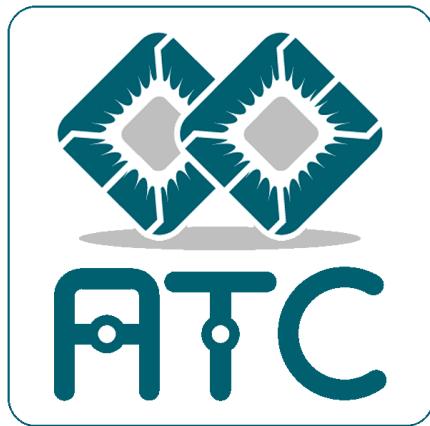




# Universidad de Granada

JMETER



Autor: Miguel Ángel Campos Cubillas.

<b>DESCRIPCIÓN DEL EJERCICIO</b>	<b>3</b>
RESUMEN DEL PROCESO	6
<b>INSTALACIÓN DE DOCKER EN UBUNTU</b>	<b>7</b>
<b>INSTALACIÓN DE DOCKER-COMPOSE EN UBUNTU</b>	<b>8</b>
<b>INSTALACIÓN DE APLICACIÓN PARA TEST CON JMETER EN UBUNTU</b>	<b>8</b>
<b>INSTALACIÓN DE JMETER EN HOST</b>	<b>9</b>
CONFIGURACIÓN DEL TEST	10
PARAMETRIZAMOS EL HOST Y EL PUERTO	10
CREAMOS DOS GRUPOS DE HEBRAS DISTINTOS PARA SIMULAR EL ACCESO DE ALUMNOS Y ADMINISTRADORES A LA API	10
GENERAMOS LA PETICIÓN HTTP	11
AÑADIMOS LA AUTORIZACIÓN	11
CONFIGURACIÓN DE CREDENCIALES ALUMNO	11
AÑADIMOS LOGIN DE ALUMNOS	12
EXTRACCIÓN DEL TOKEN DEL LOGIN ALUMNO	12
AÑADIMOS TEMPORIZADOR	12
CONFIGURACIÓN DE CREDENCIALES ADMINISTRADOR	13
RECUPERACIÓN DE DATOS DEL ALUMNO	13
GESTOR DE CABECERA DE HTTP	13
LOGIN ADMINISTRADORES	14
EXTRAEMOS EL TOKEN DEL LOGIN ADMINISTRADOR	14
MUESTREADOR DE ACCESO A LOG	14
GESTOR DE CABECERA HTTP	15
AÑADIMOS UN TEMPORIZADOR GAUSSIANO	15
SUMMARY REPORT, VIEW RESULTS TREE & AGGREGATE REPORT	15
TEST FRAGMENT	16
APLICACIÓN DEL TEST SOBRE LA API REST	16
EJECUCIÓN DEL TEST MEDIANTE COMANDO TERMINAL	18

# DESCRIPCIÓN DEL EJERCICIO

El ejercicio consiste en realizar una prueba de carga de una API Rest empleando JMeter.

La API sobre la que se desplegará la prueba de carga ha sido desarrollada empleando:

- MongoDB
- NodeJS
- Express

El código de dicha API se encuentra alojado en el [repositorio github](#) del propio profesorado de la asignatura. Su desarrollo queda fuera del ámbito de estudio de la asignatura. Nosotros nos limitaremos a trabajar en preparar una prueba de carga que lanzar sobre dicha API.

El servidor se distribuye de forma de una aplicación de contenedores de Docker sobre Compose. Ambas aplicaciones deben estar instaladas para ejecutar el servidor:

- Docker Community Edition: <https://docs.docker.com/install/>
- Docker Compose: <https://docs.docker.com/compose/>

Tras descargar el código, situarse en el directorio principal (al mismo nivel del archivo docker-compose.yml) y ejecutar:

```
docker-compose up
```

Para parar la aplicación ejecutar:

```
docker-compose down
```

Docker descargará las imágenes base y construirá nuevas imágenes para la aplicación.

Accediendo con un navegador a <http://<IPDockerContainer>:3000> (ej.- <http://localhost:3000>) se presenta la descripción básica de la api. Se tratan de dos métodos:

- /auth/login: Permite identificarse al usuario como Alumno o Administrador. El acceso a este servicio está protegido por Http Basic Auth. Una vez autenticado, se obtiene un JWT Token para ser empleado en el servicio de alumno.
- /alumnos/alumno: Devuelve el registro de las calificaciones del alumno. Los administradores pueden consultar los datos de cualquier alumno. Los alumnos solo los propios. Se debe proporcionar un JWT válido (obtenido en el login) que portará la identidad (autenticación) y rol (autorización) del usuario.

El proceso de consulta es el siguiente:

1. Identificarse en el servicio de login proporcionando credenciales válidas de alumno o administrador. Obteniendo un token.
2. Solicitar los datos del propio alumno identificado (alumno) o de un grupo de alumnos (administrador).

Para una prueba más detallada de que el entorno funciona, instala curl y ejecuta el script:

```
pruebaEntorno.sh
```

Este script contiene la secuencia descrita anteriormente en invocaciones a curl, por lo que describe las operaciones a realizar. La primera línea contiene la variable SERVER. Debe definirse la IP donde corre el contenedor de Docker. Si todo está correctamente configurado, obtendrá el perfil de un alumno, como el ejemplo siguiente:

```
{
  "_id": "5cdfd96c6731c5f7bc5b552d",
  "nombre": "Mari",
  "apellidos": "Fletcher Weiss",
  "sexo": "female",
  "email": "mariweiss@tropoli.com",
  "fechaNacimiento": "1992-04-04T00:00:00.000Z",
  "comentarios": "Aliquip dolor laboris ullamco id ex labore. Ipsum eiusmod ut aliquip non cillum deserunt sunt commodo anim ad nisi excepteur eu deserunt. Sit sunt proident Lorem irure irure minim adipisicing cillum. Nostrud officia in proident velit velit sit fugiat pariatur quis ad laboris minim dolor elit. Sint velit pariatur commodo sint veniam exercitation. Duis proident minim consequat consectetur sint et tempor labore culpa esse. Exercitation laborum non esse mollit tempor ea dolor minim adipisicing mollit in aliqua.\r\nullamco adipisicing excepteur commodo sunt nulla quis sunt velit Lorem pariatur sunt ad do incididunt. In eu nostrud ullamco laboris eu minim. Consequat sit et eiusmod officia ex sit minim sit laborum quis laborum labore non. Dolor nulla ut pariatur reprehenderit minim dolore consequat sunt aliquip ipsum esse. Excepteur consequat fugiat elit et nisi dolore aute minim nostrud et.\r\n",
  "cursos": [
    {
      "curso": 1,
      "media": 5.2
    },
    {
      "curso": 2,
      "media": 9.1
    }
  ],
  "usuario": 10
}
```

```
}
```

