérabilité pour les services de Cloud Computing déjà existants qui ne sont pas conçus selon des standards quelconques. Elle se fait par le biais d'un médiateur, le courtier.

II.14.1 La standarisation

Il s'agit de l'élaboration de standards pour «...décrire le Cloud, l'interface de communication et le format des données...» pour aboutir à l'interopérabilité entre les environnements de Cloud Computing [77].
En d'autre termes, la standardisation du Cloud Computing vise à «... fournir une approche unifiée pour les services de Cloud computing à travers les différents prestataires...»
La standardisation est bénéfique pour les acteurs du Cloud Computing, utilisateurs finaux soient-ils, ou fournisseurs. C'est parce que cette unification élimine les barrières d'interopérabilité et de portabilité. Il n'y aurait plus besoin d'apporter des modifications aux applications, données et charges de travail pour les déplacer d'un environnement de Cloud Computing à un autre.

Ainsi, l'utilisateur a plus de liberté s'il veut changer de prestataire, et le prestataire a à son tour plus de chances d'attirer plus de consommateurs [79].

De plus, la standardisation encourage l'innovation et la concurrence car elle offre la chance à plus de prestataires de rejoindre le marché ce qui engendre une baisse des prix des services mais n'affecte pas les profits car plus d'utilisateurs se dirigeraient vers le Cloud Computing [80].

Le domaine du Cloud Computing n'est pas standardisé. Ceci revient au fait que c'est un domaine relativement jeune dans le monde informatique [79]. Cependant, il existe divers groupes et organisations qui travaillent pour réaliser la standardisation du Cloud Computing. Ces organisations travaillent sur divers aspects de la standardisation. Certaines visent à développer des architectures et des modèles pour le Cloud Computing, d'autres sont focalisées sur

la sécurité et la surveillance, et il y en a qui ont choisis de traiter l'aspect de communication.

Bien que les efforts de standardisation dans le Cloud Computing sont prometteurs, il est nécessaire pour les organisations de travailler ensemble et de coordonner leurs activités pour éviter la duplication des solutions mais aussi pour avoir de meilleurs résultats qui pousseraient à la standardisation future du Cloud Computing [80].

II.14.2 Le courtage

Lorsqu'il n'est pas possible de suivre la voie de la standardisation comme lorsque le service est déjà conçu et implémenté, la réalisation de l'interopérabilité reste possible grâce aux courtiers.

Nous avons déjà expliqué ce qu'est un courtier dans le chapitre I, mais nous allons quand même y revenir dans ce chapitre pour parler davantage de ses fonctionnalités.

NIST [81] décrit la fonction de courtage comme suit :

- **Intermédiaire** : L'intermédiation est la fourniture de services à valeur ajoutée à travers l'amélioration des services existants. On entend par cette dernière l'amélioration d'une capacité comme le contrôle de la performance, la gestion de la sécurité et la gestion d'accès.
- **Agrégat**: L'agrégation est une combinaison et l'intégration de plusieurs services pour fournir un ou des services composites tout en assurant l'intégration des données et leur circulation sécurisée.
- **Arbitre** : L'arbitrage est semblable à l'agrégation, sauf que la sélection des services dans le cas de l'arbitrage est effectuée de manière flexible et dynamique. Le courtier peut choisir les services qu'il jugent être les meilleurs parmis les services provenant de plusieurs fournisseurs à travers un calcul de score par exemple.

II.15 Conclusion

L'interopérabilité est un concept riche et complexe aux dimensions multiples qui est étroitement lié à d'autres concepts dans le monde informatique.

Dans ce chapitre, nous avons parlé de l'interopérabilité et de ses concepts essentiels. Nous avons commencé par définir l'interopérabilité. Puis nous avons définit des concepts connexes comme la compatibilité, l'intégration, la portabilité et la migration. Après, nous avons présenté les types et les niveaux de l'interopérabilité avant de passer aux concepts liés à l'interopérabilité qui sont pertinents à notre travail comme la description, la découverte, l'invocation et la composition des services. Ensuite nous avons repris le concept du SLA présenté dans le chapitre

précédent pour l'explorer davantage. Puis, nous avons présenté les problèmes et les exigences de l'interopérabilité avant de passer à la réalisation de l'interopérabilité qui implique les concepts de standardisation et de courtage.

Dans le chapitre suivant, nous allons présenter les résultats de notre recherche relative aux solutions existantes à la problématique de l'interopérabilité dans le Cloud Computing en premier lieu suivi des solutions spécifiques aux SaaS. Cette recherche va prendre la forme d'une étude comparative.

Chapitre III

Solutions pour l'interopérabilité dans le Cloud Computing

III.1 Introduction

Les problèmes mentionnés dans le chapitre précédent ont fait l'objet de plusieurs efforts ayant visé la résolution de la problématique de l'interopérabilité. Dans ce chapitre nous allons présenter ces efforts. En premier, nous allons aborder les solutions proposées pour la réalisation de l'interopérabilité dans le Cloud Computing indépendamment du modèle de livraison. Puis, notre étude va être centrée sur les travaux de la littérature ayant proposé des solutions pour la problématique de l'interopérabilité dans les SaaS.

III.2 Étude comparative des solutions existantes pour l'interopérabilité dans le Cloud Computing

Dans cette section nous présentons quelques entités ayant travaillé sur la problématique de l'interopérabilité dans le Cloud Computing ainsi que les solutions qu'elles ont proposé. Ces solutions vont ensuite être assemblées dans un tableau comparatif et serviront de données pour nos statistiques que nous allons analyser et discuter.

III.2.1 Les contributeurs

- **DMTF**

  DMTF, également connu sous le nom de Distributed Management Task Force, est un consortium qui participe à l'effort de standardisation du Cloud Computing.

  Parmi ses travaux, il y a CIMI qui évalue la gestion des ressources dans la couche IaaS et OVF qui est un format qui vise à décrire les dispositifs virtuels d'une manière standard et neutre [19]. Il existe également l'OCSI qui vise à normaliser les interactions de l'environnement du Cloud Computing par le biais de protocoles de gestion des ressources [39]. Ou encore CADF qui définit des normes ouvertes pour l'audit du Cloud Computing [30].

- **Apache Software Foundation**

  L'Apache Software Foundation est une organisation composée de membres individuels et d'industries qui développent conjointement des logiciels d'entreprise libres au profit des utilisateurs du monde entier.

  Apache est également impliqué dans le Cloud Computing, visant à résoudre les problèmes d'interopérabilité en créant des solutions telles que Deltacloud et Jcloud [82].

- **OpenStack**

  OpenStack est une association fondée à l'origine par Rackspace[1] et la NASA[2]. Elle est devenue une communauté open source dans laquelle les développeurs et les techniciens du Cloud Computing travaillent ensemble

[1] https://www.rackspace.com/
[2] https://www.nasa.gov/

à l'échelle mondiale.

Elle produit des solutions pour tous les types de Clouds grâce à une implémentation simple et évolutive et de nombreuses fonctions pour le modèle de services IaaS [31].

- **OASIS**

  OASIS est l'une des organisations de normalisation à but non lucratif qui encourage l'interopérabilité.

  Elle est reconnue comme un profil standard ouvert pour le déploiement, l'approvisionnement, la gestion des identités et l'analyse des risques et des menaces dans le Cloud Computing, et fournit des suggestions pour atténuer les vulnérabilités [83].

- **SNIA**

  La SNIA ou Storage Networking Industry Association est l'un des groupes qui travaillent activement sur les normes du Cloud Computing.

  Sa mission est de « Diriger le secteur du stockage dans le monde entier en développant et en promouvant des architectures, des normes et des services éducatifs neutres qui facilitent la gestion, la circulation et la sécurité efficaces des informations.»[84]

  Parmi ses solutions, on compte le CDMI, une norme portant sur l'interopérabilité et la gestion des données de stockage dans le Cloud Computing [39] [29].

- **OMG**

  L'OMG ou Object Management Group est une organisation active qui vise une normalisation ouverte dans le Cloud Computing pour atteindre l'interopérabilité [39]. Son travail est centré sur les middlewares. L'OMG estime que l'interopérabilité est déterminée à la fois au niveau des données et des services [85].

  Les normes de l'OMG sont «pilotées par les vendeurs, les utilisateurs finaux, les institutions universitaires et les agences gouvernementales.» Elle considère également que l'adoption des normes de spécification comme étant le point de départ de son processus.

  L'OMG utilise ses normes de modélisation comme UML pour garantir la puissance de conception, de maintenance et d'exécution des logiciels et des processus [86].

- **OpenID Foundation**

  La Fondation OpenID (OIDF) est une organisation mondiale de standardisation à but non lucratif.

  Elle est composée de personnes et d'entreprises qui soutiennent et augmentent conjointement le taux d'adoption de la norme OpenID. Cela implique la gestion des droits de propriété intellectuelle et la participation mondiale à la diffusion d'OpenID [87].

- **CCIF**

  CCIF (Cloud Computing Interoperability Forum) est une organisation ouverte à but non lucratif dédiée à l'adoption rapide des services de Cloud Computing.

  Le CCIF vise à créer un écosystème mondial de Cloud Computing dans lequel les organisations peuvent collaborer de manière transparente. Le but du CCIF est d'utiliser des modèles communs, comme les ontologies, pour réaliser la standardisation des interfaces et la description sémantique unifiée des modèles de données [88] [29].

- **IETF**

  L'IETF ou Internet Engineering Task Force est une communauté internationale ouverte dont le travail tourne autour des réseaux et de l'Internet.

  Sa mission est « d'améliorer le fonctionnement de l'Internet en produisant des documents techniques pertinents et de haute qualité qui influencent la manière dont les gens conçoivent, utilisent et gèrent l'Internet.»

  Le travail d'IETF est fait par le biais des groupes de travail [89].

- **IBM**

  IBM ou International Business Machines Corporation est une société multinationale américaine spécialisée dans l'informatique matérielle et logicielle. Les efforts d'IBM sont aussi centrés sur la recherche dans divers domaines comme l'intelligence artificielle et le Cloud Computing [90]. En ce qui concerne le domaine du Cloud Computing, IBM croit en l'effort de standardisation et pousse vers un écosystème ouvert de Clouds [91]. Parmis ses travaux dans le domaine du Cloud Computing, et plus spécifiquement dans l'interopérabilité des services de Cloud Computing, on trouve l'API SimpleCloud qui est un travail collaboratif avec d'autres sociétés présentes dans le domaine [92].

- **La Commission Européenne**

  La Commission Européenne est une branche de l'Union Européenne. La tâche principale de la Commission Européenne est de promouvoir les intérêts généraux de l'Union Européenne (UE) en proposant une législation et en assurant son application, ainsi qu'en mettant en œuvre des politiques et des mesures [93].

- **OGF**

  OGF est une communauté ouverte qui travaille sur le développement de l'informatique distribuée. OGF se concentre sur le Grid Computing, le Cloud Computing et les réseaux. Le travail d'OGF se fait par le biais de forums ouverts et d'événements communautaires qui se concentrent sur l'exploration des tendances, l'optimisation des approches et la documentation des résultats de recherche pour les consolider en normes. Le travail de l'OGF concernant le Cloud Computing se fait principalement au niveau du IaaS et prend la forme

d'API comme l'OCCI [94][39].

- **CSA**

  Le CSA ou Cloud Security Alliance est un groupe qui travaille activement dans l'amélioration de l'interopérabilité dans le domaine du Cloud Computing [34]. Il est l'opérateur du registre de sécurité, de confiance et d'assurance (STAR) du CSA, qui est un programme de certification des fournisseurs de sécurité dans les Clouds offrant des fonctions d'auto-évaluation, d'audit par une tierce partie et de surveillance. La CSA est également chargée de gérer le programme de conseil global de la CSA qui permet aux utilisateurs du Cloud Computing de « travailler avec un réseau de professionnels de la sécurité et d'organisations de confiance qui offrent des services professionnels qualifiés basés sur les meilleures pratiques de la CSA. »

  Les recherches de la CSA sont caractérisées par la neutralité par rapport aux fournisseurs, l'agilité et l'intégrité des résultats [95].

- **Red Hat**

  Red Hat est une organisation de développement logiciel open source. Elle encourage l'effort collectif et l'innovation pour produire de meilleurs logiciels [96].

  Red Hat insiste sur la nécessité d'une approche de Cloud Computing ouverte qui permet d'atteindre la portabilité entre les Clouds et d'éviter le verrouillage des fournisseurs [97].

  Elle a créé le projet Aeolus pour fournir des outils de création, de gestion et de déploiement de services de Cloud Computing [98].

- **enStratus**

  enStratus est une organisation qui prend en charge la gestion de l'infrastructure Cloud publique, privée et hybride. Elle se concentre sur l'utilisation d'un ou plusieurs services de Cloud Computing (infrastructure, plate-forme ou application) via une plate-forme de gestion unifiée [99].

III.2.2 Les solutions proposées:

- **CIMI**

  CIMI (Cloud Infrastructure Management Interface) est un effort de standardisation réalisé par DMTF spécifiquement pour le domaine IaaS. Il est basé sur le protocole REST [3] et prend en charge les formats JSON et XML.

  Les principales fonctions de CIMI sont la gestion du déploiement, la gestion des machines virtuelles et d'autres fonctions telles que la surveillance et la sécurité [19].

  CIMI permet aux utilisateurs de créer des solutions basées sur le Cloud Computing et garantit que si la solution

[3] Representational State Transfer

est déplacée vers un autre fournisseur IaaS, le processus de gouvernance ne sera pas compromis [85].

- **CADF**

  CADF (Cloud Audit Data Federation) est une norme d'audit de services de Cloud Computing ouverte développée par la DMTF [30].

  L'objectif principal de CADF est de résoudre les incohérences et les problèmes d'incompatibilité, et de fournir une gestion des politiques de sécurité et une surveillance des performances pour garantir que les clients reçoivent des services de qualité [100].

- **OVF**

  OVF est une norme ouverte de la DMTF utilisée pour la description, le déploiement et la distribution de dispositifs virtuels sur différentes plateformes. OVF permet donc le déploiement de dispositifs virtuels chez n'importe quel fournisseur de Cloud Computing, ce qui la rend neutre sur le plan commercial. OVF supporte l'interopérabilité des VMs au niveau IaaS.

  OVF offre un packaging standard sous la forme d'un format commun de description des machines virtuelles qui permet aux dispositifs de fonctionner sur plusieurs machines virtuelles. Ce format est ouvert, portable, efficace et flexible et il fournit également une base pour la migration entre les machines virtuelles sous différents hyperviseurs [29] [35].

- **JClouds**

  JClouds, également connu sous le nom de Java Multicloud Toolkit [101], est une API ouverte écrite en Java. Elle offre des abstractions qui permettent la portabilité des applications et prend en charge divers fournisseurs de services de Cloud Computing et piles de logiciels comme AWS d'Amazon ou Azure de Microsoft [40].

  JCloud offre un framework qui facilite l'interaction avec les ressources de stockage et de calcul de divers Clouds [39].

- **Libcloud**

  Libcloud est une API Python standard open-source développée en 2009.

  Elle peut gérer les différentes API des fournisseurs de services de Cloud Computing dans le domaine IaaS et vise à permettre aux développeurs de créer des services qui peuvent fonctionner sur différentes plates-formes. Libcloud prend en charge les fonctions de base de gestion des services de Cloud Computing, telles que la création, le redémarrage et la destruction d'instances. Elle permet de résoudre le problème de verrouillage des fournisseurs [102].

- **δ-Cloud**

  δ-Cloud ou DeltaCloud est une API ouverte qui permet de se connecter à plusieurs fournisseurs de services de Cloud Computing en faisant abstraction des différences entre eux [39]. L'API est basée sur le protocole REST et est écrite en Ruby [40].

  δ-Cloud est utilisée pour développer des applications interopérables sur des plateformes de Cloud Computing existantes en utilisant les abstractions mentionnées ci-dessus, indépendamment des fournisseurs de services et sans avoir besoin de couches intermédiaires [35].

- **Nova**

  Nova ou Openstack Compute est un projet d'OpenStack qui peut mettre à disposition des serveurs virtuels selon les besoins. Nova est conçue pour fonctionner horizontalement sur du matériel standard sans nécessiter de matériel ou de logiciel propriétaire et peut être intégrée aux systèmes existants [103].

- **Swift**

  OpenStack Object Storage aussi appelé Swift est un logiciel de stockage dans le Cloud qui peut stocker et récupérer de grandes quantités de données à l'aide d'APIs simples.

  Il vise à développer et à améliorer la durabilité, la disponibilité et la synchronisation de toutes les données. Swift est parfait pour stocker des données non structurées qui peuvent se développer de manière illimitée [103].

- **Glance**

  OpenStack Image Service (Glance) est un projet d'OpenStack qui fournit aux utilisateurs la fonction de découverte, d'enregistrement et de restauration des images de machines virtuelles. Les images de VMs fournies par Glance peuvent être stockées dans de différents endroits [103].

- **IDCloud**

  IDCloud ou OASIS Identity in the Cloud est un comité qui s'engage à résoudre les problèmes de sécurité liés à la gestion des identités dans le Cloud Computing [40].

  Il effectue une analyse des risques et des menaces pour identifier les lacunes dans les standards de gestion des identités existantes, puis les corriger. Le comité fait cela à travers des cas d'utilisation et des recommandations [83].

- **TOSCA**

  Le comité technique de TOSCA (Topology and Orchestration Specification for Cloud Applications) est l'un des comités les plus importants et les plus dynamiques de l'histoire d'OASIS, composé de plus de 40

fournisseurs et organismes de recherche .

L'objectif ultime de TOSCA est d'améliorer la portabilité, l'interopérabilité et la gouvernance du Cloud Computing, c'est-à-dire de promouvoir le déplacement transparent des charges de travail entre les infrastructures. TOSCA décrit les services de Cloud Computing (infrastructure, plates-formes et applications) en décrivant tous les composants de service et les relations entre eux [83][85].

- **CDMI**

  CDMI, ou Cloud Data Management Interface, est une interface standard développée par le SNIA.

  Elle permet l'interopérabilité des nuages de stockage et permet la découverte, l'auto-provisionnement, l'administration et l'accès aux services de stockage de manière standardisée par le biais d'une API [29] [85].

  Le CDMI permet également la gestion des données tout en spécifiant comment elles sont créées, récupérées, mises à jour et supprimées du Cloud par le biais d'une API [39]. Le CDMI cible principalement la migration des données [85].

- **SoaML**

  SoaML ou Service oriented architecture Modeling Language est une spécification normalisée par l'OMG.

  Elle présente un métamodèle et un profil UML pouvant être utilisés dans la spécification et la conception de services dans le cadre de l'architecture orientée services [104].

  SoaML est considéré comme un langage de description de services et de modélisation [85].

- **OpenID**

  OpenID est une norme ouverte par l'OpenID Foundation qui permet aux utilisateurs de s'authentifier de manière décentralisée.

  Cela est faisable en créant un compte auprès d'un fournisseur d'identité OpenID puis en s'identifiant par le biais de ce compte dans les ressources web et les multiples fournisseurs qui supportent l'OpenID [74].

  En d'autres termes, OpenID permet « l'identification de tiers et la fonctionnalité de gestion d'accès » à travers de multiples services de Cloud Computing [30].

- **UCI**

  Unified Cloud Interface (UCI) est un projet ouvert et standardisé proposé par la CCIF qui vise à développer un courtier dans le Cloud pour l'unification de différentes APIs du Cloud Computing [39].

  Le concept sur lequel repose le projet est de « créer une couche d'abstraction qui soit agnostique à toute API, plate-forme ou infrastructure de Cloud computing. »

  En pratique, l'interface de l'UCI est censée simplifier l'intégration et la gestion [29]. L'UCI cible les niveaux

IaaS et PaaS et suggère l'utilisation des technologies du web sémantique [32].

- **OAuth**

  OAuth est un protocole ouvert de l'IETF qui permet l'accès aux ressources du serveur par les clients au nom du propriétaire de la ressource.

  Il permet ainsi aux utilisateurs d'accorder l'accès à leurs ressources de serveur à des clients tiers sans avoir à divulguer leurs identifiants [74].

  Oauth permet «l'accès unique à plusieurs Cloud.» [39], également connu sous le nom d'authentification multiple [19], et offre des fonctions de gestion d'accès [30].

- **SimpleCloud**

  SimpleCloud est une API écrite en PHP [40] fournie par IBM. C'est un effort conjoint de plusieurs fournisseurs de services de Cloud Computing pour créer une solution interopérable pouvant être utilisée avec de nombreuses offres de services de Cloud Computing.

  L'effort de Simple Cloud est axé sur l'interopérabilité du code. D'ailleurs, en ce qui concerne cette API, « L'objectif ultime est que le code écrit pour fonctionner avec un service de nuage fonctionne avec tous les services de nuage similaires. »

  Des efforts continus sont fournis pour prendre en charge cette API dans d'autres langages, tels que Java, python et perl [105].

- **mOSAIC**

  mOSAIC est un projet développé par la Commission Européenne ayant pour but de fournir une API open source pour prendre en charge les applications qui utilisent plusieurs Clouds. L'objectif initial de mOSAIC est de réaliser la portabilité et l'interopérabilité de ces applications.

  mOSAIC résout également dans une certaine mesure d'autres défis, notamment la gestion et la sécurité [40] [76].

- **OCCI**

  OCCI (Open Cloud Computing Interface) est une API et protocol ouverts initialement conçus pour des fins de gestion à distance des Clouds de type IaaS.

  Au début, le travail de cette API portait sur des tâches, telles que le déploiement et la surveillance, afin de réaliser l'interopérabilité [31].

  Cependant, OCCI a évolué et ces nouvelles capacités lui permettent d'éliminer les barrières d'interopérabilité liées aux mécanismes d'accès, aux réseaux, à la sécurité et aux accords de niveau de service dans le Cloud Computing [35].Son évolution inclut également la prise en charge d'autres modèles, notamment PaaS et SaaS [106].

- **CloudAudit**

  Le CloudAudit ou API automatisée d'audit, d'affirmation, d'évaluation (assessment) et d'assurance (A6) est une API ouverte qui fait partie du CSA depuis 2010 et qui permet aux fournisseurs du Cloud Computing et aux consommateurs autorisés de mieux contrôler l'audit, l'évaluation des affirmations et l'assurance de leur environnement [74].

- **Aeolus**

  Aeolus est un logiciel ouvert présenté par Red Hat qui permet aux utilisateurs de gérer les Clouds. Il est écrit en Ruby et conçu pour les systèmes Linux.

  Aeolus utilise l'API δ-Cloud pour permettre aux utilisateurs de faire un choix entre les nuages privés, publiques ou hybrides [40].

  Le groupe Aeolus considère que sa mission est essentiellement de fournir des outils qui permettent « la construction, la gestion et la surveillance flexibles de déploiements multi-instances à travers les Clouds. »[107]

- **Dasein**

  Dasein Cloud est une API ouverte écrite en Java sous la licence Apache V2.0 qui comprend des abstractions relatives au stockage, au calcul et au réseau [39]. La plupart des services IaaS et certains services PaaS peuvent être modélisés selon le modèle d'abstraction fourni par Dasein Cloud.

  Dasein Cloud permet aux utilisateurs d'écrire leur application dans le modèle fourni et celle-ci sera traduite dans un modèle de fournisseur sous-jacent. Dasein offre une couche de métadonnées qui permet à l'application d'explorer les capacités du Cloud avec lequel elle fonctionne de manière dynamique [108].

III.2.3 Synthèse des travaux

Le tableau suivant résume les contributeurs, les solutions qu'ils proposent et les domaines de contribution, qui ont été présentés dans la section précédente. Nous signalons que ces informations sont relatives aux travaux que nous avons étudiés dans notre recherche.

Contributeur	Lien de contribueur	Solution proposée	Niveau d'abstraction	Domaine de contribution
DMTF	https://www.dmtf.org/	CIMI [109]	-IaaS	-Déploiement -Gestion des VMs -Surveillance -Sécurité
		CADF [100]	-SaaS -PaaS -IaaS	-Surveillance -Sécurité
		OVF [110]	-IaaS	-Gestion des VMs
Apache	http://www.apache.org/	JClouds [101]	-IaaS	-Portabilité
		Libcloud [111]	-IaaS	-Interopérabilité à la création
		δ-Cloud [112]	-IaaS	-Interopérabilité à la création
OpenStack	https://www.openstack.org/	Nova [113]	-IaaS	-Gestion des VMs
		Swift [113]	-IaaS	-Gestion des données
		Glance [113]	-IaaS	-Gestion des VMs
OASIS	https://www.oasis-open.org/	IDCloud [83]	-SaaS -PaaS -IaaS	-Sécurité -Gestion d'accès
		TOSCA [114]	-SaaS -PaaS -IaaS	-Description des services
SNIA	https://www.snia.org/	CDMI [115]	-IaaS	-Gestion des données

OMG	https://www.omg.org/	SoaML [116]	-SaaS -PaaS -IaaS	-Description des services -Conception des services
OpenID foundation	https://openid.net/foundation/	OpenID [117]	-SaaS	-Gestion d'accès
CCIF	https://groups.google.com/forum/#!forum/cloudforum	UCI [118]	-PaaS -IaaS	-Description des services
IETF	https://www.ietf.org/	OAuth [119]	-SaaS -PaaS -IaaS	-Gestion d'accès
IBM	https://www.ibm.com/dz-fr	SimpleCloud [120]	-IaaS	-Portabilité
Commission européenne	https://cordis.europa.eu/fr	mOSAIC [121]	-SaaS -PaaS -IaaS	-Sécurité -Portabilité
OGF	https://www.ogf.org/ogf/doku.php	OCCI [106]	-SaaS -PaaS -IaaS	-SLA -Sécurité -Déploiement
CSA	https://cloudsecurityalliance.org/	CloudAudit [122]	-SaaS -PaaS -IaaS	-Sécurité -L'audit
Red Hat	https://www.redhat.com/	Aeolus [123]	-PaaS -IaaS	-Déploiement -Surveillance

enStratus	https://groups.google.com/forum/#!forum/dasein-cloud	Dasein Cloud [108]	-PaaS -IaaS	-Conception des services

Tableau III.1: Synthèse des solutions d'interopérabilité du Cloud Computing

Sur la base du tableau III.1 qui présente un résumé des solutions d'interopérabilité du Cloud Computing que nous avons étudié, nous avons réalisé des statistiques concernant les modèles de livraison des services que ces solutions traitent ainsi que leur domaine de contribution. Les résultats sont montrés dans la figure III.1 et la figure III.2.

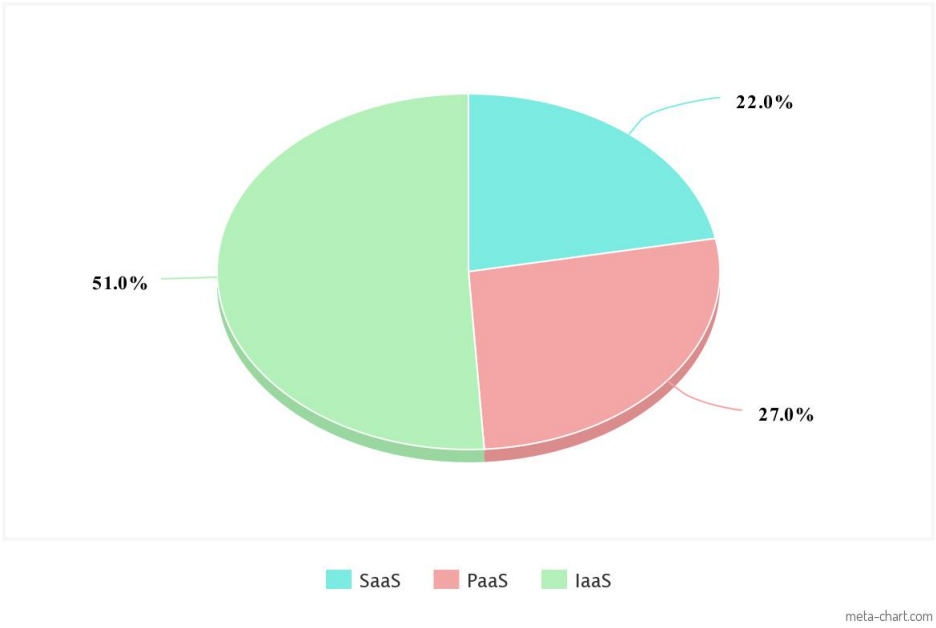


Figure III.1: Statistiques des solutions d'interopérabilité par rapport aux modèles de livraison

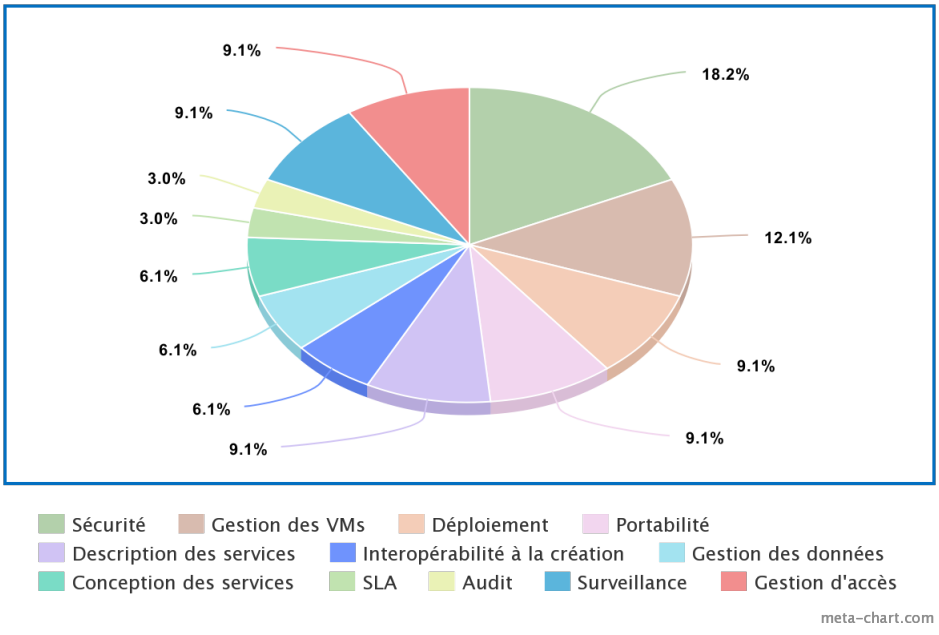


Figure III.2: Statistiques des solutions d'interopérabilité par rapport au domaine de contribution

III.2.4 Discussion

Comme le montre le tableau III.1, nous constatons que la plupart des solutions proposées sont axées sur le niveau IaaS. En d'autres termes, le type d'interopérabilité le plus traité concerne les services de type IaaS. Le traitement de l'interopérabilité des services de la couche PaaS est moins considérable à celui des IaaS.

