Rendu TEA Système Répartis

Sommaire :

Marie DIEZ & Johan GUERRERO

Rendu de Marie DIEZ

1 - Travail réalisé et choix technique

Travail réalisé :
Le projet demandait à réaliser / finaliser un serveur https, celui-ci devait remplir un certain nombre de critères :

- Envoi de pages statiques demandées dans une requête ✓
- Exécution de programme côté serveur (GCI) ✓
- Echange sécurisé (https) ✓
- Multi-threading ✓

✓ Tâches effectuées.

Choix technique :

Nous avons mis en place tout les critères demandés, nous avons donc utilisé les technologies demandées, telle que :

- Le protocole HTTP
  - Utilisation de Socket pour la communication client / serveur. (*Précision dans la partie HTPPS)
- Le CGI
  - Utilisation d'un script cgi, écrit en python. Ce script réalise un addition avec des paramètres qui sont envoyés au script, celui-ci affiche le code html affichant la réponse de l'addition.
- Le HTTPS
  - Utilisation de « SSLServerSocketFactory » pour la création du «ServerSocket » sécurisé qui acceptera les connexions avec les clients.
  - Mise en place de properties, utilisant un certificat pour mettre en place la connexion sécurisé. J'ai généré ce certificat, celui-ci est donc auto-signé, le navigateur vous avertira alors d'un risque de sécurité car celui-ci verra que le certificat est auto-signé et ne provenant donc pas d'une autorité de certification vérifiée. Vous devez passer outre cet avertissement pour le bon fonctionnement du programme.
  - Ces technologies permettes une communication sécurisé entre client et serveur → HTTPS.

- Le Multi-threading:
  - Utilisation de Thread pour le mise en place du multi-threading.
  - Le serveur écoute en permanance pour l'arrivé d'un client. Lorsqu'un client se connecte au serveur, un thread est créer pour celui-ci de façon à le traiter en parrallèle de l'écoute du serveur. Chaque client à son propre Thread de traitement.

2 - Répartition des tâches

J'ai travaillé avec Johan GUERRERO sur ce projet. Nous nous sommes répartis les tâches de façon à être plus efficace pour la réalisation de ce projet.

- Envoi de pages statiques demandées dans une requête — Marie & Johan
- Exécution de programme côté serveur (GCI) — Johan
- Echange sécurisé (https) — Marie
- Multi-threading — Marie

3 - Ma contribution

Pour commancer j'ai travaillé avec Johan GUERRERO sur la mise en place du protocole HTTP classique, c'est-à-dire la mise en place d'un serveur et d'un navigateur (client) se connectant au serveur dans le but d'afficher une page html.

Une fois que cette première partie était réalisée nous avons séparé notre travail.

Je me suis alors penchée sur la mise en place d'une connexion sécurisé entre le client et le serveur. J'ai en premier lieu créer un certificat pour sécurisé cette connexion. Par la suite j'ai mis des properties à java (setProperties) indiquant le chemin du certificat et son mot de passe associé.
Une fois cela réalisé j'ai implémenté les « SSLServerSocketFactory » pour la mise en place de « ServerSocket » sécurisé pour finaliser l'implémentation de la sécurité de la connexion.

Une fois que le programme fonctionner en HTTPS / connexion sécurisé, j'ai travaillé sur le multi-threading. Pour réaliser cela, j'ai créé une classe « Traitement » qui traitera les clients lorsqu'un auront été accépté par le serveur. Le serveur est en attente continue de client, lorsqu'un qu'un client se connecte celui-ci est alors traiter dans un thread de Traitement. Chaque client sera traité dans son propre thread de traitement, traitement de la requête.

J'ai pour finir commenté le code et réalisé le makefile.

4 - Utilisation du programme

⚠ *J'ai généré ce certificat, celui-ci est donc auto-signé, le navigateur vous avertira alors d'un risque de sécurité car celui-ci verra que le certificat est auto-signé et ne provenant donc pas d'une autorité de certification vérifiée. Vous devez passer outre cet avertissement pour le bon fonctionnement du programme.*

Pour lancer le serveur : Lancer la commande « make » à la racine du projet.

Pour lancer le client : Utiliser un navigateur et se connecter à l'addresse suivante :

https://localhost:1234/addition.html