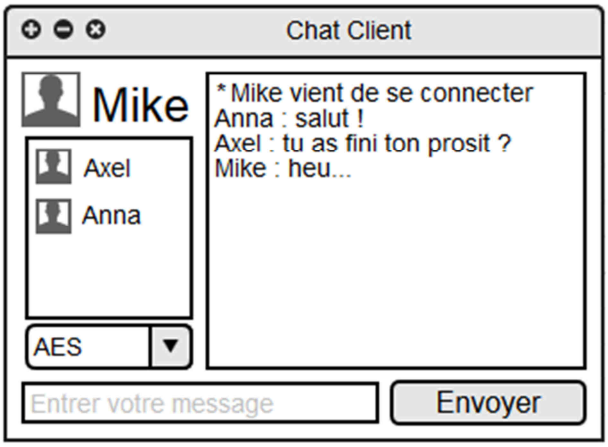
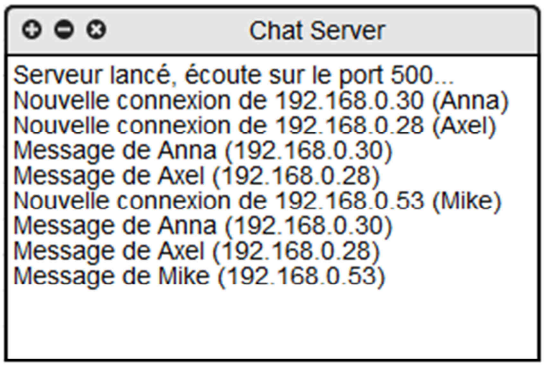
TEXTE DU PROSIT

Cela fait maintenant 2 ans que vous avez monté votre société à la sortie de l'Exia. Votre activité principale consiste à développer des solutions informatiques personnalisées pour des clients divers.

Un jour, vous êtes contacté par un nouveau client. Il vous demande de réaliser une application très simple et robuste qui permettrait d'échanger des messages texte entre plusieurs machines. En fait, vous comprenez qu'il s'agit simplement d'une application de chat instantané, avec un serveur et plusieurs clients. L'entreprise souhaite économiser au maximum sur les coûts de développement, aussi vous demande-t-elle de réaliser cette application en Java.

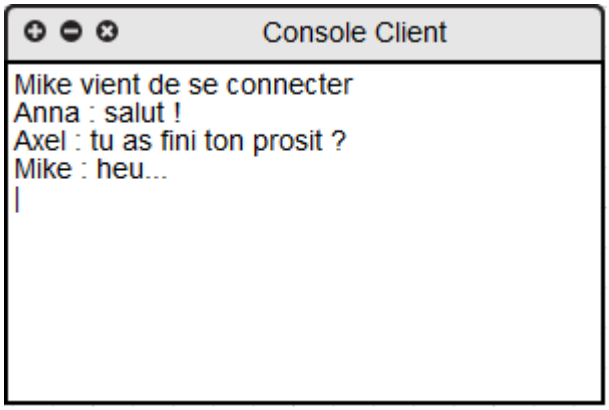
Le client vous esquisse une maquette rapidement :





L'interface est très simple, et vous rappelle d'ailleurs le bon vieux IRC que vous aviez connu étant jeune. A la différence que là, il n'y a qu'un seul channel pour discuter…

En pensant que vous n'avez que deux jours pour réaliser l'IHM et les sockets, vous prenez soudainement une teinte vert pâle. Votre client vous rassure : vous pouvez commencer par réaliser le client en ligne de commande, quitte à rajouter une IHM par la suite…