Les applications SaaS représentent le plus grand défi d'interopérabilité. Il y a très peu de solutions pour ces applications et nous constatons un manque d'efforts sur ce niveau.

En outre, chaque solution proposée aborde un ou plusieurs problèmes liés à l'interopérabilité du Cloud Computing, mais aucune solution n'a encore abordé tous les problèmes. Cela signifie que de nombreuses approches différentes et solutions possibles sont publiées, mais qu'il n'existe pas encore de standard accepté. Cela revient aux barrières de standardisation mentionnées dans le chapitre précédent, notamment le manque de coordination entre les entités travaillant sur les solutions et la jeunesse du domaine du Cloud Computing.

Dans la section suivante, nous allons présenter les solutions proposées au problème de l'interopérabilité qui sont spécifiques aux SaaS.

III.3 Étude comparative des solutions existantes pour l'interopérabilité au niveau SaaS

Dans la section précédente, nous avons fait une étude concernant les solutions existantes à la problématique de l'interopérabilité dans le Cloud Computing à travers ses trois modèles de livraison. Cette étude comparative nous a permis de déterminer que le modèle de livraison SaaS connait un manque considérable d'efforts et de solutions. Cependant, malgré ce manque, il existe quand même des solutions proposées et plusieurs chercheurs ont contribué à la réalisation de l'interopérabilité au niveau des SaaS.

Dans cette section, nous présenterons certains des travaux de la littérature portant sur l'interopérabilité dans les SaaS que nous avons étudié. Nous avons également simulé les solutions proposées dans certains de ces travaux pour avoir une meilleure compréhension de leurs fonctionnements.

III.3.1 Travaux étudiés

Reza et al. ont développé un Framework dans [124] qui peut réaliser l'interopérabilité sémantique des systèmes SaaS dans un environnement de Cloud Computing. Cette solution est considérée comme un middleware prenant en charge l'interaction entre les systèmes SaaS. Elle peut également être utilisée pour connecter des systèmes SaaS les uns aux autres sans communiquer directement entre eux. Elle permet la découverte et l'invocation dynamique des

services.

Afin d'atteindre leurs objectifs, dans le processus de mise en œuvre de ce framework, les auteurs ont utilisé l'outil protégé [4] pour créer une ontologie complète qui contient les concepts de base de la description de service, identifiant ainsi les acteurs de l'interopérabilité (fournisseurs de SaaS, courtier et les consommateurs de SaaS), et définissant les composants d'interopérabilité de ces acteurs.

Ghazouani et al. dans [61] ont défini une ontologie basée sur WSMO [5] pour décrire tous les types de services de Cloud Computing (SaaS, PaaS, IaaS) et leurs caractéristiques, telles que les modèles de déploiement, les contraintes environnementales, les critères de qualité de service (QoS) et les parties prenantes. La description couvre les aspects techniques, opérationnels, commerciaux et sémantiques.

Ils ont adopté le langage USDL en raison de sa capacité expressive, mais comme celui-ci ne couvre pas l'aspect sémantique, certaines améliorations ont été apportées en introduisant l'aspect sémantique à travers une ontologie du Cloud Computing.

Afin de tester l'applicabilité de la description de services de Cloud Computing proposée, ils ont décrit trois services qui sont sur de différentes couches d'abstraction (IaaS, PaaS, SaaS) et qui proviennent de différents prestataires via cette ontologie.

Dans [125] Ghazouani et al. ont proposé une solution qui traite le problème de composition horizontale et verticale des services de Cloud Computing. Ils ont utilisé une description de services (mentionné ci-dessus) qui comprend tous les aspects techniques, opérationnels, commerciaux et sémantiques des services et qui couvre différents types de services (SaaS, PaaS, IaaS) afin de garantir une haute interopérabilité entre les services provenant de différents Clouds et pour permettre une sélection et une composition automatiques des services. Cette description est sous forme d'une ontologie dont la création a été détaillé dans le travail [61].

Afin d'enrichir cette description et de répondre aux besoins des utilisateurs, un raisonnement basé sur des règles a été inclus dans le travail. Les auteurs ont utilisé le langage SWRL [6] pour modéliser les contraintes de qualité de service et les contraintes liées au Cloud.

Les auteurs ont également proposé deux algorithmes: "Cloud combination sélection" et "Composition plans generation". Le premier algorithme vise à sélectionner une combinaison de Clouds contenant des services répondant aux besoins des utilisateurs. La sélection de ces services se fait à l'aide d'une fonction de similarité.

Le deuxième algorithme combine les services de Cloud Computing candidats et génère des plans de composition possibles qui peuvent répondre à la requête des utilisateurs qui est exprimée sous forme de concepts ontologiques. Les auteurs ont proposé une équation pour calculer le score de chaque plan de composition afin de choisir celui qui

[4] https://protege.stanford.edu/

[5] WSMO (Web Service Modeling Ontology) est une ontologie qui sert à décrire divers aspects liés aux services Web sémantiques.

[6] Le SWRL (Semantic Web Rule Language) est un langage qui permet l'intégration des règles d'inférence dans les ontologies.

est le plus optimal.

Zeng et al. [126] présentent une approche qui repose sur le stockage des données pertinentes dans des tables de base de données relationnelles.

Les auteurs ont présenté un algorithme appelé SMA qui fait correspondre les paramètres d'entrée/sortie des services obtenus à partir de la description des services Web tout en considérant leur similarité sémantique en utilisant WordNet [7]. Le degré de correspondance de chaque paire de services est ensuite stocké dans une table et utilisé par la suite pour réaliser un graphe qui sert à trouver toutes les compositions de services possibles. L'algorithme proposé dans ce travail, Fast-EP, est l'algorithme utilisé pour la composition de services de Cloud Computing. Cet algorithme a été optimisé dans une version plus efficace appelée FastB+-EP et il est utilisé pour obtenir tous les chemins possibles dans un court laps de temps. Zeng et al. ont utilisé les critères de qualité de service (QoS) pour classer les services obtenus à partir des résultats de la recherche.

Dans [23], Kurdi et al. utilisent une approche d'optimisation combinatoire pour proposer un algorithme appelé COM2 pour la composition des services dans les nuages dans un environnement multi-cloud. L'algorithme présenté donne la priorité aux Clouds contenant le nombre maximum de services après avoir supposé que les Clouds sont triés par ordre décroissant en fonction du nombre de services qu'ils contiennent. Ceci est fait pour s'assurer que la demande de service est satisfaite avec des services provenant d'un nombre minimum de Clouds afin de réduire les coûts de communication et les charges financières. COM2 comprend un combinateur de Clouds qui sélectionne une combinaison adéquate de Clouds à partir de l'environnement multi-clouds, et un compositeur de services qui génère une composition de services qui peut satisfaire la demande de l'utilisateur à partir de la combinaison de Clouds sélectionnée.

Mezni et al. ont proposé une solution de composition de services dans un environnement multi-cloud selon une approche basée sur l'analyse formelle des concepts (FCA) dans le travail détaillé dans [127]. Ils modélisent l'environnement multi-cloud sous forme d'un réseau (treillis) de concepts formels qui détaillent les Clouds, les services qui y sont hébergés et les fournisseurs qui offerent ces services. L'objectif principal des auteurs est de fournir une solution qui peut retourner la meilleure composition de services possible tout en réduisant les coûts de communication, le temps d'exécution et le nombre de Clouds.

Afin de réaliser l'interopérabilité dans le Cloud Computing et de faciliter la collaboration entre les différents

[7]WordNet est une base de données lexicale.Elle vise à classer et mettre en relation le contenu sémantique et lexical de la langue anglaise.

fournisseurs, une solution basée sur une société d'agent a été proposée dans [6]. L'objectif principal de cette solution est de traiter les problèmes de portabilité et d'adaptation des données échangées entre les différents services de Cloud Computing de type SaaS.

Cette solution est basée sur le paradigme des agents, tel que l'agent descripteur qui décrit les services fournis, l'agent mobile qui achemine les données d'un Cloud à un autre, l'agent adaptateur qui est utilisé pour résoudre l'hétérogénéité syntaxique que les données peuvent rencontrer, et l'agent coordinateur qui s'occupe de l'intermédiation entre les agents et gère également l'invocation des services par les entreprises.

III.3.2 Synthèse des travaux du niveau SaaS

Le tableau suivant présente une synthèse sur les solutions proposées sur le niveau SaaS que nous avons étudié et qui ont été présentées dans la section précédente. Le tableau présente une comparaison entre les solutions selon un nombre de critères que nous avons choisi.

Solution	Description	Composition	QoS	Environnement	Aspect sémantique	Requête utilisateur	Adaptation de données
[124]	Ontologie	Pas Traitée	Sélection selon les QoS	Multi-cloud	Traité	Pas Traitée	Pas Traitée
[61]	Ontologie	Pas Traitée	Pas Traité	Pas Traité	Traité	Pas Traitée	Pas Traitée
[125]	Ontologie	Composition réactive	Sélection selon les QoS	Multi-cloud	Traité	Pas Traitée	Pas Traitée
[126]	Tables d'une BD relationnelle	Composition proactive	Sélection selon les QoS	Pas mentionné	Traité	Traitée	Pas Traitée
[23]	Pas Traitée	Composition réactive	Pas Traité	Multi-cloud	Pas Traité	Pas Traitée	Pas Traitée
[127]	Pas Traitée	Composition réactive	Pas Traité	Multi-cloud	Pas Traité	Pas Traitée	Pas Traitée
[6]	Pas Traitée	Pas Traitée	Pas Traité	Multi-cloud	Pas Traité	Pas Traitée	Traitée

Tableau III.2: Synthèse des solutions d'interopérabilité au niveau SaaS

A partir des résultats de notre étude comparative présentés dans le tableau ci-dessus, nous avons réalisé des statistiques sous forme d'un diagramme en barres dans la figure III.3.

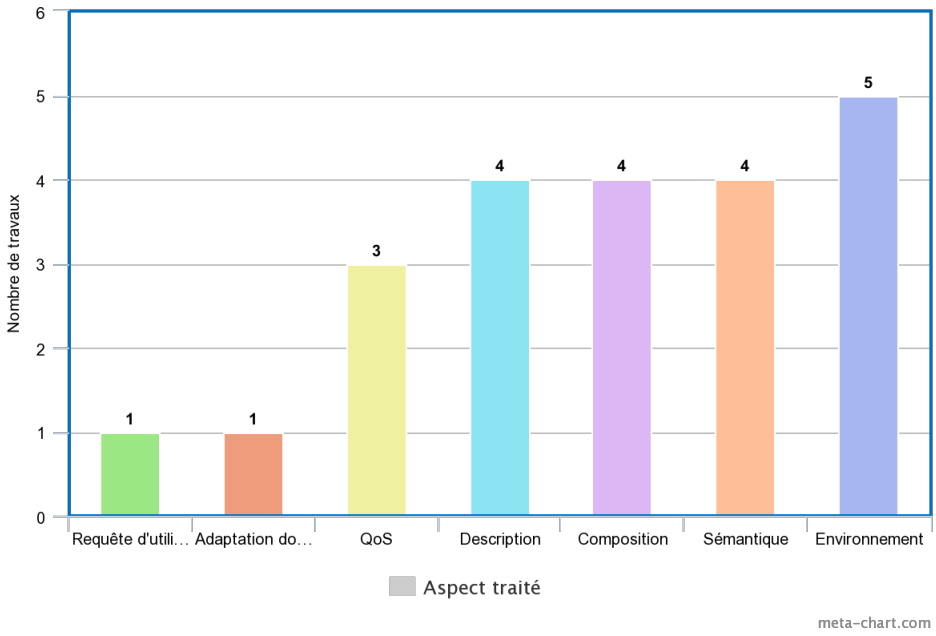


Figure III.3: Statistiques des solutions d'interopérabilité au niveau SaaS

III.3.3 Discussion

Notre étude comparative des œuvres de la littérature traitant de l'interopérabilité dans les SaaS que nous avons sélectionné nous a permis de construire le tableau III.2 et le diagramme de la figure III.3. Ces éléments nous ont permis de tirer un certain nombre de constats.

Tout d'abord, nous nottons que la vaste majorité des travaux que nous avons parcourus traitent l'interopérabilité dans des environnements multi-clouds. Puis, nous faisons remarquer que la problématique traitée dans la plupart des travaux était la composition des SaaS et qu'il s'agissait majoritairement d'une composition réactive.

Par manque de standard de description, les travaux étudiés ont choisi de décrire les SaaS traités dans leurs solutions à travers leurs propres modèles. Il y'en avait qui étaient basés sur les ontologies, par exemple, ceci leur a permis de prendre en charge l'aspect sémantique dans leurs solutions.

Dans les travaux étudiés, plusieurs solutions se sont basées sur les critères de qualité de service (QoS) pour effectuer les traitements de sélection et de composition de SaaS pour répondre à la requête de l'utilisateur. Cependant, un seul travail a pris en charge l'adaptation des données entre les services composés. De plus, nous notons que la requête en question n'a subi des prétraitements que dans l'un des travaux.

Bien que les aspects qui ont servis de critères de comparaison dans le tableau III.2 ont tous été traités dans un

travail ou l'autre, aucun travail ne les a tous pris en charge à la fois. Ceci reflète le manque d'efforts et de solutions proposés en ce qui concerne l'interopérabilité des SaaS et notamment la prise en charge de l'aspect sémantique et de l'adaptation des données.

III.4 Conclusion

Bien que la problématique de l'interopérabilité représente un obstacle majeur qui empêche l'épanouissement du domaine du Cloud Computing et malgré les efforts de recherches, elle reste un défi complexe et difficile à surmonter. Les solutions existantes sont partielles et incomplètes par faute de manque flagrant de standardisation et de divergence des parties entrant en jeu, notamment les organisations de recherche qui travaillent de manière détachée, et les fournisseurs de services qui favorisent le verrouillage.

Dans ce chapitre, nous avons présenté les résultats de notre recherche concernant les œuvres de la littérature dans la problématique de l'interopérabilité. Notre travail était structuré en deux sections tel que la première contenait une étude comparative des solutions proposées à la problématique de l'interopérabilité dans le Cloud Computing selon les modèles de livraison et les domaines de contribution, et la deuxième section présentait les résultats de notre étude comparative relative à la problématique de l'interopérabilité dans les SaaS.

Dans le chapitre suivant, nous allons présenter la modélisation de notre solution proposée. Notre conception sera expliquée en détail à travers une analyse des besoins, une architecture du système et des exemples explicatifs relatifs aux fonctions de base de notre solution.

Chapitre IV

Modélisation de la solution

IV.1 Introduction

Dans le chapitre précédent, nous avons réalisé une étude comparative de l'interopérabilité dans le Cloud Computing avant de faire une étude centrée sur les SaaS en particulier.

Dans ce chapitre, nous présenterons notre solution, qui est un service de courtage gérant l'interopérabilité au niveau sémantique dans un environnement de services de Cloud Computing de type SaaS. Nous signalons que notre travail se fait dans un environnement multi-cloud et concerne les Clouds publics.

Le courtier proposé traite des aspects tels que la description des services qui y sont publiés dans un format commun, la découverte des services selon leurs fonctions, la sélection des services basée sur les critères de qualité de service (QoS) et la composition de services en cas de besoin.

Nous allons commencer par parler de notre démarche de travail. Puis, nous aurons une section dédiée à l'analyse des besoins où nous détaillons les acteurs et chacune de leurs interactions avec le système. Ensuite, nous allons présenter notre solution proposée en montrant l'architecture de notre système et en détaillant chaque fonctionnalité essentielle à l'aide de pseudo-algorithmes et d'exemples explicatifs.

IV.2 Démarche de travail

Afin de réaliser notre projet nous avons adopté la démarche XP ou Extreme Programming [1].

L'Extreme Programming est un processus agile [2] qui a été utilisé pour la première fois sur un projet en 1996. Il s'agit d'un processus de programmation très populaire qui vise à satisfaire le client en lui fournissant les logiciels dont il a besoin au fur et à mesure. Il est également très utile aux développeurs pour s'adapter aux changements des exigences du client, même s'ils sont en avance dans le cycle de vie.

L'Extreme Programming repose largement sur le travail d'équipe car tous les membres de l'équipe sont considérés comme des participants égaux qui travaillent de manière très productive et efficace.

L'Extreme Programming applique cinq principes lorsqu'il s'agit de travailler sur un projet :

- **La communication** : il y a une communication constante entre les programmeurs et les clients ou d'autres programmeurs.
- **La simplicité** : la conception est maintenue aussi simple et propre que possible.

[1] http://www.extremeprogramming.org/

[2] Le processus agile est une méthode itérative et collaborative qui prend en compte les besoins initiaux du client. Les méthodes agiles sont basées sur un cycle de développement centré sur le client. Les clients interviennent dans la réalisation du début à la fin du projet.

- **Le feedback** : le feedback est acquis en effectuant des tests sur le logiciel depuis le début et le système est livré au client très tôt afin d'apporter les modifications nécessaires comme il le suggère.
- **Le respect** : le respect est établi et approfondi pour la contribution de chaque membre de l'équipe avec chaque succès obtenu par l'équipe.
- **Le courage** : on attend des programmeurs de l'XP qu'ils s'adaptent aux changements d'exigences et de technologie avec courage.

La méthode XP applique des règles simples mais très bénéfiques :

- **Planification** : le planning est décidé et le projet est divisé en itérations. L'équipe est censée faire de fréquentes petites mises à jour au fur et à mesure de l'avancement du projet.
- **Gestion** : un espace de travail ouvert est fourni et un rythme durable est établi. Des réunions de type standup ont lieu chaque matin et les membres de l'équipe sont déplacés si nécessaire. Si le processus XP n'est pas assez efficace, il est amélioré.
- **Conception** : la conception est basée sur la simplicité. Une manière compréhensible d'expliquer le système aux clients et aux personnes extérieures à l'équipe est décidée. Les solutions sont trouvées en isolant les problèmes du reste du système pour réduire les risques et les fonctionnalités supplémentaires sont toujours ajoutées à la fin du processus pour éviter de perdre du temps. Tout ce qui semble inutile ou redondant est réfracté pour maintenir la simplicité.
- **Codage** : le client est présent tout au long du processus de développement, pour prendre des décisions relatives aux dates de lancement, tester les fonctionnalités lancées et fournir un retour d'information. La programmation du code est effectuée par paires et selon des normes convenues. Le code est souvent intégré et se fait de manière à ce que chaque paire de programmeurs intègre son code à la fois pour éviter les conflits.
- **Tests** : des tests unitaires doivent être prévus pour tout le code et chaque test unitaire doit être réussi avant que le code ne soit publié. Des tests sont créés chaque fois qu'un bug est trouvé.

Ces règles encouragent la collaboration entre les membres de l'équipe et donnent du pouvoir à chacun d'entre eux. Les clients sont intégrés en tant que partenaires dans le processus, ce qui leur assure une excellente expérience. Tous les développeurs contribuent au travail et les gestionnaires facilitent la communication et gèrent les relations. La frustration est minimisée par le processus de réfraction et l'adaptation est facilitée en cas de changement des spécifications.

IV.3 Analyse des besoins

Dans cette section, nous allons définir les besoins de notre système en présentant les acteurs qui interagissent avec le système et ses cas d'utilisations.

Les acteurs du système:

Un acteur est une entité externe qui peut être un utilisateur humain, un dispositif matériel ou un autre système. Cette entité interagit avec un système en lui fournissant des données en entrée et en déclenchant ses fonctionnalités. Dans notre système, les acteurs sont des utilisateurs qui peuvent interagir avec le courtier et bénéficier de ses fonctionnalités. Ces utilisateurs peuvent être des clients ou des fournisseurs de SaaS.

Le tableau IV.1 présente les acteurs en question:

Acteur	Rôle
Client	L'entité qui souhaite utiliser les SaaS présents dans le système.
Fournisseur	L'entité qui publie les SaaS dans le système.

Tableau IV.1: Les acteurs et leurs rôles

Le diagramme de cas d'utilisation suivant montre les actions possibles de chaque acteur :

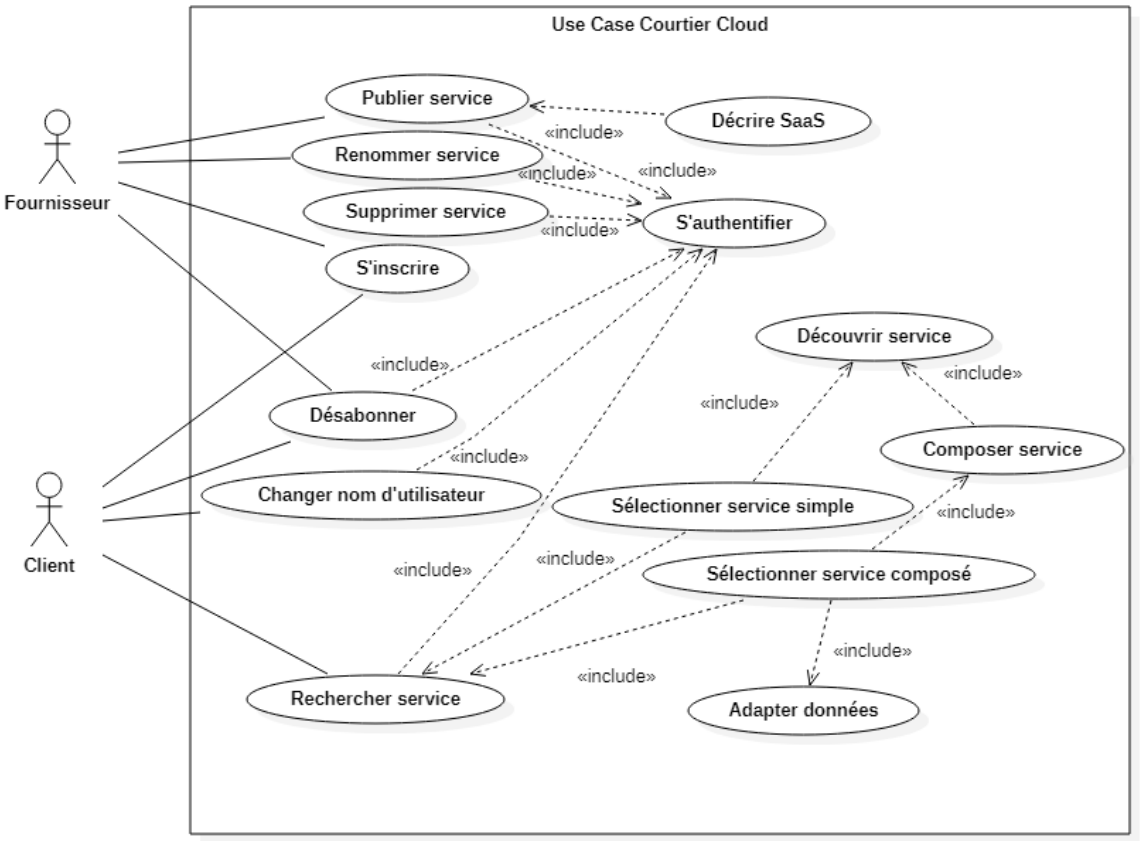


Figure IV.1: Diagramme de cas d'utilisation du système de courtage

Le tableau IV.2 présente une description détaillée de chaque cas d'utilisation:

Cas d'utilisation	Description textuelle
S'inscrire	Les fournisseurs ou clients peuvent s'abonner au courtier en remplissant les informations requises.
S'authentifier	Le client ou le fournisseur ne peut accéder au système que s'il en a le droit, il doit donc entrer un nom d'utilisateur et un mot de passe.
Publier service	Le fournisseur peut publier un service. Il doit fournir les informations et les caractéristiques relatives à ce service.
Rechercher service	Après l'authentification, un client peut rechercher des services en introduisant une requête.
Gérer compte	Un client peut modifier son compte en changeant son nom d'utilisateur.
Désabooner	Un client ou bien un fournisseur peut quitter le courtier en supprimant son compte.
Renommer service	Le fournisseur peut changer le nom d'un service qu'il a publié .
Supprimer service	Le fournisseur peut supprimer un service qu'il a publié .
Décrire service	À la publication d'un service, une description du service en question sera faite selon le modèle du système.
Découvrir service	À l'arrivée d'une requête du client, le courtier recherche le(s) service(s) souhaité(s).
Sélectionner service	Après l'étape de la découverte, une sélection basée sur les critères de qualité de service sera effectuée.
Composer service	Si aucun service ne répond au besoin de l'utilisateur, une composition de services sera effectuée.
Sélectionner service composé	Après la composition des services, une sélection du meilleur plan de composition doit être effectuée en fonction des critères de qualité du service.
Adapter données	Dans le cas d'une composition de plusieurs services, il est nécessaire d'adapter les formats des données si les services sélectionnés utilisent des formats de données hétérogènes.

Tableau IV.2: Description textuelle des cas d'utilisation

IV.4 Solution proposée

Après avoir analysé les besoins, nous présenterons la modélisation de notre solution dans cette section. Pour rappel, notre système est un courtier de Cloud Computing pour l'interopérabilité sémantique entre SaaS.

L'interopérabilité a déjà été présentée dans le Chapitre II, mais nous rappelons brièvement que c'est la capacité de différents systèmes hétérogènes à interagir et collaborer ensemble. Cette interaction implique un échange de données et une utilisation de ces données échangées.

Dans le Cloud Computing, l'interopérabilité fait référence à la collaboration entre les services de façon à ce que les données en sortie d'un service soient utilisées comme données d'entrée dans un autre service. Cette collaboration est l'origine de la problématique de composition de services qui nécessite la réalisation de la description, la découverte et la sélection des SaaS.

Notre solution est présentée sous forme de modules, tel que chaque module représente une fonctionnalité du système. Ces fonctionnalités sont les suivantes:

- La description des SaaS publiés dans notre registre par les fournisseurs inscrits dans le système selon notre modèle de description.
- La découverte des services pertinents à une requête de recherche du client.
- La sélection des meilleurs services parmi les services pertinents découverts.
- La composition de services répondant à la requête du client si celle-ci n'est pas satisfaite par un service unique.

Le traitement de ces fonctionnalités est modélisé selon les modules mentionnés ci-dessus et que nous détaillons dans les sections suivantes.

L'architecture suivante montre les éléments de notre système. En plus de détails, l'architecture présente les utilisateurs, leurs interfaces et les actions déclenchées par leurs requêtes.

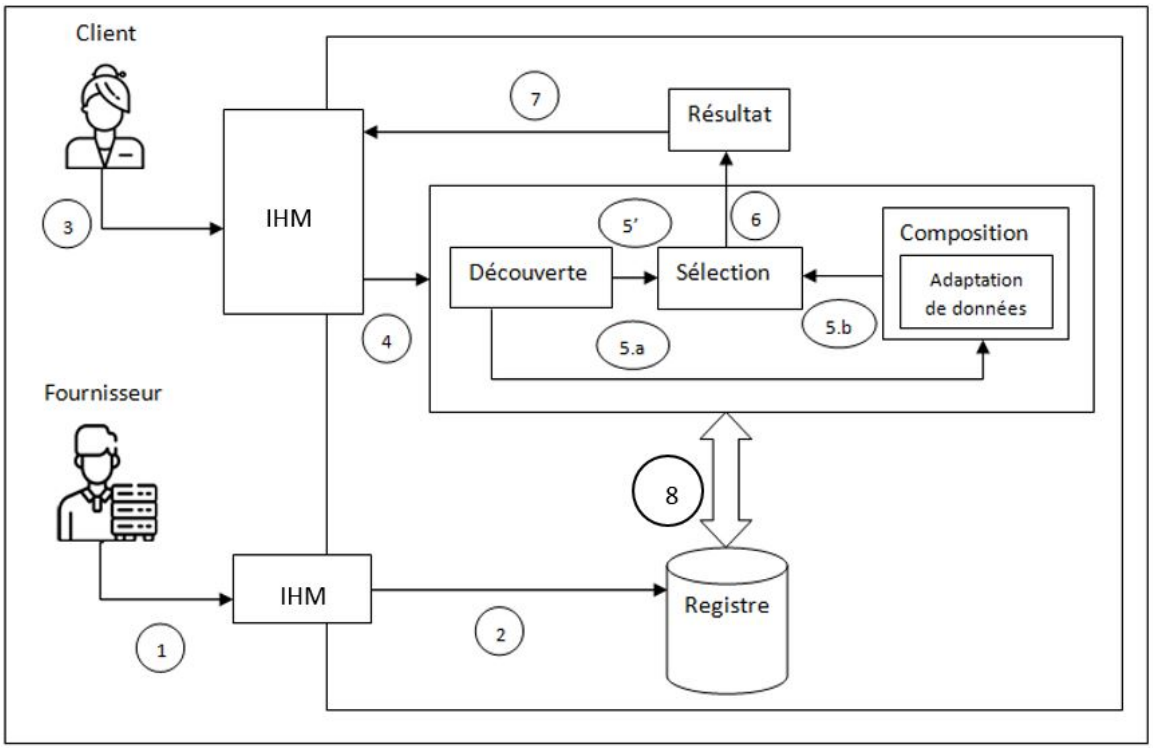


Figure IV.2: Architecture du système

Description des actions:

1- Requête de publication.

