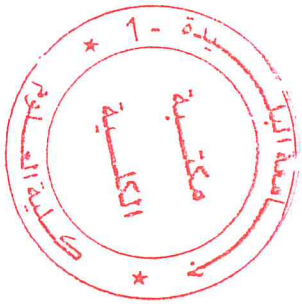
République Algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université SAAD DAHLAB de BLIDA
Faculté de science
Département informatique



MÉMOIRE DE MASTER INFORMATIQUE
SPÉCIALITÉ SYSTÈME INFORMATIQUE ET RÉSEAU

Développement de bases de tests pour l'évaluation des solutions de composition de services web

Réalisé par :
HADJZERROUK NESRINE
YAHI HADJIRA
Membres du jury :
présidente : Mme.ARKAM
examinateur : Mr.DERRAR

Promotrice :
Mme.CHIKHI IMANE
Encadreurs :
Mr.BOUKHEDOUMA HOUCINE
Mme.BENNA AMEL (CERIST)

26/09/2018

Résumé

La technologie des services web permet à des applications de dialoguer à distance via Internet, et ceci indépendament des plateformes et des languages sur lesquellles elles reposent.Cette technologie prend de plus en plus d'ampleur dans différents domaines et constitue toujours un axe de recherche trés actif. Toutefois,les travaux de recherche sur la découverte et la composition des service web souffrent d'un manque de grandes bases de tests pour l'évaluation et la validation des nouvelles solutions proposées par les chercheurs pour améliorer l'utilisation des services web.

L'objectif du travail présenté dans ce mémoire est de développer une application permettant de créer des bases de test offrant des informations liées aux intéractions d'un service avec d'autre services. Il s'agit, dans une premiére étape de concevoir l'application qui permet l'extraction des données sur l'intéraction d'un service invoqué avec d'autres services et les stocker dans des bases de tests. Dans une deuxième étape il s'agira de mettre en place la solution au niveau de la plateforme réseau Emulab pour des expérimentations à grande échelle.

Mots clefs : Service Web, Extraction des données, Composition de services, Base de tests.

Abstract

Web service technology enables applications to communicate remotely via the Internet, independently of the platforms and languages on which they are based. This technology is becoming increasingly important in various fields and is still a very important research area. active. However, research work on discovery and composition suffer from a lack of large test bases for the evaluation and validation of new solutions proposed by researchers to improve the use of web services.

The objective of this topic is to develop an application to create test bases offering information related to the interactions of a service with other services. It is a first step to design the application that allows the extraction of data on the interaction of a service invoke with other services and store them in a database. In the second step it will be a question of putting the solution at the level of the researchers to make experiments.

Keywords : Web Service, Data Extraction, Service cComposition, Test Database.

ملخص

تسمح تقنية خدمة الويب للتطبيقات بالتواصل عن بعد عبرا لإنترنت ، ، بغض النظر عن المنصات واللغات التي تعتمد عليها ، وتتزايد أهمية هذه التقنية في مختلف المجالات ولا تزال مجالًا مهمًا جدًا للبحث. ومع ذلك ، فإن العمل البحثي في الاكتشاف والتكوين يعاني من عدم وجود قواعد اختبار كبيرة لتقييم والتحقق من الحلول الجديدة التي اقترحها الباحثون لتحسين استخدام خدمات الويب.

الهدف من هذا الموضوع هو تطوير تطبيق لإنشاء قواعد اختبار تقدم معلومات تتعلق بتفاعلات الخدمة مع الخدمات الأخرى. إنها خطوة أولى لتصميم التطبيق الذي يسمح لاستخراج البيانات على تفاعل الخدمة باستدعاء مع خدمات أخرى وتخزينها في قاعدة بيانات.

في الخطوة الثانية ستكون مسألة وضع الحل على مستوى الباحثين لإجراء التجارب.

كلمات البحث: خدمة ويب ، استخراج البيانات ، تكوين الخدمة ، اختبار قاعدة البيانات.

Remerciements

Nous remercions Allah le tout puissant, qui nous a donné la foi, la force et la patience pour aller jusqu'au bout de ce travail.

Au terme de ce travail de fin d'études nous tenons à exprimer notre gratitude et nos remerciements pour toutes les personnes qui ont contribués à sa réalisation.

Nous tenons tout d'abord à remercier nos encadreurs Mlle Benna Amel et Mr Boukhedouma Houcine et notre promotrice Mme Chikhi Imane pour leurs aides, leurs conseils, leurs encouragements et leurs disponibilités dans ce mémoire.

Ainsi que tous nos professeurs qui nous ont enseignés durant nos études au département d'informatique.

Nos profonds remerciements pour les membres de jury qui ont acceptés d'évaluer ce travail.

A la fin nous tenons à remercier tous nos collègues d'études particulièrement notre promotion.

Dédicaces

A mes chers parents, pour tous leurs sacrifices, leurs amours, leurs tendresses, leurs soutiens et leurs prières tout au long de mes études.

A ma chère sœur Nassima pour son encouragement permanent, et son soutien moral.

A mes chers frères, Ibra et Jacob pour leurs appuis et leurs encouragements.

A toute ma famille pour leur soutien tout au long de mon parcours universitaire.

A mes amis Amina, Hadjer, Roumaissa, Amel et à toutes les personnes qui ont une place spéciale dans mon cœur et ma vie.

A mon binôme de travail Nesrine je te souhaite beaucoup de réussite dans ta vie.

Et touts les étudiants de M2 SIR.

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible.

Merci d'être toujours là pour moi.

YAHI

HADJIRA.

Dédicaces

Je dédie ce modeste travail

A la plus belle perle du monde… ma tendre mère

Pour son patience, son amour, son soutient, son encouragement.

A mon frère Seif Eddine.

A mes amis et mes collègues à l'université de Blida

A toutes personnes

Qui m'ont aidé à franchir un horizon dans ma vie…

HADJZERROUK

NESRINE.

Table des matières

Liste des tableaux

Introduction générale

Contexte général

L'interaction de différentes applications offertes par des entreprises et organisations en réseau a toujours été une tache complexe. Les applications peuvent être écrites dans des langages de programmation différents et exécutées sur des plates-formes hétérogènes. Plusieurs technologies permettent de résoudre ce problème de communication à travers un réseau. Les « **Services Web** » représentent actuellement la technologie la plus adaptée à ce problème Car elles permettent de construire des systèmes distribués faiblement couplés.

En effet, on peut construire des services plus complexes avec des entités logicielles faiblement couplées, et pour réaliser cela, la notion de composition de services est établie. Actuellement et pour améliorer l'utilisation des services web, les chercheurs viennent de proposer de nouvelles solutions de compositions de services, ces derniers ont besoin de grandes bases de tests pour valider et évaluer leurs solutions proposées.

Problématique

Les travaux de recherches sur les compostions de services web souffrent d'un manque d'un moyen efficace qui sert à la validation de solutions de compositions de services proposées par les chercheurs. Il est donc nécessaire de mettre à leurs dispositions de grandes bases de tests pour valider les propositions.

Objectif

L'objectif de ce travail est de proposer une application pour la résolution de problème de manque de grandes bases de tests pour l'évaluation et la validation des nouvelles solutions proposées par les chercheurs pour améliorer l'utilisation des services web. Dans le but de réaliser notre objectif, nous avons développé une application permettant la création des bases de tests offrant des informations liés aux interactions entre services, ces bases sont destinées aux travaux de recherches pour que les chercheur puissent évaluer leurs solutions de composition de services proposées.

Structure de document

Aprés une introduction générale présentant le contexte de l'étude, la problématique et l'objectif principal ; ce document se présentera en quatre grands chapitres :

Chapitre 1 : ce chapitre décrit les services web.

Chapitre 2 : ce chapitre décrit le concept de composition des services web en montrant les différents types de composition et en citant une des approches de composition des services web, puis décrit la création de bases de test d'une façon générale, une solution proposée pour la création de bases de test dans le cadre des services Web, et les techniques utilisées pour l'extraction des données.

Chapitre 3 : ce chapitre présente la solution proposée pour la création et l'exploitation de bases de test pour la composition de services Web.

Chapitre 4 : ce chapitre présente l'implémentation de la solution proposée, l'environnement de développement et l'ensemble des outils utilisés pour la réalisation de notre solution.

Enfin, une conclusion et des perspectives synthétiseront et clôturent ce document.

Chapitre 1

Généralité sur les Services Web

1.1 Introduction

Les Services Web(SW) puisent leurs origines dans l'informatique distribuée et dans l'avènement du web. Dans ce chapitre nous allons présenter l'architecture des SW ainsi que les outils qu'ils les utilisent, puis nous présenterons les SW de types REST. Et enfin nous citerons quelques avantages et inconvénients des services web.

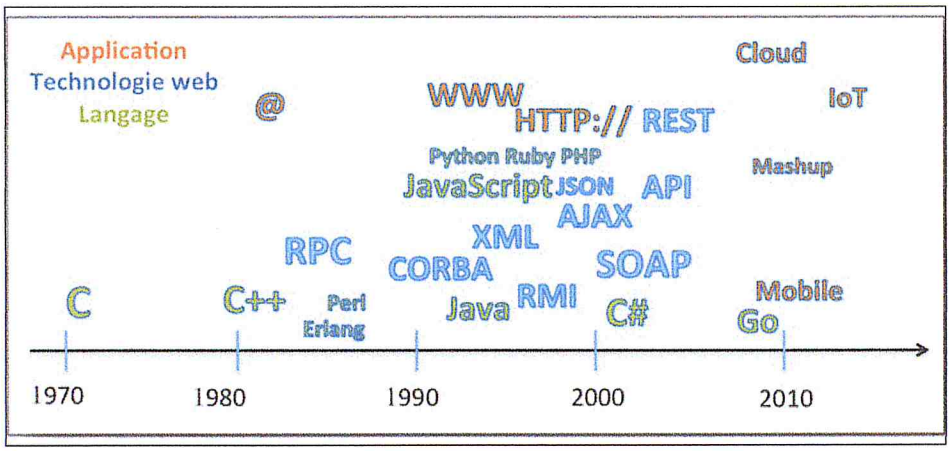


FIGURE 1.1 – Le développement du Web et ses technologies [1]

1.2 L'Architecture Orientée Services et les Services Web (SOA)

SOA est un nouveau style architectural qui facilite l'intégration de plusieurs entités logicielles en les organisant en ensembles fonctionnels appelées services [1]. L'objectif de cette intégration c'est de réaliser une fonctionnalité dont l'utilisateur à besoin. Cette architecture repose sur un concept basique : la notion de service. Un service est une application définit par un contrat, accessible par une interface et offrant une ou plusieurs fonctionnalités. Les services permettent de communiquer via l'échange de messages afin de cacher l'hétérogénéité des environnements de développements des applications. [1]. Un tel composant peut correspondre à une unité de traitement exécutable. Par exemple en J2EE1, il peut s'agir d'un EJB2 session ou d'une Servlet Java

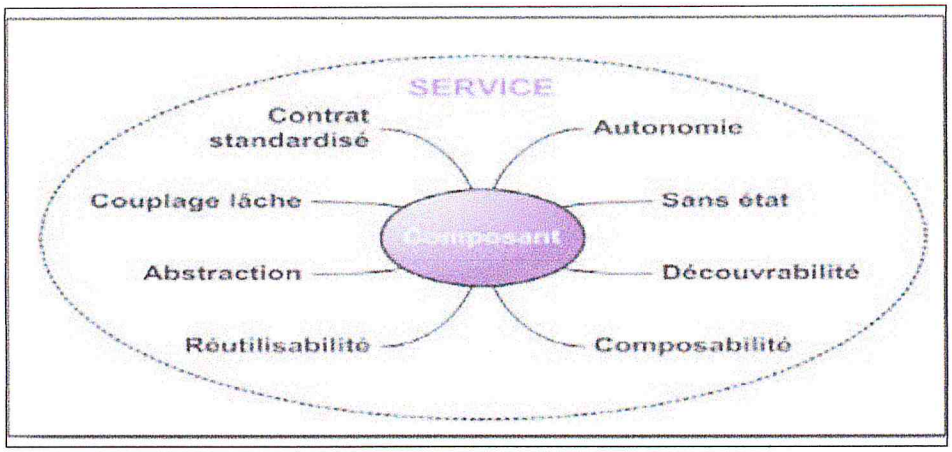


FIGURE 1.2 – Présentation d'un service [4]

1.2.1 Les caractéristiques d'un service

Thomas Erl [2] décrit le service par les caractéristiques suivantes (Figure 1.2) :

Interface : Les services d'un même système technique sont exposés au travers des interfaces respectant les mêmes règles de standardisation. Un service peut implémenter plusieurs interfaces et plusieurs services peuvent implémenter une interface commune.

Couplage faible : il s'agit d'introduire le minimum de dépendances entre les services pour permettre d'assembler ceux-ci aisément, cela se fait via des standards comme le langage XML (eXtensible Markup Language).

Abstraction : Le principe d'abstraction consiste à fournir les services d'un Système d'information sur un modèle boîte noire. Les seules informations accessibles aux consommateurs d'un service sont celles contenues dans son interface. Ainsi, les concepteurs et développeurs d'un consommateur de service ne sont pas au courant de la façon dont est implémenté le service.

Réutilisabilité : Un service doit être positionné comme une ressource réutilisable.

Autonomie : Les services sont indépendants dans leur finalité et leur exécution. Autrement dit ces services sont développés, déployés et administrés indépendamment les uns les autres. Ils fournissent des fonctionnalités directement utilisables.

Sans état : Un service doit minimiser la consommation de ressources en déléguant la gestion des informations d'état quand cela est nécessaire, il reçoit les informations nécessaires dans la requête d'un client.

Découvrabilité : Un service peut être découvert et interprété de façon effective.

Composabilité : Un service doit être conçu de façon à participer à des compositions de services.

1.2.2 Eléments de l'architecture orientée services

L'interaction dans une architecture orientée services, s'articule autour de trois grands rôles ; un producteur de service (fournisseur de service), un consommateur de service (client) et le répertoire de services (registre de stockage des services ou annuaire) 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. 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 (Figure 1.3) [3].

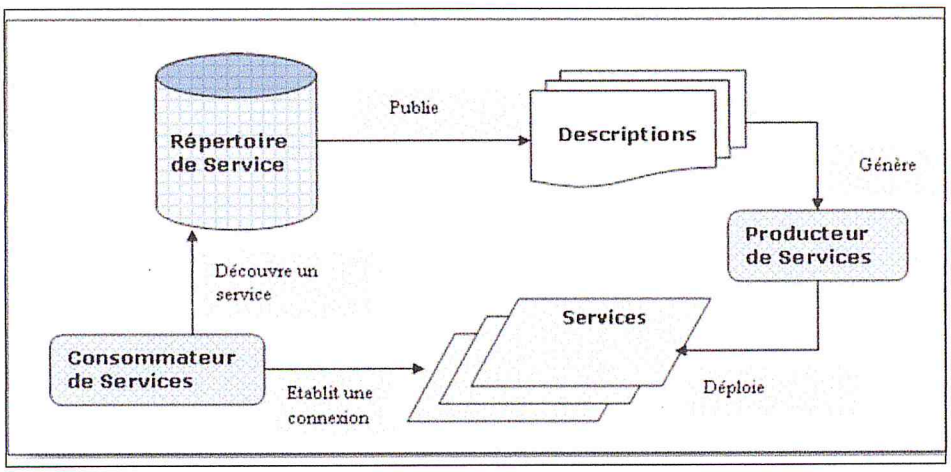


FIGURE 1.3 – Les interactions dans une architecture orientée services [3]

1.3 Définition de services web

Les services Web sont la déclinaison du paradigme des architectures orientée services, sur le Web. Ils ont été proposés initialement par IBM et Microsoft, puis en partie standardisés par W3C4. Actuellement, dans la littérature, Le concept «Service Web » est le sujet de définitions très variées.

Nagy et al. [4] définit un service Web comme étant « une application accessible à partir du Web et qui utilise les protocoles d'Internet pour communiquer et un langage standard pour décrire son interface ». En 2003, J.Daniel [5] donnait la définition suivante « les services Web sont des applications auto descriptives, modulaires, indépendantes et faiblement couplées qui fournissent un modèle simple de programmation et de déploiement d'applications, basés sur des normes s'exécutant à travers l'infrastructure Web, et qui peuvent être découvertes et invoquées dynamiquement via Internet ».

Le consortium W3C considère un service Web comme un composant logiciel vérifiant les propriétés suivantes :

- Identifié par une URI (Uniform Ressource Identifier).
- Ses interfaces et ses liens sont décrits en XML.
- Sa définition peut être découverte par d'autres services Web ;
- Peut interagir directement avec d'autres services Web à travers le langage XML et en utilisant des protocoles Internet.

1.4 Fonctionnement d'un service web SOAP

SOAP, WSDL et UDDI, sont les trois standards utilisés dans les architectures orientées service Web. La (figure 1.4) résume le principe de fonctionnement de cette architecture.

1. Une fois le service Web créé par le fournisseur, il sera déployé sur Internet et son WSDL sera publié dans le registre UDDI.
2. Le client (ayant des besoins spécifiques) va rechercher un service correspondant à ses critères à l'aide du registre UDDI.
3. Une fois le service trouvé, le client va récupérer sa description (document WSDL correspondant au service).
4. Le client va invoquer le service ; une communication va être mise en place entre l'utilisateur et le service Web.

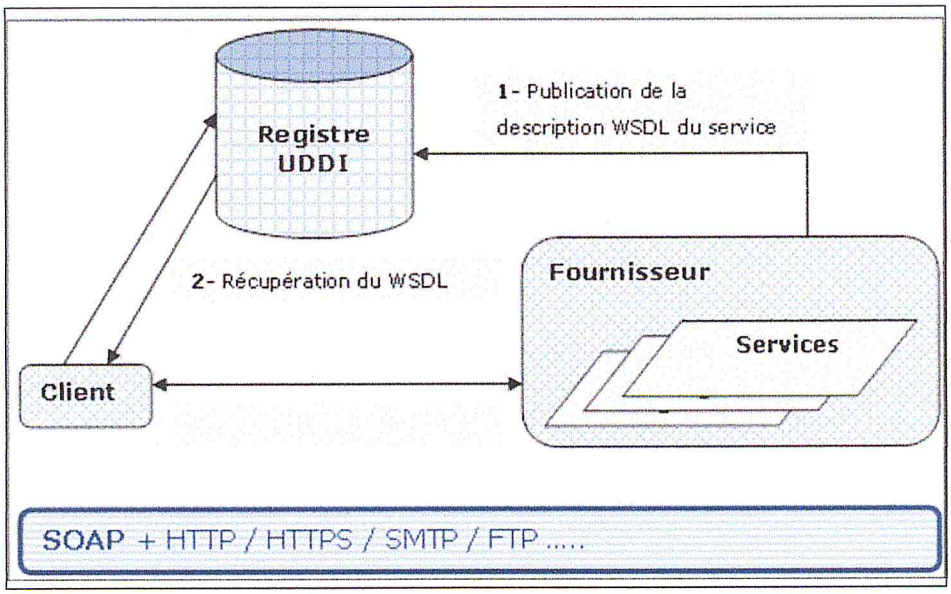


FIGURE 1.4 – Cycle de vie d'un service Web[3]

1.5 Les couches de l'architecture orientées services

Le groupe de travail WSA5 du W3C a proposé une vision multicouche de l'architecture orientée services. Cette vision n'étant pas unique, nous avons choisi une représentation simplifiée proposée par [3] et présentée dans la (Figure 1.5).

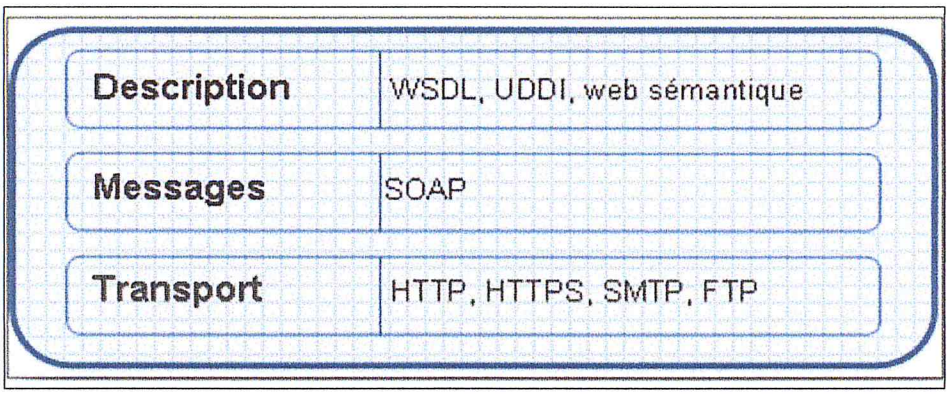


FIGURE 1.5 – Les différentes couches des architectures orientées service Web [3]

1.5.1 La couche Transport

Cette couche est responsable du transport des messages XML échangés entre les applications, c'est-à-dire les différentes communications. Il est souvent possible de spécifier le style, le mode et le protocole d'une communication. Actuellement, deux styles architecturaux totalement différents sont utilisés pour ces échanges de données :

— Le style RPC (Remote Procedure Call - appel de procédure à distance).
— Le style Document (communication sous la forme de documents XML auto-descriptifs).

Trois modes de communication peuvent être envisagés :

— le mode RPC ou mode requête-réponse,
— le mode "one-way messaging" ou mode requête simple,
— le mode "asynchronous callback" ou mode requête-réponse asynchrone.

Actuellement, cette couche inclut HTTP, SMTP, FTP, JMS (Java Message Service), et de nouveaux protocoles tels que BEEP.

Le protocole HTTP

Est un protocole simple capable de s'accommoder à la fois de la qualité de service et des temps de latence très variables de l'Internet ; ce que n'est pas le cas par exemple des protocoles d'environnements répartis tels que DCOM ou CORBA. De plus, en s'appuyant sur HTTP, on ne se heurte pas aux problèmes de pare-feu (firewall) ou de configuration de réseau IP qui rendent parfois difficile le déploiement d'applications à objets répartis au-delà du périmètre du réseau d'entreprise [6].

1.5.2 La couche messages

La communication par messages constitue un point central dans toutes architectures orientées service Web. Ces messages sont basés sur XML qui permet l'échange de données structurées, indépendamment des langages de programmation ou des systèmes d'exploitation. Les types de données utilisés sont eux aussi décrits en XML. C'est ce qu'on appelle l'encodage de type. Cet encodage peut se faire selon l'une des deux approches suivantes :

- **Literal :** suit littéralement les définitions des schémas XML. [38].
- **SOAP encoded :** suit la spécification du protocole SOAP (Simple Object Access Protocol). [39].

1.5.2.1 Le protocole SOAP

SOAP est apparu en 1998 sous l'impulsion de Microsoft, et plus tard développé en collaboration avec DevelopMentor, IBM, Lotus, et UserLand SOAP permet la transmission de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur. Le transfert se fait le plus souvent à l'aide du protocole HTTP, mais peut également se faire par un autre protocole, comme SMTP. Le SOAP est composé de deux parties [7] :

- un modèle de données, définissant le format du message, c'est-à-dire les informations à transmettre.
- une enveloppe, contenant des informations sur le message lui-même afin de permettre son acheminement et son traitement.

Le message SOAP Un message SOAP est un document XML dont la structure est spécifiée par des schémas XML. Plus précisément tout message SOAP se compose d'un élément enveloppe qui englobe un élément entête et un élément corps (figure 1.6).

