Universidad de las Ciencias Informáticas

Facultad 4

Propuesta de Procedimiento y Herramienta para el desarrollo de Pruebas de Rendimiento de Carga y Estrés en el Grupo de Calidad de FORTES

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autora: *Sulamis Cordoves Mustelier*

Tutor: *Ing. Yinimary Ortega Montoya*

La Habana. 2011

Año 53 de la Revolución

Declaración de Autoría

Declaro que soy la única autora del presente trabajo de diploma que tiene como título: “Propuesta de procedimiento y herramienta para el desarrollo de pruebas de rendimiento de carga y estrés en el Grupo de Calidad de FORTES” y autorizo exclusivamente al Centro FORTES perteneciente a la Facultad 4 de la Universidad de las Ciencias Informáticas a utilizarlo.

Para que así conste firmo la presente a los días_____ del mes _____ del año_____.

Autora	Tutor
_____	_____
Sulamis Cordoves Mustelier	Ing. Yinimary Ortega Montoya

Dedicatoria

Este trabajo está dedicado a la memoria de mis dos abuelas, especialmente a Nelida (mima) por siempre tratar de guiarme por el buen camino, nunca has dejado de estar ni un segundo en mi corazón y en mi mente…. te extraño mucho.

A mis padres por el apoyo que me han dado durante toda mi vida, por su amor y por confiar en mí, especialmente a mi madre que lo es todo para mí, los quiero con todo mi corazón.

A mis hermanos Dicsiel y Dicksan por ser mi fuente de inspiración. Los quiero mucho.

A Venus porque eres como mi segunda madre, te doy las gracias por ayudarme incondicionalmente a mí y a mi mamá, decirte que aunque no seamos más vecinas, siempre te recordaré como la persona más buena que he conocido. Te quiero.

A toda mi familia que siempre me ha apoyado los quiero mucho.

A todos mis amigos y amigas por estar siempre en las buenas y en las malas.

Agradecimiento

Quiero agradecerle en primer lugar a dios por permitirme graduarme, por estar junto a mi en todo momento, darme fuerzas para seguir luchando ante los problemas y guiarme en todos los aspecto de mi vida.

Agradezco a mi mamá por siempre está ahí en los momentos malos y buenos de mi vida, por enseñarme con su ejemplo a ser una persona correcta, fiel a dios y a mis principios. Le doy gracia por apoyarme, por malcriarme, por mimarme y por confiar en mí.

A mi papá por apoyarme en todo, por siempre decirme que para lograr algo hay que sacrificarse, que la satisfacción de lograr un objetivo es personal. Le doy gracias por velar por mis intereses.

A mi hermano Dicsiel por ser un ejemplo para mí, a mi otro hermano Dicksan por hacerme reír en todo momento.

A mi familia y especialmente a mi primo Zair por ser mi amigo.

A todos mis vecinos del reparto y a la familia de amigos del reparto a Marlon, las mellizas Charo y Estrella, a Jesusín y Dayanira mi hermanita chiquitica, a Rachel y Reinier.

A todas las amistades que he conocido en esta universidad especialmente a las de mi antiguo grupo del 8106.

A mis amigas y amigos: Annia, no me extrañes, a eve, por ser una de las personas que me a hecho reír a carcajadas, me dijo juanki que te odia, a la mora por estar siempre conmigo tomando guarapo, en las fiestas y bailando disco jaja…., a yuli por ser una exelente persona, a Janet, no protestes más que te vas a poner vieja, a lili y Daril por ser tan buenas personas, ojalá se casen, a Milena ojalá y encuentres alguien especial, a los Pedros por ser las personas más alegres y fiesteros que he conocido, a Jose por ayudarme con la programación sin ti no lo hubiese logrado, a Roger por ser tan divertido y por

ponerme el apodo de chiverica, a lisi, quiero decirte que todos tu novios han estado puesto para mi, a Danay, Lisandra y Lice, que tienes que cargarme después de la exposición de la tesis, a Eduar , Chago y Rene y especialmente a Yamila que aunque hoy no seamos las mejores amigas y quiero decirte que durante el tiempo que lo fuímos amigas, llagaste a ser como la hermana que siempre quise tener. En general los quiero a todos y no cambien nunca.

A mi tutora Yinimry y a Javier por ayudarme en la etapa que más trabajo he pasado en esta escuela.

Y a los profesores que de una forma u otra contribuyeron a mi formación como profesional y como persona.

Resumen

La calidad del software constituye un objetivo principal para el éxito de un proyecto. Y una manera de lograrlo es a través de las pruebas que se le hacen a este software, para detectar posibles fallos en el funcionamiento del mismo. Una de estas pruebas que se desarrollan en esta investigación son las de rendimiento de tipo carga y estrés, las cuales permitirán determinar la rapidez de una aplicación web ante determinada carga de trabajo ya sea cantidad de usuarios conectados concurrentemente, uso de recursos como: memoria, procesador, red, disco duro.

En el presente trabajo se propone un procedimiento para la realización de las pruebas de rendimiento de tipo carga y estrés para el Grupo de Calidad FORTES (GCF), proyecto que presta servicio de diferentes pruebas a los productos desarrollados en el centro FORTES de la facultad 4. Se toma como punto de referencia el nivel de prueba en los cuales se aplican. Se selecciona la herramienta que ayudará a ejecutar el procedimiento realizando un estudio y comparación de estas. Se realiza el procedimiento con la herramienta seleccionada, definiendo actividad, trabajadores y artefactos. Y por último se aplica el procedimiento al proyecto Alfaomega, que desarrolla la plataforma ZERA a la cual se le realizó exitosamente las pruebas de rendimiento de carga y estrés al módulo de Gestión de Softarea.

Palabras claves

Calidad de Software, Pruebas de Software, casos de prueba, pruebas de carga y estrés, procedimiento.

Índice

Introducción

En la actualidad la industria del software ha experimentado un gran auge a nivel mundial. Los constantes avances tecnológicos provocan una incesante necesidad de remodelación en diversas áreas de la informática y hacen que el profesional, esté obligado a mantenerse actualizado sobre las nuevas tecnologías para crear los productos con mayor calidad.

A medida que va pasando el tiempo las empresas necesitan producir aplicaciones informáticas de alta calidad, en el menor tiempo posible y a costos mínimos, debido a que cada vez el cliente es más exigente con respecto a la calidad de los productos. Esto ha traído consigo que se agudicen las competencias y que durante el desarrollo del software todo el equipo de trabajo esté en función de la calidad del producto. Por esta razón surgen los diferentes tipos de pruebas, que permiten encontrar errores en cuanto a rendimiento, seguridad, escalabilidad, eficiencia, operaciones internas, capacidad de soportar cambios, que son a menudo la principal causa de que los proyectos no se entreguen en tiempo o que el cliente no esté satisfecho con el producto.

Probar el funcionamiento correcto de un software, antes de ser distribuido al cliente, es una tarea muy difícil de comprobar, ya que el comportamiento de una aplicación no se puede cuantificar, pero muchas veces no se tiene una metodología clara, ni la necesaria automatización o soporte de herramientas para comprobar la calidad del trabajo. Aunque no se pueda verificar que un software funcione 100% correctamente, un buen procedimiento durante el proceso de prueba y el uso de las herramientas modernas, puede mejorar la productividad y eficiencia de las pruebas del software.

