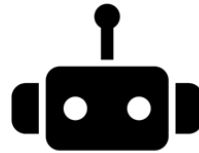




PUCP

SESIÓN DE LABORATORIO 2

Stress Test con Jmeter y Blaze Meter



HORARIO 10M1



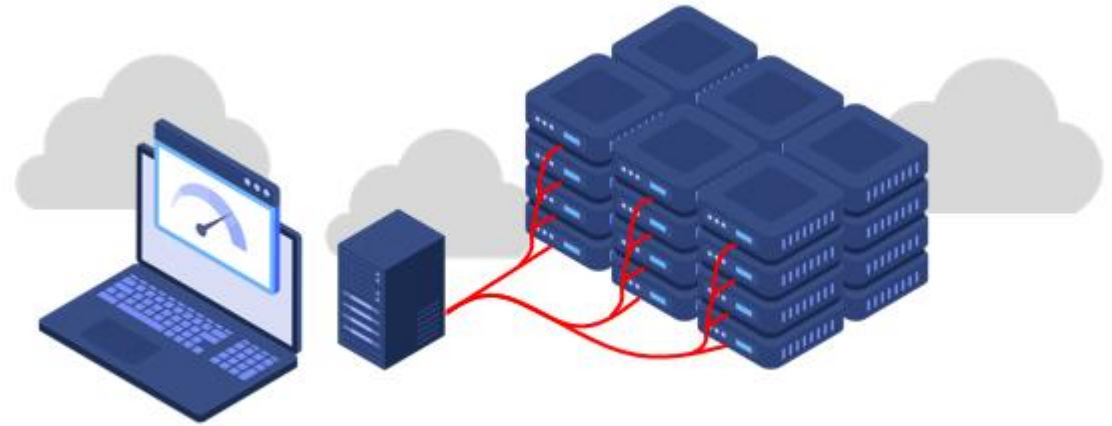
Stress Test

Es una técnica de evaluación del rendimiento que se centra en llevar al sistema más allá de sus límites normales de funcionamiento, simulando un escenario donde se presentan picos de carga inesperados, fallos de componentes, o situaciones críticas que impactan el rendimiento. En un entorno de producción, este tipo de situaciones puede ocurrir cuando, por ejemplo:

Un sitio web experimenta un aumento repentino en el tráfico debido a un evento, como una campaña de marketing o un lanzamiento de producto.

Un servidor de base de datos recibe muchas consultas simultáneas debido a un pico en la actividad del usuario.

Un servicio en la nube está al borde de su capacidad debido a fallas en otros sistemas.



Objetivos Principales

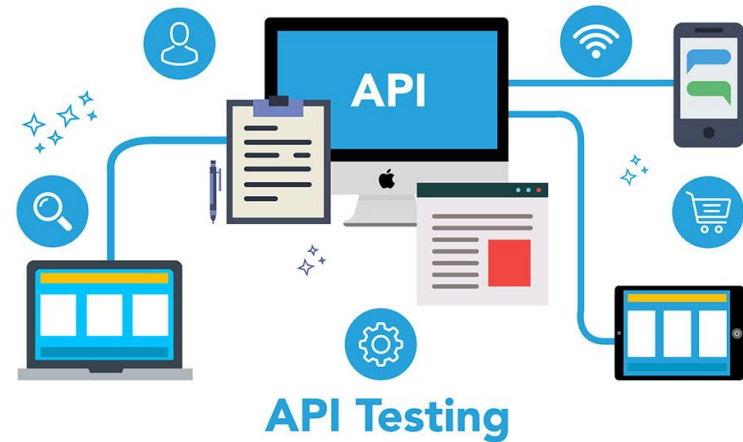
El propósito principal del *stress test* no es simplemente medir el rendimiento normal de la aplicación, sino descubrir ciertas características

Punto de quiebre: El *stress test* busca determinar en qué momento el sistema comienza a fallar, cuándo se degrada el rendimiento y cuántos usuarios o solicitudes puede manejar antes de volverse inestable o inoperativo.

Estabilidad bajo carga extrema: Evalúa si el sistema puede seguir respondiendo, aunque sea de forma lenta, en lugar de colapsar completamente bajo carga intensa.

Recuperación: Examina cómo se recupera el sistema después de un evento de sobrecarga. Algunos sistemas pueden colapsar completamente y necesitar un reinicio manual, mientras que otros pueden adaptarse y recuperar su funcionalidad lentamente.

Resiliencia: Analiza si el sistema puede volver a su estado óptimo después de la prueba de estrés, lo que es vital para su capacidad de mantenerse en operación de manera confiable.



Instalaciones



Los links de instalación se encuentran disponibles en el github.

La versión que utilizaremos será Jmeter para JAVA 8 en su versión zip

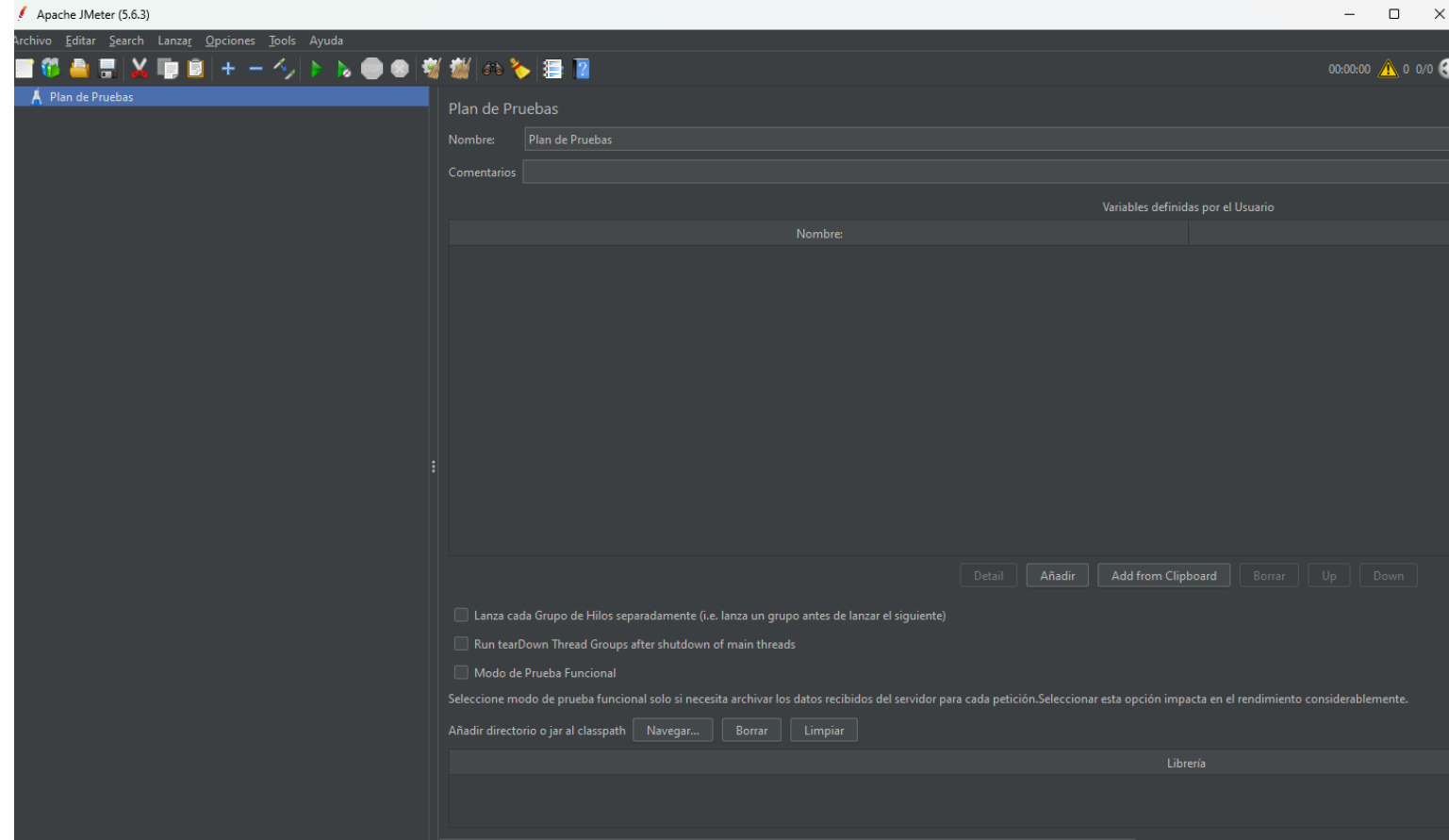
Apache JMeter 5.6.3 (Requires Java 8+)

Binaries

[apache-jmeter-5.6.3.tgz sha512 pgp](#)
[apache-jmeter-5.6.3.zip sha512 pgp](#)

