Universidad de las Ciencias Informáticas Facultad 6

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

TÍTULO: Propuesta de herramienta para pruebas de rendimiento de carga y estrés a servidores streaming

AUTOR: Yadriel Castro Rosales

TUTOR: Ing. Yusdenys Pérez Mendoza

FRASE

"Quien tenga una computadora dispone de todos los conocimientos publicados. La privilegiada memoria de la máquina le pertenece también a él. Las ideas nacen de los conocimientos y de los valores éticos. Una parte importante del problema estaría resuelta tecnológicamente, la otra hay que cultivarla sin descanso o de lo contrario se impondrán los instintos más primarios".

Fidel Castro Ruz

DEDICATORIA

A mis padres por ser ellos la inspiración de este trabajo.

AGRADECIMIENTOS

A mis padres por ser lo más grande que me ha dado la vida, por el apoyo, confianza, amor y educación que me han dado siempre bajo cualquier situación.

A mi hermana por siempre ser ejemplo en mi vida, por darme tanto amor, cariño y por lo mucho que me ha ayudado en toda mi vida.

A mis abuelos por haberme dado tanto cariño en la vida, enseñarme y educarme en todo momento.

A mi novia Diamela Palomino Sánchez por estar a mi lado en estos últimos años de mi vida estudiantil dándome tanto amor, momentos felices, apoyando e inspirándome confianza en todo momento y ayudándome todo el tiempo en la realización de este trabajo.

A mis amigos de todos los tiempos Adrián, Alberto y Tony por ser tres hermanos más y siempre estar en las buenas y las malas brindando amistad y apoyo.

A Misvel y Liannys por aguantarme en los últimos años de universidad y por brindarme tanta ayuda en el transcurso de la universidad y en la realización de éste trabajo.

A mis amigos de la Universidad: Figuera, Papito, El Franio y Jean.

A mi tutor Ing. Yusdenys Pérez Mendoza por ayudarme en cualquier momento, dedicándome tanto tiempo y brindando siempre sus conocimientos.

A todas aquellas personas que a lo largo de mi vida han contribuido a este trabajo.

DATOS DE CONTACTO

Declaración de Autoría

Ciudad de La Habana, junio, 2011

“Año 53 de la Revolución”

Yo: Yadriel Castro Rosales declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del 2011.

Yadriel Castro Rosales. Ing. Yusdenys Pérez Mendoza.

_____ _____

FIRMA DEL AUTOR FIRMA DEL TUTOR

RESUMEN

Este trabajo propone el uso de la herramienta Jakarta JMeter para su utilización dentro del proceso de pruebas en los proyectos pertenecientes al Departamento de Señales Digitales. Se describen brevemente aspectos vinculados con la calidad de software, tipos de prueba, objetivos de las pruebas de software, características, principios, procesos de prueba, así como un breve resumen del estado de los servidores streaming a nivel mundial. Se realiza una comparación entre un grupo de herramientas que son utilizadas a nivel mundial en el proceso de ejecución de pruebas automáticas, se justifica la elección de la herramienta propuesta para la automatización del proceso de pruebas en los proyectos pertenecientes al Departamento de Señales Digitales. Se aplica el proceso de pruebas con la herramienta Apache JMeter en la aplicación producida por el proyecto VideoWeb perteneciente al Departamento de Señales Digitales, quedando estas como un ejemplo más de la utilización de esta nueva herramienta.

PALABRAS CLAVES

Calidad de Software, Herramientas, Herramientas Automáticas, Jakarta JMeter, Pruebas de Software, Servidores Streaming.

ABSTRACT

This paper proposes using Jakarta JMeter tool for use in the process of testing projects in the Department of Digital Signals. We briefly describe aspects related to software quality, test types, test objectives of software, features, principles, testing procedures, as well as a brief summary of the status of the streaming servers worldwide. A comparison between a set of tools that are used worldwide in the process of implementing automated test, justifies the choice of the proposed tool for automating the testing process in the projects of the Department of Digital Signals. Applies the testing process with Apache JMeter tool in the implementation produced by the project within the Department VideoWeb Digital Signal, leaving these as just another example of using this new tool.

KEYWORDS

Software Quality, Tools, Automatic Tools, Jakarta Jmeter, Software Testing, Streaming Servers.

Tabla de contenido

ÍNDICE DE CONTENIDO

Índice de Figura

Índice de Tabla

Introducción

En el mundo actual se encuentran numerosas empresas productoras de software como Microsoft, Lunix, Apple, Sun Microsystem entre otras, las cuales desarrollan diferentes tipos de aplicaciones muy útiles para el desarrollo humano. Un ejemplo de estas empresas se encuentra en el país, la Empresa Cubana Nacional de Software más conocida por Desoft S.A, trabaja para lograr informatizar la sociedad de manera acelerada. Existe, por lo tanto, una real necesidad de que dichos software sean creados con la mayor calidad posible.

La calidad de software es el punto de partida para comenzar a elaborar un producto que se piensa comercializar, pues está enmarcada en cumplir con las necesidades del cliente y los requisitos planteados por el cliente a la hora de realizar el mismo. A lo largo de la historia, el término calidad ha sufrido numerosos cambios en cuanto a su evolución histórica, por eso se hace necesario tener bien presente su definición.

“Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente” (1).

Luis Fernández Sanz, director de Programación e Ingeniería del software de la Universidad Europea de Madrid (CEES) dice que es: “El desarrollo de software basado en estándares con la funcionalidad y rendimiento total que satisfacen los requerimientos del cliente.”

El aseguramiento de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisfará los requisitos dados de calidad (2). Este se diseña para cada aplicación antes de comenzar a desarrollarla y no después.

También es necesario destacar que sin la correcta Gestión de la Calidad de un Software el mismo no tendría sentido, es por eso que se aplica normalmente a nivel de empresa. Además puede haber una Gestión de Calidad dentro de la gestión de cada proyecto: “Es el conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implanta por medios tales como la planificación de la calidad, el control de la calidad, el aseguramiento (garantía) de la calidad y la mejora de la calidad, en el marco del sistema de calidad”.

Las pruebas de software son también un paso principal para lograr el éxito en un software, con la correcta realización de las mismas se puede alcanzar una aproximación de la calidad del producto, ellas son un conjunto de técnicas, métodos y herramientas que hacen posible el correcto funcionamiento del software ya sea en su etapa inicial, en pleno desarrollo o en su etapa final. Existe un gran interés en "¿Cómo probar software de manera eficiente?", lo que ha provocado un estudio profundo acerca del mismo.

Hacer un sistema requiere de mucho esfuerzo, probarlo requiere de más. En Cuba se cuenta con el centro de estudio universitario: La Universidad de las Ciencias Informáticas (UCI) la cual, además de impartir y enseñar a sus estudiantes, está enmarcada en todo el proceso de producción de software. En este centro de altos estudios se creó una Dirección de Calidad de Software que son los encargados de controlar la calidad del software a nivel nacional. En la facultad 6 se encuentra el Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED) que está formado a su vez por un personal capacitado en la calidad en cada uno de sus departamentos, el departamento de Señales Digitales como su nombre lo dice está enmarcado en llevar a cabo todo un proceso de innovación y desarrollo de la televisión en la universidad, el cual se ha extendido hasta la hermana República de Venezuela y está formado por los siguientes proyectos: Factoría, Plataforma de Trasmisión Abierta para Radio y Televisión (PTARTV), Primicia, Video Vigilancia Inteligente, Captura de Catalogación de Media (CCM), Video Web y DESCOMTEC.

En estos proyectos se ha estado llevando a cabo la tecnología de streaming para brindar la trasmisión de diferentes programas de televisión y radio. Streaming es una forma de transmitir información (generalmente multimedia) que permite al cliente ir consumiendo la información mientras que se está descargando, para lograr lo anteriormente dicho se han puesto en marcha los servidores streaming. El servidor debe almacenar el contenido y presentarlo en forma de streaming para que pueda ser consumido por los clientes.

Aquí se está poniendo en partida todo un proceso de aplicación de Pruebas de Software y dentro de las mismas, el uso de herramientas de Pruebas automatizadas de software que agilicen el desarrollo de aplicaciones. El software, después de ser creado, pasa por una serie de procesos que garantizan la calidad final del mismo: "Proceso de revisión del software". Este ciclo presenta una serie de deficiencias; ya que al producto se le realizan, en la mayoría de los casos, Pruebas manualmente, y en algunas

ocasiones los desarrolladores tienen muy poco tiempo para realizar un programa y por tanto se afecta el tiempo que se debe dedicar para probar las funcionalidades del producto, además la simulación de un escenario real aumenta el empleo de recursos. Otro problema relacionado con ello sería la precisión en cuanto a los resultados ya que el método manual puede traer consigo errores derivados del factor humano.

A pesar de que con el uso de herramientas de Pruebas automáticas de software se disminuye en el costo de la ejecución de las pruebas, el tiempo de realización de las mismas y se ahorra en conceptos de personal. En estos proyectos no se ejecutan este tipo de pruebas e incluso los jefes de proyectos no tienen todo el conocimiento de qué Pruebas automáticas pueden aplicarle a los productos antes de pasar por la fase de liberación. Tampoco existe un estudio sobre las Herramientas Automáticas de Pruebas de Software que les permita conocer a ellos cuáles son las más aplicables según la tecnología manejada en la producción.

Por lo anterior expuesto se determinó el siguiente **Problema Científico**: ¿Cómo garantizar que el proceso de pruebas de carga y estrés a servidores streaming sea eficiente?

El **objeto de estudio** de la investigación se centra en el Proceso de pruebas de carga y estrés a servidores streaming, siendo el **campo de acción** los Procesos de pruebas de carga y estrés de software a servidores streaming para el Departamento Señales Digitales.

El **objetivo general** del trabajo consiste en proponer una herramienta automática para pruebas de rendimiento de carga y estrés para el departamento Señales Digitales.

La **idea a defender** plantea que: la selección adecuada de una herramienta para pruebas de carga y estrés a servidores streaming permitirá un funcionamiento más eficiente de los mismos, en el Departamento Señales Digitales.

Con el objetivo de guiar, controlar y evaluar la investigación se definieron las siguientes **tareas:**

1. Delimitar los conceptos relacionados con la calidad de software.
2. Describir los tipos de pruebas automatizadas existentes en la actualidad, enfatizando en las pruebas de carga y estrés.

3. Caracterizar el estado actual de las herramientas que se utilizan para realizar pruebas de carga y estrés.
4. Identificar herramientas para pruebas de software automáticas a servidores streaming.
5. Validar la propuesta mediante la puesta en práctica en uno de los proyectos del Departamento Señales Digitales.

Posibles resultados que presupone la investigación:

- ✓ Herramienta para la realización de pruebas de rendimiento a servidores streaming.

Los **métodos teóricos** que se proponen para dar cumplimiento a las tareas y alcanzar los resultados son:

- ✓ **Analítico-Sintético:** Este método se ha empleado para poder resumir la gran bibliografía existente con respecto a los tipos de prueba, al proceso de calidad y la tecnología de streaming de la que se hace referencia en la investigación.

- ✓ **Análisis Histórico-Lógico**: Para dar seguimiento a la evolución de la calidad de software a través de los años y determinar las corrientes principales y tendencias actuales de la misma.

Así mismo se encuentran los **métodos empíricos**:

- ✓ **Entrevista**: Se realizó una entrevista a los líderes de proyectos pertenecientes al Departamento Señales Digitales con el fin de profundizar en el funcionamiento de estos servidores streaming y ver además como son aplicados en la universidad.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción

Este capítulo muestra los temas fundamentales que se abordan a lo largo de la investigación. Se mencionan conceptos relacionados con la Calidad de Software, pruebas y herramientas automáticas de pruebas, así como, otros aspectos relacionados con estos temas.

1.2 Calidad de Software

Para poder referirse al término de Calidad de Software es necesario remitirse al significado de calidad. Esta palabra puede ser interpretada de diversas maneras, ya que todo dependerá del nivel de satisfacción o conformidad del cliente. Sin embargo se puede decir que la calidad es el resultado de un esfuerzo arduo, que hace, aquel que trabaja de forma eficaz para poder satisfacer el deseo del consumidor, por lo tanto la calidad de un producto depende de la aceptación que éste tenga para el cliente.

En los libros se han puesto muchas definiciones de Calidad de Software. Por lo que a esta investigación respecta, la Calidad de Software se define como:

"Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente (1). "

La anterior definición permite hacer hincapié en tres puntos importantes:

- ✓ Los requisitos del software son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad.
- ✓ Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería del Software. Si no se siguen esos criterios, casi siempre habrá falta de calidad.
- ✓ Existe un conjunto de requisitos implícitos que a menudo no se mencionan (por ejemplo: el deseo por facilitar el uso y un buen mantenimiento). Si el software se ajusta a sus requisitos explícitos pero falla en alcanzar los requisitos implícitos, la calidad del software queda en entredicho (3).

A continuación se representará mediante una gráfica extraída del sitio web: "http://www.um.es" como se logra el alcance total de la calidad basado en la mejora de la misma en cuanto al tiempo:

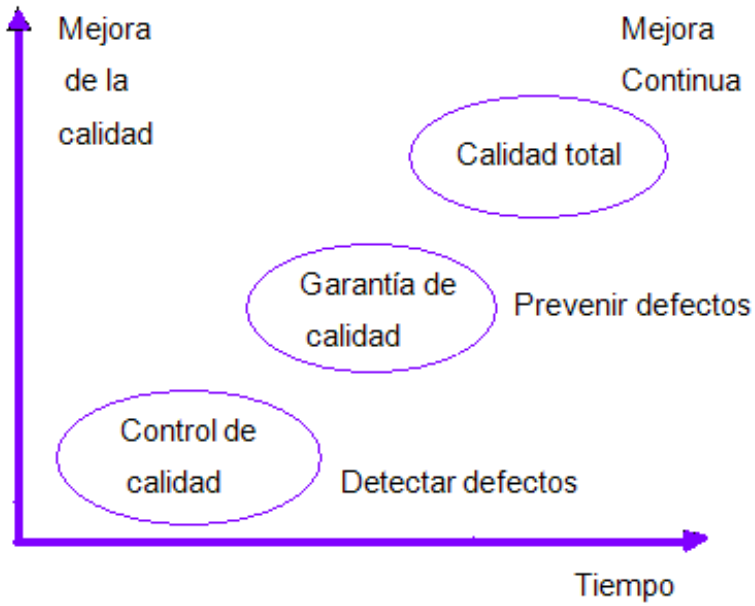


Figura #1: Alcance de la Calidad total mediante su Control y el Aseguramiento de su garantía

La adopción de una buena política contribuye en gran medida a lograr la calidad del software, pero no la asegura. Para el aseguramiento de la calidad es necesario su control o evaluación. Por lo que es fundamental tener bien definido qué es Gestión de la Calidad. "Aspectos de la función de gestión que determinan y aplican la política de la calidad" (4). Dentro de la Gestión de la Calidad se observa:

- ✓ **Gestión de la calidad de software (ISO 9000):** Conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades. Resulta de la aplicación a todas las perspectivas de procesos de software, productos, y recursos.

- ✓ **Política de calidad (ISO 9000):** Directrices y objetivos generales de una organización, relativos a la calidad, tal como se expresan formalmente por la alta dirección.

La Gestión de la Calidad se aplica normalmente a nivel de empresa, destacar también que puede haber una Gestión de Calidad dentro de la gestión de cada proyecto. Uno de los medios que implanta la Gestión de la Calidad es el Aseguramiento de la calidad este no es más que: "El conjunto de acciones planificadas

y sistemáticas necesarias para proporcionar la confianza adecuada de que un producto o servicio satisfará los requerimientos dados sobre la calidad" (4), de ahí que el Aseguramiento de la Calidad de Software está dado por: “El conjunto de acciones planificadas y sistemáticas necesarias para brindar la confianza en que el software requiere para cumplir con los requerimientos dados de calidad por parte del cliente.”

El aseguramiento de calidad del software está presente en (2):

- ✓ Métodos y herramientas de análisis, diseño, programación y prueba.
- ✓ Inspecciones técnicas formales en todos los pasos del proceso de desarrollo del software.
- ✓ Estrategias de prueba multiescala.
- ✓ Control de la documentación del software y de los cambios realizados.
- ✓ Procedimientos para ajustarse a los estándares (y dejar claro cuando se está fuera de ellos).
- ✓ Mecanismos de medida (métricas).
- ✓ Registro de auditorías y realización de informes.

Otros de los medios que implanta la Gestión de la Calidad es el Control de la Calidad: "Conjunto de técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad". Ahora cuando de Control de la Calidad del software se refiere puede decirse que son las: Técnicas y actividades de carácter operativo, utilizadas para verificar los requisitos relativos a la calidad, centrados en mantener bajo control el proceso de desarrollo y eliminar las causas de los defectos en las diferentes fases del ciclo de vida (4). A continuación se presenta un esquema extraído del sitio web: “http://synergix.wordpress.com”.

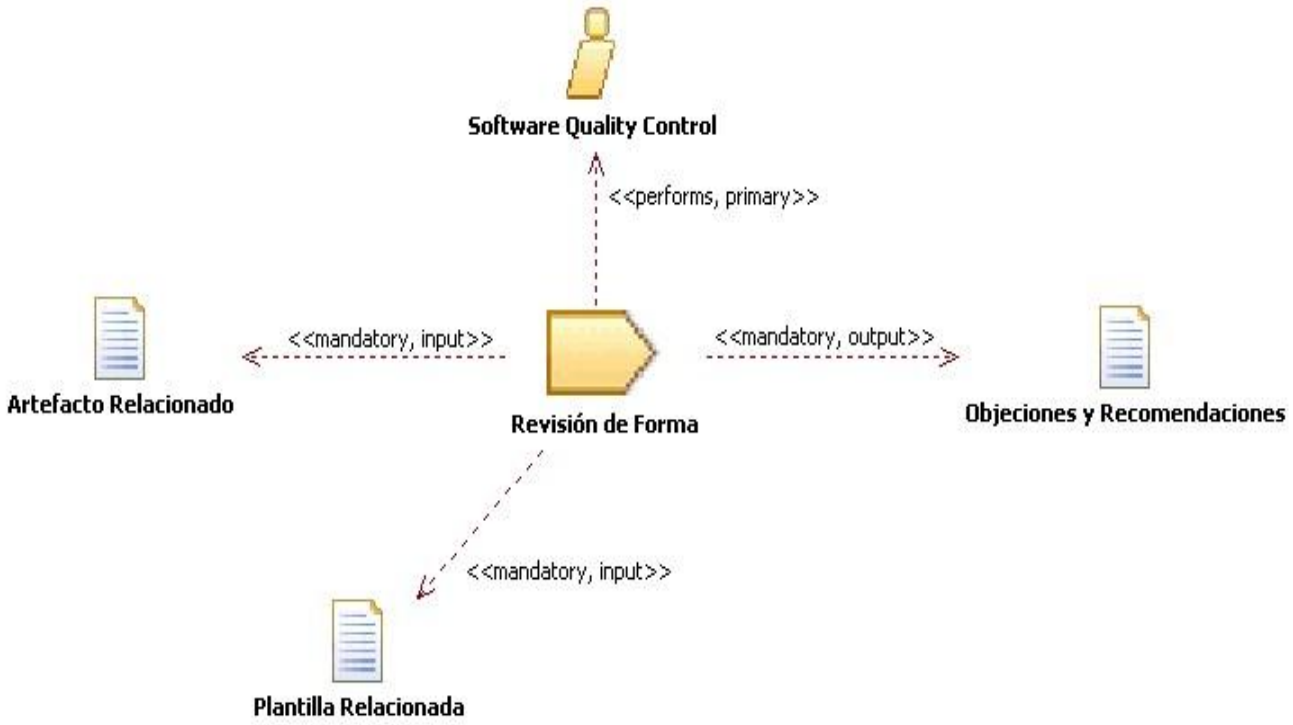


Figura #2: Control de Calidad de Software

El control de la calidad del software está centrado en dos objetivos fundamentales:

- ✓ Mantener bajo control un proceso.
- ✓ Eliminar las causas de los defectos en las diferentes fases del ciclo de vida.

En general, se puede decir que el control de la calidad del software son las actividades para evaluar la calidad de los productos desarrollados.

¿Qué es un Sistema de Gestión de la Calidad?

Sistema de gestión de la calidad: "Es la estructura de la organización, responsabilidades, procedimientos, procesos y recursos que se establecen para llevar a término la gestión de calidad" (4).

El sistema de calidad se debe adecuar a los objetivos de la calidad de la empresa. La dirección de la empresa es la responsable de fijar la política de calidad y las decisiones relativas a iniciar, desarrollar, implantar y actualizar el sistema de calidad (2).

Un sistema de calidad consta de varias partes:

- ✓ Documentación:

  Manual de calidad, es el documento principal para establecer e implantar un sistema de calidad. Puede haber manuales a nivel de empresa, departamento, producto, específicos (compras, proyectos).

- ✓ Parte física: locales, herramientas ordenadores entre otros.
- ✓ Aspectos humanos:

  Formación de personal.

  Creación y coordinación de equipos de trabajo.

Las ventajas de implantar un sistema de gestión de la calidad son las siguientes:

- ✓ Aumento de beneficios.
- ✓ Aumento del número de clientes.
- ✓ Motivación del personal.
- ✓ Fidelidad de los clientes.
- ✓ Organización del trabajo.
- ✓ Mejora de las relaciones con los clientes.
- ✓ Reducción de costes debidos a la mala calidad.
- ✓ Aumento de la cuota de mercado.

1.3 Pruebas de Software

En todo proceso de desarrollo de aplicaciones de software es indispensable la presencia de un proceso de Pruebas de Software que coexista y se integre con este primero para garantizar así el buen funcionamiento y la calidad del producto final. Para lograr lo antes expuesto se debe partir del concepto de que las mismas desempeñan un papel fundamental en esta disciplina.

Las Pruebas de Software no son más que el proceso de ejecutar un sistema o componente, para medir y mejorar su calidad, bajo condiciones específicas, con la intención de encontrar errores, observando y grabando los resultados, y haciendo una evaluación de algunos aspectos del sistema o componente en cuestión (5).

No siempre es necesario tener una aplicación funcionando para realizarle las evaluaciones pertinentes y lograr así su mejor funcionamiento, sino que si se lleva a cabo un proceso de prueba bien diseñado, que tenga pronosticado probar en cada una de las etapas del desarrollo, se eliminan muchos de los errores que posteriormente provocarían grandes gastos económicos, de tiempo y de esfuerzo humano.

Toda prueba de software desempeña un papel fundamental en el desarrollo de cualquier tipo de aplicación ya sea web o de escritorio, pero si se estudia la mejor forma de hacerlo, siguiendo los pasos de acuerdo con los especialistas en el tema, se incrementan las posibilidades de que esta llegue a un acertado final y arroje resultados más cercanos a los esperados, permitiendo así, realizar a posteriori un mejor análisis de la situación. Para ilustrar mejor esta situación, se presentan a continuación algunos objetivos, características y principios con que deben contar las Pruebas de Software.

1.3.1 Objetivos de las Pruebas de Software

El constante desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) han llevado a la sociedad a insertarse en un mundo, donde el impacto del Software (SW) ha revolucionado la forma de pensar y actuar de las personas con las que interactúan. La lucha de los grandes productores por ser cada día mejores, al ofrecer productos más baratos y asequibles a los usuarios, ha transformado en pocos años la forma de hacer y desarrollar las actividades que antes se hacían con otros programas y que ahora se realizan de forma más sencilla, rápida y menos costosa, mediante software de una alta calidad en el mercado mundial.

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces (6).

Hoy es una realidad que existen dificultades en los procesos de pruebas que se deben aplicar durante el desarrollo de aplicaciones web para lograr una satisfacción plena del cliente. Por lo que se definen algunos de los objetivos que debe perseguir todo diseño y ejecución de Pruebas de Software (6):

- ✓ Probar si el software no hace lo que debe.
- ✓ Probar si el software hace lo que no debe, es decir, si provoca efectos secundarios adversos.
- ✓ Descubrir un error que aún no ha sido descubierto.
- ✓ Encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles.
- ✓ Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.

1.3.2 Características de las Pruebas de Software

Roger Pressman (3), Kaner, Falk y Nguyen (7) sugieren algunas de las características que debe tener una «buena prueba»:

- ✓ Una buena prueba tiene una alta probabilidad de encontrar un error. El ingeniero de software debe tener un alto nivel de entendimiento de la aplicación a construir para poder diseñar casos de prueba que encuentren el mayor número de defectos.

- ✓ Una buena prueba no debe ser redundante. Uno de los objetivos de las pruebas es encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles, por lo cual no se deben diseñar casos de prueba que tengan el mismo propósito que otros, sino que se debe tratar de diseñar el menor número de casos de prueba que permitan probar adecuadamente el software y optimizar los recursos.

- ✓ Una buena prueba debería ser la mejor de la cosecha. La limitación en tiempo y recursos puede impedir que se ejecuten todos los casos de pruebas de un grupo de pruebas similares por lo cual

en estas situaciones se debería seleccionar la prueba que tenga la mayor probabilidad de descubrir errores.

- ✓ Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja.