El Comandante en Jefe Fidel Castro definió a la Universidad de las Ciencias Informáticas (UCI) como un centro experimental, docente-productor, de lo que se derivan sus dos misiones fundamentales: preparar profesionales altamente calificados comprometidos con su Patria, y contribuir a través de la actividad productiva a convertir la informática en una de las principales ramas que aportan recursos a la nación. Por esta razón, la universidad aspira a convertir la Industria Cubana del Software en un renglón fundamental de la economía, ser líder del desarrollo de las empresas de software en Cuba e insertarse en el mercado internacional, lo que constituye un reto productivo para la universidad.

La UCI se dedica a la realización de software para las diferentes esferas de la sociedad, y una de las etapas que la complementan es la de prueba de software, que no es más que un conjunto de actividades en función de encontrar la mayor cantidad de errores antes de ser distribuido al cliente, las mismas

permiten demostrar la excelencia del desempeño de una aplicación, asegurando así la calidad del software.

En la UCI existen varios centros de producción distribuidos entre las distintas facultades y específicamente en la Facultad 4 se encuentra el Centro de Tecnologías para la Formación (FORTES) que cuenta con el Grupo de Calidad que ofrece servicios para el aseguramiento, verificación y validación de los productos que se desarrollan en los proyectos de este centro y en otros, los cuales, debido a las fuertes competencias que existen en cuanto a la calidad, escalabilidad y disponibilidad, utilizan las modernas tecnologías de desarrollo y metodologías orientadas a servicios haciendo que las nuevas aplicaciones sean capaces de servir a usuarios rápidamente en entornos de constante cambio. Para comprobar si se cumple con este requisito, se hace necesario medir el rendimiento de estas aplicaciones web a través de las pruebas de carga y estrés que permitirán:

- Detectar errores de rendimiento antes del despliegue de un producto.
- Determinar lo rápido que responde el sistema en condiciones particulares de trabajo.
- Validar y verificar los atributos como escalabilidad, fiabilidad y uso de recursos (procesador, red, memoria).
- Determinar la rápida detección de los cuellos de botella (puntos débiles de la aplicación) ante tiempos de respuesta excesivos.

En el Grupo de Calidad FORTES (GCF) los servicios que se brindan son: las pruebas a nivel de sistema específicamente, pruebas funcionales. Actualmente no se están realizando pruebas no funcionales, o sea, no se está probando los requisitos no funcionales de los productos desarrollados en FORTES, debido a que en estos proyectos no se tiene claro las especificaciones de rendimiento (requisitos), es decir, no se expresa qué tiempo máximo de respuesta es aceptado para cierto número de usuarios conectados a la aplicación, se desconoce cuántos usuarios conectados simultáneamente puede soportar el sistema, se desconoce las herramientas necesarias y efectivas para ejecutar las pruebas de carga y estrés que permiten simular la utilización de una aplicación web por parte de múltiples usuarios conectados, no se tiene determinado los tiempos de respuesta por cada petición que se haga al servidor.

La ausencia de los factores mencionados anteriormente trae como consecuencia, que no exista seguridad de que los productos realizados en estos proyectos salgan con calidad, no se permita conocer que parte del sistema o que carga de trabajo hace que el sistema rinda de manera incorrecta, un incorrecto

aseguramiento, verificación y validación del software desarrollado en el centro, que una vez entregado el producto el sistema falle con determinada demanda de usuarios conectados, y además la incapacidad de predecir el comportamiento de la aplicación en cuanto a escalabilidad, fiabilidad y uso de los recursos.

Basándose en lo anterior surge el siguiente **problema a resolver:** ¿Cómo determinar el comportamiento de las aplicaciones desarrolladas en FORTES en cuanto a escalabilidad, fiabilidad y uso de recursos?

Objetivo general: Proponer un procedimiento que permita el desarrollo de las pruebas de carga y estrés en el GCF con el apoyo de una herramienta automatizada.

Objetivos específicos:

- Estudiar los tipos de pruebas de rendimiento enfocándose en las pruebas de carga y estrés.
- Definir el alcance y el objetivo de las pruebas de carga y estrés.
- Estudiar actividades para realizar las pruebas de carga y estrés.
- Seleccionar la herramienta para ejecutar las pruebas de carga y estrés.
- Proponer un procedimiento para realizar las pruebas de carga y estrés con el apoyo de una herramienta.
- Validar la propuesta del procedimiento y la herramienta en un proyecto.

Posibles resultados: Este procedimiento brindará la posibilidad de describir los pasos necesarios para guiar el desarrollo de las pruebas de carga y estrés, permitiéndole al GCF tener una herramienta que responda a la pregunta: ¿Qué tan rápido el sistema responde ante determinada carga?, lo cual ayudará a mejorar la calidad de los servicios que ofrece este actualmente y de la aplicación que se esté probando, contribuyendo al mejoramiento del sistema a la hora de que sea usado por el usuario final minimizando así la aparición de fallas que traigan consecuencias negativas durante su uso.

Objeto de estudio: Los procesos de pruebas no funcionales de rendimiento.

Campo de acción: Procesos de pruebas de carga y estrés para GCF.

Para el logro de los objetivos planteados en el presente trabajo se proponen las siguientes **tareas investigativas:**

- Estudio y valoración del estado del arte centrado en trabajos realizados sobre pruebas de rendimiento y herramientas automatizadas para ejecutar las pruebas de carga y estrés.
- Definición de los métodos para el diseño de los casos de pruebas.
- Investigación sobre las herramientas libres que se pueden utilizar.
- Análisis y selección de la herramienta candidata a utilizar, de acuerdo con sus principales funcionalidades y características en particular.
- Propuesta de un procedimiento para el desarrollo de las pruebas.
- Validación de la propuesta de procedimiento.

Idea a defender: Con la puesta en práctica de un procedimiento que permita el desarrollo de las pruebas de carga y estrés y con el empleo de herramientas automatizadas, se lograría introducir estos tipos de pruebas a los servicios que brinda el GCF de forma estructurada y estandarizada.

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

El **primer capítulo** de fundamentación teórica contiene los principales conceptos y fundamentos necesarios para el dominio del problema, las actividades para la realización de las pruebas de carga y estrés, estrategias para que sean aplicadas durante la fase de prueba.
En el **segundo capítulo** se propone el procedimiento para realizar las pruebas de carga y estrés apoyándose en la herramienta seleccionada.
En el **tercer capítulo** se aplica el procedimiento a una aplicación web, se realiza una evaluación de los resultados y se documentan los errores encontrados en estos resultados.

CAPÍTULO 1 Fundamentación Teórica

1.1 Introducción

En este capítulo se explican los temas fundamentales en los que se basa la investigación. Se hace referencia a los conceptos como calidad de software y pruebas de software, elementos que se deben tener en cuenta antes de probar un software como: objetivos, principios y alcance, que ayudarán a entender con claridad el procedimiento. Se explican los niveles de pruebas que se le aplican a un sistema durante su desarrollo y los tipos de pruebas de rendimiento haciendo énfasis en las pruebas de carga y estrés, se realiza un estudio y comparación de las herramientas que permiten ejecutar las pruebas de rendimiento de carga y estrés y finalmente se selecciona la herramienta más indicada para la automatización de estas pruebas.

1.2 Calidad de Software

Uno de los principales problemas que se afronta actualmente en la esfera de la informática es la calidad del software. Este tema ha sido motivo de preocupación por empresas, desarrolladores, analistas, diseñadores, investigadores etc. Por lo que se ha hecho muchas investigaciones en base a la siguiente interrogante ¿Cómo obtener un software con calidad? la respuesta a esta interrogante está ligada con las actividades que deben realizarse durante el desarrollo del producto, por lo que antes de que se inicien las actividades que garanticen la calidad del software, es importante definir que es calidad de software, una vez que se comprende este concepto, el equipo de desarrollo del producto, procede a identificar las actividades que garanticen la calidad del mismo.

