

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique



MÉMOIRE DE FIN D'ÉTUDE

En vue de l'obtention du diplôme de master professionnel
En Informatique

Option : *Génie logiciel*

Thème

Conception et réalisation d'une application
à base des services web
d'une agence de voyage en ligne

Présenté par : BOULAHIA NADIA

Soutenu devant le jury composé de :

Président	Mme SABRI SALIMA	U. A/Mira Béjaïa.
Rapporteur	Mme EL BOUHISSI HOUDA	U. A/Mira Béjaïa.
Examineur	Mme HALFOUNE NADIA	U. A/Mira Béjaïa.
Examineur	Mme BATTAT NADIA	U. A/Mira Béjaïa.

Juin 2017

Remerciements

Louange à Dieu qui ma donné la force, le courage, et l'espoir nécessaire Pour accomplir ce travail et surmonter l'ensemble des difficultés.

Je tiens à remercier vivement :

Mon encadreur Dr.BRAHAMI Née EL BOUHISSI H pour ses conseils et son suivi durant la réalisation de mon projet.

Les membres du jury pour avoir accepté d'examiner ce mémoire.

J'adresse mes remerciements aux personnes qui m'ont aidé et encouragé dans la réalisation de ce mémoire.

Dédicaces

Je dédie ce travail

A mes chers parents

A mes frères et mes sœurs

A tous mes amis

A tous ceux qui, par un mot, m'ont donné la force de continuer

BOULAHIA NADIA

Résumé

Les services Web sont des programmes autonomes, auto- Descriptifs, des applications modulaires qui peuvent être publiées, localisées et invoquées à travers le Web.

Notre travail consiste à mettre en place une architecture orientée service à travers la conception et la réalisation une application de gestion d'une agence de voyage en ligne.

Les différents services sont développés dans divers langages de programmation et diverses plates-formes.

Ainsi, pour permettre à un client d'organiser ses voyages de façon conviviale nous avons construit un portail en JSP qui permet un traitement d'invocation des différents services distants en utilisant les informations récupérées des fichiers WSDL des services existants.

L'objectif est de donner l'impression à l'utilisateur qu'il est sur un site web classique et que tout lui vient du serveur, alors qu'il s'agit d'un assemblage et d'une composition de services web distant et fournis par différents fournisseurs.

Mots-clés : Service web, WSDL, Invocation, JSP.

Abstract

Web services are self-contained, self-descriptive, modular applications that can be published, localized, and invoked across the web.

Our job is to set up a service-oriented architecture through the design and realization of a management application of an online travel agency.

The different services are developed in various programming languages and platforms.

Thus, in order to allow a customer to organize his trips in a user-friendly way we have built a portal in JSP that allows invocation processing of the various remote services using the information retrieved from the WSDL files of the existing services.

The goal is to give the impression to the user that it is on a classic web site and that everything comes from the server, whereas it is a assembly and a composition of remote web services And provided by different suppliers.

Keywords : Web service, WSDL, Invocation, JSP.

Table des matières

Table des figures	III
Liste des tableaux	V
Introduction générale	1
1 les services web	3
1.1 Introduction	3
1.2 Définition	3
1.3 Architecture orientée service	5
1.3.1 Définitions	5
1.3.2 SOA et les services web	6
1.4 Besoin d'utilisation des services web	7
1.5 Technologies des web services	8
1.5.1 Extensible Markup Language (XML)	8
1.5.2 L'architecture XML-RPC	9
1.5.3 L'architecture REST	10
1.5.4 Le protocole SOAP	11
1.5.5 Le débat SOAP vs. REST	13
1.5.6 Description des services web avec WSDL	15
1.5.7 Découverte des services web avec UDDI	18
1.6 Modèles d'interaction des services web	22
1.6.1 Scénario générale de fonctionnement des services web	22
1.6.2 Étape d'exécution des services web	23
1.7 Conclusion	24
2 Les plateformes de développement des services web	25
2.1 Introduction	25
2.2 La plateforme.NET	25
2.2.1 Le framework.NET	27
2.2.2 Common Language Runtime (CLR)	27
2.2.3 ASP.NET	28
2.2.4 Visual studio.net	29
2.3 La plateforme J2EE	30
2.3.1 L'architecture J2EE	30

2.3.2	Services Web et J2EE	31
2.3.3	les servlets	32
2.3.4	Java Server Pages :	32
2.3.5	Les Enterprise JavaBeans (EJB) :	33
2.3.6	JAX-WS et JAX-RS :	34
2.4	La différence entre J2EE et .NET	35
2.5	Autres plateformes de développement des services web	39
2.5.1	Apache Axis2	39
2.5.2	Sun ONE Developer Studio de Sun :	41
2.6	Conclusion	43
3	Modélisation	44
3.1	Introduction	44
3.2	Modélisation des services	44
3.2.1	Diagrammes de cas d'utilisations	44
3.2.2	diagrammes de séquence	49
3.2.3	Diagramme de classes	52
3.3	Conclusion	53
4	La réalisation	54
4.1	Introduction	54
4.2	Architecture globale du system :	54
4.3	Les outils utilisés dans le développement de différentes services	55
4.4	Présentation des services web	58
4.5	Description de l'application(présentation des différentes interfaces de l'application)	59
	Conclusion générale et Perspectives	67
	Références	69

Table des figures

1.1	Description des services web	4
1.2	Les différents composants d'un web service	4
1.3	Les architectures orientées services	6
1.4	L'architecture XML-RPC	10
1.5	L'architecture REST	11
1.6	Modèle d'échange de message SOAP	12
1.7	Structure d'un message SOAP	13
1.8	Web service RESTful	15
1.9	Web service SOAP	15
1.10	Structure de wsdl 1.1 et wsdl 2.0	16
1.11	Relations entre les structures de données UDDI	20
1.12	Utilisation de UDDI	21
1.13	Modèle d'interactions des services web	22
2.1	L'architecture .NET de Microsoft	26
2.2	L'architecture du Framework.net	27
2.3	L'architecture J2EE	31
2.4	Support web services .NET	35
2.5	Support J2EE web services	36
2.6	Pile des services de Sun ONE	42
2.7	Architecture de Sun ONE	42
3.1	diagramme de cas d'utilisation de la compagnie aérienne	45
3.2	diagramme de cas d'utilisation de l'hôtel	46
3.3	diagramme de cas d'utilisation de la voiture	47
3.4	diagramme de cas d'utilisation Global de l'agence de voyage	48
3.5	diagramme de sequence pour la recherche du vol	49
3.6	diagramme de sequence pour la réservation du vol	50
3.7	diagramme de sequence pour l'annulation d'une réservation du vol	51
3.8	diagramme de classes	52
4.1	Architecture globale du système	55
4.2	Page d'accueil de l'agence de voyage	59
4.3	La réservation d'un vol	60
4.4	Résultat d'une opération de recherche de vol	61

4.5	formulaire de réservation d'un vol	62
4.6	La réservation d'une voiture	63
4.7	Résultat d'une opération de recherche de voiture	64
4.8	formulaire de réservation d'une voiture	65
4.9	La réservation dans un hôtel	66

Liste des tableaux

1.1 Objets dans WSDL 1.1 / WSDL 2.0	18
---	----

Introduction générale

Les caractères révolutionnaires (universalité et ubiquité) d'Internet et du Web constituent une étape supplémentaire dans la chaîne évolutive qui a déjà mené, en quelques décennies, l'application monolithique du main-frame, en passant par le client-serveur et l'architecture à trois niveaux, aux assemblages d'objets répartis (CORBA¹, DCOM² et EJB³) et à leurs serveurs d'application. Mais alors que les maillons de cette évolution s'appuient sur des protocoles, des langages et des interfaces progressivement plus riches et plus raffinées, ouverts ou jalousement propriétaires, le Web impose des moyens de communication autrement plus rustiques : débit comparativement moins haut, temps de latence, défaillances, protocole HTTP⁴ d'une sobriété dégraissante après certains excès de complexité des protocoles RMI⁵ ou DCOM.

A la fin des années 1980, l'industrie du logiciel était agitée par le débat entre les interfaces graphiques riches mais consommatrices de ressources, comme dans Windows, et les pages HTML⁶, peut-être simplistes, mais accessibles à partir d'un navigateur Web léger et universel. De même, l'heure est aujourd'hui à la recherche d'un compromis satisfaisant entre les besoins complexes des applications d'entreprise et les contraintes de simplicité du Web, garantes de son universalité. Est-il finalement possible de conserver le meilleur des évolutions précédentes et de le mettre en œuvre dans le nouvel environnement ? Les premiers déploiements réussis d'applications fondées sur les services Web permettent de répondre par l'affirmative.

Les services web apparaissent comme une nouvelle possibilité exploitant plus directement et plus simplement les possibilités offertes par l'infrastructure Internet. Par rapport aux technologies CORBA/COM⁷/DCOM et Java RMI⁸, elles libèrent la contrainte d'une API⁹ unique pour l'invocation des services de distribution par les applications désirant inter-opérer et reposent sur des invocations de service par transmission de messages dont le format est auto-

-
1. Common Object Request Broker Architecture.
 2. Distributed Component Object Model
 3. Enterprise JavaBeans
 4. HyperText Transfer Protocol
 5. Remote method invocation
 6. HyperText Markup Language
 7. Component Object Model
 8. Remote Method Invocation
 9. Application Programming Interface

porteur en s'appuyant sur la technologie XML¹⁰. Cette caractéristique et l'utilisation directe des protocoles Internet la suggèrent comme une technologie de référence particulièrement indiquée pour l'interopérabilité de systèmes hétérogènes au sein de l'entreprise et plus encore lorsqu'ils sont situés dans des entreprises différentes.

Notre mémoire s'articulera autour de quatre chapitres :

Dans le **premier chapitre** nous ferons une revue de la littérature sur le concept de service web ainsi que les différentes technologies des services web.

Le **deuxième chapitre** est consacré à la description de différentes plates-formes de développement des services web, avec une comparaison entre les deux plateformes Microsoft.net et java/J2EE¹¹.

Le **troisième chapitre** sera consacré à l'implémentation, nous avons pris comme étude de cas la gestion d'une agence de voyage à base de services web.

En fin la réalisation du système, sera mentionnée dans le **quatrième chapitre**.

10. Extensible Markup Language

11. Java 2 Enterprise Edition

les services web

1.1 Introduction

L'accès aux systèmes d'information s'appuie aujourd'hui de plus en plus sur des technologies internet. Les efforts de standardisation dans ce contexte ont accentué l'engouement des personnes et des organisations (aussi bien académiques, qu'industrielles, commerciales, ou institutionnelles) pour l'utilisation de l'Internet et ont permis l'émergence des services web comme support de développement des applications accessibles par Internet. Ainsi, les technologies associées aux services Web sont devenues incontournables pour le développement d'applications interagissant les unes avec les autres par le biais de l'Internet.

1.2 Définition

Il existe de nombreuses définitions d'un web service. Citons la définition de l'organisme de normalisation du World Wide Web Consortium (W3C) en anglais :

"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" .[1]

Après traduction, on obtient la définition suivante :

"Un Web Service est un système logiciel conçu pour permettre l'interopérabilité entre les machines sur un réseau. Il possède une interface qui décrit, dans un format normalisé, le moyen de communiquer avec la machine (par exemple : WSDL¹). D'autres systèmes inter-

1. Web Services Description Language

agissent avec les web services, conformément à l'interface, en utilisant les messages SOAP² envoyés par le protocole HTTP et écrits en XML, en liaison avec d'autres normes standards du Web."[14]

Les web services créent une architecture de type client/serveur dans laquelle des clients (ordinateur de bureau, ordinateur portable, téléphone portable...) utilisent, via internet, des procédures qui sont stockées sur un serveur d'applications.

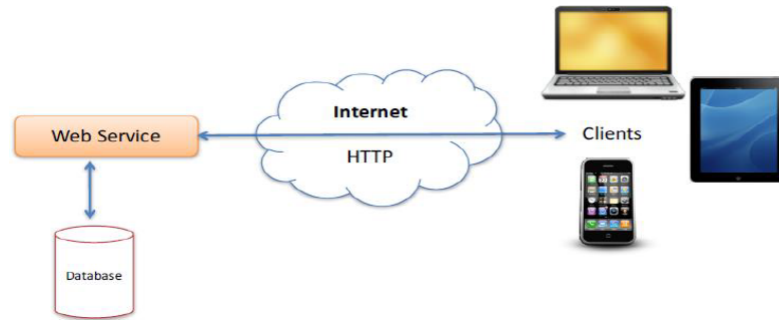


FIGURE 1.1 – Description des services web

De manière schématique, trois composants (Figure 1.2) sont nécessaires dans un web service :

- Un protocole pour décrire le service (idéalement il doit lister les méthodes disponibles et leurs arguments) ;
- Un protocole décrivant la composition des messages ;
- Un protocole de transport pour faire circuler les informations sur Internet.

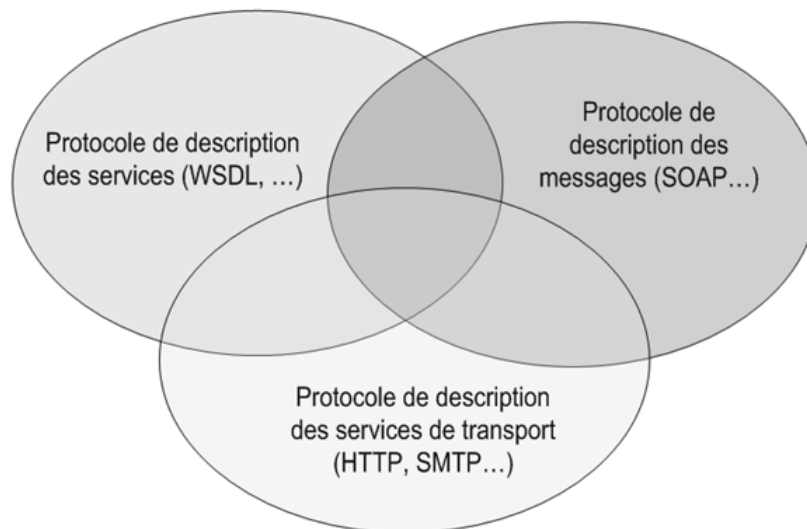


FIGURE 1.2 – Les différents composants d'un web service

2. Simple Object Access Protocol

1.3 Architecture orientée service

L'architecture orientée services est une réponse très efficace aux problématiques que rencontrent les entreprises en termes de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différents systèmes qui implémentent leurs systèmes d'information.

1.3.1 Définitions

Pour ce concept, les définitions sont nombreuses et nous retiendrons les suivantes :

« L'architecture orientée services est un paradigme permettant d'organiser et d'utiliser des savoir-faire distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoir-faire pour produire le résultat désiré avec des pré-conditions et des buts mesurables » [17]

« L'architecture orientée services permet l'intégration d'applications et de ressources de manière flexible en : représentant chaque application ou ressource sous la forme d'un service exposant une interface standardisée, permettant à un service d'échanger des informations structurées (messages, documents, objets métier), coordonnant et en organisant les services afin d'assurer qu'ils puissent être invoqués, utilisés et changés efficacement » [12].

De ces deux définitions, nous ressortons une idée principale, à savoir la notion d'organisation des services offerts par des fournisseurs. Malgré le manque de spécification officielle pour définir une architecture orientée services, trois rôles clés sont communément identifiés :

- Le rôle de producteur de services,
- Le rôle de répertoire de services,
- Le rôle de consommateur de services.

Le producteur a pour fonction de déployer un service sur un serveur et de générer une description de ce service. Cette dernière précise à la fois les opérations disponibles et leur mode d'invocation. Cette description est publiée dans un répertoire de services, aussi appelé annuaire. Les consommateurs peuvent découvrir les services disponibles et obtenir leur description en lançant une recherche sur un répertoire. Ils peuvent ensuite utiliser la description du service ainsi obtenue pour établir une connexion avec le fournisseur et invoquer les opérations du service souhaité. On peut avoir donc le schéma récapitulatif suivant :

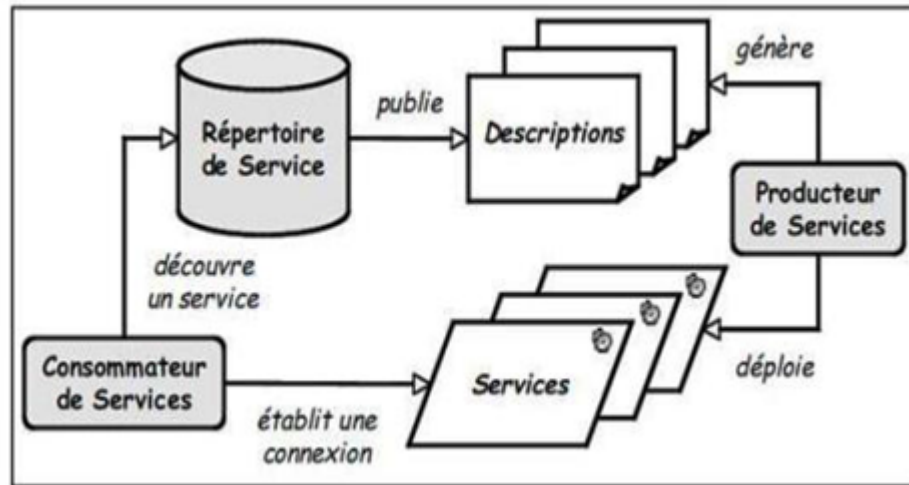


FIGURE 1.3 – Les architectures orientées services

Les caractéristiques principales d'une architecture orientée services sont le couplage faible entre les services, l'indépendance par rapport aux aspects technologiques et la mise à l'échelle. La propriété de couplage faible implique qu'un service n'appelle pas directement un autre service. Les interactions sont gérées par une fonction d'orchestration [11]. La réutilisation d'un service est alors plus facile, du fait qu'il n'est pas directement lié aux autres services de l'architecture dans laquelle il évolue. L'indépendance par rapport aux aspects technologiques est quant à elle, obtenue grâce aux contrats d'utilisation associés à chaque service. Ces contrats sont indépendants de la plateforme technique utilisée par le fournisseur du service. La mise à l'échelle est rendue possible grâce à la découverte et à l'invocation de nouveaux services lors de l'exécution.

1.3.2 SOA et les services web

Les services Web sont utilisés pour créer des applications qui peuvent envoyer / recevoir des messages à l'aide de SOAP sur HTTP. Un service Web est un ensemble publicitaire de fonctionnalités offertes sur le Web. SOA (Service Oriented Architecture) est un ensemble de concepts architecturaux utilisés pour le développement et l'intégration des services. Les services Web peuvent être utilisés pour implémenter SOA. Mais ce n'est qu'une méthode unique pour réaliser des applications basées sur SOA.

Il existe des différences clés entre les services Web et SOA. Les services Web définissent une technologie Web qui peut être utilisée pour créer des applications qui peuvent envoyer / recevoir des messages à l'aide de SOAP sur HTTP. Cependant, SOA est un modèle architectural pour la mise en œuvre d'applications basées sur les services couplés. Les services Web peuvent être utilisés pour implémenter des applications SOA. Même si l'approche de service

Web de SOA est devenue très populaire, ce n'est qu'une méthode unique de mise en œuvre de SOA. SOA peut être implémenté à l'aide de toute autre technologie basée sur le service (p. Ex. CORBA et REST³).

L'architecture orientée services est une réponse très efficace aux problématiques que rencontrent les entreprises en termes de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différents systèmes qui implémentent leurs systèmes d'information. Les architectures SOA ont été popularisées avec l'apparition de standards comme les Services Web dans l'e-commerce (commerce électronique) (B2B⁴, inter-entreprise, ou B2C⁵, d'entreprise à consommateur), basés sur des plates-formes comme J2EE ou .NET et la déclinaison libre Mono de cette dernière. Elles mettent en pratique une partie des principes d'urbanisation. Au sein de l'architecture orientée services, on distingue les notions d'annuaire, de bus, de contrat et de service, ce dernier étant le noyau et le point central d'une architecture orientée services. La déclinaison ou plus précisément l'implémentation de la SOA avec des Web Service est la WSOA (Web Service Oriented Architecture).

1.4 Besoin d'utilisation des services web

L'idée fondamentale derrière les services web est de morceler les applications et les processus en morceaux réutilisables appelés » Service » de sorte que chacun de ces segments effectue une tâche distincte. Ces services peuvent alors servir à l'intérieur et à l'extérieur de l'entreprise (exemple de l'université), facilitant l'interopérabilité entre tous ces services. De par leur nature, les services web ont les caractéristiques suivantes.[2]

Interopérabilité

L'interopérabilité correspond à la capacité, plus ou moins grande, qu'ont les applications, quelque soit leur origine, appartenance, diversité, à coopérer entre elles en terme d'échange d'informations et d'interaction. L'interopérabilité présente la caractéristique principale des services web. En effet, quelque soit la plateforme 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 et composition de services

Les services web constituent un support d'échange de documents structurés qui traversent

3. Representational State Transfer
4. Business to Business
5. Business to Consumer

les contrôles d'accès dans un environnement hétérogène (à travers le protocole http). La collaboration entre différentes applications afin d'échanger des documents se fait d'une manière directe entre objets ce qui fait qu'il n'existe pas de dépendance entre les différents services. Ces derniers sont désormais considérés comme un ensemble de fonctions indépendantes mais communiquant entre eux.

Organisation autour des protocoles standards

Le grand succès de cette technologie passe par le fait qu'elle soit construite sur les protocoles standards ouverts. Ce qui a assuré une rapide évolutivité et une maîtrise parfaite de la technologie.

1.5 Technologies des web services

Les services Web reprennent la plupart des idées et des principes du Web (HTTP, XML), et les appliquent à des interactions entre machines. Comme pour le World Wide Web, les services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les services Web sont HTTP, WSDL, REST, XML-RPC, SOAP et UDDI.

1.5.1 Extensible Markup Language (XML)

L'Extensible Markup Language (XML « langage de balisage extensible » en français) est un métalangage informatique de balisage générique qui dérive du SGML⁶. Cette syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML⁷, XSLT⁸, ... Elle est reconnaissable par son usage des chevrons (<, >) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche...) entre systèmes d'informations hétérogènes (interopérabilité).