Los datos de alumnos se han generado automáticamente empleando la herramienta [Json-Generator](#), con las plantillas situadas en [/mongodb/scripts](#).

```
json-generator_administradores.json
```

```
json-generator_alumnos.json
```

El subdirectorio JMeter contiene los archivos necesarios para realizar la sesión de prácticas:

- [alumnos.csv](#): Archivo con credenciales de alumnos.
- [administradores.csv](#): Archivo con credenciales de administradores.
- [apiAlumno.log](#): Log de acceso Http en formato apache.

La prueba de JMeter debe:

- Parametrizar el “EndPoint” del servicio mediante variables para la dirección y el puerto del servidor. Emplee “User Defined Variables” del Test plan.
- Definir globalmente las propiedades de los accesos Http y la Autenticación Basic. Emplee HTTP Request Defaults y HTTP Authorization Manager.
- Los accesos de alumnos y administradores se modelarán con 2 Thread Groups independientes. La carga de accesos de administradores se modelará empleando el registro de accesos del archivo [apiAlumno.log](#).

La imagen siguiente presenta un posible diseño de la carga:



En la URL <http://<IPDockerContainer>:3000/status> se puede monitorizar el estado de carga del servidor NodeJS.

## RESUMEN DEL PROCESO

En esta práctica debemos utilizar JMeter como herramienta para hacer pruebas de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web.

Generaremos un test sobre una aplicación que ejecuta dos contenedores, uno para base de datos y otro para una aplicación en sí.

El ejercicio consiste en crear un test de carga de una API Rest empleando JMeter. El servidor se distribuye en forma de una aplicación de contenedores Docker sobre Compose. Para ello debemos instalar Docker Community Edition y Docker Compose.

## INSTALACIÓN DE DOCKER EN UBUNTU

Seguimos los pasos indicados previamente en la descripción del ejercicio para instalar docker en Ubuntu Server. En este caso, vamos a instalar docker en Ubuntu Server, por lo tanto consultamos la documentación de instalación de docker en Ubuntu Server. Configuramos el repositorio e instalamos docker engine.

- <https://docs.docker.com/engine/install/ubuntu/>



```
UbuntuServer (SSH y Zabbix) [Running]
root@ubuntusv:~# sudo 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/
```

**Figura 1.** Comprobamos que la instalación de docker ha sido satisfactoria.

```
UbuntuServer (SSH y Zabbix) [Running]
root@ubuntusv:~# systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2020-12-28 16:20:20 UTC; 5min ago
TriggeredBy: • docker.socket
    Docs: https://docs.docker.com
   Main PID: 3482 (dockerd)
      Tasks: 10
     Memory: 106.5M
      CGroup: /system.slice/docker.service
              └─3482 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Dec 28 16:20:20 ubuntusv dockerd[3482]: time="2020-12-28T16:20:20.177245710Z" level=info msg="Loadi>
Dec 28 16:20:20 ubuntusv dockerd[3482]: time="2020-12-28T16:20:20.325641053Z" level=info msg="Defau>
Dec 28 16:20:20 ubuntusv dockerd[3482]: time="2020-12-28T16:20:20.547833696Z" level=info msg="Loadi>
Dec 28 16:20:20 ubuntusv dockerd[3482]: time="2020-12-28T16:20:20.657331466Z" level=info msg="Docke>
Dec 28 16:20:20 ubuntusv dockerd[3482]: time="2020-12-28T16:20:20.657999103Z" level=info msg="Daemo>
Dec 28 16:20:20 ubuntusv systemd[1]: Started Docker Application Container Engine.
Dec 28 16:20:20 ubuntusv dockerd[3482]: time="2020-12-28T16:20:20.712097615Z" level=info msg="API l>
Dec 28 16:23:49 ubuntusv dockerd[3482]: time="2020-12-28T16:23:49.278090386Z" level=info msg="ignor>
Dec 28 16:23:59 ubuntusv dockerd[3482]: time="2020-12-28T16:23:59.882210795Z" level=info msg="ignor>
Dec 28 16:24:13 ubuntusv dockerd[3482]: time="2020-12-28T16:24:13.819207342Z" level=info msg="ignor>
lines 1-21/21 (END)
```

**Figura 2.** Comprobamos el status de docker, para ver si está “running”.

## INSTALACIÓN DE DOCKER-COMPOSE EN UBUNTU

De la misma manera que en la instalación de docker en Ubuntu Server, seguimos los pasos de la documentación de docker relacionada. Toda la información necesaria la tenemos en el enlace siguiente:

- <https://docs.docker.com/compose/install/>

```
UbuntuServer (SSH y Zabbix) [Running]
root@ubuntusv:~# docker-compose --version
docker-compose version 1.27.4, build 40524192
```

**Figura 3.** Comprobamos que docker-compose está instalado correctamente.

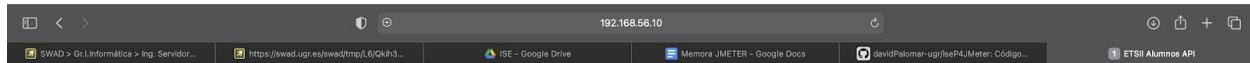
## INSTALACIÓN DE APLICACIÓN PARA TEST CON JMETER EN UBUNTU

Clonamos el repositorio de Github y accedemos al directorio llamado iseP4JMeter. Lo levantamos con docker-compose con el comando up

```
// Para levantar la aplicación a ejecutar
docker-compose up
```

```
// Para parar la aplicación a ejecutar  
docker-compose down
```

Ejecutamos `docker-compose up` y si accedemos al navegador con la dirección `http://<IPDockerContainer>:3000` se presenta la descripción básica de la API, como previamente he expuesto en la introducción.



