REMERCIEMENT

En préambule à ce mémoire nous remerciant ALLAH qui nous aide et nous donne la patience et le courage durant ces langues années d'étude.

La première personne que nous tenons à remercier est notre encadrant ***Mme. Sahli Sihem,*** *pour l'orientation, la confiance, la patience qui a constitué un apport considérable sans lequel ce travail n'aurait pas pu être menée au bon port. Qu'il trouve dans ce travail un hommage vivant à sa haute personnalité.*

Je remercie les honorables membres de jury d'avoir accepté d'être membre de mon jury de thèse, d'évaluer mon travail et pour nous avoir honorés de leur présence

Je remercie aussi ma famille, mes amis et tous les enseignants du département d'informatique qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études.

Enfin, on remercie tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail

CHetti Sami

Dédicace

C'est avec honneur et énorme joie je dédie ce modeste travail :

Á mes très chers parents pour leurs interminables conseils encouragement continu, patience illimitée et soutien en témoignage de ma reconnaissance et mon affection, dans l'espoir que vous en serez fiers. Ce travail est le fruit de vos sacrifices que vous avez consentis pour mon éducation.

A la mémoire de mon cher frère Rami

Á mes chères sœurs Meriem et Yasmine.

Á mes cousins : Yakoub, Achraf, Adam, Karim, Rahma, Asma, Ilham.

Á mes amis durant mes années d'étude qui m'a beaucoup encouragé et surtout brahim, yassine, abdl hakim, anes, bilal, abdl kader hamza , farouk , faycel , amin ,mahydine ,mehdi ,amir ,zoka ,kadirou , asma , ramla , amel , samia ,djamel ,tarek , takfarinass,walid , ahmed, houssem ,kamel ,salah , haroun, oussma ….

Á tous ceux qui me sont chers. Á tous ceux qui ont contribué d'une manière ou d'une autre à l'élaboration de ce travail.

Sami Chetti

Table des matières

Chapitre II : La Composition des Services Web

Chapitre III : Analyse et conception du Système

LISTE DES FIGURES :

Chapitre I : Services Web

Chapitre II : La Composition des Services Web

Chapitre III : Analyse et conception

Chapitre V : Implémentation du système

LISTE DES Tableaux :

Glossaire

B

B2B: Business-to- Business

B2C: Business-to- Consumer

BPMI: Business Process Management Initiative

BPML: Business Process Modeling Language

BPEL4WS: Business Process Execution Language for web Services

C

COM: Component Object Model

COBRA: Common Object Request Broker Architecture

D

DAML: DARPA Agent Markup Language

DCOM: Distributed Component Object Model

F

FTP: file transfer protocol

H

HTML: Hyper Text Markup Language

HTN: Hierarchical Task Network

HTTP: Hyper Text Transfer Protocol

O

OWL: Ontology Web Language

OWL-S: Ontology Web Language for Services

S

SMTP: Simple Mail Transfer Protocol

SOA: Service Oriented Architecture

SOAP: simple Object Access protocol

U

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

UDDI: Universal Description Discovery and Integration

W

WSCI: web services Choreography Interface

WSDL: Web Services Description Language

WSDL-S: Web Services Description Language-Semantic

WSFL: Web Services Flow Language

X

XML: Extensible Markup Language

Résumé

Un service web peut être défini comme un programme autonome qui s'exécute sur le web. Les web services sont décrits par des descriptions WSDL, qui sont enregistrées dans des registres UDDI afin de faciliter leurs recherches (découverte) par la suite. Ces derniers sont des applications accessibles sur Internet réalisant chacune une tâche spécifique. Pour fournir une solution à une tâche complexe, on peut regrouper des services web pour n'en former qu'un seul, on parle alors de composition de services web. Créer des compositions de services (des services composites) signifie ordonner les invocations aux opérations, router les messages, modifier les paramètres et gérer les exceptions. Donc la composition des services web requiert plusieurs tâches: la description, la découverte des services, l'orchestration (l'exécution), la surveillance, et la coordination de l'échange des différents messages (informations). L'objectif de ce travail est de développer un système qui vise à composer des services web. La prise en compte des informations sur les fonctionnalités et les caractéristiques des services pour développer le processus de composition est l'une des caractéristiques essentielles du système de composition.

Mots Clé: SOA, Service Web, Composition des Services Web, Ordre d'exécution de la composition

Introduction générale

Introduction générale

Introduction générale

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et particulièrement par la démocratisation de l'accès à l'internet. Cette évolution du monde informatique a entrainé le développement de nouveaux paradigmes d'interaction entre applications. Notamment, l'architecture orientée service (Service Oriented Architecture, ou SOA) qui a été mise en avant afin de permettre des interactions entre applications distantes. Cette architecture est construite autour de la notion de service, qui est matérialisé par un composant logiciel assurant une fonctionnalité particulière et accessible via son interface. Un service est toujours accompagné d'une description fournissant aux applications les informations nécessaires à son utilisation. Les services web, tels qu'ils sont présentés, sont conceptuellement limités à des fonctionnalités relativement simples qui sont modélisés par une collection d'opérations. Toutefois, pour certains types d'applications, il est nécessaire de combiner un ensemble de services web (Services Web simples) en services web plus complexes (services web agrégés ou composites) afin de répondre à des exigences plus complexes. L'objectif de la composition de service est de créer de nouvelles fonctionnalités en combinant des fonctionnalités offertes par d'autres services existants, composés ou non en vue d'apporter une valeur ajoutée. Un service web est dit composite lorsque son exécution implique des interactions avec d'autres services web, et des échanges de messages entre eux afin de faire appel à leurs fonctionnalités. La composition de services web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'interaction.

Il y'a plusieurs approches pour réaliser la composition des services web, mais dans notre travail on a utilisée Langage d'exécution des processus d'affaires BPEL ,cet langage permet d'organiser plusieurs services web fourni par un utilisateur à partir d'un service actuel ,la communication entre l'utilisateur et notre système ce faire à partir d'un environnement s'appelle Open ESB

Il permet de s'ouvrir à de nouvelles possibilités d'automatisation d'une grande quantité d'information sur le Web. Notre mémoire se compose de quatre chapitres :

Le premier chapitre est consacré aux notions fondamentales des Web services où nous avons défini l'architecture SOA et les standards de description, de publication et d'invocation de Web services, son fonctionnement et son architecture.

Introduction générale

Dans le deuxième chapitre, nous allons présenter la composition des services web, ses types, quelque langages et approches proposées dans la littérature pour la résolution du problème de composition.

Le troisième chapitre représente la conception de notre système, son architecture et ces fonctionnalités principales.

Le quatrième chapitre traite la mise en œuvre du système, son environnement de développement (les outils utilisés) et l'implémentation de notre système. Nous terminerons notre mémoire par une conclusion générale et nous énoncerons quelques perspectives de recherche.

Chapitre 01 : Services Web

Chapitre 01 : Service Web

1. Introduction

La technologie des services Web (SW) est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA (Architecture Orientée Services). D'autres technologies telles que RMI, DCOM et CORBA ont précédemment adopté ce style architectural mais ont généralement échoué en raison de la diversité des plates-formes utilisées dans les organisations et aussi parce que leur usage n'était pas adapté à Internet (problème de passage à travers des Firewalls, etc.) d'où la lenteur, voire l'absence de réponses sur ce réseau. Les applications réparties fondées sur ces technologies offrent des solutions caractérisées par un couplage fort entre les objets. Les solutions proposées par les services Web, permettent néanmoins un couplage moins fort. De plus, l'utilisation des technologies standards du Web tels que l'HTTP et XML par les services web facilite le développement d'applications réparties sur Internet, et permet d'avoir des applications très faiblement couplées. L'intégration est sans doute le facteur essentiel qui favorise l'utilisation des SW.

Dans la suite de ce chapitre, nous présenterons tout d'abord le concept des services web ainsi que son fonctionnement, Architecture en couches, et les Principaux standards utilisé dans les services Web.

2. Architecture orientée service (SOA)

L'architecture orientée services (SOA) est une architecture logicielle s'appuyant sur un ensemble de composants simples appelés **Services Web**. Son objectif est de décomposer une fonctionnalité complexe en un ensemble de fonctionnalités basiques, fournies par des services Web et de décrire finement le schéma d'interaction entre ces services Web. SOA implique quatre entités d'interaction : les services Web, les fournisseurs de services, les annuaires de service et les demandeurs des services auxquels nous rajoutons les attributs. [1]

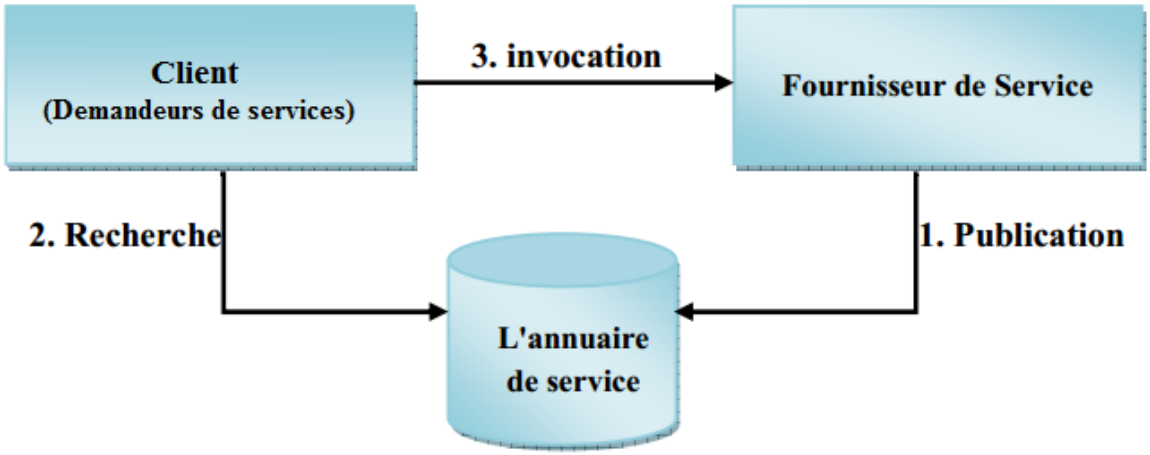

Figure [I.1] – Modèle Fonctionnel de l'architecture SOA. [1]

2.1. Définition de SOA

Selon [2], SOA peut être défini selon deux vision, une métier et l'autre technique :

- *Selon la vision métier :*

« L'architecture orientée service est un ensemble de méthodes techniques, métiers, procédurales, organisationnelles et gouvernementales pour réduire ou éliminer les frustrations avec les technologies d'information, et pour mesurer quantitativement la valeur métier des technologies d'information, pendant la création d'un environnement métier agile pour un intérêt concurrentiel. »

- *Vision technique :*

« L'architecture SOA est un paradigme permettant d'organiser et d'utiliser des savoir faires distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoirs faires pour produire le résultat désiré avec des pré-conditions et des buts mesurables. »

2.2. Propriétés de SOA

Le principe d'une architecture orientée services consiste à structurer le système d'information de l'entreprise comme un ensemble de services qui exposent leur interface fonctionnelle et qui communiquent par messages. Parmi les caractéristiques que nous jugeons fondamentales de la SOA nous citons :

2.2.1. La notion de couplage

Le couplage désigne une connexion ou une relation entre deux choses. Une mesure du couplage est comparable à un niveau de dépendance. Ce principe préconise la création d'un type spécifique de relation à l'intérieur et à l'extérieur des frontières de service, en insistant toujours sur la réduction («relâchement») des dépendances entre le contrat de service, sa mise en œuvre et ses consommateurs de services **[**3**]**.

Le but du principe de couplage **faible** entre les services est de garder une faible liaison entre les services, pour atteindre un haut degré de flexibilité dans l'architecture. Etant donné que les dépendances entre services sont le plus possible évitées, ces dernières gardent une propre autonomie, qui permet de remplacer les services ou de les relier entre eux pour créer des processus facilement. **[4]**

Le couplage faible permet en cas de disfonctionnement d'un service de ne pas faire arrêter tout le système, parce que les services ne sont pas dépendants ou parce que le service dysfonctionnant peut être facilement remplacé. [4]

2.2.2. Le principe d'abstraction

L'abstraction est liée à de nombreux aspects de l'orientation vers le service. Sur un plan fondamental, ce principe souligne la nécessité de cacher autant de détails sous-jacents d'un service que possible [3].

Le principe d'abstraction consiste à cacher l'information, liée à un service, qui n'est pas nécessairement requise lors de l'utilisation de ce dernier. Le service sera ainsi vu par le consommateur comme étant une boite noire, cachant toute information d'ordre logique ou technique propre au service. Le degré d'abstraction d'un service doit être ajusté de façon à ne pas mettre en cause le potentiel de ce dernier à être réutilisé [5].

2.2.3. La notion de réutilisabilité

La réutilisation est fortement recommandée dans le cadre de l'orientation vers les services; À tel point qu'il devient une partie essentielle des processus typiques d'analyse et de conception des services et constitue également la base de modèles de service clés [3].

La réutilisation des services indique la possibilité de réutiliser des composants dans plusieurs scénarios pour atteindre plusieurs buts. la SOA propose de généraliser les services à travers une dissociation des fonctionnalités de tout type d'application ou de processus afin qu'ils puissent être réutilisés. La caractéristique de la réutilisation des services permet de

baisser les coûts de maintenance de l'architecture. Augmenter la réutilisation implique le développement de services qui présentent plusieurs contrats d'utilisation, ces services peuvent couvrir par conséquent plusieurs scénarios d'utilisation. Les services d'une SOA ne doivent pas être doublés pour être utilisés deux fois et ne doivent pas être contenus dans chaque processus qui le nécessite [4].

2.2.4. L'autonomie d'un service

Le principe de l'autonomie de service soutient la mesure dans laquelle d'autres principes de conception peuvent être efficacement réalisés dans des environnements de production réels en favorisant des caractéristiques de conception qui augmentent la fiabilité d'un service et la prévisibilité comportementale [3].

Le terme « autonome » se réfère à la capacité à prendre des décisions ou exécuter des tâches sans appui extérieur. Appliqué au contexte de l'architecture orientée service, cet adjectif indique la capacité des services à utiliser ses ressources pour exécuter le logique métier qu'ils déploient sans l'appui de participants externes. Grâce à leur caractéristique d'autonomie, les services entretiennent un contrôle très significatif sur l'environnement qu'ils déploient et ses limites techniques sont définies de manière significative, pour éviter des redondances de services. Cette notion est liée à la propriété de couplage faible [4].

2.2.5. La compossibilité de services

La compossibilité est un concept fondamental dans la conception de logiciels. Ce concept permet de réduire la complexité de la logique des solutions informatiques et de les décomposer en modules plus simples. Ces modules peuvent être recomposés suivant d'autres configurations pour résoudre d'autres problèmes. Dans SOA, un service est considéré comme composable, s'il a l'aptitude d'être un membre effectif dans une composition d'autres services ou composants logiciel, peu importe la nature ou la complexité de cette composition [5].

2.3. Objectifs de SOA

L'architecture SOA a trois objectifs importants :

1. Identification des composants fonctionnels.
2. Définition des relations entre ces composants.
3. Etablissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

2.4. Avantages de SOA

SOA possède plusieurs avantages, parmi eux [6], [7] :

- ✓ SOA réside dans le fait que les applications réparties n'ont plus besoin d'un système de middleware réparti commun pour communiquer, mais seulement des protocoles de communication interopérables sur Internet.
- ✓ SOA peut donc être représentée par une interconnexion de multiples points d'accès.
- ✓ Le contrat de service du modèle des SOA s'inspire directement du modèle des contrats professionnels de service. Dans le cas des services web, ce contrat est un document WSDL.
- ✓ Une grande tolérance aux pannes et une maintenance facilitée.
- ✓ Une bonne modularité permettant de remplacer facilement un composant (service) par un autre puisque ça ne gêne pas le fonctionnement des autres services.
- ✓ La possibilité de l'évolution et la réutilisabilité des composants.
- ✓ La découverte des services disponibles, afin d'être sûr que les utilisateurs potentiels découvriront le service dont ils ont besoin.
- ✓ Une réduction du coût de développement et d'intégration d'application.
- ✓ La mise à l'échelle est rendue possible grâce à la découverte et à l'invocation de nouveaux services lors de l'exécution.

L'approche la plus répandue pour mettre en œuvre une architecture SOA est l'utilisation des services Web. Cette technologie vise à la transposition des SOA dans le cadre du Web.

2.5. Limites de l'architecture SOA

Les principales limites d'une architecture orientée services sont [5] :

- ➢ L'aspect sécurité qui est à surveiller surtout dans le cas où les services sont exposés via le Web.
- ➢ Les outils de développement utilisés pour la mise en œuvre d'une SOA expriment souvent un manque de productivité.
- ➢ Les plateformes d'intégration SOA souffrent d'un manque considérable de robustesse. Comme toute technologie, SOA a ses avantages et ses limites, l'important est de bien profiter de ses apports et savoir pallier ses limites, pour cela il faut adopter une stratégie bien déterminée et savoir choisir la manière avec laquelle on va mettre en place cette architecture.

3. Services web

La définition exacte du terme "service web" est encore fortement discutée. Les différentes définitions proposées s'accordent au moins sur l'idée qu'un service web est un nouveau type de composant logiciel ayant la capacité de publier ses fonctions sur Internet sous forme de services, de rendre ces services facilement invocables et de les mettre à disposition par l'intermédiaire de protocoles Internet standardisés (HTTP, XML, …). Ci-dessous, nous citons quelques définitions généralement acceptées et fournies.

Selon IBM : « Les services web sont la nouvelle vague des applications Web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les services web effectuent des actions allant de simples requêtes à des processus complexes. Une fois qu'un service Web est déployé, d'autres applications peuvent le découvrir et l'invoquer ».

Le consortium W3C définit un service Web comme étant :

« 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. ».

Autour de cette définition, la pile standard de protocoles des services Web fournit les fonctionnalités requises par les SOA de la manière suivante :

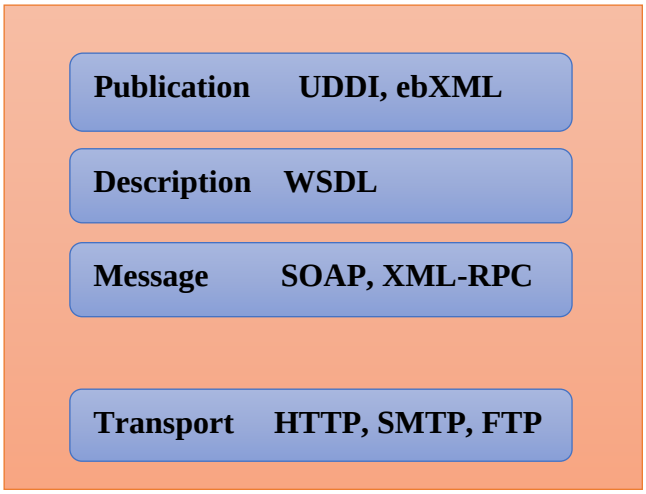

Figure [I.2] – Couches de fonctionnalités et protocoles/langages correspondants.

3.1. Les caractéristiques des services Web

La technologie des services web comporte plusieurs caractéristique sont : [8]

- **Interopérabilité**

  L'interopérabilité présente la caractéristique principale des Services Web. En effet, quels que soient la plate-forme utilisée (Windows, Unix ou autres) et le langage de développement employé, les Services Web se basent sur des standards XML pour simplifier la construction des systèmes distribués et la coopération entre ces derniers.

- **Simplicité d'utilisation**

  L'utilisation des standards tels que XML et HTTP a mis en valeur la simplicité de la manipulation des Services Web et a encouragé les acteurs forts du marché tels que Microsoft et IBM pour intégrer de nouveaux produits.

- **Couplage souple des applications**

  Les Services Web constituent un support d'échange des documents structurés qui traversent les contrôles d'accès dans un environnement hétérogène. La collaboration entre différentes applications afin d'échanger des documents se fait d'une manière directe entre objets.

3.2. Fonctionnalités et Protocoles

3.2.1. La couche publication : repose sur le protocole Universal Description, Discovery, and Integration (UDDI) [9], qui assure le regroupement, le stockage et la diffusion des descriptions de services Web. On notera aussi ebXML3 comme une alternative permettant de fournir les mêmes fonctionnalités.

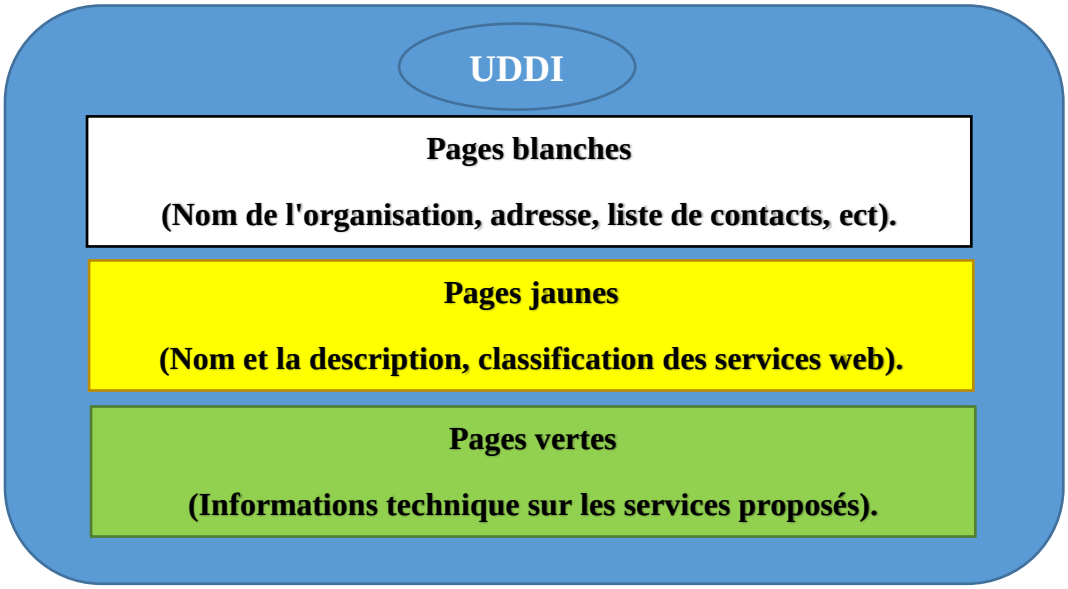

Figure [I.3] – Le contenu de l'annuaire UDDI [9]

L'UDDI est un standard très récent (2000) qui propose une architecture d'appel dynamique aux Services Web. Cette architecture est basée sur un annuaire situé quelque part sur Internet. Le modèle de données UDDI est défini sous forme de schéma W3C XML (Figure 1.5).

1) Les **« businessEntities »** sont en quelque sorte les pages blanches d'un annuaire UDDI et décrivent les organisations ayant publié des services dans le répertoire. On y trouvera notamment le nom de l'organisation, ses adresses (physiques et Web), des éléments de classification, une liste de contacts, etc. Chaque businessEntity est identifiée par une « businessKey » [7].

2) Les **« serviceEntities »** sont en quelque sorte les pages jaunes d'un annuaire UDDI et décrivent de manière non technique les services proposés par les différentes organisations. On y trouvera essentiellement le nom et la description textuelle des services ainsi qu'une référence à l'organisation proposant le service et un ou plusieurs « bindingTemplates » [7].

3) Les **« bindingTemplates »** UDDI permet de décrire des services Web utilisant HTTP, mais également des services invoqués par d'autres moyens (SMTP, FTP...). Les « bindingTemplates » donnent les coordonnées des services. Ce sont les pages vertes de l'annuaire UDDI. Ils contiennent notamment une description, la définition du **point d'accès** (une URL) et les éventuels «tModels » associés [7].

Les « **tModels** » sont les descriptions techniques des services. UDDI n'impose aucun format pour ces descriptions qui peuvent être publiées sous n'importe quelle forme et notamment sous forme de documents textuels (XHTML (Extensible HyperText Markup Language) par exemple). C'est à ce niveau que WSDL intervient comme le vocabulaire choisi pour publier des descriptions techniques des services [10].

La norme UDDI offre aussi une API aux applications clientes. L'API UDDI propose deux fonctionnalités principales : **la recherche** et **la publication** de services dans un annuaire UDDI. La recherche de services est réalisée en se basant sur différentes méthodes de recherche : mots clés, identifiant du service, nom du service, etc. La publication de services est basée sur des formulaires de publication : la description d'un document tModel, la recommandation de service (permet à des clients de services non propriétaires de ces derniers de les recommander), etc.

