REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI Tizi-Ouzou

Faculté de Génie Electrique et Informatique

Département d'Informatique

MEMOIRE

En vue de l'obtention du diplôme de Master en informatique

Spécialité : Ingénierie des Systèmes d'information

Thème

Web Service pour une application e-commerce

Cas : Vente de véhicules en ligne

Proposé et dirigé par :
M^r : KERBICHE M'HAND

Réalisé par :
M^elle : KOUROUGHLI LAMIA
M^m e :SBARGOUD OUARDIA neé BESTANI

Devant le jury :

M^r DIB AHMED	Président.
M^r CHAIB YAZID	Examinateur.
M^me YESLI YASMINE	Examinatrice.

2012/2013

Remerciements

Nous tenons à remercier Dieu, le tout puissant, pour nous avoir donné la force et le courage de mener à terme notre projet.

Nous tenons à exprimer notre profonde gratitude à notre promoteur, Monsieur KERBICHE pour nous avoir encadrés et guidés tout au long de la réalisation du projet, pour ses conseils judicieux et minutieusement prodigués. Aussi nous tenons à lui reconnaître le temps précieux qu'il nous a consacré.

Que les membres du jury trouvent ici nos remerciements les plus vifs pour avoir accepté d'honorer par leur jugement notre travail.

Nos sincères sentiments vont à tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce projet. En particulier nos chères familles et nos amis (es).

OUARDIA, LAMIA.

Dédicaces

A mes très chers parents

Je vous dois ce que je suis aujourd'hui grâce à votre amour, à votre patience et vos innombrables sacrifices.

Que ce modeste travail, soit pour vous une petite compensation et reconnaissance envers ce que vous avez fait pour moi.

Que dieu, le tout puissant, vous préserve et vous procure santé et longue vie afin que je puisse à mon tour vous combler.

A mes très chers frères, ma sœur Nassima et toutes mes amies Sarah, Naima, Manel

A toutes mes cousines Nabila, Samia, Hanane et hayet

A mes proches et toute la famille Kouroughli

A ma copine Ouardia et toute sa famille

A tous mes amis (es)

LAMIA

Dédicaces

A mes très chers parents

Pour tout l'amour don't vous m'avez entouré, pour tout ce que vous avez fait pour moi.

Je ferai de mon mieux pour rester un sujet de fierté à vos yeux avec l'espoir de ne jamais vous décevoir.

Que ce modeste travail, soit l'exaucement de vos veux tant formulés et de Vos prières quotidiennes.

A mes frères Farid et Amar, ma chère soeur Chefa

A mon mari Boualem

A mes proches et toute la famille Bestani

A ma copine Lamia et toute sa famille

A tous mes amis (es)

OUARDIA

Liste des tableaux

Liste des figures

Sommaire

De nos jours, le web n'est plus simplement un énorme entrepôt de texte et d'images, son évolution a fait qu'il est aussi un fournisseur de services.

Aujourd'hui, même si toutes les entreprises n'ont pas fondé l'essentiel de leurs services économiques sur le net, une nouvelle technologie leur facilite grandement les choses et permet une communication facile et à distance entre ces entreprises et leurs partenaires et clients c'est **les services web**. Ils font aujourd'hui une technologie révolutionnaire. Elle fournie un cadre pour trouver, décrire et exécuter ces applications à travers le réseau Internet (ou intranet) indépendamment de tout langage de programmation et de toute plate-forme d'exécution. Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.

C'est dans cette dernière approche que notre projet intervient, c'est le développement d'**un service web** pour la vente de véhicule en ligne, qui permettra au client d'effectuer le paiement en ligne des véhicules.

Pour mieux organiser notre mémoire, nous avons choisi la structure chapitrale suivante :

- Le premier chapitre porte sur « La technologie J2EE », dont on trouve les notions fondamentales des éléments de base qui permettent à notre application web, qu'est la vente de véhicule en ligne d'être opérationnelle.
- Le deuxième chapitre « Les Services web », explique d'une façon plus au moins globale l'architecture et le fonctionnement des services web.
- Le troisième chapitre « analyse et conception », décrit la partie d'analyse et de la conception de notre système.
- En fin le dernier chapitre « La réalisation », explique principalement les moyens matériel et logiciel utilisés pour le développement de notre application, ainsi que sont fonctionnement.

Chapitre 1 : Technologie J2EE

1.1 Introduction :

De nos jours, le développement des applications ne s'arrête pas qu'à leur réalisation, mais aussi à leur qualité de dépendance vis-à-vis des nouvelles technologies. L'une des ces technologies révolutionnaires est l'interconnexion des ordinateurs sur internet ou même en réseau local. Avec cette interconnexion, il devient possible de faire fonctionner des applications sur des machines distantes pour répondre aux problématiques suivantes :

- Les données peuvent être présentes uniquement sur le serveur distant.
- Le serveur distant peut disposer d'une puissance de calcul ou de capacité de stockage dont l'utilisateur local ne dispose pas.
- L'application distante peut être utilisée simultanément par un grand nombre d'utilisateurs et sa mise à jour n'intervient qu'à un seul endroit.

La problématique revient à trouver une architecture logicielle qui nous permettra de réaliser nos applications distribuées.

Dans ce chapitre nous allons présenter l'architecture J2EE ainsi que les notions de base nécessaires à sa compréhension.

1.2 Introduction à J2EE [Doudoux, 08]

J2EE est une plate-forme fortement orientée serveur proposé par SUN pour le développement et l'exécution d'applications distribuées. Elle est composée de deux parties essentielles :

- Ø un ensemble de spécifications pour une infrastructure dans laquelle s'exécutent les composants écrits en Java : un tel environnement se nomme serveur d'application.
- Ø un ensemble d'API qui peut être obtenu et utilisé séparément. Pour être utilisées, certaines nécessitent une implémentation de la part d'un fournisseur tiers.

L'utilisation de J2EE pour développer et exécuter une application propose plusieurs avantages :

- Ø une architecture d'application basée sur les composants qui permet un découpage de l'application et donc une séparation des rôles lors du développement.
- Ø la possibilité de s'interfacer avec le système d'information existant grâce à de nombreuses API : JDBC, JNDI, JMS, JCA ...
- Ø la possibilité de choisir les outils de développement et le ou les serveurs d'applications utilisés qu'ils soient commerciaux ou libres.

J2EE permet une grande flexibilité dans le choix de l'architecture de l'application en combinant les différents composants. Ce choix dépend des besoins auxquels doit répondre l'application mais aussi des compétences dans les différentes API de J2EE. L'architecture d'une application se découpe idéalement en au moins trois tiers :

- la partie cliente (tiers Web) : c'est la partie qui permet le dialogue avec l'utilisateur. Elle peut être composée d'une application standalone[1], d'une application web ou d'applets.
- la partie métier (tiers métier ou business) : c'est la partie qui encapsule les traitements (dans des EJB ou des JavaBeans).
- la partie données (tiers Enterprise Information System**)** : c'est la partie qui stocke les données.

1.3 Topologie(s) d'une application J2EE :

1.3.1 Application multi tiers :

Son Architecture logicielle sépare les fonctions en plusieurs étages (states) de traitements. Il existe trois modèles de tiers :

- Modèle 2-tier : client/serveur : 2niveaux =client et serveur.
- Modèle 3-tier : client (utilisateur), serveur d'application, serveur de données.
- Modèle n-tier avec n>3 : client (utilisateur), divers niveaux de liaisons et de traitements de règles (serveur Web, serveurs d'applications, brokers d'objets distants, moniteurs transactionnel, Web service), serveur de données.

1.3.2 Application web :

En informatique, une application Web (aussi appelée *Web application*) est un logiciel applicative, elle est placé sur un serveur et de et se manipule en actionnant des widgets à l'aide d'une interface web, via un réseau informatique (Internet, Intranet, réseau local). Exemples d'application web :

- Les messageries web, les wiki, les blogs sont des applications Web.
- Les moteurs de recherches, les logiciels de commerce électronique, les jeux en ligne, les logiciels de forum peuvent être sous forme d'application Web.
- Des appareils réseau tels par exemple les routeurs sont parfois équipés d'une application Web dans leurs micros logiciels.

1. Standalone : est un qualificatif qui indique qu'un produit peut être utilisé seul.

1.4 Architecture J2EE :

J2EE ajoute de nombreuses couches de niveau au-dessus de la plate-forme J2SE (Java Standard Edition).Chaque couche est conçue pour supporter une différente technologie de développement.

- **Technologie Web Application** : Technologie liée à la production des interfaces web dynamique, par exemple JSP (Java Servlet Page) et servlet.
- **Technologie Enterprise Application** : Technologie plus directement liée à la logique de business : EJB (Entreprise Java Bean), Java Mail, JMS (Java Message Service), JTA (Java Transaction), etc.
- **Technologie Web Services** : technologie utile au développement des application adhérentes au paradigme SOA (Services Oriented Architecture) :web services, JAX-WS (java API for XML-based web services), JAX-RPC(Java API for XML-Based RPC)
- **Technologie Management and Security** : technologie liée à la gestion de la technologie entreprise à fin de réaliser l'accès et l'échange d'information entre machines et services distribués : JAAS (Java Authentification and Authorization Service), JCA (Java Connector Architecture).

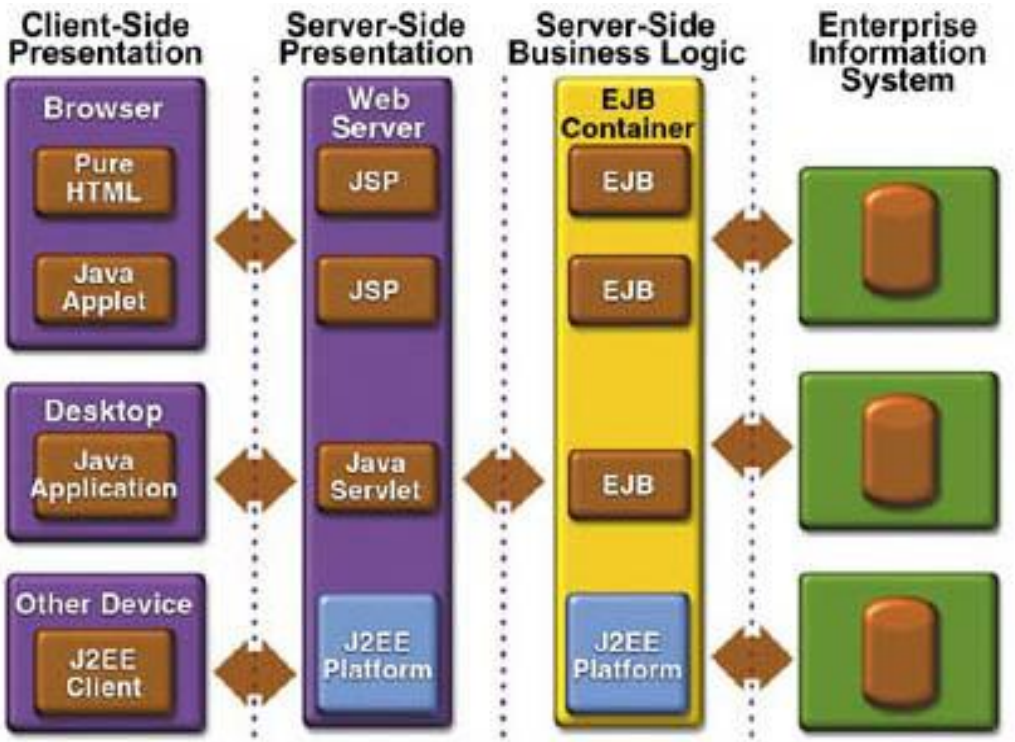


Figure 1.1 : Architecture et les composants de J2EE

Pour expliquer l'utilisation de ces technologies on peut imaginer que les technologies entreprise sont utilisées pour gérer l'accès aux données (généralement une ou plusieurs database), les technologies web application sont utilisées pour montrer les données aux utilisateurs génériques. Dans un contexte Business to Business, les technologies **web service** seront utilisées pour échanger les informations avec les partenaires commerciales et les technologies de gestion gèrent tous les processus informationnels assurant la sécurité des transactions.

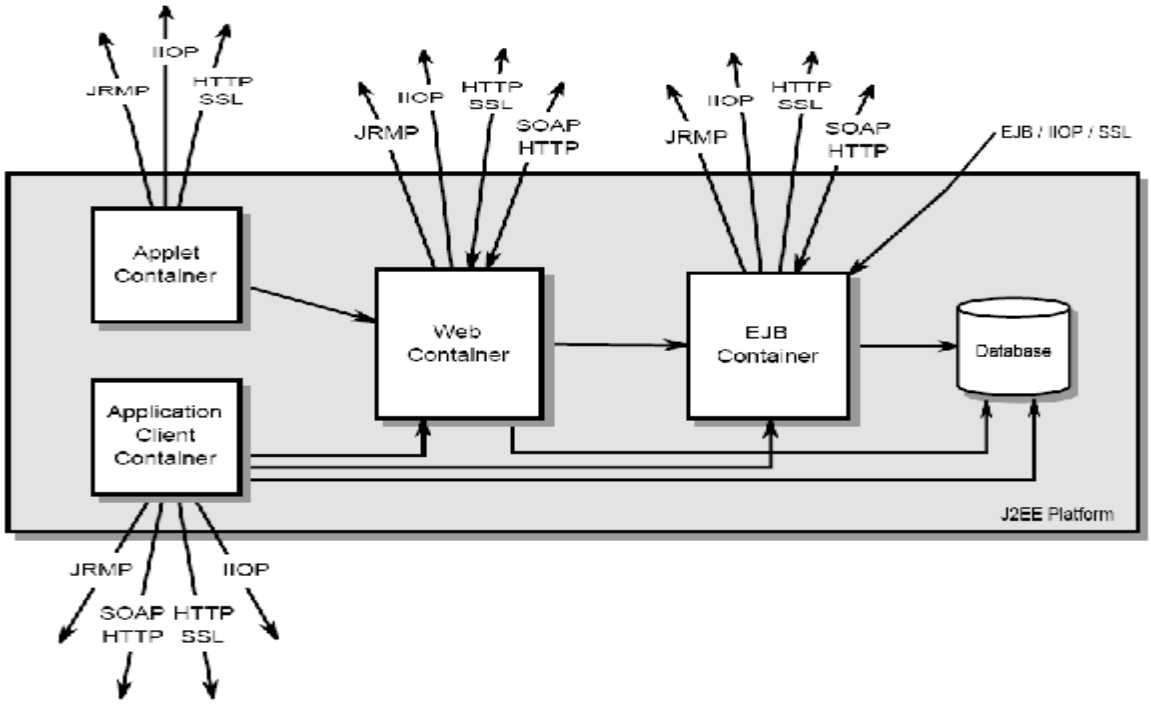


Figure 1.2 : schéma de l'interopérabilité entre la plateforme J2EE et les autres programmes (les différents protocoles utilisés).

Ce schéma est très important dans le domaine de l'interopérabilité entre différentes applications. Il fournit les informations concernant les protocoles utilisés pour chacune des connexions distantes possibles. Par exemple, « Web Container » fournit des accès via HTTP / SSL ou SOAP ; « EJB Container » fournit des accès HTTP / IIOP (RMI : Internet Inter-Orb Protocol) / SSL …

Nous pouvons alors penser qu'un client en C#, par exemple, peut se connecter sur un EJB en mode HTTP (dans l'idéal) ou via le protocole IIOP (plus répandu).

1.5 Les services de J2EE :

1.5.1 Les services d'infrastructure :

Ø JDBC - Java Database Connectivity :

C'est une API d'accès aux bases de données.

Ø JNDI :

C'est une API d'accès aux services de nommage et aux annuaires d'entreprises tels que DNS, NIS, LDAP, etc.

Ø JTA / JTS - Java Transaction Api / Java Transaction Services :

C'est une API définissant des interfaces standard avec un gestionnaire de transactions.

Ø JCA (J2EE Connector Architecture) :

C'est une API de connexion au système d'information de l'entreprise, notamment aux systèmes dits «Legacy» tels que les ERP.

Ø JMX (Java Management eXtension) :

Cette API fournit des extensions permettant de développer des applications web de supervision d'applications.

1.5.2 Les services de communication :

Ø JAAS (Java Authentification and Authorization Service) :

C'est une API de gestion de l'authentification et des droits d'accès.

Ø RMI (Remote Method Invocation) :

C'est une API permettant la communication synchrone entre objets.

Ø Web services :

Les Web services permettent de « partager » un ensemble de méthodes qui pourront être appelées à distance. Cette technologie utilise XML, ce qui permet d'être utilisée par n'importe quel langage et n'importe quelle plateforme.

Ø JMS (Java Message Service) :

Cette API fournit des fonctionnalités de communication asynchrone (appelées MOM pour Middleware Object Message) entre applications.

Ø Java Mail :

C'est une API permettant l'envoi de courrier électronique.

1.6 Composants de l'architecture J2EE : [S. ALLAMARAJU; 01]

- Conteneur de composants
- Clients J2EE
- Composants web
- Composants Entreprise JavaBeans [EJB]
- Serveur EJB

1.6.1 Conteneur de composants J2EE

Les conteneurs assurent la gestion du cycle de vie des composants qui s'exécutent en eux. Ils fournissent des services qui peuvent être utilisés par les applications lors de leur exécution.

Pour cela, il existe plusieurs conteneurs définit par J2EE :

- Conteneur web : pour exécuter les servlets et les JSP.
- Conteneur d'EJB : pour exécuter les EJB.
- Conteneur client : pour exécuter des applications standalone sur les postes qui utilisent des composants J2EE.

Les serveurs d'applications peuvent fournir un conteneur web uniquement (exemple : Tomcat) ou un conteneur d'EJB uniquement (exemple : JBoss, Jonas,...) ou les deux (exemple : Websphere, Weblogic,...).

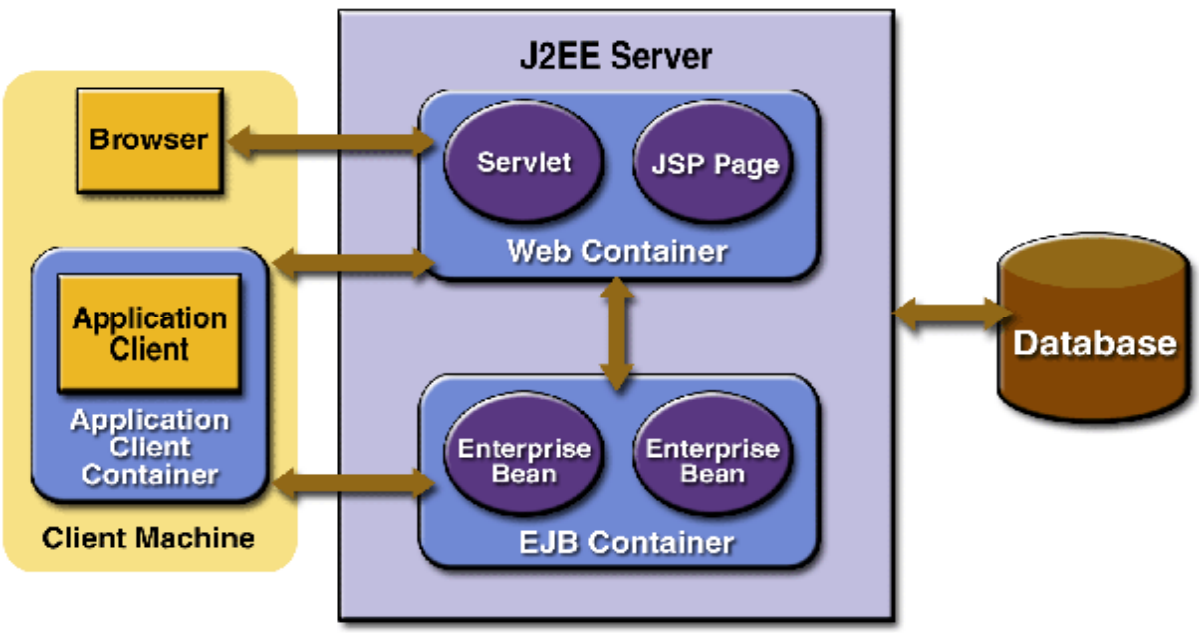


Figure 1.3 :l'architecture des conteneurs

1.6.2 Clients J2EE :

Il s'agit de composants spécifiques ou composants lourds chargés des traitements propres à un secteur d'activité (on parle de *logique métier* ou de *logique applicative*).

1.6.3 Composants web :

1.6.3.1 Pages JSPs

Les JSP (Java Server Pages) sont une technologie Java qui permettent la génération de pages web dynamiques. Permettent d'introduire du code Java dans des tags prédéfinies à l'intérieur d'une page HTML. La technologie JSP mélange la puissance de Java côté serveur et la facilité de mise en page d'HTML côté client.

1.6.3.2 Les servlets

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

Une servlet s'exécute dynamiquement sur le serveur web et permet l'extension des fonctions de ce dernier, typiquement : accès à des bases de données, transactions de commerce, etc. Une servlet peut être chargée automatiquement lors du démarrage du serveur web ou lors de la première requête du client. Une fois chargées, les servlets restent actives dans l'attente d'autres requêtes du client.

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

- **le fonctionnement d'une servlet (cas d'utilisation de http) :**

Un serveur d'application permet de charger et d'exécuter les servlets dans une JVM. C'est une extension du serveur web. Ce serveur d'application contient entre autre un moteur de servlets qui se charge de manager les servlets qu'il contient. Pour exécuter une servlet, il suffit de saisir une URL qui désigne la servlet dans un navigateur.

a) Le serveur reçoit la requête http qui nécessite une servlet de la part du navigateur si c'est la première sollicitation de la servlet, le serveur l'instancie. Les servlets sont stockées (sous forme de fichiers .class) dans un répertoire particulier du

serveur. Ce répertoire dépend du serveur d'application utilisé. La servlet reste en mémoire jusqu'à l'arrêt du serveur. Certains serveurs d'application permettent aussi d'instancier des servlets dés le lancement du serveur.

b) La servlet en mémoire, peut être appelée par plusieurs threads lancés par le serveur pour chaque requête. Ce principe de fonctionnement évite d'instancier un objet de type servlet à chaque requête et permet de maintenir un ensemble de ressources actives tel qu'une connexion à une base de données.

1.6.4. Composants Entreprise JavaBeans [EJB] :

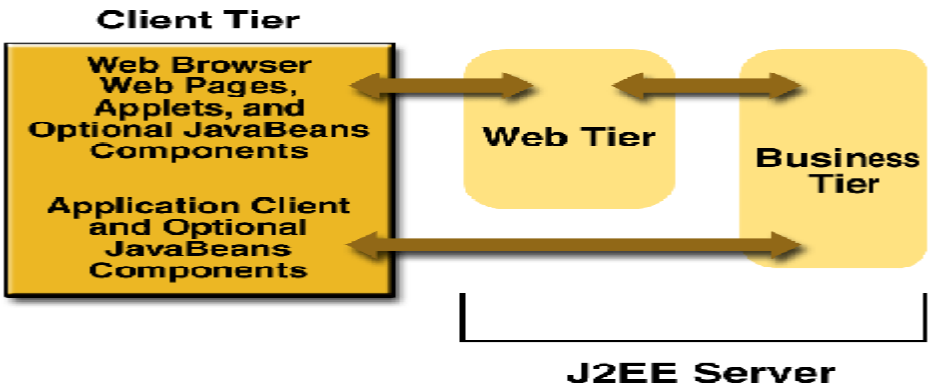
L'architecture Entreprise JavaBeans est une technologie côté serveur pour développer et déployer des composants contenant la logique métier d'une application d'entreprise. Il ya 3 type d'entreprise beans de sessions, d'entité et de messages.

