République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Saad Dahlab Blida 1

N° D'ordre :.....

Faculté des sciences

Département d'informatique

Mémoire Présenté par :

SALI Abderrahmane SAHLAOUI Sofiane

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique

Spécialité : Informatique

Option : génie des systèmes informatiques

Sujet : Réalisation d'une Plateforme Collaborative à l'aide des Technologies Web Temps Réel

Soutenu le : 26 / 06/2016

Devant le jury :

M.	DJENOURI Youcef	Président
Mme.	MANCER Yasmine	Examinateur
M.	CHIKHI Nacim Fateh	Promoteur

Promotion :

2015/2016

ملخص

من أجل تحسين أدائها، تعتمد الكثير من الشركات على أدوات لتسهيل للتعاون بين أفرادها، مما أدى الى ظهور أسلوب جديد في العمل، أين يعمل الكثير من الأشخاص باستعمال تكنولوجيات الاعلام و الاتصال، بحيث أن هذه الأخيرة تحتوي على مجموعة من الوسائل المعلوماتية التي تسمح بتطوير علاقة جديدة بين الموظفين، هذا النوع من الوسائل يدعى منصات العمل التعاوني.

و لكن المشكل الأساسي مع هذه الوسائل هو التكلفة، لأن المنصات الجيدة تتطلب دفع مبالغ مالية، أما المنصات المجانية تحتوي على خدمات أقل من المستوى.

الهدف من هذا المشروع هو تطوير منصة للعمل التعاوني على شكل تطبيق ويب، والذي يتضمن الخدمات الأساسية لمثل هذا النوع من التطبيقات، و تتمثل في أدوات الاتصال و تقاسم الموارد، و نستعمل كميدان تطبيق اعداد رسوم تخطيط يو أم أل (UML) .

يعتمد هذا التطبيق على وسائل تكنولوجية جديدة تدعى "الويب في الوقت الحقيقي" و التي تتميز بكفاءة عالية في الأداء و هي:

Node.js: من أجل تسريع تنفيذ البرامج و استعمال لغة البرمجة JavaScript من جهة الخادم.

WebSockets: من اجل التبادل الثنائي المتزامن في المعلومات بين الزبون و الخادم.

WebRTC: من اجل ارسال المعطيات مباشرة بين الزبائن.

JavaScript و HTML5 من اجل انشاء صفحات الزبائن بشكل ديناميكي.

Résumé

Afin d'améliorer leurs performances, les entreprises utilisent des outils facilitant la collaboration entre leurs acteurs. Ainsi, un nouveau mode de travail collaboratif apparait, où se joignent de nombreuses personnes au moyen de technologies de l'information et de la communication, de sorte qu'elles englobent un ensemble d'outils informatiques, qui permettent de développer des nouvelles relations entre collaborateurs d'une organisation. Ce type d'outils est appelé les plateformes de travail collaboratif.

Mais le problème essentiel est que les bonnes plateformes sont payantes, tandis que celles qui sont gratuites ont des fonctionnalités très basiques.

Le but de ce projet est de développer une plateforme de travail collaboratif sous la forme d'une application web qui assure des fonctionnalités principales liées à la communication et au partage de ressources. On utilise comme domaine d'application l'édition de diagrammes UML.

Notre solution est basée sur un ensemble de technologies web temps réel récentes et performantes qui seront nécessaire pour atteindre cet objectif :

- Node.js pour l'exécution rapide de programmes JavaScript côté serveur.
- les WebSockets pour l'échange bilatéral synchrone entre le client et le serveur.
- WebRTC pour la transmission de données en P2P entre les clients.
- JavaScript et HTML5 pour la réalisation des clients dynamiques.

travail collaboratif, plateforme collaborative, Node.js, WebSockets, WebRTC.

Abstract

In order to improve their performances, companies use some tools to facilitate the collaboration between their elements, a new work mode appeared where many people have joint using information and communication technologies, so that they encompass all technical and computer tools, which enable the development of new relations among employees of an organization.This type of tools is called collaborative work platform.

But the main problem is : the best platforms are premium, while the Free have very basic features.

The purpose of this project is to develop a collaborative work platform, as a web application form which concentrates only in the essential features, to properly treat a collaborative work, it is in particular those related to communication and broadcast resources. Is used as scope editing UML diagrams.

Our solution is based on a set of recent and efficient real-time web technologies that will be needed to achieve this goal:

- Node.js for a quick running of JavaScript programs on server side.
- WebSockets for synchronous bilateral exchange between client and server.
- WebRTC for transmitting P2P data between clients.
- JavaScript and HTML5 for the realization of dynamic customers.

collaborative work, collaborative platform, Node. js, WebSockets , WebRTC.

Remerciement

Ce mémoire n'aurait pas été réalisé sans l'intervention , d'un grand nombre de personnes, nous souhaitons ici les en remercier.

Nous tenons d'abord à remercier très chaleureusement Dr. CHIKHI Nacim Fateh, notre promoteur, pour son sujet de projet de fin d'étude qui nous a passionné, et également pour ca précieuse aide et ses conseils , la patience, la confiance qu'il nous a témoignés ont été déterminants dans la réalisation de notre travail.

Nous voudrions également remercier les jurés pour avoir accepter d'examiner et de juger notre travail. Nos remerciements s'étendent également à tous nos enseignants durant les années des études. Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragées au cours de la réalisation de ce mémoire.

Merci à tous et à toutes.

Table des matières

ملخص I
Résumé II
Abstract III
Remerciement IV
Introduction générale 11
Plateformes de travail collaboratif 13
1. Introduction 13
2. Définitions 13
2.1. La collaboration 13
2.2. Le travail collaboratif 14
3. Plateforme de travail collaboratif 15
3.1. Objectifs des plateformes de travail collaboratif 15
3.2. Types des plateformes de travail collaboratif 15
3.3. Les fonctionnalités des plateformes de travail collaboratif 16
3.4. Avantages et inconvénients des plateformes de travail collaboratif 17
4. Choix d'une plateforme de travail collaboratif 18
4.1. Modes d'accès à une plateforme collaborative 18
4.2. Critères de choix d'une plateforme de travail collaboratif 18
4.3. Outils d'une plateforme de travail collaboratif 18
4.4. Le choix d'un outil collaboratif 22
4.5. Classification des plateformes de travail collaboratif 22
5. Conclusion 24
Les Technologies Web temps réel 25
1. Introduction 25
2. JavaScript 25
3. Node.js 26
3.1. Définition 26
3.2. Enjeux de Node. js 27
3.3. Le JavaScript côté client et côté serveur 27
3.4. Performances de Node js 29
3.5. Quelques chiffres sur Node.js 29
3.6. Moteur JavaScript V8 30
3.7. Modèle bloquant vs modèle non bloquant 30

4. WebSockets 31
4.1. Limites du protocole HTTP 32
4.2. Définition 35
4.3. Le protocole WebSocket 35
4.4. Le potentiel de la technologie 36
5. WebRTC (Real-Time Communication) 37
5.1. Définition 37
5.2. Le potentiel de la technologie 37
5.3. Les APIs JavaScript de WebRTC 39
5.4. Triangle WEBRTC 41
6. Conclusion 41
Conception de la plateforme de travail collaboratif 42
1. Introduction 42
2. L'extension WAE d'UML 42
2.1. Le langage UML 42
2.2. Web Application Extension (WAE) 43
3. Démarche de développement 47
3.1. Le processus UP 47
3.2. Phases et itérations du processus UP 48
4. Conception de la plateforme collaborative 49
4.1. Diagrammes de cas d'utilisation 49
4.2. Les diagrammes de séquence système 54
4.3. Les diagrammes de classe 58
4.4. Diagrammes de séquence 61
5. Conclusion 67
Réalisation de la plateforme de travail collaboratif 68
1. Introduction 68
2. Architecture de la plateforme de travail collaboratif 68
3. Langages de développement 69
4. Environnement de développement 70
5. Implémentation 71
5.1. Côté serveur 71
5.1.1. Serveur Node.js 71
5.1.2. Socket.io 72
5.1.3. EasyRTC 73

5.1.4. Express.js 75
5.1.5. WampServer 2.5 76
5.2. Côté client : 77
5.2.1. Pages HTML et code JavaScript 77
6. Présentation de l'application 78
6.1. La page d'accueil 78
6.2. La page d'inscription 79
6.3. Page d'accueil « Utilisateur » 79
6.4. Outils : 80
7. Conclusion 86
Conclusion générale 87
Références 89

Table des figures

Figure 1 : De la tache individuelle au travail partagé 14
Figure 2 : Projet traditionnelle et Projet en mode collaboratif 14
Figure 3 : Le trèfle fonctionnel d'Ellis 19
Figure 4: Interface de bitrix24 23
Figure 5 : Évolution du JavaScript.....26
Figure 6 : Le schéma classique : PHP sur le serveur, JavaScript chez le client 28
Figure 7 : JavaScript côté serveur 28
Figure 8 : Traitement rapide de Node.js.....29
Figure 9 : Modèle non bloquant en programmation.....31
Figure 10 : Le polling.....32
Figure 11 : Le long polling.....33
Figure 12 : Streaming.....33
Figure 13 : La communication asynchrone : le client demande, le serveur répond.....34
Figure 14 : Connexion à un WebSocket.....36
Figure 15 : La pile de protocoles utilisés par WebRTC dans un échange de données.....40
Figure 16 : Etablissement d'une connexion entre deux clients utilisant WebRTC.....41
Figure 17 : La hiérarchie des diagrammes UML 2.5 sous forme d'un diagramme de classes. 43
Figure 18 : Page serveur 45
Figure 19 : Page client.....45
Figure 20 : Formulaire HTML 46
Figure 21 : Frameset[IBMKC].....46
Figure 22 : Diagramme de cas d'utilisation global. 51
Figure 23 : Diagramme de cas d'utilisation « Gérer comptes ».....51
Figure 24 : Diagramme de cas d'utilisation « Communiquer ». 52
Figure 25 : Diagramme de cas d'utilisation « Consulter messagerie».....52
Figure 26 : Diagramme de cas d'utilisation « Gérer agenda ».....53
Figure 27 : Diagramme de cas d'utilisation « Gérer fichier ».....53
Figure 28 : Diagramme de cas d'utilisation « Editer un projet UML ». 54
Figure 29 : Diagramme de séquence système « authentification ».....55
Figure 30 : Diagramme de séquence système «recherche et demande d'accès au projet». 55
Figure 31 : Diagramme de séquence système « communiquer ».....56

Figure 32 : Diagramme de séquence système « consulter messagerie ».....57
Figure 33 : Diagramme de séquence système «Editer diagramme ».....58
Figure 34 Diagramme de classe de la base de données.....59
Figure 35 Diagramme de classe de la plateforme.....61
Figure 36 : Diagramme de séquence d'authentification.....62
Figure 37: Diagramme de séquence pour un appel video.....63
Figure 38 : Diagramme de séquence pour un envoi d'un message.....64
Figure 39 : Diagramme de séquence de la recherche d'un projet.....65
Figure 40 : Diagramme de séquence d'édition de diagramme.....66
Figure 41 : Architecture de notre plateforme de travail collaborative.....69
Figure 42 : Node.js - L'interpréteur Node.js sous Windows.....70
Figure 43 : Partie du code de serveur.js qui montre l'usage des modules.....72
Figure 44: Extrait du code server.js.....72
Figure 45 Portion du code d'utilisation socket.io.....73
Figure 46 : Extraits de code EasyRTC.....74
Figure 47 : Invocation d'un appel vidéo.....75
Figure 48: Exemple de configuration de routage.....76
Figure 49 : Récupération des librairies dans une page html.....77
Figure 50 : Affichage de fenêtre de discussion.....78
Figure 51 :Page d'accueil de la plateforme.....78
Figure 52 : Page d'inscription.....79
Figure 53 : Page d'accueil utilisateur.....80
Figure 54 : Boite de réception d'un utilisateur.....81
Figure 55 :Agenda partagé d'un utilisateur.....81
Figure 56 : Espace de stockage d'un utilisateur.....82
Figure 57: Création d'un nouveau projet.....83
Figure 58: Liste de projets de l'utilisateur.....84
Figure 59: Edition d'un diagramme de classe.....85
Figure 60: Edition d'un diagramme de cas d'utilisation.....85
Figure 61: Edition d'un diagramme de séquence.....86

Liste des tableaux

Tableau 1 : Apache vs Node.js. 30

Introduction générale

Nous vivons actuellement une transformation absolue de la société ; Peter Drucker dans "au-delà du capitalisme" l'avait annoncé : « *Le facteur de production absolument décisif, ce n'est plus le capital, ni la terre, ni le travail, c'est le savoir. Le défi économique de la société postcapitaliste consistera à assurer la productivité du savoir et des travailleurs du savoir.*» [DF, 93].Nous sommes passés d'une économie de gestion des matières premières à une économie immatérielle. La nouvelle économie est celle des données numériques, de l'information et de la connaissance.

Le développement des technologies de l'information (l'Internet) à travers les systèmes de gestion de contenu (wiki, blogs) a permis à n'importe qui, n'importe quand et n'importe où d'éditer des documents web (texte, image, vidéo...) pour les diffuser sur internet. Ces technologies ont également ouvert la voie à de nouveaux outils de travail appelés plateformes de travail collaboratif qui permettent à l'entreprise d'améliorer sa productivité.

Pour gérer leurs projets, les entreprises ont de plus en plus besoin de mettre en commun des ressources avec différents acteurs. Le développement des technologies de l'information et de la communication (TIC) a permis l'émergence de nouvelles applications via Internet : audio/ vidéoconférences, téléréunions, chat électronique, agenda partagé, etc. Tous ces outils concourent au principe du travail collaboratif, pour lier l'échange d'informations, partage de documents et collaboration instantanée.

Cependant, il existe une panoplie de solutions logicielles sur le marché qui proposent un certain nombre d'outils collaboratifs qui peuvent être utilisés dans différents contextes, mais il reste bien fréquemment essentiel de spécifier la solution en fonction des besoins.

La création d'un espace dédié pour mettre à disposition les informations centralisées, l'optimisation de la gestion des connaissances, la diffusion de documents, tout cela nécessite de réaliser une plateforme de travail collaboratif qui soit performante pour offrir des services tels que la gestion de projets ou le partage des connaissances entre acteurs d'une mission.

Bien que les plateformes de travail collaboratif soient de plus en plus populaires, leur adoption par les petites entreprises ou institutions reste très limitée. En effet, la majorité des plateformes "intéressantes" sont payantes tandis que celles qui sont gratuites n'offrent que des

fonctionnalités très limitées.

Notre travail vise à développer une plateforme collaborative en utilisant des technologies récentes dites web temps réel. Ces dernières offrent en effet de nombreux avantages tels que la réduction de la charge sur le serveur d'applications, l'augmentation de la quantité de données qui peuvent être traitées par le serveur, l'établissement de communications peer 2 peer entre deux navigateurs web, etc. Ces technologies incluent entre autres la plateforme logicielle Node.js, le protocole WebSocket et l'API WebRTC.

La mise en oeuvre d'une plateforme collaborative est un travail lourd qui nécessite beaucoup de ressources. C'est pourquoi on se limitera dans le cadre de ce projet aux fonctionnalités essentielles permettant de mener à bien un travail collaboratif, en particulier celles liées à la communication (texte, audio et vidéo) et au partage de ressources.

Afin d'illustrer l'intérêt du travail réalisé, nous avons choisi comme domaine d'application l'édition de diagrammes UML.

La suite de ce mémoire est organisée comme suit :

- Le premier chapitre présente une vision globale sur les plateformes de travail collaboratif. Il décrit leurs principaux objectifs et présente leurs fonctionnalités ainsi que leurs avantages et inconvénients.
- Dans le deuxième chapitre nous présentons en détail les technologies web que nous avons utilisées pour la mise en œuvre de notre solution.
- La conception de notre système est décrite dans le troisième chapitre et enfin le dernier chapitre décrit l'implémentation de notre plateforme.

Chapitre 1

Plateformes de travail collaboratif

1. Introduction

La notion de « travail » fait partie de l'histoire de l'humanité depuis toujours. Afin d'accomplir des tâches de plus en plus complexes, il est devenu nécessaire de les répartir et de les coordoner entre les travailleurs. La collaboration est l'élément essentiel au concept de travail collaboratif dont le but est d'achever un objectif commun au sein d'un groupe.

Ces dernières années, avec l'évolution des technologies de l'information et de la communication, eventuellement internet, le travail collaboratif a acquis de nouvelles capacités. Il devient ainsi possible de travailler à plusieurs sur un même projet, à distance, de manière synchrone ou asynchrone en relation directe entre les collaborateurs sur des plateformes dédiés appelées plateformes de travail collaboratif. Ce chapitre introductif présente d'une façon générale les plateformes de travail collaboratif.

2. Définitions

2.1.La collaboration

Avant d'avancer sur ce qu'est un travail collaboratif, voyons d'abord ce que veut dire le mot collaboration. D'après le lexique pédagogique en ligne :

Une collaboration «*est un travail en commun. un travail entre plusieurs personnes qui génère la création d'une œuvre commune.* » (Le petit Robert, 1995)[1].

[1] http://www.oasisfle.com/documents/lexique.htm#C

2.2. Le travail collaboratif

La notion de travail collaboratif doit être comprise comme étant une forme d'organisation du travail, où des individus concourent ensemble à la réalisation d'objectifs communs, en dehors de toute forme de hiérarchie. Le travail collaboratif ajoute au travail coopératif les dimensions de coproduction, de concrétisation d'innovation. [LV, 04] définissent le travail collaboratif par :

«*Le travail collaboratif est une forme d'organisation délibérée, complexe, qui se construit dans l'intelligence de l'action, afin de réaliser des chaînes d'activités impliquant plusieurs acteurs opérant en réseau* ».

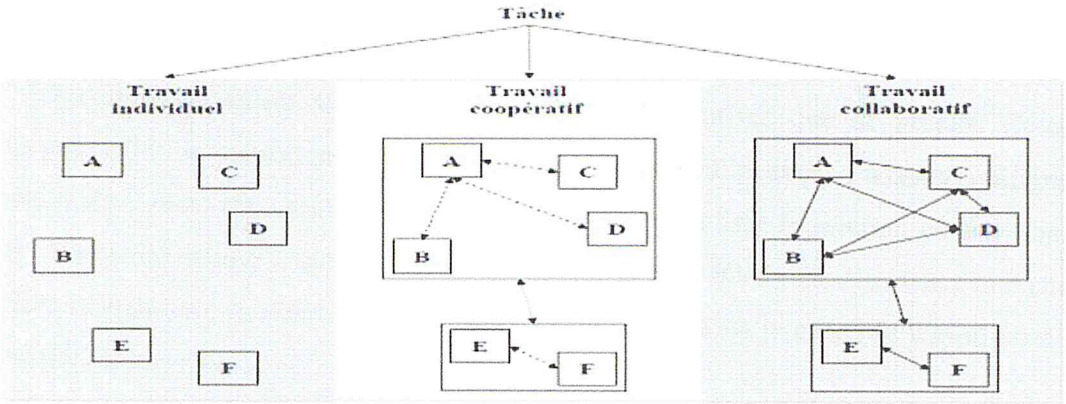

Figure 1 : De la tache individuelle au travail partagé [Ben, 11].

La Figure 1 présente la réalisation d'une tache au niveau d'un travail individuel, d'un travail coopératif et de travail collaboratif.

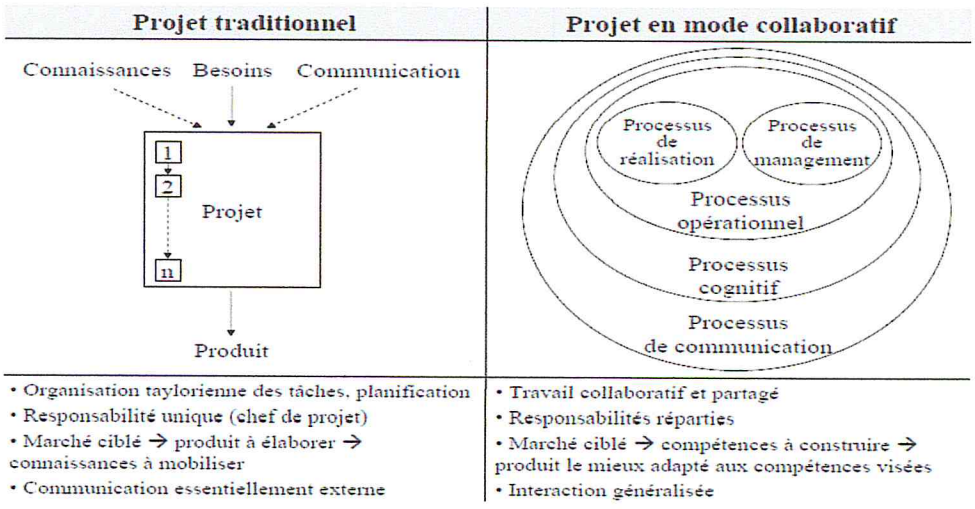

Figure 2 : Projet traditionnelle et Projet en mode collaboratif [Ben, 11].

- Les **plateformes dédiées** : généralement spécialisées sur un domaine fonctionnel spécifique, offrant un éventail très large de fonctionnalités: collaboration en temps-réel, collaboration asynchrone via des espaces de travail partagés …
- Les **plateformes « métiers » collaboratives** :comportent des applications métiers de gestion de projets, de gestion du cycle de vie des produits, de gestion de code informatique …

  Elles permettent particulièrement d'observer et d'organiser les plannings des membres du projet, d'allouer des tâches, de gérer le workflow[5], de générer des plannings, de suivre la progression du projet, etc.
- Les **plateformes généralistes :** qui résultent de la convergence entre les produits de portail, de gestion de contenu et de travail collaboratif, c'est-à-dire qu'elles intègrent des fonctionnalités de chacun des autres types d'outils.

3.3. Les fonctionnalités des plateformes de travail collaboratif

Plusieurs fonctionnalités sont offertes par les plateformes de travail collaboratif, les plus communes sont :

1. Discussions asynchrones :

- Possibilité de créer des forums de discussion ou des listes de diffusion.
- Possibilité de choisir le type de modération : aucune, limitée, restrictive, etc.
- Création de messages à partir du système ou du client email.

2. Discussions synchrones (messagerie instantanée et chat) :

- Possibilité de voir les membres présents en ligne et d'initier un dialogue.
- Exportation des échanges dans un fichier texte.
- Création de salle de chat sur des sujets spécifiques avec des membres sélectionnés.

3. Capitalisation des échanges :

- possibilité de créer une conférence web: intégration de données audio, vidéo et Powerpoint.
- partage d'application pour la co-création.

4. Notifications par emails :

[5] Workflow : la représentation d'une suite de tâches ou opérations effectuées par une personne, un groupe de personnes.

- notification automatique lors de la création de nouveaux objets.
- notification envoyée par un membre vers les autres membres.

3.4. Avantages et inconvénients des plateformes de travail collaboratif

L'utilisation des plateformes de travail collaboratif est favorisée par les sociétés au sein de ses groupes de travail, rien d'étonnant quand on constate les nombreux avantages qu'apportent aux organisations qui en possèdent.

Les avantages [6]:

• centraliser l'information et les documents en assurant un accès immédiat aux informations indexées.

• faciliter la coordination et le travail entre les équipes grâce au partage très simple de toutes les informations (document, contact, tâche, base de donnée ...).

• faciliter la communication au sein des différentes entités de l'entreprise (mail, chat, visioconférence).

• plusieurs utilisateurs peuvent se connecter en même temps et mettre à jour les informations.

Les inconvénients [7]

• Cette solution n'est pas obligatoirement adaptée à toutes les situations ; il ne suffit pas de mettre à disposition des outils logiciels pour que le travail collaboratif soit productif. La clé du succès réside bien souvent dans l'implication des membres (c'est-à-dire leur motivation à réaliser un objectif commun).

• la confidentialité et la sécurité des données : l'anonymat n'est pas garanti.

• informations ou données redondante (par exemple plusieurs fichiers de même nom)

• les plateformes non centralisés sont couteuses en termes de ressources et difficiles à administrer.

• dans les plateformes centralisées, en cas de panne du serveur, le système n'est plus opérationnel, et l'augmentation du nombre de clients implique la nécessité de deploiement de

[6] https:www.oodrive.fr/record/quels-sont-les-avantages-dune-plateforme-collaborative-basee-sur-les-contenus-medias/

[7] https://travaillerencollaboration.wordpress.com/2011/10/31/avantages-et-inconvenients-du-travail-collaboratif/

nouvelles ressources matérielles .

4. Choix d'une plateforme de travail collaboratif

4.1. Modes d'accès à une plateforme collaborative

Pour acquérir une plateforme collaborative, deux options sont disponibles :

• Certains éditeurs les commercialisent en mode SaaS [8](Software as a Service). La plateforme est alors hébergée sur des serveurs distants, et les utilisateurs y accèdent via un navigateur web. On peut donc y accéder depuis n'importe quel appareil, y compris un Smartphone. Les outils en SaaS sont généralement avantageux en termes de coût et de rapidité d'installation.

• Le mode « sur site » est le mode traditionnel : la plateforme est stockée sur le serveur du client. Les données peuvent être mieux protégées, mais le coût et le temps d'implémentation seront plus élevés.

4.2. Critères de choix d'une plateforme de travail collaboratif

Les critères de choix d'une plateforme de travail collaboratif se basent sur [Ben, 11] :

• Les fonctionnalités offertes par la plateforme.

• Sécurité offerte par la plateforme.

• Le comfort et les facilités offertes par la plateforme.

• Les coûts d'intégration de la plateforme.

4.3. Outils d'une plateforme de travail collaboratif

4.3.1. Critères d'une plateforme de travail idéale:

Avant d'avancer sur les outils, voyons d'abord les trois fonctions principales auxquelles doivent répondre les outils de travail collaboratif. Ces outils s'inspirent de la configuration du trèfle fonctionnel d'Ellis qui répond aux critères d'une plateforme de travail idéale [LDK, 05]:

• l'espace de production : il désigne les objets qui résultent d'une activité de membres parties prenantes de l'activité de groupe. L'espace de production correspond au modèle conceptuel tel que nous l'entendons en conception des systèmes mono-utilisateur.

