REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IA13/M2/2019

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Intelligence Artificielle**

Approche automatique de composition des services dans le cloud

---

Par :

TOUMI SARRA

Soutenu le 6 juillet 2019, devant le jury composé de :

Kerdoudi Mohamed Lamine	M C B	Président
Bendahmane Asma	M A A	Rapporteur
Moussaoui Manel	M A A	Examinateur

Dédicace

Je dédie ce modeste travail, aux deux êtres les plus chers à mon Cœur auxquels je dois mon existence:

À la mémoire de mon Père « Nour Eddine » (رحمه الله).

À ma chère mère pour leur patience, leur amour,

Leur soutien et leur encouragement.

À ma très chère Grand-mère.

À mes très chers frères et mes très chères sœurs.

À tous mes chers oncles et mes tantes, cousins et cousines.

À tous ma famille Toumi.

À tous celles et tous ceux qui m'ont aidé dans mes études.

À mes très chers amis et toutes mes amies.

Enfin, à toutes personnes ayant contribué de prés ou de loin à la réalisation de travail.

Remerciements

Je tiens en premier lieu à remercier Allah omnipotent, qui m'a donné la santé, le courage et la volonté pour finir ce travail.

Je tiens à remercier *Mme Bendahmane Asma* qui a accepté de m'encadrer, pour avoir brillamment dirigé cette étude et m'avoir fait partager son savoir faire et sa rigueur scientifique, ainsi que ses multiples compétences, durant la période de notre travail.

Mes vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont portés à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

Pour finir, j'adresse mes remerciements à ma famille et tous mes amis et ma profonde reconnaissance à toutes celles et tous ceux qui m'ont aidé pour finaliser ce travail.

Merci à tous et à toutes.

ملخص
تعد الحوسبة السحابية نموذجًا جديدًا للحوسبة حيث يتم تقديم البنية التحتية والمنصات والبرامج كخدمات. يمكن للحوسبة السحابية توفير بيئات تطوير قابلة للتطوير ونشرها عند الطلب. غالبًا ما يتضمن طلب المستخدم العديد من الخدمات وليس خدمة واحدة فقط. هذا هو السبب في أن هذه الخدمات يجب أن تتكون لبناء خدمة عالمية لتلبية الطلب المطلوب.
في هذا المشروع ، نعالج مشكلة تكوين الخدمة في السحابة. من ناحية أخرى ، نستخدم خوارزمية التحسين وهي الخوارزمية الجينية لاكتشاف أفضل تكوين قائم على القيود الوظيفية وغير الوظيفية جودة الخدمات .
الكلمات المفتاحية: الخوارزمية الجينية ، تكوين الخدمة ، الحوسبة السحابية ، التحسين ، الخدمات.

Résumé

Le cloud computing est un nouveau paradigme informatique où les infrastructures, les plates-formes et les logiciels sont proposés comme des services. Le Cloud Computing permet de provisionner des environnements évolutifs de développement et de déploiement, à la demande. Souvent, la requête de l'utilisateur implique de nombreux services et pas seulement un seul. C'est pour cela ces services doivent être composés pour construire un service global afin de satisfaire la demande désirée.

Dans ce projet, nous adressons le problème de composition de services dans le Cloud. En revanche, nous utilisons l'algorithme d'optimisation à savoir l'algorithme génétique pour découvrir la meilleure composition en se basant sur des contraintes fonctionnelles et non fonctionnelles(QoS).

Mots clés : algorithme génétique, composition des services, Cloud Computing, optimisation, services, QoS.

Abstract

Cloud computing is a new computing paradigm in which infrastructures, platforms and software are offered as services. Cloud computing can provide scalable development and deployment environments on demand. Often, the user's request involves several services and not just one. This is why these services must be composed to constitute a global service responding to the desired demand.

In this project, we address the problem of the composition of services in the cloud. On the other hand, we use the optimization algorithm, namely the genetic algorithm, to discover the best composition based on functional and non-functional constraints (QoS).

Key words: genetic algorithm, service composition, cloud computing, optimization, services, QoS.

Table des matières

Table des figures

Table des figures

Liste des tableaux

Liste des abréviations

AmazonVPC	Amazon Virtual Privat Cloud
API	Application Programming Interface
CSA	Composite Service Concrète
CRM	Customer Relationship Management
EC2	Amazon Elastic Compute Cloud
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
GUI	Graphic User Interface
IaaS	Infrastructure as a Service
IBM	International Business Machine
IDE	Integrated Development Environment
LCC	Loose Class Cohésion
LCOM1	Lack of Cohesion in Methods1
LCOM2	Lack of Cohesion in Methods2
LCOM3	Lack of Cohesion in Methods3
LCOM4	Lack of Cohesion in Methods4
LCOM5	Lack of Cohesion in Methods5
LAS	List Abstract Services
NIST	National Institute of Standards and Technology.
PaaS	Platform as a Service
QoS	Quality of Service

Liste des abréviations

RDF	Resource Description Framework
SA	Abstract Service
SaaS	Software as a Service
SC	Concert Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TCC	Tight Class Cohésion
TIC	Technologies de l'Information et de la Communication
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WCS	Web Coverage Service
WFS	Web Feature Service
WPS	Web Processing Service
WMS	Web Map Service
WSDL	Web Services Description Language
VPN	Virtual Private Network
XML	Extensible Markup Language

Introduction général

Les progrès de la technologie de l'information exigent un nouveau paradigme informatique qui prend en charge la fourniture des services informatiques sans les installer sur les sites locaux avec un cout minimal. Le cloud computing propose le même modèle décrit ci-dessus, il présente des services qui sont fournis sur Internet, à la demande pour laquelle les frais sont payés au moment de la libération des ressources.

En général, un cloud est un paradigme technologique, qui est une extension de plusieurs technologies existantes à savoir, le calcul parallèle et distribué, les architectures SOA (Service-Oriented Architecture), la virtualisation et la mise en réseau, etc.[10].

Le cloud computing est une collection de ressources accessibles, qui devraient répondre aux besoins des consommateurs. Les services Cloud sont définis et fournis selon trois niveaux d'offres de services: Software as a Service (SaaS), Platform as a Service (PaaS) et Infrastructure as a Service (IaaS) ; ce qui a permis aux utilisateurs des services de se concentrer sur ce que les service leur offrent plutôt que sur la façon dont les services sont implémentés ou hébergés. L'objectif du Cloud Computing est de fournir, à la demande, des services sécurisés, qualitatifs, évolutifs, rapides, plus réactifs, rentables et automatiquement provisionnés. Les services sont fournis de manière transparente (localisation indépendante). Le Cloud Computing peut aider à améliorer les performances de l'entreprise, tout en apportant une contribution pour contrôler le coût de fourniture des ressources informatiques pour toute entreprise. [10]

L'une des caractéristiques de Cloud Computing est l'élasticité, est une caractéristique cruciale pour les fournisseurs dont les applications évoluent dans des environnements hautement variables. Il s'agit de répondre aux demandes des utilisateurs en temps normal mais aussi dans les périodes de forte activité avec une réactivité maximale. Les systèmes doivent être capables de supporter une montée en charge puis un retour à la normale, sans interruption de service, et si possible, sans répercussions sur le niveau de service, c'est-à-dire dans le respect des contrats (SLAs) signés. Il existe d'autres événements imprédictibles bien que récurrents : pannes logicielles, matérielles, coupures de courant, etc.

L'élasticité doit là aussi permettre d'amortir ces événements sans impacter le service rendu à l'utilisateur. Du point de vue du fournisseur, l'élasticité est un des rouages nécessaires à l'optimisation des ressources, tout en délivrant aux utilisateurs le niveau de service souhaité, dans un contexte d'exécution de plus en plus dynamique.

Le développement rapide de l'utilisation du cloud computing conduit à une croissance du nombre de fournisseurs cloud et l'augmentation de leurs services offerts. En raison de la complexité de la demande des utilisateurs dans un environnement Cloud, un seul service simple ne peut pas satisfaire les exigences fonctionnelles et non fonctionnelles (QoS ; le temps de réponse, la fiabilité, la disponibilité, le coût, …) requises par l'utilisateur pour de nombreux cas réels. Donc, pour répondre à la demande de l'utilisateur, il est nécessaire de combiner un ensemble de services atomiques simples qui fonctionnent les uns avec les autres. Suite à cette tendance, la composition de services tiers est devenue un paradigme de référence pour le développement d'applications robustes et riches, ou encore pour l'automatisation de processus métiers. Avec la disponibilité de centaines de milliers de services et APIs web, la réalisation de telles intégrations devient lourde et fastidieuse quand effectuée manuellement. Par ailleurs, chaque client peut exiger des besoins et politiques d'intégration différentes, ce qui complexifie davantage la tâche. De plus, fournir une telle solution qui soit à la fois robuste et scalable est une tâche non-triviale. Il est donc primordial d'étudier comment coordonner de manière efficace les interactions entre les services existants [33].

La demande d'une composition de services vise généralement à développer des applications déployées dans des environnements Cloud, dont le but est d'effectuer des tâches aidant au bon fonctionnement d'une entreprise. Ces applications sont représentées par la combinaison et la collaboration de plusieurs composants, dont leur rôle est d'effectuer des fonctionnalités précises ,ces composants sont des services métiers.

Les services métiers sont des packages applicatifs, fournissant des fonctionnalités métiers. Ils sont stockés sur différentes plateformes, déployable sur différentes infrastructures Cloud, et composables avec d'autres services. Leurs déploiements et composition se font à l'aide de scripts. Les modules d'un ERP (Enterprise Ressource Planning) sont des exemples de services métiers.

L'objectif de ce travail est :

- Proposer une approche pour la composition automatique des services métiers dans le Cloud.
- Définir les différentes relations qu'un service doit avoir avec les autres services pour fonctionner.
- Implémenter une technique de regroupement des services basés sur les critères de cohésion.
- Utiliser un algorithme d'optimisation à savoir l'algorithme génétique pour découvrir la meilleure composition en se basant sur des contraintes fonctionnelles et non fonctionnelles.

Afin de présenter notre travail, nous avons organisé ce mémoire en quatre chapitres à savoir :

Le premier chapitre (Cloud Computing) est consacré complètement à définir le Cloud, décrire son fonctionnement général, présenter ses principaux acteurs, à savoir ses avantages, ses problèmes et ses inconvénients.

Le deuxième chapitre (Composition des services dans le Cloud Computing) dans ce chapitre nous explorons les différentes définitions des services et une comparaison entre eux, avec une analyse des différents travaux de recherche relatifs à la composition de services.

Le troisième chapitre (Approche automatique de composition des services dans le cloud) décrit l'approche proposée pour la composition automatique des services métiers dans le cloud.

Le quatrième chapitre (Implémentation) englobe les aspects expérimentaux tels que : les outils et les techniques ayant servi à la réalisation du simulateur de notre approche proposée et les résultats obtenus.

Nous terminerons ce manuscrit par une conclusion générale y incluant quelques perspectives possibles.

Etat de l’art

Cloud Computing

1

Chapitre 1 : Cloud Computing

1.1. Introduction

Les technologies de l'information et de la communication (TIC) se développent plus rapide et de manière progressive. Dans ces dernières années il y a une nouvelle destination, son but est d'améliorer les services dans le domaine TIC, il s'agit du "Cloud Computing". Ce dernier est un nouveau concept informatique qui consiste à proposer des services informatiques sous forme de services à la demande, accessibles de n'importe où, n'importe quand et par n'importe qui. Cette nouvelle technologie permet à des entreprises d'externaliser le stockage de leurs données et de leur fournir une puissance de calcul supplémentaire pour le traitement de grosse quantité d'information. **[1]**

L'objectif de ce chapitre est d'aborder la notion de Cloud Computing en présentant son historique et ses différentes définitions, ses principales caractéristiques, ses différentes modèle de service et modèles de déploiements, ses avantages et inconvénients.

1.2. Historique du cloud computing

Le cloud computing n'est pas nouveau, il est exploité depuis les années 2000, les changements qui ont permis l'apparition du cloud computing sont nombreux. Ainsi on peut citer l'apparition du SaaS (Software as a Service), le produit délivré par le cloud. Puis il y a le concept de virtualisation qui permet une mutualisation des serveurs et offre donc une mise en production simplifiée et un meilleur taux d'utilisation des ressources. Le cloud computing est donc la juxtaposition de ces technologies pour passer à la vitesse supérieure sur l'exploitation de données à travers l'Internet. Le concept du Cloud Computing a été mis en œuvre en 2002 par Amazon, un leader de e-business, pour absorber la charge importante des commandes faites sur leur site Internet au moment des fêtes de Noël. Récemment, d'autres acteurs comme Google et Microsoft proposent à leur tour des services similaires.**[2]**

1.3. Définition

Dans la littérature, il existe plusieurs définitions concernant le concept du Cloud Computing. Certaines sont plus générales que d'autres.

Le National Institute of Standards and Technology (NIST) a donné une définition qui reprend ses principes de base *« Le Cloud Computing est un modèle pratique, à la demande,*

pour établir un accès par le réseau à un ensemble partagé de ressources informatiques configurables (réseaux, serveurs, stockages, applications et services) qui peuvent être rapidement mobilisées et mises à disposition en minimisant les efforts de gestion ou les contacts avec le fournisseur de services. » **[1].**

Une deuxième définition donnée en 2009 par l'université de Californie à Berkeley par Armbrustet à qui ne considère que le *« Cloud Computing désigne à la fois les applications livrées comme services sur l'Internet et le matériel et logiciel de système dans les centres de données qui fournissent ces services ».* **[5]**

Selon **[3]**, le Cloud Computing est défini comme une utilisation de la technologie informatique qui peut exploiter la puissance de traitement de nombreuses machines inter-réseau tout en cachant la structure réelle. Les utilisateurs n'ont pas besoin d'avoir des connaissances en expertise ou de contrôler l'infrastructure technologique dans le Cloud qui les soutient.

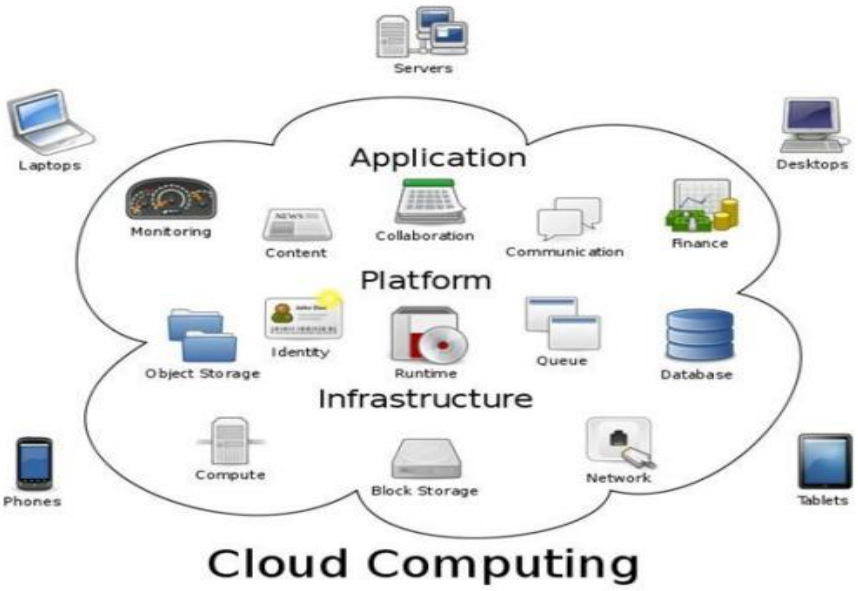

Figure 1.1: différents composants du Cloud Computing. [7]

1.4. Les cinq caractéristiques essentielles du Cloud computing

La technologie du Cloud Computing a été émergée avec des caractéristiques qui la différencient des autres technologies (Grid computing, SOA, . . .). Cinq caractéristiques essentielles correspondantes au Cloud Computing.