1.6.5 Serveur EJB :

C'est une machine de l'environnement EJB qui gère les conteneurs d'EJB. Les systèmes d'entreprise distribués requièrent un grand nombre de services communs. Cela inclut :

- la gestion de l'état.
- l'accès aux données partagées.
- la participation à des transactions.
- le service d'un grand nombre de clients.
- la fourniture de services d'accès distant aux données et le contrôle de l'accès à ces données.

Le tier central de l'architecture J2EE qu'est le serveur d'EJB tente de réunir tous ces services.



.

Figure 1.4: les communications d'un serveur J2EE

- **Client EJB :**

Le client EJB fournit généralement la logique d l'interface utilisateur sur la machine cliente. Il passe les appels aux composants EJB distants hébergés sur un serveur et doit savoir comment trouver le serveur EJB et interagir avec les composants EJB. Un composant EJB peut faire office de client EJB en appelant les méthodes d'un autre composant EJB.

1.7 Technologie J2EE

1.7.1 JPA: [A. Goncalves, 07]

Littéralement « Java Persistence API », il s'agit d'un standard faisant partie intégrante de la plate-forme Java EE, une spécification qui définit un ensemble de règles permettant la gestion de la correspondance entre des objets Java et une base de données, ou autrement formulé la gestion de la persistance.

Ce mécanisme qui gère la correspondance entre des objets d'une application et les tables d'une base de données se nomme ORM, pour « Object-Relational Mapping ».

Cette technologie a pour objectif d'offrir un modèle d'ORM (**Object Relational Mapping**) indépendant d'un produit particulier (comme Hibernate, TopLink, etc.). Elle est basée sur

- un jeu d'interfaces et de classes permettant de séparer l'utilisateur d'un service de persistance (votre application) et le fournisseur d'un service de persistance,
- un jeu d'annotations pour préciser la mise en correspondance entre classes Java et tables relationnelles,
- un fournisseur de persistance (par exemple Hibernate),
- un fichier XML « persistence.xml » décrivant les moyens de la persistance (fournisseur, *datasource*, etc.)

JPA est utilisable dans les applications Web (conteneur Web), ou dans les EJB (serveur d'applications) ou bien dans les applications standards (Java Standard Edition.

- **Entités**

Une entité est un objet de domaine de persistance léger. Typiquement, une entité représente une table dans une base de données relationnelle, et chaque instance d'entité correspond à une ligne dans cette table. L'artefact de programmation principal d'une entité est la classe d'entité, même si les entités peuvent utiliser des classes d'assistance.

L'état persistant d'une entité est représentée soit par des champs persistants ou des propriétés persistantes. Ces champs ou propriétés utilisent des annotations de mapping

Objet / Relationnel pour mapper les entités et les relations de l'entité à des données relationnelles dans le magasin de données sous-jacente.

- **Exigences pour les classes d'entité**

Une classe d'entité doit respecter les exigences suivantes:

- La classe doit être annotée avec le javax.persistence.Entity annotation.
- La classe doit avoir un constructeur sans argument public ou protégé. La classe peut avoir d'autres constructeurs.
- La classe ne doit pas être déclarée finale. Pas de méthodes ou variables d'instance persistants doivent être déclarés finale.
- Si une instance d'entité est passé par valeur comme un objet détaché, comme à travers l'interface d'affaires distant d'un bean session, la classe doit implémenter l' interface Serializable interface.
- Les entités peuvent étendre les classes d'entités et non-entité, et les classes non-entité peuvent étendre les classes d'entité.
- Variables d'instance persistante doivent être déclarées private, protected ou package-privé, et ne peuvent être directement accessibles par les méthodes de l'entité classe. Les clients doivent accéder à l'état de l'entité par le biais d'accesseur ou des méthodes d'affaires.

1.7.2 JavaMail

JavaMail est une API qui permet d'utiliser le courrier électronique (e-mail) dans une application écrite en java (application cliente, applet, servlet, EJB, ...). Son but est d'être facile à utiliser, de fournir une souplesse qui permette de la faire évoluer et de rester le plus indépendant possible des protocoles utilisés.

Cette API permet une abstraction assez forte de tout système de mails, ce qui lui permet d'ajouter des protocoles non gérés en standard. Pour gérer ces différents protocoles, il faut utiliser une implémentation particulière pour chacun d'eux, fournis par des fournisseurs tiers. En standard, JavaMail 1.2 fournit une implémentation pour les protocoles SMTP, POP3 et IMAP4.

L'API JavaMail(Javax.mail) utilise 4 classes/interfaces principales :

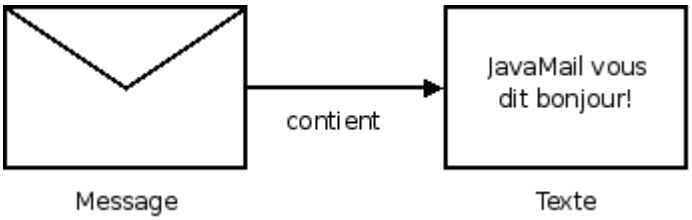
a) Session

Tout d'abord, toute application utilisant l'API JavaMail doit commencer par *ouvrir une session*. Dans JavaMail, la **session** gère les informations de configuration (nom d'utilisateur, mot de passe, hôte) nécessaires pour utiliser les fonctionnalités de JavaMail. Il ne s'agit pas d'une session au sens de *connexion* avec un serveur!

b) Message

Dans ce premier exemple, le message contiendra uniquement du texte.

La seconde étape va être de créer le **message**. Ici encore, il s'agit d'un objet *passif* essentiellement chargé d'encapsuler des informations: sujet, destinataire, contenu, etc.



C'est le seul format de message définit par les spécifications de JavaMail. Néanmoins, chaque mise en œuvre de JavaMail est susceptible de fournir des formats de messages spécialisés. Ainsi, la mise en œuvre de référence fournie par Sun offre par exemple la classe com.sun.mail.smtp.SMTPMessage qui permet de régler certaines options liées à la transmission de ce message par SMTP.

c) InternetAdress

Javax.mail.internet.InternetAdress hérite de la classe javax.mail.Adress et représente une adresse email au format contact@serveurmail.com

d) Transport

Enfin, le code le plus intéressant – et le plus actif: celui chargé de transmettre le message. Code le plus intéressant, mais aussi le plus court:

```
/* ... */
Transport.send(message);
```

L'appel à la méthode Transport.send ouvre une connexion avec le serveur SMTP, transmet le message, puis ferme la connexion avec le serveur.
Tous les paramètres de connexion sont extraits du message et de l'objet session associé.

1.8 L'environnement d'exécution des applications J2EE :

J2EE propose des spécifications pour une infrastructure dans laquelle s'exécutent ses composants. Ces spécifications décrivent les rôles de chaque élément et précisent un ensemble d'interfaces pour permettre à chacun de ces éléments de communiquer.

Ceci permet de séparer les applications et l'environnement dans lequel elles s'exécutent. Les spécifications précisent à l'aide des API un certain nombre de fonctionnalités que doit implémenter l'environnement d'exécution.

Ces fonctionnalités sont de bas niveau ce qui permet aux développeurs de se concentrer sur la logique métier. Pour exécuter ces composants de natures différentes, J2EE définit des conteneurs pour chacun de ces composants. Il définit pour chaque

composant des interfaces qui leur permettront de dialoguer avec les composants lors de leurs exécutions.

Les conteneurs permettent aux applications d'accéder aux ressources et aux services en utilisant les API.

Les appels aux composants se font par des clients via les conteneurs. Les clients n'accèdent pas directement aux composants mais sollicitent le conteneur pour les utiliser.

1.9 L'assemblage et le déploiement d'application J2EE :

J2EE propose une spécification pour décrire le mode d'assemblage et le déploiement d'une application J2EE. Une application J2EE peut regrouper différents modules : modules web, modules EJB ... Chacun de ces modules possède son propre mode de packaging. J2EE propose de regrouper ces différents modules dans un module unique sous la forme d'un fichier EAR (Entreprise Archive).

Le format de cette archive est très semblable à celui des autre archives :

- Un contenu : les différents modules qui composent l'application (module web, EJB, fichier RAR, ...)
- Un fichier descripteur de déploiement

Les serveurs d'application extrait chaque module du fichier EAR et les déploient séparément un par un.

1.10 Les acteurs d'une application J2EE : [Web, 01]

La réalisation d'une application basée sur l'architecture J2EE fait appel à différents types de compétences que l'on trouve rarement chez une même personne car cela va de la conception jusqu'à la supervision de l'application en passant par le développement et le déploiement. Afin de pouvoir maîtriser ce processus J2EE adopte l'approche des partages des responsabilités.

- Le fournisseur des EJB: c'est l'acteur qui fournit des composants métiers réutilisables soit par l'achat à un fournisseur de composants, soit par développement interne.

- Le déployeur : l'acteur qui récupère l'application et s'occupe de son déploiement dans un serveur d'applications.
- L'administrateur: l'acteur qui contrôle le fonctionnement du serveur d'application et assure la supervision des applications.

- Le fournisseur de conteneur : l'éditeur qui commercialise un conteneur web ou un conteneur EJB; cet éditeur commercialise souvent un serveur d'application incluant ces conteneurs.

- Le fournisseur de serveur : c'est l'éditeur qui commercialise un serveur d'application (BEA, IBM etc...).

1.11 Architecture Client /Serveur :

1.11.1 Définition : [Web, 02]

Une application est bâtie selon une architecture client-serveur lorsqu'elle est composée de deux programmes, coopérant l'un avec l'autre à la réalisation d'un même traitement. La première partie, appelée module client, est installée sur le poste de travail alors que la seconde, appelée module serveur, est implantée sur l'ordinateur (ou même des ordinateurs éventuellement situés dans des lieux géographiques différents) chargé de rendre le service (micro, mini ou grand système). L'architecture client-serveur répond aux objectifs précédemment cités.
Dans l'architecture client-serveur, une application est constituée de trois parties :

- l'interface utilisateur
- la logique des traitements
- la gestion des données

Le client n'exécute que l'interface utilisateur (interfaces graphiques de type windows) et la logique des traitements (formuler la requête), laissant au serveur de bases de données la gestion complète des manipulations de données. Le - qui apparaît dans client-serveur correspond à tout un ensemble complexe de logiciels appelé *middleware* qui se charge de toutes les communications entre les processus.

1.11.2 Fonctionnement d'un système client/serveur

Un système client/serveur fonctionne selon le schéma suivant :

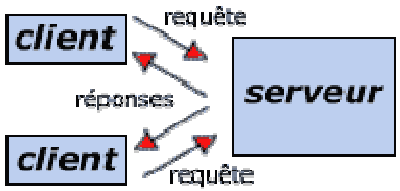


Figure 1.5 Modèle client/serveur

- Le client émet une requête vers le serveur grâce à son adresse IP et le port, qui désigne un service particulier du serveur
- Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine cliente et son port

1.11.3 Avantages de l'architecture client/serveur

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

- **des ressources centralisées** : étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple

une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction

- **une meilleure sécurité** : car le nombre de points d'entrée permettant l'accès aux données est moins important
- **une administration au niveau serveur** : les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés
- **un réseau évolutif** : grâce à cette architecture il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeure.

1.11.4 Inconvénients du modèle client/serveur

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles :

- **un coût élevé** dû à la technicité du serveur
- **un maillon faible** : le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui ! Heureusement, le serveur a une grande tolérance aux pannes (notamment grâce au système RAID)

1.12 Quelques protocoles applicatifs :

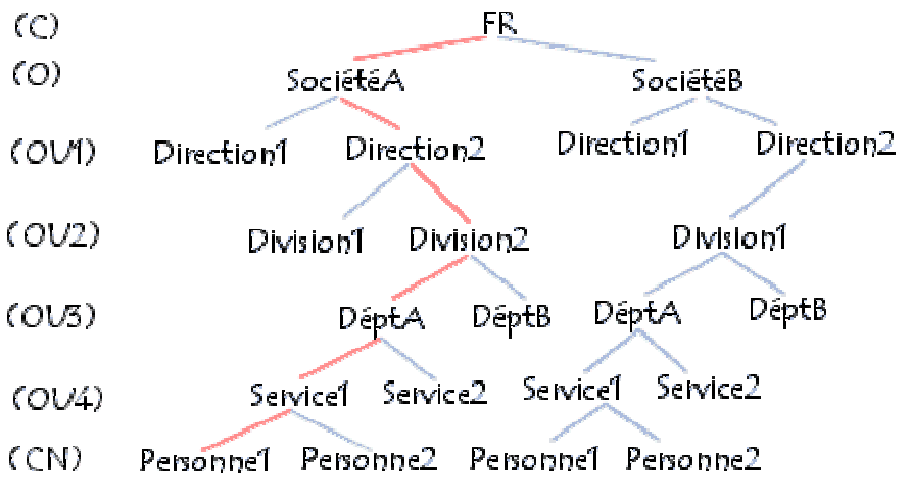
1.12.1 LDAP [Web 03]

LDAP (*Lightweight Directory Access Protocol*, traduisez *Protocole d'accès aux annuaires légers*) est un protocole standard permettant de gérer des annuaires, c'est-à-dire d'accéder à des bases d'informations sur les utilisateurs d'un réseau par l'intermédiaire de protocoles TCP/IP.
Les bases d'informations sont généralement relatives à des utilisateurs, mais elles sont parfois utilisées à d'autres fins comme pour gérer du matériel dans une entreprise.
Le protocole LDAP, développé en 1993 par l'université du Michigan, avait pour but de supplanter le protocole DAP (servant à accéder au service d'annuaireX.500 de l'OSI), en l'intégrant à la suite TCP/IP. A partir de 1995, LDAP est devenu un annuaire natif (*standalone LDAP*), afin de ne plus servir uniquement à accéder à des annuaires de type X500. LDAP est ainsi une version allégée du protocole DAP, d'où son nom de *Lightweight Directory Access Protocol*.
Le protocole LDAP définit la méthode d'accès aux données sur le serveur au niveau du client, et non la manière de laquelle les informations sont stockées.

v L'arborescence d'informations (DIT)

LDAP présente les informations sous forme d'une arborescence d'informations hiérarchique appelée **DIT** (*Directory Information Tree*), dans laquelle les informations, appelées **entrées** (ou encore *DSE, Directory Service*

Entry), sont représentées sous forme de branches. Une branche située à la racine d'une ramification est appelée racine ou suffixe (en anglais *root entry*). Chaque entrée de l'annuaire LDAP correspond à un objet abstrait ou réel (par exemple une personne, un objet matériel, des paramètres, ...). Chaque entrée est constituée d'un ensemble de paires clés/valeurs appelées **attributs**.



1.12.2 DNS

DNS (Domain Name System, système de noms de domaine) est un système de noms pour les ordinateurs et les services réseau organisé selon une hiérarchie de domaines. Le système DNS est utilisé dans les réseaux TCP/IP tels qu'Internet pour localiser des ordinateurs et des services à l'aide de noms conviviaux. Lorsqu'un utilisateur entre un nom DNS dans une application, les services DNS peuvent résoudre ce nom en une autre information qui lui est associée, par exemple une adresse IP.
La plupart des utilisateurs préfèrent en effet un nom convivial comme exemple.microsoft.com pour accéder à un ordinateur tel qu'un serveur de messagerie ou un serveur Web dans un réseau. Un nom convivial est plus facile à retenir. Cependant, les ordinateurs utilisent des adresses numériques pour communiquer sur un réseau. Pour faciliter l'utilisation des ressources réseau, des systèmes de noms comme DNS fournissent une méthode qui établit la correspondance entre le nom convivial d'un ordinateur ou d'un service et son adresse numérique.
L'illustration suivante représente une utilisation élémentaire de DNS qui consiste à trouver l'adresse IP d'un ordinateur à partir de son nom.

Ce système propose :

- un espace de noms hiérarchique permettant de garantir l'unicité d'un nom dans une structure arborescente, à la manière des systèmes de fichiers d'Unix.

- un système de serveurs distribués permettant de rendre disponible l'espace de noms.
- un système de clients permettant de « résoudre » les noms de domaines, c'est-à-dire interroger les serveurs afin de connaître l'adresse IP correspondant à un nom.

1.12.3 SMTP

Le protocole SMTP (*Simple Mail Transfer Protocol*, traduit *Protocole Simple de Transfert de Courrier*) est le protocole standard permettant de transférer le courrier d'un serveur à un autre en connexion point à point.
Il s'agit d'un protocole fonctionnant en mode connecté, encapsulé dans une trame TCP/IP. Le courrier est remis directement au serveur de courrier du destinataire. Le protocole SMTP fonctionne grâce à des commandes textuelles envoyées au serveur SMTP (par défaut sur le port 25). Chacune des commandes envoyées par le client (validée par la chaîne de caractères ASCIICR/LF, équivalent à un appui sur la touche entrée) est suivi d'une réponse du serveur SMTP composée d'un numéro et d'un message descriptif.

Voici un scénario de demande d'envoi de mail à un serveur SMTP

- Lors de l'ouverture de la session SMTP, la première commande à envoyer est la commande *HELO* suivie d'un espace (noté <*SP*>) et du nom de domaine de votre machine (afin de dire "bonjour je suis telle machine"), puis valider par entrée (noté <*CRLF*>). Depuis avril 2001, les spécifications du protocole SMTP, définies dans le RFC 2821, imposent que la commande HELO soit remplacée par la commande *EHLO*.
- La seconde commande est "*MAIL FROM:*" suivie de l'adresse email de l'expéditeur. Si la commande est acceptée le serveur renvoie le message "*250 OK*"
- La commande suivante est "*RCPT TO:*" suivie de l'adresse email du destinataire. Si la commande est acceptée le serveur renvoie le message "*250 OK*"
- La commande *DATA* est la troisième étape de l'envoi. Elle annonce le début du corps du message. Si la commande est acceptée le serveur renvoie un message intermédiaire numéroté *354* indiquant que l'envoi du corps du mail peut commencer et considère l'ensemble des lignes suivantes jusqu'à la fin du message repéré par une ligne contenant uniquement un point. Le corps du mail contient éventuellement certains des en-têtes suivants :
  - Date
  - Subject
  - From

Si la commande est acceptée le serveur renvoie le message "*250 OK*".

Les spécifications de base du protocole SMTP veulent que tous les caractères transmis soient codés en code ASCII sur 7 bits et que le 8ème bit soit explicitement mis à

zéro. Ainsi pour envoyer des caractères accentués il faut faire recours à des algorithmes intégrant les spécifications MIME :

- **base64** pour les fichiers attachés
- **quoted-printable** (d'abréviation *QP*) pour les caractères spéciaux contenus dans le corps du message

1.12.4 HTTP :

Le protocole HTTP (HyperText Transfer Protocol) est le protocole le plus utilisé sur Internet depuis 1990. La version 0.9 était uniquement destinée à transférer des données sur Internet (en particulier des pages Web écrites en HTML. La version 1.0 du protocole (la plus utilisée) permet désormais de transférer des messages avec des en-têtes décrivant le contenu du message en utilisant un codage de type MIME. Le but du protocole HTTP est de permettre un transfert de fichiers (essentiellement au format HTML) localisés grâce à une chaîne de caractères appelée URL entre un navigateur (le client) et un serveur Web (appelé d'ailleurs *httpd* sur les machines UNIX).

- **Communication entre navigateur et serveur**

La communication entre le navigateur et le serveur se fait en deux temps :

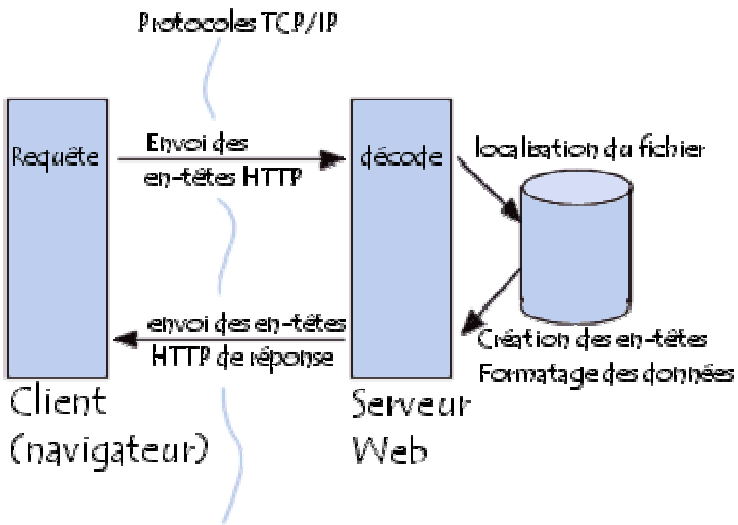


Figure 1.6 Communication entre navigateur et serveur

- Le navigateur effectue une requête HTTP
- Le serveur traite la requête puis envoie une réponse HTTP

Requête HTTP

Une requête HTTP est un ensemble de lignes envoyé au serveur par le navigateur. Elle comprend :

- **Une ligne de requête**: c'est une ligne précisant le type de document demandé, la méthode qui doit être appliquée, et la version du protocole utilisée. La ligne comprend trois éléments devant être séparés par un espace :
  - La méthode
  - L'URL
  - La version du protocole utilisé par le client (généralement *HTTP/1.0*)
- **Les champs d'en-tête de la requête**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la requête et/ou le client (Navigateur, système d'exploitation, ...). Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la requête**: c'est un ensemble de lignes optionnelles devant être séparées des lignes précédentes par une ligne vide et permettant par exemple un envoi de données par une commande POST lors de l'envoi de données au serveur par un formulaire.

Réponse HTTP

Une réponse HTTP est un ensemble de lignes envoyées au navigateur par le serveur. Elle comprend :

- **Une ligne de statut**: c'est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. La ligne comprend trois éléments devant être séparés par un espace :
  - La version du protocole utilisé
  - Le code de statut
  - La signification du code
- **Les champs d'en-tête de la réponse**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la réponse**: il contient le document demandé

1.12.5 RLogin

Rlogin apparu avec BSD 4.2 et était destiné à la connexion à distance uniquement entre les hôtes Unix. Cela en fait un protocole simple que Telnet, depuis la négociation de l'option n'est pas requise lorsque le système d'exploitation sur le client et le serveur sont connus à l'avance. Au cours des dernières années, Rlogin a également été porté sur plusieurs environnements non Unix.

Le rlogin le plus généralement est déployé sur les réseaux de corporation ou d'universitaire, là où l'information de compte d'utilisateur est partagée entre toutes les machines d'unix sur le réseau (employant souvent NIS). Ces déploiements font confiance

essentiellement que la plupart des autres machines (et l'infrastructure de réseau elle-même) et le protocole de rlogin compte sur cette confiance. le rlogind permet des ouvertures sans mot de passe (où le rlogind fait confiance à un client à distance de rlogin) si le centre serveur à distance apparaît dans/etc..centres serveurs.dossier équivalent, ou si l'utilisateur en question a a.dossier de rhosts danS leur répertoire local (qui est fréquemment employer partagé NFS).

Le rlogin a plusieurs problèmes sérieux de sécurité :

- Toute l'information, y compris des mots de passe, est transmise non codée (le rendant vulnérable à l'interception).
- rlogin (il est facile abuser dossier de rhosts, potentiellement laissant *n'importe qui* à l'ouverture sans mot de passe) pour cette raison beaucoup d'interfaces gestionnaire de corporation interdisent les dossiers de rlogin et recherchent activement leurs réseaux des contrevenants.
- Le protocole se fonde en partie sur le client du rlogin de la partie à distance fournissant des informations honnêtement (port y compris de source et nom d'hôte de source). Un client corrompu peut ainsi forger ceci et accéder, car le protocole de rlogin n'a aucun moyen d'authentifier les identités d'autres machines, ou s'assurant que le client de rlogin sur une machine de confiance est *vrai* client de rlogin.
- La pratique courante des répertoires locaux des utilisateurs de support par l'intermédiaire du NFS expose le rlogin à l'attaque au moyen d'article truqué (les rhosts classe) ceci signifie qu'une partie quelconque de sécurité de NFS (légion) censure automatiquement le rlogin de peste.