2- Intégration du nouveau service dans le registre.

3- Requête de recherche d'un service.

4- Passage de la requête de l'utilisateur au système.

5'- Passage du service découvert au module de sélection.

5.a- Passage des services découverts au module de composition.

5.b- Passage des plans de composition au module de sélection.

6- Retour du service sélectionné comme résultat.

7- Renvoie du résultat à l'utilisateur.

8- Récupération des informations relatives au service demandé depuis le registre.

IV.4.1 Module de description

Nous rappelons que ce module se charge de la description des SaaS publiés par le fournisseur inscrit dans le système.

Vu qu'il n y a pas de modèle de description standardisé pour les services de Cloud Computing, notre description sert à situer les services dans le système selon un modèle unifié.

Pour décrire les SaaS, nous avons d'abord définit les exigences de la description. Ce sont les aspects du service

que nous avons décidé de prendre en charge dans notre description.

E1: Prise en charge de l'aspect sémantique

L'aspect sémantique améliore les résultats de la découverte des services. Il permet une meilleure interprétation et compréhension du sens des concepts liés aux SaaS.

Dans notre solution, cette exigence est assurée par l'utilisation d'une ontologie comme support pour notre modèle de description. La conception de notre modèle a été influencée par le travail [61].

Les concepts du service sont introduits lors de la publication de service et sont représentés sous forme de classes de l'ontologie.

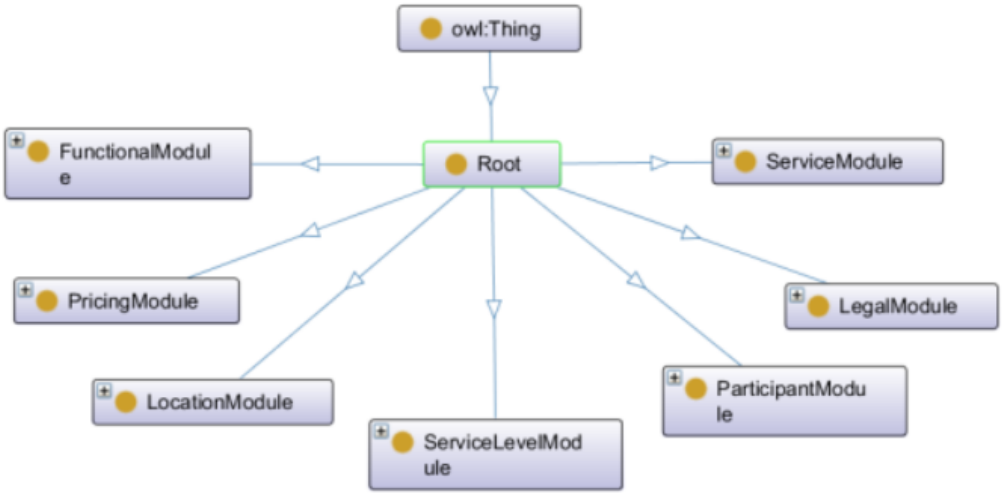


Figure IV.3: Les modules de l'ontologie

E2: Spécification des mesures d'évaluation des services

Le principe de l'évaluation des services est crucial dans l'environnement du Cloud Computing. Cette évaluation est relative aux aspects non-fonctionnels des services offerts et sert à informer les clients de la qualité de ces services pour leur permettre de faire les bons choix.

Par conséquent, notre description prend en charge cet aspect et permet de représenter la qualité de service pour assurer l'évaluation et la comparaison des services. Notre ontologie prend en compte certains critères de qualité de services comme le montre la figure IV.4.

E3 : Spécification de l'aspect fonctionnel du service

Les aspects fonctionnels permettent de définir ce que fait le service. C'est les aspects qui permettent de déterminer les opérations du service et ses traitements.

La sélection des services se fait généralement sur la base d'aspects fonctionnels en premier lieu. Notre description prend en compte cette exigence en l'exprimant dans l'ontologie comme l'illustre la figure IV.5.

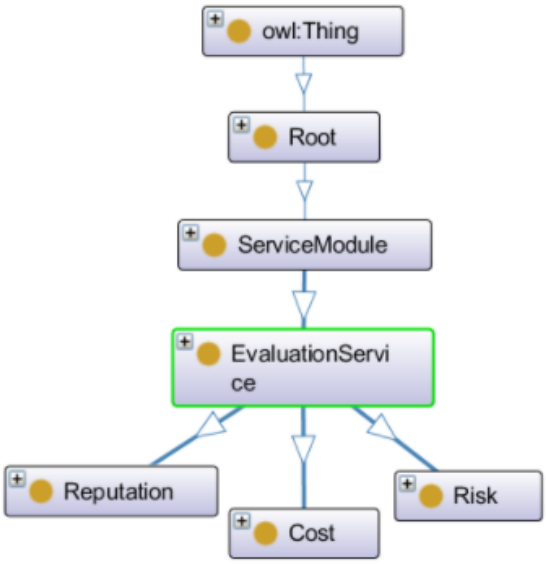


Figure IV.4: Les mesures d'évaluation

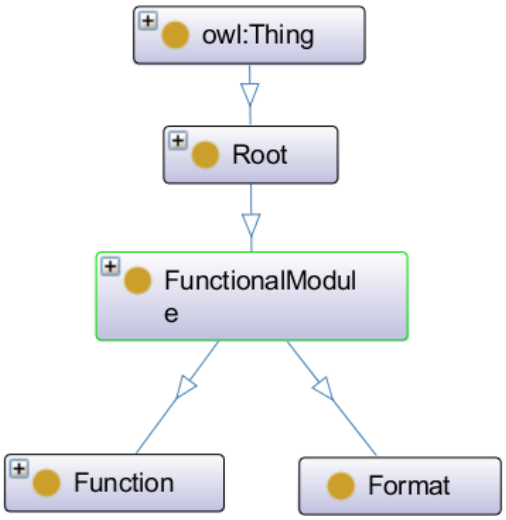


Figure IV.5: L'aspect fonctionnel

E4 : Représentation de la communication entre les services.

Nous avons déjà mentionné que notre système effectue une composition de services si nécessaire. Dans ce cas, les services doivent communiquer entre eux pour l'échange de messages et de données. Notre description prend donc en compte l'aspect de communication entre les services en précisant le point d'accès au service (le port) et le protocole de communication comme l'illustre la figure IV.6.

E5 : Représentation des aspects juridiques et financiers des services

Les services mis à disposition appartiennent à des prestataires ayant des politiques et des restrictions légales sur leurs activités. Notre description permet de spécifier les détails juridiques relatifs aux services et à leur utilisation en précisant ces détails dans l'ontologie.

La figure IV.7 illustre les aspects juridiques et financiers des services dans notre ontologie.

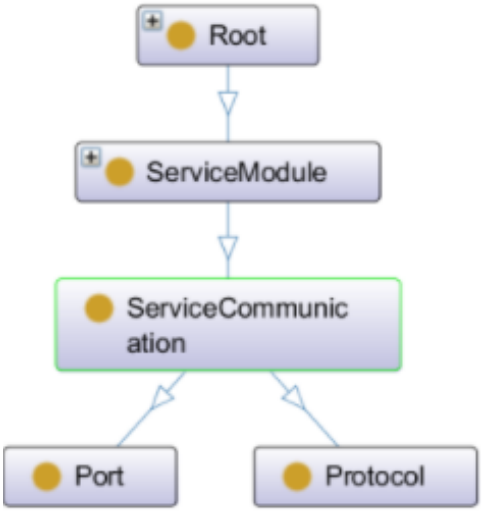


Figure IV.6: Aspects de communication

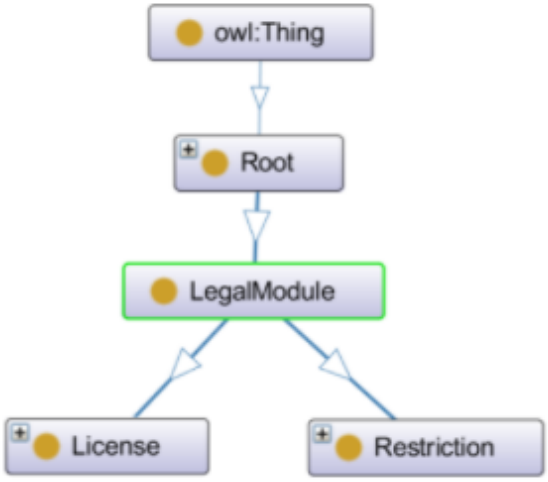


Figure IV.7: Aspects juridiques et financiers

E6 : Prise en charge du SLA

Après la sélection des services, il est important de consulter le contrat (L'accord de niveau de service ou SLA). Celui-ci doit couvrir les particularités de chaque service sélectionné. Notre description à base d'ontologie inclut le contrat de niveau de service (SLA) ainsi que les garanties et pénalités qui y figurent.
La figure IV.8 présente les aspects de l'accord de niveau de service (SLA) dans notre ontologie.

E7 : Prise en charge des contraintes de localisation

Les services fournis sont situés dans différents serveurs à travers le monde. Il se pourrait que les utilisateurs exigent des services qui existent dans des endroits spécifiques pour des raisons quelconques. Pour cela, notre description comprend les contraintes de localisation selon les zones géographiques et administratives comme le montre la figure IV.9.

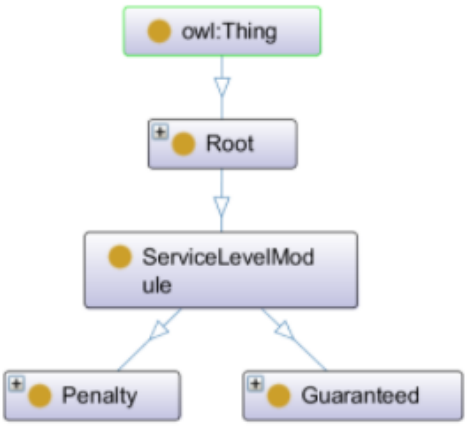


Figure IV.8: Le SLA

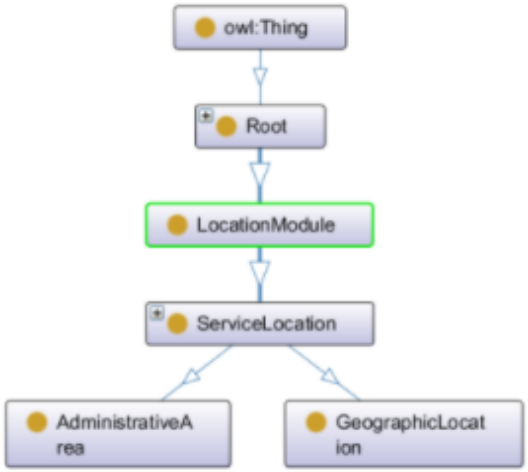


Figure IV.9: Contraintes de localisation

Dans le module suivant, nous allons expliquer le rôle de notre description dans la découverte des SaaS.

IV.4.2 Module de découverte

Nous avons déjà définit le concept de découverte de services dans le chapitre II.

Dans notre cas, nous traitons la découverte des SaaS selon la description sémantique à base d'ontologie que nous avons conçu et qui a été présentée dans le module précédent.

Nous utilisons également une base de données lexicale pour améliorer la gestion de similarité entre les concepts de la requête du client et les concepts des services publiés.

Nous allons expliquer en plus de détail à l'aide de l'exemple suivant:

Supposons que le client envoie une requête contenant des concepts spécifiques qui représentent ses besoins. Généralement, il s'agit de la fonction du service.

- **Requête: "Service to access account"**

Avant tout, un traitement initiale de la requête est nécessaire pour l'élimination des mots vides.

Notre traitement consiste à parcourir la requête du client et à vérifier si certains des mots de la requête sont présents dans notre inventaire des mots vides. Si des mots vides sont trouvés, ils sont éliminés de la requête.

- **Requête traitée: "account"**

Le registre de services peut contenir des services qui peuvent satisfaire la requête. Cependant, il se peut que les noms des concepts des services tels qu'ils sont dans le registre ne correspondent pas aux noms des concepts de la requête. On est donc confrontés à un problème de nomenclature.

- **Concept de service du registre : “?”**

Dans notre système, ce problème est traité à l’aide de la base de données lexicale. On y récupère tous les synonymes du concept de la requête du client. Cette étape est inspirée du travail [126].

La figure IV.10, représente les concepts similaires à celui de la requête du client.

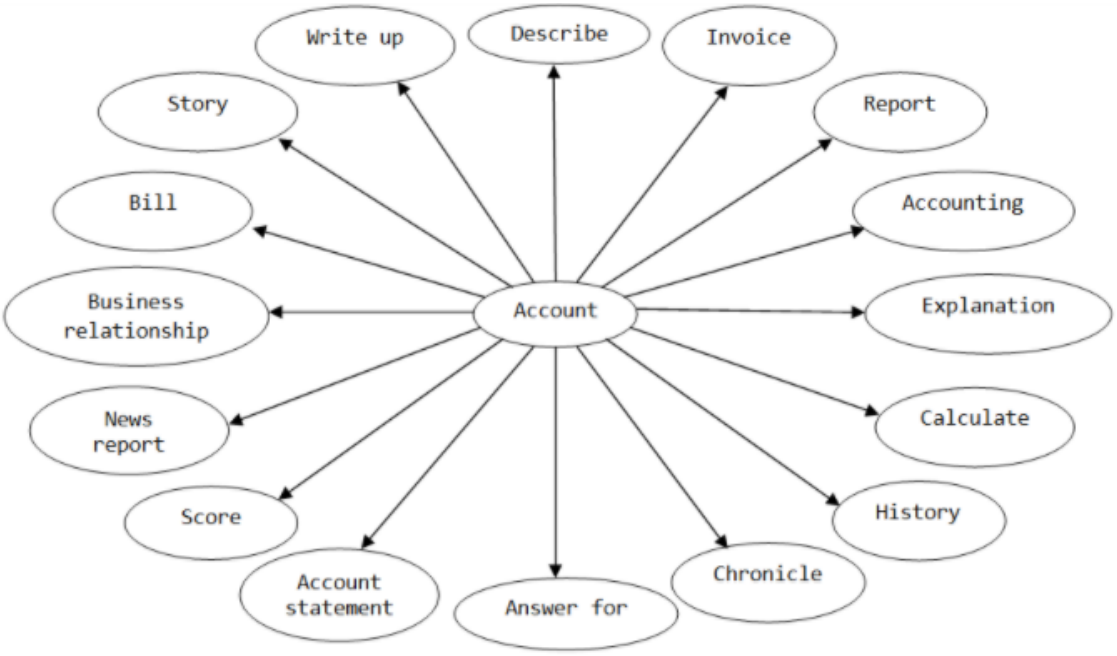


Figure IV.10: Représentation des concepts similaires

On obtient alors une liste des noms possibles du concept recherché.

- **Noms possibles des concepts du service dans le registre :**

{ “Account”,“Write up”, “Describe”, “Invoice”, “Repository”, “Accounting”, “Explanation”, “Calculate”, “History”, “Chronicle”, “Answer for”, “Account statement”, “Score”, “News report”, “Score”, “News report”, “Business relationship”, “Bill”, “Story” }

Maintenant que nous avons réglé cet aspect, il est nécessaire de s’assurer que le service souhaité est présent dans le registre. C’est à dire, on parcourt le registre pour voir si l’un des concepts de la liste des synonymes obtenue est présent dans le registre en tant que fonction d’un service.

Nous rappelons que dans notre description, les fonctions des services sont spécifiées dans l’ontologie qui représente notre registre de service.

Les informations récupérées du registre sont triées de sorte que les SaaS hébergés dans des serveurs de Cloud Computing qui contiennent le plus grand nombre de SaaS sont traités en premier (tri décroissant). Cela sert à

améliorer les performances de notre système en terme de temps de réponse car plus un serveur contient de services, plus il y a de chances de trouver le service souhaité dans ce serveur et une fois le service trouvé, on n'a plus besoin de chercher dans d'autres serveurs. Ce tri est inspiré du travail de Kurdi et al. [23].

Fonctions des services du registre	**Concepts recherchés**
Storage	Account
Accounting	Invoice
Design	**Accounting**
Messaging	Bill

Tableau IV.3: Le parcours des concepts du registre

Après la comparaison des concepts similaires aux concepts de la requête avec les concepts du registre, on trouve une correspondance avec le concept "Accounting". Donc, le SaaS recherché est bel et bien dans le registre et la découverte s'achève.

Dans le module suivant, nous allons montrer comment se passe la sélection des services découverts de sorte à ce que le meilleur service soit choisi.

IV.4.3 Module de sélection

Après la découverte des services qui répondent à l'aspect fonctionnel de la demande du client, une sélection de service sera effectuée.

Toujours suivant l'exemple utilisé dans le module de découverte, nous allons expliquer comment la sélection des services pertinents est effectuée.

Lorsqu'un serveur hébergeant le service souhaité est trouvé, il se peut que ce serveur héberge plus d'un seul service qui fournit la fonction désirée. On est donc confrontés à un problème de prise de décision. Quel service choisir?

Dans notre travail, cet aspect est traité par une sélection basée sur les critères de qualité de service (QoS). Chaque service hébergé a une valeur qui quantifie chacun de ces critères de qualité de service.

Le tableau suivant montre les valeurs de qualité de service de trois services dont la fonction est "Accounting" et qui peuvent satisfaire la requête de l'utilisateur.

	Service1	Service2	Service3
Coût	0.4	0.1	0.5
Réputation	0.2	0.3	0.5
Risque	0.4	0.2	0.4

Tableau IV.4: Les services candidats et les valeurs de leurs critères de QoS

L'utilisateur fournit des valeurs de poids de priorité pour chaque critère de qualité de service qui servent d'indice d'importance à ces critères. Plus un critère de qualité de service est important pour l'utilisateur, plus grand doit être son poid. La somme de tous les poids doit être égale à 1.

Ces poids sont ensuite utilisés pour le calcul de scores des services. Le meilleur service sera choisi selon ces scores. Le score en question est calculé selon la formule suivante, qui est inspiré du travail [125].

$$ScoreS(S) = \Sigma_{j=1}^{n} Wj.Crj$$

ScoreS: le score du service. Il représente la somme des produits des valeurs des critères multipliés par leurs poids respectifs.

n: nombre de critère de qualité de service traités

Wj: Poids du critère j

Crj: Valeur du critère j

Les tableaux suivants montrent les valeurs de qualité de service ainsi que les valeurs de poids de priorité associés pour chaque service.

Nous allons démontrer le calcul de score pour le service 1 en détail, mais pour le reste des services, la valeur du score sera donnée directement.

Service1:

Critères de qualité de service (QoS)	Valeur associée au critère de qualité de service	Poid de priorité de chaque critère
Coût	0.4	0.3
Réputation	0.2	0.2
Risque	0.5	0.5

Tableau IV.5: Valeurs des critères de QoS et des poids pour le Service1

ScoreS (Service1) = (0.4 × 0.3) + (0.2 × 0.2) + (0.5 × 0.5)

=0.12 + 0.04 + 0.25

= **0.41**

Service2:

Critères de qualité de service (QoS)	Valeur associée au critère de qualité de service	Poid de priorité de chaque critère
Coût	0.2	0.3
Réputation	0.3	0.2
Risque	0.5	0.5

Tableau IV.6: Valeurs des critères de QoS et des poids pour le Service2

ScoreS (Service2) = 0.37

Service3:

Critères de qualité de service (QoS)	Valeur associée au critère de qualité de service	Poid de priorité de chaque critère
Coût	0.4	0.3
Réputation	0.2	0.2
Risque	0.4	0.5

Tableau IV.7: Valeurs des critères de QoS et des poids pour le Service3

ScoreS (Service3) = 0.36

Après le calcul de score, on obtient les résultats suivants:

Service	Score
Service1	**0.41**
Service2	0.37
Service3	0.36

Tableau IV.8: Les scores des services candidats

Le service ayant le meilleur score est: **Service1**.
C'est donc **Service1** qui sera sélectionné et retourné comme résultat de la requête du client.

IV.4.4 Module de composition

Dans le cas où aucun service unique ne répond à la demande du client, une composition doit être effectuée. Cette composition implique la sélection de deux services ou plus, tel que l'ensemble de ces services répond aux besoins du client.

Notre composition de services est une composition réactive basée sur la méthode de sélection par planification globale. C'est à dire qu'à la réception de la requête du client, une recherche des services demandés a lieu et la sélection des services ne se passe pas selon les valeurs de critères de qualité de chaque service qui entre dans la composition, mais plutôt selon la valeur des critères de qualité des services en tant que services composés.
Nous allons utiliser un exemple différent pour montrer comment cela se passe.

- **Requête: "Service to do the sum of values and store them"**
- **Requête traité: "addition store"**
- **Concept de services du registre: "? ?"**
- **Noms possibles des concepts des services dans le registre :** [{"Add", "Total", "Summation", etc. }, {"Store", "Storage", "Depot", etc }].
- Fonctions désirées retournées après la découverte: **"Add"**, **"Store"**
- Services associés aux fonctions retournées:

Add	**Store**
Service Add1	Service Store1
Service Add2	Service Store2

Tableau IV.9: Les services candidats pour la requete de composition

Nous allons maintenant résumer tout ce qui a été dit concernant les modules de découverte, de sélection et de composition à travers des pseudo-algorithmes avant de passer au module de l'adaptation des données.

Algorithm IV.1 Cloud selection

Input: A set of Cloud Computing servers C(C1,C2,...) with services that belong to them , User request processed and written as keywords UR(S1,S2,...)
Output: Set of Cloud Computing servers containing services qualified for composition
Assumption: Cloud Computing servers are sorted in a descending order based on the number of services they host
Begin
1. $QC \leftarrow \phi$// Initializing the output
2. $QS \leftarrow \phi$ // Initializing the set of services which are qualified for service composition
3. $FS \leftarrow \phi$// Initializing the set of functions similar to the concepts of the user request
4. $SC = SimilarConcepts(UR) \cup UR$// Getting all concepts similar to the concepts of the user request UR using lexical database
5. $FU \leftarrow$ getallFunctions() //Getting all functions in the registry
6. For each F $\in FU$ // Running through every function F in the set FU
7. For each Concept Co $\in SC$ // Running through every concept Co in the set SC
8. If Equals(F , Co) then
9. $FS \leftarrow FS \cup F$ // Keeping all functions in set of functions FU that are similar to the concepts from the set SC
10. End for
11. End for
12. For each Cloud Ci $\in C$
13. $CF \leftarrow$ getallFunctions(Ci)//get all functions in Cloud Computing server Ci
14. If $FS \cap CF \neq \phi$ then //Check if this server contains a service that satisfies the user request
15. $QC \leftarrow QC \cup Ci$ //Keep the server in a set of qualified Cloud Computing servers QC
16. $QS \leftarrow QS \cup Service(FS \cap CF)$ //Keep the services found in the server in a set of qualified services QS
17. End if
18. If $|QS| = |UR|$ then break
19. End if
20. End for
21. Return QC
22. End

Explication de l'Algorithme IV.1 :

- On suppose que les serveurs de Cloud Computing du registre sont triés dans un ordre décroissant en fonction du nombre de services qui y sont hébergés afin d'augmenter les chances d'obtenir tous les services pertinents à partir d'un nombre minimum de serveurs (il est plus probable qu'un serveur héberge plusieurs services pouvant satisfaire la demande de l'utilisateur s'il contient un grand nombre de services).
- Nous commençons par traiter les concepts de la demande du client. Tous les concepts similaires possibles d'un service demandé sont rassemblés et stockés dans un ensemble SC. Ceci est fait en utilisant une base de données lexicale (ligne 4).
- De (ligne 6) à (ligne 11), nous conservons toutes les fonctions des services du registre qui sont similaires aux concepts de l'ensemble SC dans un ensemble FS.
- Nous parcourons chaque serveur du registre afin d'obtenir les services répondant à la demande (ligne 12).
- Si des services satisfaisant la demande du client ont été trouvés, nous ajoutons le serveur qui les héberge à un

ensemble QC qui stocke tous les serveurs qui hébergent des services pertinents, et nous ajoutons ensuite ces services à l'ensemble QS. (ligne 13,14,15).

- Une fois que tous les services demandés ont été trouvés, l'algorithme se termine (ligne 18) et renvoie un ensemble de serveurs qui hébergent des services pertinents pour la demande du client (ligne 21).

Algorithm IV.2 Service selection

Input: Set of Cloud Computing servers containing qualified services for the composition QC(C1, C2, . . . , Cn) , Requested services RS.
Output: Best service composition plan BP
Begin
1. $SSC \leftarrow \phi$ //Set of all sets of services from QC that satisfy the request
2. $ForeachCi \in QC$
3. $SS \leftarrow GetRelevantServices()$ //Get all services that satisfy the request
4. $SSC \leftarrow SSC \cap SS$
5. End for
6. $SSF \leftarrow Sort(SSC)$ //Sort the relevant services by function into separate sets
7. $CP \leftarrow \phi$ //Set of services in the composition plan
8. $CPS \leftarrow \phi$ //Set of all composition plans
9. $CP \leftarrow Compose(SSF)$ //Generate a service composition using a service from each set of services that are sorted based on function
10. $CPS \leftarrow CPS \cup CP$ //Add the generated composition plan to the set of all composition plans
11. $BP \leftarrow SelectBestPlan(CPS)$ //Select the best composition plan based on QoS score
12. return BP
13. End

Explication de l'Algorithme IV.2 :

- Cet algorithme prend en entrée un ensemble de serveurs de Cloud Computing QC contenant des services qualifiés pour la composition du service qui satisfait la demande du client ainsi qu'un ensemble de services demandés RS.
- La sortie est le meilleur plan de composition BP choisi parmi un ensemble de plans de composition générés.
- Nous commençons par passer en revue l'ensemble QS de tous les serveurs de Cloud Computing qui hébergent des services qui satisfont la demande de l'utilisateur. À la ligne 3, nous sauvegardons chaque service du serveur actuel qui correspond à l'un des services demandés dans un ensemble SS.
- Ensuite, à la ligne 4, nous ajoutons les services collectés à partir du serveur courant et sauvegardés dans SS à un ensemble SSC qui contiendra tous les services pertinents collectés à partir de tous les serveurs.
- Dans (ligne 6), nous trions tous les services pertinents obtenus à partir de tous les serveurs de l'ensemble SSC par ordre de fonction (tous les services ayant la même fonction sont regroupés) et nous les stockons dans des ensembles séparés de sorte que chaque ensemble contienne des services ayant la même fonction. Ces ensembles sont ensuite stockés dans un ensemble SSF.
- Ensuite, nous générons plusieurs plans de composition en utilisant tous les services stockés dans tous les ensembles de SSF, de sorte que chaque plan de composition de service contient un service provenant de l'un des ensembles SSF et dont la fonction a été demandée dans la requête du client (ligne 9). Tous les plans de composition sont stockés dans un ensemble CPS (ligne 10).

- Une fois que nous avons généré les plans de composition des services, nous avons sélectionné le meilleur plan de composition BP parmi l'ensemble des plans de composition CPS (ligne 11). Cette sélection est basée sur un score calculé à l'aide de critères de qualité de service (QoS).
- À la fin de l'algorithme, le meilleur plan de composition des services BP est retourné (ligne 12).

IV.4.5 Module d'adaptation des formats de données

L'adaptation des données est nécessaires lorsqu'une composition a lieu. Ceci est parce qu'il se pourrait que les services qui sont dans le registre et qui seront potentiellement composés appartiennent à de différents fournisseurs qui n'utilisent pas forcément les mêmes modèles et les mêmes formats de données. Ceci pose un problème car, comme nous l'avons déjà dit au début, une composition implique l'utilisation des données de sortie d'un service comme données d'entrée d'un autre service.

Nous gardons l'exemple utilisé dans le module de la composition. Après les calculs nécessaires, la meilleure composition s'est avérée être celle qui implique: **Service Add1** et **Service Store2**.

