Realizado por Andres Mauricio Diaz de la Rue

Fecha:08/06/2022

# RETO JMETER

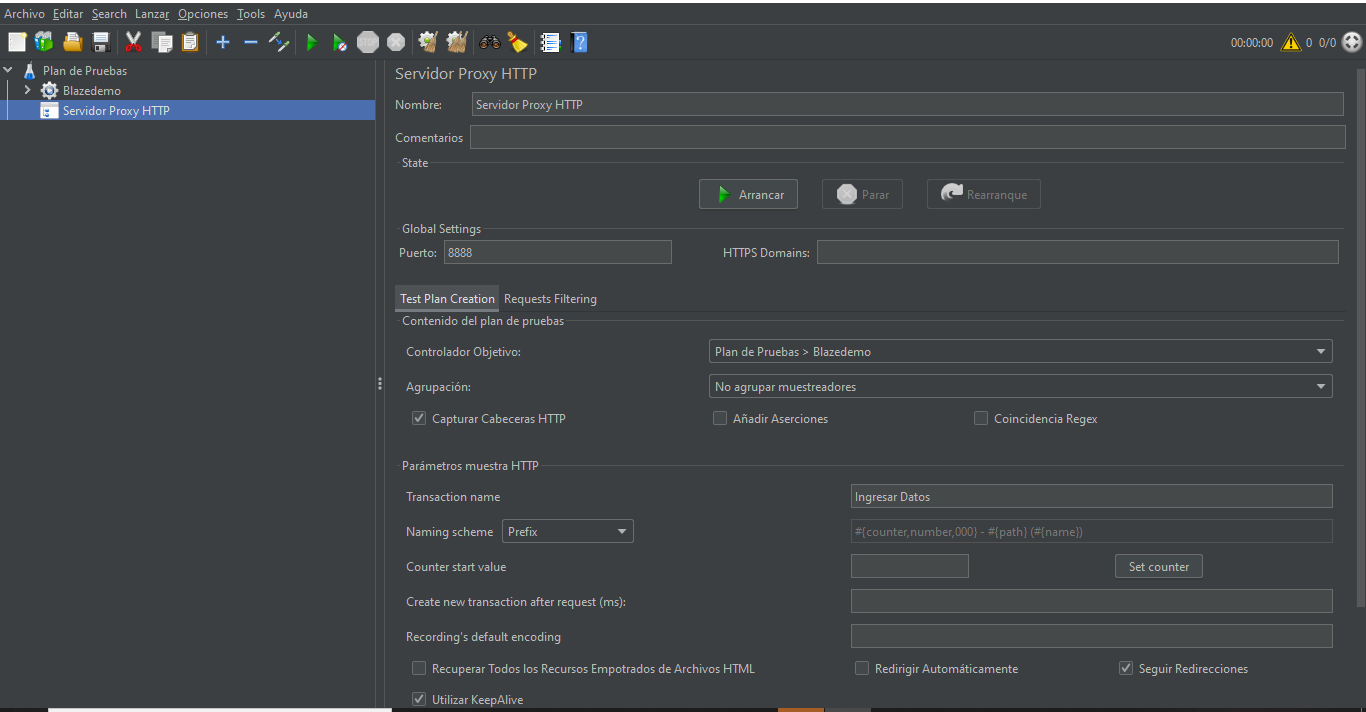
## Pasos:

Se realiza la creación de un grupo de hilos que se encargara de almacenar todas las peticiones y la información que este viajando entre JMETER y el navegador al cual se le asigna el nombre de Blazedemo, este grupo de hilos será utilizada mas adelante para gestionar toda información relacionada con la concurrencia de los usuarios en el sistema.

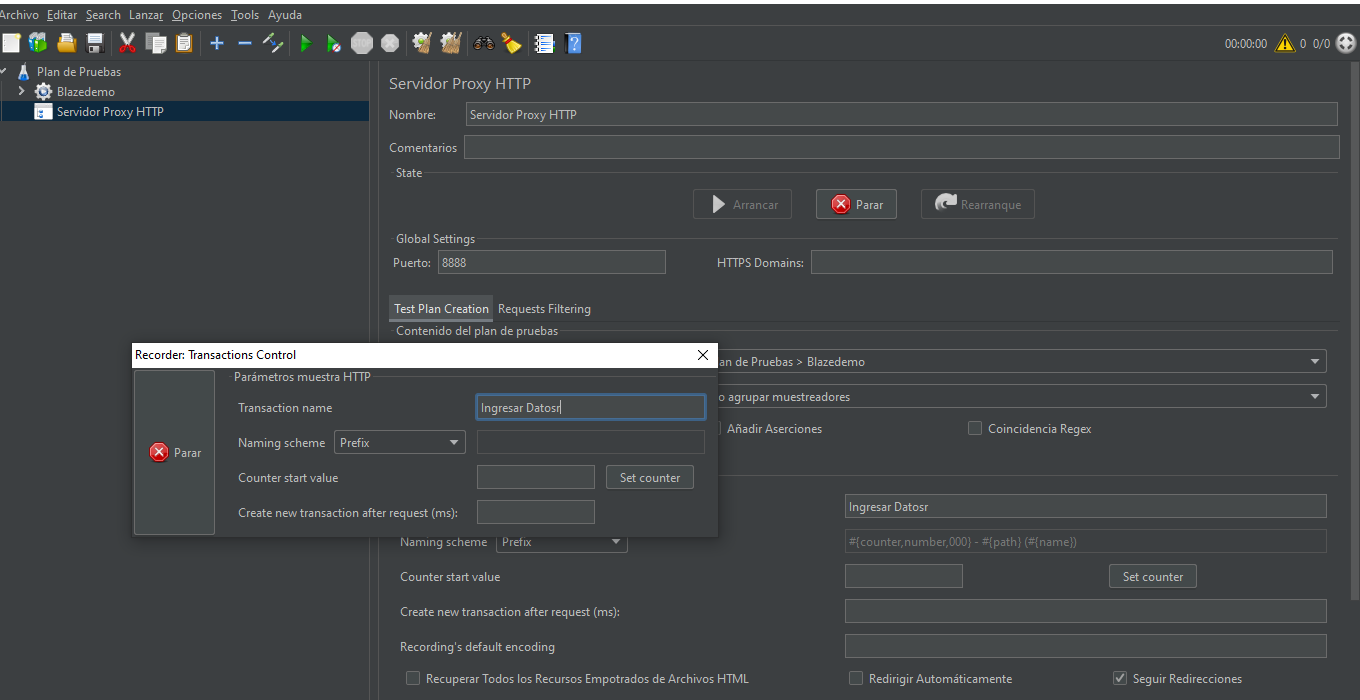
Pero inicialmente lo estaremos utilizando para almacenar todo lo que viaje entre el protocolo que se ha configurado en JMETER , que será el Servidor Proxy HTTP por el puerto 8888 el cual se debe configurar de igual manera en el navegador.

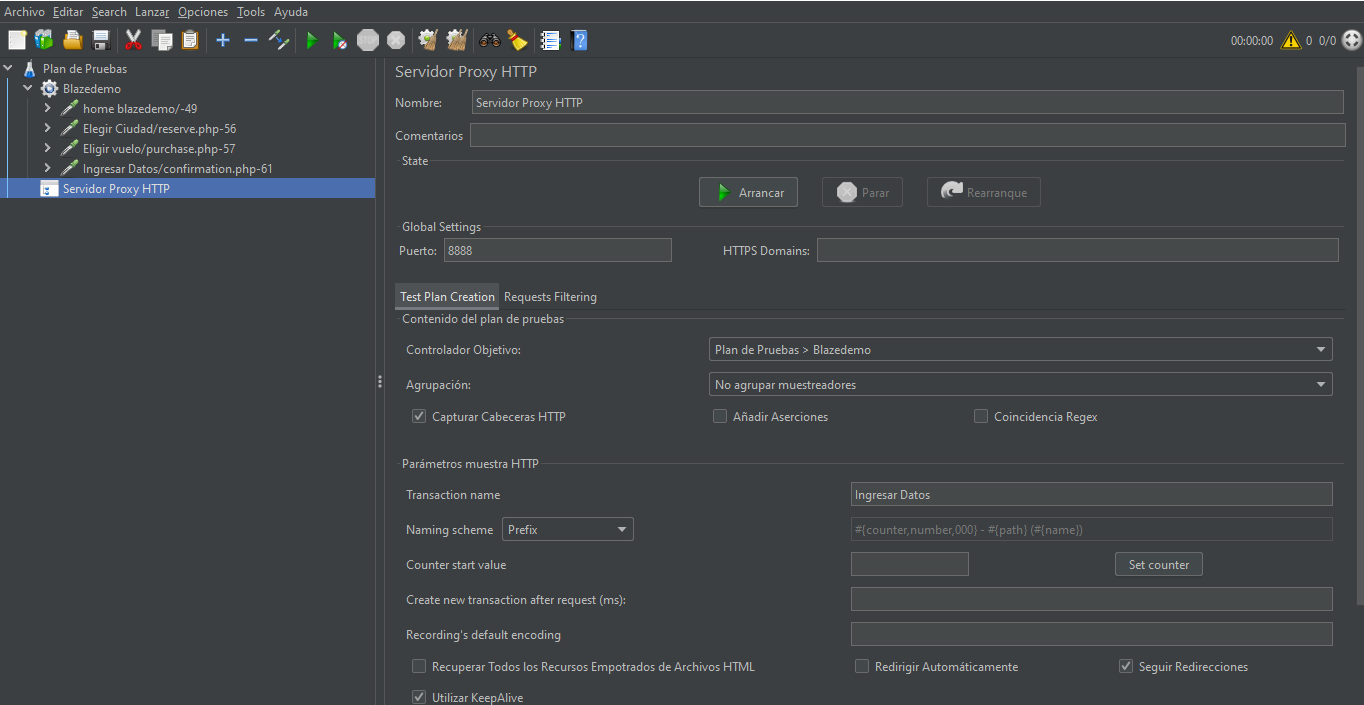
Entonces el siguiente paso sería, para poder iniciar la grabación seria seleccionar el CONTROL DE OBJETIVOS y seleccionar el grupo de hilos que hemos creado con el nombre de Blazdemo, ya que una vez se empiecen a grabar todas las peticiones y la información referente a la prueba que se va a estar ejecutando se va a estar almacenando en el grupo de hilos internamente.

El siguiente paso es realizar un onclick en el botón ARRANCAR automáticamente se va a mostrar un certificado al cual se le debe de dar aceptar, que está relacionado con las conexiones seguras lo que quiere decir es HTTPS que lo debemos tener en cuenta.



Automáticamente nos aparece un POP UP el cual es útil para poder asociar cada petición a cada acción que esté realizando en el script y poderlas identificar esto no da la posibilidad de poder entender, comprender, y estructurar organizar el script.

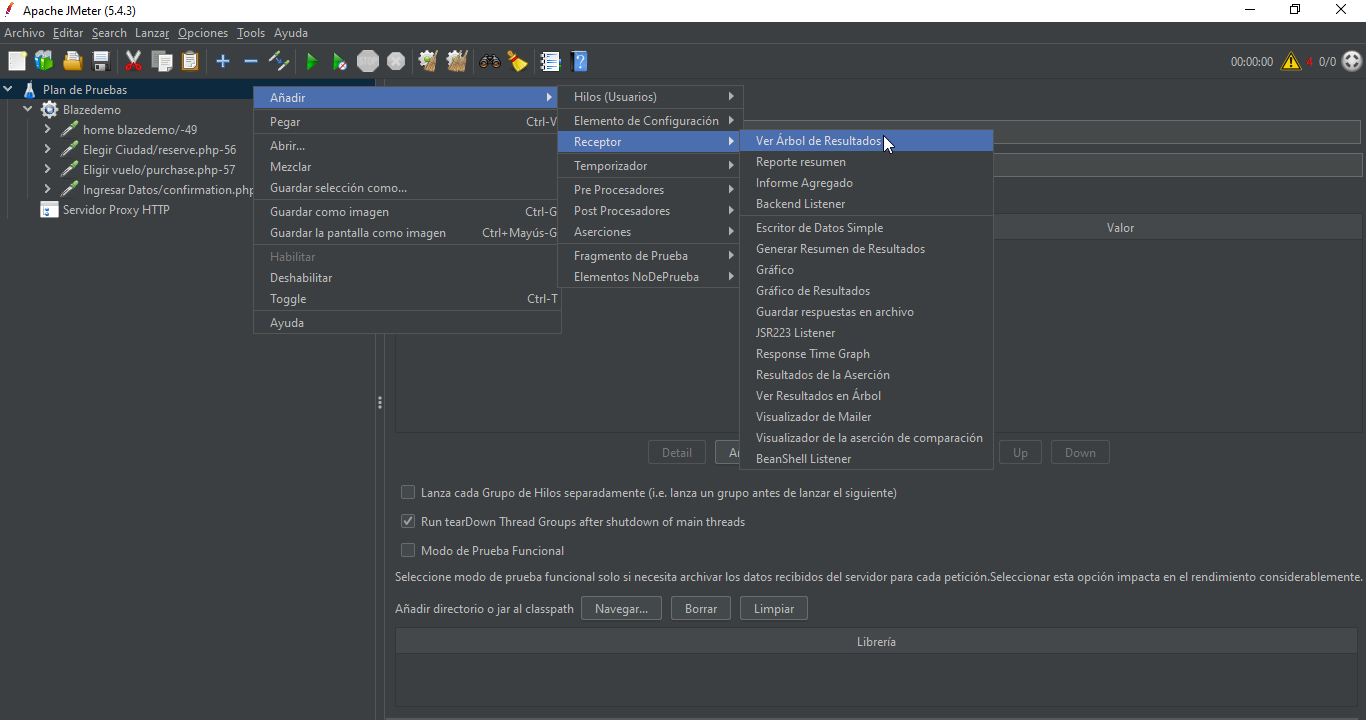


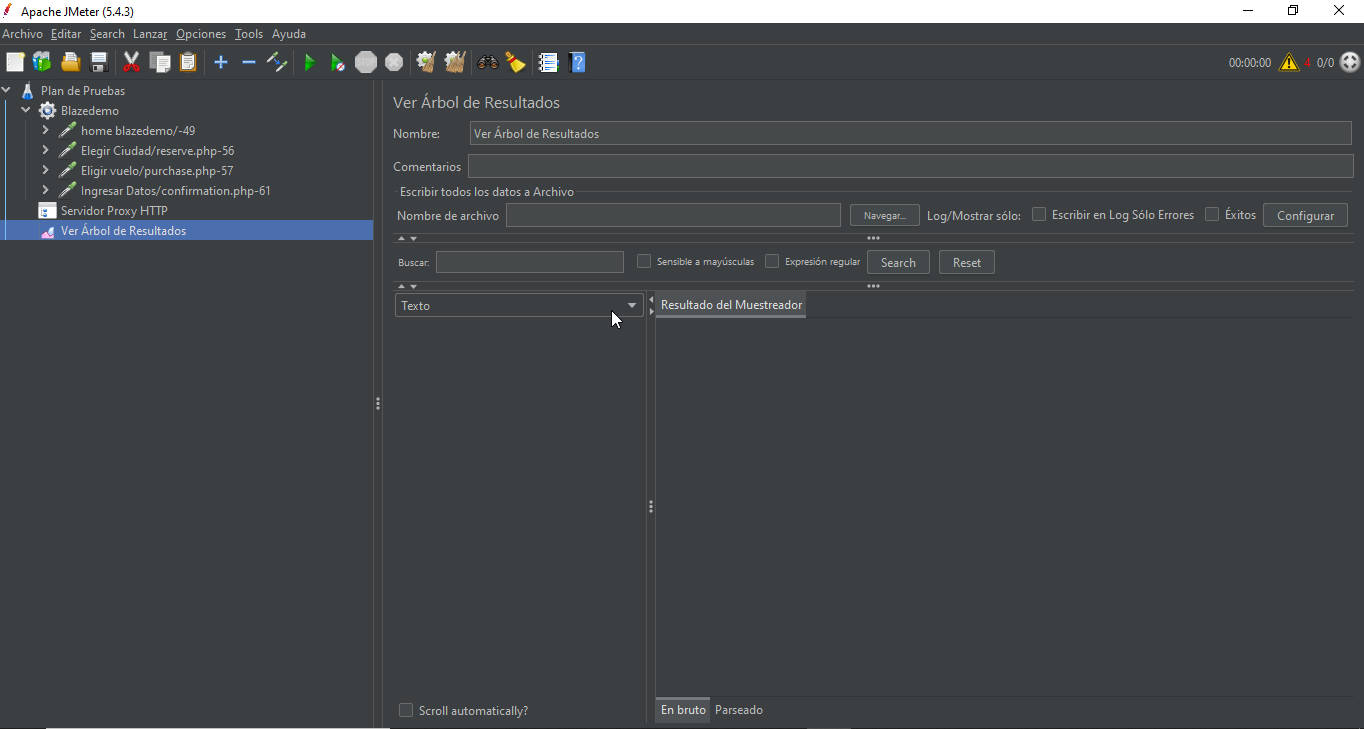


# A continuación la introducción hacia los Listeners o Receptores de JMETER

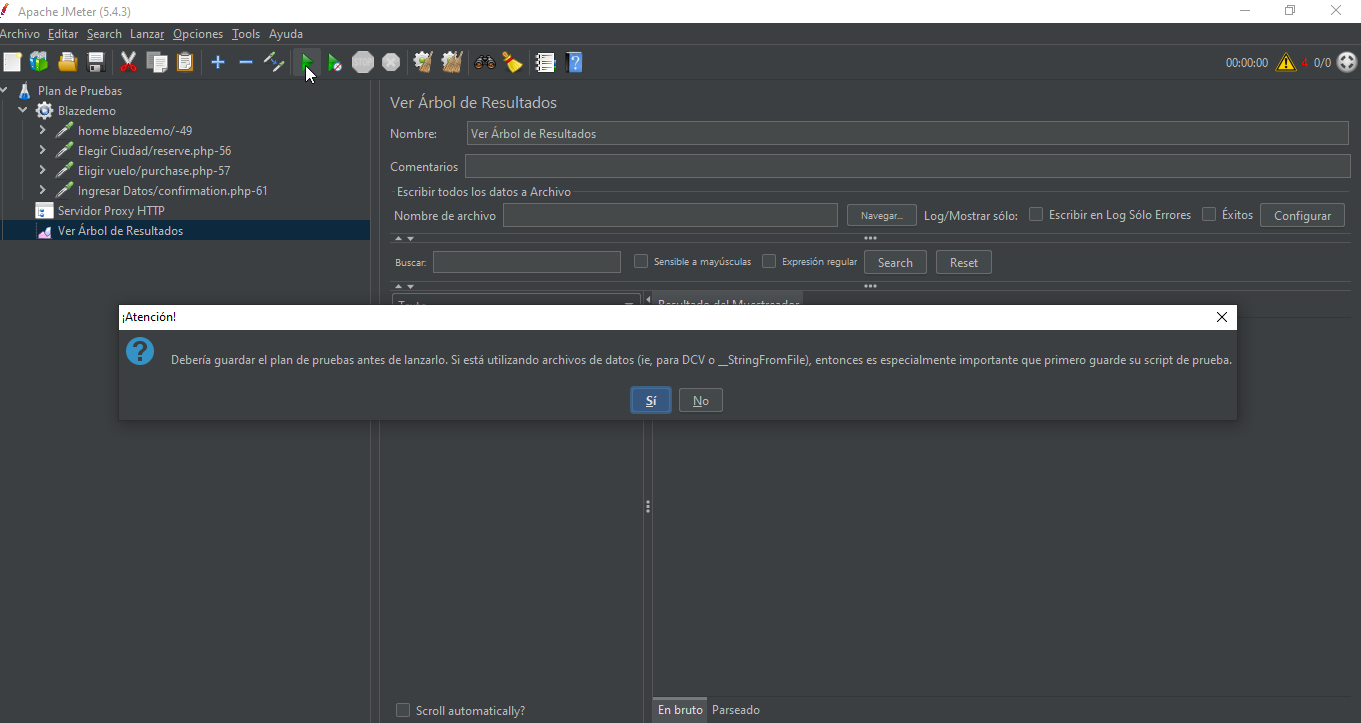
Estos serán los que voy a utilizar en el script como el nombre lo indica son escuchadores que estarán captando todos los datos que se estarán generando a partir de la ejecución de las pruebas los cuales son dos principales:

Árbol de Resultados: Lo que nos indica es que peticiones están fallando y cuales han sido exitosas y además nos van a mostrar en si lo que llevan esas peticiones, los datos que están internos no solo de las peticiones si no también nos muestra las respuestas a asociadas a la petición.

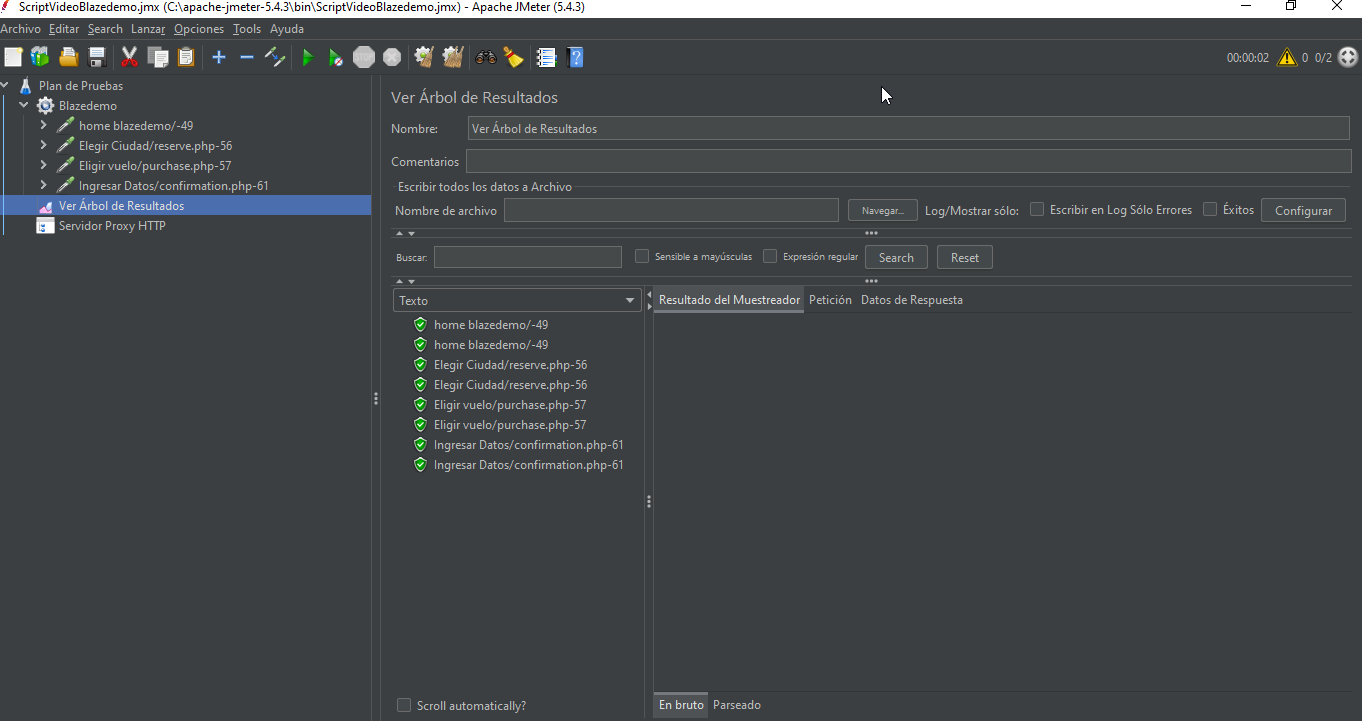




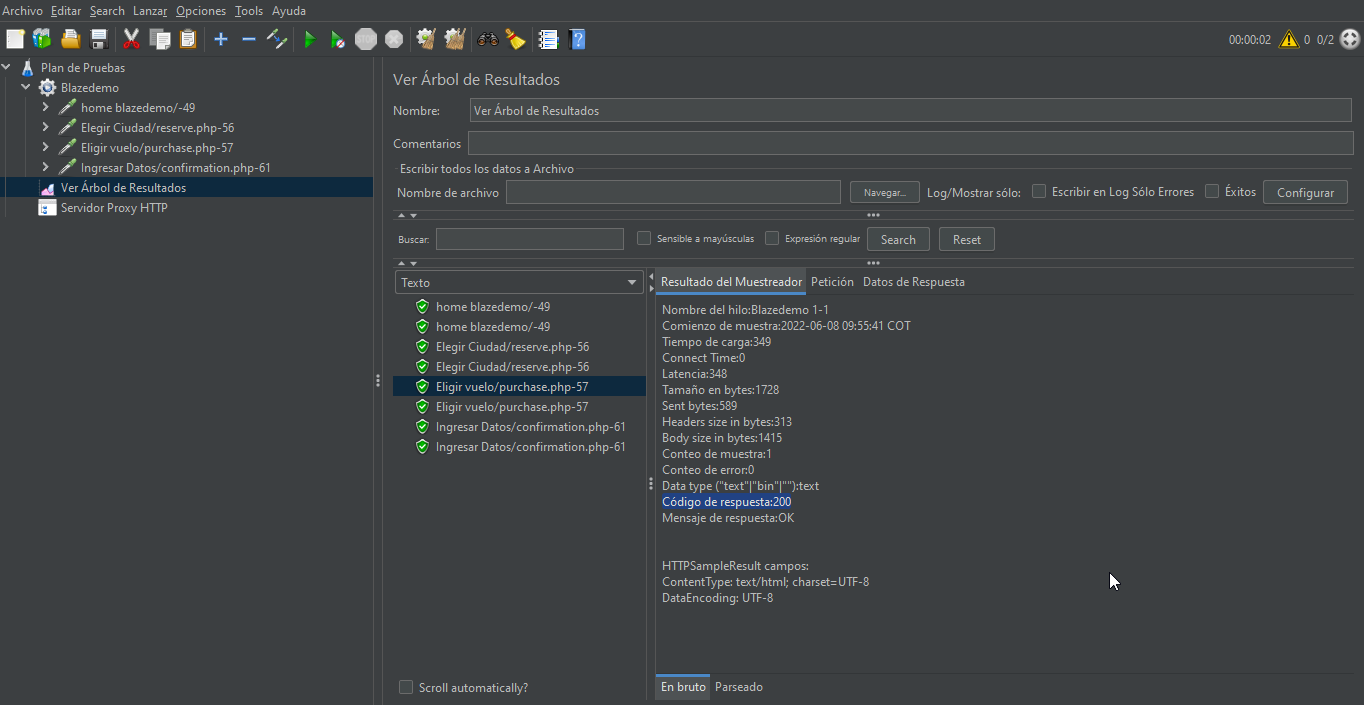
Después se ejecuta el programa el cual nos va mostrar un POP UP que me indica que debo guardar el script en alguna carpeta de nuestra máquina y se le debe dar un nombre representativo con relación a las pruebas a realizar al cual debemos darle click a la opción (SI) por lo general nos sugiere guardarlo en la carpeta BIN que muestra por defecto



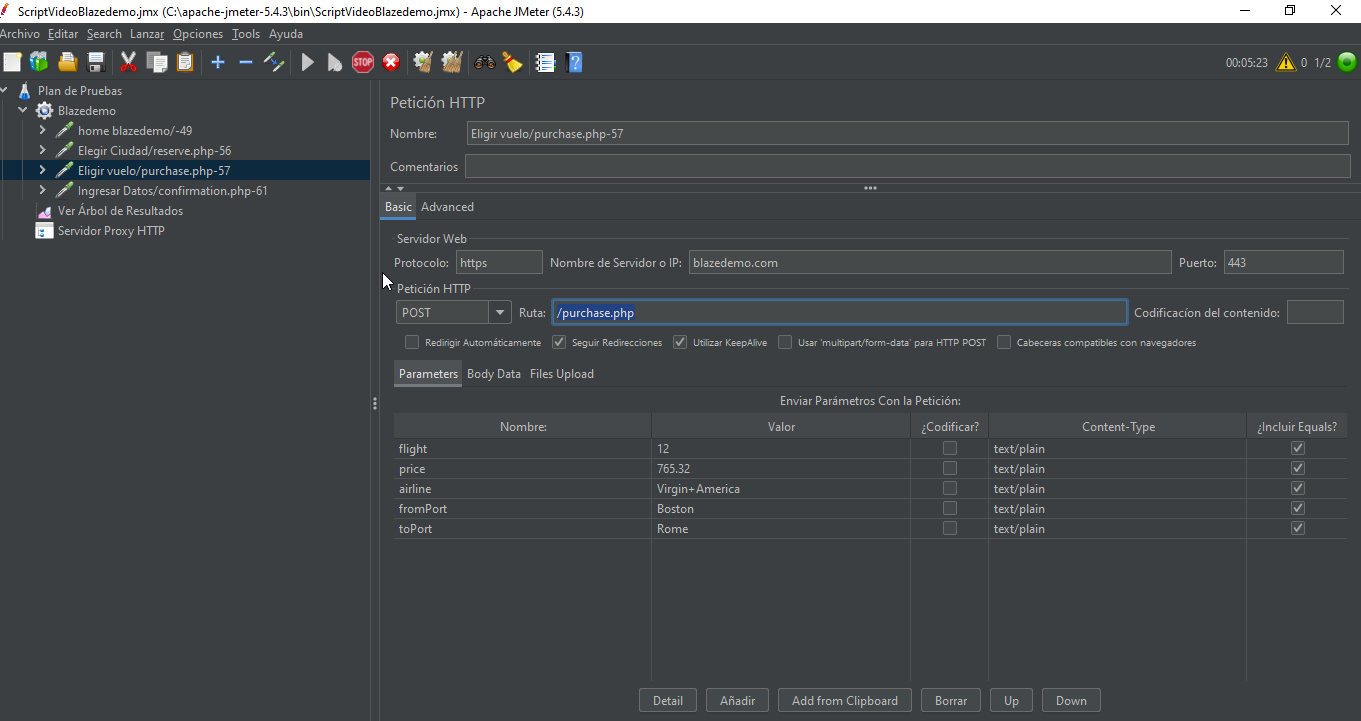
Luego realizó la ejecución del script para poder visualizar en Árbol de Resultados los resultados asociados a cada petición el grupo de hilos para lo prueba y toma de evidencia lo mantendré en 2 realizare una ejecución.



