INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE | DGEI - 4IR

Rapport Projet

Conception et Programmation Orientée Objet

Système de clavardage distribué Multi-utilisateur temps réel

28 Janvier 2022

Étudiants :

Alae NOURDINE nourdine@etud.insa-toulouse.fr

Ghita Maher maher@etud.insa-toulouse.fr

Enseignants :

Sami YANGUI

Nawal GUERMOUCHE

Sommaire

Introduction

Dans le cadre de notre formation en 4ème année IR, nous avons participé au projet de création d'un système de Clavardage. Dans ce projet de Programmation Orientée Objet, nous avions pour objectif la création d'une application de clavardage fonctionnelle respectant un cahier des charges établissant un ensemble de fonctionnalités.

Lors du processus de création de toute application, et même de tout projet de manière générale, deux étapes au moins sont nécessaires : la conception et l'implémentation.

Le rapport suivant rassemble ainsi une présentation de notre projet, notre conception, nos choix d'implémentation mais également un manuel d'utilisation permettant la prise en main notre ChatBox.

I. Conception

https://github.com/AlaeNourdine/ChatSystem/tree/main/Conception

a. Use Case

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.

b. Diagrammes de classes

Nous nous basons sur l'architecture MVC. Le contrôleur appelé Controller 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 Model pour récupérer les données nécessaires et avec Network pour gérer les connexions, déconnexions et changements de pseudo.

c. Diagrammes de séquence

Les différents diagrammes de séquence qu'on a décidé d'effectuer 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. Il est important de notifier tous les utilisateurs actifs du réseau en connexion, déconnexion et en changeant le pseudo.

- ***Connexion :*** L'utilisateur émet une demande de connexion
- ***Send Message*** *:* Envoi des messages à un autre utilisateur
- ***Receive Message :*** Réception des messages depuis un autre utilisateur
- ***Nickname Change*** *:* Chaque utilisateur possède un pseudonyme unique qu'il peut changer à tout moment.

II. Architecture du Projet

a. Packages

Pour l'implémentation de notre application, nous avons choisi de suivre le modèle MVC traditionnel (Model View Controller) auquel nous avons ajouté le package 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 Controller.

Ces attributs correspondent à :

- L'utilisateur connecté sur le système
- La liste des utilisateurs actifs
- L'instance de la classe ControllerChat permettant à l'user de gérer connexion, déconnexion et changement de pseudo
- La base de données de l'utilisateur (historique des messages)
- L'instance de la classe UDPReceive modélisant le côté serveur en UDP

b. Echange TCP

Cependant une fois la liste des utilisateurs actifs 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

c. Broadcast

En tentant de se connecter, l'utilisateur entre un pseudonyme. On identifie l'utilisateur par User contenant son pseudonyme, son adresse IP et son numéro de port qui seront envoyés en broadcast sur le réseau. Chaque utilisateur répond si le pseudo de l'émetteur est identique au sien ou pas.

Un utilisateur n'a pas donc la possibilité de se connecter que si chacun des autres utilisateurs sur le réseau répond et ne possède pas déjà le pseudo qu'il souhaite utiliser. En recevant les réponses des autres utilisateurs, le nouvel arrivant les ajoutera par la même occasion à sa liste des utilisateurs actifs.

Notre broadcast se fait en UDP. En effet, ce mode de transport est le plus adapté dans ce cas parce qu'il ne possède aucune contrainte de qualité de service.

