

Programmation Orientée Objet

SYSTÈME CLAVARDAGE
// system name : The Chit-Chat //

AFONSO PERRINE - SOUMARE Coumba
4IR_B

**The
Chit Chat**



Table des matières

-----Introduction -----	2
1.Architecture du projet.....	3
A. ChatSystem.....	3
1.1 Organisation des packages.....	3
1.2 Connexion en broadcast.....	3
1.3 Communication en TCP.....	4
B. Web Presence Server.....	4
2. Manuel d'installation & d'utilisation.....	4
A. ChatSystem.....	4
2.1 Récupération du "logiciel".....	4
2.2 Interface de connexion.....	5
2.3 Interface d'accueil.....	5
B. Web Presence Server.....	7
2.4 Installation.....	7
2.5 Utilisation.....	8
3. Choix d'implémentations, technologiques.....	8
3.1 Notre base de données sur SQLite.....	8
3.2 L'interface sous Swing.....	9
3.3 Maven.....	10
3.4 Adaptation du système.....	10
4. Batterie de tests.....	10
4.1 Test avec 2 ordinateurs.....	11
4.2 Test avec 3 ou 4 ordinateurs.....	12
4.3 Tests ChitChatOutdoor.....	13
5. Futures améliorations.....	13
5.1 Renforcement de sécurité.....	13
5.2 Adaptation sous Android.....	13
5.3 Regroupement du ChatSystem & du Web Presence Server.....	14

5.4 Ajout de fonctionnalités.....	14
-----------------------------------	----

-----**Conclusion**-----

15

Introduction

Lors de ce premier semestre de 4IR, nous avons participé au projet de création d'un système de Clavardage. Dans ce projet de programmation Orientée Objet, nous avons pour objectif la création d'une application de clavardage fonctionnelle respectant un cahier des charges établissant un ensemble de fonctionnalités.

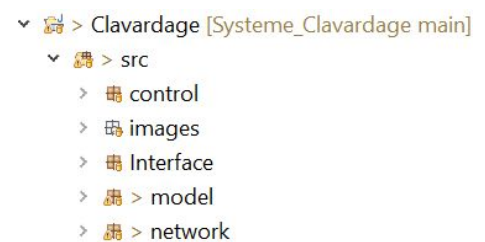
Le rapport suivant rassemble ainsi une présentation de notre projet et de nos choix d'implémentation mais également un manuel d'utilisation permettant la prise en main de notre application The ChitChat.

1. Architecture du projet

A. ChatSystem

1.1 Organisation des packages

Notre projet est organisé selon le pattern MVC, le code étant regroupé par fonctionnalités. La partie "Contrôleur" est représentée par le package **control**, la vue est gérée par le package **interface** et la partie "Modèle" équivaut au package du même nom : **model**. Nous avons aussi un dernier package utile à toutes les fonctions UDP et TCP nommé **network**.



Afin de centraliser nos informations et de n'avoir qu'une instance de chaque attribut, nous avons regroupé les différents attributs utilisés par les classes dans la classe Application du package control. Ces attributs correspondent à :

- a) l'utilisateur connecté sur le système
- b) la liste de contacts connecté associé à ce dernier
- c) l'instance de la classe InteractiveChatSystem permettant à l'utilisateur de gérer connexion, déconnexion et changement de pseudo
- d) la base de données de l'utilisateur (pour la gestion d'historique)
- e) l'instance de la classe UDP Runner permettant d'être à l'écoute en udp

1.2 Connexion en broadcast

La première phase lors de l'utilisation de l'application de clavardage est celle de connexion. Pour celle-ci, l'objectif est pour l'utilisateur de pouvoir se connecter à l'aide d'un pseudonyme dont l'unicité a été vérifiée et en obtenant la liste des utilisateurs avec lesquels il pourrait converser.

Pour assurer ces propriétés, un broadcast UDP est effectué. En tentant de se connecter, l'utilisateur entre un pseudonyme. Son passeport User contenant son pseudonyme, son adresse IP et son numéro de port est alors envoyé en broadcast sur le réseau. Chaque utilisateur connecté recevant ce paquet répond

selon deux cas : le pseudo du nouvel entrant est différent du sien ou le pseudo est identique au sien et donc indisponible.