Una buena prueba además debe ser capaz de encontrar errores que no se ven a simple vista. Conociendo estas características el probador debe tener una actitud de probar para romper, o sea, la habilidad de conseguir el punto de vista del cliente, así como ser paciente y tener la habilidad de saber encontrar los problemas y las omisiones. Esto le permitirá identificar aplicaciones de alto riesgo o con prioridad de riesgo, así como obtener los requisitos de forma clara.

1.3.3 Principios de las Pruebas de Software

En todo este proceso de software se tiene que tener en cuenta, algunos de los principios básicos que guían las pruebas del software, estos ayudan a mejorar y planificar las mismas siempre y cuando se haga un análisis exhaustivo de los mismos. Roger Pressman en el libro de Ingeniería de Software: “Un enfoque Práctico” (3), así como John Wiley y Myers en su libro: “El arte de las pruebas de software” (8) hacen alusión a algunos de estos principios:

- ✓ A todas las pruebas se les deberían poder hacer un seguimiento hasta los requisitos del cliente.
- ✓ Deberían planificarse mucho antes de que empiecen. La planificación de las pruebas puede empezar tan pronto como esté completo el modelo de requisitos y se tenga consolidado el modelo de diseño.
- ✓ Deberían empezar por «lo pequeño» y progresar hacia «lo grande». Las primeras planeadas y ejecutadas se centran generalmente en módulos individuales del programa. A medida que se avanza en ellas, el enfoque de las pruebas cambia en un intento de encontrar nuevos errores relacionados con la integración de estos módulos y finalmente con la interacción del sistema completo.
- ✓ No son posibles las pruebas exhaustivas. El número de permutaciones de caminos para incluso un programa de tamaño moderado es demasiado grande. Por lo cual, es imposible ejecutar todas las combinaciones de caminos durante las mismas. Sin embargo, es posible

elegir y ejecutar una serie de caminos lógicos importantes que permitan probar adecuadamente el software.

- ✓ Para ser más eficaces, deberían ser realizadas por un equipo independiente. El ingeniero de software que creó el sistema no es el más indicado para realizar las pruebas debido a que consciente o inconscientemente puede omitir casos de prueba importantes que conlleven a descubrir nuevos errores. Por consiguiente, es recomendable organizar un grupo de trabajo independiente para las mismas, que suministre una visión más objetiva del software.
- ✓ Se debe inspeccionar a fondo los resultados de cada prueba.
- ✓ Examinar un programa para ver que haga lo que se espera, es sólo la mitad de la batalla; la otra mitad es ver si hace lo que no se espera.
- ✓ No se debe hacer una planificación de los esfuerzos de prueba bajo la suposición de que no se encontrarán errores.
- ✓ La probabilidad de que existan más errores en una sección de programa es proporcional al número de éstos que se hayan encontrado en esa sección.
- ✓ Son una tarea extremadamente creativa e intelectualmente desafiante.

1.4 Procesos de Prueba de Software

Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software debe ir acompañado de una actividad que garantice la calidad. El proceso de Pruebas de Software es una parte importante en la obtención de productos con la calidad requerida y representa una revisión final de las especificaciones, del diseño y de la codificación.

Con el objetivo de encontrar el mayor número posible de errores, las pruebas deben conducirse sistemáticamente; y los casos de prueba deben diseñarse utilizando técnicas definidas. El software debe probarse desde dos perspectivas diferentes (8):

- ✓ La lógica interna del programa se comprueba usando técnicas de diseño de casos de prueba de caja blanca.
- ✓ Los requisitos del software se comprueban utilizando técnicas de diseño de casos de prueba de caja negra.

En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo.

El Proceso de Pruebas de Software se encuentra regido por los casos de prueba, que tienen el propósito o la intención de descubrir un error, un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces. La prueba demuestra hasta qué punto las funciones del software parecen funcionar, de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Además los datos que se van recogiendo a medida que se lleva a cabo la prueba, proporcionan una buena indicación de la fiabilidad del software y de alguna manera, indican la calidad del software como un todo. Pero, la prueba no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el software (3). En la actualidad existe gran cantidad de metodologías orientadas al proceso de desarrollo de software, siendo una de las más significativas Rational Unified Process (RUP).

RUP se caracteriza por ser Dirigido por casos de uso donde los casos de uso definen lo que el usuario desea a partir de la captura de requisitos y la modelación del negocio. Es Centrado en la Arquitectura, característica que brinda una visión completa del sistema, se describen los procesos del negocio que son más importantes, para comprenderlo, desarrollarlo y producirlo de una forma eficaz. Iterativo e Incremental donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo.

1.4.1 Flujo Trabajo de Prueba en RUP

En el flujo de trabajo de prueba se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros.

Los objetivos de las pruebas de acuerdo a lo planteado en la Metodología de Desarrollo de Software Rational Unified Process son:

- ✓ Planificar las pruebas necesarias en cada iteración, incluyendo las de integración y las de sistema. Las pruebas de integración son necesarias para cada construcción dentro de la iteración, mientras que las de sistema son necesarias sólo al final de la iteración.

- ✓ Diseñarlas e implementarlas, creando los casos de pruebas que especifican qué probar, creando los procedimientos de pruebas que especifican cómo realizarlas y creando, si es posible, componentes de pruebas ejecutables para automatizarlas.
- ✓ Realizar las diferentes pruebas y manejar los resultados de cada una sistemáticamente. Las construcciones en las que se encuentran defectos son probadas de nuevo y posiblemente devueltas a otro flujo de trabajo, como diseño o implementación, de forma que los defectos importantes puedan ser arreglados.

El Flujo de Trabajo de Prueba dentro de RUP está compuesto por el papel que éstas desempeñan en la vida del software, los artefactos que utilizan, como son: los modelos de pruebas; los casos de pruebas; los procedimientos; los componentes; los planes de prueba; los defectos y la evaluación de las mismas; además, se detallan los trabajadores involucrados en este proceso como son: el diseñador de pruebas, el ingeniero de componentes, el ingeniero de pruebas de integración y el ingeniero de pruebas del sistema. También se abordan los flujos de trabajo que este proceso contiene como: planificar las pruebas, diseñarlas, implementarlas, realizar las pruebas y por último, evaluar las pruebas.

Papel de la prueba en el ciclo de vida del software:

Durante la fase de inicio los ingenieros de prueba se van poniendo al corriente de la naturaleza general del sistema propuesto, van considerando qué pruebas requerirá y van desarrollando algunos planes provisionales de prueba. En la fase de elaboración el objetivo es asegurarse que los subsistemas de todos los niveles y de todas las capas funcionen, solo se pueden probar los componentes ejecutables. Sin embargo, las pruebas se llevan a cabo sobre todo en la fase de construcción, cuando el resultado de la implementación es sometida a pruebas de unidad, integración y de sistema. Esto quiere decir que la realización de las mismas se centra en las fases de elaboración, cuando se prueba la línea base ejecutable de la arquitectura, y de construcción, cuando el grueso del sistema está implementado.

RUP se divide en cuatro fases:

- ➢ Inicio (define el alcance del proyecto)
- ➢ Elaboración (definición, análisis, diseño)
- ➢ Construcción (implementación)
- ➢ Transición (fin del proyecto y puesta en producción)

Es importante mantener el modelo de prueba a lo largo del ciclo de vida del software completo, aunque el modelo de prueba cambia constantemente debido a:

- ✓ La eliminación de los casos de pruebas obsoletos (y los correspondientes procedimientos y componentes de las mismas).
- ✓ El refinamiento de algunos casos de pruebas en casos de pruebas de regresión.
- ✓ La creación de nuevos casos de uso para cada nueva construcción.

1.4.2 Artefactos del Flujo de Trabajo de Prueba

Los artefactos que RUP propone para llevar a cabo el proceso de prueba son los siguientes:

- ✓ **Modelo de pruebas:** describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. Además describe cómo han de ser probados los aspectos específicos del sistema; por ejemplo, si la interfaz de usuario es utilizable y consistente o si el manual de usuario del sistema cumple con su cometido.

- ✓ **Caso de prueba:** especifica una forma de probar el sistema, incluyendo la entrada o resultados con la que se ha de verificar y las condiciones para esto.

- ✓ **Procedimiento de prueba:** específica cómo realizar un caso de prueba, varios o parte de éstos. Por ejemplo, puede ser una instrucción para un individuo sobre cómo ha de realizar un caso de pruebas manualmente, o puede ser una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas para crear componentes ejecutables.

- ✓ **Componente de prueba:** automatiza un procedimiento de prueba, varios o parte de ellos. Puede ser grabado por una herramienta de automatización. Se utiliza para probar los componentes en el modelo de implementación, proporcionando entradas de prueba, controlando y monitorizando la ejecución de los componentes y, posiblemente, informando de los resultados de las pruebas.

- ✓ **Plan de prueba:** describe las estrategias, recursos y planificación de la prueba. Incluye la definición del tipo de prueba a analizar para cada iteración y sus objetivos, el nivel de cobertura de

ésta y de código necesario además del porcentaje de pruebas que deberían ejecutarse con un resultado específico.

- ✓ **Defecto:** es una anomalía del sistema y puede ser utilizado para localizar cualquier aspecto que los desarrolladores necesiten registrar como síntoma de un problema en el sistema que se necesita controlar y resolver.

- ✓ **Evaluación de prueba:** este artefacto se encarga de la evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, la cobertura de código y el estado de los defectos.

Durante las cuatro fases de RUP se integran las pruebas de software. Este proceso de prueba realiza una serie de actividades organizadas, como son: planificar las pruebas, diseñar pruebas, ejecutar pruebas y por último evaluar el resultado de las pruebas. Este conjunto de actividades son desarrolladas por cinco roles que intervienen en dicho proceso, como son: administrador de la calidad, diseñadores de pruebas, probadores, analista de prueba y revisor técnico. Seguidamente se detallan las funciones específicas que realizan cada rol.

- ➢ Administrador de calidad (a veces denominado asesor de calidad o director de calidad según el tamaño de la empresa): individuo con altos conocimientos y apto para instruir un equipo de desarrollo, este rol es capaz de:

    - ✓ Verificar que se cumpla el plan de pruebas planificado.
    - ✓ Asegurar que el producto cumpla con los requerimientos establecidos por el cliente. Determinar las pruebas que se ejecutarán
    - ✓ Monitorear la aplicación de las mismas.
    - ✓ Evaluar el resultado de las pruebas de software.

- ➢ Diseñador de pruebas: individuo con grandes conocimientos de la ingeniería de software, de metodología basada en componentes, de pruebas de software y de herramientas de pruebas, este es capaz de:

- ✓ Seleccionar las herramientas (ya sean automáticas o manuales) con las que se efectuarán las pruebas
- ✓ Diseñar los casos de prueba.

- Probador: individuo con conocimientos de metodología de desarrollo basada en RUP y de pruebas de software, este es capaz de:

  - ✓ Ejecutar las pruebas tal y como fueron planificadas por el administrador.
  - ✓ Ser un especialista en el empleo y uso de las herramientas seleccionadas por el diseñador para la ejecución de las pruebas.

- Analista de prueba: individuo con total dominio de las características del sistema creado en el proyecto, este es capaz de:

  - ✓ Identificar y delimitar las pruebas que se requieran.
  - ✓ Monitorear el progreso de las pruebas y el resultado en cada ciclo de prueba.
  - ✓ Evaluar la calidad total experimentada como un resultado de las actividades de prueba.

- Revisor técnico: individuo con conocimientos de el plan de revisiones y las listas de chequeos, este es capaz de:
  - ✓ Evaluar la calidad de los productos desarrollados en cuanto al cumplimiento del proceso de desarrollo del software.

1.5 Tipos de Pruebas Automatizadas

Para comenzar a hablar de las pruebas automatizadas es necesario el conocimiento de dos conceptos básicos (9):

- ✓ Prueba manual: “Es aquella prueba realizada por una o más personas que interactúan directamente con el sistema. Estas personas verifican si los resultados obtenidos son válidos o no”.

- ✓ Prueba automática: “Es aquella realizada por un programa o herramienta que prueba el sistema sin necesidad de la interacción de una persona. La herramienta suministra una serie de valores de

prueba, o acciones de prueba al sistema y verifica los resultados devueltos por éste con los resultados esperados".

Las Pruebas de Software se pueden ejecutar de cualquiera de estas dos formas. Toda forma automática trae grandes beneficios, pues simplifica el trabajo y el esfuerzo humano necesario para realizar cualquier tipo de tarea, además de aportar velocidad, eficiencia, precisión, aunque generalmente ante estas ventajas, se le anteponen ciertas desventajas, como por ejemplo, una gran cantidad de aplicaciones que contienen elementos que no son compatibles con las herramientas que se utilizan para ponerlas a prueba, en consecuencia esto requiere la búsqueda de soluciones creativas para que las herramientas se adapten a la aplicación, lo cual constituye un obstáculo al que se tendrá que enfrentar el equipo de trabajo.

La razón principal de hacer pruebas es garantizar que las cosas estén funcionando según lo solicitado, asegurando así la calidad, las pruebas encuentran los errores existentes, pero no es sólo eso, las pruebas automatizadas evitan que los errores sean introducidos. Mediante la realización de pruebas unitarias se pueden identificar y corregir los errores detectados con mayor rapidez, por lo que una prueba falla. Esta es una ventaja que las pruebas unitarias tienen sobre las pruebas de aceptación. Cuando una prueba de aceptación falla, significa que algún comportamiento esperado no está funcionando como fue solicitado. La prueba unitaria indica por qué este comportamiento esperado no está funcionando. Más adelante se explica en qué consisten estas pruebas (10).

Algunos de los objetivos de las pruebas automatizadas son (10):

- ✓ Las pruebas deben ayudar a aumentar la calidad
- ✓ Las pruebas deben ayudar a comprender el sistema que está siendo probado
- ✓ Las pruebas deben reducir (y no introducir) el riesgo
- ✓ Las pruebas deben ser fáciles de ejecutar
- ✓ Las pruebas deben ser de fácil lectura y de fácil mantenimiento
- ✓ Las pruebas deben requerir un mantenimiento mínimo, así como el sistema probado.

¿Por qué se realizan pruebas de software? (10).

- ✓ Para encontrar defectos ya sea en la fase inicial, en el desarrollo o en la fase final del software.
- ✓ Para validar todos los artefactos, no solo el código fuente.
- ✓ Para crear una suite de pruebas que le de confianza al equipo de desarrollo de seguir avanzando en el desarrollo de una aplicación.

Existen dos tipos de pruebas:

1. Funcionalidad de Software.
2. Rendimiento de Software.

1. Funcionalidad de Software:

Se realizan para verificar la funcionalidad de una unidad dentro de la aplicación (clases, componentes, módulos, etc.). Permiten detectar errores en el proceso de desarrollo.

Dentro de las pruebas de funcionalidad se encuentran (10):

- ✓ **Pruebas unitarias:** Permiten probar una unidad concreta (clase, componente, etc.). Estas pruebas aseguran que un componente produce una salida determinada para una entrada específica.
- ✓ **Pruebas funcionales:** Permiten validar una característica funcional completa, así mismo validan procesos y requieren un escenario específico de funcionamiento.
- ✓ **Pruebas de regresión:** Permiten comprobar que los cambios sobre un componente de la aplicación, no cambian el comportamiento ni generan errores en otros componentes de la aplicación. Se realizan apenas se introducen cambios en la aplicación, es necesario comprobar todos los componentes, no solo el componente modificado.
- ✓ **Pruebas de aceptación:** Son pruebas orientadas al cliente, ya que permiten verificar si se cumplen los requisitos funcionales de la aplicación.
- ✓ **Pruebas de integración:** Estas pruebas se realizan para verificar la integración con aplicaciones desarrolladas por terceros.

2. Rendimiento de Software:

Es aconsejable disponer de un entorno independiente para realizar estas pruebas, con condiciones similares al entorno de producción (10).

- ✓ Permiten verificar si la aplicación cumple los criterios de rendimiento.
- ✓ Comparan rendimiento de dos o más aplicaciones.
- ✓ Medir las cargas de trabajo soportadas por las aplicaciones.
- ✓ Establecer intervalos o umbrales de cargas en los que la aplicación tiende a fallar.

Es aconsejable disponer de un entorno independiente para realizar estas pruebas, con condiciones similares al entorno de producción, las cuales están enfocadas a los requisitos no funcionales del proyecto.

Dentro de las pruebas de rendimiento se encuentran (10):

- ✓ **Prueba de estabilidad:** Se realiza para determinar el comportamiento de la aplicación frente a una carga continua.
- ✓ **Pruebas de picos:** Se realiza para determinar la respuesta de la aplicación frente a los cambios súbitos de carga.
- ✓ **Pruebas de carga:** Se realizan para observar el comportamiento de una aplicación frente a una carga (peticiones, transacciones, etc.) esperada. Muestra los tiempos de respuesta a cada petición. Permite detectar los cuellos de botella en la aplicación. Verifican el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado y la capacidad del sistema para manejar volúmenes de datos extremos de acuerdo al tiempo de respuesta establecido para el sistema.

Estas pruebas de carga son de gran importancia en el mundo actual. En 1993 fue fundado RadView Software Lrd., es un proveedor líder de software de comprobación de rendimiento y carga. Sus productos de comprobación están instalados en más de 2000 sitios de clientes de todo el mundo, proporcionando a las compañías soluciones robustas y altamente eficientes para la comprobación del rendimiento, escalabilidad y confiabilidad de sus aplicaciones de internet. RadView es la elección inteligente para los requerimientos de comprobación de rendimiento y carga de las compañías de todos los tamaños. En abril del 2009 fue lanzado WebLOAD Professional es el software ganador del galardón RadView para la comprobación del rendimiento y carga de las aplicaciones de internet. WebLOAD Professional simula tráfico de hasta decenas de miles de usuarios y proporciona información detallada de cómo sus aplicaciones funcionan bajo esas cargas. El entorno avanzado de autoría en JavaScript de WebLOAD Professional permite la creación de scripts fieles a la realidad que utilizan una gran variedad de protocolos con facilidad. La sofisticada consola de run-time ofrece un seguimiento y análisis exhaustivo a través de la monitorización de las comprobaciones de carga en tiempo real. WebLOAD Analytics proporciona completas herramientas analíticas e informes profesionales para el análisis y elaboración de informes post-comprobación (11).

Pruebas de estrés

- ✓ Se realizan para asegurar que el sistema funciona como se espera bajo grandes volúmenes de carga.

- ✓ Consiste en sobrecargar la aplicación hasta que ésta falle.
- ✓ Permite determinar la solidez de la aplicación con cargas extremas.
- ✓ Permite testear el comportamiento de la aplicación en casos en que la carga real sea mayor a la esperada.

Las pruebas de estrés tienen una importancia considerable en el mundo, en la actualidad son muy utilizadas en grandes bancos. En julio del pasado año se estuvo haciendo test de estrés a los noventa y uno principales bancos europeos. Estas tuvieron como objetivo reducir la incertidumbre reinante de sobre la estabilidad de los sistemas financieros nacionales y fomentar la confianza entre los propios bancos. Muchas personas han dudado de que estas pruebas vayan a ser de gran ayuda en la estabilización del sistema financiero; sin embargo pueden ser importantes, si sus resultados no son tomados a la ligera. En primer lugar, miden de forma individual la capacidad de resistencia del capital de un banco ante un determinado impacto en su balance. En 2009 en EE.UU., los resultados de la prueba de resistencia fueron la base para establecer el nivel de recapitalización que se exigió a varias entidades, entre las que destacaba Bank of América (12).

1.6 Servidores streaming

Hace unos años atrás al cliente se le hacía difícil visualizar cualquier archivo ya sea de video o de audio en internet sin antes descargarlo. A razón de esto surge la tecnología de streaming en abril de 1995, utilizada para aligerar la descarga y ejecución de audio y video en la web, pues permite escuchar y visualizar los archivos mientras se están descargando. Este fue un gran avance en la tecnología, ya que cuando se pretende incluir audio o video en las páginas lo mejor es utilizar la tecnología streaming.

El proceso de streaming se realiza de la siguiente manera. Primeramente el cliente establece conexión con el servidor y éste le comienza a mandar el archivo. El ordenador (cliente) empieza a recibir el archivo y construye un buffer donde guarda la información. Luego cuando se ha llenado el buffer con una pequeña porción del archivo, el ordenador (cliente) inicia su trasmisión y al mismo tiempo continúa con la descarga. La tecnología posee la sincronización que permite la visualización del archivo mientras que este se descargue, de tal forma que cuando el archivo termina de descargarse el fichero también ha acabado de visualizarse. Si en algún momento el sistema sufre problema con la conexión es decir algún descenso de velocidad en la misma se utiliza la información que hay en el buffer. Si la comunicación se fragmenta por

mucho tiempo, el buffer se vacía y la puesta en marcha del archivo se fragmentaría hasta que se restaure la señal.

El proceso de streaming está puesto en marcha en muchas computadoras. Este se puede ver en la práctica mediante los programas como Media Player, Real Player, VLC u otros programas más que se instalan como plugins[1] en los navegadores para recibir y mostrar información generalmente en multimedia a través de la tecnología de streaming.

En principio no es necesario contar con un servidor especial para colocar archivos de audio o video con descarga streaming en la web. Cualquier servidor normal puede mandar la información y es el cliente (ordenador) el que se encarga de procesarla para poder mostrarla a medida que la va recibiendo. En determinados casos, como la puesta en marcha de una radio o la transmisión de un evento en directo, si que será imprescindible contar con un servidor de streaming al que se le mandará la señal y con ella, la enviará a todos los clientes a medida que la va recibiendo (13).

La transmisión de contenido multimedia a través de la web será cada vez más importante. La tecnología de streaming es un mercado con futuro y grandes compañías como Microsoft, Real Networks entre otras; ya están luchando por el mercado. La velocidad de Internet aumentará con el tiempo y con ella aumentará la calidad de las transmisiones, para hacer posible tanto la radio como la televisión en Internet (13).

1.7 Conclusiones del Capítulo

Este capítulo ha demostrado la importancia que tienen las pruebas de software tratando temas relacionados con las mismas como conceptos, principios, características y objetivos. Aquí se ha visto también las ventajas y desventajas de las pruebas automatizadas, así como los tipos de pruebas automatizadas que existen haciendo énfasis en las pruebas de carga y estrés, se vio también la relación existente entre gestión, aseguramiento y control de la calidad, además de brindar un pequeño resumen sobre el estado actual de los servidores streaming a nivel mundial. Todo lo abordado en este capítulo dará una visión a la hora de caracterizar el estado actual de las herramientas que se utilizan para realizar pruebas de carga y estrés.

[1] Plugins: es simplemente una herramienta que sirve para accionar otra, es como el motor de arranque de un auto, es un pequeño motor eléctrico que se usa para poner en funcionamiento el motor del auto.

Capítulo 2: Descripción de la Propuesta de Solución

2.1 Introducción

Una manera de optimizar un proceso de Pruebas de Software es la introducción de herramientas que ayudan a agilizarlo y hacerlo menos tedioso. En el presente capítulo, se abordará el estado actual de las herramientas utilizadas para realizar pruebas de carga y estrés. Dada la existencia de un gran número de dichas herramientas el equipo de pruebas necesita criterios sobre los cuales establecer una comparación y así poder seleccionar el software de prueba que más se adapte a sus necesidades. También se describe la forma de selección de dicho software, teniendo en cuenta las características de los servidores streaming.

El uso de herramientas de testeo de software puede simplificar dramáticamente las pruebas, aumentar la tasa de encontrar defectos y, finalmente, lograr una calidad con mayor liberación. Más allá de esto puede conducir mejoras en la fiabilidad de las soluciones entregadas haciéndolos más productivos y eficaces desde la perspectiva de los clientes.

En la actualidad existen numerosas herramientas con libre acceso a la hora de adquirirlas, sin embargo las herramientas propietarias son un baluarte fundamental a la hora de realizar las pruebas de software, su privacidad es lo que las hace un producto sumamente robusto. Esencialmente las herramientas de prueba deben encajar y mejorar los actuales procesos de negocio. Desde luego, no deberían obligar a un proceso de cambio en el equipo de pruebas.

2.2 Ventajas y desventajas del uso de herramientas para automatizar Pruebas de Software

Para comenzar el tema de las pruebas automatizadas es necesario conocer una serie de ventajas y desventajas que conduce al uso de este recurso, con el objetivo de compendiar[2] o no diferentes alternativas y vías de solución a problemas que puedan existir. Estas son:

Ventajas:

[2] Compendiar: Sinónimo de resumir, abreviar, simplificar, seleccionar.

- ✓ Rapidez en la ejecución de pruebas de regresión. En todas las metodologías de desarrollo de software el ciclo codificación-prueba-corrección se repite un número ilimitado de veces. La prueba de una funcionalidad específica tiene que ser ejecutada muchas veces durante el desarrollo, esto impide que ante la corrección de errores encontrados, otros no sean introducidos. Este proceso de repetición de pruebas es conocido como pruebas de regresión. Esta práctica es fundamental debido a las modificaciones que sufren los sistemas durante su elaboración. Si esta tarea se encuentra automatizada debido a su previa ejecución en versiones anteriores del producto, sólo se necesita seleccionarla y volverla a ejecutar con un mínimo esfuerzo manual.

- ✓ La ejecución de un mayor número de pruebas en una unidad de tiempo finita. Esta característica hace que el producto final sea mucho más confiable y por supuesto tenga mucha más calidad.