- **Accès aux services par l'utilisateur à la demande**

La mise en œuvre des systèmes est entièrement automatisée et c'est l'utilisateur, au moyen d'une console de commande, qui met en place et gère la configuration à distance. [4]

- **Accès réseau large bande**

Ces centres de traitement sont généralement raccordés directement sur le backbone internet pour bénéficier d'une excellente connectivité. Les grands fournisseurs répartissent les centres de traitement sur la planète pour fournir un accès aux systèmes en moins de 50 ms de n'importe quel endroit. [4]

- **Réservoir de ressources (non localisé)**

La plupart de ces centres de cloud comportent des dizaines de milliers de serveurs et de moyens de stockage pour permettre des montées en charge rapides. Il est souvent possible de choisir une zone géographique pour mettre les données "près" des utilisateurs. [4]

- **Redimensionnement rapide (élasticité)**

La mise en ligne d'une nouvelle instance d'un serveur est réalisée en quelques minutes, l'arrêt et le redémarrage en quelques secondes. Toutes ces opérations peuvent s'effectuer automatiquement par des scripts. Ces mécanismes de gestion permettent de bénéficier pleinement de la facturation à l'usage en adaptant la puissance de calcul au trafic instantané. [4]

- **Facturation à l'usage**

Il n'y a généralement pas de coût de mise en service, c'est l'utilisateur qui réalise les opérations. La facturation est calculée en fonction de la durée et de la quantité de ressources utilisées. [4]

1.5. Les modèles de services

Le Cloud Computing est composé de trois services, que nous allons exposer

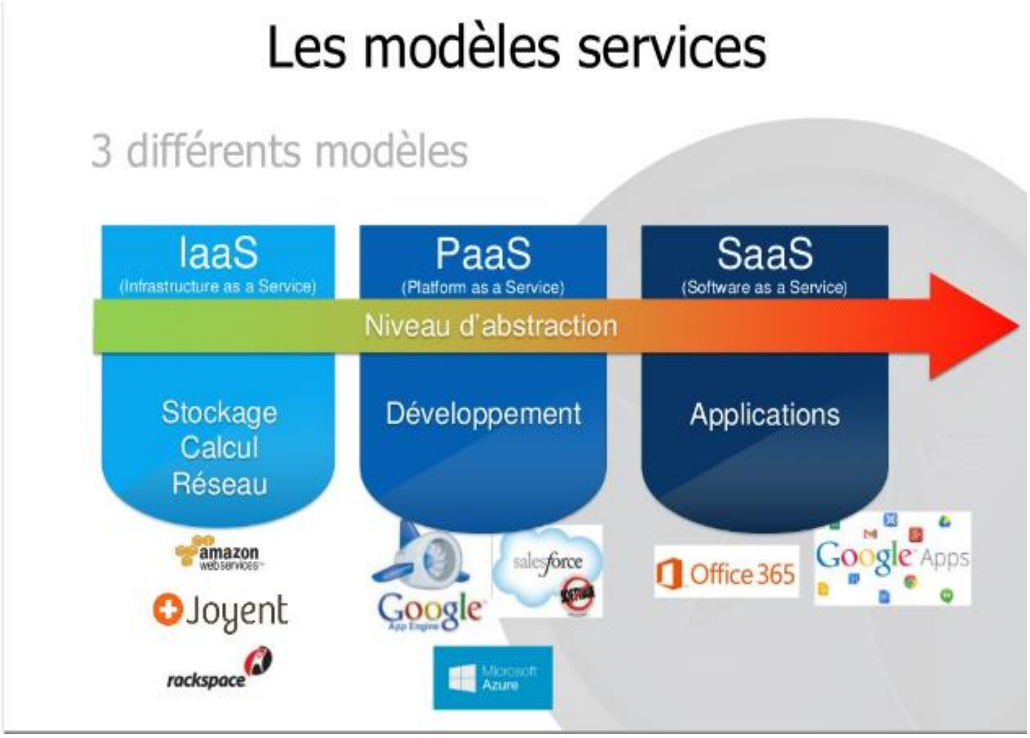

Figure 1.2:modèles de service de Cloud Computing [9]

1.5.1. IAAS (Infrastructure as a Service)

IaaS (Infrastructure as a service) en Français "L'infrastructure en tant que service", est un service principal dans le Cloud, ce service fourni à l'entreprise différents composants informatiques comme les espaces de stockages, équipements réseaux, des unités centrales, etc.

Les utilisateurs peuvent accéder à ces services à la demande via l'internet sans restriction, comme s'ils travaillent sur un matériel local. [6]

1.5.2. PAAS (Platform as a Service)

PaaS (Platform as a service) en Français "plate-forme en tant que service", est un modèle composé de tous les éléments et les services nécessaires pour facilités les développements des applications où PasS prépare des environnements spécialisés pour aide les utilisateurs dans la construction, la livraison, la extension de leurs projets. L'utilisateur embauche une plateforme sur laquelle il peut développer, tester et exécuter ses applications, parce que PaaS évite d'acheter et d'installer des logiciels, aussi il ne gère ni ne contrôle l'infrastructure sous jacente, mais contrôle les applications déployées. Aussi est une plateforme d'exécution hébergée par un opérateur relié au réseau internet [6].

1.5.3. SAAS (Software as a Service)

SaaS (Software as a Service) en Français " L'application en tant que service" est le modèle le plus utilisé dans le monde après le service d'email, est un modèle de distribution des logiciels et des applications qu'ils sont hébergées dans des centres de données, qu'il donne la possibilité des clients pour consommer ces application à la demande via l'internet avec une facturation à l'usage réel. Ces applications, prêts à l'emploi, et ne nécessitent pas de maintenance, d'installation de logiciel, et de mise à jour, toutes ces opérations sont effectuées par le fournisseur d'application, dans SaaS l'utilisation d'application reste transparente pour les utilisateurs, qui ne se soucient ni de la plateforme, ni du matériel [6].

1.6. Modèles de déploiement

On peut distingue quatre type principaux dans le Cloud sont : le Cloud privé, le Cloud public, le Cloud hybride (Mixte) et le Cloud communautaire, nous avons expliques chaque type présente [6].

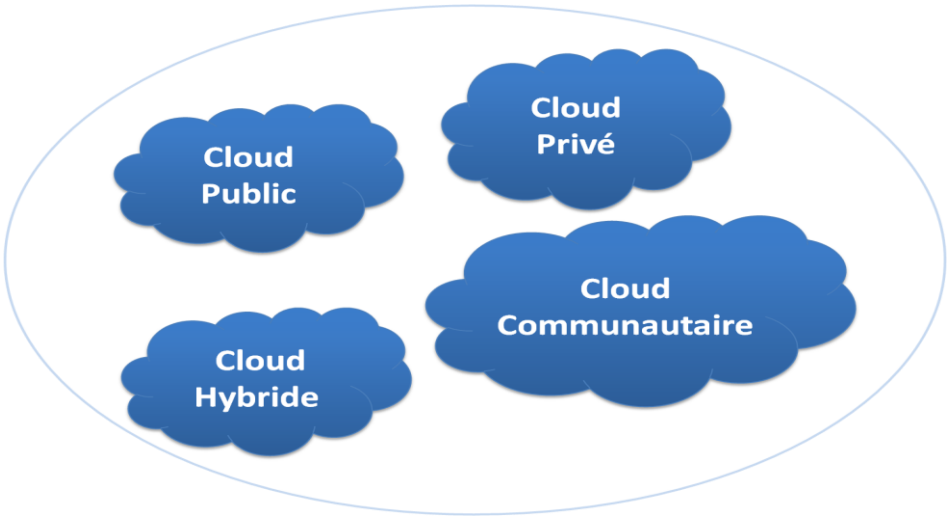

Figure 1.3:Les modèles de déploiement de Cloud Computing

1.6.1. Cloud public

En anglais "Public Cloud" est un ensemble des services et des ressources accessibles par Internet et gérées par un prestataire externe, ces ressources et services sont partagés entre plusieurs clients, qu'ils les utilisent à la demande et à tout moment sans savoir où elles existent, Ces services peuvent être gratuits ou payants.

On cas les services sont payants, il existe des contrats SLA (Service Level Agreement) entre les clients et les fournisseurs, SLA est un document qui définit la qualité de service requise entre les deux. Quelques exemples de Cloud public: Amazon Elastic Compute Cloud (EC2) et Amazon S3, Sun Cloud, IBM's Blue Cloud, Google AppEngine et Windows Azure Services Platform. [2].

1.6.2. Cloud privé

En anglais " Private Cloud " est un ensemble des services et des ressources disponible à un seul client par exemple une entreprise. Le Cloud privé peut être géré par l'entreprise elle-même, ou bien avec ses branches, dans ce cas il s'appel "Le Cloud privé Interne", come il peut être géré par un prestataire externe qu'il est louée par l'entreprise, dans ce cas s'appel "Le Cloud privé Externe ". Il est accessible via des réseaux sécurisés de type VPN (Virtual Private Network), par exemple Amazon Virtual Private Cloud (Amazon VPC) [6].

1.6.3. Cloud communautaire

En anglais "Cloud Community" est un ensemble des infrastructures, qu'ils permettent à plusieurs clients ou organisations de partager les déférentes ressources, ces ressources sont généralement spéciales à des organisations, ce type de Cloud peut être géré par ces organisations elles-mêmes ou par des fournisseurs externes. Aussi il permet à un groupe des utilisateurs de créer leur propre Cloud avec des caractéristiques de Cloud privé tel que la sécurité, ressources dédiées et un coût réduit [6].

1.6.4. Cloud hybride

En anglais " Hybrid Cloud" est une composition de deux ou trois types de Cloud (Privé, communautaire et public). Par exemple l'utilisation des applications dans un Cloud public mais ces applications nécessitent des données stockées sur un Cloud privé [6].

1.7. Les avantages et les limites du Cloud

Toute analyse sérieuse du Cloud Computing doit tenir compte des avantages et des inconvénients offerts par cette technologie. Dans ce qui suit, nous allons montrer les avantages et les inconvénients du Cloud Computing. [11]

1.7.1. Avantages

Le Cloud Computing offre beaucoup d'avantages et de flexibilité à ses utilisateurs. L'utilisateur peut opérer n'importe où et à tout moment de manière sécurisée. Vu

le nombre croissant d'appareils compatibles avec le Web qui sont utilisés aujourd'hui (par exemple, les tablettes, les téléphones intelligents, etc.), l'accès à l'information et aux données doit être rapide et plus simple. Certains de ces avantages, très pertinents concernant l'utilisation d'un cloud peuvent être les suivants [10] :

1. Réduire le coût de gestion et de l'investissement initial : avec le cloud, les entreprises ne se soucient pas de la gestion des ressources où du personnel nécessaire à la supervision de leurs plateformes. Le Cloud minimise les risques commerciaux.
2. Fournir une infrastructure dynamique qui offre des coûts réduits et des services améliorés avec moins de coûts de développement et de maintenance .
3. Fournir des services à la demande, flexibles, évolutifs, améliorés et adaptables grâce au modèle de paiement à l'usage « Pay-as-you-go » .
4. Fournir une disponibilité et des performances cohérentes avec des charges maximales provisionnées automatiquement.
5. Se rétablir rapidement et améliorer les capacités de restauration pour améliorer la résilience des entreprises.
6. Fournir une capacité de traitement, de stockage, de réseau illimité, etc. de manière élastique.
7. Offrir des mises à jour automatiques de logiciels, compatibilité du format de document améliorée et compatibilité améliorée entre les différents systèmes d'exploitation.
8. Offrir une collaboration de groupe facile, c'est-à-dire une flexibilité pour les utilisateurs à l'échelle mondiale de travailler sur le même projet.
9. Offrir un calcul respectueux de l'environnement car il utilise uniquement l'espace serveur requis par l'application.

1.7.2. Les limites du Cloud Computing

Certains des inconvénients lors de l'utilisation d'un coud sont comme suit [10]:

1. Le Cloud nécessite un réseau avec une haute vitesse de communication et une connectivité constante.
2. Les données et les applications sur un Cloud public pourraient ne pas être très sécurisées, ce qui pose le problème de la confidentialité et de la sécurité.
3. Nécessite une surveillance et une application constante des accords de niveau de service (SLA).

1.8. Conclusion

Dans ce chapitre nous avons présenté les principaux concepts et définitions concernant le domaine des cloud computing.

Dans le prochain chapitre est consacré aux représentants des notions et technique de composition des services dans le cloud computing.

Chapitre 02

Composition des services dans le Cloud Computing

2

Chapitre 2 : Composition des services dans le cloud computing

2.1. Introduction

Dans ce chapitre, nous allons présentes la notion de service, service web, service métier et service cloud. Ensuite nous introduisons les notions de composition des services dans le cloud ainsi que leurs défis rencontrés.

2.2. Définition de service

Le service est une entité logicielle fonctionnelle déployée et invocable à distance. Il existe plusieurs définitions du terme "service", les plus populaires sont :

> *« A service represents some functionality (application function, business transaction, system service, etc.) exposed as a component for a business process».* [13]

D'après cette définition on peut dire qu'un service représente certaines fonctionnalités (application fonctionnelle, transaction commerciale, un service du système de base, etc.) exposées sous la forme d'un composant au sein d'un processus métier

> *« A service in SOA is an exposed piece of functionality with three properties: (1) The interface contract to the service is platform-independent, (2) The service can be dynamically located and invoked, (3) The service is self-contained. That is, the service maintains its own state. »* [14]

Un service, doit respecter trois propriétés :

- le contrat du service est exposé dans une interface indépendante de toute plate-forme.
- le service peut être dynamiquement localisé et invoqué.
- le service est autonome et sait maintenir son propre état courant.

> *« Un service permet d'exposer une ou plusieurs fonctionnalités, offertes par un fournisseur, à des clients potentiels. »* [21].

2.3. Définition de SOA

Il n'y a pas une définition exacte de l'architecture orienté service. En effet plusieurs définitions ont été proposées, mais elles sont toutes d'accord que SOA est un

paradigme destiné à résoudre les problèmes d'hétérogénéité et interopérabilité des logiciels qui constituent le système d'information.

- Selon la Consortium W3C [18] :

  « SOA is a set of components which can be invoked, and whose interface descriptions can be published and discovered. »

  Selon cette définition SOA est un ensemble de composants qui peuvent être invoqués, et dont les descriptions d'interface peuvent être publiées et découverts

- Selon Thomas Erl [19]:

  « SOA establishes an architectural model that aims to enhance the efficiency, agility, and productivity of an enterprise by positioning services as the primary means through which solution logic is represented in support of the realization of strategic goals associated with service-oriented computing. »

  Selon cette définition, SOA est un modèle d'architecture. Son but est de progresser l'efficacité, l'agilité et la production en positionnant les services comme le primaire, c.-à-d. avec quel solution est représenté dans un support de la réalisation des buts stratégiques associées avec SOA

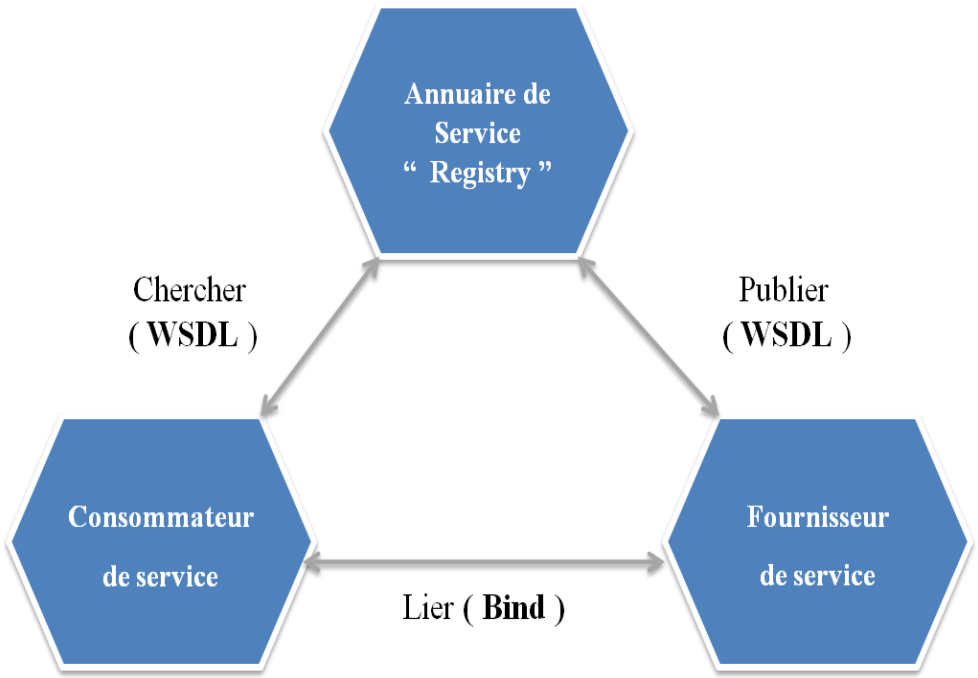

Figure 2.1: Le concept SOA [15].

L'architecture orienté service est basée sur trois acteurs principaux (Figure 2.1) définis comme suit [15] :

- **Fournisseur de service** : correspond au propriétaire du service. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service.
- **Consommateur de service** : correspond au demandeur de service, d'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être elle-même un service Web.
- **Annuaire de service** : correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

2.4. Service Web

Les Web services ont été proposés initialement par IBM et Microsoft, puis en partie standardisés par le consortium du World Wide Web : le W3C [18].