  En raison de ces problèmes sérieux que le rlogin a été rarement employé à travers untrusted des réseaux (comme l'Internet public) et même dans des déploiements fermés il est tombés dans la désuétude relative (avec des beaucoup d'unix et Linux distributions plus comprenant lui par défaut). Beaucoup de réseaux qui se sont autrefois fondés sur le rlogin et le telnet l'ont remplacé avec SSH et son slogin rlogin-équivalent.

  Le paquet original de Berkeley qui fournit le rlogin également comporte RCP (à distance-copie, laisser classe pour être copié au-dessus du réseau) et rsh (à distance-coquille, permettant à des commandes d'être couru sur une machine à distance sans utilisateur notant dans elle). Ceux-ci partagent les centres serveurs équivalent et arrangement de contrôle d'accès de rhosts (bien qu'ils se relient à un démon différent, rshd), et en tant que tels souffre des mêmes problèmes de sécurité. La suite de ssh contient les substituts appropriés pour tous les deux : le scp remplace le RCP, et le ssh lui-même remplace le rlogin et le rsh.

1.13 Conclusion :

De plus en plus d'entreprises ont besoin de rendre leurs applications accessibles sur le web. Les motivations sont multiples : élargir l'audience des utilisateurs, vendre

des services en ligne, faire communiquer des applications existantes écrites dans des langages de programmation différents. Dans ces conditions, ces entreprises sont amenés parfois à créer des liens entre elles, dans le but de faciliter certaines tâches, c'est là qu'elles vont faire face aux problèmes connus dans l'informatique distribuée, en manière d'échange d'information (documents commerciaux). Une des solutions est l'utilisation des web services, que nous abordons dans le chapitre suivant.

Chapitre 2 :
Les services web

2.1 Introduction :

Aujourd'hui lorsque l'on parle de système d'information ou d'infrastructure, nous voulons souvent exprimer le fait que cet ensemble de machines et de logiciels applicatifs représente un environnement de nature hétérogène. Rare sont les entreprises qui ont des SI utilisant uniquement un environnement propriétaire complet, par exemple IBM, Sun ou Microsoft. L'hétérogénéité est donc une réalité commune pour les entreprises qui doivent souvent faire face à la complexité que cela peut entrainer. L'arrivé des web services constitue une nouvelle tentative, après CORBA, DCOM, RMI, d'adresser cette problématique et d'y répondre le plus élégamment possible en utilisant les technologies à notre disposition tel que le web HTTP et le XML.

2.2 Présentation des services web: [M a, Be, Le]

2.2.1 Architecture orientée services :

Lorsqu'on parle de Web Services, on parle aussi d'architecture orientée services. On définit l'architecture orientée services comme un style d'architecture qui a comme objectif une interdépendance faible (loose coupling) entre différents agents logiciels (modules, services). L'architecture orientée services promeut la réutilisation de composants logiciels au niveau macro. (Comparée à la programmation orientée objet qui promeut la réutilisation au niveau micro, classes, objets).

2.2.2 Définition d'un service web:

Un service Web est un composant logiciel identifié par une URI, dont les interfaces publiques sont définies et appelées en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Les services Web peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par les protocoles Internet.

En d'autres termes, Un service web est une fonctionnalité accessible au travers du réseau grâce à des messages au format XML. Le format de ces messages est généralement SOAP bien que d'autres formats existent (REST, XML-RPC, ...).

Les services Web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde.

Les services Web sont **normalisés** car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes. C'est dans ce contexte qu'un intérêt très particulier a été attribué à la conception des services Web puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les services Web

n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les services Web ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards comme c'est le cas de la plate-forme CORBA qui définit des ponts spécifiques entre le langage de définition IDl et différents langages de programmation. Ainsi, les fournisseurs d'outils de développement peuvent facilement différencier leurs produits avec ceux de leurs concurrents en offrant différents niveaux de sophistication.
Les services Web représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

2.2.3 Les caractéristiques d'un service web :

La technologie des services Web repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web.

Un service Web possède les caractéristiques suivantes :

- il est accessible via le réseau ;
- il dispose d'une interface publique (ensemble d'opérations) décrite en XML ;
- ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire ;
- il communique en utilisant des messages XML, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...) ;
- l'intégration d'application en implémentant des services web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur.
  Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire.

2.3 Fonctionnement des services web :

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par le schéma suivant :

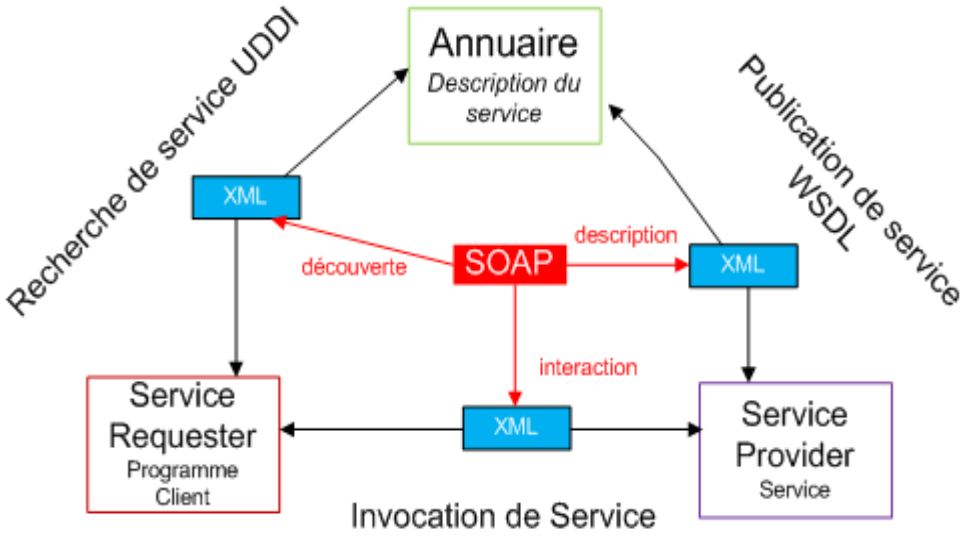


Figure 2.1 Fonctionnement des services web

Ø Service provider

Le fournisseur de service met en application le service Web et le rend disponible sur Internet ; comme aussi il peut être le client d'un autre fournisseur (Interopérabilité).

Il est représenté par un serveur d'applications (J2EE par exemple) et il détient un ou plusieurs services qui sont représentés par des EJBs ou des servlets qui sont enveloppés d'une couche « service ».

Ø Service requester (programme client)

C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML (REST, XML-RPC, SOAP).

Ø Service registry (Annuaire)

Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver. Les interactions entre ces trois acteurs suivent plusieurs étapes :

- **La publication du service** : le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.
- **La recherche du service** : le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir les descriptions et les URL des services demandés afin de lui permettre de les invoquer.
- **L'invocation du service** : une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services.

2.4 XML et les trois standards SOAP, WSDL, UDDI [J. Chauvet]

2.4.1 XML [E. Niedermair]

2.4.1.1 Définition

Le XML créé en1999, est resté durant prés de deux ans un concept plutôt abstrait et théorique faute de moyens fiables pour en afficher le résultat. Avec le développement des nouvelles techniques comme le XSL, il est devenu possible de percevoir concrètement les énormes potentialités de ce nouveau langage.

Le déploiement des services Web (SOAP, WSDL, UDDI) est assuré par le langage XML, qui est utilisé comme un format d'importation et d'exportation d'informations et dans les appels RPC pour l'invocation des procédures à distance. L'apport du langage XML et son impact sur l'informatique distribuée en particulier les services web, sont résumés dans les points suivants :

- XML est un langage simple, définit un standard de description de donnée. Les messages sont autodescriptifs ils permettent le traitement des données sans en connaitre le format. De tels messages sont indépendants des systèmes d'exploitation, des plates-formes, des langages de programmation et de formats d'affichage. Ceci facilite l'échange de données entre les différents partenaires.
- XML et les technologies qu'y sont liées (SOAP, WSDL, UDDI) forment un Framework des services web. Une association qui a révolutionné la manière dont les applications communiquent entre elles. Ce langage a fait des services web des applications à accès universel à partir de n'importe quel ordinateur ou appareil, indépendamment de sa technologie propriétaire de base.
- Les services web XML ont fait d'Internet un réseau orientée services, sur lequel à peu prés tous les services ou applications dont nous pourrions avoir besoin s'y trouvent.
- XML suivi par tous les éditeurs (Microsoft, Sun, Oracle, etc.) ce qui rend disponible les outils nécessaires pour le développement des services Web, tel est le cas pour les plates-formes J2EE de Sun Microsystems et .Net de Microsoft.

2.4.2 Protocole SOAP

2.4.2.1 Définition

SOAP (*Simple Object Access Protocol*) est un protocole standard de communication. Basé sur XML, SOAP a pour principal objectif d'assurer la

communication entre machines. Le protocole permet d'appeler une méthode RPC et d'envoyer des messages aux machines distantes via HTTP. Ce protocole est très bien adapté à l'utilisation des services Web, car il permet de fournir au client une grande quantité d'informations récupérées sur un réseau de serveurs tiers.

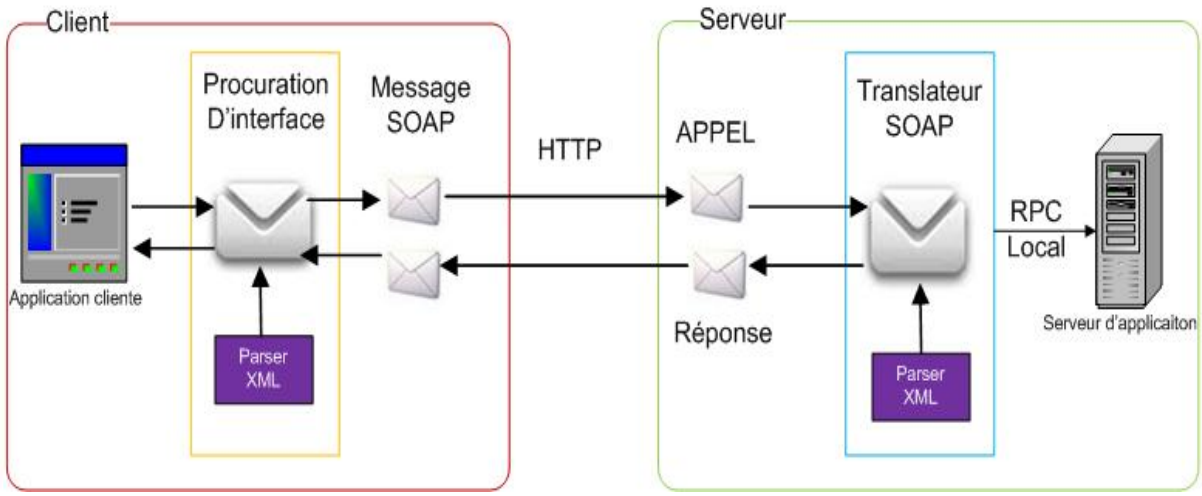


Figure 2.2: Le protocole de communication SOAP

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

2.4.2.2 Structure d'un message SOAP

Un message SOAP est composé de deux parties obligatoires : l'enveloppe SOAP et le corps SOAP ; et une partie optionnelle : l'en-tête SOAP.

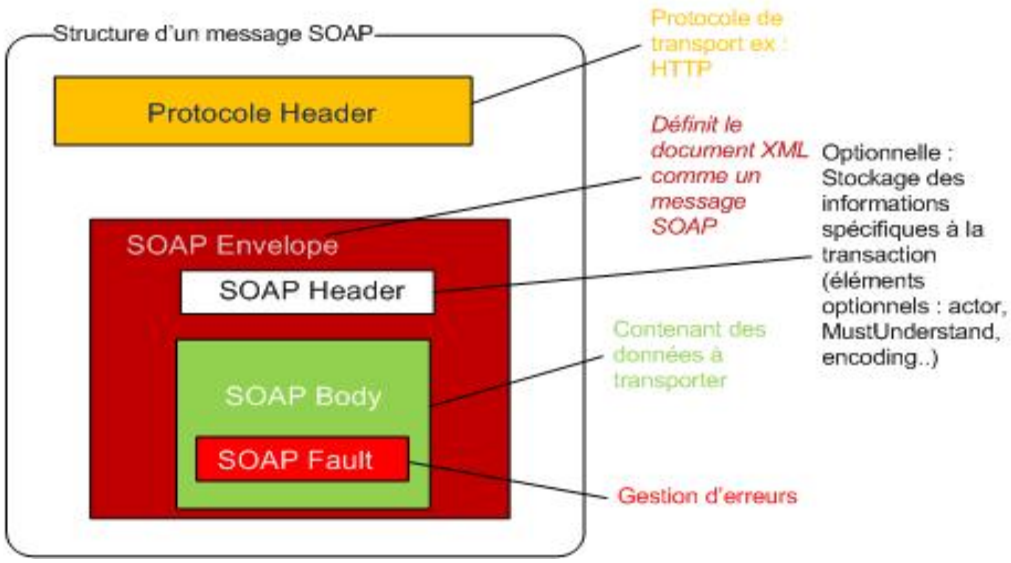


Figure 2.3 Structure d'un message SOAP

- **SOAP envelope** (enveloppe) est l'élément de base du message SOAP. L'enveloppe contient la spécification des espaces de désignation (namespace) et du codage de données.

- **SOAP header** (entête) est une partie facultative qui permet d'ajouter des fonctionnalités à un message SOAP de manière décentralisée sans agrément entre les parties qui communiquent. C'est ici qu'il est indiqué si le message est mandataire ou optionnel. L'entête est utile surtout, quand le message doit être traité par plusieurs intermédiaires.

- **SOAP body** (corps) est un *container* pour les informations mandataires à l'intention du récepteur du message, il contient les méthodes et les paramètres qui seront exécutés par le destinataire final.

- **SOAP fault** (erreur) est un élément facultatif défini dans le corps SOAP et qui est utilisé pour reporter les erreurs.

L'enveloppe SOAP

L'enveloppe SOAP sert de conteneur aux autres éléments du message SOAP, elle est définie au début par la balise <soap:Envelope> et se termine par la balise </soap:Envelope>. Les messages SOAP ne peuvent pas être envoyés en lots, autrement dit l'enveloppe contient un seul message constitué d'un entête facultatif (SOAP header) et d'un corps obligatoire (SOAP body).

```
<? xml version="1.0" encoding="utf-8"?>
 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
  <soap:Header>
       <!-- en-tête -->
  </soap:Header>

  <soap:Body>
        <!-- corps -->
  </soap:Body>

 </soap:Envelope>
```

Toutes les balises XML associées à SOAP ont le préfixe soap (on trouve des développeurs utilisant "soap-env"). L'entête est <soap:Header> et le corps <soap:Body>.

SOAP repose entièrement sur les espaces de noms XML. Dans cet exemple, les espaces de noms sont introduits à l'aide d'un mot-clé « xmlns » *XML namespace* qui signifie espace de noms XML. L'espace de noms est utilisé pour identifier toutes les balises afin d'éviter les conflits. La spécification impose que tous les attributs contenus dans l'enveloppe SOAP soient explicitement associés à un *namespace*, de manière à supprimer toute ambiguïté.

Le corps SOAP

Le corps SOAP est un élément obligatoire dans le message SOAP. Il contient l'information destinée au receveur. Le corps (*body*) doit fournir le nom de la méthode invoquée par une requête ainsi que les paramètres associés à celle-ci.

Le contenu du corps SOAP est utilisé pour spécifier un appel de méthode à un ordinateur distant avec les valeurs de paramètre. Par exemple, la demande du solde d'un compte bancaire.

L'extrait suivant représente un corps SOAP qui fait appel de procédure distante (RPC) à une méthode appelée checkAccountBalance().

```
<soap:Body>
        <checkAccountBalance>
            <accountNumber xsi: type="xsd:int">1234567890</accountNumber>
        </checkAccountBalance>
</soap:Body>
```

Le corps du message SOAP commence avec la balise <soap:Body> et se termine avec la balise </soap:Body>.
L'élément <checkAccountBalance> fournit le nom de la méthode à appeler : checkAccountBalance.
L'élément accountNumber est un paramètre qui est passé dans la méthode checkAccountBalance. Les attributs xsi et xsd définissent les espaces de noms qui vont être utilisés dans le corps du message. La définition de xsi permet d'utiliser xsi:type dans le corps du message, le xsd:int signifie que cette valeur est de type entier. 1234567890 est la valeur donnée au paramètre.

Une différence importante entre SOAP et XML-RPC est que les méthodes SOAP prennent des paramètres nommés. L'ordre des paramètres, lui, n'a pas d'importance, contrairement à XML-RPC.

L'en-tête SOAP

L'en-tête SOAP est un élément facultatif dans un message SOAP. Toutefois, si un en-tête est présent, il doit être le premier élément qui apparaît dans l'enveloppe SOAP. Le format de l'en-tête n'est pas défini dans le cahier des charges et par conséquent, il est à la disposition des clients et des services pour leur propre usage. Cet usage typique serait de communiquer des informations authentifiant l'émetteur ou bien encore le contexte d'une transaction dont le message SOAP doit passer par plusieurs intermédiaires SOAP pour arriver au destinataire final. Un intermédiaire SOAP est toute entité capable de recevoir et transmettre des messages SOAP.

L'en-tête d'un message SOAP commence avec la balise <soap:Header> et se termine avec la balise </soap:Header>, On peut aussi faire <sopa-env:Header></sopa-env:Header>.

Trois attributs associés à l'en-tête SOAP peuvent être utilisés :

- **soap:mustUnderstand** : cet attribut prend la valeur 1 ou 0. La valeur 1 signale que le récepteur doit reconnaître l'information présente dans l'en-tête et que son traitement est obligatoire. La valeur 0 indique que l'en-tête peut être ignoré par le récepteur.

- **soap:role** : sert à indiquer le destinataire SOAP auquel un bloc d'en-tête SOAP particulier est destiné.

- **soap:relay** : est utilisé pour indiquer si un bloc d'en-tête SOAP ciblé sur un récepteur SOAP doit être réacheminé (relayé) s'il n'est pas traité.

<soap:role> et <soap:relay> sont utilisés conjointement par l'ensemble des nœuds SOAP intermédiaires qu'un message SOAP doit traverser pour arriver au destinataire final.

Exemple de Bloc Header, Message à destination de Plusieurs Nœuds SOAP

```
<!--
  Élément USER : À destination du nœud RightManager
-->
  <soap:Header>
      <m:User xmlns:m="http://www.monsite.com/rights/"
          soap:actor="http://www.monsite.com/rights/RightsManager">
         Thunderseb
      </m:User>
   <!--
    Élément Session : À destination du nœud final
    -->
  <m:Session xmlns:m="http://www.monsite.com/session/"
     soap:mustUnderstand="1">
  12AE3C
    </m:Session>
    <!--
        Élément USER : À destination du prochain nœud
    -->
     <m:Lang xmlns:m="http://www.monsite.com/lang/"
   soap:actor="http://schemas.xmlsoap.org/soap/next" soap:mustUnderstand="0">
FR
    </m:Lang>
 </soap:Header>
```

Message d'erreur SOAP

Afin de récupérer le plus grand nombre d'erreurs, l'approche SOAP se base essentiellement sur le bon usage de la balise <soap:fault> qui est contenue dans le corps SOAP. Cette balise est utilisée pour communiquer un problème qui a eu lieu dans la tentative de réalisation de la demande adressée au service Web. L'élément

d'erreur est facultatif et figure uniquement dans les messages de réponse, il ne peut y apparaître qu'une seule fois. La balise <soap:fault> peut contenir quatre autres balises facultatives : faultcode, faultstring, faultactor, detail.

Exemple : Bloc Fault

```
<soap:Body>
<soap:Fault>
<!--
 Identifiant de l'erreur ? défini par SOAP
 -->
 <faultcode>soap:Server</faultcode>
 <!--
   Description brève du message
  -->
<faultstring>Impossible de router le message.</faultstring>
 <!--
 Composant qui a généré l'erreur (URL).
  -->
  <faultactor>http://www.monsite.com/messageDispatcher</faultactor>
  <!--
    Message spécifique à l'application
   -->
  <detail>
  <m:error xmlns:m="http://www.monsite.com/errors"> E_NO_ROUTE
</m:error>
  </detail>
</soap:Fault>
</soap:Body>
```

2.4.3 WSDL

2.4.3.1 Définition

WSDL (*Web Services Description Language*) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. L'utilité de WSDL est de décrire et publier le format et les protocoles d'un web service de manière homogène par l'utilisation du format XML. Cela permettra au requérant et à l'émetteur d'un service de comprendre les données qui seront échangées.

2.4.3.2 Structure d'un document WSDL

Un document WSDL est tout d'abord un document XML, il contient sept éléments :

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

Le document WSDL peut être divisé en deux parties. Une partie pour **les descriptions concrètes**, tandis que la deuxième contient **les définitions abstraites.**

a) **La description concrète** est composée des éléments qui sont orientés vers le client pour le service physique. Les trois éléments concrets XML présents dans un WSDL sont :

- § <wsdl:service>
- § <wsdl:port>
- § <wsdl:binding>

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

- § <wsdl:types>
- § <wsdl:message>
- § <wsdl:operation>
- § <wsdl:portType>

2.4.4 L'annuaire UDDI

2.4.4.1 Définition

UDDI (*Universal Description, Discovery and Integration*) est un annuaire de services. Il fournit l'infrastructure de base pour la publication et la découverte des services Web. UDDI détermine comment on doit organiser l'information concernant une entreprise et le web service qu'elle offre à la communauté afin de permettre à cette dernière, d'y avoir accès.

Les informations qu'il contient peuvent être séparées en trois types :

- les pages blanches qui incluent l'adresse, le contact et les identifiants relatifs au service Web ;
- les pages jaunes qui identifient les secteurs d'affaires relatifs au service Web ;
- les pages vertes qui donnent les informations techniques sur le service web, ainsi que des références sur des fichiers de description, son url, les mécanismes de découverte propre au serveur (fichier disco…).

Les recherches ou ajouts de services web dans l'annuaire se font par requêtes XML (définies dans le schéma fournit par uddi.org) avec une couche SOAP pour le transport.
Ces requêtes peuvent être faites par le web (comme pour les 3 serveurs existants pour le moment) ou de manière logicielle, 2 API étant définies pour l'utilisation de ces requêtes (Inquiry API, Publication API).

2.4.4.2 Structures de données UDDI

La structure de données est exprimée en utilisant les schémas XML .Ils sont divisées en quatre types de structure de données :

- ***Business Entity*** : c'est l'équivalent des pages blanches de l'annuaire; cette structure peut inclure un BusinessKey (clé d'affaire) assurant la protection de l'information contenu dans le web service.
- ***BusinessService*** : c'est l'équivalent des pages jaunes de l'annuaire. on y retrouve les informations concernant le nom et la description du web service offert.
- ***BindingTemplate*** : c'est l'équivalent des pages vertes de l'annuaire, il contient les informations concernant les points d'accès au web service et les aspects

techniques permettant la liaison entre le web service et l'API du requérant de celui-ci. Il fait référence à un ou plusieurs tModel.

- ***tModel*** : il s'agit du mécanisme permettant l'échange de métadonnée à propos du web service. Il peut s'agir d'un pointeur à un fichier WSDL, mais il peut aussi pointer sur d'autres types de fichiers ebXML ou Rosetanet par exemple. Mais chaque tModel ne définira qu'un type d'interface avec laquelle elle peut interagir. Cependant un fichier UDDI peut contenir plusieurs tModel.

