Universidad de las Ciencias Informáticas

Facultad 4

Título: Procedimiento para pruebas de rendimiento de Carga y Estrés al Sistema Único de Aduanas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Anay Caridad Barban Frómeta

Paula Zenaida Hernández Figueredo

Tutor: Ing. Maurice Cabrejas Martínez

Ciudad de La Habana, junio 2009

"Año del 50 aniversario del triunfo de la Revolución"

"En la vida hay que tener fuerza para cambiar las cosas que pueden ser cambiadas, paciencia para soportar las que no, y sabiduría para distinguirlas."

San Agustín

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Paula Z. Hernández Figueredo

_____

Anay C. Barban Frómeta

_____

Ing. Maurice Cabrejas Martínez

_____

AGRADECIMIENTOS

A nuestros padres por su apoyo incondicional, a nuestros amigos, a la Revolución, y a todas aquellas personas que de una forma u otra hicieron posible la realización de este trabajo.

A todos ustedes muchas gracias.

DEDICATORIA

De Anay:

A mi mamá Esther y a mi abuela María que han sido toda la inspiración que he tenido para llegar hasta aquí.

A mis tíos Daniel y Ernesto, mi tía Graciela y primos David, Daymi y Sheila que me han apoyado tanto y han confiado en mí.

A mi neni Raymond que durante estos 5 años ha sido mi guía y mi apoyo en todo momento.

De Paula:

A mi mamá Alicia y mi papá Angel por todo su amor y comprensión, por ser lo más importante de mi vida.

A mis abuelos Paula y Octavio por una niñez tan maravillosa.

A mis hermanas de la vida Yaima y Yanet por todos los momentos que hemos pasado juntas.

RESUMEN

Este trabajo está centrado en el desarrollo y aplicación de un procedimiento para realizar pruebas de rendimiento al Sistema Único de Aduanas (SUA), con el objetivo de lograr que el producto tenga una mayor calidad y que exista una documentación que quede como constancia del trabajo realizado, que la misma pueda servir para futuras verificaciones de la calidad del software. Se diseñó una plantilla para casos de prueba de rendimiento aplicable al software utilizado para documentar las pruebas realizadas. Todo el procedimiento fue desarrollado bajo la metodología de desarrollo RUP, Proceso Unificado de Desarrollo de Software, para lograr una organización óptima en el trabajo.

Los aportes principales de este trabajo consisten en la obtención de toda la documentación organizada por pasos del procedimiento aplicado y que a partir de estas pruebas, una vez que se realice el proceso de revisión y eliminación de errores, el software tendrá una mayor calidad.

PALABRAS CLAVE

Calidad, pruebas, casos de prueba, pruebas de rendimiento.

Tabla de contenido

INTRODUCCIÓN

La producción de software ha alcanzado en la actualidad gran importancia a nivel mundial, demandando la creación de mejores software en menos tiempo y menos costo, aplicando el uso de técnicas y metodologías que permiten una mayor productividad.

Uno de los factores por los que se ha incrementado la calidad del software a nivel mundial es la exigencia de los clientes. La calidad del software juega un papel importante dentro del desarrollo del software influyendo positivamente en la decisión de un cliente a la hora de escoger el producto que necesita.

Cuba presenta actualmente un crecimiento en la producción de software, y con esto un incremento en el interés por garantizar calidad en el software que se produce. Teniendo en cuenta la importancia de la vinculación de todas las ramas de la economía cubana con el mundo informático es de primordial interés para el Estado Cubano la consolidación de esta vinculación, no solo por los beneficios que trae desde el punto de vista del desarrollo de sistemas para el uso interno, sino también con el objetivo de introducirse en el mercado mundial.

La Universidad de las Ciencias Informáticas forma parte de la industria cubana de software. Sus estudiantes tienen la peculiaridad de vincular en sus actividades el estudio con la producción, ayudando de esta forma a dar respuesta a las solicitudes de desarrollo de aplicaciones que llegan continuamente a la Universidad, ya sean de origen nacional o internacional, por lo que se han desarrollado una serie de actividades con el objetivo de garantizar que los productos que se realizan tengan la calidad requerida. La calidad de un producto puede ser medible y viene dada por la correcta implementación de los requisitos del software y la corrección de errores.

Actualmente la Universidad de las Ciencias informáticas y la Aduana General de la República, trabajan en conjunto en el desarrollo de aplicaciones para mejorar la informatización de la Aduana, las cuales son desarrolladas en la facultad 4. Uno de los pasos para garantizar que el producto sea óptimo es el desarrollo de un correcto procedimiento de pruebas de rendimiento al software para mejorar la calidad del mismo. Sin embargo no existe una planificación concreta y eficaz de un procedimiento para realizar pruebas de rendimiento para el cumplimiento de estos objetivos, y debido a la envergadura e importancia que representa es conveniente la creación de un procedimiento para la realización de pruebas de rendimiento y de esta forma verificar la calidad del software. Basándose en lo anterior surge la interrogante:

¿Qué procedimiento seguir para realizar las pruebas de rendimiento al Sistema Único de Aduanas?

El **objetivo general** que se persigue con la realización de este trabajo es elaborar un procedimiento que permita planificar, realizar, medir, controlar y documentar la ejecución de pruebas de rendimiento al Sistema Único de Aduanas.

El **objeto de estudio** está enmarcado en el análisis de procedimientos de pruebas de rendimiento para la gestión de la calidad del software.

El **campo de acción** está definido por las pruebas de rendimiento al Sistema Único de Aduanas.

Como objetivos específicos se plantean:

Elaborar y desarrollar una propuesta para realizar pruebas de rendimiento.

Definir el diseño de los casos de pruebas necesarios para evaluar los productos.

Definir un procedimiento para realizar pruebas de rendimiento.

Llegar a conclusiones sobre el cumplimiento de los objetivos del procedimiento para pruebas de rendimiento.

Las principales tareas que se realizan en este trabajo son:

Describir las principales actividades que requieren las pruebas de rendimiento.

Análisis del entorno donde se aplicarán las pruebas.

Estudio de las modalidades de las pruebas de rendimiento.

Estudiar métodos de planificación, medición y control de pruebas.

Estudiar métodos de documentación de pruebas.

Definir las pruebas de rendimiento a aplicar.

Definir métodos de planificación, medición y control de las pruebas a aplicar.

Definir un método de documentación de pruebas.

Estructura del documento

El presente trabajo se encuentra estructurado por tres capítulos y anexos, en los cuales se exponen todo el trabajo investigativo y práctico realizado:

Capítulo 1: En este capítulo se aborda fundamentalmente el estado actual de las actividades relacionadas con las pruebas. Se realiza además un estudio sobre metodologías y conceptos fundamentales del proceso de pruebas de software. La calidad del software, los niveles de prueba, los tipos de pruebas por niveles, las técnicas y los pasos para un correcto diseño de pruebas.

Capitulo 2: *S*e explica el procedimiento de pruebas de rendimiento desarrollado donde se plantea una propuesta de plantilla para documentar los casos de prueba para la aplicación de las pruebas de rendimiento al Sistema Único de Aduanas.

Capítulo 3: El procedimiento desarrollado se aplica en el Módulo de Información Adelantada de Pasajeros del Sistema Único de Aduanas (API), se realiza la evaluación de los resultados para la ejecución de las pruebas de rendimiento y se exponen los errores encontrados a partir de los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Una de las problemáticas más grandes que se presentan en el desarrollo de sistemas de software es el aseguramiento de la calidad. La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software. Para determinar dicho nivel de calidad se deben efectuar pruebas de software, estas son una vía óptima para la detección y prevención de errores durante todos los ciclos de desarrollo del mismo, permitiendo comprobar el grado de cumplimiento respecto a las especificaciones iniciales del sistema. La enorme cantidad de tiempo y esfuerzo que requiere la fase de pruebas hoy en día se calcula que representa más de la mitad del valor total del producto, pues requiere un tiempo similar al de la programación, y el coste de corregir los errores crece exponencialmente según avanza el proyecto.

En este capítulo se presenta el estado del arte de las pruebas de rendimiento, teniendo en cuenta los objetivos, son analizados algunos temas como: la relación que existe entre las pruebas y la calidad de un software, los objetivos y estrategias a seguir a la hora de aplicar las mismas, los tipos de pruebas que existen, métodos, niveles, herramientas que se utilizan para realizar pruebas, las metodologías posibles a usar y como se aplican.

1.2 Calidad de software

La calidad del software se define como la “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” (Pressman)

Existen estándares o metodologías que definen un conjunto de criterios de desarrollo para guiar la forma en que se debe aplicar la Ingeniería del Software, si no se sigue ninguna metodología siempre habrá falta de calidad ya que precisamente el único fin que persiguen todas las metodologías y herramientas es producir software de alta calidad.

Las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo y sin un buen proceso de desarrollo es casi imposible obtener un buen producto. El control de calidad es una serie de inspecciones, revisiones y pruebas utilizadas a lo largo del proceso del

software para asegurar que cada producto cumple con los requisitos que le han sido asignados. (Pressman)

La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. Es posible medir los principales atributos que forman o caracterizan a un software de buena calidad. La idea es que la calidad del software se identifica por ciertos atributos: fiabilidad, flexibilidad, robustez, comprensión, adaptabilidad, modularidad, complejidad, portabilidad, usabilidad, reutilización, eficiencia, y sin lugar a dudas es posible medir cada uno de ellos, y por consiguiente, caracterizar o medir la calidad del software en cuestión. Teniendo en cuenta que todos los atributos por muy difíciles que sean pueden ser medidos relacionándolos con otro que sea más fácil de medir. (Guzmán Arenas, 2003)

- La fiabilidad (software confiable, pocos errores) se mide a través del número de mensajes de error, mientras más mensajes existan, contiene menos errores el software.
- La flexibilidad (capacidad de adaptación a diferentes tipos de uso, a diferentes ambientes de uso) o adaptabilidad.
- La robustez (pocas fallas catastróficas, el sistema “no se cae”) se mide a través de pruebas y uso prolongado.
- La comprensión (capacidad de entender lo que el sistema hace) se mide viendo la cantidad de comentarios que posee el software, y la extensión de sus manuales de usuarios.
- El tamaño se mide en bytes, o lo que es lo mismo el espacio que ocupa en memoria, (esta medición no tiene objeción, se mide lo que se quiere medir).
- La eficiencia de un programa se mide en segundos, es la rapidez en su ejecución (esta medición no tiene objeción, se mide lo que se quiere medir).
- La modularidad de un programa se mide contando el número de módulos que lo conforman.
- La complejidad de un programa se mide contando el número de anidaciones en expresiones o postulados, (es lo que se le llama complejidad ciclomática).
- La portabilidad es la facilidad con que se eche a andar en otro sistema operativo distinto al que fue creado. Se mide preguntando a usuarios que han hecho estos trabajos.

- La usabilidad de un programa es alta cuando el programa aporta gran valor agregado a nuestro trabajo. “Es indispensable contar con él”. Se mide viendo que porcentaje de nuestras necesidades cubre ese programa.

- La reutilización de un programa se mide por la cantidad de veces que partes del mismo se han reutilizado en otros proyectos de desarrollo de software.

- La facilidad de uso (ergonomía) caracteriza a los programas que no cuesta trabajo aprender, que se amoldan al modo intuitivo de hacer las cosas. Se mide observando la cantidad de pantallas que interaccionan con el usuario, y su sofisticación.

Cuando se desarrolla un software es difícil asegurar que en este no habrá errores. La calidad de un sistema software es algo subjetivo que depende del contexto y del objeto que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar medidas o pruebas que permitan comprobar el grado de cumplimiento de las especificaciones iniciales del sistema. Las Pruebas de software se integran en las diferentes fases del Ciclo del software dentro de la Ingeniería de software. Los errores pueden comenzar a aparecer desde el inicio del proceso de elaboración de los requisitos y continuar surgiendo en las etapas posteriores del diseño y desarrollo. La Calidad de software es la vía de comprobación que permitirá verificar el cumplimiento de los objetivos de eficiencia que se planificaron para el software, garantizando un proceso ordenado y objetivo que conlleve a conclusiones exitosas garantizando la calidad óptima del software que será probado.

1.3 Pruebas de software

Dentro de cada una de las etapas de desarrollo de un software las pruebas son fundamentales ya que a partir de ellas es posible controlar que los productos cumplan requisitos mínimos de operabilidad además de garantizar la calidad de estos productos.

“Las pruebas constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.” (IEEE, 1990)

Las pruebas se pueden traducir como una revisión del sistema con el objetivo de encontrar problemas antes que éstos sean encontrados por el cliente final. El éxito de su ejecución puede mejorar la percepción de calidad del usuario final, evitando que un defecto grave se entregue juntamente con el sistema.

De manera general las pruebas de software deben garantizar la calidad del producto desarrollado, y para lograr esto es necesario: (Jacobson, y otros, 2004)

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

- Diseñar e implementar las pruebas creando los casos de prueba que especifican qué probar, creando componentes de pruebas ejecutables para automatizar las pruebas.

- Realizar las diferentes pruebas y manejar los resultados de cada una de forma sistemáticas. Las construcciones en las que se detectan 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.

La prueba del software es un elemento de un tema más amplio que, a menudo, es conocido como verificación y validación (V&V). La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. La validación se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requisitos del cliente. (Pressman)

Resumiendo se puede decir que las pruebas de software son aquellas encargadas de demostrar que un sistema que es ejecutado bajo ciertas condiciones específicas está incapacitado para ser liberado. 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.

1.3.1 Objetivo de las pruebas de software

Probar es el proceso de ejecutar un programa con el fin de encontrar errores o fallas.

Dentro de los objetivos fundamentales que se persiguen al aplicarles las pruebas a un software se encuentran los siguientes: (López Quesada, 2005)

- Un buen caso de prueba es aquel que tiene una alta probabilidad de descubrir un error no encontrado hasta entonces.

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Brindar un mayor nivel de confiabilidad en los productos que se van generando.
- Detectar fallas o errores.
- Aumentar la calidad del producto final.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Los objetivos planteados anteriormente desacreditan la filosofía acerca de otorgarle éxito a una prueba cuando esta no descubre errores. Por el contrario tiene éxito si encuentra un error no detectado hasta el momento.
Las pruebas no se realizan solamente al código también a la documentación y a la ayuda.
El objetivo principal es diseñar pruebas que sistemáticamente reflejen diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.
Si las pruebas se llevan a cabo con éxito se descubrirán errores en el software, dándole mayor fiabilidad.
Es importante tener en cuenta una frase de Pressman:
“La prueba no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el software”. (Pressman)

1.3.2 Principios de las pruebas

Las pruebas se rigen por una serie de principios, una buena comprensión de estos facilitará el posterior uso de los métodos en un efectivo diseño de casos de prueba (López Quesada, 2005)

Hacer un seguimiento de las pruebas hasta los requisitos del cliente (trazabilidad).

Deben ser planificadas antes de que empiecen.

Plantear y diseñar las pruebas antes de generar ningún código.

El principio de Parapeto es aplicable a las pruebas de software (“Donde hay un defecto, hay otros”).

Empezar las pruebas en módulos individuales y avanzar hasta probar el sistema entero, empezar por “lo pequeño” y progresar hacia “lo grande”.

No son posibles las pruebas exhaustivas.

No deben realizarse planes de prueba suponiendo que prácticamente no hay defectos en los programas y, por tanto, dedicando pocos recursos a las pruebas.

Se puede decir que los principios constituyen para las pruebas, requerimientos a seguir para certificar su éxito: detectar la mayor cantidad de errores.

1.3.3 Tareas a realizar para probar un software

Hay una serie de actividades, que se deben realizar para conseguir la ejecución de las pruebas, estas son: (Vegas, y otros, 2005)

Diseño de las pruebas: Comprende la identificación de la técnica o técnicas de pruebas que se utilizarán para probar el software. Distintas técnicas de prueba ejecutan diferentes criterios como guía para realizar las pruebas.

Generación de los casos de prueba: Consiste en la confección de los distintos casos de prueba según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso para detectar un posible fallo en el programa. Los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba.

Definición de los procedimientos de la prueba: Conlleva a una especificación de cómo se va a llevar a cabo el proceso, quién lo va a realizar y cuándo.

Ejecución de la prueba: Es el momento de aplicar los casos de prueba generados previamente e identificar los posibles fallos producidos al comparar los resultados esperados con los resultados obtenidos.

1.3.4 Importancia de probar

El desarrollo de software como actividad humana a fin no es perfecto, aun cuando el progreso sea sustancial. Inherente al desarrollo de software hay toda una serie de deficiencias al programador, solo detectadas si el producto es probado. Las pruebas al software arrojan un panorama claro de las deficiencias detectadas en el sistema y de los riesgos que asocian a la organización, y aunque esto

directamente no mejora la calidad del producto si permite que la administración tome medidas a la hora de la asignación de recursos para elevar la misma.

A través de las pruebas se puede comprobar que las especificaciones establecidas están siendo cumplidas con las funcionalidades que presenta el software, de igual manera que los datos que están siendo arrojados sean un indicador de la fiabilidad del sistema.

La prueba de software es un elemento crítico e imprescindible para la garantía de la calidad, y de ahí la necesidad de aplicarla. (Pressman)

Dentro de las causas que fundamentan las pruebas, las tres más sobresalientes son: (Fernández, 2002)

Propensión a equivocarse: El ser humano es propenso a cometer equivocaciones; estas se manifiestan en diversos problemas contenidos en los modelos (defectos o faltas) y pueden manifestarse como fallas en tiempo de ejecución.

Fallas de hardware: La infraestructura empleada para el desarrollo de software (hardware, sistemas operativos, compiladores). No está exenta de fallas, lo que introduce defectos adicionales o permite que subsistan inadvertidos los que introdujo el desarrollador.

Creatividad del desarrollo: El desarrollo de software es una labor creativa y por ello es común que el producto desarrollado no coincida con el modelo contenido en las especificaciones.

1.3.5 Procedimiento de pruebas

Un procedimiento de prueba especifica cómo realizar uno o varios casos de prueba o partes de estos. Este puede ser una instrucción para un individuo sobre cómo va a realizar un caso de prueba manualmente o puede ser una especificación de cómo interaccionar manualmente con una herramienta de automatización de las pruebas para crear componentes ejecutables de pruebas.

El cómo llevar a cabo un caso de prueba puede ser especificado por un procedimiento de prueba pero es a menudo útil reutilizar un procedimiento de prueba para varios casos de prueba y reutilizar varios procedimientos para un caso de prueba.

1.3.6 Estrategias de prueba

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. La estrategia proporciona un mapa que describe los pasos que hay que llevar a cabo como parte de la prueba, cuándo

se deben planificar y realizar esos pasos, y cuánto esfuerzo, tiempo y recursos se van a requerir. Por lo tanto cualquier estrategia de pruebas debe incorporar la planificación de la prueba, el diseño de los casos de prueba, la ejecución de las pruebas y la agrupación y evaluación de los datos resultantes.

Para aplicar pruebas al software se deben seguir un conjunto de estrategias para lograr que estas se hagan en el menor tiempo posible y con la calidad requerida, además de lograr que arrojen los resultados esperados.

Existen características que se pueden aplicar a todas las estrategias de pruebas, muchas de ellas definidas por Pressman.

Dentro de las características generales de las estrategias de prueba se encuentran (Pressman):

1. La prueba comienza en el nivel de módulo y trabaja "hacia fuera", hacia la integración completa del sistema completo.

2. En diferentes puntos es adecuada la utilización de técnicas de prueba distintas.

3. La prueba la lleva a cabo el que desarrolla el software y para grandes proyectos, un grupo de prueba independiente.

4. La prueba y la depuración son actividades diferentes, pero la depuración puede entrar en cualquier estrategia de prueba.

Se puede concluir que en un proyecto la prueba en ocasiones requiere mayor esfuerzo que cualquier otra actividad de la Ingeniería de Software; si se efectúa sin un plan estratégico, el tiempo se desaprovecha y el esfuerzo es consumido innecesariamente y en el peor de los casos, los errores inadvertidos quedarán sin detectar, por tanto, puede ser muy conveniente establecer una estrategia sistemática para probar el software.

1.3.7 Niveles de prueba

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Cada una de las pruebas se realiza en determinados momentos del ciclo de vida del software. Teniendo en cuenta esto, las pruebas se agrupan por niveles, de acuerdo con las diferentes etapas del proceso de desarrollo:

1.3.7.1 Pruebas a nivel de Unidad

Es la escala más pequeña de la prueba, está basada en la funcionalidad de los módulos del programa, como funciones, procedimientos, módulos de clase. En ciertos sistemas también se verifican o se prueban los drivers y el diseño de la arquitectura. Para probar los componentes implementados como unidades individuales, se realizan las pruebas de especificación o de caja negra, que verifica el comportamiento de la unidad observable externamente y pruebas de estructura, o de caja blanca, que verifica la implementación interna de la unidad. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera.

Los casos de prueba pueden cubrir errores como: (Pressman)

- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o de procedencia incorrectos.
- Igualdad esperada cuando los errores de precisión la hacen como probable.
- Variables o comparaciones incorrectas.
- Terminación de bucles inapropiada o inexistente.
- Fallo de salida cuando se encuentra una iteración divergente.
- Variables de bucles modificadas de forma inapropiada.

1.3.7.2 Pruebas a nivel de Integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño (Pressman). Estas pruebas son ejecutadas para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación, se diseñan para descubrir errores en las especificaciones de las interfaces de los paquetes y son responsabilidad de desarrolladores y de independientes, sin solaparse las pruebas.

Las pruebas de integración se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como conjunto.

1.3.7.3 Prueba Independiente

Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores. El objetivo de estas pruebas es proporcionar una perspectiva diferente y en un ambiente más rico que los desarrolladores una vez que la arquitectura del software este completa. Una vista de la prueba independiente es la prueba independiente de los stakeholder, que son pruebas basadas en las necesidades y preocupaciones de los stakeholders.

1.3.7.4 Prueba a nivel de Desarrollador

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas han sido consideradas solo para la prueba de unidad, aunque en algunos casos pueden ejecutar pruebas de integración. Se recomienda que cubran más que las pruebas de unidad.

1.3.7.5 Pruebas a nivel de Aceptación

Son las pruebas finales antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. La mayoría de los desarrolladores de productos de software llevan a cabo un proceso denominado pruebas Alfa o Beta para descubrir errores que solo el usuario final puede descubrir.

1.3.7.6 Prueba a nivel de Sistema

Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.

Las pruebas de sistemas caen fuera del ámbito del proceso de software y no las realiza únicamente el desarrollador del software. Sin embargo los pasos durante el diseño del software y durante la prueba pueden mejorar enormemente la posibilidad de éxito de las pruebas de sistema. Las pruebas de sistema principalmente se centran en verificar la interacción de los actores con el sistema, por lo que a menudo los casos de pruebas se obtienen a partir de las descripciones de los casos de uso. Aunque también se le aplican pruebas al sistema como un todo.

Se puede resumir que todos los niveles de prueba son importantes a la hora de verificar que el software se ha desarrollado correctamente, pero para verificar que se ha cumplido con los requisitos de rendimiento especificados el nivel de prueba a usar es el de pruebas a nivel de sistema.

1.3.8 Tipo de pruebas por niveles

Las pruebas en conjunto tienen como objetivo general verificar y validar un software, independientemente de las características y el entorno donde se desarrollen, además de los recursos y los factores vinculados al proceso de desarrollo. A continuación se exponen algunos ejemplos ordenados por niveles:

1.3.8.1 Pruebas a nivel de Unidad

Prueba de Interfaz

Esta prueba es realizada después que los módulos o subsistemas son integrados para la creación de un sistema completo. Cada módulo o subsistema contiene una interfaz definida que es llamada por otros componentes del sistema. Esta prueba detecta los errores introducidos en el sistema por errores de las interfaces o suposiciones inválidas sobre las interfaces.

Caminos independientes

Una vez que se ejerciten todos los caminos independientes o básicos de la estructura de control se asegura probar con su ejecución, al menos una vez, todas las sentencias del mismo.

Prueba de estructura de datos

El análisis de la estructura de datos locales tiene el propósito de comprobar si los datos que se mantienen temporalmente conservan su integridad durante todos los pasos de ejecución del algoritmo.

Condiciones límites

Se prueban las condiciones límites para verificar la correcta ejecución del módulo en los límites establecidos de condiciones como restricciones de procesamiento.

1.3.8.2 Pruebas a nivel de Integración

Integración ascendente

En este tipo de prueba se integran y prueban los componentes de los niveles más bajos de la estructura del programa antes del desarrollo de los niveles altos. La integración de los modelos de abajo hacia arriba, posibilita la disponibilidad del proceso requerido de los módulos y la eliminación de la necesidad de resguardos.

Integración descendente

La prueba de integración parte del módulo de control principal (mayor nivel) para luego ir incorporando los módulos subordinados progresivamente. La selección del próximo módulo a incorporar es libre, solo exige que haya sido probado anteriormente. Existen principalmente dos métodos para la selección de módulos:

- Primero en profundidad: Primero se mueve verticalmente en la estructura de módulos.

- Primero en anchura: Primero se mueve horizontalmente en la estructura de módulos.

Prueba de regresión

Esta prueba es aplicada luego de la prueba de integración, en la medida que se añade un nuevo módulo. Abarcan un conjunto de pruebas realizadas anteriormente para comprobar que los cambios introducidos por el módulo añadido no hayan traído problemas colaterales no surgidos hasta el momento. Tienden a aumentar a medida que se progresa en la integración de los módulos, por lo que se sugiere sean diseñadas para incluir solo aquellas pruebas que traten una o más clases de errores en cada una de las funcionalidades del programa.

Prueba de humo

Se aplica en productos software desarrollados por paquetes, generalmente proyectos críticos por tiempo. Con su ejecución se prueba al software de principio a fin y aunque no sea exhaustiva descubre errores importantes. Permite minimizar los riesgos de integración, perfeccionar la calidad el producto, simplificar el diagnóstico y la corrección de errores además de seguir con la observación del progreso final.

1.3.8.3 Pruebas a nivel de Sistema

Pruebas Funcionales

Entre todos los tipos de pruebas que se realizan en un sistema está el tipo que evalúa la funcionalidad de este, estas son las llamadas pruebas funcionales ya que se centran en las funciones, las entradas y las salidas. Su objetivo fundamental es asegurar el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados, estas pruebas tienen como meta:

- Verificar el procesamiento, recuperación e implementación adecuada de las reglas del negocio.
- Verificar la apropiada aceptación de datos.

Estas pruebas se realizan aplicando las pruebas de caja negra ejecutando cada caso de uso, flujo de caso de uso, o función, usando datos válidos e inválidos, para verificar lo siguiente:

- ❖ Que se aplique apropiadamente cada regla de negocio.
- ❖ Que los resultados esperados ocurran cuando se usen datos válidos.
- ❖ Que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos.

Pruebas de Seguridad

Las pruebas de seguridad garantizan que los usuarios están restringidos a funciones específicas o su acceso está limitado únicamente a los datos que están autorizados a acceder, sólo aquellos usuarios autorizados a acceder al sistema son capaces de ejecutar las funciones del sistema. Su objetivo fundamental es comprobar los niveles de seguridad lógica del sistema.

Pruebas de esfuerzo

Estas pruebas comprueban el desempeño y fiabilidad del sistema una vez integrado. Se basan en la respuesta del sistema ante el procesamiento de la carga para la cual fue diseñado. La prueba consiste en el incremento de la carga de forma constante hasta que el desempeño del sistema sea inaceptable.

Pruebas de rendimiento

Las pruebas de rendimiento pueden servir para diferentes propósitos. Pueden demostrar que el sistema cumple los criterios de rendimiento. Pueden comparar dos sistemas para encontrar cuál de ellos funciona mejor. O pueden medir que partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico, los ingenieros de software utilizan herramientas como pueden ser monitorizaciones que midan qué partes de un dispositivo o software contribuyen más al mal rendimiento o para establecer niveles (y umbrales) del mismo que mantengan un tiempo de respuesta aceptable.

Es fundamental para alcanzar un buen nivel de rendimiento de un nuevo sistema, que los esfuerzos en estas pruebas comiencen en el inicio del proyecto de desarrollo y se amplíen durante su construcción. Cuanto más se tarde en detectar un defecto de rendimiento, mayor es el coste de la solución. Esto es cierto en el caso de las pruebas funcionales, pero mucho más en las pruebas de rendimiento, debido a que su ámbito de aplicación es de principio a fin.

En las pruebas de rendimiento, a menudo es crucial (y con frecuencia difícil de conseguir) que las condiciones de prueba sean similares a las esperadas en el uso real. Esto es, sin embargo, casi imposible en la práctica. La razón es que los sistemas en producción tienen un carácter aleatorio de la carga de trabajo y aunque en las pruebas se intente dar lo mejor de sí para imitar el volumen de trabajo que pueda tener el entorno de producción, es imposible reproducir exactamente la variabilidad de ese trabajo, salvo en el sistema más simple. Dentro de los elementos que son comprobados, con las pruebas de rendimiento, se pueden encontrar:

- La determinación de los tiempos de respuesta.
- La verificación del espacio que ocupa el módulo en disco o memoria.

- La verificación del flujo de datos que se genera a través de un canal de comunicaciones.

Dentro de las pruebas de rendimiento se pueden encontrar otras pruebas más específicas como son:

- ✓ **Prueba de carga**

Encargada de probar el funcionamiento del software bajo condiciones extremas. Estudia la especificación del software, las funciones que debe realizar, las entradas y las salidas analizando los valores límites (AVL). (Vega Jara, y otros, 2004)

- ✓ **Prueba de resistencia (Estrés)**

Estas pruebas están diseñadas para enfrentar al programa a condiciones anormales. La prueba ejecuta un sistema de manera que demande recursos en cantidad, frecuencia o volúmenes anormales. (Pressman)

- ✓ **Pruebas de pico**

La prueba de picos, como el nombre sugiere, trata de observar el comportamiento del sistema variando el número de usuarios, tanto cuando bajan, como cuando tiene cambios drásticos en su carga.

