Creación de test en JMeter

Ignacio Vellido Expósito

Se explicarán los pasos para la realización de la parte de la práctica 4 relacionada con JMeter, que incluye la instalación de los recursos necesarios (Docker y Docker Compose) y la configuración del test.

A través de JMeter se simularán peticiones a una página web docente, donde los usuarios alumnos recibirán un token para identificarlos y de esta manera mostrarle recursos específicos; los administradores de igual manera tras identificarse pueden recibir información del perfil de un alumno.

Requisitos previos

- Instalamos Docker
  Añadimos llave GPG para validar el repositorio
  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

  Añadimos repositorio
  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

  Actualizamos lista de repositorios
  sudo apt update

  Instalamos el repositorio de docker
  sudo apt install docker-ce

```
ive@ubuntuISEC3:~/iseP4JMeter$ sudo systemctl status docker
[sudo] password for ive:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since jue 2018-11-22 16:16:44 CET; 2 weeks 3 days ago
     Docs: https://docs.docker.com
 Main PID: 6578 (dockerd)
```

(Se puede comprobar con **sudo systemctl status docker**)

  Añadimos el usuario al grupo docker
  sudo usermod –aG docker ive

(Debemos salir con **exit** y volver a entrar, ahora no hace falta usar sudo para los comandos de docker)

```
ive@ubuntuISEC3:~/iseP4JMeter$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

(Se puede probar con **docker info** o con **docker run hello-world**)

- Instalamos Docker Compose
  sudo apt install docker-compose

```
ive@ubuntuISEC3:~/iseP4JMeter$ docker-compose --version
docker-compose version 1.8.0, build unknown
```

(Se puede probar con **docker-compose --version**)

Instalación y uso de la app

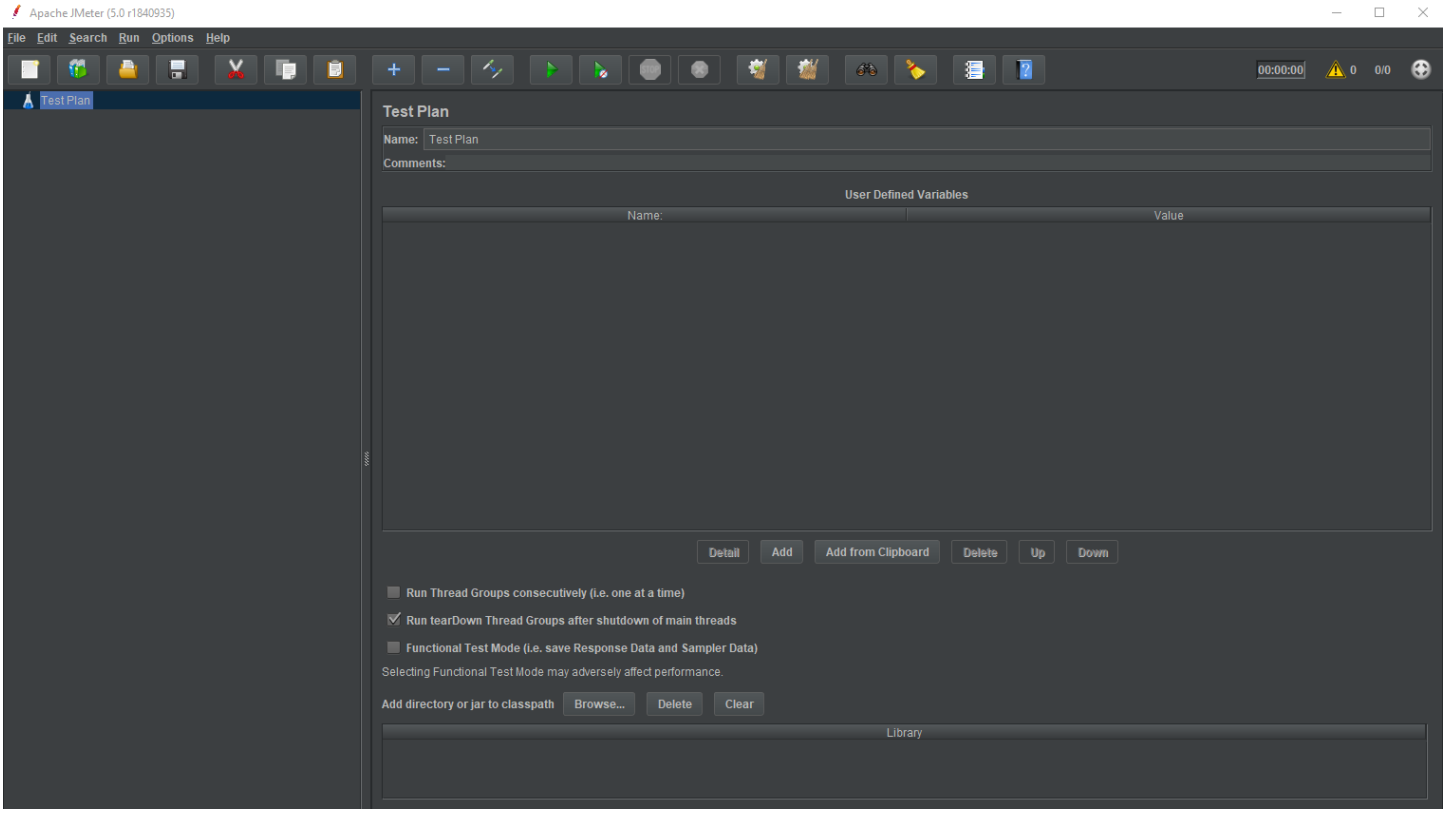
- Instalamos la app
  Clonamos desde git
  git clone https://github.com/davidPalomar-ugr/iseP4JMeter.git

  Accedemos al directorio y lanzamos la aplicación
  cd iseP4JMeter
  docker-compose up

```
node_js_1        | GET / 304 48.057 ms - -
