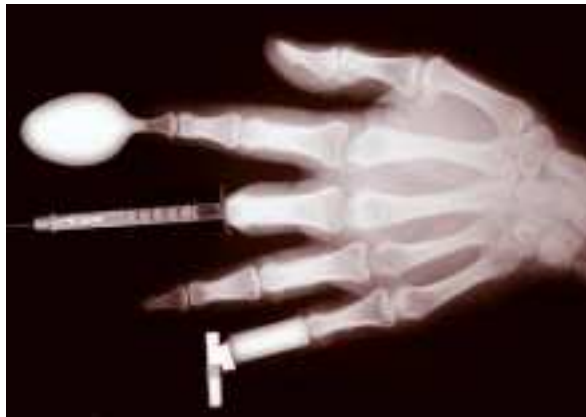


Master 1 Informatique Professionnel et Recherche Unifié

# Réalisation d'un logiciel de Visioconférence

Rapport final



Réalisé par :

ESCANDELL Frédéric, NOUAILLES Yoann, MORA Michaël et QUINTON Clément

Projet encadré par Anne-Elisabeth BAERT et Vincent BOUDET

Année universitaire 2007-2008

# Remerciements

Nous tenons à remercier tout particulièrement nos tuteurs, Mme Baert Anne-Elisabeth et M. Boudet Vincent, pour leur aide précieuse et leur disponibilité tout au long du développement de notre projet.

Nous avons pu compter sur leur présence et leur soutien à tout moment, que ce soit par mail ou lors de nos rencontres au LIRMM ou à l'université.

Nous remercions également M. Sapede Frédéric, Ingénieur Réseaux et Sécurité chez Expertise Radiologie, pour son implication et ses conseils prodigués lors des différents problèmes techniques rencontrés. Son écoute et sa patience nous ont grandement servi.

# Glossaire

**API :** Application Programming Interface ou API. Une interface de programmation permet de définir la manière dont un composant informatique peut communiquer avec un autre. C'est donc une interface de code source fournie par un système informatique ou une bibliothèque logicielle, en vue de répondre à des requêtes pour des services qu'un programme informatique pourrait lui faire. La connaissance des API est indispensable à l'interopérabilité entre les composants logiciels.

**Classpath :** chemin d'accès au fichier .class ou aux bibliothèques nécessaires au programme.

**Code natif :** un programme informatique en code natif (ou langage machine) est composé d'instructions directement reconnues par un processeur, à la différence des programmes écrits en suivant les conventions d'un langage de programmation (langage d'assemblage, ou langage de haut niveau comme C, C++, Pascal. . .). Certains compilateurs produisent du bytecode au lieu du code natif (voir Java).

**Java :** Java est à la fois un langage de programmation informatique orienté objet et un environnement d'exécution informatique portable créé par James Gosling et Patrick Naughton employés de Sun Microsystems. Il fut présenté officiellement le 23 mai 1995 au SunWorld. Le langage Java a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels que Unix, Microsoft Windows, Mac OS ou Linux avec peu ou pas de modifications.

**JDBC :** Java DataBase Connectivity. Cette API est constituée d'un ensemble d'interfaces et de classes qui permettent l'accès, à partir de programmes java, à des données tabulaires (i.e. triées sous forme de table ou de tableur). Par données tabulaires, on entend généralement des bases de données contenues dans des SGBD relationnels. Mais JDBC n'est pas restreinte à ce type de source de données. On peut aussi accéder à des sources de données sous forme de fichiers (fichiers XML par exemple).

**JID :** le « Jabber ID » ou « Jabber IDentifier » (identifiant Jabber) est l'adresse Jabber de la forme loginclient@jabberserver.fr.

**JVM** : la Java Virtual Machine (abrégé JVM, en français Machine virtuelle Java) est une machine virtuelle permettant d'interpréter et d'exécuter le bytecode Java. Ce programme est spécifique à chaque plate-forme ou couple machine/système d'exploitation et permet aux applications Java compilées en bytecode de produire les mêmes résultats quelle que soit la plate-forme, tant que celle-ci est pourvue de la machine virtuelle Java adéquate. La machine virtuelle la plus utilisée est celle de Sun Microsystems. Elle est gratuite, propriétaire jusqu'à la version 6 (stable) et libre à partir de la version 7 (non encore officielle).

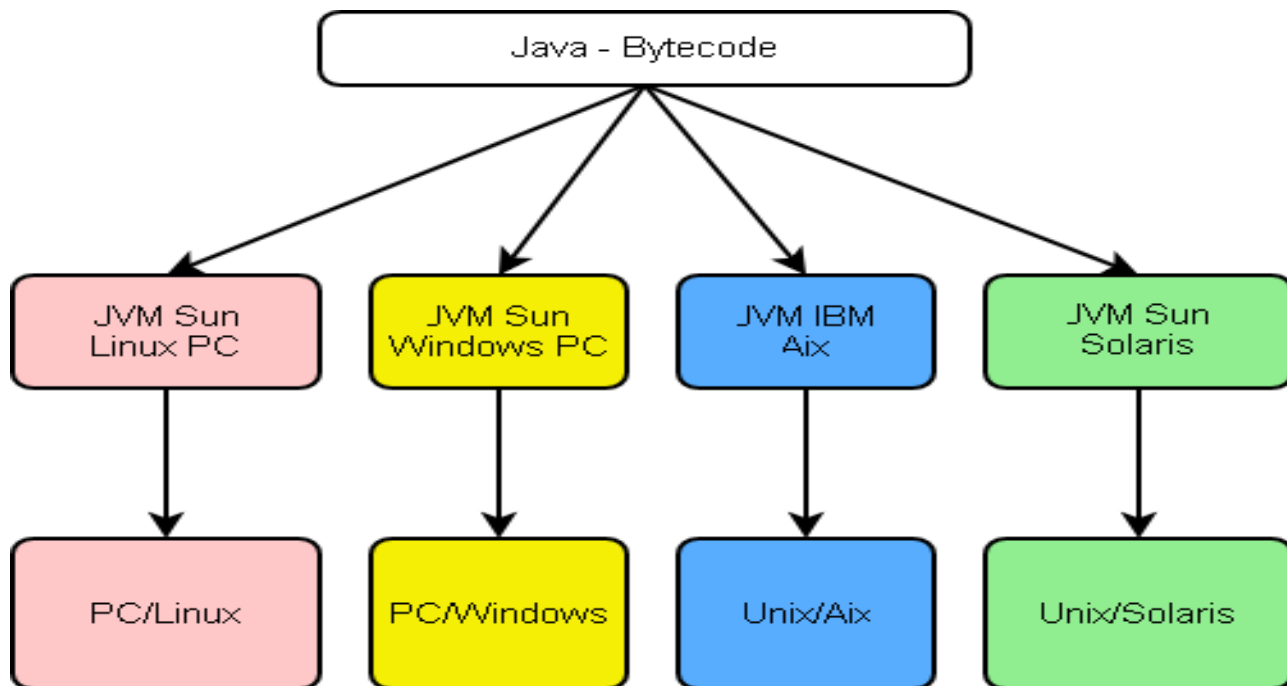


FIG. 1 – Architecture générale : illustration de l'adage *Compile once, run everywhere*

**OSI (modèle) :** le modèle OSI (de l'anglais Open Systems Interconnection, « Interconnexion de systèmes ouverts ») d'interconnexion en réseau des systèmes ouverts est un modèle de communications entre ordinateurs proposé par l'ISO (Organisation internationale de normalisation). Il décrit les fonctionnalités nécessaires à la communication et l'organisation de ces fonctions. Le modèle OSI comporte 7 couches.

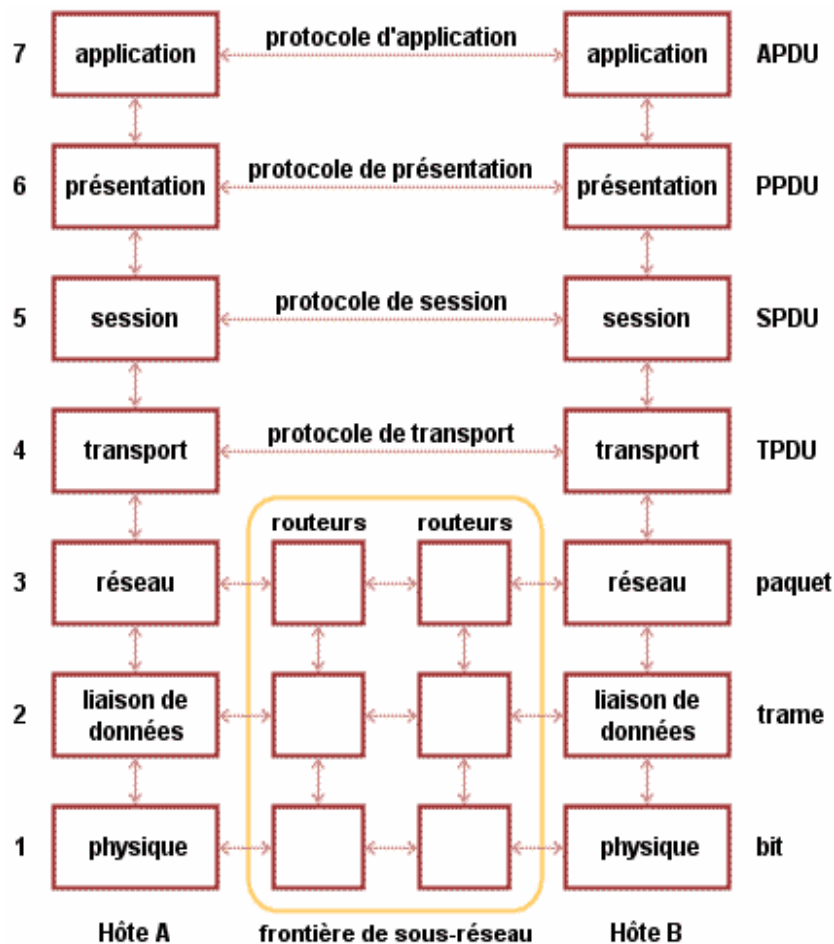


FIG. 2 – Les 7 couches du modèle OSI

**Protocole :** langage normalisé permettant à des logiciels différents de communiquer. Le protocole sur lequel repose Jabber s'appelle XMPP. Le client se charge de la traduction entre le langage humain et le protocole.

**RFC :** les Request For Comment (RFC, littéralement demande de commentaires) sont une série de documents et normes concernant Internet, commencée en 1969. Peu de RFC sont des standards, mais tous les standards de Internet sont enregistrés en tant que RFC. Les RFC sont rédigées sur l'initiative d'experts techniques, puis sont revues par la communauté Internet dans son ensemble. Pour cette raison, elles continuent à être appelées RFCs une fois devenues des standards.

**Roster** : on désigne ainsi la liste des adresses Jabber (ou JID) enregistrées par un utilisateur. On l'appelle parfois « liste de contacts » mais c'est incorrect parce que cette liste peut aussi inclure des adresses de passerelles, de salons de discussion, etc. Prononcer « rosteur ».

**RTCP** : Real-time Transfert Control Protocole. Le protocole RTCP est basé sur des transmissions périodiques de paquets de contrôle par tous les participants dans la session. C'est un protocole de contrôle des flux RTP, permettant de véhiculer des informations basiques sur les participants d'une session, et sur la qualité de service.

**RTP** : Real-time Transfert Protocole. Le but de RTP est de fournir un moyen uniforme de transmettre sur IP des données soumises à des contraintes de temps réel (audio, vidéo, ...). Le rôle principal de RTP consiste à mettre en oeuvre des numéros de séquence de paquets IP pour reconstituer les informations de voix ou vidéo même si le réseau sous-jacent change l'ordre des paquets.

**TCP** : le Transmission Control Protocol (TCP, « protocole de contrôle de transmissions »), est un protocole de transport fiable, en mode connecté, documenté dans la RFC 793 de l' IETF. Dans le modèle TCP/IP, TCP est situé au niveau de la couche transport (entre la couche de réseau et la couche application). Les applications transmettent des flux d'octets sur le réseau. TCP découpe le flux d'octets en segments, dont la taille dépend de la MTU du réseau sous-jacent (couche liaison de données).

**UDP** : Le User Datagram Protocol (UDP, en français protocole de datagramme utilisateur) est un des principaux protocoles de télécommunication utilisé par Internet. Il fait partie de la couche transport (voir modèle OSI ci-dessus) et est détaillé dans la RFC 768.

Le rôle de ce protocole est de permettre la transmission de paquets de manière très simple entre deux entités, chacune étant définie par une adresse IP et un numéro de port (pour différencier différents utilisateurs sur la même machine).

**SWING** : Swing est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package Java Foundation Classes (JFC), inclus dans J2SE. Swing constitue l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures.

Elle offre la possibilité de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent, au prix de performances moindres qu'en utilisant Abstract Window Toolkit (AWT). Elle utilise le principe Modèle/View-Contrôleur (MVC, les composants Swing jouent en fait le rôle de Contrôleur au sens du MVC) et dispose de plusieurs choix d'apparence (de vue) pour chacun des composants standard.

**XEP** : XEP signifie « XMPP Extension Protocol ». Une XEP est une norme définissant une extension au protocole XMPP (en général, une nouvelle fonctionnalité pour Jabber). Le terme XEP a récemment remplacé le terme JEP.

**XML :** XML (eXtensible Markup Language, « langage de balisage extensible ») est un langage informatique de balisage générique. Son objectif initial est de faciliter l'échange automatisé de contenus entre systèmes d'informations hétérogènes.

**XMPP :** c'est le nom du protocole sur lequel repose Jabber. Il est normalisé par l'IETF, l'organisme qui normalise l'ensemble des protocoles de l'Internet.

**vCard :** une vCard est une carte de visite électronique. Elle contient des informations sur l'identité de l'individu telles que son nom, son prénom, ses coordonnées, son métier, des commentaires, une photo, etc.

Jabber permet d'associer à chaque compte utilisateur une vCard. Les vCards sont stockées sur le serveur et peuvent être retrouvées par tout individu, même si le propriétaire de la vCard n'est pas connecté et qu'il ne fait pas partie de ses contacts.

La XEP 0054 (vCard-temp) définit la façon dont les utilisateurs peuvent publier une vCard et en retrouver une.

**Visioconférence :** on nomme visioconférence la combinaison de deux techniques :

- la visiophonie ou vidéotéléphonie, permettant de voir et dialoguer avec son interlocuteur ;
- la conférence multipoints ou « conférence à plusieurs », permettant d'effectuer une réunion avec plus de deux terminaux.

Dans la pratique, le terme visioconférence reste toutefois utilisé même lorsque les interlocuteurs ne sont que deux.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>12</b>
<b>2</b>	<b>Présentation d'Expertise Radiologie</b>	<b>13</b>
<b>3</b>	<b>Matériel et méthodes</b>	<b>15</b>
3.1	Cahier des charges . . . . .	15
3.1.1	Planning général du projet . . . . .	15
3.1.2	Exigences sur la capacité du logiciel . . . . .	16
3.1.3	Fonctionnalités nécessaires pour la visioconférence . . . . .	17
3.1.4	Fonctionnalités nécessaires pour le tableau blanc . . . . .	17
3.1.5	Contraintes sur la sûreté du logiciel . . . . .	17
3.1.6	Contraintes sur l'environnement du logiciel . . . . .	18
3.1.7	Contraintes de conception . . . . .	18
3.1.8	Facteurs de qualité du logiciel . . . . .	18
3.2	Organisation du groupe . . . . .	19
3.2.1	Rythme des réunions au sein du groupe et avec nos tuteurs . . . . .	19
3.2.2	Division des tâches . . . . .	19
3.3	Méthodes et outils nécessaires / à disposition . . . . .	21
3.4	État de l'art . . . . .	25
3.4.1	Analyse et description des différents logiciels de visioconférence . . . . .	25
3.4.2	Système de présence et de messagerie . . . . .	28
3.4.3	Visioconférence . . . . .	32
3.4.4	Le tableau blanc . . . . .	34
3.5	Réalisation . . . . .	35
3.5.1	Visioconférence . . . . .	35
3.5.2	Le chat . . . . .	38
3.5.3	Le tableau blanc . . . . .	39
<b>4</b>	<b>Résultats</b>	<b>41</b>



4.1	L'interface principale . . . . .	41
4.1.1	L'écran d'accueil . . . . .	41
4.1.2	L'interface . . . . .	43
4.2	Visioconférence . . . . .	46
4.2.1	Paramétrages . . . . .	46
4.2.2	Affichage de la vidéo . . . . .	47
4.3	Le chat . . . . .	48
4.4	Le tableau blanc . . . . .	54
4.5	Fonctionnalités supplémentaires . . . . .	58
4.5.1	Rendez-vous . . . . .	58
4.5.2	Mise à jour automatique . . . . .	60
4.5.3	Internationalisation . . . . .	62
4.5.4	Packaging de l'application . . . . .	64
4.6	Divers : installation d'Openfire sur le serveur d'Expertise Radiologie . . . . .	66
<b>5</b>	<b>Discussion</b>	<b>67</b>
<b>6</b>	<b>Conclusion</b>	<b>68</b>
<b>7</b>	<b>Bibliographie</b>	<b>69</b>
<b>A</b>	<b>Annexes</b>	<b>70</b>
A.1	Analyse et conception . . . . .	70
A.1.1	Diagramme de classes . . . . .	71
A.1.2	Diagramme de cas d'utilisation . . . . .	72
A.1.3	Diagramme de déploiement . . . . .	73
A.2	Fichier XML de paramétrage d'ANT . . . . .	74
A.3	Manuel de l'utilisateur . . . . .	75
A.3.1	Comment se connecter ? . . . . .	75
A.3.2	Comment se déconnecter ? . . . . .	75
A.3.3	Comment démarrer une conversation ? . . . . .	76
A.3.4	Comment interagir avec le tableau blanc ? . . . . .	77
A.3.5	Comment créer un nouveau compte Jabber ? . . . . .	78
A.3.6	Comment ajouter un contact ? . . . . .	79
A.3.7	Comment consulter l'aide ? . . . . .	79
A.3.8	Comment planifier un rendez-vous ? . . . . .	80
A.3.9	Comment changer la langue du logiciel ? . . . . .	81

A.3.10	Comment changer de login ? . . . . .	81
A.3.11	Comment mettre à jour le logiciel ? . . . . .	82
A.3.12	Comment changer de statut ? . . . . .	82
A.3.13	Comment changer le look and feel de l'application ? . . . . .	83
A.4	Compte rendu . . . . .	84

# Table des figures

1	Architecture générale : illustration de l'adage <i>Compile once, run everywhere</i> . . . . .	3
2	Les 7 couches du modèle OSI . . . . .	4
2.1	Fonctionnement d'Expertise Radiologie . . . . .	13
2.2	Le double service . . . . .	14
3.1	Diagramme de Gantt . . . . .	15
3.2	Croquis de l'interface . . . . .	16
3.3	Exemple de VPN d'accès . . . . .	18
3.4	Java . . . . .	21
3.5	Subversion . . . . .	22
3.6	Debian . . . . .	23
3.7	Trac . . . . .	23
3.8	Eclipse . . . . .	24
3.9	Netbeans . . . . .	24
3.10	Une partie du modèle OSI . . . . .	32
3.11	Acquisition de la vidéo . . . . .	33
3.12	Paint . . . . .	34
3.13	L'application après intégration de la vidéo . . . . .	35
4.1	Affichage de l'écran d'accueil . . . . .	41
4.2	L'interface principale . . . . .	44
4.3	Les bulles d'aide . . . . .	45
4.4	Détection des périphériques audio et/ou vidéo . . . . .	46
4.5	Paramètres du flux vidéo . . . . .	46
4.6	Affichage de la vidéo . . . . .	47
4.7	Console . . . . .	48
4.8	Conversation dans un panel . . . . .	49
4.9	Contacts et leur JID . . . . .	49
4.10	Utilité de la vCard . . . . .	51

4.11	Les changements de statut chez les contacts . . . . .	52
4.12	Le tableau blanc à ses débuts . . . . .	54
4.13	Le dessin de formes dans le tableau blanc . . . . .	55
4.14	Un tableau blanc plus évolué . . . . .	56
4.15	Le tableau blanc et le dessin à main levée . . . . .	57
4.16	Le formulaire de prise de rendez-vous . . . . .	58
4.17	Le rendez-vous est pris . . . . .	59
4.18	L'écran d'accueil en français et en anglais . . . . .	62
4.19	Le dictionnaire de l'écran d'accueil . . . . .	63
4.20	Fonctionnement du lanceur . . . . .	64
4.21	Les différentes fenêtres de l'installation . . . . .	65
A.1	Diagramme de classes . . . . .	71
A.2	Diagramme de cas d'utilisation . . . . .	72
A.3	Diagramme de Déploiement . . . . .	73
A.4	Fichier XML de paramétrage d'ANT . . . . .	74
A.5	Connexion . . . . .	75
A.6	Déconnexion . . . . .	75
A.7	Une conversation . . . . .	76
A.8	Les dessins sur le tableau blanc . . . . .	77
A.9	Le bouton <i>Compte</i> . . . . .	78
A.10	Le formulaire . . . . .	78
A.11	Ajout d'un contact . . . . .	79
A.12	L'icône d'aide . . . . .	79
A.13	L'icône <i>RDV</i> . . . . .	80
A.14	Le formulaire de saisie . . . . .	80
A.15	L'onglet <i>RDV</i> . . . . .	80
A.16	Changement de la langue . . . . .	81
A.17	Changement de login . . . . .	81
A.18	Mise à jour de l'application . . . . .	82
A.19	Changement de statut . . . . .	82
A.20	Changement du look and feel . . . . .	83

# 1 Introduction

Le but de ce TER est de développer un logiciel de visioconférence interactif. Cet outil doit permettre d'établir une communication entre un radiologue et un patient lorsque celui-ci se trouve chez son médecin.