#### ETSII Alumnos API

```
Descripción de la API Restful:  
POST /api/v1/auth/login  
Parametros:  
    login:<emailUsuario>  
    password:<secreto>  
Seguridad:  
    Acceso protegido con BasicAuth (etsiiApi:laApiDeLaTSIIDaLache)  
Retorna:  
    JWT Token  
  
GET /api/v1/alumnos/alumno/<email>  
Seguridad:  
    Token JWT valido en cabecera estandar authorization: Bearer <token>  
    Alumnos solo pueden solicitar sus datos. Administradores pueden solicitar cualquier alumno valido  
Retorna:  
    Objeto Json con perfil de alumno
```

Figura 4. Descripción básica de la API tras ejecutar docker-compose up.

## INSTALACIÓN DE JMETER EN HOST

Nos dirigimos a la página oficial de JMeter: <http://jmeter.apache.org/> y en el apartado downloads descargamos la última versión disponible para nuestro sistema operativo anfitrión.

Descomprimimos el archivo comprimido y ejecutamos JMeter desde `./bin/ApacheJMeter.jar`

Aquí tenemos la app una vez la iniciamos:

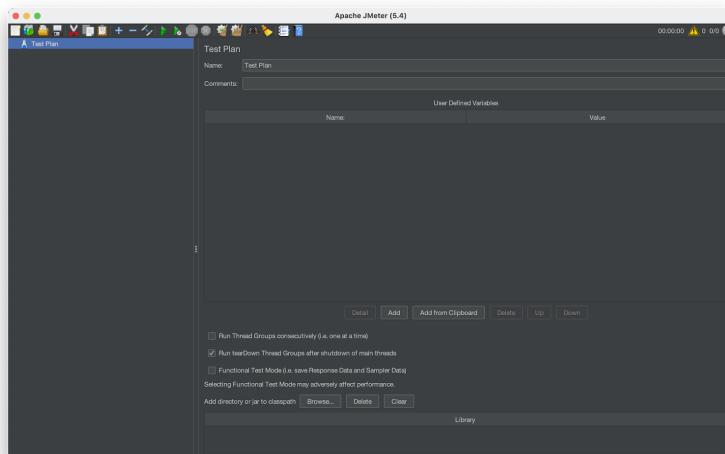


Figura 5. JMeter por defecto.

# CONFIGURACIÓN DEL TEST

## 1. PARAMETRIZAMOS EL HOST Y EL PUERTO

- Host → <IPDockerContainer> -- en mi caso 192.168.56.10
- Puerto → 3000

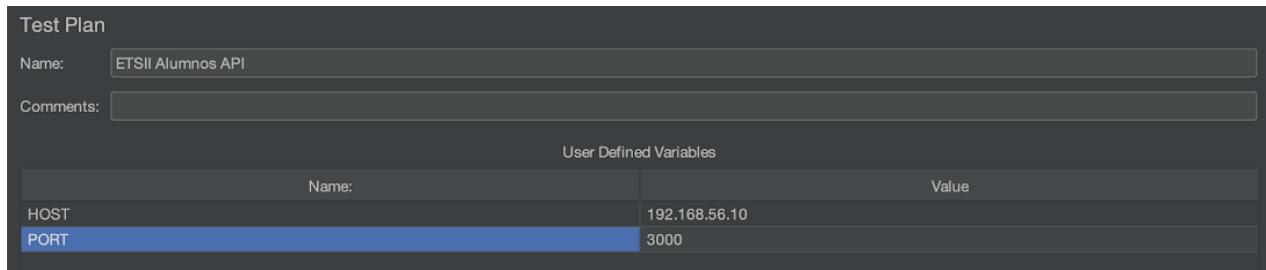


Figura 6. Configuración de los parámetros host y puerto.

## 2. CREAMOS DOS GRUPOS DE HEBRAS DISTINTOS PARA SIMULAR EL ACCESO DE ALUMNOS Y ADMINISTRADORES A LA API

Para ello, seleccionamos [Edit > Add > Threads \(users\) > Thread Group](#).

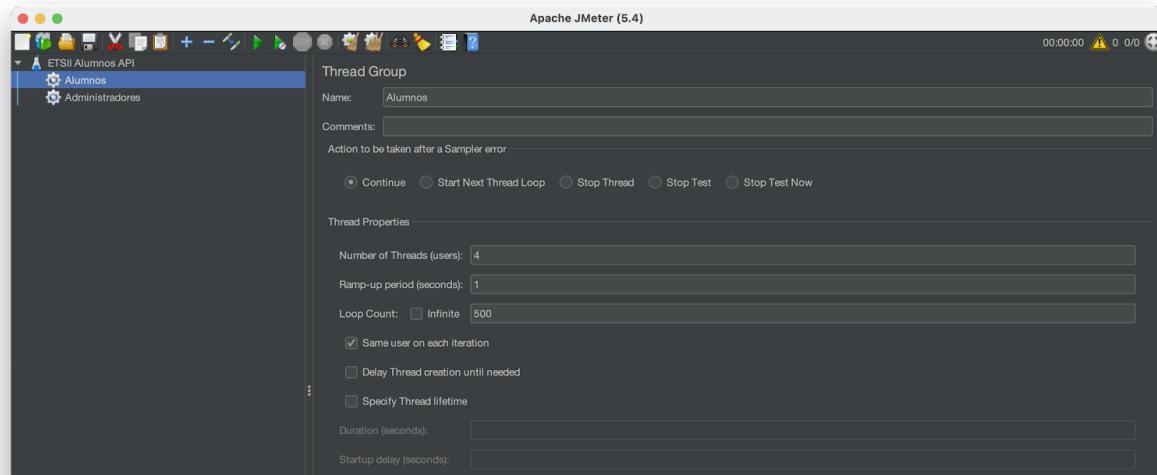


Figura 7. Creación de grupo de hebras alumnos y administradores.