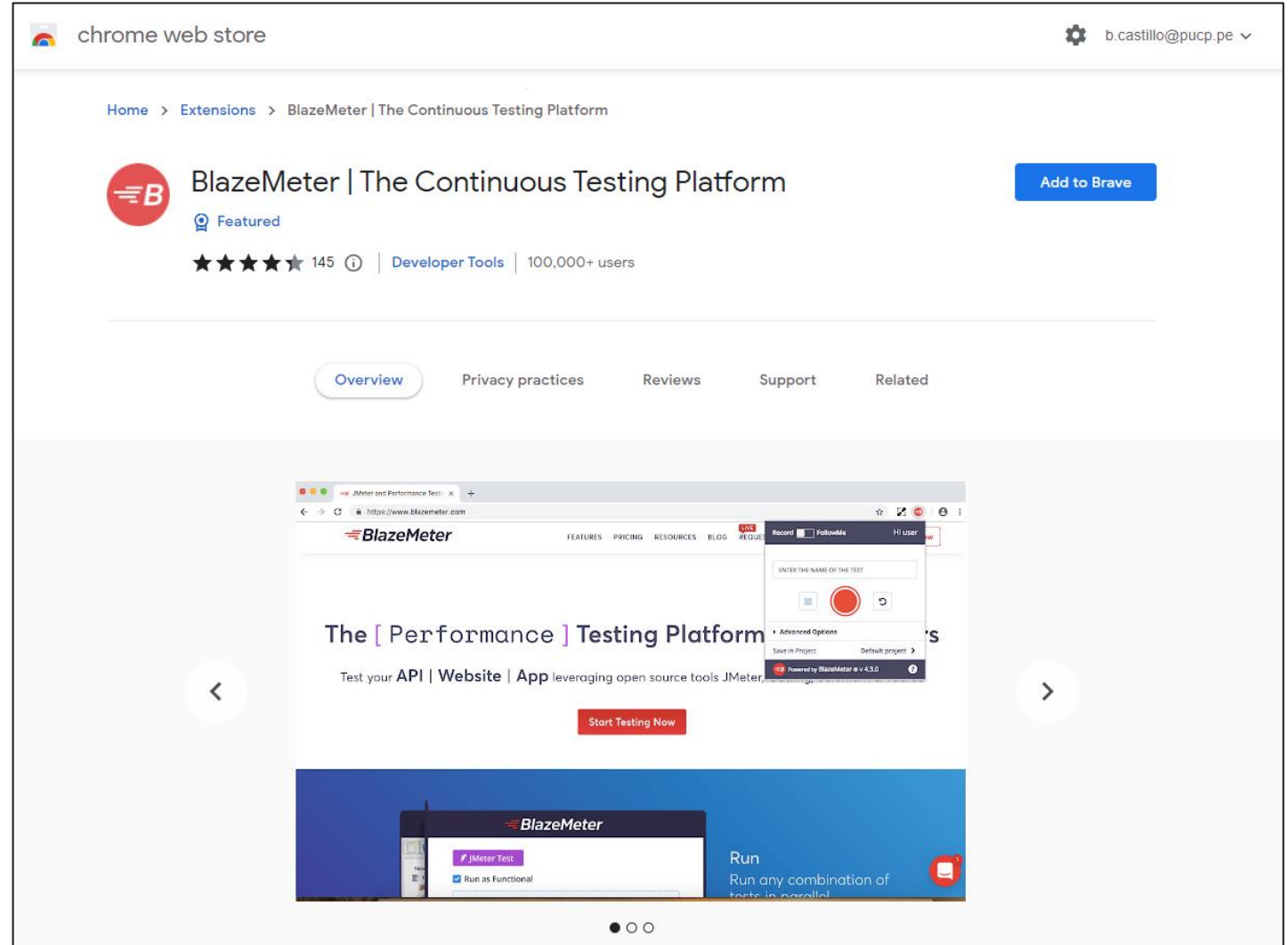
Instalaciones

Una vez tengamos lista la descarga procederemos a descomprimir el archivo en una carpeta donde buscaremos la ruta para bin y ejecutaremos el archivo ApacheJmeter



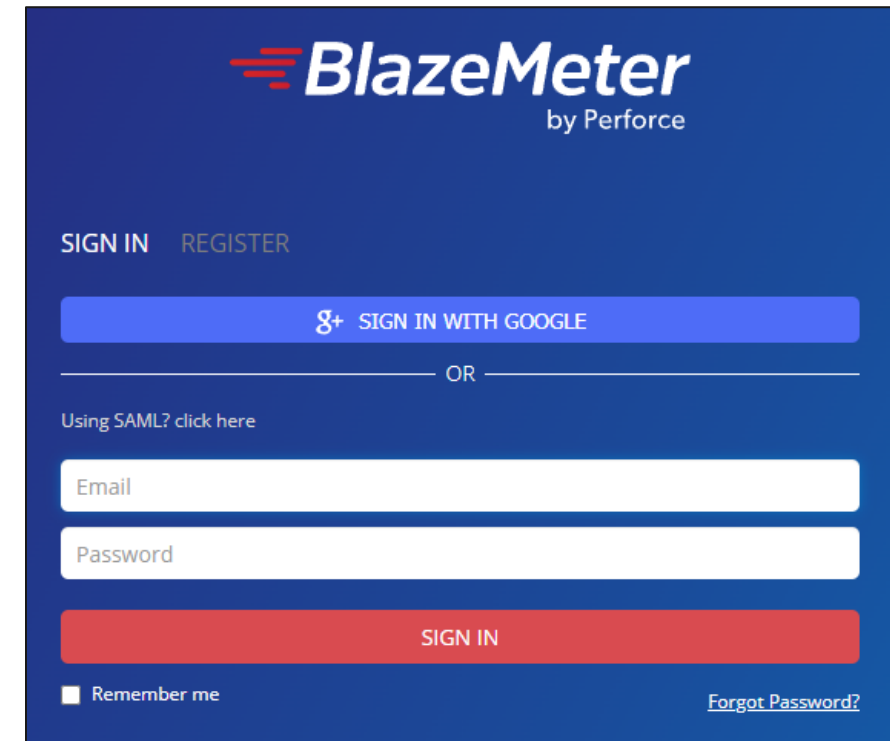
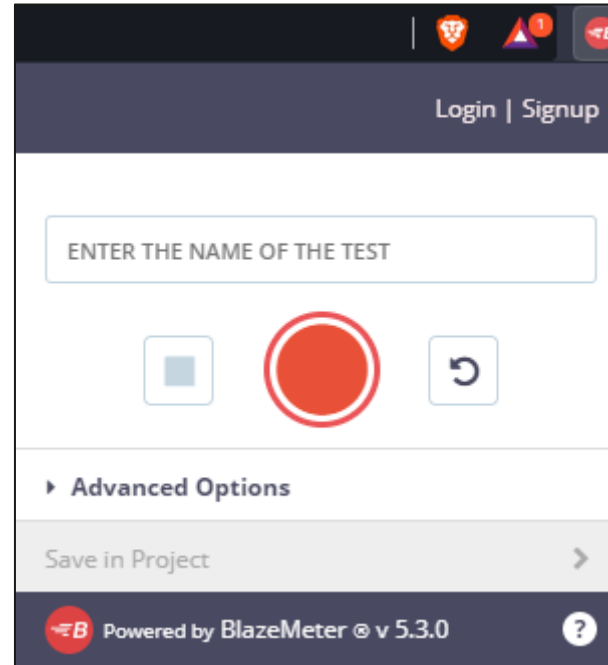
Instalaciones

Para poder usar blaze meter solo es necesario hacer la descarga directamente desde la Chrome store , de igual forma el link esta accesible desde el GitHub



Instalaciones

Una vez tengamos instalada la extensión solo debemos hacerle click , iniciar sesión y podremos empezar

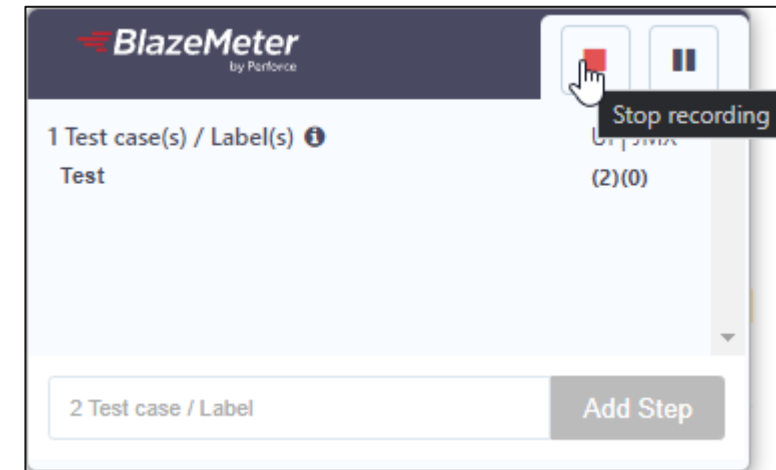
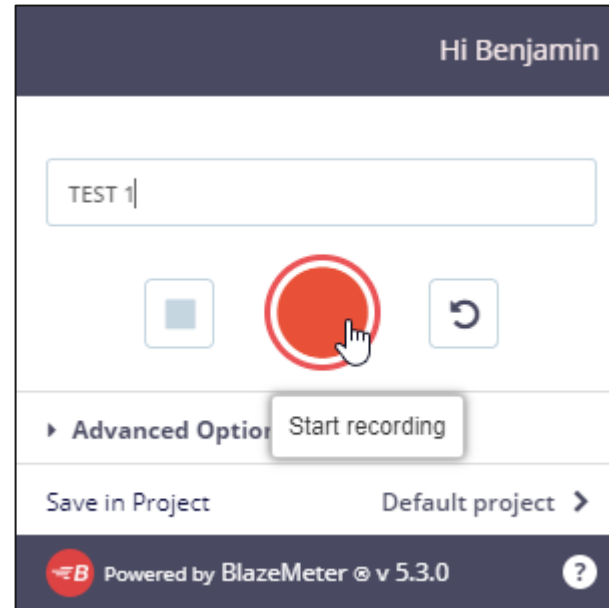