---

[8] SaaS : un logiciel sous la forme d'un service hébergé. Tarifées par abonnement

• l'espace de communication : il fournit aux acteurs de la plateforme la possibilité d'échanger de l'information (forum, chat, newsletter, commentaires, journal en ligne par exemple)

• l'espace de coordination : il s'agit de définir les acteurs (et notamment les individus, les groupes, les rôles, voire des agents logiciels "intelligents"), d'identifier les activités et les tâches (et notamment leurs relations temporelles), de désigner enfin les acteurs responsables des tâches et des activités. Alors que l'espace de production offre une vue statique du système, l'espace de coordination en définit la dynamique.

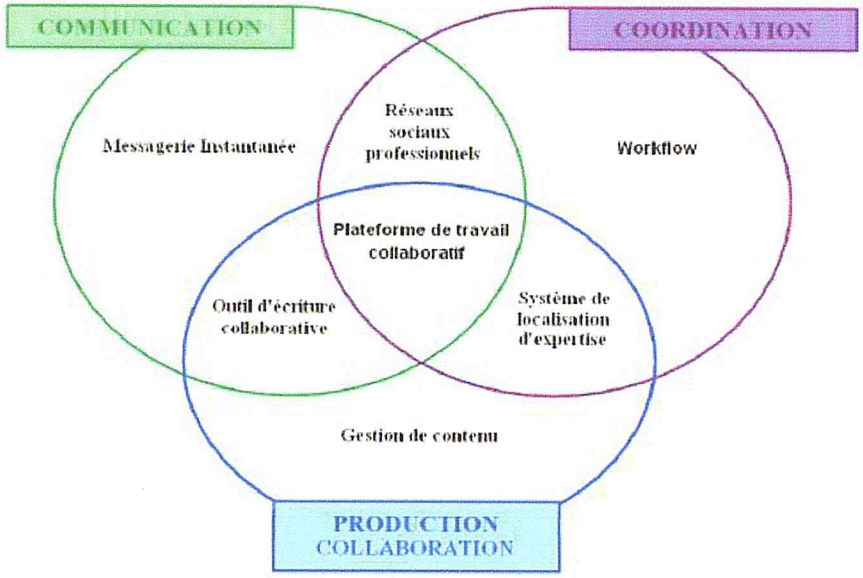

Figure 3 : Le trèfle fonctionnel d'Ellis [Bal, 05].

Le schéma dans la Figure 3 représente les trois principales fonctions des outils de travail collaboratif. En effet, selon les fonctionnalités visées d'un outil, celui- ci s'inscrira soit dans une seule et même fonction soit à l'intersection de deux ou de trois de ces fonctions.

4.3.2. Classification des outils

Les outils matériels et immatériels sont classifiés selon leurs fonctions (production, coordination, communication) et leur mode d'accès.

❖ **Par rapport aux fonctions**

- **Production** : se sont des outils qui aident à construire des produits (des documents par exemple) avec la participation de plusieurs acteurs. Exemple d'outils : outils de partage d'application, de fichiers, les wikis …

- **Communication** : se sont les outils qui permettent la circulation et l'échange d'informations entre collaborateurs. Exemple d'outils : Mails, chat, visioconférence, Portail, bibliothèque, etc.
- **Coordination** : se sont les outils qui facilitent la coordination entre les collaborateurs. Exemple d'outils : agenda partagé, workflow, etc.

❖ **Par rapport au mode d'accés :** il existe deux catégories :

• synchrones / asynchrones.

• présentiel / à distance.

4.3.2.1. Les outils synchrones

Le mode de communication est synchrone si les correspondants sont connectés en même temps et communiquent en temps réel :

La messagerie instantanée : l'avantage majeur de ce mode de communication est que les conversations sont écrites en temps réel. En effet cet outil est très utilisé, dans les domiciles comme dans les entreprises[9] . Aujourd'hui, la messagerie contient une collection d'outils multimédia (vidéos, documents écrits ...). Les services de chat privé font partie des fonctions ordinaires des platesformes mais ils sont aussi proposés aux groupes de travail par des éditeurs de logiciels spécialisés (HipChat, Campfire ... [10]).

Les outils synchrones rassemblent aussi les différents types de téléconférences : audioconférence, visioconférence, web conférence, etc. Ces conférences à distance s'affranchissent si bien des difficultés d'atteindre le lieu, bien qu'il soit possible de participer à une réunion virtuelle lorsqu'on est coincé dans un embouteillage.

4.3.2.2. Les outils asynchrones

Le mode de communication est asynchrone si les correspondants n'ont pas besoin d'être connectés en même temps pour communiquer.

• Les agendas partagés : par Internet ou en interne, permettent de fixer des dates de réunion,

[9] http://www.journaldunet.com/encyclopedie/definition/191/52/22/messagerie-instantanee.shtml

[10] HipChat, Campfire : Applications web temps réel afin de permettre les conversations privées ou en groupe, partage de fichiers et intégrations

de transmettre des messages téléphoniques, de consulter ou modifier des plannings, ... ; chacun est maître de son agenda mais accepte que celui-ci soit partiellement ou totalement accessible à d'autres personnes. Exemples d'outils : Outlook, Doodle (pour la planification des réunions), etc[11].

• Le workflow : il s'agit d'un «ensemble de logiciels pro-actifs[12] qui permettent de gérer les procédures de travail, de coordonner les charges et les ressources et de superviser le déroulement des tâches» [Sou, 94].

• Le co-travail : il regroupe les outils de partage et de stockage (Google Drive, Dropbox, Zoho Docs ...), les wikis, les éditeurs de textes collaboratifs en ligne, les blogs, etc.

- Le wiki est un site web dynamique, modifiable soit par tous, soit sur autorisation. Wiki signifie « vite », en hawaïen. Wikipédia est le plus connu d'entre eux mais il existe de nombreux wikis sur des sujets précis et même internes à certaines organisations. Les wikis d'entreprise ont des finalités variées : certains se limitent à la réservation de salles de réunions alors que d'autres ont un rôle central en recherche et développement. Lorsqu'une modification est considérée comme défavorable, il est possible de revenir à la situation précédente[13].
- Les traitements de texte en ligne (Zoho Writer, Google Documents, OpenGoo ...) permettent la révision de fichiers en mode collaboratif : chaque acteur peut modifier et annoter un document. Bien sûr, les versions successives sont sauvegardées. Mais tout est transparent : on sait qui a apporté une modification et quand. Les modifications peuvent bien sûr être refusées par les autres intervenants. Les pads[14] sont quant à eux de simples éditeurs de texte collaboratifs.

• Les logiciels de gestion de tâches et de projet : ils comportent des moteurs de recherche, des to-do lists, des dispositifs bureautiques avec fonctions de partage, la possibilité de réaliser des diagrammes de Gantt, des tableaux de répartition des tâches, des diagrammes heuristiques, etc. Exemples : Producteev, daPulse, etc.

Certains outils sont très spécialisés, soit dans leurs fonctionnalités, soit dans un domaine

---

[11] http://www.dicodunet.com/definitions/developpement/agenda-partage.htm

[12] pro-actif: signifie que ce n'est pas l'utilisateur qui invoque le logiciel mais l'inverse.

[13] http://urfist.enc.sorbonne.fr/anciensite/rss/wiki.htm

[14] Les pads : outils en ligne gratuits d'écriture collaborative

d'activité particulier : choix collaboratif de photos (Bunchcut), tableau blanc collaboratif (Twiddla), enseignement (Edmodo, Sketchlot), enseignement supérieur (Renater), etc.

• La gestion des connaissances est la collecte, la sauvegarde et la mise à disposition de connaissances et de savoir-faire. Comme indiqué par Métais et Moingeon [MMGE, 99] :

« Faire du Knowledge Management ne saurait se résumer à la gestion des systèmes d'information. Il s'agit en fait de gérer un mix comportant quatre dimensions : technologie (les systèmes d'information), stratégique (le portefeuille de connaissances), organisationnelle (une structure favorisant la création et le partage de connaissances) et identitaire (une identité apprenante). Ces quatre facettes constitutives du Learning mix, sont les composantes-clés d'un management centré sur le savoir et la connaissance. »

4.4. Le choix d'un outil collaboratif

Le Web 2.0 est la révolution des outils collaboratifs. Le terme a été popularisée par Tim O'Reilly, président-fondateur de la maison d'édition informatique O'Reilly, dans un article publié le 30 septembre 2005 qui a posé les principes[15].

Le nombre d'outils et de réseaux collaboratifs grandit chaque jour. Des outils collaboratifs open source et/ou gratuits ont ainsi vu le jour. Le choix d'un tel outil dépend cependant de plusieurs facteurs : contraintes (temporelles ou de distance), fonctionnalités recherchées, matériel disponible, ergonomie, taille du groupe, …[HLC, 01].

4.5. Classification des plateformes de travail collaboratif

Les plateformes de travail collaboratif peuvent se divisées sur plusieurs axes :

- **Par rapport aux coûts**

1- Plateformes de travail collaboratif gratuites

Elles comportent généralement des outils limités en nombre et en qualité pour usage personnel, le nombre de collaborateurs est souvent petit.

2- Plateformes de travail collaboratif payantes

a. Version professionelle

Elles offrent généralement les avantages suivants par rapport aux plateformes gratuites :

- Plus de fonctionnalités.
- Qualité des outils.

[15] http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

- Facilité d'utilisation.
- Support professionnel.
- Maintenance.
- Prestations professionnelles.

b. Version entreprise

Elles sont destinées aux entreprises et peuvent être développées sur mesure à la demande de l'entreprise, ou bien basées sur une maquettes standard déjà prête. Le nombre de collaborateurs est illimité, et les outils sont de très bonne qualité.

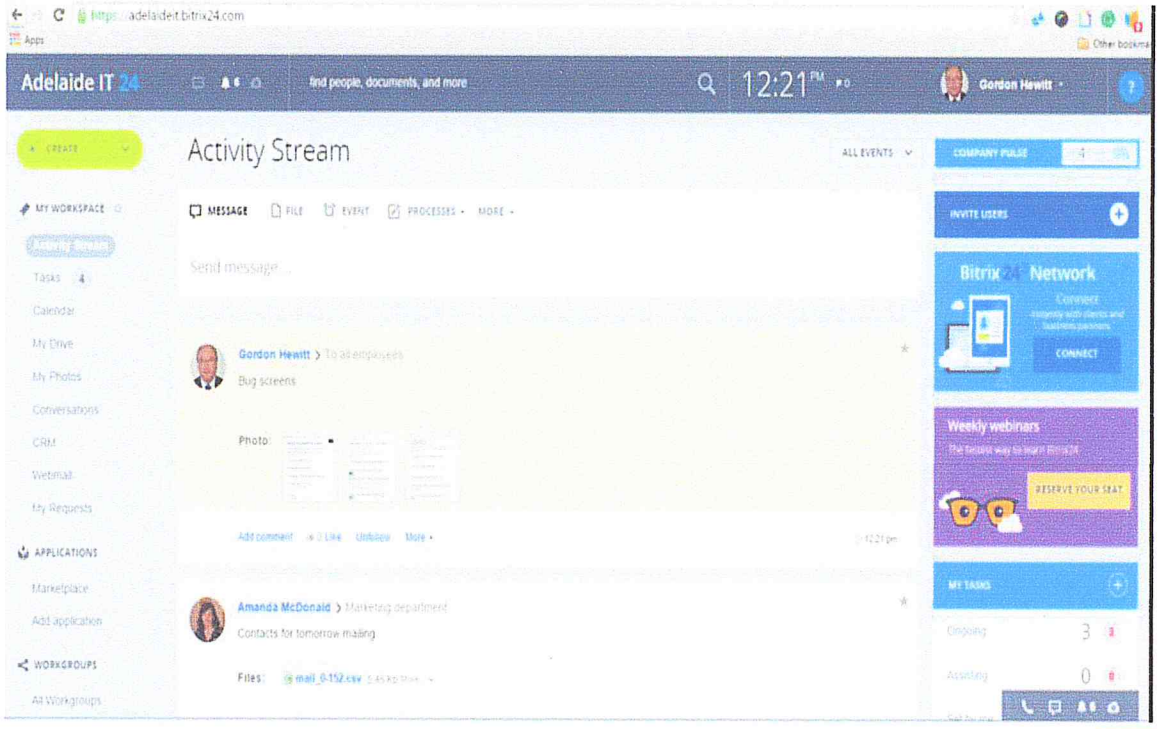

Figure 4: Interface de bitrix24 [16].

La Figure 4 montre un exemple d'une plateforme initialement utilisée par les petites entreprises où le nombre d'individus ne dépasse pas 12 membres sur le mode gratuit et illimité sur le mode professionnel. Cette plateforme comporte plus de 35 outils de travail collaboratif.

[16] https://www.bitrix24.com/

- **Par rapport aux fonctionnalités offertes**

1- Plateformes de travail collaboratif standards

Elles ont des outils standards qui répondent généralement aux besoins. Leur qualité dépend de leur coût : les bonnes sont chères, et les simplistes sont moins chères ou bien gratuites.

2- Plateformes de travail collaboratif sur mesure

Ce sont des plateformes développées sur mesure, selon les besoins, avec une interface graphique personnalisée pour une organisation professionnelle, et les outils sont conçu selon la demande du client. Ces plateformes peuvent être développées par l'équipe informatique de cette organisation, ou bien commandées chez des professionnels.

5. Conclusion

Tout au long de ce chapitre nous avons abordé l'ensemble des notions importantes liées au travail collaboratif.

Ensuite, nous avons présenté les plateformes de travail collaboratif, en décrivant leurs types ainsi que les outils et foctionnaités qu'elles peuvent proposer.

Enfin, nous avons décrit les avantages et les inconvénients des plateformes de travail collaboratif.

Avant de présenter notre application web de travail collaboratif, nous présentons dans le chapitre qui suit les technologies web temps réel que nous utiliserons afin de pallier aux inconvénients des plateformes collaboratives classiques.

Chapitre 2

Les Technologies Web temps réel

1. Introduction

L'apparition d'Internet a eu comme impact une forte croissance au niveau de la productivité des entreprises et de l'économie. Les entreprises et les informaticients ont toujours cherché à exploiter les ressources offertes par un tel réseau. La fréquence des innovations ces dernières années masque l'intérêt technologique important pour le web, le web dynamique et le web communautaire ; on passe effectivement de plus en plus aux communications en temps réel, à la collaboration en temps réel ou encore à la recherche en temps réel.

Dans ce chapitre, nous allons présenter un ensemble de technologies web qui vont servir à la réalisation de notre projet. Nous présentons ainsi les technologies Node.js, WebSocket, et WebRTC.

2. JavaScript

Avant de présenter Node.js, nous nous faison d'abord un tour du côté du langage de programmation JavaScript, en indiquant ce qu'il permet de faire, quand il peut ou doit être utilisé et comment il a considérablement évolué depuis sa création en 1995 par Brendan Eich.

JavaScript est un langage de script incorporé sous forme de texte dans un document HTML. Historiquement il s'agit même du premier langage de script pour le Web. Ce langage est un langage de programmation qui permet de fournir des améliorations au langage HTML en permettant d'exécuter des commandes du côté client, c'est-à-dire au niveau du navigateur et non du serveur web [Hon, 09].

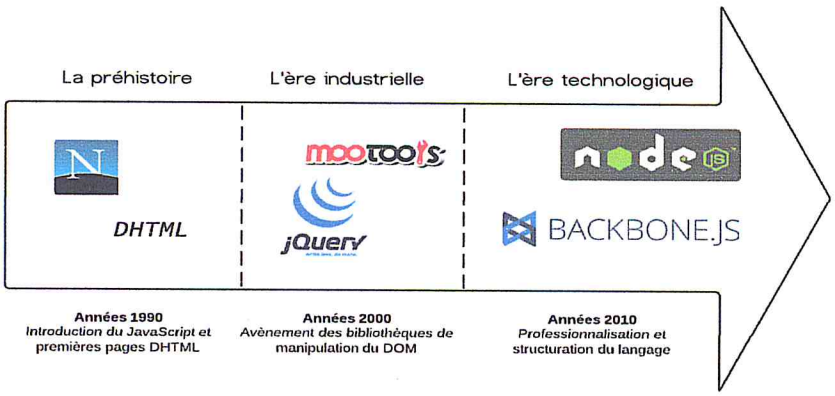

Figure 5 : Évolution du JavaScript [Neb, 13].

La Figure 5 illustre l'évolution de JavaScript depuis ca création : d'abord dans les années 90, on parlait de DHTML (Dynamic HTML). C'était l'époque de Netscape et d'Internet Explorer 5.5. Ensuite, durant les années suivantes, le langage a permis de créer des interfaces côté client. C'est là que des bibliothèques comme jQuery[1] ou MooTools[2] sont apparues. Aujourd'hui, cet usage de JavaScript est très répandu et mature. Puis depuis 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. Des outils comme Node.js sont ensuite apparus qui permettent l'utilisation de JavaScript pour le côté serveur.

3. Node.js

3.1. Définition

Tout d'abord, on donne la définition exacte de ce qui figure sur le site officiel de cette

[1] jQuery : est une bibliothèque JavaScript libre et multi-plateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web

[2] MooTools: est un framework JavaScript libre, compact, modulaire et orienté objet, dispose plusieurs classes dédiés à AJAX, aux animations graphiques ou au glisser-déposer etc

libraire conçue par Ryan Dahl en 2009 :

« *Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.*»[3].

«*As an asynchronous event driven JavaScript runtime, Node is designed to build scalable network applications.*»[4].

C'est à dire Node.js est une plateforme logicielle à base du moteur JavaScript des navigateurs chromium d'exécution événementielle et non bloquante de programme pour les applications web. Node.js est léger et efficace pour les applications qui doivent monter en charge.

Node.js permet d'exécuter du code JavaScript côté serveur.

C'est ce dernier point qui nous intéresse ; cette unification via Node.js de la logique d'une application fait que tout marche indifféremment côté client ou côté serveur,

3.2. Enjeux de Node. js

Node.js a un rôle principal en permettant l'utilisation de JavaScript sur le serveur. Il bénéficie de la puissance de JavaScript pour proposer une toute nouvelle façon de développer des applicatins web riches. Node.js offre les avantages suivants[5] :

- Performance: grâce à la puissance du moteur V8 de Google.
- Mode asynchrone, qui permet de traiter une volumétrie importante de requêtes.
- Gestion native du http.
- Multi-plateformes (mobile, desktop, tv).
- Portabilité du code JavaScript aussi bien côté serveur que client.
- Une communauté très active.

3.3. Le JavaScript côté client et côté serveur

3.3.1. JavaScript côté client

JavaScript est un langage de scripts exécuté par le navigateur côté client en effectuant des actions sur la page web localement. Cela diffère des langages de scripts qui sont exécutés par le

[3] https://nodejs.org/en/
[4] https://nodejs.org/en/about/
[5] http://www.journaldunet.com/developpeur/expert/56253/node-js---la-tendance-javascript-cote-serveur.shtml

serveur Web. C'est le cas des langages comme PHP[6].

Voici un schéma reprenant ce fonctionnement.

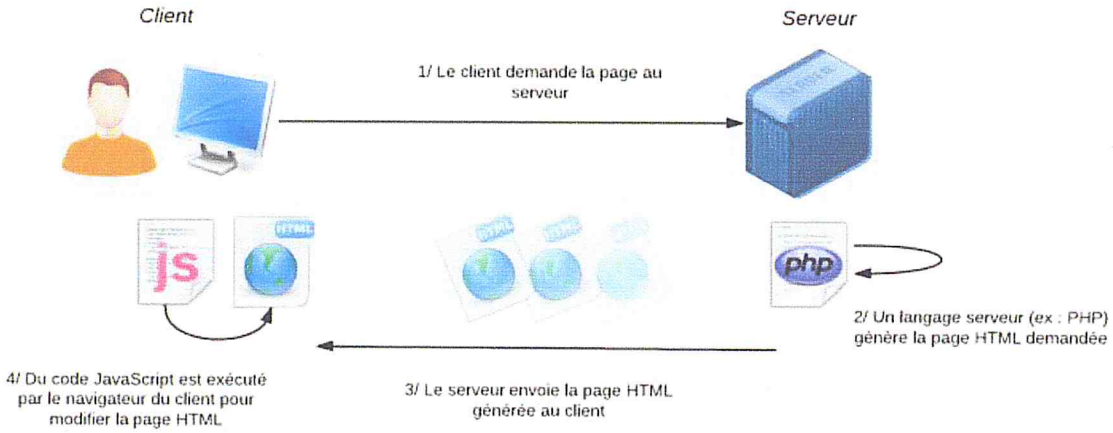

Figure 6 : Le schéma classique : PHP sur le serveur, JavaScript chez le client [Neb, 13].

La Figure 6 montre que JavaScript était utilisé seulement du côté du client ; le navigateur web du visiteur (Firefox, Chrome, IE ...) exécute le code JavaScript et effectue des actions sur la page web.

3.3.2. JavaScript côte serveur : Node.js

Node.js permet d'utiliser le langage JavaScript sur le serveur. Il permet donc de faire du JavaScript en dehors du navigateur, et c'est de là que vient sa puissance.

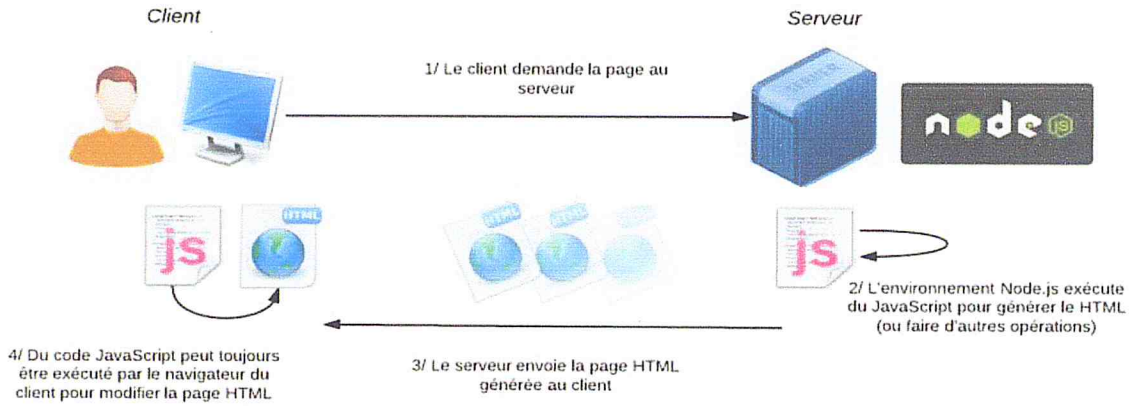

Figure 7 : JavaScript côté serveur [Neb, 13].

[6] PHP : Hypertext Preprocessor, est un langage de programmation principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP

La Figure 7 montre qu'on peut toujours utiliser du JavaScript côté client pour manipuler la page HTML. De plus, Node.js offre un environnement côté serveur qui 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.

3.4. Performances de Node js

Node.js permet une exécution rapide des traitements (voir Figure 8). En effet Node.js est asynchrone ce qui lui permet de traiter un grand nombre de requêtes en même temps. De plus, Node.js est codé en JavaScript, cela permet que le côté serveur et le côté client soient codés de la même façon. Un autre avantage est la disponibilité de différents modules. L'utilisation de ces modules permet un codage simple, efficace et rapide.

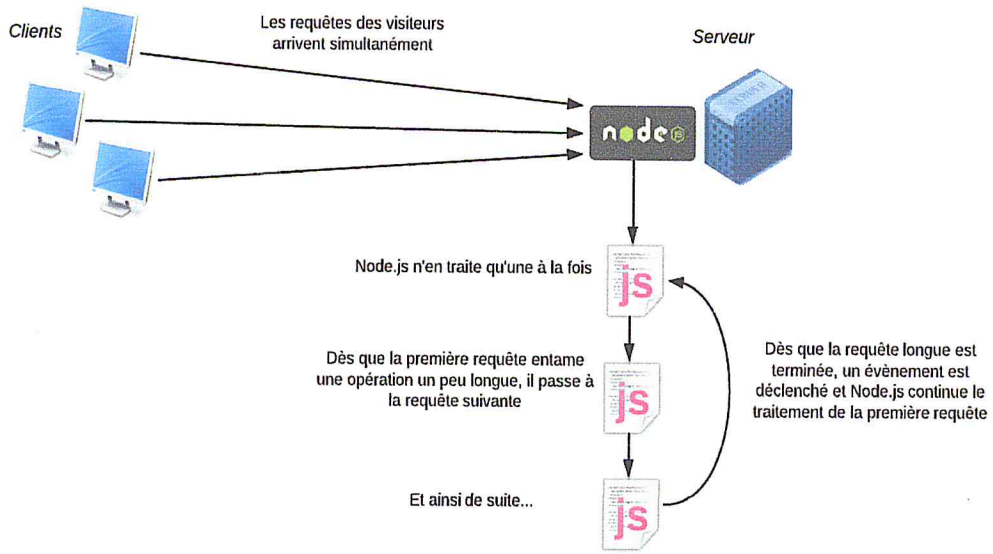

Figure 8 : Traitement rapide de Node.js [Neb, 13].

3.5. Quelques chiffres sur Node.js

Des tests de performances ont été effectués sur Node.js pour le comparer au serveur web Apache. On a ainsi mesuré le nombre de requêtes traitées par seconde des deux serveurs web en effectuant un total de 10 000 requêtes. Ces requêtes demandent simplement le chargement d'une

page web statique[7].

Plateforme	Nombre de requêtes traités / seconde
Apache (php)	3187
Node.js	5570

Tableau 1 : Apache vs Node.js.

Les résultats du Tableau 1 montrent que Node.js est 75% plus rapide sur ces types de traitements. Et en général, Node.js est plus performant qu'un serveur Apache car toutes les fonctionnalités utilisées dans Apache ne le sont pas dans Node.js. Cela tient principalement à deux choses : le moteur V8 et son fonctionnement non bloquant.