III. Manuel Utilisateur

a. Installation

Afin de pouvoir utiliser notre ChatBox, il faut tout d'abord récupérer le logiciel. Pour ça, il vous faut cloner le répertoire git suivant:

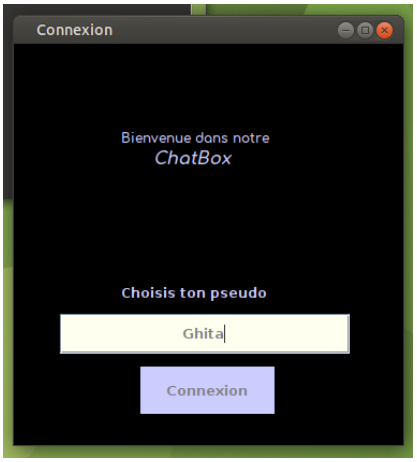
git clone https://github.com/AlaeNourdine/ChatSystem

Vous pouvez alors lancer l'exécutable "ChatBox" :

- Sur Linux, à l'aide de la commande : java –jar ChatBox.jar
- Sur Windows cliquez sur le fichier jar exécutable ChatBox.jar

b. Page de connexion

Vous serez donc accueilli par notre interface de connexion. Celle-ci vous permet de choisir votre pseudo (en moins de 12 caractères sinon "Pseudo trop long" s'affiche). Une fois votre pseudo choisi, vous pouvez alors vous connecter en cliquant sur "Connexion".

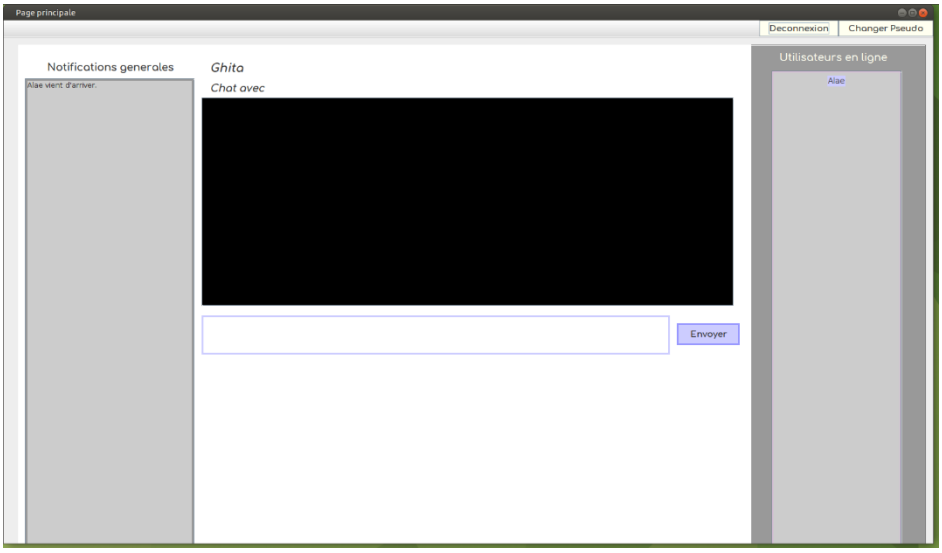


En revanche, si le message "*Pseudo déjà utilisé. Veuillez en choisir un autre*." apparaît, c'est qu'un autre utilisateur est déjà connecté avec le même pseudo devez en choisir un nouveau. Si aucun message n'apparaît, votre pseudo est alors validé par le système et vous entrerez dans l'interface d'accueil.

c. Page principale (General)

L'interface générale vous permet de voir l'ensemble des utilisateurs actifs sur le réseau à droite de l'écran.

Pour lancer une conversation avec un des utilisateurs, il vous faut - Sélectionner le destinataire dans la liste - Taper votre message - L'envoyer avec entrée ou en appuyant sur le bouton *Envoyer.*