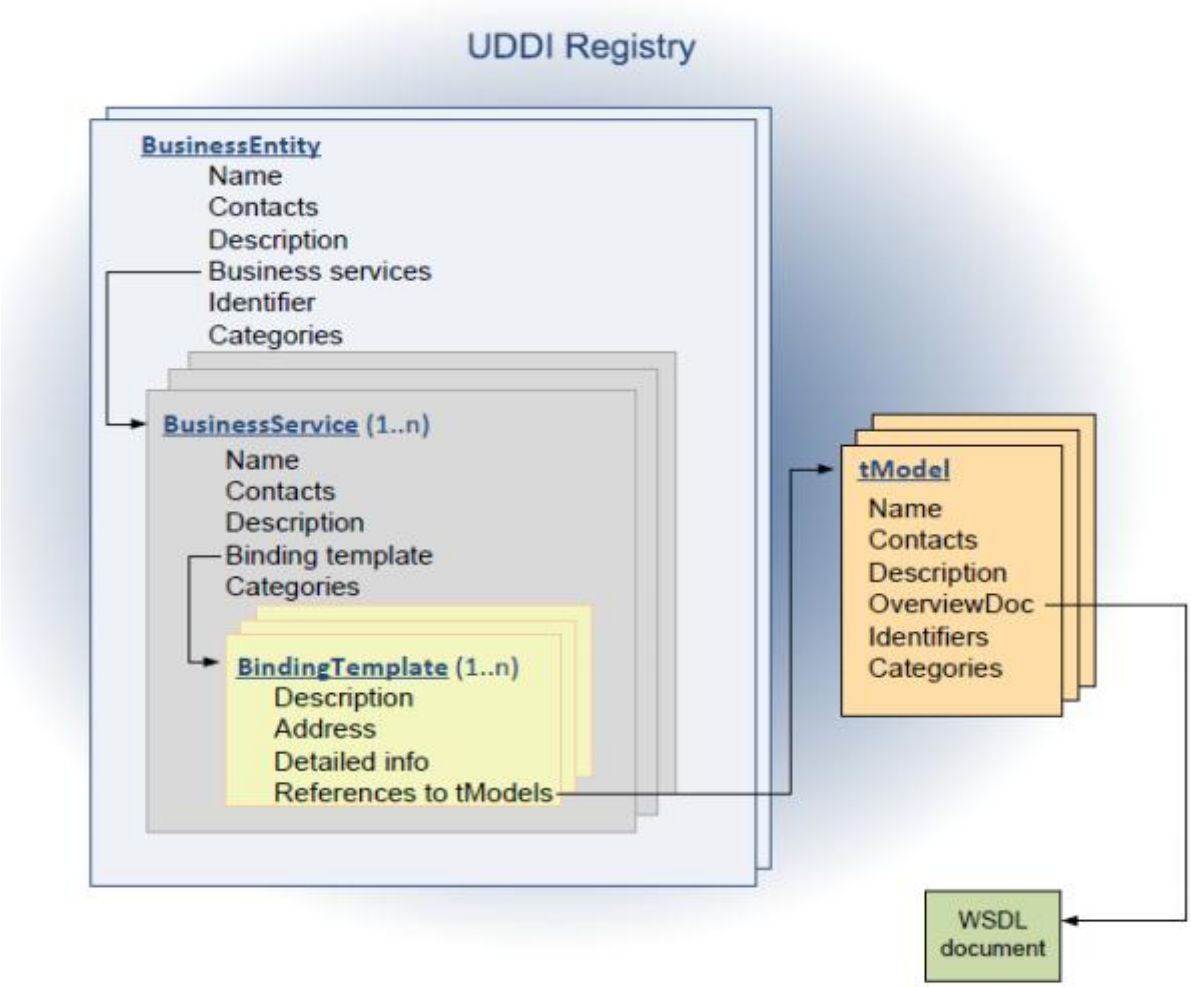

Figure [I.4] – Entités composant un annuaire UDDI [11].

Nous distinguons deux types d'annuaires UDDI (publiques et privés) :

❖ **L'annuaire UDDI publique :**

Est implémenté sous forme d'un réseau de nœuds UDDI, ces nœuds sont synchronisés, chacun d'eux est possédé par une entreprise donnée (telles que SAP, IBM et Microsoft).

La publication d'un service chez une entreprise Propage automatiquement ses informations (pages blanches, jaunes et vertes) aux différents nœuds UDDI. L'accès à l'ensemble des informations des registres peut se faire à n'importe quel nœud UDDI. Ce type d'annuaire est gratuit.

❖ **L'annuaire UDDI privé** :

Permet à une entreprise de choisir les partenaires pour lesquels elle autorise la publication et l'invocation de ses services web.

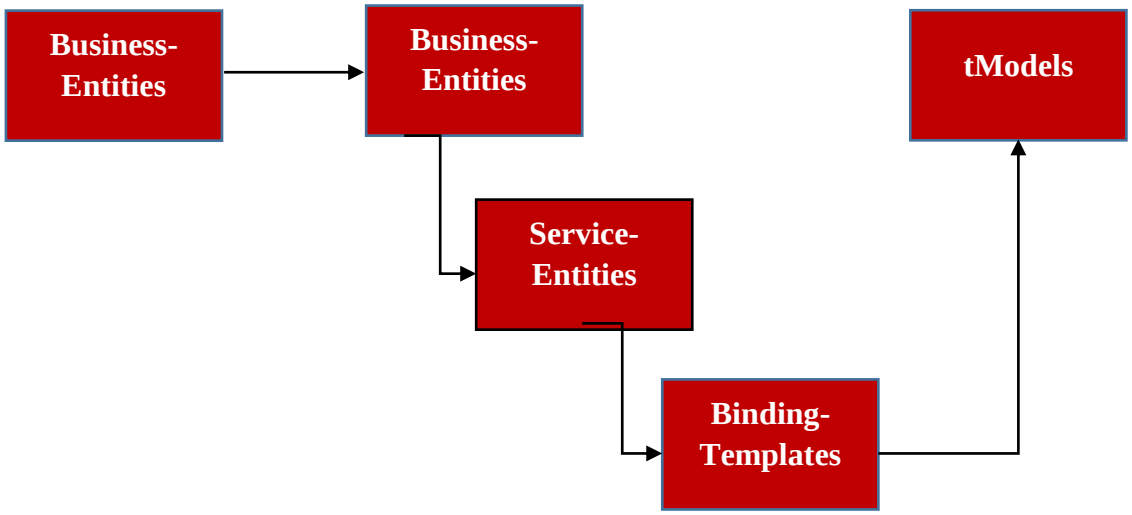

Figure [I.5] – Les structures de données d'UDDI

3.2.2. La couche description : est prise en charge par le Web Service Description Langage (WSDL) [14], qui décrit les fonctionnalités fournies par le service Web, les messages reçus et envoyés pour chaque fonctionnalité, ainsi que le protocole adopte pour la communication. Les types des données contenues dans les messages sont décrits à l'aide du langage XML Schéma [15].

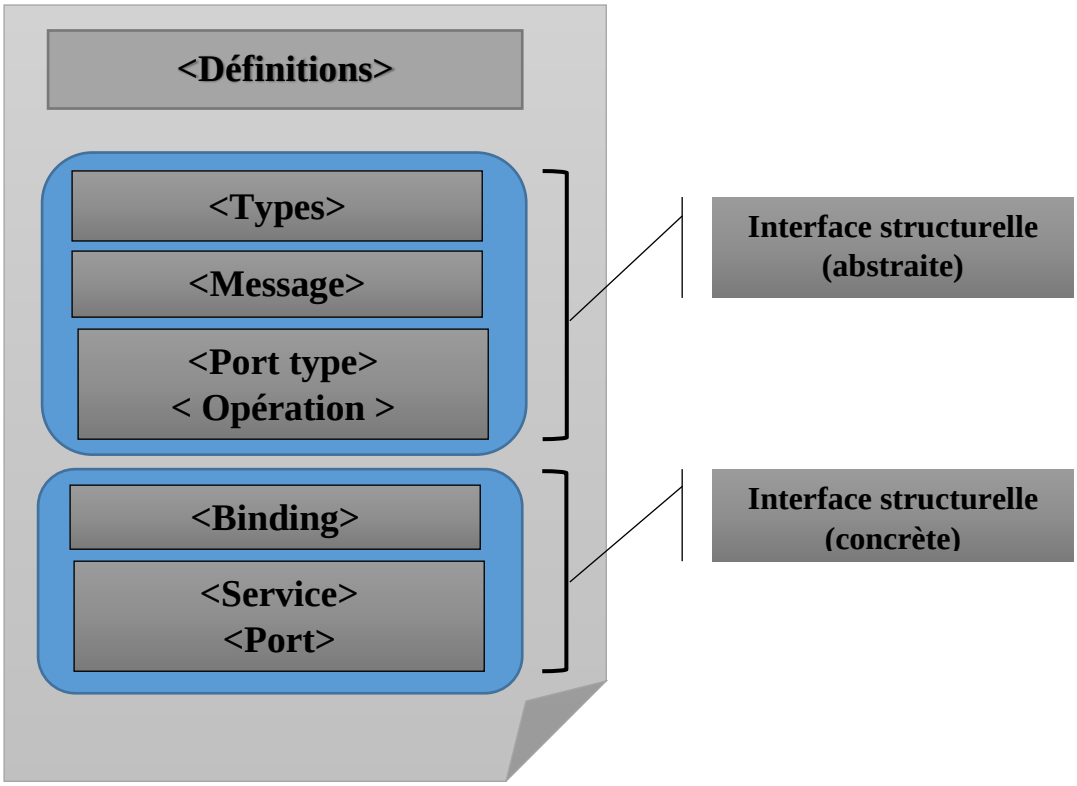

Figure [I.6] –: Structure d'une description WSDL

Nous notons que cette interface décrit juste la structure du service (la partie quoi) et non pas le comportement (la partie comment) [12]. Le document WSDL comporte deux parties indépendantes [Chinnici et al. 2007], [6] :

1) **La partie concrète** est constituée de la description des protocoles d'accès au Service Web (information particulière à un service). Ce niveau est seulement utilisé lors de l'invocation des méthodes du Service Web.

2) **La partie abstraite** regroupe les informations pouvant être réutilisées. Cette partie est utilisée principalement lors du processus de sélection du service.

☆ L'élément « **définitions** » représente la racine du document WSDL. Il définit le nom du service web, déclare les différents espaces de nommage utilisés dans le reste du document, et contient tous les éléments qui décrivent le service web.

- ☆ **Types** : fournit la définition de types de données utilisés pour décrire les messages échangés.
- ☆ **Messages** : représente une définition abstraire (noms et types) des données en cours de transmission.
- ☆ **PortTypes** : décrit un ensemble d'opérations. Chaque opération à zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs.
- ☆ **Binding** : spécifie une liaison entre un et un protocole concret (SOAP, HTTP...).
- ☆ **Service** : indique les adresses de port de chaque liaison.
- ☆ **Port** : représente un point d'accès de services défini par une adresse réseau et une liaison.
- ☆ **Opération** : c'est la description d'une action exposée dans le port.

L'intérêt d'avoir ces deux parties, est que la partie concrète propose une ou plusieurs réalisations de la partie abstraite. Les documents WSDL ne sont jamais générés par des développeurs, mais le sont grâce à des outils qui automatisent la tâche (par exemple, il existe des outils qui prennent une classe Java et qui créent le WSDL correspondant).

3.2.3. La couche message : utilise des protocoles reposant sur le langage XML, car sa syntaxe unique résout les conflits syntaxiques lors de l'encodage des données. Actuellement, Simple Object Access Protocol (SOAP) [16] et XML-Remote Procedure Call (XML-RPC) sont les deux protocoles utilises pour cette couche, SOAP étant le protocole prédominant.

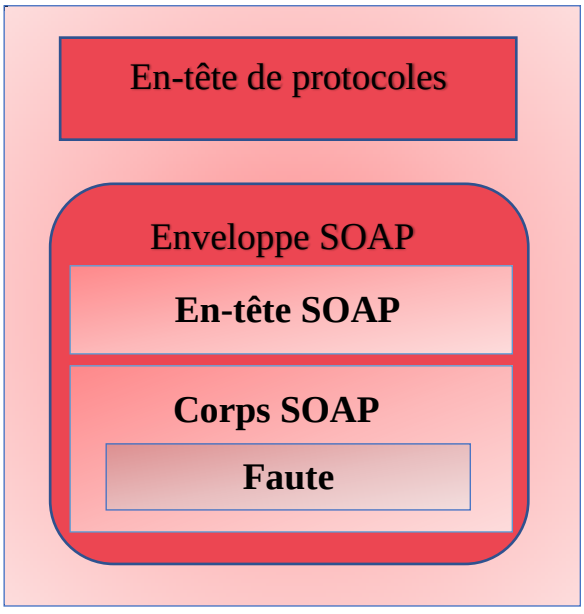

Figure [I.7] – Structure d'un message SOAP [16]

1. **L'en-tête du protocole de transport** : qui dépend de protocole de transport utilisé, par exemple si le protocole HTTP est utilisé, l'en-tête contient :
   - La version de protocole HTTP utilisée.
   - La date de génération de message SOAP.
   - Le type d'encodage du contenu (généralement de type XML).

2. **L'enveloppe SOAP** : la partie principale d'un message SOAP est l'enveloppe (symbolisée par la balise enveloppe), cette dernière est subdivisée en deux sous-parties : la partie en-tête (Header) et la partie corps du message (Body).

- **L'en-tête du message SOAP** : est optionnelle et extensible. Les balises XML qui permettent de symboliser cette partie sont : **<env:Header>** et **</env:Header>**. Ces balises peuvent être complétées par des attributs permettant de définir le domaine de noms du service Web. En fait, l'en-tête permet principalement d'ajouter des informations sur le comportement des différents nœuds intermédiaires, lors de traitement du message.

  Un nœud étant un intermédiaire SOAP, incluant le récepteur et l'émetteur SOAP, désignable depuis un message SOAP. Son rôle est de traiter l'en-tête, ensuite de transférer le résultat (le message SOAP modifié) à un autre intermédiaire (qui peut être le récepteur final).

  Les principaux attributs des éléments qui forment le bloc d'en-tête sont :
  env:role : permet d'indiquer à quel nœud la fonction décrite est destinée.
  env:mustUnderstand : c'est une valeur booléenne, elle permet de préciser que le traitement devient obligatoire pour un nœud intermédiaire, par exemple, pour un calcul très long, il peut être utile d'envoyer un e-mail à chaque étape.
  env:relay : cet attribut permet de relayer un message à un autre nœud si le premier nœud n'est pas capable de le traiter.

- **Le corps du message SOAP** : l'élément corps de message SOAP contient des données spécifiques à l'application. Les balises symbolisant cette partie sont : **<env:Body>** et **</env:Body>**. Les données doivent donc être sérialisées selon l'encodage XML. En plus,

des données de cette partie peuvent transporter un type spécial comme : les messages d'erreurs (SOAP Fault).

3.2.4. La couche transport : le HyperText Transfert Protocol (HTTP) [17] est devenu le standard de facto. Ce protocole omniprésent sur Internet est généralement tolère des pare-feu. Il est donc particulièrement adapte aux communications entre organisations. Cependant, d'autres protocoles peuvent être utilises, tels que le Simple Mail Transfer Protocol (SMTP) ou le File Transfer Protocol (FTP), permettant ainsi aux services Web de rester indépendants du mode de transport utilise.

L'atout principal des protocoles SOAP et UDDI ainsi que du langage WSDL est de reposer sur l'extensible Markup Language (XML) [18].

XML est un langage de description des données qui possède les avantages suivants :

– Une syntaxe unique qui permet son adoption par divers systèmes d'exploitation.

– Une structure arborescente qui facilite sa lisibilité par les machines et par les utilisateurs.

– Une grande extensibilité, car il n'impose aucune restriction d'utilisation en dehors de sa syntaxe.

– Séparation entre structure du document et présentation du document.

XML constitue la technologie de base des architectures services Web à cause de sa simplicité, sa portabilité sur différentes plates-formes, son auto-description ainsi que son extensibilité et flexibilité qui ont conduit à sa grande popularité et son adoption comme standard d'échange à travers le Web [18].

L'objectif initial est de structurer les données et faciliter l'échange automatisé de contenus complexes (arbres, texte riche...) entre systèmes d'informations hétérogènes (interopérabilité).Aussi, grâce à la structuration, XML permet la distinction entre les données des applications et les données des protocoles.

3.3. Structure d'un document XML

Un document XML est structuré en trois parties [19] :

1. **Un prologue** correspond à la première ligne de votre document XML. Il donne des informations de traitement ; il permet d'indiquer la version de la norme XML utilisée pour

créer le document, ainsi que l'encodage de caractères utilisé dans le document et les informations sur un éventuel schéma (à l'aide d'un fichier DTD Document Type Définition).

2. **Le corps :** constitué de l'ensemble des balises qui décrivent les données. Il y a cependant une règle très importante à respecter dans la constitution du corps : une balise en paires unique doit contenir toutes les autres. Cette balise est appelée élément racine du corps.
3. **Des commentaires**.

```
<?xml version="1.0" encoding="UTF-8"?>
<bibliotheque>
  <livre1>
                <liv>
                        <langue> Français </langue>
                        <reference> 125864 </reference>

                </liv>
                <titre>
                        <nom> la-cryptographie-asymetrique </nom>
                        <type> informatique </type>
                        <genre> biblio </genre>

                </titre>
                <author>
                        <nom> MIMOUNE </nom>
                        <prenom> RADOUANE </prenom>


                </author>
  </livre1>
 </bibliotheque>
```

Figure [I.8] – Exemple d'un document XML.

3.4. XML schéma

Un schéma XML définit d'une part la structure d'un document pour qu'un document soit valide. Il est aussi possible de restreindre les données acceptées dans un document XML et faire de l'imbrication entre les éléments, ce qui s'apparente aux DTD, et d'autre part, le type des éléments et de leurs attributs. L'information fournie par le schéma est donc plus riche que celle dans le DTD car XML schéma propose des nouveautés en plus que DTD comme [20] :

- Support de nombreux types de données de contenu de l'attribut.

- Possibilité de définir ses propres types de données.
- Le support des espaces de nommage.
- Les indicateurs d'occurrences des éléments peuvent être tout nombre non négatif (dans une DTD, on était limité à 0, 1 ou un nombre infini d'occurrences pour un élément).
- Identification de l'ordre et de la relation entre les éléments.
- Les schémas sont très facilement concevables par modules.
- L'ajout de contraintes d'intégrité sur les données.
- La séparation entre le modèle de validation et l'instance XML.

Il existe une panoplie d'outils permettant l'exploitation des documents XML. Parmi celles-ci nous pouvons citer [18] :

XPath : Pour définir la manière d'adresser les parties de chaque élément d'un document XML.

XQuery : S'intéresse à la structure logique abstraite d'un document XML.

XSL : Pour définir la présentation de document XML.

Ainsi, l'utilisation exclusive de langages et protocoles reposant sur le langage XML, ainsi que des normes actuelles d'Internet comme le protocole HTTP, favorise l'interopérabilité entre systèmes d'information distribue. Les services Web restent indépendants des systèmes d'exploitation et des plateformes de développement, ce qui facilite les interactions avec les applications clientes, mais aussi entre plusieurs services Web participant à une composition.

Des exemples de Services Web actuellement disponibles concernent les prévisions météorologiques[1], la réservation de voyage en ligne[2], les services bancaires ou des fonctions entières d'une entreprise comme la mise en œuvre de la gestion de la chaîne logistique[3].

4. Architecture des services Web

L'architecture des services Web est une instance de SOA. L'interopérabilité qu'impliquent les services Web doit être soutenue par une architecture stable et fiable. Deux types d'architecture existent pour les services Web : La première dite architecture de *référence*, elle contient trois couches principales. La seconde architecture est plus complète, elle utilise les couches standards de la première architecture en ajoutant au-dessus d'autres couches plus spécifiques. Elle est appelé architecture *étendue* ou encore en Pile.

4.1. Architecture de référence

L'architecture classique de « référence » des Services Web se fonde sur le modèle SOA (Service Oriented Architecture) qui fait intervenir trois acteurs : un client, un fournisseur ainsi qu'un intermédiaire jouant le rôle d'annuaire (**Figure [I.9]**).

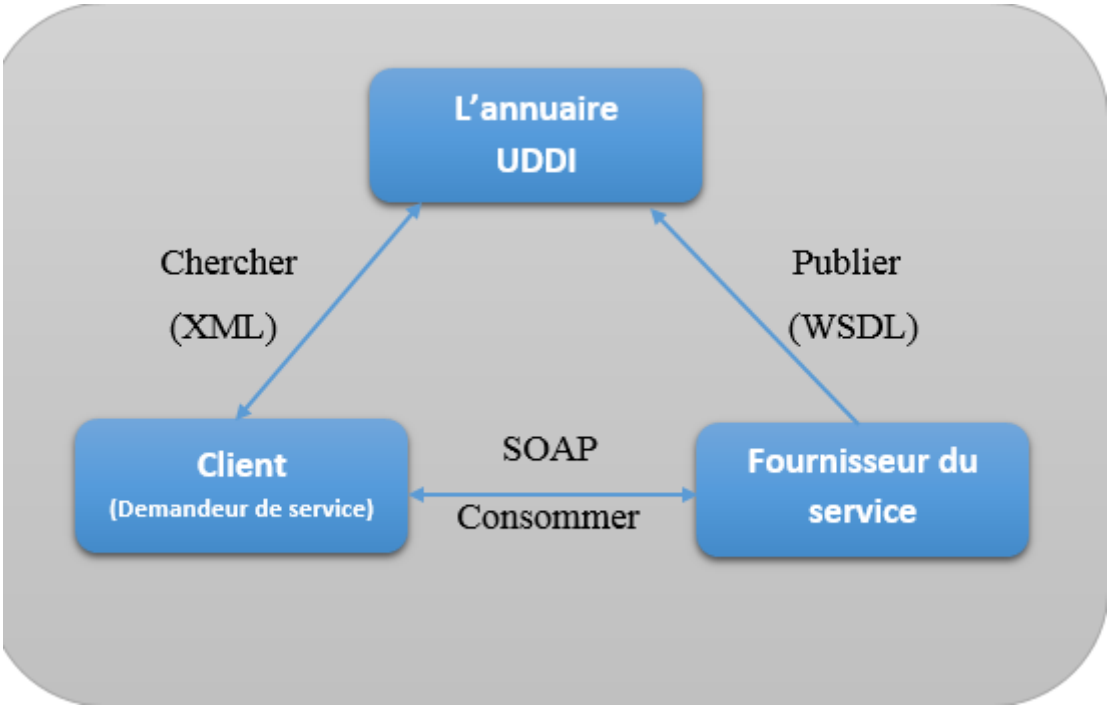

Figure [I.9] – Modèle fonctionnel de l'architecture SOA.

4.1.1. Les acteurs :

4.1.1.1. **Le Fournisseur du service :** correspond au propriétaire d'un service.

Il correspond à l'entité qui réalisera effectivement le service demandé. Il publie son service en fournissant la description des services (les propriétés fonctionnel et non-fonctionnel) au format WSDL dans l'annuaire de services afin que les clients puissent découvrir et accéder au service. Alors, il représente l'environnement d'hébergement et d'exécution du service. Il est constitué de trois couches de bases [21] :

- **La couche de données** : contient les différentes bases de données utilisées par le service.
- **La couche applicative** : c'est la plateforme de développement qui assure l'exécution du service web.
- **La couche de description** : elle expose les fonctionnalités du service via un fichier WSDL.

4.1.2. 2 L'annuaire du service : correspond à un registre de description de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de service à l'intention des clients. Il offre aux fournisseurs la capacité de publier leurs services et aux clients le moyen de localiser les services répondant à leurs besoins, l'annuaire joue le rôle d'intermédiaire entre les clients et les fournisseurs de services. Comme, il assure le regroupement, le stockage et la diffusion des descriptions des services web [10].

4.1.3. 3 Le client : Il représente n'importe quel consommateur du service Web. D'un point de vue technique, le demandeur de service est constitué de l'application qui va rechercher et invoquer un service en envoyant une demande en XML (REST, XML-RPC, SOAP). Le client peut être une simple application Web, comme il peut être un autre service Web [10].

4.1.2. Les opérations principales :

Le fonctionnement des Services Web repose sur un modèle en phases, dont les trois phases fondamentales sont les suivantes : la publication d'un service, la découverte d'un service et l'interaction avec un service. Ce fonctionnement est normalisé à travers un certain nombre de standards : un protocole abstrait de description et de structuration des messages SOAP, une spécification XML qui permet la publication et la localisation des services dans les annuaires UDDI (Universal Discovery, Description and Integration) et un format de description des Services Web publiés dans les annuaires, WSDL (Web Services Description Language).