3.6. Moteur JavaScript V8

La plupart des navigateurs exécutent 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.

Depuis que Chrome a été lancé, il incluait un puissant moteur Javascript : V8, conçu pour permettre à la nouvelle génération d'applications web de s'exécuter plus rapidement dans le navigateur.

Le moteur V8 de Google Chrome, qui est réutilisé par Node.js, fonctionne de manière complètement différente. 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 ce qui montre la puissance de Node.js[8].

3.7. Modèle bloquant vs modèle non bloquant

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.

3.7.1. Modèle bloquant

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

[7] www-igm.univ-mlv.fr/~dr/XPOSE2012/**NodeJS/performances**.html
[8] https://developers.google.com/v8/

```
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. Ensuite le programme peut faire d'autres choses (effectuer d'autres actions).

3.7.2. Modèle non bloquant

Schématiquement, l'exécution du programme peut donc se représenter comme indiqué par la Figure 9:

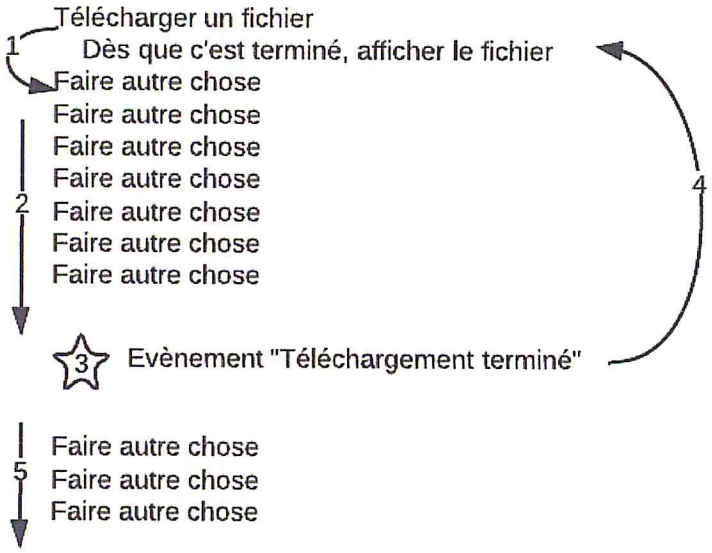

Figure 9 : Modèle non bloquant en programmation.

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

4. WebSockets

Les besoins de données en temps réel sont de plus en plus importants : bourse en ligne, jeux multi-joueurs, sites communautaires. Le principal problème vient du fait que le protocole

HTTP[9] est à la fois half-duplex (les données ne circulent que dans un sens à la fois), stateless (il est nécessaire d'ouvrir une nouvelle connexion à chaque échange) et orienté requêtes (il faut que le client demande explicitement des données au serveur). Il était donc grand temps de permettre aux applications web d'ouvrir de véritables connexions TCP[10] : C'est précisément le rôle des WebSockets.

4.1. Limites du protocole HTTP

HTTP est un protocole sans état qui fonctionne sur le modèle requête/réponse. Il a été conçu pour obtenir des éléments du web. Il répond à de nombreux besoins mais il possède plusieurs inconvénients notamment pour une utilisation dans une application web interactive :

- half duplex : le client envoie une requête au serveur qui répond en lui renvoyant une réponse. Le client doit attendre la réponse. La transmission de données ne peut se faire que dans une seule direction à la fois.
- verbeux : chaque requête et réponse HTTP doit avoir un en-tête (header) contenant plus ou moins d'informations qui fait partie des données échangées, ce qui augmente le trafic sur le réseau.
- il n'est pas possible d'utiliser un mode push de la part du serveur (le serveur ne peut pas envoyer à son initiative des données au client).

Plusieurs techniques ont été développées pour contourner ces limitations et récupérer des données fraîches dans une application web, nous citons par exemple :

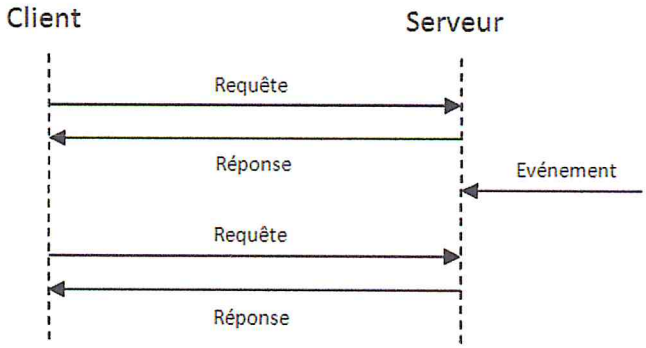

Figure 10 : Le polling.

- polling : le client effectue périodiquement des requêtes synchrones au serveur pour obtenir des

[9] http : est un protocole de communication client-serveur développé pour le World Wide Web.
[10] TCP : (protocole de contrôle de transmissions) est un protocole de transport fiable, en mode connecté

données ou pas selon qu'il y en ait de disponible. Cette technique est simple mais peu efficace car elle nécessite beaucoup de connexions selon la fréquence utilisée par le client pour obtenir potentiellement peu de données. Cette technique peut être intéressante si les données sont périodiquement modifiées côté serveur, ce qui permet de synchroniser les requêtes sur les modifications. Malheureusement ce cas de figure est plutôt rare et généralement de nombreuses requêtes sont inutiles[11]. (voir Figure 10)

- long polling : le client ouvre une connexion et envoie une requête HTTP au serveur qui ne renvoie la réponse que si un événement force l'envoi de données au client ou après un certain timeout. Le nombre de requêtes/réponses peut ainsi être réduit sauf si le nombre d'événements est très important[12].(voir Figure 11)

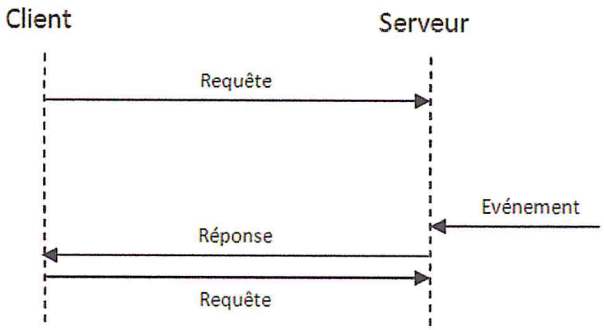

Figure 11 : Le long polling.

- Streaming : le client envoie une requête au serveur qui maintient le flux de la réponse ouvert en y envoyant des données au besoin. La durée du maintien de la réponse ouverte peut-être limitée par un timeout ou infini. Cette technique reposant sur HTTP, elle pose généralement

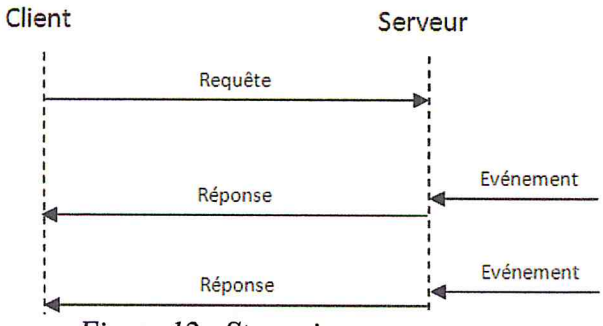

Figure 12 : Streaming.

[11] http://whatis.techtarget.com/definition/polling

[12] https://www.pubnub.com/blog/2014-12-01-http-long-polling/

des soucis avec certains éléments réseaux comme les firewalls ou les proxys [13]. (voir Figure 12)

- Server Side Event : cette technologie permet à un navigateur de recevoir des mises à jour de la part d'un serveur. Elle est supportée par la majorité des navigateurs sauf Internet Explorer. HTML5 propose de standardiser une API pour utiliser cette technologie[14].
- XMLHttpRequest : c'est un objet de programmation, utilisé dans les programmes en langage JavaScript pour assurer la communication entre le navigateur et un serveur Web. Il est utilisé pour la communication asynchrone : envoyer les requêtes vers le serveur et déclencher des opérations lors de la réception de réponses de celui-ci. L'avantage principal est dans le côté asynchrone. La page entière ne doit plus être rechargée en totalité lorsqu'une partie doit changer, ce qui entraîne un gain de temps et une meilleure interaction avec le serveur et donc le client.

Cependant, il était nécessaire de définir un standard qui permet la communication entre les clients et le serveur de manière bidirectionnelle utilisant un canal en mode full duplex. Le mode full-duplex indique qu'une WebSocket permet d'envoyer des messages du côté client et/ou du côté serveur indépendamment l'un de l'autre.

Au début du web, la communication entre client et serveur était asynchrone : le client demande et le serveur répond (voir Figure 13). Mais ces dernières années ce type de communication est devenu très contraignant car on a besoin d'une communication plus réactive et immédiate et où le serveur peut décider de lui-même d'envoyer quelque chose au client.

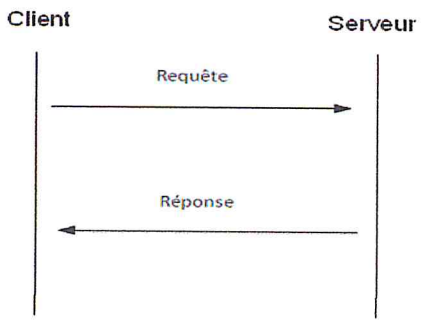

Figure 13 : La communication asynchrone : le client demande, le serveur répond.

[13] http://www.webopedia.com/TERM/S/streaming.html
[14] http://www.w3schools.com/html/html5_serversentevents.asp

WebSocket a été proposé pour cela et permet une communication ouverte entre le client et le serveur. Le navigateur et le serveur restent connectés entre eux et peuvent s'échanger des messages dans un sens comme dans l'autre dans ce canal. Maintenant le serveur peut donc lui-même décider d'envoyer un message au client.

4.2. Définition

WebSocket est apparu plus ou moins en même temps que HTML5 et est défini par :

« WebSocket est une technologie évoluée qui permet d'ouvrir un canal de communication interactif entre un navigateur (côté client) et un serveur. Avec cette API vous pouvez envoyer des messages à un serveur et recevoir ses réponses de manière événementielle sans avoir à aller consulter le serveur pour obtenir une réponse. » [15].

WebSocket désigne un protocole réseau et une interface de programmation qui est supportée par l'ensemble des navigateurs récents. Initialement développé pour HTML5, WebSocket a été normalisé par l'IETF [16] et le W3C[17]. Il permet un échange bilatéral synchrone entre le client et le serveur, c'est-à-dire qu'il permet d'établir des connexions TCP full-duplex directement depuis le navigateur. Elles pourront donc traverser les proxys et les firewalls de manière transparente.

4.3. Le protocole WebSocket

Le protocole WebSocket vise à développer un canal de communication bidirectionnel et full-duplex pour les navigateurs et les serveurs web. L'utilisation d'une WebSocket dans une page web peut se faire avec l'API JavaScript dédiée, proposée par HTML5. Ceci facilite son adoption dans les applications web. Le protocole est composé de deux phases[18]:

• Handshake : La connexion est ouverte via un premier Handshake avec un serveur compatible WebSockets sur accord réciproque et une fois les vérifications de sécurité effectuées, la connexion est ensuite promue en connexion TCP standard.

La finalité première de la phase de handshake est d'assurer l'initialisation de la communication entre le client et le serveur. Elle se compose d'un échange HTTP (une requête et

---

[15] https://developer.mozilla.org/fr/docs/WebSockets

[16] IETF : (Internet Engineering Task Force) est un groupe informel, international, ouvert à tout individu, qui participe à l'élaboration de standards Internet.

[17] W3C : (World Wide Web Consortium) est un organisme de standardisation à but non lucratif, chargé de promouvoir la compatibilité des technologies du web

[18] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers

une réponse) un peu particulier et se conclue par l'établissement de la connexion client/serveur via WebSocket.

• Data Transfer : échange de données au format texte ou binaire en mode bidirectionnel, full duplex. Le format et le contenu des données échangées entre le client et le serveur est libre: les deux parties doivent donc connaître le format utilisé pour pouvoir exploiter les données. Les données de type texte reçues d'une WebSocket sont encodées en UTF-8.

Les frames envoyées sur cette connexion peuvent être de type texte ou binaire, la différenciation se faisant grâce à un unique octet d'entête[19].

La Figure 14 illustre le principe du protocole WebSocket.

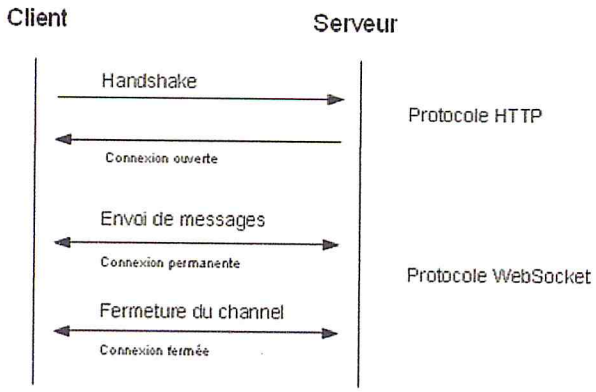

Figure 14 : Connexion à un WebSocket.

4.4. Le potentiel de la technologie

Les WebSockets sont efficaces et performantes car :

• elles requièrent moins de bande passante puisqu'elles ne requièrent pas d'en-tête dans chaque message.

• la latence est réduite.

• elles permettent de mettre en place des solutions quasi temps réelles pour recevoir des données.

• la notification au client d'un changement d'état du serveur.

• l'envoi de données en mode « push » du serveur vers le client (sans que ce dernier ait à effectuer une requête, contrairement à Ajax).

[19] https://www.pubnub.com/blog/2013-09-11-what-are-websockets/

5. WebRTC (Real-Time Communication)

Les échanges directs entre navigateurs ne sont pas une nouveauté mais les recherches et les implémentations précédentes nécessitent l'installation de plugins propriétaires tels que Adobe Flash ou Microsoft AcriveX. Ces plugins (ou protocoles) furent à la source de difficultés de s'adapter et de mise à jour pour les sites utilisant ces systèmes.

Cette approche pose plusieurs problèmes :

• De nombreux web services utilisent déjà ce genre de technologie mais nécessitent de télécharger une application ou un plugin. C'est le cas de Skype, Facebook et Google Hangouts (qui utilise le plugin Google Talk).

• Télécharger, installer et mettre à jour ses plugins peut être compliqué, générateur d'erreurs et ennuyeux.

• Les plugins peuvent être difficiles à déployer, débugger, corriger, tester et maintenir.

Cependant, l'arrivée récemment d'une nouvelle technologie permettant les communications directes entre navigateurs a changé la donne : Il s'agit de la technologie WebRTC qui est un standard du W3C et est soutenue par les grands acteurs du web tel que Google, Mozilla et Opera, De plus, Android et iOS soutiennent également l'application..

5.1. Définition

Tout d'abord, donnons la définition exacte de WebRTC qui figure sur le site officiel de cette technologie :

«*WebRTC is a free, open project that enables web browsers and mobiles with Real-Time Communications (RTC) capabilities via simple APIs.*»[20].

C'est à dire que WebRTC, littéralement (communication web en temps réel) est un projet open source et gratuit qui permet aux différents navigateurs web et Smartphone de communiquer en temps réel via des APIs simples.

