INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
TOULOUSE

Département de Génie Électrique & Informatique

Rapport Projet Clavardage

Conception Orientée Objet

14 février 2021

Étudiants :

Auriane	LARTIGUE	alartigu@etud.insa-toulouse.fr
Nabil	MOUKHLIS	moukhlis@etud.insa-toulouse.fr

Enseignants :

Sami	YANGUI
Meryem	BILLAMI

Table des matières

Introduction

Le projet ChatApp a pour but de créer une application de clavardage qui sera déployée sur des ordinateurs d'un même réseau local. Cela permettrait à deux utilisateurs connectés en même temps de pouvoir échanger des messages textuels. Afin de créer cette application, nous devions passer par les indispensables phases d'analyse du cahier des charges et de conception. La phase de conception et l'élaboration des divers diagrammes a plusieurs vocations. La première est de nous assurer d'avoir saisi toutes les subtilités du cahier des charges. A savoir toutefois que les diagrammes ci-dessous se concentrent uniquement sur l'architecture peer-to-peer de l'application, c'est-à-dire celle ne considérant qu'un réseau local et pas d'utilisateurs distants devant passer par un serveur de présence. La deuxième vocation de ces diagrammes est bien entendu de nous permettre de poser les premières bases de notre conception. Une fois les diagrammes bien établis, nous aurons toutes les cartes en main pour mettre en place le squelette de l'application et les principales classes du projet. Ce n'est qu'une fois tout cela effectué que l'implémentation peut être commencée de manière efficace.

1 Diagramme de cas d'utilisation

Dans ce diagramme de cas d'utilisation, nous avons effectué une première approche du projet au travers de son cahier des charges. En analysant les différentes demandes du client (dont quelques-unes sont retranscrites dans le tableau ci-dessous), nous avons modélisé les interactions possibles entre les différents acteurs et le système.

Le système pourra être déployé sur un poste de travail fonctionnant sur le système d'exploitation Windows, Linux, Android et OS X. L'utilisateur pourra bien entendu réduire l'agent.
L'utilisateur pourra choisir un pseudonyme unique à tout moment. Ce changement de pseudonyme n'influe pas sur les sessions de clavardage en cours. Les autres utilisateurs du réseau seront avertis automatiquement du changement.
L'utilisateur aura accès à la liste des utilisateurs actuellement connectés sur le réseau local. Chaque utilisateur pourra entamer et stopper des sessions de clavardage avec ces utilisateurs actifs. Tous les messages échangés seront horodatés et l'historique des messages sera accessible.

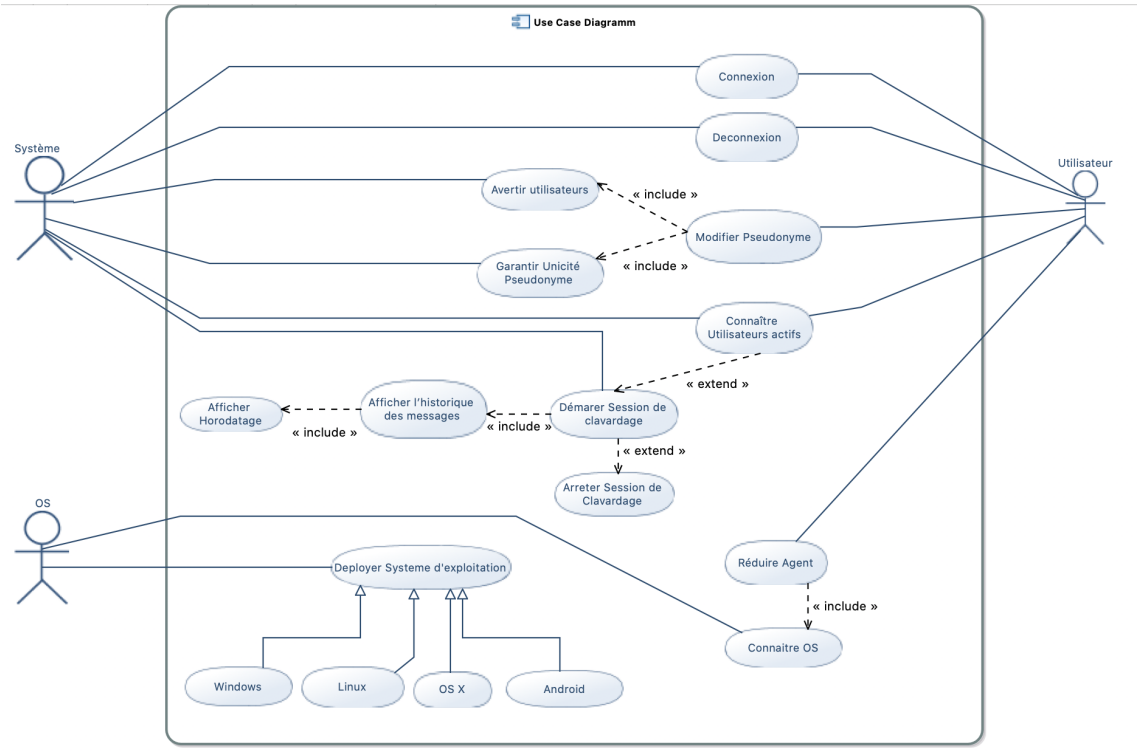

FIGURE 1 – Diagramme de cas d'utilisation

Nous considérons ici 3 acteurs, l'utilisateur qui va agir directement sur l'agent, le système qui va permettre à l'utilisateur de faire tout type d'actions nécessitant de passer par le réseau et l'OS qui va gérer la compatibilité de l'agent avec le système d'exploitation dans lequel il est déployé. La présence de l'acteur "système" sur ce diagramme s'explique par le fait que notre première approche de l'architecture du projet était de passer en permanence par un serveur qui aurait pour rôle de faire le lien entre tous les utilisateurs. Comme nous avons par la suite revu notre conception de l'architecture, cet acteur "système" représente en réalité les autres utilisateurs du réseau avec lesquels l'utilisateur principal communique.