XML est un standard promulgué par le W3C, l'organisme chargé de standardiser les évolutions du Web. On retrouve dans XML une généralisation des idées contenues dans HTML et SGML. XML permet de définir des balises et de leur associer une interprétation. Dans HTML, on n'utilise les balises que pour décrire l'aspect graphique que doit revêtir la page dans le navigateur Web. Dans XML, les balises permettent d'associer toutes sortes d'informations au fil du texte.

La norme XML comporte deux parties : XML à proprement parler et les DTD (Document

6. Standard Generalized Markup Language

7. Extensible HyperText Markup Language

8. eXtensible Stylesheet Language Transformations

Type Définition) qui définissent les balises qui sont utilisées dans une famille de documents XML. XML a été conçu pour des documents arbitrairement complexes, tout en s'appuyant sur cinq grands principes simples et clairs :

- Lisibilité à la fois par les machines et par les utilisateurs ;
- Définition sans ambiguïté du contenu d'un document ;
- Définition sans ambiguïté de la structure d'un document ;
- Séparation entre documents et relations entre documents ;
- Séparation entre structure du document et présentation du document.

Messages XML entre applications et services Web

Pour les services Web, on utilise systématiquement XML avec les Namespaces et la spécification XML Schéma, tous deux indispensables pour exprimer les structures des données habituellement complexes figurant dans les messages échangés.

Dans le mécanisme de transport entre services Web, les requêtes, leurs résultats et les erreurs éventuelles résultant de leur invocation sont tous écrits en XML. Ce sont donc des documents émis par le client vers le service Web et traités sur le serveur en réponse aux demandes.

Le choix de documents XML comme format pour les données circulant entre applications clientes et services Web ou entre services Web présente des avantages sur les middleware de la génération précédente tels que CORBA, RMI ou DCOM. Il n'y a pas de programmation à proprement parler et l'on est loin des mécanismes de compilation particuliers que le recours à des RPC1 ou à de tels middleware imposent aux programmeurs.

De plus, en s'appuyant sur HTTP, on ne se heurte pas aux problèmes de pare-feu ou de configuration de réseau IP⁹ qui rendent parfois difficile le déploiement d'applications à objets répartis au-delà du périmètre du réseau d'entreprise.

1.5.2 L'architecture XML-RPC

Cette architecture XML-RPC est apparue en 1998. Comme son nom l'indique, elle est composée de XML et de RPC. RPC (qui signifie en anglais Remote Procedure Call) est un protocole réseau. Son fonctionnement consiste, à partir d'un client, à faire appel à des procédures qui sont hébergées sur un serveur distant. Pour qu'une telle technologie fonctionne il faut :

- que le client possède une copie (nommée Stub) de l'objet distant ;
- que le client appelle les méthodes de la copie ;
- que le serveur possède un objet nommé Skeleton qui assure la connexion avec l'objet réellement appelé sur le serveur.

9. Internet Protocol

L'avantage principal de cette technologie est de permettre qu'un appel à une procédure distante soit identique à celui d'une procédure locale. Ainsi, pour le client toute la complexité de l'appel est cachée.[14]

Pour qu'un appel RPC fonctionne il faut envoyer au serveur :

- le nom de la méthode à exécuter ;
- les paramètres de la méthode.

Cette architecture est apparue lorsque l'idée d'appeler des services sur le web s'est fortement développée. Son fonctionnement est simple. La couche transport est assurée par le langage HTTP. En ce qui concerne l'envoi des requêtes et des réponses, c'est la méthode POST de HTTP qui gère les envois. Le langage de description des messages est le langage XML.

Le schéma ci-dessous montre le fonctionnement de cette architecture XML-RPC (Figure 1.5). Dans un premier temps, un client envoie un document XML par la méthode POST du protocole HTTP. Le serveur le reçoit puis procède à un traitement. Lorsqu'il a terminé, il envoie à son tour la réponse sous la forme d'un document XML qui est transféré par le protocole HTTP.

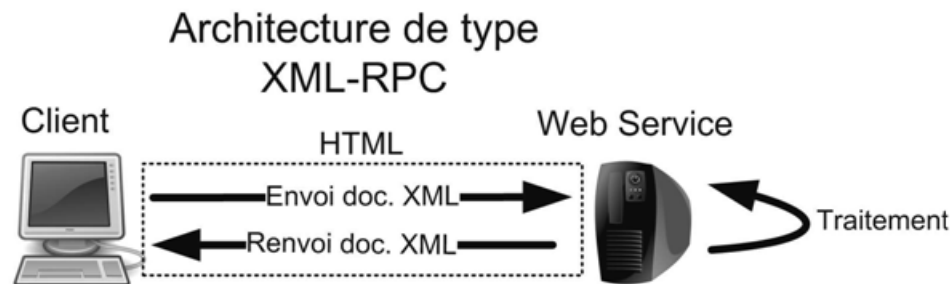


FIGURE 1.4 – L'architecture XML-RPC

1.5.3 L'architecture REST

REST a été définie dans la thèse écrite par Roy Fielding en 2000. REST est une architecture, ce n'est surtout un protocole comme par exemple SOAP. Elle n'a pas de spécification W3C. beaucoup de web service utilisent cette architecture tels que :eBay, Amazone.com, Google ou Yahoo.

La particularité d'une architecture REST est que le web service ne conserve pas l'état courant : il est sans état (Stateless). Le client doit toujours envoyer toutes les informations au web service ; ainsi, le serveur qui héberge le web service ne consomme pas trop de mémoire pour l'utilisation de celui-ci. Il utilise le protocole HTTP pour communiquer entre le client et le serveur grâce aux méthodes : GET ; POST ; PUT ; DELETE.

L'architecture REST oblige à ce que toutes les ressources soient identifier par une URI(Uniform Ressource Identifier). UNE URI permet d'identifier une ressource sur un réseau de façon

unique ; par exemple, une URL est une URI.

Les informations qui sont échangé entre le client et le serveur peuvent être de formes multiples (XML ;JSON ¹⁰,...). [14]



FIGURE 1.5 – L’architecture REST

REST est une architecture où le web service est sans état ; c’est-à-dire qu’il est très facile à utiliser car il tire profit des méthodes de HTTP. Aucune technologie, autre qu’un serveur web ; n’est nécessaire. N’importe quel langage de programmation autorisant la manipulation des sockets peut facilement permettre l’utilisation d’un web service REST.

Chaque service REST est identifié par une URL et les informations à passer au service sont transmises par les méthodes de HTTP.

Le protocole REST ne donne aucune restriction sur le format des réponses. Une réponse peut être un simple fichier XML ou encore un fichier JSON. Le client peut demander un type particulier de réponse en utilisant les attributs des requêtes. Par exemple en HTTP, l’attribut `accept =text/xml` spécifier qu’une réponse est attendue sous la forme d’un fichier XML.

1.5.4 Le protocole SOAP

Simple Object Access Protocole (SOAP) est un protocole d’invocation de méthodes sur des services distants. Basé sur XML, SOAP a pour principal objectif d’assurer la communication entre machines. Le protocole permet d’appeler une méthode RPC et d’envoyer des messages aux machines distantes via HTTP. Ce protocole est très bien adapté à l’utilisation des services Web, car il permet de fournir au client une grande quantité d’informations récupérées sur un réseau de serveurs tiers.

10. JavaScript Object Notation

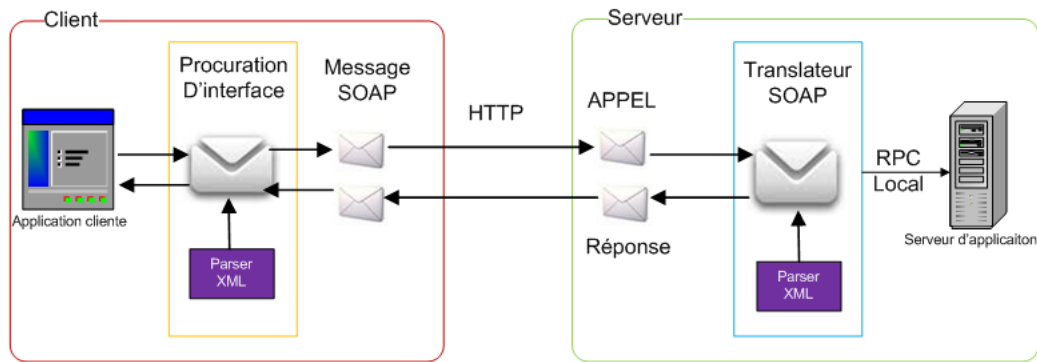


FIGURE 1.6 – Modèle d'échange de message SOAP

SOAP est bien plus populaire et utilisé que XML-RPC. C'est une recommandation du W3C. D'après cette recommandation, SOAP est destiné à être un protocole léger dont le but est d'échanger des informations structurées dans un environnement décentralisé et distribué. Une des volontés du W3C vis-à-vis de SOAP est de ne pas réinventer une nouvelle technologie. SOAP a été construit pour pouvoir être aisément porté sur toutes les plates-formes et les technologies existantes.

SOAP est une spécification de communication entre services Web par échange de messages en XML au travers du Web. Simple et facile à implémenter dans les serveurs Web ou dans les serveurs d'applications, SOAP est indépendant des langages de programmation ou des systèmes d'exploitation employés pour l'implémentation des services Web. Les concepteurs de SOAP ont, en effet, réussi à préserver la plus grande généralité dans la représentation en XML des principes des protocoles de communication. S'inspirant des générations précédentes de RPC (Remote Procedure Call) et de RPC objet (Corba, DCOM), ils se sont attachés à en formaliser la définition dans des documents XML échangés sur le Web, sous forme de dialogue entre le client et le serveur. SOAP repose sur XML et sur quelques standards dérivés, les NameSpaces et XML Schema, en particulier.

SOAP est développé conjointement par Microsoft, IBM, Lotus Development (une division d'IBM), DevelopMentor et UserLand Software sous les auspices du W3C. SOAP 1.1 fit l'objet d'une note soumise au W3C en mai 2000, et SOAP 1.2 d'un document de travail (working draft) en juillet 2001. Microsoft a déjà, quant à lui, annoncé l'intégration de SOAP dans la plateforme .NET, en particulier dans BizTalk Server, le fer de lance de la stratégie Internet de l'éditeur de Redmond.

Structure d'un message SOAP

La grammaire de SOAP est assez simple à comprendre. Elle procure un moyen d'accès aux objets par appel de méthodes à distance. Les deux plus fortes fonctionnalités de SOAP

sont sa simplicité et le fait que tout le monde a accepté de l'utiliser. Un message SOAP est composé de deux parties obligatoires : l'enveloppe SOAP et le corps SOAP ; et une partie optionnelle : l'en-tête SOAP.

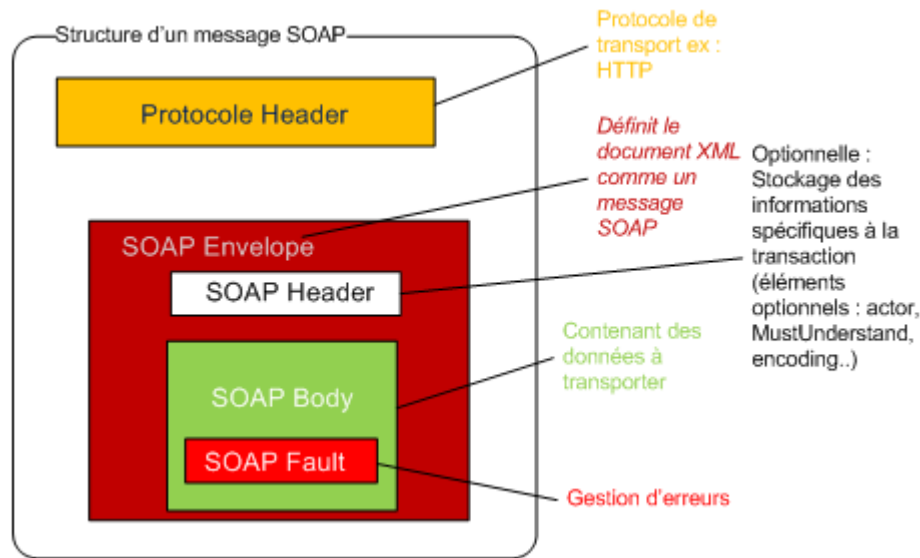


FIGURE 1.7 – Structure d'un message SOAP

- SOAP envelope (enveloppe) est l'élément de base du message SOAP. L'enveloppe contient la spécification des espaces de désignation (namespace) et du codage de données.
- SOAP header (entête) est une partie facultative qui permet d'ajouter des fonctionnalités à un message SOAP de manière décentralisée sans agrément entre les parties qui communiquent. C'est ici qu'il est indiqué si le message est mandataire ou optionnel. L'entête est utile surtout, quand le message doit être traité par plusieurs intermédiaires.
- SOAP body (corps) est un container pour les informations mandataires à l'intention du récepteur du message, il contient les méthodes et les paramètres qui seront exécutés par le destinataire final.
- SOAP fault (erreur) est un élément facultatif défini dans le corps SOAP et qui est utilisé pour reporter les erreurs.

1.5.5 Le débat SOAP vs. REST

la façon la plus répandue d'utiliser SOAP consiste à s'appuyer sur le protocole HTTP en utilisant la liaison SOAP-HTTP. Dans cette liaison, il est possible d'associer plusieurs opérations à la même URL. Par exemple, un service de ventes peut être placé à l'URL *www.unservicedevente.fr/serviceweb*, et toutes les opérations de ce service (demande de devis,

placement de bon de commande, suivi de bon de commande, etc.) peuvent être fournies sur cette même URL. Une application donnée peut alors appeler chacune de ces opérations en utilisant la méthode HTTP POST, et en incluant le nom de l'opération concernée par la requête dans l'en-tête HTTP « SOAPAction »[16].

- il rajoute peu de fonctionnalités au-dessus de ce qu'il est déjà possible de faire avec HTTP et XML (sans les extensions apportées par SOAP);
- en associant plusieurs opérations à une même URL, il rend difficile, voire impossible, l'utilisation de l'infrastructure de « caching » associée au protocole HTTP, qui constitue sans doute l'un des points forts de HTTP.

Dans l'approche REST, chaque opération d'un service est associée à une URL distincte et l'accès à chaque URL peut être réalisé en utilisant l'une des quatre méthodes fournies par HTTP : POST, GET, PUT et DELETE. Le contenu des messages est alors encodé en XML, et la distinction entre en-tête et contenu de message est laissée à la charge des applications qui requièrent cette distinction. Le résultat est un ensemble de conventions plus simples que celles de SOAP, et la possibilité d'utiliser des bibliothèques existantes pour l'échange de messages sur HTTP, éliminant ainsi le besoin de plates-formes implantant SOAP, qui dans certains cas peuvent être considérées comme étant plus « lourdes » et plus difficiles à utiliser.

Plusieurs services Web populaires disponibles à l'heure actuelle utilisent l'approche REST (appelée aussi « XML sur HTTP »). Ceci est le cas notamment de la maison de ventes au enchères en ligne eBay, qui rend une partie de ses fonctionnalités, accessibles sous forme de services Web « style REST » (voir <http://developer.ebay.com/rest>). Il en est de même du site de vente par Internet Amazon (voir <http://www.amazon.com/webservices>).

Le débat SOAP vs. REST est encore ouvert et les avis divergent considérablement dans la communauté. Les avantages et désavantages relatifs de SOAP et REST peuvent être résumés comme suit :

- REST ne requiert pas d'infrastructure spécifique pour le développement et l'exécution des services Web. Il repose sur l'infrastructure HTTP existante, qui a fait ses preuves dans le contexte des applications HTML classiques. De nombreux outils de programmation d'applications au-dessus de HTTP existent, et peuvent être combinés avec des outils de manipulation de documents XML pour construire des services Web « style REST ». Dans le cas de SOAP, il est nécessaire d'utiliser des outils spécifiques qui sont parfois difficiles à déployer et n'ont pas complètement fait leurs preuves.
- SOAP peut opérer au-dessus d'autres protocoles que HTTP, en particulier sur des protocoles permettant des communications asynchrones telles que SMTP¹¹, JMS¹², MSMQ¹³ et MQSeries.

11. Simple Mail Transfer Protocol

12. Java Message Service

13. Microsoft Message Queuing

- Lorsqu'il est utilisé en conjonction avec des spécifications WS-*¹⁴, SOAP permet de prendre en compte des aspects liés à la sécurité, la fiabilité et l'adressage et routage de messages. Bien sûr, il serait possible de faire de même en utilisant l'approche REST, mais les standards et l'infrastructure correspondant ne sont pas en place.

En conclusion, l'approche REST est viable (voire préférable) dans le cas de services Web avec les caractéristiques suivantes :

- ils n'ont besoin d'être exposés que sur HTTP ou HTTPS¹⁵ ;
- les communications asynchrones (et donc l'adressage explicite) ne sont pas requises ;
- ils ne requièrent pas de garanties de sécurité au delà de celles offertes par HTTPS ;
- la fiabilité peut facilement être traitée au niveau des applications.



FIGURE 1.8 – Web service RESTful

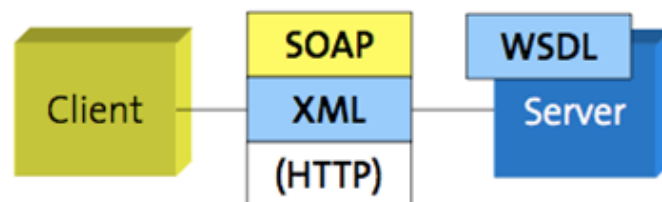


FIGURE 1.9 – Web service SOAP

1.5.6 Description des services web avec WSDL

Le WSDL (Web Services Description Language) ou Le langage de description des services Web (WSDL) est un langage de description d'interface basé sur XML qui est utilisé pour décrire les fonctionnalités offertes par un service Web. Une description WSDL d'un service Web (également appelé fichier WSDL) fournit une description lisible par machine de la façon

14. Web Services

15. HyperText Transfer Protocol Secure

dont le service peut être appelé, les paramètres attendus et les structures de données qu'il retourne. Cela correspond donc à un but qui correspond approximativement à celui d'une signature de méthode dans un langage de programmation.

WSDL 1.1 a été proposé en 2001 au W3C pour standardisation mais n'est pas approuvée par le W3C. La version 2.0 a été approuvée le 27 juin 2007 et est désormais une recommandation officielle du W3C. Le WSDL décrit une interface publique d'accès à un service web, notamment dans le cadre d'architectures de type SOA (Service Oriented Architecture). C'est une description fondée sur le XML qui indique « comment communiquer pour utiliser le service ». Le WSDL sert à décrire [5]

- le protocole de communication (SOAP RPC ou SOAP orienté message).
- le format de messages requis pour communiquer avec ce service.
- les méthodes que le client peut invoquer.
- la localisation du service.

La version actuelle de WSDL est WSDL 2.0. La signification de l'acronyme a changé depuis la version 1.1, où le D représentait Définition.

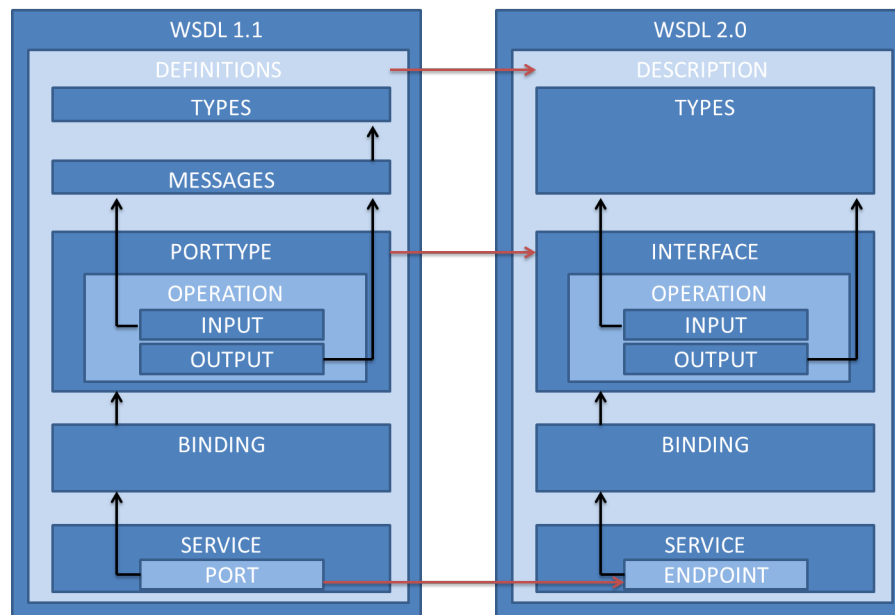


FIGURE 1.10 – Structure de wsdl 1.1 et wsdl 2.0

Le WSDL décrit les services en tant que collections de points d'extrémité réseau ou de ports. Les spécifications WSDL fournissent un format XML pour les documents à cette fin. Les définitions abstraites des ports et des messages sont séparées de leur utilisation ou de leur utilisation concrète, ce qui permet de réutiliser ces définitions. Un port est défini en associant une adresse réseau à une liaison réutilisable, et une collection de ports définit un service. Les messages sont des descriptions abstraites des données échangées, et les types de ports sont

des collections abstraites d'opérations supportées. Le protocole concret et les spécifications de format de données pour un type de port particulier constituent une liaison réutilisable, où les opérations et les messages sont alors liés à un protocole de réseau concret et à un format de message. De cette façon, WSDL décrit l'interface publique du service Web.

WSDL est souvent utilisé en combinaison avec SOAP et un schéma XML pour fournir des services Web par Internet . Un programme client se connectant à un service Web peut lire le fichier WSDL pour déterminer les opérations disponibles sur le serveur. Tous les types de données spécifiques utilisés sont incorporés dans le fichier WSDL sous la forme de XML Schéma. Le client peut ensuite utiliser SOAP pour appeler réellement l'une des opérations répertoriées dans le fichier WSDL en utilisant par exemple XML via HTTP. La version actuelle de la spécification est 2.0 ; La version 1.1 n'a pas été approuvée par le W3C, mais la version 2.0 est une recommandation du W3C. En acceptant la liaison à toutes les méthodes de requêtes HTTP (non seulement GET et POST comme dans la version 1.1), la spécification WSDL 2.0 offre un meilleur support pour les services Web RESTful et est beaucoup plus simple à mettre en œuvre.