Es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. [1]

Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario. [1]

Es el desarrollo de software basado en estándares con la funcionalidad y rendimiento total que satisfacen los requerimientos del cliente. [2]

Procesos de desarrollo, artefactos, gestión de proyectos, análisis y diseño, especificación de requerimientos, arquitectura, son solo algunos de los componentes que se aglomeran para conformar la ingeniería de software como disciplina para la creación y mantenimiento de software. Dentro de ésta,

existe un subconjunto de teorías, herramientas y métodos orientados a lo que se denomina la calidad del software. [2]

Características que hacen que a un software con calidad

Mantenibilidad: el software debe ser diseñado de tal manera, que permita ajustarlo a los cambios en los requerimientos del cliente. Esta característica es crucial, debido al inevitable cambio del contexto en el que se desempeña un software.

Confiabilidad: incluye varias características además de la confiabilidad, como la seguridad, control de fallos, etc.

Eficiencia: tiene que ver con el uso eficiente de los recursos que necesita un sistema para su funcionamiento.

Usabilidad: el software debería ser utilizado sin un gran esfuerzo por los usuarios para los que fue diseñado, documentado, etc. [2]

Para las empresas la calidad de un software constituye un reto actualmente, ya que cada vez surgen nuevos paradigmas en las tecnologías para mantenerla y, por lo tanto, hay más competencias que dan lugar, a que continuamente, se esté probando, si el software tiene determinadas propiedades o características como rendimiento, fiabilidad, seguridad, eficiencia, que se correspondan con los estándares, las expectativas del cliente, y requerimientos especificados desde el inicio del desarrollo del producto o servicio. La calidad de software es sinónimo de flexibilidad, corrección, usabilidad, portabilidad, integridad. Es el conjunto de cualidades que lo caracteriza, que determinan su utilidad y su atracción por parte del cliente más allá de la demanda actual.

La calidad del software constituye la principal característica que debe poseer un producto, y una de las actividades que se hacen para comprobarla, es a través de las pruebas que se realizan durante todo el ciclo de desarrollo del software, buscando errores que no se correspondan con distintos factores que tiene el producto como: la capacidad de soportar cambios, las características en sus operaciones y la adaptabilidad a nuevos entornos.

1.3 ¿Qué es una prueba de software?

En el desarrollo del software se realiza una serie de actividades. Todas realizadas por el personal del equipo de implementación del producto, que como seres humanos que son, pueden cometer errores en cuanto a la implementación del código, uso de tecnologías, interfaz del usuario o cambiar de idea sobre el

diseño del producto, es por eso que se necesita probar varias veces detectar defectos antes de que el software sea entregado al cliente final.

Simultáneamente con el proceso de desarrollo deben ir actividades que garanticen la calidad del producto que se está construyendo y una de estas actividades son las Pruebas de Software, que básicamente es una fase en el ciclo de vida del software que consiste en probar las aplicaciones construidas, y desde que se definan los requisitos funcionales como no funcionales, se debe ir probando, ya que, la mayor parte de los defectos se concentran en las fases tempranas del proceso de desarrollo.

Las pruebas de software son un elemento crítico para la garantía de la calidad del software y presenta una revisión final de las especificaciones, del diseño, y de la codificación. [3]

Según la bibliografía consultada las pruebas de software 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. [4]

Además son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador. [4]

Prueba de Software es la ejecución del código usando combinaciones de entradas, en un determinado estado, para revelar defectos. [5]

Se considera además que es el diseño e implantación de un software especial: uno que ejercita otro software con la intención de hallar defectos. [5]

Otros actores plantean que son las ejecuciones de programas con el objetivo de encontrar defectos y fallas. [6]

Analizando cada concepto que se expone anteriormente, se puede decir que el concepto de: “Las pruebas de software son las ejecuciones de programas con el objetivo de encontrar defectos y fallas.”, es el que más se acerca, a lo que se quiere desarrollar en esta investigación, ya que para hacerle las pruebas no funcionales específicamente a los requisitos de rendimiento de los productos desarrollados en los proyectos, que soliciten los servicios del GCF, se empleará una herramienta que al ejecutarse, permitirá encontrar fallos en el funcionamiento en dicho sistema.

Realizar una prueba de software es la única manera adecuada para determinar el estado de la calidad de un producto software durante el proceso de pruebas, es la que permite ejecutar el software bajo ciertas condiciones establecidas por equipo de desarrollo y evaluar resultados, además permite comprobar si este

cumple con sus características operativas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos especificados. Una forma de ejecutar las pruebas es a través de los métodos de pruebas que permiten diseñar casos de prueba, especificados de forma estructurada mediante las técnicas de pruebas.

1.4 Objetivo de las pruebas

Se debe tener bien claro con qué objetivos se prueba un software, para enfocar el trabajo en función de dar cumplimiento a estos para garantizar la calidad del producto desarrollado.

Según Glen Myers en el libro de Pressman establece normas que pueden servir para los objetivos de las pruebas:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- La prueba tiene éxito si se descubre un error no detectado hasta entonces. [3]
- Encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzo posibles.
- Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.
- No debe ser redundante.
- Una buena prueba no debe ser ni muy sencilla ni excesivamente compleja: es mejor realizar cada prueba de forma separada si se quiere probar diferentes casos. [4]

1.5 Principios de las pruebas

Para realizar un buen proceso de pruebas es fundamental, guiarse por los principios básicos que facilitará a los ingenieros de pruebas mediante métodos y técnicas crear un buen diseño de casos de pruebas.

- A todas las pruebas se les debería hacer un seguimiento hasta ver si se cumplen los requisitos del cliente.
- Las pruebas deberían planificarse mucho antes de que empiecen.

- El principio de Pareto es aplicable a la prueba del software.
- Las pruebas deberían empezar por lo pequeño y progresar hacia lo grande.
- No son posibles las pruebas exhaustivas
- Son más eficientes las pruebas dirigidas por un equipo independiente.[3]

1.6 Procedimiento de pruebas

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. Son los pasos para llevar a cabo uno o varios caso de pruebas. [7]

Definir el procedimiento de las pruebas en un proyecto servirá de guía, para el equipo de realización de prueba. Este procedimiento será una instrucción para cada miembro del equipo de prueba, que le permitirá conocer: las fases por las que se transita el procedimiento durante la etapa de prueba, los roles, las actividades, los artefactos, el tipo de hardware y software, que se debe utilizar para ejecutar ciertas pruebas.

Ejemplo general de procedimientos y técnicas que se aplican a todos los programas y pruebas:

- Para probar los programas debe usarse siempre datos de entrada bien definidos para los cuales se conozca de antemano los resultados correctos que deben obtenerse.
- Debe tratar de detectarse primero lo defectos obvios y para ello se utilizan datos de prueba muy simples y luego sí, realizar las pruebas más complejas.
- Si el programa se modifica mientras se aprueba, cambie una sola cosa a la vez y utilice el mismo dato de prueba con que detectó el defecto. El defecto es corregido cuando al repetirse la prueba, el defecto ya no se detecta.
- Debe probar su programa para verificar si es capaz de detectar entradas incorrectas. [8]

La propuesta de procedimiento de esta investigación basará su uso en casos de pruebas específicamente en sus escenarios.

Un **caso de prueba** específica como probar un sistema, incluyendo entradas o resultados con el cual será probado y condiciones bajo las cuales ha de probarse.

