

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

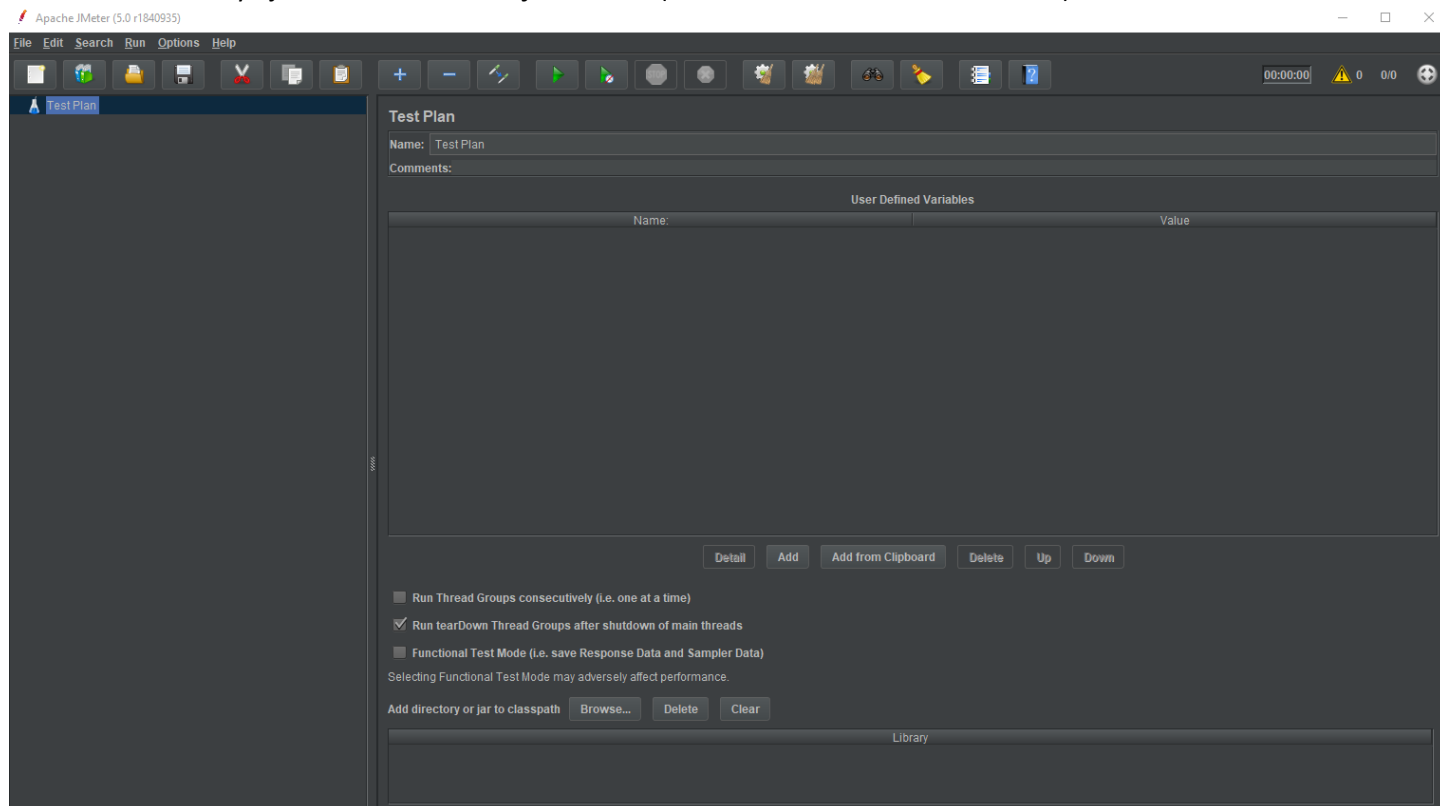
cd iseP4JMeter

docker-compose up

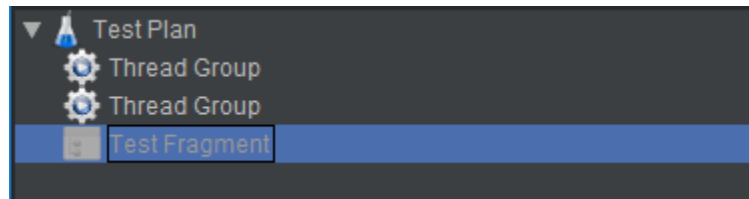
```
node.js_1 | GET / 304 48.057 ms - -
node.js_1 | GET / 304 1.256 ms - -
node.js_1 | GET /stylesheets/style.css 304 2.484 ms - -
node.js_1 | GET /stylesheets/style.css 304 0.539 ms - -
node.js_1 | GET /favicon.ico 404 3.497 ms - 41
```

Creación del test en JMeter

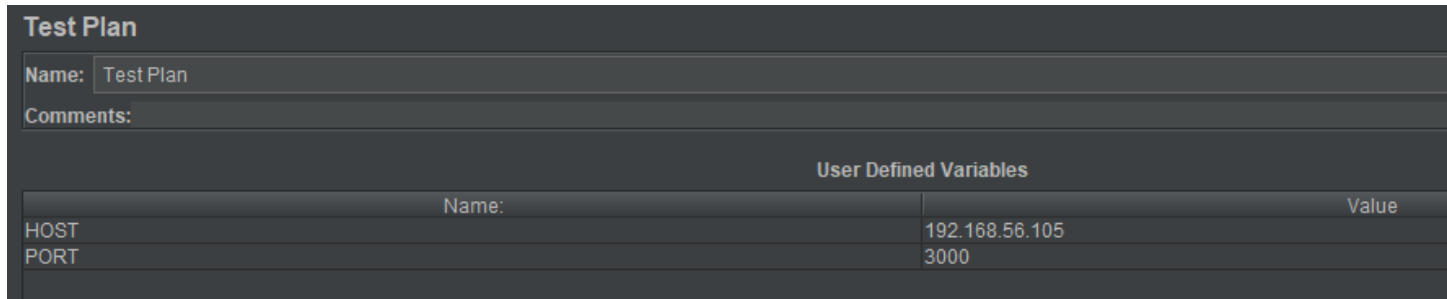
- Instalamos JMeter y ejecutamos el archivo **jmeter.bat** (localizado en el subdirectorio **bin**)



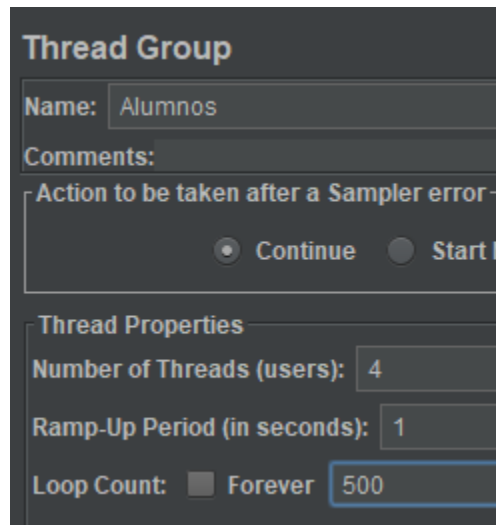
- Creamos un **Test Plan** con dos **Thread Group** y un **Test Fragment**



- 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.

HTTP Request Defaults

Name: Access to ETSIIT API

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: \${HOST} Port Number: \${PORT}

HTTP Request

Path: /api/v1 Content encoding:

Parameters Body Data

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include

- ETSIIT Alumnos API
- Access to ETSIIT API
- HTTP Authorization Manager**
- Alumnos
- Administradores
- WorkBench Test Fragment

HTTP Authorization Manager

Name: Basic-Auth API

Comments:

Options

☐ Clear auth on each iteration?

Authorizations Stored in the Authorization Manager

Base URL	Username	Password	Domain	Realm	Mechanism
http://\${HOST}:\${PORT}/api/v1/auth/login	etsiitApi	laApiDeLaETSIITDaLache			BASIC

- 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).