2 Diagramme de séquence

Les différents diagrammes de séquence répertoriés dans cette section permettent de mettre en lumière les échanges au cours du temps des différents composants du système. Il est ainsi plus simple de visualiser le fonctionnement des différentes phases rencontrées lors du cycle de vie de l'application.

2.1 Diagramme récapitulatif

Le diagramme ci-dessous récapitule les fonctionnalités générales du système de chat. Vous pouvez remarquer 4 acteurs différents : Utilisateur qui va interagir sur l'agent, Chat app qui correspond au contrôleur de l'application, l'OS et le Réseau qui représente tous les utilisateurs actifs sur le réseau. En effet, notre système fonctionnant sur des échanges UDP, lorsque l'utilisateur utilisant l'application a besoin d'effectuer certaines actions (connexion, changement de pseudonyme, déconnexion), il lui est nécessaire de demander l'autorisation aux autres membres du réseau. Tous ces autres membres sont contenus dans l'acteur Réseau. Ce même acteur Réseau représente le second utilisateur avec lequel on communique en TCP lors d'une session de clavardage.

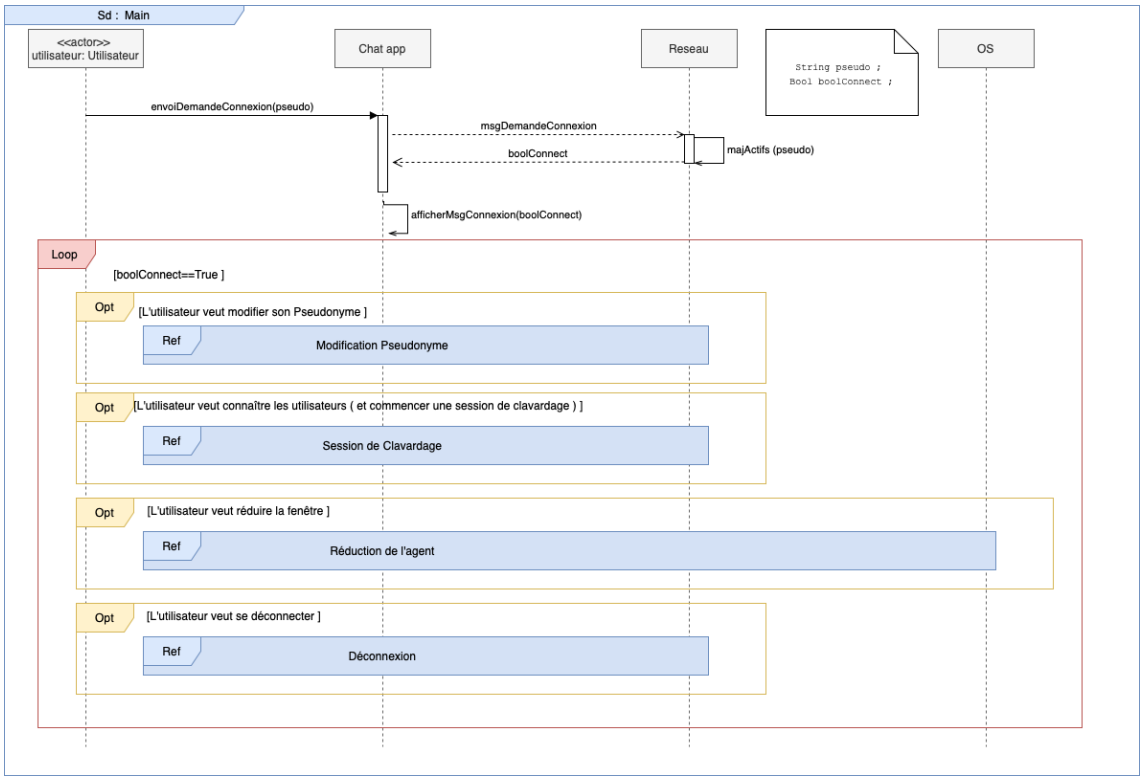

FIGURE 2 – Diagramme de séquence général

2.2 Phase de déconnexion

Lorsque l'utilisateur émet une demande de déconnexion, il est important de notifier tous les utilisateurs actifs du réseau du départ de celui-ci. Cette notification se fera par l'office d'un message UDP qui sera envoyé en broadcast sur le réseau.

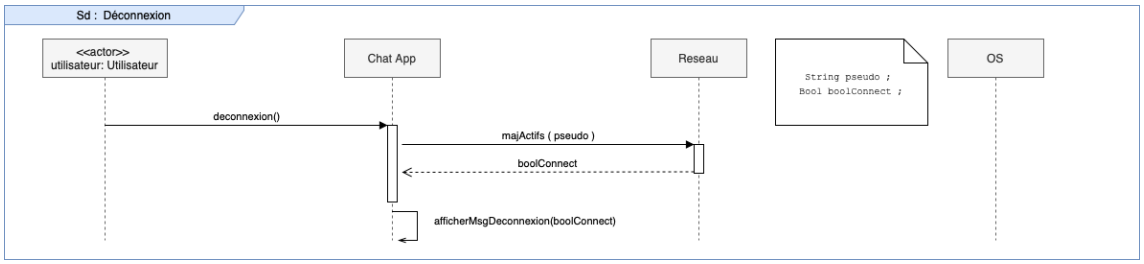

FIGURE 3 – Diagramme de séquence, phase de déconnexion

2.3 Phase de modification du pseudonyme

Chaque utilisateur possède un pseudonyme unique qu'il peut changer à tout moment. Nous mettons l'accent sur le mot "unicité" puisque une action de vérification sera effectuée par les pairs à chaque changement. Des messages UDP seront envoyés en broadcast à tous les utilisateurs connectés du réseau.