Un utilisateur n'a donc la possibilité de se connecter que si chacun des autres utilisateurs connectés répond et ne possède pas déjà le pseudonyme qu'il souhaite adopter. En recevant les réponses des autres utilisateurs, le nouvel entrant les ajoutera par la même occasion à sa liste de contacts.

Ici, le broadcast se fait en UDP. En effet, ce mode de transport ne possède aucune contrainte de qualité de service. Pour un besoin en connexion rapide, c'est donc le mode adapté. En TCP, les contraintes de connexions rapides et avec des délais d'attente moindres ne seraient pas respectées.

1.3 Communication en TCP

Cependant une fois la liste de contacts établie, la communication est effectuée en TCP. La liste de contact est d'ailleurs mise à jour à chaque connexion ou déconnexion des utilisateurs. La communication en TCP permet, elle, de respecter la qualité de service en fiabilité. En effet, nous cherchons ici un échange de messages sans perte.

B. Web Presence Server

Pour ce qui est de la deuxième partie du projet, permettant la connexion d'utilisateurs en dehors du réseau local à travers un serveur, nous avons décidé de reprendre la plupart des classes de notre projet initial. Nous voulions simplement réadapter les fonctions préalablement modulées par une communication udp par de nouvelles procédures reliées à notre serveur. Cela comprend la connexion, le changement de pseudo, la déconnexion ainsi que la récupération des contacts connectés. L'envoi de message est toujours effectué dans une communication TCP. La classe rassemblant les fonctions de gestion des messages udp correspond à la classe InteractiveChatSystem, dont l'instance est un attribut de l'application dans le projet initial. Pour cette deuxième partie, nous avons tout simplement géré ces fonctions directement dans la classe Application.

2. Manuel d'installation & d'utilisation

A. ChatSystem

2.1 Récupération du "logiciel"

Afin de pouvoir utiliser The Chit-Chat, il faut tout d'abord récupérer le logiciel. Pour ça, il vous faut cloner le répertoire git suivant:

```
git clone https://github.com/peafonso/Systeme\_Clavardage
```

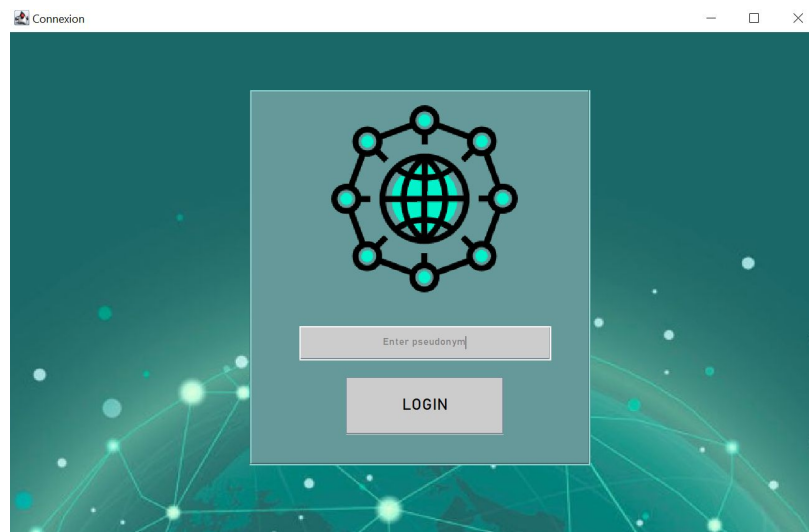
Il vous faudra ensuite vérifier que le **dossier sqLite est bien vide** (prêt à accueillir votre base de données). Une fois cette vérification faite, vous pourrez alors lancer l'exécutable "ChitChat". Ouvrez un invite de commandes dans le dossier cloné et lancer cette commande :

```
java -cp Chitchat/target/ChitChat.jar Interface.AppInterface
```

Vous pouvez aussi tester directement en cliquant sur le fichier exécutable ChitChat dans le dossier target du dossier ChitChat.