Pruebas de requerimientos

Los requerimientos de software deben tener una explicación clara, precisa y completa del problema que facilite el análisis de errores y la generación de casos de prueba. Un asunto de gran importancia es asegurar la corrección, coherencia y exactitud de los requisitos. Durante el proceso de pruebas de requerimientos, una persona, designada por el Equipo de Aseguramiento de Calidad, revisará el documento de especificación de requerimientos, con la lista de chequeo general del documento y la lista de chequeo de requerimientos.

La corrección del contenido del documento será responsabilidad del analista y el usuario líder, quienes son los encargados de aprobar los requerimientos definidos en el documento.

En el proceso de verificación de requerimientos se determinan si los objetivos son claros, verificables y necesarios y el resultado de esta investigación se recoge en una lista de chequeo de objetivos, mediante un proceso iterativo se define la funcionalidad esperada del software y esta información debe ser recogida en el documento de requisitos del sistema, se debe verificar además el documento de requerimientos usando la lista de chequeo general del documento de especificación de requerimientos, lista de chequeo del documento general del documento de prueba y análisis de los requerimientos. El proceso de

verificación de los requerimientos comienza con el análisis de esos requerimientos y una inspección en la cual se busca evaluar la consistencia, completitud y factibilidad de los requerimientos, tanto individualmente como juntos. Adicionalmente los requerimientos deben ser revisados y validados por los distintos actores involucrados con el sistema.

Para evitar sorpresas de variada índole a la hora de entregar el software, es conveniente especificar claramente que se va a hacer para determinar que el sistema satisface sus requerimientos. (Teruel, 2001)

Prueba de recuperación

Esta prueba es la encargada de forzar al software a la ocurrencia de varios errores, verificando que la recuperación se haga satisfactoriamente, la cual puede ser automática o por la acción humana. En caso de ser automática el análisis recae sobre la corrección de la inicialización sobre los mecanismos de recuperación del estado del sistema, de los datos y del proceso de re arranque. En el otro caso se analizan los tiempos medios de reparación para determinar si están dentro de los límites establecidos.

1.3.8.4 Pruebas a nivel de Aceptación.

Las pruebas de aceptación, son las que se hacen con los clientes y define su aceptación del sistema. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable de caras al cliente; sino una vez pasadas todas las pruebas de integración por parte del desarrollador. La experiencia ha mostrado que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo.

Prueba Alfa

Esta prueba es llevada a cabo por un cliente en el lugar de desarrollo. El software es usado de forma natural con el desarrollador como observador del usuario. Las pruebas alfas se llevan a cabo en un entorno controlado. Para que estas pruebas sean válidas primero debe crearse un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente, una vez logrado esto se procede a realizar las pruebas y a documentar los resultados.

Prueba beta

Esta prueba es llevada a cabo por los usuarios finales del software, a diferencia de la prueba alfa el desarrollador no está presente. Esto convierte a la prueba beta en una aplicación en vivo del software, en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador.

1.3.9 Métodos de pruebas.

Existen diversos métodos para realizar las pruebas de software, entre los más importantes se encuentran la prueba de Caja Blanca y prueba de Caja Negra.

La prueba de Caja Blanca es la mejor de su tipo para verificar que se recorran todos los caminos y detectar un mayor número de errores. La Caja Negra brinda la posibilidad de cubrir la mayor parte de las combinaciones de entradas y lograr con ello un juego de pruebas más eficaz.

Las pruebas mencionadas permiten probar cada una de las condiciones existentes en el programa, identificar claramente las entradas, salidas y estudiar las relaciones que existen entre ellas, permitiendo así maximizar la calidad de las pruebas y en dependencia del resultado se constará con un sistema más estable y confiable.

1.3.9.1 Prueba de Caja Blanca.

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Se centra en la estructura interna del programa para elegir los casos de prueba.

Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

- ✓ Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- ✓ Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

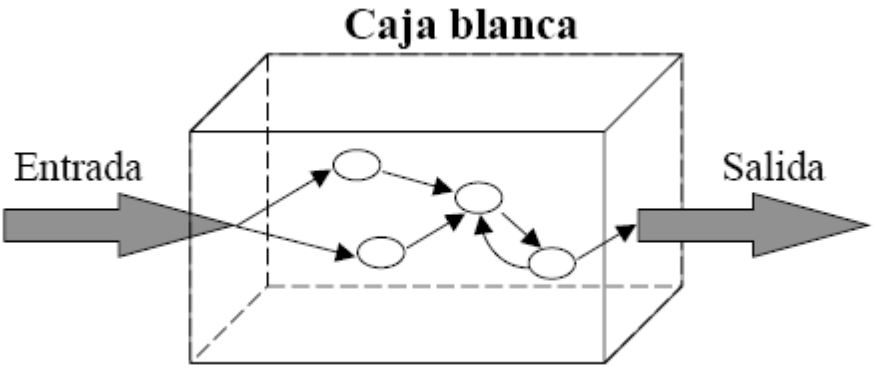


Fig. 1 Representación Prueba de Caja Blanca.

1.3.9.1.1 Técnicas de pruebas de Caja Blanca.

Prueba de camino básico

La prueba del camino básico es una técnica de prueba de la Caja Blanca. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control.

Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Pruebas de ciclo

Esta técnica se enfoca solamente a verificar los ciclos que se encuentren en el programa, identifica cuatro tipos de ellos: simples, concatenados, anidados y no estructurados. Para cada uno de ellos actúa de manera distinta y sugiere el rediseño de los ciclos no estructurados.

Pruebas de condiciones

Son métodos de diseño de casos de prueba que verifican cada condición o decisión en un programa. Los tipos de errores pueden ser: un operador booleano, una variable booleana, un operador relacional.

Prueba de flujo de datos

Es un método de diseño de casos de prueba que selecciona caminos en un programa de acuerdo a la ubicación de las definiciones y uso de las variables en él. Dado que las instrucciones de un programa están relacionadas entre sí de acuerdo a las definiciones, el enfoque de flujo de datos es muy efectivo, aunque la selección de caminos es más compleja.

1.3.9.2 Prueba de Caja Negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose solamente en los requisitos funcionales del sistema. Muchos autores consideran que estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Fig. 2 Representación Prueba de Caja Negra.

1.3.9.2.1 Técnicas de pruebas de Caja Negra.

Para desarrollar la prueba de Caja Negra existen varias técnicas, entre ellas se encuentran:

a. **Técnica de la Partición de Equivalencia:** Es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una serie de errores que, de otro modo, requerirían la ejecución de muchos casos de prueba antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de casos de prueba que hay que desarrollar.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. (Pressman)

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

- Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

- Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.

- Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.
Aplicando estas directrices se ejecutan casos de pruebas para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez.
Para definir las clases de equivalencia hace falta tener en cuenta un conjunto de reglas:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y 2 inválidas.

- Si una condición de entrada especifica la cantidad de valores, identificar una clase de equivalencia válida y dos inválidas.

- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, identificar una clase válida para cada uno de ellos y una clase inválida.

- Si una condición de entrada especifica una situación de tipo “debe ser”, identificar una clase válida y una inválida.

- Si existe una razón para creer que el programa no trata de forma idéntica ciertos elementos pertenecientes a una clase, dividirla en clases de equivalencia menores.

Luego de tener las clases válidas e inválidas definidas, se procede a definir los casos de pruebas, pero para ello antes se debe haber asignado un identificador único a cada clase de equivalencia. Luego entonces se pueden definir los casos teniendo en cuenta lo siguiente:

- ❖ Escribir un nuevo caso de cubra tantas clases de equivalencia válidas no cubiertas como sea posible hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba.

- ❖ Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida hasta que todas las clases de equivalencias inválidas hayan sido cubiertas por casos de pruebas.

Con la aplicación de esa técnica se obtiene un conjunto de pruebas que reduce el número de casos de pruebas y nos dicen algo sobre la presencia o ausencia de errores.

A menudo se plantea que las pruebas a los software nunca terminan, simplemente se transfiere del desarrollador al cliente. Cada vez que el cliente usa el programa está llevando a cabo una prueba. Aplicando el diseño de casos de pruebas al software en cuestión se puede conseguir una prueba más completa y descubrir y corregir el mayor número de errores antes de que comiencen las “pruebas del cliente”.

2. **Técnica del Análisis de Valores Límites:** Esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

3. **Técnica de Grafos de Causa-Efecto:** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. Consiste en crear un grafo causa/efecto a partir de las especificaciones, y seleccionar suficientes casos de pruebas para asegurar la cobertura del grafo, y mediante él, construir una tabla de decisión para reflejar las dependencias de los resultados.

1.4 Metodología de desarrollo

Con los avances que ha alcanzado la informática a nivel mundial, fundamentalmente en la evolución tecnológica ha hecho necesario desarrollar metodologías para asegurar la calidad de los productos de software y obtener un mejoramiento continuo de todos los procesos relacionados con el desarrollo de software. No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Sin embargo cada proyecto debe tener una metodología definida para su desarrollo.

1.4.1 RUP Rational Unified Process (Proceso Unificado de Rational)

RUP es un proceso unificado de desarrollo de software que define claramente quién, cómo, cuándo y qué debe hacerse; y como su enfoque está basado en modelos, utiliza un lenguaje bien definido para tal fin, el UML.

El proceso de software propuesto por RUP tiene tres características esenciales: está dirigido por casos de uso, centrado en la arquitectura, y es iterativo e incremental permitiendo establecer trazabilidad entre los

artefactos que son generados en las diferentes actividades del proceso de desarrollo, además se presta especial atención al establecimiento temprano de una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores, y el trabajo se divide en partes más pequeñas o mini proyectos.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los primeros 6 son conocidos como flujos de ingeniería y los 3 últimos de apoyo. Para esta investigación el flujo principal es el flujo de pruebas.

Consta de 4 **fases**: Inicio, Elaboración, Construcción y Transición en cada una de estas fases está presente el flujo de pruebas:

- **Inicio:** Se desarrolla el prototipo exploratorio de demostración. No se requiere la elaboración de las pruebas.
- **Elaboración:** Se prueban los componentes ejecutables que se han implementado y que deben corresponderse con la arquitectura básica de la aplicación.
- **Construcción:** Se desarrollan los casos de prueba y procedimientos de prueba ya planteados.
- **Transición:** En esta fase el producto se encuentra en su entorno de operación por lo que es probado por usuarios reales.

Flujos de trabajo

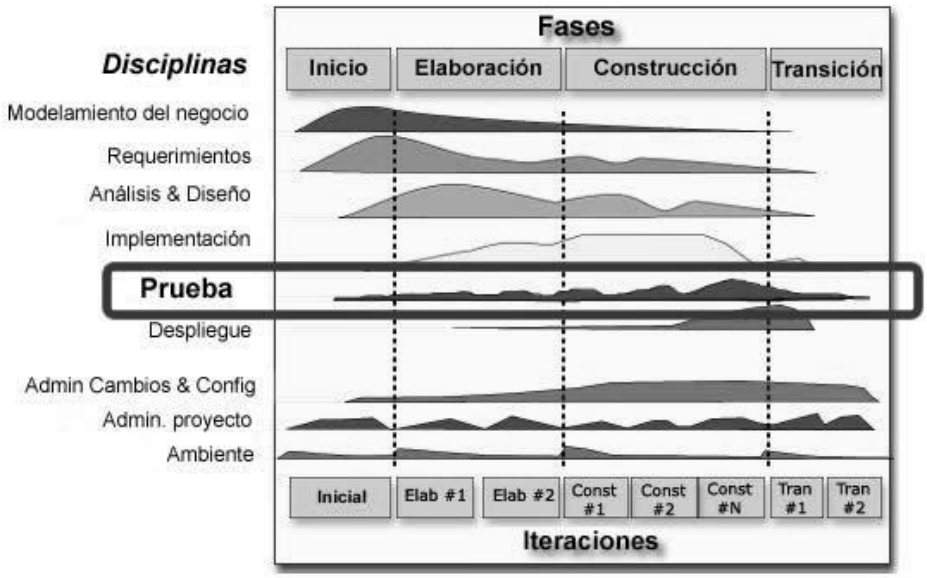


Fig. 3 Flujo de trabajo de pruebas según RUP

El **desarrollador del software** es responsable de probar los módulos o subsistemas asegurándose que cada uno lleva a cabo la funcionalidad para el que fue diseñado.

Una vez que la arquitectura del software esté completa, entra en juego un grupo independiente de prueba que elimina el conflicto de interés que para el desarrollador presenta que él tiene que probar su propio producto.

Aunque las pruebas pueden ser realizadas por usuarios finales y otros trabajadores del equipo, RUP define que los **roles** que intervienen en el flujo de prueba son los siguientes:

- **Diseñador de pruebas:** Planifica, diseña y evalúa los resultados de la prueba. Es el responsable de definir el método de prueba y asegurar su implementación exitosa. El rol incluye identificar técnicas apropiadas, herramientas e instrucciones para implementar las pruebas necesarias y encauzar los recursos correspondientes para las pruebas.

- **Ingeniero de componentes:** Responsable de la elaboración de los componentes de prueba para los procedimientos que pueden ser automatizados.

- **Ingeniero de pruebas de integración:** Realizan las pruebas de integración.

- **Ingeniero de pruebas de sistema:** Realiza las pruebas de sistema.

- **Administrador de Prueba:** Es el responsable del éxito de la prueba. Dirige el comportamiento de las pruebas y define los tipos de pruebas necesarias, además de manejar los resultados de las mismas.

- **Analista de Prueba:** Es el responsable de identificar y definir las pruebas requeridas, monitorear el progreso de la prueba y el resultado en cada ciclo de prueba, evaluando la calidad total experimentada como un resultado de las actividades de prueba. Este rol lleva la responsabilidad para representar apropiadamente las necesidades de los stakeholder que no tienen representación regular y directa en el proyecto.

- **Probador:** Es el responsable de la ejecución de las pruebas necesarias y el registro del resultado de las mismas.

1.4.1.1 Pruebas según RUP

Las pruebas según RUP se centran en 3 objetivos fundamentales, estos son:

1. Planificar las pruebas de integración; tanto de integración para cada construcción, como del sistema, dentro de cada iteración.

2. Diseñar e implementar las pruebas con la elaboración de casos de prueba; así como evaluar la utilización de algún software que permita automatizar las pruebas.

3. Realizar las pruebas según lo planificado anteriormente y arreglar los defectos detectados como resultado de la ejecución de las pruebas.

Artefactos del flujo de pruebas

Un artefacto es todo producto o subproducto resultante del proceso de desarrollo del software, es un trozo de información que es producido, modificado o usado. No siempre en todo proyecto se crean los mismos artefactos, esto dependerá de las características del proyecto y los requisitos del cliente. Para la fase de pruebas se tienen los siguientes artefactos:

Modelo de Pruebas: Describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema. Puede describir también cómo se han probado aspectos específicos del sistema, por ejemplo, si la interfaz de usuario del sistema cumple con su objetivo y describe el cumplimiento de los requisitos funcionales y no funcionales del sistema. Es una colección de casos de pruebas, procedimientos de prueba y componentes de prueba.

Caso de Prueba: Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultado con el cual será probado y las condiciones bajo las que ha de probarse. Un caso puede derivarse de las descripciones de los casos de uso del diseño.

Procedimiento de Prueba: Un procedimiento de prueba especifica cómo realizar uno o varios casos de pruebas o partes de estos. Un procedimiento de prueba puede ser una instrucción para un individuo sobre cómo realizar un caso de prueba manualmente, o una especificación de cómo interactuar manualmente con una herramienta de automatización de pruebas, para crear componentes ejecutables de prueba.

Componente de Prueba: Un componente de prueba automatiza uno o varios procedimientos de prueba o partes de ellos. Estos pueden ser desarrollados utilizando un lenguaje de guiones o un lenguaje de programación, o pueden ser gravados con una herramienta de automatización de pruebas. Los componentes de pruebas se utilizan también para probar los componentes en el modelo de implementación, proporcionando entradas de pruebas, controlando y motorizando la ejecución de los

componentes a probar. Los componentes de pruebas pueden ser implementados usando tecnología de objetos.

Plan de Prueba: El plan de prueba describe las estrategias, recursos y planificación de la prueba, el nivel de cobertura de prueba y de código necesario, además del porcentaje de prueba que deberían ejecutarse con un resultado específico. El propósito del plan de pruebas es dejar de forma explícita el alcance, el enfoque, los recursos requeridos, el calendario y los responsables del proceso de pruebas. Cada prueba debe dejar claro qué tipo de propiedades se quieren probar así como tener una visión clara de cómo medir los resultados. La construcción de un buen plan de pruebas es la piedra angular y en consecuencia el principal factor crítico de éxito para la puesta en práctica de un proceso de pruebas que permita entregar un software de mejor nivel.

Defecto: Un defecto es un síntoma de un fallo del software o un problema descubierto en una revisión, el cual puede ser utilizado para localizar cualquier cosa que los desarrolladores necesiten registrar cómo síntoma de un problema en el sistema.

Evaluación de Prueba: Es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, de código y el estado de los defectos. Los diseñadores evalúan los resultados de la prueba comparando los resultados obtenidos con los objetivos trazados en el plan de prueba. Se realizan métricas que les permiten determinar el nivel de calidad del software y qué cantidad de pruebas es necesario realizar.

Estrategias de Pruebas: Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software. Es el proceso de analizar cómo plantear las pruebas en el Ciclo de Vida.

La metodología RUP a diferencia de otras metodologías, posee una gran cantidad de artefactos que brinda la posibilidad de que se tenga un mayor entendimiento del proceso de desarrollo a realizar y una mejor forma de organizar el trabajo. RUP incluye el artefacto procedimiento de pruebas que es el objetivo principal de este trabajo. El flujo de pruebas que propone RUP recorre las 4 etapas de la metodología lo que posibilita la aplicación de este procedimiento en cualquiera de ellas garantizando una vía factible para corregir los errores partiendo de que los productos que son desarrollados con RUP garantizan desde sus comienzos una arquitectura lo suficientemente fuerte como para no verse afectada por la aplicación de cambios que puedan surgir en el camino. La utilización de la metodología RUP es extremadamente confiable dadas las características de este sistema además de que el Sistema Único de Aduanas (SUA) está desarrollado bajo esta metodología.

1.5 Realización de Pruebas en el mundo.

Muchas organizaciones desconocen los aspectos fundamentales del proceso de pruebas, este desconocimiento lleva con frecuencia a subestimar la complejidad de las actividades involucradas en este proceso, lo cual cobra un alto precio en los proyectos. Debido a esto, muchas empresas productoras de software y algunas empresas consumidoras contratan servicios especializados de pruebas de software independiente orientados a mejorar la calidad de sus servicios. Algunas de esas empresas son:

SQS (Software Quality Systems) es la compañía líder en servicios de Consultoría de Calidad de Software y Pruebas. Ofrece servicios de Validación y Verificación independiente, es experta en diseño e implantación de entornos de pruebas, en revisiones formales de documentación y código, y otros servicios que ofrecen a sus clientes un mayor control sobre el proceso, una identificación temprana de errores y problemas y una reducción drástica de los costes para subsanar estos errores.

Otro ejemplo es el CES (Centro de Ensayos de Software), que desarrolla su actividad desde el año 2004, fundamentalmente ofrece servicios de pruebas independientes de productos, consultoría y capacitación en pruebas. Este Centro ha participado en diferentes eventos en el área de Ingeniería de Software, aportando sus conocimientos en la disciplina de pruebas y en la actualidad ha concretado proyectos con diferentes empresas, posicionándose como institución de referencia. Brinda servicios especializados en pruebas de software para evaluar la calidad de los productos. Realizan la evaluación de productos en diferentes plataformas para ello utilizan las pruebas de Carga que estudian el desempeño de productos bajo distintos niveles de carga esperables para lo cual fue diseñado, las pruebas de Estrés que se centran en el desempeño de productos ante situaciones de carga para los cuáles no fue diseñado, la prueba de Escalabilidad donde se ve el desempeño del producto respecto a cambio en los recursos disponibles y la prueba de Configuración se enfoca en el producto ante cambios de la plataforma base, tanto de hardware como de software.

1.6 Herramientas utilizadas para la realización de pruebas

Existen una serie de herramientas muy útiles para permitir a las empresas asegurar alta fiabilidad y rendimiento de manera sencilla, con productos y servicios que ayudan a los desarrolladores a emplear las mejores prácticas en el proceso de producir aplicaciones de alta calidad. Con estas herramientas se proporciona un análisis detallado de la calidad y rendimiento de las aplicaciones propiciando a los desarrolladores un experto automatizado capaz de advertirles, localizar y corregir sus problemas.

1.6.1 ACT Application Center Test (Centro de aplicaciones de prueba)

Application Center Test (ACT) es una herramienta de Microsoft incluida en Visual Studio .NET Enterprise Developer y Enterprise Architect, es un software propietario y está diseñada especialmente para realizar pruebas de carga y estrés, que permite obtener información para detectar problemas de rendimiento y escalabilidad. A pesar de estar especializada en este tipo de pruebas, ACT también permite realizar pruebas funcionales gracias a los test (pruebas) dinámicos, lo que puede resultar útil para aplicaciones de complejidad media-baja. El funcionamiento de ACT reside en simular un gran número de usuarios abriendo múltiples conexiones al servidor y enviando peticiones HTTP. Algunas de las características más interesantes de ACT son:

- Permite probar cualquier página Web que responda a una petición HTTP (.asp, .aspx, .php.).Graba scripts de pruebas desde una sesión de Internet Explorer.
- Soporta SSL.
- Acumula los resultados de los test para su análisis.
- Gestiona cookies.
- Soporta diferentes tipos de esquemas de autentificación.
- Se puede considerar que el predecesor directo de ACT es Web Application Stress Tool (WAST).

1.6.2 Herramienta Web Application Stress

La herramienta Web Application Stress activa el entorno de prueba mediante la simulación realista de varios exploradores que solicitan páginas desde una aplicación web, es un software propietario y permite grabar una secuencia de comandos mediante el acceso, desde el explorador, a las páginas que se desea incluir en la prueba. La secuencia se puede guardar y ejecutar desde cualquier equipo cliente Windows NT o Windows 2000 que disponga de la aplicación. No es necesario que se disponga del mismo número de equipos clientes que los que tendrán la aplicación de producción, ya que la herramienta web Application Stress es capaz de simular varios clientes desde cada estación de trabajo.

Cuando se ejecute la prueba de carga, hay que tener cuidado de no aumentar el nivel de carga en los equipos clientes hasta el punto de que los equipos de prueba pasen más tiempo realizando cambios de contexto entre subprocesos que trabajando. Para asegurar que los subprocesos se distribuyen entre los

equipos clientes, se deben utilizar varios equipos clientes cuando se ejecute una prueba de una aplicación basada en Web, de ese modo disminuirán los límites de recursos de un equipo cliente.

1.6.3 OpenSTA

OpenSTA es una herramienta open-source, basada en arquitectura CORBA que permite el testeo de estrés de aplicaciones web, de una forma similar a ACT (herramienta para test de stress de aplicaciones ASP.NET). Más potente que ACT en cuanto a que permite un mayor control del número y distribución de usuarios virtuales. Utiliza una variante de Basic para la generación de los scripts. Permite pruebas de HTTP y HTTPS. Es desarrollado bajo los estándares de software libre.
Para realizar las pruebas, el proceso que hay que seguir se puede dividir en dos pasos:

- **Creación de pruebas sencillas a partir de la grabación de acciones de usuario.**

  Se configura el programa para crear un proxy que se encargará de grabar las acciones del usuario, es decir, las peticiones y las respuestas del servidor. Luego se configura el navegador para conectar a ese proxy, y se pone a grabar las acciones en el OpenSTA. Se realiza la prueba manualmente y cuando se termina, se para la grabación, y se guarda el script generado de la prueba.

- **Realización de baterías de pruebas.**

  Una vez se tengan creadas unas cuantas pruebas a partir de las acciones del usuario (altas, bajas, modificaciones…), se utilizan para realizar baterías de pruebas intensas con uno o varios usuarios. El programa ejecuta las pruebas simples y cuando termina, proporciona gran cantidad de datos acerca de las pruebas.

1.6.4 JMeter

JMeter es un proyecto de Apache Jakarta que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web. Es desarrollado bajo los estándares de software libre.
JMeter puede ser usado como una herramienta de pruebas unitarias para conexiones de bases de datos con JDBC, FTP, LDAP, Servicios web, JMS, HTTP y conexiones TCP genéricas. JMeter puede también ser configurado como un monitor, aunque es comúnmente considerado una solución ad-hoc respecto de soluciones avanzadas de monitoreo.

Mientras que JMeter es clasificado como una herramienta de "generación de carga", no es una descripción completa de la herramienta. JMeter soporta aserciones para asegurarse que los datos recibidos son correctos, por cookies de hilos, configuración de variables y una variedad de reportes.
Una característica importante del JMeter es que es extensible y ofrece la posibilidad de que el propio usuario desarrolle en Java un controlador (“Controller”) a medida, cumpliendo una interfaz Java y depositando el .jar correspondiente al desarrollo en el directorio “lib” de JMeter. Además JMeter permite realizar pruebas distribuidas en distintos ordenadores que actúan como clientes. Puede simular una gran carga en el servidor, HTTP y FTP testing y bases de datos mediante JDBC, multitarea y con grandes facilidades para extender su funcionalidad mediante plugings.

Los elementos jerárquicos del JMeter son:

- Listeners (elementos de escucha).
- Config Elements (elementos de configuración).
- Post-processors (Post- procesadores).
- Pre-processors (Pre- procesadores).
- Assertions (afirmaciones).
- Timers (Cronómetros).

Los Elementos ordenados serían:

- Controllers (controladores).
- Samplers (agentes de pruebas).

Existen varias formas de generar la carga de trabajo para la realización de las pruebas de rendimiento. La más sencilla es de forma manual, sin embargo es la menos eficiente. Otra vía es desarrollando un software que permita la realización de estas pruebas, que mirándolo desde este punto de vista representa un ahorro considerable de dinero, sin embargo estas herramientas son enormemente complejas en su desarrollo, lo que representaría un buen gasto de dinero y tiempo, a no ser que se limitaran los requisitos.
Es por lo antes expuesto que se toma la decisión de usar algún software existente, ya sea libre o propietario, preferentemente libre, que sirva para la realización de las pruebas, que permita configurar un

entorno de pruebas propio. A la hora de escoger la herramienta uno de los aspectos más importantes a tener en cuenta es su flexibilidad respecto a la configuración y la confianza en los resultados que este exponga, pues de esta forma se garantiza un rápido aprendizaje en su uso y el éxito en la aplicación de las pruebas.

El JMeter es la herramienta seleccionada para la realización de pruebas pues es una herramienta libre muy confiable que muestra los resultados de las pruebas en una amplia variedad de informes y gráficas, con una gran cantidad de variables diferentes que permiten interpretar los resultados desde diferentes puntos de vista. Además facilita la rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo.

Entre las ventajas que nos favorecen de la herramienta JMeter está la estructura de árbol que presenta, que le da una increíble potencia. Permitiendo que sea la imaginación de quien realiza las pruebas la que le ponga límites a las mismas. Hay que tener en cuenta además que esta herramienta brinda gran cantidad de variantes para la obtención de resultados, como pueden ser el reporte agregado que muestra una tabla con todos los valores que va dando como respuesta la aplicación por peticiones, el grafico agregado que además de mostrar la tabla anterior da la posibilidad de graficar, aunque este es más usado cuando no existen un número grande de páginas por probar, pues de lo contrario la gráfica se distorsiona, también está el gráfico de resultados que muestra una gráfica en tiempo real de lo que ocurre con los resultados de las pruebas y otra de las opciones es ver los resultados en árbol que reúne la información resultante de la prueba por aspectos que puedan resultar de interés, todas estas opciones permiten un análisis mucho más profundo de los resultados.

1.7 Conclusiones

En este capítulo se definieron los diferentes métodos, niveles y tipos de pruebas, herramientas, así como metodologías de desarrollo a usar para asegurar la realización de un buen procedimiento de pruebas. En la Universidad existe una gran comunidad de aprendizaje de la metodología RUP por su demostrada validez a la hora de diseñar una aplicación, lo que amplía la posibilidad de un óptimo uso de la misma, aprovechando sus características en pos de un correcto diseño del procedimiento de pruebas. Se estudiaron las herramientas de pruebas para la selección de la que mejor cumpla con nuestros objetivos para su aplicación del procedimiento de pruebas de rendimiento a realizar. Igualmente se enfocó el estudio en las pruebas a nivel de sistema pues este nivel incluye las pruebas de rendimiento, que son el objetivo de este trabajo.

Es importante mencionar que no existe un procedimiento establecido que se relacione con el desarrollado en este trabajo que sirva como guía o base. Todos los elementos establecidos dentro del procedimiento están realizados de acuerdo a las necesidades planteadas luego del análisis del proceso de pruebas que se desea realizar.

CAPÍTULO 2: DESCRIPCIÓN DEL PROCEDIMIENTO DE PRUEBAS DE RENDIMIENTO

2.1 Introducción

En el presente capítulo se describe el procedimiento de pruebas de rendimiento diseñado para ser aplicado al Sistema Único de Aduanas (SUA), que cuenta con varios aspectos que organizan y especifican paso a paso las actividades que serán realizadas para llevar a cabo exitosamente el procedimiento. Primeramente se define el alcance del procedimiento con el objetivo de mostrar la magnitud que tendrá el mismo abarcando cómo, y a qué será enfocado, hecho esto serán seleccionados los roles que intervienen en el procedimiento con sus respectivas responsabilidades. Luego se definirán los objetivos del procedimiento que especificarán las tareas a realizar. Una vez establecidos y explicados los elementos anteriores se plantean las 4 fases por las cuales transitará este procedimiento, estas fases son: Planificación, Aseguramiento, Ejecución y Análisis e Interpretación de los resultados. En cada una de ellas se realizan una serie de actividades que permiten darle cumplimiento a los objetivos establecidos inicialmente dentro del procedimiento.