Au bout d'un laps de temps que nous fixerons, l'utilisateur pourra ou non prendre le pseudo demandé selon les différentes réponses reçues. Il devra alors notifier de nouveau les utilisateurs de la confirmation de ce changement.

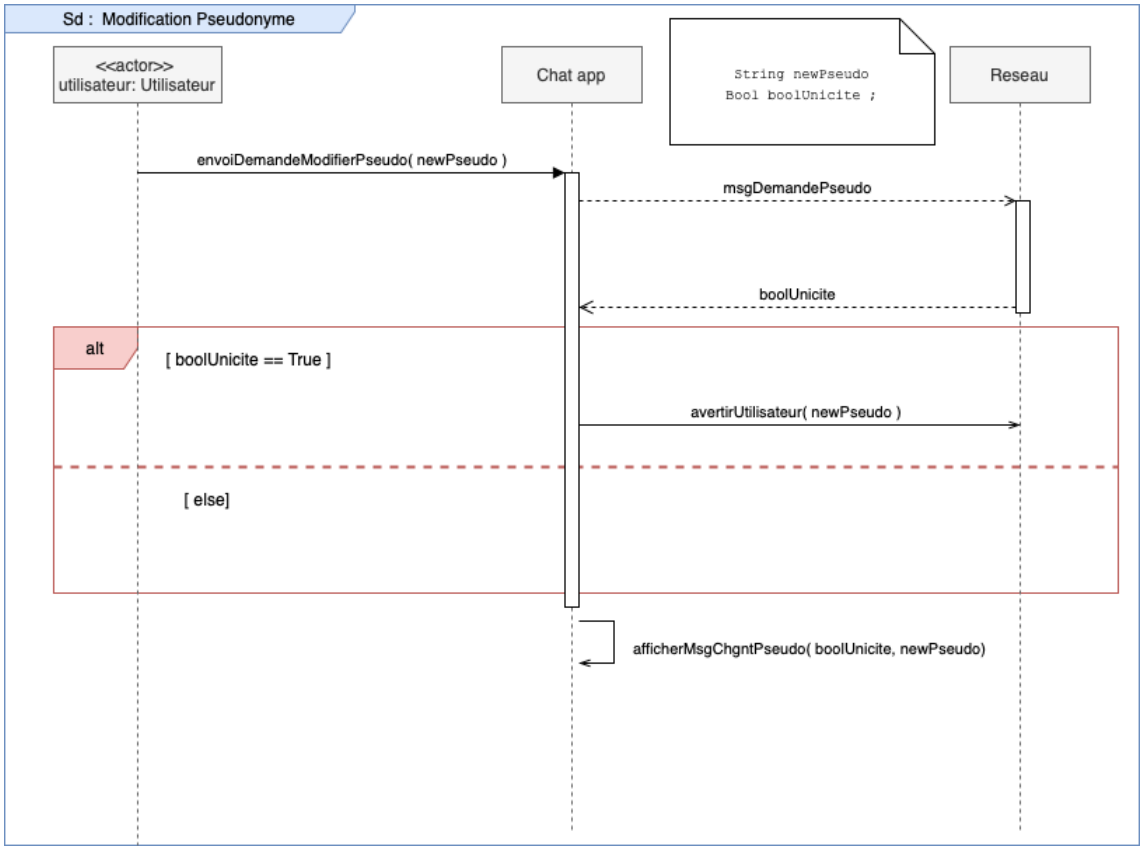

FIGURE 4 – Diagramme de séquence, phase de modification du pseudonyme

2.4 Phase de réduction de l'agent

Selon le système d'exploitation utilisé, il est possible ou non de réduire la fenêtre de chat. En connaissant le système d'exploitation utilisé, on peut donc réduire ou non la fenêtre.