2.2 Interface de connexion

Vous serez donc accueilli par notre interface de connexion. Celle-ci vous permet de choisir votre pseudo (qui doit tenir en moins de 12 caractères). Une fois votre pseudo choisi et renseigné, vous pouvez alors entrer dans le système en appuyant sur entrée ou en cliquant sur le bouton "LOGIN".




Si le message "Only 12 characters are allowed" est affiché, il vous faut opter pour un pseudo plus court. En revanche, si le message "Pseudonym already in use. Try Again." apparaît, c'est qu'un autre utilisateur est déjà connecté avec le même pseudo et pour garantir l'unicité vous devez en choisir un nouveau. Si aucun message n'apparaît, votre pseudo est alors validé par le système et vous entrerez dans l'interface d'accueil.

2.3 Interface d'accueil

L'interface d'accueil vous permet de voir l'ensemble des utilisateurs connectés sur le système à droite de l'écran.

Pour lancer une conversation avec un de vos contacts, il vous faut

- Sélectionner le destinataire dans la liste
- Taper votre message
- L'envoyer avec entrée ou en appuyant sur le bouton send

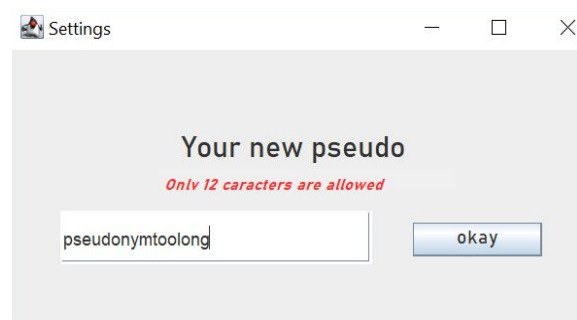
Lorsque vous appuyez sur le pseudo d'un de vos contacts, vous lancez une conversation. Cela entraîne l'apparition du pseudo de l'utilisateur avec qui vous discutez à côté du label "Talking with". Cela vous permet de savoir à qui vous parlez à tout instant. De plus, en cas de trou de mémoire, vous pourrez retrouver votre pseudo sur l'écran à côté de l'icône .

Vous recevrez également toutes les notifications de connexions, déconnexions, changements de pseudos et messages entrants dans le rectangle au-dessus de la zone de lecture de conversation.

La barre de menus du système situé en haut de la page est composée de deux items. Le premier **Settings** est lui même constitué de deux options :

a) Change Pseudo

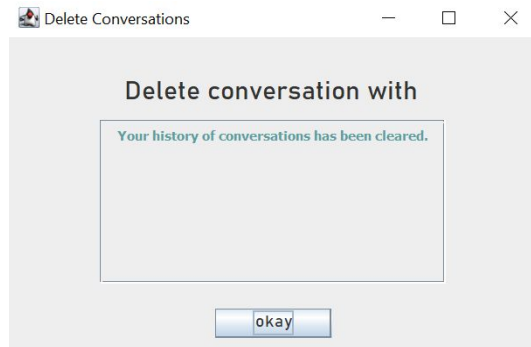
Cette première option permet à l'utilisateur de changer son pseudo initialement choisi. Cependant, les conditions restent les mêmes : pas plus de 12 caractères et le pseudo doit être unique.



b) Clear Conversations

Cette deuxième option du menu Settings permet de supprimer une conversation. C'est -à -dire de nettoyer l'historique d'une conversation. Pour cela, il faut que le contact soit connecté.

Une fois la fenêtre ouverte, la liste des contacts connectés est affichée. Il faut alors sélectionner l'utilisateur dont on veut supprimer la conversation, puis appuyer sur le bouton "okay". Bien entendu, cela n'efface l'historique que pour la personne effectuant l'opération.



Le deuxième et dernier item de notre barre de menus est l'onglet **Disconnect**.



Il propose à l'utilisateur une nouvelle chance de rester connecter. En appuyant sur le bouton "no", la fenêtre Disconnect se ferme, et rien ne change sur l'interface d'accueil. En revanche, en appuyant sur "yes", l'utilisateur se déconnecte et cette action entraîne donc la fermeture de l'application. Cependant, ce n'est pas la seule manière de se déconnecter : en fermant la page Home, l'utilisateur se déconnecte automatiquement.