Pruebas

Para realizar una prueba en Blaze Meter debemos introducir el nombre de la prueba (PRUEBA 1)

Luego hacer clic en Iniciar grabación (botón rojo)

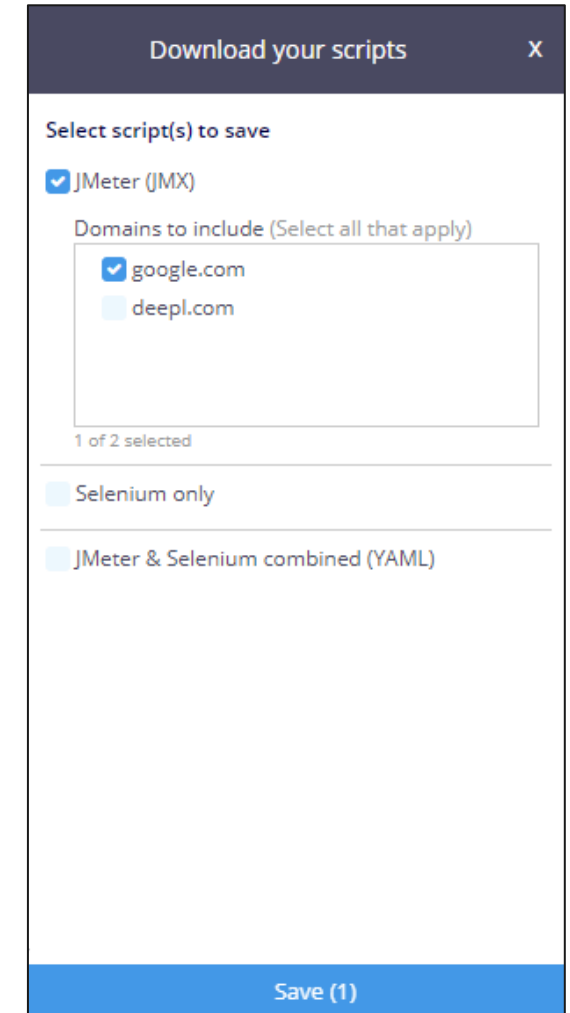
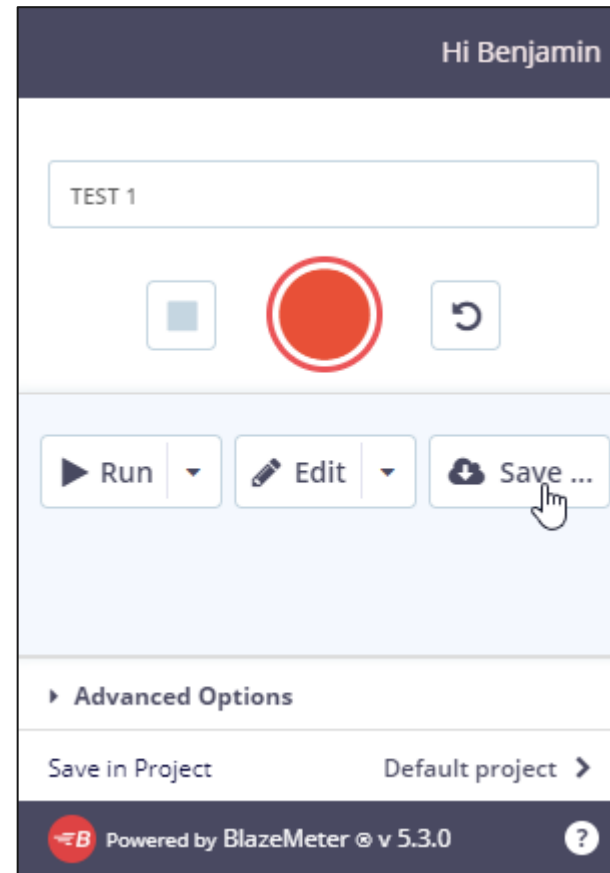
Luego cuando el caso de prueba haya terminado hacer clic en el botón Detener (botón rojo).



Pruebas

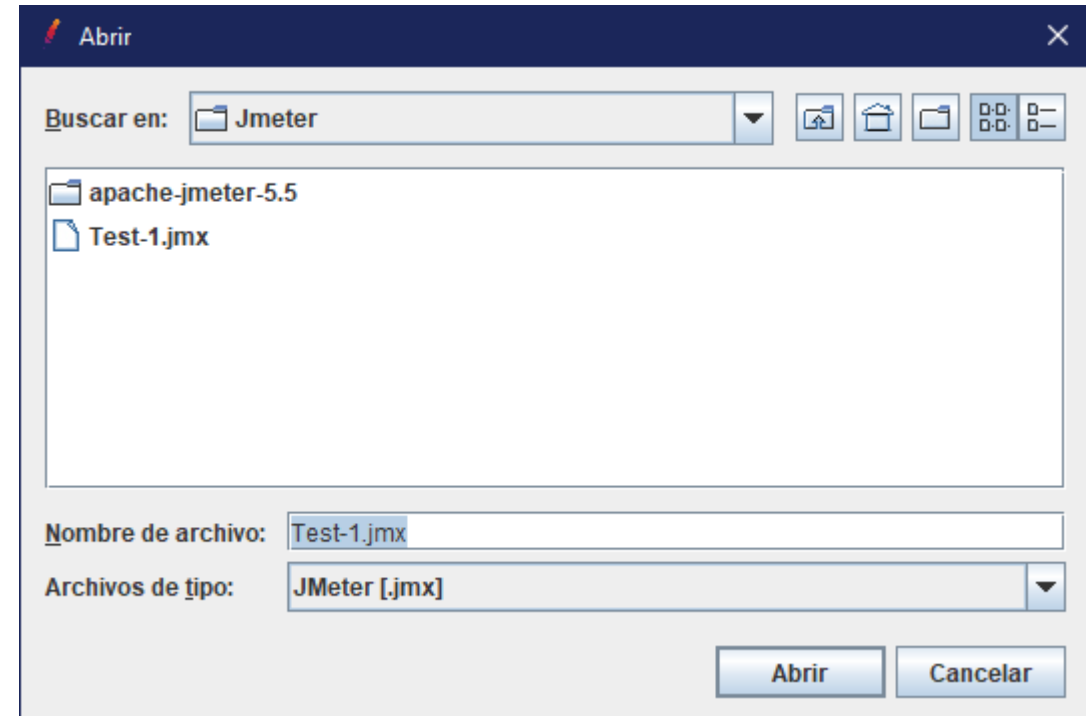
Después de procesar la grabación tenemos que hacer clic en Guardar, y guardar el archivo.

Luego elegimos descargar en JMeter y hacemos clic en guardar y esperamos a que se descargue y guardamos el archivo