Plan	**Score**
P1= Service Add1 × Service Store1	0.395
P2 = Service Add1 × Service Store2	0.425
P3= Service Add2 × Service Store1	0.32
P4= Service Add2 × Service Store2	0.35

Tableau IV.10: Plans de composition

Nous rappelons que lorsqu'un fournisseur inscrit son service dans notre registre, il fournit également des informations relatives au format de données utilisé par ce service. Donc, une fois que nous savons quels services entrent dans la composition, nous avons aussi la connaissance de leurs formats de données.

Supposons que les deux services, Service Add1 et Service Store2, ont des formats de données différents comme le montre les figures IV.11 et IV.12.

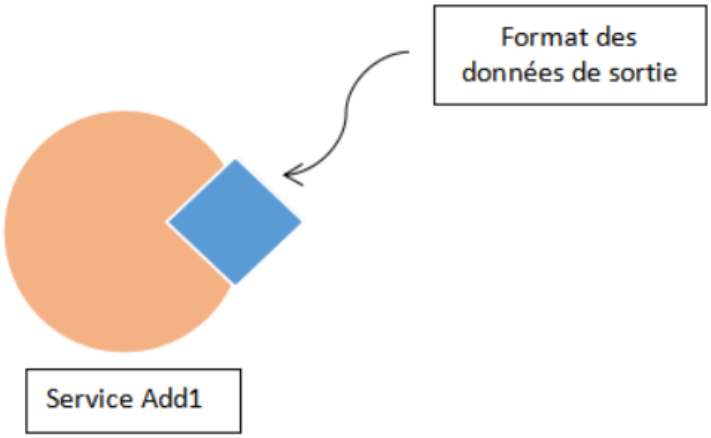


Figure IV.11: Exemple du format du Service Add1

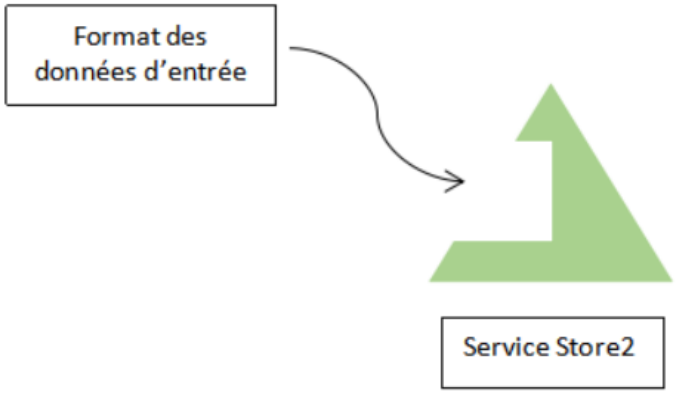


Figure IV.12: Exemple du format du Service Store2

Dans ce cas, le Service Store2 ne peut pas utiliser les données de sortie du Service Add1 comme données d'entrée car les formats ne sont pas compatibles comme nous l'expliquons dans la figure (IV.13).

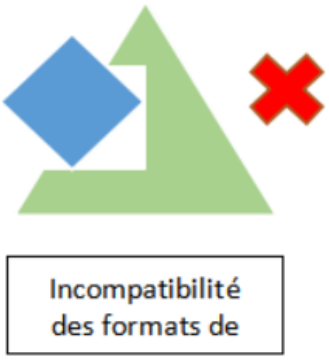


Figure IV.13: Exemple de formats incompatibles

Cette différence et incompatibilité de formats pose un problème pour la composition des deux services.

Le système doit donc adapter le format des données de sortie du **Service Add1** pour que ces données puissent être utilisées par le **Service Store2**. Cette adaptation des formats de données est basée sur l'adaptation du format d'un service dès sa publication avec un format pivot qui sert d'intermédiaire entre les formats de données des autres services. Avec cette adaptation, le format de données du premier service sera compatible avec le format de données du deuxième service par le biais du format pivot.

La figure IV.14 explique comment se passe l'adaptation des formats de données

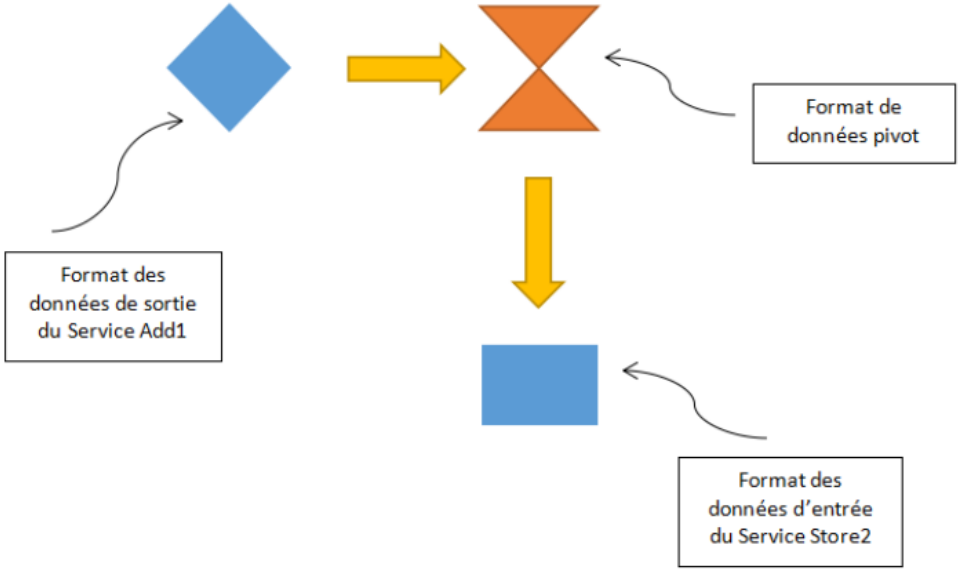


Figure IV.14: Exemple d'adaptation des formats de données

Après que le format des données du Service Add1 soit adapté, le format de données produit en sortie devient compatible avec le format de données d'entrée du Service Store2. L'utilisation des données du premier service par le

deuxième service est maintenant possible et le problème d'incompatibilité de données est réglé.

L'intérêt de l'utilisation d'un format pivot est la diminution de la complexité du système. Si le format de chaque nouveau service publié devait être adapté avec les formats des autres services qui sont dans le registre, ceci va engendrer une grande complexité de traitement. Alors que lorsqu'on adapte le format de données d'un nouveau service lors de sa publication avec un format pivot, ce dernier va servir d'intermédiaire avec les formats de données des autres services qui sont déjà adaptés avec lui. L'exemple suivant illustre ce point en détail:

Supposons que nous avons 5 formats de données différents qui sont représentés par les extensions suivantes: ".a", ".b", ".c", ".d", ".e". Si on procède à une adaptation de données directe d'un format de données à l'autre, le résultat obtenu est montré dans le tableau IV.11 où chaque adaptation de données est marquée par un X.

	.a	**.b**	**.c**	**.d**	**.e**
.a	**/**	**X**	**X**	**X**	**X**
.b	**X**	**/**	**X**	**X**	**X**
.c	**X**	**X**	**/**	**X**	**X**
.d	**X**	**X**	**X**	**/**	**X**
.e	**X**	**X**	**X**	**X**	**/**

Tableau IV.11: Adaptation des formats des données sans format pivot

Maintenant, supposons que nous allons effectuer une adaptation des formats de données en passant par un format pivot que l'on représente par l'extension ".w". Les résultats apparaissent dans le tableau IV.12 où chaque adaptation de données est marquée par un X.

	.w	**.a**	**.b**	**.c**	**.d**	**.e**
.w	**/**	**X**	**X**	**X**	**X**	**X**
.a	**X**	**/**	**/**	**/**	**/**	**/**
.b	**X**	**/**	**/**	**/**	**/**	**/**
.c	**X**	**/**	**/**	**/**	**/**	**/**
.d	**X**	**/**	**/**	**/**	**/**	**/**
.e	**X**	**/**	**/**	**/**	**/**	**/**

Tableau IV.12: Adaptation des formats des données avec format pivot

On observe que dans le tableau IV.11, lorsque chaque format de données est adapté avec les autres formats, le nombre d'opérations d'adaptation a atteint 20. Alors que dans le tableau IV.12, lorsque le format de chaque service

est adapté avec le format pivot, on voit qu'il n'y a eu que 10 opérations d'adaptation. Soit, une réduction de 10 opérations de passage d'un format à un autre.

Après que le format des données du Service Add1 soit adapté, un nouveau format de données compatible avec le Service Store2 est produit. L'utilisation des données du premier service par le deuxième service est maintenant possible et le problème d'incompatibilité de données est réglé.

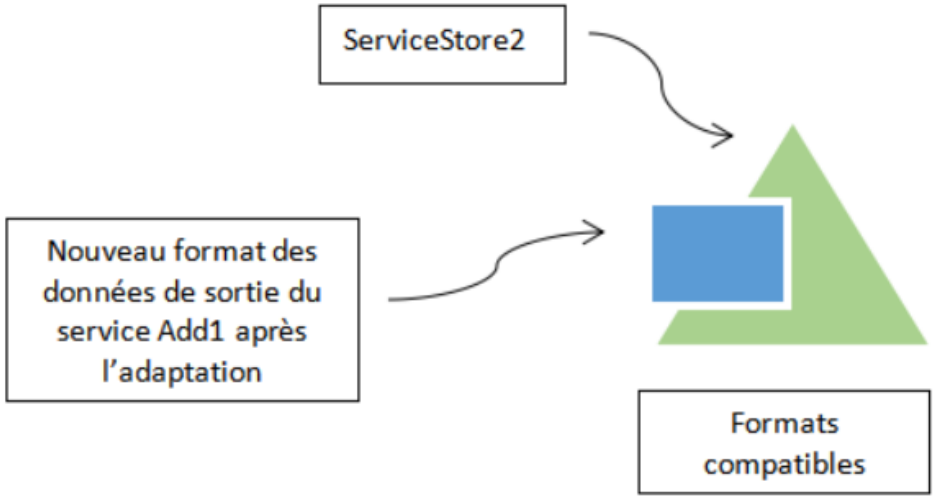


Figure IV.15: Exemple du format compatible

IV.5 Conclusion

Dans ce chapitre, nous avons présenté notre solution proposée au problème de l'interopérabilité sémantique entre les SaaS.

Nous avons commencé par introduire la démarche de travail suivie. Puis, nous avons fait une analyse des besoins pour identifier les acteurs du système et leurs actions respectives. Ensuite, nous nous sommes lancées dans la présentation de notre solution en commençant par son architecture suivie de ses modules. Nous avons expliqué en détail chacun des modules de description, de découverte, de sélection, de composition et d'adaptation des données en utilisant des exemples illustratifs et des algorithmes.

Dans le chapitre suivant, nous allons montrer en détail comment nous avons implémenté notre solution en donnant les outils utilisés et le résultat final.

Chapitre V

Implémentation de la solution

V.1 Introduction

Dans le chapitre précédent, nous avons présenté la conception de notre solution. Dans ce chapitre nous allons passer à l'implémentation de notre système.

Ce chapitre va comporter une partie pour spécifier les ressources matérielles et logicielles utilisées ainsi que des imprimes écrans qui montrent chaque interface de notre système suivant leurs cas d'utilisation. De plus, nous allons montrer comment nous avons effectué le déploiement d'un module de notre solution.

V.2 Ressources matérielles

Afin de réaliser notre application, nous avons utilisé les ressources matérielles suivantes:

- Ordinateur portable Samsung NP350V5C-S05
- Processeur Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz 2.50GHz
- Mémoire RAM 6.00 Go (5.89 Go utilisable)
- Système d'exploitation 64 bits, processeur x64
- Windows 10 Professionnel version 2004

V.3 Outils et environnement de développement

- **Eclipse**

  Eclipse IDE est un environnement de développement intégré écrit en Java publié par IBM, qui peut permettre l'utilisation de n'importe quel langage de programmation pour créer des projets de développement [128].

  La particularité d'Eclipse IDE vient du fait que son architecture est entièrement développée autour du concept de plug-ins, et que toutes les fonctions du logiciel sont développées sous forme de plug ins [129].

- **JDK**

  Le JDK ou Java Development Kit est un environnement de développement qui permet aux utilisateurs de créer des applications, des applets et des composants en Java. Il contient des outils qui peuvent être utilisés non seulement pour le développement mais aussi pour le test de programmes qui fonctionnent sur la plate-forme Java [130].

  Le kit de développement Java contient le Java Runtime Environment (JRE) qui est la plate-forme sur laquelle les programmes Java s'exécutent [131] un compilateur Java qui transforme le fichier source java en un fichier bytecode [132], ainsi que de nombreuses API JAVA [133].

- **Java EE**

  Java EE (Java Platform, Enterprise Edition) est une plateforme qui permet la création d'applications web.

Elle offre également des fonctionnalités qui sont particulièrement intéressantes dans le développement d'applications au sein des entreprises telles que: « l'injection de dépendance (CDI, EJB), la gestion des transactions (JTA), la fonctionnalité de page web dynamique (JSP, JSF), ainsi que des fonctionnalités de création de services web (JAX-RS, JAX-WS) » [134]. Java EE compte aussi l'avantage d'un ensemble d'APIs performantes qui servent à faciliter le développement d'une application web tout en réduisant le temps de développement et sa complexité. Cette plateforme a été développée au sein du programme JCP (Java Community Process) selon des standards de stabilité et de compatibilité [135].

- **Tomcat**

  Tomcat est un logiciel open source qui représente une implémentation des technologies Java Servlet, Java Server Pages, Java Expression Language ainsi que Java WebSocket [136]. Tomcat est un serveur web et un conteneur de servlet, c'était d'abord un projet de Sun Microsystems avant d'être passé à l'Apache Foundation. Ce serveur est particulièrement utilisé pour le déploiement d'applications développées selon le modèle MVC (Model-View-Controller) et se basant sur les technologies de servlets et de JSP (Java Server Pages) [137]. Tomcat est caractérisé par sa stabilité et sa capacité à travailler avec d'autres serveurs web [138].

- **Protégé**

  Protégé est un éditeur d'ontologies libre et gratuit développé par l'université de Stanford. Il est riche en fonctionnalités permettant la création, l'édition et la visualisation d'ontologies. Protégé offre un support complet pour OWL 2 et des connexions directes aux raisonneurs de logique de description comme Pellet. Il dispose d'une interface utilisateur personnalisable et d'outils de visualisation qui permettent d'interagir avec les ontologies et leurs relations. Protégé bénéficie d'une large communauté et est disponible sous la forme d'une application de bureau ainsi que d'une version accessible en ligne appelée WebProtégé [139].

- **MYSQL**

  MySQL est un système de gestion de bases de données relationnelles à code source ouvert, développé, distribué et soutenu par Oracle Corporation. Il permet aux utilisateurs d'accéder et de traiter facilement les données stockées dans les différentes tables de la base de données. Il se caractérise par sa rapidité, son évolutivité, sa sécurité et sa simplicité d'utilisation [140].

  MySQL est utilisé par des sociétés bien connues comme Facebook, Google, Adobe, Alcatel Lucent et Zappos [141].

- **SQL**

  SQL (Structured Query Language) est un langage d'interrogation commun et standardisé pour l'accès et la manipulation de bases de données relationnelles. Les instructions SQL sont des instructions à la base de données et permettent aux utilisateurs d'effectuer diverses tâches allant de l'insertion, la mise à jour et la suppression des tables à la récupération de données spécifiques et au contrôle de l'accès à la base de données [142].

Les instructions SQL peuvent être utilisées directement par les systèmes de gestion de base de données, mais elles peuvent également être intégrées dans du code écrit dans des langages comme Java [140].

- **L'API JDBC**

  L'API JDBC ou Java Database Connectivity est une bibliothèque java qui permet l'accès aux données stockés sur une multitude de supports allant des fichiers Excel aux serveurs de bases de données relationnelles [143].

- **WordNet**

  WordNet est une base de données lexicale en ligne contenant des noms, verbes et adjectifs anglais. Ces entités sont organisées en ensembles de synonymes reliées par de différentes relations.

  WordNet était conçue au début, en 1985 par un groupe de psychologues et de linguistes de l'université de Princeton pour « fournir une aide à la recherche dans les dictionnaires de manière conceptuelle, plutôt que simplement alphabétique - elle devait être utilisée en étroite conjonction avec un dictionnaire en ligne de type classique.»[144]

- **JAWS API**

  Java API for WordNet Searching ou simplement JAWS est une API qui permet aux utilisateurs d'ajouter à leurs applications la possibilité de recherche et de récupération de données depuis la base de données WordNet. Cette API est caractérisée par sa simplicité et sa rapidité.

  JAWS est compatible avec les versions 2.1 et 3.0 de la base de donnée WordNet et est fonctionnelle sur JAVA 1.4 et les versions ultérieurs [145].

- **L'API WS4J**

  WS4J ou bien WordNet Similarity for Java est une API java qui prend en charge des algorithmes de calcul de similarité [146]. Cette API est conçue pour être utilisée avec les instances de la base de données lexicale WordNet [147].

- **OWL API**

  OWL API est une API Java qui permet la création, l'analyse, la manipulation et la sérialisation des ontologies OWL. Elle est utilisée depuis 2003 [148] et est publiée sous une double licence open source : LGPL et Apache. Avec l'évolution de OWL, l'API OWL a également évolué et a connu différentes versions. Par exemple, la version 3.1 de l'API OWL et les versions plus récentes sont conçues pour OWL 2.

  OWL API comprend divers composants comme [149] :

  - L'implémentation de référence en mémoire pour l'API OWL2.
  - Des parseurs et rédacteurs pour RDF/XML, OWL/XML, OWL Functional Syntax et Turtle.
  - Des interfaces de raisonneurs pour les raisonneurs externes comme : FaCT++, HermiT, Pellet, Racer, JFact et Chainsaw

- **API Jena**

  Jena est une API Java open source développée dans les laboratoires HP en année 2000 avant d'être adoptée par

la fondation Apache en 2010. Elle entre en jeu dans la création d'applications du web sémantique [150].Cette API fournit des méthodes pour créer, interroger et manipuler des données en RDF, RDFS et OWL en utilisant le langage SPARQL et selon les recommandations du W3C. Jena permet la manipulation des ontologies en OWL et RDFs et elle comporte également un moteur d'inférence basé sur les règles pour effectuer des opérations de raisonnement sur ces ontologies [151].

- **SPARQL**

  SPARQL (SPARQL Protocol and RDF Query Language) est un langage qui permet d'exprimer des requêtes d'interrogation de données en RDF recommandé par le W3C.

  SPARQL supporte quatre types de requêtes, SELECT permet de sélectionner tout ou partie du graphe, CONSTRUCT permet de construire un nouveau graphe RDF, ASK retourne vrai s'il y a une correspondance sinon elle retourne faux, et la requête DESCRIBE retourne un graphe RDF qui décrit les ressources trouvées [152].

- **Docker**

  Docker est une technologie de conteneurisation qui permet le packaging et la distribution des applications et de leurs dépendances[21].

  Une image Docker est en réalité un package léger et exécutable qui comporte le code, les libraires, la configuration et les outils nécessaires pour l'exécution d'une application. Cette image devient un conteneur à l'exécution [20].

- **Docker Desktop**

  Docker Desktop est un outil pour les machines utilisants les systèmes d'exploitation Windows et MacOS qui permet la création et le partage d'applications conteneurisées. Il est caractérisé par sa vitesse et sa sécurité.

  Docker Desktop offre une grande flexibilité en termes de langages de programmation et d'environnements de déploiement et est également facile à manipuler simplement à travers une invite de commande. Les images de conteneurs créées avec Docker Desktop peuvent être stockées et partagées sur le registre en ligne Docker Hub. En plus de faciliter le travail collaboratif, ce dernier met à la disposition des développeurs de nombreuses images qui peuvent être utilisées comme ressources dans leurs travaux [153].

V.4 L'implémentation du registre

Nous avons mentionné précédemment que notre registre est une ontologie. Pour l'implémentation de cette ontologie, nous avons utilisé le logiciel Protégé qui a été présenté dans la section précédente.

Les classes de notre ontologie ont déjà été montrées dans le chapitre de conception suivant les exigences auxquelles elle répondent. La figure suivante montre les classes du premier niveau de la hiérarchie.

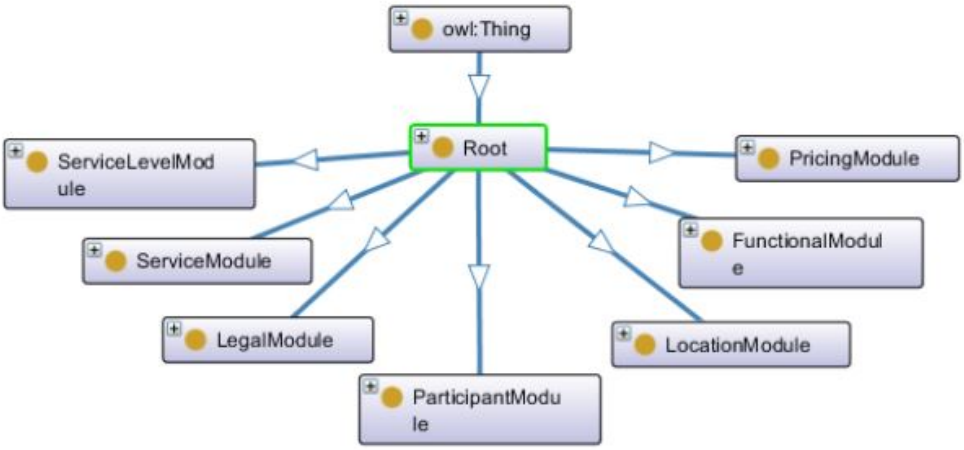


Figure V.1: Modules du registre

Chaque classe est reliée à sa classe supérieure dans la hiérarchie par une relations “Subclass Of”. Ces relations sont représentées par des flèches en bleu dans la figure suivante:

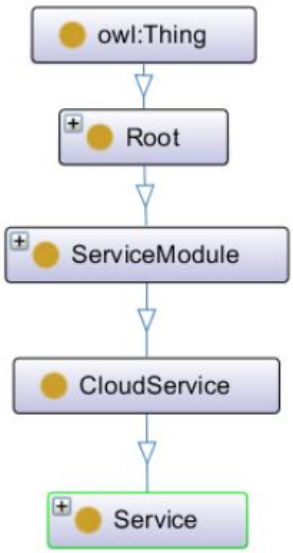


Figure V.2: Relations "SubClassOf" entre les classes

Les classes sont caractérisées par des propriétés telles que les “Object Properties” qui relient les classes entre elles, et les “Data Properties” qui définissent les types de valeurs que les instances des classes peuvent avoir. Les deux figures suivantes montrent ces deux types de propriétés:

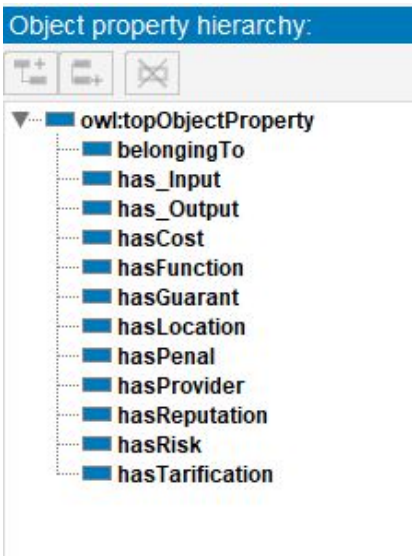


Figure V.3: ObjectProperties

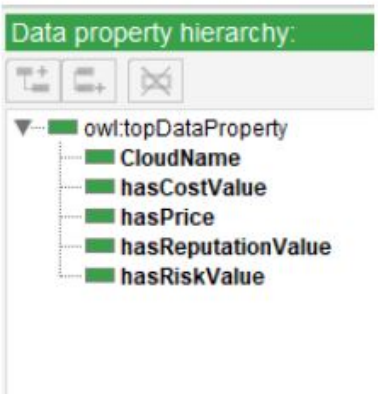


Figure V.4: DataProperties

Les instances de classes sont appelées “Individuals”. La figure suivante montre des instances de la classe service:

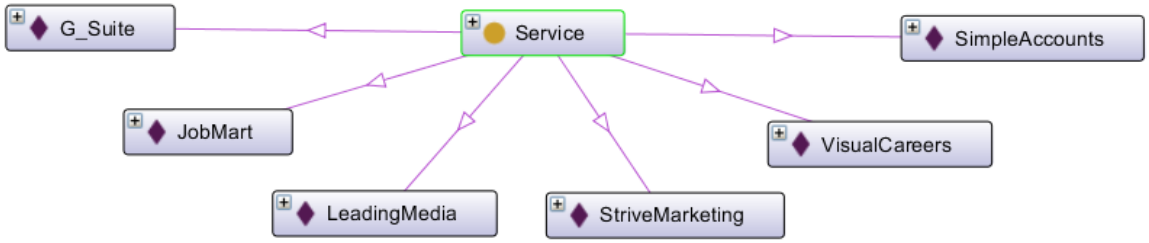


Figure V.5: Instances de la classe Service

V.5 L’implémentation de l’application

La réalisation de notre solution a pris la forme d’une application web pour laquelle nous avons choisi le nom “Courtesy”.

“Courtesy” est un mot anglais qui signifie “Courtoisie” mais c’est également un jeu de mots entre le mot “Courtier” et le mot “Easy” qui veut dire “Facile” en anglais.

Courtier + Easy = Courtesy

Nous avons aussi conçu un logo simple est significatif pour représenter notre application. Notre logo apparaît dans la figure suivante:

Figure V.6: Logo de Courtesy

Pour implémenter notre application web, nous avons utilisé Eclipse comme environnement de développement. Notre application est faite en JavaEE tel que nos fonctionnalités sont codées en Java et nos interfaces sont des JSP qui sont essentiellement écrites en HTML. La liaison entre le code applicatif (Code métier Java) et les vues (JSP) est établie à travers des Servlets et celles-ci sont déclarées dans le fichier web.xml qui est le descripteur de déploiement qui les expose au serveur.

Nous allons maintenant détailler le travail de notre application.

Au début, quand l'utilisateur essaie d'accéder à Courtesy à travers son navigateur web, il atterrit dans la page de présentation du site.

L'utilisateur peut s'inscrire ou se connecter s'il a déjà un compte. S'il n'est pas déjà inscrit, il peut se créer un compte en cliquant sur le bouton "sign up" dans la barre de menu qui apparaît en haut de page. Ce bouton va le rediriger vers l'interface d'inscription où il doit introduire ses informations. L'utilisateur doit également préciser s'il souhaite rejoindre Courtesy en tant que fournisseur ou en tant que client.

L'interface d'inscription est illustrée dans la figure V.7.

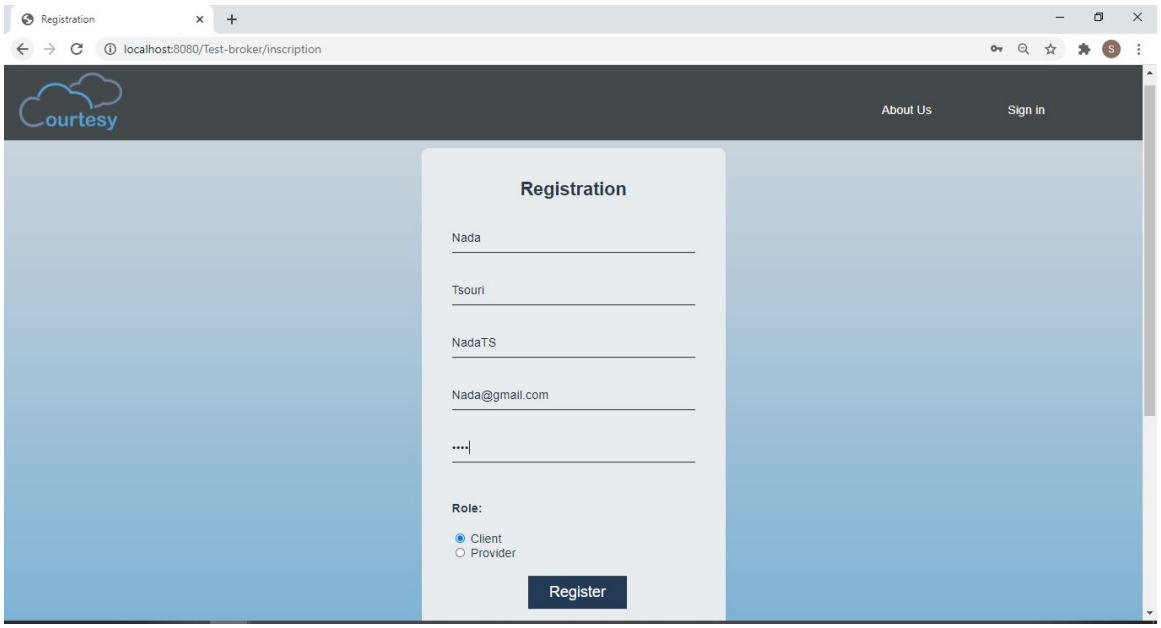
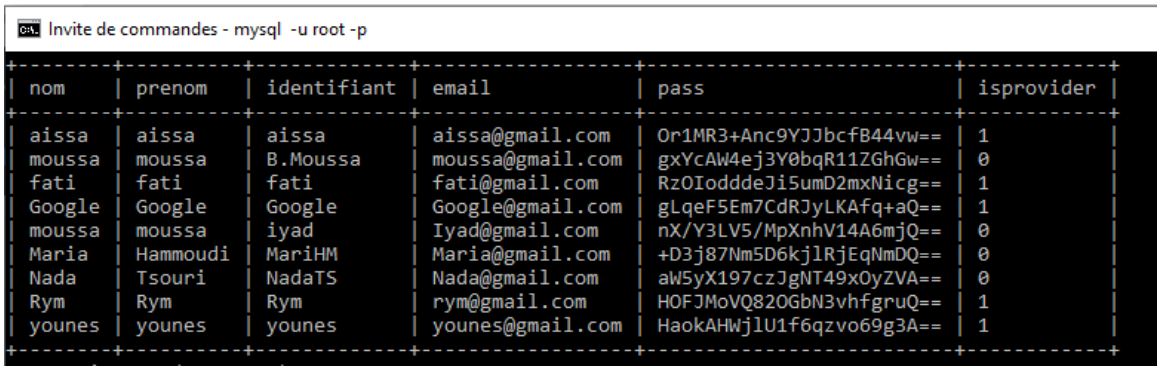


Figure V.7: Interface d'inscription

Les informations des utilisateurs sont gardées dans notre base de données où leurs mots de passe sont sécurisés à travers l'algorithme de cryptage AES [1]. La table des utilisateurs dans notre base de données est illustrée dans la figure V.8.



nom	prenom	identifiant	email	pass	isprovider
aissa	aissa	aissa	aissa@gmail.com	Or1MR3+Anc9YJJbcfB44vw==	1
moussa	moussa	B.Moussa	moussa@gmail.com	gxYcAW4ej3Y0bqR11ZGhGw==	0
fati	fati	fati	fati@gmail.com	RzOIodddeJi5umD2mxNicg==	1
Google	Google	Google	Google@gmail.com	gLqeF5Em7CdRJyLKAfq+aQ==	1
moussa	moussa	iyad	Iyad@gmail.com	nX/Y3LV5/MpXnhV14A6mjQ==	0
Maria	Hammoudi	MariHM	Maria@gmail.com	+D3j87Nm5D6kjlRjEqNmDQ==	0
Nada	Tsouri	NadaTS	Nada@gmail.com	aW5yX197czJgNT49xOyZVA==	0
Rym	Rym	Rym	rym@gmail.com	HOFJMoVQ82OGbN3vhfgruQ==	1
younes	younes	younes	younes@gmail.com	HaokAHWjlU1f6qzvo69g3A==	1

Figure V.8: Table des utilisateurs

Nous avons utilisé le serveur de base de données Mysql. La création de la base de donnée ainsi que le stockage des informations ont été faits à travers des requêtes SQL.

