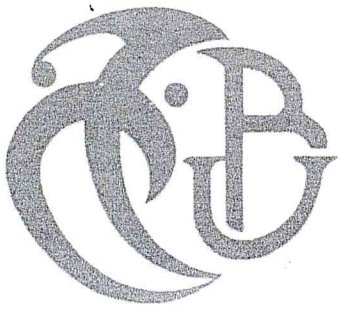
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Saad Dahlab Blida

N° D'ordre :.....



Faculté des sciences

Département d'informatique

Mémoire Présenté par :

KERDJADJ Abdelhamid ZIANE Islam Abdelkader

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Sujet : Conception et réalisation d'une application web pour le partage de fichiers en peer 2 peer

Soutenu le : 24 Juin 2015

Devant le jury :

M. O. Hadj-Yahia	Président
M. N. Tamboline	Examinateur
M. S. Arroussi	Examinateur
M. CHIKHI Nacim Fateh	Promoteur

Promotion
2014 / 2015

Remerciements

Avant tout, nous rendons grâce à Dieu qui nous a donné la force et la patience pour terminer nos études.

Nous remercions aussi tous nos professeurs pour le savoir qu'ils nous ont transmis.

Nous remercions plus particulièrement Monsieur Chikhi Nacim Fateh, notre promoteur, qui nous a grandement aidés pour la réalisation de ce document et mener à bien le travail qui nous a été confié.

Nous remercions aussi nos familles qui se sont sacrifiées pour que nous puissions réussir nos études, sans oublier nos amis qui nous ont encouragés.

Dédicace

Sans elle je ne suis rien, je n'aurai rien réussi ni rien accompli, je l'aime de tout mon cœur, car elle m'a non seulement mis au monde, mais elle a fait de moi l'homme que je suis, elle n'a jamais cessé de croire en moi, à mes capacités et à mon courage, pour l'être le plus chère dans ma vie, ***maman*** *je te dédie ce travail pour t'honorer et te remercier du plus profond de mon cœur.*

Je dédie ce travail aussi à mon ***père****, qui m'a tant appris et tant aidé et ne m'a jamais refusé quoi que ce soit durant toute ma période de scolarisation.*

A mes deux frères que m'a mère n'a pas mis au monde, mes camarades et mes complices depuis mon enfance, ***Akram Bouaklin*** *et* ***Mustapha Allali****.*

A mes deux grandes sœurs, ***Amel*** *et* ***Hanane*** *qui m'ont toujours bien conseillées, et dirigées vers mon bien, et sans oublier leurs maris* ***Farouk*** *et* ***Sidali*** *et leurs petits bouts de chou,* ***Yasmine*** *et* ***Mohamed****.*

A mes deux petites sœurs ***Sarah*** *et* ***Kamelia*** *qui ont contribuées à ma réussite avec leurs encouragements et amour.*

A mon binôme ***Kerdjadj Abdelhamid****, qui m'a accompagné dans cette aventure.*

Votre Fils, Frère, Ami et Camarade ***ZIANE Islam Abdelkader****.*

Dédicace

Par la grâce de Dieu je viens de terminer mon projet de fin d'étude.

Que je dédie :

A mes parents, qui grâce à leurs encouragements et leurs grands sacrifices, ils ont pu créer le climat affectueux et propice à la poursuite de mes études.

Aucune dédicace ne pourrait exprimer mon respect, ma considération et mes profonds sentiments envers eux.

A mes frères Riad et Fayçal et leurs femmes.

A mes sœurs Khadîdja, Aicha et Meriem et à leurs maris.

A tous mes neveux qui sont la joie de notre petite famille.

A ma deuxième mère ; ma tante Zineb et ses enfants.

A mes frères et amis Syphax, Yacine, Chouaib, Abderrahmane, Hamid et Khaled qui m'ont aidé et encouragé.

A mon frères et binôme Ziane Islam Abdelkader qui m'a encouragé et aidé pour faire ce travail.

Abdelhamid.

Table des matières

Liste des figures

ملخص

مع تطور الويب 2.0 في السنوات الاخيرة، ومع ظهور تقنيات الويب في الوقت الحقيقي، أصبح من الممكن إنشاء تطبيقات ويب افضل من البرامج "الكلاسيكية" التي اعتدنا استخدامها في أجهزة الكمبيوتر. وفي هذا السياق فأن العمل الموكل لنا في الماستر هو تطوير تطبيق ويب لتبادل الملفات في الند للند. و من شأن هذا التطبيق تجنب أن يضطر المستخدم إلى تثبيت برامج على أجهزة الكمبيوتر الخاصة به، في حين يقدم ميزات افضل من تلك التي يقدمها برامج الند للند التقليدية. على عكس المناهج الماضية التي تتطلب تركيب الإضافات محددة لتكون قادرة على الاستفادة من الاتصالات الند للند بين اثنين من المتصفحات ويب، وEasyP2P تطبيق الذي انجزناه باستخدام تقنيات الويب في الوقت الحقيقي التي أدرجت مؤخرا في متصفحات الويب.

لتصميم وانشاء EasyP2P، استخدمنا تمديد WAE ل UMLالذي يأخذ بعين الاعتبار خصوصيات تطبيقات الويب. من أجل انشاء اخترنا حل يستند على JavaScript استخدام هذه اللغة ينتج لنا مدونة خفيفة إما على جانب العميل او الخادم وبالتالي ضمان أداء عالي لتطبيق. لتدقيق اكثر، قد استخدمنا Node.jsلإنشاء الخادم، المكتبة socket.ioلاستخدام Websocketو أخيرا مكتبة EasyRTC للاتصال الند للند

Résume

Avec le développement du web 2.0 ces dernières années, et plus récemment l'avènement des technologies web temps réel, il est désormais possible de créer des applications web aussi performantes que les logiciels "classiques" que nous avons tous l'habitude d'utiliser sur nos ordinateurs. C'est dans cette optique que s'inscrit ce travail de Master dont l'objectif est de développer une application web pour le partage de fichiers en peer-to-peer. Une telle application permettrait d'éviter à l'utilisateur d'avoir à installer un logiciel sur son ordinateur, tout en lui offrant des fonctionnalités aussi avancées que celles proposées par les logiciels classiques de peer-to-peer. En effet, contrairement aux anciennes approches qui nécessitaient l'installation de plugins spécifiques pour pouvoir bénéficier d'une communication peer-to-peer entre deux navigateurs, l'application web EasyP2P que nous avons réalisée utilise les technologies web temps réel incorporées récemment dans les navigateurs web.

Pour la conception de l'application EasyP2P, 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 opté pour une solution basée sur JavaScript. L'utilisation de ce langage permet d'avoir un code léger que ce soit du côté client et du côté serveur assurant ainsi de hautes performances pour notre application web. Plus précisément, nous avons utilisé Node.js pour la mise en œuvre du serveur, la librairie socket.io pour l'utilisation des websockets et enfin la bibliothèque EasyRTC pour la communication peer-to-peer.

Mots clés: peer to peer, web socket, WebRTC, node.js, architecture p2p centralisée, application web

Abstract

With the development of web 2.0 these last years, and more recently the advent of real-time web technologies, it is now possible to create web applications as powerful as the "classic" software that we all used to use on our computers. It is in this light that this work is part of Master whose goal is to develop a web application for file sharing in peer-to-peer. Such an application would avoid the user having to install software on their computer, while offering advanced features such as those offered by traditional software peer-to-peer. Unlike past approaches that required the installation of specific plugins to be able to benefit from a peer-to-peer communication between two browsers, the web application EasyP2P we realized using the real-time web technologies embedded recently in web browsers.

For the design of the EasyP2P application, we used the WAE extension (Web Application Extension) of UML, which allows taking into account the specific web applications. For the implementation, we opted for a solution based on JavaScript. The use of this language allows for a lightweight code either on the client side or on server side thus ensuring high performance for our web application. Specifically, we used Node.js for the implementation of the server, the library socket.io for using Websockets and finally EasyRTC library for peer-to-peer communication.

Introduction générale

En quelques années seulement, internet a connu un véritable développement, offrant aux utilisateurs du monde entier un moyen rapide et efficace pour partager des informations. L'augmentation considérable du nombre de transferts de fichiers est aujourd'hui de plus en plus importante et la taille des fichiers transférés a également été multipliée ces dernières années. Il n'est plus rare aujourd'hui de voir des échanges de plusieurs Giga octets de données sur internet.

Cet accroissement des échanges montre aujourd'hui les limites du vieux modèle client/serveur : Malgré une augmentation considérable de leur puissance de calcul, les serveurs ont de plus en plus de difficultés à répondre aux demandes de tous les clients. C'est ainsi que l'on s'oriente de plus en plus vers des architectures N-tiers pour tenter de répartir la charge.

D'un autre côté, les réseaux d'échange de fichiers ont connu ces dernières années un engouement important de la part des internautes. Ces échanges, dits "Peer-to-Peer", représentent aujourd'hui une partie considérable des échanges sur internet. Leur développement est lié en grande partie à l'augmentation des capacités matérielles des internautes, notamment en termes de bande passante. En effet, dans une architecture Peer-to-peer, les ressources de toutes les machines connectées au réseau sont partagées entre celles-ci. Le peer-to-peer intéresse aujourd'hui même les grandes entreprises. A titre d'exemple, Red Hat utilise le Peer-to-Peer pour distribuer ses produits dans le but de soulager ses serveurs web.

Avec le développement du web 2.0 ces dernières années, et plus récemment l'avènement des technologies web temps réel, il est désormais possible de créer des applications web aussi performantes que les logiciels "classiques" que nous avons tous l'habitude d'utiliser sur nos ordinateurs. C'est dans cette optique que s'inscrit ce travail de Master dont l'objectif est de développer une application web pour le partage de fichiers en peer-to-peer. Une telle application permettrait d'éviter à l'utilisateur d'avoir à installer un logiciel sur son ordinateur, tout en lui offrant des fonctionnalités aussi avancées que celles proposées par les logiciels classiques de peer-to-peer.

CHAPITRE I :

Le partage de fichiers en peer-to-peer

1. Introduction :

Le peer-to-peet (P2P), souvent considéré comme remplacent de l'ancienne architecture Client/Serveur, a connu depuis ces dernières années une véritable révolution. Beaucoup de logiciels et d'applications de partage de fichier en peer-to-peer ont ainsi vu le jour. La recherche dans ce domaine ne cesse d'avancer notamment dans le but d'améliorer les échanges ou encore la confidentialité des données échangées.

Dans ce chapitre, nous étudions ce modèle d'échange, son historique, ses avantages et inconvénients ainsi que les logiciels peer-to-peer les plus utilisés.

2. Définitions

Définition 1 :

Le peer-to-peer est un modèle de réseau dans lequel chaque machine fait à la fois office de serveur et de client. Les éléments composant le réseau sont appelés des nœuds [1].

Définition 2 :

Peer-to-peer signifie littéralement pair à pair. Ce concept introduit ainsi une relation d'égal à égal entre deux ordinateurs.

Dans son essence, l'informatique pair à pair se définit comme le partage des ressources et des services par échange direct entre systèmes. Ces échanges peuvent porter sur les informations, les cycles de traitement, la mémoire cache ou encore le stockage sur disque des fichiers.

Contrairement au modèle client/serveur, chaque système est une entité réseau complète qui remplit à la fois le rôle de serveur et celui de client en même temps. Avec le peer-to-peer, les ordinateurs personnels ont le droit de faire partie du réseau **[2].**

3. Historique :

3.1. Le projet Napster

A la fin des années 90, Shawn Fanning, un étudiant américain passionné d'informatique alors âgé de 19 ans vient de bouleverser le monde bien établi du client/serveur. Il décide de quitter l'université et se lance dans l'écriture d'un logiciel pour permettre l'échange de fichiers musicaux.

Après quelques mois de travail acharné, une première version du logiciel est disponible. Fanning décide de tester une première version le 1er juin 1999 et appelle son logiciel Napster.

Le logiciel qui ne devait être testé que par quelques-uns de ses amis remporte un succès des plus rapides. Il conquiert notamment les universités. Shawn Fanning se retrouve propulsé à la tête d'une start-up pleine d'avenir. Lui, qui déclare à propos de Napster qu'il n'avait "aucune envie d'en faire un busines", voit les utilisateurs arriver en masse. En septembre 2000, Napster atteint un nombre de téléchargements record. 1,39 milliard serait le nombre de chansons échangées par ses utilisateurs (source : Webnoize, Cabinet d'études américain) **[2].**

3.2. La fin de Napster

Tombent alors les premières interdictions de la part des universités : les étudiants l'utilisent tellement qu'ils saturent les bandes passantes. Les groupes de musique demandent à ce qu'on protège leurs droits. Le groupe Metallica ouvre le bal et entame un procès ; en décembre 1999 le RIAA intente également un procès à Napster. En novembre 2000, il est prévu que Napster intègre un système anti-piratage, mais en janvier 2001, le verdict de la 9ème cour d'appel de San Francisco tombe : Napster viole la loi sur les droits d'auteurs et devra cesser dans un bref délai l'échange gratuit de fichiers musicaux MP3. C'est une victoire importante pour les maisons de disques, même si elles n'obtiennent pas la fermeture immédiate de Napster. En février 2001, un système de cotisations est mis en place. Napster échappe aux procès des maisons de disques mais les utilisateurs le délaissent.

4. Architecture des réseaux Peer-to-Peer :

Il existe trois types d'architectures distinctes pour les réseaux « P2P » : le réseau centralisé, le réseau distribué et enfin le réseau hybride qui, comme son nom l'indique, combine les deux première formes de réseau.

- ***L'architecture centralisée***

Dans cette architecture (Figure I.1), un client (un logiciel utilisé par les membres) se connecte à un ou plusieurs serveurs qui gèrent les partages, la recherche, et l'insertion d'informations, bien que celles-ci transitent directement d'un utilisateur à l'autre. Toutes ces informations sont collectées à l'aide d'un système d'indexation appelé la « table de hachage distribuée » qui permet en théorie d'éviter la multiplication de fichiers inutiles.

L'avantage de l'architecture centralisée est bien sûr d'avoir une vue globale et très complète du réseau, à condition que le nombre d'utilisateurs connectés soit limité pour ne pas se retrouver submergé par une liste interminable d'informations si l'on ne recherche rien de particulier. Mais dans le cas où la recherche de fichiers est ciblée, le réseau centralisé se montre très rapide. Le serveur reçoit la requête, indique quel utilisateur est susceptible d'héberger le fichier recherché, puis établit une connexion directe avec le (ou les) PC concerné. À aucun moment le fichier ne passe par le serveur central. Malheureusement, le réseau centralisé est très vulnérable. En cas d'arrêt du serveur, c'est le réseau tout entier qui disparaît **[3].**

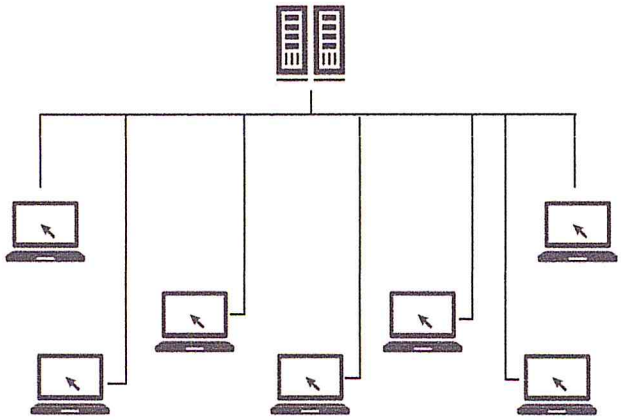


Figure I.1: Réseau peer-to-peer centralisé

- ***L'architecture distribuée ou décentralisée***

Beaucoup moins vulnérable, le réseau distribué (Figure I.2) n'utilise pour sa part aucun serveur, mais continue de s'appuyer sur les tables de hachage distribuées. Chaque utilisateur est à la fois un client et un serveur (en anglais "servant"), et tous se découvrent et se connectent dynamiquement entre eux en relayant les requêtes d'ordinateur à ordinateur. Si cette architecture présente l'avantage d'être robuste, elle génère en revanche un trafic important, et la recherche de fichier prendra plus de temps. Chaque requête sera adressée à chaque utilisateur connecté, chacun d'entre eux faisant de même, ce qui peut prendre un temps considérable si des milliers d'utilisateurs sont présents. De même, l'absence de serveur oblige chaque pair à prendre en charge une partie du rôle du serveur et donc une partie de la table de hachage. Mais dans le cas où plusieurs pairs viendraient à disparaître, une partie de l'annuaire recensant les pairs et toutes les ressources partagées disparaîtraient également **[3].**

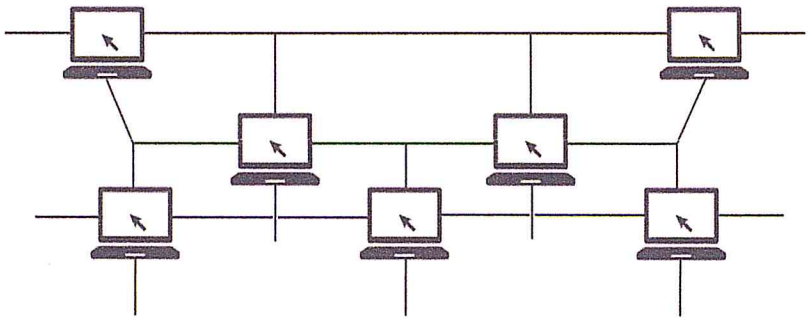


Figure I.2: Réseau peer-to-peer décentralisé

- ***Le réseau hybride ou semi-centralisé***

Le troisième type d'architecture, le réseau hybride (Figure I.3) est le plus complexe à mettre en œuvre et combine à la fois le réseau centralisé et distribué. Le réseau de ce type s'appuie sur un ensemble de serveurs (l'ordinateur d'un utilisateur peut devenir un serveur) gérant un groupe d'utilisateurs suivant l'architecture centralisée. Puis, chaque serveur est ensuite connecté à d'autres serveurs suivant l'architecture distribuée. De cette façon, si un fichier recherché par un utilisateur n'est pas indexé par le serveur auquel il est rattaché, celui-ci transmet alors la requête à un autre serveur. Cette architecture permet de bénéficier d'une meilleure bande passante en réduisant le trafic de requêtes **[3].**

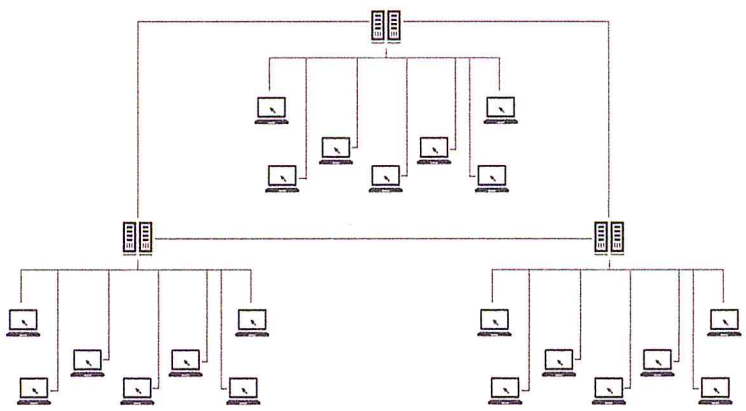


Figure I.3: Réseau peer-to-peer semi-centralisé

5. Caractéristiques des réseaux peer-to-peer :

Les principales caractéristiques du réseau pair à pair sont **[4]** :

– Localisation des fichiers dans un environnement distribué

– Métadonnées ou index du réseau P2P

– Libre circulation des fichiers entre les pairs

– Capacité de connexion variable suivant les pairs

– Echanges d'information non sécurisés

– Certains pairs peuvent être non fiables

– Aucune vue globale du système.