Ce dialogue doit permettre au radiologue d'éclairer les patients sur leurs interrogations et de fournir une explication plus précise au médecin l'accompagnant, grâce au support que représente la radiographie numérique.

Un travail sur la présentation et la disposition des divers modules de l'application est nécessaire afin de rendre l'interface la plus intuitive possible, pour permettre à tout utilisateur de prendre en main le logiciel sans difficulté.

Cet outil se doit donc d'être ergonomique et facile d'utilisation. De nombreux logiciels en tout genre sont aujourd'hui disponibles et peuvent éventuellement répondre en partie aux contraintes posées par le sujet, c'est pourquoi ce TER débute par une étude approfondie de l'existant.

## 2 Présentation d'Expertise Radiologie



FIG. 2.1 – Fonctionnement d'Expertise Radiologie

Expertise Radiologie est une jeune entreprise créée par Vincent Costala et Frédéric Sapède, proposant un service de télé-radiologie (Figure 2.1). L'entreprise a pour objectif de mettre en relation des clients (radiologue, expert médical, patient) et de leur fournir un double service : un service de télé-expertise et un service de télé-diagnostic (Figure 2.2).

La télé-expertise permet de solliciter l'avis d'un expert sur un diagnostic médical pré-établi par le biais d'images médicales numérisées.

Le télé-diagnostic permet à un radiologue d'établir un diagnostic depuis un poste à distance.

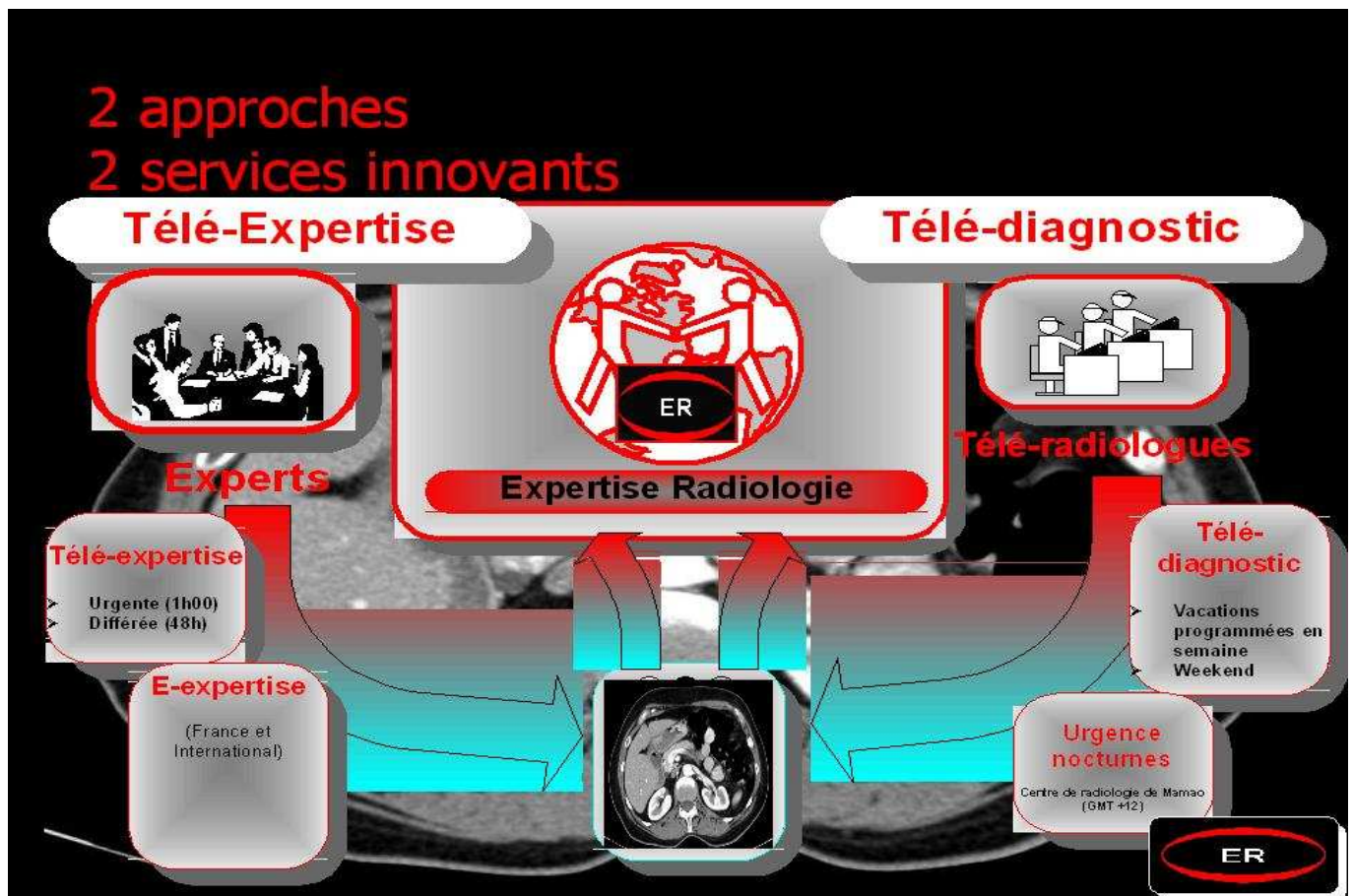


FIG. 2.2 – Le double service

Notre application doit intervenir dans le cadre du service de télé-diagnostic, c'est à dire de la communication radiologue/centre de radiologie et radiologue/patient. On distingue donc à ce stade deux profils d'utilisation différents. L'objectif de notre logiciel est de proposer une interface la plus interactive possible grâce à l'utilisation de la visioconférence.

# 3 Matériel et méthodes

## 3.1 Cahier des charges

### 3.1.1 Planning général du projet

L'illustration ci-dessous présente les différentes phases du cycle de vie en V (i.e. toute étape doit être achevée avant de passer à l'étape suivante) de notre application de visioconférence.

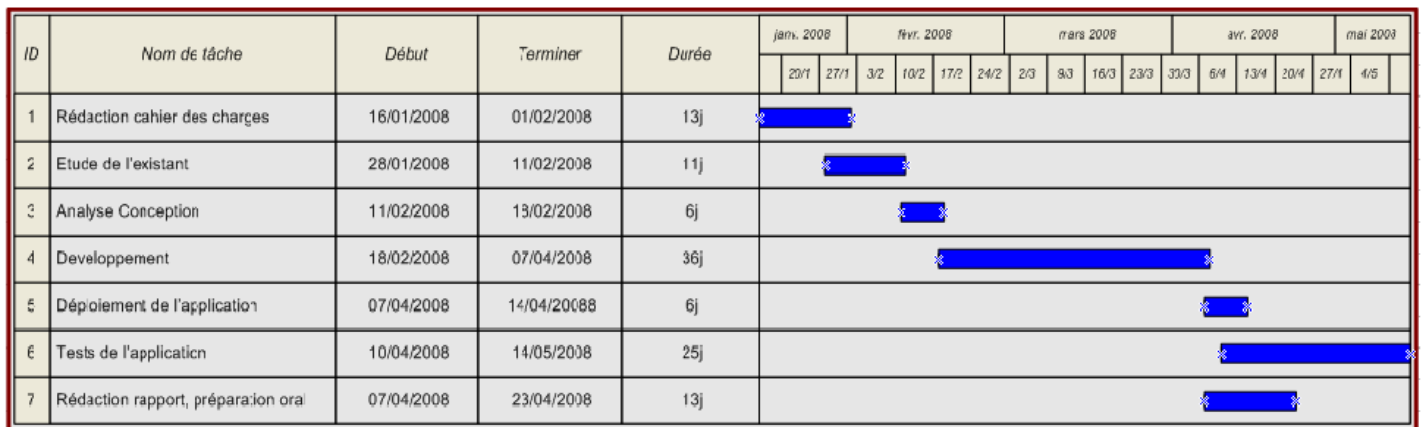


FIG. 3.1 – Diagramme de Gantt



### 3.1.2 Exigences sur la capacité du logiciel

Ce logiciel a pour objectif de réaliser une visioconférence entre deux personnes. Ces personnes peuvent se voir mutuellement et converser via un support audio. Une fonctionnalité complémentaire doit permettre aux docteurs de réaliser des annotations en temps réel sur une radiographie numérique visible sur chaque ordinateur distant.

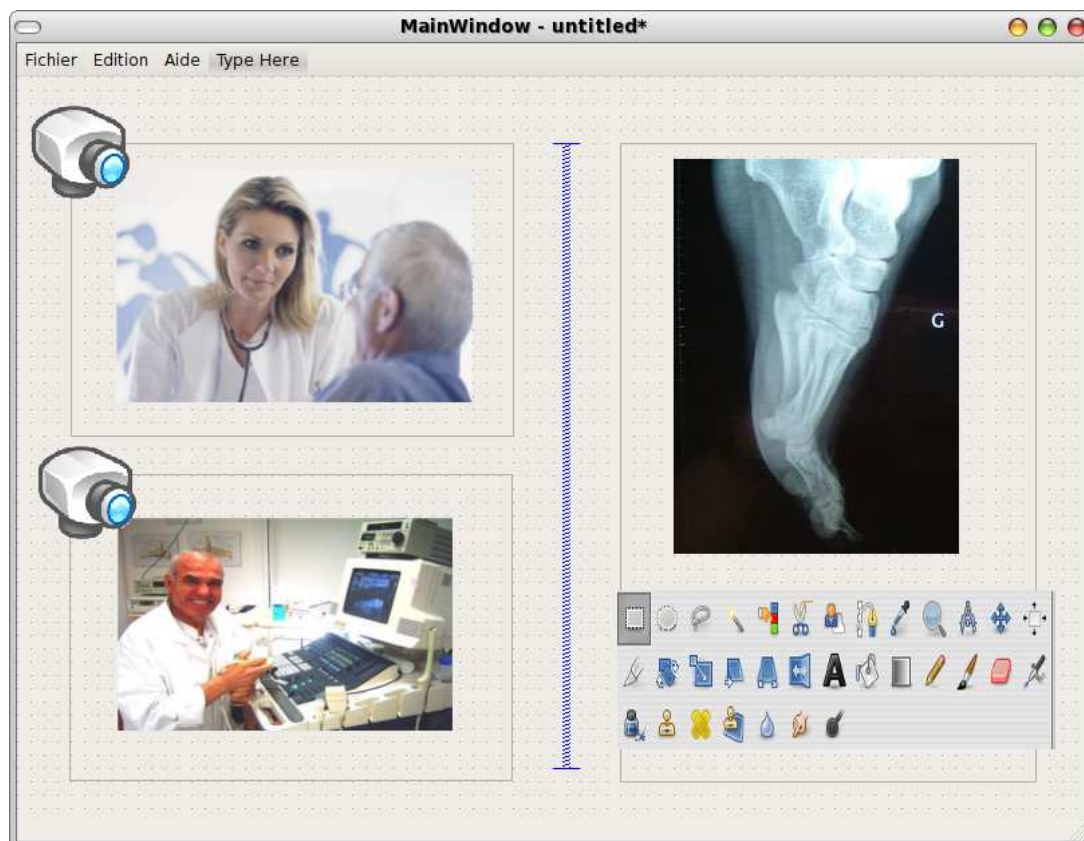


FIG. 3.2 – Croquis de l'interface

### **3.1.3 Fonctionnalités nécessaires pour la visioconférence**

Le logiciel permet à deux personnes distantes de converser comme représenté dans l'illustration ci-dessus. Notre interface doit donc comprendre une partie dédiée à l'affichage des webcams correspondant au médecin et au radiologue. Elle doit également permettre une communication audio entre les deux interlocuteurs. Par exemple, un médecin travaillant à Pézénas peut demander le diagnostic d'un radiologue situé au CHU de Montpellier concernant son patient.

L'utilisateur dispose d'une liste de contacts et peut décider de lancer une visioconférence. L'invitation doit être acceptée par l'utilisateur destinataire pour démarrer la visioconférence. Chaque utilisateur doit avoir la possibilité de lancer plusieurs conversations privées (afin de conserver la confidentialité des patients) en parallèle.

Concernant l'interface du logiciel et d'après l'étude réalisée, nous avons pensé qu'il serait préférable d'implémenter deux types d'interface différents : une simpliste pour le médecin traitant lui permettant de communiquer avec le radiologue, une plus élaborée autorisant le radiologue à communiquer avec plusieurs médecins en laissant en attente ceux ne faisant pas partie de la communication courante. Ainsi, chaque IHM (Interface Homme Machine) est adaptée aux besoins de l'utilisateur.

### **3.1.4 Fonctionnalités nécessaires pour le tableau blanc**

Notre application doit fournir un panel d'outils pour réaliser des annotations sur une radiographie préalablement chargée par les deux utilisateurs. Les modifications effectuées sur la radiographie sont envoyées en temps réel à l'utilisateur avec lequel on converse.

### **3.1.5 Contraintes sur la sûreté du logiciel**

Le logiciel doit fonctionner sur le réseau VPN (Virtual Private Network) afin de préserver les données confidentielles échangées entre les médecins et leurs patients. Un réseau VPN repose sur un protocole appelé « protocole de tunneling » (Figure 3.3).

Ce protocole permet de faire circuler les informations de l'entreprise de façon cryptée d'un bout à l'autre du tunnel. Ainsi, les utilisateurs ont l'impression de se connecter directement sur le réseau de leur entreprise.

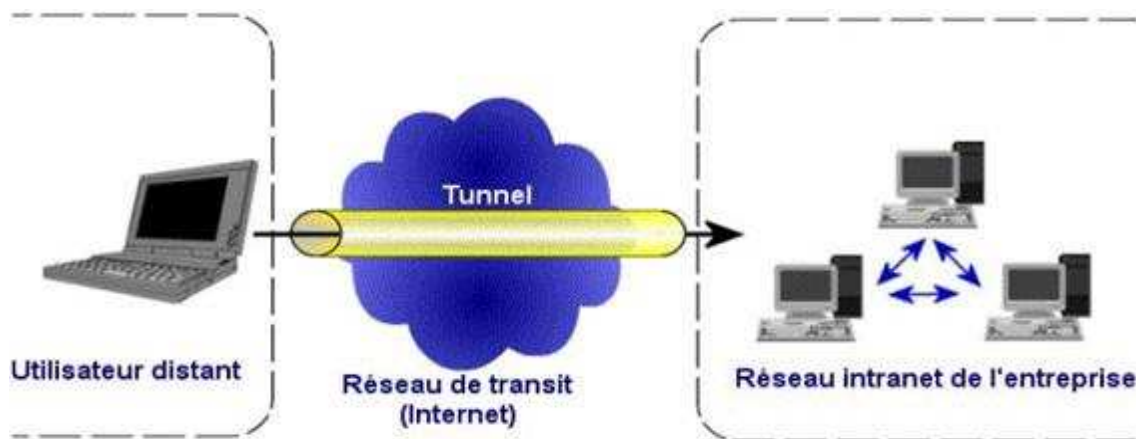


FIG. 3.3 – Exemple de VPN d'accès

Le principe de tunneling consiste à construire un chemin virtuel après avoir identifié l'émetteur et le destinataire. Par la suite, l'émetteur chiffre les données et les achemine en empruntant ce chemin virtuel.

### 3.1.6 Contraintes sur l'environnement du logiciel

Le logiciel sera développé afin de fonctionner sous Windows majoritairement et sous Linux éventuellement. Nous privilégions en effet les systèmes Windows Xp et Windows Vista, qui sont ceux utilisés par les médecins qui se serviront du logiciel. Cependant, il serait préférable d'obtenir un logiciel portable sur tout type de plateformes, vu la place grandissante prise par le logiciel libre dans le monde de l'informatique.

### 3.1.7 Contraintes de conception

Le logiciel devra s'intégrer facilement dans une future application web que développera la société. Il serait intéressant de prendre en compte dans la mesure du possible le type de machine sur laquelle tournera l'application (mono ou bi-cœur, 32 ou 64 bits), afin d'en tirer le meilleur profit possible en terme de performances, optimisation, etc...

### 3.1.8 Facteurs de qualité du logiciel

La société nous demande un logiciel convivial, c'est à dire un logiciel offrant une prise en main simple et rapide et au graphisme intuitif et agréable (limiter le nombre de boutons et d'options, ne pas surcharger l'interface graphique). Ce logiciel sera en effet utilisé par de nombreuses personnes, peu enclines à passer du temps à comprendre le fonctionnement de celui-ci, surtout quand on sait ce que le temps représente pour des médecins.

## 3.2 Organisation du groupe

### 3.2.1 Rythme des réunions au sein du groupe et avec nos tuteurs

Tous les étudiants de ce groupe étant dans le même cursus, l'emploi du temps nous a laissé assez de disponibilités pour faire des réunions régulières et travailler ensemble. D'autre part nous avons défini une réunion bimensuelle avec nos tuteurs afin de faire un point sur l'avancement du projet et valider nos choix.

### 3.2.2 Division des tâches

Dans un premier temps nous avons choisi de fonctionner de la manière suivante : chaque membre du groupe s'est focalisé sur l'étude d'un ou plusieurs outils existant(s) et le(s) a présenté aux autres membres du groupe.

Etudiant	Logiciels étudiés
ESCANDELL Frédéric	Clients Java réseau Jabber : JBrother API JAVA pour client Jabber : Smack API 3.0.4
NOUAILLES Yoann	VLVC
MORA Michaël	Openwengo
QUINTON Clément	Windows Live Messenger Skype

Cette manière de procéder nous a permis d'avoir une vision plus globale de l'état d'art et d'envisager une possible intégration d'un outil existant. Une fois cette étude terminée et les choix retenus pour la conception, nous avons établi une nouvelle division des tâches.

Dans un second temps, et une fois les choix de conception établis, nous avons pu nous lancer dans l'implémentation du logiciel. Pour cela nous avons mis en place une interface graphique générale à laquelle sont venus se greffer les différents modules de l'application (chat, tableau blanc, fenêtre vidéo). Ceci étant fait, nous avons décidé de diviser le développement de l'application en plusieurs parties distinctes, chaque membre du groupe ayant la charge d'implémenter un de ces modules. La division des tâches s'est effectuée comme suit :

Etudiant	Réalisation
ESCANDELL Frédéric	Modules audio et vidéo
NOUAILLES Yoann MORA Michaël	Implémentation du protocole Jabber(Smack), modules de gestion des contacts et chat
QUINTON Clément	Module du tableau blanc

Cette façon de travailler nous a permis de gagner en efficacité et en rapidité car la partie intégration de code était quasiment nulle, grâce à l'utilisation du serveur SVN. De plus, même si les différents composants ont été développés individuellement, chaque membre du groupe a fait un compte rendu régulier de son travail aux autres. Les différents problèmes que nous avons rencontrés ont ainsi été résolus en groupe. Chacun d'entre nous a ainsi pu travailler à son rythme, l'ensemble du logiciel évoluant à chaque modification effectuée. Grâce à ce système de développement, chaque module de notre application peut évoluer indépendamment du reste du logiciel.

### 3.3 Méthodes et outils nécessaires / à disposition

JAVA, pourquoi ?



FIG. 3.4 – Java

De nos jours, Java est le langage le plus utilisé pour le développement d'applications. Il est en constante évolution, ce qui permet de dire qu'un programme développé en Java aujourd'hui et relancé dans 10 ans fonctionnera très bien et même, avec de meilleures performances. Java est un langage complet, qui comprend d'origine un ensemble d'API permettant l'écriture de fichier XML (JAXP), la connexion aux bases de données (JDBC) et bien sur SWING pour créer des interfaces adaptables aux systèmes d'exploitation.

Concernant la rapidité, Sun a incorporé en 1995 la technologie HotSpot, un compilateur JIT (Just-In-Time) dans ses JVM. Il s'agit en fait d'une autre étape de la compilation, mais dont le développeur n'a pas à s'occuper. Vous connaissez certainement la loi des 20/80 de l'économiste italien Vilfredo Pareto qui dit que *80% des richesses sont détenus par 20% de la population*. Et bien ce principe est applicable en programmation, que l'on pourrait formuler de la manière suivante : la JVM repère les morceaux de code très utilisés et les compile en code natif, qui ne passe donc plus par une interprétation et qui devient par conséquent forcément plus rapide.

Cette technologie devient de plus en plus performante à chaque nouvelle version. D'après des tests récents avec la dernière JVM 1.6, Java est aussi rapide voir même plus que C++, et ce sur  $\frac{3}{4}$  des tests effectués.

## Gestionnaire de projet, et gestionnaire de version



FIG. 3.5 – Subversion

Pour mettre en commun et partager le travail de chaque membre du groupe, nous avons utilisé le logiciel Subversion. Subversion est un logiciel de gestion de version, qui centralise tout le travail effectué sur un serveur web. Il permet de garder un historique des différentes versions des fichiers du projet. Il offre la possibilité de revenir rapidement à un moment particulier du développement, ou de comparer et voir les différences entre deux versions.

Subversion permet au groupe d'accéder et de travailler en même temps sur différents fichiers du logiciel. Cela facilite le travail car le code source du logiciel n'a pas besoin d'être envoyé par e-mail. Le code source est accessible via Internet et protégé avec un login et un mot de passe.

