

Université Sâad Dahleb de Blida



Faculté des Sciences

Département d' : Informatique

Mémoire présenté par :

BOUGUENA Sarah

GUERIBA Nadjeh



En vue d'obtenir le diplôme de Master

Domaine : MI

Filière : Informatique

Spécialité : Ingénierie des logiciels

Sujet : Conception et réalisation d'un composant de messagerie instantanée pour le CMS Joomla

Promoteur : Mr Cherif-Zahar Amine

2012/2013

RESUME

Dans ce travail, notre but est de mettre en œuvre un système de messagerie instantanée via le web sous la forme d'un composant Joomla. Ce composant devra permettre aux utilisateurs du site web de bénéficier des services de messagerie instantanée sans qu'ils n'aient à installer des plugins spécifiques sur leur navigateur.

La messagerie instantanée à laquelle nous nous intéressons ici ne consiste pas simplement en une application de chat (ou de discussion) où les utilisateurs pourront simplement envoyer et recevoir des messages textes. Notre système de MIW devra offrir, en plus des fonctionnalités classiques de chat, des fonctionnalités avancées tel que la communication audio/vidéo et le transfert de fichiers.

Nous proposons dans le cadre de ce travail une solution basée sur des standards très récents introduits dans HTML5, la dernière spécification du langage HTML. Il s'agit du protocole WebSocket et de l'API WebRTC (Web Real-Time Communication).

ABSTRACT

In this project, our goal is to develop a web-based instant messaging system in a form of a Joomla component. This component should allow users of a Joomla-based website to benefit from instant messaging services without the need to install any additional plugin.

The instant messaging services provided by our component should not only allow basic text chat where users can simply send and receive text messages, but should also offer other advanced functionalities such as real-time audio/video communication and file transfer.

In this work, we propose a solution based on very recent standards introduced in HTML5, the latest specification of the HTML language. These technologies include the WebSocket protocol and the WebRTC (Web Real-Time Communication) API.

Remerciement

Louange à ALLAH, le miséricordieux, sans Lui rien de tout cela n'aurait pu être.
Nous remercions ALLAH qui nous a orientés au chemin du savoir.

Nous remercions notre promoteur, Mr Cherif-Zahar Amine, d'avoir accepté de nous encadrer et de nous avoir proposé un sujet passionnant. Nous espérons être dignes de la confiance qu'il a placée en nous.

Nous tenons à remercier Mr Chikhi Nacim Fateh pour ses précieux conseils et son aide inestimable sans lesquels ce travail n'aurait pu aboutir.

Nous remercions également chacun des membres du jury pour nous avoir fait l'insigne honneur d'accepter de juger notre travail.

Un grand merci à nos familles et nos amis qui par leurs prières et leurs encouragements, on a pu surmonter tous les obstacles.

Enfin, Nous tenons à remercier toute personne qui a participé de près ou de loin à l'exécution de ce modeste travail.



Dédicace

Je dédie ce modeste travail aux êtres qui me sont les plus chers au monde. Ils ont toujours été à mes cotés pour m'encourager et me guider vers le chemin du succès ; que Dieu le tout puissant les protège.

A ma très chère sœur Nour el houda et mes deux frères Mohamed et Riad.

A mon amie et binôme Nadjeh et à toute sa famille.

A tous mes amis, qu'ils m'excusent de ne pas pouvoir les citer au risque d'oublier quelqu'un.

Sarah

Dédicaces

A mes très chers parents

Pour tout l'amour dont vous m'avez entouré, pour votre patience et vos innombrables sacrifices.

Je ferai de mon mieux pour rester un sujet de fierté à vos yeux

A mes très chères sœurs Dalila, Nedjma et Hayet, et frères Anouar et Merouan

En souvenirs de toutes les joies et forces qui unissent notre formidable famille.

Vous occupez une place particulière dans mon cœur

A mes compagnons de lutte et à mes amis

Sincèrement que mon Dieu vous comble de grâce et prolonge vos jours de la joie.

A Hayet je garderai toujours un attachement profond pour toi. Je te serai toujours reconnaissante pour

l'assistance que tu m'as apportée dans ma vie, Je t'aime.

A Sarah, mon binôme et ma chère amie, En souvenir de tout ce qu'on a vécu ensemble. Pour ton sincère amitié, je te remercie infiniment d'avoir trouvé une source de paix pour moi ! à sa famille aussi

Je tiens particulièrement à dédier ce travail à mes proches qui m'ont manqué durant ces quelques années...



Gueriba Nadjeh

Sommaire

.....	8
Chapitre 1 Le CMS Joomla.....	20
1.1 Introduction.....	21
1.2 Le web :	21
1.2.1 Définition :.....	21
1.2.2 L'évolution du web.....	22
1.2.3 Les concepts du web.....	23
1.3 Les systèmes de gestion de contenu.....	25
1.3.1 Fonctionnalités d'un CMS.....	25
1.3.2 Avantages attendus de l'utilisation d'un CMS.....	26
1.3.3 Quelques domaines d'application.....	27
1.3.3.1 Les sites éditoriaux.....	27
1.3.3.2 Les communautés en ligne.....	27
1.3.3.3 Le e-Learning.....	28
1.3.3.4 Les bases de connaissance.....	28
1.3.4 Quelques exemples de CMS :	28
1.4 Joomla.....	29
1.4.1 Définition.....	29
1.4.2 Versions de Joomla.....	29
1.4.3 Architecture de Joomla.....	31
1.4.4 Extensions de Joomla :	32
1.4.4.1 Module.....	33

1.4.4.2 <i>Plugin</i>	34
1.4.4.3 <i>Template</i>	34
1.4.4.4 <i>Langues</i>	34
1.4.4.5 <i>Composant</i>	35
1.5 Architecture du CMS Joomla et MVC	35
1.5.1 Historique du design pattern MVC	35
1.5.2 Définition du design pattern MVC	37
1.5.2.1 <i>Le Modèle</i>	37
1.5.2.2 <i>La vue</i>	38
1.5.2.3 <i>Le contrôleur</i>	38
1.5.3 Avantages et inconvénients de MVC :	38
1.5.4 L'architecture MVC dans les applications web/php	39
1.5.5 L'architecture MVC sous le framework Joomla	40
1.6 Conclusion	42
Chapitre 2	43
Messagerie instantanée basée sur le web	43
2.1 Introduction	44
2.2 La messagerie instantanée	44
2.2.1 Définition	44
2.2.2 Historique de la messagerie instantanée	45
2.2.3 Fonctionnalités générales d'un logiciel de MI	47
2.2.4 Les protocoles les plus communs	48
2.2.4.1 <i>Protocoles/applications propriétaires</i>	48
2.2.4.2 <i>Protocoles libres (ouverts, standards, ...)</i>	49
2.2.5 Différence entre la messagerie instantanée et la messagerie électronique	49

2.2.6 Avantages et inconvénients de la messagerie instantanée	50
2.3.1 Définition	51
2.3.2 Fonctionnalités	52
2.3.3 Avantages	52
2.3.4 Comparatif des MIWs existantes	53
2.4. HTML5	55
2.4.1. Historique du langage HTML	55
2.4.2. Les nouveautés dans le code HTML	57
2.4.2.1 <i>Un allègement du code</i>	57
2.4.2.2. <i>Les nouvelles balises sémantiques</i>	58
2.4.2.3 <i>Les balises multimédia</i>	58
2.4.3 Les nouveautés dans le code JavaScript	61
2.4.3.1 <i>Le Local Storage</i>	61
2.4.3.2. <i>La géolocalisation</i>	61
2.4.3.3. <i>Le Drag and Drop</i>	61
2.5 WebSocket	62
2.5.1 Définition	62
2.6. L'API WebRTC	65
2.6.3. <i>Stream API</i>	67
2.6.4 <i>PeerConnection API</i>	69
2.6. Fonctionnement générale de l'application	72
.....	72
Chapitre 3	75



Implémentation et Réalisation	75
Introduction.....	76
3.1. Environnement logicielle :	76
3.2. Choix technique :	76
3.2.1. Choix de javascript :	76
3.2.2. Choix de node.js	78
3.2.2.1. JavaScript Coté serveur :	78
3.2.2.2. Le moteur V8	79
3.2.2.3. Le modèle non bloquant	80
3.2.2.4. Installation de node.js	82
3.2.2.5. Des serveurs web et des threads	83
3.2.2.6. Les modules Node.js et NPM.....	85
1.1. Trouver un module.....	86
1.2. Installer un module	86
3.2.3. Choix de socket.io.....	87
3.2.4. Choix de php5 :	88
3.2.5. Choix de EasyRTC :	88
3.3. Implémentation	88
Conclusion	91
Les modules Node.JS et NPM	105
1. Utiliser NPM pour installer des modules	106
1.1. Trouver un module.....	106
1.2. Installer u nmodule	107
1.3. Mettre à jour les modules	107

4.2.3. Choix de socket.io.....	107
4.2.4. Choix de php 5	108
4.2.5. Choix de EasyRTC	108
4.1. Implémentation	109
4.1.1. Interfaces de l'application	109
Conclusion	111
Conclusion générale	112
Bibliographie	113

Chapitre 1 : Le CMS Joomla

- Figure 1.1 :** Du web 1.0 au web 2.0
- Figure 1.2 :** les versions de Joomla
- Figure 1.3 :** les 3 couches de Joomla
- Figure 1.4 :** Menu Extensions de Joomla
- Figure 1.5 :** Exemples d'extensions de Joomla
- Figure 1.6 :** Séparation entre la partie logique et l'interface utilisateur dans Smalltalk
- Figure 1.7 :** Implémentation de MVC dans Smalltalk
- Figure 1.8 :** L'architecture MVC dans les applications web
- Figure 1.9 :** L'architecture MVC dans Joomla
- Figure 1.10 :** La communication entre les différentes classes de Joomla

Chapitre 2 : Messagerie instantanée basée sur le web

- Figure 2.1 :** Moyens de communication sur internet
- Figure 2.2 :** Exemple de client de MI
- Figure 2.3 :** La messagerie instantanée via le web ICQ2Go
- Figure 2.4 :** Exemple d'implémentation de la balise <video>
- Figure 2.5 :** Exemple d'implémentation de la balise <audio> par différents navigateurs
- Figure 2.6 :** Exemple d'implémentation de la balise <canvas>
- Figure 2.7 :** Exemples d'implémentation de la balise <input>
- Figure 2.8 :** Exemple d'implémentation de la balise input pour la saisie d'une URL
- Figure 2.9 :** Bandeau de partage de la position géographique
- Figure 2.10 :** Schéma du protocole WebSocket
- Figure 2.11 :** Les différentes étapes du fonctionnement de la Stream API
- Figure 2.12 :** Fonctionnement des API MediaStream et PeerConnection
- Figure 2.13 :** Fonctionnement générale de l'application

Chapitre 3: Implémentation et Réalisation

- Figure 3.1 :** Les 3 vies de JavaScript
- Figure 3.2 :** Le schéma classique : PHP sur le serveur, JavaScript chez le client
- Figure 3.3 :** du JavaScript sur le serveur
- Figure 3.4 :** Le modèle non bloquant en programmation

- Figure 3.5 :** Différence entre le modèle bloquant et non bloquant
- Figure 3.6 :** La page de téléchargement de Node.js
- Figure 3.7 :** L'interpréteur Node.js sous Windows (sera peu utilisé)
- Figure 3.8 :** La console Node.js (sera fréquemment utilisée)
- Figure 3.9 :** Le serveur Apache est multithread
- Figure 3.10 :** La souplesse de Node.js Grace aux événement
- Figure 3.11 :** Le site web de NPM

Liste des tableaux :

Chapitre 2 : Messagerie instantanée basée sur le web

Tableau 2.1 : Fonctionnalités des messageries instantanées vie le web existantes

Tableau 2.2 : Prise en charge de WebSocket par les principaux navigateurs

Introduction Générale

Grâce à Internet, de nombreux outils de communication ont vu le jour (messagerie électronique, voix sur IP, ...). La messagerie instantanée (MI) représente un des nouveaux moyens de communication les plus utilisés. Ce type de logiciel est devenu quasi-indispensable pour garder le contact en ligne avec ses amis, sa famille ou ses collègues de travail. Un outil de messagerie instantanée classique est un logiciel que l'utilisateur installe sur son ordinateur afin de pouvoir entrer en contact avec d'autres personnes. Un client de messagerie instantanée propose généralement plusieurs fonctionnalités tel que le chat textuel, la communication audio/vidéo, la gestion des contacts, etc.

Depuis quelques années, un nouveau type de messagerie instantanée est apparu qui connaît de plus en plus de succès auprès des internautes. Ce nouveau type, appelé messagerie instantanée via le web (MIW), épargne à l'utilisateur l'installation d'un client de MI sur son ordinateur en lui permettant d'utiliser ce type de service rien qu'avec un navigateur web, qui est en principe disponible sur n'importe quel ordinateur connecté à Internet.

Dans ce travail, notre but est de mettre en œuvre un système de messagerie instantanée via le web sous la forme d'un composant Joomla. Ce composant devra permettre aux utilisateurs du site web de bénéficier des services de messagerie instantanée sans qu'ils n'aient à installer des plugins spécifiques sur leur navigateur.

La messagerie instantanée à laquelle nous nous intéressons ici ne consiste pas simplement en une application de chat (ou de discussion) où les utilisateurs pourront simplement envoyer et recevoir des messages textes. Notre système de MIW devra offrir, en plus des fonctionnalités classiques de chat, des fonctionnalités avancées tel que la communication audio/vidéo et le transfert de fichiers.

La solution la plus simple pour notre système de MIW serait d'adopter une architecture client-serveur où toutes les données (messages, flux audio/vidéo, fichiers, etc.) transiteraient par le serveur. Cette solution s'avère toutefois inadaptée à nos objectifs en raison de la charge que devra supporter le serveur en terme de quantité de données qu'il devra gérer en temps réel. L'adoption de l'architecture client-serveur risquerait ainsi de dégrader les performances du serveur web du site puisque le service

de MIW nécessitera beaucoup de ressources que ce soit en bande passante ou en mémoire nécessaire pour le traitement en continu de flux importants de données.

Nous proposons dans le cadre de ce travail une solution basée sur des standards très récents introduits dans HTML5, la dernière spécification du langage HTML. Il s'agit du protocole WebSocket et de l'API WebRTC (Web Real-Time Communication).

Le protocole WebSocket permet une communication bidirectionnelle (full duplex) entre un navigateur web (client) et un serveur web. C'est donc un choix parfait pour des applications temps réel "légères" comme le chat mais reste inadapté pour des applications "lourdes" comme la communication audio/vidéo. C'est là justement qu'intervient l'API WebRTC qui a été développée spécialement pour répondre à un tel besoin. Il s'agit d'une API permettant une communication bidirectionnelle (full duplex) directe entre deux navigateurs web (i.e. peer 2 peer). Cette technologie permet ainsi de "décharger" le serveur en liant directement les clients (i.e. navigateurs web) entre eux.

Ce mémoire est organisé en quatre chapitres :

Dans le chapitre 1, nous introduisons le CMS Joomla en présentant son architecture et les différentes extensions qu'il supporte. Dans le chapitre 2, nous abordons la messagerie instantanée via le web et faisons un tour d'horizon sur les nouveautés apportées par le langage HTML5 que nous avons exploitées pour la mise en œuvre de notre composant de messagerie instantanée. Le Chapitre 3 expose la conception de notre composant de messagerie instantanée. Enfin, le chapitre 3 décrit le composant réalisé ainsi que les différentes technologies utilisées pour sa réalisation

Chapitre 1

Le CMS Joomla

1.1 Introduction

Dans ce chapitre nous présentons le cadre général de notre travail. Après un bref rappel sur l'origine et l'évolution du web, nous commençons par donner une vue globale sur les systèmes de gestion de contenu (CMS), puis nous nous intéressons au CMS Joomla en décrivant son principe, les différentes extensions qu'il supporte (composants, modules, etc.) ainsi que son architecture MVC.

1.2 Le web :

Le Web a été inventé plusieurs années après Internet, mais c'est lui qui a contribué à l'explosion de l'utilisation d'Internet par le grand public, notamment grâce à sa facilité d'emploi. Depuis, le Web est fréquemment confondu avec Internet alors qu'il n'est en réalité qu'un de ses services.

1.2.1 Définition :

Le World Wide Web, littéralement la « toile d'araignée mondiale », communément appelé le Web, parfois la Toile ou le WWW, symbolisant le réseau maillé de serveurs d'informations, est un système hypertexte public fonctionnant sur Internet qui permet la consultation d'informations, grâce à des liens créés entre des documents : les pages web.

La page web permet à la fois l'affichage de textes, d'images et de formulaires de saisie mais peut également appeler et afficher différents autres types de documents numériques : son, vidéo, applications... (Cette liste n'étant pas limitative compte tenu du progrès technique en la matière). Sa consultation par le *client* nécessite un logiciel de navigation (navigateur ou browser).

Le concept du World Wide Web a été créé à partir de 1989 au CERN (Organisation Européenne pour la Recherche Nucléaire) par Tim Berners-Lee, puis développé par lui-même et Robert Cailliau en 1990 dans le but de concevoir un système permettant de naviguer simplement d'un espace à un autre d'Internet à l'aide de liens hypertextes et grâce à un navigateur.

En 1993, un navigateur Web graphique, nommé Mosaic, reposant sur les principes de la Toile tels qu'ils ont été formulés par l'équipe du CERN de Tim Berners-Lee, notamment le HTTP, est développé par Eric Bina et Marc Andrsen au NCSA.

NCSA Mosaic jette les bases de l'interface graphique des navigateurs modernes et cause un accroissement exponentiel de la popularité du Web.



1.2.2 L'évolution du web

Le Web est caractérisé par une évolution constante du fond et de la forme des pages Web. Dans sa conception initiale, le web dit web 1.0 comprenait des pages statiques au contenu codé en HTML qui étaient rarement mises à jour, voire jamais. Ces pages sont non-interactives et ne disposent que de peu d'informations.

Une première évolution fut réalisée par des solutions se basant sur un web dynamique appelé web 1.5. Ce web dynamique est généralement basé sur l'association du langage de programmation PHP et des bases de données MySQL. Lorsque l'internaute accède au site dynamisé, il fait exécuter sur le serveur le langage PHP qui va chercher l'information dans la base de données pour la retranscrire dans la page HTML sur le poste utilisateur.

Le web a subi une nouvelle évolution avec l'apparition de nouvelles technologies comme le langage AJAX qui rend les pages interactives et fluides et le flux RSS, qui permet de rester informé des actualités d'une interface web : c'est l'avènement du Web collaboratif, interactif et participatif. Ce Web, dit web 2.0, rend l'internaute acteur. Ainsi, il lui est possible sur certains sites web de modifier, de rajouter ou d'effacer du contenu et d'échanger des informations par des techniques synchrones comme les messageries instantanées, la téléphonie sur internet, ... ou des méthodes asynchrones comme les forums, les wikis, les blogs, ...

Le Web qui est présentement en cours de développement est le Web 3.0. Ce sera l'arrivée du Web sémantique où les informations ne seraient plus stockées mais «comprises» par les ordinateurs afin d'apporter à l'utilisateur ce qu'il cherche vraiment. Le Web sémantique est ce que l'on pourrait appeler l'avènement d'outils permettant de transformer automatiquement les données en informations, et les

informations en savoir. L'enjeu du web sémantique est donc de réussir à étiqueter de manière pertinente le contenu disponible sur la toile, pour permettre un accès intelligent. Le principe repose sur l'intelligence collective des utilisateurs.

A l'avenir Internet ne sera plus qu'une immense base de données dont le mot d'ordre sera: diffuser l'information la plus pertinente de la manière la plus rapide qui soit. Le web 4.0 existera-t-il? Sous quelle forme? Certains affirment qu'il s'agit de l'intelligence artificielle mais qu'en sera-t-il vraiment?

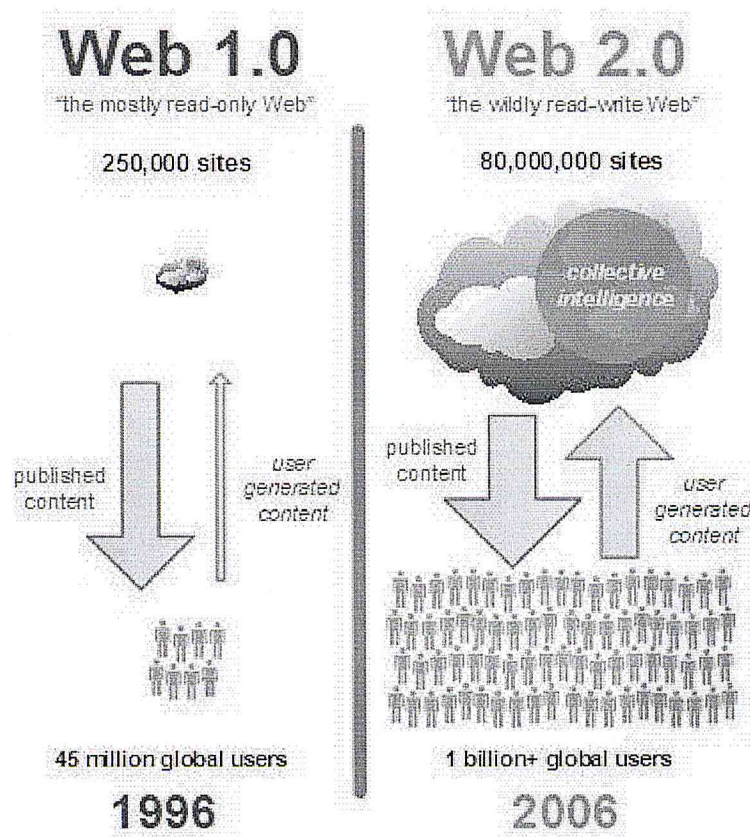


Figure 1.1 _ Du web1.0 au web 2.0 _ [DR.Mike.E, 2008, p.10]

1.2.3 Les concepts du web

Le principe du web repose sur l'exploration d'une myriade de pages web (fichiers HTML) contenant du texte mis en forme, des images, du son des vidéos, ... L'exploration est rendue possible grâce à un logiciel appelé navigateur (Firefox, Internet Explorer, Safari, ...) qui offre une interface d'accès aux différentes informations disponibles sur le web.

Chapitre 1 : le CMS Joomla

Les pages web sont liées les unes aux autres par des liens hypertextes (ou hyperliens). En pratique, un lien hypertexte est un simple mot affiché sur un document permettant de passer à un autre document lorsqu'on clique dessus. Le document lié peut être un fichier à transférer, un son, une animation ou une page web.

Un site web (ou site internet) est ainsi un ensemble logiquement structuré de pages web reliées entre elles et formant un tout (le site) dédié à une organisation donnée.

La création et la mise à jour d'un site peut être confiée soit à une personne, le webmaster, ou à une entreprise spécialisée, l'agence Web. Pour être accessible sur internet, un site web doit être hébergé sur un serveur web.

Le site web est habituellement architecturé autour d'une page centrale, appelée «page d'accueil» et proposant des liens vers les autres pages du site, et parfois des liens dits externes, vers des pages étrangères au site

Pour rechercher une information sans connaître a priori le site susceptible de la fournir, il est nécessaire d'utiliser des outils de recherche. Il existe deux types de système de recherche sur Internet : les moteurs de recherche et les annuaires web.

+ Un *moteur de recherche* est un système automatisé de recherche d'informations sur le web. Un "robot" se charge de parcourir les sites pour visiter les pages web en suivant récursivement tous les hyperliens rencontrés, puis d'analyser le contenu des pages et de l'indexer (par extraction des mots clés associés). L'utilisateur tape alors un mot ou une phrase dans le champ de recherche, et le moteur de recherche retourne une liste de liens vers les pages web pertinentes.