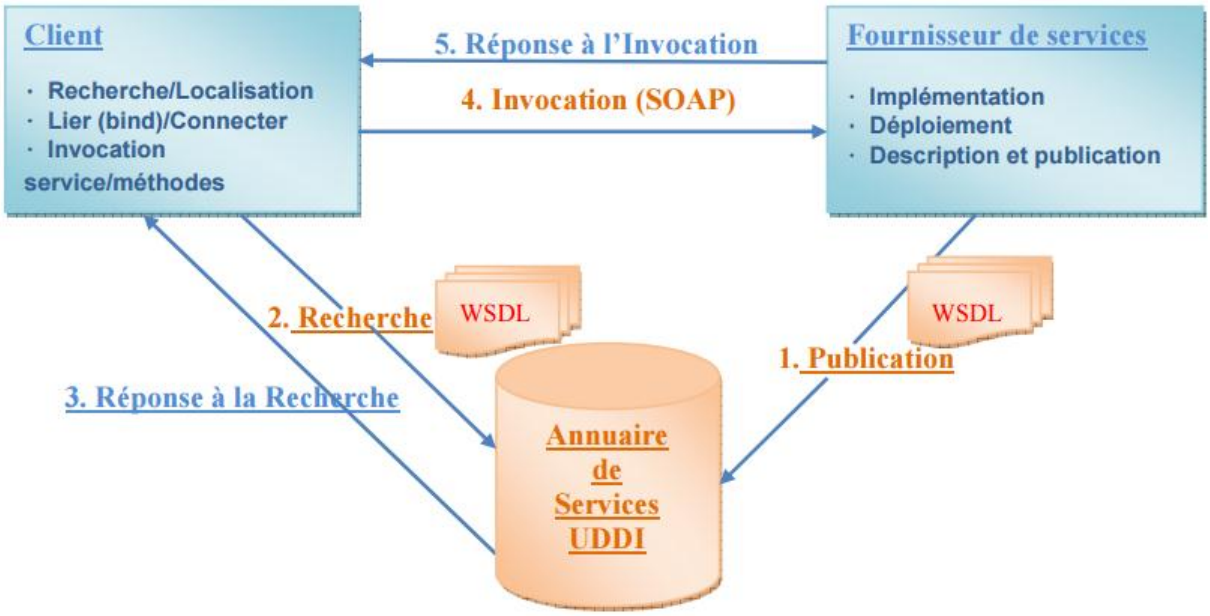

Figure [I.10] – Principe de Fonctionnement des Services Web [18].

Tous ces couches et standards sont détaillés dans les sections suivantes :

- **Publication** (Publier) : Une fois que le fournisseur a construit son Service Web, il a besoin de publier ce service afin que les autres (personnes, machines) soient capables de le découvrir puis de l'utiliser. Un annuaire UDDI est utilisé pour publier les Services Web sur un dépôt central. Afin d'être publié dans un annuaire UDDI, le Web Service a besoin d'être décrit. La description de ce service est effectuée dans le langage WSDL. Cette description indique les fonctions de ce service, ainsi que les paramètres d'entrée/sortie et le protocole de transport.
- **Découverte** (rechercher) : Une fois que les Services Web sont publiés dans un annuaire par les fournisseurs, l'utilisateur peut chercher des Services Web en lançant une requête auprès de l'annuaire UDDI pour recevoir la liste des Services Web pertinents par rapport à sa requête.
- **Réponse à la recherche :** Le résultat de la recherche dans l'annuaire est un fichier WSDL qui sera transmis, dans un message SOAP, au client demandeur. Ce fichier contient la description du service et les informations techniques nécessaires pour son invocation [18].
- **Invocation** (consommer) : Une fois que l'utilisateur a découvert le Service Web via l'interface UDDI, et qu'il a pris la décision d'utiliser ce service dans son application, il a besoin d'invoquer le Service Web.
- **Réponse à l'invocation** : Le fournisseur réagit à la réception du message d'invocation de la part du client en envoyant au client la réponse résultant de l'exécution de la procédure. Cette réponse est transmise sous la forme d'un document XML via SOAP [18].

4.2. Architecture étendue

Cette architecture est aussi appelée « pile des services Web», du fait qu'elle est constituée de plusieurs couches se superposant les unes aux autres où chaque couche s'appuyant sur un standard particulier et répond à des préoccupations fonctionnelles différentes telles que la sécurité, la description, la messagerie fiable, le transport et les transactions. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment. Ces couches rendent viable l'utilisation effective des services dans le monde industriel.

Nous apportons une explication des différents types de couches de l'architecture qui sont représentées dans la **Figure [I.11]** [22] :

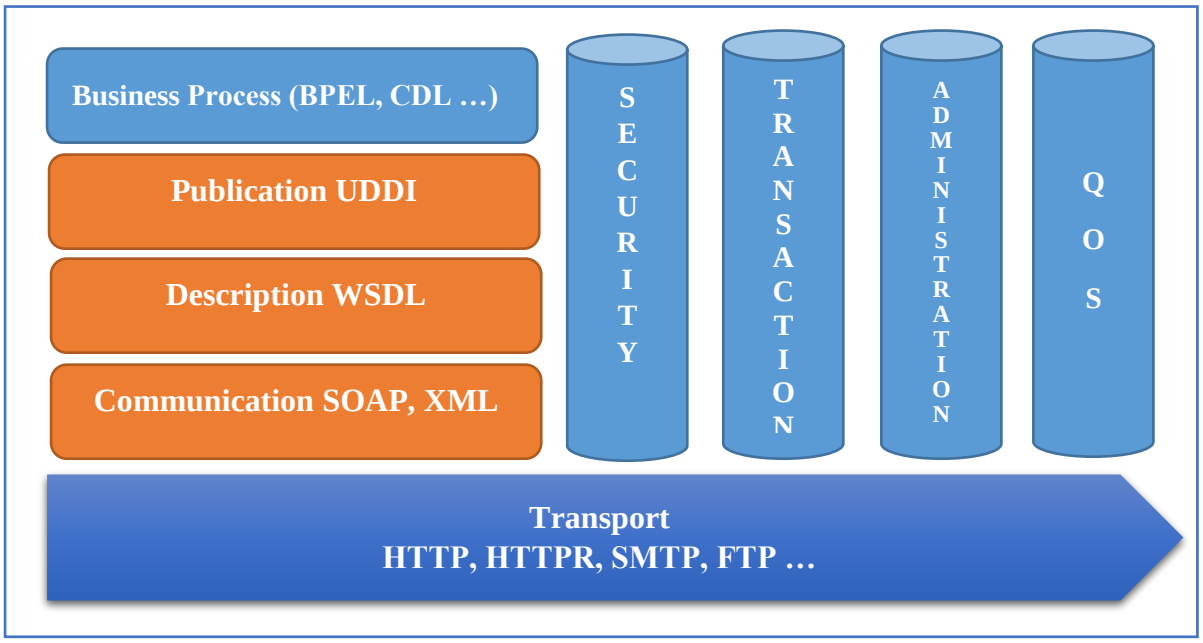

Figure [I.11] – Architecture en Pile des services web.

- *L'infrastructure de base (Publication, Description, Communication)* : Définit les composants de l'architecture de base (section : architecture de référence).
- *Le transport :* Est le mécanisme utilisé pour déplacer les requêtes de service du consommateur de services au fournisseur, et les réponses des services du fournisseur au consommateur de service pour pouvoir se découvrir et dialoguer entre eux. Le protocole le plus utilisé dans cette couche est HTTP. Cependant, d'autres protocoles peuvent être utilisés, tels que le SMTP, FTP . . . permettant ainsi aux services Web de rester indépendants du mode de transport utilisé.
- *Le protocole de communication :* Est un mécanisme utilisé par le fournisseur et le client de service pour communiquer le contenu des requêtes et des réponses. Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Cette couche utilise des protocoles reposants sur le langage XML, car sa syntaxe unique résout les conflits syntaxiques lors de l'encodage des données. Actuellement, SOAP est le protocole le plus utilisé pour cette couche.
- *Couches transversales :* (Sécurité, Transactions, Administration, QoS) :

Ce sont les aspects concernant la qualité de services : [22]

- **La sécurité :** La gestion de la sécurité est actuellement le frein le plus important à la mise en place d'architectures distribuées à base de services Web. Elle représente la normalisation des moyens permettent de couvrir les problématiques d'authentification

et de gestion des droits d'accès. En plus, l'ensemble de règles qui peuvent être appliqués lors de d'authentification, de l'autorisation, et du contrôle d'accès des consommateurs qui invoquent les services. Les consommateurs peuvent aussi demander l'utilisation de règles de sécurité. Par exemple, dans le cas où il veut garantir l'intégrité des données transmises.

- **Transaction :** Normalisation des moyens permettant de garantir l'intégrité des transactions longues impliquant plusieurs services Web. Un environnement d'intégration de services peut contenir un élément chargé de garantir la cohérence des données utilisées et des résultats retournés par une collaboration (ou composition) de services.
- **Management :** Est le processus chargé de surveiller et de contrôler de manière proactive le comportement d'un service ou d'un ensemble de services.

➢ *La couche Business Processus (BusinessProcess) :*

Cette couche supérieure permet l'intégration de services Web, elle établit la représentation d'un « Business Process » comme un ensemble de service Web. De plus, la description de l'utilisation de différents services composant ce service est disponible par l'intermédiaire de cette couche.

5. Les avantages et inconvénients des services Web

5.1. Avantages :

L'idée essentielle derrière les services Web est de partager les applications et des programmes en un ensemble d'éléments réutilisables appelés service, de sorte que, chacun de ces éléments effectuent une tâche principale et efficace, afin de faciliter l'interopérabilité entre tous ces services Web [23,24].

- Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.
- Permettent de profiter de différents environnements et langages de développement par une publication, localisation, description et une invocation via XML.

[1] http://www.meteofrance.com/previsions-meteo-france/metropole consulté le 10/02/2017.

[2] https://www.expedia.fr/Vols consulté le 10/02/2017.

[3] https://particuliers.secure.lcl.fr/index.html consulté le 10/02/2017.

- Les services Web sont très flexibles, indépendants des langages de programmation et des systèmes d'exploitation.
- Les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.
- Basés sur le protocole HTTP, les services Web peuvent fonctionner au travers de nombreux pare-feu sans nécessiter des changements sur les règles de filtrage.
- Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants.

5.2. Inconvénients

Les inconvénients des services web sont [25] :

- La sémantique n'est pas prise en charge de façon efficace car le WSDL décrit les services de manière syntaxique.
- Les services Web ont de faibles performances par rapport à d'autres approches de l'informatique répartie telles que le RMI, CORBA, ou DCOM.
- En l'utilisation du protocole HTTP, les services Web peuvent contourner les mesures de sécurité mises en place à travers les firewalls.
- Ils ne sont pas sécurisés à100 %.

6. Quelques domaines d'application de services Web

Les services Web peuvent être utiles dans la plupart des scénarios applicatifs lorsque la communication peut être établit sur un modèle bidirectionnel (requête/réponse).

☆ L'application des services Web est multiple, autant dans les domaines du B2B, B2C que pour les domaines de gestion de stock, etc.

☆ B2C (Business to Consumer) : qualifie une application, un site internet destiné au grand public.

☆ B2B (Business to Business) : qualifie une application, un site internet destiné au commerce professionnel à professionnel [7].

7. Conclusion

Ce chapitre couvre le développement des applications orientées services où nous avons présenté les principes de l'architecture SOA, par la suite nous avons introduit les services Web comme étant des technologies standards qui implémentent SOA en permettant d'interagir, de gérer et d'automatiser plus facilement et plus rapidement les processus métier intra-entreprise et inter-entreprises en échangeant des informations au format XML. Ensuite nous avons introduit quelques protocoles et fonctionnalité de services qui permettent l'intégration et la réutilisation des services dans diverse applications.

Dans le prochain chapitre nous allons présenter les approches de composition des services web qui se basent sur des critères de qualité de service (fonctionnels et non fonctionnels) et aussi les différentes méthodes multicritères qui aident à la décision (MCDM). Puis, nous détaillerons un algorithme de composition multi critères que nous allons utiliser dans notre proposition.

Chapitre 02 :

Composition des services web

1. Introduction

Dans nos jours, l'utilisation de services web a connu une popularité grandissante. Ces services sont très utilisés notamment par les entreprises pour rendre accessible leurs métiers ou leurs données via le Web. Les fonctionnalités des services web sont limitées à des tâches simples qui présentent des collections d'opérations. L'exigence de remplir de nouveaux besoins a obligé les concepteurs à composer des nouveaux services en se basant sur des autres qui existent déjà afin de répondre à des exigences plus complexes. La composition de services web est considérée comme l'une des motivations les plus importantes de la technologie service web. Elle détient un potentiel énorme dans la réorganisation et l'intégration des applications d'entreprise.

L'objectif de ce chapitre est de présenter la composition des services web, ses différentes approches et types et les langages de composition utilisés.

2. Scénario de la composition

Bien que les services Web sont des briques de bases de l'architecture orientée service SOA et qui s'affranchissant de toute contrainte de compatibilité logicielle ou matérielle, peuvent ne pas être très utiles sans faire appel à une combinaison avec d'autres services.

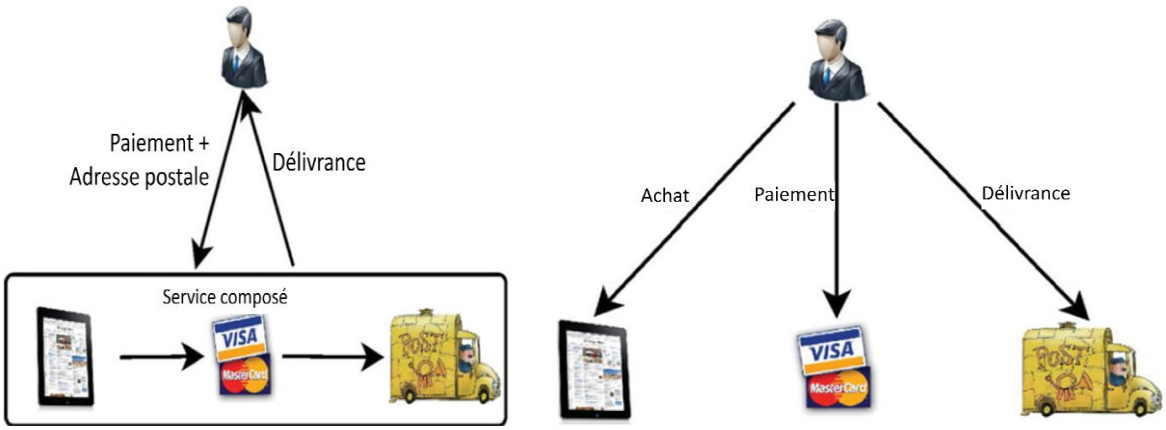

Figure [II.1] – Deux scénarios possibles pour un achat en ligne en utilisant les services web. [26]

Dans le scénario illustré dans la **Figure [II.1]**, un client a besoin de faire plusieurs appels à des services Web élémentaires pour compléter un achat en ligne. En outre, les appels doivent être faits dans un ordre spécifique dans lequel une ou plusieurs références de l'appel précédent sont utilisées dans l'invocation suivante. En plus d'être un processus lourd

ce scénario augmente le risque d'erreurs. D'où l'appel à une ***composition des services Web*** est indispensable.

3. La Composition des services web

3.1. Définition

La composition des services Web fait référence au processus qui consiste à combiner les fonctionnalités de plusieurs services, simples ou eux-mêmes composés au sein d'un même processus métier. Dans le but de répondre à des demandes complexes qu'un seul service ne pourrait satisfaire [27].

3.2. Cycle de vie d'une composition des services Web

La composition au sens large englobe plusieurs activités qui correspondent aux différentes phases de son cycle de vie. D'une façon générale on peut dire que le cycle de vie de la composition des services web inclue quatre phases :

- **La phase de définition :** Durant cette phase le demandeur de service précise la spécification de la composition, qui devrait fournir suffisamment d'informations sur les besoins des utilisateurs et les priorités pour le service composé [27]. Ensuite la spécification est décomposée en un modèle abstrait qui englobe un ensemble des activités, le contrôle et le flux des données entre ces activités les comportements exceptionnels.
- **La phase de sélection :** cette phase consiste à découvrir pour chaque activité dans le service composé les services web appropriés qui correspondent aux besoins définies, cette découverte est fondée sur les informations contenues dans les descriptions des services web publiés. Il est possible que plus d'un service réponde aux besoins. Par conséquent, le meilleur service doit être sélectionné. Après que tous les services Web requis sont identifiés et liés aux activités correspondantes, le service composé est produit.
- **La phase de déploiement :** Dans cette phase, le service composé est déployé pour permettre son instanciation et l'invocation par les utilisateurs finaux. Le résultat de cette phase est le service composé exécutable.
- **La phase d'exécution :** Dans cette phase, l'instance de service composé est créée et exécutée par un mécanisme d'exécution, qui est également chargé d'invoquer les services participants. Pendant cette étape d'exécution les tâches de contrôle et de

surveillance de la composition, mesure de la performance et la gestion des exceptions, doivent être assurées. [27]

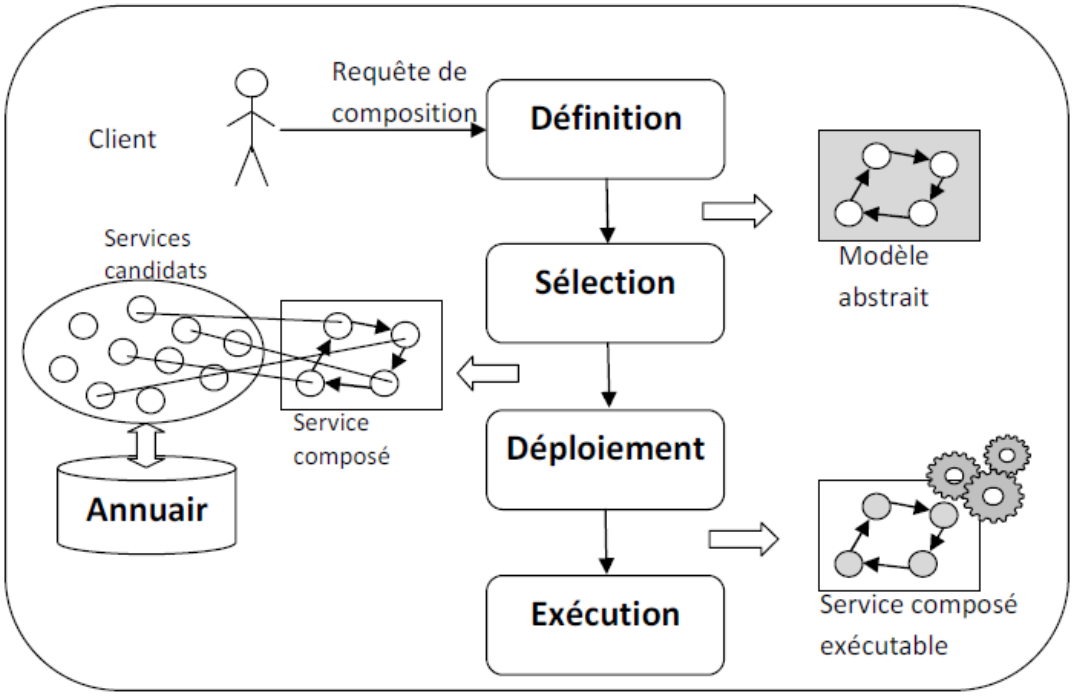

Figure [II.2] – Cycle de vie de composition des services web.[27]

3.3. Défis de composition des services Web

Le problème de la composition des services Web est en plein essor. Nous soulignons ici quelques sources de ses complexités [28] :

- la complexité liée à la conception de services Web : un Web service n'est pas adaptable. Il est passif jusqu'à ce qu'il soit invoqué. Il a des connaissances de lui-même mais pas de ses applications ou de ses utilisateurs clients
- le nombre des Web services disponibles est très grand, il est déjà au-delà des capacités humaines pour les analyser manuellement.
- puisque le nombre de Web services augmente jour âpres jour, il devient plus difficile de trouver artificiellement le Web service qui peut effectuer la tâche à accomplir et encore plus difficile de composer un ensemble de services avec des approches classiques.

➢ les Web services peuvent être crées et réactualises à la volée. Par conséquent, le système de composition doit détecter la mise `à jour lors de l'exécution. Ainsi, le schéma de la composition devrait s'adapter en fonction des nouvelles informations.

- les Web services sont généralement établis par des organisations différentes qui utilisent différents modèles conceptuels pour la présentation des caractéristiques des services. Cela exige l'utilisation d'informations pertinentes pour faire correspondre (matching) et composer les services Web.

4. Classification des approches de composition des services Web

Plusieurs approches de composition des services Web ont été développées par des chercheurs, ces approches sont aussi classées en plusieurs catégories selon différentes critères. Le plus souvent les approches de composition des services Web sont classées selon la nature et le moment de processus de sélection et de liaison des services composants en statiques et dynamiques, et selon le degré de participation de l'utilisateur dans la définition de schéma de composition en approches manuelles, semi automatiques ou automatiques .

4.1. Composition manuelle, semi-automatique et automatique

- *La composition manuelle de services Web :* signifie que le processus décomposition est réalisé totalement par un utilisateur qui a accès aux services, il les sélectionne, les ordonne, et les exécute en les surveillant pendant l'exécution. La composition manuelle reste à portée limitée car elle requiert un niveau technique très élevé [29] (généralement par un expert), et un temps de conception important sans garantie que le résultat de l'exécution satisfait en effet les besoins de l'utilisateur.
- *La composition automatique de services Web :* prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise. Dans ce type de composition l'utilisateur final ou le développeur de l'application spécifie un objectif (exprimé dans un langage de description ou notation mathématique) et un moteur de composition "intelligent" sélectionne des services adéquats et propose la composition en toute transparence pour l'utilisateur. Les principaux défis de la composition automatique sont dans la façon d'identifier les services candidats, de les composer, et de vérifier dans quelle mesure ils correspondent à la requête de l'utilisateur.
- *La composition semi-automatique de services Web :* a pour but de fournir à l'utilisateur final un environnement de création de services composés. Cet environnement offre un support pour l'utilisateur par l'automatisation de certaines parties de la composition. La composition semi-automatique étant à mi-chemin entre les deux autres a l'avantage principal de faire participer l'utilisateur au processus de composition en exploitant, les informations générées par l'utilisateur ou de sa

communauté. Dans la composition semi-automatique l'utilisateur maintiendra certain contrôle sur le processus de composition mais il n'aura pas besoin de connaissances de programmation.

4.2. Composition statique et composition dynamique

- *Composition statique de services Web :* dans ce type de composition la construction d'un modèle abstrait des tâches qui doit être réalisé au cours de l'exécution de la composition se fait à la conception et/ou la compilation avant que la planification de la composition commence. Ce modèle abstrait est rien qu'une représentation d'un ensemble des tâches et des dépendances entre eux, ou chaque tâche correspond à un service Web [30]. Les services Web participants à la composition sont choisis, liés ensemble, compilés puis déployés.
- *Orchestration et chorégraphie :* Deux approches principales sont actuellement étudiées pour la composition statique de services Web. La première approche, appelée services Web *orchestration*, combine les services participants à la composition par l'ajout d'un coordonnateur central (l'orchestrateur) qui est responsable de l'invocation et la combinaison des sous-activités simples. Elle offre une vision centralisée, le procédé est toujours contrôlé du point de vue de l'un des partenaires. La seconde approche, appelée services Web *chorégraphie*, ne suppose pas l'exploitation d'un coordinateur central mais définit plutôt des tâches complexes via la définition de la conversation qui devrait être entrepris par chaque participant. Selon cette approche, l'activité globale est obtenue comme la composition des interactions peer-to –peer entre les services participants. .Elle est de nature plus collaborative. Qu'il s'agisse d'orchestration ou de chorégraphie, les procédés et services Web sont trop fortement couplés, l'agrégation obtenue est rigide et donc difficilement modifiable, et les langages sont de trop bas niveau, c'est à- dire trop proches des langages de programmation classiques.
- *Composition dynamique de services Web :* 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 au moment de l'exécution en fonction des fournisseurs de services Web disponibles.

Une autre classe dite semi-dynamique est introduite dans [31] comme le résultat de chevauchement des deux classifications précédentes comme illustré dans Figure [II.3].