6. Quelques logiciels pair à pair :

- **NAPSTER :** est une application de partage de fichiers musicaux mp3. Le service est fondé en juin 1999 par Shawn Fanning.
- **Gnutella** : est un protocole de partage de fichiers qui utilise l'architecture décentralisée. La première version (version 0.4) date de mars 2000 et fut développée en une quinzaine de jours par Justin Frankel et Tom Pepper. Les applications qui implémentent le protocole Gnutella autorisent les utilisateurs à rechercher et charger des fichiers sur tous les autres utilisateurs connectés à la communauté Gnutella, avec plusieurs dizaines de milliers d'utilisateurs simultanés. Il existe plusieurs implémentations compatibles apportant des extensions telles que : Limewire, ToadNode, BearShare.
- **eDonkey :** est né en septembre 2000 mais c'est au cours de l'année 2001 qu'il a connu son véritable essor. Son efficacité est dûe au fait qu'un client peut transférer un fichier à partir de la machine d'un client qui, lui aussi, transfère au même moment le même fichier à partir d'une autre machine client. Comme plusieurs autres, ce produit est hybride, il est décentralisé et l'architecture est proche du super peer. Mais son application en elle-même fonctionne selon un principe centralisé.
- **DirectConnect** : est original de par son fonctionnement. Les serveurs sont ici appelés des hubs (une traduction possible est "salon"). Chaque hub est créé pour accueillir une catégorie de fichiers ainsi qu'une véritable communauté autour d'un sujet précis.

CHAPITRE II :

Le web temps réel

1. Introduction

Ces dernières années le web a connu un développement grâce aux technologies web temps réel comme Ajax et node.js ou encore les protocoles WebSocket et WebRTC. Ce chapitre est consacré à ces technologies, leur définition, leur principe de fonctionnement et une comparaison entre les applications web classiques et les applications web qui utilisent ces technologies. Nous terminons enfin par une présentation de quelques API de WebRTC.

2. Web temps réel

2.1. Définition

Le « Web temps Réel » est l'ensemble des informations envoyées sur le Web par des personnes de manière instantanée. Ces informations sont envoyées à un groupe de destinataires, publiées sur le Web et analysables par des logiciels de traitement de l'information [7].

3. JavaScript

3.1. Définition

JavaScript est un langage de script, distinct du langage HTML ; JavaScript est en quelque sorte un petit langage de programmation dont les lignes de code viennent s'ajouter au langage HTML pour agrémenter la présentation et ajouter de l'interactivité aux pages web. Ce langage est inspiré plus ou moins du langage C [8].

3.2. Historique :

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

- ✓ Dans les années 90, on parlait de DHTML (Dynamic HTML). On utilisait en fait les toutes premières versions de JavaScript pour faire des petits effets dans ses pages web. C'était l'époque de Netscape et d'Internet Explorer 5.5.
- ✓ Dans les années 2000, on a commencé à utiliser le langage pour créer des interfaces côté client. C'est là que des bibliothèques comme jQuery ou Mootools sont apparues. Aujourd'hui, cet usage de JavaScript est très répandu et mature.
- ✓ Puis, aux alentours de 2010, JavaScript est entré dans une nouvelle ère. Google a commencé à rendre le langage beaucoup plus rapide avec l'apparition du navigateur Google Chrome. Avec ce navigateur est né le moteur d'exécution V8 qui a considérablement permis d'accélérer l'exécution de code JavaScript. Des outils comme Node.js sont ensuite apparus. Les bibliothèques dont le nom finit par .js se sont alors multipliées : Backbone.js, Ember.js, Meteor.js [9].

4. Asynchronous JavaScript and XML « AJAX »

4.1.Definition

Asynchronous JavaScript and XML est un acronyme qui représente un ensemble de technologies permettant l'échange d'informations entre le navigateur et le serveur web sans recharger l'ensemble de la page web ouverte. Ces échanges s'effectuent de façon asynchrone à l'aide des fonctions développées en langage JavaScript. Il ne s'agit donc pas d'une technologie proprement dite mais de l'utilisation conjointe de technologies déjà existantes et courantes sur le web [10].

4.2. Principe de fonctionnement

Au début de l'année 2005, Jesse James Garrett dans son article « Ajax, A new approche to web application » décrit le fonctionnement d'Ajax :

« Ajax n'est pas une technologie, il s'agit de plusieurs technologies se développant chacune de leur côté et combinées pour donner des résultats aussi nouveaux que puissants, Ajax comporte :

- Une présentation fondée sur les standards XHTML et CSS.
- Un affichage dynamique et interactif grâce à DOM.
- Un système d'échange et de manipulation de données utilisant XML et XSLT mais aussi JSON.
- Un mécanisme de récupération de données asynchrones utilisant XMLHttpRequest.
- JavaScript pour lier le tout. » [10]

La figure II.1 montre le principe d'Ajax selon la définition de Jesse James Garrett.

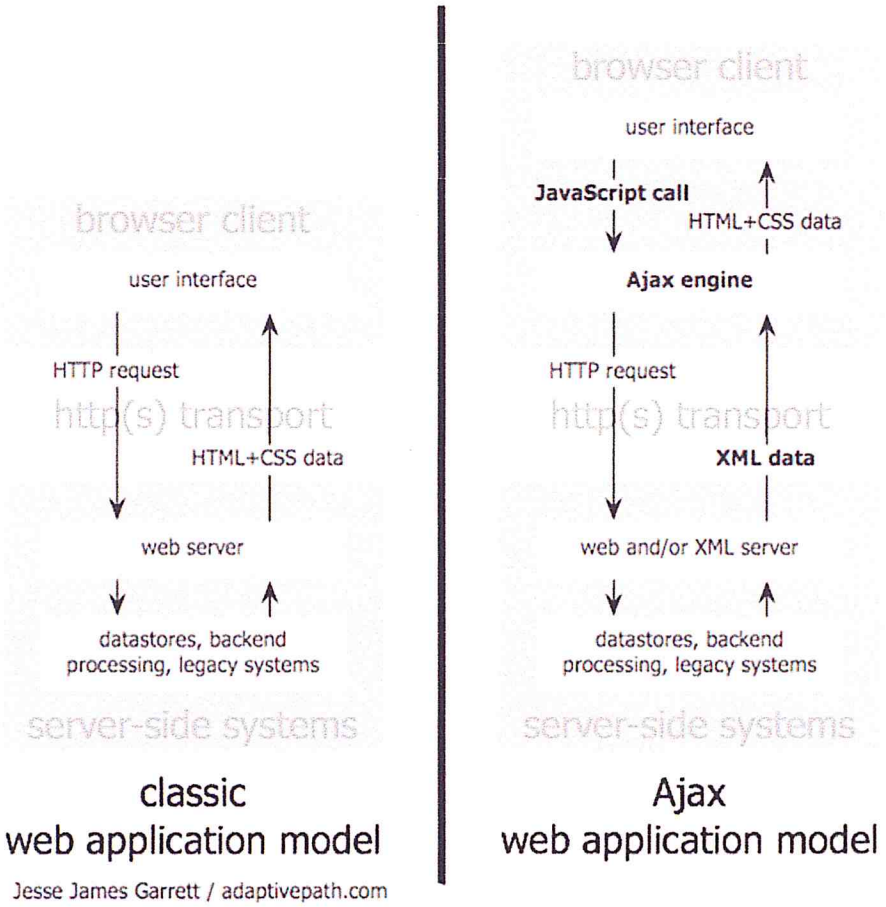


Figure II.1 : Principe de fonctionnement d'une application Ajax [11]

4.3. Différence entre une application web classique et une application web utilisant Ajax

La figure II.2 ci-dessous montre la différence entre le modèle classique et le modèle Ajax. Dans le modèle classique, le client lance une requête et attend qu'elle soit traitée pour qu'il puisse en faire une autre. On appelle cela le modèle synchrone ; par contre Ajax évite toute cette perte de temps puisqu'il utilise le modèle asynchrone.

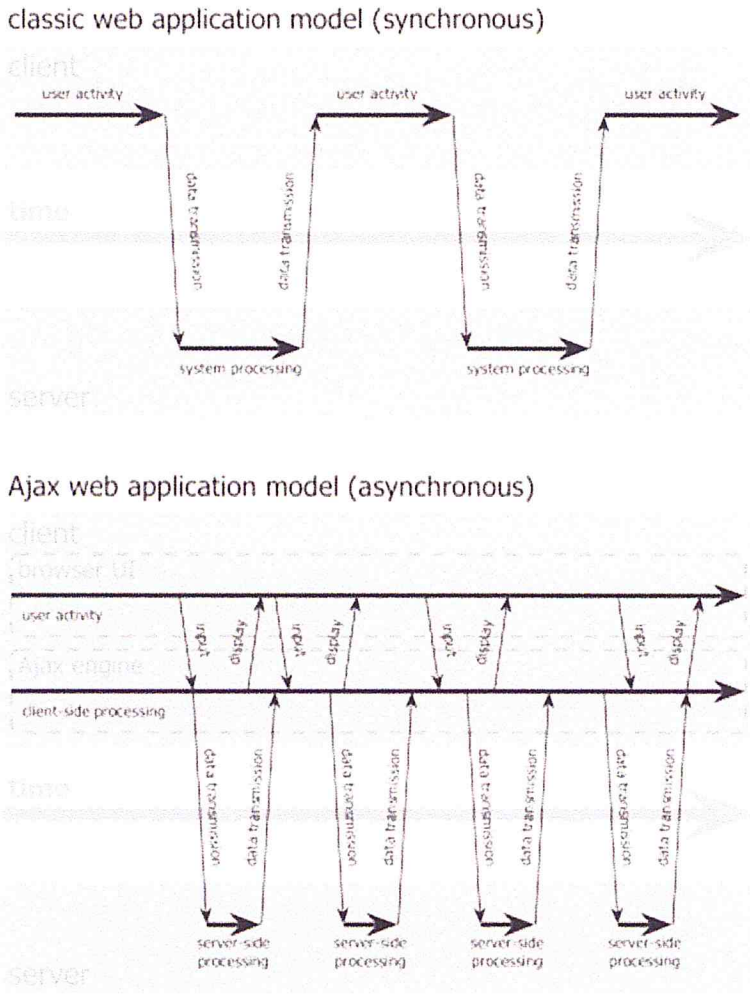


Figure II.2 : Le modèle synchrone vs modèle asynchrone des application web. [11]

5. Node.Js

5.1. Définition :

Node.Js est une plateforme logicielle libre et événementielle en JavaScript pour construire facilement des applications de réseau rapides et évolutives. Elle utilise la machine virtuelle V8 de Google Chrome. Node.js contient une bibliothèque de serveur HTTP intégrée, ce qui rend possible de construire ou de créer un serveur web sans avoir besoin d'un logiciel externe comme Apache ou autres. Il permet aussi de mieux contrôler la façon dont le serveur web fonctionne, tout en étant idéal pour les applications en temps réel intensives en données qui courent à travers des dispositifs distribués.

5.2.Principe de fonctionnement

JavaScript avait toujours été utilisé du côté du client, c'est-à-dire l'utilisateur qui navigue sur un navigateur web comme Chrome, Opera ou autre … Ce navigateur exécute du code JavaScript qui effectue des changements visibles que pour ce navigateur.

Contrairement à PHP (Figure II.3) qui s'exécute du côté du serveur, Node.Js (Figure II.4) offre un environnement coté serveur qui permet d'utiliser le langage JavaScript pour la génération des pages web. En résumé, Node.Js vient comme remplaçant moderne et technologique de PHP ou de J2EE.

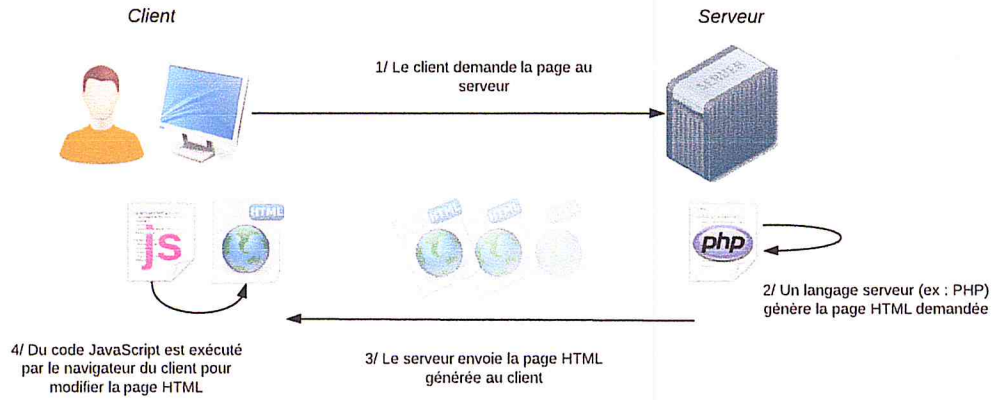


Figure II.3: Schéma avec serveur PHP [9]

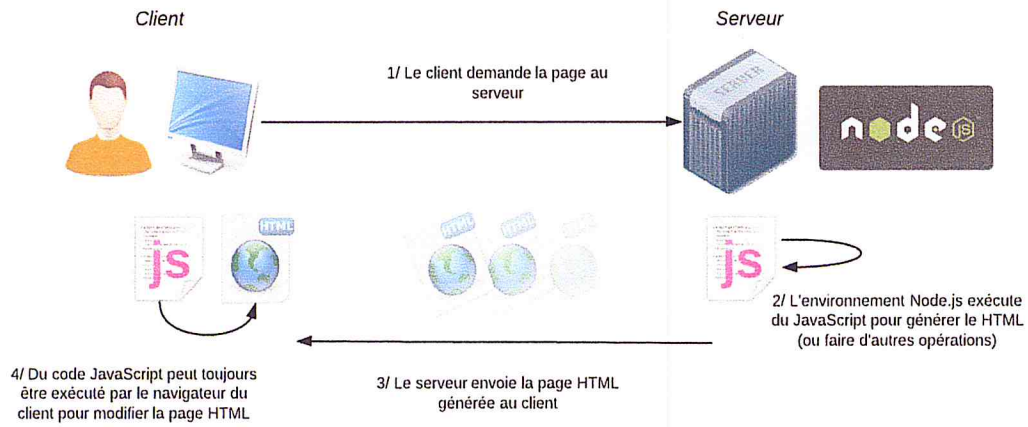


Figure II.4: Schéma avec serveur Node.Js [9]

Comme JavaScript est un langage basé sur les évènements, Node.js est aussi basé sur les évènements. Donc, c'est toute la façon d'écrire des applications web qui change. Et c'est de la que Node.js tire toute sa puissance et sa rapidité.

5.3. L'extensibilité :

Il est intéressant de signaler que le noyau de Node.js est tout petit. A la base, Node.js ne sait en fait pas faire grand-chose. Pourtant, Node.js est très riche grâce à son extensibilité. Ces extensions de Node.js sont appelées des « modules ».

Il existe des milliers de modules qui offrent des fonctionnalités variées : de la gestion des fichiers uploadés à la connexion aux bases de données MySQL ou à Redis, en passant par des Framework, des systèmes de Template et la gestion de la communication temps réel avec les utilisateurs ou clients. Il existe ainsi une panoplie de modules et de nouveaux apparaissent chaque jour [12].

6. Websocket

6.1.Définition

Web socket est un nouveau protocole qui permet la communication bidirectionnelle entre serveur et client. Il est soutenu par tous les navigateurs web modernes. Ce protocole est implémenté au-dessus de HTTP et peut être facilement utilisé par les serveurs Web, et même sur des canaux sécurisés, tels que le protocole HTTPS. En raison de ces avantages, beaucoup de personnes choisissent souvent ce protocole pour leurs projets client-serveur [13].

6.2.Avantages des Websocket

Websocket est livré avec quelques bonnes caractéristiques et de grands avantages par rapport aux autres méthodes de communication, nous citerons:

- ✓ offre une connexion full-duplex sans beaucoup de frais généraux
- ✓ une connexion légère qui utilise moins de bande passante
- ✓ Sécurité
- ✓ Requiert moins d'effort pour le développeur en termes d'apprentissage et de mise en œuvre dans les différentes technologies
- ✓ Websocket présente une faible latence qui diminue de près de 150 ms à 50 ms
- ✓ fonctionne sur le protocole de contrôle de transmission (TCP)
- ✓ Soutenu par presque tous les navigateurs Web et les serveurs Web, y compris les navigateurs mobiles.

Nous pouvons traiter Websocket comme une caractéristique qui améliore l'expérience des applications web. En particulier avec les fonctionnalités améliorées de HTML5, on peut facilement créer des applications dynamiques temps réel [14].

7. WebRTC

7.1. Définition

WebRTC est un nouveau modèle de navigation Web. Car pour la première fois, les navigateurs sont en mesure d'échanger directement des données en temps réel avec d'autres navigateurs en utilisant le modèle peer-to-peer.

WebRTC est une nouvelle norme qui permet aux navigateurs de communiquer en temps réel en utilisant une architecture peer-to-peer. Il permet l'échange de données y compris l'audio/vidéo entre les navigateurs en peer-to-peer. Cette évolution est perturbatrice dans le monde des applications web car elle permet pour la toute première fois aux développeurs web de créer des applications multimédia en temps réel sans avoir besoin de plugins propriétaires [15].

7.2.L'API WebRTC

L'API WebRTC permet à une application JavaScript de profiter des capacités des technologies temps réel du navigateur. La fonction de navigation en temps réel mise en œuvre dans le noyau du navigateur fournit la fonctionnalité nécessaire pour établir les canaux audio, vidéo et de données.

L'API est conçue autour de trois concepts principaux: MediaStream, PeerConnection et Datachannel.

- MediaStream : est une représentation abstraite d'un flux de données audio et vidéo. Ce type d'applications pourrait être utilisé pour afficher, enregistrer et envoyer son contenu à un autre. Deux types de flux sont disponibles: Local MediaStream et Remote MediaStream. Local MediaStream est un flux capturé à partir du système et Remote MediaStream est un flux qui est reçu à partir d'un autre pair.

- PeerConnection : permet la communication directe entre les utilisateurs. Pour établir une connexion et une négociation de signalisation, il faut un canal de signalisation. Ceci est fourni par le biais d'un script mis en œuvre dans un serveur web, en utilisant websockets ou XMLHttpReequest. Il utilise le protocole ICE avec STUN et TURN pour permettre des flux multimédias sûr, traversant des réseaux NAT et traversant les pare-feu.
- DataChannel : est un canal de données bidirectionnel en peer-to-peer. Il permet de transférer des données et non pas des médias qui recourent à un autre protocole tels que SCTP encapsulés dans DTLS. De cette manière, il est possible d'avoir une solution de NAT avec la confidentialité, l'authentification de l'origine et l'intégrité des données nécessitant une transmission [16].

Chapitre III :

Conception de l'application EasyP2P

1. Introduction :

Comme dans chaque système informatique qui possède une base de données, notre système ou autrement dit notre application web nommée « **EasyP2P** » a besoin aussi d'une conception générale et détaillée sur ces différents fonctionnements, interactions et utilisations par un internaute. Dans ce chapitre nous aborderons le langage utilisé pour spécifier cette conception, puis nous détaillerons les différents modèles de notre application web.

2. Unified Modeling Language (UML)

1. Définition :

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue [17].

2. Les diagrammes d'UML :

UML s'articule autour de treize types de diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Ces types de diagrammes sont répartis en deux grands groupes :

- **Six diagrammes structurels** :
  - Diagramme de classes – Il montre les briques de base statiques : classes, associations, interfaces, attributs, opérations, généralisations, etc.
  - Diagramme d'objets - Il montre les instances des éléments structurels et leurs liens à l'exécution.
  - Diagramme de packages- Il montre l'organisation logique du modèle et les relations entre packages.
  - Diagramme de structure composite – Il montre l'organisation interne d'un élément statique complexe.
  - Diagramme de composants – Il montre des structures complexes, avec leurs interfaces fournies et requises.
  - Diagramme de déploiement – Il montre le déploiement physique des « artefacts » sur les ressources matérielles.
- **Sept diagrammes comportementaux** :
  - Diagramme de cas d'utilisation - Il montre les interactions fonctionnelles entre les acteurs et le système à l'étude.
  - Diagramme de vue d'ensemble des interactions - Il fusionne les diagrammes d'activité et de séquence pour combiner des fragments d'interaction avec des décisions et des flots.
  - Diagramme de séquence - Il montre la séquence verticale des messages passés entre objets au sein d'une interaction.

