RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ SAAD DAHLAB DE BLIDA

Faculté des Sciences
Département d'Informatique

Mémoire de Master
Option : Ingénierie du logiciel

présenté par
HADJER YKHLEF

Conception et implémentation d'un système de communication sécurisé pour les applications e-Gouvernement

Rapporteur : Dr. Djamel BENNAOUR

Mr BALA Président du jury.
Mme Boumehdi Examinatrice.
Melle Azzouz Examinatrice.
Dr Bennoumar Promoteur

Soutenue publiquement en Juillet 2011

À ma famille.

Résumé

ملخص

يندرج هذا العمل في اطار انشاء الحكومة الالكترونية و توصيل مختلف خدمات الحالة المدنية بشبكة الانترنيت . الهدف من هذا المشروع هو انشاء نظام اتصالات آمن لتطبيقات و برمجيات الحكومة الالكترونية . من اجل تحقيق هذا الهدف اقترحنا تكوين بنية هيكلية مشيدة علي اساس الخدمات المقترحة من طرف الهيئات التالية البلديات ، الدوائر ، المحاكم و الوزارة ، بحيث تمركز كل هيئة في مستوي معين من البنية الهيكلية و توفر خدمات متعددة .

كلمات المفاتيح البني الهيكلية ، الحكومة الالكترونية ، الانظمة الموزعة ، امن الحاسب ، انظمة الاتصالات ، خدمة الويب.

Résumé

Ce travail est effectué dans le cadre d'informatisation du secteur état civil « E-APC ». Le but de ce projet est de réaliser un système de communication sécurisé pour les applications e-Gouvernement. La solution proposée consiste à construire une architecture orientée service bâtie sur les différentes entités : les APCs, les Daïras, les Tribunaux et le Ministère. Chacune occupe un niveau dans la hiérarchie de l'architecture globale du système et offre des fonctionnalités prédéfinies.

Mots clés : AOS, SSL, e-Gouvernement, Système répartie, Sécurité informatique, Système de communication, Service Web.

Abstract

The context of this work is E-APC project. This later consists of the utilization of the Internet and the world-wide-web for delivering services to the citizens. The aim of this project is to realize a secure communication system for e-Government applications. We proposed to build a service oriented architecture based on the different participants : APCs, Dairas, Court and Department. Each participant occupies a level in the system architecture hierarchy and provides well-defined services.

Keywords : SOA, SSL, e-Government, Distributed systems, Computer security, Communication system, Web service.

Remerciements

Quelques lignes ne pourront jamais exprimer la reconnaissance que j'éprouve envers tous ceux qui, de pris ou de loin, ont contribué, par leurs conseils, leurs encouragements ou leurs amitiés à l'aboutissement de ce travail.

Mes vifs remerciements accompagnés de toute ma gratitude vont tout d'abord à mon promoteur Dr. Djamel BENNOUAR, Maître de conférences à l'Université de Blida pour m'avoir proposé ce sujet, pour les conseils qu'il n'a cessé de me prodiguer, son aide et surtout pour la confiance qu'il m'a accordée pour la réalisation de ce projet.

Je remercie également les membres du jury pour avoir accepté d'évaluer et de juger ce modeste travail.

Enfin, je tiens à remercier et à exprimer ma profonde gratitude à mon frère Farid YKHLEF, Maître de conférences à l'Université de Blida pour son aide, ses encouragements et ses conseils.

Acronymes

Pour des raisons de lisibilité, la signification d'une abréviation ou d'un acronyme n'est souvent rappelée qu'à sa première apparition dans le texte d'un chapitre.

i.e.	: that is to say (c'est-à-dire)
APC	: Assemblée Populaire Communale
SOA	: Service Oriented Architecture
UML	: Unified Modeling Language
SoaML	: Service oriented architecture Modeling Language
OMG	: Object Management Group
OCL	: Object Constraint Language
WDSL	: Web Services Description Language
UDDI	: Universal Description, Discovery and Integration
SOAP	: Simple Object Access Protocol
XML	: Extensible Markup Language
XSD	: XML Schema
DAO	: Data Access Object
DTO	: Data Transfer Object
TLS	: Transport Layer Security
SSL	: Secure Sockets Layer
BPMN	: Business Process Modeling Notation
SHAPE	: Service-oriented Heterogeneous Architecture and Platforms Engineering

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Aujourd'hui, l'Internet et les technologies de l'information et de la communication deviennent de plus en plus indispensables dans la vie des organisations, quelle que soit sa taille et quel que soit son secteur d'activité, en effet l'Internet tisse de nouveaux points de contacts avec le monde extérieur.

A cet effet, et dans le but d'améliorer les services administratives, le projet E-APC a été mis en place par l'équipe **RSI** (**R**éseaux et **S**ystèmes d'**I**nformation) de la division **ASM** (**A**rchitecture des **S**ystèmes et **M**ultimédias) du **CDTA** (**C**entre de **D**éveloppement des **T**echnologies **A**vancées). Leur objectif était de réaliser un système qui permettrait de mettre sur Internet les divers services et activités que nous trouvons dans une **APC** (**A**ssemblée **P**opulaire **C**ommunale) et qui sont fortement sollicités par les citoyens d'une commune tels que : le service d'état civil, le service de vote et les divers aspects liés aux délibérations de l'APC.

Le projet E-APC réalisé par l'équipe RSI a fixé plusieurs objectifs, parmi lesquels nous citons :

- Permettre le retrait en ligne des divers documents de l'état civil (acte de naissance, acte de mariage, etc.) ;
- Permettre l'interaction sur Internet avec le service de vote (inscription, radiation, etc.) ;
- Mise en place d'un mécanisme de vérification/validation de l'information ;
- L'authentification des documents produits par le système ;
- Le suivi des responsabilités : Toute action importante pouvant avoir un impact sur les transformations des informations, notamment données personnelles du citoyen, devra être journalisée et exploitée.

Notre travail consiste à étudier et réaliser un système de communication entre les instances de l'E-APC selon l'approche d'architecture orientée service. Pour cela nous devons définir l'architecture du système global, c'est-à-dire l'ensemble des instances des E-APC déployées à travers le territoire national et les connexions entre elles. Ce système devra assurer entre autres :

- La cohérence et la mise à jour du système global ;
- La consolidation des données au niveau régional et national ;
- La gestion de l'échange des avis de mentions ;
- Le suivi des responsabilités ;
- Réduire le taux des livraisons non instantanées des actes de l'état civil ;
- La restauration des données lors d'un incident.

Le système de communication devra être conçu pour permettre d'une part l'intégration aisée d'E-APC 2.0 et d'autre part fixer les éléments que les versions ultérieures d'E-APC doivent respecter.

Les autres objectifs sont liés au comportement des instances de l'E-APC en cas d'indisponibilité ou de pannes réseau. C'est ainsi que le système de communication devra permettre à l'E-APC de maintenir et gérer localement les informations qui appartiennent naturellement à d'autres instances. Ces informations seront par la suite transférées aux E-APC concernées dès leur reprise suite à une panne ou dès leur mise en service pour la première fois. Dans ce même contexte, nous devons réaliser des assistants pour diriger le processus de restauration des données.

La conception selon l'approche d'architecture orientée service nécessite par la suite un passage au niveau implémentation. Un des objectifs qui nous a été assigné concerne la détermination de la technologie adéquate pour le support des services. ***Seraient-ils des services basés sur la technologie RMI ? CORBA ? Service Web ?*** Une étude comparative est ainsi nécessaire pour le choix de la technologie adéquate.

Ce mémoire est divisé en deux parties :

Une **première partie** composée de trois chapitres, se concentre sur la présentation du contexte dans lequel notre travail a été exécuté. Dans le premier chapitre, nous présenterons une étude comparative entre les différentes technologies d'implémentation des systèmes distribués afin de sélectionner la technique la plus adaptée à notre projet. Le chapitre 2 sera consacré à l'étude des différentes solutions de sécurité existantes pour la technologie choisie dans le premier chapitre. Le chapitre 3 montre une étude détaillée de l'application E-APC 2.0, en couvrant toutes les étapes de sa réalisation : en commençant par l'analyse des besoins jusqu'à l'implémentation, ainsi nous présenterons une analyse approfondie de cette application.

La **deuxième partie** est consacrée à la conception et l'implémentation du système de communication. Au chapitre 4, nous présenterons dans un premier temps la solution avec le détail nécessaire. La suite de ce chapitre traitera de la conception du système en utilisant le langage **SoaML**, ainsi une description brève de la technique d'adaptation est fournie afin d'intégrer l'application E-APC 2.0 à notre système.

L'architecture globale du système sera décrite au début du chapitre 5. La suite de ce chapitre sera consacrée à la présentation détaillée des techniques utilisées pour la sécurisation des échanges et l'adaptation de l'application E-APC 2.0 à notre système. Nous terminerons ce chapitre par la présentation de quelques exemples de mise en œuvre du système développé.

Chapitre 1

Etat de l'art sur les systèmes de communication

1.1 Introduction

Avec l'essor de l'Internet, il devient de plus en plus intéressant pour la majorité des entreprises et les organisations de combiner leurs système existants. Cela permet d'accélérer et de réduire le budget de développement des nouveaux systèmes, en réutilisant ce qui existe. L'ensemble de ces applications qui coopèrent via un système de communication forme ce qu'on appelle un système réparti, on parle aussi de l'intégration des systèmes d'informations. Il existe plusieurs standards permettant la réalisation de cette tâche, notamment les architectures basées sur : **CORBA**, **RMI** et les **services Web**.

Dans ce chapitre, on va limiter notre étude sur trois grandes plateformes d'interconnexions des éléments d'un système distribué : les RMI de Java, le bus CORBA de l'OMG et les services Web. A la fin de cette étude, nous présenterons une comparaison de ces trois technologies et nous choisirons l'approche la plus intéressante pour notre application.

1.2 Les architectures orientées service

1.2.1 Le concept de service

L'un des objectifs du génie logiciel est de rationaliser le développement d'applications de qualité. En particulier, la question de la réutilisation des briques logicielles déjà développées constitue un enjeu majeur. Il y a déjà une vingtaine d'années, l'approche objet donnait un premier élément de réponse à cette problématique en apportant un ensemble de principes de conception, comme par exemple : l'encapsulation, le polymorphisme, l'héritage, etc.
Les composants sont ensuite apparus pour permettre des regroupements cohérents et réutilisables d'objets :
"Un composant est une unité de composition, avec des interfaces spécifiées à l'aide de contrats et dépendant de contextes. Un composant logiciel peut être déployé de manière indépendante et est sujet à être composé par des tiers" [2].
Techniquement, les composants se déploient et s'exécutent dans des contextes qui leur sont spécifiques. Ces contextes impactent la manière dont sont déployés les composants.

Le concept de service est actuellement le sujet de définitions très variées. Nous en avons retenu seulement deux :

- "Un service représente certaines fonctionnalités (application fonctionnelle, transaction commerciale, un service du système de base, etc.) exposés sous la forme d'un composant au sein d'un processus métier" [3] ;
- "Un service, dans le cadre des architectures orientées services, expose une partie de la fonctionnalité fournie par l'architecture et respecte trois propriétés : (1) le contrat du service est exposé dans une interface indépendante de toute plate-forme, (2) le service peut être dynamiquement localisé et invoqué, (3) le service est autonome et sait maintenir son propre état courant" [4].

De ces deux définitions, nous ressortons une idée principale, à savoir qu'un service permet d'exposer une ou plusieurs fonctionnalités, offertes par un fournisseur, à des clients potentiels.
De façon similaire aux approches par objet ou par composant, l'approche par service cherche à fournir un niveau d'abstraction encore supérieur, en encapsulant des fonctionnalités et en permettant la réutilisation de services déjà existants.
Voyons maintenant comment faire interagir ces services au sein d'un environnement. On parle alors d'architecture orientée service.

1.2.2 Définition de l'architecture orientée service

L'architecture orientée service, connue sous l'acronyme SOA (Services Oriented Architecture), est une approche architecturale permettant la création des systèmes basés sur une collection de services développés dans différents langages de programmation, hébergés sur différentes plates-formes avec divers modèles de sécurité et processus métier [1].
L'SOA est axée autour de trois concepts fondamentaux :

- **Le fournisseur de services** : il permet l'accès à son service à travers une interface ;
- **Le client de services** : il désigne une personne, un serveur ou une autre application qui accède au service et l'invoque à travers son interface ;
- **L'annuaire de publication** : il joue le rôle d'intermédiaire entre le fournisseur et le client. Les fournisseurs y enregistrent leurs services, et les clients y cherchent le service satisfaisant leurs besoins.

L'SOA encapsule plusieurs avantages bénéfiques pour le domaine de la technologie d'information et de communication. Elle offre la simplicité à travers les concepts de décomposition, de découplage et de réutilisation. En plus, elle participe à la réduction du coût de développement des grands projets et rend le développement plus efficace [6].

L'architecture orientée service est apparue pour pallier les limites des architectures distribuées. Cette architecture se place dans la continuité logique des multiples tentatives de distribution de traitements, de répartition de données, d'intégration d'applications, d'homogénéisation du système d'informations, etc. L'adoption de la SOA a été grandement facilitée par l'émergence opportune de la technologie des services Web et leurs standards bien définis. La technologie des services Web représente la technologie la plus utilisée pour migrer vers ce type d'architectures [5].

1.3 Service Web

1.3.1 Définitions

Le concept de service Web va maintenant être défini selon plusieurs parties.

- Selon Jérôme Daniel : "Un service Web est un moyen d'accéder à un service par l'intermédiaire du Web" [9].
- Selon W3C[1] : "Un service Web est un logiciel identifié par une URI (Uniform Resource Identifier) [RFC2396], dont l'interface est publique et les liaisons sont définies et décrites en utilisant XML. L'environnement du service offre le moyen à d'autres logiciels de découvrir celui-ci. Ce service interagit avec les autres services Web en respectant cette définition, donc en utilisant des messages basés sur XML acheminés par des protocoles Internet" [8].
- Selon Gilbert Babin : "Les Web Services sont des applications qui relient des programmes, des objets, etc. à l'aide de XML et de protocoles Internet standards. Les Web Services sont des compléments aux programmes et applications existantes, développées à l'aide de langages tel que Visual Basic, C, C++, C#, Java ou autre, et servent de pont pour que ces programmes communiquent entre eux" [10].

De ces trois définitions on peut conclure qu'un Web service est une brique de base pour créer une architecture de communication, dont les entités sont les applications développées à l'aide des langages de programmation différents.

1.3.2 Les composants de l'architecture des services Web

A partir de la définition du consortium W3C, on peut identifier les composants de l'architecture des services Web (ou SOA) :

Echange : comment échanger les messages entre les services Web ?

Un protocole de communication qui utilise le formalisme XML pour, à la fois, définir les messages envoyés entre les applications et représenter les données échangées ; ce protocole appelé **SOAP** (**S**imple **O**bject **A**ccess **P**rotocol), spécifié au sein du W3C, est largement accepté dans le but d'offrir un mécanisme de communication aussi bien pour l'Intranet que pour l'Internet [9].

Description : comment exposer les fonctions des services Web ?

Un langage de description qui permet de définir précisément quels sont les services disponibles ainsi que la façon dont on doit interagir avec ceux-ci ; ce langage de description est appelé **WSDL** (**W**eb **S**ervice **D**escription **L**anguage) est également fondé sur le formalisme XML [9].

Découverte : comment identifier et localiser les services Web ?

Un moyen pour publier les services Web que l'on offre mais aussi un outil permettant de rechercher un service Web à partir de plusieurs critères ; ce rôle est joué par le référen-

1. World Wide Web consortium

tiel **UDDI** (**U**niversal **D**escription, **D**iscovery and **I**ntegration) [9].
La définition du W3C ne présuppose ni l'utilisation du protocole SOAP (Simple Object Access Protocol), ni une description du service Web par le langage de description WSDL (Web Services Description Language). Mais l'architecture de référence d'un service Web suppose que ce service possède un niveau d'abstraction dans lequel il est décrit par le langage de description WSDL et qu'il emploie le protocole de communication SOAP. Autrement dit, ces trois technologies sont les plus utilisées dans le monde des services Web.

1.3.3 Invocation d'un service Web

Les étapes les plus importantes de l'invocation d'un service Web sont les suivantes (voir figure 1.1) [9] :

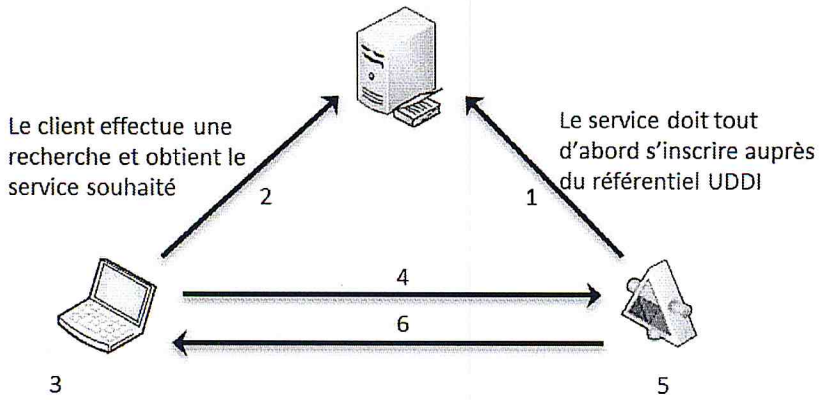


FIGURE 1.1 – Les étapes d'invocation d'un service Web.

1. Tout d'abord, le service doit s'inscrire auprès d'un référentiel UDDI en indiquant plusieurs de ces caractéristiques, y compris sa description WSDL ;
2. L'application cliente va consulter un référentiel UDDI afin de sélectionner le service Web qu'elle souhaite utiliser ;
3. Une fois que le service Web est sélectionné, on peut consulter sa description WSDL pour savoir comment interagir avec celui-ci. Cette description permet de connaitre quel message SOAP est à envoyer et quel message SOAP sera reçu en retour du traitement souhaité ;
4. Il faut à présent contacter le service Web et donc établir la communication avec ce dernier. Il faut noter que cette connexion a lieu lors de l'envoi du premier message SOAP ;
5. Le message reçu du côté service Web implique un traitement. Le service Web pouvant être un frontal d'un environnement réparti, le traitement en cours peut nécessiter à son tour divers interaction au sein même du système d'information impliqué dans le traitement ;
6. Si aucune erreur ne se produit, une réponse est émisse à l'application cliente.

1.3.4 Apports des services Web

L'utilisation des services Web engendre plusieurs avantages dont on peut citer [10] :

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 [8], 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.

1.3.5 Les couches de l'architecture orientée service Web

Dans cette étape, nous allons présenter la partie qui nous intéresse de la vision des différentes couches de l'architecture orientée service Web définie par [4].

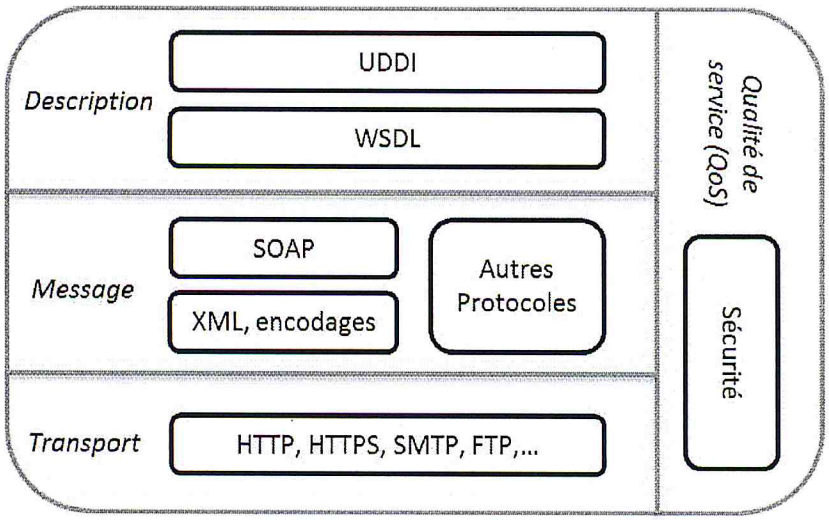


FIGURE 1.2 – Partie de la vision des différentes couches de l'architecture orientée service Web.

La couche transport

Cette couche de base adresse les aspects liés au transport des messages. Il est souvent possible de spécifier le style, le mode et le protocole d'une communication. On distingue au moins deux styles de communication :

- 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.

Quant au protocole de communication, le support le plus souvent utilisé est HTTP mais il peut aussi être SMTP, FTP, JMS (Java Message Service), etc.

La couche messages

La communication par messages constitue un point central dans toutes architectures orientées service Web afin de coller au paradigme SOA, en particulier promouvoir un faible couplage, et ainsi couvrir les recommandations du modèle de référence **OASIS** (**O**rganization for the **A**dvancement of **S**tructured **I**nformation **S**tandards). Les messages qui transitent au sein d'une architecture orientée service Web sont, en général, basés sur XML afin de permettre l'échange de données structurées, indépendamment des langages de programmation ou des systèmes d'exploitation. Les types de données utilisés sont eux aussi basés sur XML, c'est ce qu'on appelle l'encodage [4]. Deux cas peuvent être distingués :

- "Literal" (suit littéralement les définitions de schéma XML - XML Schema [12]) ;
- "SOAP encoded" (suit la spécification de SOAP [11]).

SOAP

Le protocole **SOAP** (**S**imple **O**bject **A**ccess **P**rotocol) est, entre autre, issu des efforts communs de Microsoft et IBM dès la fin de l'année 1999. D'autres tentatives eurent lieu auparavant ou en parallèle afin de définir un protocole de communication adapté à la fois à l'Internet et aux Intranet. Bien entendu, le poids d'acteurs comme Microsoft ou IBM n'est pas sans effets sur le succès de SOAP et l'acceptation progressive de ce protocole par tous les acteurs du marché [9].
Un message SOAP est un document XML dont la structure est spécifiée par des schémas XML [4]. La structure d'un message SOAP sera étudiée en détail dans la prochaine section. Le protocole SOAP est une sur-couche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP, FTP, etc. [13].

Structure d'un message SOAP [13]
Cette structure est schématisée par la figure 1.3.

L'en-tête du protocole de transport
Cette partie dépend du protocole de transport utilisé. Par exemple, si le protocole HTTP est utilisé, cet en-tête contient :

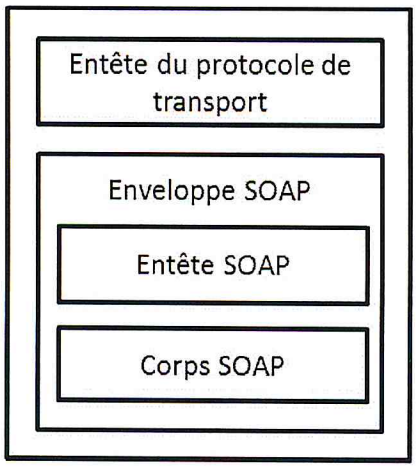


FIGURE 1.3 – Schéma d'un message SOAP.

- la version de HTTP utilisée ;
- la date de génération de la page (qui est ici le message SOAP lui-même) ;
- le type d'encodage du contenu (ici, l'encodage est généralement le type text/xml), etc.

Les messages SOAP (l'enveloppe)
La partie principale d'un message SOAP est l'enveloppe (symbolisée par la balise `envelope`). Cette enveloppe est subdivisée en deux sous-parties : la partie en-tête et la partie corps du message. Elle permet également de spécifier des environnements de noms XML utilisés dans la suite du message.

L'en-tête du message SOAP
L'en-tête SOAP (SOAP Header) est optionnelle et extensible. Les balises XML permettant de symboliser cette partie sont `<env:Header>` et `</env:Header>`. Ces balises peuvent être complétées par des attributs permettant de définir le domaine de noms du service Web.
En fait, l'en-tête permet principalement d'ajouter des informations sur le comportement que doivent avoir les différents nœuds intermédiaires, lors du traitement du message. Un nœud étant un intermédiaire SOAP, incluant le récepteur et l'émetteur SOAP, désignable depuis un message SOAP. Son rôle est de traiter l'en-tête (et d'effectuer les actions qui y sont décrites) et ensuite de transférer le résultat (le message SOAP modifié) à un autre intermédiaire (qui peut être le récepteur final). Par extension, la description du comportement des différents nœuds permet également de réaliser une "composition de services", car le message peut être routé entre différents nœuds, chacun étant capable de réaliser une action précise et décrite dans le bloc d'en-tête. Les principaux attributs des éléments formant le bloc sont :

`env:role` : permet d'indiquer à quel nœud la fonction décrite est destinée. Et donc par extension, cela permet le routage d'un message ;

`env:mustUnderstand` : c'est une valeur booléenne, elle permet de préciser que le traitement devient obligatoire pour un nœud intermédiaire ;

`env:relay` : cet attribut permet de relayer un message à un autre nœud si le premier

nœud n'est pas capable de le traiter.

Le corps du message SOAP
Les données spécifiques à l'application se trouvent dans le corps du message SOAP (SOAP Body). Tout ce qui est présent dans cette section est défini lors de la conception de l'application. Enfin, les balises symbolisant cette partie sont `<env:Body>` et `</env:Body>`. Les données doivent donc être sérialisées selon l'encodage choisi. L'utilisation du XML 1.0 à l'intérieur des blocs XML `<element>` permet d'envoyer absolument tous les types de documents comme par exemple des feuilles de styles, des documents XML, des images au format binaire, etc.
En plus des données, cette partie peut transporter un type spécial : les messages d'erreurs (SOAP Fault).
On peut également ajouter un attribut dans la balise `<env:Fault>` permettant d'indiquer un type d'encodage des données. Il existe aussi d'autres attributs permettant la gestion de l'erreur, comme `Code`, `Reason`, `Node`, `Role` et `Detail`.

La couche description

Dans le cadre des recommandations modèle de référence **OASIS** (**O**rganization for the **A**dvancement of **S**tructured **I**nformation **S**tandards), 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.
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 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. Toutefois, dans un environnement ouvert comme Internet, le modèle de description des services Web n'est d'aucune utilité s'il n'existe pas un moyen de localiser aussi bien les services que leurs descriptions WSDL. Un troisième standard a été conçu pour réduire l'écart entre les applications clientes et les services Web, appelé UDDI [4].

a. WSDL

La spécification WSDL (Web Services Description Language) a été normalisée par le W3C [13]. Cette spécification présente les services comme des boites noires et s'intéresse à fournir une abstraction fonctionnelle du service. Elle est composée de deux parties [4] :

- une définition abstraite des services en termes de messages échangés ;
- la définition des mécanismes de liaison entre les définitions abstraites.

Chaque document WSDL est divisé en sept sections différentes [9] :

1. **La section "type"** : Dans cette section, on va définir les structures XML qui constituent une partie du contenu des messages échangés avec un service Web. Pour définir ces structures, on utilise le modèle XML Schéma également spécifié par le W3C ;
2. **La section "message"** : par l'intermédiaire de cette section, on définit le format des messages échangés. Cette section peut faire référence aux types définis dans la

section "type". A noter plusieurs types prédéfinis sont à disposition ;

3. **La section "opération"** : cette section décrit les opérations invoquées de manière distante sur le service Web. Chaque opération est décrite à l'aide de deux ou trois messages :
   - ▷ Le message reçu par le service Web lorsqu'il est sollicité pour une requête ;
   - ▷ Le message émis en réponse par le service Web ;
   - ▷ L'éventuel message d'erreur retourné par le service Web en cas de problème.

   A noter également : un ou plusieurs des messages précédents peuvent être absents de la description d'une opération ;
4. **La section "type de port"** : la section type de port (ou portType) regroupe simplement un ensemble d'opérations. Cette section sera par la suite utilisée pour préciser que l'ensemble de ces opérations est offert par le même service Web. On peut comparer cette section à une interface Java ;
5. **La section "liaison"** : (ou Binding) effectue la liaison précise entre les descriptions effectuées dans les sections "message" et "opération" et le type de protocole utilisé pour véhiculer ces données. En effet, le langage WSDL est défini pour décrire les services Web mais est indépendant du protocole utilisé, en l'occurrence SOAP. La section "liaison" précise justement quel est le protocole employé.
   Ainsi, dans le cas où l'on fait appel au protocole SOAP, on va caractériser les messages et opérations en complétant les descriptions précédemment effectuées des messages et des opérations par des informations relatives à SOAP ;
6. **La section "port"** : l'objectif de cette section est de préciser le point d'accès à un service Web. Chaque point d'accès est unique et associe un URI à une liaison, c'est-à-dire un point d'accès à un mécanisme d'utilisation (par exemple l'utilisation du protocole SOAP) ;
7. **La section "service"** : cette dernière section caractérise un service Web en énumérant ses divers points d'accès, c'est-à-dire en listant une ou plusieurs sections "port". Ainsi, un même service Web peut être accessible par l'intermédiaire de plusieurs points d'accès où chaque point d'accès est caractérisé par un mode de communication différent. On peut très bien imaginer qu'un service Web comporte deux sections "port" : une pour un accès SOAP/http et une autre pour SOAP/SMTP.

b. UDDI

UDDI (**U**niversal **D**escription, **D**iscovery and **I**ntegration) est un annuaire de service accessible sur le Web décrivant les sociétés et les services proposés, aussi bien d'un point de vue fonctionnel que technique, permettant aux utilisateurs de découvrir les services disponibles [15]. Il fournit un protocole, une API et une plateforme permettant aux utilisateurs de services Web, depuis n'importe quel système, de localiser dynamiquement à travers Internet les services Web qu'ils désirent utiliser [13].

Les différents rôles d'UDDI
UDDI fournit trois services de bases [13] :

Publish : ce service gère comment le fournisseur de service Web s'enregistre lui-même ainsi que ses services (en utilisant UDDI) ;

Find : ce service gère comment un client peut localiser le service Web désiré (cela peut passer par des invocations de services Web pour une utilisation automatique par un programme ou par une consultation d'annuaire en utilisant des mots clés) ;

Bind : ce service gère également comment un client peut se connecter et utiliser le service Web une fois celui-ci localisé.

Contenu de l'annuaire

Le contenu de l'annuaire est représenté sous forme d'entités liées (figure 1.4). Il s'agit de **pages blanches** définissant les sociétés participantes (*Business Entity*), de **pages jaunes** décrivant les services (*Business Service*) et de **pages vertes** (Binding Templates) donnant les informations techniques [15].
Plusieurs modèles techniques sont supportés (*tModel*), en particulier la description en WSDL du service. Des relations entre parties peuvent aussi être publiées (*Publisher Assertion*) [15].

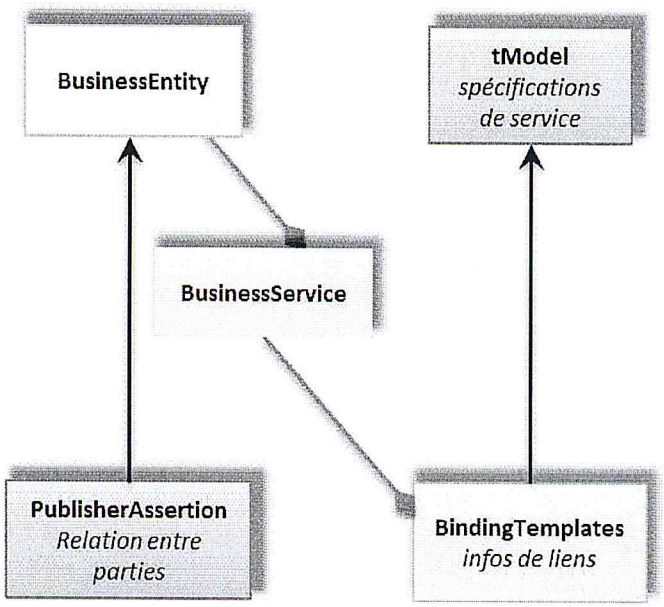


FIGURE 1.4 – Entités composant un annuaire [15].

La couche qualité de service

a. La sécurité

Cette partie sera étudiée en détail dans le prochain chapitre.

1.4 CORBA

L'architecture **CORBA** (Common Object Request Broker Architecture), définie par l'**OMG** (Object Management Group) et dont l'élaboration a commencé en 1989, est un standard ouvert de la programmation distribuée orientée objets. Les points prépondérants

adressés par CORBA sont la réutilisabilité, la portabilité et l'interopérabilité d'applications orientées objets dans un environnement distribué et hétérogène en terme de systèmes et de langages de développement [17].

1.4.1 Architecture CORBA

CORBA est l'une des premières spécifications de l'OMG. Elle détaille les interfaces et les caractéristiques de l'**ORB** (**O**bject **R**equest **B**roker). La figure suivante illustre les composantes de CORBA introduites dans la spécification CORBA.

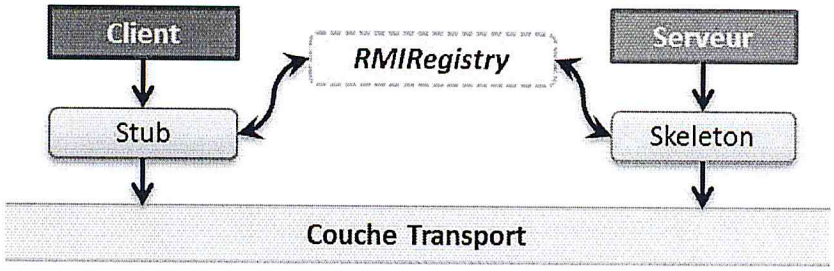


FIGURE 1.5 – Architecture CORBA.

Bus objet (ORB)

L'ORB (Object Request Broker) constitue le noyau de l'architecture CORBA. Il définit le support grâce auquel un client peut invoquer les services d'un objet dans un environnement distribué hétérogène. Le but de l'ORB est de fournir un certain nombre de services au client, pour faciliter l'invocation d'une opération sur un objet en faisant abstraction de la distance et de la localisation, mais aussi de l'implémentation de l'objet serveur [16]. L'ORB gère les problèmes liés à la localisation d'un objet, à sa disponibilité, et à la gestion d'autres services comme la sécurité [16].

Langage de définition d'interfaces (IDL)

Pour décrire les interfaces des objets CORBA, l'OMG a défini le langage IDL qui est fortement inspiré de C++ dans sa syntaxe. L'interface fournit au client les spécifications des opérations invoquables sur l'objet serveur. La spécification des interfaces est la première étape de l'implémentation d'une application objet distribuée. L'IDL doit permettre, dans le processus de développement, une mise en correspondance rapide entre le modèle objet de l'application et son implémentation dans le langage cible. L'IDL ne permet de spécifier que les interfaces des objets. C'est pourquoi il n'inclut aucune structure algorithmique et ne permet pas la définition de variables [16].

Un exemple d'interface définie au moyen de l'IDL est :

```
interface compte-bancaire {
attribute readonly double balance;
void dépôt(double montant);
void retrait(double montant);
double balance();
};
```

Une caractéristique importante de l'IDL de OMG est son indépendance de tout langage. Comme c'est un langage déclaratif, et non un langage de programmation, les interfaces sont définies indépendamment des implémentations. Ceci permet aux objets d'être construits en utilisant différents langages de programmation et de communiquer les uns avec les autres. L'IDL offre un ensemble de types qui sont similaires à ceux trouvés dans plusieurs langages de programmation. Il offre des types de base tels que long, double et boolean, et des types construits tels que **struct**, **union**, **sequence** et **string**. Les types sont utilisés pour préciser les types des paramètres et des valeurs de retour des opérations. Comme vu dans l'exemple ci-dessus, les opérations sont employées dans des interfaces pour préciser les services offerts par les objets qui supportent ce type particulier d'interface [16].

a. Traduction de l'IDL vers d'autres langages

Comme indiqué ci-dessus, l'IDL est seulement un langage déclaratif, et non un langage complet de programmation. Ainsi, il n'offre pas de caractéristiques comme les structures de contrôle et il n'est pas utilisé directement pour réaliser des applications distribuées. En effet, les traductions de langage (language mapping) déterminent comment les caractéristiques de l'IDL sont converties en concepts d'un langage de programmation. L'OMG a standardisé jusqu'à présent les transformations de l'IDL vers les langages C, C++, Smalltalk, Ada 95, Java, et COBOL [16].

Répertoire d'interfaces (Interface Repository)

Le répertoire d'interfaces (IR) représente la composante de l'architecture OMA qui permet le stockage des définitions IDL des interfaces des objets, des méthodes qu'ils supportent et des paramètres qu'ils nécessitent (signatures des méthodes). Il constitue une base de données des interfaces d'objets. L'IR fournit l'information nécessaire à l'émission de requêtes utilisant l'Interface d'Invocation Dynamique (Dynamic Invocation Interface) de la figure 1.5 [16].