5.2. Le potentiel de la technologie [21]

WebRTC a pour objectif d'assurer une communication en temps réel de haute qualité,

[20] http://webrtc.org

[21] http://www.itpro.fr/a/pourquoi-faut-il-sinteresser-webrtc-en-2016/

c'est à dire que la technologie va permettre de lier des applications comme la voix sur IP via des flux audio/vidéo bidirectionnels et permettre à n'importe quel navigateur d'avoir la possibilité de partager les données d'application entre les clients de navigateur, sans la nécessité d'installer des plugins ou logiciels propriétaires tiers jusqu'alors nécessaires[22] .

Dans une étude de marché réalisée par Webtorials, 90% des professionnels de l'IT estiment que WebRTC a la capacité d'améliorer la performance des centres de contact et 67% le voient comme une solution potentielle pour les communications vidéo externes.[23]

Ces chiffres sont susceptibles d'augmenter au fur et à mesure que les professionnels de l'IT se familiarisent avec cette technologie et qu'ils trouvent de nouvelles façons de la déployer. Voici un aperçu de certains de ses avantages potentiels :

❖ Pour les utilisateurs terminaux :

1. **Simplicité d'utilisation :** pas besoin des application ou des plugins.
2. **Sécurité :** WebRTC a plusieurs caractéristiques pour gérer la securité:

- Il utilise des protocoles sécurisés tels que DTLS[24] et SRTP[25].
- Le chiffrement est obligatoire pour tous les composants WebRTC.
- WebRTC n'est pas un plugin: ses composants s'exécutent dans le sandbox du navigateur et non dans un processus séparé.
- L'accès au microphone et au caméra doit être accordé de façon explicite et, lorsque la caméra ou le microphone sont en cours d'exécution, cela est clairement indiqué.

❖ Pour les entreprises :

1. **Une réduction des coûts**

Le WebRTC réduit la nécessité d'une infrastructure de communication traditionnelle, et il est aussi plus facile à déployer.

---

[22] https://tokbox.com/about-webrtc

[23] http://www.webtorials.com/main/resource/papers/webtorials/2015-WebRTC/2015_WebRTC.pdf?l=WT-15-07-22-SN

[24] DTLS : Datagram Transport Layer Security, fournit une sécurisation des échanges basés sur des protocoles en mode datagramme (Couche session).

[25] SRTP : Secure Real-time Transport Protocol, définit un profil de RTP (Real-time Transport Protocol) qui a pour but d'apporter le chiffrement, l'authentification et l'intégrité des messages et la protection contre le rejeu (replay) de données RTP

2. Une collaboration instantanée

Le WebRTC permet aux entreprises de déployer rapidement une solution simple de vidéoconférence de haute qualité, quel que soit l'endroit où travaille l'employé. Les équipes peuvent collaborer spontanément, et le WebRTC facilite la connexion avec des participants externes..

3. Une meilleure assistance pour les travailleurs mobiles

Les travailleurs mobiles sont peut-être parmi les plus grands bénéficiaires du WebRTC, ce qui donne aux entreprises la possibilité de proposer un large éventail d'appareils. Puisque l'audio, la vidéo et le partage de données sont lancés directement dans le navigateur, il n'est plus nécessaire d'imposer aux employés un équipement particulier.

5.3. Les APIs JavaScript de WebRTC

WebRTC permet de développer des applications de communication en temps réel et de très bonne qualité.

WebRTC utilise plusieurs APIs JavaScript [26] :

1. MediaStream (getUserMedia) : donne l'accès aux ressources media (caméra et microphone).
2. RTCPeerconnection : permet d'établir les appels audio et vidéo.
3. RTCDatachannel : permet de transférer des données en pair à pair (texte, fichier, etc).

- **getUserMedia**

Elle représente la source unique de synchronisation audio ou vidéo. Par exemple, dans l'ordinateur individuel, il existe de nombreux dispositifs externes. Nous allons donc utiliser la méthode getUserMedia() en JavaScript pour accéder à ces dispositifs de médias locaux.

- **RTCPeerConnection**

L'API PeerConnection gère pratiquement tout : la négociation de SDP[27], l'envoi des flux médias et la réception, la gestion des problèmes de réseau tels que les pertes de paquets, les

[26] http://www.nexcom.fr/2012/03/api-webrtc-1-0/

[27] SDP : (Session Description Protocol) est un protocole de communication de description de paramètres d'initialisation d'une session de diffusion en flux (streaming).

implémentations de codec, la traversée des NAT, etc. L'API enveloppe le tout dans un paquet simple qui permet d'ajouter les flux multimédia dans un seul objet PeerConnexion.

- **RTCDataChannel**

Elle est similaire à WebSocket de JavaScript. Elle est utilisée pour envoyer des données non-médias à travers le réseau pour échanger des données. Au sein d'un canal de données, les applications peuvent transmettre des messages de façon ordonnée ou désordonnée. Un flux de données est créé lorsque l'un des pairs appelle la méthode CreateDataChannel() pour la première fois après avoir créé un objet PeerConnection. Chaque appel suivant à CreateDataChannel() créera un nouveau flux de données au sein de la connexion SCTP existante.

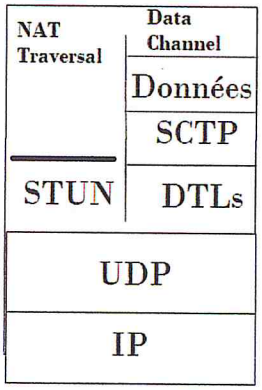

Figure 15 : La pile de protocoles utilisés par WebRTC dans un échange de données.[28]

La Figure 15 représente les composants de WebRTC qui permettent l'échange de données telles que des images ou du texte. Ces données autres que les flux médias sont échangées via le protocole SCTP[29] qui supporte nativement plusieurs flux de données de façon bidirectionnelle ; SCTP est à son tour encapsulé dans DTLS. Cette solution permet au flux de données d'être intégré dans le même paquet que les flux de médias et donc de partager le même numéro de port pour les échanges.

[28] http ://chimera.labs.oreilly.com/books/1230000000545/ch18.html
[29] SCTP : Stream Control Transmission Protocol, protocole de transport équivalent dans un certain sens au TCP ou à l'UDP (Couche transport)

5.4. Triangle WEBRTC

Le WebRTC comprend une construction triangulaire qui implique un serveur et deux clients.

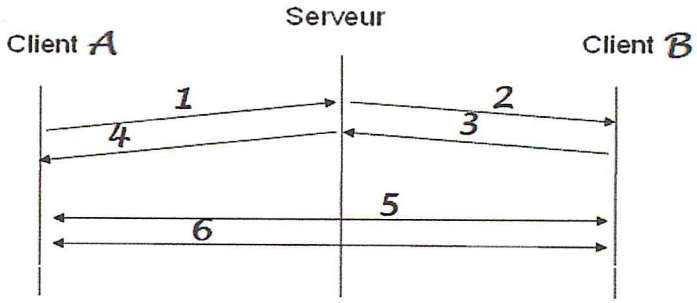

Figure 16 : Etablissement d'une connexion entre deux clients utilisant WebRTC.

La Figure 16 est un petit schéma qui montre établissement d'une connexion entre deux clients utilisant WebRTC. A et B exécutent une application HTML5 à partir d'un navigateur web [JB, 12] :

- 1: A demande au serveur une connexion avec B.
- 2: Le serveur relaie la demande de A à B.
- 3: Si B accepte, il envoie une demande de connexion à A.
- 4: Le serveur relaie la demande à A.
- 5 et 6 : Les Peer Connexion bidirectionnelles sont établies.

6. Conclusion

Dans ce chapitre, nous avons présenté la technologie « Web Temps Réel » ayant donné récemment naissance à des innovations permettant la communication en temps réel dans le Web dynamique grâce à un ensemble de technologies web temps réel. Nous avons également présenté les API nécessaires à la mise en œuvre de notre application web.

Chapitre 3

Conception de la plateforme de travail collaboratif

1. Introduction

Avant de se lancer dans la programmation, nous devons dans un premier temps déterminer l'ensemble des fonctionnalités de notre application et aussi étudier les liens entre celles-ci.

Une fois les fonctionnalités déterminées, nous décrivons l'étude conceptuelle de notre application Web en utilisant l'extension Web Application Extension (WAE) du langage UML qui permet de prendre en compte les spécificités des applications web.

Dans ce chapitre, nous commençons par définir l'extension pour les applications web (WAE) du langage UML, ensuite nous détaillons les différents diagrammes qui modélisent notre application web (diagrammes de cas d'utilisation, diagrammes de séquence et diagrammes de classe) ainsi que les tables de notre base de données.

2. L'extension WAE d'UML

2.1. Le langage UML

Face à la diversité des formalismes utilisés par les méthodes d'analyse et de conception objet, UML (Unified Modeling Language : langage unifié pour la modélisation) permet de modéliser une application selon une vision objet, indépendamment du langage de programmation.

UML se définit comme un langage de modélisation graphique, il fournit les fondements pour spécifier, construire, visualiser et décrire les objets d'un système logiciel, il permet de

modéliser de manière claire et précise la structure et le comportement d'un système[1]. UML est fondé sur des concepts orientés objets, il propose de décrire un système à l'aide de quatorze types de diagrammes regroupé en deux grandes classes : des diagrammes structurels et des diagrammes comportementaux[2].

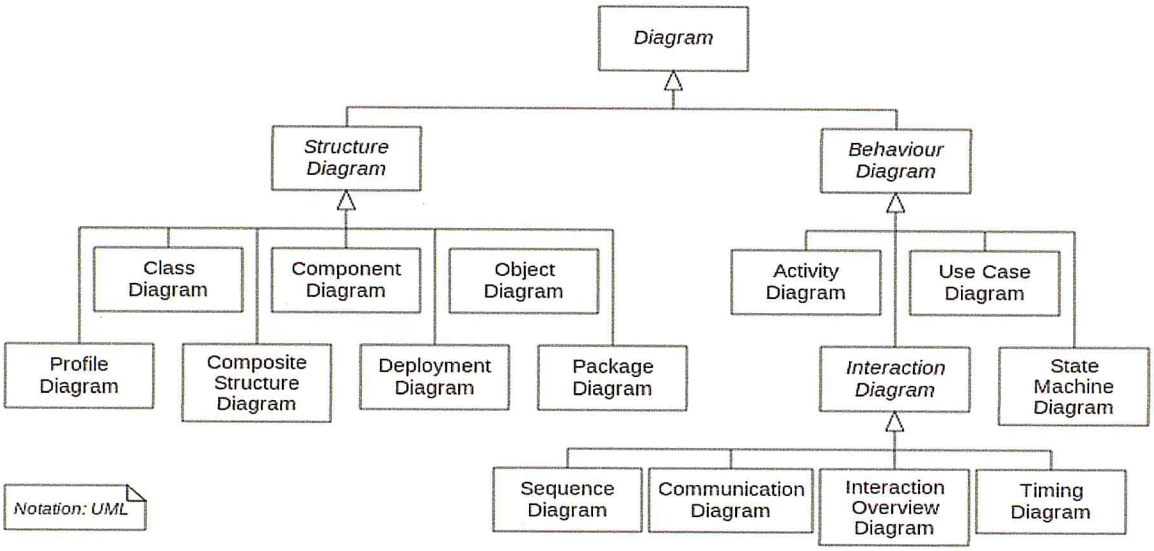

Figure 17 : La hiérarchie des diagrammes UML 2.5 sous forme d'un diagramme de classes[3].

La Figure 17 illustre les quatorze types de diagrammes UML qui sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie.

2.2. Web Application Extension (WAE)

Avec le temps, UML a connu de nombreuses améliorations et extensions. Son niveau d'abstraction ayant augmenté par la même occasion, il est parfaitement possible de l'utiliser dans d'autres domaines, comme l'ingénierie système, l'architecture SI, la modélisation de bases de données, etc. Mais ceci uniquement parce qu'au final, ces domaines sont intrinsèquement liés aux réflexions amenées par le développement logiciel, dont découlent les diagrammes UML : déploiement, mise en persistance, SOA, gestion de projet, etc.

[1] http://www.omg.org/cgi-bin/doc?formal/2005-04-01.pdf
[2] http://www.omg.org/cgi-bin/doc?formal/15-03-01.pdf
[3] https://en.wikipedia.org/wiki/Unified_Modeling_Language

Ainsi, pour que UML puisse parfaitement s'adapter à ces autres domaines, il est indispensable de passer par des mécanismes d'extension. Ainsi l'extension WAE d'UML a été proposée pour la modélisation d'applications Web.

2.2.1. Définition

L'extension d'UML pour le web (Web Application Extension) permet la représentation des pages Web et d'autres éléments importants architecturalement aux côtés des classes «normales» et ce afin d'exprimer avec précision l'ensemble du système dans un modèle et de maintenir sa traçabilité et son intégrité.

Cette extension UML est exprimée par un ensemble de stéréotypes, de valeurs marquées, et de contraintes. Combinés, ces mécanismes permettent d'étendre la notation UML et de créer de nouveaux types de blocs de construction [Con, 02].

- **Stéréotype :** Un stéréotype est une extension du vocabulaire d'UML. Il permet l'association d'une nouvelle signification à un élément du modèle. Il est représenté par une chaine de caractères entre guillemets (« »). Il peut également être représenté par une nouvelle icône.

- **Valeur marquée :** c'est une extension d'une propriété d'un élément du modèle est la définition d'une nouvelle propriété qui peut être associée à un élément du modèle. La plupart des éléments du modèle ont des propriétés qui leurs sont associées. Par exemple les classes ont des noms, la visibilité, la persistance, et d'autres attributs. Une valeur marquée est rendue sur un diagramme sous forme de chaîne fermée par des parenthèses.

- **Contrainte :** c'est une extension de la sémantique de la langue qui précise les conditions dans lesquelles le modèle peut être considéré comme bien formé. Une contrainte est une règle qui définit la façon dont le modèle peut être mis en place. Les contraintes sont rendues sous forme de chaînes entre une paire d'accolades [Con, 02].

2.2.2. Les différents types de stéréotype

Les stéréotypes offerts par WAE concernent les classes, les associations et les attributs.

Ils sont définis dans ce qui suit [Con, 02] :

a. Les classes :

• **«Page serveur » :** représente une page web dynamique qui possède des scripts exécutés par le serveur. Ces scripts interagissent avec des ressources du serveur, telles que les bases de données.

Les opérations de l'objet représentent les fonctions dans le script et ses attributs représentent les variables qui sont accessibles par les fonctions de la page. La Figure 18 illustre l'icône utilisée pour représenter une telle classe.