- ✓ Simulación de condiciones reales de explotación. Existen pruebas de vital importancia que no es factible realizarlas de forma manual. Tomando como ejemplo una prueba de estrés en la que se quiere simular el uso de una aplicación por 500 usuarios de manera concurrente. Como es de suponer es costoso reunir 500 computadoras y 500 personas para realizar esta labor, sin embargo existen herramientas con las que se simula esta situación.
- ✓ Programación de la ejecución de pruebas en horarios no laborales como la noche y los fines de semana. Muchas herramientas brindan prestaciones que permiten la programación automática de un sinnúmero de pruebas, utilizando así el horario de trabajo para idear posteriores planes de prueba.

Desventajas:

- ✓ Por lo general una gran cantidad de aplicaciones contienen elementos que no son compatibles con las herramientas que se utilizan para ponerlas a prueba, en consecuencia esto requiere la búsqueda de soluciones creativas para que éstas se adapten a la aplicación, lo cual constituye un obstáculo al que se tendrá que enfrentar el equipo de trabajo.
- ✓ Poco conocimiento de lenguajes de scripting por parte de los probadores. Éste puede ser un punto determinante en el proceso de automatización de pruebas, pues una gran cantidad de herramientas utilizan estos lenguajes para el mantenimiento de las mismas.

- ✓ Si se automatiza el caos sólo se logra un caos mucho peor. No es recomendable la práctica de las pruebas automáticas por equipos que tengan mal organizada su realización. Ejemplos de malas prácticas son: poca o inconsistente documentación, pruebas que no son muy efectivas en la búsqueda de defectos.
- ✓ Problema del mantenimiento. Cuando los sistemas sufren algún tipo de cambio, como ya se ha visto anteriormente, las pruebas también tienen que cambiar. El esfuerzo empleado en la actualización de las mismas es a veces causa de abandono de la iniciativa de su automatización y de un retorno a la ejecución manual.

2.2.1 Descripción de Herramientas a comparar

Antes de entrar a comparar las siguientes herramientas, es necesario hacer una pequeña descripción de las mismas así como destacar las principales características de cada una de ellas.

NeoLoad

NeoLoad es una herramienta de prueba de carga y estrés para aplicaciones Web. Esta herramienta prueba la robustez y rendimiento del servidor bajo distintas cargas. Brinda un informe claro y completo de la prueba, marcando los errores y cuellos de botella.

NeoLoad responde a las siguientes preguntas:

- ✓ ¿Cuántos usuarios pueden soportar el sitio?
- ✓ ¿Sobrevendrá un fallo por sobrecarga en el sitio?
- ✓ ¿Cuáles son los tiempos de respuestas?

NeoLoad prueba la aplicación, no importa lo compleja que sea ésta. Por ejemplo, extrae dinámicamente enlaces y datos mientras la prueba se está ejecutando para inyectarlos en el escenario.

Características principales

- ✓ Grabadora basada en el navegador.

- ✓ Monitorización de Sistema. Obtiene estadísticas de los servidores (CPU, Memoria) en Windows, Linux, Solaris, dispositivos SNMP, Apache, RSTAT, Oracle, Tomcat.
- ✓ Presenta sustitución de datos.
- ✓ Diseño de escenarios dedicados SOAP y Webservices
- ✓ Presenta un escenario avanzado: utilizando acciones lógicas: if...then...else, while...
- ✓ Extracción de datos: extrae enlaces y formularios generados.
- ✓ Validación de contenido: Revisa el contenido de la página bajo carga
- ✓ Informes: Genera un informe de prueba (HTML/Word/PDF/XML).
- ✓ Es multiplataforma (Windows, Linux, Solaris) y puede generar una gran cantidad de tráfico usando generadores de carga adicionales.

Webserver Stress Tool

Webserver Stress Tool simula grandes números de usuarios accediendo a un sitio web vía HTTP/HTTPS. El software puede simular hasta 10000 usuarios que independientemente van recorriendo su camino a través de una serie de URLs. Permite patrones de URL simples, así como patrones complejos, a través de un archivo de Script. Basado en los parámetros que especifica, la aplicación no sólo pide el HTML de una URL, sino también frames[3], imágenes, archivos de Flash, etc. Cada usuario es simulado con un hilo de ejecución separado con su propia información de sesión (ej. las cookies para cada usuario simulado se guardan separadamente) y "navega" las URLs independientemente de los otros usuarios tal como en un uso real.

Características principales

[3] Frames: Un frame es una especie de marco o recuadro independiente en el que se puede cargar una página web.

- ✓ Maximiza el tiempo de disponibilidad.
- ✓ Resuelve situaciones críticas para el desempeño del servidor antes de que algún problema lo tire.
- ✓ Maximiza el desempeño.
- ✓ Asegura que los sitios web y aplicaciones cuenten con los recursos que necesitan cuando los necesitan.
- ✓ Se instala en 5 minutos y funciona con cualquier servidor web.

WebLoad Professional

Es un software para pruebas de rendimiento de aplicaciones de Internet. Instalado en más de 2000 empresas, WebLOAD Professional es la mejor solución probada en campo para probar la capacidad de una aplicación de Internet para manejar las demandas de uso de los clientes reales y puesta en marcha con la confianza.

Posee una potente solución por lo que brinda una gama completa de capacidades de prueba de carga:

1. **Pruebas de rendimiento** de las funciones de prueba, el tiempo de respuesta de una petición que se realiza mientras el sistema está bajo carga.
2. **Escalabilidad** probar las funciones de determinar los parámetros para la planificación de las adquisiciones de hardware en el contexto de aumento de solicitudes simultáneas.
3. **Ensayos de fiabilidad** revela problemas potenciales derivados de carreras extendida.

Características principales

- ✓ WebLOAD ofrece la eficiencia del hardware, pues se requiere menos hardware para simular más usuarios.
- ✓ Simula tráfico de hasta decenas de miles de usuarios.
- ✓ Proporciona información detallada de como sus aplicaciones funcionan bajo esas cargas.
- ✓ Proporciona completas herramientas analíticas e informes profesionales para el análisis y elaboración de informes post-comprobación.

- ✓ Soporta una amplia gama de protocolos de Internet que van desde HTTP / S, WAP, SOAP y Ajax para RTSP / RTP, FTP, SMTP y TCP / IP.

QALOAD de Compuware

QALoad es un software de pruebas de rendimiento para aplicaciones basadas en servidor y bases de datos. Simula la actividad de los usuarios asociados con el despliegue de cliente / servidor, basada en medios UNIX y las aplicaciones alojadas sin afectar de una organización para el usuario final.

Características principales

- ✓ Simula miles de usuarios al mismo tiempo realizando diferentes operaciones.
- ✓ Crea scripts de pruebas que se utilizan para simular la aplicación.
- ✓ Permite crear las cargas del sistema predecible y repetible.
- ✓ Permite visualizar los tiempos de respuestas de la medida desde la perspectiva del cliente.
- ✓ Permite recoger y analizar información sobre el sistema de respuesta.

2.2.2 Tabla comparativa

A continuación se aprecia una tabla que resume una parte del estudio realizado, con el objetivo de mostrar claramente las docilidades de algunas de estas herramientas y facilitar la comparación entre las mismas.

Los aspectos a tener en cuenta para realizar dicha comparación son**:**

- ✓ **Tipos de pruebas:** se analizan los tipos de pruebas que se pueden automatizar con la herramienta. Es una medida importante pues permite conocer si la herramienta se acomoda al enfoque de automatización que se espera obtener, o sea, para ver si es útil según el plan de automatización elaborado.

- ✓ **Modo de adquisición:** se analiza la manera en que se puede acceder a la herramienta, si es de forma gratuita o bajo el pago de alguna licencia.

- ✓ **Tipo de Reportes:** se analiza la forma en que la herramienta presenta las salidas de las pruebas realizadas, sobre todo, hacia qué formatos exporta esta información. Por ejemplo si permite

hacerlo en forma de texto, de tablas, de gráficos, hacia formatos amigables como son: pdf, doc, xls, HTML. Ésta es una característica importante a analizar, sobre todo en las herramientas que permiten realizar Pruebas de Estrés.

- ✓ **Integración con entornos de desarrollo:** se examina si la herramienta brinda la posibilidad de integrarse con uno o varios medios de desarrollo, pues a partir del que sea usado por el equipo de trabajo se escogen las herramientas para la realización de las pruebas.
- ✓ **Soporte para AJAX:** analiza si la herramienta cubre la necesidad de localizar defectos en sistemas implementados con este tipo de tecnología, pues dada su novedad se hace de vital importancia este análisis.
- ✓ **Bibliografía:** se analiza la existencia de materiales de estudio para el conocimiento acerca del uso de la herramienta.
- ✓ **Multiplataforma:** se analiza si la herramienta funciona en diferentes sistemas operativos y/o ordenadores.

En la tabla se hace alusión a 6 herramientas de pruebas, es válido destacar que existen en la actualidad muchas más, pero las escogidas son a consideración de esta investigación, las más completas.

Tabla 1: Comparación de las herramientas de pruebas

Aspectos/Herramientas	Jakarta JMeter	WebLOAD Professional	NeoLoad	QALOAD de Compuware	Webserver Stress Tool	PushToTest TestMaker
Tipos de Prueba que realizan.	Funcionales, carga, estrés y regresión.	Rendimiento y carga.	Funcionales, carga, estrés y regresión.	Carga, estrés y Monitoreo de la aplicación	Rendimiento, estrés, carga y disponibilidad.	Funcionales, carga, estrés, volumen, regresión y monitoreo de la aplicación.

Tipo de Reportes.	Tablas de resultados y gráficos.	Consola, informes personalizados y plantillas predefinidas.	Genera un informe de prueba (HTML/Word/ PDF/XML).	Informes y gráficas.	Registros detallados de prueba y varios gráficos de fácil lectura.	Reporte HTML que contiene una serie de gráficos y tablas.
Soporte de Ajax.	Sí	Sí	Sí	Sí	No	Sí
Modo de Adquisición.	Gratuita.	Gratuita.	Privativa.	Compra de una licencia.	Compra de licencia.	Gratuita.
Integración con entornos de desarrollo.	No	No	No	No	No	No
Multiplataforma.	Sí	Sí	Sí	Sí	No	Sí
Bibliografía.	Amplia documentación.	Amplia documentación.	Abundantes libros mediante pago de licencias.	Abundantes tutoriales mediante pago de licencia.	Los libros se obtienen mediante el pago de la licencia.	Amplia documentación en el sitio de la aplicación.

Analizando el resultado del estudio de las herramientas, se llegó a la conclusión, de que las herramientas que no son de libre distribución ofrecen una serie de características adicionales que las hacen mucho más sólidas, como son las generaciones de casos de prueba o los diferentes tipos de reportes que poseen, por señalar algunos ejemplos. Aunque siempre tendrán como principal restricción su modo de adquisición.

Enfoque de la investigación

La investigación está enfocada en los proyectos pertenecientes al Departamento Señales Digitales, por lo tanto es de vital importancia el conocimiento de las características de los software que tienen que ver con

la trasmisión de flujos de datos a partir de la tecnología de streaming, ya que esto sería un punto importante a tener en cuenta durante el proceso de propuesta de la herramienta para la realización de las pruebas automáticas.

2.3 Características de los servidores streaming en los proyectos

Los servidores streaming existentes en los proyectos pertenecientes al Departamento Señales Digitales tienen características particulares, sobre las cuales se debe tener conocimiento a la hora de realizar una selección de herramienta para la realización de pruebas automatizadas. Las siguientes características ayudan a realizar este proceso de selección.

- ✓ **Servicio:** El tipo de servicio que brindan los servidores streaming son web y desktop.
- ✓ **Complejos:** Exhiben un alto grado de concurrencia.
- ✓ **SGBD :** El Sistema de Gestión de Base de Datos utilizado por estos es el PostgreSQL versión 1.12.
- ✓ **Intensivos en datos:** Manejan grandes volúmenes de datos.
- ✓ **Sistema operativo:** Funcionan bajo GNU/Linux, distribución Ubuntu 9.4 en adelante.
- ✓ **Durabilidad:** El tiempo de explotación es largo.

2.4 Proceso de Selección de la Herramienta

En el proceso de selección de la herramienta se evalúa cuál es la apropiada para las necesidades del equipo de trabajo. Para que el proceso tenga éxito no se puede comenzar la búsqueda centrándose en las herramientas que existen, lo primero y más importante son: tipos de prueba empleadas y característica de los servidores streaming en los proyectos, las cuales fueron obtenidas a través de una entrevista realizada a los líderes de cada proyecto pertenecientes al Departamento Señales Digitales.

Para la realización de la selección de la herramienta se debe tener en cuenta también, que no se cuenta con el conocimiento de ninguna herramienta que sea capaz de realizar pruebas a aplicaciones web y desktop, por lo que la selección propone el uso de una herramienta que sea capaz de realizar pruebas a aplicaciones web. Además se realiza especial énfasis en aquella que sea software libre y gratis. Aunque nunca se debe dejar pasar por alto la calidad de las herramientas que no son de libre distribución, ya que

las mismas presentan una mayor bibliografía, brindan mayor facilidad para la realización de las pruebas e incluso, algunas poseen la funcionalidad de creación de casos de prueba, lo que constituye un elemento que carecen, las herramientas libres encontradas.

Una incorrecta elección de la herramienta puede ocasionar consecuencias indeseables que impactan de forma significativa en la realización de las pruebas, un ejemplo de esto son los problemas técnicos que surgen a la hora de hacer funcionar la herramienta en el medio, los usuarios encuentran dificultad en su uso, lo cual retrasa su trabajo lejos de hacerlo más ameno.

2.4.1 Resultados de la Entrevista realizada a los Líderes de Proyectos pertenecientes al Departamento Señales Digitales

Por la necesidad de tener conocimiento de las particularidades de los servidores streaming empleados en los proyectos pertenecientes al Departamento Señales Digitales para tener en cuenta estas características en el proceso de selección de la herramienta, así como, la necesidad de conocer el nivel de conocimiento y empleo de herramientas automáticas para Pruebas de Software de carga y estrés se realizó una entrevista a los líderes de estos proyectos.

La entrevista constó de diez preguntas previamente elaboradas (Ver Anexo #1), en esta se hace alusión al uso de servidores streaming en los proyectos, al tipo de servicio que brindan estos, al grado de concurrencia de los servidores streaming y SGBD (Sistema de Gestión de Base de Datos) empleados en el proyecto, además se basó en saber si estos servidores son intensivos en datos y conocer su durabilidad, ver también bajo qué sistema operativo funcionaban, y sobre la metodología de desarrollo que se empleaba en los proyectos y si se le han hecho prueba a estos servidores estar al tanto si utilizaron o utilizan herramientas automáticas para realizar estas pruebas. De esta manera quedó creada la entrevista que posibilitó la obtención de la información necesaria para la realización de la propuesta de solución.

Teniendo en cuenta todo lo anteriormente planteado, se plasmará una propuesta de herramienta que se espera sea utilizada por los líderes de proyectos para la realización de sus pruebas automáticas. La selección se realizará teniendo en cuenta las características que presentan los servidores streaming y a su vez los resultados de la entrevista realizada, por ser estos aspectos primordiales para la realización de la misma.

2.5 VideoWeb

Con la entrevista realizada al jefe del proyecto VideoWeb, se pudo comprender que dicho proyecto es una solución encaminada a la gestión y publicación de archivos multimedia. Los materiales a gestionar dentro del entorno web están clasificados según su Tipologías, ya sea Videos musicales, Documentales, Materiales internos, Materiales docentes, Películas, Series y dentro de cada una de ellas existe una clasificación interna. Los archivos gestionados aquí son enviados al servidor de medias para posteriormente publicarlos. El proyecto está compuesto por dos subsistemas:

- ✓ Subsistema de Administración.
- ✓ Subsistema de Gestión y presentación de contenido.

De los anteriormente mencionados se le aplicará la prueba al Subsistema de Gestión y presentación de contenido, el cual cuenta con veintiocho casos de uso, de estos se le realizará la prueba a los más críticos los cuales son: Crear publicación de archivo multimedia, Eliminar publicación de archivo multimedia, Modificar publicación de archivo multimedia y Publicar archivo multimedia.

2.6 Propuesta de Herramienta

Al llevar a cabo la incorporación de la herramienta para pruebas automatizadas a servidores streaming es necesario tener en cuenta todos los aspectos mencionados en los epígrafes anteriores, pues éstos constituyen la base para lograr este objetivo. Como conclusión de un estudio realizado en materia de herramientas para la automatización de pruebas, las características de los servidores streaming empleados en los proyectos pertenecientes al Departamento Señales Digitales y los resultados de la entrevista realizada a los líderes de los proyectos del mismo, se propone el uso de una herramienta para la realización de pruebas automáticas, por las funcionalidades que brinda es la que mejor se adapta al medio de desarrollo de los proyectos productivos pertenecientes al Departamento Señales Digitales.

La herramienta automática a seleccionar pertenece al movimiento del software libre. La propuesta no comprende la automatización de las etapas de Requisitos y Análisis-Diseño, pues no se cuenta con herramienta que automaticen la revisión de los requisitos, ni el proceso de generación de casos de pruebas funcionales a partir de la descripción expandida de los casos de uso.

2.6.1 Pruebas No Funcionales

Durante el proceso de investigación y búsqueda de herramientas se obtuvo como resultado que la mayoría de ellas son de software propietario. Siendo PushToTest TestMaker y JMeter las aplicaciones de software libre y de libre distribución más completas encontradas.

PushToTest TestMaker es una aplicación de comprobación de servicios Web de PushToTest. Requiere Java 1.4 (o más reciente) para funcionar. TestMaker está diseñado para probar las aplicaciones modernas de Internet, incluyendo aplicaciones dinámicas de Internet (RIA, utilizando Ajax, Flex, Flash) y Arquitectura Orientada a Servicios (SOA). Ejecuta pruebas de millones de usuarios virtuales a muy bajo costo, posee un servicio de vigilancia que se encarga de monitorear el funcionamiento de la aplicación web cada un tiempo determinado por el usuario. Posee gráficas para la visualización de los resultados de los test, así como un monitoreo del funcionamiento de los recursos (RED, CPU, Memoria) de los nodos que realizan la prueba.

JMeter es un software de código abierto, creado 100% en Java destinado a carga funcional, comportamiento de prueba y medición del rendimiento. Originalmente fue diseñado para probar las aplicaciones Web, pero desde entonces se ha expandido a otras funciones de prueba.

Apache JMeter puede ser utilizado para probar el rendimiento tanto en los recursos estáticos y dinámicos (archivos, Servlets, Perl, Java Objects, bases de datos y consultas, servidores FTP y mucho más). Puede ser utilizado para simular una carga pesada en un servidor, de red o un objeto para poner a prueba su resistencia o para analizar el rendimiento general en diferentes tipos de carga. Puede usarlo para hacer un análisis gráfico de rendimiento o para probar su servidor / script / comportamiento del objeto con carga pesada concurrentes.

JMeter puede probar muchos tipos de servidores (Web – HTTP, HTTPS, SOAP, LDAP, JMS, Base de datos a través de JDBC). Presenta un marco completo que permite el muestreo simultáneo de muchas discusiones y toma de muestras simultáneas de diferentes funciones de los grupos de hilos separados. Además un cuidadoso diseño de interfaz gráfica de usuario que permite una operación más rápida y más precisa en cuanto a tiempo.

La propuesta de la herramienta elegida para esta investigación fue **JMeter.** Pues ésta herramienta se destaca por su versatilidad y estabilidad dentro de las herramientas de libre distribución. Además facilita a una rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo. La principal razón por la cual se seleccionó fue por presentar una estructura en árbol que le da potencia, permitiendo que sea la imaginación de quien la use la que ponga los límites a la hora de diseñar el plan de prueba, brindando mayor cantidad de variantes para recoger los resultados obtenidos, que el resto de las herramientas de libre distribución, lo que permite hacer un análisis exhaustivo de las pruebas realizadas.

2.6.1.1 Instalación de Apache JMeter

Para la instalación de esta aplicación es preciso tener instalado previamente en la computadora la máquina virtual de java luego:

- ✓ Descargar JMeter en: http://jakarta.apache.org/jmeter.
- ✓ Descomprimir el archivo Tar de JMeter en un directorio temporal (/temp por lo general) a través del comando: Jakarta-jmeter-<numero_de_versión>
- ✓ El paso anterior genera un directorio por nombre Jakarta-jmeter-<numero_de_versión>, dentro del cual se encuentran los diversos componentes de JMeter, descritos a continuación.

La carpeta de instalación de Apache JMeter está compuesta por los siguientes archivos:

- ✓ **bin:** este directorio contiene los ejecutables utilizados por JMeter, tanto para ambientes Linux (jmeter) así como Windows (jmeter.bat).
- ✓ **docs:** contiene documentación acerca de JMeter.
- ✓ **printable_docs:** contiene documentación en modalidad de impresión acerca de JMeter.
- ✓ **lib:** este directorio contiene los archivos Jars empleados por JMeter requeridos en cualquier modalidad.

La ejecución de JMeter es llevada a cabo del directorio bin, que contiene las siguientes opciones:

- ✓ **jmeter | jmeter.bat:** Corresponde a los ejecutables de la interfaz principal para plataformas *nix y Windows, respectivamente.

- ✓ **jmeter-server | jmeter-server.bat:** Representan los ejecutables para el emulador de Servidor JMeter para plataformas Linux y Windows, respectivamente.
- ✓ **jmeter.properties:** Contiene propiedades de arranque para JMeter que son utilizadas por cualquiera de sus ejecutables.
- ✓ **jmeter.log:** Representa los registros ("logs") generados al ejecutar JMeter.
- ✓ **users.xml | users.dtd:** Un archivo XML y su correspondiente DTD, empleados para definir características de usuarios que serán simulados por JMeter.
- ✓ **ApacheJMeter.jar:** Archivo JAR que contiene las principales clases de JMeter.

2.6.1.2 Utilización de Apache JMeter

Para la utilización de esta herramienta se debe contar con una PC con característica de 1GB de RAM y la instalación de la máquina virtual de java o superior instalada en el sistema operativo. Las especificaciones de recursos deben ser entregadas al administrador del sistema en los días previos a la concepción de los scripts de prueba.

Los scripts de prueba se pueden realizar de diferentes formas en dependencia de la importancia del sistema que se desea probar. En una aplicación web donde solo se modele la navegación de los usuarios se pueden realizar los scripts de prueba simples o pruebas de navegación. En sistemas más complejos experimentados en el manejo de datos y acciones de interacción con otros periféricos como puede ser un software de gestión, se recomienda utilizar el tipo de script que realiza pruebas más complejas, donde se modela la interacción del usuario con la aplicación mediante la entrada de datos.

Componentes

Para la realización del plan de pruebas que define los scripts de prueba, se definen un conjunto de elementos los cuales serán utilizados en dependencia a la acción que se quiera realizar.

Los elementos pueden ser:

- ✓ **Elementos jerárquicos:**
    - Listeners (elementos en escucha)
    - Elementos de configuración

- Post-procesadores
- Pre-procesadores
- Aserciones (afirmaciones)
- Temporizador (cronómetros)

✓ **Elementos ordenados:**

- Controladores
- Samplers (agentes de pruebas)

El plan de pruebas que se crea en la herramienta se va realizando sobre la base de una **lista ordenada** de peticiones (Peticiones http) utilizando **Samplers,** que representa los pasos a ejecutar en el plan. Normalmente, las peticiones se organizan dentro de controladores lógicos. Algunos tipos de controladores afectan el orden de ejecución de los elementos que controlan, lo cual contrasta con lo real que se espera alcanzar en la simulación. Apoyándose en los Samplers, se pueden realizar peticiones a determinado servidor, también pueden ser configurados a través de los "Elementos de configuración".

Los controladores lógicos funcionan de manera diferente. Estos permiten controlar el comportamiento de la prueba, en ellos se modelan las decisiones en función a las necesidades del plan de pruebas. Por ejemplo, un controlador **If Controller** permite decidir si realizar o no una petición http en función de una condición. Cada controlador, puede tener uno o más elementos por defecto.

Utilizando el elemento **Grupo de Hilos** que se muestra en la figura #3, se inicia la cabecera de la prueba en la que se definirán la cantidad de usuarios que se modelarán en la prueba. En este elemento se definen las acciones a tomar en caso de causar error en alguna de las muestras de la prueba, que podría ser continuar la prueba, parar el hilo que ha causado error o parar la prueba en su conjunto.

Se definen las propiedades del Hilo (usuario) en:

✓ Número de Hilos, donde se definen la cantidad de usuarios que se van a simular en la prueba.

- ✓ Periodo de subida (en segundos) donde se especifica el periodo intermedio en segundos en los cuales va a ir iniciándose cada uno de los usuarios.
- ✓ Contador del bucle, gracias al cual se puede realizar la simulación varias veces o especificarle sin fin. Con lo cual si se especifica la prueba para 10 hilos y se le agrega al contador de bucles 4, se realiza un total de 40 muestras, ya que la prueba se ejecutará 40 veces, a razón de 10 usuarios por vez.

Otro de los sub-elementos que contiene el elemento **Grupo de Hilos** es el Planificador (ver figura #3). El Planificador se puede utilizar si se desea realizar las pruebas en un determinado momento y en un determinado tiempo. Este planificador está compuesto por:

- ✓ Tiempo de Arranque: Fecha y Hora en la que se desea iniciar la prueba.
- ✓ Tiempo de Finalización: Fecha y Hora en la que se desea Finalizar la prueba.
- ✓ Duración (segundos): Duración de cada uno de los Hilos.
- ✓ Retardo de Arranque (segundos): Tiempo de retardo en el arranque de cada uno de los hilos.