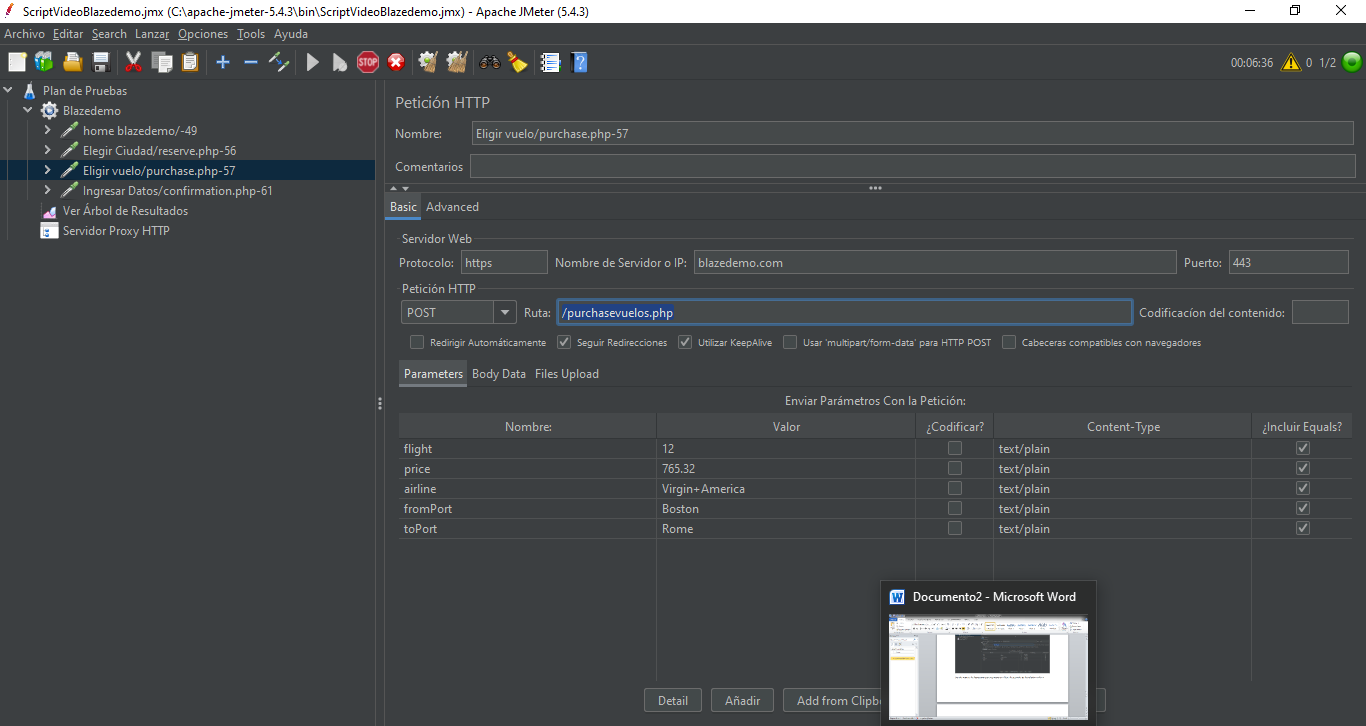
Como se puede evidenciar al visualizar el árbol de resultados todas las peticiones llegaron en un color verde ya que obtuvieron un código de respuesta 200 que me indica un código de confirmación que todo fue exitoso para esta ejecución de prueba.



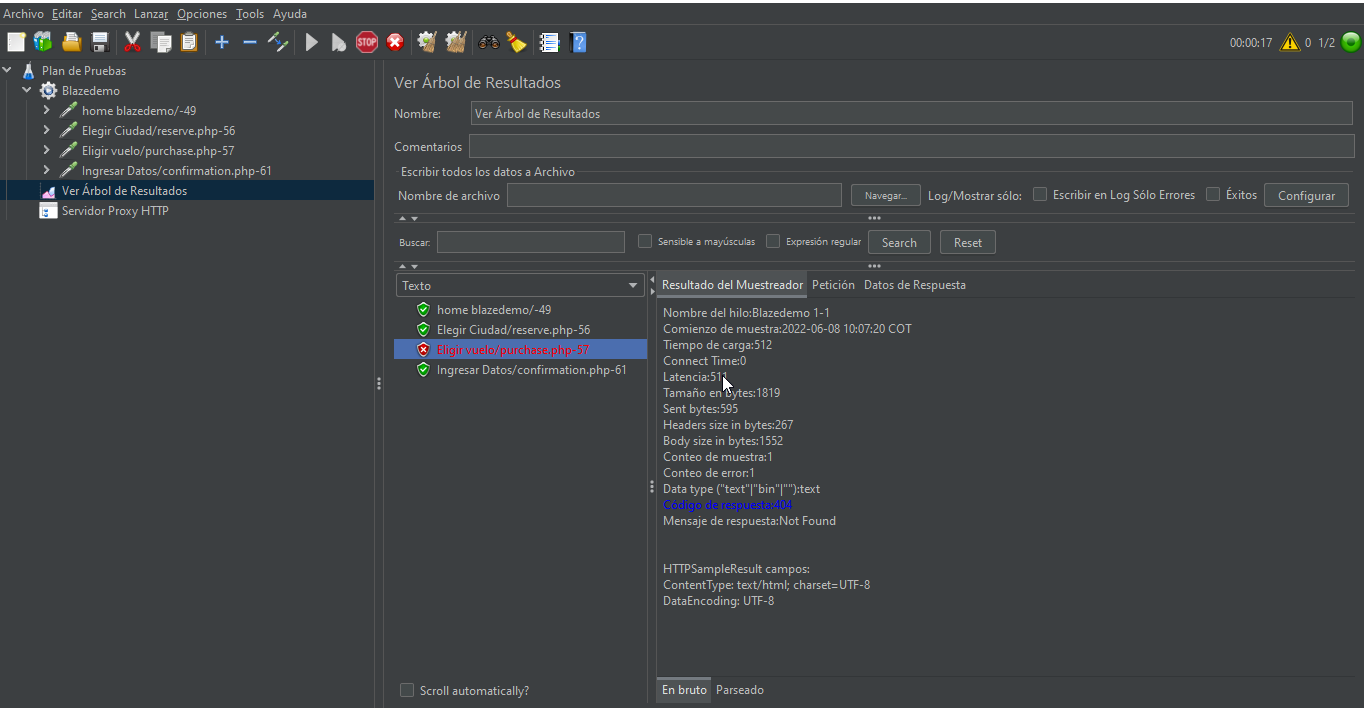
Pero pueden ocurrir que la petición nos genere otros códigos de respuesta como el de un error si tal vez la Url está mal estructurada como ejemplo voy a alterar la Url de esta petición llamada Elegir vuelo y esto me generara un error así es como esta para que la petición sea exitosa



De esta manera la altero para que me genere un código de respuesta de la petición con Error y realizo la ejecución del script de nuevo

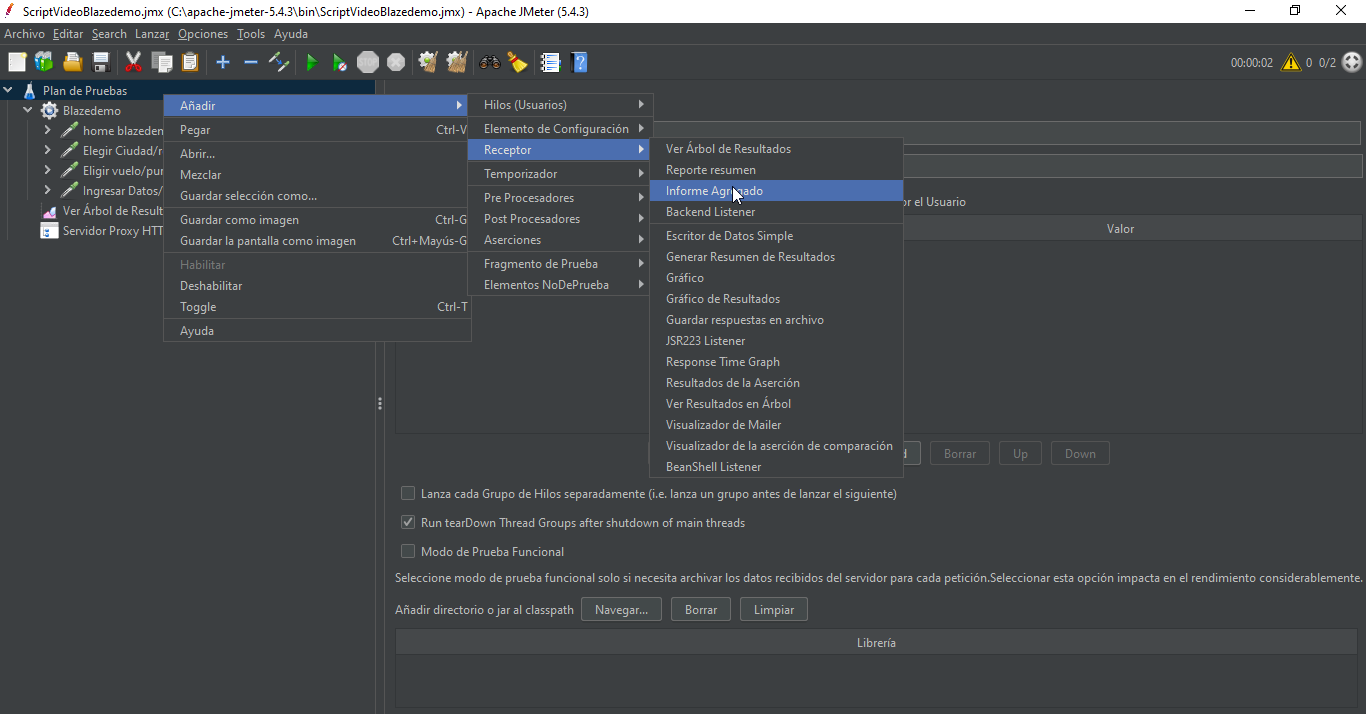


Como se puede evidenciar la petición de Elegir vuelo que le altere la url me genero un código de respuesta 404 que es que el recurso no se encuentra ya que no concuerda con la url de la petición que debe ser.

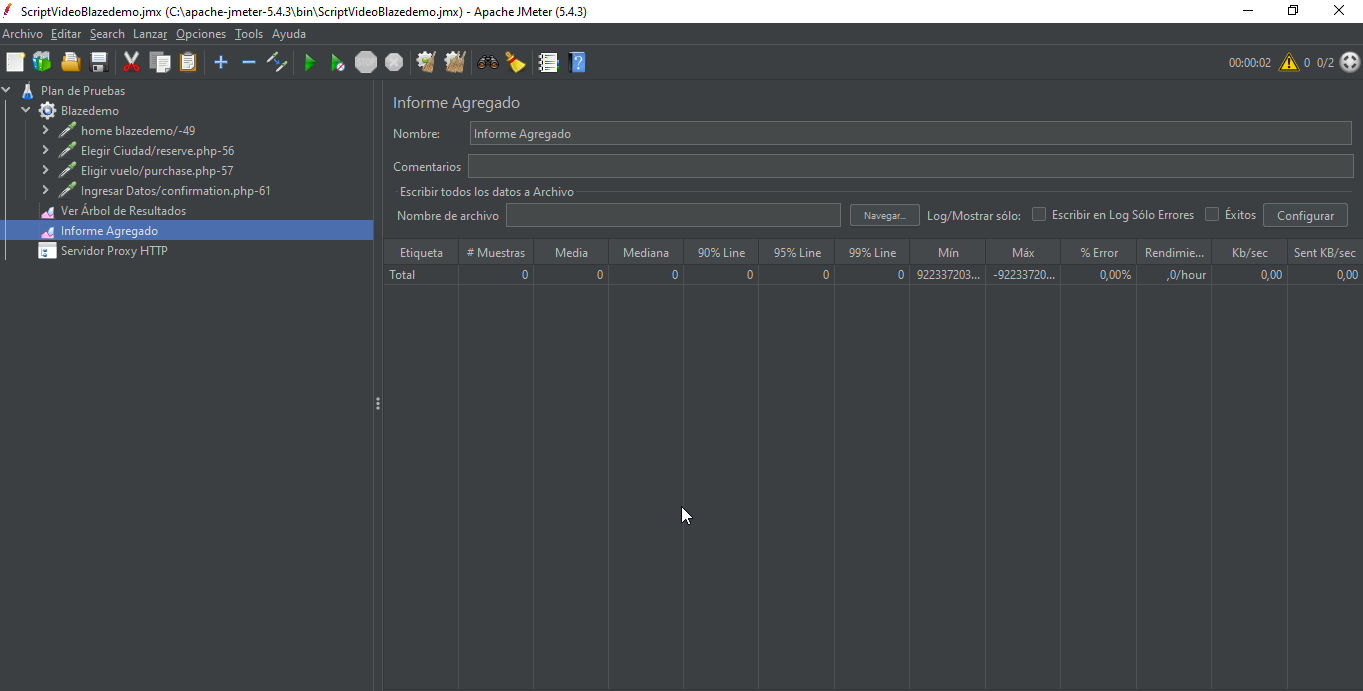


# A continuación la introducción al segundo Listener o Receptor que es el de Informe Agregado de JMETER

Este listener hace énfasis en los tiempos de respuesta asociados a las peticiones se implementa seleccionando de igual manera que el listener de Arbol de Resultados como se evidencia a continuación.

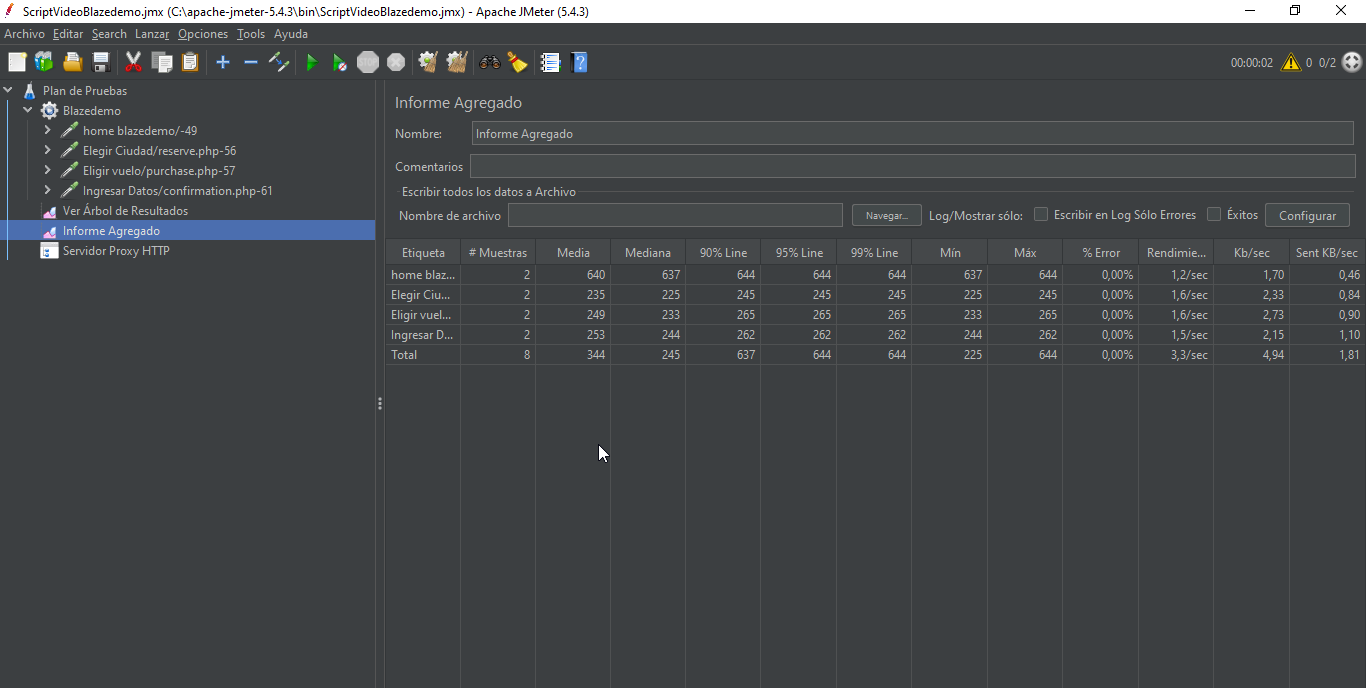


Es una tabla bien referenciada con varios datos que no evidencia inicial mente el mínimo y el máximo donde la aplicación muestra cuanto demoro en realizar la petición también tener en cuenta el 90% de las líneas y el 95% para esto realizare la prueba de ejecución con 2 usuarios y posterior mente con 20 usuarios para evidenciar la diferencia de los tiempos con esos usuarios.

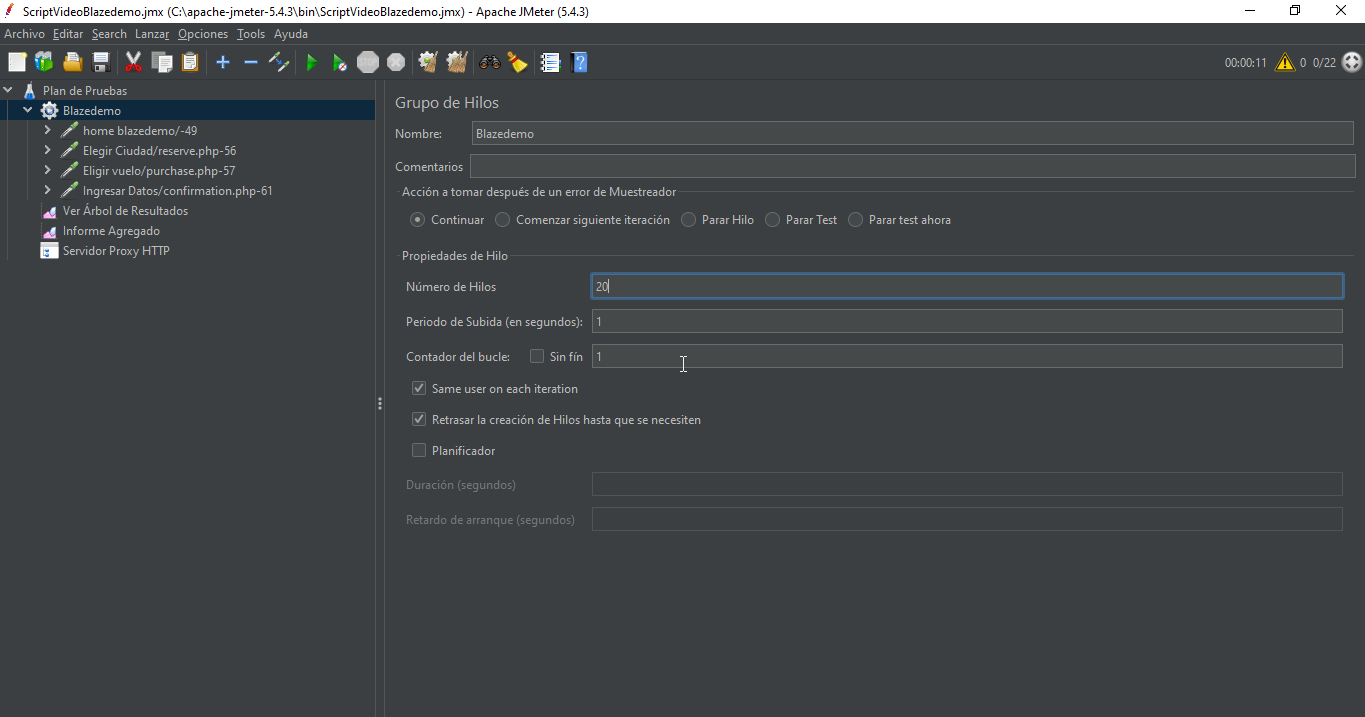


# Resultado de la Ejecución

Puedo evidenciar los tiempos asociados a cada una de las peticiones y me deja visualizar si alguna tiene error por ejemplo en este caso el máximo de tiempo fue 0,644 milisegundos para todos porque solo fueron 2 usuarios.

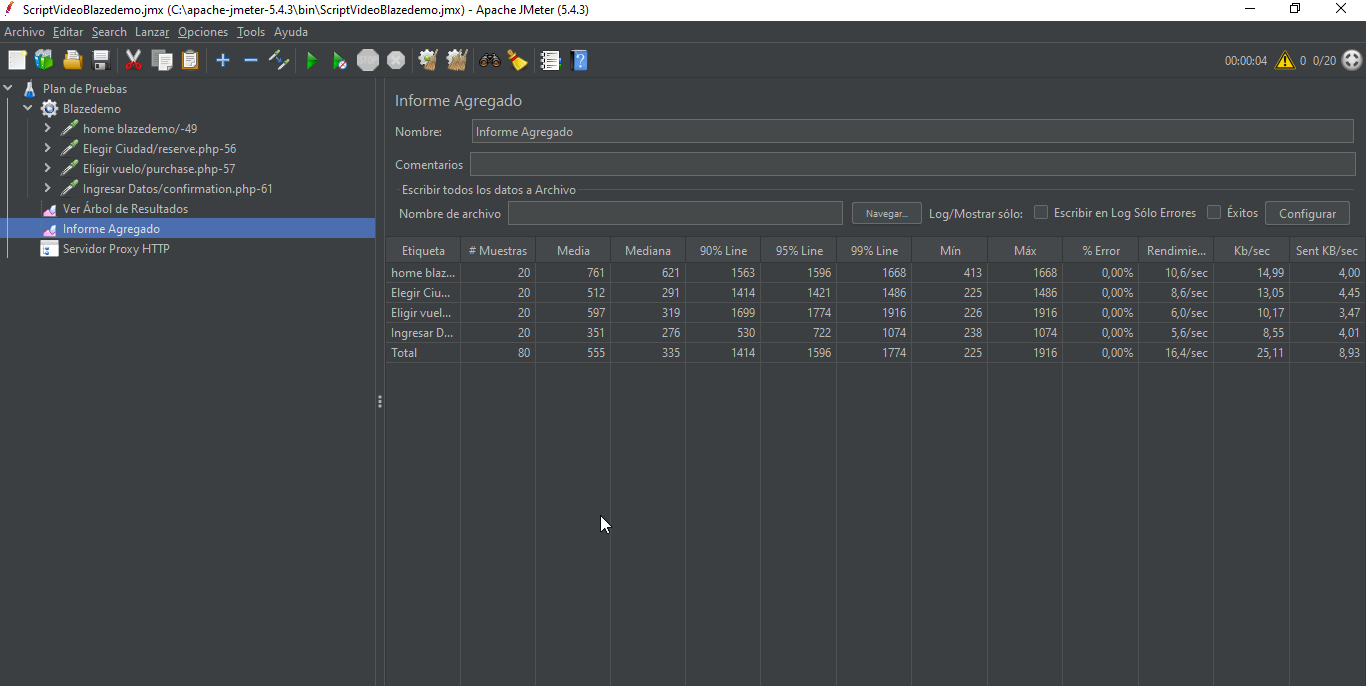


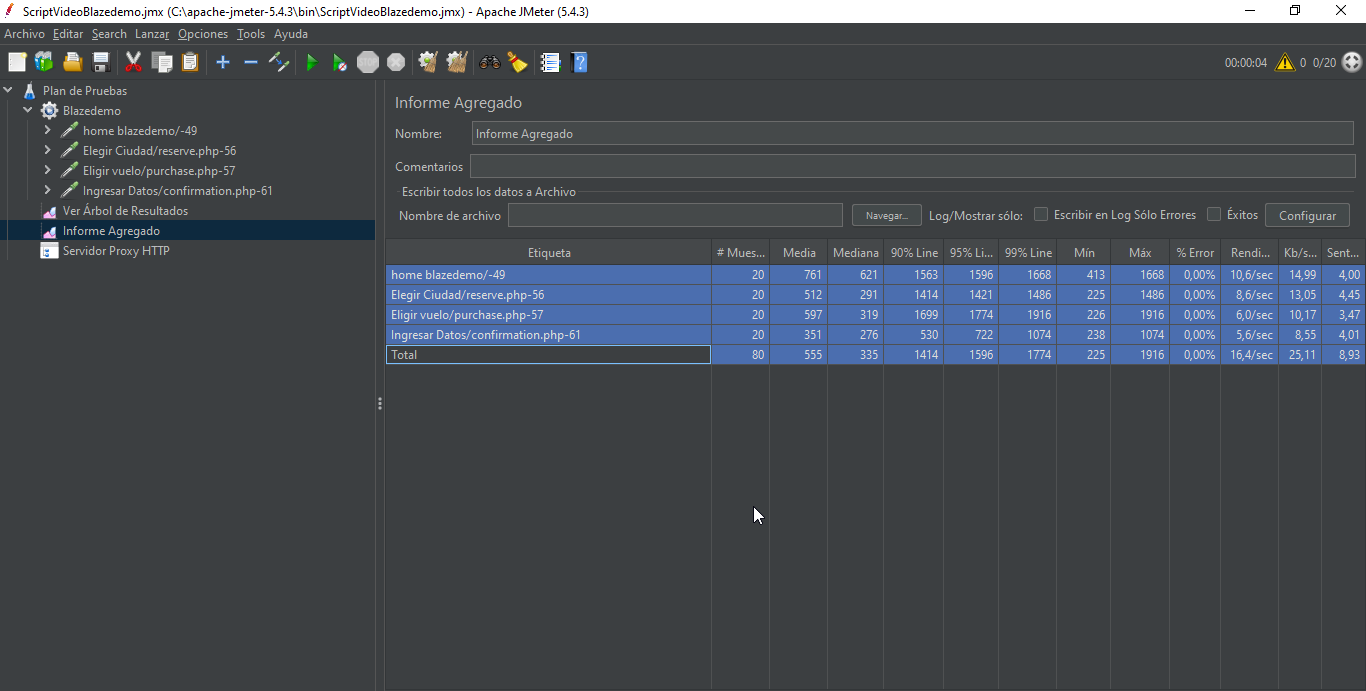
A continuación lo realizare para 20 usuarios y mostrare los valores de respuesta.



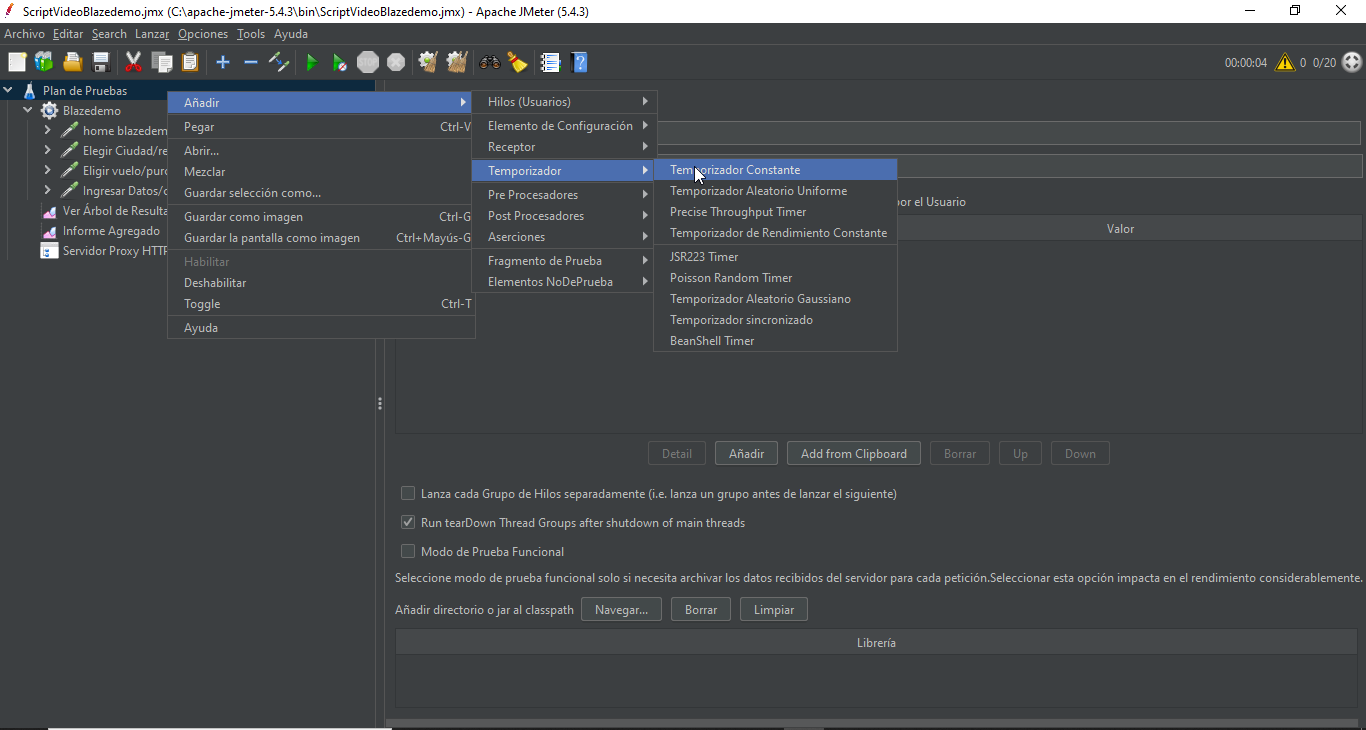
# Resultado de la Ejecución

Como se puede evidenciar los tiempos aumentaron debido a que realice un aumento en la concurrencia de usuarios por ejemplo en este caso el máximo de tiempo fue 1668 milisegundos con un mínimo de 0,413 milisegundos en tiempo de respuesta y no me genera ningún error debido a que está dentro de lo establecido. Y está asociado a cada una las peticiones que estoy ejecutando en el momento en la media evidenciamos el promedio de tiempo.