- Diagramme de communication - Il montre la communication entre objets dans le plan au sein d'une interaction.
- Diagramme de temps – Il fusionne les diagrammes d'états et de séquence pour montrer l'évolution de l'état d'un objet au cours du temps.
- Diagramme d'activité - Il montre l'enchaînement des actions et décisions au sein d'une activité.
- Diagramme d'états – Il montre les différents états et transitions possibles des objets d'une classe. [17]

L'ensemble des treize types de diagrammes UML peut ainsi être résumé par la figure III.1, en mettant en évidence les cinq diagrammes les plus utilisés.

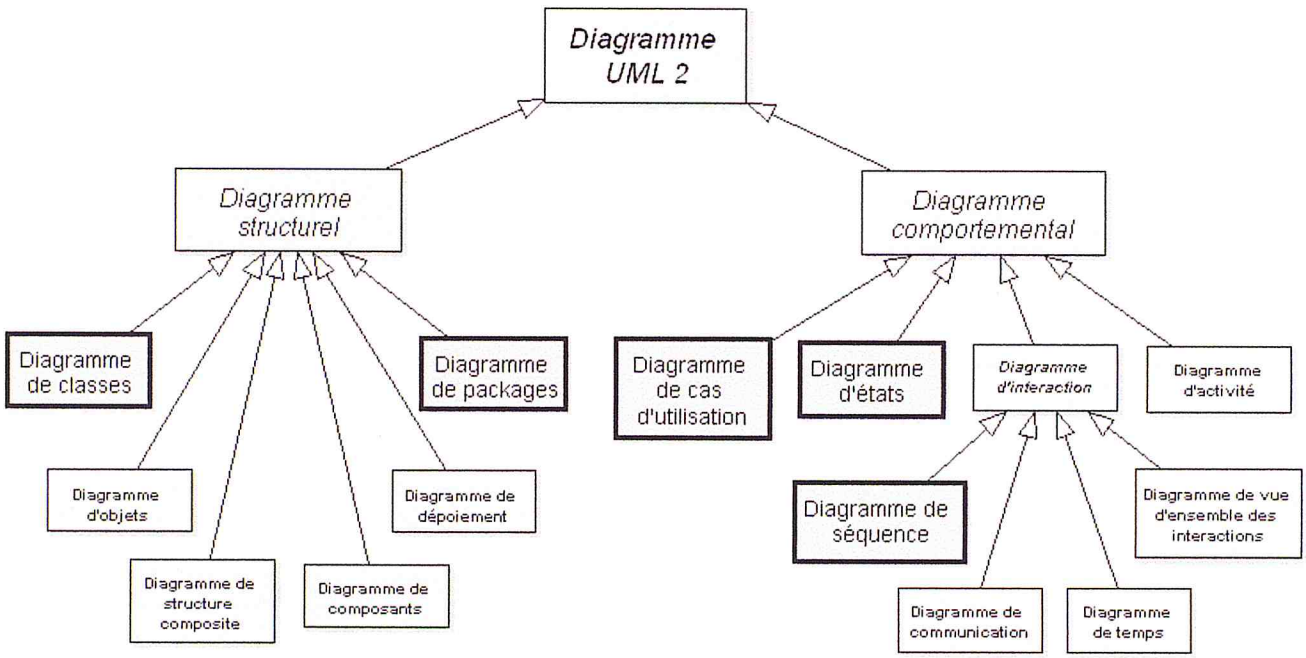


Figure III.1 : Les diagrammes d'UML [17]

3. Web Application Extension

3.1.Définition

Web Application Extension (WAE) pour UML est une extension du langage UML pour la modélisation d'applications web. Il permet de représenter des pages Web et d'autres éléments architecturalement significatifs dans le but d'exprimer avec précision la totalité du système dans un modèle.

Cette extension UML est exprimée en termes de stéréotypes, de valeurs marquées, et de contraintes. Ces mécanismes permettent d'étendre la notation UML, ce qui permet de créer de nouveaux types de blocs de construction.

- Stéréotype : c'est une extension du vocabulaire de la langue, permettant de joindre une nouvelle signification sémantique à un élément de modèle. Les stéréotypes peuvent être appliqués à presque tous les éléments du modèle et sont généralement représentés comme une chaîne entre une paire de guillemets : « ».

- Valeurs associées : 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 sont associées.
- Contrainte : c'est est une règle qui définit la manière dont le modèle peut être mis en place [18].

3.2. Les différents types de stéréotype

WAE définit trois stéréotypes de classe de base et divers stéréotypes de l'association :

➢ **Classe :**

❖ *Server page* : représente une page Web dynamique qui contient le contenu assemblé sur le serveur à chaque fois qu'il est demandé. Typiquement, une page serveur contient des scripts qui sont exécutés par le serveur qui interagit avec les ressources côté serveur : bases de données, les systèmes externes, et ainsi de suite. Il peut s'agir par exemple de pages PHP ou JSP. La figure III.2 montre les différentes représentations graphiques d'une telle classe.

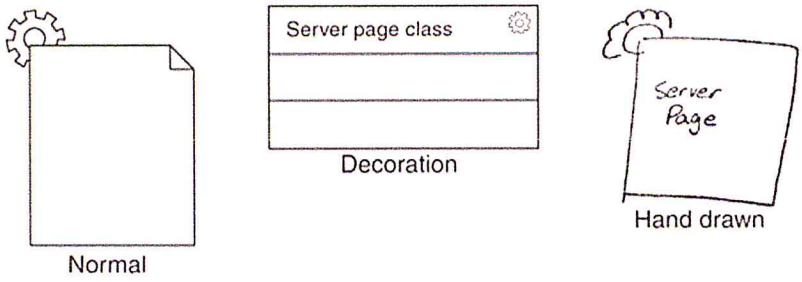


Figure III.2 : les différents icones des pages serveurs [18].

- *Client page* : est une page Web au format HTML avec un mélange de données, de présentation et autres. Les pages client sont rendues par les navigateurs clients et peuvent contenir des scripts qui sont interprétés par le navigateur. La figure III.3 montre les différentes représentations graphiques d'une telle classe.

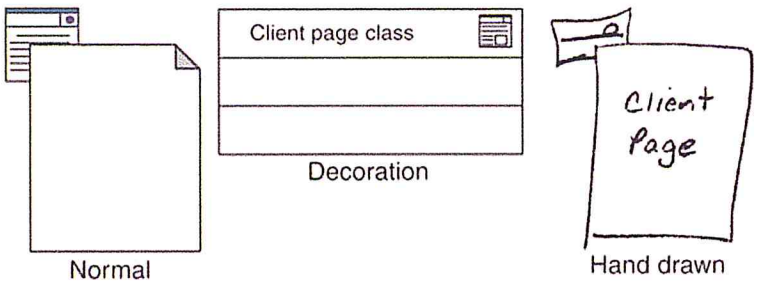


Figure III.3 : les différents icones des pages clients [18].

- *HTML form* : est une collection de champs de saisie qui font partie d'une page client. Cette classe correspond directement à la balise HTML <form>. La figure III.4 montre les différentes représentations graphiques d'une telle classe.

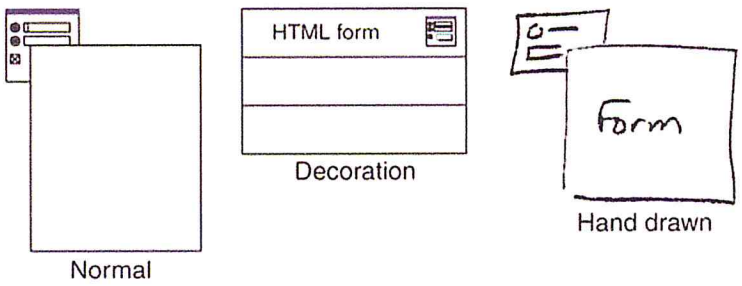


Figure III.4 : les différents icones des formulaires HTML [18].

➢ **Association :**

- Include: Une association entre une page serveur ou client et une autre page serveur ou client. Cette association indique que la page incluse contient un traitement et que son contenu ou son produit est utilisé par la société mère.
- Link: Une relation entre une page client et une page serveur ou client. Une relation «link» est généralement utilisée lorsque l'attribut *href* est défini.
- Submit: Une relation entre un formulaire HTML et une page serveur.
- Redirect: Une relation entre une page client ou une page serveur et une autre page. Cette association indique que le client demande une autre ressource.
- Build: Une relation entre une page serveur et une page client. Cette relation représente la création du code HTML après l'exécution d'une page serveur [18].

4. Processus de développement

4.1. Définition :

Un processus définit une séquence d'étapes, partiellement ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant. L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles [17].

Plus simplement, un processus doit permettre de répondre à la question fondamentale : « Qui fait quoi et quand ? ».

4.2. Processus unifié :

4.2.1. Définition :

Le Processus Unifié (UP, pour Unified Process) est un processus de développement logiciel « itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques »

- ✓ Itératif et incrémental : le projet est découpé en itérations de courte durée (environ 1 mois) qui aident à mieux suivre l'avancement global. A la fin de chaque itération, une partie exécutable du système final est produite, de façon incrémentale.
- ✓ Centré sur l'architecture : tout système complexe doit être décomposé en parties modulaires afin de garantir une maintenance et une évolution facilitées. Cette

architecture (fonctionnelle, logique, matérielle, etc.) doit être modélisée en UML et pas seulement documentée en texte.

- ✓ Piloté par les risques : les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.
- ✓ Conduit par les cas d'utilisation : le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d'utilisation du futur système sont identifiés, décrits avec précision et priorisés [17].

4.2.2. Les phases du processus UP :

La gestion d'un tel processus est organisée suivant les quatre phases suivantes : initialisation, élaboration, construction et transition.

La phase d'initialisation conduit à définir la « vision » du projet, sa portée, sa faisabilité, son business case, afin de pouvoir décider au mieux de sa poursuite ou de son arrêt.

La phase d'élaboration poursuit trois objectifs principaux en parallèle :

- ➢ identifier et décrire la majeure partie des besoins des utilisateurs,
- ➢ construire (et pas seulement décrire dans un document !) l'architecture de base du système,
- ➢ lever les risques majeurs du projet.

La phase de construction consiste surtout à concevoir et implémenter l'ensemble des éléments opérationnels (autres que ceux de l'architecture de base). C'est la phase la plus consommatrice en ressources et en effort.

Enfin, la phase de transition permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux [17].

5. Conception de l'application EasyP2P :

Dans cette partie on va aborder la partie conceptuelle du notre système « EasyP2P » qui utilise l'architecture Peer-to-Peer centralisée. Nous commencerons par les cas d'utilisation qui traduisent les exigences 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 on termine avec les diagrammes de séquence.

5.1. Etude de l'existant et des besoins :

Le monde Peer-to-Peer a connu un grand succès avec l'accroissement des échanges de données ces dernières années, ce qui explique l'existence de plusieurs logiciels comme les clients BiTorrent. Cependant, l'utilisation d'un tel logiciel nécessite la présence d'une application déjà installée permettant de se connecter au réseau peer-to-peer. Une autre possibilité consisterait à utiliser une application web qui utilise des plugins spécifiques.

Dans le cadre de ce travail, nous proposons une nouvelle application web pour le partage de fichiers en peer-to-peer qui n'utilise aucun plugin propriétaire et qui ne nécessite aucune installation de la part de l'utilisateur. Le développement d'une telle application est rendu possible grâce aux nouvelles technologies web temps réelles qui ont récemment vu le jour.

- Exigences fonctionnelles :
  L'application web « **EasyP2P** » devra regrouper toutes les fonctionnalités nécessaires de recherche, de partage des fichiers et de téléchargement.
- Exigences non fonctionnelles :
  L'application devra proposer des fonctionnalités comme la messagerie instantanée et la possibilité de rejoindre des groupes privées pour limiter les téléchargements aux membres de ces groupes.

5.2. Cas d'utilisation :

Pour assurer un bon fonctionnement du système et satisfaire les besoins et les attentes des utilisateurs, il faut prendre en considération les besoins fonctionnels comme des repères et les analyser à travers les principaux cas d'utilisations.

Ces repères doivent décrire le fonctionnement du système pour garantir un meilleur service aux différents utilisateurs de notre futur système en faisant interagir un ensemble d'entités qui sont :

- Le système : Le système définit l'application informatique, il ne contient donc pas les acteurs, mais les cas d'utilisation et leurs associations.
- Les acteurs : Un acteur représente une personne ou un périphérique qui joue un rôle (interagit) avec le système.
- Les cas d'utilisations : Chaque cas d'utilisation décrit un ensemble d'interactions successives d'une entité en dehors du système (utilisateurs) avec le système lui-même pour réaliser une fonctionnalité.

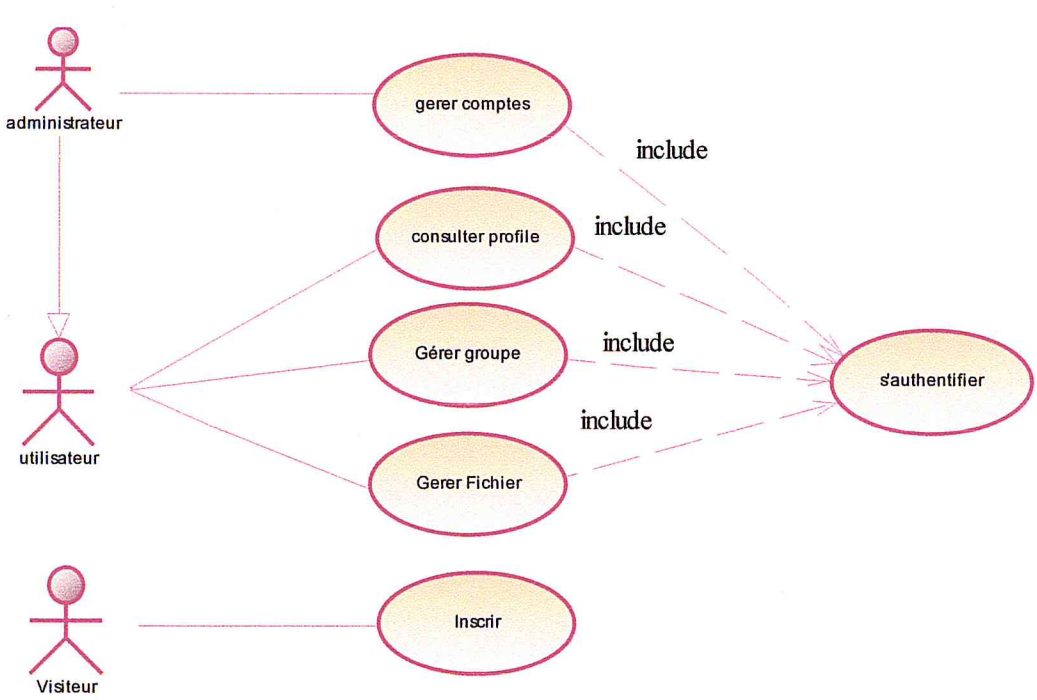


Figure III.5 : Diagramme de cas d'utilisation du système.

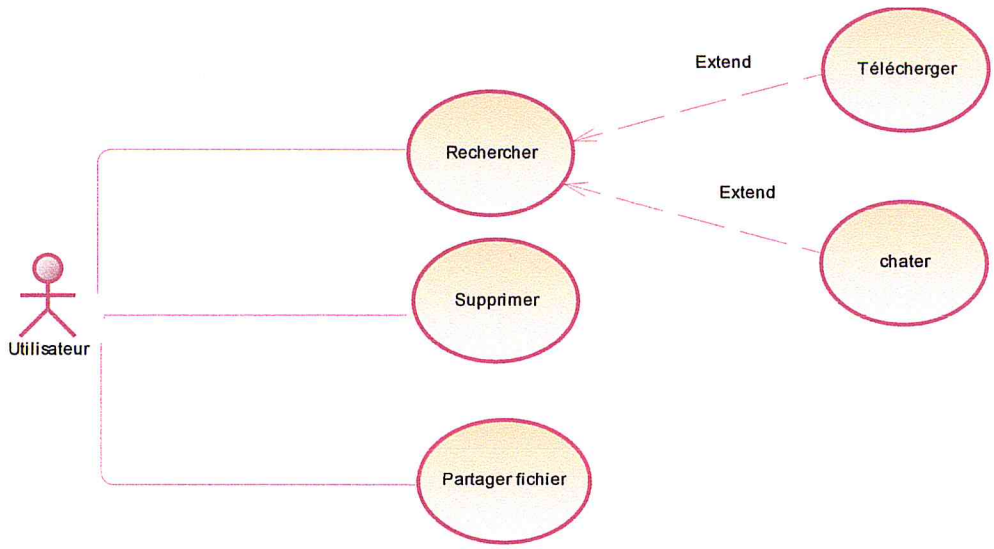


Figure III.6 : Diagramme du cas d'utilisation « Gérer fichiers »

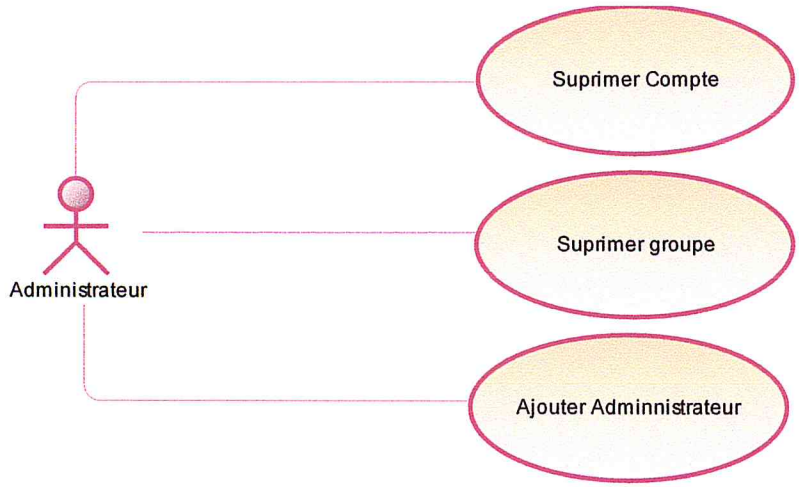


Figure III.7 : Diagramme du cas d'utilisation « Gérer comptes »

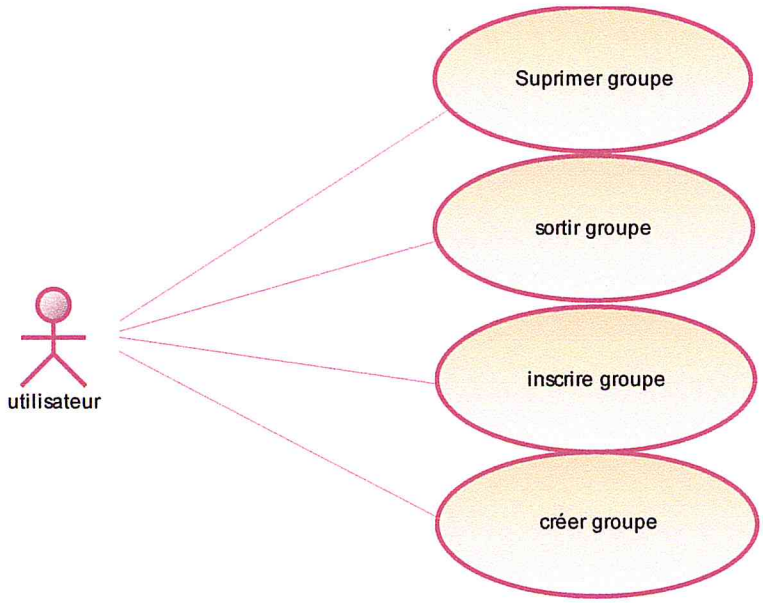


Figure III.8 : Diagramme du cas d'utilisation « Gérer groupe »

Pour notre application web, nous avons identifié trois acteurs :

- Administrateur : il s'agit de l'administrateur de l'application web
- Utilisateur : il s'agit d'un internaute qui dispose d'un compte et qui s'est authentifié
- Visiteur : il s'agit d'un internaute qui ne s'est pas encore authentifié

Nous donnons dans ce qui suit la description textuelle des différents cas d'utilisation.