A savoir : en vous déconnectant par la fenêtre Disconnect, l'application se ferme correctement mais suite à un bug, il vous faudra terminer le processus par un ctrl+c dans l'invite de commandes où vous aviez lancer la commande java. (En vous déconnectant par la fermeture de la page (croix rouge en haut à droite), vous n'aurez pas ce problème).

B. Web Presence Server

2.4 Installation

L'installation de la 2ème partie est similaire à la première, il vous faut tout d'abord cloner le répertoire git suivant:


```
git clone https://github.com/peafonso/Test
```

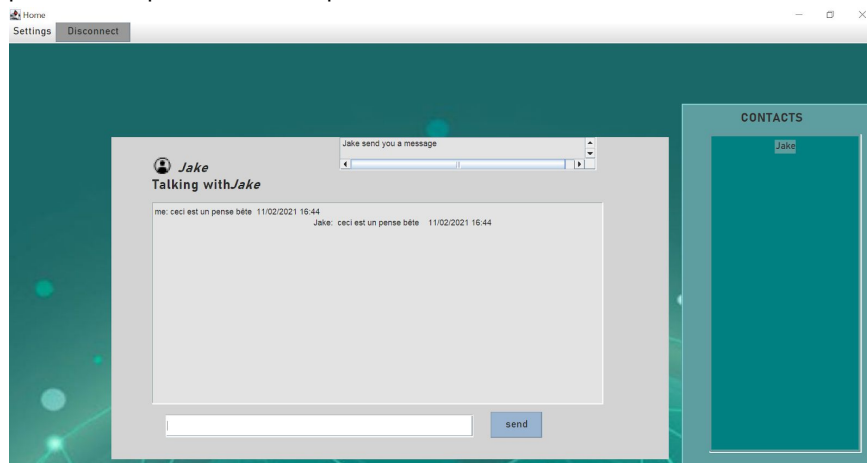
De la même manière, il vous faudra ensuite vérifier que le **dossier sqlite est bien vide**. Il faut rester conscient que cette partie n'a pas pu être terminée et que nos tests n'ont pas pu être réalisés comme ils se devaient. L'utilisation peut donc être altérée. (Nous avons réalisé nos tests sur Eclipse avec le "Run on Server" sur Tomcat Apache 9, l'url utilisé est `http://localhost:8080/Test/welcome`.)

Le fichier war se trouve dans le dossier, il faut lancer **Test.war** sur un serveur. Puis lancer cette commande dans un invite de commandes ouvert dans le dossier Test cloné :

```
java -cp ChitChatOutdoor.jar Interface.AppInterface
```

2.5 Utilisation

Pour ce qui est de l'utilisation, elle reste la même que lors du premier projet. Cependant, il y a un nouvel outil : il est possible de parler à soi-même. Cela peut servir par exemple en tant que bloc notes.



3. Choix d'implémentations, technologiques

3.1 Notre base de données sur SQLite

La base de données est l'outil principal de notre gestion d'historique. Nous avons fait le choix d'avoir une base de données locale sur machine locale. L'hypothèse étant que *chaque utilisateur se connecte toujours avec le même hôte*.

Nous avons opté pour la technologie SQLite pour plusieurs raisons. La première étant bien sûr la facilité d'implémentation mais aussi d'utilisation. Les seules missions que nous voulions que notre base de données effectuent sont satisfaites par SQLite : créer plusieurs tables, y stocker des données en assurant leur intégrité et la possibilité d'y accéder facilement. Ce système est aussi public

et gratuit d'utilisation. Enfin, il nous permet de gérer notre base de données sans serveur et sans authentification, uniquement par accès aux fichiers sur le disque.