node_js_1        | GET / 304 1.256 ms - -
node_js_1        | GET /stylesheets/style.css 304 2.484 ms - -
node_js_1        | GET /stylesheets/style.css 304 0.539 ms - -
node_js_1        | GET /favicon.ico 404 3.497 ms - 41
```

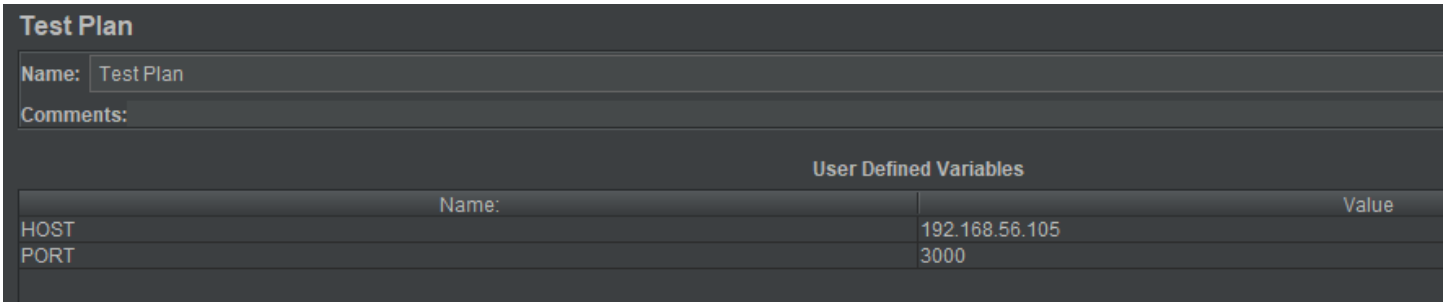
Creación del test en JMeter

- Instalamos JMeter y ejecutamos el archivo **jmeter.bat** (localizado en el subdirectorio **bin**)

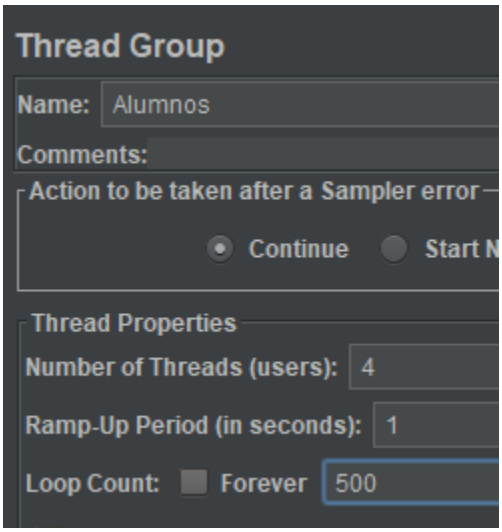


- Creamos un **Test Plan** con dos **Thread Group** y un **Test Fragment**

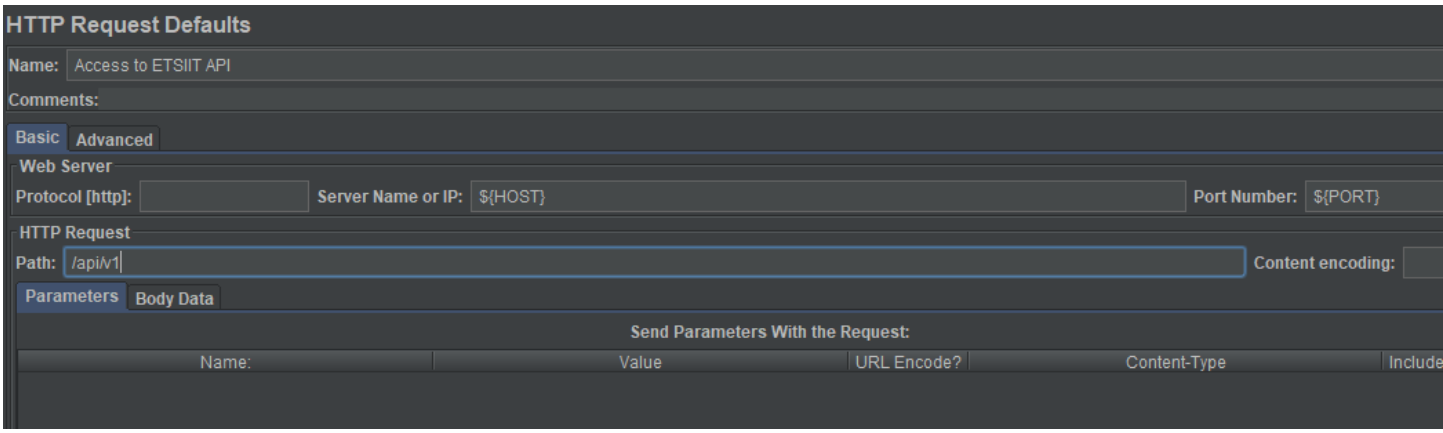
- Definimos variables globales para la IP (HOST – 192.168.56.105) y para el puerto (PORT - 3000). Se acceden con la nomenclatura **${…}**

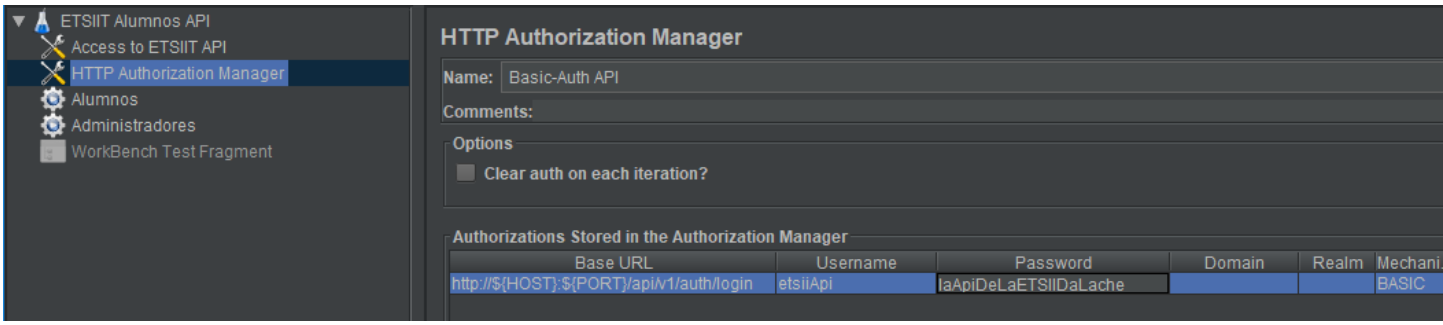


- Indicamos en los **Thread Group** el número de hebras, el tiempo en el que deben estar lanzadas y el número de repeticiones.

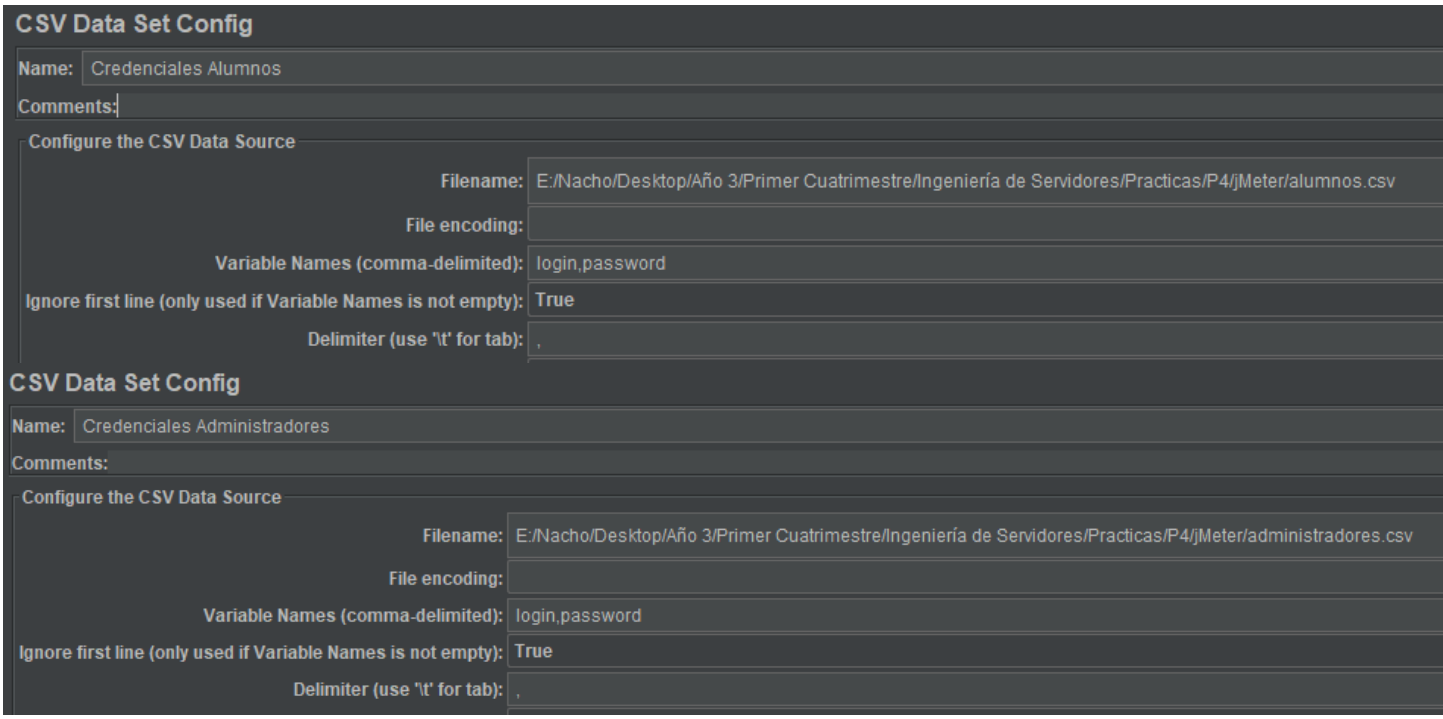


- Creamos un **HTTP Request Defaults** con acceso a **/api/v1**, también un **HTTP Authorization Manager** con la URL, el usuario y la contraseña.

