Rapport intermédiaire

Programmation réseau : Projet JarRet

UPEM

https://github.com/LiazidiAmine/oasis
Afin de lancer le client et le serveur et effectuer des tests : upem.jarret.test.Main:

I/Le client:

Le client upem.jarret.client.HTTPClient est basé sur l'architecture client vus en TP..

Organisation du travail :

- Mise en place du code basique qui permet une connexion à un serveur
- Effectuer une requête GET à dés la connexion du client
    - Vérifier la réponse du serveur et sa validité
    - Parser la réponse et la stocker
    - Si le serveur répond qu'il n'a pas de job, retenter une nouvelle connexion.
- Récupérer le JAR
    - Créer une instance du Worker
    - Lancer le calcul
    - Vérifier la réponse du worker
- Effectuer une requête POST en utilisant les résultats précédents

Ses méthodes:

- public Optional<String> sendTaskRequest() throws IOException

Envoie une requête GET au serveur afin de récuperer un worker. Elle lit ensuite la réponse du serveur, et renvois la réponse si elle existe.

- private Optional<Worker> checkWorkers(Map<String,String> job) throws MalformedURLException, ClassNotFoundException, IllegalAccessException, InstantiationException.

Prends le job envoyé par le serveur, et renvois le worker s'il existe déja.

- public Optional<Map<String,String>> runWorker(Map<String,String> job) throws MalformedURLException, ClassNotFoundException, IllegalAccessException, InstantiationException.

Prend le job envoyé par le serveur, et renvoi le résultat du worker.

- public void sendAnswerTask(String json, String result, String error) throws IOException.

Envois d'une requête POST avec le résultat du worker au serveur.

Utilisation d'une classe HTTPRequest qui permet la construction des requêtes du client, leurs vérification et leurs parsing.

-

II/Le serveur:

Le serveur upem.jarret.server.Server est basé sur l'architecture serveur vus en TP.

Organisation du travail :

- Mise en place d'un serveur basique qui accepte plusieurs clients en non bloquant.
- Mise en place d'un timeout pour les clients
- Lecture des workers depuis le fichier de configuration
- Gestion des requêtes POST et GET des clients

La classe TaskReader s'occupe de récupérer les workers, et les parser.
La classe ResponseBuilder permet de construire les réponses aux requêtes des clients.

Travail manquant :

- Gestion des cas d'erreurs et renvoi des erreurs 404
- Ajout d'une console afin d'interagir avec le serveur.