WSDL 1.1	WSDL 2.0	la description
un service	un service	Contient un ensemble de fonctions système qui ont été exposées aux protocoles basés sur le Web.
Port	endpoint	Définit l'adresse ou le point de connexion d'un service Web. Il est généralement représenté par une simple chaîne d'URL HTTP
binding	binding	Spécifie l'interface et définit le style de liaison SOAP (RPC / Document) et le transport (protocole SOAP). La section de liaison définit également les opérations.
Opération	Opération	Définit les actions SOAP et la façon dont le message est encodé, par exemple, "littéral". Une opération est comme une méthode ou un appel de fonction dans un langage de programmation traditionnel.
message	/	Généralement, un message correspond à une opération. Le message contient les informations nécessaires à l'exécution de l'opération. Chaque message est composé d'une ou plusieurs parties logiques. Chaque partie est associée à un attribut de saisie de message. L'attribut de nom de message fournit un nom unique parmi tous les messages. L'attribut de nom de pièce fournit un nom unique parmi toutes les parties du message inclus. Les parties sont une description du contenu logique d'un message. Dans la liaison RPC, une liaison peut faire référence au nom d'une partie afin de spécifier des informations spécifiques à la liaison sur la pièce. Une partie peut représenter un paramètre dans le message ; Les liaisons définissent la signification réelle de la partie. Les messages ont été supprimés dans WSDL 2.0, dans lequel les types de schémas XML pour définir les corps d'entrées, de sorties et de défauts sont mentionnés de manière simple et directe.
Les types	Les types	Décrit les données. Le langage du schéma XML (également appelé XSD) est utilisé (en ligne ou référencé) à cette fin.

TABLE 1.1 – Objets dans WSDL 1.1 / WSDL 2.0

1.5.7 Découverte des services web avec UDDI

A partir du moment où un Web Services est créé, une entreprise doit pouvoir le rendre disponible et inversement, les développeurs doivent pouvoir le trouver. La découverte d'un Web Services est une étape essentielle car c'est à cette occasion que l'on pourra récupérer le contrat WSDL sans lequel rien ne serait possible. C'est suite à ce besoin que les sociétés Ariba, IBM et Microsoft ont débutées l'élaboration du standard UDDI (Universal Description, Discovery and Integration), finalement repris puis défini par l'OASIS, dans le but de fournir

une manière standard de publier et d'interroger les Web Services. La spécification UDDI adopte une approche basée sur un annuaire virtuel distribué qui permet à un utilisateur, via une interface Web, de publier ou de rechercher un Web Services. UDDI est pour simplifier comparable à un annuaire téléphonique papier traditionnel (en plus complet).

Types de structure de données (registre UDDI) :

L'enregistrement d'un service web implique quatre types de structure de données essentiels : informations métier, informations sur les services, informations sur les liaisons et informations décrivant les spécifications relatives aux services. Les relations entre ces types de données sont décrites dans la Figure 1.11.

- **Informations métier** : Informations contenues dans une structure *businessEntity*. La structure *businessEntity* regroupe des informations sur l'entreprise ayant publié le service, tel que le nom, la description, les contacts et les identificateurs de cette entreprise.
- **Informations sur les services** : Informations décrivant un groupe de services Web. Elles figurent dans une structure *businessService*. La structure *businessService* contient des informations sur les familles de services techniques. Elle regroupe un ensemble de services Web associés à un processus métier ou à un groupe de services.
- **Informations de liaison** : Informations représentées par la structure *BindingTemplate*. La structure *bindingTemplate* détient des informations techniques servant à déterminer le point d'entrée et les spécifications de construction pour l'appel d'un service Web. Elle fournit les descriptions de service Web utiles aux développeurs d'applications souhaitant rechercher et appeler un service Web. La structure *bindingTemplate* pointe vers des descriptions d'implémentation d'un service, par le biais d'une adresse URL, par exemple.
- **Informations décrivant les spécifications relatives aux services** : Les métadonnées afférentes aux différentes spécifications implémentées par un service Web donné, sont représentées par la structure *tModel*. *tModel* fournit un système de références facilitant la reconnaissance des services Web.

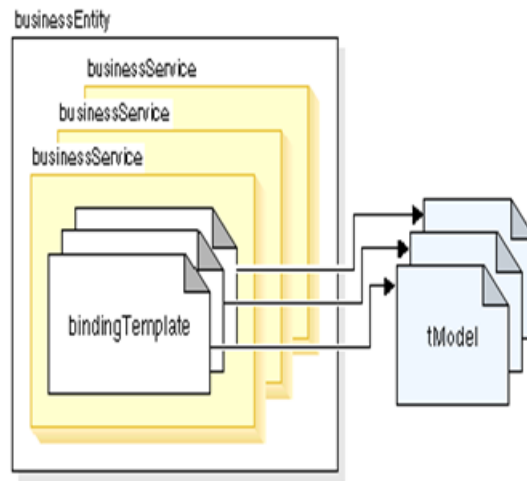


FIGURE 1.11 – Relations entre les structures de données UDDI

Registres UDDI

UDDI gère la reconnaissance des services Web en s'appuyant sur un registre réparti de métiers et sur la description de leurs services implémentés sous la forme d'un fichier commun au format XML. Avant de publier l'entité métier et le service Web dans un registre public, vous devez d'abord enregistrer votre entité métier dans un registre UDDI. Il existe deux types de registre UDDI : public et privé. Ces deux types relèvent des mêmes spécifications.

Un registre public est une collection de répertoires homologues contenant des informations sur les métiers et les services. Il localise les services enregistrés sur l'un de ses noeuds homologues et facilite la reconnaissance des services Web publiés. Les données sont répliquées sur chacun des registres à intervalles réguliers. Cela assure une certaine cohérence dans les formats de description de service et facilite le traçage en temps réel des modifications. IBM gère deux registres publics : le registre d'entités UDDI IBM (UDDI Business Registry) et le registre UDDI de test IBM (UDDI Test Registry). Le registre UDDI de test IBM vous permet de développer votre service Web et d'expérimenter le processus d'enregistrement UDDI sans déposer votre service Web dans un registre officiel. Utilisez ce registre pour vous familiariser avec UDDI, et pour tester et valider votre service Web.

Un registre privé vous permet de publier et tester vos applications e-business internes dans un environnement privé sécurisé.

les entreprises alimentent le registre avec les descriptions des services qu'elles prennent en charge. UDDI affecte un identificateur unique à chaque description de service et enregistrement d'entreprise. Ces deux données deviennent respectivement la clé du métier et la clé du service. Les moteurs de recherche et les applications demandent au registre de détecter des

services. Les serveurs UDDI sont constitués d'un répertoire de services et de fournisseurs de services disponibles.

Consultation de l'annuaire

L'annuaire UDDI se concentre sur le processus de découverte de l'architecture orientée services (SOA), et utilise des technologies standards telles que XML, SOAP et WSDL qui permettent de simplifier la collaboration entre partenaires dans le cadre des échanges commerciaux. L'accès au référentiel s'effectue de différentes manières.

- Les pages blanches comprennent la liste des entreprises ainsi que des informations associées à ces dernières (coordonnées, description de l'entreprise, identifiants...).
- Les pages jaunes recensent les services Web de chacune des entreprises sous le standard WSDL.
- Les pages vertes fournissent des informations techniques précises sur les services fournis.

Les entreprises publient les descriptions de leurs services Web en UDDI, sous la forme de fichiers WSDL. Ainsi, les clients peuvent plus facilement rechercher les services Web dont ils ont besoin en interrogeant le registre UDDI.

Lorsqu'un client trouve une description de service Web qui lui convient, il télécharge son fichier WSDL depuis le registre UDDI. Ensuite, à partir des informations inscrites dans le fichier WSDL, notamment la référence vers le service Web, le client peut invoquer le service Web et lui demande d'exécuter certaines de ses fonctionnalités.

Le scénario classique d'utilisation de UDDI est illustré ci-dessous. L'entreprise B a publié le service Web S, et l'entreprise A est client de ce service :

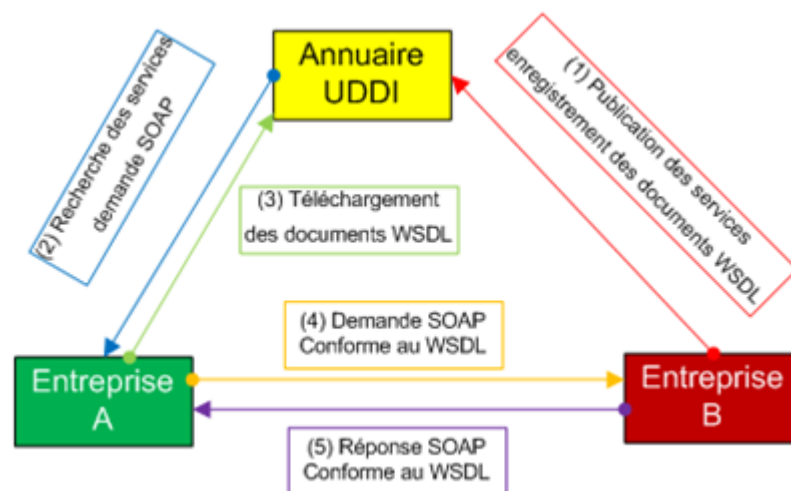


FIGURE 1.12 – Utilisation de UDDI

1.6 Modèles d'interaction des services web

D'après la figure 1.13, la collaboration entre service web s'appuie sur un modèle d'interaction dont les composants assurent trois rôles : le fournisseur de services , l'annuaire de services et le demandeur de services.

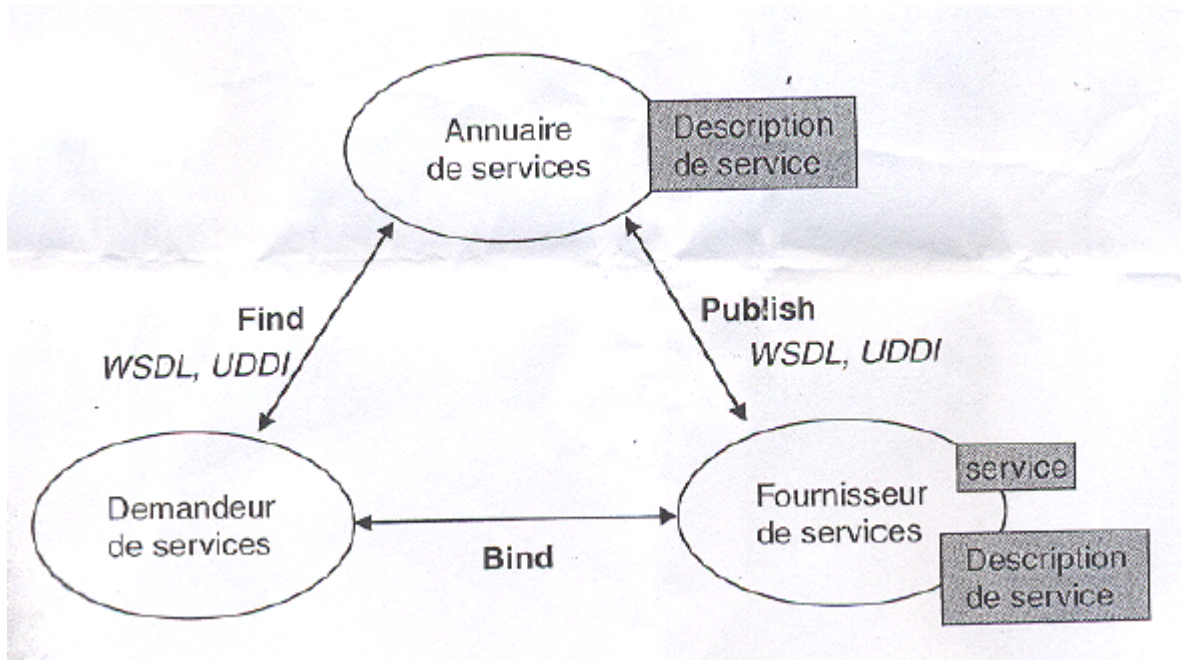


FIGURE 1.13 – Modèle d'interactions des services web

1.6.1 Scénario générale de fonctionnement des services web

Dans le scénario de fonctionnement normal, un fournisseur de services héberge un module logiciel implémentant un ou plusieurs service web accessibles via de le réseau. Il définit une description du service et le publie en le faisant enregistrer dans un annuaire de service. Le demandeur de services invoque une opération de recherche pour trouver la description du service dans l'annuaire de service. il utilise alors la description du service pour établir une connexion avec le fournisseur de services et invoquer ou interagir avec l'implémentation du service web. ce scénario peut être récursif dans le cadre des invocations de services récursives. Il existe trois types d'opérations pour tirer pleinement parti de ce modèle :

- la publication de description de services (*publish*).
- la recherche et la découverte de la bonne description du service (*find*) ;
- l'association ou l'invocation des services basées sur la description (*bind*).

Afin d'assurer la collaboration entre applications, chacun des composants de ce modèle d'interaction présente une facette métier et une facette d'architecture technique :

- en ce qui concerne le fournisseur de services : d'un point de vue métier, il s'agit du propriétaire du service et d'un point de vue architecture technique, il s'agit de la plate-forme qui héberge l'accès au service.
- en ce qui concerne le demandeur de services : d'un point de vue métier, cela consiste à demander certaines fonctionnalités à satisfaire et d'un point de vue technique, il s'agit d'une application qui recherche et qui invoque ou initialise une interaction avec un service. Le rôle du demandeur du service peut être assuré par un browser piloter par une personne ou un programme sans interface utilisateur, par exemple un autre service web.
- En ce qui concerne l'annuaire de services : c'est un annuaire de recherche de description de services où les fournisseurs de service publient leurs descriptions de services. Les demandeurs de services trouvent des services et obtiennent les informations correspondantes à leur demande pendant le développement de manière statique ou en cours d'exécution dynamiquement.

1.6.2 Étape d'exécution des services web

D'une manière simplifiée, les principales étapes d'exécution d'un service web sont les suivantes :

- **Découverte du service** : Le demandeur de service lance la recherche d'un service correspondant à ses besoins sur un annuaire UDDI qui peut être public ou privé.
- **Récupération des informations de description du service** :Le demandeur de service récupère de l'annuaire UDDI la description de ce service au format WSDL.
- **Connexion au service web** :la communication entre le composant demandeur du service et celui fournisseur du service est assuré en phase d'exploitation à travers des wrappers (*listener et proxy*) SOAP qui servent d'interface entre ces composants et les protocoles de communication de l'infrastructure de déploiement. Le proxy du composant demandeur du service émet une requête SOAP au composant fournisseur du service. Le protocole HTTP véhicule le message SOAP jusqu'au listener du fournisseur du service.
- **le service web renvoie sa réponse** :Le service web du fournisseur renvoie sa réponse au demandeur sous la forme d'un document XML via SOAP et dans la pratique, la synchronisation de traitements impliquant plusieurs services web reste difficile. L'orchestration de processus étendus reste difficile à réaliser faute de la couche de transport SOAP et d'outils adaptés pour effectuer le déploiement.

1.7 Conclusion

Dans ce chapitre nous avons, au premier lieu, présenté le concept de service web qui fait aujourd'hui une technologie révolutionnaire. Elle fournit un cadre pour trouver, décrire et exécuter ces applications à travers le réseau Internet indépendamment de tout langage de programmation et de toute plate-forme d'exécution.

Par la suite, nous avons présenté les technologies sur lesquels reposent les services web tels que SOAP, WSDL et UDDI qui permettent à des applications distantes de dialoguer entre elles.

Il existe plusieurs plates-formes de développement et de déploiement de services web. Les deux les plus célèbres sont : Microsoft .NET et J2EE. et ce qu'on va entamer en détaille dans le chapitre suivant.

Les plateformes de développement des services web

2.1 Introduction

Le modèle de services Web devient rapidement l'approche privilégiée pour créer, acquérir et implémenter des applications métier en raison de ses caractéristiques de développement rapide des applications, de son faible coût d'entrée et de la cohérence de l'échange d'informations. Ses interfaces d'application XML basées sur le texte sont une évolution des prédécesseurs propriétaires et dépendant de la plate-forme tels que CORBA et COM IDL (Interface Definition Language). Les informations commerciales échangées dans les applications de services Web sont conformes aux normes verticales de transaction et de données en matière de finances, d'assurance, de fabrication, de santé, de produits pharmaceutiques et de la plupart des autres entreprises. Ces normes sont également exprimées en XML, ce qui les rend accessibles à tout système qui comprend le texte et peut se connecter au Web. Les fournisseurs se précipitent pour fournir des logiciels de frameworks d'applications Web Services qui profitent de ce modèle et basent leur offre sur deux frameworks d'applications : Microsoft .NET et Sun J2EE.

2.2 La plateforme.NET

Les services Web sont la composante essentielle de la stratégie .NET de Microsoft : De manière assez simple, .NET est la plate-forme de Microsoft dédiée aux services Web XML. La plate-forme .NET de Microsoft comprend une famille de produits bâtis autour d'XML et des standards industriels d'Internet, qui couvre tous les aspects du développement, de la gestion, de l'usage courant ou de l'expérimentation des services Web XML. Annoncée officiellement en juin 2000, la stratégie .NET de Microsoft est un projet d'une

envergure sans doute comparable au développement de Windows dans les années quatre-vingt-dix. Et il faut bien parler de stratégie car il ne s'agit pas de développer un produit ou même une ligne de produits mais d'axer tous les développements de la société vers un objectif global qui correspond à une nouvelle vision de l'informatique distribuée et coopérante (les logiciels deviennent des services). Le projet est ambitieux et fait évidemment couler beaucoup d'encre.

Dans les faits, la stratégie .NET s'articule autour de trois axes (voir figure 2.1) :

- **Les produits Serveur d'entreprise .NET** : Ils regroupent tous les logiciels serveur de Microsoft : SQL Server 2000, Exchange 2000, BizTalk Server, Commerce Server, etc. Tous ont évolué et prennent ou prendront en charge XML et les services Web. Si Exchange et SQL Server sont des logiciels bien connus dans le monde Microsoft, il n'en est pas de même pour BizTalk Server. Pourtant, cet outil occupe une place prépondérante dans ce grand ensemble puisqu'il est dédié aux problèmes d'échanges de données Informatisé (EDI, XML) et d'orchestration (flux de données y compris à faible couplage).
- **Le framework .NET et les outils** : Le framework .NET, que nous allons présenter en détail dans ce chapitre, est la nouvelle plate-forme de développement et d'exécution de Microsoft. Cette plate-forme est par ailleurs la cible principale de la nouvelle version de Visual Studio, l'environnement de développement de Microsoft.
- **Les services Web de base regroupés dans le cadre du projet HailStorm** : Les services Web de base que l'on nomme aussi .NET MyServices, sont constitués d'une gamme de services Web offrant des fonctions essentielles comme l'authentification avec Microsoft Passport ou un carnet d'adresses avec HotMail. Cela est donc la preuve tangible que des services Web peuvent être développés et utilisés par n'importe qui pour répondre à une demande de service [15].

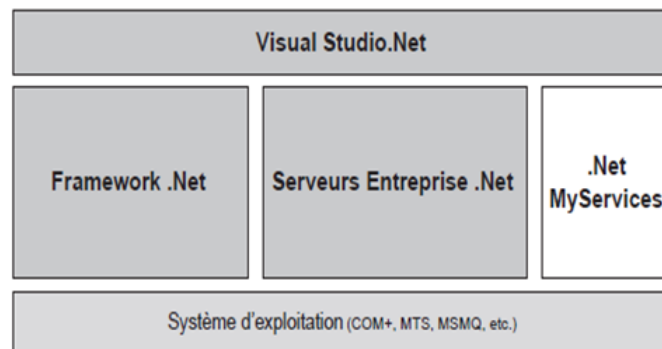


FIGURE 2.1 – L'architecture .NET de Microsoft

2.2.1 Le framework.NET

Le framework .NET est la nouvelle plate-forme logicielle de Microsoft qui permet de construire, de déployer et d'exécuter des services Web et des applications qui les utilisent (voir figure 2.2). Cette plate-forme est en principe indépendante des outils de développement, même s'il faut reconnaître que Visual Studio .NET est le seul à l'exploiter pleinement (rien ne vous empêche d'utiliser un simple éditeur de texte et d'appeler le compilateur en ligne de commande).

Au-delà de la prise en charge des services Web, la plate-forme est censée répondre à tous les besoins des développeurs, c'est-à-dire qu'elle permet de développer des applications Internet mais aussi des applications classiques s'exécutant sur Windows (ou, en principe, tout système d'exploitation prenant en charge le framework .NET).

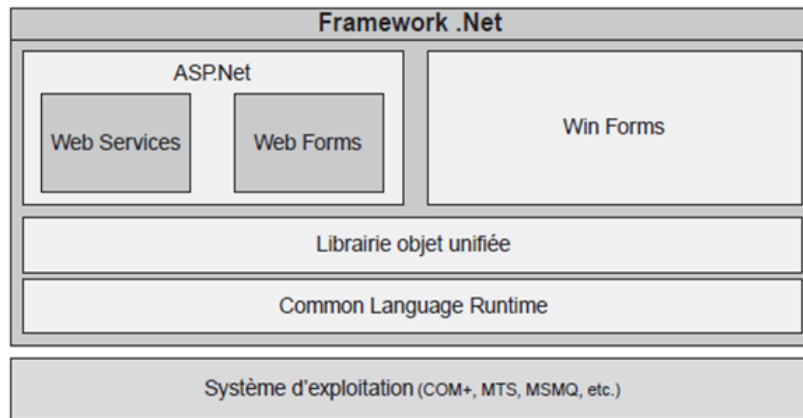


FIGURE 2.2 – L'architecture du Framework.net

Le framework .NET est constitué de quatre composants principaux que nous allons décrire en détail :

- une machine virtuelle appelée CLR (Common Language Runtime);
- un ensemble hiérarchisé et unifié de bibliothèques objets (.NET Framework Class Library);
- un environnement d'exécution d'applications et de services Web (ASP .NET);
- un environnement d'exécution d'applications graphiques « natives » (Win Forms).[15]

2.2.2 Common Language Runtime (CLR)

Le composant central de l'architecture .NET est le CLR qui est un environnement d'exécution pour des applications réparties écrites dans des langages différents. En effet, avec .NET on peut véritablement parler d'interopérabilité entre langages. Cette interopérabilité s'appuie sur le CLS¹ qui définit un ensemble de règles que tout compilateur de la plate-forme doit

1. Common Language Specification

respecter.

