INF3405 – Réseaux informatiques

TP1
Traitement d'images par réseau

Groupe 3

1847812 – William VALIQUETTE

1794431 – Nicole JOYAL

Soumis à : Esther Guerrier

Hiver 2020

14/02/2020

Table des matières

Introduction

Ce travail pratique nous propose d'effectuer une simulation d'un logiciel client-serveur. Le logiciel doit créer et utiliser le serveur pour traiter et modifier une image que le client lui envoie pour la rendre en négatif. Le tout fonctionne avec des sockets qui se servent directement de l'adresse IP et du port que l'utilisateur du serveur lui donne et le client fonctionne sur le même principe. Les objectifs de ce TP sont de se familiariser avec les connaissances de base des adresses IP, du fonctionnement des *sockets* et de la transmission de l'information entre un client et un serveur. Pour accomplir cela, nous devons implémenter deux programmes qui agiront respectivement comme serveur et client, et qui communiqueront entre eux à travers une connexion *socket*.

Présentation de nos travaux

Tout d'abord, parlons de l'interface. Nous avons fait une interface en ligne de commande avec encapsulée dans une classe appelée Interface qui allait servir au client et au serveur. Cette classe se sert d'un objet Scanner pour lire ce que l'utilisateur écrit dans la console.

Ensuite, afin d'authentifier un utilisateur, le serveur utilise la classe Credentials. Cette classe sert à valider le nom d'utilisateur et le mot de passe entrés contre une base de données. La base de données est un fichier JSON nommé users.json, qui contient une liste de pairs de String, chaque pair représente un nom d'utilisateur avec son mot de passe correspondant. Pour lire et écrire dans un fichier JSON, il a fallu se servir d'une librairie externe. Nous nous sommes servi de la librairie *json-simple*, qui contient les structures nécessaires pour traiter les objets des fichiers JSON.

Du côté serveur, nous avons implémenté une classe intermédiaire qui étend le type *Thread*, pour gérer la communication avec les clients. Cette classe, *ClientHandler*, se charge de l'ouverture et de la fermeture des points de communication du côté du serveur; *DataInputStream* et *DataOutputStream*. Plus particulièrement, c'est elle qui fait la validation contre la base de données comme mentionnée ci-dessus. Elle utilise les objets mentionnés pour transmettre les informations de validation et les images à traiter.

Finalement, les classes *Client* et *Server* ont servi comme classes d'exécution principale (*main*). Autrement dit, ces classes contiennent les programmes exécutables du

client et du serveur. Elles se servent des classes énumérées ci-dessus pour accomplir la tâche demandée.

Difficultés rencontrées

Nous avons eu beaucoup de difficulté à faire fonctionner notre environnement, ainsi qu'ajouter des librairies externes sur nos ordinateurs personnels, comme *json-simple*. En effet, cela nous a pris plusieurs heures et même après cela un de nos ordinateurs refusait de compiler avec java. Il a donc fallu utiliser la console intégrée dans notre IDE (Eclipse) pour faire fonctionner le tout.

Afin de cacher les informations sensibles de l'utilisateur, notamment son mot de passe, nous nous sommes servis de la librairie *Console* pour lire le mot de passe entré. Cependant, cette librairie causait une erreur dans la console intégrée d'Eclipse. Cela nous a pris du temps de débogage pour en venir à une solution. La solution trouvée était d'exécuter le code du serveur dans Eclipse et le code du client dans un terminal Windows, car le terminal de l'ordinateur ne renvoie pas d'erreur. Sinon, pour nous servir entièrement des consoles intégrées d'Eclipse, nous avons pu remplacer l'utilisation de la librairie *Console* par la libraire *Scanner*. Par contre, cette solution rend le mot de passe visible et, en pratique, compromet la sécurité de l'utilisateur.

Ensuite, pour envoyer les données entre le client et le serveur, nous avons utilisé les classes java *DataInputStream* et *DataOutputStream*. Avec les fonctions *write/readUTF*, transmettre des données en *String* s'avérait plutôt simple. Cependant, pour envoyer une image il a fallu écrire et lire l'image octet par octet. La difficulté se situait surtout du côté serveur, essayer de trouver une manière d'arrêter la lecture (sinon nous étions pris dans une boucle infinie de lecture). Une solution était d'envoyer un octet nul, par contre cette stratégie causait une erreur du côté client. La meilleure solution a été de lire la taille en octets de l'image et de l'envoyer avant de lire les octets pour limiter le nombre d'octets lus et ainsi éviter une boucle infinie.

Critiques et améliorations

Dans l'ensemble, l'énoncé du travail pratique a été clair et simple à suivre. Nous voyons clairement le lien entre les objectifs et le travail accompli. Afin d'améliorer le travail pratique, nous recommandons d'ajouter des exemples de gestion d'erreur. Par exemple,

devons-nous demander l'adresse IP et le port jusqu'à la connexion, ou demander le nom d'utilisateur et le mot de passe jusqu'à la validation, etc.

Conclusion

Bref, nous avons pu atteindre les objectifs de ce laboratoire avec succès. Comme prévu, nous avons appris à faire une connexion à travers un *socket* afin de faire communiquer un client avec un serveur. Puisque la majorité des logiciels dans le monde aujourd'hui fonctionnent sur le principe de communication client-serveur, ce laboratoire nous a été utile pour apprendre les bases de cette communication. Dans le domaine du génie logiciel, ces connaissances seront particulièrement utiles. Surtout, apprendre ces bases en *Java* nous sera utile, car ce langage est très répandu dans le domaine professionnel. Dans la totalité, le laboratoire nous a permis d'appliquer les principes de communication sans fil et d'adressage IP de manière pratique.