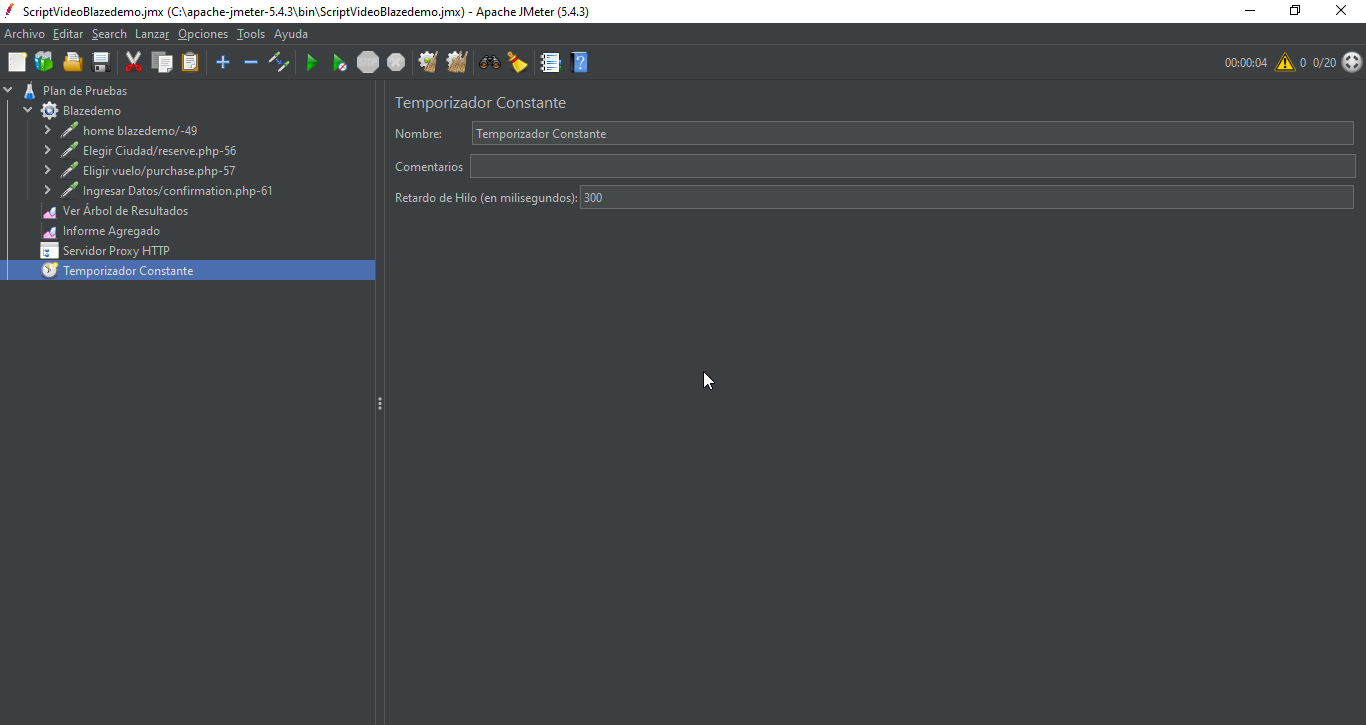




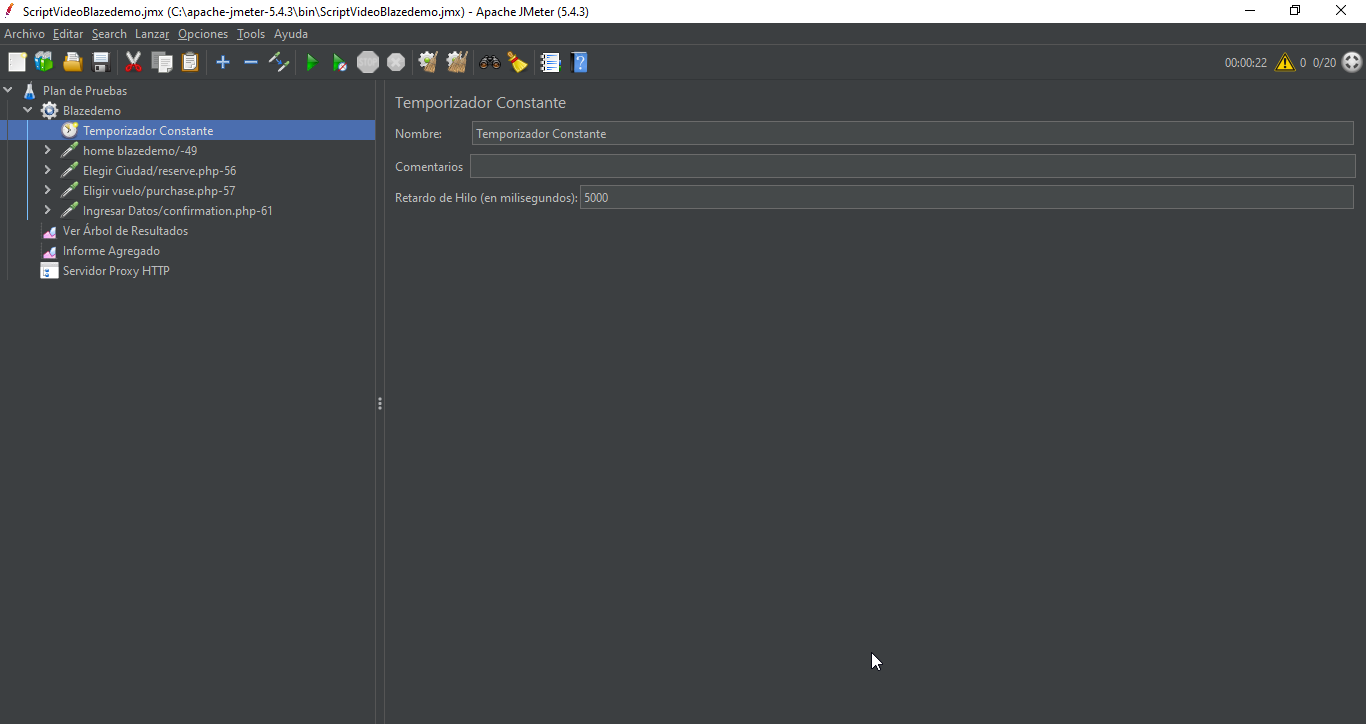
# Temporizadores o In Time que se implementaran en la pruebas

Con estos realizare la simulación de un comportamiento real cuando estoy interactuando con la aplicación de Blazedemo en la mediciones de tiempo el propósito es detener en un momento indicado la ejecución de la petición. Anexo evidencias de cómo se realiza la implementación de los temporizadores los cuales utilizare los dos más referenciados o implementados a nivel estándar: Temporizador Constante: Sera utilizado para agregar un tiempo de espera constante entre cada petición o si se prefiere a nivel general en el script 

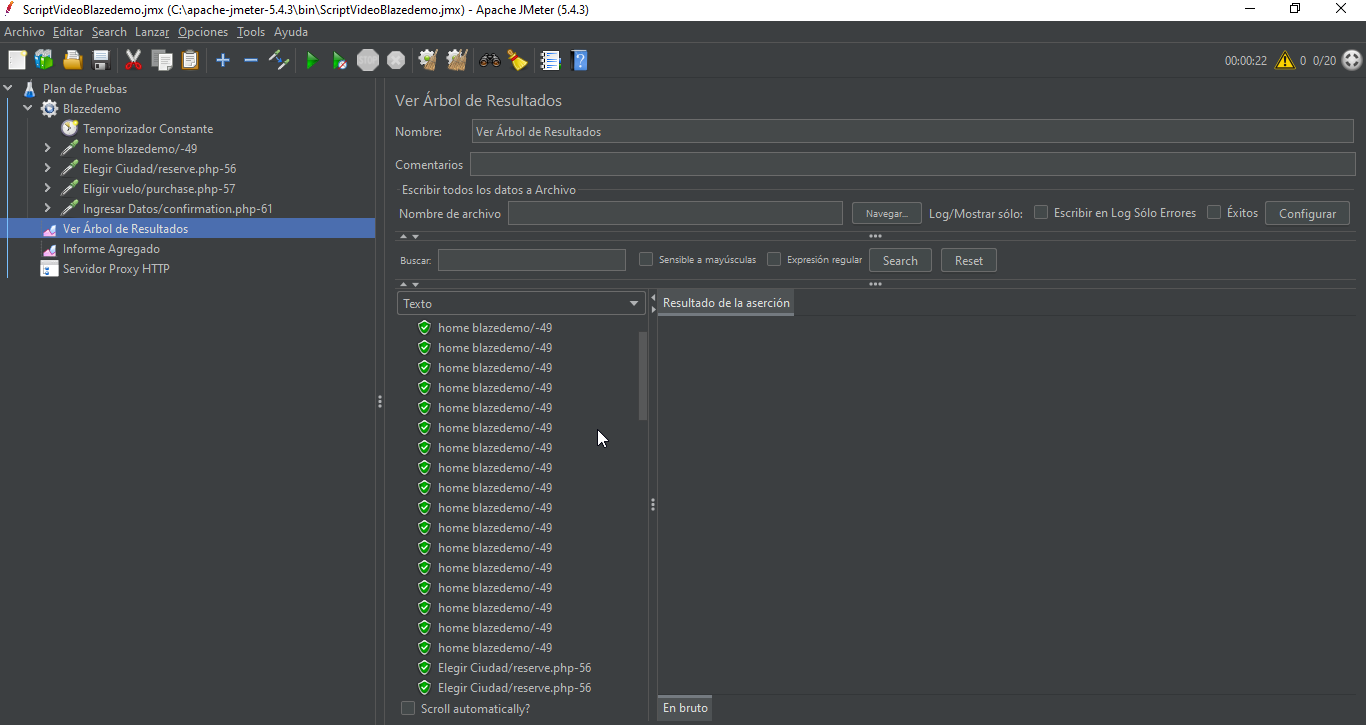
Como se puede observar en la imagen Tiene un campo para introducir el tiempo de espera que será constante para cada petición dependiendo del el lugar de la ubicación o requerimiento



Realizo la entrada de 5000 milisegundos lo cual serán 5segundos para al realizar la prueba sea visible la evidencia de la ejecución del temporizador lo ubicare en la parte superior d las peticiones para que tengan todas el mismo tiempo todas las peticiones.



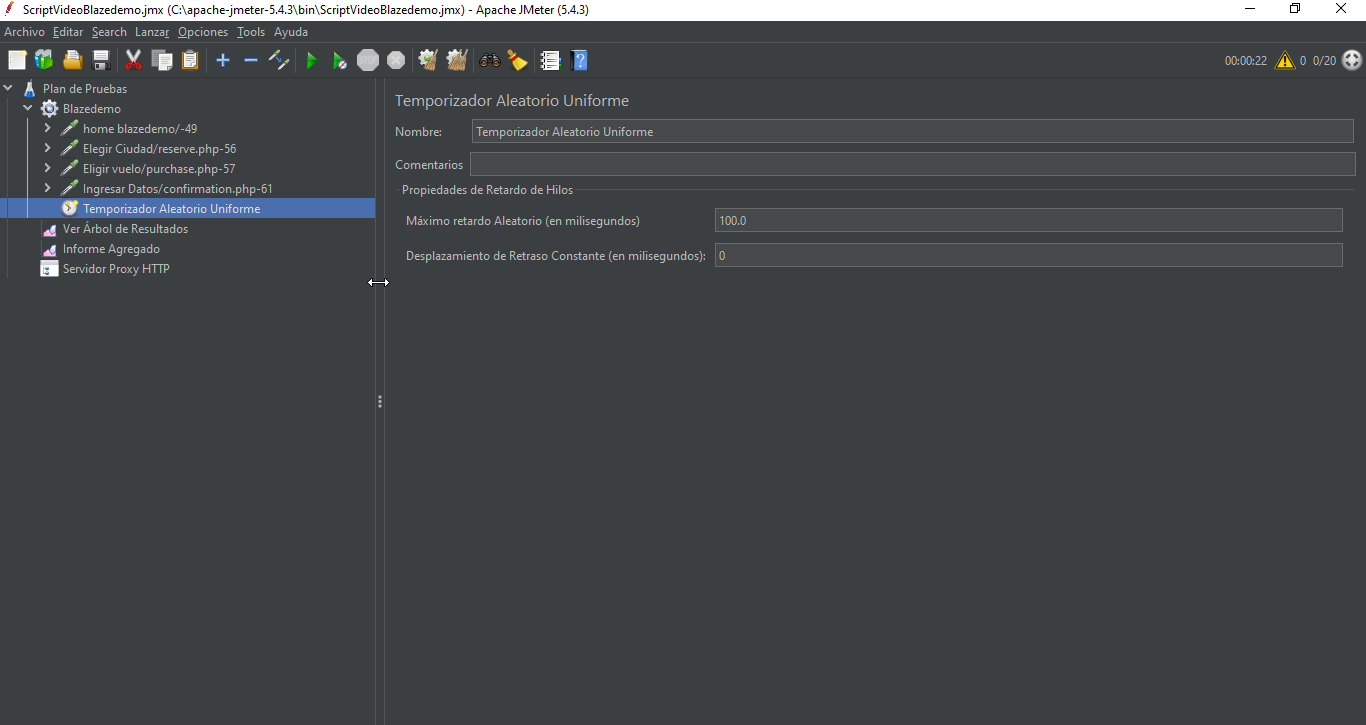
Como se puede evidenciar en la respuesta espero 5 segundos por cada petición entre 20 usuarios y va realizando la ejecución de cada escenario o petición implementada



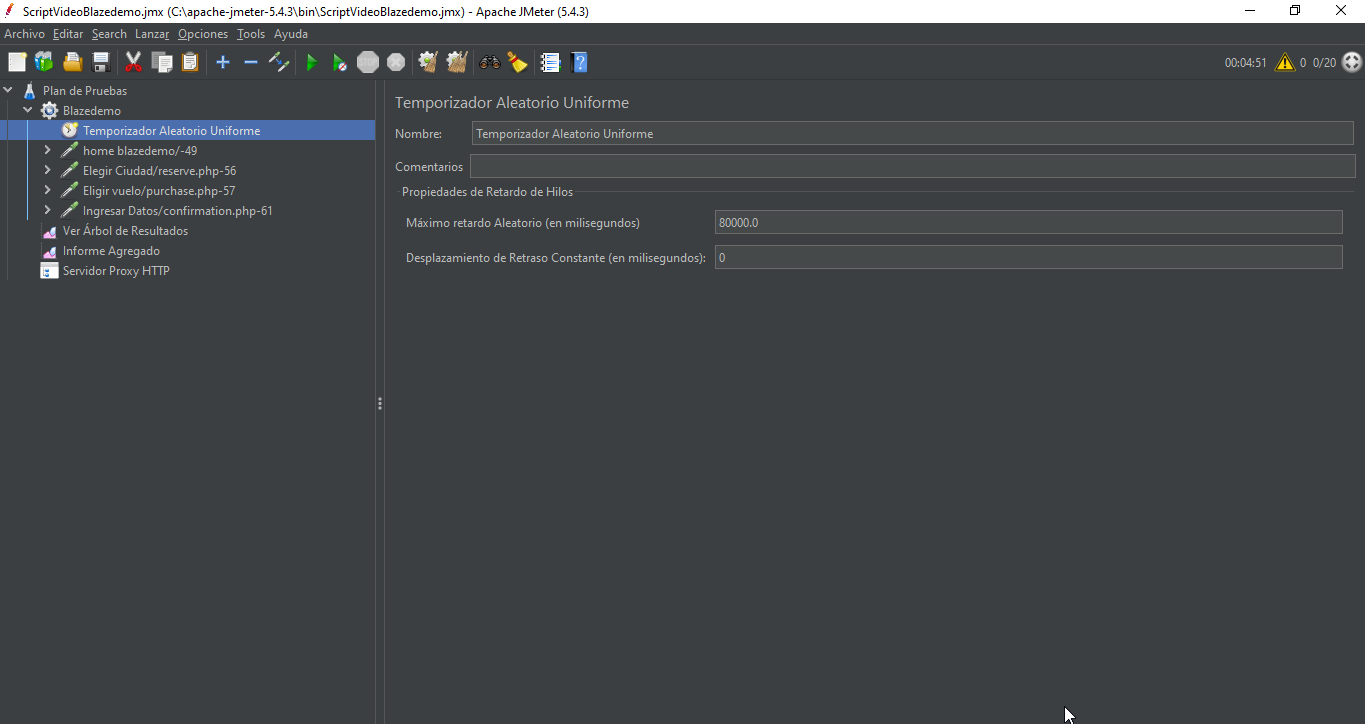
Temporizador Aleatorio Uniforme: Sera utilizado para pausar cada solicitud de cada petición durante un tiempo aleatorio teniendo en cuenta cada intervalo de tiempo es la misma probabilidad para cada una de las peticiones.

De esta forma se agrega el temporizador:



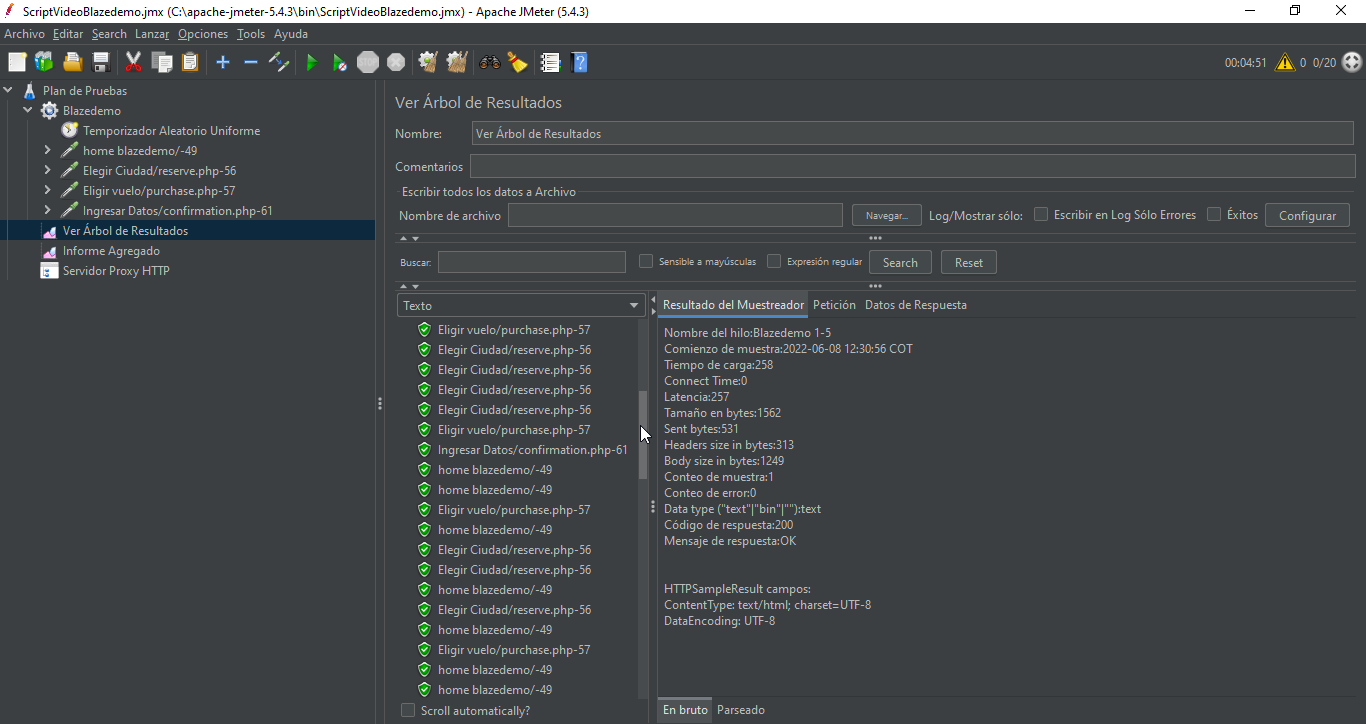


Realizo la entrada de con 8000.0 milisegundos para hacerlo visible a la ejecución el temporizador lo ubicare en la parte superior para que me tome todas de las peticiones



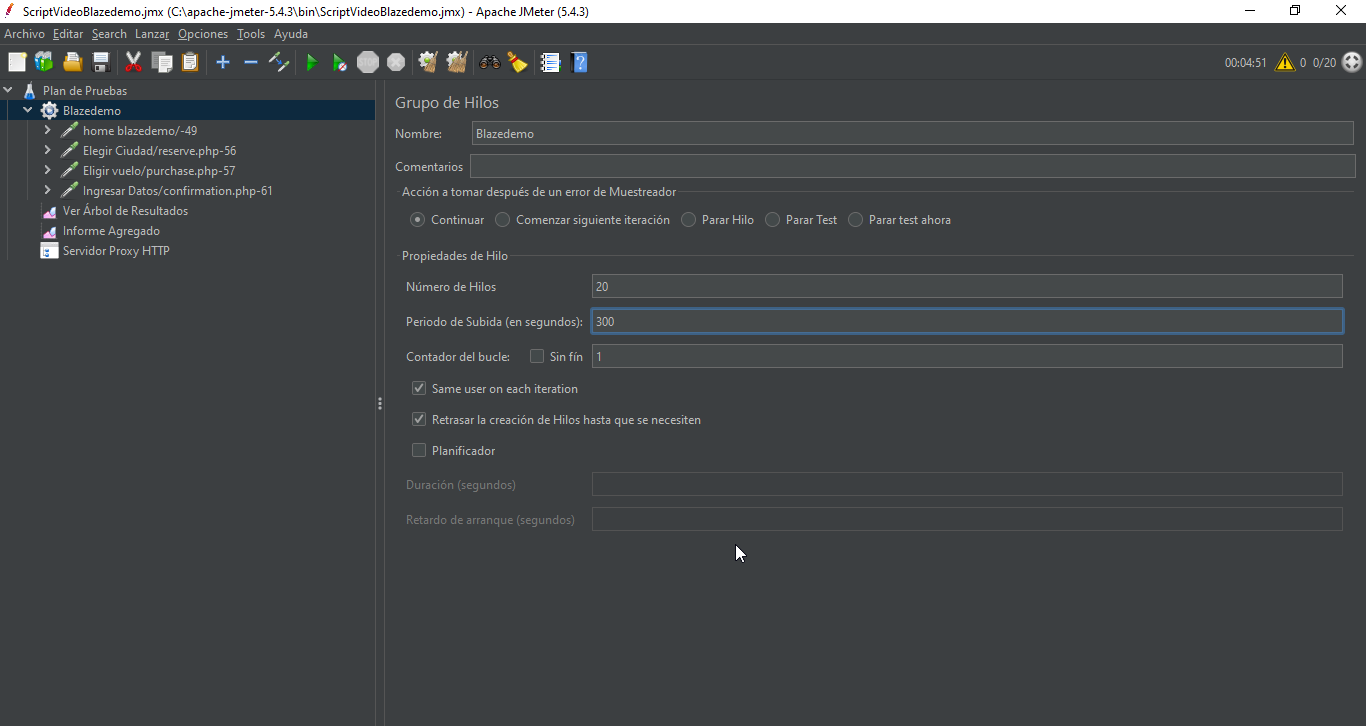
Al realizar esta prueba el tiempo de respuesta con este temporizador y con la cantidad de tiempo implementada la respuesta entre cada petición inicio con una diferencia de 2 segundos después paso a 6 segundos se mantuvo por 6 peticiones más después paso a 8 segundo durante 4 peticiones más y finalizo con una diferencia de 18 segundos en la última petición realizado con 20 usurios de concurrencia lo cual tuvo una duración de 4 minutos 51 segundos.

Eato quiere decir que hay usuarios q ingresan más rápido q otros.

.

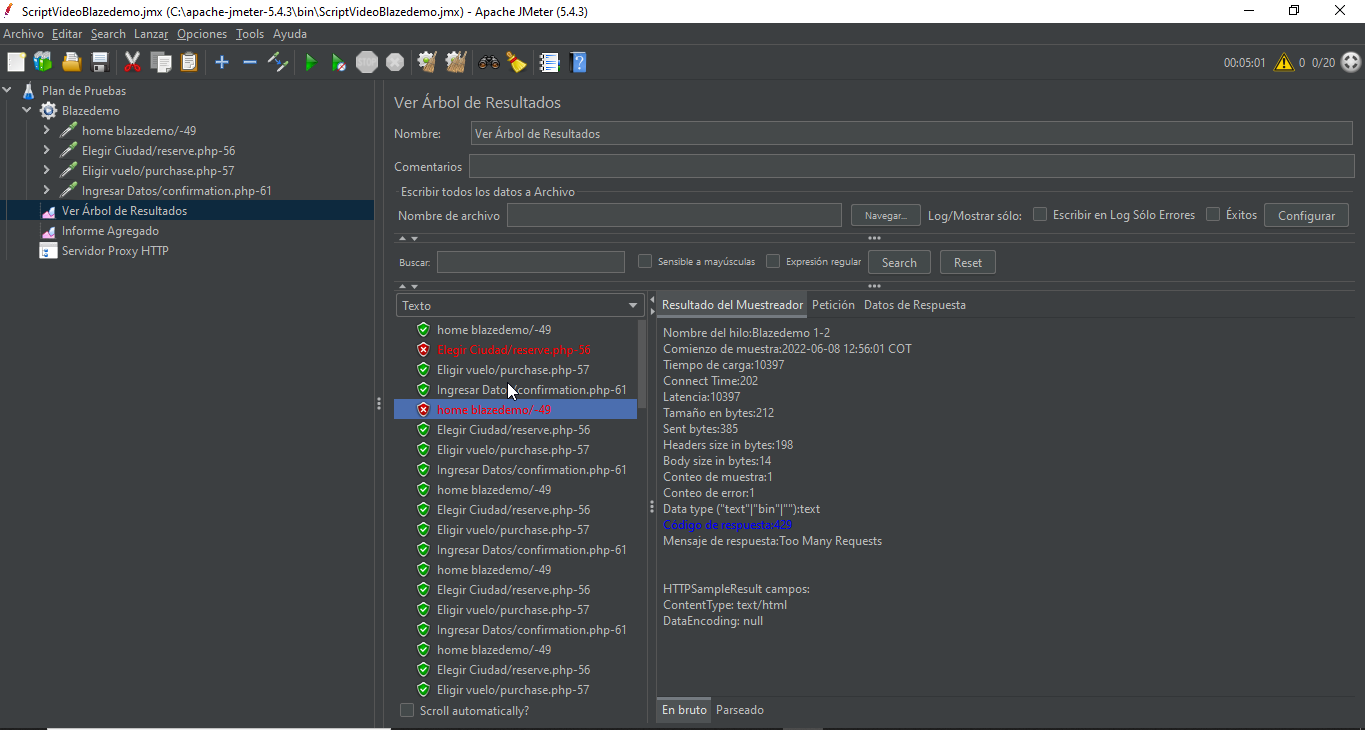
# A continuación verificare cómo se comporta el sistema donde se conecten hasta 20 usuarios en un lapso de 5 minutos, estos usuarios deben realizar pruebas a todo un flujo de compra de manera concurrente en JMETER

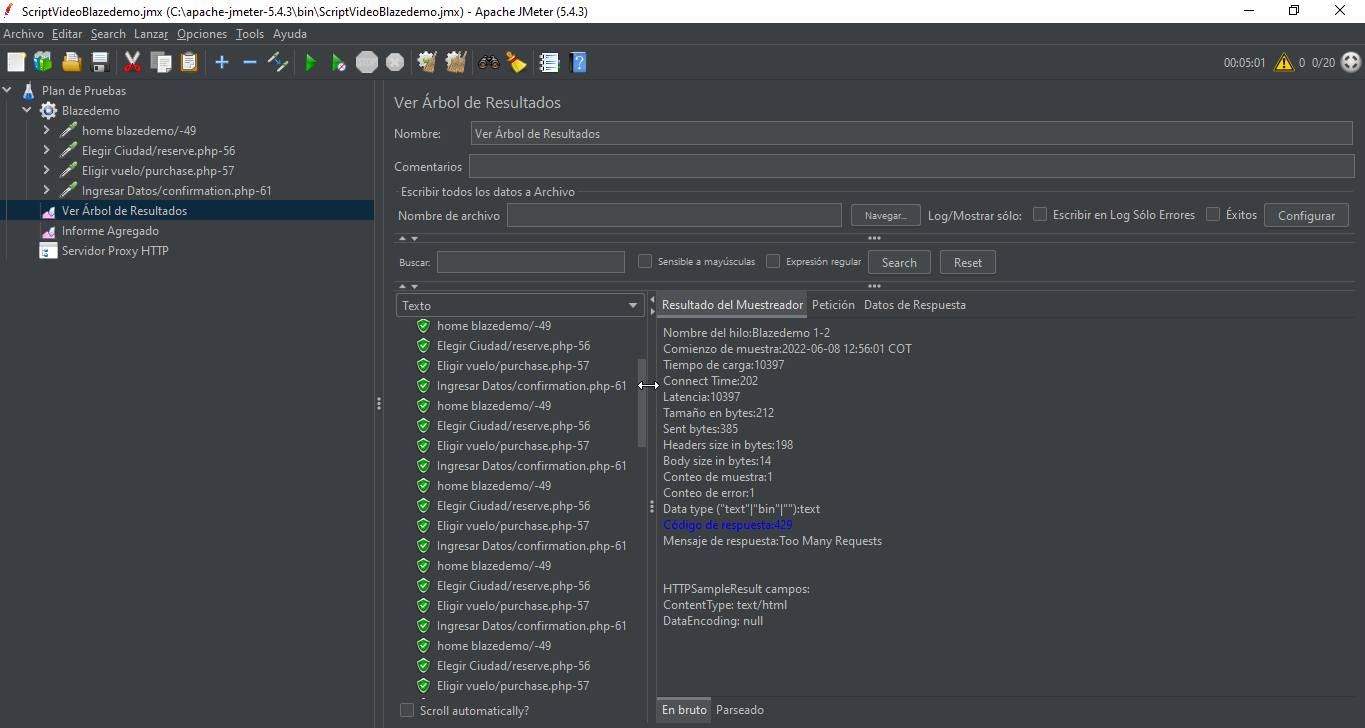
En este segmento implementare el tiempo en la raíz que será con una duración de 300 segundos que equivale a 5 minutos como lo requiere el documento para la prueba con 20 usuarios concurrentes :



Y procederé a la ejecución:

Como se observa a continuación en la captura de pantalla la primera petición inicia con una respuesta exitosa pero en el segundo flujo se presenta un error en el primer usuario en la petición donde elige la ciudad error **429 Too Many Requests** porque el usuario ha enviado demasiadas peticiones en un tiempo determinado pero en la siguiente petición del flujo se restablece y continúan con los flujos donde nuevamente vuelve a generar la caída de la petición en el inicio de entrada del segundo usuario con el mismo código de respuesta anteriormente mencionado y vuelve y retoma las peticiones de los siguientes 18 usuarios en la concurrencia y salen satisfactorios todos.