CSV Data Set Config

Name: Credenciales Alumnos

Comments:

Configure the CSV Data Source

Filename: E:/Nacho/Desktop/Año 3/Primer Cuatrimestre/Ingeniería de Servidores/Practicas/P4/jMeter/alumnos.csv

File encoding:

Variable Names (comma-delimited): login,password

Ignore first line (only used if Variable Names is not empty): True

Delimiter (use '\t' for tab): ,

CSV Data Set Config

Name: Credenciales Administradores

Comments:

Configure the CSV Data Source

Filename: E:/Nacho/Desktop/Año 3/Primer Cuatrimestre/Ingeniería de Servidores/Practicas/P4/jMeter/administradores.csv

File encoding:

Variable Names (comma-delimited): login,password

Ignore first line (only used if Variable Names is not empty): True

Delimiter (use '\t' for tab): ,

- 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.

HTTP Request

Name: Login Alumno

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: \${HOST} Port Number: \${PORT}

HTTP Request

Method: POST Path: api/v1/auth/login Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equ
login	\${login}	<input checked="" type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	\${password}	<input checked="" type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

HTTP Request

Name: Recuperar Datos Alumnos

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: \${HOST} Port Number: \${PORT}

HTTP Request

Method: GET Path: /api/v1/alumnos/alumno/\${login} Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

HTTP Request

Name: Login Administradores

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: \${HOST} Port Number: \${PORT}

HTTP Request

Method: POST Path: api/v1/auth/login Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equ
login	\${login}	<input checked="" type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	\${password}	<input checked="" type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

- 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**.

Regular Expression Extractor

Name: Extract JWT Token

Comments:

Apply to: ☐ Main sample and sub-samples ☒ Main sample only ☐ Sample

Field to check: ☒ Body ☐ Body (unescaped) ☐ Body as a Document

Name of created variable: token

Regular Expression: .+

Template (\$i\$ where i is capturing group number, starts at 1): \$0\$

- 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.