Si l'utilisateur est déjà inscrit, il peut se connecter à son compte en cliquant sur le bouton "Sign in" dans la barre de menu. Il va être redirigé vers l'interface de connexion où il doit introduire son identifiant et son mot de passe.

Un utilisateur connecté peut se déconnecter en cliquant sur le bouton "Sign out" quand il le veut. Dans ce cas, il est renvoyé vers l'interface de présentation de Courtesy.

[1] Advanced Encryption Standard

Il lui est aussi possible de se désabonner de notre système en choisissant l'option "Delete your Courtesy Account", ce qui le renvoi aussi à l'interface de présentation mais il ne lui serait plus possible de se connecter.

V.5.1 Courtier pour fournisseur

Une fois que le fournisseur se connecte à Courtesy, il est directement redirigé vers son tableau de bord où il peut choisir une tâche à effectuer.

Le fournisseur peut publier un nouveau service ou gérer les services qu'il a déjà publié auparavant.

Vu que nous n'avons pas de source de données sur des SaaS réels existants, nous nous sommes rendu sur le site web du service G Suite pour recueillir des informations sur ce dernier afin de l'utiliser comme exemple pour montrer que n'importe quel SaaS peut être publié dans notre registre.

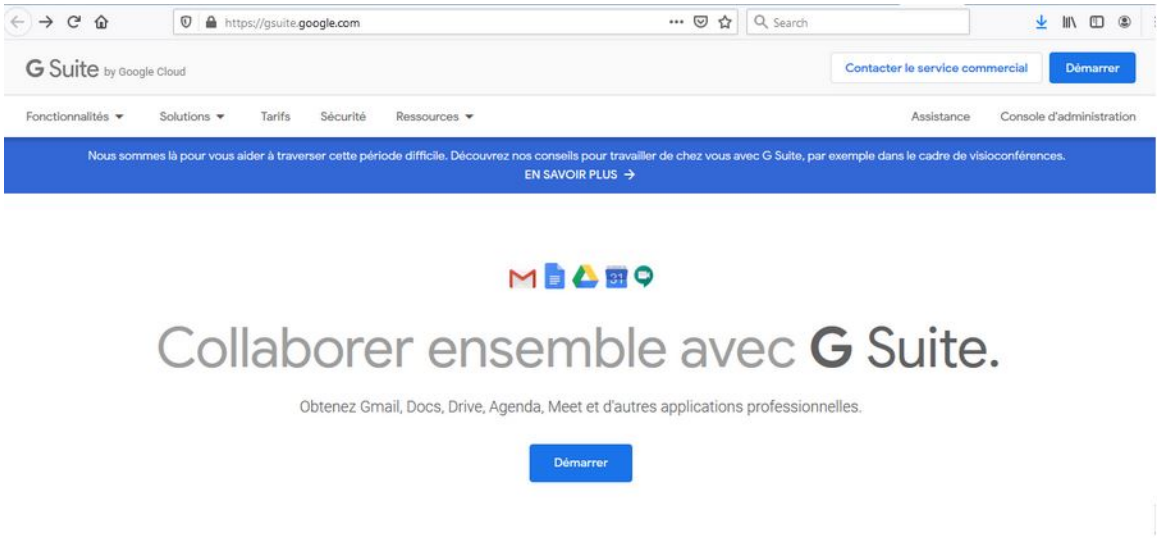


Figure V.9: Site Web du service G Suite de Google

Les figures V.10, V.11, V.12 et V.13 montrent l'interface de publication d'un service où le fournisseur doit introduire les informations relatives au service en question ainsi que les valeurs des critères de qualité de service.

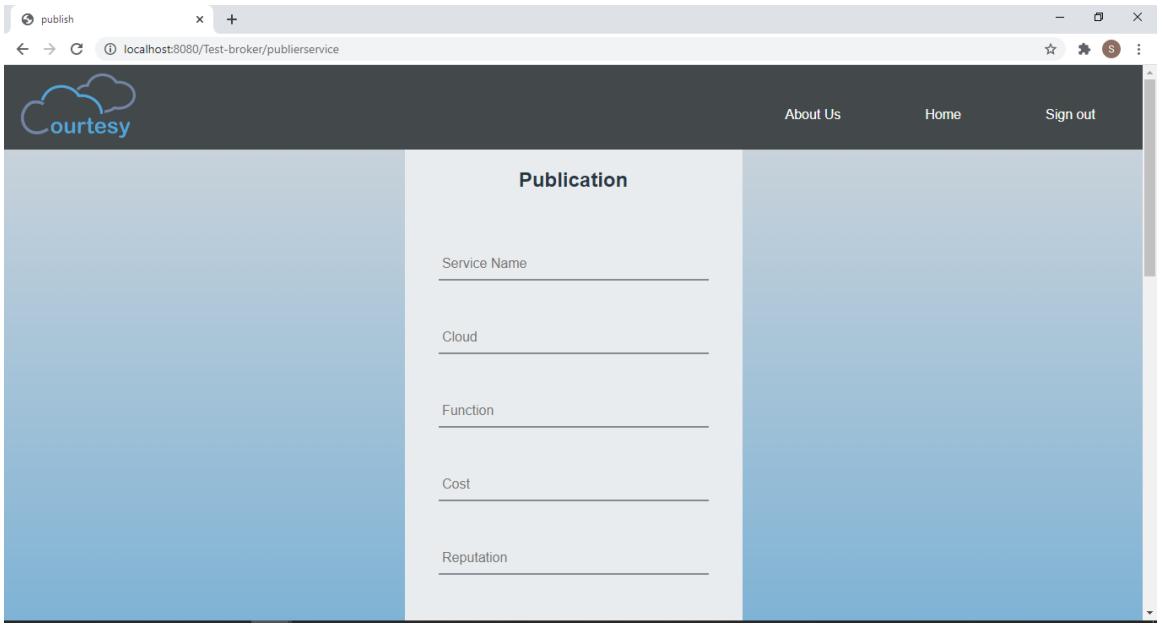


Figure V.10: Interface de publication d'un SaaS (partie1.1)

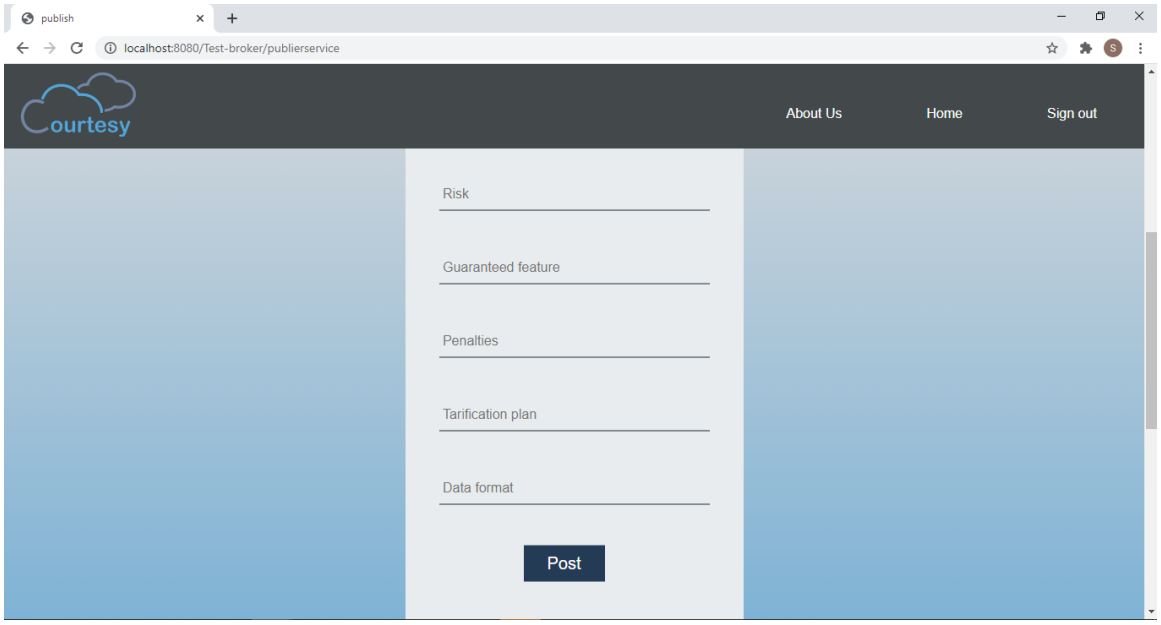


Figure V.11: Interface de publication d'un SaaS (partie1.2)

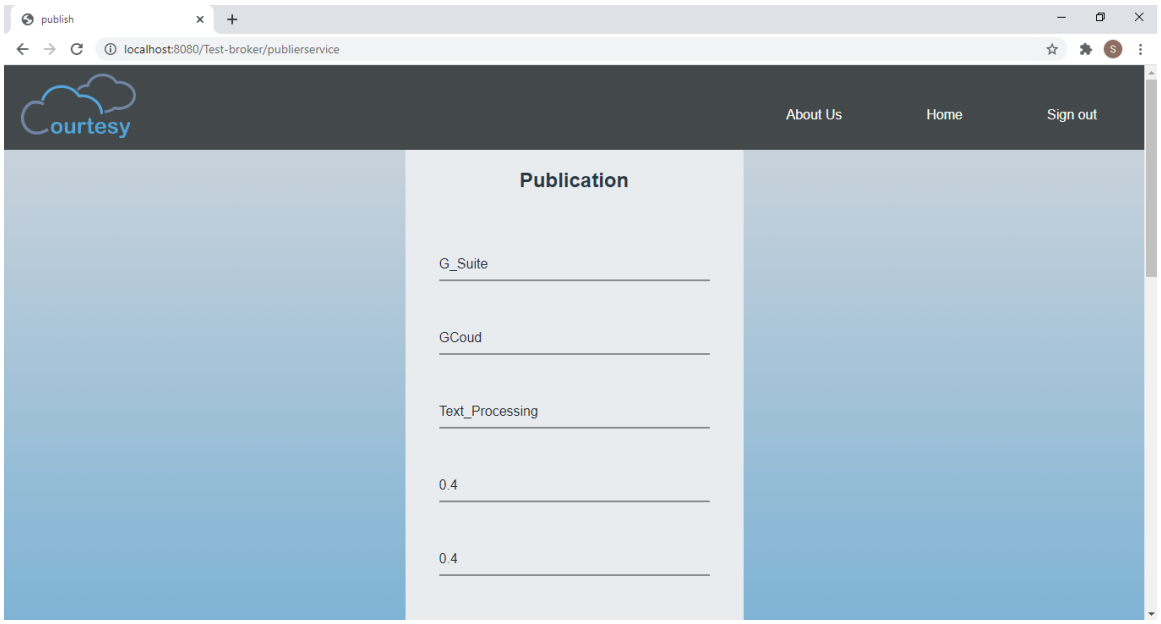


Figure V.12: Interface de publication d'un SaaS (partie2.1)

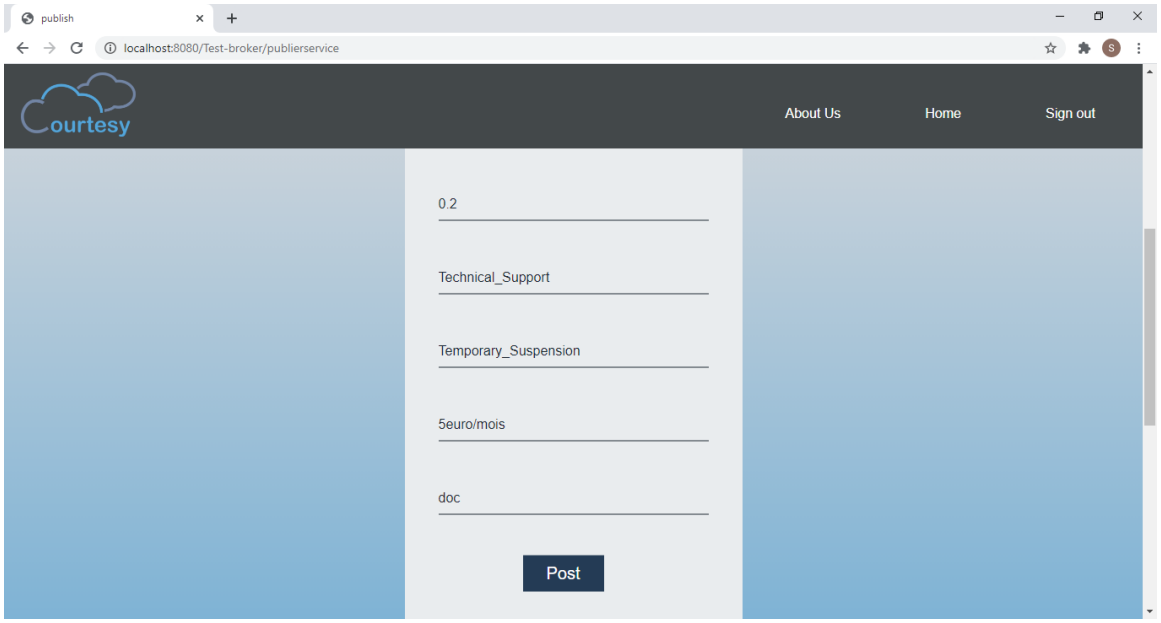


Figure V.13: Interface de publication d'un SaaS (partie2.2)

Lors de la publication d'un service, chaque information relative au service est jointe à la classe qui lui correspond dans l'ontologie (notre registre) sous la forme d'instance de la classe en question, ou "Individual".

Les figures suivantes montrent l'instanciation des informations relatives à la publication de service montrée dans les figures précédentes.

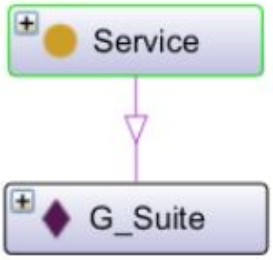


Figure V.14: Exemple d'instance de type service

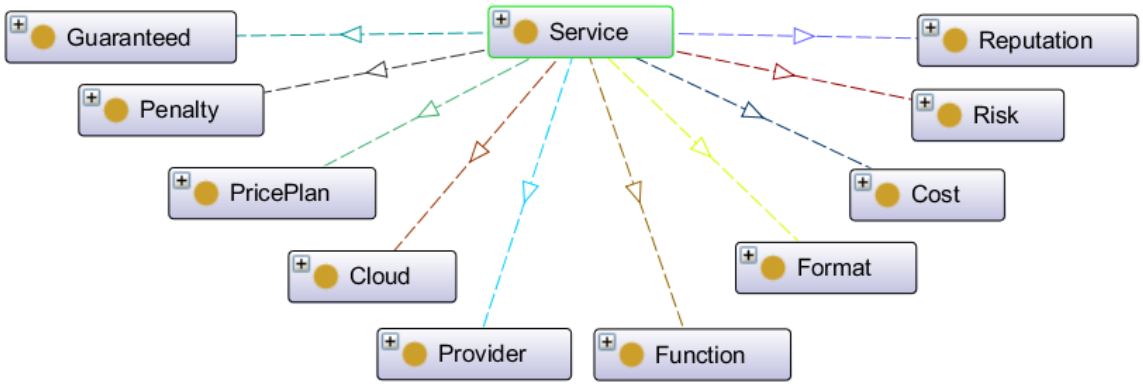


Figure V.15: Concepts relatifs à un service

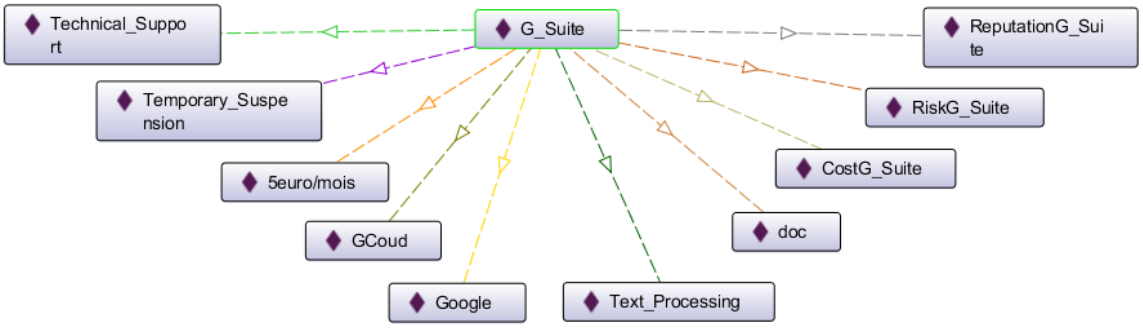


Figure V.16: Instances des concepts relatifs à un service

Notre ontologie est écrite en langage OWL ce qui nous a poussé à utiliser la bibliothèque OWLAPI pour manipuler (Créer, modifier, supprimer) nos instances. Un exemple pris de notre code où nous avons utilisé OWLAPI est montré dans la figure V.17.

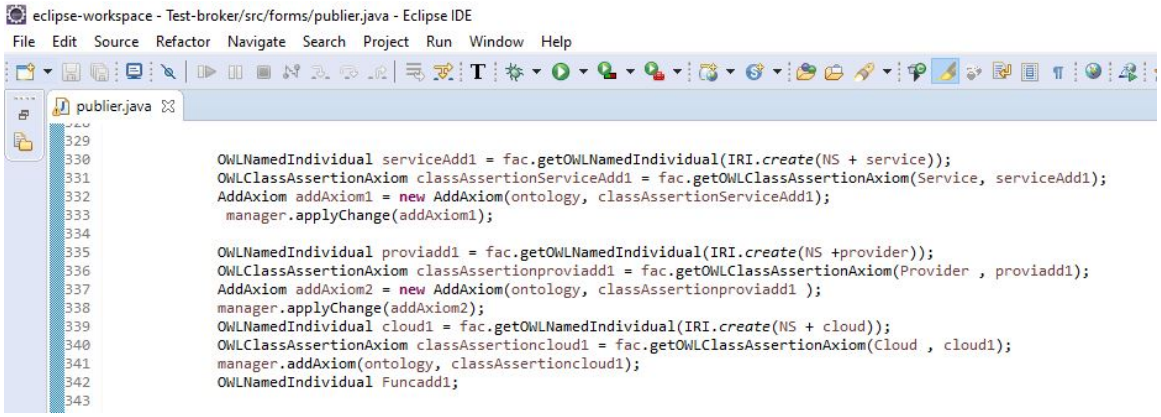


Figure V.17: Code de création des instances

Un fournisseur peut gérer ses services qui sont publiés dans Courtesy où il peut:

- Supprimer un service en donnant son nom.
- Renommer un service en donnant le nom du service et son nouveau nom.

Le fournisseur peut également supprimer son compte comme nous l'avons déjà mentionné. Lorsqu'un fournisseur se désabonne du service courtier, tous ses services sont supprimés du registre. Cette suppression est également faite à travers OWLAPI.

V.5.2 Courtier pour client

Lorsque le client se connecte, il est directement redirigé vers son tableau de bord où il peut choisir une tâche à effectuer. Il peut chercher un service qu'il souhaite utiliser ou changer son nom d'utilisateur.
Il peut également supprimer son compte comme nous l'avons déjà mentionné.

- **Requête simple:**
  La figure V.18 montre l'interface de recherche d'un service où le client doit introduire la requête précisant le service désiré et les poids de priorité qu'il doit accorder à chaque critère de qualité de service. Nous rappelons que les poids de priorité sont utilisé par notre système pour choisir le meilleur service suivant les préférences du client.

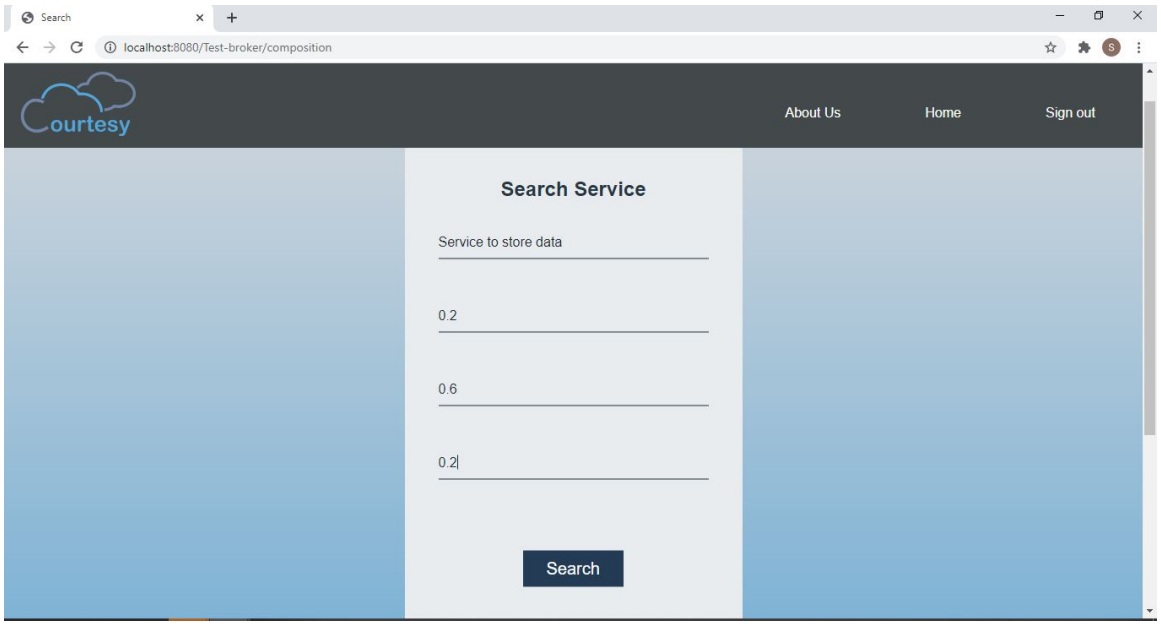


Figure V.18: Interface de recherche

Lorsque sa requête est traitée, un message lui est affiché pour lui dire que la recherche a été effectuée avec succès et qu'il peut voir le résultat de sa requête en défilant en bas, comme on peut le voir dans la figure V.19.

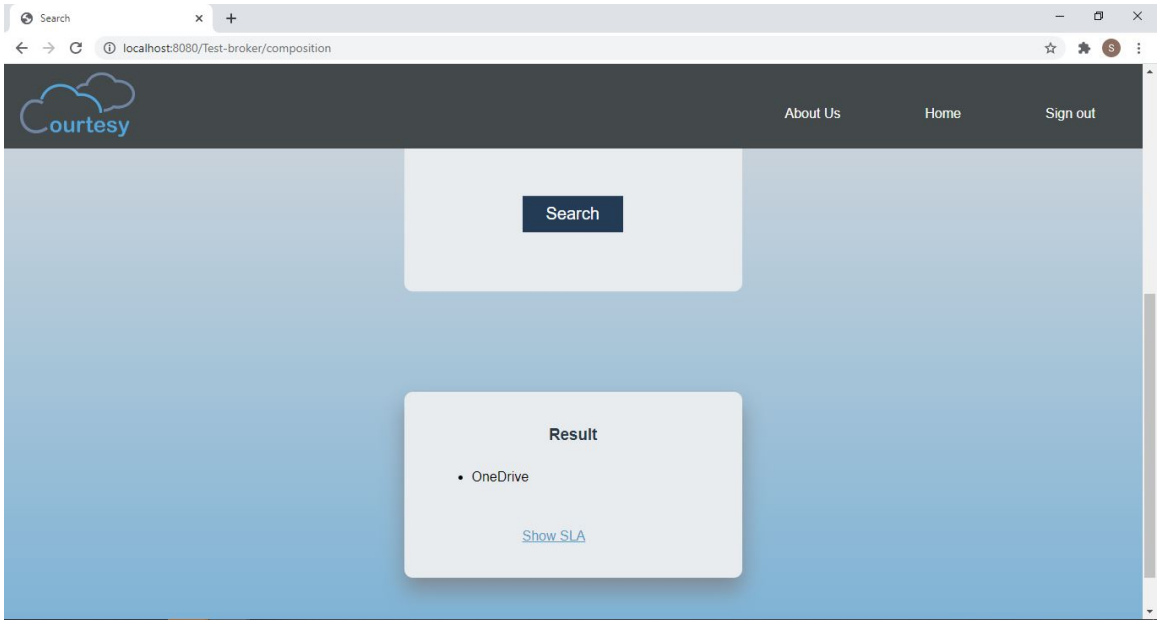


Figure V.19: Résultat de la recherche

La récupération du service souhaité se fait à travers des requêtes SPARQL que l'on a pu effectuer en utilisant l'API Jena qui manipule des données écrite en langage RDF. La figure suivante montre la requête de récupération de service écrite en SPARQL.

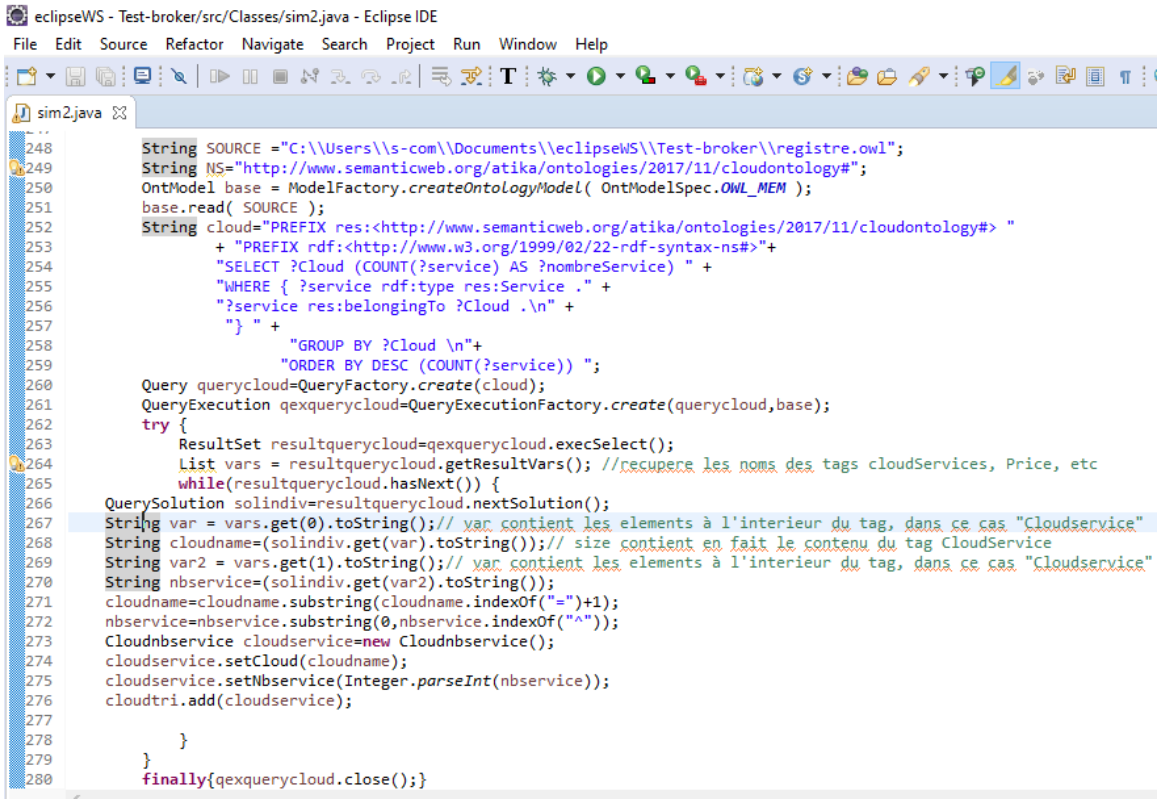


Figure V.20: Exemple d'une requête SPARQL

Nous rappelons que nous avons mentionné dans le chapitre précédent que nous récupérons tous les synonymes des mots clés de la requête. Ceci est fait en utilisant la base de donnée lexicale WordNet par le biais des bibliothèques JAWS API et WS4J API.