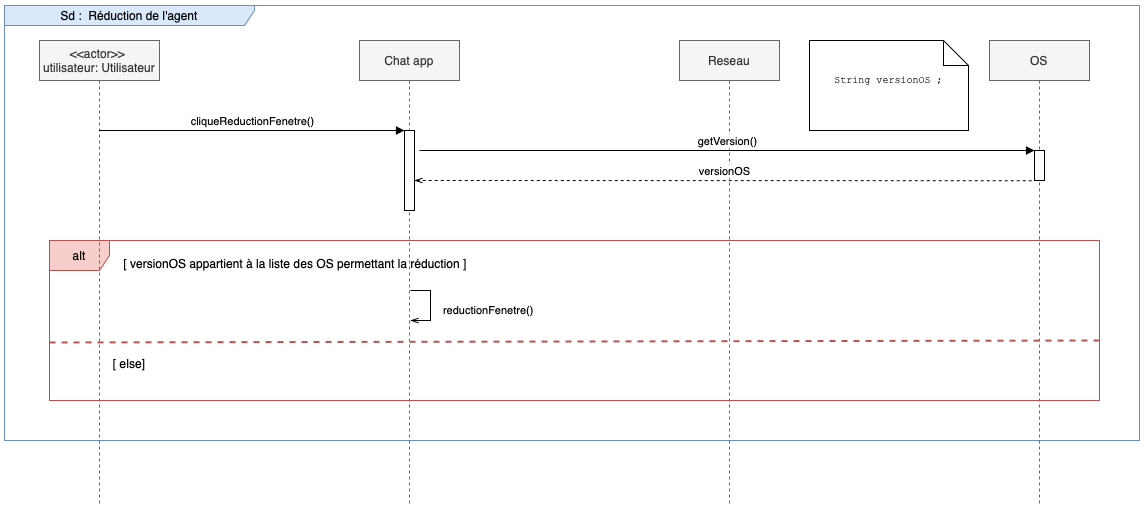

FIGURE 5 – Diagramme de séquence, phase de réduction de l'agent

2.5 Phase de clavardage

Une session de clavardage se découpe en plusieurs grandes étapes. L'utilisateur doit d'abord vérifier que l'utilisateur avec lequel il souhaite discuter est bien connecté en même temps que lui. En vérifiant la liste des utilisateurs actifs, il peut donc choisir de démarrer une session de clavardage. S'il choisit de démarrer une session de clavardage avec un utilisateur B, l'utilisateur A peut récupérer l'historique de conversation entre les utilisateurs A et B situé sur la BDD (= base de données). Une fois cela fait, une connexion TCP est créée entre les deux utilisateurs et l'utilisateur A peut envoyer des messages ou en recevoir si l'utilisateur B lui en envoie. Quand on envoie un message à un utilisateur, on envoie par la même occasion une requête à la base de donnée pour ajouter ce même message à l'historique. Quand l'utilisateur le souhaite, il peut décider de mettre fin à la session de clavardage. L'autre utilisateur sera prévenu de cette fin de session.

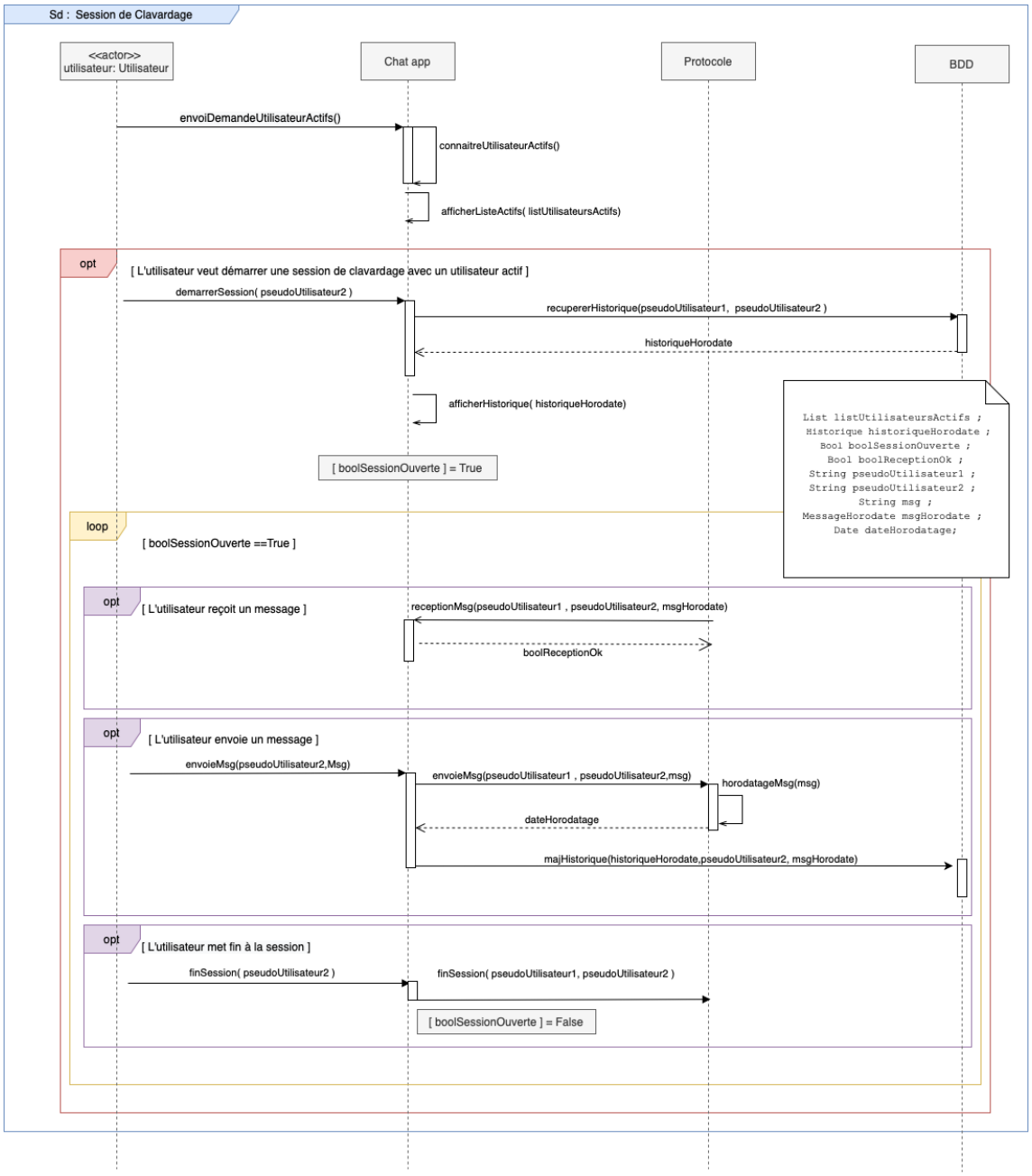

FIGURE 6 – Diagramme de séquence, phase de clavardage

3 Diagramme de classe

Les diagrammes de classes représentent les dépendances existantes entre chaque classe du projet. Nous avons opté pour une architecture MVC (= Modèle Vue Contrôleur). Nous avons dans cette section répertorié deux diagrammes de classe. Vous trouverez tout d'abord notre premier diagramme de classe, réalisé en amont du projet puis ensuite les diagrammes de classe correspondants à la version finale de notre projet.