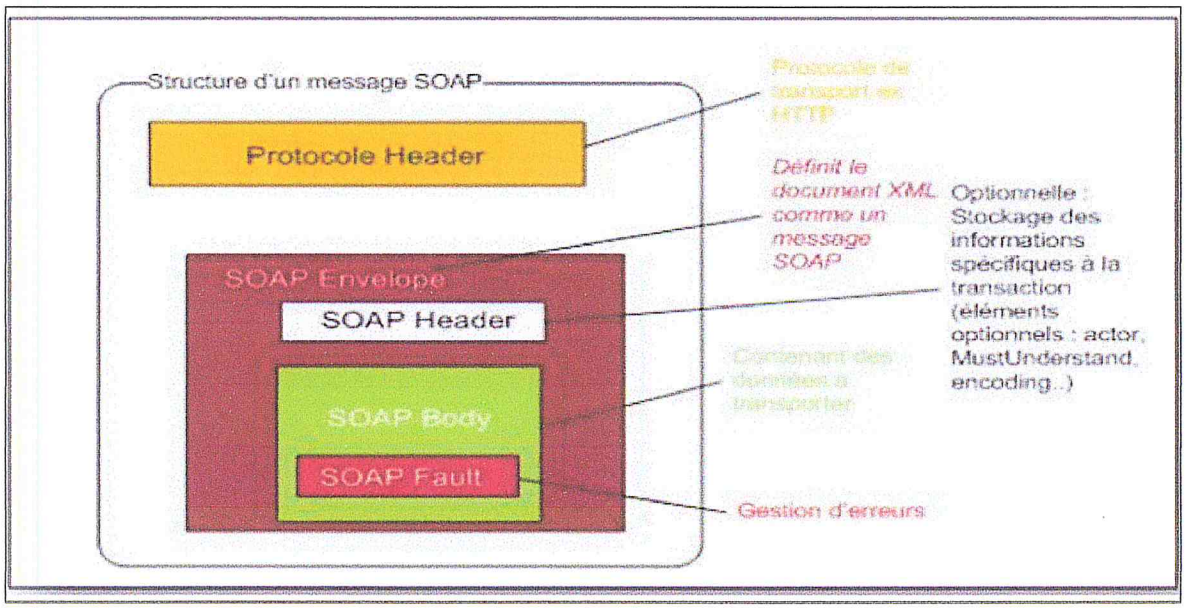


FIGURE 1.6 – Structure d'un d'un message SOAP [7]

SOAP enveloppe : L'enveloppe est le premier élément du document XML, représentant le message. C'est l'élément racine du message. L'Enveloppe sert également d'introduction, de point de chargement des règles d'encodage SOAP. Ces règles sont utilisées pour indiquer la manière d'interpréter le 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.
— Le premier élément fils de <Enveloppe> est l'élément <Header>, optionnel. Le but principal d'un tel élément est de fournir des extensions au protocole, n'ayant pas directement de lien avec telle ou telle méthode spécifiée, mais apportant plutôt des informations contextuelles comme l'identifiant de transactions et/ou des informations relatives à la sécurité.
— 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. Les utilisations typiques du SOAP Body sont :

  • Contenir des entrées applicatives : RPC
  • Permettre l'encodage des entrées : déclaration d'objet, de valeur.
  • Reporter les erreurs (message retour).
— SOAP fault (erreur) : est un élément facultatif défini dans le corps SOAP et est utilisé pour reporter les erreurs.

SOAP est :

— Assez ouvert pour s'adapter à différents protocoles de transport.
— Indépendant de la plate-forme.
— Indépendant du langage.
— Extensible.

1.5.3 La couche description

Dans le cadre des recommandations du modèle de référence OASIS, une description de service représente les informations nécessaires afin d'utiliser un service et facilite la visibilité et l'interaction entre les consommateurs et les fournisseurs de services [40]. Le protocole SOAP met à la disposition des services Web, un moyen standard de structuration et d'échange de messages XML. Il ne fournit en aucun cas une indication sur la structure que le message doit respecter vis à vis du service web sollicité. La spécification WSDL [41] [42] a vu le jour afin d'offrir une grammaire qui décrit l'interface des services Web de manière générique. Ces deux standards, SOAP et WSDL, définissent ensemble l'aspect le plus basique du développement de l'infrastructure des services Web. Un troisième standard a été con¸cu pour réduire l'écart entre les applications clientes et les services Web, appelé UDDI [43].

1.5.3.1 WSDL – Web Service Description Language

Langage né des efforts d'IBM et de Microsoft, WSDL est une note du W3C en mars 2001, dérivée de XML et qui décrit l'interface d'utilisation d'un service Web (méthodes et propriétés des composants de l'application), ses liaisons de protocoles (un service peut proposer une communication via SOAP sur un protocole de transport, via HTTP GET ou POST, ou via MIME), et ses détails de déploiement. Le fichier WSDL peut être généré automatiquement par les boites à outils existant sur le marché. [7]

Structure d'un fichier WSDL

Un document WSDL se compose d'un ensemble d'éléments décrivant les types de données utilisés par le service, les messages que le service peut recevoir, ainsi que les liaisons SOAP associées à chaque message. Le schéma suivant (figure 1.7) illustre la structure d'un fichier type WSDL qui est un document XML, en décrivant les relations entre les sections constituant un document WSDL.

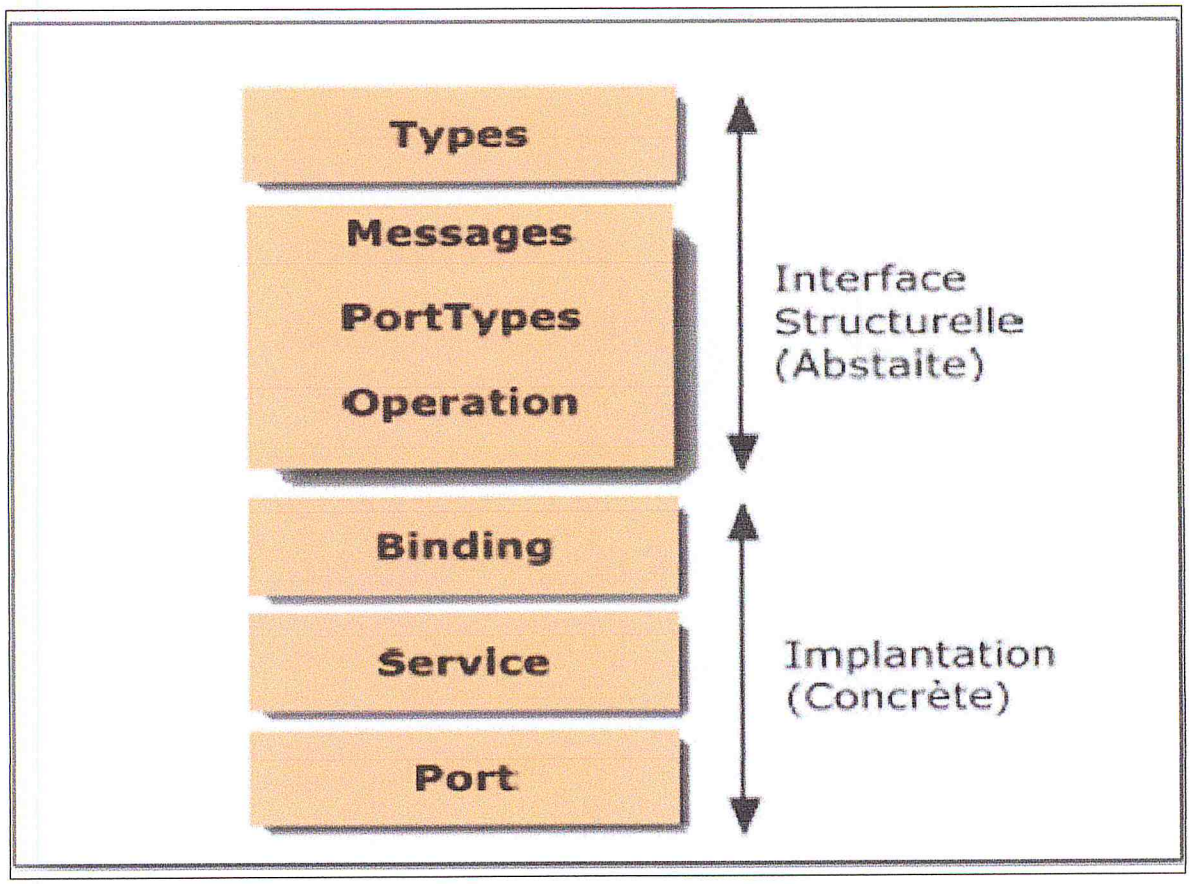


FIGURE 1.7 – Structure d'un document WSDL [7]

Un fichier WSDL contient donc sept éléments :

— Types : qui fournissent les définitions abstraites de types de données utilisées pour décrire les messages échangés.

— Message : représente une définition abstraire (noms et types) des données en cours de transmission. Un message comporte une ou plusieurs parties logiques, chacune étant associée avec une définition dans un système de type.
— Port Type : Ils sont utilisés pour définir les traite décrivent un ensemble d'opérations. Ces opérations représentent une unité d'action pour le service décrit plusieurs messages de sortie ou d'erreurs.
— Binding : spécifie une liaison entre un HTTP...).
— Service : indique les adresses de port de chaque liaison.
— Port : représente un point d'accès de services défini par une adresse réseau et une liaison.
— Opération : décrit une action exposée dans le port.

Le document WSDL peut être divisé en deux parties. Une partie pour les définitions abstraites, et une autre pour les descriptions concrètes.

La description concrète est composée des éléments qui sont orientés vers le client pour le service physique. Les trois éléments concrets XM L présents dans un WSDL sont :
— <wsdl :service>
— <wsdl :port>
— <wsdl :binding>.

La description abstraite est composée des éléments qui sont orientés vers la description des capacités du service Web. Ses éléments abstraits définissent les messages SOAP de façon totalement indépendante de la plate-forme et de la langue. Cela facilite la définition d'un ensemble de services pouvant être implémentés par différents sites Web. Les quatre éléments abstraits XML qui peuvent être définis dans un WSDL sont :
— <wsdl :types>
— <wsdl :message>
— <wsdl :operation>
— <wsdl :portType>

```
<definitions>
  <types> <!--type abstrait de données> </types>
  <message> <!--structure du message> </message>
  <portType><!--interface du Web service>
        <operation>
          <!-- une description d'une action supportée par le service
        </operation>
  </portType>
  <binding>
            <!--comment peut-on y accéder>
  </binding>
  <service>
         <!--qui  fournit le service>
         <port> <!--point de terminaison> </port>
  </service>
</definitions>
```

FIGURE 1.8 – Exemple d'un fichier WSDL

1. L'élément types : décrit sous la forme d'un schéma XML les types des données échangées entre le client et le fournisseur de services.
2. L'élément message : décrit les noms et les types d'un ensemble de champs à transmettre. Il peut être comparé aux paramètres d'un appel de procédure.
3. L'élément opérations : Décrit les opérations invoquées à l'aide des messages reçu, émis par le service et éventuellement des messages d'erreur.
4. L'élément portType : définit les opérations (collection des éléments operation) en terme de paramètres d'entrée et de sortie, il peut être comparé à une interface Java.
5. L'élément binding : spécifie le protocole de transfert (HTTP / SMTP / FTP) et le format d'encodage des données (encodage RPC, Document, etc.) pour une liste d'opérations,
6. L'élément port : un point de terminaison (End point), identifié de manière unique par la combinaison d'une adresse Internet et d'une liaison.
7. L'élément service : regroupe un ensemble de points de terminaison (éléments endpoint), offrant chacun une alternative (différents protocoles, etc.) pour accéder aux opérations du service en identifiant de manière unique la combinaison d'un élément binding et d'une adresse interne.

1.5.3.2 UDDI – Universal Description, Discovery and Integration

Défini initialement par Ariba, IBM et Microsoft, UDDI est plus qu'un simple protocole d'annuaire : il fournit un protocole, une API (Application Programming Interfaces) et une plate-forme permettant de trouver le service web recherché (de façon manuelle et automatique), mais aussi d'en annoncer la disponibilité . Un annuaire UDDI est constitué de pages blanches (nom de l'entreprise, adresse, contacts), de pages jaunes (services classés par catégories industrielles) et de pages vertes (information d'implémentation des services Web proposés). Toute information saisie dans un annuaire est répliquée sur l'ensemble des annuaires. Concrètement, ces annuaires sont des fichiers XML hébergés par des entreprises appelées opérateurs UDDI ou nœuds d'opérateurs . Grâce à UDDI, les entreprises peuvent enregistrer des données les concernant, des renseignements sur les services qu'elles offrent et des informations techniques sur le mode d'accès à ces services.

UDDI, (figure 1.9), fournit trois services de bases :
— **Publish :** ce service gère la méthode dont le fournisseur de service s'enregistre lui même ainsi que ses services
— **Find :** ce service gère la manière dont un client peut localiser le service désiré, cela peut passer par des invocations de services web pour une utilisation automatique ou par une consultation.
— **Bind :** ce service gère la façon dont un client peut se connecter service web une fois celui ci localisé.

Le scénario classique d'utilisation d'UDDI est illustré ci-dessous (figure 1.9) L'entreprise B a publié le service Web et l'entreprise A est le client de ce service.

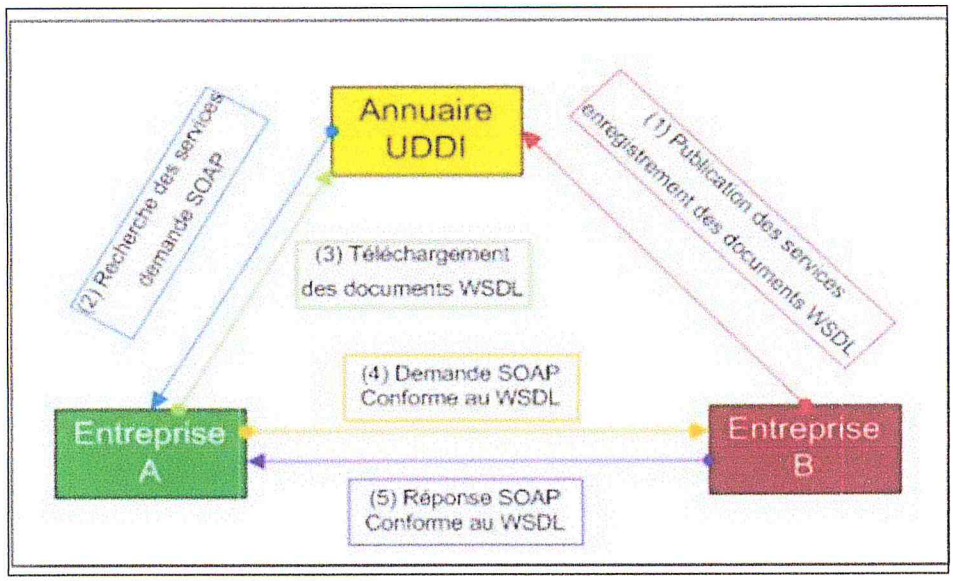


FIGURE 1.9 – Annuaire UDDI [9]

1.6 Services web REST

Le style REST (Representational State Transfer) est une abstraction d'éléments architecturals dans un système hypermédia distribué. REST ignore les détails d'implémentation du composant et la syntaxe du protocole afin de se concentrer sur les rôles des composants, contraintes sur leurs interactions avec d'autres composants, et leur interprétation des éléments de données significatives. Il englobe les contraintes fondamentales sur les composants, connecteurs, et les données qui définissent la base de l'architecture Web, et donc l'éssence de son comportement en tant qu'application réseau [44].

De nombreux fournisseurs de services tels que Google, Yahoo, Amazon etc., proposent une variété de service de REST. [44] Il a été décrit la première fois dans sa thèse. Tableau1 sont certains services REST.

TABLE 1.1 – Utilisation de REST dans les entreprises [44]

Fournisseurs	Service RESTfull
Google	Youtube, Blogger, BookSearch, Calendar, Picasa, CodeSerach, DocumentsList, Finance, Portlolio, Notebook, Health, Spreadsheets ...etc.
Yahoo	BOSS, Social, Address Book, HotJobs, Travel, Local, traffic,Maps,Music,Delicious, Flickr,...etc
Amazon	eCommerce, S3, ItemSearch
Others	FaceBook, Twitter, Digg, BBC,Wikipedia, MySpace

Un Service RESTful est donc un autre moyen de construire les services web. Ce type de WS suscite un grand intérêt croissant dans l'industrie et il a été largement adopté par certaines entreprises en raison de sa simplicité et sa légèreté (light-weight). L'idée de WS RESTful est d'appliquer les principes de REST dans le développement de WSs :

- **Resource-centrique :** les entités conceptuelles et les fonctionnalités sont modélisées comme des ressources identifiées par universal resource identifier (URI).
- **L'interface uniforme :** les ressources sont accessibles et manipulées via les opérations standardisées (GET, POST, PUT, DELETE, TRACE, CONNECT) dans le protocole HTTP [45]. GET est une opération pour prendre la représentation des ressources ; POST, PUT et DELETE sont utilisées pour créer, mette à jour, et supprimer les ressources respectivement.
- **L'état de transfert :** Les composants du système communiquent via ces interfaces d'opération standard et échangent les représentations de ces ressources (une ressource peut avoir multiples représentations). Dans un environnement de REST, serveurs et clients généralement transitent via l'état des représentations des ressources différentes en suivant les inter-liens entre les ressources.

L'architecture de REST introduit une série d'éléments architecturaux (agent d'utilisateur, proxies, passerelles et les serveurs originaux) qui sont destinés à être combinés pour construire un système scalable, il permet un grand nombre de clients (ou les agents d'utilisateur) pour accéder aux ressources publiées par un serveur d'origine unique illustré dans la (Figure 1.10). Chaque élément est relié à l'aide de protocole HTTP.

Les proxies et passerelles sont optionnelles, ils sont généralement ajoutés à une architecture compatible avec le style REST pour effectuer le contrôle d'accès, la mise en cache, et une sorte de traduction de protocoles. Un proxy de contrôle d'accès peut être connecté à plusieurs agents d'utilisateur, il permet à un sous-ensemble d'entre eux d'accéder au serveur d'origine.

Chaque demande de client peut donc passer par l'élément intermédiaire (d'ici proxy ou passerelle) et être entretenu de façon indépendante par un serveur d'origine.

En considérant l'existence de plusieurs serveurs, ils sont considérés comme des res-

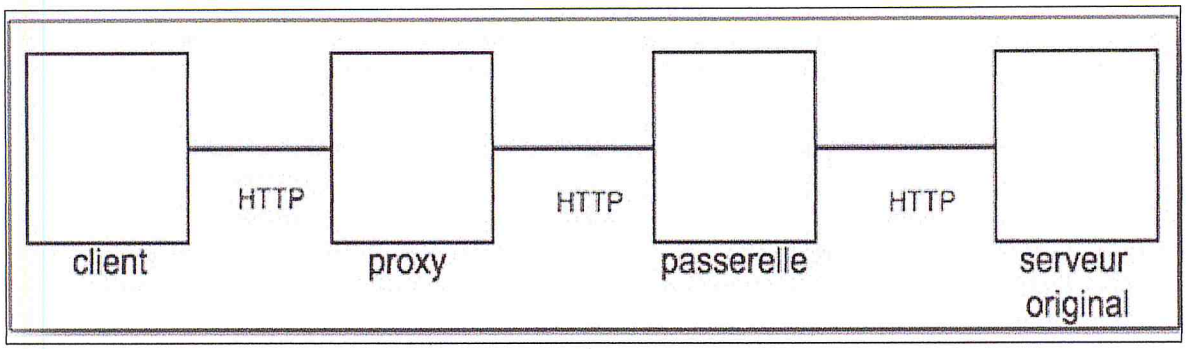


FIGURE 1.10 – REST basique élément architectural et connecteur [44]

sources autonomes et débranchés de l'information dont l'état évolue de façon indépendante. Les clients peuvent accéder séquentiellement à plusieurs serveurs (par exemple ils suivent les hyperliens de l'un à l'autre). Ce faisant, les clients peuvent aussi en quelque sorte recueillir les informations provenant de plusieurs serveurs et les regrouper localement. Ainsi, REST semble soutenir une forme de composition limitée au client. Aucun élément intermédiaire qui peut regrouper des informations à partir de plusieurs serveurs n'est explicitement prévu.

1.6.1 WADL

Le langage WADL (Web Application Description Language) est une description XML lisible par machine des services Web basés sur HTTP. WADL modélise les ressources fournies par un service et les relations entre elles. WADL est destiné à simplifier la réutilisation des services Web basés sur l'architecture HTTP existante du Web. Il est indépendant de la plate-forme et du langage et vise à promouvoir la réutilisation d'applications au-delà de l'utilisation de base dans un navigateur Web. WADL a été soumis au consortium World Wide Web par Sun Microsystems le 31 août 2009 , mais le consortium n'a pas actuellement l'intention de le normaliser. WADL est l'équivalent REST du langage WSDL (Web Services Description Language) de SOAP, qui peut également être utilisé pour décrire les services Web REST. [w1]

1.7 Comparaison entre SOAP et REST

Nous présentons ci-dessous la liste de différences entre SOAP et REST :

— SOAP adopte une approche de service donc chacun est représenté par un contrat formel pour définir l'interface de service web et un ensemble de méthodes spécifiques. Quand à REST, il adopte une approche de ressource qui est identifiée par des URIs.

— Dans SOAP, l'interface dépend de la méthode invoquée qui est spécifique pour chaque service. L'interface d'innovation dans REST est standardisée et uniforme pour toutes les ressources. SOAP est donc plus extensible mais moins simple.

— REST suit modèle de stateless, par rapport à SOAP, qui a des spécifications de mise en oeuvrede stateful.

— REST adopte une interaction client/serveur et synchrone dans le Web. Quand à SOAP, il permet une composition de service synchrone ou asynchrone, une meilleure interopérabilité et plus de sémantique.

— SOAP utilise des interfaces et des opérations nommées pour exposer la logique de métier.

— REST utilise URI et des méthodes similaires (GET, PUT, POST, DELETE) pour exposer les ressources.

— SOAP a un ensemble de spécifications standard. Par exemple, WS-Security est la norme pour la sécurité dans la mise en œuvre. Il s'agit d'une norme détaillée fournissant des règles de sécurité dans la mise en œuvre de l'application. Pareillement, nous avons les spécifications particulières pour la messagerie, transactions etc.

— Contrairement à SOAP, REST repose principalement que sur les protocoles HTTP/HTTPS.

— REST est utilisé essentiellement par les développeurs de Web et surtout dans le cadre du Web 2.0. Il répond aux exigences d'intégration simples au niveau de l'interface d'utilisateur (User interface, UI). SOAP est plutôt utilisé dans les compositions de services complexes qui nécessitent de la sémantique. En raison des points de vue différents de construire les WS.

1.8 Avantages et inconvénients

Malgré les intérêts et l'utilité major des SW, ces derniers présentent quelques inconvénients , dans ce qui suit nous citerons est les avantages et les inconvénients des SW :

Avantages

L'interopérabilité : C'est la capacité des services web d'interagir avec d'autres composantes logicielles via des éléments XML et utilisant des protocoles de l'Internet.

La simplicité : Les services web réduisent la complexité des branchements entre les participants. Cela se fait en ne créant la fonctionnalité qu'une seule fois plutôt qu'en obligeant tous les fournisseurs à reproduire la même fonctionnalité à chacun des clients selon le protocole de communication supporté. Une composante logicielle légèrement couplée : L'architecture modulaire des services Web, combinée au faible couplage des interfaces associées, permet l'utilisation et la réutilisation de services qui peuvent facilement être recombinés à différents autres applications.

L'hétérogénéité : Les services web permettent d'ignorer l'hétérogénéité entre les différentes applications. En effet, ils décrivent comment transmettre un message (standardisé) entre deux applications, sans imposer comment construire ce message.

Auto-descriptivité : Les services web ont la particularité d'être auto-descriptifs, c'est à dire capables de fournir des informations permettant de comprendre comment les manipuler. La capacité des services à se décrire par eux-mêmes permet d'envisager l'automatisation de l'intégration de services.