- Selon *W3C* [18] :

  « A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.»

  Cette définition met en valeur les avantages principaux d'un service Web, à savoir [17] :

  ✓ Son interface décrite d'une manière interprétable par les machines, qui permet aux applications clientes d'accéder aux services de manière automatique.

  ✓ Son utilisation de langages et protocoles indépendants des plates-formes d'implantation, qui renforcent l'interopérabilité entre services.

  ✓ Son utilisation des normes actuelles du Web, qui permettent la réalisation des interactions faiblement couplées et favorisent aussi l'interopérabilité.

- Selon IBM [16]:

  « Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the Web. »

2.5. Service métier

- Selon Benfenatki [23] :

> *« Des packages applicatifs, fournissant des fonctionnalités métiers, qui sont développés par des tiers. Ils sont stockés sur différentes plateformes, sont déployables sur différentes infrastructures Cloud, et composables avec d'autres services. Leurs déploiements et composition se font à l'aide de scripts. Les modules d'un ERP (Enterprise Ressource Planning) sont des exemples de services métiers ».*

Les auteurs ont défini pour chaque service métier ses relations de composition, ses contraintes de déploiement et les caractéristiques techniques d'un service IaaS.

2.6. Service cloud

Un service Cloud est un protocole d'interface informatique de la famille des technologies Web permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit d'un ensemble de fonctionnalités exposées par des fournisseurs Cloud. Le service Cloud peut être décrit, publié, localisé, et agrégé dynamiquement avec d'autres services afin de construire des systèmes distribués, interopérables et évolutifs [25].

Les services Cloud sont publiés dans un annuaire avec toutes leurs informations nécessaires à leurs utilisations. Cet annuaire est un registre mettant à la disposition de l'utilisateur un ensemble d'informations spécifiant les propriétés des services Cloud accessibles à travers l'Internet. Il indique principalement le fonctionnement du service, les informations de l'accessibilité pour assurer un accès structuré entre le client du service et son fournisseur.

2.7. Comparaison entre service cloud et service métier

Dans cette section, nous allons citer quelques différences entre les deux services. La différence moyenne réside au niveau de la notion de déployabilité.

Le déploiement d'un service Cloud est méconnu pour l'utilisateur. Il ne se soucie pas ni comment ni quand le service a été déployé. Ce que l'utilisateur sache c'est qu'il choisit le service et commence à l'utiliser. En revanche, les services métiers sont des packages déployables dans un environnement remplissant les contraintes de

déploiement. Par conséquent, les contraintes de déploiement doivent être connues par l'utilisateur.

2.8. Comparaison entre service web et service métier

Dans la littérature, Les auteurs dans **« Benfenatki, 2017** » ont présenté la comparaison suivante entre les services métiers et les services web, qui montre clairement la différence entre les deux :

- Un service Web assure une fonctionnalité particulière appliquée sur des entrées.
- Un service métier assure des fonctionnalités plus globales.

Par exemple, un service Web peut assurer la correction d'orthographe d'un texte en entrée, alors que, dans le même contexte, un service métier assurera une fonctionnalité de mailing, i.e., correction orthographique de mail, envoi/réception de mails, archivage, etc.[23]

Cependant, les services métiers peuvent invoquer des services Web où une composition de services Web. Ces derniers assurent une fonctionnalité sur une entrée pour retourner le résultat en sortie, alors que, le service métier est une application que nous utilisons. A titre d'exemple, le service ERP, où le service Geoserver se compose à partir des services web suivants pour fournir des informations géo-spatiales3: Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS) et Web Processing Service (WPS).

2.9. Composition des services

La composition de services peut être vue comme un mécanisme qui permet l'intégration des services pour réaliser une application. Le résultat d'une composition est un nouveau service, appelé service composite. Ce type de composition est dite récursif ou hiérarchique. D'autre sort la composition consiste à combiner les fonctionnalités de plusieurs services au sein d'un même processus métier dans le but de répondre à des demandes complexes qu'un seul service ne pourrait pas satisfaire. [27]

2.9.1. Le cycle de vie de composition de services

La composition de services est basée sur un ensemble d'étapes présentées comme suit [29] :

- **Phase de définition**

Dans cette étape, l'utilisateur introduit la demande qui comporte les informations sur le service voulu. Ensuite, ils seront décomposées (requête comporte plusieurs informations) de façon automatique en compte les qualités de services suggérés par l'utilisateur.

- **Phase de sélection**

Dans cette phase, à l'intérieur de chaque activité dans le service composite, on cherche dans un registre de services ceux qui concernent les éléments convenable à chaque activité à partir de la description du service déployé. On peut trouver des milliers de service qui rencontrent les demandes de l'utilisateur. Pour choisir le service adéquat, on doit utiliser un algorithme de matching. A la fin obtenu un ensemble de services retenus une alliance peuvent répondre aux besoins de l'utilisateur.

- **Phase de déploiement**

Dans cette phase, le service composite va être déployé afin d'être utilisé à partir des opérations d'instanciation et d'invocation d'après l'utilisateur final. A la fin on obtient un service composite exécutable.

- **Phase d'exécution**

Dans cette étape, une instance d'un service composite sera créée et exécutée par le moteur d'exécution qui est responsable à l'invocation des composants du service individuel. Au cours de l'exécution, plusieurs opérations seront réalisées telle que : les taches de surveillance, y'compris l'enregistrement, suivi de l'exécution, mesure du rendement et gestion des exceptions.

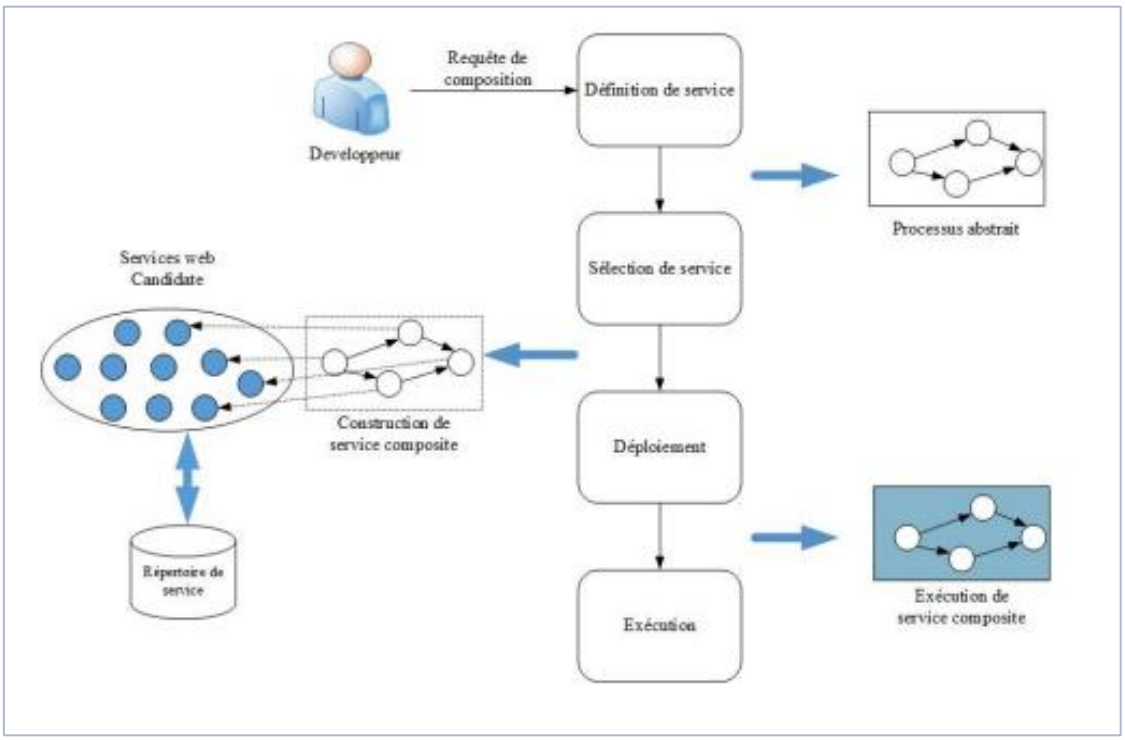

Figure 2.2:Le cycle de vie de composition de service [29].

2.9.2. Les techniques de composition des services

2.9.2.1. Composition statique

La composition des services Web peut se faire de deux manières différentes qui sont l'orchestration et la chorégraphie

- **Chorégraphie**

La chorégraphie de services décrit, d'un point de vue global, une collaboration de l'ensemble des services participants, qui interagissent afin d'atteindre un but commun. Elle fournit essentiellement les informations sur les relations entre les différentes opérations sur le service Web, elle modélise la séquence des échanges de messages entre services composants , et définit les conditions dans lesquelles ces messages sont échangés entre des clients, des fournisseurs et des partenaires. [28]

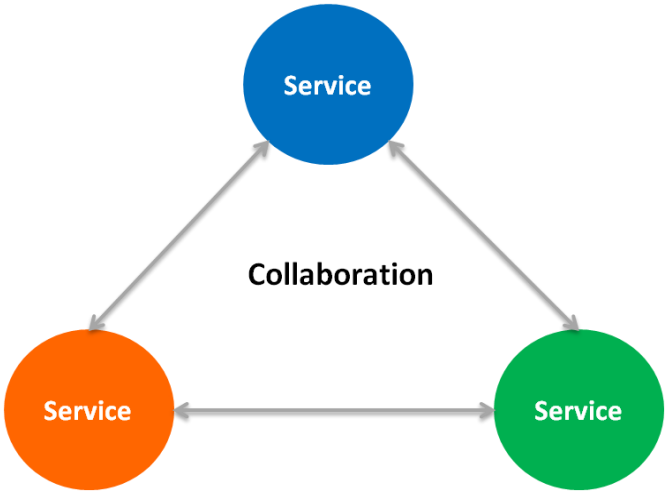

Figure 2.3:Chorégraphie de Services [28].

- **Orchestration**

L'orchestration de services permet d'expliquer l'enchainement des exécutions des services participants. Elle décrit la façon dans laquelle les services peuvent interagir collectivement au niveau des messages, incluant le logique métier et l'ordre d'exécution des interactions. L'orchestration fait habituellement référence à un processus exécutable qui interagit avec d'autres services dans l'objectif de réaliser les objectifs de l'orchestrateur qui a la responsabilité de les gérer et les invoquer [28]

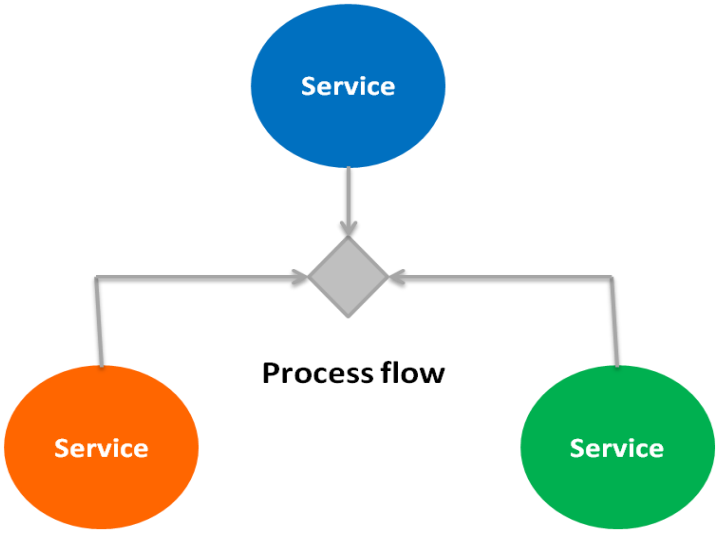

Figure 2.4:Orchestration de Services [28].

2.9.2.2. Composition dynamique

contrairement à la composition statique , la sélection des services composants et la réalisation des liaisons sont retardées jusqu'au moment de l'exécution. Les étapes de planification et de construction sont réalisées, en temps réel, en fonction des fournisseurs de services Web disponibles. [30]

La technologie de la composition dynamique est généralement contestée par :

- ✓ Le grand nombre de services qui deviennent disponibles chaque jour.
- ✓ La nature volatile de services web (par exemple : ils peuvent disparaître, être modifiés ou être temporairement indisponibles).
- ✓ Le nombre sans cesse croissant de fournisseurs de services.

2.9.3. Composition services web versus Composition services métiers dans le Cloud

Les auteurs dans [31], [32], [24] ont montré la différence entre la composition des services web et la composition des services métiers dans le Cloud.

Lorsqu'on compose des services Web, nous pouvons évaluer le matching des entrées/sorties de deux services à composer pour savoir s'ils sont composables, i.e., si les sorties du premier service sont compatibles avec les entrées du deuxième service.

La composition de services métier que nous considérons est différente de la composition des services Web. En effet, elle ne représente pas un processus métier et ne se fait donc pas par la gestion des flux de données et de contrôle entre les services. Composer des services métier revient à configurer un service de façon à ce qu'il puisse fonctionner en collaboration avec un autre service.

Exemple :

Composer une base de données avec un service ERP revient à construire les tables et les relations de la base de données en fonction du besoin de l'ERP. Donc la composition de services métiers dans le Cloud consiste à établir un lien où une connexion entre ces services, et à configurer les services pour qu'ils puissent fonctionner ensemble**[31]**.

2.9.4. Défis de composition des services dans les environnements Cloud

Le problème de la composition des services dans le cloud computing est en plein essort. Nous soulignons ici quelques sources de ses complexités:

- **L'adressage des ressources Cloud incomplètes**

Vu l'existence d'un nombre illimité de services dans un environnement Cloud et sa nature dynamique, il n'existe pas un service centralisé qui offre des informations complètes concernant les autres services dans l'environnement, chaque service possède des informations incomplètes à propos des autres services. La sélection optimale du service par un broker dépend de la disponibilité d'informations complètes et à jour sur les services. [33]

- **La dépendance/conflit entre les services**

La dépendance où les conflits qui existent entre deux ou plusieurs services entraînent un problème de composition de service complexe. Dans les scénarios du monde réel, la rencontre des dépendances et des conflits entre les services est assez courante et devrait être prise en compte dans la composition des services. [33]

- **La sécurité**

La conception et la mise à jour des règles de sécurité, des stratégies et des instructions font partie des responsabilités fondamentales des fournisseurs de services Cloud. Cependant, un cadre auto-administré fondé sur des principes pour fournir des services dans lesquels les préoccupations et les politiques de sécurité des fournisseurs sont observées, doit être fourni. [33]

- **La définition des paramètres de QoS du service composé dans le Cloud**

Ce concept fait référence aux attributs de qualité de service pris en compte pour évaluer la performance de chaque approche. Selon [35], le coût d'exécution et la latence sont les attributs de QoS les plus sensibles vis-à-vis de la croissance de la taille des données [35] ,Par contre les auteurs dans [34] ont trouvé que les paramètres suivants sont les plus importants : le temps, le coût, l'évolutivité, l'optimisation et l'efficacité.

- **La description et la mesure des valeurs de paramètres de QoS**

L'absence de méthodes pour la description des paramètres de QoS reste un défi important pour les développeurs des applications dans le Cloud. Il est à noter que, l'absence de moyens convenus pour mesurer la qualité de service est un autre problème qui n'est pas complètement résolu et qui devrait être pris en compte. [33]

- **L'adaptation**

La composition des services dans des environnements Cloud est caractérisée par l'instabilité, il est probable que de nombreux changements surviennent après le déploiement. Par exemple, de nouveaux services sont fournis, certains services sont interrompus où les attributs de QoS de certains services sont modifiés. Pour faire face à ces changements, les environnements Cloud actuels sont confrontés à la nécessité d'une adaptation continue des processus de composition automatique [34] , [35] .

- **La dimension du Cloud**

La composition des services dans le Cloud Computing peut être implémentée dans un cloud unique où dans des environnements multi-Cloud. La composition des services dans un cloud unique coordonne les services d'un fournisseur Cloud. Alors que la composition des services dans des environnements multi-Cloud coordonne les services de différents fournisseurs Cloud [35]

- **L'implémentation de la composition de services**

La plupart des recherches sur la composition de services ont principalement considéré des outils basés sur un simulateur pour les évaluations. Par conséquent, l'évaluation des approches proposées dans des scénarios du monde réel est très intéressante. [34]

2.10. Travaux de recherches connexes

Dans les travaux on été proposées sur le domaine composition des services on trouve de nombreuses études, on va diviser cette section en trois catégories représentées comme suit :

2.10.1 Composition service web

- **Jun Li et All [36],** ont proposé un algorithme heuristique pour la sélection efficace et fiable de l'ensemble de services,

  Cet algorithme proposé comprend trois étapes successives comme suit:

- Sélection basée sur la méthode de confiance est utilisée pour filtrer les services de composants non fiables.
- Construire les services associés pour réduire l'espace de recherche dans le processus de composition des services.
- A la dernière étape, une approche heuristique pour l'optimisation globale est utilisée pour obtenir une solution proche de la solution optimale.