2.5 Description en couche des services Web

Les services Web emploient un ensemble de technologies qui ont été conçues afin de respecter une structure en couches sans être dépendante de façon excessive de la pile des protocoles. Cette structure est formée de quatre couches majeures :

- **Couche transport**

Cette couche est responsable du transport des messages XML échangés entre les applications. Actuellement, cette couche inclut HTTP, SMTP, FTP, et de nouveaux protocoles tels que BEEP.

- **Couche communication**

Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Actuellement, deux styles architecturaux totalement différents sont utilisés pour ces échanges de données. Nous avons d'un côté l'architecture orientée opérations distribuées (protocoles RPC) basée sur XML et qui comprend XML-RPC et SOAP et de l'autre côté une architecture orientée ressources Web, REST (Representational State Transfer) qui se base uniquement sur le bon usage des principes du Web (en particulier, le protocole HTTP).

- **Couche description de service**

Cette couche est responsable de la description de l'interface publique du service Web. Le langage utilisé pour décrire un service Web est WSDL qui est la notation standard basée sur XML pour construire la description de l'interface d'un service. Cette spécification définit une grammaire XML pour décrire les services Web comme des ensembles de points finaux de communication (ports) à travers lesquels on effectue l'échange de messages.

- **Couche publication de service**

Cette couche est chargée de centraliser les services dans un registre commun, et de simplifier les fonctionnalités de recherche et de publication des services Web. Actuellement, la découverte des services est assurée par un annuaire UDDI (Universal Description, Discrovery, and Integration).

2.6 Avantages des services web

Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.

Utilisent des standards et protocoles ouverts.

Les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.

Basés sur le protocole HTTP, les services Web peuvent fonctionner au travers de nombreux pare-feu sans nécessiter des changements sur les règles de filtrage.

Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants.

L'architecture modulaire des services web combinée au faible couplage des interfaces associées, permet l'utilisation et la réutilisation de services qui peuvent facilement être recombinées à différents autres modules;

2.7 Apports des services web aux entreprises : [Cirano, 03]

Les services web devraient permettre aux entreprises de *:*

- donner aux clients un accès direct à l'information, aux données et aux fonctionnalités dont ils ont besoin pour interagir avec une entreprise;
- donner aux partenaires d'une entreprise un accès direct à la fonctionnalité dont ils ont besoin pour mieux servir les clients qu'ils ont en commun avec cette entreprise;
- donner aux fournisseurs d'une entreprise un accès direct à l'information et la fonctionnalité dont ils ont besoin pour leur permettre d'ajuster les inventaires selon le modèle « juste à temps »;
- intégrer des applications de bout en bout, de manière abordable, facile à implanter autant hors des frontières de l'entreprise qu'à l'intérieur de celles-ci;
- avoir des équipes de développement travaillant indépendamment et efficacement sur des systèmes qui interagiront, parce que ces équipes travailleront à développer des interfaces communes plutôt que d'avoir à synchroniser les processus.

Les services web offrent donc aux entreprises la flexibilité de réponse et d'anticipation des besoins changeant des clients, la rationalisation des infrastructures logiciels et la flexibilité d'interaction et de configuration des alliances externes avec les partenaires et fournisseurs.

2.8 Conclusion

Les services web sont le résultat d'une évolution technologique majeure, et les bases d'une révolution organisationnelle et économique. Le changement est considérable : les services Web vont modifier en profondeur l'organisation du travail et les processus métier des organisations (entreprises, administrations, associations). En outre, ils vont changer les pratiques des professionnels de l'informatique. Il est évident que les services web reposent sur diverses technologies, mais ils représentent surtout une volonté commune des manufacturiers, des organismes de standards et des utilisateurs de développer des outils permettant une réelle interopérabilité.

Le chapitre suivant de notre mémoire sera consacré pour l'analyse et la conception de notre système.

Chapitre 3 :
Analyse & conception

3.1 Introduction:

Dans le but d'une meilleure organisation et une bonne maitrise du travail, tout processus de développement d'application ou système doit suivre une méthode ou démarche bien définie.

Dans ce chapitre, nous allons entamer le travail par une analyse qui mettra en évidence les différents acteurs intervenant dans le système cible ainsi que leurs besoins. La phase conception, s'appuyant sur les résultats de la phase analyse donnera la modélisation des objectifs à atteindre. Pour ce faire, notre démarche va s'appuyer sur le langage UML, conçu pour la visualisation, la spécification et la construction des systèmes logiciels.

3.2 Objectif de l'application :

L'objectif de notre application est la vente des véhicules en assurant un e- paiement (par carte bancaire) via la mise en place d'un service web visant à permettre l'interopérabilité entre les différents systèmes bancaires hétérogènes.

3.3 L'e-paiement :

Pour notre système, nous allons développer un service web qui est une passerelle de paiement en ligne qui permet de payer les frais par carte bancaire.

Le processus de paiement en ligne implique plusieurs acteurs. Pour un paiement par carte bancaire, sont impliqués l'acheteur et l'E-Marchand, la banque de l'acheteur, celle de l'e-marchand, les réseaux de cartes bancaires sans oublier les services d'authentification. Entre ces entités des contrats, des commissions et des cotisations se mettent en place en plus des mouvements directement rattachés à la transaction.

- ***Le processus de paiement effectué par notre service web :***

Après avoir passé la commande, le payeur communique ses coordonnées (Nom utilisateur, Mot de passe)

Une fois ces coordonnées validées, le service débite le compte de client et crédite le compte de marchand du montant de véhicule auprès de deux banques distinctes mais il existe toujours un problème d'intégration, avec la construction de passerelles entre ces systèmes bancaires c'est ce que nous allons le résoudre avec le développement d'un outil (service web) permettant et favorisant une interopérabilité entre les plates formes, les systèmes d'exploitation, les langages et les programmes.

3.4 Analyse:

Chaque analyse commence par l'examen du modèle des cas d'utilisation et de leurs scénarios, ainsi que des besoins fonctionnels du système, pour aboutir à un modèle d'analyse

constitué par les classes et leurs collaborations, qui traduisent les comportements dynamiques, détaillés dans les cas d'utilisation.
Pour concevoir notre application web, on a choisit d'utiliser l'extension d'UML pour le Web.

3.4.1 Diagramme de contexte :

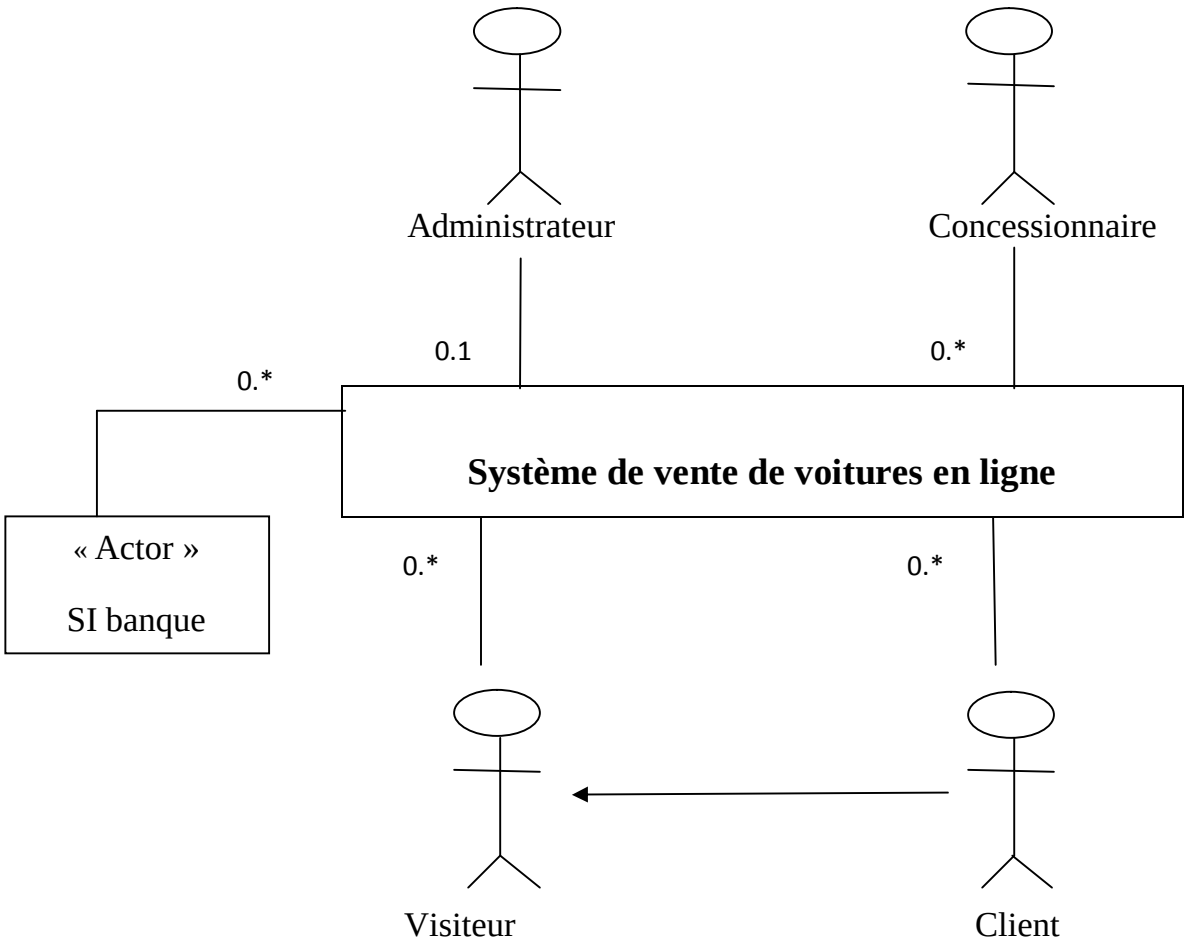


Figure 3.1 Diagramme de contexte

2.4.2 Spécification des besoins :

2.4.2.1 Identification des acteurs :

Définition : Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

Les acteurs de notre système sont :

- **Visiteur :** Est toute personne se connectant au site pour visualiser ou acheter une voiture. Notons qu'à tout moment, le visiteur peut choisir de créer un compte afin de devenir client.
- **Client :** Connu du système, qu'est le seul à pouvoir effectuer une commande ou enregistrer. Ce client peut-être un particulier ou une entreprise.
- **Concessionnaire :** Il s'occupe de la gestion des véhicules, factures et le traitement des commandes, ...
- **Administrateur :** Son rôle est la gestion des tables client et concessionnaire (consulter, modifier, supprimer, ajouter des enregistrements aux tables,..).
- **SI banque :** collabore dans le processus de paiement.

Notons que nous considérons ce dernier comme un acteur, puisque nous nous n'intéressons pas à cette partie.

2.4.2.2 Identification des cas d'utilisations :

Définition : Un cas d'utilisation représente un ensemble de séquences d'interactions entre le système et ses acteurs. Les cas d'utilisation permettent de décrire ce que le système devra faire, sans spécifier comment le fera.

Chacun des acteurs que nous avons définis précédemment, effectue un certain nombre de tâches qu'on résume dans le tableau suivant :

Acteurs	Tâches
Client/visiteur	**T0 :** se connecter au site **T1 :** naviguer dans le site **T2 :** rechercher un véhicule **T3 :** payer en ligne **T4 :** Annuler une commande **T5 :** créer un compte

	T6 : s'authentifier **T7 :** se déconnecter
Concessionnaire	**T8:** se connecter au site **T9** : S'authentifier **T10 :** Edition des factures **T11 :** gérer les véhicules **T12 :** se déconnecter
Administrateur	**T13 :** Se connecter **T14 :** Changer le mot de passe et/ou de l'E-mail **T15 :** gérer les table des clients et concessionnaires **T16** : se déconnecter

Tableau 3.1 : Récapitulatif des cas d'utilisation

2.4.2.3 Spécification des scénarios :

Chacun des acteurs effectue des tâches et chaque tâche est décrite par des scénarios qu'on résume comme suit :

- **Pour un visiteur :**

Tâches	Scénarios
T0 : se connecter au site	**S0 :** Taper l'adresse URL
T1 : naviguer dans le site	**S1 :** Consulter les différents liens du site
T2 : rechercher un véhicule	**S2 :** Introduire les informations nécessaires **S3 :** Cliquer sur le bouton Rechercher
T3 : payer en ligne	**S4 :** Choisir un véhicule **S5 :** Cliquer sur le bouton « Afficher les détails » **S6 :** Remplir le formulaire de la commande après inscription. **S7 :** Saisir les coordonnées de carte bancaire

	après inscription **S8 :** Valider le paiement **S9 :** Imprimer la facture
T4 : créer un compte	**S10 :** Cliquer sur le lien « s'inscrire » **S11:** Accéder au formulaire d'inscription **S12 :** Remplir le formulaire

Tableau 3.2 : Récapitulatif des scénarios par tâches d'un visiteur

- **Pour un client :**

Tâches	Scénarios
Idem que les tâches du visiteur	**S0 ;** . . **S3 ;**
T5 : Payer en ligne	**S13 :** Choisir un véhicule **S14 :** Cliquer sur le bouton « Afficher les détails » **S15 :** Remplir le formulaire de la commande **S16 :** Cliquer sur le bouton « Payer en ligne » **S17 :** Saisir les coordonnées de carte bancaire. **S18 :** Valider le paiement **S19 :** Annuler le paiement **S20 :** Imprimer la facture
T6 : S'authentifier	**S21 :** Saisir l'E-mail et le mot de passe **S22 :** Accéder à l'espace Client
T7 : Annuler une commande	**S23** : Cliquer su le lien Mes Commandes **S24 :** Cliquer sur le bouton Annuler
T8 : Se déconnecter	**S25 :** Cliquer sur le bouton déconnecter

Tableau 3.3 : Récapitulatif des scénarios par tâches d'un Client

∨ Pour un concessionnaire :

Tâches	Scénarios
T9: se connecter au site	**S26 :** Taper l'adresse URL
T10 : s'authentifier	**S27 :** Saisir l'E-mail et le mot de passe **S28 :** Accéder à l'espace concessionnaire
T11 : Edition des factures	**S29** : Cliquer sur le bouton « Consulter les commandes» **S30** : Visualiser les commandes des clients et leurs états. **S31 :** Cliquer sur le bouton « Etablir une facture » **S32 :** Régler la facture **S33 :** Imprimer la facture
T12 : gérer les véhicules	**S34 :** Cliquer sur le lien Mes annonces **S35 :** Ajouter/ Supprimer/ Modifier un véhicule.
T13 : Se déconnecter	**S36 :** Cliquer sur le bouton déconnecter

Tableau 3.4 : Récapitulatif des scénarios par tâches d'un Concessionnaire

∨ Pour un administrateur :

Tâches	Scénarios
T14 : Se connecter	**S37** : Saisir l'E-mail et le mot de passe **S38 :** Accéder à l'interface administrateur
T15 : Changer le mot de passe et/ou l'E-mail	**S39** : Cliquer sur le lien changer le mot de passe et/ou l'E-mail. **S40 :** Remplir le formulaire de changement de mot de passe et/ou l'E-mail. **S41 :** Valider le changement.

T16 : gérer les tables clients, concessionnaires	**S42 :** Sélectionner le lien Gestion des clients. **S43 :** Ajouter/ Supprimer/ Modifier un compte client. **S44** : Sélectionner le lien Gestion des concessionnaires. **S45 :** Ajouter/ Supprimer/ Modifier un compte concessionnaire.
T17 : Se déconnecter	**S46 :** Cliquer sur le lien Déconnexion

Tableau 3.5 : Récapitulatif des scénarios par tâches de l'Administrateur

2.4.2.4 Spécification des cas d'utilisation :

On présentera ci-dessous, une description de quelques cas d'utilisation

- **Cas d'utilisation « Créer un compte client» :**

Use case: Créer un compte client.
Scénario: $S_{10}.S_{11}.S_{12.}$
Acteur: Visiteur.
Résumé : Cette fonctionnalité permet aux visiteurs non inscrits d'ouvrir un compte.
Description :

- Le visiteur saisie l'URL du site.
- Le système lui affiche la page d'accueil du site.
- Il clique ensuite sur le lien « S'inscrire ».
- Le système le dirigera vers le formulaire d'inscription.
- Il remplit le formulaire et il l'envoie.
- Le système intervient pour faire des vérifications et introduire le nouveau client à la base de données.
- Le visiteur devient alors client.

v Cas d'utilisation « S'authentifier » :

Use case: S'authentifier.
Scenario: $S_{21}.S_{22}$.
Acteur : Client.
Résumé : Cette fonctionnalité permet aux clients d'entrer dans leurs espaces personnels, pour pouvoir gérer les commander des véhicules.
Description :

- Le Client saisie l'URL du site.
- Le système lui affiche la page d'accueil.
- Le Client saisit son E-mail et mot de passe, puis il clique sur le bouton « Se connecter » dans l'espace d'authentification.
- Le système vérifie les données, les compare avec celles de la base de données, puis lui attribut son espace.

v Cas d'utilisation « Payer en ligne » :

Use case: Payer en ligne.
Scenario: S_{13}. S_{14}. $S_{15.}S_{16}$. $S_{17}.S_{118}$.
Acteur : Client.
Résumé : Cette fonctionnalité permet au client d'acheter un véhicule en ligne.
Description :

- Après authentification.
- Le système affiche l'espace client concerné.
- Le client choisit un véhicule, il clique sur le bouton « Afficher les détails ».
- Le système lui affiche la fiche technique (Prix, Equipement...) et le formulaire de commande.
- Le client remplit le formulaire ensuite il clique sur le bouton « Payer ».
- Le système lui affiche la page de paiement sécurisée.
- Le client saisit ses coordonnées bancaires et clique sur le Bouton « Valider le paiement ».
- Le système invoque le service de paiement

- **Cas d'utilisation « Edition des factures » :**

Use case: Edition des factures.
Scenario: $S_{29}.S_{30.}S_{31}.S_{32.}S_{33}$.
Acteur : Concessionnaire.
Résumé : Cette fonctionnalité permet au concessionnaire de visualiser les commandes passées par le client et générer les factures.
Description :

- Après authentification.
- Le concessionnaire sélectionne le bouton« Consulter Mes commandes ».
- Le système lui affiche un tableau des commandes ainsi que leurs états.
- Le concessionnaire clique sur le bouton : Etablir une facture.
- Le système affiche un formulaire.
- Le concessionnaire règle le formulaire.
- Le concessionnaire sélectionne sur le bouton « Imprimer ».

- **Cas d'utilisation « Changer le mot de passe et/ou l'E-mail» :**

Use case: Changer le mot de passe et/ou de l'E-mail.
Scenario: $S_{39}.S_{40}.S_{41}$.
Acteur: Administrateur.
Résumé : Cette fonctionnalité permet à l'administrateur de changer son E-mail et/ou son mot de passe.
Description :

- L'administrateur clique sur le lien changer le mot de passe et/ou l'E-mail.
- Le système lui affiche un formulaire qui contient un champ E-mail et un autre mot de passe à remplir.
- L'administrateur remplit ce formulaire et le valide.
- Le système lui confirme le changement en renvoyant un message à l'administrateur.

3.4.2.5 Diagramme des cas d'utilisation:

Le diagramme des cas d'utilisation est destiné à représenter les besoins des utilisateurs par rapport au système. Il est considéré comme l'un des diagrammes les plus structurants dans l'analyse d'un système.