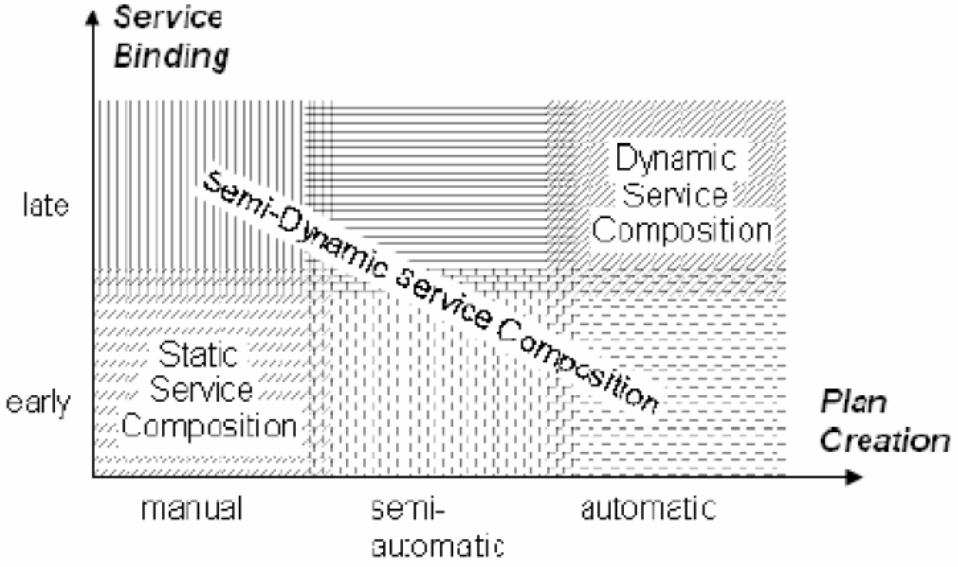

Figure [II.3] – classification générale des approches de composition des services web. [31]

Il y a plusieurs avantages de la composition dynamique des services. Contrairement à la composition statique, où le nombre de services fournis aux utilisateurs finaux est limité et les services sont spécifiés au moment de la conception, la composition dynamique peut servir des applications ou des utilisateurs à la demande. Avec la composition dynamique un nombre illimité de nouveaux services peut être créé à partir d'un ensemble limité des services composants.

La composition dynamique est l'idéal dans un environnement de services Web dont la nature est très dynamique. Elle est flexible et adaptable dans le cas où il y aura fréquemment des changements au cours de l'exécution. Contrairement la composition statique qui fonctionne bien dans les environnements fermés (n'évoluent pas souvent) où les fonctionnalités de service ou la spécification de la composition ne change pas ou rarement.

Dans le tableau ci-dessus **Tab II.1** on donne les principales différences entre la composition statique et dynamique de services Web.

Critère	statique	dynamique
Génération de modèle de processus	A la conception /compilation	A l'exécution
Découverte /sélection des services	A la conception/compilation	A l'exécution
Liaison des services	A la conception (i e : chaque instanciation du service composé sera des mêmes services composants)	A l'exécution
Conception de la composition	A la conception	A l'exécution
Personnalisation a l'exécution	impossible	possible
Possibilité d'extension a l'exécution	impossible	possible
Nombre de services	Limité	Illimité
Cout	Constant	Varie selon les services sélectionnés
Fiabilité et tolérance aux fautes	Faible	Elevée (spécifiquement dans le cas où un service devient indisponible après un certain temps qui peut être substitué par l'invocation du service fonctionnellement équivalent)
Adaptabilité aux changements d'environnements	Faible	Elevée

TAB [II.1] – Comparaison entre la composition statique et dynamique.[32]

5. Autres approches de composition des services Web

La plupart des travaux de recherche dans le domaine de composition des services Web se concentrent dans un coté sur l'automatisation de certaines parties ou la totalité de processus de la composition (semi-automatique/automatique), et dans un autre coté sur la dynamicité de processus de la composition. De nombreuses approches de composition des services Web existent, ces approches peuvent être regroupées en plusieurs axes et courants.

5.1. Approches sémantiques

Les approches sémantique sont le résultat de l'utilisation de la sémantique dans la description des services Web qui est plus compréhensible par les machine (ou même par les services Web) et fournit la base d'une interopérabilité transparente entre les différents services. Dans une description sémantique les paramètres d'entrée et de sortie d'un service

Web sont des concepts référents à des ontologies, les préconditions et les effets sont exprimés par des assertions logiques sur les paramètres d'entrée et de sortie. La correspondance (en anglais *matching*) vise à trouver une similarité sémantique entre un paramètre de sortie et un paramètre d'entrée de deux services. Les liens causaux stockent cette similarité entre les paramètres de services Web.

Dans la découverte des services Web, le matching est effectué sur les mêmes catégories de paramètres, c'est-à-dire sur les paires d'entrée, ou les paires de sortie. A l'opposé pour la composition des services Web, le problème est adressé par l'étude de matching sur des paires distinctes de paramètres d'entrée et de sortie des différents services Web.

Les descriptions sémantiques fournissent des agents logiciels avec un moyen de raisonnement automatique sur la sémantique du service. Dans un environnement de services sémantiquement annotés, les utilisateurs pourraient être aidés par des agents logiciels qui s'identifient automatiquement et si nécessaire, composent dynamiquement des services afin d'atteindre les objectifs de l'utilisateur. Cela peut être soit explicitement déclarés ou issus de la situation de l'utilisateur actuellement en jeu. [33]

5.2. Approches orientées intelligence artificielle

Plusieurs recherches actuelles dans le domaine de la composition s'articulent sur des techniques d'intelligence artificielle est plus particulièrement sur la planification (AI planning) :

5.2.1. La planification

La planification est un des domaines de l'Intelligence Artificielle qui permet de choisir et d'organiser des actions en produisant un plan. L'exécution de plan modifie les propriétés du système en le faisant évoluer de l'état initial jusqu'au but désiré.

Le problème de composition des services Web peut être modélisé comme un problème de planification où les services sont vus comme des actions et la composition comme un plan. Un plan solution c'est le schéma d'exécution d'un ensemble de services (actions) à partir de l'état initial jusqu'à atteindre l'état but.

Dans la suite nous présentons des méthodes de composition qui se basent sur la planification.

- **La planification classique :** Les approches de la composition des services Web à base de la planification classique traduit généralement des modèles de processus OWL-S en des représentations internes tels que PDDL (Planning Domain Definition Language).[34] PDDL décrit les similarités entre l'approche de description et de composition OWL-S et le PDDL. Il présente une approche pour traduire les descriptions OWL-S en PDDL afin de modéliser la

composition comme un problème de satisfaction d'un "planning". Il propose la notion de « value of an action » qui modélise l'exécution d'un service. Elle exprime des changements qui peuvent avoir lieu dans l'environnement.

- **Planification hiérarchique :** Dans [35], [Wu et al] recommandent l'utilisation du planificateur SHOP2 pour la composition automatique de services Web à partir de leur description sémantique. SHOP2 est un planificateur HTN (Hierachical Task Network). Les auteurs pensent que le concept de décomposition de tâches dans le planificateur HTN est très similaire au concept de décomposition de processus composés, décrits dans l'ontologie de processus proposée par DAML-S (remplacé par OWL-S). Afin de rendre leur proposition réalisable, les auteurs font deux hypothèses concernant le modèle de processus :
  - Les processus atomiques ont, soit des sorties, soit des effets, mais pas les deux en même temps.
  - Les structures de contrôle Split et Split+Join ne sont pas utilisées dans les processus composites [Sirin et Parsia] dans [36] sont allés plus loin en intégrant un raisonneur sur les langages de descriptions dans le planificateur SHOP2. La planification HTN est très puissante pour les domaines où la connaissance est complète et détaillée.
- **Calcul situationnel (situation calculus)** : les auteurs de [37] proposent d'adapter et d'étendre le langage Golog pour la construction automatique de services Web. Golog est un langage de programmation logique de haut niveau pour la spécification et l'exécution des actions complexes dans les domaines dynamiques qui permet de faire du calcul situationnel (i.e. langage logique qui sert à représenter des changements ou évolutions en termes de situations, d'actions et d'objets). Le problème de la composition des services Web est abordé de la façon suivante : la requête de l'utilisateur et les contraintes des services sont représentées en termes de prédicats du premier ordre dans le langage de calcul situationnel. Les services sont transformés en actions (primitives ou complexes) dans le même langage. Puis, à l'aide de règles de déduction et des contraintes, des modèles sont ainsi générés et sont instanciés à l'exécution à partir des préférences utilisateur, et l'ensemble de services atomiques sont reliés dans un langage de programmation procédural (if-then-else, while …etc).
- **Planification basée sur les règles (Rule-based planning)** :[Medjahed et al] [38] présentent une technique pour générer des services composites à partir de descriptions déclaratives de haut niveau. Cette méthode utilise des règles de composabilité pour déterminer dans quelle mesure deux services sont composables. L'approche proposée se déroule en quatre phases :

- La phase de spécification. Elle offre une spécification de haut niveau de la composition désirée en utilisant le langage CSSL (Composite Service Spécification Langage).
- La phase de correspondance. Elle utilise des règles de composabilité pour générer des plans conformes aux spécifications du service demandeur. La phase de sélection. Si plus d'un plan est généré, la sélection est effectuée par rapport à des paramètres de qualité de la composition.
- La phase de génération. Une description détaillée du service composite est automatiquement générée et présentée au demandeur.

La principale contribution de cette approche est la notion de règles de composabilité. Les règles de composabilité considèrent les propriétés syntaxiques et sémantiques des services Web. Les règles syntaxiques incluent des règles pour les types d'opérations possibles et pour les liaisons protocolaires entre les services (i.e., les bindings). Les règles sémantiques incluent des règles concernant la compatibilité des messages échangés et la compatibilité des domaines sémantiques des services, mais également des règles de qualité de la composition. Cette notion de règles de compossibilité met en avant les attributs possibles des services Web qui peuvent être utilisés dans la composition des services et peut ainsi jouer le rôle de directive pour d'autres méthodes de composition par planification.

- **Preuve de théorèmes (Theorem proving)** : [Waldinger et al] dans [39] proposent de générer les services Web composés à partir de preuves de théorèmes. L'approche est basée sur la déduction automatique de preuves. La requête de l'utilisateur est décrite comme un théorème que l'on souhaite prouver. Initialement, les services Web disponibles et les prés requis de l'utilisateur sont décrits dans la logique du premier ordre. la requête est formulée comme un théorème ou la description de service est la spécification et l'objectif recherché est l'entrée. Ensuite, une preuve est générée par le générateur de preuve de théorème SNARK. Enfin, la réponse de la requête (la description de la composition de services) est extraite d'une preuve particulière en fonction de la disponibilité des services Web.

5.2.2. Systèmes Multi-Agents (SMAs)

Du fait de leur autonomie et leur hétérogénéité, la composition des services Web peut être vue comme un système multi-agents ou les agents sont les services Web que chacun sert à satisfaire une partie de la requête de l'utilisateur en utilisant ses propres capacités.

[Müller et Kowalczyk] [40] travaillent sur un système multi-agent pour la composition des services basé sur la concurrence entre coalitions de services : un agent utilisateur fournis les objectifs aux agents représentants les services, ces derniers se contactent les uns les autres

pour proposer leurs services en fonction de leur capacité de raisonnement et ainsi former des coalitions d'agents capables de résoudre les buts. Puis les différentes coalitions vont faire une offre la plus compétitive possible. Les solutions seront notées par l'agent utilisateur. C'est donc la solution ayant la note la plus élevée qui sera choisie. Ainsi, seuls les services Web correspondant le plus aux attentes de l'utilisateur seront utilisés dans la composition.

5.2.3. Approches orientées qualité de service (QoS-Aware)

Plusieurs approches de composition des services Web se concentrent sur le problème de la sélection de services à inclure dans un schéma de composition lorsque les exigences fonctionnelles sont satisfaites par plus d'un service, ces approches discutent que le choix de services en fonction de leur qualité de service (QoS : Quality of Service) est un problème important qui doit être abordée comme un problème distinct.

La qualité de service (QoS) d'un service Web est un critère de performance orienté utilisateur elle définit les propriétés non - fonctionnelles de service telles que le temps de réponse , le cout , la disponibilité …etc. les propriétés QoS sont divisés en deux sous catégories : les propriétés mesurables (débit , temps de réponse , et la latence , etc.) et les propriétés non mesurable (la réputation et la sécurité , etc.). [41] Dans une composition des services Web la qualité de service (QoS) dépendra étroitement de celles des services Web qui la composent. Dans ce qui suit nous présentons des propriétés non-fonctionnelles liées à un service Web (contraintes locales) :

- ✓ **Le coût :** c'est le montant qu'un demandeur de service doit payer pour exécuter le service. Cette valeur est indéterminée lorsque le service n'est pas exécuté.
- ✓ **Temps de réponse :** c'est la durée d'exécution entre le moment où la demande est envoyée et le moment où les résultats sont retournés.
- ✓ **La disponibilité :** c'est la probabilité que le service peut être consulté et utilisé. Ça peut être défini comme le rapport du nombre d'invocations réussies sur le nombre total de demandes.
- ✓ **La réputation :** c'est une mesure de crédibilité d'un service web. Elle peut être définie comme étant la moyenne des classements donnés au service Web par les utilisateurs finaux.

En outre, il pourrait aussi y avoir des contraintes globales, qui peuvent être des politiques, des règlements ou des contraintes dures qui sont appliquées sur toute la sélection, tels que le budget de l'utilisateur.

Le problème de qualité de service (QoS) s'une composition peut être caractérisé comme un problème d'optimisation dont l'objectif par exemple est de réduire le coût et le temps, et de maximiser la disponibilité et la réputation tout en respectant les contraintes globales. [34]

[Lécué] [42] propose une approche orienté QoS dans le contexte du Web sémantique. En particulier, il propose de considérer la qualité des liens sémantiques comme critère d'optimisation supplémentaire. La liaison sémantique entre deux services est définie par les correspondances sémantiques entre les paramètres d'entrée et de sortie des deux. Il argumente que les algorithmes génétiques fournissent une solution plus évolutive au problème d'optimisation.

6. Langage d'exécution des processus d'affaires « BPEL »

6.1. Définition

BPEL (Business Process Execution Language) Conçu par IBM, BEA et Microsoft, c'est la représentation XML d'un processus exécutable, qui peut être déployée sur n'importe quel moteur de processus métier [43]. Un moteur BPEL (BPEL Engine) permet d'orchestrer les différents services selon la définition du processus [44].

L'élément premier d'un processus BPEL est une « activité », qui peut être l'envoi d'un message, la réception d'un message, l'appel d'une opération (envoi d'un message, attente d'une réponse), ou une transformation de données. L'activité est définit par la combinaison de Services Web. BPEL utilise WSDL pour décrire les actions d'un processus [45].

6.2. Les objectifs de BPEL

L'objectif principal de BPEL est de normaliser les processus d'automatisation entre les services Web. En effet, BPEL est considéré comme la technologie clé dans les environnements où les fonctionnalités sont déjà ou seront exposées via des services Web. Parmi les facilités que BPEL peut offrir, nous pouvons citer [46]:

- Décrire la logique des processus d'affaires à travers la composition de services.
- Composer des processus larges et complexes en services et processus moins complexes.
- Corréler les échanges dans et à travers des processus d'affaires.

6.3. Rôle de BPEL

BPEL permet de décrire les processus d'affaires de deux façons différentes [46]:

- **BPEL exécutable :** dans ce type de BPEL, on spécifie tous les détails nécessaires à l'exécution du processus d'affaires. Le BPEL exécutable peut être directement exécuté par un moteur d'orchestration.
- **BPEL abstrait (Abstract BPEL)** : dans cette variante de BPEL, on prend en considération juste l'échange de messages publics réalisé entre les différentes parties participantes au processus d'affaires. Ce type de BPEL ne comprend pas les détails internes du processus et ne peut pas être exécuté. Un BPEL de type abstrait permet principalement de décrire le comportement d'un processus, vu de l'extérieur.

7. Bus de services d'entreprise (ESB) :

Les services Web ont ouvert la voie à d'autres solutions comme par exemple l'ESB (Enterprise Service Bus) ou Bus de Services d'Entreprise. L'ESB est un concept qui a été défini en 2003 par le Gartner Group et est considéré comme la convergence des outils EAI (Enterprise Application Intégration) et des services Web [47].

7.1. Définition d'ESB

Un Bus de Services d'Entreprise (Enterprise Service Bus) est un middleware support de l'Architecture Orientée Services (SOA), intermédiaire de communication entre plusieurs applications de l'entreprise. Son objectif est de répondre au besoin d'interopérabilité technologique entre des systèmes qui n'étaient pas conçus pour fonctionner ensemble, comme deux ERP provenant de deux éditeurs différents, par exemple. Ceci explique que les ESB sont parfois présentées comme une nouvelle génération d'EAI (Enterprise Application Intégration). Support des architectures à base de services, les ESB utilisent des standards tels que XML 47 (comme langage de communication) et les technologies liées aux services web ce qui assure leur ouverture et interopérabilité technologique [48].

7.2. Le rôle d'ESB

La *Figure II.4* présente le Bus de Service d'Entreprise, est un inter logiciel qui permet de regrouper et de faire communiquer toutes les applications informatiques à travers des technologies standards variées.

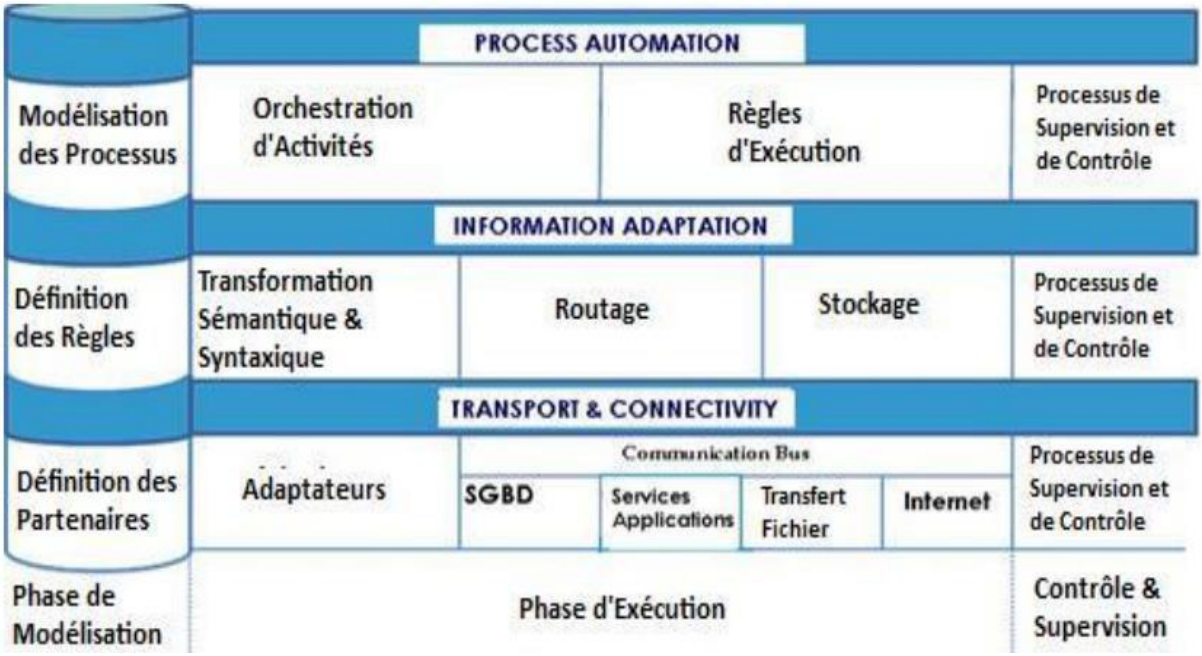

Figure [II.4] – Bus de Services d'Entreprise [48].

Il relie les applications métier et procure une communication entre elles. L'ESB en tant que médiateur entre les clients et les fournisseurs de services s'appuie sur les principes suivants [48] :

- **La découverte dynamique:** les services ainsi que la sémantique associée sont enregistrés dans un annuaire partagé.
- **La chorégraphie des processus métier et l'orchestration des services associés :** un outil permet d'orchestrer automatiquement les services nécessaires à l'implémentation des processus collaboratifs représentés graphiquement.
- **La distribution forte :** les services sont distribués sur le réseau de l'entreprise ou sur Internet.
- **La communication par messages:** les services échangent des messages (représentés par des documents textuels).

7.3. Architecture d'un ESB

L'ESB est une solution « packagée » qui permet de mettre en œuvre l'approche SOA grâce à son concept de base: le bus. Ce dernier permet d'implémenter une solution d'intégration distribuée dépassant l'aspect monolithique des EAI. En effet, le bus permet aux différents services métier (encapsulant des applications d'entreprise) de communiquer et de coopérer. Le bus s'appuie sur un ensemble de services de base [47]:

- Le moteur d'orchestration qui joue le rôle de service d'orchestration permettant d'exécuter des processus,
- Le service de localisation qui permet de trouver de façon transparente les services,
- Les services utilitaires qui sont des services techniques sollicités par les services métier (par exemple, les services permettant le routage, la transformation, etc.),
- Les services d'infrastructure qui sont des services permettant de fournir un support d'infrastructure aux services métier (par exemple, services liés à la sécurité et au monitoring, etc.).

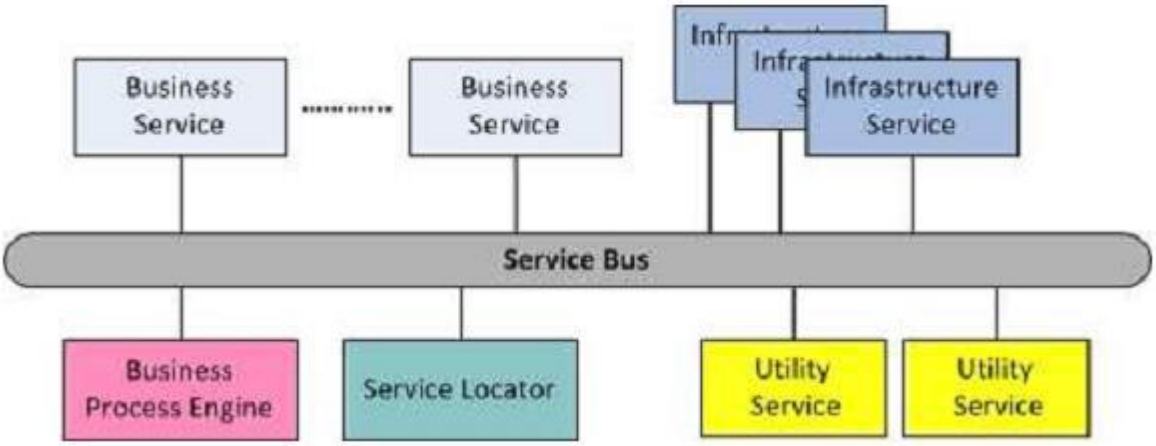

Figure [II.5] – Architecture d'un ESB [47].

Les ESB s'appuyant sur l'utilisation des standards de services Web (SOAP, WSDL, UDDI) et les normes WS-*. Cette particularité facilite en retour l'interopérabilité et l'interconnexion des systèmes d'information des entreprises partenaires en se basant sur un échange des messages transitant d'un bus à un autre (voir *Figure II.5*). Cependant, il s'agit des architectures techniques dédiées à des coopérations à long terme [47].

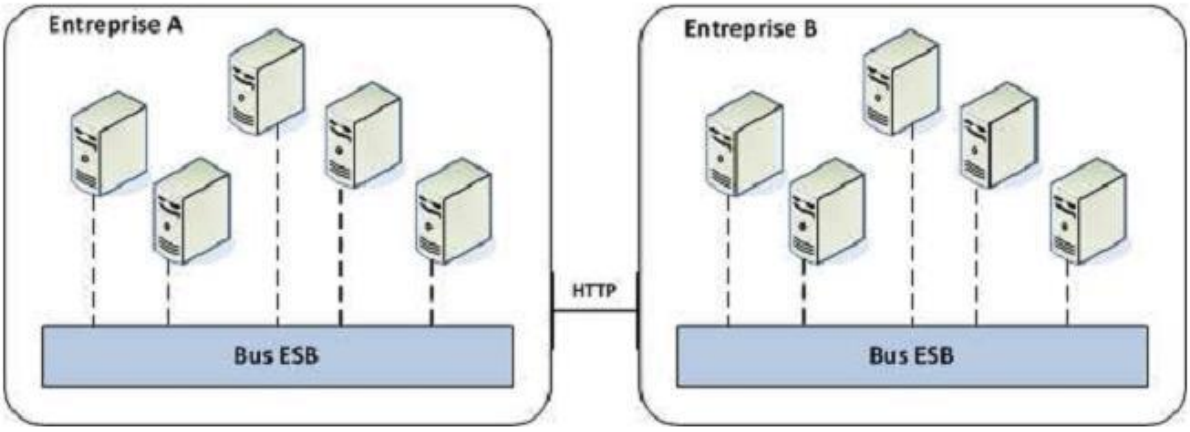

Figure [II.6] – Exemple d'entreprises reliées par leurs bus ESB [47].