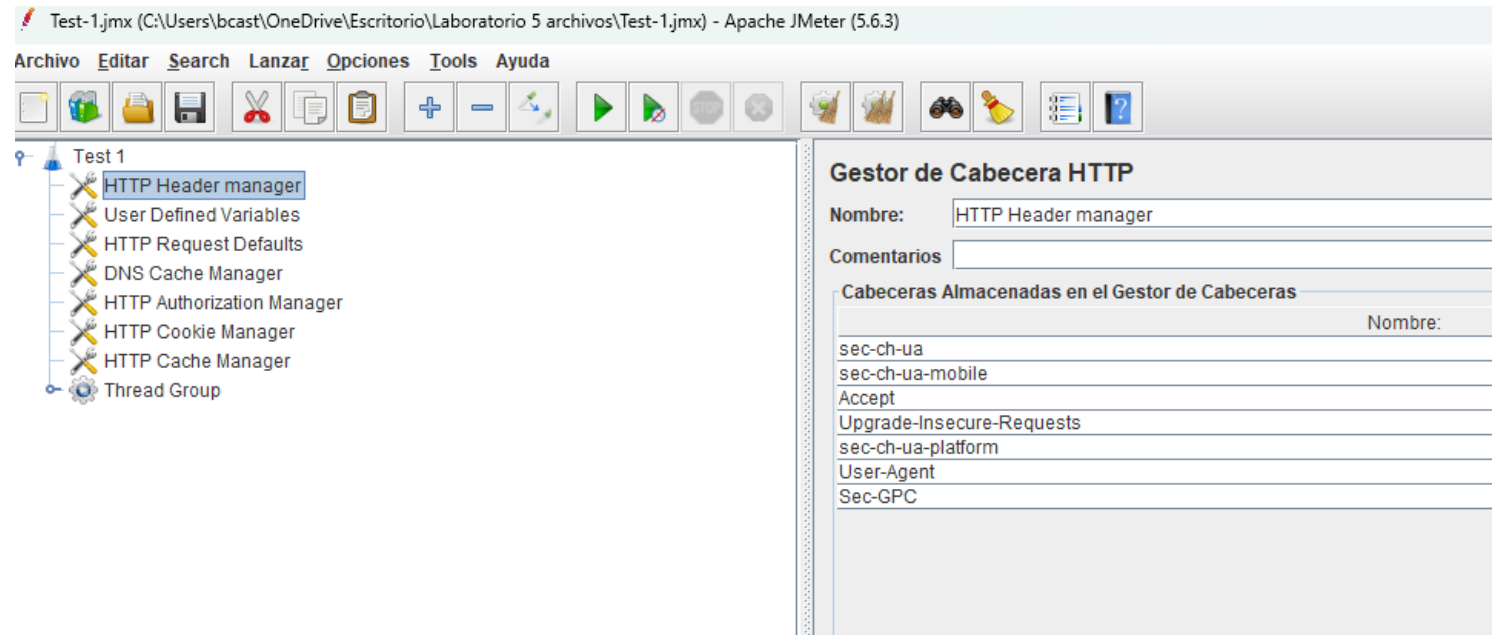
Pruebas

Ahora que tenemos el archivo, lo que haremos será ir a Jmeter e iremos a archivo/abrir y seleccionaremos el archivo que acabamos de descargar de Blaze meter



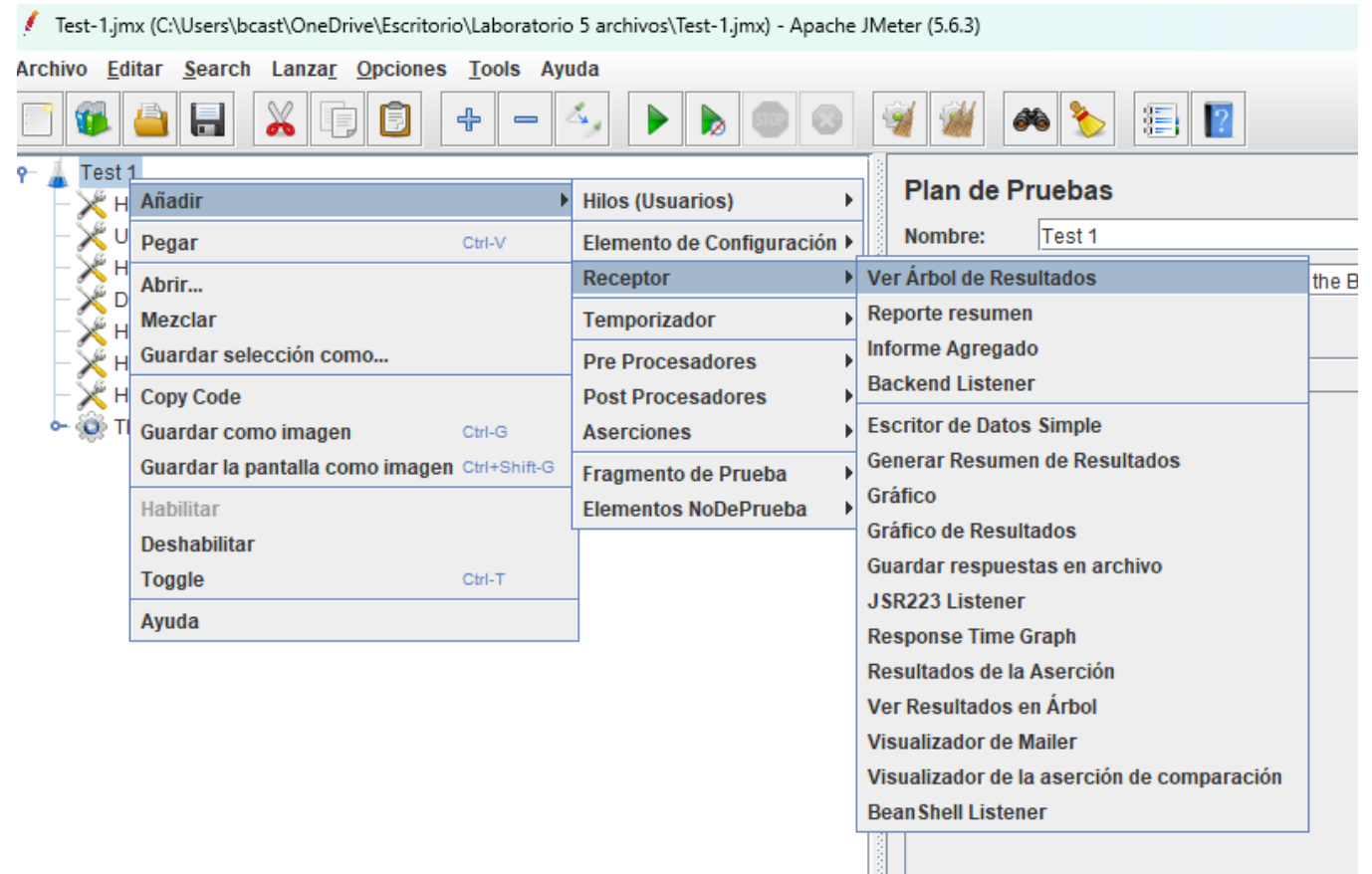
Pruebas

Una vez hecho esto , ya podremos visualizar el plan de pruebas de Jmeter



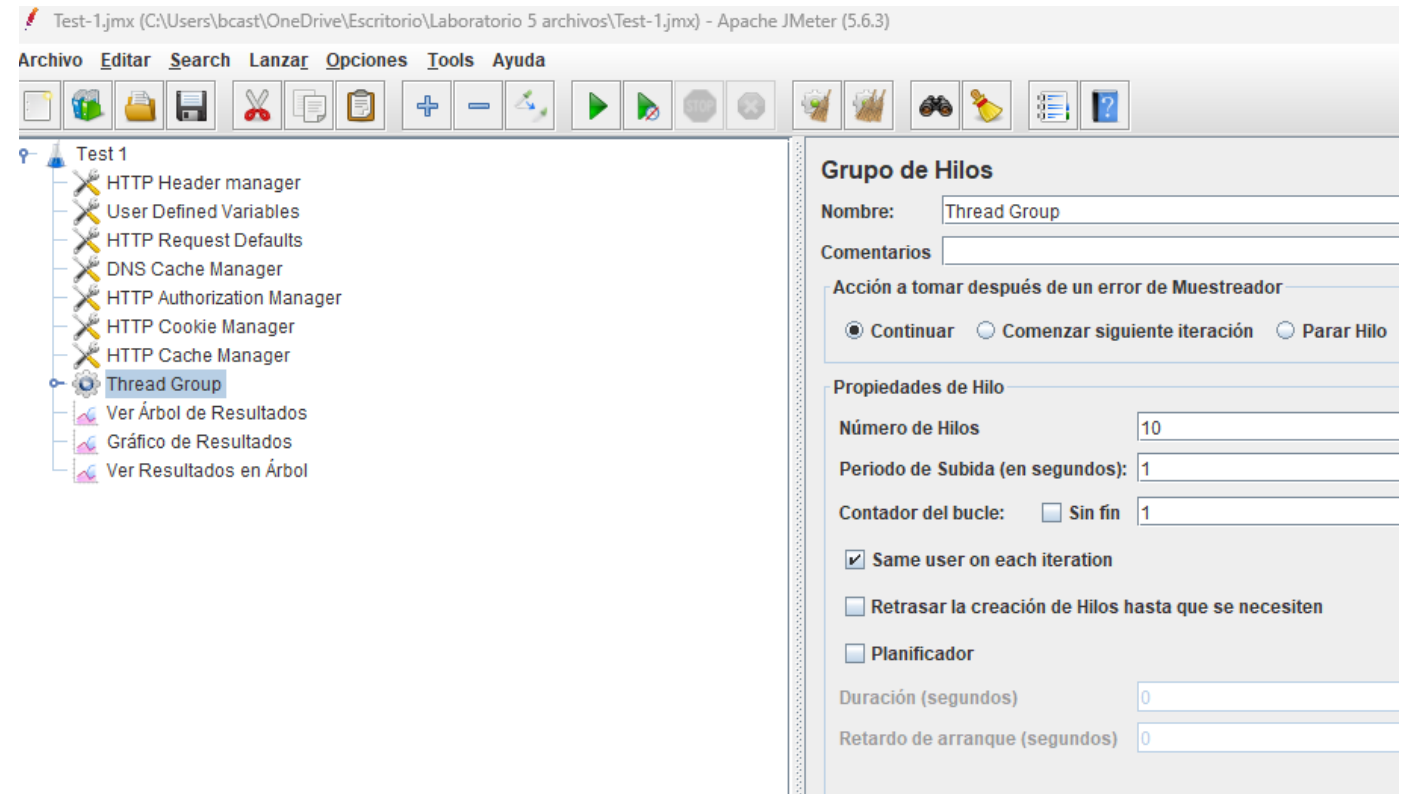
Pruebas

Ahora para poder empezar con el desarrollo procederemos a implementar ciertas vistas como el árbol de resultados, la grafica de resultados y los resultados en arbol