Talons clients et talons serveurs (client IDL stubs and Static skeletons)

L'étape qui suit la définition des interfaces en IDL est la génération des modules d'interaction avec l'ORB correspondant aux interfaces spécifiées. En effet, la spécification IDL est purement abstraite et n'intègre aucun élément propre aux communications client-serveur via l'ORB. Ces éléments sont intégralement générés par la compilation des interfaces dans le langage cible. A cet effet, les solutions de développement CORBA intègrent systématiquement un compilateur IDL permettant de générer les fichiers nécessaires à l'implémentation du client et du serveur et qui sont les stubs et les skeletons de la figure

1.5.
Les talons serveur (skeletons) permettent le traitement des requêtes provenant des talons client (stubs) ou de l'interface d'invocation dynamique. L'adaptateur d'objets (Object Adapter) délègue aux talons serveur l'invocation des méthodes de l'objet, le décodage des requêtes CORBA, le codage des résultats de la requête, et le choix du bon "code" pour traiter la requête [16].

L'interface DII

Elle permet à un client en cours d'exécution, d'invoquer un serveur sans passer par un talon et donc sans connaître au préalable l'interface IDL de ce serveur [17].

L'interface DSI

Elle permet à un serveur de traiter toute requête dynamiquement, c'est-à-dire, sans avoir de squelette statique [17].

Interface ORB (ORB Interface)

Cette interface est directement accessible à la fois par le client et par l'objet serveur. Elle regroupe un ensemble d'opérations et de fonctionnalités de l'ORB qui sont communes à tous les objets, telles que la conversion des références objet en chaînes de caractères ou encore la copie d'objet. Elle ne permet pas à proprement parler d'invoquer des services. Elle s'occupe de la gestion des objets dans le système [16].

Adaptateur objet (Object Adapter)

Dans un environnement hétérogène, les implémentations des objets serveurs sont fortement liées au système hôte. La diversité de cet environnement oblige à définir une interface entre le noyau de l'ORB et l'implémentation. Cette interface est appelée Adaptateur Objet (Basic Object Adapter ou BOA). A travers l'adaptateur objet, l'implémentation de l'objet peut accéder aux services offerts par l'ORB [16]. La première invocation d'un objet se traduit au niveau de l'ORB et de l'adaptateur objet par :

1. L'activation de l'implémentation de l'objet serveur, c'est-à-dire le lancement du programme qui correspond à l'implémentation de l'objet ;
2. L'implémentation activée s'enregistre auprès de l'adaptateur objet en signalant sa disposition à traiter ou non la requête ;
3. L'objet serveur est ensuite activé par l'adaptateur objet ;
4. L'adaptateur objet invoque enfin la méthode correspondante à la requête.

1.5 RMI

1.5.1 Définition

RMI (Remote Method Invocatoin) est une API de communication entre objets distants de Java présente depuis longtemps dans la version standard de Java JRE (Java Runtime Environement). Contrairement à CORBA (section précédente), RMI ne permet pas que

la communication entre programmes Java à l'exclusion de tout autre langage, par contre, il assure des échanges entre des systèmes d'exploitation différents possédant une machine virtuelle Java. RMI s'appuie sur un protocole de type RPC (Remote Procedure Call) [18].

1.5.2 Architecture RMI

La figure 1.6 donne un aperçu général de l'architecture de RMI [19].

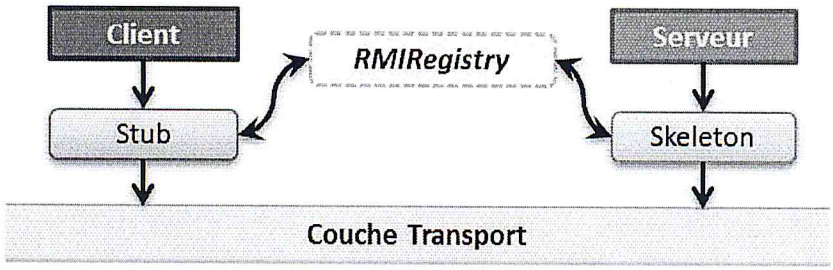


FIGURE 1.6 – Architecture RMI.

Un compilateur RMI (***rmic***) est utilisé pour générer les souches côté client (***stub***) et côté serveur (***skeleton***) selon les mécanismes utilisés au fil des différentes versions. Dans la version actuelle de la plate-forme Java, cette étape peut être omise et le ***stub*** et ***skeleton*** seront générés à la volée, durant l'exécution.
Le service ***RMIRegistry*** est le service de nommage propre à RMI. Il permet aux objets distants d'être référencés dans un annuaire, et aux clients de retrouver les références des objets distants à partir de leur nom de service associé. Il est fortement recommandé pour des raisons de sécurité, que le serveur et le **rmiregistry** soient sur la même machine.

1.5.3 La mise en œuvre de RMI

Selon [19], la mise en œuvre d'une application RMI passe par deux phases majeures :

1.5.4 Phase de Définition/Génération

La déclaration des services distants s'effectue à l'aide d'une interface en langage Java qui étend, de manière directe ou indirecte, l'interface **java.rmi.Remote**. Un objet serveur doit implémenter cette interface pour être utilisable avec RMI. De manière optionnelle, il peut également étendre la classe **UnicastRemoteObject**, qui propose quelques méthodes d'automatisation des tâches de mise en liaison avec le bus RMI. A partir de cette implémentation, le compilateur ***rmic*** génère un ***Stub*** qui sera utilisé côté client.

1.5.5 Mise en place d'un service et invocation

Lorsqu'une instance de Serveur est créée, celle-ci doit être exportée sous la forme d'une référence distante. L'objet **UnicastRemoteObject** propose des méthodes dédiées à cette tâche. Lors de l'enregistrement du serveur dans la ***RMIRegistry***, la référence distante ainsi créée est transmise, accompagnée du nom utilisé pour référencer le service.
Pour accéder au serveur, le client fait également appel à la ***RMIRegistry***, à laquelle il transmet le nom du service sous lequel le serveur est enregistré. La référence distante est

alors transmise au ***Stub***, qui se charge de transmettre les invocations du client vers le serveur. L'encodage et le décodage des données sont réalisés par sérialisation, mécanisme standard à la plate-forme Java pour la transmission d'objets à travers des flux, de quelque nature qu'ils soient.

1.6 Comparaison

	RMI	CORBA	Service Web
Niveau d'interopérabilité	Bonne interopérabilité	Très Bonne interopérabilité	Meilleur interopérabilité
Description de l'interface	Interface Java	IDL	WSDL
Utilisation de l'annuaire	Obligatoire (RMIRegistry)	Obligatoire (CORBA services)	Non obligatoire (UDDI)
Utilisation d'un Middleware commun	Non [2]	Oui	Non
Contourner Firewall	Non	Non	Oui
Performance (Temps de réponse)	Performance moyenne (moins que WS)	Performance très élevée (plus que WS)	Performance élevée

TABLE 1.1 – Comparaison entre CORBA, RMI et service Web.

1.6.1 Niveau d'interopérabilité

L'auteur de [20], affirme que les services Web permettent un grand niveau d'interopérabilité sur l'Internet que les standards RMI et CORBA. En effet, les Services Web ont été conçus pour répondre en premier lieu à des problèmes d'interopérabilité sur l'Internet, que les middlewares conventionnels comme CORBA, DCOM et RMI ont été incapables de résoudre de façon satisfaisante.
Ces inter-logiciels ont été développés à une époque où les systèmes distribués étaient limités à un réseau local ou privé d'une entreprise, d'un gouvernement ou d'une université. Ils ont été conçus pour répondre à des exigences spécifiques et dans un contexte bien limité. L'idée de créer un inter-logiciel assez robuste pour s'adapter à n'importe quelle situation ou problème a été mise en place dans la création de CORBA. Par exemple, quand l'Internet a été disponible pour l'utilisation civile, CORBA a été rapidement adapté pour travailler sur TCP/IP et le tunneling HTTP.
Techniquement, ceci a été possible grâce au protocole GIOP/IIOP. Cependant, il s'agit d'une adaptation et non d'une solution faite pour prendre en charge le contexte de l'Internet [21].

2. JRE (Java Runtime Environnement) doit être installé sur toutes les machines

1.6.2 Description de l'interface

Java RMI est un inter-logiciel spécifique à la plate-forme Java, les interfaces sont définies en Java. Par contre les services Web et CORBA définissent une description d'interface indépendante de la plateforme.
Les services Web présentent des avantages significatifs par rapport à CORBA. Le WSDL contient deux parties, l'une abstraite et l'autre concrète. La partie abstraite contient le contrat du service, et la partie concrète contient les liaisons (bindings) et la localisation du service. ***IDL-CORBA*** ne fournit par contre que la partie abstraite, c.-à-d. le contrat du service. Les types de données codifiées en ***IDL-CORBA*** sont prédéfinis, bien qu'existe la possibilité de créer des types complexes (par exemple, structures, unions) et des types dynamiques avec le constructeur ***any*** [21].
Par opposition, WSDL permet la description de types qui ne sont pas prédéfinis dans sa spécification, ceci grâce à l'extensibilité de XML et les schémas XML, en plus XML a été conçu pour être extensible et compréhensible par les humains et aussi par les ordinateurs [21].

1.6.3 Utilisation de l'annuaire

L'utilisation de l'UDDI est secondaire si l'application cliente possède le fichier **wsdl**. Tandis que la présence de l'annuaire dans le cas d'RMI et CORBA est obligatoire, car ce dernier permet à l'application cliente de trouver la référence d'un objet réalisant un service (la fréquence de changement de la référence est plus élevée que la fréquence de changement d'un fichier **wsdl**).

1.6.4 Utilisation d'un Middleware commun

Un des avantages des SOA réside dans le fait que les applications réparties n'ont plus besoin d'un système de middleware réparti commun pour communiquer, mais seulement des protocoles et des technologies de communication interopérables sur Internet [23].
Dans le cas du RMI, la présence du JRE (Java Runtime Environnement) sur toutes les machines est obligatoire, pour que l'application fonctionne.

1.6.5 Contourner Firewall

Par l'utilisation du protocole HTTP, les Services Web peuvent contourner les mesures de sécurité mises en place au travers des pare-feux [22]. D'après une citation du Gartner Group faite en 2003, "Les Services Web XML vont rouvrir 70% des chemins d'attaques fermés par les pare-feux lors de la dernière décennie. Ils peuvent transporter virtuellement toutes les données utiles sur le port 80 et le pare-feu ne peut les arrêter".
Par contre, RMI et CORBA utilisent autres protocoles de communication (JRMP et GIOP respectivement), qui nécessite l'établissement des règles de filtrages autorisant le passage de flux des données par un port spécifique (utilisé par JRMP et GIOP).

1.6.6 Performance

RMI vs service Web

Dans [24], l'auteur présente une étude détaillée de performance entre Java RMI et les services Web. Cette étude a démontré que l'utilisation de Java RMI est 8.5 plus performante que l'utilisation de services Web sur un réseau local. Cependant, l'auteur remarque que sur l'Internet, l'utilisation de Java RMI demande souvent une technique de tunneling comme HTTP-to-port, HTTP-to-CGI et HTTP-to-Servlet. L'effort pour déployer et configurer Java RMI et les composants de tunneling est plus significatif que dans le cas de services Web. De plus, Java RMI et les techniques de tunneling ont présenté une performance inférieure aux services Web. L'utilisation de Java RMI et HTTP-to-port a été quatre fois moins performante que les services Web. Et l'utilisation de Java RMI et HTTP-to-servlet a été trois fois moins performante que les services Web. Selon les résultats obtenus, l'auteur conseille l'utilisation de services Web au détriment de Java RMI avec la technique de tunneling, et il recommande l'utilisation de Java RMI (sans tunneling) pour les scénarios qui n'ont pas des problèmes avec les pare-feux ou qui demandent plus de performance que d'interopérabilité. Dans [25], l'auteur a montré que, dans certains contextes et avec l'utilisation de techniques d'optimisation, les services Web peuvent être plus performants que Java RMI (sans tunneling).

CORBA vs service Web

Dans [26], l'auteur présente une étude détaillée de performance entre CORBA et les services Web. Cette étude a démontré que CORBA est 400 fois plus performant que les services Web, mais avec l'utilisation d'un parseur XML le plus adapté au document utilisé dans l'application, ce rapport peut être réduit à sept.
L'article [20] montre une étude comparative entre CORBA, RMI et les Web services, l'auteur a commencé par présenter trois applications de tests :

- Application $n°1$: un calculateur simple, afin de montrer un transfert des données simples (entiers) ;
- Application $n°2$: consiste à un transfert des données structurées (structure des données) ;
- Application $n°3$: consiste à un transfert des données complexes (objets), il l'a nommée application réelle.

Ensuite, l'auteur a signalé qu'un service Web reste moins performant que CORBA malgré toutes les améliorations proposées, mais ces différences sont moindre dans le cas des applications réelles que les applications un et deux. Donc les services Web sont plus adoptés aux applications réelles que les petites applications.
A partir de cette comparaison, nous avons choisi d'utiliser les services Web puisque ces derniers permettent :

- Un haut niveau d'interopérabilité sur l'internet ;
- Une description des interfaces indépendante de la plateforme utilisée ;
- De contourner les pare-feux ;
- Une performance acceptable ;
- Ne nécessite pas la présence d'un annuaire (UDDI).

1.7 Conclusion

Dans ce chapitre, nous avons présenté une étude comparative entre trois solutions de réalisation des systèmes distribués (service Web, CORBA et RMI). L'approche service Web est choisie pour implémenter notre application " système de communication ", ce choix est dû aux avantages qu'il possède comme : l'utilisation de standards universels, l'indépendance de plate-forme, un environnement universel pour les systèmes d'information distribués, l'utilisation de plusieurs protocoles de transfert (par exemple HTTP, SMTP et FTP), le codage des messages en XML, un comportement compatible aux pare-feu, la localisation par URI (Uniform Resource Identification) et les techniques proposées pour sécuriser un service Web.
Dans le chapitre suivant, nous allons concentrer notre étude sur les attaques contre les services Web, les failles éventuelles ainsi les techniques proposées pour sécuriser un service Web.

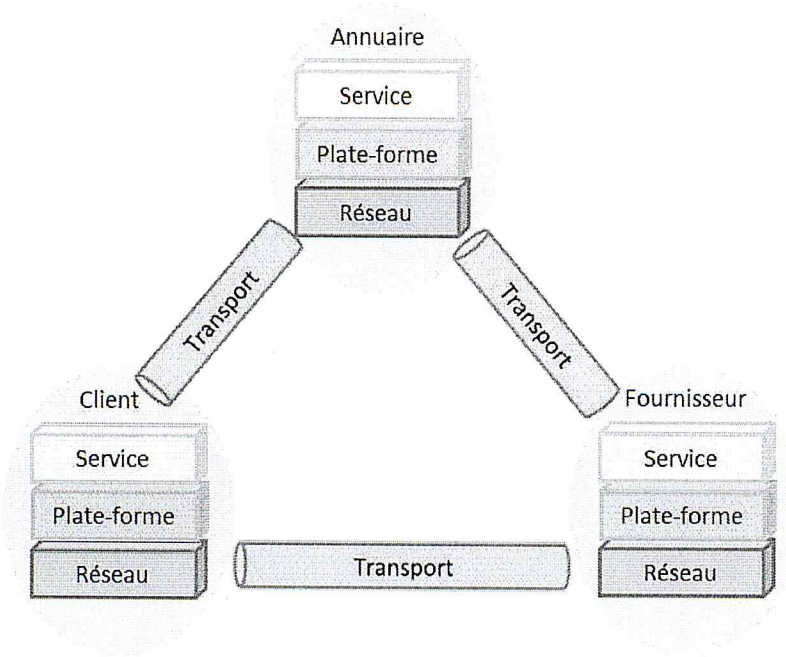


FIGURE 2.1 – Communications à travers différentes couches techniques [27].

internes de sécurité : par exemple, s'il nécessite un accès à une base de données, cette dernière doit aussi être sécurisée.
A ceci s'ajoutent les failles de sécurité qui existent **au niveau physique du réseau**. Un réseau physique est composé de nombreuses machines qui sont, par exemple, des serveurs pour les services, pour les bases de données... Chaque machine a des failles de sécurité en fonction des ports de communication qui sont ouverts, de son système d'exploitation... Pour prévenir d'une partie des risques, il existe des moyens techniques à mettre en place comme créer une zone démilitarisée et installer des pare-feux.
Dans cette section, nous allons décliner les quatre types d'attaques possibles pour les architectures à services, en particulier entre le client et le fournisseur.

2.2.1 Attaque passive d'écoute du réseau et d'analyse du trafic

Le consommateur et le fournisseur de service communiquent via un réseau Internet ou Intranet. Ces communications peuvent être à tout moment interceptées par une personne ou une machine malveillante à des fins délictueuses, comme illustrées dans la figure 2.2. Le but de cette attaque est de récupérer des informations provenant du consommateur et du fournisseur de service. Pour ce faire, une machine écoute les communications et vole les informations transmises entre les deux parties. Ces informations peuvent être ensuite divulguées à n'importe qui. Pour lutter contre ce type d'attaque, il faut que les informations qui circulent sur le réseau soient et restent ***confidentielles***, c'est-à-dire compréhensibles uniquement par le consommateur et le fournisseur de service. Une solution pour rendre les informations confidentielles est de les ***chiffrer***.

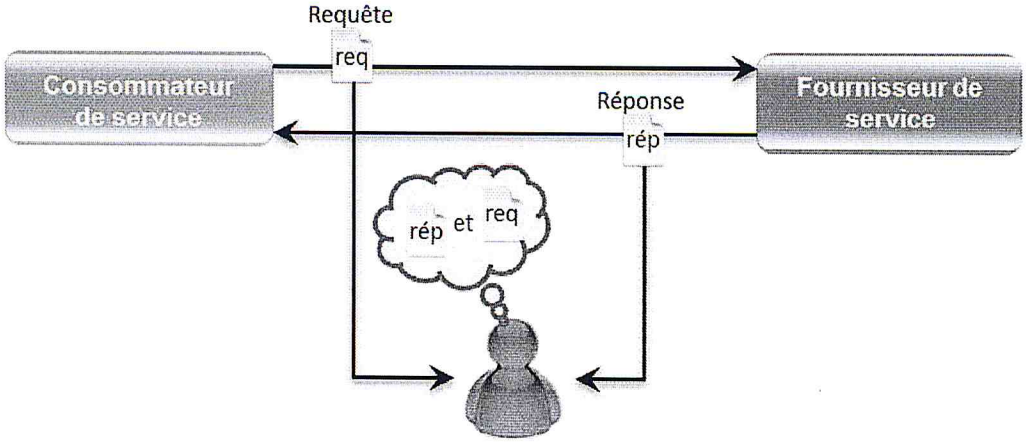


FIGURE 2.2 – Divulgation d'informations.

2.2.2 Tromperie

L'attaque par tromperie est une attaque plus sophistiquée que l'attaque par écoute du réseau : une personne ou une machine malveillante se fait passer pour un "vrai" client auprès du fournisseur de service. La machine malveillante interroge le fournisseur avec les informations d'un "vrai" client et récupère des informations, qui peuvent être confidentielles. La figure 2.3 synthétise le fonctionnement de cette attaque. Pour cette

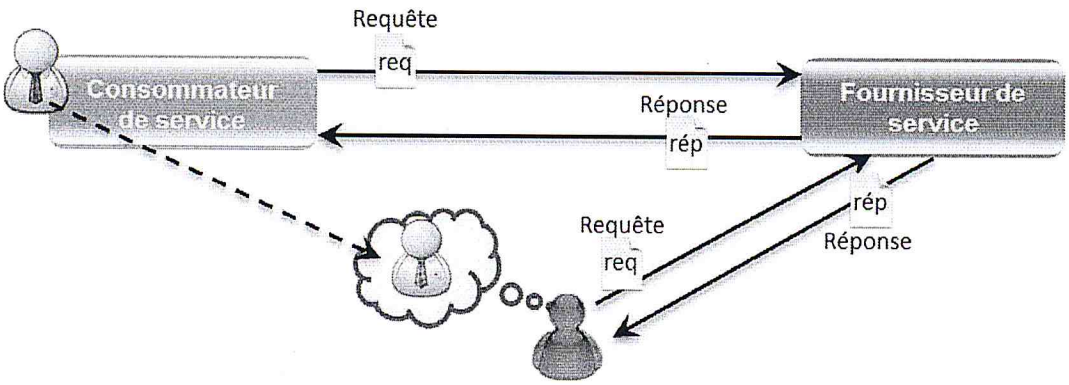


FIGURE 2.3 – Tromperie sur l'identité du client.

attaque, la machine malveillante émet des messages en volant l'identité d'un "vrai" client. Les messages sont donc échangés entre le fournisseur et la machine malveillante sans que le fournisseur se rende compte de la supercherie. Ce type d'attaque peut se faire après avoir écouté longtemps le réseau pour utiliser les informations récupérées ou en piratant les informations du client sur sa machine.
Pour que le fournisseur de service soit sûr de l'identité de son client, il faut mettre en place un système d'***authentification*** qui permette de reconnaître les "vrais" clients des pirates. En plus de ce système d'authentification, il faut que les informations d'authentification ainsi que les autres informations échangées restent ***confidentielles*** entre le client et le fournisseur pour qu'il n'y ait pas de vol ni d'écoute par une machine malveillante.

2.2.3 Détournement d'informations

Le détournement d'informations est une attaque qui a des effets différents par rapport aux deux attaques présentées précédemment. En effet, les informations du consommateur sont détournées par une personne ou une machine malintentionnée et elles sont modifiées à des fins malveillantes.
Le principe de cette attaque est d'altérer les données du consommateur afin d'endommager le service du fournisseur. La personne ou la machine malintentionnée intercepte un message d'un "vrai" client s'adressant au fournisseur. Elle modifie le message initial en y insérant du code malveillant. Le message est ensuite envoyé au fournisseur de service qui l'interprète comme un message provenant du client qu'il connaît. Le code ajouté peut, par exemple, être un virus qui endommage le fournisseur de service. La Figure 2.4 illustre le scénario de cette attaque. Cette attaque a pour conséquence immédiate de faire ***perdre***

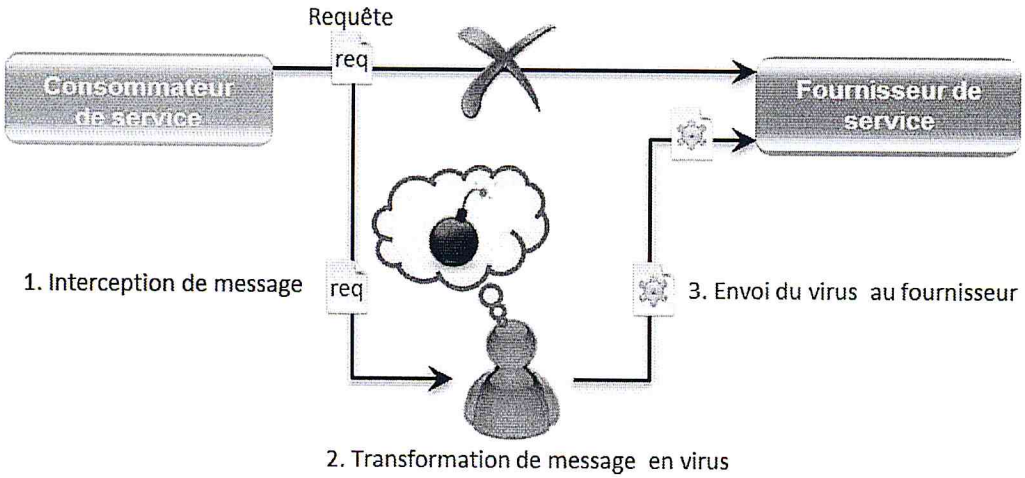


FIGURE 2.4 – Détournement d'informations.

les informations contenues dans le message du client. Dans un deuxième temps, elle ***endommage le serveur*** qui est alors dans l'incapacité de continuer à répondre correctement aux autres requêtes.
Pour lutter contre ce type d'attaque, il faut que les informations qui circulent restent ***intègres***.

2.2.4 Déni de service

L'attaque par déni de service est une attaque qui a pour but de rendre indisponible le service pendant un temps indéterminé. Cette menace concerne en général le fournisseur de service. Son but n'est pas de s'approprier ou d'altérer les données du service, mais uniquement de nuire à la performance et à la réputation du service. Le fournisseur de service peut ainsi perdre la confiance de ses clients. Le principe de l'attaque par déni de service consiste à saturer le fournisseur de service, en lui envoyant, notamment trop de messages à traiter ou des messages empoisonnés. Pour la technologie des services Web, les messages XML permettent d'attaquer les services. En effet, il suffit d'envoyer un message XML empoisonné. Ce message peut nécessiter, par exemple, une récursion infinie pour l'étape de **parsing** du message à la réception de celui-ci par le fournisseur. Une autre

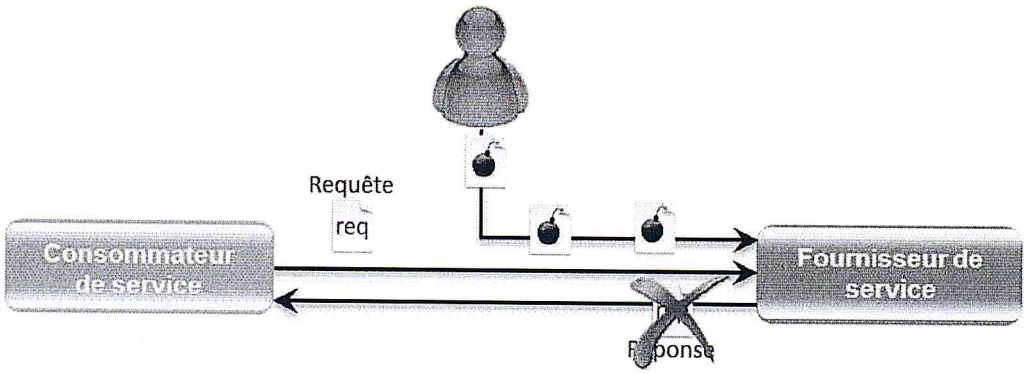


FIGURE 2.5 – Interruption de service.

méthode est de saturer le service en envoyant un trop grand nombre de messages, le serveur prendra beaucoup de temps pour les traiter tous et sera donc indisponible pour les "vrais" clients.
Les méthodes d'attaque pour saturer un service sont finalement assez simples mais assez difficiles à détecter, puisque les messages malveillants se font passer pour des messages normaux. Le seul moyen pour s'assurer du bon fonctionnement des services est de mettre en place une surveillance des services qui permettent de suivre le temps de traitement des messages et des opérations.

2.3 Notions de base sur la sécurité informatique

2.3.1 Définition de la sécurité

La sécurité informatique c'est l'ensemble des moyens et méthodes mis en œuvre pour assurer la protection des ressources [28].
Il est possible de préciser la notion de risque en la décrivant comme le produit d'un préjudice par une probabilité d'occurrence [29] :

$$\text{risque} = \text{préjudice} \times \text{probabilité d'occurrence}$$

Cette formule exprime qu'un événement dont la probabilité est assez élevée, par exemple la défaillance d'un disque dur, mais dont il est possible de prévenir le préjudice qu'il peut causer, par des sauvegardes régulières, représente un risque acceptable.

2.3.2 Critères fondamentaux de la sécurité informatique

Les solutions de sécurité qui seront mises en place doivent contribuer à satisfaire les critères suivant [30] :

La disponibilité

Pour un utilisateur, la disponibilité d'une ressource est la probabilité de pouvoir mener correctement à terme une session de travail. La disponibilité d'une ressource est indisso-

ciable de son accessibilité : il ne suffit pas qu'elle soit disponible, elle doit être utilisable avec des temps de réponse acceptable.

L'intégrité

L'intégrité permet de certifier que les données, les traitements ou les services n'ont pas été modifiés, altérés ou détruits tant de façon intentionnelle qu'accidentelle.

La confidentialité

La confidentialité peut être vue comme "la protection des données contre une divulgation non autorisée", il existe deux actions complémentaires permettant d'assurer la confidentialité des données :

- Limiter leur accès par un mécanisme de contrôle d'accès ;
- Transformer les données par des procédures de chiffrement afin qu'elles deviennent inintelligibles aux personnes ne possèdent pas les moyens de les déchiffrer.

L'identification et l'authentification

Un nom associé à des caractéristiques identifie une entité : individu, ordinateur, programme, document, etc. L'identification est la reconnaissance de cette entité.
L'authentification permet de vérifier l'identité annoncée et de s'assurer de la non-usurpation de l'identité d'une entité. Pour cela, l'entité devra produire une information spécifique telle que par exemple un mot de passe (un code, un mot de passe, une empreinte biométrique, etc.). L'identification et l'authentification assurent :

- La confidentialité et l'intégrité de données : seules les entités identifiées et authentifiées peuvent accéder aux ressources et les modifier s'ils sont habilités à le faire ;
- La non-répudiation et l'imputabilité : seules les entités identifiés et authentifiées ont pu réaliser telle action par exemple : preuve de l'origine d'un message ou d'une transaction.

La non-répudiation

Est le fait de ne pouvoir nier ou rejeter qu'un événement (action, transaction) a eu lieu. A ce critère de sécurité sont associées les notions d'imputabilité, de traçabilité et éventuellement d'auditabilité.

- L'imputabilité se définit par l'affectation certaine d'une entité à une action ou un événement.
- La traçabilité est la fonction de sécurité qui comprend, l'imputation, mais qui mémorise l'origine d'un message, d'un événement ou d'une donnée. Elle permet par exemple, de retrouver l'adresse à partir de laquelle ces données ont été envoyées.
- L'auditiabilité se définie par la capacité d'un système à garantir la présence des informations nécessaires à une analyse ultérieure d'un événement (courant ou exceptionnelle) dans le but de déterminer s'il y a effectivement eu violation de la sécurité, et dans ce cas, quelles informations ou autres ressources ont été compromises.

2.3.3 Techniques de base de sécurité

Le chiffrement

Le chiffrement est une technique de codage consistant à rendre des données indéchiffrable pour tout autre utilisateur que le destinataire de message. Il existe deux types de chiffrement : le chiffrement à clef secrète ou chiffrement symétrique, et le chiffrement à clef publique ou chiffrement asymétrique [31].

a. Le chiffrement à clef secrète

Le chiffrement à clef secrète repose sur un algorithme qui peut être connu de tous et une valeur, appelée clef secrète, uniquement connue des deux interlocuteurs. Le message est codé par l'émetteur avec la clef secrète et il est décodé par le récepteur avec la même clef (voir figure 2.6) [31]. Les algorithmes à clef secrète se séparent en deux sous catégories,

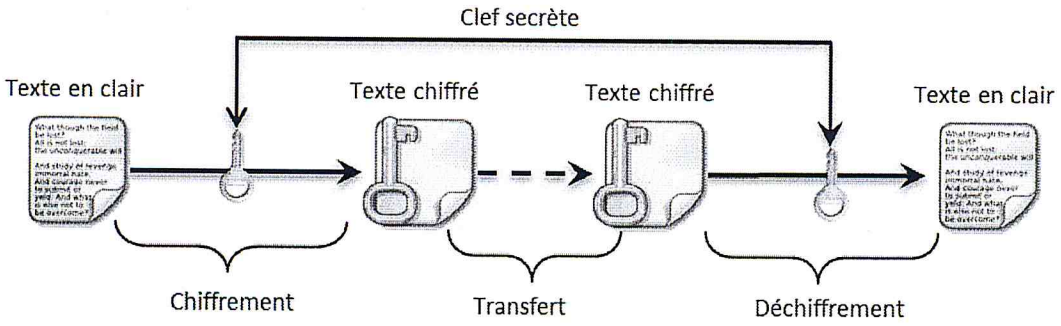


FIGURE 2.6 – Chiffrement à clef secrète.

les chiffrements par blocs (block cipher) et les chiffrements en continu (stream cipher). Les algorithmes de chiffrement par blocs chiffrent les blocs de données de taille fixe alors que les algorithmes de chiffrement en continu chiffrent une suite d'octets de longueur arbitraire.

Parmi les algorithmes à clef secrète, les plus courants sont DES (Data Encryption Standard), RC2 (Rivest Cipher-2), RC4 (Rivest Cipher-4) et 3DES-EDE (DES appliqué trois fois en mode Encrypt-Decrypt-Encrypt). Ces algorithmes sont basés sur des permutations et des opérations de "ou exclusif". Leur fiabilité est liée à la longueur des clefs secrètes qu'ils utilisent. Seul RC4 est un algorithme de chiffrement en continu. Les autres utilisent des algorithmes de chaînage de blocs tels que : CBC (Cipher Bloc Chaining) ou ECB (Electronic Code Book) pour coder des messages de taille arbitraire. L'algorithme de chiffrement le plus rapide est RC4 et le plus lent 3DES-EDE. En termes de sécurité, l'algorithme de chiffrement le plus performant est 3DES-EDE avec une clef de 168 bits. Actuellement, les algorithmes RC4 ou RC2 avec une clef de 128 bits ou DES avec une clef de 56 bits offrent également une bonne sécurité [31].

b. Le chiffrement à clef publique

Le chiffrement à clef publique repose sur un algorithme qui peut être connu de tous et une paire de clefs, une clef publique $Kpub$ du destinataire et une clef privée $Ksec$ du destinataire. La clef privée est connue par une seule personne (le destinataire) et la clef publique peut être divulguée. Le message M est chiffré avec la clef publique et

peut uniquement être déchiffré avec la clef privée (voir figure 2.7) [31] ; autrement dit le chiffrement (resp. déchiffrement) est une transformation mathématique simple pour obtenir le chiffré C (resp. message M) [32] :

$$C = Chiffrer(M, Kpub_{(du\ destinataire)})$$

$$M = Chiffrer^{-1}(C, Ksec_{(du\ destinataire)})$$

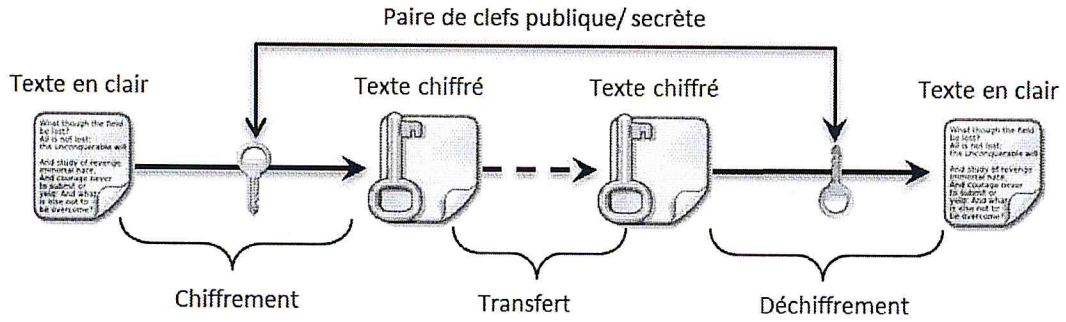


FIGURE 2.7 – Chiffrement à clef publique.

Parmi les algorithmes de chiffrement à clef publique, le plus utilisé est **RSA** (**R**ivest, **S**hamir et **A**dleman) [1], mais il en existe d'autres tels qu'Elgamal. Ces algorithmes de chiffrement sont basés des calculs d'exposants. Actuellement, pour disposer d'une bonne sécurité, il est conseillé d'utiliser des clefs de chiffrement de taille supérieure ou égale à 1024 bits.

Signature électronique [29]

Outre sa fonction de chiffrement, *Chiffrer* est aussi utilisable de façon très simple pour signer de façon sûre et non répudiable un document. Il est important qu'une signature ne puisse pas être répudiée, c'est-à-dire que le signataire ne puisse pas prétendre qu'il n'est pas l'auteur du document signé, que cette signature n'est pas son œuvre. La qualité de résistance à la répudiation doit résider dans une preuve de la signature, détenue par le destinataire du document signé, vérifiable par un tiers, et inaltérable par le signataire. Pour conférer cette qualité prisée à sa signature, il suffit que le signataire la chiffre avec sa clé privée : le destinataire la déchiffrera avec la clé publique du signataire, et si le déchiffrement réussit ce sera la preuve que la signature est authentique, en d'autres termes une authentification sûre. Une autre méthode consiste à signer un "résumé numérique" du message. Ce résumé, appelé condensat, est produit par un algorithme de condensation, telMD5 créé par Ronald Rivest, ou SHA (Secure Hash Standard). Le principe d'une fonction de condensation (parfois appelée ***hachage***) est le suivant : soient M et M' deux messages, et H la fonction :

1. si M $\neq$ M', la probabilité que H(M) = H(M') est très voisine de 0 ;
2. quel que soit M, il est difficile de trouver un M' $\neq$ M tel que H(M') = H(M).