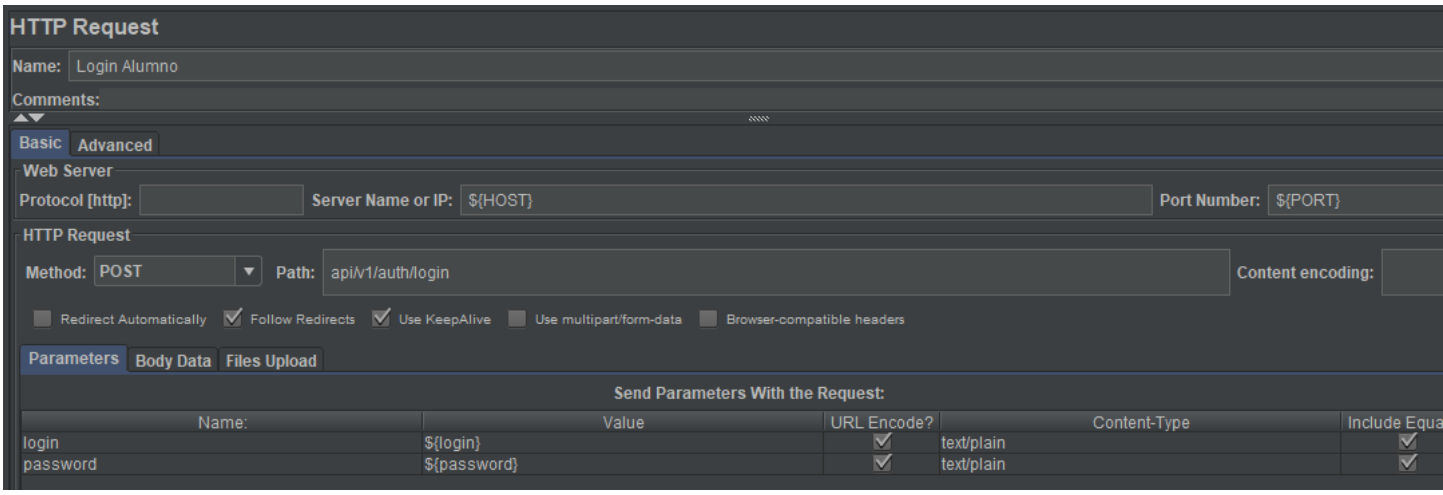


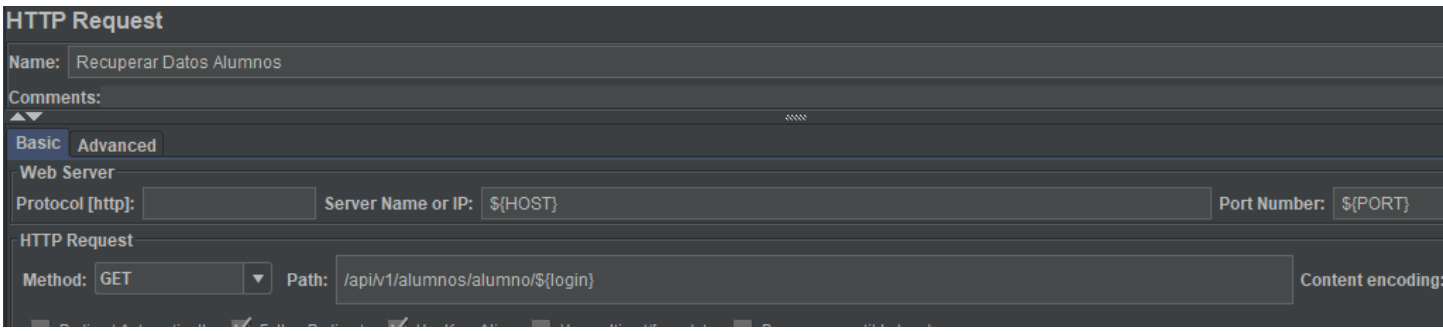


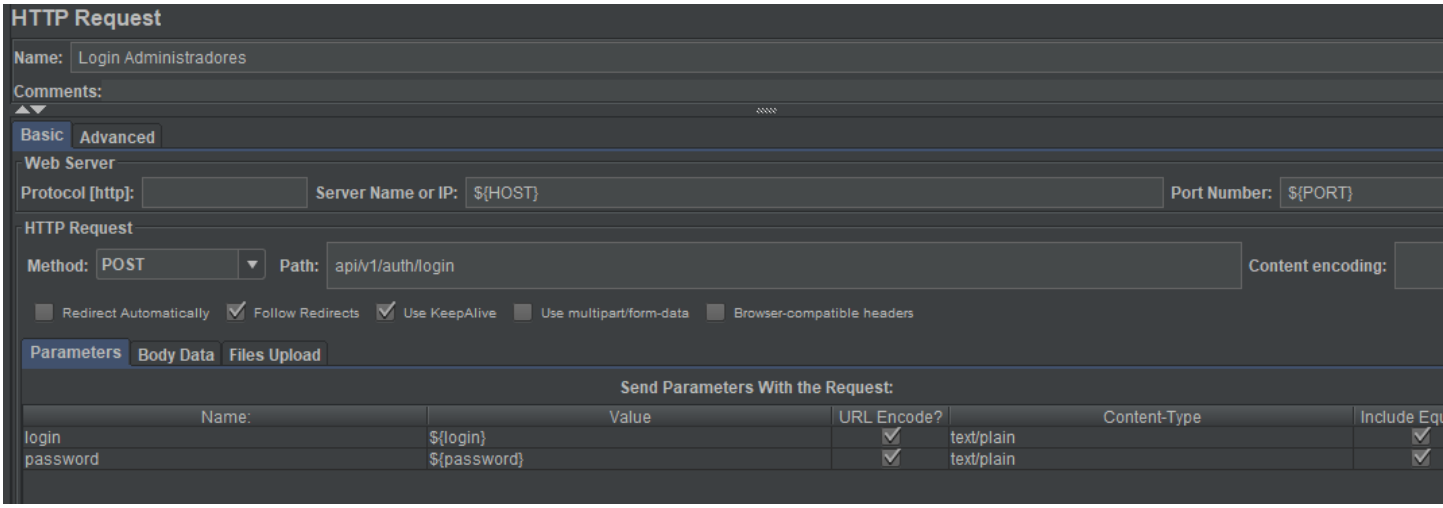
- Para cada **Thread Group** añadimos un **CSV Data Set Config** indicando el nombre de las variables, que se ignore la primera línea, el archivo del que se coge y el delimitador (en nuestro caso la coma).