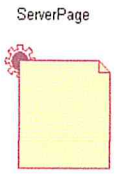

Figure 18 : Page serveur [IBMKC].

• **«Page client » :** représente une page web formatée en HTML : un mélange de données, de représentation et même de logique. Les fonctions d'une page client correspondent aux fonctions dans le script et ses attributs représentent les variables qui sont accessibles par les fonctions de la page. La Figure 19 illustre l'icône utilisée pour représenter une telle classe.

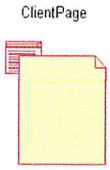

Figure 19 : Page client [IBMKC].

• **« Formulaire HTML» :** est un ensemble de champs de saisie faisant partie d'une page client. La classe formulaire correspond à une balise HTML « Form », ses attributs sont les éléments de saisie telle qu'une zone de saisie, une zone de texte, un bouton d'option...

Un formulaire n'a pas d'opérations puisqu'il ne peut pas les encapsuler. Toute opération qui interagit avec le formulaire appartient à la page qui le contient. La Figure 20 illustre l'icône utilisée pour représenter une telle classe.

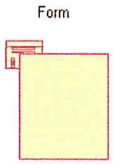

Figure 20 : Formulaire HTML [IBMKC].

• **« Framset » :** est un conteneur de plusieurs pages web. La zone d'affichage rectangulaire est divisée en cadres. Une classe stéréotype « Framset » est associée à une structure de cadre de page web par la balise HTML <framset>. La Figure 21 illustre l'icône utilisée pour représenter une telle classe.

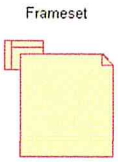

Figure 21 : Frameset [IBMKC].

b. Les associations :

• **« link » :** c'est une relation entre une page client et une autre page client ou une ressource du serveur. La cible peut être une classe «client Page» ou bien une classe «Server Page», Elle représente un pointeur entre ces pages. Elle représente aussi une abstraction de l'élément d'ancrage HTML.

• **« submit » :** c'est une relation directionnelle entre un formulaire et une page serveur. Les valeurs des champs du formulaire sont envoyées au serveur qui les traite par l'intermédiaire de pages serveur.

• **« build » :** c'est une relation directionnelle entre les pages client et les pages serveur. Elle indique quelle page serveur est responsable de la création de la page client. Une page serveur peut construire plusieurs pages client, mais une page client n'est construite que par une et une seule page serveur.

• « **redirect** » : c'est une relation directionnelle qui relie deux pages client ou serveur.

• « **include** » : c'est une association directionnelle à partir d'une page serveur à une autre page serveur ou client ; elle indique que la page incluse est traitée.

• **« forward »** : Une relation directionnelle entre une page serveur et une autre page serveur ou client. Cette association représente la délégation de traiter la demande d'un client pour une ressource à une autre page côté serveur.

c. Les attributs :

• **«input» :** correspond à la balise <input> d'un formulaire HTML. Cet attribut est utilisé pour un mot ou une zone de texte.

• **«select element» :** permet à l'utilisateur de sélectionner une ou plusieurs valeurs dans une liste d'options.

• **«textarea element» :** correspond à la balise HTML <textarea>, permet à l'utilisateur de saisir un texte sur plusieurs lignes.

3. Démarche de développement

La démarche que nous avons choisie de suivre pour le développement de notre plateforme est le processus **UP (Unified Process)** qui est un cadre général et complet de développement.

3.1. Le processus UP

Le processus unifié est un modèle de développement générique conçu autour du langage UML. Etant générique, ce modèle permet d'adapter le processus à n'importe quel projet.

Avec ce modèle la conception du logiciel s'effectue de manière itérative et incrémentale. Le processus est développé en phases, chaque phase se déclinant en itérations [Dug, 05].

Le processus de développement UP associé à UML, s'appuie sur les principes suivants [GG, 08] :

• **Processus guidé par les cas d'utilisation :** Le système à construire se définit d'abord avec les utilisateurs. Les cas d'utilisation permettent d'exprimer les interactions du système avec les utilisateurs, donc de capturer leurs besoins.

• **Processus itératif et incrémental :** Le développement progressif, par incrément, est recommandé en s'appuyant sur la décomposition du système en cas d'utilisation.

• Les avantages du développement itératif se caractérisent comme suit :

- les risques sont évalués et traités au fur et à mesure des itérations,
- les premières itérations permettent d'avoir un feed-back des utilisateurs,
- les tests et l'intégration se font de manière continue,
- les avancées sont évaluées au fur et à mesure de l'implémentation.

• **Processus centré sur l'architecture :** Il est important de définir le plus tôt possible, même à grandes mailles, l'architecture type qui sera retenue pour le développement, l'implémentation et ensuite le déploiement du système.

• **Processus orienté par la réduction des risques :** L'analyse des risques doit être présente à tous les stades de développement d'un système. Il est important de bien évaluer les risques des développements afin d'aider à la bonne prise de décision.

3.2. Phases et itérations du processus UP

Le processus UP répète un certain nombre de fois une série de cycles qui s'articulent sur quatre phases [GG, 08] :

• **Lancement :** Cette phase correspond à l'initialisation du projet où l'on mène une étude d'opportunité et de faisabilité du système à construire. Une évaluation des risques est aussi réalisée dès cette phase.

• **Élaboration :** Cette phase a pour but d'analyser le domaine technique du système à développer afin d'aboutir à une architecture stable.

• **Construction :** Elle est centrée sur les activités de conception, d'implémentation et de test.

• **Transition :** il s'agit dans cette phase de livrer le produit pour une exploitation réelle. C'est ainsi que toutes les actions liées au déploiement sont traitées dans cette phase.

Itérations : Une phase peut-être divisée en itérations. Une itération est un circuit complet de développement d'un produit exécutable. Ce produit est un sous-ensemble du produit final en cours de développement, qui croît d'itération en itération pour devenir le système final.

4. Conception de la plateforme collaborative

Les plateformes de travail collaboratif sont de plus en plus populaires parce qu'elles facilitent et optimisent la communication entre les individus dans le cadre d'un projet ou d'une tâche donnés.

Dans le cadre de ce travail, nous allons développer une plateforme de travail collaborative en utilisant des technologies récentes dites web temps réel. On se limitera aux fonctionnalités essentielles permettant de mener à bien un travail collaboratif, en particulier celles liées à la communication (texte, audio et vidéo) et au partage de ressources.

Afin d'illustrer l'intérêt du travail réalisé, nous utiliserons comme domaine d'application l'édition de diagrammes UML (diagrammes des cas d'utilisation, diagrammes de classes et diagrammes de séquences) en temps réel ce qui facilite la modélisation avec les collaborateurs.

Dans ce qui suit nous présentons la partie conceptuelle de notre « plateforme collaborative ». Nous commençons par les diagrammes de cas d'utilisation qui représentent les fonctionnalités du système, suivis par des diagrammes de séquence système. On s'intéresse par la suite à l'aspect structurel avec les diagrammes de classe, et à l'aspect dynamique avec les diagrammes de séquence.

4.1. Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation permet une description du système à construire en prenant le point de vue de l'utilisateur, les deux concepts de base de ce diagramme sont l'acteur et le cas d'utilisation.

4.1.1. Les acteurs

Un acteur représente un ensemble cohérent de rôles joués par des entités externes (utilisateur humain, matériel ou autre système) qui interagissent directement avec le système étudié par échange de messages et pouvant consulter ou modifier directement l'état du système, il existe

deux catégories d'acteurs :

Les acteurs principaux : ce sont des entités qui sollicitent le système pour réaliser une tâche bien précise.

Les acteurs secondaires : ce sont des entités qui sont sollicitées par le système lors de la réalisation d'une tâche.

Pour notre application web, nous avons déterminé cinq acteurs :

- **Administrateur :** c'est l'administrateur de la plateforme.
- **Utilisateur :** c'est un internaute qui est *déjà inscrit (possède un compte)* et qui s'est authentifié.
- **Visiteur :** c'est un internaute qui ne s'est pas encore authentifié.
- **Chef de projet :** c'est un Utilisateur qui crée le projet.
- **Membre de projet :** c'est un Utilisateur qui a accès au projet sans permission préalable du chef de projet.

4.1.2. Les cas d'utilisation

Un cas d'utilisation (ou use case en Anglais) modélise un service rendu par le système, il exprime les interactions entre les acteurs et le système.

i. Diagramme de cas d'utilisation général

Dans le diagramme de cas d'utilisation général (Figure 22), on regroupe tous les cas d'utilisation de base afin d'avoir une vue globale du fonctionnement de notre système et de mettre en évidence les éventuelles relations entre les cas d'utilisation

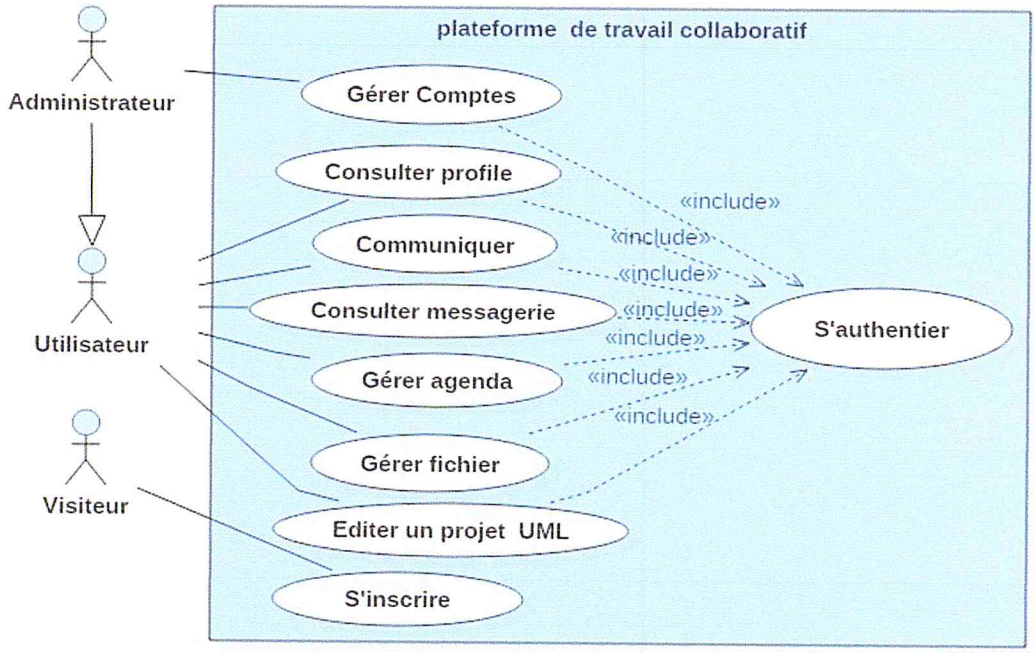

Figure 22 : Diagramme de cas d'utilisation global.

ii. Diagramme de cas d'utilisation gestion des comptes

La gestion des comptes est effectuée par l'administrateur qui peut supprimer ou afficher un compte ou encore d'ajouter un nouvel administrateur (voir Figure 23).

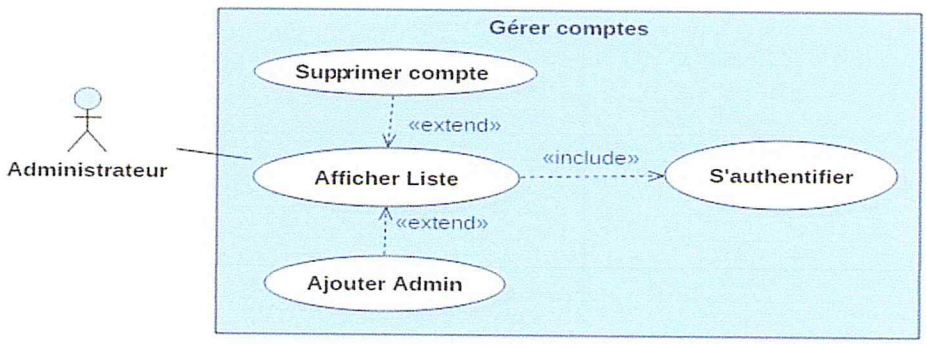

Figure 23 : Diagramme de cas d'utilisation « Gérer comptes ».

iii. Diagramme de cas d'utilisation «communiquer»

Un utilisateur peut communiquer avec un autre utilisateur en effectuant un appel audio et/ou vidéo, ou envoyer un message, il peut même envoyer des fichiers (voir Figure 24).

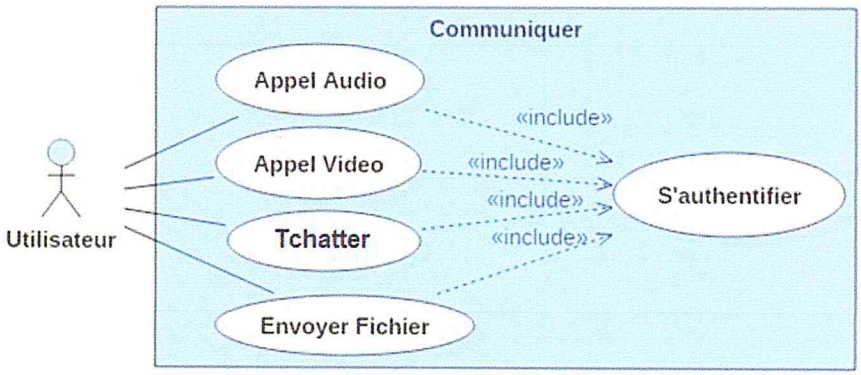

Figure 24 : Diagramme de cas d'utilisation « Communiquer ».

iv. Diagramme de cas d'utilisation «consulter messagerie»

Un utilisateur peut aussi consulter sa messagerie, il peut envoyer des messages en gardant la traçabilité des envois et des réceptions (voir Figure 25)

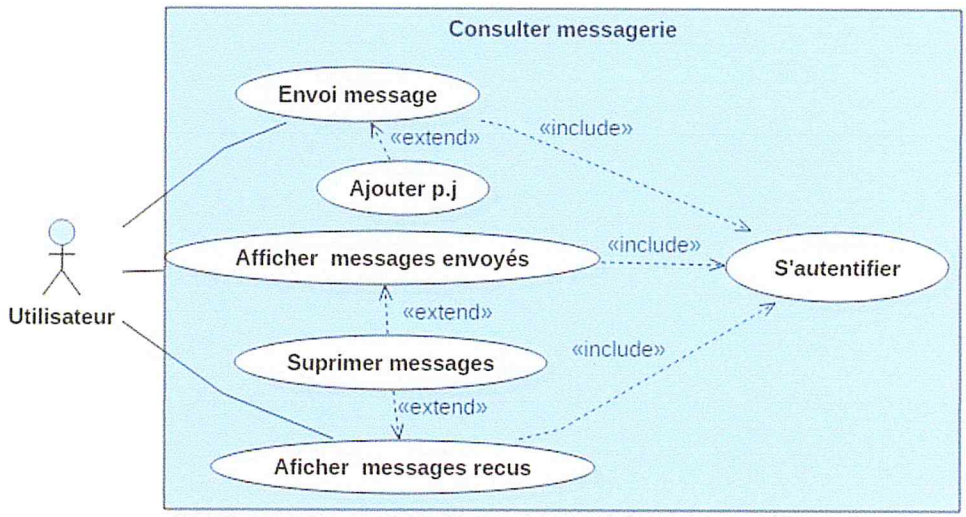

Figure 25 : Diagramme de cas d'utilisation « Consulter messagerie».

v. Diagramme de cas d'utilisation «gérer agenda»

Un utilisateur peut ajouter, modifier, ou supprimer un événement dans sa liste d'événements, il peut également proposer un événement et éventuellement valider un événement proposé (voir Figure 26).

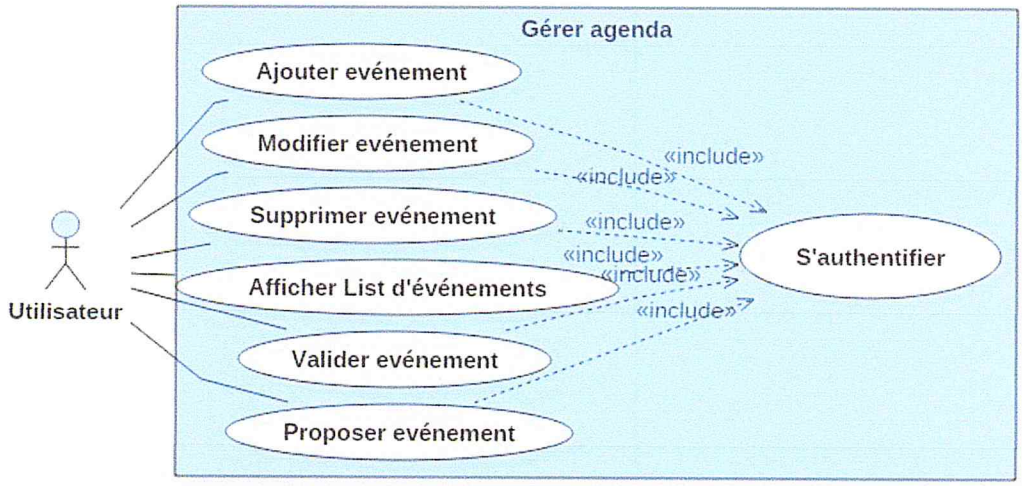

Figure 26 : Diagramme de cas d'utilisation « Gérer agenda ».

vi. Diagramme de cas d'utilisation «gérer fichier»

Un utilisateur peut ajouter, supprimer, ou partager un fichier, il peut également rechercher un fichier et éventuellement le télécharger (voir Figure 27).

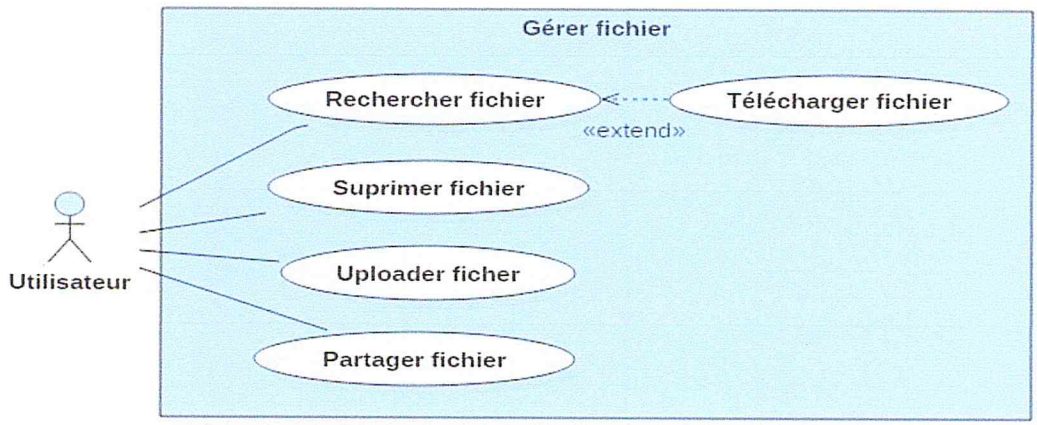

Figure 27 : Diagramme de cas d'utilisation « Gérer fichier ».

vii. Diagramme de cas d'utilisation «édition de projet UML»

Un utilisateur peut rechercher un projet qui est visible, il peut également accéder au projet directement s'il est membre de celui-ci, sinon il peut demander au chef de projet la permission d'accès. Un membre de projet peut éditer le projet (ajouter des diagrammes, importer des diagrammes, etc.), il peut également éditer les diagrammes s'il dispose de droits. Un chef de projet

peut gérer son projet, ainsi que les membres de ce projet (voir Figure 28).

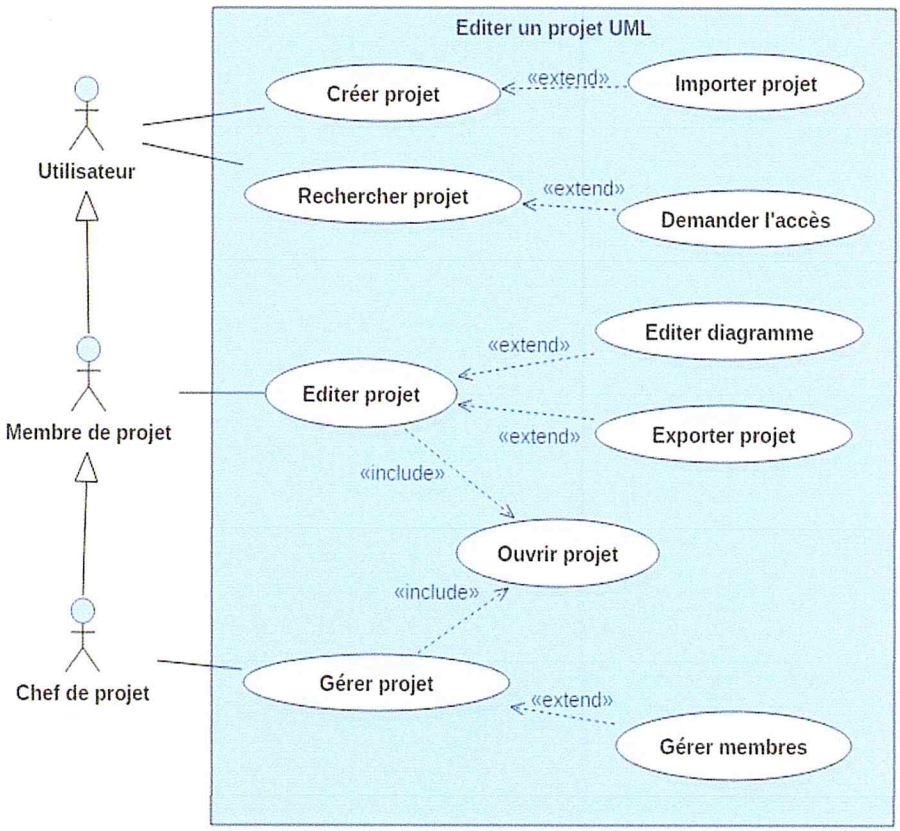

Figure 28 : Diagramme de cas d'utilisation « Editer un projet UML ».

4.2. Les diagrammes de séquence système