1. Les premières lettres des noms de ses créateurs (RSA)

Cette propriété d'un algorithme de condensation que l'on ne puisse pas trouver facilement deux textes différents qui donnent le même résumé est appelée **la résistance aux collisions**, elle est essentielle.
Un auteur peut par conséquent signer en calculant le condensat de son message, en le chiffrant grâce à sa clé privée puis en le diffusant. Tout détenteur de sa clé publique et du message sera en mesure de vérifier la signature. Outre une signature non répudiable, ce procédé garantit en pratique l'intégrité du message.

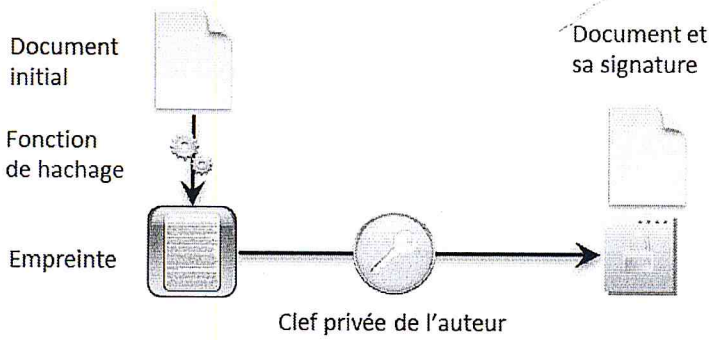


FIGURE 2.8 – Signature d'un document.

La signature est effectuée en deux temps (voir figure 2.8) :

- Le logiciel (de courrier électronique, par exemple) de l'ordinateur d'auteur calcule un résumé du message selon un des algorithmes convenus et publics, tels que MD5 ou SHA-1, qui répondent aux conditions suivantes :
  - connaissant le résumé R d'un message M, il est très difficile de fabriquer un message M' différent auquel corresponde le même résumé R ;
  - la probabilité que deux messages donnent le même résumé est très faible.
- L'auteur chiffre le résumé avec sa clé privée.

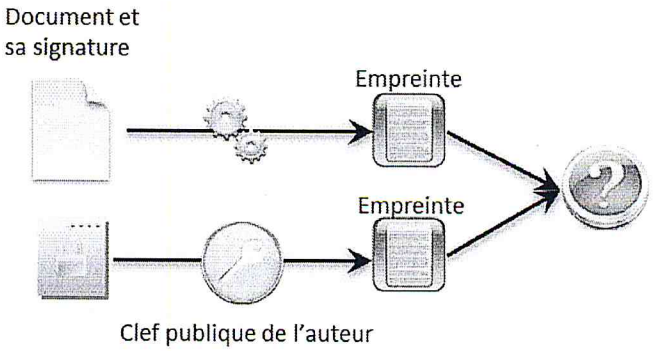


FIGURE 2.9 – Réception du document avec sa signature.

A la réception du message, le lecteur fait deux choses (ou plutôt c'est son logiciel de courrier électronique qui le fait) (voir figure 2.9) :

- il déchiffre le résumé chiffré avec la clé publique de l'auteur ;
- il calcule le résumé du message par le même algorithme que l'auteur ;
- si les deux résumés sont égaux, le message a bien été signé par l'auteur, seul détenteur de sa clé privée.

Certificat électronique

Un certificat électronique est une assurance de sécurité sur l'identité électronique d'un individu ou d'un système. Les infrastructures PKI (Public Key Infrastructure) sont conçues pour mettre en œuvre l'architecture correspondante. Une PKI est une infrastructure composée d'un ensemble de systèmes, de procédures et de politiques, dont les fonctions sont les suivantes :

- enregistrer les entités désirant obtenir des certificats électroniques ;
- fabriquer des bi-clefs, c'est-à-dire des paires de clés privée et publique ;
- certifier des clés publiques afin de créer des certificats et de publier ces derniers sur des annuaires publics ;
- révocation de certificats et gestion de listes de révocation.

L'obtention d'un certificat numérique doit suivre des procédures et politiques très strictes, comme l'illustre la figure 2.10. Les chiffres indiqués sur les flèches de la figure indiquent le séquencement des étapes pour délivrer un certificat électronique. De manière très sim-

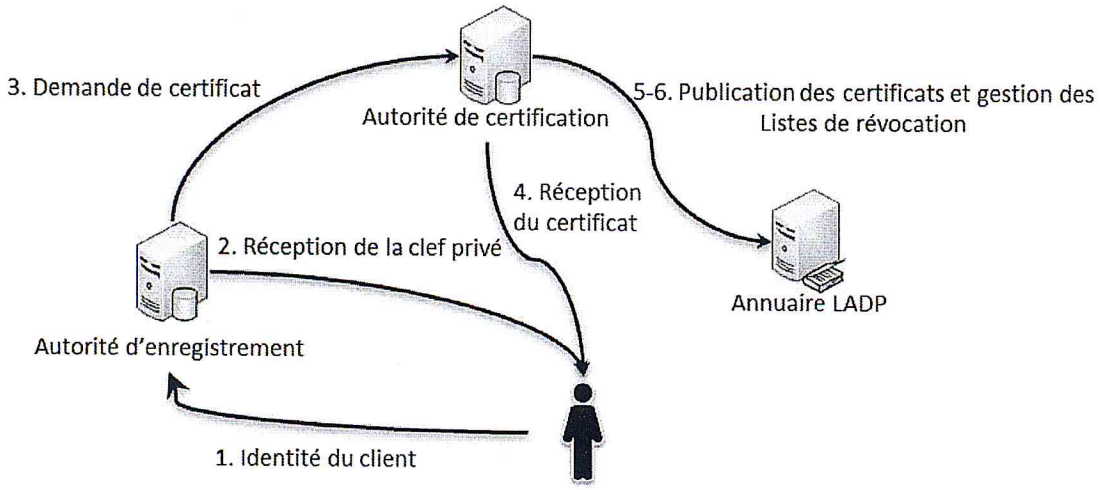


FIGURE 2.10 – Les échanges dans une infrastructure à clé publique.

plifiée, l'utilisateur désirant obtenir un certificat électronique fait une demande auprès de l'autorité d'enregistrement (AE). Après validation de l'identité du demandeur, l'AE génère un couple de clés (publique, privée), envoie la clé privée suivant des procédures sécurisées à l'utilisateur (chemin de confiance) et certifie la clé publique par l'autorité de certification en apposant sa signature électronique sur le certificat. Le certificat est alors installé sur un annuaire public accessible à tous.

Un certificat électronique, ou passeport numérique, contient toutes les informations relatives à l'identité d'une personne :

- numéro de version associé au certificat, par exemple X.509 v3 ;
- numéro de série fourni par l'autorité de certification ayant délivré le certificat ;
- algorithme utilisé pour la signature du certificat ;
- nom de l'autorité ayant délivré le certificat ;
- date de validité du certificat (dates de création et d'expiration) ;
- nom de la personne de destination du certificat ;
- clé publique de la personne certifiée.

D'autres informations concernant des attributs spécifiques associés au certificat dépendent de la version du certificat, etc. A partir de ces informations, dont l'autorité de certification vérifie préalablement la validité, cette même autorité de certification génère une signature de certification en créant dans un premier temps une empreinte de ces informations grâce à un algorithme de hachage et en chiffrant cette empreinte par un algorithme de chiffrement asymétrique grâce à la clé privée de l'autorité de certification. Pour vérifier une signature d'une autorité de certification, il suffit de prendre l'ensemble des informations du certificat, excepté la signature, afin de créer une empreinte puis de déchiffrer la signature de l'autorité de certification grâce à la clé publique de cette même autorité afin de retrouver l'empreinte initiale certifiée. La dernière étape consiste à comparer les deux empreintes. Si elles correspondent, le certificat est valide, sinon il ne peut être considéré comme de confiance.
Un certificat est disponible dans le domaine public. En revanche, la clé privée associée au certificat est précieusement protégée sur un support physique sécurisé.
Les certificats révoqués sont regroupés dans des listes de révocation, ou CRL (Certificate Revocation List). Les CRL sont des structures de données signées, dont le format est défini par le protocole X.509 v2. Ces structures doivent être mise à jour le plus fréquemment possible pour qu'un client ne puisse pas récupérer un certificat révoqué mais non encore ajouté à la CRL.

Synthèse

Le Tableau suivant récapitule pour chacune des techniques de base présentées précédemment les critères qui sont assurés.

Techniques	Concepts/Critères
Chiffrement/Déchiffrement	- Confidentialité
Signature électronique	- Intégrité
Certificat électronique	- Authentification - Intégrité - Confidentialité - Non-répudiation

TABLE 2.1 – Synthèse des technologies de sécurité.

2.4 Technologies de sécurité pour les services Web

Dans cette partie, nous détaillons les technologies qui mettent en œuvre les techniques de sécurité précédemment présentées en les adaptant aux nouvelles problématiques posées par la distribution des applications et l'hétérogénéité des technologies d'implantation. Nous détaillons en premier les technologies de niveau transport, ensuite nous présentons quelques standards et techniques propre aux services web.

2.4.1 Sécurité de niveau transport

La sécurité de niveau transport s'appuie sur les mécanismes de la couche de transport sous -jacente pour offrir les fonctions de sécurité. Avec ce modèle, la sécurité n'est garantie que par la plate- forme et le transport.

SSL (Secure Sockets Layer)

Conçu et développé par Netscape, le protocole SSL a été développé au-dessus de la couche TCP afin d'offrir aux navigateurs Internet la possibilité d'établir des sessions authentifiées et chiffrées. La première version de SSL date de 1994 [32].
La figure 2.11 illustre le principe de fonctionnement du protocole SSL. SSL s'insère entre les couches applicatives et la couche TCP afin d'offrir ses services de sécurité. Il est possible de ne pas utiliser le protocole SSL. Les couches applicatives se connectent alors directement à la couche TCP [32].

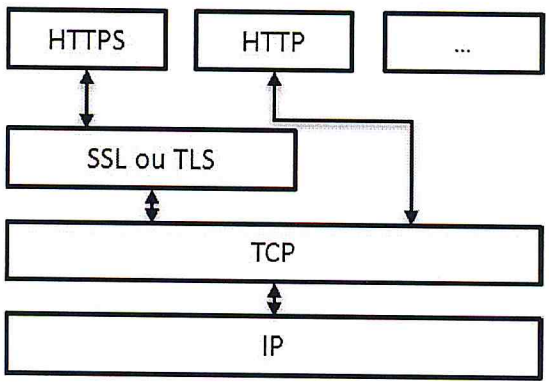


FIGURE 2.11 – Représentation en couches du protocole SSL.

Les services de sécurité offerts par la version 3 de SSL sont les suivants [32] :

Authentification L'authentification s'appuie sur des certificats électroniques X.509 v3. La vérification du certificat du serveur est obligatoire, tandis que celle du client reste optionnelle. L'algorithme cryptographique à clé publique RSA est utilisé pour l'authentification et la signature numérique.

Confidentialité La confidentialité s'appuie sur des algorithmes cryptographiques à clés symétriques négociées lors de la phase d'établissement de la session. Les clés utilisées sont générées pour une session donnée. Les algorithmes de chiffrement peuvent être IDEA (International Data Encryption Algorithm), DES, 3DES, etc.

Intégrité L'intégrité des messages échangés s'appuie sur la fonction de hachage HMAC (Hash-based Message Authentication Code), qui nécessite pour le calcul du MAC (Message Authentication Code), une clé secrète partagée par la session pour le chiffrement des données et une fonction de hachage primaire (MD5 et SHA-x).

SSL est constitué de quatre sous-protocoles, comme illustré à la figure 2.12. Ces différents protocoles interagissent, mais chacun avec des fonctions précises [32] :

Handshake est le protocole d'établissement d'une connexion SSL. Il permet d'authentifier les parties client-serveur et de négocier les paramètres cryptographiques (choix de l'algorithme de chiffrement, de l'algorithme de calcul du code d'authentification MAC, des clés de chiffrement, etc.).

Record est un protocole d'enregistrement. Pour une même connexion SSL, il offre à la fois les services de confidentialité, par le biais de l'algorithme cryptographique à clé secrète retenu, et d'intégrité des messages échangés, par le biais du calcul du code d'authentification MAC pour chaque message échangé. Des fonctions de fragmentation et de compression des données sont en outre réalisées.

Alert est un protocole d'alerte. Il permet d'échanger des messages prédéfinis sur les états d'une connexion SSL, tels que la fermeture d'une connexion, notamment lorsqu'un certificat a été révoqué, qu'il a expiré, qu'il est vicié, etc.

CCS (Change Cipher Security) est un protocole de modification des spécifications de chiffrement. Il permet de modifier les paramètres de chiffrement d'une connexion SSL.

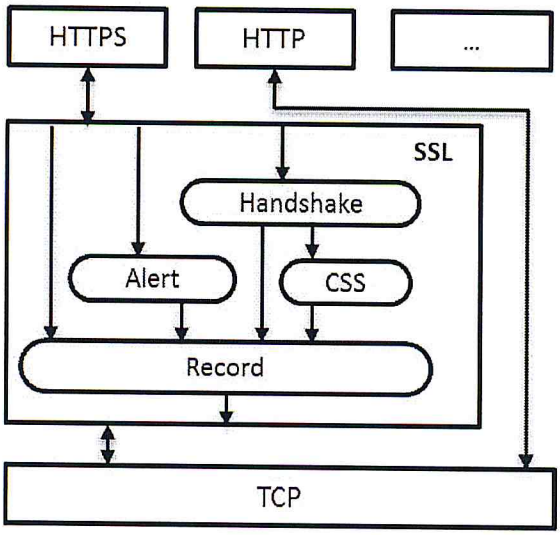


FIGURE 2.12 – Représentation détaillée des sous-protocoles de SSL.

Problématique

La gestion de la sécurité au niveau transport est certainement nécessaire aux architectures de services Web, mais elle n'est pas suffisante pour mettre en œuvre des architectures orientées services complexes sur Internet. Nous nous trouvons dans la situation illustrée en figure 2.13, dans laquelle les messages entre Alice et Bob sont chiffrés et leur intégrité ainsi que l'authentification des ports sont garanties par des mécanismes de signature [41]. La situation change si l'architecture demande qu'un ou plusieurs intermédiaires puissent se glisser entre Alice et Bob. Or, ces intermédiaires, ainsi que des tierces parties, sont nécessaires pour mettre en œuvre des architectures plus élaborées.
Dans une chaîne d'acheminement, un message est émis par l'expéditeur (Alice) à destination de Bob, et ce message transite par un intermédiaire (Carol). Cet intermédiaire peut évidemment lire le message, doit éventuellement en consommer une partie et/ou en produire une autre partie à l'intention de Bob. Dans le cadre d'une chaîne d'acheminement, la mise en œuvre d'un contexte de sécurité au niveau transport ne permet de gérer la sécurité que selon un mode ***point à point*** (voir figure 2.13) [41]. Un contexte de sécurité est établi entre Alice et l'intermédiaire Carol. Un autre contexte est établi entre Carol et

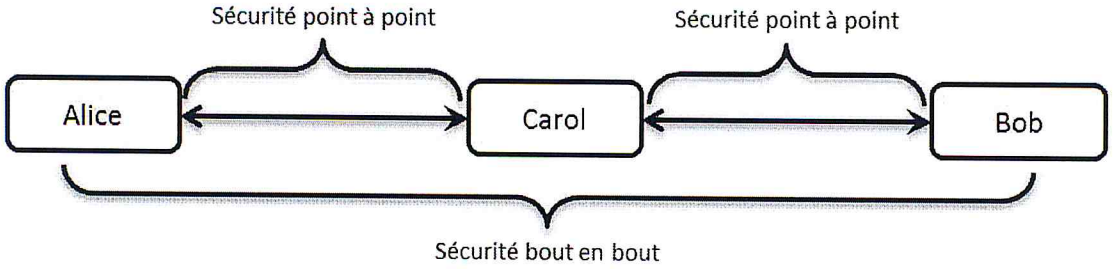


FIGURE 2.13 – Contextes de la sécurité.

Bob. Cette architecture ne résiste pas à une usurpation de l'identité de Carol de la part d'un intermédiaire indélicat, ni à une attaque réussie, qui se termine par la prise en main de Carol de la part d'un logiciel malveillant. La mise de ce type d'architectures réparties, nécessite l'établissement d'un contexte de sécurité de ***bout en bout***, c'est-à-dire du producteur/expéditeur au destinataire/consommateur, aux niveaux échange et application (voir figure 2.13) [41].

2.4.2 Autres technologies

Le problème présenté dans la section précédente a pris de l'importance dès lors que les services Web se sont multipliés [27] et après la naissance du B2B (Business To Business) [10]. Une transaction de service Web passe souvent par de nombreuses mains (intermédiaires), dont chacune a besoin d'accéder à certaines parties de la transaction mais pas à d'autres. Deux spécifications font aujourd'hui référence sur le plan de la sécurité au sein d'une architecture orientée service Web : WS-Security et WS-SecurityPolicy [27]. WS-Security permet de définir la manière de décrire, au sein d'un message SOAP, les droits utilisateur correspondants aux systèmes en présence. Pour ce faire un message SOAP peut être complété par différentes assertions en utilisant par exemple les langages XML Signature, XML Encryption, SAML (Security Assertion Markup Language) ou encore XKMS (XML Key Management Specification) [27].

2.5 Conclusion

Dans ce chapitre, nous avons commencé par mettre en évidence les failles de sécurité qui existent dans l'architecture orientée services. Les messages échangés sont une cible particulièrement intéressante pour des personnes malveillantes. Les messages peuvent être écoutés, divulgués et/ou altérés. Des mécanismes de sécurité sont donc nécessaires pour que l'approche à services puisse être couramment utilisée dans les entreprises pour intégrer des applications.
Nous avons identifié, dans ce chapitre, un ensemble de concepts, de techniques et de technologies qui permettent de répondre aux menaces détectées dans les architectures orientée services. Les principaux concepts sont l'authentification, la confidentialité et l'intégrité qui permettent de protéger les applications distribuées, telles que celles réalisées suivant l'architecture orientée services. Pour chacun de ces concepts, il existe des techniques de base qui leur sont associés tels que le chiffrement, la signature électronique et le certificat électronique. Nous avons étudié aussi le protocole SSL, qui permet de mettre en œuvre ces

techniques.
Avec le développement d'Internet et des applications pour le Web, les technologies de sécurité au niveau transport ne permettent pas de gérer des situations complexes impliquant un ou plusieurs intermédiaires. Afin de palier à ce problème, beaucoup de standards et de recommandations ont été établis par exemple : XML Signature, XML Encryption, SAML ou encore XKMS.
Pour finir, le protocole SSL permet un haut niveau de sécurité dans le cas où le système est simple et basique (il n'existe pas des services intermédiaires).

Chapitre 3

Présentation de l'application E-APC

3.1 Introduction

Dans le cadre de ce projet, nous exploitons les travaux réalisés par l'équipe **RSI** (**R**éseaux et **S**ystèmes d'**I**nformation) de la division **ASM** (**A**rchitecture des **S**ystèmes et **M**ultimédias) du **CDTA** (**C**entre de **D**éveloppement des **T**echnologies **A**vancées). Leur objectif est de réaliser un système qui permettrait de mettre sur internet les divers services et activités que nous trouvons dans une APC et qui sont fortement sollicités par les citoyens d'une commune tels que le service d'état civil, le service de vote et les divers aspects liés aux délibérations de l'APC.

Les résultats de ces travaux ont menés à la réalisation de deux systèmes :
- E-APC v 1.0 ;
- E-APC v 2.0.

Dans le cadre de ce projet, on concentrera notre étude sur la version la plus récente (E-APC 2.0).

3.2 Objectifs de l'application E-APC 2.0

Les objectifs globaux suivants ont été définis :
- Réduire le taux d'erreurs dans le processus de saisie d'informations. L'objectif est un système à zéro erreur au niveau des données relatives aux citoyens ;
- Protection des Informations personnelles persistantes ;
- Le suivi des responsabilités : Toute action importante pouvant avoir un impact sur les transformations des informations, notamment données personnelles du citoyen, devra être journalisée et exploitée ;
- Implémentation d'un mécanisme de vérification/validation de l'information ;
- Permettre le retrait en ligne des divers documents (acte de naissance, acte de mariage, etc.) ;
- L'authentification des documents tirés par l'utilisateur est faite par un numéro secret.

3.3 Analyse des besoins

3.3.1 Les acteurs de l'application E-APC 2.0

Ce système possède deux types d'acteurs [37] :

Utilisateur

C'est une personne qui possède une session de travail dans le système. L'accès à sa session nécessite toujours une authentification. Un utilisateur a la possibilité d'effectuer différente tâches selon son profil qui peut être : Administrateur du système, un officier d'état civil principal, un officier d'état civil interne, un officier d'état civil externe, un officier d'état civil validateur, un citoyen, un officier de justice, un employé d'hôpital.

Anonyme

C'est une personne qui accède au système sans aucune authentification.

Chaque acteur possède des tâches bien définies. Afin d'illustrer les fonctionnalités qu'offre l'application E-APC 2.0, nous présentons le diagramme des cas d'utilisation global.

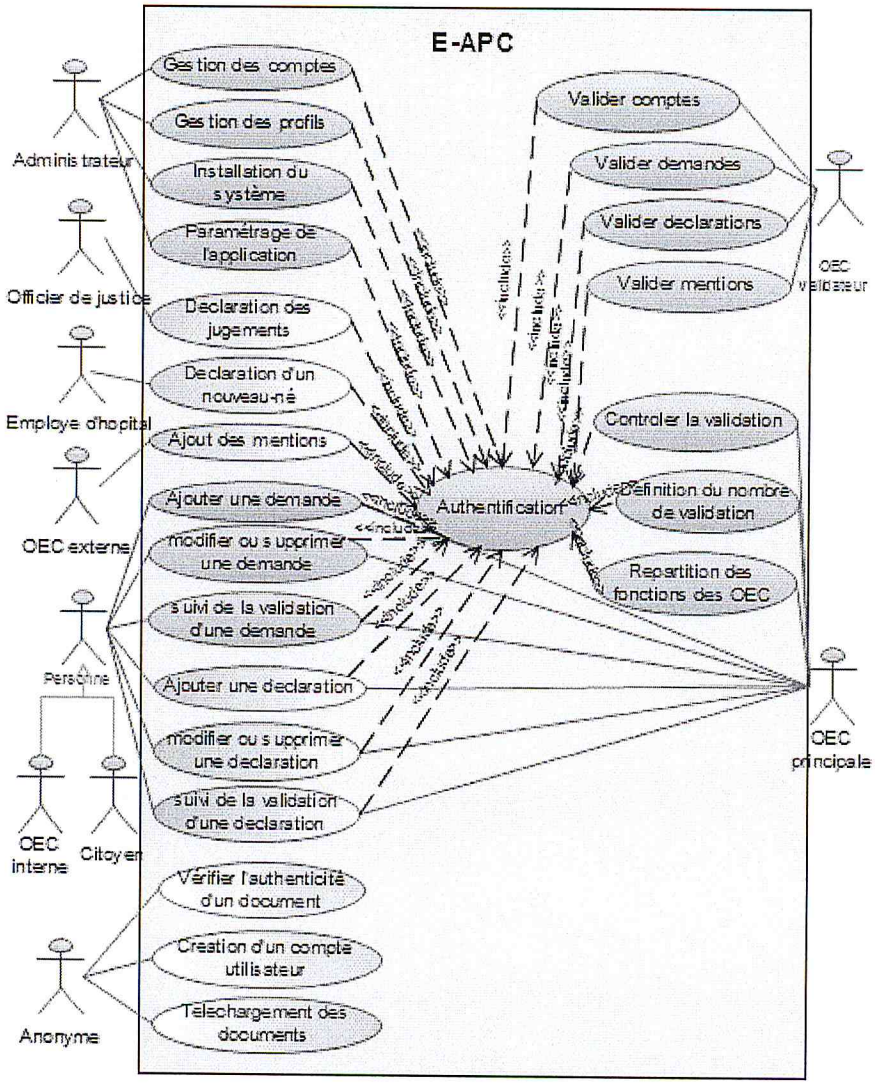


FIGURE 3.1 – Diagramme des cas d'utilisation global [37]

Dans ce qui suit, nous définissons les tâches que peut réaliser chaque acteur.

Administrateur du système : L'administrateur effectue les tâches suivantes

- Gestion des profils
  - Créer un profil ;
  - Modifier et supprimer un profil.
- Gestion des comptes
  - Créer un compte utilisateur ;
  - Activer un compte ;
  - Modifier et supprimer un compte ;
  - Définir les droits d'accès.
- Paramétrage du système : l'administrateur peut définir des paramètres du système comme :

- Préciser la méthode d'activation d'un compte (soit le compte sera activé automatiquement ou bien il sera activé par l'administrateur du système) ;
- Modifier l'année de découpage administrative.

Officier d'état civil Principal : L'officier d'état civil principal (OECP) peut organiser les fonctions dans la commune, pour cela il a la possibilité de réaliser :

- La répartition des tâches entre les OEC ;
- La création d'une nouvelle tâche ;
- Gestion des demandes ;
- Gestion des déclarations.

Officier d'état civil interne : L'officier d'état civil effectue les tâches suivantes :

- Gestion des demandes ;
- Gestion des déclarations.

Officier d'état civil externe : L'officier d'état civil effectue seulement la gestion des mentions.

Officier d'état civil validateur : Chaque information enregistrée dans le système doit être validée par l'officier d'état civil validateur, le système gère les déclarations, les demandes et les mentions.

Officier de la justice : Cet acteur a le droit de déclarer des jugements de mariage et de divorce.

Employé d'hôpital : L'employé d'hôpital réalise les déclarations d'un nouveau-né et de décès.

Citoyen : Le citoyen effectue des déclarations et des demandes (naissance, mariage, divorce et décès).

3.4 Conception de l'application E-APC 2.0

Dans cette partie, nous détaillons le diagramme des cas d'utilisation global par des diagrammes de séquence. A la fin de cette partie, on va présenter le diagramme des classes de l'application E-APC 2.0.

3.4.1 Description des cas d'utilisation

Cas d'utilisation «ajouter une déclaration»

Cas d'utilisation	Ajouter une déclaration
Acteur	OEC principal, OEC interne, citoyen
Résumé métier	Lorsqu'il aura un évènement de naissance, mariage, décès ou divorce.
Pré-condition	Authentification de l'utilisateur.
Post-condition	Évènement déclaré.

Diagramme de séquence Le diagramme suivant décrit le cas d'utilisation «ajouter une déclaration».

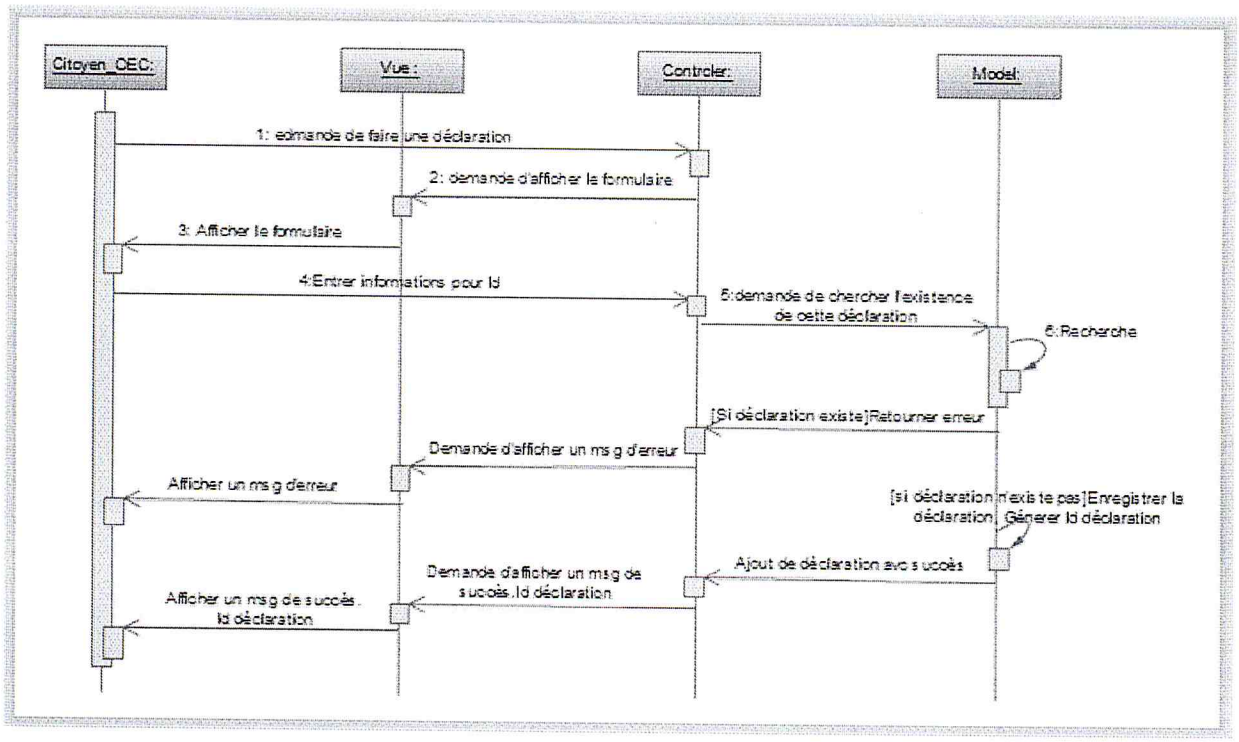


FIGURE 3.2 – Diagramme de séquence «ajouter une déclaration» [37]

Cas d'utilisation «valider une déclaration»

Cas d'utilisation	Valider une déclaration
Acteur	Officier d'état civil validateur
Résumé métier	Après avoir fait une déclaration, celle-ci doit être validée avant l'enregistrement dans la base des données.
Pré-condition	Authentification de l'officier d'état civil.
Post-condition	Déclaration validée.

Diagramme de séquence Le diagramme suivant détaille le cas d'utilisation «valider une déclaration».

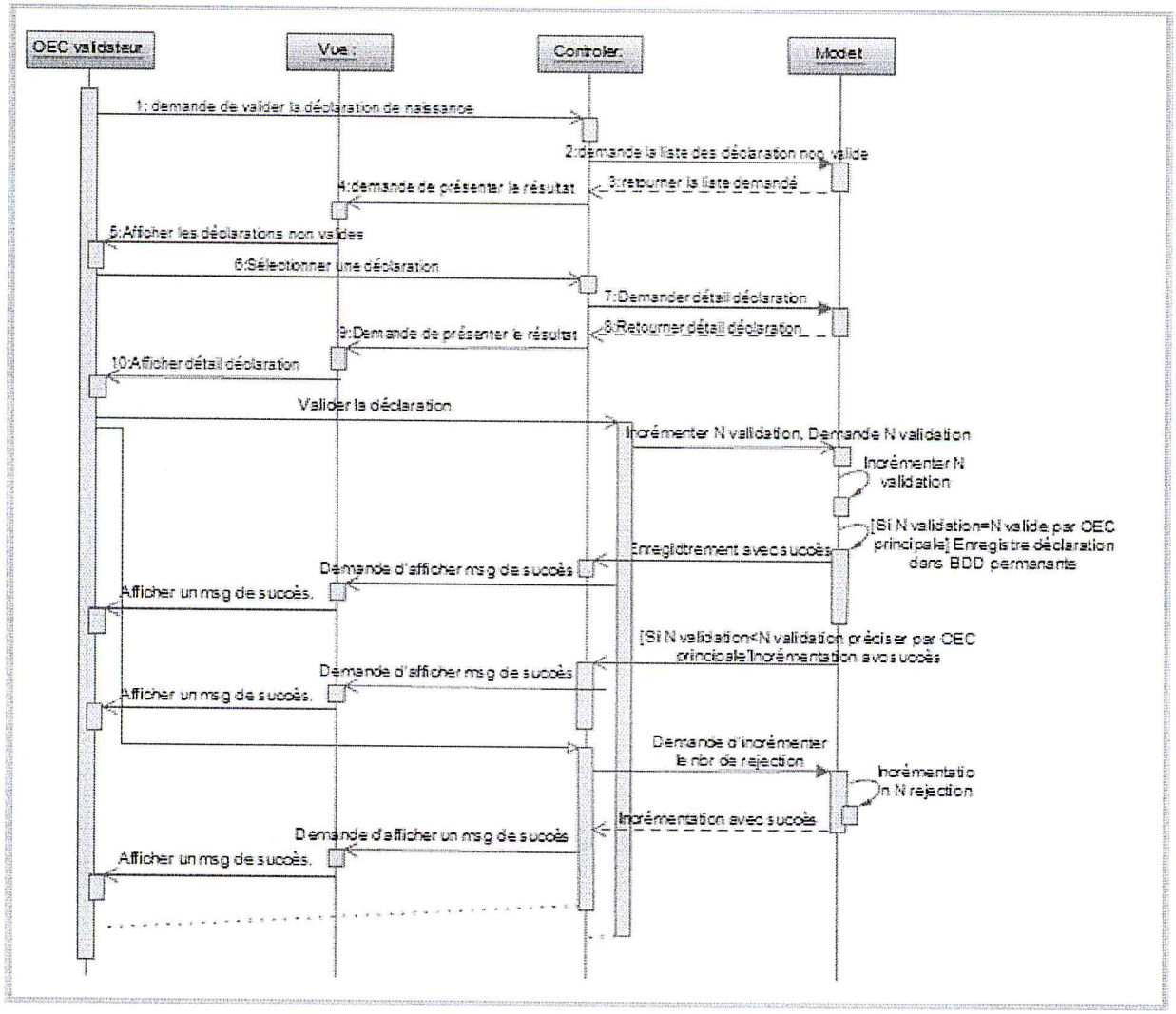


FIGURE 3.3 – Diagramme de séquence «valider une déclaration» [37]

3.5 Diagramme des classes

3.5.1 Dictionnaire des données

Le tableau suivant fournit pour chaque classe ses attributs, leurs types et les opérations qu'offre cette classe.