+ Un *annuaire web* est un site Internet dont l'objet est le référencement et la classification de sites web, en différentes catégories (ou rubriques). L'utilisateur peut alors parcourir l'annuaire rapidement grâce à un système de sélection de catégories et de sous-catégories qui permet d'affiner la recherche.

Bien qu'étant de plus en plus performants, les moteurs et annuaires de recherche n'indexent pas la totalité des documents présents sur internet. Il existe en effet, une grande masse d'informations inaccessibles pour les outils de recherche "traditionnels" : c'est le *web invisible*. Le web invisible

comprend des bases et des banques de données ainsi que des bibliothèques en ligne gratuites ou payantes...

1.3 Les systèmes de gestion de contenu

"Un système de gestion de contenu est un logiciel qui permet d'administrer chaque contenu d'un site internet. Un contenu peut être un simple texte, photos, musique, vidéos, documents, etc. L'avantage d'utiliser un CMS (Content Management System) est qu'il ne nécessite pas de compétences ou de connaissances techniques particulières." [Hasin. h, 2008, p.50]

1.3.1 Fonctionnalités d'un CMS

Parmi les fonctionnalités requises dans un système de gestion de contenu nous citons :

→ Gestion du contenu :

- Gestion des documents structurés (fournis par une base de données ou par des fichiers XML) ou non structurés (fichiers HTML, PDF, Word,...) ainsi que les versions, l'archivage, l'historique, les formats, et les mises à jour.
- Utilisation des métadonnées.
- Gestion des pages et des rubriques.

→ Gestion des utilisateurs et des droits :

- Gestion des utilisateurs, des profils et des droits d'accès en référence aux documents ou aux actions liés à eux.

→ Processus de publication :

- Présence de modules supplémentaires qui permettent d'enrichir le CMS et de nouvelles fonctions pouvant être personnalisées selon les besoins de l'utilisateur.
- Une communication basée sur un système de validation des documents (cycle de vie d'un document).
- Différents types de diffusion (selon le CMS)

→ Fonctions supplémentaires :

- Intégration de données externes : base de données, annuaires, fichiers XML, ...
- Fonctions de recherche, des forums, des FAQs, aide contextuelle, sondage, chat, questionnaires statistiques, gestion des versions, etc. ...
- Présence de templates, de gabarit (thème) pour la présentation des sites.

→ Sécurité et confidentialité :

- Maintien et maîtrise des comptes des utilisateurs.
- Administration et configuration des droits.
- Sauvegarde de copies et de l'historique des modifications.

1.3.2 Avantages attendus de l'utilisation d'un CMS

L'énumération des principales fonctionnalités des CMS permet de dégager les avantages de leur utilisation pour une organisation :

- **Éliminer le goulot d'étranglement de la production Web** : nous l'avons vu, la maintenance et la mise à jour du contenu des sites web repose traditionnellement sur une ou deux personnes chargées d'administrer les sites parce qu'elles sont les seules à disposer des compétences requises ; mais elles sont rapidement débordées, ce qui peut entraîner du retard dans la mise à jour.

Les CMS visent à permettre à n'importe quel membre d'une organisation de mettre de l'information en ligne sans difficulté technique. Cette décentralisation évite le passage par des échelons intermédiaires et de nombreuses manipulations de fichiers.

- **Faciliter la production de contenu** : avec les CMS, la publication de contenu ne nécessite qu'un simple navigateur web : le contributeur peut ainsi publier de l'information de n'importe où et à tout moment. De plus, le système de collecte permet d'importer des documents produits avec les outils bureautiques et de les convertir facilement au format nécessaire. Enfin, le contenu stocké dans la base reste accessible et modifiable par les utilisateurs autorisés.

- **Gérer la qualité de l'information** : la mise en place d'une chaîne de validation, via le workflow, réduit le risque d'erreur dans les informations mises en ligne. De plus, il est toujours possible de

commenter un contenu pour y ajouter des informations supplémentaires. Enfin, la normalisation des templates, le suivi du cycle de vie du document et l'automatisation de la gestion des liens, notamment, sont des gages de cette qualité.

- **Organiser la production de contenu** : tout détenteur d'information dans une organisation ou une communauté peut, à l'intérieur de son périmètre de responsabilité et de manière autonome, produire du contenu sans empiéter sur le travail des autres. Cela permet de valoriser le travail de chacun et de gagner en productivité.

- **Permettre la multidiffusion et la mutualisation** : le CMS permet de saisir l'information une fois et de la diffuser sur plusieurs sites. En permettant de répliquer facilement des structures de sites identiques, il autorise la création d'espaces d'information partagés et mutualisés ce qui peut être intéressant dans un Intranet d'entreprise.

1.3.3 Quelques domaines d'application

1.3.3.1 Les sites éditoriaux

Les sites éditoriaux sont un genre très répandu sur le web. Ils offrent la possibilité à un individu ou à un groupe de se positionner comme source d'informations. Ceux-ci peuvent donc devenir des « infomédiaires », voire des veilleurs sur des sujets spécifiques.

Les sites éditoriaux les plus fréquemment rencontrés sont les portails d'informations comme *Zdnet*, les journaux en ligne comme *lemonde.fr*, et les *Weblogs*.

1.3.3.2 Les communautés en ligne

Une communauté en ligne rassemble des internautes qui partagent des centres d'intérêt communs. Elle leur offre la possibilité de publier des articles et d'éclairer la communauté sur des informations en leur possession. Ces informations peuvent provenir d'autres sites web, mais aussi de l'expérience des membres de la communauté. Elle peut aussi offrir des outils de collaboration, comme une messagerie interne ou un agenda partagé. Un forum permet, en plus, de réagir aux contributions ou de compléter l'information.

1.3.3.3 Le e-Learning

Les concepteurs de ressources pédagogiques se sont intéressés eux aussi aux

CMS et ont développé des outils spécialisés : les LCMS (Learning Content Management Systems).

Les LCMS sont au carrefour entre les CMS traditionnels et les LMS – système de gestion de formation

– dont ils intègrent toutes les fonctions :

- individualisation et distribution des parcours de formation,
- suivi de ces parcours,
- gestion des apprenants,
- mise à disposition d'outils coopératifs destinés à faciliter la collaboration entre le tuteur et l'apprenant.

Ainsi les LCMS vont permettre de créer, valider, publier et gérer des contenus de formation. Ils

s'appuient sur le modèle Learning Objet, qui comprend les objectifs de formation, d'évaluations et le

contenu. Des métadonnées y sont associées pour permettre l'individualisation de ces contenus selon des profils.

1.3.3.4 Les bases de connaissance

Dans le cadre d'applications Intranet ou Extranet, les bases de connaissances permettent de capitaliser l'information et le savoir-faire au sein de l'entreprise (idée, documentation, procédure, ...). Cette capitalisation nécessite des technologies capables de gérer des informations aussi bien structurées que non structurées.

Ce type d'utilisation s'inscrit dans un concept plus global : l'ECM (Entreprise Content Management).

1.3.4 Quelques exemples de CMS :

Il existe plusieurs CMS à savoir : Joomla, Spip, Dotclear, Mambo, Drupal, Magnolia, open CMS,

Triade, Typo 3, Plume, Postnuke, Guppy, Xoops, WordPress, etc ; chacun de ces CMS ayant ses

propres avantages et inconvénients. Parmi cette pléthore de CMS, Joomla a retenu notre attention et a été choisi pour la suite de notre travail.

1.4 Joomla

1.4.1 Définition

Joomla est un CMS open source, libre et gratuit pour la gestion de contenu. Il est issu du projet Mambo mené par l'entreprise Miro qui a décidé par la suite de créer une fondation supervisant le projet Mambo dans le but de créer un projet propriétaire du même nom. Cette initiative n'a pas été appréciée par l'ensemble des développeurs travaillant dessus de peur de perdre l'aspect libre du logiciel et dans le but d'assurer : l'internalisation du projet, l'extensibilité et la comptabilité des produits, ce qui a conduit à la séparation et la création du CMS Joomla en août 2005.

Joomla est le CMS le plus populaire et le plus utilisé pour la création et la gestion des sites web parmi les CMS; il présente les caractéristiques suivantes :

- Un logiciel web convivial, populaire et ergonomique qui facilite la gestion du site
- Offre la possibilité d'étendre ses fonctionnalités en proposant la plus grande diversité de composants gratuits et d'autres payants
- Joomla est open source ce qui permet de l'adapter aux besoins et de l'étendre si nécessaire

Joomla permet de concevoir des extensions sous forme de package sans avoir besoin de modifier le code source de l'application. Pour personnaliser le CMS, il suffit de créer un composant, un module ou un plugin selon l'architecture MVC (Model-View-Controller) et de l'installer. Cela permet une grande flexibilité et une capacité de mise à jour optimale (on ne gère que l'extension qu'on a rajoutée), ceci est également valable pour les templates qui gèrent la présentation du site.

1.4.2 Versions de Joomla

La figure 1.2 montre l'évolution des versions du CMS Joomla.

La version la plus ancienne Joomla 1.0 est arrivée à la release stable 1.0.15. Immédiatement après la 1.0.15, Joomla 1.5 a été publiée et aucune version intermédiaire n'a vu le jour, c.à.d. les versions 1.1 jusqu'à 1.4 n'ont jamais existé. Ce saut a marqué la naissance de Joomla en tant que CMS à part entière. En effet, alors que la version 1.0 était basée sur le Framework de l'ancien CMS Mambo et que

Chapitre 1 : le CMS Joomla

tout y était codé en dur, même les libellés des différentes fonctions, l'arrivée de la version 1.5 a constitué une refonte complète du code. Il s'agit d'un nouveau Framework.

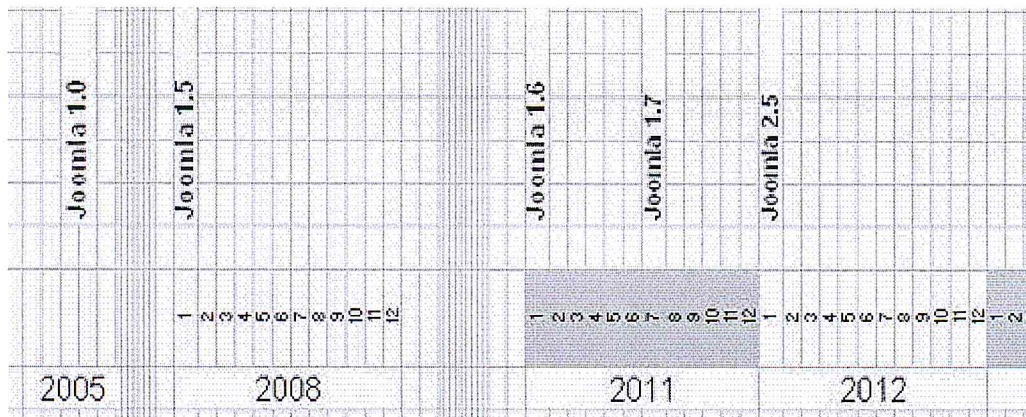


Figure 1.2 _ Versions de Joomla _ [serjio.v, 2009, p.45]

“Voici les évolutions majeures de cette nouvelle version:

- ➔ Internationalisation avec support du codage UTF-8 qui offre la possibilité de gérer des sites multilingues en mode natif.
- ➔ Respect du standard MVC (Modèle-Vue-Contrôleur : méthode de conception de logiciels séparant les couches logiques et présentation) qui rend Le Framework plus facile à gérer et plus flexible facilitant ainsi le développement de nouvelles extensions.” [John.d, 2010, p.99]

Joomla repose, depuis la version 1.6, sur le même Framework de base que Joomla 1.5.

Le CMS Joomla est actuellement à la version 3.0 qui est une version courte durée recommandée pour les développeurs.

La version longue durée la plus récente et recommandée pour les sites web en production est Joomla 2.5 qui est stable depuis le 24 Janvier 2012.

1.4.3 Architecture de Joomla

Joomla repose sur trois couches essentielles :

- ➔ Couche d'extension : qui regroupe les trois extensions modules, composants ainsi que les templates et le gestionnaire de langues.
- ➔ Couche d'application : présente l'ensemble des applications qui étend la classe application de Joomla qui se charge chacune d'un rôle spécifique :
 - JInstallation qui se charge de l'installation de Joomla sur un serveur ;
 - JAdministrator qui se charge de l'arrière-plan (Back-end) d'administration ;
 - JSite qui se charge du site (Front-end) ;
 - XML-RPC qui se charge de l'administration à distance du site Joomla.
- ➔ Couche du Framework : qui se compose de :
 - L'ensemble des classes qui composent le Framework de Joomla ;
 - Les bibliothèques utilisées par le Framework ou ajoutées par les développeurs ;
 - Les plugins qui étendent les fonctionnalités du Framework.

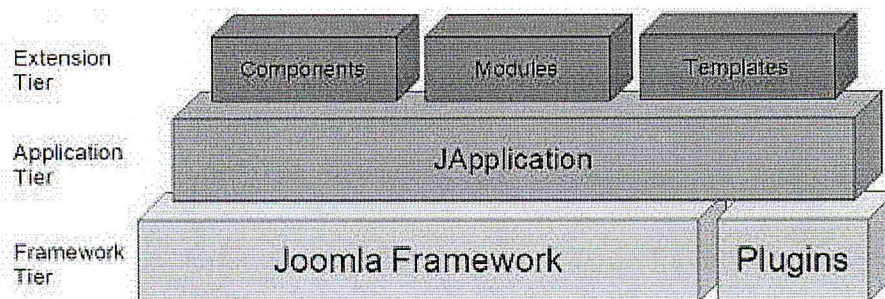


Figure 1.3 _ Les 3 couches de Joomla _ [Taty Sena, 2010, p.20]

1.4.4 Extensions de Joomla :

Il existe cinq différents types d'extensions sous Joomla: les Composants, les Modules, les Plug-ins, les Templates et les Langues . Chacune de ces extensions gère une fonctionnalité bien spécifique.

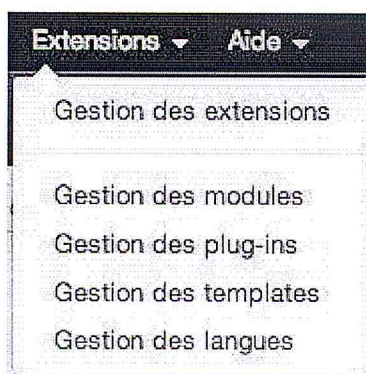


Figure1.4_ Menu Extensions de Joomla _ [Taty Sena, 2010, p.30]

Joomla comporte nativement de nombreuses fonctionnalités ; une installation Joomla contient :

- 27 Composants
- 39 Modules (24 pour le Front-end et 15 pour le Back-end)
- 42 Plug-ins
- 5 templates (3 pour le Front-end et 2 pour le Back-end)
- 2 langues (si la version installée correspond au pack Joomla proposé sur le site de l'Association Francophone des Utilisateurs de Joomla (AFUJ)), l'anglais et le français

Ces extensions vont servir à organiser le contenu, et à apporter diverses fonctionnalités. La figure 1.5 illustre les différents types d'extension disponibles sous Joomla.

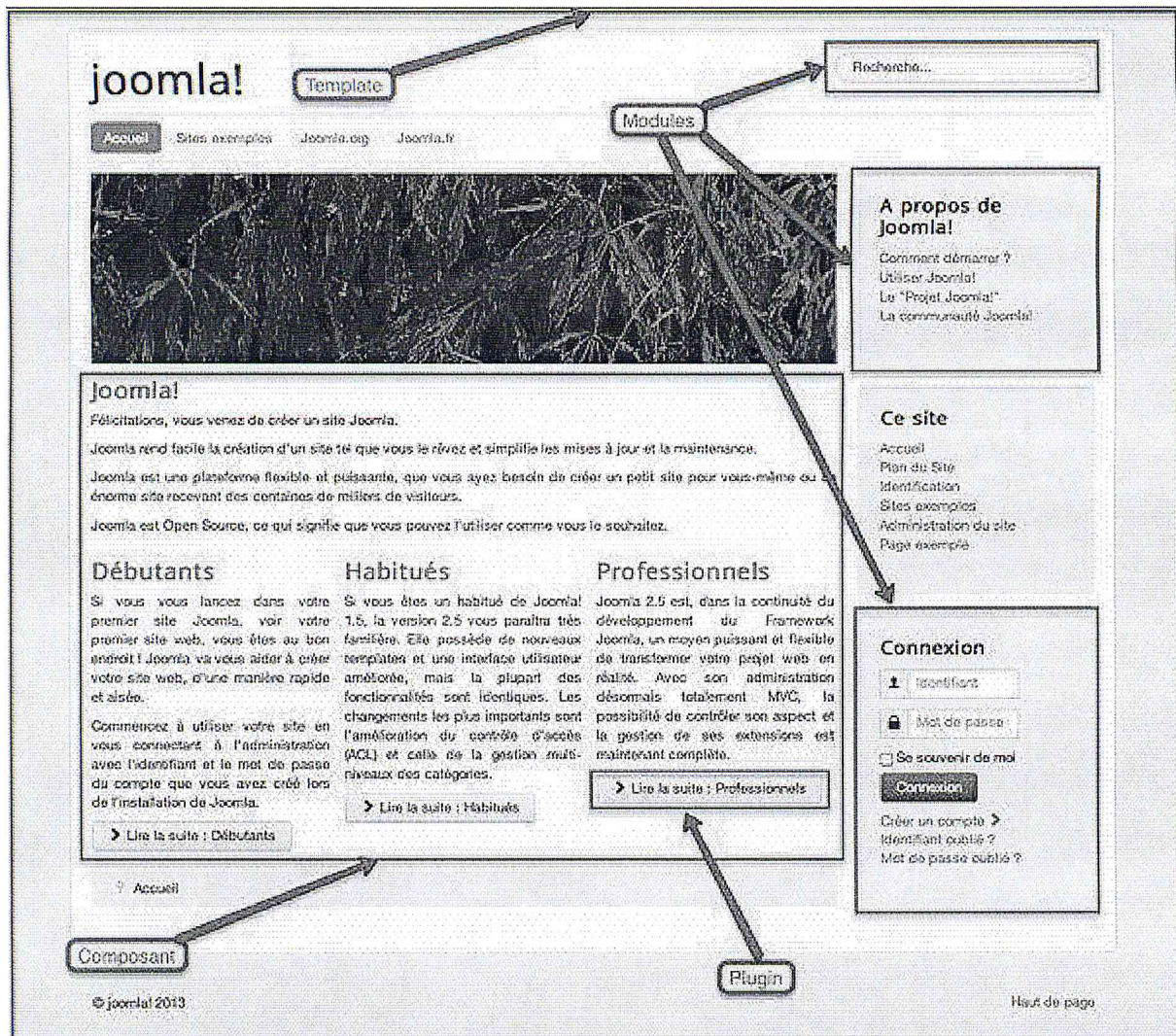


Figure 1.5 _ Exemples d'extensions de Joomla _ [Loïc.F, 2008, p.82]

1.4.4.1 Module

“Les modules sont des extensions légères et flexibles utilisées pour le rendu des pages. Les modules peuvent également être comparés à des “boîtes” disposées autour d’un composant, par exemple : le module de connexion. Le pied de page aussi est un module. Les modules sont assignés par éléments de menu ; il est ainsi possible de décider d’afficher ou de cacher le module de connexion suivant l’élément de menu que l’utilisateur affiche. Parfois, les modules sont liés à des composants comme par exemple le module “Derniers Articles” qui est lié à l’extension com_content et qui affiche les liens vers les nouveaux éléments de contenu. Cependant, les modules n’ont pas besoin d’être liés obligatoirement à

des composants et n'ont, en fait, pas besoin d'être reliés à quoi que ce soit et peuvent simplement être du HTML statique ou du texte.

Exemples : Bannières, Menus, Qui est en ligne, ...

Les modules sont administrables depuis : Menu haut de l'administration → Extensions → Gestion des modules" [Mihàli. M, 2011]

1.4.4.2 Plugin

"Les Plugins sont des extensions plus avancées. Dans l'exécution de n'importe quelle partie de Joomla!, que ce soit dans le noyau, un module ou un composant, un événement peut être déclenché. Quand un événement est déclenché, les plugins qui sont enregistrés avec comme application de gérer cet événement sont déclenchés. Par exemple, un plugin peut être utilisé pour intercepter les articles soumis par un utilisateur et filtrer certains mots." [Mihàli.M, 2011]

1.4.4.3 Template

"Un Template représente est en réalité le design du site Joomla. Avec un Template, il est possible de changer le look ou l'atmosphère d'un site web. Les Templates sont assez simples à construire (avec de bonnes bases en HTML et CSS), et apportent un maximum de flexibilité pour afficher le site web comme souhaité. On note deux types de templates : les templates pour le Front-end du site, et ceux pour le Back-end.

Les Templates sont administrables depuis : Menu haut de l'administration → Extensions → Gestion des Templates" [Mihàli.M , 2011]

1.4.4.4 Langues

"Les Langues sont certainement les extensions les plus basiques. Les fichiers de langues peuvent être packagés de deux manières différentes, soit comme un pack de fichiers de langue pour le noyau de Joomla!, soit comme un pack de fichiers de langue pour une extension. Ces fichiers sont essentiellement des clés/valeurs ou des chaînes de caractères. Ces chaînes de caractères permettent la

traduction des textes statiques qui sont assignés avec le code source de Joomla! Ces packs de langue affecteront aussi bien le front-end que le back-end." [Mihàli.M, 2011]

Les langues sont administrables depuis : Menu haut de l'administration → Extensions → Gestion des Langues

1.4.4.5 Composant

"Les composants sont les extensions les plus importantes et les plus complexes. Ils peuvent être comparés à des mini-applications. La majorité des composants possède deux parties distinctes, une partie site (front-end) et une partie administration (back-end). A chaque fois qu'une page est chargée, un composant est appelé pour afficher le corps principal de cette page. Par exemple, Content (com_content) est le composant qui gère l'affichage du contenu ; les utilisateurs peuvent voir ce contenu sur le front-end du site, et les administrateurs peuvent l'éditer dans l'administration. Les composants sont la partie majeure de votre page car un composant est dirigé par un élément de menu et tous les éléments de menu exécutent un composant.

Exemples : Bannières, Contacts, Fils d'actualité, Liens Web, Recherche Avancée, ...

Les composants sont administrables depuis : Menu haut de l'administration → Composants"

[Mihàli.M,2011]

1.5 Architecture du CMS Joomla et MVC

1.5.1 Historique du design pattern MVC

Le Modèle Vue Contrôleur (MVC) est un pattern d'architecture largement répandu ; " il a été mise au point en 1979 par Trygve Reenskaug, qui travaillait alors sur Smalltalk dans les laboratoires de recherche Xerox PARC ".[Glenn.K, Stephen.D, 1999]

- ❖ Smalltalk est un langage de programmation orienté objet qui est très interactif, l'objectif de l'utilisation de MVC dans Smalltalk était pour la séparation entre l'application logique et

l'interface d'utilisateur, permettant la réutilisation du Model pour implémenter plusieurs interfaces d'utilisateurs.