Pruebas

Ahora antes de iniciar la prueba debemos setear los hilos a 10 y luego dar click en inicio para iniciar la prueba



Resultados

Test-1.jmx (C:\Users\bcast\OneDrive\Escritorio\Laboratorio 5 archivos\Test-1.jmx) - Apache JMeter (5.6.3)

Archivo Editar Search Lanzar Opciones Tools Ayuda

Test 1

- HTTP Header manager
- User Defined Variables
- HTTP Request Defaults
- DNS Cache Manager
- HTTP Authorization Manager
- HTTP Cookie Manager
- HTTP Cache Manager
- Thread Group
 - Ver Árbol de Resultados
 - Gráfico de Resultados
 - Ver Resultados en Árbol

Ver Resultados en Árbol

Nombre: Ver Resultados en Árbol

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo

Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo de Muestra (ms)	Estado
1	23:23:58.534	Thread Group 1-5	https://puntoedu.pucp.edu...	1558	✓
2	23:23:58.533	Thread Group 1-5	Test	1558	✓
3	23:23:58.180	Thread Group 1-1	https://puntoedu.pucp.edu...	2289	✓
4	23:23:58.137	Thread Group 1-1	Test	2289	✓
5	23:23:58.332	Thread Group 1-3	https://puntoedu.pucp.edu...	2359	✓
6	23:23:58.331	Thread Group 1-3	Test	2359	✓
7	23:23:58.835	Thread Group 1-8	https://puntoedu.pucp.edu...	1858	✓
8	23:23:58.834	Thread Group 1-8	Test	1858	✓
9	23:23:58.735	Thread Group 1-7	https://puntoedu.pucp.edu...	2017	✓
10	23:23:58.734	Thread Group 1-7	Test	2017	✓
11	23:23:58.434	Thread Group 1-4	https://puntoedu.pucp.edu...	2330	✓
12	23:23:58.433	Thread Group 1-4	Test	2330	✓
13	23:23:58.234	Thread Group 1-2	https://puntoedu.pucp.edu...	2562	✓
14	23:23:58.233	Thread Group 1-2	Test	2562	✓
15	23:23:58.633	Thread Group 1-6	https://puntoedu.pucp.edu...	2222	✓
16	23:23:58.632	Thread Group 1-6	Test	2222	✓
17	23:23:58.933	Thread Group 1-9	https://puntoedu.pucp.edu...	1924	✓
18	23:23:58.932	Thread Group 1-9	Test	1924	✓
19	23:23:59.034	Thread Group 1-10	https://puntoedu.pucp.edu...	1888	✓
20	23:23:59.033	Thread Group 1-10	Test	1888	✓

Ver Árbol de Resultados

Nombre: Ver Árbol de Resultados

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo

Buscar: ☐ Sensible a mayúsculas ☐ Expresión

Texto

- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test
- ✓ https://puntoedu.pucp.edu.pe/coyuntur
- ✓ Test

Resultado del Muestreador

Pruebas con Apis

Para esta prueba utilizaremos un LOGIN de la API REST utilizando la plataforma OctoPerf como punto de acceso.

En esta prueba necesitaremos de los siguientes parametros :

Http: En este caso haríamos uso de una petición Get y una Post.

Esquema Http: Para el ejemplo haríamos uso de https ya que la API Rest que estamos utilizando está protegida por SSL

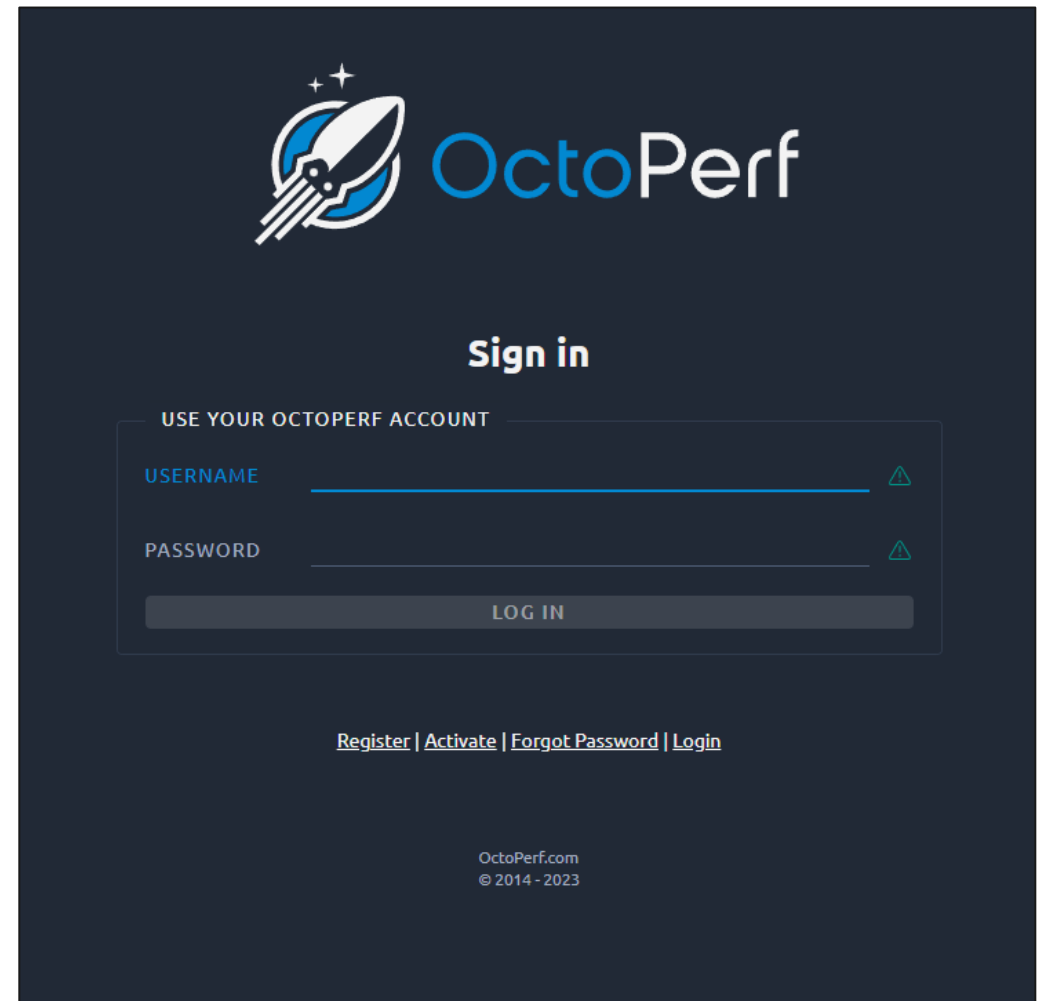
Hostname: api.octoperf.com

Ruta: /public/users/login (Lugar donde se realizará el login)

Otros Requerimientos: Es necesario tener una cuenta previamente creada para poder apuntar a la API con un Usuario y una Contraseña

