

ISE Práctica 4

Daniel Alconchel Vázquez

Ejercicio 1: Phoronix

Si nos vamos a la página que se nos indica en el guión, [Phoronix](#), nos aparece, nada más cargar la página, las opciones **Learn About OpenBenchmarking** y **Learn About the Phoronix Test Suite**.

Podemos ver que tenemos dos opciones:

- Descarga de *Tests Individuales*
- Descarga de *Suites*: Programa Open-Source que contiene una gran cantidad de test y se encarga del proceso de descarga, instalación, dependencias, ejecución, etc... de los mismos.

Vamos a comenzar con **Ubuntu Server**

La documentación que viene en la [página]([Phoronix Test Suite - Download](#)), te indica como instalarlo a través de paquetes o docker. Para instalarlo manualmente desde terminal, he mirado la siguiente [web](#).

Para copiar los comandos siguientes, he usado ssh e instalaremos la última versión estable, que es la 10.8:

```
 wget http://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_10.8.3_all.deb  
 sudo dpkg -i phoronix-test-suite_10.8.3_all.deb  
 sudo apt -f install
```

Si ejecutamos el comando

```
 phoronix-test-suite
```

podremos ver todas las opciones que incluye. Algunos comandos interesantes son:

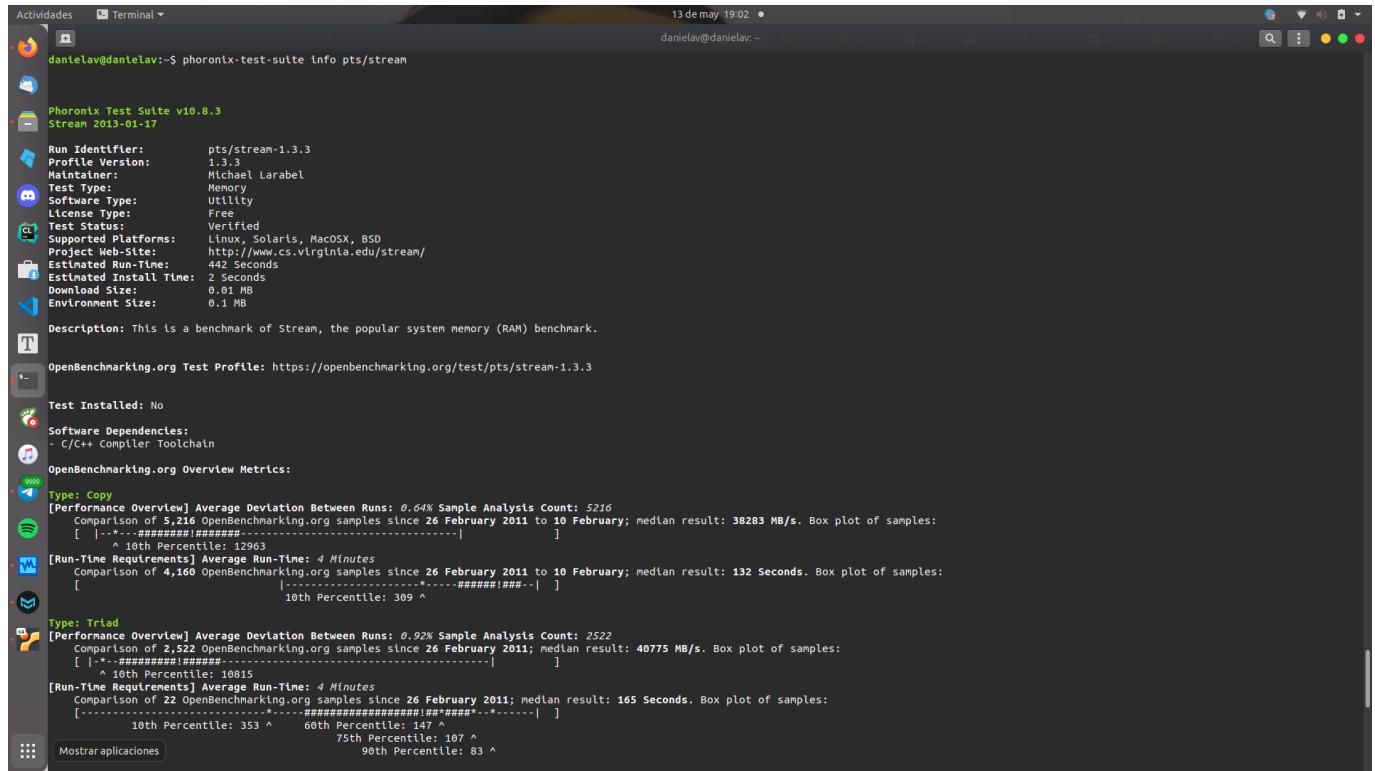
```
# Información del equipo
phoronix-test-suite system-info
# Información de los sensores
phoronix-test-suite system-sensors
# Lista de suites disponibles
phoronix-test-suite list-available-suites
```

Este último comando puede dar error, ya que la información viene comprimida, por lo que necesitaremos `unzip` instalado. Para ello `sudo apt install unzip`.

Si volvemos a ejecutar el último comando, esta vez si nos listará los test-suites disponibles. Vemos que hay distintos tipos. En concreto, los de *gráficos* no funcionarán, ya que hace falta entorno de ventanas. Por lo que optaremos por test de *procesadores y memoria*.

Para ver la información de un test, ejecutamos:

```
phoronix-test-suite info <pts/nombre>
```



The screenshot shows a terminal window titled "Terminal" with the command "phoronix-test-suite info pts/stream" entered. The output displays detailed information about the Stream test suite, including its identifier, version, maintainer, and supported platforms. It also includes a description of the test, which is described as a benchmark of Stream, the popular system memory (RAM) benchmark. The terminal window is part of a desktop environment with icons for various applications like a web browser, file manager, and system monitor.

```
danielav@danielav:~$ phoronix-test-suite info pts/stream
[...]
Run Identifier: pts/stream-1.3.3
Profile Version: 1.3.3
Maintainer: Michael Larabel
Test Type: Memory
Software Type: Utility
License Type: Free
Test Status: Verified
Supported Platforms: Linux, Solaris, Mac OSX, BSD
Project Web-Site: http://www.cs.virginia.edu/stream/
Estimated Run-Time: 442 Seconds
Estimated Install Time: 2 Seconds
Download Size: 0.01 MB
Environment Size: 0.1 MB
Description: This is a benchmark of Stream, the popular system memory (RAM) benchmark.
[...]
OpenBenchmarking.org Test Profile: https://openbenchmarking.org/test/pts/stream-1.3.3
[...]
Test Installed: No
Software Dependencies:
- C/C++ Compiler Toolchain
OpenBenchmarking.org Overview Metrics:
[...]
Type: Copy
[Performance Overview] Average Deviation Between Runs: 0.64% Sample Analysis Count: 5216
Comparison of 5,216 OpenBenchmarking.org samples since 26 February 2011 to 10 February; median result: 38283 MB/s. Box plot of samples:
[ |----#####|----| ]
^ 10th Percentile: 12963
[Run-Time Requirements] Average Run-Time: 4 Minutes
Comparison of 4,160 OpenBenchmarking.org samples since 26 February 2011 to 10 February; median result: 132 Seconds. Box plot of samples:
[ |-----*-----|----| ]
^ 10th Percentile: 309 ^
[...]
Type: Triad
[Performance Overview] Average Deviation Between Runs: 0.92% Sample Analysis Count: 2522
Comparison of 2,522 OpenBenchmarking.org samples since 26 February 2011; median result: 40775 MB/s. Box plot of samples:
[ |----#####|----| ]
^ 10th Percentile: 10815
[Run-Time Requirements] Average Run-Time: 4 Minutes
Comparison of 22 OpenBenchmarking.org samples since 26 February 2011; median result: 165 Seconds. Box plot of samples:
[ -----*-----|----| ]
^ 10th Percentile: 353 ^ 60th Percentile: 147 ^ 75th Percentile: 107 ^ 90th Percentile: 83 ^
```

Todos estos procesos no ocupan memoria, ya que son simples consultas.

He decidido ejecutar el test `pts/php`. Para ejecutar los test tenemos dos opciones:

```
# Bajar y ejecutar el test (todo en un solo comando)
phoronix-test-suite benchmark pts/php
# Bajar y ejecutar el test (por separado)
phoronix-test-suite install pts/php
phoronix-test-suite run pts/php
```

```
danielav@danielav:~$ phoronix-test-suite benchmark pts/php

Phoronix Test Suite v10.8.3

To Install: pts/php-1.0.0

Determining File Requirements .....  
Searching Download Caches ..........
```

1 Test To Install
1 File To Download [14.33MB]
128MB Of Disk Space Is Needed
9 Seconds Estimated Install Time

pts/php-1.0.0:
Test Installation 1 of 1
1 File Needed [14.33 MB / 1 Minute]
Downloading: php-7.0.14-0.b22
Estimated Installation Time: 1m
Appropriate Install Size: 128 MB
Estimated Install Time: 9 Seconds
Installing Test @ 10:17:31

PHP Micro Benchmarks:
pts/php-1.0.0
System Test Configuration
1: Zend bench
2: Zend micro_bench
3: Test All Options
** Multiple items can be selected, delimit by a comma. **
Test: █

```
danielav@danielav: ~
```

DISK:
File-System: 2 x 11GB VBOX HDD
Mount Options: ext4
Disk Scheduler: relative rw
Disk Scheduler: MQ-DEADLINE
Disk Details: Block Size: 4096

OPERATING SYSTEM:
Kernel: Ubuntu 20.04
System Layer: 5.4.0-107-generic (x86_64)
Security: Oracle VMware
itlb_multithit: KVM: Vulnerable
+ l1tf: Mitigation of Page Inversion
+ mds: Mitigation of Cleanammers; SMT Host state unknown
+ meltdown: Mitigation of PTI
+ spec_store_bypass: Vulnerable
+ spectre_v1: Mitigation of usercopy/swaps barriers and __user pointer sanitization
+ spectre_v2: Mitigation of Retpolines STIBP: disabled RSB filling
+ srbd: Unknown: Dependent on hypervisor status
+ tsx_async_abort: Not affected

Would you like to save these test results (Y/n): y
Enter a name for the result file: isetest
Enter a unique name to describe this test run / configuration: tse

If desired, enter a new description below to better describe this result set / system configuration under test.
Press ENTER to proceed without changes.

Current Description: Oracle VMware testing on Ubuntu 20.04 via the Phoronix Test Suite.

New Description:

PHP Micro Benchmarks:
pts/php-1.0.0 [Test: Zend bench]
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 1 Minute [10:19 UTC]
Started Run 1 @ 10:18:43
Started Run 2 @ 10:18:47
Started Run 3 @ 10:18:52

Test: Zend bench:
0.357
0.359
0.358

Average: 0.358 Seconds
Deviation: 0.28K

Comparison of 1,107 OpenBenchmarking.org samples since 23 November 2018; median result: 0.53 Seconds. Box plot of samples:
[----] This Result (87.5% Percentile): 0.358 ^
2 x Intel Xeon E5-2680 0: 2.71 ^ Intel Pentium 4: 1.63 ^ Intel Core i7 950: 1.64 ^ AMD Ryzen 5 2600: 0.7202 ^
Intel Atom x5-Z8350: 2.208 ^ AMD Ryzen 7 2800X: 0.307 ^
ARMv8 Cortex-A72: 2.55 ^ Intel Xeon E3-1275 V6: 0.407 ^
AMD Ryzen 3 2200G: 0.455 ^

Do you want to view the text results of the testing (Y/n):

Para ver los resultados almacenados, podemos ejecutar:

```
phoronix-test-suite list-saved-results  
phoronix-test-suite result-file-to-text isetest
```

```
danielav@danielav:~$ phoronix-test-suite result-file-to-text isetest  
isetest  
Oracle VMWare testing on Ubuntu 20.04 via the Phoronix Test Suite.  
  
lse:  
Processor: Intel Core i7-8750H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX 82441FX PMC, Memory: 1024MB, Disk: 2 x 11GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel 82801  
AA AC 97 Audio, Network: 2 x Intel 82540EM  
OS: Ubuntu 20.04, Kernel: 5.4.0-107-generic (x86_64), File-System: ext4, Screen Resolution: 2048x2048, System Layer: Oracle VMWare  
  
PHP Micro Benchmarks  
Test: Zend bench  
Seconds < Lower Is Better  
lse . 0.358 |=====
```

Vamos a elegir otro test más. En mi caso, he escogido `pts/idle`. Nuevamente, ejecutamos:

```
phoronix-test-suite benchmark pts/idle
```

```
danielav@danielav:~$  
Core Count: 1  
Extensions: SSE 4.2 + AVX2 + AVX + RDRAND + FSGSBASE  
Cache Size: 9 MB  
Core Family: Kaby/Coffee/Whiskey Lake  
  
GRAPHICS: VMware SVGA II  
Screen: 2048x2048  
  
MOTHERBOARD: Oracle VirtualBox v1.2  
BIOS Version: VirtualBox  
Chipset: Intel 440FX 82441FX PMC  
Audio: Intel 82801A AC 97 Audio  
Network: 2 x Intel 82540EM  
  
MEMORY: 1024MB  
  
DISK:  
File-System: ext4  
Mount Options: relatime rw  
Disk Scheduler: MO-DEADLINE  
Disk Details: Block Size: 4096  
  
OPERATING SYSTEM:  
Ubuntu 20.04  
Kernel: 5.4.0-107-generic (x86_64)  
System Layer: Oracle VMWare  
Security:  
+ intel_mitigation: KVM: Vulnerable  
+ l1tf: Mitigation of PTE Inversion  
+ mds: Mitigation of cleared buffers; SMT Host state unknown  
+ meltdown_mitigation: PTE  
+ spectre_v1: Mitigation of usercopy/swaps barriers and __user pointer sanitization  
+ spectre_v2: Mitigation of Retpolines STIBP: disabled RSB filling  
+ srbds: Unknown: Dependent on hypervisor status  
+ tsx_async_abort: Not affected  
  
Would you like to save these test results (Y/n): y  
Recently Saved Test Results:  
isetest [Today]  
Enter a name for the result file: isetest  
Enter a unique name to describe this test run / configuration:  
If desired, enter a new description below to better describe this result set / system configuration under test.  
Press ENTER to proceed without changes.  
Current Description: Oracle VMWare testing on Ubuntu 20.04 via the Phoronix Test Suite.  
New Description:  
Timed Idle:  
pts/idle-1.2.0 [Time To Idle (In Minutes): 1]  
Test 1 of 1  
Estimated Trial Run Count: 1  
Estimated Time To Completion: 3 Minutes [10:24 UTC]  
Started Run 1 @ 10:22:05
```

```
phoronix-test-suite list-saved-results  
phoronix-test-suite result-file-to-text isetest2
```

```
danielav@dantelav:~$ phoronix-test-suite result-file-to-text lsetest2
lsetest2
Oracle VMWare testing on Ubuntu 20.04 via the Phoronix Test Suite.

Intel Core i7-8750H - VMWare SVGA II - Oracle:
    Processor: Intel Core i7-8750H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX 82441FX PMC, Memory: 1624MB, Disk: 2 x 11GB VBOX HDD, Graphics: VMWare SVGA II, Audio: Intel 82801
    AA AC 97 Audio, Network: 2 x Intel 82540EM
    OS: Ubuntu 20.04, Kernel: 5.4.0-107-generic (x86_64), File-System: ext4, Screen Resolution: 2048x2048, System Layer: Oracle VMWare

    Timed Idle
    Time To Idle (In Minutes): 1
    Intel Core i7-8750H - VMWare SVGA II - Oracle . PASS

danielav@dantelav:~$
```

Vamos ahora con **CentOS**

Nuevamente, vamos a instalar phoronix en **CentOS**. Para ello, he consultado esta [página](#). Como vemos, el proceso es bastante similar al de **Ubuntu** y, para realizarlo, voy a usar ssh para que sea más fácil copiar los comandos:

```
# Notemos que necesitamos instalado algunas dependencias previas
# Recuerda levantar las interfaces de red
sudo yum install wget php-cli php-xml bzip2
sudo wget https://phoronix-test-suite.com/releases/phoronix-test-suite-
8.4.1.tar.gz
sudo tar xvfz phoronix-test-suite-8.4.1.tar.gz
cd phoronix-test-suite
sudo ./install-sh
```

Para ver los test disponibles:

```
/usr/bin/phoronix-test-suite list-available-suites
```