Subversion intègre au mieux les différentes modifications, même si elles portent sur le même fichier. Si c'est exactement la même partie du même fichier qui a été modifiée, le serveur prévient alors l'utilisateur qu'il y a un conflit, et lui propose de modifier sa version locale, ou celle en ligne.

## Debian



FIG. 3.6 – Debian

Le serveur web utilisé pour ce projet a été mis en place sur un ordinateur utilisant le système d'exploitation Debian. Le serveur utilise une connexion ADSL haut-débit, disponible à l'adresse <http://visiojava.sicot.fr/>. L'accès au site nécessite un login et un mot de passe, identique à celui nécessaire à Subversion. Debian est une distribution GNU/Linux. Elle fournit un système d'exploitation composé uniquement de logiciels libres.

La version serveur de Debian est utilisée. Elle ne dispose pas d'interface graphique (l'écran du serveur n'est pas utilisé), le code source du projet est donc téléchargé à distance via le logiciel Subversion installé sur le serveur. Debian a été choisie pour sa stabilité et sa sécurité contre les virus ou les attaques extérieures. Grâce à cela le serveur est allumé 24h/24 et permet aux membres du groupe de travailler aux horaires qu'ils souhaitent. En plus du logiciel Subversion, le logiciel Trac est également installé sur le serveur Debian.

## Trac



FIG. 3.7 – Trac

Trac est un système de gestion de projet logiciel et de suivi de défauts / bogues via le web. Il propose une interface pour le système de contrôle de source Subversion, un Wiki intégré, et un nombre intéressant d'options permettant de rester au courant des événements et changements du projet.



Trac inclus un Wiki. C'est un système de gestion de contenu de pages Web modifiables par tous les visiteurs autorisés. Cela facilite l'écriture collaborative de documents avec un minimum de contrainte. Toutes les pages Wiki sont éditables via le navigateur web si vous êtes authentifié et possédez les droits nécessaires. Grâce au Wiki, nous avons mis en ligne des pages expliquant aux membres du groupe comment utiliser les outils nécessaires au développement du logiciel.

Trac nous permet de créer des rapports de bogues. Ces rapports sont appelés tickets. Ils sont attachés à une version du logiciel en cours de développement, et sont validés lorsqu'ils ont été traités. Un historique permet de visualiser et d'accéder rapidement aux dernières modifications du projet, classées par date. Le logiciel Trac propose également un explorateur Subversion, qui permet de naviguer dans le code source du logiciel à partir d'un navigateur web.

### Logiciels utilisés pour le développement



FIG. 3.8 – Eclipse

Nous avons utilisé les deux IDE les plus utilisés dans le développement d'application Java : Eclipse 3.3, et Netbeans 6.0 qui sont des logiciels libres. Ils permettent une grande productivité dans le développement grâce à des fonctionnalités telles que la complétion automatique, la génération des class path, l'intégration de la gestion de fichier Subversion, etc.



FIG. 3.9 – Netbeans

## 3.4 État de l'art

### 3.4.1 Analyse et description des différents logiciels de visioconférence

Dans une première phase, nous avons procédé à une étude de l'existant en matière de visioconférence en analysant des logiciels gérant cette fonctionnalité. Pour chacun, nous avons étudié leurs caractéristiques techniques, leurs points forts et leurs points faibles.

#### Skype

Skype est un logiciel permettant de passer des appels partout dans le monde depuis un ordinateur. Les appels entre utilisateurs Skype sont gratuits sans obligation d'achat ni d'abonnement. Il permet aussi des appels vidéo gratuits et des appels dans le monde entier vers des téléphones ordinaires à petit prix.

- Le protocole utilisé est unique et propriétaire. Les systèmes d'exploitation supportés sont Windows, Linux, Mac Os X et Pocket PC.
- Au niveau de l'ergonomie, l'interface est conviviale (user-friendly) et simple d'accès. Au niveau de la sécurité, tout le trafic de données de Skype est crypté. L'utilisateur n'a pas accès aux paramètres de chiffrement et, par conséquent, n'a pas à prendre des décisions techniques concernant le cryptage.
- Il ne permet pas de faire de la visioconférence entre plusieurs personnes.

#### Windows Live Messenger

Anciennement connu sous le nom de MSN Messenger (MSN), et aujourd'hui appelé WLM (Windows Live Messenger), il est un logiciel client lié au service de messagerie instantanée propriétaire (utilisable gratuitement) pour Windows XP, Windows Vista et Windows Mobile et produit par Microsoft. Il offre les services de VoIP et de visioconférence à compter de sa version 8.0.

- Windows Live Messenger utilise le protocole MSNP (Mobile Status Notification Protocol). Les systèmes d'exploitation supportés sont Microsoft Windows XP/Vista.
- En terme d'ergonomie, l'interface est conviviale et facile d'accès. Il est possible de partager des fichiers (texte, image). De plus, la liste des contacts dispose d'une grande capacité (jusqu'à 600 contacts). Enfin, le transfert de fichier via le chat est supporté, et ce logiciel est gratuit.
- Il ne permet pas de faire de la visioconférence entre plusieurs personnes. Il fonctionne seulement sur les systèmes Windows et il n'est pas compatible avec d'autres logiciels de messagerie instantanée. Le protocole de MSN est fermé et non documenté.

## **VLVC**

VLVC est un logiciel de visioconférence développé dans le cadre d'un projet de fin d'études par des étudiants d'Épitech. Ce logiciel est distribué sous la forme d'un plugin qui s'intègre au célèbre lecteur multimédia VLC.

- VLVC étant un module du client de VideoLAN, il est disponible sur plusieurs plateformes, telles que Linux, Windows, ou MacOS. Le langage C est imposé par VLC. Il est donc utilisé pour développer le cœur du module VLVC tandis que le langage C++ est utilisé pour l'interface graphique qui utilise les wxWidgets et Qt4.
- Logiciel multiplateforme, Open Source et bien documenté. Il permet de faire de la visioconférence avec plusieurs personnes.
- VLVC est distribué sous forme d'un plugin ce qui est un point faible dans notre cas puisqu'il impose des fonctionnalités dont on n'a pas besoin.

## **EKIGA**

EKIGA (préalablement nommé GnomeMeeting) est un logiciel libre de téléphonie et de visioconférence par Internet (voix sur IP ), pour GNU/Linux dont l'interface a été développée avec les bibliothèques de l'environnement GNOME (il fonctionne aussi sur les autres environnements en téléchargeant les librairies nécessaires).

- EKIGA fonctionne sous GNU/Linux et utilise le protocole H.323 et SIP. Il est disponible dans plusieurs langues. Au niveau des types de conférences, il propose des conférences audio, texte, et vidéo. Du fait que EKIGA utilise le protocole H.323, il est compatible avec Netmeeting mais uniquement pour l'audio et la vidéo.
- EKIGA est compatible avec les logiciels utilisant le protocole H.323. C'est un logiciel gratuit et multilingue.
- Ce logiciel ne fonctionne que sur Linux.

## OpenWengo

OpenWengo est un logiciel libre de communication audio (Softphone), visio (Webcam), et chat texte. Les interfaces de programmation libres phApi, oRTP, oSIP et eXosip sont utilisées. OpenWengo peut aussi servir de client de messagerie instantanée, grâce à la bibliothèque de Pidgin (libpurple), qui permet l'utilisation du protocole standard ouvert Jabber, mais aussi l'accès possible aux protocoles propriétaires comme Yahoo! Messenger, AIM/ICQ.

L'analyse comparative des différents logiciels de visioconférence nous a permis d'avoir une vue globale sur les techniques et les protocoles utilisés. Afin de répondre aux attentes du cahier des charges, il nous faut pouvoir gérer un système de présences, de listes de contacts et autres fonctionnalités pourvues dans les logiciels de messagerie instantanée.

Nous avons d'office écarté les protocoles propriétaires pour nous tourner vers les libres. Dans cette optique, nous avons vu au cours de notre analyse un protocole libre récurrent utilisé dans des applications telles GTalk ou OpenWengo. Avec ses millions d'utilisateurs et des avantages que nous allons vous exposer, le protocole Jabber est celui que nous avons retenu.

## 3.4.2 Système de présence et de messagerie

### - Analyse et description du protocole Jabber

Dans un premier temps nous allons décrire les caractéristiques principales de Jabber qui nous ont amené à choisir ce protocole pour l'implémentation de notre logiciel. Ensuite, nous établirons un descriptif des différentes commandes indispensables pour faire fonctionner le protocole, sans toutefois entrer dans le détail de l'implémentation.

#### **Caractéristiques et avantages**

Jabber est un système standard et ouvert de messagerie instantanée : il permet aux personnes de communiquer en temps réel par Internet et de voir quand leurs contacts sont connectés. Jabber n'est pas composé juste d'un logiciel, mais d'une multitude de logiciels (clients) pouvant se connecter au même service (réseau) mais en utilisant des points d'accès différents (serveurs). Voici les avantages qui nous ont incités à nous tourner vers cette solution.

#### – Standard ouvert

Tout comme le Web, Jabber est basé sur des standards ouverts, c'est à dire que son mode de fonctionnement (aussi appelé protocole) est décrit en détail et est accessible gratuitement. Cela a permis la création de nombreux logiciels pouvant utiliser Jabber. Ces logiciels appelés clients existent sur toutes les plateformes (Windows, Linux, Mac OS X, dans les navigateurs web, dans les téléphones mobiles et assistants personnels, etc.) et peuvent tous discuter entre eux. Ces standards ne sont pas dépendants d'une entreprise mais sont gérés par une organisation à but non lucratif, la XSF.

#### – Décentralisé

Jabber est un système décentralisé : comme pour les emails, des serveurs sont situés partout dans le monde. Vous pouvez communiquer avec des personnes sur d'autres serveurs, mais vous ne serez pas affecté si un autre serveur est indisponible. Un autre avantage de Jabber est que toutes les entreprises, fournisseurs d'accès internet (FAI) ou même particuliers peuvent installer leur propre serveur Jabber. Cet aspect est primordial pour le bon fonctionnement de notre application et l'installation d'un serveur Jabber chez l'entreprise ER était la solution adéquate.

- Facilité d'utilisation

La majeure partie de la complexité du réseau Jabber est située sur les serveurs. Cela permet aux logiciels clients de rester très simple : un client Jabber ne consommera pas toutes vos ressources et sera facile à configurer et utiliser, tout en disposant de la puissance de Jabber.

- Sécurité

Jabber est très sécurisé : toutes les communications peuvent être chiffrées. Il permet aussi un chiffrement asymétrique de chaque message par clé PGP ou GnuPG. C'est le seul système de messagerie instantanée qui permette tant de sécurité.

## Protocole

Le protocole est le langage utilisé par les différents logiciels (clients, serveurs, etc.) du réseau pour communiquer entre eux. La dénomination de « protocole Jabber » est cependant maladroite, puisque ce n'est pas un seul mais un ensemble de plusieurs protocoles que nous allons vous présenter.

- XMPP : le noyau de Jabber

C'est le nom de la norme qui définit toutes les fonctionnalités basiques (connexion, échange de messages, d'informations de présence, etc.). La norme est publique, elle est standardisée par l'IETF, organisme de normalisation des principaux standards de l'internet : XMPP-Core définit le cœur du protocole, principalement la communication entre les clients et le serveur, et entre les serveurs. XMPP-IM définit la manière dont sont échangés les présences et les messages par XMPP. Enfin, XMPP-CPIM définit une correspondance entre XMPP et CPIM.

- Les XEP : les extensions du protocoles XMPP

Alors qu'XMPP ne définit que les fonctionnalités essentielles de Jabber, la Jabber Software Foundation (JSF) propose des extensions à ce protocole pour permettre aux utilisateurs de bénéficier de fonctionnalités additionnelles (transfert de fichier (utilisé dans notre application), discussion à plusieurs, voix, etc.). Ces extensions s'appellent « XMPP Extension Protocols » (XEPs).

## - Etude comparative des bibliothèques

Après le choix de Java comme langage de programmation et Jabber comme protocole nous nous sommes intéressés aux API disponibles. Un large choix s'offrait à nous ce qui dans un sens nous donnait plus de liberté mais rendait plus difficile notre étude de l'état d'art et notre décision.

Voici un récapitulatif des différentes librairies que nous avons testées.

### **JSO : Jabber Stream Object**

JSO est un acronyme pour « Jabber Stream Objects », bibliothèque pour Java. JSO donne accès à un support bas niveau pour les commandes du protocole Jabber/XMPP, ainsi qu'une interface de connexion de flux totalement contrôlable. Son but est de fournir une plateforme très flexible et personnalisable pour construire des applications Jabber sur tous les niveaux (serveurs, clients, composants). JSO propose une interface de flux, avec les implémentations pour « jabber :client », « jabber :component :accept », « jabber :component :connect », et « jabber :server » qui sont disponibles.

API la plus complète au premier abord, qui donne la possibilité de créer des serveurs et des clients mais très mal documentée. Nous nous sommes rendus compte en la testant qu'elle était de trop bas niveau et nous aurait ralenti par le manque de documentation et sa difficulté d'assimilation.

### **JXA : Jabber XMPP API**

JXA est une Jabber XMPP API pour J2ME. Ce projet inclut un API complète (contenant un XML Reader et Writer) et un exemple d'implémentation pour un client Jabber pour téléphone mobile. JXA était une API intéressante par son souci d'apporter des supports pour le passage XML mais elle se tournait plus vers le monde des téléphones mobiles et cet aspect n'était pas primordial pour notre application.

### **Tweeze**

Tweeze est une bibliothèque implémentant XMPP-CORE (RFC 3920) et est écrite en Java. Il est important de noter que Tweeze implémente seulement le côté client du protocole. Cette librairie semble peu utilisée et n'implémente pas XMPP-IM (RFC 3921), version plus récente du protocole. Nous nous sommes donc détournés de ce choix.

## **JabberWookie**

La bibliothèque JabberWookie pour Jabber a pour but d'être une implémentation Java complète, extensible, facile d'utilisation pour le protocole XMPP. Elle fournit des fonctionnalités pour les deux côtés clients et serveurs. Le manque de mise à jour et de feedback utilisateur sur cette API ne nous a pas incité à l'utiliser.

## **Smack**

Smack est une bibliothèque Open Source pour le protocole XMPP (Jabber) qui propose une API côté client pour la messagerie instantanée et le système de présence. La bibliothèque est en pur Java et peut être incluse dans tout projet comme un XMPP client très complet à un simple client pour recevoir et envoyer des messages et notifier les présences.

Une javadoc et la documentation proposée par l'API permettent d'obtenir des résultats plus que satisfaisants en peu de temps. Un forum actif et les différentes classes de l'API nous permettent de répondre aux attentes demandées par notre application.

Nous nous sommes donc tout naturellement tourné vers Smack après les tests décrits ci-dessus.



### 3.4.3 Visioconférence

L'étude de l'existant pour cette partie s'établit en deux phases : l'étude du transfert de données et la compression vidéo et l'étude des API disponibles en Java pour l'acquisition vidéo.

#### Transfert de données

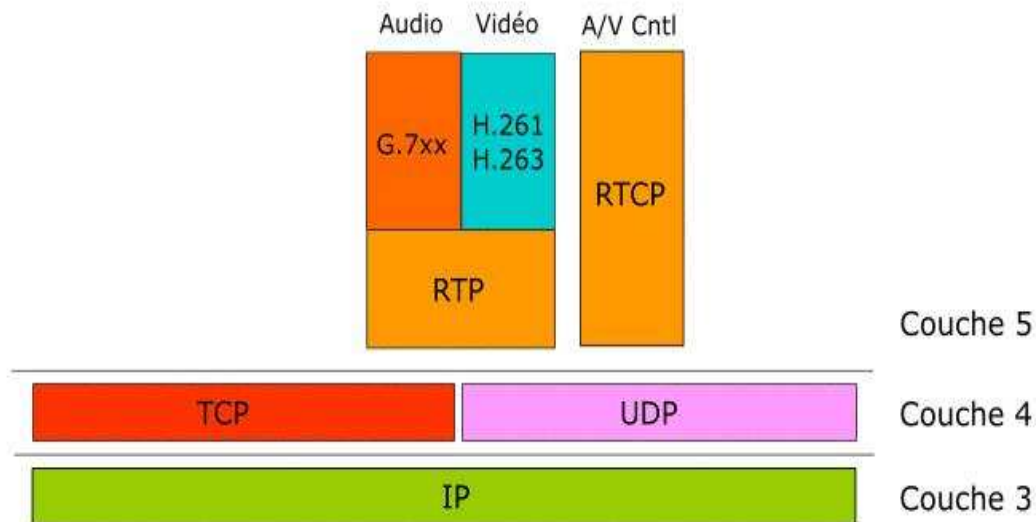


FIG. 3.10 – Une partie du modèle OSI

Deux protocoles sont utilisés pour le transfert vidéo : RTP / RTCP (RTCP étant la version protégée de RTP) et SIP.

RTP (Real Time Transport Protocol) décrit le format de paquet uniforme de la transmission audio et vidéo sur Internet. Il se place dans le modèle OSI au dessus de la couche transport (Figure 3.3) et utilise UDP pour la partie transport. Il est défini dans la RFC 1889 et développé par le Audio Vidéo Transport Group. Il a été publié pour la première fois en 1996.

Session Initiation Protocol (SIP) est un protocole standard ouvert de télécommunications multimédia (son, image, etc.). Il est un protocole normalisé et standardisé par l'IETF et décrit par le RFC 3261. Le SIP n'est pas seulement destiné à la VoIP mais aussi à de nombreuses autres applications telles que la visiophonie, la messagerie instantanée et les jeux vidéo. Il se charge de l'authentification et de la localisation des multiples participants. SIP ne transporte pas les données échangées durant la session comme la voix ou la vidéo, c'est le protocole RTP (Real-time Transport Protocol) qui assure le plus souvent les sessions audio et vidéo.

## Compression Vidéo

Nous nous sommes tout de suite tournés vers la seule API Java permettant l'acquisition et le transfert vidéo : JMF (Java Media Framework). Cette API est utilisée par de nombreuses applications de messagerie instantanée comme Sip Communicator pour assurer la partie visioconférence. Elle bénéficie d'une bonne documentation, d'exemples de code et d'une communauté très présente sur les forums pour répondre à toute question.

Concernant les exemples, Sun met à notre disposition une application libre (JMStudio) qui expose toutes les possibilités de cette librairie. Elle met en place l'acquisition d'un flux vidéo ou audio d'un périphérique de type webcam ou micro et le transfert via RTP.

Les premiers tests que nous avons réalisés se sont donc centrés sur cette application afin de comprendre son fonctionnement. Nous avons d'abord essayé d'acquérir la vidéo et de l'envoyer à une autre personne par l'intermédiaire d'Internet. L'ordinateur A transmet le flux vidéo à l'ordinateur B via la connexion Internet suivant l'architecture définie sur le schéma explicatif ci-dessous. L'ordinateur B obtient ainsi une vidéo en direct.

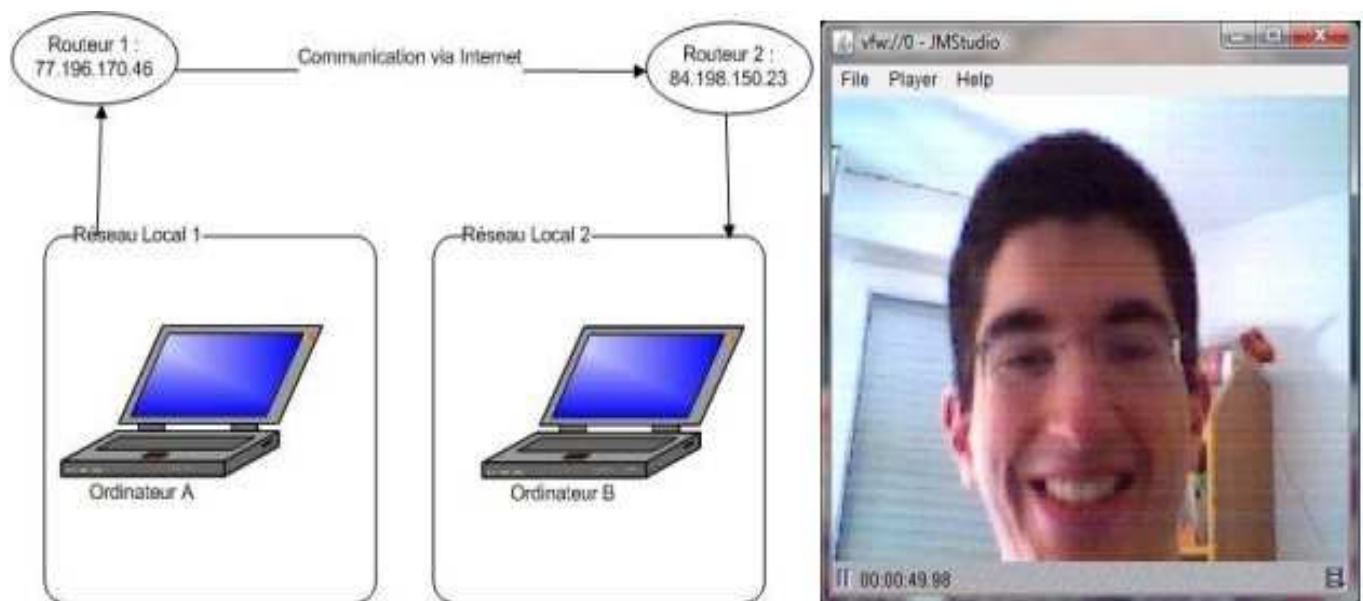


FIG. 3.11 – Acquisition de la vidéo

Le second test que nous avons réalisé était similaire au précédent mais nous avons cette fois utilisé une connexion VPN. Nous avons donc appris comment installer un serveur VPN sous Windows. Le test a été concluant : JMStudio fonctionne très bien sur ce type de connexion sécurisée utilisée par la société Expertise Radiologie, à l'origine du projet.