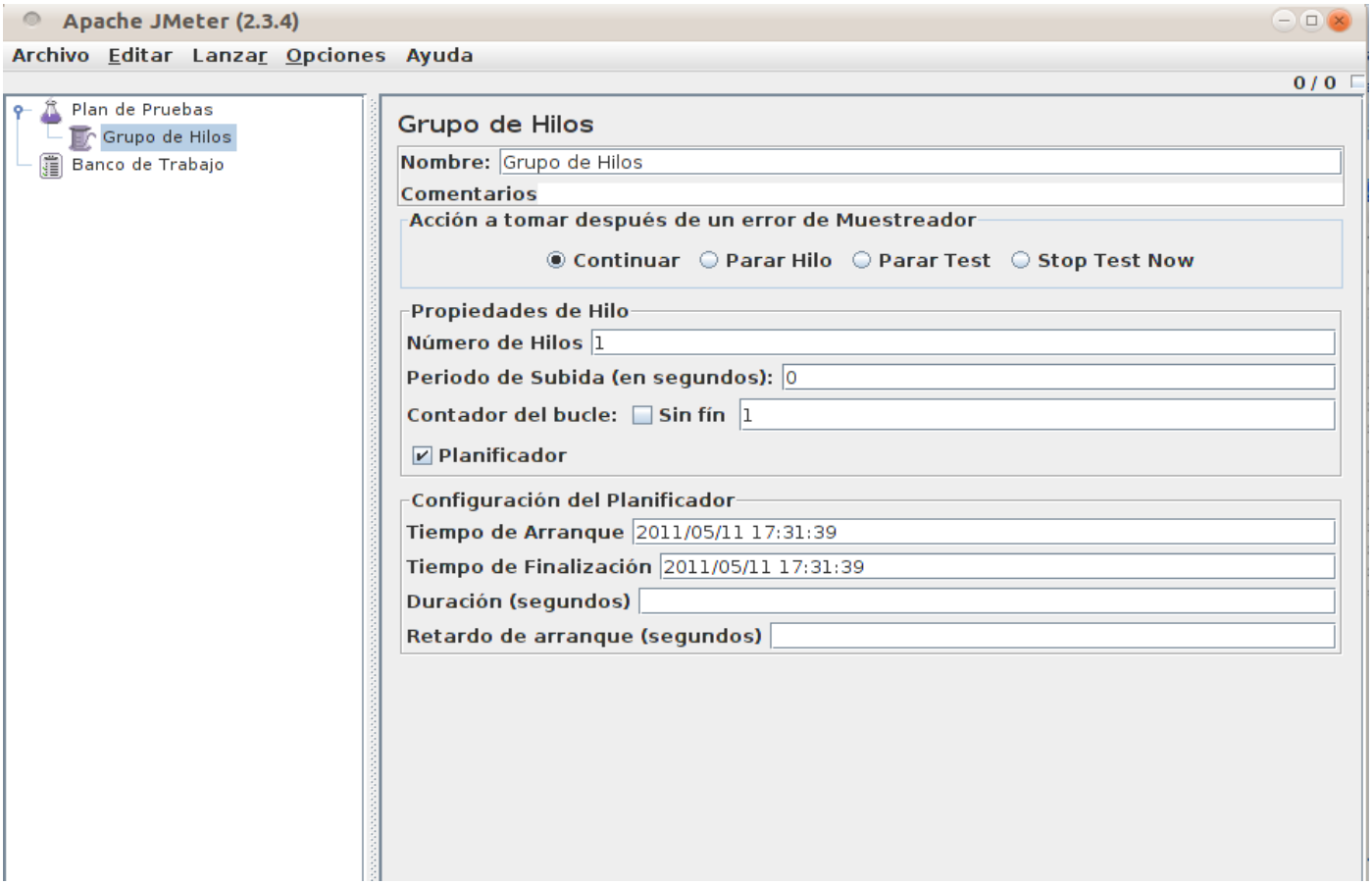


Figura #3: Grupo de Hilos

Teniendo los parámetros necesarios para la interacción con el sistema, es necesaria la utilización de una Aserción de Respuesta, figura #4, en ésta se especificará el patrón que se necesita validar en la prueba.

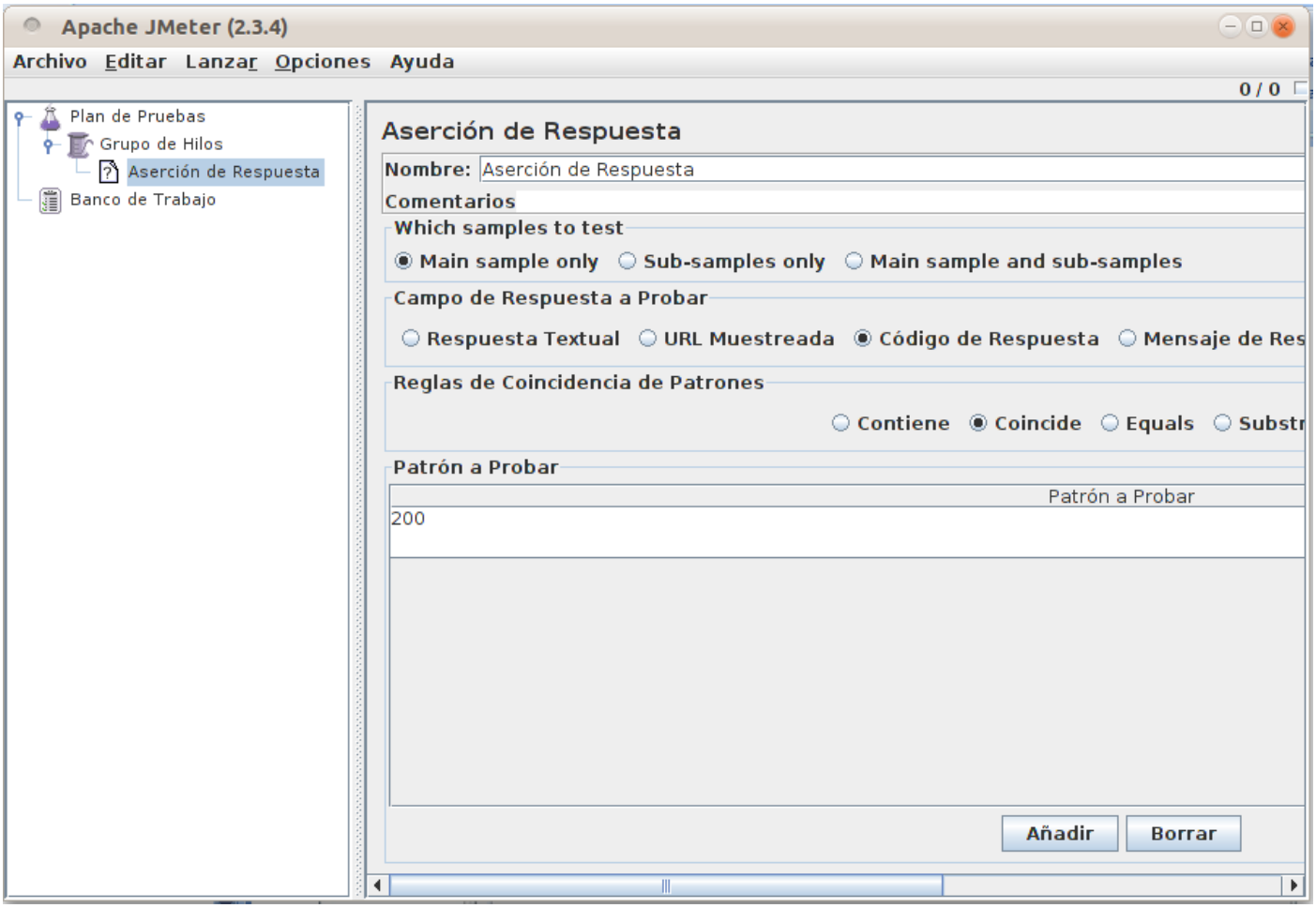


Figura #4: Aserción de Respuesta

Se pueden añadir tipos de **Aserción** cómo:

- ✓ Aserción de Respuesta, para comprobar la respuesta. Puede comprobarse el texto, o la URL, o el código de respuesta, o el mensaje de respuesta, e indicar si coincide con una serie de patrones, o no.
- ✓ Aserción de Duración, para indicar un tiempo máximo de ejecución.
- ✓ Aserción HTML, para verificar que el HTML, XML o XHTML esté correctamente construido.
- ✓ Aserción XML, para verificar que el resultado es un XML bien formado.

Finalmente solo se necesita alguno de los elementos en los cuales se registran los resultados de las pruebas, para ello se pueden utilizar una gama de Listeners, cada uno de ellos especializado en mostrar los resultados de las pruebas por aspectos y características diferentes en correspondencias a las necesidades del test.

Los más utilizados para las pruebas de Carga y Estrés son el Listener Informe agregado figura #5, y Ver Árbol de Resultados figura #6.

En el informe agregado se muestran una serie de datos, los cuales exponen el estado en el que se encuentra el software con respecto a los módulos probados.

- ✓ **Muestras:** Cantidad de páginas (Hilos) que simulan la cantidad de usuarios. Que están interactuando con el sistema desde la misma URL.
- ✓ **Media:** Media de páginas que se cargaron de manera satisfactoria.
- ✓ **Mediana:** Tiempo promedio que han tardado en cargarse las páginas.
- ✓ **Min:** Tiempo mínimo que ha demorado en cargarse una página.
- ✓ **Max:** Tiempo Máximo que ha tardado en cargarse una página.
- ✓ **Línea 90 %:** 90 por ciento de las páginas que se cargaron de manera satisfactoria.
- ✓ **%Error:** Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
- ✓ **Kb/Seg:** Velocidad de carga de las páginas.
- ✓ **Tiempos de Respuestas:** Total del tiempo que demoró en cargarse la cantidad de hilos de esa prueba.

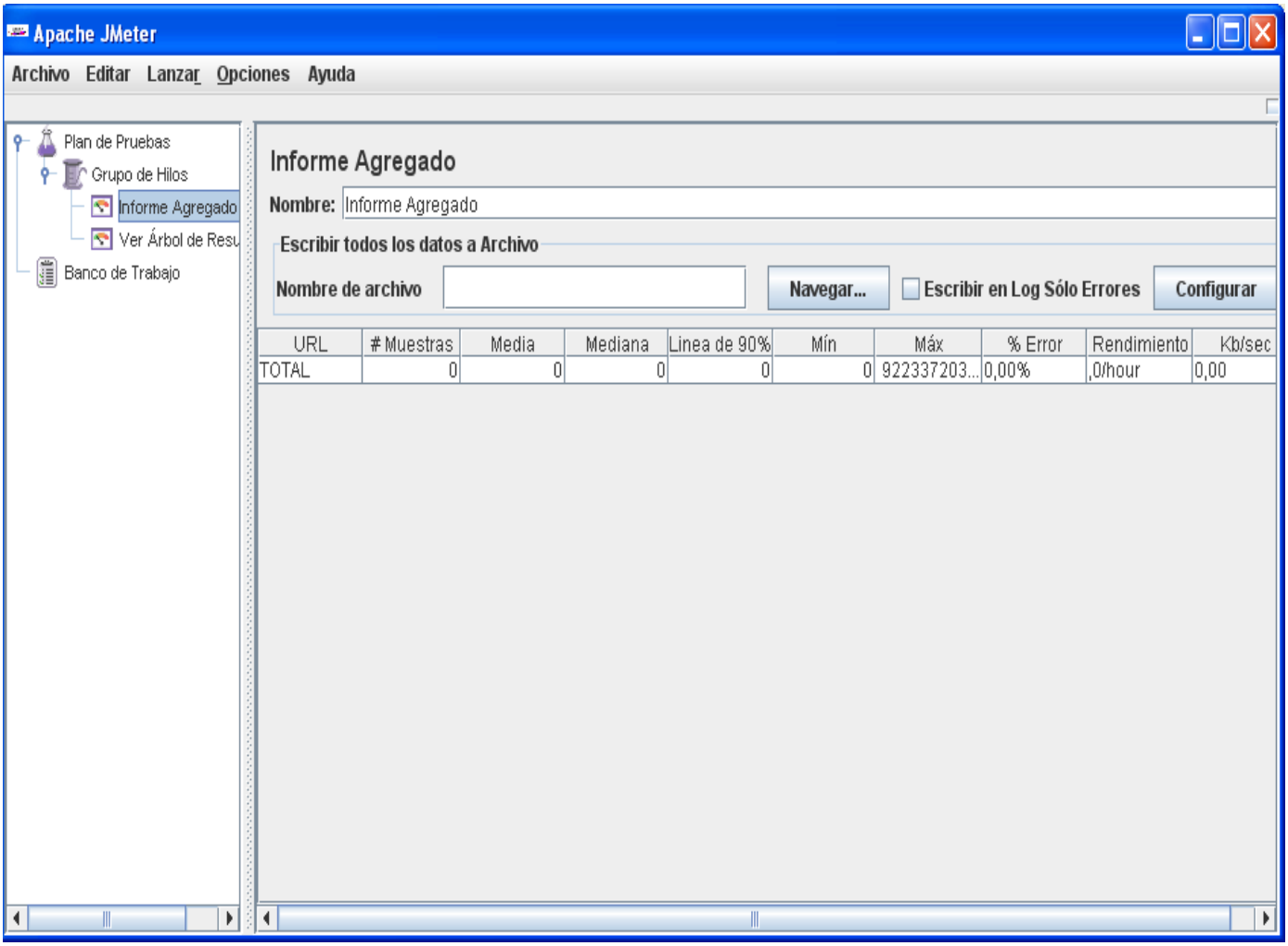


Figura #5: Informe Agregado

En el Listener "**Ver Árbol de Resultado**", se puede observar el instante en el que se van cargando las peticiones http de manera factible o no, el Resultado del Muestreador, así como la respuesta de sus códigos. Para los casos en los cuales ha sido fallo la petición http se mostraran en rojo dichas peticiones y para el caso en que sean cargadas satisfactoriamente se mostraran en verde. No obstante es necesario revisar la respuesta del Resultado del Muestreador ya que en algunas ocasiones se cargan las respuestas a las peticiones en verde y los códigos sin embargo no son los que en realidad tienen que mostrarse.

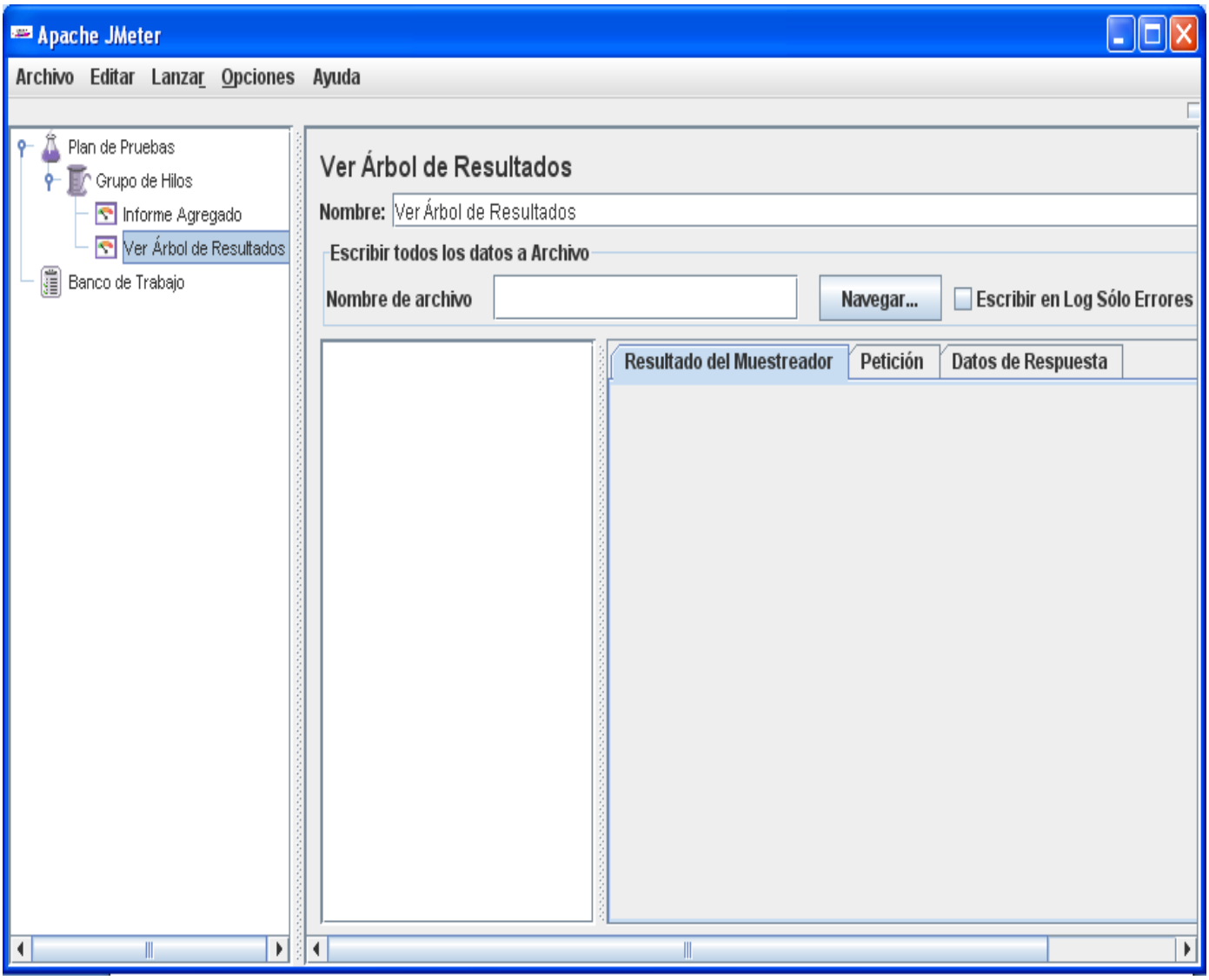


Figura #6: Ver Árbol de Resultados

2.6.2 Utilización de Apache JMeter en Escenarios de Prueba Simples de Carga

La elaboración de los scripts simples está realizada sobre la base de direcciones http del sitio web que se desea probar. Se deben tener en cuenta como materia prima para la realización de estas pruebas la dirección o el nombre del servidor donde se encuentra la aplicación, el puerto por el cual se estará accediendo a ella, el protocolo que se especifica. Adicional se le especifica el tipo de respuesta que se

espera del servidor.La confección de los scripts simples, es aconsejable para sitios estáticos, donde hay poca interacción con los usuarios y poco flujo de datos. Lo primero a tener en cuenta para elaborar el plan de pruebas es seleccionar el elemento “Grupo de Hilos”.

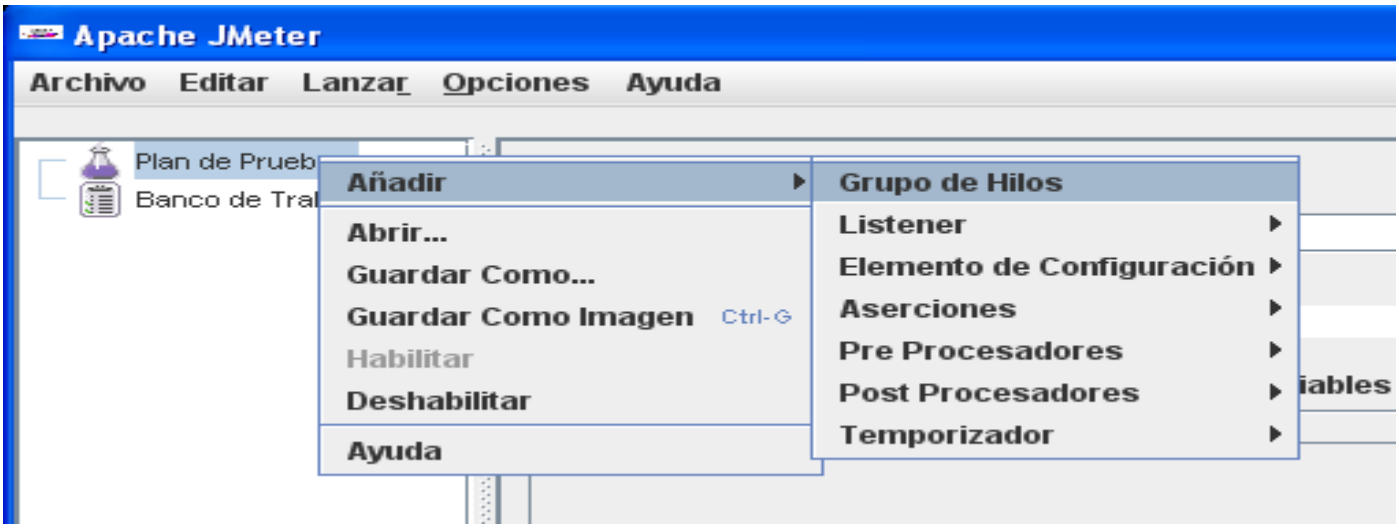


Figura #7: Seleccionar el elemento Grupo de Hilos

Luego se especifican la cantidad de usuarios que se desean simular, el tiempo en que irán subiendo a la aplicación y la cantidad de veces que lo estarán haciendo. Como se muestra en la figura 8.

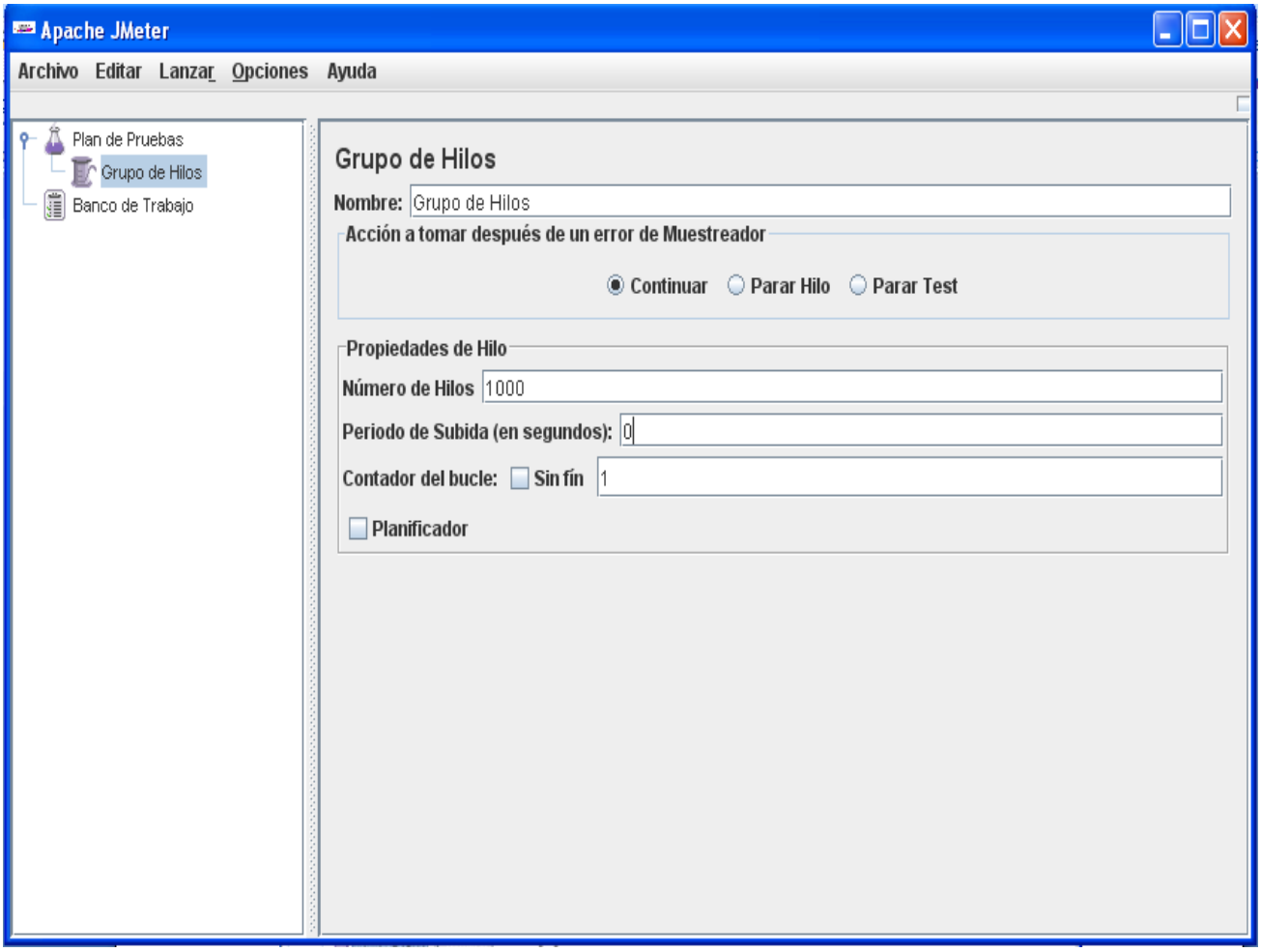


Figura #8: Añadir los datos al Grupo de Hilos

Teniendo el Grupo de Hilos se selecciona una Petición HTTP o un elemento de Valores por defecto para Petición HTTP donde se debe especificar todos los datos referentes a la conexión con la aplicación.