2.10.2 Composition service cloud

- **Fateh Seghir et All [37],** ont proposé une approche hybride utilisant l'algorithme génétique et algorithme d'optimisation fruit fly basée sur la qualité de service afin de résoudre le problème de composition de services cloud.
  - ✓ Algorithme génétique utilisé pour la recherche globale
  - ✓ Algorithme d'optimisation fruit fly utilisé pour la recherche locale.

- **Karimi et al [38],** ont proposé une approche pour composition réduisant l'espace de recherche lors de la sélection de services locaux. Cette approche en utilisant :
  - Algorithme génétique pour réaliser une optimisation globale en ce qui concerne l'accord de niveau de service (SLA).
  - Le regroupement de services a été utilisé pour réduire l'espace de recherche du problème
  - Règles d'association ont été utilisées pour un service composite en fonction de leur historique afin d'améliorer l'efficacité de la composition des services.

2.10.3 Composition service métier

Benfenatki et All [23], ont proposé une méthodologie pour la composition automatisée des services métiers, qui est basée sur la combinaison de l'utilisation des architectures SOA et le Cloud Computing afin de permettre la réutilisation, le couplage faible, la composition et l'interopérabilité entre les différents Cloud, tel que les auteurs ont défini pour chaque service métier ses contraintes et ses possibilités de composition. Cependant, la sélection proposée dans cette dernière donne la priorité à l'infrastructure et le coût en premier lieu, ensuite elle prend en considération les autres paramètres de QoS.

Le tableau suivant montre la comparaison entre des approches de composition de services avec quelques travaux existants dans la littérature.

Approche	Découverte	Composition	Paramètres de QoS
Jun Li et All [36]	X	Algorithme heuristique	Coût, Fiabilité, Temps de réponse, disponibilité,
Fateh Seghir et All [37]	Processus de découverte	Approche hybride(Algorithme génétique + Algorithme fruit fly)	Coût, Fiabilité, disponibilité, Temps de réponse, Réputation,
Karimi et al [38]	Service de découverte	Algorithme de composition	Coût, Fiabilité, Temps de réponse, disponibilité,
Benfenatki et All [23]	Service de découverte	Service de composition	Coût, Confidentialité, disponibilité, Temps de réponse,

Tableau 2.1: Comparaison entre des approches de composition de services.

2.11. Conclusion

Dans ce chapitre, nous avons présenté les différentes définitions des services web ,métiers, cloud présentées dans la littérature, et une comparaison entre eux. Ensuite, nous avons cité les défis de la composition des services Cloud.

Dans le prochain chapitre, Nous allons d'écrire l'approche proposé dans ce travail pour assurons une composition automatique des services.

Chapitre 03

Approche automatique de composition des services dans le cloud

3

Chapitre 3 : Approche automatique de composition des services dans le cloud

3.1. Introduction

Après avoir présenté l'aspect théorique de notre projet et pour comprendre son fonctionnement, nous allons aborder notre architecture pour la composition automatique des services dans le cloud.

La composition de services représente un problème majeur à cause de la diversité et la croissance exponentielle des services déployés d'après les fournisseurs.

Nous décrivons dans ce chapitre la structure interne de notre architecture combinant les technologies de services métiers, cloud et outils d'optimisation. Son fonctionnement sera éclaircisse à l'aide d'un diagramme de séquence UML. La représentation détaillée inclue pour chaque composant sa structure et son fonctionnement, ainsi que les différents algorithmes implémentés.

3.2. Eude de cas

Nous considérons le scénario suivant pour illustrer ce travail. Un propriétaire d'une entreprise (client) souhaite mettre en place application d'affaire de "commande de produits en ligne". Le client doit spécifier ses besoins fonctionnels comprennent les fonctions de l'application désirées, et les besoins non fonctionnels incluent les préférences du développeur et les paramètres de qualité de service.

3.3. Spécification des besoins

Cette section présente les exigences suivant :

3.3.1. La description des besoins

Le client spécifie visuellement ses besoins via une interface utilisateur. Les besoins fonctionnels sont décrits soit par un mot clé, soit par le nom des services les satisfaisant. Au niveau de notre scénario, tandis que les besoins non-fonctionnels sont liés à des valeurs quantitatives. Les besoins non fonctionnels sont spécifiés sous forme de qualité de service et les préférences du développeur vis à vis de l'application désirée.

Pour cette raison, nous utilisons des poids que le développeur affecte aux paramètres de qualité de service.

Les valeurs des poids sont associées à des intervalles tels que :

- "low" correspond à l'intervalle [0.1, 0.4 [
- "medium" correspond à l'intervalle [0.4, 0.7 [
- "high" correspond à l'intervalle [0.7, 1]

Dans notre scénario, nous avons utilisés les paramètres de QoS les plus cités dans la littérature [33] qui sont : le coût, la confidentialité des données, la disponibilité, et le temps de réponse.

3.3.2. La description des services

Chaque service métier S1 disponible dans le registre des services est lié à un autre service métier S2 selon les contraintes de composition définie dans [31].

3.3.2.1. Les contraintes de composition

Les auteurs dans [31] définissent les contraintes de composition comme suit (voir Figure 3.1) :

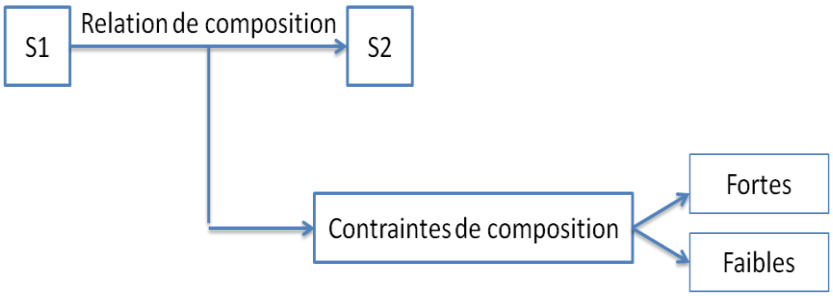

Figure 3.1:Relations d'un service [31].

Les contraintes de composition représentent l'ensemble des services qui doivent être composé avec le service S1 afin qu'il puisse fonctionner. Par exemple, on dit qu'il existe une contrainte de composition entre le service S1et le service S2, si et seulement si, le service S1 à besoin de se lier avec le service S2 pour fonctionner correctement.

Les contraintes de composition peuvent être fortes ou faibles :

- Une contrainte de composition forte est une contrainte qui impose les services qui doivent être composés avec le service S1. Par exemple, le service WordPress doit être composé avec un service de base de données MySQL.
- Une contrainte de composition faible est une contrainte qui n'impose pas les services qui doivent être composé, mais elles offrent le choix de sélectionner un service dans un ensemble de services fournissant la même fonctionnalité. Par exemple, Joomla peut être composé avec une base de données MSSQL, PostgreSQL, ou MySQL. [31]

Dans notre travail, on appliquer les contraintes de composition forte c-à-d pour chaque service, nous avons défini ses contraintes de composition fortes.

3.3.2.2. Les contraintes de cohésion

La cohésion de classe est un attribut important de qualité logicielle orienté objet. Il indique la relation entre les méthodes et les attributs d'une classe [22].

- **Métriques de cohésion**

Il existe plusieurs Métriques de cohésion proposées afin de mesurer la cohésion d'une classe à savoir le tableau suivant (voir Tableau 3.1) :

Métriques	Définition
LCOM1	Il compte le nombre de paires de méthodes qui n'accèdent pas aux mêmes attributs.
LCOM2	LCOM1- nombre de paires de méthodes qui accèdent aux mêmes attributs.
LCOM3	Défini comme le nombre de composants connectés de G, c'est-à-dire le nombre de méthodes "clusters" fonctionnant sur des ensembles disjoints de variables d'instance. LCOM3 peut être calculé par: • Créer un graphe G dont le sommet V est une méthode de classe; • Ajouter un bord E pour chaque paire de méthodes similaires; • LCOM3 est le nombre de composants connectés de G
LCOM4	où le graphe G a en plus un bord entre les sommets représentant méthodes Mi et Mj, si Mi invoque Mj ou vice versa
LCOM5	Est un formule considère un ensemble de méthodes m_i=(i =1….n) en accédant à un ensemble d'attributs Aj=(j=1……...a), et le nombre de méthodes qui accèdent à chaque attribut Alors, LCOM5 est défini par la formule suivante:

	LCOM5 = $\frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(Aj)\right) - m}{m-1}$
LCC	est définie comme le nombre relatif de méthodes connectées directement et indirectement . Pour faire le calcul LCC on utilise les formules suivantes : NP = N * (N- l) / 2 LCC = (NDC + NIC) / NP
TCC	est définie comme le nombre relatif de méthodes connectées directement. Pour faire le calcul on utilise les formules suivantes : NP = N * (N- l) / 2 TCC = NDC/NP

Tableau 3.1:Métriques de cohésion [22].

- **N :** Le nombre total des méthodes
- **NDC** : Le nombre de paires des méthodes directement liées.
- **NIC** : Le nombre de paires des méthodes indirectement liées.
- **NP** : Le nombre maximal possible de paires des méthodes connectées.

- **Critère de cohésion**

Il existe deux critères fondamentaux pour mesure la cohésion :

- ✓ Usage attribut.
- ✓ Méthode d'invocation.

- **Exemple**

L'exemple (voir Figure 3.2) suivant pour calculer TCC

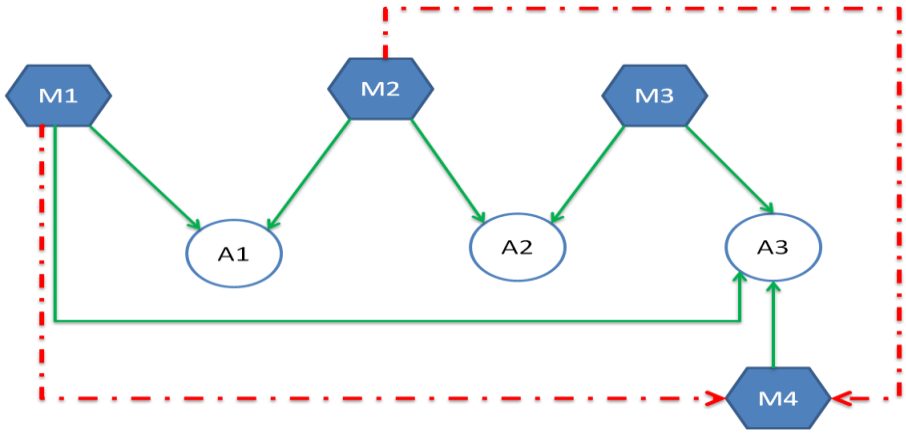

Figure 3.2:Graphe pour calculer TCC.

La figure suivant contient un ensemble de méthodes Mi (i =1,….,m) en accédant à un ensemble d'attributs Aj (j = 1,…,n) .

Et pour calculer TCC :

- Définie les relations direct entre les méthodes selon critères de cohésion

  « ***Usage attribut***; ***Méthode d'invocation*** ».

- Nombre total de méthodes.

  M1 et M2 accéder directement à attribut A1 en commun.

  M2 et M3 accéder directement à attribut A2 en commun.

  M1 et M3 accéder directement à attribut A3 en commun.

  M3 et M4 accéder directement à attribut A3 en commun.

  NDC = (M1, M2), (M2, M3), (M1,M3), (M3, M4).

  NDC = 4

- Calculer **TCC** :

  NP = N *{N- l)/2

  NP = 4 * (4 - l)/2 = 6

  TCC = 4/6.

Les auteurs dans [26] définissent la cohésion des services web comme un degré de force de la relation fonctionnelle des opérations au sein d'un service.

Dans notre travail, nous allons utiliser et adapter ces critères pour évaluer la cohésion entre les services. Nous utilisons les métriques TCC pour mesurer la cohésion des services, et nous voulons appliquer de critères pour évaluer la cohésion.

Pour mesurer la **TCC** (mesurer la cohésion du service métier composé), est définie :

- **NP** : est le nombre maximal possible de paires des services, **NP= N * (N- l)/2**
- **NDC** : est le nombre de connexions directes entre les services.

Est alors est définie **TCC** comme le nombre relatif de services connectées directement.

$$\mathbf{TCC = NDC/NP} \qquad (3.1)$$

- **Les critères de cohésion des Services**

Il existe des critères fondamentaux pour mesure la cohésion :

- Usage ressource.
- Fonctionnalités.

3.4. Conception globale de approche

La Figure 3.3 ci-dessous présente la conception globale de notre approche qui est composée essentiellement trois modules principaux utilisés afin d'atteindre l'objectif global de notre approche qui est la composition et déployer de service métier dans le cloud computing.

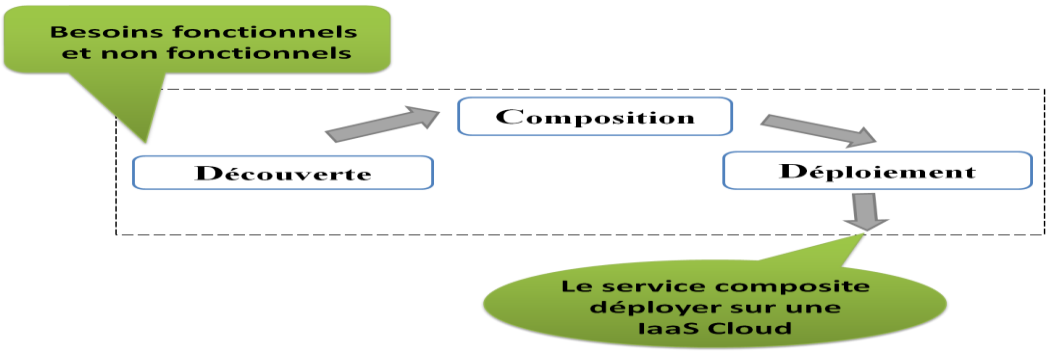

Figure 3.3: Conceptions globale du système

- **Entrée :** Besoins fonctionnels et non fonctionnels
- **Sortie :** Le service composite déployer sur une IaaS Cloud.

Module 1 : Trouver les services métiers qui répondent aux fonctionnalités désirées par le développeur (Besoins fonctionnels).

Module 2 : Composition des services à partir des services métiers découverts pendant le module 1.

Module 3 : Génération du script de composition et de déploiement.

3.5. Architecture générale du système

Notre architecture proposée support les caractéristique d'architecture base cloud notamment, l'élasticité qui représente un point clé et important pour l'offrir une adaptation automatique et transparente vis-à-vis le client en répondant a ces besoins initialement déclarés. Cette propriété est assurée via le biais de fournir un processus d'allocation de plusieurs ressources selon le nombre de requêtes, en fonction de la demande, les ressources et les capacités peuvent être vendues rapidement et même dans certains cas automatiquement, provisionnées et libérées élastiquement. Pour le consommateur, les capacités disponibles pour l'approvisionnement semblent souvent illimitées et peuvent être appropriées à n'importe quelle quantité à n'importe quel moment.