2.2 Procedimiento para la realización de pruebas de rendimiento

2.2.1 Alcance

En el alcance se define lo que será evaluado en las pruebas, el nombre de los sistemas o módulos a probar junto al tipo de prueba de rendimiento que se le realizará.

2.2.2 Selección de Roles

Para la aplicación de este procedimiento de pruebas de rendimiento se realiza la elección de un equipo de trabajo seleccionado por el jefe de proyecto que será el encargado de revisar la realización de las pruebas, en compañía del jefe del módulo de calidad.
Durante el procedimiento de pruebas es necesario asumir 4 roles fundamentales por el equipo que desarrollará el trabajo, estos roles son:

- Administrador o Jefe de pruebas
- Diseñador de las pruebas
- Analista de Prueba
- Probador

Antes de la realización de las tareas por roles es importante saber las competencias básicas necesarias para seleccionar cada uno de estos roles. Estas son:

- Conocer cómo funciona el flujo de pruebas de la metodología RUP.
- Conocer a plenitud la aplicación a probar.
- Tener toda la documentación necesaria para aprender a usar la aplicación y dominarla antes de la planificación y realización de las pruebas.

Tabla de distribución de tareas por roles:

Rol	Responsabilidades
Administrador de Prueba	✓ Asegurar la planificación apropiada y dirección de los recursos de las pruebas. ✓ Evaluar el progreso y efectividad del proceso de pruebas. ✓ Evaluar resultados obtenidos.
Diseñador de Prueba	✓ Realizar el procedimiento de las pruebas, decidir los objetivos de prueba apropiados. ✓ Identificar, priorizar, seleccionar y describir los casos de pruebas y los procedimientos correspondientes. ✓ Identificar las técnicas apropiadas, herramientas y pautas para llevar a cabo las pruebas. ✓ Definir la configuración del entorno para realizar las pruebas.
Analista de Pruebas	✓ Verificar y llevar el control de la ejecución de las pruebas. ✓ Grabación de los escenarios para Carga y Estrés.
Probador	✓ Preparar y ejecutar las pruebas. ✓ Conformar documentación al culminar el proceso.

2.2.3 Objetivos

Las pruebas de rendimiento desarrolladas en este procedimiento tienen como objetivo medir las posibilidades reales del sistema y compararlas con las esperadas para llegar a conclusiones respecto a la calidad real de la aplicación, además de realizar la documentación de los errores encontrados para que sean corregidos posteriormente. Para esto se usan como base los objetivos de rendimiento establecidos:

- Verificar que la aplicación soportará la demanda de trabajo que debe soportar, respondiendo a las peticiones de los usuarios en un tiempo menor o igual que el especificado por el grupo de diseñadores.

- Detectar posibles errores límites e ineficiencias con una correcta proyección de las evaluaciones.

- Verificar si es necesario llevar la aplicación a tope, teniendo en cuenta los tiempos para el 50% de los resultados, pues si estos están por encima de lo planificado se puede decir que la aplicación no cumple con las expectativas.

Sin embargo para un mejor cumplimiento de las tareas del procedimiento se establecen objetivos mucho más específicos:
Estos objetivos pueden ser:

- ❖ Verificar que el sistema esté ajustado para soportar la máxima carga de trabajo posible usando la infraestructura actual.

- ❖ Determinar el tiempo medio de respuesta que obtendrá el usuario.

- ❖ Determinar el número máximo de usuarios concurrentes que pueden acceder a una página específica, o transacciones por segundo que la aplicación es capaz de soportar.

- ❖ Identificar las páginas que responden más lentas y las más rápidas.

2.2.4 Fases a desarrollar

Una vez establecidos el alcance, los roles que participan en el procedimiento y los objetivos del mismo, se desarrollarán las 4 fases que forman el procedimiento para pruebas de rendimiento, estas fases son:

1. Planificación

2. Aseguramiento

3. Ejecución

4. Análisis e Interpretación de los resultados

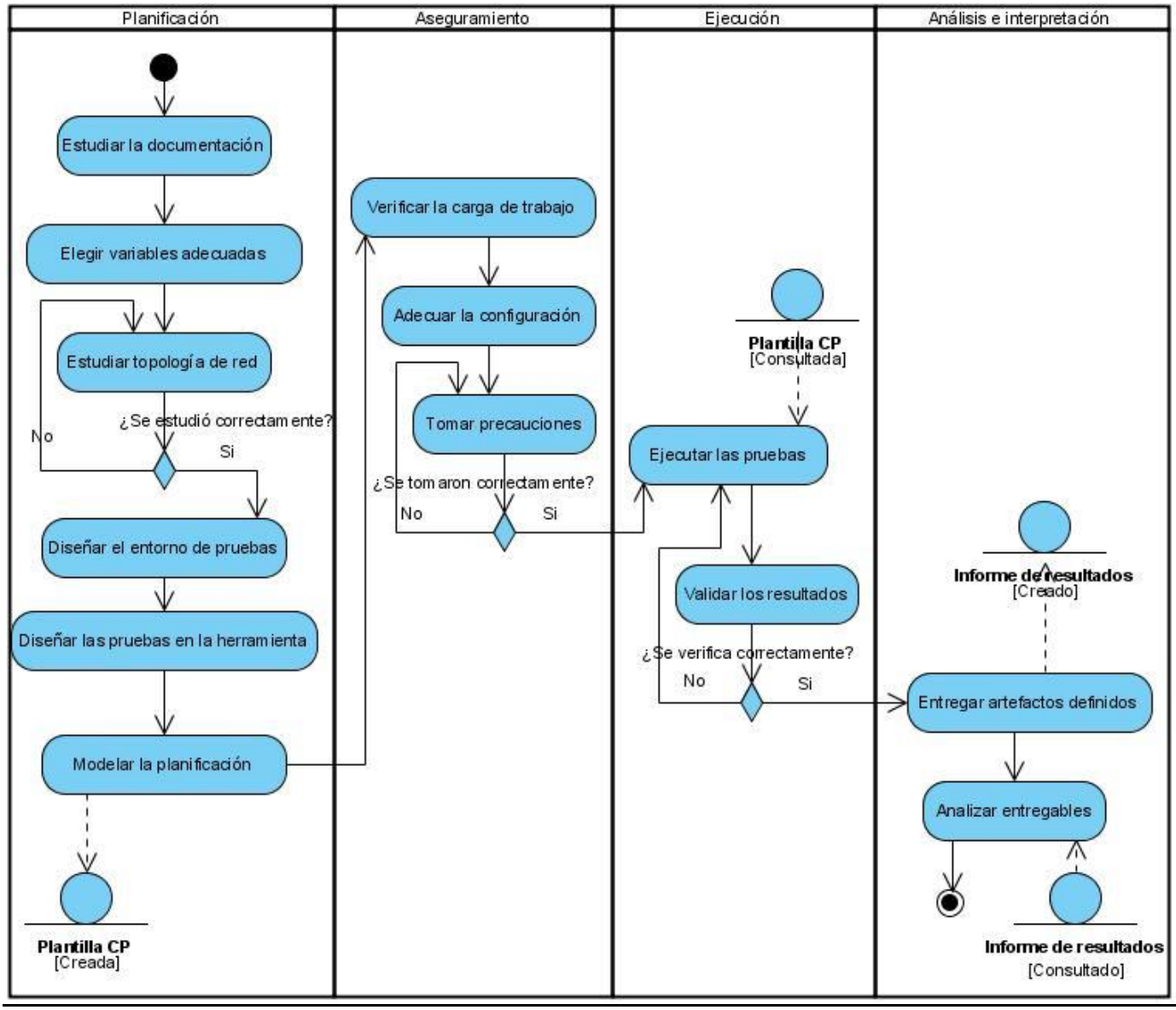


Fig. 3 Actividades por fase.

2.2.4.1 Planificación

En esta primera fase, para lograr una correcta organización de las tareas, se necesita una planificación que incluya la mayor cantidad de aspectos relacionados con la preparación que lleva la creación de un correcto procedimiento de pruebas de rendimiento. Esto permitirá el éxito del trabajo realizado. Es por esto que a continuación se mencionan los aspectos a tener en cuenta en la fase de planificación.

- Estudio de la documentación

- Elección de las variables adecuadas
- Estudio de la topología de red
- Diseño del entorno de pruebas
- Diseño de las pruebas en la herramienta
- Modelo de la planificación

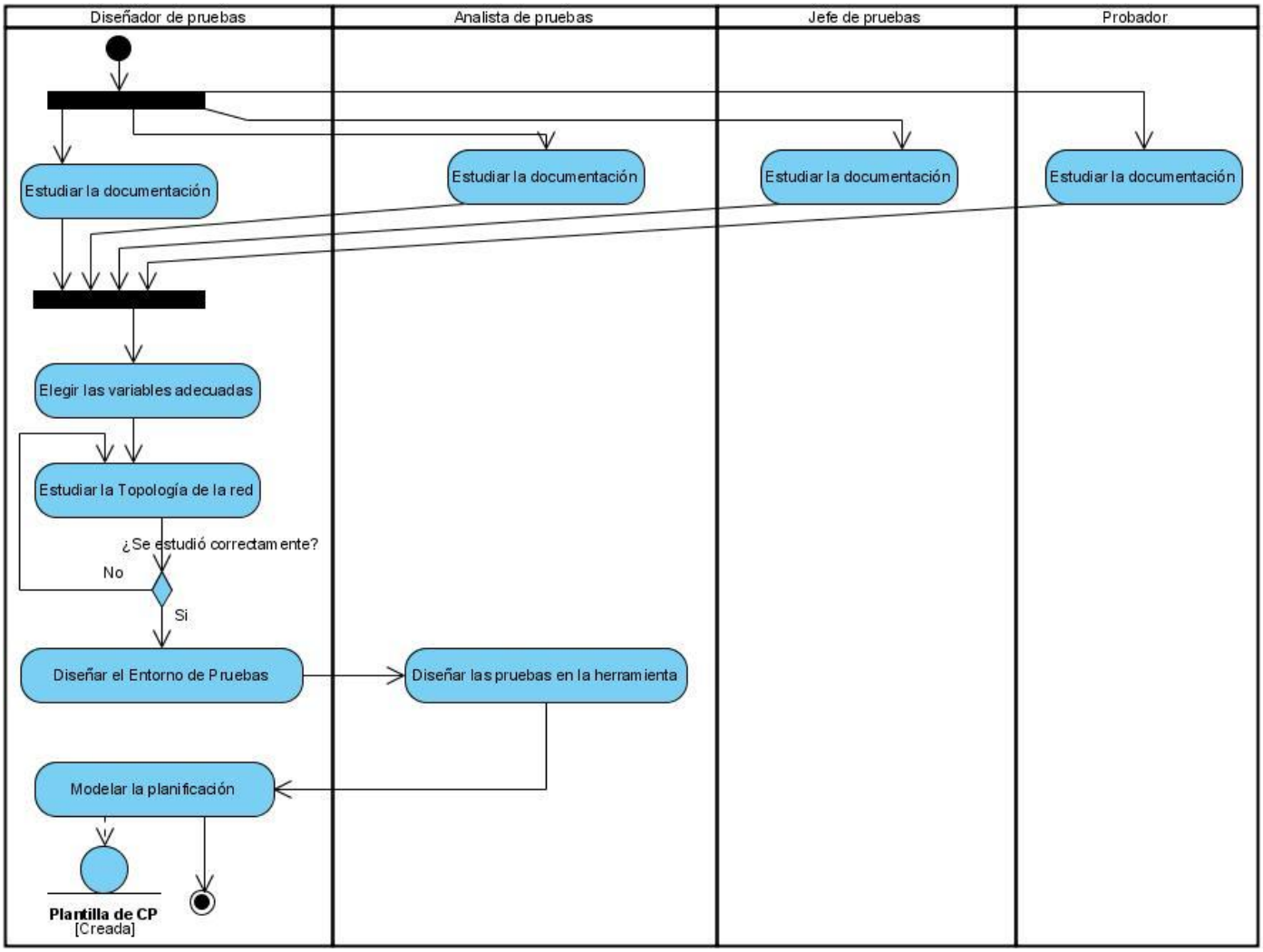


Fig. 4 Actividades por roles fase Planificación.

2.2.4.1.1 Estudio de la documentación

Dentro del proyecto existen una serie de documentos que es necesario conocer antes de realizar las pruebas. Todos los requisitos de rendimiento pertenecientes al sistema deben ser profundamente estudiados antes de ponerlos a prueba, de igual forma es importante tener conocimiento de las funcionalidades del sistema. Así se evitan errores que atrasarían las pruebas y podrían influir negativamente en sus resultados. De igual forma es importante realizar el estudio de la documentación sobre el software que se vaya a utilizar para la realización de las pruebas, así como todos los detalles de lo que son las pruebas de rendimiento. A continuación se mencionan algunos documentos que son imprescindibles conocer antes de la realización de las pruebas.

- Documento de especificación de requerimientos.
- Glosario de términos.
- Manual de usuario del software a utilizar para la realización de las pruebas.
- Requerimientos mínimos de hardware y software para realizar las pruebas.
- Especificación de los casos de uso (CU).

2.2.4.1.2 Elección de las variables adecuadas:

En las pruebas de rendimiento deben ser determinadas cuáles son las variables de entrada y salida con las que se va a trabajar, se tienen en cuenta los valores que tomarán las variables que necesita el software con el cual se realizarán las pruebas. De esta forma se garantiza una correcta interpretación de los resultados. Es por esto que tienen que ser definidas antes de la realización de las pruebas. Para este procedimiento las variables adecuadas son:

Variables de entrada:

- Número de usuarios.

Es importante mencionar que a veces es un poco difícil definir el número de usuarios que se usarán para la prueba, hay demasiados parámetros involucrados a la hora de darle carga a un sistema. Es por esto que la guía será el establecimiento de una carga que simule el uso normal del sistema. Variándola en dependencia del tipo de prueba de rendimiento, verificando siempre los valores extremos de usuarios que debe soportar el sistema.

Variables a generar:

A continuación se muestran las variables que serán usadas para la interpretación de los resultados:

- Niveles de rendimiento.
- Número de muestras.
- Tiempo de respuesta.
- Por ciento (%) de errores.

2.2.4.1.3 Estudio de la topología de red disponible

En este punto se debe realizar el estudio de la topología disponible en el lugar donde se vaya a aplicar el procedimiento de pruebas. Exponiéndose la mayor cantidad de detalles que sirvan de conocimiento para el equipo de pruebas. Estos aspectos pueden ser:

- Topología del área donde se simulará el entorno.
- Análisis y estudio de la organización de la Red.
- Características más significativas de los Switch.
- Características principales de la red del entorno a usar.
- Características de los servidores y PC clientes.

2.2.4.1.4 Diseño del entorno de pruebas

Para la aplicación del procedimiento de pruebas de rendimiento al software diseñado, se identifica el entorno de pruebas, para esto se estudian todos los aspectos desarrollados en el punto anterior y se resumen sus características más importantes de hardware y software que intervienen en el diseño y aplicación de las pruebas, los aspectos a resumir aquí son:

Especificaciones de Hardware.

En este aspecto se incluyen los elementos del hardware que se verán envueltos en el procedimiento de pruebas de rendimiento, estos pueden ser:

- Microprocesador.
- Memoria RAM.
- Ancho de Banda y tipo de conexión de red.

- Capacidad del disco duro para máquinas clientes y servidores.

Especificaciones de Software.

Se incluyen todas las aplicaciones que se necesitan para la realización del procedimiento, incluyendo el sistema operativo, ej.

- Sistema Operativo.
- Gestor de Base de Datos.
- Antivirus.
- Software de prueba.
- Elementos necesarios para el funcionamiento del software de prueba.

2.2.4.1.6 Diseño de las pruebas en la herramienta

Teniendo en cuenta la especificación de las variables independientes y las que se van a generar se procede al diseño de las pruebas en la herramienta especificada, este diseño consta de dos elementos fundamentales:

- Grabación de los escenarios de prueba.
- Confección de los Planes de Prueba.

Grabación de los escenarios de prueba.

La grabación de los escenarios de prueba se conforma por dos grabaciones independientes. Estas grabaciones recogen de manera general toda la navegación del sitio. Una de las grabaciones se encuentra dirigida a la navegación plena y la otra grabación se crea con el objetivo de recoger todas las peticiones que se generan al realizar acciones entre las PC clientes y el servidor, en este caso para cada sistema específico.

Confección de los Planes de Pruebas (Test Plan).

La confección de los planes de prueba para las pruebas de carga y estrés se realizan utilizando el método de pruebas complejas, debido a que entrar datos al sistema y enviarlos a la Base de Datos del mismo, es uno de los objetivos. Esta nueva acción se realiza para lograr una mayor similitud con los escenarios reales de navegación en el sistema.

Los planes de pruebas que se confeccionan para probar los Sistemas deben poseer la siguiente estructura jerárquica en los elementos:

1. Servidor Proxy HTTP
2. Grupo de Hilos
3. Árbol de Resultados
4. Aserción de Respuesta
5. Informe Agregado

2.2.4.1.7 Modelo de la planificación

Este punto recoge los detalles de la planificación realizada para la recogida y análisis de los resultados luego de la aplicación de las pruebas. Aquí se explica el uso de plantillas redactadas para las pruebas de rendimiento a realizar.

A la hora de confeccionar los casos de prueba para las pruebas de Caja Negra se hace referencia a las pruebas de rendimiento donde se puede apreciar los dos tipos de criterios que interesan en este caso, estos son las Pruebas de Estrés y las Pruebas de Carga.

Para este tipo de pruebas se traza una estrategia en la que convergen las páginas que reportan mayor cantidad de usuarios concurrentes en el sistema.

A continuación se exponen los detalles más relevantes de la plantilla desarrollada para casos de prueba de rendimiento usando la herramienta JMeter.

Fecha	Versión	Descripción	Autor
dd/mm/aaaa	V 1.0	Descripción de los casos de prueba.	

Esta plantilla cuenta con dos partes dedicadas a los dos tipos de prueba a realizar, que pueden ser de Carga o de Estrés. En ambos casos se da una breve explicación sobre los aspectos más importantes a la hora de la realización de la prueba dependiendo del tipo escogido.

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Nombre de la funcionalidad	[Detalles de lo que realiza esa funcionalidad]

En la tabla **Funcionalidades** que aparece en los dos tipos de prueba, se escribirán las funcionalidades del sistema que serán probadas (con las funcionalidades se hace referencia a las ventanas que serán visitadas dentro de la aplicación). En la parte de la tabla que lleva por encabezado **Comentario**, se pondrán detalles de lo que realiza la funcionalidad antes mencionada.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	
Tiempo entre conexión y conexión: [Período de subida (en segundos)]	[Tiempo que ocurrirá entre la conexión del los usuarios de la primera iteración y todas las demás]
Número de iteraciones: [Contador del bucle]	[Cantidad de veces que se repetirá la simulación de conexión de usuarios]

En la tabla se aprecian tres variables las cuales son pedidas por el software de pruebas JMeter y definidas por el usuario que realizará las pruebas:

Número de usuarios concurrentes (Número de Hilos): Variable que establece la cantidad de usuarios conectados simultáneamente que simulará el programa para el tipo de prueba establecida. Es importante saber qué **Número de Hilos** es el nombre que trae definido el software para esta variable.

Tiempo entre conexión y conexión (Período de subida (en segundos)): Variable que representa el tiempo que ocurrirá entre las conexiones de los usuarios definidos en Número de Hilos. Para el caso de la prueba de Carga esta variable tendrá por valor 0, pues las conexiones ocurrirán simultáneamente sin

tiempo entre un grupo de conexión y otro. En el caso de la prueba de Estrés el valor de esta variable será definido por el probador, para especificar el tiempo de espera.

Número de iteraciones (Contador del Bucle): Esta variable es la encargada de recoger la cantidad de veces que el usuario desea que se conecte el grupo de hilos definido. Para el caso de las pruebas de Carga, esta variable toma por defecto el valor 1, pues el grupo de usuarios (Hilos) se conectará una sola vez. En el caso de las pruebas de Estrés esta variable tomará el valor definido por el probador, pues éste es el encargado de definir la cantidad de veces que el grupo de hilos se conectará, llegando en cierta cantidad de tiempo al número de usuarios definido en la prueba.

Resultados Generales:

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg

La tabla de **Resultados Generales** es la encargada de recoger el resumen de los valores de las pruebas efectuadas. A través del análisis y de la comparación de los mismos se llegará a los resultados y se darán las conclusiones. Los resultados finales se obtendrán de la comparación y análisis de los valores cantidad de muestras (**Muestras**), el por ciento de error (**% Error**) y el valor del rendimiento (**Rendimiento**) medido por el número de peticiones por segundo, pues estas variables resumen el contenido que es necesario verificar dentro de la aplicación para comprobar su rendimiento.

(Para más información sobre las variables de la tabla dirigirse al Manual de Usuario)

Lista de links con errores

Lista de links con errores (En caso de existir):	Comentario

En la tabla lista de links con errores se insertan los vínculos que dieron error en el transcurso de la prueba, y en comentario se especifica qué tipo de error es.

(Para más información respecto a la plantilla de casos de prueba dirigirse al Anexo 15)

2.2.4.2 Aseguramiento

Luego de la realización de la planificación de las pruebas de rendimiento se pasa a la segunda fase dentro del procedimiento, donde se especifican las condiciones para construir dichas pruebas, los aspectos a analizar aquí son:

1. Verificación de la carga de trabajo.
2. Adecuación de la configuración.
3. Precauciones a tomar para proceder con las pruebas.

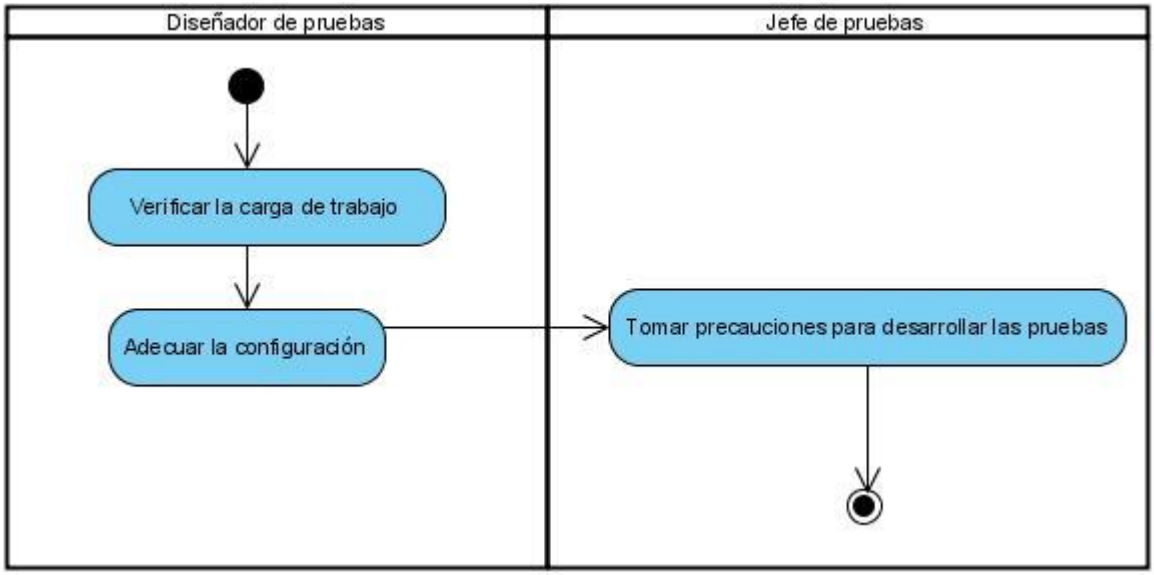


Fig. 5 Actividades por roles en la fase de Aseguramiento.

2.2.4.2.1 Verificación de la carga de trabajo

Una de las tareas más importantes antes de la realización de las pruebas de rendimiento es la verificación de la carga de trabajo, pues una decisión equivocada respecto a las posibilidades reales que el sistema debe cumplir, puede llevar a la toma de conclusiones erróneas respecto a los tiempos de respuesta del sistema. Una vía de solución para esto sería:

La determinación de la carga real de trabajo del sistema: Tiene como objetivo la determinación de los valores reales con los cuales el sistema debe trabajar sin problemas para garantizar un rendimiento conforme a las necesidades establecidas. De esta forma se comprobará el sistema en base a condiciones específicas y se obtendrán resultados completamente reales.

Los principales errores que deben evitarse a la hora de configurar la carga real de trabajo son:

- **Representar valores medios:** La aplicación debe ser comprobada con los valores reales, nunca debe asumirse que con el uso de valores medios el sistema responderá con valores promedios que se puedan usar para llegar a las conclusiones finales respecto a su rendimiento.

- **Configuración errónea de los datos de entrada:** En el momento en que se configuran las entradas al sistema respecto a los usuarios y su trabajo en la aplicación, debe tomarse como medida de partida que los usuarios posean un conocimiento previo de la aplicación y que los mismos realizan una navegación acorde a los tiempos normales de trabajo, evitando de esta forma realizar las pruebas teniendo en cuenta usuarios que no saben trabajar con el sistema y que lógicamente los tiempos de respuesta del sistema para estos casos van a ser mucho mayores.

- **No verificación de los tiempos de prueba:** Es importante comprobar la variación de tiempo de conexión usada en el diseño de las pruebas para los usuarios concurrentes, garantizando que sean las planificadas para garantizar una respuesta que se adapte a las necesidades establecidas. Un tiempo de entrada erróneo entre conexiones de grupos de usuarios concurrentes lleva a la toma de resultados equivocados y al fallo de las conclusiones del procedimiento de pruebas establecido.

Es necesario de forma específica, aun cuando se tienen en cuenta los errores planteados anteriormente, verificar a la hora de la configuración de la carga, algunas variables respecto a los usuarios, con las cuales se entiende mejor la importancia de una definición correcta de la carga de trabajo, estas variables son:

- **Tiempos de espera:** El tiempo que cada usuario se toma entre cada actuación en el sistema es significativo para la eficiencia del mismo. Evidentemente, no provoca la misma carga el usuario que conoce a la perfección el sistema y pulsa compulsivamente los enlaces antes de que las páginas se carguen por completo, que el usuario que mantiene la aplicación en segundo plano y actualiza casualmente.

- **Puntos de sincronización:** Puede surgir algún problema en la aplicación cuando un determinado número de usuarios coinciden en un determinado enlace: sin embargo aunque la actividad de un usuario es por regla general asíncrona, las principales herramientas de modelado de carga poseen la forma de sincronizar la actuación de los usuarios virtuales en un determinado punto de su ejecución.

- **Cadencias de conexión:** La forma en que los usuarios entran en el sistema también puede influir en la respuesta del mismo: no provocan los mismos efectos 100 usuarios que van entrando escalonadamente en el sistema a lo largo del día que los mismos 100 usuarios que se conectan casi simultáneamente en un determinado momento.

- **Plataformas:** Los distintos clientes (emuladores de terminal, navegadores) y los sistemas operativos desde los que se ejecutan (NT, Win32, Win16, Linux) pueden provocar diferencias en el tráfico generado hacia y desde el sistema.

- **Velocidad de conexión:** Existe una gran variedad de conexiones diferentes en función de su tecnología, velocidad de acceso y ancho de banda disponible. Esto repercute en el rendimiento.

Es importante recalcar que se hace imposible tener en cuenta todos los factores expuestos anteriormente, a la hora de realizar los casos de prueba, pues las pruebas de rendimiento pueden variar de sistema en sistema, pero si es importante recordar que el establecimiento de una pauta para las pruebas respecto a la mayor cantidad de aspectos permitirá un análisis y obtención de mejores resultados. Aunque la consideración puede ser también respecto a cuales de estos factores serán decisivos en el sistema a probar.

2.2.4.2.2 Adecuación de la configuración

Para la realización de las pruebas de estrés es necesario señalar si se está interesado en hacer pruebas duras sobre el sistema. Pues para esto lo primero es caracterizar un ciclo de trabajo muy corto, de apenas unos pocos segundos o menos, que realice un elevado número de peticiones. Luego se lanzan un número determinado de usuarios virtuales, y se verifica que los tiempos de respuesta se mantengan más o menos constantes al menos durante los primeros minutos. Después se van aumentando los usuarios virtuales hasta obtener la cantidad con los que la aplicación no aguanta.

2.2.4.2.3 Precauciones a tomar para desarrollar las pruebas

Para la realización exitosa de las pruebas de rendimiento, específicamente carga y estrés es necesario tomar algunas precauciones, estas son:

1. La aplicación a la cual se le van a desarrollar las pruebas debe ser una copia estable de la versión original del sistema que se esté desarrollando, de esta forma se evitará el resultado de pruebas equivocados con valores alterados por la entrada de usuarios externos a la prueba en el momento en que se realice.

2. Es tarea de los responsables de los sistemas que serán probados, comprobar que no exista ninguna fuga de información de los sistemas que se estarán probando, en caso de que esto ocurra es responsabilidad de los encargados garantizar la seguridad de los datos.

2.2.4.3 Ejecución

Ejecución es la tercera fase del procedimiento y como su nombre lo indica en él se procede a la realización de las pruebas con la herramienta especificada, los pasos a desarrollar dentro de la ejecución son:

1. Ejecución de las pruebas.
2. Validación de los resultados.

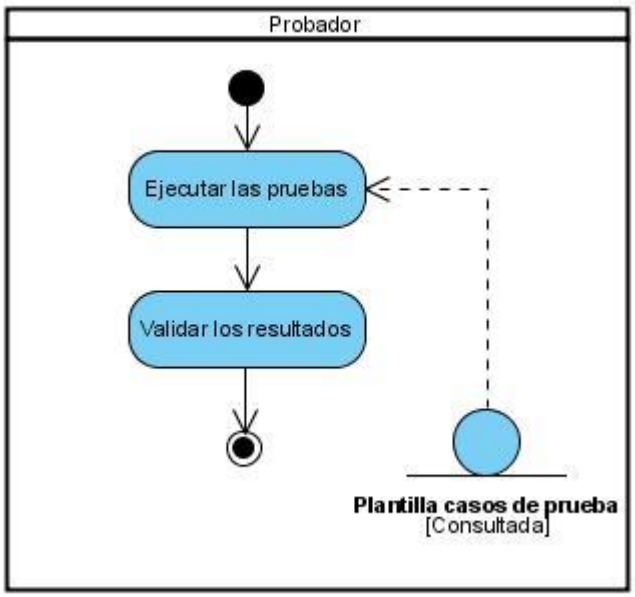


Fig. 6 Actividades por roles en la fase de ejecución.