### 3.4.4 Le tableau blanc

Les outils de dessin style « tableau blanc » sont nombreux aujourd'hui. Parmi les plus connus, Paint (Figure 3.12) et Photoshop exclusivement pour Windows et Gimp, certainement le meilleur outil de dessin de la communauté Open Source.

Pour notre application, nous avons repris les outils de bases qui nous semblaient essentiels, qui sont le dessin à main levée, le dessin de formes simples, rectangle et cercle, et le choix de la couleur.

En effet, les outils tels que brosse, remplissage, aérographe, etc... ne nous ont pas semblé nécessaires, le tableau blanc devant rester un outil simple d'utilisation. A noter que Windows Live Messenger intègre un outil de tableau blanc quasi-similaire à Paint.

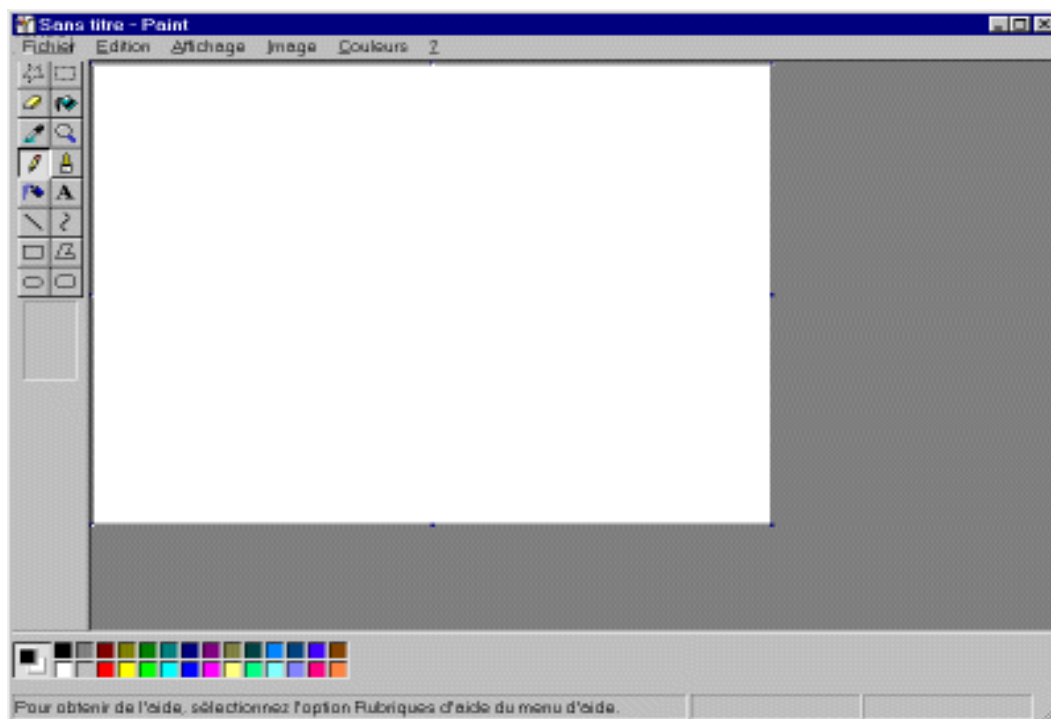


FIG. 3.12 – Paint

## 3.5 Réalisation

### 3.5.1 Visioconférence

#### Intégrer JMStudio dans notre application

Notre premier objectif a été l'intégration de JMStudio logiciel dans notre application. En effet, elle comprend toutes les fonctionnalités demandées et mentionnées dans le cahier des charges et son utilisation est concluante dans l'ensemble des tests. Nous nous sommes donc consacrés à l'étude du code de l'application pour comprendre son fonctionnement ainsi que celui de la librairie. Pour cela nous avons étudié avec précision la documentation présente sur le site Internet et les tutoriaux écrits par des développeurs ayant utilisés la librairie dans de précédents projets. Nous nous sommes penchés sur les points essentiels suivants : comment récupérer les périphériques audio et vidéo, comment acquérir un flux vidéo ou audio et comment envoyer le flux vidéo compressé via RTP. Une fois le fonctionnement du logiciel compris, nous l'avons intégré et obtenu l'application suivante :

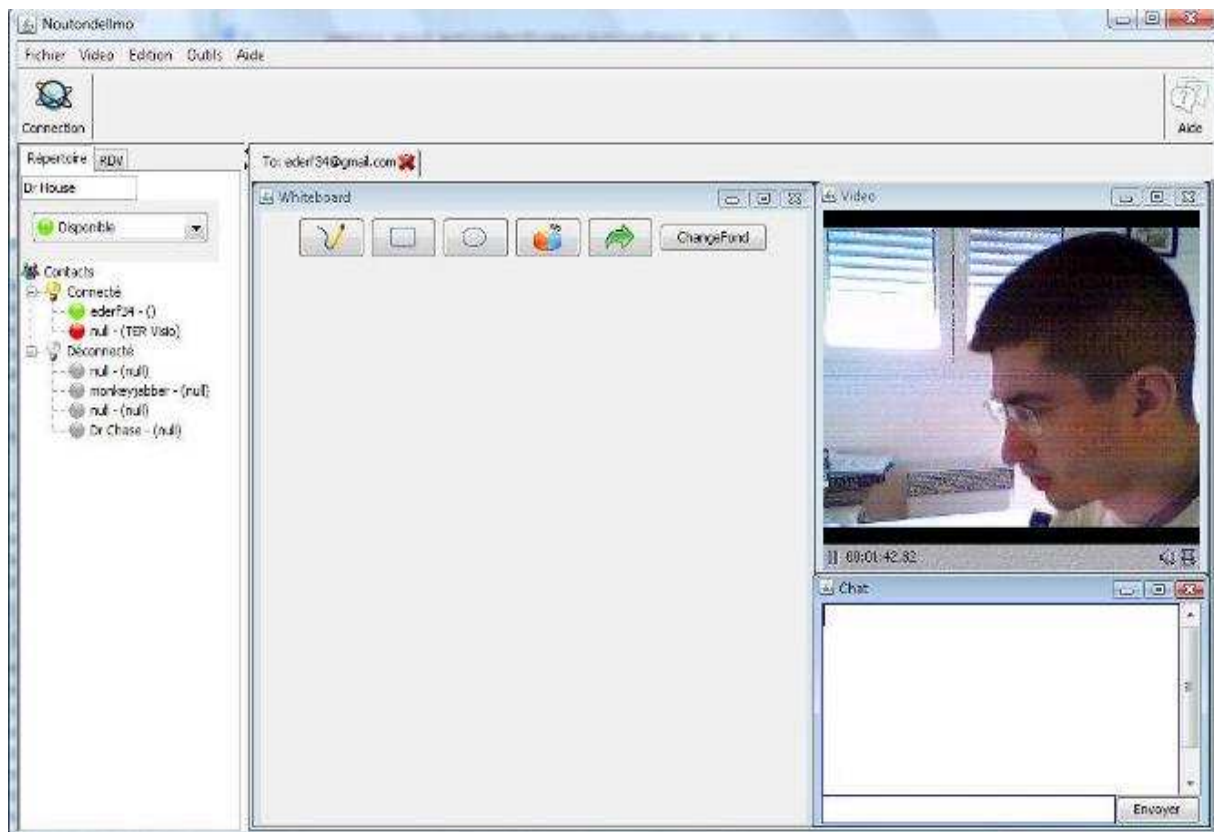


FIG. 3.13 – L'application après intégration de la vidéo

Après avoir discuté avec nos tuteurs, nous avons pensé qu'il serait préférable de superposer la vidéo de l'utilisateur et de son interlocuteur pour lui assurer un feedback.

Nous avons donc essayé de réaliser cette fonctionnalité avec les modules présents dans JMStudio. En effet, JMStudio utilise un module MediaPlayer responsable de l’affichage d’un flux vidéo. Ainsi nous avons tenté d’intégrer deux MediaPlayer séparément mais dans une même fenêtre. Ceci s’est avéré impossible : après plusieurs tests, nous n’avons jamais réussi à avoir deux lecteurs fonctionnant simultanément. Après une recherche plus fine dans la documentation existante et en consultant différents forums, nous avons compris que le problème provenait de la non synchronisation des deux lecteurs (chaque lecteur s’exécutant dans des tâches distinctes, ils rafraîchissent le panel à des instants différents, ce qui crée un problème de visualisation).

## Développement de notre propre lecteur

Pour remédier au problème précédent, nous avons décidé de nous lancer dans la création d’un lecteur binaire. Ce lecteur binaire est composé d’un panel rafraîchi régulièrement par une tâche. Celle-ci possède deux contrôleurs de flux, un sur la webcam de l’utilisateur et l’autre sur le flux vidéo envoyé via le protocole RTP de son interlocuteur. A chaque intervalle de temps la tâche crée une nouvelle image composée des deux dernières images provenant des lecteurs et les superpose. Voici le code simplifié de la tâche :

```
public class LecteurBinaire extends Thread{

    private LecteurMedia lecteur ;
    private LecteurMedia lecteur1 ;
    private JPanel panelv ;

    public LecteurBinaire (generalChatPane g) {
        panelv = new JPanel ();
        this.gChatPane = g;
    }

    public void run () {
        super.run ();

        // Lancement de la transmission
        transmitMedia ();
        // Lancement de la reception video en RTP
        openRtp ();

        Buffer buf = null;
        Buffer buf1 = null;
        FrameGrabbingControl fgc1 = null;
```

```

FrameGrabbingControl fgc = null;

while (true)
{
    // Recupere un Buffer contenant l'image demandee
    if (fgc!=null)
        buf = fgc.grabFrame();
    else
    {
        // Recupere un controle sur le flux du lecteur 1
        if (lecteur !=null)
            fgc = lecteur.getGrabbingControl();
    }

    if (fgc1!=null)
        buf1 = fgc1.grabFrame();
    else
    {
        // Recupere un controle sur le flux du lecteur 2
        if (lecteur1 !=null)
            fgc1 = lecteur1.getGrabbingControl();
    }

    // Construction de l'image composee de l'image de l'utilisateur
    // et de celle de son interlocuteur
    BufferedImage tmp = constructionImage(buf1, buf);
    panelv.setImg(tmp);
    panelv.repaint();

    try {
        Thread.sleep(55);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
}
}

```

### 3.5.2 Le chat

L'implémentation de notre application étant divisée en parties distinctes au départ, chacune des fonctionnalités a été développée avec plus ou moins de facilités. La principale difficulté consistait à réunir ces différentes fonctionnalités dans le même projet, phase durant laquelle de nombreux problèmes survenaient. Nous pouvons citer par exemple le problème entre le transfert vidéo via le protocole RTP et le système de présence gérant les conversations.

Même si les deux parties étaient fonctionnelles, les réunir engendrait de nouveaux problèmes et des difficultés. En effet, pour démarrer un transfert vidéo via le protocole RTP, il est absolument nécessaire de connaître l'adresse IP de notre interlocuteur. Cependant cette donnée n'est pas fournie dans le protocole Jabber par souci de confidentialité.

On se rend alors compte que l'intégration des modules est une phase difficile pour tout projet à cause des nouvelles demandes et directions de l'application auxquelles les développeurs n'ont pas pensé auparavant. Nous avons retenu deux solutions répondant à ce problème, une côté client (celle implémentée pour l'instant) et une autre côté serveur (version que nous souhaitons mettre en place plus tard).

Côté client : le client envoie son adresse IP par le biais d'un paquet XML (en utilisant le protocole Jabber) tel un message ou un changement de présence. Pour récupérer notre adresse IP nous passons par une requête sur un site internet qui nous renvoie la bonne adresse.

Côté serveur : une solution plus élégante consiste à modifier les propriétés de notre présence en lui rajoutant l'adresse IP. En effet cette information est disponible sur le serveur Jabber mais n'est pas communiquée. En changeant la forme de notre paquet XML de présence et en lui ajoutant l'adresse IP du client, chaque client peut connaître l'adresse IP d'un contact de son roster (si celui ci est connecté). Cette solution est implémentable par un plugin de notre serveur Jabber (Openfire) mais demande beaucoup plus de précautions et de tests pour ne pas risquer des problèmes.

### 3.5.3 Le tableau blanc

Notre application intègre un tableau blanc, permettant aux deux interlocuteurs de visualiser la même image - la radiographie du patient - et au radiologue de dessiner sur cette image comme dans un outil de dessin classique, afin de mettre l'accent sur un point précis de cette image (une fracture par exemple). Chaque dessin effectué par le radiologue doit être également dessiné sur l'ordinateur distant (celui du patient) de telle sorte qu'il puisse voir la même chose au même moment ou presque.

L'affichage de l'image se fait dans un JPanel. Nous avons optimisé l'affichage de l'image en utilisant un format toujours compatible avec le périphérique d'affichage, obtenu par le code suivant :

```
GraphicsConfiguration configuration =  
GraphicsEnvironment.getLocalGraphicsEnvironment().getDefaultScreenDevice()  
.getDefaultConfiguration();  
BufferedImage compatibleImage =  
configuration.createCompatibleImage(image.getWidth(), image.getHeight());
```

L'image obtenue bénéficie ainsi d'accélération matérielle importantes. Cela est d'autant plus important que nous avons doté notre tableau blanc de barres de défilement et d'un zoom, permettant une meilleure visibilité. L'image est ainsi redessinée à chaque fois que l'utilisateur zoome ou fait défiler le panneau. Dans la première version du tableau blanc, ces « options » n'étaient pas disponibles et l'utilisation du tableau blanc n'était que très peu pratique. En effet, en cas d'image de grande taille, une partie de l'image débordait du panneau et n'était pas visible par l'utilisateur, qui ne pouvait donc pas dessiner dessus.

Deux sortes de dessin sont à disposition du radiologue : le dessin classique, dit à main levée, et le dessin de formes, rectangles et/ou cercles. Chaque dessin à main levée est stocké dans une liste de points, elle-même stockée dans un vecteur. Les formes sont également stockées dans un vecteur, de shape cette fois-ci. Une shape est constituée de deux points, le point haut gauche et le point bas droit du rectangle englobant la forme (qui est donc égal au rectangle lui-même dans le cas du dessin d'un rectangle).

Une fois ces formes enregistrées, on peut les envoyer au destinataire à l'aide du bouton prévu à cet effet. Un simple clic déclenche un parcours de notre vecteur afin d'envoyer chaque shape rencontrée sous forme de chaîne de caractères, chaque chaîne constituant un message. Ainsi, si trois formes sont dessinées (un rectangle et deux cercles par exemple), trois messages invisibles par l'utilisateur sont envoyés au destinataire et traités par celui-ci afin d'afficher les formes sur son écran.

Il en va de même pour les listes de points. Les points sont concaténés et chaque liste est envoyée sous forme de chaîne de caractères. À la réception, un scanner (disponible depuis Java 5) récupère chaque information et la shape est donc redessinée avec ses attributs (forme et couleur) d'origine. De même pour les dessins à main levée.



Le défaut majeur de cette version est qu'il ne tient compte ni de la taille du tableau blanc, ni du zoom en application au moment du dessin. En effet, un dessin effectué avec un zoom de 2x ne s'affiche pas au même endroit de l'image chez le destinataire si celui-ci est dans une autre configuration.

Il a donc fallu modifier la méthode de récupération des points. Pour cela, nous avons choisi de ne plus récupérer les coordonnées « brutes » des points, mais leur position relatives par rapport à l'image en fonction du zoom utilisé. Ainsi, la forme redessinée chez le patient se retrouve au bon endroit, et ce quel que soit le zoom utilisé par lui et le radiologue.

Un deuxième défaut est rapidement apparu : la lenteur de l'affichage des dessins chez le destinataire. En effet, nous nous sommes aperçu que lorsque la chaîne de caractères était très grande (comme ça peut être le cas pour un dessin à main levée), le temps entre l'envoi et la réception du message était très élevé, dépassant parfois les 30 secondes.

Nous avons pallié à ce problème en modifiant notre méthode d'envoi. Désormais, les points sont envoyés un à un en même temps que le traçage du dessin, et sont également affichés un à un chez le destinataire.

## 4 Résultats

### 4.1 L'interface principale

#### 4.1.1 L'écran d'accueil



FIG. 4.1 – Affichage de l'écran d'accueil

Au démarrage de l'application une fenêtre de dialogue est affichée. Elle permet à l'utilisateur de saisir son login composé de son pseudo et du nom du serveur sur lequel a été créé son compte. Il peut choisir le type de profil avec lequel il veut lancer le logiciel, cette option permettant de réduire les fonctionnalités de l'application selon le profil.

Tous les logins saisis par l'utilisateur sont sauvegardés au format XML dans un dossier différent suivant le système d'exploitation : (Windows Vista : HOME/AppData/Roaming/Nom du logiciel, Linux : HOME/.Nom du logiciel/, Windows XP : HOME/Application Data/Nom du logiciel). La case à cocher située au bas de la fenêtre permet à l'utilisateur de désactiver son affichage à chaque démarrage. Cette option est enregistrée grâce à l'API Preferences dans la base de registre du système si celui-ci le permet. Dans le cas contraire, Java se chargera de simuler une base de registre dans son répertoire d'installation.

## 4.1.2 L'interface

Après le passage de l'écran d'accueil, nous arrivons à la véritable interface de notre application, celle contenant le cœur des fonctionnalités de la visioconférence. L'aspect graphique de celle-ci a été beaucoup modifié tout le long du développement mais cette interface est restée proche du schéma original que nous avons proposé lors du cahier des charges (cf Figure 3.2 page 16). Pour vous présenter au mieux l'interface finale nous pouvons la découper en plusieurs parties, représentées dans le schéma ci-dessous (Figure 4.2) par différentes zones.

Notre principale directive était de rendre notre application facile à utiliser et intuitive pour les docteurs/patients car ces utilisateurs peuvent ne pas être familiers avec l'outil informatique. L'utilisation de couleurs, d'icônes explicites et la réduction des possibilités au maximum doit rendre notre interface utilisable par des novices. Voici les différentes parties de notre interface :

- Barre de menu (en bleu foncé).
- Barre d'outils (en rouge).
- Panneau des contacts et rendez-vous (en violet).
- Onglets des conversation (en vert pâle).
- Panneau de conversation (en orange).

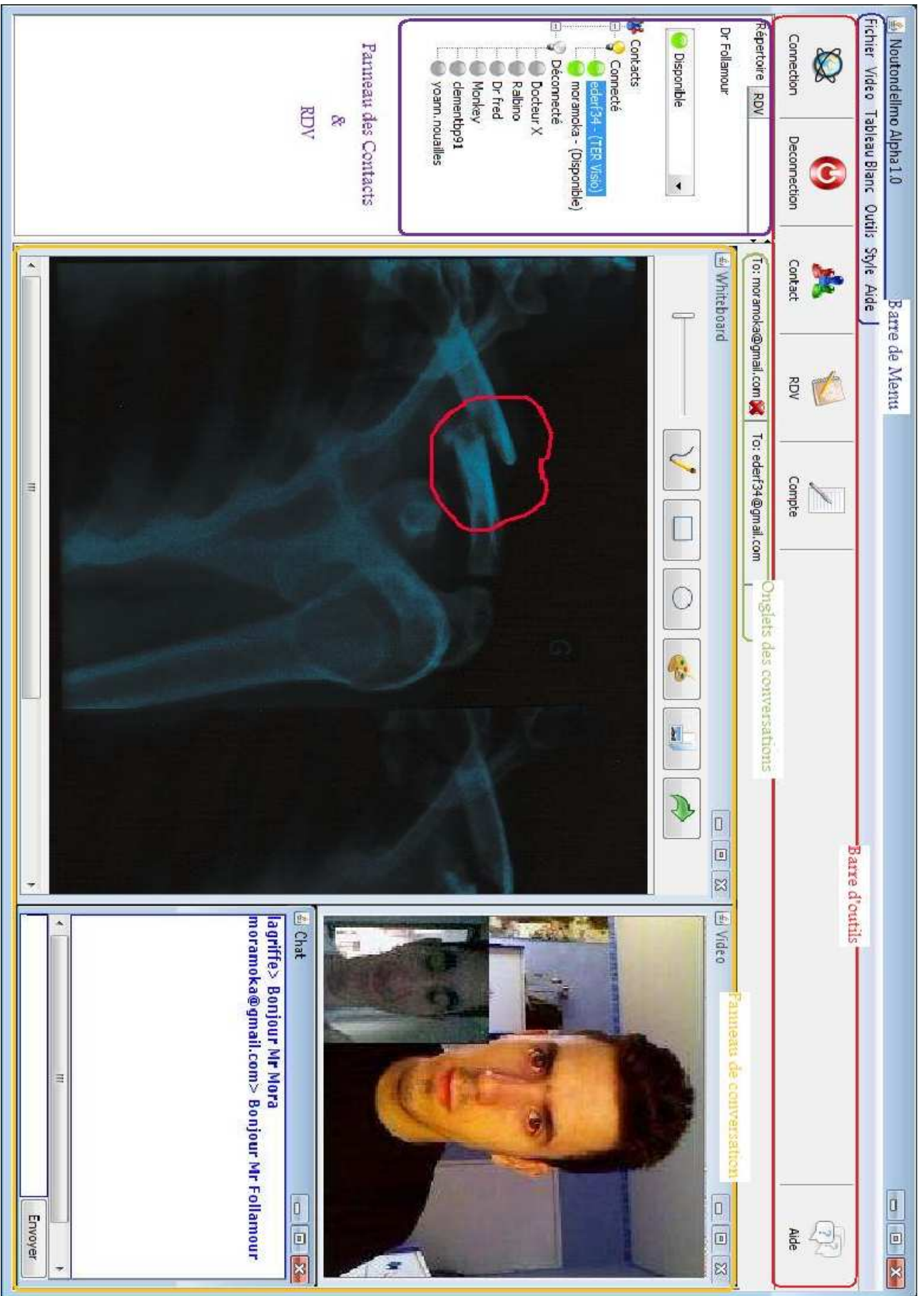


FIG. 4.2 – L'interface principale

Comme dans toute application graphique, une barre de menu est présente. Elle sert à ordonner les différentes fonctionnalités et donne accès à certaines actions non accessibles d'un simple clic. Les catégories du menu sont « Fichier » (comprenant la fonction Quitter par exemple), « Vidéo », « Tableau Blanc » (paramètres de celui-ci), « Outils », « Style » (pour changer le thème graphique de l'application) et « Aide ». La barre d'outils, quant à elle, permet un accès facile aux principales fonctionnalités de l'application, représentées par des boutons aux images explicites.