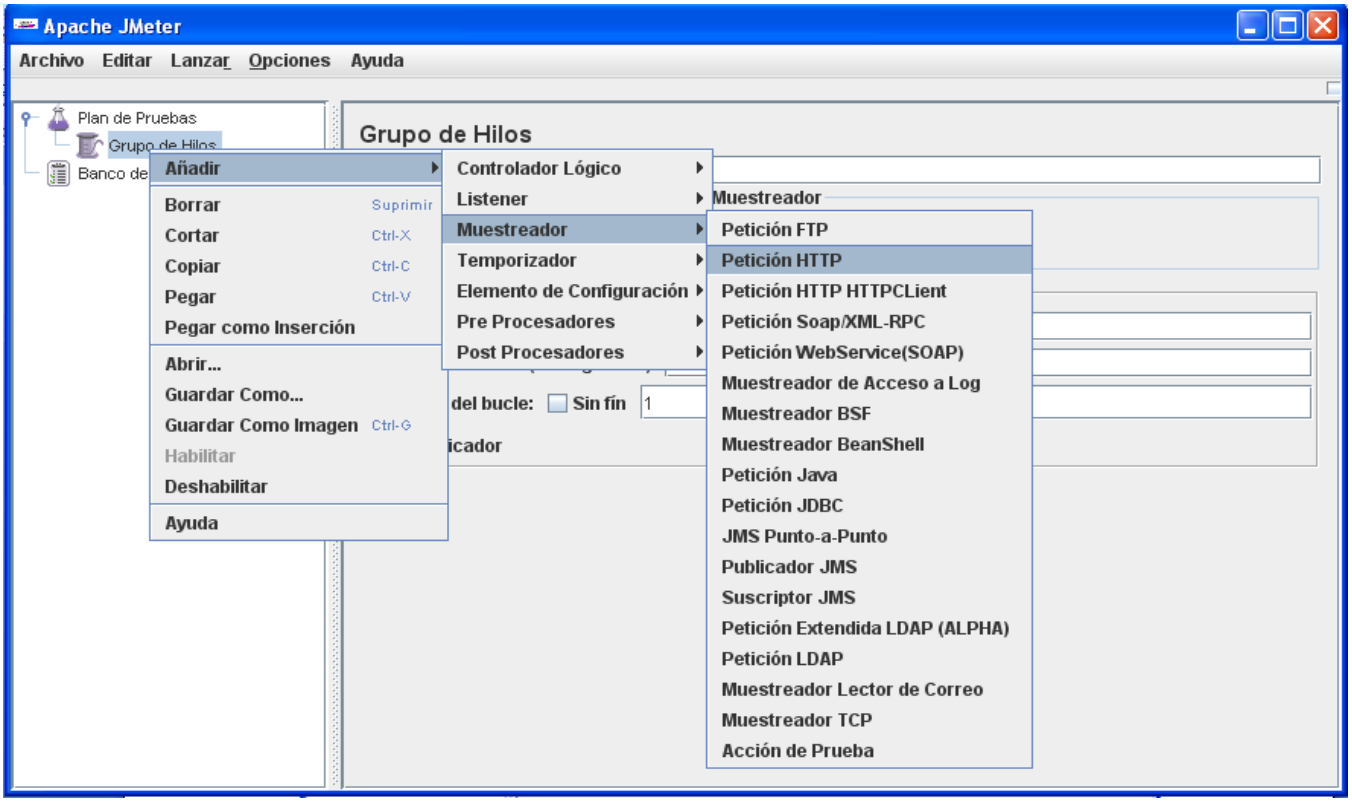


Figura #9: Seleccionar elemento de Petición HTTP

Figura #10: Insertar los Datos de la Aplicación

Al crear el Plan de Pruebas se crea una "lista ordenada" de peticiones (Peticiones Http) por lo que se utilizan "**Controladores Lógicos**" y dentro las peticiones, ya que éstos afectan el orden de las peticiones

y suelen simular un escenario mucho más real. El siguiente paso puede ser la selección de un controlador y luego anidar el resto de las peticiones.

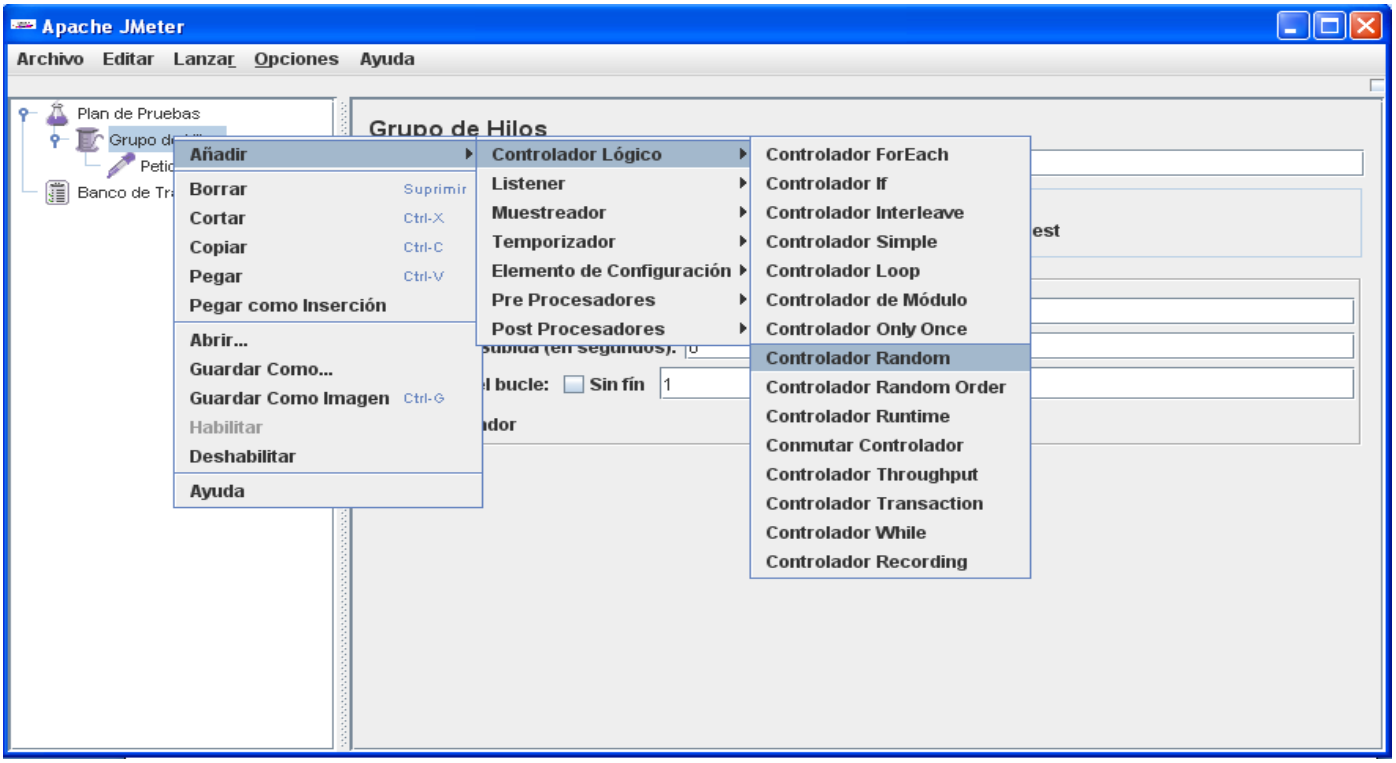


Figura #11: Seleccionar elemento Controlador Lógico

Teniendo el Controlador Lógico se adicionan tantas Peticiones HTTP como sean necesarias a probar en la aplicación. Si alguna de estas peticiones tiene un orden para ejecutarse, debe analizarse bien en caso de utilizar algún Controlador Lógico**.**

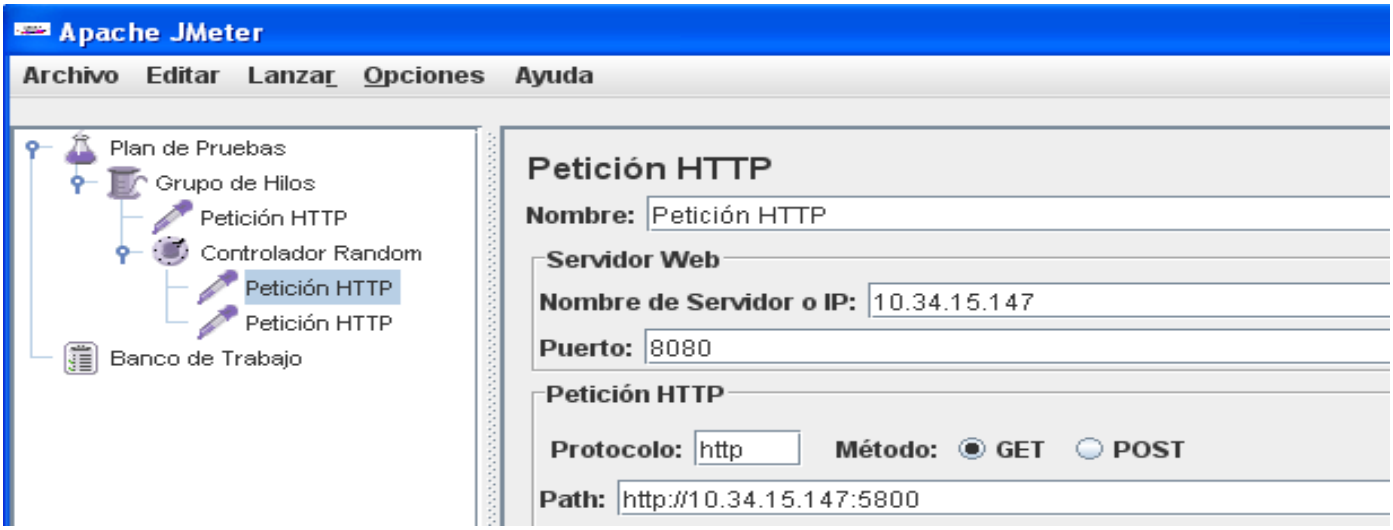


Figura #12: Anidar Peticiones HTTP

Teniendo los elementos con los cuales se van a interactuar para la conexión de la aplicación y la herramienta de prueba se adicionan elementos en los cuales se detallarán los parámetros que se quieren comprobar en el test, esto puede realizarse apoyándose en elementos como la Aserción de Respuesta.

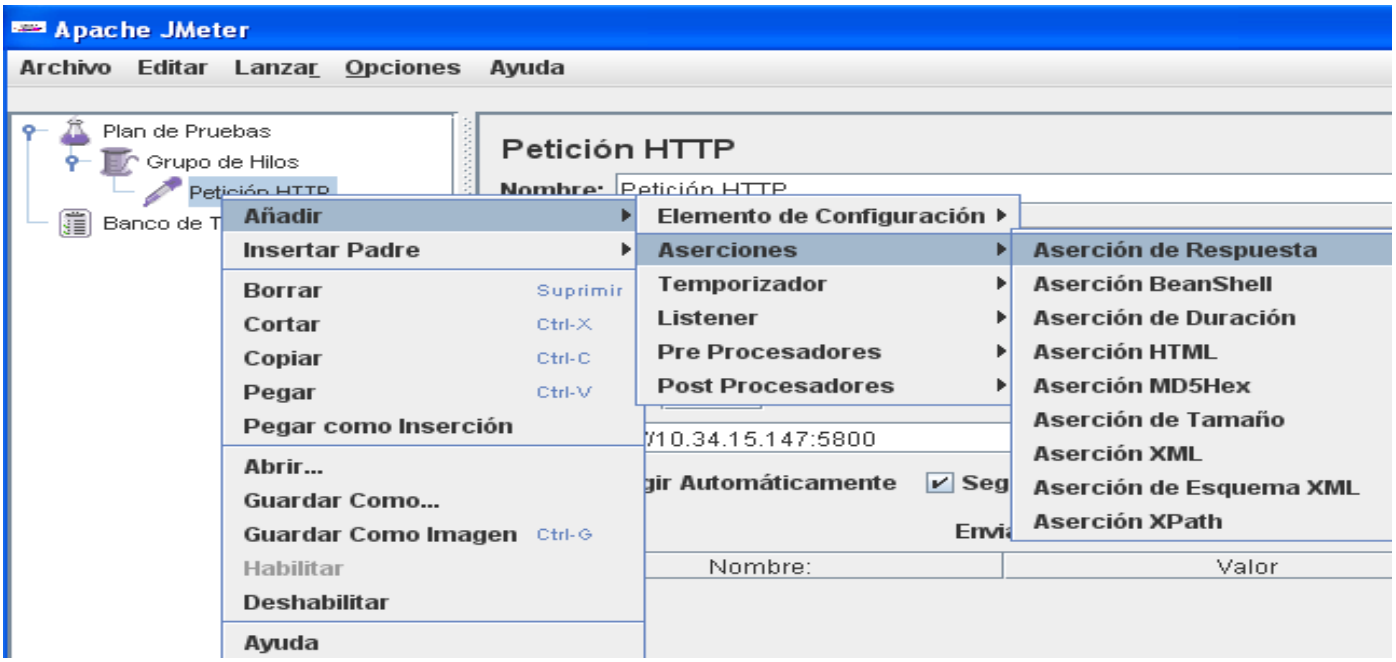


Figura #13: Seleccionar Aserción de Respuesta.

Para este caso lo único que se necesita saber es que la página a la cual se quiere acceder se ha cargado de manera satisfactoria, por tanto se especifica el código para una página cargada satisfactoriamente como Patrón de Entrada y se le especifica que el código de la página probada coincida con el código que se pasa en el test.

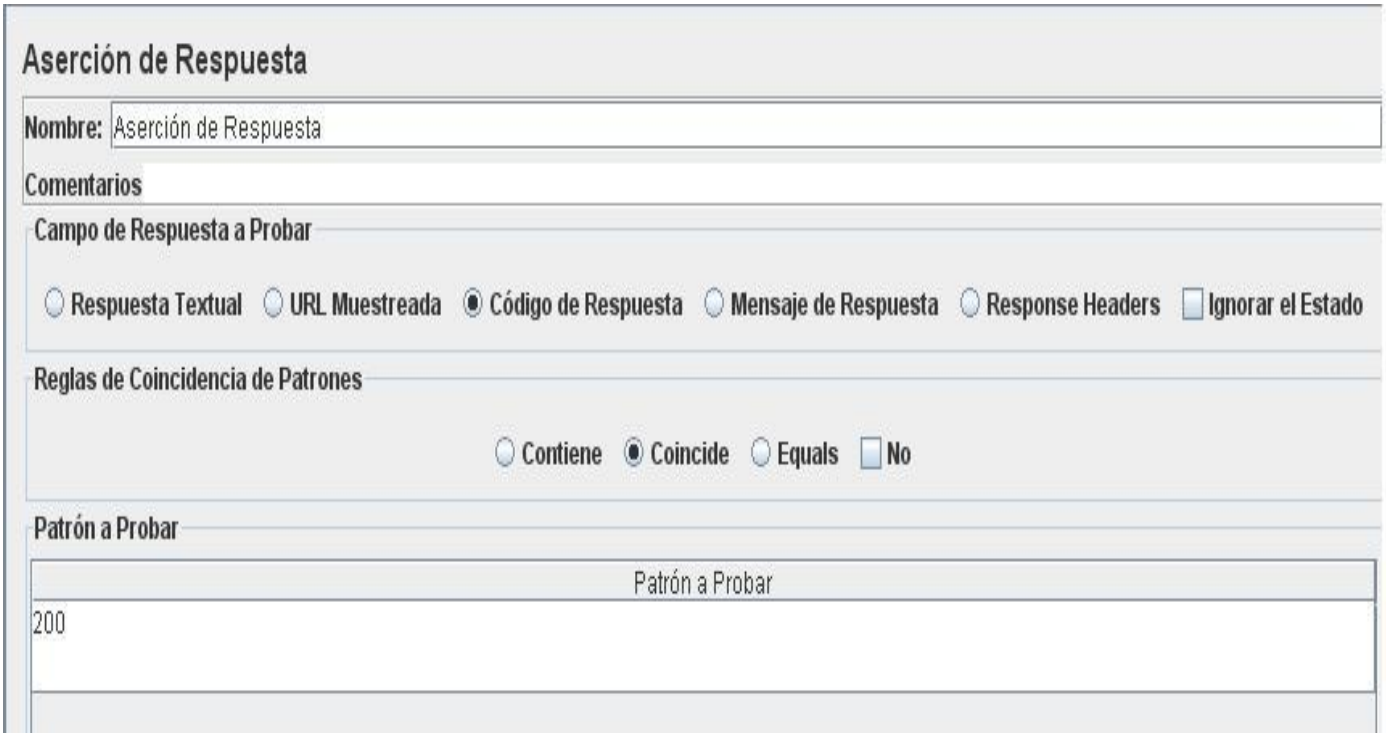


Figura #14: Datos a insertar en la Aserción de Respuesta

Para este tipo de prueba se necesita un informe en el que se muestren resultados relacionados a los datos necesarios para comprobar el comportamiento del sistema en cuanto a estos aspectos. Para ello se adiciona un Informe Agregado.

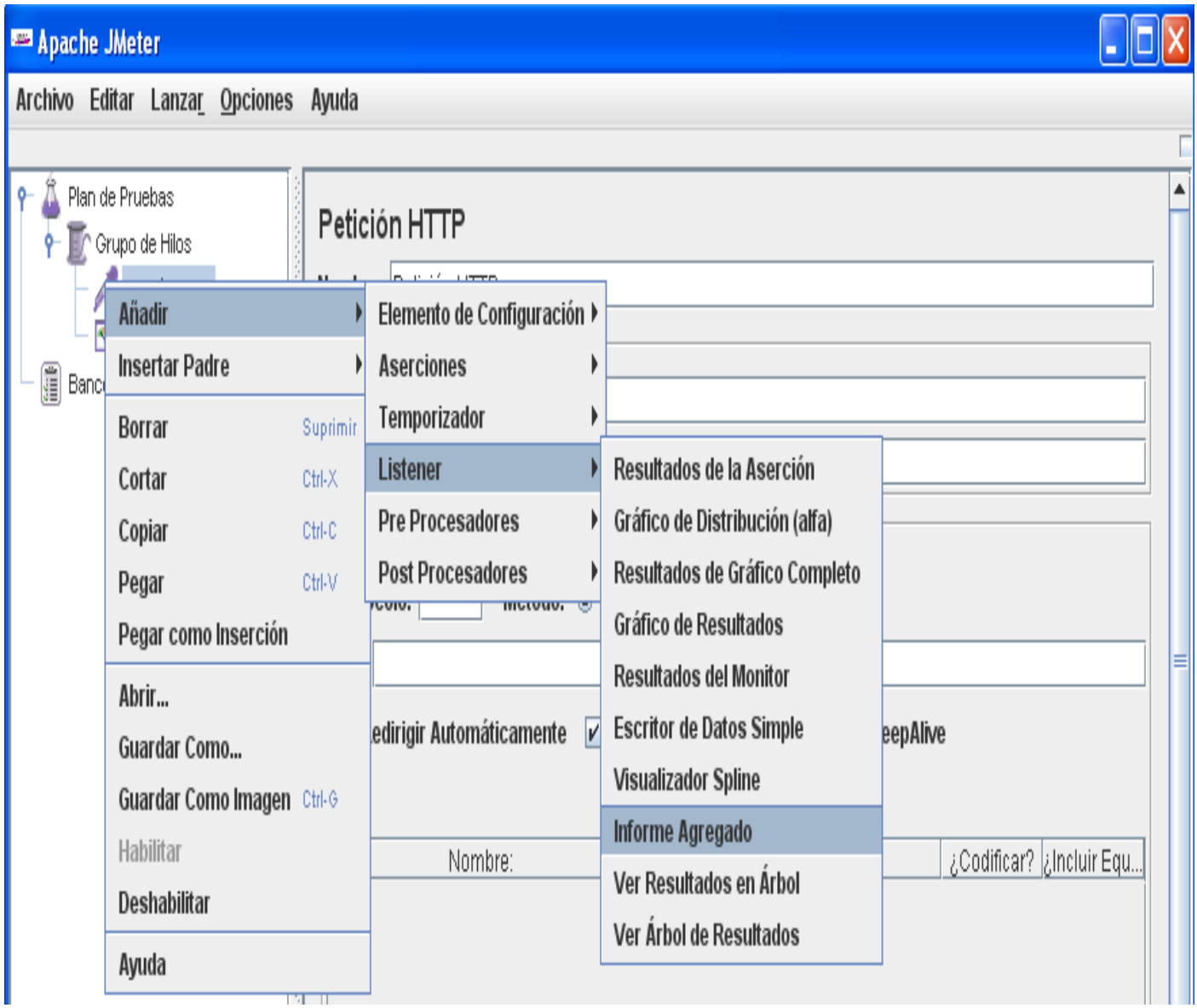


Figura #15: Informe Agregado

Uno de los elementos a utilizar a la hora de recoger los resultados es el Árbol de Resultados, ya que este muestra en detalle la información que ha sido cargada. Mostrando el código que ha dado la pagina, el texto y otros atributos.

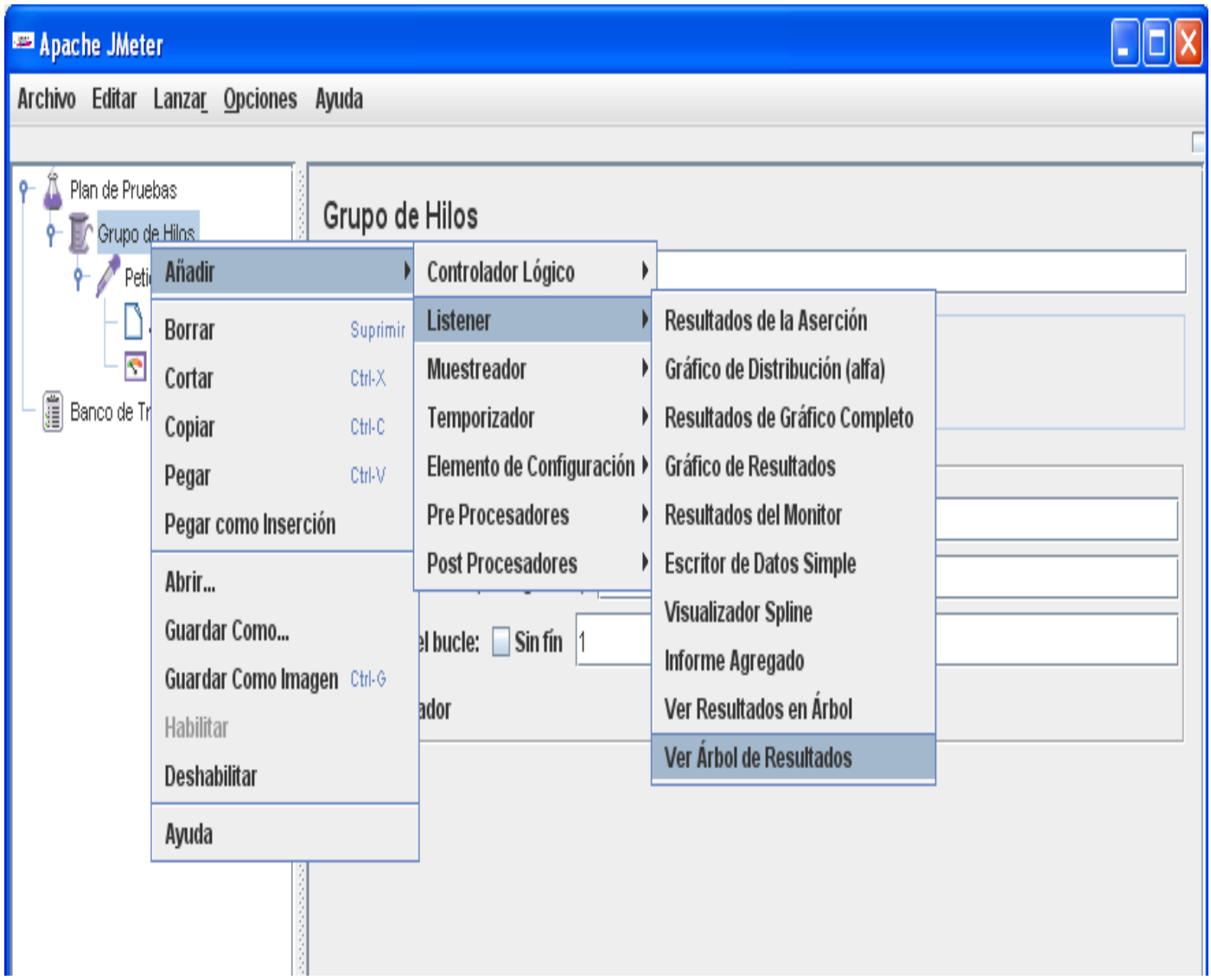


Figura #16: Ver Árbol de Resultados

Luego de tener todos los recursos para realizar la prueba se accede a la parte superior de la herramienta en la opción Lanzar se presiona Arrancar (Ctrl + R) y se puede ejecutar la prueba. Luego se espera a que paren todos los hilos y se muestren los resultados, los cuales serán utilizados por el equipo de desarrolladores para saber las medidas que deben tomar para mejorarlos en caso de no ser muy buenos en comparación con los requeridos.