Pour rentrer dans les détails de notre base de données, nous avons décidé de créer une table par conversation. Chaque table est nommée selon l'adresse IP de la personne avec qui la conversation est effectuée, sachant que l'adresse IP étant le seul attribut ne changeant jamais étant donné que le pseudo est volatile. Les entrées dans les tables correspondent aux messages (entrants et sortants). Nous avons donc définis 4 colonnes pour nos tables :

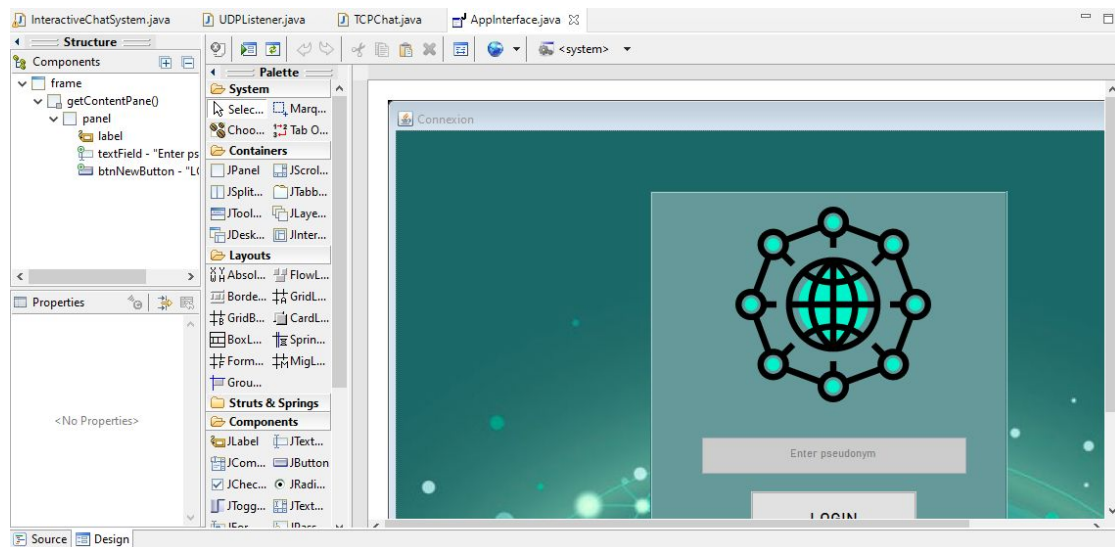
- * **id** , de type integer, représentant les numéros des messages (1 étant le plus ancien)
- * **time**, de type text, représentant l'horodatage des messages
- * **message**, de type text, représentant les messages envoyés par les participants de la conversation
- * **sender**, de type integer, permettant de savoir l'expéditeur du message (0 -> on a envoyé le message 1 -> on a reçu le message)

Chatwith_IP2
id: integer
time : text
message : text
sender : integer

3.2 L'interface sous Swing

Pour ce qui est de notre interface graphique, nous avons adopté Swing. Celui-ci offrant la possibilité de créer des interfaces identiques quel que soit le système d'exploitation. De plus, pour nous faciliter la mise en place des différentes classes de la vue de notre application, nous avons utilisé le plugin WindowBuilder.

Une fois installé sur Eclipse, ce plugin permet une modification et une prise en main graphique simplifiée. En effet, au lieu de coder directement chaque partie de l'interface mot à mot, ce plugin fonctionne à travers l'utilisation d'une fenêtre *design*. En créant une nouvelle classe, il suffit de spécifier celle-ci en tant qu'*Application Window*. Ainsi, dans l'onglet de code il apparaît un bouton design ouvrant l'onglet de mise en page ci-dessous:



Nous pouvons au choix modifier le code source ou la mise en page à travers le mode design.

3.3 Maven

Par soucis de facilité de prise en main de notre application, nous nous sommes appuyées sur l'utilisation de Maven. Ainsi, en intégrant le plugin *Eclipse m2e* nous avons pu convertir notre projet en un projet Maven.

Le format nous permet ainsi d'obtenir un fichier directement exécutable après le téléchargement de notre projet, et ce via une commande sur le terminal ou directement via le lancement de l'exécutable. Ceci pour une expérience utilisateur facilitée et la plus intuitive possible.

3.4 Adaptation du système

Afin d'adapter notre application sur un ensemble de systèmes d'exploitations, nous avons placé notre base de données à l'intérieur du dossier de notre projet.

Nous avons d'ailleurs pu tester l'application *The Chit-Chat* sur deux environnements : Windows (environnement de prédilection) et Linux. Ces tests ont été concluants, malheureusement nous n'avons pas pu les effectuer sur des machines sous OS X et Android.