Classe	Attributs	Code : type	Valeur	Les méthodes
Personne	id de la personne Nom en arabe Prénom en arabe Sexe Profession Adresse Nom en français Prénom en français	#id_p : Chaine +nom_p_Ar : Chaire +prenom_p_Ar :Chaine +sexe_p : Chaine +prof_p : Chaine +address_p : Chaine +nom_p_Fr : Chaine +prenom_p_Fr : Chaine	Non nul Non nul Non nul M / F / / Non nul Non nul	Ajouter_p() Modifier_p() Supprimer_p() Rechercher_p() Consulter_p()
Utilisateur	Nom d'utilisateur Mot de passe E_mail Etat de compte	#nom_u : Chaine +mot_pass_u : Chaine +email_u : Chaine +etat_u : Chaine	Non nul Non nul / Activé/ désactivé	Ajouter_u() Modifier_u() Supprimer_u() Rechercher_u() Consulter_u() Activer_u() Désactiver()
Profil	Id de profil Nom de profil	#id_pr : Entier +non_pr : Chaine	Non nul Non nul	Ajouter_pr() Modifier_pr() Supprimer_pr() Rechercher_pr() Consulter_pr()
Action	Nom d'action	#nom_ac : Chaine	Non nul	Rechercher_ac() Consulter_ac()
SousAction	Nom sous action	#nom_sou_ac :Chaine	Non nul	/
Commune	Id commune Nom commune	#id_com :Chaine +nom_com :Chaine	Non nul Non nul	/
Daïra	Id daïra Nom de daïra	#id_daira :Chaine +nom_daira :Chaine	Non nul Non nul	/
Wilaya	Id wilaya Nom de wilaya	#id_wilaya : Chaine +nom_wilaya :Chaine	Non nul Non nul	/
Naissance	Id naissance Date de naissance Heure de naissance	# id_naiss : Chaine +date_nais : Chaine +Heure_naiss :Chaine	Non nul Non nul Non nul	/
Déclaration naissance	Id déclaration Numéro secret Date déclaration	#id_dec_naiss :Chaine +n°_sec_naiss :Chaine +date_dec_naiss :Date	Non nul Non nul Non nul	Ajouter_dec() Modifie_dec() Supprimer_dec() Recherche_dec() Consulter_dec() Valider_dec()
Demande extrait naissance	Id demande Numéro secret Date de demande	#id_dem_naiss :Chaine +n°_sec_naiss :Chaine +date_dem_naiss :Date	Non nul Non nul Non nul	Ajouter_dem() Modifie_dem() Supprimer_dem() Recherche_dem() Consulter_dem() Valider_dem()
Mariage	Id de mariage Date de mariage Heure de mariage	#id_mar : Chaine +date_mar : Chaine +Heure_mar :Chaine	Non nul Non nul Non nul	/
Déclaration mariage	Id déclaration Numéro secret Date déclaration	#id_dec_mar :Chaine +n°_sec_mar :Chaine +date_dec_mar :Date	Non nul Non nul Non nul	Ajouter_dec() Modifie_dec() Supprimer_dec() Recherche_dec() Consulter_dec() Valider_dec()

Demande acte mariage	Id demande Numéro secret Date de demande	#id_dem_mar :Chaine +n°_sec_mar :Chaine +date_dem_mar :Date	Non nul Non nul Non nul	Ajouter_dem() Modifie_dem() Supprimer_dem() Recherche_dem() Consulter_dem() Valider_dem()
Mention mariage	Id mention Date établissement Type mention	#id_men_mar :Chaine +date_men_mar :Date +type_men	Non nul Non nul Epoux/ Epouse	Ajouter_men() Valider_men() Recherche_men() Consulter_men()
Divorce	Id divorce Num jugement Tribunal jugement Date de jugement Heure de jugement	#id_div : Chaine +n_jug : Entier +tribunal_jug :Chaine +date_jug : Date +Heure_mar :Chaine	Non nul Non nul Non nul Non nul Non nul	/
Déclaration divorce	Id déclaration Numéro secret Date déclaration	#id_dec_div :Chaine +n°_sec_div :Chaine +date_dec_div :Date	Non nul Non nul Non nul	Ajouter_dec() Modifie_dec() Supprimer_dec() Recherche_dec() Consulter_dec() Valider_dec()
Demande acte divorce	Id demande Numéro secret Date de demande	#id_dem_div :Chaine +n°_sec_div :Chaine +date_dem_div :Date	Non nul Non nul Non nul	Ajouter_dem() Modifie_dem() Supprimer_dem() Recherche_dem() Consulter_dem() Valider_dem()
Mention divorce	Id mention Date établissement Type mention	#id_men_div :Chaine +date_men_div :Date +type_men	Non nul Non nul Epoux/ Epouse	Ajouter_men() Valider_men() Recherche_men() Consulter_men()
Décès	Id décès Date décès Heure décès	#id_dec : Chaine +date_dec : Date +Heure_dec : Chaine	Non nul Non nul Non nul	/
Déclaration décès	Id déclaration Numéro secret Date déclaration	#id_dec_dec :Chaine +n°_sec_dec :Chaine +date_dec_dec :Date	Non nul Non nul Non nul	Ajouter_dec() Modifie_dec() Supprimer_dec() Recherche_dec() Consulter_dec() Valider_dec()
Demande acte décès	Id demande Numéro secret Date de demande	#id_dem_dec :Chaine +n°_sec_dec :Chaine +date_dem_dec :Date	Non nul Non nul Non nul	Ajouter_dem() Modifie_dem() Supprimer_dem() Recherche_dem() Consulter_dem() Valider_dem()
Mention décès	Id mention Date établissement Type mention	#id_men_dec :Chaine +date_men_dec :Date +type_men	Non nul Non nul Epoux/ Epouse	Ajouter_men() Valider_men() Recherche_men() Consulter_men()

3.5.2 Codification

* wilaya
  |__ Numéro wilaya _|
  |__2 Cases __|

* Daïra
  |_ Numéro wilaya __|_ Numéro daïra _|
  |_ 2 Cases _|_ 3Cases _|
* Commune
  |_ Numéro wilaya _|_ Numéro daïra _|_ Numéro commune _|
  |_ 2 Cases _|_ 3 Cases _|_4 Cases _|
* Naissance, mariage, décès ou divorce
  |_ Année découpage administrative|_ Num commune _|_ Année _|_N ° séq _|
  |_4 Cases _|___6 Cases __4 Cases __|_ 5 Cases _|
* Déclaration de naissance
  |_ Année de déclaration _|_ 01_|_01__|__ N ° séquentiel _|
  |_ 4Cases __|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Déclaration de mariage
  |_ Année de déclaration _|_ 01_|_02_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Déclaration de divorce
  |_ Année de déclaration _|_ 01_|_03_|_ N ° séquentiel _|
  |_ 4Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Déclaration de décès
  |_ Année de déclaration _|_ 01_|_04_|_ N ° séquentiel _|
  |_ 4Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Demande extrait naissance
  |_ Année de demande _|_ 02_|_01_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Demande acte mariage
  |_ Année de demande _|_ 02_|_02_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Demande acte divorce
  |_ Année de demande _|_ 02_|_03_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Demande acte décès
  |_ Année de demande _|_ 02_|_04_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Mention de mariage
  |_ Année de mention _|_ 03_|_01_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Mention de divorce
  |_ Année de mention _|_ 03_|_02_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Mention de décès
  |_ Année de mention _|_ 03_|_03_|_ N ° séquentiel _|
  |_ 4 Cases _|_2 Cases _|_ 2 Cases _|_5 Cases _|
* Numéro secret de déclaration ou Numéro de secret de demande
  |__ Numéro aléatoire générer par le système _|
  |__15 Cases _|
* Numéro de secret de l'acte de naissance, mariage, divorce ou décès
  |_ N aléatoire _|_ Jour _|_Mois _|_ Années _|_Heure _|_ Minute _|_Seconde _|
  |_10Cases _|_2 Cases _|_2Cases _|_4Cases _|_2Cases_|_2Cases_|_2Cases_|

3.5.3 Diagramme des classes

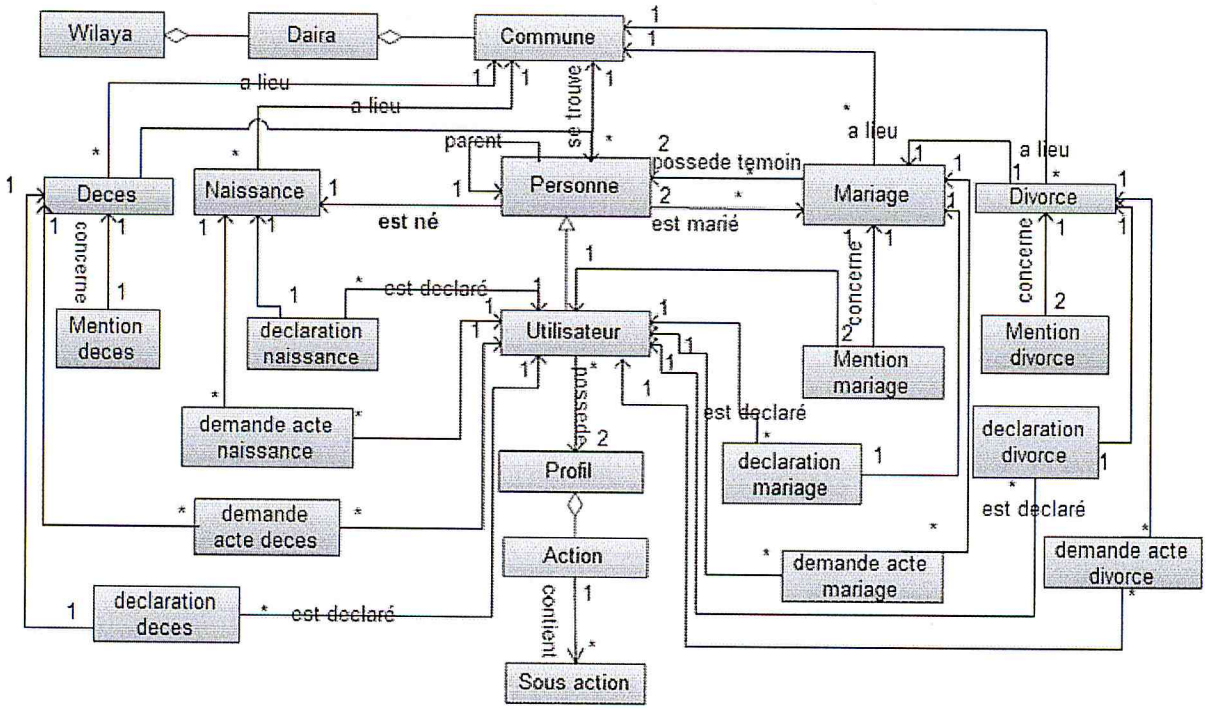


FIGURE 3.4 – Diagramme des classes de l'application E-APC 2.0 [37]

3.6 Analyse de l'application E-APC 2.0

Suite à l'analyse de ce système, nous avons identifié les règles métiers suivantes :

- Chaque élément (déclaration, demande ou mention) manipulé par cette application possède plusieurs états : Nouvel élément, élément validé, élément rejeté, élément en cours de validation ;
- Une demande (déclaration ou mention) en cours de validation ne peut pas être ni modifiée ni supprimée ;
- Un officier d'état civil validateur ne peut valider/rejeter une demande (déclaration ou mention) qu'une seule fois durant tout le processus de validation de cette information ;
- Un citoyen possède au plus un compte utilisateur ;
- Un utilisateur ne peut pas retirer des actes concernant d'autres personnes, sauf s'il est autorisé ;
- L'application manipule deux types d'informations valide et temporaire :
  - Une demande (déclaration ou mention) validée est sauvegardée dans la zone des informations valides ;
  - Une demande (déclaration ou mention) rejetée est supprimée du système ;

- Une nouvelle demande (déclaration ou mention) est enregistrée dans la zone temporaire du système ;
- Une information temporaire devient validée si et seulement s'elle est validée par N validateurs différents (N étant le nombre de validation nécessaire, sa valeur est configurée par l'officier d'état civil principal et dépend de la politique suivie par l'APC où l'application est installée).

3.7 Implémentation de l'application E-APC 2.0

- Le langage JAVA ;
- L'environnement de développement intégré Eclipse JEE ;
- Le Framework Struts 2 ;
- L'SGBD MySQL ;
- AJAX (Asynchronous JavaScript And Xml) pour gérer les appels asynchrones au serveur ;
- JasperReport comme outil de reporting ;
- Apache Tomcat comme conteneur des Servlets.

L'application E-APC 2.0 a été implémentée en utilisant : La figure suivante montre l'architecture globale de l'application E-APC 2.0 (architecture 3 tiers).

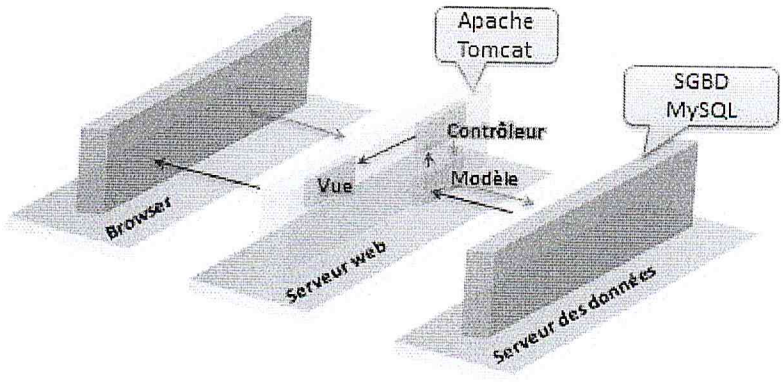


FIGURE 3.5 – Architecture globale de l'application E-APC 2.0

3.7.1 Présentation de l'application E-APC 2.0

Les figures suivantes montrent quelques fonctionnalités du système.

FIGURE 3.6 – Page d'accueil d'application E-APC 2.0

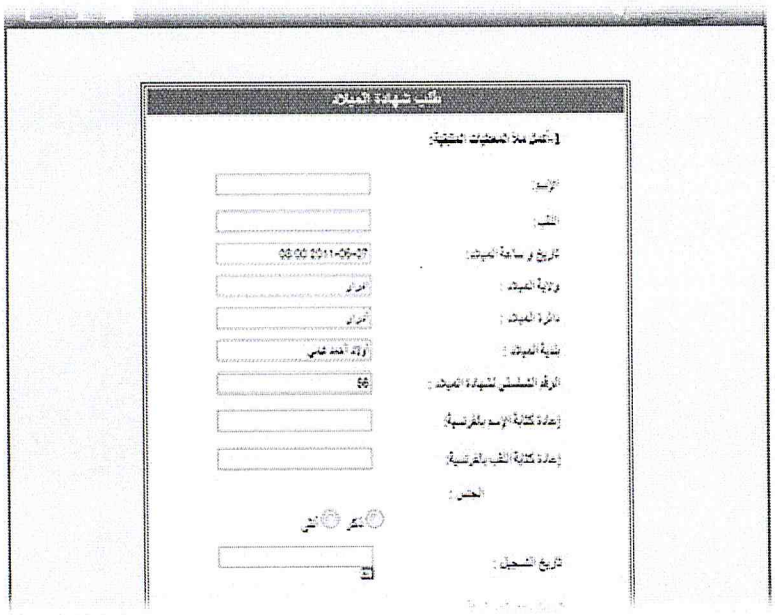


FIGURE 3.7 – Formulaire d'une demande d'un acte de naissance

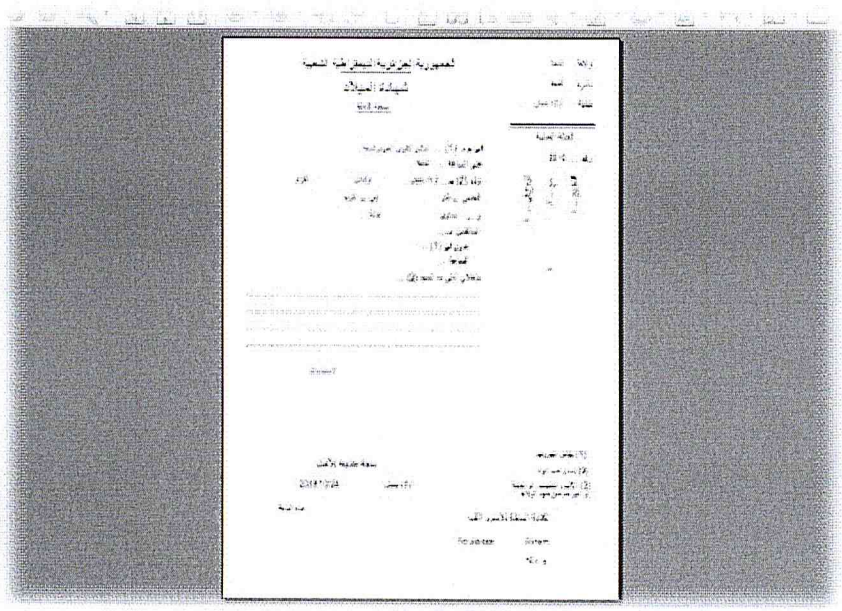


FIGURE 3.8 – Acte de naissance première page

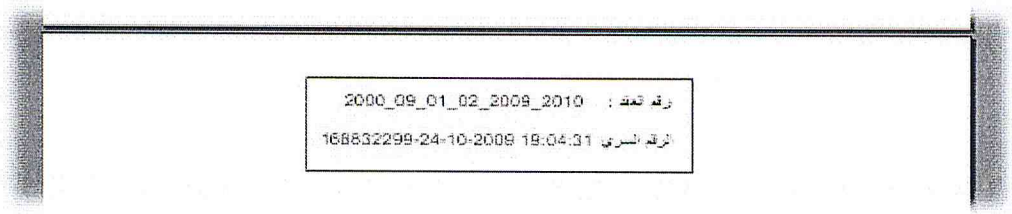


FIGURE 3.9 – Acte de naissance deuxième page

3.8 Conclusion

Dans ce chapitre, nous avons décrit en détail l'application E-APC 2.0. Cette étude nous a permis de bien comprendre les fonctionnalités qu'offre l'application, ce qui nous a mené par la suite à tracer les grandes lignes à suivre pour concevoir et implémenter notre système. Cette étape est très nécessaire vu que cette application constitue une brique importante de notre futur système de communication.

Chapitre 4

Conception du système

4.1 Introduction

Nous avons présenté dans les chapitres précédents une étude comparative entre les différentes techniques d'implémentation d'un système de communication, et nous avons choisi de réaliser notre système selon l'approche des architectures orientée services supporté par la technologie des services Web.
Ce chapitre traite la conception du système développé. Dans un premier temps, on se concentre sur la présentation de la démarche suivie pour la réalisation du projet, puis on entame la modélisation des services en utilisant le langage **SoaML** (Service-oriented architecture Modeling Language), un profil **UML** (Unified Modeling Language) dédié aux architectures orientées services. Finalement, on présentera une solution d'intégration de l'application E-APC 2.0 à notre système.

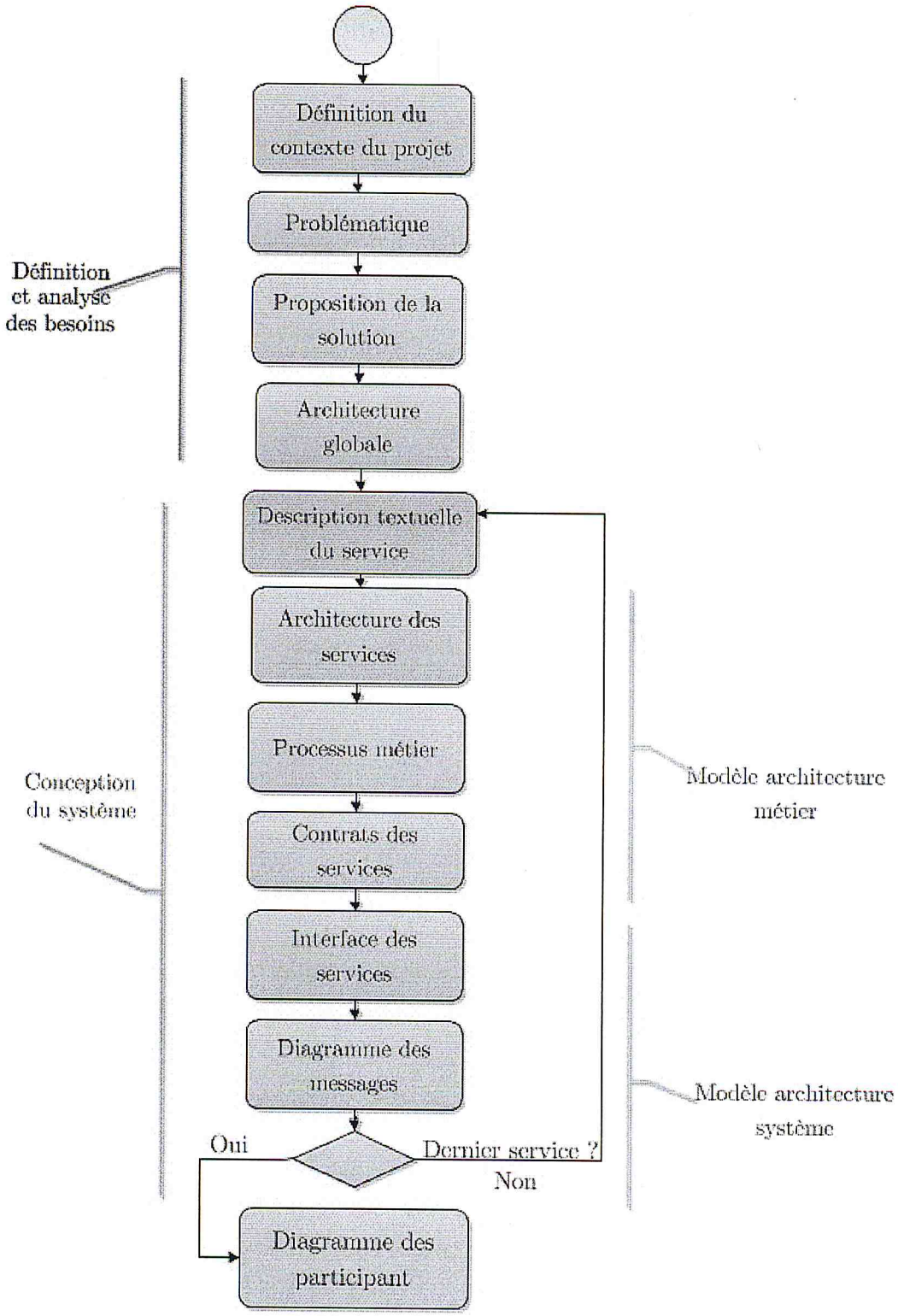
4.2 Démarche

Afin de réaliser convenablement ce projet, nous avons suivi la démarche présentée dans la figure 4.1. Certaines parties de ce diagramme sont inspirées de la méthode proposée dans le cadre du projet SHAPE (Service-oriented Heterogeneous Architecture and Platforms Engineering) [35].
Nous commençons cette section par une brève présentation du langage **SoaML**, ainsi la méthode de conception.

4.2.1 Présentation du langage SoaML

SoaML est une nouvelle spécification présentée par l'organisation OMG en 2009 [33]. Le but de SoaML est de fournir aux utilisateurs du langage UML (Unified modeling language) les moyens de modéliser une architecture orientée services comprenant donc des notions de **consommateurs** et de **fournisseurs** de services, ainsi que la notion de **contrats**.
SoaML propose plusieurs types de diagrammes :

- Diagramme d'architecture des services ;
- Diagramme de processus métier ;
- Diagramme de contrats des services ;
- Diagramme d'interface des services ;


Définition du contexte du projet
Problématique
Proposition de la solution
Architecture globale
Description textuelle du service
Architecture des services
Processus métier
Contrats des services
Interface des services
Diagramme des messages
Oui
Dernier service ?
Non
Diagramme des participant
Définition et analyse des besoins
Conception du système
Modèle architecture métier
Modèle architecture système

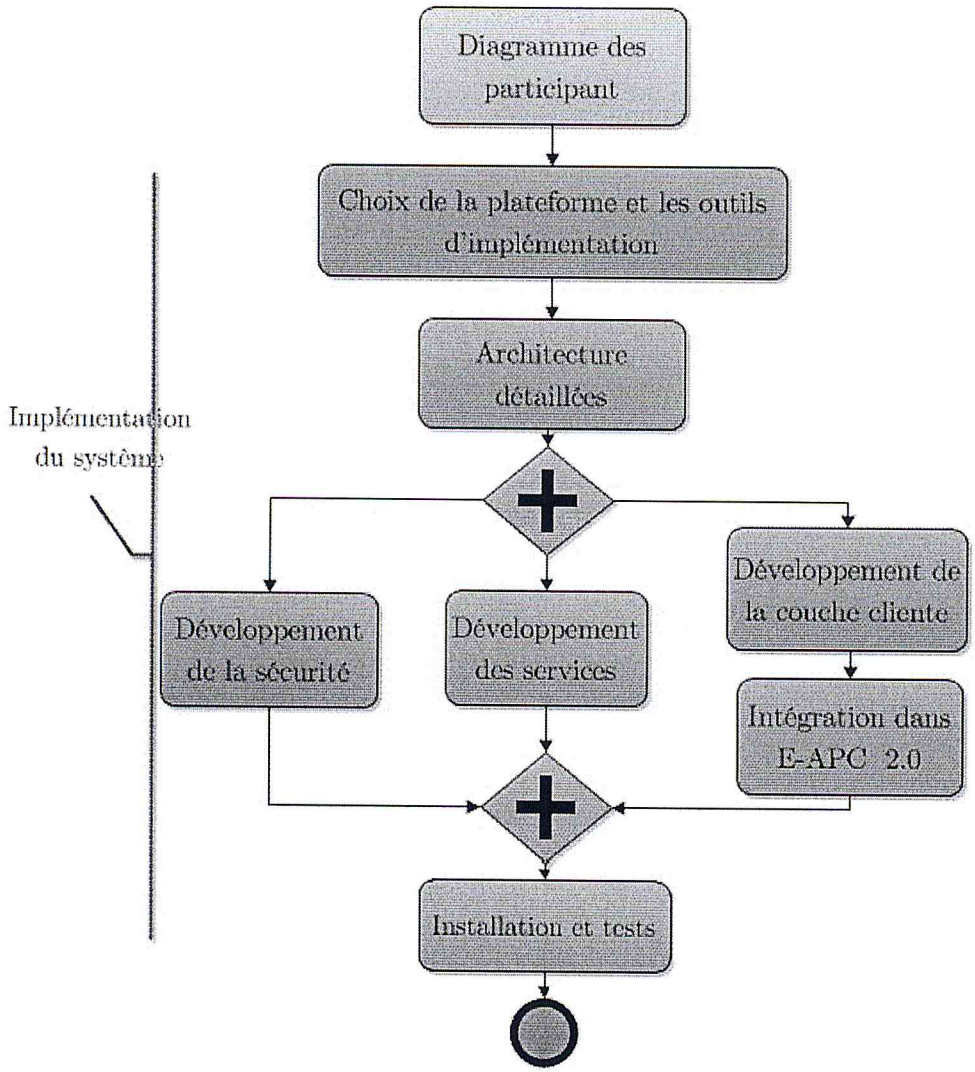


FIGURE 4.1 – Diagramme représentant la démarche du projet.

- Diagramme des messages ;
- Diagramme des participants (composants).

Présentation de la méthode de conception

Cette méthode exige la construction de trois modèles :
- Modèle d'architecture métier (haut niveau d'abstraction) ;
- Modèle d'architecture système ;
- Modèle de plateforme spécifique (JEE, Service Web...).

Le deuxième modèle est le résultat de l'opération de raffinement du premier, à ce stade on est plus proche de l'implémentation (dernier modèle).

Une description plus détaille sur ces diagrammes et sur la méthode SoaML est reportée dans l'Annexe A.

Outil de conception

Les diagrammes présentés dans ce chapitre ont été établi en utilisant l'outil **Modelio**, et le module "**Modelio SoaML Designer**", dédié à la modélisation d'architectures SOA selon le standard OMG SoaML [36].

Modelio est un outil de modélisation UML disponible sur les plates-formes Windows et Linux. Il intègre également la modélisation BPMN (Business Process Modeling Notation), et le support de la modélisation des exigences, du dictionnaire, des règles métier et des objectifs.

La version utilisée dans ce projet est **Modelio** free Edition 1.2.2.

Remarque
La partie implémentation est détaillée dans le chapitre suivant.

4.3 Définition et analyse des besoins

4.3.1 Contexte du projet

Dans le but d'informatisation d'état civil, plusieurs travaux ont été réalisés par l'équipe **RSI** (**R**éseaux et **S**ystèmes d'**I**nformation) de la division **ASM** (**A**rchitecture des **S**ystèmes et **M**ultimédias) du **CDTA** (**C**entre de **D**éveloppement des **T**echnologies **A**vancées).
Les résultats de ces travaux ont menés à la réalisation de deux systèmes :

- E-APC v 1.0 ;
- E-APC v 2.0 (étudié dans le chapitre 3).

Dans le cadre de ce projet, on se concentrera sur la version la plus récente (E-APC 2.0).

4.3.2 Problématiques

Avant d'entamer cette section, il est nécessaire de définir les différents types d'informations manipulés par le système E-APC 2.0.

- **Informations temporaires** : sont des données en cours de validation ;
- **Informations validées 13** : sont des données définitives. Ce type d'informations est passé par le processus de vérification/validation. Dans ce cas, ces données sont validées à base d'un document administratif présenté par le citoyen (par exemple le carnet familial) ;
- **Informations validées 12** : sont des données définitives. Ce type d'informations est passé par le processus de vérification/validation. Dans ce cas, ces données sont validées à base du registre.

Le schéma suivant représente l'architecture globale du système E-APC.

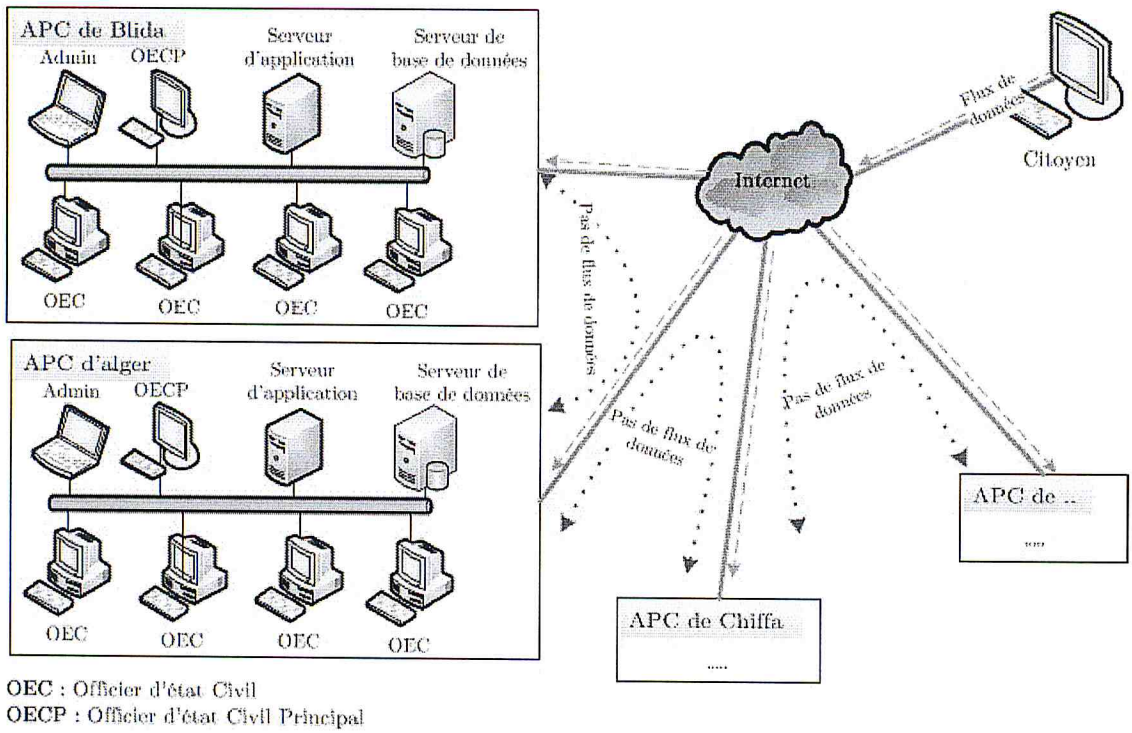


FIGURE 4.2 – Architecture globale d'E-APC.

A partir de notre étude sur le système E-APC, nous avons rencontré les problèmes suivants :

- Evolution séparée des APCs ;
- Redondance des informations au niveau du système complet ;
- L'opération de création des avis de mentions est manuelle ;
- L'absence d'un mécanisme de restauration/récupération des données ;
- L'incrémentation progressive des informations validées 13.

4.3.3 Solution proposée

Notre objectif est de :

- Réaliser un système de communication qui assure :
  - La cohérence et la mise à jour du système complet ;
  - La génération automatique des avis de mentions ;
  - La restauration des données lors d'un incident.
- Sécuriser le système :
  - Sécurité des échanges ;
  - Sécurité d'installation.
- Intégration et adaptation de l'application E-APC au nouveau système.

Nous proposons de construire une architecture orientée service, tel que :

- Chaque entité du système global (APC, Daïra, Tribunal et Ministère) expose et consomme des services et occupe un niveau bien précis dans cette architecture ;
- L'application E-APC 2.0 représente le portail (ou client de service) du système.

La figure 4.3 schématise la solution proposée.

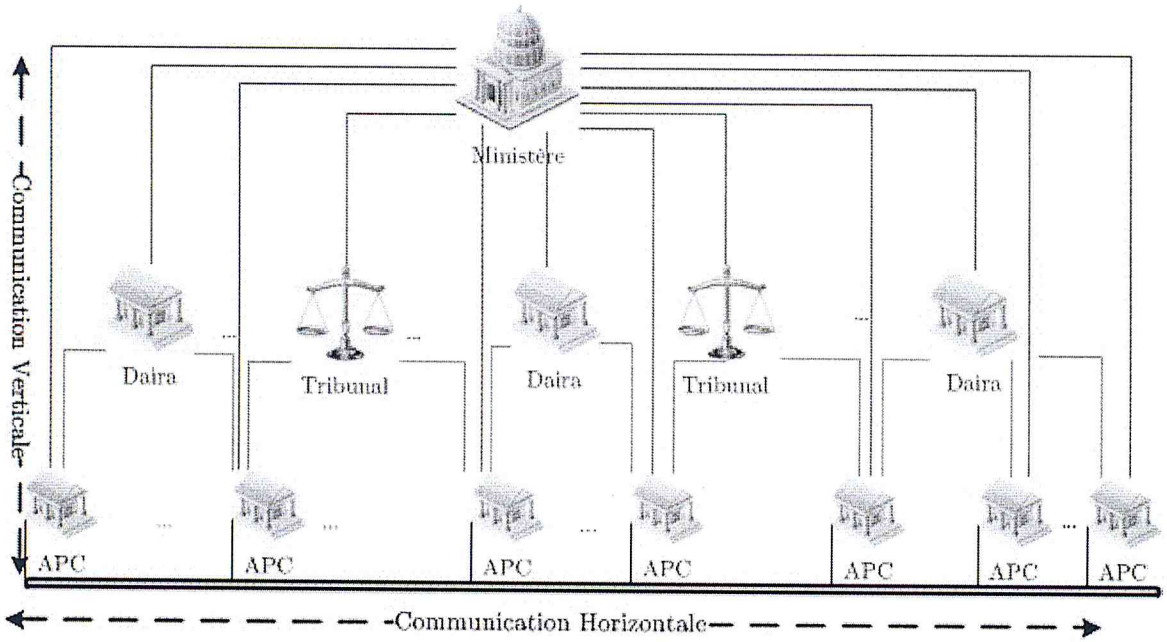


FIGURE 4.3 – Architecture globale du système de communication.

a. Les entités du système : Nous avons classifié les entités du système selon la nature des services fournis en trois niveaux de la hiérarchie :

- **Les APCs** : chaque entité expose des services de mise à jour, retrait des actes... ;
- **Les Daïras et les Tribunaux** : ces entités fournissent des services de restauration et de mise à jour du système ;
- **Le Ministère** : cette entité offre un service particulier nommé **Synchronizer**, ce dernier guide et surveille la communication entre toutes les autres entités. On peut le comparer à un **orchestrateur**.

b. Types de communication : Nous avons défini deux types de flux de communication :

- **Communication horizontale** : c'est la communication entre les entités de même nature, à noter que dans notre cas, ce type de communication réunie seulement les APCs ;
- **Communication verticale** : c'est la communication entre deux entités de nature différente, à noter aussi qu'il n'existe pas de communication entre l'entité Daïra et Tribunal.

Grâce à la communication horizontale, on assure que le système soit cohérent, mis à jour et qu'une information est validée **validation 12** et introduite une seule fois dans tout le système, ce qui réduit le taux des informations **validées 13**.

D'autre part, la communication verticale assure la restauration du système lors d'un incident, par le faite de la **duplication** d'information au niveau de la Daïra et du Tribunal aux quels appartient l'APC.

4.4 Conception du système

4.4.1 Services APCs

L'APC expose les services suivants :
- Service de retrait des actes ;
- Service de rapatriement des données ;
- Services de gestion des demandes d'actes ;
- Services de gestion des avis de mentions.

I. Description textuelle

a. Service de retrait des actes : Ce service permet aux citoyens de récupérer un acte (naissance, décès, divorce et mariage), via son identifiant.
Ce service offre quatre opérations :
- `getActeNaissance(in id) : ActeNaissance` ;
- `getActeMariage (in id) : ActeMariage` ;
- `getActeDivorce(in id) : ActeDivorce` ;
- `getActeDeces (in id) : ActeDeces`.

Chaque opération renvoie deux types de réponse :
- Un extrait de naissance numéro 12 : dans le cas du succès ;
- NULL : dans le cas d'erreur ou si les informations demandées n'existent pas dans la base des données.

b. Service de gestion des demandes d'actes de naissance : Il fournit les opérations de manipulation d'une demande d'un acte de naissance. Ce service offre quatre opérations :

`add(in demande de naissance) : booléen` → Permet d'ajouter une demande d'un acte de naissance, elle renvoie vrai en cas du succès et faux dans le cas contraire.