Estos son los pasos básicos para realizar una prueba de carga y estrés, el éxito de la misma, depende del estilo que se desea seguir. Ya que para realizar estas pruebas se deben seguir sus principios básicos de concurrencia y carga y esto solo se logra, definiendo cuales han de ser los caminos más importantes a probar y a sobrecargar en el sistema y cuáles podrían ser los fallos posibles en el mismo. Teniendo los puntos débiles se puede medir la capacidad del sistema de reaccionar ante situaciones extremas.

2.6.3 Utilización de Apache JMeter en Escenarios de Pruebas Complejas de Carga

En el acápite anterior se muestra una guía para realizar pruebas de forma sencilla a una aplicación web en la cual no se realiza intercambios de datos entre el cliente y el servidor. Para la ejecución de pruebas de carga y estrés en un software de gestión los conocimientos seguirían siendo los mismos, en este caso se realizarán diferentes pasos y se agregarán nuevos elementos los cuales permitirán entrar datos al sistema y enviarlos a la base de datos del mismo.

Inicialmente cuando se ejecuta la herramienta se muestra el Plan de Pruebas y el Banco de Trabajo (ver figura #17) se procede a Añadir -> ElementosNoDePrueba -> Servidor Proxy HTTP. Luego se define el puerto por el que se accederá a la aplicación en este caso (8080) y en Agrupación se selecciona (Poner cada grupo en un nuevo controlador, ver figura #18) de ahí se arranca el servidor proxy y se procede a configurar las opciones de internet (figura #19). Se va a conexiones se pone localhost y se mantiene 8080 porque es el puerto que se definió. Posteriormente se accede a Plan de Pruebas: Añadir -> Grupos de Hilos (figura #20) y aquí se define la cantidad de usuarios que se piensa simular para la realización de la prueba de carga. Después de seguir todos estos pasos se accede a la aplicación y se empieza a grabar los escenarios a probar, estas grabaciones se pueden ver dentro de Grupos de Hilos. Luego se eliminan los elementos de respuestas pues éstos no se utilizarán, posteriormente se guarda el Plan de Pruebas y se cierra la aplicación de ahí se modifica las conexiones a internet para restablecer las mismas de esta forma se concluye la grabación de las pruebas.

Se accede a la herramienta nuevamente para agregarle el resto de los elementos de prueba se va a archivo -> abrir y se busca la grabación que se realizó. Se procede con agregar el resto de los elementos de prueba para ello se necesita el árbol de resultado: Grupo de Hilos -> Añadir -> Listener -> Ver Árbol de Resultados para mostrar los resultados en forma de árbol, además la Aserción de Respuesta: Grupo de Hilos -> Añadir -> Aserciones -> Aserción de Respuesta que va ser la que controlará si las peticiones

están respondiendo de manera correcta, en este caso 200 (ver figura #22) va ser el código de respuesta del servidor satisfactoriamente, la cual se pone antes de las peticiones. A parte de agregar el árbol de resultados también se suma el informe agregado: Grupo de Hilos -> Añadir -> Listener -> Informe Agregado. Tanto el árbol de resultado como el informe agregado son elementos de respuesta de la prueba y la aserción de respuesta sería la que comprobaría la prueba; para grabar la prueba se presiona Control + R o la opción arrancar dirigiéndose a la parte superior de la herramienta en Lanzar. Luego se puede ver en la parte superior derecha como se muestra la cantidad de hilos que va a simular la cantidad de usuarios en la aplicación, cuando quede en cero te ha indicado que la prueba ha finalizado. A continuación se representarán las figuras antes descritas.

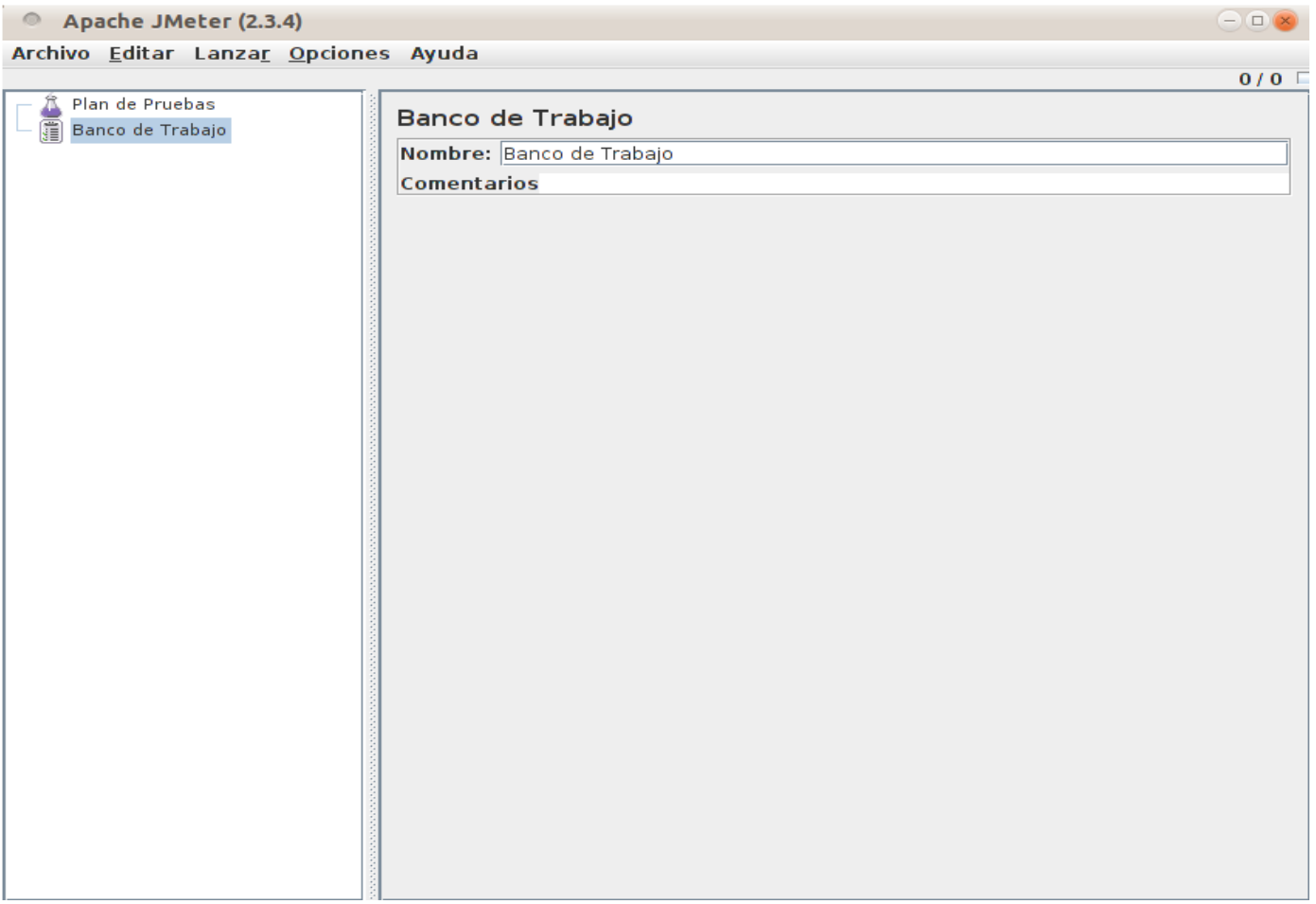


Figura #17: Muestra la Interfaz de la Herramienta Apache JMeter

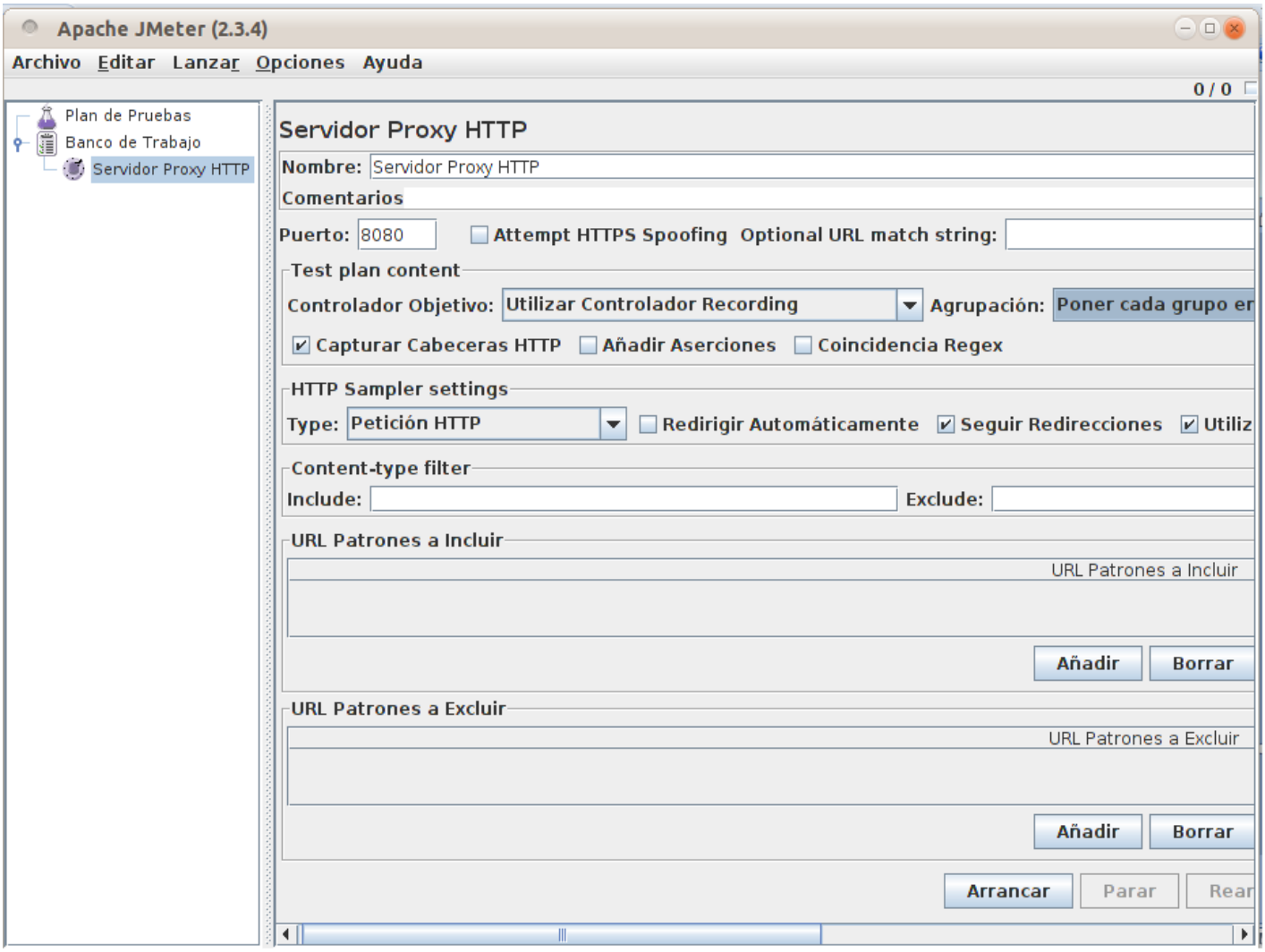


Figura #18: Muestra la Configuración del Servidor Proxy HTTP

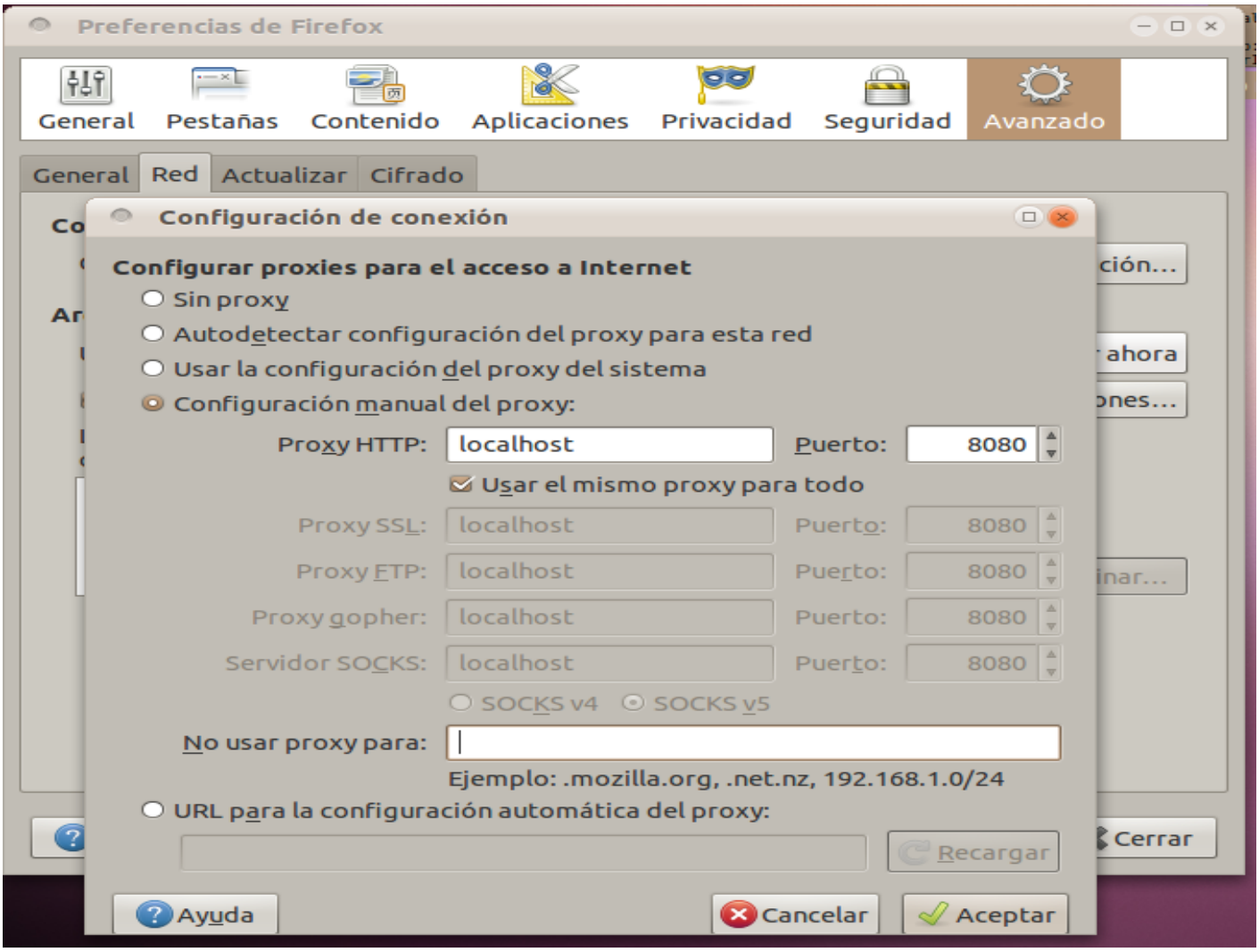


Figura #19: Valores de entrada que no necesitan proxy

La aplicación que se vaya a probar debe quedar fuera de las entradas que no necesitan proxy. De esta manera se le estará indicando que el proxy que esta utilizará será el perteneciente al Apache JMeter.

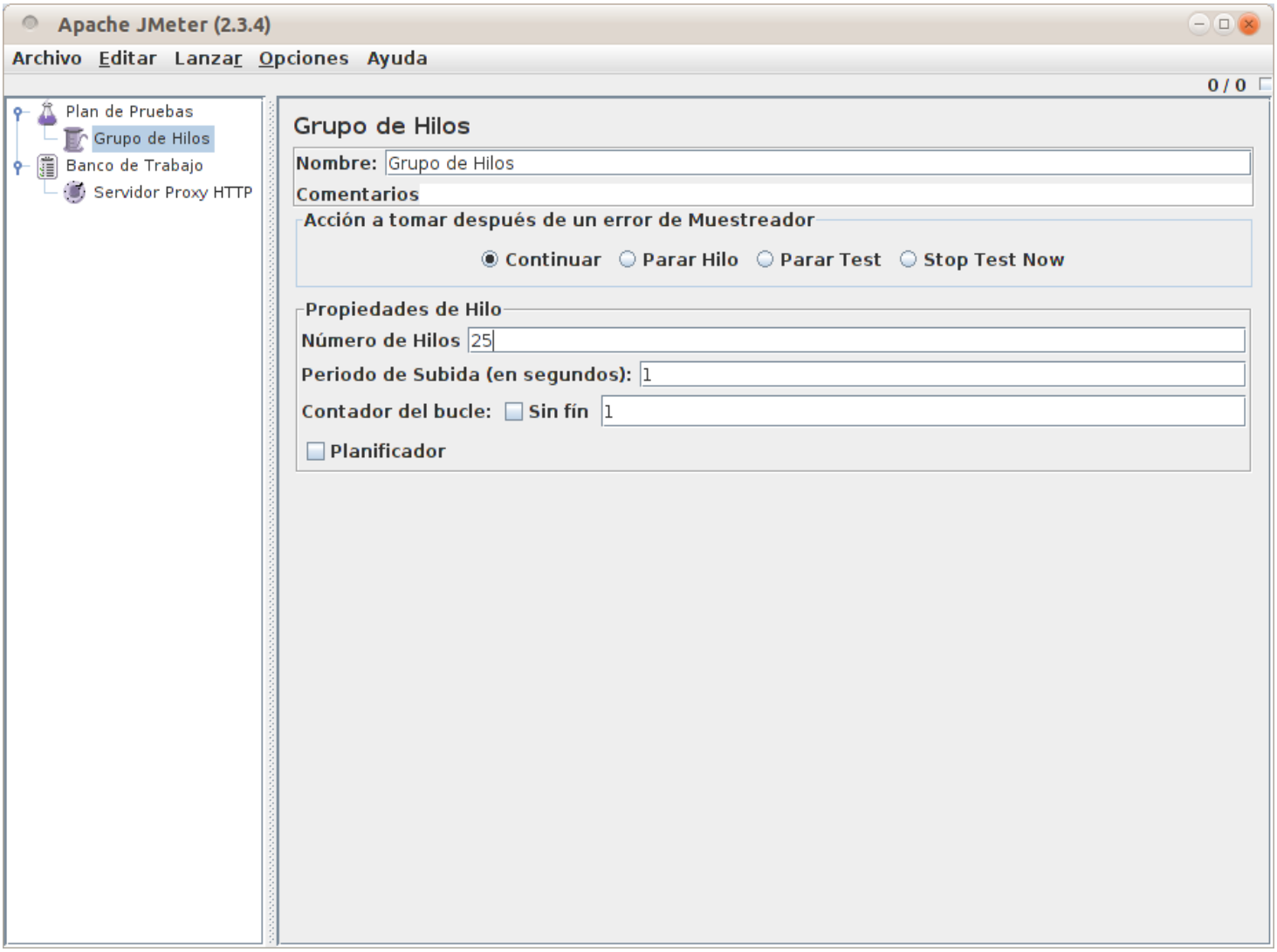


Figura #20: Muestra la Carga a simular dentro de Grupo de Hilos

En la sección siguiente cada paso realizado en la Web será registrado en el JMeter, de manera organizada de la forma que se le especificó. Mostrando para este ejemplo cada número de llamadas en controladores lógicos por separado. Lo que permite identificar cada uno de los componentes que han dado respuesta de parte del sistema.

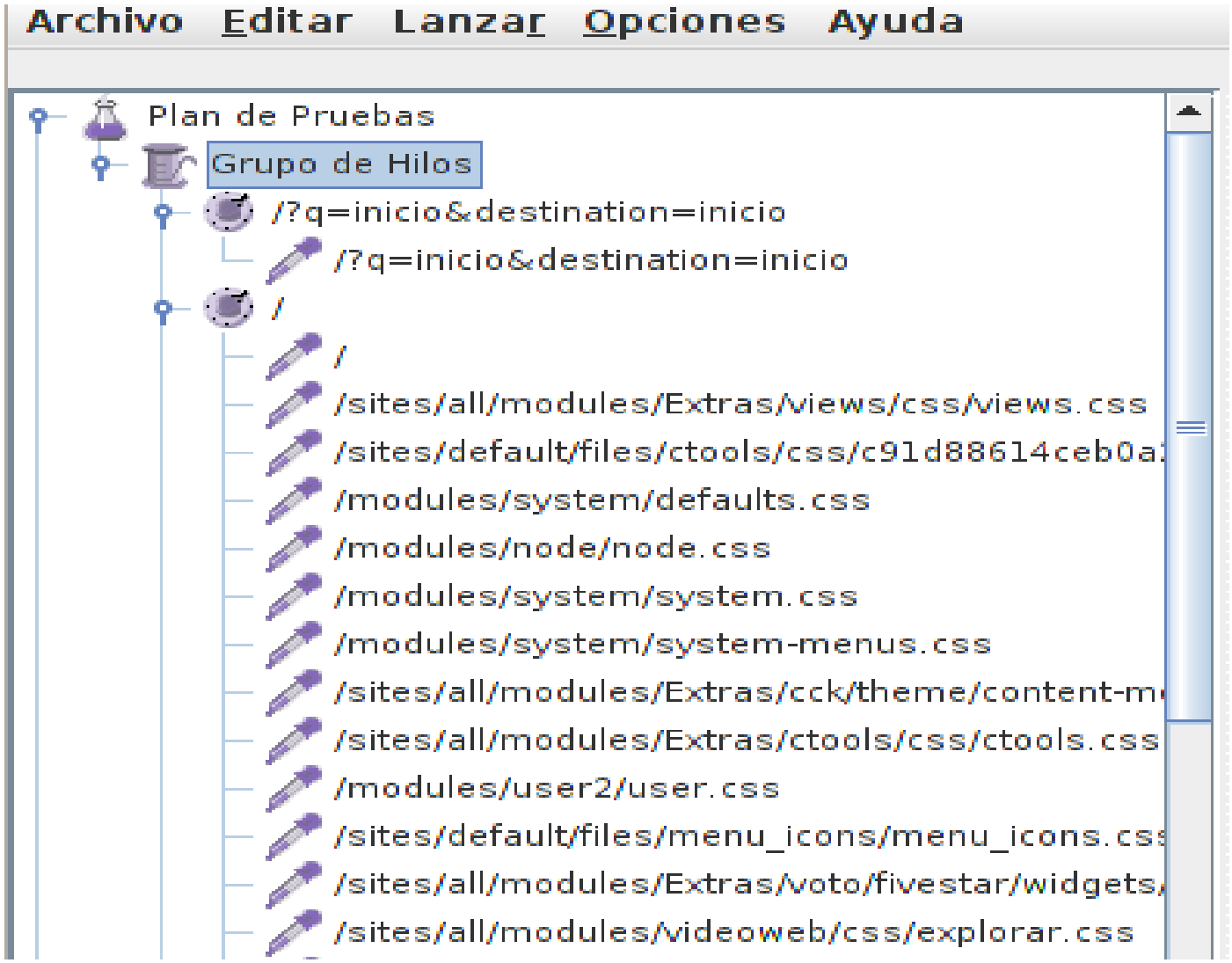


Figura #21: Grabación de Navegación de la Web

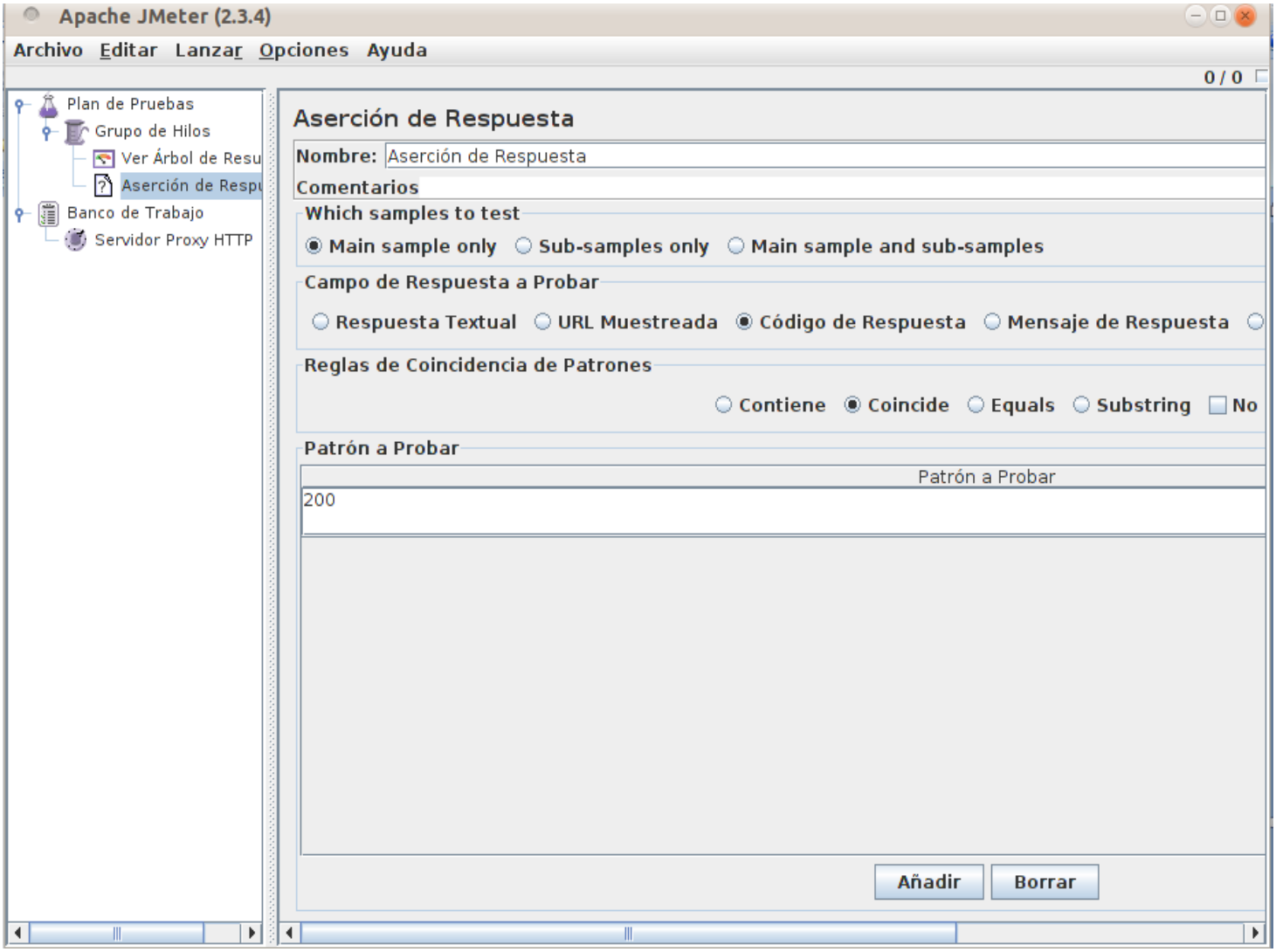


Figura #22: Muestra el Control de las peticiones mediante la Aserción de Respuesta

2.6.3.1 Utilización de Apache JMeter para Pruebas de Estrés

En el epígrafe 1.5 se exponen los distintos tipos de pruebas automatizadas y dentro de ellas las diferentes pruebas que se pueden aplicar. Aunque existen marcadas semejanzas entre los conceptos de prueba de carga y prueba de estrés, se diferencian principalmente en el objetivo de cada una, mientras que las pruebas de carga se encargan de probar el comportamiento del sistema bajo una carga soportable, las pruebas de estrés son dedicadas a someter al sistema a una carga tan grande que sea capaz de colapsar

el mismo. A la hora de diseñar, poseen implementaciones casi parecidas; si antes se simuló la prueba de carga para 25 usuarios, para este tipo de prueba es necesario simular un número de usuario realmente grande, en este caso 70 usuarios y de ahí ir probando hasta estresar la aplicación o el servidor aumentando el número de usuarios conectados concurrentemente. Ejemplo de número de hilos para pruebas de estrés.