Con la duración visualizada en pantalla requerida por el documento enviado para el reto de 5 minutos el cual sobrepaso por un segundo de más.

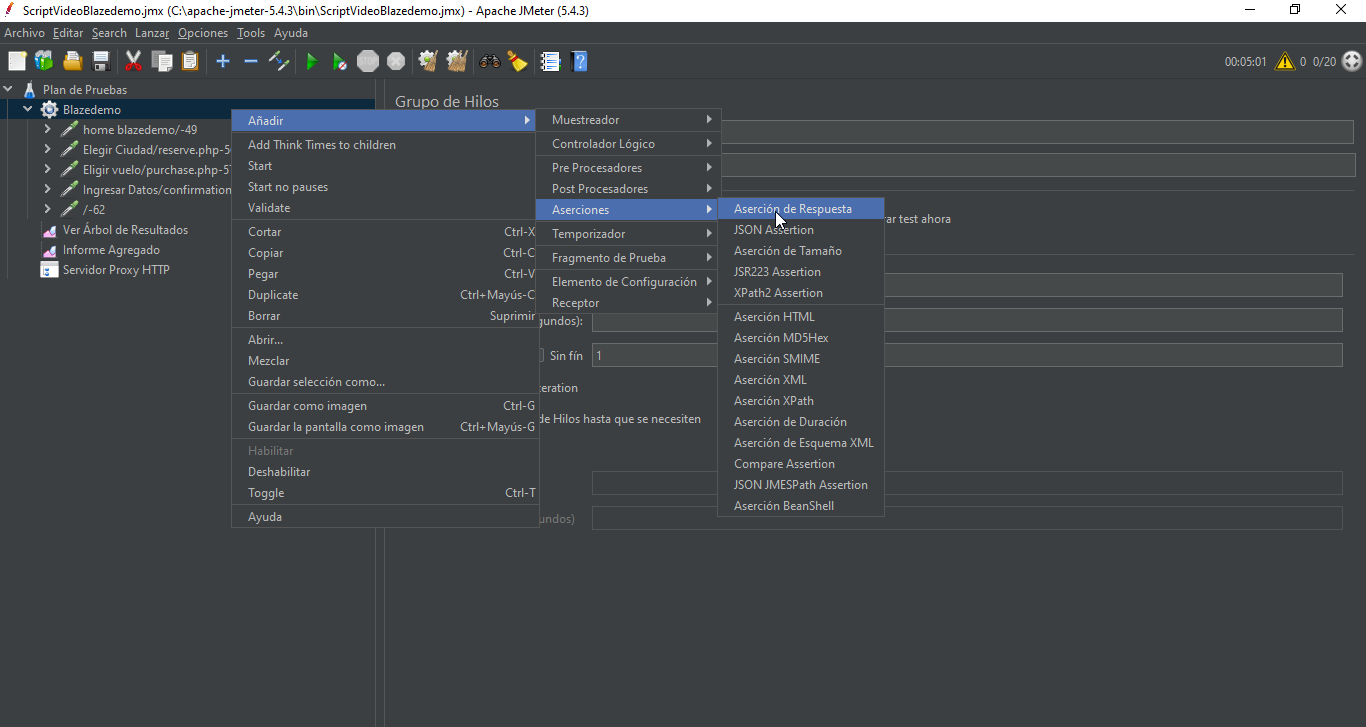
# Aserciones JMETER

Las cuales voy a utilizar para hacer las validaciones e implementarlas en el script para saber si cuando ejecuto la petición la respuesta que se envía desde el servidor es correcta o es invalida.

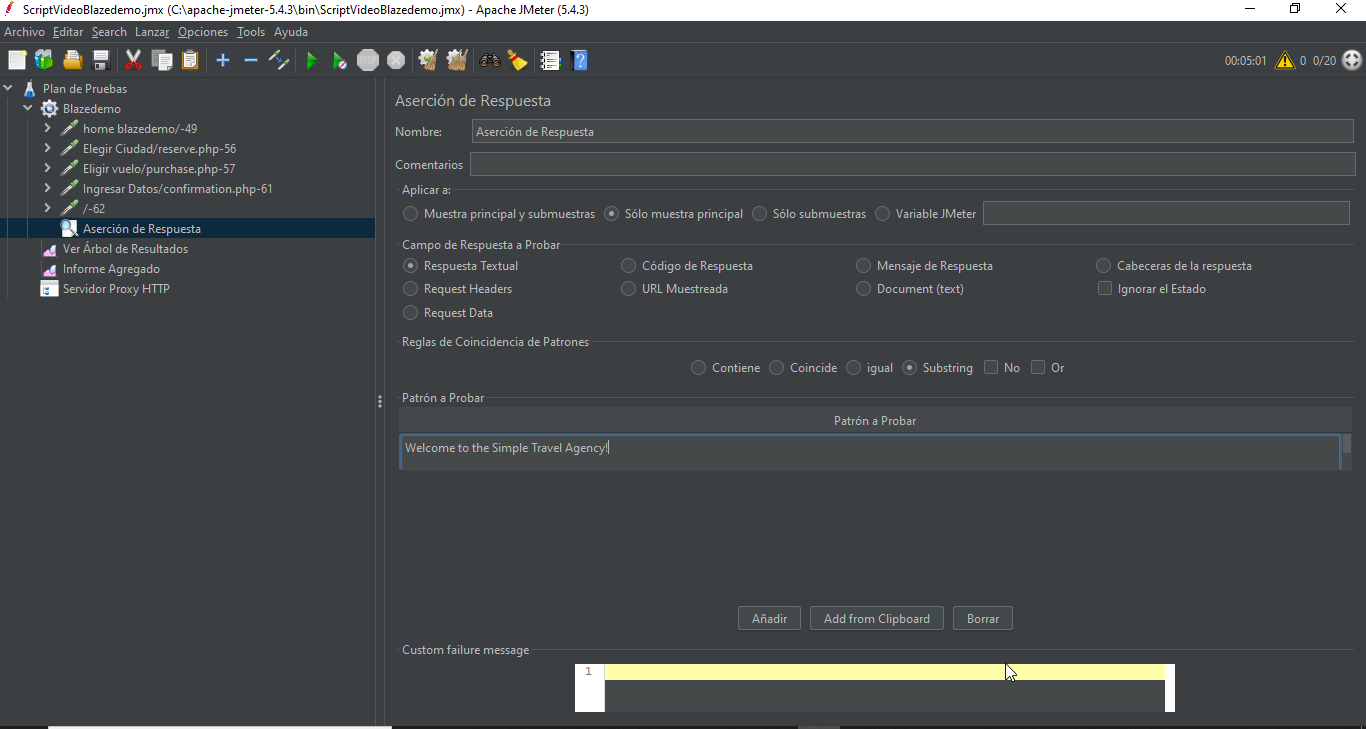
Utilizare las siguientes en este reto:

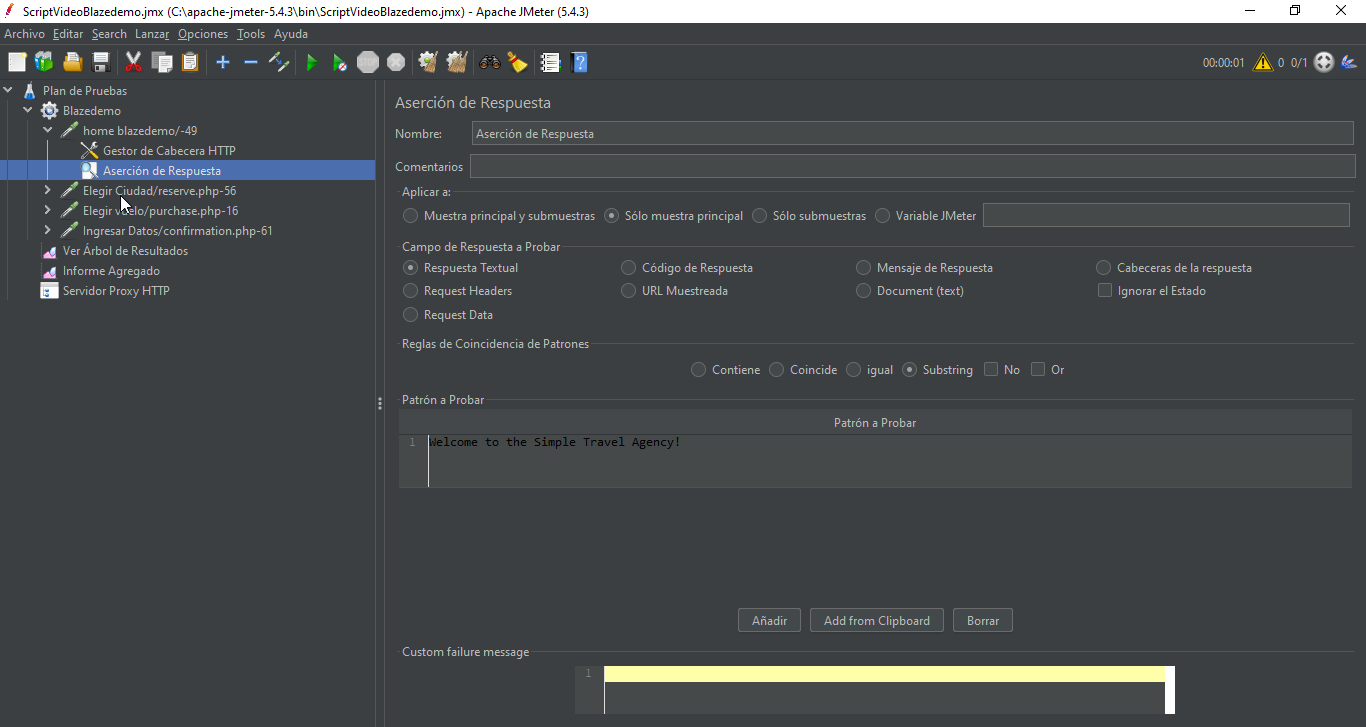
Aserción de Respuesta: Con esta aserción voy a comprobar la respuesta que viene desde el servidor con respuesta textual.

Implementación de la aserción:

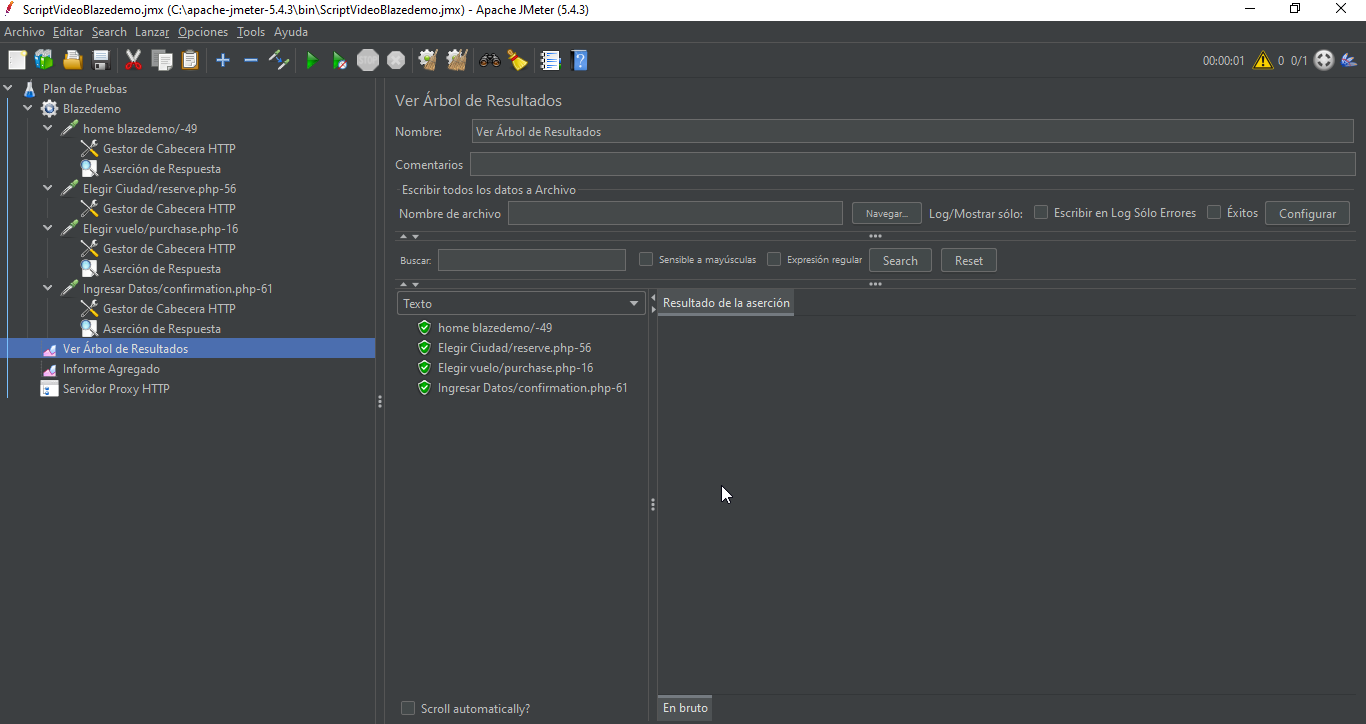


Configuración de la aserción en la petición que voy a validar en mi caso será en la primera como se muestra en la captura de pantalla



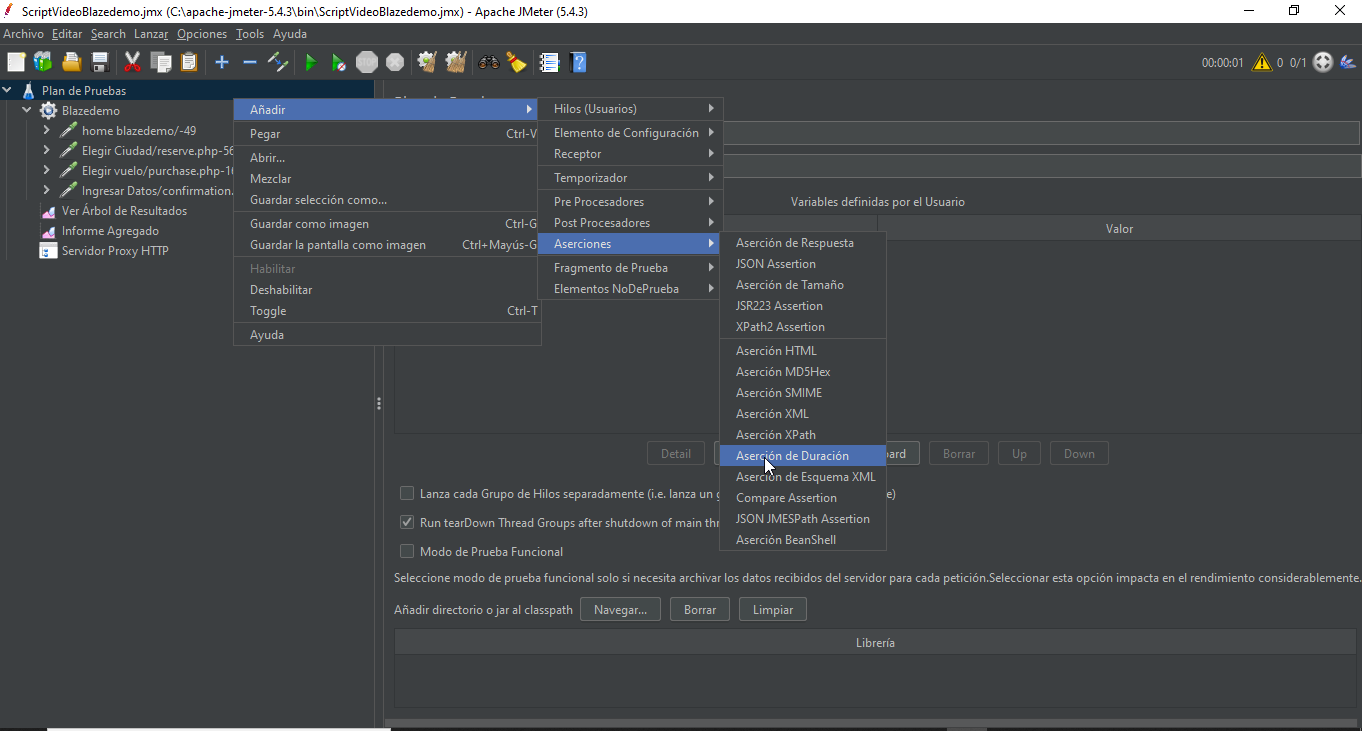


Al ejecutar las peticiones podemos evidenciar que fue exitosa la validación de la aserción ya que nos generó un código de respuesta 200 y en verde realizado a la interface de home captura el texto de respuesta que envía el servidor y asi repetitivamente se le realizo la misma implementación a cada una de las peticiones para poder realizar las validaciones de cada uno de los flujos exepto a la de Elegir Ciudades que es la misma de Home.



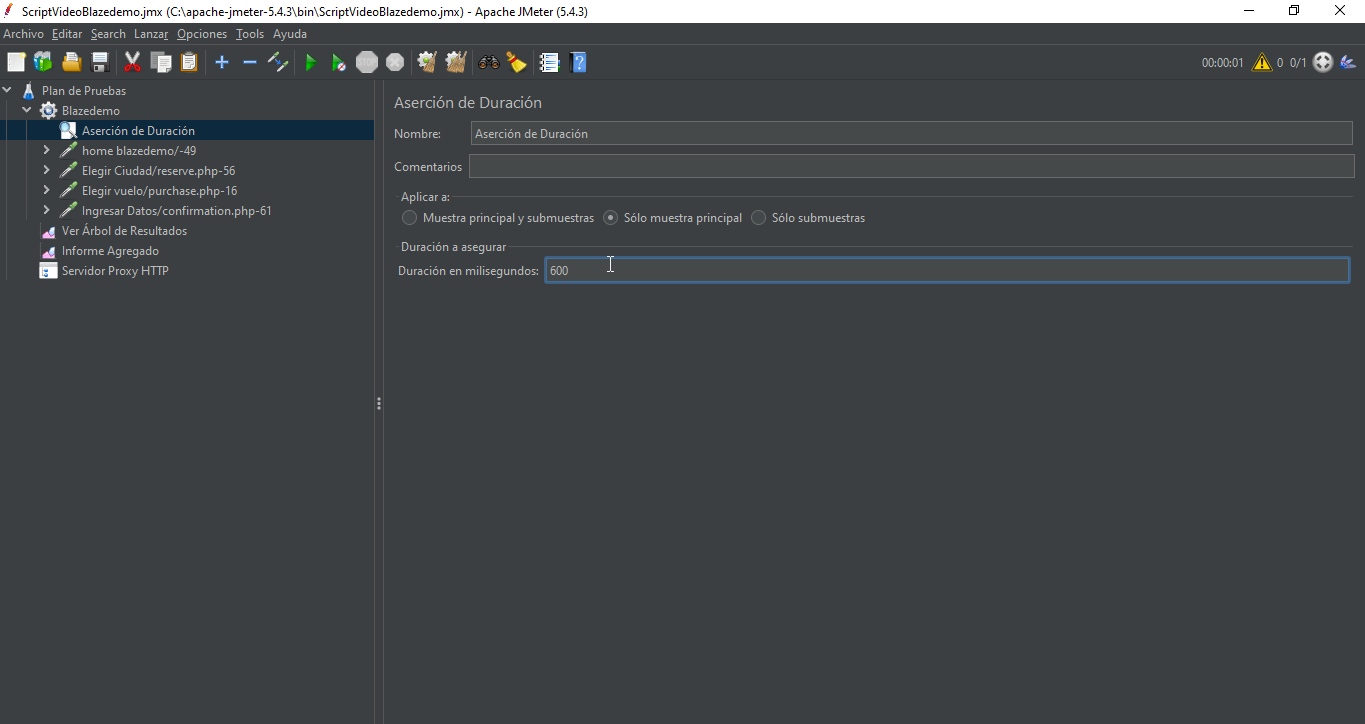
Aserción de Duración:

Como primer paso realizare la implementación:

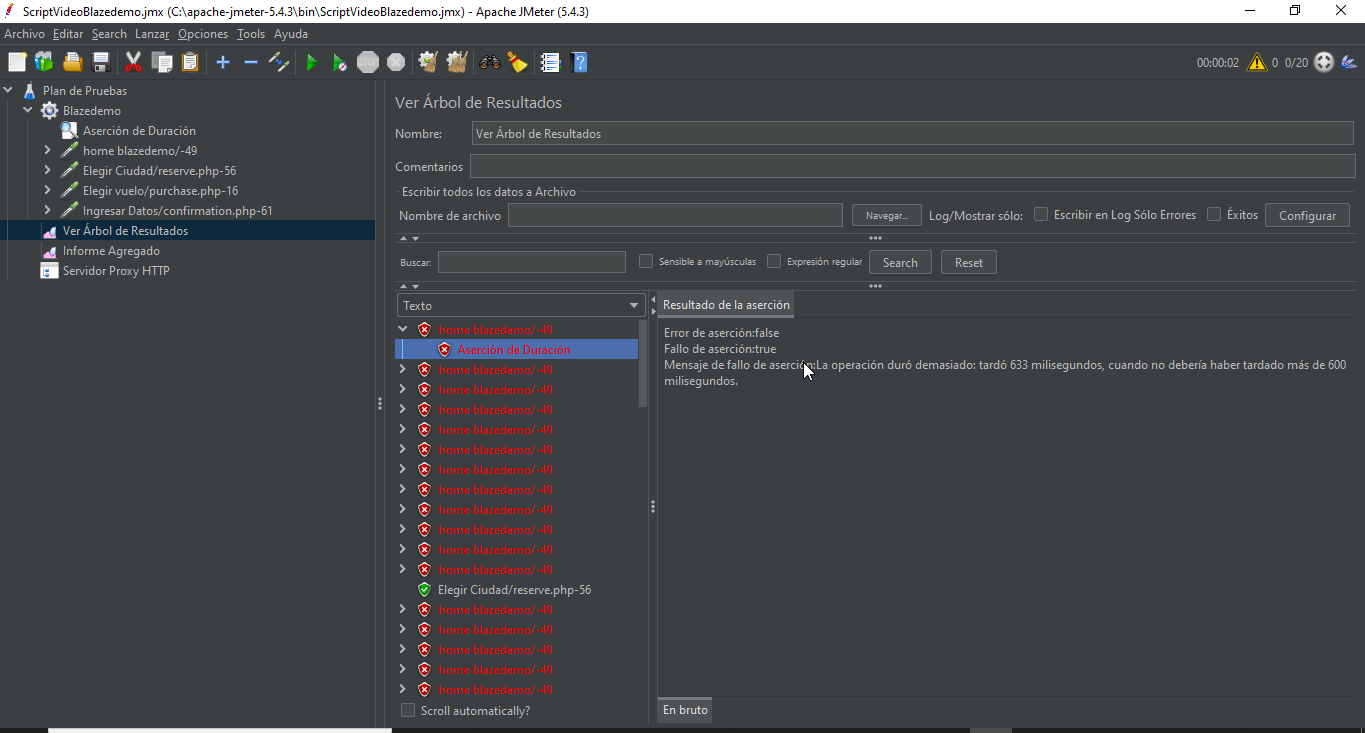




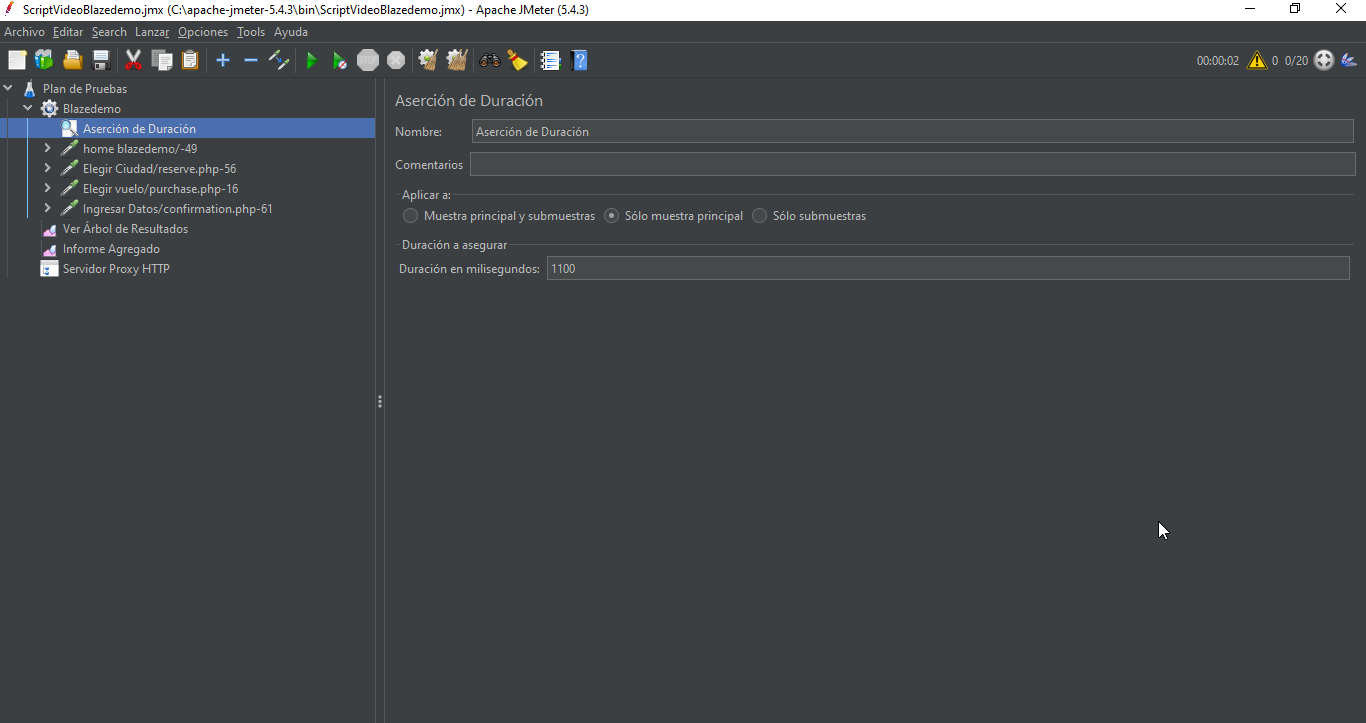
Esta aserción permitirá ver cuáles de la peticiones que implemente dura más de 600 ms, al ejecutarse el paso a seguir es implementar el tiempo en mi caso 600 ms al ejecutarlo y validar la respuesta de la aserción colocándola al nivel de todas las peticiones me dará la respuesta de la duración en todas las peticiones con 20 usuarios concurrentes en simultaneo.