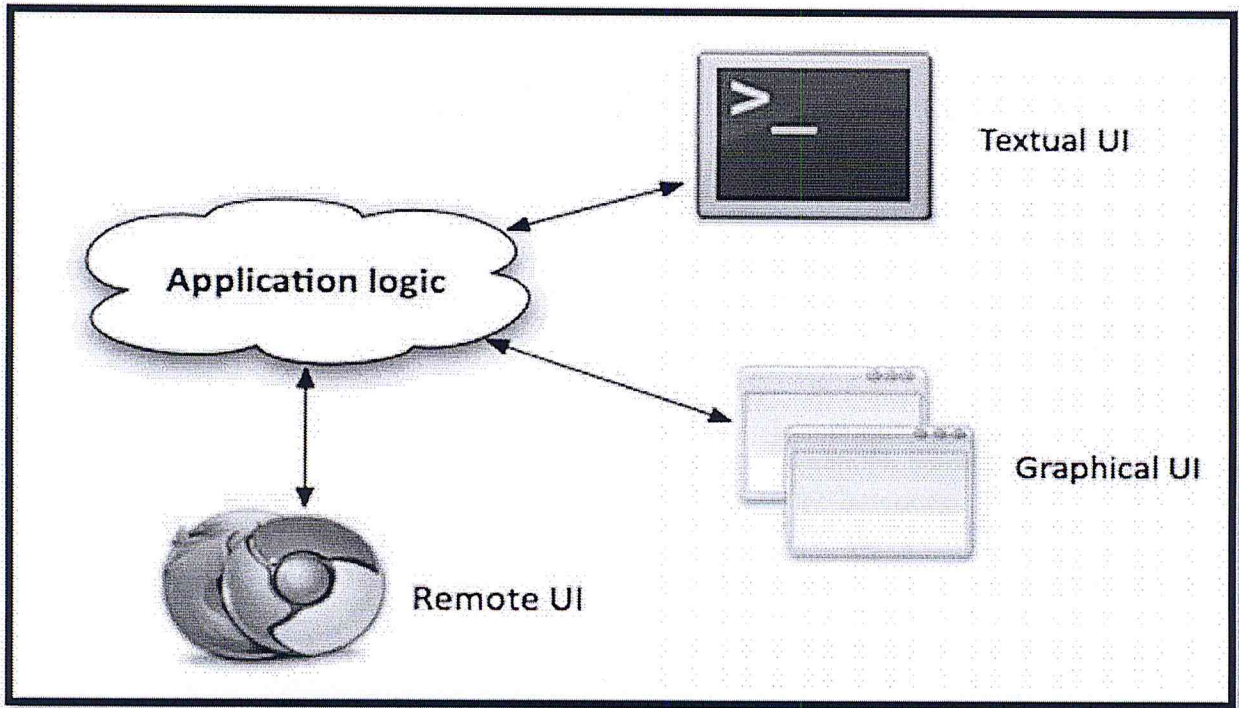


Figure 1.6 _ Séparation entre la partie logique et l'interface utilisateur dans Smalltalk_ [Glenn.K, Stephen D,1999]

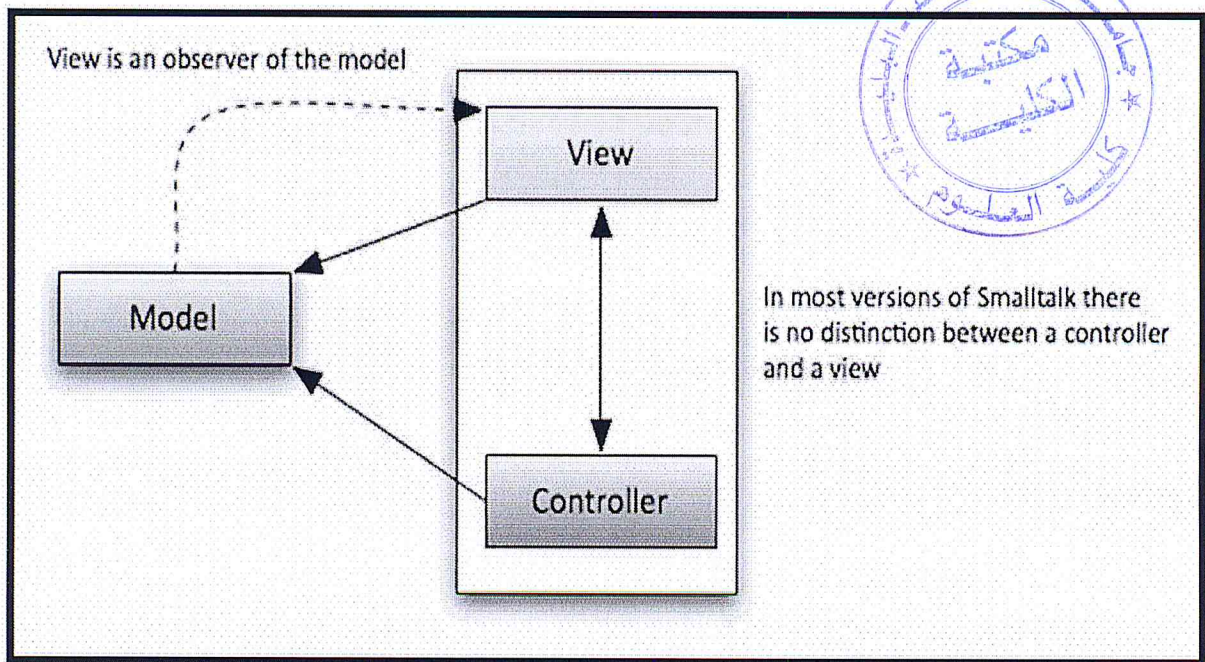


Figure1.7 _ Implémentation de MVC dans Smalltalk _ [Glenn.K, Stephen.D,1999]

Chapitre 1 : le CMS Joomla

dans la figure 1.7 On remarque que l'implantation de MVC dans Smalltalk n'a pas séparé les vues du contrôleur. Il y a en outre un Observateur (patron de conception) (observer pattern) qui est utilisé pour la mise à jour des vues lorsque le modèle est modifié, ce dernier ne dépend ni de la vue ni du contrôleur.

Après plus de 30 ans, MVC existe toujours ; il est utilisé aujourd'hui dans des différents Frameworks comme :

- Ruby On Rails : un Framework web populaire appelé RoR ou Rails, il est écrit en Ruby.
- Apple Cocoa : un Framework d'Apple pour le développement de MAC OS et des applications iOS.
- ASP.Net Framework : s'appuie sur le Framework Microsoft.NET 3.5 pour réaliser des applications ASP.NET selon le modèle MVC avec Visual Studio.
- Apache Struts : un Framework web populaire de java.
- CakePHP : un Framework de développement rapide pour PHP, gratuit et open-source.
- Zend Framework : est un Framework pour PHP5 créé par Zend Technologies.
- Yii Framework : (Y : yes, i :it, i :is) est un Framework destiné au développement d'applications web, il utilise le paradigme de programmation orienté objet.

1.5.2 Définition du design pattern MVC

MVC est un modèle d'architecture qui cherche à organiser une application interactive en séparant :

- Les données
- La représentation des données (l'interface utilisateur)
- Le comportement de l'application (la logique de contrôle)

On va donc distinguer :

1.5.2.1 Le Modèle

Représente le cœur de l'application, il encapsule la logique métier et la manipulation des sources de données. C'est dans ce composant que s'effectuent les traitements liés au cœur du métier. Le modèle comporte des méthodes standards pour mettre à jour les données et pour les récupérer.

1.5.2.2 La vue

Représente l'interface utilisateur. Elle n'effectue aucun traitement, elle se contente simplement d'afficher les données que lui fournit le modèle. Il peut tout à fait y avoir plusieurs vues qui présentent les données d'un même modèle. La vue peut aussi offrir à l'utilisateur la possibilité de changer de vue.

1.5.2.3 Le contrôleur

Qui se charge d'intercepter les requêtes de l'utilisateur, d'appeler le modèle puis de rediriger vers la vue adéquate. Il ne doit faire aucun traitement, il ne fait que de l'interception et de la redirection.

On peut appliquer une architecture MVC à de nombreux cas de mise en œuvre de systèmes informatiques : infographie, applications ; on peut aussi appliquer cette architecture à la mise en œuvre des applications Web et logicielles. Plus récemment il a été recommandé comme modèle pour la plateforme J2EE de Sun et gagne fortement en popularité auprès des développeurs, quel que soit le langage utilisé.

1.5.3 Avantages et inconvénients de MVC :

MVC apporte plusieurs avantages tels que :

- La séparation des compétences (design, BD, application).
- La simplicité de mise à jour.
- La vitesse de création de pages.
- MVC est souvent utilisé dans le développement des applications web grâce à la facilité de synchronisation des vues.
- L'économie de temps de développement.
- La possibilité d'accès aux sites web par tous (overrides).
- La sécurité et la facilité de maintenance.

Malgré ses nombreux avantages, MVC peut dans certains cas engendrer quelques inconvénients tels que :

- Pages plus lentes à afficher (hors cache).
- Plus de ressources consommées.
- Développement initial plus long.
- pour le développement en parallèle exige de multiples programmeurs.
- la connaissance de plusieurs technologies est essentielle.

1.5.4 L'architecture MVC dans les applications web/php

Le MVC, comme l'orientation objet du code, semble être devenu un standard dans le développement d'applications web avec la réputation d'être une bonne pratique de conception.

Une application web respectant ce modèle sera architecturée de la façon suivante :

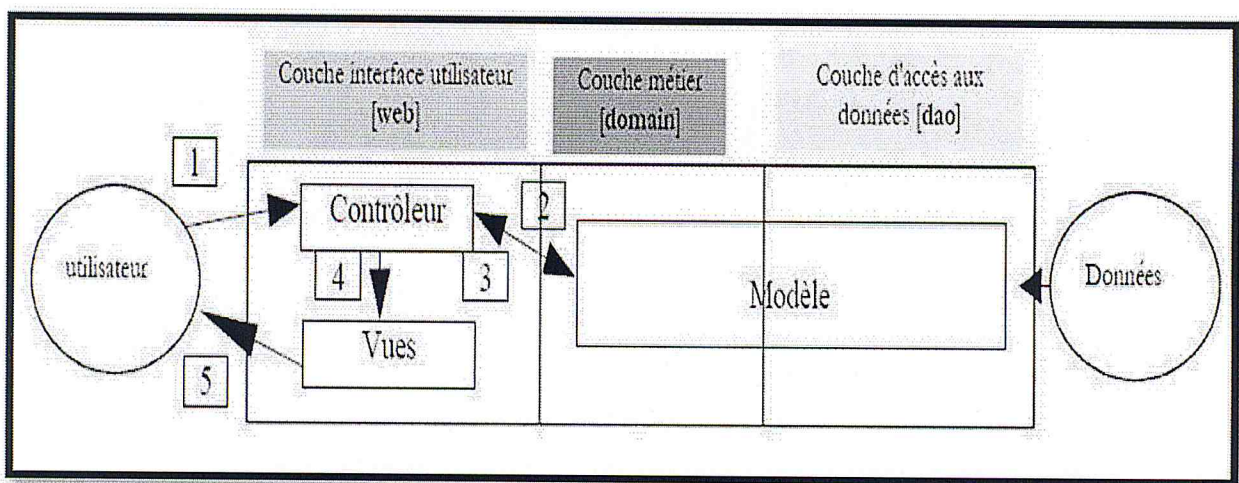


Figure 1.8 _ L'architecture MVC dans les applications web _ [Salihefendic.A, 2011]

- Couche IHM : c'est l'interface utilisateur appelée aussi interface homme machine.
- Couche métier : c'est le cœur de l'application où résident les objets traités par l'application.

Chapitre 1 : le CMS Joomla

- Couche DAO : couche d'accès aux données (Data Access Object). Cette couche permet une indépendance de la logique métier et du stockage des données associées.

La figure ci-dessus montre les différentes étapes de traitement d'une requête utilisateur par une application web développée selon l'architecture MVC : le client fait une demande au contrôleur. Ce dernier voit passer toutes les demandes des clients.

- 1) le client fait une demande au contrôleur. Ce contrôleur voit passer toutes les demandes des clients.
- 2) le contrôleur doit traiter la demande. Pour ce faire, il peut avoir besoin de la couche métier, cette dernière peut éventuellement accéder aux données (via la couche DAO).
- 3) le contrôleur effectue les traitements nécessaires sur les objets renvoyés par la couche métier.
- 4) le contrôleur sélectionne et nourrit la (les) vue(s) pour présenter les résultats du traitement qui vient d'être effectué.
- 5) la vue est enfin envoyée au client par le contrôleur.

1.5.5 L'architecture MVC sous le framework Joomla

Depuis la nouvelle version, Joomla 1.5 propose un véritable Framework de développement basé sur une architecture MVC qui est à la fois une architecture et un modèle de conception, l'utilisation du MVC a pour objectif de séparer complètement la couche métier de la couche présentation.

Joomla 1.6 intègre un moteur de Template dans son modèle MVC. C'est grâce à ce moteur de Template qu'on pourra changer complètement l'aspect du site web, sans modifier son fonctionnement.

Le langage php permet bien évidemment de mettre en place ce type d'architecture mais l'utilisation des classes disponibles de Joomla comme JModel, JView et JController qui implémentent le modèle MVC dans le Framework rend le développement et la mise à jour des composants plus rapides et plus stables.

Joomla implémente des fonctionnalités basiques dans les classes abstraites JModel, JView, JController pour que :

Chapitre 1 : le CMS Joomla

- Le Modèle puisse stocker les données de l'application
- La Vue puisse accéder aux données du Modèle et les afficher, en le faisant passer par un Template.
- Le Contrôleur puisse gérer les requêtes de l'utilisateur, et évoquer le Model à changer son état.

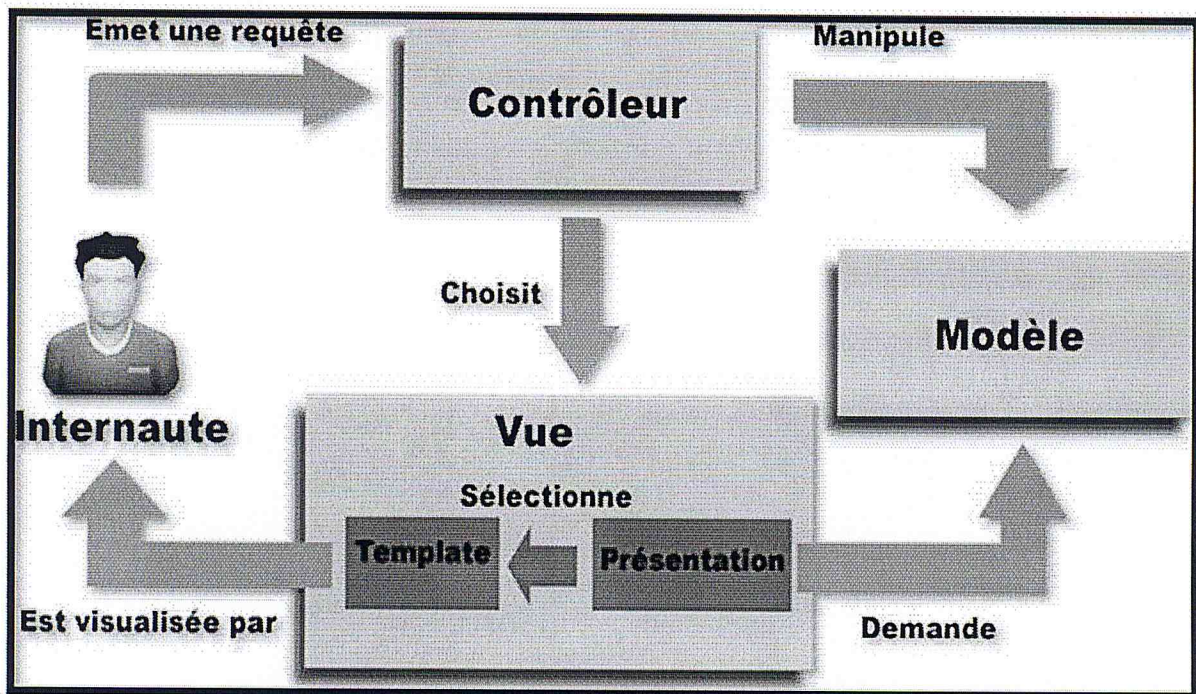


Figure 1.9 _ L'architecture MVC dans Joomla _ [Salihefendic.A, 2011]

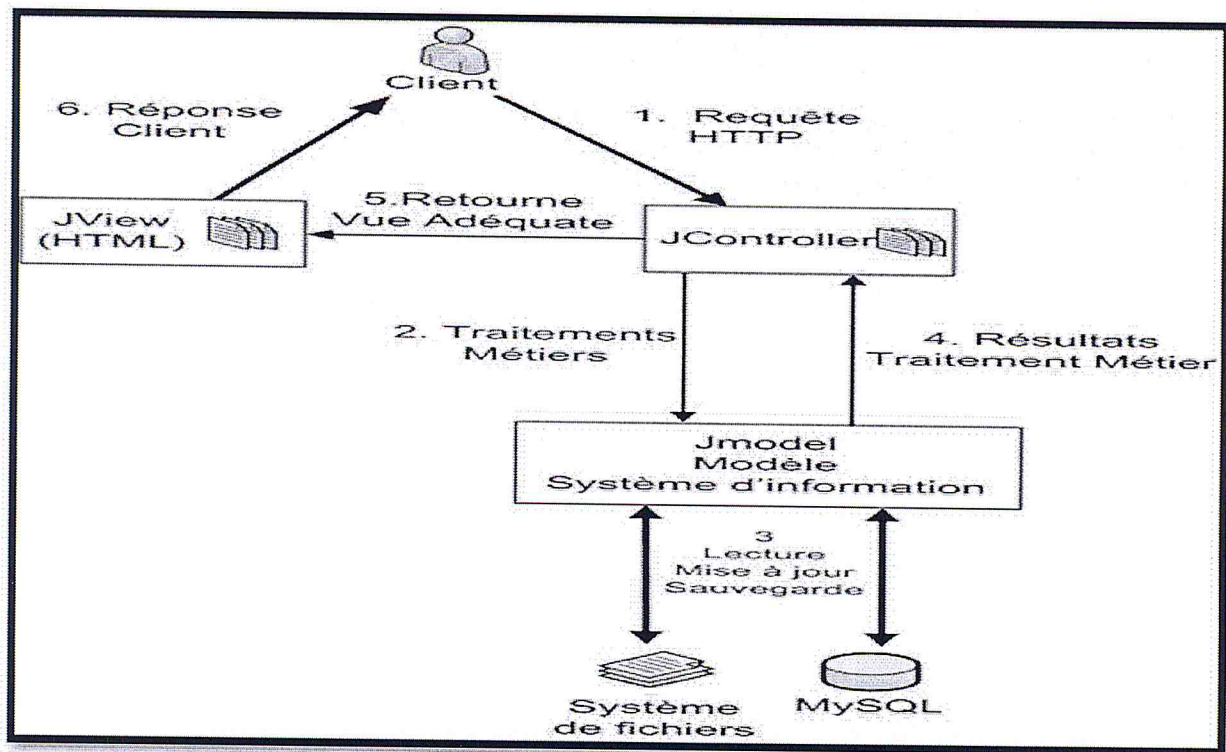


Figure 1.10_ La communication entre les différentes classes de Joomla _ [Adam D, 2010, p.96]

1.6 Conclusion

Nous avons introduit dans ce chapitre la notion de CMS en présentant les fonctionnalités que propose ce type de systèmes. Nous nous sommes ensuite intéressés au CMS Joomla qui a connu un grand succès ces dernières années dans la communauté du développement web. Nous avons abordé l'historique et l'architecture de ce CMS en accordant une attention particulière aux différentes extensions qu'il supporte, à savoir : les templates, les langues, les plugins, les modules et les composants. Enfin, nous avons étudié le modèle MVC que nous utiliserons par la suite pour le développement de notre composant de messagerie instantanée.

Chapitre 2

Messagerie instantanée basée sur le web

2.1 Introduction

Il existe aujourd'hui un grand nombre d'outils de communication à travers Internet . La messagerie instantanée (MI) représente un des nouveaux moyens de communication les plus utilisés. Ce type de logiciel est devenu quasi-indispensable pour garder le contact en ligne avec ses amis, sa famille ou ses collègues de travail. Dans ce chapitre, nous nous intéressons à ce type de service. Dans un premier temps, nous donnons les définitions de quelques notions relatives à la messagerie instantanée classique ainsi qu'à la messagerie instantanée via le web. Dans un second temps, nous décrivons les standards et technologies permettant la réalisation de messageries instantanées via le web.

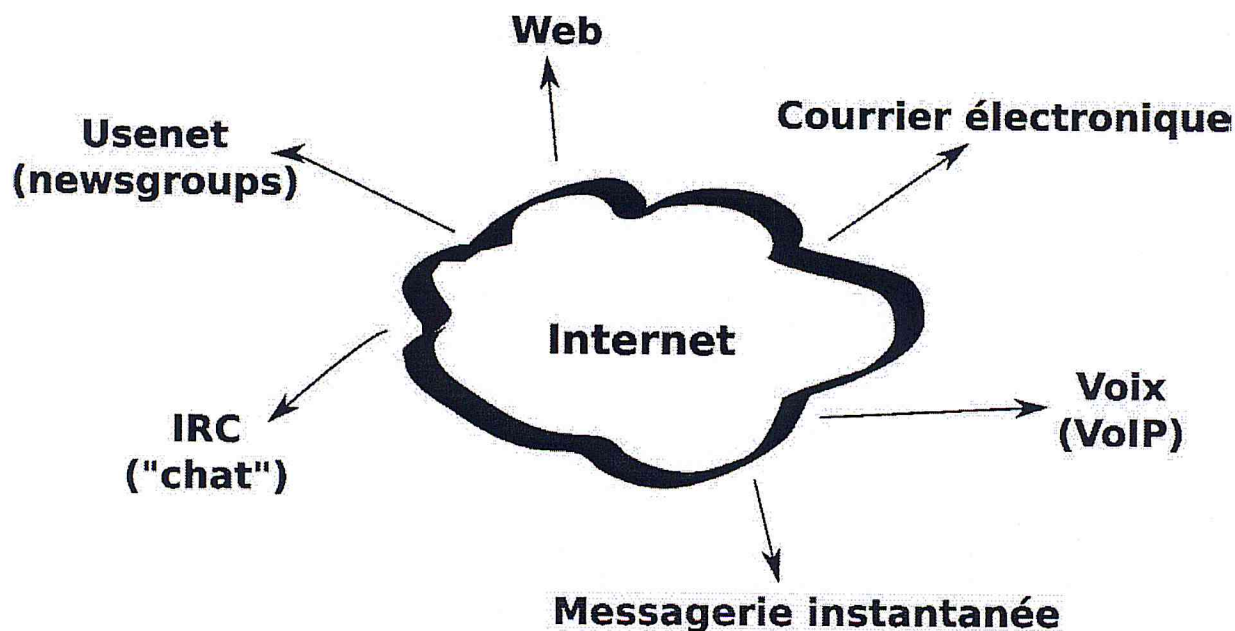


Figure 2.1_ Moyens de communication sur internet _ [Elmagnif. C, 2008,p.40]

2.2 La messagerie instantanée

2.2.1 Définition

La messagerie instantanée, souvent désignée par l'anglicisme « Chat » mais aussi par d'autres expressions comme « clavardage » au Québec, "est une application qui permet l'échange

Chapitre 2 : Messagerie Instantanée Basée sur le Web

instantané de messages textuels entre plusieurs ordinateurs connectés au même réseau informatique et plus communément celui de l'Internet." [Elmagnif. C, 2008,p.40]

En évoluant, la messagerie instantanée a intégré les fonctionnalités de voix et de vidéo grâce à une « Web Cam » (caméra miniature incorporée à l'ordinateur), mais aussi toutes sortes d'applications collaboratives (édition de texte, jeux, etc.), d'envoi des messages automatiques et des notifications : ex : Fixter. Skype.