Según el " Modelo de Proceso para el Servicio de Prueba" de GCF, las pruebas que se le hacen a los proyectos, que realizan la solicitud a este, se le aplican en la etapa de liberación, donde minuciosamente se van revisando los módulos con varias iteraciones para asegurarse de detectar la mayor cantidad de No Conformidades (NC) en el sistema y una vez terminada se pasa al siguiente módulo hasta terminar completamente la revisión del sistema emitiendo un acta de liberación por parte del GCF, para que sea probado por Calisoft.

Por tan tanto este procedimiento se aplicará de la misma forma y se revisará con la ayuda de una herramienta automatizada a un módulo del sistema, para corregir NC antes de la entrega de la aplicación al centro de Calisoft.

1.7 Métodos de Pruebas

Para diseñar casos de pruebas se debe estudiar diferentes métodos de pruebas, que le permitan al analista comprender como puede diseñarlos de manera adecuada, lo más completo posible, que tengan una alta probabilidad de encontrar el mayor número de errores con los mínimos esfuerzos y en el menor tiempo posible, que se diseñen de acuerdo con el tipo de prueba y a la funcionalidad que se quiera probar. Los métodos más utilizados son las pruebas de caja negra y las pruebas de caja blanca, que estos a su vez, poseen técnicas para diseñar casos de pruebas.

1.7.1 Pruebas de caja negra

Pruebas que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. Estas pruebas comprueban si el sistema cumple con los requisitos funcionales, ignorando la estructura lógica interna del software.

Se centran principalmente en los requisitos funcionales del software que 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. [3]

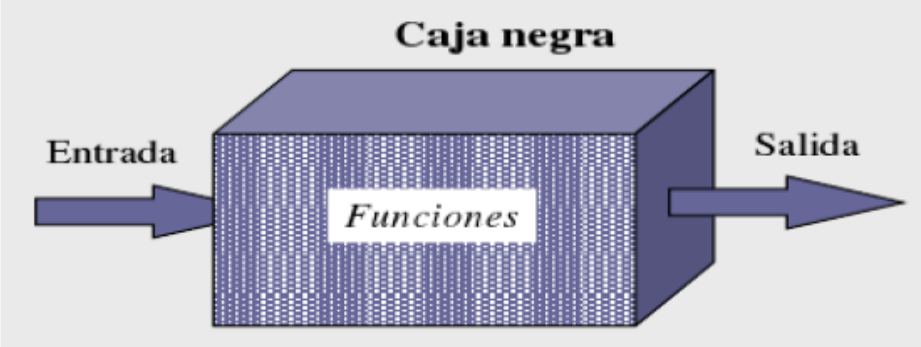

Figura. 1 Representación método de caja negra

Las pruebas de caja negra tienen un conjunto de técnicas que permiten diseñar los casos de pruebas que satisfacen el siguiente criterio: reduce el número de casos de pruebas y dicen algo sobre la presencia o ausencia de errores. Estas técnicas son **pruebas basadas en grafo, análisis de valores límite, partición equivalente.**

Partición equivalente

Este es el método más utilizado, se enfoca en la realización de los casos de pruebas mediante la partición equivalencia, se basa en la evaluación de las clases de equivalencia para una condición de entrada, o sea, busca los distintos datos de entrada tanto válidos como no válidos que se puedan entrar a un programa para descubrir cualquier fallo que éste tenga con respecto a los valores entrados. Una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. [9]

El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. [9]

Reglas que ayudan a definir clases de equivalencia

- Si una condición de entrada especifica un rango, se identifica una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada requiere un valor específico, se identifica una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un miembro de un conjunto, se identifica una clase de equivalencia válida y una inválida.

- Si una condición de entrada es lógica, se identifica una clase de equivalencia válida y una inválida. [3]

Una vez definido la clase de equivalencia se procede a diseñar casos de pruebas para la partición de equivalencia, especificar que existen dos tipos de clases de equivalencia, la válida (entradas de valores válidos al programa) y la inválida (entradas de valores inválidos al programa). Después de asignarle un identificador a la clase de equivalencia, se procede a definir casos de pruebas con cada clase equivalencia. [9]

- Primero se escriben caso de pruebas que cubran tantas clases equivalencias válidas sean posibles y así sucesivamente hasta que todas sean cubiertas por casos de pruebas.
- Segundo se escriben caso de pruebas que cubran tantas clase equivalencia inválidas sean posibles y así sucesivamente hasta que todas sean cubiertas por casos de pruebas.

Pruebas basadas en grafo

Definir cuáles son los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones de conexión entre ellos. Se modela un grafos de estos objetos y sus relaciones y después se define una serie de pruebas que cubran todo el grafo y verifiquen que las relaciones entre estos objetos son las esperadas. [3]

Análisis de valores límite

La mayoría de las veces los errores tienden a darse más en los extremos de los valores de entrada que en el centro, es por eso que con la técnica análisis de valores límite (AVL) se crean casos de pruebas que analicen los valores límite. Esta técnica complementa la de partición equivalente. [3]

En resumen, las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se corresponda con su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario. O también la mayoría de las veces se diseñan los casos de pruebas para cuando el sistema esté completamente construido, o sea, cuando se hagan las pruebas de integración que es donde se involucran una serie de módulos y se

termina probando el sistema en conjunto. Dentro de estas pruebas utilizan distintas técnicas que reducen el número de casos de pruebas y ayudan a detectar mayor cantidad de errores.

1.7.2 Pruebas de caja blanca

Se comprueban los caminos lógicos del software. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado (sobre el código).

Requieren del conocimiento de la estructura interna del programa, estas pruebas deben garantizar como mínimo que se ejercite por lo menos una vez todos los caminos independientes para cada módulo, que se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa, que se ejerciten las estructuras internas de datos para asegurar su validez, además que se ejecuten todos los bucles en sus límites y con sus límites operacionales. [3]

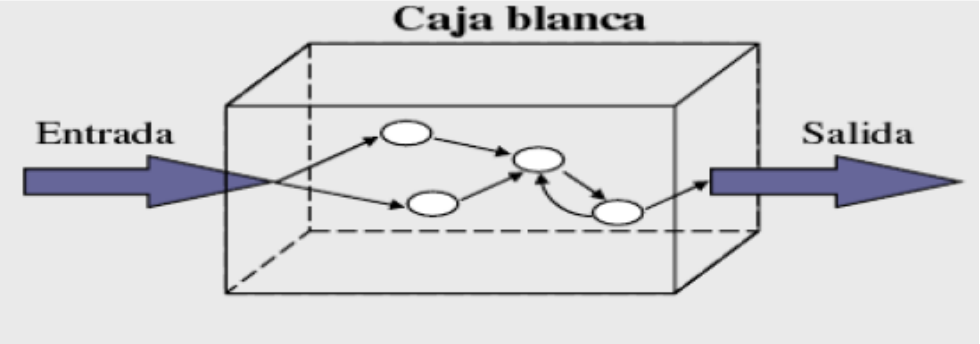

Figura 2 Representación método de caja blanca.

Las pruebas de Caja Blanca tienen un conjunto de técnicas que permiten diseñar los casos de pruebas que buscan errores en la estructura lógica de la aplicación. Entre estas técnicas se encuentra la prueba del camino básico.