3.1 Première Version

Cette première version du diagramme reprend l'architecture MVC. Le contrôleur appelé Chatapp communique avec la View pour afficher les différents éléments nécessaires comme les messages d'avertissement et les messages reçus ou pour permettre à l'utilisateur d'entrer des données comme

son pseudo, des messages et de valider ses choix. Le contrôleur communique aussi avec le Modèle pour récupérer les données nécessaires.

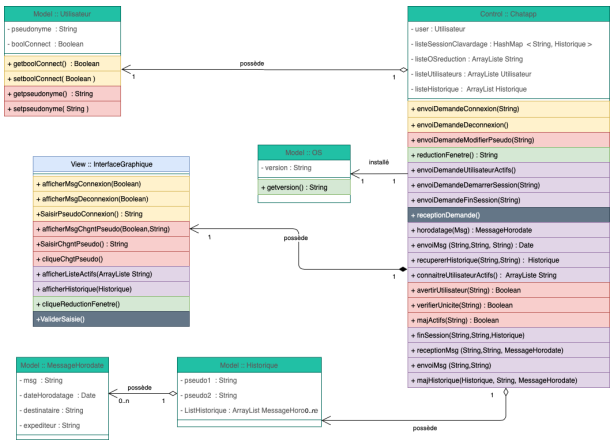

FIGURE 7 – Diagramme de classe, première version MVC

3.2 Version Finale

Dans cette version, qui représente toutes nos classes à la fin du projet, nous retrouvons quatre packages : Model, View, Controller et Protocol.

3.2.1 Model

Le package Model permet de répertorier toutes les données utilisées par l'application et le format que doivent avoir celles-ci. Ainsi, les classes MessageHorodate et Utilisateur indique quel format doivent avoir les messages que l'on envoie lors des sessions de clavardage et les instances représentant les différents utilisateurs connectés. Ces différents utilisateurs sont stockés dans la classe ListUtilisateurs et la classe DataBase permet de communiquer avec la base de données.

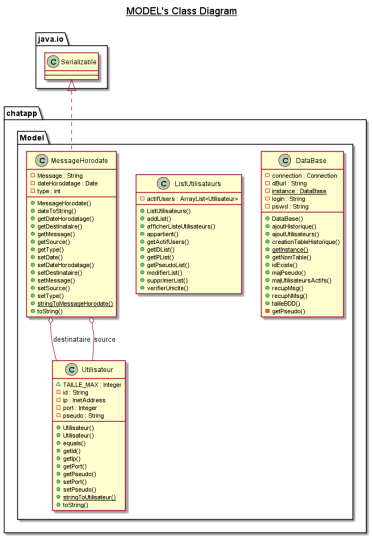

FIGURE 8 – Diagramme de classe, Model

3.2.2 View

La Vue ou View permet de gérer l'interface graphique de l'application et fait le lien entre l'utilisateur et le contrôleur. On y retrouve les différentes fenêtres de l'application, chacune représentée par une classe.

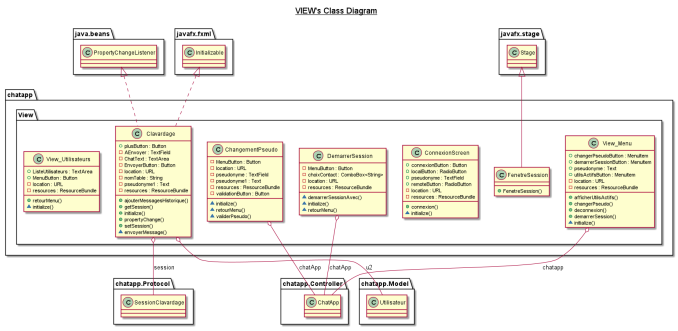

FIGURE 9 – Diagramme de classe, View

3.2.3 Controller

Le Contrôleur ou Controller gère les différentes actions que l'utilisateur souhaite effectuer. Pièce maîtresse de l'application, il permet notamment de gérer les demandes de connexion, déconnexion, changement de pseudo, démarrage de session de clavardage et de les envoyer au package protocole. Il va aussi se charger d'effectuer des modifications sur les données enregistrées dans les classes du Model.