Al hacerlo, me sale el siguiente error:

```
Actividades Terminal 19 de may 19:59 • danielav@localhost:~/phoronix-test-suite
[danielav@localhost phoronix-test-suite]$ /usr/bin/phoronix-test-suite list-available-suites
The following PHP extensions are REQUIRED:
JSON      JSON support is required for OpenBenchmarking.org.
The following PHP extensions are OPTIONAL but recommended:
GD        The GD Library is recommended for improved graph rendering.
POSIX    POSIX support is highly recommended.
[danielav@localhost phoronix-test-suite]$
```

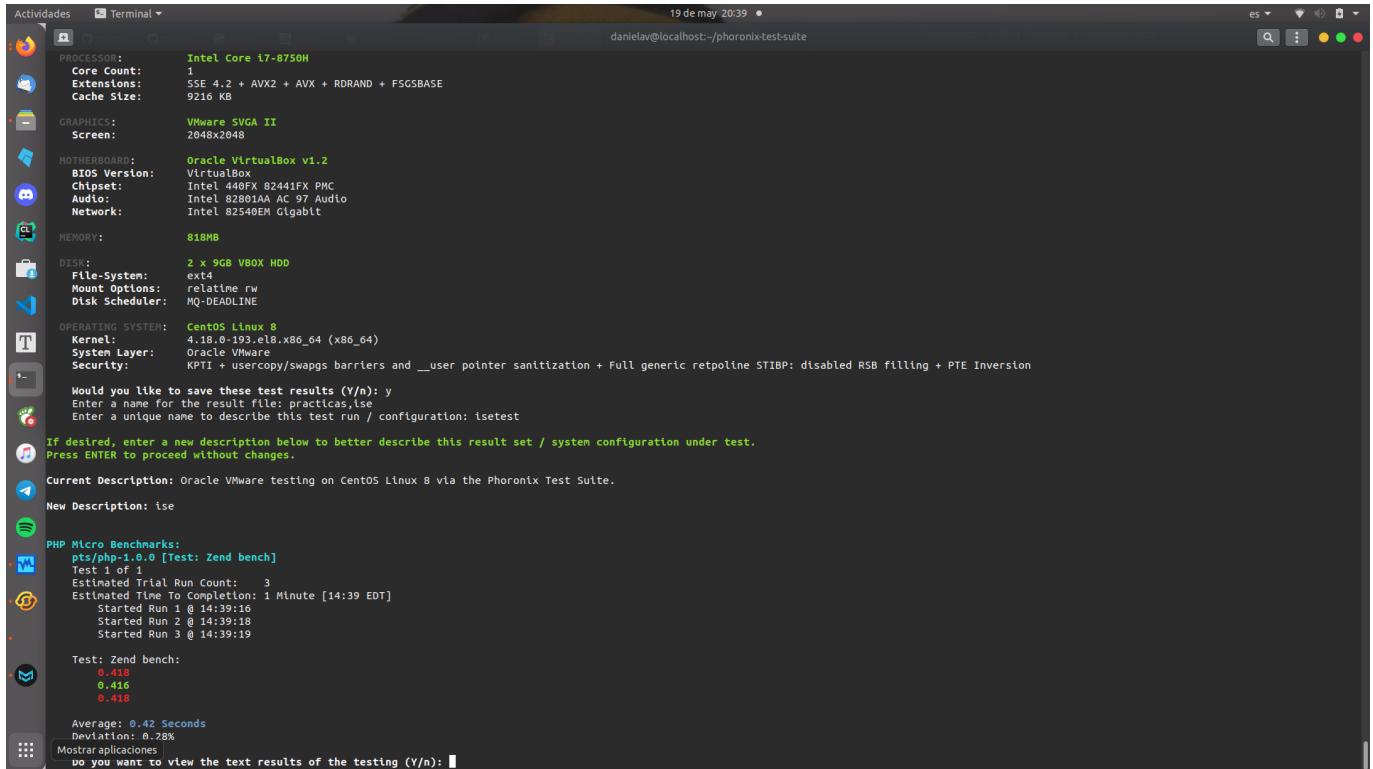
Luego, vamos a instalar dicha extensión requerida. Para ello, ejecutamos `sudo yum install php-json`. Ahora, al ejecutar el comando anterior, si nos dará una salida válida:

```
Actividades Terminal 19 de may 20:02 • danielav@localhost:~/phoronix-test-suite
[danielav@localhost phoronix-test-suite]$ /usr/bin/phoronix-test-suite
Test Suite, or any associated stakeholder be liable to any party
for any direct or indirect damages for any use of
OpenBenchmarking.org -- including, without limitation, any lost
profits, business interruption, loss of programs, loss of
programmed data, or otherwise.
- For enterprise/commercial support, sponsorship, or other
professional inquirers, contact
commercial@phoronix-test-suite.com. Community support can be
found in the Phoronix Forums at https://www.phoronix.com/forums/.
- If you opt to submit your test results to OpenBenchmarking.org,
the final results as well as basic hardware and software details
(what is shown in the results viewer) will be shared and publicly
accessible through https://www.openbenchmarking.org/ and
optionally relevant system details like dmesg and /proc/cpuinfo.
- Public bug reports, feature requests, and other issues can be
brought up in the Phoronix Test Suite forums, mailing list, or a
direct email to Phoronix Media.
Anonymous Usage Reporting / Statistics: If enabling the anonymous
usage reporting / statistics feature, some information about the
Phoronix Test Suite runs will be submitted to
OpenBenchmarking.org. This information is used for analytical
purposes, including the determining of the most popular tests /
suites and calculating average run-times for different test
profiles. The test results are not reported in this process nor
the installed software / hardware information, but ambient
information about the testing process. This information is stored
anonimously. More information on this feature is available with
the included documentation.
For more information on the Phoronix Test Suite and its features,
visit https://www.phoronix-test-suite.com/ or view the included
documentation or by contacting support@phoronix-test-suite.com.
Do you agree to these terms and wish to proceed (y/n): y
Enable anonymous usage / statistics reporting (y/n): y
AN OUTDATED VERSION OF THE PHORONIX TEST SUITE IS INSTALLED.
THE VERSION IN USE IS 8.4.1 (8410), BUT THE LATEST IS PTS-CORE 10830.
VISIT HTTPS://WWW.PHORONIX-TEST-SUITE.COM/ TO UPDATE THIS SOFTWARE.

Available Suites
pts/audio-encoding          - Audio Encoding           System
pts/av1                      - AV1                  System
pts/informatics              - Informatics          System
pts/browseers                 - Web Browsers        System
pts/cad                      - CAD                 System
                               - Chess Test Suite Processor
Mostrar aplicaciones
```

Como para *Ubuntu* he ejecutado suites, ahora voy a ejecutar test directamente para **CentOS**. Los comandos son análogos, pero llamamos al programa mediante `/usr/bin/phoronix-test-suite`:

```
/usr/bin/phoronix-test-suite list-tests  
/usr/bin/phoronix-test-suite benchmark pts/php  
/usr/bin/phoronix-test-suite list-saved-results  
/usr/bin/phoronix-test-suite result-file-to-text practicasise
```



```
/usr/bin/phoronix-test-suite benchmark pts/idle
/usr/bin/phoronix-test-suite list-saved-results
/usr/bin/phoronix-test-suite show-result isetest
```

```
Actividades Terminal • 19 de may 20:46 •
danielav@localhost:~/phoronix-test-suite

OPERATING SYSTEM: CentOS Linux 8
Kernel: 4.18.0-193.el8.x86_64 (x86_64)
System Layer: Oracle VMWare
Security: KPTI + usercopy/swaps barriers and __user pointer sanitization + Full generic retpoline STIBP: disabled RSB filling + PTE Inversion

Would you like to save these test results (Y/n): y

Recently Saved Test Results:
practicasise [Today]

Enter a name for the result file: isetests
Enter a unique name to describe this test run / configuration: isetests

If desired, enter a new description below to better describe this result set / system configuration under test.
Press ENTER to proceed without changes.

Current Description: Oracle VMware testing on CentOS Linux 8 via the Phoronix Test Suite.

New Description: isetests

T Timed Idle:
pts/dlidle-1.2.0 [Time To Idle (In Minutes): 1]
Test 1 of 1
Estimated Trial Run Count: 1
Estimated Time To Completion: 3 Minutes [14:47 EDT]
Started Run 1 @ 14:45:06
[NOTICE] Undefined: max_result in pts_test_result_parser:462
[NOTICE] Undefined: max_result in pts_test_result_parser:462

Final: PASS

Do you want to view the text results of the testing (Y/n): y
isetests
isetests

isetests:

Processor: Intel Core i7-8750H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX 82441FX PMC, Memory: 818MB, Disk: 2 x 9GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel 82801AA
AC 97 Audio, Network: Intel 82540EM Gigabit

OS: CentOS Linux 8, Kernel: 4.18.0-193.el8.x86_64 (x86_64), File-System: ext4, Screen Resolution: 2048x2048, System Layer: Oracle VMWare

Timed Idle
Time To Idle (In Minutes): 1
isetests .. PASS

Would you like to upload the results to OpenBenchmarking.org (y/n): y
Would you like to attach the system logs (lspci, dmesg, lsusb, etc) to the test result (y/n): y

Results Uploaded To: https://openbenchmarking.org/result/2205198-SK-ISETEST6895
Mostrar aplicaciones [danielav@localhost phoronix-test-suite]$
```

```
Actividades Terminal • 19 de may 20:48 •
danielav@localhost:~/phoronix-test-suite

debug-render-test
debug-self-test
help
version

MODULES
auto-load-module
list-modules
module-info [Phoronix Test Suite Module]
module-setup [Phoronix Test Suite Module]
test-module [Phoronix Test Suite Module]
unload-module

USER CONFIGURATION
enterprise-setup
network-info
network-setup
user-config-reset
user-config-set

WEB / GUI SUPPORT
gui

PHOROMATIC
start-phromatic-server

[danielav@localhost phoronix-test-suite]$ sudo /usr/bin/phoronix-test-suite show-result isetests

Phoronix Test Suite v8.4.1
AN OUTDATED VERSION OF THE PHORONIX TEST SUITE IS INSTALLED.
THE VERSION IN USE IS 8.4.1 (8410), BUT THE LATEST IS PTS-CORE 10830.
VISIT HTTPS://WWW.PHORONIX-TEST-SUITE.COM/ TO UPDATE THIS SOFTWARE.

isetests
isetests

isetests:
Processor: Intel Core i7-8750H (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel 440FX 82441FX PMC, Memory: 818MB, Disk: 2 x 9GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel 82801AA
AC 97 Audio, Network: Intel 82540EM Gigabit

OS: CentOS Linux 8, Kernel: 4.18.0-193.el8.x86_64 (x86_64), File-System: ext4, Screen Resolution: 2048x2048, System Layer: Oracle VMWare

Timed Idle
Time To Idle (In Minutes): 1
isetests .. PASS

Mostrar aplicaciones [danielav@localhost phoronix-test-suite]$
```

Comparación

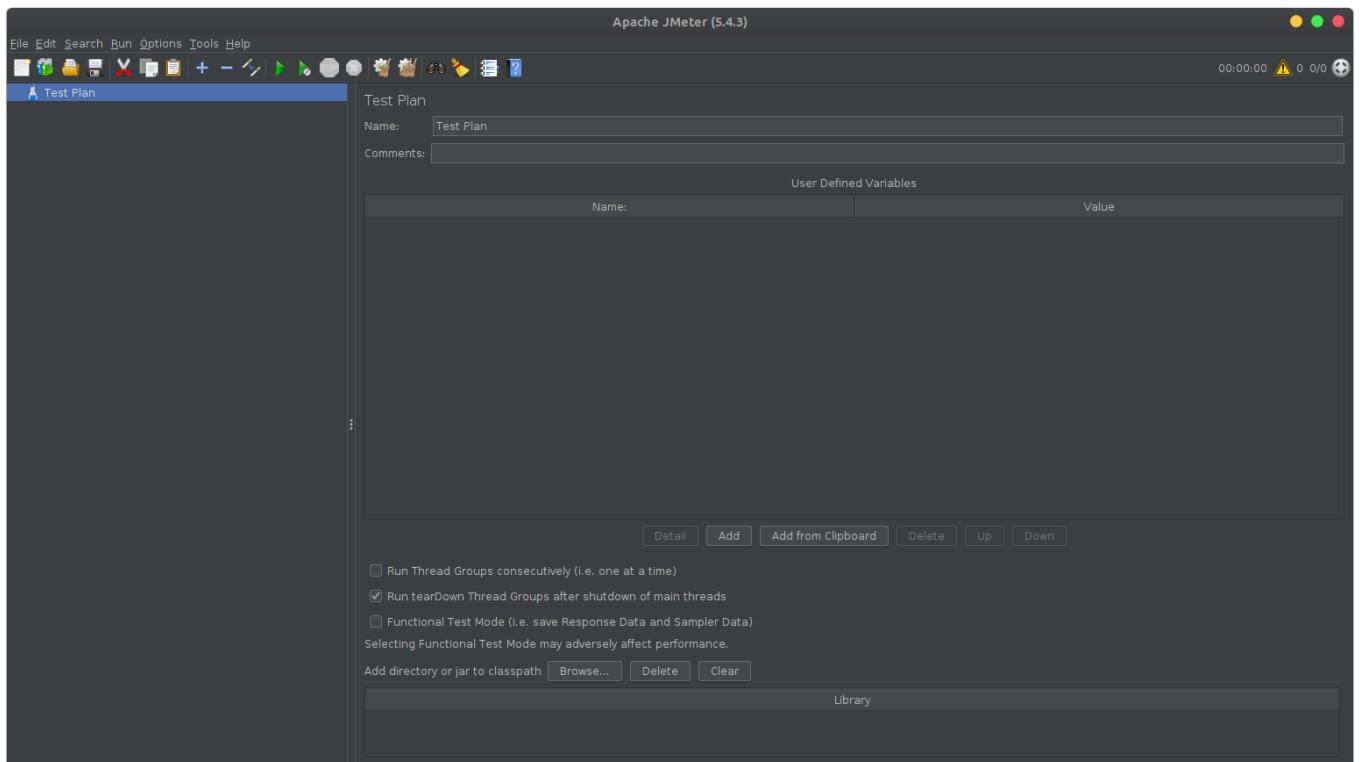
En caso del primer test, ambos realizan 3 pruebas y muestran la media de los tiempos obtenidos en cada una. Podemos ver que el rendimiento de UbuntuServer ha sido algo superior al de CentOS.

Por otro lado, el segundo test solo suspende el sistema para hacer mediciones de sensores, baterías, etc..., y, podemos ver que ambos los superan sin problemas.