Pruebas del camino básico

La prueba del camino básico es una técnica de prueba (propuesta por Tom McCabe) y la más usada. Permite al diseñador de caso de prueba obtener una medida de la complejidad lógica de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, los casos de pruebas obtenidos del conjunto básico garantizan que cada camino se ejecuta al menos una vez o sea cada sentencia del programa. [3]

Como apoyo para llevar a cabo la prueba del camino básico se utiliza los grafos de flujo que son para representar el flujo de control lógico de un programa. Y a este grafo se le calculará la complejidad

ciclomática que no es más, que una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática, define el número de caminos independiente del conjunto básico de un programa y brinda el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.
En el grafo de flujo el camino independiente está constituido por al menos una arista que no haya sido recorrida anteriormente. [3]

Con estas pruebas se está siempre observando el código, es por eso que también se les llaman pruebas estructurales o caja de cristal. Este tipo de pruebas permitirá diseñar casos de pruebas que encuentren errores, cuando se esté implementando funcionalidades en el desarrollo del software, cuando hallan condiciones o controles anormales en el código. Estas pruebas realizan un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos y bucles del programa y examinado el estado del programa en varios puntos.

1.8 Estrategia de aplicación de las pruebas

La estrategia de prueba debe estar enfocada a descubrir una serie de errores en el sistema y para esto se debe definir una serie de pruebas según el nivel de construcción del software, a medida de que se definan más pruebas se podrá realizar actividades de verificación y validación que permitan evaluar la calidad de los productos que se generan.
En fin se analiza como plantear las pruebas durante el ciclo de vida del software, paralelo al proceso de construcción del software deben desarrollarse pruebas por niveles de prueba.

1.9 Niveles de prueba

Existen varios niveles de prueba definidos y estos niveles a su vez especifican qué tipos de pruebas se deben utilizar en cada uno de ellos y cuáles son sus objetivos. A continuación se hace referencia a los niveles de pruebas.

1.9.1 Nivel de Unidad

Se centra el proceso de verificación a la menor unidad de diseño del software: el componente software o módulo. [3]

1.9.2 Nivel de Integración

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. El objetivo de este tipo de pruebas es la verificación del correcto ensamblaje entre los distintos componentes o módulos. [3]

1.9.3 Nivel de Validación

Después de la integración, el software está completamente ensamblado como un paquete, se ha encontrado y corregido los errores de interfaz y puede comenzar una serie final de pruebas de software llamadas pruebas de validación. La validación se realiza con el objetivo de identificar, si el producto cumple con requisitos especificados por el cliente final. [3]

Criterios de la prueba de validación*:* La validación de software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad de los requisitos. Un plan de prueba traza la clase de prueba que se ha de llevar a cabo y un procedimiento de prueba define los casos de prueba específicos para descubrir errores de acuerdo con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfagan todos los requisitos funcionales, de rendimiento y que la documentación es correcta e intangible. [3]

1.9.4 Nivel de Sistema

Prueba el software funcionando como un todo. Aceptable para cuando el software se encuentra en la fase de construcción. Trata de probar que los objetivos para los que fue construida la aplicación no se cumplen en su totalidad y que por tanto hay que cambiar elementos en la aplicación. Se usa como base los objetivos originales. No existe un método en específico, sino que se dan lineamientos, a la hora de preparar los casos de prueba. Enfocadas a los requisitos funcionales del software, a su interacción con el cliente de la forma que ha sido pactada y enfocadas a los requisitos no funcionales del proyecto. [10]

1.9.5 Nivel de Liberación

Prueba el software funcionando como un todo y está prácticamente listo para ser presentado al cliente. Trata de probar que los objetivos para los que fue construida la aplicación no se cumplen en su totalidad y que por tanto el software no está en condiciones de ser presentado al cliente y hay que hacer modificaciones a la aplicación. [10]

En cada etapa de desarrollo del software se le debe aplicar estas pruebas por niveles, donde cada una de ellas, tiene diferentes objetivos, forma de aplicación, estrategias, diseño, construcción, herramientas

manuales o automatizadas, procedimientos que guían el proceso para asegurar la captura de errores y verificación de requisitos tanto funcionales como no funcionales.

1.10 Pruebas a Nivel de Sistema

Este nivel se divide en dos grupos las pruebas funcionales y las pruebas no funcionales con sus respectivas pruebas.

1.10.1 Pruebas Funcionales

Enfocadas a los requisitos funcionales del software, verifica el correcto funcionamiento de todas las operaciones detalladas en los requerimientos definidos por los usuarios finales. Estos requisitos son propiedades que el cliente no puede observar directamente como mantenibilidad, flexibilidad, se define también como capacidades o condiciones que el sistema debe cumplir. [10]

1.10.2 Pruebas No Funcionales

Las pruebas de sistema no funcionales se utilizan con el fin de probar de forma dinámica que el sistema funciona mediante la búsqueda de fallos en los atributos no funcionales (seguridad, prestaciones y rendimiento, volúmenes y cargas) del sistema. Estos requisitos o atributos, son propiedades o cualidades que el producto debe tener, para que clientes y usuarios puedan apreciar sus características y una vez conocida, se podrá comprobar cuan usable, seguro, conveniente y agradable puede ser el sistema, se podrá marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. [10]

1.10.2.1 Pruebas de Seguridad

Las pruebas de seguridad aseguran que el sistema modificado o nuevo incluya controles apropiados de acceso y que no se introduzca ningún agujero de seguridad que pueda comprometer a otros sistemas. Las pruebas de seguridad comprobarán la confidencialidad, la integridad y la disponibilidad del sistema. [3]

1.10.2.2 Pruebas de Rendimiento

Según la IEEE: “Las pruebas de rendimientos es el grado en que un sistema o componente realiza sus funciones designadas dentro de las limitaciones dadas, tales como la velocidad, precisión o el uso de la memoria”. [11]

Las pruebas de rendimiento tienen como propósito validar o determinar la velocidad, escalabilidad y las características de estabilidad de una aplicación que esté a prueba.

1.11 Tipos de pruebas de rendimiento

Existen diferentes tipos de pruebas de rendimiento que ayudarán a mejorar las capacidades de una aplicación observando y evaluando las respuestas del sistema ante todas las posibles situaciones que se puedan dar. Cada una de estas pruebas tiene sus objetivos y características.

Para ajustar los resultados obtenidos a los objetivos y necesidades que persigue esta investigación se centrará más en las pruebas de carga y estrés, aclarar que estas pruebas aunque estén en el grupo de las pruebas no funcionales también se pueden incluir en las funcionales, ya que además de ver los requisitos de hardware, software y del servidor, se especifican en la interfaz de la aplicación que se está probando, ejemplo: si con determinada cantidad de usuarios conectados concurrentemente, no se muestra alguna imagen, texto o gráfica.

1.11.1 Pruebas de Estrés

Estas pruebas son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento mediante la ejecución de un número de usuarios muy superior al esperado. Esta prueba tiene como finalidad el determinar la robustez de una aplicación cuando la carga es extrema y ayuda a administradores a determinar si la aplicación se comportará debidamente ante diferentes situaciones. [12]

Esta prueba consiste en sobrecargar la aplicación hasta que falle y determinar el punto donde empieza a tener problema en su funcionamiento. Ayuda a los administradores determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada. Su principal objetivo es analizar cómo se comporta el sistema cuando la carga sobrepasa su capacidad, saber si su integridad no se afecta en ningún momento durante la prueba.

1.11.2 Pruebas Carga

Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. El resultado de esta prueba nos dará el tiempo de respuesta de todas las transacciones críticas. [12]

Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación. Si la base de datos, el servidor de aplicaciones, y otros componentes sufren afectaciones durante la carga, la carga se diseña lo más real posible y mide anticipadamente el rendimiento que

proporcionará la aplicación. Identifica los puntos de ruptura de la aplicación por debajo de la demanda máxima.