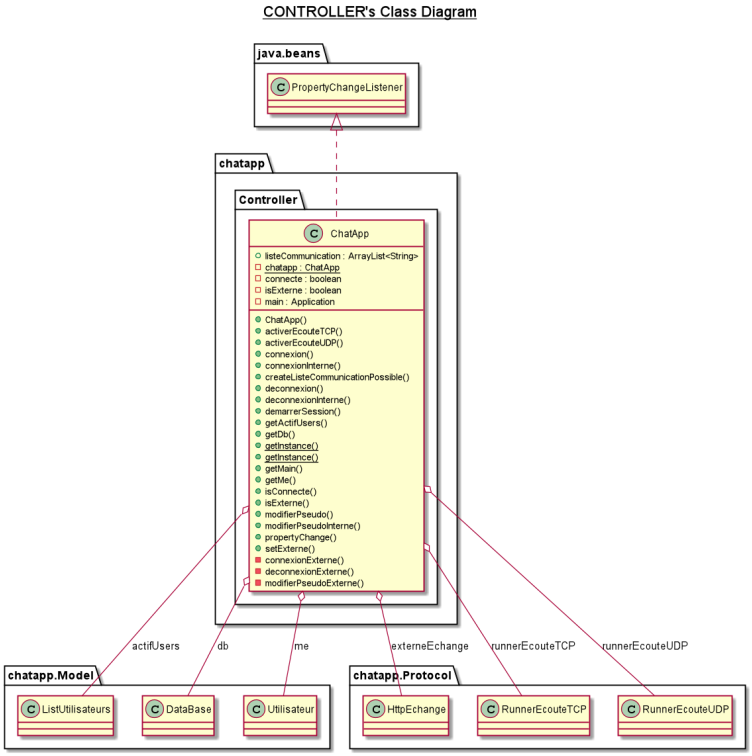

FIGURE 10 – Diagramme de classe, Controller

3.2.4 Protocol

Le package Protocol regroupe tous les threads et méthodes nécessaires à l'envoi et la réception des messages TCP et UDP. C'est donc ce package qui va envoyer toutes les requêtes en broadcast sur le réseau et établir des connexions TCP avec les autres utilisateurs.

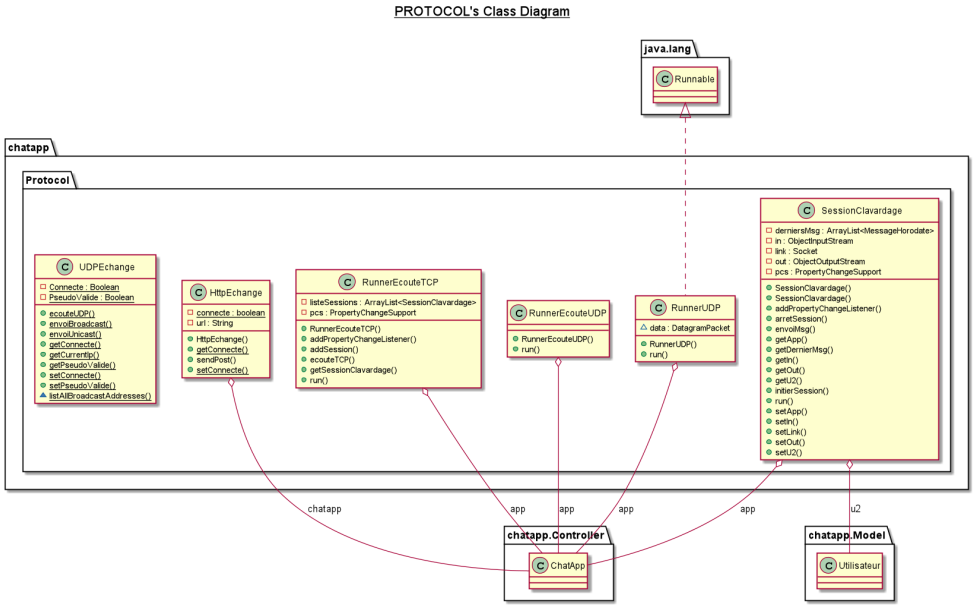

FIGURE 11 – Diagramme de classe, Protocol

4 Diagramme de structure composite

Le diagramme de structure composite permet de visualiser les différentes interactions entre les éléments de l'application. Ces interactions peuvent comprendre des envois de données ou des requêtes spécifiques.

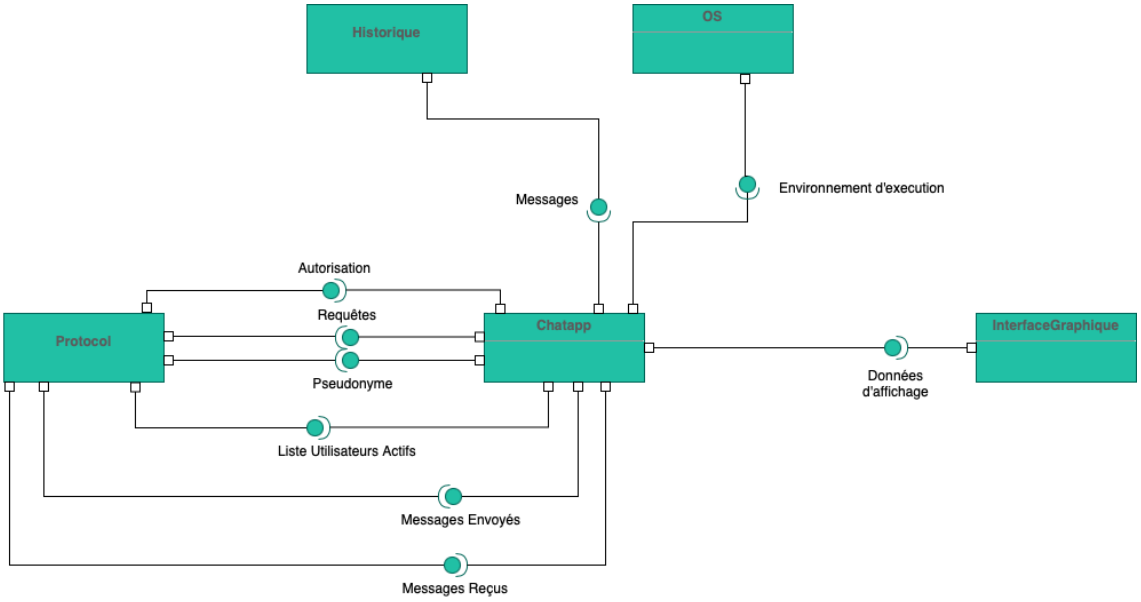

FIGURE 12 – Diagramme de structure composite

5 Diagramme des machines à état

Le statechart permet de visualiser les différents états par lesquels peut transiter l'application. Il nous permet ainsi de comprendre comment nous allons devoir articuler les différentes étapes du cycle de vie de l'application et d'avoir une première idée de la mise en place de l'interface graphique. Nous avons découpé le diagramme en quatre parties pour permettre une meilleure lisibilité aux lecteurs.