7.4. Les objectifs de l'architecture du middleware

L'architecture du middleware support des services industriels doit donc permettre [48]:

• d'assurer une médiation inter-métier au sein de l'entreprise et avec ses partenaires

• de composer les processus en suivant la chaine de la valeur et gérer l'allocation des ressources au plus juste

• de superviser la production et assurer une reprise sur défaillance.

7.5. Les fondements d'un ESB :

ESB s'appuie sur les fondements suivants [48]:

- **Le Middleware Orienté Message (MOM) :** permet d'échanger des messages de manière asynchrone. Ainsi, chaque message est déposé dans une file d'attente avant d'être consommé par le destinataire.
- **Les services web :** permettent d'interfacer les applications avec le bus. A noter qu'un service qui se trouve dans un conteneur de service n'est pas forcément un service web.
- **Les transformations :** concernent les messages circulant sur le bus. Elles sont essentielles dans un ESB car son rôle est de permettre à des applications de converser même si elles définissent différemment leurs données.
- **Le routage :** découple l'expéditeur du message de son destinataire. C'est en fait l'ESB qui va déduire la destination du message. Pour cela, il se base sur le contenu du message et les règles qui ont été définies.

8. Conclusion

La composition des services web est un point crucial qui a un grand impact sur plusieurs domaines de recherches. Beaucoup d'efforts ont été fourni afin de permettre une composition utilisable et acceptable de service web. Le but de la composition est de réutiliser des services existants pour réaliser des fonctionnalités complexes, elle permet de réduire le cout et le temps de développement des applications services web. Dans ce chapitre nous avons défini ce qu'est la composition des services web et nous avons exposé les types et techniques de composition ainsi les différents approches existantes. Et enfin, nous avons présenté le bus de

services d'entreprise (ESB) : définition d'ESB, le rôle d'ESB, architecture d'un ESB, les objectifs de l'architecture du middleware et les fondements d'un ESB.

Chapitre 03 :

Analyse et conception

Chapitre 03 : Analyse et conception

1. Introduction

Les services Web sont considérés comme des implémentations de SOA, ils sont devenus incontournables pour la réalisation d'applications réparties sur le Web. La technologie des services Web est aujourd'hui largement utilisée comme support de l'interopérabilité entre les applications distribuées, qui fonctionnent indépendamment des caractéristiques conceptuelles (architecture, modèle d'interaction,..) et des caractéristiques techniques (plateforme, langage de programmation, ...) afin de réaliser une fonctionnalité établie auparavant.

Dans cette partie du mémoire, nous allons proposer une architecture progressive de composition dynamique de services Web, en partant d'une requête déclarative simple spécifiée par un utilisateur. Notre principale contribution est de proposer une architecture complète pour la composition dynamique des services Web, car nous considérons qu'une composition de services Web doit s'engager depuis la découverte de services Web jusqu'à l'interaction avec les utilisateurs. La découverte des services Web participants, leur composition ainsi que leur interopération doivent être effectuées de façon automatique et transparente à l'utilisateur.

2. Architecture générale du système

Dans notre travail, nous allons proposer un système pour la composition automatique des services web, dans le but d'offrir à l'utilisateur de nouvelles fonctionnalités en combinant des fonctionnalités déjà existantes par d'autres services.

L'objectif de notre système est alors de créer des services composites, en réponse à une requête de l'utilisateur qui ne peut être satisfaite par un seul service. La composition que nous proposons est de type automatique et statique:

- **Automatique**: parce que le compositeur proposé effectue la composition entièrement de manière automatique sans l'intervention de l'utilisateur.
- **Statique**: le compositeur proposé peut composer des services selon un ordre bien définit à l'avance.

Le système est générique et peut être appliqué à n'importe quel domaine. Toutefois nous avons choisi une étude de cas qui est la vérification de la possibilité pour les citoyens d'extraire le passeport ou demander un visa.

Le service composite est conçu pour satisfaire des requête complexes de l'utilisateur (quand il entre plusieurs données). Ces services interrogent une base de données qui contient

les informations nécessaire pour le système. Tout d'abord, nous illustrons la vue globale de web composite dans la figure suivante :

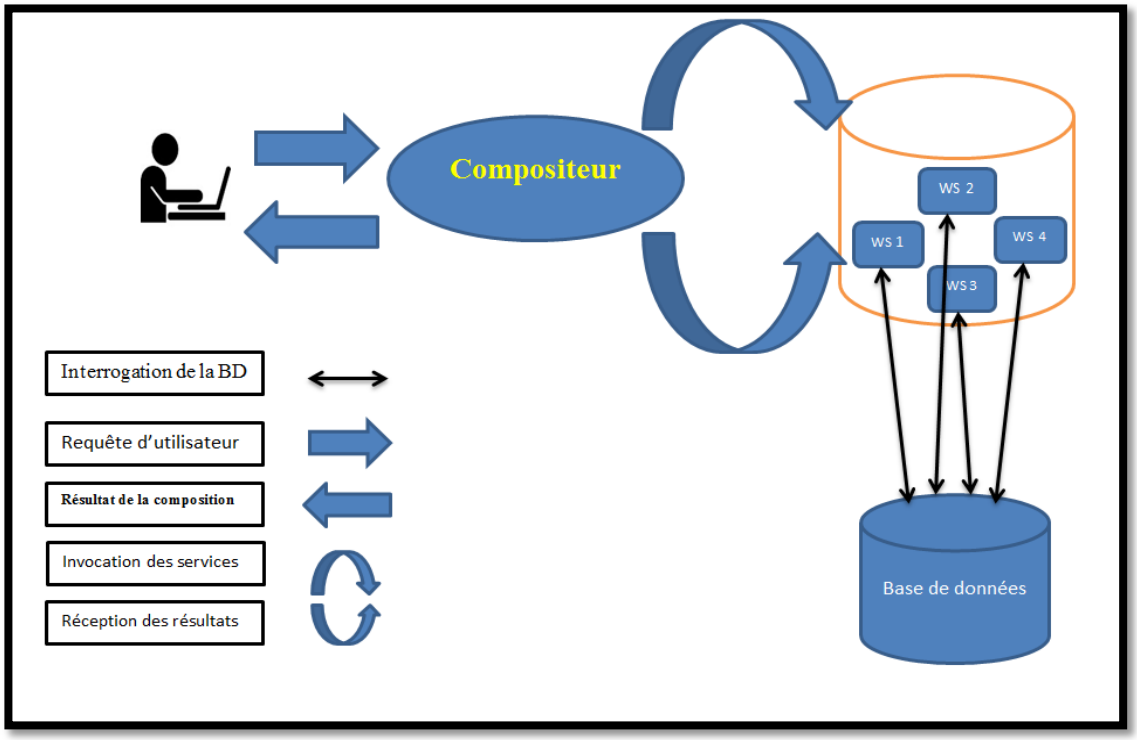

Figure [III.1]: L'architecture générale du système.

Les parties importantes du système sont décrites ci-après :

2.1 L'utilisateur

Il interagit avec le système par envoi des requêtes et réception des résultats, à savoir la composition qui répond à sa requête si elle existe.

2.2 Le compositeur

Nous proposons à l'utilisateur un web composite pour les services dédiés (état civil, sécurité, tribunal). Pour répondre au plus aux besoins des utilisateurs Il se charge des fonctionnalités suivantes *:*

- Produire la composition satisfaisant la requête de l'utilisateur à partir d'une base de services.
- Interagir avec l'utilisateur afin de lui fournir la meilleure composition.

3. La composition des services web

Cette opération consiste à analyser la requête envoyée par l'utilisateur via une interface graphique proposé par le système et respectant l'ordre dans lequel ces services vont être composés en utilisant des opérateurs bien définis.

3.1 Ordre d'exécution de la composition

Nous avons utilisés deux types d'exécution des opérations: parallèle et séquentielle.

Soit a et b deux web services :

- Si l'exécution de **a** et **b** est séquentielle (**a*b**): exécuter **a** avant **b**
- Si l'exécution de **a** et **b** est parallèle (**a||b**): exécuter en même temps **a** et **b**

- **Séquence :** Dans une séquence, un service Web est disponible une fois que les services Web précédent ont terminés leur exécution.

Figure [III.2] : Schématisation d'une Séquence.

- **Parallélisme:** Le parallélisme est un lien de contrôle se divisant en plusieurs liens s'exécutant en parallèle. En d'autres termes, les services Web qui appartiennent à ce type de contrôle des flux s'exécutent en parallèle.

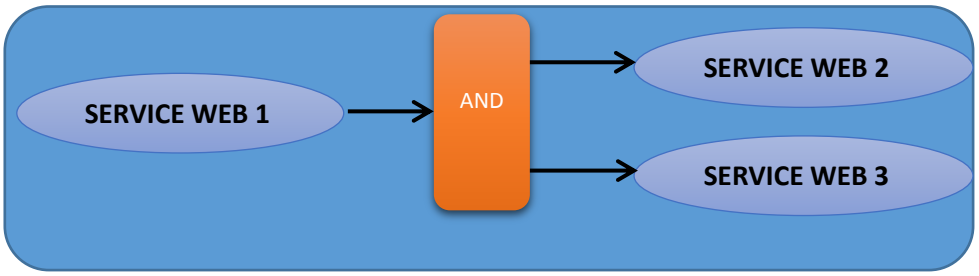

Figure [III.3] : Schématisation d'un parallélisme.

- **Synchronisation :** La synchronisation est un point dans le procédé où plusieurs flux de contrôle Convergent et s'exécutent comme un seul flux de contrôle synchronisant tous ces liens.

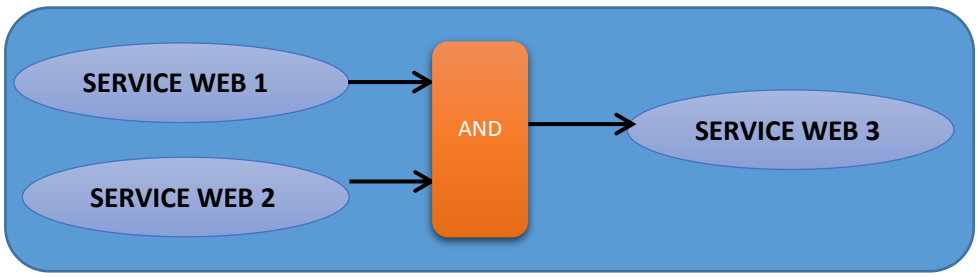

Figure [III.4]: Schématisation d'un Synchronisation.

- **Choix exclusif :** le Choix exclusif est un point dans le procédé ou un chemin est choisi parmi plusieurs, Ce choix est fait a l'aide d'une décision prise au moment de l'exécution. Cette décision basée sur une information ou une donnée procédé

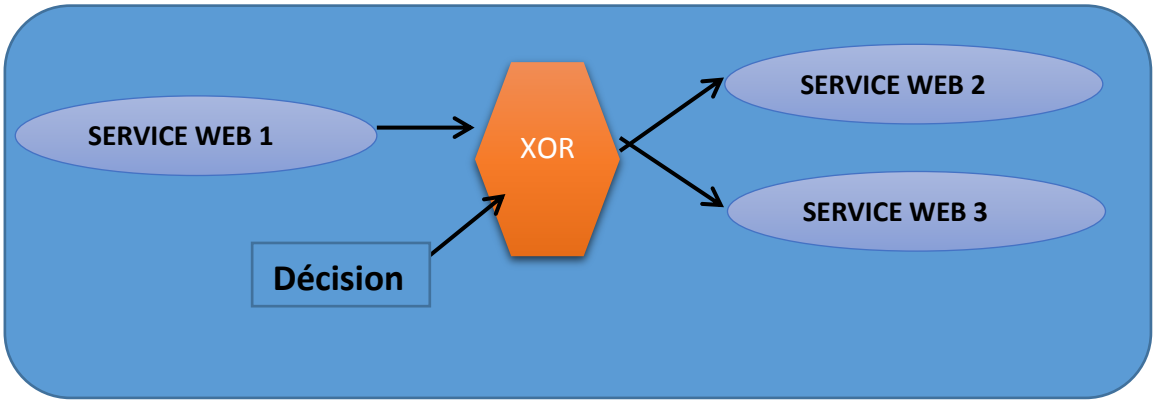

Figure [III.5]: Schématisation d'un choix exclusif.

- **Fusion simple :** le fusion simple est un point dans le procédé dans lequel une ou plusieurs branches du flux de contrôle se joignent sans nécessité de synchronisation . au moment d'arriver a la terminaison de l'exécution du service web 1 ou du service web 2 ou des deux, le procédé continue avec l'exécution du service web 3.

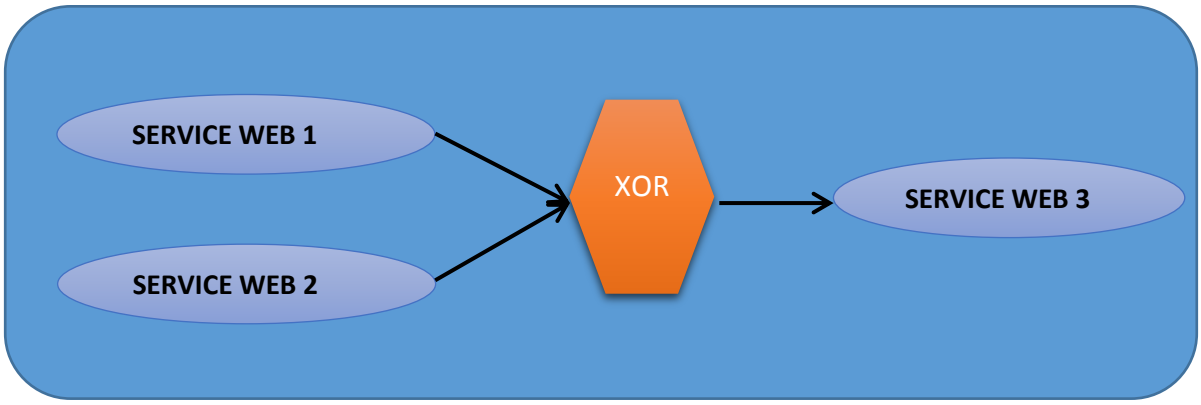

Figure [III.6]: Schématisation de la fusion simple.

- **Choix Multiple :** Le choix multiple est un point dans le procédé où plusieurs branches sont sélectionnées et exécutées en parallèle. Ce choix est fait à l'aide d'une décision

prise au moment de L'exécution Cette décision est basée sur une information ou une donnée du procédé. Après la terminaison de l'exécution du service Web 1, plusieurs services seront exécutés en parallèle.

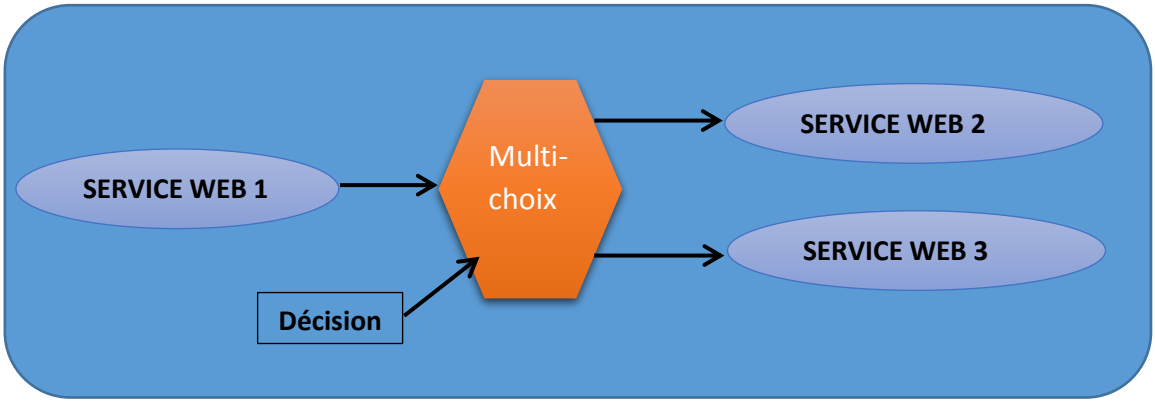

Figure [III.7]: Schématisation du choix multiple.

- **Choix différé :** ce lien représente un point dans un procédé dans lequel une branche est choisie parmi plusieurs .ce choix est basé sur une information ou une donnée qui n'est pas nécessairement disponibles au moment où ce point du procédé est atteint . la sélection de la branche est alors retardée jusqu'à ce que certains événement arrivent pour donner au procédé l'information requise.

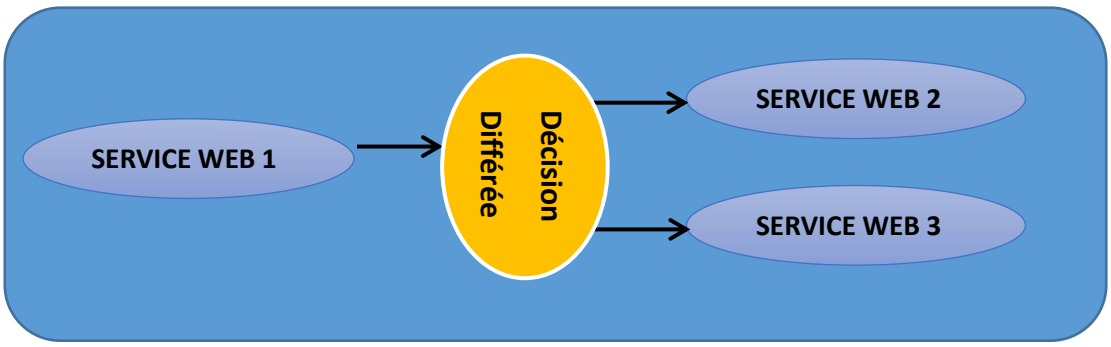

Figure [III.8]: Schématisation du choix différé.

4. Description détaillée de notre système de composition

Dans cette partie nous allons présenter la conception de notre système de composition des services web qui est articulé auteur des modules suivant: Module de composition, nodule de

découverte, module d'exécution et module de publication. Ces modules sont développés indépendamment et peuvent collaborer entres eux pour Réaliser leurs tâches respective (compositions, recherche, exécution et publication). Ils permettent Ainsi d'atteindre l'objectif global du système qui est la composition automatique des services web et L'invocation (exécution) du service composite.

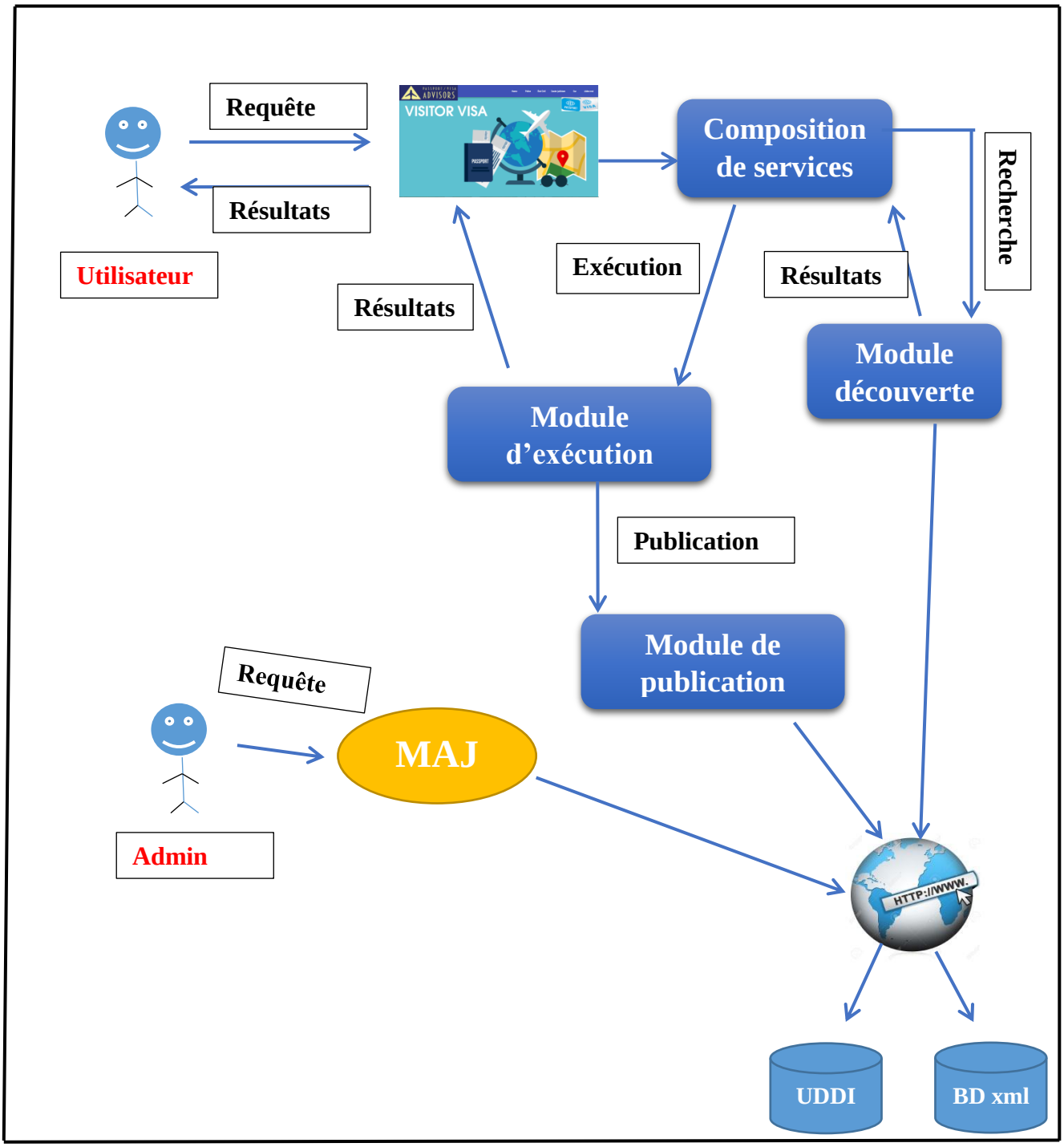

Figure [III.9]: l'architecture détaillée de notre système de composition des services web

4.1. Module d'Interface

Le module interface est considéré comme la première fenêtre du système vers le monde des utilisateurs , il représente la partie visible de l'architecture , ce module est responsable de la communication entre le système et le différents utilisateurs ,il fournit des outils a l'utilisateurs pour formuler des requêtes et visualiser les résultats de notre système , l'utilisateur possède a son niveau une interface conviviale, simple et adaptable, qui permet a l'utilisateur de formuler sa requête selon ses besoin pour interroger les services web , et ses préférences en termes de qualité de service Web. Le module interface assure les fonctionnalités suivantes :

. Identification de l'utilisateur.

. Transmission des requêtes de l'utilisateur au système pour les traiter.

. Présentation des résultats de notre système d'une manière adaptée aux besoins de l'utilisateur.

4.2. Module de la composition des services web

La composition est la fonctionnalité principale du système. Lors de cette étape, le module Composition fait appel au module de découverte qui permet de faire une recherche des services dans L'annuaire UDDI et sélectionner ceux qui peuvent satisfaire la requête de l'utilisateur, le résultat ce module Est une liste des services web. Le module de composition compose les services Sélectionnés afin d'obtenir un service composite exécutable.

4.3. Module d'exécution des services web composites

Cette fonctionnalité permet à l'utilisateur d'exécuter ou d'invoquer un service composite résultant de la composition.

4.4. Module de publication des services web composites

Le module de composition envoie au système de publication le service web composite à publier dans l'annuaire des services pour la réutilisation inter-applications.

5. La conception de l'architecture

La conception s'attache à décrire le fonctionnement de notre architecture du système que nous baptisons « la composition des services web», ceci se traduit en règle générale par la définition des classes de service pour cette étape nous allons utiliser un langage de modélisation UML.

5.1. UML

UML (Unified Modeling Language), que l'on peut traduire par « langage de modélisation unifié » est un langage graphique de modélisation de différents problèmes de façon standard, Initialement conçu pour représenter, spécifier, concevoir et documenter les éléments d'un modèle de systèmes logiciels [siteweb1].

Dans ce projet nous avons utilisés quelque diagrammes tels que diagramme de cas d'utilisation, diagramme de class et diagramme de séquence.

- *Diagramme de cas d'utilisation* : Représente les fonctions du système de point de vue des utilisateurs
- *Diagramme de classe*: Représente la structure statique en termes de classe et relation
- *Diagramme de séquence* : sont une représentation temporelle des objets et de leurs interactions.