Les diagrammes de séquence présentent la vue dynamique du système, l'objectif du diagramme de séquence est de représenter les interactions entre les objets en indiquant la chronologie des échanges, cette représentation se réalise par cas d'utilisation.

4.2.1. Cas d'utilisation « s'authentifier »

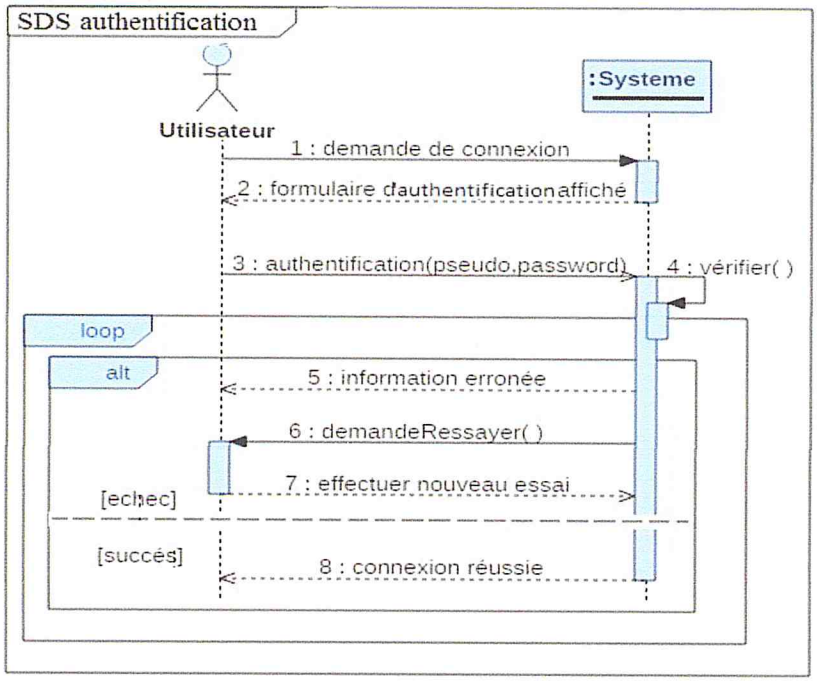

Figure 29 : Diagramme de séquence système « authentification ».

4.2.2. Cas d'utilisation «rechercher et demande d'accès au projet »

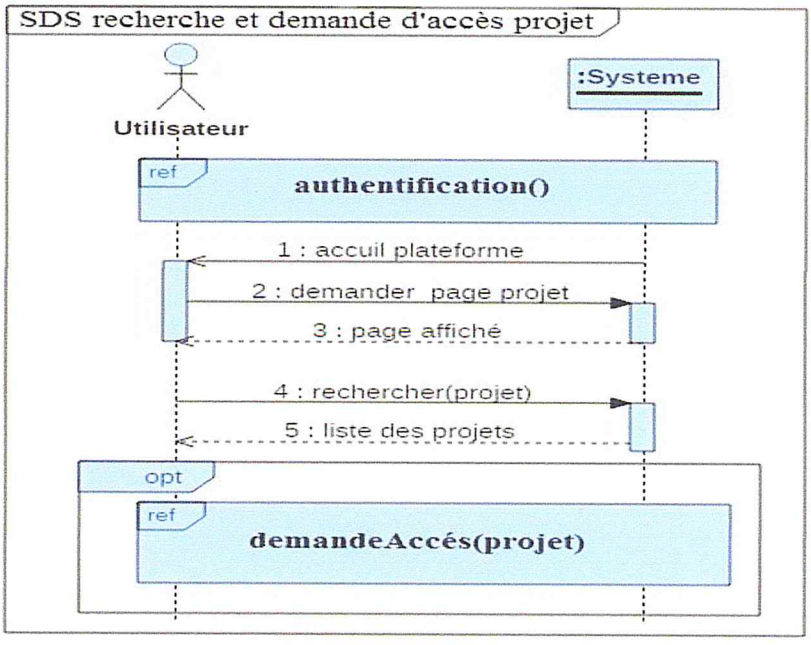

Figure 30 : Diagramme de séquence système «recherche et demande d'accès au projet».

4.2.3. Cas d'utilisation « communiquer»

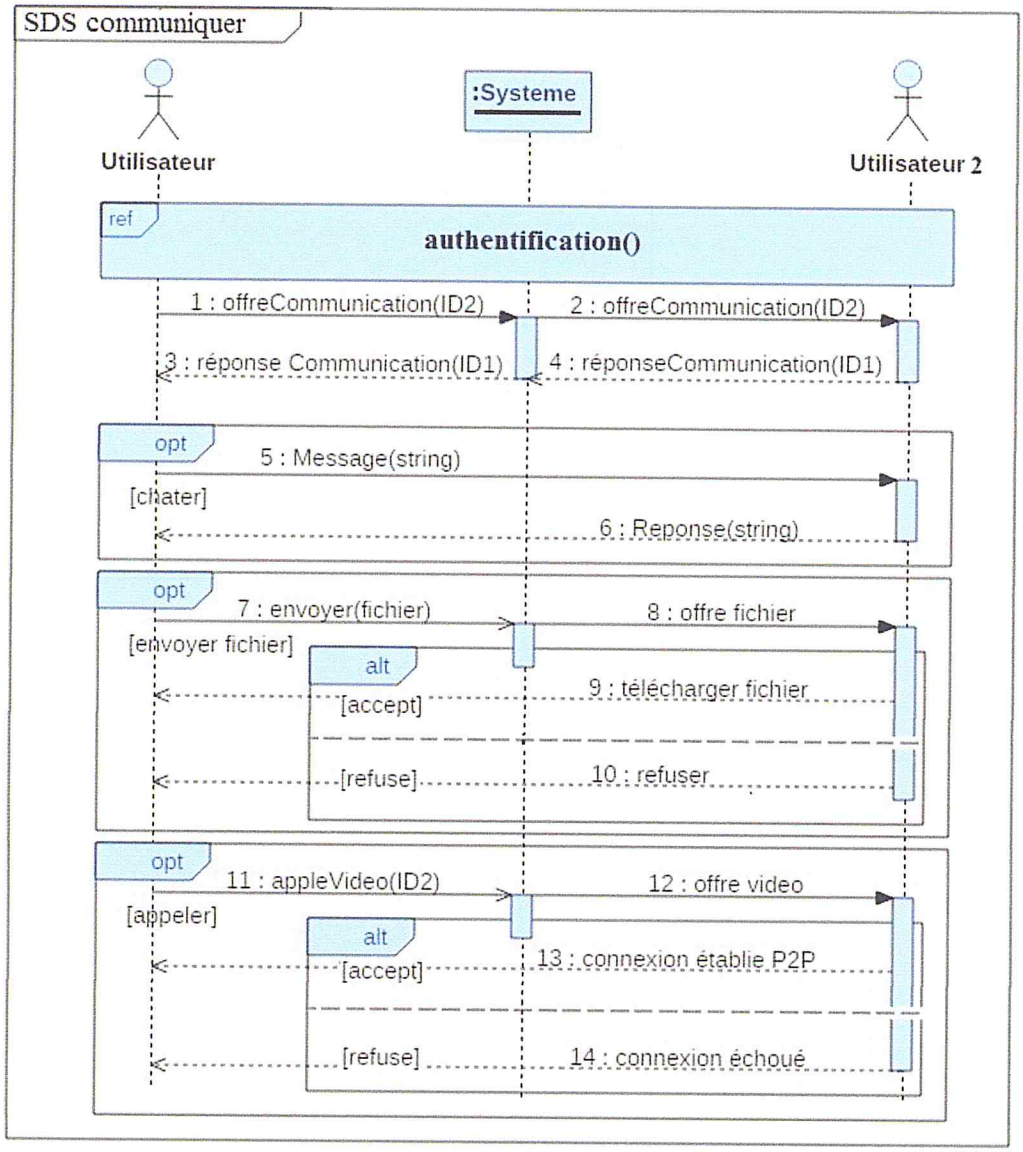

Figure 31 : Diagramme de séquence système « communiquer ».

4.2.4. Cas d'utilisation «consulter messagerie»

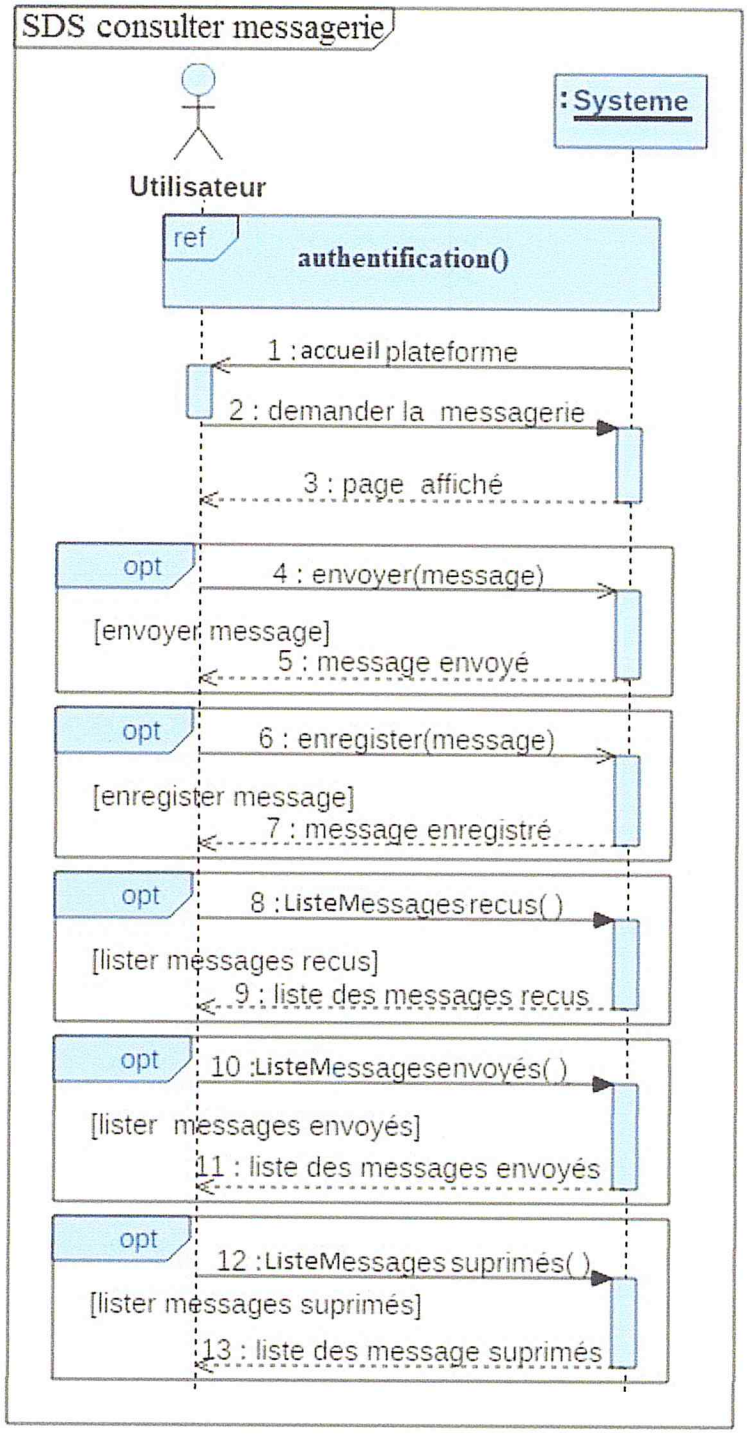

Figure 32 : Diagramme de séquence système « consulter messagerie ».

4.2.5. Cas d'utilisation «Editer diagramme »

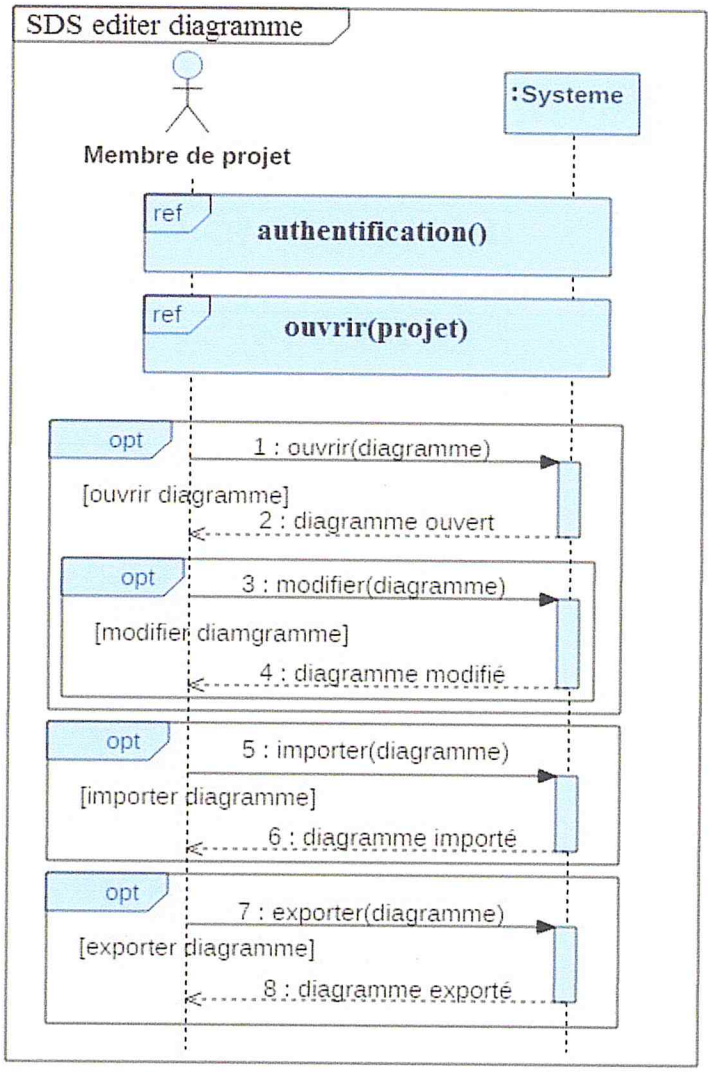

Figure 33 : Diagramme de séquence système «Editer diagramme ».

4.3. Les diagrammes de classe

Afin de bien modéliser notre plateforme de travail collaboratif, nous présentons dans ce qui suit deux diagrammes de classes : la Figure 34 décrit la partie base de données de notre application web ; la Figure 35 montre les différentes parties de notre plateforme et les relations entre elles (pages web, serveur, etc.) en utilisant l'extension WAE de UML puisque le diagramme de classes « classique » d'UML ne suffit pas pour décrire correctement les différents composants d'une application web.

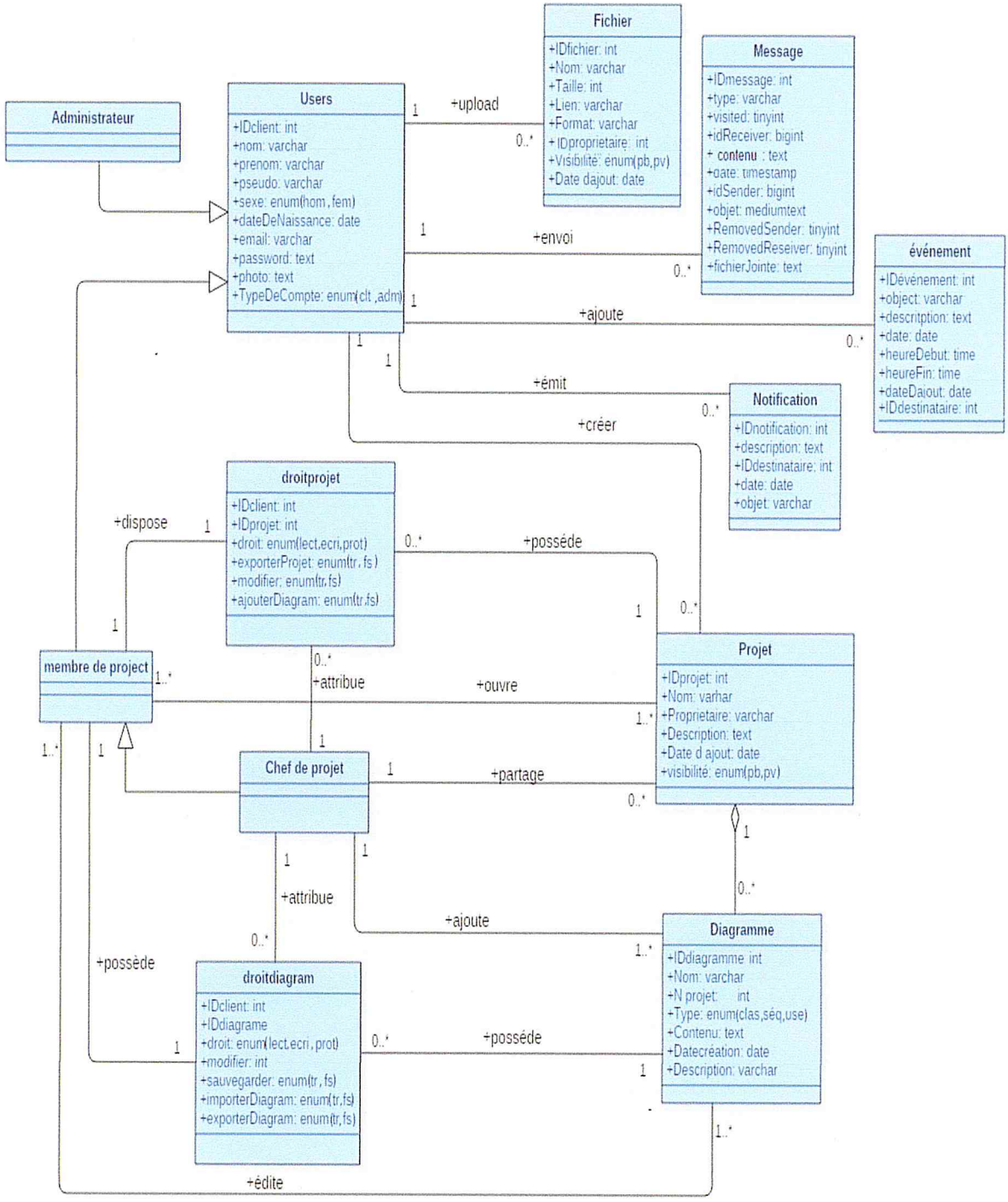

Figure 34 Diagramme de classe de la base de données.

A partir du diagramme de classes de la Figure 34 nous avons appliqué les règles de passage

au modèle relationnel. Le schéma relationnel obtenu est le suivant :

users (*IDclient*, nom, prenom, pseudo, sexe, dateDeNaissance, email, password, photo, typeDeCompte) ;

message (*IDmessage*, type, visited, contenu, date, object, removedSender, removedReceiver, fichierJointe, *IDsender*, IDreceiver**) où IDsender et IDreceiver nouveaux nom de IDclient qui font référence au schéma de relation users.

fichier (*IDfichier*, nom, taille, lien, format,visibilité ,dateDajout, *ID*propriétaire*) où IDpropriétaire nouveau nom de IDclient qui fait référence au schéma de relation users.

evénement (*IDévénement*, object, description, date, heureDebut, heureFin, dateDajout, *IDdestinataire**) où IDdestinataire nouveau nom de IDclient qui fait référence au schéma de relation users.

notification (*IDnotification*, objet, description, objet, *IDdestinataire***) où IDdestinataire nouveau nom de IDclient qui fait référence au schéma de relation users.

projet (*IDprojet*, nom, visibilité, description, date d'ajout, *IDchef**) où *IDchef* nouveau nom de IDclient qui fait référence au schéma de relation users.

diagramme (*IDdiagramme*, nom, type, contenu, date_création, description, *IDprojet**)

droitprojet (droit, exporterProjet, modifier, ajouterDiagram, *IDclient*, IDprojet**) ;

droitdiagram (droit, modifier, sauvegarder, importerdiagrame, exporterDiagram, *IDclient*, IDdiagramme**) ;

ouvrir (*IDclient*, IDprojet**) ;

editer (*IDclient*, IDdiagramme**) ;

NB : pour la notation, nous avons choisi de souligné les clés primaires et de mettre * à la fin de chaque clé étrangère.

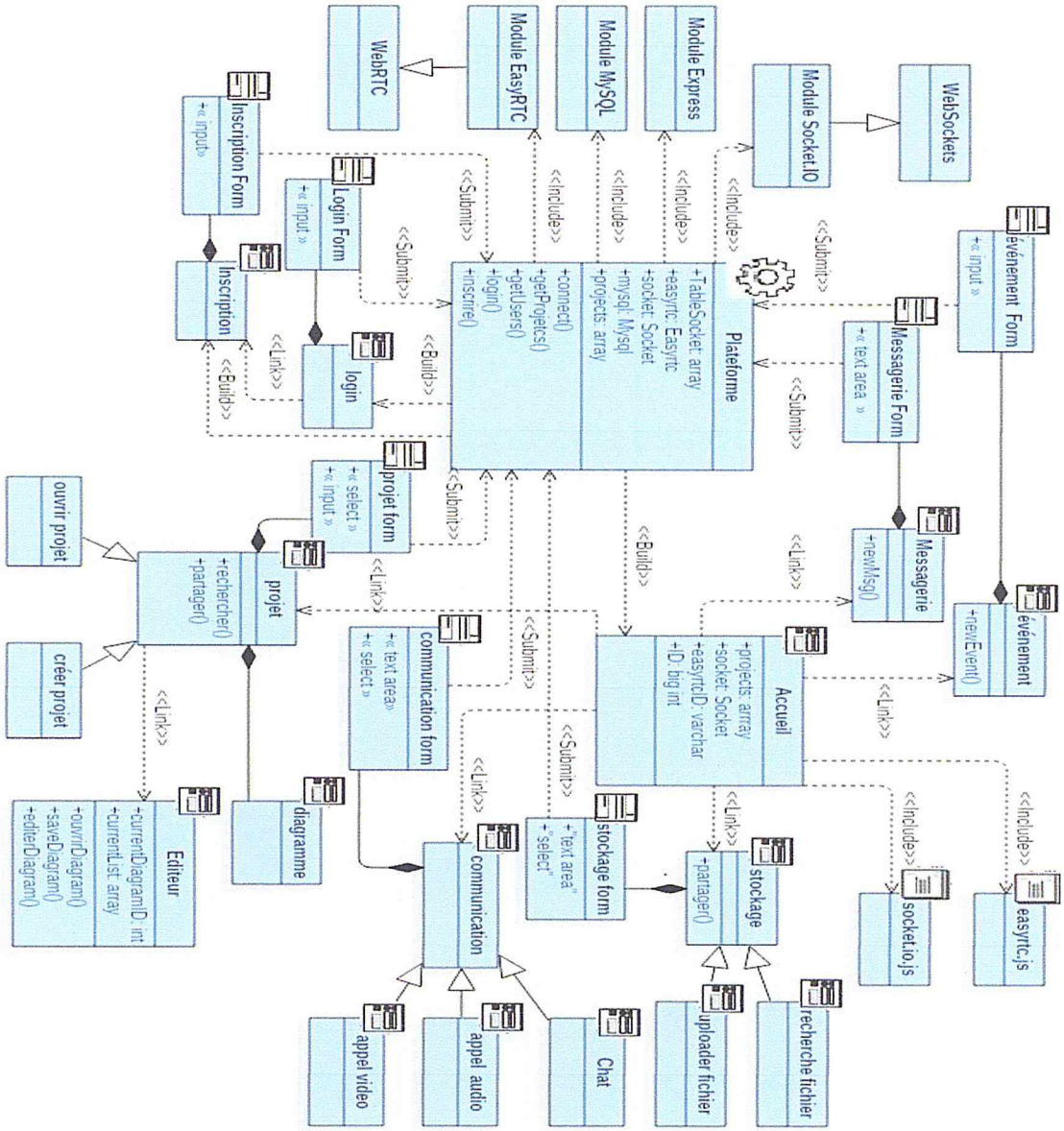

Figure 35 Diagramme de classe de la plateforme.

La Figure 35 représente les différents stéréotypes utilisés dans notre application : pages serveur, pages client, différents types de formulaire utilisés avec leur attributs, ainsi que les bibliothèques JavaScript et les modules utilisés. Plusieurs interactions entre différents classes définissent leurs relations.

4.4. Diagrammes de séquence

4.4.1. Diagramme de séquence d'authentification

Ce diagramme décrit de manière détaillée l'authentification d'un utilisateur.

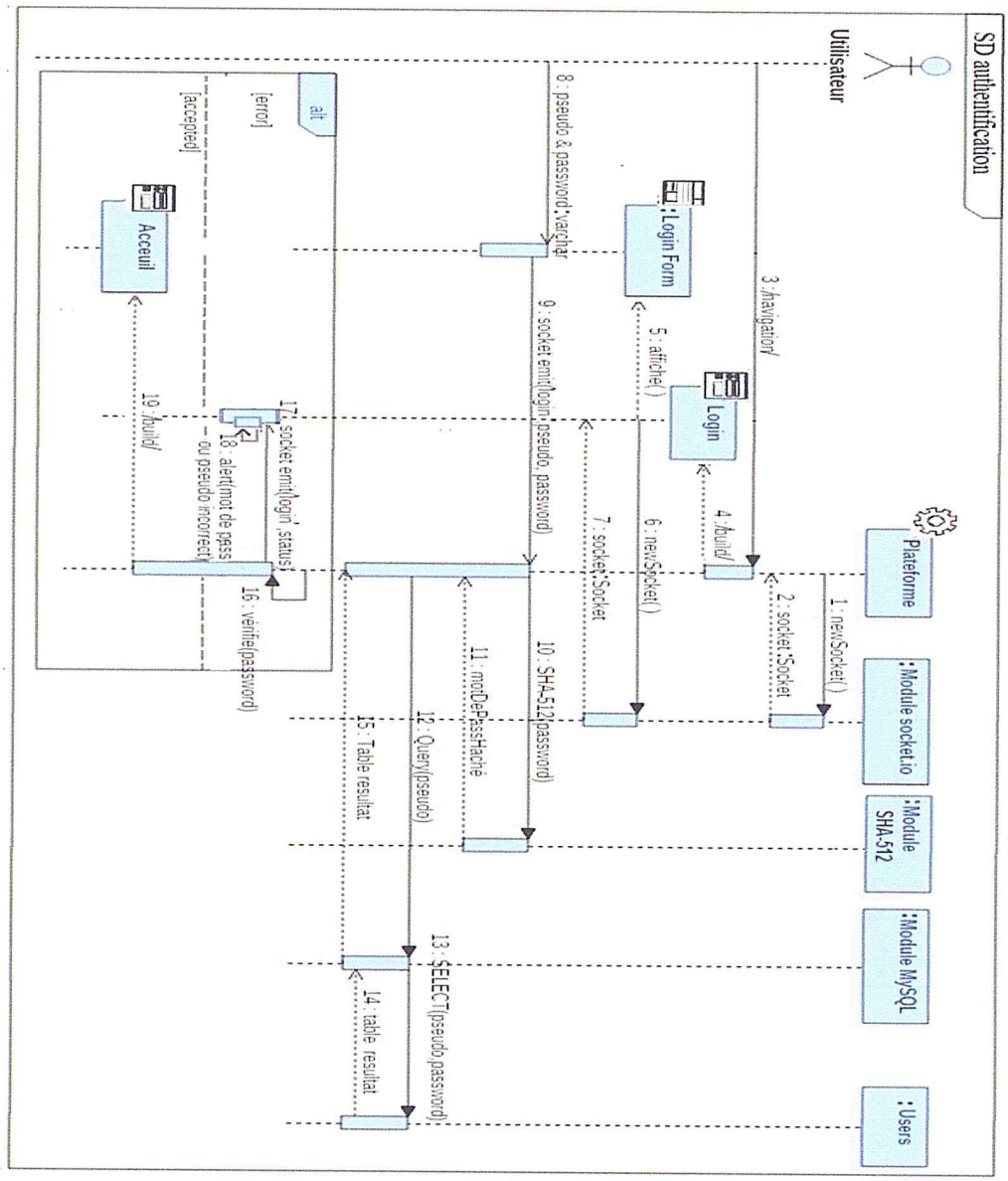

Figure 36 : Diagramme de séquence d'authentification.

4.4.2. Diagramme de séquence pour l'appel vidéo

Ce diagramme défini dans un ordre chronologique. Les séquences d'événements nécessaires pour établir un appel vidéo entre deux utilisateurs.

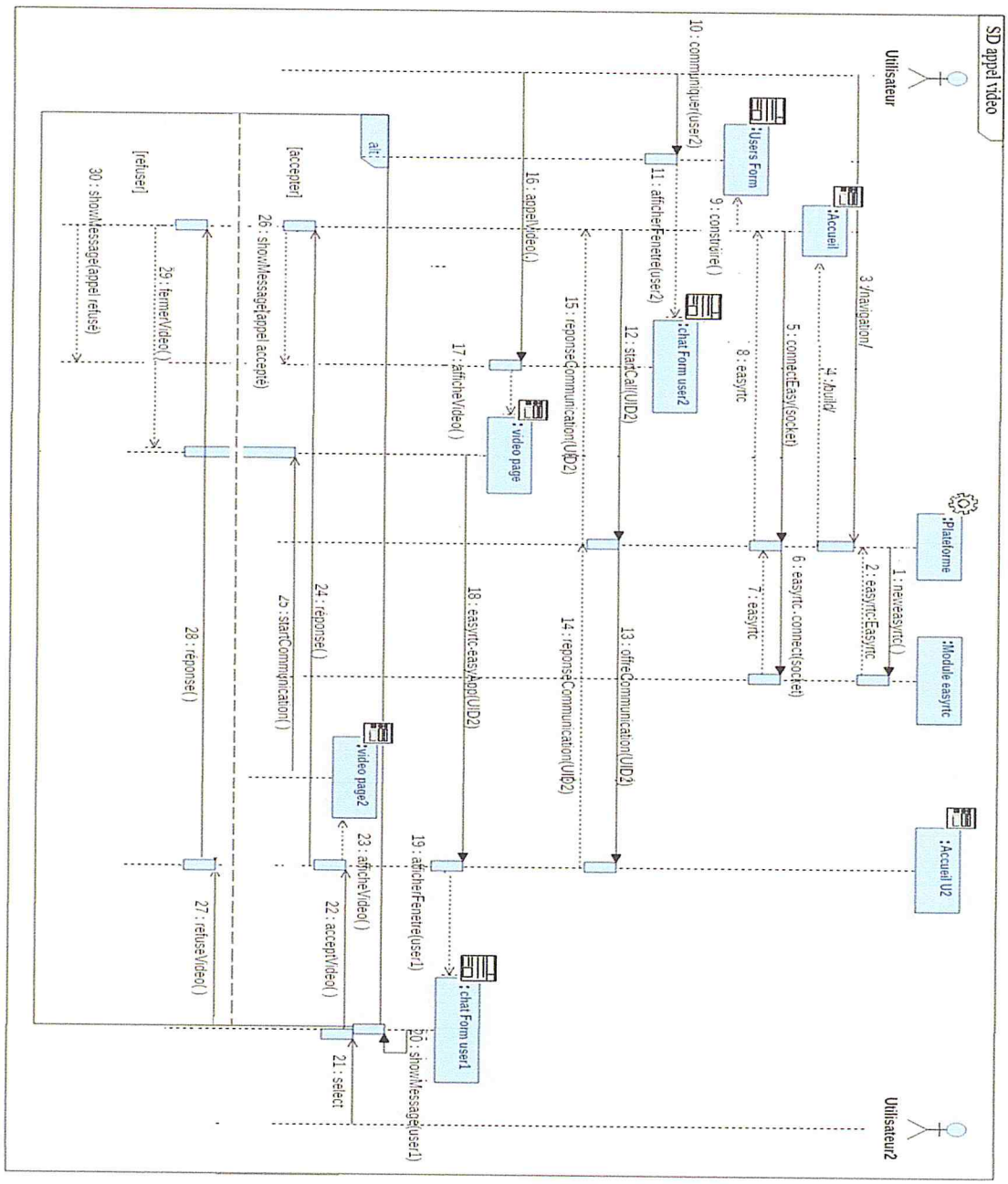

Figure 37: Diagramme de séquence pour un appel video.

4.4.3. Diagramme de séquence pour l'envoi d'un message

Ce diagramme présent en détail le processus d'envoyer un message par l'utilisateur.

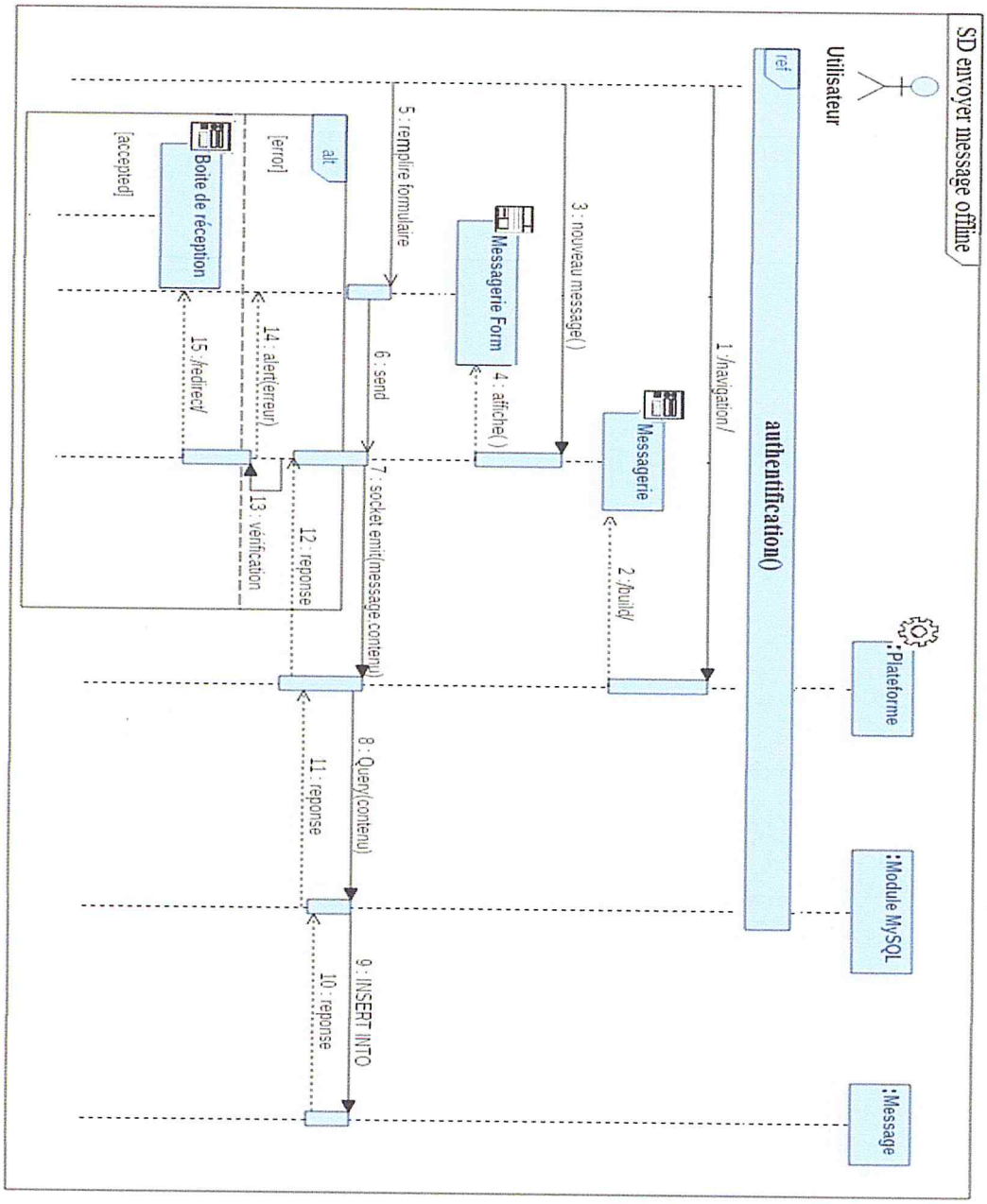

Figure 38 : Diagramme de séquence pour un envoi d'un message.

4.4.4. Diagramme de séquence de la recherche d'un projet

Ce diagramme présente le processus de recherche d'un projet.

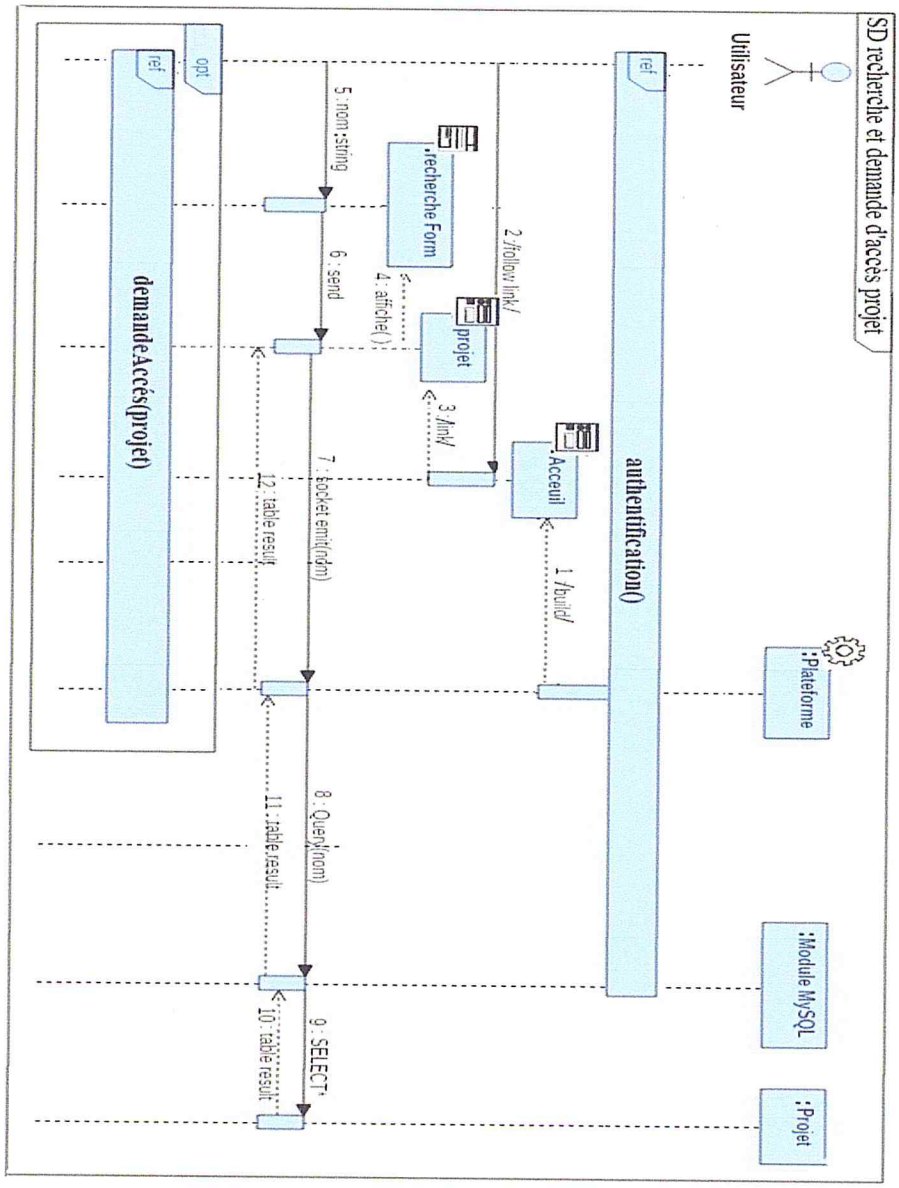

Figure 39 : Diagramme de séquence de la recherche d'un projet.

4.4.5. Diagramme de séquence pour l'édition de diagramme

Ce diagramme représente le processus d'édition d'un diagramme par un membre de projet.

Si ce diagramme est ouvert chez un autre membre, l'ouverture de ce diagramme nécessite la demande du contenu auprès d'un des membres qui ont l'ouvert, sinon le contenu sera

chargé depuis la base de données.

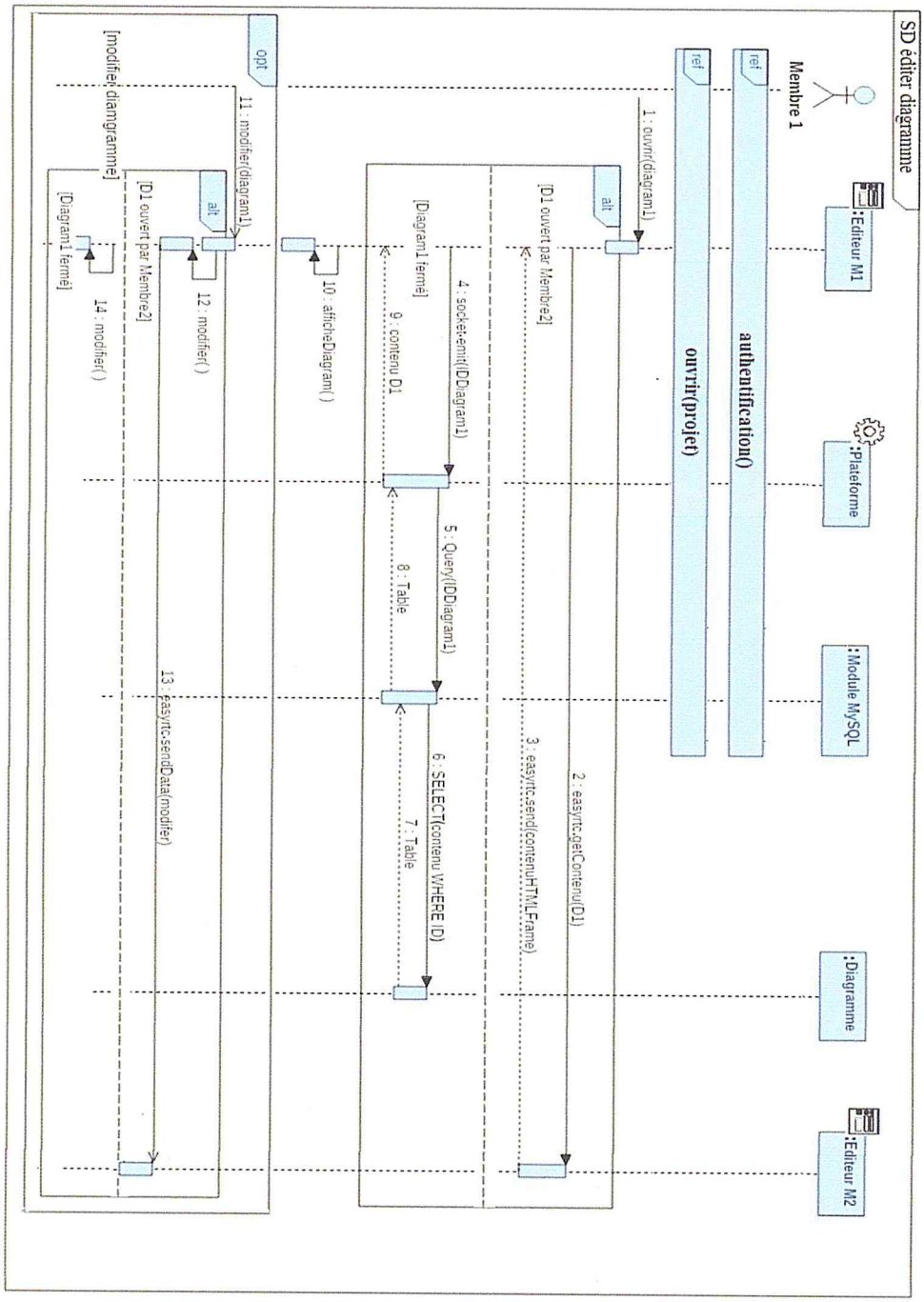

Figure 40 : Diagramme de séquence d'édition de diagramme.

5. Conclusion

En clair, comme nous venons de le voir, ce chapitre était consacré, à la conception détaillée en utilisant l'extension WAE de UML qui nous a décrire le futur fonctionnement de notre application web afin d'en faciliter sa réalisation. des diagrammes, des cas d'utilisation qui nous ont permis de dégager l'architecture générale de notre application web représentée ici par ce diagramme de séquence qui nous renseigne sur la relation et plusieurs interactions utilisateur et notre application selon un ordre chronologique .

Dans le chapitre suivant nous montrerons les étapes, plus en détails, que nous avons suivies pour implémenter et réaliser notre plateforme de travail collaborative.

Chapitre 4

Réalisation de la plateforme de travail collaboratif

1. Introduction

Notre but étant fixé et sa conception élaborée, nous allons passer à la phase de réalisation de notre projet. Nous présentons dans ce qui va suivre les différents moyens et ressources mis en pratique que nous avons utilisés dans l'implémentation de notre projet, ensuite, dans un second temps, nous présentons quelques fonctionnalités de notre plateforme de travail collaboratif.

2. Architecture de la plateforme de travail collaboratif

Les plateformes de travail collaboratif, très largement utilisées chez les particuliers et aussi en interne chez les entreprises, reposent sur l'utilisation de clients légers.

Dans notre cas, notre plateforme de travail collaboratif se déploie sur une architecture réseau qui associe le modèle client/serveur avec le modèle distribué Peer-to-Peer. Il s'agit en fait d'une architecture hybride.

Le réseau est constitué donc d'un serveur centralisé et de clients, les échanges s'effectuent soit en mode client/serveur, soit par connexion directe entre les pairs concernés (en peer-2-peer). Globalement, on favorise ce dernier quand c'est possible pour réduire la charge sur le serveur.

On peut citer l'exemple du service de messagerie hors ligne (modèle client-serveur) : lors de l'envoi d'un message, ce dernier sera obligatoirement acheminé vers le serveur où il sera stocké, le destinataire peut consulter le message à sa reconnexion au serveur.

Ainsi, notre plateforme de travail collaboratif s'appuie sur l'utilisation de l'architecture client web léger : une application Web qui satisfait une seule contrainte qui est la disponibilité d'un

navigateur web compatible HTML5 et JavaScript. La Figure 41 montre l'architecture de notre application ainsi que les différents protocoles et technologies utilisés.

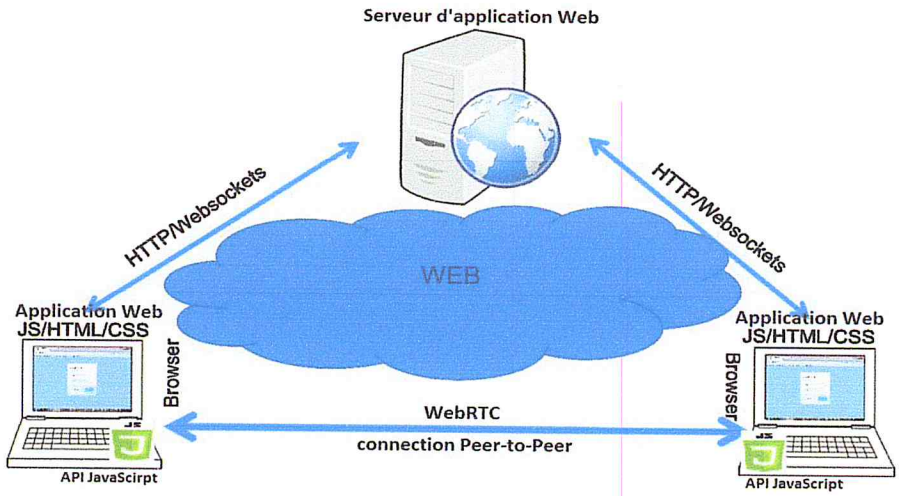

Figure 41 : Architecture de notre plateforme de travail collaborative.

Les différents éléments de notre plateforme collaborative sont :

- Le serveur web : Node.js est utilisé notamment pour la création du serveur web puisqu'il utilise un modèle d'entrée/sortie non bloquant qui lui permet de gérer plusieurs tâches de façon asynchrone.
- Les pages web : elles sont implémentées en HTML et JavaScript.
- La communication bidirectionnelle entre le client et le serveur est établie grâce à socket.io, une bibliothèque qui implémente Websockets.
- La communication temps réel entre les pairs est établie grâce à la bibliothèque EasyRTC qui facilite l'utilisation de l'API WebRTC.

3. Langages de développement

Notre projet vise à développer une plateforme de travail collaboratif temps réel qui utilise un ensemble de technologies web et d'outils JavaScript qui vont servir à réaliser le but notre projet.

Pour créer un site web, HTML et CSS sont indispensables. Cependant, ces langages ne suffisent pas pour le réaliser, il faut compléter ces deux langages avec d'autres ; dans notre cas nous avons utilisé le langage JavaScript car WebRTC et Node.js sont basés seulement sur ce langage de

script, sans oublier le langage de requêtes SQL qui permet de communiquer avec notre base de données.

Remarque **:**

Il y a eu plusieurs versions des langages HTML et CSS, pour notre part nous avons utilisé les dernières versions : HTML5 et CSS3.

4. Environnement de développement

Lors du développement de notre plateforme, nous avons utilisé les outils logiciels suivants :

- **Notepad++ :** c'est un éditeur de texte et de code pour Windows. Il permet l'édition de plusieurs documents simultanément qui s'adapte aux langages de développement web, il offre de nombreuse facilités : coloriage adapté au langage, recherche avancé dans les dossiers, auto complétion, simplicité d'utilisation, légèreté de démarrage et de manipulation, etc.[1]

- **L'interpréteur Node.js (Node.js command prompt)**

C'est une console de Windows configurée pour reconnaître Node.js [Neb, 13]. Elle permet de lancer nos applications Node.js (Voir Figure 42).

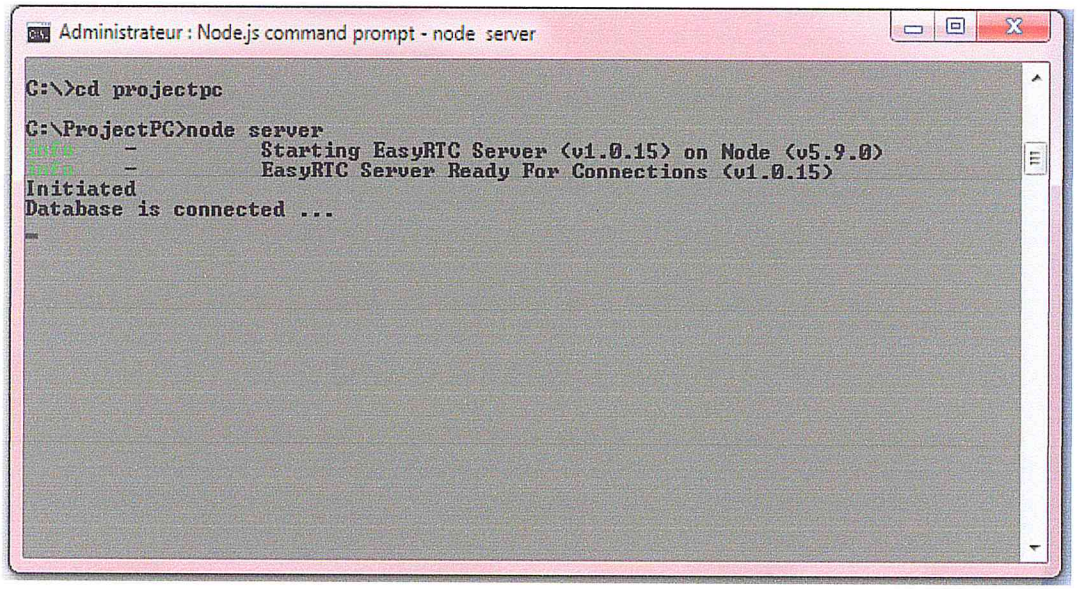

Figure 42 : Node.js - L'interpréteur Node.js sous Windows

[1] https://notepad-plus-plus.org/fr/

5. Implémentation

5.1.Côté serveur

5.1.1. Serveur Node.js

Node.js peut être comparé à Python, Ruby, JavaEE, PHP. On trouve une multitude de comparatifs, mais Node.js présente de nombreux intérêts : comme les performances du moteur V8, et la disponibilité des modules externes, ce qui nous a facilité l'utilisation de Node.js.

5.1.1.1. Modules NPM utilisés

Node.js est très riche grâce à son extensibilité. Les extensions sont appelées modules. Il existe des milliers de modules qui offrent des fonctionnalités variées. Node.js dispose d'un système de gestion de module appelé NPM (Node Package Manager) qui facilite la gestion des modules (chercher, installer, désinstaller, gérer les dépendances, etc).

Pour notre application, nous avons utilisé les modules suivants :

http : interface de Node.js qui facilite la configuration du serveur http traditionnel.

express : permet un routage statique et dynamique robuste.

socket.io : offre la communication bidirectionnelle et en temps réel entre le serveur et les clients.

Easyrtc : librairie qui facilite l'utilisation de WebRTC et qui prend en charge le transfert de données audio, video, texte en pair à pair afin de réduire la charge sur le serveur.

Easyrtc_ft : permet d'envoyer les fichiers entre les pairs.

Sha512 : permet de hacher les mots de passe pour les sécuriser.

delivery : facilite l'envoi des fichiers entre le client et le serveur et vice versa.

fs : permet la gestion de fichiers (création, lecture, écriture, etc.).

MySQL : initialisation et interaction avec la base de données.

EJS : permet d'intégrer du code JavaScript au milieu d'une balise HTML.

xmlbuilder : permet de générer les documents XML avec JavaScript côté serveur.

Jsdom : permet de construire l'arborescence DOM d'un fichier côté serveur.

La Figure 43 est une portion du code JavaScript du serveur qui expose les différents modules utilisés dans notre plateforme de travail collaboratif.