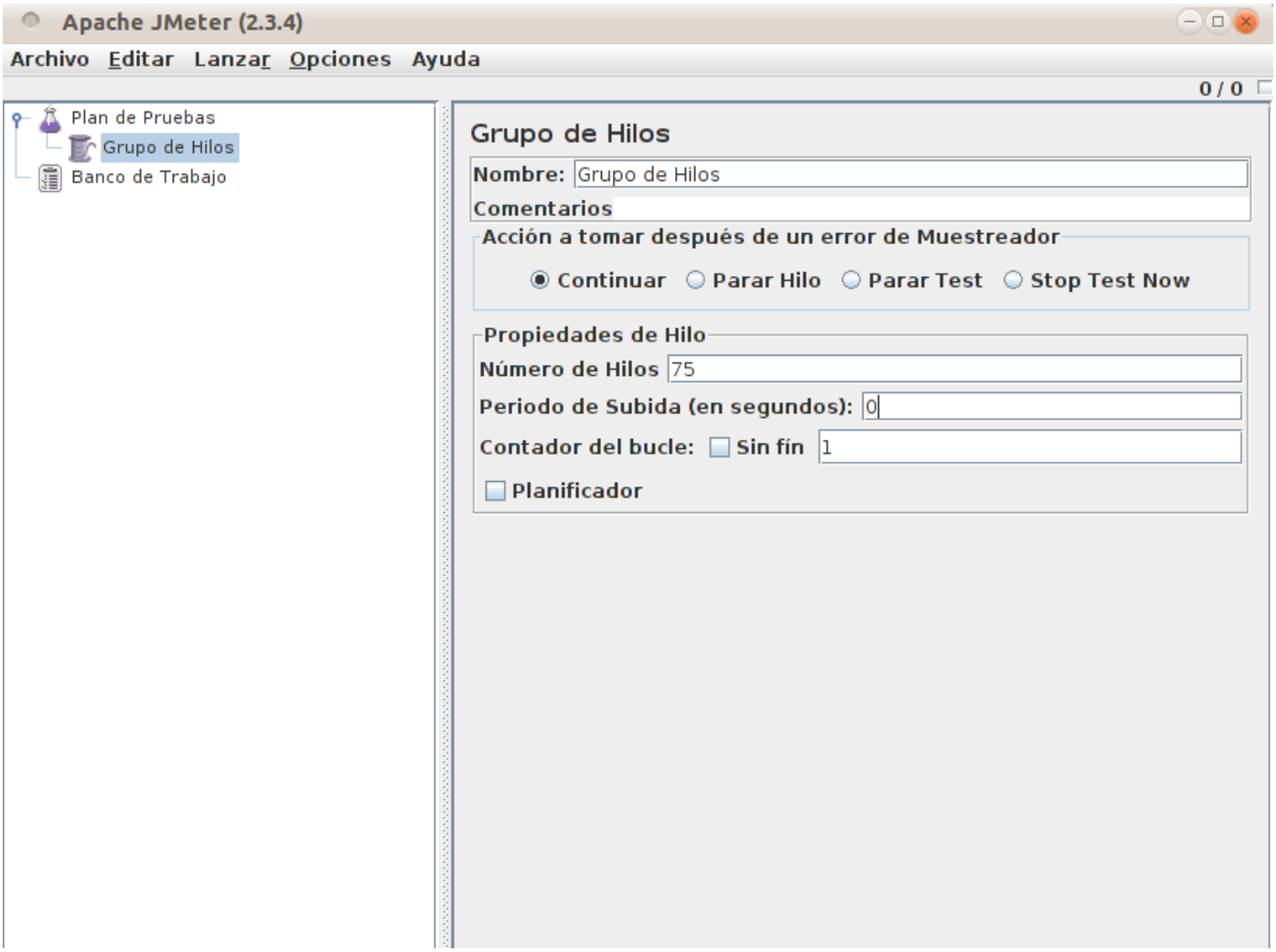


Figura #23: Muestra los usuarios simulados para estresar la aplicación

2.7 Conclusiones del Capítulo

En el presente capítulo se realizó la propuesta de la herramienta para el proceso de prueba de desarrollo de software. Mediante la descripción de un conjunto de herramientas automatizadas, se llevó a cabo una comparación de las principales características de las mismas. También se detalló el proceso de instalación y utilización de la herramienta propuesta, mostrando ejemplo de su uso y de esta manera le dio paso a la puesta en práctica en uno de los proyectos pertenecientes al Departamento Señales Digitales.

Capítulo 3: Validación de Resultados

3.1 Introducción

En este documento se ha demostrado la necesidad de la utilización de una herramienta para la realización de Pruebas de Software automáticas. En este capítulo se describirá cómo se ha aplicado el proceso y la utilización de la herramienta propuesta en el capítulo 2, la cual es Apache JMeter, el análisis de los resultados obtenidos durante la aplicación de esta herramienta a uno de los proyectos del Departamento de Señales Digitales así como el proceso de validación de la propuesta.

3.2 Planificación y Concepción de las Pruebas de Carga y Estrés

3.2.1 Estrategia de Pruebas

- ✓ **Objetivos**

Para las pruebas de carga y estrés el objetivo es obtener un índice de resultados de comportamiento del sistema en determinadas condiciones, variándose las condiciones en dependencia del tipo de pruebas que se quiere hacer. Al realizar estas pruebas de aplicación se obtienen resultados de un comportamiento que puede ser muy cercano a un comportamiento real y en dependencia con esto el equipo de trabajo podría hacer cambios favorables para mejorar el comportamiento de la aplicación.

- ✓ **Técnica**

Para la realización de las pruebas con la herramienta Apache Jmeter se utilizó el método de Caja Negra, haciendo énfasis en la técnica de partición equivalente para la realización de pruebas de carga y estrés. Estas pruebas necesitan que se realice una entrada de datos y la validación de estos datos contra resultados esperados es por ello que se define como técnica utilizada para este proceso de pruebas la de partición equivalente.

- ✓ **Entorno de Prueba**

El entorno de prueba es un ambiente más o menos similar al ambiente donde se realiza la aplicación, no tiene que tener una total semejanza con el ambiente de producción sino que solo es factible simular en una variedad de grados de aproximación, ya que en ocasiones cuesta mucho hacer una semejanza total

del ambiente de producción. Para la grabación de los escenarios de prueba se necesitan los siguientes datos recogidos en el artefacto: Especificación de Recursos. Se deben acondicionar los recursos tanto de software como de hardware.

Tabla 2: Especificación de recursos para las pruebas al proyecto VideoWeb

Tarea	Recurso	Cantidad de Personas	Fecha Limite
Elaboración del plan de pruebas	**1 PC. Apache JMeter**	**1**	**30/04/2011**
Realizar los test en la herramienta	**1 PC. 1Gb de Ram. M Office. Java Virtual Machine 1.3 o superior. Apache JMeter. Navegador IExplorer. Se necesita además que se encuentre corriendo el servidor de la aplicación.**	**1**	**03/05/2011**
Ejecución de las pruebas	**1 PC. 1Gb de Ram. M Office. Java Virtual Machine 1.3 o superior. Apache JMeter. Navegador IExplorer. Se necesita además que se encuentre corriendo el servidor de la aplicación.**	**1**	**03/05/2011**
Análisis de los resultados	**1 PC. M Office**	**1**	**03/05/2011**

Proceso de Prueba

El proceso de automatización de las pruebas consta de 4 etapas fundamentales:

- ✓ Grabación de los escenarios de prueba.
- ✓ Confección de los test.
- ✓ Ejecución de las pruebas y recolección de los resultados.
- ✓ Análisis de los resultados.

- ✓ **Grabación de los Escenarios de Prueba:**

La grabación de los escenarios de prueba se conformó por 5 grabaciones independientes para la realización de las mismas. Debido a que la herramienta Apache JMeter consume una porción considerable del rendimiento de la máquina.

Habiendo previamente configurado el navegador se accedió a la dirección del servidor web al que se le realizan las pruebas y se navegó a través de todos sus módulos.

- ✓ **Confección de los Test:**

Para la creación de los diferentes tipos de test, se utilizaron las grabaciones de los escenarios explicados anteriormente y se modificaron la simulación de los usuarios en Grupos de Hilos según el tipo de pruebas a realizar y las necesidades que se necesite probar.

- ✓ **Descripción de los elementos de prueba:**

Los elementos que componen la prueba son configurados de la forma que se explica en el capítulo anterior. El número de usuarios concurrentes que se simulará para cada uno de los escenarios del portal web perteneciente al proyecto VideoWeb del Departamento Señales Digitales son: para Crear publicación archivo multimedia 50, para Eliminar publicación archivo multimedia 45, para Modificar publicación archivo multimedia 35, y para Publicar archivo multimedia 50. Los valores por defecto para las peticiones HTTP fueron:

- ✓ Dirección IP o nombre del servidor donde se encuentra el Portal Web.
- ✓ Tipo de protocolo que se utiliza.
- ✓ Puerto por el que se accede.

Los demás elementos de prueba se añaden como se explica en epígrafe **2.7.1.2** donde se describe el uso de la herramienta Apache JMeter.

- ✓ **Ejecución de las pruebas y recolección de los resultados:**

Para la realización de los distintos tipos de prueba se comprobó que todos los recursos de software como de hardware se encontraran en perfecto estado o sea, que la aplicación web se encontrara en funcionamiento, que estuviera disponible el servidor web, que no existieran problemas con la red.

Casos de Prueba: La prueba se realizó con un único caso de prueba siguiendo la estructura planteada en la confección de los test.

Calendario y Plazos: En el cronograma asociado a estas pruebas se definen las tareas que se realizaron acompañadas de sus fechas en las que deben ser ejecutadas. A continuación se representa el cronograma del portal web de VideoWeb.

Tabla 3: Cronograma del Proceso de Pruebas Automatizadas de VideoWeb

Inicio	Fecha(Inicio - Fin)	Tiempo de Desarrollo (min)
Elaboración de las grabaciones de prueba	**03/05/2011**	**5**
Realizar los test en la herramienta	**03/05/2011**	**7**
Ejecución de las pruebas	**03/05/2011**	**10**
Análisis de los resultados	**03/05/2011**	**20**
Total	**03/05/2011**	**42**

Para el proyecto al cual se le aplicó las pruebas, el tiempo de la elaboración de las grabaciones de prueba fue de 5 minutos. A la realización de los test se le dedicaron 7 minutos, en la ejecución fueron 10 minutos y 20 para el análisis de los resultados dando esto un total de 42 en la ejecución de las pruebas.

- ✓ **Realización de los Test**

Para el uso de la herramienta Apache JMeter se conformaron 5 test compuesto por la grabación realizada sobre el sitio web, donde se simuló la interacción de 50, 45, 35, 50 y 75 usuarios de forma simultánea.

3.2.2 Ejecución de las Pruebas

Las pruebas fueron realizadas al portal Web que presenta las siguientes características de software:

Portal Web VideoWeb

- ✓ Servidor web: Apache Tomcat 6.0.13
- ✓ Lenguaje utilizado: PHP

- ✓ Con el siguiente hardware de servidor:
- ✓ Disco Duro: (160 GB, 7200 RPM)
- ✓ Característica de la Red: 100.0 Mbps
- ✓ Memoria del sistema: 1 Gb (PC2900)
- ✓ Procesador: Intel (R) Core™ 2 Duo CPU E4500 1200,00 MHz

3.2.3 Resultados de las Pruebas de Carga

Después de hacerle las pruebas a los escenarios críticos del portal web perteneciente al proyecto VideoWeb se hizo un registro de los resultados en los Listeners Informe Agregado y Ver Árbol de Resultados. A continuación se mostrarán los resultados obtenidos de aplicar las pruebas al escenario Crear publicación de archivo multimedia y las correspondientes a los escenarios: “Eliminar publicación de archivo multimedia”, “Modificar publicación de archivo multimedia” y “Publicar archivo multimedia” serán añadidas en los Anexos # 2, 3 y 4.

Para el Caso de Uso **Crear publicación de archivo multimedia:**

La figura #24 muestra una parte del resultado de algunas de las peticiones HTTP mediante el Árbol de Resultados, aquí se ve como estas peticiones se ejecutan trayendo resultados satisfactorios, ya que las mismas se mostraron en color verde.

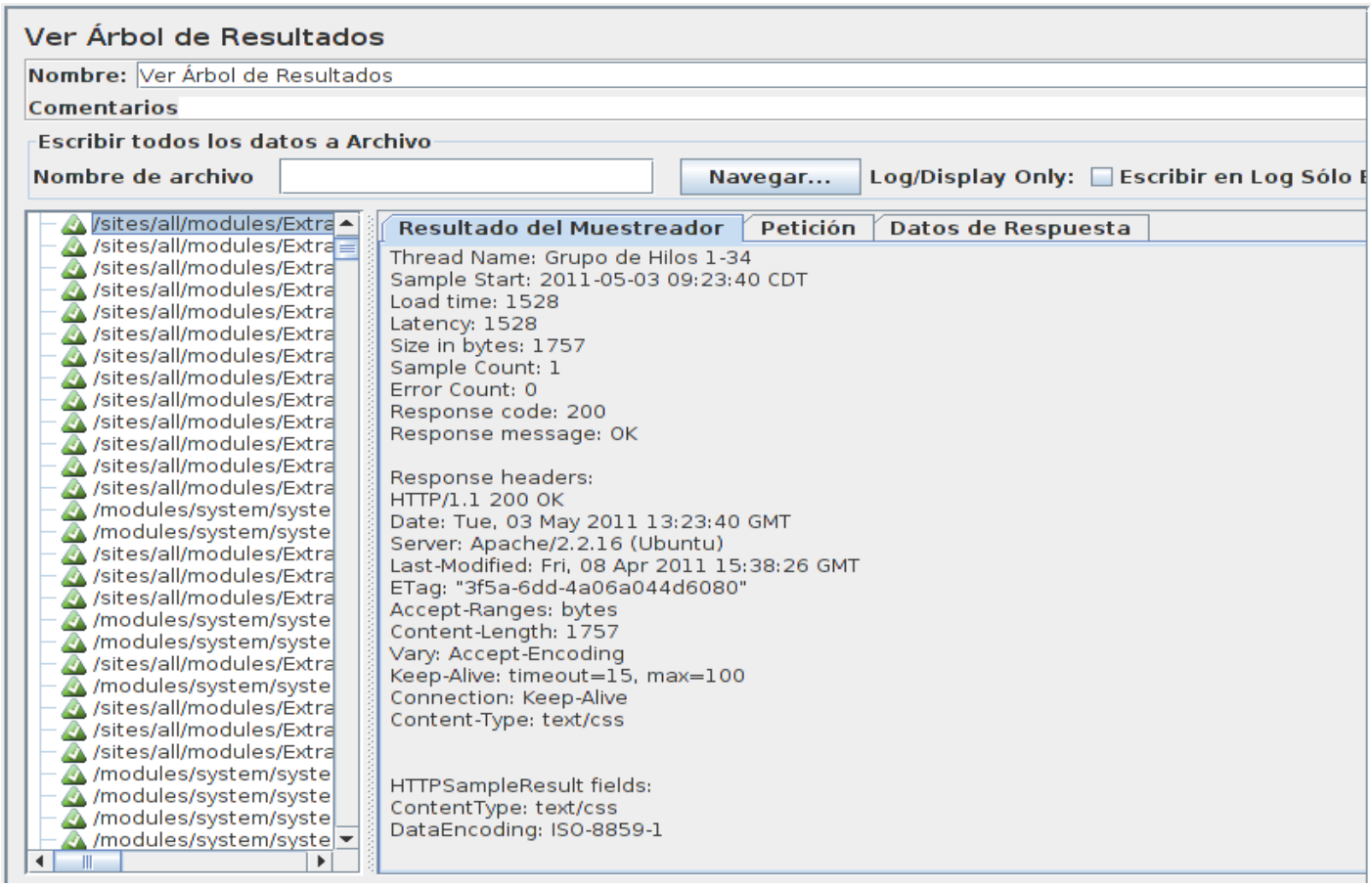


Figura #24: Árbol de Resultado de Crear publicación de archivo multimedia

En la figura anterior se destaca la sección Resultado del Muestreador donde se vieron los aspectos correspondientes a la primera petición HTTP, mostrando dentro de los aspectos más importantes la cantidad de errores **0** (Error count), el código de respuesta **200** (Response code) y como mensaje de respuesta **ok** (Response message) significando esto que el resultado de la petición HTTP analizada fue de manera satisfactorio. A continuación se muestran otros de los resultados obtenidos de la realización de los test al escenario **Crear publicación de archivo multimedia:**

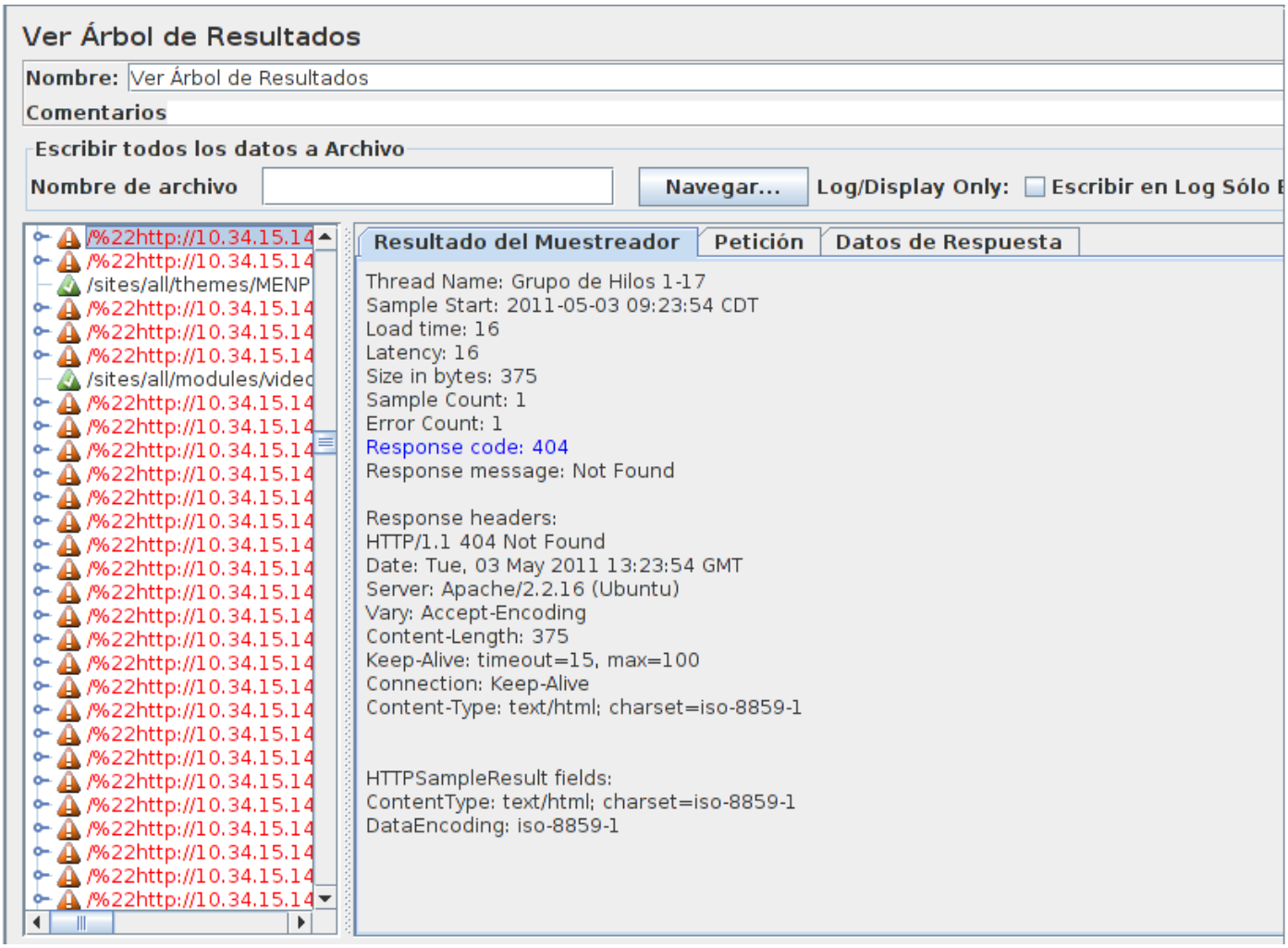


Figura #25: Árbol de Resultado de Crear publicación de archivo multimedia

La figura #25 muestra otros resultados de la misma grabación de prueba pero a otras peticiones destacando aquí un resultado negativo, pues se ve, como, la mayoría de las respuestas presentan error, esto se aprecia viendo el color de las peticiones HTTP. Todas las que están en rojo significan que tuvieron fallo a la hora de cargarse las mismas apreciando como el código de respuesta es 404 y en el mensaje de respuesta se resalta que no funciona (Not Found).