Access Log Sampler

Name: Acceso Administradores

Comments: Se podría poner Parse Images para descargar las imágenes

Default Test Values

Protocol: http

Server: \${HOST}

Port: \${PORT}

Parse Images: False

Plugin Classes

Parser: org.apache.jmeter.protocol.http.util.accesslog.TCLogParser

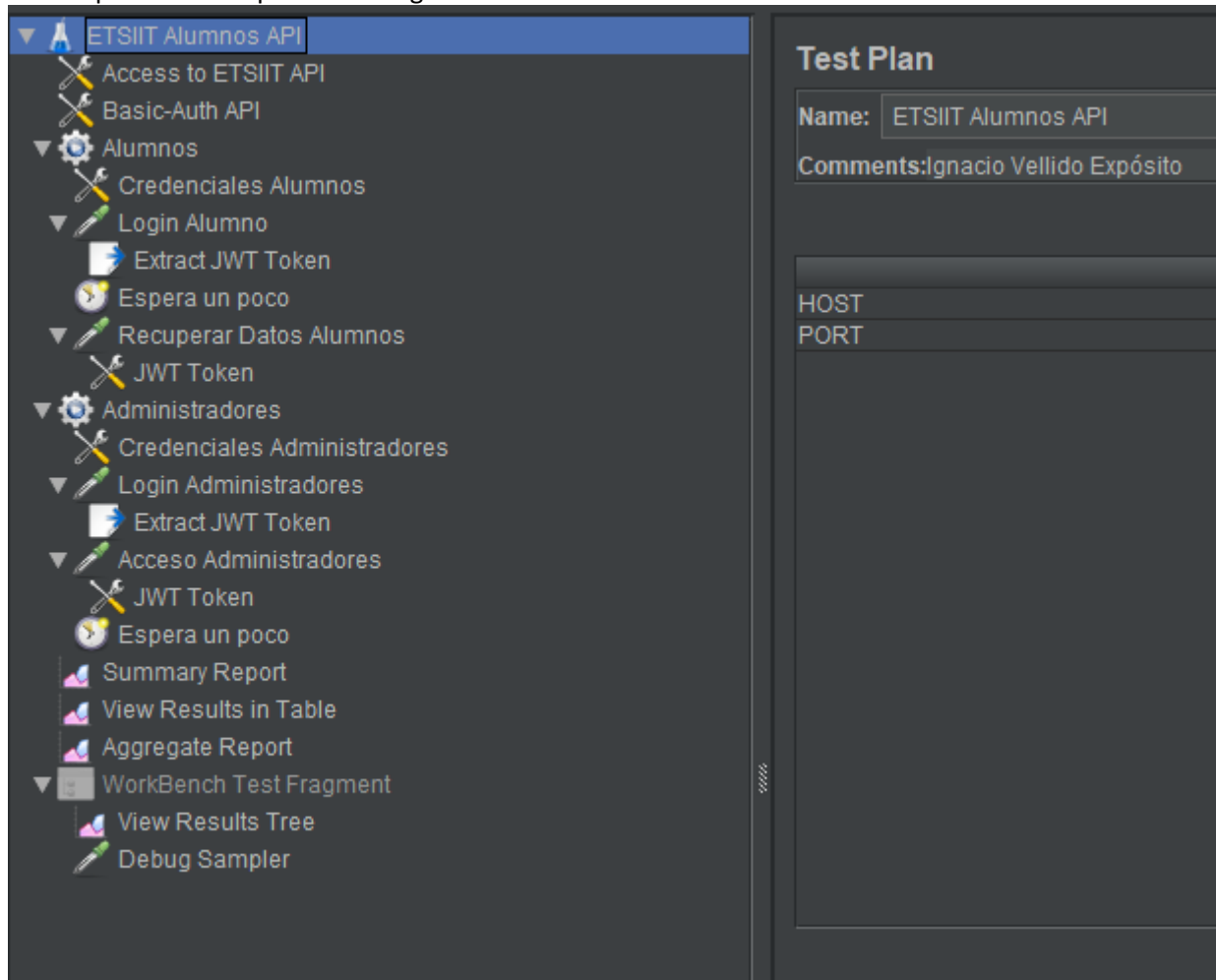
Filter (Optional): Undefined

Log File Location

Log File: E:/Nacho/Desktop/Año 3/Primer Cuatrimestre/Ingeniería de Servidores/Practicas/P4/jMeter/apiAlumnos.log

- 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:

The screenshot displays two JMeter listeners: 'View Results Tree' and 'View Results in Table'.

View Results Tree: This listener shows a list of test results with green checkmarks indicating successful execution. The list includes:

- Login Alumno
- Login Administradores
- Login Administradores
- Login Administradores
- Login Alumno
- Login Administradores
- Login Alumno
- http://192.168.56.105:3000
- Recuperar Datos Alumnos
- http://192.168.56.105:3000
- Recuperar Datos Alumnos
- Recuperar Datos Alumnos
- http://192.168.56.105:3000
- Login Administradores
- Login Alumno
- Login Administradores
- Login Alumno
- Login Administradores
- Login Alumno
- Login Administradores
- http://192.168.56.105:3000

View Results in Table: This listener displays a table of test results. The table has the following columns: Sample #, Start Time, Thread Name, Label, Sample Time(ms), Status, Bytes, Sent Bytes, Latency, and Connect Time(ms). The table contains 18 rows of data, all with a status of 'Success' (indicated by a green checkmark).