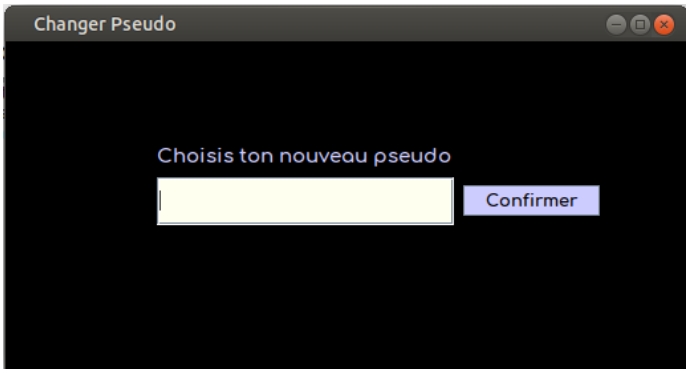


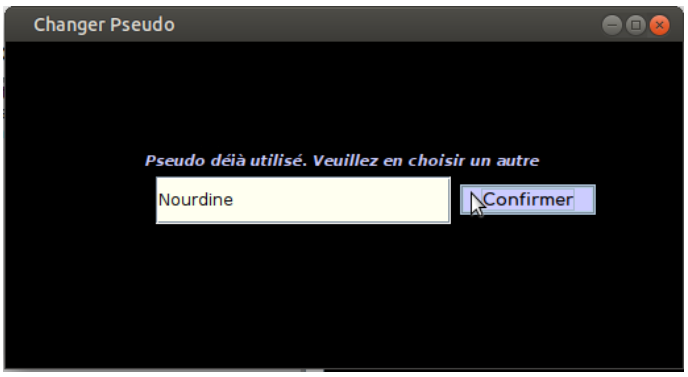
Lorsque vous appuyez sur le pseudo d'un de vos contacts, vous lancez une conversation. Cela entraîne l'apparition du pseudo de l'utilisateur avec qui vous discutez à côté du label *Chat avec*. Cela vous permet de savoir à qui vous parlez à tout instant.

A gauche de l'écran, vous recevrez également toutes les notifications de connexions, déconnexions, changements de pseudos et des messages reçus.

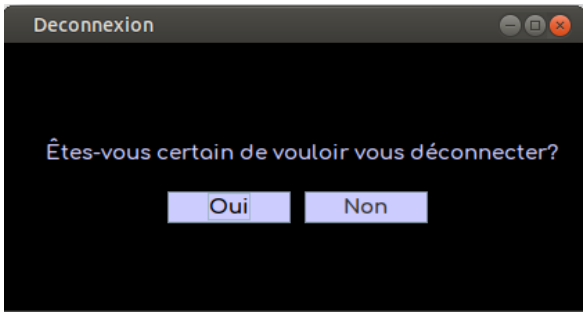
En haut de la page, nous avons deux boutons :

- *Changement de pseudo :* permet à l'utilisateur de changer son pseudo initialement choisi. Les conditions restent les mêmes.





- *<u>Déconnexion</u> :* Il propose à l'utilisateur de confirmer sa demande de déconnexion. En appuyant sur le bouton *Non*, la fenêtre se ferme, et rien ne change sur l'interface d'accueil. 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 *General*, l'utilisateur se déconnecte automatiquement.



IV. Choix d'implémentation

a. Base de données

Afin de gérer les échanges de messages et en particulier la gestion de l'historique, nous avons décidé d'utiliser une base de données locale sur machine locale. Nous avons donc posé l'hypothèse que chaque utilisateur se connecte toujours avec le même hôte.

L'implémentation de cette base de données a été faite sur le serveur MySQL déployé à l'INSA. Celle-ci contient les tables des conversations. Chaque table est nommée *Chatwith_'@IPdest'*, avec *@IPdest* l'adresse du destinataire. Cette adresse IP est donc unique et propre à chaque utilisateur d'après l'hypothèse émise ci-dessus.

Chaque table contient 3 colonnes :

- *message* : text qui représente les messages échangés
- *time* : text qui représente l'horodatage des messages
- *émetteur* : int qui permet de savoir si le message a été envoyé ou reçu. Le message a été émis par l'hôte => 0, le message a été reçu par l'hôte=> 1

b. Swing

Pour créer 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.

c. 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 ChatBox en un projet Maven.