```
1   var http     = require("http");
2   var express  = require("express");
3   var io       = require("socket.io");
4   var easyrtc  = require("easyrtc");
5    var sha512 = require('sha512')
6     var dl   = require('delivery'),
7     util = require('util')
8       fs   = require('fs');
9       var builder = require('xmlbuilder');
10      var jsdom = require("jsdom");
11  var httpApp = express();
12  var mysql      = require('mysql');
```

Figure 43 : Partie du code de serveur.js qui montre l'usage des modules.

5.1.1.2. Mise en place des comportements

La figure ci-dessous est un extrait du code du serveur (fichier serveur.js) qui montre l'authentification d'un utilisateur sur la plateforme ; la plateforme vérifie l'existence du pseudonyme et la correspondance du mot de passe saisi au niveau de la base de données, si les données sont conformes, le serveur le redirige vert la page d'accueil de la plateforme sinon il retourne vers la page d'authentification.

```
725     socket.on('login',function(data){
726     var query ='SELECT * FROM users WHERE pseudo = ?';
727 connection.query(query,[data.pseudo],function(err, result) {
728     var existe = false;
729     for(i in result){
730  if(result[i].password==sha512(data.password+'plateforme').toString('hex')){
731     existe=true; break;
732     }
733  }
734    socket.emit("login_response",existe);
735 });
736     });
737
```

Figure 44: Extrait du code server.js.

5.1.2. Socket.io

- **Définition**

Socket.io est une librairie qui implémente le protocole WebSockets. Elle permet de faire la liaison bidirectionnelle en temps réel avec le serveur afin d'appeler ses propres fonctions, de même pour le serveur qui peut appeler des fonctions propres au client .Cette liaison se fait par le

biais d'événements que l'on crée et que l'on appelle. Elle est compatible avec un très grand nombre de navigateurs, plateformes, et d'appareils mobiles, en fournissant des résultats rapides et fiables.

- **Principe de fonctionnement**

Socket.io utilise un système d'événements ; en effet, nous avons créé des événements que nous appellerons par la suite à partir du client vers le serveur et vice versa.

Comme on peut voir dans la Figure 45, lorsqu'un client à partir de son navigateur se connecte sur le serveur avec socket.io, l'événement 'connection' sera déclenché et celui-ci inclut d'autres événements qui peuvent se déclencher lorsqu'un client lui émet un message.

Dans cet exemple, l'utilisateur émet l'évènement 'login' comportant un message qui inclue le nom d'utilisateur et son mot de passe. Une fois que le serveur traite ces données, il envoie un message contenant le résultat de son traitement au client.

```
71  socketServer.sockets.on('connection',function(socket, pseudo){
140     socket.on('login',function(data){
141     var query ='SELECT * FROM users WHERE pseudo = ?';
142 connection.query(query,[data.pseudo],function(err, result) {
143     var existe = false;
144     for(i in result){
145  if(result[i].password==md5(data.password+'plateforme')){
146     existe=true; break;
147     }
148  }
149    socket.emit("login_response",existe);
150 });
151     });
```

Figure 45 Portion du code d'utilisation socket.io.

5.1.3. EasyRTC

- **Définition**

Il peut être défini comme suit :

«EasyRTC is a JavaScript library that consists of a peer/browser-side and a backend JavaScript server built on top of node.js. It is open-source and claims to be the fastest, most robust, easiest way to implement secure WebRTC video, audio and data applications for the enterprise» [Jak, 15].

EasyRTC est donc une bibliothèque JavaScript qui se compose d'un pair côté navigateur et d'une extrémité JavaScript serveur construite sur Node.js. Il est open-source et prétend être le plus rapide, robuste, simple pour la mise en œuvre d'applications audio, vidéo et de données avec WebRTC.

- **Utilisation de EasyRTC**

Pour effectuer un appel audio, vidéo ou envoyer et recevoir des données, une récupération des informations de chaque client est nécessaire pour établir la connexion entre eux. EasyRTC utilise des identifiants uniques appelés *easyrtcid* qui identifient chaque pair. Cet identifiant est fourni par le serveur de signalisation que l'application utilise. Le serveur de signalisation crée une session pour chaque utilisateur qui utilise le serveur.

➢ Les canaux de données

L'utilisation de 'EasyRTC' pour envoyer et recevoir des données est effectuée en utilisant l'objet *easyrtc*, en le configurant de sorte qu'il utilise le canal de données.

La Figure 46 montre des extraits de code sur la façon d'utiliser EasyRTC pour créer une connexion P2P entre deux pairs et envoyer des données entre eux. Au lieu d'utiliser sendDataP2P, la fonction sendData pourrait être utilisée, elle tente d'envoyer les données en mode peer-to-peer, sinon ils seront envoyés en mode client-serveur en utilisant socket.io.

```
38  function connect() {
39      //easyrtc.enableDebug(true);
40       easyrtc.enableDataChannels(true);
41      easyrtc.enableVideo(false);
42       easyrtc.enableAudio(false);
43       easyrtc.enableVideoReceive(false);
44       easyrtc.enableAudioReceive(false);
45       easyrtc.setUsername(nom);
47       easyrtc.setDataChannelOpenListener(openListener);
48       easyrtc.setDataChannelCloseListener(closeListener);
49       easyrtc.setPeerListener(Listener);
50       easyrtc.setRoomOccupantListener(convertListToButtons);
54       easyrtc.connect("easyrtc.dataMessaging", loginSuccess, loginFailure);
57  }
58  connect();
230 function sendStuffP2P(otherEasyrtcid) {
231     if (easyrtc.getConnectStatus(otherEasyrtcid.destination) === easyrtc.IS_CONNECTED) {
232         easyrtc.sendData(otherEasyrtcid.destination, 'message', otherEasyrtcid);
233     }
```

Figure 46 : Extraits de code EasyRTC.

➢ Les appels multimédia

Pour effectuer les appels audio et vidéo, EasyRTC est utilisé d'une manière similaire au canal de données, en configurant l'objet *easyrtc* pour autoriser l'envoi et la réception de données multimédia (voir Figure 46).

La fonction setRoomOccupantListener est un gestionnaire d'événements qui se déclenche quand un utilisateur entre dans la même chambre que l'utilisateur.

Pour lancer un appel audio/vidéo, il faut utiliser la fonction 'easyApp' (voir Figure 47).

```
case ("video"):{
    //messageOrVideo("video");
    //console.log(video1+" , "+video2);
    easyrtc.easyApp("videoConference", video1,
[video2], loginSuccess, loginFailure);
```

Figure 47 : Invocation d'un appel vidéo

5.1.4. Express.js

- **Définition**

« *Express est une infrastructure d'applications Web Node.js minimaliste et flexible qui fournit un ensemble de fonctionnalités robuste pour les applications Web et mobiles.* »[2].

Ce module permet de faciliter la gestion des routes d'une application, et d'utiliser les Template pour créer des pages web.

- **Principe de fonctionnement**

Le routage est l'opération de détermination de la méthode de réponse d'une application à une demande client adressée par un chemin précis par une méthode HTTP (GET, POST, etc.).[3]

Chaque route peut avoir une ou plusieurs fonctions de gestionnaire, qui sont exécutées lorsque la route est mise en correspondance.

[2] http://expressjs.com/fr/
[3] http://expressjs.com/fr/starter/basic-routing.html

```
16
17  var httpApp = express();
18  httpApp.use('/', express.static(__dirname))
19 .get(/^\/(.*)\/(.*)/, function(req, res) {
20      res.render('main.ejs', {url:"/"+req.params[0]+".html",fonction:req.params[1]});} )
21
22 .get('/:principal', function(req, res) {
23      res.render('main.ejs', {url: req.params.principal+".html",fonction:''});})
24
```

Figure 48: Exemple de configuration de routage.

La Figure 48 illustre l'utilisation du module Express.js dans notre serveur, dans notre cas nous avons utilisé deux types de routage :

- Routage dynamique pour les demandes de pages qui correspondent aux expressions régulières « /* » ou bien « / * /*», les étoiles sont des paramètres qui seront récupérés et passés à la page HTML de destination grâce au module EJS pour les exécuter lors du chargement de cette page.
- Routage statique pour toutes autres demandes qui ne correspondent pas aux expressions régulières.

5.1.5. WampServer 2.5

WampServer (Windows Apache MySQL PHP) est une plate-forme de développement d'applications Web dynamiques sous Windows. Il intègre un serveur Apache 2 (serveur HTTP), un serveur MySQL (un SGBD : Système de Gestion de Base de Données), un interpréteur PHP, ainsi qu'une interface Web PhpMyAdmin permettant la gestion et l'administration de base de données.[4]

Dans notre cas nous avons utilisé WampServer 2.5 qui intègre : MySQL 5.6.17, PHP 5.5.12 : et PHPMyAdmin 4.1.14.

[4] http://www.wampserver.com/

5.2.Côté client :

5.2.1. Pages HTML et code JavaScript

Les pages HTML qui nécessitent l'utilisation de socket.io, delivery.js, EasyRTC ou bien d'autres modules doivent récupérer les fichiers JavaScript qui permettent d'effectuer des actions du côté du client pour communiquer avec le serveur.

La Figure 49 montre comment se fait la récupération des fichiers JavaScript du côté du client comme socket.io.js. Celui-ci est automatiquement fourni par le serveur node.js via le module socket.io.

```
<html>
  <head>
    <title>Chat instantanié</title>
    <link rel="stylesheet" href="css/jquery.chatjs.css" />
        <link rel="icon" href="images/usdb.png">
  </head>
    <script src="/socket.io/socket.io.js" charset="UTF-8"></script>
<script type="text/javascript" src="/easyrtc/easyrtc.js"></script>
<script type="text/javascript" src="js/easyrtc_ft.js"></script>
<script type="text/javascript" src="js/delivery.js"></script>

    <script type="text/javascript" src="js/chat.js"></script>
     <script src="client3.js" type="text/javascript" charset="ISO-8859-1" ></script>


</html>
```

Figure 49 : Récupération des librairies dans une page html.

La Figure 50 représente une partie du code de la fonction qui permet d'afficher la fenêtre de discussion d'un utilisateur précis. Cette opération sera invoquée dans deux situations : soit lors d'un click de la part de l'utilisateur lui-même sur le nom d'un autre utilisateur dans la barre des utilisateurs connectés, soit dans le cas où il reçoit un message de la part d'un autre utilisateur.

Cette fenêtre comporte une partie d'affichage des messages envoyés et reçus, et un champ de saisie pour saisir un texte à envoyer, ainsi qu'un ensemble de boutons qui permettent d'initier un appel audio, appel audio/vidéo, et d'envoyer un message vers cet utilisateur.

Cette fonction permet aussi d'initier un appel de communication pair à pair en utilisant Easyrtc qui utilise lui-même socket.io (phase de signalisation), pour permettre ensuite à l'utilisateur d'envoyer des messages directes aux autres utilisateurs afin de réduire la charge sur le serveur.

```
function afficherFenetre(id){
    var nom ;
    for(i in usersGlobal){
     if(usersGlobal[i].id==id){
     nom=usersGlobal[i];
     }
     }
     var body = document.getElementById('body');
     var exist = false;
     for(i in fenetres){
     if(fenetres[i].id=='chat-window'+nom.id){
     exist=true;
     }
     }
     if(  !exist  ){
     var fenetre = document.createElement('div')
     fenetre.id='chat-window'+nom.id;

       textarea.onkeyup=function() {
          if (event.keyCode == 13) {
      try{
      this.removeChild(this.lastChild);
      }catch(err){}

      this.innerHTML= this.innerHTML.replace(/ /g," ");

