

BENCHMARKING Y AJUSTE DEL SISTEMA

PHORONIX

Ejercicio: una vez que haya indagado sobre los benchmarks disponibles, seleccione como mínimo dos de ellos y proceda a ejecutarlos en Ubuntu y CentOS. Comente las diferencias.

Documentación: <https://www.phoronix-test-suite.com/documentation/phoronix-test-suite.pdf>

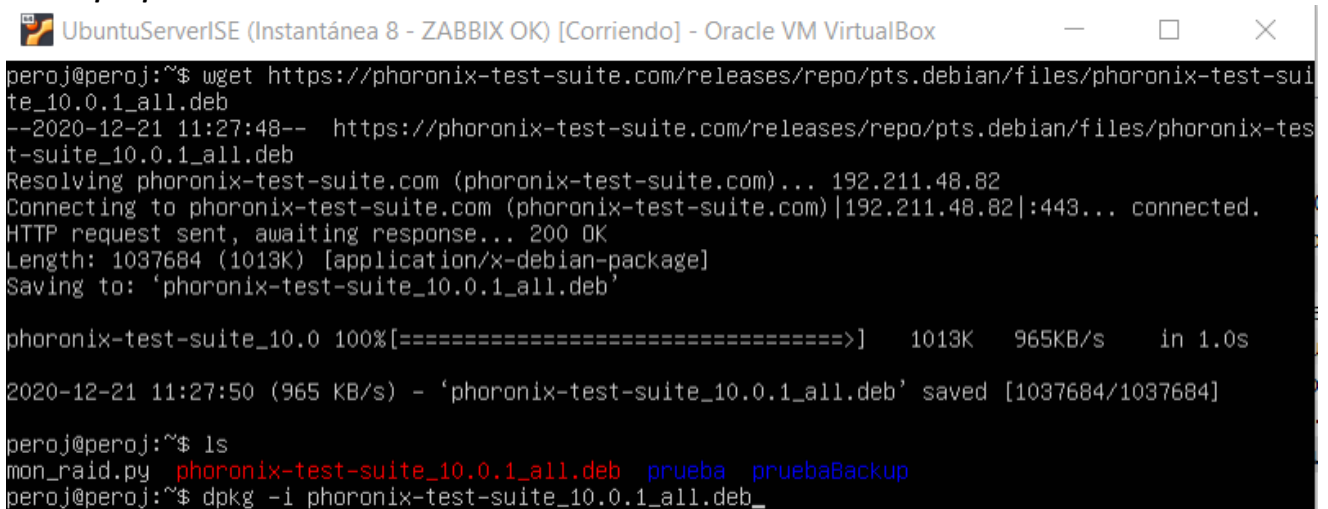
Phoronix es una plataforma que permite ejecutar un conjunto de benchmarks bajo agrupación openbenchmarking.org. En esta organización existen benchmarkings de todo tipo (videojuegos,cpu,memoria...) y mediante la herramienta Phoroxix podremos lanzar todos estos benchmarks.

Para proceder con la instalación en las nuevas distribuciones Linux de 2020 no podemos hacerlo desde los gestores de paquetes dado que no contiene los repositorios. Por lo que, para instalarlo, tanto en Ubuntu Server como en CentOS, deberemos descargarnos los paquetes desde el siguiente enlace de Phoronix ejecutando los siguientes comandos:

wget https://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_10.0.1_all.deb

dpkg -i *.deb

apt update



```

peroj@peroj:~$ wget https://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_10.0.1_all.deb
--2020-12-21 11:27:48-- https://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_10.0.1_all.deb
Resolving phoronix-test-suite.com (phoronix-test-suite.com)... 192.211.48.82
Connecting to phoronix-test-suite.com (phoronix-test-suite.com)|192.211.48.82|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1037684 (1013K) [application/x-debian-package]
Saving to: 'phoronix-test-suite_10.0.1_all.deb'

phoronix-test-suite_10.0 100%[=====] 1013K 965KB/s in 1.0s
2020-12-21 11:27:50 (965 KB/s) - 'phoronix-test-suite_10.0.1_all.deb' saved [1037684/1037684]

peroj@peroj:~$ ls
mon RAID.py phoronix-test-suite_10.0.1_all.deb prueba pruebaBackup
peroj@peroj:~$ dpkg -i phoronix-test-suite_10.0.1_all.deb_

```

```

peroj@peroj:~$ sudo dpkg -i phoronix-test-suite_10.0.1_all.deb
[sudo] password for peroj:
Selecting previously unselected package phoronix-test-suite.
(Reading database ... 14472 files and directories currently installed.)
Preparing to unpack phoronix-test-suite_10.0.1_all.deb ...
Unpacking phoronix-test-suite (10.0.1) ...
Setting up phoronix-test-suite (10.0.1) ...
Processing triggers for mime-support (3.64ubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
peroj@peroj:~$ sudo apt update

```

Procedemos de igual manera para la instalación en CentOS para posteriormente instalar y probar algunos benchmarks en ambas máquinas.

```
CentOS-LVM (Instantánea 6 - ZABBIX AGENT OK) [Corriendo] - Oracle VM VirtualBox
Descargando paquetes:
wget-1.19.5-10.el8.x86_64.rpm                2.4 MB/s | 734 kB      00:00
-----
Total                                         1.5 MB/s | 734 kB      00:00
Ejecutando verificación de operación
Verificación de operación exitosa.
Ejecutando prueba de operaciones
Prueba de operación exitosa.
Ejecutando operación
Preparando : 1/1
Instalando : wget-1.19.5-10.el8.x86_64 1/1
Ejecutando scriptlet: wget-1.19.5-10.el8.x86_64 1/1
Verificando : wget-1.19.5-10.el8.x86_64 1/1

Instalado:
  wget-1.19.5-10.el8.x86_64

¡Listo!
[pero.j@localhost ~]$ wget https://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_10.0.1_all.deb
--2020-12-21 06:39:02-- https://phoronix-test-suite.com/releases/repo/pts.debian/files/phoronix-test-suite_10.0.1_all.deb
Resolviendo phoronix-test-suite.com (phoronix-test-suite.com)... 192.211.48.82
Conectando con phoronix-test-suite.com (phoronix-test-suite.com)[192.211.48.82]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 1037684 (1013K) [application/x-debian-package]
Grabando a: "phoronix-test-suite_10.0.1_all.deb"

phoronix-test-suite_10.0 100%[=====>] 1013K 1001KB/s en 1,0s
2020-12-21 06:39:04 (1001 KB/s) - "phoronix-test-suite_10.0.1_all.deb" guardado [1037684/1037684]

[pero.j@localhost ~]$ ls
mon RAID.py phoronix-test-suite_10.0.1_all.deb UbuntuBackup
```

A parte de instalar las herramientas *wget* y *dpkg*, al hacer *dpkg -i phoronix...deb* vemos que en este caso nos da error debido a que no están instaladas las dependencias "*php-cli*" ni "*php-xml*".

```
Instalado:
  dpkg-1.18.25-12.el8.x86_64

¡Listo!
[pero.j@localhost ~]$ sudo dpkg -i phoronix-test-suite_10.0.1_all.deb
Seleccionando el paquete phoronix-test-suite previamente no seleccionado.
(Leyendo la base de datos ... 0 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar phoronix-test-suite_10.0.1_all.deb ...
Desempaquetando phoronix-test-suite (10.0.1) ...
dpkg: problemas de dependencias impiden la configuración de phoronix-test-suite:
 phoronix-test-suite depende de php-cli | php5-cli; sin embargo:
   El paquete `php-cli' no está instalado.
   El paquete `php5-cli' no está instalado.
 phoronix-test-suite depende de php5-cli | php-xml; sin embargo:
   El paquete `php5-cli' no está instalado.
   El paquete `php-xml' no está instalado.

dpkg: error al procesar el paquete phoronix-test-suite (--install):
problemas de dependencias - se deja sin configurar
Se encontraron errores al procesar:
 phoronix-test-suite
[pero.j@localhost ~]$
```

Pasamos a instalarlas:

```
[peroj@localhost ~]$ sudo yum install php-cli php-xml
Última comprobación de caducidad de metadatos hecha hace 0:01:24, el lun 21 dic 2020 06:45:55 EST.
El paquete php-cli-7.2.24-1.module_el8.2.0+313+b04d0a66.x86_64 ya está instalado.
Dependencias resueltas.
=====
Paquete      Arquitectura Versión      Repositorio  Tam.
=====
Instalando:
php-xml      x86_64      7.2.24-1.module_el8.2.0+313+b04d0a66  AppStream    188 k
Instalando dependencias:
libxslt      x86_64      1.1.32-5.el8  BaseOS       250 k
Resumen de la transacción
=====
Instalar 2 Paquetes

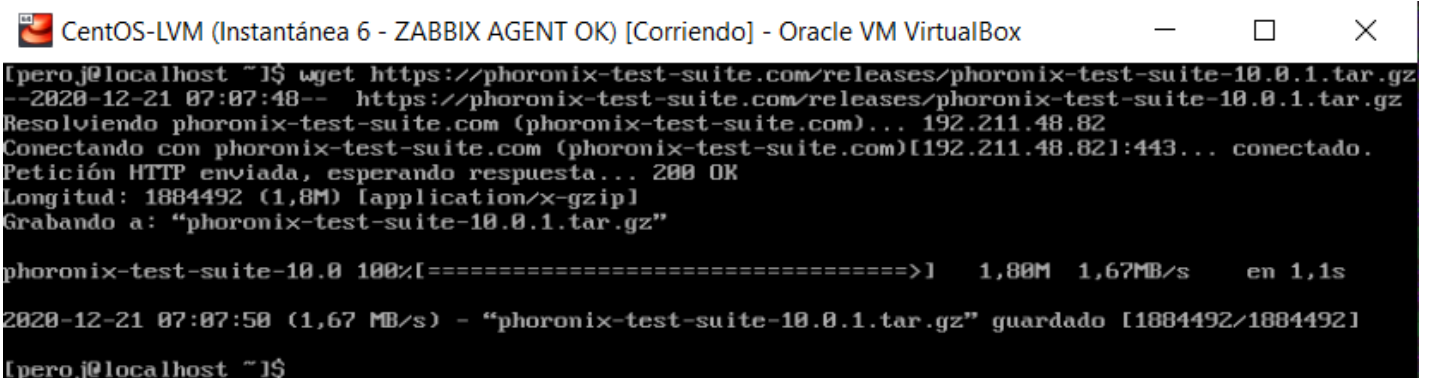
Tamaño total de la descarga: 437 k
Tamaño instalado: 1.2 M
¿Está de acuerdo [s/N]? : s
```

Hacemos un **`sudo yum update`** y volvemos a usar el **`dpkg -i`** y vemos que nos sigue dando el mismo error, seguramente porque no es capaz de procesar ciertas dependencias. Por lo que probamos con **`tar xvfz`** y vemos que nos da error por el formato .zip. Por lo que pasamos a instalar la herramienta **`bzip2`**.

```
[root@localhost perojl# ls
mon RAID.py phoronix-test-suite-10.0.1_all.deb UbuntuBackup
[root@localhost perojl# tar xvfz phoronix*.deb
gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now
[root@localhost perojl# yum install bzip2
```

Nota: me acabo de dar cuenta de que descargué el .deb y procedí como en Ubuntu con **`dpkg`**, por lo que debo volver a descargar con **`wget`** el paquete pero esta vez en .tar.gz para proceder correctamente con **`tar xvfz phoronix*.tar.gz`**.

El paquete para CentOS está en este caso en: <https://phoronix-test-suite.com/releases/phoronix-test-suite-10.0.1.tar.gz>




```
CentOS-LVM (Instantánea 6 - ZABBIX AGENT OK) [Corriendo] - Oracle VM VirtualBox
[peroj@localhost ~]$ wget https://phoronix-test-suite.com/releases/phoronix-test-suite-10.0.1.tar.gz
--2020-12-21 07:07:48-- https://phoronix-test-suite.com/releases/phoronix-test-suite-10.0.1.tar.gz
Resolviendo phoronix-test-suite.com (phoronix-test-suite.com)... 192.211.48.82
Conectando con phoronix-test-suite.com (phoronix-test-suite.com)[192.211.48.82]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 1884492 (1,8M) [application/x-gzip]
Grabando a: "phoronix-test-suite-10.0.1.tar.gz"


phoronix-test-suite-10.0 100%[=====>] 1,80M 1,67MB/s en 1,1s
2020-12-21 07:07:50 (1,67 MB/s) - "phoronix-test-suite-10.0.1.tar.gz" guardado [1884492/1884492]
[peroj@localhost ~]$
```

Descomprimos con **`tar xvfz phoronix*.tar.gz`**.

E instalamos con:

 CentOS-LVM (Instantánea 6 - ZABBIX AGENT OK) [Corriendo] - Oracle VM VirtualBox

```
[root@localhost perojl# cd phoronix-test-suite
[root@localhost phoronix-test-suite]# ./install-sh
```

 CentOS-LVM (Instantánea 6 - ZABBIX AGENT OK) [Corriendo] - Oracle VM VirtualBox

```
[root@localhost perojl# cd phoronix-test-suite
[root@localhost phoronix-test-suite]# ./install-sh
which: no xdg-mime in (/sbin:/bin:/usr/sbin:/usr/bin)

Phoronix Test Suite Installation Completed

Executable File: /usr/bin/phoronix-test-suite
Documentation: /usr/share/doc/phoronix-test-suite/
Phoronix Test Suite Files: /usr/share/phoronix-test-suite/

[root@localhost phoronix-test-suite]# _
```

Después de echarle un ojo a los benchmarks, he decidido probar con *sudokut* que es un test dedicado a estresar la CPU mediante la resolución de sudokus y con *ramspeed* que pone a prueba el rendimiento de la memoria RAM del sistema.

Es muy importante, al menos en este momento, no ejecutar ningún benchmark categorizado con *disk* dado a que puede estropear los discos duros físicos de nuestro pc (aunque estemos trabajando con máquinas virtuales).

-En CentOS:

-ramspeed: <http://www.alasir.com/software/ramspeed/>

Nos damos cuenta de que nos falta la extensión JSON para php. Por lo que la instalamos antes de instalar los test con `sudo yum install php-json`.