5.2. Le diagramme de cas d'utilisation :

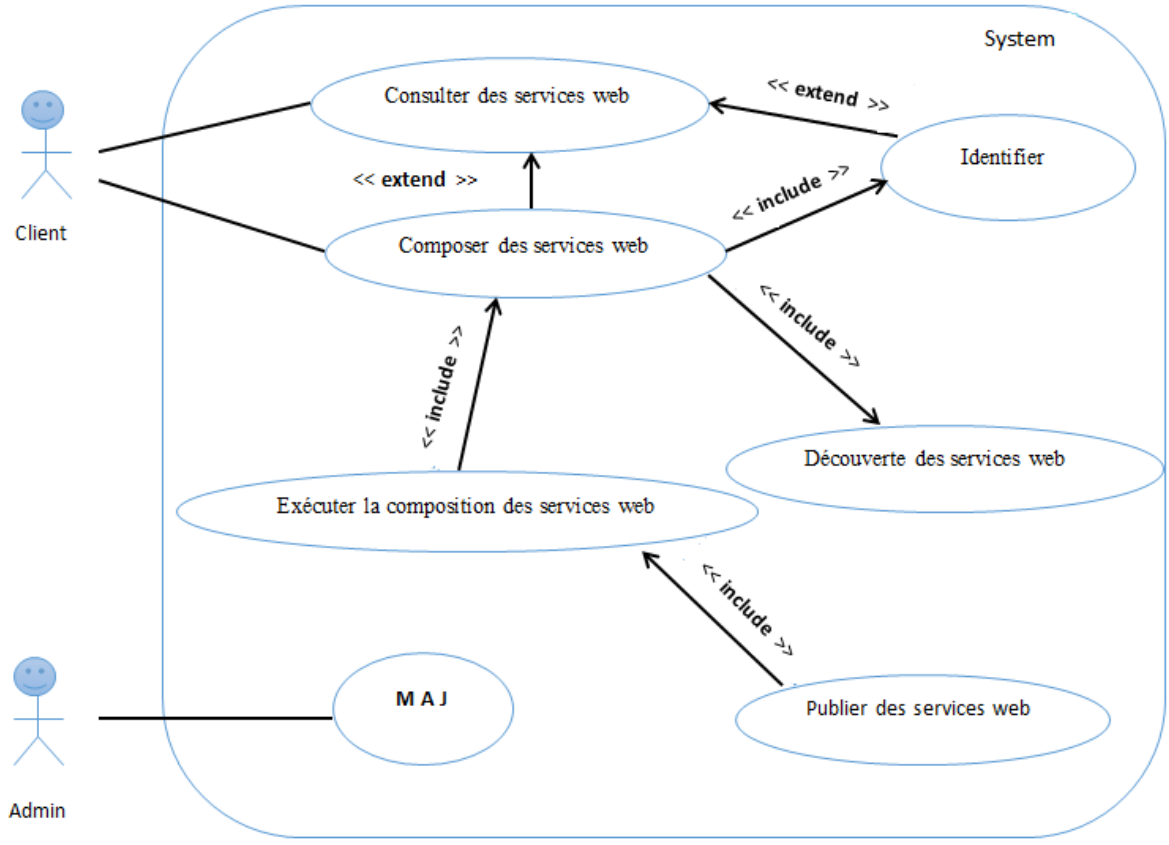

Figure [III.10]: Système de composition des services web.

5.3. Le module de gestion du profil utilisateurs

Le module profil utilisateurs contient une fonctionnalité :

- **Création de profil utilisateur** : dans cette étape, il est demandé à l'utilisateur de ses informations telles que numéro de la carte d'identité, nom, prénom…, ces informations obligatoires pour accéder à notre système de composition ;

5.4. Le module de découverte

Ce module assure la fonctionnalité de recherche des services indépendamment des autres modules. il permet d'analyser la requête de l'utilisateur envoyée au système et recherche les services web dans l'annuaire UDDI qui répondent le plus aux besoins de l'utilisateur. Le résultat de ce module est une liste de services participants à la composition

5.5. Le module de composition

Le module de composition est le composant essentiel du système. Il permet la construction du Service composite qui répond à la requête de l'utilisateur .Ce module s'exécute lorsque la requête est complexe et nécessite une composition de plusieurs services.

Le module de composition reçoit la liste des services participants **à** la composition grâce au module précédant (module de découverte)puis exécute un algorithme de composition proposé dans notre travail, à la fin de cette étape le système exécuter et publier le résultat de la composition.

Le diagramme de séquence de la figure dessous nous montre les différentes phases de composition

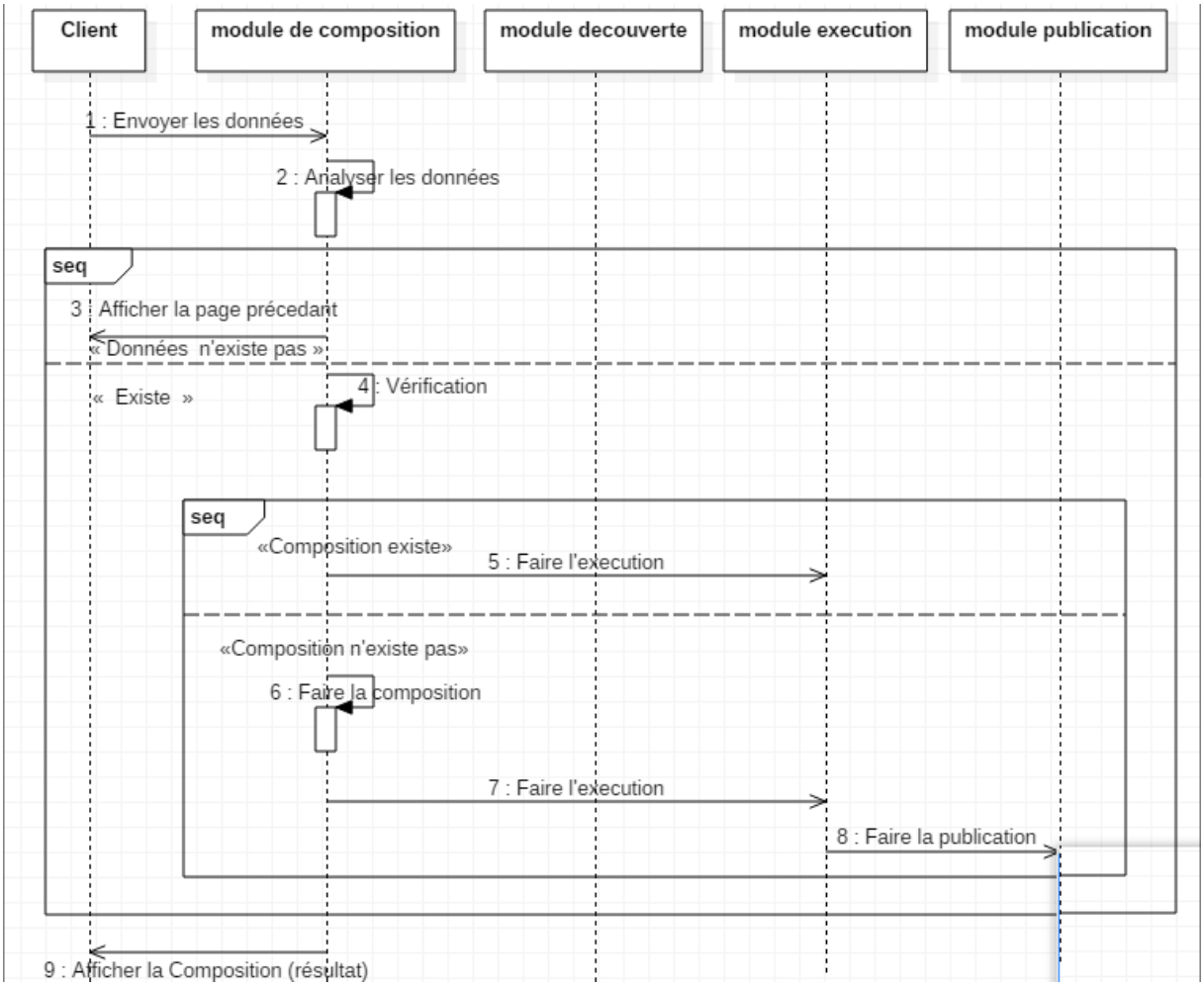

Figure [III.11]: diagramme de séquence de composition

5.6. MAJ

- Le Diagramme de Séquence suivant décrit le processus d'authentification effectué par l'administrateur afin d'accéder au profil

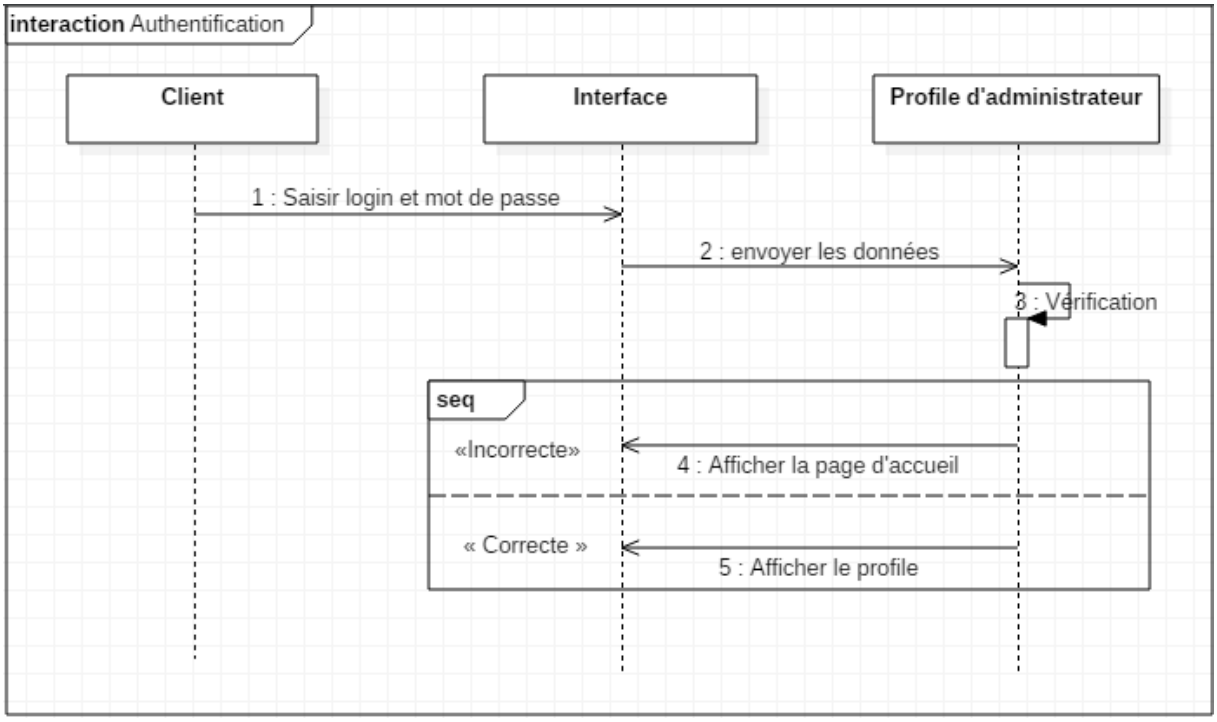

Figure [III.12]: Diagramme de séquence d'authentification.

L'administrateur peut être modifié, supprimé et ajouté des services web selon les besoins de l'utilisateur.

- Le diagramme de séquence de la figure dessous nous montre les différentes phases du processus de MAJ

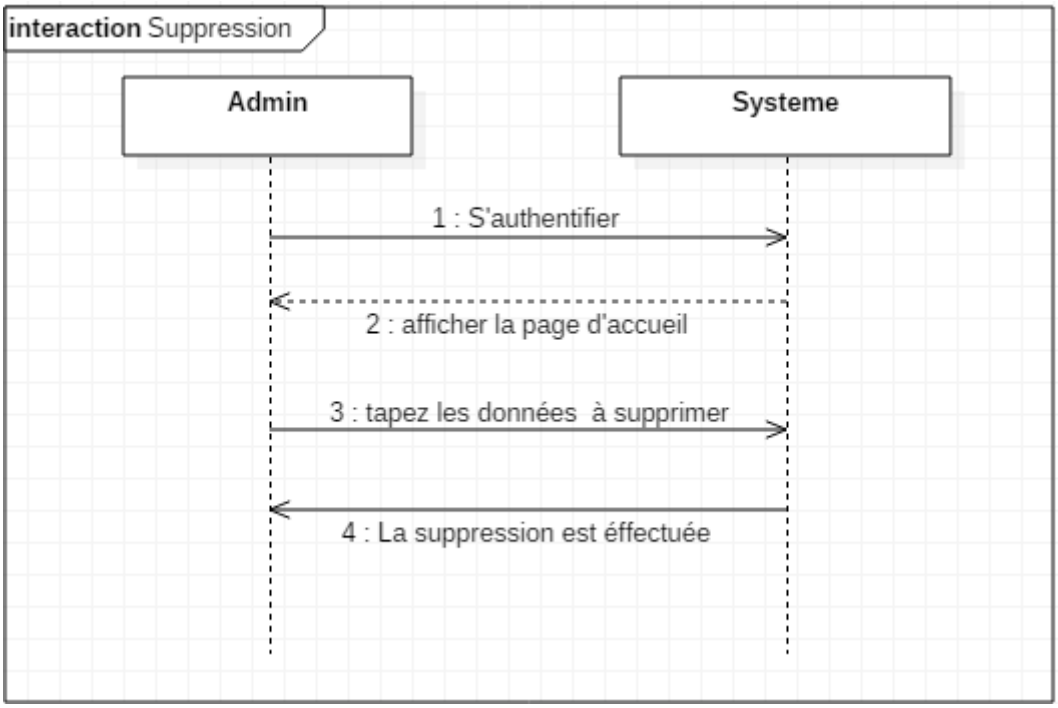

Figure [III.13]: Diagramme de séquence de suppression.

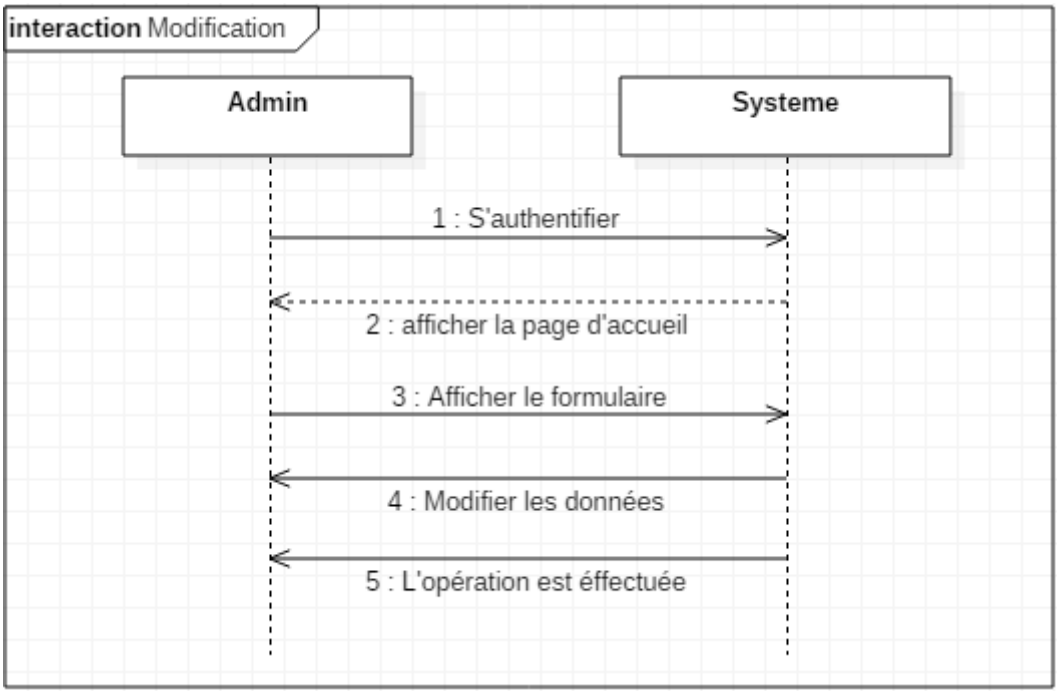

Figure [III.14]: Diagramme de séquence de Modification.

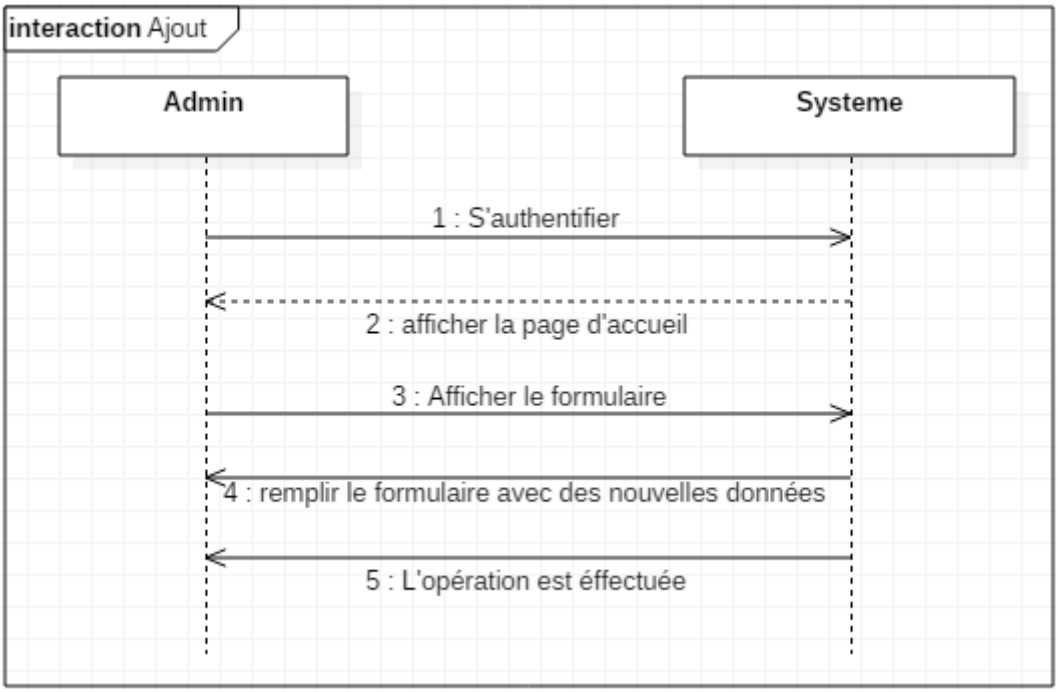

Figure [III.15]: Diagramme de séquence d'ajout.

5.7. Diagramme de classe :

Diagramme de classe de la figure dessous nous montre la différente table de base donnée

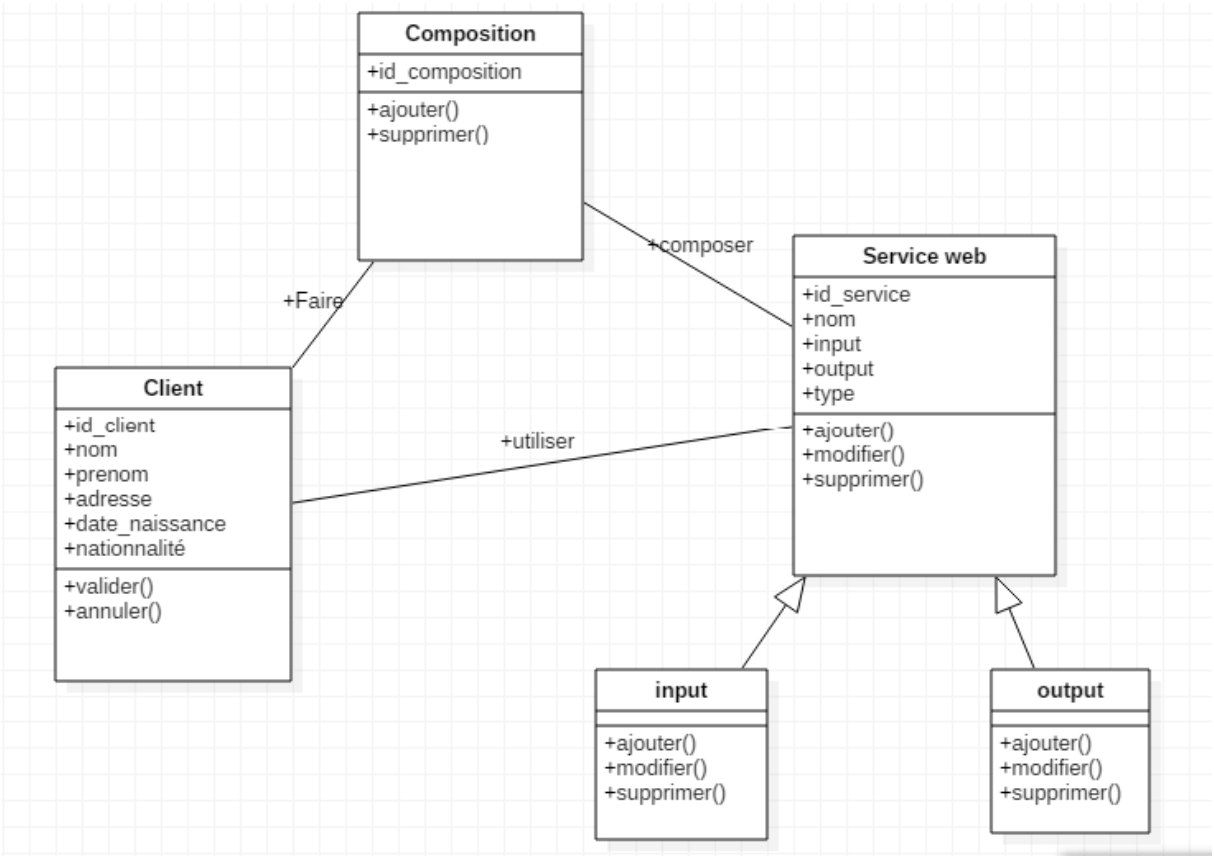

Figure [III.16]: Diagramme de classe de système.

6. conclusion

La composition de services web a pour but d'utiliser les compétences de plusieurs services afin de résoudre un problème qu'aucun ne saurait résoudre individuellement, dans ce chapitre nous avons abordé la partie conception du système, nous avons présenté une architecture globale qui décrit le fonctionnement générale du système, nous avons par la suite procédé à spécifier les différentes composants du système, ainsi que leurs fonctionnellement.

Dans le chapitre qui suit, nous préciserons les outils d'implémentation ayant servi la réalisation de ce travail.

Chapitre 04 :

Implémentation du système

Chapitre 04 : Implémentation

1. Introduction

Après la présentation de la conception du notre système dans le chapitre précédent, nous allons dans ce chapitre décrire les différents aspects techniques liés à l'implémentation et la mise en œuvre de notre application de composition des services web.

Nous avons fait appel à un ensemble d'outils et de langages de développement permettant la réalisation de notre système, nous tenons à les présenter tous en argumentant nos choix avant d'entamer la description de notre implémentation. Et pour bien illustrer notre système, nous allons présenter les différentes interfaces ainsi que chacune de leurs fonctionnalités.

2. Environnement de développement

Avant de commencer l'implémentation de notre système, nous allons tout d'abord spécifier les langages de programmation et les outils utilisés qui nous ont semblé être un bon choix vu les avantages qu'ils offrent.

2.1. Environnement matériel et logiciel

Pour implémenter notre système, nous avons utilisé un PC I5 doté de Windows 7 (64bits), figure IV .1 :

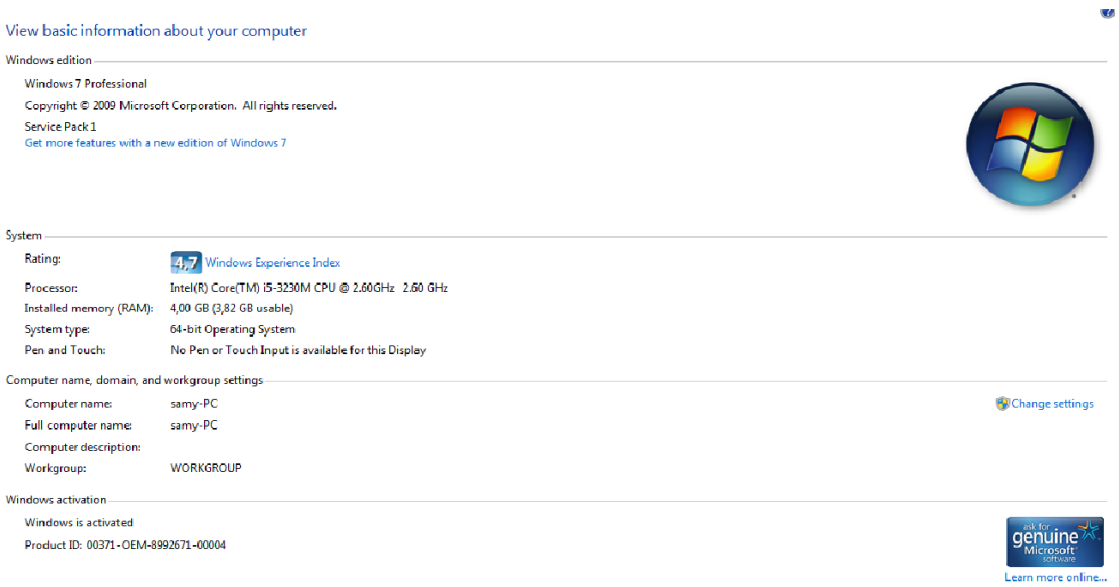

Figure [IV .1] – Environnement logiciel utilisé.