2.2.2 Historique de la messagerie instantanée

Les hommes cherchent à communiquer depuis longtemps, mais ils n'ont pas toujours été aussi efficaces qu'aujourd'hui. En 1800, envoyer un message de Londres à **Calcutta** prenait deux ans. On écrivait une lettre et la confiait à un bateau à voiles qui naviguait le long des côtes occidentales de l'Europe, de l'Afrique, passait le cap de Bonne Espérance, remontait le long des côtes orientales de l'Afrique, puis à travers l'océan Indien... avec probablement des arrêts dans presque tous les ports.

"En 1914, envoyer le même message prenait un mois. Le canal de Suez était désormais ouvert et les bateaux à vapeur traversaient la mer Méditerranée pour atteindre la mer Rouge : grande innovation.

Dans les années 80, FedEx et autres étaient capables de transmettre le même message en deux jours par avion.

Au début des années 90, quand Internet a été ouvert à l'utilisation commerciale, l'utilisation de l'e-mail s'est généralisée, avec un temps de transport de l'ordre de 10 minutes, dépendant du nombre et de l'engorgement des serveurs intermédiaires (vérifications anti-spam et anti-virus) et de la fréquence avec laquelle l'utilisateur relève sa boîte aux lettres.

Puis la messagerie instantanée est apparue, apportant :

- ➔ La vitesse : chaque message passait de Londres à Calcutta en quelques dixièmes de secondes (et presque toujours en moins d'une seconde) ;

Chapitre 2 : Messagerie Instantanée Basée sur le Web

- Le statut: on peut savoir quand la personne avec qui on veut parler est en ligne, et si elle est en face de son ordinateur ou absente ;
- La discussion en temps réel : on peut discuter en temps réel avec son interlocuteur, quasiment comme dans une conversation téléphonique.

La messagerie instantanée un à un est une idée assez ancienne : sous UNIX, elle existe depuis bien longtemps, sous forme de texte, grâce à la commande talk ; puis sous Windows, il y a eu l'équivalent fenêtré avec WinPopUp.

Tous les systèmes de messagerie instantanée ont leurs racines dans le même protocole : l'ancien, mais encore populaire Internet Relay Chat (IRC), datant de la fin des années 80. Il s'agit d'un protocole standard et ouvert proposant des fonctionnalités simples de discussion à plusieurs. Le protocole ouvert Zephyr, créé au MIT la même année, est un ensemble très simple de services de base, utilisé dans le monde universitaire états-unien.

Ces deux manières de converser sur le réseau ne sont toutefois pas encore ce qu'on appelle la messagerie instantanée, du fait qu'il n'y a pas ou peu d'authentification ni de gestion de présence.

La messagerie instantanée moderne grand public a s'est révélée en 1996 par le logiciel ICQ qui se distinguait par un client unique propriétaire lié à un service unique propriétaire. Il est authentifié, permet de diffuser sa présence et son statut et de gérer une liste de contacts personnels ; ce sont ces fonctionnalités qui introduisent la définition de "messagerie instantanée". Il fut racheté en 1998 par AOL.

Depuis le succès d'ICQ, de nombreux protocoles de communication incompatibles, propriétaires et fermés ont été développés et gratuitement proposés par des fournisseurs de contenu d'Internet tels que : AOL Instant Messenger (AIM) en 1997 (copie conforme d'ICQ, basé sur le même protocole), puis Microsoft MSN Messenger le 22 juillet 1999 (qui deviendra Windows Live Messenger en 2006) et Yahoo! Messenger toujours en juillet de la même année. L'avantage pour eux est de se constituer une large base de clients captifs, puis de pouvoir leur envoyer de la publicité, leur proposer des services étendus payants, etc.

Chapitre 2 : Messagerie Instantanée Basée sur le Web

En 2002, on enregistre autant de courriers électroniques que de messages instantanés échangés dans le monde ; le nombre d'utilisateurs de la messagerie instantanée est estimé à 360 millions.

La fermeture des réseaux, leur cloisonnement, leur incompatibilité et leur non-interopérabilité rend la messagerie instantanée sur Internet dans un état de fragmentation qui n'existe pas dans le domaine du courriel et du web.

En 2004, Jabber/XMPP est normalisé comme standard ouvert par l'IETF, l'organisation de normalisation des protocoles de l'Internet. Jabber, créé en 1998, est à ce jour le seul système normalisé, standard ouvert, non fermé et non propriétaire, qui est très activement développé par des centaines voire milliers de développeurs, administrateurs et des millions d'utilisateurs passionnés, ainsi que par des grands noms de l'industrie informatique : Google, IBM, Sun, France Telecom/Wanadoo/Orange Internet, etc.

En 2005, le travail sur le support des sessions multimédia, dont la voix sur IP, est relancé grâce au protocole Jingle (Jabber) livré par Google Talk.

En 2006, les conventions de nommage pour les identifiants de messagerie instantanée (« IRI/URI scheme ») sont adoptées par l'IETF : elles sont basées sur le protocole Jabber. "

[Elmagnif. C, 2008,p.23]

2.2.3 Fonctionnalités générales d'un logiciel de MI

Un client de messagerie propose généralement plusieurs fonctionnalités, les plus communes sont :

- Inscription (parfois couplée avec enregistrement sur un site web)
- Gestion de contacts (acceptation, demande, bannissement, ...)
- Envoi de messages instantanés
- Envoi de fichiers, de smileys, ...
- Gestion de son statut (en ligne, occupé, reviens dans 1 instant, ...)
- Historique des messages
- Gadgets : webcam, voix, Wizz, radio, ...

Chapitre 2 : Messagerie Instantanée Basée sur le Web

La figure 2.2 montre l'interface graphique d'un client de MI ainsi que les fonctionnalités qu'il propose.

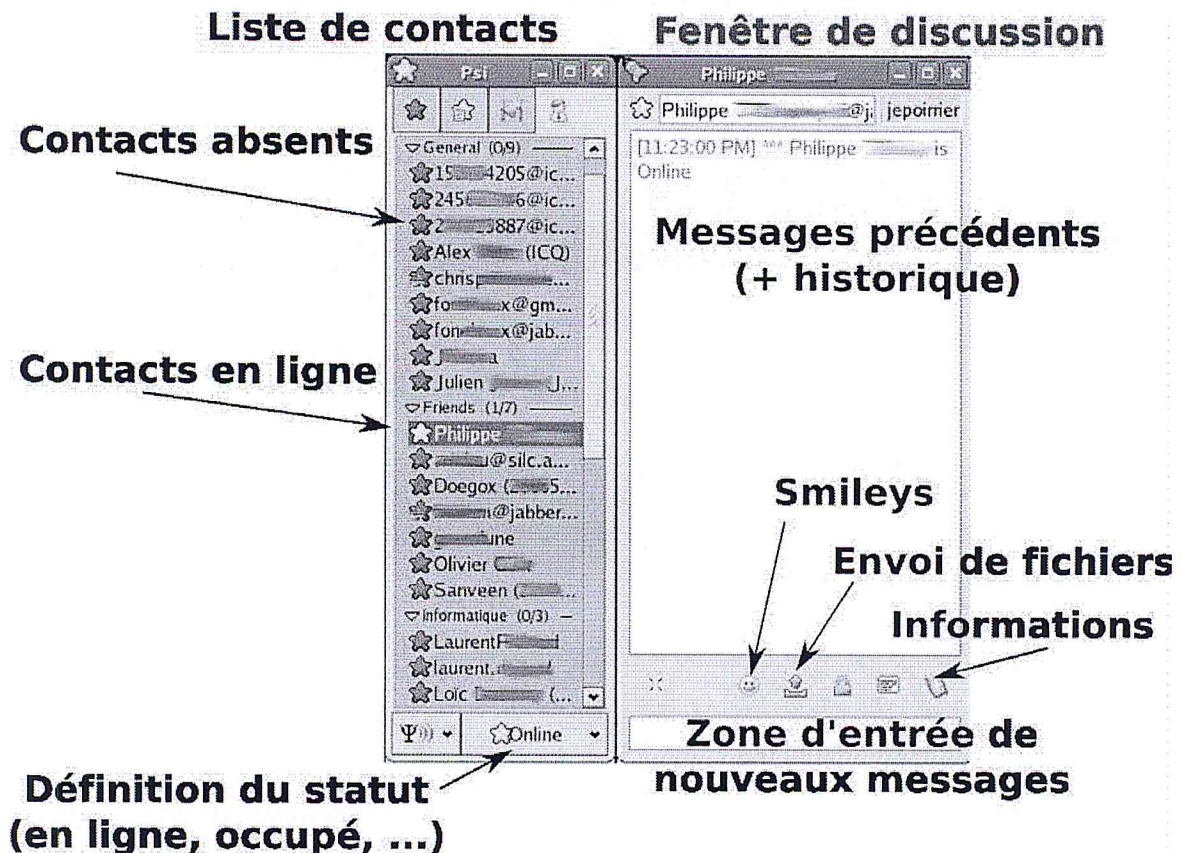


Figure 2.2_ Exemple de client de MI _ [Elmagnif. C, 2008,p.45]

2.2.4 Les protocoles les plus communs

Il existe de nombreux protocoles (ou applications) de messagerie instantanée qui peuvent être classés en protocoles propriétaires et protocoles libres.

2.2.4.1 Protocoles/applications propriétaires

Les applications les plus connues dans cette catégorie sont :

- ICQ
- MSN Messenger
- Yahoo Messenger
- AOL Instant Messaging
- QQ, Skype, Gadu-Gadu, ...

2.2.4.2 Protocoles libres (ouverts, standards, ...)

De nombreux protocoles libres et open source ont été proposés. Les plus populaires sont :

- Jabber /XMPP (Extensible Messaging and Presence Protocol) : peut être utilisé via des salons de discussion qui se trouvent sur un serveur XMPP.
- IRC : est un protocole de communication en mode texte.
- SILC (Secure Internet Live Conferencing) : est un protocole de communication en mode texte, il ressemble à IRC mais son fonctionnement est très différent. La principale caractéristique de ce protocole est qu'il utilise beaucoup le chiffrement pour sécuriser la communication.

2.2.5 Différence entre la messagerie instantanée et la messagerie électronique

La messagerie électronique et la messagerie instantanée sont deux moyens d'envoi de messages sur l'Internet qui diffèrent sur plusieurs points :

- ▶ La messagerie instantanée permet de voir quand ses contacts sont en ligne.
- ▶ La messagerie instantanée permet d'envoyer des messages **en temps réel**. Les messageries classiques rassemblent les messages pour qu'ils puissent être lus ultérieurement.
- ▶ Les messages instantanés sont temporaires. Ils disparaissent une fois que la fenêtre de messages est fermée, à moins que l'utilisateur ne les enregistre expressément. Les messages électroniques ne disparaissent que si l'utilisateur les efface.
- ▶ Dans la fenêtre de messages instantanés, il est possible de parler à plus d'une personne à la fois, comme lors d'une session de conversation en ligne.

La messagerie instantanée constitue un complément idéal au courrier électronique. L'utilisateur peut ainsi voir quand ses contacts sont en ligne et leur envoyer des messages instantanés. S'ils ne sont pas connectés ou que l'utilisateur veuille leur envoyer un message qu'ils sont susceptibles de vouloir garder, il utilise alors la messagerie électronique.

2.2.6 Avantages et inconvénients de la messagerie instantanée

La messagerie instantanée possède de nombreux avantages par rapport à d'autres moyens de communication. Parmi ces avantages :

- Des économies de téléphone et de temps
- Des échanges de messages et de documents en temps réel
- Moins de spams (sollicitations actives des internautes par mail sans leur accord avec des messages publicitaires de toutes sortes afin de les inciter à souscrire à des offres de gains en ligne, de jeux, etc.)
- Une connaissance des personnes en ligne et ainsi pouvoir poser une question qui nécessite une réponse rapide
- Pas d'encombrement du disque dur : les messages s'effacent une fois la fenêtre du message fermée.
- Réaliser un travail collaboratif
- Échanger des fichiers
- De travailler en direct
- De faire en même temps de la téléphonie

Malgré ses nombreux avantages, la MI possède toutefois quelques inconvénients tels que :

- La sécurité : une messagerie instantanée est très facile à pirater et les pièces jointes sont susceptibles de contenir des virus
- Le contrôle : aucune possibilité de savoir à quelles fins est utilisé l'outil
- L'incompatibilité des différents logiciels, de leurs versions et des systèmes d'exploitation
- La compréhension des messages : limite du nombre de mots, utilisation d'un langage abrégé
- L'absence de formation : mauvaise utilisation ou utilisation partielle et donc perte du temps

2.3. Messagerie Instantanée via le Web (MIW)

2.3.1 Définition

La messagerie instantanée via le web (en anglais "web-based instant messaging") est un nouveau type de messagerie instantanée apparu au début des années 2000. Elle permet de se connecter à un ou plusieurs protocoles ou systèmes de messagerie sans nécessiter l'installation d'un logiciel dédié, mais en utilisant une *interface web* par le biais d'un *navigateur*.

Les sites web proposant des services de MIW (appelés aussi des *webmessengers*) peuvent être officiels (à l'instar de MSN Web Messenger¹, Yahoo! Messenger for the Web², AIM Express³ ou Google Talk Gadget⁴), ou exploités par des tierces parties (telles que meebo ou eBuddy).

De plus, certaines MIW sont directement intégrées dans l'interface d'un webmail, comme Yahoo! Messenger dans Yahoo! Mail, ou Google Talk dans Gmail.

Historiquement, ICQ2Go est la première messagerie instantanée via le web ; elle permet de se connecter à ICQ au moyen d'un applet java. La figure montre l'interface la dernière version d'ICQ2Go appelée Web-ICQ.



Figure 2.3_ La messagerie instantanée via le web ICQ2Go _ [ICQ, 2005]

Chapitre 2 : Messagerie Instantanée Basée sur le Web



Conferences chat are available



Chat history tracking



Registration in the chat site is required (not in the network)

	Networks						Mobile	Chat Conference	History Recording	Registration Required
eBuddy	✓	✓		✓			✓	✓		
iLoveIM	✓	✓	✓	✓				✓		
Yahoo Web Messenger		✓					✓	✓	✓	✓
Msn Web Messenger	✓						✓	✓	✓	✓
Google Talk			✓				✓	✓	✓	✓
Meebo	✓	✓	✓	✓	✓	✓		✓		
Kool IM	✓	✓	✓	✓	✓					
Radius IM	✓	✓	✓	✓				✓		
Snimmer	✓	✓	✓	✓	✓	✓		✓	✓	
Mabber	✓	✓	✓	✓	✓	✓	✓		✓	✓
Easy Messenger	✓	✓		✓	✓	✓				
IMunitive	✓	✓								✓
Messenger FX	✓									
AIM Express				✓			✓		✓	✓

Tableau 2.1_ Fonctionnalités des messageries instantanées via le web existantes _ [Hazem.T, 2007]

2.4. HTML5

L'HTML5 est une *évolution* de l'HTML 4.01 (et de l'XHTML 1.0). Cette évolution consiste en une multitude de nouvelles fonctionnalités qui ont été apportées au langage HTML ainsi qu'au JavaScript.

2.4.1. Historique du langage HTML

Les premières versions de HTML voient le jour dans les années 1990, dérivées de la grande famille des langages SGML (Standard Generalized Markup Language). Il ne s'agit cependant pas de normes (il n'y a aucune spécification HTML 1.0), car le langage reste en pleine évolution, principalement motivée par les développements des navigateurs. Les pages utilisent le protocole HTTP (HyperText Transfer Protocol) pour transiter sur le réseau et établir le dialogue entre le navigateur et le serveur. L'IETF (*Internet Engineering Task Force*) héberge les premiers groupes de travail HTML.

En 1995, le W3C (World Wide Web Consortium) publie HTML 2.0 (sans les ajouts spécifiques à Netscape Navigator 2) et débute le brouillon HTML 3.0 qui ne débouchera pas directement sur des implémentations concrètes. **JavaScript**, langage de programmation créé pour ajouter de l'interactivité aux pages web, est inventé pour le compte de Netscape par Brendan Eich.

En 1997, le W3C publie **HTML 3.2**, officialisant certaines des inventions propriétaires des navigateurs. Internet Explorer 4.0 installé par défaut avec Windows 98 supplante Netscape. Il fait appel malheureusement à des nouveautés dont Microsoft est propriétaire (JScript, VBScript, ActiveX). Le travail continue immédiatement sur HTML 4.0 avec la standardisation de nombreuses fonctionnalités avancées (support des styles, des scripts, des *frames* et des objets) et des améliorations relatives à l'accessibilité.

Après la publication de **HTML 4.0** en décembre 1997, le premier groupe de travail (*HTML Working Group*) cesse son activité. Le second, qui sera en réalité le groupe de travail XHTML (*Extensible HyperText Markup Language*), est chargé de redéfinir HTML comme une application de XML, avec un rôle limité de maintenance pour HTML 4.0 puis 4.01. Ce groupe est à l'origine de **XHTML 1.0**, mais ne produit aucune avancée concrète pour le langage HTML.

Chapitre 2 : Messagerie Instantanée Basée sur le Web

La spécification XHTML 1.0 reprend les balises HTML 4.01 à l'identique, il s'agit juste d'adopter une syntaxe plus stricte suivant les règles du XML que l'on promettait à un bel avenir. Cette syntaxe définissant un cadre avec des règles plus structurées et combinées à CSS, conquiert de nombreux auteurs web qui y voient un ensemble de bonnes pratiques à favoriser.

Dans l'élan, le W3C publie **XHTML 1.1** qui se révèle complexe à mettre en place, compte tenu des contraintes imposées par la spécification et des moteurs des navigateurs utilisés par les internautes. En effet, un document créé en XHTML et servi avec le type MIME adéquat, ne pouvait même pas être compris par le navigateur le plus répandu à ce moment-là, Internet Explorer.

Les soucis commencent lorsque le W3C décide d'évoluer à terme vers **XHTML 2.0** dont le premier brouillon est dévoilé le 5 août 2002. Cette nouvelle version, qui se veut un nouveau départ, allégée de son lourd héritage, est incompatible avec les précédentes.

Certaines balises très courantes ne sont plus valides, telles que *img* pour les images.

À l'initiative du groupe, Ian Hickson d'Opera travaille sur Web Forms 2.0 pour étendre les possibilités des formulaires HTML – spécification qui sera intégrée à HTML 5 par la suite. Le W3C tente de persévérer sur la voie du XML.

Le **WhatWG** (*Web HyperText Application Technology Working Group*) est constitué par des passionnés souhaitant améliorer HTML, issus d'équipes de la fondation Mozilla, d'Apple et d'Opera. Leur but est de faire évoluer "rapidement" le langage pour répondre aux besoins concrets de l'explosion du Web, tout en maintenant sa simplicité et sa rétrocompatibilité.

Le WhatWG débute son travail en juin 2004 sous la dénomination de Web

Applications 1.0. Ian Hickson, éditeur officiel du document HTML 5, rejoint finalement Google (membre du W3C).

La spécification **HTML 5**, soutenue en avril 2007 devant le *World Wide Web Consortium* par la fondation Mozilla, Apple et Opera, est acceptée comme point de départ du nouveau groupe de travail HTML. Ce dernier publie le premier brouillon (*Working Draft*) le 22 janvier 2008 et admet que XHTML 2.0 est trop ambitieux

2.4.2. Les nouveautés dans le code HTML

2.4.2.1 Un allègement du code

Tout d'abord, certaines balises ont été simplifiées afin d'alléger le code.

C'est le cas par exemple du Doctype (la première ligne indigeste en haut de chaque page HTML), de la balise <html>, de la balise <meta>, de l'encodage des caractères, et des balises de feuilles de style et de script.

Alors qu'auparavant on pouvait avoir ce genre de code :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
  <link rel="stylesheet" type="text/css" href="design.css" />
  <script type="text/javascript" src="script.js"></script>
</head>
<body>
</body>
</html>
```

En HTML5, ça nous donne :

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8" />
  <link rel="stylesheet" href="design.css" />
  <script src="script.js"></script>
</head>
```

De manière générale, on peut dire que l'HTML5 est beaucoup plus léger et laxiste sur l'écriture du code HTML. Un document HTML5 sera valide W3C même si on écrit les balises en majuscules, ou si on n'écrit pas le caractère '/' de fermeture d'une balise auto-fermante telle que . La philosophie est donc à présent de laisser chacun adopter le style de code qu'il préfère. Ce choix est assez contestable mais il ne tient qu'au développeur de conserver les bonnes pratiques adoptées grâce à l'XHTML1.0.

2.4.2.2. Les nouvelles balises sémantiques

L'HTML5 introduit également un ensemble de nouvelles balises afin de donner plus de sémantique (de sens) à nos pages. Par exemple, au lieu d'utiliser une `<div>` avec un `id="header"`, nous pouvons utiliser tout simplement la balise `<header>`. Parmi ces balises sémantiques on trouve entre autres :

- `<header>` : qui indique que l'élément est un en-tête
- `<footer>` : qui indique que l'élément est un pied-de-page
- `<nav>` : qui indique un élément de navigation tel qu'un menu
- `<aside>` : qui correspond à une zone secondaire non liée au contenu principal de la page
- `<article>` : qui représente une portion de la page qui garde un sens même séparée de l'ensemble de la page (comme un article de blog par exemple)

Ces noms n'ont pas été choisis au hasard. Google a fourni au W3C les chiffres de la fréquence d'utilisation de tous les *id* utilisés sur les sites indexés par le moteur de recherche, et ceux-ci font partie de ceux qui sont les plus employés.

2.4.2.3 Les balises multimédia

- La balise `<video>` : cette balise intègre directement un lecteur vidéo dans la page, avec des boutons Lecture, Pause, une barre de progression, du volume... Un vrai petit Youtube intégré à la page et natif au navigateur. La figure 2.4 montre un exemple d'intégration de cette balise.



Figure 2.4 _ Exemple d'implémentation de la balise `<video>` _ [Rodolphe.R, Raphaël G,2012, p.45]

Chapitre 2 : Messagerie Instantanée Basée sur le Web

- La balise `<audio>` : cette balise est l'équivalent de la balise vidéo mais pour l'audio. Grâce à cette balise, on dispose en trois lignes de code d'un lecteur MP3 !

Nous pouvons d'ailleurs voir sur la figure ci-dessous que chaque navigateur utilise un design qui lui est propre pour styliser son lecteur. Dans tous les cas il est possible de créer son propre design si l'on préfère un rendu uniforme quel que soit le navigateur utilisé.



Figure 2.5 _ Exemple d'implémentation de la balise `<audio>` par différents navigateurs _ [Rodolphe.R, Raphaël G,2012, p.46]

- La balise `<canvas>` : cette balise est probablement la plus prometteuse de toutes, puisqu'il s'agit d'une surface sur laquelle il est possible de tracer des formes et de les animer. En résumé, c'est dans cette zone que sont réalisées des animations ou des jeux. La figure ci-dessous illustre l'utilisation de cette balise.