Nos descargamos el test *ramspeed*:

```
[perojl@localhost ~]$ phoronix-test-suite install pts/ramspeed

NOTICE: The following PHP extensions are OPTIONAL but recommended:
GD      The GD library is recommended for improved graph rendering.
POSIX    POSIX support is highly recommended.

Phoronix Test Suite v10.0.1
User Agreement

Phoronix Test Suite User Agreement / Notices:

- The Phoronix Test Suite is open-source and licensed under the GNU GPLv3. A copy of the GPLv3
  license is included with this software or can also be found at
  https://www.gnu.org/licenses/gpl-3.0.en.html. However, some tests supported by the Phoronix Test
  Suite are not open-source software or require commercial software packages.

- The Phoronix Test Suite contains tests which may stress your system and in some cases could
  exhibit stability problems of the system's hardware or software configuration. The Phoronix Test
  Suite is provided WITHOUT ANY WARRANTY. In no event shall OpenBenchmarking.org, Phoromatic,
  Phoronix Media, the Phoronix 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.

- 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.

- Anonymous usage reporting / statistics: If enabling the anonymous usage reporting / statistics
```

Aceptamos los términos de uso y recopilación de datos anónimos y comenzará a instalarse el test.

```
Updated Suite Available: pts/vulkan-compute v1.0.3
Updated OpenBenchmarking.org Repository Index
system: 36 Distinct Tests, 98 Test Versions, 0 Suites
Changes Since 21 October To 21 December
Updated Test Available: system/darktable v1.0.5
Updated Test Available: system/selenium v1.0.20
Updated OpenBenchmarking.org Repository Index
git: 8 Distinct Tests, 9 Test Versions, 0 Suites
Evaluating External Test Dependencies .....

The following dependencies are needed and will be installed:

- gcc
- gcc-c++
- make
- autoconf
- automake
- patch
- expat-devel

This process may take several minutes.
```

Una vez instalado lo ejecutamos como sigue y elegimos la primera de las opciones (por copia) y tanto de valores enteros como en coma flotante (3).

```
[peroj@localhost ~]$ phoronix-test-suite run pts/ramspeed

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.3
Memory Test Configuration
 1: Copy
 2: Scale
 3: Add
 4: Triad
 5: Average
 6: Test All Options
*** Multiple items can be selected, delimit by a comma. ***
Type: 1

 1: Integer
 2: Floating Point
 3: Test All Options
*** Multiple items can be selected, delimit by a comma. ***
Benchmark: 3
```

Nos enseña el sistema donde se lanzará el test y nos pregunta si queremos guardar los resultados del test, le decimos que si e introducimos el nombre del archivo y descripción si lo vemos necesario.

```

Phoronix Test Suite v10.0.1
System Information

PROCESSOR:           Intel Core i7-7700HQ
  Core Count:         1
  Extensions:         SSE 4.2 + AUX2 + AUX + RDRAND + FSGSBASE
  Cache Size:         6144 MB
  Core Family:        Kaby/Coffee/Whiskey Lake

GRAPHICS:             VMware SUGA II
  Screen:             2048x2048

MOTHERBOARD:          Oracle VirtualBox v1.2
  BIOS Version:        VirtualBox
  Chipset:             Intel 440FX 82441FX PMC
  Audio:              Intel 82801AA AC 97 Audio
  Network:            2 x Intel 82540EM

MEMORY:               4096MB

DISK:                 2 x 9GB VBox HDD
  File-System:         xfs
  Mount Options:       attr2 inode64 noquota relatime rw seclabel
  Disk Scheduler:      MQ-DEADLINE

OPERATING SYSTEM:     CentOS Linux 8
  Kernel:              4.18.0-193.el8.x86_64 (x86_64)

System Layer:         Oracle VMware

Security:             SELinux

```

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:

```

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.3 [Type: Copy - Benchmark: Integer]
Test 1 of 2
Estimated Trial Run Count:    3
Estimated Test Run-Time:    11 Minutes
Estimated Time To Completion: 22 Minutes [08:09 EST]
Started Run 1 @ 07:48:19_

```

Ejecutará el benchmark durante unos minutos y nos mostrará los resultados.

```

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.3 [Type: Copy - Benchmark: Floating Point]
Test 2 of 2
Estimated Trial Run Count:    3
Estimated Time To Completion: 11 Minutes [08:07 EST]
Started Run 1 @ 07:56:47
Started Run 2 @ 07:59:47
Started Run 3 @ 08:02:29
Started Run 4 @ 08:05:11 *
Started Run 5 @ 08:07:53 *
Started Run 6 @ 08:10:37 *
Started Run 7 @ 08:13:18 *
Started Run 8 @ 08:15:59 *
Started Run 9 @ 08:18:40 *
Started Run 10 @ 08:21:21 *
Started Run 11 @ 08:24:02 *
Started Run 12 @ 08:26:44 *

Type: Copy - Benchmark: Floating Point:
9330.64
10387.56

```



```

Processor: Intel Core i7-7700HQ (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel
l 440FX 82441FX PMC, Memory: 4096MB, Disk: 2 x 9GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel
82801AA AC 97 Audio, Network: 2 x Intel 82540EM

OS: CentOS Linux 8, Kernel: 4.18.0-193.el8.x86_64 (x86_64), File-System: xfs, Screen Resolut
ion: 2048x2048, System Layer: Oracle VMware

RAMspeed SMP 3.5.0
Type: Copy - Benchmark: Integer
MB/s > Higher Is Better
ramspeed1 . 10068.96 |=====

RAMspeed SMP 3.5.0
Type: Copy - Benchmark: Floating Point
MB/s > Higher Is Better
ramspeed1 . 10295.87 |=====

```

Podemos consultar el histórico de los resultados de los test realizados que se guarda por defecto en la siguiente carpeta:

1000

Podemos crear una carpeta en `/var/www/html/phoronix` y copiar dentro los resultados de los test, y como tenemos el servidor web activo, si en el navegador del host buscamos la `ip/phoronix/test` podremos ver los resultados gráficamente.

```
[peroj@localhost ramspeed1centos]$ sudo cp -r /home/peroj/.phoronix-test-suite/test-results/ramspeed1centos /var/www/html/phoronix
```

```
[peroj@localhost ~]$ ls /var/www/html/phoronix/
ramspeed1centos
```

Levantamos el servicio `httpd` y comprobamos desde el navegador de la máquina anfitrión:

Index of /phoronix/ramspeed1centos

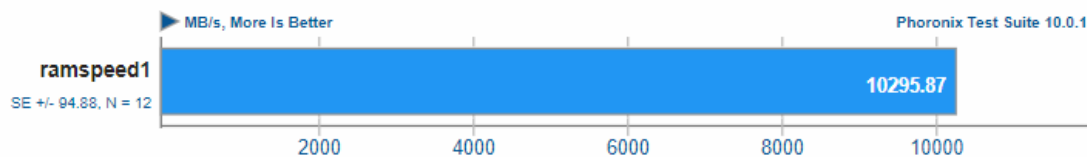
Name	Last modified	Size	Description
Parent Directory		-	
composite.xml	2020-12-21 10:18	2.9K	
index.html	2020-12-21 10:18	9.2K	
installation-logs/	2020-12-21 10:18	-	
result-graphs/	2020-12-21 10:18	-	
system-logs/	2020-12-21 10:18	-	
test-logs/	2020-12-21 10:18	-	

Vemos los resultados, por ejemplo de coma flotante:

RAMspeed SMP 3.5.0

Type: Copy - Benchmark: Floating Point

pts



1. (CC) gcc options: -O3 -march=native

-sudokut: <https://sourceforge.net/projects/sudokut/>

Procedemos con la instalación y ejecución del test como con el anterior caso. Instalamos con *phoronix-test-suite install pts/sudokut* y ejecutamos con la opción *run*.

```

Sudokut 0.4:
pts/sudokut-1.0.1
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 2 Minutes [10:40 EST]
  Started Run 1 @ 10:39:15
  Started Run 2 @ 10:40:05
  Started Run 3 @ 10:40:53
  Started Run 4 @ 10:41:42 *

Total Time:
46.693
44.075
44.412
45.892

Average: 45.268 Seconds
Deviation: 2.73%
Samples: 4

Result compared to 9,983 OpenBenchmarking.org samples since 26 February 2011: median result: 30.
63. Box plot of samples:
[ |x-----x-----x-----x-----x-----x-----x-####x-----x- * ]
                                This Result (15th Percentile): 45.268 ^
                                ARMv7 rev 3: 75 ^ Intel Core i7-9700K: 8.627 ^
                                ARMv7 rev 2: 130 ^ 2 x Intel Xeon Gold 6138: 12.3 ^
                                ARMv7 rev 4: 143 ^ 2 x Intel Xeon E5-2628 v3: 16.86 ^

Do you want to view the text results of the testing (Y/n): y
sudokut1centos

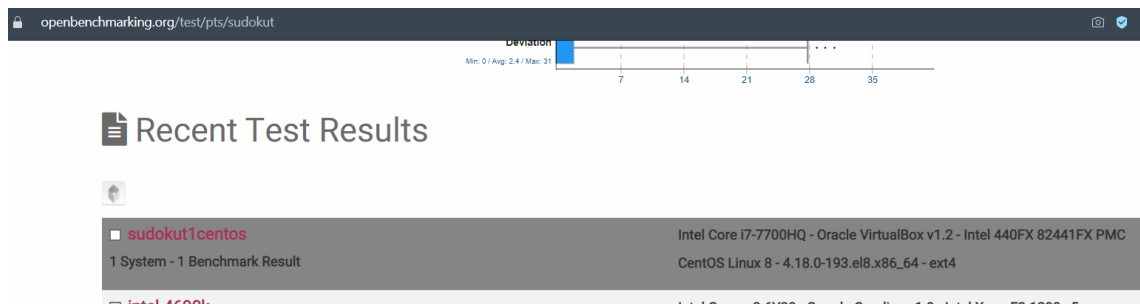
sudokut1centos:
Processor: Intel Core i7-7700HQ (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Inte
l 440FX 82441FX PMC, Memory: 4096MB, Disk: 2 x 9GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel
82801AA AC 97 Audio, Network: 2 x Intel 82540EM

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

Sudokut 0.4
Total Time
Seconds < Lower Is Better
sudokut1centos . 45.27 |=====

```

También podemos visualizar los resultados desde un navegador como en el caso anterior copiando los resultados del test. Pero tenemos la alternativa de poderlos ver desde la página de la organización de openbenchmarking.org/test/pts/sudokut.



-En Ubuntu Server:

-ramspeed:

Lo instalamos con ***phoronix-test-suite install pts/ramspeed*** como hicimos anteriormente. Esto nos llevará un par de minutos después de introducir la contraseña de usuario.



UbuntuServerISE (Instantánea 8 - ZABBIX OK) [Corriendo] - Oracle VM VirtualBox

```
peroj@peroj:~$ phoronix-test-suite install pts/ramspeed

NOTICE: The following PHP extensions are OPTIONAL but recommended:

Bzip2      The bzcompress/bzip2 support can be used for greater file compression.
SQLite3    SQLite3 is required when running a Phoromatic server.
CURL       CURL is recommended for an enhanced download experience.

Phoronix Test Suite v10.0.1
User Agreement

Phoronix Test Suite User Agreement / Notices:
```

Y lo ejecutamos de igual manera con la opción ***run*** y vemos también la información sobre el sistema donde se ejecuta:

```
peroj@peroj:~$ phoronix-test-suite run pts/ramspeed

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.3
Memory Test Configuration
  1: Copy
  2: Scale
  3: Add
  4: Triad
  5: Average
  6: Test All Options
  ** Multiple items can be selected, delimit by a comma. **
Type: 1

  1: Integer
  2: Floating Point
  3: Test All Options
  ** Multiple items can be selected, delimit by a comma. **
Benchmark: 3_
```

```

UbuntuServerISE (Instantánea 8 - ZABBIX OK) [Corriendo] - Oracle VM VirtualBox

Phoronix Test Suite v10.0.1
System Information

PROCESSOR:      Intel Core i7-7700HQ
Core Count:     1
Extensions:     SSE 4.2 + AVX2 + AVX + RDRAND + FSGSBASE
Cache Size:     6 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 82801AA AC 97 Audio
Network:        2 x Intel 82540EM

MEMORY:         4096MB

DISK:           11GB VBOX HDD
File-System:    ext4
Mount Options:  relatime rw
Disk Scheduler: MQ-DEADLINE

OPERATING SYSTEM: Ubuntu 20.04
Kernel:         5.4.0-52-generic (x86_64)

Compiler:       GCC 9.3.0

System Layer:   Oracle VMWare

Security:       itlb_multihit: KVM: Vulnerable

```

Comienza el test y esperamos unos 22 minutos hasta ver los resultados (también podremos verlos a través de *openbenchmarking.org* o desde nuestro navegador si tenemos el servidor web activo como hicimos en el caso anterior):

```

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.3 [Type: Copy - Benchmark: Integer]
Test 1 of 2
Estimated Trial Run Count:      3
Estimated Test Run-Time:      11 Minutes
Estimated Time To Completion: 22 Minutes [17:08 UTC]
  Started Run 1 @ 16:47:23
  Started Run 2 @ 16:50:13
  Started Run 3 @ 16:52:58

Type: Copy - Benchmark: Integer:
  9834.93
  10244.66
  10136.09

Average: 10071.89 MB/s
Deviation: 2.11%

Result compared to 4,298 OpenBenchmarking.org samples since 26 February 2011; median result: 169
98. Box plot of samples:
[ * |-----#*#####!#####*-----*-----* | * *]
    ^ This Result (25th Percentile): 10072
    ^ 1 x 1024 MB DDR2: 839      8 x 8192 MB DDR4-2400MHz: 23525 ^ 16 x 32 GB DDR4-3200MT: 35624 ^
    ^ 4 x 8192 MB DDR4-3200MT: 33249 ^
    ^ 8 x 16384 MB DDR4-2666MT: 31634 ^

```

```

RAMspeed SMP 3.5.0:
pts/ramspeed-1.4.3 [Type: Copy - Benchmark: Floating Point]
Test 2 of 2
Estimated Trial Run Count: 3
Estimated Time To Completion: 11 Minutes [17:06 UTC]
  Started Run 1 @ 16:55:50
  Started Run 2 @ 16:58:49
  Started Run 3 @ 17:01:55
  Started Run 4 @ 17:04:55 *
  Started Run 5 @ 17:07:49 *
  Started Run 6 @ 17:10:44 *
  Started Run 7 @ 17:13:38 *

Type: Copy - Benchmark: Floating Point:
  9596.69
  8999.56
  9292.25
  9810.82
  9755.14
  9575.9
  9643.28

Average: 9524.81 MB/s
Deviation: 2.99%
Samples: 7

Result compared to 3,354 OpenBenchmarking.org samples since 26 February 2011; median result: 145
89. Box plot of samples:
[      |-----#*#####!#####-----*-----*| * * ]
      ^ This Result (26th Percentile): 9525

```

```

ramspeediubuntu:

Processor: Intel Core i7-7700HQ (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Inte
l 440FX 82441FX PMC, Memory: 4096MB, Disk: 11GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel 828
01AA AC 97 Audio, Network: 2 x Intel 82540EM

OS: Ubuntu 20.04, Kernel: 5.4.0-52-generic (x86_64), Compiler: GCC 9.3.0, File-System: ext4,
Screen Resolution: 2048x2048, System Layer: Oracle VMWare

RAMspeed SMP 3.5.0
Type: Copy - Benchmark: Integer
MB/s > Higher Is Better
ramspeediubuntu . 10071.89 |=====

RAMspeed SMP 3.5.0
Type: Copy - Benchmark: Floating Point
MB/s > Higher Is Better
ramspeediubuntu . 9524.81 |=====

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/2012214-FI-RAMSPED107

```

Vemos que en este caso es un poco mejor el test de memoria cuando copia datos enteros a cuando los copia en coma flotante.

-sudokut:

```

 UbuntuServerISE (Instantánea 8 - ZABBIX OK) [Corriendo] - Oracle VM V
root@peroj:/home/peroj# phoronix-test-suite install pts/sudokut

Updated OpenBenchmarking.org Repository Index
pts: 432 Distinct Tests, 1676 Test Versions, 49 Suites
Changes Since 21 October To 21 December
Updated Test Available: pts/ai-benchmark v1.0.1
Updated Test Available: pts/apache-siege v1.0.5
Updated Test Available: pts/asmfish v1.1.2

The following dependencies are needed and will be installed:
- tcl
- tclsh
- mesa-utils
- vulkan-utils
- unzip
- apt-file

This process may take several minutes.