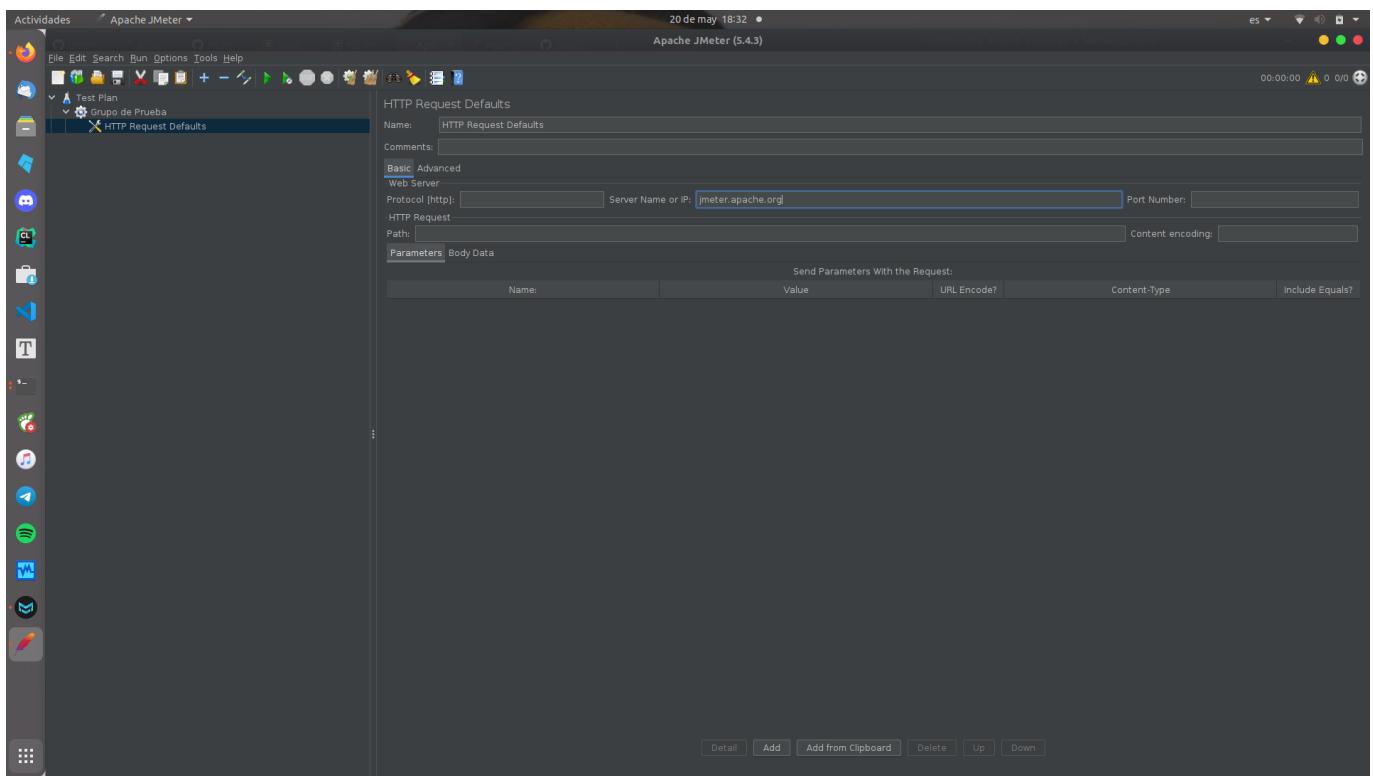
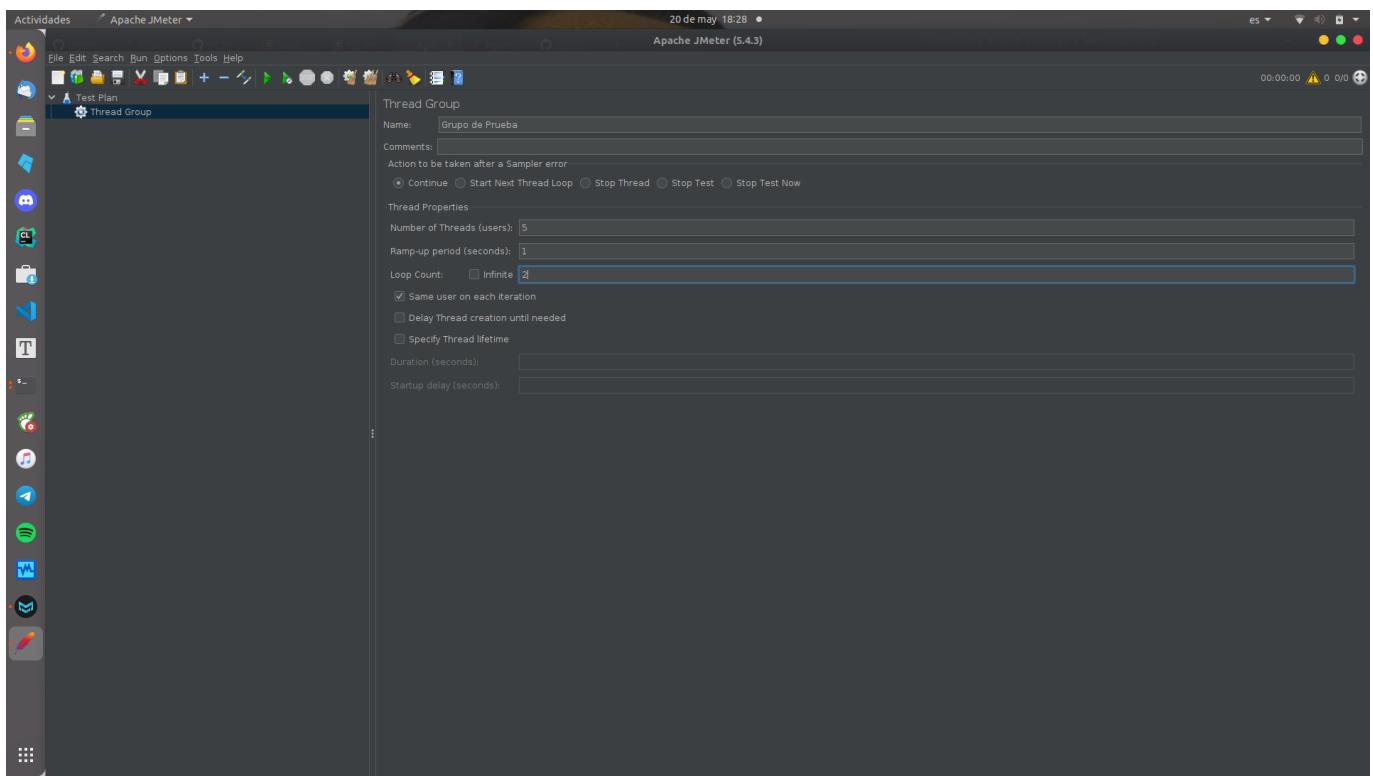
Ejercicio 2: JMeter

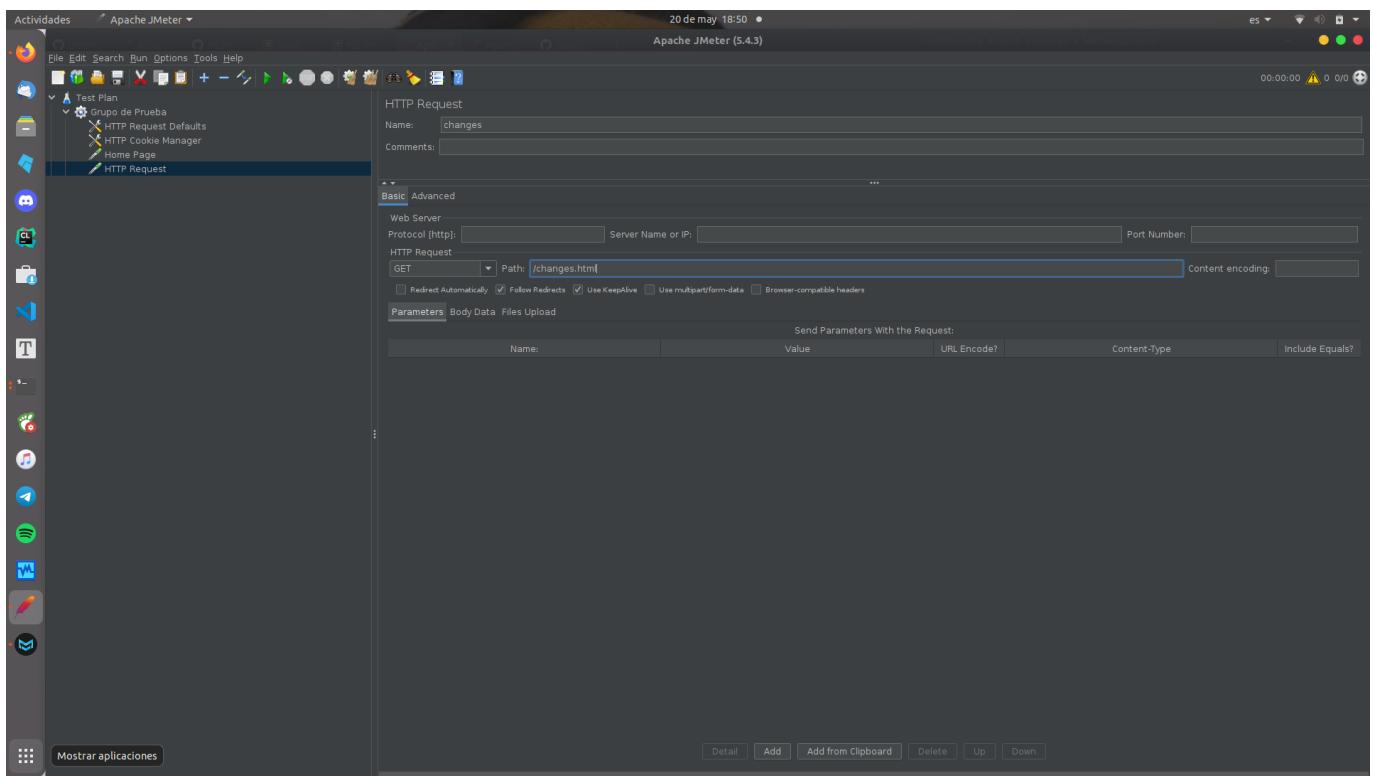
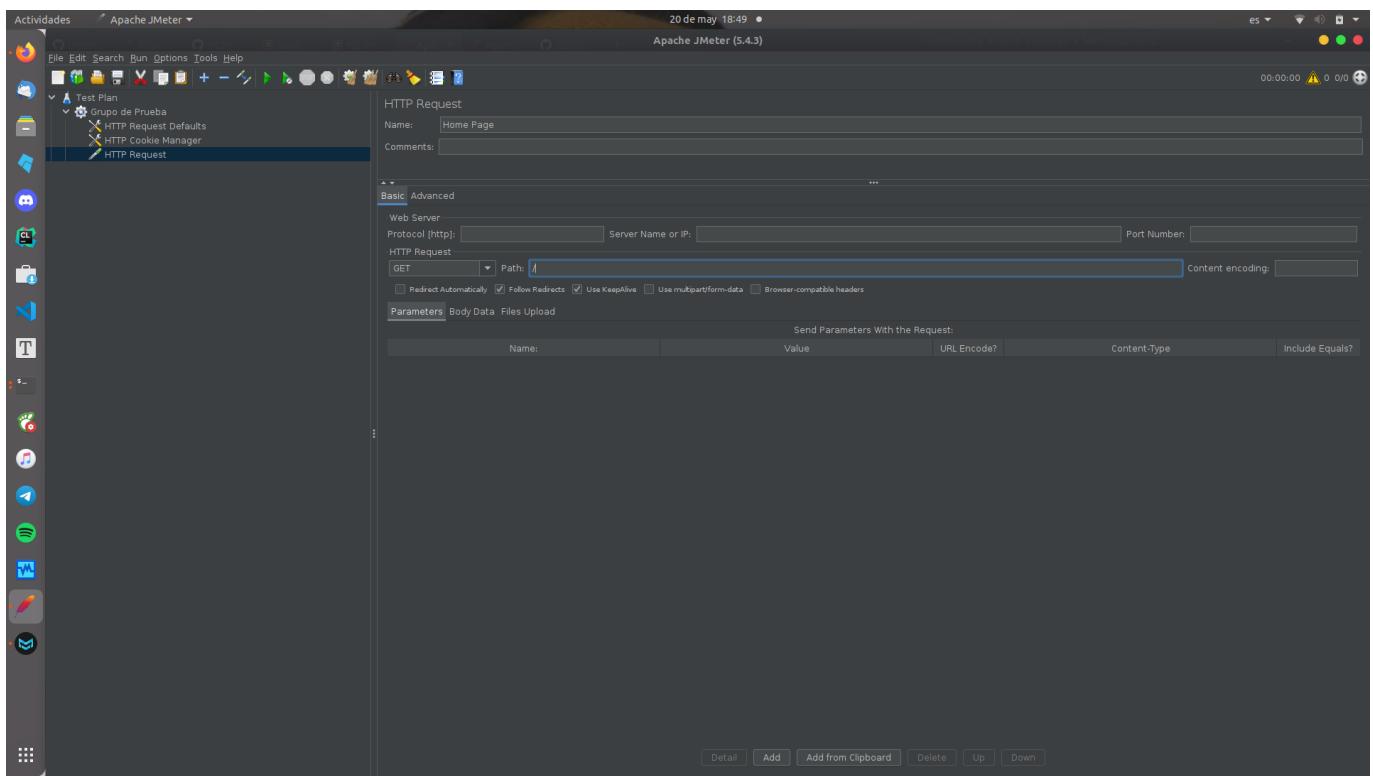
Vamos a comenzar instalando **JMeter** en nuestro ordenador. Para ello, basta con ir a la web oficial al apartado de descargas: https://jmeter.apache.org/download_jmeter.cgi

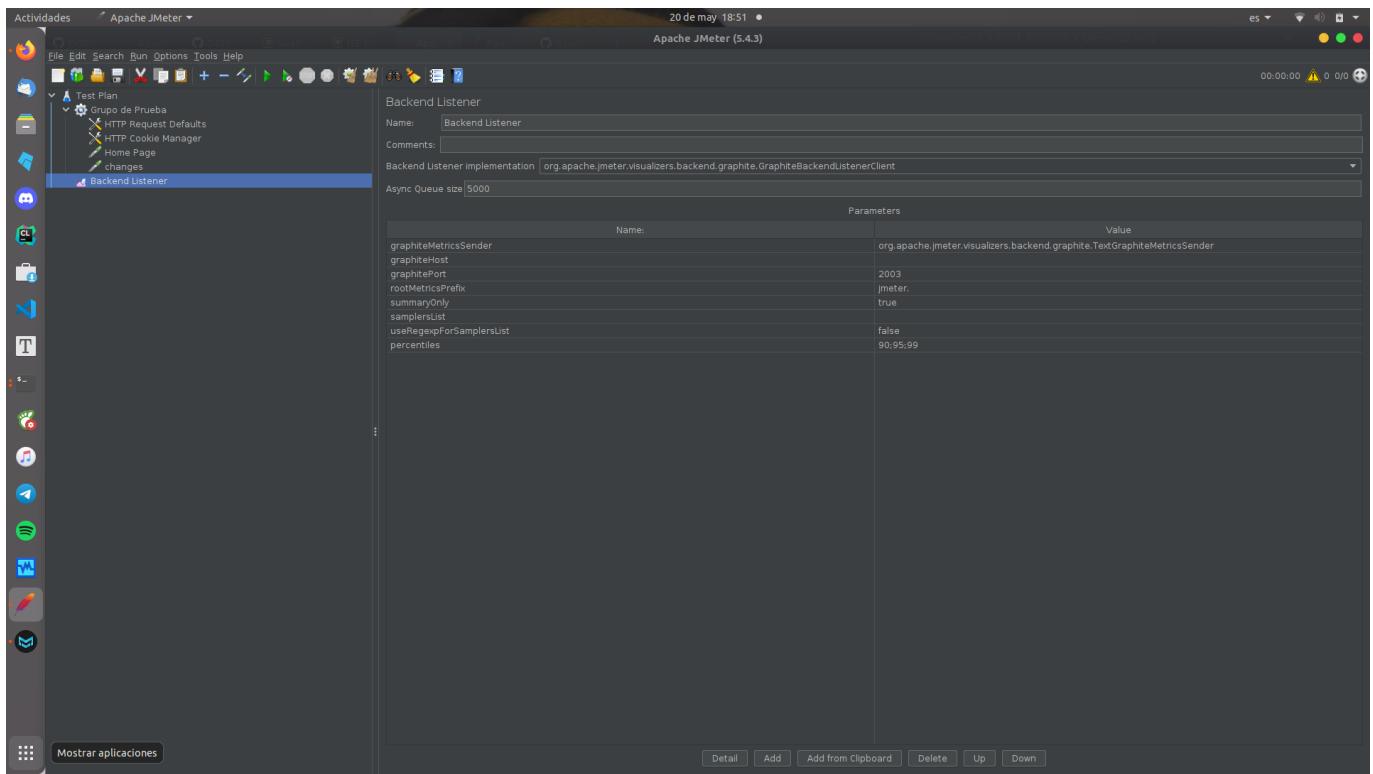
Descargamos los binarios y los descomprimimos. Para ejecutar el porgrama, basta con abrir una terminal en la carpeta donde hemos instalado el programa, acceder a la carpeta `bin` y ejecutar `jmeter`



Si consultamos la [documentación](#), podemos ver como crear un test básico para la web. Vamos a realizar dicho tutorial antes de pasar a docker (*No explicaré los pasos, solo capturas de los mismos, ya que viene explicado detalladamente en la documentación*).