Le CLS utilise entre autres un système de types unifié permettant d'avoir le même système de type entre les différents langages et le même système de type entre les types prédéfinis et les types définis par l'utilisateur.

Un autre concept majeur, lié au CLR est celui de code géré, c'est à dire de code exécuté sous le contrôle de la machine virtuelle. Dans cet environnement, un ensemble de règles garantissent que les applications se comporteront d'une manière uniforme, et ce indépendamment du langage ayant servi à les écrire.

Pour répondre à ce souci d'interopérabilité entre les langages, la plate-forme .NET contient une bibliothèque de classes très complète, utilisable depuis tous les langages et permettant aux développeurs d'utiliser une même interface de programmation pour toutes les fonctions offertes.

Dans .NET, le langage lui-même est essentiellement une interface syntaxique des bibliothèques définies dans le CLR. Grâce au jeu commun de ces bibliothèques, tous les langages disposent théoriquement des mêmes capacités car ils doivent, exception faite de la déclaration des variables, passer par ces bibliothèques pour créer des threads, les synchroniser, sérialiser des données, accéder à internet, etc[15].

2.2.3 ASP.NET

ASP.Net est une plate-forme de développement Web fournie par Microsoft. Il est utilisé pour créer des applications Web. ASP.Net a été publié pour la première fois en 2002.

La première version d'ASP.Net déployée était 1.0. La version la plus récente d'ASP.Net est la version 4.6. ASP.Net est conçu pour fonctionner avec le protocole HTTP. C'est le protocole standard utilisé dans toutes les applications Web.

ASP.NET est donc la nouvelle plate-forme unifiée de développement que propose Microsoft pour construire tout type d'application serveur orientée Internet. Cette infrastructure offre aux développeurs deux fonctionnalités principales qui peuvent être combinées entre elles :

- les Web Forms ;
- les services Web XML.

Dans les deux cas, les moyens mis en œuvre sont les mêmes et les évolutions par rapport à la précédente version d'ASP sont majeures.

Le CLR a été conçu pour prendre en charge différents scénarios d'applications : depuis une application serveur Internet jusqu'à une application Windows en passant par une application console. Chaque type d'application nécessite un programme hôte spécifique qui doit s'occuper de :

1. charger le CLR dans un processus ;
2. créer un domaine d'application dans ce processus ;

3. charger le code utilisateur dans ce domaine.

ASP.NET, tout comme Internet Explorer et le shell de Windows, est l'un de ces programmes hôtes fournis par le framework .NET : son rôle consiste donc à charger le CLR dans le processus qui doit traiter la requête Internet puis à créer un domaine d'application pour chaque application Internet qui tourne sur le serveur.[15]

2.2.4 Visual studio.net

Microsoft Visual Studio est une suite de logiciels de développement pour Windows conçue par Microsoft. La dernière version s'appelle Visual Studio 2017.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications web ASP.NET, des services web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# utilisent tous le même environnement de développement intégré (IDE), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications web ASP et de services web XML grâce à Visual Web Developer.

Services Web dans Visual Studio .NET

Microsoft Visual Studio .NET simplifie la création et le déploiement de Services Web en automatisant la création de plusieurs des fichiers nécessaires à leur implémentation. Ainsi, il met à votre disposition un type de projet Service Web ASP.NET qui facilite la création et la manipulation des services XML Web.

Lorsque vous créez un projet Service Web ASP.NET, Visual Studio .NET construit automatiquement, sur le serveur Web (nous supposons que vous développez le service sur un serveur Web), le projet et les fichiers nécessaires à l'implémentation du service Web XML. Par exemple, il crée automatiquement un fichier .asmx qui contient les classes et les méthodes ajoutant des fonctionnalités au service Web XML.

De plus, Visual Studio .NET crée les fichiers WSDL et WSML². Il affiche même une page d'informations contenant les méthodes et les paramètres utilisés dans le service Web XML. L'outil Web Service References extrait les informations du fichier WSDL et crée une classe proxy Microsoft Visual Basic pour Applications (VBA³). Cette classe proxy est une représentation locale du service Web XML dans l'application cliente. De plus, étant donné que la classe de service Web XML est liée à la classe proxy de l'application cliente, la fonctionnalité Microsoft IntelliSense vous facilite la tâche lors de la création d'objets sur le client. Pour

2. Web Service Modeling Language

3. Visual Basic for Applications

pouvoir commencer à utiliser le service Web, il ne vous reste plus qu'à ajouter dans votre application le code nécessaire pour appeler les méthodes de la classe proxy.

2.3 La plateforme J2EE

La plate-forme Java EE (Enterprise Edition, anciennement J2EE) propose un développement simplifié, rapide et robuste reposant désormais sur la configuration par annotations, directement dans les fichiers source. La sortie de Java EE version 6 en 2009 correspond à l'anniversaire des dix ans de la plate-forme. La première version standard J2EE 1.2 a été développée par Sun en 1999 et contenait 10 spécifications ou Java Specification Requests (JSR). Les Entreprises JavaBeans (EJB) permettaient de gérer la couche métier d'accès aux données alors que les Servlets et JavaServer Pages (JSP) permettaient de développer les applications serveur. La version J2EE 1.3 sortie en 2001 n'apportait pas de nouveautés essentielles mais plutôt des améliorations de la plate-forme à la suite des demandes utilisateur, comme la configuration par fichier XML [4].

2.3.1 L'architecture J2EE

Java Enterprise Edition (JEE) est un ensemble de recommandations et de spécifications techniques destinées à étendre le framework Java Standard Edition (JSE) de Sun Microsystems. JEE est particulièrement destiné aux applications d'entreprise. Toutefois, il ne s'agit que de spécifications qui sont ensuite supportées en tout ou partie par les serveurs d'applications. Les spécifications JEE portent sur quatre éléments particuliers : Un ensemble d'API fournissant au développeur des opérations génériques qu'il n'aura donc pas à redévelopper ou à rechercher. La plupart de ces bibliothèques sont disponibles aussi avec JSE (Collections, etc.). Les opérations fournies peuvent être des opérations sur des bases de données (JDBC), des opérations sur des annuaires (JNDI), des opérations pour la gestion d'email (JavaMail), des opérations pour traiter des données XML (JAXP), des opérations pour la communication synchrone (JAX-RPC) ou asynchrone (JAXM, Java RMI, JMX), etc.

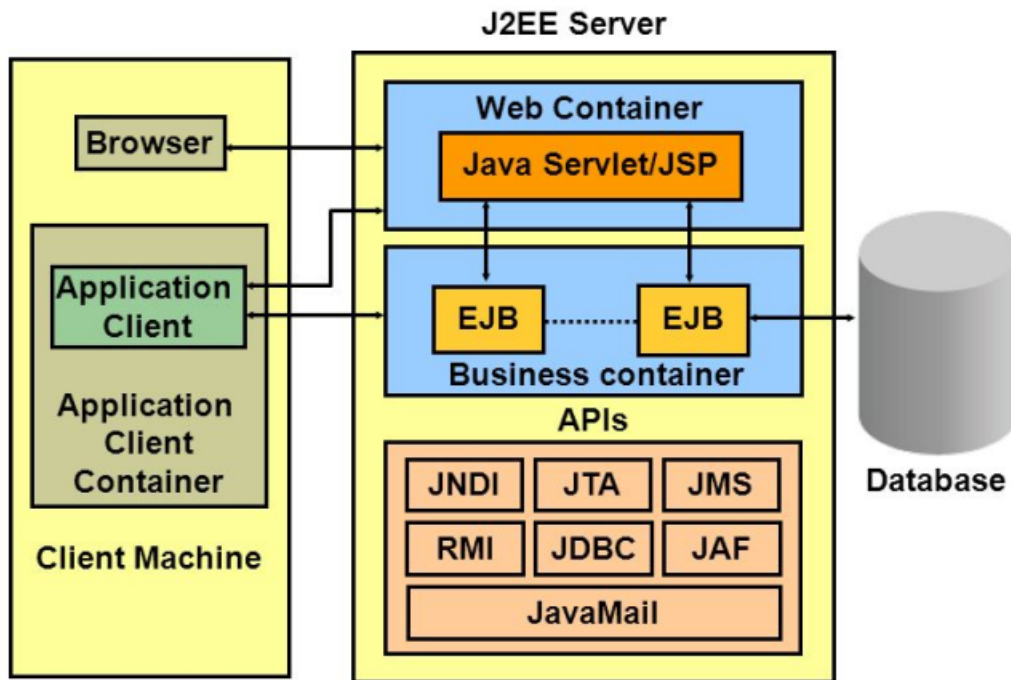


FIGURE 2.3 – L'architecture J2EE

2.3.2 Services Web et J2EE

À partir de J2EE 1.4, la plate-forme Java EE a pris en charge le développement de services Web orientés vers les appels à la procédure à distance basés sur SOAP via un composant appelé JAX-RPC.

Avec la sortie de Java 6, la plate-forme Java a été mise à jour pour fournir un support beaucoup plus complet pour le développement de services Web. Cela s'effectue via une pile intégrée d'API et d'outils conçus pour faciliter le développement des services Web SOAP et RESTful, qui fait désormais partie des fonctionnalités Java principales.

De plus, plutôt que d'être spécifique à l'édition d'entreprise de Java, la nouvelle pile de services Web est incluse dans les éditions Java SE et EE. Bien que la pile soit identique dans les deux éditions, lorsqu'elles sont utilisées avec JavaEE, des composants supplémentaires permettent de définir les services et exposés dans des environnements basés sur des conteneurs d'entreprise, et interagissent avec JMS, JNDI⁴ et d'autres composants JavaEE.

La pile de services Web Java se compose de 5 composants principaux :

- JAX-WS - est une API de langage de programmation Java pour créer des services Web , en particulier des services SOAP.
- JAXB - Ce composant aide JAX-WS à traiter XML en permettant une liaison facile du schéma XML aux formats Java standard.

4. Java Naming and Directory Interface

- WSIT - Ce composant étend JAX-WS pour permettre aux services Web Java d'interagir avec .NET et d'autres composants WCF⁵.
- XWS-Security - Ce cadre peut être intégré dans un service Web JAX-WS pour permettre la configuration des politiques de sécurité pour les demandes et les réponses à plusieurs niveaux tout au long du service.
- JAX-RPC - Ce cadre est inclus pour fournir un support hérité pour les services basés sur RPC, comme indiqué dans la spécification SOAP 1.1.[6]

2.3.3 les servlets

Un(e) servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont le plus généralement présentées au format HTML, mais elles peuvent également l'être au format XML ou tout autre format destiné aux navigateurs web. Les servlets utilisent l'API Java Servlet (package `javax.servlet`).

Un(e) servlet s'exécute dynamiquement sur le serveur web et permet l'extension des fonctions de ce dernier, par exemple : l'accès à des bases de données, transactions d'e-commerce, etc. Un(e) servlet peut être chargé automatiquement lors du démarrage du serveur web ou lors de la première requête du client. Une fois chargé(e)s, les servlets restent actifs dans l'attente d'autres requêtes du client.

L'utilisation de servlets se fait par le biais d'un conteneur de servlets (framework) côté serveur. Celui-ci constitue l'environnement d'exécution de la servlet et lui permet de persister entre les requêtes des clients. L'API définit les relations entre le conteneur et le/la servlet. Le conteneur reçoit la requête du client, et sélectionne le/la servlet qui aura à la traiter. Le conteneur fournit également tout un ensemble de services standards pour simplifier la gestion des requêtes et des sessions.[7]

2.3.4 Java Server Pages :

Le JavaServer Pages ou JSP est une technique basée sur Java qui permet aux développeurs de créer dynamiquement du code HTML, XML ou tout autre type de page web. Cette technique permet au code Java et à certaines actions prédéfinies d'être ajoutés dans un contenu statique. Depuis la version 2.0 des spécifications, la syntaxe JSP est complètement conforme au standard XML. Les JSP sont compilées par un compilateur JSP pour devenir des servlets Java. Un compilateur JSP peut créer une servlet Java en code source Java qui peut à son tour être compilé par le compilateur Java, ou peut créer le pseudo-code Java interprétable directement. Dans les deux cas, il est bon de comprendre comment le compilateur JSP transforme la page en servlet Java.[8]

5. Windows Communication Foundation

2.3.5 Les Enterprise JavaBeans (EJB) :

Enterprise JavaBeans (EJB) est une architecture de composants logiciels côté serveur pour la plateforme de développement Java EE.

Cette architecture propose un cadre pour créer des composants distribués (c'est-à-dire déployés sur des serveurs distants) écrit en langage de programmation Java hébergés au sein d'un serveur applicatif permettant de représenter des données (EJB dit entité), de proposer des services avec ou sans conservation d'état entre les appels (EJB dit session), ou encore d'accomplir des tâches de manière asynchrone (EJB dit message). Tous les EJB peuvent évoluer dans un contexte transactionnel.

De la version 1.0 à la version 2.1, un EJB était accompagné d'un ou plusieurs fichiers de déploiement écrit en XML qui permettait au serveur applicatif de déployer correctement l'objet au sein d'un conteneur. C'était notamment dans ces fichiers de déploiement que le développeur avait la possibilité de préciser le cadre transactionnel dans lequel l'objet allait s'exécuter. Depuis la version 3.0, le modèle EJB utilise le principe d'annotation java (meta-données) pour spécifier toute la configuration et les propriétés transactionnelles de l'objet. Le fichier de code source de l'EJB se suffit à lui-même.

Il existe 3 types de beans enterprise en Java (EJB).

- **Les EJB session (Session Bean)** Les EJB entité sont des *beans* ayant majoritairement pour vocation d'être persistants, c'est-à-dire pouvant être stockés sur un support physique entre deux sessions. Les EJB entité peuvent être de deux sortes : BMP (Bean Managed Persistence) ou CMP (Container Managed Persistence) Les EJB BMP sont des beans dont la persistance a dû être programmée par le développeur . Les EJB CMP sont eux des beans dont la persistance est directement assurée par le conteneur d'EJB ; le mapping entre l'objet et son support de persistance est indiqué au conteneur via les fichiers descripteurs de déploiement. Le développeur, une fois le fichier de déploiement réalisé, n'a pas besoin d'écrire le code de persistance. Depuis la version 3.0 de la spécification EJB, la notion de bean BMP/CMP n'existe plus : les EJB entité sont directement liés à la base de données via un mapping objet-relationnel. Ce mapping est défini soit dans un fichier de configuration XML, ou directement dans le code Java en utilisant des annotations.

- **Les EJB entité**

Les EJB entité sont des beans ayant majoritairement pour vocation d'être persistants, c'est-à-dire pouvant être stockés sur un support physique entre deux sessions. Les EJB entité peuvent être de deux sortes : BMP (Bean Managed Persistence) ou CMP (Container Managed Persistence) (voir Java Persistence API). Les EJB BMP sont des beans dont la persistance a dû être programmée par le développeur (ce dernier doit respecter un format pour la classe et les méthodes à implémenter sont imposées par la

norme). Les EJB CMP sont eux des beans dont la persistance est directement assurée par le conteneur d'EJB ; le mapping entre l'objet et son support de persistance est indiqué au conteneur via les fichiers descripteurs de déploiement. Le développeur, une fois le fichier de déploiement réalisé, n'a pas besoin d'écrire le code de persistance. Depuis la version 3.0 de la spécification EJB, la notion de bean BMP/CMP n'existe plus : les EJB entité sont directement liés à la base de données via un mapping objet-relationnel. Ce mapping est défini soit dans un fichier de configuration XML, ou directement dans le code Java en utilisant des annotations.

- **Les EJB message**

Depuis la norme EJB 2.0, cette architecture propose un troisième type de composant : les EJB message permettant de déclencher un processus côté serveur applicatif lors de la publication d'un message asynchrone. Pour ces composants, le client ne s'adresse pas directement aux composants mais publie un message sur un réceptacle JMS (queue ou topic) configuré sur le serveur applicatif qui va alors déclencher l'activation par ce serveur d'une instance de l'EJB concerné pour pouvoir traiter ce message.[9]

2.3.6 JAX-WS et JAX-RS :

JAX-WS (Java API for XML-Based Web Services) :

L' API Java pour les services Web XML (JAX-WS) est une API de langage de programmation Java pour créer des services Web , en particulier des services SOAP . JAX-WS est l'une des API de programmation Java XML . Cela fait partie de la plate-forme Java EE . JAX-WS est une technologie conçue pour simplifier la construction de services Web et de clients de services Web en Java. Il fournit une pile de services Web complète qui facilite la tâche de développement et de déploiement de services Web.

En outre, JAX-WS accélère le développement des services Web en fournissant une bibliothèque d'annotations pour transformer les anciennes classes d'objets Java (POJO) en services Web. Il spécifie également un mappage détaillé à partir d'un service défini dans le langage de description des services Web (WSDL) aux classes Java qui implémentent ce service.[10]

JAX-RS (Java API for RESTful Web Services) :

JAX-RS est l'API Java API des services web RESTful. Il fournit un support pour construire des services web avec REST. Les services web RESTful sont conçus pour exposer les API sur le web. Ils sont destinés à améliorer la performance, l'adaptabilité, et la flexibilité par rapport aux services web traditionnels, tout en permettant aux clients d'accéder à des données et à des ressources par des URL prédictibles.

2.4 La différence entre J2EE et .NET

J2EE et .NET fournissent chacun un cadre d'application pour les services Web, mais diffèrent dans la conception et le support, la mise en œuvre, le prix, la portabilité, les outils et les serveurs, le soutien des fournisseurs et la maturité et la popularité.

Design et Services :

Le .NET Framework comporte trois composants principaux. La plate-forme .NET se compose des outils et de l'infrastructure fournis pour créer des services .NET et un logiciel de périphérique .NET. Les produits et les services .NET sont fournis par un ensemble de services d'entreprise basés sur .NET qui prennent en charge le framework. Ceux-ci incluent BizTalk Server, SQL Server, Windows.NET, Visual Studio.NET et Office.NET, les fournisseurs tiers fournissent .NET Web Services construit sur la plate-forme .NET (voir la figure 2.4).

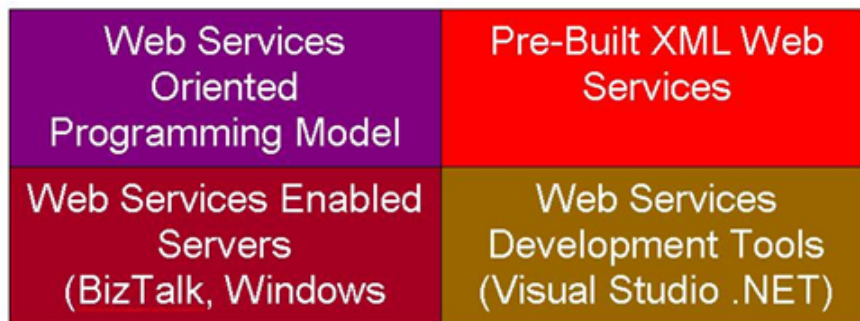


FIGURE 2.4 – Support web services .NET

Le cadre J2EE (voir la figure 2.5) répond à ces exigences avec un ensemble complet de normes, y compris Enterprise JavaBeans (EJB), J2EE Connector Architecture (JCA), Java Database Connectivity (JDBC), JavaServer Pages Standard Tag Library (JSTL), Servlets, Java Transaction API (JTA), Java Messaging Services (JMS), Java Name et Directory Interface (JNDI), Java Remote Method Invocation (RMI), RMI / Inter-Orb Operability Protocol (RMI / IIOP) Service d'authentification et d'autorisation Java, (JAAS), JavaMail et l'API Java pour XML (JAXP).

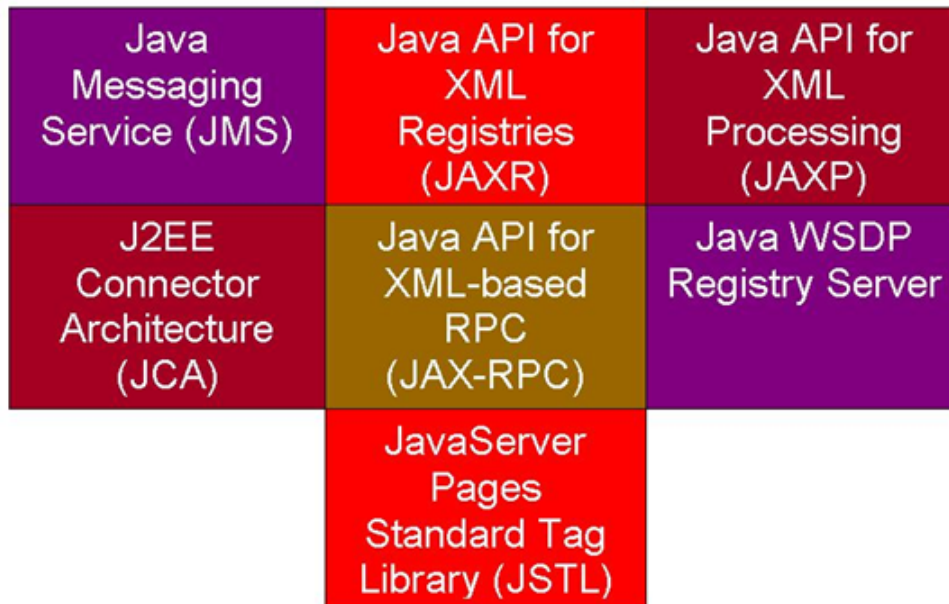


FIGURE 2.5 – Support J2EE web services

La différence essentielle est que J2EE prend en charge les services Web via les API Java, tandis que .NET Web Services est intégré à la plate-forme.

la mise en œuvre

Les services Web J2EE sont généralement implémentés avec Enterprise Java Beans, bien que les applications Java standard puissent également effectuer le travail. Le choix dépend de la façon dont les couches de traitement métier et de logique de données sont conçues et construites. Les services .NET sont généralement implémentés dans des composants gérés par la plate-forme .NET. Cela comprend la classe gérée et les composants COM ou COM +. .NET prend en charge Managed C ++, JScript.NET, VB.NET et C #, une version Web Services de C ++. Le framework J2EE, bien sûr, est limité à Java.