Sur la gauche de l'interface se trouve le panneau des contacts et de rendez-vous. Sa largeur est modulable selon les besoins, laissant une liberté à l'utilisateur quant à l'agencement des panneaux. Ce panneau permet de gérer les rendez-vous et son statut et de visualiser les autres contacts pour pouvoir notamment engager une conversation (en double cliquant sur le contact souhaité).

Le radiologue donnant ses diagnostics doit avoir la possibilité de gérer plusieurs conversations en même temps. Nous avons rendu ceci possible grâce aux onglets dits « de conversations », qui permettent une navigation aisée entre celles-ci. Chacune d'entre elles est affichée dans le panneau principal. C'est le plus important de tous, celui contenant le cœur des fonctionnalités (chat, visioconférence et tableau blanc). Chacun de ses modules est présenté dans le chapitre suivant.

Nous pouvons aussi signaler un ajout intéressant pour notre application qui est l'intégration de bulles d'informations dans la barre des tâches (Figure 4.3). Ainsi, lorsque l'on reçoit un message et que l'application ou l'onglet de conversation n'est pas au premier plan (n'est pas actif), une bulle contenant le message reçu s'affiche permettant à l'utilisateur d'être tenu informé.



FIG. 4.3 – Les bulles d'aide

## 4.2 Visioconférence

### 4.2.1 Paramétrages

Nous proposons à l'utilisateur plusieurs fenêtres de dialogue lui permettant de paramétrer l'application. Voici le panel responsable de la détection des périphériques audio ou vidéo présents sur l'ordinateur.

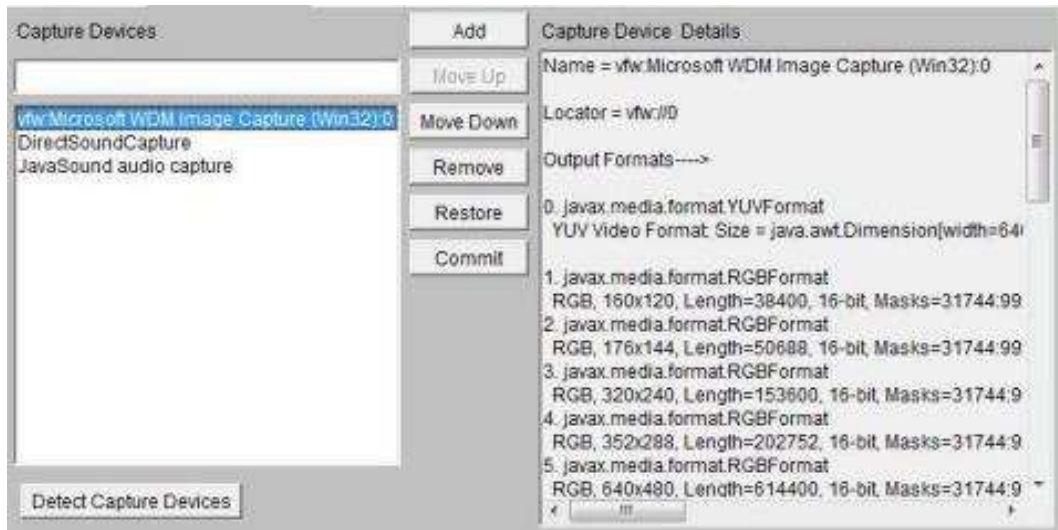


FIG. 4.4 – Détection des périphériques audio et/ou vidéo

Parallèlement on donne la possibilité à un utilisateur expert de paramétrer le flux vidéo envoyé.

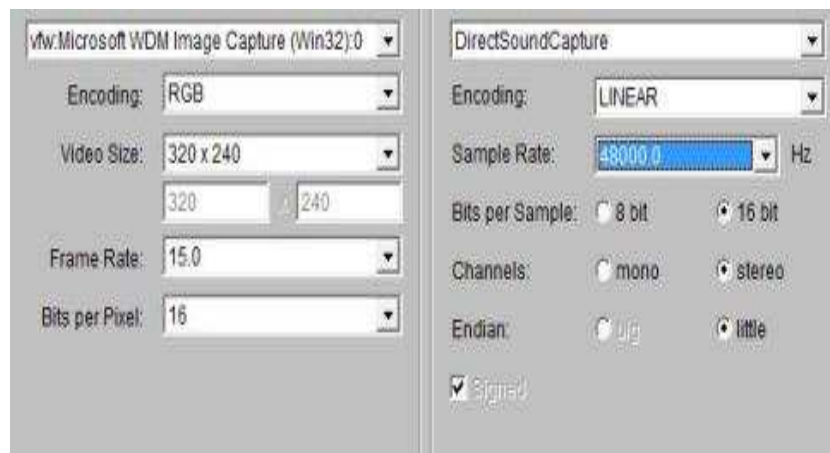


FIG. 4.5 – Paramètres du flux vidéo

## 4.2.2 Affichage de la vidéo



FIG. 4.6 – Affichage de la vidéo

Une fenêtre interne à chaque onglet de conversation permet de visionner la vidéo. Tant que l'utilisateur ne reçoit pas le flux vidéo de son interlocuteur, la capture de sa webcam prend toute la fenêtre. Dès qu'il reçoit le flux de son interlocuteur, l'image de sa webcam se rétrécit et vient se loger dans le coin inférieur gauche de la fenêtre.



## 4.3 Le chat

Une fois le choix du langage et de la librairie fait nous avons pu commencer l'implémentation du chat.

La librairie Smack implémente le protocole Jabber, la mise en place de ce protocole nous permet donc de gérer notre liste de contacts ainsi que les différents statuts. Nous avons choisi de proposer un chat en plus de la vidéo lors des conversations. Tous les tests d'implémentation du protocole Jabber ont été réalisés sur un serveur public Jabber. Ainsi une première version du module de gestion de contacts a été développée. Celle-ci permet d'établir la communication avec un contact en mode console.

```
TestSmack [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (17 avr. 08 12:28:06)
JAJABER V.1
*****
Veuillez entrez votre login:
lagriffie
Mot de passe:
bozozox
Voici votre liste de contacts
null (albino@12jabber.com) Statut: unavailable null
yoann.nouailles (yoann.nouailles@gmail.com) Statut: unavailable null
ederf34@12jabber.com (ederf34@12jabber.com) Statut: unavailable null
ederf34 (ederf34@gmail.com) Statut: unavailable null
null (clementbp91@12jabber.com) Statut: unavailable null
monkeyjabber (monkeyjabber@jabber.org) Statut: unavailable null
griffondor (griffondor@12jabber.com) Statut: unavailable null
moramoka (moramoka@gmail.com) Statut: unavailable null
Taper chat pour discuter, contact pour revoir la liste de contact, statut pour changer son statut et quitter pour se déconnecter.
```

FIG. 4.7 – Console

Les premières bribes de conversations sont donc réalisées en mode console en tapant différentes commandes. On peut engager la conversation, afficher la liste des contacts, se connecter et se déconnecter. Cette première version nous a permis de nous familiariser avec la librairie Smack et de mettre en place les fonctionnalités de base. Cependant le mode console ne nous permettait pas de gérer plusieurs conversations simultanées ni les statuts des contacts.

Une fois ces différentes fonctionnalités implémentées nous avons intégré nos méthodes dans une interface graphique. Les contacts sont désormais affichés dans un arbre et la conversation dans un panel (voir Figure 4.8). Un contact possède deux statuts, en ligne et hors ligne et un double clic permet d'engager la conversation avec un autre contact.



FIG. 4.8 – Conversation dans un panel

A ce stade du développement, l'unique moyen d'identifier un contact est son JID (Jabber ID) qui est de la forme d'une adresse dupont@serverjabber.fr (voir Figure 4.9).

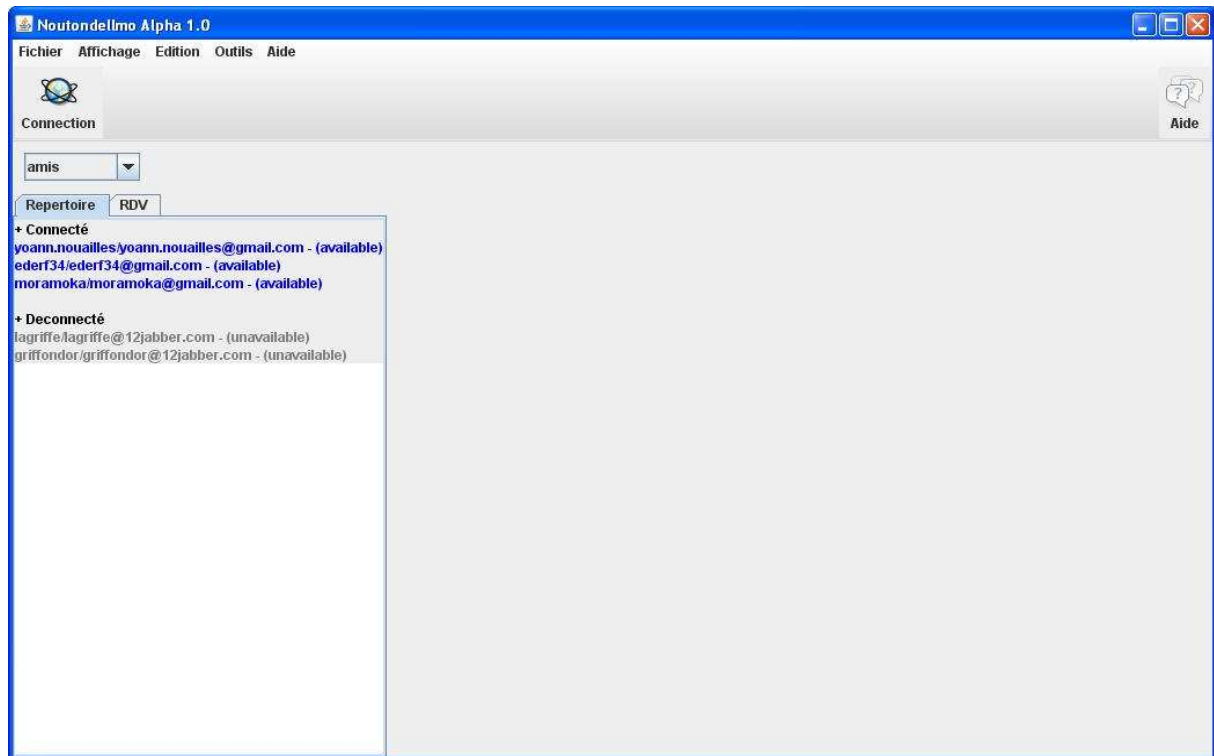


FIG. 4.9 – Contacts et leur JID

Pour pallier à ce manque, nous nous sommes tournés vers l'utilisation d'une fonctionnalité de Jabber, la vCard.

La vCard est un format standard ouvert d'échange de données personnelles (Visit Card soit carte de visite). La version 3.0 de vCard est définie par les RFC 2425 et 2426. C'est un moyen performant pour échanger des éléments de carnet d'adresses. Une vCard contient des éléments relatifs au statut personnel (nom, prénom, âge, pseudo, ...) ou encore professionnel (numéro de téléphone, lieu de travail, ...). Nous avons donc utilisé ce système pour gérer les pseudonymes correspondant à chaque personne. Grâce à ce moyen, on peut identifier les docteurs par le nom qu'ils définissent pour se reconnaître (Ex : Docteur X.).

Exemple d'utilisation (voir Figure 4.10) :

Nous sommes connectés sur l'adresse `docteur@server.fr` et nous n'utilisons pas la vCard pour commencer. Nous voyons que l'espace alloué au pseudonyme situé au dessus du menu déroulant est vide. Remplissons le avec Docteur X et utilisons désormais la vCard.

Nous pouvons observer que du côté du docteur distant, le JID est d'abord utilisé (mais pas la vCard) puis laisse place ensuite à Docteur X (avec l'utilisation de la vCard).

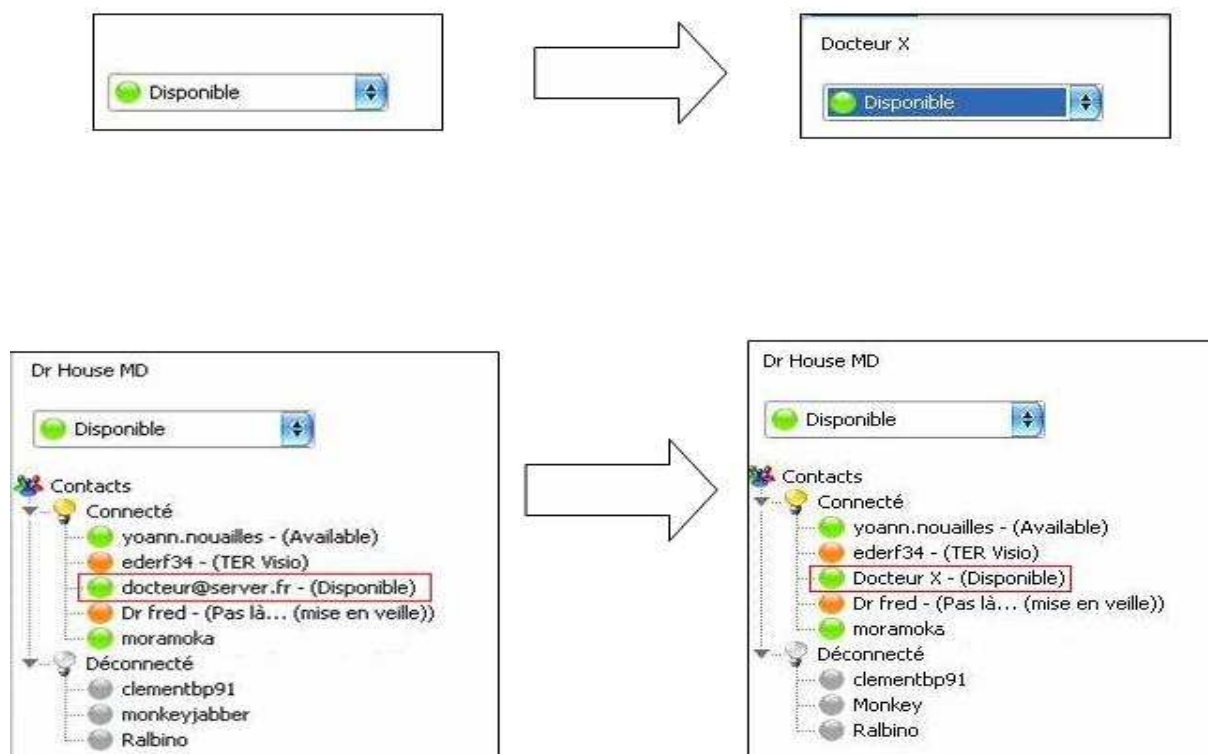


FIG. 4.10 – Utilité de la vCard

Pour permettre une communication plus intuitive et plus aisée nous avons intégré au logiciel une gestion des présences. En effet, un utilisateur peut avoir lancé son application sans pour autant être encore en train de l'utiliser. Un feedback est donc nécessaire pour permettre aux utilisateurs de signaler leur présence ou non. Nous avons choisi quatre statuts pour implémenter cette fonctionnalité. Les noms des statuts, explicites, sont : disponible (available), en consultation (do not disturb), absent (away) et hors ligne (offline). Un code de couleur vient s'ajouter à ce système, permettant ainsi une visualisation rapide de l'état courant.

Reprenons notre Docteur X et faisons le changer successivement d'état. Que se passe-t-il du coté de ses contacts connectés ?

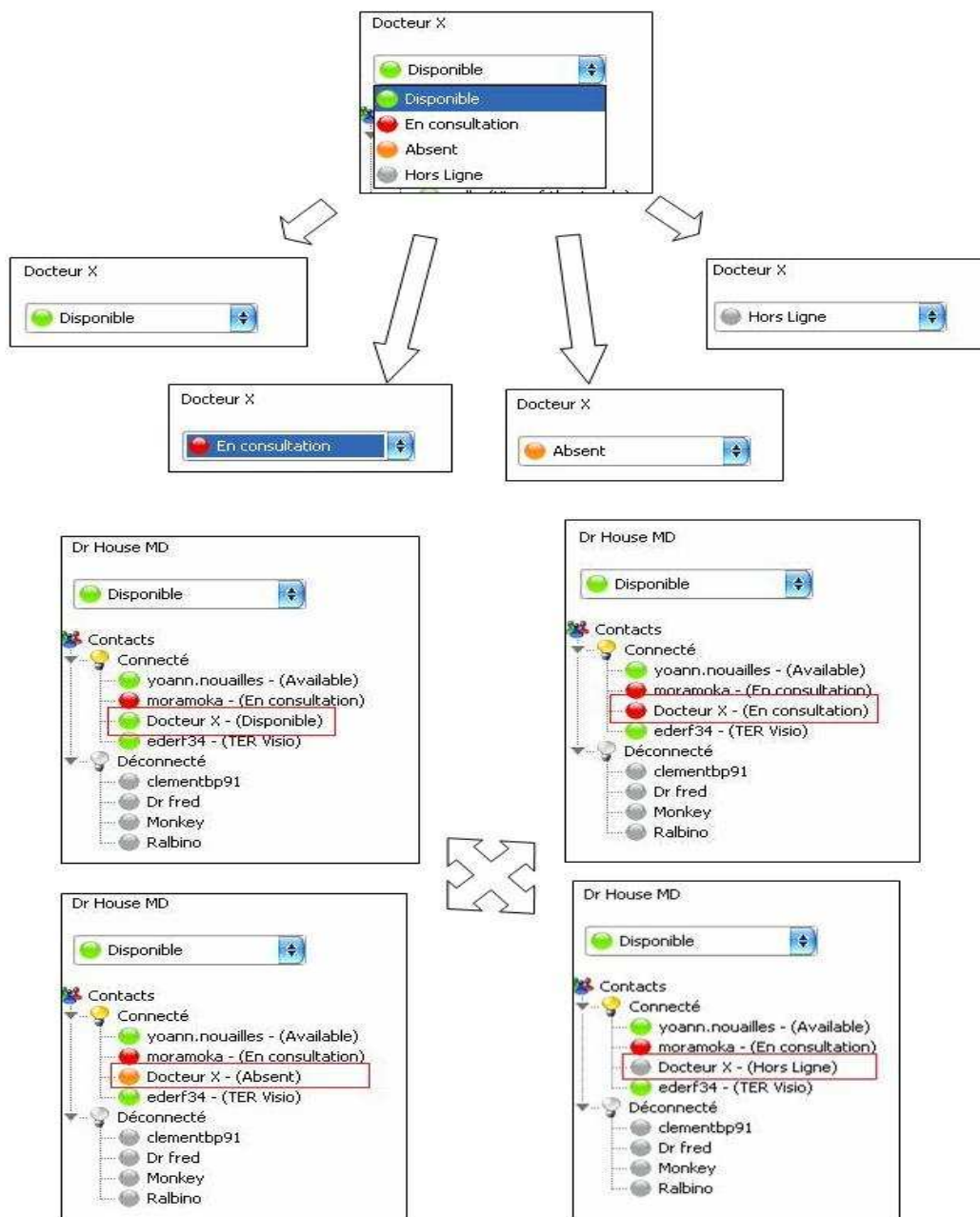


FIG. 4.11 – Les changements de statut chez les contacts

Une fois notre module de gestion des contacts finalisé, nous avons pu ajouter des fonctionnalités à notre application telles que l'ajout de nouveaux contacts ou la création de compte. Ces deux fonctionnalités s'effectuent par le biais de formulaires de saisie et provoquent la mise à jour de l'arbre de contacts.

## **Décomposition Administrateur/Docteur/Patient**

Notre module offre donc diverses fonctionnalités pour tout type d'utilisateur. Par manque d'informations au cours de notre développement, nous avons initialement implémenté nos propres choix. Par la suite, divers entretiens avec Expertise Radiologie nous ont éclairci sur les points sombres du sujet et nous avons pu décomposer les fonctionnalités suivant le type d'utilisateur.

Voici la hiérarchie de nos utilisateurs avec les commandes disponibles :

### 1. Administrateur

- Ajouter un compte jabber.
- Supprimer un compte jabber.
- Paramétrer un compte login (c'est le nom que les contacts verront dans la listes des personnes connectées, par ex : Dr House), mot de passe, nom du serveur.
- Modifier la Vcard.