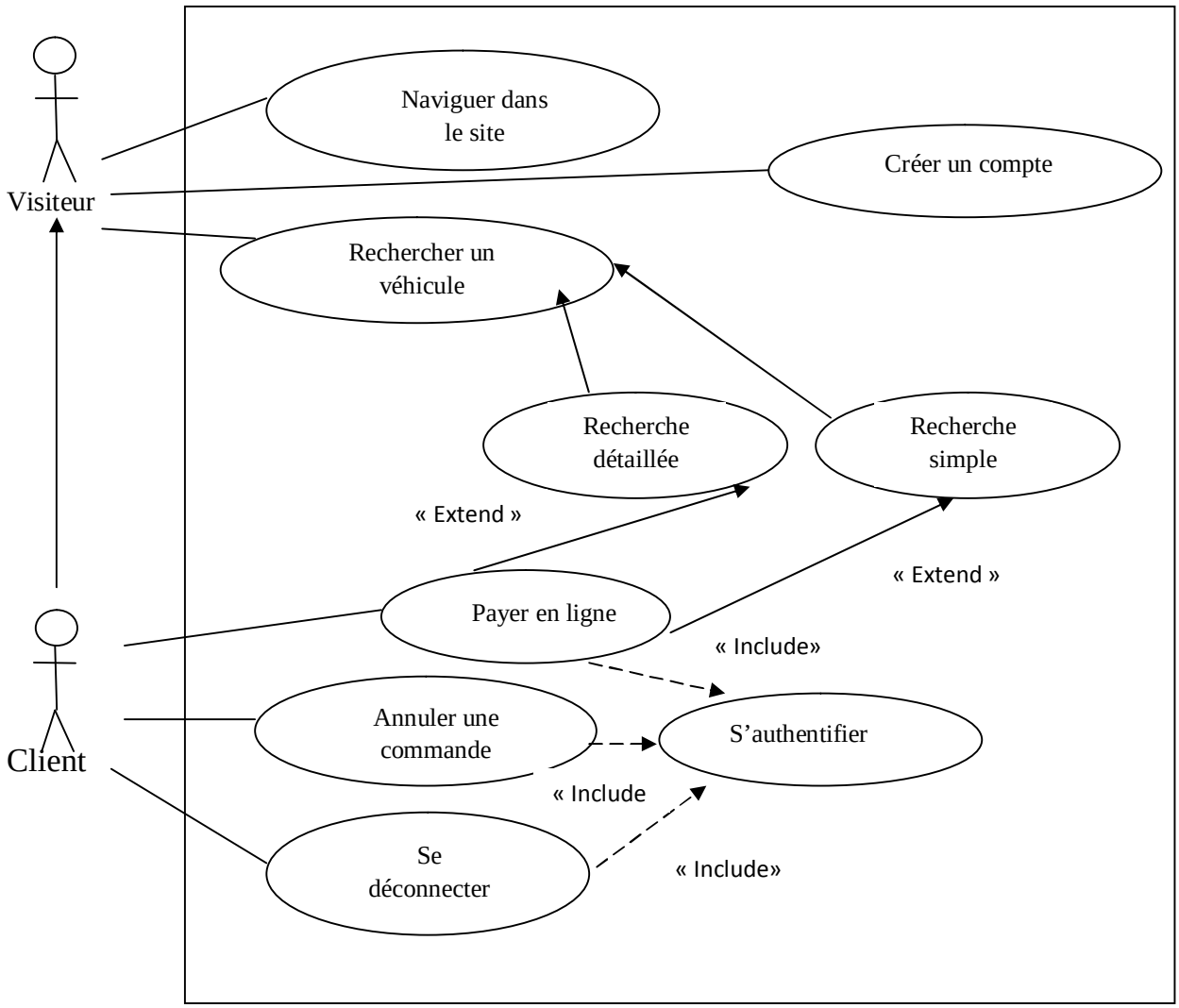


Figure 3.2: Diagramme de cas d'utilisation de Client

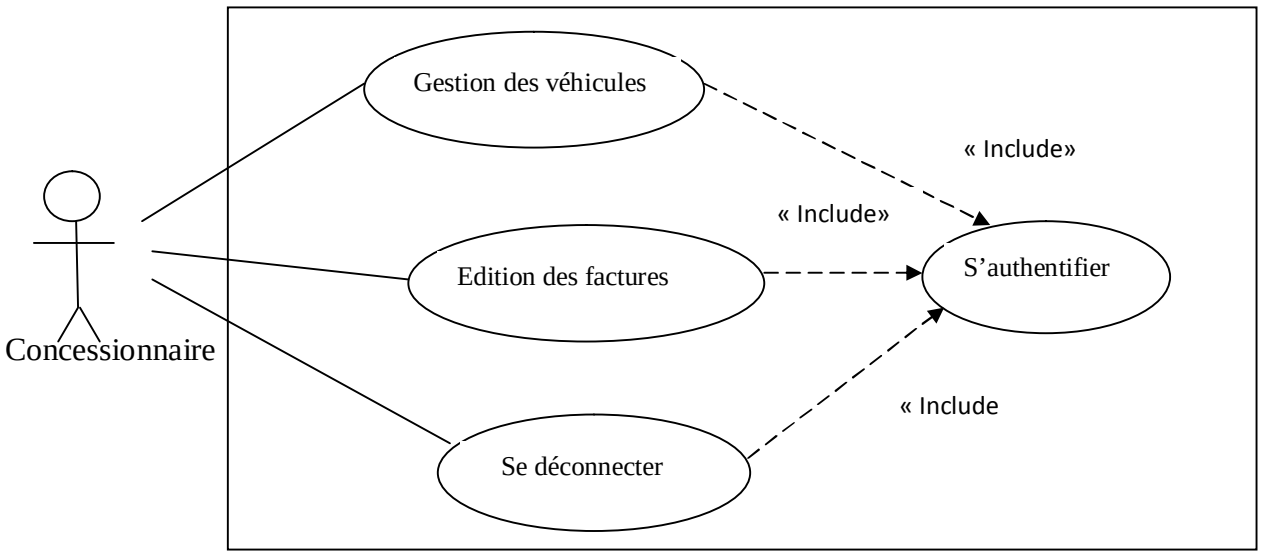


Figure 3.3: Diagramme de cas d'utilisation de concessionnaire

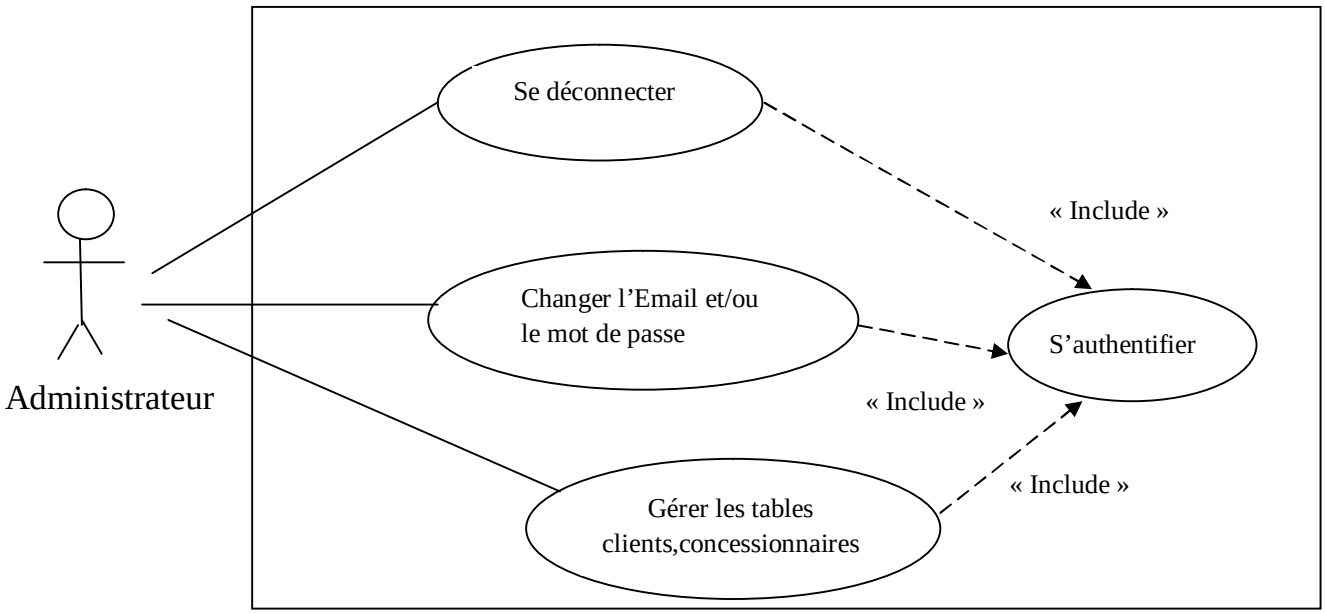


Figure 3.4: Diagramme de cas d'utilisation d'Administrateur

3.5 Conception :

Le but principal de la conception est de rendre le modèle d'analyse réalisable sous forme logicielle. Pour modéliser la conception de notre application, nous avons utilisé l'extension

d'UML pour le Web, et ce en construisant les diagrammes de séquence suivis des diagrammes de classes.

3.5.1 Diagrammes de séquence de quelques cas d'utilisation :

Les diagrammes des séquences documentent les interactions à mettre en œuvre entre les classes pour réaliser un résultat, tel qu'un cas d'utilisation. UML étant conçu pour la programmation orientée objet, ces communications entre les classes sont reconnues comme des messages.

L'ordre d'envoi d'un message est déterminé par sa position sur l'axe vertical du diagramme ; Le temps s'écoule "de haut en bas" de cet axe.

Vu le nombre important de cas d'utilisations qu'on a, nous limitons à représenter les cas d'utilisations suivants:

- Créer compte client.
- S'authentifier.
- Rechercher un véhicule.
- Annuler une commande.

Pour chacun des cas d'utilisations qu'on vient de citer, nous allons lui représenter son diagramme de séquence.

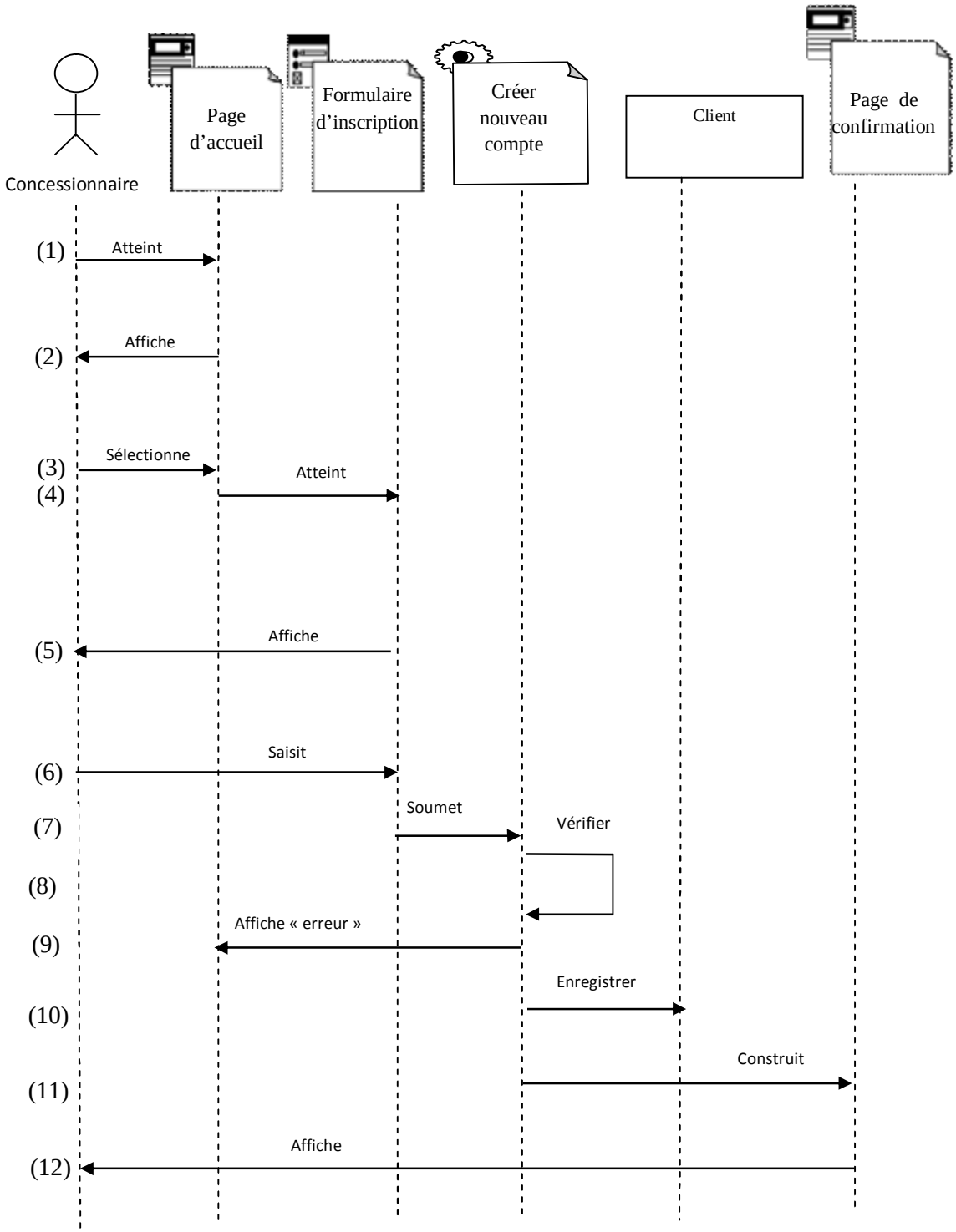


Figure 3.5 Diagramme de séquence de cas d'utilisation « Créer compte client »

(1), (2), (3), (4): Le client atteint la page d'accueil et sélectionne le lien « S'inscrire ».

(5) : Le système affiche le formulaire d'inscription de client.

(6), (7) : Le client remplit le formulaire et le soumet.

(8), (9), (10) : Le système vérifier les données soumises puis les enregistres s'elles sont bonnes, sinon il affiche un message d'erreur.

(11), (12) : Le système construit un message de confirmation et il l'affiche.

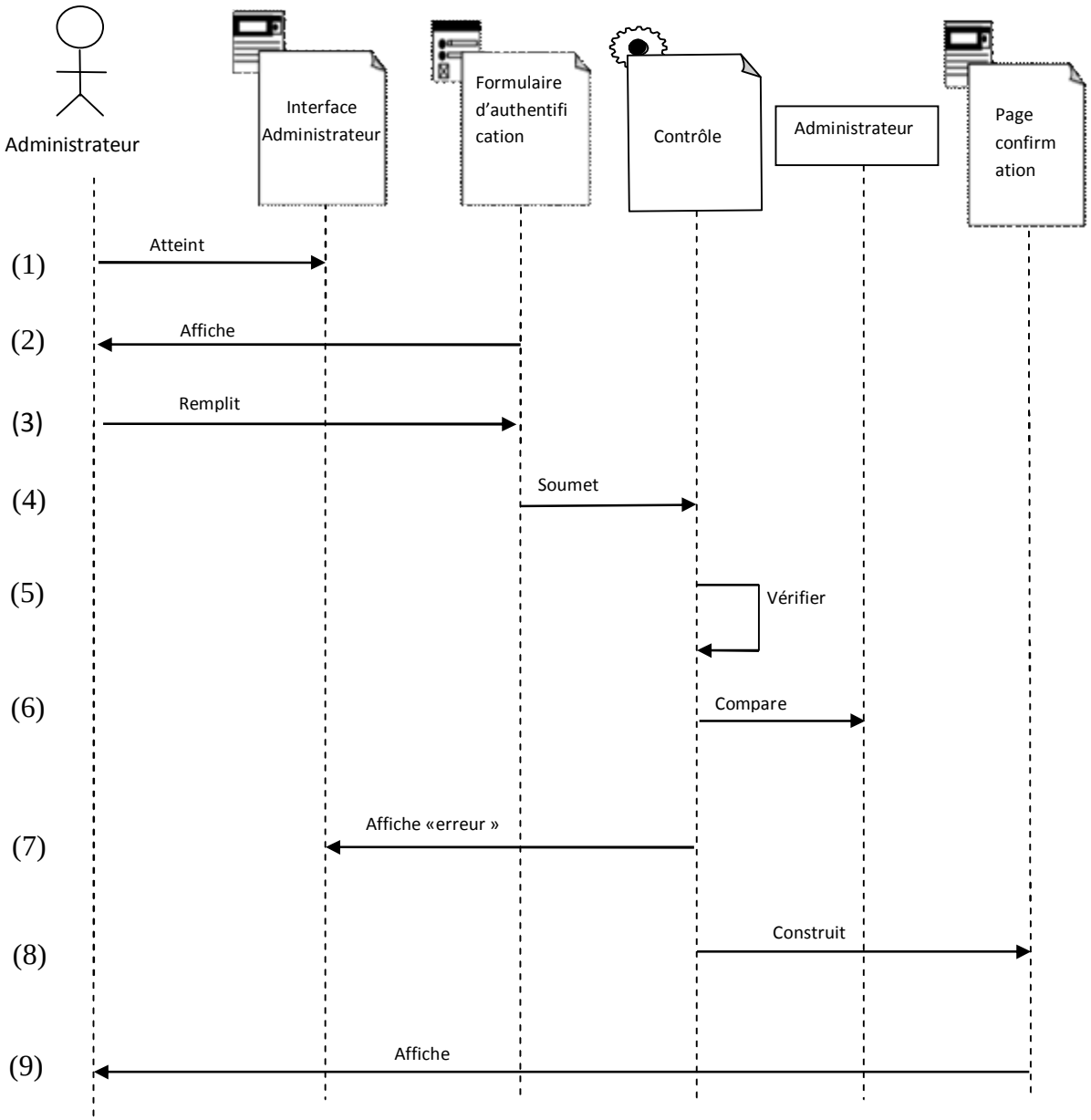


Figure 3.6 Diagramme de séquence de cas d'utilisation «S'authentifier »

(1) : L'administrateur atteint son interface

(2) : Le système lui affiche le formulaire pour authentification.

(3), (4) : L'administrateur remplit le formulaire puis le soumet.

(5), (6), (7), (8) : Le système contrôle les informations soumises, si les informations sont erronées. Le système lui affiche un message d'erreur, sinon il atteint la page d'accueil.

(9) : Le système affiche la page d'accueil administrateur.

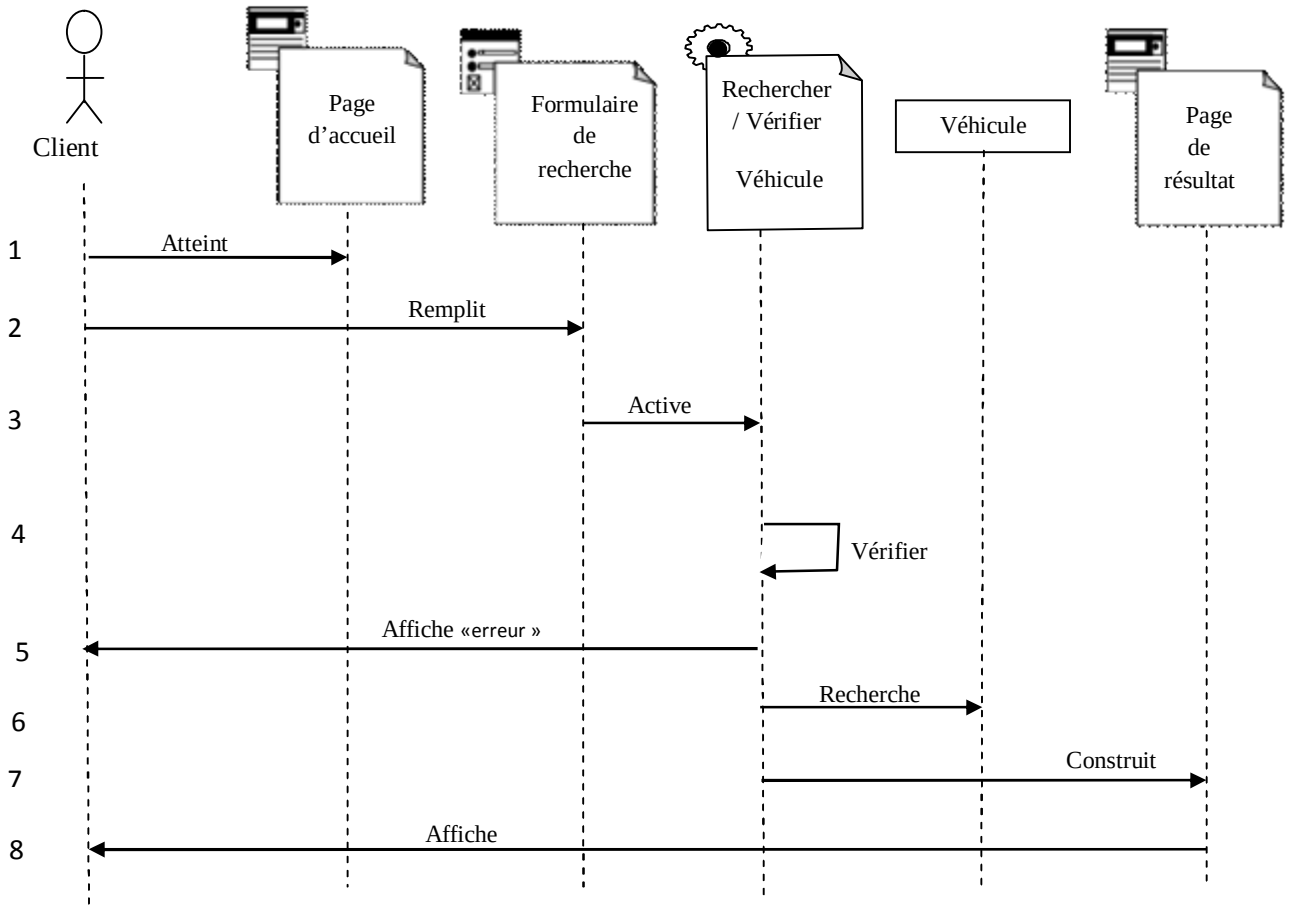


Figure 3.7 Diagramme de séquence de cas d'utilisation «Rechercher un véhicule »

(1): Le client atteint la page d'accueil.

(2), (3) : Le client remplit le formulaire puis active la recherche.

(4), (5), (6), (7) : Le système effectue une vérification des données du formulaire. En cas de données erronées, il affiche un message d'erreur sinon il fait une recherche sur l'existence de véhicule et construit la page de résultat.

(8) : Le système affiche le résultat de la recherche.

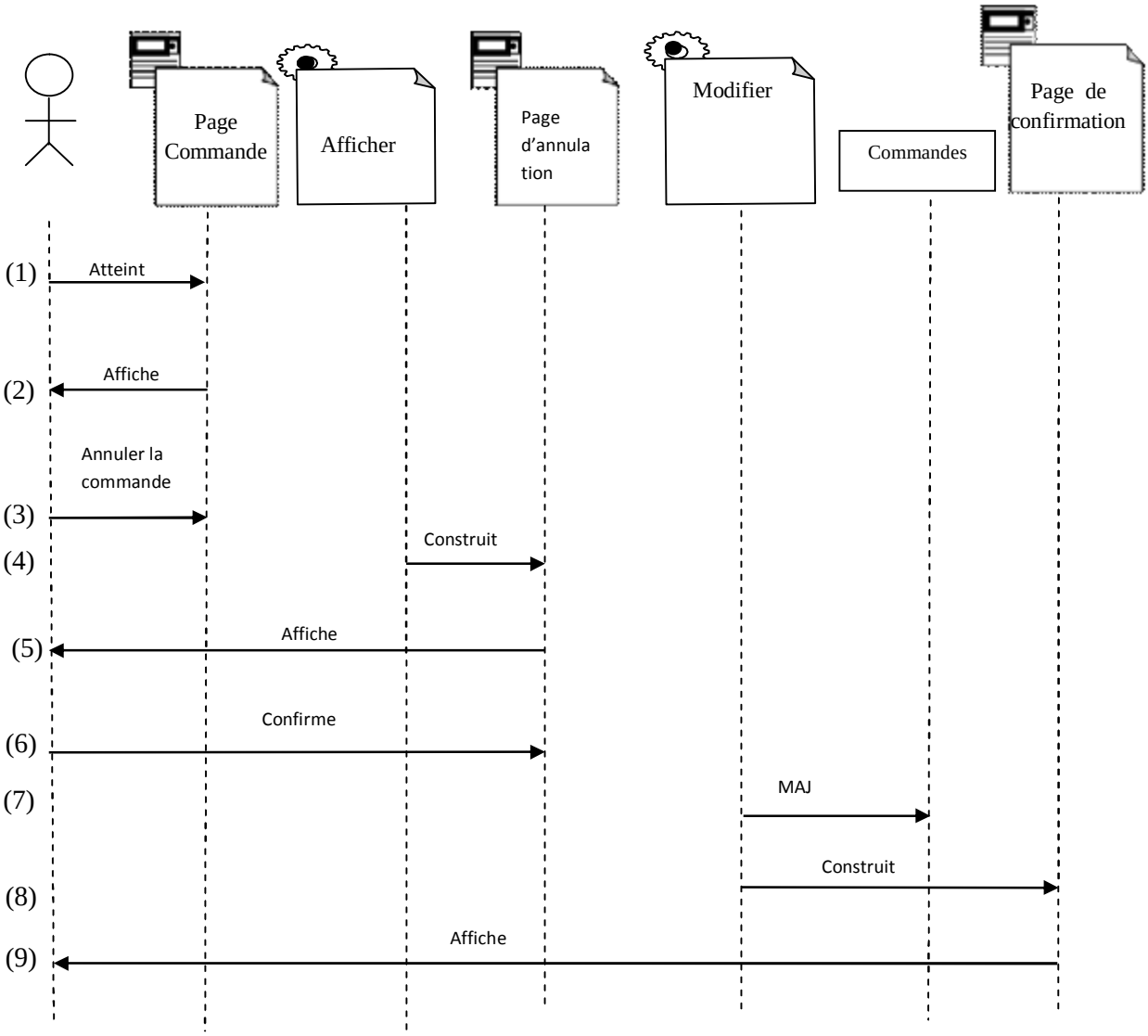


Figure 3.8 Diagramme de séquence de cas d'utilisation « Annuler une commande »

(1), (2) : Le client atteint la page commande et le système la lui affiche.

(3) : Le client clique sur le bouton « Annuler ».

(4), (5) : Le système lui affiche un message pour confirmer l'annulation.

(6): Le client clique sur le bouton « ok ».

(7), (8), (9) : Le système mis à jour les table des commandes effectuées par le client ensuite il affiche un message de confirmation.

3.5.2 Quelques diagrammes de classe :

Un diagramme de classe montre la structure statique d'un système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C'est donc le diagramme pivot de l'ensemble de la modélisation d'un système.