`update(in id,demande de naissance) : booléen` → Modifie la demande d'acte de naissance identifiée par **id**, cette opération retourne un booléen de valeur vrai si la mise à jour a réussi et faux sinon. Elle échoue si la demande concernée est entrée dans le processus de validation/vérification d'information.

`delete(in id) : booléen` → Permet de supprimer une demande d'un acte de naissance identifiée par **id**, elle renvoie vrai en cas du succès et faux dans le cas contraire. Cette opération échoue si la demande concernée est entrée dans le processus de validation/vérification d'information.

`getEtat(in id) : EtatValidation` → Permet de récupérer l'état de la demande identifiée par **id**, cette méthode retourne deux types de réponse :
- **NULL** : si la demande concernée n'existe pas ;
- **Etat de validation de la demande** : chaque demande possède quatre états :
  - **Nouvelle demande** ;
  - **Demande en cours de validation** ;
  - **Demande rejetée** : une demande est en état **rejetée** si ses informations ont été refusées par un ou plusieurs validateurs (ça dépond de la politique de l'APC) ;

- **Demande validée** : une demande est en état **validée** si ses informations ont été acceptées par un ou plusieurs validateurs (ça dépond de la politique de l'APC).

c. Service de gestion des demandes d'actes de mariage : Ce service offre la possibilité aux citoyens de créer, modifier et supprimer une demande d'un acte de mariage. Il fournit quatre opérations :

`add(in demande de mariage) : booléen` → Permet d'ajouter une demande d'un acte de mariage, elle renvoie vrai en cas du succès et faux dans le cas contraire.

`update(in id,demande de mariage) : booléen` → Cette opération modifie la demande identifiée par **id**, elle retourne faux s'il y a eu des erreurs et vrai en cas du succès. La mise à jour réussit si et seulement si la demande n'est pas rentrée dans le processus de validation/vérification d'information.

`delete(in id) : booléen` → Permet de supprimer une demande d'un acte de mariage identifiée par **id**, elle renvoie vrai en cas du succès et faux dans le cas contraire. Cette opération échoue si la demande concernée est entrée dans le processus de validation/vérification d'information.

`getEtat(in id) : EtatValidation` → Récupère l'état de la demande identifiée par **id**, cette méthode retourne deux types de réponse :

- **NULL** : si la demande concernée n'existe pas ;
- **Etat de validation de la demande** : voir section précédente.

Le service gestion des demandes d'acte de mariage est un peu plus complexe, en effet son exécution nécessite la collaboration avec des partenaires externes tels que **Service gestion des demandes d'acte de naissance**.

d. Service de rapatriement des données : Ce service permet aux APCs de collecter les informations sur ses citoyens à partir d'autres APCs. En théorie, on suppose que chaque APC expose ses services, mais malheureusement, en réalité ces services ne seront pas déployés sur toutes les APCs du territoire, donc certaines APCs seront en retard par rapport à d'autres. Afin de pallier à ce problème, chaque APC doit obligatoirement collecter toutes ces informations auprès de ses partenaires avant le lancement du système (c'est-à-dire à l'installation). Les données assemblées doivent passer par le processus de validation/vérification des informations.
Ce service fournit les opérations suivantes :

`getAllNaissance(in idAPC):Personne[*]` → Récupère toutes les informations qui concernent les citoyens de l'APC identifiée par **id**, et qui correspondent à un événement de naissance.

`getAllMariage(in idAPC):Mariage[*]` → Elle retourne tous les évènements de mariages qui concernent l'APC identifiée par **id**.

`getAllDivorce (in idAPC):Divorce[*]` → Elle retourne tous les évènements de divorces qui concernent l'APC identifiée par **id**.

`getAllDeces (in idAPC):Deces[*]` → Elle retourne tous les évènements de décès qui concernent l'APC identifiée par **id**.

e. Services de gestion des avis de mentions : Il fournit les opérations d'ajout des avis de mentions (décès, divorce ou mariage). Ce service expose trois méthodes :

`addMentionMariage(in Mention de marriage):booléen` → Cette opération ajout une mention de mariage, elle revoie faux s'il y a eu des erreurs, et vrai dans le cas de réussite.

`addMentionDivorce(in Mention de divorce): booléen` → Elle permet d'ajouter une mention de divorce, cette opération retourne faux dans le cas d'échoue et vrai dans le cas contraire.

`addMentionDeces(in Mention de décès): booléen` → Ajout une mention de décès, cette opération retourne un booléen de valeur vrai en cas du succès et faux dans le cas contraire.

II. Modèle architecture métier

Au niveau de ce modèle, nous avons pu identifier cinq participants :

- APC : ce participant joue deux rôles majeurs :
  - Consommateur du service
    - Administrateur : dans le cas où l'action à accomplir nécessite une présence humaine ;
    - APC : dans le cas où l'action à réaliser est automatique et ne nécessite aucune surveillance.
  - Fournisseur du service (S-APC) ;
- Citoyen : au niveau des services, on ne s'intéresse pas au sujet des profils des utilisateurs clients, ainsi un citoyen représente soit un administrateur, soit un officier d'état civil... ;
- Daïra : joue deux rôles majeurs :
  - Consommateur du service (Daïra) ;
  - Fournisseur du service (S-Daïra).
- Tribunal : ce participant joue deux rôles majeurs :
  - Consommateur du service (Tribunal) ;
  - Fournisseur du service (S-Tribunal).
- Ministère : joue un seul rôle : fournisseur des services.

Remarque :
Vu que le nombre de ces diagrammes est élevé, les autres services sont détaillés dans le manuel attaché à ce mémoire.

a. Diagramme d'architecture des services : La figure 4.4 représente l'architecture globale des services proposés par l'APC (**S-APC**).

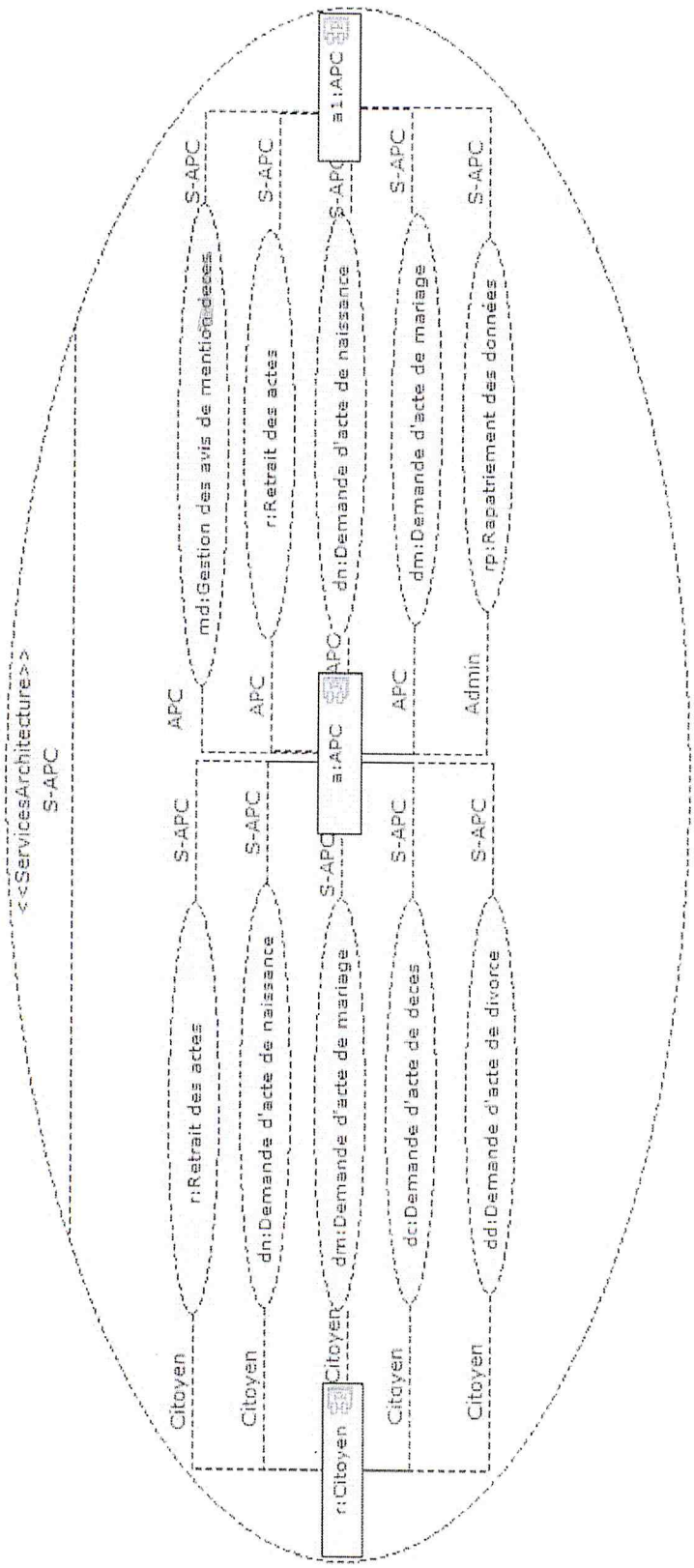


FIGURE 4.4 – Architecture des services APC.

b. Diagramme de processus métier : Dans cette section, on va présenter deux processus métier :

- Processus de retrait d'un acte de mariage ;
- Processus de déclaration d'un événement de mariage.

Vu que la taille de ces deux diagrammes est grande, on a décidé de les présenter comme un document attaché au mémoire.

Processus de retrait d'un acte de mariage : Afin de lever l'ambigüité sur les noms d'activités présentées dans ce diagramme, le tableau suivant a été construit :

Activité	Signification
Récupérer ActeNEpse	Récupérer acte de naissance de l'épouse.
Récupérer ActeNEpx	Récupérer acte de naissance de l'époux.
Générer DmdNEpse	Générer demande acte naissance pour l'épouse.
Générer DmdNEpx	Générer demande acte naissance pour l'époux.
Ajouter demandeM	Ajouter demande de mariage.
Débloquer demandeM	Débloquer demande mariage.

TABLE 4.1 – Processus de retrait d'un acte de mariage.

Ce processus implique les participants suivants :

- Citoyen ;
- Synchronizer ;
- E-APC : le portail (application E-APC 2.0) ;
- APC LNEpoux : l'APC de naissance de l'époux ;
- APC LNEpouse : l'APC de naissance de l'épouse ;
- APC-M : lieu de mariage.

Scénario

- Le citoyen demande un acte de mariage ;
- Le système E-APC vérifie l'existence du service APC-M ;
- Si le service n'existe pas
  - La demande sera traitée localement (système E-APC) ;
  - L'E-APC signal auprès du **Synchronizer** qu'elle détient des informations concernant l'APC-M.
- Sinon
  - Le système E-APC sollicite le service de retrait des actes de l'APC de mariage, et récupère le résultat ;
  - Si l'acte existe, l'E-APC génère l'acte sous forme PDF, puis renvoie le résultat au citoyen.
  - Sinon (**Ajouter une demande d'un acte de mariage**)

- ◇ L'E-APC génère puis envoie une demande d'acte de mariage au service de l'APC-M ;
- ◇ Le service de l'APC-M récupère les adresses des services **APC LNEpoux** et **APC LNEpouse** ;
- ◇ Si le service en question (**APC LNEpoux** ou **APC LNEpouse**) existe
  - * Le service de l'APC-M récupère l'acte de l'époux et de l'épouse ;
  - * Si ces actes n'existent pas, le service de l'APC-M génère les demandes appropriées, et les envoie à l'APC **LNEpoux** (**LNEpouse**).
- ◇ Sinon, le service de l'APC-M signal auprès du **Synchronizer** qu'il détient des informations concernant un citoyen de l'APC **LNEpoux** (**LNEpouse**).
- ◇ Ajouter la demande d'acte de mariage (demande à l'état bloqué) ;
- ◇ Lorsque la demande d'acte de naissance de l'époux (respectivement épouse) est traitée :
  - * Si au moins une demande a été rejetée, le système (APC-M) supprime la demande d'acte de mariage puis informe l'E-APC sur ce résultat.
  - * Sinon
    - ▷ Débloquer la demande d'acte de mariage ;
    - ▷ Lorsque la demande d'acte de mariage est traitée, le système (APC-M) informe l'E-APC sur ce résultat ;
    - ▷ Si la demande d'acte de mariage est validée
      - L'E-APC génère l'acte sous forme PDF ;
      - Renvoyer le résultat au citoyen.
    - ▷ Sinon
      - L'E-APC génère un message d'erreur ;
      - Renvoyer le résultat au citoyen.

Processus de déclaration d'un événement de mariage : Afin de lever l'ambigüité sur les noms d'activités présentées dans ce diagramme, le tableau suivant a été construit :

Activité	Signification
Déclarer evt mariage	Déclarer évènement de mariage.

TABLE 4.2 – Processus de déclaration d'un événement de mariage.

Ce processus implique les participants suivants
- Citoyen ;
- **Synchronizer** ;
- E-APC : le portail ;
- APC LNEpoux : l'APC de naissance de l'époux ;
- APC LNEpouse : l'APC de naissance de l'épouse ;
- Daïra.

Scénario

- Le citoyen déclare un événement de mariage ;
- Lorsque cette déclaration est traitée :
  - Si la déclaration n'est pas valide
    - L'E-APC génère un message d'erreur ;
    - Renvoyer le résultat au citoyen.
  - Sinon
    - L'E-APC vérifie l'existence du service de l'APC **LNEpoux** (respectivement **LNEpouse**) auprès du Synchronizer ;
      - Si le service en question existe
        - Générer une mention de mariage ;
        - Envoyer cette mention à l'APC **LNEpoux** (respectivement **LNEpouse**).
    - L'E-APC vérifie l'existence du service **Daïra** auprès du Synchronizer ;
      - Si ce service existe
        - Générer une déclaration de mariage ;
        - Envoyer cette déclaration au service Daïra.
- L'E-APC génère un message de succès ;
- Renvoyer le résultat au citoyen.

c. Diagramme de contrats des services : Dans cette section, on décrit les services suivants :

- Service de retrait des extraits ;
- Service de gestion des demandes de naissance ;
- Service de rapatriement des données ;
- Service de gestion des avis de mentions.

Certains participants possèdent un ou plusieurs rôles, et il est nécessaire d'établir un diagramme de contrat du service pour chacun de ses rôles (les participants sont décrits dans la section 4.4.1.II), ce qui explique la présence de deux diagrammes de contrat du service presque identiques qui décrivent le même service (par exemple figure 4.5 et 4.6). Les figures 4.5 et 4.6 représentent les contrats du service de retrait des actes.

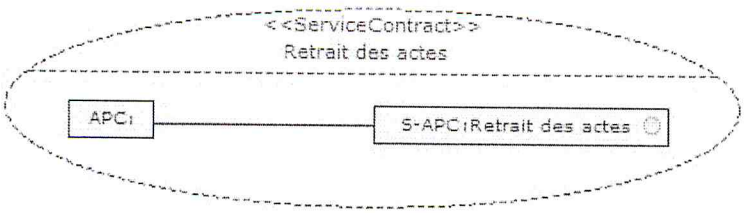


FIGURE 4.5 – Contrat du service de Retrait d'acte pour le client du rôle APC.

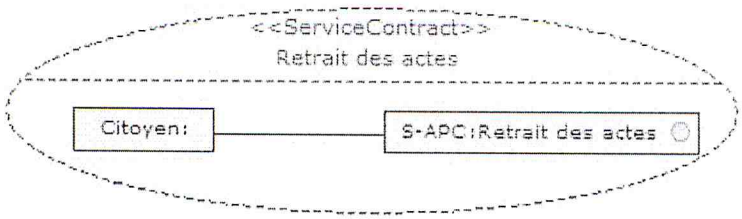


FIGURE 4.6 – Contrat du service de Retrait d'acte pour le client du rôle Citoyen.

Les figures 4.7 et 4.8 représentent les contrats du service gestion des demandes d'actes de naissance.

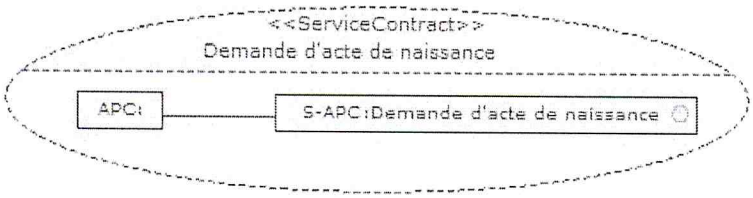


FIGURE 4.7 – Contrat du service Demande d'acte de naissance pour le client du rôle APC.

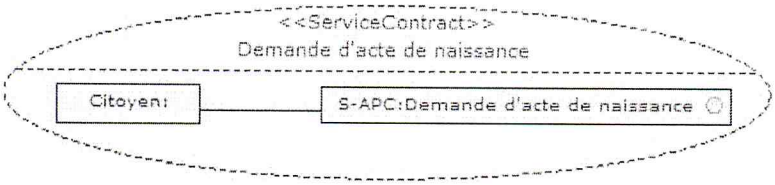


FIGURE 4.8 – Contrat du service Demande d'acte de naissance pour le client du rôle Citoyen.

La figure 4.9 décrit le contrat du service de rapatriement des données.

FIGURE 4.9 – Contrat du service Rapatriement des données.

La figure 4.10 présente le contrat du service gestion des avis de mentions.

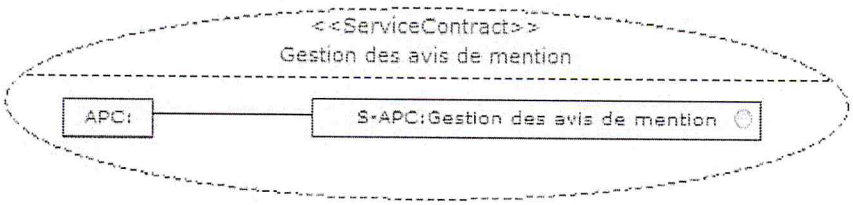


FIGURE 4.10 – Contrat du service Gestion des avis de mention.

III. Modèle Architecture système

a. Diagramme d'interface du service : Dans cette section, on décrit seulement les services suivants :

- Service de retrait des extraits ;
- Service de gestion des demandes de naissance ;
- Service de rapatriement des données ;
- Service de gestion des avis de mentions.

Au niveau de ce type de diagrammes, on ne s'intéresse ni aux rôles joués par les clients du service ni à la structure des messages requête et réponse du service lui-même.
La figure 4.11 décrit en détail l'interface du service de retrait des actes. Les types de messages présentés dans ce diagramme seront étudiés dans le paragraphe **b**.

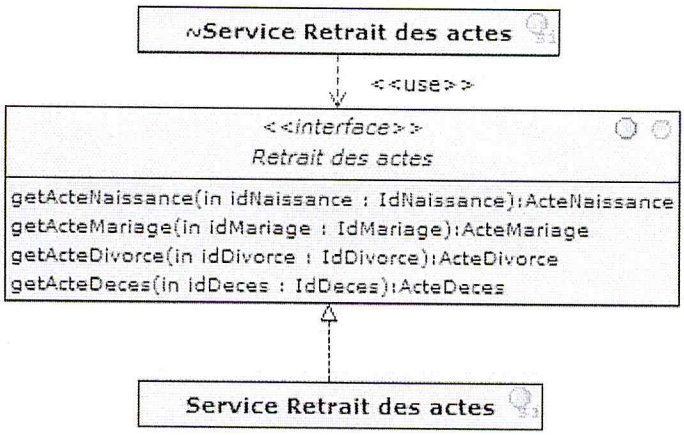


FIGURE 4.11 – Interface du service Retrait des actes.

La figure 4.12 présente l'interface du service de gestion des demandes d'actes de naissance. Les types de messages manipulés ici seront étudiés dans le paragraphe **b**.

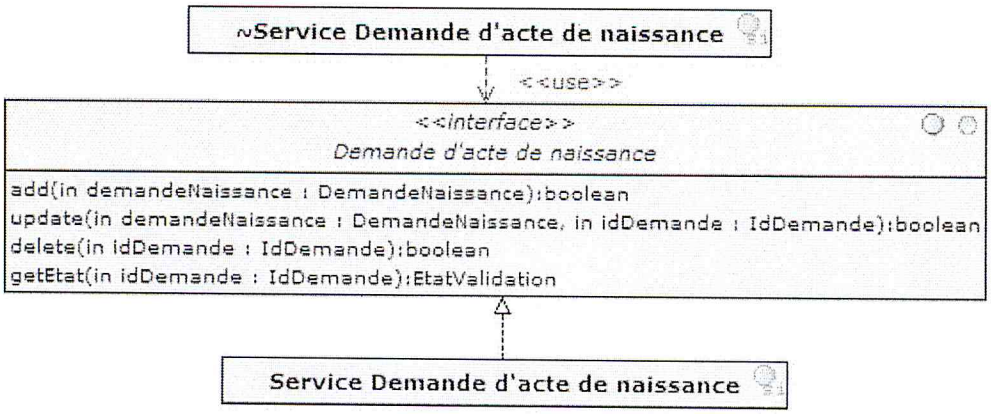


FIGURE 4.12 – Interface du service Demande d'actes de naissance.

La figure 4.13 décrit l'interface du service de rapatriement des données. Les types de messages présentés dans ce diagramme seront étudiés dans le paragraphe **b**.

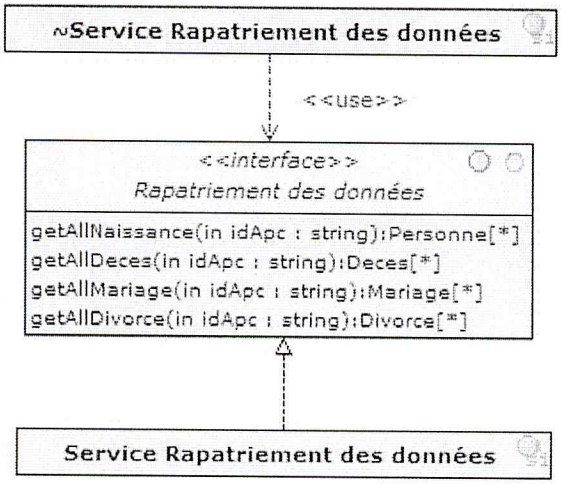


FIGURE 4.13 – Interface du service Rapatriement des données.

La figure 4.14 montre l'interface du service de gestion des avis de mentions. Les types de messages traités dans ce diagramme seront étudiés dans le paragraphe **b**.

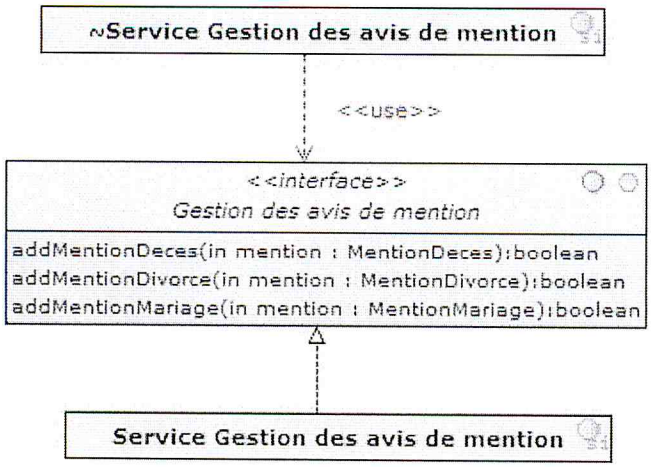


FIGURE 4.14 – Interface du service Gestion des avis de mention.

b. Diagramme des messages : Dans cette section, on décrit les services suivants :

- Service de retrait des extraits ;
- Service de gestion des demandes de naissance ;
- Service de rapatriement des données ;
- Service de gestion des avis de mentions.

Nous avons décrit dans les sections précédentes les opérations qu'offre chaque service, sans détailler la structure et la forme des messages échangés. Dans cette partie, on s'intéresse à étudier cet aspect en présentant quelques diagrammes des messages.
Chaque message échangé doit être identifié, dans ce projet, nous avons proposé d'utiliser l'idApc comme identifiant.
Le tableau 4.3 donne pour chaque opération la structure globale des messages requête et réponse du service **Retrait actes**.

Opération	Type de message	Forme globale	Structure détaillée
`getActeNaissance`	Requête	**IdNaissance** anneeDecoupage : string idCommune : Commune dateNaissance : date numSequentiel : integer	– –
	Réponse	**ActeNaissance**	Voir figure 4.15
`getActeMariage`	Requête	IdMariage	même que IdNaissance
	Réponse	**ActeMariage**	Voir figure 7 du manuel
`getActeDivorce`	Requête	IdDivorce	même que IdNaissance
	Réponse	**ActeDivorce**	Voir figure 8 du manuel
`getActeDeces`	Requête	IdDeces	même que IdNaissance
	Réponse	**ActeDeces**	Voir figure 9 du manuel

TABLE 4.3 – Diagramme des messages pour le service "Retrait actes".

La figure 4.15 montre le diagramme de message ActeNaissance.

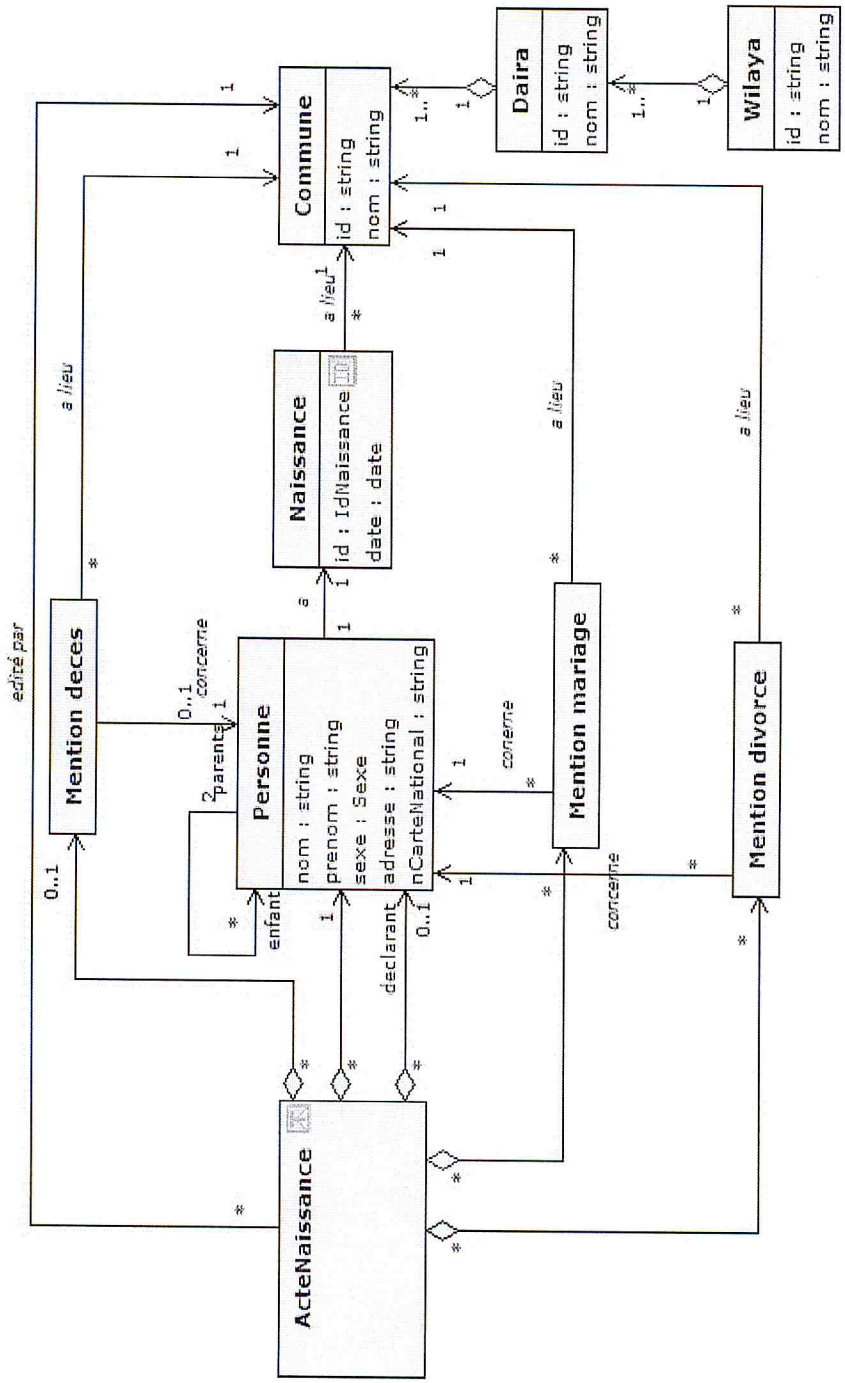


FIGURE 4.15 – Type de message ActeNaissance.

Pour que cette modélisation soit juste, on lui associée les contraintes suivantes (écrite en OCL) :

Contrainte sur la relation parentale

```
context Personne
def: parent1 = self.parents -> at(0)
def: parent2 = self.parents -> at(1)

context Personne
inv:
if parent1.sexe = #masculin
then -- parent1 est un homme
     parent2.sexe = #feminin
else -- parent1 est une femme
     parent2.sexe = #masculin
endif
```

Le tableau 4.4 décrit pour chaque opération la structure globale des messages requête et réponse du service **Demande actes de naissance**.

Opération	Type de message	Forme globale	Structure détaillée
Add	Requête	**DemandeNaissance**	Voir figure 4.16
	Réponse	**Boolean**	--
Update	Requête	**IdDemande** anneeDemande : string codeDemande : string numSequentiel : string apcSource : Commune	--
		DemandeNaissance	Voir figure 4.16
	Réponse	**Boolean**	--
Delete	Requête	**IdDemande**	--
	Réponse	**Boolean**	--
getEtat	Requête	**IdDemande**	--
	Réponse	<<enumeration>> EtatValidation Nouvelle Rejeté Validé EnCours	--

TABLE 4.4 – Diagramme des messages pour le service "Demande actes de naissance".

La figure 4.16 montre le diagramme de message **DemandeNaissance**.

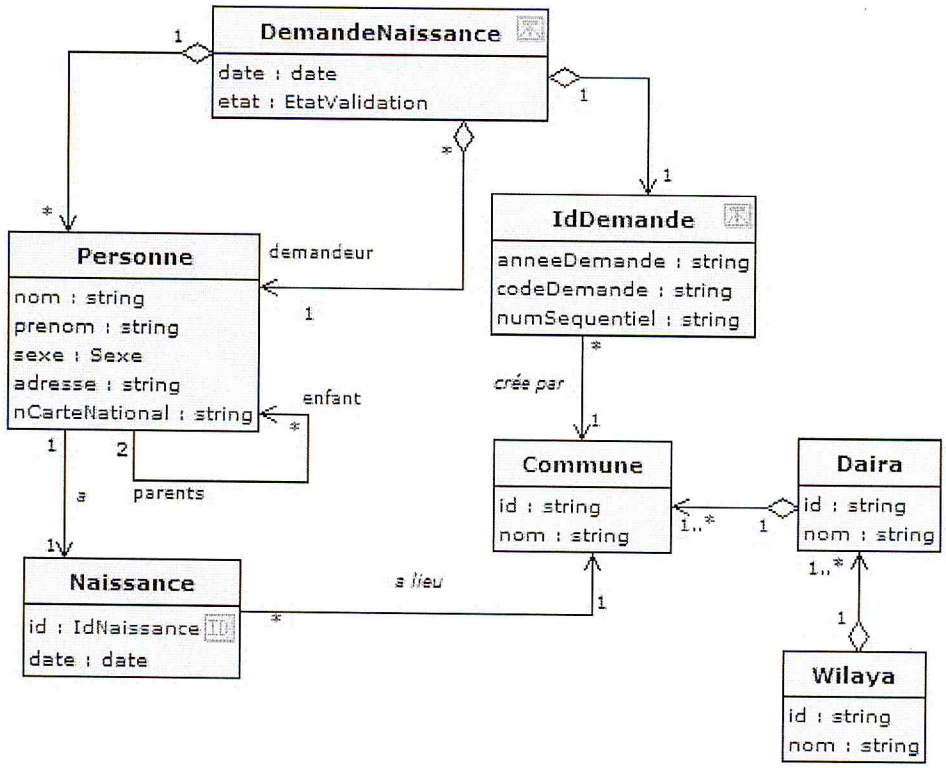


FIGURE 4.16 – Type de message DemandeNaissance.

Le tableau 4.5 donne pour chaque opération la structure globale des messages requête et réponse du service **Rapatriement des données**.

Opération	Type de message	Forme globale	Structure détaillée
`getAllNaissance`	Requête	Chaine de caractères	– –
	Réponse	**Personne**	Voir figure 4.17
`getAllMariage`	Requête	Chaine de caractères	– –
	Réponse	**Mariage**	Voir figure 13 du manuel
`getAllDivorce`	Requête	Chaine de caractères	– –
	Réponse	**Divorce**	Voir figure 14 du manuel
`getAllDeces`	Requête	Chaine de caractères	– –
	Réponse	**Deces**	Voir figure 15 du manuel

TABLE 4.5 – Diagramme des messages pour le service "Rapatriement des données".

La figure 4.17 montre le diagramme de message **Personne** :

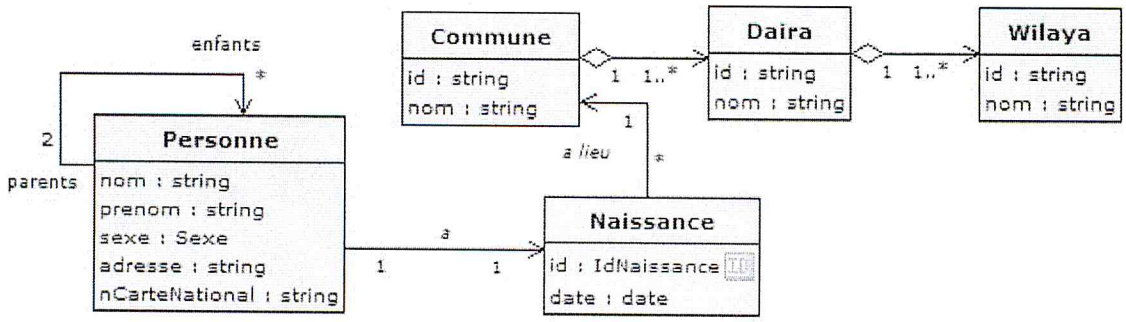


FIGURE 4.17 – Type de message Personne.

Le tableau 4.6 décrit la structure globale des messages requête et réponse échangés entre le consommateur et le service **Gestion des avis de mention.**

Opération	Type de message	Forme globale	Structure détaillée
`addMentionMariage`	Requête	**MentionMariage**	Voir figure 4.18
	Réponse	**boolean**	– –
`addMentionDivorce`	Requête	**MentionDivorce**	Voir figure 16 du manuel
	Réponse	**boolean**	– –
`addMentionDeces`	Requête	**MentionDeces**	Voir figure 17 du manuel
	Réponse	**boolean**	– –

TABLE 4.6 – Diagramme des messages pour le service "Gestion des avis de mention".

La figure 4.18 montre le diagramme de message **Mention mariage** :

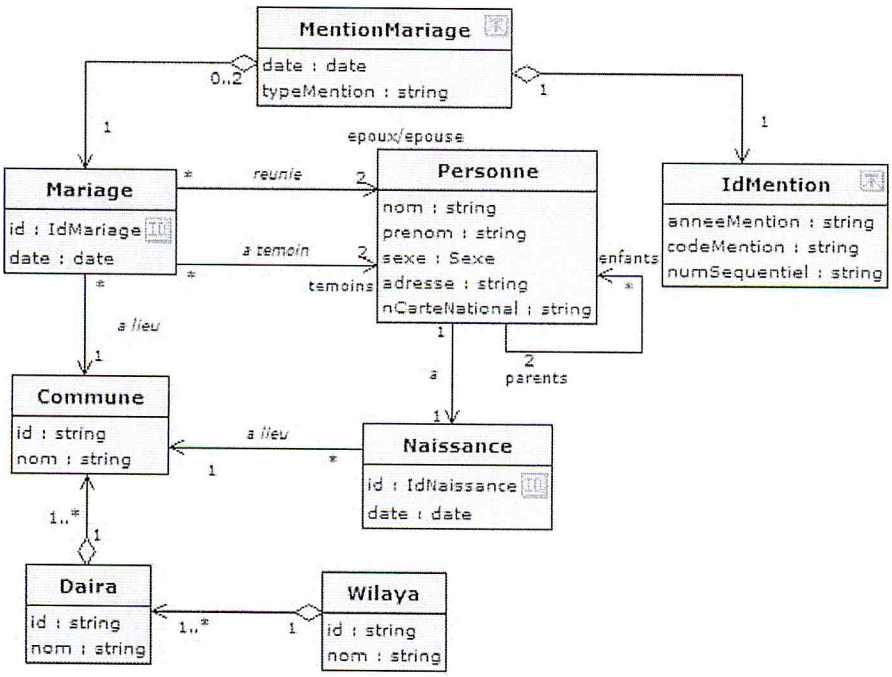


FIGURE 4.18 – Type de message Mention mariage.