### 3. GENERAMOS LA PETICIÓN HTTP

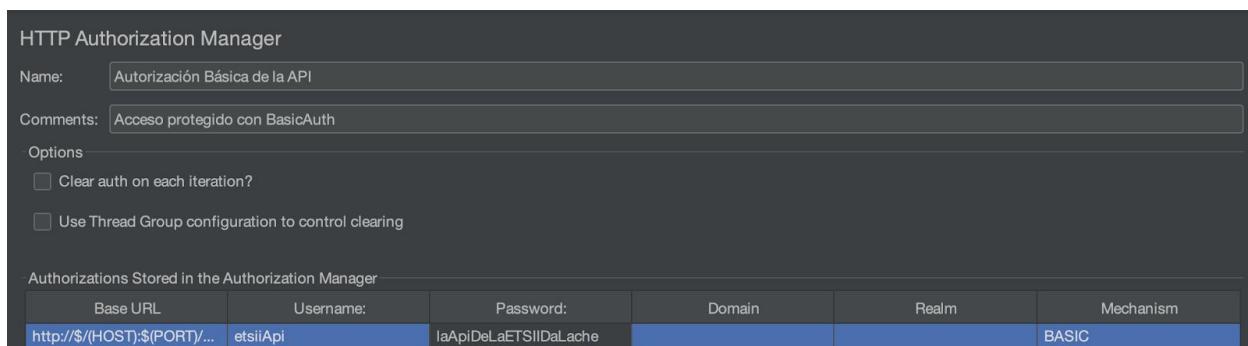
Para ello insertamos el nombre del servidor, puerto y ruta para la petición HTTP. Vamos a [Edit > Add > Config element > HTTP Request Defaults](#).



**Figura 8.** Generamos la petición HTTP con los parámetros previamente declarados.

### 4. AÑADIMOS LA AUTORIZACIÓN

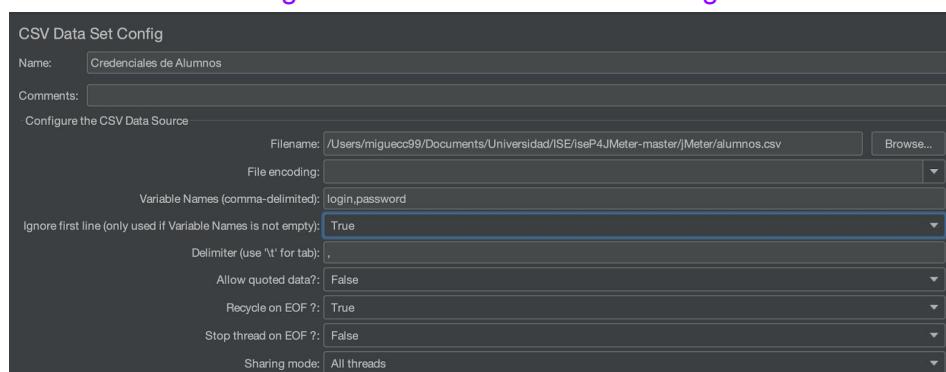
La contraseña es: `laApiDeLaETSIIDaLache`. Para ello vamos a [Edit > Add > Config element > HTTP Authorization Manager](#).



**Figura 9.** Añadimos la autorización básica de la API.

### 5. CONFIGURACIÓN DE CREDENCIALES ALUMNO

El nombre del archivo es CSV de alumnos en el directorio jMeter de la práctica. Para ello vamos a [Alumnos > Add > Config element > CSV Data Set Config](#).



**Figura 10.** Configuramos las credenciales.

## 6. AÑADIMOS LOGIN DE ALUMNOS

Añadimos el login de los alumnos (procesamiento de usuario y contraseña). Cambiamos el método (POST), añadimos la ruta y configuramos los parámetros de entrada (login y password). Para ello vamos a [Alumnos > Add > Sampler > HTTP Request](#).

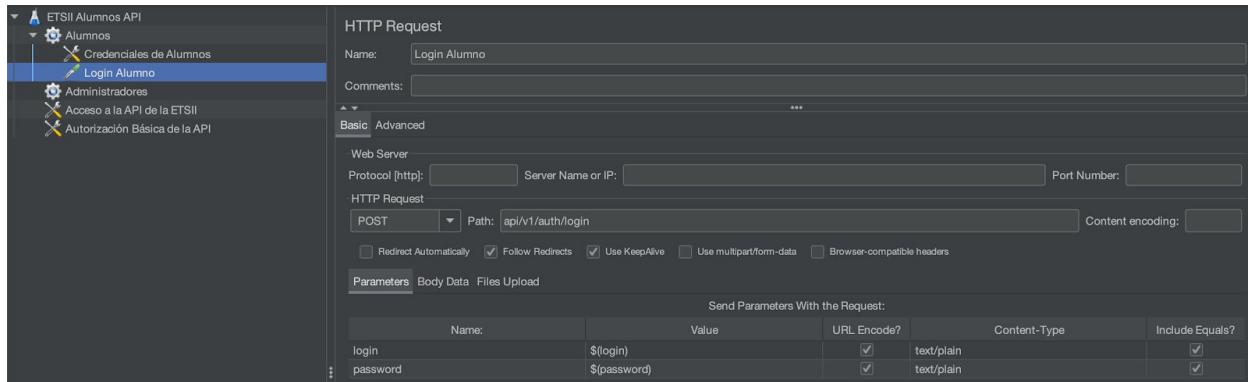


Figura 11. Login Alumnos.

## 7. EXTRACCIÓN DEL TOKEN DEL LOGIN ALUMNO

Extraemos el token del login del Alumno. Para ello nos dirigimos a [Alumnos > Login > Add > Post Processors > Regular Expression Extractor](#).

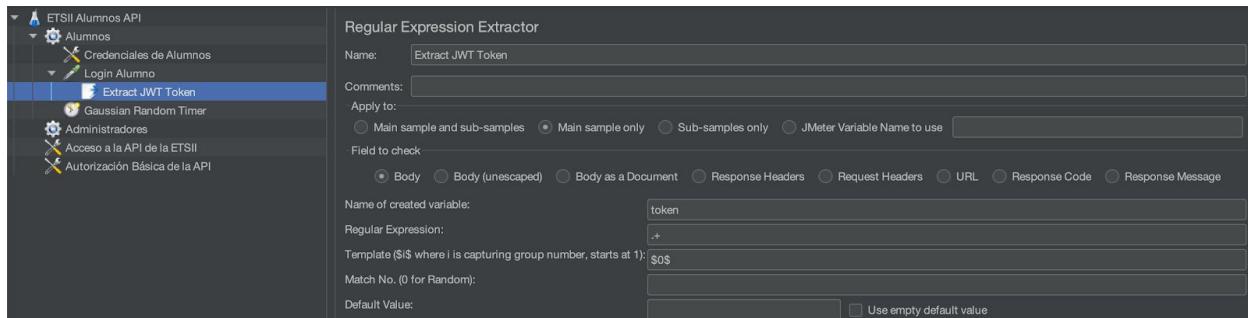


Figura 12. Extracción del token del Login Alumno.