Le prix

J2EE est coûteux par rapport à .NET, mais si une entreprise possède déjà un serveur d'applications basé sur J2EE, comme Silverstream, WebLogic ou WebSphere, il est logique de l'utiliser, car le coût supplémentaire des services Web J2EE est bas dans ce cas . Les serveurs .NET coûtent beaucoup moins que les serveurs J2EE.

Portabilité

Le code J2EE Java peut être porté sur plusieurs systèmes d'exploitation, mais généralement pas entre les serveurs. La norme permet aux fournisseurs, tels que IBM et BEA Systems, de créer des serveurs d'applications compatibles avec J2EE pour prendre en charge les EJB qui utilisent des fonctionnalités de fournisseur exclusives. Ces applications ne peuvent être transférées entre les serveurs d'applications sans modification. Cependant, il est possible de déplacer l'intégralité du conteneur d'application (une instance du serveur d'application lui-

même) entre les plates-formes, ce qui peut être un grand avantage dans un environnement multi-plateforme grand entreprise. .NET est principalement destiné aux systèmes d'exploitation Windows. Les applications .NET, cependant, ne sont pas directement exécutées dans le code de machine natif. Ils sont compilés dans Microsoft Language Intermediate (MSIL). Le MSIL est compilé en code natif au moment de l'exécution par un compilateur just-in-time et s'exécute dans une machine virtuelle appelée Common Language Runtime (CLR). Cette approche Java n'a jusqu'ici été mise en œuvre que dans Windows, bien qu'il soit possible qu'elle soit prise en charge sur d'autres plates-formes à l'avenir, car ECMA a ratifié C # et CLR en tant que normes officielles de l'industrie en décembre 2001.

Outils, serveurs et accompagnement des fournisseurs

Les fournisseurs J2EE comme IBM, BEA Systems, Oracle, Hewlett-Packard et Sun Microsystems ont déjà commencé à prendre en charge les créations, le déploiement et l'exécution de services Web. Mais le niveau de sophistication et le soutien des normes varient. De nombreux fournisseurs ne s'attendaient pas à ce que XML, la technologie de base pour les services Web, gagne en popularité si vite et ne soit pas préparé à la demande générale du marché. Les fournisseurs de J2EE sont également entravés par les frais généraux et les conflits de créer des normes industrielles pour les services ainsi que de différencier leurs offres de concurrents qui adhèrent à la même norme. J2EE fournit un cadre pour les produits concurrents, sans monopole. C'est un défi, mais pas un nouveau.

Le principal environnement de développement intégré de Microsoft (IDE) est Visual Studio .NET, qui est déjà en avance sur sa concurrence dans le soutien des services Web. Microsoft a avancé pour fournir un support au niveau des composants pour les services Web avec leur suite de serveurs d'activation de service Web, tels que BizTalk. .NET est contrôlé par une seule entreprise et, bien qu'il n'y ait aucune question sur la stabilité, la qualité et l'engagement de Microsoft en matière de services Web, les entreprises devraient explorer et reconnaître les effets d'un engagement stratégique à long terme envers un seul fournisseur.

Maturité et popularité

J2EE est une architecture robuste, évolutive et mature qui supporte depuis des années des applications d'entreprise. La prise en charge des services Web à ce stade n'est qu'une autre fonctionnalité ajoutée à ce modèle de cadre flexible. .NET hérite de nombreuses fonctionnalités de l'architecture d'ADN de Windows. en février 2002, Microsoft a publié des outils et des serveurs de développement clés, de sorte que l'image peut changer radicalement en faveur de .NET. Par exemple, .NET "Mes services" fournit un support plus large de la personnalisation et des préférences des utilisateurs que J2EE.

Faire la bonne décision

Il existe plusieurs facteurs de décision importants qui entrent en jeu lors du choix d'une stratégie .NET ou J2EE. Il s'agit notamment du cadre d'application et de l'infrastructure

informatique existants de l'entreprise ou de l'unité commerciale, du retour sur investissement (ROI) prévu, de la stratégie informatique et commerciale, de l'expertise actuelle des développeurs, de la conformité aux normes, de l'involuntivité et des besoins d'intégration, de la sécurité, de l'agrégation et de la personnalisation, Soutien à la fête.

Cadres existants et infrastructure informatique

Quel est le cadre existant ? De nombreuses entreprises avancent pour étendre les plateformes existantes. Les utilisateurs COM, par exemple, ont tendance à utiliser .NET, alors qu'un magasin WebSphere peut choisir de continuer à utiliser J2EE. Si, comme la plupart des entreprises, votre environnement comprend J2EE et Microsoft, considérez un mélange et prenez la décision projet par projet. À l'aide des normes XML standard des services Web, il est possible d'intégrer (ou au moins l'interface) les applications .NET et J2EE en utilisant des informations de liaison WSDL (Web Services Definition Language) standard et un répertoire UDDI privé (intranet).

Quel cadre peut être facilement pris en charge dans l'infrastructure informatique existante ? La réduction des coûts de soutien entraîne éventuellement un coût total de possession (TCO) moins élevé, car TCO est la somme des coûts de mise en œuvre et de maintenance pendant la durée de vie de l'application.

Expérience de développeur existante

C'est probablement l'un des facteurs les plus importants pour toute entreprise. Si la plupart ou tous les développeurs sont des experts de Microsoft, introduire J2EE est une décision difficile. D'autre part, les développeurs J2EE ont tendance à bien s'adapter à la création d'applications .NET. Les facteurs entourant cette décision soulignent le fait que l'utilisation correcte est souvent plus importante que le choix de la technologie correcte. En d'autres termes, l'utilisation de bien-fondé est un facteur de réussite plus critique que le choix de la «bonne».

Robustesse et sécurité

Quels produits sont plus robustes et évolutifs, et répondent-ils aux besoins d'intégration avec la moindre complexité ? J2EE est plus robuste et évolutif, mais est-ce important pour une entreprise, une division ou un département de petite et moyenne taille ? .NET est moins complexe à développer, à déployer et à entretenir. Cette simplicité elle-même peut augmenter la robustesse et l'évolutivité.

Quels produits ont une plus grande sécurité ? Les deux produits sont bons. .NET est basé sur les rôles, et cela peut être un avantage en raison de l'affinité naturelle des services Web avec RBAC (Role Based Access Control).

Soutien des fournisseurs de logiciels

La plupart des fournisseurs de serveurs d'applications ont commencé à fournir un support pour les services Web, et un soutien complet est inévitable bientôt. Il est hautement probable

qu'une entreprise puisse utiliser son infrastructure d'intégration EAI et B2B actuelle pour déployer des services Web, quel que soit le cadre choisi.

Quels produits sont plus flexibles dans l'intégration des services fournisseurs tiers ? La logique traditionnelle est qu'une architecture ouverte encourage le support de tiers. Cela nous conduirait à conclure que J2EE est supérieur dans ce domaine. Après tout, J2EE est une architecture ouverte. Mais c'est théorique. Le gagnant dans ce domaine dépend des services dont vous avez besoin. Tous les principaux fournisseurs d'applications emballées ont annoncé leur soutien pour les deux cadres. Une leçon que nous avons déjà appris dans les guerres des composants (COM contre J2EE) est qu'il est plus facile pour un fournisseur de supporter un produit (Microsoft) que de supporter de nombreux produits selon une norme (J2EE). Il reste à voir si cela sera vrai pour .NET.

Tous les principaux fournisseurs d'applications emballées sont en course pour soutenir et fournir des interfaces de services Web. Cela comprend la planification des ressources d'entreprise (ERP), la gestion de la chaîne d'approvisionnement (SCM) et la gestion de la relation client (CRM). La tendance actuelle de l'industrie est de rester neutre en soutenant les deux. SAP, par exemple, a annoncé et implémenté des services Web pour J2EE et .NET, en utilisant le propre serveur d'applications Web de SAP. En fin, Les frameworks J2EE et .NET fournissent un support complet pour les applications de services Web multi-niveaux, y compris la conception, le développement, le déploiement, l'exécution et le suivi. Parce que les deux technologies supportent bien les services Web, la guerre-cadre se poursuivra pendant des années. Cette concurrence améliorera la technologie des services Web en termes d'outils, de normes, d'évolutivité et de coûts.

Il n'est pas encore clair, cependant, combien de problèmes importants seront résolus dans les deux camps. Bien que le support de la sécurité, de la gestion opérationnelle, du flux de travail, des règles métier et de l'intégrité transactionnelle soit fourni par les fournisseurs de frameworks J2EE et .NET, les normes dans ces domaines ne sont pas encore définies ou encore complétées. Cela réduira l'interopérabilité dans l'ensemble, car les fournisseurs sont forcés de créer leurs propres solutions «propriétaires» . [3]

2.5 Autres plateformes de développement des services web

2.5.1 Apache Axis2

AXIS est l'acronyme de Apache eXtensible Interaction System. Il s'agit d'un projet développé par Apache. Apache Axis2 est un moteur Web Services / SOAP / WSDL, le successeur de la pile SOAP Apache Axis largement utilisée. Il existe deux implémentations du moteur de service Web Apache Axis2 : Apache Axis2 / Java et Apache Axis2 / C

Pourquoi Apache Axis2 :

Une nouvelle architecture pour Axis2 a été introduite lors du sommet d'août 2004 à Colombo, au Sri Lanka. La nouvelle architecture sur laquelle Axis2 est basé est plus flexible, efficace et configurable par rapport à l'architecture Axis1.x. Certains concepts bien établis d'Axis 1.x, comme les gestionnaires, etc., ont été conservés dans la nouvelle architecture.

Apache Axis2 supporte non seulement SOAP 1.1 et SOAP 1.2, mais il a également un support intégré pour le style REST largement populaire des services Web. La même implémentation de logique commerciale peut offrir à la fois une interface de style WS-* ainsi qu'une interface de style REST / POX(Plain Old XML) simultanément.

Apache Axis2 est plus efficace, plus modulaire et plus orienté XML que l'ancienne version. Il est soigneusement conçu pour supporter l'ajout facile de «modules» de plug-in qui étendent leur fonctionnalité pour des fonctionnalités telles que la sécurité et la fiabilité. Les modules actuellement disponibles ou en développement comprennent :

- WS-Security - pris en charge par Apache Rampart ;
- WS-Addressing -Module inclus dans le noyau Axis2 .

Apache Axis2 est construit sur Apache AXIOM, un nouveau modèle d'objet XML à haut rendement.

Axis2 est doté de nombreuses nouvelles fonctionnalités, améliorations et implémentations de spécifications industrielles. Les principales fonctionnalités proposées sont les suivantes :

- **La vitesse** : Axis2 utilise son propre modèle d'objet et l'analyse StAX (Streaming API for XML) pour atteindre une vitesse nettement supérieure à celle des versions antérieures d'Apache Axis.
- **Impression de pied de mémoire faible** : Axis2 a été conçu pour la mise au point de la base, en gardant un pied de mémoire faible en tête.
- **AXIOM** : Axis2 est livré avec son propre modèle d'objet léger, AXIOM, pour le traitement des messages qui est extensible, très performant et qui est pratique pour le développeur.
- **Déploiement à chaud** : Axis2 est équipé de la possibilité de déployer des services Web et des gestionnaires pendant que le système est opérationnel. En d'autres termes, de nouveaux services peuvent être ajoutés au système sans devoir fermer le serveur. Il suffit de déposer les archives du service Web requises dans le répertoire des services du référentiel et le modèle de déploiement déploiera automatiquement le service et le mettra à disposition.
- **Services Web asynchrones** : Axis2 prend désormais en charge les services Web asynchrones et l'invocation asynchrone de services Web utilisant des clients et des transports non bloquants.
- **Prise en charge du MEP** : Axis2 offre maintenant la souplesse nécessaire pour

prendre en charge les Patterns d'échange de messages (MEP) avec un support intégré pour les MEP de base définis dans WSDL 2.0.

- **Flexibilité** : L'architecture Axis2 offre au développeur une liberté totale pour insérer des extensions dans le moteur pour le traitement d'en-tête personnalisé, la gestion du système et tout ce que vous pouvez imaginer
- **Stabilité** : Axis2 définit un ensemble d'interfaces publiées qui changent relativement lentement par rapport au reste de l'Axe.
- **Déploiement orienté composant** : Vous pouvez facilement définir des réseaux réutilisables de gestionnaires pour implémenter des modèles communs de traitement pour vos applications ou distribuer à des partenaires.
- **Cadre de transport** : Nous avons une abstraction propre et simple pour intégrer et utiliser Transports (c'est-à-dire les expéditeurs et les auditeurs pour SOAP sur différents protocoles tels que SMTP, FTP, middleware orienté message, etc.) et le cœur du moteur est complètement transporté, indépendant.
- **Prise en charge de WSDL** : Axis2 prend en charge le Web Service Description Language, version 1.1 et 2.0, qui vous permet de construire facilement des stubs pour accéder à des services distants et d'exporter automatiquement des descriptions de vos services déployés à partir d'Axis2.
- **Composition et extensibilité** : Les modules et les phases améliorent le support de la composabilité et de l'extensibilité. Les modules supportent la composabilité et peuvent également prendre en charge les nouvelles spécifications WS-* de manière simple et propre. Ils ne sont cependant pas déployables à chaud car ils modifient le comportement global du système.

2.5.2 Sun ONE Developer Studio de Sun :

L'architecture Sun ONE est la vision, l'architecture, la plate-forme et l'expertise de logiciels de Sun pour résoudre l'intégration d'entreprise, l'interopérabilité et les problèmes de développement d'aujourd'hui. Basé sur la plate-forme Java 2, Enterprise Edition (J2EE), il offre une évolution simple des applications d'entreprise non intégrées à des services Web entièrement intégrés et interopérables.

Un développeur de logiciels trouvera utile d'apprendre l'architecture Sun ONE et de découvrir comment résoudre les problèmes d'intégration et d'interopérabilité. Ses technologies et produits associés fournissent une large gamme de services automatisés qui réduisent considérablement les coûts de construction d'applications Web et de clients.

Description :

La plate-forme Sun ONE est construite sur une pile de services présentée sur la figure 2.6 qui sont basés sur un nombre important de standards d'API et de protocoles dont les principaux

sont résumés sur la figure 2.7.

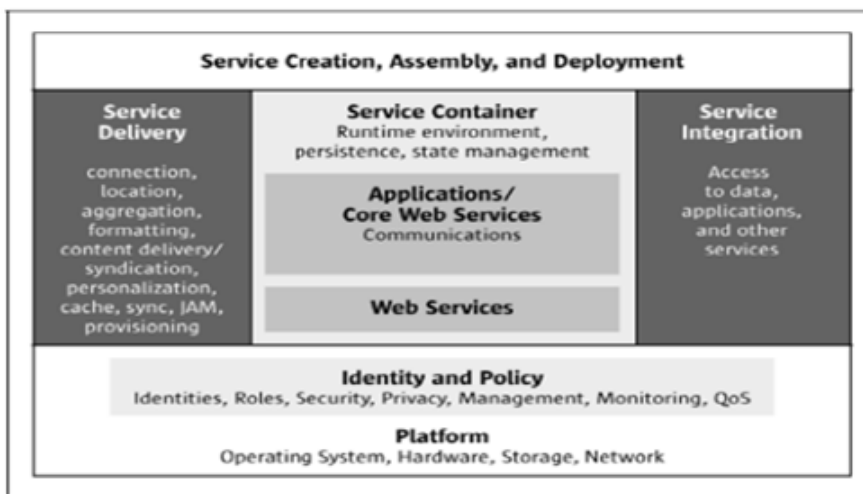


FIGURE 2.6 – Pile des services de Sun ONE

Sur la figure 2.6, le système d'exploitation et les ressources matérielles se trouvent au fond de la pile. Sur cette base, on retrouve dans la partie centrale les trois couches habituelles dans une application (présentation logique (service délivre) business logique (service container) data Access logique (service intégration)).

Dans Sun ONE, il n'y a pas une unique méthode d'utilisation des outils disponibles pour implémenter ces différents services

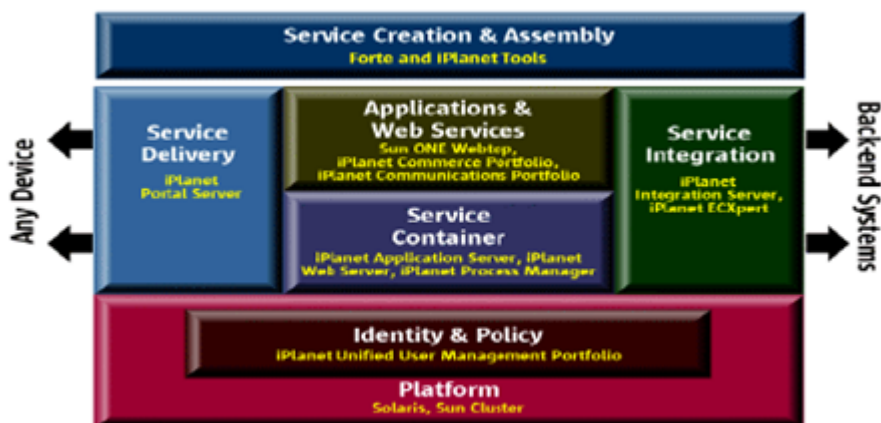


FIGURE 2.7 – Architecture de Sun ONE

Concernant l'interopérabilité de Sun ONE avec des systèmes existants ou d'autres Plateformes telles que Microsoft .NET, elle est basée entre autre sur les standards d'interfaces

définis pour le Web. Dans cette optique, Sun ONE propose ces outils pour interagir avec : des systèmes existants : les Connectors et CORBA(Common Object Request Broker Architecture). Microsoft .NET : les services Web construits à l'aide des standards SOAP, UDDI et WSDL.

Sun ONE Studio :

Sun ONE Studio n'est pas un outil monolithique, mais plutôt une architecture permettant aux outils plus spécialisés de Sun ONE d'y être intègres. Sun ONE Studio permet aussi bien de créer des composants basés sur la technologie J2EE que des services Web (en utilisant les servlets et les EJB) qui peuvent être ensuite assembles en des applications et déployés dans le serveur d'application de Sun ONE. Sun ONE Studio crée et publie également ces services en utilisant WSDL et UDDI.[13]

Sun ONE Application Framework :

Sun ONE Application Framework simplifie le processus de développement d'applications. Elle est basée sur l'expérience de développeurs de logiciels, de consultants et de développeurs d'applications Web. Elle est principalement destinée aux développeurs d'applications Web de moyenne a grande taille. Elle utilise les meilleurs design patterns disponibles, tels que MVC (Model-View-Controller), Business Delegate, ierarchical View, etc... Sun ONE Application Framework se présente sous forme d'un module a ajouter a Sun ONE Studio. Ensuite, il est possible de créer un template pour une application Web consistant en un ensemble de modules comprenant une vue (view) et un modèle (model).

2.6 Conclusion

Ce chapitre nous a permis de présenter les deux plateformes les plus répondues dans le développement des applications orientées services web, a savoir .NET de Microsoft et J2EE de Sun.

Une comparaison nous a permis de constater que chaque plateforme possède ses points forts et ses point faibles, ainsi nous avons décidé de combiner entre les deux plateformes dans le développement de notre application.

Modélisation

3.1 Introduction

Dans ce chapitre nous présentons les détails de notre conception et Pour mener à bien le projet, nous devons tout naturellement avoir recours à un formalisme de conception à savoir UML « Unified Modeling Language ».

Dans notre sujet de fin d'étude,nous avons pris comme étude de cas la gestion d'une agence de voyage qui est un exemple classique dans le domaine du E-Tourisme.