La figure (voir Figure 3.4) suivante représente l'approche de composition proposée**:**

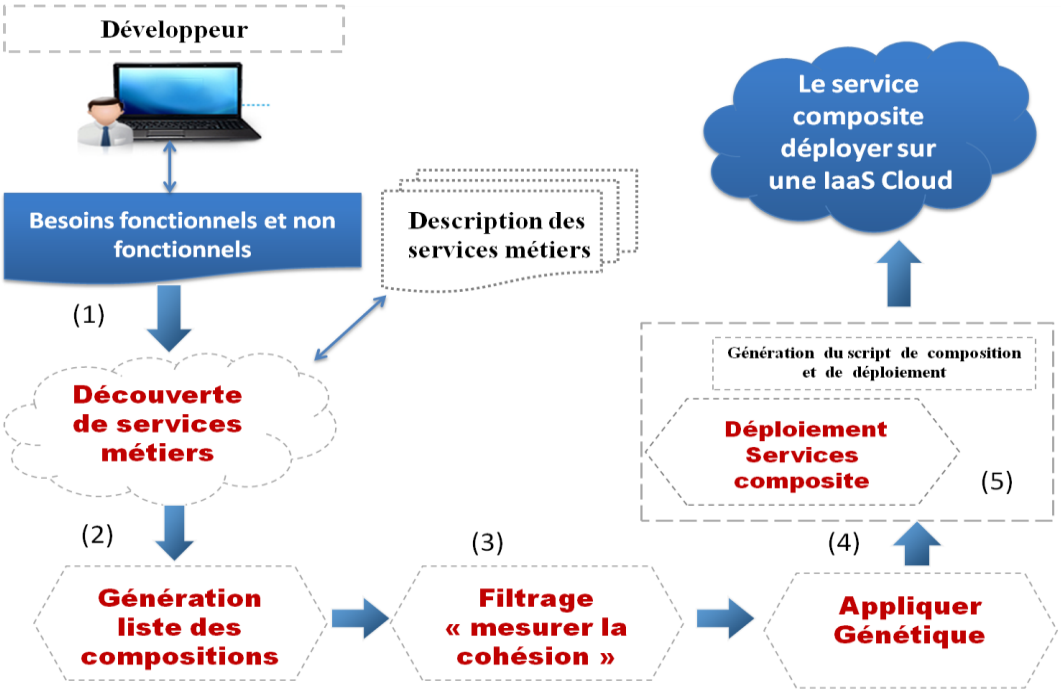

Figure 3.4:Approche de composition proposée.

Comme il est montré dans approche précédente, on peut deviser notre approche en cinq grandes parties :

3.5.1. Phase découverte de services métiers

Dans cette phase on fait trouver les services métiers qui répondent aux fonctionnalités désirées par le développeur (Besoins fonctionnels),en faisant correspondre les besoins fonctionnels du développeur avec la description des services métiers.

La figure (voir Figure 3.5) suivante représente un exemple illustré cette phase **:**

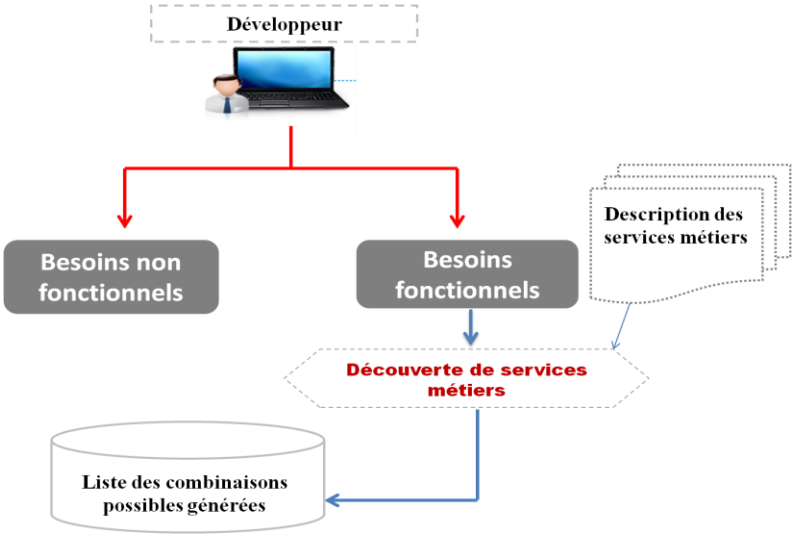

Figure 3.5: Phase découverte.

3.5.2. Phase générations liste de composition

Dans cette phase, nous pouvons générer liste de composition à partir des services métiers découverts pendant la phase président, en prenant en considération les contraintes de composition et les contraintes de cohésion.

Cette phase basée sur deux étapes présentées comme suit :

- ***Génération liste des services abstraits***

C'est une étape qui consiste à générer liste des services abstraits à base des contraintes de composition, nous commençons par le premier service métier découvert durant la phase de découverte, et nous vérifions ses contraintes de composition (services requis) à partir de son fichier descriptif. Si le service possède des contraintes, alors elles sont ajoutées à la liste des services abstraits. On répète cette étape jusqu'à ce que toutes les contraintes de compositions seront vérifiées et les liste des services abstraits possibles seront construits.

- ***Génération liste des services concrets***

C'est une étape qui consiste à générer liste des services concrets à base des contraintes de cohésion, Chaque service concret disponible dans la liste des services est associé à un service abstrait selon ses contraintes d e cohésion.

La figure 3.6 montre la liste de services abstrait (LAS) généré à partir la phase découverte

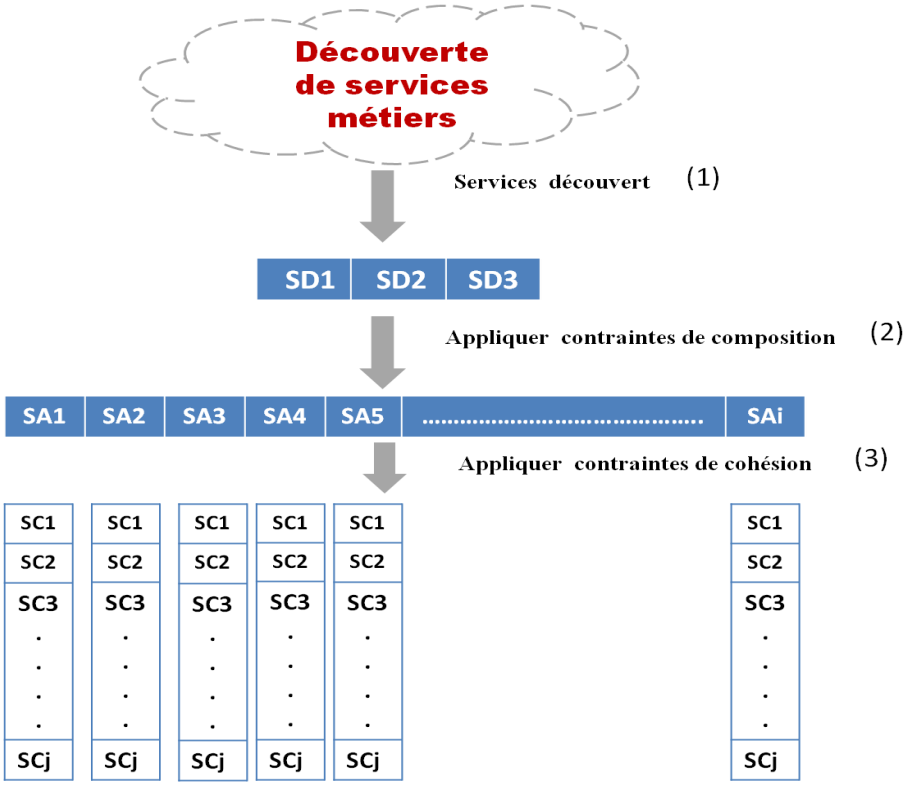

Figure 3.6:Phase générations liste de composition.

3.5.3. Phase de filtrage

Etant donné l'augmentation exponentielle du nombre de compositions avec le nombre de services concrets, la phase de filtrage selon la mesurer la cohésion s'avère nécessaire, elle va, en plus de la réduction du nombre de services candidats, garder que ceux qui offrent une fiabilité élevée pour participer au processus de composition.

Dans cette phase, nous appliquons équation TCC (Équation 3.1) pour mesurer la cohésion de liste des services concrets. Selon [22], une classe (un liste des services concrets dans notre cas) est considérée comme non cohésif lorsque TCC < 0.5, ce qui signifie que toutes les méthodes sont connectées.

Dans notre approche, nous avons testé ces métriques de manière expérimentale et nous avons découvert qu'avec un TCC> 0.3, les listes de services concrets obtenus sont «assez cohésives».

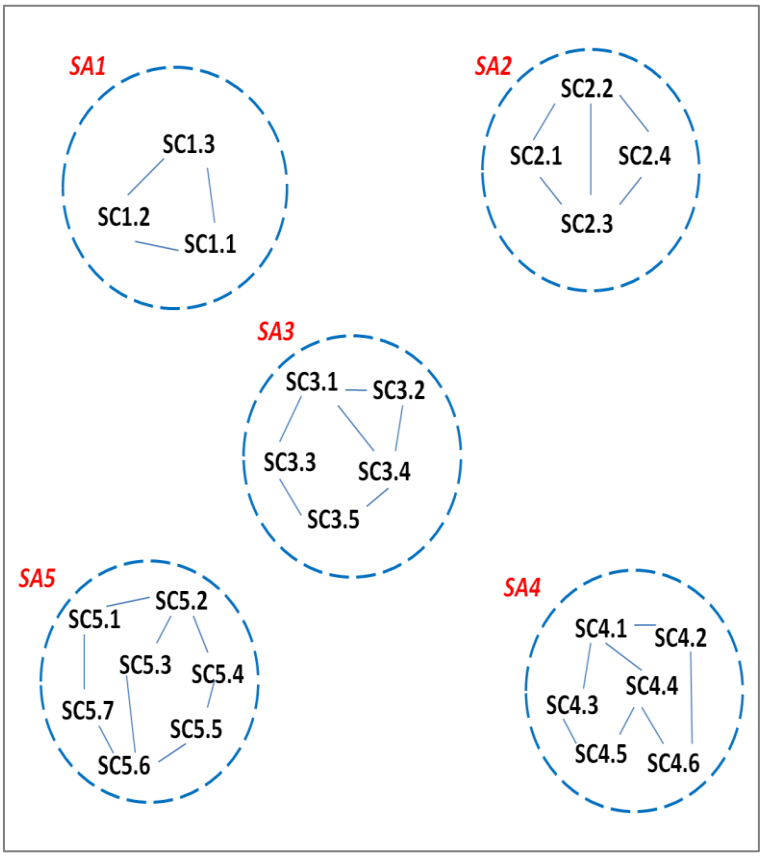

Figure 3.7:Phase de filtrage.

Pour illustrer notre phase , nous montrons comme exemple la figure (voir Figure 3.7) suivante , qui représente les composant des services concret lié pour chaque service abstrait, ainsi les relation direcet entre les services concret.

La mesure de cohésion des services concret donnée les valeur suivantes :

$$\text{TCC (SA1)} = \frac{3}{3} = 1$$

$$\text{TCC (SA2)} = \frac{4}{6} = 0.6$$

$$\text{TCC (SA3)} = \frac{5}{15} = 0.33$$

$$\text{TCC (SA4)} = \frac{6}{21} = 0.28$$

TCC (SA5) = $\frac{7}{28}$ = 0.25

Apris le mesures, les composant des services concret cohesive obtenus sont les suivants **SA1**(SC1.1 ,SC1.2,SC1.3), **SA2**(SC2.1 ,SC2.2, SC2.3 ,SC2.4) , **SA3**(SC3.1 ,SC3.2, SC3.3, SC4.4 ,SC5.5) et les composant SA4,SA5 non cohésive.

3.5.4. Phase Exécution de l'algorithme génétique

Cette phase fournit un ensemble de solutions qui respectent les contraintes du client en utilisant l'algorithme d'optimisation basé sur l'algorithme génétique traditionnel, Cet algorithme essaye de générer des solutions à partir des générations dans le but d'améliorer les résultats. Lorsque nous obtenons les meilleurs résultats conformément à la demande.

- **Entrée :** Liste de service abstrait composé de n service abstrait SAj .
  Ensemble de m services concrets lié a un service abstrait SCj.i.
- **Sortie :** Service composite optimal.

La figure 3.8 Présente les différentes étapes de l'algorithme d'optimisation.

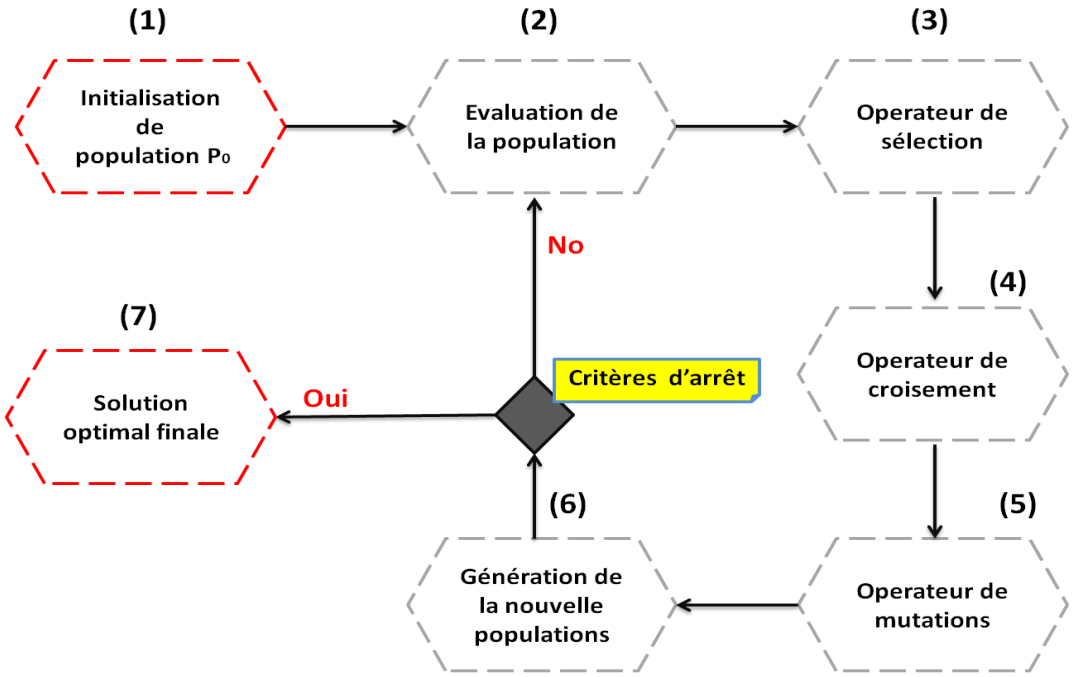

Figure 3.8: L'organigramme d'algorithme d'optimalisation

La figure 3.8 montre les différentes étapes de l'algorithme d'optimalisation, ces étapes sont présentées comme suit :

- **Initialisation de la population**

Dans cette étape, la population initiale est générée aléatoirement. Elle est constituée de N individus. Chaque individu représente une solution de composition des services concrets. L'individu (CSA) est codé par un tableau d'entier. L'entier (j : 1 ... m) dans le tableau représente le service concret sélectionné (SC j) à partir de l'ensemble des services concrets liés au service abstrait (SA_i) (i : 1 ... n) (voir figure 3.9).

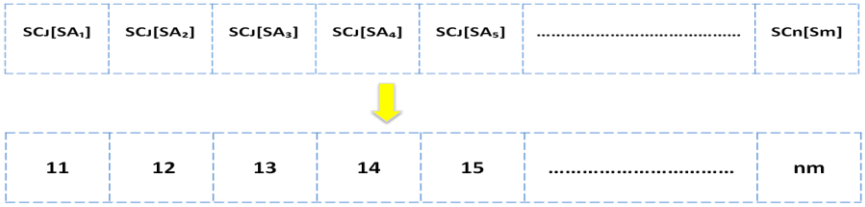

Figure 3.9:Codage de la solution.

- **Evaluation de la population**

Dans cette étape, nous allons évaluer chaque individus de population, lorsqu'on est dans un problème d'optimisation, nous avons utilisé des valeurs à maximiser et d'autres à minimiser. Pour cela, nous avons utilisé l'équation (3.2) suivante **:**

$$NoteQoS(Si\,, Qj) = \begin{cases} NoteQoSupper \\ NoteQoSlower \end{cases} \qquad (3.2)$$

Soit Si un service et Qj un paramètre de qualité de service. Le calcul de la note QoS d'un service (service Concret) pour un paramètre QoS donné se fait en fonction du type de ce dernier (Équation 3.2). Deux scénarii sont possibles :

Cas1: plus grande est la valeur du paramètre de qualité de service, meilleur est le service, par exemple, la disponibilité du service. Dans ce cas, la note QoS associée à ce paramètre pour un service donné est calculée comme suit (Équation 3.3):

$$NoteQoSupper = \frac{Val(Si,Qj)}{Max(Qj)} * Coefficient \qquad (3.3)$$

Où :

- Val est la valeur du paramètre de qualité de service pour un service (service concret) donné.
- Max est la valeur maximale du paramètre de qualité de service parmi tous les services fournissant la même fonctionnalité (i.e., les services appartenant à la même catégorie).
- Coefficient est le poids précédemment assigné au paramètre de qualité de service par l'utilisateur.