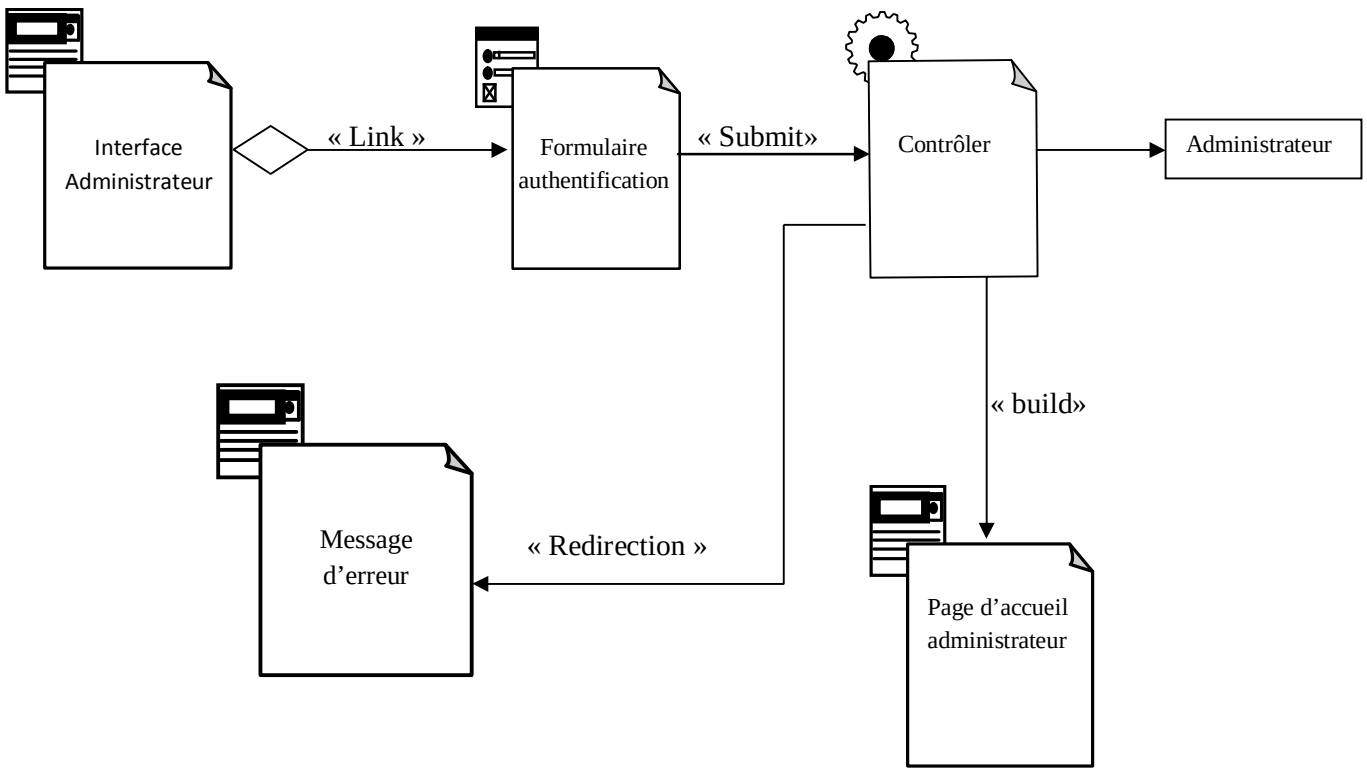


Figure 3.9 Diagramme de classe du cas d'utilisation « S'authentifier »

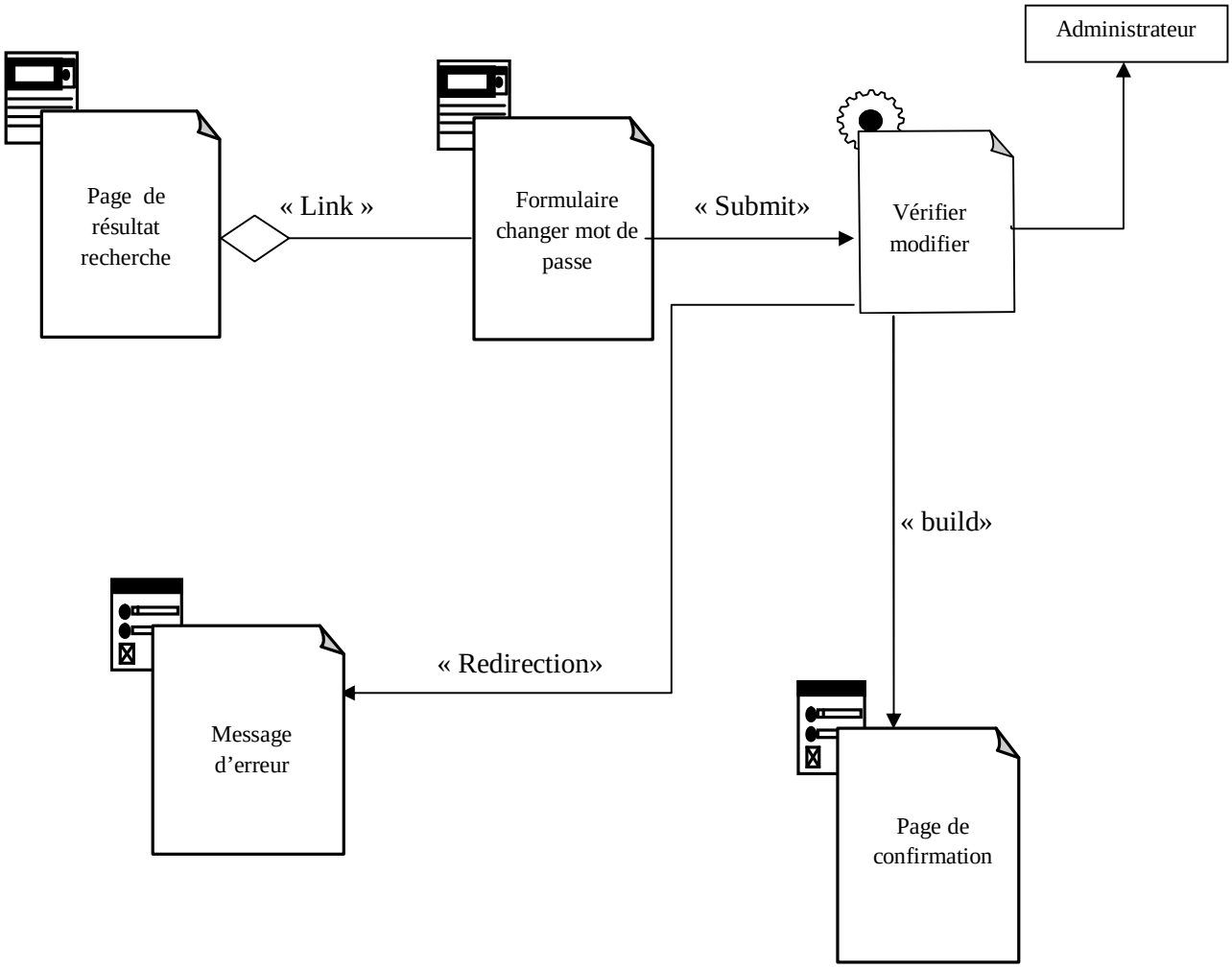


Figure 3.10 Diagramme de classe du cas d'utilisation « Changer mot de passe »

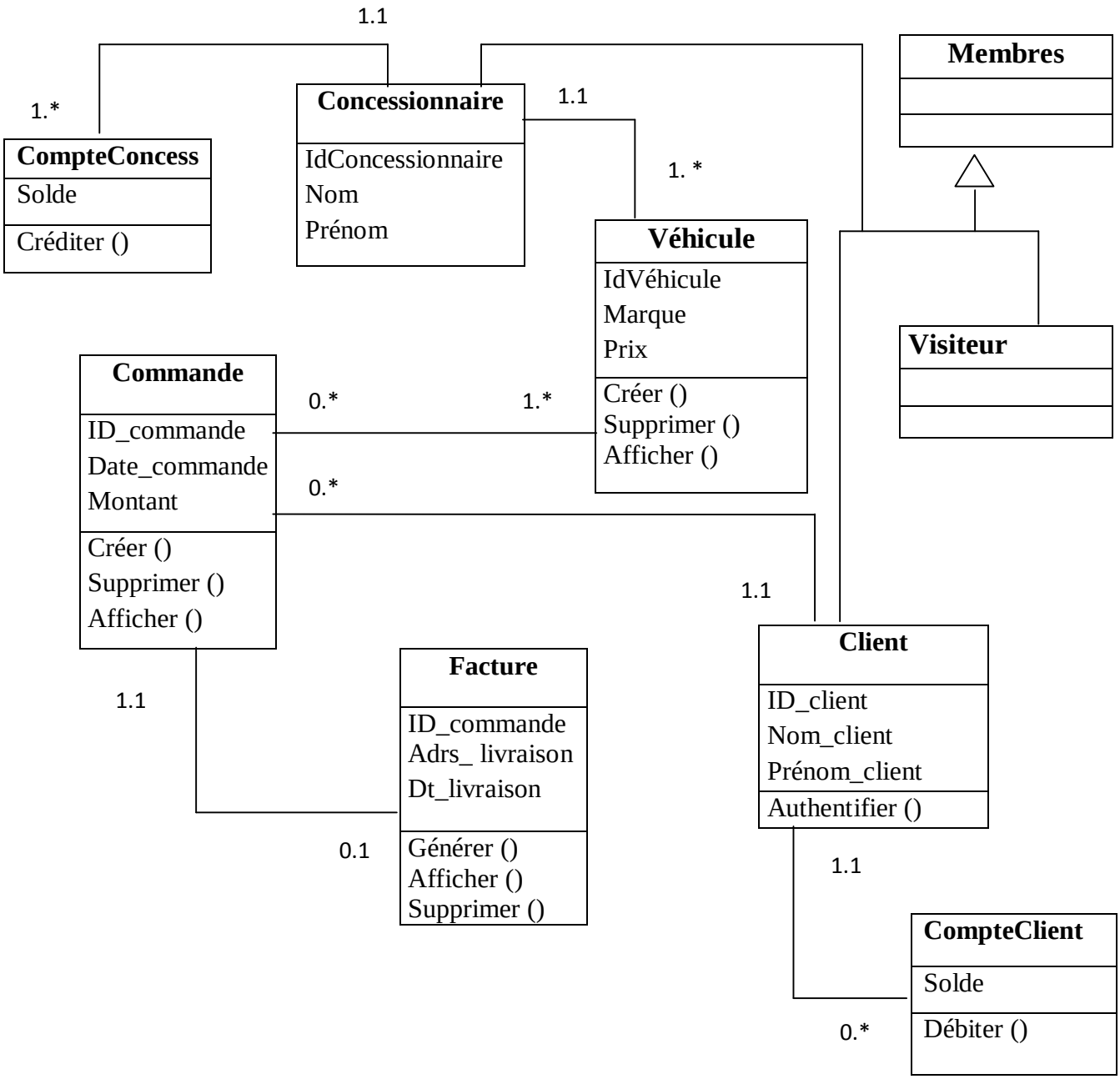


Figure 3.11 Diagramme de classe globale du système

3.5.3 Diagramme d'activité :

Un diagramme d'activités permet de décrire une méthode pour laquelle peu de classes différentes interviennent. Ce diagramme se révèle efficace lorsque la méthode met en jeu un algorithme avec des étapes et de nombreuses alternatives.

La figure qui suit représente le diagramme d'activité « payer en ligne ».elle décrit les activités principales qui sont déroulées au cours de paiement d'une commande par un client.

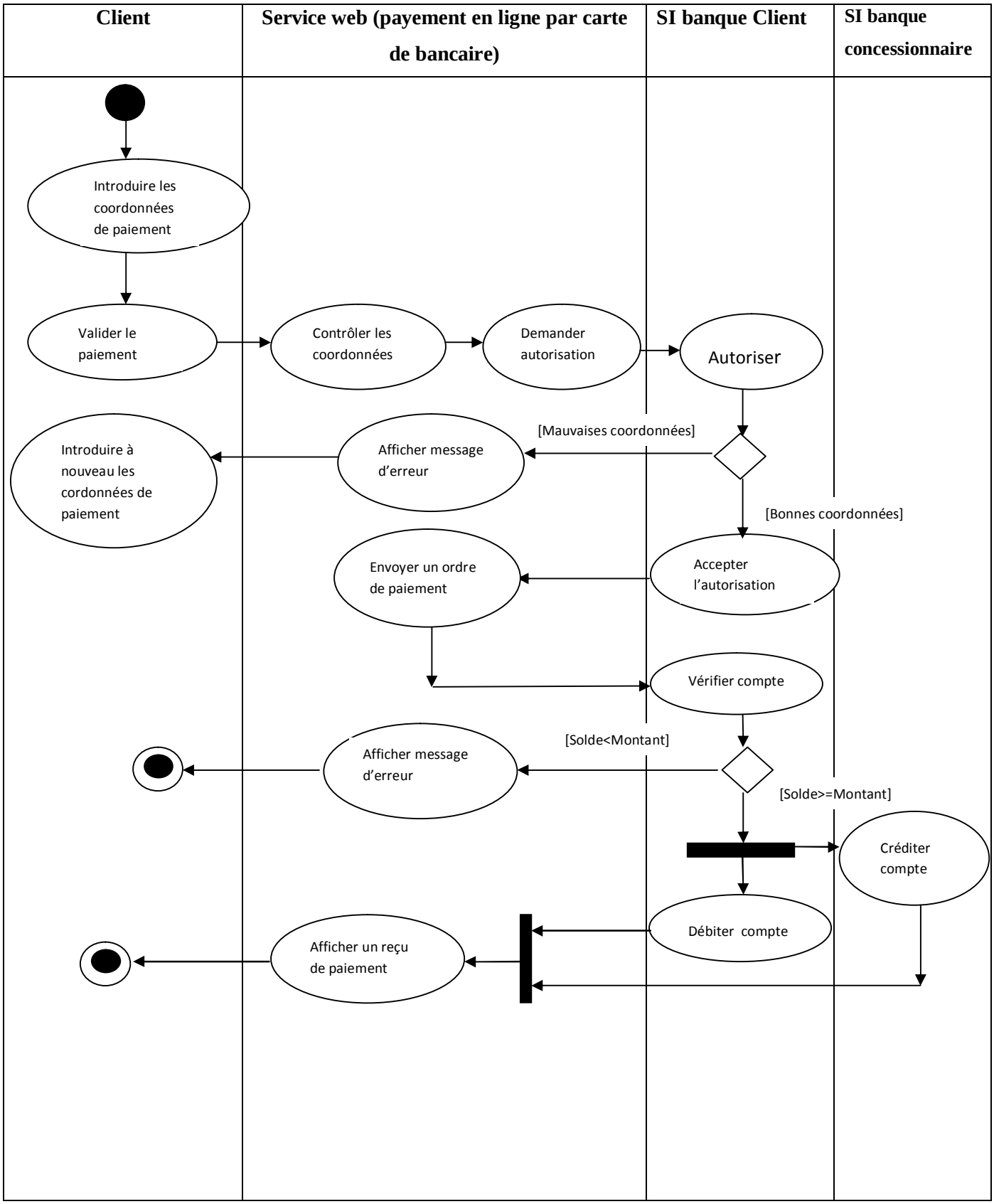


Figure 3.12 Diagramme d'activité de Paiement en ligne

3.6 Conclusion :

Dans ce chapitre, nous avons présenté une démarche de modélisation pour développer notre application, cette démarche est basée sur le langage de modélisation UML pour le Web.

Pour l'étape d'analyse de notre application, nous avons commencé par la définition des scénarios qui concernent chaque acteur, ensuite on a présenté la spécification pour les cas d'utilisations qu'on a choisit, et on a terminé l'analyse par les diagrammes des cas d'utilisations.

Pour la conception, Nous avons représenté les diagrammes de séquences suivis par celui des classes, ensuite on a représenté le processus de paiement par un diagramme d'activité. On a aussi représenté le diagramme de classe global.

Le dernier chapitre de notre mémoire sera consacré pour la présentation de modèle logique de données, les outils et l'environnement de développement de notre application.

Chapitre 4 : Réalisation

4.1 Introduction :

Après avoir fait une étude détaillée et avoir défini une conception mieux appropriée aux besoins de l'application, nous allons nous pencher sur la mise en œuvre de notre système; en présentant d'abord l'environnement de travail, les différents outils de développement utilisés et notre base de données implémentée, puis expliquer le fonctionnement de l'application en présentant quelques interfaces illustratives.

4.2 Conception de la base de données :

A fin d'implémenter notre base de données, on aura donc besoin d'élaborer un modèle logique de données.

4.2.1 Le modèle logique de données :

Le modèle logique nous permet d'extraire les attributs, leurs appartenances aux différentes classes, ainsi que les associations entre celles-ci.

Pour notre cas, nous avons définis le modèle logique suivant :

- **Client** (ID_client, Civilité, Nom, Prénom, E_mail, Mot_de_passe, Debit_client, LoginSW_client, PasswordSW_client)
- **Concessionnaire** (ID_concessionnaire, Nom_entreprise, Nom, Prénom, Date_naissance, Lieu_naisssance, Adresse, Num_tel, Num_fax, E_mail, Site_web, Num_registre_commerce, Mot_de_passe, Debit_concessionnaire, LoginSW_concessionnaire, PasswordSW_concessionnaire)
- **Administrateur (**ID_admin, Nom, Prénom, E_mail, Mot_de_passe)
- **Véhicule (**ID_véhicule, Carosserie, Marque, Modèle, Version, Carburant, Année, Couleur_exterieure, Equipement, Localisation, Critère_supplimentaire, Certificat_de_qualité, Prix, Image, ID_concessionnaire)
- **Commande (**ID_commande, Date_commande, Montant, ID_client, Etat_commande, ID_concessionnaire, civilite, Nom, Prenom, Adresse, E_mail, Code_postal, Num_tel)
- **Facture (**ID_commande, ID_client, Adresse_livraison, Date_livraison, ID_concessionnaire)

Les tables de la base de données seront alors :

- **Client :**

Champs	Type de données	Description	Contrainte
ID_client	int(10)	Identifiant Client	**PK**
Civilité	varchar(30)	La civilité du client	
Nom	varchar(50)	Nom du client	
Prénom	Varchar(50)	Prénom du client	
E_mail	varchar(100)	Adresse Email du client	
Mot_de_passe	varchar(20)	Mot de passe du client	
PasswordSW_client	varchar(16)	Mot de passe de paiement (coordonnée bancaire)	
Debit_client	Double	Solde client	
LoginSW_client	varchar(16)	Login de paiement (coordonnée bancaire)	

- **Concessionnaire :**

Champs	Type de données	Description	Contrainte
ID_concessionnaire	int(10)	Identifiant Concessionnaire	**PK**
Nom_entreprise	varchar(100)	Le nom d'entreprise de concessionnaire	
Nom	varchar(50)	Nom du concessionnaire	
Prénom	varchar(50)	Prénom du concessionnaire	
Date_naissance	Date	Date de naissance de concessionnaire	
Lieu_naissance	varchar(100)	Lieu de naissance de concessionnaire	
Adresse	Text	Adresse de concessionnaire	
Num_tel	varchar(20)	Numéro de téléphone de concessionnaire	
Num_fax	varchar(20)	Numéro de fax de concessionnaire	
E_mail	varchar(50)	Adresse Email du client	
Site_web	varchar(100)	Site web de concessionnaire	

Num_registre_commerce	int(50)	Numéro de registre de commerce	
Mot_de_ passe	Varchar(20)	Mot de passe du concessionnaire	
PasswordSW_concessionnaire	varchar(16)	Mot de passe de paiement (coordonnée bancaire)	
LoginSW_concessionnaire	varchar(16)	Login de paiement (coordonnée bancaire)	
Debit_concessionnaire	Double	Solde concessionnaire	

- **Administrateur :**

Champs	Type de données	Description	Contrainte
ID_admin	int(10)	Identifiant Administrateur	**PK**
Nom	varchar(50)	Nom d'administrateur	
Prénom	Varchar(50)	Prénom d'administrateur	
E_mail	varchar(100)	Adresse Email d'administrateur	
Mot_de_passe	varchar(20)	Mot de passe d'administrateur	

- **Véhicule :**

Champs	Type de données	Description	Contrainte
ID_vehicule	int(11)	Identifiant Véhicule	**PK**
Carosserie	varchar(20)	La forme de véhicule	
Marque	varchar(40)	La marque de véhicule	
Modele	varchar(40)	Le modèle de véhicule	
Version	varchar(20)	La version de véhicule	
Carburant	varchar(40)	Type de carburant	

Année	varchar(15)	L'année de sortie	
Couleur_exterieure	varchar(20)	Couleur extérieur de véhicule	
Equipement	varchar(20)	Equipement de véhicule	
Localisation	varchar(20)	La wilaya	
Critère_suplémentaire	varchar(20)	Critères de véhicules supplémentaires	
Certificat_de_qualite	varchar(20)	Le certificat de qualité de véhicule	
Prix	double	Le prix de véhicule	
Image	blob	L'image de véhicule	
ID_concessionnaire	Int(11)	Identifiant concessionnaire	

- **Commande :**

Champs	Type de données	Description	Contrainte
ID_commande	int(11)	Identifiant Client	**PK**
Date_commande	Date	La date de la commande	
Montant	Double	Le montant de la commande	
ID_client	Int(10)	Identifiant Client	
Etat_commande	Bit	Commande payée ou non	
ID_concessionnaire	Int(11)	Identifiant concessionnaire	
Civilité	varchar(10)	La civilité du client	
Nom	varchar(30)	Nom de client	
Prénom	varchar(30)	Prénom de client	
Adresse	text	Adresse de client	
E_mail	varchar(50)	Adresse Email du client	
Code_postal	Int(11)	Code postal de	

		client	
Num_tel	varchar(15)	Numéro de téléphone de client	

Ø **Facture :**

Champs	Type de données	Description	Contrainte
ID_commande	int(11)	Identifiant Commande	**PK**
ID_concessionnaire	int(11)	Identifiant concessionnaire	**PK**
ID_client	int(11)	Identifiant Client	**PK**
Adresse livraison	Text	Adresse de livraison	
Date livraison	Date	Date de livraison	

4.3 Environnement de travail :

Pour l'environnement logiciel, nous avons opté pour la configuration suivante :

a) Configuration logicielle :

- NetBeans 7.3
- Adobe Flash Builder 4.0
- Flex BlazeDS
- Axis 2
- Apache Tomcat
- Serveur de base de données : Wamp, contenant à son tour :
  - § Un serveur MySQL.
  - § Une interface graphique PHPMyAdmin.

b) Langages de programmation :

- *MXML* et *ActionScript* 3.0 *:* pour côté client.
- *Java* : pour le traitement de données.
- *SQL* : pour l'intégration de la base de données.

On donne ci-dessous un bref aperçu sur l'ensemble des logiciels, langages et outils qu'on vient de citer.