3.2 Modélisation des services

3.2.1 Diagrammes de cas d'utilisations

Pour la conception des services de l'agence de voyage, nous avons distingué trois catégories de services web :

Catégorie " Compagnie aérienne " : Cette catégorie propose des services chargés de la gestion en ligne des réservations de vol des clients ce qui est mentionné dans le diagramme suivant :

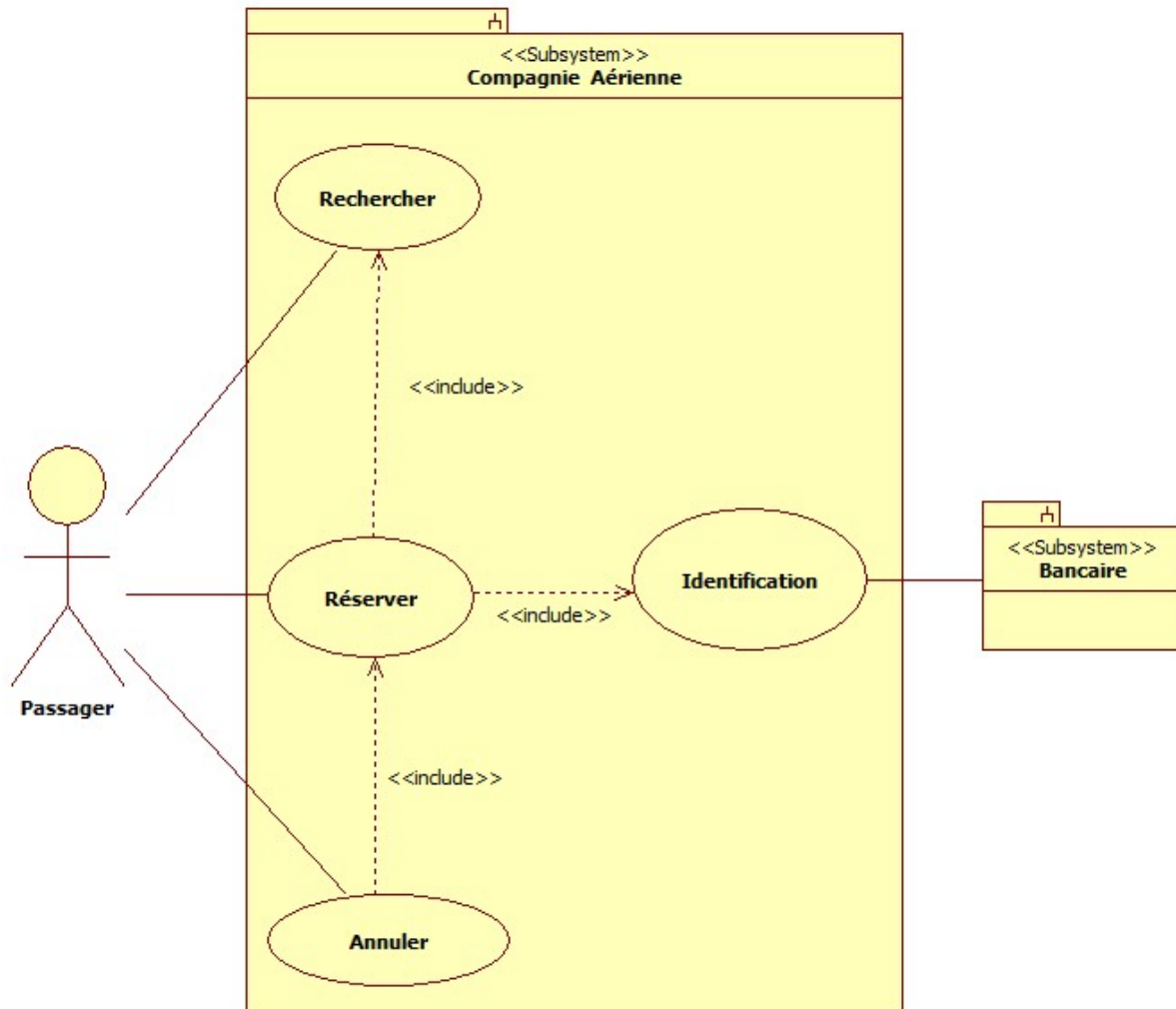


FIGURE 3.1 – diagramme de cas d'utilisation de la compagnie aérienne

Catégorie " Compagnie hôtelière " : Elle offre des services qui ont comme fonction, la gestion en ligne des hôtels et de réservation des clients.

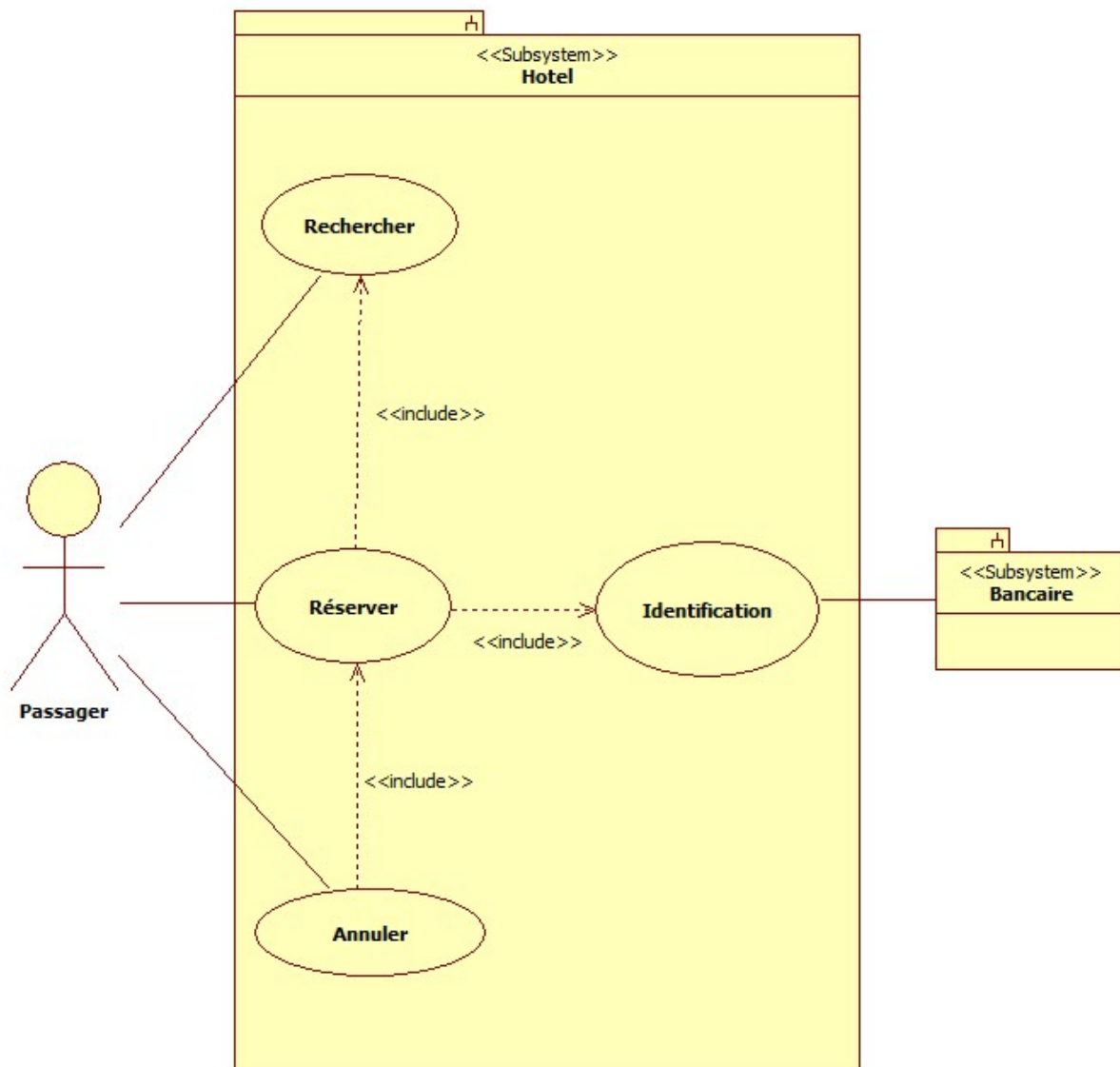


FIGURE 3.2 – diagramme de cas d'utilisation de l'hôtel

Catégorie " Compagnie Location voiture " :Nous avons classé dans cette catégorie tous les services chargés de la gestion en ligne des voitures et de location de voiture.

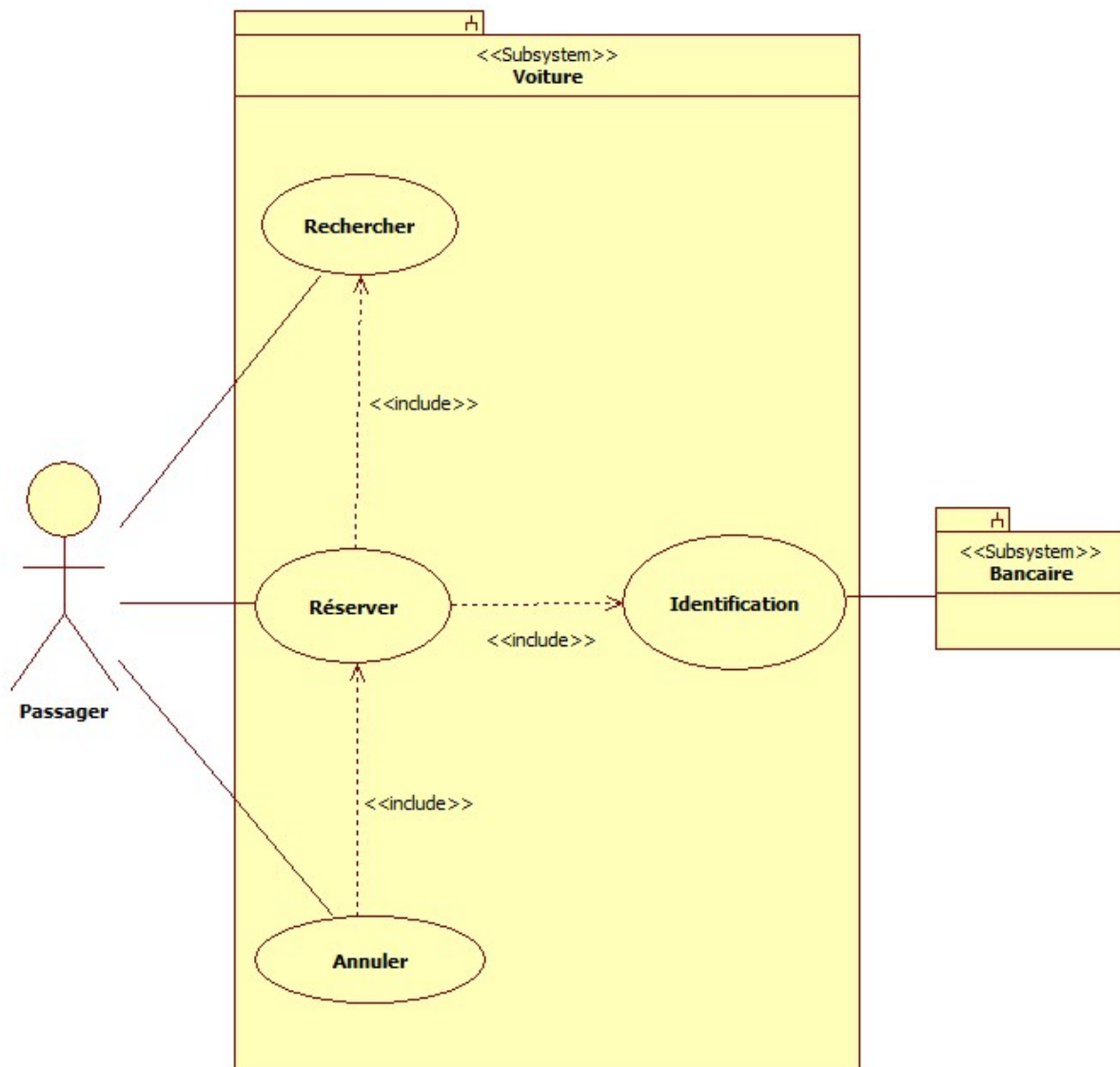


FIGURE 3.3 – diagramme de cas d'utilisation de la voiture

Finalement le diagramme de cas d'utilisation globale du système se présente comme suite :

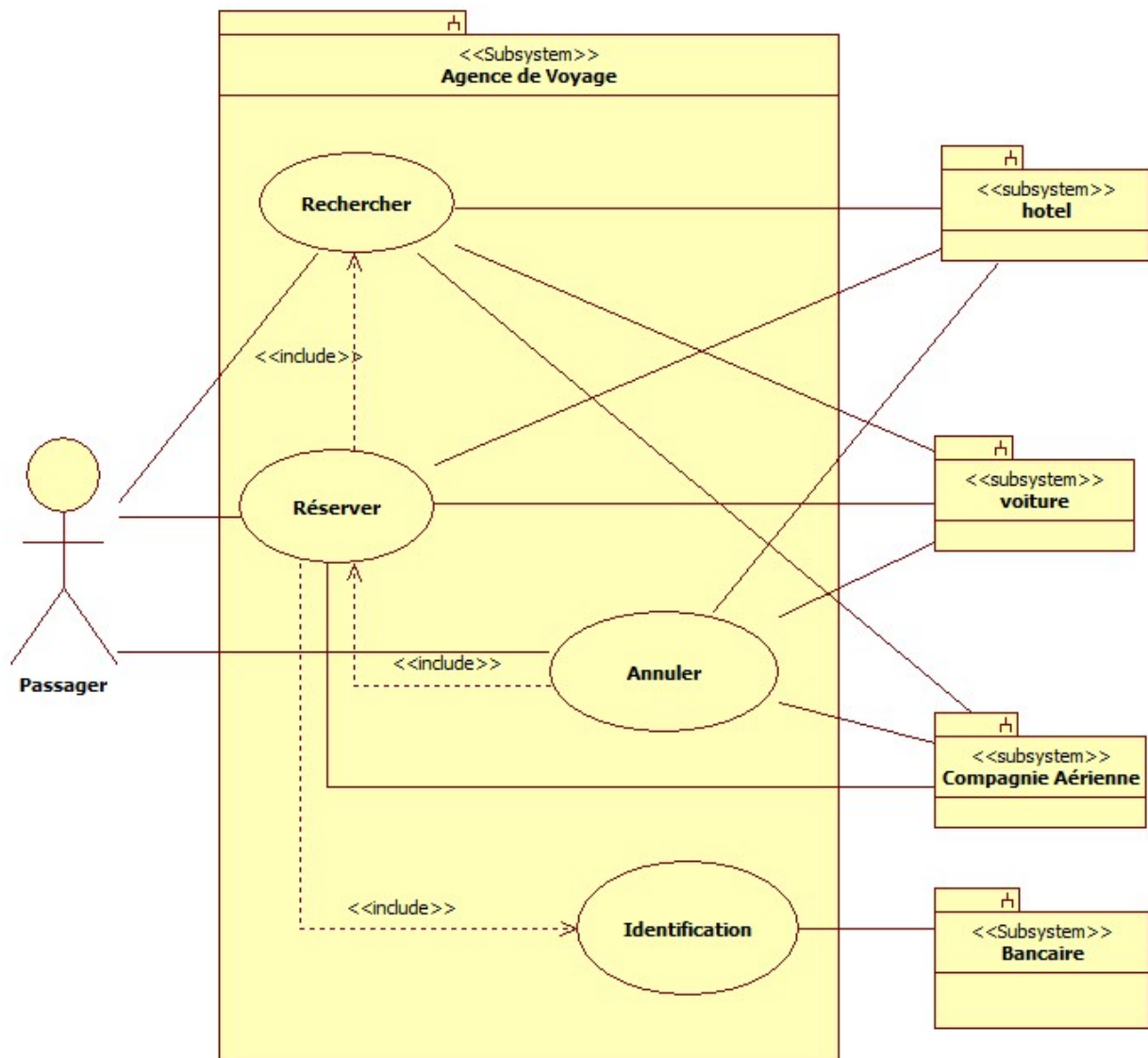


FIGURE 3.4 – diagramme de cas d'utilisation Global de l'agence de voyage

3.2.2 diagrammes de séquence

Pour mieux comprendre les scénarios des cas d'utilisation, nous décrivons graphiquement chaqu'un par un diagramme de séquence.