2.2.4.3.1 Ejecución de las pruebas

Para la ejecución de las pruebas de rendimiento en aplicaciones web es importante realizar la planificación, pero se hace mucho más necesaria la especificación de definiciones de varios aspectos, estos pueden ser:

- Configuración del ambiente de pruebas.
- Instalación de las herramientas utilizadas.

- Instalación de las herramientas automáticas de pruebas.
- Selección del equipo de probadores.
- Capacitación del personal involucrado en las pruebas.

Personal calificado:

El personal encargado de poner en práctica el procedimiento de pruebas de rendimiento para aplicaciones web debe tener dominio completo de la aplicación y una forma de acceder a ella.

Es importante además que se tengan instaladas las herramientas necesarias para la realización de las pruebas, en este caso deben ser:

- JMeter.
- Máquina Virtual de Java.

Las pruebas comenzarán una vez que se tengan instaladas las herramientas, el ambiente este definido y listo y los casos de pruebas estén bien diseñados. Lo antes expuesto garantiza evitar demora y mala utilización de la aplicación por falta de conocimiento y pruebas mal hechas a causa de un mal diseño de los casos de prueba.

2.2.4.3.2 Validación de los resultados

Para realizar la validación de los resultados de las pruebas se verifica la influencia que ejercen algunos parámetros sobre el tiempo de respuesta de la aplicación, elementos que pueden provocar cierto retraso en la generación de la carga en las pruebas e interferencia en la obtención de los resultados, dichos parámetros a tener en cuenta son:

Utilización del CPU: Una utilización por debajo del 70% demuestra un rendimiento aceptable, sin embargo por encima de este valor demuestra que el sistema está sobrecargado. Para la verificación de la utilización del CPU, se accede a su valor a través del administrador de tareas de Windows.

Utilización de la memoria: Un uso de la memoria física de la máquina menor al 75 % es aceptable, sin embargo si el valor es igual a este o está por encima de él, demuestra la invalidación de los resultados de las pruebas.

Existen factores adicionales que se pueden usar para validar los resultados de las pruebas, estos pueden ser:

Consumo del Ancho de banda: EL ancho de banda del que se dispone para la aplicación de las pruebas es inferior al ancho de banda que tiene por defecto la red. Por esto es necesario verificar que el ancho de banda no sea superado durante el proceso, pues esto provocaría un retraso en la respuesta del sistema.

Porcentaje de error: En el resultado de las pruebas un elevado por ciento de errores puede representar varias cosas como por ejemplo, problemas con los drivers de comunicaciones de las PC Clientes, problemas con las funcionalidades del sistema que invalida la posibilidad de ser sometido a pruebas de rendimiento y errores en la introducción de datos en la herramienta. Por lo general no se aceptan porcentajes de error superior al 1% de las transacciones realizadas por el sistema.

Margen de error: Es importante saber que por muy específico que sea el proceso de pruebas, los resultados nunca serán 100 % reales en comparación a cuando el sistema esté en explotación, aun así los resultados ayudarán a mejorar al correcto funcionamiento del mismo.

2.2.4.4 Análisis e interpretación de los resultados

Análisis e interpretación de los resultados es la cuarta y última fase del procedimiento de pruebas, en él se procede al análisis de los resultados en función de las pruebas realizadas y sus valores.

Los criterios que se deben tener en cuenta para situaciones en las que se decida terminar los ciclos de pruebas están encaminados al rendimiento del sistema. Debido a que el sistema no se encuentra en las condiciones en las cuales será implementada la prueba se confecciona en la base de obtener una serie de resultados que permiten mejorar y detectar problemas que interfieran en su rendimiento.

Finalmente cuando todas las pruebas sean realizadas, y se pase a la última fase del procedimiento, Análisis e interpretación de los resultados, estos serán documentados y revisados con el objetivo de la obtención de los errores y el respectivo análisis de los mismos para trabajar en su completa eliminación y conseguir que el software funcione con el máximo de rendimiento.

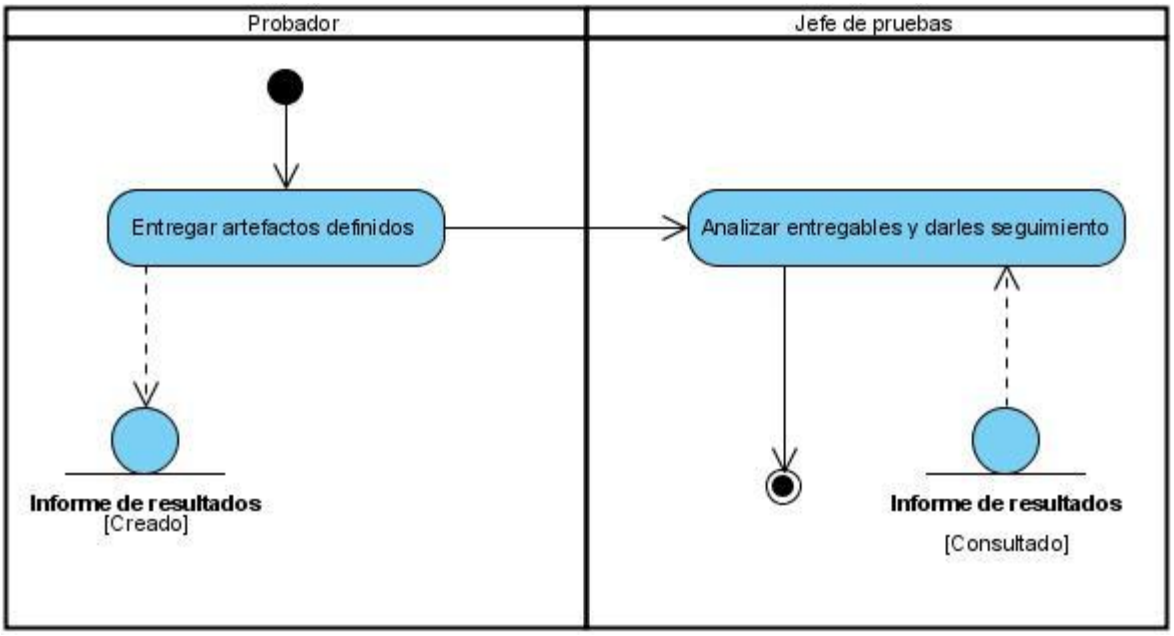


Fig. 7 Actividades por roles en la fase de Análisis e Interpretación de los resultados.

2.2.4.4.1 Entregables definidos

El proceso de pruebas automatizadas para el Sistema de Aduana genera 2 artefactos que serán requeridos a la hora de documentar todo el procedimiento de pruebas realizado:

- Casos de prueba de Rendimiento (Utilizando la propuesta de plantilla para casos de prueba de Rendimiento V1.0.)
- Análisis general de los resultados. (Informe)

2.2.4.4.2 Análisis de entregables y Seguimiento

El posterior análisis y seguimiento de los resultados obtenidos en las pruebas será realizado por los desarrolladores del software. De estos resultados se destacan los aspectos más importantes que pueden interferir en gran medida en el rendimiento del sistema. Tomándose decisiones por parte del equipo de desarrollo para mejorarlos garantizando que no interfieran en próximas versiones del producto.

La base de los seguimientos está centrada en evitar la reincidencia de los elementos encontrados en las pruebas, los cuales deben ser depurados antes de la siguiente iteración de pruebas. Para verificar el seguimiento de los resultados de las pruebas, en la próxima iteración se deben comparar los valores resultados con los obtenidos en esta iteración y de esta comparación deben salir valores diferentes, de lo

contrario se podrá asumir que los errores no fueron corregidos. Esto será controlado por las versiones de las ejecuciones de pruebas en cada versión del producto.

Conclusiones

En este capítulo se realizó la descripción del procedimiento de pruebas de rendimiento diseñado. A lo largo del procedimiento se verificaron y se les dio respuesta a todos los aspectos que se definieron en un principio como objetivos para desarrollar un proceso correcto y eficaz de pruebas de rendimiento. Se espera que en los resultados de su aplicación se demuestre su fiabilidad, partiendo de que no existe una iniciativa precedente a la que se expone en este trabajo para la aplicación de pruebas de rendimiento.

CAPÍTULO 3: APLICACIÓN DEL PROCEDIMIENTO PARA PRUEBAS DE RENDIMIENTO Y EVALUACIÓN DE LOS RESULTADOS

3.1 Introducción

El objetivo de este capítulo es aplicar el procedimiento diseñado en el capítulo anterior y exponer los resultados obtenidos.

En este capítulo se realizarán y documentarán las pruebas diseñadas en el capitulo anterior, con el objetivo de llegar a conclusiones respecto a las mismas. Se aplicará la estrategia desarrollada para realizar pruebas de rendimiento en Aduana, para esto es necesario la selección de un sistema para verificar su rendimiento de acuerdo a los parámetros establecidos. El sistema seleccionado es Información Adelantada de Pasajeros (API).

Breve explicación del Sistema a probar

El Sistema de Información Adelantada de Pasajeros (API), es uno de los sistemas que forma parte del Sistema Único de Aduanas (SUA). API se desarrolla basándose en que nuestro país, como muchos otros países, ha incrementado el flujo de pasajeros, con ello ha aumentado el tráfico internacional de drogas, actividades delictivas, el terrorismo, entre otras ilegalidades que afectan la seguridad de los países, por lo que la Aduana exige que le sea entregada la Información Adelantada de Pasajeros y Tripulantes que entran al país desde el exterior. Este sistema posee un total de 14 casos de uso, los cuales son Información de Vuelos, Búsqueda Fonética, Conformar Parte, Vuelos Pasajeros, Estado Mensajes, Mensajes Basura, Conformar Mensaje, Subir Mensaje, Procesar Mensaje, Validar Mensaje, Obtener Correo, Obtener Sita, Generar Fichero API y Mostrar detalle. De ellos los primeros 8 son funcionalidades externas del sistema que poseen pantalla, los últimos 6 son funcionalidades internas que interactúan con las externas en su funcionamiento y no poseen interfaz.

3.2 Aplicación del procedimiento para pruebas de rendimiento al Sistema Único de Aduanas.

3.2.1 Alcance

Se realizarán pruebas de rendimiento, Pruebas de Carga y Estrés, para las mismas se utiliza la técnica de Caja Negra, estas pruebas se dirigen a probar el rendimiento de las funcionalidades que poseen interfaz de usuario del Sistema de Información Adelantada de pasajeros (API), las cuales son Información de Vuelos, Búsqueda Fonética, Conformar Parte, Vuelos Pasajeros, Estado Mensajes, Mensajes Basura, Conformar Mensaje y Subir Mensaje.

3.2.2 Objetivos

Verificar el rendimiento de las funcionalidades del Sistema API descritas en el alcance, comprobando cada una de ellas en base a los valores suministrados por los diseñadores de la aplicación, respecto a los resultados esperados para los cuales fue implementado. Comparar valores respecto a diferentes entradas, buscando diferencias que no deben existir. Y en caso de que existan documentarlas e informarlas para su posterior corrección.

3.2.3 Fases a desarrollar

3.2.3.1 Planificación

3.2.3.1.1 Estudio de la Documentación

El sistema API donde se van a realizar las pruebas, tiene documentados los siguientes aspectos dentro del expediente del proyecto: la especificación de los casos de uso, el glosario de términos, el manual de usuario, y el documento de especificación de requerimientos.

3.2.3.1.2 Elección de las variables adecuadas

De acuerdo a los requerimientos establecidos para el sistema API, las variables de entrada que se van a utilizar son 50 usuarios concurrentes para cada una de las funcionalidades Información de Vuelos, Búsqueda Fonética, Conformar Parte, Vuelos Pasajeros, Estado Mensajes y Mensajes Basura, pues está establecido que debido a que en Cuba existen 13 Aduanas a nivel nacional y que API va a tener no más de 3 usuarios por aduana trabajando en este sistema, el sistema debe permitir sin problemas la conexión de 50 personas simultáneamente. Entonces queda definido que en las pruebas de carga para estas funcionalidades este será el valor utilizado.

En el caso de las funcionalidades pertenecientes a Gestionar mensajes, las cuáles son Conformar Mensaje y Subir Mensaje, teniendo en cuenta que estas opciones son poco usadas por la Aduana y que solo se utiliza en caso de un vuelo especial determinado, se estableció como requerimiento que la variable de entrada tendrá como valor 5.

Para las pruebas de Estrés se utilizará el valor definido para las pruebas de carga con el objetivo de comprobar si el sistema está apto para recibir esa carga de trabajo bajo condiciones de estrés y a su vez se establecerán casos de prueba para las funcionalidades buscando el valor de conexión que provoca que la aplicación deje de responder.

Para ambos casos las variables de salida serán analizadas verificando los valores obtenidos como respuesta del sistema, comprobándolos con una variación de cantidad en las conexiones de usuarios. Esto se refiere a que en las pruebas de Carga la variable que tiene como valor 50, tomara primero valor 25 para comprobar el rendimiento con un número de peticiones media, logrando de esta forma valores para comparar a la hora de dar resultados. En el caso de la variable que tiene por valor 5 será doblada con el mismo objetivo.

3.2.3.1.3 Estudio de la topología de red disponible

Topología de los laboratorios pertenecientes al proyecto Aduana

La Red de los laboratorios pertenecientes a Aduana cuenta con una Topología de Árbol, la cual posee un nodo de enlace troncal, ocupado por un switch, desde el cual se ramifican los demás nodos. Desde una visión topológica, la conexión en árbol, es parecida a una serie de redes en estrellas interconectadas salvo en que estas tienen un nodo central. La falla de un nodo no implica interrupción en las comunicaciones. Se comparte el mismo canal de comunicaciones.

Análisis y Estudio de la Organización de la Red.

El docente donde se encuentran los laboratorios pertenecientes al proyecto Sistema Único de Aduanas cuenta con dos nodos centrales, la conexión entre los laboratorios se efectúa a través de switches capas 2 y 3, entre ellos hay un Firewall, el Switch de Capa 3 es el encargado de todos los laboratorios con que cuenta el Centro.

Características más significativas de los Switch.

Switch capa 2	Switch capa 3
• Equivalentes a los bridges multipuertos. • Baja latencia y alto rendimiento. • Segmentar la red en dominios de colisión por puertos y dominios de broadcast. • Entrega de tráfico en base a dirección MAC.	• Combinación de la funcionalidad de los switch capa 2 y de las características de los routers. • Alto rendimiento. • Entrega trafico basado en direcciones IP (cuando enruta la primera vez) y direcciones MAC (cuando conmuta).

Características principales de la red de los laboratorios del proyecto Sistema Único de Aduanas:

- Se expande en un área relativamente pequeña (dos laboratorios). Conformada por un número de 62 computadoras, las cuales se conectan entre sí por fibra óptica y cable UTP.
- Los nodos que conforman esta red son máquinas (PC) que cuentan con su propio CPU, disco duro y software y tienen la capacidad de conectarse a la red en un momento dado.
- Esta red es capaz de transmitir datos a velocidades muy rápidas, algunas incluso más rápido que por línea telefónica; pero las distancias son limitadas.
- Esta red cuenta con dos servidores, encargados de llevar el control de la red: Servicios web, Bases de Datos, Proxy, DNS.
- El medio de conexión entre las PC de los laboratorios es por cable Par Trenzado, de tipo UTP de categoría 5, están conectados a puntos de red y estos a un Switch que existe en cada laboratorio.

Características de los servidores

Características de los Servidores Web y DNS

Estos servidores de un modo muy concreto permiten la comunicación entre los usuarios a través de la red, la transferencia rápida de documentos, ficheros y el acceso a un conjunto de páginas WEB que ya conforman parte de los servicios que brinda la Intranet, de manera que esté creado un sistema de acceso multiusuario que incorpora en sí sistemas de búsqueda de información en bibliotecas, archivos, guías telefónicas.

Características del Servidor de Base de Datos

Todos los datos del proyecto se manejan a través de un sistema gestor de base de datos (DBMS, Database Management System) cliente-servidor, más específicamente la versión Oracle Standard Edition ONE versión 11g. Las estaciones de trabajo que actúan como clientes, pueden enviar peticiones al servidor por la red y luego este responde. Las estaciones de trabajo clientes manejan la presentación de los datos e interactúan con los usuarios mientras el servidor realiza las operaciones más duras. Su utilización reduce la probabilidad de corrupción de la información, siendo fácil de mantener.

Características de las PC Clientes:

Los dos laboratorios de producción pertenecientes al proyecto Aduana cuentan con 62 PC. A continuación se exponen las características más relevantes de las mismas:

- **Pentium D:** Con 2 GB de RAM y 80 como capacidad del disco duro. Estas PC funcionan como servidores dentro del proyecto.

- **Core 2 Duo:** Con 1 GB de RAM y 160 GB de capacidad en disco duro.

3.2.3.1.4 Diseño del entorno de pruebas

Para la aplicación del procedimiento de pruebas de rendimiento diseñado al software se identifica un entorno con las siguientes características:

Dos servidores locales, los cuales poseerán Oracle Standard Edition ONE versión 11g como gestor de Bases de Datos y como sistema operativo Linux.

Las PC clientes a utilizar en las pruebas tienen instalado como sistema operativo Windows XP Servi Pack 2 o Linux, con Kaspersky Workstation 6.0 como antivirus. Para el apoyo de los usuarios las máquinas deben tener instalado el Paquete de Office 2003 o 2007 o al menos las herramientas básicas (Microsoft Word, Microsoft Excel). Estas PC cuentan con 1 GB de RAM.

Especificaciones de Hardware:

Hardware	Datos
Microprocesador	Intel(R) Core(TM)2 Duo CPU E4500 a 2.20 GHz
Memoria RAM	1 GB (PC Clientes) 2GB (Servidores)
Ancho de Banda y tipo de conexión de red	10/100 Mbps UTP 5 LAM
Disco Duro	160 GB(PC Clientes) 80 GB (Servidores)

Especificaciones de Software:

Software	Datos
Sistema Operativo	Windows XP Servi Pack 2 o Linux.
Gestor de Base de Datos	Oracle Standard Edition ONE version 11g
Antivirus	Kaspersky Workstation 6.0
Otras herramientas	Microsoft Office 2003 o 2007
Maquina Virtual	Java Virtual Machine 1.3 o superior
Software de prueba	JMeter Versión 2.3.1

3.2.3.1.5 Diseño de las pruebas en la herramienta

El proceso de diseño de las pruebas del Sistema Único de Aduanas usando la herramienta JMeter consta de 2 elementos fundamentales, estos son:

- Grabación de los escenarios de prueba.
- Confección de los Planes de Prueba.

Para la realización de la grabación de los escenarios de prueba es necesario saber que las pruebas a realizar son de tipo complejas. Para su realización se configura el navegador Firefox con el proxy definido en el JMeter, y se pone a correr la aplicación realizando las tareas en ella que se desea que sean grabadas por el programa para su posterior simulación. Una vez que esto ocurre queda grabado el escenario de prueba.

Para la confección de los planes de prueba se utilizan los elementos definidos en el procedimiento.

Para más información ver anexo Manual de Usuario del JMeter.

3.2.3.1.6 Modelo de la planificación

En este paso se prosigue a llenar los datos solicitados en la plantilla para casos de prueba de Carga y Estrés.

Caso de prueba 1:

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Información de Vuelos	El sistema muestra los vuelos que han llegado en la fecha actual.

Variables:

Variables	Valores
Número de usuarios concurrentes:	25

[Número de hilos]	
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	50
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Los restantes casos de prueba de Carga se encuentran en los anexos:

Anexo 8: Caso de prueba Carga: Búsqueda Fonética

Anexo 9: Caso de prueba Carga: Conformar Parte

Anexo 10: Caso de Prueba Carga: Conformar Mensaje

Anexo 11: Caso de Prueba Carga: Estado de Mensajes

Anexo 12: Caso de Prueba Carga: Mensajes Basura

Anexo 13: Caso de Prueba Carga: Subir Mensaje

Anexo 14: Caso de Prueba Carga: Vuelos Pasajeros

Caso de prueba 2:

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Estrés

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Búsqueda Fonética	El sistema muestra a las

	personas que han viajado con anterioridad y sus datos coinciden fonéticamente con los datos especificados por el especialista de la aduana.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Los restantes casos de prueba de Carga se encuentran en los anexos:

Anexo 1: Caso de prueba Estrés: Conformar Mensaje

Anexo 2: Caso de prueba Estrés: Conformar Parte

Anexo 3: Caso de prueba Estrés: Estado de Mensajes

Anexo 4: Caso de prueba Estrés: Información de Vuelos

Anexo 5: Caso de prueba Estrés: Mensajes Basura

Anexo 6: Caso de prueba Estrés: Subir Mensaje

Anexo 7: Caso de prueba Estrés: Vuelos Pasajeros

3.2.3.2 Aseguramiento

3.2.3.2.1 Diseño de la carga de trabajo

Se toman todas las medidas necesarias para no cometer ninguno de los errores explicados en el procedimiento dentro de este punto.

3.2.3.2.2 Adecuación de la configuración

Partiendo del punto de que se desea realizar pruebas duras al sistema buscando los valores para los cuales este no funciona, se realizarán periodos cortos para la conexión de los usuarios basándonos en lo que plantea este aspecto en el procedimiento.

3.2.3.2.3 Precauciones a tomar para proceder con las pruebas

Los sistemas a probar son copias estables de los sistemas originales, de esta forma se garantiza que los valores sean completamente confiables y que en caso de alguna perdida de información el sistema principal no se ve afectado.

3.2.3.3 Ejecución

3.2.3.3.1 Ejecución de las pruebas

Se procede a la ejecución de las pruebas, una vez que esté todo listo y bien definido y el personal esté capacitado para proceder con las mismas.

3.2.3.3.2 Validación de los resultados

- Utilización de la CPU: Menor de 70%
- Utilización de la memoria: Menor de 75 %
- Tasa de errores: Menor de 1%

3.2.3.4 Análisis e interpretación de los resultados

Todas las páginas previstas fueron visitadas y probadas, pues respecto a la conexión no surgieron problemas. Los sistemas fueron accesibles en todo momento, garantizando la calidad de los resultados. Los resultados para los casos de prueba expuestos anteriormente se muestran a continuación:

Caso de prueba 1:

Tipo de prueba: Prueba de Carga

Resultados Generales:

Número de usuarios: 25

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Información de Vuelos	2650	332	157	0	13500	343	2.83	70.2/seg	831.4

Número de Usuarios: 50

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Información de Vuelos	5300	699	390	0	28000	797	2.83	65.6/seg	776.8

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de imagen

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 2.83%, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que le sistema responde satisfactoriamente, pues para 50 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 25 usuarios concurrentes. El número de peticiones esperado era 2625 para 25 usuarios concurrentes y 5275 para 50 usuarios concurrentes, en ambos casos se realizaron 25 peticiones más pues uno de los vínculos de error requirió ese número de peticiones extras.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Información de Vuelos cumple con los requerimientos establecidos.

Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Debe realizarse la verificación del código de los vínculos que se muestran a continuación pues presentan un tiempo de respuesta inaceptable por encima de los valores esperados.

- /sua/web/sarpia_dev.php/internet/jsonInformacionPasajerosVuelo
- /sua/web/sarpia_dev.php/internet/jsonReporteVuelosPorFechas

Aun así el tiempo general de respuesta de la funcionalidad es correcto pues el mismo está por debajo de 1 segundo en las peticiones.

Caso de prueba 2:

Tipo de prueba: Prueba de Estrés

Resultados Generales:

Cantidad de usuarios: 10

Tiempo entre conexión y conexión: 5

Numero de iteraciones: 26

Total: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Búsqueda Fonética	15193	106	78	0	3640	188	3.83	74.5/seg	1023.4

Cantidad de usuarios: 130

Tiempo entre conexión y conexión: 2

Numero de iteraciones: 2

Total: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Búsqueda Fonética	11230	776	250	0	32390	1750	4.82	18.8/seg	307.6

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de imagen

Durante la ejecución de la prueba el CPU se mantuvo entre el 55 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 3.83% y el 4.82%, valor que no cumple

con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 15193 peticiones lo que equivale a 187 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 11230 peticiones, valor que equivale a 142 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable cayendo a valores inaceptables y llevando a un lento avance de la aplicación que tardo 3 minutos en reponerse.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 142 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de este no se acepta.

Problemas generales:

Luego de la aplicación de las pruebas de rendimiento al Sistema de Información Adelantada de Pasajeros se puede llegar a las siguientes conclusiones respecto a los errores encontrados:

- **Errores en vínculos de la aplicación:**

Se encuentran errores al 100 % en los vínculos que se mencionan en los casos de prueba, pues estos vínculos no responden en ningún momento a ninguna petición, logrando retrasos en la respuesta del rendimiento del sistema.

- **Tiempos de ejecución largos para las extensiones de Java Script:**

Se recomienda la revisión de las páginas implementadas con Java Script, pues consumen un tiempo muy por encima de lo establecido, retardando el tiempo de respuesta de la aplicación.

- **Errores en el sistema por la carga en la herramienta:**

El sistema presentó errores de respuesta debido a que la herramienta JMeter muestra grandes niveles de estrés cuando se sobrecarga la concurrencia, lo que obliga al uso de la misma en computadoras con memorias RAM altas y de gran velocidad.

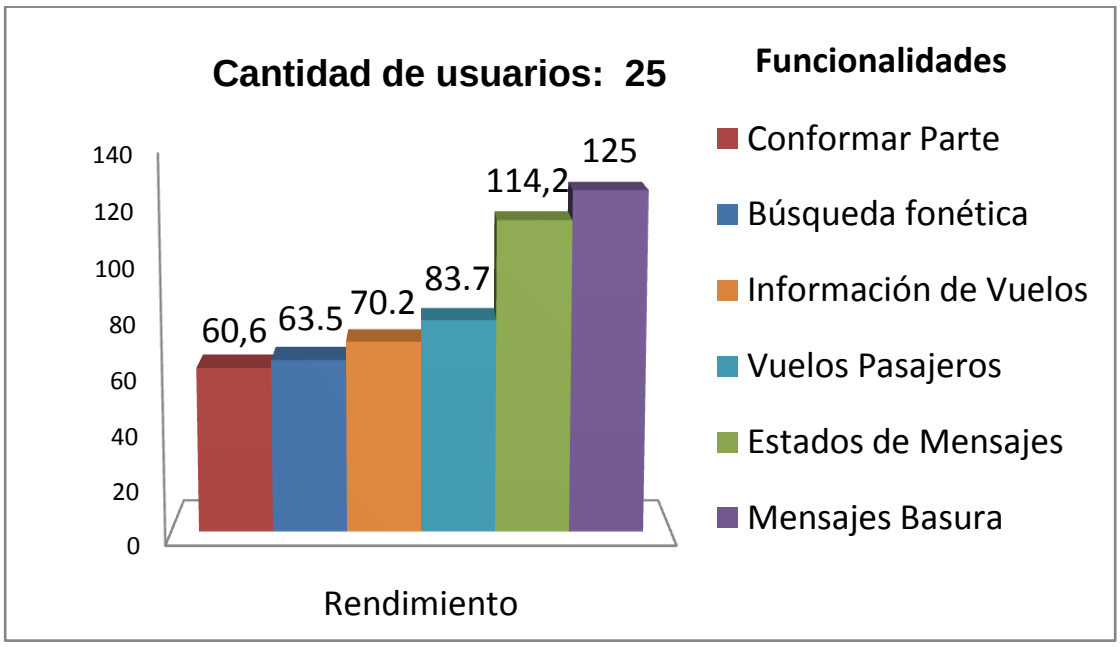
- **Problemas de resistencia de la Base de Datos:**

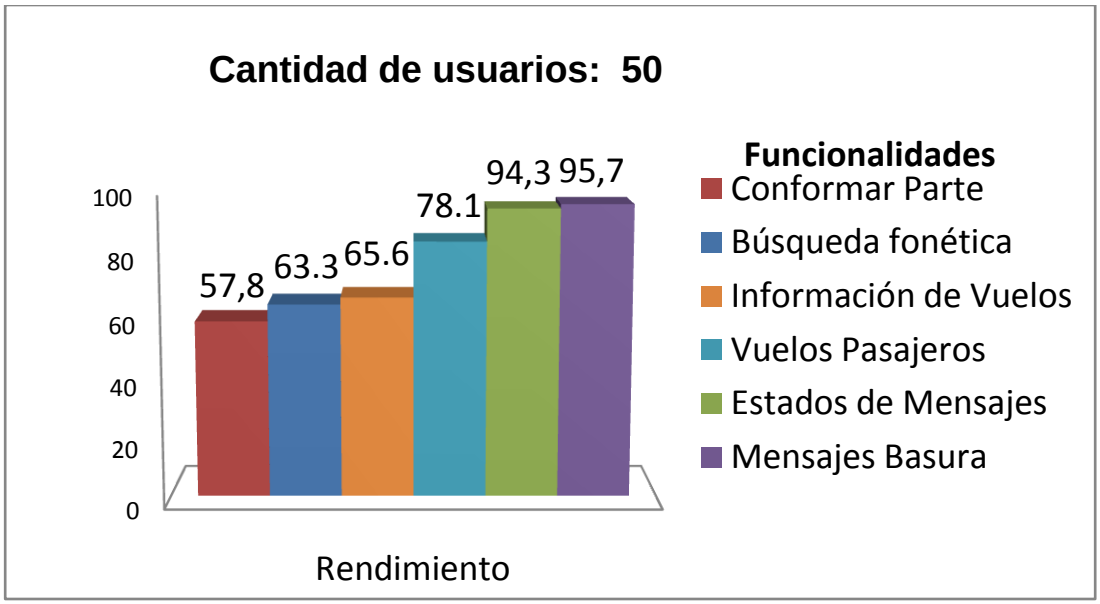
En ocasiones cuando se realizaban pruebas de estrés con un número de concurrencia alto, la base de datos dejaba de responder por la cantidad de peticiones que recibía simultáneamente. Provocando la caída de la aplicación, y pruebas fallidas por demasiados errores.