Pour que cette modélisation soit correcte, on lui associée les contraintes suivantes (écrite en OCL) :

Contrainte sur la relation de mariage

```
context Mariage
def: personne1 = self.epoux/epouse -> at(0)
def: personne2 = self.epoux/epouse -> at(1)

context Mariage
inv:
if personne1.sexe = #masculin
then -- personne1 est l'epoux
     personne2.sexe = #feminin
else -- personne1 est l'epouse
     personne2.sexe = #masculin
endif
```

Contrainte sur la relation de témoignage

```
context Mariage
def: temoin1 = self.temoins -> at(0)
def: temoin2 = self.temoins -> at(1)

context Mariage
inv: temoin1.sexe = #masculin and temoin2.sexe = #masculin
```

4.4.2 Services Daïra/Tribunal

La Daïra (Tribunal) expose les services suivants :
- Service de mise à jour du système ;
- Service de restauration du système.

I. Description textuelle

a. Service de restauration du système : Ce service permet aux APCs de restaurer leurs systèmes en cas d'un incident. Il offre quatre opérations :

`getAllDeclarationNaissance(in idAPC) : DeclarationNaissance[*]` → Retourne tous les évènements de naissances déclarés au niveau de l'APC identifiée par **idAPC**.

`getAllDeclarationMariage(in idAPC) : DeclarationMariage[*]` → Renvoie tous les évènements de mariage déclarés au niveau de l'APC identifiée par **idAPC**.

`getAllDeclarationDivorce(in idAPC) : DeclarationDivorce[*]` → Elle récupère tous les évènements de divorce déclarés au niveau de l'APC identifiée par **idAPC**.

`getAllDeclarationDeces(in idAPC) : DeclarationDeces[*]` → Cette opération renvoie tous les évènements de décès déclarés au niveau de l'APC identifiée par **idAPC**.

b. Service de mise à jour des informations de naissance : Le but de ce service est de fournir aux APCs un mécanisme, qui permet de créer une copie des données concernant un évènement de naissance au niveau de la daïra. Ces informations doivent passer par un processus de validation/vérification avant l'opération de sauvegarde.

Ce service offre trois opérations :

`addDeclaration(in déclaration de naissance) : booléen` → Créer une sauvegarde d'une déclaration de naissance au niveau de la Daïra, elle renvoie vrai en cas du succès et faux dans le cas contraire.

`updateDeclaration(in id, déclaration de naissance) : booléen` → Permet de modifier une déclaration de naissance identifiée par **id**, elle renvoie vrai en cas du succès et faux dans le cas contraire.

`deleteDeclaration(in id) : booléen` → Supprime une déclaration de naissance identifiée par **id**, elle renvoie vrai en cas du succès et faux dans le cas contraire.

II. Modèle architecture métier

a. Diagramme d'architecture des services : La figure 4.19 représente l'architecture globale des services proposés par la Daïra (S-Daira).

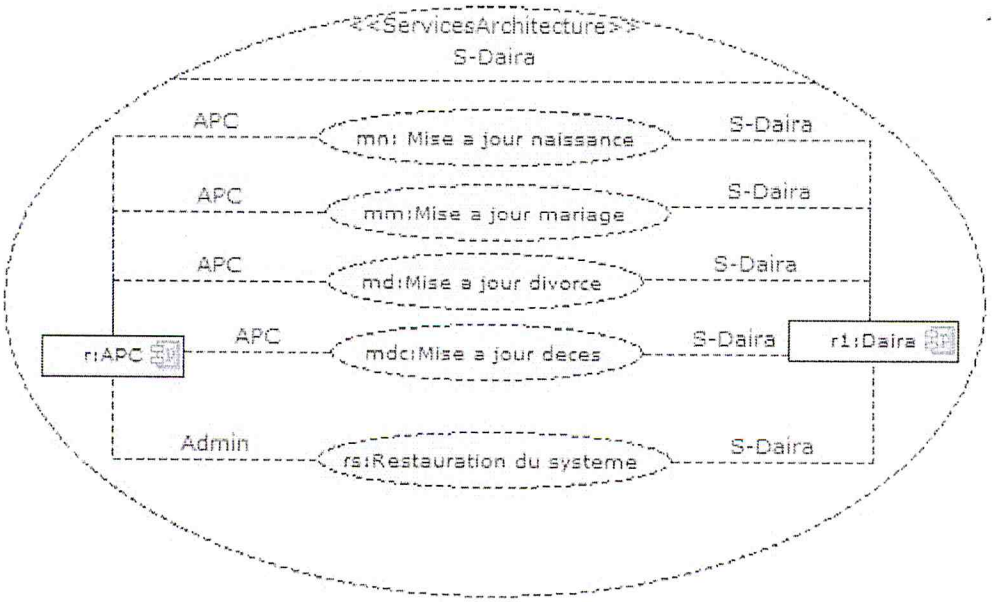


FIGURE 4.19 – Architecture des services Daïra.

b. Diagramme de contrats des services : Dans cette section, on décrit les services suivants :

- Service de restauration du système ;
- Service de mise à jour des informationsde naissance.

La figure 4.20 représente le contrat du service de restauration du système.

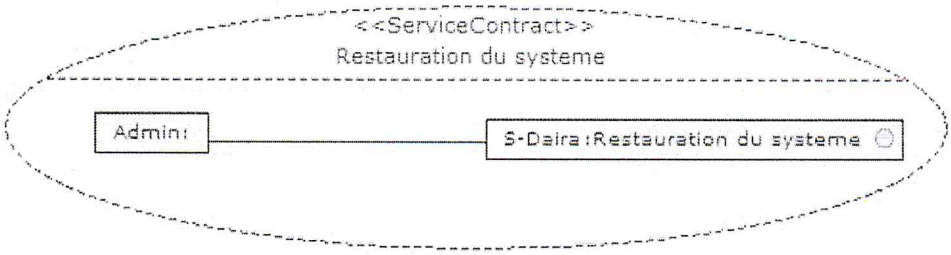


FIGURE 4.20 – Contrat du service Restauration du système.

La figure 4.21 montre le contrat du service de mise à jour des informations de naissance.

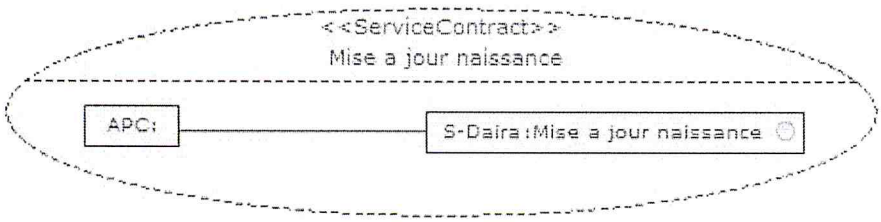


FIGURE 4.21 – Contrat du service de mise à jour naissance.

III. Modèle Architecture système

a. Diagramme d'interface du service : Dans cette section, on décrit les services suivants :

- Service de restauration du système ;
- Service de mise à jour des informationsde naissance.

La figure 4.22 décrit en détail l'interface du service restauration du système. Les types de messages présentés dans ce diagramme seront étudiés dans le paragraphe **b.**

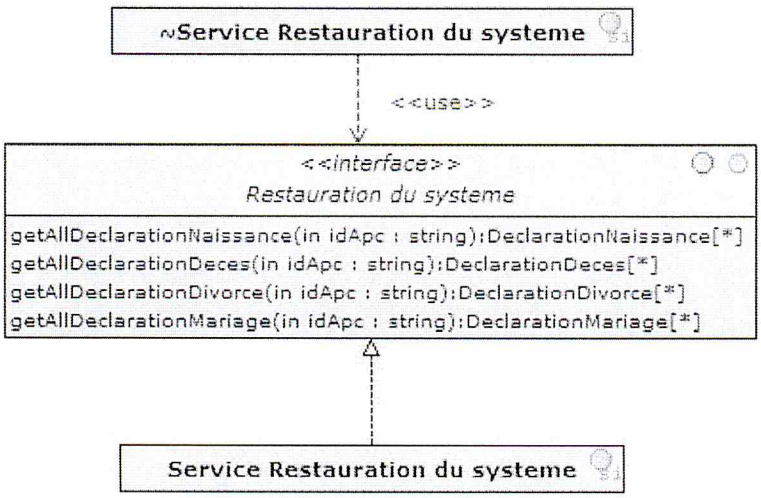


FIGURE 4.22 – Interface du service Restauration système.

La figure 4.23 montre l'interface du service de mise à jour des informations de naissance. Les types de messages traités ici seront étudiés dans le paragraphe **b.**

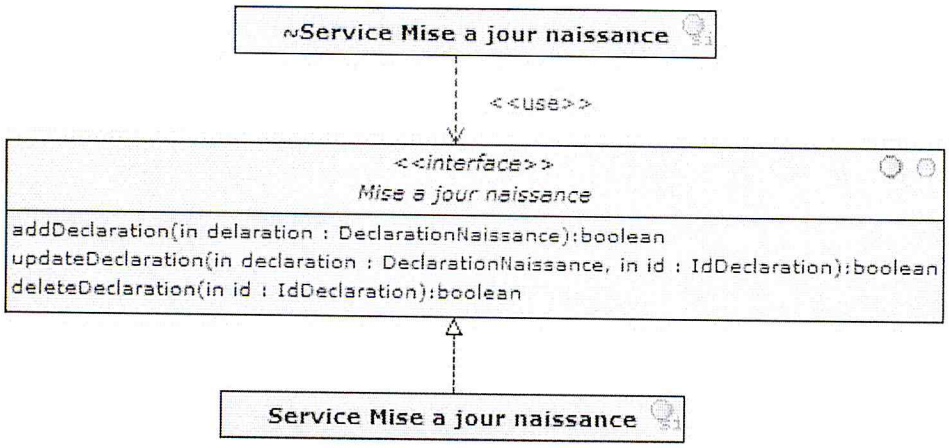


FIGURE 4.23 – Interface du service Mise à jour naissance.

b. Diagramme des messages : Dans cette section, on décrit les services suivants :

- Service de restauration du système ;
- Service de mise à jour des informations de naissance.

L'identifiant de l'APC respecte la codification proposée par les réalisateurs du système E-APC 2.0 (voir chapitre 3).

Le tableau 4.7 traite pour chaque opération la structure globale des messages requête et réponse du service **Restauration système**.

Opération	Type de message	Forme globale	Structure détaillée
`getAllDeclarationNaissance`	Requête	Chaine de caractères	– –
	Réponse	**DeclarationNaissance**	Voir figure 4.24
`getAllDeclarationMariage`	Requête	Chaine de caractères	– –
	Réponse	**DeclarationMariage**	Voir figure 24 du manuel
`getAllDeclarationDivorce`	Requête	Chaine de caractères	– –
	Réponse	**DeclarationDivorce**	Voir figure 25 du manuel
`getAllDeclarationDeces`	Requête	Chaine de caractères	– –
	Réponse	**DeclarationDeces**	Voir figure 26 du manuel

TABLE 4.7 – Diagramme des messages pour le service "Restauration système".

La figure 4.24 montre le diagramme de message **DeclarationNaissance**.

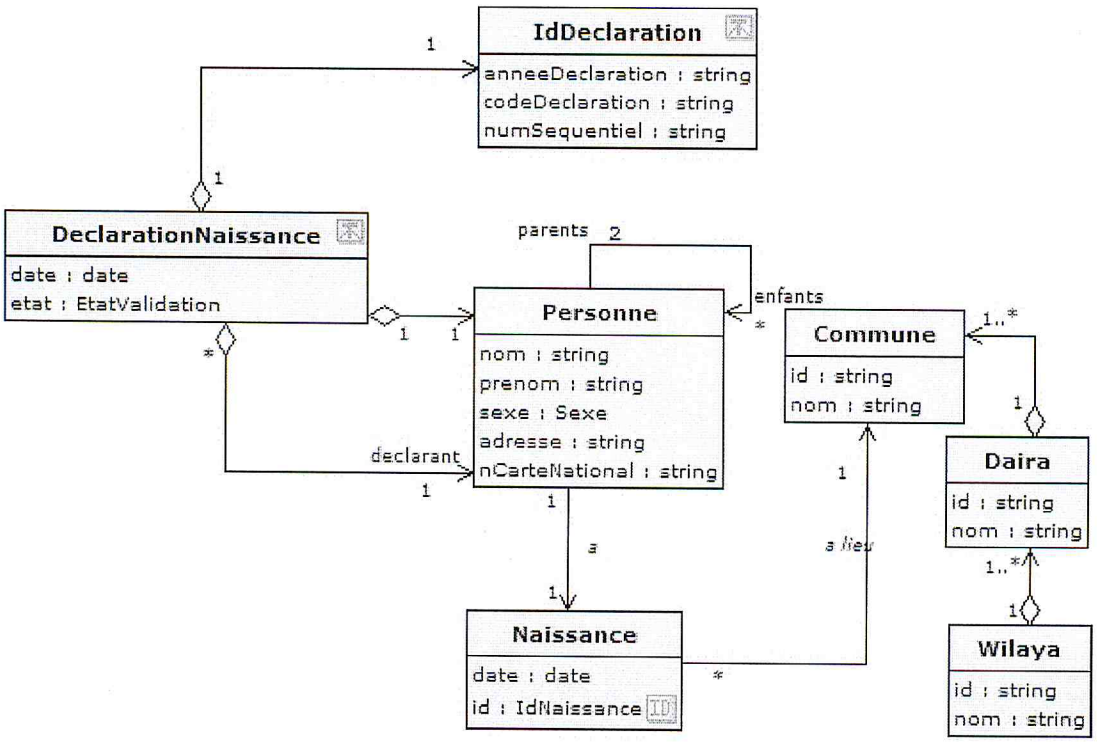


FIGURE 4.24 – Type de message DeclarationNaissance.

Le tableau 4.8 décrit pour chaque opération la structure globale des messages requête et réponse échangés entre le consommateur et le service **Mise à jour des informations de naissance**.

Opération	Type de message	Forme globale	Structure détaillée
addDeclaration	Requête	**DeclarationNaissance**	Voir figure 4.24
	Réponse	**booléen**	– –
updateDeclaration	Requête	**DeclarationNaissance**	Voir figure 4.24
	Réponse	**IdDeclaration** anneeDeclaration : string codeDeclaration : string numSequentiel : string idApcSource : Commune	– –
		booléen	– –
deleteDeclaration	Requête	**IdDeclaration**	– –
	Réponse	**booléen**	– –

TABLE 4.8 – Diagramme des messages pour le service "Mise à jour des informations de naissance".

4.4.3 Services Ministère

Le Synchronizer (déployé au ministère ou un autre lieu sécurisé) fournit les services suivants :

- Service de localisation des entités ;
- Service d'allocation de configuration réseau ;
- Service de gestion des comptes utilisateur ;
- Service de gestion des informations rapatriées.

I. Description textuelle

a. Service de localisation des entités : Son but est de fournir des configurations réseau aux entités du système (APC, Daïra et Tribunal). Ainsi chaque nouvelle instance installée sur le système doit récupérer une configuration auprès de ce service, dès son premier lancement. Le fonctionnement de ce service est inspiré du protocole **DHCP** (**D**ynamic **H**ost **C**onfiguration **P**rotocol).

Ce service offre une seule opération :

`allocate(in idEntite) : Chaine des caractères` → Permet d'allouer une configuration réseau à l'entité identifiée par **idEntite** et l'enregistrer dans le système. Cette opération retourne l'adresse de l'entité sous forme d'une chaine de caractères.

b. Service d'allocation de configuration réseau : Ce service fournit une configuration réseau pour l'entité en question. En effet, chaque service doit changer régulièrement sa configuration réseau, ce qui le rend dynamique et difficilement localisé, donc le taux d'attaques contre ces services sera affaibli. Par contre, la configuration réseau du service Synchronizer ne varie pas, sinon ce service ne sera plus accessible.

Ce service fournit trois opérations :

`locateAPC(in idAPC) : chaine de caractère` → Renvoie une configuration réseau pour l'APC identifiée par **idAPC** sous forme d'une chaine de caractères.

`locateDaira(in idDaira) : chaine de caractère`

`locateTribunal(in idTribunal) : chaine de caractère`

c. Service de gestion des comptes utilisateur : Ce service a été conçu afin d'assurer qu'un citoyen possède un et un seul compte utilisateur dans tous le système (toutes les APCs).

Il expose deux opérations :

`hasCompte(in idPersonne):Boolean` → Cette opération permet de vérifier si une personne possède déjà un compte utilisateur dans tout le système, elle renvoie vrai si la personne en question (identifiée par **idPersonne**) possède un compte et faux dans le cas contraire.

`addCompte(in utilisateur): Boolean` → Elle permet d'ajouter un compte utilisateur dans le système Synchronizer, cette entrée sert de référence pour des vérifications ultérieures (via l'opération **hasCompte**).

d. Service de gestion des informations rapatriées : Il permet d'accélérer le processus de rapatriement des informations d'une APC, en effet le système **Synchronizer** stocke pour chaque **APC** la liste des entités (APCs) qui détiennent des données concernant les citoyens de l'APC en question.

Ce service fournit deux opérations :

`getApcMap(in idApc) : Chaine[ ]` → Cette opération retourne la liste des APCs qui détiennent des données concernant les citoyens de l'APC identifiée par **idApc**.

`createMap(in idApcO, idApcD) : Boolean` → Cette opération permet de créer une instance de **MAP** (**idApcO**, **idApcD**) telle que :

- **idApcO** : représente l'identifiant de l'APC d'origine ;
- **idApcD** : représente l'identifiant de l'APC qui détient les informations concernant les citoyens d'**idApcO**.

Elle retourne un booléen qui indique si l'opération de la création de MAP a réussi.

II. Modèle architecture métier

a. Diagramme d'architecture des services : La figure 4.25 représente l'architecture globale des services proposés par le ministère (**Synchronizer**).

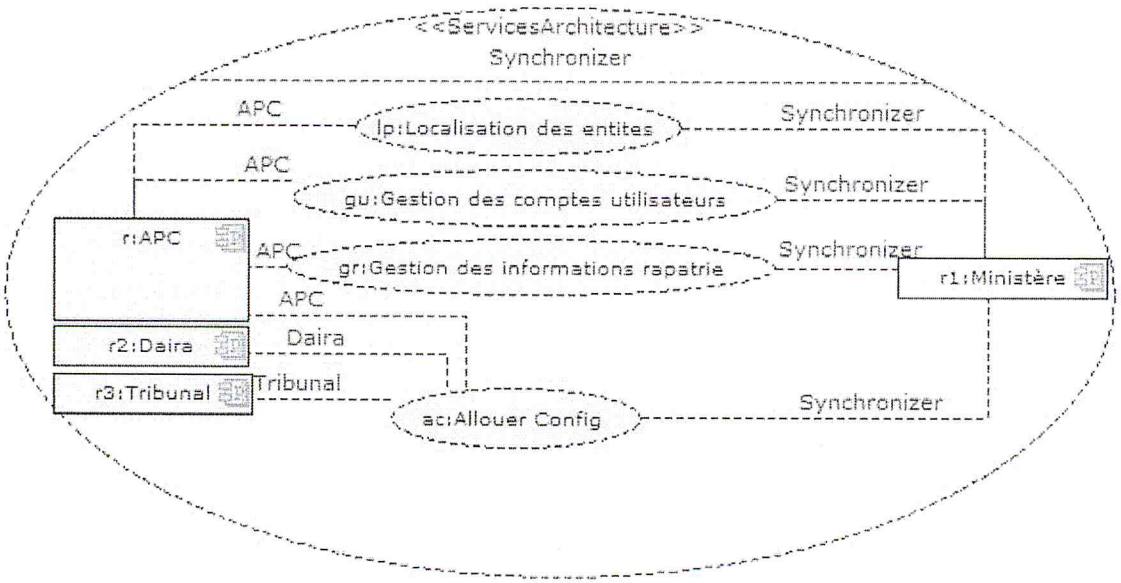


FIGURE 4.25 – Architecture du Synchronizer.

b. Diagramme de contrats des services : La figure 4.26 représente le contrat du service de localisation des entités.

FIGURE 4.26 – Contrat du service localisation des entités.

Les figures 4.27, 4.28 et 4.29 montrent les contrats du service d'allocation de configuration réseau.

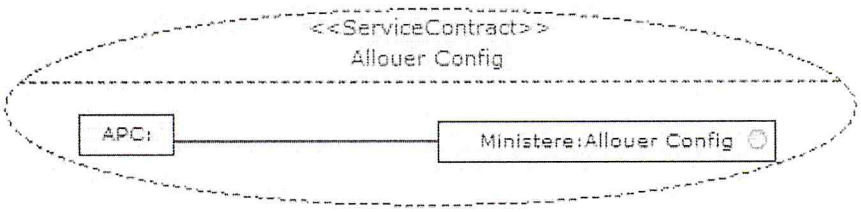


FIGURE 4.27 – Contrat du service Allouer configuration pour le client "APC".

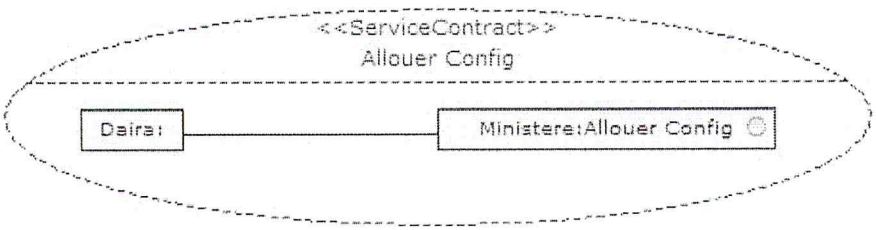


FIGURE 4.28 – Contrat du service Allouer configuration pour le client "Daïra".

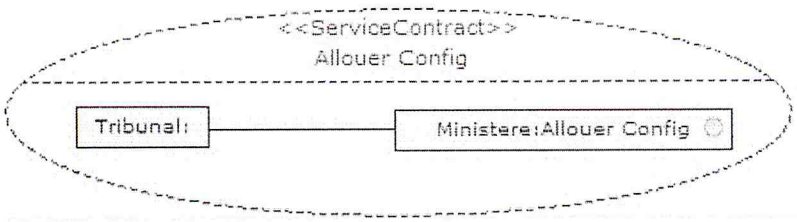


FIGURE 4.29 – Contrat du service Allouer configuration pour le client "Tribunal".

La figure 4.30 décrit le contrat du service gestion des comptes utilisateur.

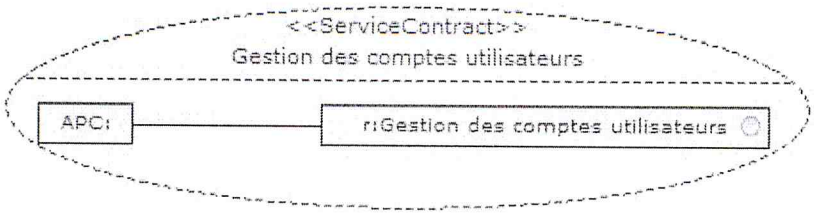


FIGURE 4.30 – Contrat du service gestion des comptes utilisateur.

La figure 4.31 fournit le contrat du service gestion des informations rapatriées.

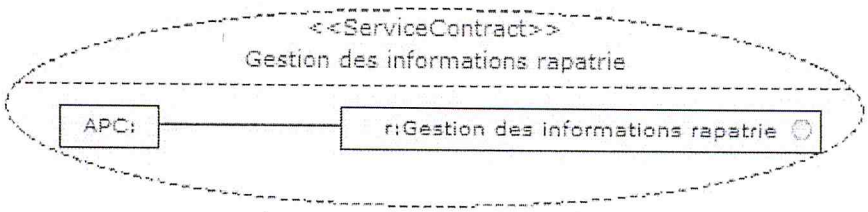


FIGURE 4.31 – Contrat du service gestion des informations rapatriées.

III. Modèle Architecture système

a. Diagramme d'interface du service : La figure 4.32 décrit en détail l'interface du service de localisation des entités.

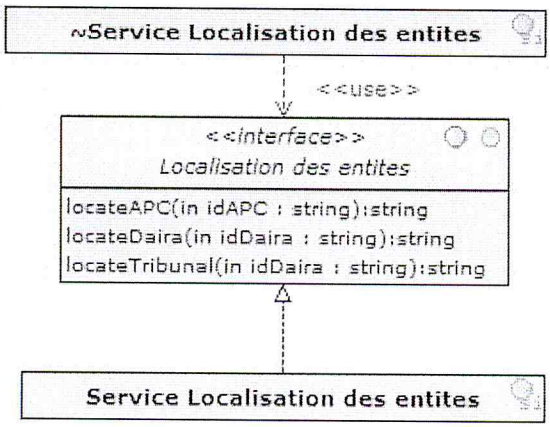


FIGURE 4.32 – Interface du service de localisation des entités.

La figure 4.33 représente l'interface du service d'allocation de configuration réseau.

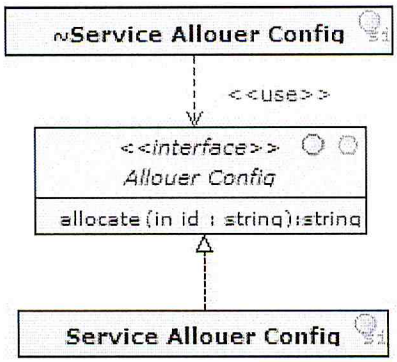


FIGURE 4.33 – Interface du service d'allocation de configuration réseau.

La figure 4.34 montre l'interface du service de gestion des comptes utilisateur.

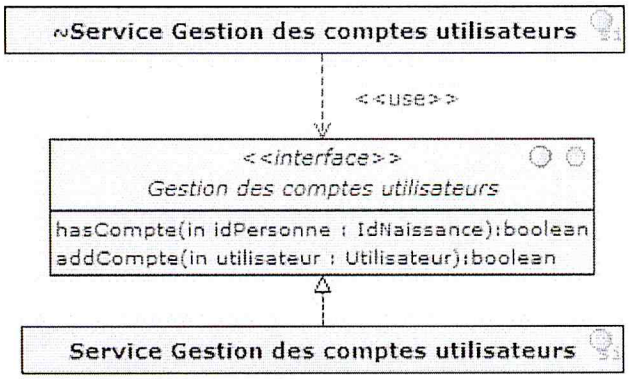


FIGURE 4.34 – Interface du service gestion des comptes utilisateur.

La figure 4.35 montre l'interface du service de gestion des informations rapatriées.

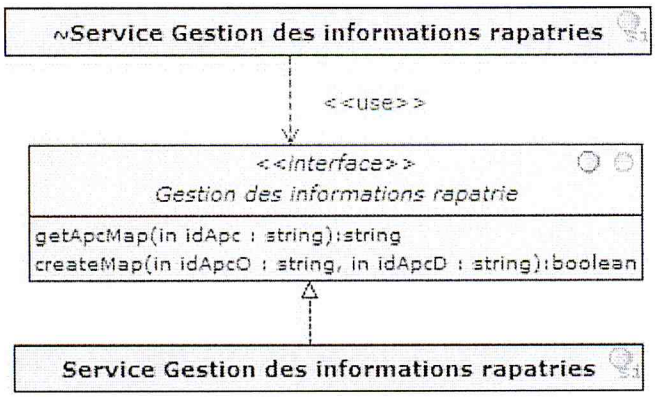


FIGURE 4.35 – Interface du service gestion des informations rapatriées.

b. Diagramme des messages : Vu que ces services ne manipulent que des chaines de caractères, nous n'avons pas des diagrammes de messages à établir.

c. Diagramme des participants : Au niveau de ce modèle, nous avons pu identifier plusieurs participants organisés par deux catégories comme suit :

- Fournisseur de services (Providers) : cette catégorie regroupe tous les participants qui offrent un service, ils sont :
  - Entité de retrait des actes :expose le service de Retrait des actes ;
  - Entité de gestion des avis de mentions : offre le service gestion des avis de mentions ;
  - Entité des demandes de naissance : expose le service de gestion des demandes de naissance ;
  - Processus des demandes de décès : expose le service de gestion des demandes de décès ;
  - Processus des demandes de mariages : expose le service de gestion des demandes de mariage ;
  - Processus des demandes de divorces : expose le service de gestion des demandes de divorce ;
  - Entité de rapatriement des données : expose le service de rapatriement des données ;
  - Entité de mise à jour : expose les services suivants :
    - Service de mise à jour des informations de naissance ;
    - Service de mise à jour des informations de mariage ;
    - Service de mise à jour des informations de divorce ;
    - Service de mise à jour des informations de décès.
  - Entité de restauration du système : expose le service de restauration du système ;
  - Configuration provider : expose le service d'allocation de configuration réseau ;
  - Entité de localisation des services : expose le service de localisation des entités ;
  - Entité de gestion des informations globales : fournit les services suivants
    - Service de gestion des informations rapatriées ;
    - Service de gestion des comptes utilisateurs.
- Client de services (Customer) :cette catégorie regroupe tous les participants qui demandent un service, ils sont :
  - Portail E-APC2.0 : c'est l'application E-APC 2.0, ce participant (ou composant) requiert les services suivants :
    - Service de Retrait des actes ;
    - Service de gestion des demandes de naissance ;
    - Service de gestion des demandes de décès ;
    - Service de gestion des demandes de mariage ;
    - Service de gestion des demandes de divorce ;
    - Service de localisation des entités ;
    - Service de mise à jour des informations de naissance ;
    - Service de mise à jour des informations de mariage ;
    - Service de mise à jour des informations de divorce ;
    - Service de mise à jour des informations de décès ;
    - Service de gestion des informations rapatriées ;
    - Service de gestion des comptes utilisateurs ;
    - Service de gestion des avis de mentions.
  - Wizard d'installation des services APC (S-APC) : c'est l'assistant de l'installation des services offerts par l'APC S-APC, ce composant requiert les services suivants :

- ◇ Service d'allocation de configuration réseau ;
- ◇ Service de gestion des informations rapatriées ;
- ◇ Service de rapatriement des données.

- Wizard de la restauration du système : c'est l'assistant de la restauration des données du système, il requiert les services suivants :
  - ◇ Service de localisation des entités ;
  - ◇ Service de restauration du système.
- Wizard d'installation des services Daïra (S-Daïra) : c'est l'assistant de l'installation des services proposés par la Daïra S-Daïra, ce composant requiert le service d'allocation de configuration réseau ;
- Wizard d'installation des services Tribunal (S-Tribunal) : c'est l'assistant de l'installation des services proposés par le Tribunal S-Tribunal, ce composant requiert le service d'allocation de configuration réseau.

Vu que la taille du diagramme des participants est grande, il est préférable de le diviser en plusieurs diagrammes, un diagramme par client de service.

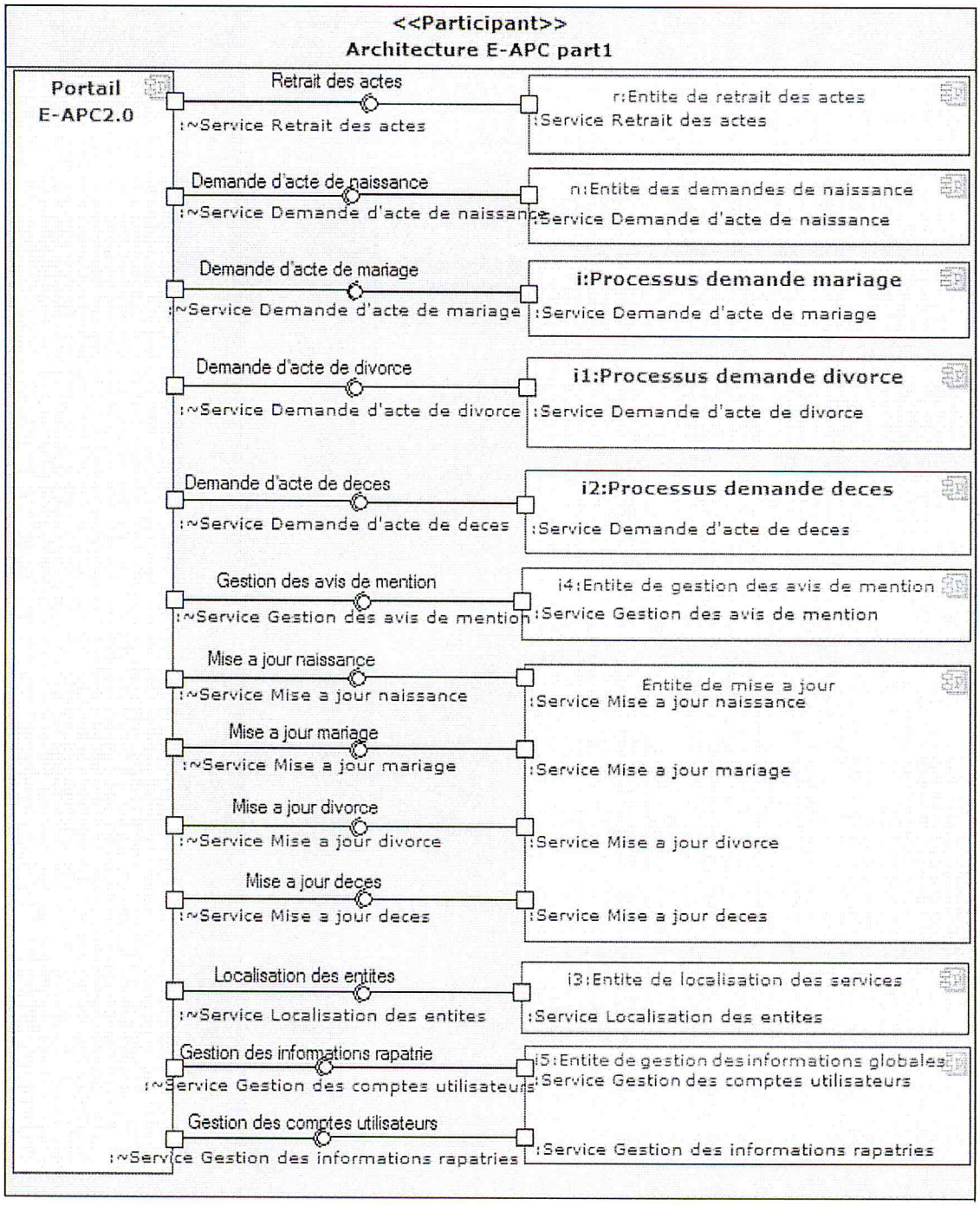


FIGURE 4.36 – Diagramme des composants du système (partie 1).

Remarque :
Les entités représentées dans la figure 4.36 en gras (Processus divorce et décès) sont détaillées par les diagrammes des figures : 27 et 28 du manuel.