- **Requête de composition:**

  Dans certains cas, la requête du client ne peut pas être satisfaite par un seul service. Dans ce cas, notre application effectue une composition de services existants pour retourner un service composé au client. Un cas pareil est illustré dans les figures V.21 et V.22.

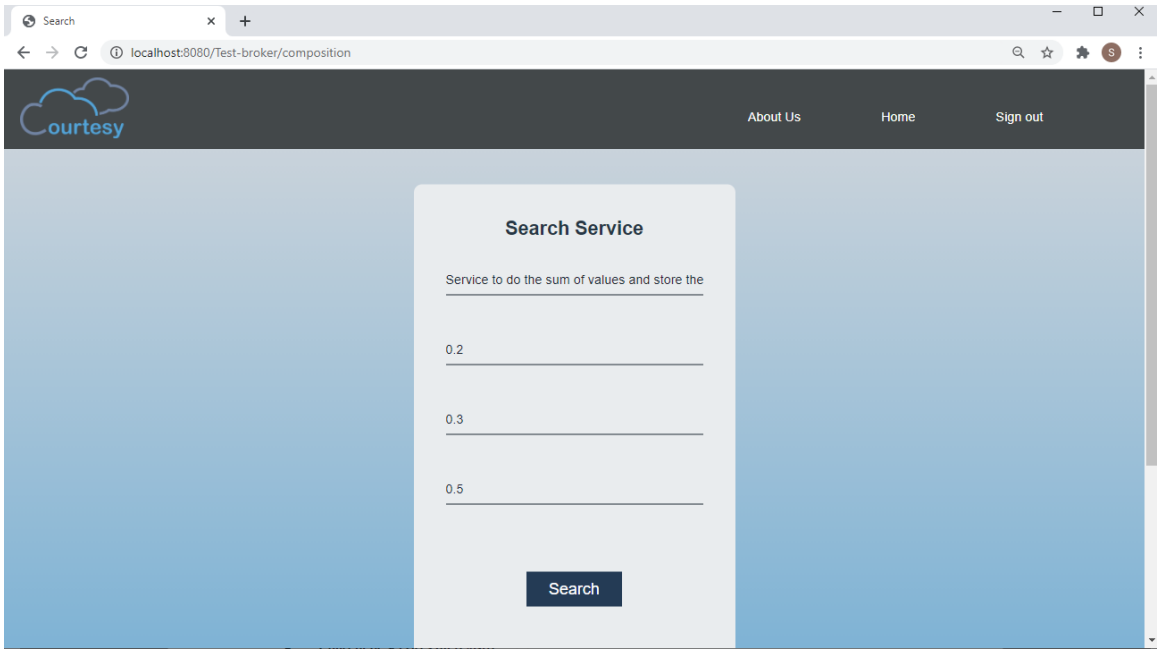


Figure V.21: Interface de recherche d'un service composé

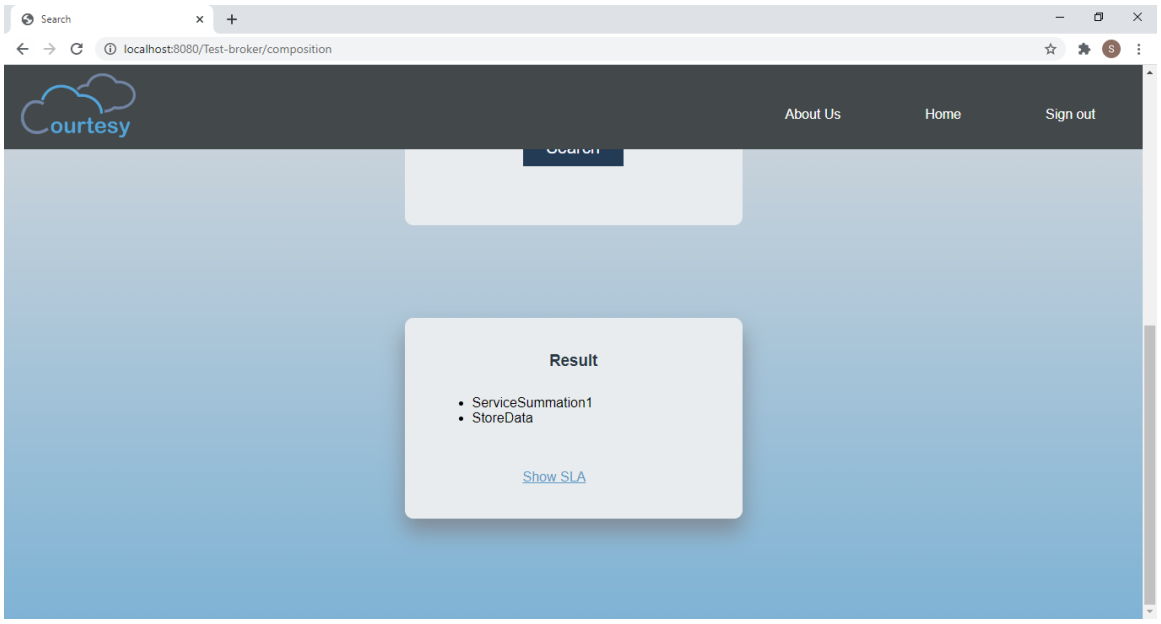


Figure V.22: Résultat de recherche d'un SaaS composé

Si le client clique sur "Show SLA", il sera redirigé vers l'interface qui affiche le contrat SLA et ses détails tels que les services concernés, leurs critères de qualité de service, les responsabilités des parties prenantes, la tarification, les garanties et les pénalités.

Pour les services composites,les pénalités et garanties sont représentées par la combinaison des garanties et

des pénalités de chaque service composant. En ce qui concerne la tarification, il s'agit de la somme des tarifs des services composants, tandis que Courtesy bénéficie de 10% de la somme des tarifs, ce qui signifie que la tarification totale devient la somme des tarifs des services entrant dans la composition plus la part de Courtesy.

Les figures suivantes montrent le détail du SLA d'un service composé de deux services.

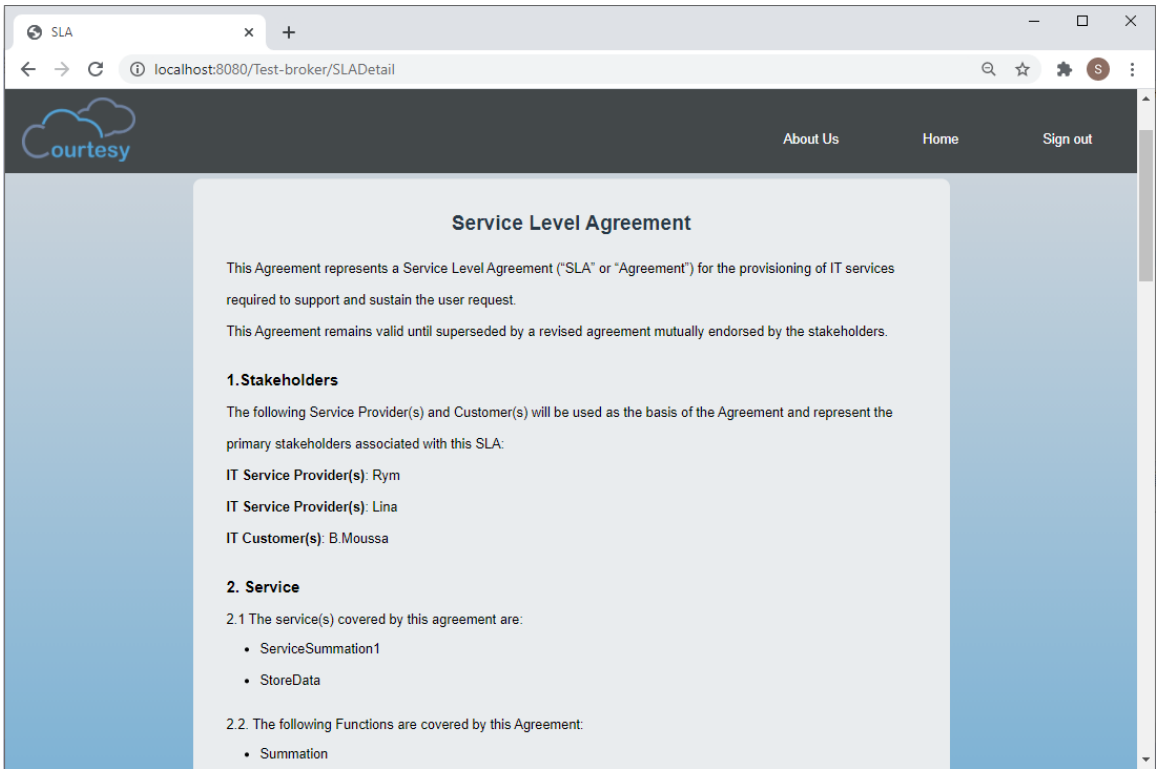


Figure V.23: SLA partie 1

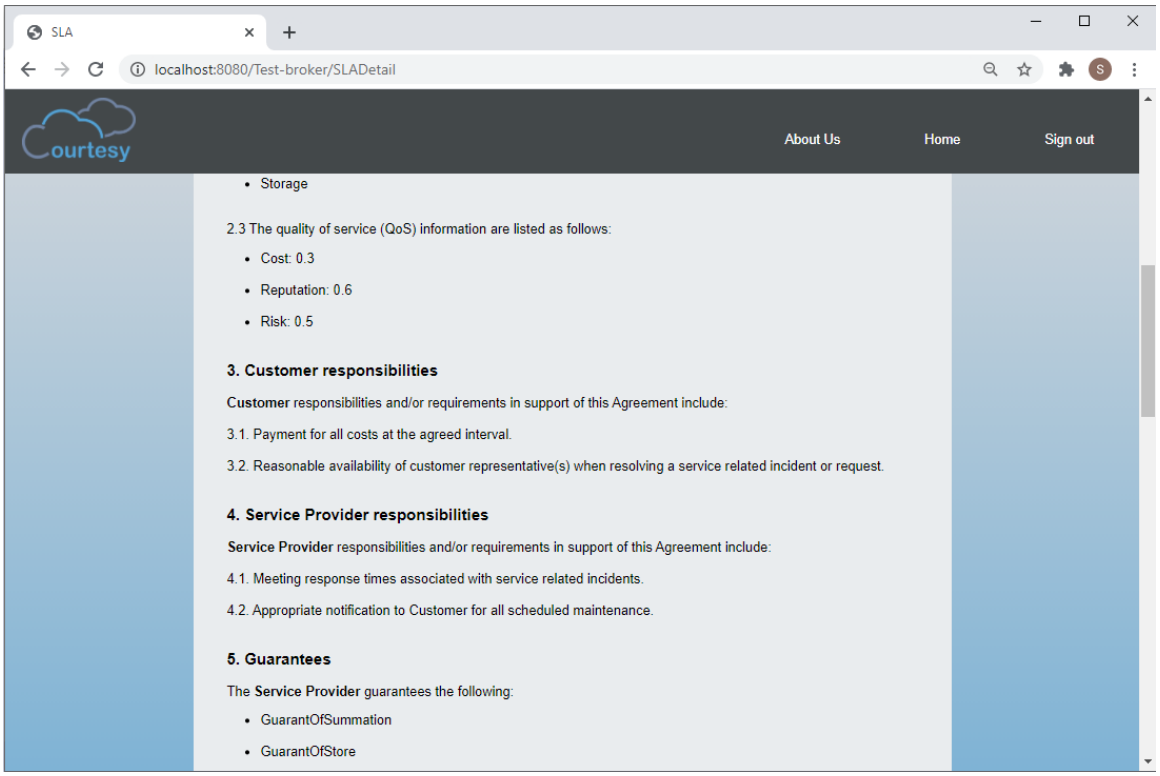


Figure V.24: SLA partie 2

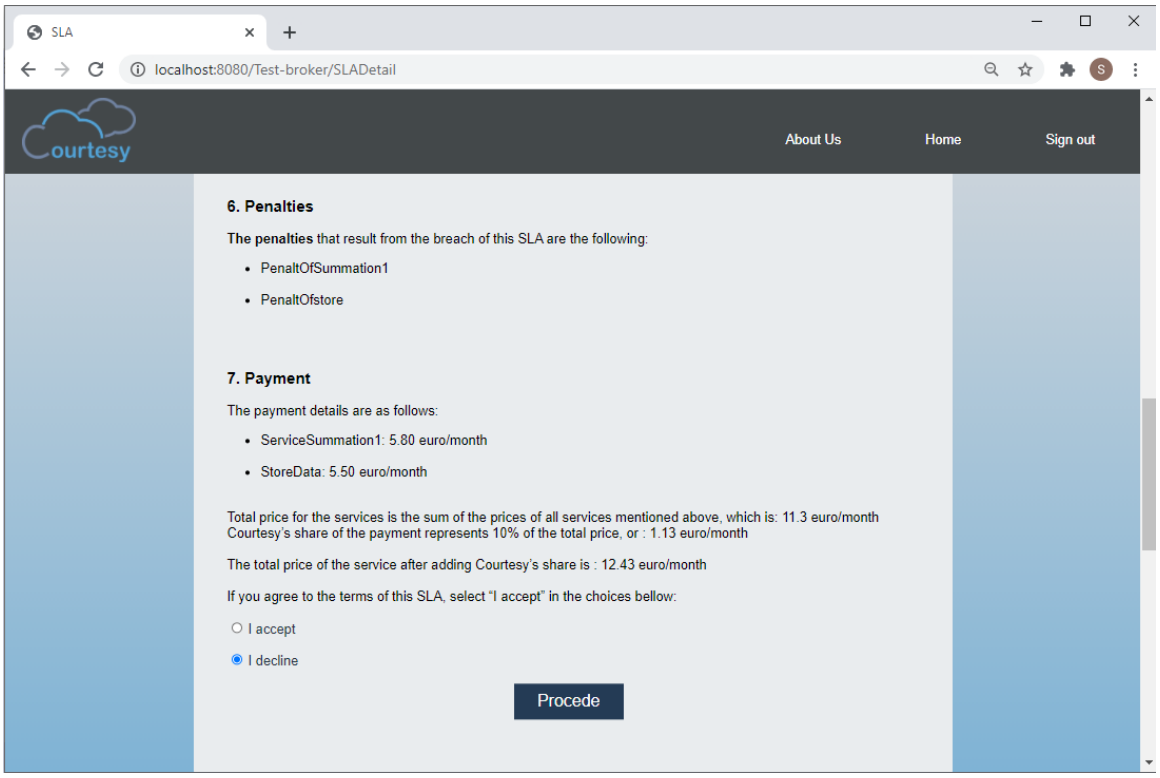


Figure V.25: SLA partie 3

Un client peut changer son nom d'utilisateur. L'utilisateur doit se déconnecter et se reconnecter pour pouvoir observer les changements effectués car son ancien nom est gardé jusqu'à la fin de la session.
Comme il en est le cas pour la création et la suppression, cette modification est faite à l'aide de la bibliothèque OWLAPI.

V.5.3 Module d'adaptation de données

La composition de services implique un échange de données. Ces données n'ont pas forcément le même format et ceci nécessite un traitement d'adaptation pour permettre l'utilisation de ces données.

Pour montrer cette partie de notre travail, et vu que nous n'avons pas d'accès à des SaaS réels, nous nous sommes basées sur la suppositions qu'il y ait deux services composés pour répondre aux besoins du client tel que le premier SaaS exécute et génère des données de sortie au format SQL et le second nécessite des données d'entrée au format JSON.

Comme mentionné dans le chapitre précédent, notre solution est basée sur le concept de format pivot, dans notre cas, il s'agit du format XML que nous avons choisi à cause de sa grande popularité et au fait qu'il soit un format de données standard. Notre solution effectue donc une conversion du format des données du premier SaaS qui est en SQL au format XML, puis une autre conversion du format XML au format JSON.
La figure V.26 montre le code pour convertir les données SQL en données XML.

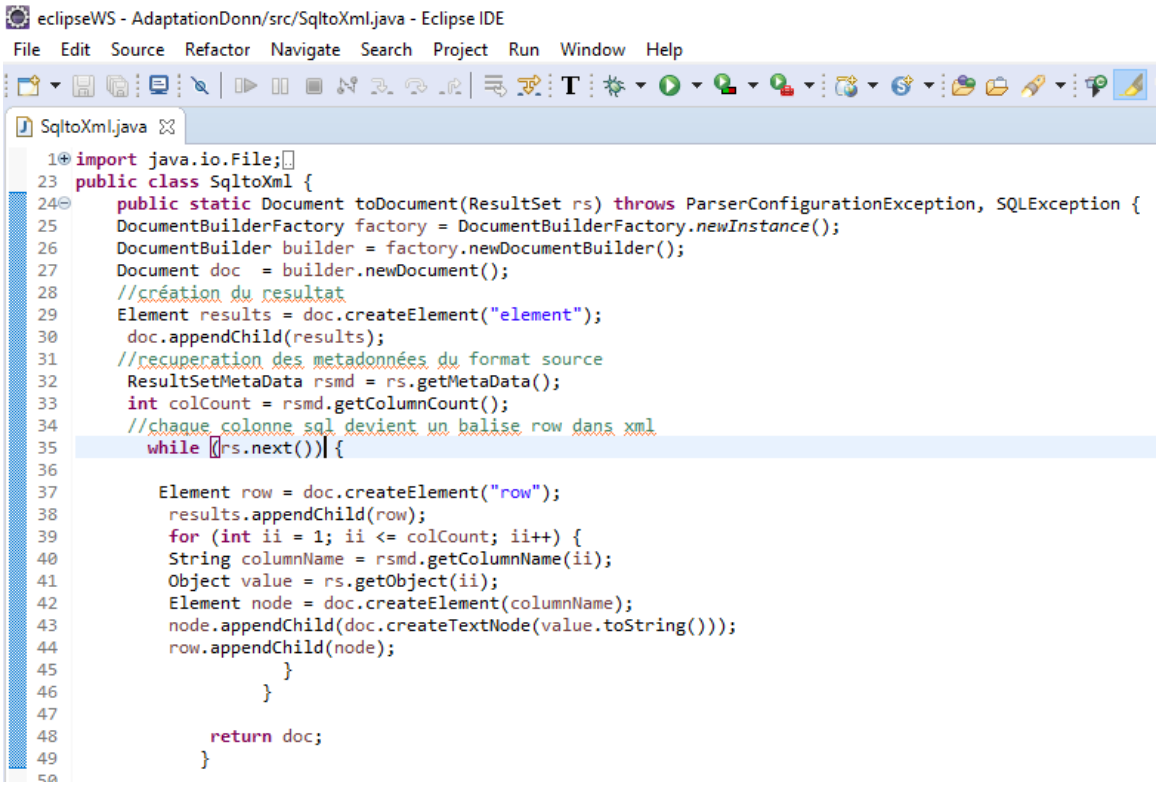


Figure V.26: Code de conversion de SQL en XML

La figure ci-dessous montre le résultat de cette conversion

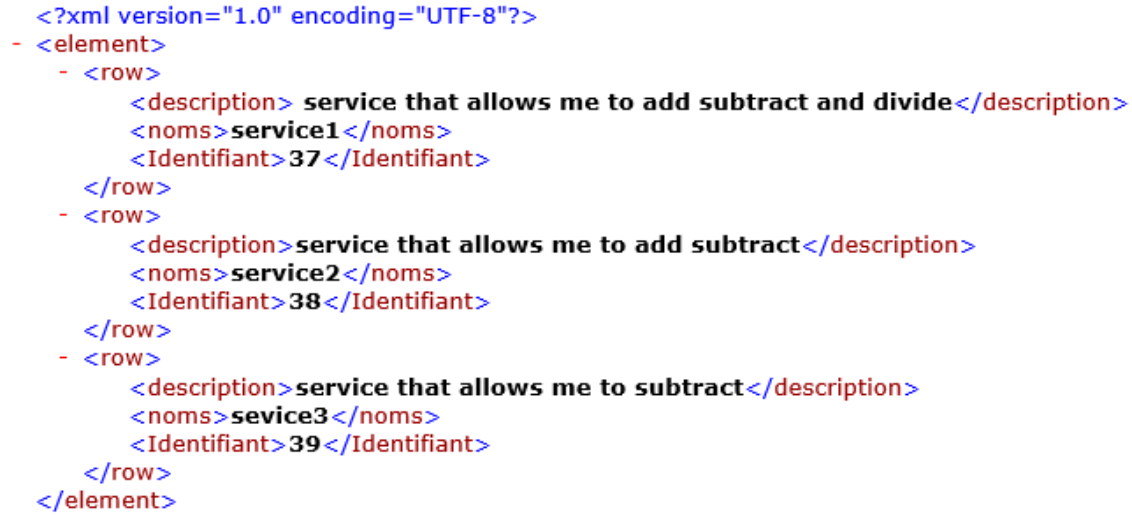


Figure V.27: Résultat de conversion de SQL en XML

La figure V.28 montre le code pour convertir les données XML en données sous format JSON.

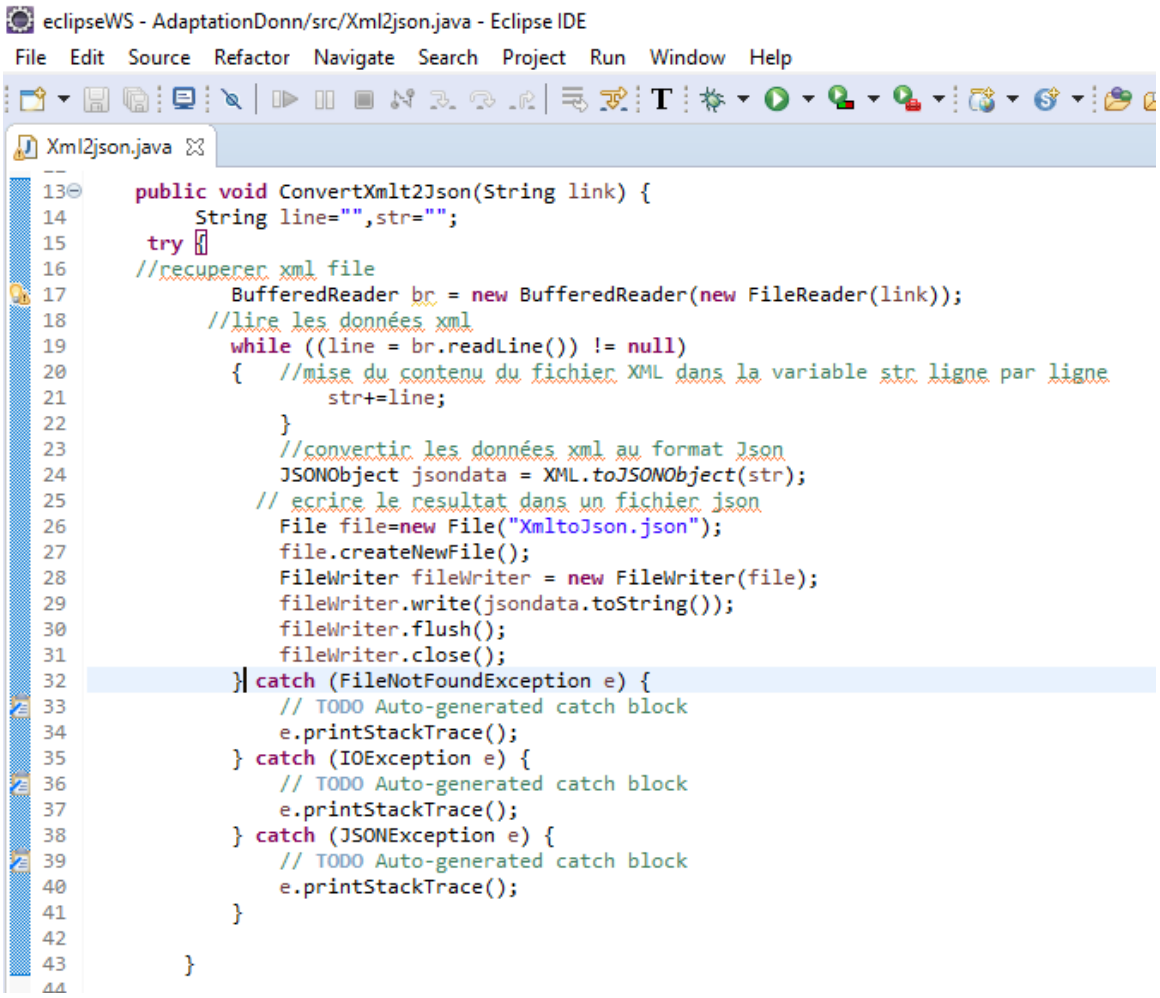


Figure V.28: Code de conversion de XML en JSON

La figure suivante présente le résultat de cette conversion.


```
{"element":{"row":[{"noms":"service1","description":"service
that allows me to add subtract and
divide","Identifiant":37},{"noms":"service2","description":"se
rvice that allows me to add
subtract","Identifiant":38},{"noms":"sevice3","description":"s
ervice that allows me to subtract","Identifiant":39}]}}
```


Figure V.29: Résultat de conversion de XML en JSON

V.6 Évaluation de la solution par rapport à l'état de l'art

Pour évaluer notre solution, nous avons décidé de la comparer avec les solutions traitant le problème de l'interopérabilité dans les SaaS que nous avons étudié dans le chapitre III selon les critères d'évaluation utilisés dans l'étude comparative. Ceci est la seule façon qui nous permet de montrer notre apport car il n'existe pas de benchmarks relatifs à notre problématique.
Cette évaluation est présentée dans le tableau ci-dessous:

	Description	**Composition**	**QoS**	**Environnement**	**Aspect sémantique**	**Requête utilisateur**	**Adaptation de données**
Courtesy	Ontologie	Composition réactive	Sélection selon les QoS	Multi-cloud	Traité	Traitée	Traitée

Tableau V.1: Évaluation de notre solution

V.7 Déploiement de l'application dans un conteneur

Pour rappel, notre application est censée être un courtier en tant que service. Aucune des solutions existantes n'est présentée en tant que tel.

Notre solution est mise à disposition en tant que service pour régler les problèmes de latence et pour assurer un bon temps de réponse et par conséquent, une bonne qualité de service.

Vu que la recherche d'un service est la fonctionnalité la plus affectée par le temps de réponse, nous avons choisi cette partie de notre application pour montrer comment se fait le déploiement.

Pour déployer notre application, nous avons utilisé l'outil de conteneurisation Docker.
Nous avons commencé par installer Docker Desktop sur notre système. Puis, nous avons exporté notre application autant qu'un fichier WAR dans Eclipse.

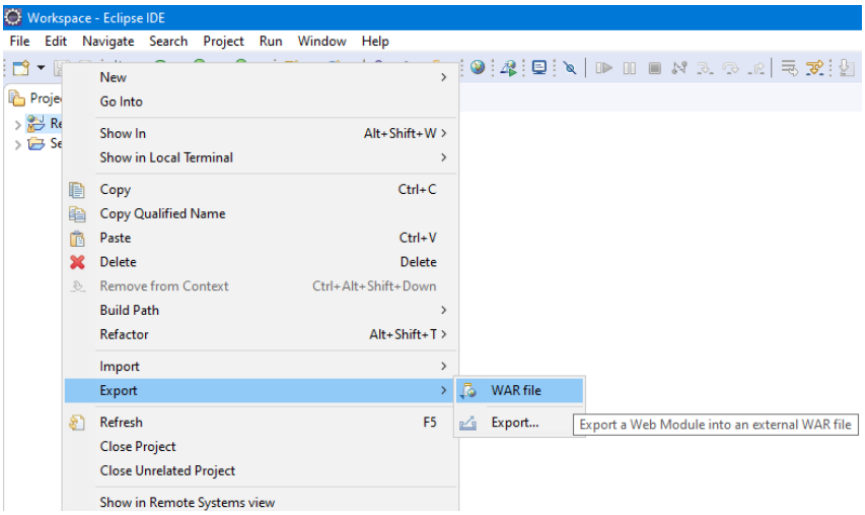


Figure V.30: Exportation de l'application

Nous avons sélectionné le dossier où nous voulons avoir notre fichier WAR, dans notre cas, il s'agit du dossier target qui se trouve à la racine de notre projet dans l'espace de travail Eclipse.
Ensuite, nous avons créé le Dockerfile qui contient toutes les instructions nécessaires à la création de l'image de notre application. Pour ce faire, nous avons simplement créé un nouveau fichier texte sans extension dans la racine du projet que nous avons nommé **Dockerfile**. Le Dockerfile contient les instructions montrées dans la figure suivante:

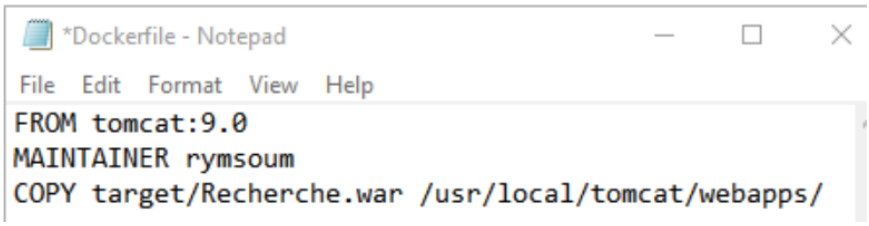


Figure V.31: Instruction Dockerfile

FROM tomcat:9.0 signifie que nous voulons construire notre image sur l'image officielle de Tomcat. Docker va d'abord chercher cette image en local puis, s'il ne la trouve pas, il va la chercher auprès du registre Docker Hub et la télécharger pour l'utiliser en tant que serveur.
MAINTAINER désigne le propriétaire de l'image, dans ce cas c'est "rymsoum".
COPY target/Recherche.war /usr/local/tomcat/webapps/ est une instruction qui met le fichier WAR qui est dans le dossier target dans le dossier webapps du serveur Tomcat. C'est là où l'application va être extraite et déployée sur Tomcat.