Vamos ahora con la instalación de **docker** y **docker-compose** en **Ubuntu Server**. Para ello vamos seguir los pasos indicados en el guión de la asignatura (*los comandos son directamente copiados del guión*):

Añadimos la llave GPG para validar el repositorio:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Añadimos el 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
```

Buscamos el repositorio de docker y lo instalamos:

```

apt search docker-ce
sudo apt install docker-ce
# Comprobamos que el servicio está instalado y activo
sudo systemctl status docker

```

```

danielav@danielav: ~
Selecting previously unselected package containerd.io.
Preparing to unpack .../1-containerd.io_1.6.4-1_amd64.deb ...
Unpacking containerd.io (1.6.4-1) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../2-docker-ce-cli_5%3a20.10.16-3-0-ubuntu-focal_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.16-3-0-ubuntu-focal) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../3-docker-ce-rootless-extras_5%3a20.10.16-3-0-ubuntu-focal_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:20.10.16-3-0-ubuntu-focal) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../4-docker-ce-rootless-extras_5%3a20.10.16-3-0-ubuntu-focal_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:20.10.16-3-0-ubuntu-focal) ...
Selecting previously unselected package docker-scan-plugin.
Preparing to unpack .../5-docker-scan-plugin_0.17.0-ubuntu-focal_amd64.deb ...
Unpacking docker-scan-plugin (0.17.0-ubuntu-focal) ...
Selecting previously unselected package slirp4netns.
Preparing to unpack .../6-slirp4netns_0.4.3-1_amd64.deb ...
Unpacking slirp4netns (0.4.3-1) ...
Setting up slirp4netns (0.4.3-1) ...
Setting up docker-scan-plugin (0.17.0-ubuntu-focal) ...
Setting up containerd.io (1.6.4-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/c
ontainerd.service.
Setting up docker-ce-cli (5:20.10.16-3-0-ubuntu-focal) ...
Setting up pip (2.4.1) ...
Setting up docker-ce-rootless-extras (5:20.10.16-3-0-ubuntu-focal) ...
Setting up docker-ce (5:20.10.16-3-0-ubuntu-focal) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker
.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.so
cket.
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for systemd (245.4-4ubuntu3.15) ...
danielav@danielav:~$ sudo 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 2022-05-23 09:46:54 UTC; 3min 25s ago
TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
 Main PID: 20034 (dockerd)
   Tasks: 1
    Memory: 30.6M
      CGroup: /system.slice/docker.service
              └─20034 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

May 23 09:46:54 danielav dockerd[20034]: time=2022-05-23T09:46:54.449436879Z level=warning msg="You
May 23 09:46:54 danielav dockerd[20034]: time=2022-05-23T09:46:54.449525840Z level=warning msg="You
May 23 09:46:54 danielav dockerd[20034]: time=2022-05-23T09:46:54.449612683Z level=warning msg="You
May 23 09:46:54 danielav dockerd[20034]: time=2022-05-23T09:46:54.449879632Z level=info msg="Loadin
May 23 09:46:54 danielav dockerd[20034]: time=2022-05-23T09:46:54.518039641Z level=info msg="Default
May 23 09:46:54 danielav dockerd[20034]: time=2022-05-23T09:46:54.738399405Z level=info msg="Loadin
May 23 09:46:54 danielav dockerd[20034]: time=2022-05-23T09:46:54.837141380Z level=info msg="Docker
May 23 09:46:54 danielav dockerd[20034]: time= 2022-05-23T09:46:54.837474462Z level=info msg="Daemon
May 23 09:46:54 danielav dockerd[20034]: time= 2022-05-23T09:46:54.883109740Z level=info msg="API li
May 23 09:46:54 danielav systemd[1]: Started Docker Application Container Engine.
danielav@danielav:~$ 

```

Ahora, añadimos el usuario al grupo docker

```

sudo usermod -aG docker danielav
# Ahora tenemos que volver a loguearnos o llamar a bash
# Una vez hecho esto, podemos probar los comandos
docker info; docker run hello-world

```

```
danielav@danielav: ~
Default Runtime: runc
Init Binary: docker-init
Containerd Version: 212e0b6fa2f44b9c21b2798135fc6fb7c53efc16
Runc Version: v1.1.1-0-g52de29d
Init Version: de40ad0
Security Options:
    apparmor
    seccomp
    Profile: default
Kernel Version: 5.4.0-100-generic
Operating System: Ubuntu 20.04.1 LTS
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 976.9MiB
Name: danielav
ID: DNW6IPQGBCQQ:6J52:2B7M:AH2T:T42K:LROF:426W:KU7L:XRJP:YH33
Docker Root Dir: /var/lib/docker
Debug Mode: false
Registry: https://index.docker.io/v1/
Labels:
Experimental: false
Insecure Registries:
    127.0.0.0/8
Live Restore Enabled: false

WARNING: No swap limit support
danielav@danielav:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
zdb29710123e: Pull complete
Digest: sha256:80f31da1ac7b312ba29dd05080ffdf797dd7eacf870e677f390d5acba9741b1
Status: Downloaded newer image for hello-world:latest

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.
   (and64)
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/
danielav@danielav:~$
```

Vamos ahora con la instalación de **docker-compose**:

```
sudo apt install docker-compose
# Probar
docker-compose
docker-compose --version
```

```
Oracle VM VirtualBox Administrador

Processing triggers for man-db (2.9.1-1) ...
danielav@danielav:~$ docker-compose --version
docker-compose version 1.25.0, build unknown
danielav@danielav:~$ docker-compose
Define and run multi-container applications with Docker.

Usage:
  docker-compose [-f <arg>...] [options] [COMMAND] [ARGS...]
  docker-compose -h|--help

Options:
  -f, --file FILE          Specify an alternate compose file
                           (default: docker-compose.yml)
  -p, --project-name NAME  Specify an alternate project name
                           (default: directory name)
  --verbose                Show more output
  --log-level LEVEL        Set log level (DEBUG, INFO, WARNING, ERROR,
                           ... etc)
  --no-ansi                Do not print ANSI control characters
  -v, --version             Print version and exit
  -H, --host HOST          Daemon socket to connect to

  --tls                    Use TLS; implied by --tlsverify
  --tlscacert CA_PATH      Trust certs signed only by this CA
  --tlscert CLIENT_CERT_PATH Path to TLS certificate file
```

Por último, vamos a desplegar la aplicación en nuestro Ubuntu Server, para lo que debemos primero clonar el repositorio de [David Palomar](#). Para ello:

```
git clone https://github.com/davidPalomar-ugr/iseP4JMeter.git
cd iseP4JMeter
docker-compose up
# Si quieres dejar docker como un demonio usa la opción -d
```

```
danielav@danielav:~/iseP4JMeter$ docker-compose up
Creating network "isep4jmeter_default" with the default driver
Pulling mongodb (mongo:...
latest: Pulling from library/mongo
d5fd17ec1767: Extracting [>                                              ] 294.9kB/28.57MB
b2c91407ff49: Download complete
05bd3555a96c: Download complete
49fe477b9b26: Download complete
61c30709c846: Download complete
0c74ee88dd5a: Download complete
77b48c757eb5: Download complete
8aecb2f59c4d: Downloading [=====]                                         ] 34.36MB/211.6MB
6a1aab21c2a9: Download complete
e53674bec5aa: Download complete
[]
```

Tras cargar, se nos debe quedar así:

```
danielav@danielav:~/isePJMeter
mongod_1 | {"t": {"$date": "2022-05-23T10:48:54.844+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653302934:843975][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2171, snapshot max: 2171 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:49:54.900+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653302994:900453][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2173, snapshot max: 2173 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:50:54.948+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303054:948217][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2175, snapshot max: 2175 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:51:54.998+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303114:998519][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2177, snapshot max: 2177 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:55:05.049+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303175:49860][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2179, snapshot max: 2179 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:55:106+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303295:106715][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2181, snapshot max: 2181 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:55:160+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303295:1606631][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2183, snapshot max: 2183 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:55:212+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303355:212540][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2185, snapshot max: 2185 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:56:55.249+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303415:249862][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2187, snapshot max: 2187 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:57:55.303+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303475:303274][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2189, snapshot max: 2189 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:58:55.354+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303535:354650][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2191, snapshot max: 2191 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T10:59:55.408+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303595:408845][1:0x7f7d3cd0700], WT_SESSION.checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2193, snapshot max: 2193 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
mongod_1 | {"t": {"$date": "2022-05-23T11:00:55.457+00:00"}, "s": "I", "c": "STORAGE", "id": 22430, "ctx": "Checkpoint", "msg": "WiredTiger message", "attr": {"message": "[1653303655:457158][1:0x7f7d3cd0700], WT_SESSION.Checkpoint: [WT_Verb_CHECKPOINT_PROGRESS] saving checkpoint snapshot Min: 2195, snapshot Max: 2195 snapshot count: 0, oldest timestamp: (0, 0) base write gen: 1"}}
```

Si leemos la documentación que nos ha dejado David en el git, vemos que tenemos que abrir el puerto 3000. Para ello:

```
sudo ufw allow 3000/tcp
```

Ahora podemos acceder a la dirección <http://192.168.56.105:3000/>

ETSII Alumnos API

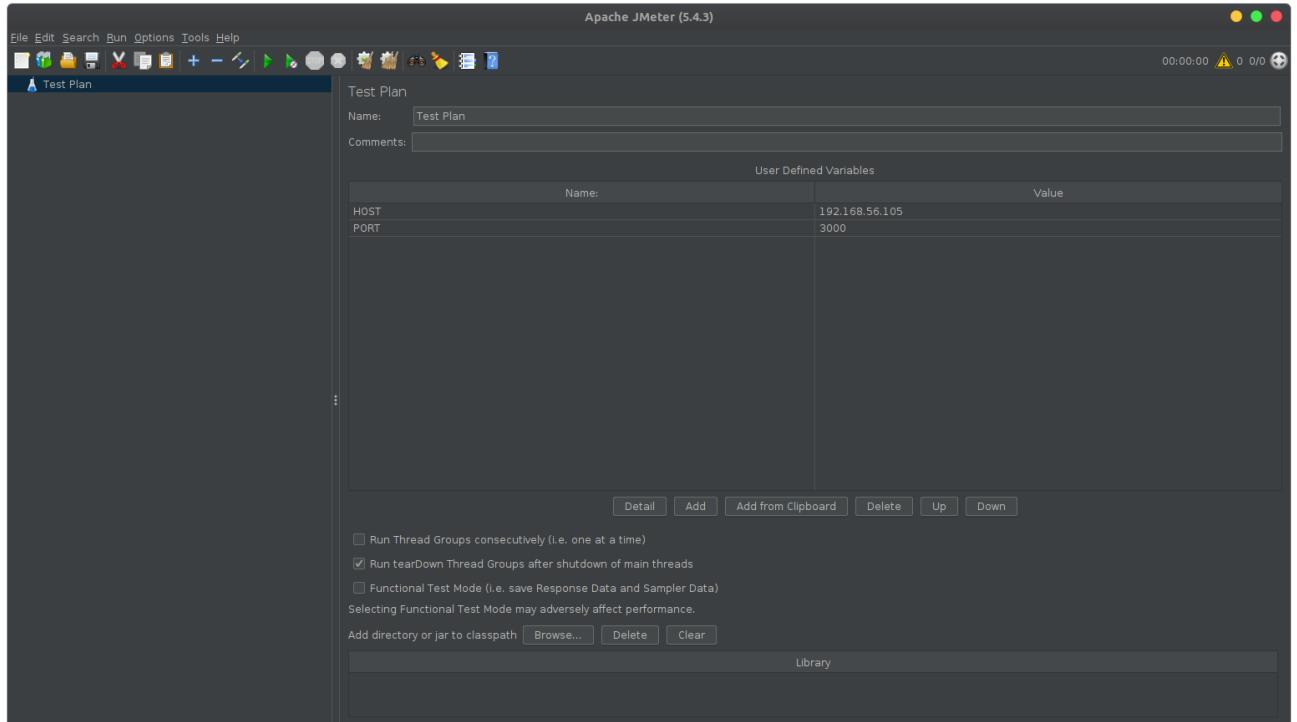
Descripción de la API Restful:
 POST /api/v1/auth/login
 Parámetros:
 login:<emailusuario>
 password:<secreto>
 Seguridad:
 Acceso protegido con BasicAuth (etsiiApi:laApiDeLaETSIIDaLache)
 Retorna:
 JWT Token

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 válido
 Retorna:
 Objeto Json con perfil de alumno

Volvemos a **JMeter** y vamos a crear el test con las condiciones que nos pide el enunciado:

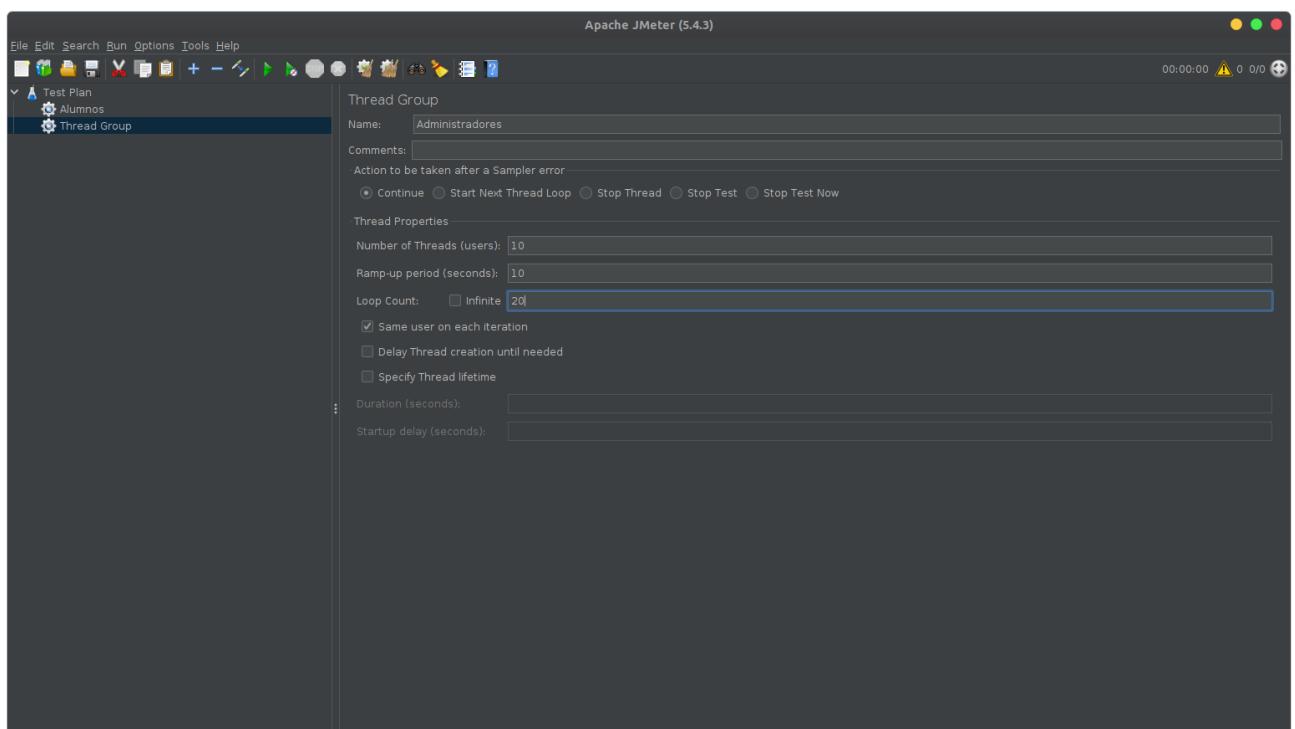
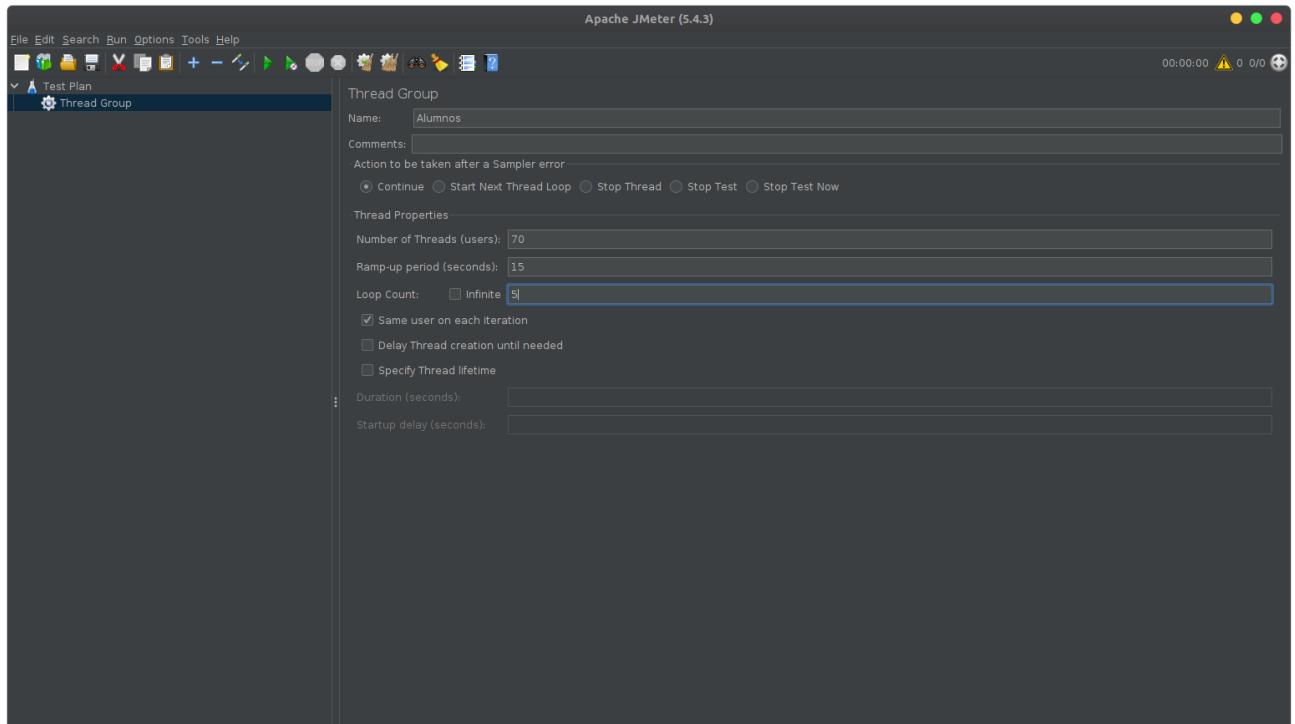
- El primer paso será **parametrizar el host y el puerto**, tal y como nos indica el primer punto del enunciado. Para ello, teniendo en cuenta que al iniciar la app aparece un test ya

creado sin ninguna especificación (llamado TestPlan), lo hacemos desde *Add* y *rellenando los campos* tal y como muestro a continuación:



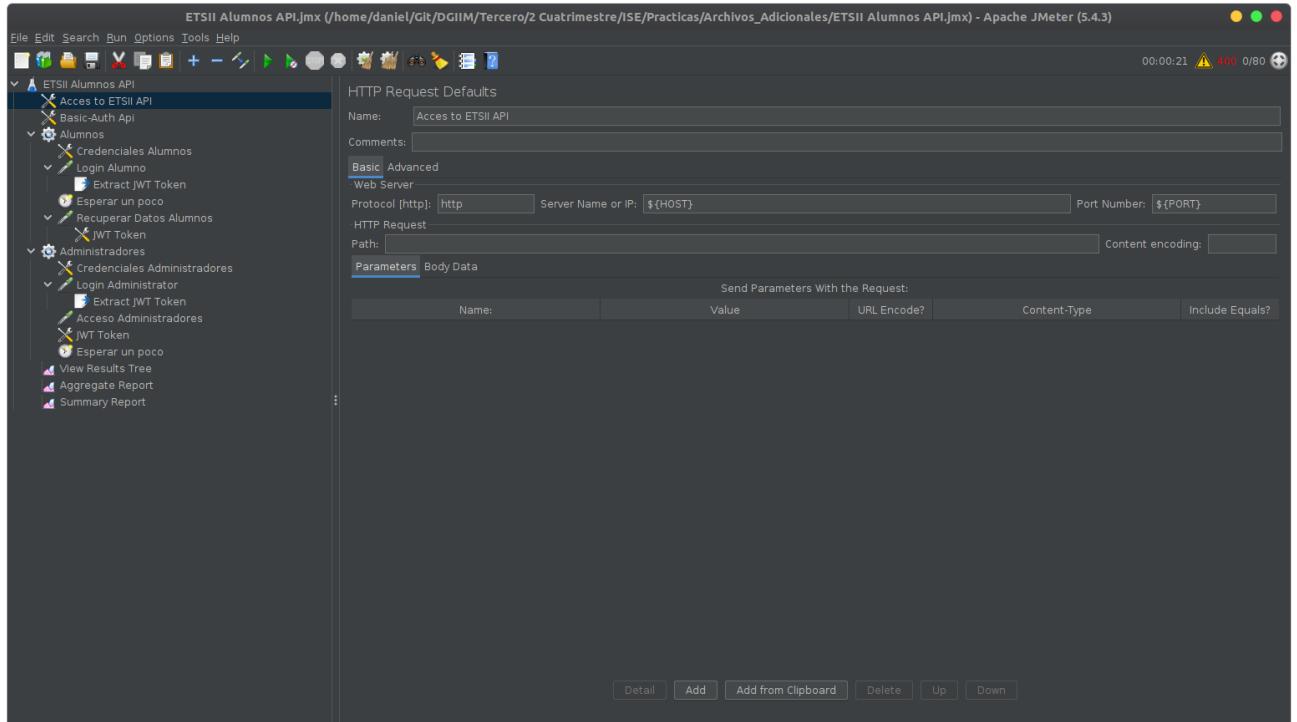
- Lo siguiente que nos piden es crear dos grupos de hebras distintos para simular el acceso de los alumnos y los administradores. Para ello, *Edit>Add>Threads>Thread Group*. Esta ventana ya la hemos utilizado anteriormente, si hemos hecho el tutorial que viene en la documentación.

Modificamos el nombre, el número de threads (usuarios), el período de subida (*ramp-up period*, que es el tiempo que tarda en alcanzar el número máximo de peticiones) y el número de iteraciones del bucle. Nos debe quedar tal que así:

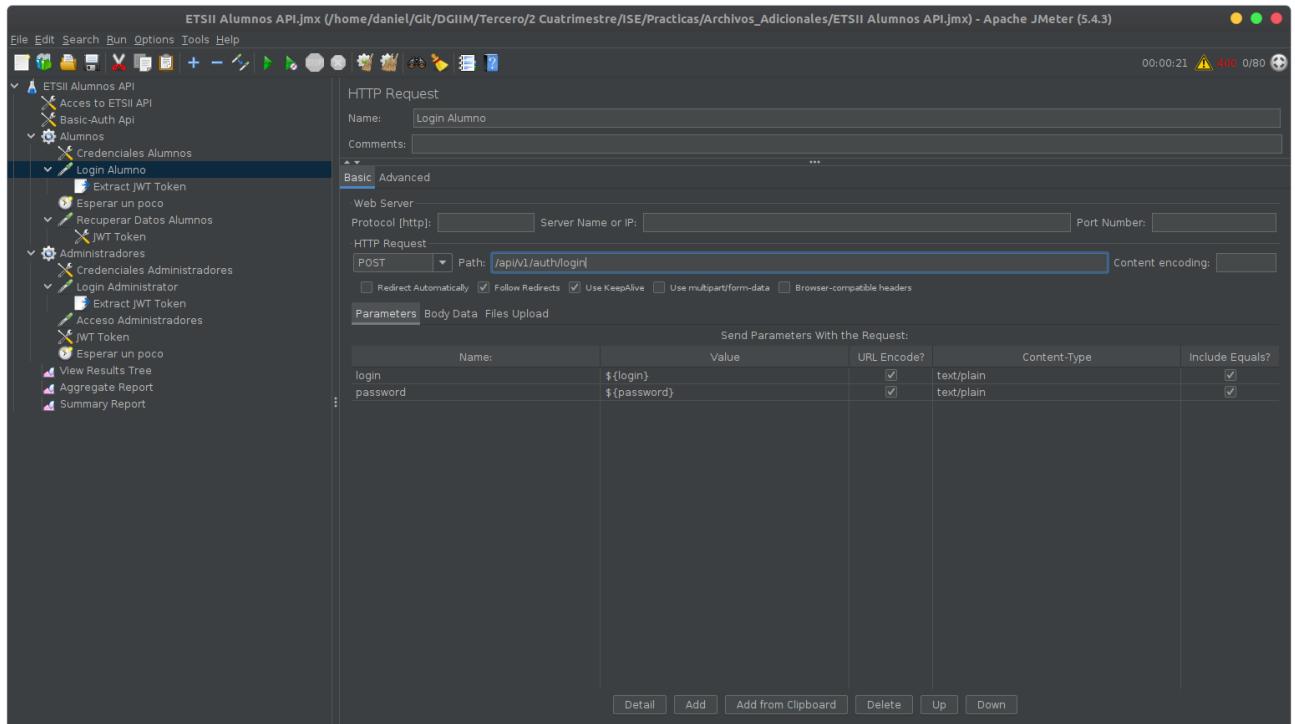


- Tal y como dice el enunciado el login de los alumnos y administradores, así como sus consultas, son peticiones **http**, por lo que procedemos a **crear los valores por defecto de dichas peticiones**. Para ello, desde la ventana inicial del test, *Edit>Add>Config Element>HTTP Request Default*. Lo colocamos arriba de los dos grupos de hebras, ya que es algo común a ambas. Esta ventana también la hemos manejado anteriormente en el tutorial. Dentro de los valores por defecto, modificamos el campo de puerto al parámetro

`${PORT}` y la ip a `${HOST}`, los cuales hemos definido al principio de esta sección. Al final nos debe quedar tal que así:



Ahora, **simularemos las peticiones HTTP de acceso de los alumnos**. Para esto, creamos una petición http desde los alumnos haciendo *Edit>Add>Sampler>HTTP Request*. Si leemos la descripción de la [API](#), vemos que los alumnos realizan una petición **POST** a la dirección `/api/v1/auth/login`, por lo que especificaremos exactamente esta petición. Además, tenemos que pasarle los valores de usuario y contraseña codificados. Al final, debe quedar algo de esta forma:



Los valores de host y puerto ya los hemos especificado en la petición http por defecto

- Si vemos la descripción de la API, vemos que la petición POST requiere de dos parámetros, login y password, luego, previamente de que se ejecuten estas peticiones, necesitamos que se haya leido estas variables del archivo `alumnos.csv`

ETSI Alumnos API

```

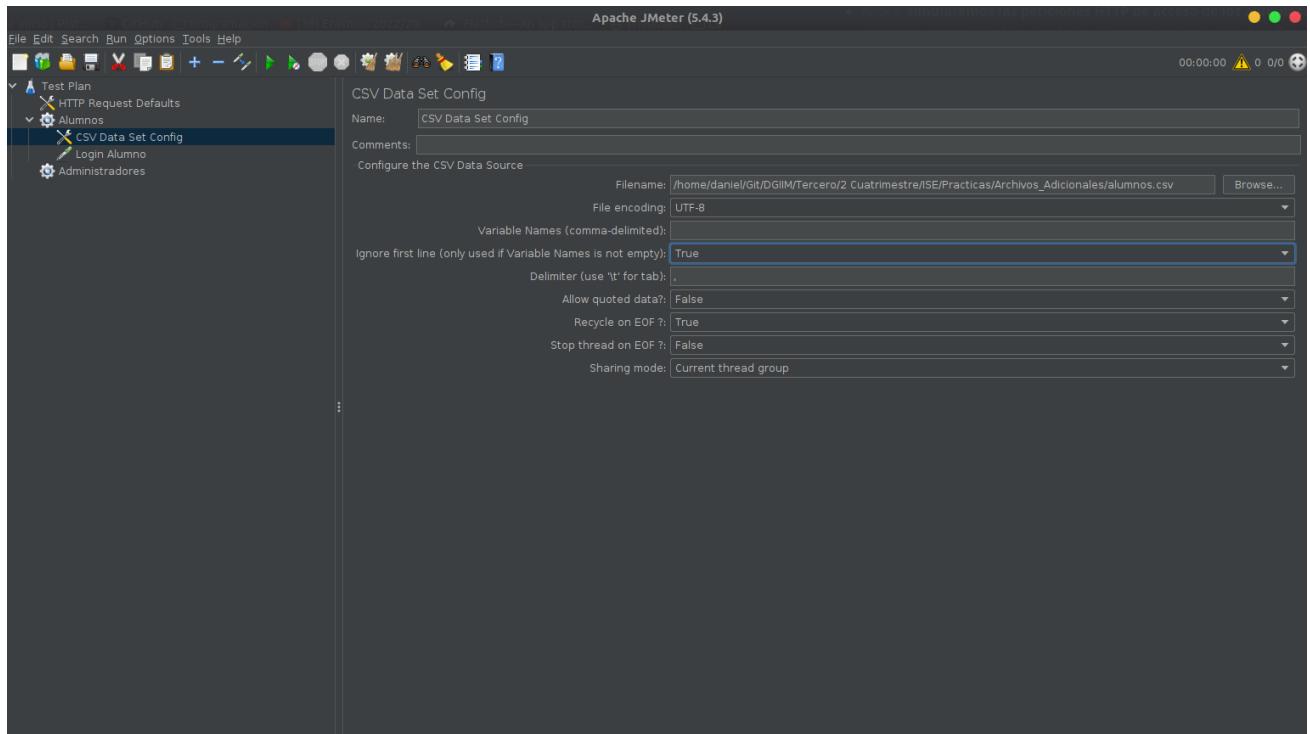
Descripción de la API Restful:
  POST /api/v1/auth/login
    Parámetros:
      login:<emailUsuario>
      password:<secreto>
    Seguridad:
      Acceso protegido con BasicAuth (etsiiApi:laApiDeLaETSIIDaLache)
    Retorna:
      JWT Token