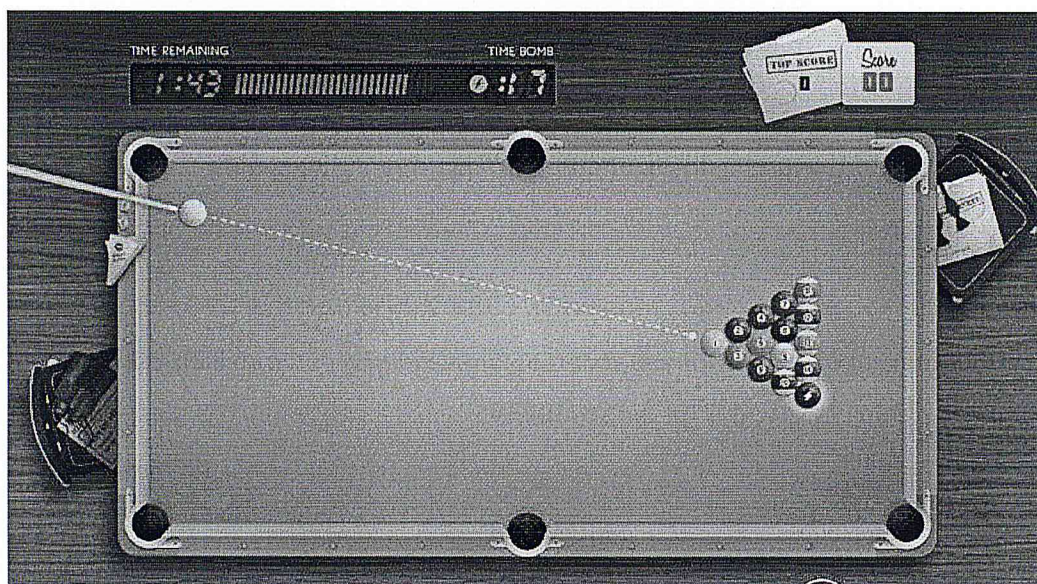


Figure 2.6 _ Exemple d'implémentation de la balise `<canvas>` _ [Rodolphe.R, Raphaël G,2012, p.47]

Chapitre 2 : Messagerie Instantanée Basée sur le Web

2.4.2.4 Des formulaires améliorés

Enfin parmi les nouveautés côté HTML on peut également citer une évolution des formulaires. Il est possible de spécifier de toutes nouvelles valeurs dans le champ "type" des balises `<input>`, afin d'indiquer le type de contenu du champ. C'est particulièrement utile afin d'effectuer une première validation du contenu avant l'envoi des informations au serveur. C'est également très important pour la navigation depuis un Smartphone, qui affichera un clavier adapté selon le type de contenu.

Par exemple, pour le champ "input" suivant :

```
<input type="tel" />
```

On aura un clavier similaire à celui de la figure ci-dessous.

Sur un iPhone



ou sur un Android

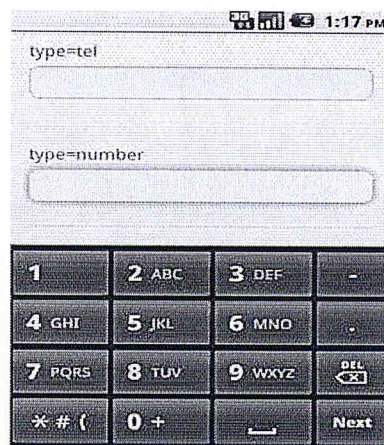


Figure 2.7 _ Exemples d'implémentation de la balise `<input>` _ [Rodolphe.R, Raphaël G,2012, p.48]

Pour le champ `<input type="url" />`, on aura un clavier similaire à celui de la figure 2.8.



Figure 2.8 _ Exemple d'implémentation de la balise `input` pour la saisie d'une URL _ [Rodolphe.R, Raphaël G,2012, p.49]

2.4.3 Les nouveautés dans le code JavaScript

Il est loin le temps où le JavaScript était un point noir du développement web... L'apogée de l'AJAX et du web 2.0 ainsi que des bibliothèques JavaScript telles que jQuery a rendu son utilisation bien plus agréable et efficace. De plus, de moins en moins d'utilisateurs désactivent le JavaScript et il est même souvent impossible de le désactiver sur certains navigateurs. C'est dans ce contexte que le JavaScript a été propulsé vers l'avant avec l'HTML5. L'API JavaScript (les objets et les méthodes utilisables) a été généreusement enrichie avec de nouvelles fonctionnalités ; nous présentons ci-dessous quelques-unes.

2.4.3.1 Le Local Storage

Le Local Storage est une manière élégante de stocker dans le navigateur des informations facilement.

2.4.3.2. La géolocalisation

Il ne s'agit pas strictement d'une spécification de l'HTML5, mais elle y est souvent associée. Il est possible, grâce à l'API de géolocalisation, d'accéder aux coordonnées de l'utilisateur si celui-ci a accepté de partager sa position via le bandeau s'affichant en haut de page comme indiqué par la figure ci-dessous.



Figure 2.9_ Bandeau de de partage de la position géographique _ [Rodolphe.R, Raphaël G,2012, p.50]

On peut alors très facilement disposer d'informations telles que :

- La latitude, la longitude, et l'altitude de l'utilisateur
- Son orientation par rapport au Nord
- La vitesse à laquelle il se déplace

2.4.3.3. Le Drag and Drop

Il est également possible d'effectuer des "glisser-déposer" dans une page web. Cette fonctionnalité n'aura donc plus besoin d'être visuellement émulée par des bibliothèques JavaScript tierces mais sera présente nativement dans l'API JavaScript.

2.5 WebSocket

2.5.1 Définition

Il s'agit d'un nouveau protocole ambitieux de communication avec le serveur. Un navigateur ne peut habituellement qu'effectuer des requêtes au serveur puis recevoir sa réponse. C'est une communication unidirectionnelle (dite par canal simplex). Les WebSockets apportent la communication bidirectionnelle (dite full-duplex) entre le client et le serveur.

Les communications en temps réel sont désormais une nécessité en premier lieu pour le Web qui est devenu le plus fort pourvoyeur d'échanges entre humains. Avec la venue des applications web transcendant les banales pages statiques, les développeurs ont le besoin de reproduire en HTML ce qui existe déjà du côté des systèmes d'exploitation : des canaux bidirectionnels et connectés en permanence, que l'on nomme plus prosaïquement les « sockets » avec le protocole TCP/IP. Ces couches de bas niveau sont elles-mêmes couramment exploitées par tous les programmes souhaitant établir une connexion TCP : les navigateurs eux-mêmes, les clients de messagerie, les services de partage de fichiers, les logiciels de vidéoconférence, les jeux massivement multi-joueurs, les radios en ligne, etc.

2.5.2 Principe

Le principe des WebSockets est d'offrir un canal connecté full-duplex entre le client et le serveur directement implémenté par le navigateur et qui va donc transiter sur le même port HTTP que l'application. L'idée est d'économiser les coûts des entêtes HTTP classiques qui sont envoyés lors des requêtes AJAX. En effet, lors de l'envoi de petites quantités de données, l'approche AJAX est coûteuse car elle nécessite un entête HTTP classique d'où généralement un paquet bien plus conséquent que la taille de données utiles à envoyer.

Les WebSockets limitent la taille des paquets au strict minimum, tout en assurant une connexion ouverte, limitant également ainsi les temps de reconnexion et donc un temps de latence plus réduit qu'avec AJAX.

On gagne beaucoup de temps à utiliser une API conçue pour un besoin particulier plutôt que d'émuler avec des technologies qui n'étaient pas prévues pour.

Chapitre 2 : Messagerie Instantanée Basée sur le Web

"Les WebSockets s'inscrivent dans une approche HTML5 dans laquelle la partie présentation est gérée entièrement côté navigateur et non sur le middleware (pas de « *server-side generated HTML* »). Le serveur ne fait que gérer les messages de ses clients, ce qui le rend beaucoup plus léger, plus réactif, et donc permettant de mieux répondre à la charge." [Peter. M,2012]

Dans le cadre du protocole WebSocket pour la communication bidirectionnelle, l'application serveur et l'application cliente doivent connaître les détails du protocole. Cela signifie qu'il est nécessaire d'avoir une page Web compatible WebSocket qui appelle un point de terminaison compatible WebSocket.

Une interaction WebSocket commence par l'établissement de liaison dans laquelle les deux parties (le navigateur et le serveur) confirment mutuellement leur intention de communiquer via une connexion permanente. Ensuite, un groupe de paquets de messages est envoyé via TCP dans les deux sens. La figure 2.10 représente le fonctionnement du protocole WebSocket.

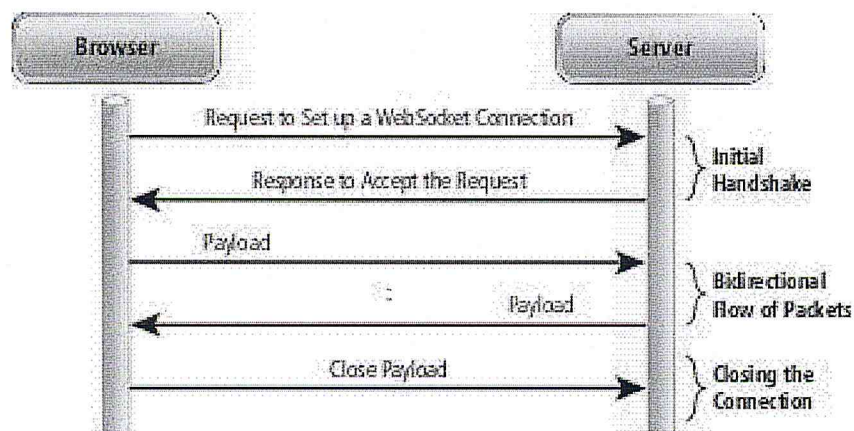


Figure 2.10 _ Schéma du protocole WebSocket _ [Peter. M,2012]

Outre les détails figurant à la ci-dessous, notons que lorsque la connexion est fermée, les deux points de terminaison échangent une trame de fermeture afin de fermer proprement la connexion. La demande d'établissement de liaison initiale est composée d'une requête HTTP ordinaire envoyée par le client au serveur Web. La requête est un HTTP GET configuré comme une demande de mise à niveau, par exemple :

Chapitre 2 : Messagerie Instantanée Basée sur le Web

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhIHNhbXBsZSBub25jZQ==
Origin: http://example.com
```

Avec HTTP, une requête du client avec l'en-tête Upgrade indique l'intention du client de demander à ce que le serveur passe à un autre protocole. Avec le protocole WebSocket, la demande de mise à niveau effectuée au serveur contient une clé unique que le serveur retournera altérée comme preuve qu'il a accepté la demande de mise à niveau. Il s'agit d'une démonstration pratique permettant de prouver que le serveur comprend le protocole WebSocket. Voici un exemple de réponse à une demande d'établissement de liaison :

```
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

"Le code de statut de réussite est toujours 101. Tout autre code de statut sera interprété comme un refus de mettre à niveau vers le protocole WebSocket.

Le client peut également envoyer d'autres en-têtes tels que Sec-WebSocket-Protocol afin d'indiquer quels sous-protocoles il est prêt à utiliser. Un sous-protocole est un protocole au niveau de l'application qui est créé sur le protocole WebSocket de base. Si le serveur comprend certains des sous-protocoles suggérés, il en choisit un et renvoie son nom au client via le même en-tête.

À la suite de l'établissement de liaison, le client et le serveur peuvent envoyer des messages librement via le protocole WebSocket. La charge commence par un opcode qui indique l'opération effectuée. L'un de ces opcodes, à savoir 0x8, indique une requête pour fermer la session. Notons que les messages WebSocket sont asynchrones et qu'une demande d'envoi ne recevra donc pas nécessairement une réponse immédiate, comme avec HTTP. Avec le protocole WebSocket, il est conseillé de réfléchir en termes de messages généraux allant du

Chapitre 2 : Messagerie Instantanée Basée sur le Web

client vers le serveur, ou inversement, et d'oublier le modèle classique demande/réponse de HTTP." [Dino E, 2012]

2.5.3. Prise en charge du protocole WebSocket par les navigateurs

Le tableau ci-dessous présente un résumé de la prise en charge du protocole WebSocket par les navigateurs les plus populaires à l'heure actuelle.

Navigateur	Prise en charge de WebSocket
Internet Explorer	Le protocole WebSocket sera pris en charge dans Internet Explorer 10. Les applications de type Metro écrites à l'aide de JavaScript et HTML5 prendront également en charge WebSocket.
Firefox	WebSocket est pris en charge depuis la version 6 du navigateur lancée mi-2011. Une prise en charge très précoce était proposée dans la version 4, mais elle a été abandonnée dans la version 5.
Chrome	WebSocket est pris en charge depuis la version 14, lancée en septembre 2011.
Opera	La prise en charge de WebSocket a été retirée dans la version 11.
Safari	Prend en charge une version antérieure du protocole WebSocket.

Tableau 2.2_ Prise en charge de WebSocket par les principaux navigateurs _ [Dino E, 2012]

À l'exception de Firefox, il est possible de vérifier par programme si WebSocket est pris en charge par un navigateur en examinant l'objet "window.WebSocket". Pour Firefox, il faut actuellement vérifier l'objet "MozWebSocket". Il est important de noter que la plupart des fonctionnalités liées à HTML5 peuvent être vérifiées dans les navigateurs à l'aide d'une bibliothèque spécialisée, telle que Modernizr.

2.6. L'API WebRTC

L'API WebRTC (Web Real-Time-Communication) s'inscrit dans le cadre d'un projet qui porte le même nom et qui vise à intégrer des fonctions de communication temps-réel dans les navigateurs Web.

2.6.1 Objectifs de WebRTC

Les fonctions offertes par le navigateur Web sont de plus en plus étendues et lui permettent de déborder de son rôle historique d'affichage de pages Web pour devenir un véritable environnement d'exécution d'applications riches. Cette tendance a ainsi été popularisée par des services tels que GMail et la suite Google Apps.

Les fonctions multimédias et de communications interactives ont jusqu'à récemment été rendues possibles uniquement par l'utilisation de plugins propriétaires tels que Flash Player (les objets de type ActiveX, Java ou autres n'ayant eu qu'un succès très limité dans ce cadre) qui a permis notamment le développement massif de la vidéo Web avec des services comme YouTube. Il devient maintenant possible de téléphoner à partir d'une page Web intégrant un applet Flash.

Le développement de la technologie HTML5 qui intègre une multitude de nouvelles fonctions permet maintenant de standardiser ces fonctionnalités. Ainsi les services de diffusion vidéo (Youtube par exemple) ou de partage de documents (tel que Slideshare) sont engagés dans une logique de remplacement de fonctions auparavant développées en Flash par du HTML5. Ces fonctions restent pour l'instant relativement simples et sans interactions complexes mais l'évolution rapide des standards permet maintenant d'envisager le développement d'applications Web plus sophistiquées.

L'initiative WebRTC vise ainsi à standardiser l'infrastructure implémentée au sein des navigateurs Web permettant d'établir des communications audio et vidéo, directes, interactives et en temps-réel, entre des utilisateurs Web.

Cet effort se déroule au sein de deux groupes de travail :

- à l'IETF dans le cadre du RTCWeb Working Group pour la définition des formats et protocoles utilisés pour la communication entre navigateurs
- au W3C dans le cadre du WebRTC Working Group pour la définition des APIs mises à disposition des applications Web

Ces deux groupes ont démarré depuis peu et de nombreux points restent ouverts et sujets à (vives) discussions, le sujet intéressant les plus grands acteurs de l'Internet (Google, Cisco, Mozilla, Apple, Skype ...).

2.6.2. Description de l'API

L'API WebRTC, encore à l'état de *working draft*, est développée par le W3C (*World Wide Web Consortium*) et utilise de nombreux éléments introduits en HTML5 (balise *video* par exemple).

Notre objectif donc est de décrire le fonctionnement global de l'établissement de session entre deux navigateurs. L'implémentation d'une telle fonction repose sur 3 phases principales : l'acquisition des ressources audio/vidéo dans le navigateur de l'émetteur, la transmission de ces flux sur le réseau, et le rendu sur le navigateur du récepteur. Nous décrivons ainsi les deux principales primitives WebRTC permettant d'atteindre ce but : la *Stream API* (appelée aussi *MediaStream API*) qui permet la manipulation de ressources multimédias et l'interface *RTCPeerConnection* (anciennement *PeerConnection*) qui assure la communication à travers le réseau. Nous présentons les principes généraux de ces deux primitives sans rentrer, à ce stade, dans les détails de l'API car ces derniers ne sont pas stabilisés et vont encore faire l'objet de nombreuses modifications. Notons par ailleurs que l'API WebRTC comporte une troisième primitive, la *RTCDataChannel* (anciennement *DataChannel*), qui permet d'envoyer des données (autres que des flux audio/vidéo) entre deux navigateurs.

2.6.3. Stream API

La *Stream API*, et en particulier son objet *MediaStream*, est au cœur du fonctionnement de WebRTC. Toute la partie multimédia passe par elle, quelle que soit la source et la destination du trafic.

Un flux média (*media stream* en anglais) contient et traite les pistes audio et vidéo. Ces flux peuvent provenir de différentes sources (locales ou distantes), et sont émis à des destinations variables (locales ou distantes également). Au sein de cette interface, seul l'objet *MediaStream* reste immuable quelle que soit le cas de figure considéré.

Les différentes étapes illustrées sur la figure 2.11 sont les suivantes :

1) Construction de l'objet *MediaStream* : L'objet *MediaStream* prend en paramètre soit une connexion distante (*RTCPeerConnection*), soit un objet *LocalMediaStream* ayant accès aux

Chapitre 2 : Messagerie Instantanée Basée sur le Web

ressources locales du terminal. Ces accès, pour des raisons de sécurité, doivent être explicitement donnés par l'utilisateur au cas par cas, via l'usage de boutons (« Diffuser ma vidéo » pour autoriser le flux vidéo par exemple) ou par des préférences dédiées à chaque page et gérées par le navigateur.

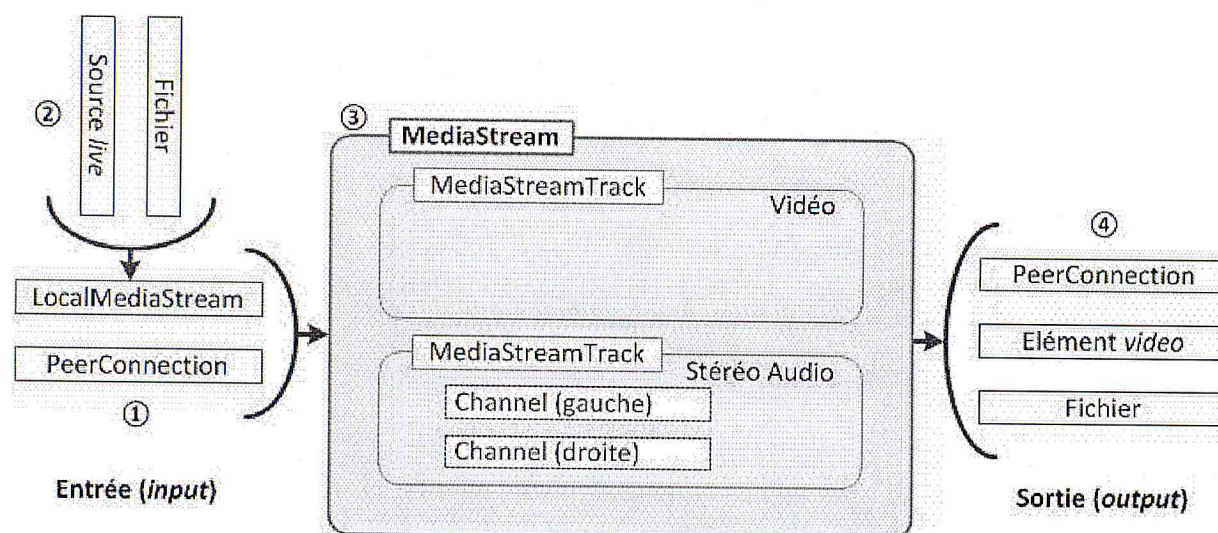


Figure 2.11_ Les différentes étapes du fonctionnement de la Stream API _ [Odin .G ,2012,p.1]

La fonction `getUserMedia ()` récupère, simultanément ou séparément (il est possible de l'appeler plusieurs fois), les différentes sources afin de les traiter par la suite.

2) Sources locales : Ces ressources peuvent émaner de différentes origines, notamment d'équipements de capture en temps réel (microphone, webcam, caméra ...) ou de fichiers locaux au terminal. Le support de ces différentes sources est assuré grâce aux nouvelles fonctionnalités offertes par HTML5 (*File API, Media Capture*).

3) Gestion des différents flux : Un *MediaStreamTrack* est créé pour chacun des flux, audio ou vidéo. Puisqu'il peut exister de multiples flux simultanés (plusieurs microphones distincts par exemple), ceux-ci sont stockés au sein de listes de type *MediaStreamTrackList*. Ces listes sont utilisées pour créer le flux média (*MediaStream*). Il est ici intéressant de noter plusieurs points :

- Grâce à ces listes, l'objet *MediaStream* est dynamique. Il devient en effet possible d'ajouter et de retirer à la volée des flux multimédias sans avoir besoin de recréer un nouvel objet (fonction `add()` de l'objet *MediaStreamTrackList*).

- Les pistes audio peuvent contenir un nombre variable de canaux (mono, stéréo, 5.1 ...), permettant leur utilisation dans de multiples situations différentes. Ces canaux sont les plus petites unités d'un *MediaStream*.

4) **Utilisation du *MediaStream*** : Cet objet *MediaStream* peut ensuite être utilisé dans divers cas de figures :

- le flux peut être directement affiché dans le navigateur local, ce qui peut servir à afficher le retour d'une webcam locale ou le flux en provenance d'un tiers via un objet *RTCPeerConnection* en entrée ;
- les flux peuvent être enregistrés dans des fichiers locaux ;
- les flux peuvent être envoyés à un tiers à l'aide d'un objet *PeerConnection*, ce qui est l'objectif même de WebRTC.

La pratique veut que ces objets *MediaStream* disposent d'une URL pour l'objet binaire qu'ils représentent (BLOB ou *Binary Large Object*). La fonction *createObjectURL* crée cette URL. Les données de l'objet que cette URL représente doivent être compréhensibles directement par les éléments HTML5 *audio* ou *video*.

Dernier point intéressant au sujet de cette *Stream API*, il est possible d'adjoindre à ce schéma général un objet *MediaStreamRecorder* permettant, comme son nom le laisse suggérer, d'enregistrer en temps réel les flux audio/vidéo. Cet enregistrement est représenté sous la forme d'un BLOB. Cet enregistreur doit être capable de gérer efficacement l'ajout ou la suppression de *MediaStreamTrack* du flux multimédia.

La figure 2.12 illustre un cas d'usage typique du fonctionnement de *MediaStream*. La fonction *getUserMedia* () (1), déclenchée par un bouton, autorise l'accès aux différents équipements (microphone, webcam ...) (2) afin d'en récupérer les flux audio/vidéo.

Ces *LocalMediaStream* ou *MediaStreamTrack* sont traités par la fonction appelée (*callback*) en cas de succès, afin de les ajouter aux *MediaStreamTrackList* d'un objet *MediaStream*, créant effectivement le flux média. Celui-ci peut ensuite être ajouté à une connexion sortante (4) *PeerConnection* afin de transmettre le flux à un destinataire.

2.6.4 PeerConnection API

Si l'objet *MediaStream* est incontournable pour la manipulation des flux multimédias, l'objet *PeerConnection* de son côté est primordial pour l'établissement d'une connexion, de manière transparente, entre deux entités RTCWeb. Définie au sein de la même spécification que

Chapitre 2 : Messagerie Instantanée Basée sur le Web

la *Stream API*, cette API risque d'être altérée en profondeur dans les mois à venir puisque l'IETF est en train de définir un *Framework* d'établissement de session pour RTCWeb/WebRTC (JSEP ou *JavaScript Session Establishment Protocol* actuellement en *draft00*) qui va avoir un impact certain sur les routines de l'objet *PeerConnection*. Partant de ce constat, nous faisons ici le choix de n'énoncer que les éléments généraux qui seront probablement traités par l'objet, d'une manière ou d'une autre.