### 2. Docteur

- Voir les contacts en ligne (uniquement ceux définis ou autorisés par les administrateurs Jabber en fonction des rendez-vous de vacation).
- Commencer une conversation (chat, vidéo, tableau blanc).
- Changer de login/statut.
- Connexion/déconnexion.
- Prise de rendez-vous.
- Consulter l'aide.
- Mettre à jour le logiciel.

### 3. Patient

- Voir son téléradiologue (unique) en ligne auquel a été attribué la vacation. Il ne peut donc voir que lui (défini par les administrateurs).
- Commencer une conversation (chat, vidéo, tableau blanc).
- Connexion/déconnexion.
- Consulter l'aide.

## 4.4 Le tableau blanc

Le tableau blanc de l'application s'inspire largement des outils de dessin habituellement utilisés (Paint, Gimp...). Il est constitué d'une fenêtre de visualisation de l'image (un JPanel) et d'une barre d'outils (un ensemble de JButton intégré dans un JPanel) permettant d'effectuer diverses opérations de dessin sur cette image.



FIG. 4.12 – Le tableau blanc à ses débuts

Dans sa première version, le tableau blanc permet à l'utilisateur de dessiner deux formes de base, des rectangles et des cercles, le dessin à main levée n'étant pas encore implémenté. Il suffit pour cela de choisir la forme désirée en cliquant sur le bouton approprié. Ensuite, un simple clic de souris sur l'image suivi d'un « drag » (appui continu sur le bouton gauche de la souris accompagné d'un déplacement) affiche la forme sélectionnée sur l'image.

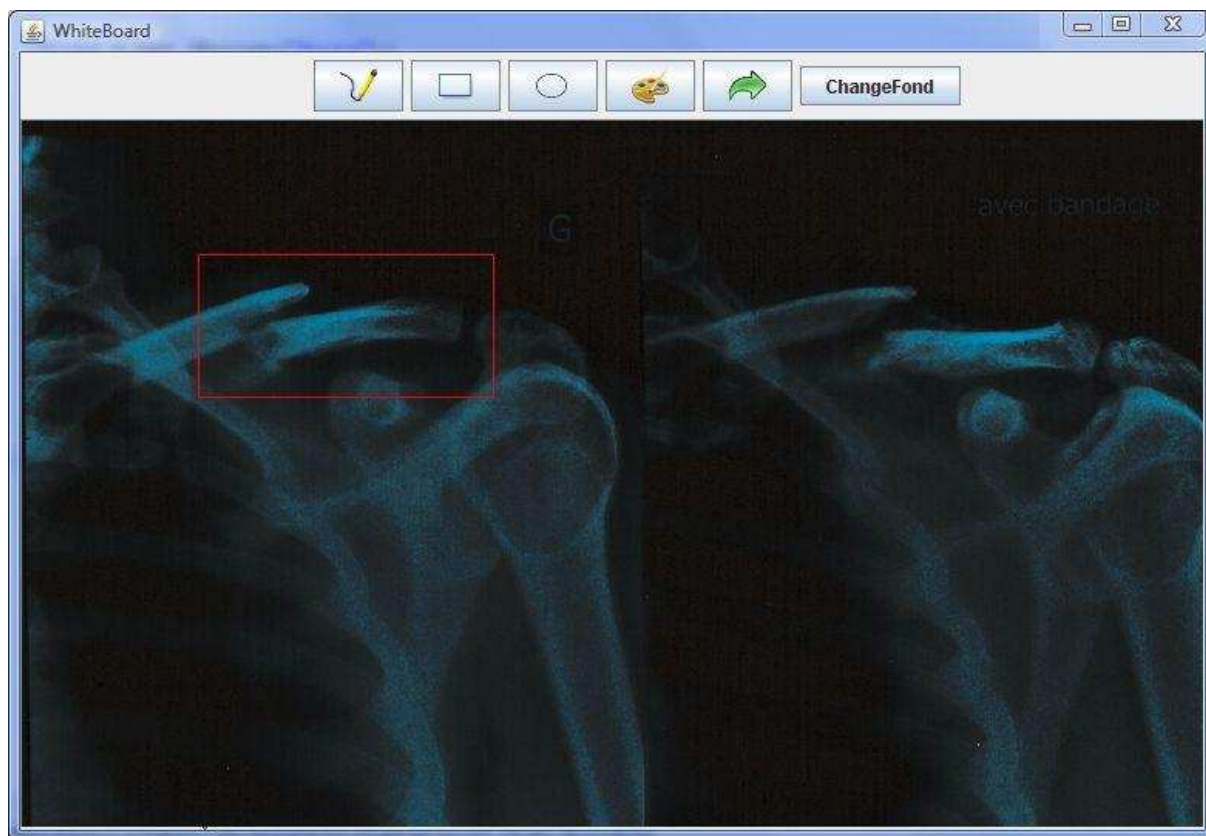


FIG. 4.13 – Le dessin de formes dans le tableau blanc

Cette version n'est véritablement pas très avancée mais permet déjà d'afficher les modifications effectuées sur l'écran de l'ordinateur distant en cliquant sur le bouton d'envoi. Elle a plusieurs défauts. Elle ne permet pas l'affichage de photos de grande taille, puisque le JPanel est « figé ». Elle ne permet pas non plus le simple dessin à main levée et l'affichage des shapes sur l'ordinateur distant ne se fait pas forcément au bon endroit, l'utilisateur distant ayant pu retaillé sa fenêtre entre temps (les coordonnées envoyés ne correspondent alors plus).

Le tableau blanc intègre donc dans sa version suivante des barres de défilement horizontale et verticale, un zoom, et une gestion des coordonnées différente de la précédente, par position relative des points. Les barres de défilement permettent désormais l'affichage d'images de grande taille.





FIG. 4.14 – Un tableau blanc plus évolué

Cette version permet également le dessin à main levée. Désormais, l'utilisateur n'est plus contraint à utiliser les formes mises à sa disposition et peut dessiner à sa guise. Pour « envoyer » ses dessins et permettre au patient de les voir, il lui suffit alors de cliquer sur le bouton de validation, représenté en haut à droite du tableau blanc par une flèche verte.

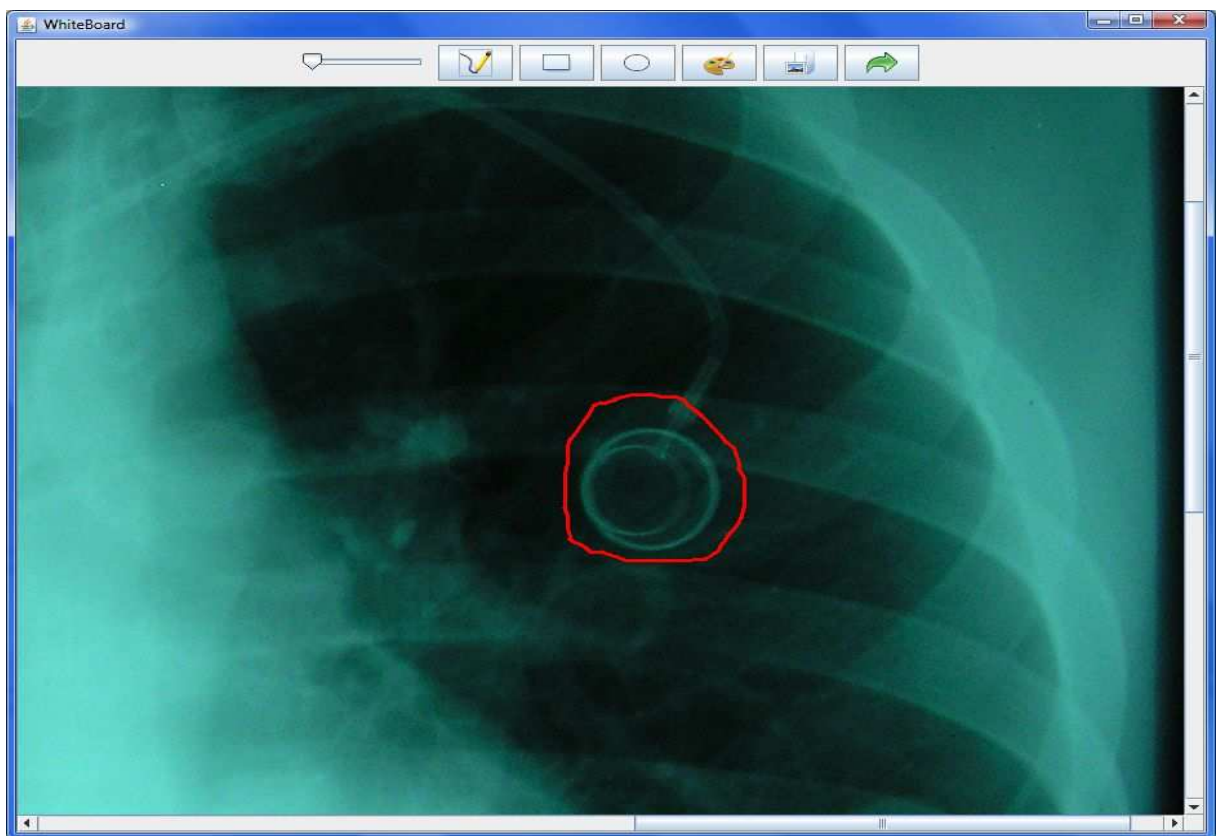


FIG. 4.15 – Le tableau blanc et le dessin à main levée

## 4.5 Fonctionnalités supplémentaires

### 4.5.1 Rendez-vous

L'application de visioconférence doit répondre à un certains nombres de contraintes imposées par son contexte d'utilisation. En effet cette application doit proposer un service de télé-diagnostic et donc évoluer dans un cadre spécifique qu'est le milieu médical. Cependant, devant le manque de précisions de l'entreprise, nos tuteurs nous ont laissé un large choix concernant l'implémentation des fonctionnalités du logiciel. C'est pourquoi la possibilité de gérer des rendez-vous a été ajouté à l'application, donnant ainsi à l'utilisateur la possibilité de planifier des rendez-vous vidéos avec ses contacts. L'organisation des rendez-vous se fait par le biais d'un formulaire de création dans lequel on doit choisir un contact, une date et l'objet du rendez-vous.

Ajouter un RDV

Contact: ederf34@gmail.com

Date: 17 avril 2008

Heure:

avril 2008						
lun.	mar.	mer.	jeu.	ven.	sa...	dim.
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

Creer

FIG. 4.16 – Le formulaire de prise de rendez-vous

Les rendez-vous une fois créés sont accessibles par un onglet et rangés par ordre chronologiques. En choisissant une date, l'utilisateur accède aux différents rendez-vous prévus dans la journée.

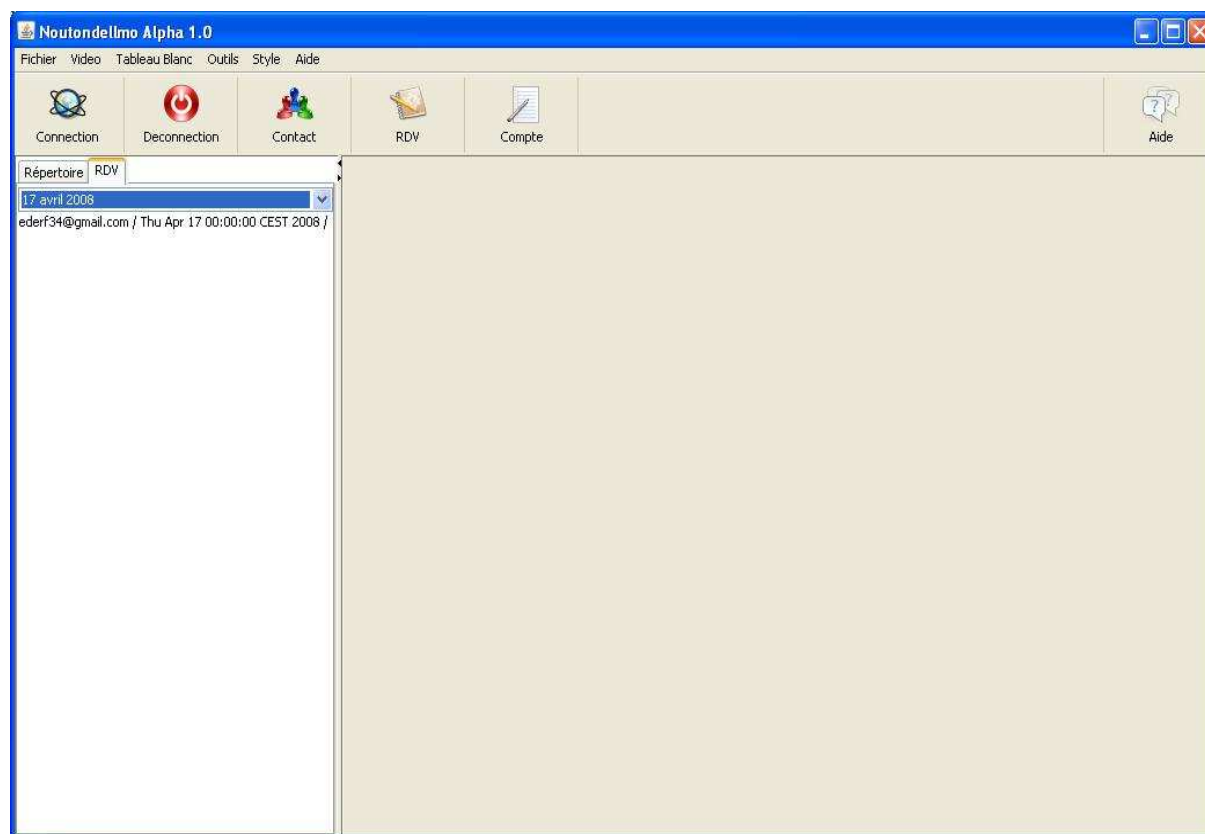


FIG. 4.17 – Le rendez-vous est pris

Lors de notre première rencontre avec l'entreprise cette fonctionnalité a été écartée car les rendez-vous étaient déjà gérés par une autre application. Cependant la possibilité d'intégrer ce module de rendez-vous à notre application de visioconférence (notamment en exportant des rendez-vous par le biais de fichier XML) a été envisagée. Cela pourrait faire l'objet d'une future évolution de notre application.

## 4.5.2 Mise à jour automatique

### Pourquoi ?

Afin d'assurer l'évolutivité de l'application, il est important que celle-ci puisse intégrer la possibilité de se mettre à jour automatiquement via Internet. Ainsi le logiciel peut proposer de nouvelles fonctionnalités à l'utilisateur, augmentant de ce fait sa durée de vie.

### Fonctionnement global

Le système de mise à jour se fait via Internet, l'utilisateur doit donc posséder une connexion Internet afin de télécharger les nouvelles versions du logiciel disponibles. Les différentes versions du logiciel se présente sous forme d'archive (.jar) contenant toutes les sources. Chaque version possède un numéro de version (par défaut ce numéro est incrémenté à chaque nouvelle version proposée).

La gestion du système de mise à jour se fait par l'intermédiaire d'un fichier XML hébergé par un serveur (administré actuellement par un membre du groupe). Ce fichier XML nous renseigne sur la dernière version du logiciel disponible. Il dispose de plusieurs balises contenant le numéro de version, l'url de l'archive correspondant et le répertoire de destination. Ainsi seule la dernière version du logiciel est disponible, les autres versions n'étant pas conservées.

```
<?xml version="1.0" encoding="UTF-8" ?>
<versions>
<version>
<nom>898</nom>
<files>
<file>
<url>http://projetinfos6.free.fr/VisioJava_1.jar</url>
<destination>NEW.jar</destination>
</file>
</files>
</version>
</versions>
```

Lorsque l'utilisateur veut effectuer une mise à jour, le logiciel va lire les différentes balises du fichier XML. Il compare ensuite le numéro de version proposé par le fichier XML à celui de la version installée sur l'ordinateur de l'utilisateur. Si ce numéro est différent, le logiciel propose de débiter le téléchargement et l'installation de la nouvelle version. Sinon l'utilisateur est averti qu'aucune nouvelle version n'est disponible. La mise à jour peut s'effectuer soit au démarrage de l'application, soit manuellement.

Dans le premier cas, l'application va automatiquement vérifier les mises à jour sans en informer l'utilisateur. Ici les différentes erreurs que peut rencontrer le système de mise à jour ne seront pas communiquées à l'utilisateur (problème de connexion, problème d'url...). Si une nouvelle version est disponible, on va alors proposer à l'utilisateur de l'installer s'il le désire, sinon l'application va se lancer normalement.

Dans le deuxième cas, l'utilisateur peut lui-même effectuer les mises à jour par le biais de l'option *Vérifier les mises à jour* dans la barre de menu. Dans ce cas là, si aucune mise à jour n'est disponible, on informe l'utilisateur que son système est à jour, sinon on lui propose le téléchargement et l'installation de la nouvelle version. Contrairement à la mise à jour automatique, on va ici gérer les différentes erreurs que l'on peut rencontrer et en informer l'utilisateur.

### 4.5.3 Internationalisation

#### Fonctionnement

Le projet, bien que destiné en tout premier lieu à des médecins français, doit tout de même pouvoir être utilisé par des experts de divers horizons. Ainsi, nous avons choisi de l'implémenter en français et en anglais, ce dernier choix étant considéré comme un choix « par défaut » pour les pays autres que l'Hexagone. En effet, nous prenons en compte le pays d'origine de l'ordinateur sur lequel est lancé notre logiciel. Pour cela, nous regardons dans quelle langue est configuré le système d'exploitation. De ce fait, lors de l'installation du logiciel sur un ordinateur « non français », le choix de la langue par défaut sera positionné sur « anglais ».

Une fois l'installation terminée, l'utilisateur a tout de même la possibilité de modifier la langue du logiciel. Pour cela, il lui suffit de passer par la barre de menu, et de choisir la langue qu'il désire utiliser.

Lorsque l'utilisateur a sélectionné la langue voulue, le logiciel lui demande de bien vouloir confirmer son choix. Après validation, l'application ferme et le logiciel est relancé dans la langue sélectionnée.

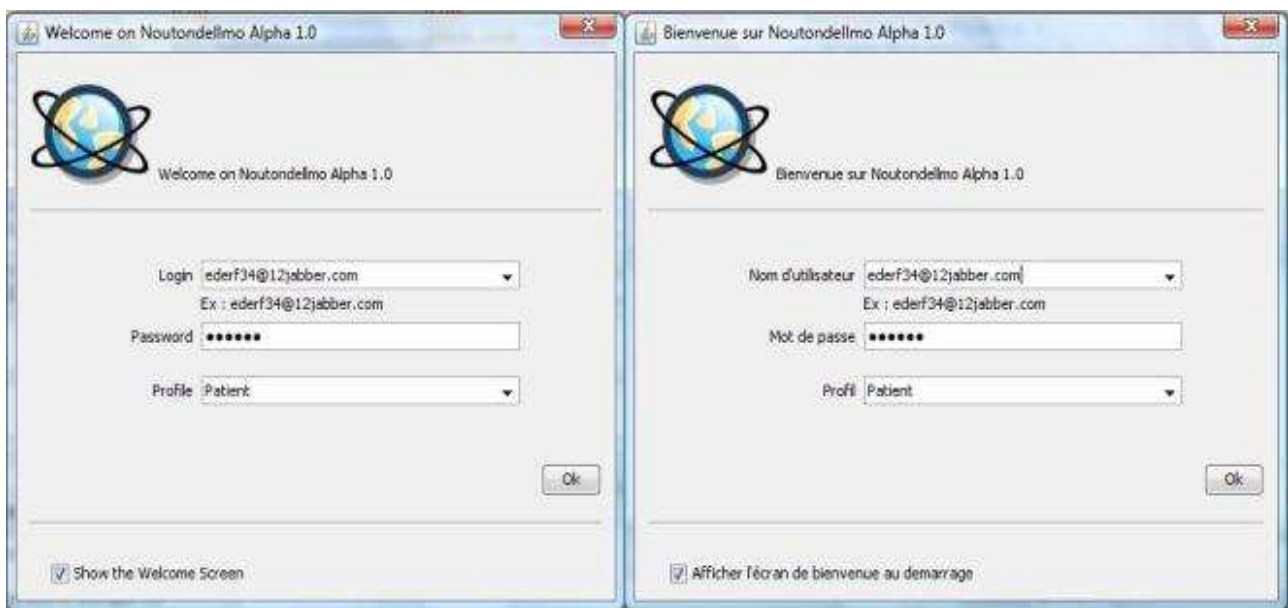


FIG. 4.18 – L'écran d'accueil en français et en anglais

## Implémentation

Pour implémenter cette fonctionnalité nous avons utilisé l'IDE Eclipse qui offre une interface particulièrement réussie pour réaliser l'internationalisation d'application.

Chaque composant (bouton, label, menu ...) de notre interface fait référence à une variable d'un dictionnaire plutôt qu'à une chaîne de caractère. Il est donc très simple de rajouter une nouvelle langue en redéfinissant tous les dictionnaires définis dans l'application.



The screenshot shows the Eclipse IDE interface with three tabs: 'ToolBarResources(default language)', 'ToolBarResources.properties', and 'WelcomeScreen.properties'. The 'WelcomeScreen.properties' tab is active, displaying a table of resource keys and their translations for three languages: default language, en\_GB - anglais (Royaume-Uni), and fr - français.