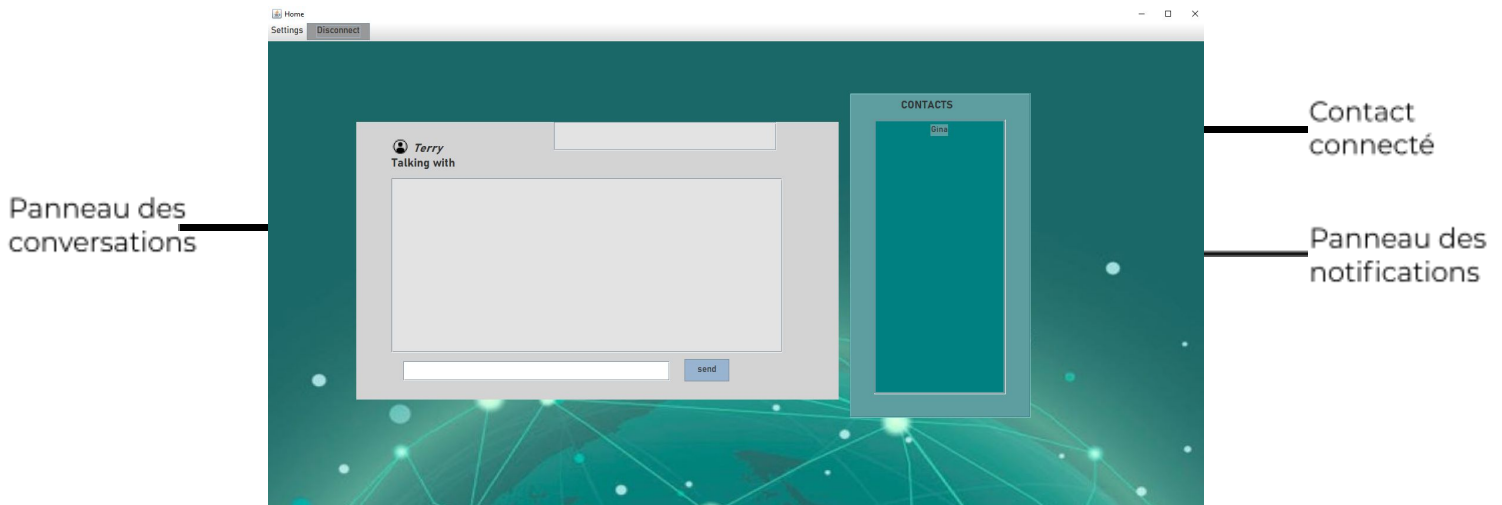
4. Batterie de tests

Nos tests peuvent se présenter en deux majeures parties. La première partie représentant l'ensemble de nos tests unitaires effectués au cours du développement de l'application assurant la vérification des fonctions ajoutées au code. Cela comprend notamment les tests de Broadcast, tests de communication en TCP, tests de l'affichage interface, tests de la création et de la modification de la base de données, etc. Et ce, jusqu'à l'obtention d'une application testable composée de l'ensemble de ses sous-fonctions testées au fur et à mesure.

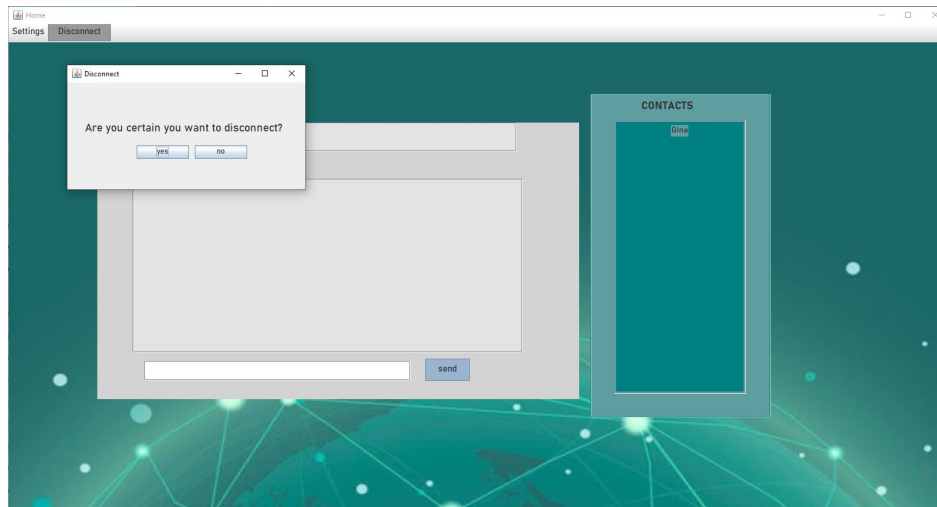
La seconde partie des tests constitue ainsi l'ensemble des tests de l'application entre plusieurs ordinateurs. Cette deuxième partie nous a permis d'améliorer le projet mais aussi de vérifier le respect du cahier des charges. Ainsi, cette batterie de tests se présente comme la checklist de l'ensemble des fonctionnalités de l'application pour une communication entre 2 ordinateurs ou plus.