Grâce au plugin Apache Maven Assembly Plugin, le format nous permet ainsi d'obtenir un fichier exécutable, unique, qui contient toutes les librairies et les dépendances du projet. Ceci pour une expérience utilisateur facilitée et plus intuitive.

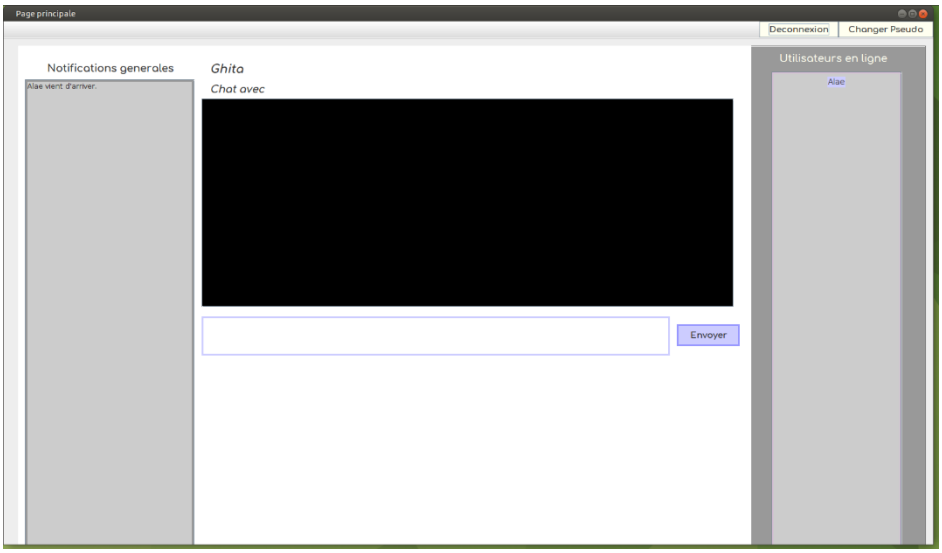
V. Test réalisés

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.

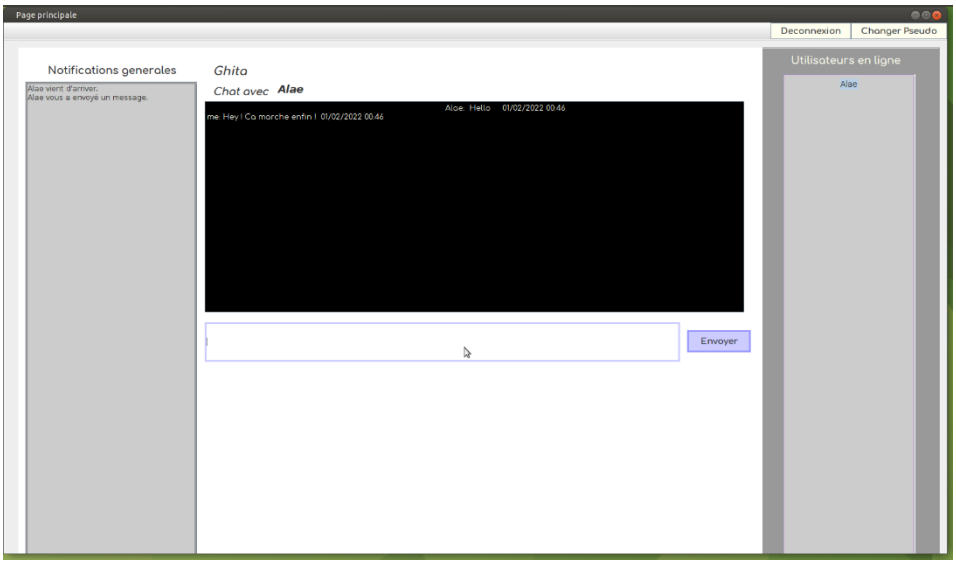
La seconde partie des tests constitue ainsi l'ensemble des tests de l'application entre plusieurs ordinateurs. Cette deuxième partie nous a permis d'améliorer le projet mais aussi de vérifier le respect du cahier des charges. Ainsi, cette partie du rapport se présente comme la checklist de l'ensemble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