Inconvénients

Les services Web ont de faibles performances par rapport aux autres approches de l'informatique répartie telles que le RMI, CORBA, ou DCOM. En l'utilisation du protocole http, les services Web peuvent contourner les mesures de sécurité miss en place à travers les firewalls. Les transferts reposent sur le XML, ce qui pose un problème sur la taille des fichiers échangés. Les fichiers XML sont le plus souvent de très gros fichiers et ceci entrainera une lourdeur considérable. Ils ne sont pas sécurisés à100

1.9 Conclusion

Dans ce chapitre nous avons décrit les principaux concepts de services web d'où nous avons discuté les interactions dans une architecture orientée services et le fonctionnement d'un service web et ses standards, puis nous avons mentioné la notion des services REST. Dans le prochain chapitre, nous allons discuter la notion de composition de services web.

Chapitre 2

Composition et base de tests des services web

2.1 Introduction

Dans le chapitre précédent nous avons discuté les services web. Dans ce chapitre nous présenterons, dans un premier temps le concept de composition des services Web, les principaux types de compositions et une des approches de composition automatique des services Web. Dans un second temps,nous décrivons la création de base de tests de façon générale , nous présentons une solution pour la création de base de tests dans le cadre des services web et les techniques relatives à cette solution.

2.2 Composition des services Web

La composition de services Web est la plus importante fonctionnalité assurée par une architecture SOA. Celle-ci offre un environnement homogène pour la composition dans la mesure où toutes les parties de la composition sont des services idéalement décrits de la même façon et communiquant par les mêmes standards d'échange de messages. La composition permet de combiner des services pour former un nouveau service dit composé ou composite. L'exécution d'un service composé implique des interactions avec des services partenaires en faisant appel à leurs fonctionnalitées. Le but de la composition est avant tout la réutilisation de services (simples ou composés) et de préférence sans aucune modification de ces derniers. [8].

2.3 Cycle de vie d'une composition des services Web

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

La phase de définition : Durant cette phase le demandeur de service précise la spécification de la composition, qui devrait fournir suffisamment d'informations sur les besoins des utilisateurs et les priorités pour le service composé [9], Ensuite la spécification est décomposée en un modèle abstrait qui englobe un ensemble des activités, le contrôle et le flux des données entre ces activités, la qualité de service (QoS) et les comportements exceptionnels.

La phase de sélection : cette phase consiste à découvrir pour chaque activité dans le service composé les services web appropriés qui correspondent aux besoins définies, cette découverte est fondée sur les informations contenues dans les descriptions des services web publiés. Il est possible que plus d'un service répond aux besoins. Par conséquent, le meilleur service doit être sélectionné. Après que tous les services Web requis sont identifiés et liés aux activités correspondantes, le service composé est produit.

La phase de déploiement : Dans cette phase, le service composé est déployé pour permettre son instanciation et l'invocation par les utilisateurs finaux. Le résultat de cette phase est le service composé exécutable.

La phase d'exécution : Dans cette phase, l'instance de service composé est créée et exécutée par un mécanisme d'exécution, qui est également chargé d'invoquer les services participants. Pendant cette étape d'exécution les tâches de contrôle et de surveillance de la

composition, mesure de la performance et la gestion des exceptions, doivent être assurées. [11]

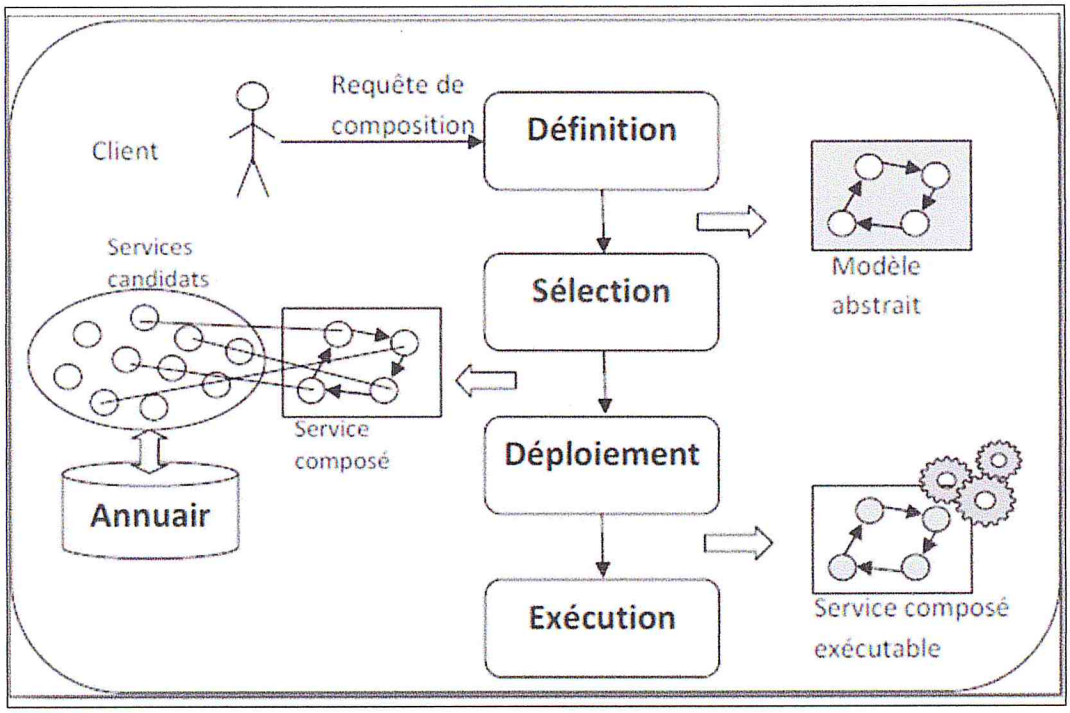


FIGURE 2.1 – Cycle de vie de composition des services web. [9]

2.4 Défis de composition des services Web

Le problème de la composition des services Web est en plein essort.
Nous soulignons ici quelques sources de ses complexités [10] :

— La complexité liée a la conception de services Web : un Web service n'est pas adaptable. Il est passif jusqu'à ce qu'il soit invoqué. Il a des connaissances de lui-même mais pas de ses applications ou de ses utilisateurs clients.
— Le nombre des Web services disponibles est très grand, il est déjà au-delà des capacités humaines pour les analyser manuellement.
— Puisque le nombre de Web services augmente jour âpres jour, il devient plus difficile de trouver artificiellement le Web service qui peut effectuer la tache à accomplir et encore plus difficile de composer un ensemble de services avec des approches classiques.
— Les Web services peuvent être crées et réactualises a la volée. Par conséquent, le système de composition doit détecter la mise 'à jour lors de l'exécution. Ainsi, le schéma de la composition devrait s'adapter en fonction des nouvelles informations.
— Les Web services sont généralement établis par des organisations différentes qui utilisent différents modèles conceptuels pour la présentation des caractéristiques des

services. Cela exige l'utilisation d'informations pertinentes pour faire correspondre (matching) et composer les services Web.

2.5 Types de composition de services Web

2.5.1 La composition statique des services Web

La composition statique a lieu au moment de la conception d'une application. Ainsi, les composants concernés sont choisis, liés et assemblés avant d'être déployés.une telle composition et adaptée aux environnements fermés où les composants n'évoluent pas souvent. Ce type de composition rend les applications peu flexibles et parfois inappropriées aux exigences des clients. La composition statique des services web peut se faire de deux manières : par Orchestration ou Chorégraphie [11]

2.5.1.1 Orchestration

[Barros et al., 2005b] définissent l'orchestration comme un ensemble de processus exécutés dans un ordre prédéfini afin de répondre à un but. Ce type de composition permet de centraliser l'invocation des services Web composants. Chaque service est décrit en termes d'actions internes. Les contrats entre deux services sont constitués selon le processus à exécuter. À l'instar de [46], [47] définissent l'orchestration comme un processus exécutable. [47] ajoutent que l'orchestration est un ensemble d'actions à réaliser par l'intermédiaire de services Web. Un moteur d'exécution, un service Web jouant le rôle de chef d'orchestre, gère l'enchaînement des services Web par une logique de contrôle. Pour concevoir une orchestration de services Web, il faut décrire les interactions entre le moteur d'exécution et les services Web. Ces interactions correspondent aux appels, effectués par le moteur, d'action(s) proposée(s) par les services Web composants.

La (Figure 2.2) illustre l'orchestration. La requête du client (logiciel ou humain) est transmise au moteur d'exécution (Moteur). Ce dernier, d'après le processus préalablement défini, appelle les services Web (ici, SW1, SW2, SW3 et SW4) selon l'ordre d'exécution.

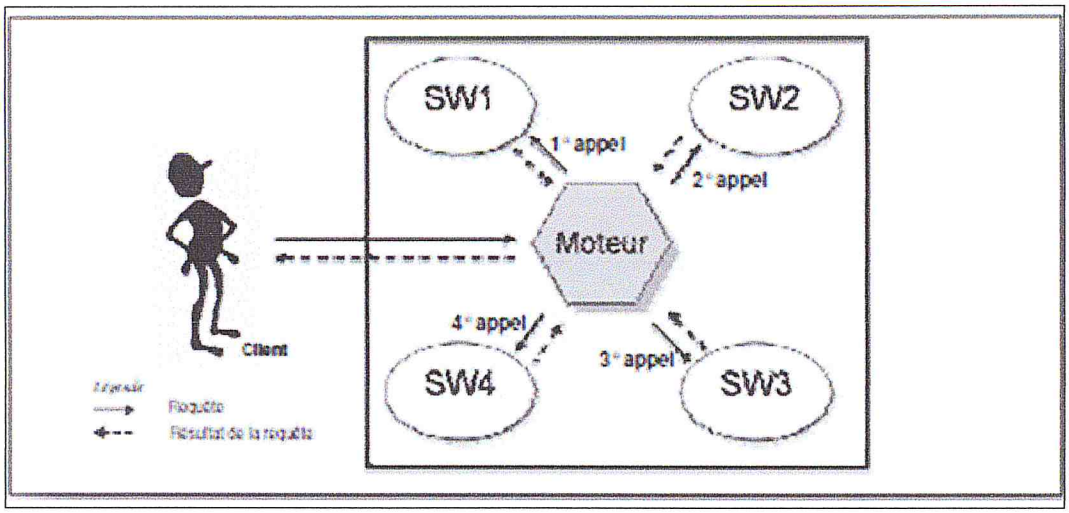


FIGURE 2.2 – Vue générale de l'orchestration [46]

En d'autres termes, l'orchestration de services Web exige de définir l'enchaînement des services Web selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Ces derniers (le canevas et le script) décrivent les interactions entre services Web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. Le module exécutant le script d'orchestration de services Web est appelé un moteur d'orchestration. Ce moteur d'orchestration est une entité logicielle qui joue le rôle d'intermédiaire entre les services en les appelants suivant le script d'orchestration.

2.5.1.2 Chorégraphie

Contrairement a l'orchestration, la chorégraphie n'a pas de coordinateur central. Chaque service web mêlé dans la chorégraphie sait exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu. Elle est associée à l'échange de message entre services web plutôt qu'à un processus métier exécuté par un seul partenaire. La chorégraphie est un effort de collaboration dans lequel chaque participant du processus décrit l'interaction qui l'appartient. Elle trace la séquence des messages qui peut impliquer plusieurs services web [8].

La collaboration dans la chorégraphie des services web peut être représentée da la manière suivante :

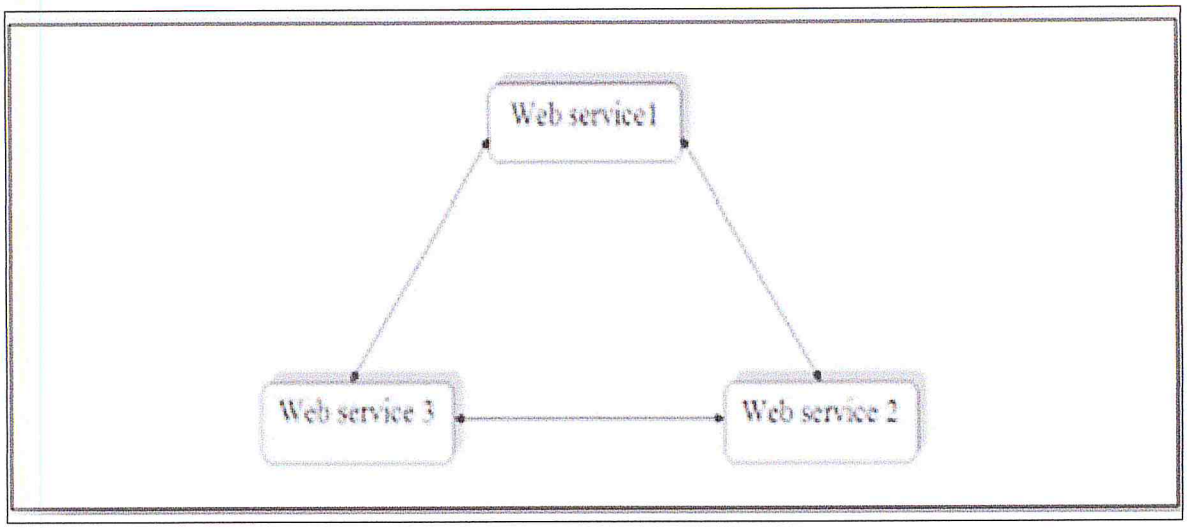


FIGURE 2.3 – Chorégraphie des services web [8]

2.5.2 La composition dynamique des services Web

Une composition de services est dite dynamique si les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur [12]. Une approche dynamique pour la composition de services offre le potentiel de réaliser des applications flexibles et adaptables en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur. Ce type de composition peut engendrer de nombreuses applications utiles qui n'ont pas été prévues à l'étape de conception. Par conséquent, la composition dynamique de services est propice dans un environnement tel que le web et l'informatique pervasive où les composants disponibles sont dynamiques et les attentes des utilisateurs variables et personnalisées.

2.5.3 Selon le degré d'automatisation

En fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, ces propositions sont : manuelles, semi-automatiques ou automatiques [13] :

2.5.3.1 La composition manuelle

La composition manuelle des services Web suppose que l'utilisateur gère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés.

2.5.3.2 La composition semi-automatique

Les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans le sens qu'elles font des suggestions sémantiques pour aider à la sélection des services Web dans le processus de composition

2.5.3.3 La composition automatique

La composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise.

2.6 Une approche de composition automatique des services web

Plusieurs approches ont été proposées pour la compositon des services web comme l'Approches basée sur le web sémantique, Composition orientée intelligence artificielle et approche basée sur les workflow... etc . Dans notre cas, nous avons adopté pour l'approche basée workflow que nous détaillons dans ce qui suit.

2.6.1 approche basée sur les workflow

La définition de la composition de services correspond alors à un ensemble de services atomiques (ou composés) sur lesquels on effectue un contrôle du flux.
D'autre part, un workflow a également besoin de définir un flux d'activités. Un workflow est une abstraction d'un processus de type business. Il est composé d'un nombre d'échelons logiques (aussi appelés tâches ou activités), de dépendances entre tâches, de règles et de participants. Dans ce dernier, une tâche peut représenter une activité humaine ou un système (logiciel). Il est nécessaire d'associer une tâche à un service quand il est appliqué aux services web. Une composition d'un workflow implique la sélection de tâches appropriées à des fonctionnalités désirées, nous devant prendre en compte les connections entre ces tâches (flux de contrôle et de données).
Les workflows qui gèrent les services web (aussi appelé d'E-Services) sont appelés des E-Workflows [48]. Les compositions de services qui utilisent cette approche sont normalement basées sur les workflows manuels, aussi appelé statiques. Dans le workflow manuel, l'utilisateur doit définir l'ensemble des tâches et des dépendances parmi les données.
Chaque tâche contient des requêtes qui permettent de chercher des services concrets pour la satisfaire, puis l'exécuter. Dans ce cas, seules la sélection et la liaison du service sont faites automatiquement. Du coté des compositions dynamiques, le modèle du processus ainsi que la sélection du service sont faits automatiquement, une composition automatique demande des workflows capables de reconnaitre les services web correspondants à chaque tâche, mais aussi de trouver d'autres services au cas où ceux-ci soient indisponibles.

2.7 Création de base de test

Un test est un ensemble de cas à tester (état de l'objet à tester avant exécution du test, actions ou données en entrée, valeurs ou observations attendues, et état de l'objet après exécution), éventuellement accompagné d'une procédure d'exécution (séquence d'actions à exécuter). Il est lié à un objectif. Le but d'une base de tests est de modéliser et stocker de manière informatique un ensemble de connaissances, données, idées, concepts ou données et de permettre leur consultation/utilisation. Dans le cadre de notre projet nous s'intéressons à la création de base de tests qui permet à des chercheurs de vérifier leurs propositions de composition de services web.

2.8 Techniques d'extraction de données

2.8.1 Les Crawler

Depuis sa création au début des années 1990, le World Wide Web, communément appelé Web, a connu une croissance exponentielle. De quelques centaines de sites en 1993, sa taille estimée dépasse aujourd'hui les 45 milliards de pages1. Effet direct de cette croissance : les données qu'il contient sont progressivement devenues extrêmement intéressantes pour les entreprises. Un nouveau défi est alors apparu, il leur faut apprendre à recueillir, comprendre et exploiter les informations issues de cette source quasi inépuisable et sans cesse renouvelée. On observe, par exemple, la multiplication des comparateurs pour tous les types de produits (assurances, banques, produits culturels, etc.) qui centralisent les données collectées de différents sites. On peut également imaginer une enseigne de supermarché étudier la concurrence de façon automatique pour ajuster ses prix. C'est dans le cadre de cette problématique que s'inscrivent les concepts de crawling et scraping, outils majeurs de la collecte de données sur le Web.

2.8.1.1 Définition

Le crawler est un logiciel, programme ou script informatique qui explore automatiquement le World Wide Web d'une manière méthodique et ordonnée. Il est généralement conçu pour collecter les ressources (pages Web, images, vidéos, documents Word ou PDF, etc.) afin de permettre à un moteur de recherche de les indexer. Différentes appellations existent, certaines sont en langue anglaise Web crawler ou Web spider et d'autres sont en français explorateur du Web, robot d'indexation ou littéralement araignée du Web. Le crawler est géré par un moteur de recherche pour construire un résumé du contenu d'un site Web (index de contenu) [15]. Le crawler crée un résumé textuel du contenu des adresses URL de chaque page du site (dictionnaire de mots). Donc il fait que l'indexation, du contenu textuel des sites Web par exemple des images (tels que des photos, graphiques, textes), des éléments vidéo et Flash nécessitent un HTML-Texte complémentaire pour être vu par un moteur de recherche. En général, le crawler commence par une liste d'URL à visiter, appelées les graines. En visitant ces URL, le crawler identifie tous les liens hypertexte dans la page et les ajoute à la liste des URL à visiter, appelée la frontière crawl. Les URL de la frontière sont récursivement visitées selon un ensemble de règles.

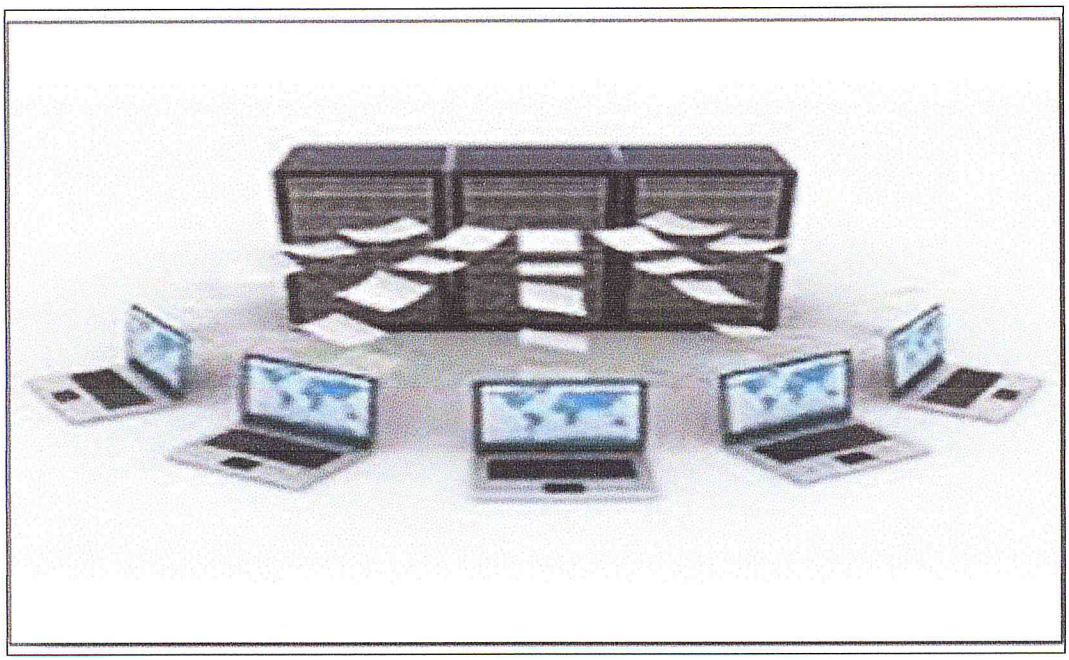


FIGURE 2.4 – Opération de Crawling

Le grand volume de données implique que le crawler ne peut télécharger un nombre limité de pages Web dans un temps donné, il faut donner la priorité à ses téléchargements. Le taux élevé de changement des pages Web implique qu'elles aient déjà été mises à jour ou même supprimées, ceci crée un problème pour les crawlers car ils doivent trier une infinité de combinaisons de sélections HTTP GET basées sur l'URL afin d'avoir des changements relativement infimes de script pour récupérer un contenu unique.

Les moteurs de recherches utilisent les crawlers comme un moyen pour assurer la mise à jour des données et principalement pour créer une copie de toutes les pages visitées pour un traitement ultérieur, suite à une indexation les moteurs de recherches fournissent des recherches rapides.

Les crawlers peuvent également être utilisés pour automatiser les tâches de maintenance sur un site Web, telles que la vérification ou la validation des liens du code HTML. De plus, ils peuvent être utilisés pour recueillir certains types d'informations à partir des pages Web, telle que la récolte des adresses e-mail généralement pour envoyer du Spam.

Comme Edwards et al. [16] ont déclaré : «Étant donné que la bande passante pour la réalisation d'un crawl n'est ni infinie ni libre, il devient essentiel d'explorer le Web, non seulement de façon évolutive, mais efficace, si une mesure raisonnable de la qualité ou la fraîcheur doit être maintenu».

2.8.1.2 Fonctionnement

Pour comprendre comment fonctionne un crawler, il est nécessaire de rappeler le fonctionnement du Web lui même. Celui-ci peut être considéré comme un ensemble de ressources, chacune identifiée par une adresse unique appelée URL2 ou adresse Web. Dans le cas où la ressource considérée est une page web, celle-ci pourra être affichée dans un navigateur web. Chaque page web est susceptible de contenir un certain nombre de liens

vers d'autres ressources, celles-ci pouvant être d'autres pages web. De façon conceptuelle, l'algorithme permettant d'implémenter un crawler est alors très simple. Dans sa version la plus élémentaire, celui-ci n'utilise qu'une seule structure de données : un ensemble d'adresses web connues. La structure ne contient au départ qu'un petit nombre d'adresses web appelées candidats initiaux. Pour chacune de ces adresses, le programme télécharge la page web associée et identifie les liens qu'elle contient. Pour chacun de ces liens, si l'adresse est inconnue, elle est ajoutée à la liste. L'algorithme choisit alors un nouvel ensemble d'adresses à visiter parmi celles qu'il vient de découvrir et répète le processus.

2.8.1.3 Limites

Le fonctionnement du web étant presque le même depuis sa création, les limites rencontrées par les premières implémentations de crawler [17] sont les mêmes que celles auxquelles peuvent être confrontées les entreprises actuelles [18]. Elles se déclinent selon différents aspects.