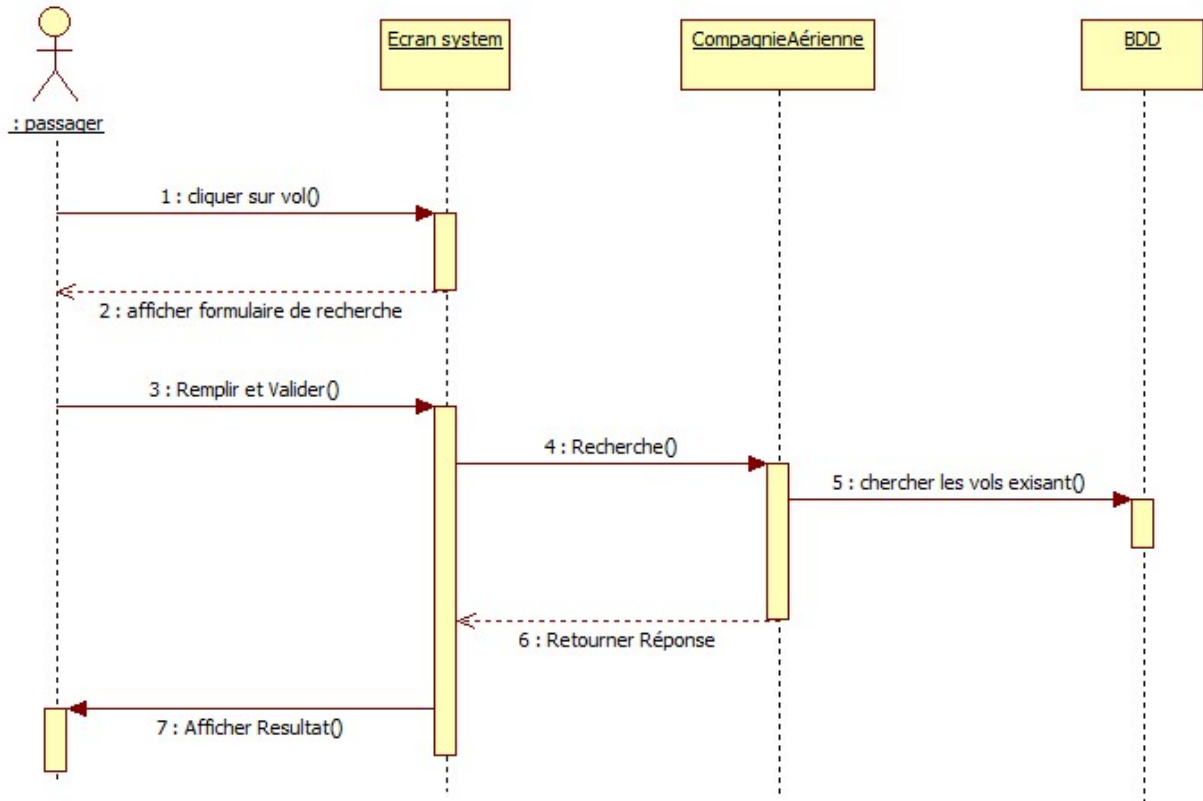


FIGURE 3.5 – diagramme de séquence pour la recherche du vol

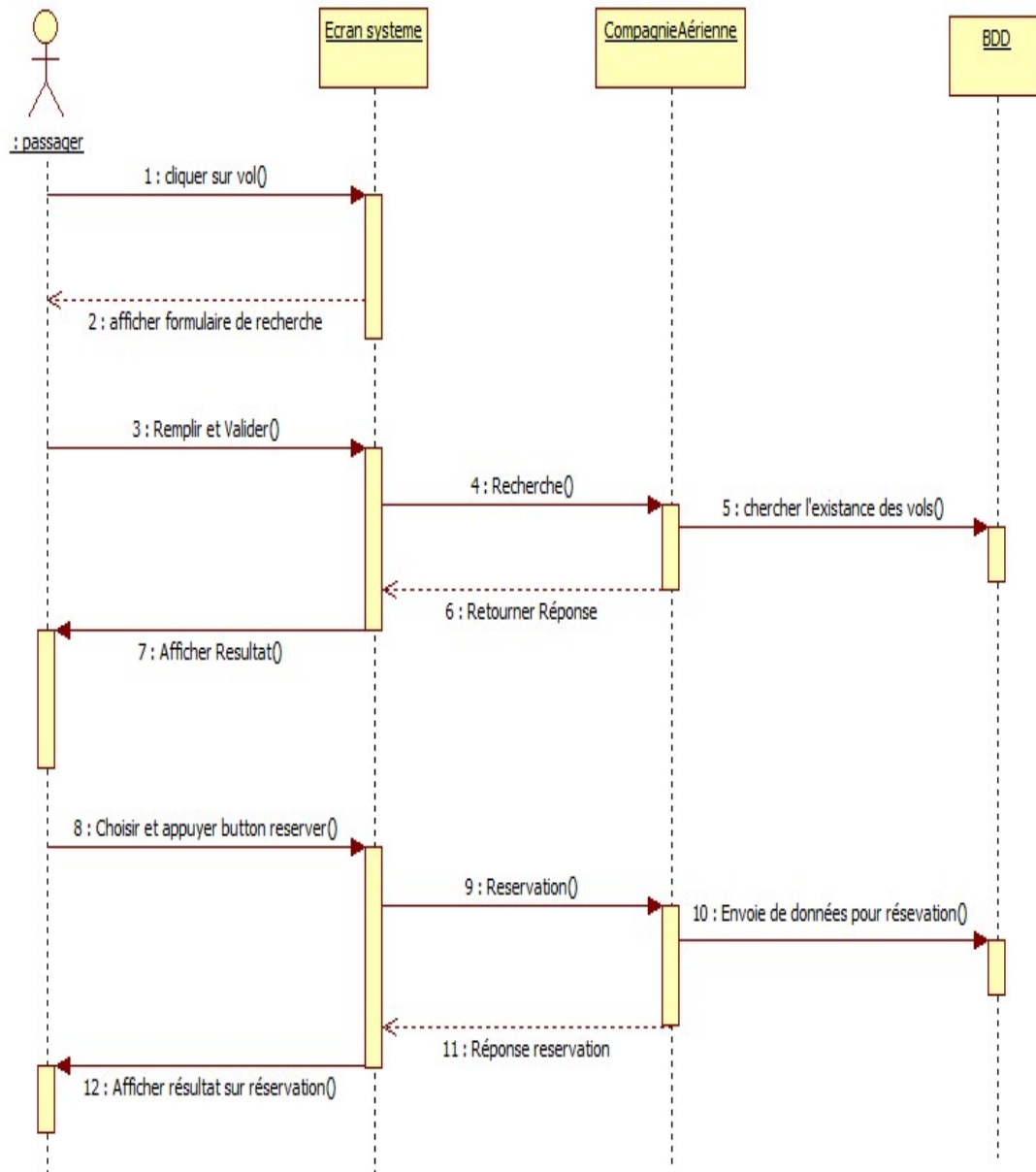


FIGURE 3.6 – diagramme de sequence pour la réservation du vol

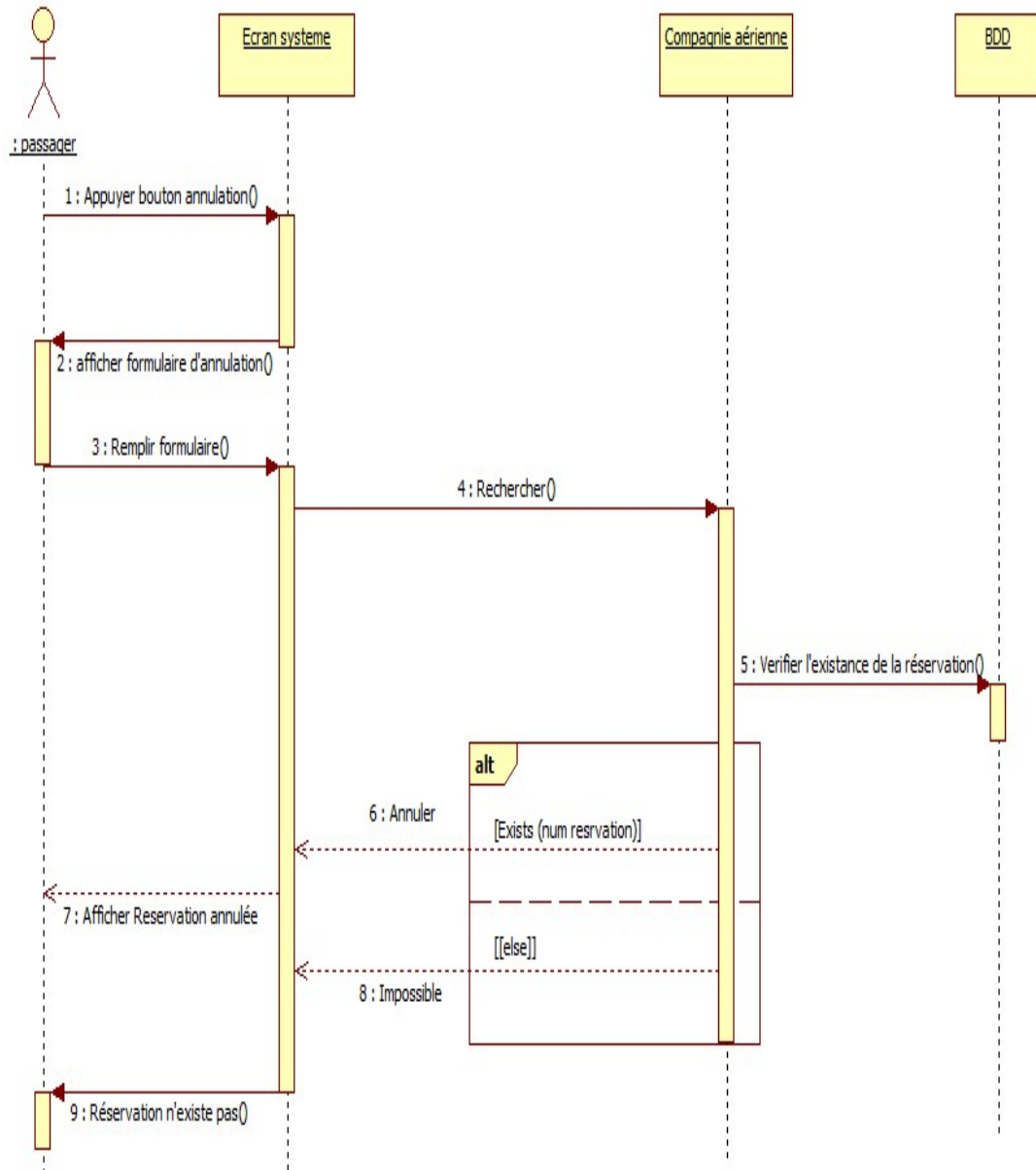


FIGURE 3.7 – diagramme de sequence pour l’annulation d’une réservation du vol

3.2.3 Diagramme de classes

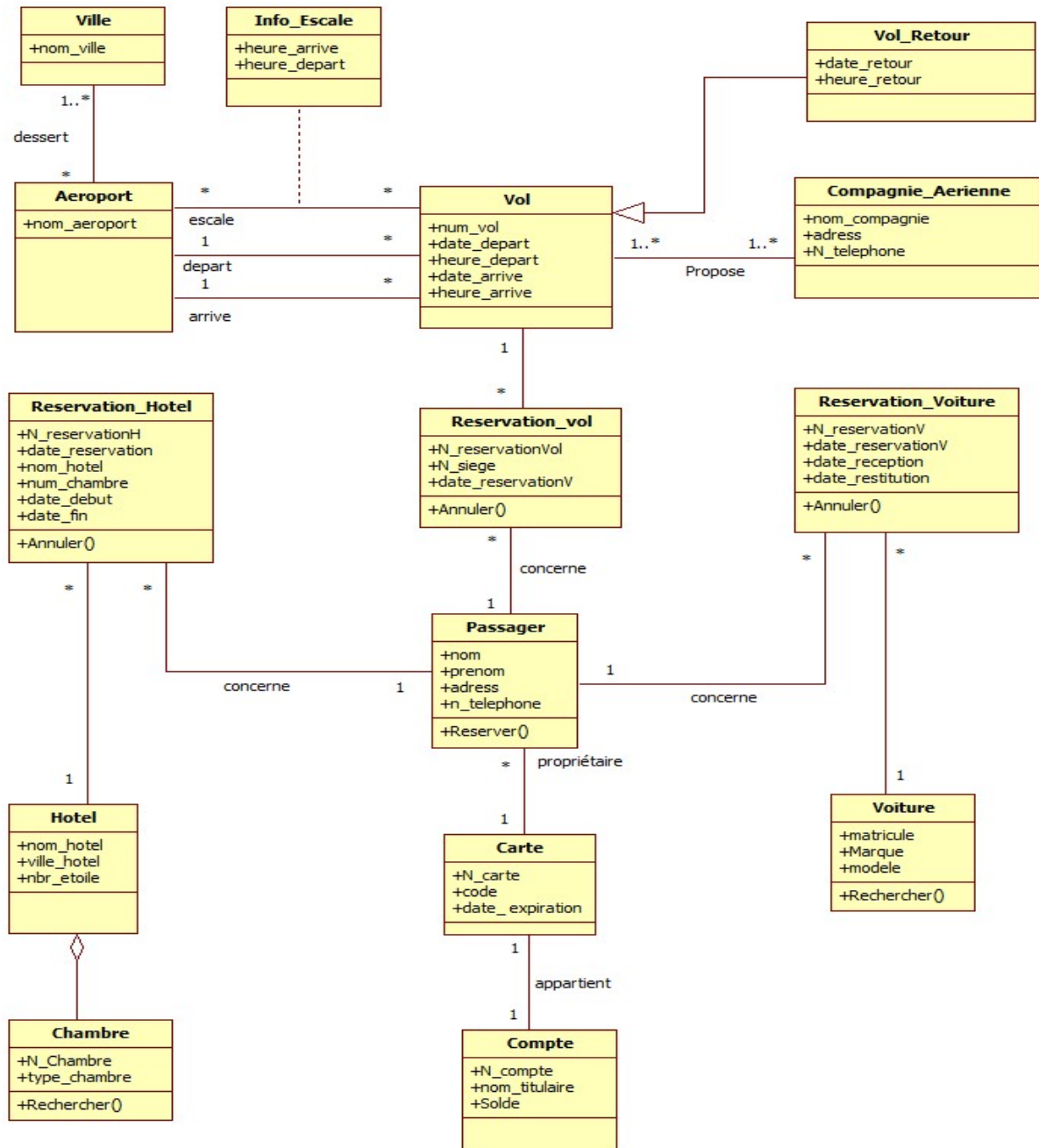


FIGURE 3.8 – diagramme de classes

3.3 Conclusion

Dans ce chapitre nous avons réalisé la conception de notre système, agence de voyage, où nous avons réalisé les diagrammes de cas d'utilisation, les diagrammes de séquences et le diagramme de classes. On se basant sur cette conception, le chapitre suivant sera consacré à l'implémentation d'une application à base des services web pour la réservation en ligne des différentes services fournit par notre système .

La réalisation

4.1 Introduction

Dans ce chapitre nous présenterons les différents outils exploités dans la réalisation des différents services web et les différentes interfaces de la mise en œuvre.

Dans un premier temps, nous avons développé les services des différents sous systèmes nécessaires à la gestion de l'agence de voyage à savoir : les services réservation de chambre d'hôtel, réservation de vol et réservation de voiture. Nous avons ensuite construit un portail qui réunit tous les services web distants, ce portail a pour objectif de permettre aux clients de préparer et d'organiser leurs voyages de façon très conviviale.

4.2 Architecture globale du system :

Description des étapes mentionnées sur la figure 4.1

1. Le client lance une recherche.
2. Interaction entre le serveur web de l'agence de voyage et le SGBDR de l'UDDI.
3. Interaction entre le serveur UDDI et sa base de donnée.
4. Le client reçoit la réponse de sa recherche.
5. Le client sélectionne et valide son choix .
6. L'agence de voyage récupère les services pour les afficher au client.
7. Le client reçoit les services demandés.

L'architecture de notre système se présente comme suit :

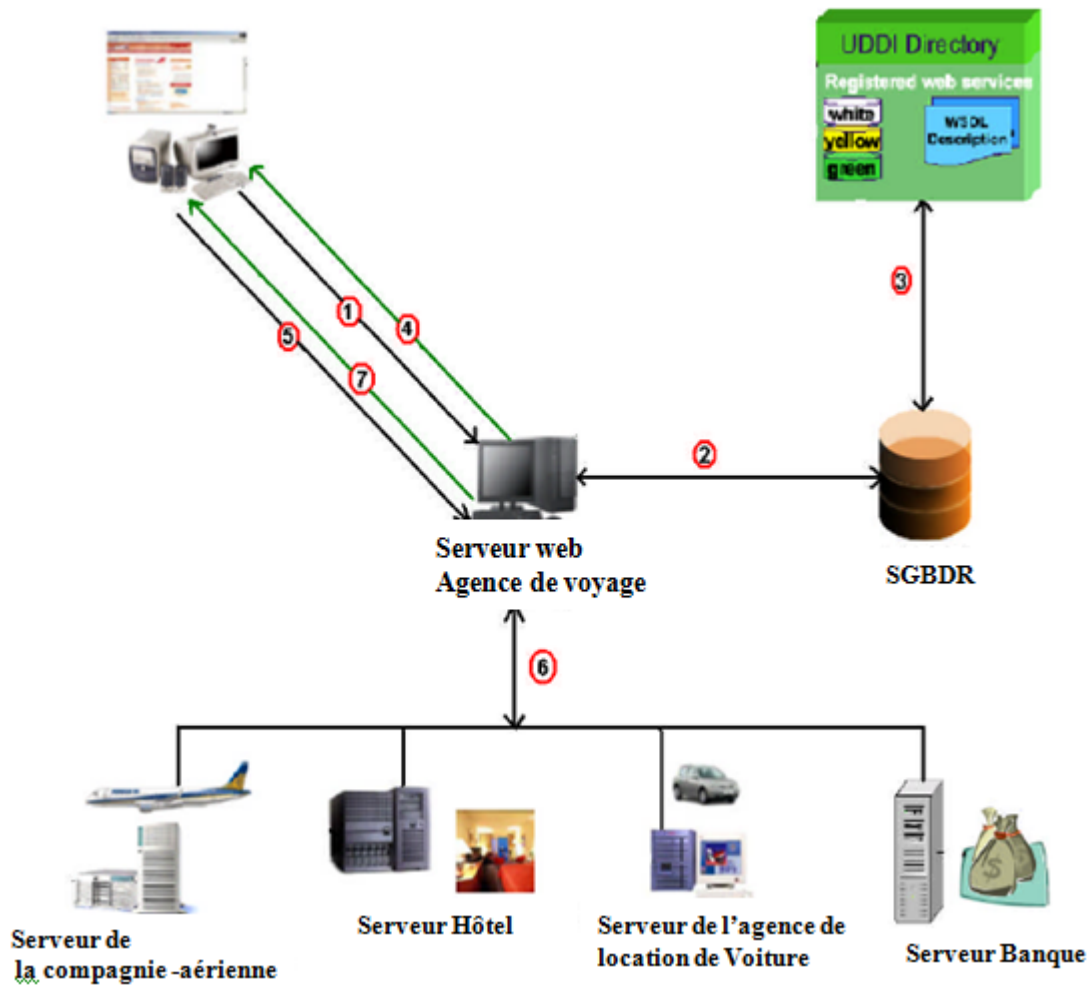


FIGURE 4.1 – Architecture globale du système

4.3 Les outils utilisés dans le développement de différents services

L'ensemble des besoins technique a été évalué sur les bases d'une logique open-source, ou ,dans la configuration la moins favorable dans une logique de gratuite. Les solutions multi plat formes ont été aussi privilégiées afin de pouvoir facilement installer sur d'autres machines certains éléments clés de l'architecture logicielle, qui est une architecture distribuée.



L'environnement de développement Eclipse : Eclipse est un environnement

de développement intégré dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques. Eclipse, de part le fort soutien de la communauté open source et à la base crée sur une initiative d'IBM, s'est imposé comme un environnement de développement multi-langages de première ordre. Sa capacité à être

étendu à travers des plugins en fait un outil de choix lors d'un développement demandant l'utilisation et l'intégration de composants logiciels hétérogènes. La version utilisée pendant l'ensemble du développement a été la version Eclipse 4.6.



Java EE : Java Platform, Enterprise Edition, ou Java EE (anciennement J2EE), est une spécification pour la plate-forme Java d'Oracle, destinée aux applications d'entreprise. La plate-forme étend Java Platform, Standard Edition (Java SE) en fournissant une API de mapping objet-relationnel, des architectures distribuées et multitières, et des services web. La plate-forme se fonde principalement sur des composants modulaires exécutés sur un serveur d'applications. Alors que Java SE constitue le framework de référence pour Java (avec des bibliothèques standards répondant à la plupart des besoins), Java EE complète ce framework avec des bibliothèques logicielles additionnelles dédiées à des applications professionnelles, facilitant par exemple le développement d'applications pour architecture distribuée. la version utilisée est Java 8 Update 111.



Apache Tomcat : est un conteneur web libre de servlets et JSP Java EE. Issu du projet Jakarta, c'est un des nombreux projets de l'Apache Software Foundation. Il inclut des outils pour la configuration et la gestion. Il comporte également un serveur HTTP. Tomcat a été écrit en langage Java. Il peut donc s'exécuter via la machine virtuelle Java sur n'importe quel système d'exploitation la supportant.) la version utilisée Apache Tomcat 7.0.77.



NetBeans : est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL¹ et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). la version utiliser est Netbeans 8.0.2².



GlassFish : est le nom du 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 Server4 (anciennement Sun Java System Application Server5 de Sun Microsystems). C'est la réponse aux développeurs Java désireux d'accéder aux sources et de contribuer au développement des serveurs d'applications de nouvelle génération. La distribution dite Open Source Edition est placée sous double licence CDDL et GPLv2. GlassFish est certifié Java EE 5 (EJB3 + JPA +

1. Common Development and Distribution License

2. Téléchargeable au <https://netbeans.org/downloads/8.0.2/>

JSF + JAX-WS 2.x + ...) et Java EE 6 (EJB 3.1, CDI, JSF 2.0, JAX-RS 1.1, ...), la version utilisée est GlassFish 4.1.1.



Java Server Pages (JSP) : est une technologie de programmation côté serveur

qui permet la création d'une méthode dynamique et indépendante de la plate-forme pour créer des applications Web. JSP a accès à toute la famille d'API Java, y compris l'API JDBC pour accéder aux bases de données.



WampServer (anciennement WAMP5) : est une plateforme de développement

Web de type WAMP, permettant de faire fonctionner localement (sans avoir à se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant deux serveurs (Apache et MySQL), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL. Il dispose d'une interface d'administration permettant de gérer et d'administrer ses serveurs au travers d'un tray icon (icône près de l'horloge de Windows). La version utilisée est wampserver 2.4.



PHP (Hypertext Preprocessor) : est un langage de programmation libre,

principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet. Il est considéré comme une des bases de la création de sites web dits dynamiques mais également des applications web.



SoapUI : est une application open source permettant le test de web service dans

une architecture orientée services (SOA). Ses fonctionnalités incluent l'inspection des web service, l'invocation, le développement, la simulation, le mocking, les tests fonctionnels, les tests de charge et de conformité. Il est entièrement basé sur la plate-forme Java et utilise Swing pour l'interface utilisateur. Ce qui signifie que SoapUI est multiplateforme. SoapUI supporte aujourd'hui IDEA, Eclipse et NetBeans.



L' API Java pour les services Web XML (JAX-WS) est une API de lan-

gage de programmation Java pour créer des services Web , en particulier des services SOAP . JAX-WS est l'une des API de programmation Java XML . Cela fait partie de la plate-forme Java EE .



HTML (L'HyperText Markup Language) : est le format de données conçu

pour représenter les pages web. C'est un langage de balisage permettant d'écrire de l'hyper-texte, d'où son nom. HTML permet également de structurer sémantiquement et logiquement

et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Il est souvent utilisé conjointement avec le langage de programmation JavaScript et des feuilles de style en cascade (CSS).



Les CSS, Cascading Style Sheets (feuilles de styles en cascade) : servent à mettre en forme des documents web, type page HTML ou XML. Par l'intermédiaire de propriétés d'apparence (couleurs, bordures, polices, etc.) et de placement (largeur, hauteur, côte à côte, dessus-dessous, etc.), le rendu d'une page web peut être intégralement modifié sans aucun code supplémentaire dans la page web. Les feuilles de styles ont d'ailleurs pour objectif principal de dissocier le contenu de la page de son apparence visuelle.

4.4 Présentation des services web

Pour la conception des services de l'agence de voyage, nous avons distingué trois types de services web :

Service de la compagnie aérienne : Il propose des services chargés de la gestion en ligne des réservations de vol. Le service est programmé avec Java sur la plate-forme J2EE avec l'environnement de développement Netbeans.

Service de Location voiture : Il classe tous les services chargés de la gestion en ligne des voitures et de location de voiture. Le service est programmé avec Java sur la plate-forme J2EE avec l'environnement de développement Eclipse.

Service d'hôtelière : Il offre des services qui ont comme fonction, la gestion en ligne des hôtels et de réservation des clients. Le service est programmé le langage PHP.

pour développer chaque service le travail à réaliser se devise en trois étapes :

- Créer le service web : consiste à créer la partie serveur d'un web service à l'aide d'un environnement de développement tel que NetBeans.
- Déployer le web service sur un serveur d'application.
- Créer un client pour utiliser le web service dans notre système nous avons utilisé le langage JSP pour développer la partie client de chaque service .

4.5 Description de l'application (présentation des différentes interfaces de l'application)

Nous avons créé un portail de services qui sert de point d'accès aux différents services web. Ce portail ne stocke aucune donnée sur sa base physique, mais agit comme un fournisseur de services à valeur ajoutée. L'objectif est de donner l'impression à l'utilisateur qu'il est sur un site web classique et que tout lui vient du serveur de ce site, alors qu'il s'agit d'un ensemble de services web distants fournis par différents fournisseurs. Voici une capture d'écran de la page d'accueil du portail.