      while(/^ /.test(this.innerHTML)){
     this.innerHTML= this.innerHTML.substring(1,this.innerHTML.length);
      }
      if( /\S/.test(this.innerHTML)&&this.getElementsByTagName("br").length==
      addMessage(nom.id,{id:selfEasyrtcid,destination:nom.id,msg:this.innerHTI
      sendMessage({id:selfEasyrtcid,destination:nom.id,msg:this.innerHTML,typ
        }
     this.innerHTML='';
      }
```

Figure 50 : Affichage de fenêtre de discussion.

6. Présentation de l'application

6.1.La page d'accueil

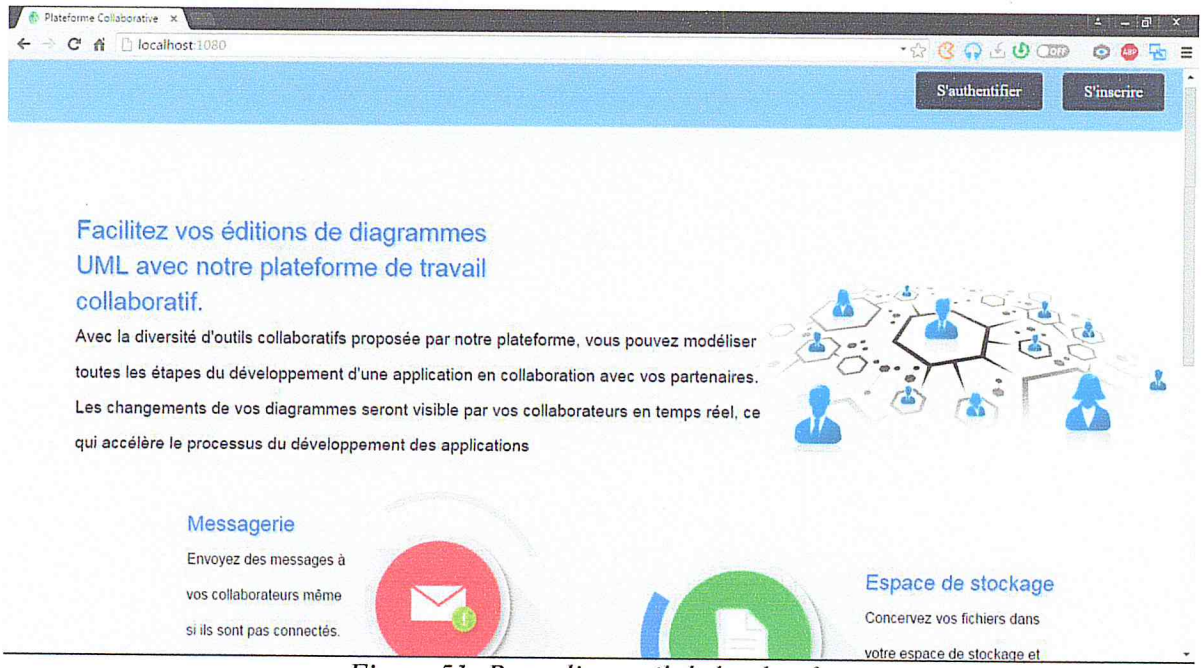

Figure 51 :Page d'accueil de la plateforme.

La Figure 51 présente la page d'accueil de notre plateforme. Elle comporte une petite description de chaque outil de la plateforme et permet de passer vers la page d'inscription et la page d'authentification.

6.2.La page d'inscription

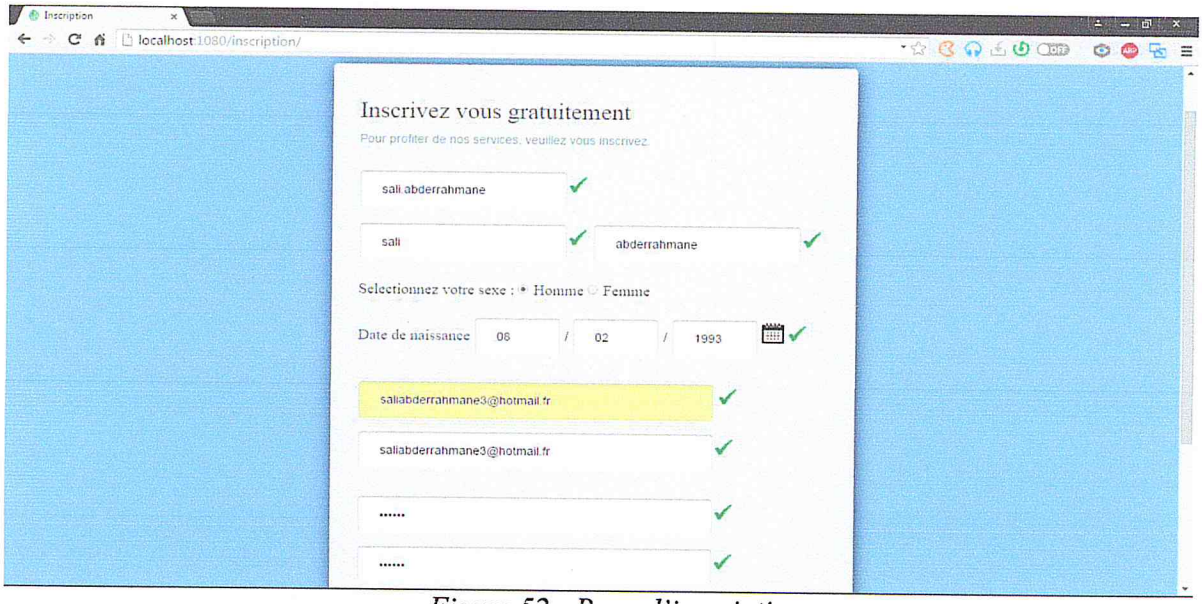

Figure 52 : Page d'inscription

La Figure 52 présente la page d'inscription : un visiteur peut s'inscrire facilement en remplissant le formulaire affiché. Cette page donne des indices de validité de chaque case remplie, ce qui facilite la détection de fautes au moment de saisie.

6.3.Page d'accueil « Utilisateur »

La Figure 53 présente la page d'accueil d'un utilisateur. Elle comporte un calendrier et une horloge, ainsi d'un panneau des événements prochains ; elle comporte également une liste des utilisateurs connectés ancrée (disponible sur toutes les pages de la plateforme). L'utilisateur peut initier une conversation (à n'importe quel moment) avec un autre utilisateur en cliquant sur son nom dans la liste.

Cette page comporte aussi un panneau des boutons d'accès vers les outils de la plateforme. Notons que ce panneau est disponible en haut de page de chaque page de la plateforme (voir Figure 54) pour faciliter le déplacement d'un outil vers l'autre.

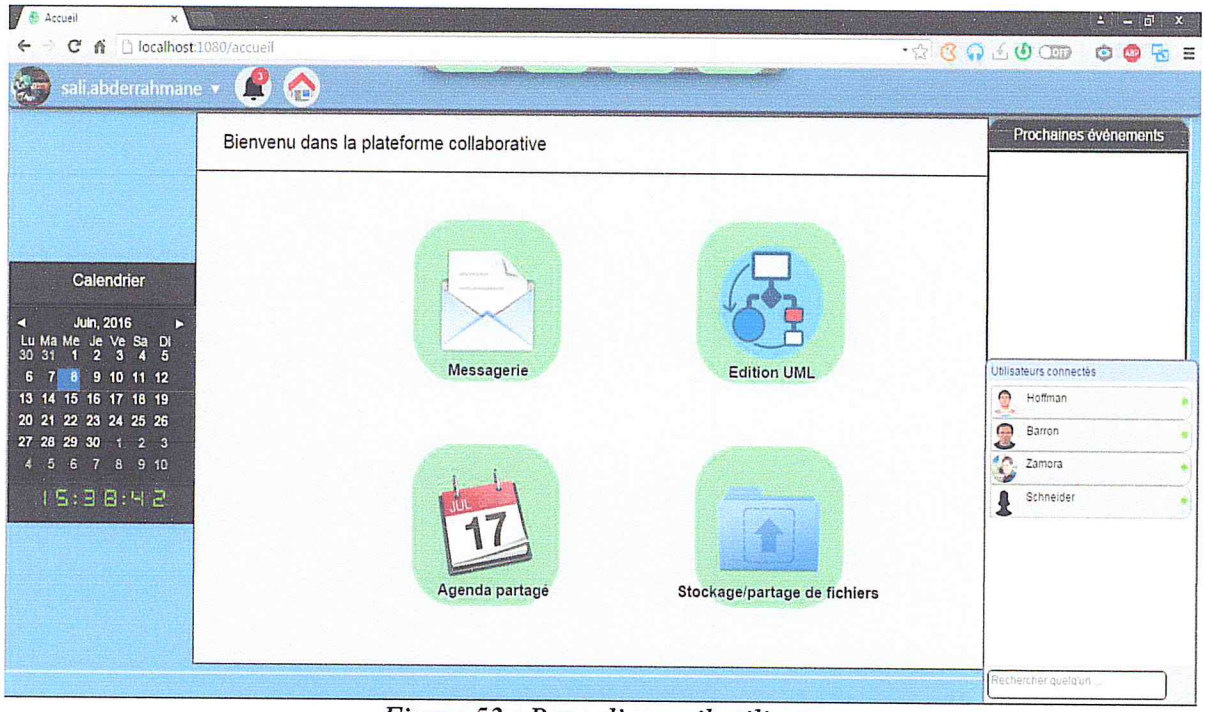

Figure 53 : Page d'accueil utilisateur.

6.4.Outils :

6.4.1. Messagerie

La Figure 54 illustre la boite de réception d'un utilisateur : l'utilisateur peut rédiger un nouveau message en cliquant sur le bouton « Nouveau message », il peut aussi consulter les messages envoyés et les messages supprimés.

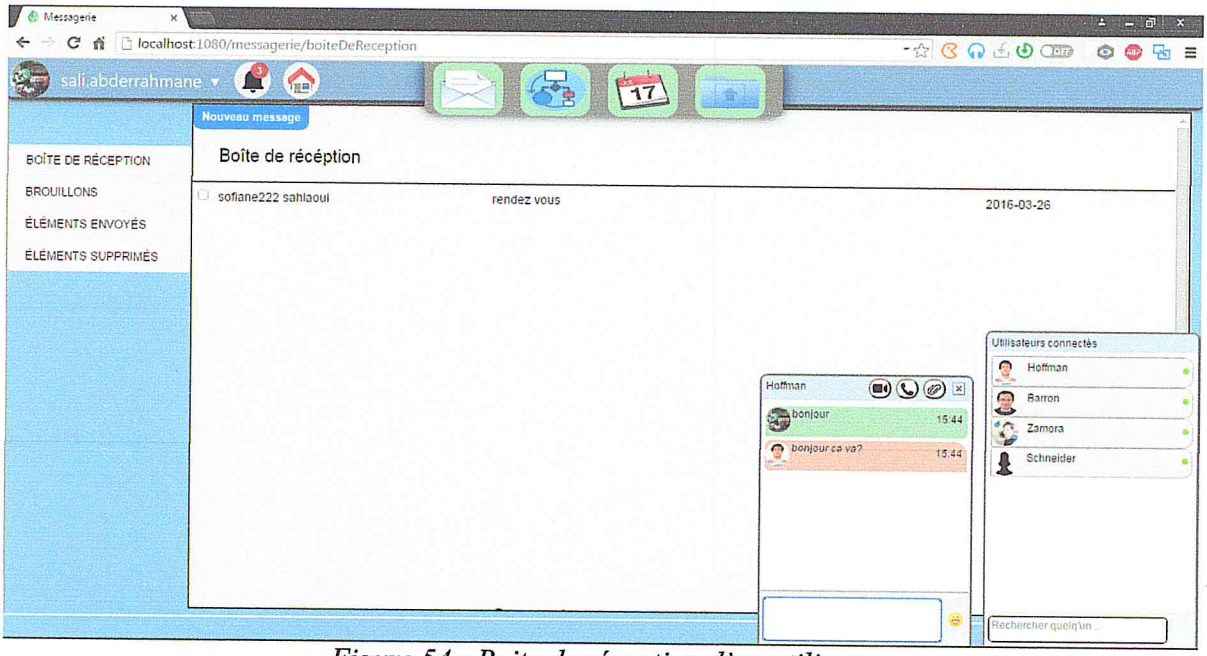

Figure 54 : Boite de réception d'un utilisateur.

6.4.2. Agenda partagé

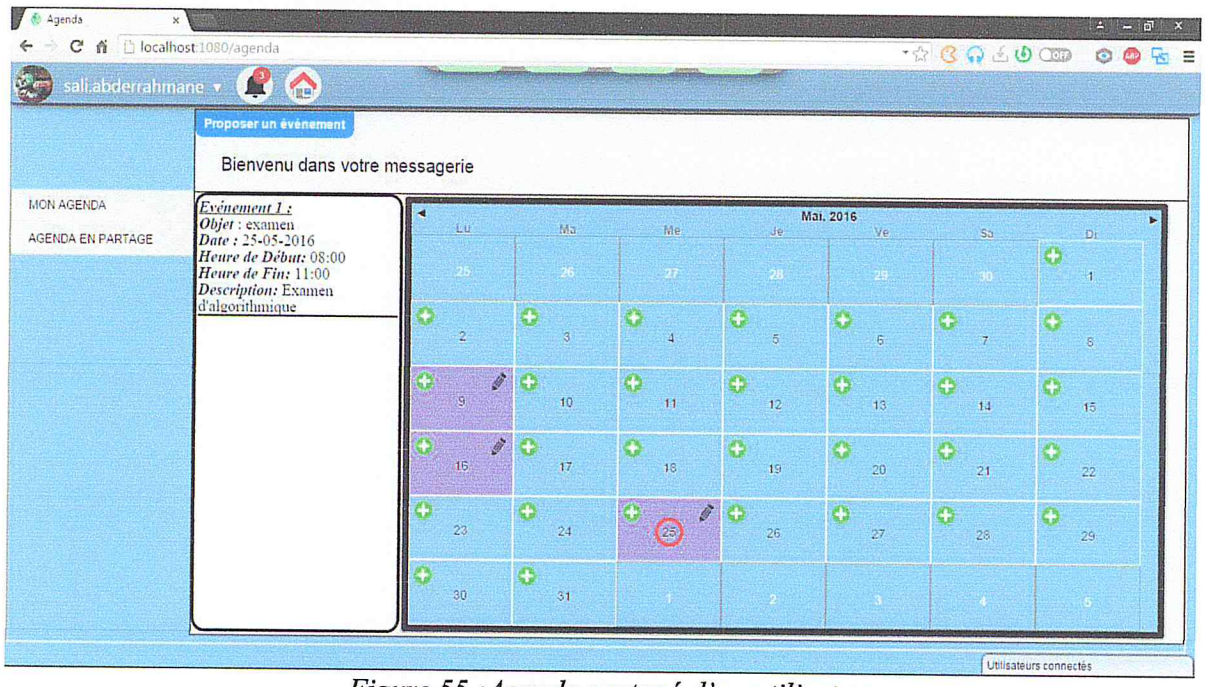

Figure 55 :Agenda partagé d'un utilisateur.

La Figure 55 illustre l'agenda d'un utilisateur : cette page comporte un calendrier et un panneau d'affichage, l'utilisateur peut créer un nouvel événement en cliquant sur le bouton « + »

de la date qu'il souhaite ; il peut aussi consulter les événements d'une date en cliquant sur cette date. Le panneau d'affichage permet de lire les détails des événements d'une journée. Les dates ayant des événements sont affichées en couleur mauve pour faciliter la distinction entre les dates.

L'utilisateur peut aussi consulter les agendas des utilisateurs qui mettent leurs agendas en public, il peut aussi proposer un événement à un autre utilisateur.

6.4.3. Espace de stockage

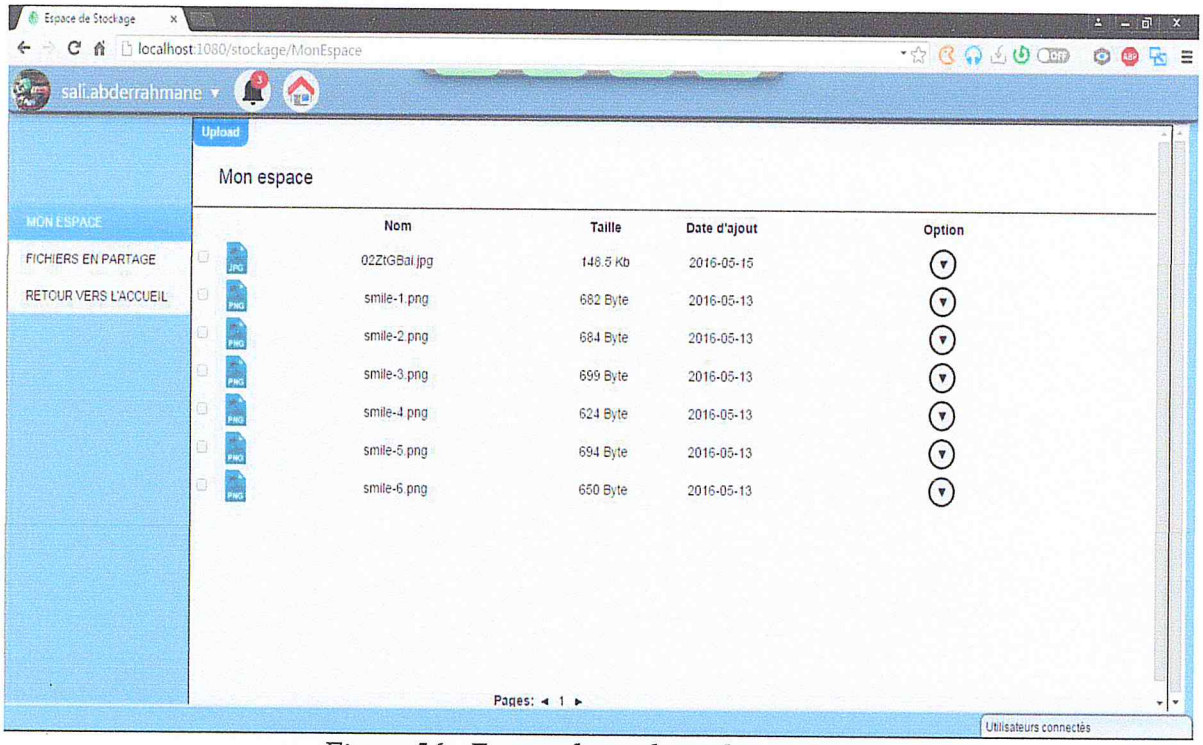

Figure 56 : Espace de stockage d'un utilisateur.

La Figure 56 présente l'espace de stockage d'un utilisateur : un utilisateur peut stocker des fichiers dans son espace en les mettant en public ou en privé, il peut aussi rechercher des fichiers qui sont publics en consultant la page « fichiers en partage » et les télécharger.

6.4.4. Edition UML

6.4.4.1. Création de projet

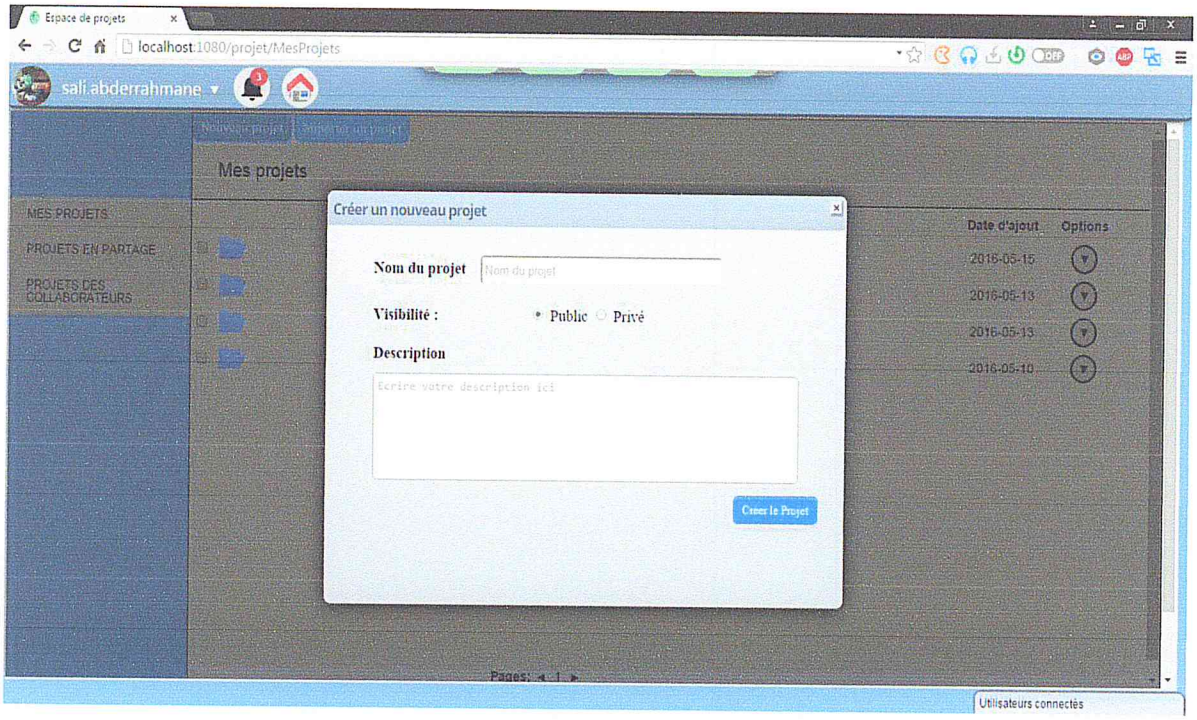

Figure 57: Création d'un nouveau projet

La Figure 57 présente la boite de dialogue de création d'un nouveau projet : l'utilisateur (futur chef de projet) doit indiquer le nom et la visibilité de son projet, un projet public implique la possibilité de le retrouver en cas de recherche par d'autre utilisateurs, un projet privé implique que seuls le chef de projet et les membres peuvent le retrouver.

6.4.4.2. Liste de projets

La Figure 58 présente la liste des projets d'un utilisateur, cet utilisateur (chef de projet) peut modifier les informations de ses projets en cliquant sur le bouton des options.

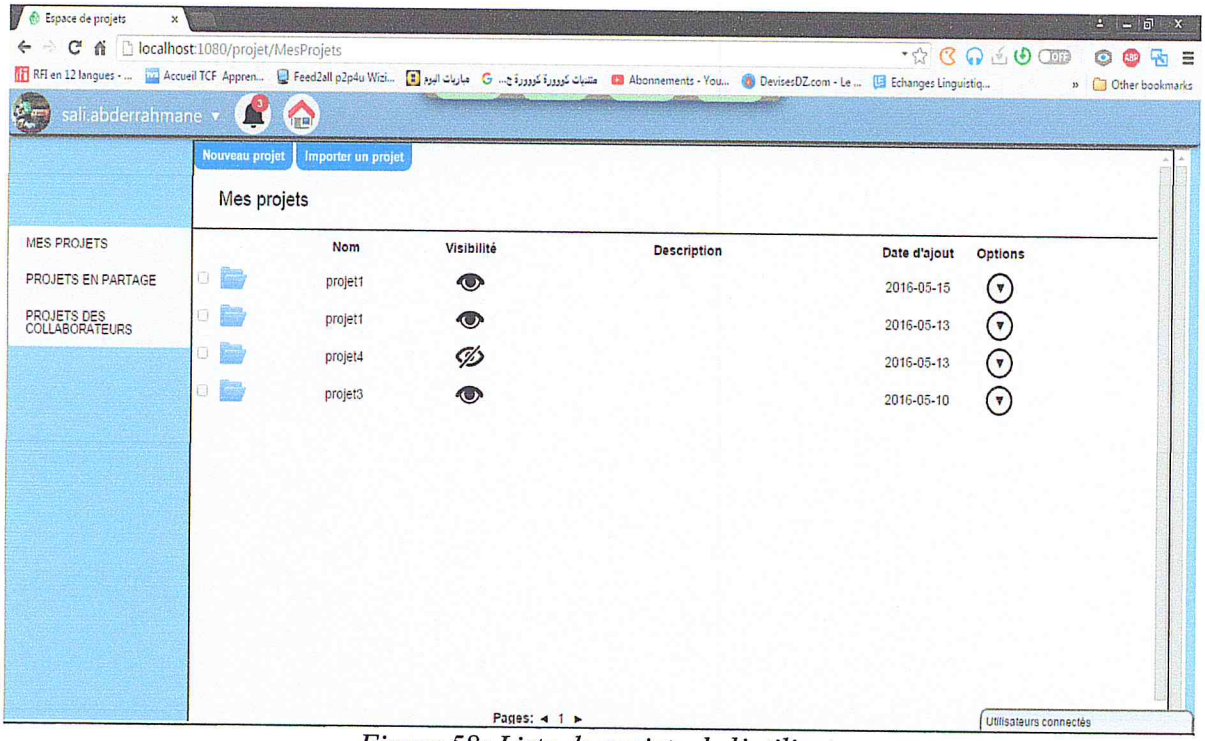

Figure 58: Liste de projets de l'utilisateur

6.4.4.3. Edition de diagramme

Les trois figures suivantes présentent l'éditeur de diagrammes UML. La plateforme dispose d'une palette d'outils qui permet une édition aisée des diagrammes de classe (Figure 59), de cas d'utilisation (Figure 60) et de séquence (Figure 61).

Si l'utilisateur est le chef de projet, il peut gérer les membres (ajouter, retirer, attribuer les droits de modification, …), sinon (s'il est membre) il peut visualiser les diagrammes s'il dispose des droits de lecture de diagramme, modifier s'il dispose des droits d'écriture.

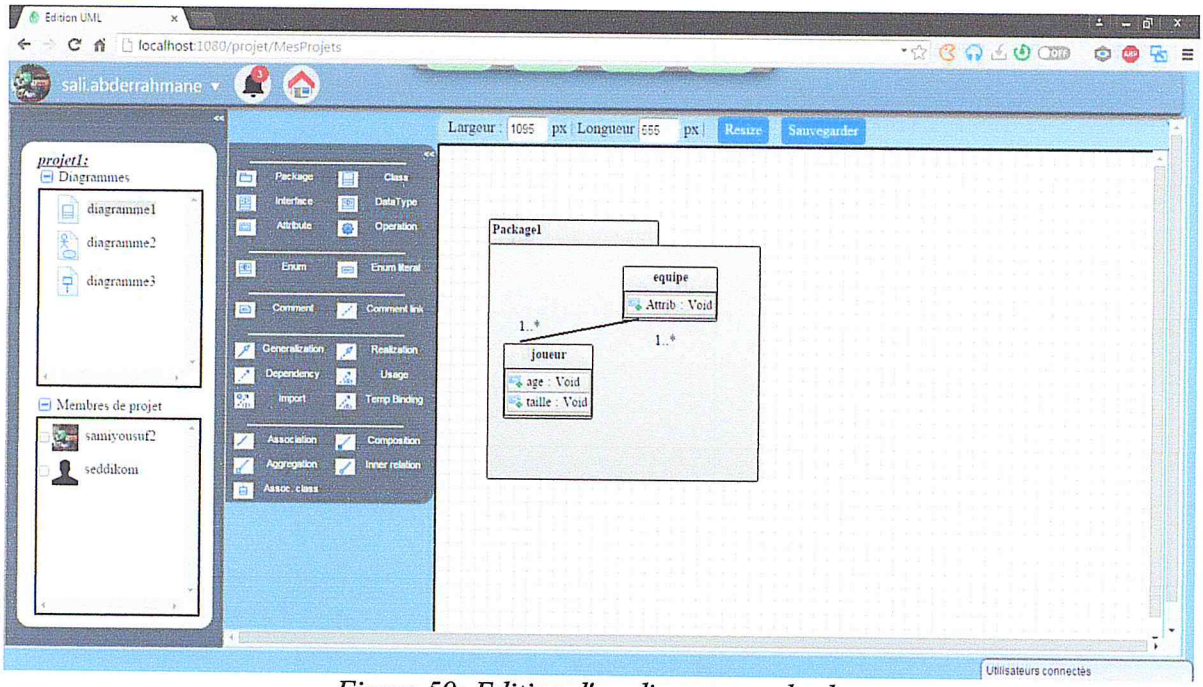

Figure 59: Edition d'un diagramme de classe.

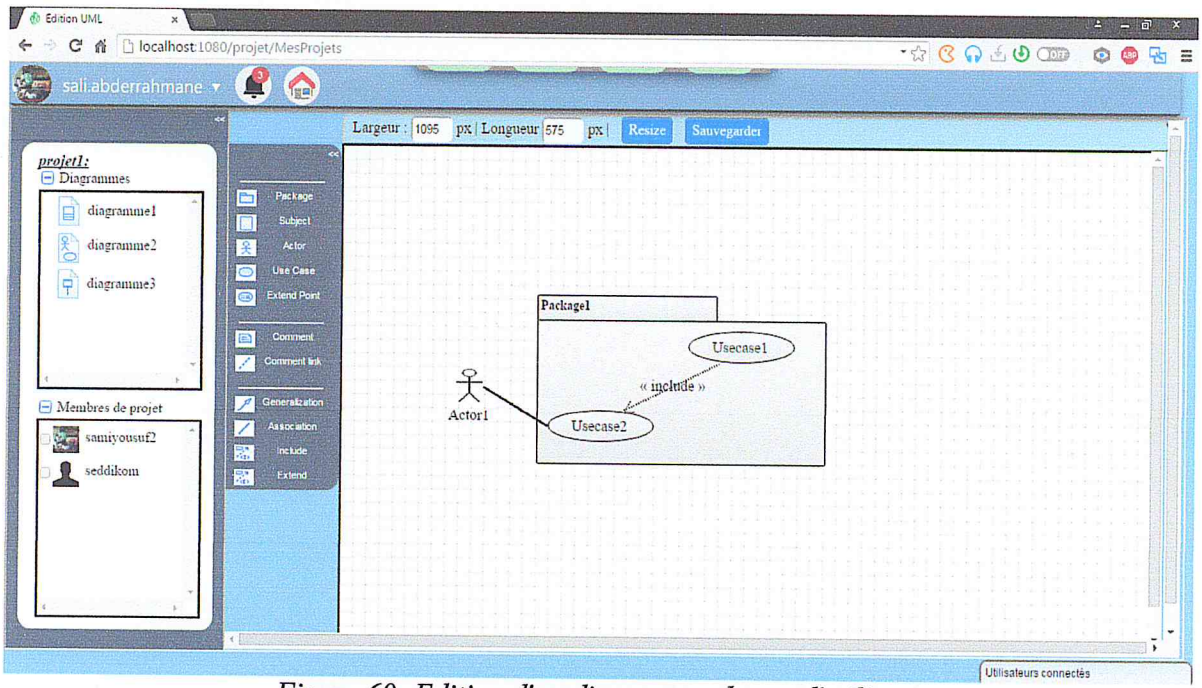

Figure 60: Edition d'un diagramme de cas d'utilisation.

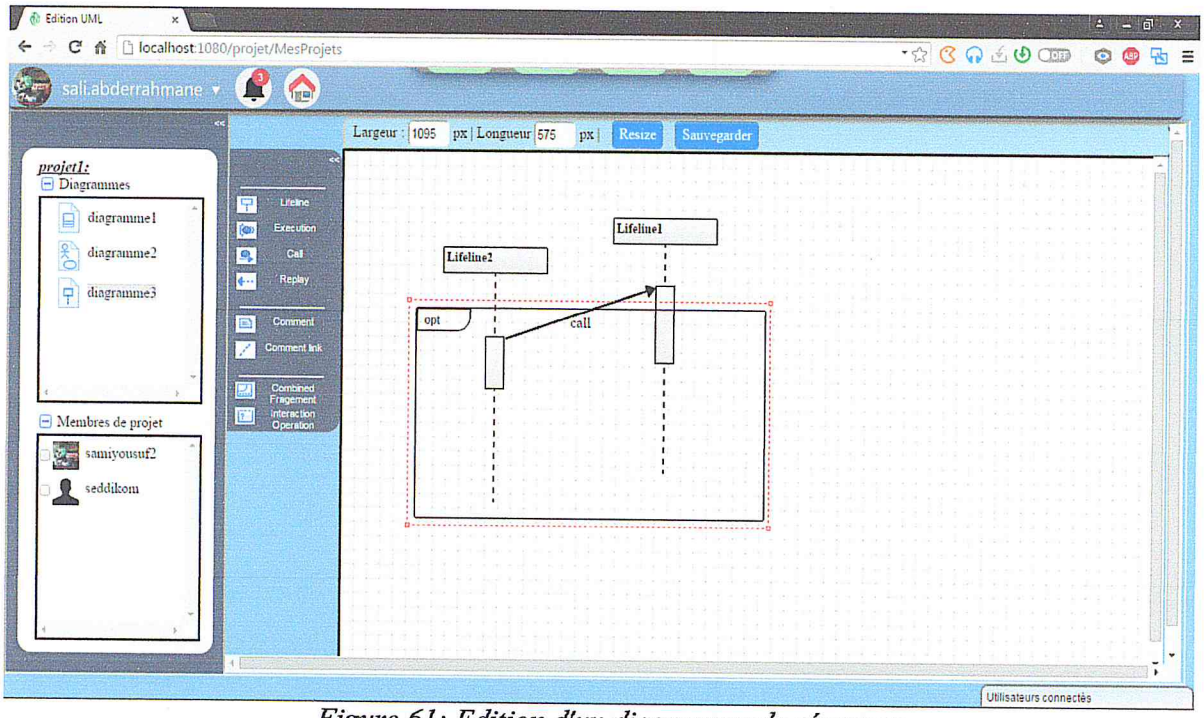

Figure 61: Edition d'un diagramme de séquence.

7. Conclusion

Nous avons vu dans ce chapitre l'architecture réseau de notre plateforme, les environnements logiciels sur lesquels s'est basé notre travail, ainsi que les langages d'implémentation utilisés. Nous avons également justifié le choix de l'utilisation de certains modules JavaScript.

Enfin nous avons présenté quelques captures d'écran de notre plateforme de travail collaboratif en illustrant les différents outils qu'elle contient.

Conclusion générale

Favoriser la collaboration entre les employés est un enjeu de plus en plus important pour les entreprises. Les outils de travail collaboratif basés sur les nouvelles technologies de l'information et de la communication ont été développés dans le but de favoriser cette collaboration.

L'objectif principal de notre projet a été de développer une plateforme de travail collaboratif en utilisant des technologies récentes dites web temps réel. Ces technologies incluent entre autres la plateforme logicielle Node.js pour l'exécution rapide des programmes JavaScript côté serveur, l'API WebSockets pour l'échange bilatéral synchrone entre le client et le serveur et l'API WebRTC pour la communication Peer-to Peer entre deux navigateurs.

Pour la conception de la plateforme collaborative, nous avons utilisé l'extension WAE (Web Application Extension) du langage UML qui permet de prendre en compte les spécificités des applications web.

Pour la réalisation, nous avons choisi d'utiliser HTML et JavaScript pour le côté client ; quant à la partie serveur nous avons utilisé Node.js comme serveur d'application ainsi que d'autres modules tels que express, socket.io et easyrtc.

Bien que notre plateforme de travail collaboratif ne soit qu'à sa première version, elle est néanmoins totalement fonctionnelle et offre un certain nombre de fonctionnalités intéressantes telles que :

- ✓ La communication (texte, audio et vidéo) et le partage de ressources entre utilisateurs en temps réel.
- ✓ La possibilité d'envoyer et de recevoir des messages hors ligne.
- ✓ La possibilité de créer et de proposer des événements (agenda partagé).
- ✓ La possibilité de stocker et de partager des fichiers.
- ✓ L'édition collaborative de diagrammes UML en temps réel entre les membres d'un projet.

Dans la prochaine version de notre plateforme, nous envisageons de rajouter les fonctionnalités suivantes :

- L'ajout d'autres outils collaboratifs tel que l'éditeur collaboratif de texte pour réaliser les spécifications textuelles des diagrammes, un outil de gestion des tâches, un forum

pour partager les idées et les techniques de modélisation à travers des articles, une actualité pour partager les dernières nouveautés , etc.

- L'amélioration de l'éditeur UML en rajoutant la possibilité de modéliser d'autres types de diagrammes, un système de gestion de versions pour les diagrammes ainsi qu'un outil de génération de code source.

Enfin, ce travail nous a permis non seulement de nous initier aux nouvelles technologies web mais aussi à approfondir nos connaissances dans le domaine de la programmation web et de conforter nos connaissances en conception des plateforme collaborative.

Références

[Bal, 05] G. Balmisse. *Guide des outils du knowledge management: panorama, choix et mise en oeuvre*. Collection Entreprendre informatique. Vuibert, (2005).

[Ben, 11] Ben Diaf, Zouhour, *Résumé du cours travail collaboratif et collecticiel*, institut supérieur d'informatiques et mathématiques, Monastir (2011).

[Con, 02] Jim Conallen. *Building Web applications with UML*. Addison-Wesley Longman Publishing Co., Inc., (2002), p 236-239.

[DF, 93] P.F. Drucker and J. Fontaine. *Au-delà du capitalisme: la métamorphose de cette fin de siècle*. Dunod, (1993).

[Dug, 05] Philippe Dugerdil. *Impact des décisions informatiques: introduction à l'informatique pour le décideur non-informaticien*, volume 18. PPUR presses polytechniques, (2005), p 57.

[GG, 08] Joseph Gabay and David Gabay. *UML 2 Analyse et conception-Mise en oeuvre guidée avec études de cas: Mise en oeuvre guidée avec études de cas*. Dunod, (2008), p 113-116 .

[HLC, 01] France Henri and Karin Lundgren-Cayrol. *Apprentissage collaboratif à distance: pour comprendre et concevoir les environnements d'apprentissage virtuels*. Puq, (2001).

[Hon, 09] Olivier Hondermarck. *JavaScript: Le guide complet*. MA éditions, (2009).

[Jak, 15] Christer Jakobsson. Peer-to-peer communication in web browsers using webrtc. 2015.

[JB, 12] Alan B Johnston and Daniel C Burnett. *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC, (2012).

[LDK, 05] Mylène LEITZELMAN, Henri DOU, and Jacky KISTER. Système dâ€™information de travail collaboratif assisté par ordinateur implémenté dans une umr de recherche pluridisciplinaire et multi-sites. In *Veille stratégique, scientifique et technologique*, (2005).

[MMGE, 99] Bertrand Moingeon, Emmanuel Métais, HEC Groupe, and Groupe EDHEC. *Stratégie de rupture basée sur des innovations radicales: Etude du cas de l'entreprise Salomon à la lumière de ses compétences et capacités organisationnelles*. Groupe HEC, (1999).

[Neb, 13] Mathieu Nebra. *Des applications ultra-rapides avec Node.js*. Simple IT, (2013).

[Sou, 94] S Soubbarayer. Les enjeux du workflow. In *Actes de conférences IT FORUM*, volume 94, pages 8–11, (1994).

[WT, 09] Luke Welling and Laura Thomson. *PHP et MySQL*. Pearson Education France, (2009).

Sites web:

[IBMKC] IBM Knowledge Center, http://www.ibm.com/support/knowledgecenter/SS6RBX_11.4.3/com.ibm.sa.oomethod.doc/topics/c_Web_app_Extensions_WAE.html.