```

Phoronix Test Suite v10.0.1

```

To Install: pts/sudokut-1.0.1

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

1 Test To Install
  1 File To Download [0.02MB]
  1MB Of Disk Space Is Needed
  2 Seconds Estimated Install Time

pts/sudokut-1.0.1:
  Test Installation 1 of 1
  1 File Needed [0.02 MB]
  Downloading: sudokut0.4-1.tar.bz2 [0.02MB]
  Downloading .....
  Approximate Install Size: 0.1 MB
  Estimated Install Time: 2 Seconds
  Installing Test @ 17:42:54

root@peroj:/home/peroj# _

```

Una vez instalado lo ejecutamos y esperamos un par de minutos:

UbuntuServerISE (Instantánea 8 - ZABBIX OK) [Corriendo] - Oracle VM Vi

```
root@peroj:/home/peroj# phoronix-test-suite run pts/sudokut
```

Phoronix Test Suite v10.0.1
System Information

```

PROCESSOR:          Intel Core i7-7700HQ
  Core Count:        1
  Extensions:        SSE 4.2 + AVX2 + AVX + RDRAND + FSGSBASE
  Cache Size:        6 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 82801AA AC 97 Audio
  Network:            2 x Intel 82540EM

MEMORY:              4096MB

DISK:                 11GB VBOX HDD
  File-System:        ext4
  Mount Options:      relatime rw
  Disk Scheduler:     MQ-DEADLINE

OPERATING SYSTEM:    Ubuntu 20.04
  Kernel:             5.4.0-52-generic (x86_64)

System Layer:        Oracle VMWare

Security:             itlb_multihit: KVM: Vulnerable

```

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:

Sudokut 0.4:

```

pts/sudokut-1.0.1
Test 1 of 1
Estimated Trial Run Count: 3
Estimated Time To Completion: 2 Minutes [17:46 UTC]
  Started Run 1 @ 17:45:10
  Started Run 2 @ 17:45:53
  Started Run 3 @ 17:46:35

```

Total Time:

```

39.237
38.315
38.385

```

Average: 38.646 Seconds

Deviation: 1.33%

Result compared to 9,903 OpenBenchmarking.org samples since 26 February 2011; median result: 30.63. Box plot of samples:

```

[ *-----*-----*-----*-----*-----*-----*-----*-----*-----* ]
                                     This Result (21st Percentile): 38.646 ^
                                AArch64 rev 4: 117 ^ ARMv7 rev 3: 75 ^ Intel Core i7-9700K: 8.627 ^
                                ARMv7 rev 2: 130 ^                      2 x Intel Xeon Gold 6138: 12.3 ^
                                ARMv7 rev 4: 143 ^                      2 x Intel Xeon E5-2620 v3: 16.86 ^

```

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

```

sudokut1ubuntu:

Processor: Intel Core i7-7700HQ (1 Core), Motherboard: Oracle VirtualBox v1.2, Chipset: Intel
1 440FX 82441FX PMC, Memory: 4096MB, Disk: 11GB VBOX HDD, Graphics: VMware SVGA II, Audio: Intel 828
01AA AC 97 Audio, Network: 2 x Intel 82540EM

OS: Ubuntu 20.04, Kernel: 5.4.0-52-generic (x86_64), File-System: ext4, Screen Resolution: 2
048x2048, System Layer: Oracle VMware

Sudokut 0.4
Total Time
Seconds < Lower Is Better
sudokut1ubuntu . 38.65 |=====

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/2012215-FI-SUDOKUT1U37

```

-Comparando resultados

-ramspeed:

CentOS:

Integer 10068.96 Mb/s

Floating Point 10295.87 Mb/s

Ubuntu:

Integer 10071.89 Mb/s

Floating Point 9524.81 Mb/s

Como podemos comprobar y sabiendo que cuanto mayores sean los resultados mejores serán en este tipo de test, en el caso de la ejecución del test para enteros, los resultados son muy parejos entre CentOS y Ubuntu Server. En el caso de las lecturas y escrituras de los datos en coma flotante tiene mejores resultados en CentOS, con una diferencia de unos 750 Mb/s, quedando así un poco por detrás Ubuntu Server en este aspecto del test *ramspeed*.

-sudokut:

CentOS: 45.27 seconds

Ubuntu: 38.65 seconds

En este caso Ubuntu Server nos muestra mejores resultados en este test de estrés al procesador que en CentOS. Cuanto menor sea el tiempo mejor, dado que más rápido resuelve este test.

También podríamos fijarnos en los percentiles y tener una visión más global del rendimiento del sistema ante estos test.

APACHE BENCHMARK Y JMETER

Ejercicio 2: tras probar un test básico para una web [7], utilizaremos Jmeter para hacer un test sobre una aplicación que ejecuta sobre dos contenedores (uno para la BD y otro para la aplicación en sí). El código está disponible en <https://github.com/davidPalomar-ugr/iseP4JMeter> donde se dan detalles sobre cómo ejecutar la aplicación en una de nuestras máquinas virtuales. El test de Jmeter debe incluir los siguientes elementos:

- El test debe tener parametrizados el Host y el Puerto en el Test Plan (puede hacer referencia usando \$param)
- Debe hacer dos grupos de hebras distintos para simular el acceso de los alumnos y los administradores. Las credenciales de alumno y administrador se cogen de los archivos: alumnos.csv y administrador.csv respectivamente.
- Añadimos esperas aleatorias a cada grupo de hebras (Gaussian Random Timer)
- El login de alumno, su consulta de datos (recuperar datos alumno) y login del administrador son peticiones HTTP.
- El muestreo para simular el acceso de los administradores lo debe coger el archivo apiAlumnos.log (usando un Acces Log Sampler)
- Use una expresión regular (Regular Expresión Extractor) para extraer el token JWT que hay que añadir a la cabecera de las peticiones (usando HTTP Header Manager)

Docker

La MV, dada la infraestructura del sistema monta un SO para cada una de las máquinas virtuales con su hardware virtualizado. Sin embargo, docker no, docker usa la infraestructura del sistema y el SO del host para montar la interfaz de docker y aquí desplegar las aplicaciones, así las aplicaciones pueden tener conexión entre ellas de forma nativa (en MV no se podría de forma nativa, si a través de servidores web por ejemplo) y las aplicaciones comparten la misma memoria.

Gracias a docker se han popularizado mucho los microservicios que son pequeños servicios que se abren, hacen lo que tienen que hacer y se cierran, ahorrando así recursos, ya que no tiene que estar siempre activo en estado “ocioso” o esperando peticiones.

Dado al desarrollo de la tecnología web muchas veces no nos interesa tener muchos servicios a la vez, entonces, la solución sería tener los servicios principales activos y si llega una petición que requiere otro servicio en concreto, se levanta este servicio con docker, se devuelve la petición y se vuelve a bajar el servicio (microservicios).

Instalación de Docker en Ubuntu Server:

Añadimos llave GPG para validar el repositorio:

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

UbuntuServerISE (Instantánea 9 - Phoronix y test OK) [Corriendo] - Oracle VM VirtualBox

```
peroj@peroj:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
[sudo] password for peroj:
OK
```

Añadimos repositorio (si no funciona, sustituir `$(lsb_release -cs)` por `focal`):

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

```
peroj@peroj:~$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
Get:1 https://download.docker.com/linux/ubuntu focal InRelease [36.2 kB]
Hit:2 http://es.archive.ubuntu.com/ubuntu focal InRelease
Get:3 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [5637 B]
Get:4 https://download.docker.com/linux/ubuntu focal/stable amd64 Contents (deb) [1324 B]
Get:5 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:6 http://repo.zabbix.com/zabbix/5.0/ubuntu focal InRelease
Get:7 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:8 http://es.archive.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Get:9 http://es.archive.ubuntu.com/ubuntu focal amd64 Contents (deb) [40.9 MB]
Get:10 http://es.archive.ubuntu.com/ubuntu focal-updates amd64 Contents (deb) [23.1 MB]
Get:11 http://es.archive.ubuntu.com/ubuntu focal-backports amd64 Contents (deb) [7902 B]
Get:12 http://es.archive.ubuntu.com/ubuntu focal-security amd64 Contents (deb) [18.1 MB]
Fetched 82.5 MB in 2min 3s (671 kB/s)
Reading package lists... Done
peroj@peroj:~$ _
```

Una vez que tenemos el repositorio añadido, actualizamos los repositorios y procedemos con la instalación de docker.

`sudo apt update`

```
peroj@peroj:~$ sudo apt update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
Hit:2 http://es.archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:4 http://repo.zabbix.com/zabbix/5.0/ubuntu focal InRelease
Hit:5 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:6 http://es.archive.ubuntu.com/ubuntu focal-security InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
119 packages can be upgraded. Run 'apt list --upgradable' to see them.
peroj@peroj:~$ _
```

Buscamos el repositorio docker (community edition) y lo instalamos:

`apt search docker-ce`

`sudo apt install docker-ce`

```
peroj@peroj:~$ apt search docker-ce
Sorting... Done
Full Text Search... Done
docker-ce/focal 5:20.10.1~3-0~ubuntu-focal amd64
  Docker: the open-source application container engine

docker-ce-cli/focal 5:20.10.1~3-0~ubuntu-focal amd64
  Docker CLI: the open-source application container engine

docker-ce-rootless-extras/focal 5:20.10.1~3-0~ubuntu-focal amd64
  Rootless support for Docker.

peroj@peroj:~$ sudo apt install docker-ce_
```

Nos salta un error del *dpkg* por lo que volvemos atrás y lo hacemos con la opción *focal* como hemos visto anteriormente y ya si se instala (aunque me sigue dando algún problema *dpkg*).

Comprobamos que el servicio está activo con *systemctl*.

```
update-initramfs: Generating /boot/initrd.img-5.4.0-53-generic
I: The initramfs will attempt to resume from /dev/dm-3
I: (/dev/mapper/vg0-lv--0--swap)
I: Set the RESUME variable to override this.
Error 24 : Write error : cannot write compressed block
E: mkinitramfs failure cpio 141 lz4 -9 -1 24
update-initramfs: failed for /boot/initrd.img-5.4.0-53-generic with 1.
run-parts: /etc/kernel/postinst.d/initramfs-tools exited with return code 1
dpkg: error processing package linux-image-5.4.0-53-generic (--configure):
 installed linux-image-5.4.0-53-generic package post-installation script subprocess returned error e
xit status 1
Errors were encountered while processing:
 linux-image-5.4.0-53-generic
E: Sub-process /usr/bin/dpkg returned an error code (1)
peroj@peroj:~$ systemctl status docker
• docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-12-22 16:19:37 UTC; 2min 26s ago
   TriggeredBy: • docker.socket
     Docs: https://docs.docker.com
    Main PID: 22953 (dockerd)
      Tasks: 8
     Memory: 40.7M
    CGroup: /system.slice/docker.service
            └─22953 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Para poder usar docker sin tener que ser root, debemos añadir nuestro usuario al grupo docker y volver a logearnos:

sudo usermod -aG docker peroj

```
peroj@peroj:~$ sudo usermod -aG docker peroj
peroj@peroj:~$ exit_
```

Después de relomearnos comprobamos que está instalado con:

docker info

docker run hello-world

```
WARNING: No swap limit support
WARNING: No blkio weight support
WARNING: No blkio weight_device support
peroj@peroj:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:1a523af650137b8accdaed439c17d684df61ee4d74feac151b5b337bd29e7eec
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.
    (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/
```

Instalación de Docker Compose en Ubuntu Server:

Para orquestar contenedores tenemos docker-compose (entre otras herramientas, véase buidah, kubernetes...) que nos permite desplegar aplicaciones con varios contenedores de modo que cada uno tenga una función. Los pasos para instalarlo son:

Como ya tenemos el repo configurado, instalamos con ***apt install docker-compose***.

```
peroj@peroj:~$ sudo apt install docker-compose
[sudo] password for peroj:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
```

Y probamos con:

docker-compose

docker-compose --version

```
Commands:
  build           Build or rebuild services
  bundle          Generate a Docker bundle from the Compose file
  config          Validate and view the Compose file
  create          Create services
  down            Stop and remove containers, networks, images, and volumes
  events          Receive real time events from containers
  exec            Execute a command in a running container
  help            Get help on a command
  images          List images
  kill            Kill containers
  logs            View output from containers
  pause           Pause services
  port            Print the public port for a port binding
  ps             List containers
  pull            Pull service images
  push            Push service images
  restart         Restart services
  rm              Remove stopped containers
  run             Run a one-off command
  scale           Set number of containers for a service
  start           Start services
  stop            Stop services
  top             Display the running processes
  unpause        Unpause services
  up              Create and start containers
  version         Show the Docker-Compose version information
peroj@peroj:~$ docker-compose --version
docker-compose version 1.25.0, build unknown
```

Instalación de la aplicación para el test con JMeter

Una vez tengamos instalado docker y docker-compose comenzamos con el despliegue de la aplicación en nuestro Ubuntu Server, para ello, lo primero que debemos hacer es clonar el repositorio de la aplicación:

git clone <https://github.com/davidPalomar-ugr/iseP4JMeter.git>

Tras esto, tendremos un directorio nuevo: iseP4JMeter al cual podremos acceder y levantar la aplicación por el puerto 3000 con docker-compose:

cd iseP4JMeter

docker-compose up

```
peroj@peroj:~$ git clone https://github.com/davidPalomar-ugr/iseP4JMeter.git
Cloning into 'iseP4JMeter'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 3778 (delta 0), reused 0 (delta 0), pack-reused 3774
Receiving objects: 100% (3778/3778), 7.78 MiB | 4.33 MiB/s, done.
Resolving deltas: 100% (706/706), done.
peroj@peroj:~$ ls
iseP4JMeter  mon RAID.py
peroj@peroj:~$ cd iseP4JMeter/
peroj@peroj:~/iseP4JMeter$ ls
README.md  docker-compose.yml  images  jMeter  mongodb  nodejs  pruebaEntorno.sh
peroj@peroj:~/iseP4JMeter$ docker-compose up
Creating network "iseP4JMeter_default" with the default driver
Pulling mongodb (mongo:)...
latest: Pulling from library/mongo
f22ccc0b8772: Extracting [=====> ] 25.66MB/26.71MB
3cf8fb62ba5f: Download complete
e80c964e6e6a: Download complete
329e632c35b3: Download complete
3e1bd1325a3d: Download complete
4aa6e3d64a4a: Download complete
035bca87b778: Download complete
874e4e43cb00: Download complete
08cb97662b8b: Download complete
f623ce2ba1e1: Downloading [=====] 21.46MB/142.7MB
f100ac278196: Download complete
6f5539f9b3ee: Waiting
```

- Nociones básicas de la aplicación:

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

La API se ha desarrollado empleando: MongoDB, NodeJS y Express

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

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

docker-compose up -d

Para parar la aplicación ejecutar:

docker-compose down

```

root@peroj:/home/peroj/iseP4JMeter# docker-compose up -d
Creating network "iseP4JMeter_default" with the default driver
Creating isep4jmeter_mongodb_1 ... done
Creating isep4jmeter_mongodbin_1 ... done
Creating isep4jmeter_nodejs_1 ... done
root@peroj:/home/peroj/iseP4JMeter# docker-compose down
Stopping isep4jmeter_nodejs_1 ... done
Stopping isep4jmeter_mongodb_1 ... done
Removing isep4jmeter_nodejs_1 ... done
Removing isep4jmeter_mongodbin_1 ... done
Removing isep4jmeter_mongodb_1 ... done
Removing network isep4jmeter_default

```

Docker descargará las imágenes base y construirá las 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 BasicAuth. Una vez autenticado, se obtiene un [JWT](#) Token para ser empleado en el servicio de alumno.
- /alumnos/alumno: Devuelve el registro de 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.

→ ↺ 🏠 192.168.56.105:3000

ETSII Alumnos API

Descripción de la API Restful:

POST /api/v1/auth/login

Parametros:

login:<emailUsuario>

password:<secreto>

Seguridad:

Acceso protegido con BasicAuth (etsiiApi:laApiDeLaETSIIIDaLache)

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

El proceso de consulta es el siguiente:

1. Identificarse en el servicio de login proporcionando credenciales de 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, ejecutamos el script desde el host:

pruebaEntorno.sh

```

connectionCount":3}}
mongodb_1 | {"t":{"$date":"2020-12-22T17:49:29.917+00:00"},"s":"I", "c":"NETWORK", "id":22944
, "ctx":"conn33","msg":"Connection ended","attr":{"remote":"172.18.0.4:33774","connectionId":33,"c
onnectionCount":2}}
mongodb_1 | {"t":{"$date":"2020-12-22T17:49:29.917+00:00"},"s":"I", "c":"NETWORK", "id":22944
, "ctx":"conn34","msg":"Connection ended","attr":{"remote":"172.18.0.4:33776","connectionId":34,"c
onnectionCount":1}}
isep4jmeter_mongodbininit_1 exited with code 0
nodejs_1 | POST /api/v1/auth/login 200 51.088 ms - 184
nodejs_1 | POST /api/v1/auth/login 200 51.088 ms - 184
nodejs_1 | GET /api/v1/alumnos/alumno/mariweiss%40tropoli.com 200 30.568 ms - 1162
nodejs_1 | GET /api/v1/alumnos/alumno/mariweiss%40tropoli.com 200 30.568 ms - 1162

```

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 a la IP donde corre el contenedor de Docker. Si todo está correctamente configurado, obtendrá el perfil de un alumno.