## 8. AÑADIMOS TEMPORIZADOR

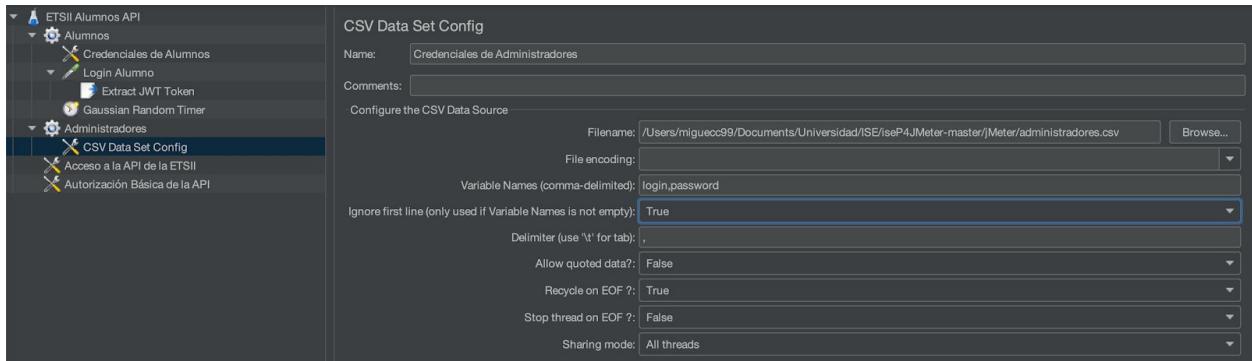
Nos dirigimos a [Alumnos > Login Alumno > Add > Timer > Gaussian Random Timer](#).



Figura 13. Gaussian Random Timer.

## 9. CONFIGURACIÓN DE CREDENCIALES ADMINISTRADOR

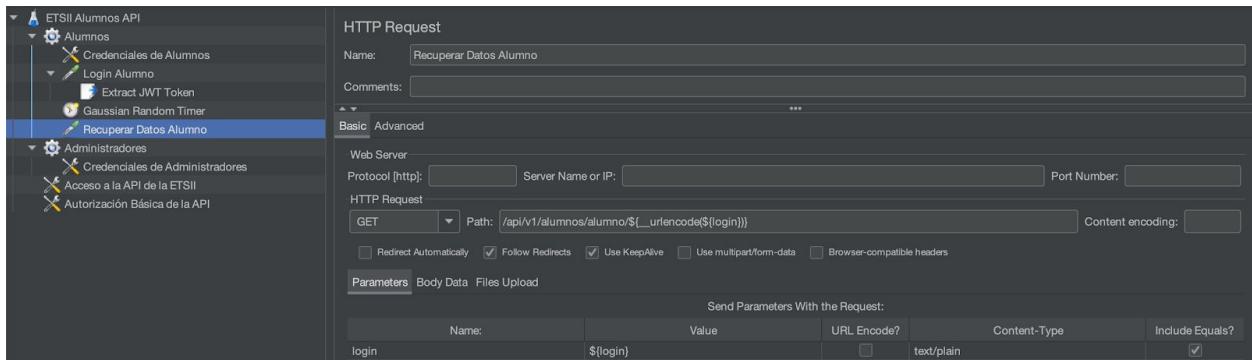
Hacemos la configuración de credenciales de los administradores. El nombre de archivo es el CSV de administradores del directorio jMeter de la P4. Hay que prestar especial atención en la configuración de campos. Nos dirigimos a **Administradores > Add > Config Element > CSV Data Set Config**.



**Figura 14.** Configuración de credenciales Administrador.

## 10. RECUPERACIÓN DE DATOS DEL ALUMNO

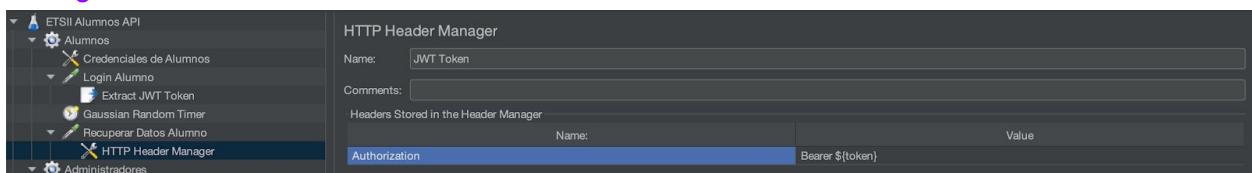
Añadimos a la petición HTTP la ruta y el parámetro login. Campos a **Alumnos > Add > Sampler > HTTP Request**.



**Figura 15.** HTTP Request para recuperar datos del alumno.

## 11. GESTOR DE CABECERA DE HTTP

Usamos el HTTP Header Manager: al autenticarnos con un nombre de dominio, si la app nos redirige a otro nombre de dominio, el token ya no serviría, y esto resuelve el problema. Nos dirigimos a **Alumnos > Recuperar Datos Alumno > Add > Config element > HTTP Header Manager**.



**Figura 16.** HTTP Header Manager.

## 12. LOGIN ADMINISTRADORES

Cambiamos el método (POST), añadimos la ruta y configuramos los parámetros de entrada (login y password). Nos dirigimos a Administradores > Add > Sampler > HTTP Request.