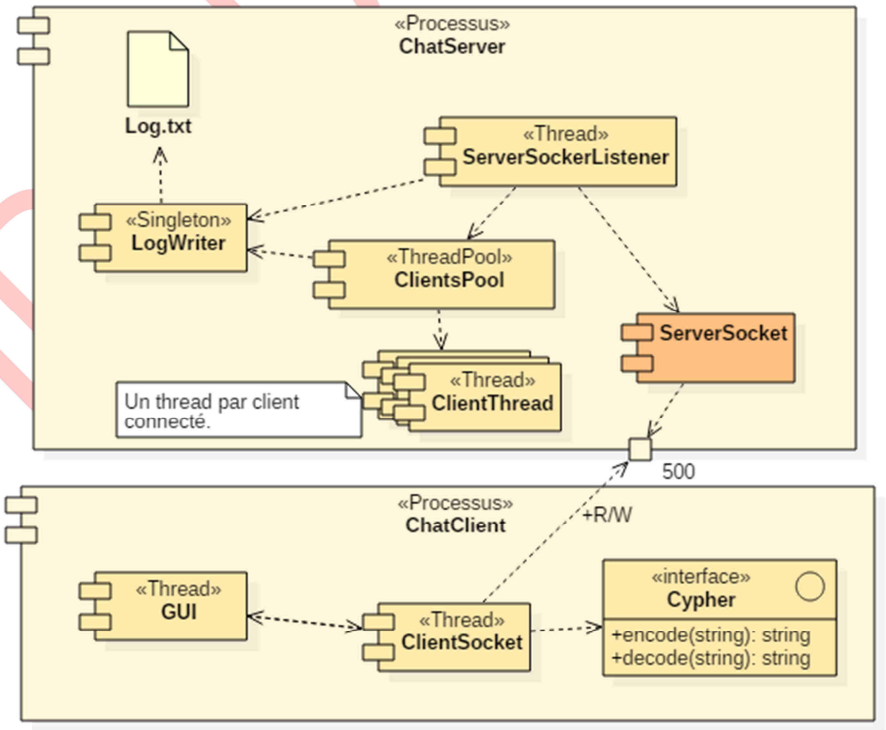
Votre client vous fait comprendre également qu'il souhaiterait dans un second temps permettre le chiffrement des communications échangées. Pour commencer, du simple Base64 suffira pour les tests, mais le client souhaiterait pouvoir en changer plus tard. Cet aspect aussi est facultatif pour une première version, mais impressionnerait votre client…

Les processus étant sur des machines distantes et interconnectés via un réseau TCP, vous pensez immédiatement au Socket. Mais vous n'avez encore jamais travaillé directement sur cette couche… Quelques petites révisions s'imposent…

Vous vous souvenez qu'il faut un SocketServer pour écouter sur un port donné, et que ce travail monopolise à lui seul un thread. Vous vous souvenez également que lorsqu'un client se connecte sur le serveur, un autre Socket est ouvert, et que là aussi il faut un thread pour gérer les entrées/sorties. Donc s'il y a plusieurs clients, il y aura plusieurs threads… hum, ça sent la pool vous dites-vous à juste titre...

On vous demande également que le serveur soit à même de logger tous les événements (connexion et déconnexion d'un client, ou envoi d'un message) et d'écrire ces logs dans un fichier texte.

Vous aurez alors plusieurs threads, souhaitant accéder à une ressource unique… hum, il vous vient immédiatement un problème à l'esprit : si les différents threads des clients écrivent en même temps, cela risque de corrompre le fichier de log. Il serait bien de synchroniser tout cela… Vous vous souvenez que certains mécanismes de communication inter-processus (IPC) permettent justement de synchroniser deux threads, mais votre mémoire est un peu rouillée depuis le temps.



Votre expert technique (qui s'y connait en UML, et oui il n'est pas expert pour rien) dresse rapidement un diagramme de composant. « *Voilà* ! clame-t-il une fois terminé : *il ne reste plus qu'à réfléchir aux accès concurrents dans l'application et placer les mécanismes d'IPC. A vous de jouer.* » Puis il s'en va d'un aire digne le pas majestueux, comme seul les experts techniques savent le faire…

Vous décidez pour commencer de faire l'inventaire des IPC, et de les classer proprement dans un tableau en fonction de leurs capacités. Une fois ce travail réalisé, vous prévoyez également de compléter la modélisation logicielle de l'application.

Le client reviendra dans 2 jours pour récupérer une première version de l'application. Vous avez du pain sur la planche, il va falloir s'y mettre d'arrache-pied ! Les pauses cigarettes vont peut-être se faire rares….

RESSOURCES

Comprendre les processus

https://openclassrooms.com/courses/la-programmation-systeme-en-c-sous-unix/les-processus-1

Les IPC en environnement Windows

https://msdn.microsoft.com/en-us/library/windows/desktop/aa365574%28v=vs.85%29.aspx

Les IPC en environnement Linux

http://www.tldp.org/LDP/lpg/node7.html
beej.us/guide/bgipc/output/html/multipage/index.html

La gestion des threads en Java

http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html

https://en.wikipedia.org/wiki/Thread_pool_pattern

http://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-and-threadpoolexecutor.html

Socket en Java

https://openclassrooms.com/courses/introduction-aux-sockets-1

http://docs.oracle.com/javase/7/docs/api/java/net/Socket.html

https://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html

Bibliographie

Mémento UML 2.4, de Eyrolles

Thinking in Java, Bruce Eckel, 4ème edition