Key	default language	en_GB - anglais (Royaume-Uni)	fr - français
serverName	Server Name	Server Name	Server Name
login	Login	Login	Nom d'utilisateur
affichageScreen	Show the Welcome Screen	Show the Welcome Screen	Afficher l'écran de ...
welcomeLabel	Welcome on	Welcome on	Bienvenue sur
ok	Ok	Ok	Ok
profile	Profile	Profile	Profil
password	Password	Password	Mot de passe
ex	Ex : ederf34@12jabbe...	Ex : ederf34@12jabbe...	Ex : ederf34@12jabbe...

FIG. 4.19 – Le dictionnaire de l'écran d'accueil

Si l'utilisateur décide de changer la langue après l'installation, celle-ci est sauvegardée dans un paramètre de base de registre (comme pour la fenêtre d'accueil) et lue au redémarrage de l'application.



## 4.5.4 Packaging de l'application

### Création d'un installateur et d'un exécutable Windows

Afin de finaliser notre projet, nous avons créé un exécutable et un installateur pour les plateformes Windows, puisque Expertise Radiologie utilise de préférence ce système d'exploitation. Plusieurs logiciels existant permettent de les réaliser, et après en avoir rapidement fait le tour, nous avons choisi Inno Setup pour la création du Wizard d'installation et Launch4j pour la création du .exe (fichier exécutable).

L'application que nous avons développée nécessite l'utilisation de la bibliothèque JMF (Java Media FrameWork) pour l'affichage de la vidéo. Nous devons donc créer un processus d'installation amorçant l'installateur JMF.

D'autre part, plusieurs solutions sont possibles pour lancer une application Java : créer un fichier Jar exécutable ou un fichier exécutable avec launch4j qui lance la commande Java en spécifiant le classpath. Le classpath est une variable contenant la concaténation de tous les chemins d'accès des librairies essentielles au lancement de l'application. Les deux solutions précédentes ne prennent en compte que les librairies nécessaires à notre programme qui se trouvent dans le dossier d'installation de JMF (JMF pouvant être installé à des endroits différents suivant l'ordinateur). Ceci implique que nous devons trouver une solution pour lancer le logiciel avec un classpath dynamique (non défini à la création de l'exécutable).

Après une recherche approfondie, nous avons résolu le problème en créant un exécutable avec Launch4j, initiant un lanceur qui démarre l'application en spécifiant le bon classpath (Figure 4.20).

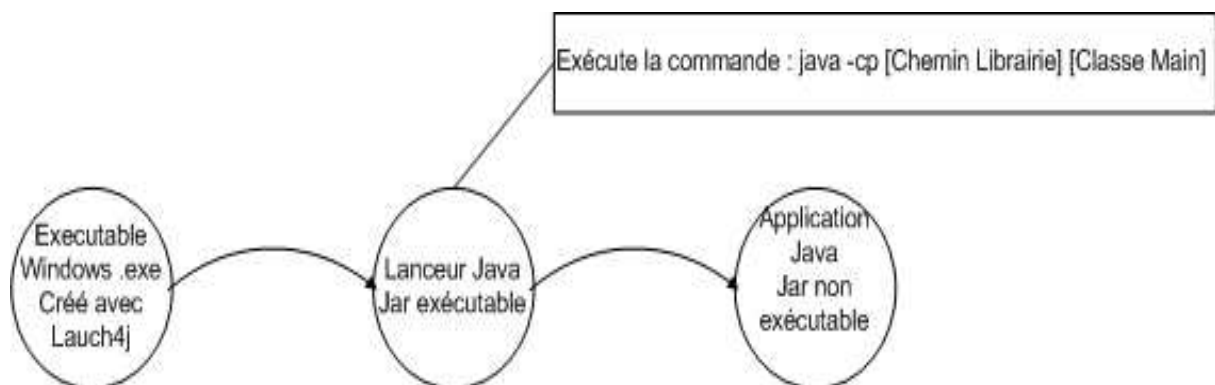


FIG. 4.20 – Fonctionnement du lanceur

Nous avons commencé par modifier l'installateur Delphi généré avec Inno Setup, pour y intégrer ensuite une nouvelle fenêtre demandant à l'utilisateur où est installé JMF.

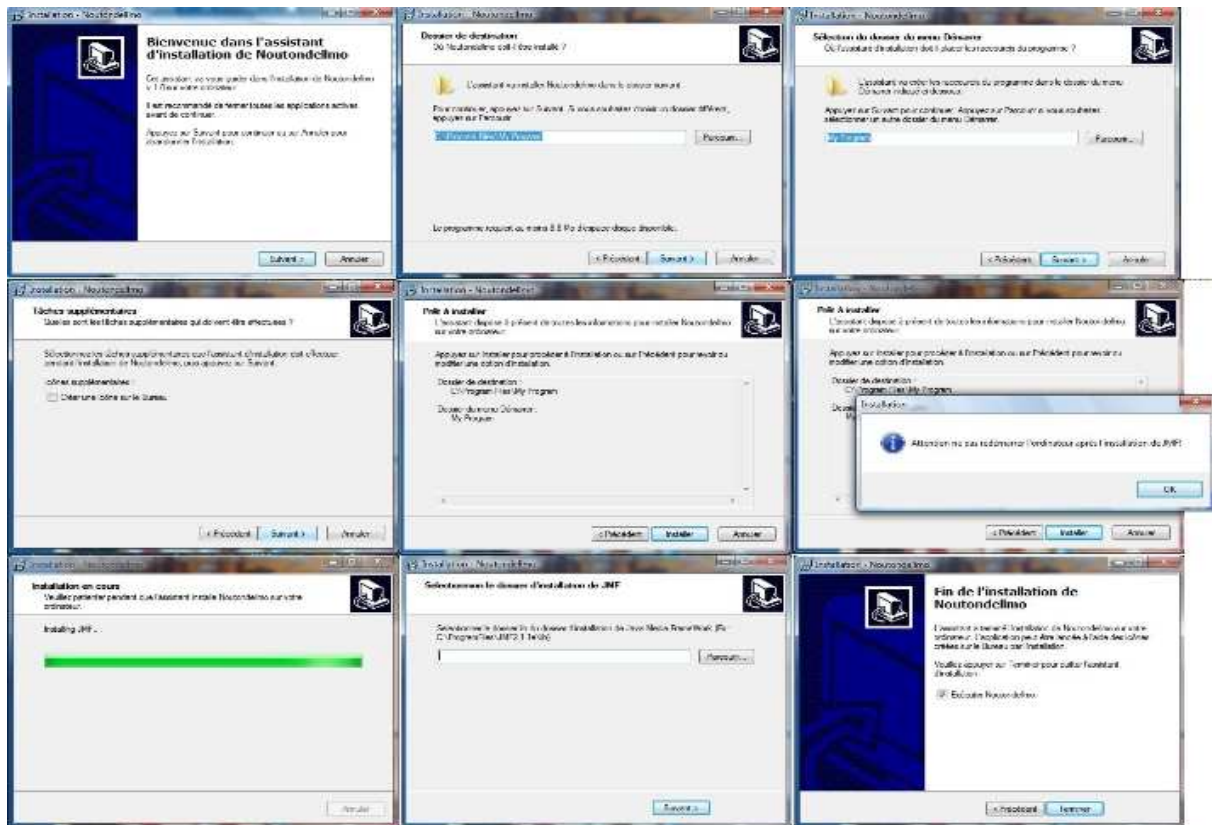


FIG. 4.21 – Les différentes fenêtres de l'installation

Ce chemin est mémorisé dans une variable d'environnement du système d'exploitation. Nous avons ensuite implémenté un lanceur qui récupère cette variable et exécute la commande suivante :

```
Java -cp "C :/Program Files/JMF2.1.1e/lib/jmf.jar";"C :/Program Files/JMF2.1.1e/lib/mediaplayer.jar";"C :/Program Files/JMF2.1.1e/lib/sound.jar";./lib/javadatepicker.jar; ./lib/jmf.jar; ./lib/mediaplayer.jar; ./lib/smack.jar; ./lib/smackx.jar; ./lib/sound.jar; ./NTDM.jar; kernel.Main
```

## Automatisation de cette tâche avec ANT

Le programme ANT a pour mission de compiler et de regrouper automatiquement divers composants d'un projet Java en tenant compte des dépendances entre ces derniers. Il remplit à ce titre le même rôle que la commande make, bien connue des programmeurs C et C++. Nous avons donc utilisé cet outil pour automatiser la tâche expliquée précédemment. Le gain de temps est considérable lorsque nous voulons tester le logiciel dans son futur environnement pendant la phase de développement. Le fichier XML de paramétrage d'ANT est disponible en annexe. On peut y reconnaître les fonctions de création et suppression de dossier, copie de fichier, compilation, création de jar et lancement d'application externe comme launch4j ou Inno Setup.

## 4.6 Divers : installation d'Openfire sur le serveur d'Expertise Radiologie

### Descriptif d'Openfire

Openfire est un serveur Jabber libre (GPL) écrit en Java par Jive Software, la même compagnie éditrice de l'API Smack que nous utilisons. Openfire bénéficiant du soutien d'une importante communauté, nous nous sommes naturellement tournés vers ce serveur. Il est stable et est réputé pour sa facilité d'installation et d'administration. De plus, nous étions sûrs qu'il prendrait en compte toutes les fonctionnalités de l'API Smack et du client que nous allions lui fournir. En ce qui concerne les avantages d'Openfire nous pouvons citer :

- support complet de XMPP.
- facile à installer.
- bénéficie d'une interface d'administration très complète et intuitive.
- peut s'interfacer avec un grand nombre de composants externes (bases de données, annuaires LDAP, etc.).
- intègre de nombreux services (serveur de discussion, pubsub, proxy de transfert de fichiers, etc.).
- permet l'utilisation de nombreux plugins (et bien sûr d'en créer ce qui nous envisagions pour récupérer l'IP de nos clients).

Le seul défaut à noter est son impossibilité d'avoir des hôtes virtuels (un seul nom de domaine par serveur) mais cet aspect est négligeable et ne correspond pas à nos demandes.

### Installation

Le déploiement de l'application que nous avons développée pour Expertise Radiologie comprend l'installation d'un serveur Jabber. Chaque docteur aura ainsi un compte sur le serveur et pourra s'identifier sur le serveur Jabber de l'entreprise.

Nous nous sommes donc connectés avec le terminal serveur de Windows, qui permet de se connecter à distance sur un ordinateur. Puis nous avons procédé à l'installation de la machine virtuelle Java et d'Openfire le serveur Jabber que nous avons sélectionné.

Expertise Radiologie sauvegardant toutes les expertises réalisées ou en cours sur ce serveur, il était hors de question de faire une mauvaise manipulation. En guise de prévention, toutes les actions réalisées sur le serveur ont donc été listées dans un compte rendu envoyé à Expertise Radiologie (voir annexes). Cette opération s'est terminée avec succès, nous avons pu nous connecter et créer des comptes sur le serveur grâce à notre application.

## 5 Discussion

Ce projet, bien qu'étant laissé à la charge d'étudiants, doit répondre aux attentes fixées par des professionnels et doit s'inscrire dans un cadre d'utilisation bien précis. Notre application répond aux principaux objectifs listés dans le cahier des charges et incorpore donc les fonctionnalités qui correspondent à ces besoins.

Cependant, certaines d'entre elles se rapprochent de ces objectifs sans toutefois y répondre parfaitement. En effet, un large choix concernant l'implémentation nous a été donné dû au manque d'échanges réguliers avec l'entreprise initiatrice du projet. Nous aurions souhaité être suivis dès l'origine du développement afin d'être recadrés régulièrement et nous permettre ainsi d'obtenir des résultats plus proches du logiciel désiré.

Nous regrettons également de ne pas avoir disposé de plus de temps afin pour implémenter de nouvelles fonctionnalités ou d'améliorer celles déjà existantes. En effet, il aurait été intéressant de pouvoir intégrer le logiciel de gestion de rendez-vous déjà utilisé par l'entreprise ce qui aurait pu fournir une double fonctionnalité à notre application.

Ce projet nous a également demandé une certaine adaptation. En effet, un grand nombre des fonctionnalités de l'application étaient orientées réseau, avec lequel nous ne sommes pas forcément familiers puisque tous issus du parcours Génie Logiciel. C'est toutefois cette possibilité de travailler sur deux aspects différents qui nous a orienté lors du choix du TER.

## 6 Conclusion

Les principaux objectifs fixés à l'origine du projet étant atteints, nous pouvons affirmer que ce projet est une réussite, bien que de multiples évolutions soient encore possibles. L'interface utilisateur est conviviale et facile à prendre en main, la visioconférence fonctionne et la radiographie numérique, sur laquelle on peut dessiner, est bien disposée au coeur de l'ensemble.

Nous avons également rajouté à notre logiciel des fonctionnalités que nous avons jugées utiles et intéressantes, bien qu'elles n'aient pas été exigées à l'origine. Ainsi, un installeur d'application, un système de mise à jour et l'internationalisation du logiciel ont été greffés au projet.

Nous restons cependant pleinement satisfaits du travail accompli tout au long du développement de ce projet ainsi que du résultat obtenu. Quant à l'expérience acquise, tant en développement Java qu'en organisation et en travail en équipe, elle ne peut nous être que bénéfique pour la suite de notre cursus scolaire et pour notre intégration prochaine dans le monde du travail.

# 7 Bibliographie

## Expertise Radiologie

<http://expertise-radiologie.com/default.aspx>;

## Logiciels étudiés

- JBrother : <http://www.jbrother.org/index.rb?command=screenshots>;
- Smack API 3.0.4 : <http://www.igniterealtime.org/projects/smack/index.jsp>;
- VLC : <http://www.vlvc.net/fr-downloads.html>;
- Openwengo : <http://openwengo.orgopenwengo.org>;
- Windows Live Messenger : <http://www.windowlive.fr/messenger/default.asp>;
- Skype : <http://www.skype.com/intl/fr/>;

## Références diverses

- JavaStyle : <http://java.sun.com/docs/codeconv/CodeConventions.pdf>;
- Trac : <http://trac.edgewall.org/>;
- VPN : <http://www.frameip.com/vpn/>;
- XMPP protocols : <http://www.xmpp.org/protocols/>;
- RFC de XMPP : <http://www.ietf.org/rfc/rfc3920.txt>; <http://www.ietf.org/rfc/rfc3921.txt>; <http://www.ietf.org/rfc/rfc3922.txt>; <http://www.ietf.org/rfc/rfc3923.txt>;
- Extension de XMPP XEP : <http://www.xmpp.org/extensions/>;
- Smack doc : <http://www.igniterealtime.org/builds/smack/docs/latest/javadoc/>;
- *Programmez!* : N°93 janvier 2007 - Les nouveautés de Java - XML;
- *Programmez!* : N°105 février 2008 - JAVA - Utiliser l'API Preferences
- Java Media Framework API : <http://java.sun.com/products/java-media/jmf/>
- Manuel ANT : <http://ant.apache.org/manual/CoreTasks/>
- Inno Setup : <http://www.jrsoftware.org/isinfo.php>
- Launch4j : <http://launch4j.sourceforge.net/>

# A Annexes

## A.1 Analyse et conception

Quelques explications concernant le diagramme de classes :

Le diagramme de classes de conception qui suit est un diagramme succinct de notre application. Nous avons choisi de diviser le contrôleur de l'application en deux pour ne pas alourdir le premier (classe Controller) puisqu'il gère déjà les événements de tous les sous composants graphiques (MenuBar, Toolbar, ...).

Le deuxième contrôleur (classe ControllerContact) est responsable de toute la partie liée à la création de conversation depuis le PanelContact et à la récupération de tous les messages. GeneralChatPane est la classe incluse dans chaque onglet de conversation.

## A.1.1 Diagramme de classes

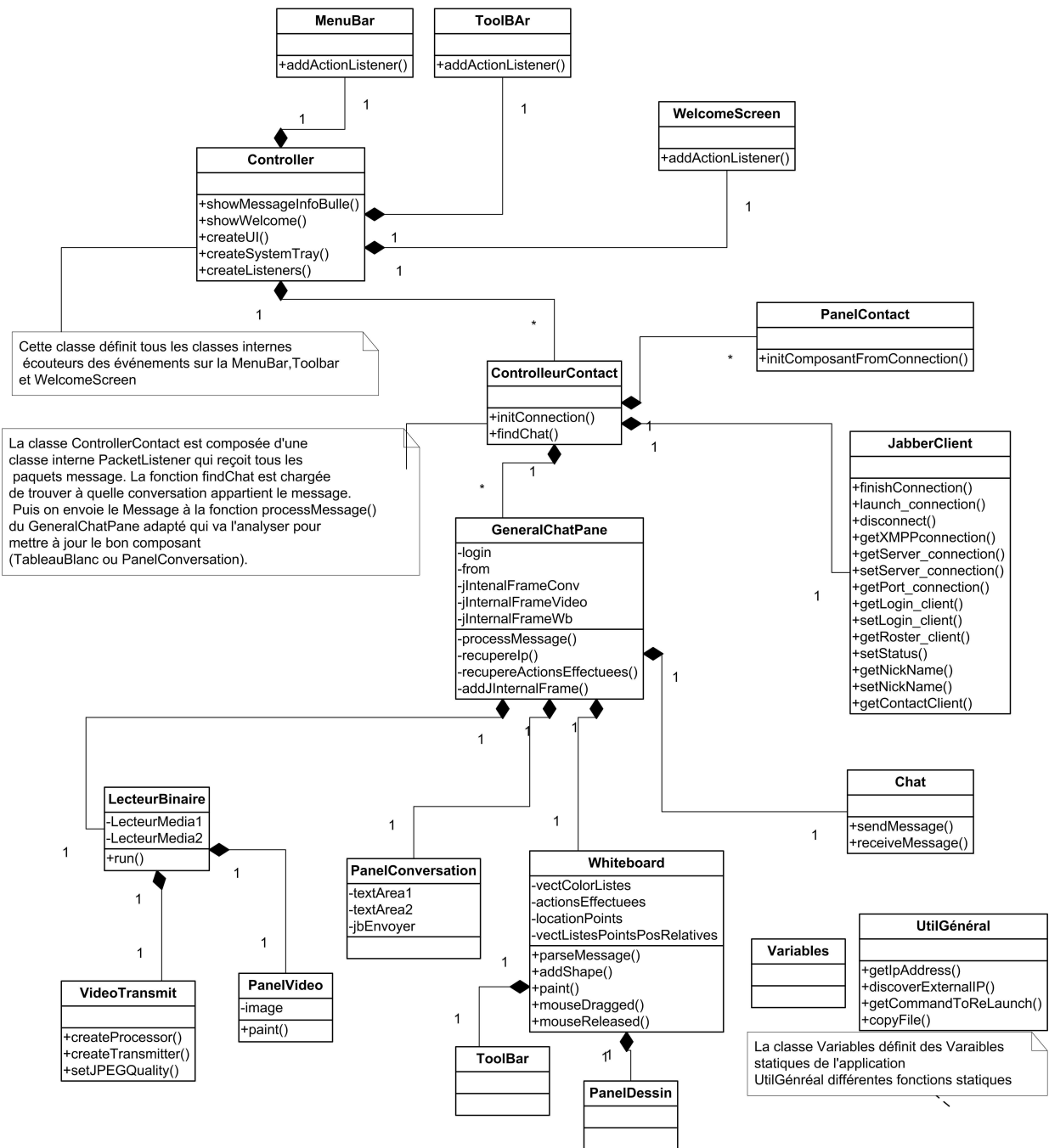


FIG. A.1 – Diagramme de classes



## A.1.2 Diagramme de cas d'utilisation



FIG. A.2 – Diagramme de cas d'utilisation

### A.1.3 Diagramme de déploiement

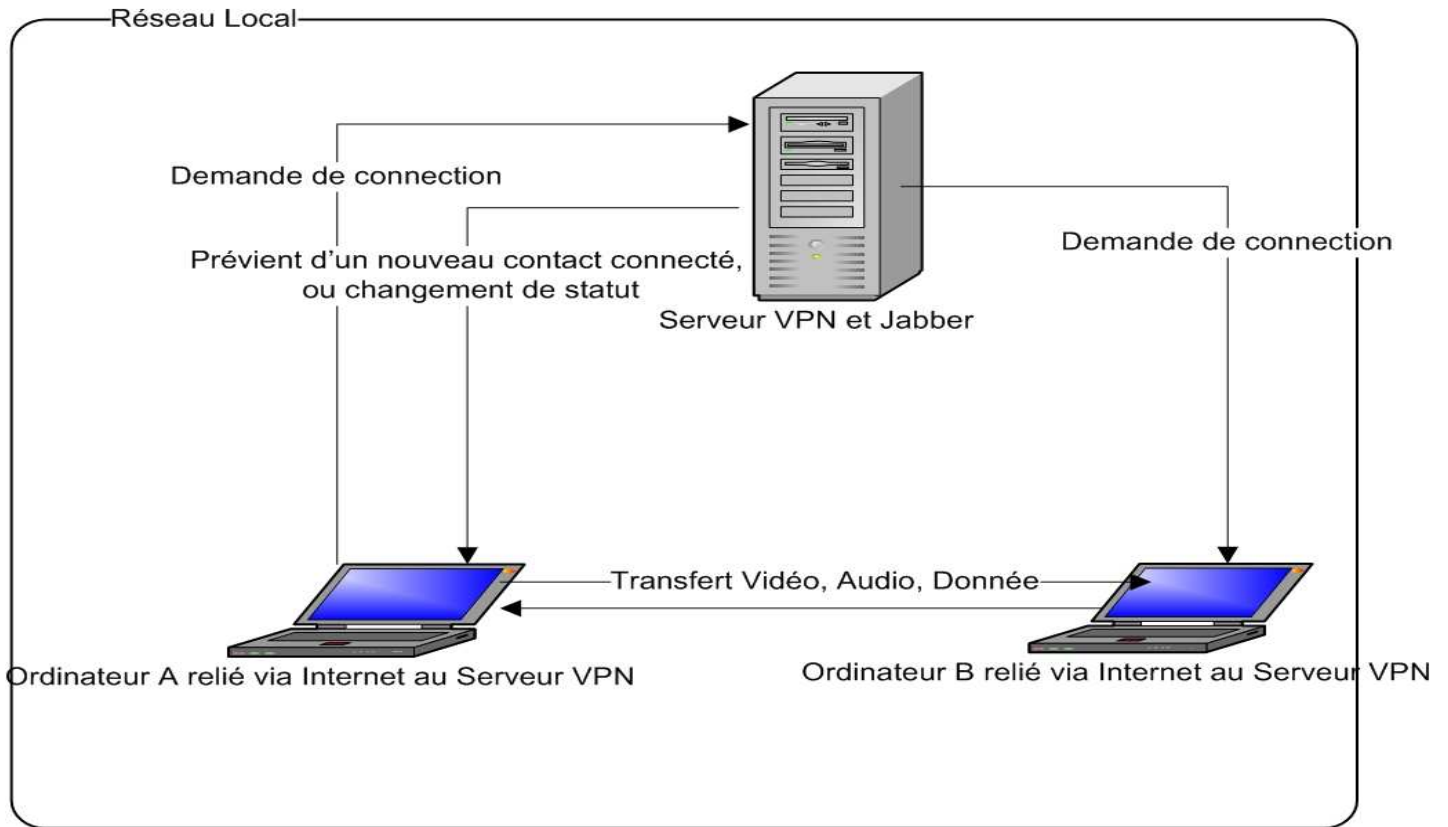


FIG. A.3 – Diagramme de Déploiement

## A.2 Fichier XML de paramétrage d'ANT

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="changeme" default="all" basedir=".">
  <target name="all">