Número de Puerto : 443

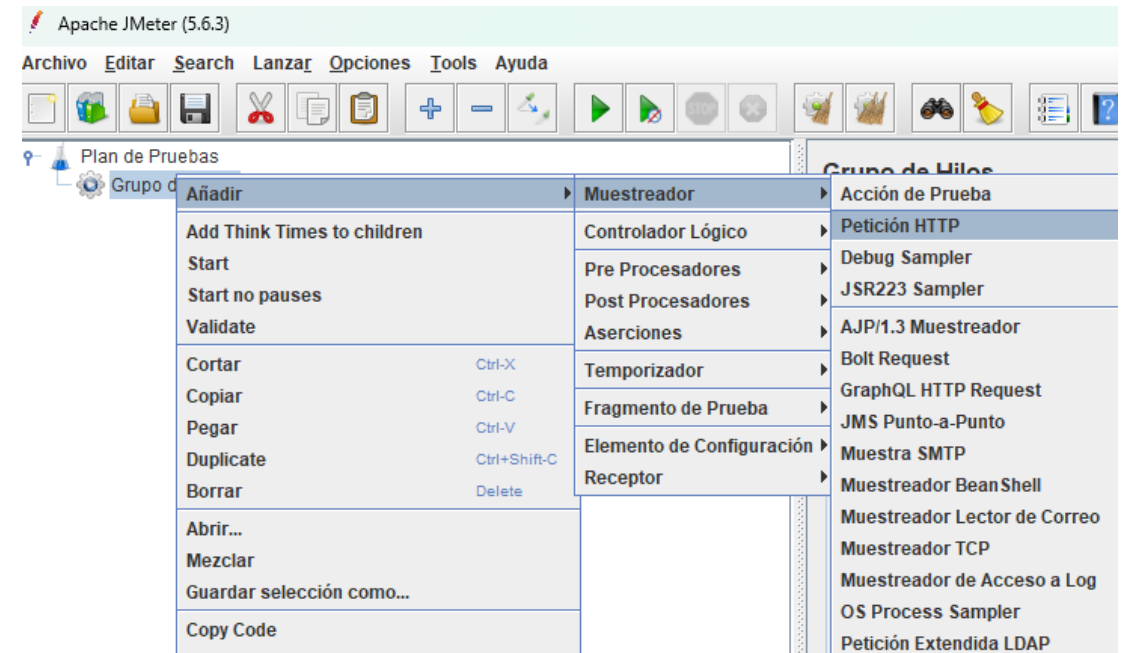
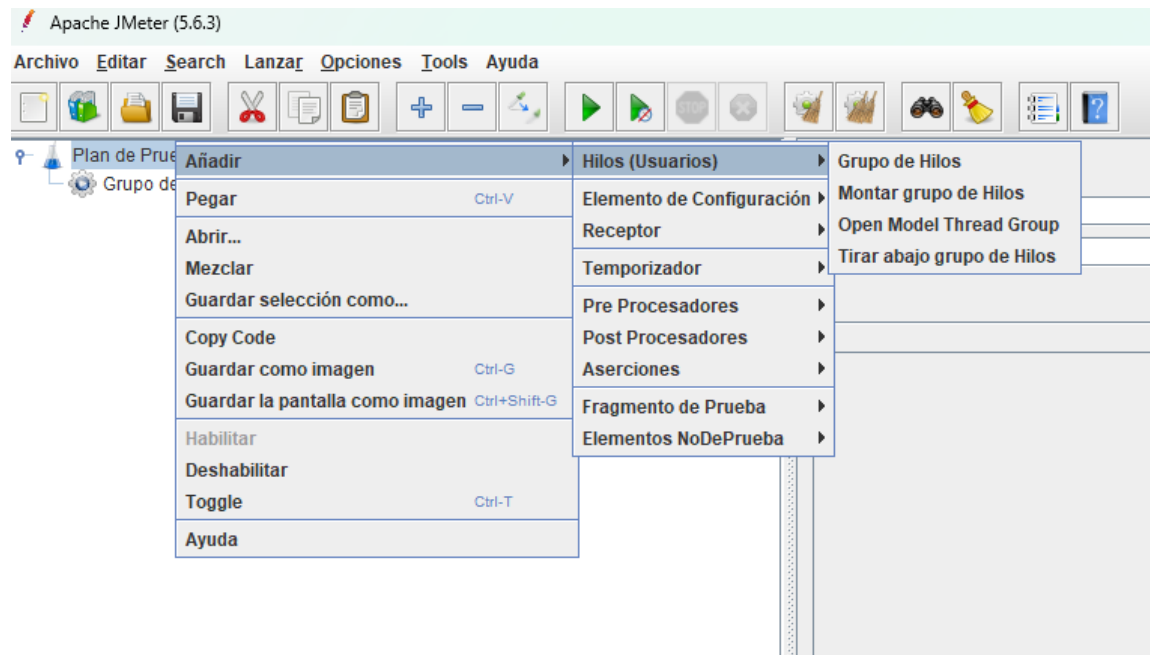
Esta información es accesible en <https://doc.octoperf.com/design/edit-virtual-user/configuration/servers/>



The image shows the OctoPerf Sign in interface. At the top, there is a logo consisting of a stylized rocket ship icon and the text "OctoPerf". Below the logo, the text "Sign in" is displayed. Underneath, there is a section titled "USE YOUR OCTOPERF ACCOUNT" which contains two input fields: "USERNAME" and "PASSWORD". Each input field has a small green triangle icon to its right. Below the input fields is a "LOG IN" button. At the bottom of the form, there are links for "Register", "Activate", "Forgot Password", and "Login". At the very bottom, the text "OctoPerf.com © 2014 - 2023" is visible.

Configuración y desarrollo

Lo primero que necesitamos hacer es adicionar un grupo de hilos para poder empezar con el testeo y a continuación luego debemos adicionar una petición http



Configuración y desarrollo

En la petición HTTP lo que haremos será adicionar el protocolo , el nombre del servidor , el numero de puerto , la ruta y los parámetros que en este caso serian el nombre y la contraseña , además para poder ver los resultados haremos uso de un árbol de pruebas

Petición HTTP

Nombre:

Petición HTTP

Comentarios

Basic

Advanced

Servidor Web

Protocolo:

https

Nombre de Servidor o IP:

api.octoperf.com

Puerto:

443

Petición HTTP

POST

Ruta:

public/users/login

Codificación del contenido:

☐ Redirigir Automáticamente

☒ Seguir Redirecciones

☒ Utilizar KeepAlive

☐ Usar 'multipart/form-data' para HTTP POST

☐ Cabeceras compatibles con navegadores

Parameters

Body Data

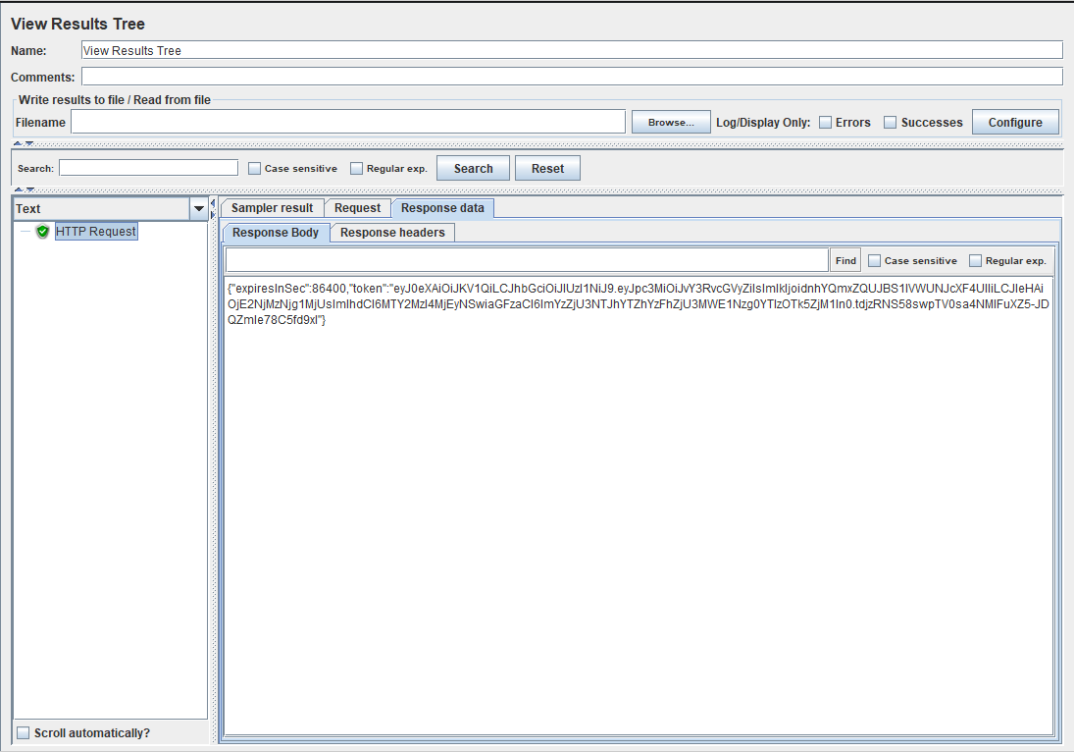
Files Upload

Enviar Parámetros Con la Petición:

Nombre:	Valor	¿Codificar?	Content-Type
username	b.castillo@pucp.pe	<input type="checkbox"/>	text/plain
password		<input type="checkbox"/>	text/plain

Configuración y desarrollo

Como respuesta se nos brinda un token que es el que esta asignado hacia nuestro usuario.



Dado que la respuesta en el momento de hacer el post es un token, podemos decir que su autenticación se basa en el uso de un token, siendo un mecanismo donde cada token identifica a un usuario en una única sesión.

