Rapport :

- Fuchs Thomas
- Comte Gabriel

Algorithme serveur principale :

```
Début

// Initialisation des attributs du serveur
Définir reader comme objet ReaderFileConf
Définir filter comme objet FilterIP
Définir logAccess comme objet Log
Définir logError comme objet Log
Définir status comme objet Status
Définir loader comme objet LoaderHtml

// Constructeur du serveur
Fonction Server()
  Initialiser reader avec "/etc/myweb/myweb.conf"
Fin Fonction

// Méthode de filtrage des IP
Fonction filter(ip: Chaîne) : Booléen
  Définir list comme ArrayList<String>
  Définir list2 comme ArrayList<String>

  list = reader.readConf("accept")
  list2 = reader.readConf("reject")
```

```
    Initialiser filter avec ip
    Définir accept comme Faux

    Pour chaque s dans list
        Si filter.passIP(s) alors
            accept = Vrai
        Fin Si
    Fin Pour

    Pour chaque s dans list2
        Si filter.passIP(s) alors
            accept = Faux
        Fin Si
    Fin Pour

    Retourner accept
Fin Fonction

// Méthode principale
Fonction main(args: Array<String>)
    Définir server comme objet Server
    Initialiser logAccess avec server.reader.readConf("accesslog")[0]
    Initialiser logError avec server.reader.readConf("errorlog")[0]

    Définir interpret comme objet Interpret avec "C:\\msys64\\ucrt64\\bin\\python3.exe"

    Essayer
```

```
    Définir serverSocket comme objet ServerSocket avec
Integer.parseInt(server.reader.readConf("port")[0])

    Tant que Vrai
      Définir socket comme objet Socket après serverSocket.accept()

      // Récupérer l'adresse IP du client
      Définir ip comme socket.getInetAddress().getHostAddress()
      Si ip est "0:0:0:0:0:0:0:1" alors
        ip = "127.0.0.1"
      Fin Si

      // Filtrer l'adresse IP
      Si !server.filter(ip) alors
        server.logError.writeLog("IP " + ip + " refused")
        Définir message comme "HTTP/1.1 403 Forbidden\n\n"
        socket.getOutputStream().write(message.getBytes())
      Sinon
        server.logAccess.writeLog("IP " + ip + " connected")

        Définir reader comme objet BufferedReader avec
InputStreamReader(socket.getInputStream())
        Définir request comme Chaîne

        Répéter
          request = reader.readLine()
        Jusqu'à (request == null ou !request.split(" ")[0].equals("GET"))

        request = request.split(" ")[1].substring(1)
```

```
            Si request.equals("@Status") alors
              Définir runtime comme objet Runtime
              Définir freeMemory comme runtime.freeMemory()
              Définir availableProcessors comme runtime.availableProcessors()
              Définir freeDiskSpace comme 0

              Pour chaque store dans FileSystems.getDefault().getFileStores()
                freeDiskSpace += store.getUsableSpace()
              Fin Pour

              server.logAccess.writeLog("Status of the server")
              server.status = new Status(Long.toString(freeMemory),
Long.toString(freeDiskSpace), Integer.toString(availableProcessors))
              socket.getOutputStream().write(server.status.getStatus().getBytes())
            Sinon
              server.loader = new LoaderHtml(server.reader.readConf("root")[0] + request)
              socket.getOutputStream().write(server.loader.load(server.logAccess,
server.logError).getBytes())
              socket.getOutputStream().write(interpret.interpreteurPyhton().getBytes())
            Fin Si
          Fin Si
        Fin Tant que
      Attraper Exception e
        e.printStackTrace()
      Fin Essayer
Fin Fonction

Fin
```