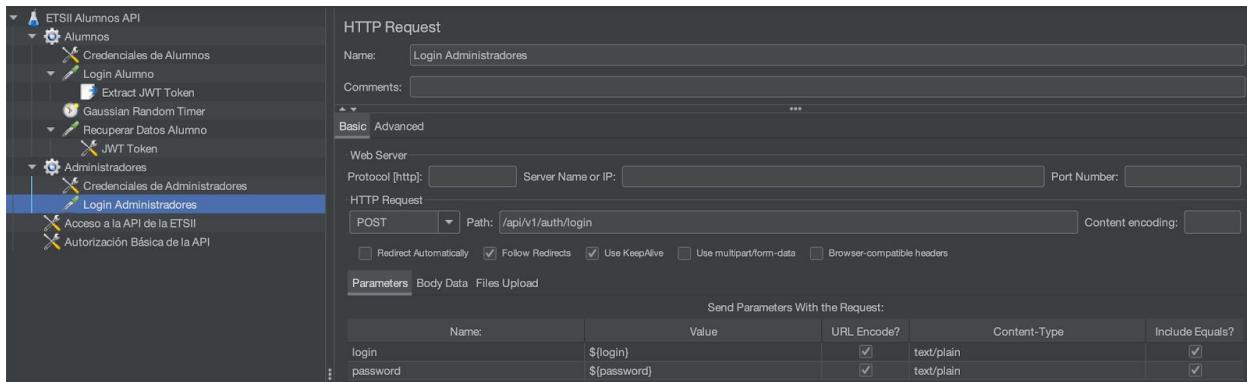


Figura 17. Login Administradores.

## 13. EXTRAEMOS EL TOKEN DEL LOGIN ADMINISTRADOR

Nos dirigimos a Administradores > Login > Add > Post Processors > Regular Expression Extractor.

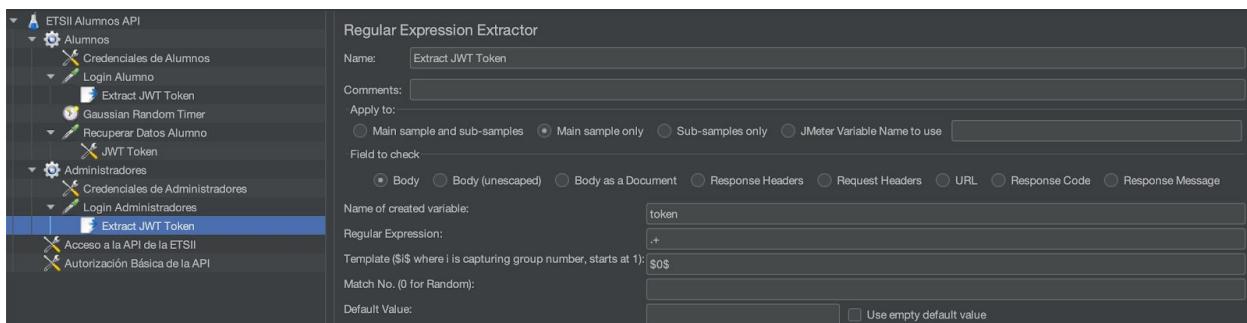


Figura 18. Extracción del token del login de administradores.

## 14. MUESTREADOR DE ACCESO A LOG

Muestrea el archivo .log de accesos al sistema. Añadimos el servidor, puerto y ubicación del archivo de Log(apiAlumnos.log en el directorio JMeter de la P4). Para ello nos dirigimos a Administradores > Add > Sampler > Log Access Sampler.

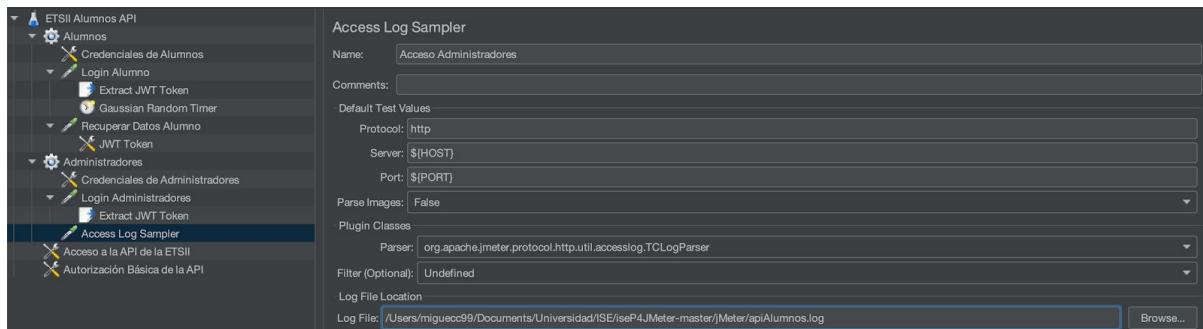


Figura 19. Muestreador del acceso a log.

## 15. GESTOR DE CABECERA HTTP

Al autenticarnos con un nombre de dominio, si la aplicación nos redirige a otro nombre de dominio, el token ya no serviría, y esto nos lo resuelve. Para ello vamos a [Administradores > Acceso Administradores > Add > Config element > HTTP Header Manager](#).

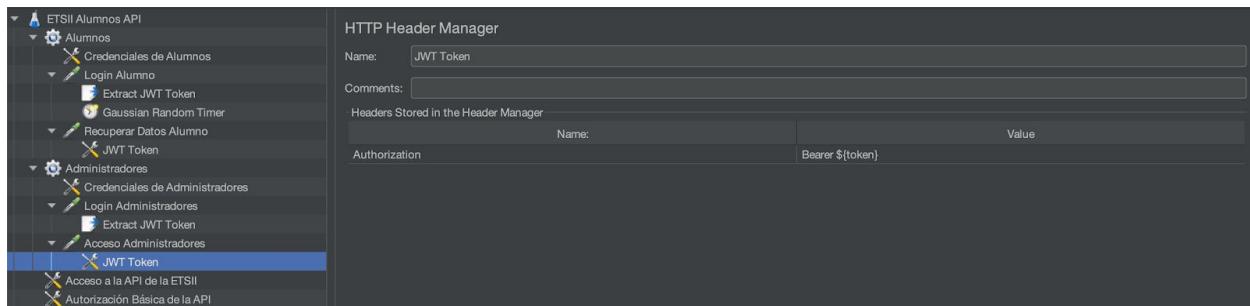


Figura 20. Gestor de cabecera HTTP.

## 16. AÑADIMOS UN TEMPORIZADOR GAUSSIANO

Nos vamos a [Administradores > Acceso Administradores > Timer > Gaussian Random Timer](#).