    <!-- Création du dossier classes -->
    <mkdir dir="classes"/>
    <!-- Création du dossier lanceur -->
    <mkdir dir="lanceur"/>

    <!-- Compilation des classes du programme -->
    <javac compiler="modern" srcdir=".." destdir=".." classpath="C:\Users\Frederic\workspace\libsmack\smack_3_0_4\smack.jar;C:\Users\Frederic\workspace\libsmack\smack_3_0_4\smackx.jar" />
    <!-- Compilation des classes du lanceur -->
    <javac srcdir="./lanceur/" destdir="lanceur"/>

    <!-- Création du dossier install/lib -->
    <mkdir dir="install/lib"/>

    <copy file="C:\Users\Frederic\workspace\libsmack\smack_3_0_4\smack.jar" tofile="install/lib/smack.jar" />
    <copy file="C:\Users\Frederic\workspace\libsmack\smack_3_0_4\smackx.jar" tofile="install/lib/smackx.jar" />
    <copy file="C:\Users\Frederic\workspace\libdatepicker\javadatepicker.jar" tofile="install/lib/javadatepicker.jar" />

    <!-- Création du fichier NTDM.jar -->
    <jar destfile="install/NTDM.jar" basedir=".." />

    <!-- Création du fichier lanceur.jar -->
    <jar destfile="install/lanceur.jar" basedir="lanceur" manifest="MANIFEST.MF" />

    <!-- Envoi du jar sur un serveur pour la mise à jour auto -->
    <ftp server="ftpperso.free.fr"
        userid="projetinfos6"
        password="4d4FGt9E" verbose="True">
      <fileset dir="install/">
        <include name="NTDM.jar"/>
      </fileset>
    </ftp>

    <!-- Suppression du dossier classes -->
    <delete dir="lanceur"/>

    <delete defaultexcludes="false">
      <fileset dir=".." includes="**/*.class"/>
    </delete>
    <delete dir="classes"/>

    <!-- Création de l'executable visiojava.exe avec Launch4j -->
    <exec executable="C:\Program Files\Launch4j\launch4jc.exe">
      <arg value="\${basedir}\installeurLanceurLaunch4j.xml"/>
    </exec>

    <!-- Création du fichier setup.exe avec Inno Setup -->
    <exec executable="C:\Program Files\Inno Setup 5\ISCC.exe">
      <arg value="\${basedir}\setup.iss"/>
    </exec>

    <!-- Suppression des dossier install et classes-->
    <delete dir="install"/>
  </target>
</project>
```

FIG. A.4 – Fichier XML de paramétrage d'ANT

## A.3 Manuel de l'utilisateur

### A.3.1 Comment se connecter ?



FIG. A.5 – Connexion

Un formulaire de connexion s'ouvre au lancement de l'application. Pour se connecter à l'application depuis un compte Jabber, il faut spécifier son login (*mon login@le nom du serveur*), son mot de passe et le profil d'utilisation.

### A.3.2 Comment se déconnecter ?



FIG. A.6 – Déconnexion

Pour se déconnecter, il suffit de cliquer sur l'icône *Déconnexion* dans la barre d'outils (qui est ici en anglais).

### A.3.3 Comment démarrer une conversation ?

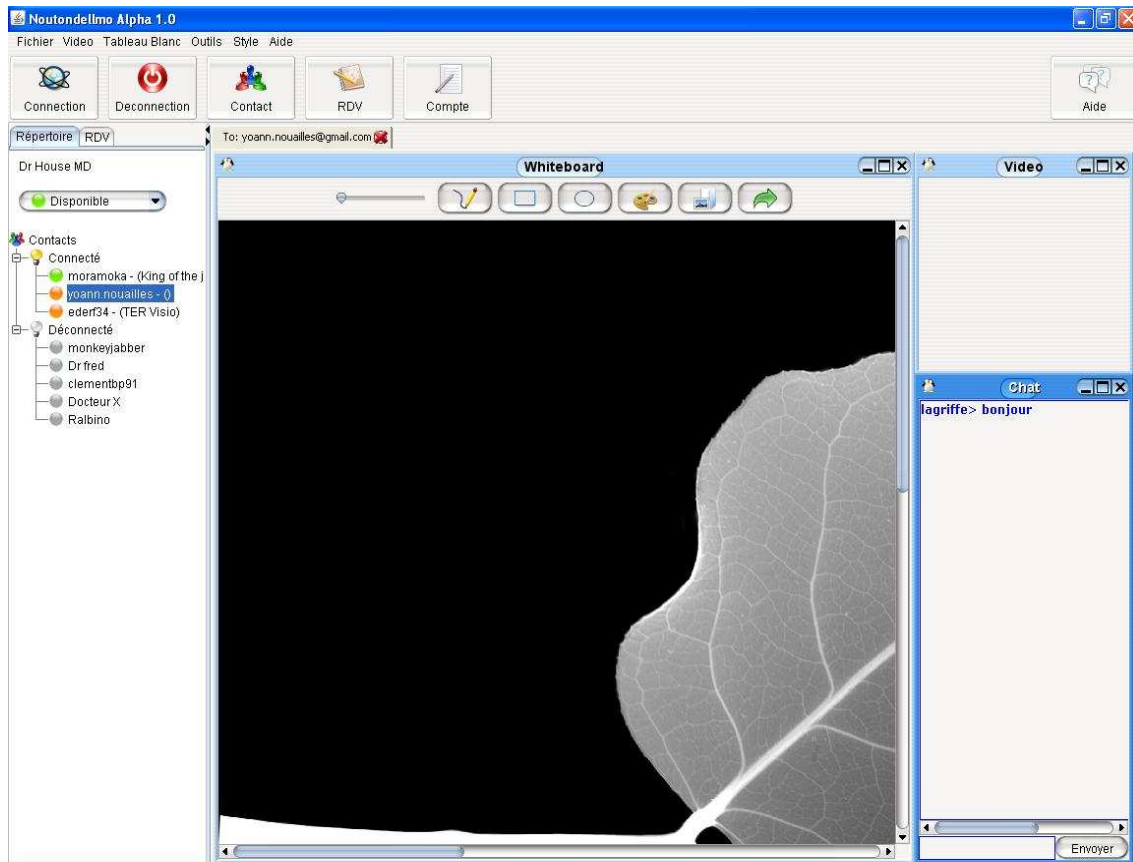


FIG. A.7 – Une conversation

Pour démarrer une conversation avec un contact connecté, il suffit de cliquer sur l'adresse du contact dans l'onglet *Répertoire*. Une nouvelle conversation va alors s'ouvrir. On ne peut pas engager de conversation avec un contact indisponible ou démarrer deux conversations avec le même contact.

### A.3.4 Comment interagir avec le tableau blanc ?

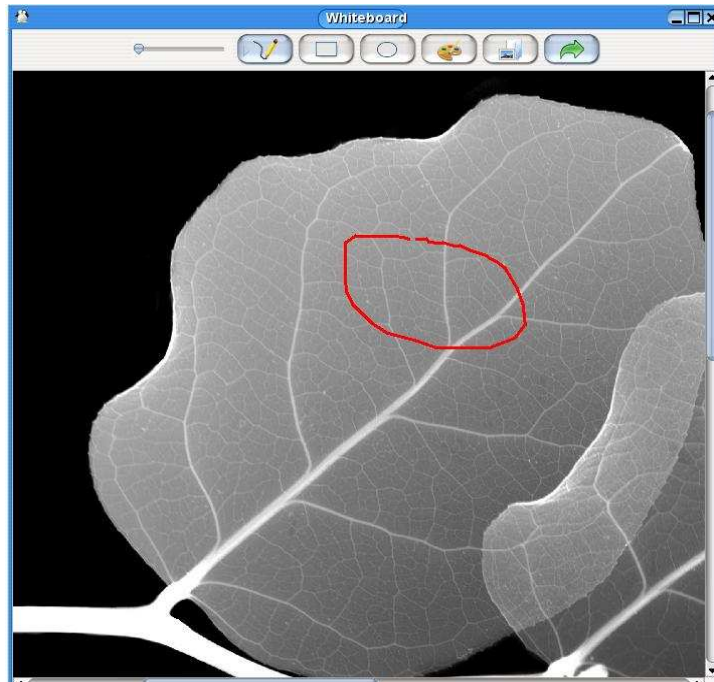


FIG. A.8 – Les dessins sur le tableau blanc

Pour interagir avec le tableau blanc il faut préalablement avoir démarré une conversation avec un contact. Une fois le panneau de conversation ouvert, on doit spécifier l'option de dessin utilisée (main levée, rectangle, cercle, couleur) et l'on peut ensuite dessiner sur l'image affichée. Une fois le dessin terminé il faut valider les modifications pour les envoyer à son interlocuteur.

### A.3.5 Comment créer un nouveau compte Jabber ?

Pour créer un nouveau compte Jabber il faut cliquer sur l'icône *compte* dans la barre d'outils.



FIG. A.9 – Le bouton *Compte*

Un formulaire de saisie va s'ouvrir dans lequel il est demandé de saisir les paramètres de ce nouveau compte (login, mot de passe, le serveur).

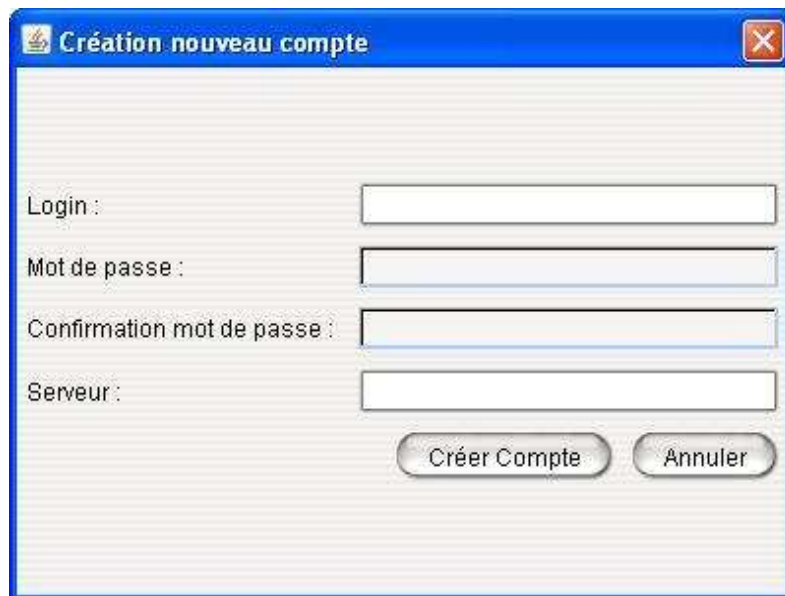
The image shows a dialog box titled "Création nouveau compte" with a close button in the top right corner. The dialog contains four text input fields: "Login :", "Mot de passe :", "Confirmation mot de passe :", and "Serveur :". At the bottom of the dialog, there are two buttons: "Créer Compte" and "Annuler".

FIG. A.10 – Le formulaire

### A.3.6 Comment ajouter un contact ?

Pour ajouter un nouveau contact au répertoire il faut cliquer sur l'icône *Contact* dans la barre d'outils. On doit ensuite entrer l'adresse du contact.



FIG. A.11 – Ajout d'un contact

### A.3.7 Comment consulter l'aide ?

Pour consulter l'aide il suffit de cliquer sur l'icône *Aide* dans la barre d'outils.

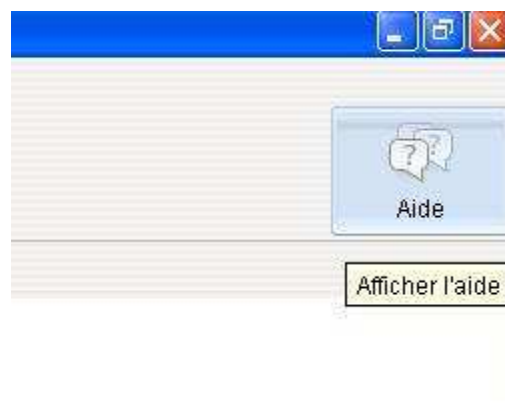


FIG. A.12 – L'icône d'aide



### A.3.8 Comment planifier un rendez-vous ?

Pour planifier un rendez-vous avec un contact il faut cliquer sur l'icône *RDV* dans la barre d'outils.

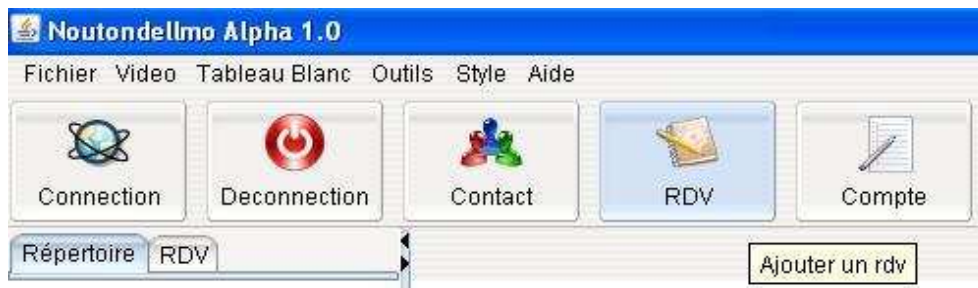


FIG. A.13 – L'icône *RDV*

Un formulaire de saisie va s'ouvrir dans lequel il faut spécifier le contact, la date et l'heure du rendez-vous.



FIG. A.14 – Le formulaire de saisie

Les rendez-vous sont consultables dans l'onglet *RDV*.

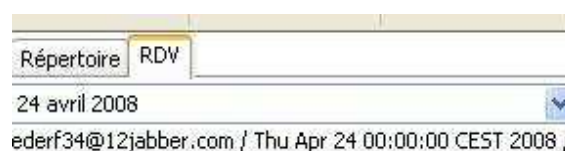


FIG. A.15 – L'onglet *RDV*

### A.3.9 Comment changer la langue du logiciel ?

Pour changer la langue du logiciel il faut cliquer sur *Outils / Français* ou *Anglais* dans la barre de menu.



FIG. A.16 – Changement de la langue

### A.3.10 Comment changer de login ?

Pour changer de login il faut cliquer sur la zone de texte placée en haut de l'onglet *Répertoire*. On pourra alors saisir un nouveau login.

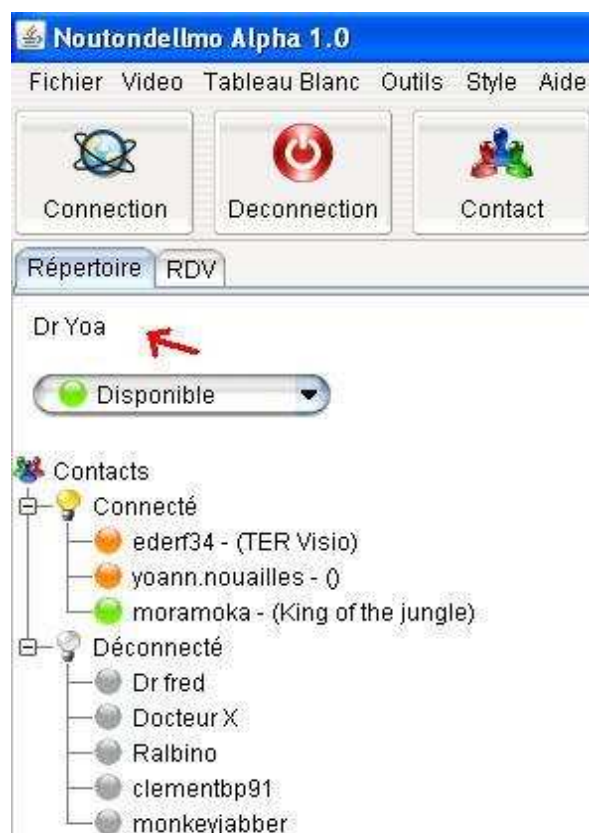


FIG. A.17 – Changement de login

### A.3.11 Comment mettre à jour le logiciel ?

Pour mettre à jour le logiciel il faut cliquer sur *Outils / Vérifier les mises à jours* dans la barre de menu.



FIG. A.18 – Mise à jour de l'application

### A.3.12 Comment changer de statut ?

Pour changer de statut de connexion il faut cliquer sur le menu déroulant présent dans l'onglet *Répertoire*. On pourra alors choisir un nouveau statut.

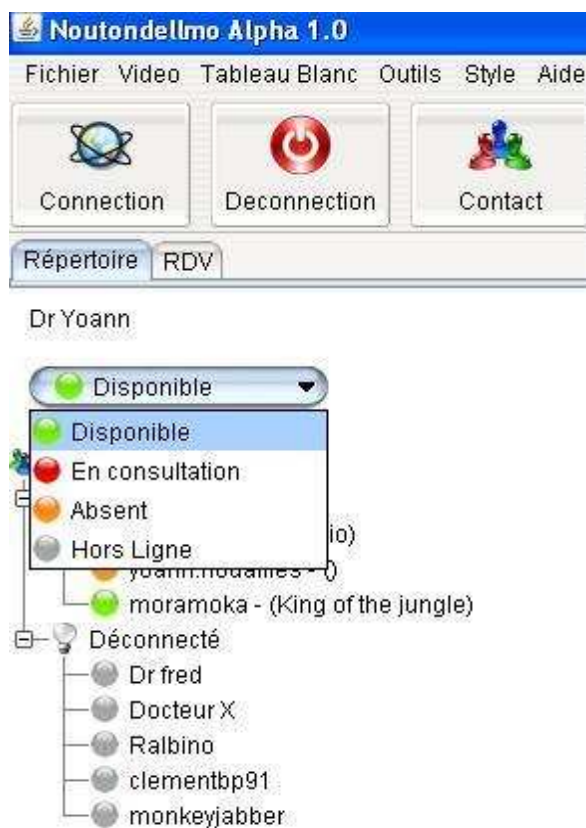


FIG. A.19 – Changement de statut

### A.3.13 Comment changer le look and feel de l'application ?

Pour changer le look and feel de l'application il cliquer sur *Style / nouveau look and feel* dans la barre de menu.



FIG. A.20 – Changement du look and feel

## A.4 Compte rendu

Ces manipulations ont été faites lors du rendez-vous téléphonique du 16/04/08 avec M.SAPEDE Frederic de la société Expertise Radiologique

Listes des manipulations :

- 1) Terminal Server depuis Windows (connexion ordinateur à distance) 193.227.249.198  
— *connexion réussie OK (20h20)*
  
- 2) Téléchargement OpenFire + JRE depuis le serveur (Serveur JABBER Utilisant la librairie Smack)  
— *téléchargement OK (20h30)*
  
- 3) Création dossier jabber à la racine contenant les exécutables (C :/Jabber/OpenFire.exe)
  
- 4) Installation de OpenFire sur le serveur  
— *installation terminée OK (20h35)*
  
- 5) Lancement et Configuration du serveur (depuis localhost 127.0.0.1) (20h37)
  - 5.1) Paramètres du serveur :  
Domaine : exp001 ; Port de la console d'administration : 9090 ; Port sécurisé de la console d'administration : 9091 (On garde les paramètres par défaut)
  - 5.2) Paramètres de la base de données :  
- Installation de la base de données embarquées (hsqldb)
  - 5.3) Paramètres profil :  
- on garde les paramètres par défaut
  - 5.4) Compte administrateur :  
Adresse : frederic.sapede@expertise-radiologie.com  
Mot de passe : erjabber2008  
— *configuration terminée OK (20h49)*
  
- 6) Lancement de la console d'administration avec admin et mot de passe  
— *OK*

Installation terminée, tout semble fonctionner correctement.

— *Premier test de création de compte OK!!*