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	11:10:42.466	Administradores ...	Login Administra...	875	Success	595	338	875	54
2	11:10:42.466	Alumnos 1-1	Login Alumno	876	Success	582	345	876	54
3	11:10:42.466	Administradores ...	Login Administra...	884	Success	595	336	884	54
4	11:10:42.969	Administradores ...	Login Administra...	382	Success	595	338	382	3
5	11:10:42.493	Alumnos 1-2	Login Alumno	860	Success	582	332	860	27
6	11:10:42.779	Administradores ...	Login Administra...	575	Success	595	337	575	1
7	11:10:42.571	Alumnos 1-3	Login Alumno	789	Success	582	335	789	1
8	11:10:42.835	Alumnos 1-4	Login Alumno	525	Success	582	335	525	1
9	11:10:43.538	Administradores ...	http://192.168.56...	208	Success	1141	0	208	0
10	11:10:43.563	Alumnos 1-3	Recuperar Datos ...	190	Success	1983	376	190	0
11	11:10:43.573	Administradores ...	http://192.168.56...	180	Success	1141	0	180	0
12	11:10:43.626	Administradores ...	http://192.168.56...	128	Success	1141	0	128	0
13	11:10:43.596	Alumnos 1-4	Recuperar Datos ...	164	Success	1055	376	164	0
14	11:10:43.629	Alumnos 1-1	Recuperar Datos ...	131	Success	1048	379	131	0
15	11:10:43.689	Alumnos 1-2	Recuperar Datos ...	72	Success	1547	374	72	0
16	11:10:43.748	Administradores ...	http://192.168.56...	20	Success	1141	0	20	0
17	11:10:43.899	Administradores ...	Login Administra...	35	Success	595	334	35	4
18	11:10:43.899	Alumnos 1-2	Login Alumno	41	Success	583	336	41	4

```

node.js_1 | GET /api/v1/alumnos/alumno/cervantesadams%40tropoli.com 200 9.735 ms - 804
node.js_1 | GET /api/v1/alumnos/alumno/woodardoneill@tropoli.com 200 2.597 ms - 1076
node.js_1 | GET /api/v1/alumnos/alumno/mariweiss@tropoli.com 200 2.308 ms - 1162
node.js_1 | POST /api/v1/auth/login 200 3.102 ms - 196
node.js_1 | GET /api/v1/alumnos/alumno/cervantesadams%40tropoli.com 200 3.147 ms - 804
node.js_1 | GET /api/v1/alumnos/alumno/maribelwhite%40tropoli.com 200 2.860 ms - 1136
node.js_1 | GET /api/v1/alumnos/alumno/staceytownsend@tropoli.com 200 2.662 ms - 603
node.js_1 | GET /api/v1/alumnos/alumno/cervantesadams%40tropoli.com 200 12.254 ms - 804
node.js_1 | POST /api/v1/auth/login 200 11.444 ms - 184
node.js_1 | POST /api/v1/auth/login 200 6.031 ms - 196
node.js_1 | POST /api/v1/auth/login 200 3.033 ms - 184
node.js_1 | POST /api/v1/auth/login 200 6.588 ms - 184
node.js_1 | GET /api/v1/alumnos/alumno/maribelwhite@tropoli.com 200 7.777 ms - 1136
node.js_1 | POST /api/v1/auth/login 200 14.203 ms - 196
node.js_1 | GET /api/v1/alumnos/alumno/deborawalker@tropoli.com 200 16.382 ms - 739
node.js_1 | GET /api/v1/alumnos/alumno/maribelwhite%40tropoli.com 200 3.310 ms - 1136
node.js_1 | GET /api/v1/alumnos/alumno/velezguthrie%40tropoli.com 200 3.021 ms - 1575
node.js_1 | POST /api/v1/auth/login 200 3.573 ms - 196
node.js_1 | GET /api/v1/alumnos/alumno/soliswest@tropoli.com 200 3.350 ms - 592
node.js_1 | GET /api/v1/alumnos/alumno/cervantesadams@tropoli.com 200 3.469 ms - 804
node.js_1 | POST /api/v1/auth/login 200 9.540 ms - 184
node.js_1 | GET /api/v1/alumnos/alumno/maribelwhite%40tropoli.com 200 4.952 ms - 1136
node.js_1 | POST /api/v1/auth/login 200 4.226 ms - 183

```

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>

Subdirectorío 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>