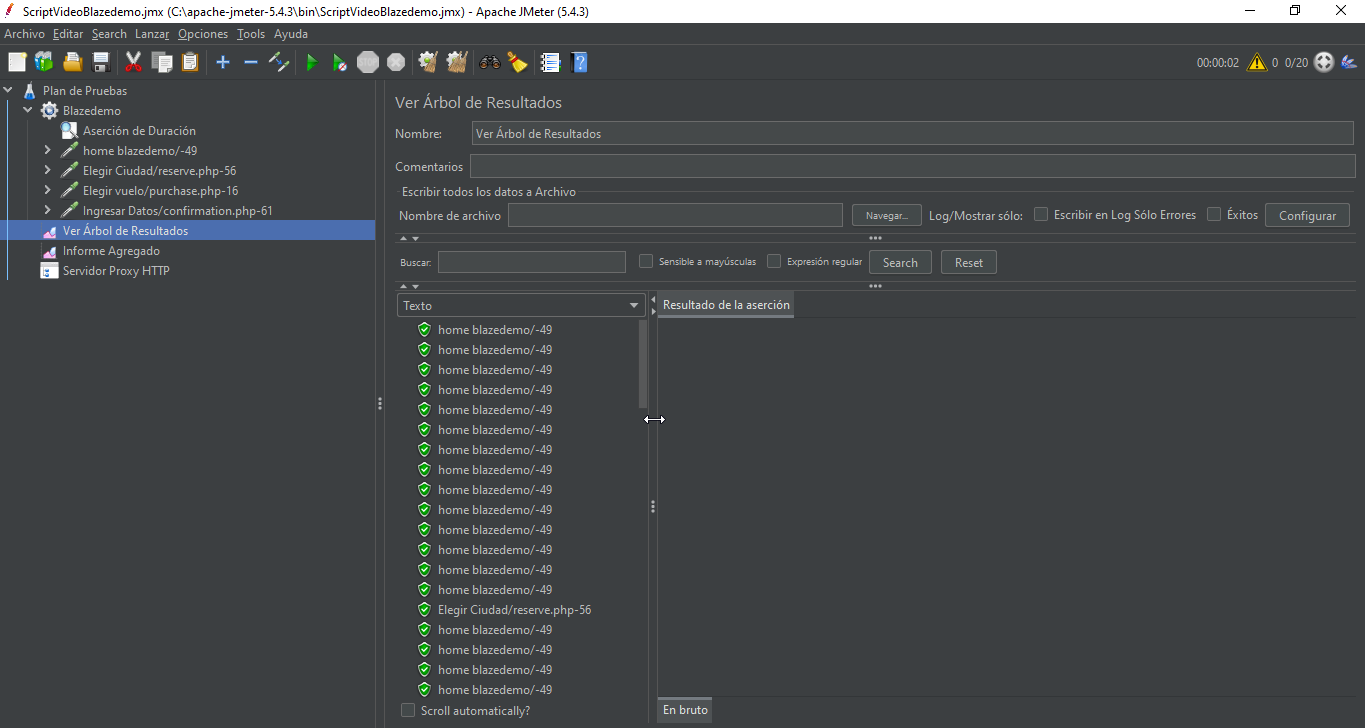


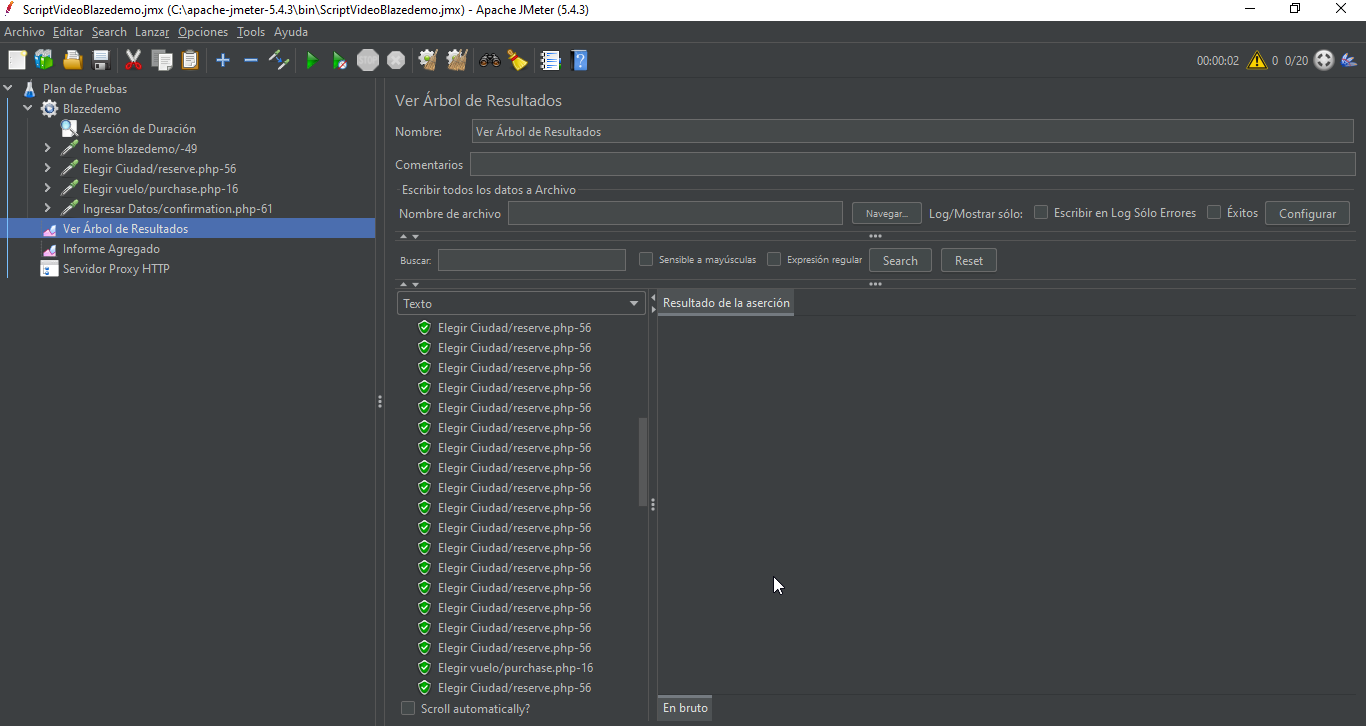
Como se puede evidenciar en la captura de pantalla todos los inicios de Home de los 20 usuarios en concurrencia fallaron ya que sobrepasaron el tiempo estipulado de respuesta requerido para la prueba

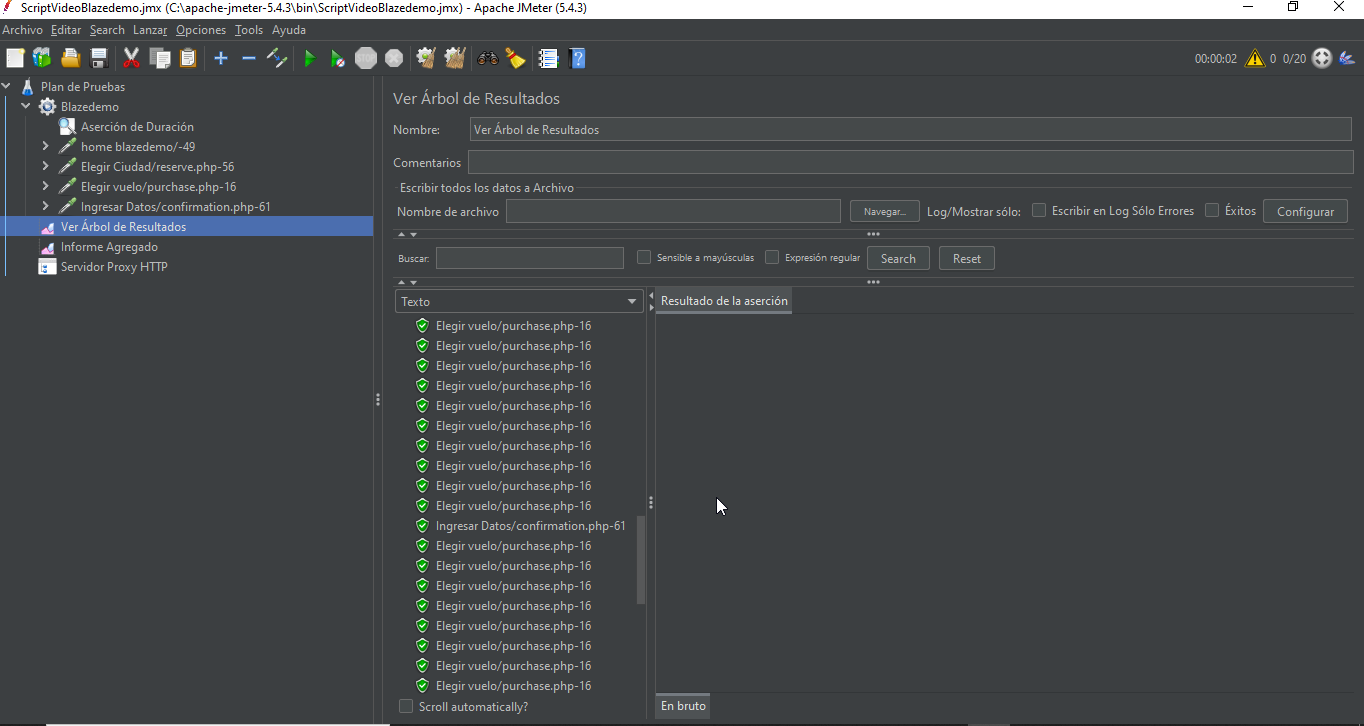


Solo algunos muy pocos en el siguiente flujo de peticiones de Eligir Ciudad lograron tener el tiempo de respuesta que se requiere para la prueba y únicamente 14 peticiones de Elegir vuelo lograron el tiempo requerido o menos y únicamente 17 peticiones de Ingresar Datos lograron el tiempo requerido o menos del requerido por la prueba.Se realiza una prueba aumentando el tiempo de duración a 1100 ms para ver un flujo exitoso con 20 usuarios concurrentes:

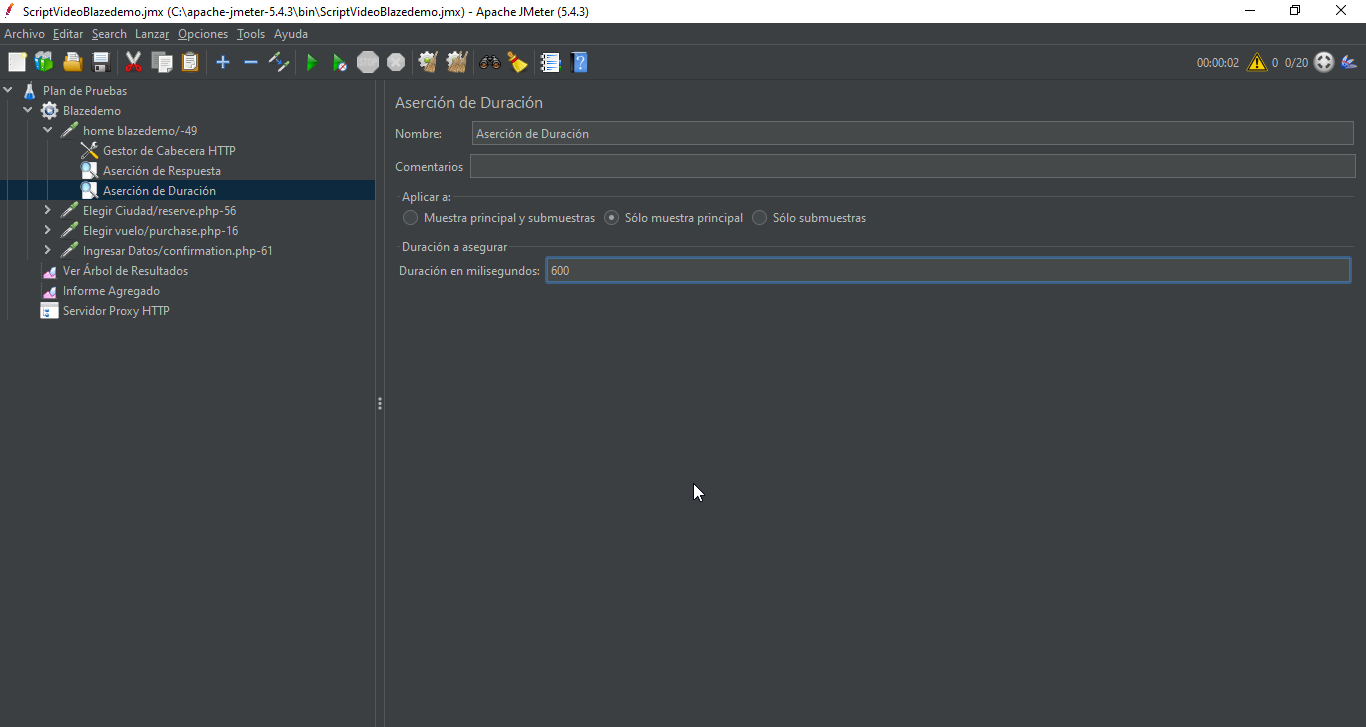




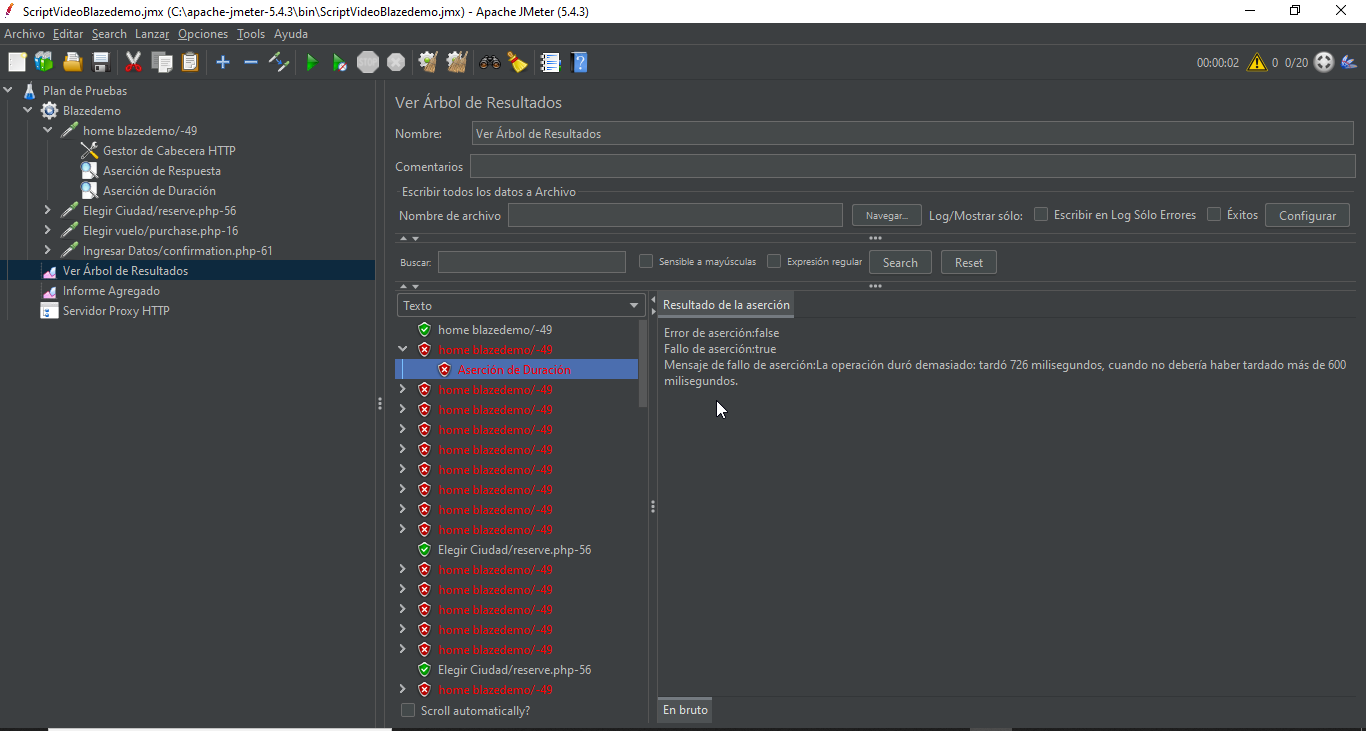




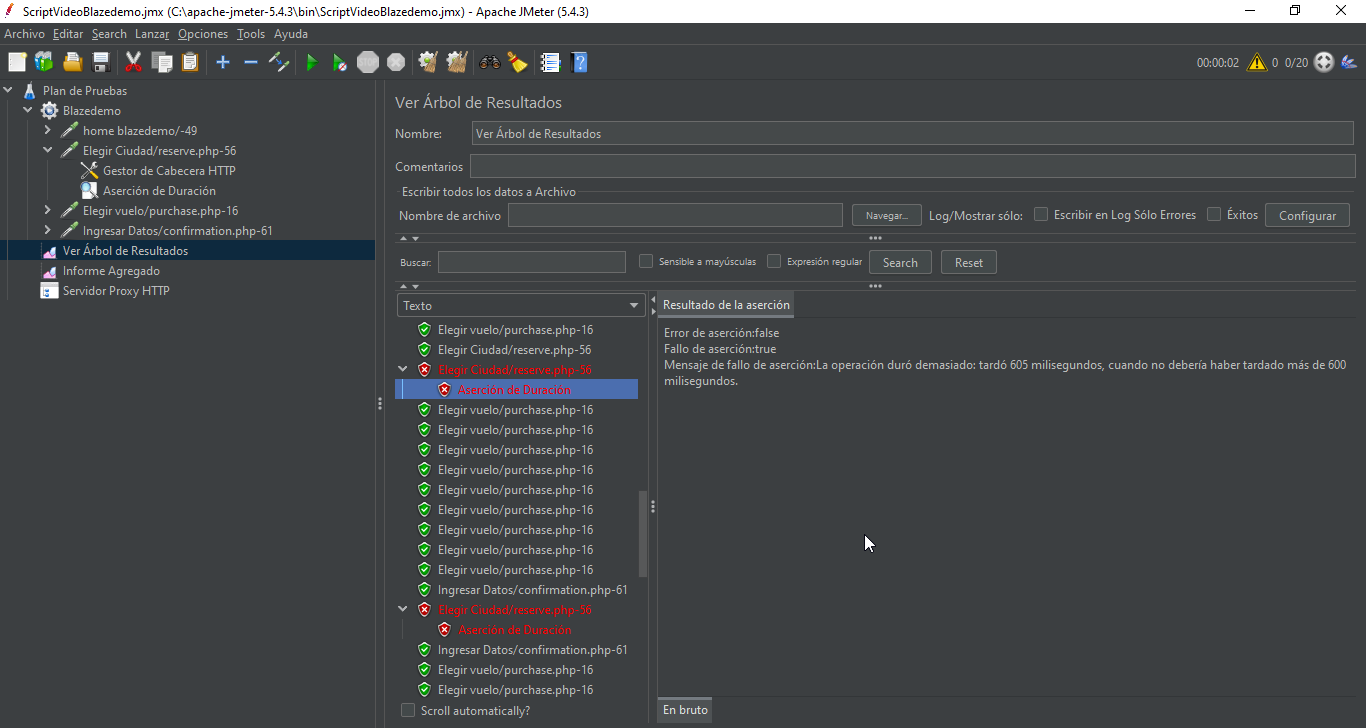
Ahora lo realizare únicamente con cada una de las peticiones implementando el tiempo de duración en cada petición por individual.



Como se evidencia en la captura fallo el tiempo de duración nuevamente en esta petición ya que sobrepaso los 600 ms.

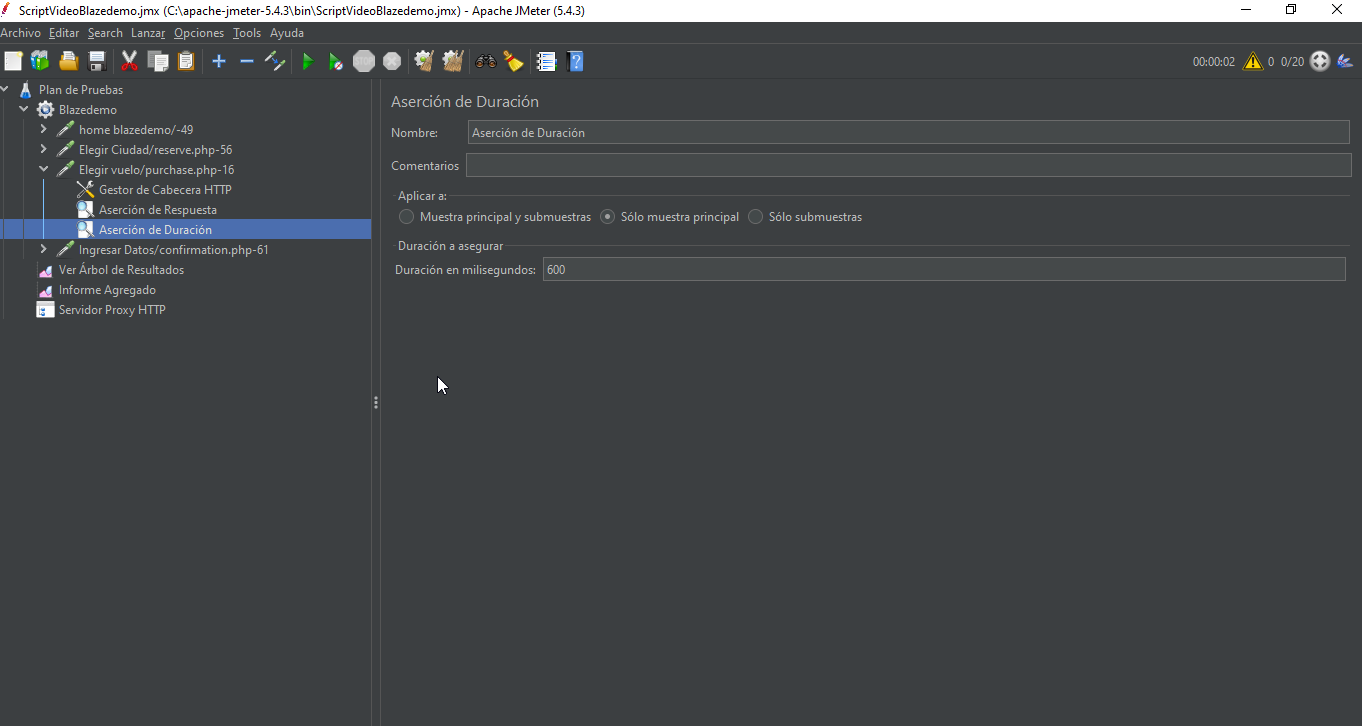


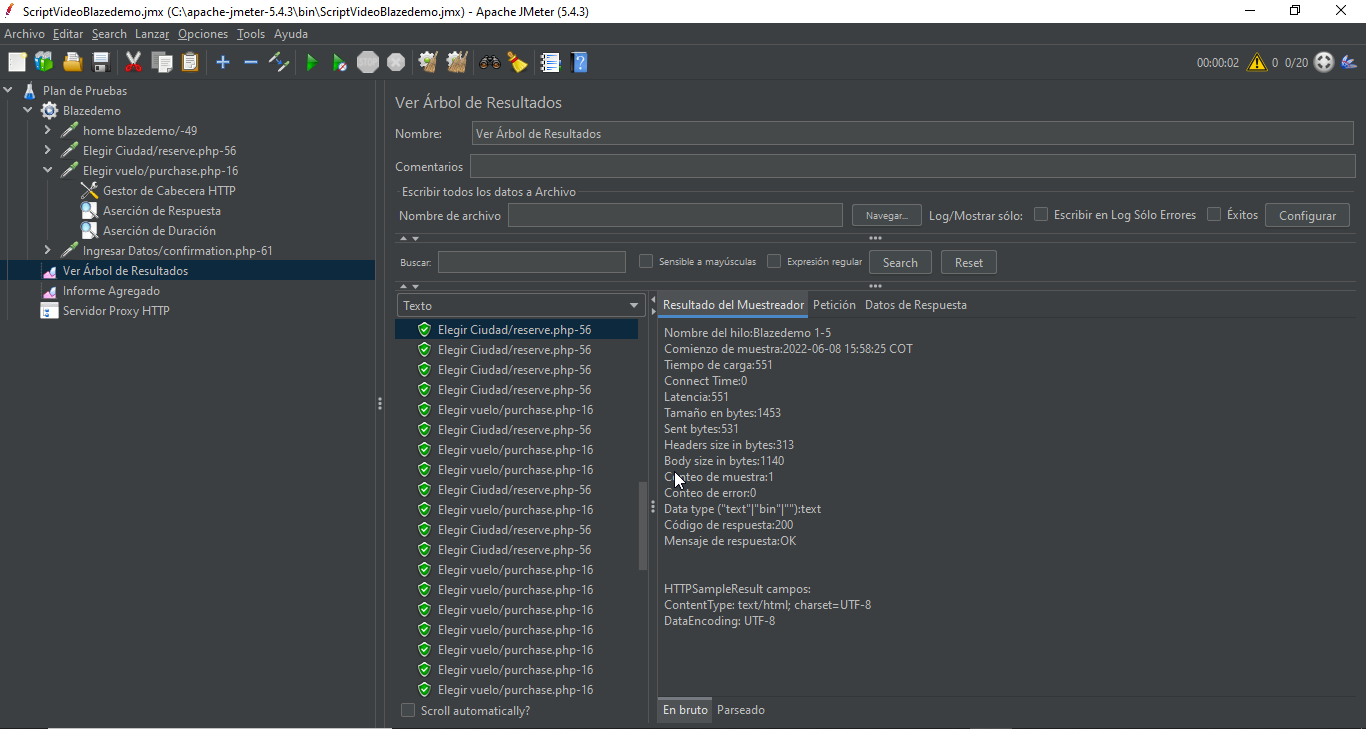
En esta captura de la segunda petición únicamente fallaron dos peticiones en el tiempo de duración ya que sobrepasaron nuevamente el tiempo establecido por 5 ms de más al responder con 20 usuarios en concurrencia.



Tercera petición:

Como se puede observar en la captura todas las peticiones fueron exitosas ya que no sobrepasaron los 600 ms estipuladas en la aserción.