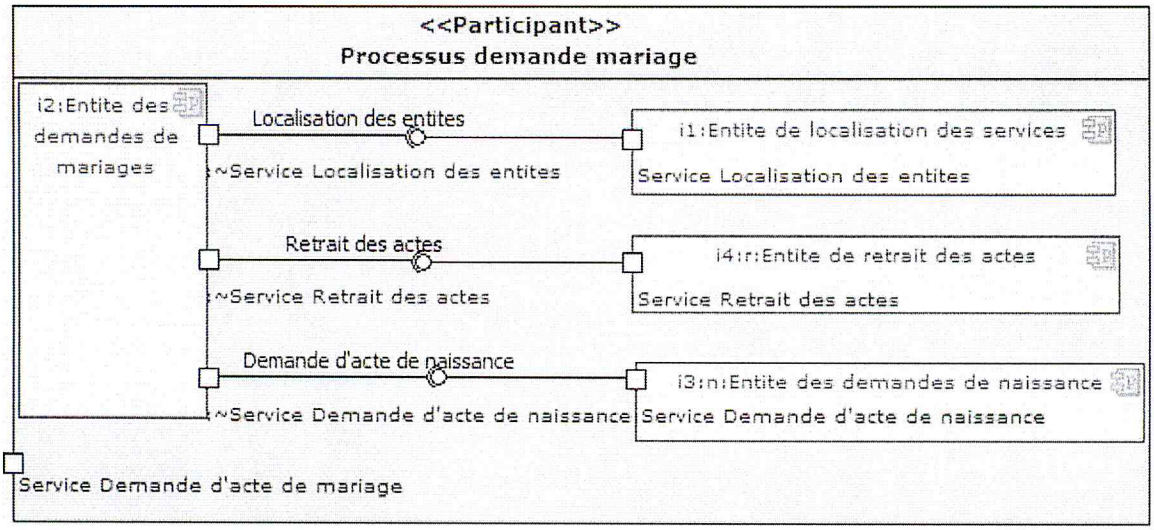


FIGURE 4.37 – Diagramme du participant "Processus demande mariage".

4.5 Adaptation de l'application E-APC 2.0

On a présenté jusqu'à maintenant la conception des services proposés, mais pour que le système de communication soit fonctionnel, l'application E-APC 2.0 doit gérer les appels aux services implémentés.

Afin d'atteindre cet objectif, deux solutions sont envisagées

- La modification du code source de l'application E-APC2.0 ;
- La réalisation d'un mécanisme (enveloppe) qui permet de modifier le comportement de l'application E-APC, autrement dit : l'ajout des nouvelles fonctionnalités (la gestion d'invocation des services).

Nous avons choisi de réaliser la deuxième approche. Ce choix est justifié par les raisons suivantes :

- L'application E-APC 2.0 est en cours de réalisation et la modification du code source entraine la mise à jour de tout le projet ;
- Assurer l'indépendance entre l'application E-APC 2.0 et le module qui gère l'invocation des services, ce qui facilite par la suite la maintenance du système.

La figure 4.38 schématise la solution adoptée :

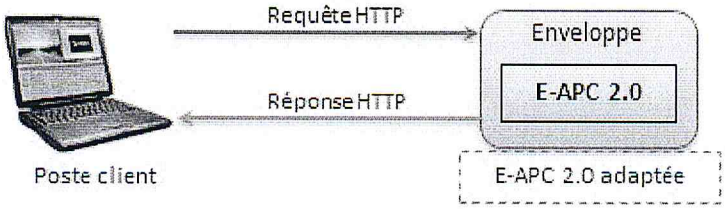


FIGURE 4.38 – Schéma de la solution d'adaptation de l'E-APC 2.0.

4.6 Conclusion

Nous avons présenté à travers de ce chapitre la solution proposée pour établir le système de communication. Nous nous sommes concentré tout d'abord sur la description détaillée des services définis en utilisant le langage SoaML, puis on a décrit la méthode d'adaptation et d'intégration de l'application E-APC 2.0 à notre système.

Ce chapitre nous a permis de tracer les squelettes des différentes classes à implémenter, ainsi l'identification des différentes fonctionnalités que doivent assurer les services Web.

Chapitre 5

Implémentation du système

5.1 Introduction

Après la présentation de l'architecture de notre système dans le chapitre précédent, nous allons, dans ce qui suit, décrire les différents aspects techniques liés à l'implémentation et le déploiement.

Pour mener à bien notre travail, nous avons fait appel à une panoplie d'outils et langages de développement permettant la réalisation et la mise en œuvre de notre application. Nous tenons à les présenter tout en argumentant nos choix avant d'entamer la description de notre système.

Nous décrivons ensuite les différentes couches composant notre système ainsi leurs interactions. On se concentre après sur la solution proposée pour l'intégration et l'adaptation de l'application E-APC au système développé.

Enfin, nous étalerons les différentes fonctionnalités offertes par le système réalisé. Cela sera illustré par des prises d'écrans permettant d'expliquer au mieux le fonctionnement du système.

5.2 Environnement de développement

Dans cette partie, on présentera l'environnement matériel et logiciel, ainsi que les outils de développement.

5.2.1 Environnement de développement matériel

L'application a été développée en utilisant un seul ordinateur (voir 5.1).

5.2.2 Environnement de développement logiciel

Langage de programmation

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

Caractéristique	valeur
Type de processeur	Intel® Core™2 Duo
Fréquence du processeur	2.93 GHz
Mémoire centrale	4 Go
Disque dur	500 Go
Système d'exploitation	Windows 7 32 bits

TABLE 5.1 – Environnement matériel.

- JAVA est un langage orienté objet simple ce qui réduit les risques d'incohérence.
- JAVA est portable. Il peut être utilisé sous Windows, sur Macintosh et sur d'autres plates- formes sans aucune modification. JAVA est donc un langage multi-plateforme, ce qui permet aux développeurs d'écrire un code qu'ils peuvent exécuter dans tous les environnements.
- JAVA 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.

Nous avons réalisé notre application Java en utilisant JDK 1.6.0_21.

Environnement de Développement Intégré

Notre choix d'environnement de développement s'est porté sur Eclipse EE 3.6.

Eclipse est un environnement de développement intégré (Integrated Development Environment - IDE) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques. Il est développé par IBM à partir de ses ancêtres Visual Age et Visual Age For Java. Il a été rendu open source et son évolution est maintenant gérée par la fondation Eclipse. Il est distribué depuis Septembre 2005 sous licence CPL (Common Public Licence).

Sa conception modulaire est basée sur un moteur de chargement de plugins et de différents plugins, ce qui fait d'Eclipse une boite à outils facilement améliorable ou modifiable. La licence d'Eclipse permet de fournir différents plugins : open source, closed-source, gratuits ou payants.
Nous avons opté pour Eclipse car il possède de nombreux avantages qui sont à l'origine de son énorme succès. Nous en citons les plus essentiels :

- Une plate-forme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug-ins ;
- Support de plusieurs plates-formes d'exécution : Windows, Linux, Mac OS X, ...
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT ;
- Une ergonomie entièrement configurable qui propose selon les activités à réaliser

différentes perspectives ;
- Le gestionnaire de mise à jour permet de télécharger de nouveaux plug-ins ou nouvelles versions d'un plug-in déjà installé à partir de sites web dédiés ;
- Eclipse EE possède WTP (Web Tool Application) comprenant des outils pour le développement des services web, éditeurs des fichiers XML et XML-Schema, etc... Par exemple : Web Service Explorer.

Apache Tomcat

Tomcat est un serveur web qui gère les servlets J2EE et les JSP. Il est souvent employé en combinaison avec un serveur web Apache : Apache s'occupe de toutes les pages web traditionnelles, et Tomcat uniquement des pages d'une application web Java. Issu du projet Jakarta, Tomcat implémente les spécifications des servlets et des JSP de Sun Microsystems. Il inclut des outils pour la configuration et la gestion, mais peut également être configuré en éditant des fichiers de configuration XML. Comme Tomcat inclut un serveur HTTP interne, il est aussi considéré comme un serveur HTTP, il est aussi une plateforme pour le développement et le déploiement d'applications web et des web services.
Les membres de la fondation Apache et des volontaires indépendants développent et maintiennent Tomcat. Les utilisateurs ont accès au code source et aux binaires sous Apache Software License.
Tomcat a été écrit en langage Java, il peut donc s'exécuter via la JVM (machine virtuelle java) sur n'importe quel système d'exploitation.
Nous avons réalisé notre application en utilisant Tomcat 6.0.

MySQL

MySQL est le serveur de base de données le plus utilisé dans le monde. Son architecture logicielle le rend extrêmement rapide et facile à personnaliser.
Les principaux avantages de MySQL sont sa rapidité, sa robustesse et sa facilité d'utilisation et d'administration. Un autre avantage majeur de MySQL est sa documentation très complète et bien construite.

Technologies des web services

a. Apache Axis 2 : Apache Axis2, est la troisième génération de moteur de web services, après ses prédécesseurs Apache SOAP et Apache Axis, non seulement il est plus efficace, modulaire, et orienté XML, mais il est aussi, flexible, extensible et il implémente des dispositifs puissants pour les entreprises, comme la sécurité et la fiabilité. Apache Axis2 est la meilleure implémentation open source des web services, ces trois dernière années.
Axis2 puise sa puissance et sa robustesse dans les quatre points suivants : meilleure performance, support de messagerie, meilleur support pour les extensions des web services, et un meilleur support de déploiement. Pour l'implémentation des services web, nous avons exploité deux plugins qu'on a installés dans l'environnement Eclipse et qui sont décrits ci-après.

b. Apache Axis2 service Archive Generator Wizard : Le générateur d'archives de service est un outil important qui permet la génération des archives d'un service

web (fichier aar ou jar) qui seront déployé comme un service web dans axis2.

c. **Apache Axis2 Code Generator Wizard** : C'est un autre outil qui permet de générer automatiquement d'une part le fichier wsdl à partir du code source d'un service (Java2WSDL) et d'autre part nous utilisons l'utilitaire WSDL2Java de ce plugin qui permet de générer à partir de la description wsdl d'un service les différentes classes et interfaces clientes nécessaire à l'appel de ce service côté client.

5.3 Déploiement du système

5.3.1 Architecture de l'application

Les figures 5.1 et 5.2 montrent l'organisation en couches du système (côté consommateur et fournisseur du service). Chaque couche communique avec une couche adjacente (celle du dessus ou celle du dessous). Et chacune utilise ainsi les services des couches inférieures et en fournit à celle de niveau supérieur.
L'intérêt de cette décomposition est de séparer le problème en plusieurs parties, chaque partie réalise une tâche bien précise. Selon la couche Service web, Métier ou DAO, l'application manipule deux types d'objets : les **Bean** et les **DTO**. Pour construire la première architecture, nous avons utilisé les patterns DAO/DTO. Ces patterns permettent de :

- Isoler la persistance des données ;
- Faciliter la modification du code qui gère la persistance (changement de SGBD ou même de modèle de données) ;
- Factoriser le code d'accès à la base de données ;
- Faciliter l'optimisation des accès à la base en les regroupant au sein d'objets particuliers ;
- Rendre la couche métier réutilisable et stable ;
- Transporter plusieurs objets distants en un seul appel ; par exemple un acte de naissance avec des informations sur la personne, ses parents, etc.

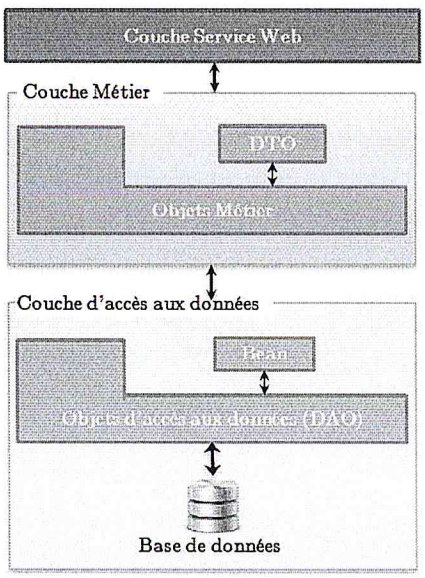


FIGURE 5.1 – Architecture du Service "Vision en couche".

Couche d'accès aux données

Cette couche s'occupe de l'accès à la base de données. Les objets manipulés à ce niveau sont les Bean.

a. Objets d'accès aux données : Les objets d'accès aux données s'occupent de persister les objets Bean. Ils réalisent toutes les fonctionnalités concernant l'accès aux données, i.e. les opérations CRUD (Create, Retrieve, Update et Delete), la connexion à la base, l'exécution des requêtes, etc.

b. Bean : Les objets Bean sont les objets persistants de l'application (communicant avec la base de données). Ces objets permettent de :
- o récupérer les informations de la base de données ;
- o mettre à jour la base de données.

Couche métier

S'occupe de la logique métier de l'application. Cette couche est indépendante de toute forme d'interface avec l'utilisateur. Ainsi elle doit être utilisable aussi bien avec une interface console, une interface web, une interface de client riche. C'est la couche la plus stable de l'architecture. Elle ne change pas si on change l'interface utilisateur ou la façon d'accéder aux données nécessaires au fonctionnement de l'application.

a. Objets métier : Ils implémentent la logique métier, ces objets utilisent la couche d'accès aux données pour stocker et récupérer des informations de la base.

b. DTO : Les objets DTO (Data Transfert Object) ont pour but de transporter les informations métier complexes (combinaison de plusieurs Bean) et d'assurer la communication entre le consommateur et le fournisseur du service. Cette communication s'effectue dans les deux sens :
- o Bean -> DTO, dans le cas d'une opération de récupération des données. Dans ce cas-ci, le DTO sera alimentée à partie d'un ou plusieurs Bean. En effet un Bean a une structure qui correspond à une table, et une DTO a une forme qui rassemble à un ensemble de table. Cette phase d'assemblage est assurée par la couche Métier.
- o DTO -> Bean, dans le cas où la base de données doit être mise à jour par les données provenant du client (consommateur du service). La couche Métier transforme la DTO en Bean où plusieurs Bean (dans le cas où plusieurs tables doivent être mises à jour) et fait appel à la couche d'accès aux données.

Chaque classe respecte les spécifications fonctionnelles tant au niveau de ses champs que des propriétés, mais n'implémente aucune logique métier au niveau des méthodes. En effet, il ne s'agit là que d'un conteneur permettant le transport des informations métier.

Couche Services web

Est la couche la plus importante de toute l'architecture. Celle-ci constitue l'interface entre l'utilisateur et l'exécution des demandes. Elle est composée des classes qui implémentent l'interface du service, donc elle fournit les opérations nécessaires qui assurent la communication entre les différentes entités du système.

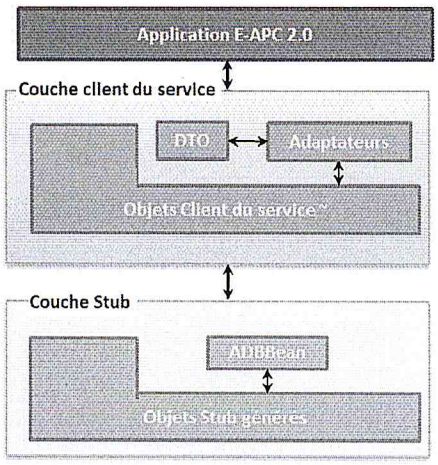


FIGURE 5.2 – Architecture du Client "Vision en couche".

Couche Stub

Cette couche s'occupe de gérer l'accès aux ressources réseaux. Elle réalise la sérialisation en XML, l'envoi, la réception et la construction des messages SOAP.

a. Objets Stub : Les objets Stub sont générés par le plugin **Apache Axis2 Code Generator Wizard** (présenté dans la section précédente) à partir des fichiers WSDL. Ces objets implémentent des interfaces spécifiques à Axis 2 pour gérer proprement l'envoi, la réception la construction des messages SOAP.

b. ADBBean : Ils représentent les types d'objets échangés entre le fournisseur et le consommateur du service. Ces classes implémentent l'interface **ADBBean** (fournit par Axis 2) dans le but de sérialiser (dé-sérialiser) les données à envoyer (recevoir).

Couche client du service

Elle constitue l'intermédiaire entre le consommateur du service (E-APC 2.0) et la couche Stub, cette couche joue un rôle très important, elle réalise les adaptations nécessaires et gère l'appel au service web en utilisant les services de la couche Stub.
L'intérêt de la centralisation des opérations d'invocation du service est de **rendre transparent** à l'application E-APC 2.0 (ou toute autre application bénéficiant du service) les procédures d'appel du service.

a. Objets client du service : Ils ont pour but de gérer l'invocation du service, ces objets offrent les mêmes opérations que le service lui-même, et ils implémentent l'interface conjugué (marquée généralement par le caractère , voir chapitre précédant section diagramme d'interface du service). Ces objets collaborent avec les classes **Adaptateurs** pour gérer deux types de conversions :

- DTO -> ADBBean : dans le cas où l'application E-APC 2.0 invoque un service. Les objets client du service transforment d'abord les DTO en ADBBean par les Adaptateurs, puis ils font appel à la couche Stub pour envoyer un message SOAP au service Web.
- ADBBean -> DTO : dans le cas où le service Web envoie une réponse, à sa réception, les objets client du service récupèrent les ADBBean fournis par la

couche Stub, puis les transforment en DTO par les Adaptateurs, et enfin le résultat obtenu est ils transféré à l'application E-APC 2.0.

b. Adaptateurs Ils réalisent deux types de conversions : DTO -> ADBBean et ADB-Bean -> DTO (ces deux transformations sont expliquées en détails dans la section précédente).

La figure 5.3 montre les étapes d'exécution de la procédure d'invocation d'un service, et schématise l'interaction entre les différentes parties de chaque couche, ainsi les objets échangés durant chaque étape.

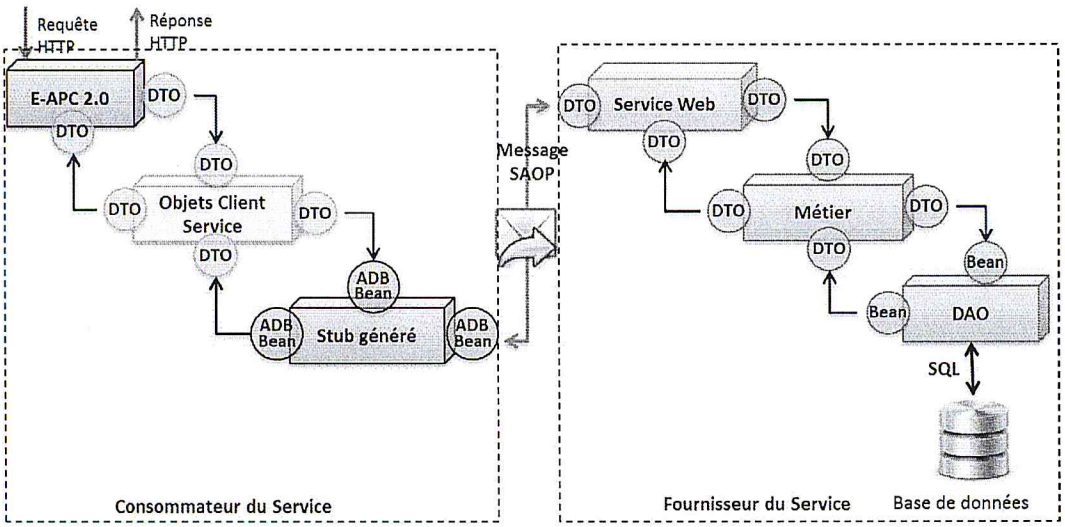


FIGURE 5.3 – Interactions entre le fournisseur et le consommateur du service.

5.3.2 Adaptation de l'application E-APC 2.0

Dans cette section, nous détaillerons la solution choisie pour l'adaptation de l'application E-APC. Le Framework utilisé pour développer le système E-APC offre un mécanisme d'**interception** qui facilite la réalisation de l'approche adoptée.

Intercepteurs Struts : Les intercepteurs sont des filtres qui permettent de réaliser des tâches afin de simplifier et d'améliorer le travail des développeurs. Dans la plupart des cas, les intercepteurs proposés en standard par Struts sont suffisants et il est essentiel de comprendre l'intérêt et l'apport de chacun d'eux [38]. Les intercepteurs les plus utilisés sont présentés ci-dessous.

params : Gestion du mapping entre les paramètres des requêtes et les propriétés des actions ;

servletConfig : Gestion de l'accès aux classes HttpServletRequest et HttpServletResponse ;

execAndWait : Gestion des chargements de pages lors des traitements plus ou moins longs ;

debugging : Gestion du débogage des applications ;

exception : Gestion du traitement des exceptions ;

store : Gestion des portées des messages de succès et d'erreurs ;

chain : Gestion du chaînage des actions et du passage de paramètres.

Nous proposons de réaliser un **intercepteur personnalisé** entre le filtreur Struts (*FilterDispatcher*) et l'ensemble des actions. Celui-ci permet de :

- Détecter les points d'appel aux services Web ;
- Rediriger les requêtes vers d'autres classes qui gèrent l'invocation des services Web.

La figure 5.4 montre l'architecture de la solution d'adaptation proposée. Cette architecture a été inspirée du schéma architectural Struts 2 présenté dans [38].

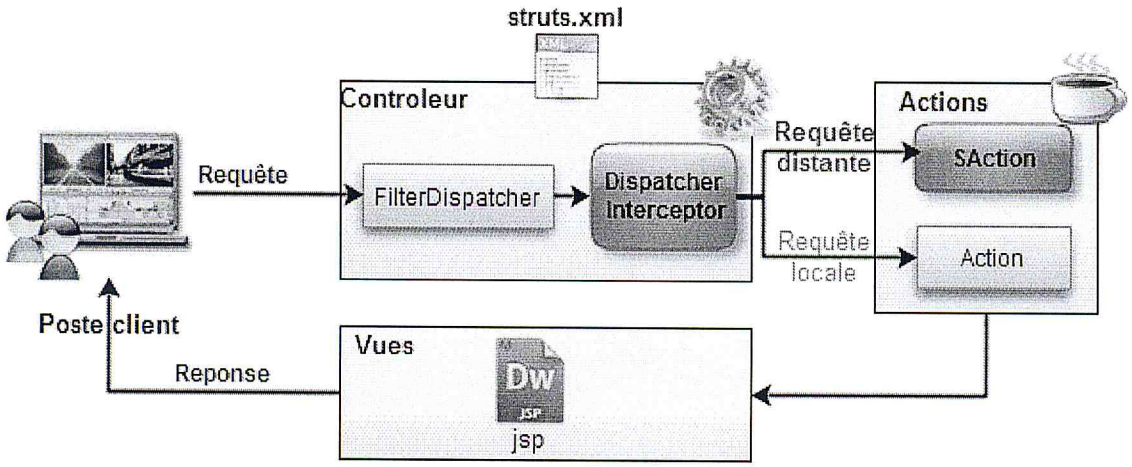


FIGURE 5.4 – Architecture de l'application E-APC 2.0 adaptée

1. Le client envoie des requêtes vers l'application E-APC 2.0 ;
2. Le fichier de configuration (**struts.xml**) de l'application est consulté ;
3. Le **Dispatcher** traite la requête :
   - Si la requête nécessite l'invocation d'un service Web
     - Rediriger la requête vers **SAction** ;
   - Sinon
     - Rediriger la requête vers Action ;
4. La classe d'action transmet les données nécessaires à la vue ;
5. La vue affiche au client les résultats traités.

5.3.3 Sécurisation des services Web

Sécurité des échanges

D'après l'étude présentée dans le chapitre deux, nous avons décidé de sécuriser les échanges des données en utilisant le protocole SSL.

Le moteur des services Web utilisé dans ce projet (Axis 2) ne gère pas la communication au niveau transport, en effet cet aspect est assuré par le conteneur du moteur service Web

(i.e. Tomcat), donc afin de sécuriser convenablement nos services, il faut tout d'abord activer l'utilisation du protocole SSL au niveau du conteneur.
Pour implémenter la sécurité proposée, nous avons suivi les étapes suivantes :

Activation du SSL au niveau du conteneur

Etape 1 : Génération du fichier keyStore Un keyStore ou réserve de clef, est un conteneur de clefs secrètes, de paires de clefs publique/privée et de certificat attestant la validité d'une clef publique [38] . Le JRE (Java Runtime Environment) offre l'outil **keytool** pour gérer la génération des keyStore.

La commande suivante permet de produire un keyStore :

```
Keytool -genkey -keypass testPW -keystore JKS.jks -storepass testPW
```

Tel que :

-genkey : génère une paire de clefs et un certificat X.509 v1 pour la clef publique ;

-keypass keypass : fournit le mot de passe utilisé pour protéger la clef privée d'une entrée de clef ;

-keystore keystore : spécifie le fichier de mémoire de clefs à utiliser ;

-storepass storepass : spécifie le mot de passé utilisé pour protéger l'intégrité de la mémoire de clefs ;

Jks : Java Key Store.

Etape 2 : Configuration Tomcat La configuration du Tomcat consiste à modifier le fichier **server.xml** de la manière suivante :

```
<Connector protocol= "org.apache.coyote.http11.Http11Protocol"
 port="8443" maxThreads="200" scheme="https" secure="true"
 SSLEnabled="true" keystoreFile="webapps/JKS.jks" keystorePass="testPW"
 clientAuth="true" sslProtocol="TLS" />
```

Restreindre l'accès aux services qu'aux clients authentifiés correctement Afin de limiter l'accès aux services, il est nécessaire d'ajouter ces balises au niveau du descripteur de déploiement (**web.xml**)

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name> Nom Application </web-resource-name>
        <url-pattern> /* </url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <user-data-constraint> CONFIDENTIAL <user-data-constraint>
    </user-data-constraint>
<security-constraint>
```

Réajustement de la couche client Pour que la **couche client** (voir figure 5.2) développée supporte les appels SSL aux services Web, il est nécessaire d'ajouter ce bout de code avant chaque invocation du service :