Performances

Le premier de ces problèmes est celui de la performance. Les premières versions des crawlers ont été confrontées à la croissance exponentielle du Web tandis que de nos jours, c'est plus le volume de données et la fréquence de leur changement qui est mis en cause. Les problèmes de performance se présentent sous trois aspects principaux d'après un article sur les meilleures stratégies de crawling [19]. Tout d'abord, c'est la bande passante et l'espace disque qui limitent la quantité de pages crawlées car ils sont ni infinis, ni gratuits. On estime aujourd'hui à 8x109 pages indexées par Google. Avec une moyenne de 15kB par page, il faudrait 4 millions de dollars pour récupérer et stocker tout le Web. Pour maintenir une base relativement à jour, un cawler doit donc télécharger chaque seconde un nombre de pages pouvant atteindre plusieurs milliers. Au delà de la contrainte de la politesse qui impose de ne pas surcharger les sites que l'on est en train de parcourir, il faut s'assurer que l'infrastructure réseau utilisée par la machine qui héberge le crawler soit suffisante. Ensuite, les pages sont en perpétuel changement. Le crawling est une image à un instant T mais lorsqu'une recherche est effectuée, on peut trouver des liens morts et des pages inexistantes. Le crawler doit donc par moment arrêter de télécharger des nouvelles ressources et vérifier voire mettre à jour les pages déjà indexées. Enfin, en considérant qu'une page est définie par son URL, alors le Web est infini surtout avec l'apparition de pages dynamiques. Afin de conserver de bonnes performances, un crawler doit donc mettre en place une stratégie pour télécharger les pages les plus "importantes" en premier lieu et ne pas oublier de mettre à jour la base de données.

Politesse et bonnes pratiques

Vient ensuite la question de la politesse et des bonnes pratiques du Web. Le but d'un crawler n'est bien évidemment pas de nuire aux sites visités. Cependant, plusieurs facteurs peuvent avoir des effets négatifs sur celui-ci. Dans le cas d'un site comportant un grand nombre de pages, après quelques itérations du crawler, la liste des pages à télécharger peut atteindre plusieurs milliers. Si aucune précaution n'est prise par le crawler et que celui commence le téléchargement simultané de plusieurs pages, le site ciblé est susceptible de subir des ralentissements ou même d'être rendu indisponible. Les mesures utilisées généralement consistent à limiter la bande passante utilisée lors du crawl d'un site ou à ne télécharger qu'une seule page à la fois.

Protection

Pour formaliser ces pratiques, il existe plusieurs façons de limiter voir bloquer l'action des crawlers [20]. Certaines entreprises comme Impreva [21] commercialisent même des solutions mettant en œuvre ces techniques de protection. Celles-ci peuvent prendre divers aspects, à commencer par le plus simple : le fichier "robots.txt" ou la balise HTML "meta robots". Ils permettent de spécifier de façon normalisée quelles parties du site un crawler est autorisé à visiter ou non. L'algorithme utilisé par un crawler respectant la convention du fichier robots.txt reste relativement simple. Avant de télécharger la première page d'un site qui n'a jamais été visité, le crawler télécharge le fichier robots.txt depuis le serveur. Le fichier contient une liste d'adresses interdites. Pour chaque adresse que le crawler découvrira sur ce site, il lui suffira de vérifier qu'elle n'est pas dans la liste de celles qui sont interdites avant de télécharger la ressource associée.

Un site pourra par exemple demander aux crawlers de ne pas visiter des pages qui demandent beaucoup de ressources pour être générées ou qui n'ont aucun intérêt. Ces deux mesures relèvent cependant uniquement de la bonne volonté du crawler, qui pourra toujours décider de les ignorer et parcourir l'intégralité du site. Bien entendu, la majorité des crawlers commerciaux qui parcourent le Web (ex : les moteurs de recherche) respectent ces instructions, protégeant effectivement les sites de la surcharge qui aurait pu être occasionnée par le rendu des pages en question.

Il est également possible pour les sites web de mettre en place des méthodes de filtrage basées sur le "user agent". Toute application qui se connecte à un site web fournit cet identifiant au site sous forme d'une chaîne de caractères. Il est propre à chaque navigateur web, client en ligne de commande, crawler, etc. Un site pourra donc restreindre l'accès à certaines ressources en fonction du user agent fourni lors de la connexion. Ceci ne pouvant se faire de manière immédiate comme avec l'utilisation du fichier robots.txt, l'intérêt des solutions que peuvent proposer des entreprises comme Impreva commence à se préciser.

Cependant, rien n'empêche un crawler de fournir un user agent de navigateur web afin de se faire passer pour un utilisateur réel. D'autres méthodes plus avancées existent pour contrer les crawlers. On pourra par exemple détecter de trop nombreuses connexions depuis la même machine et bannir son adresse IP pour empêcher toutes tentatives de crawl ultérieures. Authentifier les utilisateurs pour accéder à certaines parties du site est également une bonne méthode pour s'assurer qu'elles ne seront pas visitées par un robot. L'utilisation du javascript pour générer certaines parties des pages web pourra empêcher un crawler d'y avoir accès et sera complètement transparent pour un utilisateur réel. De telles méthodes seront plus utilisées pour empêcher la collecte d'information par du scraping, que nous verrons plus en détail par la suite, plutôt que pour empêcher le

crawling.

2.8.1.4 Architecture physique

Le Web ayant connu une croissance exponentielle à ses débuts, les implémentations simplistes se sont très vite retrouvées insuffisantes. Pour augmenter les capacités des crawlers, il a donc été nécessaire de réfléchir à de nouvelles implémentations permettant d'utiliser plusieurs machines afin de répartir la charge de travail. Typiquement, le stockage des données récoltées par les crawlers sera répartie entre plusieurs machines qui pourront se situer dans des lieux géographiques différents. Par exemple, les données issues du crawling de sites web de chaque continent seront stockées dans des datacenter locaux. De la même façon, les machines effectuant le crawling auront tendance à être placées de façon optimisée par rapport aux sites ciblés. Mettre en place une architecture décentralisée impose cependant de répondre à de nouvelles contraintes. Il faut s'assurer que les sites visités par les différentes machines exécutant le crawler sont bien disjoints. Ceci pourra se faire en partitionnant l'espace des adresses web et en assignant chaque partie à une instance différente du crawler. Ce type d'architecture physique peut être modélisé par le schéma suivant [22] :

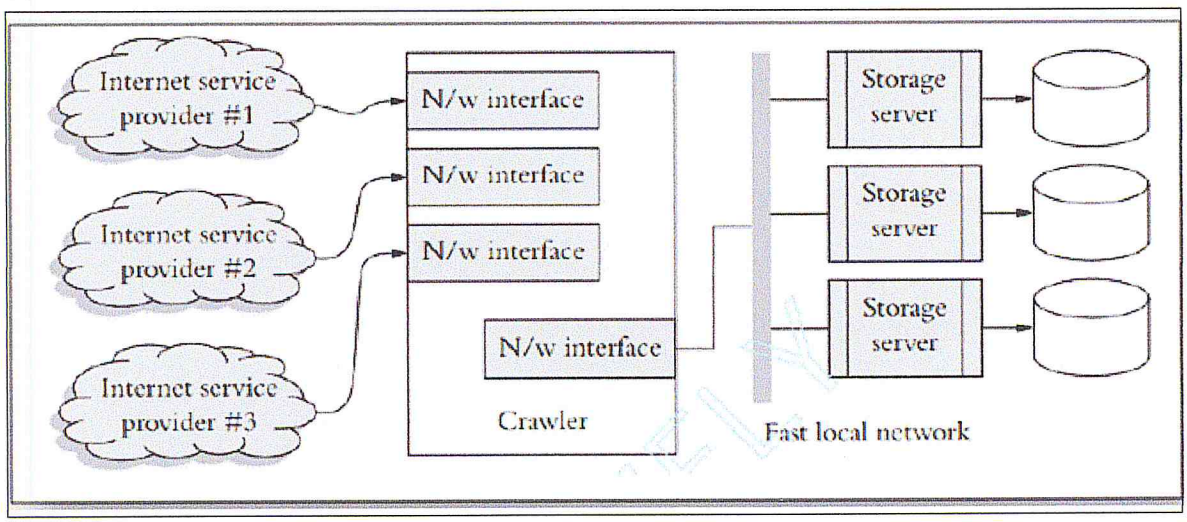


FIGURE 2.5 – Architecture physique d'un crawler [22]

Dans ce cas, le crawler en lui-même n'utilise qu'une seule machine mais plusieurs interfaces réseau afin de bénéficier d'une plus grande capacité de téléchargement. Le stockage des informations récoltées est effectué sur plusieurs machines différentes accessibles en réseau local. De cette façon, les temps d'écriture des données sont moins limitants. Cette architecture très similaire à celle proposée dans l'article de Marc Najork [17]. En effet, celui-ci décrit également une architecture dans laquelle le processus de téléchargement est distribué et les données réparties entre plusieurs serveurs. Cependant, dans son cas, le crawler lui-même est réparti entre plusieurs machines, ce qui n'est pas le cas ici. La solution qu'il propose semble donc encore plus évolutive et flexible que celle présentée en (figure 3.2) .

2.8.1.5 Architecture logicielle

On peut modéliser les différents composants d'un crawler de la façon suivante [23] :

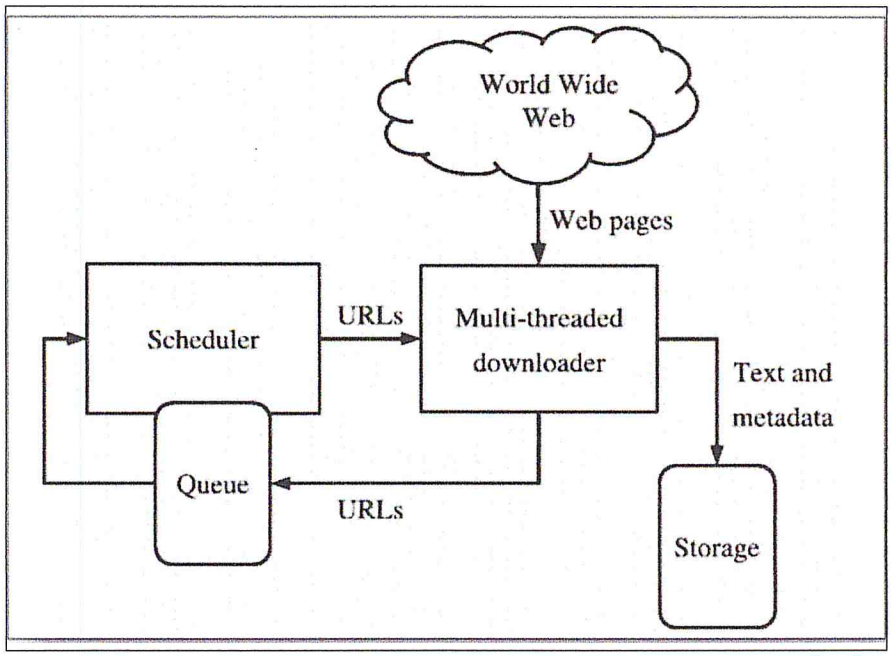


FIGURE 2.6 – Composants logiciels d'un crawler [23]

Cette représentation assez haut niveau représente la base requise pour créer un crawler extensible et efficace. L'utilisation d'un ordonnanceur (scheduler) pour prioriser les URLs à visiter permet de mieux gérer les différents temps morts entre le téléchargement des pages. Soumen Chakrabarti [22] propose par exemple de grouper les pages d'un même site web et d'utiliser le temps nécessaire à leur téléchargement pour émettre les requêtes de résolution DNS des URLs qui suivent dans la file. Ainsi, celles-ci seront prêtes à être visitées quand leur tour viendra et le téléchargement pourra commencer directement.

2.8.1.6 Utilisations

Comme précisé en introduction, le but premier des crawlers était de mesurer et cartographier le Web. Le contenu des pages téléchargées par les crawlers n'était alors utilisé que pour découvrir de nouvelles URLs. Rapidement, les exemples de programmes utilisant les données issues de crawlers sont multipliées. Si le fonctionnement de base restait le même, le contenu des pages était cette-fois analysé, notamment à des fins d'indexation : c'est la naissance des moteurs de recherche. Le nombre de pages téléchargées par un crawler pouvant atteindre de très grands nombres, une fois que des données ou meta-données en sont extraites, il est très souvent intéressant d'analyser ces dernières. Entrent alors en jeu des techniques de data mining permettant d'en extraire des informations. Ces informations peuvent ensuite permettre de créer d'innombrables services : comparateur de prix, aggrégateur de données, annuaires... L'analyse des données téléchargées par le crawler peut également aider celui-ci à mieux prioriser les URLs à visiter. Il est en effet possible de créer différents indicateurs de valeur potentielle d'une URL à partir des données de la page qui la contient ou les autres pages d'un même site. En procédant de la sorte, le crawler pourra choisir de laisser de côté certaines URLs et d'en visiter certaines autres en priorité. Ainsi, la valeur apportée par le programme est supérieure à celle qu'il aurait dans le cas d'une approche plus naïve. Cette façon de procéder est appelée "focused crawling". L'article de recherche de S. Chakrabarti [24] explique la méthode utilisée pour choisir les liens à suivre. La solution contient trois composants :

— Un classifié qui détermine la catégorie à laquelle appartient une page en le comparant à un ensemble de page dont les catégories sont déjà connues. En réalité, une page appartient souvent à plusieurs catégories mais leur solution est une première approche.
— Un distiller qui défini la valeur d'une page en fonction de sa catégorie et les liens qu'elle contient.

— Enfin le crawler qui interroge le classifier et le distiller pour choisir les liens à suivre.
— Ainsi un ensemble de catégories est choisi au début du crawling et les pages téléchargées sont en principe toutes en rapport avec les sujets recherchés.

2.8.1.7 Applications Crawling

Il y a différents scénarios dans lesquels les crawlers sont utilisés pour l'acquisition des données. Nous allons décrire dans la suite quelques exemples de stratégies utilisées dans le crawling.

Crawling en profondeur d'abord

Afin de construire un moteur de recherche important ou un grand dépôt comme l'archive d'Internet [25], les crawlers commencent avec un petit ensemble de pages, puis ils crawlent d'autres pages en suivant les liens dans un mode en "profondeur d'abord". En réalité, les pages web ne sont pas souvent parcourues dans un mode en profondeur d'abord stricte, mais en utilisant une variété de politiques, par exemple, pour élaguer les crawls à l'intérieur d'un site web ou pour crawler les pages les plus importantes en premier.

Recrawling des Pages pour des mises à jour

Une fois les pages sont initialement acquises elles peuvent être périodiquement recrawlées et contrôlées pour des mises à jour. Pour le faire, il faut soit commencer un autre crawl en profondeur d'abord ou tout simplement revisiter toutes les URLs collectées. Cependant une variété de méthodes heuristiques peut être utilisée pour recrawler les pages les plus importantes, les sites ou les domaines les plus fréquents. De bonnes stratégies de recrawling sont cruciales pour le maintien d'une mise à jour d'un index de recherche avec une bande passante limitée. Des travaux récents de Cho et Garcia-Molina [26] ont étudié les techniques d'optimisation de la «fraîcheur» de telles collections selon des observations données au sujet de l'histoire de mise à jour d'une page.

Crawling axé (ciblé)

Des moteurs de recherche plus spécialisés peuvent utiliser les politiques de crawling qui tentent de se concentrer uniquement sur certains types de pages. Par exemple, des pages sur un sujet particulier ou dans une langue particulière, des images, des fichiers mp3, ou de documents de recherche en sciences informatiques. En plus des heuristiques, des approches plus générales ont été proposées sur la base de l'analyse de structure des liens [28, 27] et les techniques d'apprentissage automatique [29, 30, 31]. L'objectif d'un crawler ciblé ou axé est de trouver de nombreuses pages d'intérêt sans utiliser beaucoup de bande passante. Ainsi, la plupart des travaux antérieurs n'utilisaient pas un crawler à haute performance, même si cela pourrait supporter de grandes collections spécialisées qui sont significativement à jour d'un moteur de recherche général.

La marche aléatoire et l'échantillonnage Plusieurs techniques ont été étudiées qui utilisent des marches aléatoires sur le graphe du web pour échantillonner des pages ou estimer la taille et la qualité des moteurs de recherche.

Crawling le "Web Invisible

Un grand nombre de données accessibles via le web réside en fait dans les bases de données et ne peut être récupéré en affectant des requêtes appropriées et/ou remplir des formulaires sur des pages Web. Récemment, un grand intérêt a porté sur l'accès automatique à ces données aussi appelé le « Web Invisible » ou « Web Profond », ou « Faits et chiffres fédérés ». Un travail dans [32] a étudié les techniques de crawling pour ces données. Un crawler tel qu'il est décrit ici pourrait être étendu et utilisé comme un frontal efficace pour un tel système. Notons, toutefois qu'il y a beaucoup d'autres défis liés à l'accès au Web invisible, et l'efficacité de l'extrémité frontale n'est probablement pas le problème le plus important.

2.8.1.8 Exigences pour un crawler

Nous allons maintenant discuter les exigences et les approches pour un bon crawler.

Flexibilité : il est préférable d'utiliser dans le système une variété de scénarios, avec moins de modifications possibles.

Un coût faible et une haute performance : le système doit évoluer pour atteindre au moins plusieurs centaines de pages par seconde, des centaines de millions de pages par

cycle d'exécution et devrait fonctionner sur du matériel peu coûteux. Une utilisation efficace pour l'accès au disque est cruciale afin de maintenir une vitesse élevée auprès des structures de données principales, telles que les "URLs visitées" la structure et la frontière crawl deviennent trop grandes pour la mémoire centrale. Cela se produira seulement après le téléchargement de plusieurs millions de pages.

Robustesse : Il y a plusieurs aspects ici. Tout d'abord, puisque le système va interagir avec des millions de serveurs, il doit tolérer un code HTML erroné, comportements et configurations anormaux du serveur et de nombreux autres problèmes. Le but ici est d'ignorer avec prudence des pages et des serveurs qui ont un comportement étrange, puisque dans de nombreuses applications, il est possible de télécharger un sous-ensemble des pages. Deuxièmement, étant donné qu'une analyse peut prendre des semaines, voire des mois, le système doit être capable de tolérer les accidents et les interruptions de réseau sans perdre (trop) les données. Ainsi, l'état du système doit être gardé sur disque. Les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) ne sont pas vraiment demandées.

L'étiquette et le control de vitesse : Il est extrêmement important de suivre les conventions standards d'exclusion pour les crawlers (Robots.txt et les robots avec les méta-tags), pour fournir une URL de contact pour le crawler, et de superviser le crawl. En outre, on doit être en mesure de contrôler la vitesse d'accès de plusieurs manières différentes. On doit éviter de mettre trop de charge sur un serveur unique, en contactant chaque site une seule fois toutes les 30 secondes. Il est également souhaitable de ralentir la vitesse au niveau du domaine, afin de ne pas surcharger les petits domaines, ainsi de contrôler la vitesse de téléchargement total du crawling si c'est une connexion partagées par plusieurs utilisateurs.

Gestion et reconfiguration : Une interface appropriée est nécessaire pour surveiller le crawl, y compris la vitesse du crawl, les statistiques sur les hôtes et les pages, et les tailles des ensembles de données principales. L'administrateur doit être en mesure de régler la vitesse, ajouter et supprimer des composants, éteindre le système, forcer un verrouillage, ou ajouter des hôtes et des domaines d'une «liste noire» des liens que le crawler doit éviter. Après un accident ou un arrêt, le logiciel du système peut être modifié pour résoudre les problèmes, et l'on peut vouloir continuer l'exploration en utilisant une configuration de machine différente. Le principal facteur de ralentissement étant l'analyse des textes afin d'établir un tri de pertinence de résultats, d'autres voies ont été explorées. Les meilleurs résultats ont été obtenus par des méthodes utilisant les propriétés déduites de l'analyse de la topologie d'Internet.

2.8.2 Scraping

2.8.2.1 Fonctionnement

Le principe de base du scraping est simple : transformer des informations non structurées présentes dans des pages web en données structurées facilement exploitables. Le programme qui met ce concept en œuvre est généralement appelé “scraper”. Contrairement à un crawler qui découvre lui-même les sites parcourus et les pages web téléchargées, le scraper travaille sur un site ou ensemble de sites connus par avance. Le scraper pourra alors être un logiciel commercial paramétré de façon à récupérer les données souhaitées

sur le site en question ou bien un programme développé spécifiquement pour cette tâche et donc parfaitement adapté au site. Suivant le type, la quantité d'informations à récupérer ainsi que la complexité pour y accéder, les architectures logicielles et matérielles nécessaires à la mise en place d'un scraper varient d'un extrême à l'autre. Ainsi les démonstrations effectuées par Andrew Peterson [33] et Charles Severance [34] nécessitent uniquement un ordinateur personnel et le code nécessaire à l'implémentation de leurs scrapers est très limité. Dans les deux cas, l'information récupérée provient d'un seul site web et est utilisable de façon quasi immédiate.

2.8.2.2 Mise en œuvre

Comme nous venons de le voir, les objectifs d'un scraper peuvent varier d'un extrême à l'autre. En conséquence, les moyens employés pour la mise en oeuvre sont très variés. Ondistingue cependant deux grandes catégories : les logiciels prêts à l'emploi et les librairies permettant de développer des scrapers. Dans chacune de ces catégories, le panel de solution est également très large.

Logiciels prêts à l'emploi

Les premiers logiciels de cette catégorie sont les navigateurs internet. En effet, en visualisant la page et avec de simples copier/coller, un utilisateur peut récupérer les données d'une ou plusieurs pages afin de les réutiliser dans un autre contexte, ce qui correspond à la définition du scraping. Cette méthode étant bien entendu très fastidieuse et peu efficace, il existe des addons pour navigateurs permettant de faciliter ce processus (ex : DownThemAll3 pour Firefox). Viennent ensuite toutes les solutions logicielles, professionnelles ou non, qui une fois paramétrées correctement peuvent se charger de toutes les étapes d'acquisition des données structurées.

Librairies

La plupart des langages de programmation disposent au moins d'une librairie basique permettant de travailler avec le web. La bibliographie étudiée comporte un certain nombre d'exemples d'utilisation de ces librairies dans plusieurs d'entre eux : PHP [35], java [22] ou encore Python [36]. En plus de ces librairies basiques permettant uniquement de télécharger des pages web, il existe de nombreux exemples permettant d'implémenter rapidement le reste du travail d'un scraper : parsing des pages, exécution du javascript, gestion de l'authentification...

Limites

Le support de travail des scrapers étant le même que les crawlers, les pages web, on retrouve la plupart des limites qui s'appliquaient à ces derniers. Le scraping étant globalement plus contraignant que le crawling (nous rappelons qu'il est nécessaire de connaître par avance la structure du site ciblé), il existe des mesures spécifiques pour l'empêcher. Certains sites changent régulièrement la structure du code html de leurs pages. S'il est possible que cela ne change rien visuellement dans un navigateur web, cela va empêcher le programme d'en extraire les données. Bien entendu, le créateur du scraper pourra toujours le corriger pour prendre en compte les modifications mais il ne sera pas à l'abri d'autres changements ultérieurs. Ce type de mesure ne change en revanche

absolument rien au fonctionnement d'un crawler qui se contente de détecter les liens présents dans une page, quelle que soit la structure html de celle-ci.

2.8.2.3 Architecture physique

Contrairement aux crawlers pour lesquels le domaine exploré est virtuellement illimité, les scrapers sont destinés à agir sur un nombre réduit et connu par avance de sites web. S'il sera possible de rencontrer des crawlers utilisant un parc de plusieurs milliers de machines, un scraper n'aura en général besoin que d'une seule machine, en tout cas pour la partie chargée du téléchargement des données.