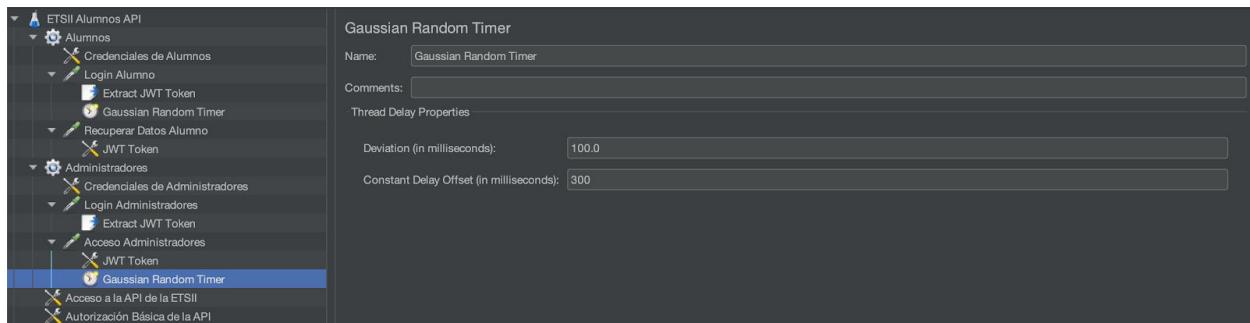


Figura 21. Temporizador Gaussiano.

## 17. SUMMARY REPORT, VIEW RESULTS TREE & AGGREGATE REPORT

Agregamos lo siguiente para ver el resultado del test:

- [Add > Listener > Summary Report](#)
- [Add > Listener > View Results Tree](#)
- [Add > Listener > Aggregate Report](#)

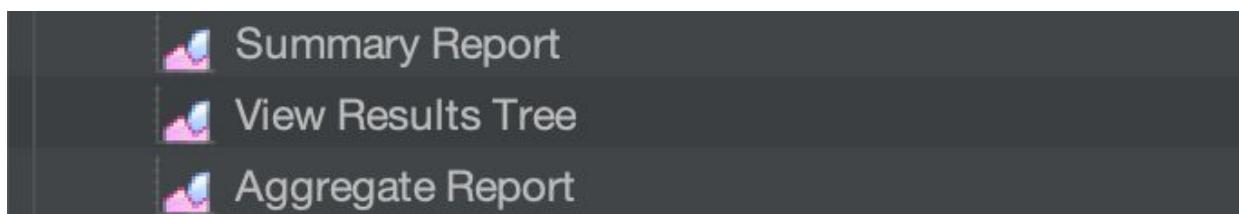


Figura 22. Listeners.

## 18. TEST FRAGMENT

Add > Test Fragment > Test Fragment

Add > Listener > View Results Tree

Add > Sampler > Debug Sampler

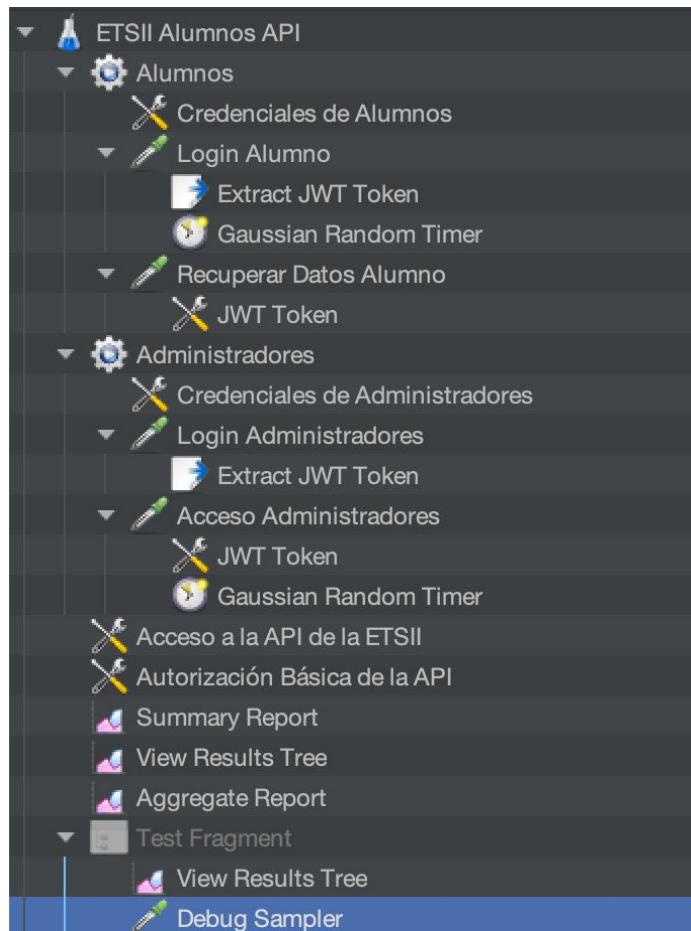


Figura 22. Árbol final del test sobre la API Rest.