1.11.3 Prueba de Estabilidad

Esta prueba se realiza para determinar si la aplicación puede aguantar varias pruebas de cargas seguidas o sea que se prueba continuamente según el total de carga que soporte el sistema. Estas pruebas buscan posibles deterioros o degradaciones en el rendimiento del sistema. [10]

1.11.4 Prueba de Pico

Esta prueba observa el comportamiento del sistema variando el número de usuarios concurrentes que se le introducen en forma de pico, o sea, se iría variando gradualmente la carga en determinados momentos de esta prueba. [13]

Las pruebas de rendimiento pueden tener distintos objetivos. Por ejemplo, pueden demostrar que el sistema cumple los criterios de rendimiento preestablecidos o pueden medir que partes del sistema o que carga hacen que el sistema rinda de forma incorrecta, permite saber qué tan bien se comportará una aplicación y cuántos usuarios concurrentes soportará en condiciones extremas. Pueden servir para validar y verificar requisitos en la calidad del sistema como escalabilidad, fiabilidad, estabilidad, velocidad y uso de los recursos.

1.12 Estudios sobre trabajos realizados de pruebas de rendimiento.

De acuerdo a estudios realizados sobre las pruebas de rendimiento en la UCI y mediante la consulta de diferentes bibliografías, se puede ver una serie de trabajos realizados para desarrollar las pruebas de carga y estrés en aplicaciones web, donde se definen actividades que posibilitan el correcto manejo de estas pruebas y que sirvan de guía para el desarrollo de esta investigación.

Cuatro etapas para realizar las pruebas de rendimiento y cada una de ella tiene una serie de actividades.

Definición de escenarios. En esta etapa se debe tener en cuenta el entorno que el sistema tendrá en producción, dígase (infraestructura tecnológica, normativa de uso) y cómo el usuario interactúa con el sistema ejemplo: cuántos usuarios y transacciones puede soportar el sistema, estos datos son imprescindibles para que la simulación se realice lo más real posible a la realidad.

Automatización. En esta etapa el uso del sistema debe ser automatizado por herramientas llamadas generadoras de carga. Las cuáles simularán la carga que se defina en la etapa anterior. Estas

herramientas permiten realizar actividades como: grabación de las transacciones, generalización de los scripts, modelado de escenario y distribución de la generación de carga.

Preparación de infraestructura. De manera paralela a la etapa de automatización se de crear todo el ambiente necesario para realizar las pruebas de rendimiento. Esta infraestructura debe tratar de simular el entorno de producción del sistema puesto a prueba, debe tener en cuenta tanto hardware como software.

Ejecución de las pruebas. En esta etapa finalmente de ejecutan las pruebas donde se conoce como funciona el sistema, así como la detección de los posible fallos del sistema (cuellos de botella). Conociendo los cuellos de botella, se pueden analizar las soluciones y así mejorar el rendimiento de la aplicación. [14]

A continuación se presenta una guía centrada en las pruebas de rendimiento que se le hace a aplicaciones web, que contiene diferentes actividades que se deben desarrollarse durante la ejecución de las mismas.

Identificar el entorno de prueba. Identificar el entorno de pruebas físicas y el entorno de producción, así como las herramientas y recursos disponibles para el equipo de pruebas. El entorno físico incluye hardware, software y configuraciones de red.

Identificar criterios de aceptación de rendimiento. Identificar el tiempo de respuesta, el rendimiento y la utilización de los recursos objetivos y limitaciones.

Plan de diseño y pruebas. Identificar los principales escenarios, determinar la variabilidad entre los usuarios representativos y cómo simular que la variabilidad, definir los datos de prueba, y establecer indicadores que deben recogerse.

Configurar el entorno de prueba. Preparar el entorno de prueba, herramientas y recursos necesarios para ejecutar cada estrategia como las características y componentes en el mercado para la prueba.

Implementar el diseño de la prueba. Desarrollar las pruebas de rendimiento de acuerdo con el diseño de la prueba.

Ejecutar la prueba. Ejecutar y supervisar las pruebas. Validación de las pruebas, datos de prueba, y los resultados de recogida.

Análisis de resultados, informe y vuelva a probar consolidar. Compartir los resultados y datos. Analizar los datos tanto de forma individual y como en equipo multifuncional. Cambiar la prioridad de las

pruebas restantes y volver a ejecutar, según sea necesario. [15]

Esta guía presenta una amplia información sobre las pruebas de rendimiento, incluyendo pruebas de carga y estrés, contiene información de cómo aplicar estas pruebas de acuerdo con las características del modelo de mejora de procesos (CMMI) para el desarrollo del software, elemento importante para la UCI, ya que se encuentra inmersa en el desarrollo del mismo. Además muestra una variedad de recomendaciones y consejos sobre cómo llevar a la práctica dichas pruebas de la mejor manera posible, para lograr que durante la ejecución de las mismas, se recojan resultados lo más preciso posible a la realidad. El contenido de esta guía se basa en lecciones aprendidas y experiencias obtenidas por expertos del tema.

La cuatro etapas que se describen anteriormente tienen mucha similitud con las actividades de esta guía, pero esta investigación realizará el procedimiento, apoyándose en la guía, debido a la amplia información, organización, estructura y calidad de los contenidos que esta presenta.

1.13 Funcionamiento de las pruebas automatizadas

El uso del sistema a probar deber ser automatizado por las herramientas conocidas como generadoras de carga. Estas simularán determinada carga lo más real posible a la realidad.

Aunque las herramientas automatizadas no sustituyen completamente las manuales presentan muchas ventajas con respecto a su flexibilidad, a la forma de utilización, que se requiere de menos esfuerzo. Sin la utilización de estas herramientas se haría necesario cierta cantidad de probadores y usuarios que ejecuten las transacciones sobre el sistema de manera concurrente y coordinada. Las herramientas automatizadas son específicamente para sistemas cambiantes en cuanto al aumento en el volumen de casos de prueba y aumento o cambio de los requisitos y se puede realizar un seguimiento de forma eficaz de los resultados de la prueba localizada. [16]

1.13.1 Herramienta de automatización para ejecutar las pruebas de Carga y Estrés

Existen diferentes formas de generar la carga de trabajo que se necesita para realizar pruebas de rendimiento, están las manuales y las automatizadas, la primera es la más fácil de implementar, pero también la más tediosa, ineficiente, la menos flexible y posiblemente la más cara, ejemplo: para realizar las pruebas con 100 usuarios cada vez que se haga una prueba, se necesitaría un laboratorio con espacio suficiente, cosa que en la UCI es muy difícil, y computadoras desde las que se generarán la carga, más toda infraestructura necesaria de software y hardware precisa, también que la mayoría de estos sistemas

cambian frecuentemente sus requisitos y es muy probable que aparezcan nuevos errores por eso que la segunda opción es la más adecuada, ya que se utilizaría una herramienta automatizada que corrigiera los errores cada vez que se cambiaran los requisitos, se emplearían desde el propio entorno de prueba y se actualizarían periódicamente con nuevas variables de entradas para compararlas con los nuevos posibles resultados.

1.13.2 Estudio de las herramientas para ejecutar las pruebas de Carga y Estrés

Para la automatización de las pruebas se han creado una serie de herramientas que ayudan en gran medida a su realización, ya que en la actualidad los sistemas que se necesitan son cada vez más grandes y complejos y con estas herramientas se puede reforzar el desarrollo de todo el proceso de pruebas, disminuyendo tiempo, esfuerzo y costo. A la hora de escoger la herramienta que van a automatizar las pruebas, se debe tener en cuenta de que existen varios tipos de herramientas, por lo que se hace necesario analizar cuál será más beneficiosa para realizar el proceso de pruebas, la más flexible en cuanto a su configuración.