Cas2: plus petite est la valeur du paramètre de qualité de service, meilleur est le service, par exemple, le temps de réponse. Dans ce cas, la note QoS associée au paramètre de qualité de service pour un service donné est calculée comme suit (Équation 3.4):

$$NoteQoSlower = \left(1 - \frac{Val(Si,Qj)}{Max(Qj)}\right) * Coefficient \qquad (3.4)$$

Soit n le nombre de paramètres QoS considérés. La note QoS globale d'un service Si est calculée comme la somme des notes QoS de ce service pour chacun des paramètres de qualité de service considéré (Équation 3.5).

$$NoteQoS(Si) = \sum_{j=0}^{n-1} NoteQoS(Si, Qj) \qquad (3.5)$$

Le tableau (Tableau 3.2) suivant décrit les paramètres de qualité pour chaque service de l'individu de population

QoS	SC1	SC7	SC13	SC24	SC5	SC16	SC11	SC8	SC9	SC10
Temps de réponse (ms)	200	400	554	300	350	150	230	443	543	567
Cout ($)	14	16	30	20	25	25	12	17	12.5	13
Disponibilité (%)	80	60	50	90	88	65	80	85	70	55
Confidentialité de données(%)	50	80	80	60	70	70	50	90	50	70

Tableau 3.2:Paramètres de qualité des services concrets.

Suivant nous calculons la note associée au service concret SC1. NoteQoSSC1 (Équation 3.3) et NoteQoSSC1 (Équation 3.4)

On a:

$$\text{NoteQoSSC1} = \left(\left(1 - \frac{200}{567}\right) * 0.3\right) + \left(\frac{14}{30} * 0.2\right) + \left(\frac{80}{90} * 0.5\right) + \left(\frac{50}{90} * 0.7\right)$$

$$\text{NoteQoSSC1} = 1.12$$

$$\text{NoteQoSSC7} = \left(\left(1 - \frac{400}{567}\right) * 0.3\right) + \left(\frac{16}{30} * 0.2\right) + \left(\frac{60}{90} * 0.5\right) + \left(\frac{80}{90} * 0.7\right)$$

$$\text{NoteQoSSC7} = 1.94$$

Tableau (Tableau 3.3) suivant présente la note QoS calculée pour chaque service concret avec (Equation 3.2) par suit on à appliquer l'Equation 3.5 pour calculer la somme des notes l'individu de population

SCi	SC_1	SC_7	SC_{13}	SC_{24}	SC_5	SC_{16}	SC_{11}	SC_8	SC_9	SC_{10}
$NoteQoSSC_i$	1.12	1.94	1.09	1.23	1.38	1.28	1.08	1.34	0.90	0.89

Tableau 3.3:Notes QoS des services concrets.

- **Operateur de la sélection**

Elle consiste à sélectionner le meilleur service composite en termes de la probabilité de sélection pour une opération génétique. On a deux cas :

1. Le cas ou l'opération génétique est une mutation, la sélection retourne alors un seul service (parent) qui a la Prob i maximale.
2. Le cas ou l'opération génétique est un croisement, la sélection retourne alors deux services composite (parent1 et parent2) ayant la Prob1 et Prob2 respectivement représentant les deux meilleures probabilités de la population. Le service composite qui a la probabilité maximale sera sélectionné comme parent1 et l'autre sera sélectionné comme parent2

$$Propi = \frac{NoteQoS(SCi)}{\sum_{j=0}^{n-1} NoteQoS(SCj)} \quad (3.6)$$

Max(Probi), retourne le service composite qui a la probabilité maximum dans la population.

- **Operateur de croisement**

Dans cette étape, consiste à combiner deux individus (type de CSA) fournis par la l'étape présidente. Une position de découpage est choisie aléatoirement chez les deux individus. Après cela, les gènes provenant de la position du découpage sont échangés entre les deux individus (voir figure3.10). L'opération de croisement est répétée N/2 fois pour obtenir N enfants.

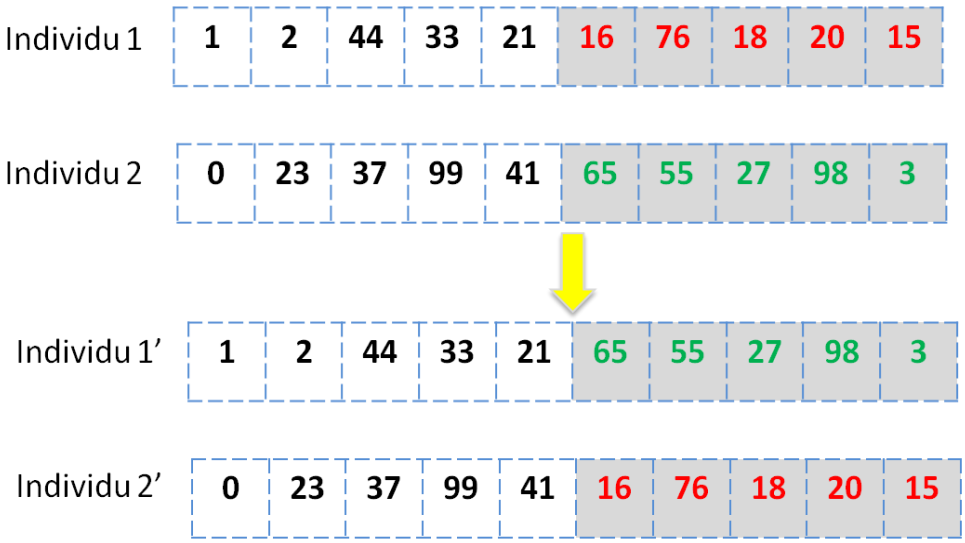

Figure 3.10: Opération de croisement.

- **Operateur de mutation**

Dans cette étape, nous changeons un génome (type de SCji) de façon aléatoire avec d'autre individu depuis la population courante (voir figure 3.11).

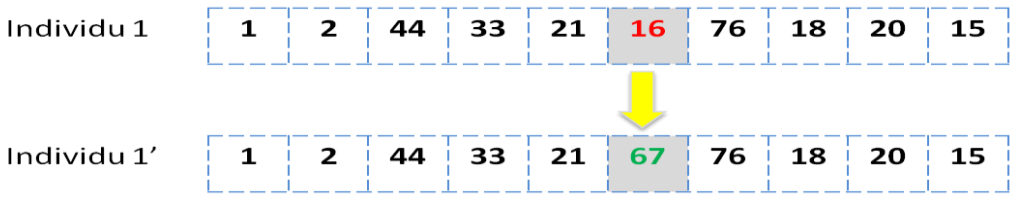

Figure 3.11: Opération de mutations

- **La génération de la nouvelle population (reproduction)**

On utilise des opérateurs génétiques (croisement et mutation) pour produire la nouvelle génération. L'opérateur de mutation modifie un individu pour en former un autre tandis que l'opérateur de croisement engendre un ou plusieurs enfants à partir de combinaisons de deux parents.

- **Critères d'arrêt**

Le cycle est répété jusqu'à ce que le nombre de générations atteigne le maximum ou atteigne maximum la fonction de fitness.

3.5.5. Phase de déployer de service composite optimal

Ce processus consiste à déployer le services composite sur IaaS, par la génération du script de composition et de déploiement. Selon notre étude.

3.6. Fonctionnement du composant approche proposé

Dans cette section, on réalise un diagramme de séquence qui décrit le scénario nominal de l'approche de composition automatiquement des services dans le cloud.

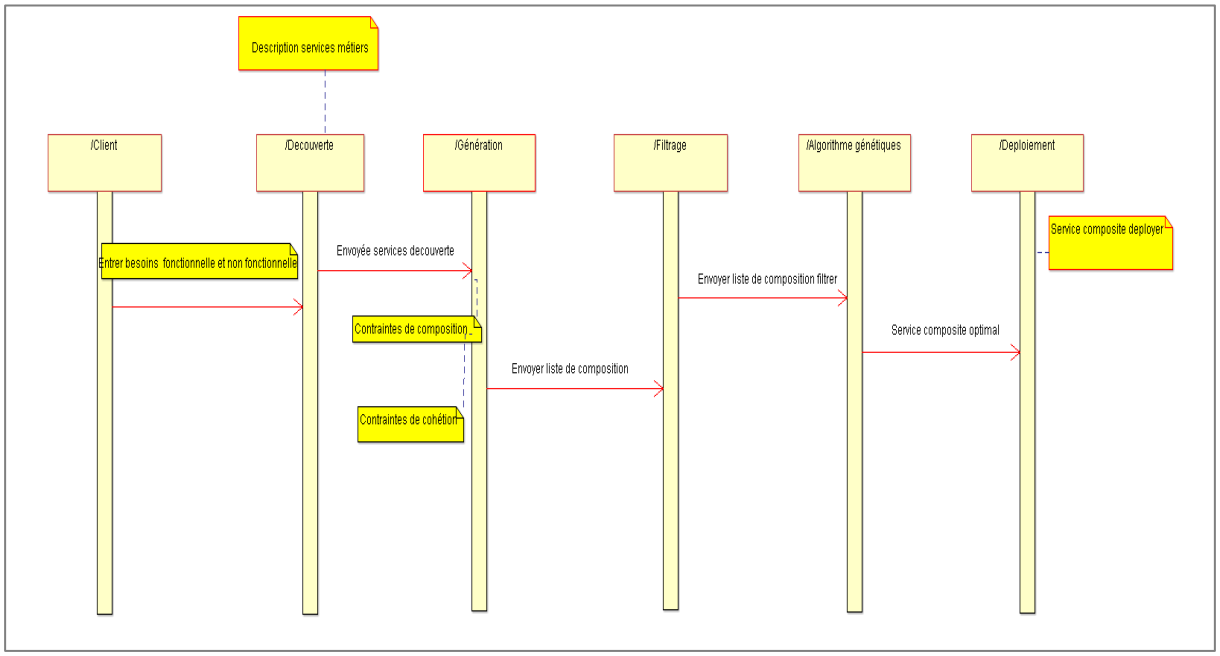

Figure 3.12: diagramme de séquence pour un scénario de composition automatiquement des services dans le cloud.

Le fonctionnement de l'approche est schématisé sur la Figure 3.12, est comme suit :

- Le développeur introduit les besoins fonctionnelles et non fonctionnelles
- Lancer la découverte pour construire liste de comparaison possible des services métiers avec la description des services métiers
- Générer liste de composition à partir des services métiers découverts pendant la phase président, en prenant en considération les contraintes de composition et les contraintes de cohésion.
- Mesurer le niveau de cohésion (applique Équation 3.1) de la liste des services concrets pour filtrer les meilleure candidates.
- Appliquer l'algorithme génétique pour fournir un ensemble de solutions qui respectent les contraintes du client (Service composite optimal).
- Déployer Service composite optimal dans infrastructures IaaS cloud.

3.7. Conclusion

Dans ce chapitre, nous avons présenté une approche pour la composition automatique de services métiers dans le Cloud. Nous avons commencé, tout d'abord, par une conception globale de l'approche et Ensuite, nous avons décrit les différentes phases de l'approche.

Dans le prochain chapitre, nous allons présenter un prototype implémenté en java qui concrétise note approche.

Chapitre 04

Implémentation

4

Chapitre 4 : Implémentation

4.1. Introduction

Dans Ce chapitre nous allons présenter l'implémentation prototype de notre approche. D'abord nous allons présenter les langages et les outils de développement utilisé lors de l'implémentation de ce système, par la suite nous décrivant l'application réalisée sous forme de présentation et de description. Nous terminons ce chapitre par une conclusion.

4.2. Outils logiciels utilisé

Nous avons énuméré au cours de cette partie les différents outils logiciels utilisés tout au long de phase de programmation

4.2.1. Environnement de développement

- **Netbeans**

NetBeans IDE est un environnement de développement intégré (EDI) modulaire basé sur des normes, écrit dans le langage de programmation Java. Le projet de NetBeans IDE consiste en un EDI Open Source complet écrit dans le langage de programmation Java et en une plate-forme d'application cliente riche, qui peut être utilisée comme structure générique pour créer n'importe quel type d'application. [41]

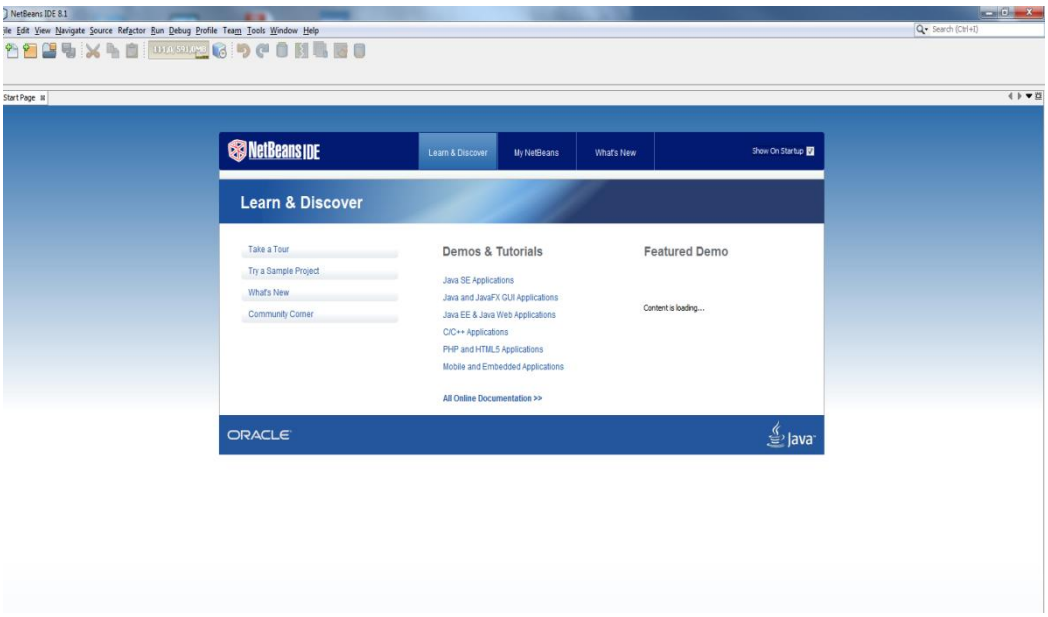

Figure 4.1 : Version de Netbeans-IDE adaptée

- **WampServer**

Nous avons utilisé l'interface PhpMyAdmin, du serveur Wamp [41] **(**voir figure 4.2**).** PhpMyAdmin est une interface de gestion de base de données MySQL. Cet outils nous a permet la création des déférents table de notre BD, afin de la connecté avec notre application Netbeans.

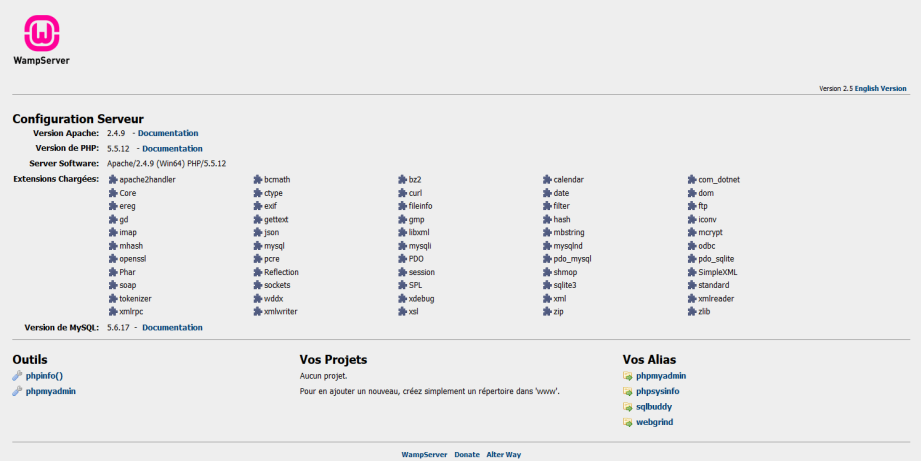

Figure 4.2 : GUI de phpMyAdmin.

4.2.3. Les Langages de programmation utilisés

- **JAVA**

Pour l'implémentation de notre application nous avons opté pour le langage Java (version du JDK : 1.7). Ce langage est apparu vers la fin de l995 et obtint rapidement un énorme succès .Nous nous sommes donc orientés vers ce langage pour le développement de notre Système car il[42]:

- Est un langage orienté objet, c'est à dire que nous n'allons pas manipuler des fonctions et des procédures mais des objets qui vont s'échanger des messages. Le principal avantage de cette orientation est la capacité de réaliser une programmation modulaire (tous les objets peuvent être mis au point séparément).
- Permet un accès simplifié aux bases de données, soit à travers la passerelle JDBCODBC ou à travers un pilote JDBC spécifique au SGBD.