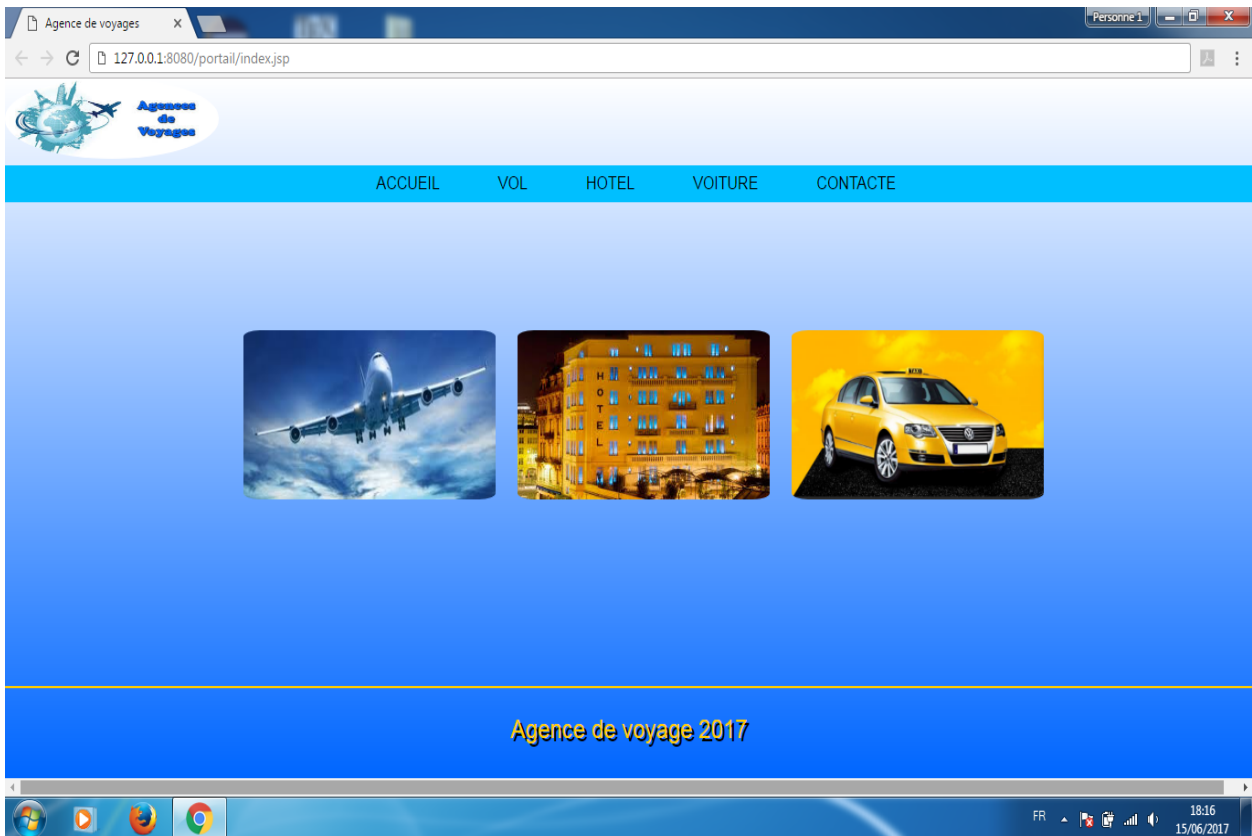


FIGURE 4.2 – Page d'accueil de l'agence de voyage

Après la sélection de service vol :

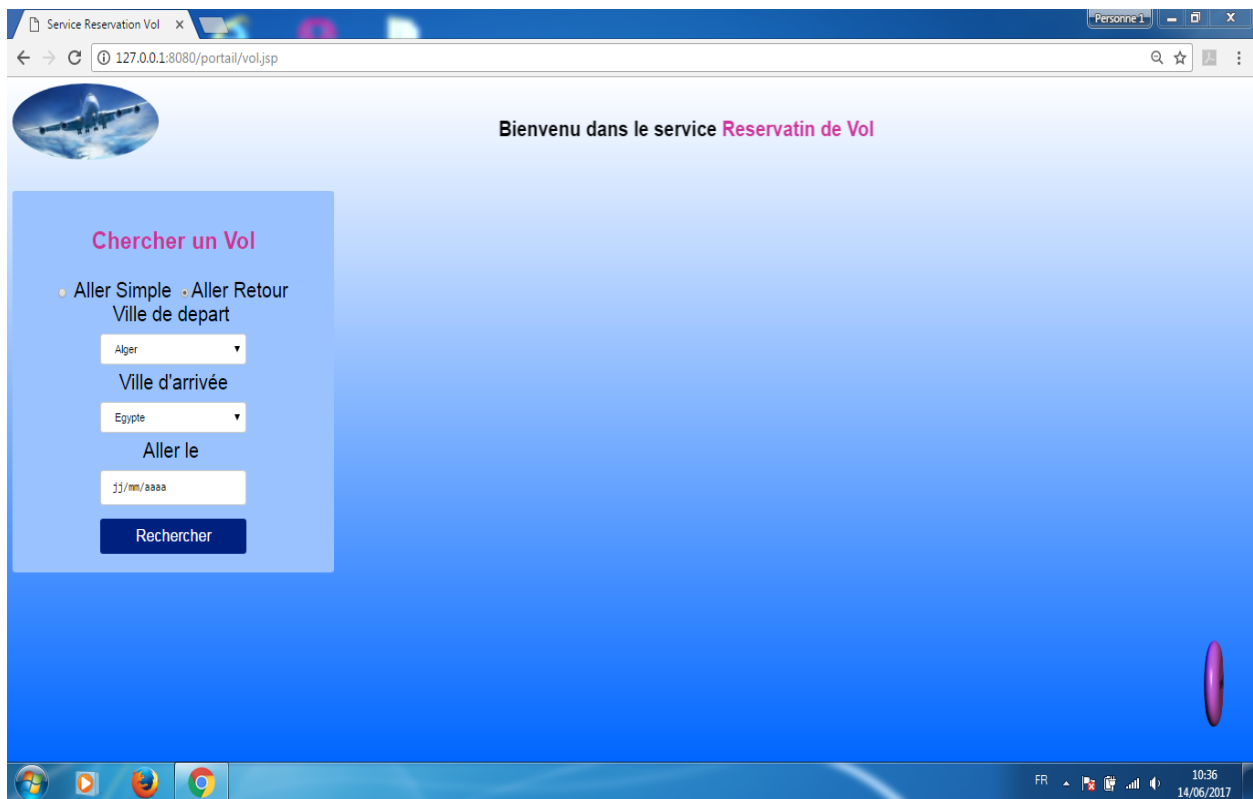


FIGURE 4.3 – La réservation d'un vol

Après une opération de recherche de vol :



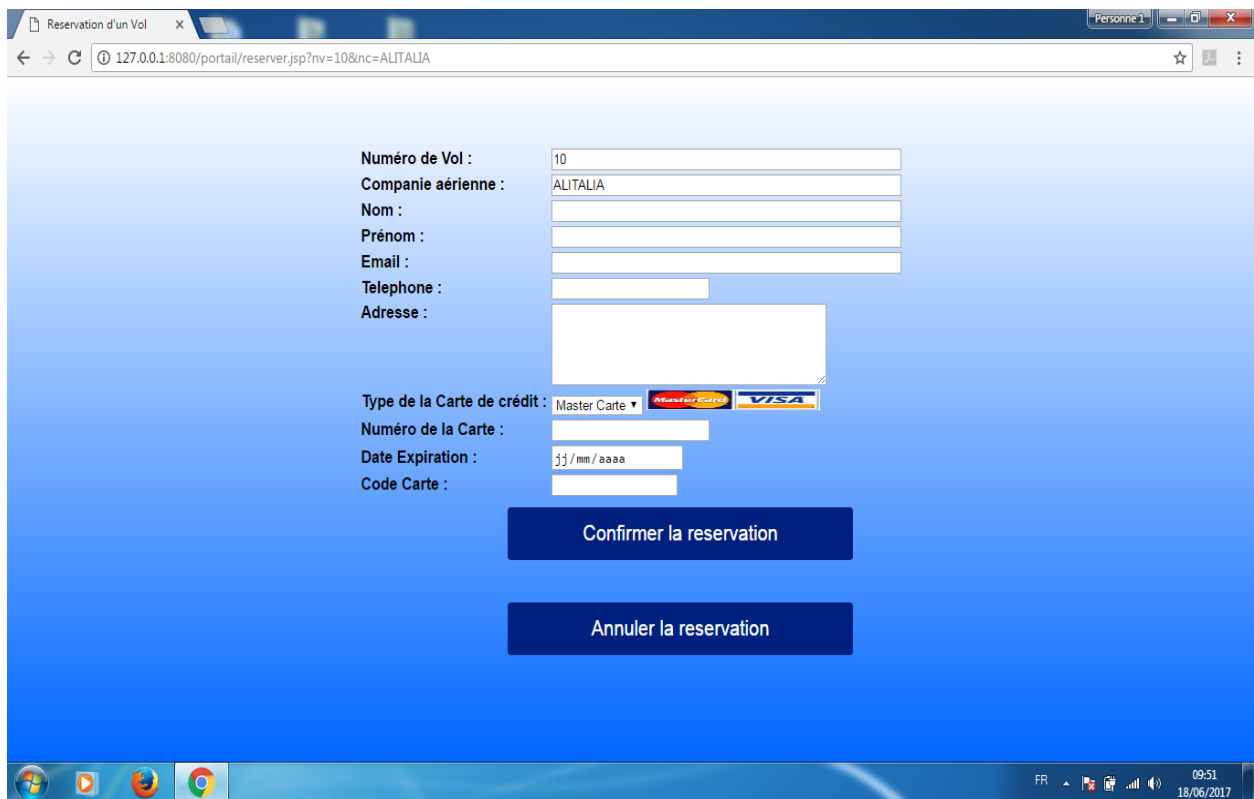
The screenshot shows a web browser window with the URL `127.0.0.1:8080/portail/vol.jsp`. The page title is "Service Reservation Vol". The main heading is "Bienvenu dans le service Reservatin de Vol". On the left, there is a search form titled "Chercher un Vol" with options for "Aller Simple" and "Aller Retour". The search criteria are: "Ville de depart" (Alger), "Ville d'arrivée" (Paris), and "Aller le" (05/07/2017). A "Rechercher" button is at the bottom of the form. On the right, a table displays the search results:

N_Vol	Heure Départ	Company	Selectionner
6	10:30	Air France	Reserver
7	15:30	Air Algerie	Reserver
8	07:00	British Airways	Reserver
10	12:30	ALITALIA	Reserver

Below the table, it indicates "Nombre de vols = 4". A "Retour" button is located in the bottom right corner of the page content area. The Windows taskbar at the bottom shows the date and time as 11:59 on 14/06/2017.

FIGURE 4.4 – Résultat d’une opération de recherche de vol

Après une opération de sélection d'un vol pour la réservation :



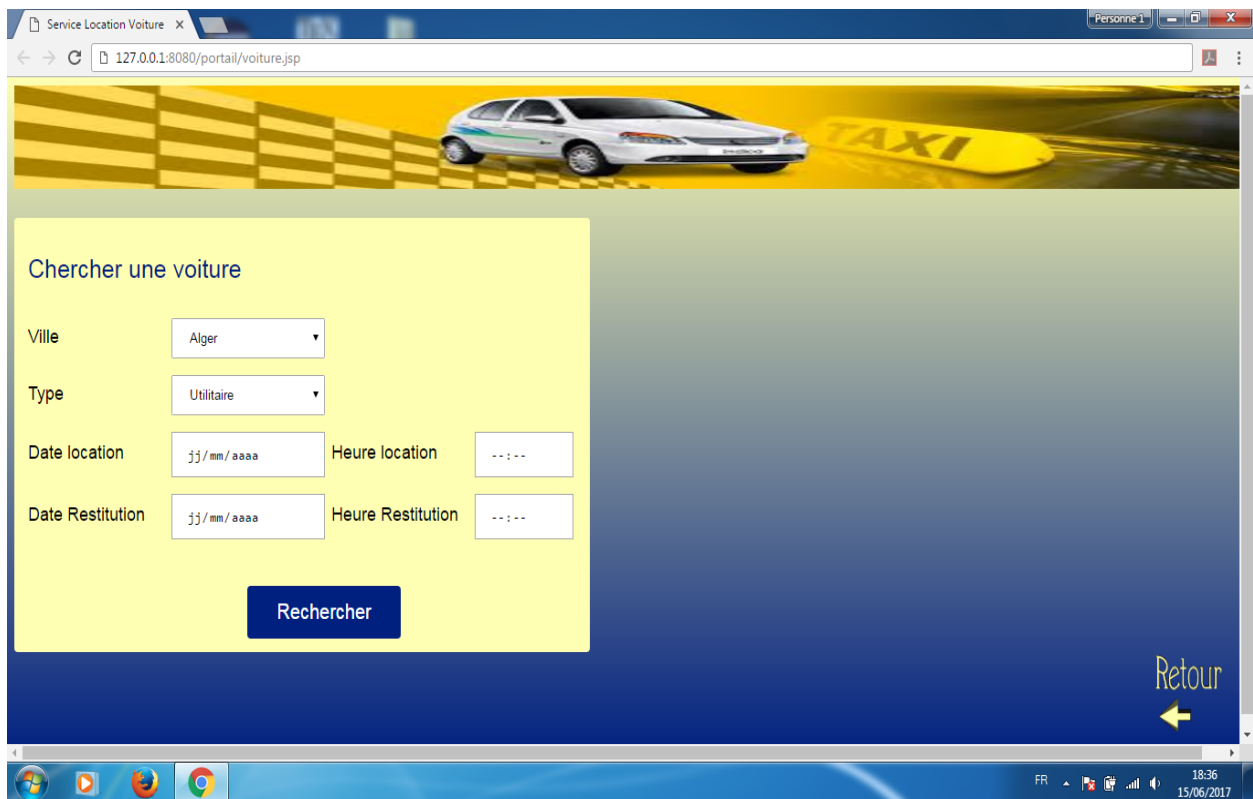
The image shows a web browser window with the title "Reservation d'un Vol". The address bar contains the URL "127.0.0.1:8080/portail/reserver.jsp?nv=10&nc=ALITALIA". The form is set against a blue gradient background and contains the following fields and controls:

- Numéro de Vol :** Input field containing "10".
- Compagnie aérienne :** Input field containing "ALITALIA".
- Nom :** Empty input field.
- Prénom :** Empty input field.
- Email :** Empty input field.
- Telephone :** Empty input field.
- Adresse :** Large empty text area.
- Type de la Carte de crédit :** Dropdown menu set to "Master Carte", with "Master Card" and "VISA" icons displayed.
- Numéro de la Carte :** Empty input field.
- Date Expiration :** Input field with the placeholder "jj/mm/aaaa".
- Code Carte :** Empty input field.

At the bottom of the form are two dark blue buttons: "Confirmer la reservation" and "Annuler la reservation". The Windows taskbar at the bottom shows the system tray with the date "18/06/2017" and time "09:51".

FIGURE 4.5 – formulaire de réservation d'un vol

Après la sélection du service voiture :



The screenshot shows a web browser window with the title "Service Location Voiture" and the URL "127.0.0.1:8080/portail/voiture.jsp". The page features a yellow header with a car and a "TAXI" sign. Below the header is a search form titled "Chercher une voiture". The form includes the following fields:

- Ville: A dropdown menu with "Alger" selected.
- Type: A dropdown menu with "Utilitaire" selected.
- Date location: A date input field with the placeholder "jj/mm/aaaa".
- Heure location: A time input field with the placeholder "--:--".
- Date Restitution: A date input field with the placeholder "jj/mm/aaaa".
- Heure Restitution: A time input field with the placeholder "--:--".

A blue "Rechercher" button is located below the form. In the bottom right corner of the page, there is a "Retour" link with a left-pointing arrow. The Windows taskbar at the bottom shows the system tray with the date "15/06/2017" and time "18:36".

FIGURE 4.6 – La réservation d’une voiture

Après une opération de recherche d'une voiture :

Service Location Voiture x

127.0.0.1:8080/portail/voiture.jsp

Chercher une voiture

Ville:

Type:

Date location: Heure location:

Date Restitution: Heure Restitution:

Matricule_Voiture	Marque	Prix	Type	Selectionner
555	hundai accent	4000 DA	touristique	Reserver
1233	pegeot 207	3500 DA	touristique	Reserver
12333	toyota yaris	5000 DA	touristique	Reserver
56987	Hundai Tuc son	9500 DA	touristique	Reserver
587922	VolSwaGen Arteon	1200 DA	touristique	Reserver

Nombre de voiture = 5

[Retour](#)

Haut-parleurs / Casque : 41%

FR 09:40 19/06/2017

FIGURE 4.7 – Résultat d'une opération de recherche de voiture

Après une opération de sélection d'une voiture pour la réservation :

Reservation d'une voiture: X

127.0.0.1:8080/portail/reserver_voiture.jsp?nv=12333&dl=2017-07-24&hl=12:30&dr=2017-07-24&hr=11:20&vi=Alger

Personne1

Matricule : 12333

Ville : Alger

Date Reservation : 2017-07-24

Heure Reservation : 12:30



Date Restitution : 2017-07-24

Heure Restitution : 11:20

Nom : _____

Prénom : _____

Telephone : _____

Type de la Carte de crédit : Master Carte  

Numéro de la carte : _____

Code Carte : _____

Date Expiration : jj/mm/aaaa

Confirmer la demande de reservation

Annuler la demande de reservation

FR 09:39 18/06/2017

FIGURE 4.8 – formulaire de réservation d'une voiture

Après la sélection du service hôtellerie :

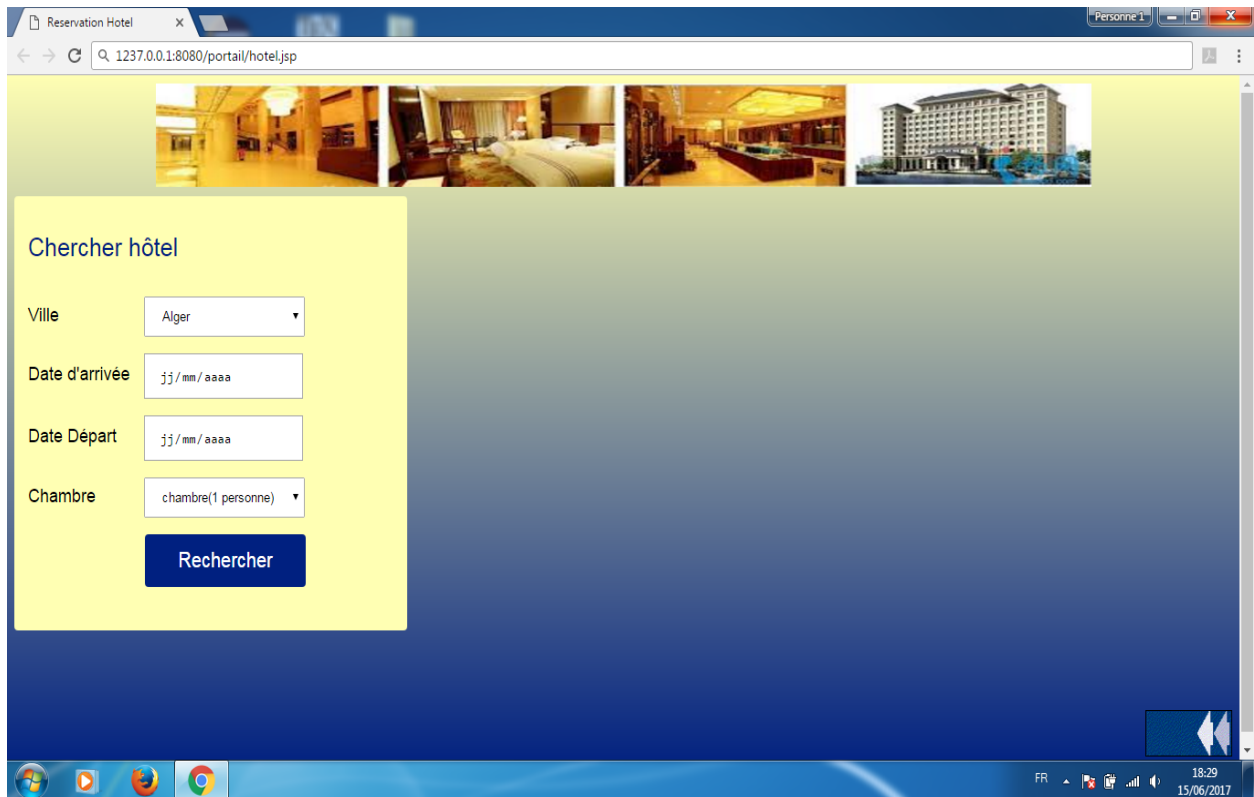


FIGURE 4.9 – La réservation dans un hôtel

Conclusion générale et Perspectives

Les web services sont apparus comme un outil qui tend à donner plus interactions pour permettre à deux entités hétérogènes (entreprises, clients, applications, etc. ...) de dialoguer au travers du réseau Internet.

Ils rendent en effet accessibles depuis Internet des fonctionnalités d'une application existante tout en ne modifiant pas en profondeur le système d'information de l'entreprise. Pour cela, les Web Services exploitent les standards d'échange Internet.

Les services Web avec leurs protocoles et leurs standards avancent vers toujours plus de normalisation. Déjà, le protocole d'échange de messages SOAP et le langage WSDL pour la définition de l'interface standardisent la couche de transport. Cette standardisation permet une grande interopérabilité entre des applications de technologie différente : les protocoles SOAP, WSDL et UDDI permettent de simplifier l'utilisation des services auxquels ils sont liés par l'utilisation de fichiers textuels, donc lisibles par l'utilisateur. Leur contenu se limitant aux éléments essentiels d'interopérabilité, ces standards assurent une grande indépendance de l'implémentation par rapport au système d'exploitation, à l'architecture de la machine et au standard utilisé.

Du point de vue économique les Services Web ont de fortes implications. Ils permettent de changer le mode de consommation des logiciels. Le logiciel n'étant plus en locale il peut être loué. Mais cela pose des problèmes tels que les difficultés de facturation à cause du manque de sécurité et le changement d'habitude que cela requiert pour le consommateur, les clients ont l'habitude de posséder les logiciels et d'acheter des mises à jour.

Pour finir les Services Web ont une autre limitation qui devrait ralentir leur utilisation. Cette limitation est le fait que les requêtes se font sur un réseau et donc cela empêche les Services Web d'être des services nécessitant une forte réactivité.

Dans ce mémoire nous avons présenté une conception et implémentation d'une application de gestion d'une Agence de réservation de voyage à base de service web.

L'application que nous avons développé permet à un client d'éviter de faire plusieurs recherches sur les web (compagnies aérienne, hôtel, voiture,...), pour planifier son voyage. Nous avons élaboré un portail de service qui fournit des interfaces utiles à la planification des voyages à travers l'utilisation de la technologie des services web.

Une extension possible de ce travail peut être la composition dynamique des services web. L'utilisation des annuaires et des ontologies permettra une meilleure découverte, une sélection, et une composition de services web au moment de l'exécution.

Références

- [1] <http://www.w3.org/TR/ws-arch/>).
- [2] <http://www.institut-numerique.org/13-besoins-dutilisation-des-services-web>.
- [3] http://www.technologytransfer.eu/article/21/2002/12/.NET_or_J2EE_-_Choosing_the_Right_Web_Services_Framework.html, 2002.
- [4] <http://jlafosse.developpez.com/java/developpement-n-tiers/plate-forme-java-EE/>, 2012.
- [5] www.dineshonjava.com/2013/05/introduction-to-wsdl.html, 2013.
- [6] <https://www.mulesoft.com/resources/esb/introduction-jee-j2ee-web-services>, 2017.
- [7] <https://fr.wikipedia.org/wiki/Servlet>, 2017.
- [8] https://fr.wikipedia.org/wiki/JavaServer_Pages, 2017.
- [9] https://fr.wikipedia.org/wiki/Enterprise_JavaBeans, 2017.
- [10] <https://www.ibm.com/developerworks/webservices/tutorials/ws-jax/ws-jax.html>, 2017.
- [11] P. COLLET. *Etat de l'art sur la contractualisation et la composition*. RNTL FAROS, Livrable F-1.1, 2006.
- [12] M. DODANI. *From objects to services : A journey in search of component reuse nirvana*. Journal of Object Technology, 2004.
- [13] Sun Microsystems INC. *Sun ONE Studio 4, Enterprise Edition for Java*. Universe Inc, 2002.
- [14] R.Libio J.Fontanel, PH.Lacomme. *Les services web*. Ellipses, 2013.
- [15] X.legalles L. Measano, G.Bernard. *Service Web avec J2EE et .NET Conception et Implimentation*. EYROLLES, 2003.
- [16] M.-C. Fauvet M. Dumas. *Intergiciel et Construction d'Applications Réparties*. Licence Creative Commons, 2008.

- [17] P.F Brown M. MACKENZIE F.McCabe, R.Metz. *Reference model for service oriented architecture 1.0*. OASIS – Technical Report wd-soa-rs, 2006.