- *Unicité du pseudo choisi lors d'une première connexion (cas d'unicité en changeant de pseudo traité plus haut)*

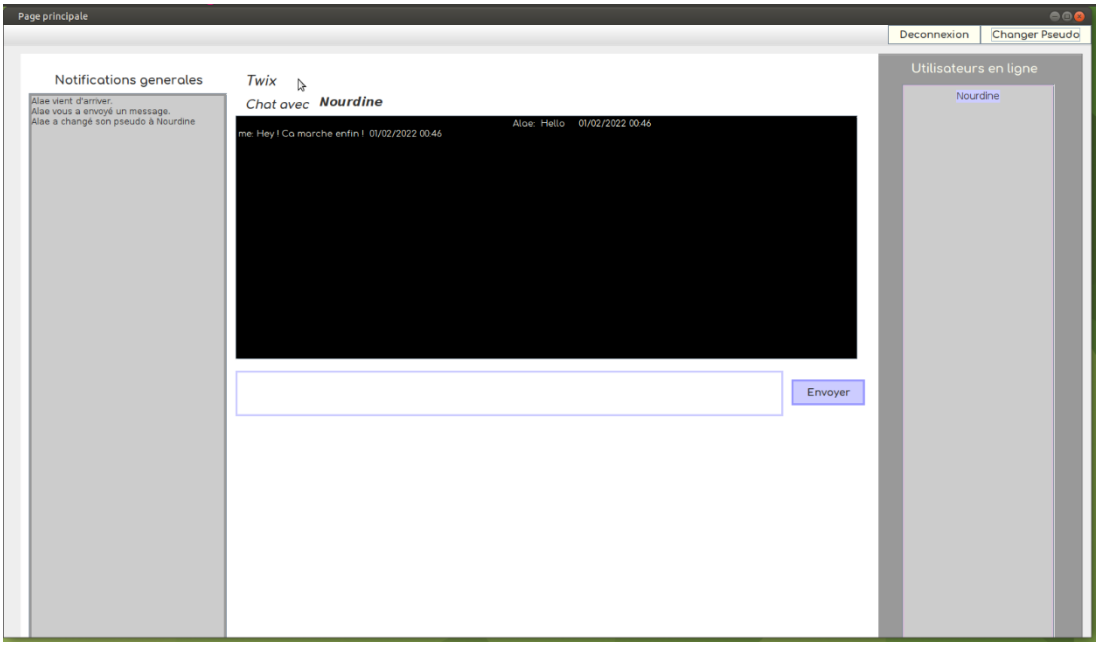
- *Connexion :* L'utilisateur ***Ghita*** se connecte en premier, on remarque la notification apparaitre lorsque l'utilisateur ***Alae*** se connecte après. Tous les utilisateurs actifs reçoivent une notification pareille.