```

josele@MSI:~$ chmod +x pruebaEntorno.sh
josele@MSI:~$ ./pruebaEntorno.sh
{"_id":"5fe231a76999a5c9ee0c60a3","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}josele@MSI:~$

```

<dockerfile>:

En este caso el docker se ha creado de un docker previo de node (FROM). Y ha hecho una carpeta dentro del docker que se llama app(RUN) y lo ha copiado al docker a la carpeta app (COPY). Lo ha expuesto el docker por el puerto 3000 (EXPOSE) ya que se entra por este puerto a este docker. Como directorio de trabajo ha puesto la carpeta que ha creado (WORKDIR). Actualiza npm(RUN), pone el entorno de node como producción(ENV) y lo arranca(CMD).

Esto es un dockerfile, como se construye un docker. Normalmente heredas de una imagen que ya existe en docker y ejecutas los comandos que tu quieres. Y se te queda como una especie de máquina virtual, un CONTENEDOR con esa configuración. Cuando tu haces el docker run lo que hace es arrancar esa configuración. Esto para node.

Para mongo igual. Hereda de mongo. Copia un script sh y le cambia los permisos y lo inicializa. En el script inicializa una base de datos de mongo.

```

FROM node:8
RUN mkdir -p /usr/src/app
COPY . /usr/src/app

EXPOSE 3000

WORKDIR /usr/src/app
RUN build: ./nodejs
ENV ports: - "3000:3000"
CMD links: - mongodb

```

```

FROM mongo
COPY ./scripts/* /tmp/
RUN chmod 755 /tmp/initializeMongoDB.sh
WORKDIR /tmp
CMD ./initializeMongoDB.sh

```


Archivo *compose*:

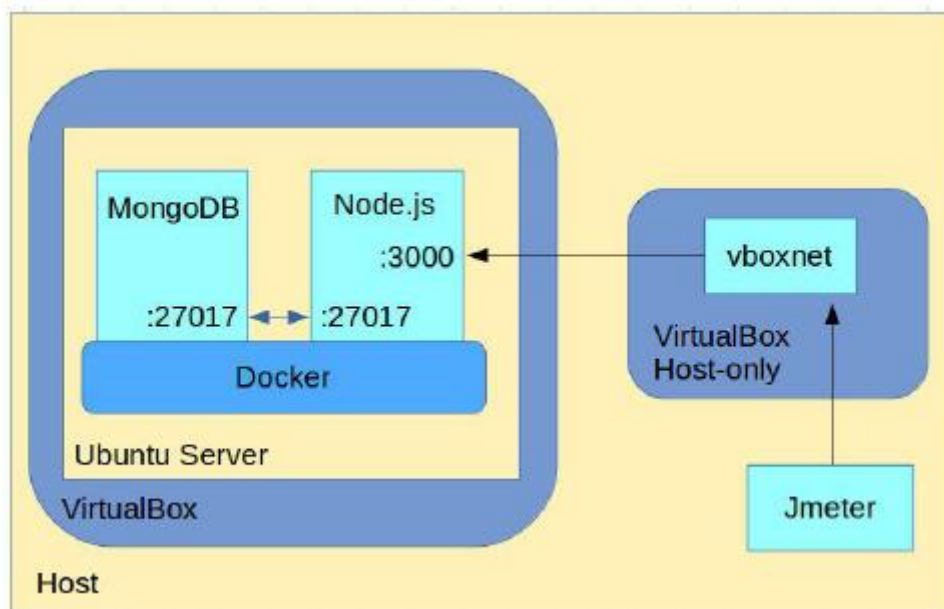
```
version: '2.0'
services:
  #MongoDB based in the original Mongo Image
  mongodb:
    image: mongo
    ports:
      - "27017:27017"