- Est particulièrement adapté au développement d'application communiquant à l'intermédiaire d'un réseau. L_API java est très riche, différents packages permettant d'accéder aux réseaux, aux entrée/sortie et aux différents composants graphiques.
- Assure une totale indépendance des applications vis à vis l'environnement d'exécution. C'est à dire que toute machine supportant Java est en mesure d'exécuter un programme sans aucune adaptation (ni recompilation, ni paramétrage de variables d'environnement).

4.2.4. Outils de déploiement et de configuration des services

Pour déployer le meilleur service composite, nous avons utilisé Docker comme un outil de déploiement et de configuration.

- **Docker** 

Solon [39]: « *Docker is an open platform for developers and system admins to build, ship, and run distributed applications* »

Docker est un moyen d'enfermer les services dans des environnements isolés, appelés conteneurs, ces services peuvent être emballés avec leurs besoins en termes de bibliothèques et de dépendances. Les services peuvent s'exécuter dans n'importe quelle machine à condition que le Docker soit installé.

Docker est un projet Open Source permettant d'automatiser le déploiement d'applications en tant que conteneurs portables et autonomes pouvant fonctionner sur le cloud ou sur un site. Docker travaille en collaboration avec les fournisseurs de cloud, Linux et Windows, y compris Microsoft.

- **Architecture du Docker**

Docker utilise une architecture client-serveur. Le client Docker communique avec Docker daémon, qui se charge de la construction, de l'exécution et de la distribution de vos conteneurs Docker. Le client et le démon Docker peuvent s'exécuter sur le même système ou vous pouvez connecter un client Docker à un daémon Docker distant. Le client et le daémon Docker communiquent à l'aide d'une API REST, via des sockets UNIX ou une interface réseau. [39]

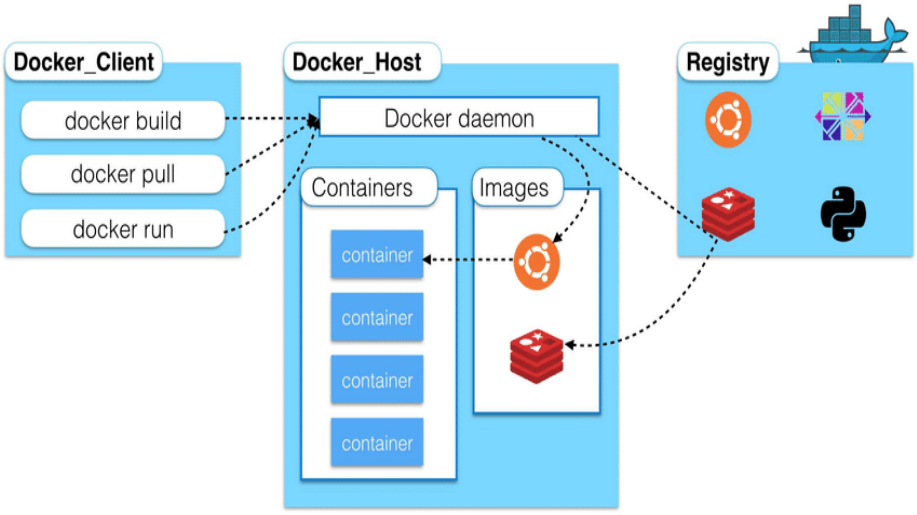

Figure 4.3:Architecture du Docker.

- **Docker Client**

Docker Client est l'utilisateur ou le développeur final de Docker.

- **Docker Daemon**

Docker daemon est celui qui exécute réellement les commandes envoyées au Docker Client, par exemple la construction, l'exécution et la distribution des conteneurs. Le Docker daemon fonctionne sur la machine hôte. Nous ne pouvons pas communiquer directement avec le daemon. Le Docker client, qui fonctionne sur la machine hôte, peut également communiquer avec le daemon.

Le Docker daemon est un service qui s'exécute sur le système d'exploitation hôte. Il ne fonctionne actuellement que sous Linux car il dépend d'un certain nombre de fonctionnalités du noyau Linux, mais il existe également plusieurs façons d'exécuter Docker sous Mac OS ou Windows [39].

- **Conteneurs**

Un conteneur est une instance exécutable d'une image, pouvons être créé, démarrer, arrêter, déplacer ou supprimer un conteneur à l'aide de l'API Docker ou de l'interface de ligne de commande, ainsi connecter un conteneur à un ou plusieurs réseaux, lui associer un stockage ou même créer une nouvelle image en fonction de son état actuel. Par défaut, un conteneur est relativement bien isolé des autres conteneurs et de sa machine hôte, et contrôler

l'isolement du réseau, du stockage ou d'autres sous-systèmes sous jacents d'un conteneur à partir d'autres conteneurs ou de la machine hôte.

Un conteneur est défini par son image ainsi que toutes les options de configuration que vous lui fournissez lorsque vous le créez ou le démarrez. Lorsqu'un conteneur est supprimé, les modifications de son état qui ne sont pas stockées dans le stockage persistant disparaissent [39].

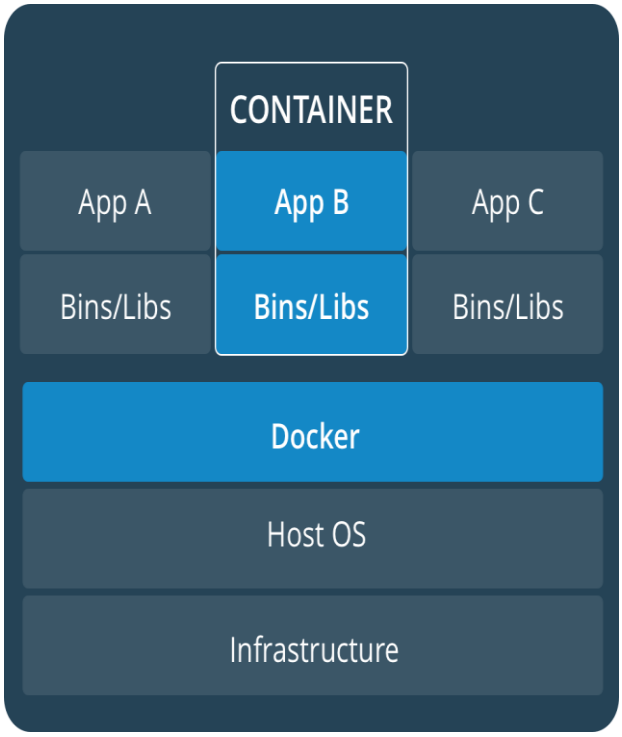

Figure 4.4: Les conteneurs

Nous avons choisi d'utiliser docker comme un outil de déploiement, par ce qu'il est une plateforme Open Source, elle permet d'automatiser le déploiement des services en tant que conteneurs portables et autonomes. Ces derniers peuvent fonctionner sur le cloud ou sur un site web.

1. Docker est une solution logicielle
2. Docker est une couche d'abstraction qui permet de dématérialiser la fonction serveur.
3. Docker simplifie la logistique liée à l'exploitation des logiciels.
4. Docker est un mini-cloud dans lequel des conteneurs sont exécutés.
5. Très adapté au cycle de vie itératif du logiciel.
6. Grande modularité, chaque conteneur étant isolé.

7. Une « scalabilité » grâce aux déploiements automatiques.
8. Partage des applications simplifiées.

4.3. Présentation de l'application

Nous allons présenter dans cette partie les interfaces principales de notre l'application.

4.3.1. Fenêtre d'accueil

C'est la fenêtre d'accueil de l'application ou le client lance notre application (voir la figure 4.5):

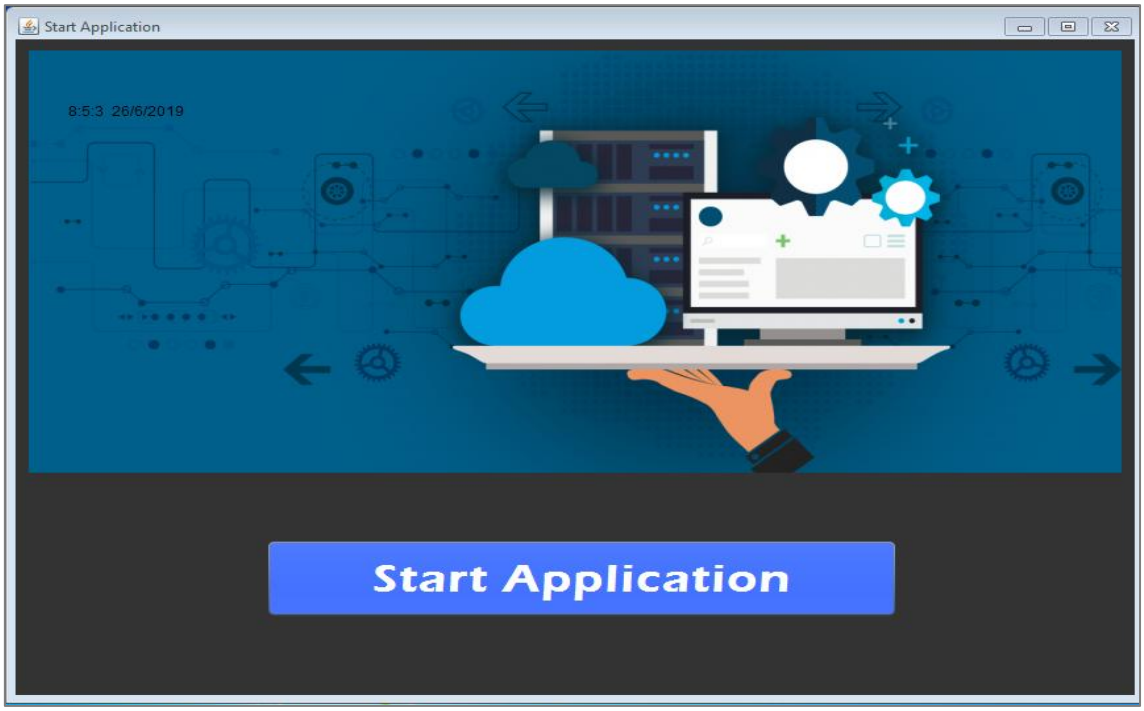

Figure 4.5:Fenêtre d'accueil

4.3.2. GUI pour les besoins fonctionnels du client

Dans cette section nous allons présenter les interfaces qui lui permet de spécifier ses besoins fonctionnels tels que le type d'activité, les entrées/sorties du processus "Commande de produit en ligne" composé des services, Par exemple, un client a besoin d'un processus "Vente" ayant comme entrées 'information client', 'désignation produit', 'type de paiement'……etc. , et comme sortie 'ticket de réception et d'achat'. Le moteur de découverte générer liste de combinaison possible des services métiers en se basant sur les besoins fonctionnels du client.

Les interfaces suivantes (voir Figure 4.6 et Figure 4.7), présente comme entrer la désignation de produit

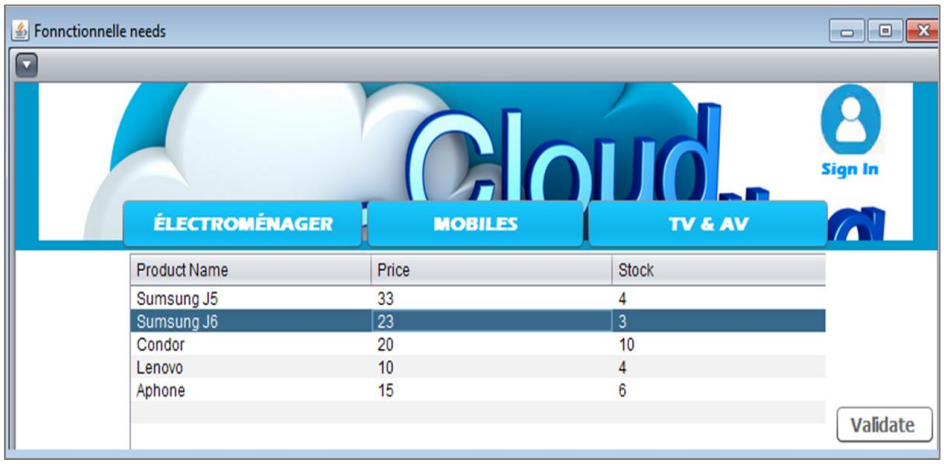

Figure 4.6 : Sélection un produit

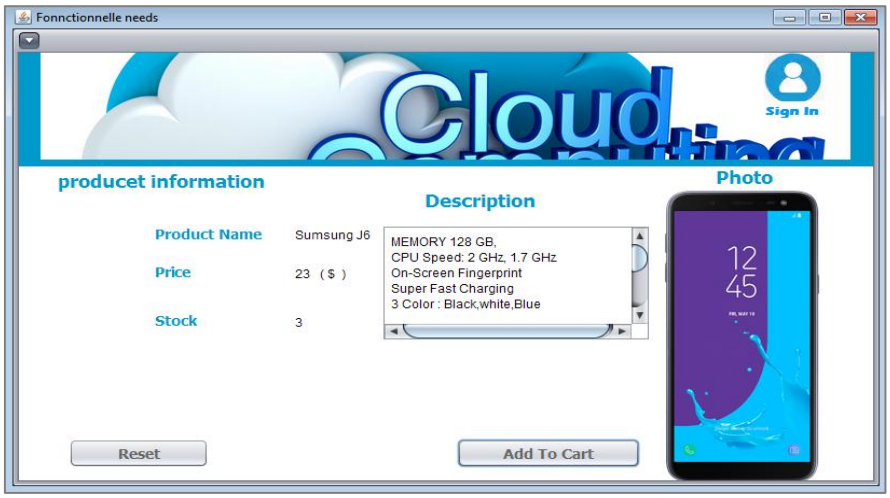

Figure 4.7: Information de produit

Les interfaces suivantes (voir Figure 4.8), présente comme entrer type de paiement

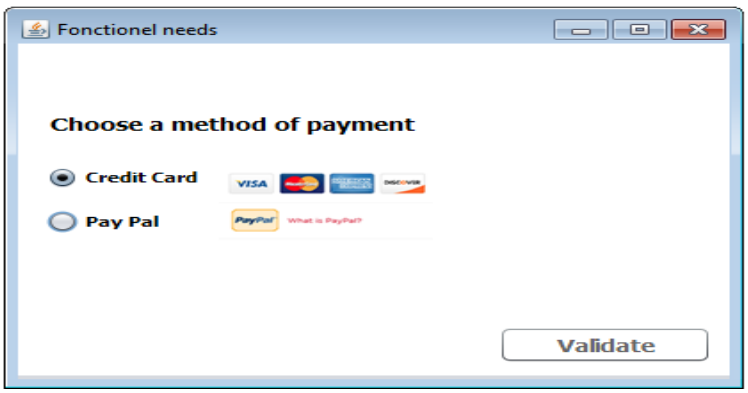

Figure 4.8 : Type de paiement

Les interfaces suivantes (voir Figure 4.9), présente opération de livraison

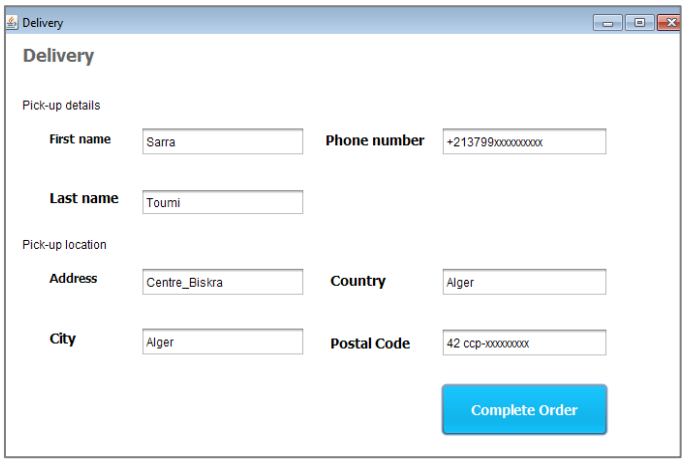

Figure 4.9: Fenêtre de livraisons

4.3.3. Interface pour la spécification des préférences QoS

Le client doit spécifier l'ordre de ses préférences comme indiqué sur la figure 4.10.