2.2. Les outils utilisés

Notre avons utilisé le langage de programmation Java sous L'environnement NetBeans et les deux GlassFish et OpenESB Standalone comme des serveurs d'applications. MySQL a aussi été utilisé comme système de gestion des bases de données.

2.2.1. Langage de programmation

Pour l'implémentation de notre système nous avons opté pour le langage Java (version du JDK : 1.6.0_29). 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 [49]:

- 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).
- 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 JDBC-ODBC 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.

2.2.2. MySQL

MySQL est un véritable serveur de base de données SQL très rapide, multithreads. Multiutilisateurs et robuste. SQL « Structured Query Language » : le langages standard pour les Traitements de bases de données est le plus populaire langage de base de données dans le made. MySQL est une configuration client/serveur ce qui consiste en un serveur démon MySQL différents programmes clients et des librairies.

SQL est un langage standardisé qui rend facile le stockage, la mise à jour et l'accès à l'information. Par exemple. Nous pouvons utiliser le SQL pour récupérer des informations sur un produit ou stocker des informations client sur un site web. MySQL est suffisamment rapide et flexible pour gérer des historiques et des images [50].

2.2.3 OpenESB Standalone

- ✓ *Standardisation des concepts* Les ESB supportent des standards tels que XML, JMS, JCA, JMX et les nombreux standards relatifs aux Web Services. Ce qui implique une intégration des systèmes en place plus rapide, plus économique et plus souple. [54]
- ✓ *Intelligence du routage* Les bus ESB automatisent le routage des transactions commerciales en fonction du contenu des documents XML et des règles commerciales établies. Il n'est ainsi plus nécessaire de programmer cette fonctionnalité dans le code de l'application. A
- ✓ *Architecture de services* Les fonctionnalités fournies par les ESB sont implémentées comme des services spécialisés distincts (service de transformation, service de routage intelligent, service de loging…). Ces services implémentés dans de petits containers peuvent être déployés indépendamment les uns des autres, de façon sélective.
- ✓ *Architecture distribuée* L'architecture de services des ESB est distribuée avec beaucoup plus de modularité qu'une architecture monolithique, ce qui permet d'apporter une réponse précise et souple aux besoins de scalabilité rencontrés dans les problématiques de montée en charge.
- ✓ *Fiabilité* Les ESB permettent de construire des architectures sans point individuel de défaillance (SPOF)). Ainsi, quand un serveur tombe en panne, le reste du système peut continuer à fonctionner.

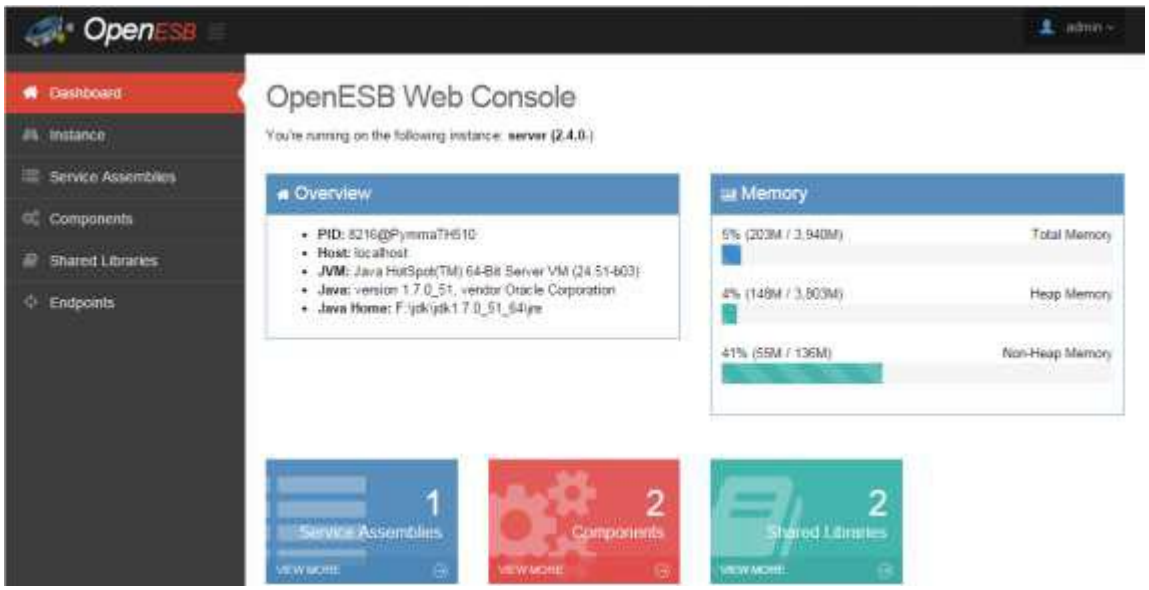

Figure [IV.2] – OpenESB Web admin console [54]

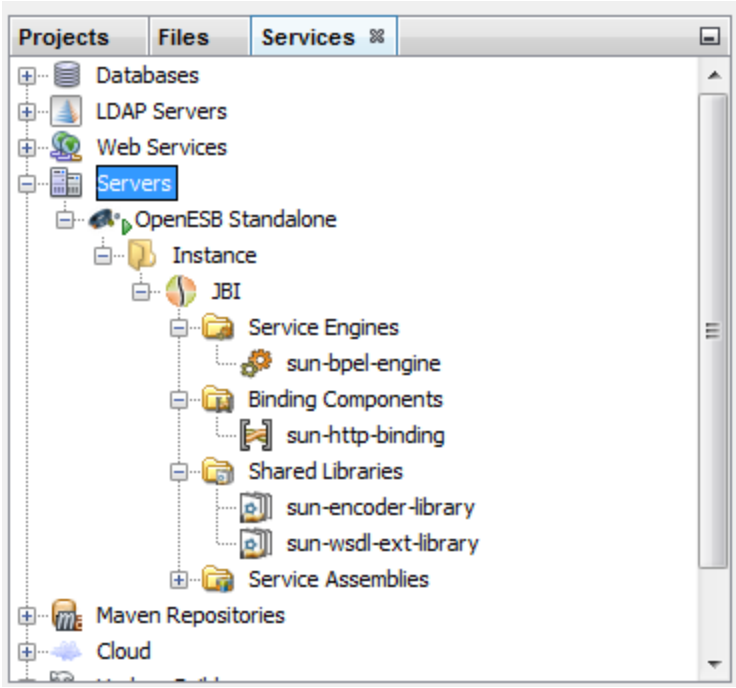

Figure [IV.3] – OE Studio Connection to OE SE instance. [54]

2.2.4 NetBeans 8.2

Nous avons utilisé Netbeans version 8.2, le choix de Netbeans était fondamental puisqu'il est un logiciel permettant principalement le développement en Java. Mais aussi il permet également de supporter différents autres langages, comme C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). Il fournit un environnement standard de développement pour créer des interfaces très puissantes. NetBeans est un environnement de développement intégré (IDE) basé sur des normes, en une plate-forme d'application cliente riche, qui peut être utilisée comme structure générique pour créer n'importe quel type d'application avec une plus grande assurance de robustesse et de concevoir des applications qui résisteront à l'épreuve du temps. Il est placé en open source par Sun en juin 2000 sous licence CDDL (common development and Distribution license).). NetBeans est disponible sous Windows, Linux et d'autres systèmes d'exploitation. Le projet de NetBeans IDE consiste en un EDI Open Source complet écrit dans le langage de programmation Java [51].

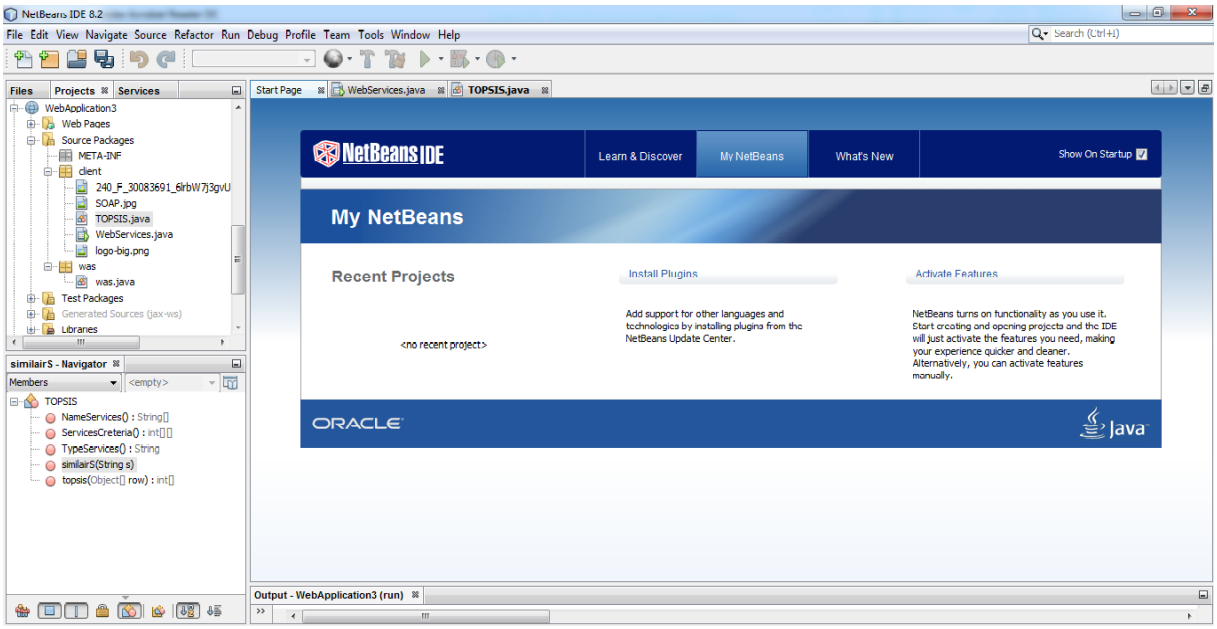

Figure [IV.4] – Interface principale de NetBeans.

2.2.5 GlassFish

GlassFish est un serveur d'applications Open Source Java EE 5 et désormais Java EE 7 avec la version 4.1 qui sert de socle au produit Oracle GlassFish Server2 (anciennement Sun Java System Application Server3 de Sun Microsystems). Sa partie Toplink persistence4provient d'Oracle. C'est la réponse aux développeurs Java désireux d'accéder aux sources et de contribuer au développement des serveurs d'applications de nouvelle génération.[52]

2.2.6 Star UML

StarUML est un logiciel de modélisation UML, cédé comme open source par son éditeur. StarUML gère la plupart des diagrammes Spéciñés dans la norme UML 2.0 [53]. Il a été utilisé, dans notre cas, pour la présentation des diagrammes UML.

2.2.7 JavaFX

JavaFX est une technologie créée par Sun Microsystems qui appartient désormais à Oracle, à la suite du rachat de Sun Microsystems par Oracle le 20 avril 2009.

Avec l'apparition de Java 8 en mars 2014, JavaFX devient l'outil de création d'interface graphique ('GUI toolkit') officiel du langage Java, pour toutes les sortes d'application (applications mobiles, applications sur poste de travail, applications Web...), le développement de son prédécesseur Swing étant abandonné (sauf pour les corrections de bogues).

Notre choix s'est porté sur le JavaFX pour diverses raisons :

- JavaFX contient des outils très divers, notamment pour les médias audio et vidéo, le graphisme 2D et 3D, la programmation Web, la programmation multi-fils etc.
- Le SDK de JavaFX étant désormais intégré au JDK standard Java SE, il n'y a pas besoin de réaliser d'installation spécifique pour JavaFX.
- Des projets libres complètent JavaFX en fournissant des composants de haute qualité absents de JavaFX proprement dit

3. Etude de cas

Nous avons mis en œuvre un système pour la demande d'un passeport ou un visa (état civil, sécurité, tribunal) basé sur les services web.

3.1. Scénario

Cette étude de cas et son implémentation montrent la faisabilité des différentes phases de notre conception et vise à exposer les différents concepts utilisés dans cette conception. Notre conception implémente l'algorithme de composition des services web, pour cela, un exemple de la vérification situation des citoyens pour extraite les papiers du passeport ou visa. La demande de l'utilisateur (client) en question doit avoir recours durant son exécution à des services web.

Notre agence de consultation fournit typiquement les services pour : Affirmer le statut légal du citoyen par rapport à la justice, vérifiez le statut du citoyen pour la police, vérifier toutes les informations d'identité relatives au citoyen. Afin de fournir ces services à ses clients, nous allons aborder quelques définitions des fonctionnalités citées précédemment.

1. **Service web Etat Civil** : Retourne la disponibilité des informations d'identité d'un demandeur

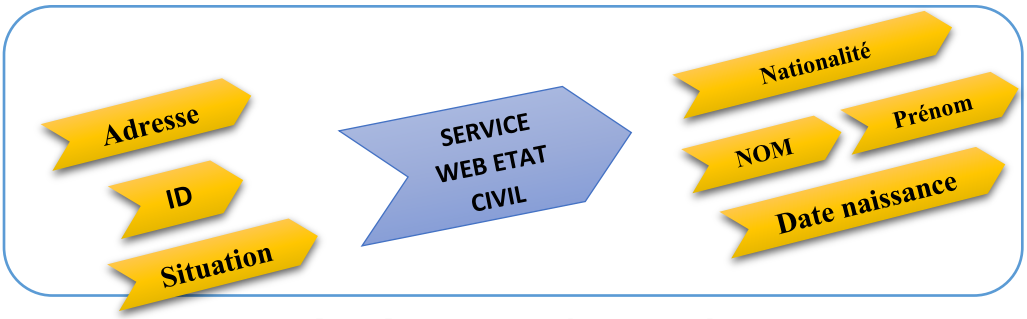

Figure [IV.5] – Service web Etat Civil

2. **Service Web sécurité :** retourne l'information concernant la situation programmée selon le numéro d'identité.

Figure [IV.6] – service web Sécurité.

3. **Service Web Tribunal :** retourne l'information concernant la situation programmée selon le numéro d'identité.

Figure [IV.7] – service web Tribunal.

4. **Service web Validité demande :** assure la validité de demande si une demande déjà existe n'enregistre pas sinon on enregistrera dans une liste des demandes.

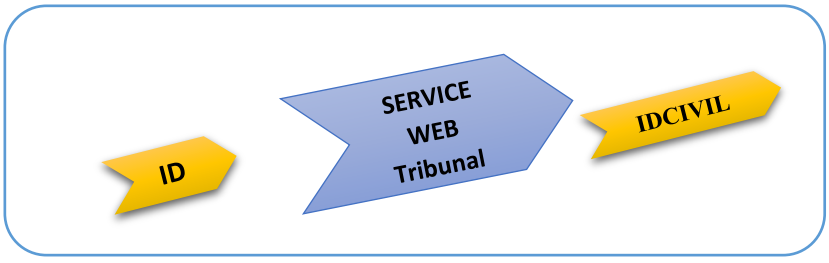

Figure [IV.8] – service web Validité demande.

3.2. Présentation des interfaces graphiques

Après avoir justifié nos choix des outils de développement, nous allons illustrer les fonctionnalités de notre système en effectuant quelques prises d'écran qui nous montrent les différentes étapes nécessaires à la réalisation d'une composition automatique des services web basée sur leurs comportements ainsi que l'exécution et la publication du service composite.

Notre système propose à l'utilisateur, des interfaces qui lui permettront de suivre en détails les différentes opérations pour la réalisation du processus de composition.

3.2.1. Interface d'accueil

La qualité de l'interface est l'une des caractéristiques qui attire l'utilisateur de cela nous avons essayé de la représenter dans une bonne forme tout en respectant l'aspect de simplicité.

Figure [IV.9] – Interface principale de l'application.

3.2.2. Consultation

Pour consulter le contenu des différent services, il suffit de cliquer sur les liens de la liste(1), puis taper (2) le numéro d'identité désiré ou les autres informations (nom, prénom,…) selon le service web, enfin appuyez sur le bouton continue(3). Les trois figures suivantes montrent la consultation de service web état civil, tribunal et sécurité.

Figure [IV.10] – page de consultation service web sécurité.

Figure [IV.11] – page de consultation service web Tribunal.

Figure [IV.12] – page de consultation service état civil.