En la universidad de las ciencias informáticas, el de proyecto Calisoft, la herramienta que se utiliza es la de Jmeter. En el grupo de calidad del Centro de Tecnología para la Formación (FORTES) no se realizan las pruebas de carga y estrés por lo que no se puede comprobar si el producto cumple con las expectativas del cliente con respecto al rendimiento de sus aplicaciones web. Debido a esta situación a continuación se describirá, y se comparará cada una de las posibles herramientas que se pueden utilizar para la realización de estas pruebas, y las que han tenido mucho éxito a nivel mundial.

- **Loadrunner**

Loadrunner es una herramienta de pruebas de rendimiento provista por la empresa tecnológica HP, que permite probar y analizar el comportamiento de una aplicación cuando esta es usada en condiciones normales, de estrés o de forma prolongada.

Otra forma de describirlo es que Loadrunner es una herramienta de rendimiento, que permite simular diferentes comportamientos cotidianos en un sistema.

En forma general tiene cuatro componentes el VUGEN, controlador, generador de carga y análisis, de manera general el VUGEN (Virtual UserGENerator) permite generar scripts que simularan el comportamiento de un usuario real, el controlador permite agrupar estos scripts y aportarles una especie de calendarización a los mismos. El generador de carga recibe instrucciones del controlador y ejecuta los

scripts según lo hayas configurado, por último, el análisis permite granular y comparar la información sobre los datos obtenidos durante la ejecución. [17]

- **Jmeter**

El JMeter es una herramienta libre, además es una herramienta Java, que permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web. Es una herramienta de carga para llevar acabo simulaciones sobre cualquier recurso de Software. [18]

JMeter una herramienta Java dentro del proyecto de Jakarta, que permite realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones web y bases de datos. Existe un gran número de herramientas para realizar pruebas gratuitas y de pago (LoadRunner), pero JMeter se destaca por su versatilidad, estabilidad, y por ser de uso gratuito. JMeter permite realizar pruebas web clásicas, pero también permite realizar test de FTP, JDBC, JNDI, LDAP, SOAP/XML-RPC y Web Service (en Beta). También permite la ejecución de pruebas distribuidas entre distintos ordenadores para realizar pruebas de rendimiento. [18]

El JMeter muestra los resultados de las pruebas en una amplia variedad de informes y gráficas. Además, facilita a una rápida detección de los cuellos de botella existentes debido al tiempo de respuesta excesivo. Todas estas herramientas pueden ser usadas para hacer las pruebas de eficiencia bajo carga intensiva, sin embargo, hay algunas que poseen ventajas con respecto a las demás, por lo que son óptimas a utilizar durante las pruebas en cuestión. En una aplicación es de vital importancia emplear algo de tiempo a preparar pruebas de eficiencia bajo carga y estrés, antes de ser entregada. El tiempo invertido es recuperado con creces, ya que se detectaran los posibles efectos laterales y se podrá comprobar si esa nueva funcionalidad soporta la cantidad de usuarios concurrentes que se especificaban en los requisitos. [18]

Las pruebas de eficiencia bajo carga intensiva deben de ejecutarse idealmente sobre un entorno estable, lo más similar posible al entorno final de producción, siguiendo objetivos como: [18]

Verificar que el sistema esté ajustado para soportar la máxima carga de trabajo posible usando la infraestructura actual, asegurar que, ante una carga de trabajo determinada, las páginas a las que se accede responden en menos del intervalo de tiempo especificado por el grupo de diseñadores, 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. [18]

Ventajas de la herramienta Jmeter

De las herramientas gratis, es la más completa y útil para el tipo de pruebas en cuestión.
Es una herramienta que sirve para realizar pruebas funcionales, pero también sirve para realizar pruebas de regresión en aplicaciones web, algo, que a veces es verdaderamente complicado, según la aplicación, pero que es casi imprescindible en el mantenimiento y evolución de las aplicaciones, si se quiere asegurar un nivel de capacidad adecuado en la entrega del producto.
Tiene una estructura en árbol que le da potencia, permitiendo que sea la imaginación de quien la use la que ponga los límites a la hora de diseñar el plan de prueba. Y brinda mayor cantidad de variantes para recoger los resultados obtenidos, que el resto de las herramientas gratis, lo que permiten hacer un análisis exhaustivo de las pruebas realizadas. [18]
En cuanto a la licencia para usar Jmeter: Costo de adquisición: $0, curva de aprendizaje: Muy corta, Soporte: Manual completo, foros, y gran contenido en online. [19]

- **Webserver Stress Tool**

Webserver Stress Tool es una aplicación HTTP-cliente/servidor que permite identificar los problemas críticos de rendimiento en un sitio web o un servidor web, que puede impedir una experiencia óptima para los visitantes de un sitio. [20]
Webserver Herramienta de tensión que simulan los usuarios independientes que dan un paso por un conjunto de URLs. Basado en parámetros específicos, la aplicación no sólo pide el HTML de una URL sino también marcos, imágenes, flash archivos, etc. Emulando el mismo comportamiento que un navegador de web mostraría cuándo accede a una aplicación web. [20]
Cada usuario es simulado por un hilo separado con su información de sesión propia (es decir, las cookies para cada usuario simulado se almacenan por separado). Las URLs pueden ser parametrizadas para cada usuario y la secuencia de URL puede ser variada. Peticiones POST y GET son compatibles, así como BASIC autenticación HTTP y otros ajustes. El inicio de licencia es de $249.95. [20]

Tabla 1.1. Comparación de las herramientas.

Herramienta	Clase de herramienta	Descripción	Organización	Plataforma

Loadrunner	Herramientas de carga que mide el rendimiento de una aplicación web. Sistema multiusuario y herramienta para pruebas de servidor.[21]	Herramienta para pruebas de Sistemas cliente/servidor. Permite realizar pruebas de rendimiento, pruebas de carga y afinación de aplicaciones para múltiples usuarios. [21]	Mercury Interactive. [21]	LoadRunner/UX: para aplicaciones UNIX/X Window, LoadRunner/PC: para aplicaciones MS Windows, LoadRunner/XL: funcionamiento y ejecución en servidores, LoadRunner/RTE: para aplicaciones basadas en estación o terminal. [21]
JMeter	Herramienta de carga que mide el rendimiento de una aplicación web, (Freeware). [21]	Apache Jmeter es una aplicación de escritorio 100% java, diseñada para realizar pruebas de carga funcional y medición de rendimiento. Esta fue diseñada originalmente para probar aplicaciones Web, se ha ampliado desde entonces a otras funciones de prueba. [21]	The Apache Jakarta Project. http://jakarta.apache.org/jmeter/.[21]	Trabaja bajo Unix (Solaris, Linux, etc.) y Windows (98, NT, 2000). (Barrios, 2007) [21]
Webserver Stress Tool	Herramienta de carga que mide el rendimiento de una aplicación web, (Freeware).[20]	Herramienta de tensión simula números grandes de los usuarios que acceden un websitevia	Paessler. http://www.paessler.com/webstress [20]	El entorno operativo es Ventanas XP/2000/2003/2008/Vista/7 en un workstation con al menos uno 1 GHz

		HTTP/HTTPS. El software puede simular hasta 10.000 usuarios. [20]		procesador, 512 RAM de MB, y una conexión de red rápida. El objetivo webservers, puede ser de cualquier sistema operativo.[20]

1.13.3 Selección de la herramienta para ejecutar las pruebas de Carga y Estrés

Para la automatización de las pruebas de carga y estrés, la herramienta Jmeter es la más indicada para utilizar en el GCF, ya que se destaca por su versatilidad, estabilidad, y por ser de uso gratuito, característica que no poseen las herramientas de Webserver *y* Loadrunner. Jmeter, posee una gran ventaja sobre estas herramientas es libre, cualidad que permite tener libertad para utilizarlo, de estudiar su código fuente, modificarlo. Es una herramienta que contiene un manual de utilización muy fácil de entender y está al alcance de todos los que quieran utilizarla. En cuanto a la licencia con respeto a otras herramientas posee una mayor ventaja, ya que Jmeter es un programa de dominio y uso público. Internamente para mostrar los resultados utiliza varias estructuras: en forma de gráfica, de tabla y el que la utilice podrá interpretar los resultados como le sea más factible. Se encuentra en el repositorio del sistema operativo Linux, lo que posibilita a los estudiantes del GCF tener un fácil acceso a esta herramienta sin necesidad de utilizar internet para poder descargarla.