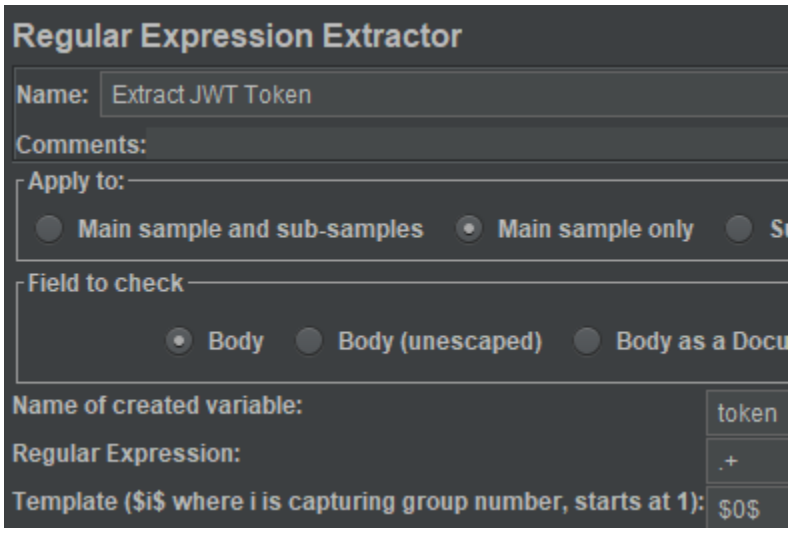
- Creamos 3 **HTTP Request**, uno para el login de los alumnos, otro para el de los administradores (indicando el path y los parámetros enviados (se cogen de las variables de CSV)), y uno adicional para solicitar la información de un alumno.