## APLICACIÓN DEL TEST SOBRE LA API REST

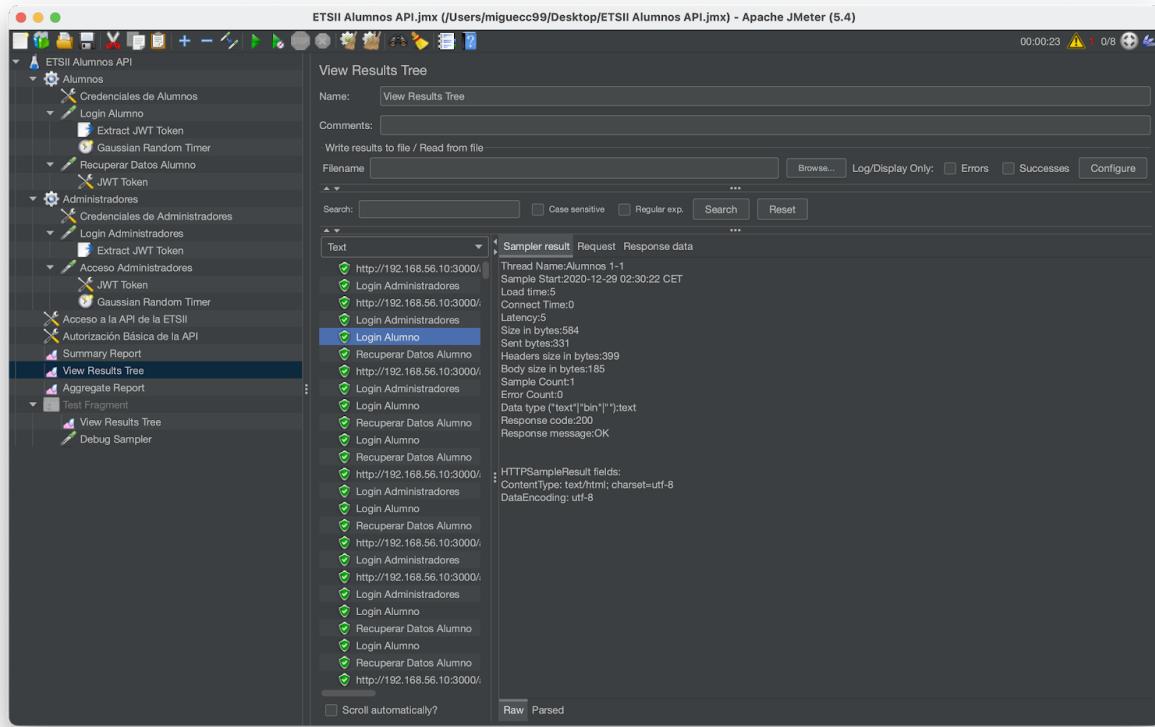
1. Una vez configurado el test, iniciamos docker en Ubuntu Server con la orden:

```
docker-compose up
```

2. En la app de JMeter arrancamos el test con el botón de play verde.

Una vez hecho esto, dejamos al test tomando datos durante una serie de segundos prudente y paramos el test para comprobar los resultados obtenidos.

En el árbol de resultados podemos comprobar como se han realizado correctamente todas las operaciones (tanto de login, como de conexión al servidor y recuperación de datos).



**Figura 23.** Árbol de resultados.

Esta información también se puede ver en forma de tabla en Summary Report, donde vemos que el porcentaje de error en todos los casos es del 0.00%

Summary Report										
Name:	Summary Report									
Comments:										
Write results to file / Read from file										
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Login Administradores	272	8	3	204	13.30	0.00%	11.8/sec	6.88	3.86	595.0
http://192.1... Login Alumno	287	7	3	78	7.34	0.00%	12.7/sec	7.27	4.15	583.6
Recuperar Datos Alumno	287	7	3	43	5.77	0.00%	12.8/sec	17.75	5.09	1422.0
http://192.1... Login Administradores	4	15	6	39	13.48	0.00%	6.2/sec	7.27	0.00	1206.0
http://192.1... Login Alumno	4	7	6	9	1.09	0.00%	5.8/sec	8.69	0.00	1539.0
http://192.1... Recuperar Datos Alumno	4	8	4	16	4.72	0.00%	4.6/sec	8.81	0.00	1978.0
http://192.1... Login Administradores	4	7	5	14	3.77	0.00%	4.1/sec	4.17	0.00	1043.0
http://192.1... Login Alumno	4	9	4	16	5.31	0.00%	3.6/sec	5.46	0.00	1542.0
http://192.1... Recuperar Datos Alumno	4	4	4	5	0.43	0.00%	4.4/sec	4.86	0.00	1126.0
http://192.1... Login Administradores	4	9	5	19	5.79	0.00%	4.0/sec	3.72	0.00	950.0
http://192.1... Login Alumno	4	7	5	13	3.34	0.00%	4.0/sec	5.40	0.00	1379.0
http://192.1... Recuperar Datos Alumno	4	4	4	5	0.43	0.00%	3.5/sec	3.62	0.00	1050.0
http://192.1... Login Administradores	4	8	6	15	3.77	0.00%	3.4/sec	3.31	0.00	999.0
http://192.1... Login Alumno	4	6	5	8	1.22	0.00%	3.0/sec	4.54	0.00	1565.0
http://192.1... Recuperar Datos Alumno	4	7	4	11	2.55	0.00%	2.7/sec	2.94	0.00	1120.0
http://192.1... Login Administradores	4	11	5	19	5.41	0.00%	2.7/sec	5.49	0.00	2109.0
http://192.1... Login Alumno	4	8	5	14	3.70	0.00%	3.0/sec	3.31	0.00	1143.0
http://192.1... Recuperar Datos Alumno	4	6	5	7	0.87	0.00%	3.6/sec	5.20	0.00	1479.0
http://192.1... Login Administradores	4	7	5	13	3.34	0.00%	4.7/sec	4.42	0.00	964.0
http://192.1... Login Alumno	4	7	5	13	3.27	0.00%	6.5/sec	8.07	0.00	1275.0
http://192.1... Recuperar Datos Alumno	4	7	5	14	3.70	0.00%	5.1/sec	5.05	0.00	1005.0
http://192.1... Login Administradores	4	9	6	14	3.11	0.00%	6.1/sec	9.51	0.00	1592.0
http://192.1... Login Alumno	4	7	4	11	3.04	0.00%	4.8/sec	6.90	0.00	1472.0
http://192.1... Recuperar Datos Alumno	4	5	4	7	1.09	0.00%	4.7/sec	10.10	0.00	2211.0
http://192.1... Login Administradores	4	11	5	19	5.07	0.00%	5.2/sec	8.03	0.00	1580.0
http://192.1... Login Alumno	4	7	4	15	4.39	0.00%	4.9/sec	8.20	0.00	1723.0
http://192.1... Recuperar Datos Alumno	4	7	6	10	1.73	0.00%	4.6/sec	6.11	0.00	1368.0

**Figura 24.** Summary Report.

## EJECUCIÓN DEL TEST MEDIANTE COMANDO TERMINAL

En la P4 se nos pide que la prueba de carga se ejecute mediante un comando en terminal (sin consola gráfica) desde la máquina host. Para hacer esto, he hecho uso del comando siguiente:

```
jmeter -n -t ETSIIAlumnosAPI.jmx -l resultados.jtl
```

Este comando vuelca los resultados de la ejecución de la prueba de carga ETSIIAlumnosAPI.jmx.

Si hacemos “cat” del fichero, podemos ver como jmeter vuelca todos los datos referentes a la prueba de carga.

**Figura 25.** resultados.jtl.