Rapport Final SAE 302

BOZENKO Wladimir RT21 (Cyber FA)

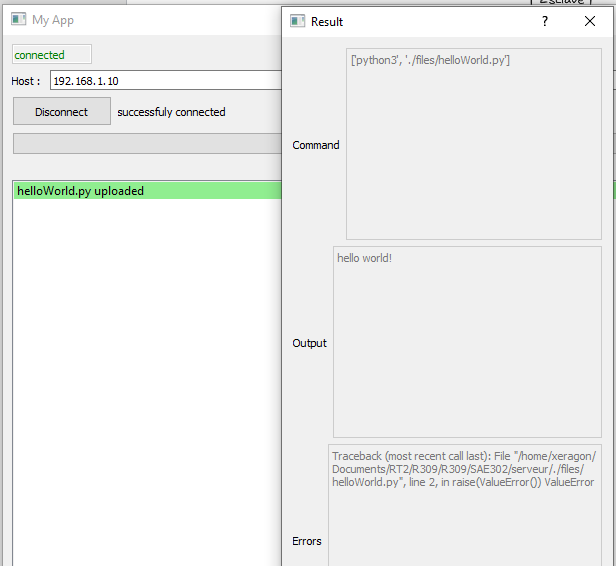
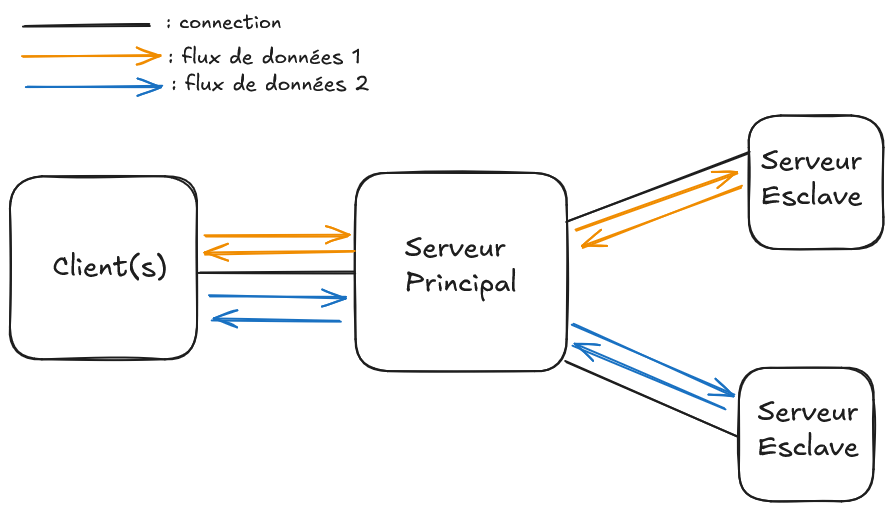
# Fonctionnalité

Au terme de cette SAE les fonctionnalités suivantes on été implémenter dans mon projet  :

## Base

Le projet implémente les fonctionnalités de base, soit la communication entre un ou plusieurs client (multi-clients) avec un serveur principales lui même communiquant avec des serveurs esclaves.

Le client envoie un fichier en .py .java ou .c au serveur, le serveur le distribue à un esclave qui se charge de le compiler si besoins puis de l’exécuter et transmet le contenu des sortie standard 1 et 2 (output et erreur) au serveurs principal qui lui à son tour les transmet au client



## Gestion d’erreurs

Les erreurs des zones à risque coté client sont intercepter et remontée via un dialogue

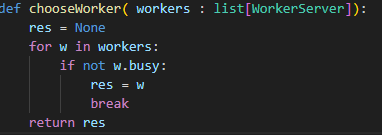
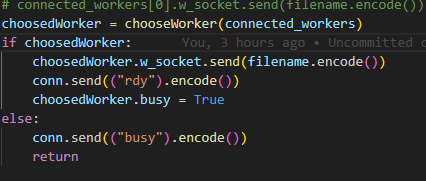
Coté serveur si la compilation échoue, le serveur renvoie un message d’erreur dans la partie errors du retour en JSON contenant un message indiquant que la compilation à échoué

## Load Balancing

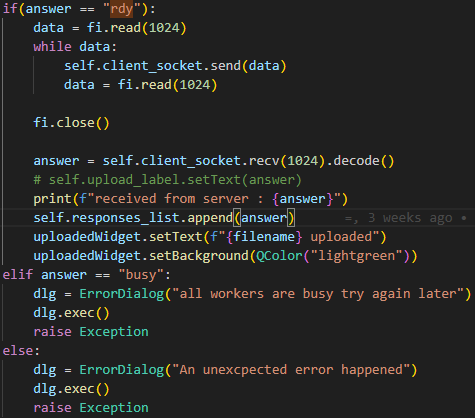
Il n’y pas d’implémentations de load balancing à proprement parler dans ce projet, cependant une implémentations « de disponibilité » des serveurs esclaves à été mise en place.

Lorsque le serveurs principal reçoit une demande d’upload, il consulte la liste des serveurs esclaves auxquelles il est connecté et transmet le flux de données au premier esclaves disponibles. Si aucun esclaves n’est disponible le serveur principale notifie le serveur client, qui lui affiche un message à l’utilisateur l’informant qu’aucun esclaves n’est disponible pour le moment.

Serveur principal :



Client :



## Scalabilité

Le lancement des serveurs esclaves et la connexion à ces derniers est faite à partir du fichier config.json



Il est donc très simple d’ajouter un grand nombre de serveurs esclaves en ajoutant des entrées dans ce fichier avant lancement des serveurs

## Cybersécurité

Aucune solution n’a été mise en place afin de sécuriser l’application, cependant voici quelque piste à explorer afin de rendre l’application plus sur :

* Mise en place d’un système d’authentification lors de la connexion client/serveur
* Contenairisation lors de l’exécution des codes envoyées par les clients afin d’éviter d’exécuter du code malveillant (avec Docker par exemple)
* Implémenter un système de reconnaissance de code malveillant à l’aide des base de données de malware/vulnérabilité existante + système de blacklist des clients ayant envoyée du code malveillants