2.8.2.4 Architecture logicielle

Le scraping étant très spécifique aux sites et données cibles, il est difficile de donner une architecture applicative type. On retrouvera cependant toujours certains composants, parfois similaires à ceux identifiés dans les crawlers. La finalité d'un scraper étant de télécharger des ressources, on retrouvera donc certainement une file d'URLs en attente de téléchargement. Pour de meilleures performances, ils auront aussi tendance à utiliser des téléchargements simultanés grâce au multithreading, exactement à la façon des crawlers. Autre élément fortement similaire aux crawlers : un scraper devra stocker les informations recueillies et disposera donc d'un module permettant la persistance, que celle-ci se fasse sur disque, en base de données, ou en réseau.

2.9 Conclusion

Dans ce chapitre, nous avons présenté une idée globale sur la composition des services web. Nous avons présenté le problème de composition en montrant les types de composition des services web et nous avons détaillé une des approches de composition des services web. Ensuite nous avons décrit les bases de tests et présenté une méthode de création de base de tests et deux techniques relatives a cette méthode. Dans le prochain chapitre nous allons faire une conception de notre solution proposée pour la création de bases de tests et leurs exploitation.

Chapitre 3

Conception

3.1 Introduction

Dans les chapitres précédents, nous avons présenté les concepts fondamentaux des services web et les méthodes de création de bases de tests dans le cadre des services web, alors que nous avons présenté la composition des services web, ainsi les techniques d'extractions des compositions (mashup) à partir d'un site web. Dans ce chapitre nous allons présenter une description de l'architecture générale et une conception détaillée de la solution proposée.

3.2 Présentation d'une approche proposée pour la composition de services REST

Dans cette section nous allons choisir une approche de compositions de services REST qui est « approche basé sur workflow » présentée dans le chapitre 2. Avant de passer au détail il est à comprendre que pour chaque service REST :

- Les ressources sont publiées par Origin Servers.
- Le Proxy/Gateway sont pour caching,acces control, adaptation.

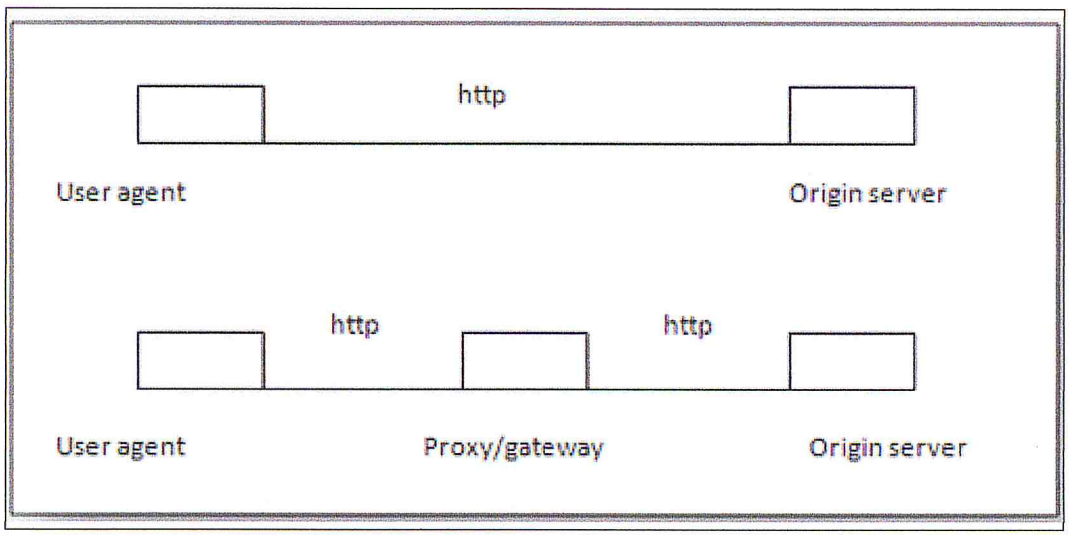


FIGURE 3.1 – Élément de service REST

Alors la composition de plusieurs services REST consiste à composer les ressources publiées par origin server de chaque service comme présenté dans les (figures 3.2 et 3.3) :

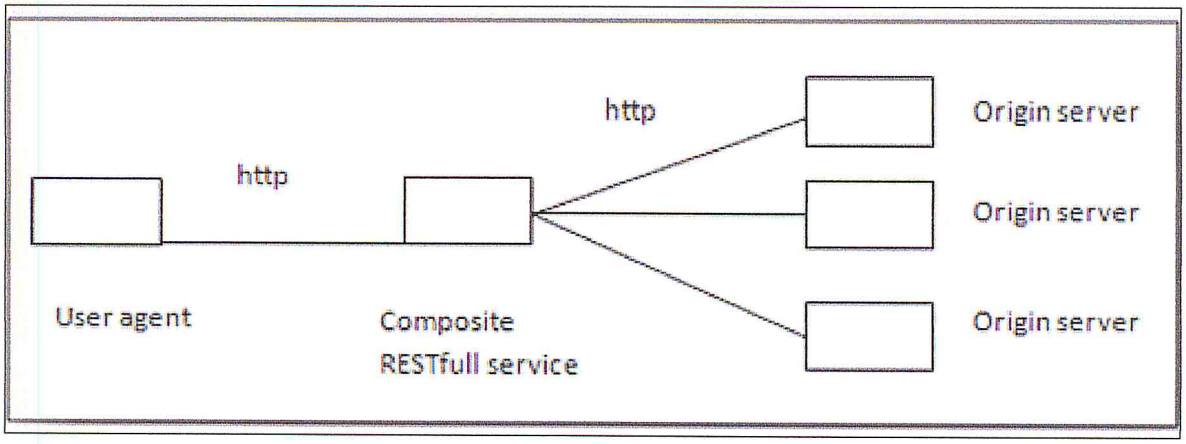


FIGURE 3.2 – Composition des origin server

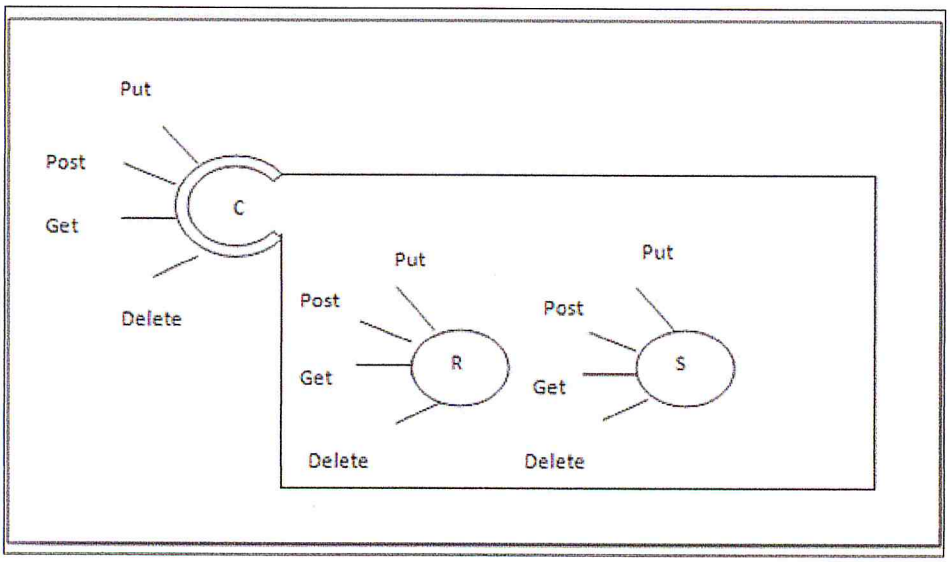


FIGURE 3.3 – Composition de plusieurs ressources

3.2.1 Approche basé sur les workflows

Pour illustrer notre travail nous allons effectuer des compositions des services REST utilisant une approche basée sur workflow via le langage de composition visuel JOpera. Ce langage fournit une notation graphique aux workflows de modèle en termes de dépendance de flux de controles et de transfert de flux de données graphiques,comme a été montionné par cesare pautassou[37].

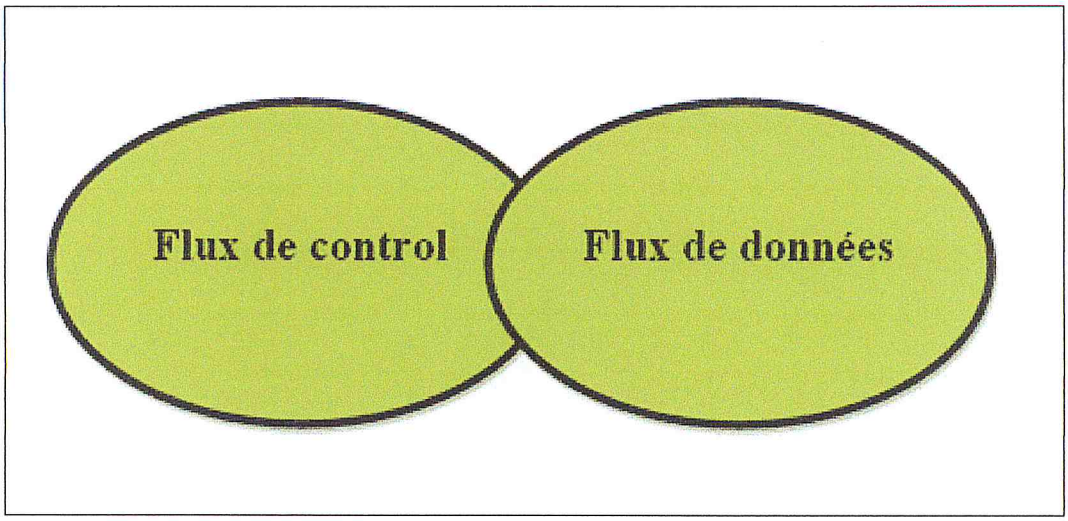


FIGURE 3.4 – Coopération des éléments du workflow

Flusx de contrôle : Il permet de visualiser l'ordre d'exécution partiel des tâches d'un workflow. Les tâches sont affichées comme les nœuds d'un graphe liés par des arêtes représentant les dépendances du flux de contrôle.

Flux de données : : JOpera fournit une représentation graphique définissant la manière dont les données circulent entre les différents paramètres d'entrée et de sortie des tâches de workflow. De cette façon, il est possible de visualiser séparément l'ordre d'exécution des tâches à gros grains et de leurs échanges de données à granularité fine. Concernant la syntaxe du graphe de flux de données, les tâches sont affichées avec des paramètres d'entrée et de sortie qui les entourent et liés à la tâche avec des paramètres d'entrée et des paramètres de sortie (InPut/OutPut). Les paramètres des tâches sont affichés en blanc, tandis que les paramètres des adaptateurs liés aux tâches sont affichés en gris (et étiquetés avec le préfixe SYS). Les arêtes à tête noire représentent une opération de transfert de données entre les paramètres de sortie et d'entrée des tâches, qui est exécuté comme une tâche est sur le point de commencer, le graphe de flux de données peut contenir des boucles.

Dans ce qui suit nous présentons trois mashups qui sont : doodlemap mashup, maptube mashup et yahoo local search mashups

3.2.1.1 DoodleMap mashup

Ce mashup est composée des services REST : Yahoo! Service de recherche local et Doodle service de sondage avec l'api Google Map.

Flux de contrôle

La figure suivante représente le flux de contrôle du mashup doodle map

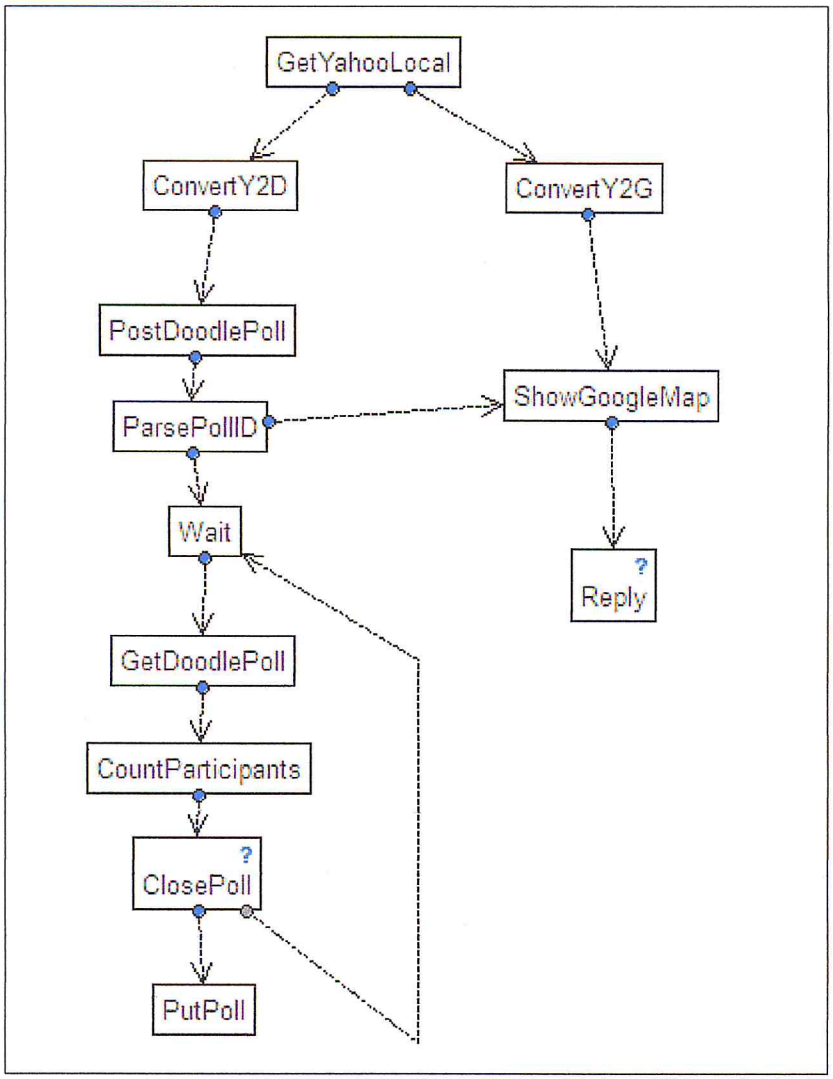


FIGURE 3.5 – flux de contrôle de "DoodleMap mashup"

Le workflow commence par une requête GET adressée à Yahoo ! Service de recherche local, représenté par la tâche GetYahooLocal. Avant les résultats de la recherche peut être utilisé dans le mashup ils sont convertis dans un format adapté pour être représenté sur un Google Map (ConvertY2G) et pour être utilisé pour créer un nouveau sondage Doodle (ConvertY2D). Les deux tâches de conversion sont exécutées en parallèle car il n'y a pas de dépendance de flux de contrôle entre elles.
Une fois la les résultats ont été convertis, l'exécution se poursuit avec une requête POST sur le Doodle API pour créer un nouveau sondage (PostDoodlePoll). Les en-têtes de la réponse renvoyés par Doodle sont analysés pour extraire le lien hypertexte identifiant le nouveau créateur de la ressource de sondage, ainsi que la clé d'autorisation pour l'administrer. Une fois ces informations disponibles, la tâche ShowGoogleMap est prête à être exécutée, comme toutes les informations nécessaires pour créer l'interface utilisateur du mashup sont disponibles.

La deuxième partie du workflow permet de suivre l'état du sondage et le fermer une fois que suffisamment de participants ont répondu. Ceci est fait avec une boucle des tâches :

— Attendre un certain temps.
— Effectuer une requête GET pour récupérer l'état actuel de la ressource de sondage (GetDoodlePoll).
— Compter le nombre des participants qui ont répondu (CountParticipants).

La boucle est répétée s'il n'y a pas assez de participants (cette condition déclenche le retour de ClosePoll à la tâche d'attente). Sinon, l'exécution continue à la tâche ClosePoll, qui modifie la copie locale de l'état du sondage et se termine après la tâche PutDoodlePoll a transféré l'état modifié à service Doodle.

Flux de données

La figure suivante représente le flux de contrôle du mashup DoodleMap .

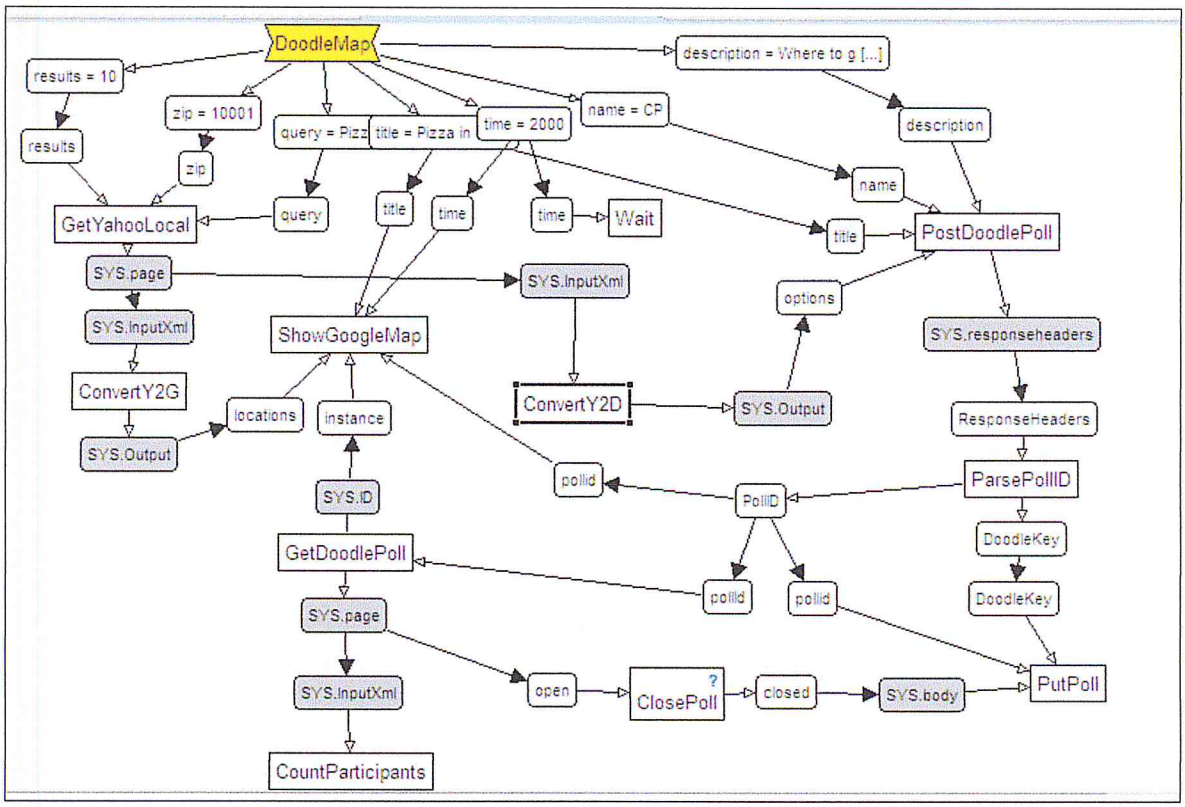


FIGURE 3.6 – flux de données de "DoodleMap mashup"

Les paramètres d'entrée de l'ensemble du work flow sont attachés à la forme étiquetée DoodleMap. Les valeurs de ces paramètres sont définies au début de l'exécution du workflow. Le workflow de composition peut être exécuté plusieurs fois avec différents paramètres d'entrée. Celles-ci définissent une Sondage DoodleMap, qui peut avoir un titre spécifique, une description, un nom d'auteur, nombre de résultats de localisation alternatifs pour un sujet de requête donné dans un code postal. Une partie de ces paramètres est

transférée pour créer Yahoo ! Requête locale URI dans la tâche GetYahooLocal. D'autres sont utilisés pour initialiser l'état du nouveau sondage créé par la tâche PostDoodlePoll. Le paramètre time permet de configurer le taux de rafraîchissement de l'interface utilisateur (représentée par la tâche ShowGoogleMap), il est également utilisé pour spécifier l'interval d'interrogation de la boucle de workflow.

Les résultats XML extraits de Yahoo ! sont stockés dans le Paramètre de sortie SYS.page de l'adaptateur HTTP lié à la tâche GetYahooLocal. Ces résultats sont copiés dans le paramètre SYS.InputXML de l'adaptateur XSLT lié aux deux tâches de conversion. Les résultats des transformations sont stockés dans les paramètres SYS.Output correspondants et sont maintenant dans un format plus approprié pour initialiser les options du nouveau sondage et configurer les emplacements des marqueurs à afficher sur la carte.

Une fois le sondage créé, l'API Doodle renvoie un lien hypertexte pour identifier la nouvelle ressource d'interrogation et une clé d'accès à utiliser pour administrer le sondage. Ceux-ci sont trouvés resp. Dans les en-têtes Location et x-DoodleKey de la réponse HTTP. Tous les en-têtes sont stockés dans le paramètre de sortie SYS.responseheaders de l'adaptateur HTTP lié à la tâche PostDoodlePoll. La tâche suivante ParsePollID est responsable de l'extraction des valeurs des deux réponses les en-têtes et les stocker dans ses paramètres de sortie PollID et DoodleKey. Le lien vers la ressource d'interrogation est envoyé à trois tâches (ShowGoogleMap, GetDoodlePoll, PutPoll), qui l'utilise pour afficher le sondage dans l'interface utilisateur, récupère l'état actuel du sondage pour surveiller le nombre de participants et pour mettre à jour l'état du sondage une fois qu'il est fermé. Seulement pour cette dernière tâche, la clé d'accès est également requise.

Le flux de données entre les tâches constituant la boucle de surveillance est utilisé pour transférer la représentation XML de la ressource d'interrogation à la tâche CountParticipants liée à l'adaptateur XPATH et utilise une simple requête XPath pour compter nombre de participants. En outre, la même représentation XML si transmise en tant qu'entrée à la tâche ClosePoll, qui permet de basculer l'état de la ressource de ouvert à fermé. Le résultat est transmis au paramètre d'entrée SYS.body de l'adaptateur HTTP lié à la tâche PutPoll. Afin d'implémenter la visualisation de l'état actuel du sondage sur la carte, nous publions une partie de l'état des compositions en tant que ressource et générons un hyperlien s'y référant. Ceci est ensuite transmis à la tâche ShowGoogleMap, qui l'intègre dans l'interface utilisateur. Une fois que ceci est chargé dans un navigateur Web, un script utilisera le lien hypertexte pour récupérer les données nécessaires du workflow publié en tant que ressource et met à jour le widget map. Plus concrètement, ceci est réalisé en identifiant la tâche qui stocke les informations requises (dans notre cas la tâche GetDoodlePoll, qui récupère l'état de la ressource d'interrogation et le stocke dans le mashup) et en connectant sa propriété d'identifiant SYS.ID à la tâche qui contient le code d'interface utilisateur. Ainsi, une fois le workflow instancié, la page Web produite par la tâche ShowGoogleMap contiendra un lien pouvant être utilisé pour récupérer l'état de la ressource d'interrogation mise en cache dans le workflow.

3.2.1.2 MapTube mashup

Ce mashup composé des deux services REST youtube et Googlemap consiste à afficher des vidéo sur youtube d'une ville choisit avec map.

Flux de contrôle

La figure suivante représente le flux de contrôle du mashup « MapTube »

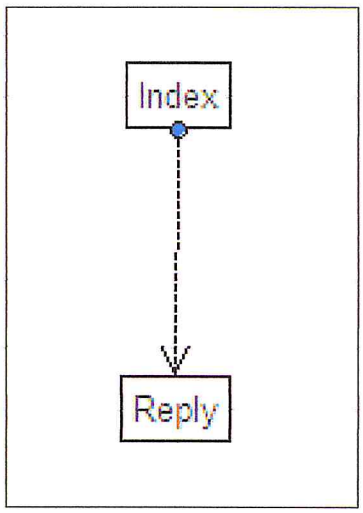


FIGURE 3.7 – flux de contrôle du Maptube mashup

Flux de données

La figure suivante représente le flux de données du mashup Maptube .

