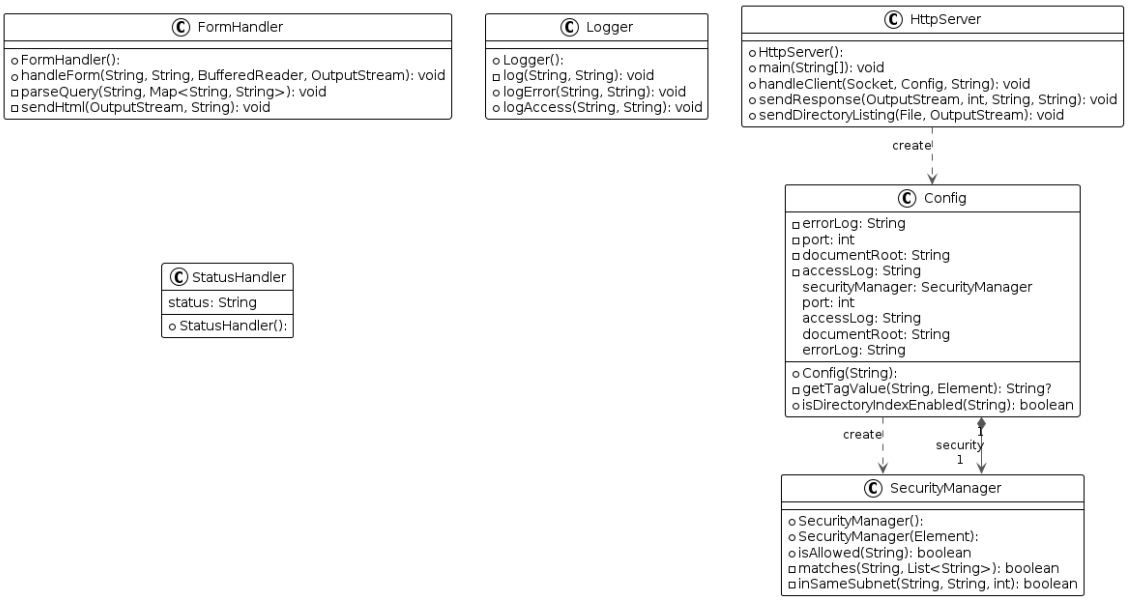
# Rapport SAE 2.04 - Mise en place d’un serveur web

## Introduction

Dans le cadre de la SAE 2.03 du BUT Informatique, nous devions développer un **serveur Web configuré en Java**, capable de répondre à des requêtes HTTP (GET et POST), de lire sa configuration depuis un fichier XML, de gérer la sécurité, les logs, l’état du système, et les formulaires HTML.

Le but était de comprendre comment fonctionne un serveur web basique.

Nous avons organisé le projet autour de plusieurs classes Java.




---

## Fonctionnalités implémentées par classe

### HttpServer.java – Serveur principal

Cette classe contient la méthode main et gère les connexions entrantes. Elle lit le fichier de configuration au démarrage.

C’est aussi dans cette classe que j’ai traité les chemins spéciaux ainsi que les réponses 404 et 403.

Le test principal a été de lancer le serveur, d’ouvrir une page dans un navigateur, et de vérifier que le bon fichier s'affiche.

### Config.java – Lecture de la configuration XML

Cette classe lit le fichier de configuration qui contient les balises.

### SecurityManager.java – Gestion des adresses IP

Cette classe vérifie si une adresse IP est autorisée ou non à accéder au serveur.

### Logger.java – Écriture dans les fichiers de log

Cette classe contient deux méthodes : une pour écrire les accès et une pour écrire les erreurs.

Elle ajoute un horodatage à chaque ligne et écrit dans les fichiers définis dans la configuration.

### StatusHandler.java– Affichage de l'état du serveur

Quand on visite l'URL avec /status, cette classe renvoie une page texte qui affiche : la mémoire libre, l'espace disque disponible, le nombre de processus, et le nombre d'utilisateurs connectés.

### FromHandler.java – Traitement des formulaires

Cette classe gère les formulaires envoyés via GET ou POST. Elle lit les paramètres et enregistre les données dans un fichier texte.

## Lancement du serveur :

**Attention ⚠️**

Une fois le package .deb récupéré il est possible que des conflits sur les scripts de lancement et d'arrêt surviennent en raison des différences de retour à la ligne sur Windows et Linux (*CRLF* vs *LF*).

Il faut donc faire ces commandes :

```
sudo dpkg -i myweb_package.deb # pour télécharger le package debian

file /usr/local/sbin/start_myweb.sh
file /usr/local/sbin/stop_myweb.sh

# les résultats de ces commandes ne doivent pas faire apparaitre que les scripts sont
# en CRLF, si c'est tout de même le cas voici les commandes à réaliser

sudo dos2unix /usr/local/sbin/start_myweb.sh # permet de convertir CRLF en LF
sudo dos2unix /usr/local/sbin/stop_myweb.sh
```

Une fois que les deux scripts sont biens formattés en LF comme il le faut sur linux on peut lancer le script de lancement du serveur avec la commande suivante :

```
sudo /usr/local/sbin/start_myweb.sh # permet le lancement du serveur
```

Vous pouvez alors vous rendre sur votre navigateur et tapez :

```
http://localhost:8080

# ou directement depuis le terminal
firefox http://localhost:8080
```

Ensuite pour fermer le serveur il vous suffit de taper la commande suivante qui lance le script éteignant le serveur grâce à son PID :

```
sudo /usr/local/sbin/stop_myweb.sh
```

## Conclusion

Nous avons réussi à implémenter toutes les fonctionnalités demandées dans le cahier des charges de la SAE 2.03 avec plus ou moins de difficultés.

Evidemment, nous nous sommes beaucoup aidé de programme déjà existants disponibles sur internet (*StackOverFlow, GitHub, DebianWiki, ...*) pour pouvoir faire toutes les fonctionnalités.