Une fois que le Dockerfile est prêt, nous pouvons l'utiliser pour construire une image. Ceci est simplement fait à travers une interface en ligne de commande ouverte autant qu'administrateur.

D'abord, il faut se positionner dans le chemin du dossier de l'application en utilisant la commande **cd C:\Users\Rym\Documents\Workspace\Recherche** comme la figure suivante le montre:

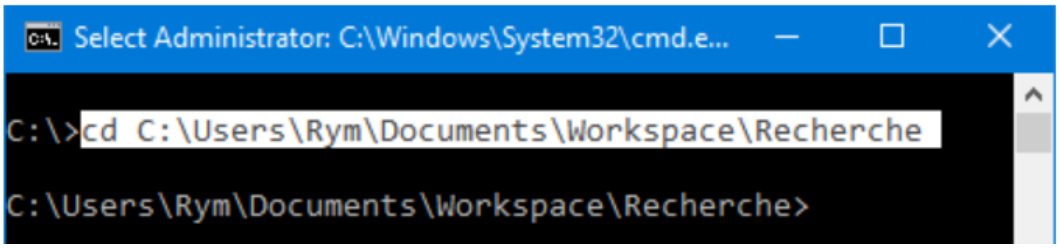


Figure V.32: Chemin de l'application

Puis, nous utilisons la commande **docker image build -t rymsoum/recherche ./** pour construire l'image de notre application. Ici, **rymsoum/recherche** est le nom de l'image. Nous rappelons que Docker est en marche et qu'il va utiliser le Dockerfile qui est dans le répertoire courant pour construire l'image.

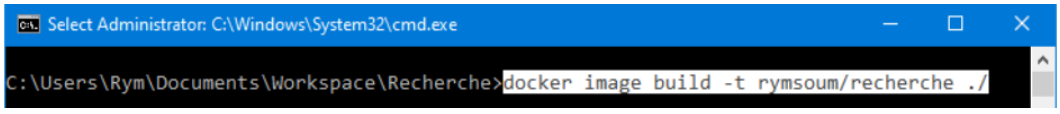


Figure V.33: Construction d'image

Nous pouvons ensuite voir la progression de la construction de l'image. Chaque instruction du Dockerfile est représentée par une **Step** et une fois que toutes les instructions sont exécutées avec succès, nous pouvons passer à l'étape suivante.

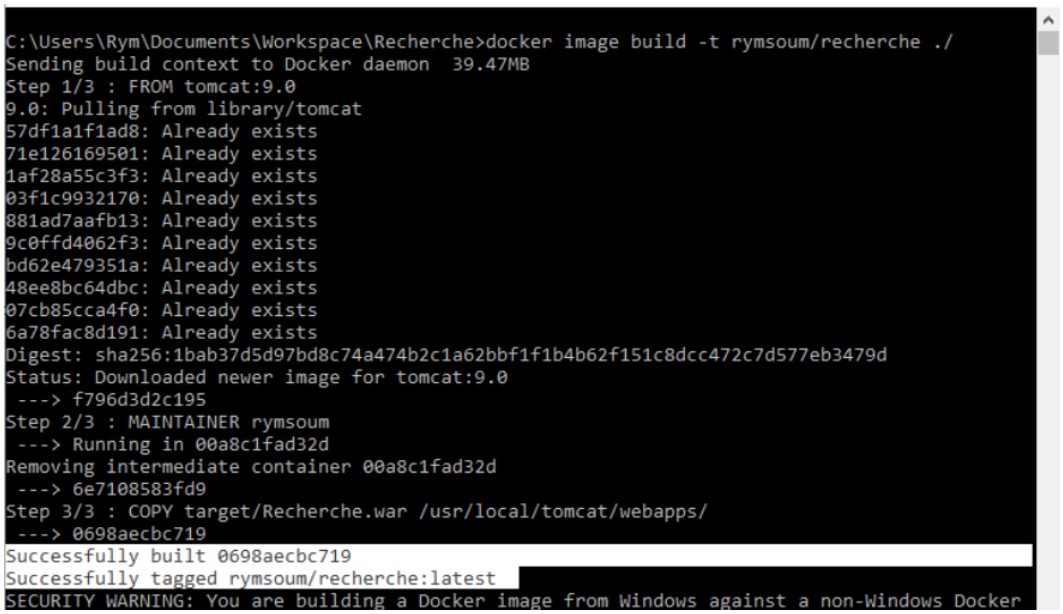


Figure V.34: Progression de construction de d'image

Il nous est possible de vérifier que l'image a bien été construite en utilisant la commande **docker image ls** qui nous affiche toutes les images existantes comme le montre la figure V.35.

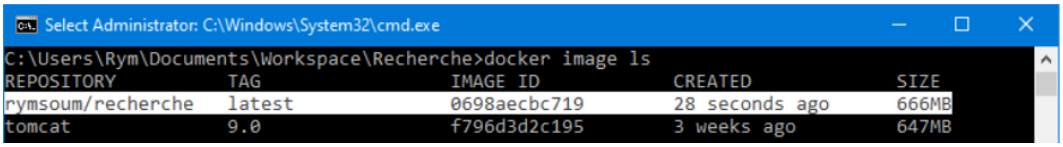


Figure V.35: Affichage des images existantes

Maintenant que nous avons une image de notre application, nous pouvons la publier sur le Docker Hub. Il faut d'abord se créer un compte en se rendant sur le site : https://hub.docker.com/ .

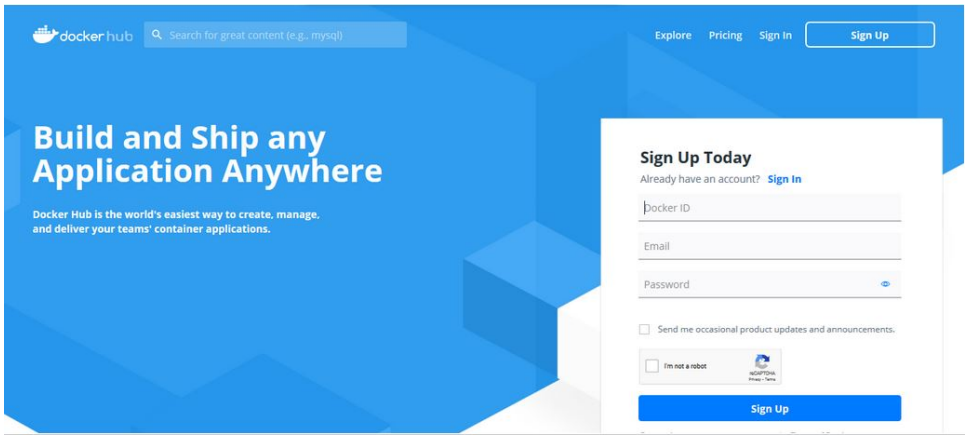


Figure V.36: Page d'accueil de Docker Hub

Puis, il faut se connecter pour pouvoir voir nos images.

Nous n'avons pas encore publié notre image, donc il n'y a aucun répertoire pour l'instant.

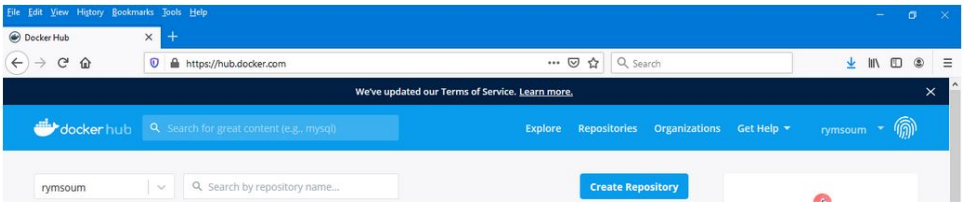


Figure V.37: Dockerhub sans répertoire

Nous devons également être connectés au compte créé depuis Docker Desktop comme le montre la figure suivante:

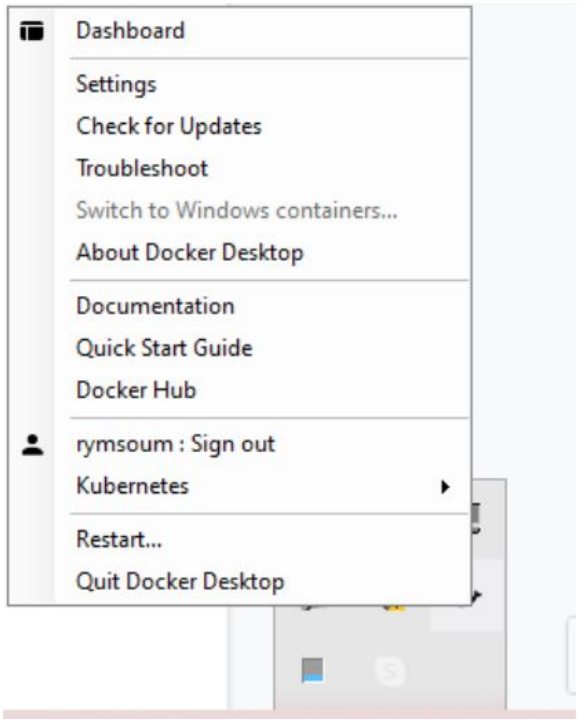


Figure V.38: Connexion au compte Docker Hub

Maintenant, nous allons publier l'image en utilisant la commande **docker push rymsoum/recherche** comme le montre la figure V.39.

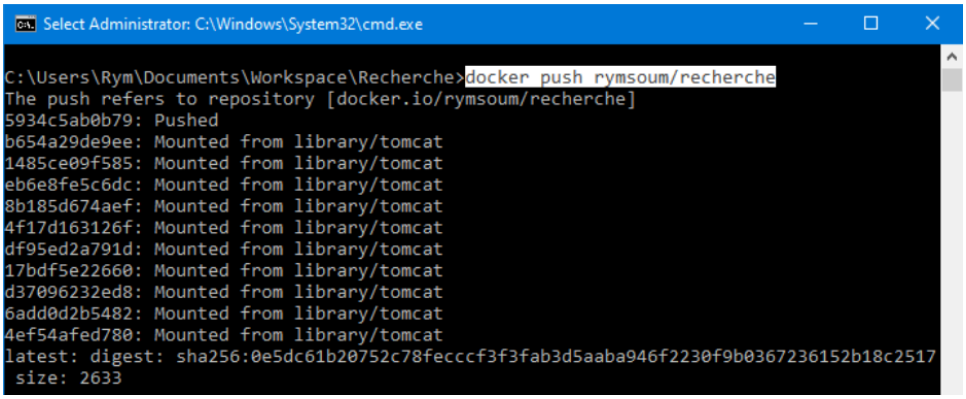


Figure V.39: Publication d'image

Une fois la publication finie, on obtient un message comme suite:

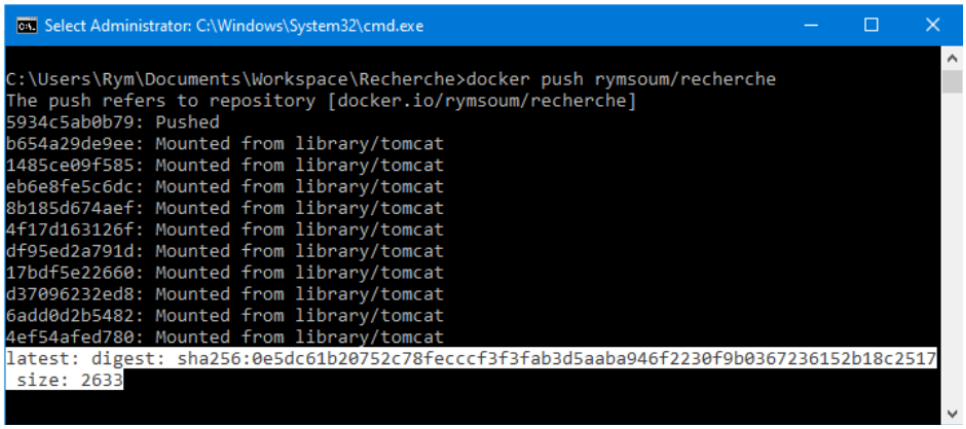


Figure V.40: Fin de publication d'image

Si on se rend maintenant sur le compte Docker Hub, on peut voir que l'image a bien été publiée.

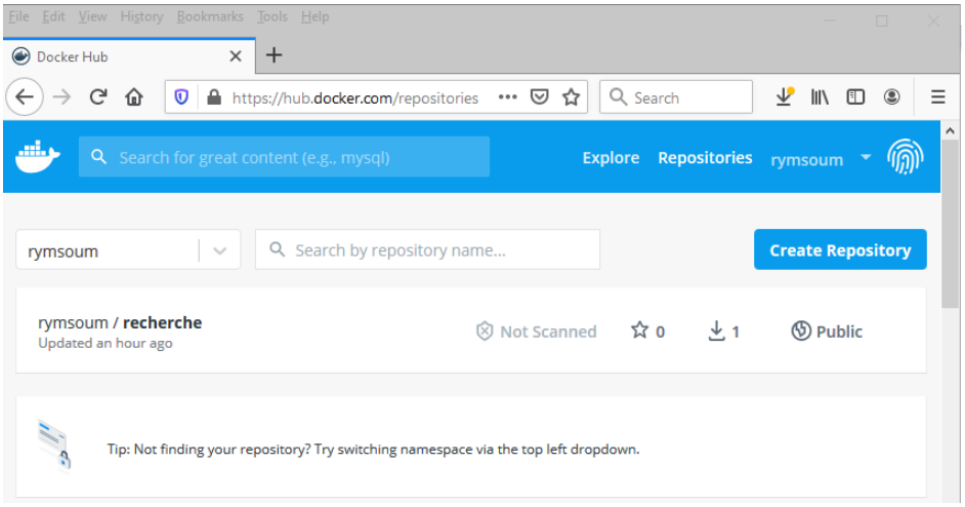


Figure V.41: Image dans Docker Hub

Il est maintenant possible de récupérer cette image depuis d'autres hôtes et de l'utiliser pour en faire un conteneur. Pour montrer cela, nous avons utilisé un autre ordinateur où Docker est déjà installé et est en marche. à travers l'invite de commande, nous utilisons la commande **docker pull rymsoum/recherche** pour récupérer l'image depuis le Docker Hub.

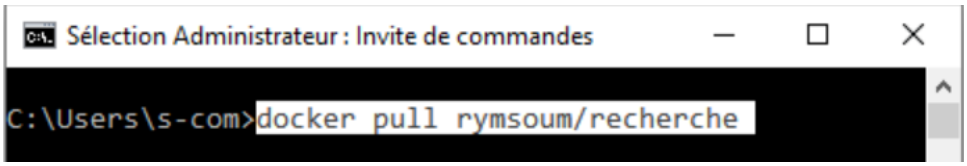


Figure V.42: Récupération d'image

À la fin, nous obtenons un message qui signale que l'image a bien été récupérée.

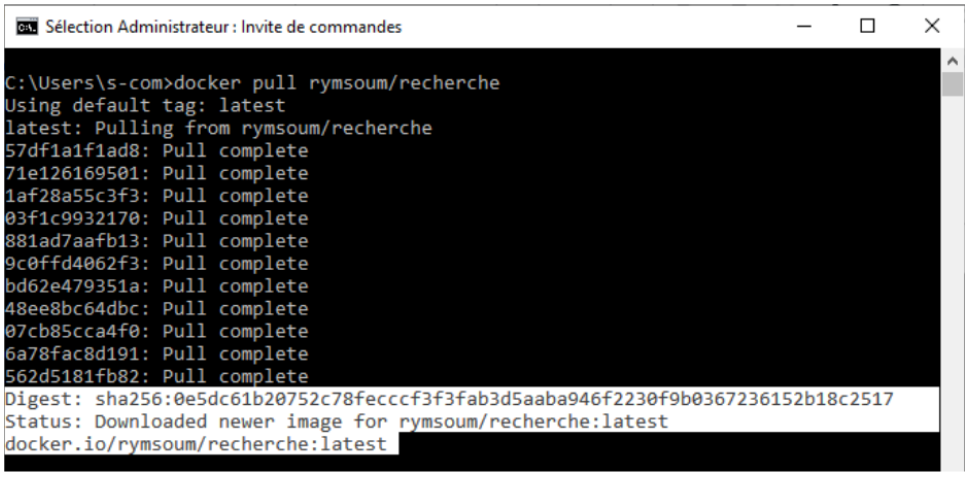


Figure V.43: Fin de récupération d'image

Nous pouvons également vérifier son existence en utilisant la commande **docker image ls**.

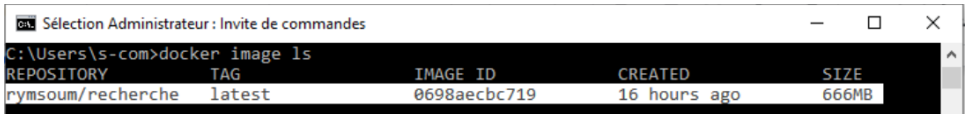


Figure V.44: Vérification de l'existance de l'image

Maintenant, nous pouvons exécuter la commande **docker container run -it –publish 8686:8080 rymsoum/recherche** pour créer un conteneur à partir de l'image. Nous précisons que **–publish 8686:8080** est utilisée pour exposer le port 8080 du conteneur sur le port 8686 de la machine hôte.

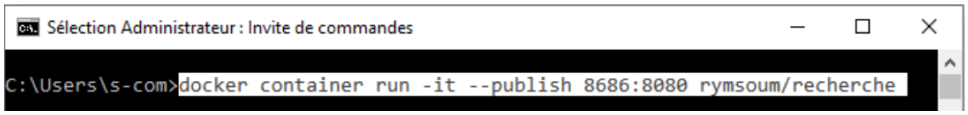


Figure V.45: Création d'un conteneur

Le lancement du serveur Tomcat commence comme le montre la figure suivante:

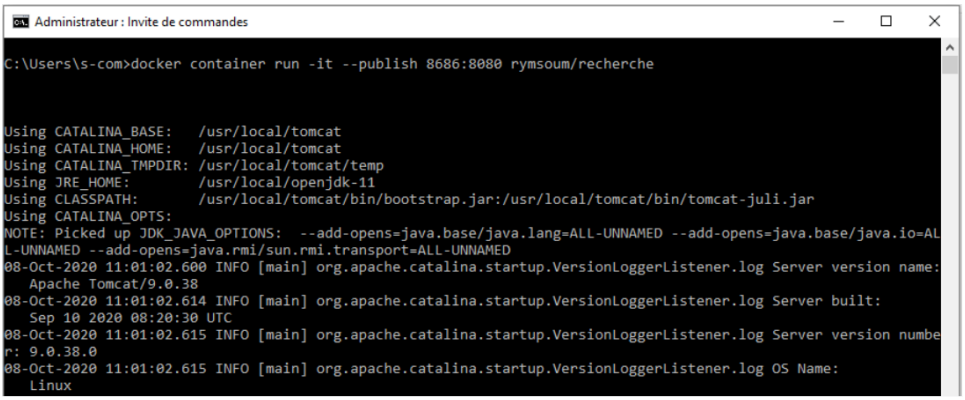


Figure V.46: Démarrage du Tomcat

Si on se rend sur l'adresse https://localhost:8686/Recherche/composition qui est l'adresse de la page de recherche, nous pouvons voir que notre application fonctionne parfaitement.

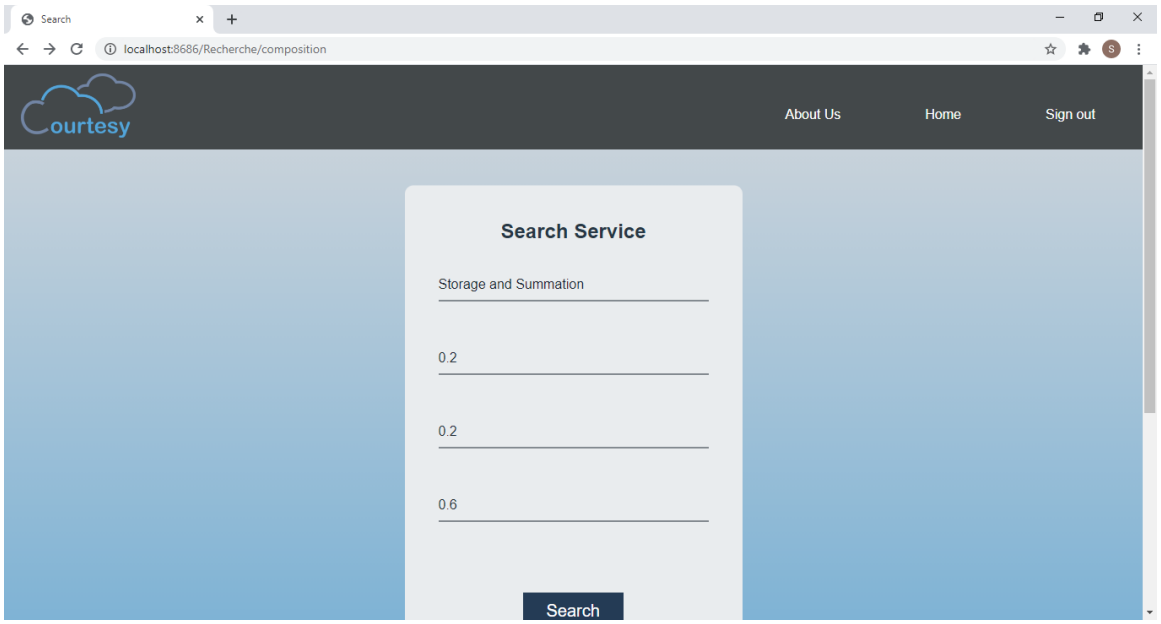


Figure V.47: Recherche d'un SaaS

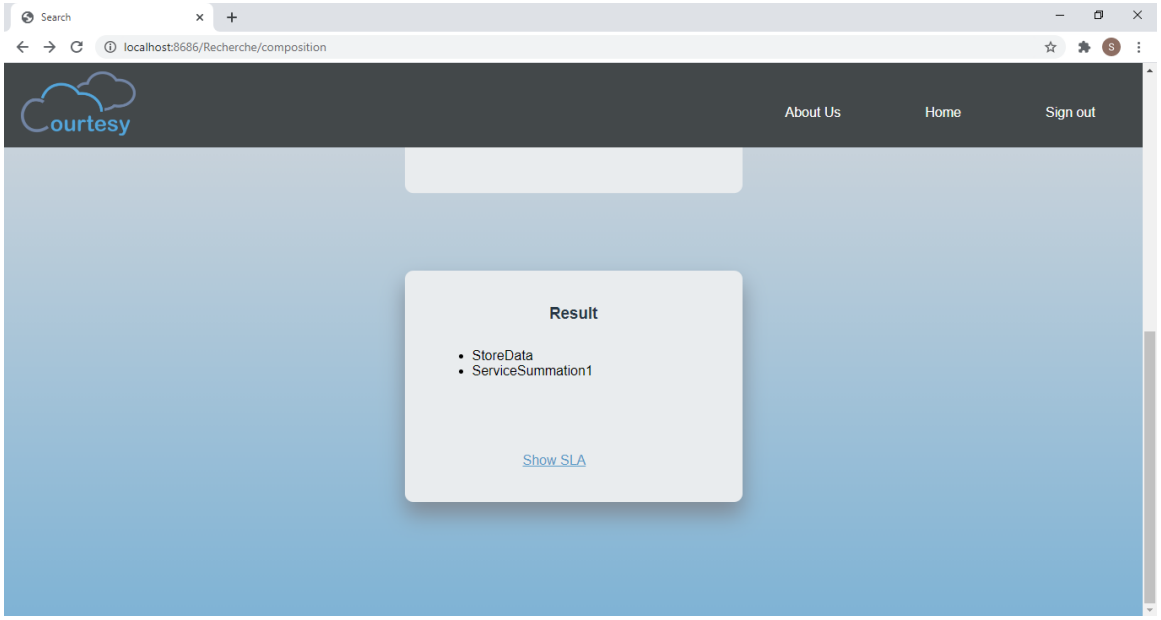


Figure V.48: Résultat de recherche

Comme nous venons de le montrer, le déploiement de l'application peut être facilement effectué sur n'importe quelle machine du moment que Docker est installé sur cette machine. Ceci constitue un avantage important de notre solution car ainsi, il est possible d'en mettre plusieurs instances à disposition dans plusieurs serveurs à travers le monde ce qui permet une meilleur disponibilité et un temps de réponse optimal.

V.8 Conclusion

Dans ce chapitre, nous avons présenté la mise en œuvre de notre solution qui est un courtier en tant que service qui aborde le problème de l'interopérabilité sémantique entre les services de Cloud Computing de type SaaS.

Au début, nous avons présenté les ressources matérielles sur lesquelles notre solution a été réalisée. Puis, nous avons présenté les outils utilisés et l'environnement de développement. Ensuite, nous avons présenté notre registre qui est sous forme d'ontologie, suivi des interfaces de notre application et de leurs fonctionnalités. Enfin, nous avons montré le déploiement de notre interface de recherche avec l'outil Docker.

Conclusion générale

Le but principal de ce travail était de réaliser une solution à base de courtier en tant que service pour la réalisation de l'interopérabilité sémantique entre les services de Cloud Computing de type SaaS.

Nous avons fait notre travail suivant la démarche Extreme Programming. Nous avons commencé par étudier les concepts de base du Cloud Computing et de l'interopérabilité. Ensuite, nous nous sommes lancées dans l'étude des travaux existants ayant proposé des solutions pour l'interopérabilité dans le Cloud Computing de manière générale puis dans les SaaS de manière plus spécifique en analysant ces travaux et en les comparant.

Notre solution consiste à établir un service de courtage permettant aux fournisseurs de publier leurs SaaS pour les mettre à disposition aux personnes désirant en tirer profit. Notre courtier ayant sa propre description qui prend en charge n'importe quel service, permet une réponse optimale aux requêtes des utilisateurs grâce à sa procédure de sélection de services basée sur les critères de qualité de service et les préférences de l'utilisateur ainsi qu'à la procédure de composition de service qui est déclenchée pour répondre aux requêtes complexes et prenant en considération l'adaptation des données entre les services composants. De plus, notre courtier prend en charge le SLA des services offerts à l'utilisateur de manière interopérable.
Notre solution prend la forme d'une application Java EE accessible, facile à déployer et à utiliser.

Ce travail a plusieurs perspectives potentielles, notamment la prise en charge des requêtes d'utilisateurs en utilisant la reconnaissance vocale et la récitation des résultats pour faciliter l'utilisation du courtier, surtout aux non-voyants. Pour l'adaptation des données, nous avons traité le problème de l'hétérogénéité des formats de données et donc de l'hétérogénéité syntaxique, à l'avenir nous voudrions prendre en considération l'hétérogénéité sémantique des données partagées entre SaaS. De plus, nous aimerions pouvoir tester le travail du courtier en utilisant des services réels pour traiter l'aspect de l'invocation des services et voir les performances de notre solution en action. Nous estimons aussi que l'integration d'un système d'audit et d'évaluation de service peut constituer une valeur ajoutée importante à notre système.

Bibliographie

[1] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Science and Technology, Special Publication, 800*, vol. 145, 2011.

[2] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," *Proceedings - 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008*, pp. 5–13, 2008.

[3] "AWS | Qu'est-ce que le cloud computing – Les avantages du cloud." https://aws.amazon.com/fr/what-is-cloud-computing/. Consulté le:2020-02-28.

[4] "What is Cloud Computing | IBM." https://www.ibm.com/cloud/learn/cloud-computing. Consulté le:2020-08-16.

[5] N. Grevet, *Le cloud computing : évolution ou révolution ?* Mémoire de recherche, INIT, 2009.