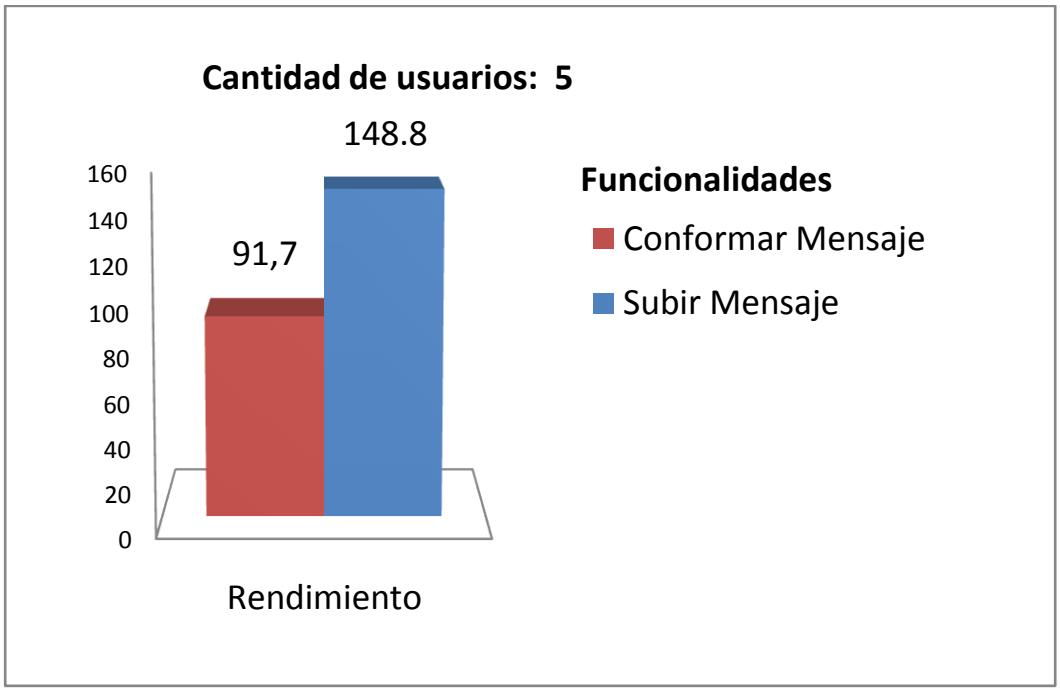
- **Problemas del navegador seleccionado:**

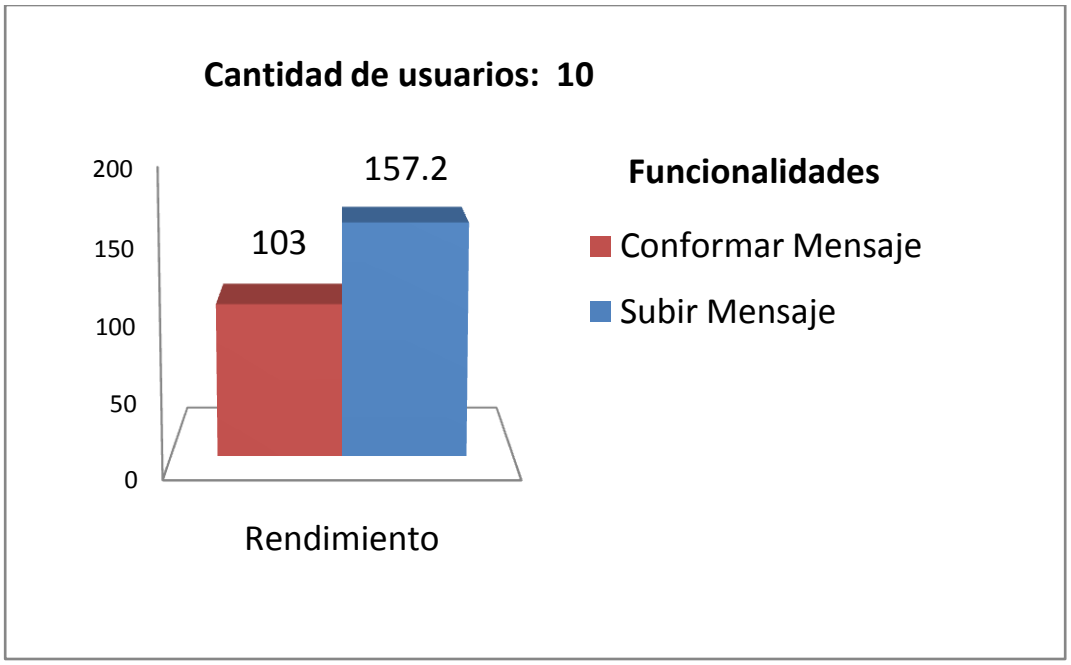
Las pruebas realizadas con el navegador Firefox presentan un por ciento de errores muy superior a cuando son efectuadas con Internet Explorer, debido a errores en elementos del mismo que no son reconocidos a la hora de responder a las peticiones.

Resultados Generales de rendimiento para pruebas de Carga:

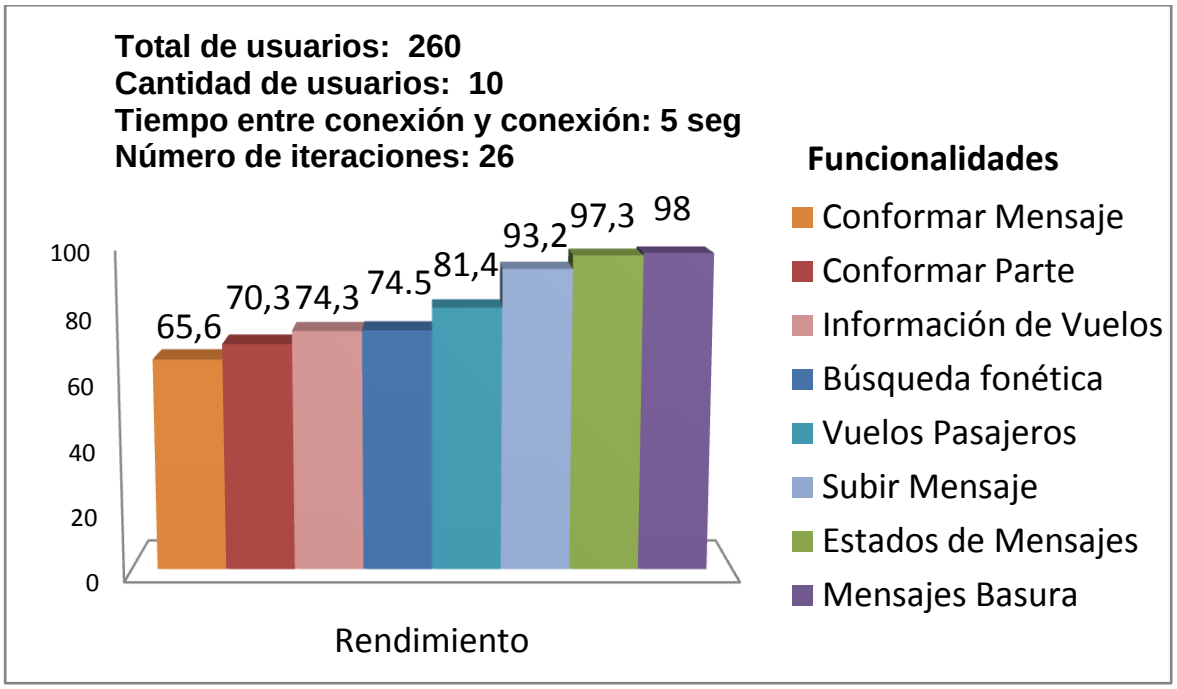


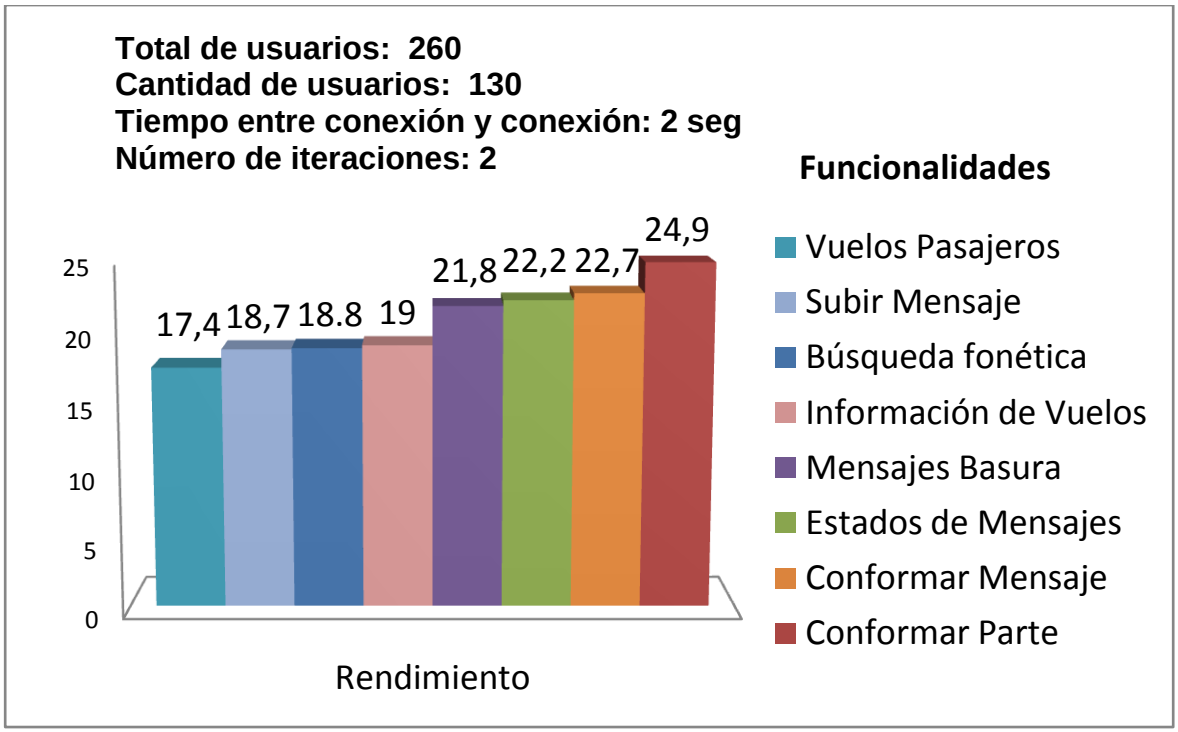

Cantidad de usuarios: 50
100
80
60
40
20
0
57,8
63.3
65.6
78.1
94,3
95,7
Rendimiento
Funcionalidades
Conformar Parte
Búsqueda fonética
Información de Vuelos
Vuelos Pasajeros
Estados de Mensajes
Mensajes Basura


Cantidad de usuarios: 5
160
140
120
100
80
60
40
20
0
91,7
148.8
Rendimiento
Funcionalidades
Conformar Mensaje
Subir Mensaje


Cantidad de usuarios: 10
200
150
100
50
0
157.2
103
Funcionalidades
Conformar Mensaje
Subir Mensaje
Rendimiento

Resultados generales de rendimiento para pruebas de Estrés:





3.3 Conclusiones

Luego de la aplicación del procedimiento para la realización de pruebas de rendimiento, en especial Carga y Estrés, a el Sistema Único de Aduanas y dentro de este al Sistema de Información Adelantada de Pasajeros (API) se llegó a la conclusión de la necesidad del establecimiento de un procedimiento para la aplicación de las mismas, que permite una organización del trabajo además de la obtención de resultados satisfactorios para la mejora de la calidad.

En este capítulo se aplicó el procedimiento desarrollado en el capítulo 2, con el objetivo de demostrar su validez y de que sea utilizado para la realización de las pruebas de rendimiento en el Sistema Único de Aduanas, partiendo de que no existe ningún antecedente de procedimientos para la realización de pruebas de rendimiento ni la existencia de plantillas como la que se propone en este trabajo para la documentación de los casos de prueba de rendimiento, incluyendo los resultados obtenidos.

El procedimiento fue aplicado con éxito, considerando que el éxito en un proceso de pruebas depende de la afirmación:

“Las pruebas tienen éxito si encuentran un error no detectado hasta ese momento.”

CONCLUSIONES GENERALES

El establecimiento de un procedimiento a la hora de realizar pruebas a un software garantiza organización, calidad, seguridad e integridad en las pruebas que serán realizadas pues el principal objetivo es velar por la calidad del software y que esta sea garantizada de la mejor forma posible.

Al aplicarle el procedimiento para pruebas de rendimiento al Sistema de Información Adelantada de Pasajeros (API) se llegaron a las siguientes conclusiones:

Las pruebas a los productos en la etapa de desarrollo son un factor determinante en la calidad y por ende en la aceptación del producto por parte de los clientes y usuarios.

Se desarrolló y aplicó un procedimiento para pruebas de rendimiento de acuerdo con las características del Sistema Único de Aduanas, lo que permite su aplicación en cualquiera de los subsistemas que forman parte del mismo, demostrando lo antes dicho con la aplicación exitosa del procedimiento para pruebas de rendimiento en el Sistema de Información Adelantada de Pasajeros.

Se desarrollaron casos de prueba que cubrieron todas las funcionalidades con interfaz que puede ejecutar el sistema garantizando de esta forma verificar el rendimiento en todas y ver sus posibilidades de generar error.

Se compararon los resultados esperados y los resultados reales y se documentaron todas las diferencias.

Se logró desarrollar una adecuada documentación de todo el procedimiento de pruebas utilizado que servirá de apoyo para la corrección de errores encontrados en cada una de las pruebas y para futuros productos que se desarrollen.

El sistema de Información Adelantada de Pasajeros tendrá mayor calidad una vez que se revise la documentación resultante de la aplicación del procedimiento y se realicen los cambios para eliminar los errores encontrados.

Luego del estudio del trabajo realizado se puede llegar a la conclusión de que este cumplió con los objetivos establecidos en un principio.

RECOMENDACIONES

Se recomienda:

Teniendo en cuenta que en este trabajo solo se realiza una iteración se recomienda la realización futura de otras iteraciones a la aplicación que permitan la verificación de la rectificación de los errores.

El estudio de herramientas para la realización de pruebas de rendimiento que puedan surgir en el futuro y que puedan ser usadas, en una modificación del procedimiento para pruebas de rendimiento desarrollado.

El estudio de los diferentes tipos de pruebas de rendimiento que existen y que no son desarrolladas en este procedimiento para su posterior aplicación.

BIBLIOGRAFÍA

Alarcos. Grupo Alarcos Universidad de Castilla. *Grupo Alarcos Universidad de Castilla.* [En línea] [Citado el: 2 de marzo de 2009.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf.

Beck, Kent. 2002. *Test-Driven Development By Example.* 2002.

Buades, Gabriel. UIB. *Departamento de Ciencias Matemáticas e Informáticas.* [En línea] [Citado el: 13 de abril de 2009.] http://dmi.uib.es/~bbuades/calidad/index.htm.

Centro de Innovación de TI. *Pruebas de Software.* [En línea] [Citado el: 16 de marzo de 2009.] http://www.citip.org.mx/servicios/software/pruebaslab/Pages/PruebasSoftware.aspx.

CES. 2004. Centros de Ensayos de Software. *Centros de Ensayos de Software.* [En línea] 2004. [Citado el: 18 de abril de 2009.] http://www.ces.com.uy.

Cueva Lovelle, Juan Manuel. 1999. GIDIS. *Grupo de J + D en Ingeniería de Software.* [En línea] 1999. [Citado el: 2009 de abril de 21.] http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

Daswani, P., Rodrigo, J.J y Rosales, J. Medicion de rendimiento de servicios WMS CON JMeter. *Medicion de rendimiento de servicios WMS CON JMeter.* [En línea] [Citado el: 24 de abril de 2009.] http://www.idee.es/resources/presentaciones/JIDEE08/ARTICULOS_JIDEE2008/Articulo84.pdf.

Departamento de Lenguas y Sistemas Informáticos. Universidad de Granada. *Departamento de Lenguas y Sistemas Informáticos. Universidad de Granada.* [En línea] [Citado el: 17 de marzo de 2009.] http://lsi.ugr.es/~ig1/docis/pruso.pdf.

EP. 1999. Extreme Programming: A gentle introduction. *Extreme Programming: A gentle introduction.* [En línea] 1999. [Citado el: 24 de marzo de 2009.] http://www.extremeprogramming.org/.

Fernández Carrasco, Oscar M, García León, Delba y Beltrán Benavides, Alfa. 1995. *Un enfoque actual sobre la calidad del software.* 1995.

Fernández, Luis. 2002. Presentación del Grupo de Calidad de Software. *Presentación del Grupo de Calidad de Software.* [En línea] 2002. [Citado el: 25 de febero de 2009.] http://www.ati.es/gt/calidad-software/presentacion.htm.

Guzmán Arenas, Adolfo. 2003. *Mitos, creencias y supersticiones sobre la calidad del software y de su enseñanza.* 2003.

Guzmán Cortés, Oscar Hernando. 2004. Aplicación práctica del diseño de pruebas de software a nivel de programación. *Aplicación práctica del diseño de pruebas de software a nivel de programación.* [En línea] 2004. [Citado el: 16 de marzo de 2009.] http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf.

HT. 2006. Lista de herramientas para testeo de aplicaciones web. *Lista de herramientas para testeo de aplicaciones web.* [En línea] 20 de octubre de 2006. [Citado el: 16 de marzo de 2009.] http://sentidoweb.com/2006/10/20/lista-de-herramientas-para-testeo-de-aplicaciones-web.php.

IEEE. 1990. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Standard Glossary of Software Engineering Terminology.* [En línea] 28 de septiembre de 1990. [Citado el: 22 de febrero de 2009.] http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2004. El Proceso Unificado de Desarrollo de Software. [En línea] 2004. http://bibliodoc.uci.cu/pdf/reg00060.pdf.

Jimenez, Darwin y Aguirre, Carlos Eduardo. Presentacion de pruebas. *Modelo de Pruebas de Software.* [En línea] [Citado el: 5 de abril de 2009.] http://www.slideshare.net/dajigar/presentacion-pruebas-presentation.

JMeter, Jakarta. 1999. The Jakarta Site. *Jakarta JMeter.* [En línea] 1999. [Citado el: 13 de mayo de 2009.] http://jakarta.apache.org/jmeter/.

Letelier, Patricio. Departamento Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. *Departamento Sistemas Informáticos y Computación. Universidad Politécnica de Valencia.* [En línea] [Citado el: 19 de febrero de 2009.] http://www.dsic.upv.es/.

Lomprey, Gérald y Hernández, Saulo. 2008. LA IMPORTANCIA DE LA CALIDAD EN EL DESARROLLO DE PRODUCTOS DE SOFTWARE. *LA IMPORTANCIA DE LA CALIDAD EN EL DESARROLLO DE PRODUCTOS DE SOFTWARE.* [En línea] 2008. [Citado el: 14 de abril de 2009.] http://fit.um.edu.mx/departamentodeinvestigacion/publicaciones/Technical%20Report%20COMP-018-2008.pdf.

López Quesada, Juan Antonio. 2005. Fundamentos de Ingeniería del Software. *Fundamentos de Ingeniería del Software.* [En línea] 2005. [Citado el: 21 de abril de 2009.] http://dis.um.es/~lopezquesada/FISw.htm.

MWAST. 2001. Microsoft Web Application Stress Tool. *The Apache Software Fundation.* [En línea] 12 de junio de 2001. [Citado el: 16 de marzo de 2009.] http://people.apache.org/~sgoeschl/presentations/was-20010628.pdf.

OpenSTA Users Home Page. *OpenSTA Users Home Page.* [En línea] [Citado el: 16 de marzo de 2009.] http://opensta.org/.

Orjuela Darte, Ailín. 2008. *Las Metodologías de Desarrollo Ágil como una oportunidad para la Ingeniería de Software educativa.* Colombia : s.n., 2008.

Pérez Lamancha, Beatriz. 2004. Centros de Ensayos de Software. *Gestión de las Pruebas Funcionales.* [En línea] 2004. [Citado el: 5 de abril de 2009.] http://www.ces.com.uy/documentos/imasd/CES-PRIS_Gestion_Testing.pdf.

Pressman, Roger. Ingenieria de Software un enfoque practico. [En línea] http://bibliodoc.uci.cu/pdf/reg02689.pdf.

2005. Prueba de Carga Básica en JMeter. *Prueba de Carga Básica en JMeter.* [En línea] 2005. http://www.osmosislatina.com/jmeter/pruebabasica.htm.

Sánchez, Liudmila. 2008. *Estrategia para Pruebas Automatizadas de Carga y Estrés.* La Habana : s.n., 2008.

Sarco, Jose Pablo. Testing en Español. *Testing en Español.* [En línea] [Citado el: 15 de abril de 2009.] http://josepablosarco.wordpress.com/2009/02/03/performance-testing-con-jmeter-paso-por-paso-para-grabar-escenarios/.

SQS. 2007. Software Quality Systems S.A. *Software Quality Systems S.A.* [En línea] 2007. [Citado el: 25 de abril de 2009.] http://www.sqs.es/es/.

Teruel, Alejandro. 2001. El plan de pruebas. *El plan de pruebas.* [En línea] Enero - Marzo de 2001. [Citado el: 27 de febrero de 2009.] http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html.

Valencia, Gustavo. Servicio de Pruebas de Software. *Servicio de Pruebas de Software.* [En línea] http://www.greensqa.com/archivos/Servicio%20Pruebas%20de%20Software.pdf.

Vega Jara, Herman y Herrera Ruztort, Cristián. 2004. GNOME Bluetooth Control Remoto Proyect. *GNOME Bluetooth Control Remoto Proyect.* [En línea] 29 de diciembre de 2004. [Citado el: 17 de abril de 2009.] http://gbtcr.chileforge.cl/informe_gbtcr.pdf.

Vegas, Sira, Juristo, Natalia y Moreno, Ana M. 2005. TÉCNICAS DE EVALUACIÓN DE SOFTWARE. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* [En línea] 17 de octubre de 2005. [Citado el: 10 de abril de 2009.] http://is.ls.fi.upm.es/docencia/erdsi/Documentacion_Evaluacion_7.pdf.

BIBLIOGRAFIA CITADA

CES. 2004. Centros de Ensayos de Software. *Centros de Ensayos de Software.* [En línea] 2004. [Citado el: 18 de abril de 2009.] http://www.ces.com.uy.

Cueva Lovelle, Juan Manuel. 1999. GIDIS. *Grupo de J + D en Ingeniería de Software.* [En línea] 1999. [Citado el: 2009 de abril de 21.] http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Calidad_software.PDF.

Fernández, Luis. 2002. Presentación del Grupo de Calidad de Software. *Presentación del Grupo de Calidad de Software.* [En línea] 2002. [Citado el: 25 de febero de 2009.] http://www.ati.es/gt/calidad-software/presentacion.htm.

Guzmán Arenas, Adolfo. 2003. *Mitos, creencias y supersticiones sobre la calidad del software y de su enseñanza.* 2003.

IEEE. 1990. IEEE Standard Glossary of Software Engineering Terminology. *IEEE Standard Glossary of Software Engineering Terminology.* [En línea] 28 de septiembre de 1990. [Citado el: 22 de febrero de 2009.] http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2004. El Proceso Unificado de Desarrollo de Software. [En línea] 2004. http://bibliodoc.uci.cu/pdf/reg00060.pdf.

Jimenez, Darwin y Aguirre, Carlos Eduardo. Presentacion de pruebas. *Modelo de Pruebas de Software.* [En línea] [Citado el: 5 de abril de 2009.] http://www.slideshare.net/dajigar/presentacion-pruebas-presentation.

JMeter, Jakarta. 1999. The Jakarta Site. *Jakarta JMeter.* [En línea] 1999. [Citado el: 13 de mayo de 2009.] http://jakarta.apache.org/jmeter/.

López Quesada, Juan Antonio. 2005. Fundamentos de Ingeniería del Software. *Fundamentos de Ingeniería del Software.* [En línea] 2005. [Citado el: 21 de abril de 2009.] http://dis.um.es/~lopezquesada/FISw.htm.

OpenSTA Users Home Page. *OpenSTA Users Home Page.* [En línea] [Citado el: 16 de marzo de 2009.] http://opensta.org/.

Pressman, Roger. Ingenieria de Software un enfoque practico. [En línea] http://bibliodoc.uci.cu/pdf/reg02689.pdf.

SQS. 2007. Software Quality Systems S.A. *Software Quality Systems S.A.* [En línea] 2007. [Citado el: 25 de abril de 2009.] http://www.sqs.es/es/.

Teruel, Alejandro. 2001. El plan de pruebas. *El plan de pruebas.* [En línea] Enero - Marzo de 2001. [Citado el: 27 de febrero de 2009.] http://www.ldc.usb.ve/~teruel/ci4713/clases2001/planPruebas.html.

Vega Jara, Herman y Herrera Ruztort, Cristián. 2004. GNOME Bluetooth Control Remoto Proyect. *GNOME Bluetooth Control Remoto Proyect.* [En línea] 29 de diciembre de 2004. [Citado el: 17 de abril de 2009.] http://gbtcr.chileforge.cl/informe_gbtcr.pdf.

Vegas, Sira, Juristo, Natalia y Moreno, Ana M. 2005. TÉCNICAS DE EVALUACIÓN DE SOFTWARE. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* [En línea] 17 de octubre de 2005. [Citado el: 10 de abril de 2009.] http://is.ls.fi.upm.es/docencia/erdsi/Documentacion_Evaluacion_7.pdf.

ANEXOS:

Anexo 1: Caso de prueba Estrés: Conformar Mensaje

Tipo de prueba: Prueba de Estrés

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Conformar Mensaje	Se recogen los datos del vuelo, pasajeros y tripulantes necesarios para conformar el mensaje API. El sistema posibilita guardar el mensaje en la PC de actor, el mensaje será generado con el formato un_edifact.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Resultados Generales:

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg

Conformar mensaje	9565	138	78	0	3469	344	6.54	65.6/seg	1499.7

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Conformar mensaje	8219	1895	1500	0	53312	3172	7.49	22.7/seg	559.9

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 72 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 6.54% y el 7.49%, valor que no cumple con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 9565 peticiones lo que equivale a 212 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 8219 peticiones, valor que equivale a 182 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable cayendo a valores que equivalen a un tercio de lo esperado y con un tiempo de respuesta superior al establecido.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 182 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de éste no se acepta.

Anexo 2: Caso de prueba Estrés: Conformar Parte

Tipo de prueba: Prueba de Estrés

Funcionalidad (es)	Comentario
Conformar Parte	Obtener un resumen de la cantidad de vuelos que entraron al país por aeropuerto, conocer la cantidad de aerolíneas que viajaron a ese aeropuerto y el total de todos los vuelos que entraron al país.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Resultados Generales:

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Conformar parte	9045	127	93	0	2172	265	9.72	70.3/seg	1707.6

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Conformar parte	8073	1738	1406	0	29954	2859	9.85	24.9/seg	634.2

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 65 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 9.72% y el 9.85%, valor que no cumple con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 9045 peticiones lo que equivale a 231 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 8073 peticiones, valor que equivale a 207 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable mostrando valores muy por debajo de los esperados y el tiempo de respuesta para el 90 por ciento de las peticiones se mantuvo por encima de los valores correctos esperados duplicando el valor esperado.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 207 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de éste conlleva a un funcionamiento inestable del sistema.

Anexo 3: Caso de prueba Estrés: Estado de Mensajes

Tipo de prueba: Prueba de Estrés

Funcionalidad (es)	Comentario
Estado de mensajes	En este caso de uso el contacto puede consultar el estado de los mensajes enviados por la aerolínea a la que pertenece. La información puede ser filtrada por fecha de llegada del mensaje, vía de llegada y estado.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Resultados Generales:

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Estado de mensajes	15756	90	63	0	2079	172	3.76	97.3/seg	1272.5

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Estado de mensajes	11758	1121	906	16	24391	1922	4.85	22.2/seg	344.2

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 70 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 3.76% y el 4.85%, valor que no cumple con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 15756 peticiones lo que equivale a 196 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 11758 peticiones, valor que equivale a 146 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable y con un tiempo de respuesta ligeramente superior al establecido.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 146 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de éste conlleva a un funcionamiento inestable del sistema.

Anexo 4: Caso de prueba Estrés: Información de Vuelos

Tipo de prueba: Prueba de Estrés

Funcionalidad (es)	Comentario
Información de Vuelos	El sistema muestra los vuelos que han llegado en la fecha actual.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Resultados Generales:

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Información de vuelos	10721	118	79	0	1703	234	8.17	74.3/seg	1500.9

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg

Información de vuelos	8490	1548	1391	15	15562	2531	9.15	19.0/seg	449.5

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 65 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 9.72% y el 9.85%, valor que no cumple con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 10721 peticiones lo que equivale a 228 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 8490 peticiones, valor que equivale a 180 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable mostrando valores muy por debajo de los esperados y el tiempo de respuesta para el 90 por ciento de las peticiones se mantuvo muy por encima de los valores correctos esperados.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 180 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de éste conlleva a un mal funcionamiento del sistema.