4.1 Test avec 2 ordinateurs

- **Phase de connexion fonctionnelle**, les deux utilisateurs se connectent sur leurs sessions respectives avec leur propre pseudo
 - > Si pseudonyme supérieur à 12 caractères cela n'est pas autorisé OK
 - > Si pseudo déjà utilisé par la premier utilisateur à se connecter: affichage "*pseudonym already in use*" OK
- **Phase de communication fonctionnelle**
 - > Possibilité de sélectionner le destinataire dans la liste des contacts OK



- > Possibilité d'envoyer un message à ce destinataire OK
 - > Réception d'une notification à la réception d'un message en TCP OK
 - > Affichage du message reçu dans la fenêtre des conversations OK
 - > Ouverture de la base de données OK
 - > Ajout des messages dans la base de données (historique) OK
- **Paramètres et fonctionnalités supplémentaires fonctionnels**
 - > Possibilité dans les settings de modifier son pseudonyme OK
 - > Changement de pseudonyme envoyé à tous les contacts OK
 - > Affichage du nouveau pseudonyme chez les contacts OK
 - > Possibilité dans les settings d'effacer la conversation avec l'autre utilisateur quand celui-ci est connecté (nettoyage historique) OK
- **Phase de déconnexion fonctionnelle**
 - > Possibilité de se déconnecter à travers le bouton "disconnect" ou à travers la fermeture de la fenêtre OK
 - > L'autre utilisateur sur le réseau est notifié de la déconnexion OK
 - > Notre pseudonyme est retiré de la liste des contacts connectés de l'autre utilisateur OK



4.2 Test avec 3 ou 4 ordinateurs

- **Phase de connexion fonctionnelle**

- > Ce test a permis les réglages des problèmes de connexion. En effet au dessus de 2 utilisateurs il apparaissait des problèmes lié au bind des sockets lors de la connexion du troisième utilisateur
- > Test connexion (pour connexion à 1/2/3 et 4 PC) OK
- > Test notifications connexions OK

- **Paramètres et fonctionnalités supplémentaires opérationnels**

- > Tests et réglages des problèmes de pseudonymes (changement de pseudonyme problématique au-dessus de 2 personnes utilisant l'application). A plus de deux utilisateurs, lorsque l'un des utilisateurs changeait de pseudonyme en adoptant un pseudonyme déjà utilisé par l'un des autres utilisateurs, il recevait une réponse lui indiquant que le pseudonyme n'était pas disponible mais d'autres réponses lui autorisant l'utilisation du pseudonyme. Il ne retenait ainsi que les réponses positives et changeait de pseudonyme. Ces tests nous ont permis de remédier à ce problème.
- > Test changement de pseudo (pour connexion à 1/2/3 et 4 PC) OK
- > Test notifications changements de pseudos ok
- > Test historique de conversation OK
- > Test nettoyage conversations ok

- **Phase de communication fonctionnelle**

- > Test envoi et réception de messages OK
- > Test notifications messages entrants OK

- **Phase de déconnexion fonctionnelle**

- > Test déconnexion (par bouton Disconnect ou fermeture fenêtre pour connexion à 1/2/3 et 4 PC) OK
- > Test notifications déconnexions OK

4.3 Tests ChitChatOutdoor

Nous avons testé la deuxième partie sur un ordinateur, la connexion ainsi que le changement de pseudonyme fonctionnent. Nous pouvons avoir une conversation “pense bête” avec nous-même, que nous pouvons aussi supprimer à l’aide de la suppression d’historique. La déconnexion fonctionne aussi correctement.