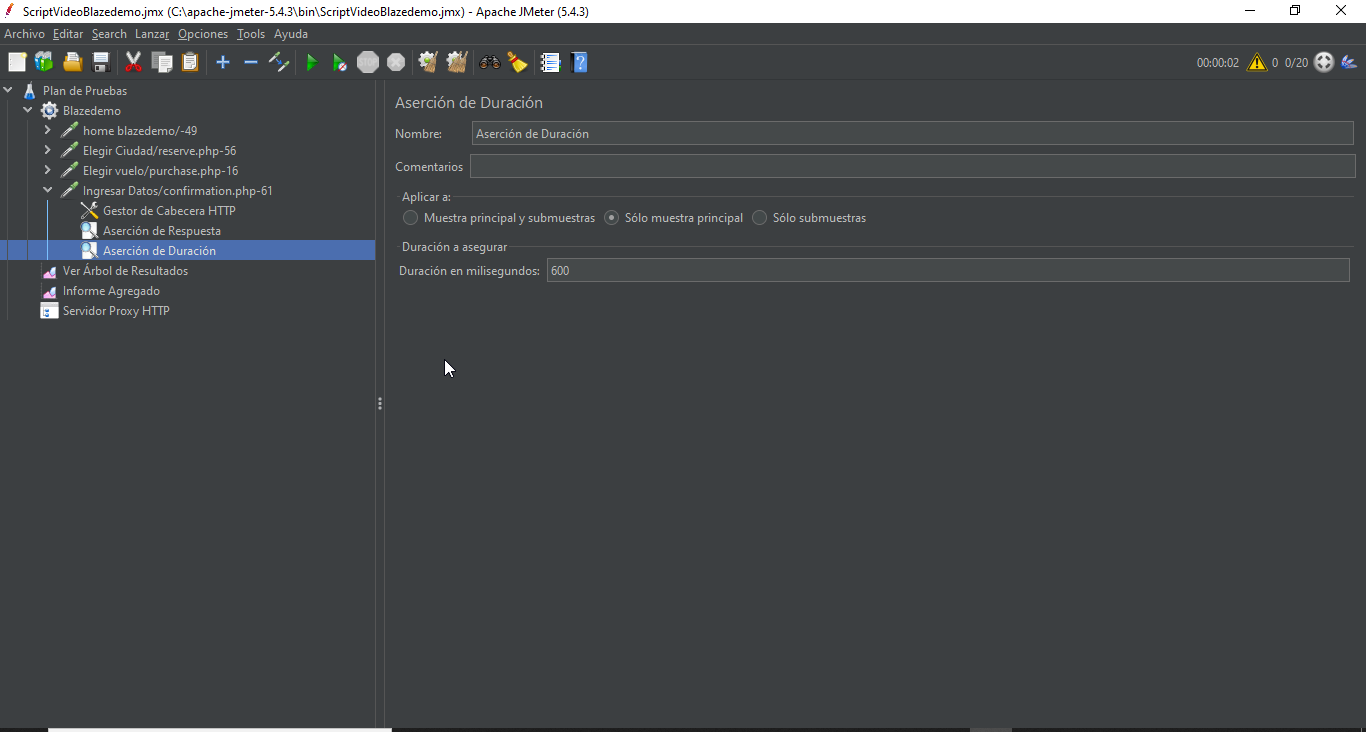
Cuarta y última petición en el flujo:

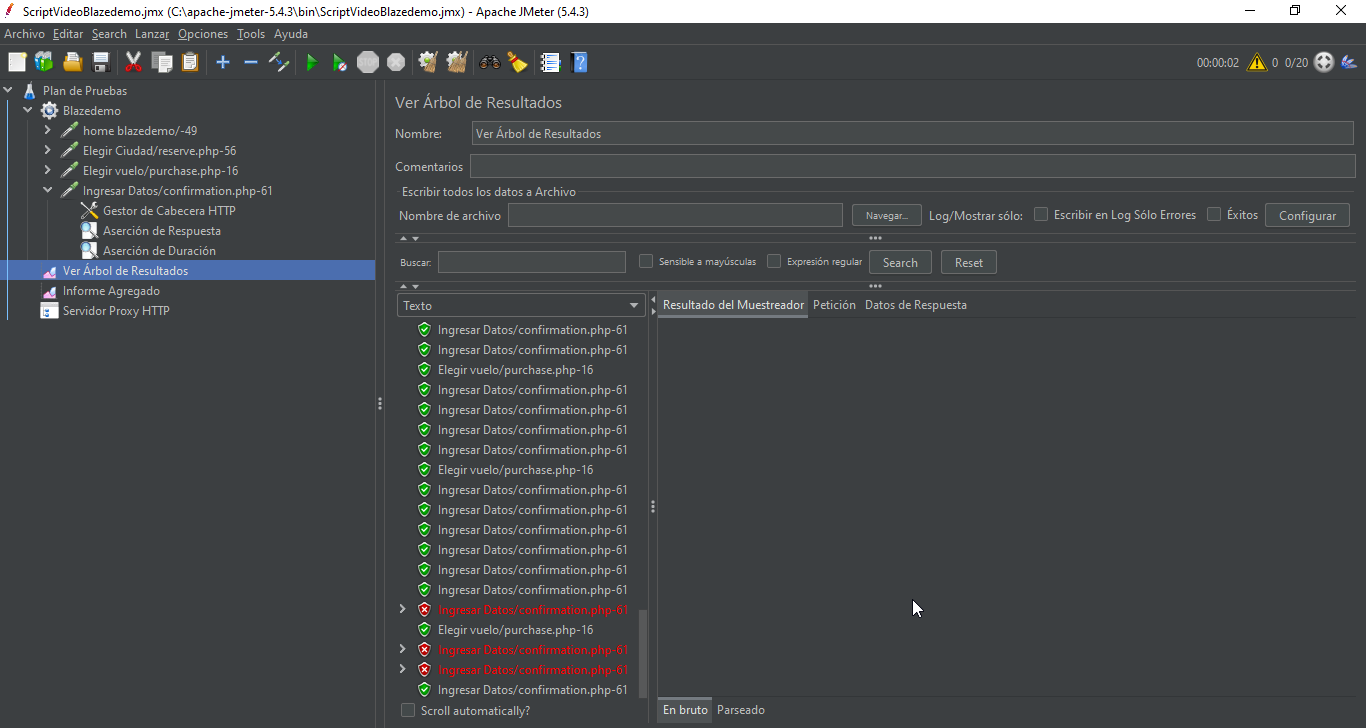
En esta captura de la cuarta y última petición únicamente fallaron tres peticiones en el tiempo de duración de respuesta ya que sobrepasaron nuevamente el tiempo establecido con 20 usuarios en concurrencia.

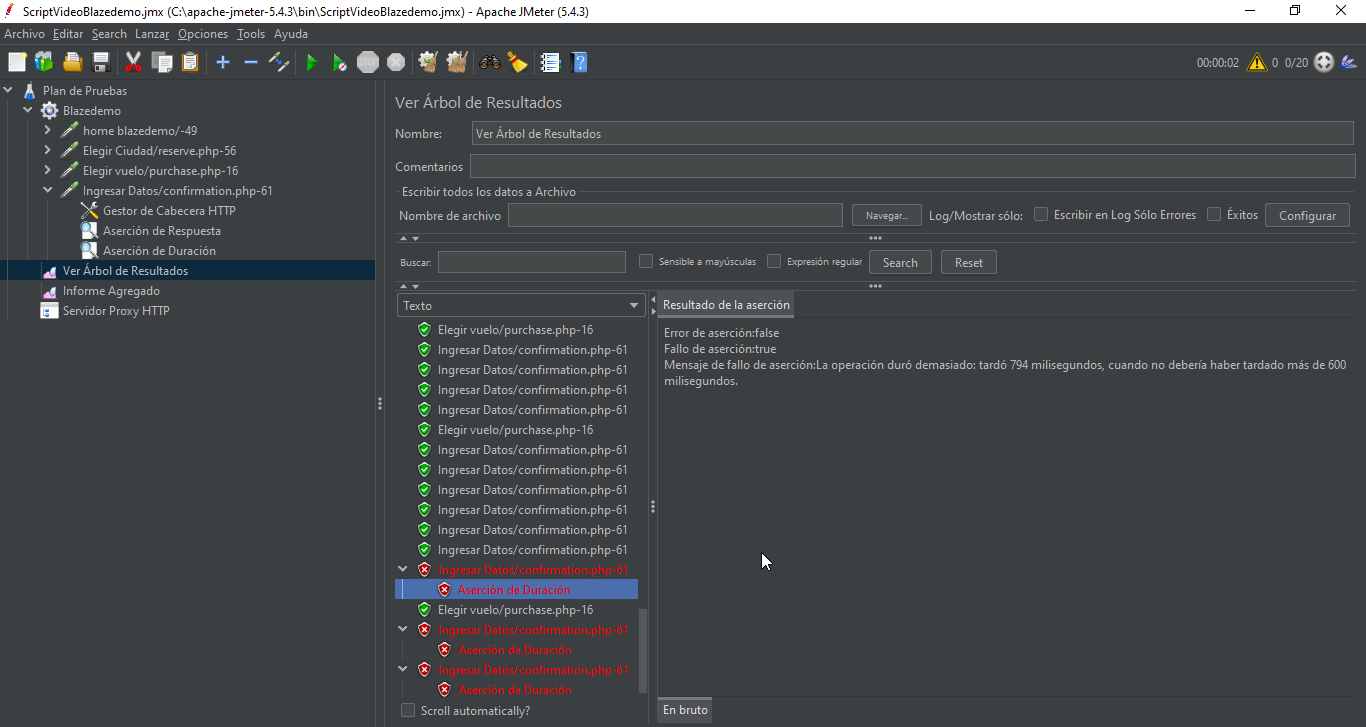
La primera tardo794 ms y lo esperado son 600 ms

La segunda tardo 858 ms y lo esperado son 600 ms

La tercera tardo 929 ms y lo esperado son 600 ms

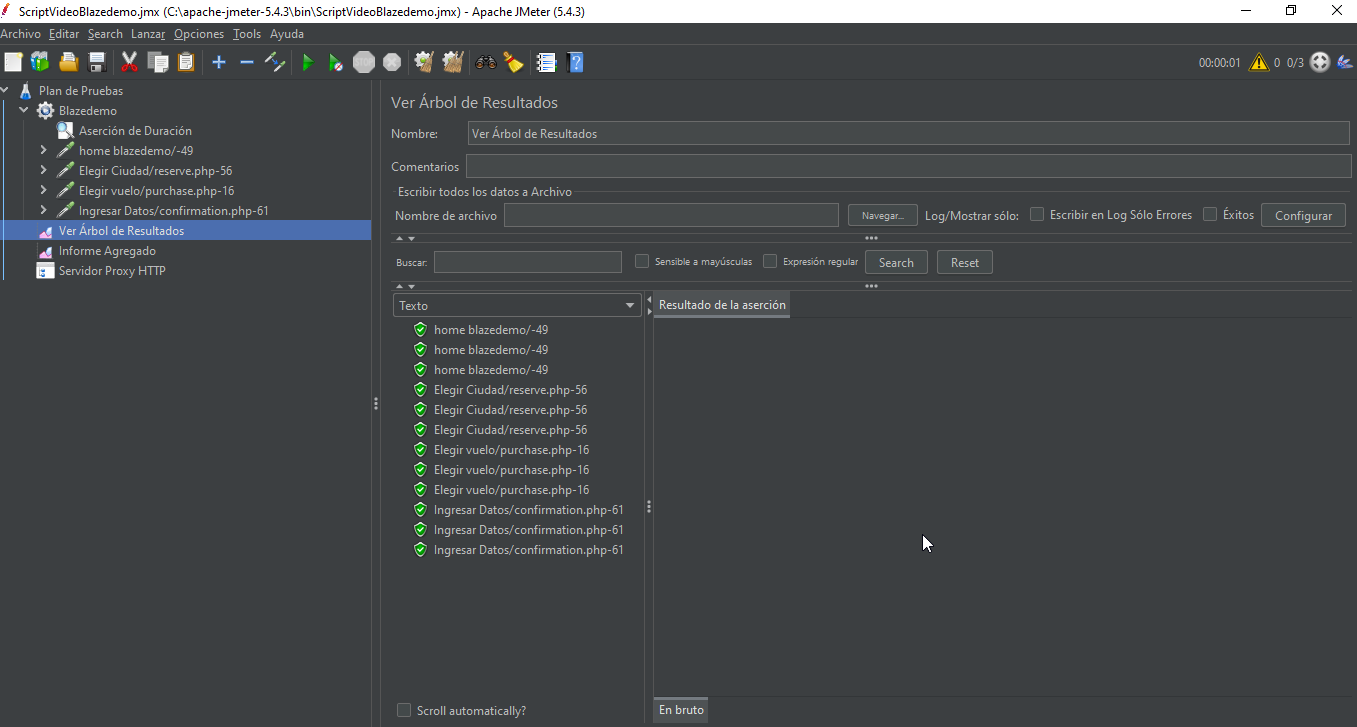




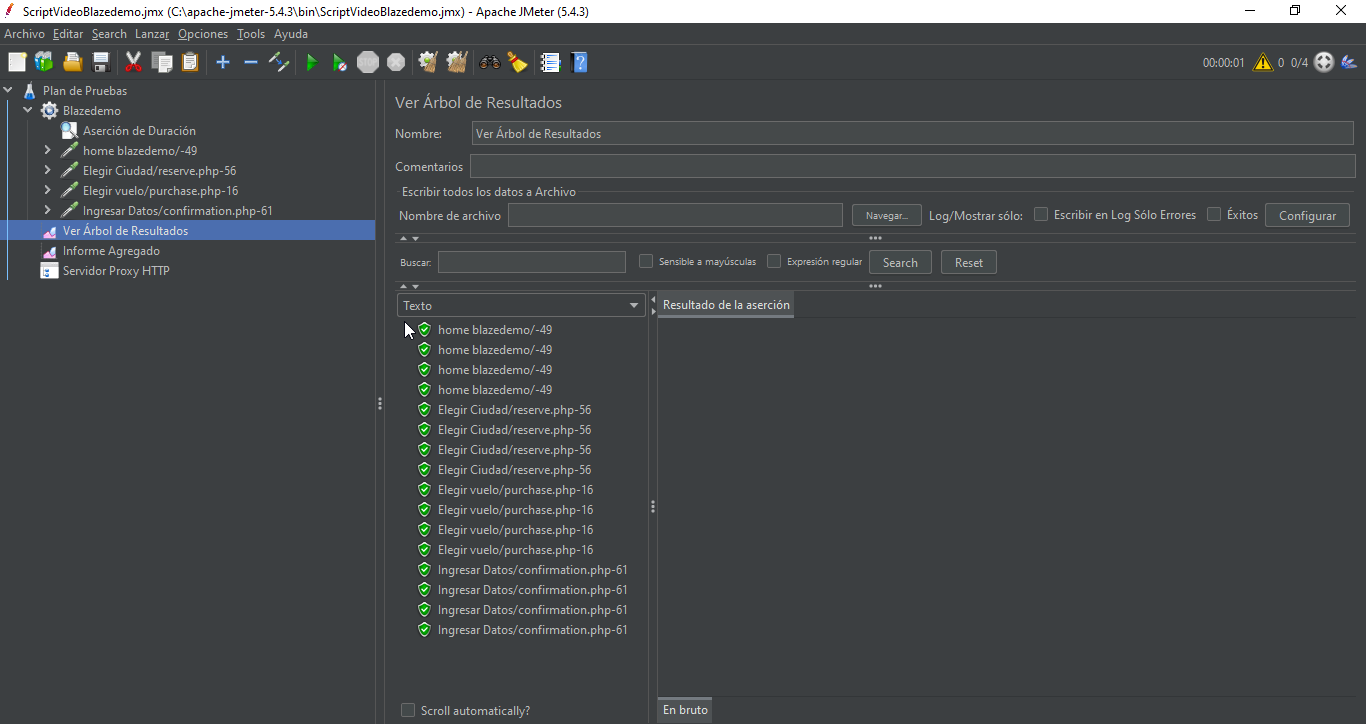


Y para realizarlo de manera exitosa le disminuyo la cantidad de usuarios en concurrencia para que me alcancen el nivel requerido implementado a ese tiempo de 600 ms y sea una ejecucion exitosa la cual después de realizar pruebas con diferentes cantidades de usuarios entre 1 y 20 la respuesta del servidor en el tiempo de duración era intermitente ya que de 5 usuarios hacia adelante no cumplia con el tiempo de duración implementado para las pruebas pero de 5 hacia ataras en ocosaiones era exitoso y en otras sobrepasaba el tiempo estipulado esto me lleva a deducir que no todo el tiempo el servidor responde en los tiempos de duración estipulados si no que varian deacuerdo a la concurrencia y cantidad de usuarios que ingresan:

Con 3 usuarios:Exitoso

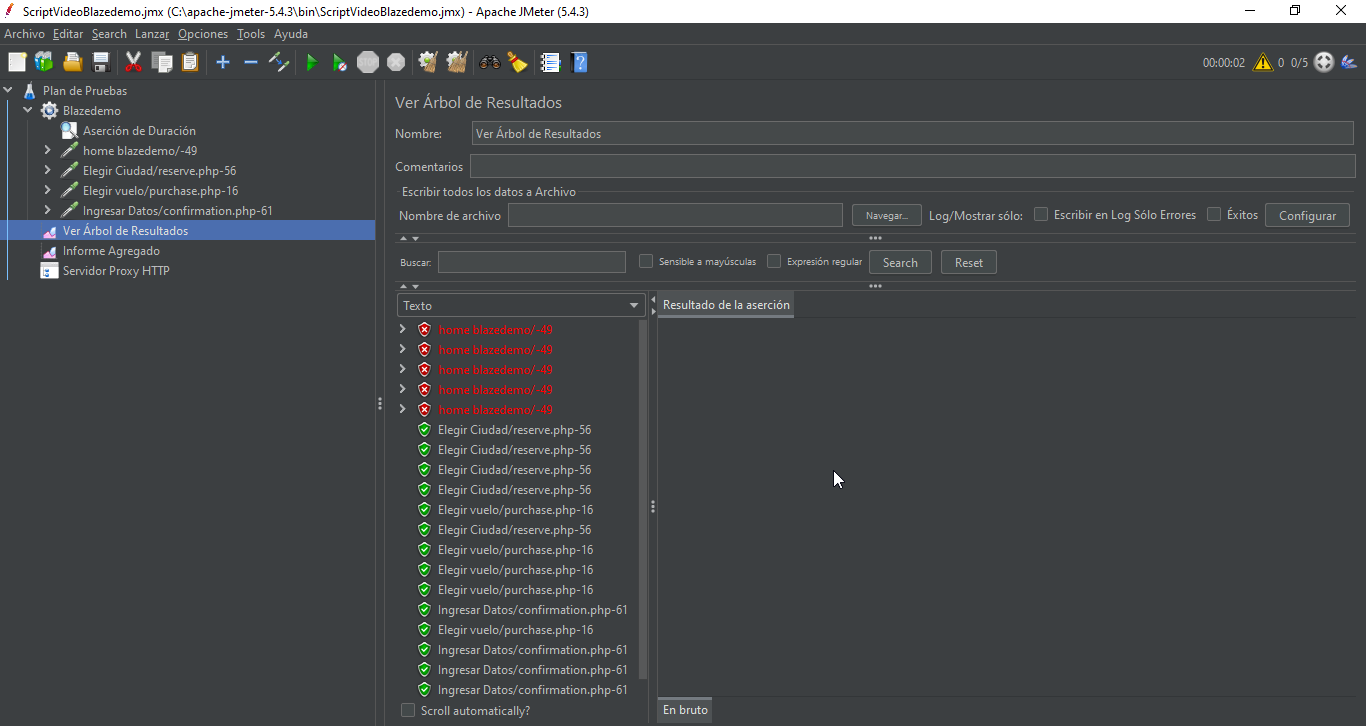


Con 4 usuarios: Exitoso



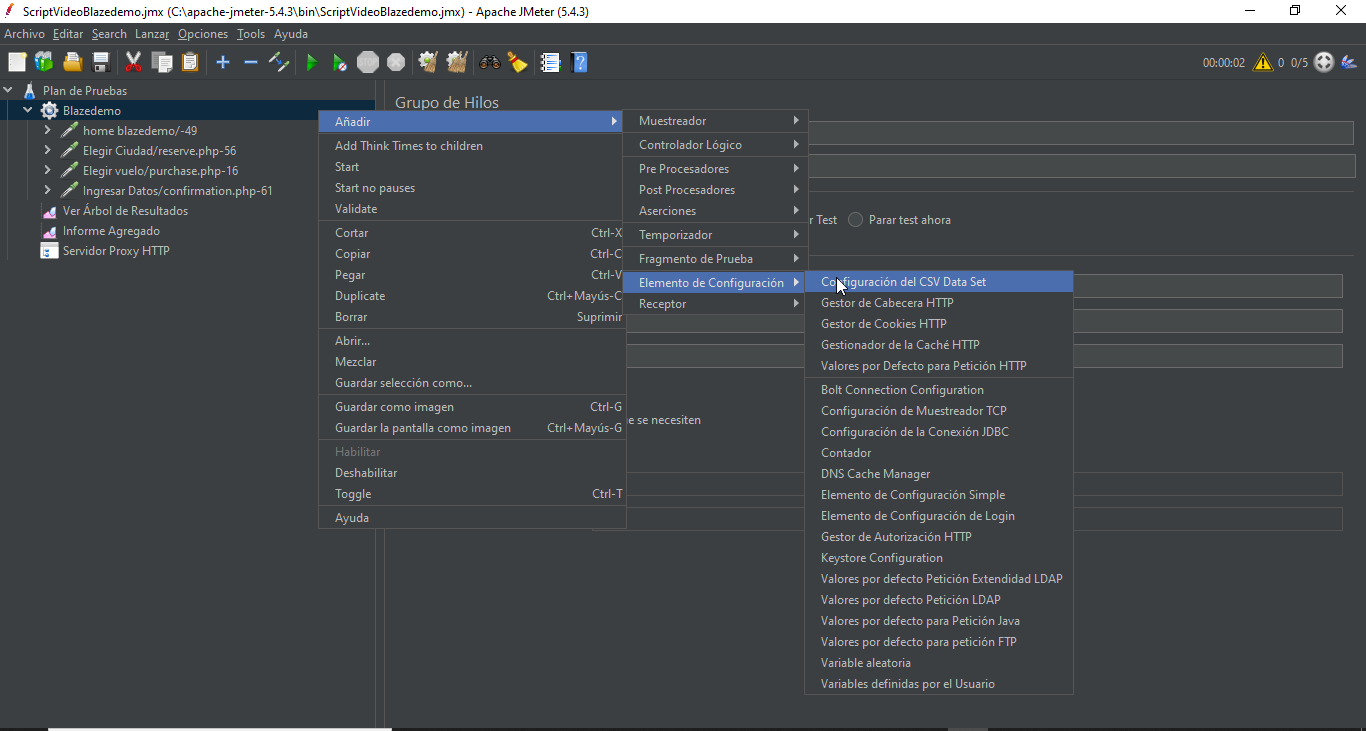
Con 5 usuarios:

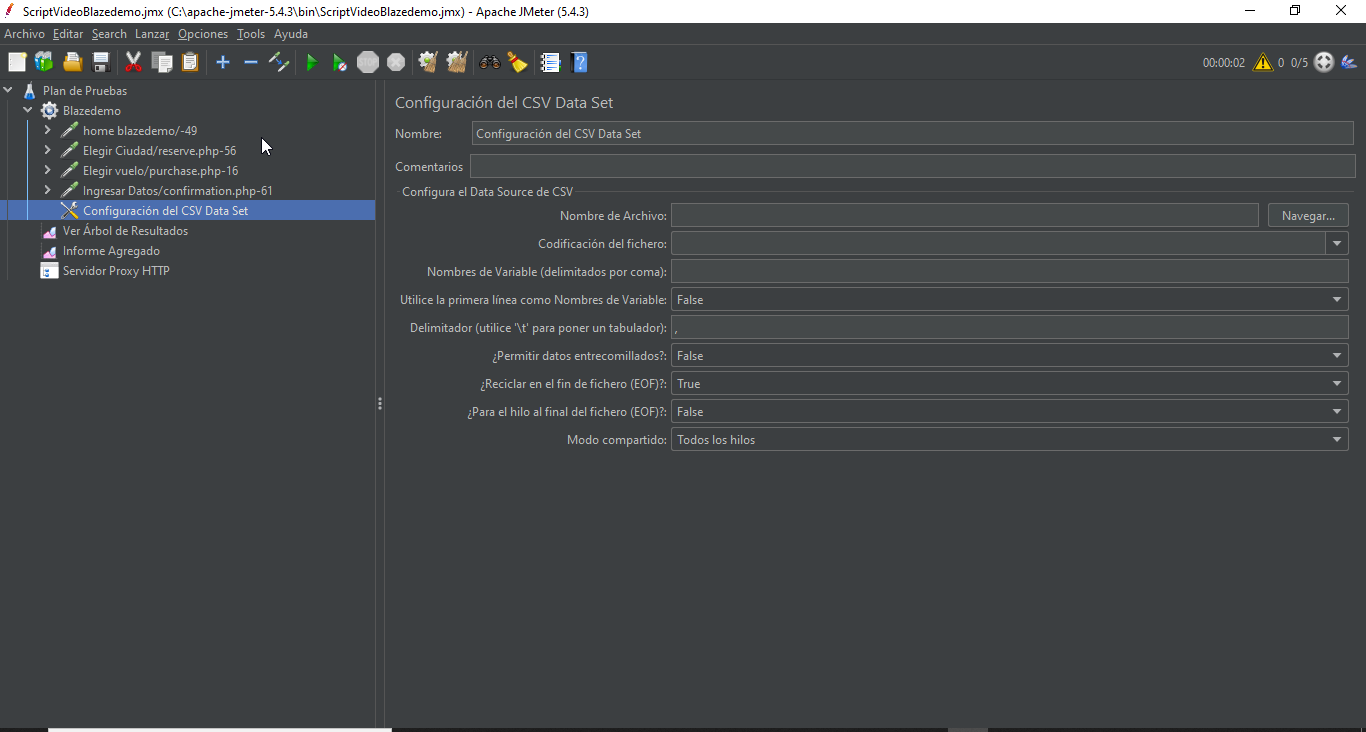
En esta captura ya sobrepasa el tiempo de duración que se implementó en las pruebas ya desde esta parte en adelante es fallido



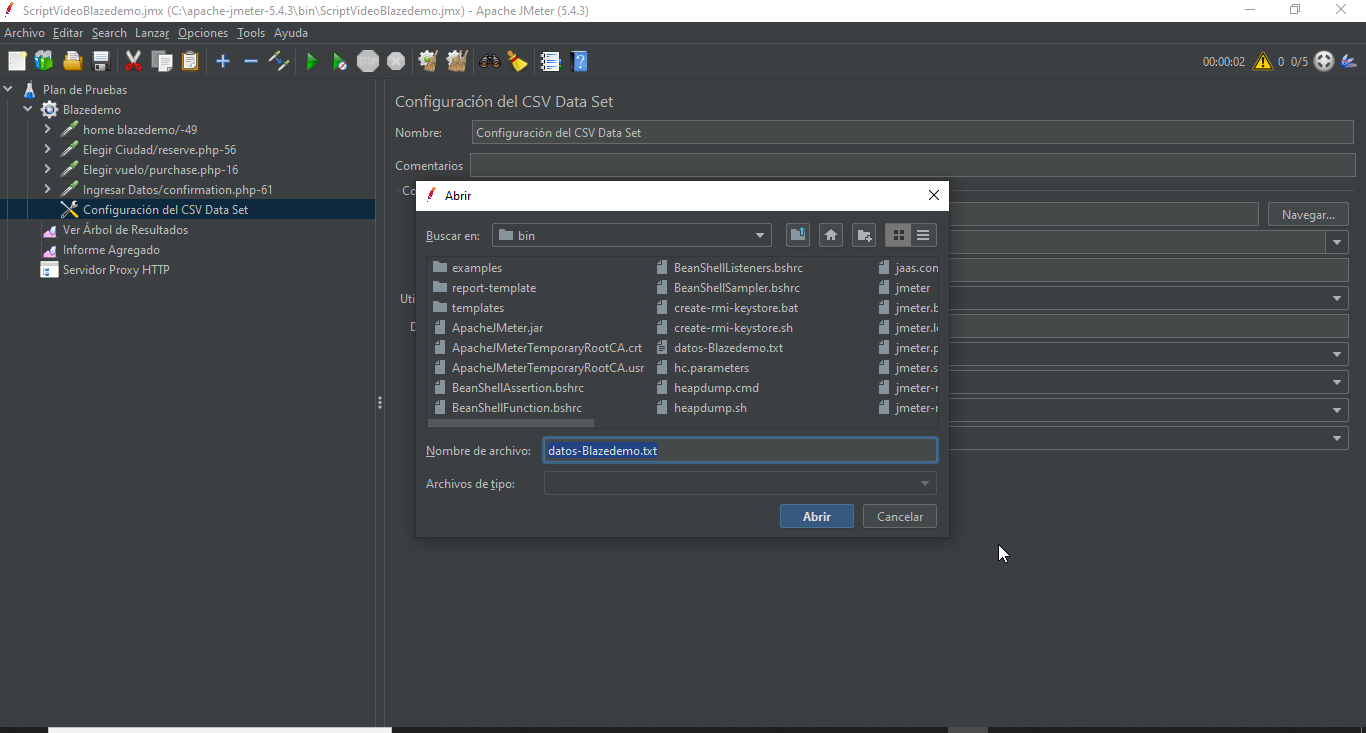
# Parametrizacion del Script JMETER

Primer paso: Selecciono la petición de Ingresar datos realizo la implementación del elemento de configuración que es Configuración del CSV Data set que me permitirá realizar la Parametrizacion este elemento lo que permitirá es alimentarse de un archivo que voy a tener almacenado en mi máquina ya sea un CSV o un TEXT DOCUMENT con este archivo puedo alimentar mi script.