- Description textuelle

Description du cas « inscrire»
Identification : Nom du cas : «inscrire » Objectif : création d'un compte d'utilisateur. Acteur principal : visiteur. Acteur secondaire : néant.
Pré-condition : – néant. Enchaînement nominal : **1-** Le visiteur lance une demande d'inscription. **2-** Un formulaire d'inscription est affiché. **3-** Le visiteur saisit ses informations. **4-** L'inscription est effectuée et le visiteur devient un utilisateur. Enchaînement d'exception : – Si un champ n'est pas rempli alors un message est affiché « veuillez renseigner ce champ ». – Si l'adresse mail existe déjà, un message est affiché « Adresse email déjà enregistrée dans notre système ». Post-condition : – Le visiteur devient un utilisateur. – L'utilisateur est connecté au système, et redirigé vers la page principale de l'application.

Description du cas « s'authentifier»
Identification : Nom du cas : «s'authentifier » Objectif : avoir accès au système. Acteur principal : utilisateur. Acteur secondaire : néant.
Pré-condition : – Avoir déjà un compte. Enchaînement nominal : **1-** L'utilisateur saisit son adresse mail et son mot de passe. **2-** Le système vérifie l'adresse et le mot de passe. **3-** Le système autorise l'accès à l'utilisateur. Enchaînement d'exception : – Si un champ n'est pas rempli alors un message est affiché « veuillez renseigner ce champ ». – Si l'adresse ou le mot de passe sont incorrects, un message est affiché « adresse ou mot de passe incorrects ». L'utilisateur sera redirigé vers une autre tentative d'identification. Post-condition : – L'utilisateur est connecté au système, et redirigé vers la page principale de l'application.

Description du cas « modifier profile»
Identification : Nom du cas : «modifier profile » Objectif : mettre à jour les informations de l'utilisateur. Acteur principal : utilisateur. Acteur secondaire : néant.
Pré-condition : – Utilisateur doit être connecté au système. Enchaînement nominal : **1-** L'utilisateur ouvre la page 'paramètres'. **2-** L'utilisateur change ses informations. **3-** Le système met à jour les informations de l'utilisateur. **4-** L'utilisateur peut modifier sa photo directement depuis son profile. Enchaînement d'exception : – Selon ce qu'il a rempli comme champ, alors un message lui est affiché « Veuillez renseigner tel ou tel champ ». – Si une image est invalide, un message s'affiche « Veuillez sélectionner un fichier de type image » – Si la taille de l'image est trop grande, un message s'affiche « Veuillez sélectionner une image de moins de 3Mo » Post-condition : – Profile utilisateur mis à jour.

Description du cas « supprimer profile»
Identification : Nom du cas : «supprimer profile » Objectif : suppression d'un utilisateur. Acteur principal : utilisateur. Acteur secondaire : néant.
Pré-condition : – Utilisateur doit être connecté au système. Enchaînement nominal : **1-** L'utilisateur ouvre la page 'paramètres'. **2-** L'utilisateur supprime son profile. **3-** Un message est affiché « vous êtes sur le point de supprimer votre profile». **4-** Si confirmation, le Profile sera supprimé. Enchaînement d'exception : ➢ néant. Post-condition : – Profile supprimé.

Description du cas « supprimer groupe»
Identification : Nom du cas : «supprimer groupe » Objectif : suppression d'un groupe. Acteur principal : utilisateur. Acteur secondaire : néant.
Pré-condition : – Utilisateur doit être connecté au système. – Utilisateur est le propriétaire du groupe ou bien est administrateur Enchaînement nominal : **1-** L'utilisateur ouvre la page 'groupe'. **2-** L'utilisateur supprime le groupe qu'il souhaite. **3-** Un message est affiché « vous êtes sur le point de supprimer votre groupe». **4-** Si confirmation, le groupe est supprimé. **5-** L'utilisateur quitte le groupe et est redirigé vers le groupe public. Enchaînement d'exception : ➢ néant. Post-condition : – Groupe supprimé.

Description du cas « supprimer compte»
Identification : Nom du cas : «supprimer compte » Objectif : suppression d'un utilisateur. Acteur principal : administrateur. Acteur secondaire : néant.
Pré-condition : – Administrateur doit être connecté au système. – L'utilisateur existe. Enchaînement nominal : **1-** Administrateur recherche un utilisateur. **2-** Une liste d'utilisateurs s'affiche. **3-** Administrateur sélectionne l'utilisateur à supprimer **4-** Un message est affiché « vous êtes sur le point de supprimer le profil de cet utilisateur ». **5-** Si confirmation, le Profile est supprimé. Enchaînement d'exception : ➢ néant. Post-condition : – Profile supprimé.

Description du cas « ajouter administrateur»
Identification : Nom du cas : «ajouter administrateur» Objectif : ajout d'un administrateur. Acteur principal : administrateur. Acteur secondaire : néant.
Pré-condition : – Administrateur doit être connecté au système. – l'utilisateur existe. Enchaînement nominal : **1-** Administrateur recherche un utilisateur. **2-** Une liste d'utilisateurs s'affiche. **3-** Administrateur sélectionne un utilisateur pour le rendre comme administrateur. **4-** Un message est affiché « Voulez-vous vraiment rendre cet utilisateur comme un administrateur ? » **5-** Si confirmation, Utilisateur devient un administrateur. Enchaînement d'exception : ➢ néant. Post-condition : ➢ néant.

Description du cas « recherche et téléchargement d'un fichier»
Identification : Nom du cas : «recherche et téléchargement» Objectif : L'Utilisateur veut trouver le plus rapidement possible un fichier et le télécharger. Acteur principal : utilisateur. Acteur secondaire : néant.
Pré-condition : – Utilisateur doit être connecté au système. Enchaînement nominal : **1-** Utilisateur recherche un fichier. **2-** Une liste de fichiers s'affiche. **3-** Utilisateur sélectionne un fichier pour le télécharger. **4-** Si confirmation par le propriétaire, le fichier est téléchargé. Enchaînement d'exception : ➢ Si le propriétaire est hors ligne un message est affiché « veuillez réessayer ultérieurement » Post-condition : ➢ néant.

Description du cas « partager fichier»
Identification : Nom du cas : «partager fichier» Objectif : partage des fichiers avec d'autres utilisateurs. Acteur principale : utilisateur. Acteur secondaire : néant.
Pré-condition : – utilisateur doit être connecté au système. Enchaînement nominal : 1- Utilisateur sélectionne un fichier à partager. 2- Fichier partagé dans le groupe où l'utilisateur est connecté. Enchaînement d'exception : ➢ Si le fichier est déjà partagé par le même utilisateur et dans le même groupe un message s'affiche « fichier déjà ajouté ». Post-condition : ➢ néant.

5.3. Diagrammes de séquence système

➢ Cas d'inscription

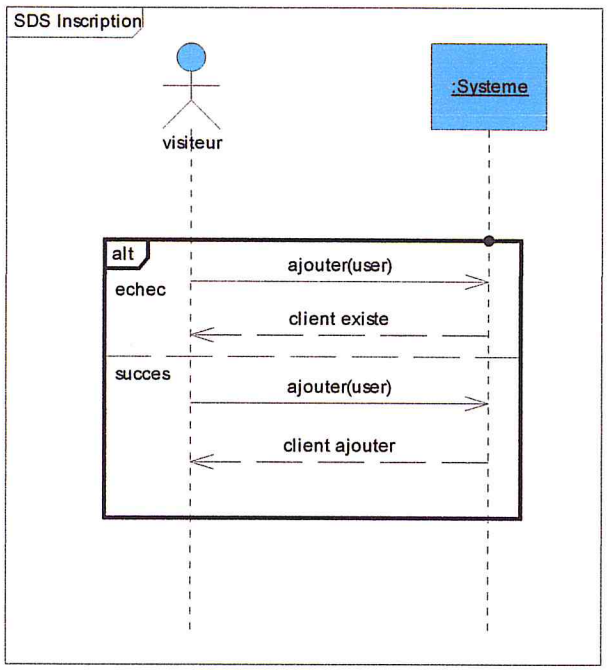


Figure III.9 : diagramme de séquence système pour l'inscription

➢ cas d'authentification

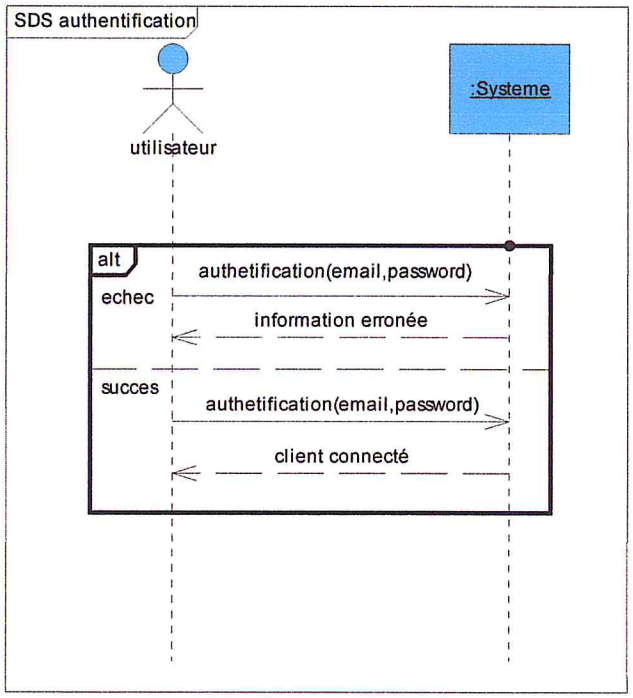


Figure III.10 : diagramme de séquence système d'authentification.

➢ **Cas de gestion de profile**

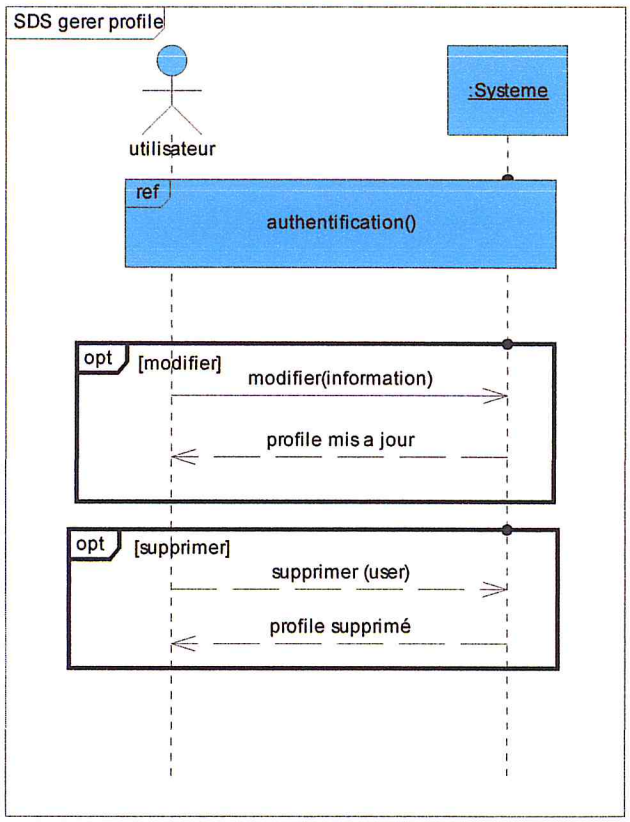


Figure III.11 : diagramme de séquence système pour la gestion de profile

➢ **Cas de chat**

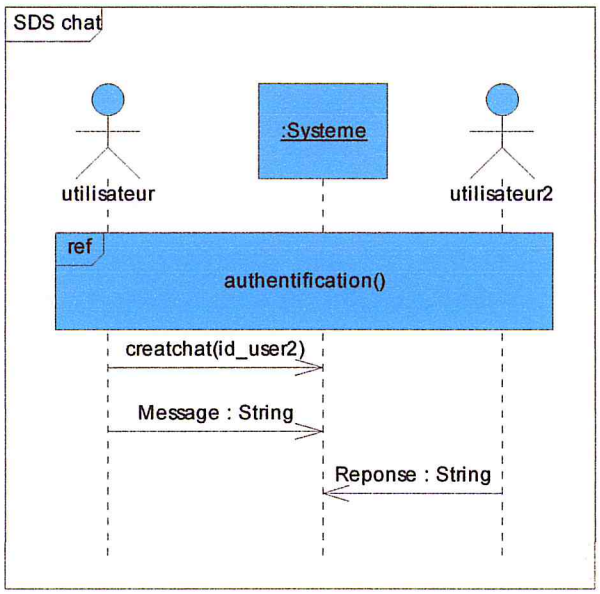


Figure III.12 : diagramme de séquence système pour le chat

➢ **cas de recherche et téléchargement de fichier**

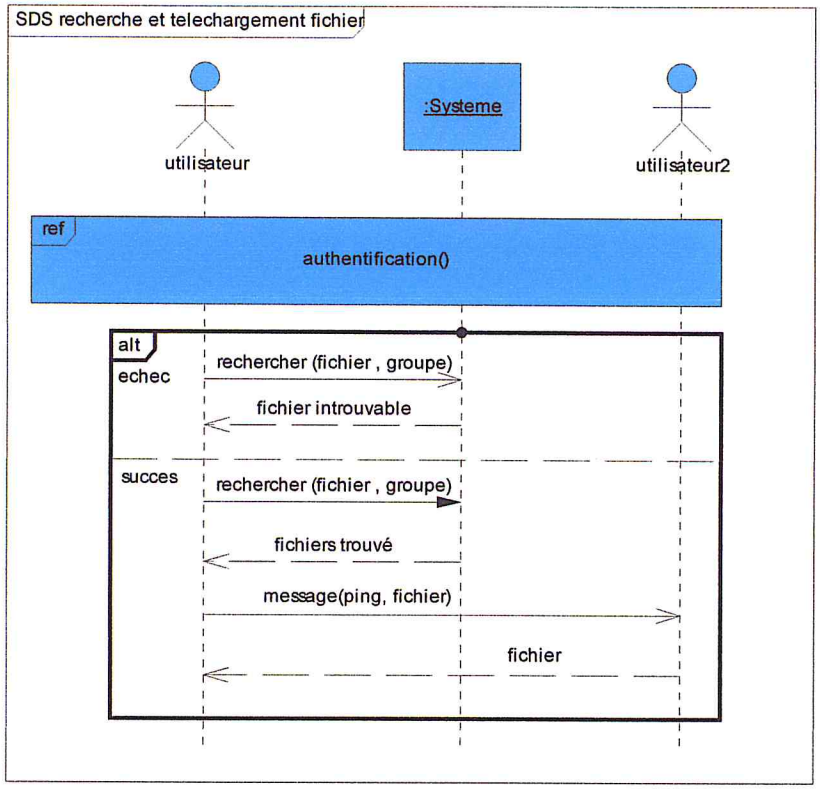


Figure III.13: diagramme de séquence système pour la recherche et le téléchargement d'un fichier.

➢ **cas du partage de fichier**

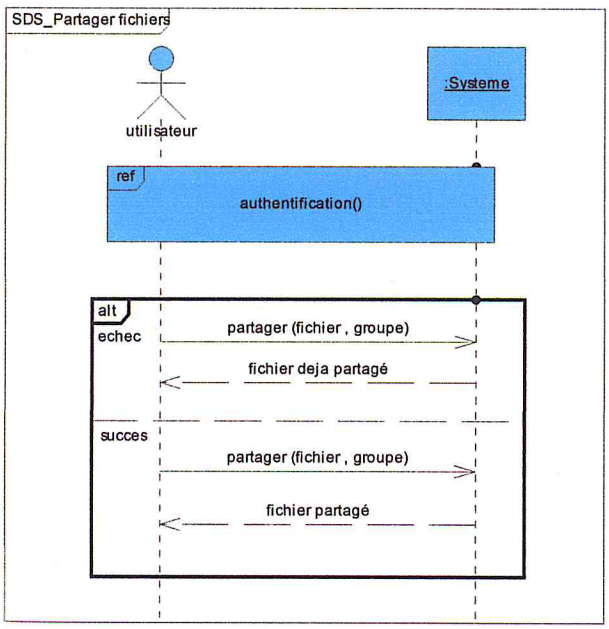


Figure III.14 : diagramme de séquence système pour le partage d'un fichier.

➢ **cas de gestion des groupes**

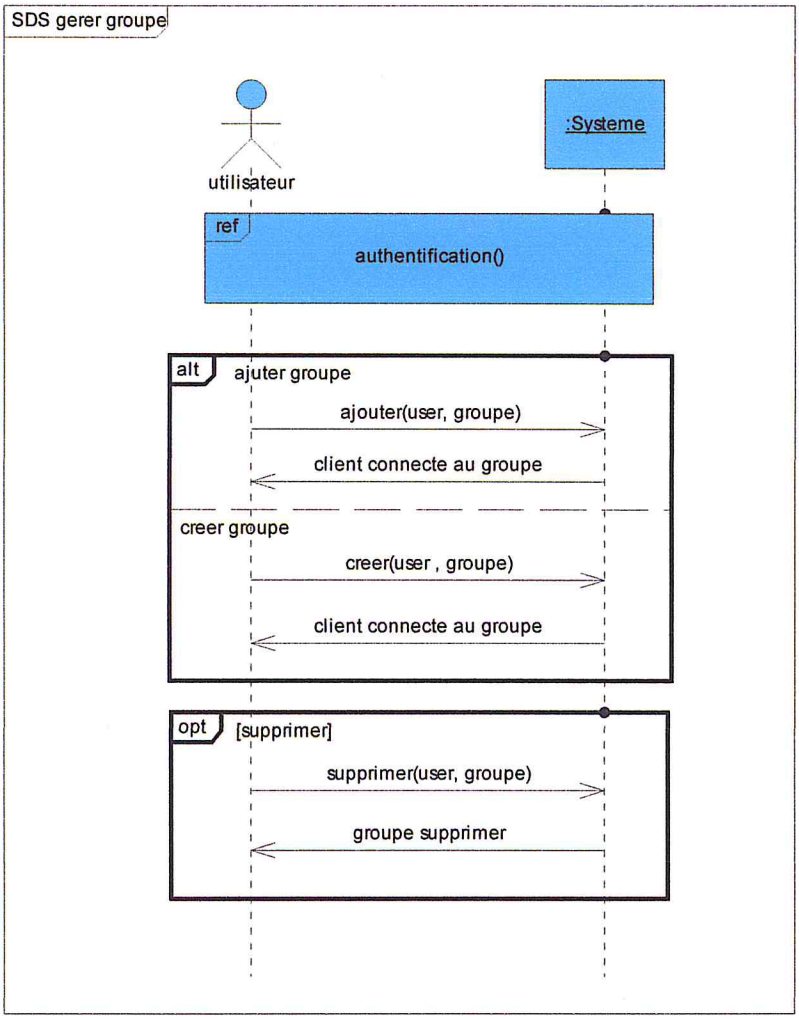


Figure III.15: diagramme de séquence système pour la gestion de groupe

➢ **cas de gestion des comptes**

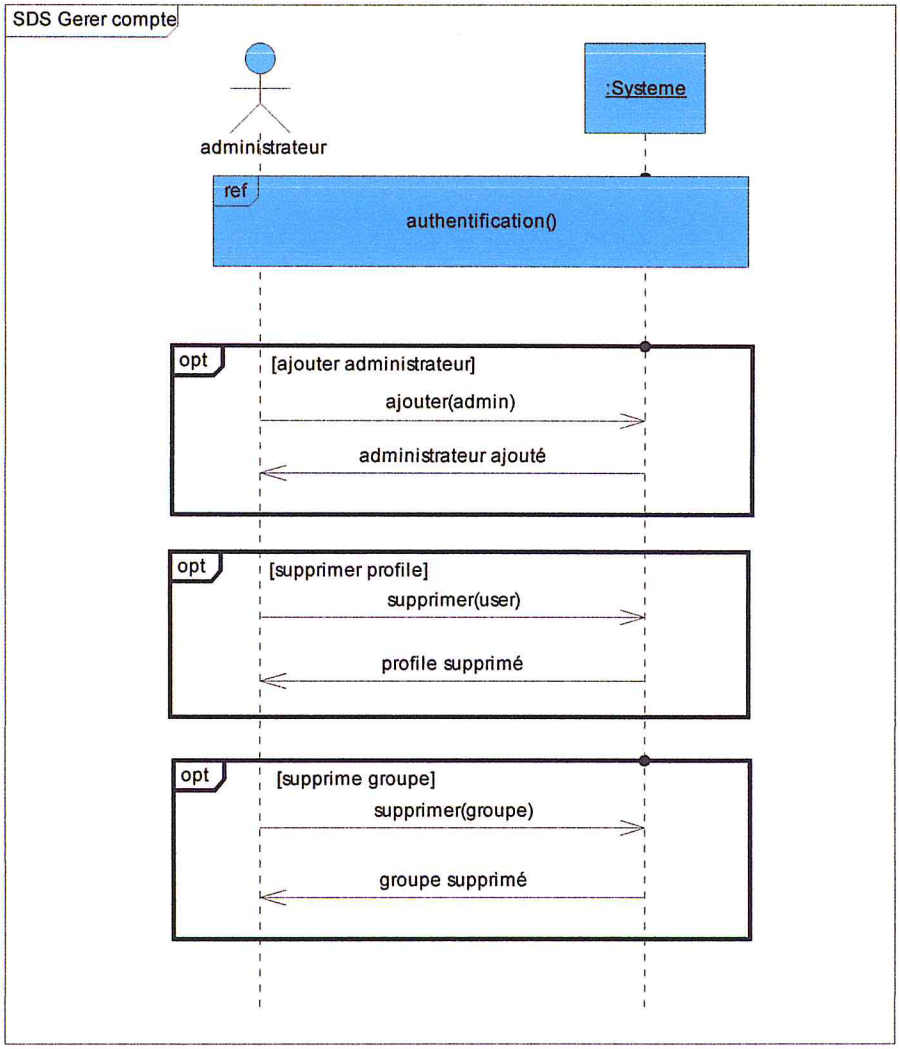


Figure III.16 : diagramme de séquence système pour la gestion de compte

5.4. Diagrammes de classe

Pour bien décrire notre application web, nous présentons dans ce qui suit deux diagrammes de classes : le premier (figure III.17) représente la partie base de données de notre application ; le second (figure III.18) représente les liens entre les différentes parties de notre application web (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.