  # Initialize mongodb with data
  mongodinit:
    build: ./mongodb
    links:
      - mongodb

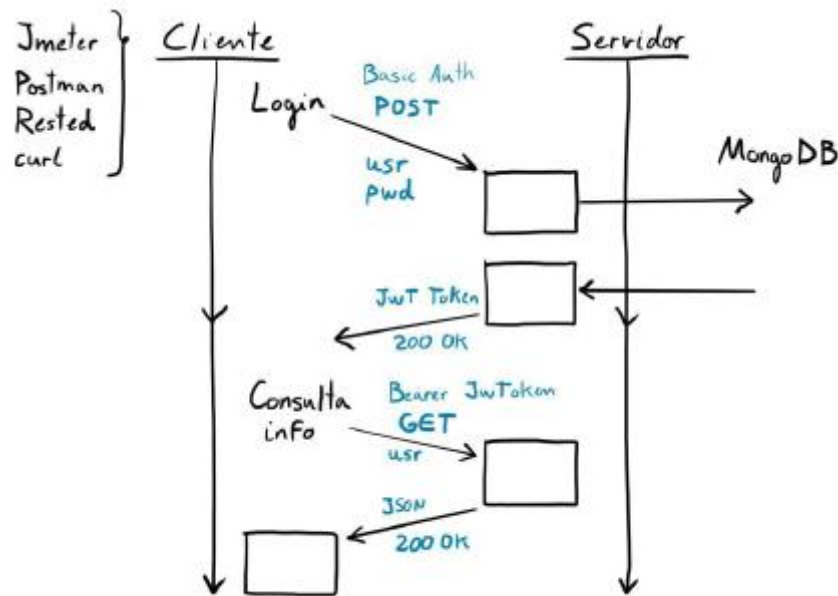
  # Nodejs App
  nodejs:
```

Para juntarlos lo que se hace es el docker compose. Una vez que están abiertos le dices que puertos quieres y cómo se inicializa.

Descripción de la aplicación:



Tenemos JMeter que manda las peticiones al servidor correspondiente (que es Ubuntu server). Ubuntu Server tiene con el docker levantado la aplicación de node y mongodb. JMeter va a hacer las peticiones al puerto 3000 e internamente ya están configurados los docker para que por el puerto 27017 node se conecta a mongo y mongo le devuelva la información (esta conexión está definida internamente).



En este caso, el cliente sería JMeter.

Primero mandamos al servidor una petición POST para que nos devuelva el token. El servidor internamente comprueba el usuario en la base de datos mongo y devuelve al servidor si todo ha ido bien un 200 con el token (en Consulta Info Jmeter recibe el token). Una vez que lo tenemos hacemos una petición GET usando el token y ya nos devolvería el servidor la información del usuario.

En JMeter se hacen 2 peticiones, una con el token y otra con la información y se va a devolver dos respuestas, una con el token si el login ha ido bien y otra respuesta con la información.

Instalaremos y configuraremos JMeter en la máquina anfitrión, Windows en mi caso y retomaremos a continuación en este punto.

Instalación y configuración de JMeter

Apache JMeter es un benchmark desarrollado en Java. Nos da una forma mucho más profesional para testear un servidor web ya que permite ejecutar varios procesos a la vez, varias hebras a la vez, simular situaciones reales a las que se puede ver sometido un servidor web en cualquier momento de su funcionamiento. Permite llevar a cabo test de carga muy reales, permitiendo crear concurrencia real en el sistema porque permite ejecutar varias hebras dentro de una cpu así como distribuir la creación de carga en varias máquinas.

Jmeter es una aplicación Java por lo que requiere disponer de una máquina virtual java instalada (JVM). La forma más común de definir los test, y que emplearemos, es con la consola gráfica. Para evitar tener que instalar un servidor de Xwindows en las MVs, vamos a ejecutar JMeter en el ordenador anfitrión. Los pasos básicos son:

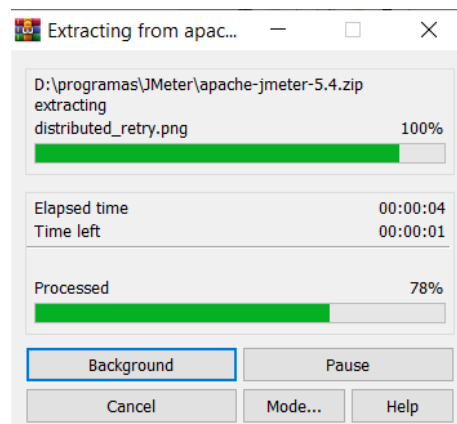
- 1º) Descargar e instalar máquina virtual java 11: <https://www.java.com/es/download/>



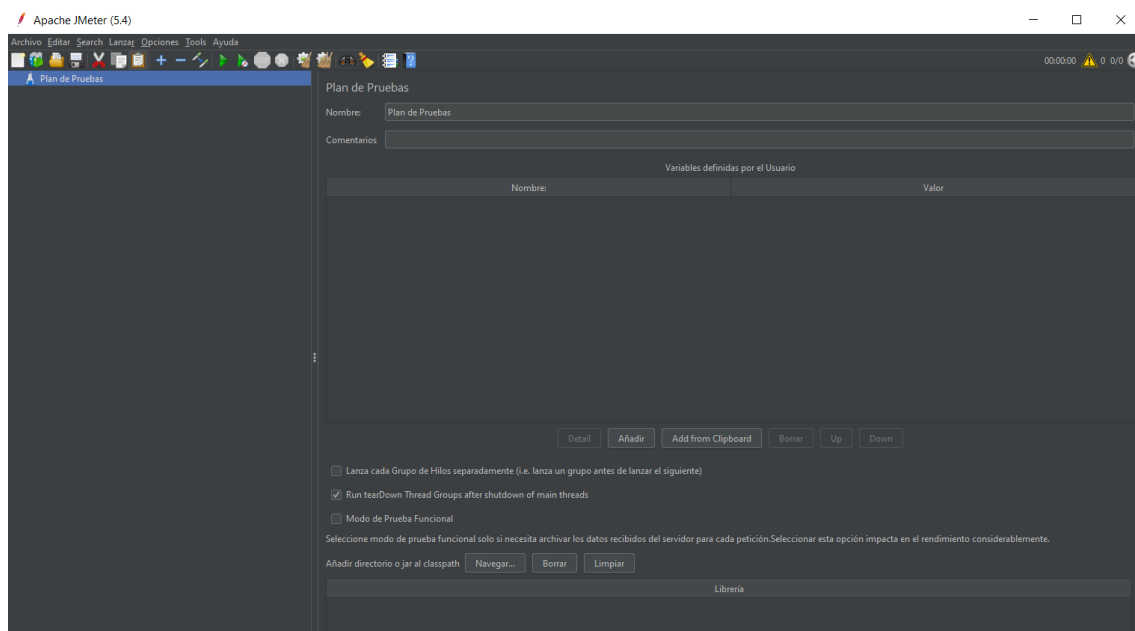
2º) Descargar JMeter desde un mirror cercano:

https://jmeter.apache.org/download_jmeter.cgi

Para instalarlo en Windows, simplemente descargamos comprobando la versión correcta y el .zip lo descomprimos en el directorio donde queramos instalarlo:



Desde el explorador de archivos (o el *cmd*) de Windows nos movemos al directorio `<jmeterlocation>\bin` y ejecutamos *jmeter.bat* o *jmeter.bat*.