Anexo 5: Caso de prueba Estrés: Mensajes Basura

Tipo de prueba: Prueba de Estrés

Funcionalidad (es)	Comentario
Mensajes Basura	El sistema muestra los datos de los mensajes indefinidos y los nombres de los mensajes que no cumplen con el formato UNEDIFACT PAXLST

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Resultados Generales:

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Mensajes basura	14927	82	63	0	3891	125	4.02	98.0/seg	1364.6

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Mensajes basura	11251	909	813	0	14828	1687	5.08	21.8/seg	360.3

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 70 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 4.02% y el 5.08%, valor que no cumple con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 14927 peticiones lo que equivale a 201 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 11251 peticiones, valor que equivale a 152 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable mostrando valores muy por debajo de los esperados y el tiempo de respuesta para el 90 por ciento de las peticiones se mantuvo por encima de los valores correctos esperados.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 152 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de éste conlleva a un funcionamiento inestable del sistema.

Anexo 6: Caso de prueba Estrés: Subir Mensaje

Tipo de prueba: Prueba de Estrés

Funcionalidad (es)	Comentario
Subir Mensaje	Mediante este caso de uso el contacto puede enviar el fichero que contiene el mensaje API, conectándose directamente al sitio.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Resultados Generales:

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Subir mensaje	11190	97	78	0	1234	172	6.07	93.2/seg	1792.6

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Subir mensaje	8770	1385	1218	16	17328	1938	7.24	18.7/seg	427.4

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 72 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 6.07% y el 7.24%, valor que no cumple con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 11190 peticiones lo que equivale a 228 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 8770 peticiones, valor que equivale a 178 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable y con un tiempo de respuesta superior al establecido.

Se puede llegar a la conclusión que el sistema para esta funcionalidad soporta 178 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de éste no se acepta.

Anexo 7: Caso de prueba Estrés: Vuelo Pasajeros

Tipo de prueba: Prueba de Estrés

Funcionalidad (es)	Comentario
Vuelo Pasajeros	El sistema muestra los vuelos y permite buscar información específica de ellos.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	5
Número de iteraciones: [Contador del bucle]	26

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	130
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	2
Número de iteraciones: [Contador del bucle]	2

Resultados Generales:

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Vuelo pasajeros	8724	108	93	0	2656	187	7.94	81.4/seg	2047.1

Total de usuarios: 260

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg

Vuelo pasajeros	7186	1554	1375	16	44250	2531	9.02	17.4/seg	501.3

Lista de links con errores:	**Comentario**
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 60 y el 65 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores varió entre el 7.94% y el 9.02%, valor que no cumple con los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el por ciento de error varía dependiendo del número de peticiones realizadas, se presenta el mismo problema con el elemento favicon.ico.

Se realizaron dos pruebas de estrés ambas con el objetivo de conectar 260 usuarios concurrentes, la primera se hizo con grupos de 10 usuarios que se conectan cada 5 segundos en un ciclo que se repetirá 26 veces, intentado lograr 260 usuarios en el último ciclo. Sin embargo al culminar la prueba se lograron 8724 peticiones lo que equivale a 235 usuarios concurrentes que logran concluir con éxito todas las peticiones. El sistema mantuvo un rendimiento estable y el tiempo de respuesta estuvo por debajo de 1 segundo, considerándose valores aceptables.

La segunda prueba se hizo con grupos de 130 usuarios concurrentes que se conectan cada 2 segundos, 2 veces, intentando lograr 260 usuarios concurrentes, al concluir la prueba se lograron 7186 peticiones, valor que equivale a 194 usuarios concurrentes que logran culminar con éxito todas las peticiones. El rendimiento del sistema no fue estable mostrando valores muy por debajo de los esperados y el tiempo de respuesta para el 90 por ciento de las peticiones se mantuvo muy por encima de los valores correctos esperados.

Podemos llegar a la conclusión que el sistema para esta funcionalidad soporta 194 usuarios concurrentes, presentando un buen rendimiento y un tiempo de respuesta dentro de los parámetros normales. Un valor por encima de éste conlleva a un mal funcionamiento del sistema.

Anexo 8: Caso de prueba Carga: Búsqueda Fonética

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Búsqueda Fonética	El sistema muestra a las personas que han viajado con anterioridad y sus datos coinciden fonéticamente con los datos especificados por el especialista de la aduana.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	25
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	50
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Resultados Generales:

Número de usuarios: 25

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Búsqueda Fonética	2425	369	187	0	11766	422	4.12	63.5/seg	939.0

Número de Usuarios: 50

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Búsqueda Fonética	4850	744	390	0	25485	813	4.12	63.3/seg	936.0

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 4.12 %, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que el sistema responde satisfactoriamente, pues para 50 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 25 usuarios concurrentes. El número de peticiones esperado era 2325 para 25 usuarios concurrentes y 4650 para 50 usuarios concurrentes, en ambos casos se realizaron 25 peticiones más, pues uno de los vínculos de error requirió ese número de peticiones extras.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Búsqueda Fonética cumple con los requerimientos establecidos.

Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Debe realizarse la verificación del código del que se muestra a continuación pues presenta un tiempo de respuesta inaceptable por encima de los valores esperados.

- /sua/web/sarpia_dev.php/internet/jsonReporteBusquedaFonetica

Aun así el tiempo general de respuesta de la funcionalidad es correcto pues el mismo está por debajo de 1 segundo en las peticiones.

Anexo 9: Caso de prueba Carga: Conformar Parte

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Conformar Parte	Obtener un resumen de la cantidad de vuelos que entraron al país por aeropuerto, conocer la cantidad de aerolíneas que viajaron a ese aeropuerto y el total de todos los vuelos que entraron al país.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	25
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	50
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones:	1

[Contador del bucle]	

Resultados Generales:

Número de usuarios: 25

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Conformar Parte	2225	386	219	0	6703	657	3.37	60.6/seg	883.2

Número de Usuarios: 50

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Conformar Parte	4450	805	500	0	10500	1203	3.37	57.8/seg	842.0

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 3.37%, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que le sistema responde satisfactoriamente, pues para 50 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 25 usuarios concurrentes. El número de peticiones esperado era 2125 para 25 usuarios concurrentes y 4250 para 50 usuarios concurrentes.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Conformar Parte cumple con los requerimientos establecidos.
Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Anexo 10: Caso de Prueba Carga: Conformar Mensaje

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Conformar mensaje	Se recogen los datos del vuelo, pasajeros y tripulantes necesarios para conformar el mensaje API. El sistema posibilita guardar el mensaje en la PC de actor, el mensaje será generado con el formato un_edifact.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	5
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Resultados Generales:

Número de usuarios: 5

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Conformar mensaje	560	44	16	0	3219	93	2.68	91.7/seg	957.9

Número de Usuarios: 10

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Conformar mensaje	1120	92	47	0	1187	156	2.68	103.0/seg	936.0

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 2.68 %, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que le sistema responde satisfactoriamente, siendo esta funcionalidad una de las que realiza muy pocas peticiones debido al poco uso que se le da por parte de la Aduana, para 10 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 5 usuarios concurrentes. El número de peticiones esperado era 550 para 5 usuarios concurrentes y 1110 para 10 usuarios concurrentes.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Conformar mensaje cumple con los requerimientos establecidos.

Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Anexo 11: Caso de Prueba Carga: Estado de Mensajes

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Estado de mensajes	En este caso de uso el contacto puede consultar el estado de los mensajes enviados por la aerolínea a la que pertenece. La información puede ser filtrada por fecha de llegada del mensaje, vía de llegada y estado.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	50
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	25
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Resultados Generales:

Número de usuarios: 25

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Estado de mensajes	2350	206	156	0	2453	343	3.19	114.2/seg	1321.9

Número de Usuarios: 50

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Estado de mensajes	4700	497	437	0	3765	765	3.19	94.3/seg	1091.1

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación
/sua/web/sarpia_dev.php/internet/viaRecepcion	Error de recepción
/sua/web/sarpia_dev.php/internet/jsonReporteEstadoMensajes	Error de estado de mensaje

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 3.19 %, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por cuatro elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que le sistema responde satisfactoriamente, pues para 50 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 25 usuarios concurrentes. El número de peticiones esperado era 2325 para 25 usuarios concurrentes y 4650 para 50 usuarios concurrentes.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Estado de mensajes cumple con los requerimientos establecidos.

Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Anexo 12: Caso de Prueba Carga: Mensajes Basura

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Mensajes Basura	El sistema muestra los datos de los mensajes indefinidos y los nombres de los mensajes que no cumplen con el formato UNEDIFACT PAXLST

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	50
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	25
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Resultados Generales:

Número de usuarios: 25

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Mensajes Basura	2125	181	140	0	2859	282	4.71	125.0/seg	1572.2

Número de Usuarios: 50

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Mensajes Basura	4250	486	422	0	4219	734	4.71	95.7/seg	1203.7

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 4.71 %, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que le sistema responde satisfactoriamente, pues para 50 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 25 usuarios concurrentes. El número de peticiones esperado era 2050 para 25 usuarios concurrentes y 4100 para 50 usuarios concurrentes.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Mensajes Basura cumple con los requerimientos establecidos.

Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Anexo 13: Caso de Prueba Carga: Subir Mensaje

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Subir Mensaje	Mediante este caso de uso el contacto puede enviar el fichero que contiene el mensaje API, conectándose directamente al sitio.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	5
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	10
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Resultados Generales:

Número de usuarios: 5

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Subir Mensaje	465	29	16	0	2969	32	6.45	148.8/seg	1280.8

Número de Usuarios: 10

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Subir Mensaje	930	62	47	0	797	109	6.45	157.2/seg	1725.5

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación
/sua/web/sarpia_dev.php/seleccion/subirImagen	Error de imagen

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 6.45 %, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por tres elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que le sistema responde satisfactoriamente, siendo esta funcionalidad una de las que realiza muy pocas peticiones debido al poco uso que se le da por parte de la Aduana, para 10 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 5 usuarios concurrentes. El número de peticiones esperado era 450 para 5 usuarios concurrentes y 890 para 10 usuarios concurrentes.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Subir Mensaje cumple con los requerimientos establecidos.
Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Anexo 14: Caso de Prueba Carga: Vuelos Pasajeros

Nombre del Proyecto: Sistema Único de Aduanas.

Nombre Módulo: Información Adelantada de Pasajeros.

Versión: 1.0

Tipo de prueba: Prueba de Carga

Funcionalidades a probar:

Funcionalidad (es)	Comentario
Vuelos Pasajeros	El sistema muestra los vuelos y permite buscar información específica de los vuelos.

Variables:

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	25
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	50
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	0
Número de iteraciones: [Contador del bucle]	1

Resultados Generales:

Número de usuarios: 25

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Vuelos Pasajeros	2475	298	188	0	3610	422	3.03	83.7/seg	938.4

Número de Usuarios: 50

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg
Vuelos Pasajeros	4950	555	375	0	7687	657	3.03	78.1/seg	1005.5

Lista de links con errores:	Comentario
/sua/web/js/ext/locale/ext-lang-es.js	Error en extensión de Java Script
/favicon.ico	Error de navegación

Durante la ejecución de la prueba el CPU se mantuvo entre el 50 y el 60 % de utilización, la memoria física estuvo por debajo del 75 % y la taza de errores se mantuvo al 3.03%, valor que no entra dentro de los parámetros establecidos, si se tiene en cuenta que el error es producido por dos elementos que no se cargan durante las peticiones a la aplicación, el elemento favicon.ico, que representa un icono que traen por defecto todos los navegadores solo da error si la prueba se realiza con el navegador Firefox, en caso de usar el Internet Explorer este error no aparece.

Analizando los valores de cantidad de peticiones, y rendimiento se puede concluir que le sistema responde satisfactoriamente, pues para 50 peticiones concurrentes mantuvo un rendimiento estable en comparación a la respuesta del sistema con 25 usuarios concurrentes. El número de peticiones esperado era 2375 para 25 usuarios concurrentes y 4750 para 50 usuarios concurrentes.

Luego del análisis de los valores, realizado anteriormente se puede llegar a la conclusión de que la funcionalidad Vuelos Pasajeros cumple con los requerimientos establecidos.

Aun así se especifica que debe realizarse la revisión de los vínculos que dan error para su corrección.

Anexo 15: Plantilla para Casos de Prueba de rendimiento usando la herramienta JMeter:

Nombre del Proyecto:<Proyecto>

Nombre Módulo:<Módulo>

Versión:<Versión del caso de prueba>

Fecha	Versión	Descripción	Autor
[00/00/0000]	[V 1.00]	[Descripción del plan de Pruebas]	[Nombre del Autor]

Tipo de Prueba (Carga o Estrés):

[Se hace necesario antes de la realización de las pruebas, la revisión y análisis del manual de uso del JMeter para efectuar pruebas de rendimiento a Sistema Único de Aduana]

Prueba de Carga:

[En caso de que la prueba a realizar sea Prueba de Carga]

La prueba de carga tiene como objetivo comprobar la respuesta del sistema una vez que se interactúe con él y se simule el trabajo de una cantidad esperada de usuarios que introduzcan datos y naveguen por el sitio, comprobando si el sistema es capaz de responder correctamente bajo condiciones extremas. En esta prueba se introducen datos en el sistema para ver cómo responde la Base de datos ante las peticiones de información.

Nombre de la Funcionalidad o funcionalidades a probar:

[Funcionalidades que serán probadas con la introducción de datos]

Funcionalidad(es)	**Comentario**
Nombre de la funcionalidad	[Detalles de lo que realiza esa funcionalidad]

Variables	**Valores**
Número de usuarios concurrentes: [Número de hilos]	
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	[cero por defecto, pues todas las conexiones se realizan de forma simultánea]
Numero de iteraciones: [Contador del bucle]	[Tiene como valor 1 por defecto, pues la iteración se realiza una sola vez.]

Prueba de Estrés:

[En caso de que la prueba a realizar sea de Estrés]

La prueba de estrés tiene como objetivo comprobar cómo responde el sistema bajo una serie de peticiones que se incrementarán con el paso del tiempo al doble, hasta llegar a un valor extremo. En este

tipo de prueba solo se comprueba la conexión a las interfaces, pero sin el intercambio de información, por lo que no se efectúa la entrada de datos al sistema.

Variables	Valores
Número de usuarios concurrentes: [Número de hilos]	
Tiempo entre conexión y conexión: [Periodo de subida (en segundos)]	[Tiempo que ocurrirá entre la conexión del los usuarios de la primera iteración y todas las demás]
Número de iteraciones: [Contador del bucle]	[Cantidad de veces que se repetirá la simulación de conexión de usuarios]

Funcionalidades	Comentario
[Nombre de la funcionalidad a la que se accederá sin incluir ni extraer información, solo se efectúa la conexión a la interfaz de la misma.]	[Detalles de lo que realiza esa funcionalidad]

Resultados generales:

Petición	Muestras	Media	Mediana	Min	Max	Línea 90 %	%Error	Rendimiento	Kb/seg

[Se toma la última línea que muestra la tabla del Reporte Agregado dentro del JMeter]

[Petición: Nombre del módulo]

[Los demás datos de la tabla se encuentran en el resultado de la prueba efectuada con el Software JMeter]

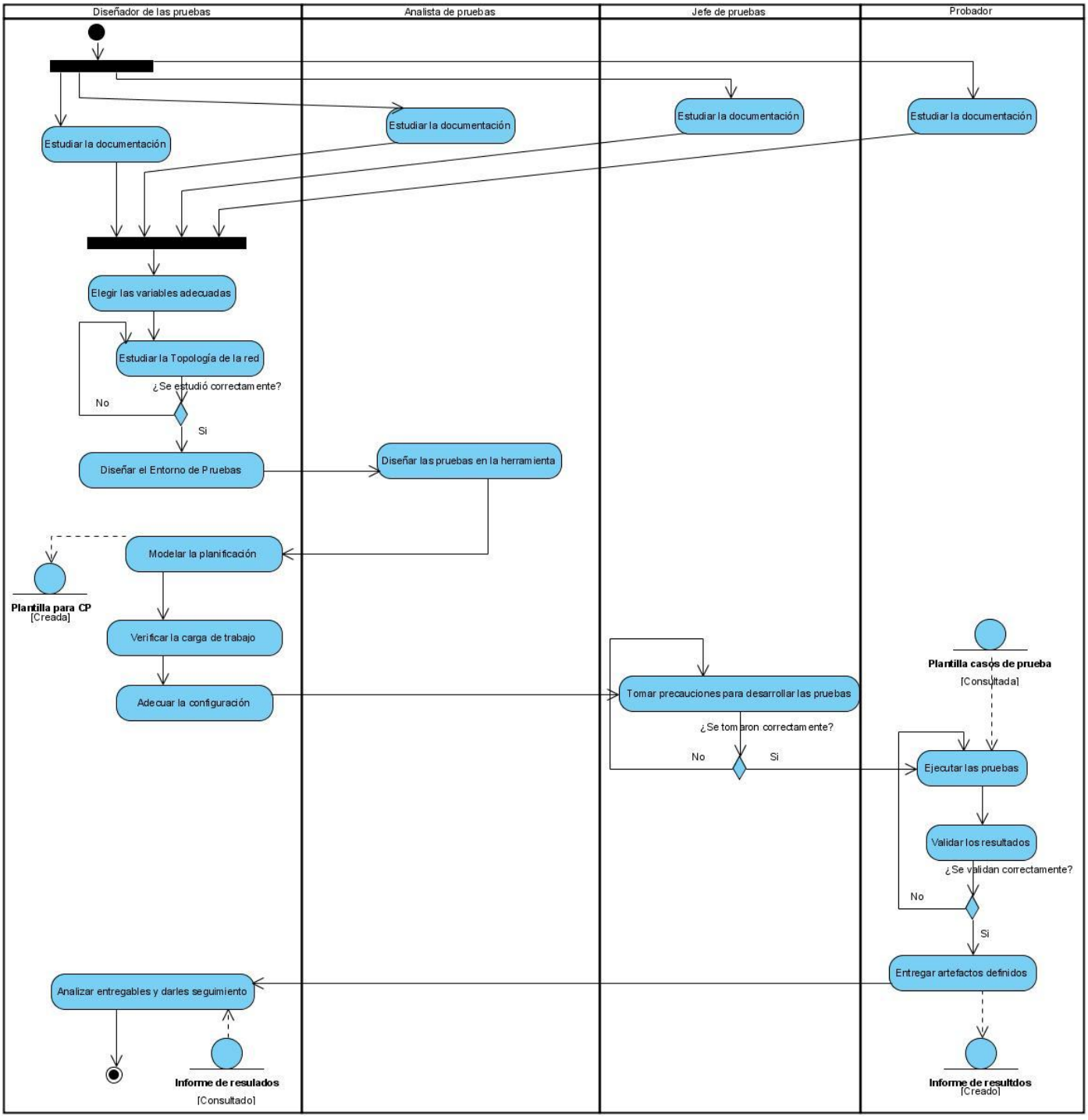
Lista de links con errores (En caso de existir):	Comentario

[Elementos que dieron error en el reporte agregado]

Conclusiones:

[Las conclusiones se dan en base al rendimiento, la cantidad de peticiones y los errores obtenidos, comparando éstos con los resultados esperados]

Anexo 16: Diagrama de actividades del procedimiento por roles.



Anexo 17: Manual de usuario

Introducción:

Este manual tiene como objetivo ayudar y aclarar posibles dudas a la hora de utilizar JMeter como herramienta para la realización de pruebas de Estrés y Carga en Aduana. Para esto se ha realizado un estudio exhaustivo de los pasos que se llevaran a cabo y de las posibles dudas que puedan surgir en el transcurso de la realización de las pruebas.

En conclusión este tutorial brinda de manera asequible y sencilla a los trabajadores implicados en la realización de pruebas, un amplio conocimiento de la herramienta para que puedan mejorar sus procesos y obtener mejores resultados.

Requerimientos:

Para poner en funcionamiento la herramienta JMeter, se debe instalar la Máquina Virtual de Java versión 1.3 o mayor, de lo contrario la aplicación no funcionará.

La planificación y diseño de las pruebas se pueden realizar de diferentes formas en dependencia de la importancia del sistema que se desea probar y del tipo de prueba que se quiera realizar. En una aplicación web donde solo se modele la navegación de los usuarios se pueden realizar una planificación de pruebas simple 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.

A continuación se exponen la forma de configuración para pruebas complejas, aclarando que para las pruebas a los sistemas de Aduana estas son las que se utilizan.

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)

Para una mayor profundización en cuanto a las características y utilidades de cada uno de estos elementos se puede utilizar uno de los tutoriales en español.

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 el realismo que se quiere 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** nos permite decidir si realizar o no una petición http en función de una condición. Cada controlador, puede tener uno o más elementos por defecto.

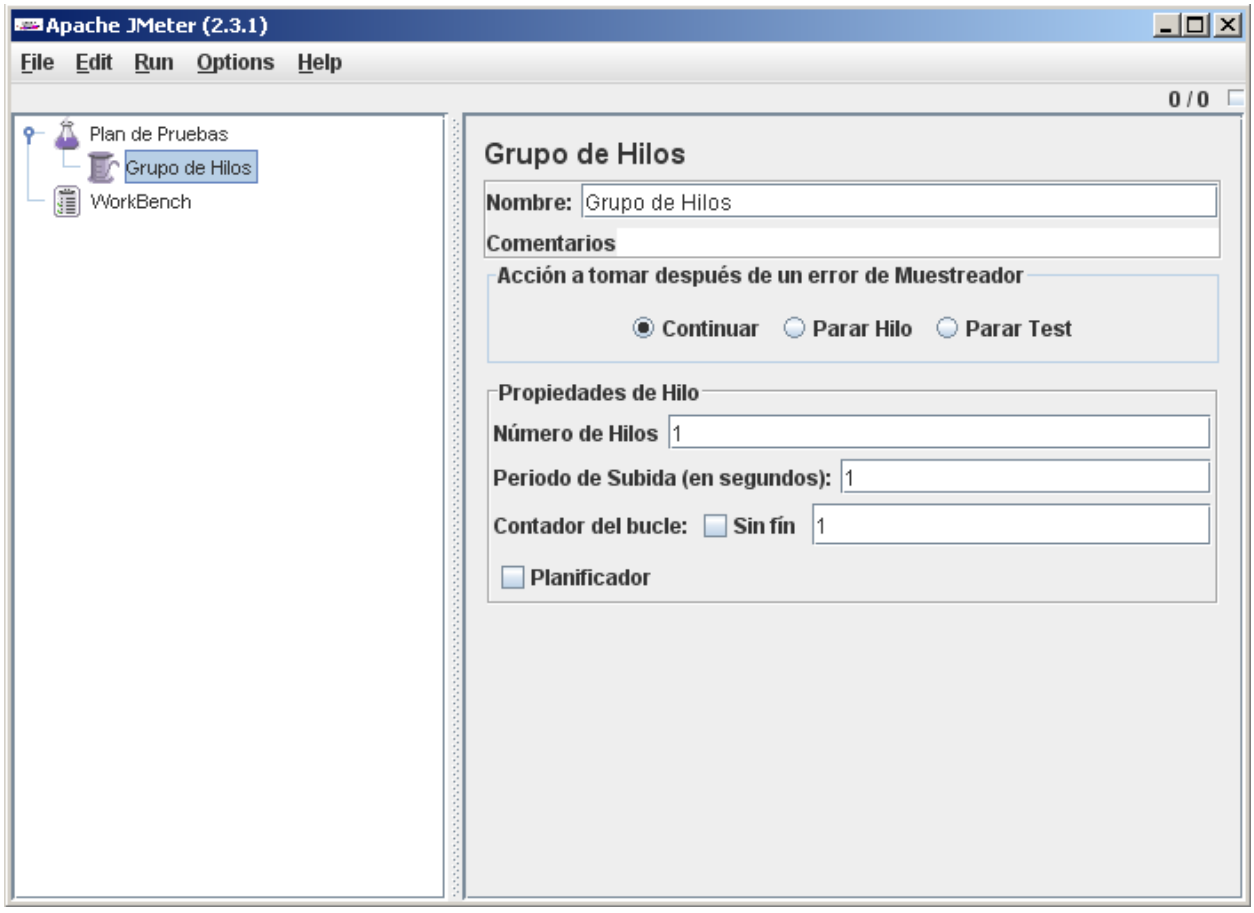


Fig. 1 Grupo de Hilos

Utilizando el elemento **Grupo de Hilos (**Fig. 1), 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 20 hilos y se le agrega al contador de bucles 4, se realiza un total de 80 muestras, ya que la prueba se ejecutará 80 veces, a razón de 20 usuarios por vez.

Otro de los sub-elementos que contiene el elemento **Grupo de Hilos** es el Planificador. 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:

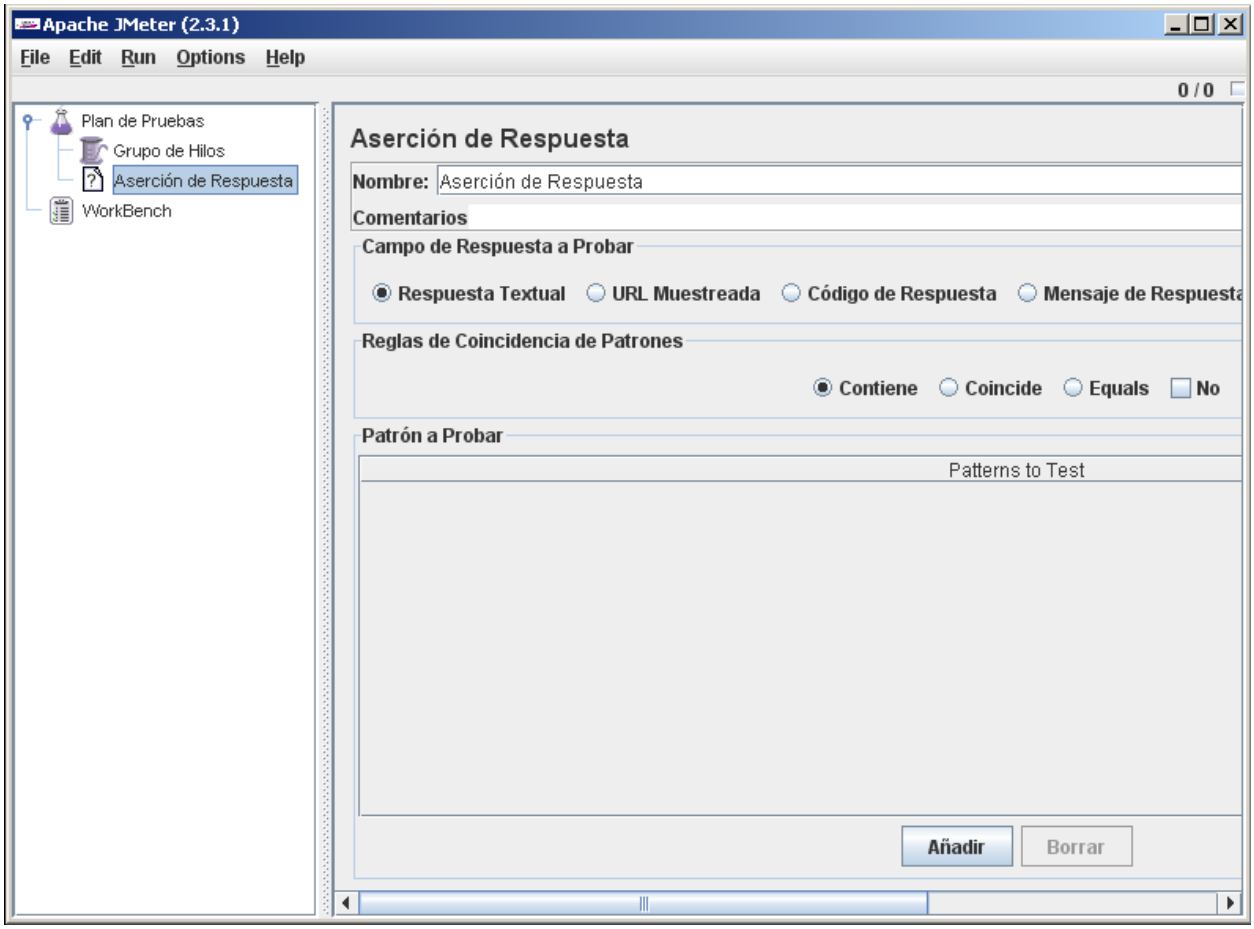
- Tiempo de Arranque: Fecha y Hora en la que se desea iniciar la prueba.
- Tiempo de Finalización: Fecha y Hora en la que se desea Finalizar la prueba.
- Duración (segundos): Duración de cada uno de los Hilos.

Retardo de Arranque (segundos): Tiempo de retardo en el arranque de cada uno de los hilos.

Otro elemento importante una vez que se tiene creado el grupo de Hilos, es la Aserción de respuesta (Fig. 2) pues en este se especificará el patrón que se necesita validar en la prueba.

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 (utiliza Tiny).
- Aserción MD5Hex, para verificar que la suma MD5 es la especificada.
- Aserción de Tamaño, para verificar que el tamaño es <,>, =, etc. que uno especificado.
- Aserción XML, para verificar que el resultado es un XML bien formado
- Aserción Beanshell, para programación de pequeños shell scripts que realizan verificaciones a medida.



También se pueden añadir Preprocesadores que realicen acciones antes de enviar la Petición:

- Contador: Para crear una variable contador, que puede ser referenciada en cualquier parte del test.
- Parámetros de Usuario: parámetros definidos por el Usuario, que nos van a permitir definir una especie de constantes dentro del test.
- Parser de Enlace HTML: Parsea la respuesta HTML del servidor, y extrae los enlaces y los formularios.
- Mascara de Parámetros HTML.
- Modificador de Re-escritura HTTP URL.
- Modificador de Parámetro de Usuario HTTP

También se pueden añadir Pos procesadores que realicen acciones después de enviar la Petición:

Extractor de Expresiones Regulares: Extrae cadenas de la respuesta (contenido o cabeceras) que cumplan una expresión regular.

Manejador de Acciones para Status de Resultados: Permite indica la acción a tomar después de que se produzca un error: continuar, parar el thread, o parar el test.