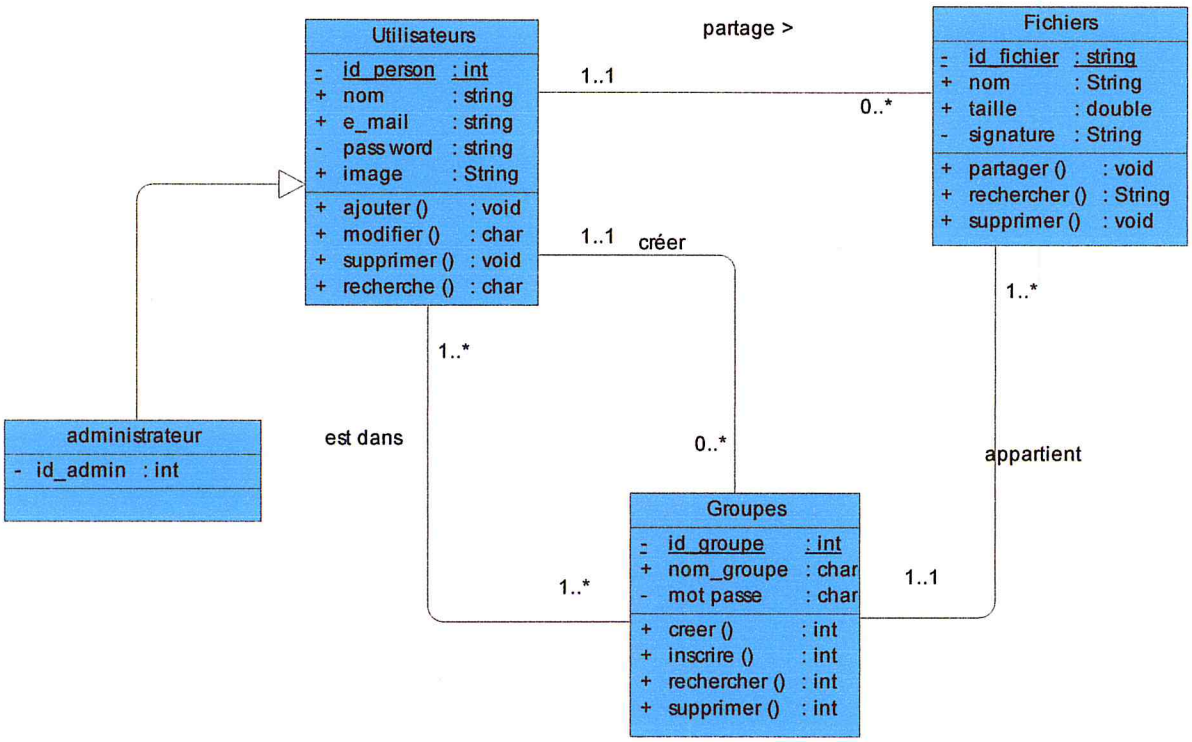


Figure III.17 : 1er diagramme de classe

Le passage du diagramme de classes de la figure III.17 au modèle relationnel donne les tables suivantes :

Utilisateurs (*id_personne* , nom, e_mail, password, image, admin) ;
Fichiers (*id_fichier*, nom, taille, signature, *id_personne#* ,*id_groupe#*) ;
Groupes (*id_groupe*, nom_groupe, mot_passe, *id_personne#*) ;
Utilisateur_groupe (*id_groupe# , id_personne#*) ;

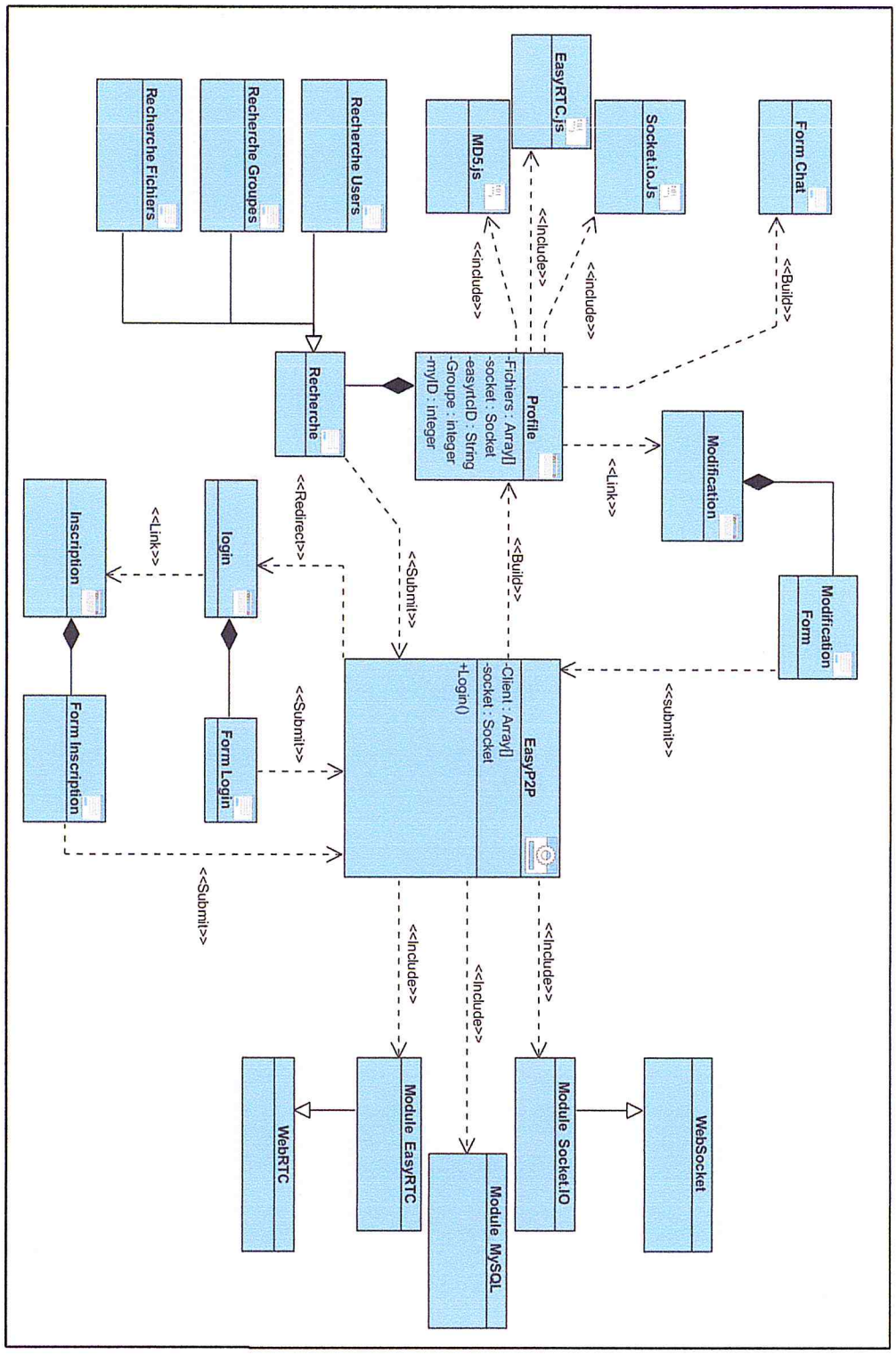


Figure III.18 : 2ème diagramme de classe

Ce deuxième diagramme de classes décrit les classes les plus importantes utilisées dans notre application à savoir les différentes pages client et serveur, les formulaires d'envoi au serveur ainsi que les bibliothèques JavaScript et les modules. Les relations entre ces différents éléments sont également indiquées sur le diagramme.

5.5.Diagrammes de séquence

➢ Diagramme de séquence d'authentification

Ce diagramme présente en détail l'authentification et l'initialisation de la page profil.

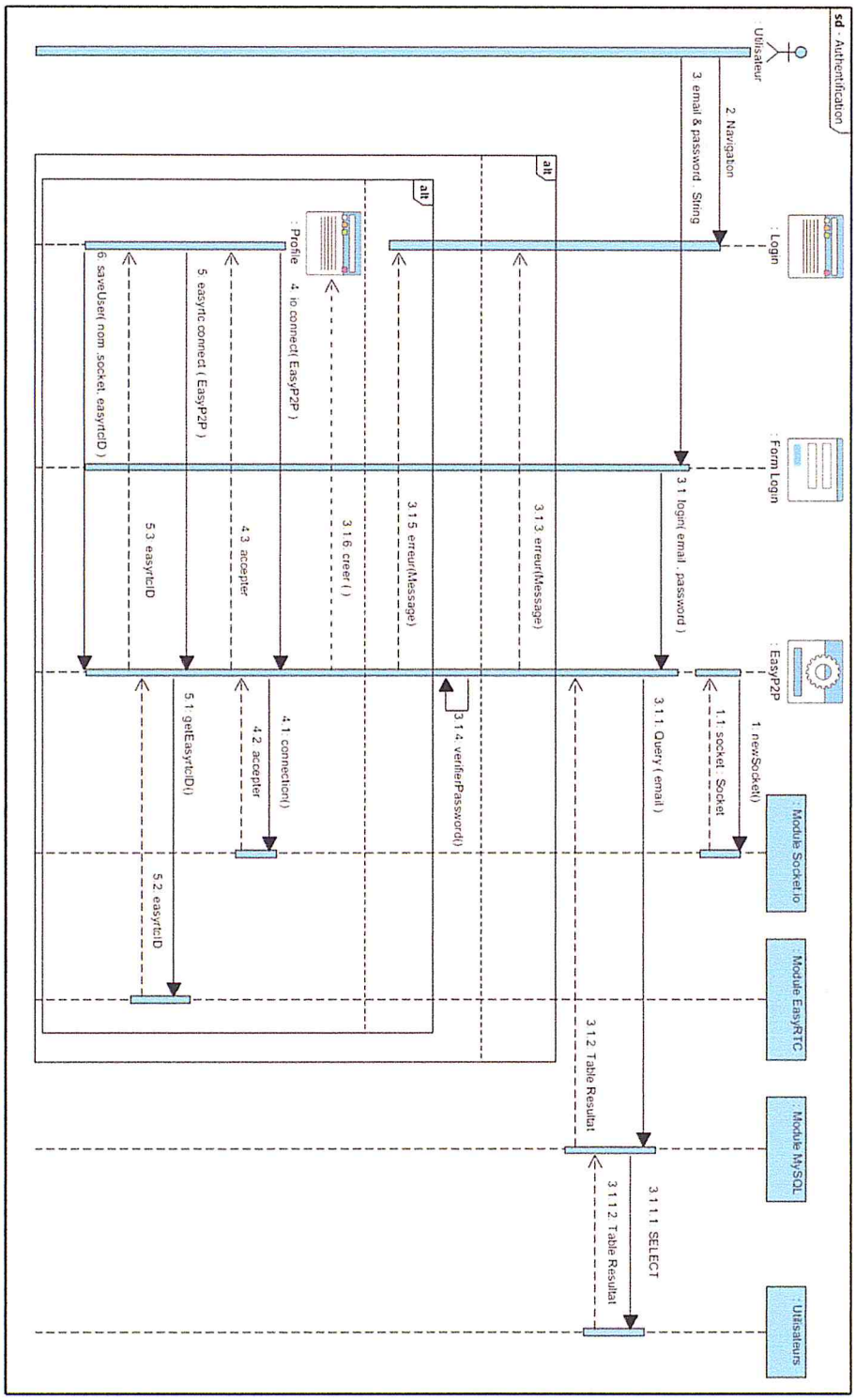


Figure III.19 : diagramme de séquence d'authentification.

➢ **Diagramme de séquence de la recherche et le téléchargement de fichier**

Ce diagramme présente le processus de recherche d'un fichier puis du téléchargement du fichier souhaité.

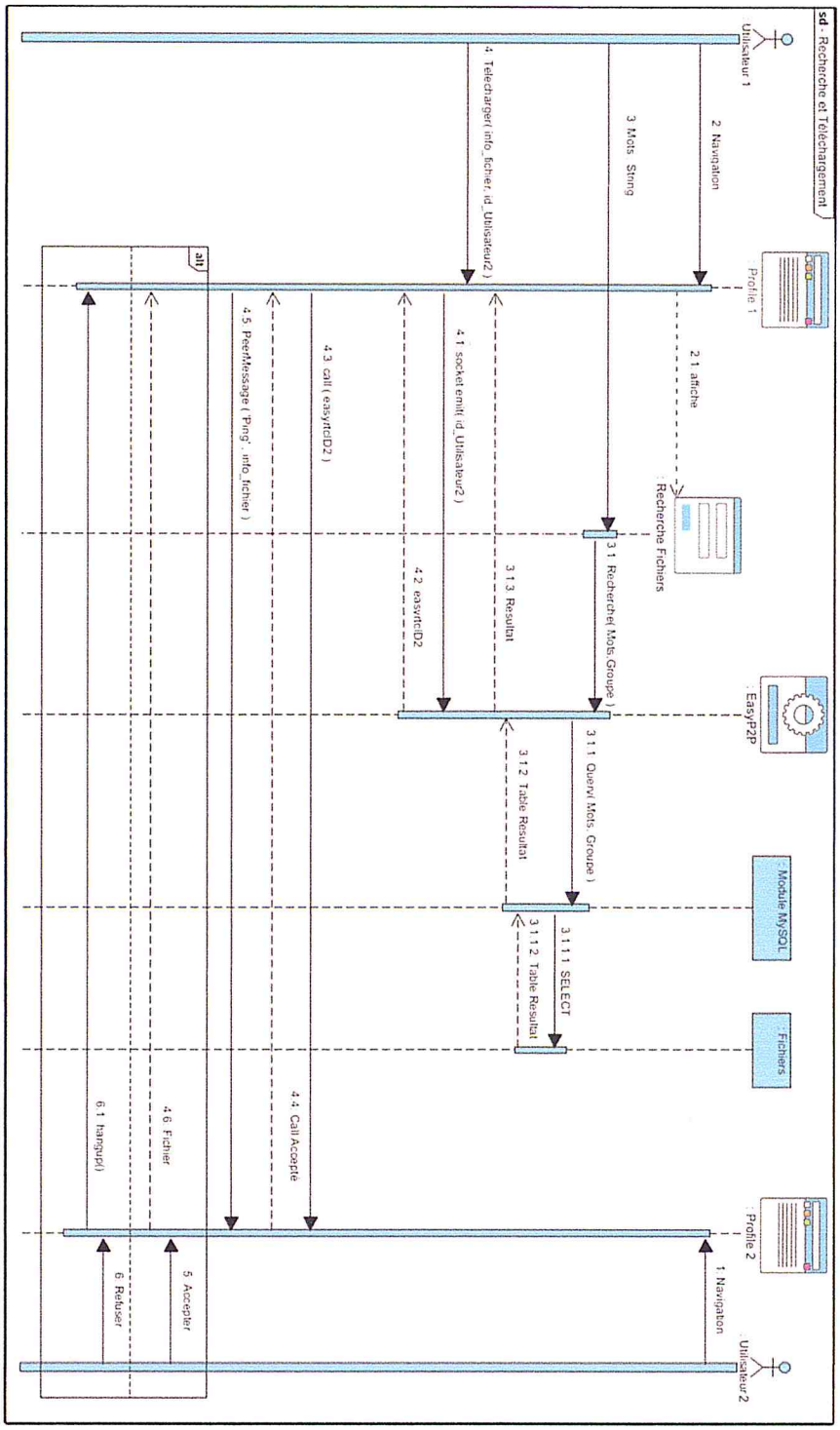


Figure III.20 : diagramme de séquence pour la recherche et le téléchargement d'un fichier.

➢ Diagramme de séquence pour le partage de fichiers

Ce diagramme représente le processus de partage d'un fichier par un utilisateur sans préciser le groupe car le partage dans le groupe public ou dans un groupe spécifique se fait de la même manière.

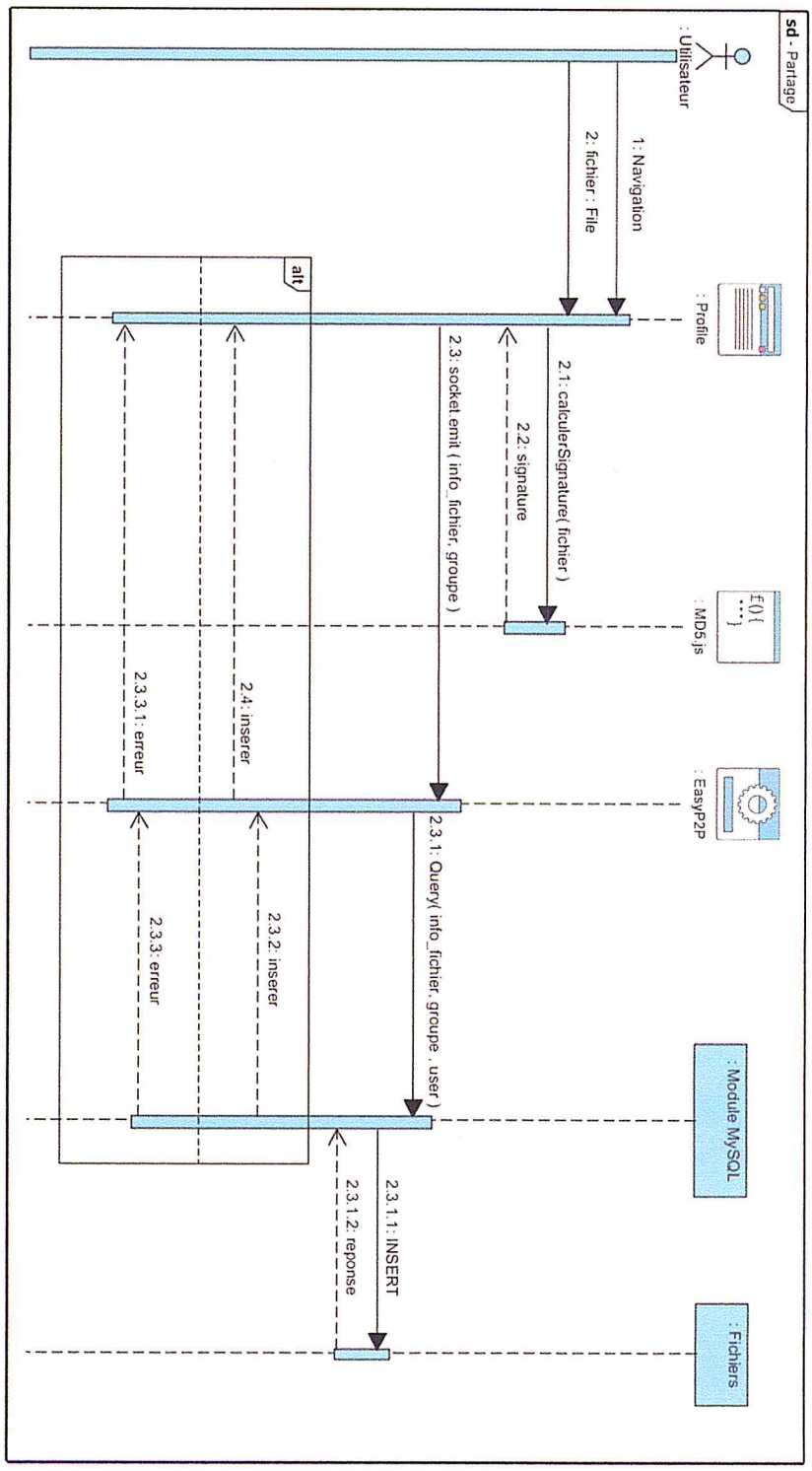


Figure III.21 : diagramme de séquence pour le partage de fichier.