3º) Seguir el tutorial para realizar la prueba de carga:

<https://jmeter.apache.org/usermanual/build-web-test-plan.html>

Configuración y prueba de carga contra iseP4JMeter

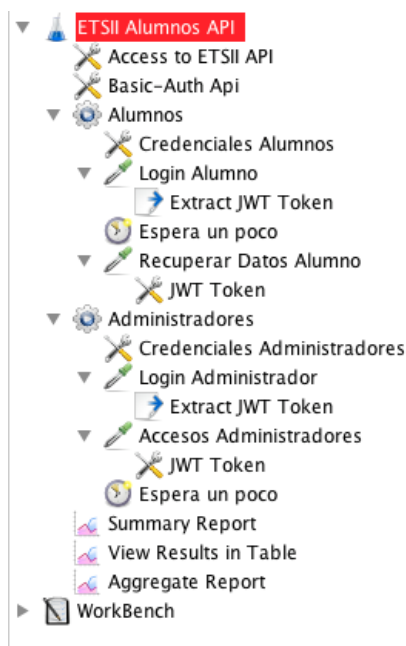
El subdirectorio JMeter contiene los archivos necesarios para realizar la sesión de prácticas:

- 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:

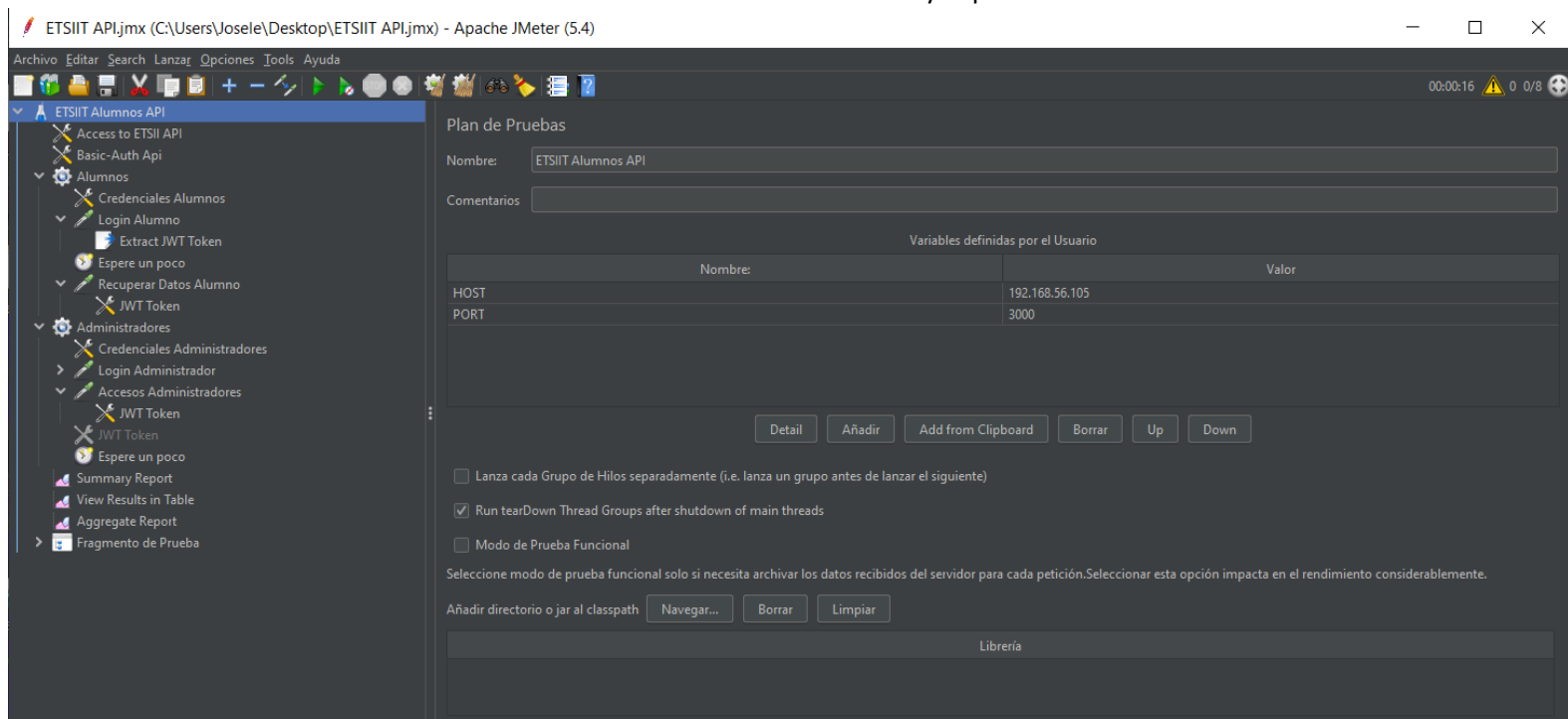
- Parametrizar el "EndPoint" del servicio mediante variables para la dirección y puerto del servidor. Emplee "User Defined Variables" del Test Plan.
- Definir globalmente las propiedades de los accesos Http y la Autenticacion Basic. Emplee HTTP Request Defatuls y HTTP Authorization Manager.
- Los accesos 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:

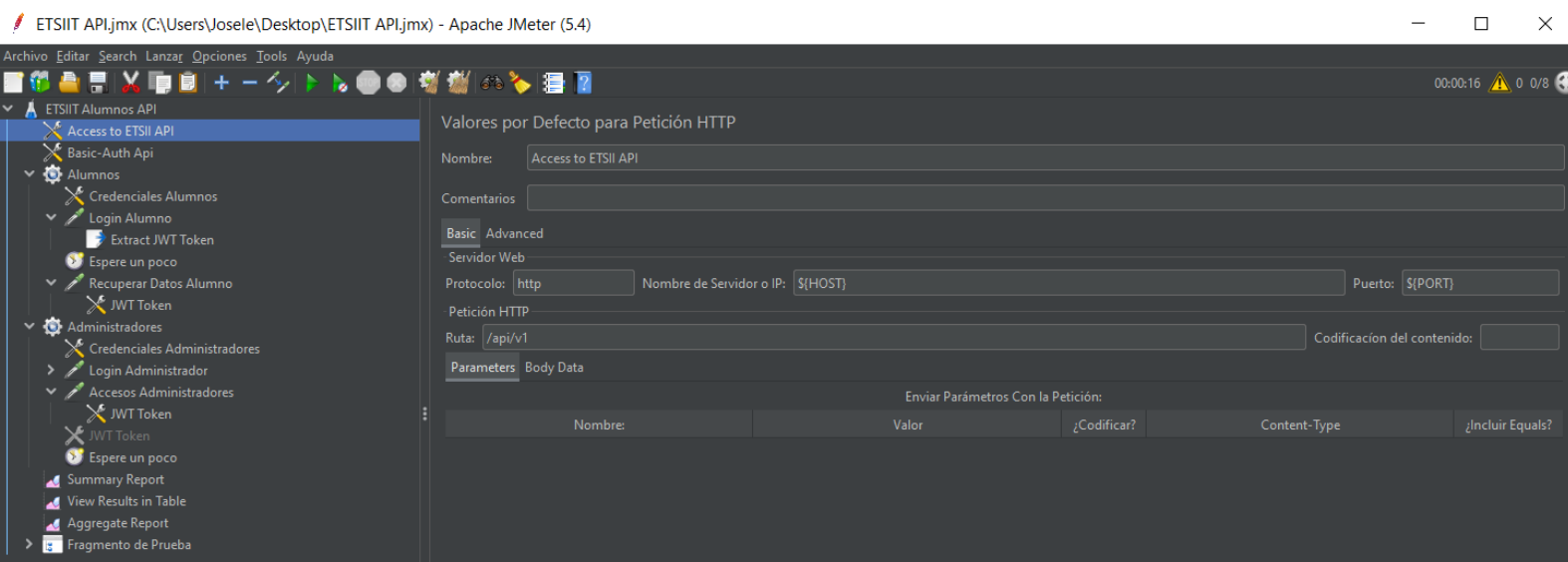


Procedemos siguiendo los pasos para la configuración de JMeter y lo requerido por la imagen anterior.

- Creamos el nuevo plan de pruebas llamado ETSIIT Alumnos API y añadimos dos variables nuevas: el host con la IP de UbuntuSever y el puerto 3000.



- Insertamos los valores por defecto para las peticiones HTTP que haremos posteriormente en:
Editar/Añadir/Elemento de configuración/Valores por defecto para petición HTTP. Y establecemos los valores como sigue:

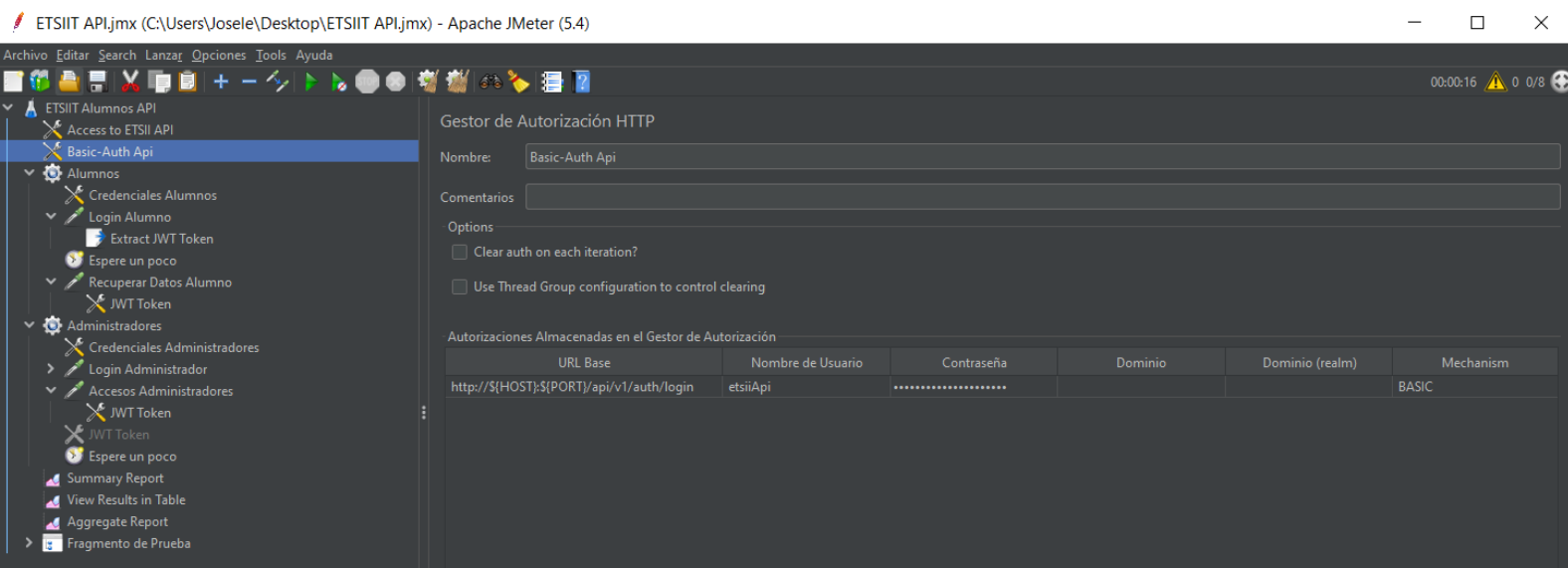


- Añadimos la autorización de la API en *ETSIIT_API/Editar/Añadir/Elemento de configuracion/Gestor de autorización HTTP*
Rellenamos con los siguientes datos, como hemos visto anteriormente en la imagen del navegador:

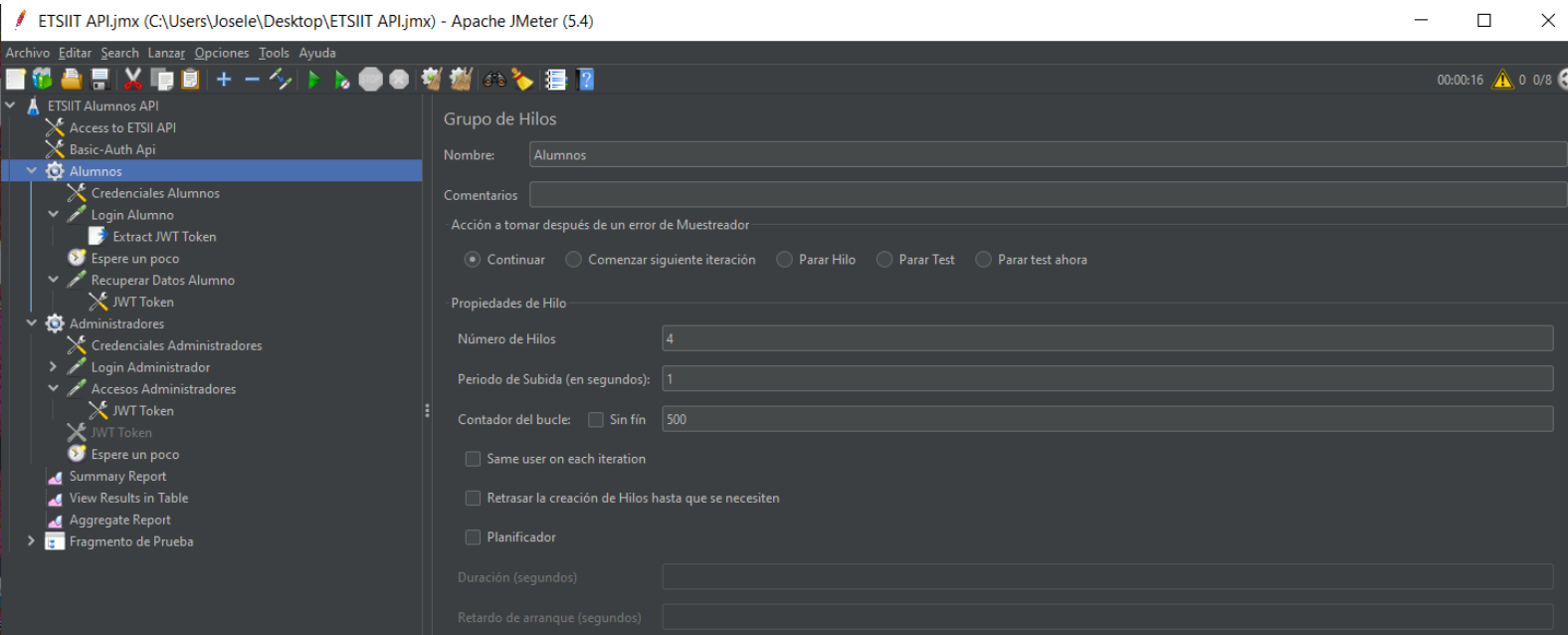
URL Base: [https://\\$\(HOST\):\\$\(PORT\)/api/v1/auth/login](https://$(HOST):$(PORT)/api/v1/auth/login)

Nombre: etsiapi

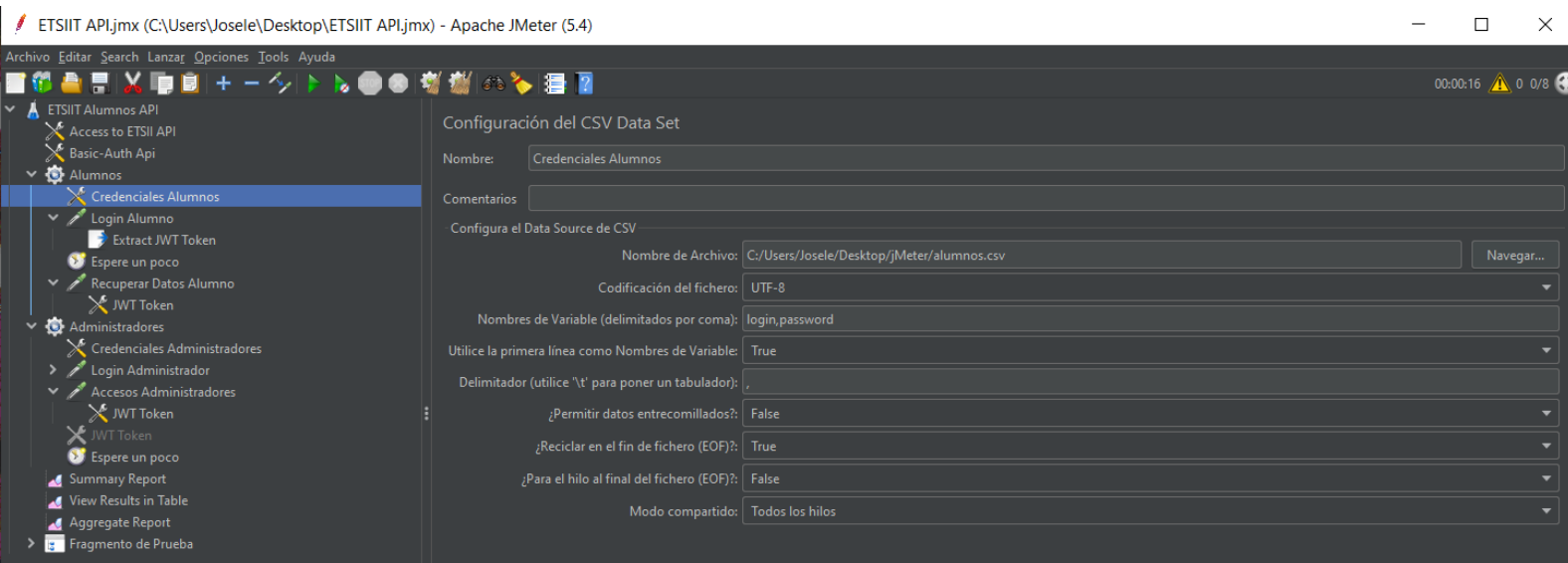
Contraseña: laApiDeLaEtsiiDaLache



- **Alumnos**
- Añadimos la hebra (grupo de usuarios) de los alumnos como sigue:
ETSIIT Alumnos API/Editar/Añadir/Hilos(Usuarios)/Grupo de hilos

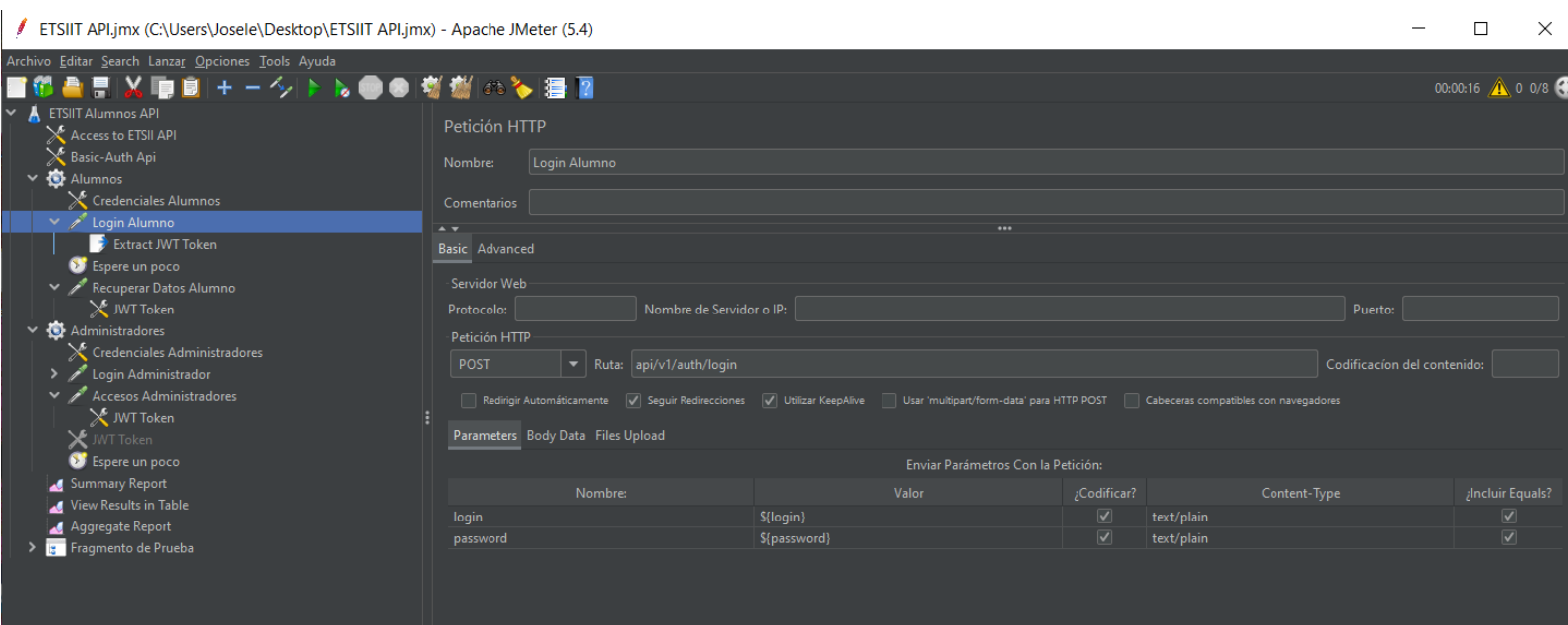


- Configuramos las credenciales de los alumnos.
Alumnos/Editar/Añadir/Elemento de configuración/Configuración del CSV Data Set



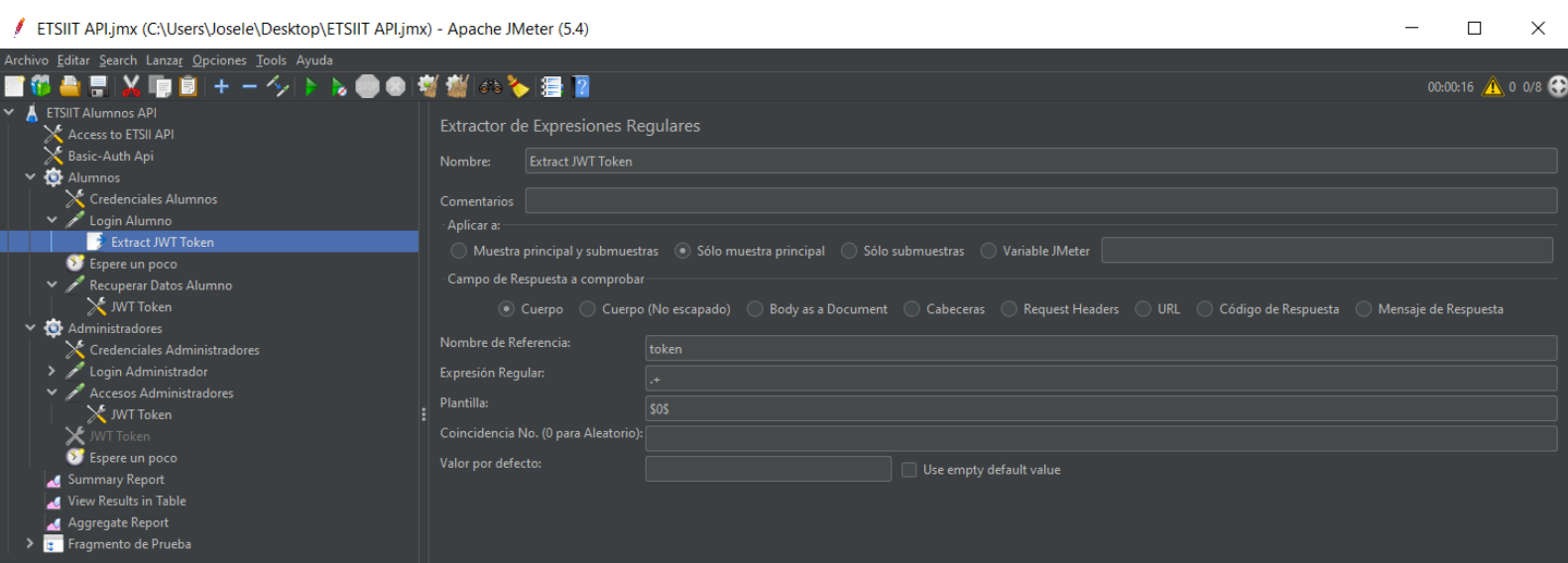
- Para poder logearnos, necesitamos el token del servidor, por lo que realizaremos una petición HTTP con el usuario y la contraseña. Una vez el servidor compruebe que los datos son correctos, nos devuelve el token.

Alumnos/Editar/Añadir/Muestreador/Peticion HTTP



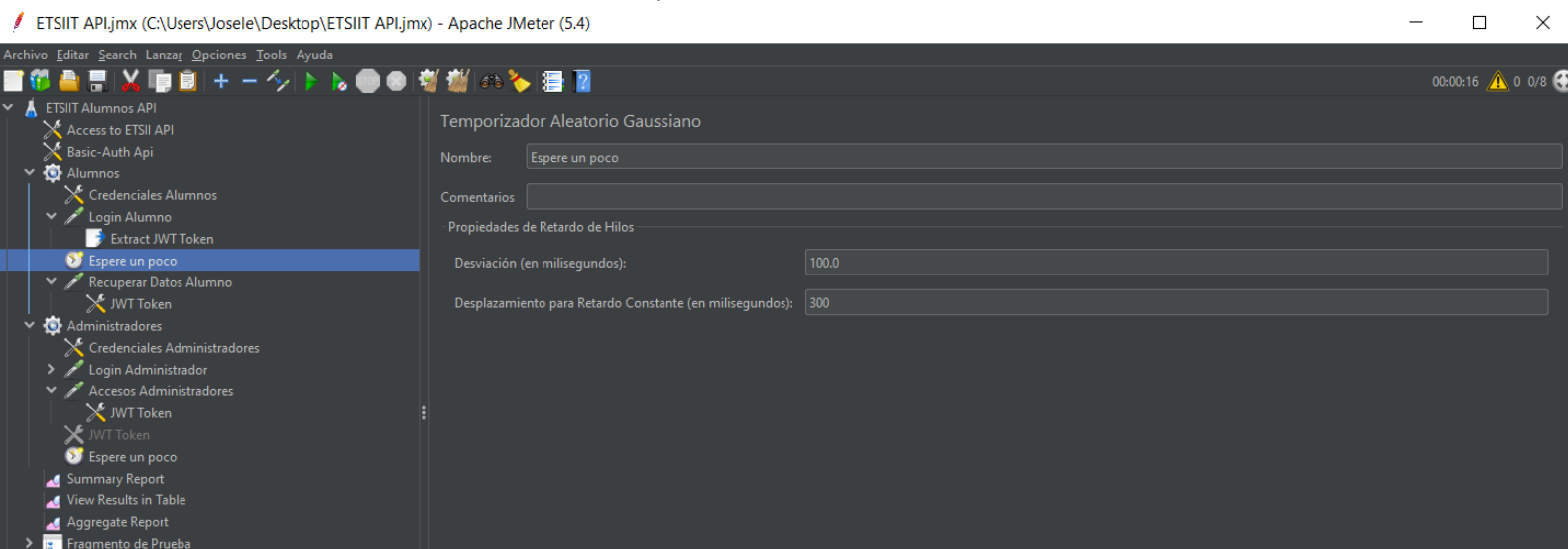
- Cuando realizamos la petición, necesitamos obtener el token que el servidor nos envía al comprobar los datos.

Alumnos/Login Alumno/Añadir/Post Procesadores/Extractor de expresiones regulares



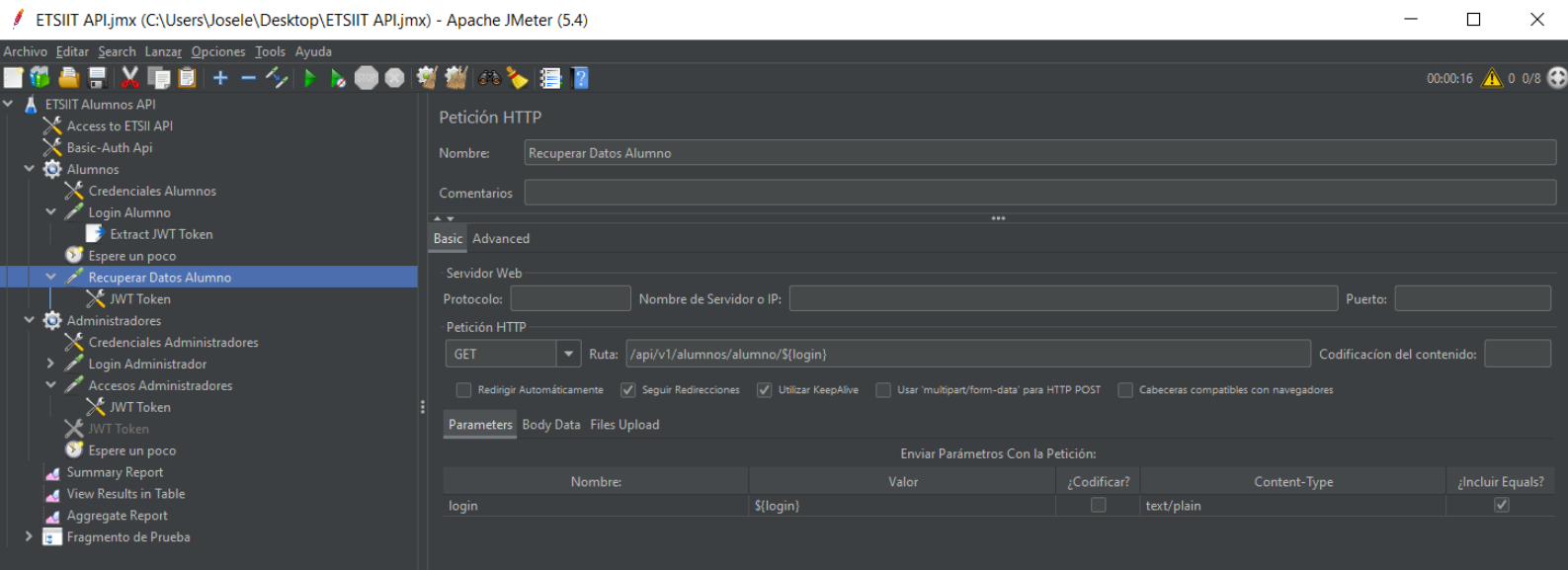
- Para hacer la simulación de los accesos de los alumnos más realista, creamos un temporizador aleatorio gaussiano.