4.3.1 Environnement logiciel :

Ø NetBeans 7.3 : est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages comme comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring moderne, éditeur graphique d'interfaces et de pages Web).Conçu en java, NetBeans est disponible sous Windows Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

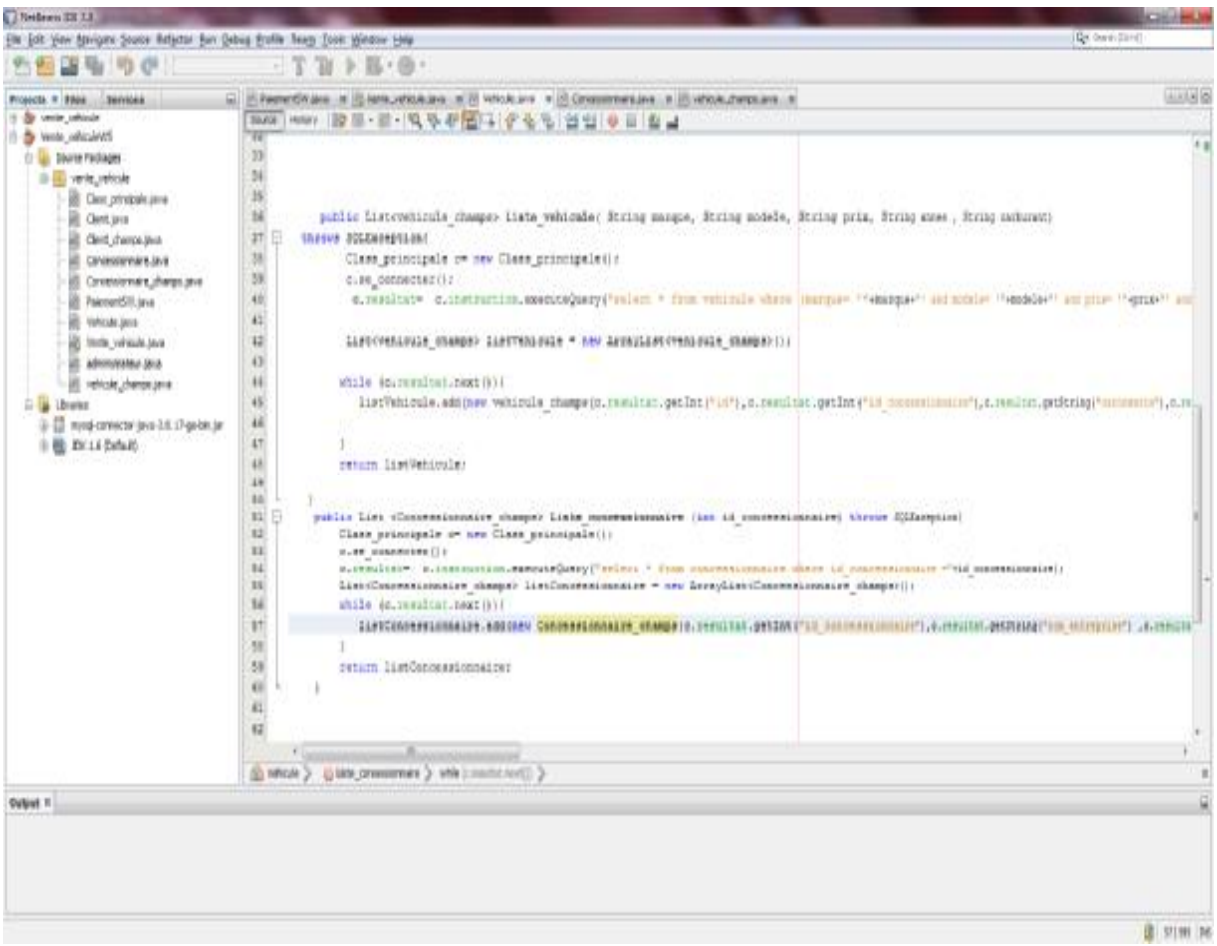


Figure 4.1 L'environnement NetBeans7.3

Ø Adobe Flash Builder 4.0 : (anciennement connu sous le nom Adobe Flex Builder) est un environnement de développement intégré(IDE) construit sur le Eclipse plate-forme qui accélère le développement d' applications Internet riches (RIA)[1] et multi-plateforme des applications bureautiques , en particulier pour l' Adobe Flash plate-forme . Adobe Flash Builder 4 est disponible en trois éditions: Standard, Premium et l'éducation.

Adobe Flash Builder offre intégrée dans les éditeurs de code pour MXML et ActionScript et un WYSIWYG éditeur pour modifier les applications MXML. Adobe Flash Builder comprend un outil interactif débogueur , permettant aux développeurs d'intensifier l'exécution de code lors de l'inspection des variables et regarder expressions. Flex Builder 3 a ajouté le support pour l'analyse des performances . La vue de profil affiche des informations statistiques sur l'utilisation de la mémoire en plus de la fonction appel moment de l'exécution.

Avant la version 4, ce produit a été connu sous le nom de Flex Builder. Le changement de nom est censé signifier sa connexion aux autres produits de l' Adobe Flash Platform et de créer une distinction claire entre le libre et open source Flex SDK et l'IDE.

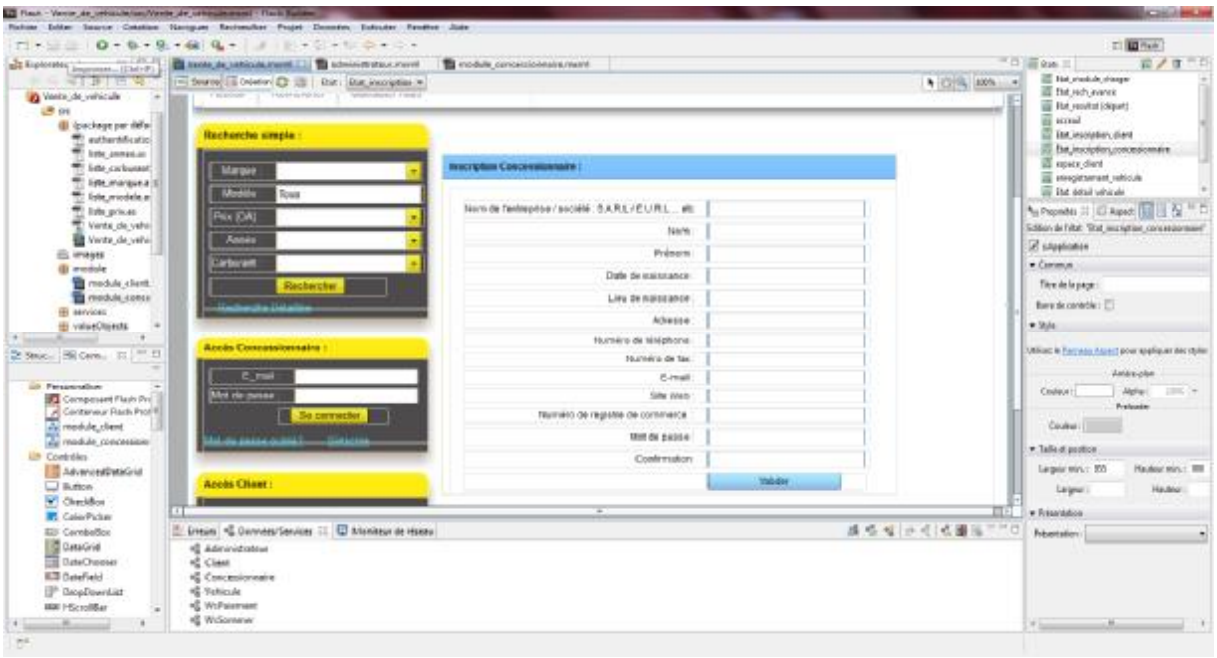


Figure 4.2 L'environnement Adobe Flash Builder

[1]**R.I.A** : sont des applications web qui s'efforcent de rapatrier chez le client (local) une partie des traitements normalement dévolus au serveur.

Ø **Axis 2 :** Axis est un ensemble de logiciels créés par Apache Software Foundation, qui vise à faciliter le développement de services Web en technologie SOAP.

Axis offre notamment :

- un environnement pouvant soit fonctionner comme un serveur SOAP/Rest[1] indépendant, soit comme un plug-in de moteurs de servlet (en particulier Tomcat),
- une API pour développer des services web SOAP RPC ou à base de messages SOAP,
- le support de différentes couches de transport : HTTP, FTP, SMTP, POP et IMAP...,
- la sérialisation/désérialisation automatique d'objets Java dans des messages SOAP,
- des outils pour créer automatiquement les WSDL correspondant à des classes Java, ou inversement, pour créer les classes Java sur la base d'un WSDL (classe proxy en quelque sorte, qui fait le lien entre l'application Java cliente et le service distant),
- des outils pour déployer, tester et monitorer des web-services.

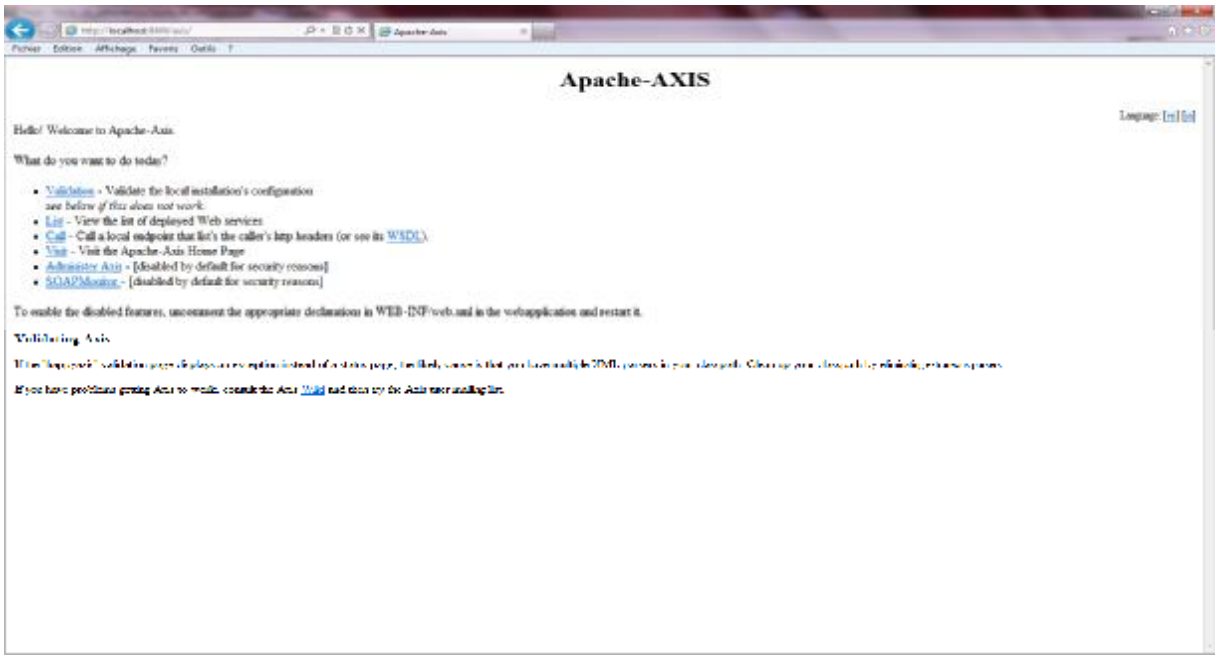


Figure 4.3 Interface Axis 2

Ø **Wamp (Windows Apache MySQL PHP):** WampServer est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement

(sans se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant deux serveurs (Apache et MySQL), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL. Pour notre cas on utilise cette plateforme pour gérer et administer le serveur de bases de données MySQL.

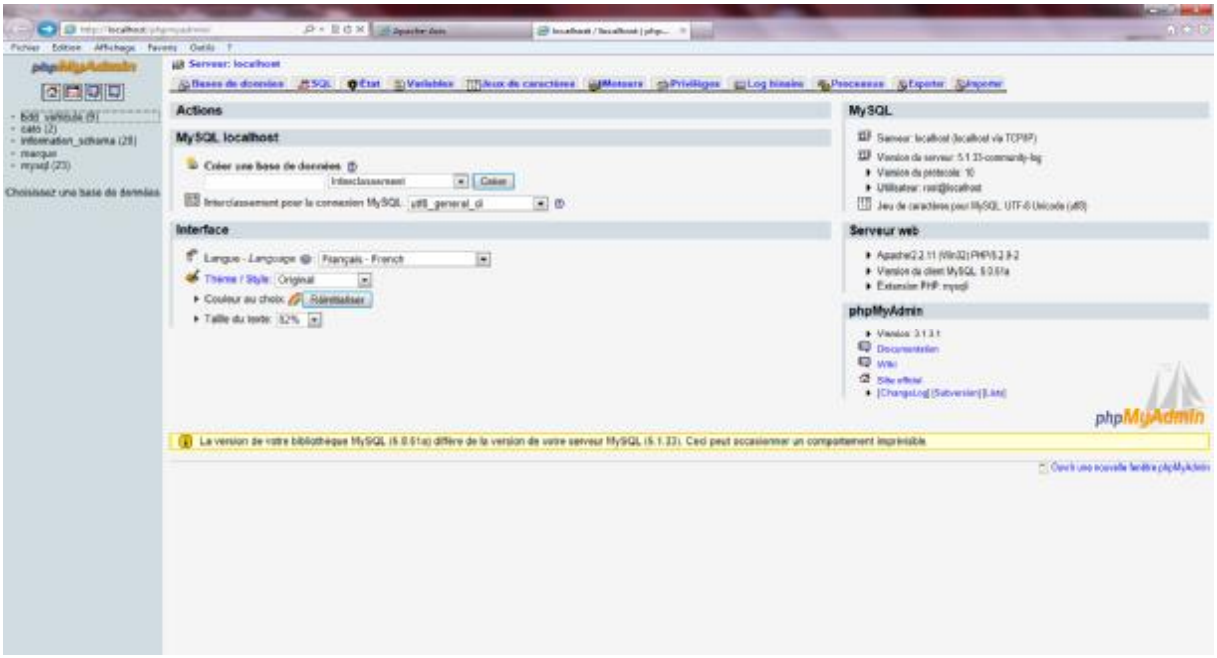


Figure 4.4 Interface phpMyAdmin

4.3.2 Les outils utilisés :

Ø **Apache Tomcat** : est un conteneur libre de servlet Java 2 Entreprise Edition. En effet, il adapte les spécifications des Servlets et des JSP (Java Server Pages) issues de Sun Microsystems. Il peut être configuré en éditant des fichiers de configuration XML. Tomcat est considéré comme un serveur d'applications Java qui assure l'exécution des Servlets et des JSP. Ainsi qu'Apache est un serveur Web : il reçoit les requêtes HTTP émises depuis un navigateur, les analyse et envoie des pages Web en réponse au navigateur.

Pour communiquer entre eux, Apache et Tomcat utilisent un module intermédiaire, tel que mod_jk.

Ø **Le serveur de base de données MySQL :** MySQL est un système de gestion de base de données (SGBD). Selon le type d'application, sa licence est libre ou propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle et Microsoft SQL Server.

Les principaux concurrents de MySQL sont : PostgreSQL, Microsoft SQL Server, et Oracle. Ainsi, le choix de ce Serveur de Bases de Données a été particulièrement

orienté par un certain nombre d'avantages qu'il offre aux développeurs. En effet, par rapport aux autres SGBD cités, MySQL est un logiciel intégrant un haut degré de portabilité, de sécurité et constitue un système de sauvegarde assez évolué avec utilisation optimale de ressources.

Ø **BlazeDS** : fournit un ensemble de services nous permettant de faire le lien entre l'application côté client et la donnée (côté serveur) ainsi que le passage de données à de multiples clients tous connectés au même serveur BlazeDS pour une messagerie temps réel entre les clients.

Une application BlazeDS se compose de deux parties: une application cliente et une application Web J2EE côté serveur.

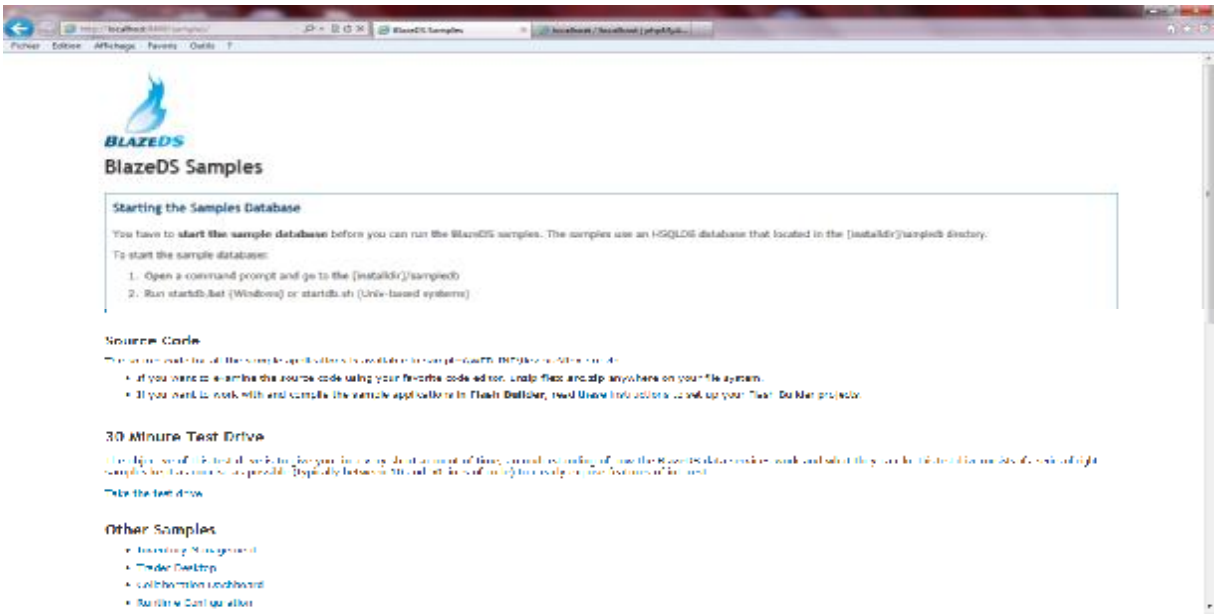


Figure 4.5 Interface BlazeDS

4.3.3 Les langages de programmation utilisés :

Ø **Java** : est un langage de développement mise en oeuvre par Sun Microsystems. Java possède une caractéristique majeure qui le distingue des autres langages de programmation est sa portabilité due à ses bibliothèques de classes indépendantes de la plate-forme, ce qui est le point essentiel de la programmation sur internet ou plusieurs machines dissemblables sont interconnectées. Le programme résultant s'exécute par la machine virtuelle (JVM pour Java Virtual Machine). Les objectifs de Java sont d'être multiplateformes et d'assurer la sécurité aussi bien pendant le développement que pendant l'utilisation d'untel programme. Java hérite partiellement la syntaxe du langage

C++ mais non ses défauts. Java est algorithmique et orienté objet et à ce niveau il peut effectuer toutes les tâches d'un tel langage (graphiques, bases de données, multimédias, environnement de développement, etc...).

Ø **Action Script 3** : est un langage de programmation utilisé au sein d'applications clientes. Ce langage permet d'ajouter de l'interactivité aux animations Flash, en répondant aux actions de l'utilisateur, et en pilotant les *movie clip*(conteneurs graphiques permettant de hiérarchiser les animations), et les différents objets multimédias (images, son, vidéo…). Il permet également la communication de l'application avec le serveur, notamment par le chargement de fichiers ou la communication avec un langage serveur comme le Java.

Ø **MXML** : est le langage de description développé par Macromedia, puis repris par Adobe Systems pour la plateforme Adobe Flex. Il est dérivé du XML et permet de décrire la présentation des interfaces utilisées dans le cadre du développement des clients riches ou RIA (Rich Internet Application)[1].

Ø **SQL** : est un langage utilisé pour effectuer des différentes opérations sur la base de données tel que la mise à jour de la base, la modification des données dans une table (ajout, suppression...etc).

4.4 Réalisation et description du système:

Nous allons dans ce qui suit, présenter quelques interfaces de notre application, en commençant par celles de visiteur, puis on passera à celles de client où le service web intervient (quand le client arrive à payer sa commande).

4.4.1 La page d'accueil :

La page d'accueil est la première page visualisée, elle présente certains services proposés par le site :
http://localhost:8400/blazeds/vente_de_vehicule_debug/vente_de_vehicule.html et elle contient deux espaces d'authentification : un pour le client et l'autre pour le concessionnaire.

Figure 4.6 Interface de la page d'accueil du site

4.4.2 Page d'inscription de visiteur :

Dans cette page le visiteur peut s'inscrire et avoir un compte, Il deviendra un client de ce site.

Figure 4.7 Interface de la page d'inscription d'un visiteur

4.4.3 Le paiement d'une commande :

Le client lance une recherche à partir de l'espace de recherche de véhicules :

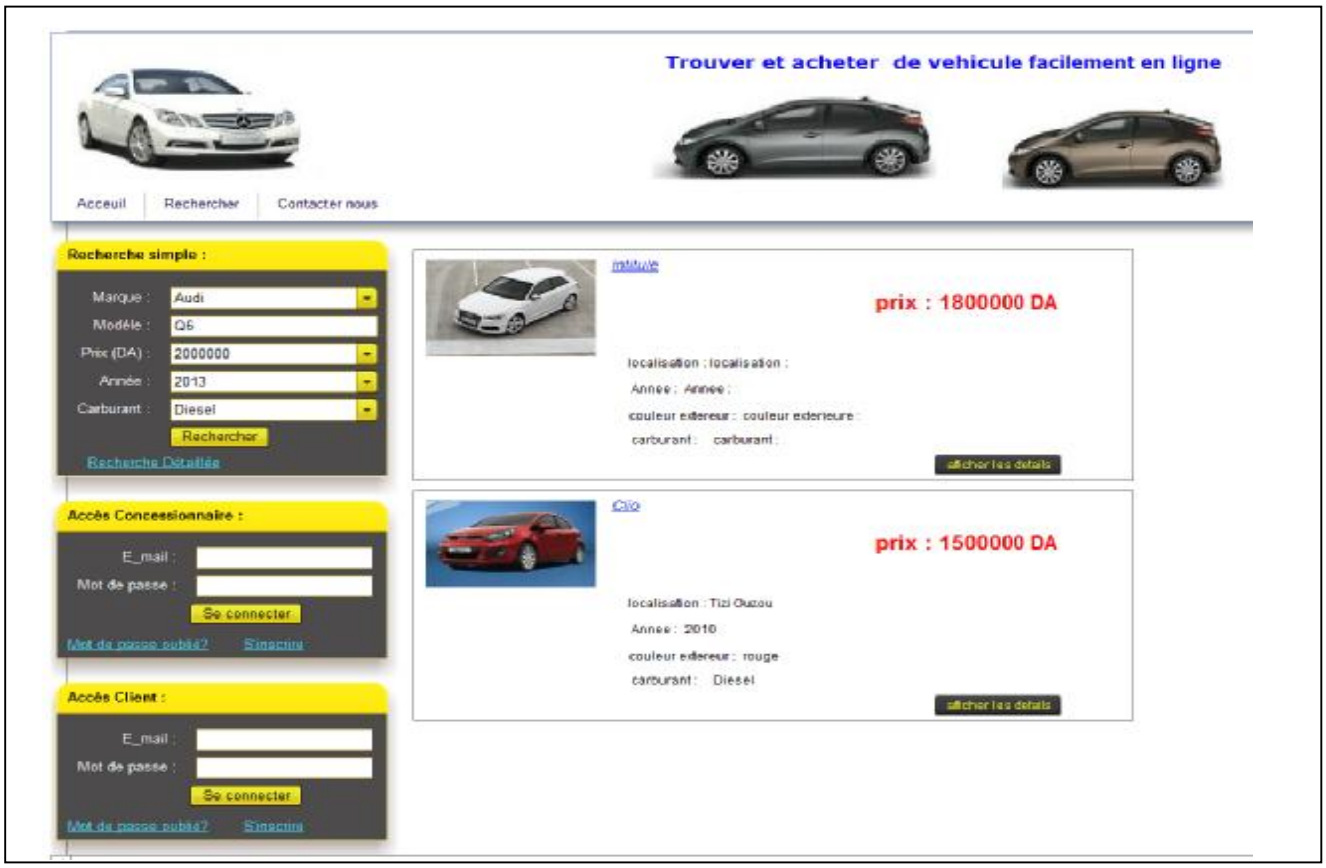


Figure 4.8 Résultats de la recherche

Le client peut sélectionner un véhicule et le paye immédiatement.