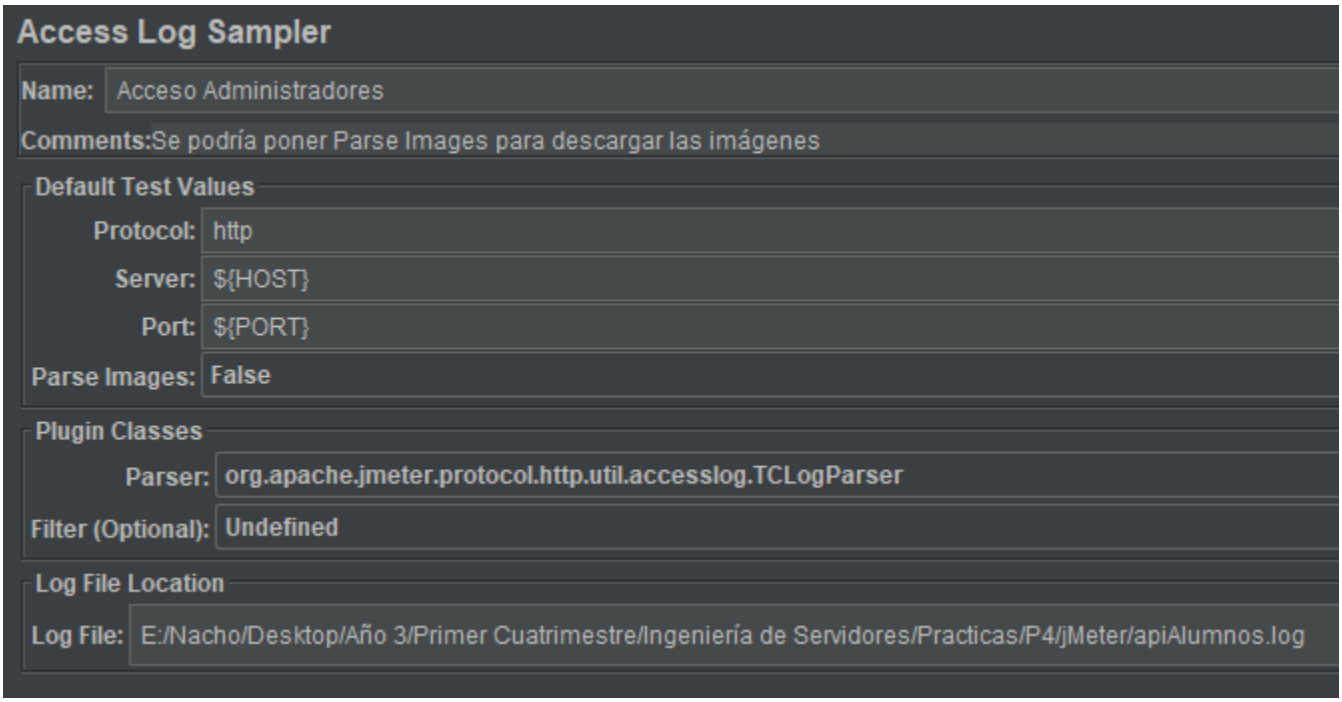


- En las peticiones Login se añade un extractor de token con expresión regular **.+** y template (la parte de la expresión encontrada que se selecciona) **0**, se define la variable **token**.



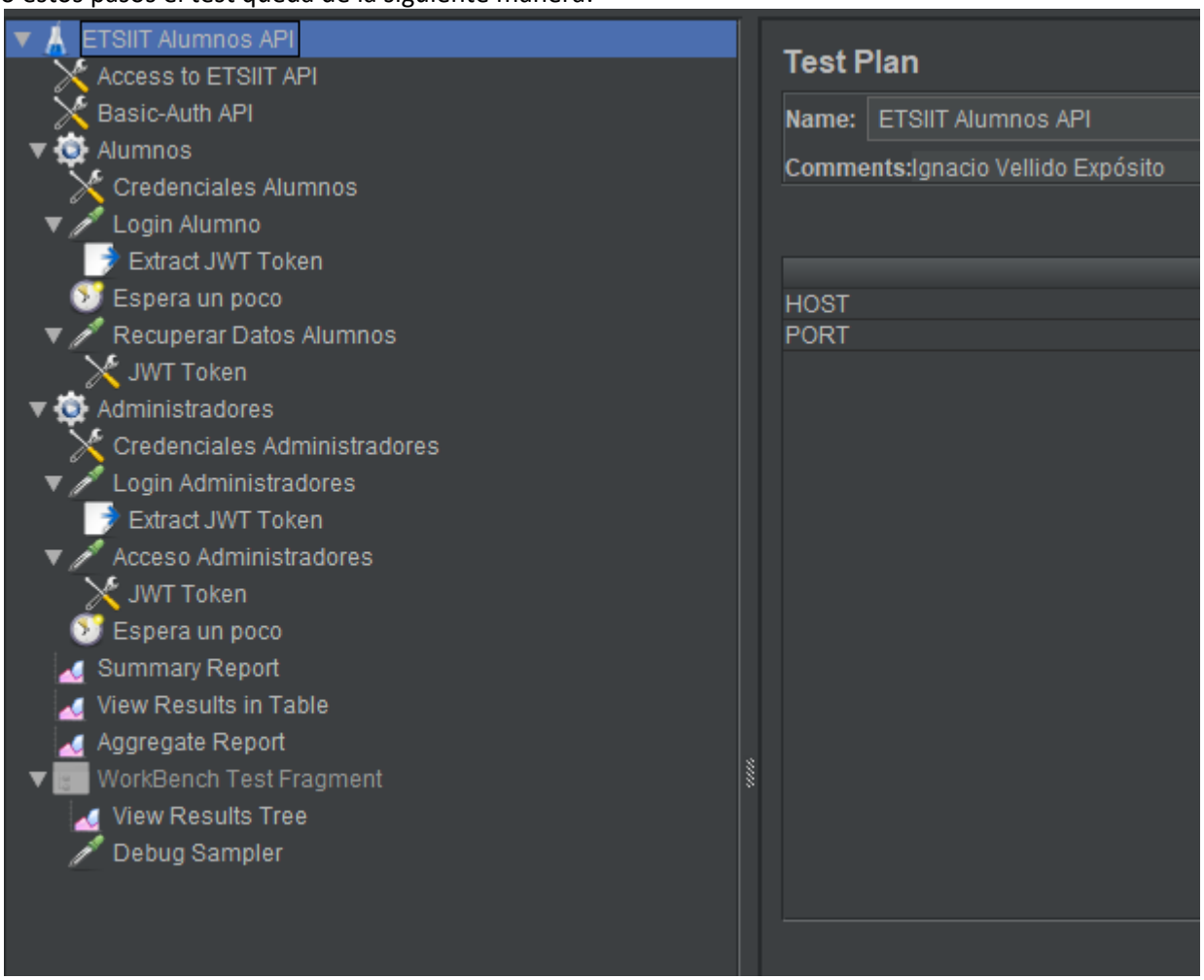
- Se añaden **Gaussian Random Timer** a los **Thread Group.**

- Se crea **Access Log Sampler** que vuelca la información en **apiAlumnos.log**, sirva para muestrear el acceso de administradores.

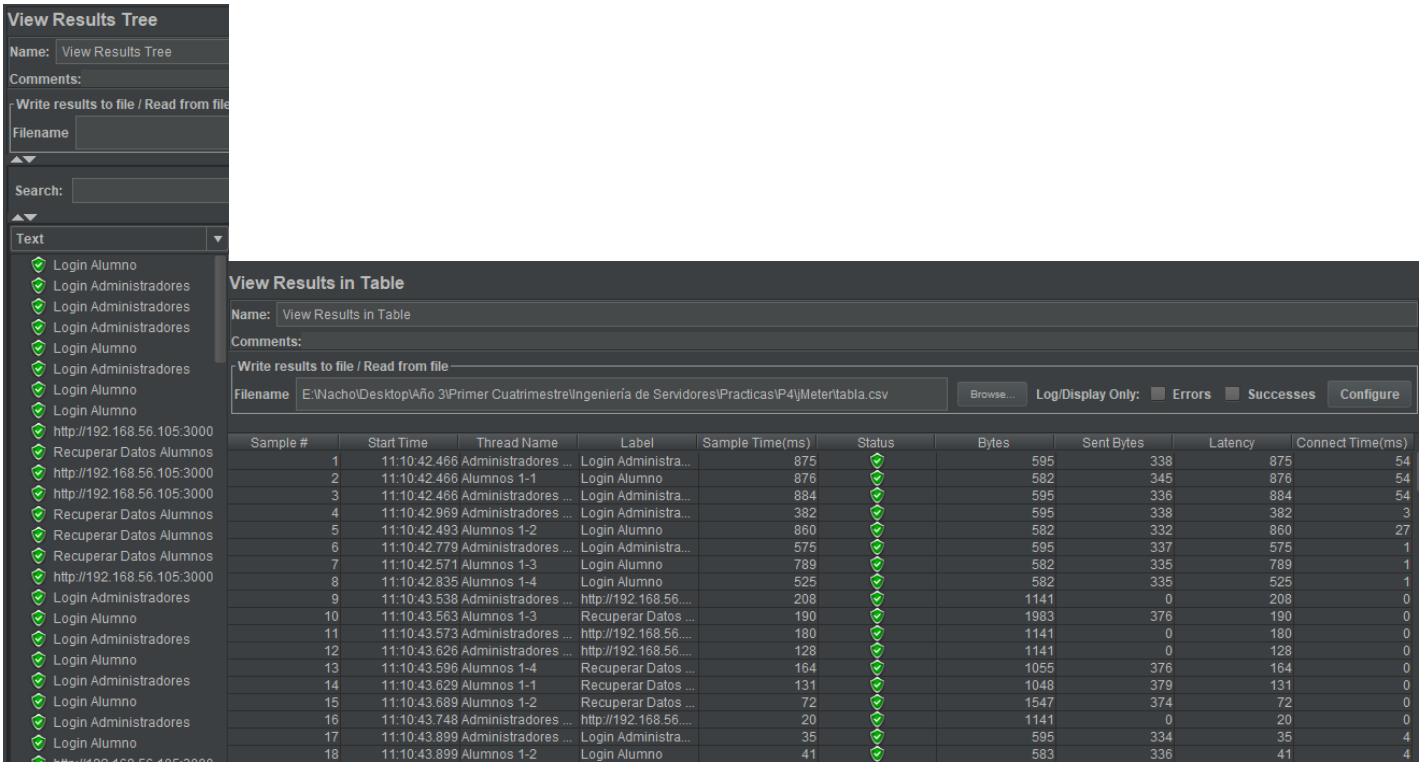


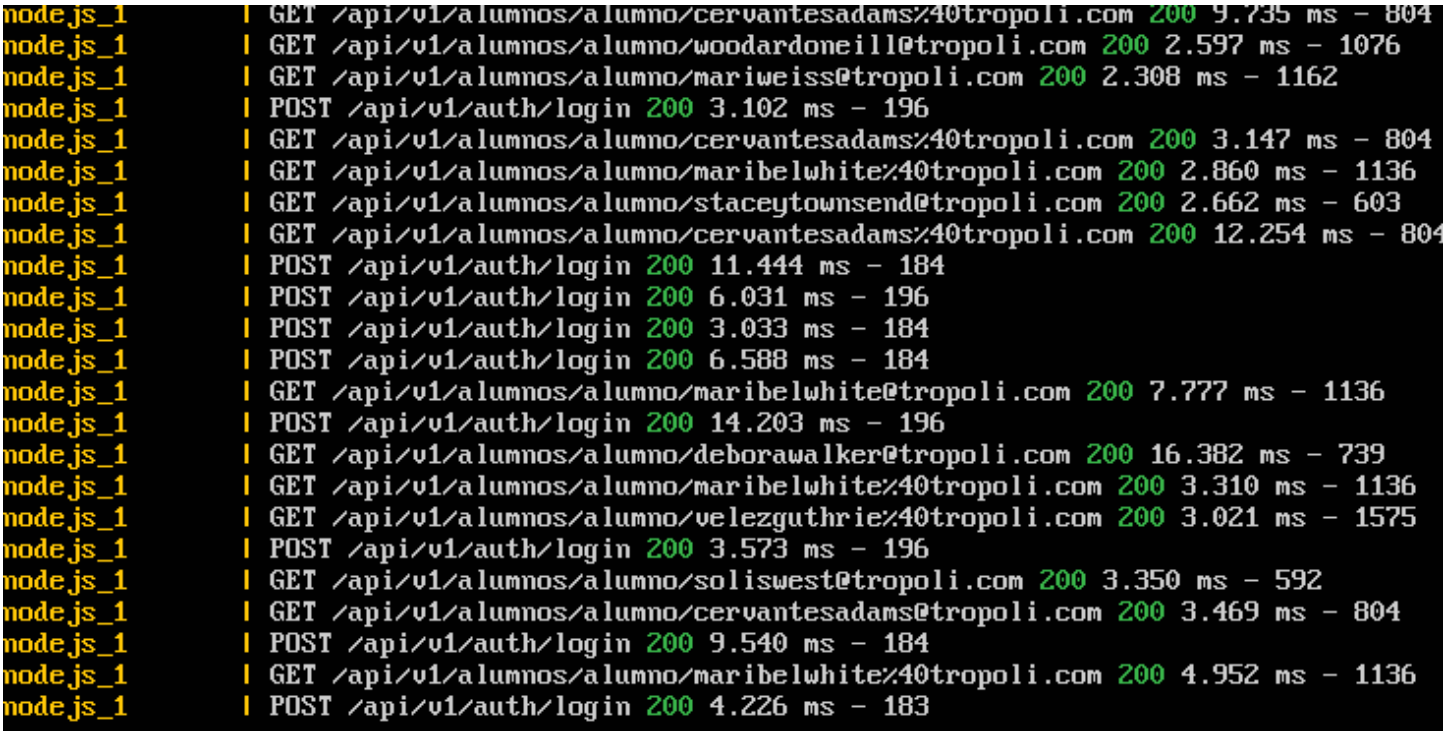
- Se introducen los siguientes **Listeners** dentro del **Test Plan**:
  Summary Report
  View Results in Table
  Aggregate Report

Siguiendo estos pasos el test queda de la siguiente manera:



Con los Listeners podemos ver que funciona correctamente:





Peticiones realizadas tras una ejecución del test

Bibliografía:

Instalación de Docker:

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04

Subdirectorio Git con los elementos necesarios para la práctica:

https://github.com/davidPalomar-ugr/iseP4JMeter

Instalación de JMeter

http://apache.rediris.es/jmeter/binaries/

Guía de uso básico de JMeter

http://jmeter.apache.org/usermanual/build-web-test-plan.html