  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 válido
    Retorna:
      Objeto Json con perfil de alumno

```

Para ello, hacemos click derecho sobre *Login Alumnos>Add>Config Element>CSV Data Set* y lo ponemos sobre *Login Alumnos*.

Vamos a requerir tener el archivo descargado, por lo que lo tomamos de github. Una vez descargado, indicamos la ruta del archivo y modificamos los parámetros siguientes parámetros:



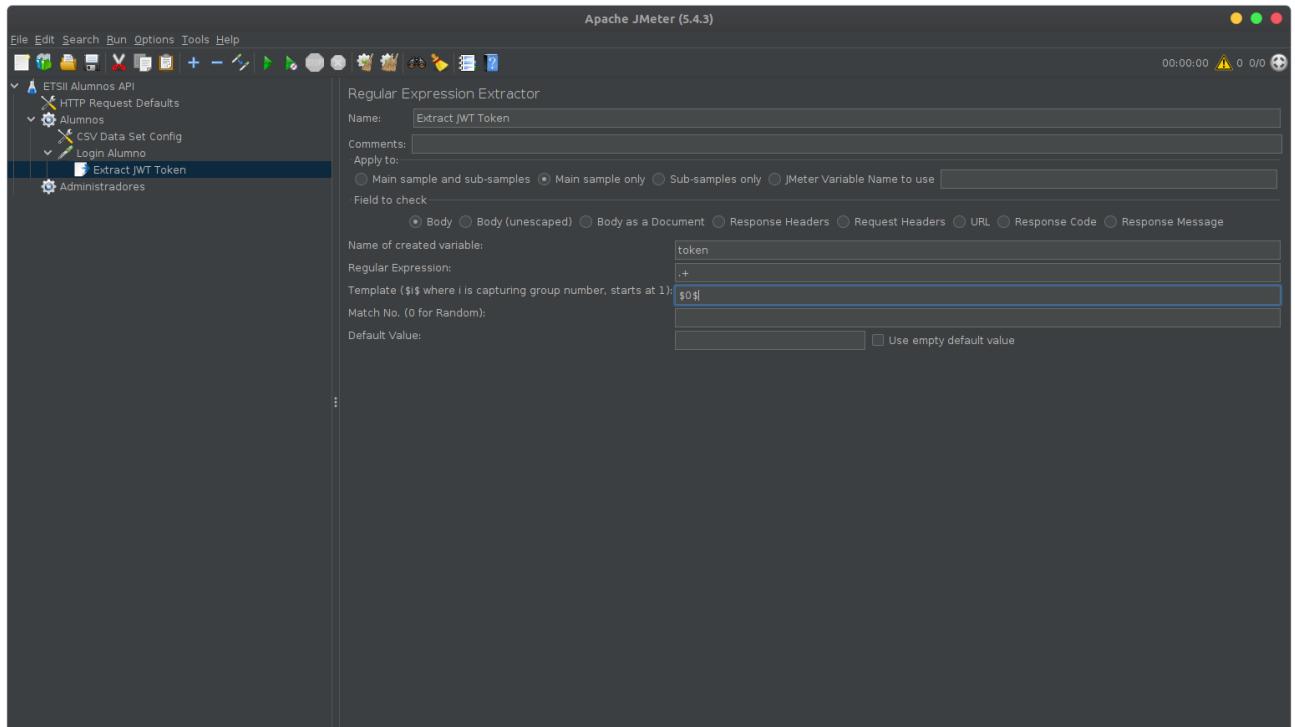
Con esta configuración, indicamos que la primera línea tiene el nombre de las variables login y password

- Si la petición POST que hemos realizado se ejecuta correctamente, la API nos devolverá el **token JWT** del usuario que hizo la petición. En este token viene la respuesta de la petición, por lo que tenemos que comprobar mediante expresiones regulares si el token es correcto.

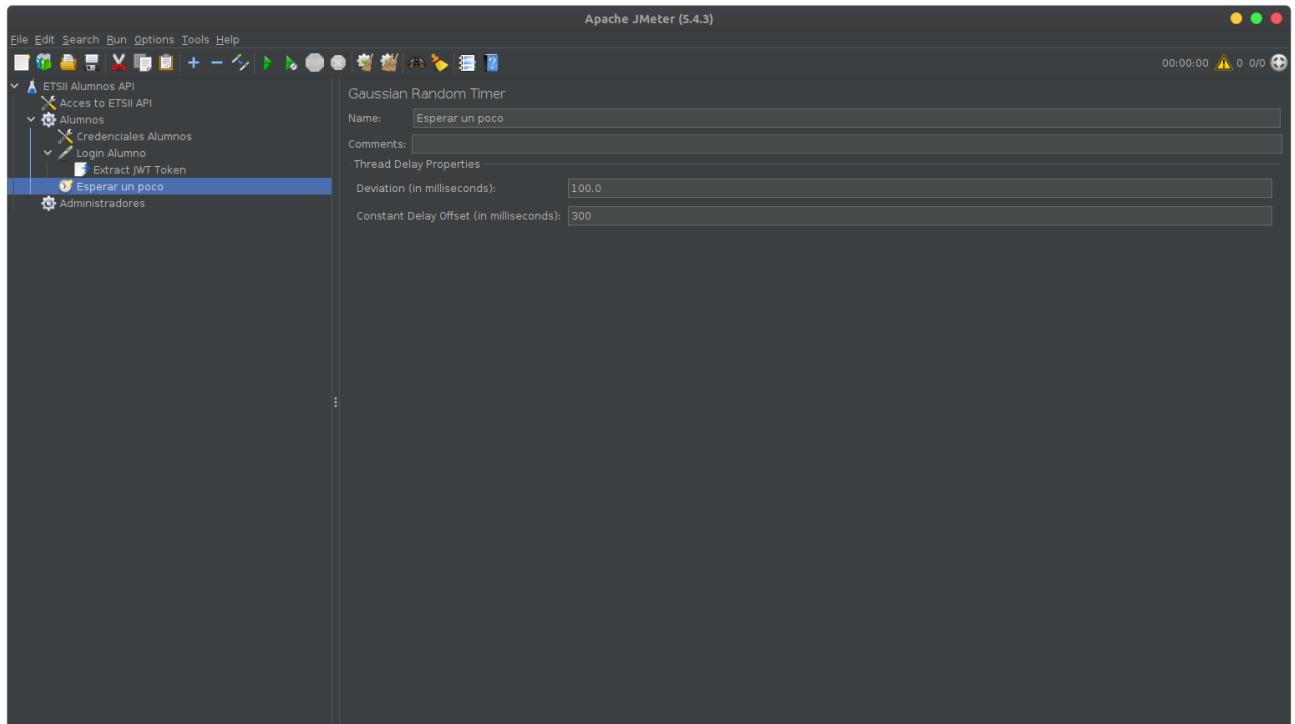
Además, si consultamos nuestra API, veremos que necesitamos almacenarlo para realizar peticiones de tipo **GET**.

Si se quiere consultar más información de los JWT ver [aqui](#).

Para almacenarlo, desde *Login Alumnos>Edit>Add>Post Processor>Regular Expression Extractor*. Tendremos que modificar *Name of created variable*, *Regular Expression* y *Template*, que hacer referencia al nombre de la variable, a la expresión regular que extraemos y lo que guardamos en la variable respectivamente. Al final nos quedará tal que así:

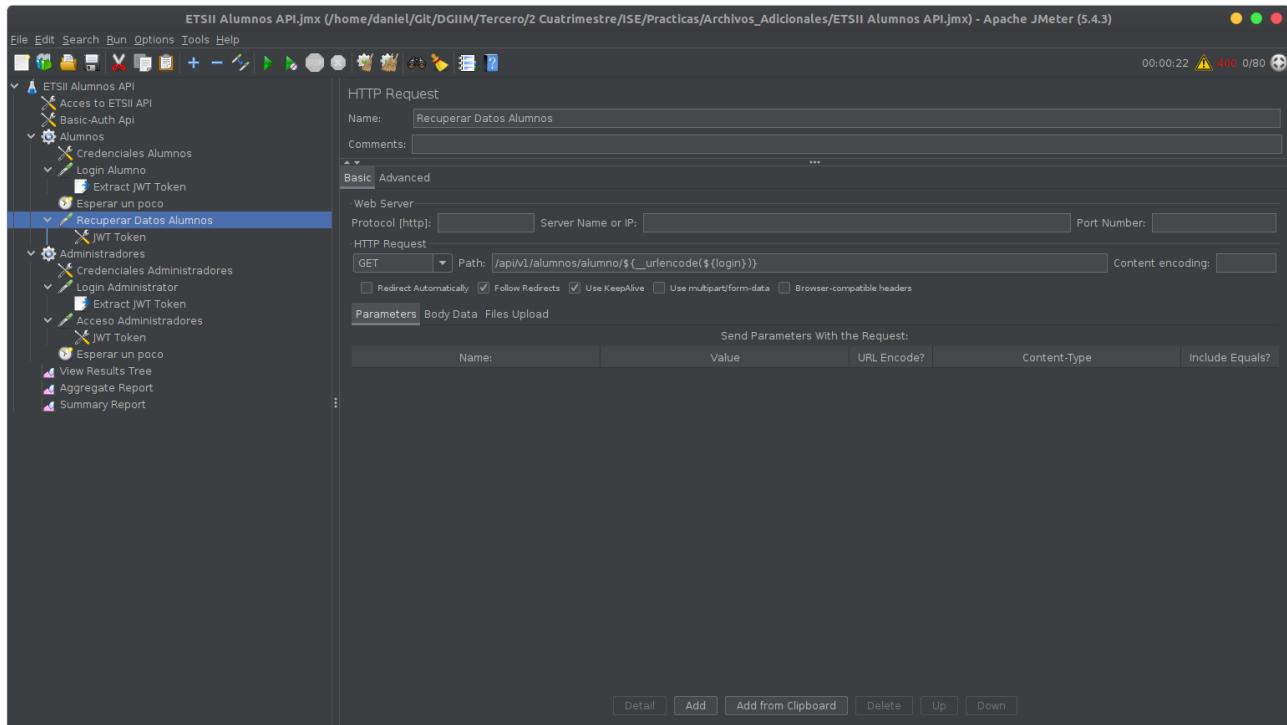


- Vamos ahora con la **pausa aleatoria Gaussiana**. Para ello, haciendo click derecho en *Alumnos>Add>Timer>Gaussian Random Timer*. Podemos modificar los parámetros para simular el comportamiento real de los usuarios mediante pausas aleatorias. Al final, deberíamos tener algo así:



He corregido los nombres para dejarlos como los indicados en la imagen de git

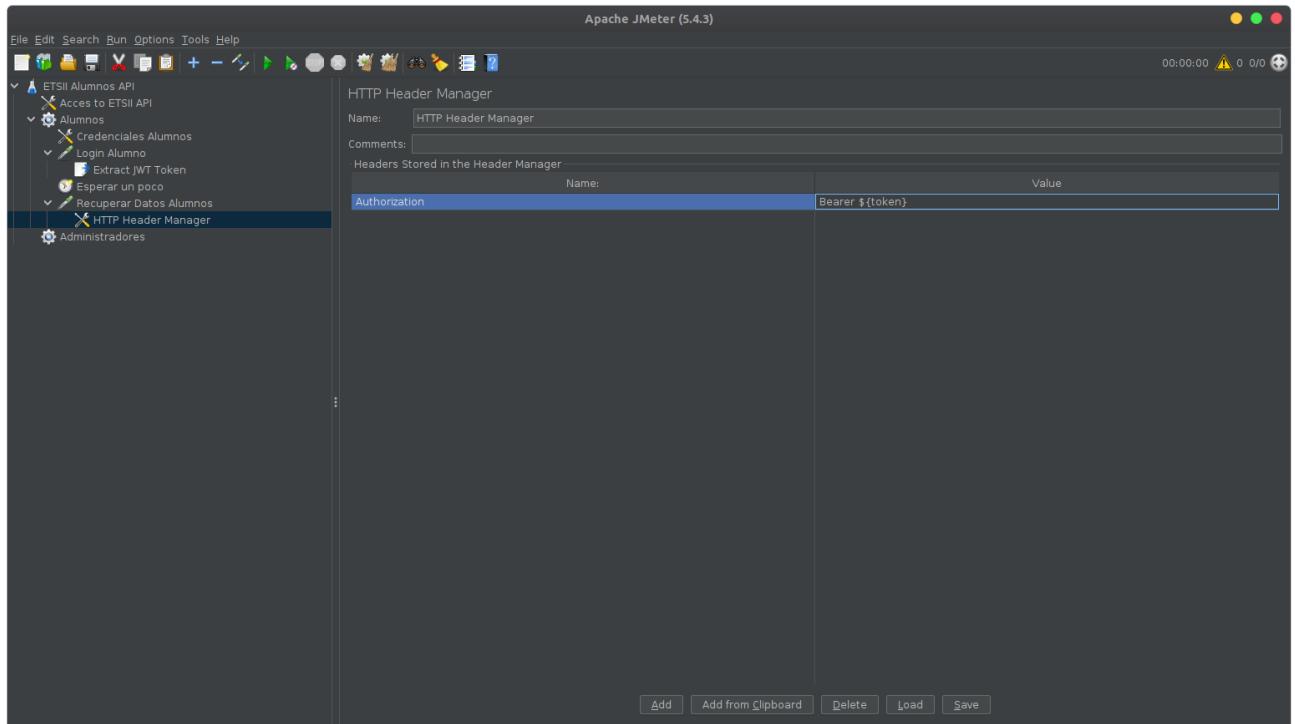
- Tras habernos autenticado con la petición http de login alumno, ahora realizaremos una petición tipo get. Con esta petición, podemos recuperar datos del alumno. Para ello, click derecho sobre *Alumnos>Add>Sampler>HTTP Request* y esta vez elegimos una petición **GET**



Notemos que la ruta depende del usuario que se ha logeado. He estado consultando y al final he llegado a esta [información](#).

- Por último en cuanto a los alumnos, como es necesario autenticarse para realizar una petición del tipo GET, creamos un **Gestor de Cabeceras HTTP** que usa el **JWT** conseguido previamente (mas información [aquí](#)).

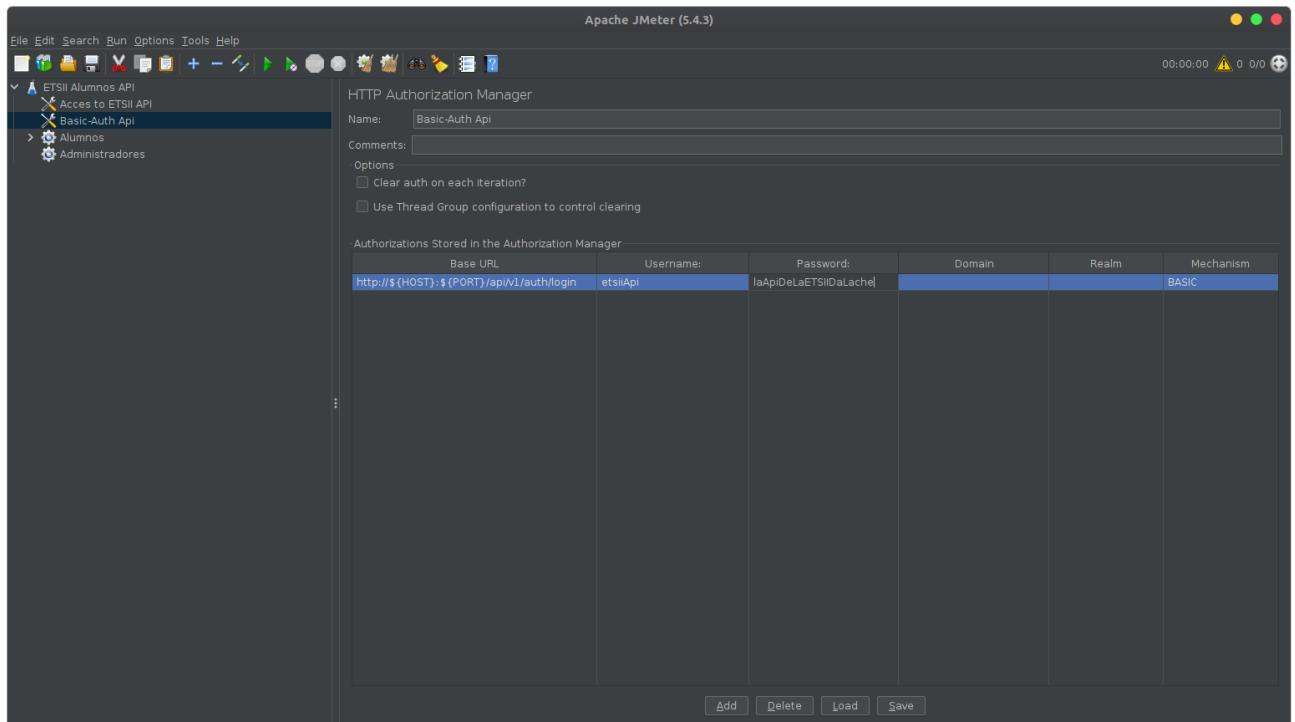
Para ello, hacemos click derecho en *Alumnos>Add>Config Element>HTTP Header Manager* y lo colocamos debajo de la petición GET y añadimos el parámetro que aparece en la imagen:



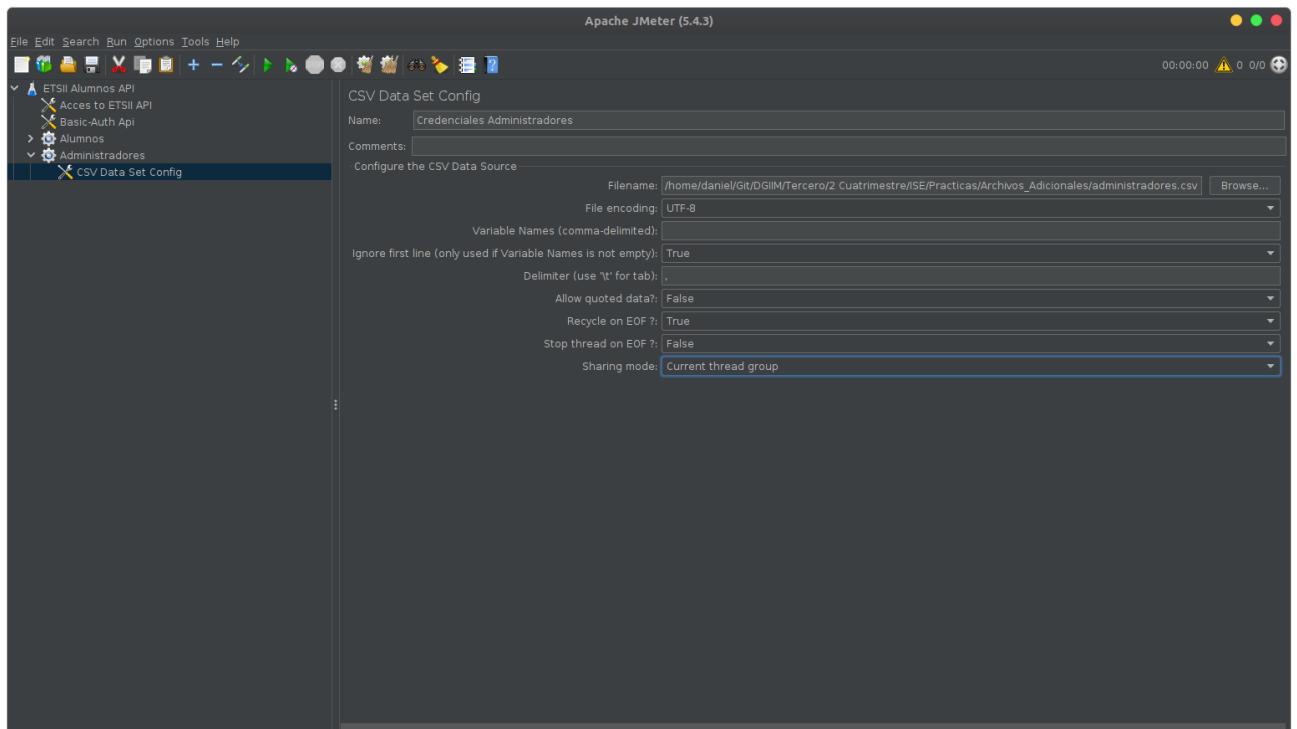
Para más información de Bearer Authentication ver [aqui](#)

Ahora que hemos acabado con los alumnos, vamos con los administradores.