➢ **Diagramme de séquence pour le chat**

Le diagramme suivant illustre l'envoi d'un message textuel entre deux utilisateurs.

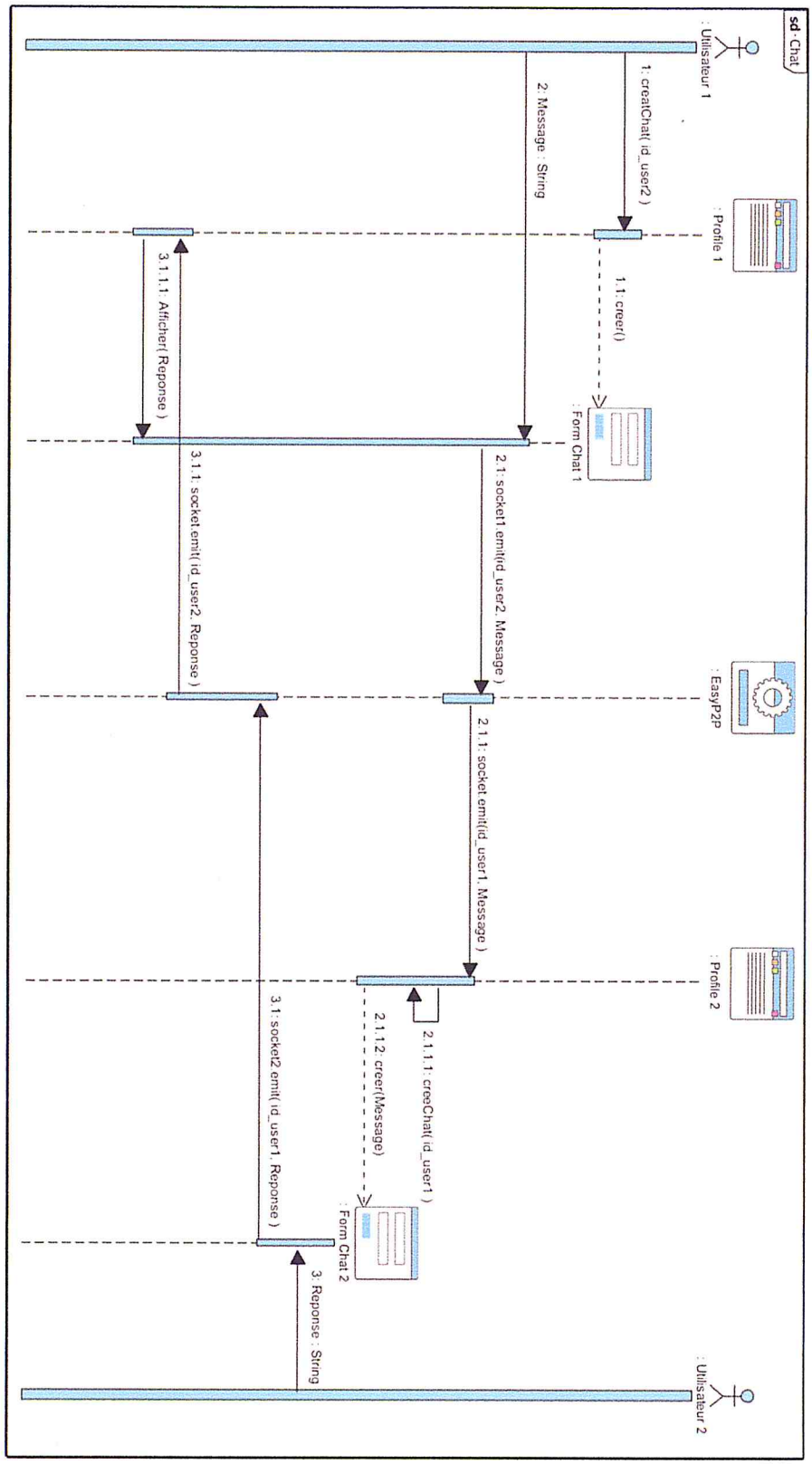


Figure III.22 : Diagramme de séquence de chat.

6. Conclusion :

Afin de mettre en œuvre notre application web EasyP2P, nous avons réalisé dans ce chapitre une conception détaillée en utilisant l'extension WAE de UML.

La conception d'une application web comme la nôtre peut être effectuée de différentes façons car il y a différents idées et différent méthodes à chaque fois, chacun sa propre conception et sa propre façon de voir les choses, car elle varie selon l'imagination et l'expérience du concepteur. Par contre, celle que nous avons proposée pour « **EasyP2P** » répond à tous nos besoins, et nous a permis d'aboutir au résultat souhaité, que nous allons découvrir dans le chapitre suivant.

Chapitre IV :

Réalisation de l'application EasyP2P

1. Introduction

Dans les chapitres précédents, nous avons présenté les différents ingrédients pour la réalisation de notre application web. Dans ce chapitre, nous décrivons les détails techniques concernant la réalisation de notre application web. Nous illustrons également les différentes fonctionnalités proposées par notre application web.

2. Réalisation

2.1.Architecture de l'application

Le nombre impressionnant de produits et de technologies liés à Internet disponibles actuellement a suscité des architectures d'application web multiples et variées.

Les patterns architecturaux les plus courants actuellement sont les suivants :

- Le client web très léger et universel : l'application n'utilise que HTML coté client, donc le client ne nécessite qu'un navigateur web standard et la logique métier ainsi que la logique de présentation sont intégralement exécutées sur le serveur.
- Le client web léger : l'application utilise HTML et JavaScript coté client, donc le client doit avoir un navigateur web assez récent, comprenant le langage JavaScript.
- Le client web alourdi : embarque dans les pages web des composants plus complexes : ActiveX, applets Java, plug-ins, ce qui permet à une partie significative de la logique métier d'être exécutée sur le poste client.
- Le client lourd : est une application à part entière, qui s'exécute sur le poste client. Ce pattern correspond typiquement aux applications Intranet pour lesquelles on maitrise la configuration [17].

Dans notre cas, on utilise le client web léger comme illustré par la figure ci-dessous qui montre l'architecture de l'application ainsi les différentes technologies et protocoles utilisés.

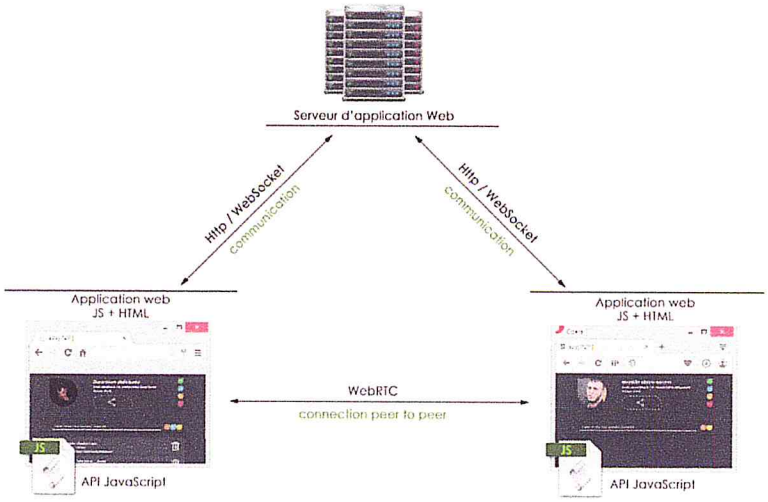


Figure IV.1 : Architecture de l'application.

Les différents composants de notre application sont :

- ✓ Le serveur web : il est créé avec Node.js.
- ✓ Les pages web : elles sont implémentées en HTML et JavaScript.
- ✓ La communication entre le client et le serveur est établie grâce à l'extension socket.io des Websocket.
- ✓ La communication entre les pairs est établie grâce à la bibliothèque EasyRTC qui facilite l'utilisation de l'API WebRTC.

2.2.Implémentation

L'implémentation de notre application web se déroule en 3 grandes étapes :

- ✓ Mise en place du serveur et des modules nécessaires.
- ✓ Implémentation de la Base de Données.
- ✓ Implémentation des pages en HTML et JavaScript.

2.2.1. Serveur NodeJs :

2.2.1.1. Les Modules npm :

Node.js contient plusieurs modules qui ont différents fonctionnements mais leur but est de faciliter l'utilisation de Node.js.

Voici la liste des modules utilisés dans notre application :

- ❖ **Express** : La création de l'application et de ses comportements.
- ❖ **Mysql** : Les interactions avec la ou les bases de données.
- ❖ **Querystring** : Gérer les requêtes en utilisant le type « *String* ».
- ❖ **Path** : Gérer les chemins.
- ❖ **Mime** : Gérer les Mimes.
- ❖ **Session** : Gérer les sessions des utilisateurs.
- ❖ **http** : Créer le serveur HTTP.
- ❖ **url** : La manipulation des URL.
- ❖ **util** : Manipuler les utilitaires.
- ❖ **formidable** : Upload de fichiers (utilisé que pour les photos de profils).
- ❖ **socket**.io : Pouvoir interagir en temps réel avec les clients et le serveur.
- ❖ **easyrtc** : Pouvoir utiliser le Framework easyrtc, et l'envoi de fichier en p2p.

Notons qu'il existe beaucoup d'autres modules npm pour Node.js qu'on aurait pu utiliser.

La figure ci-dessous est un extrait du fichier JavaScript du serveur qui montre les différents modules utilisés dans notre application.

```
1  // JavaScript Document
2
3  var express      = require("express"),
4      app          = require('express')();
5
6  var mysql        = require('mysql');
7
8  var querystring = require('querystring');
9  ///////////
10 var path         = require('path');
11 var mime         = require('mime');
12 ///////////
13 var session      = require('express-session');
14
15 var server       = require('http').Server(app);
16 var url          = require("url");
17
18 var util         = require('util');
19 var formidable   = require('formidable');
20
21 var io           = require('socket.io')(server);
22 var easyrtc      = require("easyrtc");
23
24 var rtc          = easyrtc.listen(app, io);
25
```

Figure IV.2 : Liste des Modules utilisés

2.2.1.2.Mise en place des comportements

Voici une portion de code du serveur Node.js, qui retourne le fichier « index.html » au client si aucune session ne lui a déjà été attribuée, sinon il le redirige vers son profil.

```
39
40 app.get('/', function (req, res) {
41    sess = req.session;
42    // if('') condition d'existance de variable de sessions
43    if( sess.email )
44       {
45          res.writeHead( 302, {  'Location' : '/profile' });
46          res.end();
47        }
48
49    else
50        res.sendfile(__dirname + '/index.html');
51 });
52
```

Figure IV.3 : Extrait du code serveur

Cette portion de code ne s'exécute que quand l'url n'a aucun répertoire '/', c'est-à-dire que l'url ne contient que le nom de domaine et le port, dans notre cas c'est seulement « http://localhost:360/ ».

2.2.1.3.Socket.io

- **Définition**

Socket.io est à sa version v1.3.5, considérer comme le plus rapide et le plus fiable moteur en temps réel, puisque elle permet une communication bidirectionnelle basée sur les évènements. Elle marche pratiquement sur toutes les plateformes, navigateurs ou appareils où la fiabilité et la vitesse sont importantes.

- **Exemple d'utilisation**

- Analyses en temps réel : Fournir des données aux clients responsables des calculs en temps réel, des graphiques ou des logs.
- Le streaming binaire : Depuis la version 1.0, il est possible d'envoyer n'importe quel type de données média en avant ou en arrière : vidéos, images, audio.
- Messagerie instantanée ou chat.
- La collaboration de document : Autoriser les utilisateurs à modifier simultanément un document et voir les changements des autres utilisateurs.

- **Principe de fonctionnement**

La communication entre utilisateur et serveur, se fait très facilement avec les socket.io, en utilisant les événements à chaque nouvelle émission de messages.

Comme dans le cas suivant, se lance l'événement 'connection' quand l'interface client (navigateur) se connecte au serveur via les socket.io, dedans se trouve les autres événements qui s'exécutent si un client lui émet un message.

Dans ce cas de figure, l'utilisateur émet 'get_ID' contenant un message qui porte le nom de l'utilisateur, le nom du fichier et le numéro du bloc html ou se trouve ce fichier ; la fonction cherche l'identificateur EasyRTC enregistré temporairement dans le serveur.

```
592
593  io.on('connection', function(socket) {
594
595          var nomClient;
596
597          socket.on("get_ID",function(user,name,num){
598
599                for(var i=0; i<Client.length; i++) {
600                   var elem = Client[i];
601                   if(elem[0] == user ) {
602
603                        socket.emit("set_ID", elem[1],name,num);
604                        break;
605                   }
606                } });
607
```

Figure IV.4 : Extrait du code d'utilisation de socket.io

2.2.1.4.Outil de developpement

- **DreamWeaver CS6**

Dreamweaver est un logiciel qui sert à la création de pages, de sites et d'applications web développé par la société Adobe System. Originalement édité par Macromedia, Adobe System en est aujourd'hui le principal éditeur depuis qu'il a racheté cette première en 2005. Dreamweaver est facile à employer car il intègre déjà les fonctions de base qui accélèrent la conception de la page du site web. Il est l'un des meilleurs logiciels d'édition de sites web et est compatible à la fois sur PC et sur Mac [19].

2.2.2. Implémentation de la base de données

2.2.2.1.MySQL

- **Définition**

MySQL est un système de gestion de bases de données relationnelles qui utilise le langage SQL. C'est un des SGBD les plus utilisés. Sa popularité est due en grande partie au fait qu'il s'agit d'un logiciel Open Source, ce qui signifie que son code source est librement disponible et que quiconque peut modifier MySQL pour l'améliorer ou l'adapter à ses besoins. Une version gratuite est par conséquent disponible, comme il existe une autre version commerciale payante [20].

2.2.3. Pages HTML et code JavaScript

Les pages HTML qui nécessitent l'utilisation de socket.io, du hachage MD5 ou encore d'EasyRTC doivent inclure les fichiers JavaScript qui permettent de le faire. La figure suivante montre comment se fait cette inclusion.

```
8     ...
9
10  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
11  <title>Bien venue à ....</title>
12  </head>
13
14   <script src="/js/crypto_md5.js" type="text/javascript" ></script>
15   <script src="/socket.io/socket.io.js" type="text/javascript" ></script>
16   <script type="text/javascript" src="/easyrtc/easyrtc.js"></script>
17   <script type="text/javascript" src="/easyrtc/easyrtc_ft.js"></script>
18   <script >
19
20
21      ...
22
23
```

Figure IV.5 : Inclusion de librairies JavaScript dans une page HTML

2.2.3.1. Socket.io

Les socket.io du côté client s'utilisent exactement de la même façon que du côté serveur, sauf qu'il n'y a pas d'événement de connexion, car c'est ici que doit être émis cet événement.

```
83
84      ...
85
86
87  <div class="footer" id="footer_div" >
88     </div>
89
90  </center>
91
92   <script type="text/javascript" >
93     var socket = io.connect(document.location.host);
94        //document.location.host = http://NomHost:Port/
95
96     var Fichier = new Array; var i=0;
97     var id="";
98
99
100    ...
101
```

Figure IV.6 : Portion de code HTML

2.2.3.2. EasyRTC

- **Définition**

EasyRTC est un outil WebRTC complet open source. C'est une trousse appropriée pour la construction hautement sécurisée d'applications d'entreprise avec WebRTC. Il s'agit d'un paquet d'applications web, des extraits de code, des bibliothèques de composants client et serveur méticuleusement écrites et documentées. Avec EasyRTC, les développeurs Web peuvent se mettre au diapason et obtenir des applications professionnelles qui utilisent WebRTC sur le marché [21].

- **Utilisation de easyRTC**

Avec EasyRTC il y'a d'abord trois paramètres à régler avant de commencer le travail : le flux Audio, Vidéo et Données, pour gérer ensuite les chambres de connexion et établie des appels entre les pairs. Chaque pair est identifié par un identificateur fourni automatiquement par EasyRTC lors de la connexion.

```
...

easyrtc.enableDataChannels(true);
easyrtc.enableVideo(false);
easyrtc.enableAudio(false);
...
easyrtc.call(ids,   function(caller, mediatype) {
                    console.log("Connexion etabli entre  " + id  + " et " + ids );    }
                ,   function(errorCode, errorText) {
                    console.log("Erreur de Call :"+    errorText); }
                ,   function wasAccepted(yup) {  } );

easyrtc.sendPeerMessage(ids, 'ping'
                          , name
                          , function(msgType, msgBody ){
                            console.log("message was sent"); }
                          , function(errorCode, errorText){
                            console.log("error was " + errorText);   });

easyrtc_ft.buildFileReceiver(function(otherGuy,fileNameList, wasAccepted){ wasAccepted(true); }
                         , function(otherGuy, blob, filename){ easyrtc_ft.saveAs(blob, filename); }
                         , function(otherGuy, msg) {
                                 var receiveBlock = document.getElementById("prog_"+num);
                                 if( !receiveBlock) return;
                                 switch (msg.status) {
                                   case "started"      :  break;
                                   case "done"         :
                                   receiveBlock.style.background = "rgb(225, 225, 225)";
                                   easyrtc.hangup(otherGuy);
                                   break;
                                   case "started_file" :
                                   receiveBlock.style.background = "rgb(225, 225, 225)";
                                   break;
                                   case "progress"     :
                                   receiveBlock.style.width = Math.round((msg.received * 800)/msg.size)+"px";
                                   break;
                                   default             :
                                   console.log("Message default = ", JSON.stringify(msg)); }
                                   return true; });
 ...
```

Figure IV.7: Extrait de code JavaScript utilisant EasyRTC

Avec EaysRTC, pour pouvoir envoyer ou recevoir des fichiers, il faudra créer un FileReceiver et un FileSender qui ne s'exécuteront que lorsqu'un call (appel) est établi entre deux pairs (clients ou utilisateurs).

2.2.3.3.Hachage MD5

- **Définition**

MD5 est une fonction de hachage cryptographique qui produit des empreintes de 128bits. Il s'agit de la fonction de hachage utilisée pour constituer l'empreinte de la signature de PGP. MD5 a été proposée par Rivest de la société RSA en 1991 [22].

➢ **Principe de fonctionnement**

MD5 fonctionne itérativement sur des mots de 32 bits. La fonction prend en entrée une variable de chainage de quatre mots (la valeur initiale) ainsi que le message composé d'un bloc de seize mots et engendre une sortie de quatre mots (128 bits) définissant ainsi une nouvelle variable de chainage. On chaine ensuite le bloc de message suivant avec le résultat du hachage du premier bloc.