[6] M. THABET, *Une approche à base d'agents et de composants pour l'interopérabilité des Clouds*. Thèse de doctorat, Université Constantine2- Abdelhamid Mehri, 2015.

[7] M. Hogan, F. Liu, A. Sokol, and J. Tong, "Nist cloud computing standards roadmap," *NIST Special Publication*, vol. 35, pp. 6–11, 2011.

[8] A. Chauhan, P. Suneja, S. Kumar, R. Sahota, and L. Rachna, "Comparative analysis of different actors in cloud computing reference model," pp. 297–300, 06 2014.

[9] A. Sivakumar, "A Survey on Cloud Computing Models and it's Applications," *International Research Journal of Engineering and Technology (IRJET)*, vol. 6, no. 12, pp. 1922–1928, 2019.

[10] "The Realities of XaaS (Everything-as-a-Service) - Bits & Bytes." https://blog.bittitan.com/realities-everything-service-xaas/. Consulté le:2020-02-28.

[11] "Blockchain as a Service: Enterprise-Grade BaaS Solutions." https://101blockchains.com/blockchain-as-a-service/amp/. Consulté le:2020-08-24.

[12] E. Simmon, "Evaluation of cloud computing services based on nist sp 800-145," *NIST Special Publication*, vol. 500, p. 322, 2018.

[13] P. Patel, A. Ranabahu, and A. Sheth, "Service Level Agreement in Cloud Computing," *Kno.e.sis Publications*, 01 2009.

[14] G. Kaur, "SLA : A Comprehensive Survey," *International Journal of Computer Science and Engineering*, vol. 2, no. 8, pp. 1–6, 2015.

[15] W. Edward, "Service level agreement in the data center," *Sun BluePrints Online*, pp. 1–10, 2002.

[16] L.-j. Jin, V. Machiraju, and A. Sahai, "Analysis on Service Level Agreement of Web Services ," *HP June*, vol. 19, 2002.

[17] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, 05 2010.

[18] J. W. Rittinghouse and J. F. Ransome, *Cloud computing : implementation, management, and security*. CRC Press, 2016.

[19] P. Harsh, F. Dudouet, R. G. Cascella, Y. Jegou, and C. Morin, "Using open standards for interoperability issues, solutions, and challenges facing cloud computing," *Proceedings of the 2012 8th International Conference on Network and Service Management, CNSM 2012*, pp. 435–440, 2012.

[20] "What is a Container? | App Containerization | Docker." https://www.docker.com/resources/what-container. Consulté le:2020-08-23.

[21] S. Singh and N. Singh, "Containers & docker: Emerging roles & future of cloud technology," in *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pp. 804–807, 01 2016.

[22] M. A. AlZain, E. Pardede, B. Soh, and J. A. Thom, "Cloud computing security: From single to multi-clouds," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 5490–5499, IEEE Computer Society, 2012.

[23] H. Kurdi, A. Al-Anazi, C. Campbell, and A. Al Faries, "A combinatorial optimization algorithm for multiple cloud service composition," *Computers and Electrical Engineering*, vol. 42, pp. 107–113, 02 2015.

[24] N. Grozev and R. Buyya, "Inter-Cloud architectures and application brokering: Taxonomy and survey," *Software - Practice and Experience*, vol. 44, pp. 369–390, 03 2014.

[25] B. K. Rani, B. P. Rani, and A. V. Babu, "Cloud computing and inter-clouds-types, topologies and research issues," in *Procedia Computer Science*, vol. 50, pp. 24–29, Elsevier B.V., 2015.

[26] D. Petcu, "Multi-cloud: Expectations and current approaches," in *MultiCloud 2013 - Proceedings of the International Workshop on Multi-Cloud Applications and Federated Clouds*, pp. 1–6, ACM Press, 2013.

[27] R. Nakhate and S. Gandhi, "Survey paper on cloud computing," *International Journal of Scientific and Research Publications*, p. 451, 2013.

[28] A. V. Trilochan, "Cloud Computing: Evolution and Challenges," *International Journal of Engineering Science and Computing*, vol. 7, no. 4, pp. 10197–10200, 2017.

[29] B. Rashidi, M. Sharifi, and T. Jafari, "A Survey on Interoperability in the Cloud Computing Environments," *International Journal of Modern Education and Computer Science*, vol. 5, no. 6, pp. 17–23, 2013.

[30] C. Baudoin, E. Dekel, and M. Edwards, "Interoperability and Portability for Cloud Computing: A Guide," *Cloud Standards Customer Council*, pp. 1–20, 2014.

[31] M. D. Wallace and B. Tushar, "Interoperability and Portability," Cloud Security Alliance Group 4, 2011. https://studylib.net/doc/6968414/interoperability-and-portability. Consulté le: 2020-09-02.

[32] M. Kostoska, M. Gusev, S. Ristov, and K. Kiroski, "Cloud computing interoperability approaches - Possibilities and challenges," *CEUR Workshop Proceedings*, vol. 920, pp. 30–34, 2012.

[33] N. Loutas, E. Kamateri, F. Bosi, and K. Tarabanis, "Cloud computing interoperability: The state of play," in *Proceedings - 2011 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2011*, pp. 752–757, 11 2011.

[34] S. Dowell, A. Barreto, J. B. Michael, and M. T. Shing, "Cloud to cloud interoperability," *Proceedings of 2011 6th International Conference on System of Systems Engineering: SoSE in Cloud Computing, Smart Grid, and Cyber Security, SoSE 2011*, pp. 258–263, 2011.

[35] Z. Zhang, C. Wu, and D. W. Cheung, "A survey on cloud interoperability: Taxonomies, standards, and practice," *Performance Evaluation Review*, vol. 40, no. 4, pp. 13–22, 2013.

[36] R. Cohen, "Examining cloud compatibility, portability and interoperability," *ElasticVapor: Life in the Cloud*, 2009.

[37] R. Mungrah and Z. Cadersaib, "Cloud application integration methodology using enterprise application integration," *2017 International Conference on Infocom Technologies and Unmanned Systems: Trends and Future Directions, ICTUS 2017*, vol. 2018-January, pp. 327–333, 2018.

[38] "What Is Cloud Migration? An Introduction to Moving to the Cloud." https://searchcloudcomputing.techtarget.com/definition/cloud-migration. Consulté le:2020-08-31.

[39] D. Petcu, "Portability and interoperability between clouds: challenges and case study," in *European Conference on a Service-Based Internet*, pp. 62–74, 2011.

[40] D. Petcu, "On the interoperability in multiple Clouds," *CLOSER 2013 - Proceedings of the 3rd International Conference on Cloud Computing and Services Science*, pp. 581–590, 2013.

[41] M. Becker, "Interoperability case study: Cloud computing," *Berkman Center Research Publication*, no. 2012-11, 2012.

[42] E. Nogueira, A. Moreira, D. Lucrédio, V. Garcia, and R. Fortes, "Issues on developing interoperable cloud applications: definitions, concepts, approaches, requirements, characteristics and evaluation models," *Journal of Software Engineering Research and Development*, vol. 4, no. 1, 2016.

[43] H. Saouli, *Découverte de services web via le cloud computing à base d'agent mobile*. Thèse de doctorat, Université Mohamed Khider Biskra, 2015.

[44] B. Benmedakhene and I. e. Yahyaoui, *Composition des services Web: Etat de l'art*. Thèse de master, Université Larbi Tébessi - Tébessa.

[45] M. MERZOUG, *La découverte et la sélection des services web.* Thèse de doctorat, Université Abou Bekr Belkaid, 2018.

[46] B. Alistair, C. Anis, H. Steffen, K. Tom, K. Uwe, O. Daniel, and R. Philip, "Unified service description language (usdl) pricing module," tech. rep., SAP Research, 2010.

[47] L. Sun, H. Dong, and J. Ashraf, "Survey of service description languages and their issues in cloud computing," in *Proceedings - 2012 8th International Conference on Semantics, Knowledge and Grids, SKG 2012, doi = 10.1109/SKG.2012.49*, pp. 128–135, 10 2012.

[48] "Web Service Definition Language (WSDL)." https://www.w3.org/TR/2001/NOTE-wsdl-20010315. Consulté le:2020-07-22.

[49] "RDF - Semantic Web Standards." https://www.w3.org/RDF/. Consulté le:2020-07-22.

[50] M. LADJENEF, *système à base d'ontologie pour l'aide à la maintenance des machines*. Thèse de master, MOHAMED BOUDIAF - M'SILA, 2018.

[51] D. Kalibatiene and O. Vasilecas, "Survey on ontology languages," *Lecture Notes in Business Information Processing*, vol. 90, pp. 124–141, 01 2011.

[52] "OWL-S: Semantic Markup for Web Services." https://www.w3.org/Submission/OWL-S/. Consulté le:2020-07-19.

[53] "Extensible Markup Language (XML)." https://www.w3.org/XML. Consulté le:2020-07-19.

[54] "Extensible Markup Language (XML) 1.0 (Fifth Edition)." https://www.w3.org/TR/2008/REC-xml20081126/. Consulté le:2020-07-19.

[55] "Schema - W3C." https://www.w3.org/standards/xml/schema. Consulté le:2020-07-19.

[56] G. Antoniou and F. Van Harmelen, *A semantic web primer*. MIT press, 2012.

[57] N. Noy and D. Mcguinness, “Ontology development 101: A guide to creating your first ontology,” *Knowledge Systems Laboratory*, vol. 32, 01 2001.

[58] C. Roche, “Ontology: A survey,” *IFAC Proceedings Volumes*, vol. 36, 09 2003.

[59] E. Oren, K. Möller, S. Scerri, S. Handschuh, and M. Sintek, “What are semantic annotations,” *Relatório técnico. DERI Galway*, vol. 9, p. 62, 2006.

[60] D. Bouchiha, M. Malki, D. Djaa, A. Alghamdi, and K. Alnafjan, “Semantic annotation of web services: A comparative study,” *Studies in Computational Intelligence*, vol. 492, no. July, pp. 87–100, 2013.

[61] S. Ghazouani and Y. Slimani, “Towards a standardized cloud service description based on usdl,” *Journal of Systems and Software*, vol. 132, 06 2017.

[62] A. Merizig, *Approche de composition de services web dans le Cloud Computing basée sur la coopération des agents*. Thèse de doctorat, Université Mohamed Khider-Biskra, 2018.

[63] R. Karim, *END-TO-END QOS COMPUTATION FOR VERTICAL SERVICE COMPOSITION IN THE CLOUD*. Thèse de doctorat, Ryerson University, 2015.

[64] I. HAMADA, *Utilisation de "WordNet" pour indexation sémantique & recherche d’information*. Thèse de master, Ryerson University, 2013.

[65] A. Ali, F. Alfayez, H. Alquhayz, R. Marg, S. Vihar, and N. Delhi, “Semantic Similarity Measures Between Words : a Brief Survey,” *Science International Journal*, vol. 30(6), no. November 2012, pp. 907–914, 2018.

[66] A. SIAGH and C. DEROUICHE, *Similarité sémantique entre concepts: Application à la recherche d’images*. Thèse de master, Université Kasdi Merbah Ouargla, 2016.

[67] T. Slimani, B. BenYaghlane, and K. Mellouli, “Une extension de mesure de similarité entre les concepts d’une ontologie,” in *International conference on sciences of electronic, technologies of information and telecommunications*, vol. 69, 2007.

[68] H. DAHA and S. LIFA, *Une approche formelle pour la composition des services web*. Thèse de master, Université ECHAHID HAMMA LAKHDAR - EL-OUED, 2015.

[69] B. Karim Talal and M. Rachid, “Service Discovery – A Survey and Comparison,” *International Journal of UbiComp*, vol. 4, no. 3, pp. 23–39, 2013.

[70] A. Bekkouche, *Composition des Services Web Sémantiques À base d’Algorithmes Génétiques*. Thèse de magistère, Ecole doctorale: Science et Technologie de l’Information et de la Communication (STIC), 2012.

[71] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, “Qos-aware middleware for web services composition,” *Software Engineering, IEEE Transactions on*, vol. 30, pp. 311 – 327, 06 2004.

[72] A. Bekkouche, *Vers une composition automatique des services web.* Thèse de doctorat, Université Abou Bekr Belkaid, 2018.

[73] "Web Services Glossary." https://www.w3.org/TR/ws-gloss/{#}WSCWGreqs. Consulté le:2020-08-25.

[74] G. A. Lewis, "The Role of Standards in Cloud- Computing Interoperability," *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, no. 10, pp. 1652–1661, 2012.

[75] W. Li and L. Ping, "Trust model to enhance security and interoperability of cloud environment," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5931 LNCS, pp. 69–79, 2009.

[76] D. Petcu, C. Crăciun, M. Neagul, I. Lazcanotegui, and M. Rak, "Building an interoperability API for Sky computing," *Proceedings of the 2011 International Conference on High Performance Computing and Simulation, HPCS 2011*, pp. 405–411, 2011.

[77] J. K. Wang, J. Ding, and T. Niu, "Interoperability and standardization of intercloud cloud computing," *arXiv preprint arXiv:1212.5956*, 2012.

[78] D. Petcu, "Towards Application Level Interoperability in Multi-Clouds," West University of Timisoara and Institute e-Austria Timisoara, 2013. https://staff.fmi.uvt.ro/~dana.petcu/contract.htm. Consulté le: 2020-09-02.

[79] M. Moravcik, P. Segec, and M. Kontsek, "Overview of Cloud Computing Standards," *ICETA 2018 - 16th IEEE International Conference on Emerging eLearning Technologies and Applications, Proceedings*, no. 11, pp. 395–402, 2018.

[80] R. Sandhu and I. Chana, "Cloud computing Standardization Initiatives: State of Play," *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, vol. 2, no. 5, 2013.

[81] R. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "Nist cloud computing reference architecture," in *2011 IEEE World Congress on Services*, pp. 594–596, 07 2011.

[82] "Foundation Project." https://www.apache.org/foundation/. Consulté le:2020-08-27.

[83] "OASIS Identity in the Cloud TC | OASIS." https://www.oasis-open.org/committees/tc{_}home.php?wg{_}abbrev=id-cloud. Consulté le:2020-07-23.

[84] "Vision and Mission | SNIA." https://www.snia.org/about/vision-mission. Consulté le:2020-07-23.

[85] C. Pahl, L. Zhang, and F. Fowley, "Interoperability standards for cloud architecture," *CLOSER 2013 - Proceedings of the 3rd International Conference on Cloud Computing and Services Science*, pp. 123–126, 2013.

[86] "About OMG | Object Management Group." https://www.omg.org/about/index.htm. Consulté le:2020-08-21.

[87] “OpenID Foundation | OpenID.” https://openid.net/foundation/. Consulté le:2020-07-20.

[88] “(99+) Cloud Computing Interoperability Forum (CCIF) - Google Groupes.” https://groups.google.com/forum/{#}!forum/cloudforum. Consulté le:2020-07-19.

[89] “IETF | About.” https://www.ietf.org/about/. Consulté le:2020-08-27.

[90] “About IBM - United States.” https://www.ibm.com/ibm/us/en/?lnk=fab. Consulté le:2020-08-25.

[91] “Cloud standards: Tools to ensure cloud application interoperability.” https://www.ibm.com/developerworks/cloud/library/cl-tools-to-ensure-cloud-application-interoperabilityindex.html. Consulté le:2020-08-25.

[92] “The Simple Cloud API.” https://www.ibm.com/developerworks/library/os-simplecloud/. Consulté le:2020-08-25.

[93] “European Commission | European Union.” https://europa.eu/european-union/about-eu/institutions-bodies/european-commission. Consulté le:2020-08-26.

[94] “start [Open Grid Forum].” https://www.ogf.org/ogf/doku.php. Consulté le:2020-07-19.

[95] “About | Cloud Security Alliance.” https://cloudsecurityalliance.org/about/. Consulté le:2020-07-24.

[96] “About.” https://www.redhat.com/en/about. Consulté le:2020-08-27.

[97] “Inside the IT industry’s largest commercial open source software ecosystem.” https://www.redhat.com/en/blog/inside-it-industrys-largest-commercial-open-source-software-ecosystem. Consulté le:2020-08-27.

[98] “About Aeolus - The Aeolus Project.” https://aeolusproject.github.io/about.html. Consulté le:2020-08-27.

[99] “AWS Marketplace: enStratus.” https://aws.amazon.com/marketplace/seller-profile?id=a0f7565a-4c7c-4d1a-ad8f-d72c73a8a6ea. Consulté le:2020-07-23.

[100] “CADF | DMTF.” https://www.dmtf.org/standards/cadf. Consulté le:2020-07-22.

[101] “Apache jclouds® :: Home.” http://jclouds.apache.org/. Consulté le: 2020-07-19.

[102] “Apache Libcloud: Python way to manage the cloud — Quintagroup.” https://www.quintagroup.com/cms/python/libcloud. Consulté le: 2020-07-18.

[103] “OpenStack Docs: Ussuri.” https://docs.openstack.org/ussuri/. Consulté le: 2020-07-19.

[104] “About the Service Oriented Architecture Modeling Language Specification Version 1.0.1.” https://www.omg.org/spec/SoaML/About-SoaML/{#}document-metadata. Consulté le: 2020-07-19.

[105] D. Tidwell, “The Simple Cloud API Writing portable, interoperable applications for the cloud,” tech. rep., 10 2009.

[106] “Open Cloud Computing Interface | Open Standard | Open Community.” https://occi-wg.org/. Consulté le: 2020-03-01.

[107] “About Aeolus - The Aeolus Project.” https://aeolusproject.github.io/about.html. Consulté le: 2020-08-15.

[108] “GitHub - dasein-cloud/dasein-cloud.” https://github.com/dasein-cloud/dasein-cloud/. Consulté le: 2020-08-22.

[109] “ CIMI | DMTF.” https://www.dmtf.org/standards/cimi. Consulté le: 2020-02-11.

[110] “OVF | DMTF.” https://www.dmtf.org/standards/ovf. Consulté le:2020-02-11.

[111] “Apache Libcloud is a standard Python library that abstracts away differences among multiple cloud provider APIs | Apache Libcloud.” https://libcloud.apache.org/. Consulté le: 2020-03-09.

[112] “Deltacloud API.” https://deltacloud.apache.org/. Consulté le:2020-07-21.

[113] “Open Source Cloud Computing Infrastructure - OpenStack.” https://www.openstack.org/. Consulté le: 2020-03-08.

[114] “OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC | OASIS.” https://www.oasis-open.org/committees/tc{_}home.php?wg{_}abbrev=tosca. Consulté le: 2020-07-23.

[115] “CDMI Cloud Storage Standard | SNIA.” https://www.snia.org/cloud/cdmi. Consulté le: 2020-03-03.

[116] “About the Service Oriented Architecture Modeling Language Specification Version 1.0.1.” https://www.omg.org/spec/SoaML/. Consulté le: 2020-03-03.

[117] “OpenID Foundation website.” https://openid.net/. Consulté le: 2020-03-05.

[118] “Unified Cloud Interface Project (UCI) - Google Groupes.” https://groups.google.com/g/unifiedcloud. Consulté le: 2020-08-24.

[119] “OAuth Community Site.” https://oauth.net/. Consulté le: 2020-03-05.

[120] “Simple Cloud API - Cloud Technologies & PHPilosophy.” https://simplecloudapi.org/. Consulté le: 2020-03-05.

[121] “mOSAIC Cloud.” https://www.mosaic-cloud.eu/{#}. Consulté le: 2020-09-01.

[122] “Audit all your cloud users | CloudAudit.” https://cloudaudit.app/. Consulté le: 2020-09-01.

[123] “Aeolus Project · GitHub.” https://github.com/aeolusproject. Consulté le: 2020-09-01.

[124] R. Rezaei, T. K. Chiew, S. P. Lee, and Z. Shams Aliee, “A semantic interoperability framework for software as a service systems in cloud computing environments,” *Expert Systems with Applications*, vol. 41, no. 13, pp. 5751–5770, 2014.

[125] S. Ghazouani, H. Mezni, and Y. Slimani, "Bringing semantics to multicloud service compositions," *Software - Practice and Experience*, vol. 50, no. 4, pp. 447–469, 2020.

[126] C. Zeng, X. Guo, W. Ou, and D. Han, "Cloud computing service composition and search based on semantic," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5931 LNCS, no. 2007, pp. 290–300, 2009.

[127] H. Mezni and M. Sellami, "Multi-cloud service composition using Formal Concept Analysis," *Journal of Systems and Software*, vol. 134, pp. 138–152, 2017.

[128] "What is Eclipse?." https://www.educative.io/edpresso/what-is-eclipse. Consulté le: 2020-08-18.

[129] " Eclipse (logiciel) : définition et explications." https://www.techno-science.net/definition/517.html. Consulté le: 2020-08-18.

[130] "Java SE Development Kit 8 - Downloads." https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html. Consulté le: 2020-08-18.

[131] "Java SE Runtime Environment 8 - Downloads." https://www.oracle.com/java/technologies/javase-jre8-downloads.html. Consulté le: 2020-08-18.

[132] "javac - Java programming language compiler." https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javac.html. Consulté le: 2020-08-18.

[133] "Développement de programmes Java à l'aide du kit JDK." https://www.java.com/fr/download/faq/develop.xml. Consulté le: 2020-08-18.

[134] K. Kronis and M. Uhanova, "Performance comparison of java ee and asp.net core technologies for web api development," *Applied Computer Systems*, vol. 23, pp. 37–44, 05 2018.

[135] "1 Overview (Release 7)." https://docs.oracle.com/javaee/7/tutorial/overview.htm{#}BNAAW. Consulté le: 2020-08-18.

[136] "Apache Tomcat® - Welcome!." http://tomcat.apache.org/. Consulté le: 2020-08-18.

[137] S. Agrawal and R. Gupta, "Development and comparison of open source based web gis frameworks on wamp and apache tomcat web servers," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-4, 05 2014.

[138] K. Kumar, *Review on Tomcat Security & Role of Java Security Manager (JSM) in Apache Tomcat*. Thèse de doctorat, De Montfort University, 09 2012.

[139] "protégé." https://protege.stanford.edu/products.php. Consulté le: 2020-08-15.

[140] “MySQL :: MySQL 8.0 Reference Manual :: 1.2.1 What is MySQL?.” https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. Consulté le: 2020-07-17.

[141] “MySQL :: Why MySQL?.” https://www.mysql.com/why-mysql/. Consulté le: 2020-07-17.

[142] “SQL Standards.” https://docs.oracle.com/cd/B19306{_}01/server.102/b14200/intro002.htm. Consulté le: 2020-07-17.

[143] “Java JDBC API.” https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/. Consulté le:2020-09-22.

[144] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, “Introduction to wordnet: An on-line lexical database*,” *Five Papers on WordNet", available in the World-Wide Web at http://www. cogsci. princeton. edu/~ wn*, vol. 3, 01 1991.

[145] “GitHub - jaytaylor/jaws: Java API for WordNet Searching (JAWS) .” https://github.com/jaytaylor/jaws. Consulté le: 2020-09-01.

[146] “Google Code Archive - Long-term storage for Google Code Project Hosting..” https://code.google.com/archive/p/ws4j/. Consulté le: 2020-09-22.

[147] “dmeoli/WS4J: WordNet Similarity for Java provides an API for several Semantic Relatedness/Similarity algorithms.” https://github.com/dmeoli/WS4J. Consulté le: 2020-09-22.

[148] M. Horridge and S. Bechhofer, “The owl api: A java api for owl ontologies,” *Semantic Web*, vol. 2, pp. 11–21, 01 2011.

[149] “GitHub - owlcs/owlapi: OWL API main repository.” https://github.com/owlcs/owlapi. Consulté le: 2020-07-12.

[150] “Apache Jena - What is Jena?.” https://jena.apache.org/about{_}jena/about.html. Consulté le: 2020-07-12.

[151] “Apache Jena - Welcome to Apache Jena.” https://jena.apache.org/about{_}jena/about.html. Consulté le: 2020-07-12.

[152] “SPARQL Query Language for RDF.” https://www.w3.org/TR/rdf-sparql-query/{#}initDefinitions. Consulté le:2020-07-22.

[153] “Empowering App Development for Developers | Docker.” https://www.docker.com/,. Consulté le: 2020-08-25.

[154] “Docker overview | Docker Documentation.” https://docs.docker.com/getstarted/overview/. Consulté le:2020-09-15.

Annexe A: Notions sur Docker

Dans ce chapitre annexe, nous parlerons de certains concepts liés à l'outil de conteneurisation Docker, tels que le Docker Engine, les objets Docker et l'architecture Docker,qui ont tous été extraits du site officiel de Docker [154].

Docker Engine

Docker ou Docker Engine est une application suivant une architecture client-serveur. (Docker overview website) C'est un outil qui permet l'empaquetage portable léger et portable des applications reposant sur la virtualisation par conteneurs.

Docker Engine est composé d'un serveur, un client et une API REST.

-Le serveur: est aussi appelé processus daemon, c'est un programme qui permet la gestion des objets Docker comme des conteneurs ou des images. Il peut aussi communiquer avec d'autres daemons pour gérer des services Docker.

-Le client: prend la forme d'une interface en ligne de commande. Elle permet aux utilisateurs d'interagir avec Docker à travers des commandes qui sont envoyées au daemon. Le client peut également interagir avec plusieurs daemon. L'API REST permet aux programmes de savoir quelles interfaces utiliser pour communiquer avec le daemon. C'est ce qui permet au client d'envoyer des commandes au daemon. Le daemon écoute les requêtes de l'API pour effectuer ces tâches de création et de gestions d'objets.
Le client prend la forme d'une interface en ligne de commande. Elle permet aux utilisateurs d'interagir avec Docker à travers des commandes qui sont envoyées au daemon. Le client peut également interagir avec plusieurs daemon.

-L'API REST: permet aux programmes de savoir quelles interfaces utiliser pour communiquer avec le daemon. C'est ce qui permet au client d'envoyer des commandes au daemon. Le daemon écoute les requêtes de l'API pour effectuer ces tâches de création et de gestions d'objets

Les Objets Docker

Nous avons déjà dit que le Docker daemon peut créer et gérer des objets Docker. Ces derniers peuvent être des Conteneurs, des images, des réseaux ou des volumes. Nous allons parler davantage de certains de ces objets.

-Les images: ce sont en quelque sorte des moules qui permettent de créer des conteneurs. Il s'agit de templates contenant les instructions nécessaires à la construction d'un conteneur.
La fondation d'une image exige la création d'un fichier dit Dockerfile qui contient toutes les instructions nécessaires. Ce fichier à une syntaxe simple et chaque étape décrite dans ce fichier va servir à la construction d'une couche de l'image. Une image peut utiliser d'autres images comme base et ajouter des instructions pour personnaliser la configuration nécessaire à l'exécution au lieu d'effectuer la construction de zéro.
Il est aussi possible d'utiliser des images créées et publiées par des parties tierces pour en faire des conteneurs.

-Les conteneurs:ce sont des entités exécutables déterminées par des images et des paramètres de configuration. Ils sont isolés de la machine hôte et des restes des conteneurs. Un conteneur peut être créé, démarré, déplacé, arrêté ou supprimé. Il est aussi possible de contrôler le niveau d'isolation d'un conteneur, le connecter à un réseau ou le lier à un moyen de stockage. Par défaut, une fois qu'un conteneur est supprimé, les changements effectués sur lui ne sont pas sauvegardés s'ils n'ont pas été gardés dans une source de stockage persistante.

Docker Hub

Le Docker Hub est un registre Docker publique qui sert à stocker des images docker et à les mettre à disposition des utilisateurs. La récupération d'une image depuis le Docker Hub est possible grâce à la commande pull ou la commande run. Quant à la publication et le stockage d'une image, elle se fait à travers la commande push. Lorsqu'une image est sollicitée par un client et qu'elle n'est pas disponible en local, le docker Daemon va la chercher auprès du Docker Hub par défaut.

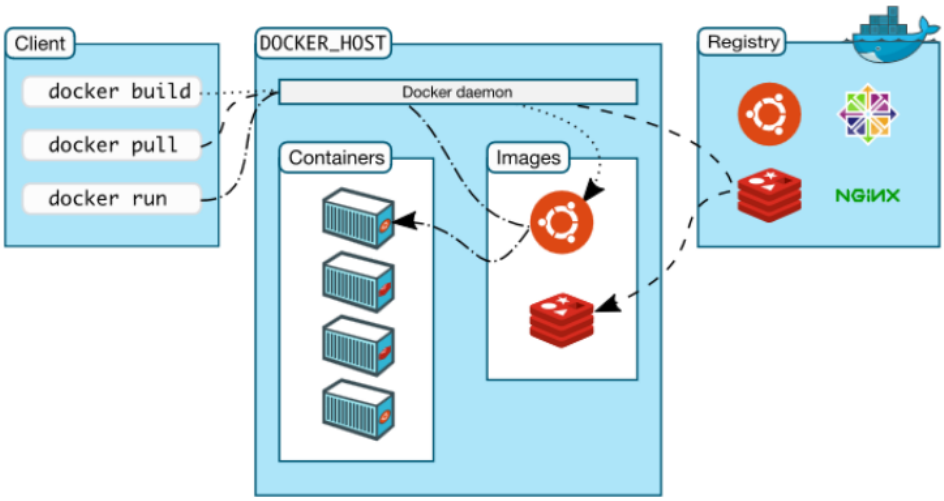


Figure 49: Architecture de Docker [154]