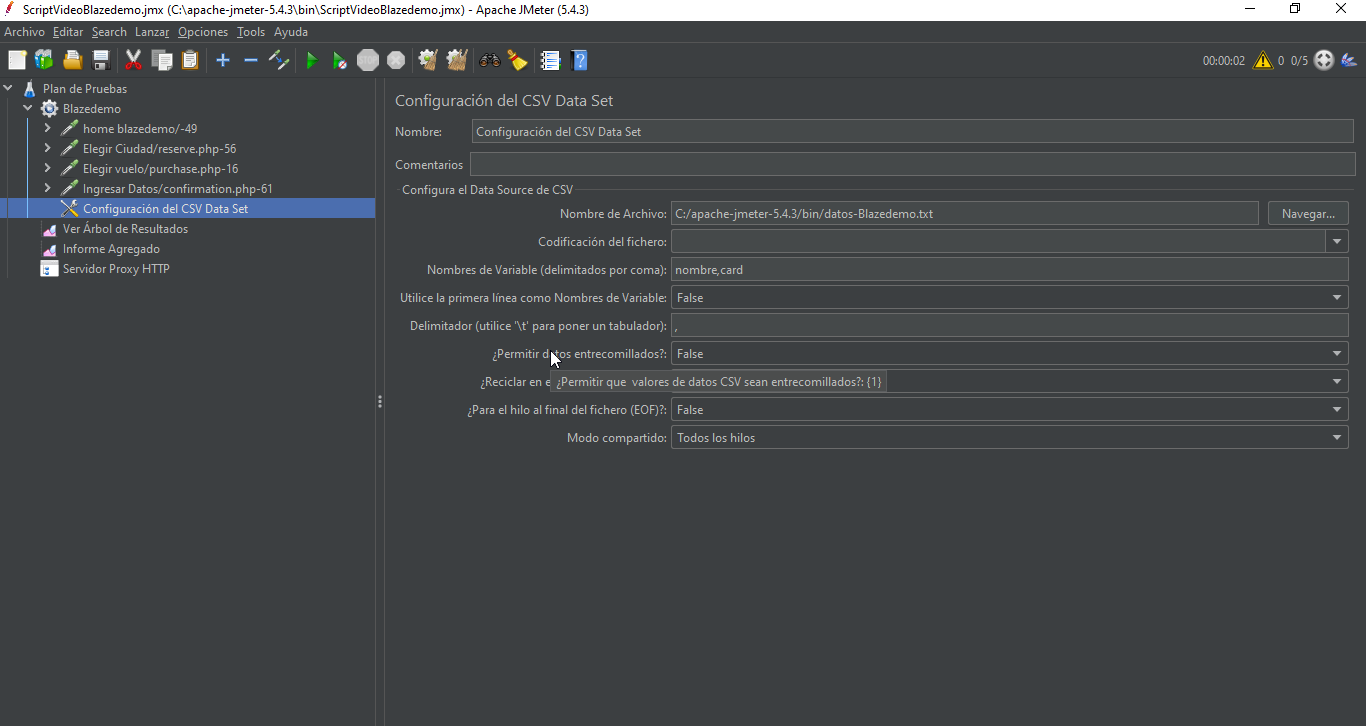




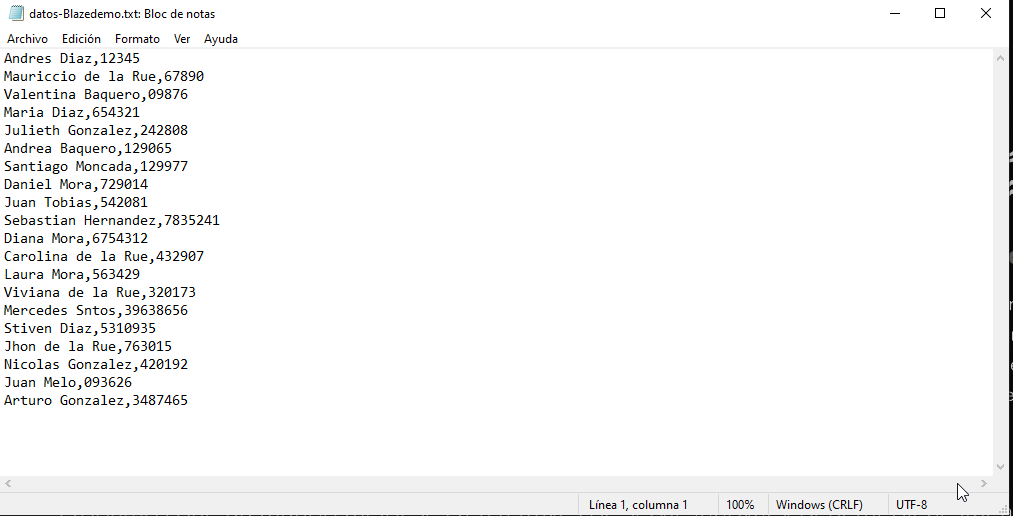
A continuación oprimo la tecla navegar y busco mi documento y lo selecciono y le doy abrir:



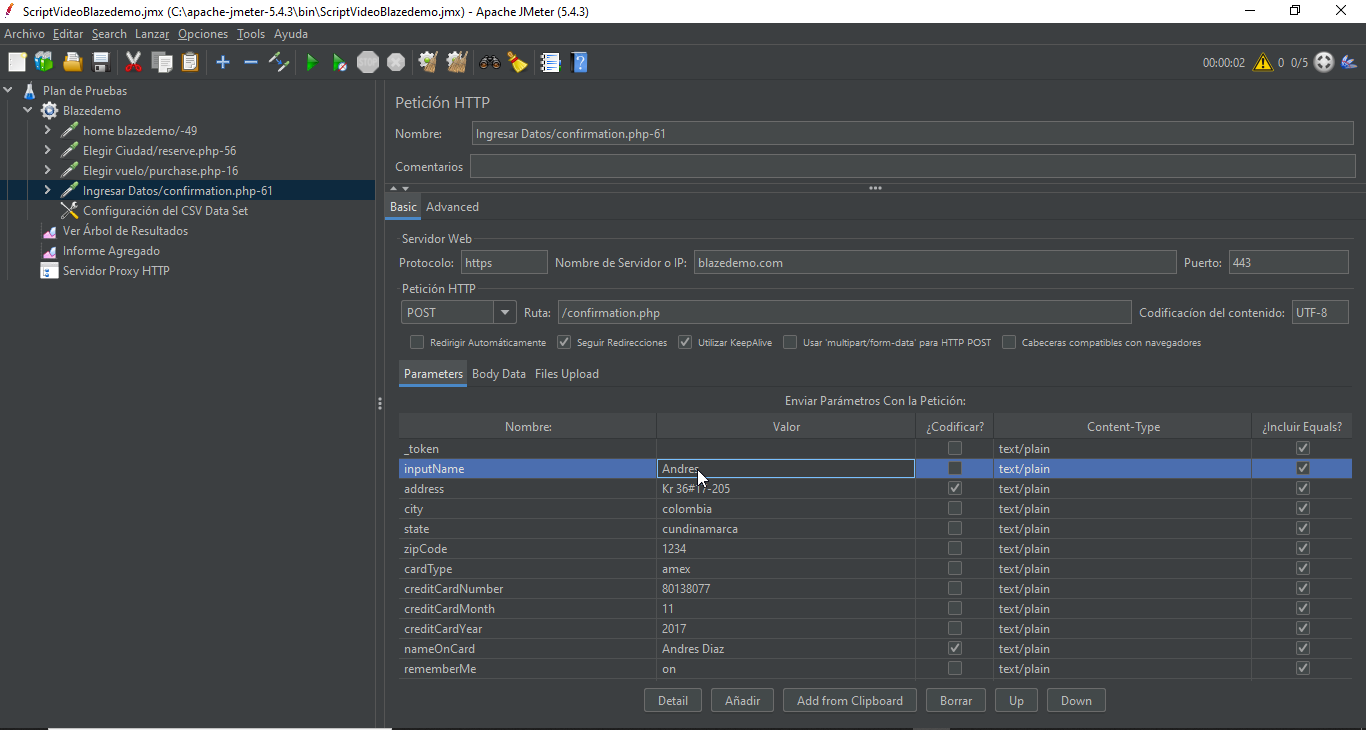
En el campo Nombre de variable inserto las dos variables que son nombre, variable que son las que tengo implementadas en mi documento que anexo en la otra captura.



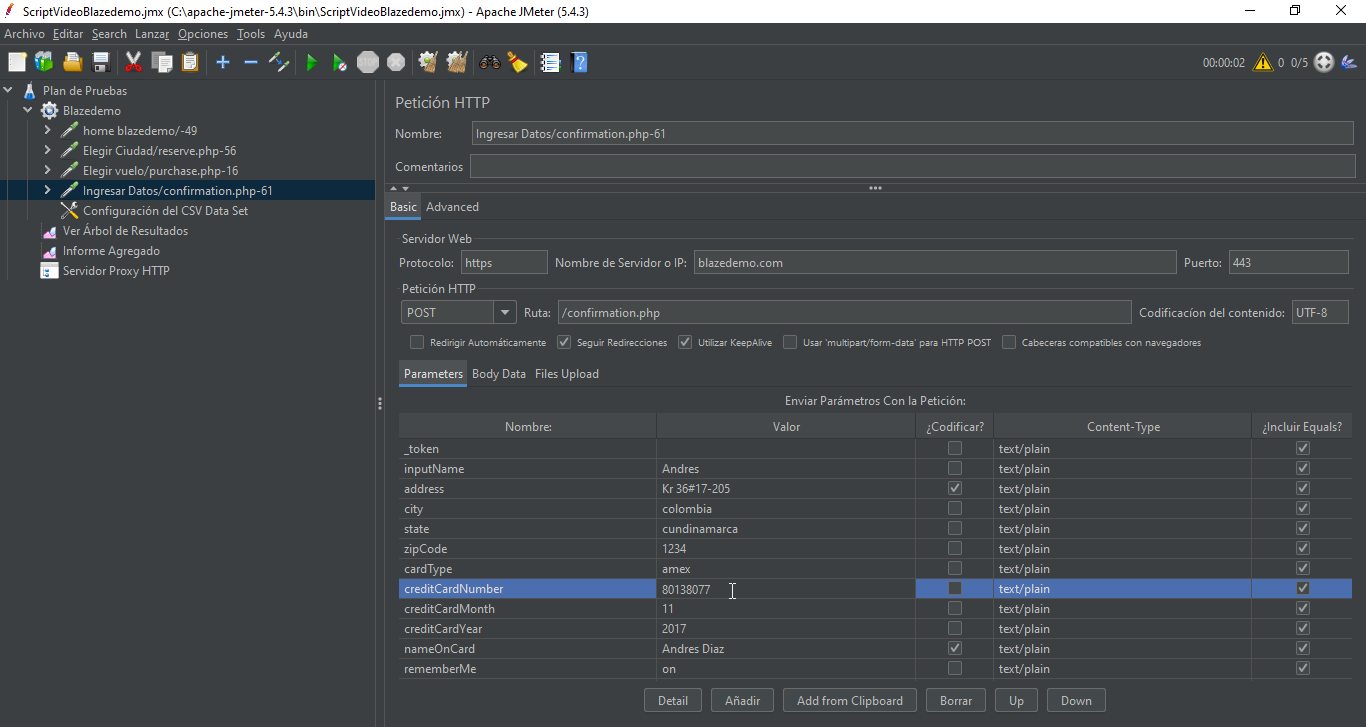
Como se evidencia en los datos la primera variable hace referencia al nombre de cada uno de los usuarios y card a los números que se van a utilizar para la tarjeta esto lo realizo para la concurrencia de 20 usuarios.



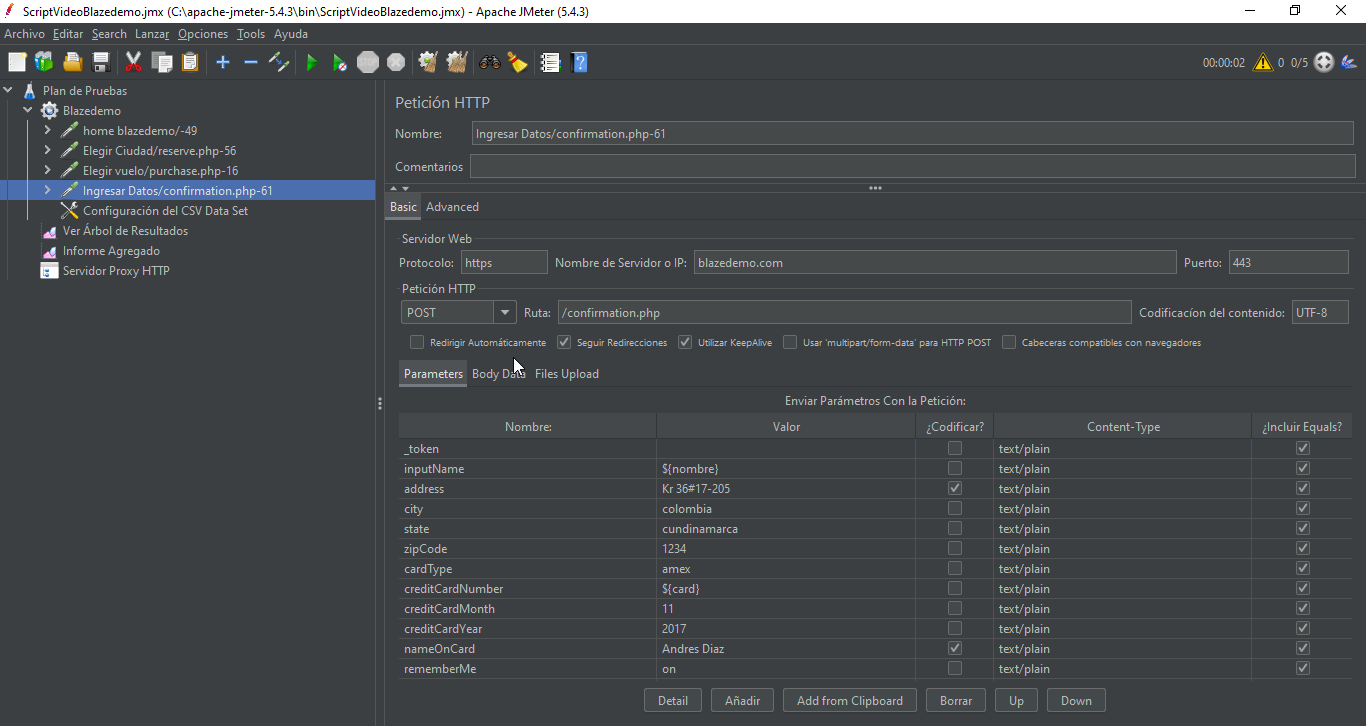
Después de esto ya no voy a enviar Andres como dato estático si no va hacer dinámico.



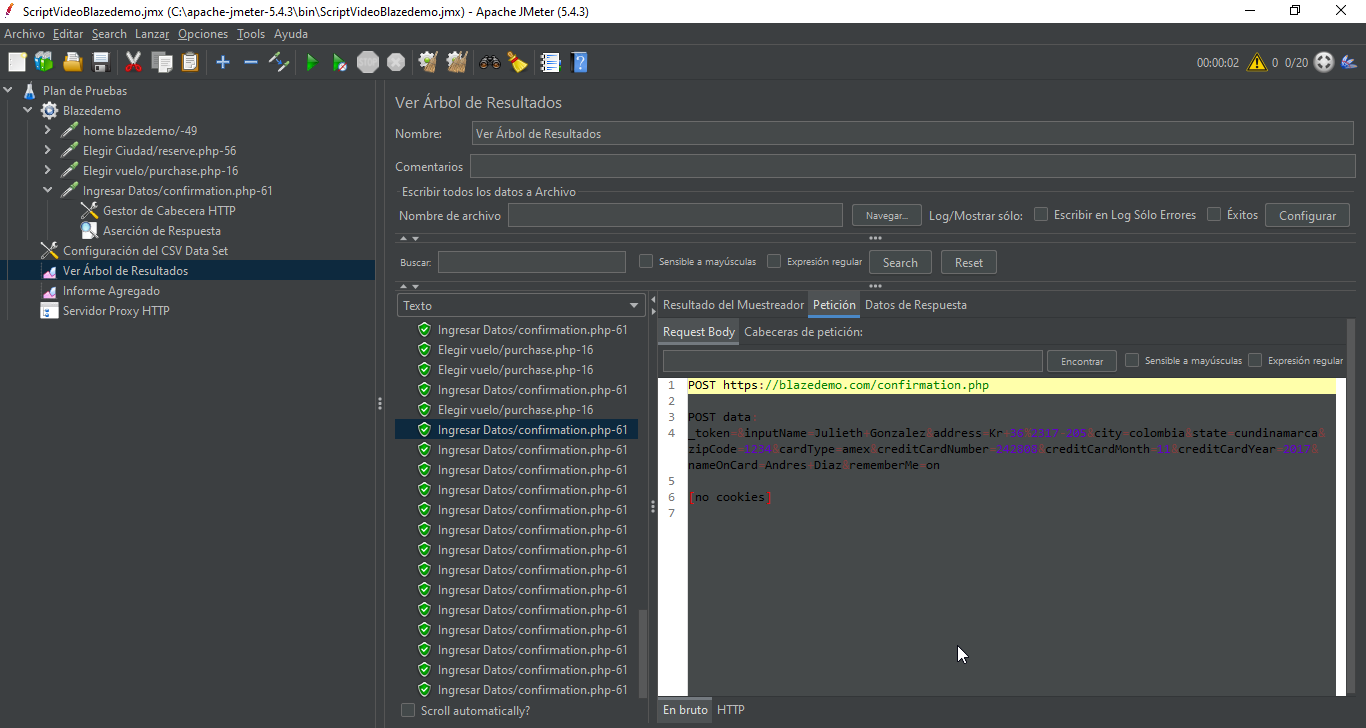
Y de igual manera con el campo de número de tarjeta ya no va hacer estático si no dinámico:

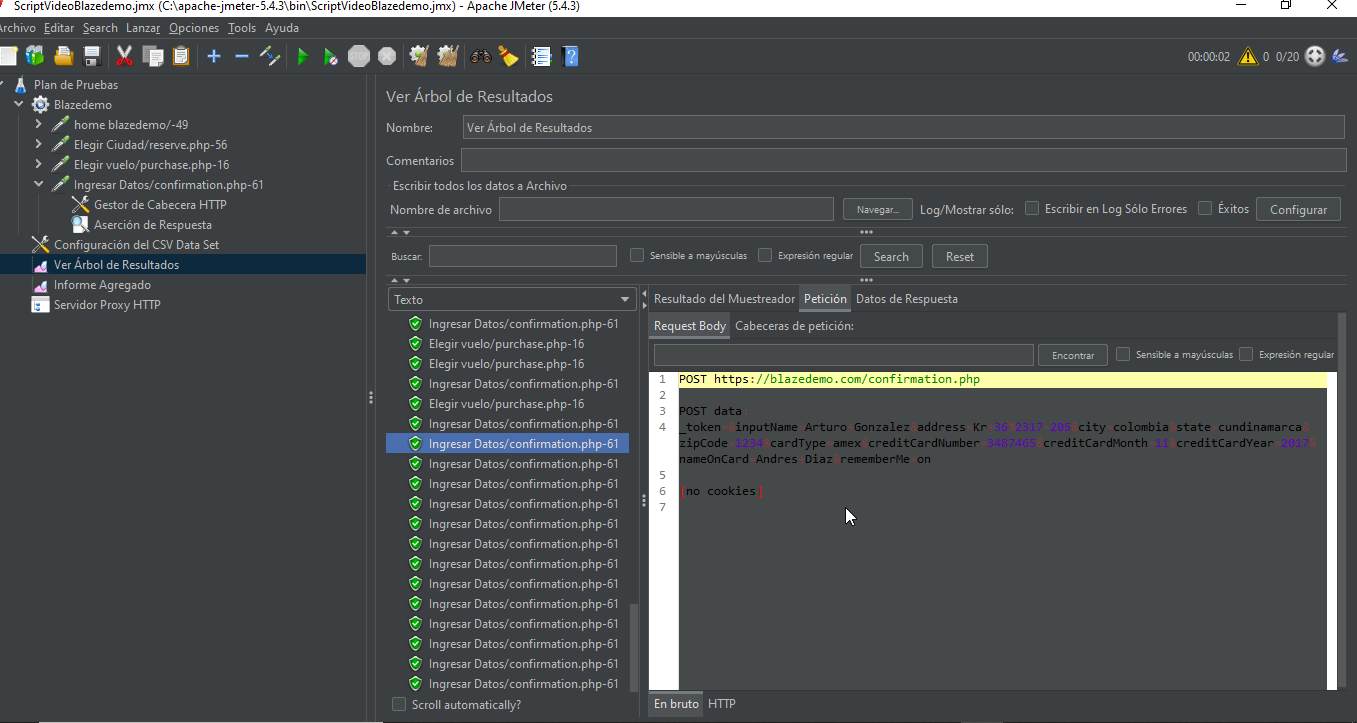


Los cuales también voy a declarar en el hilo de la petición:



Como se puede evidenciar después de que realizo la ejecución los datos ya son dinámicos y asi sucesivamente esta implementado en todas las peticiones cada una con su usuario y número de tarjeta con un nombre y un número de tarjeta únicos entre los 20 usuarios a realizar pruebas.

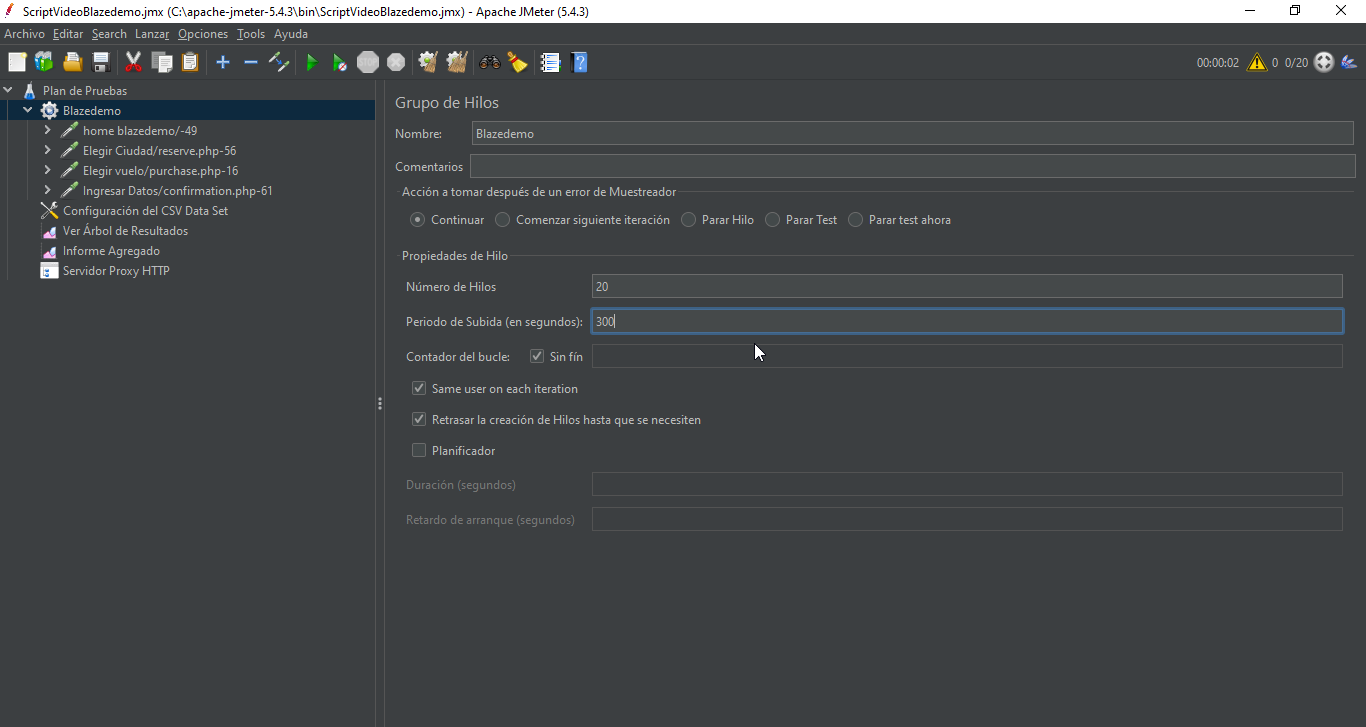




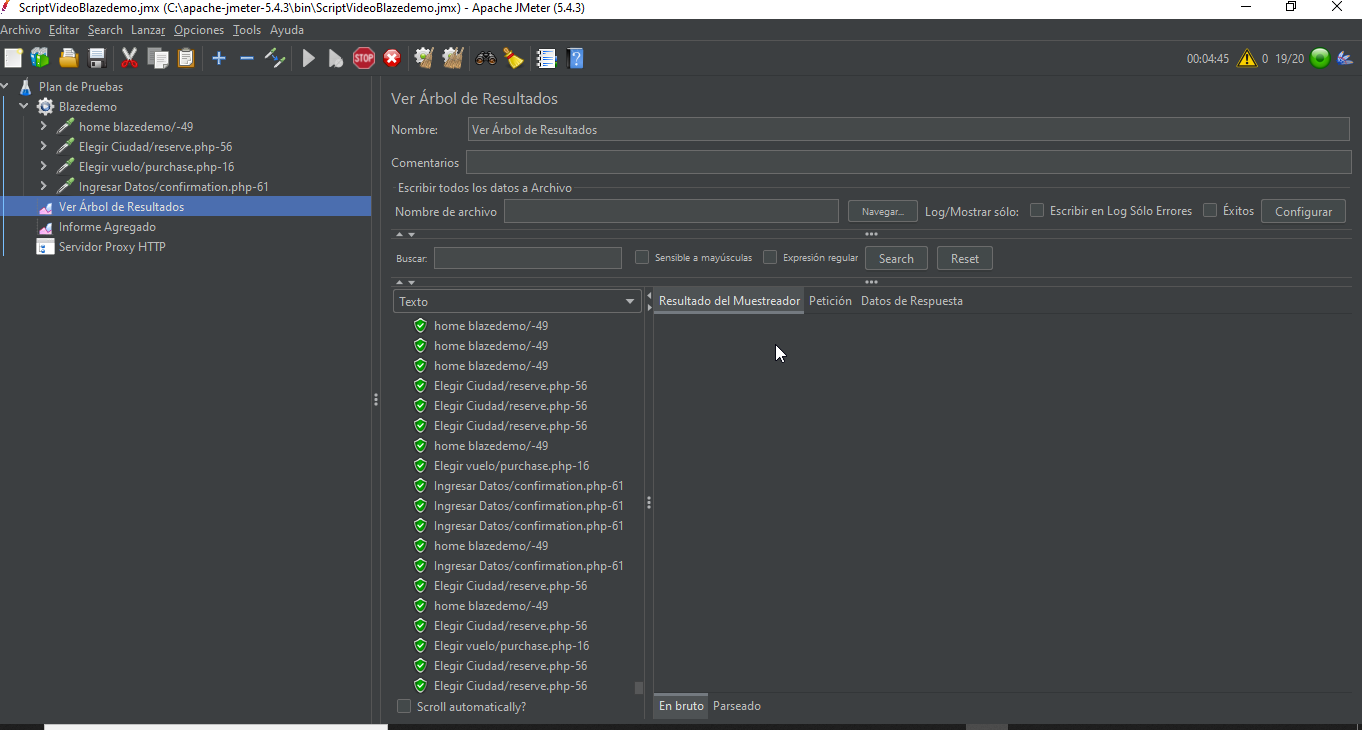
# Ejecución y Generación de carga en JMETER

En este paso realizare la parte principal que es el flujo de Hilos es en esta sección donde lo estaré desarrollando acá es donde se almacena toda la información de peticiones y grupo de hilos.

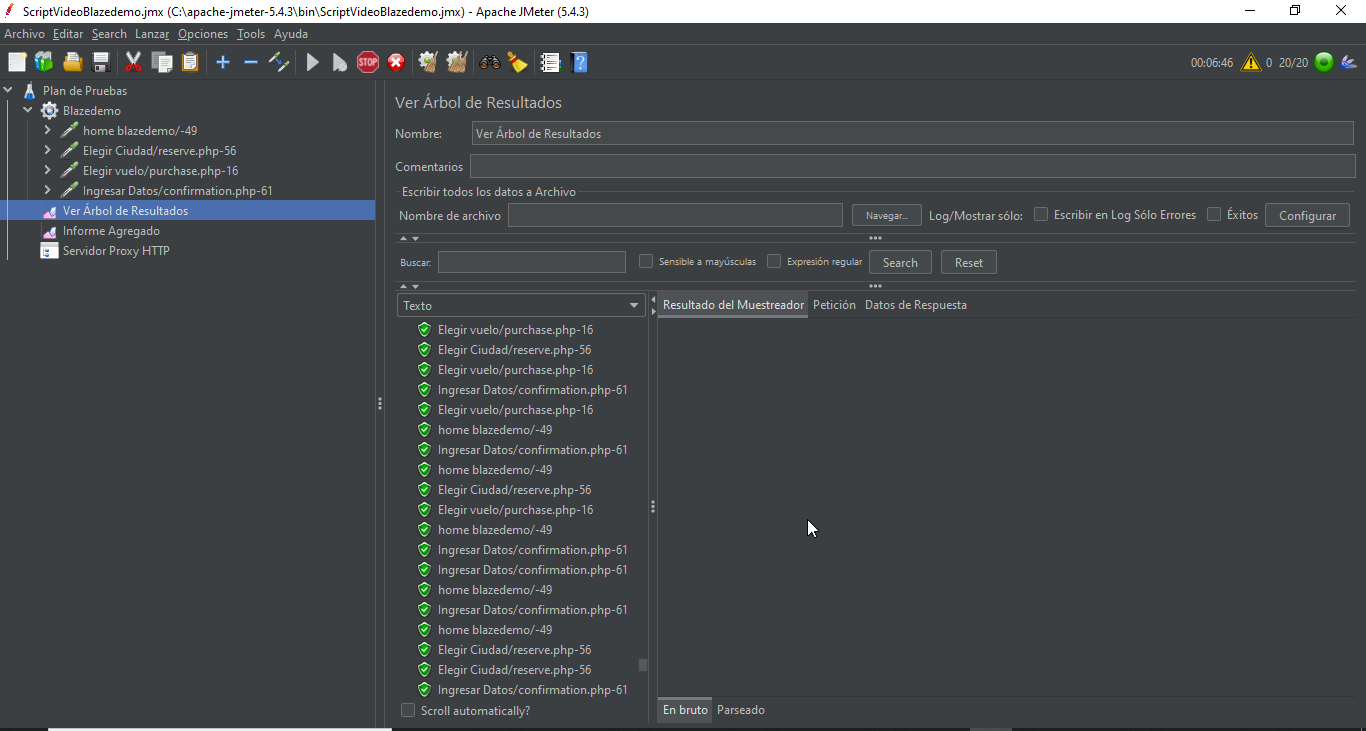
Voy Generar la carga con 20 usuarios en un lapso de tiempo de 300 ms. Y que el bucle sea infinito esto va a ejecutar varias veces.