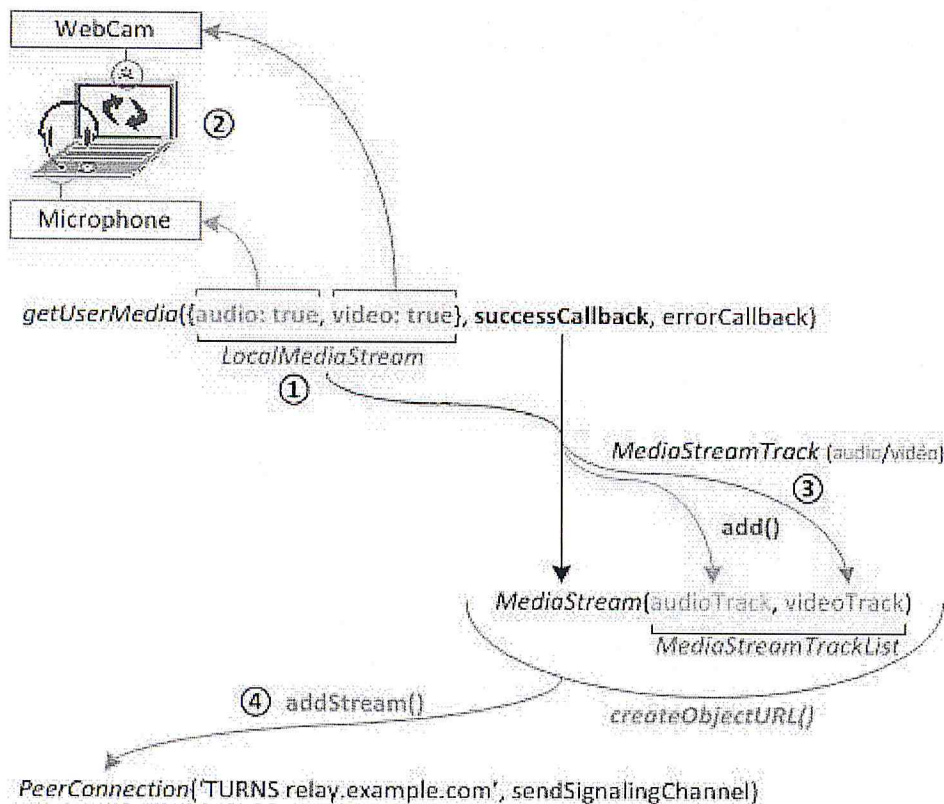


Figure 2.12_ Fonctionnement des API MediaStream et PeerConnection _ [Odin .G ,2012,p.2]

L'objet *PeerConnection* gère l'ensemble de l'établissement de connexion avec la partie distante. Actuellement, comme décrit dans la figure ci-dessus, lors de sa création, il est possible de remarquer qu'il prend deux paramètres qui traitent de deux aspects bien distincts :

- Son premier paramètre spécifie un serveur STUN/TURN pour la traversée de NAT des flux multimédias.
- Son second paramètre définit la fonction à appeler (*callback*) pour envoyer des messages de signalisation. Il est dès lors possible de manipuler et personnaliser les messages envoyés au destinataire via cette fonction.

Chapitre 2 : Messagerie Instantanée Basée sur le Web

La *signaling callback* passée en second paramètre permet de générer les messages de signalisation, nécessaires pour échanger les informations de connexion, grâce à SDP. Le protocole de signalisation en lui-même (SIP, Jabber/Jingle, propriétaire) est laissé au libre choix de l'implémentation. Afin de permettre l'utilisation de SIP dans ce cadre, un *draft* est en discussion au sein de l'IETF afin d'ajouter un transport de type WebSocket au protocole SIP.

Encore peu utilisé au sein de la VoIP actuellement, le protocole ICE va gagner, avec ces implémentations, ses lettres de noblesse puisqu'il prend ici une place centrale dans la traversée des NAT des flux multimédias. Il a pour objectif de négocier automatiquement et de manière totalement transparente la connexion en effectuant de multiples tests de connectivité entre les deux terminaux (ici, les navigateurs) après avoir récupéré son adresse publique et une éventuelle adresse relai grâce respectivement au serveur STUN ou TURN passé en paramètre. L'efficacité de ce protocole n'est plus à prouver et son déploiement dans ce cadre renforce encore la facilité d'accès par les utilisateurs finaux aux applications développées sur cette base.

2.6. Fonctionnement générale de l'application

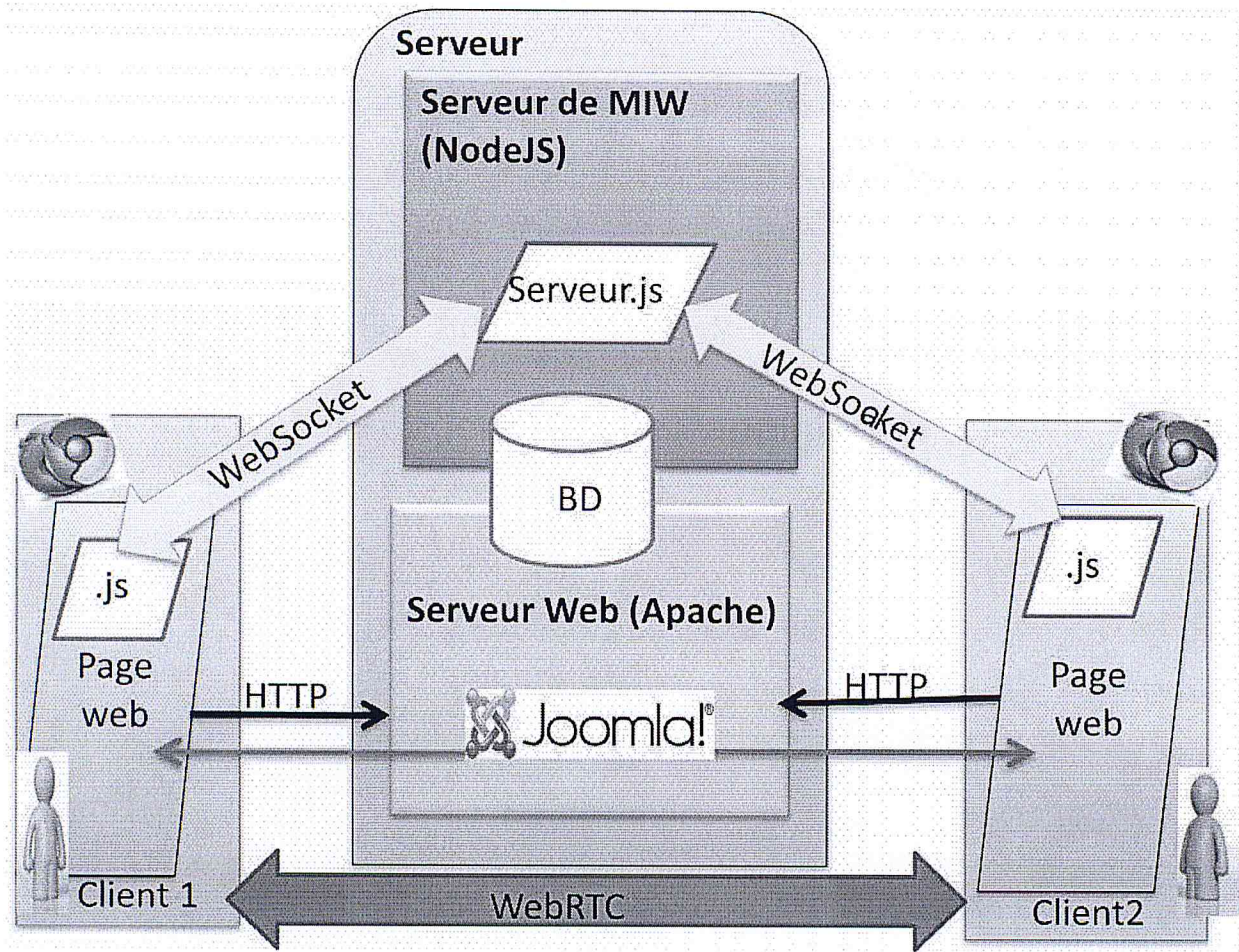
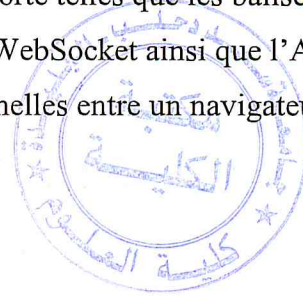


Figure 2.13 : Fonctionnement générale de l'application

2.7. Conclusion

Dans ce chapitre, nous avons fait un tour d'horizon sur les différents éléments théoriques et technologiques que nous utiliserons par la suite pour la réalisation d'une solution de messagerie instantanée via le web.

Nous avons commencé par présenter ce qu'est la messagerie instantanée "traditionnelle" puis nous avons abordé un type de MI dont la popularité ne cesse de croître ces dernières années et qui est connu sous l'appellation de messagerie instantanée via le web. Puis nous avons présenté HTML5 en mettant l'accent sur les nouveautés qu'il apporte telles que les balises multimédia audio et video. Enfin, nous avons introduit le protocole WebSocket ainsi que l'API WebRTC qui permettent d'établir des communications bidirectionnelles entre un navigateur et un serveur et entre deux navigateurs respectivement.



Chapitre 3

Implémentation et Réalisation

Introduction

Ce chapitre constitue l'âme du processus de développement du logiciel et a pour objectif la mise en œuvre de chacun des modules décrits dans le chapitre précédent. Nous consacrons la première partie à la présentation de l'environnement de l'application. Par la suite, nous exposerons quelques interfaces homme machine qui concordent avec les fonctionnalités du système.

3.1. Environnement logicielle :

Le long de la phase de développement, nous avons utilisé l'environnement logiciel suivant :

Système d'exploitation : Microsoft Windows 7

Outils de développement : NetBeans 7.0.1.

Serveur d'application : WampServer 2.2

Conception et modélisation en UML : StarUml 5.0

3.2. Choix technique :

Durant ce cycle de développement, nous avons eu recours à plusieurs outils à savoir node.js et son module socket.io, JavaScript et sa célèbre librairie jQuery, easyRTC, Shell, PHP5, MySQL, comme outils de travail.

3.2.1. Choix de javascript :

JavaScript est un langage qui avait connu plusieurs vies. Pour situer les choses, je dirais même qu'il a connu 3 vies :

1. Dans les années 90, on parlait de DHTML (Dynamic HTML). On utilisait en fait les toutes premières versions de JavaScript pour faire des petits effets dans ses pages web : afficher une image lors d'un clic sur un bouton par exemple.

C'était l'époque de Netscape et d'Internet Explorer 5.5

Chapitre 3 : Implémentation et réalisation

2. Dans les années 2000, on a commencé à utiliser le langage pour créer des interfaces côté client. C'est là que des bibliothèques comme jQuery ou Mootools sont apparues. Aujourd'hui, cet usage de JavaScript est très répandu et mature. On a pris l'habitude de manipuler le DOM (Document Object Model) pour affecter ses balises HTML en JavaScript et leur faire subir toutes sortes de traitements.

3. Puis, aux alentours de 2010, JavaScript est entré dans une nouvelle ère. Google a commencé à rendre le langage beaucoup plus rapide avec l'apparition du navigateur Google Chrome. Avec ce navigateur est né le moteur d'exécution V8 qui a considérablement permis d'accélérer l'exécution de code JavaScript (j'y reviendrai). Des outils comme Node.js sont ensuite apparus. Les bibliothèques dont le nom finit par .js se sont multipliées : Backbone.js, Ember.js, Meteor.js.

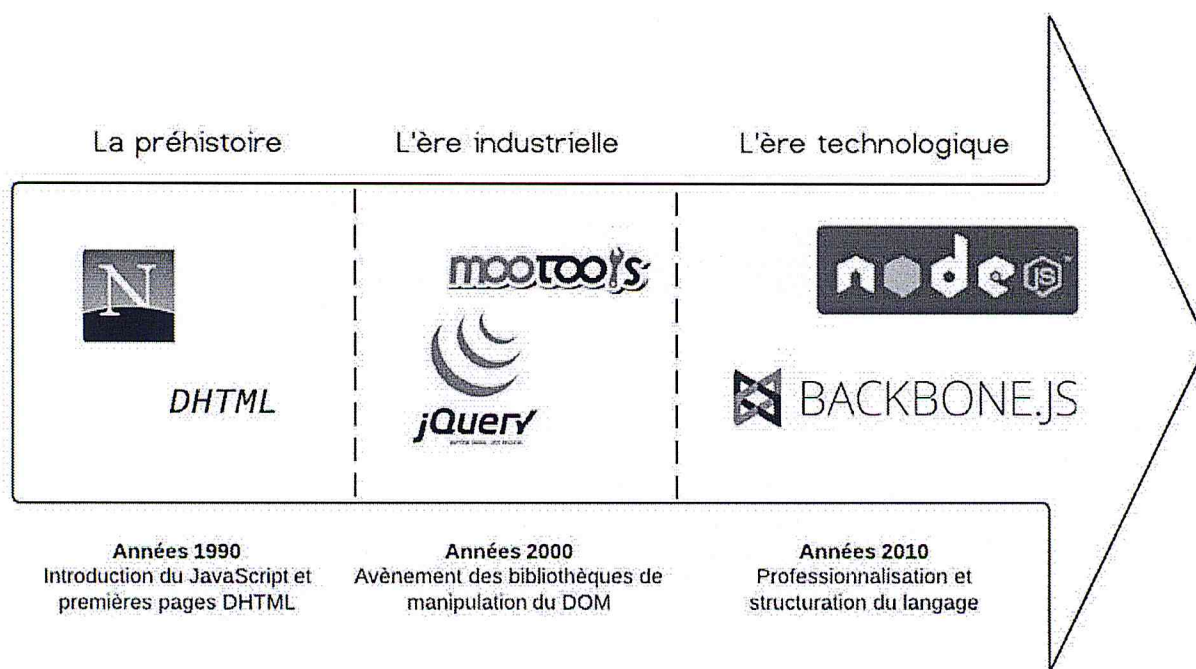


Figure 3.1_ Les 3 vies de JavaScript _ [Mathieu. N,2013,p.10]

Node.js nous permet d'utiliser le langage JavaScript sur le serveur. Il nous permet donc de faire du JavaScript en dehors du navigateur.

Node.js bénéficie de la puissance de JavaScript pour proposer une toute nouvelle façon de développer des sites web dynamiques.

3.2.2. Choix de node.js

3.2.2.1. JavaScript Coté serveur :

Jusqu'ici, JavaScript avait toujours été utilisé du côté du client, c'est-à-dire du côté du visiteur qui navigue sur notre site. Le navigateur web du visiteur (Firefox, Chrome, IE...) exécute le code JavaScript et effectue des actions sur la page web.

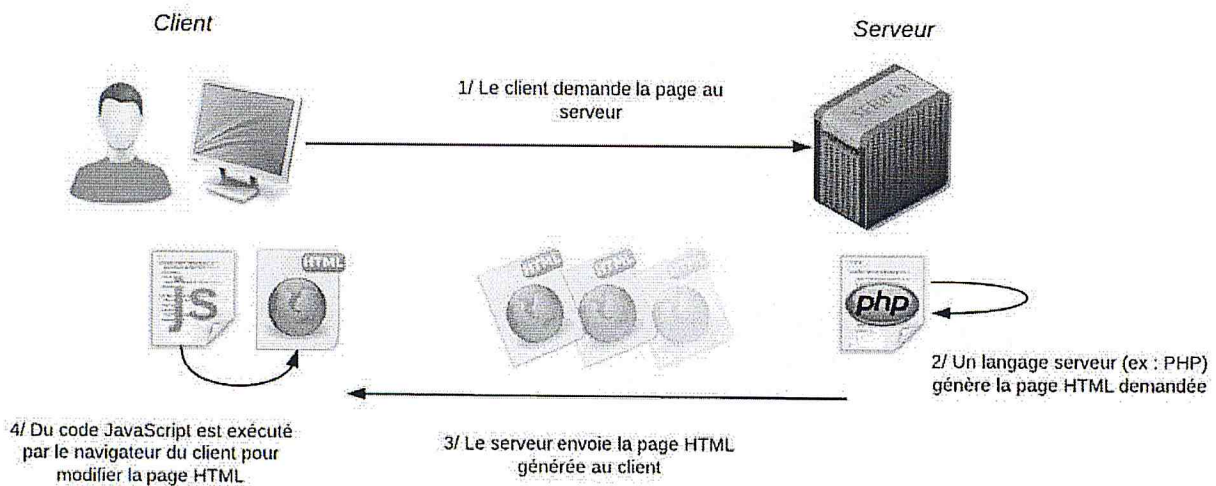


Figure 3.2_ Le schéma classique : PHP sur le serveur, JavaScript chez le client _ [Mathieu. N,2013,p.30]

On peut toujours utiliser du JavaScript côté client pour manipuler la page HTML. Ca, ça ne change pas.

Par contre, Node.js offre un environnement côté serveur qui nous permet aussi d'utiliser le langage JavaScript pour générer des pages web. En gros, il vient en remplacement de langages serveur comme PHP, Java EE, etc.

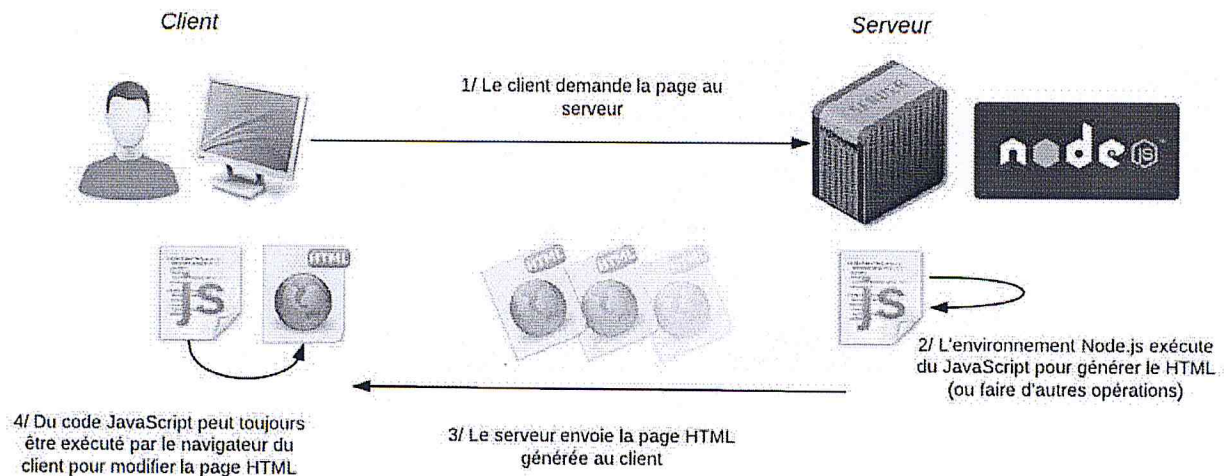


Figure 3.3_ Avec Node.js, on peut aussi utiliser du JavaScript sur le serveur _ [Mathieu. N,2013,p.31]

JavaScript est un langage basé sur les événements, donc Node.js est lui-même basé sur les événements. Du coup, c'est toute la façon d'écrire des applications web qui change ! Et c'est de là que Node.js tire toute sa puissance et sa rapidité.

Avec Node.js, vous pouvez créer des applications rapides comme :

- Un serveur de Chat
- Un système d'upload très rapide
- et de façon générale n'importe quelle application qui doit répondre à de nombreuses requêtes rapidement et efficacement, en temps réel.

Si Node.js est rapide, cela tient principalement à deux choses : le moteur V8 et son fonctionnement non bloquant.

3.2.2.2. Le moteur V8

Node.js utilise le moteur d'exécution ultrarapide V8 de Google Chrome. Ce moteur V8 avait fait beaucoup parler de lui à la sortie de Google Chrome, car c'est un outil open source créé par Google qui analyse et exécute du code JavaScript très rapidement.

Jusqu'à la sortie de Chrome, la plupart des navigateurs lisaient le code JavaScript de façon peu efficace : le code était lu et interprété au fur et à mesure. Le navigateur mettait beaucoup de temps à lire le JavaScript et à le transformer en code machine compréhensible pour le processeur.

Chapitre 3 : Implémentation et réalisation

Le moteur V8 de Google Chrome, qui est réutilisé ici par Node.js, fonctionne complètement différent. Très optimisé, il fait ce qu'on appelle de la compilation JIT (Just In Time). Il transforme le code JavaScript très rapidement en code machine et l'optimise même grâce à des procédés complexes.

3.2.2.3. Le modèle non bloquant

Comme JavaScript est un langage conçu autour de la notion d'évènement, Node.js a pu mettre en place une architecture de code entièrement non bloquante.