Toutes les opérations utilisées par MD5 sont définies sur des mots de 32 bits. La transformation consiste en quatre étapes successives et chaque étape est constituée de seize sous-étapes. Dans chacune des sous-étapes un mot des variables de chainage est modifié comme suit : on ajoute un mot du message et le résultat du calcul d'une fonction non linéaire dépendant des trois autres mots de la variable de chainage ; ce résultat subit ensuite une permutation circulaire d'un certain nombre de bits [22].

3. Présentation de l'application

3.1.Accueil

La figure suivante présente la page d'accueil de notre application web.

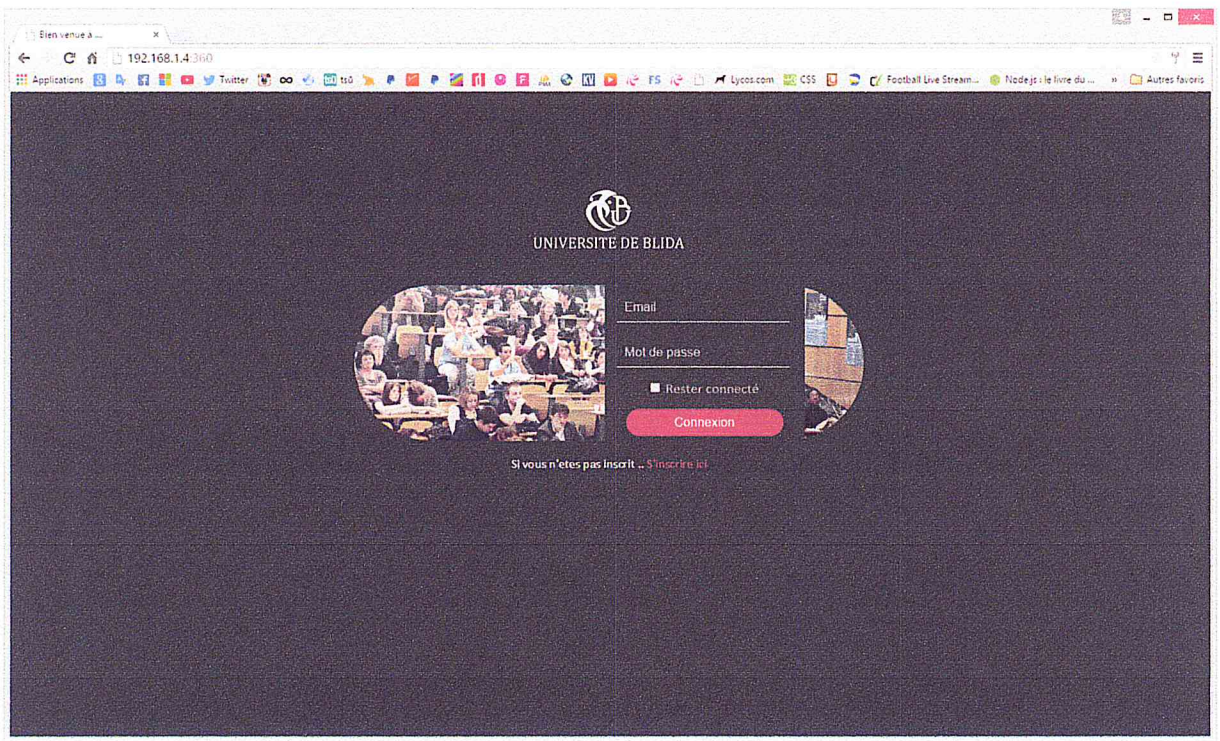


Figure IV.8 : Page d'accueil

Pour pouvoir avoir accès aux services de notre application, un utilisateur déjà inscrit doit d'abord s'authentifier, sinon il peut s'inscrire en une seule étape en cliquant sur '*S'inscrire ici'*.

3.2. Inscription d'un nouvel utilisateur :

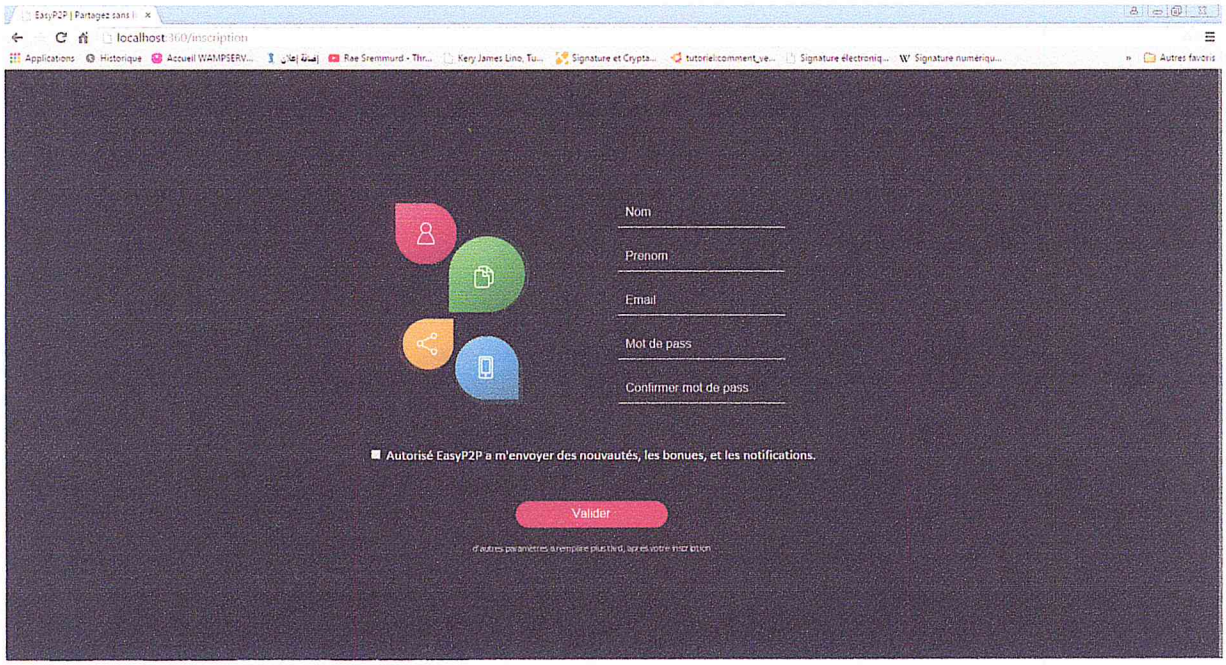


Figure IV.9 : Page d'inscription

L'inscription à notre application web (Figure IV.9) nécessite une adresse email ; une fois l'internaute clique sur le bouton « **valider** », le processus d'inscription vérifie si cette adresse email n'a pas déjà été utilisée, si oui l'utilisateur sera ajouté à la base de données et sera automatiquement redirigé vers la page d'accueil, sinon un message d'erreur s'affichera en précisant le problème qui a fait que l'inscription à échoué.

3.3.Les différents profils

3.3.1. Profil utilisateur

- **Mes fichiers**

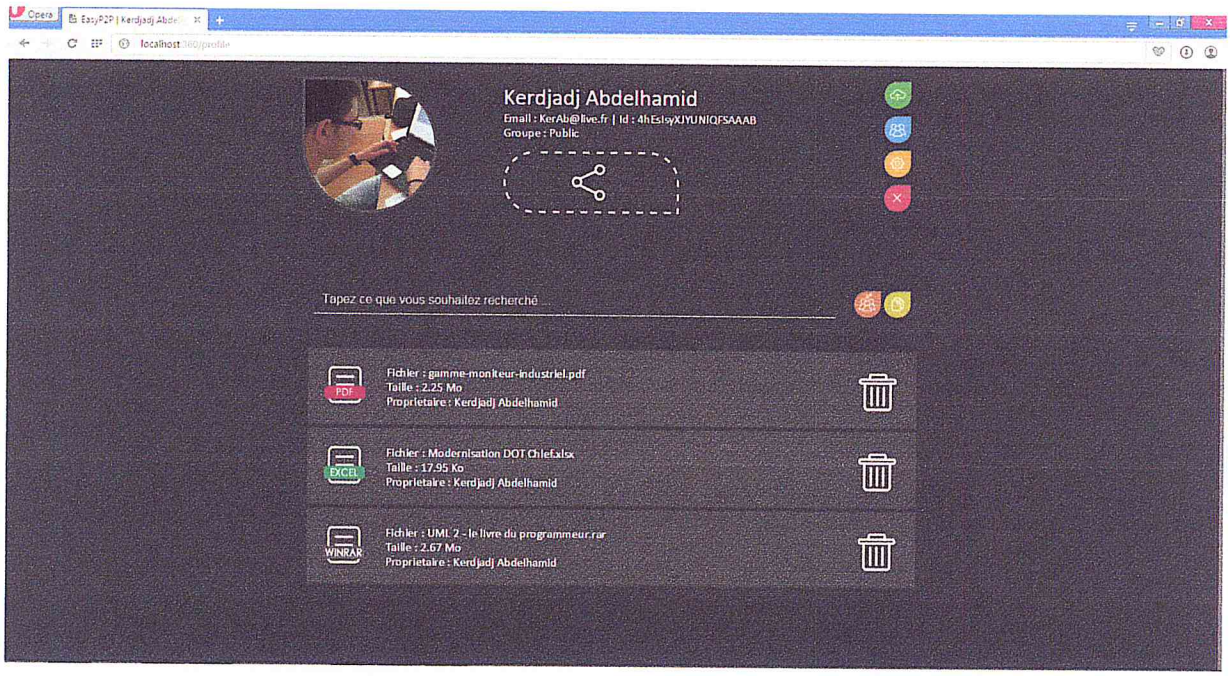


Figure IV.10 : page d'accueil de l'utilisateur « Kerdjadj Abdelhamid »

Un utilisateur identifié avec succès est redirigé vers sa propre page pour retrouver son profil ainsi que tous les fichiers qu'il a déjà partagés en public (voir Figure IV.10).

On remarque à côté de l'email de l'utilisateur un identifiant qui change à chaque nouvelle connexion de l'utilisateur. Il s'agit de l'identifiant EasyRTC de l'utilisateur qui va lui permettre d'entrer en contact avec un autre utilisateur via une communication p2p.

Ensuite il y a le nom du groupe auquel l'utilisateur est connecté, mais dans un premier temps il est dirigé vers le groupe public qui contient par défaut tous les utilisateurs.

➢ **Recherche de fichier ou de groupe**

Dans son profil, l'utilisateur possède une barre de recherche qui lui permet de rechercher des fichiers ou des groupes (voir Figure IV.11). Une fois une recherche lancée, la base de données est interrogée avec les mots qu'il a tapés, ces mots doivent être séparés par des espaces, par exemple : Livres UML SGBD. Notons que cette barre de recherche peut à la fois être utilisée pour rechercher des fichiers (Figure IV.11) ou des groupes (Figure IV.12) ; tout dépendra du bouton (orange ou jaune) sur lequel l'utilisateur clique.

Les résultats de la recherche seront affichés sur la même page sans que celle-ci ne soit actualisée. Il est à noter que les résultats affichés n'incluent pas les fichiers ou les groupes du client qui a lancé la recherche.

Chaque fichier trouvé peut, soit contenir l'icône de téléchargement et l'image du propriétaire ce qui signifie que ce dernier est connecté et qu'il est possible de télécharger le fichier et\ou de lancer une discussion avec lui, soit ne contenir aucune icône ce qui signifie que le propriétaire est hors-ligne et qu'il n'est donc pas possible de télécharger son fichier ou de discuter avec lui.

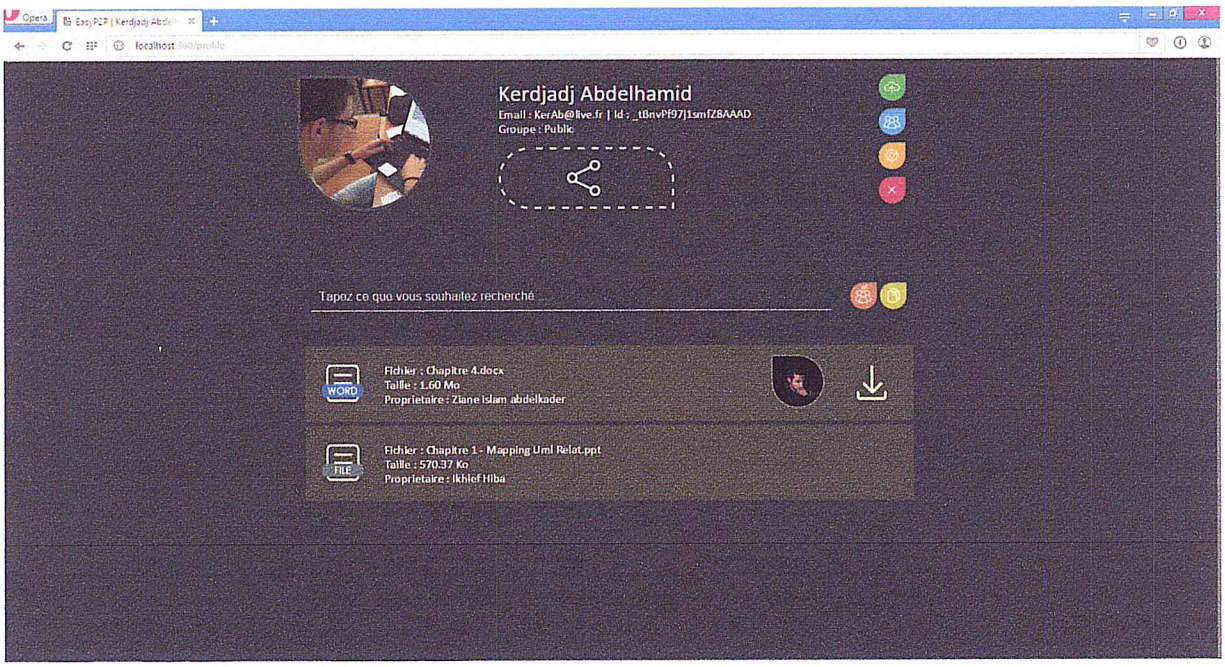


Figure IV.11 : Exécution d'une recherche des fichiers dans un profil utilisateur.

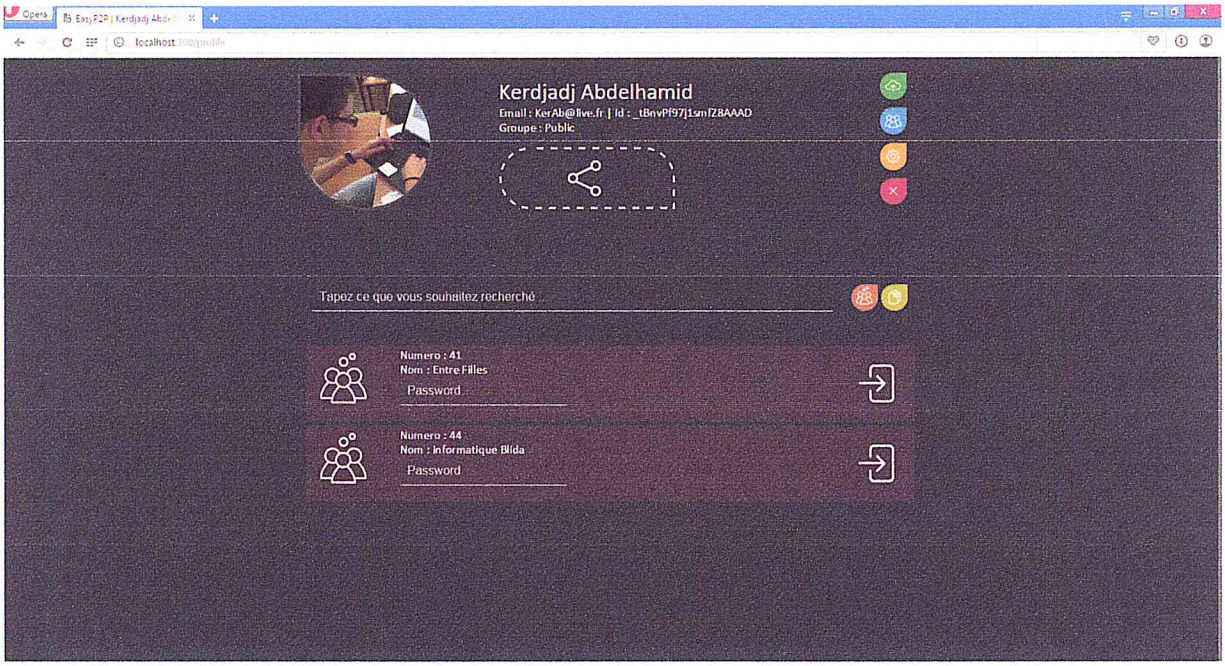


Figure IV.12 : Exécution d'une recherche des groupes dans un profil utilisateur.

- **Gestion des groupes**

Afin qu'un utilisateur puisse gérer les différents groupes auxquels il appartient, il lui suffit pour cela de cliquer sur « groupe » pour afficher la liste des groupes auxquels il est inscrit. Il peut alors soit afficher un groupe, quitter un groupe, ou encore supprimer un groupe dont il est propriétaire.

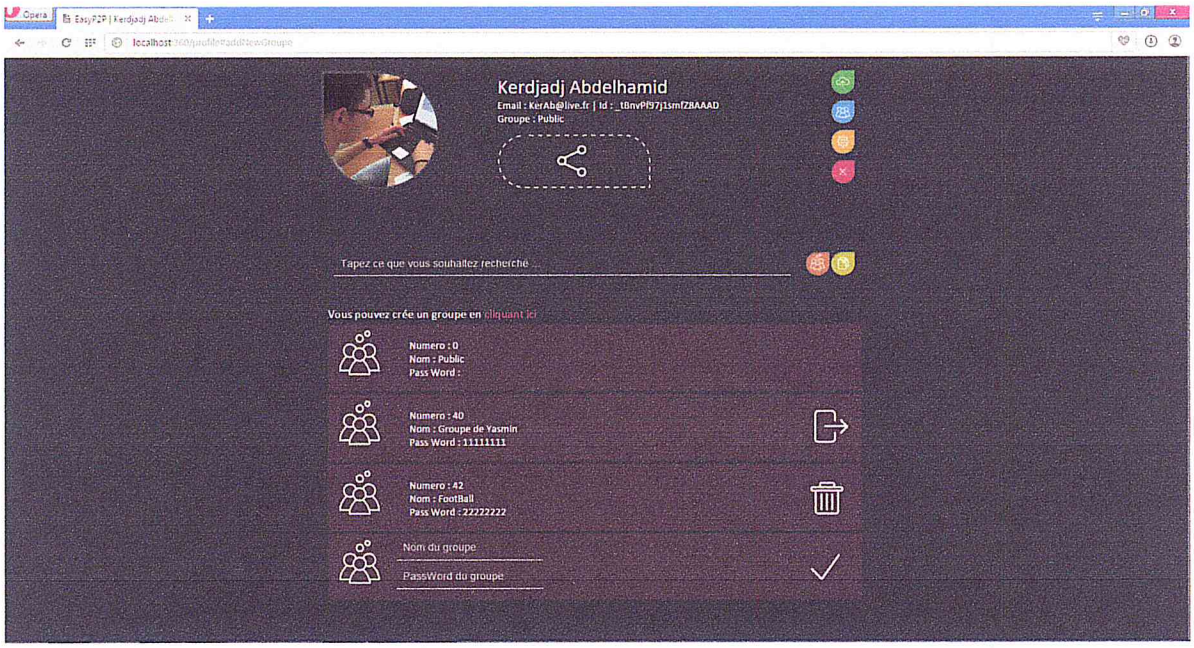


Figure IV.13 : gestion des groupes d'un utilisateur

Lorsque l'utilisateur clique sur un groupe il sera dirigé vers celui-ci (figure IV.14) et il pourra voir tous les fichiers qu'il a partagés dans ce groupe aussi que tous les utilisateurs connectés de ce groupe. Il a alors la possibilité de chercher des fichiers dans ce groupe ou de discuter avec un utilisateur du groupe.