5.1 Machine à état générale

Durant cette étape, l'application est par défaut dans un état dit "Déconnecté". Après une tentative de connexion, si celle-ci s'avère concluante, nous entrons dans un état dit "Connecté" permettant d'accéder au menu principal et d'effectuer davantage d'actions. Bien entendu, si la tentative de connexion échoue, nous restons dans l'état "Déconnecté.

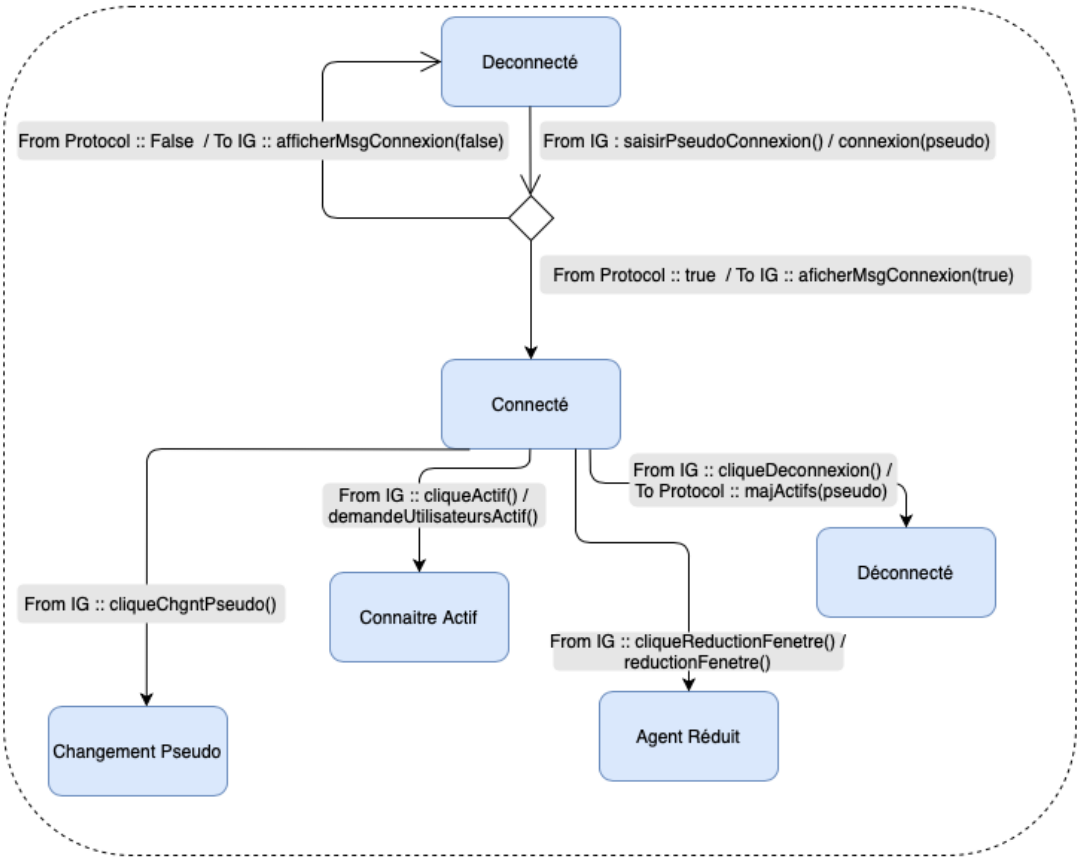

FIGURE 13 – Machine à état générale

5.2 Machine à état : Phase de modification de pseudo

Cette étape commence par l'état "Changement Pseudo" qui permet à l'utilisateur d'entrer un nouveau pseudonyme. Si sa demande est acceptée par ses pairs, on retourne dans l'état "Connecté" décrit précédemment. Sinon, l'utilisateur est prévenue par un message d'alerte et il peut tenter d'entrer un nouveau pseudo.

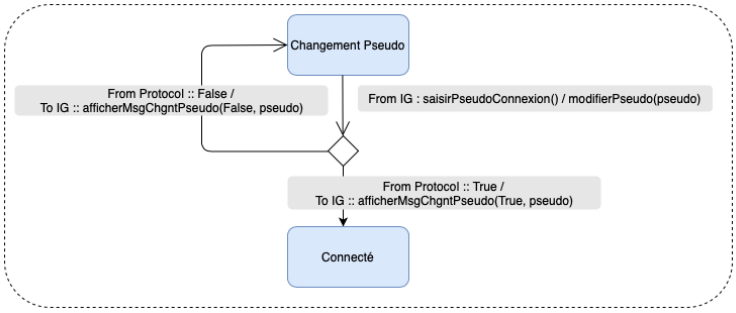

FIGURE 14 – Machine à état, Phase de modification de pseudo

5.3 Machine à état : Phase de réduction de l'agent

Si l'agent est réduit, il est possible en cliquant sur le bouton approprié de rouvrir l'agent et donc de retomber dans l'état "Connecté".