❖ Modèle bloquant vs modèle non bloquant

Imaginez un programme dont le rôle est de télécharger un fichier puis de l'afficher. Voici comment on écrirait le code dans un **modèle bloquant** :

```
Télécharger un fichier
```

```
Afficher le fichier
```

```
Faire autre chose
```

Les actions sont effectuées dans l'ordre. Il faut lire les lignes de haut en bas :

1. Le programme va télécharger un fichier sur Internet
2. Le programme affiche le fichier à l'utilisateur
3. Puis ensuite le programme peut faire d'autres choses (effectuer d'autres actions)

Maintenant, on peut écrire le même code sur un **modèle non bloquant** :

```
Télécharger un fichier
```

```
Dès que c'est terminé, afficher le fichier
```

```
Faire autre chose
```

Le programme n'exécute plus les lignes dans l'ordre où elles sont écrites. Il fait ceci :

1. Le programme lance le téléchargement d'un fichier sur Internet

Chapitre 3 : Implémentation et réalisation

2. Le programme fait d'autres choses (le programme suit son cours)
3. Dès que le téléchargement est terminé, le programme effectue les actions qu'on lui avait demandées : il affiche le fichier

Schématiquement, l'exécution du programme peut donc se représenter comme ça :

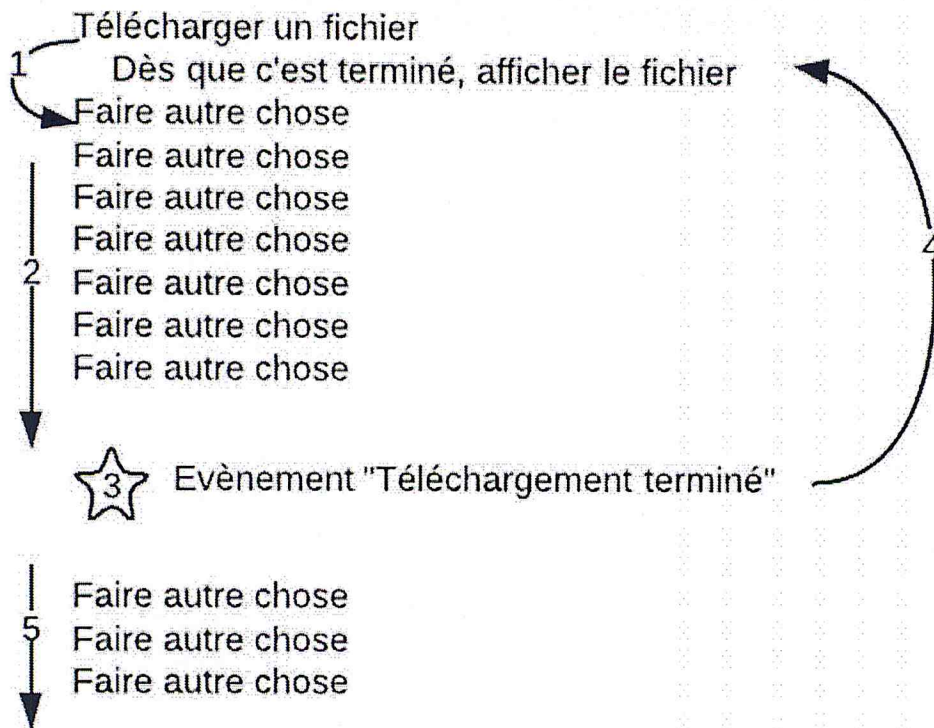


Figure 3.4_ Le modèle non bloquant en programmation _ [Mathieu .N,2013,p.24]

Node.js nous évite de perdre du temps en nous permettant de faire d'autres choses en attendant que les actions longues soient terminées

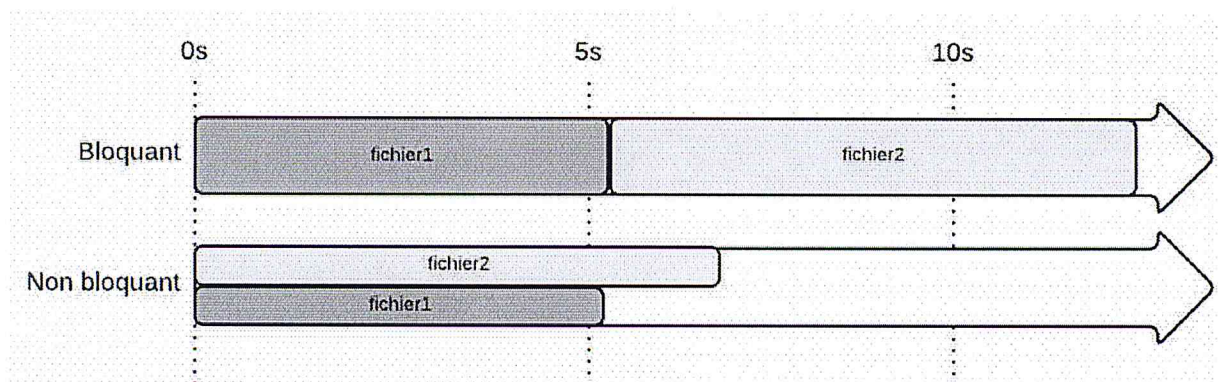
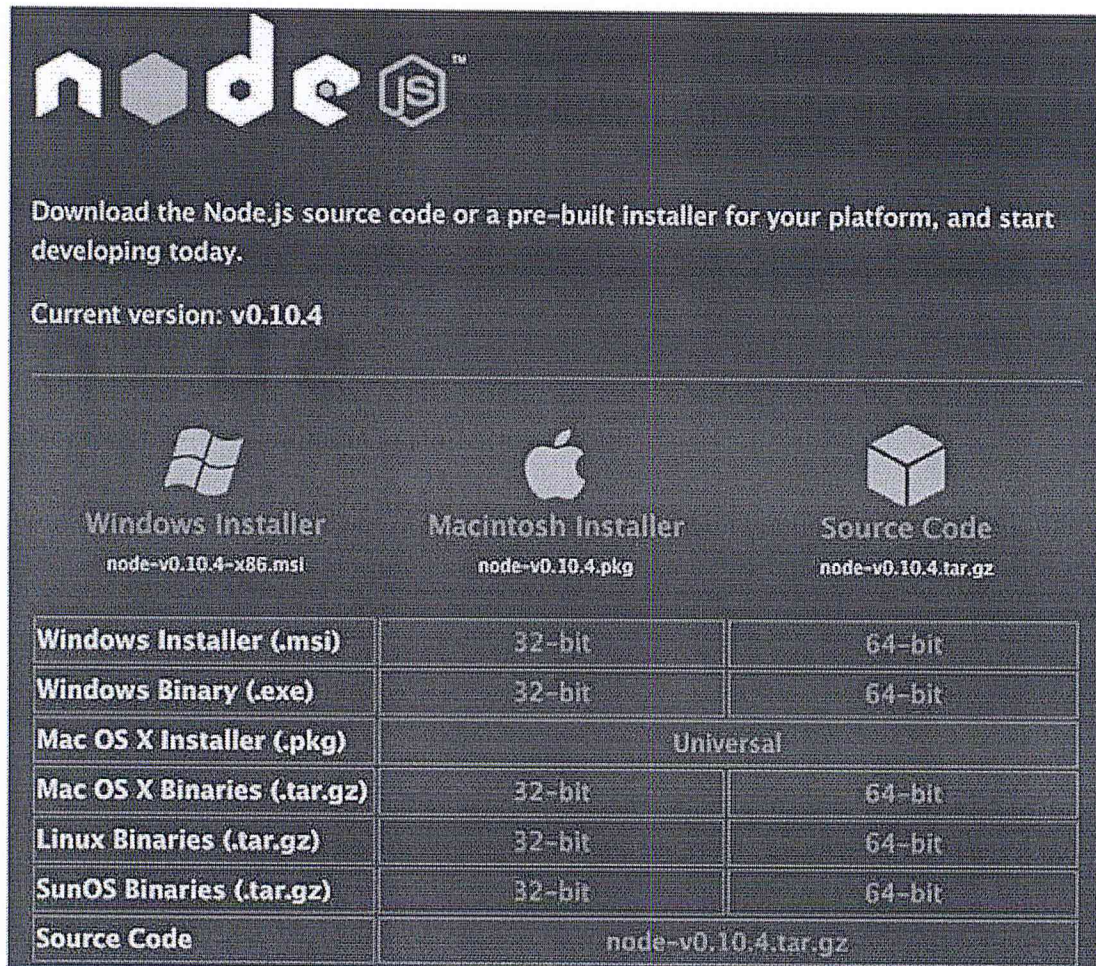


Figure 3.5_ Différence entre le modèle bloquant et non bloquant _ [Mathieu .N,2013,p.25]

3.2.2.4. Installation de node.js

Pour installer Node.js sous Windows, il suffit de télécharger l'installeur qui est proposé sur le site de Node.js. Cliquez simplement sur le lien Install.

Vous pouvez aussi vous rendre sur la page des téléchargements pour avoir plus d'options :



Download the Node.js source code or a pre-built installer for your platform, and start developing today.

Current version: **v0.10.4**

Windows Installer
node-v0.10.4-x86.msi

Macintosh Installer
node-v0.10.4.pkg

Source Code
node-v0.10.4.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.exe)	32-bit	64-bit
Mac OS X Installer (.pkg)	Universal	
Mac OS X Binaries (.tar.gz)	32-bit	64-bit
Linux Binaries (.tar.gz)	32-bit	64-bit
SunOS Binaries (.tar.gz)	32-bit	64-bit
Source Code	node-v0.10.4.tar.gz	

Figure 3.6_ La page de téléchargement de Node.js _ [Mathieu .N,2013,p.5]

Vous pouvez télécharger soit le .msi, soit le .exe (le résultat sera le même).

En fait, vous devriez voir avoir 2 programmes installés :

- **Node.js** : c'est l'interpréteur de commandes de Node.js. Nous l'utiliserons assez peu en pratique. Il sert à tester des commandes JavaScript.

Chapitre 3 : Implémentation et réalisation

- **Node.js command prompt** : c'est une console de Windows configurée pour reconnaître Node.js. C'est par là que vous passerez pour lancer vos programmes Node.js, c'est donc ce que nous utiliserons le plus souvent.

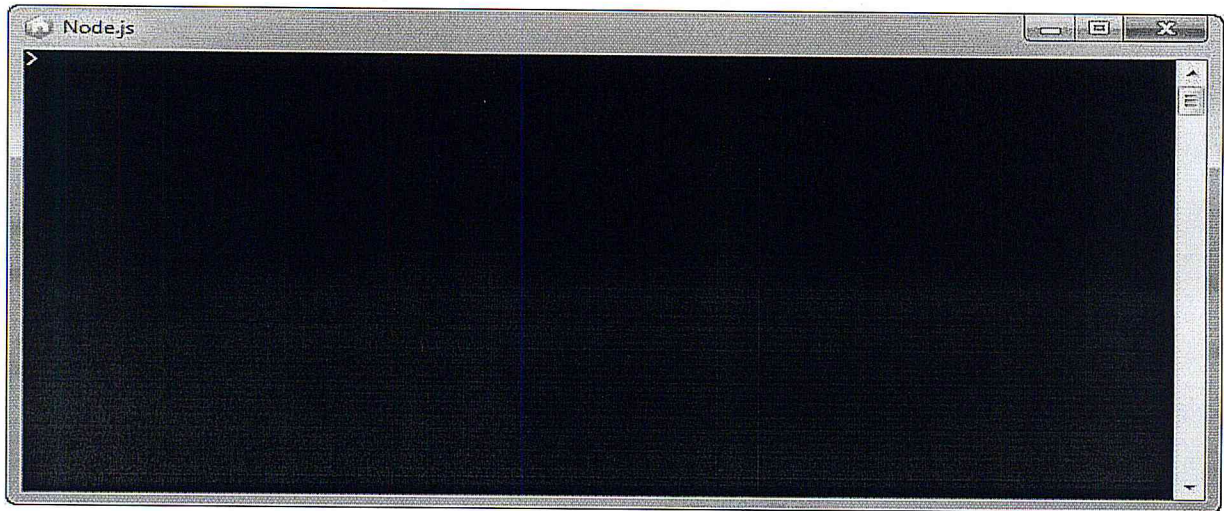


Figure 3.7_ L'interpréteur Node.js sous Windows (sera peu utilisé) _ [Mathieu .N,2013,p.6]

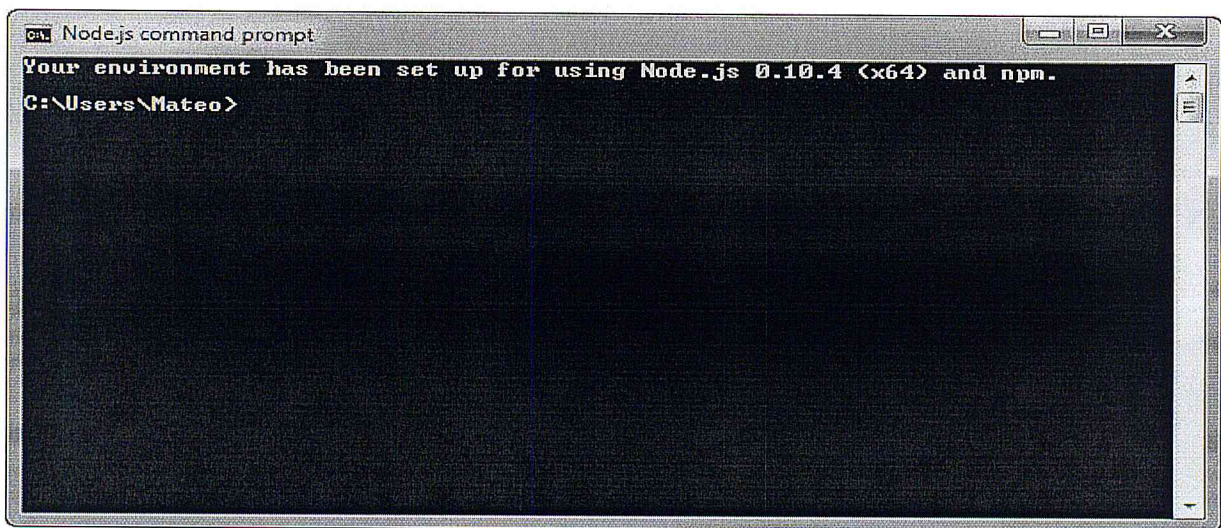


Figure 4.8_ Node.js command prompt - La console Node.js (sera fréquemment utilisée) _ [Mathieu .N,2013,p.7]

3.2.2.5. Des serveurs web et des threads

Node.js est bas niveau. Tellement bas niveau que vous allez devoir faire des choses que vous n'avez pas l'habitude de faire pour que votre programme fonctionne correctement.

Quand vous créez des sites web avec PHP par exemple, vous associez le langage avec un serveur web HTTP comme Apache. Chacun se répartit les rôles :

Chapitre 3 : Implémentation et réalisation

- Apache gère les demandes de connexion HTTP au serveur. C'est lui qui fait en quelque sorte la circulation et qui gère les entrées/sorties.
- PHP exécute le code des fichiers .php et renvoie le résultat à Apache, qui se charge à son tour de l'envoyer au visiteur.

Comme plusieurs visiteurs peuvent demander une page en même temps au serveur, Apache se charge de les répartir et de les exécuter en parallèle dans des *threads* différents. Chaque thread utilise un processeur différent sur le serveur (ou un noyau de processeur) :

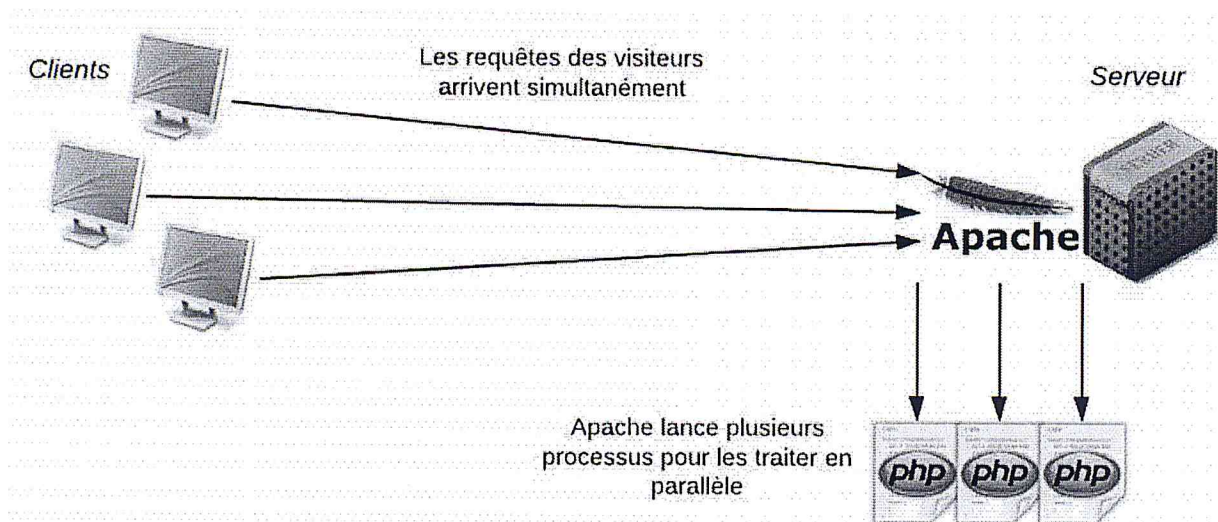


Figure 3.9 _ Le serveur Apache est multithread _ [Mathieu .N,2013,p.38]

Avec Node.js, on n'utilise pas de serveur web HTTP comme Apache. En fait, c'est à nous de créer le serveur.

Node.js est **monothread**, contrairement à Apache. Cela veut dire qu'il n'y a qu'un seul processus, qu'une seule version du programme qui peut tourner à la fois en mémoire.

En effet, il ne peut faire qu'une chose à la fois et ne tourne donc que sur un noyau de processeur. Mais il fait ça de façon ultra efficace, et malgré ça il est quand même beaucoup plus rapide !

Cela est dû à la nature "orientée événements" de Node.js. Les applications utilisant Node ne restent jamais les bras croisés sans rien faire. Dès qu'il y a une action un peu longue, le programme redonne la main à Node.js qui va effectuer d'autres actions en attendant qu'un événement survienne pour dire que l'opération est terminée.

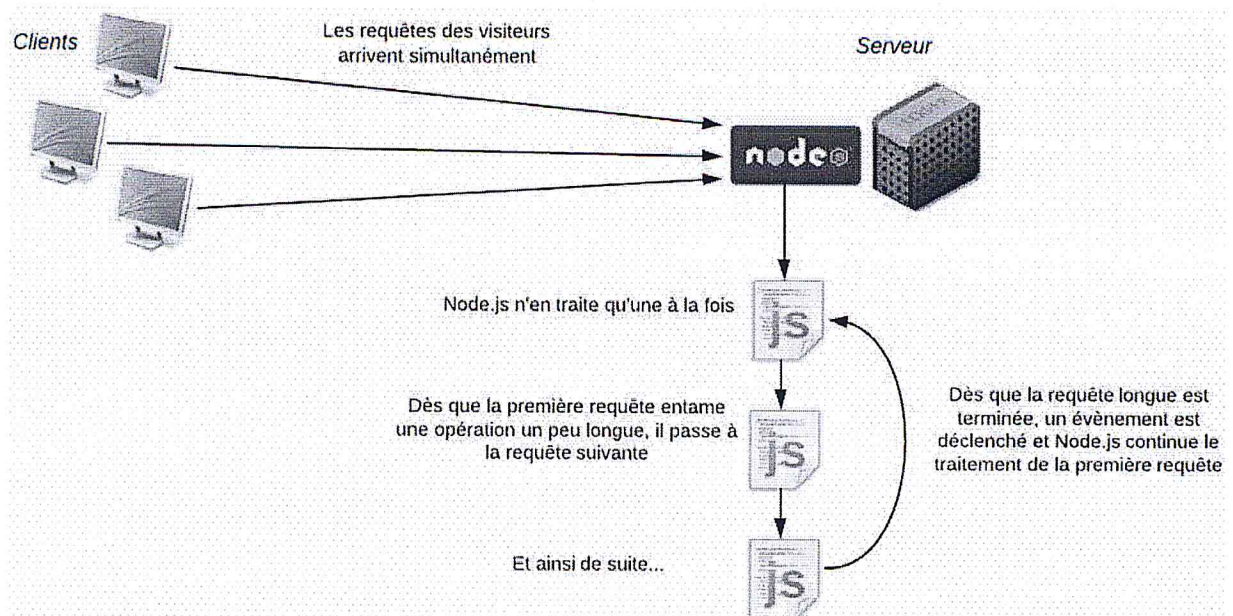


Figure 3.10_ La souplesse de Node.js Grace aux événements _ [Mathieu .N,2013,p.39]

3.2.2.6. Les modules Node.js et NPM

Le noyau de Node.js est tout petit. De base, Node.js ne sait en fait pas faire grand chose. Pourtant, Node.js est très riche grâce à son extensibilité. Ces extensions de Node.js sont appelées **modules**.

Il existe des milliers de modules qui offrent des fonctionnalités variées : de la gestion des fichiers uploadés à la connexion aux bases de données MySQL ou à Redis, en passant par des Frameworks, des systèmes de Templates et la gestion de la communication temps réel avec le visiteur ! Il y a à peu près tout ce dont on peut rêver et de nouveaux modules apparaissent chaque jour.

1- Utiliser NPM pour installer des modules

NPM (le Node Package Manager) est un moyen (formidable) d'installer de nouveaux modules développés par la communauté.

NPM est un peu l'équivalent d'`apt-get` sous Linux pour installer des programmes. Une simple commande et le module est téléchargé et installé

En plus, NPM gère les dépendances. Cela signifie que, si un module a besoin d'un autre module pour fonctionner, NPM ira le télécharger automatiquement.

NPM possède un site web : <http://npmjs.org>

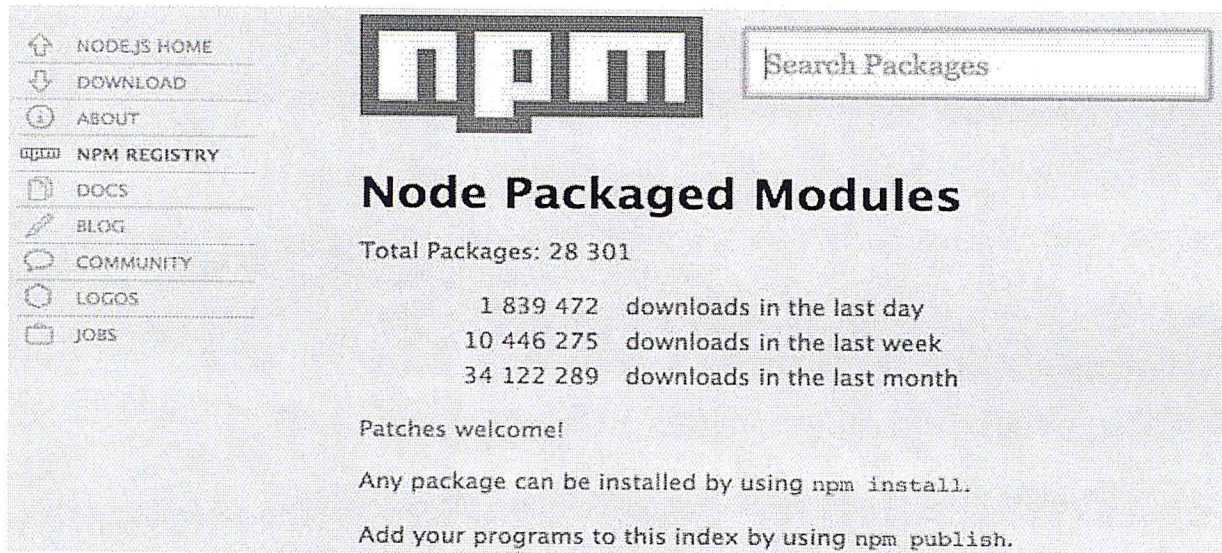


Figure 3.11_ Le site web de NPM _ [Mathieu .N,2013,p.8]

Comme vous pouvez le voir, NPM est très actif. Il y a plusieurs dizaines de milliers de modules disponibles, et on compte plusieurs millions de téléchargements par semaine.

Comme le dit le site web de NPM, il est aussi simple d'installer de nouveaux modules que de publier ses propres modules. C'est en bonne partie ce qui explique le grand succès de Node.js.

1.1.Trouver un module

Si vous savez ce que vous cherchez, le site web de NPM vous permet de faire une recherche. Mais NPM, c'est avant tout une commande. Vous pouvez faire une recherche dans la console, comme ceci :

```
npm search postgresql
```

1.2.Installer un module

Pour installer un module, rien de plus simple. Placez-vous dans le dossier de votre projet et tapez

```
npm install nomdumodule
```

Le module sera installé localement spécialement pour votre projet.

Si vous avez un autre projet, il faudra donc relancer la commande pour l'installer à nouveau pour cet autre projet. Cela vous permet d'utiliser des versions différentes d'un même module en fonction de vos projets.

1.3. Mettre à jour les modules

Avec une simple commande :

```
npm update
```

NPM va chercher sur les serveurs s'il y a de nouvelles versions des modules, puis mettre à jour les modules installés sur votre machine (en veillant à ne pas casser la compatibilité) et il supprimera les anciennes versions. Parmi les modules existants on a choisis socket.io

3.2.3. Choix de socket.io

Socket.IO a pour objectif de rendre les applications Web temps-réel possibles dans tous les navigateurs. Il permet de simplifier la logique pour déterminer la façon de gérer convenablement les WebSockets sur un client. Socket.IO utilisera Adobe Flash en dernier recours (fallback) si le WebSocket HTML 5 n'est pas supporté sur le navigateur.

1- Utilisation de socket.io

Code serveur (utilise node.js)

```
var io = require('socket.io').listen(80);  
  
io.sockets.on('connection', function (socket) {  
  
  socket.emit('actualites', {hello: 'world'});  
  socket.on('my other event', function (data) {  
    console.log(data);  
  });  
});
```

Code client

```
<script src="/socket.io/socket.io.js"></script>  
  
<script>  
  
  var socket = io.connect('http://localhost');  
  
  socket.on('actualités', function (data) {  
    console.log (data);  
    socket.emit('my other event', {my: 'data'});  
  });  
</script>
```

3.2.4. Choix de php5 :

PHP 5 est sorti en juillet 2004. Il propose un nouveau moteur, Zend Engine II, optimisé pour les nouvelles fonctionnalités que nous lui connaissons, notamment l'approche objet.

Sa popularité ne cesse d'augmenter. Sa souplesse et sa grande simplicité d'utilisation séduisent un très grand nombre de développeurs.

En revanche, exploiter l'étendue de ses possibilités nécessite, au même titre que n'importe quelle plate-forme de développement complète, de bonnes connaissances théoriques.