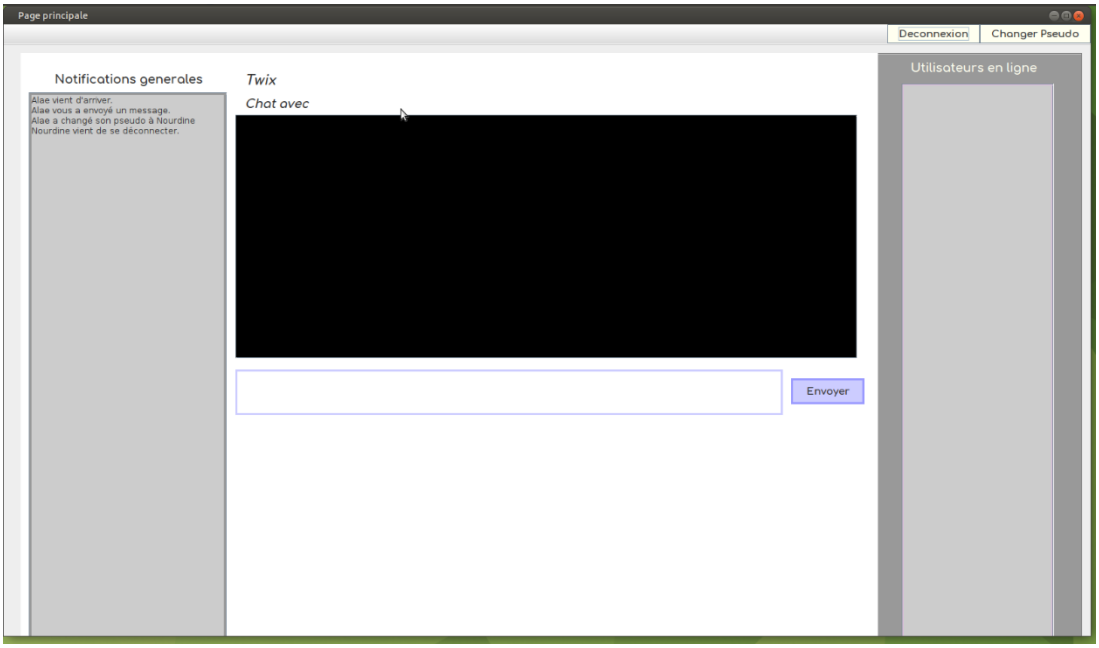
- *Envoi et réception des messages :* ***Ghita*** reçoit une notification disant que ***Alae*** a envoyé un message. Elle sélectionne ***Alae*** depuis la liste des utilisateurs actifs et leur conversation apparait. Elle peut envoyer et recevoir des messages sans problèmes.



- *Changement de pseudo de **Alae** en **Nourdine*** : Tous les utilisateurs actifs dont ***Ghita*** reçoivent la notification disant que ***Alae*** a changé son pseudo à ***Nourdine.***



- *Déconnexion de **Nourdine : Nourdine*** se déconnecte. Son pseudo est retiré de la lis te des utilisateurs actifs de Ghita ainsi que celle de tous les utilisateurs actifs. Une notification suit sa déconnexion pour informer tout le monde que Nourdine s'est déconnecté.