5. Futures améliorations

5.1 Renforcement de sécurité

Dans cette première implémentation de l’application, les contraintes en termes de sécurité sont moindres. La fonctionnalité du logiciel a pour l’instant était privilégiée. Cependant, cela apporte plusieurs pistes d’améliorations concernant le renforcement de la sécurité de notre application ChitChat.

Une première idée serait tout simplement l’ajout d’un mot de passe de connexion. En effet, pour le moment l’obtention du logiciel à télécharger et l’utilisation d’un pseudonyme unique suffit pour l’accès à une session de clavardage. Toutefois, cette session sera liée à l’adresse IP de l’utilisateur. Ainsi, un utilisateur utilisant le pseudonyme d’un utilisateur s’étant déconnecté ne suffit pas pour obtenir sa base de données contenant l’historique de ses conversations.

On pourrait également se demander ce qu’il se passerait si un utilisateur usurpait l’adresse IP d’un autre utilisateur (IP Spoofing) sur le réseau concerné.

5.2 Adaptation sous Android

Pour le moment, l’application a été conçue et testée pour fonctionner sous Windows ou Linux. Essentiellement pour des raisons techniques et matérielles, les tests sur Mac et sous Android n’ont pu être effectués. Une application fonctionnant sur Android serait une amélioration non négligeable en termes de portabilité et de facilité d’utilisation.

5.3 Regroupement du ChatSystem & du Web Presence Server

Tout d'abord, par manque de test, notre Web Presence Server n'est pas 100% opérationnel, il faudrait alors nous concentrer sur la finalisation de celui-ci. Une fois opérationnel, il serait alors intéressant de le fusionner au premier projet ChatSystem. En effet, nous avons présentement deux projets pour deux types d'utilisateurs : indoors & outdoors. Une amélioration serait donc de regrouper ces deux projets afin que les indoors et outdoors users puissent se reconnaître et communiquer entre eux.

5.4 Ajout de fonctionnalités

- Une première fonctionnalité qui pourrait être ajoutée à l'historique serait la possibilité d'avoir accès à celui-ci lorsque l'utilisateur avec qui on a discuté précédemment n'est plus connecté. Pour le moment l'accès à l'historique des communications se fait à travers la liste des contacts qui se met à jour lorsqu'un utilisateur se connecte ou se déconnecte. Ainsi, lorsqu'un utilisateur se déconnecte, il est enlevé de la liste des contacts et on ne peut ni lui envoyer de message ni accéder à la conversation que l'on a eu avec lui.
- Une seconde fonctionnalité serait dans la même optique de mettre en place une liste immuable des utilisateurs possédant l'application. On posséderait ainsi une liste contenant l'ensemble des utilisateurs avec une indication précisant si celui-ci est connecté ou déconnecté. En consultant un contact de la liste on obtiendrait ainsi la conversation en cours ou que l'on souhaite débiter. Ceci rejoint la première fonctionnalité proposée en appuyant la possibilité de converser de manière asynchrone, une option non traitée dans cette première version de l'application.
- Une autre amélioration serait l'extension du projet avec la possibilité de ne pas seulement pouvoir partager des messages textuels mais également des fichiers et images dans une certaine limite de taille. Ceci serait évidemment une fonctionnalité non négligeable dans le cadre d'une utilisation professionnelle de notre application.

Conclusion

Ce projet nous a ainsi permis de mettre en œuvre l'application de clavardage The ChitChat. En nous appuyant sur des notions de Programmation Orientée Objet et de réseau, l'application conçue permet alors la communication entre 2 utilisateurs ou plus sur un réseau local ou non. Cela inclut les fonctionnalités d'envoi, de réception de messages, de changement de pseudonyme et de création d'un historique des conversations.

Des améliorations restent à ajouter pour obtenir une application optimalement fonctionnelle, cependant l'application The ChitChat remplit notre objectif principal de respect du cahier des charges.