Así que vamos a utilizar este token para llamar a nuevos métodos

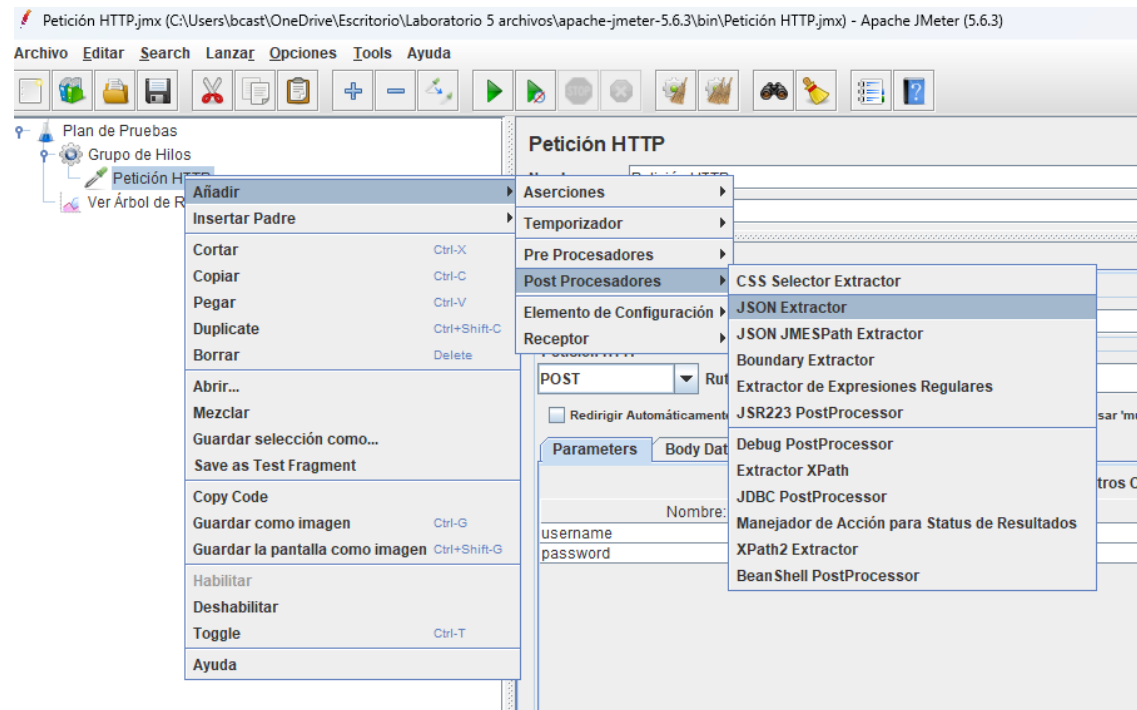
Estos métodos que puede manejar esta api los podemos ver en:

[**https://api.octoperf.com/swagger-ui/index.html#/**](https://api.octoperf.com/swagger-ui/index.html#/)

POST	/workspaces	Create a new workspace
GET	/workspaces/{id}	Find an Item by ID
PUT	/workspaces/{id}	Update an item by ID
DELETE	/workspaces/{id}	Delete an item by ID
POST	/workspaces/all	List tests by ids
GET	/workspaces/member-of	List workspaces whose logged user is a member

Desarrollo

Ahora lo que haremos será extraer el token proporcionado, para eso usaremos el extractor de Json en JMeter y retiraremos el token usando la expresión \$.token donde buscaremos el primer match y para poder visualizar el resultado utilizaremos un debug sampler



JSON Extractor

Nombre:

Comentarios:

Aplicar a:

☐ Muestra principal y submuestras ☒ Sólo muestra principal

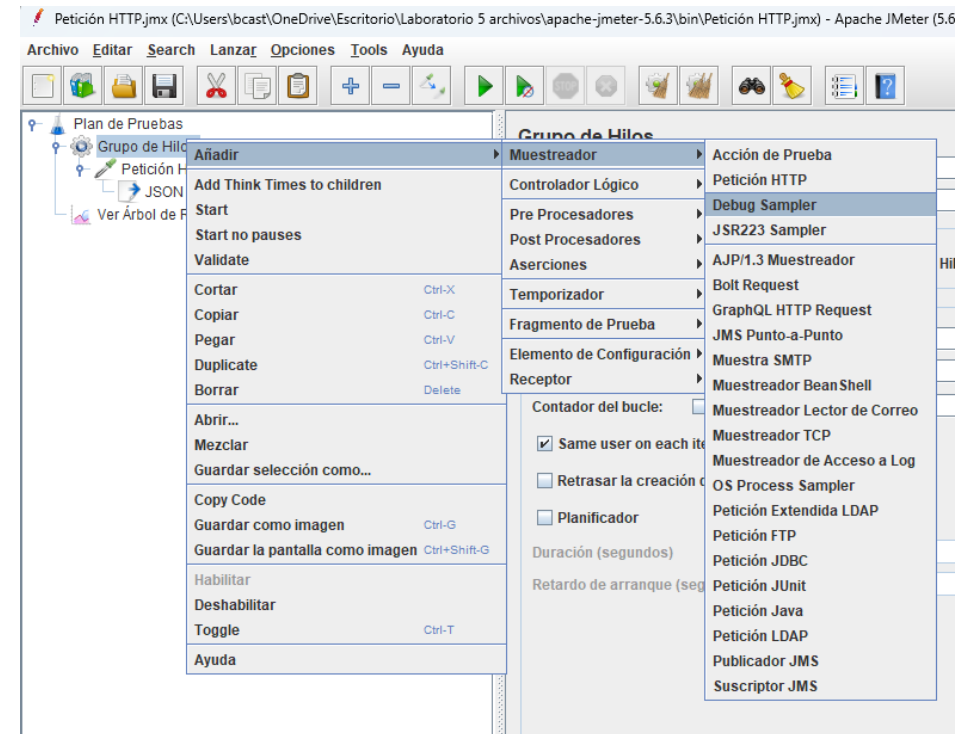
Names of created variables:

JSON Path expressions:

Match No. (0 for Random):

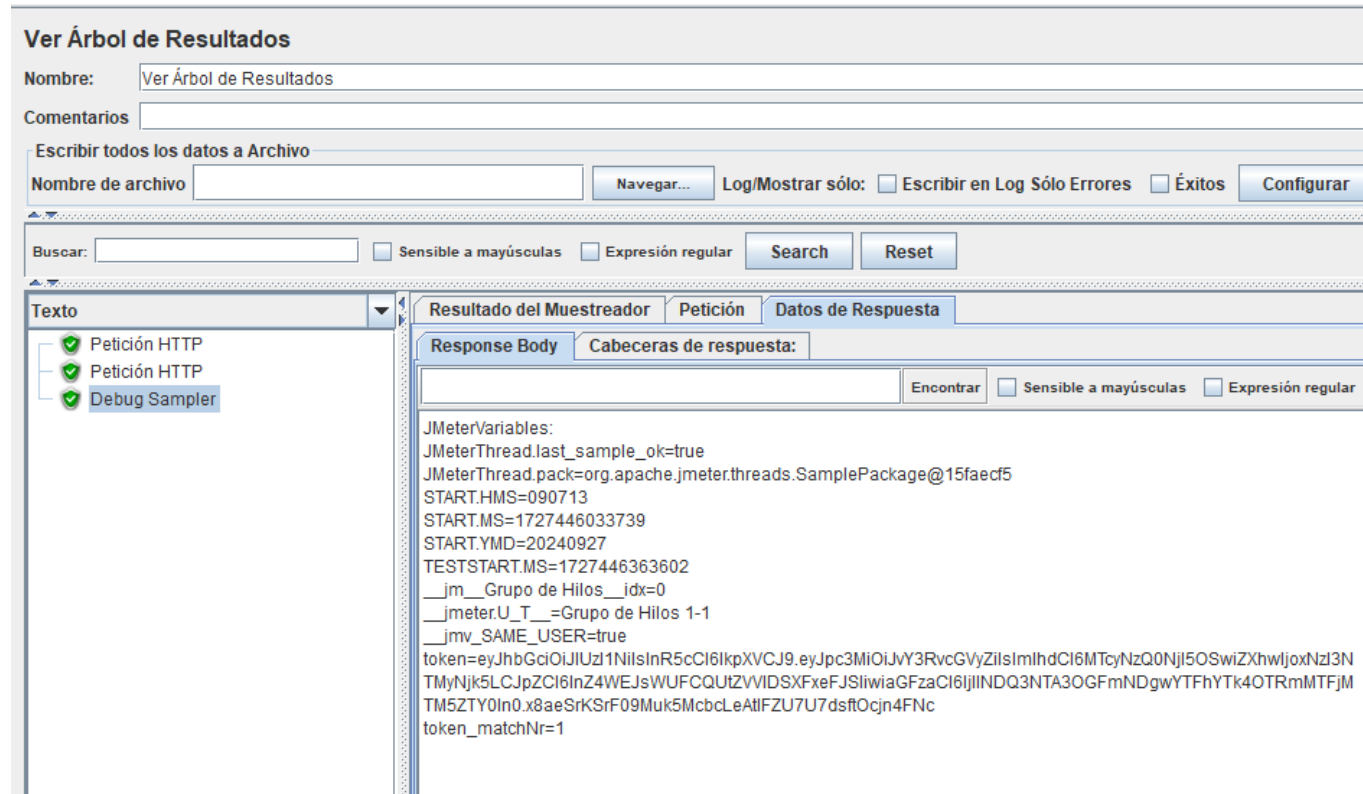
Compute concatenation var (suffix _ALL): ☐

Default Values:



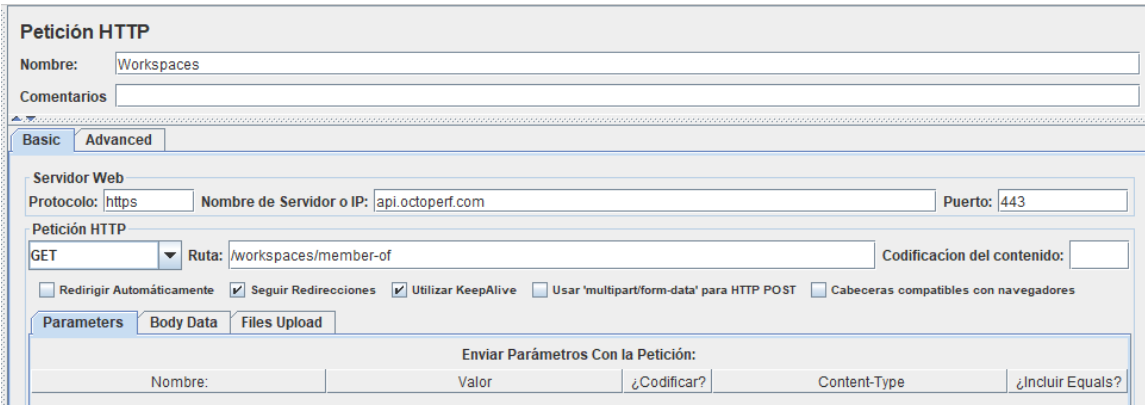
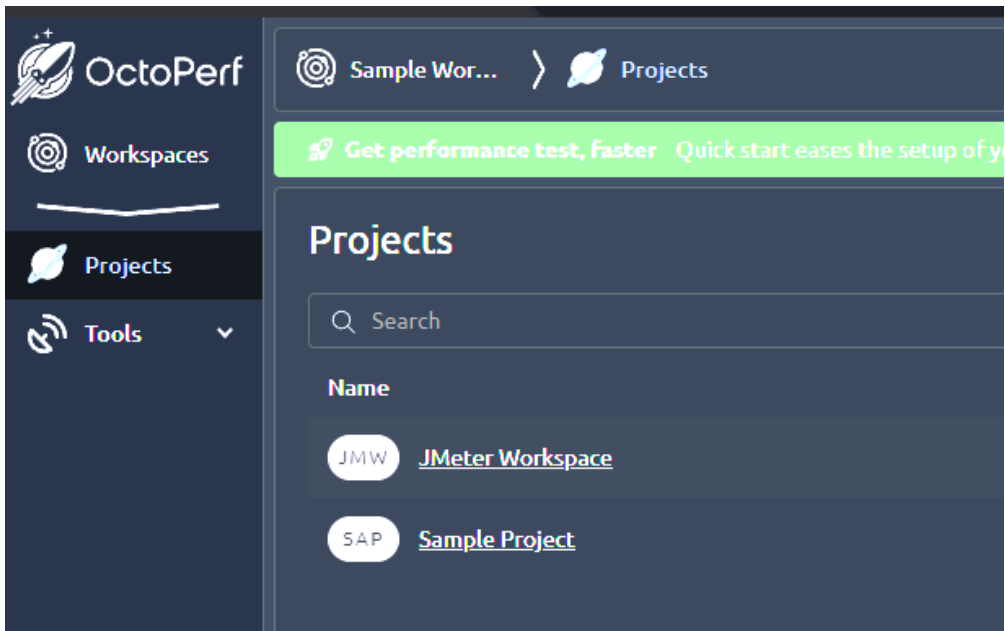
Resultados

Aquí podremos ver los resultados obtenidos donde se puede observar que el token es obtenido correctamente



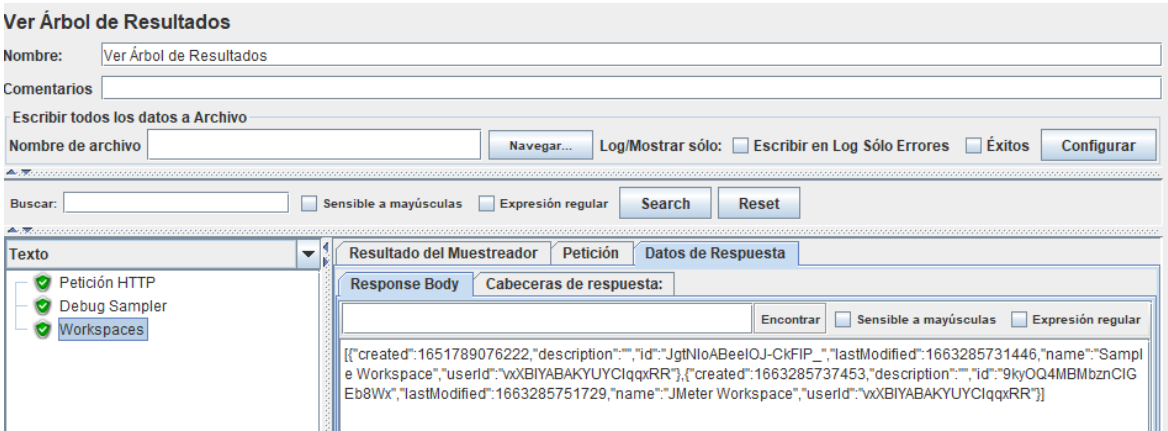
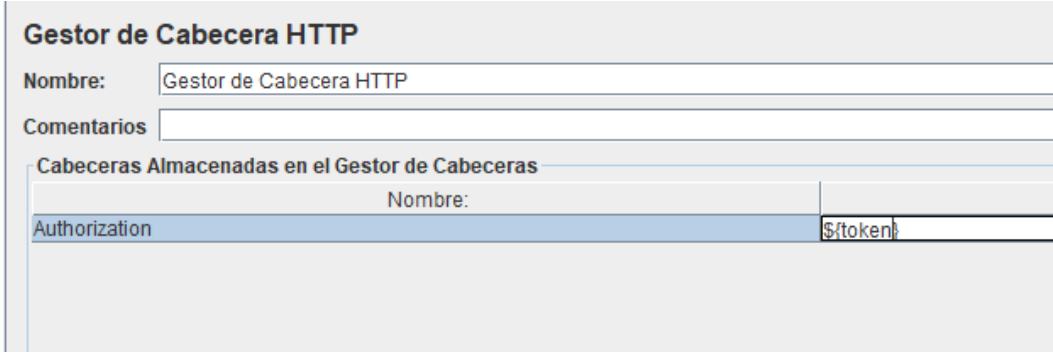
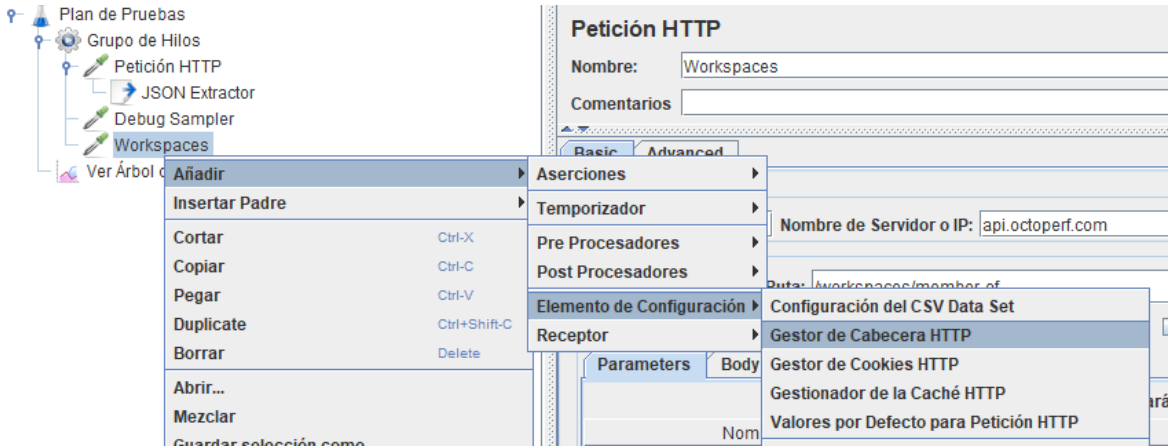
Pruebas con Apis

Ahora para poder seguir trabajadno con el API de octoperf vamos a obtener los workspaces usando el método Get donde deberíamos poder visualizar los workspaces de Sample Workspace y Jmeter Workspace para eso haremos uso de la solicitud HTTP pero apuntando ahora a los workspaces



Pruebas con Apis

Para evitar que se nos muestre el error de no autorización debemos aparte de incluir el login en la primera petición HTTP , debemos incluir el token, para esto en nuestra petición de workspaces tener que usar un gestor de cabeceras para tener la autorización utilizando el valor \${token}



Stress Test

Ahora que ya hicimos pruebas de funcionalidad , haremos uso de las pruebas de rendimiento, para esto se modificara el Thread Group a un numero superior, lo que asemejara una cadena de peticiones al mismo tiempo, por ejemplo 1000 threads serian 1000 peticiones simultaneas

Grupo de Hilos

Nombre:

Grupo de Hilos

Comentarios

Acción a tomar después de un error de Muestreador

☒ Continuar

☐ Comenzar siguiente iteración

☐ Parar Hilo

Propiedades de Hilo

Número de Hilos

1000

Periodo de Subida (en segundos):

1

Contador del bucle:

☐ Sin fin

1

☒ Same user on each iteration

☐ Retrasar la creación de Hilos hasta que se necesiten

☐ Planificador

Duración (segundos)

Retardo de arranque (segundos)