Guardar Respuestas en Archivo: Permite almacenar en un fichero la respuesta obtenida (todas o sólo las erróneas).

Generar Sumario de Resultados: Resumen de información que se envía cada cierto tiempo (utilizado en modo batch).

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** (Fig. 3) y **Ver Árbol de Resultados** (Fig. 4).

En el informe agregado se muestran una serie de datos los cuales exponen el estado en el que se encuentra en 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 tiempo total que demoraron las petición en cargarse.
- Mediana: Tiempo promedio que han tardado en cargarse las paginas.
- Min: Tiempo mínimo que ha demorado en cargarse una página.
- Max: Tiempo Máximo que ha tardado en cargarse una página.
- Línea 90 %: Tiempo máximo en que corrieron el 90 por ciento de las peticiones reales, o sea, el tiempo más probable que se puede demorar una petición.
- %Error: Por ciento de error de las paginas que no se llegaron a cargar de manera satisfactoria.
- Kb/Seg: Velocidad de carga de las paginas.
- Tiempos de Respuestas: Total del tiempo que demoró en cargarse la cantidad de hilos de esa prueba

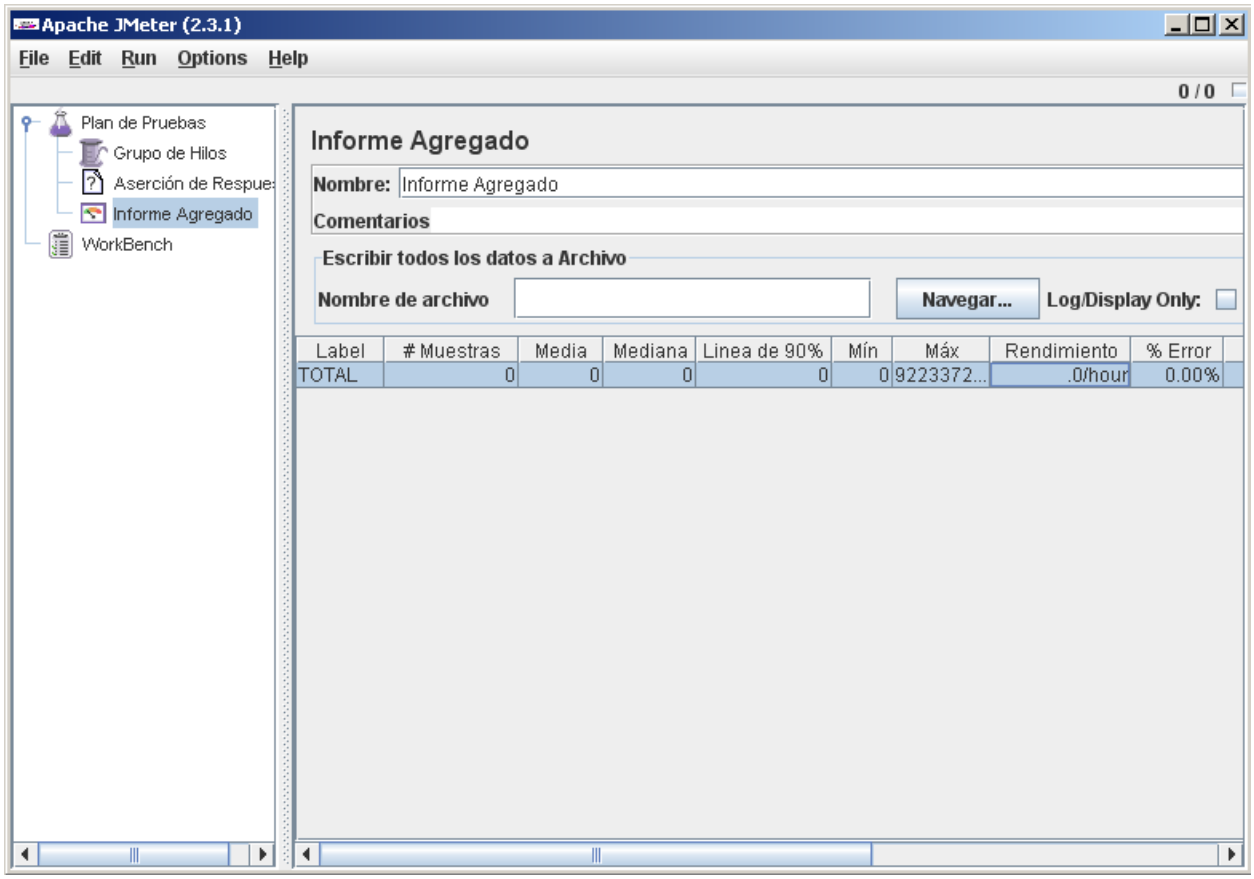


Fig.3 Informe Agregado

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 dicha 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, eso está dado por la definición de las aserciones. O sea, que si las aserciones se definen mal o no se definen, puede provocarse que se consideren validas pruebas que realmente no lo son.

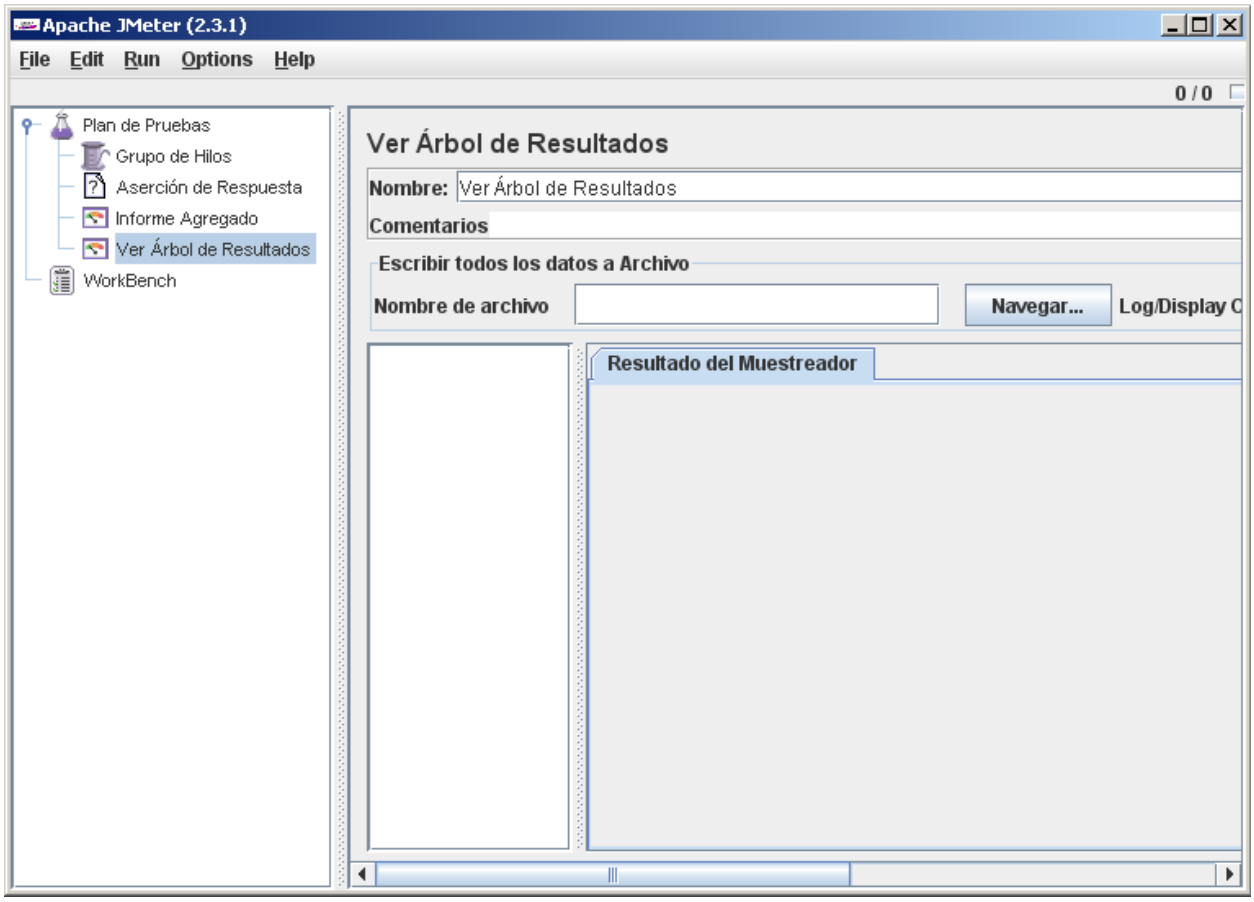


Fig. 4 Ver Árbol de Resultados

Elaboración de escenarios de prueba complejos:

Para la realización de pruebas de carga y estrés en un software de gestión se realiza la configuración de escenarios complejos de prueba, en esta ocasión se realizarán diferentes pasos y se agregaran nuevos elementos los cuales permitirán entrar datos al sistema y enviarlos a la Base de Datos del mismo. Esta nueva acción se realiza para lograr una mayor similitud con los escenarios reales de navegación en el sistema.

Para obtener una muestra fiel de los elementos de interacción del sistema a probar se realiza la grabación de sus escenarios. Esta opción se realiza a través del elemento **Servidor Proxy HTTP.** (Fig.5)

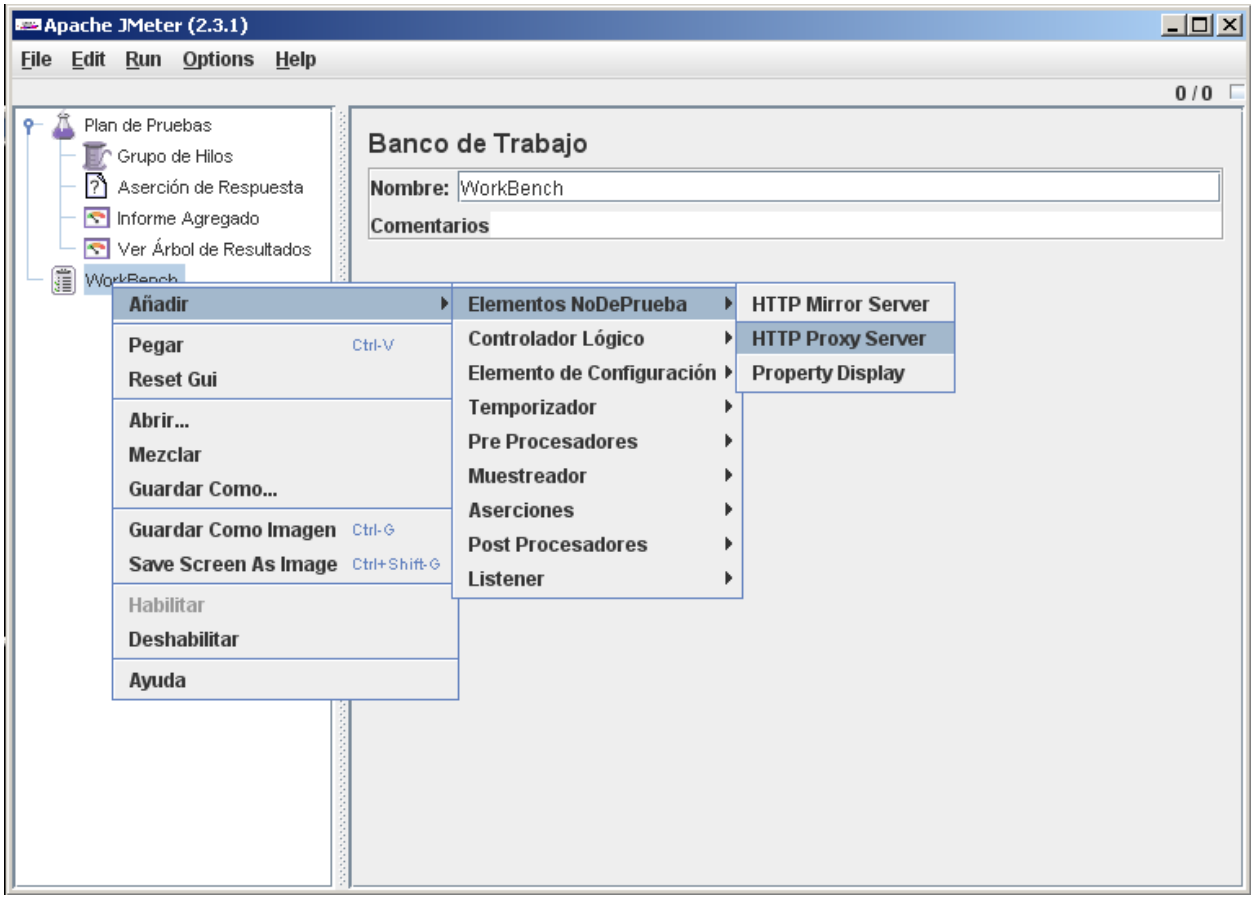


Fig. 5 Servidor Proxy HTTP.

Este elemento permite al JMeter grabar todos los pasos realizados en el software de manera que se generarán los escenarios de prueba de manera automática.

Se especifica un puerto específico para el **Servidor Proxy HTTP**. Luego se le especifica en qué manera se quiere organizar los resultados de la grabación y se presiona el botón **Arrancar.** (Fig. 6)

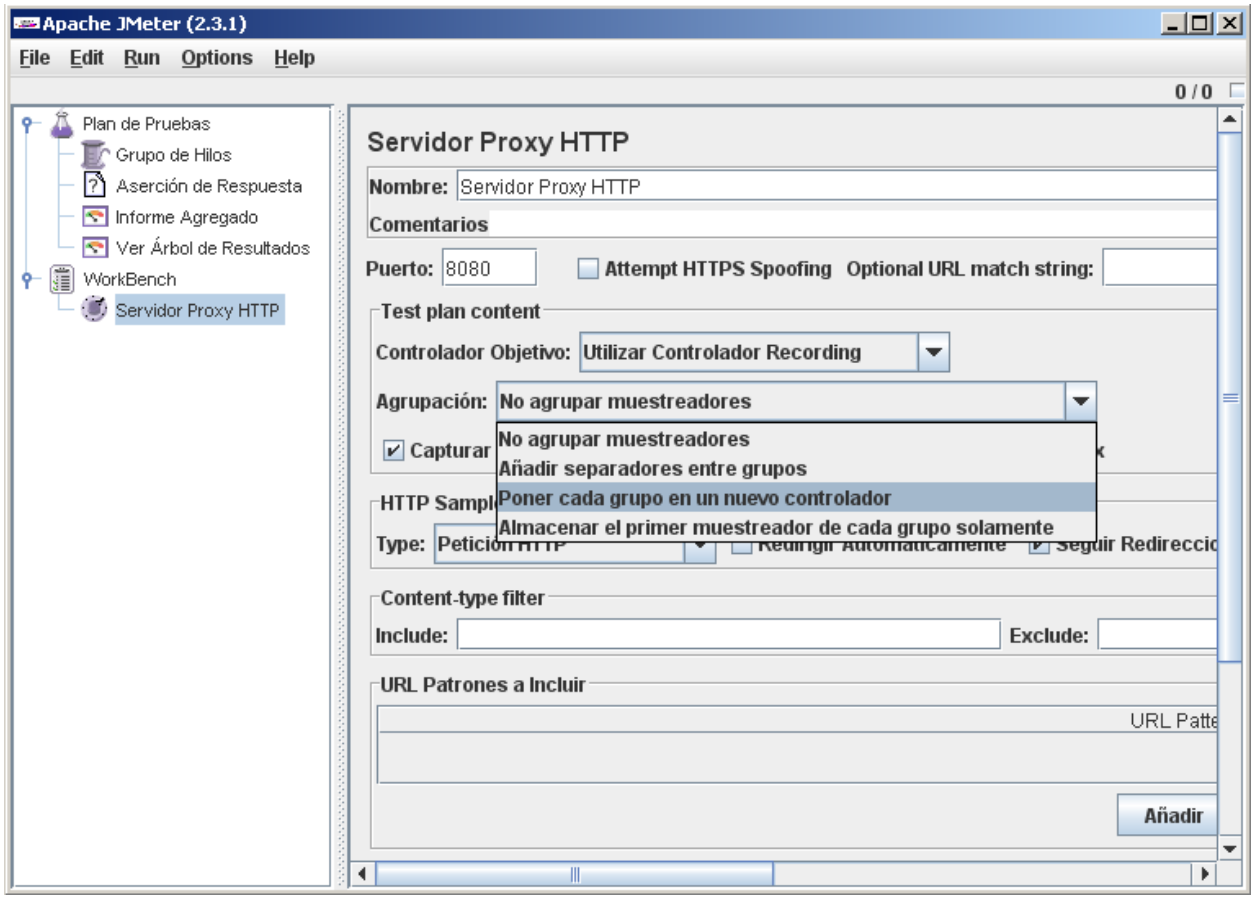


Fig.6 Configuración del Servidor

Seguidamente se configura el navegador con los datos de la herramienta para su correcto funcionamiento. Pasándole a este las opciones de proxy y servidor especificadas anteriormente en la configuración del JMeter.

(Fig. 7)

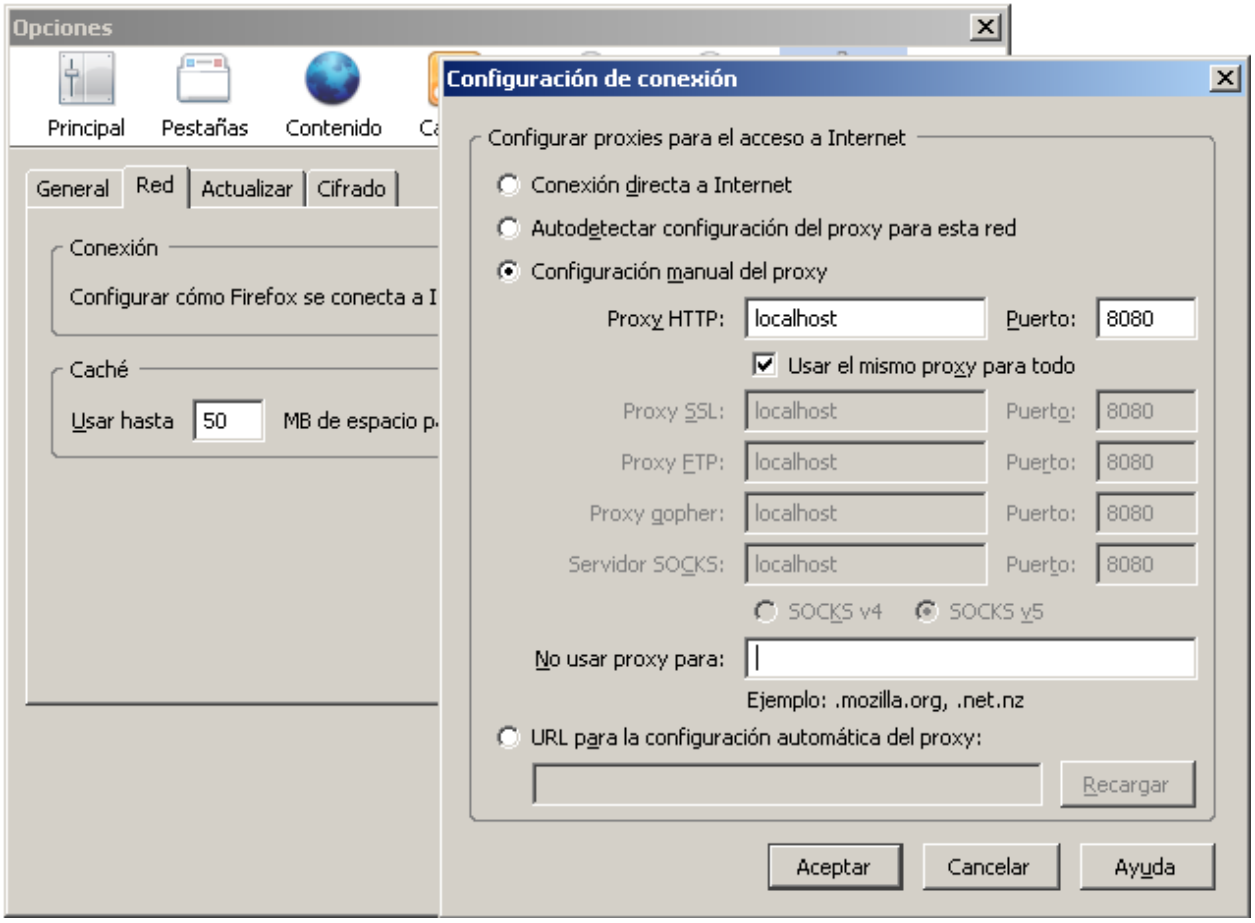


Fig. 7 Configuración de las opciones de internet

Luego se agrega el **Grupo de Hilos** al plan de pruebas y se ejecuta la aplicación por medio del Internet Explorer o el Mozilla Firefox, en ambos casos los valores en la configuración son los mismos, una vez ejecutada la aplicación se realizan todas las acciones que el usuario desee que se graben y se simulen después, una vez terminadas las acciones se regresa a la herramienta JMeter y se detiene la grabación.
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. (Fig. 8)

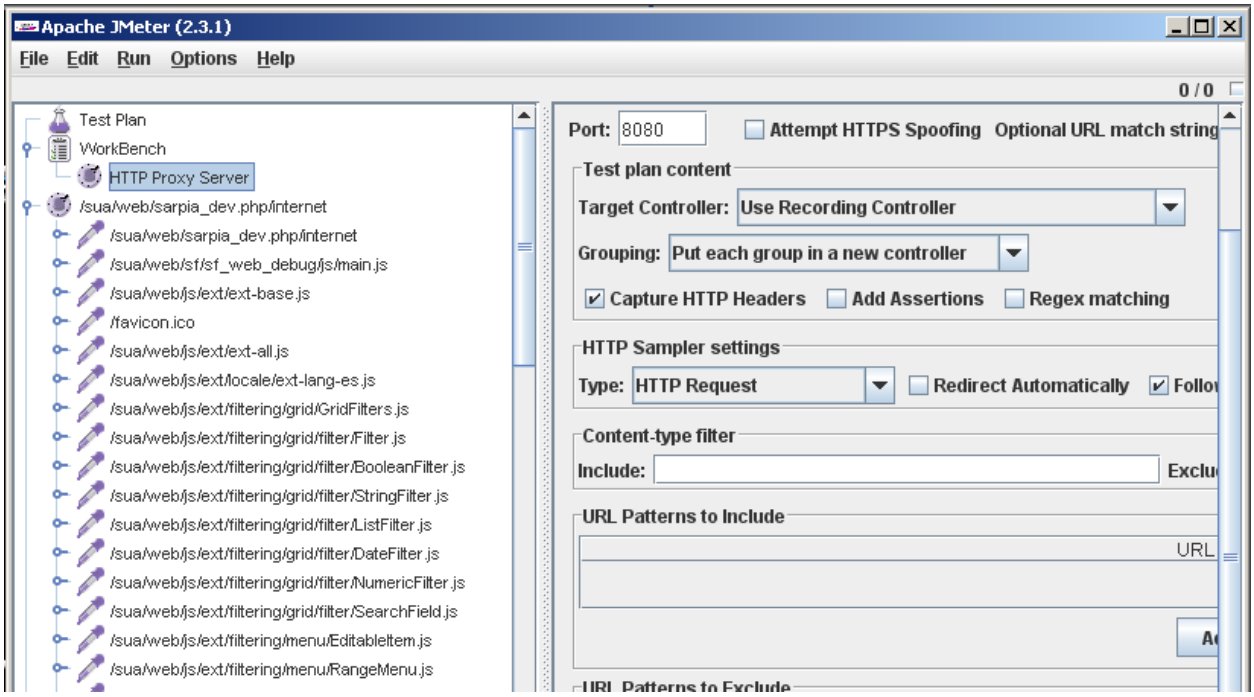


Fig. 8 Grabación de navegación por el sitio

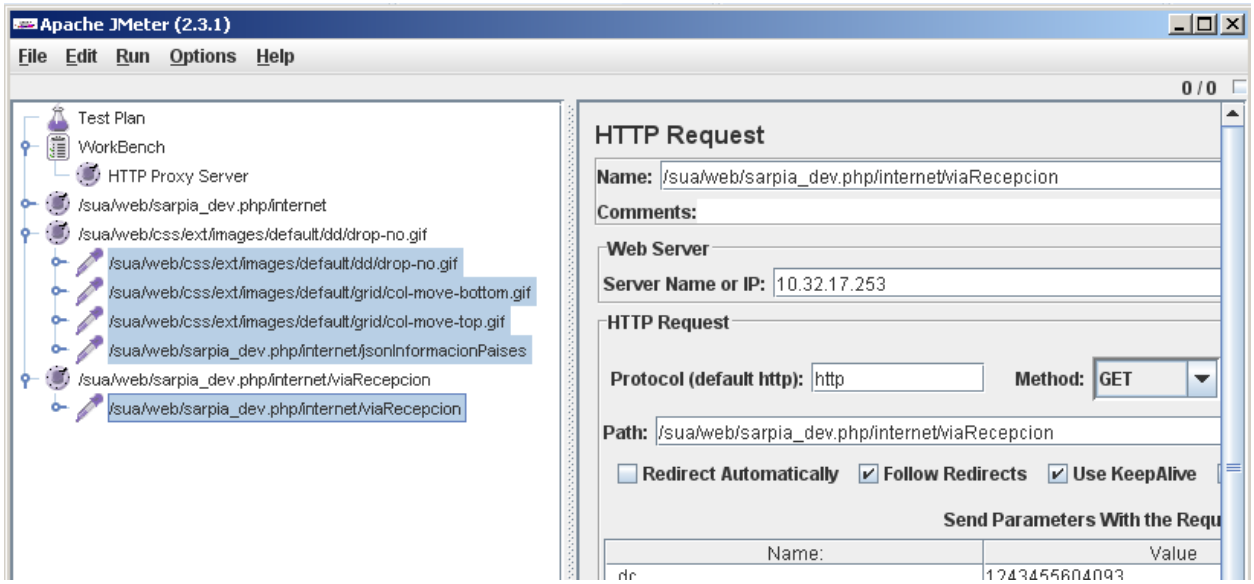


Fig. 9 Ejemplo de petición de HTTP de la grabación.

Como se puede comprobar a través del ejemplo anterior todos los datos que han sido insertados en el sistema se muestran en los parámetros enviados con la petición lo cual nos muestra información acerca de los componentes que los reciben, como el nombre, el tipo del valor y si necesita ser codificado.

Para lograr una mayor organización, es recomendable realizar un plan de prueba para cada una de las pruebas que queremos realizar, ya sea por casos de uso o por caminos lógicos.

Por cada **Petición HTTP** se genera en este caso un **Gestor de Cabecera HTTP**, los cuales se eliminan ya que al realizar una nueva llamada desde el JMeter interfieren en el éxito de las respuestas del servidor Web. (Fig. 10)

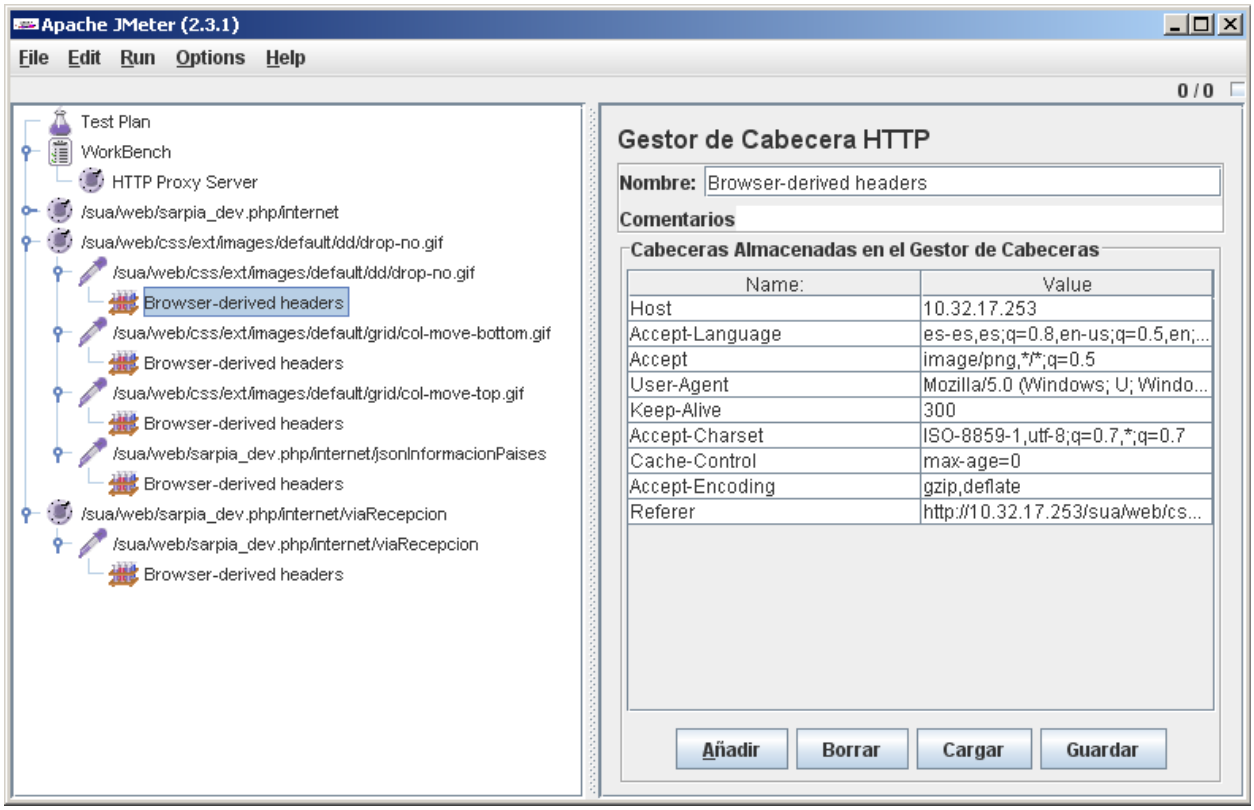


Fig. 10 Ejemplo de Gestor de Cabecera HTTP generado en la grabación.