Se ve cómo va aumentando la carga y el tiempo de respuesta y se puede ir verificando si las peticiones fallan o no hasta el momento no han fallado esta una de las maneras de dar concurrencia

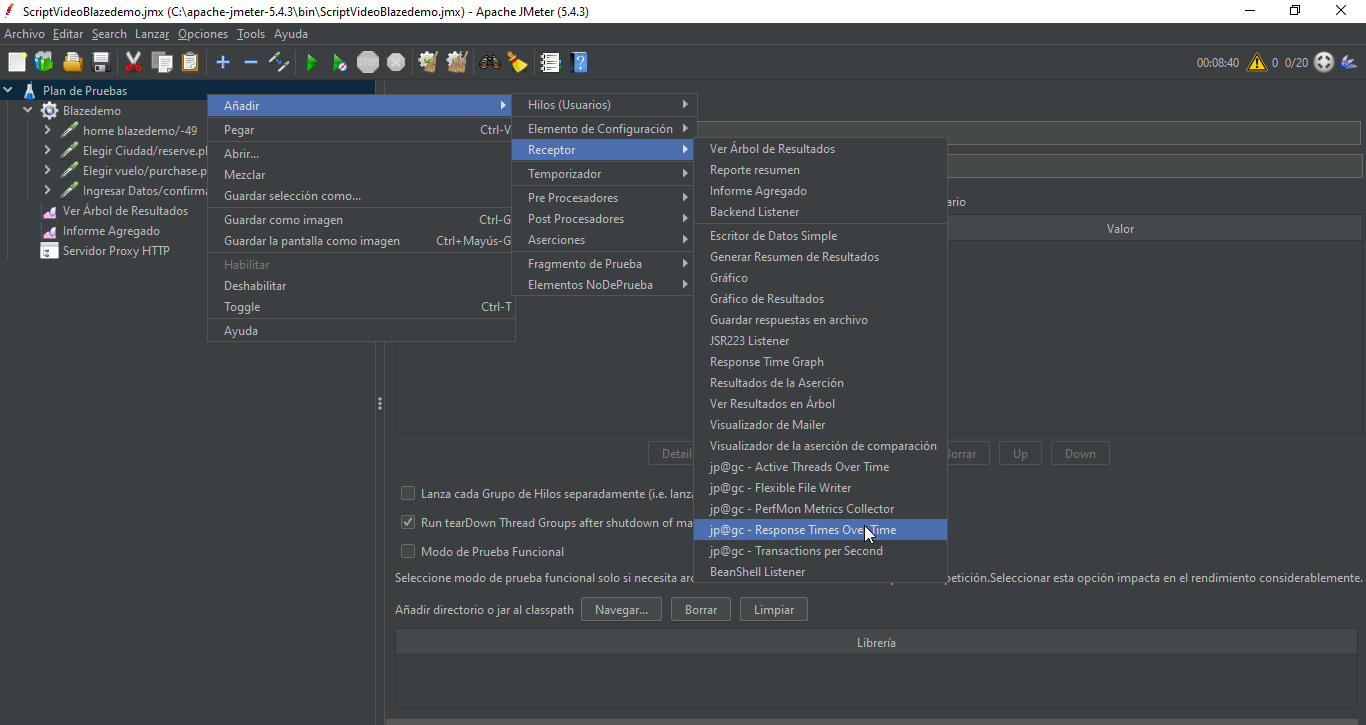


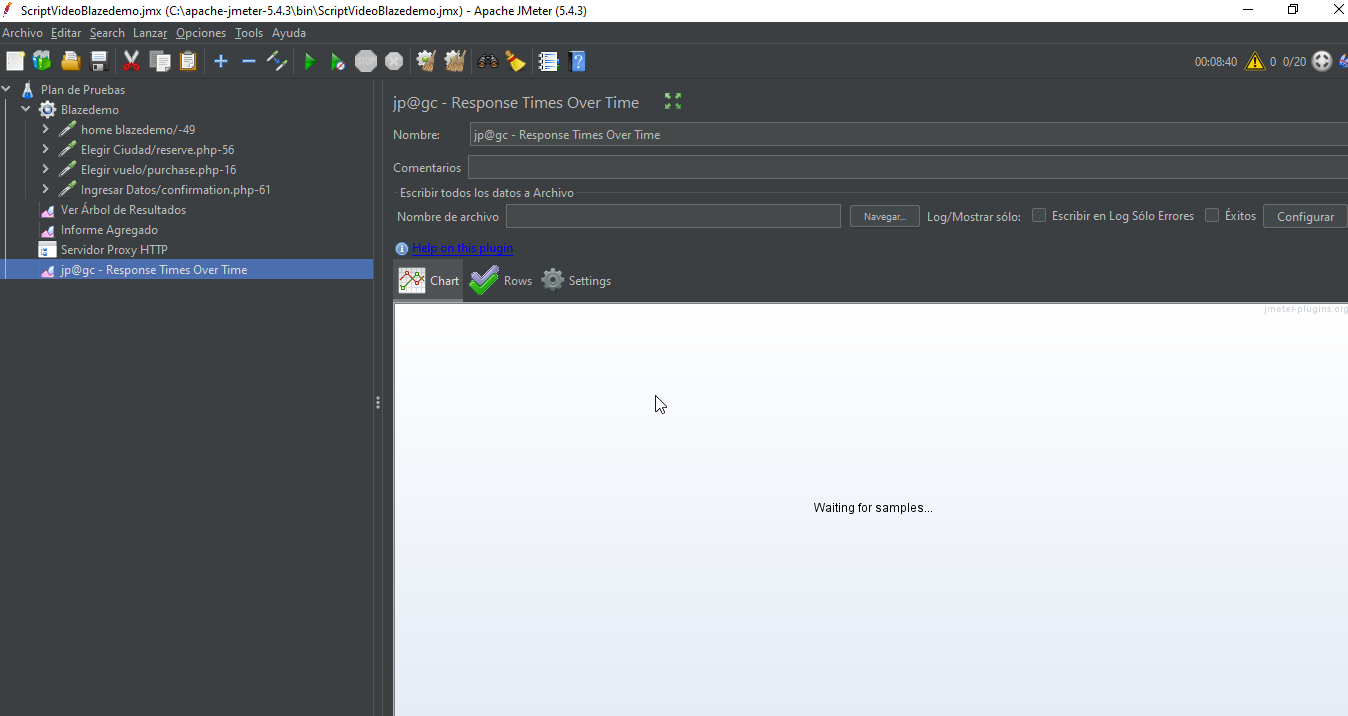
Se puede evidenciar que la carga implementada en el bucle infinito se ha comportado de manera exitosa con 20 usuarios en concurrencia



# A continuación utilizare el plugin implementado jpgc - Standard Set

Para la implementación de uso se realiza lo siguiente el primer elemento jp@gc - Response Times Over Time que significa los tiempos de respuesta sobre el tiempo :

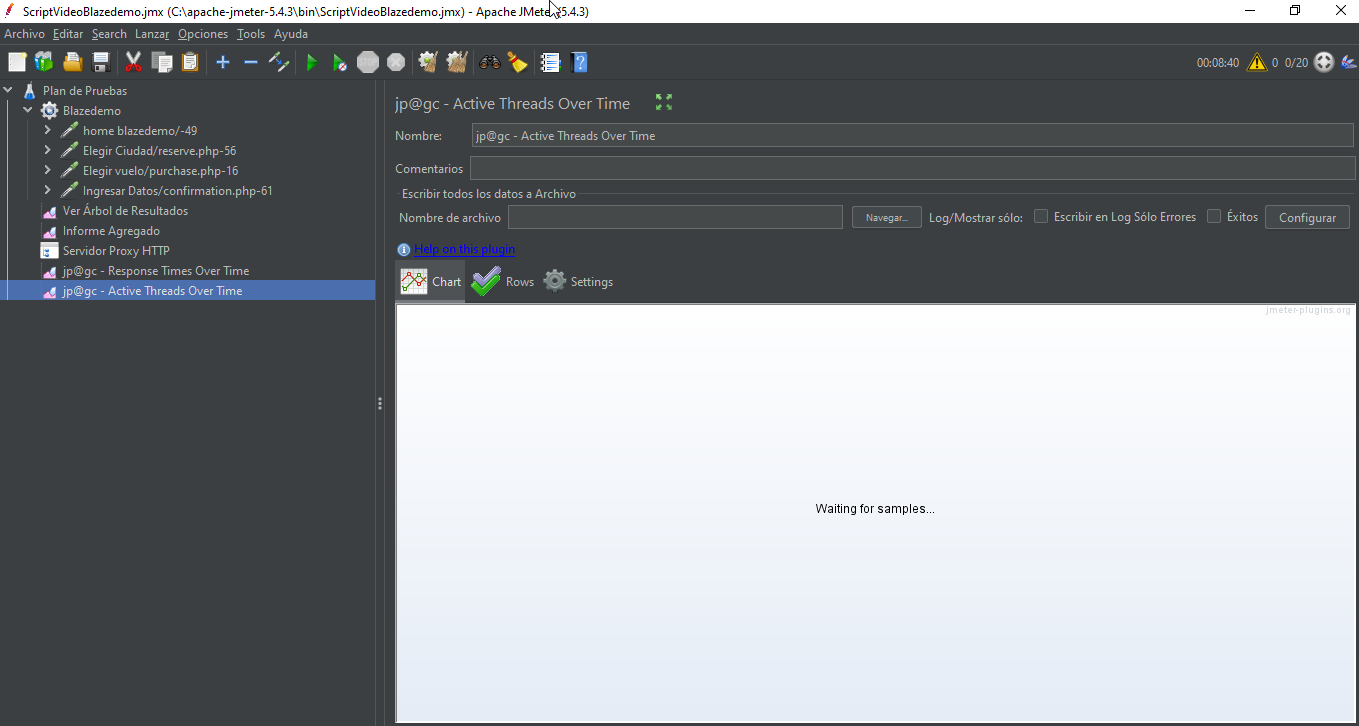




Y adiciono el otro elemento que es jp@gc - Active Threads Over Time que significa los usuarios activos sobre el tiempo:

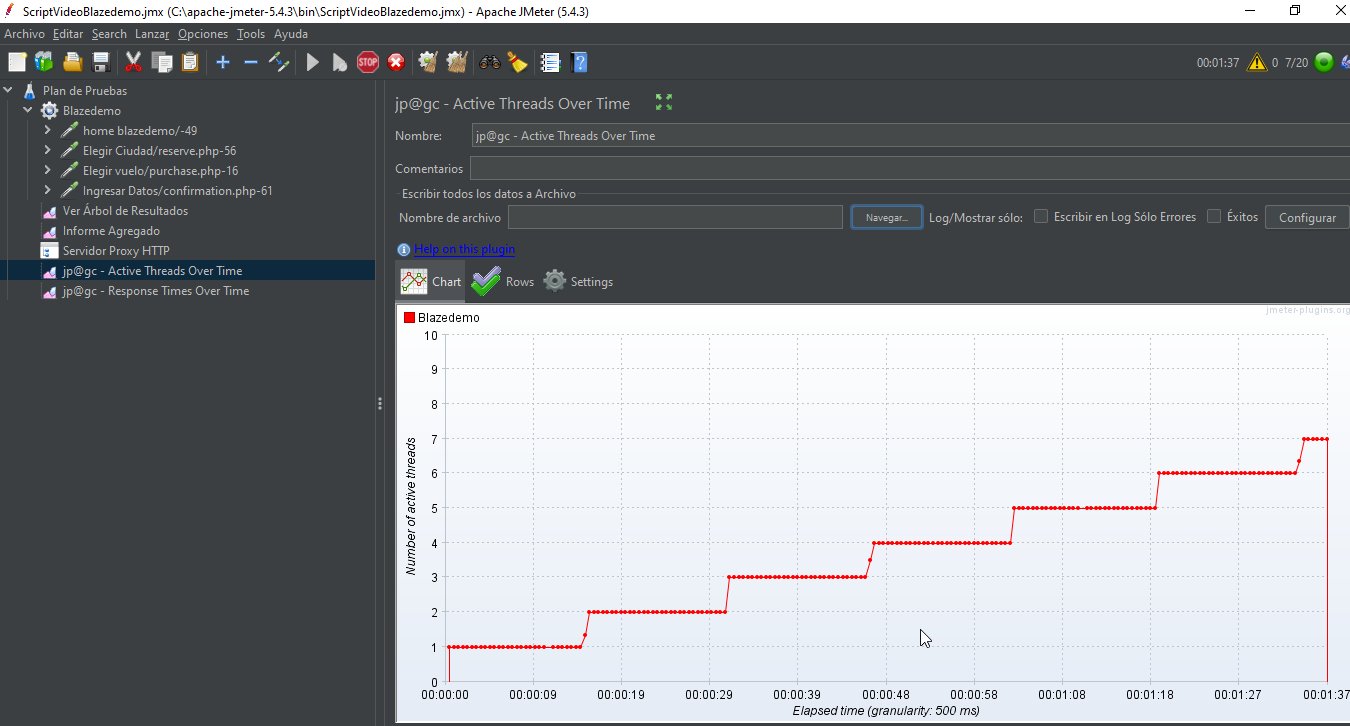
Y lo continuo realizando con la prueba infinita para generar la mayor cantidad de carga y concurrencia.



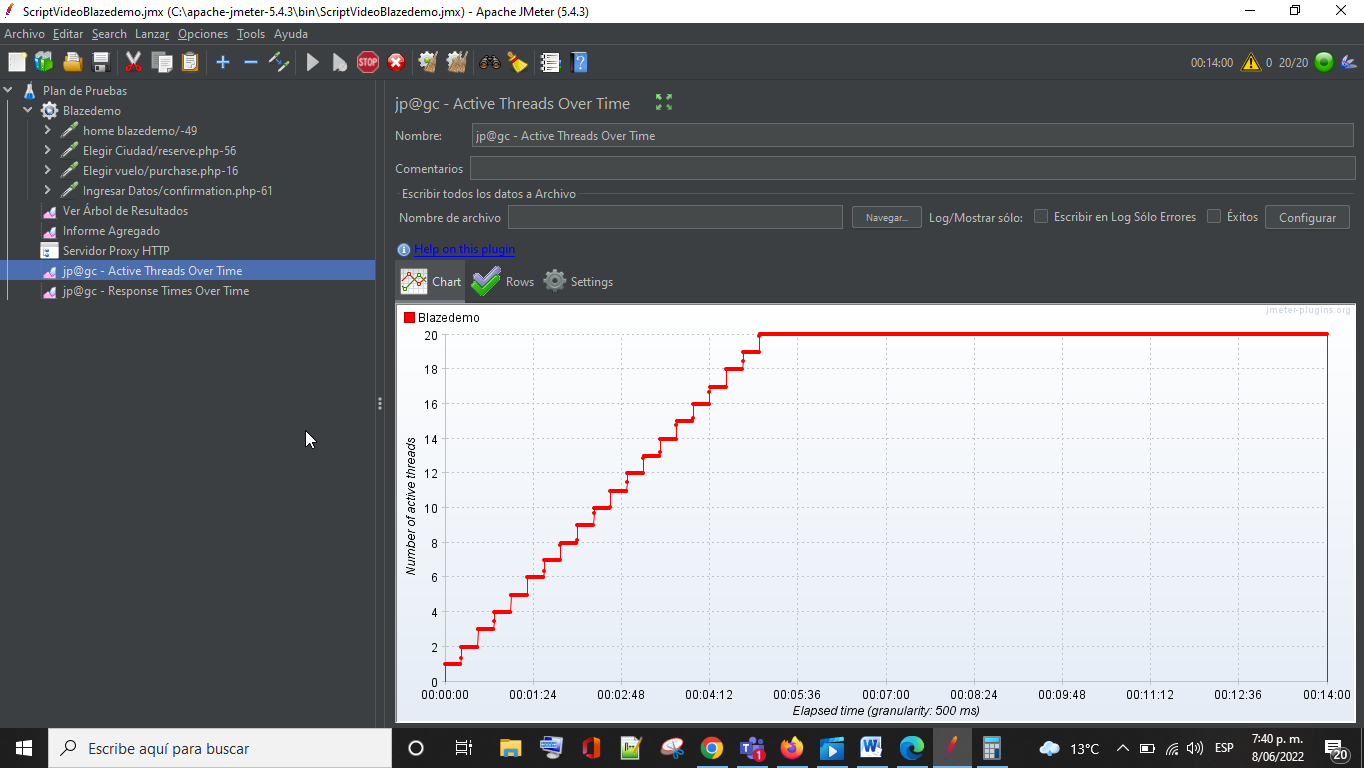


Posterior a esto lo ejecuto de nuevo y así poder visualizar todo el tema relacionado con la conexión de los usuarios, tengo 300 segundos de tiempo y 20 usuarios conectados esto me va a realizar la conexión de cada usuario cada 15 segundos lo que me dará como resultado la conexión de los 20 usuarios, esto es lo que se conoce como Ram-Up que me permite que los usuarios no se puedan conectar al mismo tiempo en el aplicativo, y si vemos de cara a la realidad esto no se produce ya que los usuarios no se ponen de acuerdo para realizar el ingreso, lo normal es que se conectan en diferentes momentos ya que el objetivo de estas pruebas de carga y rendimiento es realizar una simulación real al comportamiento que puede o tiene un aplicativo con todo y sus características generalizadas. Y así poder obtener su concurrencia y así visualizar en la gráfica y validar que se esté cumpliendo con los requerimientos implementados en la prueba como se afirma en la captura a continuación.

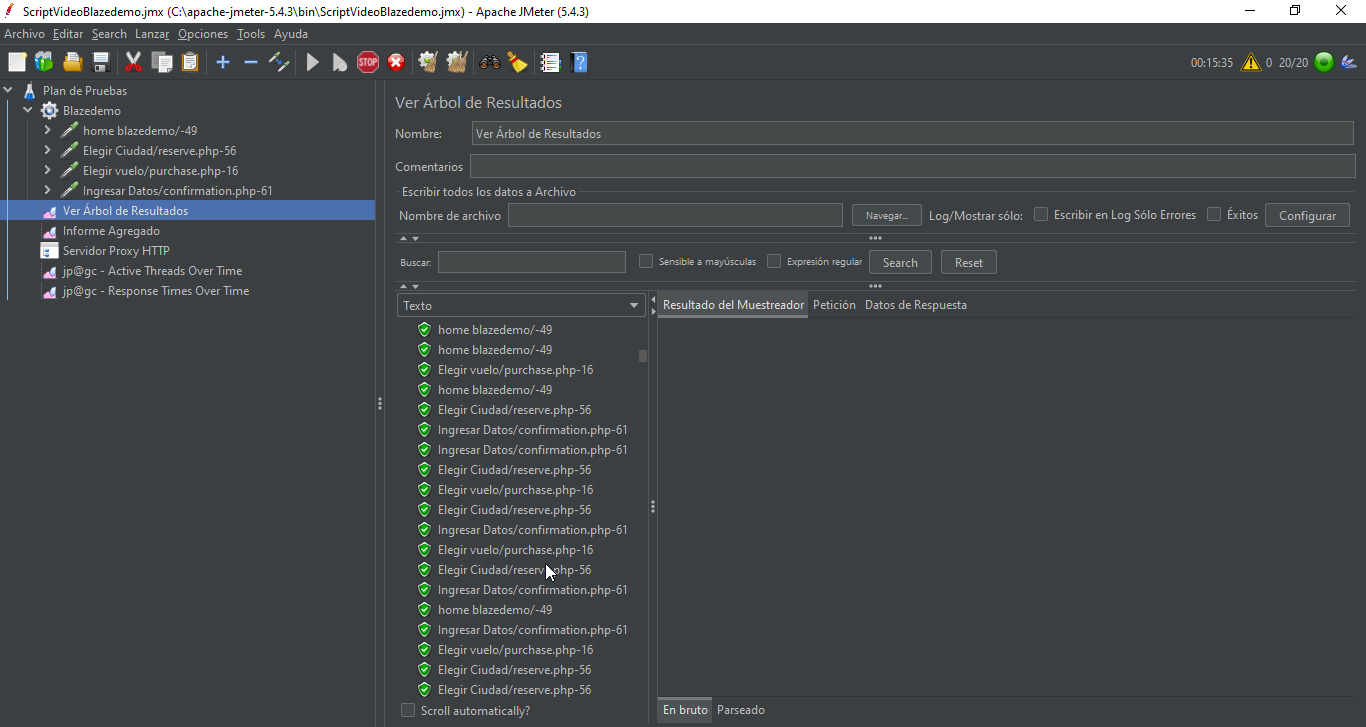
# jp@gc - Active Threads Over Time que significa los usuarios activos sobre el tiempo



En esta imagn se visualiza la finalización de la ejecucion de los 20 usuarios cada uno teniendo un margen de espera de 15 segundos



Y todas sus peticiones de manera exitosa.



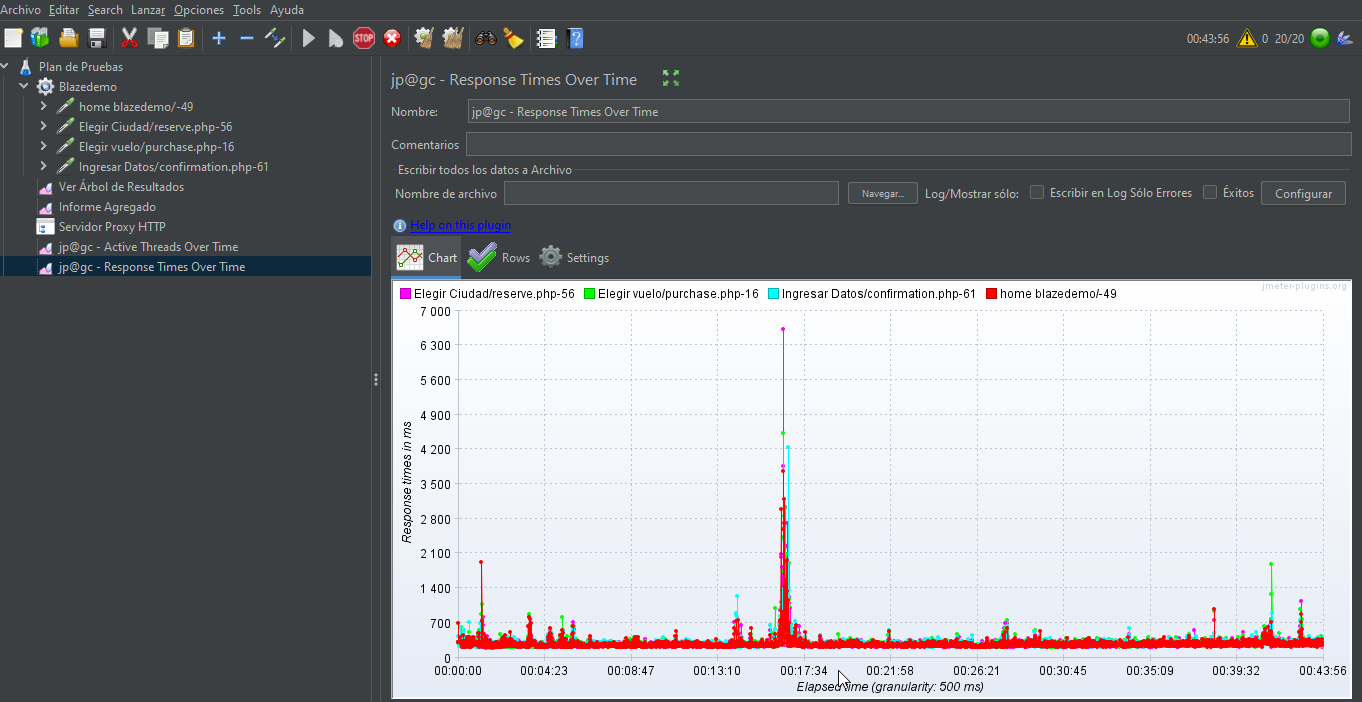
# jp@gc - Response Times Over Time que significa los tiempos de respuesta sobre el tiempo

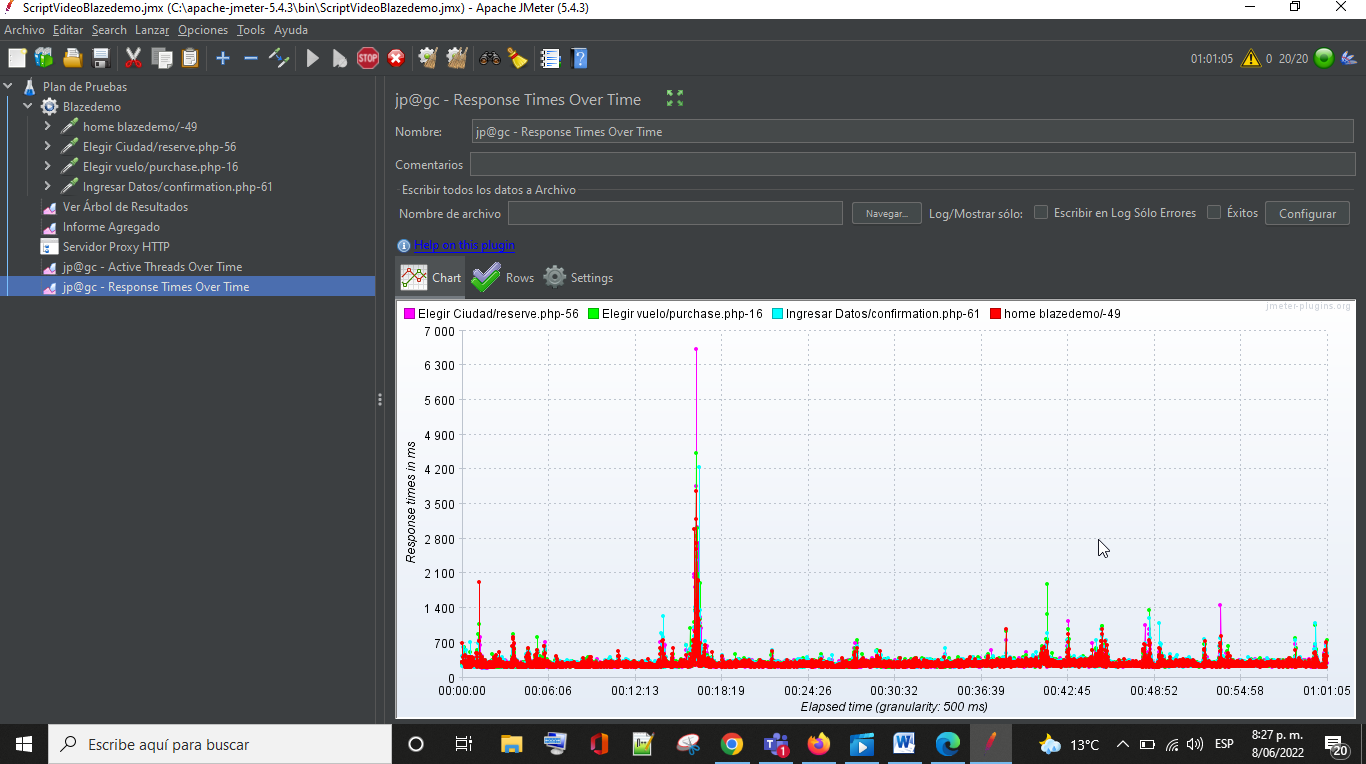
Esta grafica nos representa en colores que identifican la especificación de cada petición el color Rojo representa el home blazedemo el cual en la gráfica evidencia que está a un nivel medio de elevación de tiempo que los demás ha llegado hasta casi 4000 milisegundos,

La petición de elegir vuelo de color verde se evidencia que el tiempo de respuesta y el tiempo de ejecución se ha mantenido estable entre los 700 y 800 milisegundos sin tener elevaciones,

La petición ingresar datos de color azul a tenido una elevación en el tiempo trascurrido de los 4200 milisegundos después de estar en un nivel entre los 800 y 1200 milisegundos en el tiempo.

La petición elegir ciudades está en un estado crítico o muy elevado en el tiempo de ejecución de la carga junto con la concurrencia de los 20 usuarios ingresando llegando al consumo aproximado de 7000 milisegundos



Como se puede evidenciar después de 1hora de tiempo transcurrida el flujo de tiempo de respuesta se mantiene como se encuentra en la gráfica anterior lo que me demuestra que la carga obtenida por la concurrencia de 20 usuarios en 300 segundos que se aplicó solo obtuvo elevaciones de tiempo en las peticiones del inicio de la página, ingreso de los datos, elección de vuelos y elección de ciudades en el indicador de tiempo transcurrido entre el minuto 12.13 y el minuto 17 como indica la gráfica después se estabilizo y siguió su flujo normal dentro de los rangos de 0 a 1800 milisegundos.

Como se puede evidenciar después de 1hora de tiempo transcurrida el flujo de las peticiones es estable sin ningún fallo ni error se mantiene y es exitoso