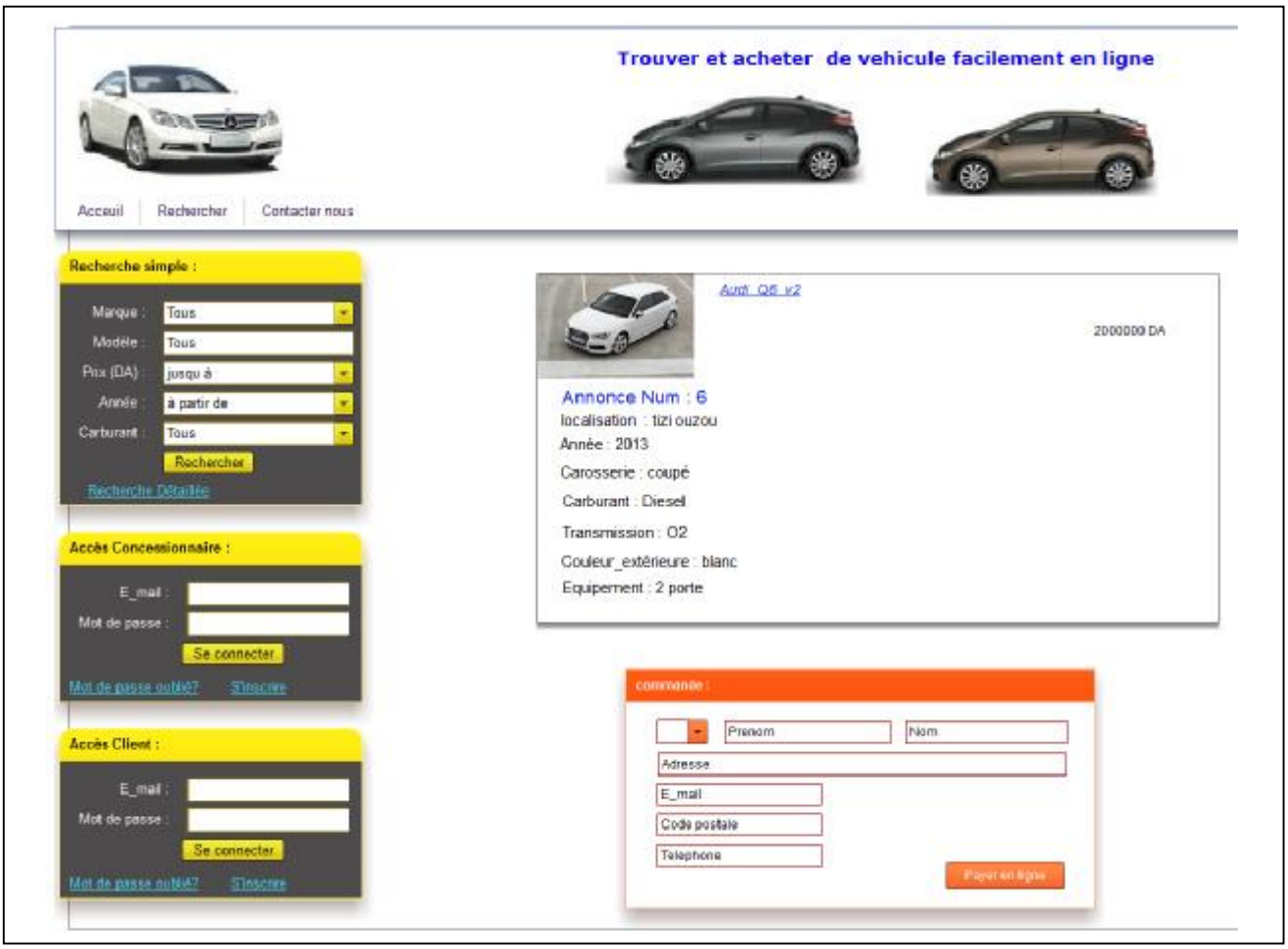


Figure 4.9 Interface de la page des détails de véhicule et formulaire de commande

Figure 4.10 Interface de paiement

Figure 4.11 Message confirmant le paiement

4.5 Conclusion :

Dans ce chapitre nous avons présenté l'environnement technique dans lequel nous avons développé notre application ainsi que les outils que nous avons utilisés. Nous avons aussi décrit quelques interfaces de notre site.

L'objectif majeur de notre application consiste à faciliter la vente et l'achat de véhicules coté client/concessionnaire, et permettre l'interopérabilité des systèmes bancaires hétérogènes (le paiement en ligne).

Au cours de notre travail, nous avons eu des difficultés qu'on a résolues par la suite, particulièrement au niveau de l'analyse et de la conception, qui sont dû essentiellement à la complexité du système (plusieurs contraintes à respecter).

Tout au long de notre travail, nos connaissances se sont enrichies concernant le développement d'applications et des services web ;

- Langage de modélisation objet UML.
- Les services web et l'interopérabilité des systèmes hétérogène.
- Les bases de données
- La programmation objet (Action Script 3/Java)
- La création d'IHM ergonomique.

Ainsi on a constaté quelques limites des services web :

- Les normes de services Web dans certains domaines sont actuellement récentes.
- Les services Web souffrent de performances faibles comparées à d'autres approches de l'informatique répartie telles que le RMI, CORBA.
- Par l'utilisation du protocole HTTP, les services Web peuvent contourner les mesures de sécurité mises en place au travers des pare-feu.

En marge de perspectives, l'intérêt croissant de vouloir gagner en temps, d'être discret, et d'autres raisons, notre application peut être étendue pour quelle seras une plateforme d'e-commerce pour un marché virtuel.

Bibliographie

[DOUDOUX, 08] : Jean Michel DOUDOUX, Développons en Java, Version 0.95.2 2008

[S. ALLAMARAJU, 01] : S. ALLAMARAJU et al. – Programmation J2EE. Conteneurs J2EE, servlets, JSP et EJB. 2001

[A. Goncalves, 07] : A. Goncalves. – Cahier du programmeur Java EE. 2007

[J. Chauvet] : Services Web avec SOAP, WSDL, UDDI, ebXML - Jean-Marie Chauvet, Eyrolles éditions, 2002

[E. Niedermair] : XML Elke Niedermair, Éditions Micro Application, 2001

[M a, Be, Le] : Libero Maesano, Christian Bernard, Xavier Le Galles- Services Web avec J2EE et .NET, Eyrolles éditions, 2003

[Cirano, 03] : Les Web Services et leur impact sur le commerceB2B Gilbert Babin, Michel Leblanc 2003

[UML2] : UML2 Joseph Gabay, David Gabay Dunod, paris, 2008

Webliographie

[Web, 01]: www.labo-sun.com

[Web, 02]: http://www.grappa.univ-lille3.fr/polys/frime/sortie002.html

[Web, 03] : http://www.commentcamarche.net/contents/525-le-protocole-ldap

http://www.memoireonline.com/07/10/3700/m_Conception-et-realisation-dune-application-web-pour-la-gestion-des-stocks-cas-detude-magasin.html puor le mémoire

http://jean-luc.massat.perso.luminy.univmed.fr/ens/jee/tp-JPA1.html

http://docs.oracle.com/javaee/5/tutorial/doc/bnbqa.html

http://jamilatmi.blogspot.com/p/definition-de-serveur-dns_8184.html

http://www.commentcamarche.net/contents/518-dns-systeme-de-noms-de-domaine

http://www.commentcamarche.net/contents/536-les-protocoles-de-messagerie-smtp-pop3-et-imap4

http://www.commentcamarche.net/contents/520-le-protocole-http

http://jlafosse.developpez.com/java/developpement-n-tiers/plate-forme-java-EE/ pour la plate forme java ee

http://www.entrepreneur.com/encyclopedia/e-commerce

http://www.iro.umontreal./dift6275/exposes/art13.ppt

http://www.riffi-amarti.voila.net/e-commerce.ppt

http://david.h.over-blog.com/article-e-commerce--avantages-inconvenients-87365596.htm

http://uml.developpez.com/outils/

http://www.memoireonline.com/06/12/5976/m_Conception-et-realisation-d-un-site-web-dynamique-pour-un-magazine-en-ligne5.html

http://uml.developpez.com/faq/?page=GENE_DIAG

www.marketing-professionnel.fr

mahjoubi.weebly.com/uploads/5/1/1/2/5112616/e_paiement_torkia.ppt

www.commentcamarche.net

Annexe A :
Le commerce électronique

I. Définition :

En cette journée et l'âge, le e-commerce est devenu un mot familier. En termes plus simples, c'est la vente de produits en ligne. Via un propre site web ou un marché en ligne, on peut promouvoir et vendre des produits en ligne, la prise de commandes et l'acceptation du paiement - le tout sans mettre le pied dans une vitrine ou jamais voir les clients en face-à-face. *Ou Selon l'office de la langue française québécoise*, Le e-commerce est l'Ensemble des activités commerciales effectuées par l'entremise des réseaux informatiques, en particulier Internet, dont la promotion et la vente en ligne de produits et services, la vente d'information et l'échange de correspondance électronique.

L'aspect le plus étonnant de l'e-commerce est sa capacité d'influer sur les efforts de vente et de marketing immédiatement. En allant en ligne, tout à coup une boulangerie de quartier ou un service de consultation à domicile étend sa portée à une base nationale, voire internationale de clients potentiels. Les ventes sur le Web ne connaissent pas de frontières internationales.

II. Les Catégories du Commerce Électronique

Les différentes formes du commerce électronique sont basées sur la nature des acteurs mis en relation :

- **Le commerce interentreprises (business to business)**

  L'échange électronique entre entreprises, souvent appelé B to B, désigne une relation commerciale électronique interentreprises ou entre une entreprise et son fournisseur basé sur l'utilisation d'un support électronique.

  - relations stables & de longue durée
  - emploie de systèmes de traitement coûteux.
  - protocoles propriétaires pour chaque réseau

- **Le commerce grand public (business to consumer)**

Le commerce électronique à destination des particuliers, ou B to C qui désigne une relation entre une entreprise et le grand public (particuliers) dont la relation ne se limite pas à l'acte de la vente, mais couvre tous les échanges qu'une entreprise peut avoir avec ses clients de la demande de devis au service après vente. Il s'agit des sites web marchands, type télé-achat.

Exemples : Minitel. Tous les sites commerciaux ou marchants sur Internet font partie de cette catégorie.

n **Le commerce de proximité ou de contact**

Un face-à-face entre l'acheteur et le vendeur, dans les centres d'achats. Cette catégorie peut aussi inclure les échanges de particulier à particulier.

Les cartes prépayées interviennent dans le commerce électronique grand public et surtout dans le commerce de proximité.

n **Autres types de commerce électronique:**

Type	Description
Le *C to C*	*E-commerce entre particuliers*
Le *B to E*	*Echanges électroniques entre l'entreprise et ses employés*
Le *B to G*	*Echanges électroniques entre entreprises privées et gouvernement*
Le *G to C*	*Echange électronique entre le gouvernement et les particuliers*
Le *A to A*	*Echange électronique entre administrations*

III. Technologies du commerce électronique

ü Accès au réseau:

La qualité de l'accès au réseau de télécommunications :

- la largeur de la bande passante

- la fiabilité de la communication en termes de temps d'indisponibilité ou de réparation

- La probabilité de blocage d'un appel pour manque de ressources dans le réseau.

ü Fonctions d'un Site Marchand :

Promotion du site : bandeaux publicitaires électroniques envoyés au client.

Accueil : est le rôle du serveur Web :

L'esthétique de la vitrine virtuelle, la mise à jour des informations.

Concevoir et de réaliser les pages d'accueil intégrants de catalogues de marchandises et d'y rattacher les bases de données multimédias.

Payement : quatre types d'échanges entre le commerçant et l'acheteur :

-La documentation

-Les conditions d'achat et de paiement

-Les instructions de paiement

-La livraison et la réception des articles achetés.

IV. Les avantages et les inconvénients du commerce électronique :

IV.1 Les avantages du commerce électronique

- La procédure d'achat / vente est plus rapide.
- Les produits sont plus faciles à trouver.
- Possibilité d'acheter 24h:24 et 7j/7.
- Plus de limites géographiques pour atteindre les clients.
- Réduction des coûts d'exploitation et une meilleure qualité de services.
- Pas besoin de compagnie physique ou de point de vente.
- Facile à démarrer et accessible à tous.
- Les clients peuvent facilement choisir les produits de différents fournisseurs, sans se déplacer physiquement.

IV.2 Les inconvénients du commerce électronique

- Toute personne, bonne ou mauvaise, peut facilement démarrer une entreprise.
- Et il y a malheureusement beaucoup de sites frauduleux qui volent l'argent des clients.
- Il n'y a aucune garantie de la qualité du produit.
- Les défaillances mécaniques peuvent provoquer des effets imprévisibles sur le processus total.
- Comme il y a un minimum de risques d'interactions clients directs d'entreprise, la fidélité des clients est toujours sur un chèque.

- Il y a de nombreux hackers qui cherchent des occasions, et donc un site de commerce électronique. Le service, les passerelles de paiement, tous sont sujets à l'attaque.

V. Le E-paiement :

L'émergence de l'e-commerce a créé de nouveaux besoins financiers que dans de nombreux cas ne peuvent pas être efficacement remplies par le paiement traditionnels des systèmes. Reconnaissant ce fait, le commerçant et le client se doivent donc absolument de comprendre les implications pratiques des différents moyens de paiement mis de l'avant au niveau des sites marchands afin de choisir celui ou ceux qui correspondent le mieux à leurs milieux et à leurs besoins.

V.1Définition :

Le E-paiement c'est le paiement sur Internet qui désigne les moyens mis en œuvre pour payer sur Internet. Outre la carte de paiement classique, on trouve aussi des moyens plus spécifiques comme les transactions entre particuliers (C2C).

Intégrité, confidentialité, authentification et sécurité sont les objectives de base d'une solution de paiement en ligne.

V.2 Les moyens de paiement en ligne :

Il est possible de payer en ligne via plusieurs outils :

Payer par e-numéro de cartes :

C'est un moyen de paiement rattaché à la carte bancaire qui permet de payer sans donner son numéro de carte bancaire. Des e-numéros sont attribués, des numéros de carte bancaire temporaires.

- e-carte bleue : Disponible à la Société générale, Banque Postale, LCL, Banque Populaire, Caisse d'Epargne, Axa banque
- Virtualis : Disponible au Crédit Mutuel

Virement par Internet :

Toutes les grandes banques proposent des services de virement par Internet. Généralement gratuits entre les comptes du titulaire dans la même banque, ils sont payants sur des comptes d'autres banques.

Payer sans carte bancaire (Transaction via un tiers) :

Certains services permettent de payer en ligne sans donner son numéro de carte bancaire (les plus connus sont PayPal et Google Checkout). Ce sont des sites internet qui gèrent des comptes et qui permettent les transactions monétaires directement.

Cette solution de paiement en ligne suppose d'avoir une adresse de courrier électronique et un numéro de carte bancaire qui est communiqué uniquement à l'intermédiaire de paiement lors de l'inscription.

Ce type de service permet aussi de recevoir de l'argent, de faire des transferts et des virements, etc.

Les frais de transaction peuvent être élevés, mais sont souvent pris en charge par les commerçants, donc indolores pour les clients.

- **Payer par carte bancaire** :

Il suffit d'indiquer le numéro de carte bancaire et sa date d'expiration. Cela est possible grâce à la signature et au certificat électroniques.

- **Autres moyens de paiement (hors ligne) :**

- **Téléphone** : le client appelle et donne les informations nécessaires au règlement.
- **Fax** : les informations requises sont faxées.
- **Courrier** : le client envoi u chèque.

mahjoubi.weebly.com/uploads/5/1/1/2/5112616/e_paiement_torkia.ppt

V.3 Les acteurs du paiement électronique :

- **Le Consommateur :** il achète des biens et/ou services à partir d'un commerçant. L'utilisateur utilise un matériel informatique (exemple : ordinateur, etc.)connecté au réseau pour sélectionner le produit à acheter et passer par la suite à l'achat et au payement.
- **Le Commerçant :** il vend des produits qui peuvent être soit des biens et/ou services à des consommateurs qui les achètent à distance à travers un réseau.
- **La Banque du Commerçant :** le vendeur traite les autorisations de paiements ; pour réaliser ces opérations, la banque du commerçant entretient des liens avec les banques des acheteurs affiliés à travers les réseaux bancaires. Elle saisit les paramètres du paiement et les envois à la banque de l'acheteur pour compensation et alimentation du compte du commerçant par le montant reçu en contre partie du bien et/ou service vendu.
- **La Banque du Consommateur** *:* la banque de l'acheteur fournit la technologie de paiement au consommateur et s'engage à rembourser la dette de son client au profit de la banque du commerçant.

- **L'autorité de certification** : ce dernier garantie la sûreté du moyen de paiement .L'autorité de certification est chargée de délivrer les certificats, et sensée de gérer les clés utilisées pour le chiffrement et la signature des données confidentielles échangées entre les acteurs de paiement. Elle n'est pas directement impliquée dans les transactions de paiement en ligne entre le consommateur et le commerçant. Les relations entre les acteurs du paiement dépendent du moyen de paiement et requièrent une autorité de certification.

V. 4 Protocoles de Payement Sécurisé :

Les protocoles iKP (i = 1, 2, 3) :

Se reposent sur cryptographie à clé publique et compatibles avec l'infrastructure de paiement existante, notamment celle des cartes de crédit

- **1KP** : la passerelle de paiement détient une paire de clés publique et privée et un certificat établi par l'autorité de certification
- **2KP** : 1KP + Le vendeur possède en outre sa propre paire de clés publique et privée et un certificat
- **3KP** : Le protocole 3KP met en jeu trois paires de clés publique et privée et les certificats de clés publique correspondantes.

CyberCash

Dans le protocole CyberCash les opérations de chiffrement utilisent l'algorithme DES avec une clé de 56 bits et RSA avec une clé de 768 bits.

Les logiciels de paiement sont mis à la disposition des clients et des commerçants gratuitement.

Cependant, la commission que CyberCash perçoit sur le commerçant s'élève à 2% du montant de la commande.

Le Protocole SET (Secure Electronic Transaction) : »:

C'est un protocole destiné à sécuriser les paiements par carte bancaire.

Le Protocole SSL (Secure Socket Layer)

VI. Conclusion :

Le commerce électronique n'en est qu'à ses débuts et de nombreuses évolutions doivent encore avoir lieu au niveau des législations, des méthodes de travail et de la sécurité. Les entreprises qui réussiront à sécuriser leurs systèmes et qui sauront s'adapter aux défis du commerce électronique auront demain un avantage comparatif sur leurs concurrents.

Annexe B : UML

I. Définition

UML (*Unified Modeling Language*) est un langage unifié de modélisation et non pas une méthode. Ce langage est né de la fusion de plusieurs méthodes existant auparavant, et est devenu la référence en terme de modélisation objet. UML a été conçu pour permettre la modélisation de tous les phénomènes de l'activité de l'entreprise indépendamment des techniques d'implémentation (système automatisé ou non, langage de programmation...) mises en œuvre par la suite.

II. A quoi sert UML ? [S11]

UML utilise l'approche objet en présentant un langage de description universel. Il permet grâce à un ensemble de diagrammes très explicites, de représenter l'architecture et le fonctionnement des systèmes informatiques complexes en tenant compte des relations entre les concepts utilisés et l'implémentation qui en découle.

UML est avant tout un support de communication performant, qui facilite la représentation et la compréhension de solutions objet :

· Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation de solutions.

· L'aspect formel de sa notation, limite les ambiguïtés et les incompréhensions.

· Son indépendance par rapport aux langages de programmation, aux domaines d'application et aux processus, en fait un langage universel.

UML est donc bien plus qu'un simple outil qui permet de "dessiner" des représentations mentales... Il permet de parler un langage commun, normalisé mais accessible, car visuel.

Il représente un juste milieu entre langage mathématique et naturel, pas trop complexe mais suffisamment rigoureux, car basé sur un métamodèle. Une autre caractéristique importante d'UML, est qu'il cadre l'analyse. UML permet de représenter un système selon différentes vues complémentaires : les diagrammes.

III. Les diagrammes UML [UML 2]

Les diagrammes UML fournissent les informations sur un problème et sa solution. Ils forment les modèles du système (spécifier, visualiser). Les combinaisons de diagrammes représentent les vues du système.

Nous nous intéressons dans ce travail à essentiellement 3 diagrammes.

- ***Diagrammes de cas d'utilisation***

Les diagrammes de cas d'utilisation permettent de représenter le fonctionnement du système vis-à-vis de l'utilisateur, c'est donc une vue du système dans son environnement extérieur.

- ***Diagrammes de séquence***

Les diagrammes de séquence présentent les messages échangés entre des objets selon un point de vue temporel (chronologie d'envoie des messages et ligne de vie des objets).

En effet les diagrammes de séquence on deux dimensions : dimension verticale qui représente le temps (Le temps avance en allant vers le bas de la page si rien n'est précisé), et dimension horizontale représente les différents objets.

- ***Diagrammes de classes***

Les diagrammes de classes permettent de décrire d'une manière générique le comportement du système. Ils présentent les différentes classes du système. Les éléments associés pour la modélisation d'un diagramme de classe sont :

-Les spécifications générales de la classe (type, parent)

-Les spécifications détaillées de la classe (cardinalité, concurrence, persistance)

-Les attributs de la classe

-Les opérations (pré-condition, post-condition, sémantique)

-Les associations

-Les dépendances

- ***Diagrammes d'objets***

Représentent les objets et les liens entre eux. Il permet d'affiner un aspect particulier d'un diagramme de classes pour un contexte donné.

- ***Diagrammes de composants***

Montre les éléments logiciels (exécutables, librairies, fichiers qui constituent le système) et leurs dépendances.

- ***Diagrammes de déploiement***

Indique la répartition physique des matériels du système (processeurs, périphériques) et leurs connexions.

- ***Diagrammes de collaboration***

Représentent les messages échangés entre les objets. Il insiste plus particulièrement sur la notion organisationnelle.

- ***Diagrammes d'états-transitions***

Décrivent le comportement interne d'un objet à l'aide d'un automate à états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets (de type signaux, invocations de méthode).

- ***Diagrammes d'activités***

Décrit le déroulement d'un processus formalisé éventuellement dans un cas d'utilisation, il modélise les actions effectuées sur le système (peut permettre de présenter un processus métier).

IV. Avantages et inconvénients d'UML

Ø Les points forts d'UML

-UML est un langage formel et normalisé :
Il permet le gain de précision, encourage l'utilisation d'outils et constitue à cet effet un gage de stabilité.

-UML est un support de communication performant :
Il cadre l'analyse et facilite la compréhension de représentations abstraites complexes. Son caractère polyvalent et sa souplesse en font un langage universel.

- Formaliser la conception d'application..

-Coordonner les activités entre les différents intervenants.

-Gérer l'évolution d'un projet informatique

-Proposer des outils standardisés prenant en compte de nombreux aspects de la conception.

Ø Les points faibles d'UML

La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation.

Même si l'Espéranto est une utopie, la nécessité de s'accorder sur des modes d'expression communs est vitale en informatique. UML n'est pas à l'origine des concepts objets, mais en constitue une étape majeure, car il unifie les différentes approches et en donne une définition plus formelle.

Le processus (non couvert par UML) est une autre clé de la réussite d'un projet. Or, l'intégration d'UML dans un processus n'est pas triviale et améliorer un processus est une tâche complexe et longue. Les auteurs d'UML sont tout à fait conscients de l'importance du processus, mais l'acceptabilité industrielle de la modélisation objet passe d'abord par la disponibilité d'un langage d'analyse.

V. Conclusion :

En conclusion, UML est un outil précieux, mais, pour bien l'utiliser et en faire un instrument de lisibilité, il nous faut l'accompagner d'un mode d'emploi pour l'élaborer, il nous faut reprendre les questions dans la tradition du génie logiciel et suivre les démarches de conception et d'analyse :

-Analyse de problème en utilisant processus unifié UP -Utilisation d'un langage de modélisation UML

-Etude préalable, construction ainsi tests et mise au point

Résumé

De nos jours, le web n'est plus simplement un énorme entrepôt de texte et d'images, son évolution a fait qu'il est aussi un fournisseur de services.

Aujourd'hui, même si toutes les entreprises n'ont pas fondé l'essentiel de leurs services économiques sur le net, une nouvelle technologie leur facilite grandement les choses et permet une communication facile et à distance entre ces entreprises et leurs partenaires et clients c'est **les services web**. Ils font aujourd'hui une technologie révolutionnaire. Elle fournie un cadre pour trouver, décrire et exécuter ces applications à travers le réseau Internet (ou intranet) indépendamment de tout langage de programmation et de toute plate-forme d'exécution. Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.

C'est dans cette dernière approche que notre projet intervient, c'est le développement d'**un service web** pour la vente de véhicule en ligne, qui permettra au client d'effectuer le paiement en ligne des véhicules.