3.2.3. Requête utilisateur :

Le système propose à l'utilisateur pour créer une requête de composition qui exprime au mieux ses besoins fonctionnels.

```
<soapenv:Envelope xsi:schemaLocation="http://schemas.xmlsoap.org/soap/er
  <soapenv:Body>
    <com:MyWsdlOperation>
      <input>
        <myt:Num>123456</myt:Num>
        <myt:Nom>CHETTI</myt:Nom>
        <myt:Prenom>SAMI</myt:Prenom>
        <myt:Adresse>COLLO</myt:Adresse>
        <myt:Date_naissance>17-12-1993</myt:Date_naissance>
        <myt:Situation>CELEBATAIRE</myt:Situation>
        <myt:Fonction>ETUDIANT</myt:Fonction>
        <myt:Nationnalite>ALGERIEN</myt:Nationnalite>
      </input>
    </com:MyWsdlOperation>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure [IV.13] – Création d'une requête de composition.

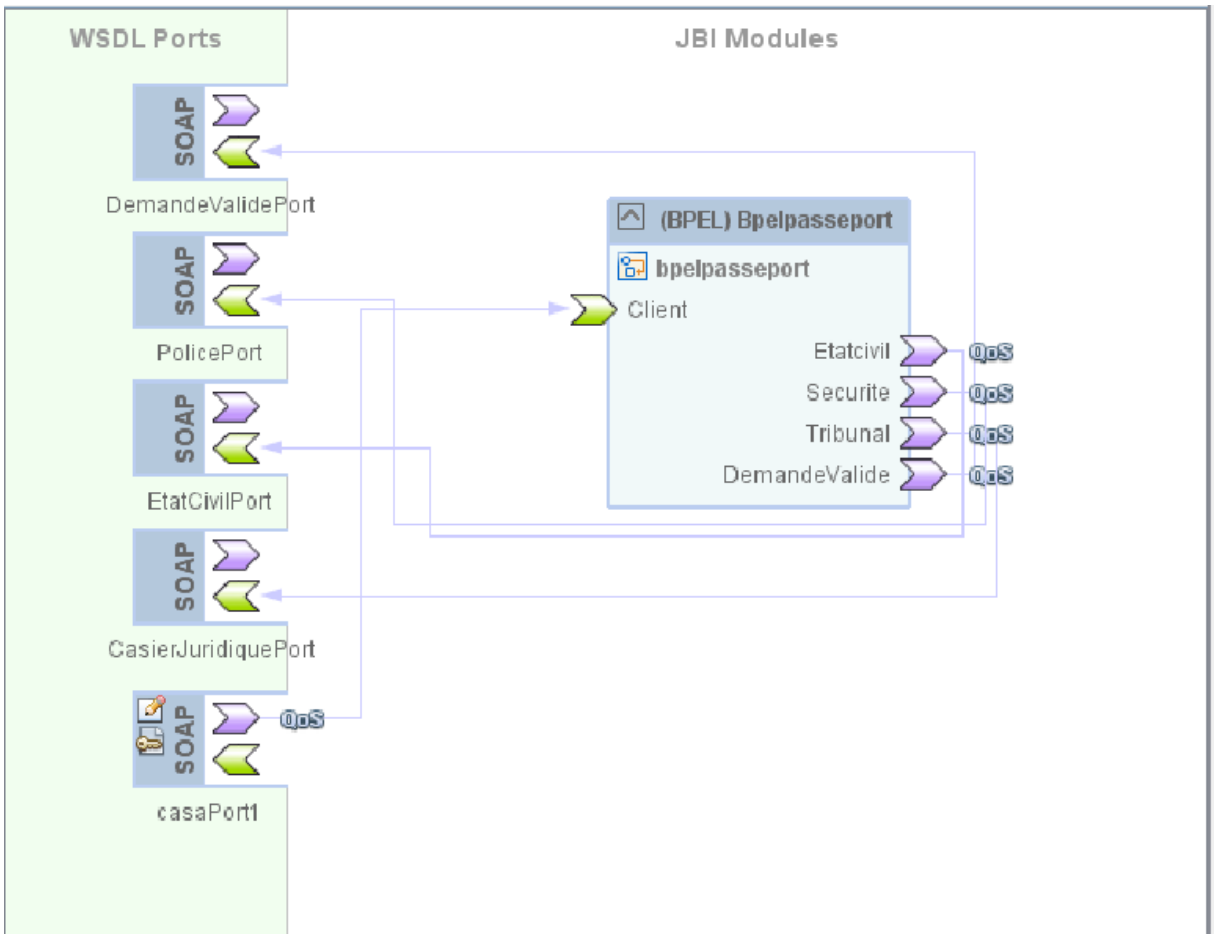

Figure [IV.14] – schéma d'un service composite.

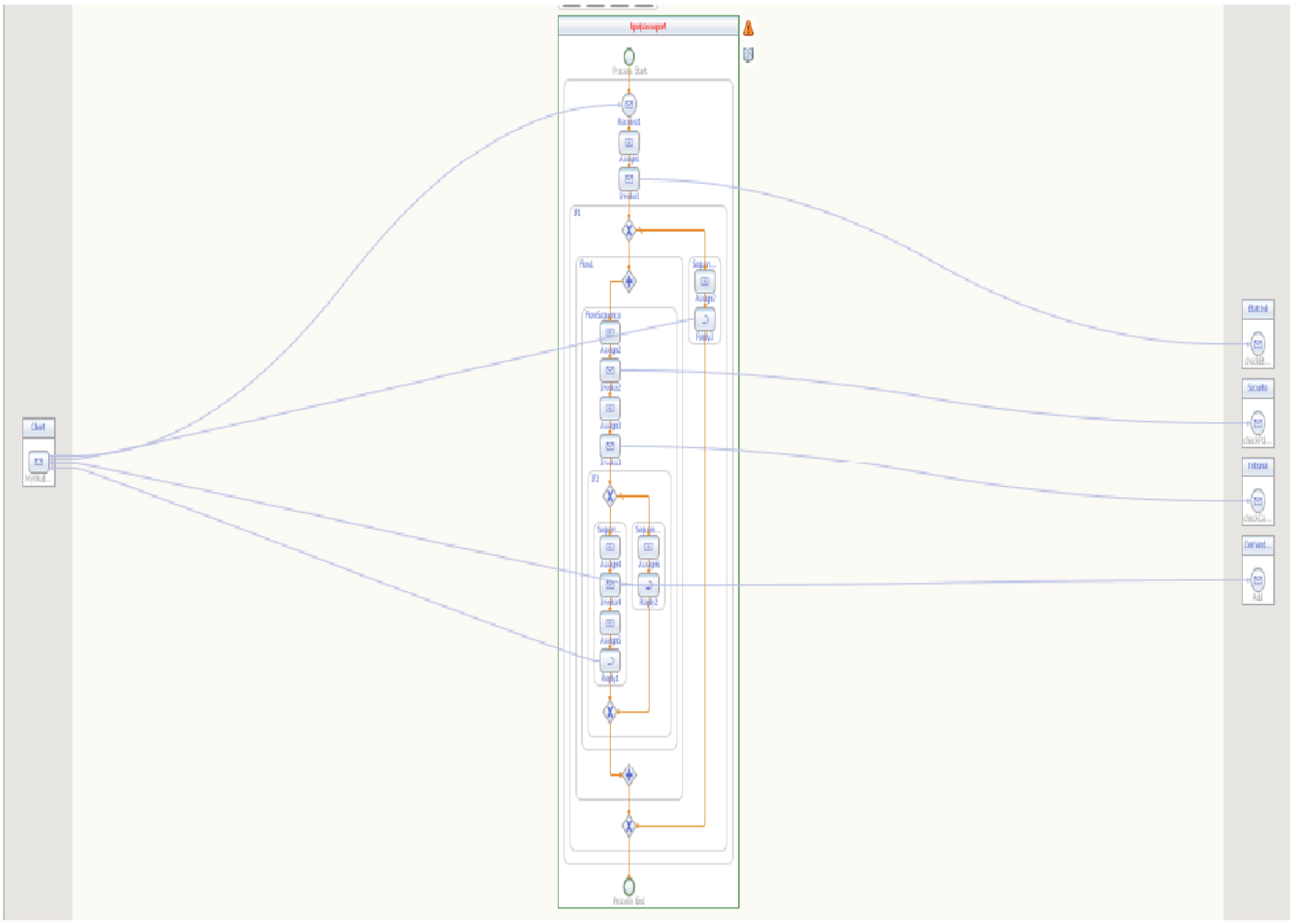

Figure [IV.15] – OpenESB BPEL Editor

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelop
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/ http://s
  <SOAP-ENV:Body>
    <m:MyWsdlOperationResponse xmlns:m="CompositeApp1" xmlns:msgns="http:/
      <part1>Votre demande est VALIDE Merci Beaucoup</part1>
    </m:MyWsdlOperationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

1

Figure [IV.16] – résultats d'une requête de composition invalide (fichier XML) .

3.2.4. Administrateur

Suite à une authentification de l'admin, il lui serait possible d'accéder à la page d'accueil et d'effectuer les différentes mises à jour (ajouter, modifier, supprimer) par chaque service comme le montre les figures suivantes.

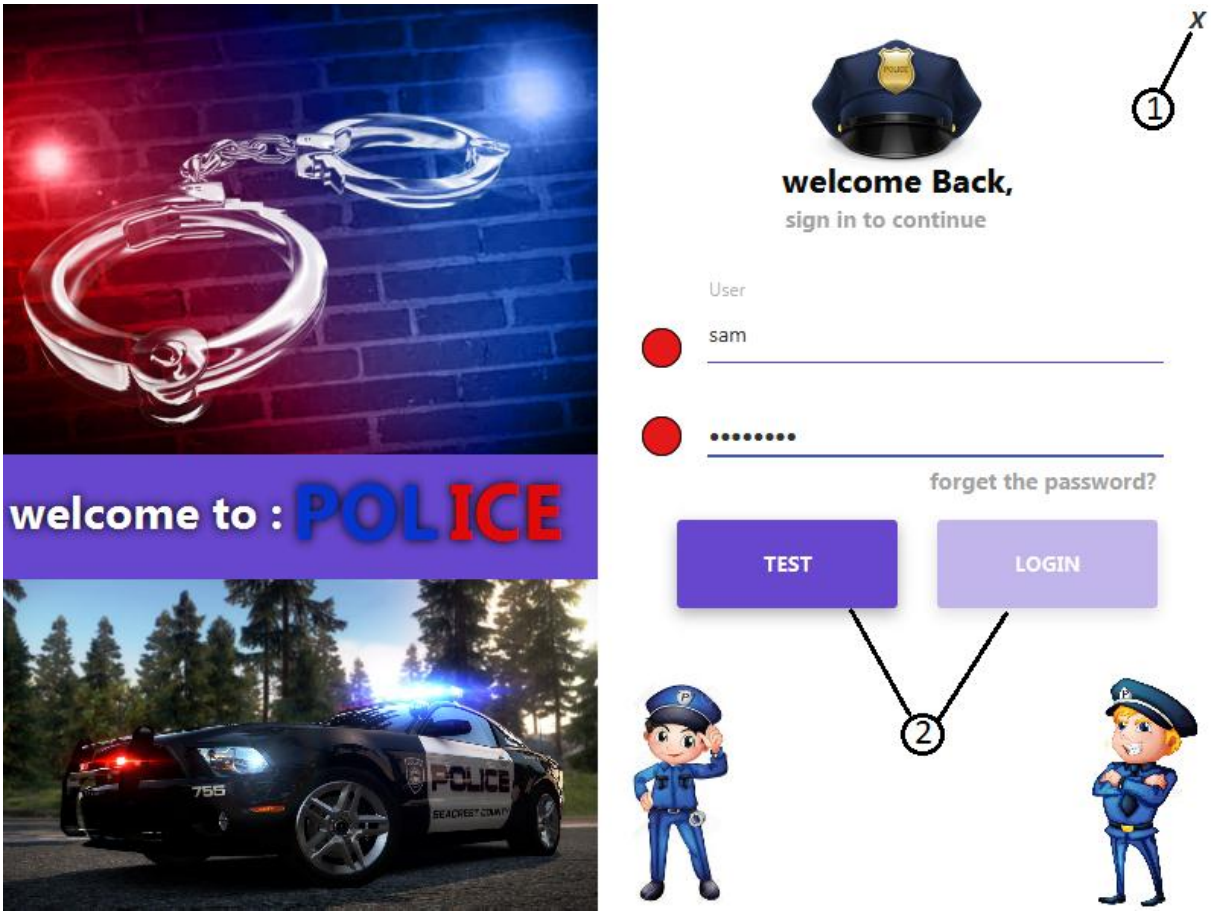

Figure [IV.17] – page d'administrateur de service police.

Par une clique(1) sur le service désiré, l'administrateur peut fermer la fenêtre. Dans le cas d'ajout d'un nouvel élément d'un service web à notre base de données par un fournisseur des services il faut d'abord authentifier l'identité de fournisseur par le bouton **TEST** (authentification).

Au début le Bouton **LOGIN** ne fonction pas(2), nécessairement l'administrateur saisie ses informations correctes pour pouvoir l'accès à la page suivant par le bouton LOGIN.

Figure [IV.18] – page d'administrateur de service état civil.

Si l'administrateur saisie ses informations correctes alors il peut accéder à ses fonctionnalités par le bouton **Se connecter** à la page suivant(1). Le bouton **Se connecter** devient fonctionnel.

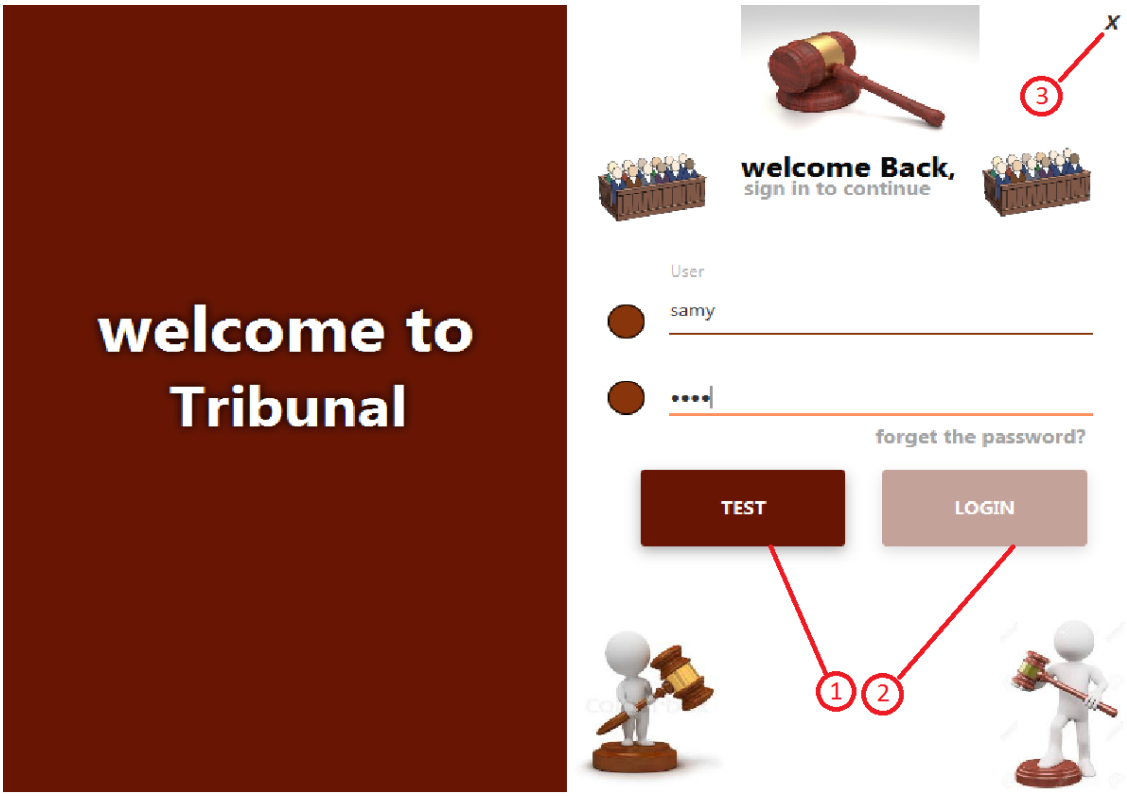

Figure [IV.19] – page d'administrateur de service tribunal.

Par une clique(1) sur le service désiré, l'administrateur peut accéder aux fonctionnalités de ce dernier Par une clique(2) sur le service désiré, le système va traiter et tester les informations de ce dernier. Par une clique(3) sur le service désiré, l'administrateur peut fermer la fenêtre.

3.2.5. Mise à jour (MAJ)

Par une clique(1) sur le service désiré, l'administrateur peut retourner à la page précédente (page d'authentification). Par une clique(2) sur le service désiré, l'administrateur peut fermer la fenêtre. Par une clique(3) sur le service désiré, l'administrateur effectué les différents MAJ.

Figure [IV .20] – page de MAJ de service police.

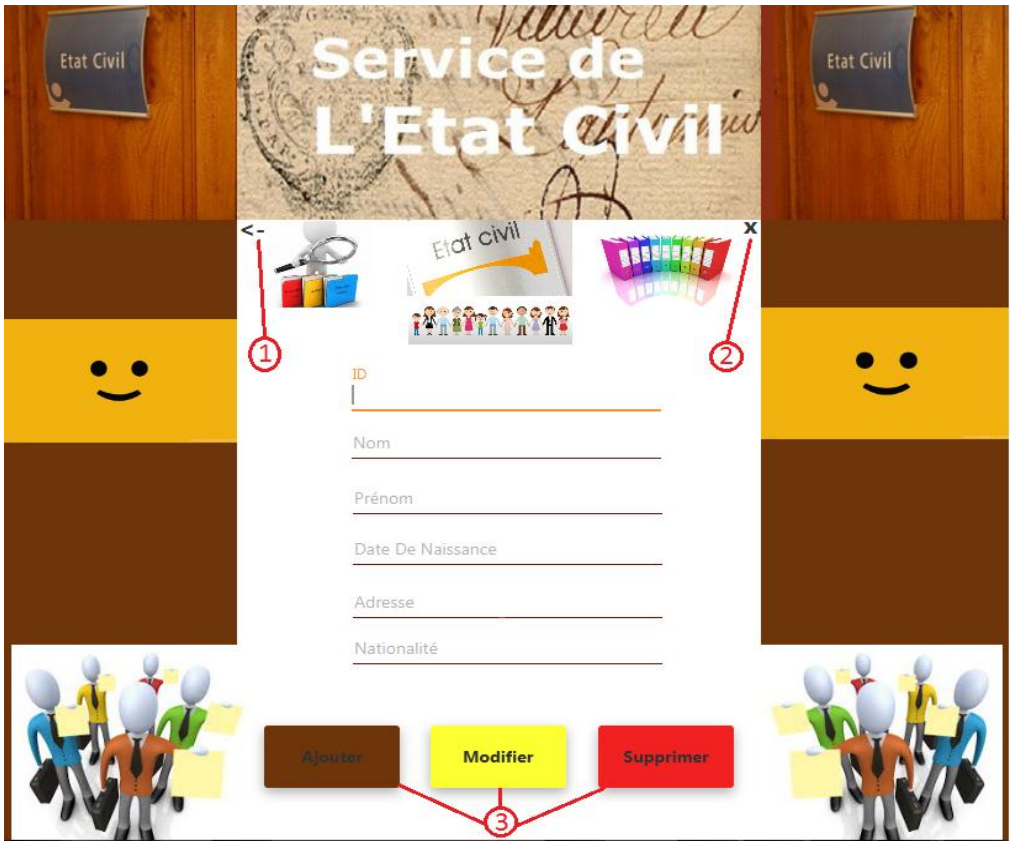

Figure [IV .21] – page de MAJ de service état civil.

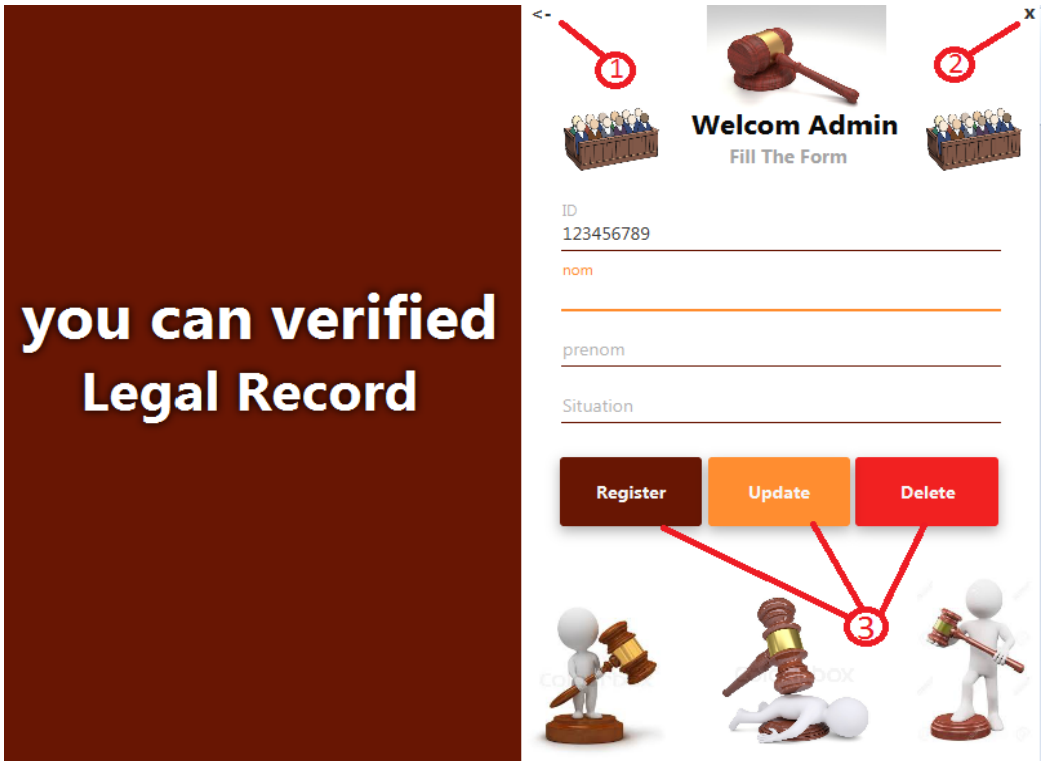

Figure [IV .22] –page de MAJ de service tribunal

4. Conclusion

Au cours de cette dernière étape de notre travail, nous avons réalisé un prototype d'un système de composition de services Web afin de valider notre proposition d'une approche de composition des services web. Nous avons donné une brève présentation des outils utilisés pour l'implémentation, et pour bien illustrer notre implémentation, nous avons fourni une démonstration des différentes fonctionnalités de notre application réalisée à travers des captures d'écran.

Conclusion générale

Conclusion générale

Conclusion Générale

Les services web sont des composants conceptuellement limités à des fonctionnalités relativement simples qui sont modélisées par un ensemble d'opérations. Leur architecture basée sur des standards Internet tels que XML et HTTP permet une communication facile et fluide entre les différents acteurs et clients. Cependant, un service web de par sa nature permet de réaliser des tâches simples, ce qui s'avère dans bien des cas insuffisant pour répondre aux besoins des utilisateurs. Et pour combler ce manque, la composition des services web devient une nécessité, afin de réaliser des tâches complexes.

Le but de la composition est de relier plusieurs services et les exécuter dans un ordre bien défini afin d'obtenir un résultat qui répond aux attentes de l'utilisateur. Les approches de composition de services web varient, on peut les classer en deux catégories **:** l'approche statique qui est adoptée principalement par le monde industriel et qui est basée essentiellement sur l'utilisation des processus métiers et les techniques d'orchestrations et de chorégraphies. L'approche dynamique, qui est sujet de recherche du monde académique et qui permet de composer dynamiquement les services web au moment de l'exécution afin d'atteindre les objectifs spécifiés par l'utilisateur et en tenant compte de ses préférences. Cette approche s'appuie sur les techniques de l'intelligence artificielle et l'ajout d'une couche sémantique à la description des services web (WSDL).

Dans ce travail nous avons proposé un algorithme de composition des services web. Notre algorithme de composition consiste à comparer les entrées (inputs)/sorties (outputs) des services web. Dans ce travail nous avons présenté la conception du système de composition proposé, notre conception est structuré en plusieurs phases : tout d'abord nous avons présenté les cas du système à la suite les diagrammes de séquences de chaque cas. Nous avons détaillé beaucoup plus le cas d'utilisation Composition des services web, cette partie contienne l'algorithme de composition proposé. A la fin ce travail, nous avons proposé une étude de cas comme une application à notre modèle de composition des services web. Nous motivons notre recherche du point de vue pratique. Nous décrivons une étude de cas sur la possibilité de demande une visa ou un passeport .Ce choix était motivé par sa simplicité et par son invocation à des services Web. Cet exemple permet en fait de tester les différentes phases développées dans notre approche de composition.

Bibliographie

Bibliographie

[1] : Appui Management de l'économie. Programme de coopération MEDA. Développement d'applicatifs métiers pour le MTP. "Mise en œuvre des services web avec JAX-WS".

[2] : Bouchekara Med Kacem , Ziani Soheyb, Mémoire de Master «La Découverte de services web à base De Dominance Probabiliste» Tlemcen, 23 Juin 2015.

[3] : Principes SOA de la conception de services. livre Thomas Erl, ISBN : 9780132361132, 2007.

[4] : Zerabi Soumaya, Mémoire de Magistère «Passage d'une architecture classique vers une architecture orientée services- Application à l'entreprise ENMTP » M'SILA, 2010.

[5] : Lequeux.J.L. Manager avec les ERP. Architecture Orientée Services (SOA). ISBN : 978-2-212-54094-9. Eyrolles, 3 eme Edition.2008.

[6] : Benamar Abdeladim et Ait Hamouda Mounir. Sélection de services web par l'optimisation d'essaim particulaire hybride. MASTER, 2013.

[7] :DEHANE Aicha Djihad et Melle KEBIR Zohra. Evaluation des techniques de codage d'ontologies sur les performances de la composition de services web. Thèse Master, 2012.

[8] : Hadjila FethAllah, Thèse de doctorat «Composition et interopération des services web sémantiques», Tlemcen, 2014.

[9] : UDDI Specification Technical Commitee. Universal Description, Discovery, and Integration of Business for the Web. Technical report, October 2001. http://www.uddi.org.

[10] : SOUKKARIEH Bouchra. Technique de l'internet et ses langages : vers un système d'information web restituant des services web sensibles au contexte. THÈSE de DOCTORAT, 2010.

[11] : BOUDJELABA Hakim. Sélection des web services sémantiques. Mémoire de Magister, 2012.

[12] : E. Christensen, F. Curbera, G. Meredith, and S.Weerawarana. Web Services DescriptionLanguage (WSDL) 1.1. W3c note, The World Wide Web Consortium (W3C), March 2001. http://www.w3.org/TR/wsdl.

[13]: W3C Working Group. XML Schema Part 2: Datatypes Second Edition. Technical report, W3C, October 2004. http://www.w3.org/TR/xmlschema-2/.

[14]: E. Christensen, F. Curbera, G. Meredith and S. Weerawarana, "Web services description language (wsdl) 1.1", Mar 2001, [Online]. Available: http://www.w3.org/TR/wsdl.

[15]: D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple object access protocol (SOAP) 1.1. Technical report, The World Wide Web Consortium (W3C), 2000. http://www.w3.org/TR/SOAP/

Bibliographie

[16]: R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1, 1999.

[17]: T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (xml). World Wide Web Journal, 2(4):27–66, 1997.

[18] : KHELLAF Radhia. Vérification de la des services compatibilité web pour une composition. Thèse de Magister, 2014.

[19] : Dardek Amina et Zouag Rafika. Interopérabilité des services web application sur la réservation de vol. Mémoire de Licence, 2010.

[20] : Michaël Mrissa et Lionel Médini. Du web de services au web des objets. Cours par Laboratoire d'Informatique en Image et Systèmes d'information, 2013.

[21] : Sofiane Chemaa. Une approche de composition de services web à l'aide des réseaux de pétri orientés objet. Thèse de Doctorat, 2014.

[22] : Faycal BACHTARZI. Une approche de composition des services web basée transformation de graphes. Une thèse de doctorat, 2014.

[23] : http://fr.wikipedia.org/wiki/Service_Web Avantages, consulté le 14/2/2017.

[24] : David Chappell, et Tyler JEWELL, Java Web Service, édition O'Reilly, Mars 2002.

[25] : https://www.w3.org/TR/xmlschema-1 consulté le 14/02/2017.

[26]: Zahir Tari, Peter Bertok, Anshuman Mukherjee, "Verification of communication protocols in web services: model-checking service compositions, Wiley series on parallel and distributed computing, p58.

[27]: Dustdar S, Schreiner W. 'A survey on web services composition', Int. J. Web and Grid Services, 2005, Vol. 1, No. 1, p 2.

[28] : Mohamad El Falou, « Contributions à la composition dynamique de services fondée sur des techniques de planification et diagnostic multiagents », thèse de doctorat, université de Caen, 2010.

[29] : Abderrahmane Maaradji, Hakim Hacid, Johann Daigremont and Noel Crespi, « Composition des services Web Basée sur les Réseau Sociaux », Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances, Tunisie, 2010.

Bibliographie

[30] : Rao J and Su X. 'A survey of automated Web service composition methods', Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, California, USA: Springer, 2004.

[31] : M. Fluegge, I. J. G. Santos, N. P. Tizzo, and E. R. M. Madeira, "Challenges and techniques on the road to dynamically compose Web services,", Proceedings of the 6th international conference on Web engineering, (New York, USA), ACM, 2006.

[32] : Abrehet Mohammed Omer, "A framework for Automatic Web Service Composition based on service dependency analysis", thèse de doctorat, Technical University of Dresden Germany, Avril 2011.

[33] : Tuo Zhang, « Vers une médiation de composition dynamique de Services Web dans des environnements ubiquitaires », Université Paris-Nord – Paris13, 2014.

[34]: Shirin Sohrabi, "CUSTOMIZING THE COMPOSITION OF WEB SERVICES AND BEYOND", thèse de doctorat, université de Toronto, 2012.

[35] : D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. In ISWC'03 ,2003.

[36] : E. Sirin, and B. Parsia. "Planning for semantic web services". In Proc of Semantic Web Services Workshop at 3rd International Semantic Web Conference, 2004.

[37] : Sheila Mcilraith, "Adapting golog for composition of semantic Web services", in: International Conference on Knowledge Representation and Reasoning (KR'02), 2002, p 482–493.

[38] : B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid,. « Composing web services on the semantic web ». The VLDB Journal, 12: 2003, pp333–351.

[39] : WALDINGER, Richard. "Web agents cooperating deductively. In : Formal Approaches to Agent-Based Systems". Springer Berlin Heidelberg, 2000. p. 250-262.

[40] : I.Muller, and R. Kowalczik. Service composition through agent-based coalition formation. In Proc of WWW Service composition with Semantic Web Services, Compiegne, France, Septembre 2005, pp 34- 43.

[41] : Furkh Zeshan and Radziah Mohamad, "Semantic Web Service Composition Approaches: Overview and Limitations," International Journal on New Computer Architectures and Their Applications, 2011, pp640-651.

[42] : Lécué, F. "Optimizing QoS-aware semantic Web service composition". In Proceedings of the 8th International Semantic Web Conference (ISWC), 2009. pp 375–391.

Bibliographie

[43] : Rabah Imache, thèse doctorat «Un modèle d'évaluation et de contrôle de l'agilité des systèmes d'information d'entreprise», Boumerdes, 16/04/2012.

[44] : Jihed Touzi, thèse doctorat « Aide à la conception de Système d'Information Collaboratif support de l'interopérabilité des entreprises » Toulouse, 2007.

[45] : G. Raymond : SOA : architecture logique, principes, structures et bonnes pratiques. Livre blanc, avril 2011.

[46] : Iheb Abdellatif, Mémoire de Master «Vers une démarche d'aide à la décision pour l'indentification des services d'une architecture orientée services» Québec, 2011.

[47] : Khouloud Boukadi, thèse doctorat « Coopération interentreprises à la demande : Une approche flexible à base de services adaptables » Saint-Étienne, 2009.

[48] : Ahlem Zayati, thèse doctorat « Mise en oeuvres des architectures orientées services pour les systèmes d'information industriels» Lyon, 2012.

[49] : Claude delannoy, «Programmer en java»,Edition EYROLLES 2008

[50] : http://dev.mysql.com/doc/refman/5.5/en/

[51] : Dardek Amina et Zouag Rafika. Interopérabilité des services web application sur la réservation de vol. Mémoire de Licence, 2010.

[52] :http://doc.ubuntu-fr.org/glassfish

[53] : Méthodologie UML,Cours du cycle B du cnam.doc, 2000-2001.

[54]: Doc: 770-001: OpenESB Standalone Enterprise Edition Installation