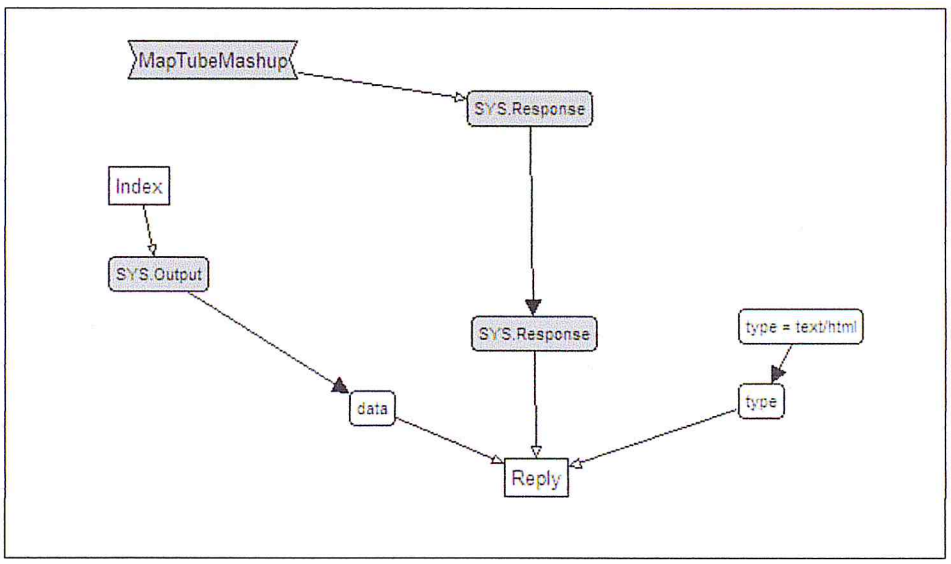


FIGURE 3.8 – flux de données du Maptube mashup

Dans ce diagramme,les taches citées précédemment et leurs Input/Output sont dépendantes et reliées via des arêtes.

3.2.1.3 Yahoo Local Search Map Mashup

Ce mashup sert à afficher les résultats d'une requête à yahoo local search sur Google map, donc ce mashup est composé des deux services REST : yahoo et googlemap

flux de contrôle

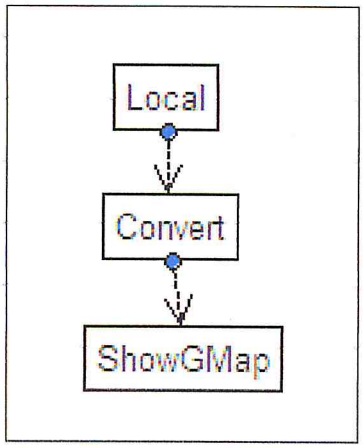


FIGURE 3.9 – flux de contrôle du mashup "yahoo local serach"

flux de données

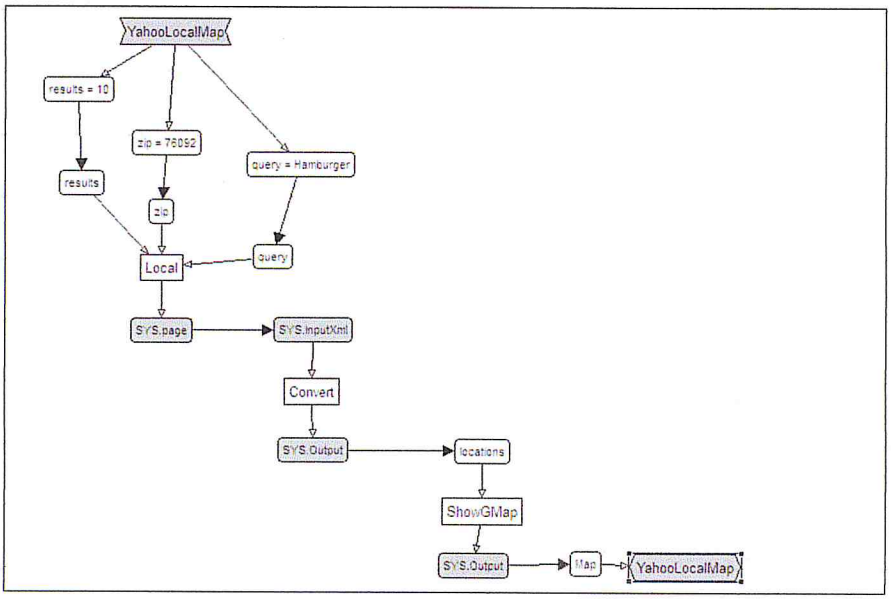


FIGURE 3.10 – flux de données du mashup "yahoo local serach"

3.3 Architecture générale de la solution proposée

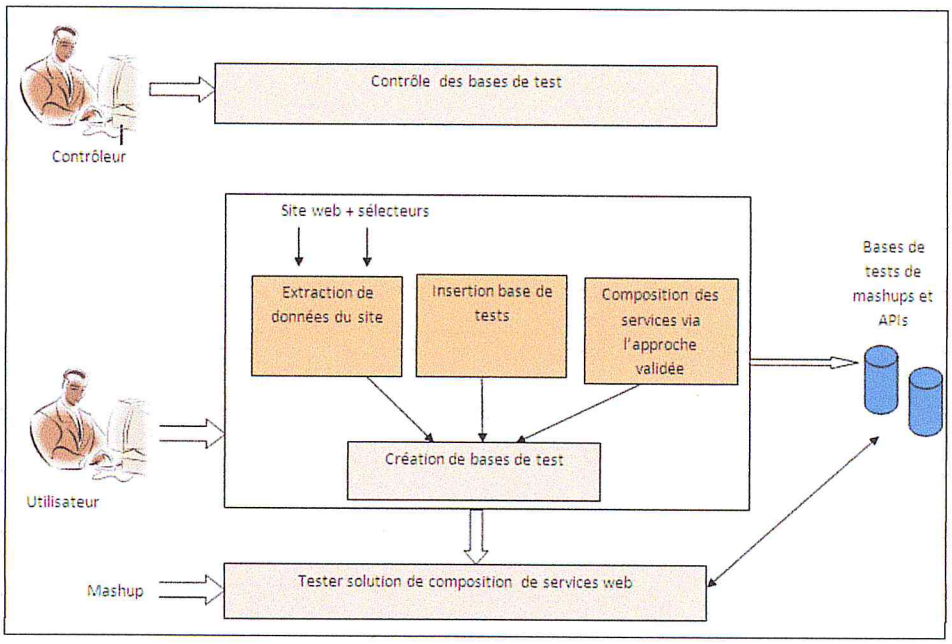


FIGURE 3.11 – Architecture générale de la solution proposée

3.4 Diagramme de cas d'utilisation général de la solution proposée

La (figure 3.12) représente un diagramme de cas d'utilisation général de la solution proposée.

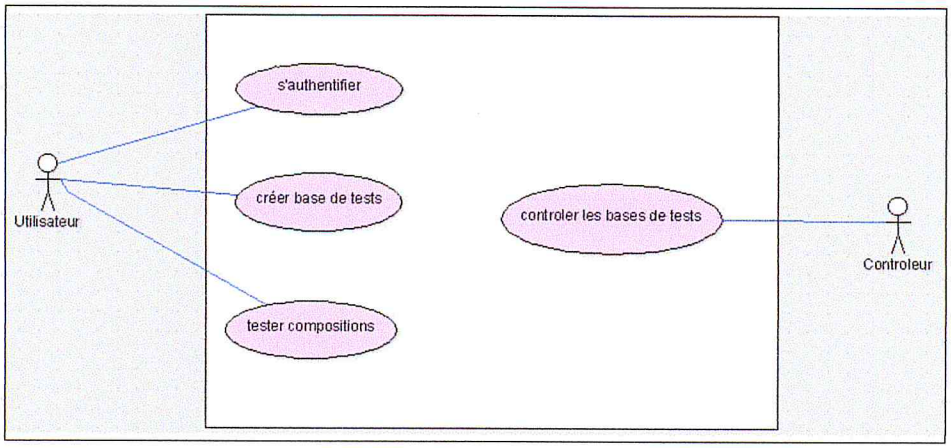


FIGURE 3.12 – Diagramme de cas d'utilisation général de la solution proposée

Ce diagramme écrit en UML représente les fonctions effectuées par les acteurs de notre solution proposée tel que :

Utilisateur : peut être un développeur ou bien un chercheur, il doit être un expert dans le domaine des services web. L'utilisateur effectue les fonctions suivantes après une authentification :

-Créer des bases de tests.

-Tester sa proposition de composition de services (mashup).

Controleur : responsable du controle des bases de tests.

3.5 Diagramme de séquence général

La (figure 3.13) représente un diagramme de séquence montrant le séquencement des fonctions citées précédemment.

Interface user : est une interface homme-machine qui permet l'utilisateur d'interagir avec le système pour exécuter les différentes taches.

Base de données : stockage des mashups et APIs

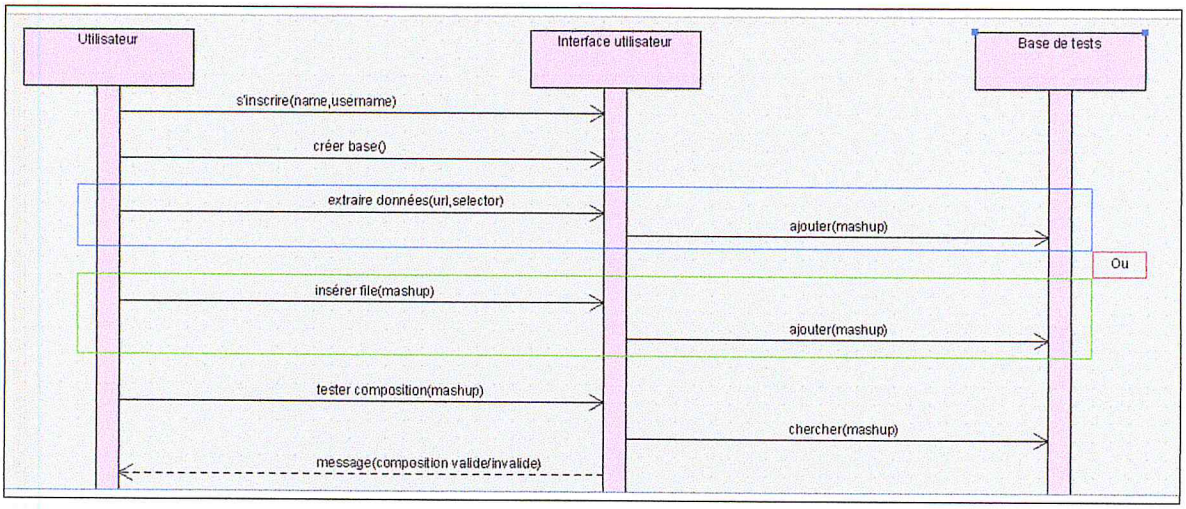


FIGURE 3.13 – Diagramme de séquence général de la solution proposée

3.6 Explication détaillée des modules de la solution proposée

3.6.1 Création de bases de tests

Ce module permet de construire de bases de tests qui aident les utilisateurs à tester leurs solutions de compositions de services web. Le diagramme de cas d'utilisation suivant (figure 3.14) représente les fonctions principales pour la création de bases.

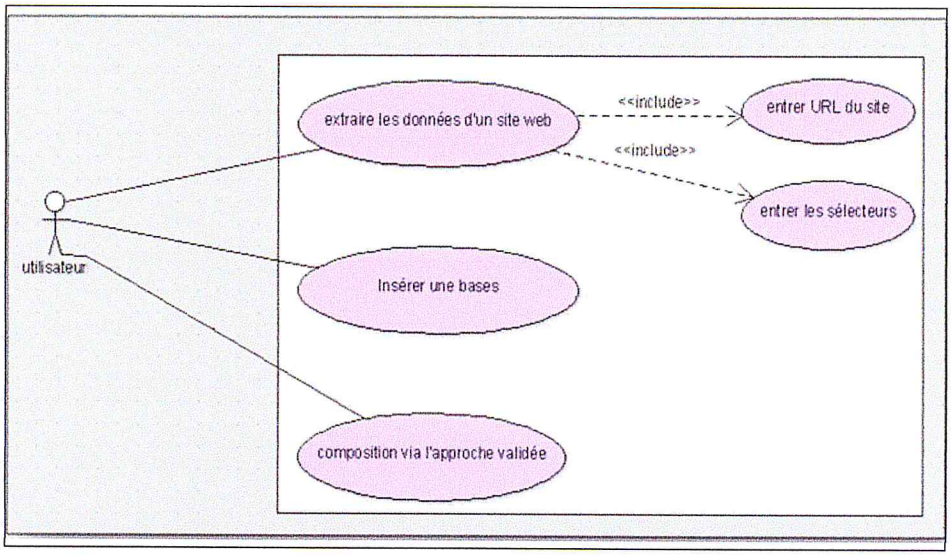


FIGURE 3.14 – Diagramme de cas d'utilisation de la fonction "création bases de tests"

3.6.1.1 Description de « extraction des mashups»

Dans le but de créer une base de tests nous avons établi une extraction des mashups à partir d'une plate-forme en ligne pour les API Web et les mashups de service appelée « ProgrammableWeb.com » Cette Plateforme crée en avril 2008, répertoriait plus de 3 000 mashups et plus de 740 API différentes. Dans notre expérience nous avons essayé d'extraire des mashups en utilisant un scraper, mais les résultats obtenus n'ont pas trop satisfaisants et n'ont pas répondus à nos besoins, car cette plateforme contient un nombre très élevé de pages. La (figure 3.15) représente le fonctionnement du scraper utilisé.

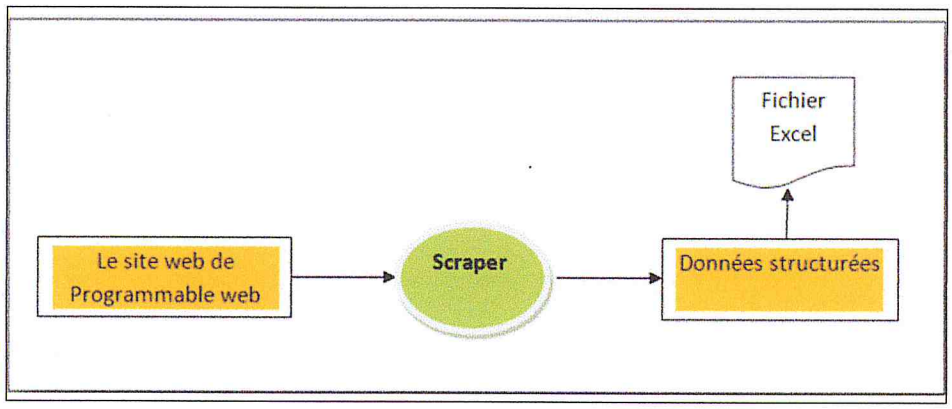


FIGURE 3.15 – fonctionnement du scraper

Et de là nous avons réfléchis à utiliser le crawling en développant un code qui permet d'extraire les mashups en un temps plus ou moins optimal et d'une manière qui répond

aux besoins.

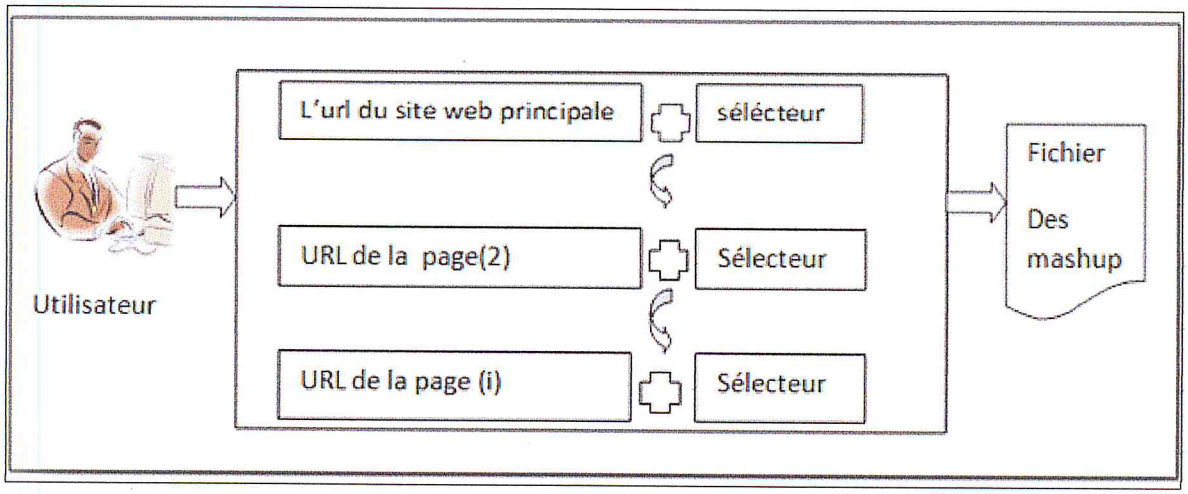


FIGURE 3.16 – Fonctionnement du crawler

Explication de la (figure 3.16) Le crawler est basé sur les url des pages html sur laquelle il va extraire les données, et les sélecteurs qui déterminent les données à extraire. Les mashups extraits et les APIs qui les composent sont stockés dans une base de données pour les futures validations de solution de composition de services web proposées par les chercheurs.

3.6.1.2 Description de « insérer une base »

Cette fonction permet à un utilisateur d'ajouter à la base de test des mashups et les APIs qui les composent, ils peuvent trouver ces mashups par exemple dans des plateformes de mashups, dans le but d'enrichir notre bases de tests. Pour éviter la redondance dans la base de test, l'utilisateur doit effectuer une vérification avant l'ajout.

3.6.1.3 Description de « composition de service via l'approche validée »

Cette fonction permet à un utilisateur de créer des mashups en utilisant l'approche validée dans le module « tester solution de composition » que nous allons le détailler dans la suite de ce chapitre.

3.6.2 Tester la solution de composition

Ce module permet aux chercheurs d'évaluer leurs approches de compositions de services web. La (figure 3.17) représente les étapes effectuées par le chercheur :

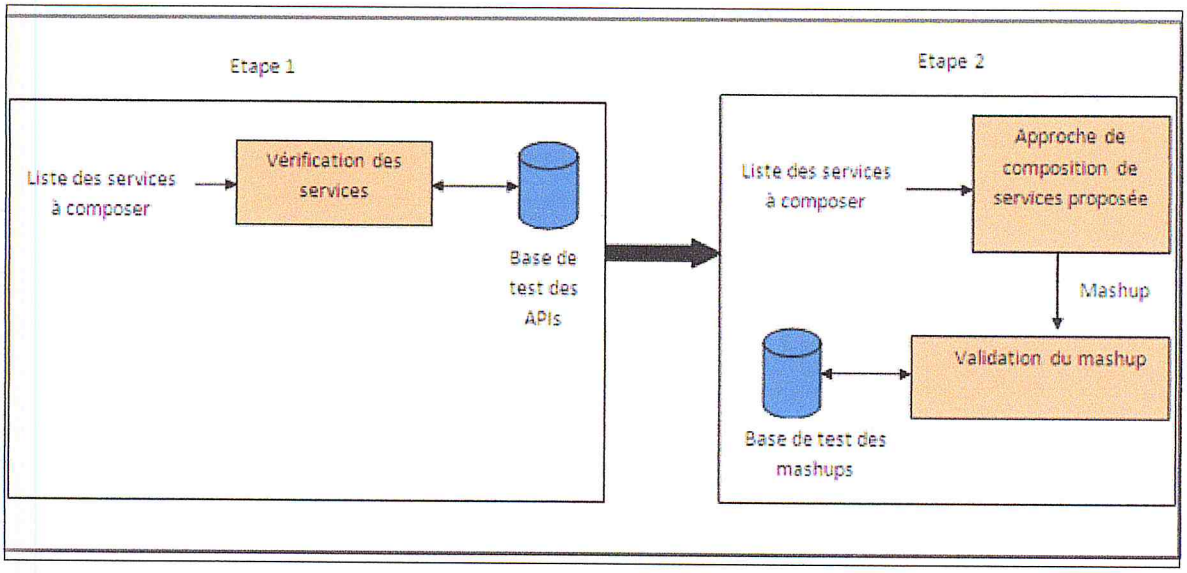


FIGURE 3.17 – Fonctionnement du test

Explication de la figure

— **Etape1 :** vérifier l'existence des services

Le chercheur doit garantir que les services à composer existent réellement dans la base de tests, c'est-à-dire il existe des mashups composées par ces services. Et cela pour ne pas faire des taches inutiles et perdre du temps.

— **Etape2 :** la composition de services

Dans cette étape l'utilisateur effectue des compositions de services (mashups) via une approche qu'il a proposée. Un matching est effectué entre ces mashups et celle qui existes dans la base de tests pour vérifier la validité de l'approche proposée.

3.7 Conclusion

Ce chapitre de conception a permis la compréhension de la création de bases en utilisant le crawling et les approches qui permettent la composition de services destinées à la validation utilisant un matching avec les bases de tests crées . L'étape qui suit est celle de la réalisation de cette solution proposée. Les bases de tests qui sont à base de l'extraction de données et le test de la composition de services proposées par l'utilisateur seront implémentées avec des outils considérés comme référence dans le domaine de développement de web.

Chapitre 4

Implémentation

4.1 Introduction

Après la présentation de l'architecture générale de notre système dans le chapitre précédent, nous allons dans ce chapitre, présenter la réalisation de notre système général.

Pour assurer la fiabilité de notre travail, nous avons fait appel à un ensemble d'outils et un langage de développement permettant la réalisation et la mise en œuvre de notre application. Nous allons ensuite décrire les différentes tâches composants notre application et effectuer des tests.

4.2 Choix de langage, environnements et outils de développement

La réalisation de notre système nécessite des choix en matière d'outils et de langage de développement. Il est question de choisir les outils les mieux appropriés à notre cas. Ces choix sont basés sur la notoriété et l'efficacité des outils et langage dans le domaine du développement d'application web.

4.2.1 Langage de développement

Notre choix du langage de programmation s'est porté sur le langage JAVA et cela pour diverses raisons :

- JAVA est un langage orienté objet simple ce qui réduit les risques d'incohérence.
- Il est portable, il peut être utilisé sous Windows, sur Macintosh et sur d'autres plates-formes sans aucune modification. C'est donc un langage multiplateforme, ce qui permet aux développeurs d'écrire un code qu'ils peuvent exécuter dans tous les environnements.
- Il possède une riche bibliothèque de classes comprenant des fonctions diverses telles que les fonctions standards, le système de gestion de fichiers, les fonctions multimédia et beaucoup d'autres fonctionnalités.
- Possibilité d'importer des bibliothèques en interne comme la bibliothèque **JSOUP** que nous avons utilisé pour réaliser l'extraction.