Figure 4.10: La spécification des préférences QoS

La figure suivante représente le résultat (Service composite optimal sélectionné) possédant les meilleures Scores des paramètres de QoS :

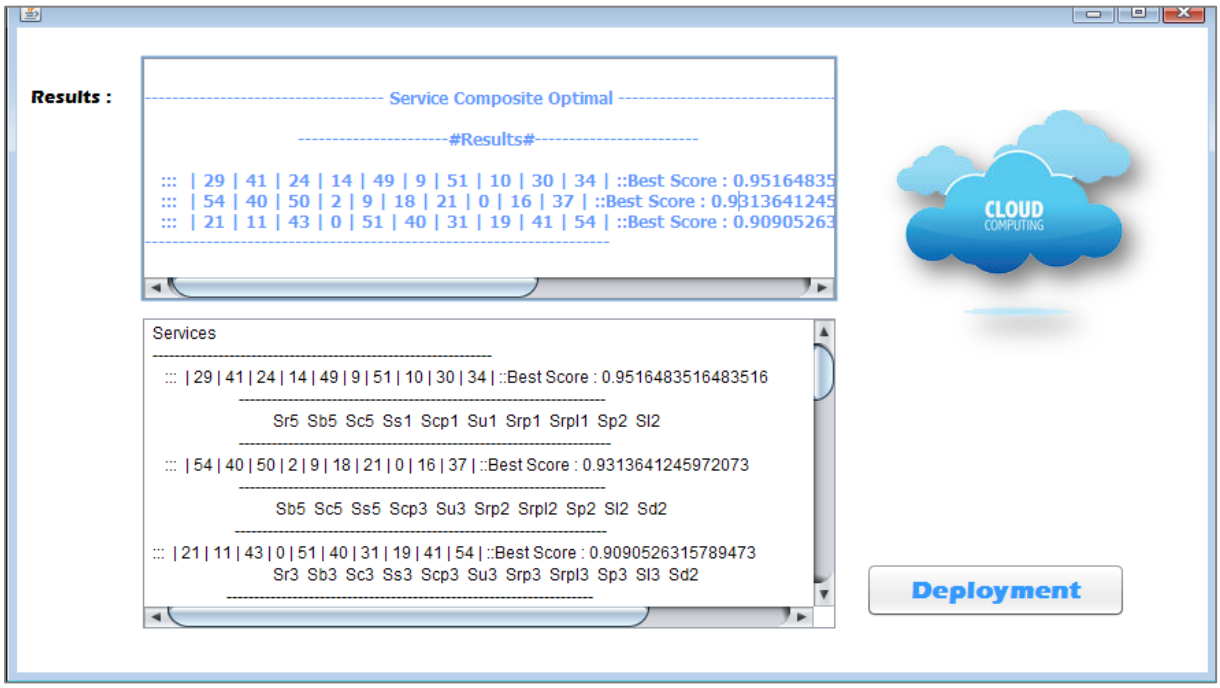

Figure 4.11: Affichage des résultats obtenus par notre application.

La figure suivante représente le script de composition et de déploiement généré, Service composite optimal sélectionné.

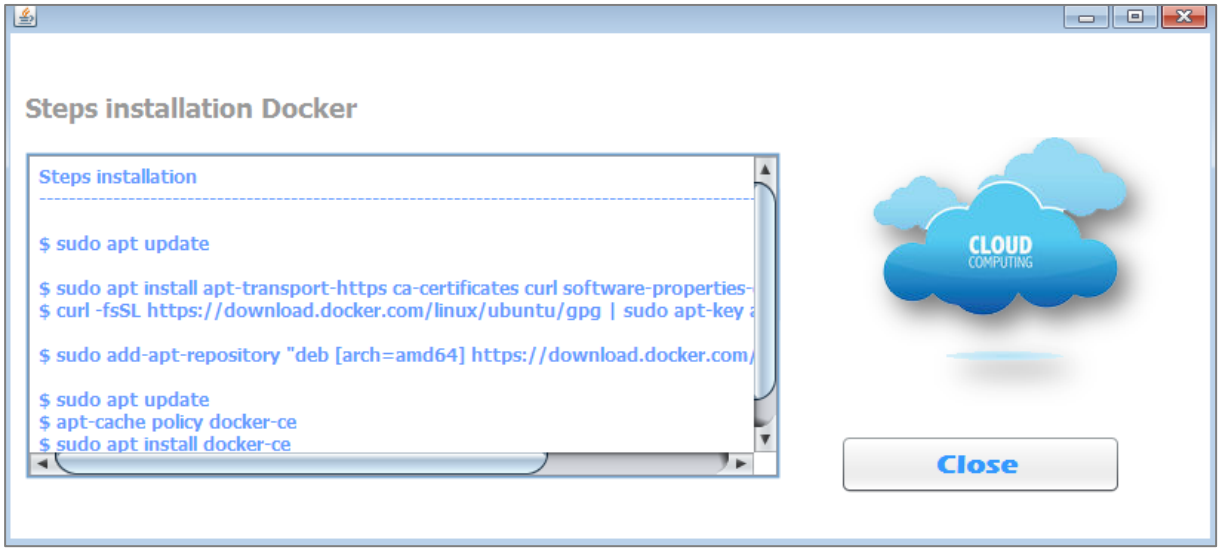

Figure 4.12 : Etape d'installation Docker

4.4. Discussion des résultats

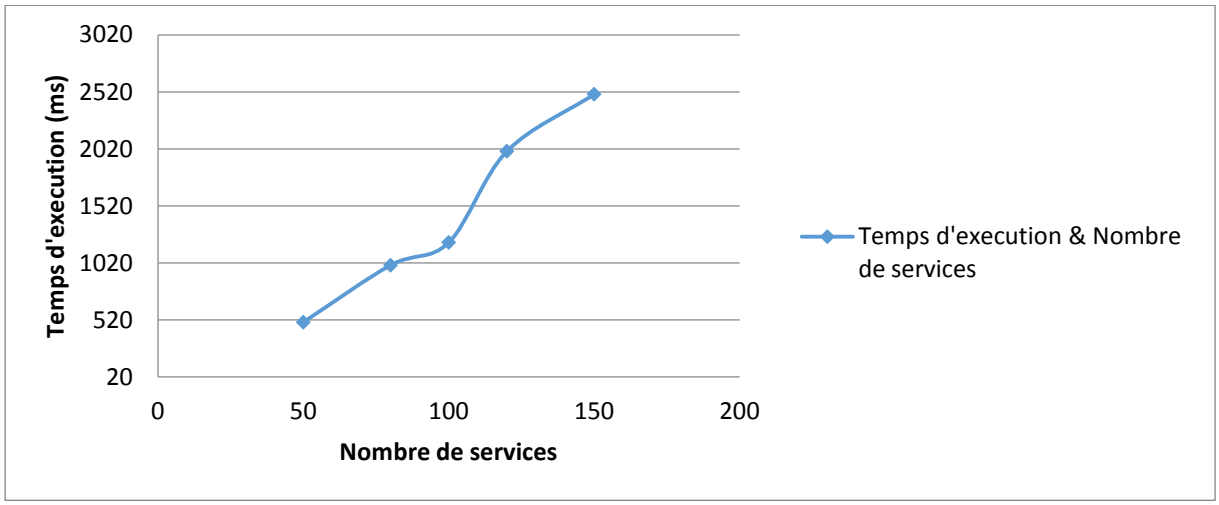

Figure 4.13:Temps d'exécution & nombre de services

La figure précédente (voir figure 4.13) décrit le temps d'exécution pour un nombre de services (tel qu'il est indiqué dans l'axe des nombres de services), et le temps d'exécution nécessaire pour trouver le service composite optimal, situé au niveau de l'axe y.

La courbe apparente dans la figure montre une relation quasi linéaire, qui indique l'efficacité de l'approche proposée pour s'adapter à la composition de services dans des environnements à grande échelle. À chaque fois que l'échelle de l'environnement augmente, le temps d'exécution augmente de façon linéaire et non exponentielle. Cela prouve la robustesse et l'efficacité de l'approche proposée pour le déploiement dans des environnements à grande échelle.

4.5. Conclusion

Dans ce chapitre, nous avons présenté toutes les étapes de la mise en œuvre de notre projet avec tous les outils et les langages de programmation utilisés.

La prochaine section est consacré a conclure ce manuscrit en présent les futurs travaux possibles.

Conclusion générale

L'objectif de ce mémoire était de proposer une approche de composition automatique des services métiers dans le Cloud en se basant sur la description des besoins fonctionnels et non fonctionnels(QoS). Une composition automatique ne nécessitant pas l'intervention humaine. Le résultat prévu sera un service composite déployé sur une IaaS Cloud. D'autre part, notre approche utilise une technique d'optimisation pour générer une composition optimale tout en éliminant les services redondants.

Dans ce travail, nous avons fait, en premier temps, un survol sur l'état de l'art qui consiste à présenter les définitions sur la technologie de Cloud ainsi que les caractéristiques, les modèles de services et leurs déploiements…etc.

Dans le deuxième chapitre, nous avons présenté les méthodes de composition des services et afin de montrer l'intérêt de notre travail, nous avons donné une présentation avec une synthèse sur les travaux réalisés pour la résolution du problème de composition de services.

Au troisième chapitre, nous avons décrit les différentes parties de l'approche proposée. En outre, nous avons défini les paramètres utilisés pour ajuster les algorithmes implémentés.

Finalement, et pour objectif de valider notre conception, dans le dernier chapitre nous avons montré les outils utilisés pour arriver à des résultats. Nous avons montré aussi quelques interfaces qui affichent les opérations du système proposé. Pour finaliser la mises-en œuvre, nous avons introduit une discussion sur les résultats obtenus afin d'analyser celles collectés.

Nous estimons que la présente approche constitue une base de travail pour plusieurs perspectives connexes. En fait, l'intégration des ontologies pour offrir au client d'exprimer librement ses requêtes avec plus de contraintes. D'autre part, notre approche peut être améliorée de manière qu'elle soit extensible à d'autres critères de QoS.

Bibliographie

[1] : Berkani, N., & Moussaoui, S. (2016). La sécurité des données dans le Cloud Computing. Mémoire de Master. université Abderrahmane Mira.

[2] : Kuhn, A. L. (2009).Cloud Computing.

[3] : Menken, I. (2008). Cloud Computing-The Complete Cornerstone Guide to Cloud Computing Best Practices Concepts, Terms, and Techniques for Successfully Planning, Implementing.Enterprise IT Cloud Computing Technology. Emereo Pty Ltd.

[4] : Figer, J. (2012). L'informatique en nuage [Cloud Computing], mode ou révolution? Article scientifique.

[5] : Armbrust, M., & Griffith, A. (2009). A Berkley View of Cloud Computing. In *UCB/EECS* (pp. 2009-28). EECS Department, UCB.

[6] : AOUAMEUR, Z., & TAHRINE, H. (2012). Comparaison et mise en place des plateformes de Cloud Computing OpenStack et Eucalyptus. Université Kasdi Merbah de Ouargla, Année Universitaire, 2013.

[7] : Cloud computing, Wikipédia. https://fr.wikipedia.org/wiki/Cloud_computing, consulté le 1/02/2019.

[8] : Delannoy, C. (2012). Programmer en Java: Java 5 à 7. Editions Eyrolles.

[9] : Ce que vous devriez savoir sur le cloud computing (owasp quebec). https://fr.slideshare.net/PatrickLeclerc/ce-que-vous-devriez-savoir-sur-le-cloud-computing-owasp-quebec, consulté le 11/1/2019.

[10] : Sajid, M., & Raza, Z. (2013, December). Cloud computing: Issues & challenges. In International Conference on Cloud, Big Data and Trust (Vol. 20, No. 13, pp. 13-15).

[11] : Miller, M. (2008). Cloud computing: Web-based applications that change the way you work and collaborate online. Que publishing.

[12] : BACHTARZI, F. (2014). Une Approche de Composition des Services Web Basée Transformation de graphes. Thèse de doctorat. Université costantine, *2*.

[13] : Dodani, M. H. (2004). From Objects to Services: A Journey in Search of Component Reuse Nirvana. Journal of Object Technology, 3(8), 49-54.

[14] : Hashimi, S. (2003). Service-oriented architecture explained. ONDotnet. com (online)(geciteerd: 20 maart 2006) Beschikbaar op URL: http://www. matcom. uh.

cu/Weboo/_Rainbow/Documents/ONDotNet. com_% 20Service-Oriented% 20Architecture% 20Explained. pdf.

[15] : Endrei, M., Ang, J., Arsanjani, A., Chua, S., Comte, P., Krogdahl & all. (2004). Patterns: service-oriented architecture and web services (pp. 17-44). IBM Corporation, International Technical Support Organization.

[16] : IBM Services Architecture Team (September 06, 2000) Web services architecture, http://www.ibm.com/developerworks/library/w-ovr/, consulté le 02/2019.

[17] : Mrissa, M. (2007). Médiation sémantique orientée contexte pour la composition de services Web (Doctoral dissertation, THESE doctorat, LIRIS).

[18] :W3C-WSA-Group, W3C (2004) "Web Service Architecture", http://www.w3.org/TR/ws-arch, consulté le 19/12 / 2018,

[19] : T. Erl: "SOA Principles of Service Design", 2007.

[20] : N.M. Josuttis , " SOA in Practice", O'Reilly Media, august 2007.

[21] : Pourraz, F. (2007). Diapason: une approche formelle et centrée architecture pour la composition évolutive de services Web (Doctoral dissertation, PhD thesis, Université de Savoie, LISTIC).

[22] : Badri, L., & Badri, M. (2004). A Proposal of a new class cohesion criterion: an empirical study. Journal of Object Technology, 3(4), 145-159.

[23] : Benfenatki, H., Da Silva, C. F., Kemp, G., Benharkat, A. N., Ghodous, P., & Maamar, Z. (2017). MADONA: a method for automated provisioning of cloud-based component-oriented business applications. Service Oriented Computing and Applications, 11(1), 87-100.

[24] : Tsai, W. T., Sun, X., & Balasooriya, J. (2010, April). Service-oriented cloud computing architecture. In 2010 seventh international conference on information technology: new generations (pp. 684-689). IEEE.

[25]: Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2008). Service-oriented computing: a research roadmap. International Journal of Cooperative Information Systems, 17(02), 223-255.

[26]: Kerdoudi, M. L. (2016). From Web components to Web services: opening development for third parties (Doctoral dissertation, Université de Biskra).

[27] : Marin, C. (2008). Une approche orientée domaine pour la composition de services (Doctoral dissertation, Université Joseph Fourier (Grenoble)).

[28] : Bruijn, J. D., Fensel, D., Kerrigan, M., Keller, U., Lausen, H., & Scicluna, J. (2008). Modeling semantic web services: the web service modeling language. Springer Publishing Company, Incorporated.

[29] : Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., & Xu, X. (2014). Web services composition: A decade's overview. Information Sciences, 280, 218-238.

[30] : Fluegge, M., Santos, I. J., Tizzo, N. P., & Madeira, E. R. (2006, July). Challenges and techniques on the road to dynamically compose web services. In Proceedings of the 6th international conference on Web engineering (pp. 40-47). ACM.

[31] : Benfenatki, H. (2016). Méthodologie de provisionnement automatique d'applications métier orientées service sur les environnements cloud (Doctoral dissertation, Université de Lyon).

[32] : Nguyen, D. K., Lelli, F., Papazoglou, M. P., & Van Den Heuvel, W. J. (2012). Blueprinting approach in support of cloud computing. Future Internet, 4(1), 322-346.

[33] : Jula, A., Sundararajan, E., & Othman, Z. (2014). Cloud computing service composition: A systematic literature review. Expert systems with applications, 41(8), 3809-3824.

[34] : Vakili, A., & Navimipour, N. J. (2017). Comprehensive and systematic review of the service composition mechanisms in the cloud environments. Journal of Network and Computer Applications, 81, 24-36.

[35] : Mohsni, T., Brahmi, Z., & Gammoudi, M. M. (2016, November). Data-intensive service composition in Cloud Computing: State-of-the-art. In 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA) (pp. 1-7). IEEE.

[36]: Li, J., Zheng, X. L., Chen, S. T., Song, W. W., & Chen, D. R. (2014). An efficient and reliable approach for quality-of-service-aware service composition. Information Sciences, 269, 238-254.

[37]: Seghir, F., & Khababa, A. (2018). A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition. Journal of Intelligent Manufacturing, 29(8), 1773-1792.

[38] : Karimi, M. B., Isazadeh, A., & Rahmani, A. M. (2017). QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm. The Journal of Supercomputing, 73(4), 1387-1415.

[39] : (Docker, 2018), Docker, http://www.docker.com, accédé le 05/06/2018

[40] : Notes de version de NetBeans IDE 8.1. https://netbeans.org/community/releases/68/relnotes_fr.html, consulté le 9/5/ 2019.

[41] : Site web: http://www.wampserver.com, consulté le 9/5/ 2019.

[42] : Claude.d.(2008).Programmer en java.Edition EYROLLES