3.2.5. Choix de EasyRTC :

EasyRTC est un framework construit au sommet du WebRTC, un standard W3C/IETF émergent pour la communication en temps réel audio, vidéo, et de données directement entre les navigateurs web. EasyRTC comprend une bibliothèque JavaScript, coté navigateur, et un serveur javascript dans le backend, construit au sommet de Node.JS.

Une application WebRTC doit généralement faire la plupart des étapes suivantes.

- + Accédez à la caméra locale et microphone sous la forme d'un «media stream ».
- + Établir une connexion à un serveur easyRTC.
- + Initier un appel à une personne sur un autre navigateur.
- + Connecter media Stream à des balises vidéo (html5).

En utilisant le framework easyRTC, plusieurs de ces étapes peuvent être regroupées en une seule communication, ce qui va simplifier considérablement la tâche des développeurs, en cas ou le développeur web essaie de prendre en charge plusieurs plates-formes.

3.3. Implémentation

L'interface graphique s'avère sans aucun doute la partie la plus cruciale dans une application web. Elle contribue à la construction de la première impression qu'à l'internaute du système. En

effet, elle constitue un critère qui peut créer la différence entre deux applications web bien qu'elles aient les mêmes fonctionnalités.

3.3.1. Interfaces de l'application :

La figure 4.12 illustre l'interface qui s'affiche à l'internaute désirant jouir des fonctionnalités de notre composant de chat. S'il dispose déjà d'un compte, il n'a qu'à fournir ses paramètres, à savoir mot de passe et pseudo, pour accéder à son profile.

Dans le cas contraire, l'internaute est invité à s'inscrire gratuitement à notre composant. Ceci peut se faire d'une seule manière. Le mode d'inscription est présenté par la figure 4.

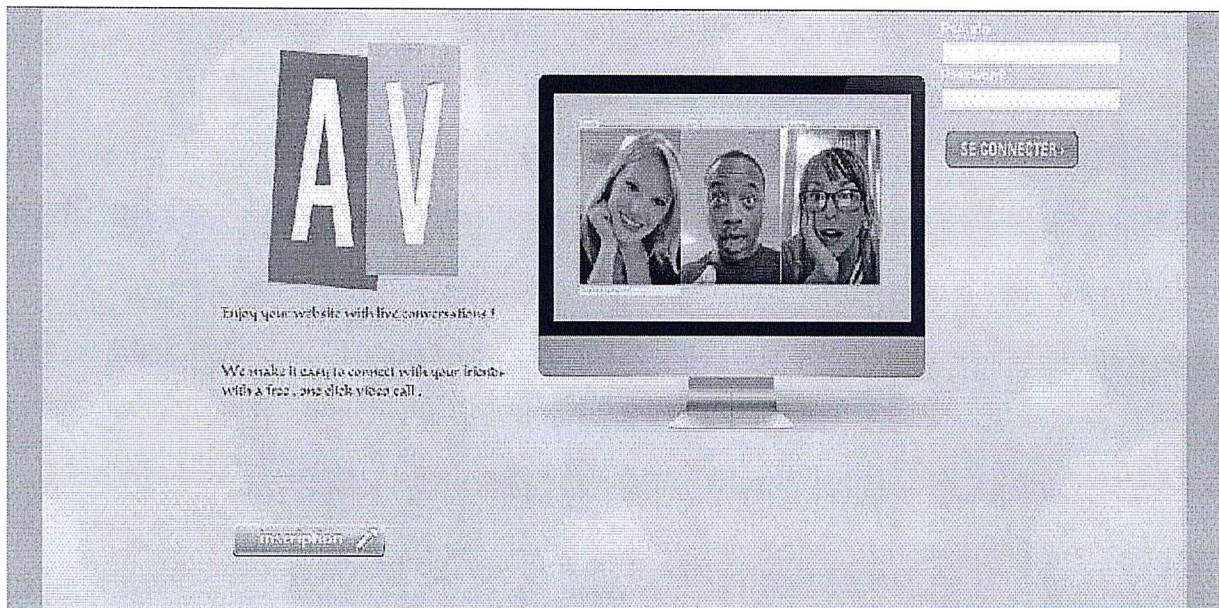


Figure 3.12 : Interface d'authentification de l'internaute

Comme le montre la figure citée, l'inscription rapide est restreinte à un nombre réduit de champs dans le formulaire d'inscription. Notre but par ceci est de faciliter l'inscription de l'internaute ainsi que lui inciter à avoir un compte dans notre réseau d'une manière simple et rapide.

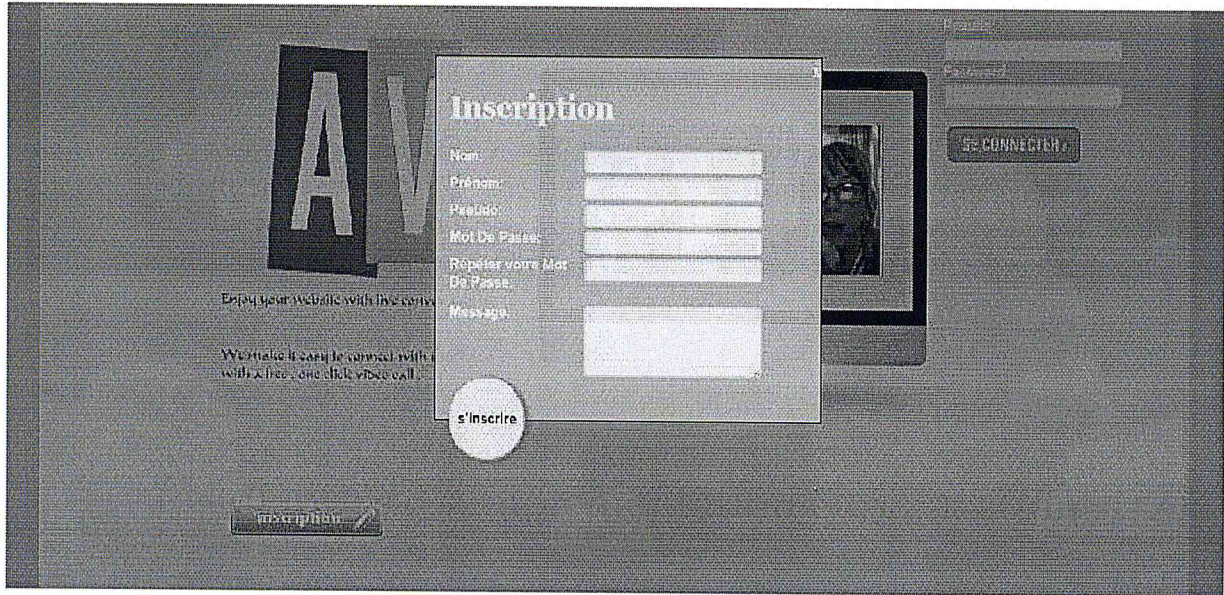


Figure 3.13 : Interface d'inscription

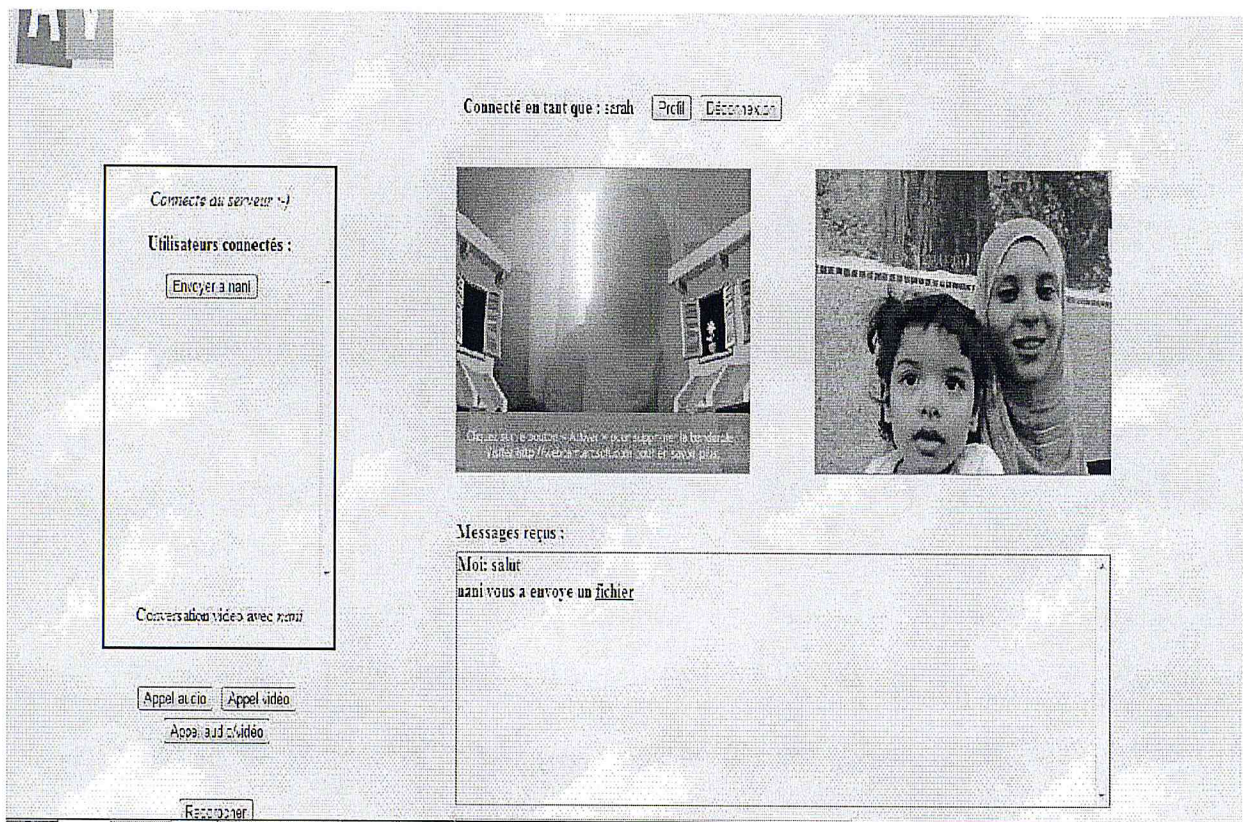


Figure 3.14 : Interface de chat privé

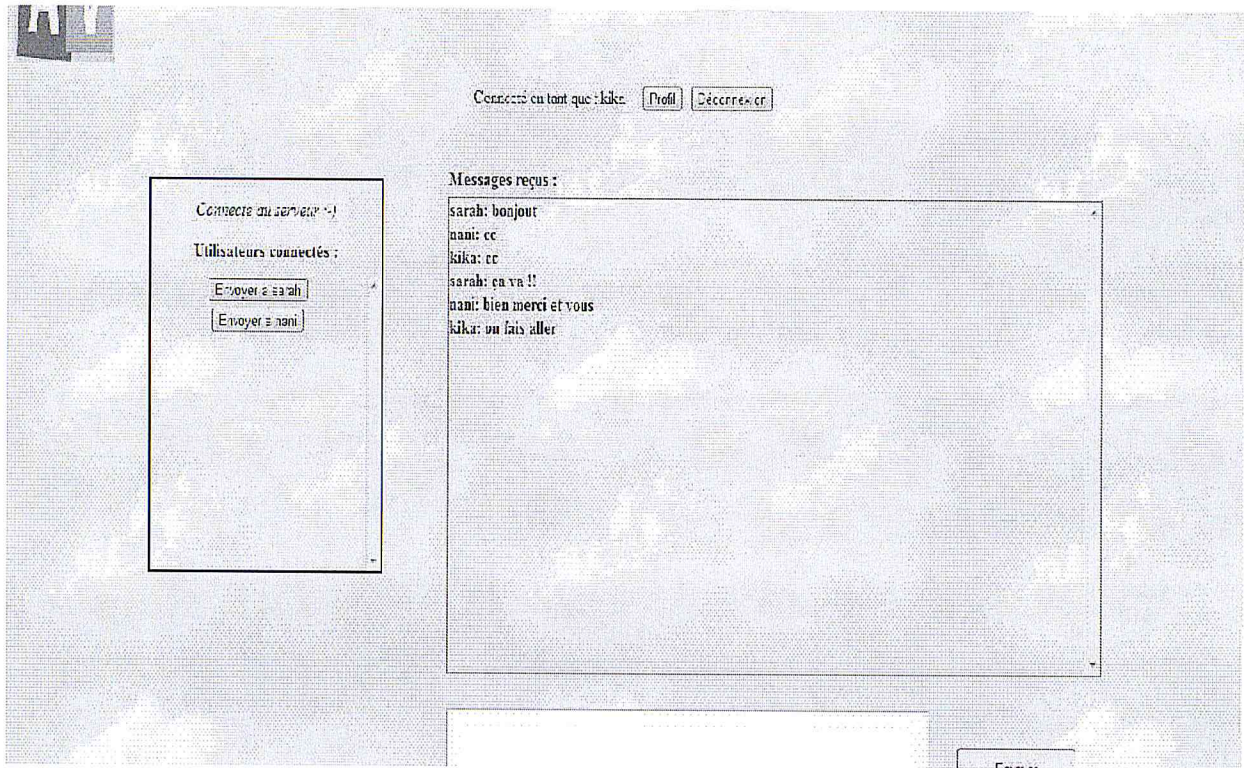


Figure 3.15 : Interface de chat public

Conclusion

Dans ce chapitre, nous avons présenté l'environnement de développement matériel et l'environnement logiciel avec lesquels ce projet a été réalisé. Nous avons présenté aussi une vue de l'application finale via quelques imprimés d'écrans ainsi que le chronogramme des tâches accomplies durant ce travail.

Conclusion générale :

Ce projet était une bonne occasion pour sortir du cadre théorique et appliquer les connaissances acquises lors des études universitaires

Sur le plan technique, ce travail nous a fait découvrir de nouveaux concepts et technologies liées à HTML5 comme les Websockets et WebRTC.

Le profit essentiel étant d'apprendre comment utiliser une technologie existante et la faire intégrer dans notre application en respectant des contraintes temporelles qui ne sont pas toujours simples à respecter.

La contrainte de temps ainsi évoquée nous a empêchés d'ajouter d'autres fonctionnalités à notre application qu'on a estimée faisables techniquement, parmi lesquels nous citons la possibilité

Dans ce rapport nous avons développé quatre chapitres. Dans le premier, nous avons présenté le CMS Joomla. Dans le deuxième chapitre nous avons étudié les différents protocoles et techniques utilisés dans la construction d'un client web de messagerie instantanée classique et la messagerie via le web ainsi que les nouvelles technologies qui vont avec. Ensuite, l'analyse fonctionnelle et non fonctionnelle ainsi que la conception ont été élaborées dans le troisième chapitre. Nous avons clôturé le rapport par la parties réalisation renfermant les principales interfaces de l'application.

Bibliographie :

- Chapitre 1 : Le CMS Joomla

- [DR.Mike.E, 2008,p.10] DR.Mike Evans, (2008), The evolution of the web – from 1.0 to web 4.0,school of system ingeneering , university of reading, 45(2), p.10
- [serjio.v, 2009, p.45] serjio vienna (2009), apprendre-joomla, 2^e éd, Masson, Paris, 3, p.45
- [Taty Sena, 2010, p.20] Taty Sena, (2010). The complete beginners guide to Joomla, 1^e éd, john j,book UK, 1, p.20
- [Taty Sena, 2010, p.30] Taty Sena, (2010). The complete beginners guide to Joomla, 1^e éd, john j,book UK, 1, p.30
- [Loïc.F, 2008, p.82] Loïc Fleischmann (2008), formation joomla !, 1e éd, CCI Lands, 1, p. 82
- [Glenn.K, Stephen.D, 1999] Glenn Krasner, Stephen Ducasse (1999), Evaluating Message Passing Control Techniques in *Smalltalk*, Journal Of Object Oriented Programming (JOOP), [en ligne],3,1 , (consulté le 12/03/2013), <http://citeseerx.ist.psu.edu/viewdoc>
- [Glenn.K, Stephen.D, 1999] Glenn Krasner, Stephen Ducasse (1999), Evaluating Message Passing Control Techniques in *Smalltalk*, Journal Of Object Oriented Programming (JOOP), [en ligne],3,1 , (consulté le 12/03/2013), <http://citeseerx.ist.psu.edu/viewdoc>
- [Salihefendic.A, 2011] Salihefendic.A, (2011), le modèe MVC de joomla, Try Joomla, [en ligne], 92(2), (consulté le 13/04/2013), <http://www.axoloth.com/index>
- [Salihefendic.A, 2011] Salihefendic.A, (2011), le modèe MVC de joomla, Try Joomla, [en ligne], 92(3), (consulté le 13/04/2013), <http://www.axoloth.com/index>
- [Adam.D, 2010,p.96] Adam D.scott (2010). Joomla ! les bases de l'administration, 3^e ed, goody, Paris, 5, p.96
- [Hasin. h, 2008, p.50] Hasin hayder (2008). Wordpress complete , 2nd ed, land, UK, 10, p.50
- [John.d, 2010, p.99] John delly, (2010). Software programming, 5th éd, john, Pearson Education,France,1, p.99
- [Mihàli.M, 2011] Mihàli Marti (2011) ,Extension of Joomla, joomla !, [en ligne], 78[3], (consulté le 10/03/2013), <http://www.lafermeduweb.com/joo/indexe>

- Chapitre 2 : Messagerie instantanée basée sur le web

- [Elmagnif. C, 2008,p.40] Elmagnif, C (2008). messagerie instantanée et interface web, Thèse d'ingénieur en réseaux, DTS-Téléinformatique, l'ESTM , dakkar, p.40
- [Elmagnif. C, 2008,p.45] Elmagnif, C(2008). messagerie instantanée et interface web, Thèse d'ingénieur en réseaux, DTS-Téléinformatique, l'ESTM , dakkar, p.45
- [ICQ, 2005] ICQ(2005), Application ICQ, [en ligne] ,(visité le 12/03/2013), www.ICQ.com
- [Rodolphe.R, Raphaël G,2012, p.45] Rodolphe Rimelé, Raphaël Goetter, (2012), HTML 5 Une référence pour le développeur web, 4^e éd, Education , France, p.45
- [Rodolphe.R, Raphaël G,2012, p.46] Rodolphe Rimelé, Raphaël Goetter, (2012), HTML 5 Une référence pour le développeur web, 4^e éd, Education , France, p.46
- [Rodolphe.R, Raphaël G,2012, p.47] Rodolphe Rimelé, Raphaël Goetter, (2012), HTML 5 Une référence pour le développeur web, 4^e éd, Education , France, p.47
- [Rodolphe.R, Raphaël G,2012, p.48] Rodolphe Rimelé, Raphaël Goetter, (2012), HTML 5 Une référence pour le développeur web, 4^e éd, Education , France, p.48
- [Rodolphe.R, Raphaël G,2012, p.49] Rodolphe Rimelé, Raphaël Goetter, (2012), HTML 5 Une référence pour le développeur web, 4^e éd, Education , France, p.49
- [Rodolphe.R, Raphaël G,2012, p.50] Rodolphe Rimelé, Raphaël Goetter, (2012), HTML 5 Une référence pour le développeur web, 4^e éd, Education , France, p.50
- [Peter. M,2012] Peter Moskovits, (10/2012), Build Living Web Applications with WebSockets, , Nante, [en ligne], (visité le 10/05/2013), <http://javateam.sodifrance.fr/?p=1050>
- [Odin .G ,2012,p.1] Odin Gremaud (publié le 27 mars 2012), communications temps-réel entre navigateurs Web, L'API WebRTC 1.0 , p.1,[en ligne], (visité le 18/04/2013), <http://www.nexcom.fr/2012/03/api-webrtc-1-0/>
- [Odin .G ,2012,p.2] Odin Gremaud (publié le 27 mars 2012), communications temps-réel entre navigateurs Web, L'API WebRTC 1.0 ,p.2, [en ligne], (visité le 18/04/2013), <http://www.nexcom.fr/2012/03/api-webrtc-1-0/>
- [Elmagnif. C, 2008,p.40] Elmagnif, C(2008). messagerie instantanée et interface web, Thèse d'ingénieur en réseaux, DTS-Téléinformatique, l'ESTM , dakkar, p.40
- [Elmagnif. C, 2008,p.23] Elmagnif, C(2008). messagerie instantanée et interface web, Thèse d'ingénieur en réseaux, DTS-Téléinformatique, l'ESTM , dakkar, p.23
- [Dino E, 2012] Dino Esposito, (publié en Mai 2012), Comprendre la puissance des WebSocket, MSDN Magazine, [en ligne], 4,(visité le 5/05/2013), <http://msdn.microsoft.com/fr->

fr/magazine/hh975342.aspx

[Hazem.T, 2007]

Hazem Torab, (publié le 1/07/2007), Comparing 16 web messengers, [en ligne],3, visité le (03/04/2013), <http://web2magazine.blogspot.com/2007/07/comparing-16-web-messengers.html>

- Chapitre 3 : Etude Conceptuel

[Juliard.F , 2002]

Juliard.F (2002), UML Unified Method Language, Journal Université de Bretagne Sud UFR SSI-IUP Vannes,60(3), p.18

- Chapitre 4 : Implémentation et Réalisation

[Mathieu .N,2013,p.10]

Mathieu Nebra alias, (publié le 3/5/2013),du javascript à node.js, Article de magazine Mateo21,2,p.10, [en ligne], (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/du-javascript-a-la-papa-a-node-js>

[Mathieu. N,2013,p.30]

Mathieu Nebra alias, (publié le 3/5/2013), Node.js :le javascript coté serveur, (visité le 30/05/2013), Article de magazine Mateo21,2,p.30, [en ligne], (visité le 30/05/2013) <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/node-js-le-javascript-cote-serveur>

[Mathieu .N,2013,p.31]

Mathieu Nebra alias, (publié le 3/5/2013),Node.js :le javascript coté serveur, Article de magazine Mateo21,2,p.31, [en ligne],(visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/node-js-le-javascript-cote-serveur>

[Mathieu.N,2013,p.24]

Mathieu Nebra alias,(publié le 3/5/2013),Pourquoi node.js est_il rapide ? , Article de magazine Mateo21,2,p.24, [en ligne] (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/pourquoi-node-js-est-il-rapide>

[Mathieu.N,2013,p.25]

Mathieu Nebra alias,(publié le 3/5/2013),Pourquoi node.js est_il rapide ? , Article de magazine Mateo21,2,p.25, [en ligne], (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/pourquoi-node-js-est-ilrapide>

[Mathieu .N,2013,p.5]

Mathieu Nebra alias,(publié le 3/5/2013), installation de node.js sous windows, Article de magazine Mateo21,2,p.5, [en ligne] , (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/installation-de-node-js-sous-windows>

[Mathieu .N,2013,p.6]

Mathieu Nebra alias,(publié le 3/5/2013), installation de node.js sous windows, Article de magazine Mateo21,2,p.6, [en ligne], (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications->

- ultra-rapides-avec-node-js/installation-de-node-js-sous-windows
- [Mathieu .N,2013,p.7] Mathieu Nebra alias,(publié le 3/5/2013), installation de node.js sous windows, Article de magazine Mateo21,2,p.7, [en ligne], (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/installation-de-node-js-sous-windows>
- [Mathieu .N,2013,p.38] Mathieu Nebra alias,(publié le 3/5/2013), des serveurs web et des threads, Article de magazine Mateo21,2,p.38, [en ligne], (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/des-serveurs-web-et-des-threads>
- [Mathieu .N,2013,p.39] Mathieu Nebra alias,(publié le 3/5/2013), des serveurs web et des threads, Article de magazine Mateo21,2,p.39, [en ligne] , (visité le 30/05/2013), <http://www.siteduzero.com/informatique/tutoriels/des-applications-ultra-rapides-avec-node-js/des-serveurs-web-et-des-threads>