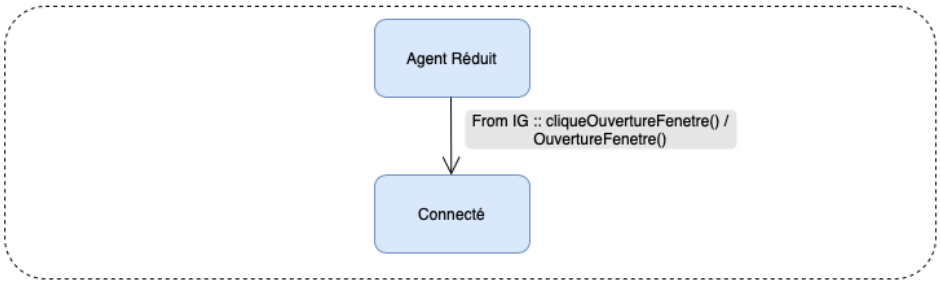

FIGURE 15 – Machine à état, Phase de réduction de l'agent

5.4 Machine à état : Phase de clavardage

En se rendant dans l'état "Connaître Actif", il est possible de choisir un utilisateur avec lequel l'on souhaite communiquer. On récupère ensuite l'historique des messages qui, une fois affiché, nous permet d'échanger des messages avec un autre utilisateur. En mettant fin à la session, on retombe dans l'état "Connaître Actif". En cliquant sur le bouton "Home", on peut aussi repasser à l'état "Connecté" décrit précédemment.

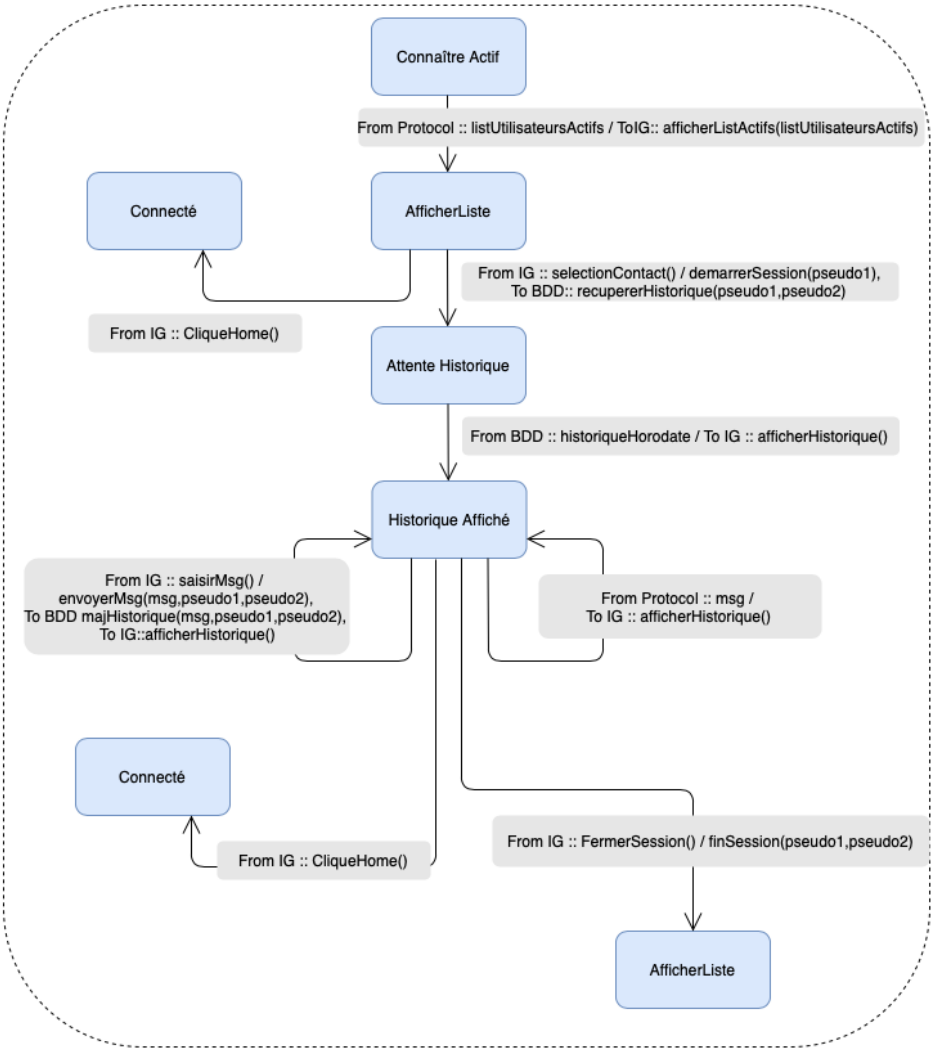

FIGURE 16 – Machine à état, Phase de clavardage

Conclusion

Les divers diagrammes que nous avons dressés nous ont permis de mieux cadrer notre projet et de réaliser quels allaient être les points délicats à traiter avec minutie. En effet, lors de la découverte du sujet, nous avons négligé beaucoup de points qui pourtant nécessitaient une inspection approfondie. Ce cas de figure s'est aussi présenté à la fin de notre conception. En effet, certaines idées que nous avons eues lors de la conception se sont avérées être trop simplistes (car nous avions fait de grossières estimations de la difficulté de leur mise en oeuvre) ou même trop recherchées (car il existait une alternative plus simple et moins coûteuse en temps et en lignes de code). C'est la raison pour laquelle notre diagramme de classes comporte deux versions et que la version finale diverge autant de la version initiale. Nous avons beaucoup appris grâce à l'implémentation et avons compris les raccourcis que nous avions faits lors de la conception. Cependant, notre conception n'était heureusement pas trop éloignée de la réalité, ce qui a grandement facilité l'implémentation.