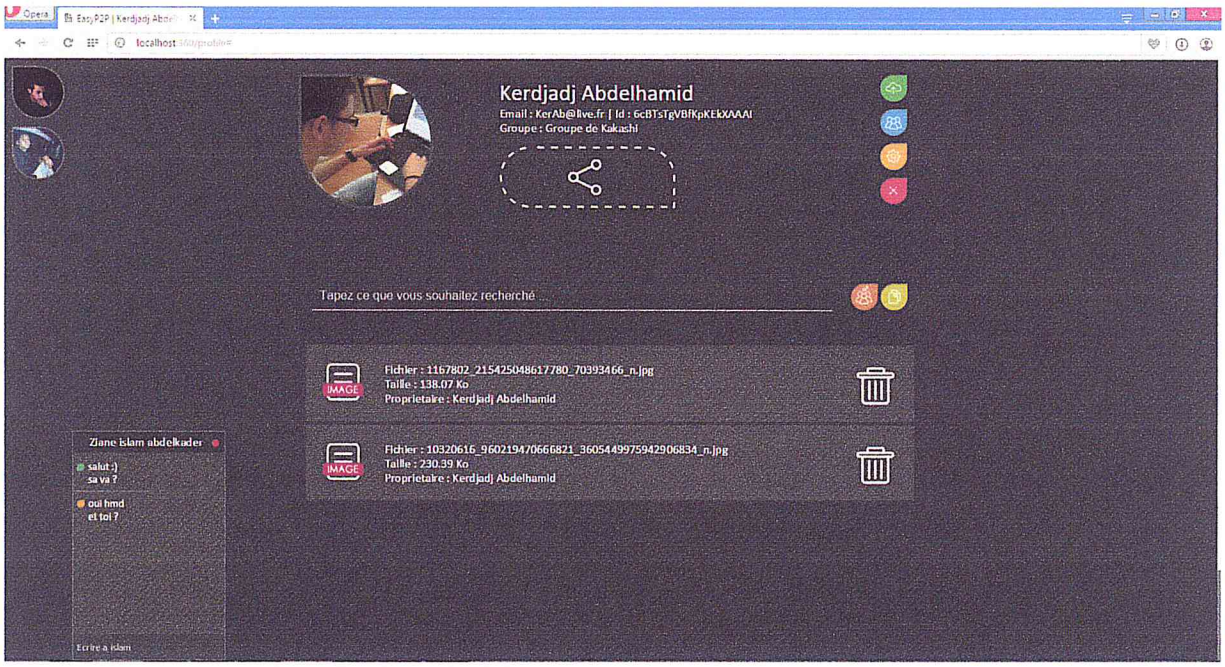


Figure IV.14 : interface d'un groupe

3.3.2. Profil administrateur

- **Page d'accueil**

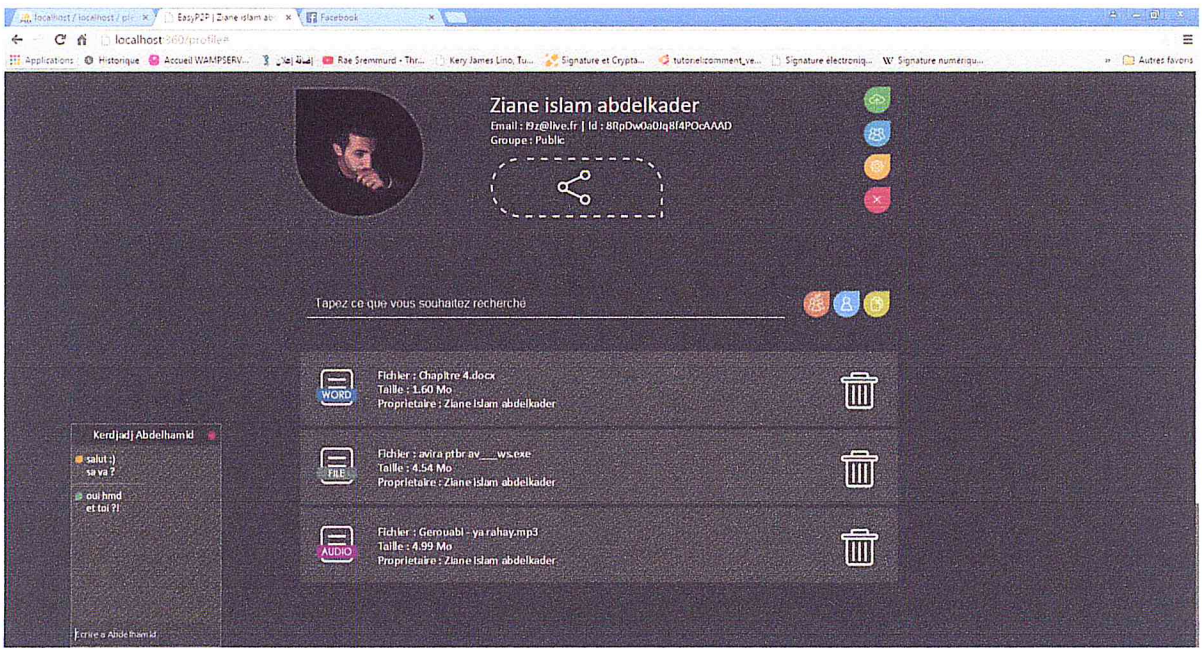


Figure IV.15 : page d'accueil de l'administrateur « Ziane Islame Abdelkader »

Lorsqu'un administrateur se connecte, il aura son propre profil exactement comme un utilisateur simple (Figure IV.15), qui contiendra ses informations ainsi que les fichiers qu'il a déjà partagés. Les administrateurs ont cependant la possibilité d'effectuer une recherche sur les utilisateurs en cliquant sur le petit bouton bleu.

- **Gestion des comptes utilisateurs**

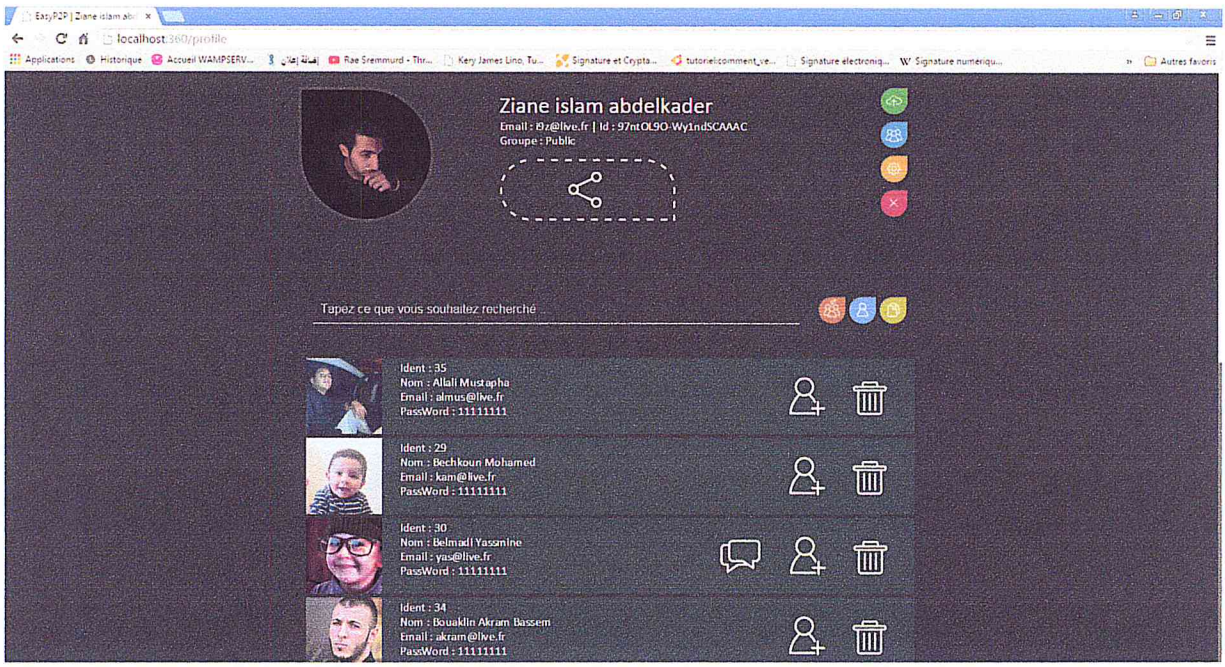


Figure IV.16 : Gestion des comptes utilisateurs

Lorsqu'un administrateur clique sur le bouton recherche pour utilisateurs, si la barre de recherche est vide, c'est-à-dire qu'il n'a rien tapé, il aura la liste de tous les utilisateurs (Figure IV.16), et pour chacun d'eux il aura trois choix : supprimer le profil d'utilisateur, rendre cet utilisateur comme administrateur, et chater avec celui-ci s'il est connecté. Les mêmes opérations sont possibles si l'administrateur recherche un utilisateur précis en tapant son nom d'utilisateur dans la barre de recherche.

➢ Recherche fichier ou groupe

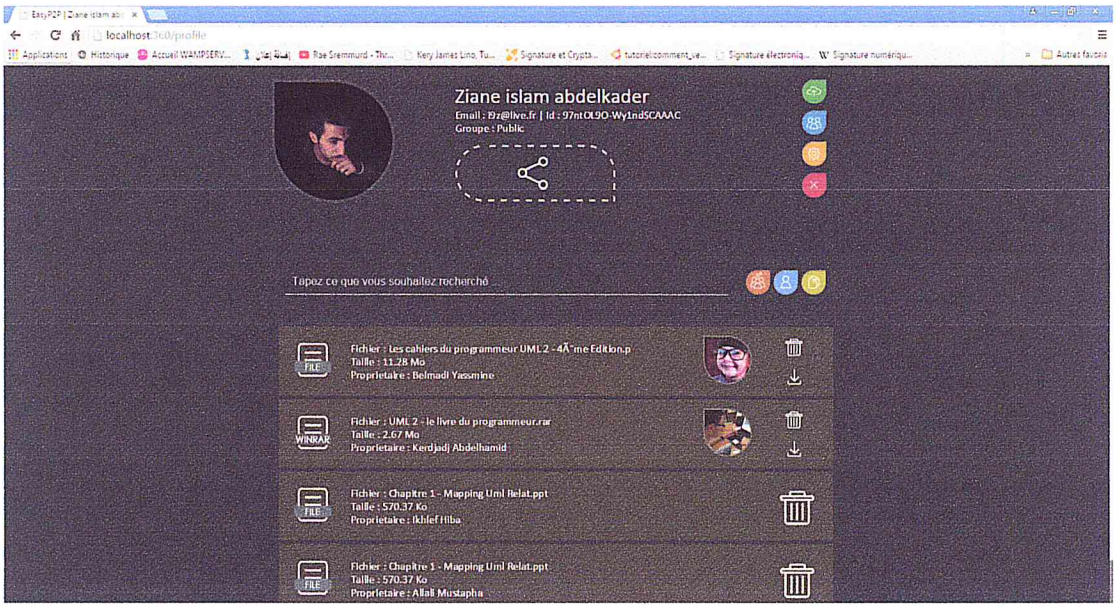


Figure IV.17 : Recherche de fichier « UML » dans un profile administrateur

Après l'affichage des résultats de recherche comme le montre la figure IV.17, l'administrateur a deux choix pour chaque fichier : le télécharger ou le supprimer de la base de données (et non pas de la machine de l'utilisateur qui l'a partagé).

De même pour les groupes, il a deux choix : soit s'authentifier et rentrer dans le groupe soit le supprimer complètement (figure IV.18)

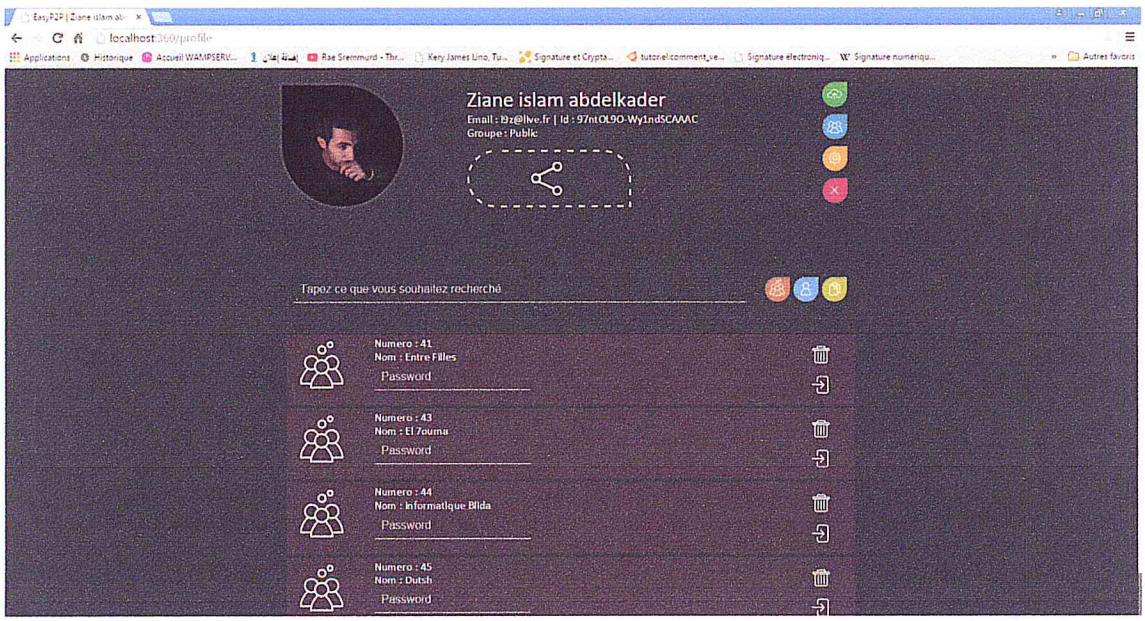


Figure IV.18 : recherche d'un groupe dans un profile administrateur

4. Conclusion

Nous avons présenté dans ce chapitre les détails techniques liés à la mise en œuvre de notre application web EasyP2P. Nous avons ainsi utilisé Node.js pour la partie serveur, les websockets pour la communication entre les navigateurs et le serveur, et enfin EasyRTC pour la communication peer-to-peer entre les navigateurs.

La fin du chapitre a été consacrée à la présentation des différentes pages et fonctionnalités de l'application EasyP2P.

Conclusion générale

Dans le cadre de ce travail, un nouveau concept pour le partage de fichiers a été proposé, conçu et mis en œuvre. Il s'agit d'une application web qui permet de partager des fichiers avec d'autres internautes en utilisant une communication peer-to-peer, et qui ne nécessite qu'un simple navigateur internet (disponible aujourd'hui sur la plupart des ordinateurs) pour pouvoir être utilisée. En effet, contrairement aux anciennes approches qui nécessitaient l'installation de plugins spécifiques pour pouvoir bénéficier d'une communication peer-to-peer entre deux navigateurs, l'application web EasyP2P que nous avons réalisée utilise les technologies web temps réel incorporées récemment dans les navigateurs web. Ces technologies incluent le protocole Websocket pour la communication bidirectionnelle entre un navigateur et un serveur, et l'API WebRTC permettant la communication peer-to-peer entre deux navigateurs.

Pour la conception de l'application EasyP2P, 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 opté pour une solution basée sur JavaScript. L'utilisation de ce langage permet d'avoir un code léger que ce soit du côté client et du côté serveur assurant ainsi de hautes performances pour notre application web. Plus précisément, nous avons utilisé Node.js pour la mise en œuvre du serveur, la librairie socket.io pour l'utilisation des Websockets et enfin la bibliothèque EasyRTC pour la communication peer-to-peer.

Bien que notre application EasyP2P ne soit qu'à sa première version, elle est néanmoins totalement fonctionnelle et offre un certain nombre de fonctionnalités intéressantes telles que :

- La possibilité de créer des groupes pour un partage restreint des fichiers.
- La possibilité de discuter (chater) avec les utilisateurs d'un groupe.
- La vérification des fichiers similaires qui ne portent pas le même nom en utilisant la signature MD5 des fichiers.

Dans la prochaine version d'EasyP2P, nous envisageons de rajouter les fonctionnalités suivantes :

- ✓ La possibilité de télécharger un fichier depuis plusieurs pairs en parallèle dans le cas où le fichier est proposé par plusieurs utilisateurs. En effet, dans sa version actuelle EasyP2P ne récupère un fichier qu'à partir d'un seul utilisateur.
- ✓ La signature des fichiers en MD5 est calculée par le navigateur ce qui fait que lorsque le fichier uploadé a une taille dépassant les 10 Mo, un ralentissement significatif des performances du navigateur sont observées. Il serait donc intéressant d'utiliser un autre algorithme qui soit plus efficace en termes de rapidité de calcul.

Liste des abréviations

DOM : Document Object Model

ICE : Interactive Connectivity Establishment

STUN : Session Traversal Utilities for NAT

TURN : Traversal Using Relays around NAT

NAT : Network Address Translation

SCTP : Stream Control Transmission Protocol

DTLS: Datagram Transport Layer Security

UML : Unified Modeling Language

WAE : Web Application Extension

UP : Unified Process.

SQL : Structured Query Language

SGBD : système de gestion de bases de données relationnelles

MD5 : Message Digest

PGP : pretty good privacy

Bibliographie

[1] Jean-Noël Anderruthy, Skype et la téléphonie IP: téléphonez gratuitement avec Internet, Editions ENI, 2006

[2] Nathalie BUDAN, Benoit TEDESCHI ; Nouvelles Technologies Réseau, Les réseaux peer-to-peer ; http://igm.univ-mlv.fr/~duris/NTREZO/20022003/Peer-to-peer.pdf

[3] Dimitri Caron, Dominique Poure ; L'empire du « Peer to peer », Université Paris-Dauphine. http://www.portices.fr/formation/Res/Internet/P2p/P2p-Web.pdf. 2007

[4] Anne Benoit ; Réseaux Pair à Pair, ENS Lyon. http://graal.ens-lyon.fr/~abenoit/reso05/cours/2-p2p.pdf.

[5] Thierry Cumps, Les bases de l'informatique et de l'internet, Lulu.com

[6] L'échange de fichiers dans les réseaux Pair-à-Pair, Université Nice Sophia-Antipolis, « THUAUX Anthony, SOUSA LOPES Eric ».

[7] Pierre GUILLOU; http://www.ideose.com/comprendre-web-temps-reel/; 2009

[8] Luc Van Lancker ; Des CSS au DHTML: JavaScript appliqué aux feuilles de style, Ediciones ENI, 2008.

[9] Mathieu Nebra, http://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js/node-js-mais-a-quoi-ca-sert,2015

[10] Bruno Catteau, Nicolas Faugout ; Ajax: le guide complet ; MA éditions, 2009.

[11] Jesse James Garrett; Ajax: A New Approach to Web Applications; http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

[12] Mathieu Nebra, http://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js/les-modules-node-js-et-npm,2015.

[13] Andrii Sergiienko ; WebRTC Cookbook ; Packt Publishing Ltd, 2015

[14] Varun Chopra, WebSocket Essentials – Building Apps with HTML5 WebSockets, Packt Publishing Ltd, 2015

[15] Salvatore Loreto, Simon Pietro Romano; Real-Time Communication with WebRTC: Peer-to-Peer in the Browser; "O'Reilly Media, Inc.", 2014

[16] Álvaro Rocha, Ana Maria Correia, Felix . B Tan, Karl . A Stroetmann; New Perspectives in Information Systems and Technologies, Volume 2; Springer Science & Business Media; 2014.

[17] Pascal Roques ; UML 2: modéliser une application web, Les Cahiers du programmeur ; Eyrolles, 2008.

[18] Jim Conallen; Building Web Applications with UML Second Edition; Addison Wesley, 2002

[19] Dreamweaver : logiciel incontournable, http://www.web-libre.org/dossiers/dreamweaver,9018.html

[20] Chantal Gribaumont ; Administrez vos bases de données avec MySQL (2ème édition) ; Livre du Zéro, 2014.

[21] http://easyrtc.com/

[22] Bruno Martin; Codage, cryptologie et applications; PPUR presses polytechniques, 2004.