```
System.setProperty("javax.net.ssl.trustStore", "JKS.jks");
System.setProperty("javax.net.ssl.trustStorePassword", "testPW");
```

Sécurité d'installation

Pour assurer la sécurité de l'infrastructure d'une APC, nous proposons de construire une zone **DMZ** (**D**e**M**ilitarized **Z**one), cette situation est schématisée par la figure 5.5 :

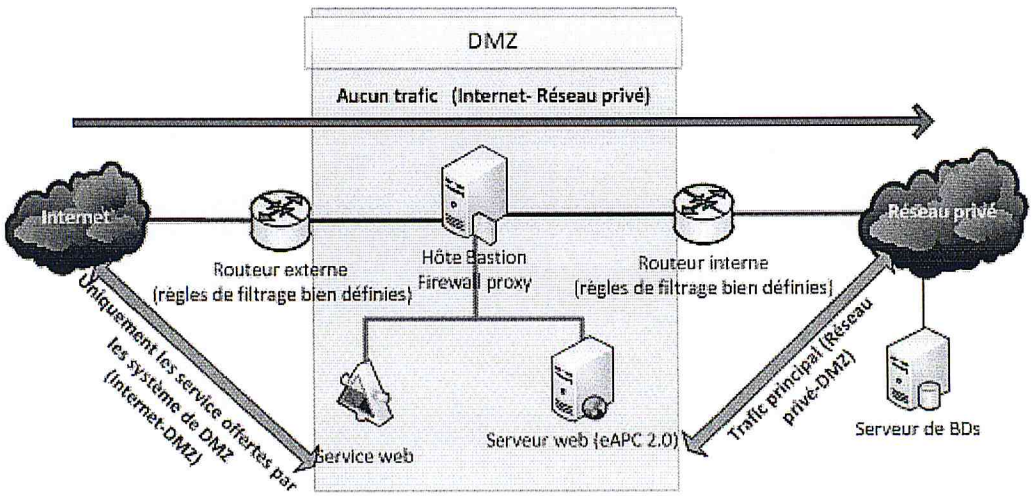


FIGURE 5.5 – Infrastructure de sécurité pour une APC.

DMZ est une zone du réseau semi-protégée. Cette zone est normalement délimitée par des contrôles d'accès au réseau tels que des Firewalls ou des routeurs dont les filtres sont finement paramétrés. Généralement tous les systèmes auxquels un utilisateur externe peut accéder doivent être placés dans la DMZ [40].

5.3.4 Diagramme de déploiement

La figure représente de diagramme de déploiement des services.

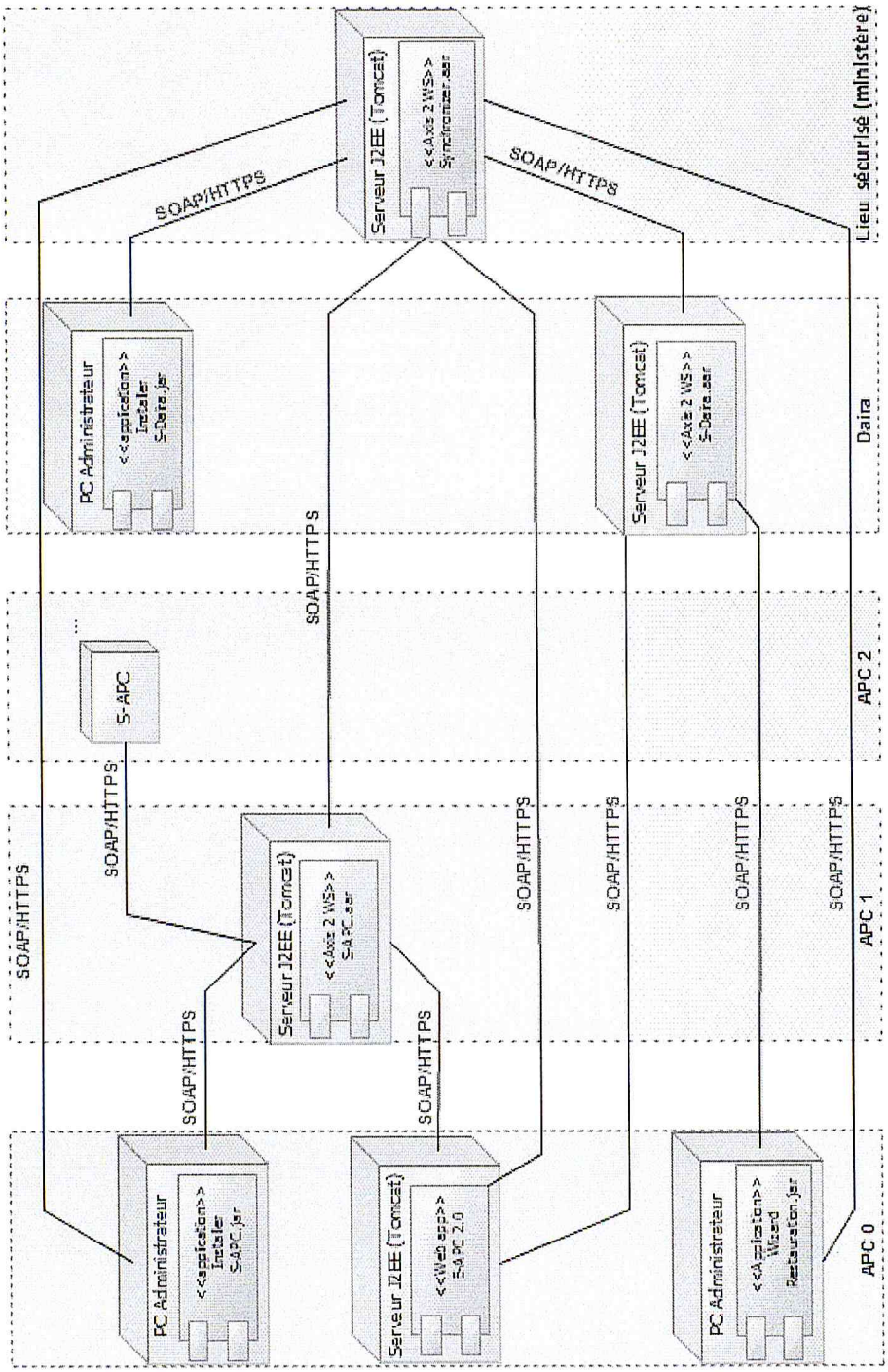


FIGURE 5.6 – Diagramme de déploiement.

Remarque : Chaque nœud est lié à un serveur de base des données, cette situation est représentée par la figure 5.7 :

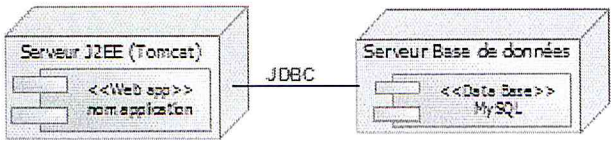


FIGURE 5.7 – Diagramme de déploiement d'un service.

5.3.5 Présentation du système

Le système développé a subi deux types de tests :

Tests unitaires : chaque couche développée a été testée séparément ;

Tests d'intégration : ce type de tests est établi après l'intégration de l'application E-APC 2.0 au système de communication.

Dans ce qui suit, nous présenterons quelques tests d'intégration en utilisant trois ordinateurs configurés comme suit :

PC 1 : héberge le service **Synchronizer** ;

PC 2 : héberge **E-APC Adrar** et les services suivants

- S-Daira Blida ;
- S-APC Adrar.

PC 3 : héberge **E-APC Blida** et les services suivants

- S-Daira Adrar ;
- S-APC Blida.

Afin d'expliquer au mieux le fonctionnement du système, les tests d'intégration seront illustrés par des captures d'écrans.

Page d'accueil

La figure 5.8 montre la nouvelle interface de l'application E-APC 2.0.

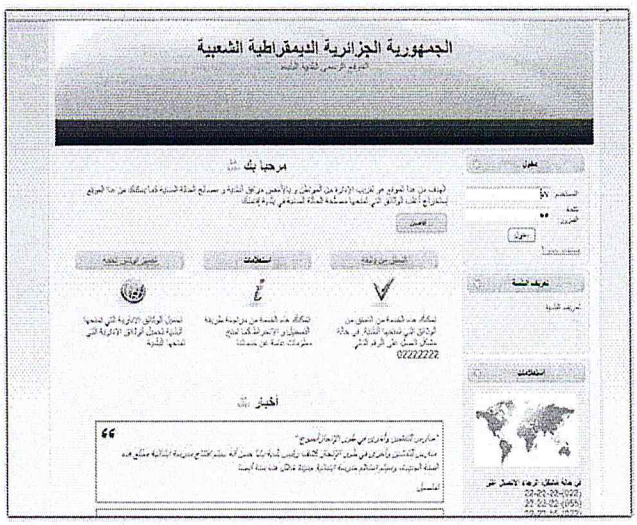


FIGURE 5.8 – Nouvelle interface de l'application E-APC 2.0.

Assistant d'installation

La figure 5.9 montre l'interface de l'assistant d'installation des services APC. L'administrateur doit fournir des informations concernant la base des données (nom d'utilisateur, mot de passe, etc.), le chemin du conteneur des services (cet assistant gère seulement le conteneur Tomcat) et l'adresse du service Synchronizer afin de réussir l'opération d'installation.

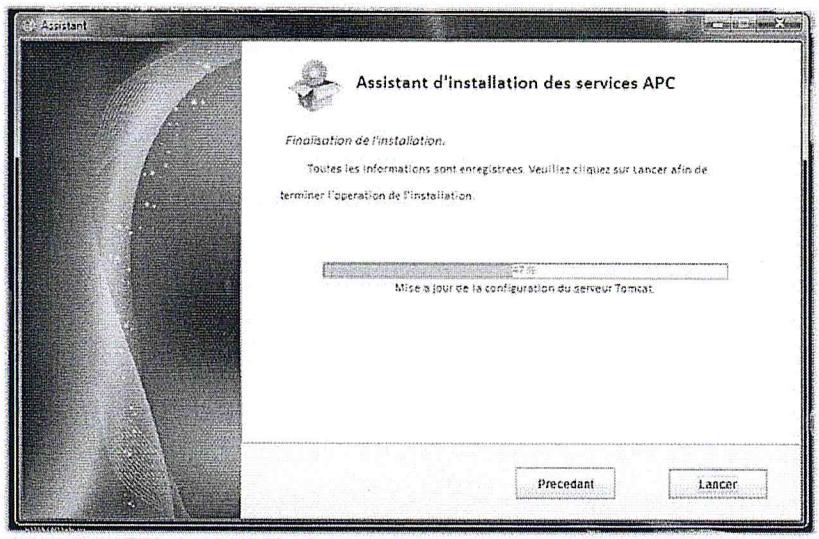


FIGURE 5.9 – Assistant d'installation des services APC.

Assistant de restauration d'informations

La figure 5.10 représente l'assistant de restauration d'informations. Cette opération nécessite le saisi d'informations concernant la base des données ainsi que l'adresse du service Synchronizer.

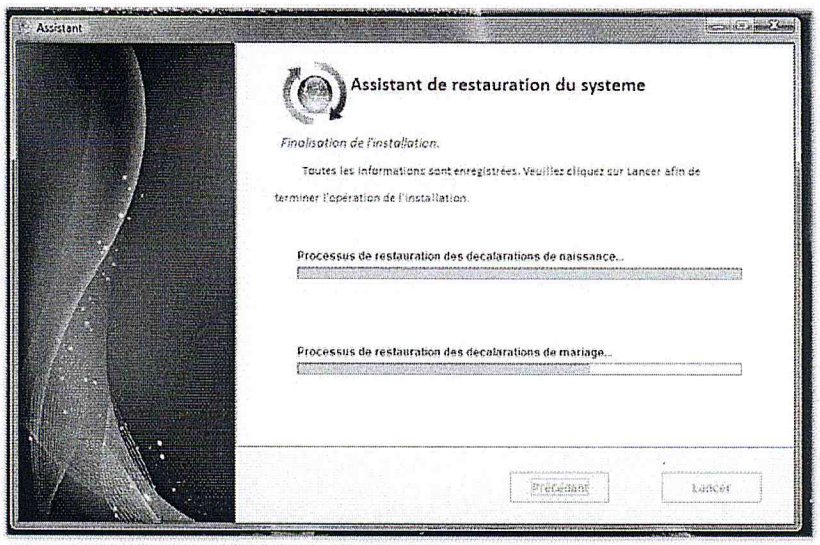


FIGURE 5.10 – Assistant de restauration d'informations.

Assistant de rapatriement des données

La figure 5.11 illustre l'interface de l'assistant de rapatriement des données, cette tâche requière les mêmes entrées que l'opération de restauration d'informations.

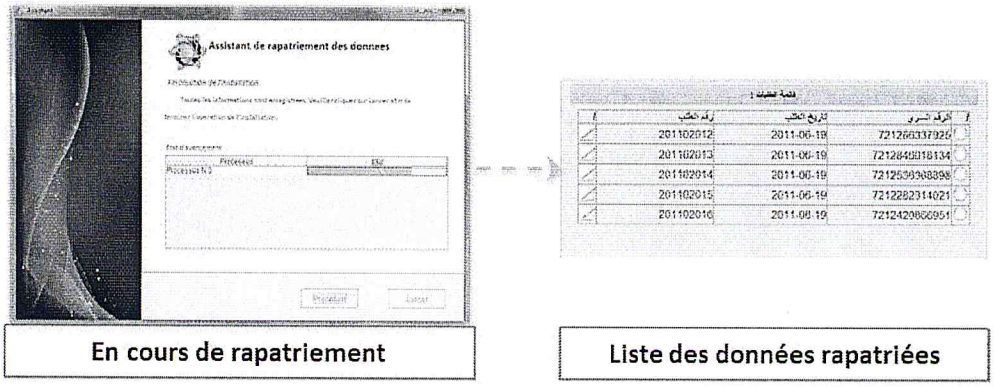


FIGURE 5.11 – Assistant de rapatriement des données.

Processus de déclaration d'un évènement

La figure 5.12 montre les étapes du processus de déclaration d'un évènement de mariage à l'APC de **Blida**, tel que : L'époux et l'épouse sont nés à **Adrar**.

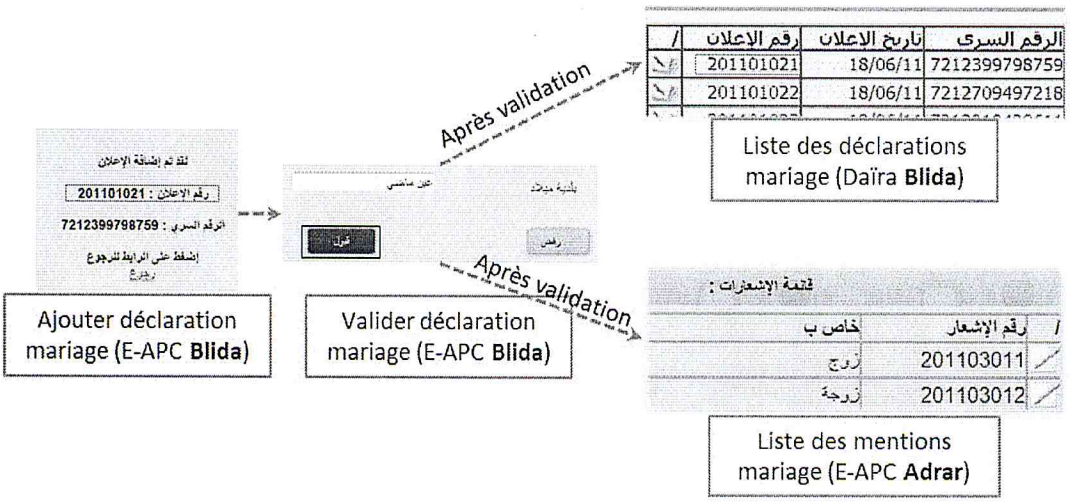


FIGURE 5.12 – Étapes de déclaration d'un évènement de mariage.

Processus de retrait d'un acte de naissance

La figure 5.13 représente le processus de retrait d'un acte de naissance d'une personne née à **Blida**, à partir de **l'E-APC d'Adrar**.

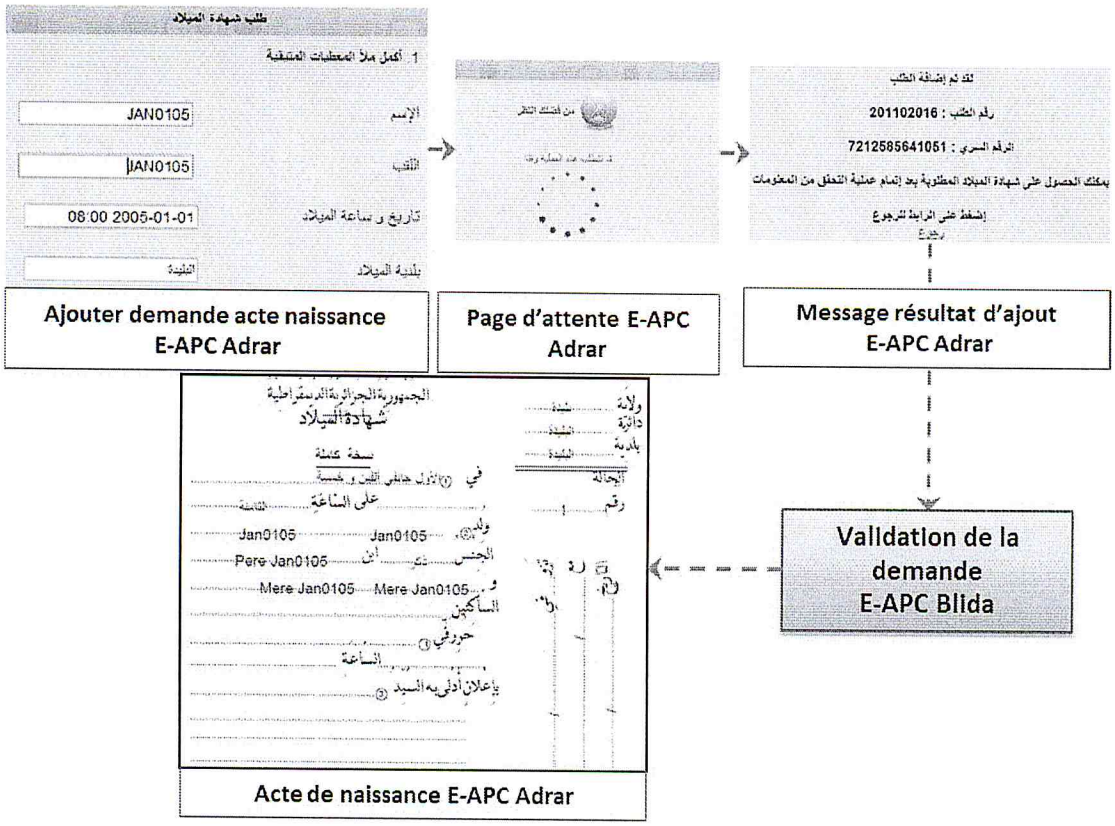


FIGURE 5.13 – Processus de retrait d'un acte de naissance.

Si l'ajout d'une demande nécessitera beaucoup de temps (plus de 5 secondes), l'utilisateur est redirigé vers une page d'attente avant l'affichage du message résultat (succès ou échoue de l'opération).

5.4 Conclusion

Dans ce chapitre, nous avons détaillé la projection de la conception du plan théorique sur le plan pratique tout en montrant la fidélité de notre application produite par rapport à la modélisation.

La phase du test a durée plus de **40%** du temps de développement du système, car il est très difficile de tester des applications réparties qui collaborent au sein d'une architecture orientée services. Grâce à cette étape, on a assuré que les services Web réalisés fonctionnent convenablement et produit les résultats prévus.

Conclusion générale et perspectives

E-APC 2.0 est un système d'e-Gouvernement, qui a été conçu dans le but de mettre sur Internet d'une part les divers services d'une APC fortement sollicités par le citoyen et d'autre part permettre la participation active du citoyen à la gestion de sa commune. La version actuelle de l'E-APC a été conçu pour fonctionner sur une seule APC, et elle ne dispose d'aucune fonctionnalité de communication permettant la collaboration ou l'échange des données entre les instances de l'E-APC déployés à travers les diverses APC du territoire national.

Le travail présent apporte une solution pour résoudre ce problème. Nous avons proposé de réaliser une architecture orientée services bâtie sur les différents participants du système : les APCs, les Daïras, les Tribunaux et le Ministère de l'intérieure. L'architecture globale de l'application est en étage (niveau ou couche). Chaque type de participant se positionne à un niveau bien précis de cette architecture et expose et consomme des divers services à travers la technologie des services Web.

Dans la version réalisée, nous distinguons trois niveaux, en allant de plus bas vers le plus haut :

Le premier niveau est composé d'APCs : expose des services de mise à jour, de retrait de documents officiels (actes, fiches) rapatriement des données, etc. ;

Le deuxième niveau est composé des Daïras et des Tribunaux : chaque entité fournit des services de restauration et de mise à jour du système ;

Le troisième niveau est formé par le Ministère des collectivités locales : offre des services particuliers notamment ceux permettant la synchronisation des diverses instances, leur configuration, etc.

Vu que l'application E-APC a été développée indépendamment du système de communication et selon une approche orientée composant, il est devenu nécessaire de l'adapter à ce nouvel environnement. C'est ainsi que dans ce contexte nous avons réalisé une **enveloppe** d'adaptation, basée sur le FrameWork **Struts 2 d'Apache**. Cette enveloppe (ou **Wrapper**) intercepte chaque requête et détermine en fonction de ses paramètres, si cette requête doit être gérée localement ou nécessitera l'intervention d'un service distant. Dans ce cas, cette enveloppe transfert le contrôle à une autre partie du système qui gère l'invocation des services adéquats.

Notre projet a atteint ses objectifs, et les exigences mises en départ ont été satisfaites, ainsi que sur le plan livrable, nous avons pu réaliser :

- √ Un système de communication qui assure :
  - La cohérence et la mise à jour du système complet ;
  - La génération automatique des avis de mentions ;
  - La restauration des données lors d'un incident ;
  - L'augmentation de taux de livraison instantanée des actes de l'état civil.
- √ Une adaptation de l'application E-APC 2.0 ;
- √ Des assistants (Wizards) de restauration/rapatriement des données ;
- √ Des assistants d'installation et d'initialisation des services développés.

Le système de communication réalisé met en œuvre plus de **800** classes, **120** interfaces, et gère plus de **75** tables au niveau APC, **13** tables au niveau Daïra et **10** tables au niveau du site contenant le Synchroniser.

Il est à noter que ce projet nous a permis :

Sur le plan technologique

- D'améliorer profondément nos connaissances en programmation orientée objet en Java, et de maitriser la mise en œuvre de la technologie des services Web sous Axis 2 d'Apache ;
- De comprendre et de mettre en œuvre des patrons de conception (Design Pattern) tels que le Singleton, le Factory, le Proxy, etc. ;
- D'approfondir nos connaissances en conception par l'apprentissage d'un nouveau langage de modélisation **SoaML**.

Sur le plan recherche, de découvrir de nouveaux domaines de recherche tels que les systèmes répartis et les architecture orientée services.

Il est à noter que notre travail n'a pas été réalisé sans difficultés remarquables. C'est ainsi que lors de la réalisation de ce projet, nous avons fait face à plusieurs problèmes, qui parfois nous ont mené à refaire certaines parties de l'E-APC 2.0. Ces problèmes peuvent se résumer par les points suivants :

- Manque de la documentation sur le langage SoaML ;
- Difficulté de réutiliser le code source de l'application E-APC 2.0 ;
- Incohérence entre la conception et la réalisation de l'application E-APC 2.0 ;
- Présence de plusieurs défauts (bugs) non encore solutionnés au niveau de l'application E-APC 2.0.

Perspectives

Afin de compléter ce travail, on peut envisager quelques travaux futurs :

- ⋆ Amélioration de la stratégie de la sécurité adoptée en lui associant d'autres techniques de sécurité telles que *WS-Security*, *WS-Trust*, *WS-Policy*. **Nous rappelons ici que l'aspect sécurité n'est pas un objectif dans le contexte du travail qui nous a été assigné ;**
- ⋆ La réalisation d'un composant de suivi des responsabilités dans le système global ;
- ⋆ L'intégration des nouveaux partenaires, par exemple e-Health, e-Justice, etc. ;
- ⋆ Réalisation d'une interface graphique pour la configuration des services ;
- ⋆ Ajout de site redondant pour rendre le système global plus robuste : le système actuel est un arbre ayant une seule racine, dans le cas où cette racine devient indisponible le système tout entier pourrait en souffrir partiellement, le doublage de la racine par un nœud redondant pourrait permettre l'entrée en action de ce dernier en cas d'indisponibilité de la racine.

Annexe A

Cette Annexe décrit le langage et la méthode **SoaML** utilisés pour la conception des services étudiés dans le chapitre quatre de ce mémoire.

A.1 Présentation du langage SoaML

SoaML est une nouvelle spécification présentée par l'organisation **OMG** (**O**bject **M**anagement **G**roup) en 2009 [33]. Le but de SoaML est de fournir aux utilisateurs du langage UML (Unified modeling language) les moyens de modéliser une architecture orientée services comprenant des notions de **consommateurs** et de **fournisseurs** de services, ainsi que la notion de **contrats**. SoaML se base sur un ensemble de concepts que nous présentons brièvement dans ce qui suit :

A.1.1 Participants

Ils sont des entités qui fournissent ou utilisent des services. Ces entités peuvent représenter des personnes, organisations ou des composants d'un système d'information. Ils ont des ports de service qui sont les points de connexion où les services sont effectivement fournis ou consommés.

A.1.2 Ports

Les participants fournissent et consomment des services via des ports. Un port représente le point d'interaction où le service est consommé ou fournit. Il est associé à une interface requise (marquée par le stéréotype **«request»**) ou offerte (désignée par le stéréotype **«service»**).

A.1.3 Contrat de service

Le contrat de service représente un accord entre les participants concernés pour savoir comment le service doit être fourni et consommé. Il a pour but de définir les rôles joués par chaque participant dans le service (fournisseur et consommateur) et les interfaces qu'ils mettent en œuvre pour jouer ce rôle.

A.2 Diagrammes du langage SoaML

Afin de modéliser une application orientée service, SoaML fournit plusieurs diagrammes qui sont :

- Diagramme d'architecture des services (Service Architecture) ;
- Diagramme de processus métier (Business Process) ;
- Diagramme des Capabilités (Capabilities) ;
- Diagramme de contrats des services (Service Contracts) ;
- Diagramme d'interface des services (Service Interface) ;
- Diagramme des messages (Message Diagram) ;
- Diagramme des chorégraphies du service (Service Choreographies) ;
- Diagramme des participants (ou des composants).

Selon l'auteur de l'article [34], les diagrammes Capabilités et Chorégraphies du service sont optionnels. Dans ce qui suit, on ne détaillera que les diagrammes les plus utilisés.

Les exemples fournis dans cette Annexe font partie de l'étude du cas **Travel Agency (Discount Voyage)** présentée dans l'article [34] .

A.2.1 Diagramme de processus métier

Ce diagramme décrit les processus métiers pertinents du domaine dont l'architecture orientée service fait partie. Il a pour objectif de proposer un moyen simple et visuel de communication entre les différents intervenants chargés de la réalisation du projet [34].

La spécification SoaML propose d'utiliser la notation **BPMN** (Business Process Modeling Notation) pour réaliser ce type de diagramme.

A.2.2 Diagramme de contrats des services

Ce diagramme est dédié à la modélisation des contrats de service qui ont été mentionnés dans l'architecture des services, en définissant pour chaque contrat le fournisseur et le consommateur d'un tel service [34].

Exemple

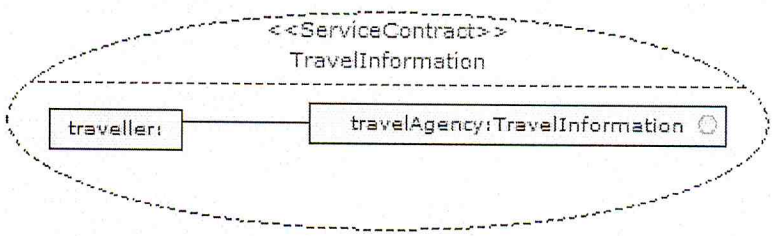


FIGURE A.1 – Diagramme Contrat des services [34].

A.2.3 Diagramme d'architecture des services

Ce diagramme est dédié à la modélisation d'une architecture orientée service, en spécifiant les différents participants et les contrats de services qui les relient [34].

Étape de modélisation

Pour construire une architecture de service avec SoaML, il faut suivre les étapes suivantes :

- Identifier les participants ;
- Identifier les contrats de service ;
- Lier les participants aux contrats de service.

Exemple

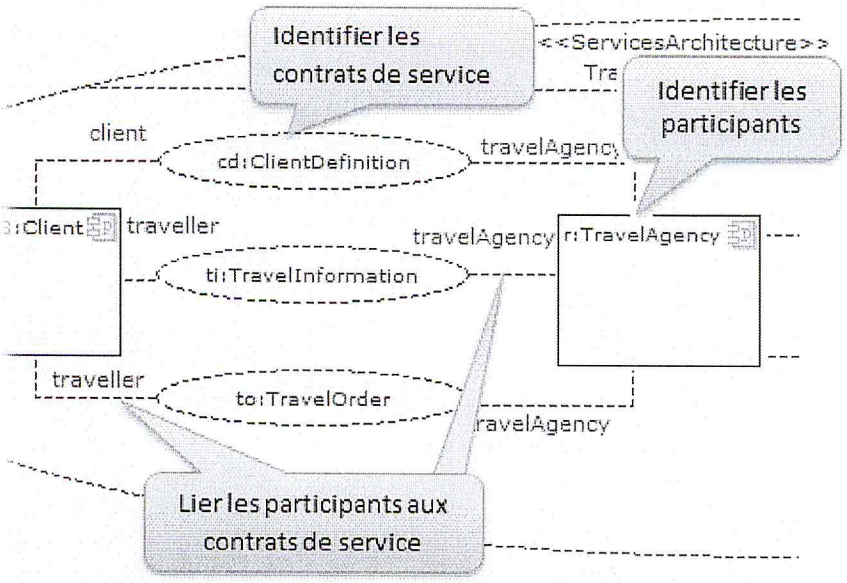


FIGURE A.2 – Diagramme d'architecture des services [34].

A.2.4 Diagramme d'interface des services

Ce diagramme définie les interfaces et les responsabilités d'un participant afin de fournir ou consommer un service. Il représente le résultat du raffinement de diagramme contrat de service [34].

Étapes de modélisation

Afin de modéliser une interface de service avec SoaML, il faut suivre les étapes suivantes :

- Définir l'interface de service à partir des contrats ;
- Définir l'interface du fournisseur avec ses opérations ;

- Définir l'interface du consommateur (L'interface du consommateur est définie lorsque des appels **CallBacks** sont nécessaires) ;
- Relier l'interface du fournisseur et du consommateur à l'interface de service comme suit :
  - Créer un lien de réalisation entre l'interface du fournisseur et l'interface de service ;
  - Créer un lien d'utilisation entre l'interface du consommateur et l'interface de service.
- Créer et relier l'interface conjuguée du service comme suit :
  - Créer l'interface conjuguée du service qui utilise l'interface du fournisseur et réalise l'interface du consommateur. A noter que le nom de l'interface conjuguée du service est précédé par «˜» qui représente le contraire de l'interface de service.

Exemple

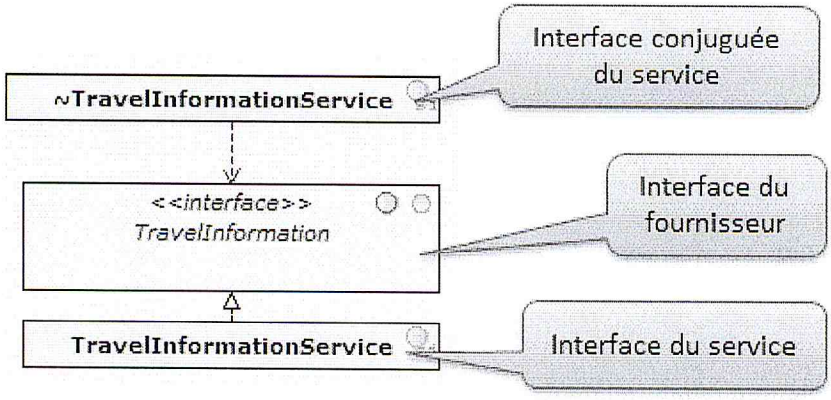


FIGURE A.3 – Diagramme d'interface du service TravelInformation [34].

A.2.5 Diagramme des participants

Ce diagramme permet de :

- Spécifier les composants logiciels qui réalisent les services définies par le diagramme d'architecture des services ;
- Offrir une vision des composants du système ;
- Décrire les relations qui existent entre les composants.

Étape de modélisation

Pour concevoir un diagramme des participants du système, il est conseillé de suivre les étapes suivantes [34] :

- Identification des composants du système
  - Raffiner les participants de l'architecture des services par la décomposition en plusieurs composants plus spécifiques ;
- Créer les ports des composants
  - Un port stéréotypé «service» a pour type l'interface de service ;
  - Un port stéréotypé «request» est typé par une interface de service conjugué ;
- Lier les ports par leurs interfaces fournies et requises.

Exemple

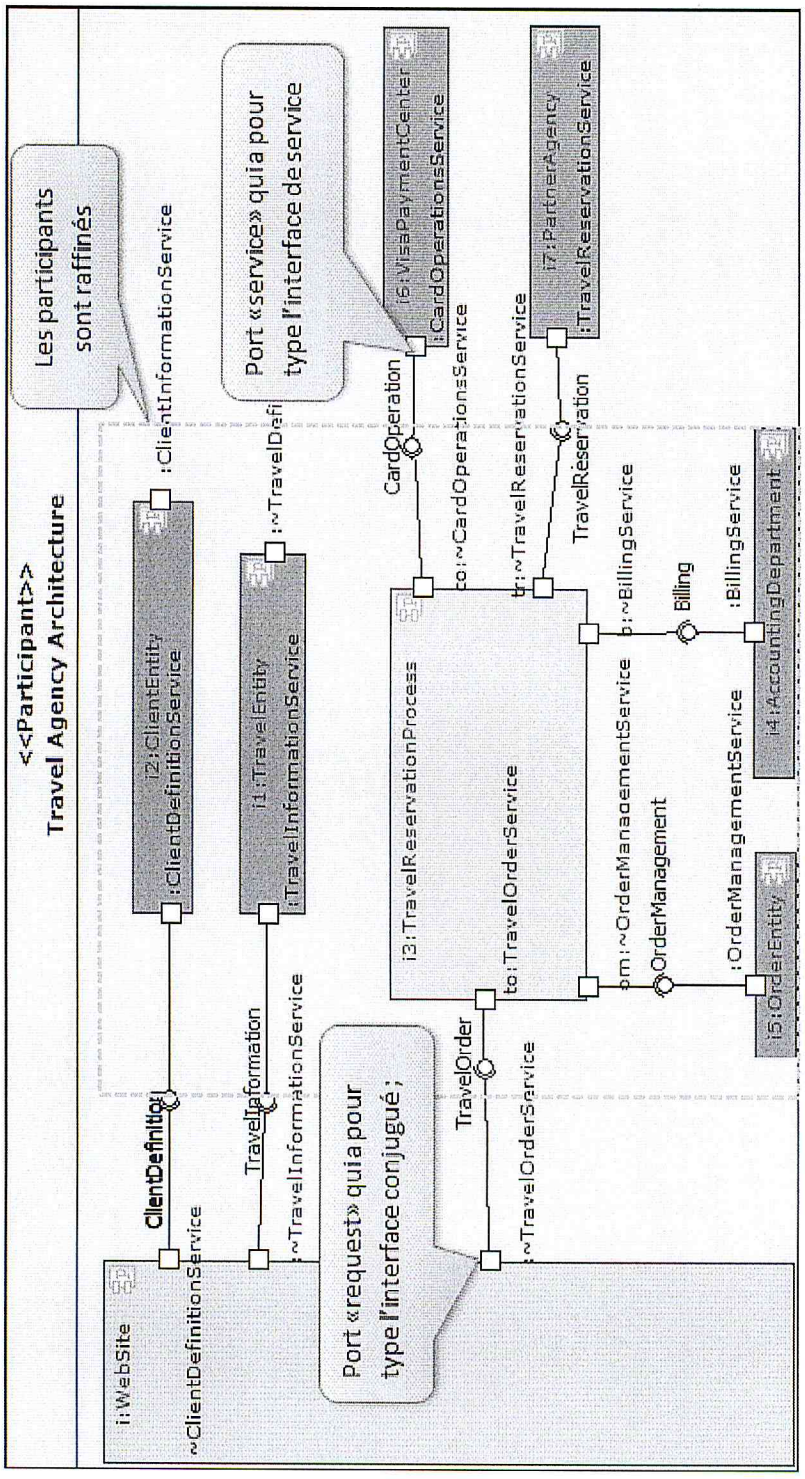


FIGURE A.4 – Diagramme des participants [34].

A.2.6 Diagramme des messages

Le diagramme des messages (ou types de messages) spécifie les informations échangées entre le fournisseur et le consommateur du service. Les types de messages représentent des données pures qui peuvent être transportées entre les différentes parties [34].

Exemple

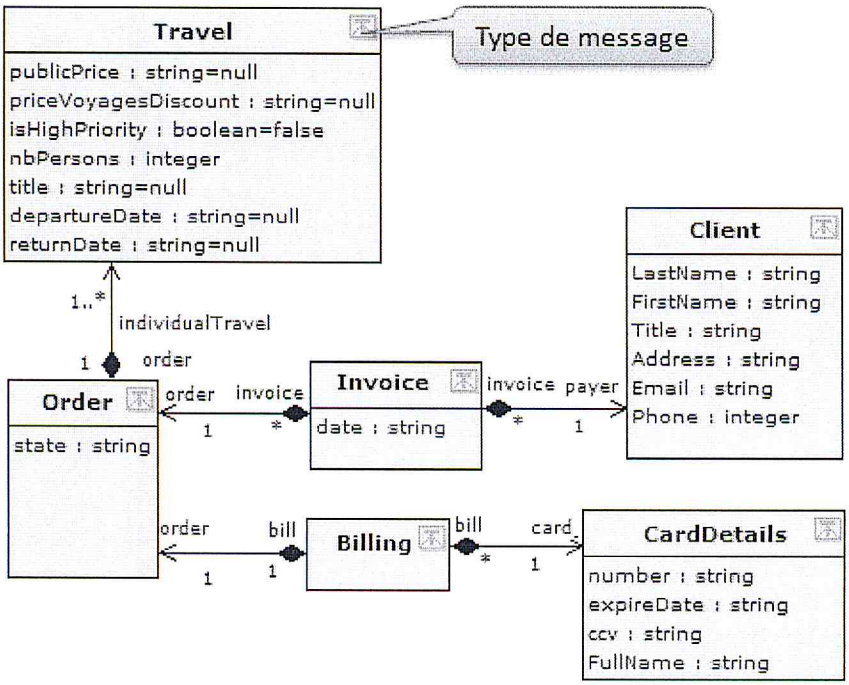


FIGURE A.5 – Diagramme des messages [34].

A.3 La méthode SoaML

Cette méthode a été développée avec le soutien de l'Union européenne **FP7** (Le Septième programme-cadre est le programme cadre actuel (2007-2013) de l'Union européenne pour la recherche et le développement technologique, il est géré par la Commission européenne) dans le cadre du projet **SHAPE**. Son objectif est de fournir les étapes préliminaires du développement d'une architecture orientée service en utilisant le langage SoaML [35]. Cette méthode exige la construction de trois modèles [34] :

- Modèle d'architecture métier **BAM** (**B**usiness **A**rchitecture **M**odelling) ;
- Modèle d'architecture système **SAM** (**S**ystem **A**rchitecture **M**odelling) ;
- Modèle de plateforme spécifique **PSM** (**P**latform **S**pecific **M**odel).

Le deuxième modèle est le résultat de l'opération de raffinement du premier, à ce stade on est plus proche de l'implémentation (dernier modèle).

A.3.1 Modèle d'architecture métier "BAM"

Le **BAM** décrit les opérations métiers ainsi l'environnement où l'architecture orientée service sera installée [34]. Le BAM capture les besoins de l'entreprise et identifie les services par la définition de :

- Les objectifs métiers ;
- La sémantique de l'information (terminologies, ontologies) ;
- Les processus métiers.

Ce modèle décrit :

- L'architecture des services ;
- Les contrats de services.

Afin de construire ce modèle, il est nécessaire d'établir les diagrammes suivants :

- Diagramme d'architecture des services ;
- Diagramme de processus métier ;
- Diagramme de contrats des services ;
- Diagramme des Capabilités (optionnel).

A.3.2 Modèle d'architecture système "SAM"

Ce modèle décrit l'architecture globale du système. À ce stade, deux aspects sont définis [34] :

Aspect structurel : décrit les composants, leurs relations de dépendances (***Dependency***) et leurs interfaces fournies/requises. Afin de construire ce modèle il faut établir les diagrammes suivants :

- Diagramme d'interface des services ;
- Diagramme des messages ;
- Diagramme des participants.

Aspect dynamique : décrit les interactions entre les composants. Le diagramme des chorégraphies du service permet de schématiser cet aspect.

A.3.3 Modèle de plateforme spécifique "PSM"

Le modèle de plateforme spécifique (PSM) contient la conception et la mise en œuvre de l'architecture orientée services spécifiés dans la plate-forme de la technologie choisie, par exemple : service Web, Java Enterprise Edition (JEE), les systèmes multi-agents (MAS), peer-2-peer (P2P), etc. [34].

Bibliographie

[1] K. Barry Douglas, *The Savvy Manager's Guide to Web Services and Service- Oriented Architectures*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[2] C. Szyperski, *Component software : Beyond object-oriented programming*, ACM Press and Addison-Wesley, NY, USA,1998.

[3] M. H. Dodani, *From objects to services : A journey in search of component reuse nirvana*, Journal of Object Technology, 2004.

[4] Frédéric Pourraz, "Diapason : une approche formelle et centrée architecture pour la composition évolutive de services Web", *Thèse de doctorat*, décembre, 2007.

[5] Thomas Erl, *SOA : Principles of Service Design*, Prentice-Hall,2007.

[6] Riadh Ben Halima, "Conception, implantation et expérimentation d'une architecture en bus pour l'auto-réparation des applications distribuées à base de services web", *Thèse de doctorat*, Mai, 2009.

[7] Xavier Fournier-Morel, *SOA, Le guide de l'architecte*, édition Dunod, 2006.

[8] D. Austin, A. Barbir, C. Ferris, S. Garg, *eds : Web Services Architecture Requirements* (W3C Working Group Note, 11 February 2004). Online at :http ://www.w3.org/TR/wsa-reqs/#id2604831

[9] Jérôme Daniel, "Service web : concepts, technique et outils", édition Vuibert, 2003.

[10] Gilbert Babin, Michel Leblanc, *Les web services et leur impact sur le commerce b2b*, CIRANO Burgundy Reports 2003rb-07, CIRANO, 2003.

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

[12] D. Fallside, P. Walmsley, *Extensible markup language schema (xml schema) 1.0.* World Wide Web Consortium, http ://www.w3.org/TR/xmlschema-0, 2004.

[13] Sylvain Rampacek, "Sémantique, interactions et langages de description des services web complexes", *Thèse de doctorat*, novembre 2006.

[14] Zakia Imane Kazi-Aoul, "Une architecture orientée services pour la fourniture de documents multimédia composés adaptables", *Thèse de doctorat*, Janvier 2008.

[15] George Gardarin, *XML, Des bases de données aux services web*, édition Dunod, 2002.

[16] Elarbi Badidi, "Architecture et services pour la distribution de charge dans les systèmes distribués objet", *Thèse de doctorat*, Mai, 2000.

[17] Eric Malville, "L'auto-organisation de groupes pour l'allocation de tâches dans les systèmes Multi-Agents :Application à CORBA", *Thèse de doctorat*, Mars, 1999.

[18] Annick Fron, *Architecture réparties en JAVA : RMI, CORBA, JMS, sockets, SOAP, services web*, édition DUNOD, 2007.

[19] David Durand,"Gestion de la Qualité de Service dans les Applications Réparties sur Bus Middleware Orientés Objet Approche Dirigée par les Modèles", *Thèse de doctorat*, Novembre, 2006.

[20] N. A. B. Gray, "Comparison of Web Services, Java-RMI, and CORBA service implementations", *Fifth Australasian Workshop on Software and System Architectures*, April 13 - 14 2004, Melbourne, Australia.

[21] Denivaldo Cicero Pavao Lopes, "Etude et applications de l'approche MDA pour des plates-formes de Services Web", *Thèse de doctorat*, Juillet, 2005.

[22] Nicolas Salatgé, "Conception et mise en œuvre d'une plate-forme pour la sûreté de fonctionnement des Services Web", *Thèse de doctorat*, décembre, 2006.

[23] Demba Coulibaly, "Un langage et un environnement de conception et de développement de services web complexes", *Thèse de doctorat*, août, 2006.

[24] Matjaz B. Juric, Bostjan Kezmah, Marjan Hericko, Ivan Rozman, Ivan Vezocnik, *Java RMI, RMI Tunneling and Web Services Comparison and Performance Analysis*, SIGPLAN Not., ACM Press, 39(5), 2004.

[25] Robert A. van Engelen, "Pushing the SOAP Envelope with Web Services for Scientific Computing", *International Conference on Web Services (ICWS 2003)*, pp 346-352, Juillet 2003.

[26] M. B. Juric, I. Rozman , M. Hericko,"Performance Comparison of CORBA and RMI", *Informationand Software Technology*, 42, pp 915-933, 2000.

[27] Stéphanie Chollet, "Orchestration de services hétérogènes et sécurisés", *Thèse de doctorat*, décembre, 2009.

[28] Jean-François Pillou, *Tout sur la sécurité informatique*, édition DUNOD, 2005.

[29] Laurent Bloch, Christophe Wolfhugel, *Sécurité informatique Principes et méthode à l'usage des DSI, RSSI et administrateurs*, 2ième édition, édition Eyrolles, 2009.

[30] Solange Ghernaouti-Hélie, *Sécurité informatique et réseau*, édition DUNOD, 2006.

[31] Gilles Roussel, Etienne Duris, Nicolas Bedon, Rémi Forax, *Java et internet : concepts et programmation*, Tome 1-coté client, 2e édition, édition Vuibert, 2002.

[32] Cédric Liorens, Laurent Levier, Denis Valois, *Tableaux de bord de la sécurité réseau*, 2ième édition, édition Eyrolles, 2006.

[33] http ://www.omg.org/spec/SoaML/

[34] Arne J. Berre, Dima Panfilenko, "Service Modeling with SoaML", *Sixth European Conference on Modelling Foundations and Applications*, 15-18 June 2010, Paris, France.

[35] http ://www.shape-project.eu/

[36] http ://www.modeliosoft.com/

[37] Wassila Seddaoui, Madina Yahiaoui,"Conception et réalisation selon une approche MVC mettant en œuvre le Framework Struts2, d'un composant représentant l'état civil d'une APC", *Mémoire d'ingéniorat*, octobre, 2009.

[38] Jérôme Lafosse, *Struts 2 : Le framework de développement d'applications Java EE* , édition eni, 2009.

[39] Jamie Jaworski, Paul J. Perrone, *Java Security*, édition CAMPUS PRESS, 2001.

[40] Eric Maiwald, *Sécurité des réseaux*, édition CAMPUS PRESS, 2001.

[41] Libero Maesano, Christian Bernard, *Services Web avec J2EE et .NET : Conception et implémentations*, édition Eyrolles, 2003.