4.2.2 environnement de développement

Eclipse : est l'environnement de la société IBM, qui a décidé en 2001 de mettre à disposition de la Source l'ébauche d'une plate-forme de développement ouverte, libre extensible, universel et polyvalent, permettant de créer des projets de développement entièrement écrite en Java, grâce à des bibliothèques spécifiques, capable d'intégrer des extensions adaptées à diverses activités débogage, modélisation, interfaces graphiques... [w7]

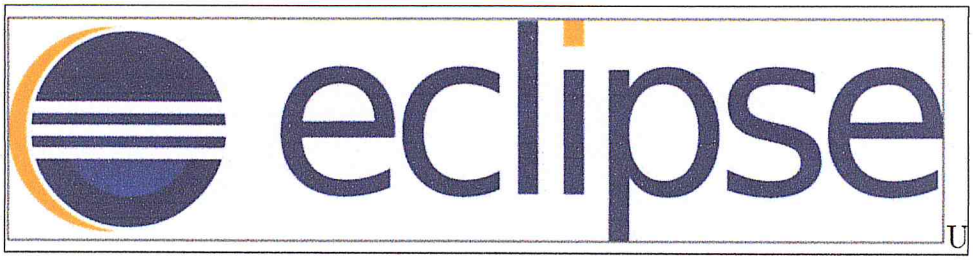


FIGURE 4.1 – L'environnement de développement « eclipse ». [w7]

Netbeans : est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate forme. [w2]

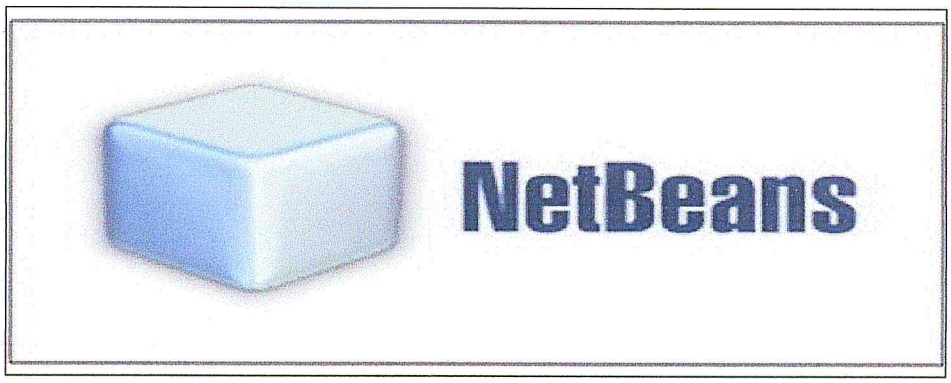


FIGURE 4.2 – L'environnement de développement « NetBeans » .[w2]

4.2.3 outil de développement

Web Scraper : est une extension de navigateur chrome conçue pour scraping (l'extraction de données à partir de pages Web). En utilisant cette extension, nous pouvons créer un plan (sitemap) comment un site Web devrait être traversé et ce qui devrait être extrait.En utilisant ces sitemaps, le Web Scraper naviguera sur le site en conséquence et extraira toutes les données. Les données récupérées plus tard peuvent être exportées au format CSV.

JOpera : JOpera for Eclipse est un outil de composition de services rapide offrant une plate-forme d'exécution visuelle et autonome pour la création d'applications distribuées à partir de services réutilisables, notamment les services Web. Il est livré avec plusieurs plugins, regroupés en différentes fonctionnalités.[w6]

4.3 La description des mashups proposées

les mashups proposées ont été réalisés sous Eclipse en utilisant le plugin JOpera, pour chaque mashup, des InPut/OutPut et des taches ont été affectées.

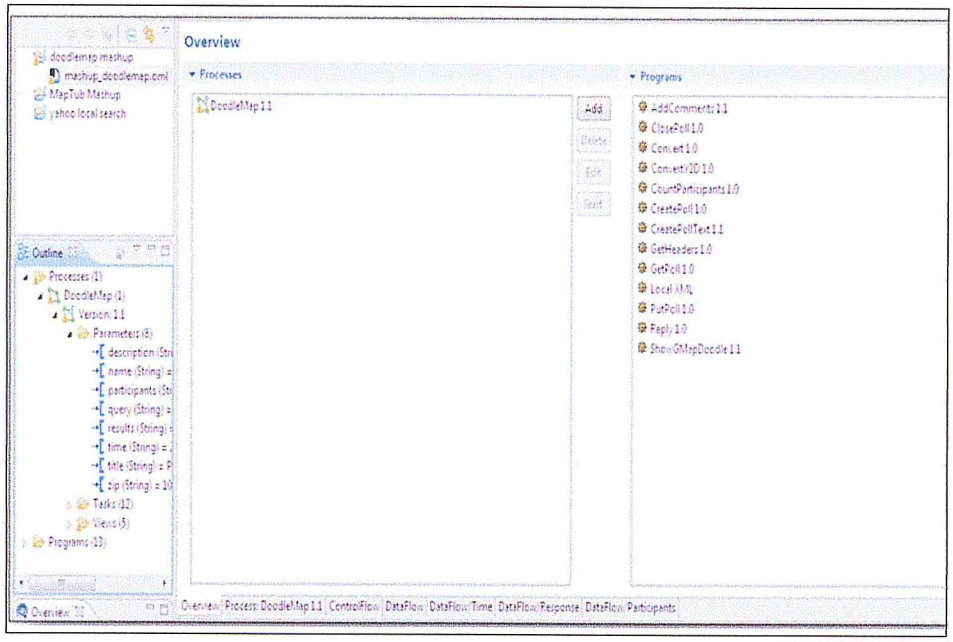


FIGURE 4.3 – mashup doodle map

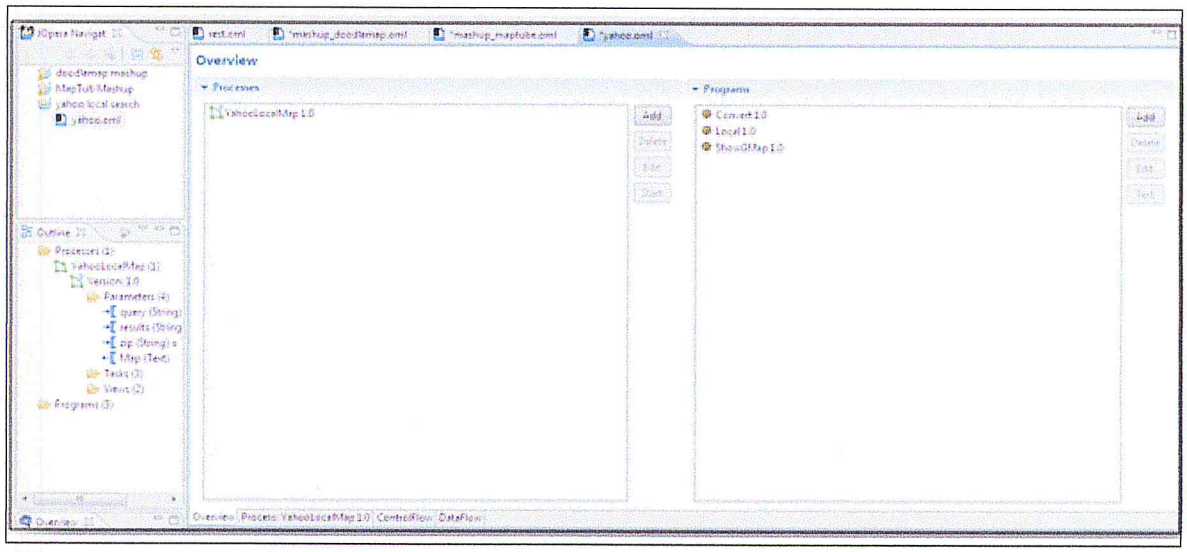


FIGURE 4.4 – mashup yahoo local search map

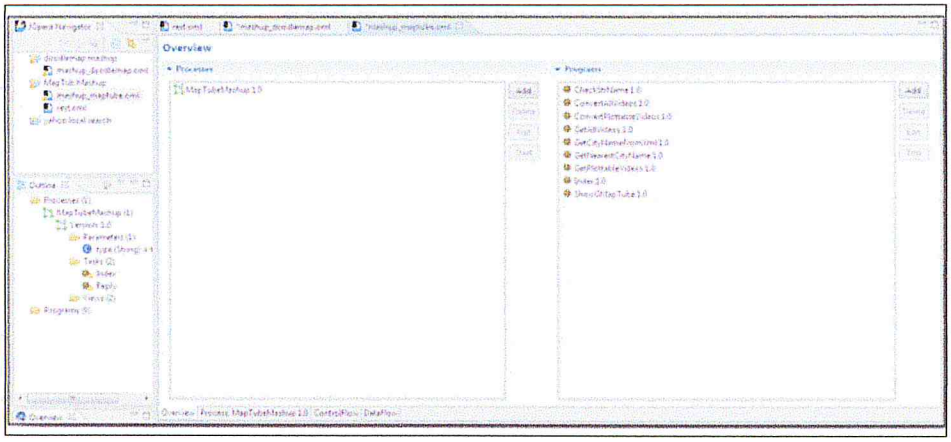


FIGURE 4.5 – mashup maptube

4.4 La réalisation du système générale

4.4.1 Organisation du projet de l'application

Notre application est développée sur Netbeans, ce dernier impose une structure bien définie pour l'organisation de l'application comme le montre la figure suivante.

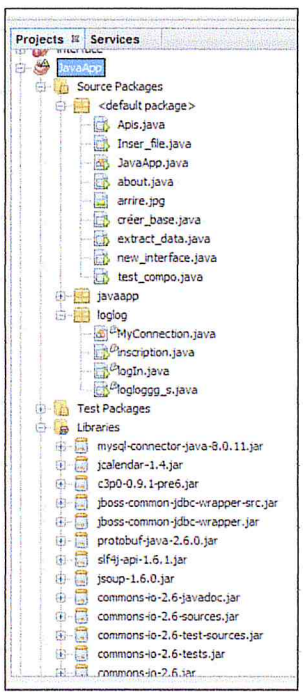


FIGURE 4.6 – Organisation de l'application de l'application

Extractdata C'est la classe charger d'extraire les compositions à partir d'un site web qui contient contient des informations massives et plusieurs paginations.

Inserfile Cette classe permet l'ajout du contenu d'un fichier externe à un fichier de l'application

API Cette classe permet de vérifier l'existence d'un api dans la base de test avant de faire la composition.

Testcompo C'est la classe qui permet le chercheur d'évaluer la validité de sa proposition de composition de services web.

JSoup
Java HTML Parser jsoup est une bibliothèque Java pour travailler avec HTML dans le monde réel. Il fournit une API très pratique pour extraire et manipuler des données, en utilisant les meilleures méthodes DOM, CSS et jquery..[w5]

Apache Commons IO
La bibliothèque Apache Commons IO contient des classes d'utilitaires, des implémentations de flux, des filtres de fichiers, des comparateurs de fichiers, des classes de transformation endian et bien plus encore.[w4]

4.4.2 Test et résultats

Après avoir montré les mashups à évaluer et décrire les classe de notre application, Maintenant on fait des tests sur l'application, et nous allons montrer via les prises d'écrans les étapes à suivre.

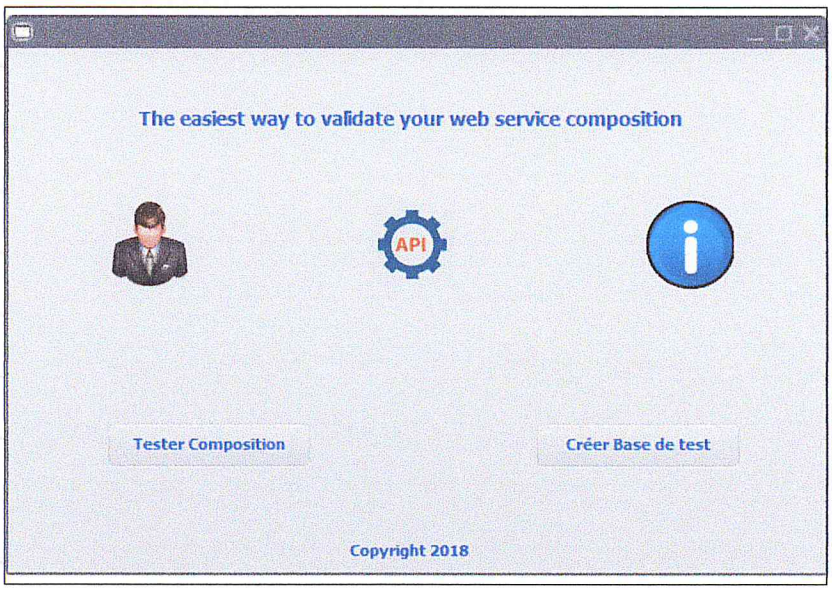


FIGURE 4.7 – Interface global de l'application

4.4.2.1 La création de bases de tests

Après l'authentification, on clique sur le bouton **créer base de test** de l'interface montrée dans la (figure 4.8).

Extraire données

Nous allons faire l'extraction des mashups en utilisant **le crawling.** En cliquant sur le bouton **extraire données** de l'interfaces de la figure suivante

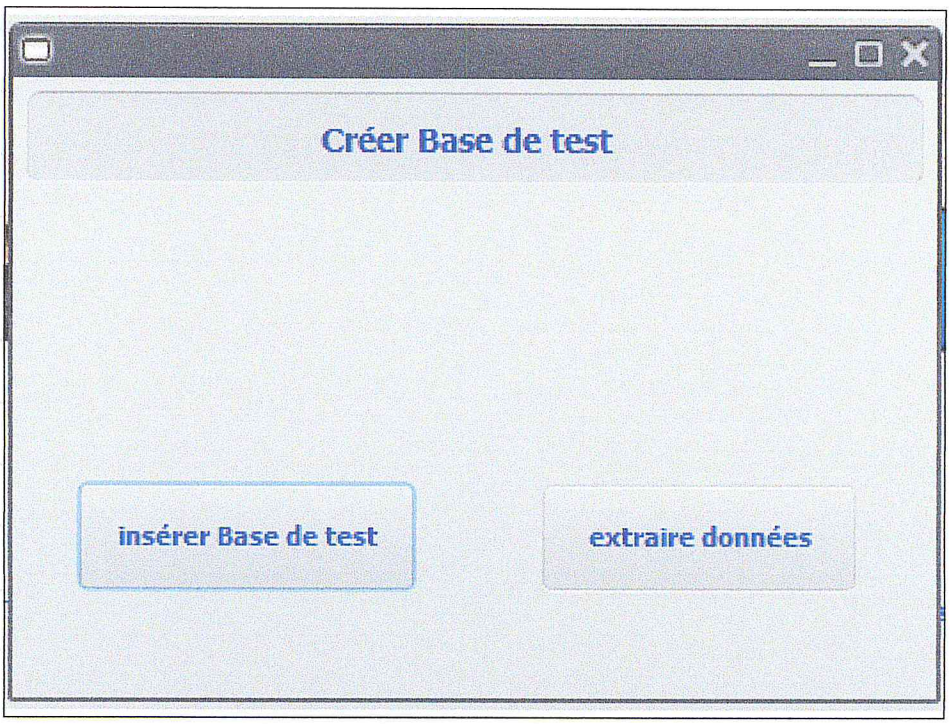


FIGURE 4.8 – Interface "créer base de test"

Nous précisons l'url et les sélecteurs puis nous lançons l'éxecution comme montré dans la figure suivante.

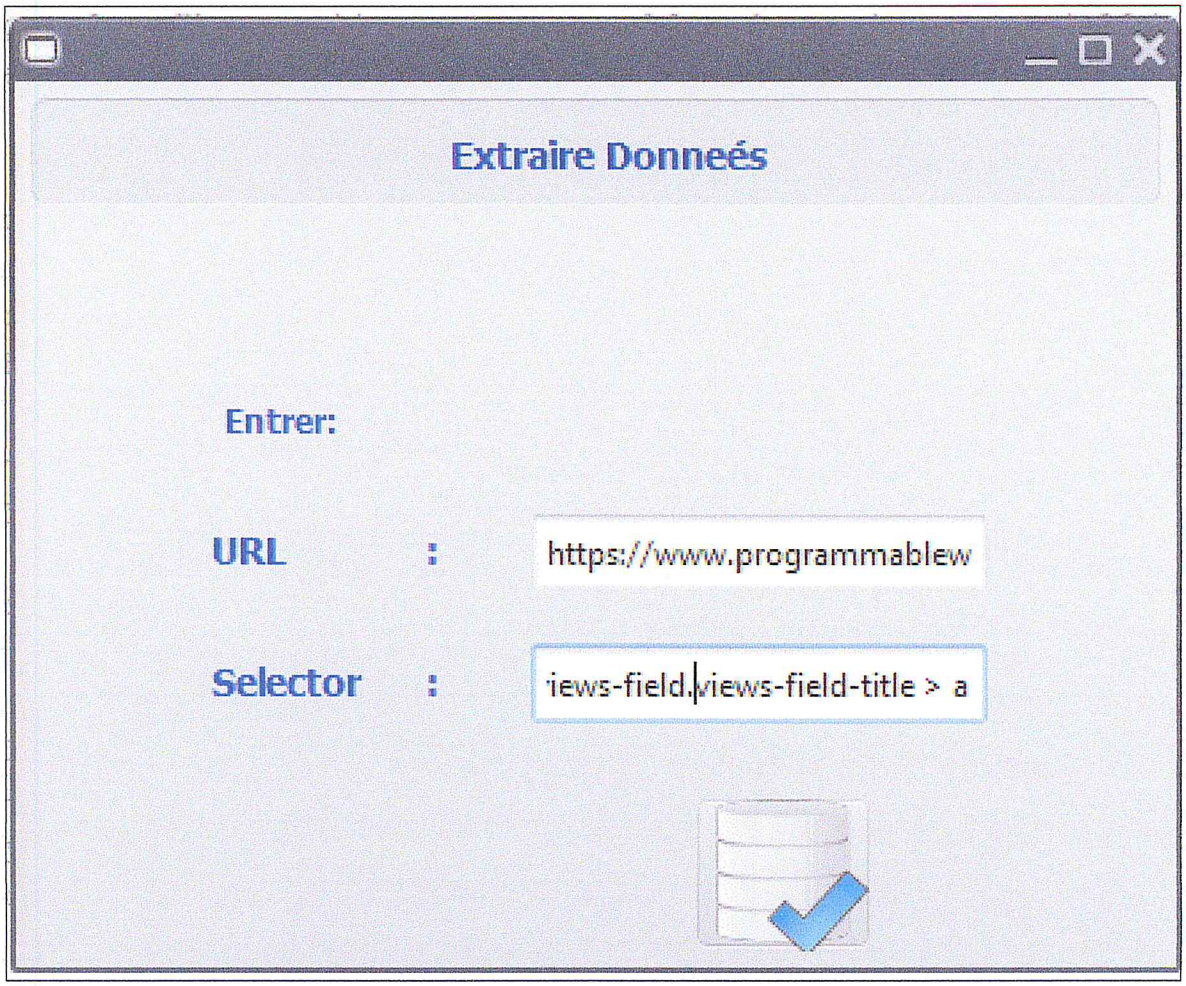


FIGURE 4.9 – Interface "extraire données"

Le résultat de cette partie est un fichier (base de test) contenant des compositions des services web que nous allons l'exploiter dans le test.

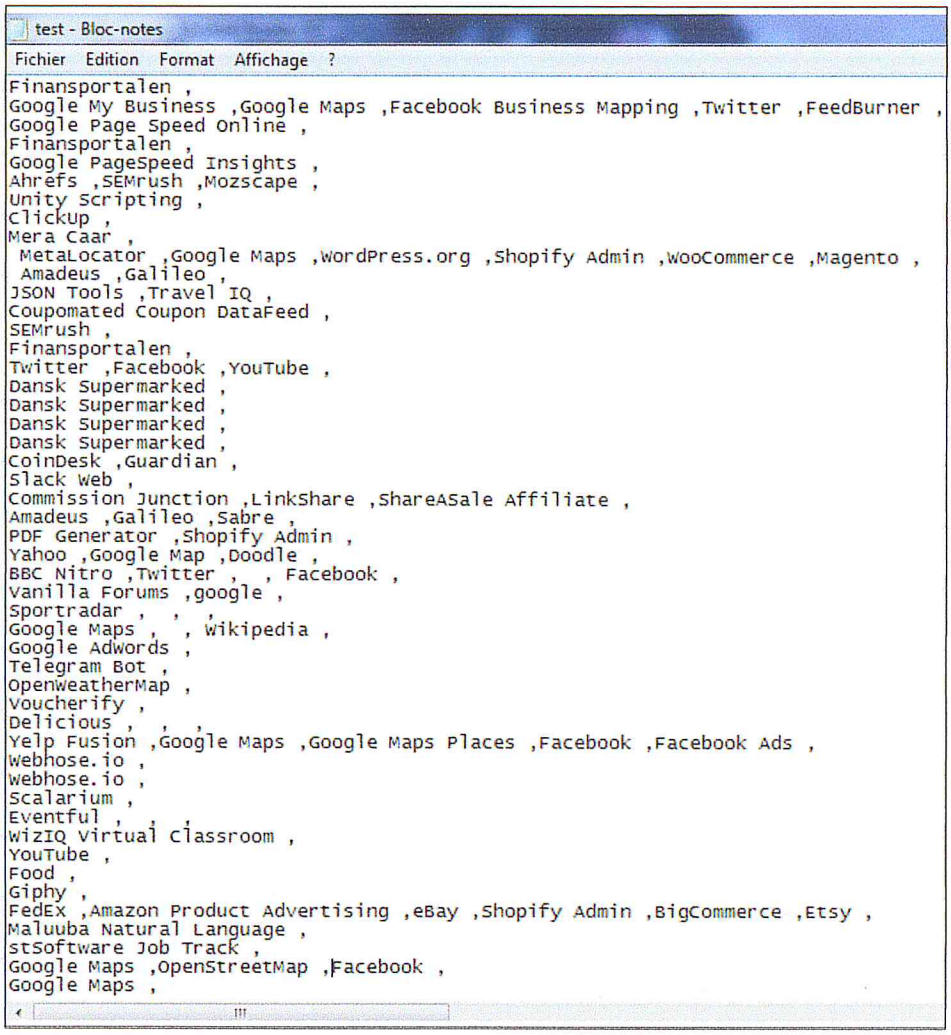


FIGURE 4.10 – Resultats du crawling

4.4.2.2 Tester composition

Avant de faire un essai de composition de services web, le chercher doit d'abord tester l'existence des services à composer dans la base de test, pour cela un petit clique sur le bouton **APIs** de l'interface montré dans la (figure 4.7), puis le nom de l'api dans le champ **Entrer api** de l'interface montré dans les figures suivantes.