Alumnos/Editar/Añadir/Temporizador Aleatorio Gaussiano



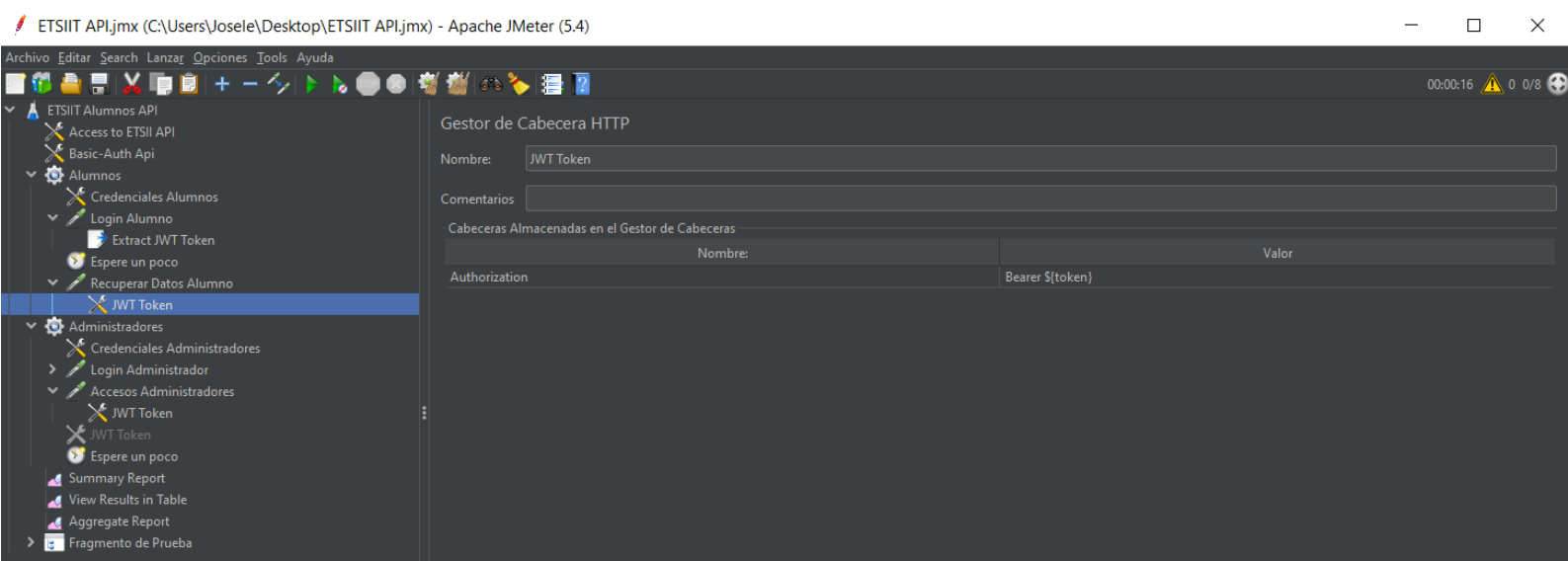
- Cuando tenemos el token para poder loguearnos, realizamos una petición al servidor con el usuario, la contraseña y el token que hemos obtenido anteriormente.

Alumnos/Editar/Añadir/Muestreador/Peticion HTTP

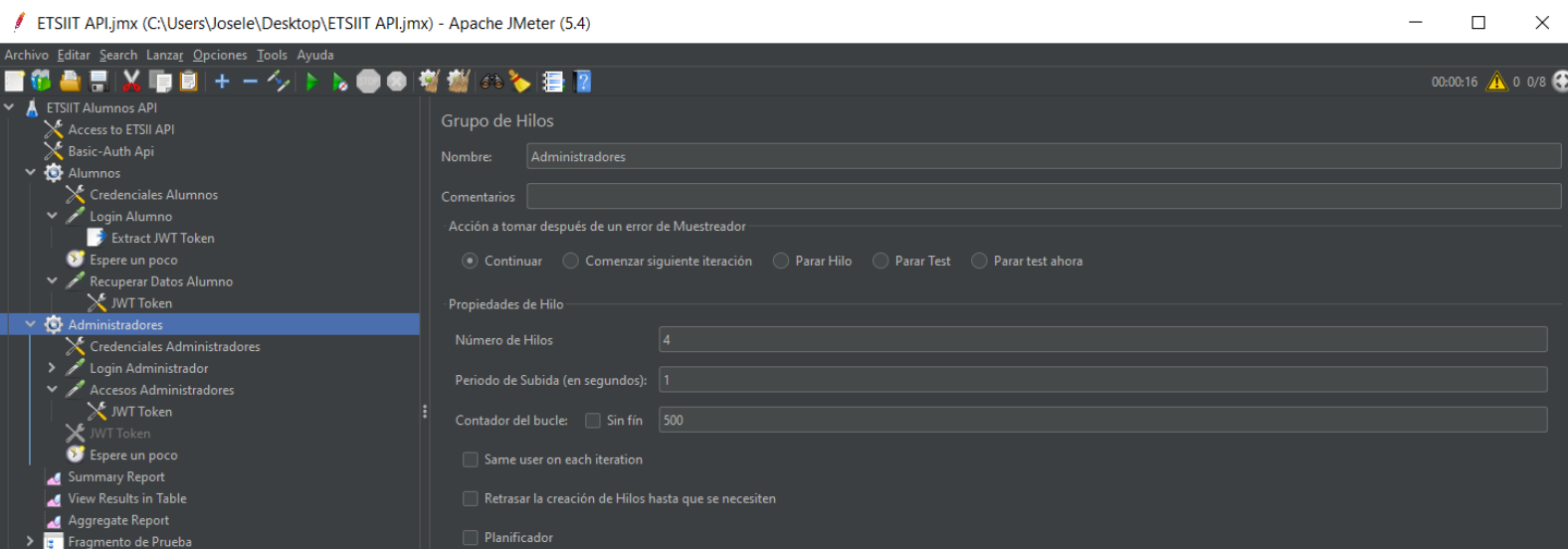


- Para resolver si al autenticarnos la aplicación nos redirige a otro nombre de dominio, entonces el token ya no sería útil, debemos mantener la sesión de la siguiente manera:

Alumnos/recuperar Datos Alumno/Editar/Añadir/Elemento de configuración/Gestor de Cabecera HTTP



- **Administradores**
- Añadimos la hebra de los administradores

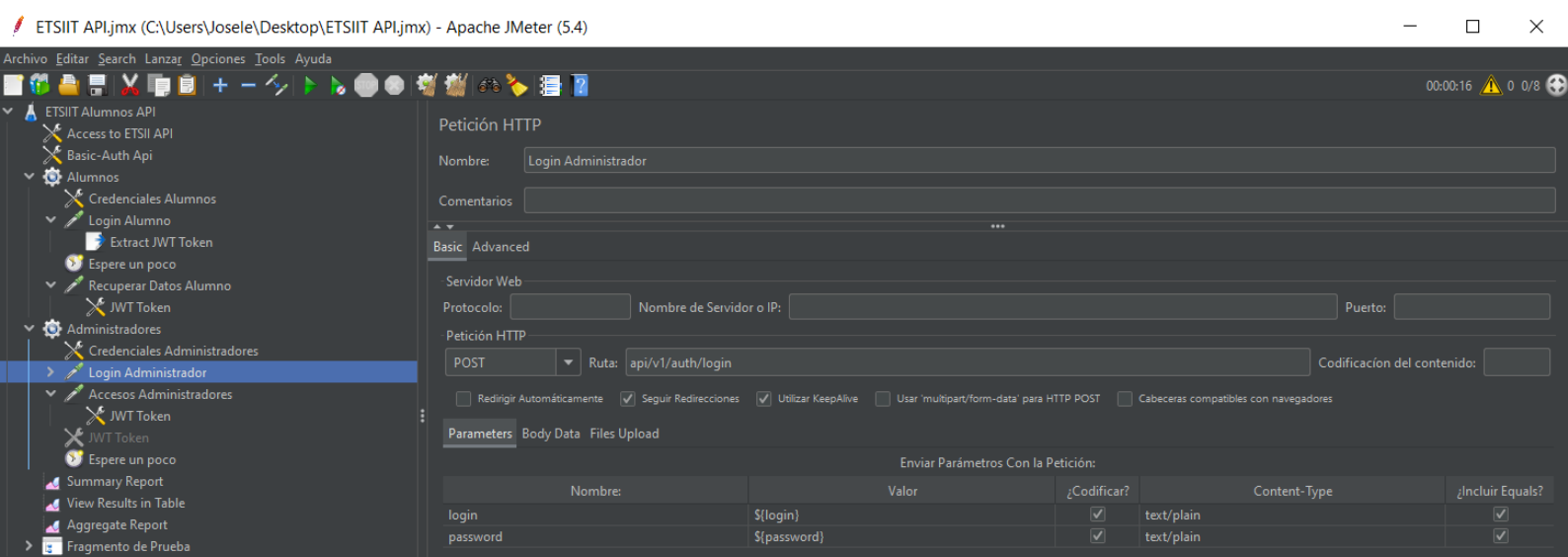


- Configuramos las credenciales de los administradores.
Administradores/Editar/Añadir/Elemento de configuración/Configuración del CSV Data Set



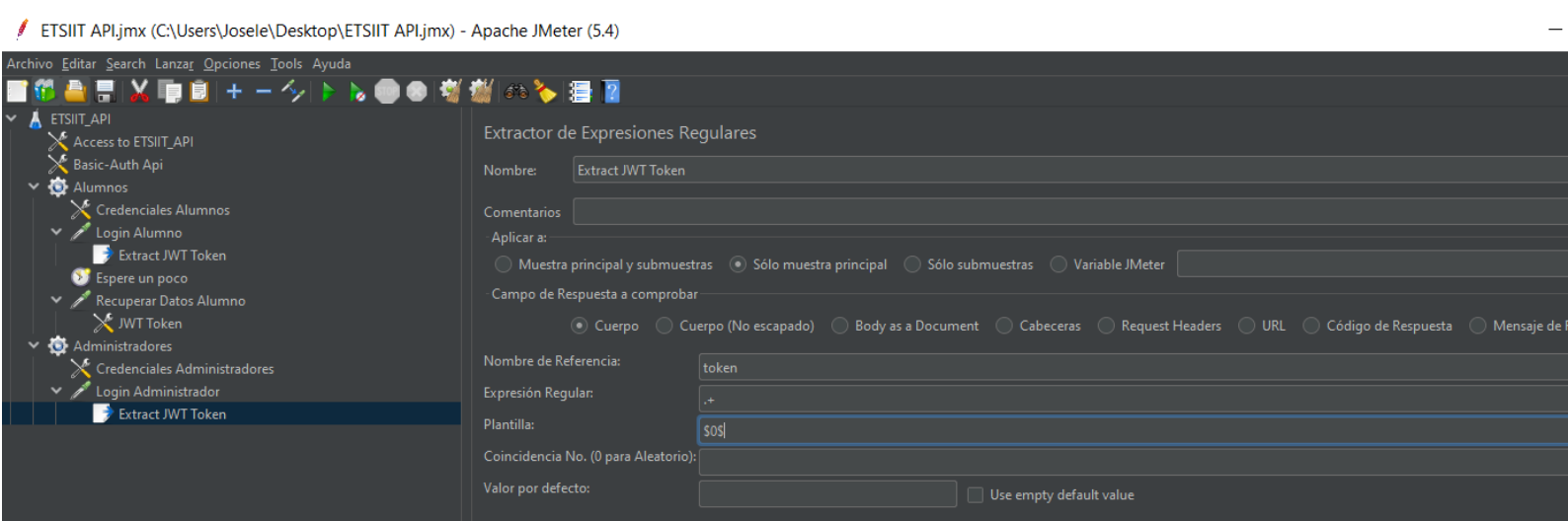
- Para poder logearnos, necesitamos el token del servidor, por lo que realizaremos una petición HTTP con el administrador y la contraseña. Una vez el servidor compruebe que los datos son correctos, nos devuelve el token.

Administrador/Editar/Añadir/Muestreador/Peticion HTTP



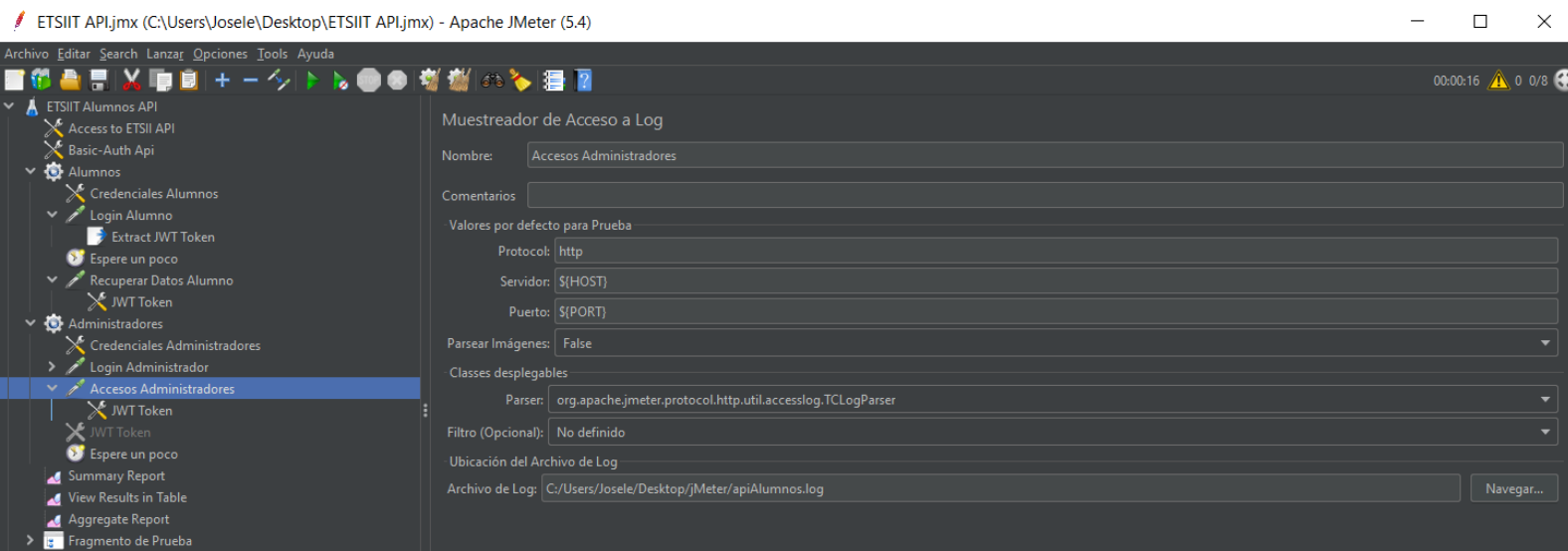
- Cuando realizamos la petición necesitamos obtener el token que el servidor nos envía al comprobar los datos.

Administradores/Login Administradores/Añadir/Post Procesadores/Extractor de expresiones regulares



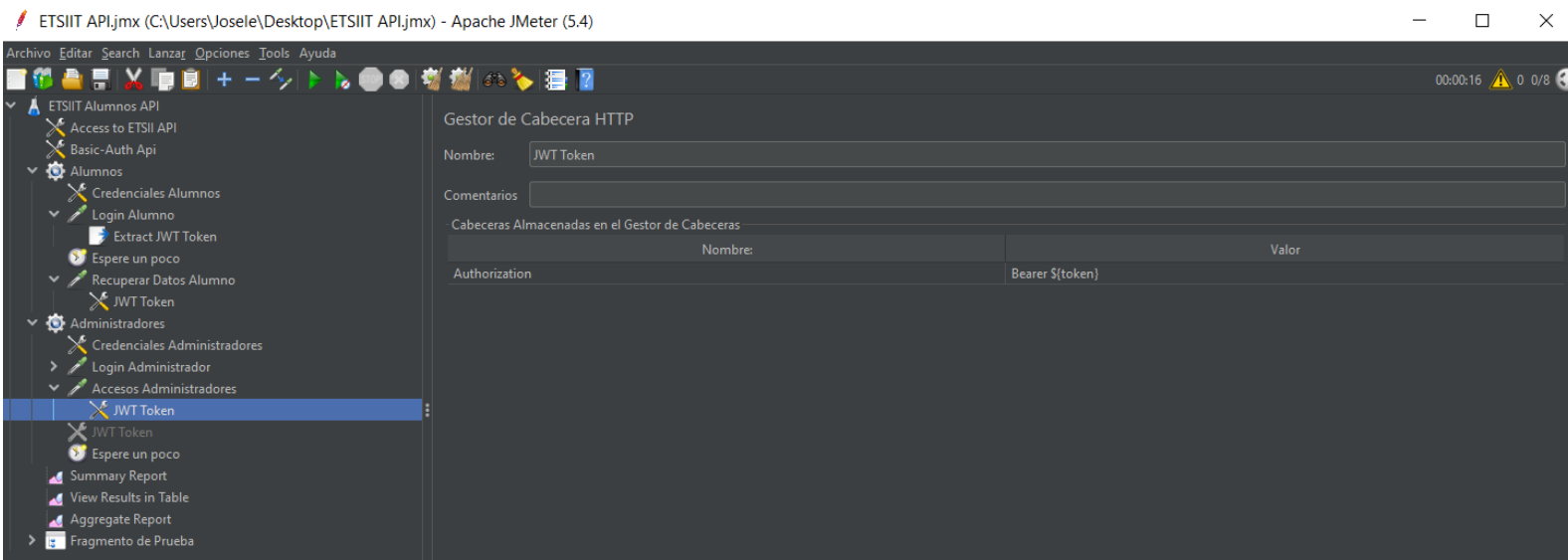
- Muestra el archivo .log de accesos al sistema.

Administradores/Editar/Añadir/Muestreador/Muestreador de Acceso a Log



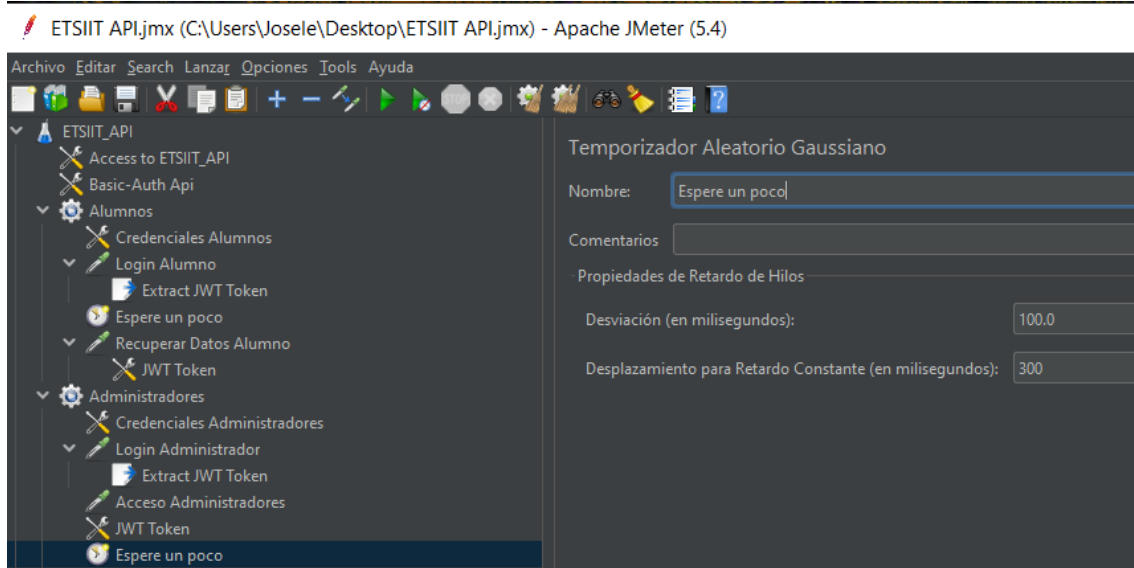
- Para resolver si autenticarnos la aplicación nos redirige a otro nombre de dominio, en lo que el token no sería útil.

Administradores/Editar/Añadir/Elemento de configuración/Gestor de Cabecera HTTP



- Para que la simulación de administradores accediendo sea más realista, creamos un temporizador aleatorio gaussiano.

Administradores/Editar/Añadir/Temporizador/Temporizador Aleatorio Gaussiano

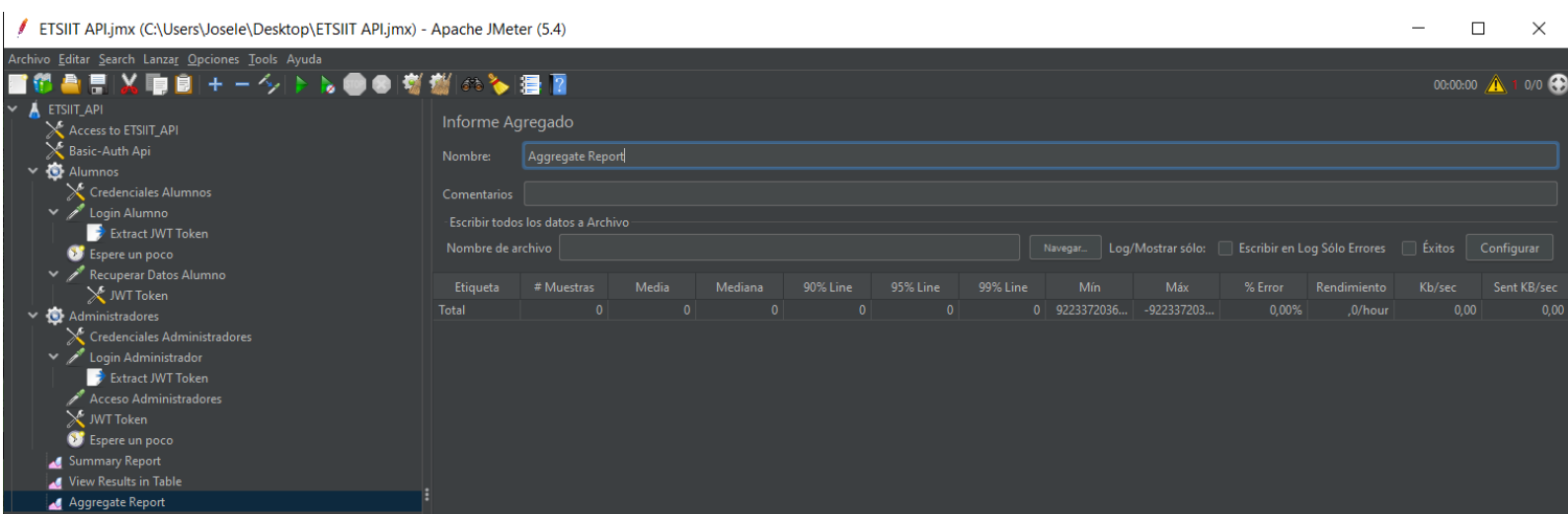


- **ETSIIT Alumnos API**
- Añadimos un reporte resumen, un informe agregado y la opción de ver los resultados en una tabla.


ETSIIT Alumnos API/Editar/Añadir/Receptor/Reporte resumen

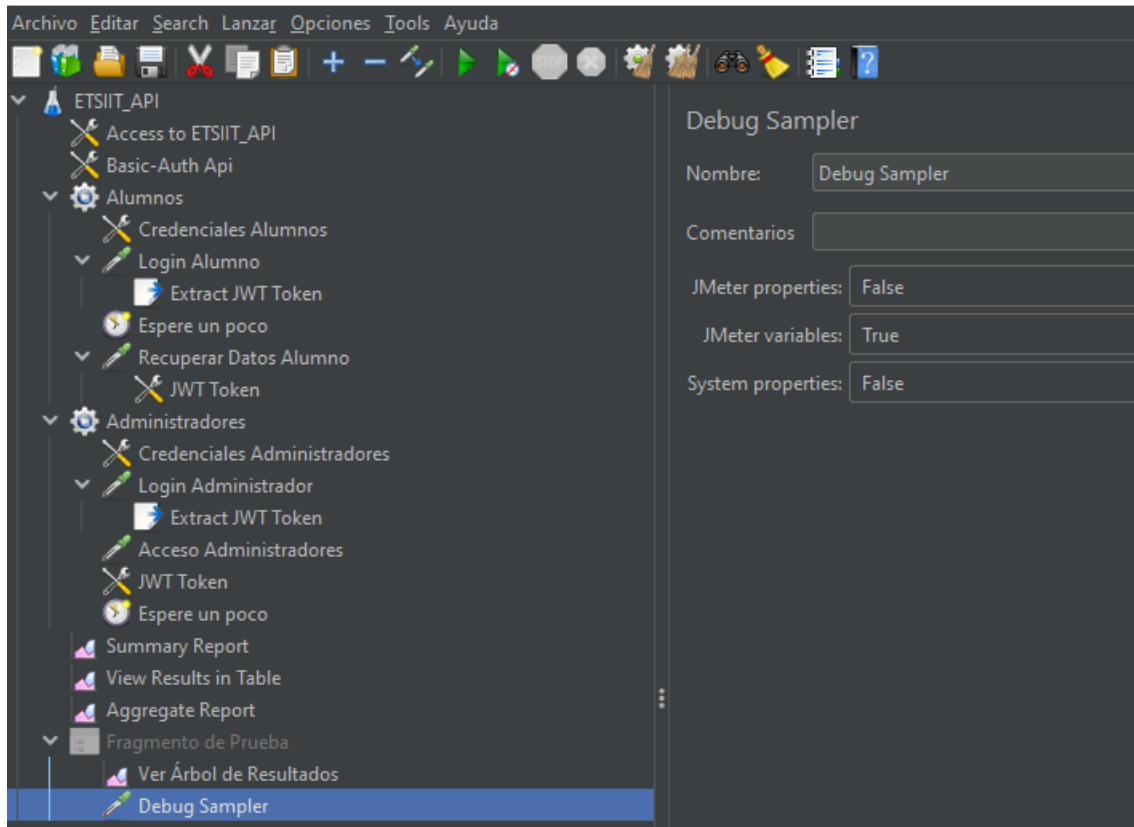
ETSIIT Alumnos API/Editar/Añadir/Receptor/Ver resultados en tabla

ETSIIT Alumnos API/Editar/Añadir/Receptor/Informe agregado



- Para realizar una prueba de la aplicación con estos parámetros:
*ETSIIT Alumnos API/Editar/Añadir/Fragmento de prueba/Fragmento de prueba
 Fragmento de prueba/Editar/Añadir/Receptor/Ver Árbol de Resultados
 Fragmento de prueba/Editar/Añadir/Muestreador/Debug Sampler*

 ETSIIT API.jmx (C:\Users\Josele\Desktop\ETSIIT API.jmx) - Apache JMeter (5.4)



- Ejecutar el programa
- En Ubuntu Server levantamos los dockers:


```
cd iseP4JMeter
docker-compose up
```
- Ejecutamos JMeter (en el play/run verde)
- Paramos después de un tiempo (un minuto, por ejemplo)
- Examinamos las tablas del Summary Report:

ETSII API.jmx (C:\Users\Josele\Desktop\ETSII API.jmx) - Apache JMeter (5.4)

Reporte resumen

Nombre: Summary Report

Comentarios:

Escribir todos los datos a Archivo

Nombre de archivo: Navegar... Log/Mostrar sólo: ☐ Escribir en Log Sólo Errores ☐ Éxitos ☐ Configurar

Etiqueta	# Muestras	Media	Min	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Byt...
Login Alumno	1127	6	4	36	4,14	0,00%	6,5/sec	3,70	2,14	583,8
Login Administrador	1134	6	4	46	4,38	0,00%	6,5/sec	3,80	2,16	595,0
Recuperar Datos Alumno	1126	5	2	52	4,11	0,00%	6,5/sec	8,98	2,60	1414,8
http://192.168.56.105:3000/api/v1/alumn...	4	5	4	9	2,06	0,00%	10,3/sec	11,43	0,00	1141,0
http://192.168.56.105:3000/api/v1/alumn...	4	11	4	19	5,49	0,00%	13,5/sec	15,92	0,00	1206,0
http://192.168.56.105:3000/api/v1/alumn...	4	4	4	6	0,83	0,00%	8,8/sec	13,27	0,00	1539,0
http://192.168.56.105:3000/api/v1/alumn...	4	4	4	5	0,43	0,00%	4,3/sec	8,27	0,00	1978,0
http://192.168.56.105:3000/api/v1/alumn...	4	12	4	26	8,34	0,00%	4,0/sec	4,08	0,00	1043,0
http://192.168.56.105:3000/api/v1/alumn...	4	7	4	14	4,09	0,00%	4,4/sec	6,65	0,00	1542,0
http://192.168.56.105:3000/api/v1/alumn...	4	5	4	10	2,49	0,00%	7,1/sec	7,85	0,00	1126,0
http://192.168.56.105:3000/api/v1/alumn...	4	7	4	14	4,09	0,00%	7,5/sec	6,96	0,00	950,0
http://192.168.56.105:3000/api/v1/alumn...	4	6	4	11	2,69	0,00%	5,2/sec	7,03	0,00	1379,0
http://192.168.56.105:3000/api/v1/alumn...	4	5	4	9	2,05	0,00%	7,0/sec	7,16	0,00	1050,0
http://192.168.56.105:3000/api/v1/alumn...	4	5	4	6	0,71	0,00%	6,0/sec	5,89	0,00	999,0
http://192.168.56.105:3000/api/v1/alumn...	4	6	4	13	3,70	0,00%	6,4/sec	9,84	0,00	1565,0
http://192.168.56.105:3000/api/v1/alumn...	4	5	4	7	1,22	0,00%	4,9/sec	5,36	0,00	1120,0
http://192.168.56.105:3000/api/v1/alumn...	4	12	4	30	10,62	0,00%	6,5/sec	13,37	0,00	2109,0
http://192.168.56.105:3000/api/v1/alumn...	4	6	4	7	1,22	0,00%	6,0/sec	6,69	0,00	1143,0
http://192.168.56.105:3000/api/v1/alumn...	4	4	4	5	0,43	0,00%	7,1/sec	10,30	0,00	1479,0
http://192.168.56.105:3000/api/v1/alumn...	4	5	4	7	1,30	0,00%	5,9/sec	5,57	0,00	964,0
http://192.168.56.105:3000/api/v1/alumn...	4	10	3	26	8,98	0,00%	6,5/sec	8,12	0,00	1275,0
http://192.168.56.105:3000/api/v1/alumn...	4	4	4	5	0,43	0,00%	5,0/sec	4,89	0,00	1005,0
http://192.168.56.105:3000/api/v1/alumn...	4	6	4	9	2,50	0,00%	5,3/sec	8,28	0,00	1592,0
http://192.168.56.105:3000/api/v1/alumn...	4	13	4	22	6,65	0,00%	4,4/sec	6,31	0,00	1472,0
http://192.168.56.105:3000/api/v1/alumn...	4	4	4	5	0,43	0,00%	4,4/sec	9,49	0,00	2211,0
http://192.168.56.105:3000/api/v1/alumn...	4	4	4	5	0,43	0,00%	3,9/sec	6,07	0,00	1580,0
http://192.168.56.105:3000/api/v1/alumn...	4	4	4	5	0,50	0,00%	4,1/sec	6,97	0,00	1723,0

☐ ¿Incluir el nombre del grupo en la etiqueta? ☒ Guardar la cabecera de la tabla

Como podemos comprobar en la tabla, tenemos un error de 0% en cada petición.

ETSII APIjmx (C:\Users\Josele\Desktop\ETSII APIjmx) - Apache JMeter (5.4)

Archivo Editar Search Lanzar Opciones Tools Ayuda

Ver Árbol de Resultados

Nombre:

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo Navegar... Log/Mostrar sólo: ☐ Escribir en Log Sólo Errores ☐ Éxitos ☐ Configurar

Buscar: Sensible a m...

Texto

- ETSII Alumnos API
 - Access to ETSII API
 - Basic-Auth Api
 - Alumnos
 - Credenciales Alumnos
 - Login Alumno
 - Extract JWT Token
 - Espera un poco
 - Recuperar Datos Alumno
 - JWT Token
 - Administradores
 - Credenciales Administradores
 - Login Administrador
 - Accesos Administradores
 - JWT Token
 - Espera un poco
- Summary Report
- View Results in Table
- Aggregate Report
- Fragmento de Prueba
 - Ver Árbol de Resultados
 - Debug Sampler

Resultado

0 3.514 ms - 1190

nodejs_1 | GET /api/v1/alumnos/alumno/jeanettekent%40tropoli.com 200 3.208 ms - 1288

nodejs_1 | POST /api/v1/auth/login 200 3.142 ms - 185

nodejs_1 | POST /api/v1/auth/login 200 3.806 ms - 185

nodejs_1 | GET /api/v1/alumnos/alumno/lucasfuller%40tropoli.com 200 2.224 ms - 1015

nodejs_1 | POST /api/v1/auth/login 200 2.888 ms - 196

nodejs_1 | POST /api/v1/auth/login 200 5.007 ms - 185

nodejs_1 | POST /api/v1/auth/login 200 2.875 ms - 185

nodejs_1 | POST /api/v1/auth/login 200 2.735 ms - 196

nodejs_1 | GET /api/v1/alumnos/alumno/beardiyork%40tropoli.com 200 2.190 ms - 1391

nodejs_1 | GET /api/v1/alumnos/alumno/veronicapratt%40tropoli.com?login=veronicapratt%40tropoli.com 200 2.234 ms - 783

nodejs_1 | GET /api/v1/alumnos/alumno/cookelevy%40tropoli.com?login=cookelevy%40tropoli.com 200 2.391 ms - 462

nodejs_1 | GET /api/v1/alumnos/alumno/waltersmelendez%40tropoli.com?login=waltersmelendez%40tropoli.com 200 2.277 ms - 634

nodejs_1 | POST /api/v1/auth/login 200 3.446 ms - 196

nodejs_1 | POST /api/v1/auth/login 200 2.783 ms - 196

nodejs_1 | POST /api/v1/auth/login 200 3.117 ms - 185

nodejs_1 | GET /api/v1/alumnos/alumno/fernandezhobbs%40tropoli.com?login=fernandezhobbs%40tropoli.com 200 2.988 ms - 321

nodejs_1 | GET /api/v1/alumnos/alumno/cannonbooth%40tropoli.com 200 2.452 ms - 1243

nodejs_1 | POST /api/v1/auth/login 200 9.799 ms - 185

nodejs_1 | POST /api/v1/auth/login 200 4.261 ms - 196

nodejs_1 | GET /api/v1/alumnos/alumno/janicewilder%40tropoli.com 200 2.918 ms - 780

nodejs_1 | POST /api/v1/auth/login 200 6.658 ms - 185

nodejs_1 | POST /api/v1/auth/login 200 3.300 ms - 185

nodejs_1 | GET /api/v1/alumnos/alumno/herrenapruitt%40tropoli.com 200 2.022 ms - 817

nodejs_1 | GET /api/v1/alumnos/alumno/parksforbes%40tropoli.com?login=parksforbes%40tropoli.com 200 2.707 ms - 538

nodejs_1 | POST /api/v1/auth/login 200 3.000 ms - 196

nodejs_1 | GET /api/v1/alumnos/alumno/steelebenjamin%40tropoli.com?login=steelebenjamin%40tropoli.com 200 2.218 ms - 1551

nodejs_1 | GET /api/v1/alumnos/alumno/helenamurphy%40tropoli.com 200 2.166 ms - 865

nodejs_1 | POST /api/v1/auth/login 200 3.113 ms - 196

En bruto

Scroll automatically?

Todo OK!!