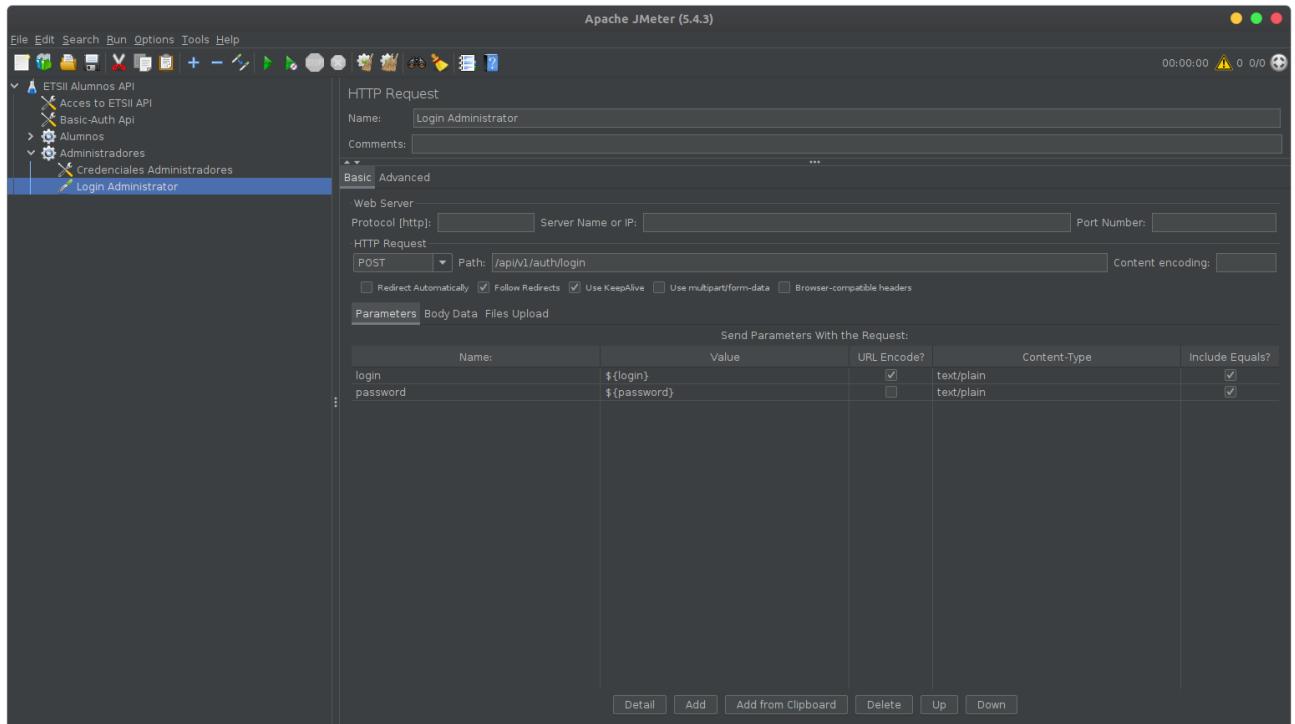
- Antes que nada, vamos a realizar la **autorización para acceder a la API**. Para ello, añadimos un gestor de autorización http haciendo click derecho en *ETSII Alumnos API>Add>Config Element>HTTP Authorization Manager* y lo colocamos debajo de *Acces to ETSII API*. Debe quedar tal que así:



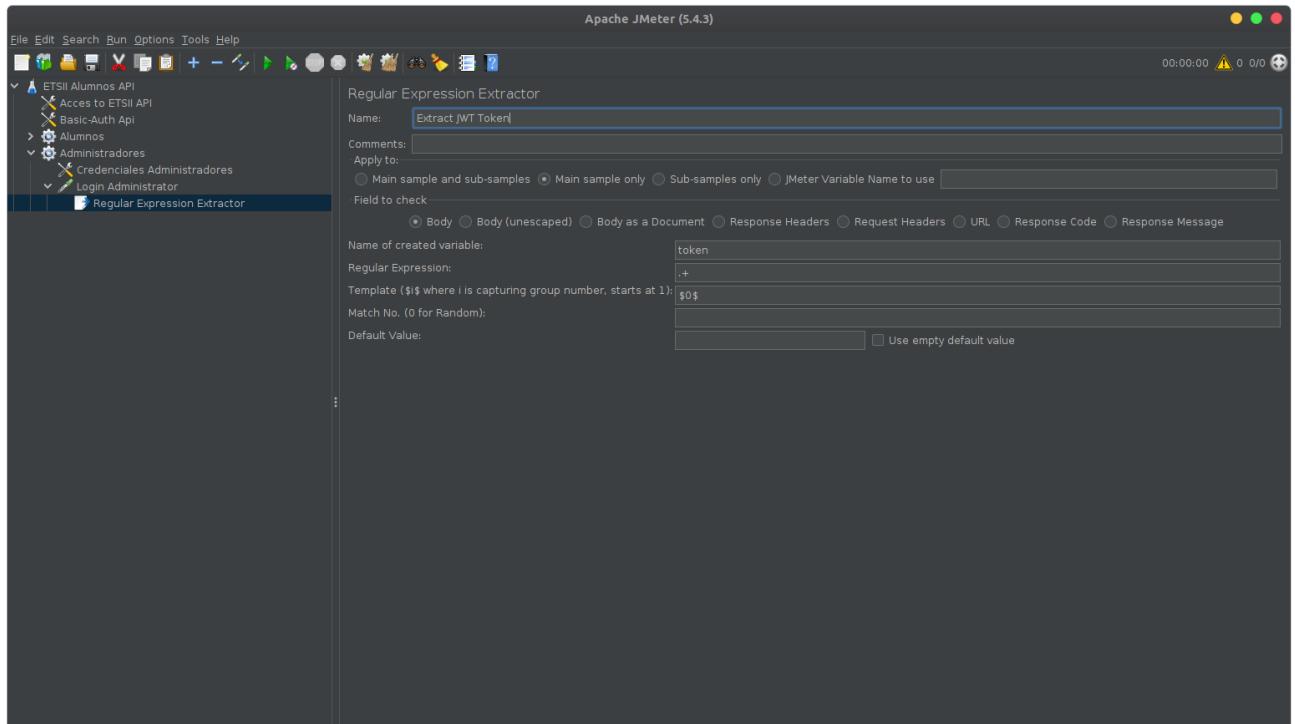
- Ahora sí, volvemos con los administradores. Estos serán parecidos a los alumnos. Empezamos con el **ingreso de las credenciales de los administradores**. Esto lo haremos igual que con los alumnos. Comenzamos haciendo click derecho sobre **Administradores>Add>Config Element>CSV Data Set Config** y los configuramos análogamente que para los alumnos, pero esta vez con el fichero `administradores.csv`, el cual debemos descargar de git. Al final debe quedar así:



- A continuación, hacemos el **login de los administradores**. Para ello, hacemos click derecho sobre **Administradores>Add>Sampler>HTTP Request** y lo hacemos análogamente al de alumnos:

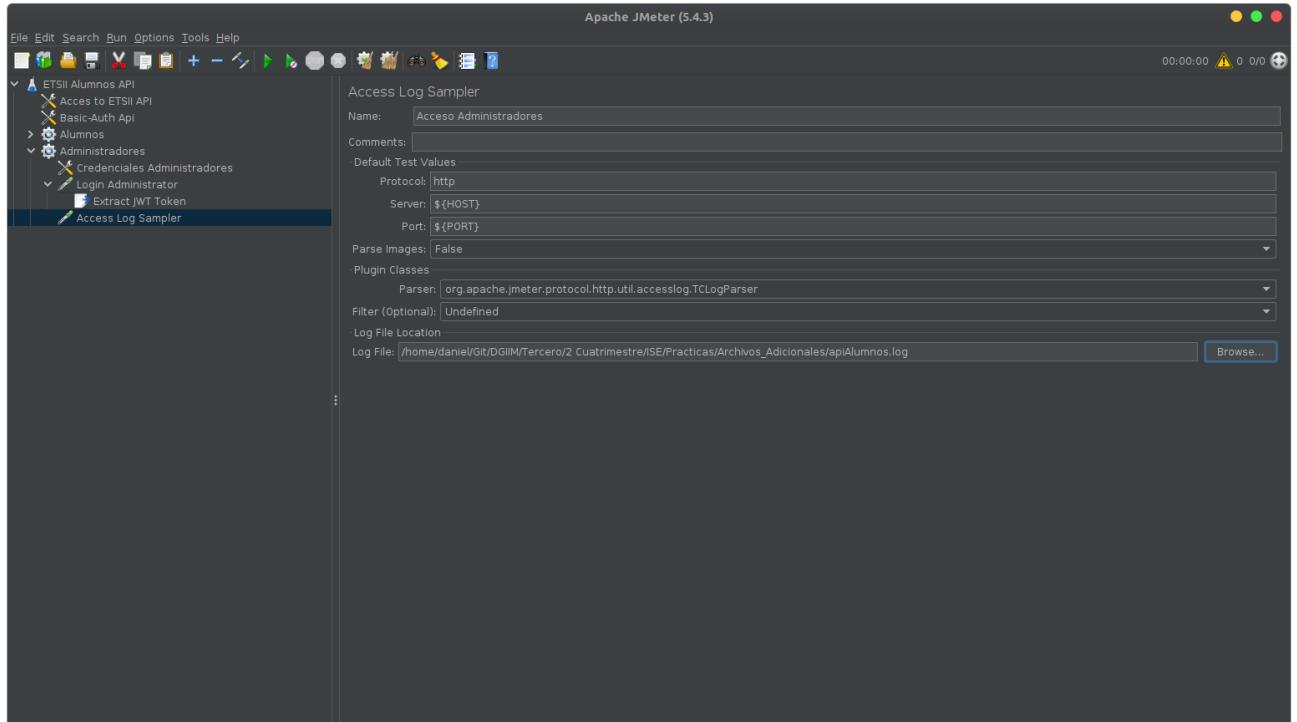


- Hecho esto, tenemos que **obtener y almacenar el JWT** que recibe tras el login. Para ello, click derecho en *Login Administradores>Add>Post Processor>Regular Expression Extractor*. Nuevamente, lo hacemos análogo al de alumnos, por lo que debe quedar algo así:

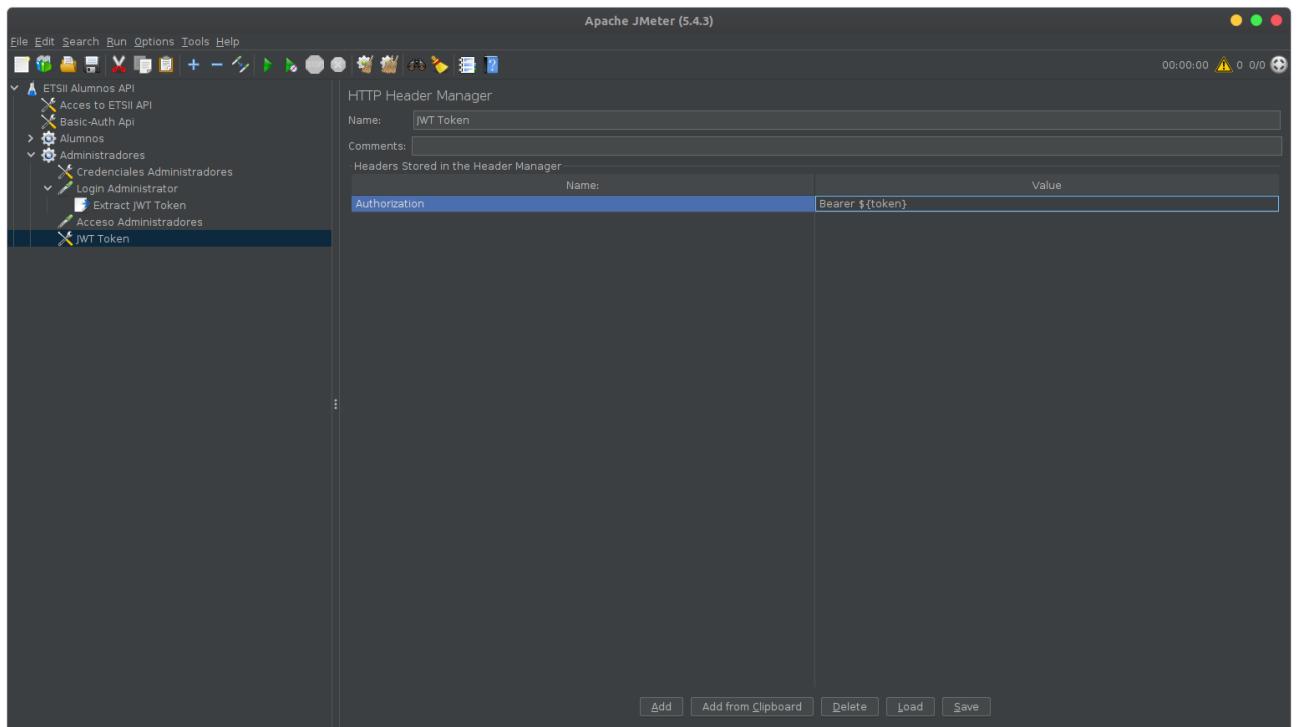


- A continuación, hacemos el **Acceso de los Administradores**, para obtener los datos de diversos alumnos. Esto lo hacemos mediante un muestrador de acceso a log, que se añade haciendo click derecho sobre *Administradores>Add>Sampler>Access Log Sampler*.

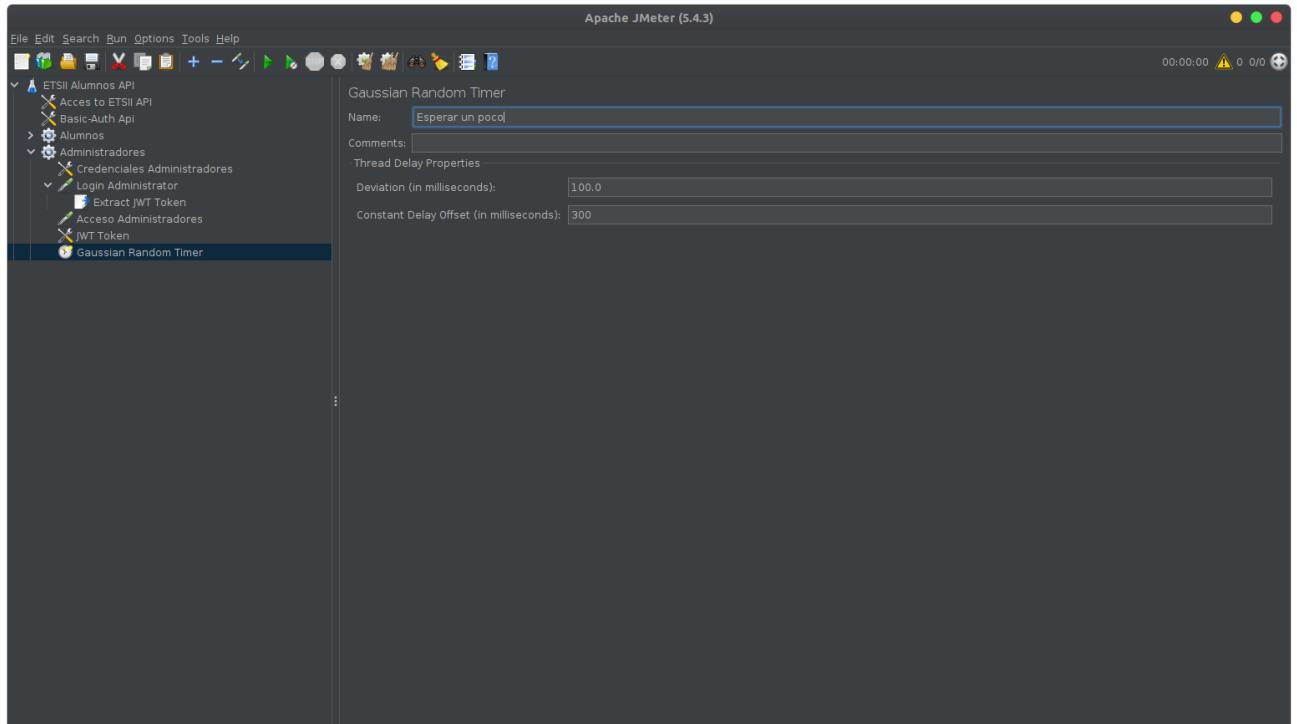
Tenemos que indicarle un Archivo de Log, el cual descargamos de git y tiene el nombre `apiAlumnos.log`. Al final debe quedarnos algo así:



- Al igual que los alumnos, tenemos que añadir un **gestor de cabecera que usará el JWT**. Para añadirlo, hacemos click derecho en *Administradores>Add>Config Element>HTTP Header Manager* y lo hacemos análogo al de alumnos:



- Por último con los administradores, añadimos el **Gaussian Random Timer**, es decir, la espera aleatoria. Para ello, click derecho en *Administradores>Add>Timer>Gaussian Random Timer* y lo dejamos como los alumnos:

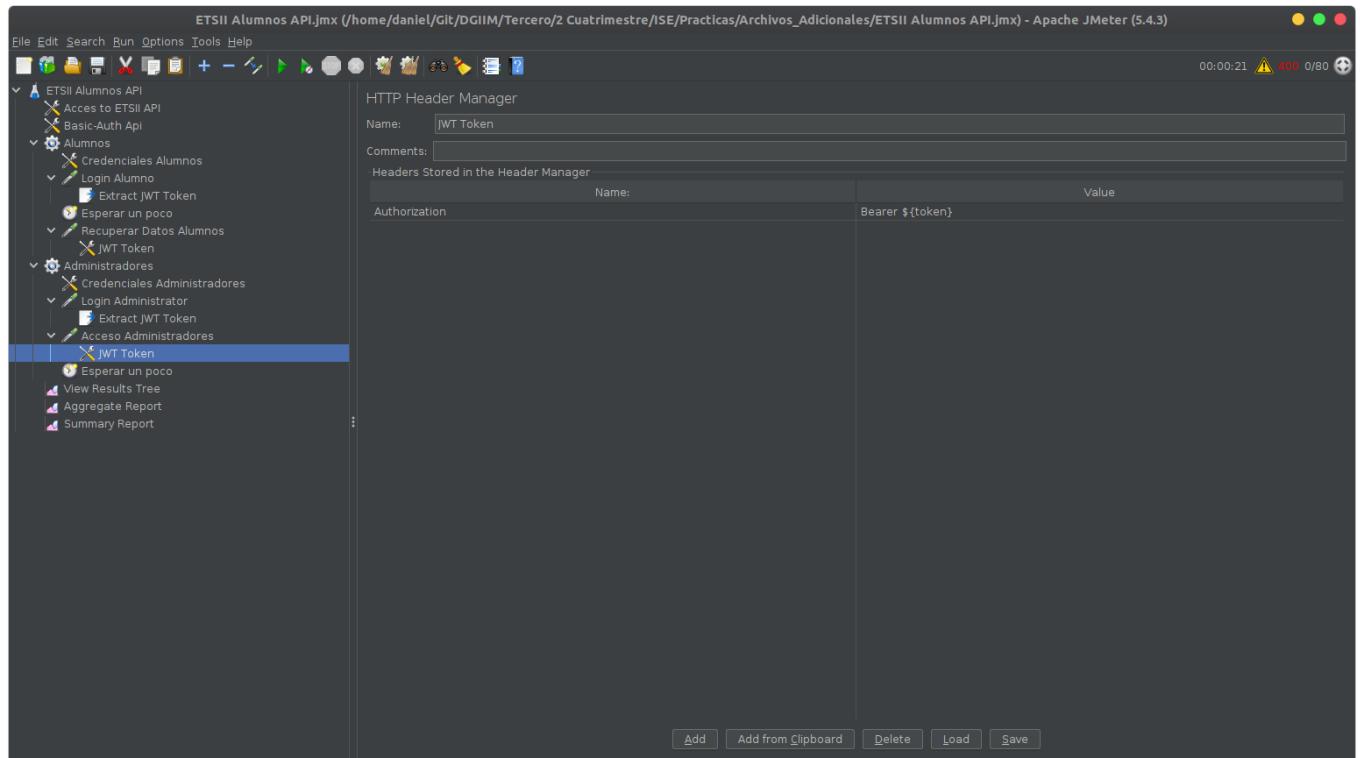


Ya por último, añadimos lo necesario para poder visualizar los resultados. Esto lo hacemos haciendo click derecho en *ETSII Alumnos API>Add>Listener>...*

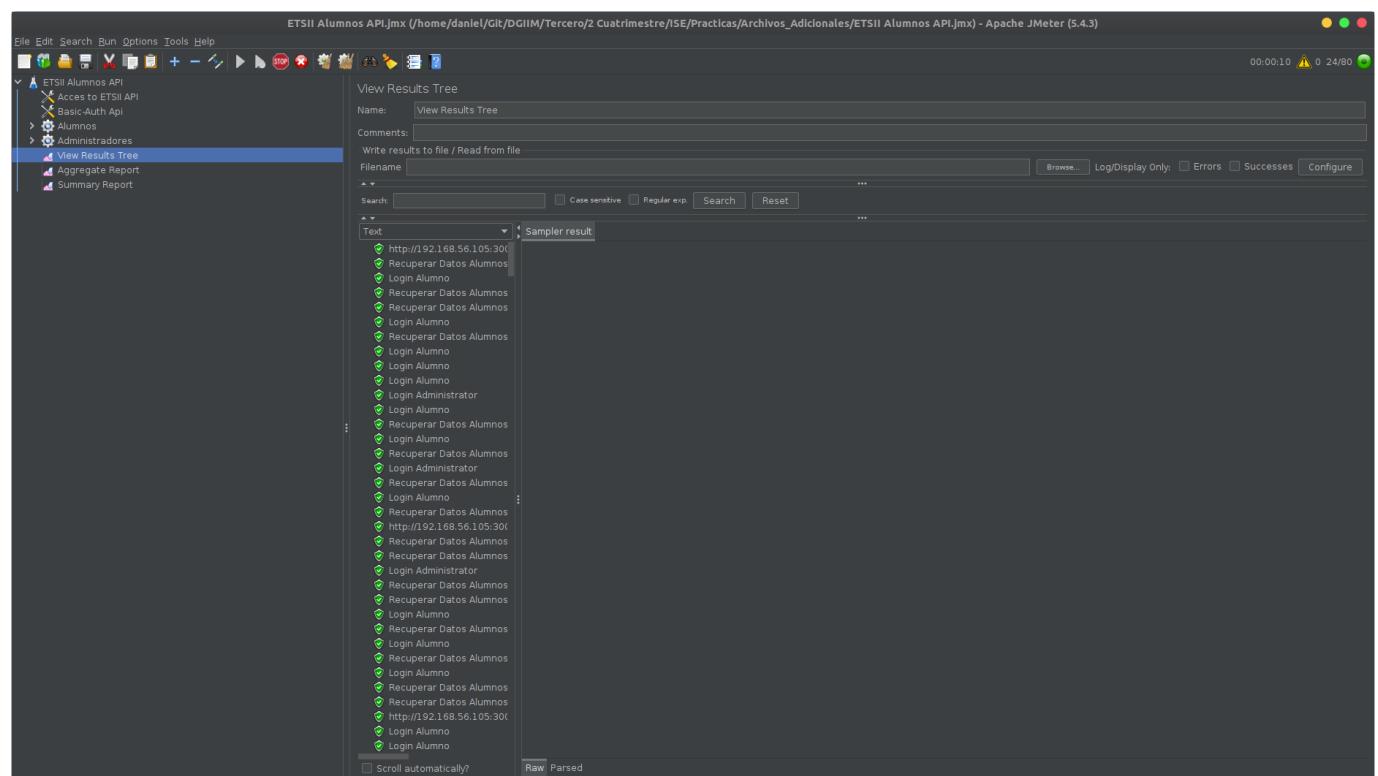
En mi caso he añadido *View Results Tree, Aggregate Report y Summartz Report*.

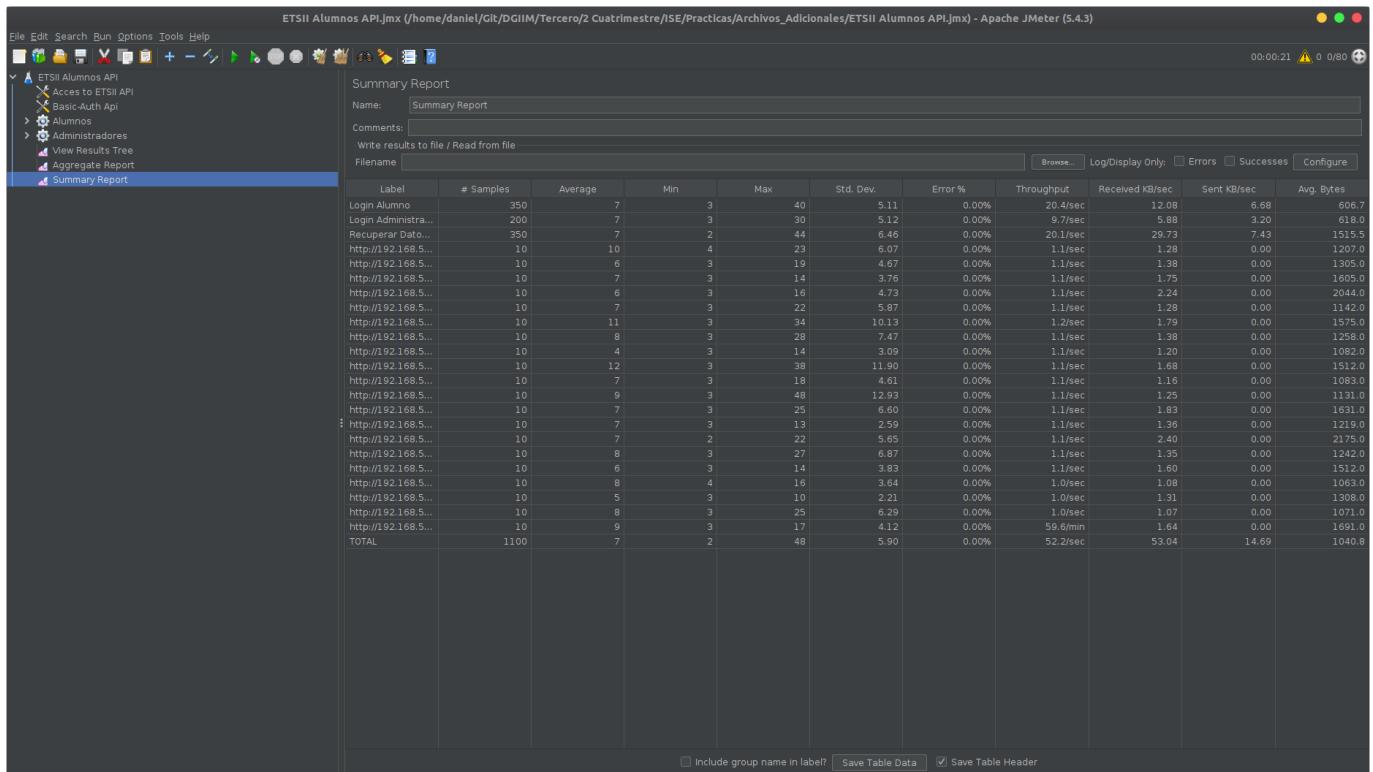
Para mas información de los listeners ver [aqui](#).

Al final, la configuración final nos queda así:



Vamos a probar a ejecutar el test y visualizar los resultados. Para ello, en la barra superior, hacemos click en *Run>Start*:





```
danielav@danielav:~/iseP4JMeter
nodejs_1 | GET /api/v1/alumnos/alumno/dominiquepotts%40tropoli.com 200 6.460 ms - 1438
nodejs_1 | POST /api/v1/auth/login 200 7.016 ms - 185
nodejs_1 | GET /api/v1/alumnos/alumno/sylvianobile%40tropoli.com 200 6.664 ms - 661
nodejs_1 | POST /api/v1/auth/login 200 18.300 ms - 185
nodejs_1 | POST /api/v1/auth/login 200 6.144 ms - 185
nodejs_1 | POST /api/v1/auth/login 200 18.300 ms - 185
nodejs_1 | GET /api/v1/alumnos/alumno/gretchenwilkins%40tropoli.com 200 9.522 ms - 669
nodejs_1 | POST /api/v1/auth/login 200 3.657 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/tracjones%40tropoli.com 200 3.704 ms - 1545
nodejs_1 | GET /api/v1/alumnos/alumno/griffithevans%40tropoli.com 200 6.563 ms - 738
nodejs_1 | POST /api/v1/auth/login 200 6.133 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/kristinearnoldk40tropoli.com 200 2.861 ms - 661
nodejs_1 | GET /api/v1/alumnos/alumno/royweaver%40tropoli.com 200 1.943 ms - 1192
nodejs_1 | GET /api/v1/alumnos/alumno/gretchenwilkins%40tropoli.com 200 1.866 ms - 645
nodejs_1 | POST /api/v1/auth/login 200 2.025 ms - 185
nodejs_1 | POST /api/v1/auth/login 200 2.085 ms - 185
nodejs_1 | POST /api/v1/auth/login 200 5.065 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/blankenshipchapman%40tropoli.com 200 6.083 ms - 1288
nodejs_1 | GET /api/v1/alumnos/alumno/janiekling%40tropoli.com 200 5.956 ms - 840
nodejs_1 | POST /api/v1/auth/login 200 2.016 ms - 185
nodejs_1 | POST /api/v1/auth/login 200 1.844 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/janiekling%40tropoli.com 200 1.884 ms - 840
nodejs_1 | POST /api/v1/auth/login 200 1.993 ms - 185
nodejs_1 | POST /api/v1/auth/login 200 6.546 ms - 1288
nodejs_1 | GET /api/v1/alumnos/alumno/janiekling%40tropoli.com 200 5.982 ms - 1295
nodejs_1 | GET /api/v1/alumnos/alumno/bairdbrady%40tropoli.com 200 6.450 ms - 550
nodejs_1 | GET /api/v1/alumnos/alumno/beatriceparker%40tropoli.com 200 17.991 ms - 697
nodejs_1 | POST /api/v1/auth/login 200 16.147 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/bentleysharp%40tropoli.com 200 1.588 ms - 986
nodejs_1 | GET /api/v1/alumnos/alumno/suttonkenney%40tropoli.com 200 3.046 ms - 1644
nodejs_1 | POST /api/v1/auth/login 200 2.545 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/woodardoneill%40tropoli.com 200 1.729 ms - 1109
nodejs_1 | POST /api/v1/auth/login 200 7.016 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/woodardoneill%40tropoli.com 200 1.958 ms - 1109
nodejs_1 | POST /api/v1/auth/login 200 2.264 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/staceytownsend%40tropoli.com 200 2.066 ms - 669
nodejs_1 | POST /api/v1/auth/login 200 3.032 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/kristinearnoldk40tropoli.com 200 7.312 ms - 661
nodejs_1 | POST /api/v1/auth/login 200 6.854 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/blankenshipchapman%40tropoli.com 200 6.584 ms - 1288
nodejs_1 | POST /api/v1/auth/login 200 6.854 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/bentleysharp%40tropoli.com 200 1.974 ms - 986
nodejs_1 | GET /api/v1/alumnos/alumno/kristinearnoldk40tropoli.com 200 6.824 ms - 661
nodejs_1 | POST /api/v1/auth/login 200 2.024 ms - 196
nodejs_1 | POST /api/v1/auth/login 200 4.862 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/staceytownsend%40tropoli.com 200 1.788 ms - 669
nodejs_1 | GET /api/v1/alumnos/alumno/bentleysharp%40tropoli.com 200 5.842 ms - 986
nodejs_1 | POST /api/v1/auth/login 200 7.558 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/blankenshipchapman%40tropoli.com 200 6.520 ms - 1288
nodejs_1 | POST /api/v1/auth/login 200 7.237 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/staceytownsend%40tropoli.com 200 6.159 ms - 669
nodejs_1 | POST /api/v1/auth/login 200 6.277 ms - 196
nodejs_1 | GET /api/v1/alumnos/alumno/blankenshipchapman%40tropoli.com 200 6.894 ms - 1288
mongod_1 | {"t":{"$date":"2022-05-23T16:39:12.053+00:00"},"s":"I", "c": "STORAGE", "id":22430, "ctx":"Checkpoint", "msg": "WiredTiger message", "attr":{"message": "[1653323952:53421][1:0x7f7d3cd0700]", "WT_SESSION.checkpoint": [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 2882, snapshot max: 2882 snapshot count: 0, oldest timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 1"}}
```

Con esto damos por finalizado el ejercicio.