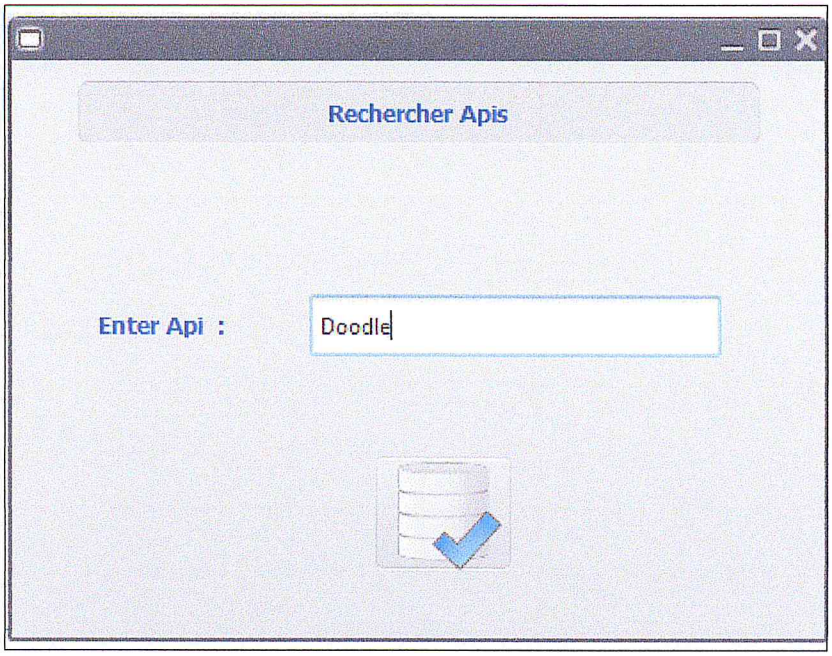


FIGURE 4.11 – la recherche de l'API Doodle

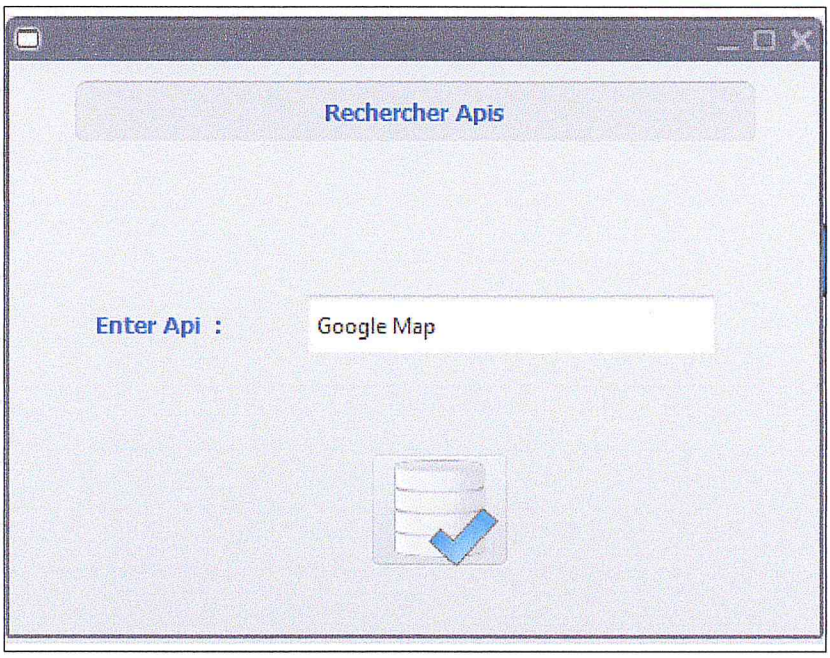


FIGURE 4.12 – la recherche de l'API goolemap

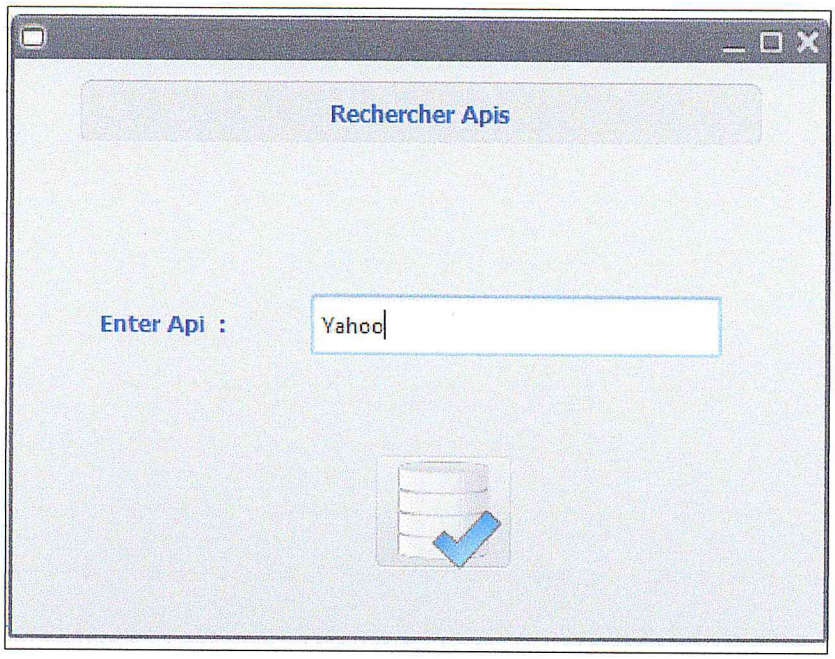


FIGURE 4.13 – la rechrehe de l'API yahoo

Nous lançons l'exécutions, et le résultat est dans la (Figure 4.14).

FIGURE 4.14 – Existence de l'api recherché

Après cette étape nous allons tester la validité des services recherchés dans les (figures 4.11, 4.12 et 4.13) et composée via l'approche proposée dans la partie conception (sous eclipse et via le JOpera).

Pour effectuer un test de proposition de composition de services, on clique sur le bouton **Tester composition** de l'interface de la (Figure 4.7), et dans l'interface montrée dans les (Figures 4.15, 4.16 et 4.17) nous écrivons les compositions proposées

FIGURE 4.15 – Proposition de composition des APIs " yahoo, googlemap et doodle"

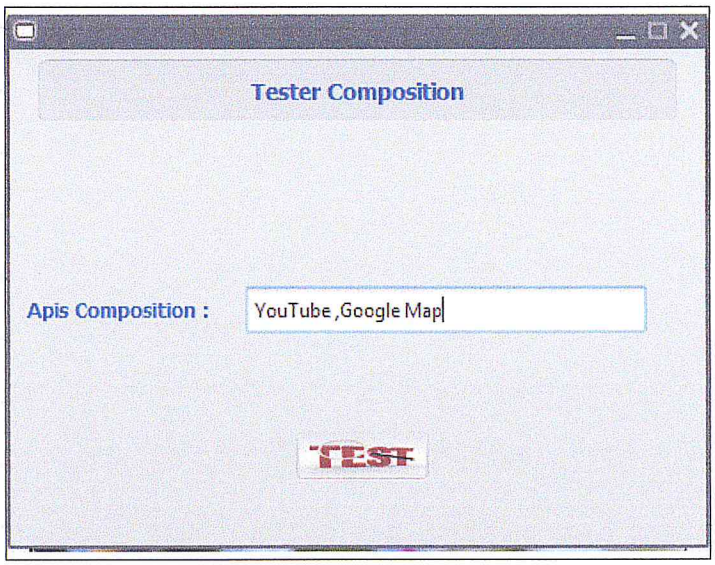


FIGURE 4.16 – Proposition de composition des APIs " youtube, googlemap "

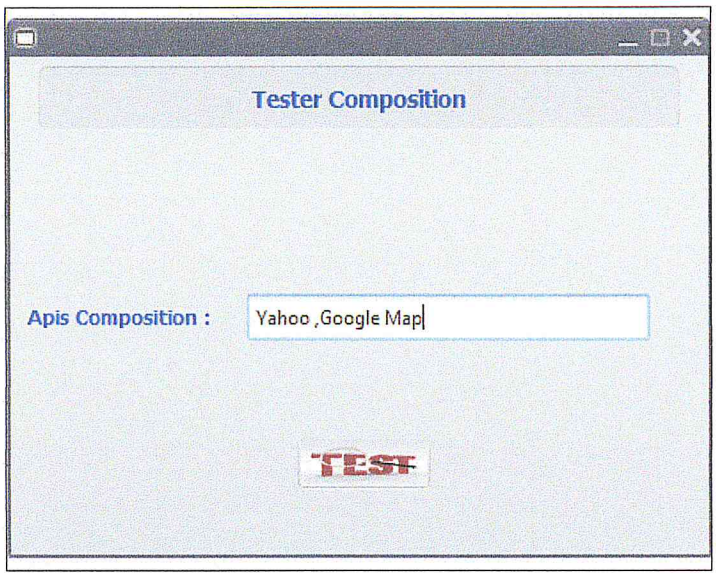


FIGURE 4.17 – Proposition de composition des APIs " yahoo, googlemap "

Apres cliqué sur le bouton **Test** de chaque interface, nous avons obtenu les résultats monteés dans les (Figure 4.18, 4.19 et 4.20).

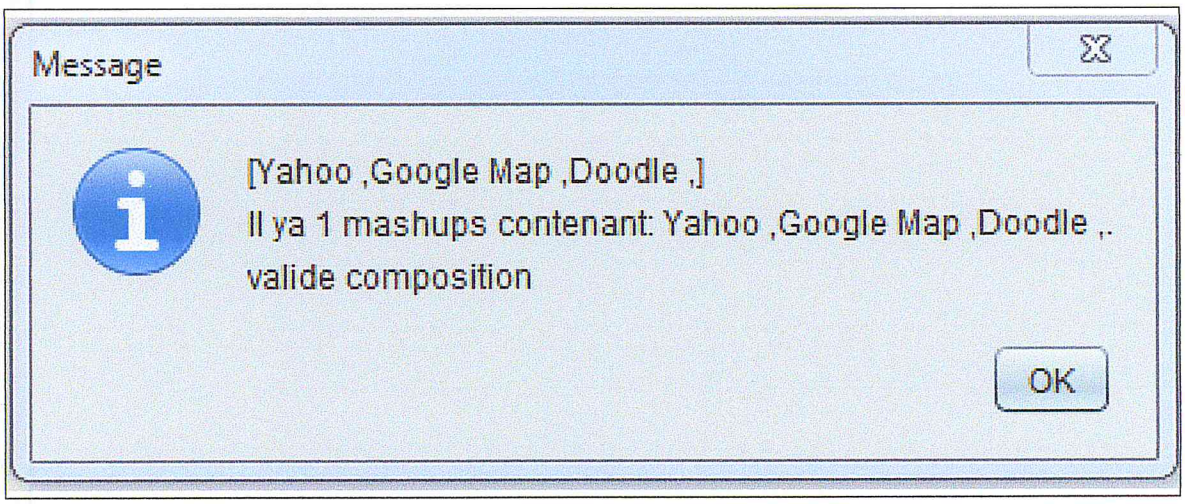


FIGURE 4.18 – message de validité de la composition "yahoo,doodle, google map"

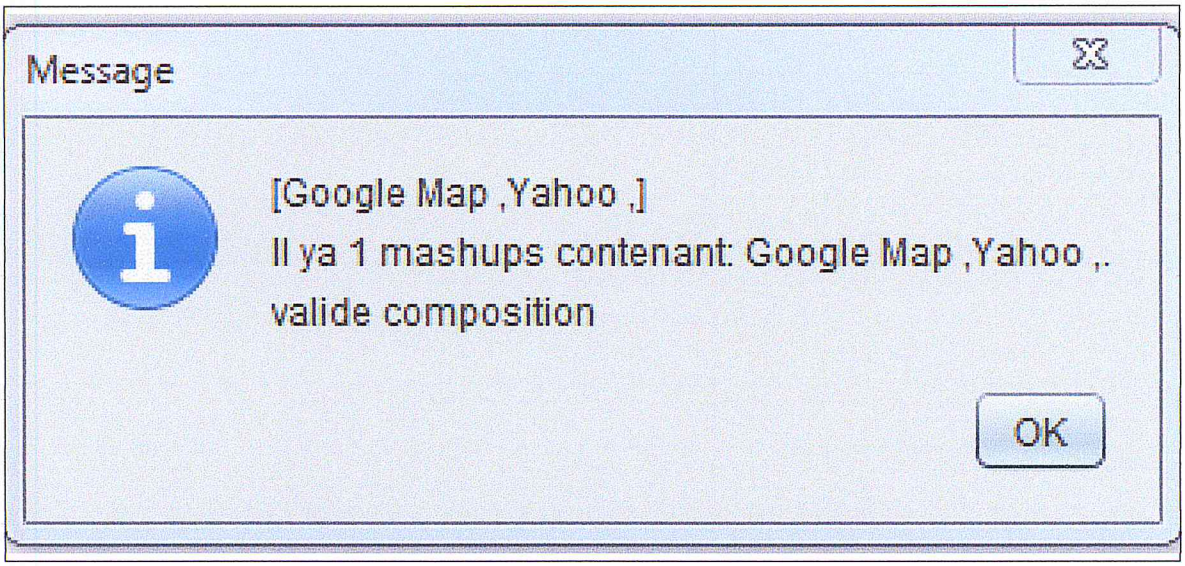


FIGURE 4.19 – message de validité de la composition "youtube, google map"

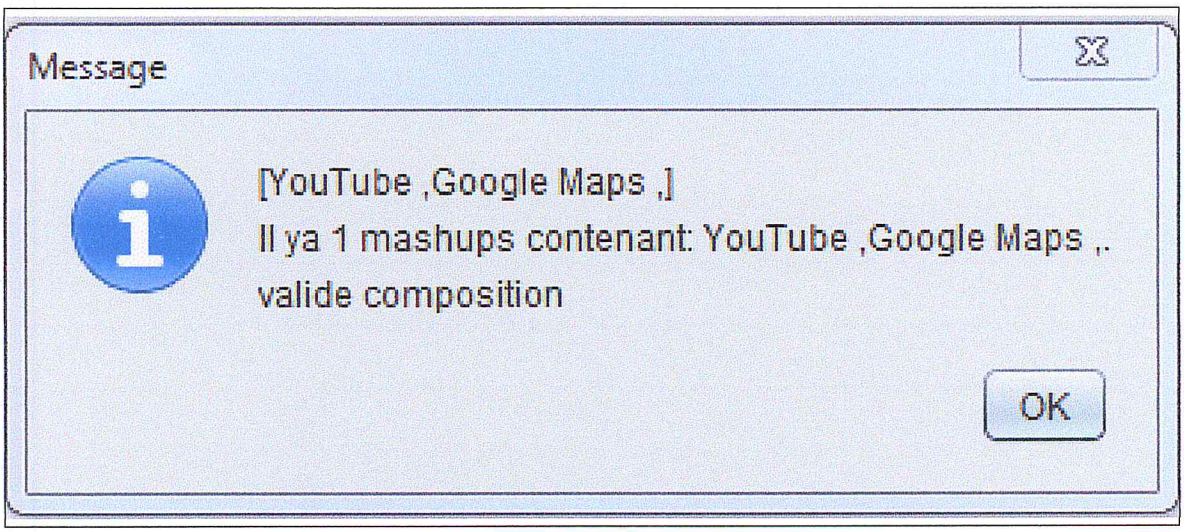


FIGURE 4.20 – message de validité de la composition "yahoo, google map"

À partir des résultats des tests effectués on en déduit que les propositions sont valides toutes à cent pour cent, ce qui veut dire que l'approche sur laquelle les compositions ont été faites est valide et on peut réaliser des compositions avec.

4.5 Conclusion

A travers ce chapitre, Nous avons réalisé une application permettant de créer de base de tests pour l'évaluation des propositions de compositions des services web. Au cours de cette dernière étape de notre travail, nous avons introduit les différents outils et technologies utilisés dans la réalisation, puis nous avons décrit les principales classes de notre système. Et pour terminer nous avons effectué des propositions de composition des services web REST pour les tester sur notre système afin de voir les résultats. En conclusion, les résultats donnent de la satisfaction puisqu'ils répondent aux objectifs du travail demandé.

Conclusion et Perspectives

Conclusion générale

La technologie des services web permet à des applications de dialoguer à distance via Internet, et ceci indépendamment des plateformes et des langages sur lesquelles elles reposent. Cette technologie prend de plus en plus d'ampleur dans différents domaines et constitue toujours un axe de recherche actif. La composition de services Web en particulier permet de combiner plusieurs fonctionnalités des services Web afin de répondre aux exigences qu'un seul service ne peut satisfaire. Cependant, les travaux de recherche sur la découverte et la composition des services web souffrent d'un manque de grandes bases de tests pour l'évaluation et la validation des nouvelles solutions proposées par les chercheurs pour améliorer l'utilisation des services web. C'est dans ce contexte que s'inscrit notre travail. Il a pour objectif de développer un système permettant la création de bases de tests pour l'évaluation de composition de services web.

Afin de mener notre travail, nous avons étudié l'architecture des services web qui est basée sur des standards Internet tels que XML et http qui permettent une communication facile et fluide entre les différents acteurs et clients.Par la suite, nous avons abordé la composition des services web. Les approches de composition des services web varient. Elles sont classées principalement en deux catégories : l'approche statique qui est adoptée notamment par le monde industriel et l'approche dynamique, qui est sujet de recherche du monde académique.

Dans ce mémoire,nous proposons une méthode de création de base de tests contenant des compositions des services web, via deux techniques : le scraping et le crawling. Nous avons présenté et développé une application mettant en œuvre la méthode proposée en se basant sur un certain nombre d'outils et standards.

Perspectives

Afin de compléter le travail présenté dans ce rapport, on peut envisager quelques améliorations et travaux futurs :

- Enrichir de plus en plus la base de tests via plusieurs méthodes de création pour garantir une variété des types de services.
- Mettre en place la solution au niveau de la plateforme réseau Emulab pour des expérimentations à grandes échelles.
- La réalisation de plusieurs compositions via différentes approches de compositions et effectuer les tests dans notre application.

Bibiographie

[1] : M. P. Papazoglou and v. d. H. Willem-Jan,"Service oriented architectures : approaches, technologies and research issues," The International Journal on Very Large Data Bases Journal .

[2] : Thomas Erl. Service-oriented Architecture : Concepts, Technology, and Design. Prentice Hall, 2005.

[3] : Frédéric POURRAZ, Diapason : une approche formelle et centrée architecture pour la composition évolutive de services Web, thèse de doctorat, UNIVERSITE DE SAVOIE (France), Décembre 2007.

[4] : F. Curbera, W. Nagy ET W. Weerawarana. Web services : Why and how ?Conférence d'Object-Oriented Programming, Systems, Languages andApplication (OOPSLA), Workshop on Object-Oriented Web Services, 2001.

[5] : Jérôme Daniel. Extrait du livre "Services Web Concepts, techniques et outils" Edition vuibert Informatique 2003.

[6] : CHOUIREF Zahira, Un système de programmation fonctionnelle pour la composition de services Web, mémoire de magister, Université M'hamed BOUGARA de BOUMERDES, 2008.

[7] : AMRANE Bakhta, Méthode de rechreche de services web basés sur l'analyse formelle de concept, University d'Oran Amed Benballa, Oran, 2015.

[8] : Manel Amel Djenouhat .Un cadre sémantique formel pour la description, sélection et composition des services web. conservatoire national des arts et métiers –CNAM,2017. Francais. <NNT :2017CNMA1137>. <tel-01743824>.

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

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

[11] : Chris Peltz.Xeb.Services orchestration and choreograph.Computer,36(10) :46-52,2003.

[12] : T. Osman, D. Thakker, and D. Al-Dabass. Bridging the Gap between Workflow and Semantic-based Web services Composition. In Proc. of the Web Service Composition Workshop WSCOMPS05, 2005.

[13] : Julien Guitton "planification multi-agent pour la composition dynamique de service web" Master 2 Université Joseph Fourien 2006.

[14] : J. Cardoso and A. Sheth. Semantic e-workflow composition. Technical Report 02-004,LSDIS Lab., Computer Science Department, Univesity of Georgia, Athens, USA, 2002.

[15] : IHADJADENE Madjid. « Usages des moteurs de recherche. In Journée d'Etude ADBS, Optimiser l'accès à l'information, une opportunité pour les langages documentaires ? », Paris, 2007.

[16] : Edwards, McCurley, K. S., and Tomlin, J. A. "An adaptive model for optimizing performance of an incremental web crawler". In Proceedings of the Tenth Conference on WorW Wide Web (Hong Kong : Elsevier Science) : 106-m. 45p-6op. 2001.

[17] : MARC NAJORK. Web Crawler Architecture [en ligne]. Mountain View, CA, USA : Microsoft Research, [2009].

[18] : DATA PUBLICA. Crawling et Scraping : exploiter les données du Web pour développer votre business [en ligne]. 2013.

[19] : RICARDO BAEZA-YATES, CARLOS CASTILLO, MAURICIO MARIN, et al. Crawling a Country : Better Strategies than Breadth-First for Web Page Ordering [en ligne]. 2005.

[20] : IMPREVA. Detecting and Blocking Site Scraping Attacks [en ligne]. 2011.

[21] : SOUMEN CHAKRABARTI. Mining the web [en ligne]. Bombay, India : Morgan Kaufmann Publishers, 2003.

[22] : CARLOS CASTILLO. Effective Web Crawling [en ligne]. Santiago, RM, Chile : University of Chile, 2004.

[23] : SOUMEN CHAKRABARTI, MARTIN VAN DEN BERG, BYRON DOM. Focused crawling : a new approach to topic-specific Web resource discovery [en ligne]. Bombay, India : Indian Institute of Technology, 1999. [24] : Will, T : « Introduction to the Singular Value Decomposition. » ; Davidson College ; 1999.

[25] : Lawrence, Steve ; c. Lee Giles "Accessibility of information on the web". Nature 400(6740) : 107. Bibcode i999Natur.400..i07L.1038/21987. pmid 10428673. 1999-07-08.

[26] : Lawrence, Steve ; c. Lee Giles "Accessibility of information on the web". Nature 400(6740) : 107. Bibcode i999Natur.400..i07L.1038/21987. pmid 10428673. 1999-07-08.

[27] : K.B.Petersen, M .S.Pedersen, « The matrix cookbook» , version 2008.

[38] : S.Brin, L.Page : « The anatomy of a large-scale hypertextual Web search engine », WWW7, Brisbane, Australia, 1998.

[29] : O.A.McBryan, « Tools For Taming The Web. In Proceedings Of The First International Conference On World Wide Web », Chicago USA 1993.

[30] :B.Pinkerton, « Finding What People Want : Experiences With The Crawler. In Proceedings Of The First International Conference On World Wide Web », Chicago USA 1993.

[31] : Boldi, Paolo; Massimo Santini, Sebastiano Vigna (2004). "Do Your Worst to Make the Best : Paradoxical Effects in PageRank Incremental Computations and Models for the Web-Graph, pp. 168-180. Retrieved 2009-03-23.

[32] : ANDREW PETERSON. BeautifulSoup : Web Scraping with Python [en ligne]. 2013.

[33] : CHARLES SEVERANCE. Scraping Web Pages [en ligne]. University of Michigan : School of Information.

[34] : MICHAEL SCHRENK. Webbots, Spiders, and Screen Scrapers. No Starch Press, 2007.

[35] : CHRIS HANRETTY. Scraping the web for arts and humanities [en ligne]. Norwich, Royaume-Uni : University of East Anglia, 2013.

[36] : Cesare Pautasso, Composing RESTful services with JOpera, University ofLugano, Switzerland, 2009.

[37] : Pautasso, C., Alonso, G. : The JOpera visual composition language. Journal of Visual Languages and Computing (JVLC) 16(1-2) (2005) 119–152.

[38] : D.Fallside and P.Walmsley.Extensible markuplangage schéma(xml schema)1.0. World Wide Web Consorium,http ://www.w3.org/TR/wmlschema-0,2004.

[39] : M.Gudgin,M.Hadley,N.Mendelsohn,J.J.Moreau ,H.F.Nielsen.Simple object access protocol(soap)1.2. World Wide Web Consorium,http ://www.w3.org/TR/soap,2003.

[40] : Frédéric POURRAZ Thése : Diapason : une approche formélle et cnetrée architecture pour la composition évolutive des services web - le 10 décembre 2007.

[41] : E.Christensen,F.Curbera,G.Meredith,and S.Weerawarana.Web service descreption language(wsdl)1.1.World Wide Web Consorium,http ://www.w3.org/TR/wsdl, 2001.

[42] : R.Chinnici,J.J.Moreau,A.Ryman, and S.Weerawarana. Web service descreptionlanguage(wsdl)version2.0.World Wide Web Consorium,http ://www.w3.org/TR/wsdl20, 2007.

[43] : L.Clement,A.Hately,C.Von Riegen, and T.Rogers.Uddi version 3.0.2.http ://uddi.org/pubs/udd v3.htm,2004.

[44] : FIELDING, R. T. Architectural styles and the design of network-based software architectures, University of California2000).

[45] : FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P. and BERNERS-LEE, T. Hypertext transfer protocol–HTTP/1.1, RFC 2616, June1999).

[46] : Barros A., Dumas, M., Oaks, P. Standards for Web Service Choreography andOrchestration : Status and Perspectives. In : Procs of the 3rd International Conference the Business Process Management (BPM 2005), 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management, Nancy, France, Sept. 2005, pp.1-15.

[47] : Benatallah B., Dijkman R., Dumas M., Maamar Z. Service Composition :Concepts, Techniques, Tools and Trends. In : Stojanovic Z., Dahanayake A., Eds. Service-Oriented Software Engineering : Challenges and Practices. Idea Group Inc (IGI), 2005, pp.48-66.

Site web

[w1] : `https://en.wikipedia.org/wiki/Web_Application_Description_Language.visitéle15.01.2018`

[w2] : `https://fr.wikipedia.org/wiki/NetBeans.visitéle22.05.2018`

[w3] : `http://forum.webrankinfo.com.visitéle2.06.2018`

[w4] : `https://mvnrepository.com/artifact/commons-io/commons-io.visitéle16.09.2018`

[w5] : `http://jsoup.org.visitéle28.07.2018`

[w6] : `http://www.jopera.org/.visitéle15.08.2018`

[w7] : `http://www.eclipse.org/.visitéle15.08.2018`