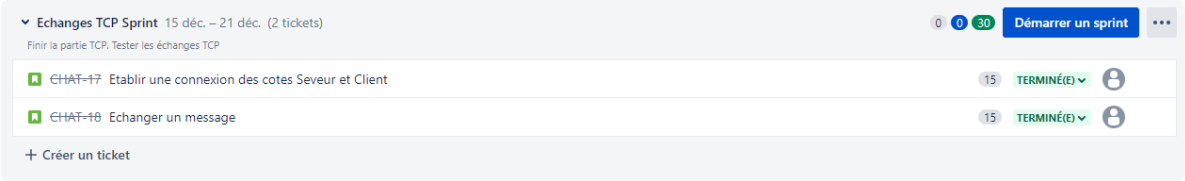
VI. Gestion de Projet

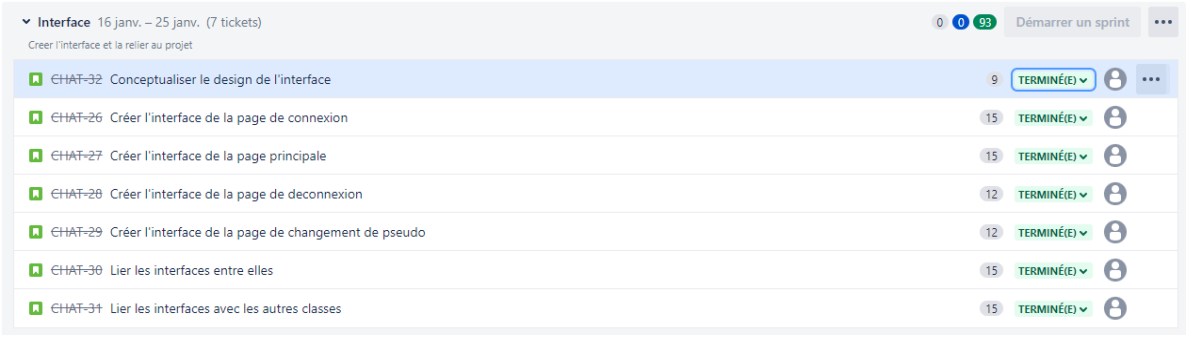
a. Jira

Pour mieux nous organiser sur ce projet, nous avons utilisé la méthode Agile vue en cours de gestion de projet. Nous avons défini les différentes tâches à réaliser, autrement dit des User Stories puis nous les avons regroupées dans des Sprint Backlog en utilisant le site Alassian et classées par priorité grâce à la méthode Poker, 15 étant la priorité maximale d'une Story.

Malgré cette organisation et cette méthodologie de gestion de projet, nous nous sommes retrouvés par moment en retard à cause d'un non-respect du délai de certaines tâches qui nous ont pris plus de temps que prévu à cause des erreurs et/ou des bugs lors d'implémentation du code.

Ci-dessous un exemple de deux de nos Sprint qui représente différentes User Story. Le premier sprint ci-dessous représente le sprint pour la partie échanges TCP et le 2ème sprint représente la création d'interface.





Comme nous pouvons le voir, les User Stories ont différentes priorités en fonction de ce qui nous semblait plus important, notamment la partie Échange TCP dont toutes les User Stories sont fixées à une haute priorité.

Nous pouvons voir aussi en plus de cela un délai différent pour les Sprints. Le sprint Échanges TCP est un exemple de sprint où nous n'avions pas respecté le délai suite à une sous-estimation du temps que l'implémentation du code ainsi que le débogage des problèmes pouvaient prendre. C'est pour cela que nous avons modifié notre Backlog au fur et à mesure pour l'adapter aux différentes deadline, et que nous avons donc consacré 10 jours pour la création de l'interface, afin d'être sûrs de respecter le délai et être dans les temps pour rendre le projet.

b. Github

Pour garantir une aisance et une facilité de travail sur ce projet en binôme, nous avons utilisé les services de GitHub. L'avantage de GitHub est de nous faciliter la résolution des éventuelles erreurs qui se produisent lors de l'implémentation de notre code, puisque tout changement dans chaque version est enregistré et mentionné. De plus, le dépôt des différentes versions et changement du code ainsi que la récupération de ces changements fait par l'un d'entre nous est très simple à gérer grâce aux commandes pull, commit et push.

Pour ce projet, nous avons utilisé une seule branche qui est la branche main. L'utilisation d'une seule branche n'a pas été un problème dans notre cas puisque nous nous sommes réparties les tâches en fonction des packages à implémenter. Quant aux commandes push, pull et commit, nous utilisons sur nos ordinateurs respectifs l'application GitHub desktop qui est très simple à manipuler et très pratique.

Conclusion

Ce projet nous a ainsi permis de mettre en œuvre l'application de clavardage The ChatBOX. 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 parfaitement fonctionnelle, cependant l'application The ChatBOX remplit notre objectif principal de respect du cahier des charges.

Pour conclure sur notre projet, nous sommes heureux d'avoir pu avoir la chance de travailler sur un tel projet dans le cadre de nos études. Cela a été pour nous l'occasion de travailler de manière concrète sur l'intégralité d'une application que nous avons nous-mêmes conçue.

Nous sommes tout de même satisfaits d'avoir expérimenté un tel projet qui nous a beaucoup apporté en termes de professionnalisme et nous tenons à remercier nos tuteurs qui nous ont accompagnés durant ce projet sur les parties conception et implémentation respectivement.