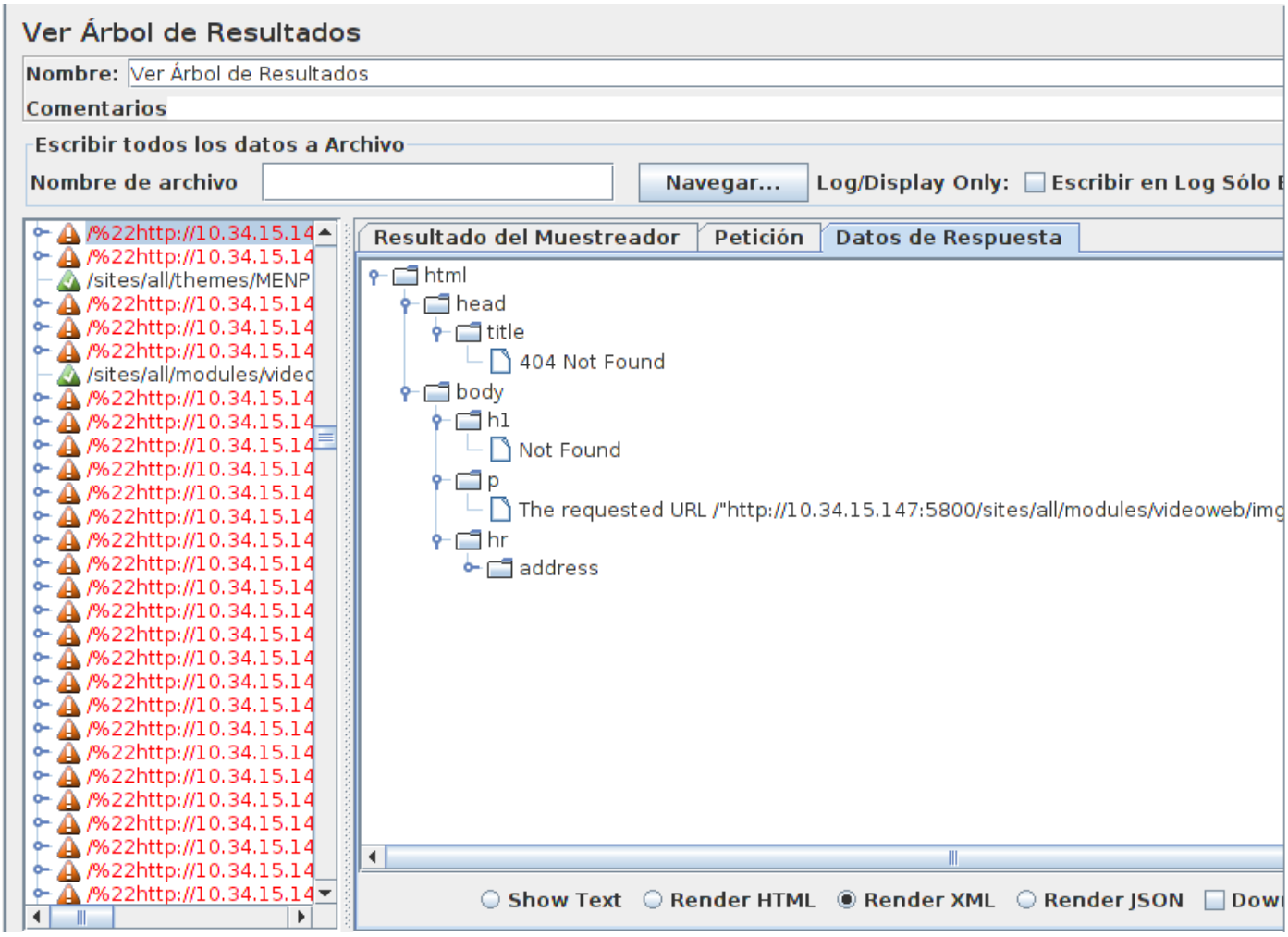
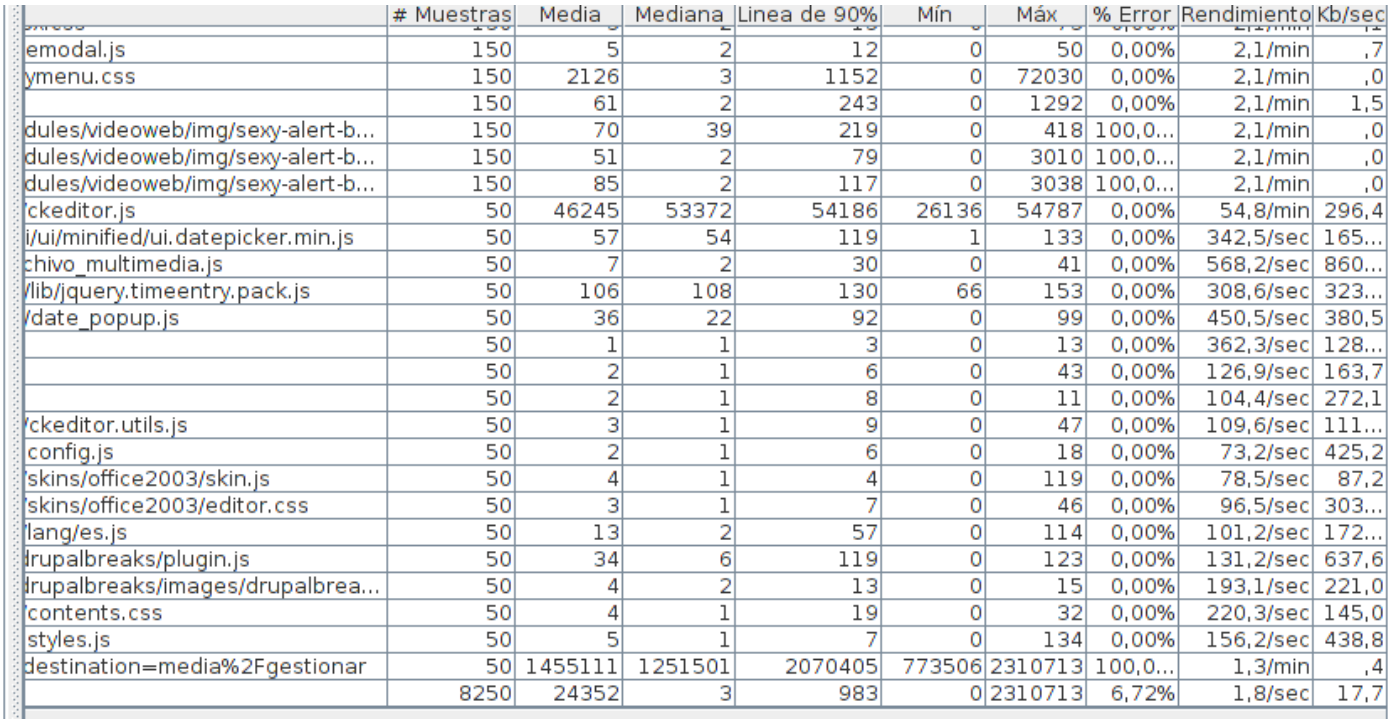


Figura #26: Árbol de Resultado de Crear publicación de archivo multimedia Rendimiento XML

En la figura #25 se mostraron los resultados de un conjunto de peticiones HTTP del Muestreador. En la figura #26, se muestran para las mismas peticiones pero en la sección Datos de Respuesta, aquí se escogió una de las variantes que brinda el **Listener Ver Árbol de Resultado** y fue ver la respuesta en Rendimiento XML (Render XML). En esta opción se detalla bien en que parte de la página es que existe el error. Por ejemplo para la petición señalada en la figura se muestra como en la parte cabecera de la página específicamente en el título se detecta el error del código de respuesta. Además se muestra el mensaje de respuesta señalándolo en el cuerpo de la página.



	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
emodal.js	150	5	2	12	0	50	0,00%	2,1/min	,7
ymenu.css	150	2126	3	1152	0	72030	0,00%	2,1/min	,0
	150	61	2	243	0	1292	0,00%	2,1/min	1,5
dules/videoweb/img/sexy-alert-b...	150	70	39	219	0	418	100,0...	2,1/min	,0
dules/videoweb/img/sexy-alert-b...	150	51	2	79	0	3010	100,0...	2,1/min	,0
dules/videoweb/img/sexy-alert-b...	150	85	2	117	0	3038	100,0...	2,1/min	,0
ckeditor.js	50	46245	53372	54186	26136	54787	0,00%	54,8/min	296,4
i/ui/minified/ui.datepicker.min.js	50	57	54	119	1	133	0,00%	342,5/sec	165...
chivo_multimedia.js	50	7	2	30	0	41	0,00%	568,2/sec	860...
/lib/jquery.timeentry.pack.js	50	106	108	130	66	153	0,00%	308,6/sec	323...
/date_popup.js	50	36	22	92	0	99	0,00%	450,5/sec	380,5
	50	1	1	3	0	13	0,00%	362,3/sec	128...
	50	2	1	6	0	43	0,00%	126,9/sec	163,7
	50	2	1	8	0	11	0,00%	104,4/sec	272,1
/ckeditor.utils.js	50	3	1	9	0	47	0,00%	109,6/sec	111...
config.js	50	2	1	6	0	18	0,00%	73,2/sec	425,2
/skins/office2003/skin.js	50	4	1	4	0	119	0,00%	78,5/sec	87,2
/skins/office2003/editor.css	50	3	1	7	0	46	0,00%	96,5/sec	303...
/lang/es.js	50	13	2	57	0	114	0,00%	101,2/sec	172...
drupalbreaks/plugin.js	50	34	6	119	0	123	0,00%	131,2/sec	637,6
drupalbreaks/images/drupalbrea...	50	4	2	13	0	15	0,00%	193,1/sec	221,0
/contents.css	50	4	1	19	0	32	0,00%	220,3/sec	145,0
styles.js	50	5	1	7	0	134	0,00%	156,2/sec	438,8
destination=media%2Fgestionar	50	1455111	1251501	2070405	773506	2310713	100,0...	1,3/min	,4
	8250	24352	3	983	0	2310713	6,72%	1,8/sec	17,7

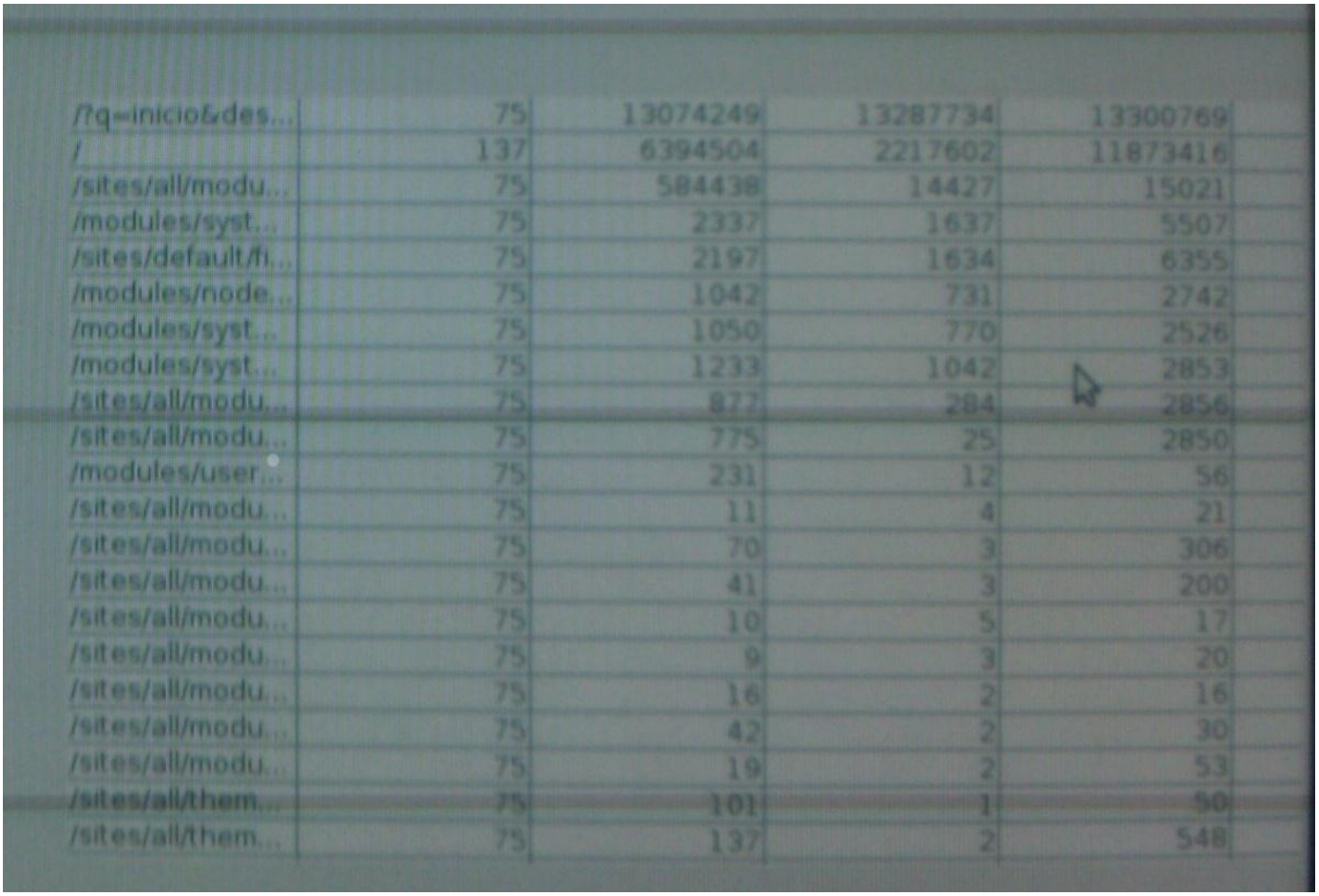
Figura #27: Informe Agregado de Crear publicación de archivo multimedia

En la figura #27 se muestra el otro **Listener** utilizado para recoger los resultados de la prueba en un informe agregado, en este se modela una serie de datos, los cuales exponen el estado en el que se encuentra el software. De esta figura se analizará la última fila que estará resumiendo los 9 aspectos señalados en la parte superior, quedando:

Para el escenario probado se tiene que para 8250 muestras que se le hicieron al servidor el tiempo mínimo fue de 0 segundos de respuesta y el tiempo máximo en cargarse la página fue de 2 310 713 milisegundos, y una línea de 90% de cargarse las páginas satisfactoriamente respondió en 983 milisegundos. Teniendo como media de páginas que se cargaron de manera satisfactoria en un total de 24352, siendo el tiempo promedio que han tardado en cargarse las páginas de 3 milisegundos. Esto implicó que el 6.72% de las páginas no se llegó a cargar de manera satisfactoria, el rendimiento fue de 1.8 segundos por cada petición y la velocidad de carga de las páginas fue de 17.7 kb/seg.

3.2.4 Resultados de las Pruebas de Estrés

La figura #28 muestra el resultado de la realización de la prueba de estrés. Está compuesta por una parte del informe agregado, pues se está simulando la conexión de 75 usuarios concurrentemente. Arrojando como resultado que el servidor no es capaz de soportar esta carga como se aprecia en la figura. Siendo 70 el número máximo de conexiones soportada por la aplicación al unísono.



/?q=inicio&des...	75	13074249	13287734	13300769
/	137	6394504	2217602	11873416
/sites/all/modu...	75	584438	14427	15021
/modules/syst...	75	2337	1637	5507
/sites/default/fi...	75	2197	1634	6355
/modules/node...	75	1042	731	2742
/modules/syst...	75	1050	770	2526
/modules/syst...	75	1233	1042	2853
/sites/all/modu...	75	877	284	2856
/sites/all/modu...	75	775	25	2850
/modules/user...	75	231	12	56
/sites/all/modu...	75	11	4	21
/sites/all/modu...	75	70	3	306
/sites/all/modu...	75	41	3	200
/sites/all/modu...	75	10	5	17
/sites/all/modu...	75	9	3	20
/sites/all/modu...	75	16	2	16
/sites/all/modu...	75	42	2	30
/sites/all/modu...	75	19	2	53
/sites/all/them...	75	101	1	50
/sites/all/them...	75	137	2	548

Figura #28: Resultado de la prueba de estrés

Realizando un resumen de los datos brindados por las pruebas. En un ambiente con las características de software: **Servidor web**: Apache Tomcat 6.0.13, como **Lenguaje utilizado**: PHP y de hardware: con 160 Gb de **Disco Duro**, 1 Gb de **RAM**, con un **Procesador**: Intel ® Core TM 2Duo y la velocidad de la Red de 100 Mbps. Se obtuvo que para un total de 50 usuarios (hilos) en el escenario Crear publicación de archivo

multimedia la aplicación generó un total de 17.7 kb/seg de transferencia de datos lo que incurrió en un rendimiento de 1.8/seg. Para un total de 45 usuarios en el escenario Eliminar publicación de archivo multimedia la aplicación generó un total de 9.9 kb/seg lo que incurrió en un rendimiento de 1.3/seg. Para un total de 35 usuarios en el escenario Modificar publicación de archivo multimedia la aplicación generó un total de 55.4 kb/seg lo que incurrió en un rendimiento de 5.9/seg. Para un total de 50 usuarios en el escenario Publicar archivo multimedia la aplicación generó un total de 20.2 kb/seg lo que incurrió en un rendimiento de 2.1/seg. Se demuestra que la aplicación es inestable ya que para estos usuarios se mantuvo prestando servicios todo el tiempo incurriendo en fallos. Además el tiempo de respuesta del servidor se encuentra por encima del tiempo que se especifica en los requerimientos del sistema.

3.5 Conclusiones del Capítulo

La realización de las pruebas al proyecto ayudó a demostrar la calidad de los productos desarrollados por el mismo y en qué medida éstos cumplen con las expectativas del cliente. Se demostró que con la utilización de las herramientas automáticas para la realización de las pruebas, se ahorra tiempo y recursos al proyecto. Las pruebas en general se realizaron de manera eficiente con una buena organización y todos los procedimientos quedaron reflejados en el presente trabajo, lo que apunta favorablemente a la definición del proceso como una buena práctica de concepción, elaboración y ejecución de las pruebas de carga y estrés de manera automatizada.

Conclusiones Generales

Este trabajo ha demostrado la importancia que tienen las pruebas de carga y estrés de manera automatizada en las aplicaciones desarrolladas por el centro GEySED, demostrando que la aplicación de herramientas en este proceso, la definición de tareas de manera planificada y organizada, que se necesitan para realizar estas pruebas favorece a los equipos de producción ya que la inversión total de tiempo es muy pequeña en comparación a la inversión que se necesitaría si se utilizara personal para simular cualquier cantidad de usuarios necesarios para probar las aplicaciones.

En el trabajo se hace una descripción del proceso de prueba de carga y estrés, así como se presenta una descripción en forma de manual en el capítulo 2 acerca del uso de la herramienta Apache JMeter, este manual está orientado al entendimiento de la herramienta y la importancia de cada uno de los elementos que ella brinda en la elaboración de las pruebas.

Con esta investigación y el manual, cualquier especialista implicado en el rol de prueba puede aplicar este proceso a su entorno de trabajo y obtener mejores resultados en menos tiempo y con menos recursos, ya que los conocimientos que aquí se reflejan se han expresado de manera organizada ejemplificando cada uno de ellos y mostrándolos de manera sencilla.

Con respecto a los resultados de las pruebas aplicadas en el proyecto productivo se considera que han sido de gran importancia, pues han confirmado la calidad de los productos desarrollados, se obtuvo una referencia del nivel de calidad del trabajo realizado en el proyecto VideoWeb perteneciente al Departamento Señales Digitales.

En esta investigación para dar cumplimiento a los objetivos planteados, se realizó:

- ✓ Un estudio bien detallado de las herramientas de Pruebas de Software Automáticas existentes en la actualidad, haciendo énfasis en aquellas que pudieran ser vinculadas, según las tecnologías que manejan los proyectos del Departamento de Señales Digitales.

- ✓ Tomando como punto de partida el estudio realizado, una propuesta de una herramienta, para el uso dentro de los proyectos productivos, que agilizará el proceso de prueba dentro de los mismos.

Recomendaciones

- ✓ Exhortar a los equipos de desarrollo de la Facultad 6 a la utilización de esta herramienta dentro del proceso de Prueba.
- ✓ Crear un repositorio en la Facultad con esta herramienta y otras más, para que los líderes de proyectos puedan contar con ellas para el proceso de prueba en los proyectos productivos.
- ✓ Designar un miembro de cada uno de los proyectos productivos (preferiblemente con el rol de probador) para que se especialice en la utilización del uso de herramientas automáticas, y que éstas sean empleadas correctamente en los proyectos productivos.
- ✓ Se recomienda la creación de una herramienta que esté capacitada para realizar pruebas a aplicaciones web y desktop, pues hoy en día la utilización de aplicaciones desktop está teniendo un auge en la producción de software a nivel mundial.

Referencias Bibliográficas

1. **Pressman, Roger.** *Ingeniería del Software: Un Enfoque Práctico.* España : Mcgraw-hill - España, 2001. 8448132149.

2. *Calidad de Software.* **Lovelle, Juan Manuel Cueva.** España : s.n., 1999.

3. **Pressman, Roger.** *Ingeniería del Software: Un Enfoque Práctico.* s.l. : McGraw-Hill, 2005. ISBN: 9701054733.

4. **López, Carlos.** Gestiopolis.com. *Gestiopolis.com.* [En línea] 2008. [Citado el: 23 de Marzo de 2011.] http://www.gestiopolis.com/recursos/experto/catsexp/pagans/ger/42/calidad.htm.

5. **Rick David Craig, Stefan P, Jaskiel.** *Systematic Software Testing.* Estados Unidos de América : ISBN, 2002. 1-58053-508-9.

6. **Piattini, Mario G.** *Análisis y Diseño de Aplicaciones Informáticas de Gestión. Incluye una Perspectiva de Ingenieria del Software.* s.l. : RA-MA EDITORIAL, 2003. ISBN - 978-84-7897-587-7 .

7. *Testing Computer Software.* **Cem Kaner, Jack Falk, Hunq Q, Nguyen.** s.l. : segunda edición, 1993.

8. *The Art of Software Testing.* **Myers.** s.l. : segunda edición, 2004.

9. **Gutiérrez, Javier Jesús.** *Generación de pruebas de sistema a partir de la especificación funcional.* España : s.n., 2005.

10. *Revista - ATIX - número 16.* **Esteban Saavedra, Joseph Sandoval, Mario Carrion.** 16, 2010.

11. Free Download Manager. *Free Download Manager.* [En línea] [Citado el: 20 de 11 de 2010.] http://www.freedownloadmanager.org/es/downloads/WebLOAD_Professional_59025_p/.

12. **Krahen, Jan Pieter.** El País.com. *El País.com.* [En línea] 23 de 07 de 2010. [Citado el: 15 de 11 de 2010.] http://www.elpais.com/articulo/economia/Pruebas/estres/buenas/elpepueco/20100723elpepueco_1/Tes.

13. **Alvarez, Miguel Angel.** desarrolloweb. *desarrolloweb.* [En línea] 09 de 07 de 2001. [Citado el: 25 de 11 de 2010.] http://www.desarrolloweb.com/articulos/482.php.

14. **Ignacio Esmite, Mauricio Farías, Nicolás Farías, Beatriz Pérez.** *Automatización y Gestión de las Pruebas Funcionales.* Montevideo, Uruguay, : s.n.

Bibliografía

1. **Pressman, Roger.** *Ingeniería del Software: Un Enfoque Práctico.* España: Editorial McGraw-Hill – España, 2007.

2. **Sestoft, Peter.** *Systematic Software Testing. Estados Unidos de América.* 2008.

3. **Craig Rick, Jaskiel Stefan.** *Systematic Software Testing.* 2008.

4. **M Oscar, Carrasco Fernandez.** *Un enfoque actual sobre la calidad de software.* 1995.

5. **Artola, Luis.** *Tipos de pruebas automatizadas de software.* 2009.

6. **Alvarez, Miguel Angel.** desarrolloweb. *desarrolloweb.* [En línea] 09 de 07 de 2001. [Citado el: 25 de 11 de 2010.] http://www.desarrolloweb.com/articulos/482.php

7. http://www.consumer.es/web/es/tecnologia/internet/2009/10/19/188347.php. Última actualización: 30 de octubre del 2009.

8. **Pressman, Roger.** *Ingeniería del Software: Un Enfoque Práctico.* España : Mcgraw-hill - España, 2001. 8448132149.

9. *Calidad de Software.* **Lovelle, Juan Manuel Cueva.** España: s.n., 1999.

10. **Pressman, Roger.** *Ingeniería del Software: Un Enfoque Práctico.* s.l.: McGraw-Hill, 2005. ISBN: 9701054733.

11. **Rick David Craig, Stefan P, Jaskiel.** *Systematic Software Testing.* Estados Unidos de América: ISBN, 2002. 1-58053-508-9.

12. **Piattini, Mario G.** *Análisis y Diseño de Aplicaciones Informáticas de Gestión. Incluye una Perspectiva de Ingeniería del Software.* s.l.: RA-MA EDITORIAL, 2003. ISBN - 978-84-7897-587-7.

13. *Testing Computer Software.* **Cem Kaner, Jack Falk, Hunq Q, Nguyen.** S.l.: segunda edición, 1993.

14. *The Art of Software Testing.* **Myers.** S.l.: segunda edición, 2004.

15. **Gutiérrez, Javier Jesús.** *Generación de pruebas de sistema a partir de la especificación funcional.* España: s.n., 2005.

16. *Revista - ATIX - número 16.* **Esteban Saavedra, Joseph Sandoval, Mario Carrion.** 16, 2010.

17. Free Download Manager. *Free Download Manager.* [En línea] [Citado el: 20 de 11 de 2010.] http://www.freedownloadmanager.org/es/downloads/WebLOAD_Professional_59025_p/.

18. **Krahen, Jan Pieter.** El País.com. *El País.com.* [En línea] 23 de 07 de 2010. [Citado el: 15 de 11 de 2010.] http://www.elpais.com/articulo/economia/Pruebas/estres/buenas/elpepueco/20100723elpepueco_1/Tes.

19. **Alvarez, Miguel Ángel.** desarrolloweb. *desarrolloweb.* [En línea] 09 de 07 de 2001. [Citado el: 25 de 11 de 2010.] http://www.desarrolloweb.com/articulos/482.php.

20. **Ignacio Esmite, Mauricio Farías, Nicolás Farías, Beatriz Pérez.** *Automatización y Gestión de las Pruebas Funcionales.* Montevideo, Uruguay: s.n. http://www.ces.com.uy/documentos/AutomCACIC07.pdf.

21. **López, Carlos.** Gestiopolis.com. *Gestiopolis.com.* [En línea] 2008. [Citado el: 23 de Marzo de 2011.] http://www.gestiopolis.com/recursos/experto/catsexp/pagans/ger/42/calidad.htm.