Una vez realizados todos los pasos anteriores se procede a la inserción de los elementos que se necesitarán para la obtención de los resultados una vez que se realicen las pruebas.

Estos serian:

1. Grupo de Hilos
2. Ver Árbol de resultados
3. Informe agregado

4. Aserción de respuesta

Grupo de Hilos

Para agregar un grupo de hilos se da clic derecho sobre el plan de pruebas y se selecciona la primera opción grupo de hilos. (Fig. 11)

Una vez se tiene el grupo de hilos se procede a la llena de datos, estos serian la cantidad de usuarios que se desean simular, el tiempo en el que irán conectándose y la cantidad de veces que lo estarán haciendo.

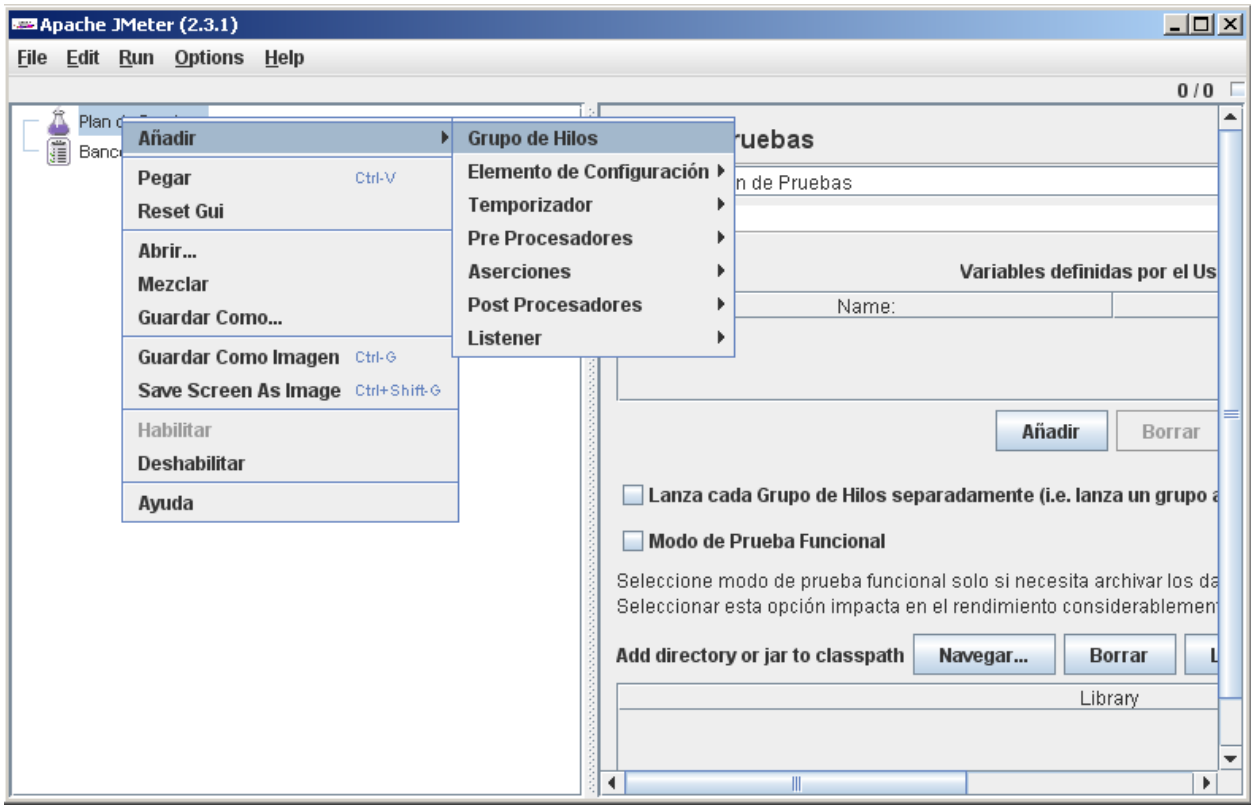


Fig. 11 Como añadir un grupo de hilos.

Luego de agregar el **Grupo de hilos** para la realización de la prueba, se procede al traslado de todas las peticiones grabadas como hijos de este grupo de hilos, esto se realiza dando clic izquierdo sostenido sobre la petición y arrastrándola hacia el grupo de hilos, una vez allí se suelta el clic y aparecen las opciones que se muestran en la figura 11, de estas tres opciones se selecciona la tercera opción **Añadir como hijo**.

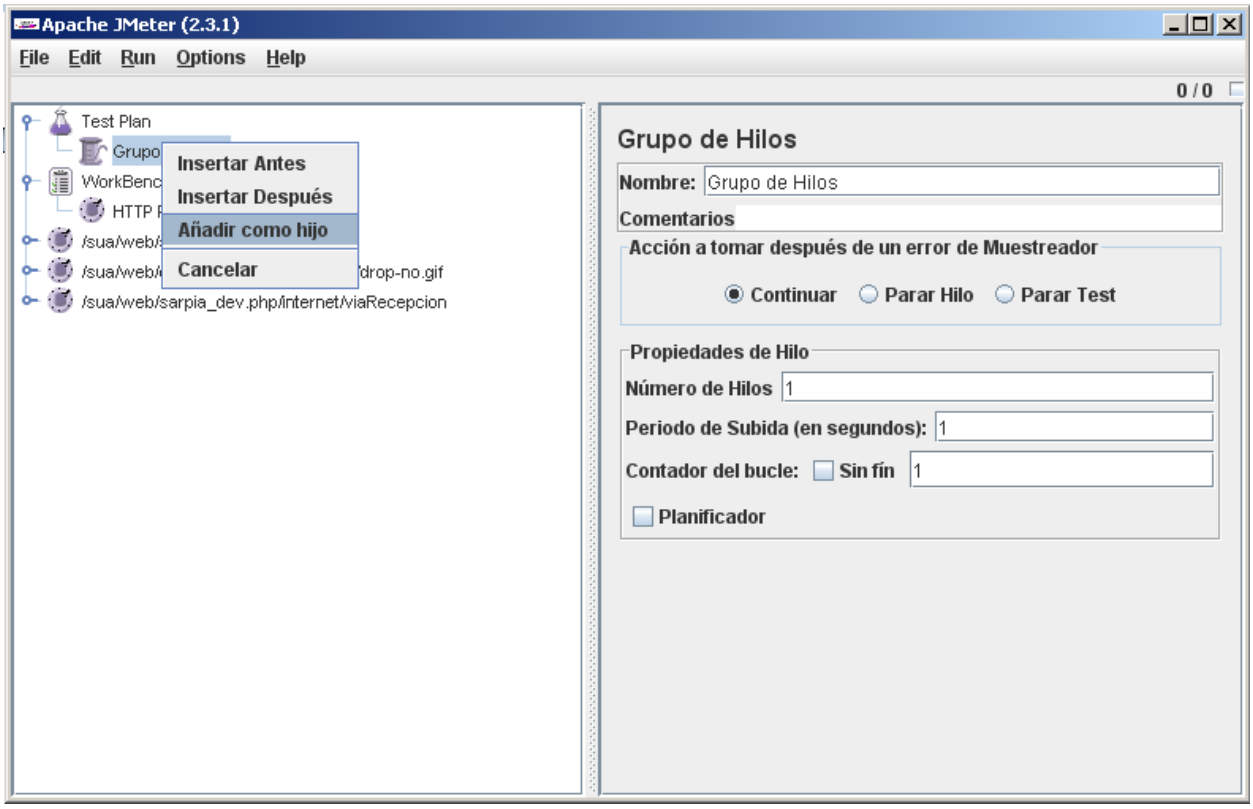


Fig. 12 Traslado de peticiones.

Ver Árbol de Resultados

Para añadir un árbol de resultados se da clic derecho sobre el plan de pruebas se va luego a los listeners y dentro de éstos se escoge la última opción. (Fig. 12)

El árbol de resultados permite apreciar en detalle la información que ha sido cargada. Mostrando el código que ha dado la página, el texto y otros atributos, además de que se pueden apreciar que peticiones han dado error y de qué tipo son los mismos.

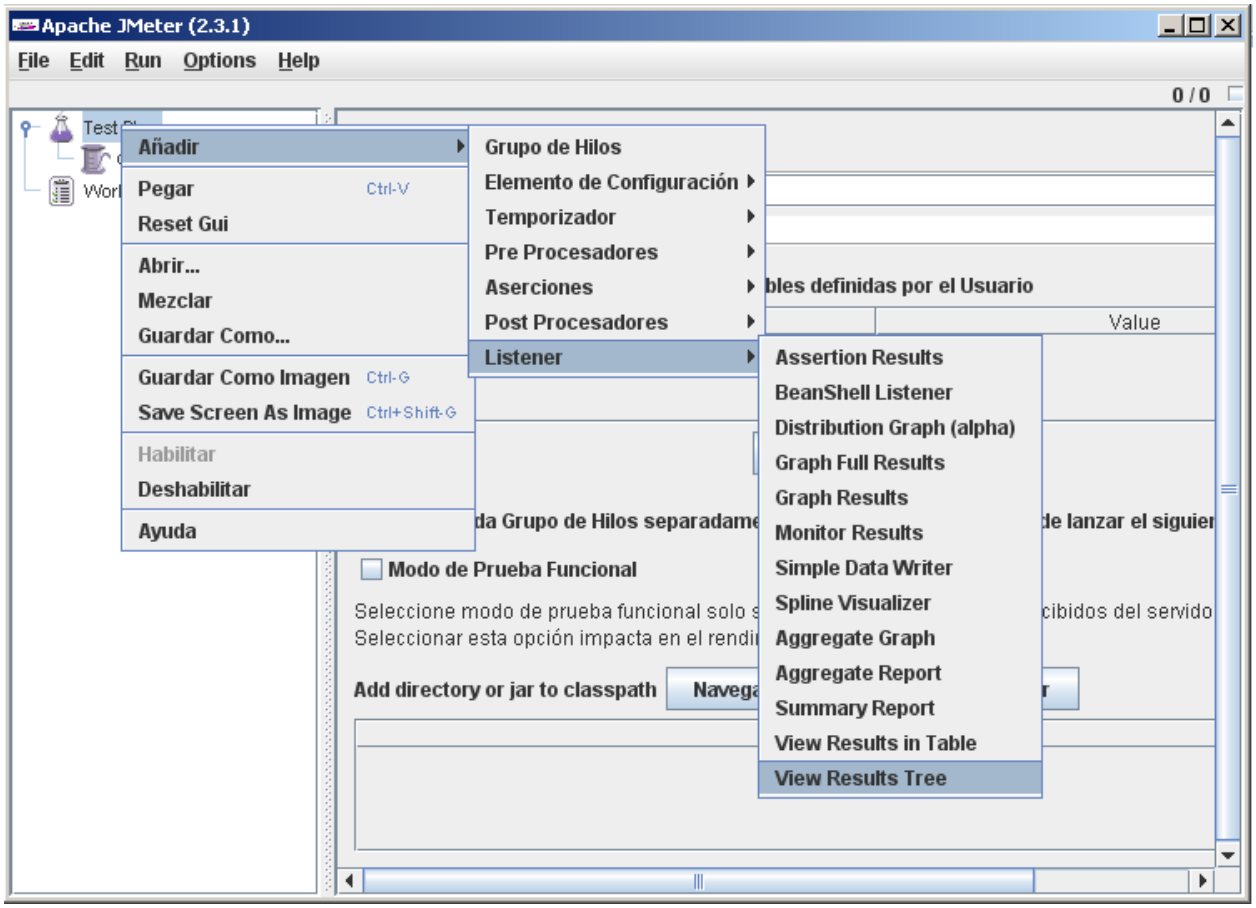


Fig. 12 Selección de la opción Árbol de resultados

Informe Agregado

El informe agregado permite obtener una serie de valores que se interpretan para dar los resultados. Es por esto que se recomienda el uso del mismo.

A continuación se muestra la imagen con los detalles para la selección del Informe Agregado.

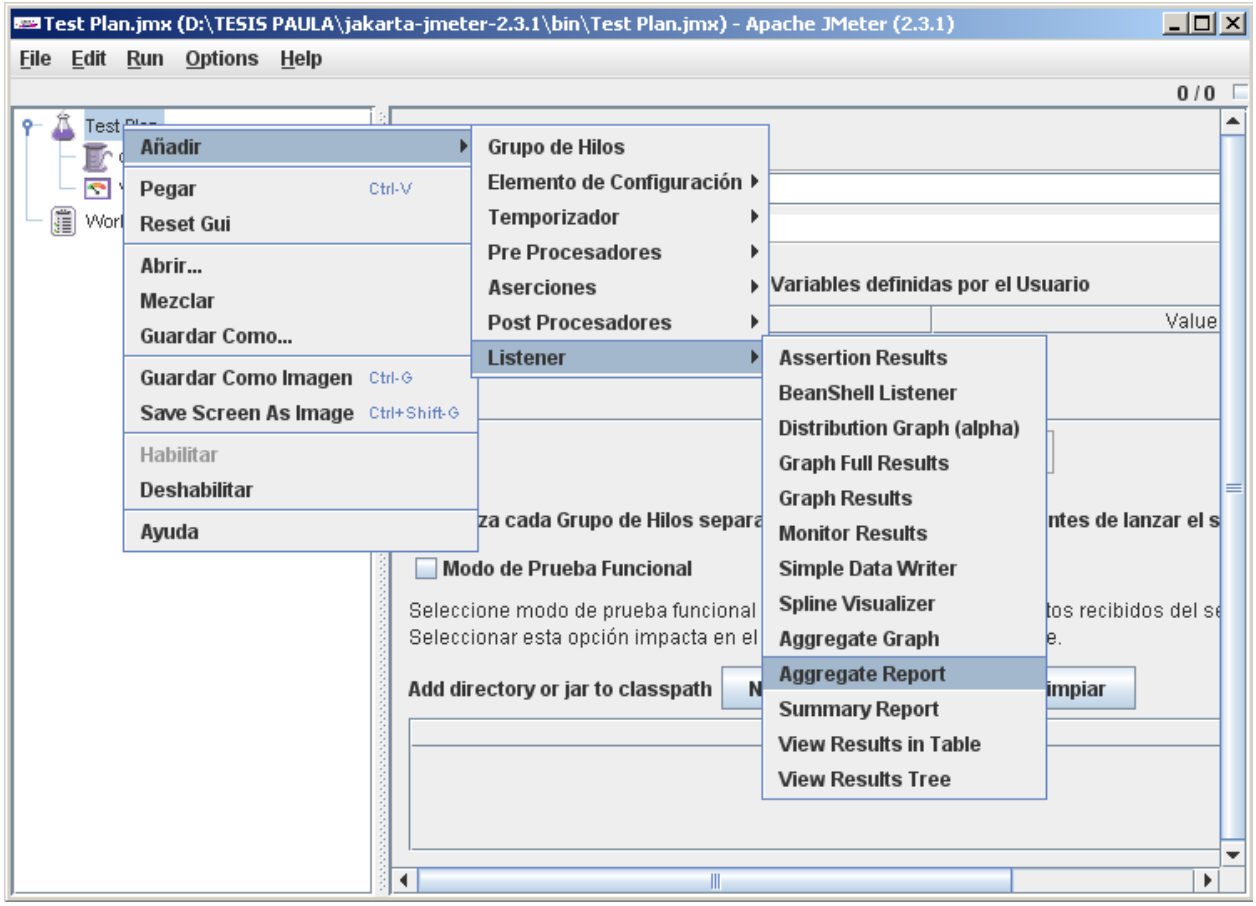


Fig. 13 Selección del Informe Agregado

Aserción de respuesta

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

A continuación se muestra la imagen con los detalles para agregar la Aserción de Respuesta.

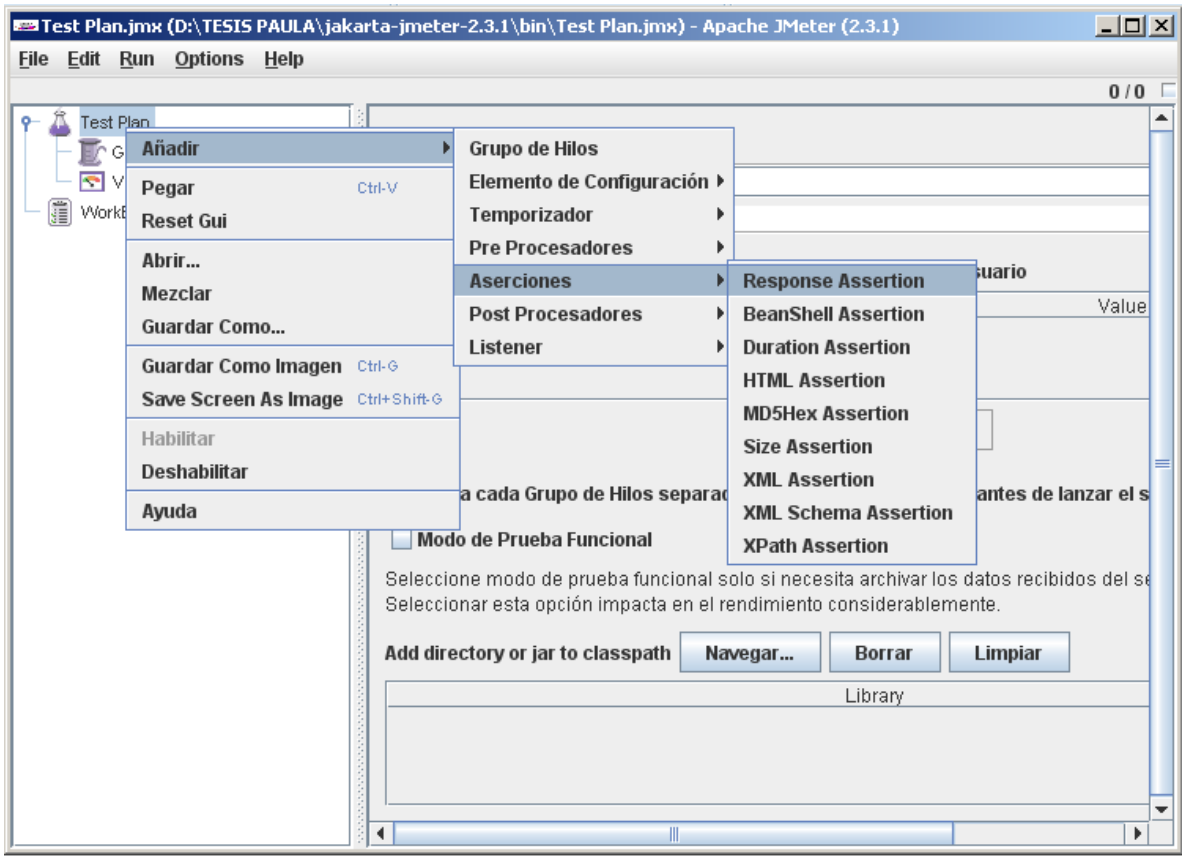


Fig. 14 Selección de Aserción de Respuesta

Para este caso solo se necesita saber 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 le pasamos en el test.

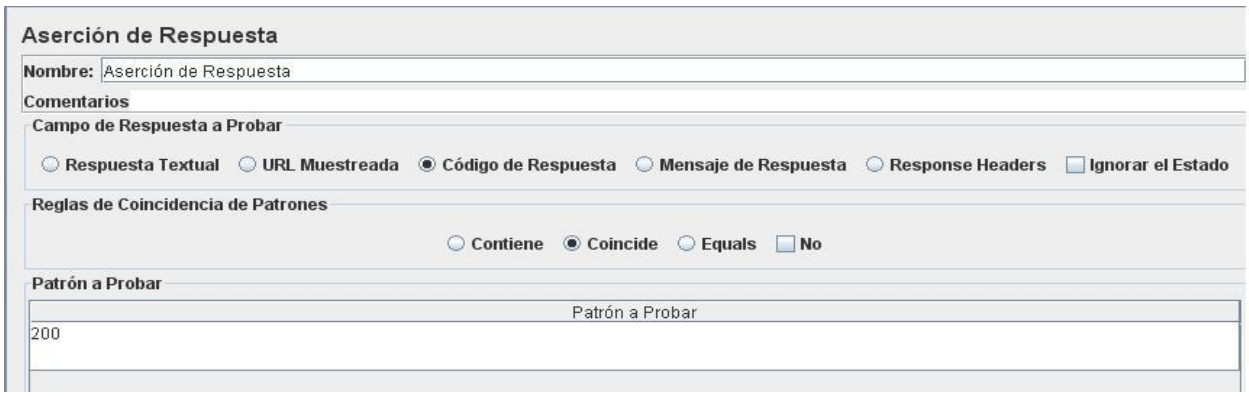


Fig. 14 Ejemplo de configuración de Aserción de Respuesta

Teniendo todos los recursos para realizar la prueba presionamos Ctrl + R o mediante la opción Run en la Barra de herramientas y la opción Start (Fig. 15) 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. Presionando Ctrl + Shift + E se limpian resultados específicos de las pruebas y con Ctrl + E, se limpian todos.

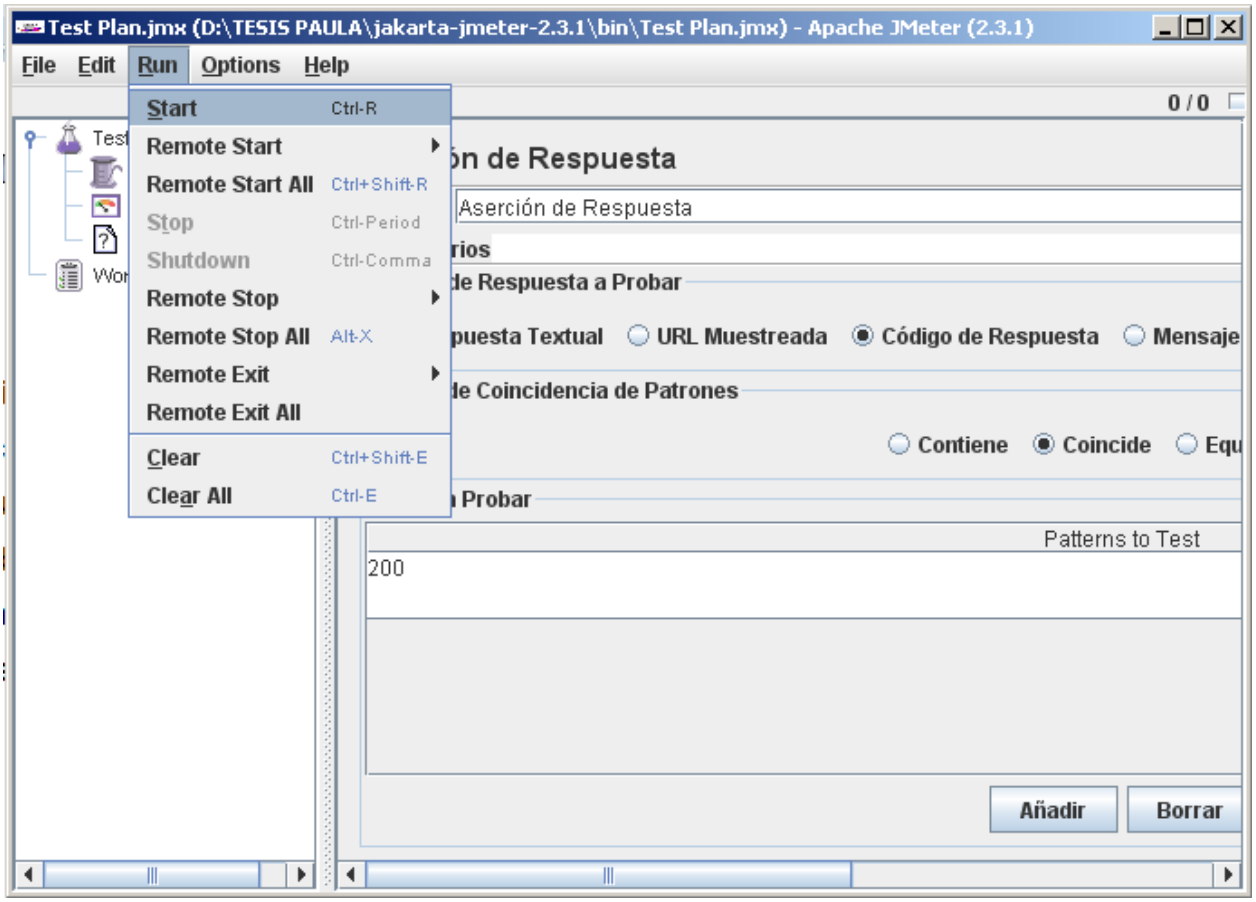


Fig. 15 Ejecución de la prueba.

Estos son los pasos básicos para realizar una prueba de carga y estrés, depende de la estrategia que se desea seguir, el éxito de la misma. Ya que para realizar estas pruebas se deben seguir sus principios básicos de concurrencia y carga y esto solo se logra definiendo cuales ha 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.

Conclusiones

Queda demostrado que es una buena práctica utilizar herramientas que automaticen los procesos de pruebas, gracias a las ventajas que traen en conjunto con su utilización.

Para lograr una mayor calidad y eficacia en el proceso de automatización es necesaria una amplia visión de software que se necesita probar; distinguir las áreas que se necesitan probar y las prioridades de las mismas; llevar a cabo la concepción de las pruebas con el régimen que dictaminan las pautas para su desarrollo y además un buen acondicionamiento del ambiente de las pruebas.

GLOSARIO DE TÉRMINOS:

Artefacto: Productos tangibles del proyecto que son producidos, modificados y usados. Pueden ser modelos, elementos dentro del modelo, documentos, gráficos, código fuente y ejecutables.

Aseguramiento de Calidad: Conjunto de actividades planificadas y sistemáticas necesarias para proporcionar confianza en que el producto software satisfará los requisitos dados de calidad.

Calidad: Calidad de software. Satisfacción de las necesidades de los usuarios.

Caso de prueba: Conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular, por ejemplo, ejercitar un camino concreto de un programa o verificar el cumplimiento de un determinado requisito. También se puede referir a la documentación en la que se describen las entradas, condiciones y salidas de un caso de prueba.

Componente: Parte discreta de un sistema capaz de operar independientemente, pero diseñada, construida y operada como parte integral del sistema.

Confiabilidad: Posibilidad que tiene un sistema de realizar las funciones para las que fue diseñado sin fallos.

Eficiencia: Medida del grado en que una actividad alcanza sus objetivos, optimizando el uso de los recursos disponibles.

Defecto*:* Consecuencia de un error. Incumplimiento de un requisito asociado a un uso previsto o especificado.

Fase: Son los pasos en que se descomponen las metodologías. Cada fase puede o no estar subordinada a otra fase, pudiendo existir entre ellas relaciones de dependencia.

Falla: Error en el funcionamiento que emite el sistema.

Hardware: Conjunto de componentes físicos de una computadora. Refiérase a objetos tangibles y palpables como son los discos, lectores de discos, monitores, teclados, las impresoras, tarjetas y chips.

Herramienta: Subprograma o módulo encargado de funciones específicas y afines entre sí para realizar una tarea. Una aplicación o programa puede contar con múltiples herramientas a su disposición. Por ejemplo, el corrector ortográfico puede ser una herramienta en una aplicación para redactar documentos, pero no es una aplicación en sí misma.

Interfaz: Una colección de operaciones que se usan para especificar el servicio de una clase o de un componente. Un juego nombrado de operaciones que caracterizan la conducta de un elemento. La Interfaz hombre-máquina es un canal comunicativo entre el usuario y el ordenador.

JUnit: es un conjunto de librerías que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

Metodología: Es un conjunto de procedimientos, técnicas, instrumentos y documentos que ayudan a los analistas y programadores en sus esfuerzos para obtener un nuevo sistema informativo. Consiste en fases que guían al diseñador en la elección de las técnicas más apropiadas en cada paso del proyecto y lo ayudan a planificar, dirigir, controlar y evaluar el mismo.

NUnit: es una herramienta utilizada para escribir y ejecutar pruebas en .NET

Portabilidad: Característica de ciertos programas que les permite ser utilizados en distintos ordenadores sin que precisen modificaciones

Prueba: Prueba de software. Ejecución de un sistema bajo condiciones específicas, se observan y se analizan los resultados realizándose una evaluación de los mismos.

Procedimiento: Forma específica de llevar a cabo una actividad. En muchos casos los procedimientos se expresan en documentos que contienen el objeto y el campo de aplicación de una actividad; que debe hacerse y quien debe hacerlo; cuando, donde y como se debe llevar a cabo; que materiales, equipos y documentos deben utilizarse; y como debe controlarse y registrarse.

Producto: Es cualquier cosa que puede ser ofrecida al mercado para su compra, para su utilización o para su consideración. Es cualquier bien, servicio o idea capaz de motivar y satisfacer a un comprador.

Proyecto: Elemento organizativo a través del cual se gestiona el desarrollo del software, el resultado de un proyecto es una versión del producto.

Proceso: Un proceso se define como un conjunto de tareas, actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido.

Roles: Papel que ejerce un actor en una actividad o proyecto.

Requerimientos: Una condición o capacidad que debe estar presente en el sistema o componentes del sistema para satisfacer un contrato estándar, especificación u otro documento formal.

Recursos*:* Son todos aquellos elementos necesarios, tanto tangibles como intangibles, para que una organización cumpla con sus objetivos.

RUP: Rational Unified Project, (Proceso Unificado de Desarrollo).

Sistema: Conjunto de elementos relacionados que interactúan entre sí para lograr un fin determinado.

Software**:** Software es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

Stakeholders: Individuo o grupo de individuos que tiene intereses directos e indirectos en una empresa que puede ser afectado en el logro de sus objetivos por las acciones, decisiones, políticas o prácticas empresariales, ya que estas tienen obligación moral con la sociedad y estas obligaciones se conoce como responsabilidad social empresarial.

1