POLYTECHNIQUE
MONTRÉAL

INF3405

RÉSEAUX INFORMATIQUES

SESSION ÉTÉ 2024

TP1

TRAITEMENT D'IMAGE PAR RÉSEAU

SOUMIS À: BEDDOUCH Sara

Soumis par :

-Francis Goudreau Caron : 2222302

-Lydia Hammadou : 2298914

-Anis Menouar : 2247873

INTRODUCTION :

Dans ce laboratoire, on va mettre de l'avant les échanges client-serveur en utilisant les sockets et le développement d'applications réseau en utilisant les threads, spécifiquement en Java. Le but premier de cette application et de permettre à aux usagers du réseau informatique de se partager des images et de pouvoir les modifier avec un filtre Sobel. Lorsque l'usager se connecte en tant que client, il pourra envoyer au serveur l'image qu'il veut traiter. Lorsque le serveur aura reçu et appliqué le filtre Sobel à partir d'un algorithme, il pourra le renvoyer au client. Le projet se concentre sur une interface console pour garantir le bon fonctionnement du programme sans s'occuper d'une interface graphique. En ce qui est des requis, le client doit être en mesure de se connecter, de se déconnecter, de se créer un compte et d'envoyer ou de recevoir des messages. Enfin, le serveur doit pouvoir authentifier ou créer un utilisateur, communiquer avec le client et maintenir une database des utilisateurs.

PRÉSENTATION :

Pour commencer, nous avons décomposé le système en plusieurs petits projets Java afin de respecter une approche orienté objet. Comme fichier principaux, nous avons évidemment le Client et le Server. Ensuite, le ClientHandler, comme son nom l'indique, va s'occuper des clients qui veulent se connecter au serveur. Pour se faire, le serveur créer un objet ClientHandler pour chaque nouveau client qui se connectent afin de pouvoir gérer les connexions individuelles de ceux-ci dans une application serveur multi-thread. Le rôle de ClientHandler est de régulé l'interaction avec les clients: il authentifie les clients, reçoit les images envoyées par ceux-ci, les traite en utilisant une méthode de la classe ImageHandler, et renvoie les images traitées aux clients. Lors de la réception de chaque image à traiter, il imprime un message détaillé dans la console du serveur, incluant le nom d'utilisateur, l'adresse IP, le port du client, la date et l'heure, et renvoie au client la confirmation de la réception.

Le ClientDataBase est la classe qui s'occupe de la base de données locale du serveur. Celui-ci est implémenté dans la classe Server, et utilise un fichier .txt comme base de données. Il comprend les méthodes servant à stocker, à lire et à vérifier l'existence d'un nom d'utilisateur et de son mot de passe pour chaque usager qui se connecte au serveur. La logique de code est relativement simple. Pour chaque ligne de la database, nous associons un nom d'utilisateurs et un mot se passe séparé par un caractère spécial permettant de définir la limitation entre les deux parties.

La classe ‘ImageHandler’ gère exclusivement le traitement des images. Elle contient les méthodes permettant de lire/écrire une image et celles qui permettent l'envoi et la réception d’une image. De plus, elle contient une méthode qui applique l'algorithme du filtre Sobel et renvoie l’image au client. La première méthode, ‘sendImage’, manipule les octets pour transmettre l'image complète dans un flux de sortie. Ensuite, ‘receiveImage’ utilise un tableau pour stocker les octets reçus et est ensuite reformaté en BufferedImage. ‘readImage’ récupère l'emplacement de l'image dans la mémoire à partir du chemin spécifié. ‘saveImage’ sert à sauvegarder une image en écrivant un nouveau fichier .jpg à partir d’une BufferedImage. Enfin, ‘sendTreatedImage’ commence par appeler ‘readImage’, puis applique le filtre Sobel avant d'envoyer l'image. D’ailleurs, toutes les méthodes de cette classe utilisent la classe java BufferedImage, permettant ainsi une meilleure manipulation des images dans la mémoire.

L’iPAdressChecker est un petit projet avec une classe qui ne fait que retourner un booléen vérifiant si l’adresse IP est bien valide selon le format IPV4 en s’assurant qu’on respecte la bonne structure d’une adresse de ce format.

En ce qui concerne la classe Server, elle encapsule les attributs suivants: le nombre de clients qui se sont connecté, l’adresse du serveur et le port du serveur. Afin de déterminer le port et l’addresse, on initialise un objet de lecture (readObject) à partir de la classe Java Scanner qui fait partie du package java.util. Cet objet est utilisé pour lire les entrées depuis la console, permettant ainsi au programme d'interagir avec l'utilisateur en acceptant/vérifiant des valeurs d'entrée, qu’il transforme en token. Il est utilisé spécifiquement pour saisir l'adresse IP et le numéro de port entrés par l'utilisateur pour configurer le socket d'écoute du serveur. De plus, on a l’objet « listener » qui est une instance de l’objet ServerSocket et qui va attendre les connexions des clients sur un port spécifié. Bref, Cette classe Server met en place un serveur qui écoute les connexions entrantes sur une adresse IP et un port spécifié par l'utilisateur. Elle utilise les classes ServerSocket, InetAddress et Scanner pour gérer l'initialisation du serveur et la lecture des entrées utilisateur, et elle crée une instance de la classe ClientHandler pour gérer chaque connexion entrante.

DIFFICULTÉS RENCONTRÉES :

Les principales difficultés rencontrées ont été au niveau de la syntaxe Java. Puisque nous n'avions jamais programmer en Java, même si nous avions une bonne idée de la logique à employer pour tel ou tel partie du code, le fait de ne pas connaitre du tout les classes existantes et leurs utilités à demander beaucoup de recherche google, stackoverflow et sur le site d’oracle afin de comprendre la syntaxe spécifique du language et les outils/classes qui sont inclus(e) dans celui-ci. Heureusement, la documentation Java était facilement accessible et le site d’oracle est assez bien fait pour pouvoir comprendre les méthodes disponibles pour chaque classe que nous voulions utilisés.

Nous avons également eu un peu de difficulté sur la façon de faire pour la transmission de l’image, vu qu’il n’était pas possible de tout transmettre d’un coup. Cependant, le forum de stackoverflow était rempli d’explications pertinente concernant les différentes méthodes disponibles pour accomplir une telle tâche, donc nous avons pu nous en sortir sans trop avoir à se casser la tête.

CRITIQUES ET AMÉLIORATIONS :

Pas de réelles critiques ou améliorations pour ce laboratoire. Nous trouvons que c’est un travail très formatif et qui permet d’avoir une introduction à l’architecture client-serveur.

CONCLUSION :

En résumé, ce projet pratique nous a permis d'explorer diverses structures de données en Java, de nous familiariser avec la syntaxe, les concepts de serveurs et de clients, de mieux comprendre le fonctionnement des threads et de créer des exécutables. Nous sommes très satisfaits de ce projet pratique, car le fonctionnement et la théorie derrière ces concepts sont extrêmement pertinents pour des futurs ingénieurs.