1.14 Conclusiones parciales del capítulo

A lo largo de este capítulo se describieron todos los conceptos de calidad de software, tipos de pruebas, metodología como guía para realizar el procedimiento, herramientas de automatización para las pruebas de carga y estrés, ayudan en el desarrollo de estas pruebas. En el mundo existen muchas herramientas que han tenido mucho éxito, pero no por eso quiere decir que deban utilizar, ya que muchas no son software libre, elemento muy importante en nuestra universidad, por lo que se realizó una comparación entre las posibles herramientas que se podían utilizar, llegando a la conclusión de que la herramienta Jmeter es la más indicada para utilizar en el GCF.

CAPÍTULO 2 Propuesta de Procedimiento para las Pruebas de Rendimiento de Carga y Estrés

2.1 Introducción

En este capítulo se modela el procedimiento para las pruebas de rendimiento de carga y estrés que se introducirán en el proyecto del Grupo Calidad FORTES (GCF), definiéndose objetivos precisos y alcance, estableciéndose un mejor dominio de su estructura y calidad para su descripción. Se explica cómo se ejecutan dichas pruebas con la ayuda de la herramienta automatizada de Jmeter, los trabajadores (roles), actividades y artefactos que se propone durante las fases por las que transita este procedimiento.

2.2 Propuesta solución

Después de la investigación sobre el alcance y objetivo de las pruebas de rendimiento, actividades propuestas por diferentes compañías y estudios realizados por personas, herramientas utilizadas para ejecutar las pruebas de rendimiento de carga y estrés, se comprueba que con la propuesta de un procedimiento y el apoyo de la herramienta automatizada de Jmeter, la cual, por sus características y ventajas es la más adecuada a utilizar, se le da solución a la situación problemática que existe en el GCF. En este capítulo se detallará el procedimiento con el apoyo: de los documentos del “Modelo de Proceso para el Servicio de Prueba” del GCF del cual se hace referencia a los responsables por cada fase que se define en el procedimiento, el tutorial “Cómo realizar pruebas de carga y estrés con la herramienta Jmeter” y la guía centrada en las pruebas que se le hacen a aplicaciones web, que anteriormente en el epígrafe 1.10 del estudio sobre trabajos realizados de pruebas de rendimiento, se hace una descripción de cada una de las actividades que esta propone y se escoge para desarrollar este procedimiento las siguientes actividades: identificar el entorno de las pruebas, identificar los criterios de aceptación de rendimiento, diseñar las pruebas y ejecutar las pruebas.

2.3 Objetivos de las pruebas de rendimiento de tipo Carga y Estrés

Antes de crear el procedimiento se debe saber los objetivos de las pruebas de carga y estrés que siempre se ejecutan a la misma vez, ya que tienen como propósito de medir y comprobar, el rendimiento de una aplicación web, comparar los resultados esperados con los obtenidos, para identificar las No Conformidades (NC), o sea, los defectos en su funcionamiento y subirlos al Redmine para que sean corregidos por el equipo de desarrollo que realizó la solicitud y termine esa iteración de prueba para dar comienzo a la otra. Específicamente estas pruebas tienen como objetivos:

- Verificar que la aplicación web puede soportar una determinada demanda de trabajo que se especificará por el equipo de desarrollo del sistema a probar.
- Modelar una carga lo más real posible y medir el rendimiento anticipado de aplicación web, o sea, ver cómo se comporta el sistema ante una carga limite en cortos períodos de tiempos. Definir tiempos de respuesta que obtendrá un usuario frente a condiciones normales.
- Identificar si se afecta la integridad del sistema con cierto número de usuarios conectado a la vez en la aplicación web.
- Identificar el número máximo de usuarios concurrentes que pueden acceder a determinada página o transacciones de por segundo que la aplicación web puede soportar.
- Identificar las páginas que responden más lenta y las más rápidas.

2.4 Elaboración del procedimiento para las pruebas de rendimiento de Carga y Estrés

En este procedimiento está estructurado por fases, en las cuales se describen todas las actividades que deben realizar por los estudiantes del GCF, basándose en los principales escenarios de los caso de pruebas del proyecto que solicite el servicio de pruebas de rendimiento de carga y estrés.

2.4.1 Objetivos del procedimiento

- Describir un procedimiento para ejecutar las pruebas de rendimiento específicamente carga y estrés de una aplicación web que solicite el servicio del GCF.
- Listar roles, actividades y los principales artefactos que se deben entregar durante el procedimiento de las pruebas de carga y estrés.
- Diseñar correctamente el entorno necesario para las pruebas de carga y estrés con la herramienta automatizada de Jmeter.

2.4.2 Alcance del procedimiento

Este procedimiento se lleva a cabo en aplicaciones web de los proyectos de FORTES o interesados. Con la ayuda de la herramienta automatizada de Jmeter y los casos de pruebas se probará el rendimiento de dicha aplicación. Se debe definir el módulo y el proyecto que será probado.

2.4.3 Ciclo de vida de las pruebas de rendimiento

Las pruebas de rendimiento de tipo carga y estrés se deberían realizar desde la primera fase del ciclo de vida del sistema. Logrando reducir desde etapas tempranas los posibles fallos en la aplicación evitando así que se arrastren a fases posteriores.

Con las pruebas de rendimiento se obtendrán resultados lo más aproximado a la carga real que pueda soportar el sistema, compararlos, y llegar a conclusiones sobre el funcionamiento del sistema. Estas pruebas deben generarse en un entorno estable, lo más similar posible al entorno final de producción, o sea, al lugar donde será explotado la aplicación por el usuario final.

2.4.4 Roles por actividades de las fases del procedimiento

Jefe de Proyecto

- Garantizar la presencia de un representante del Equipo de Desarrollo del proyecto que se va a probar.

Jefe del Laboratorio de Pruebas

- Garantizar la presencia de un representante del Equipo de Desarrollo del proyecto que se va a probar.
- Elaborar la estrategia para las pruebas de rendimiento.
- Capacitar de los Miembros del Equipo de Prueba para ejecutar la herramienta automatizada.
- Emitir acta de liberación por parte de FORTES.

Los Miembros del Equipo de Prueba

- Identificar entorno de las pruebas para realizar las pruebas de rendimiento.
- Identificar requisitos de aceptación de las pruebas de rendimiento.
- Diseñar las pruebas para realizar las pruebas de rendimiento.
- Validar el entorno de prueba.
- Validar las pruebas de rendimiento de tipo carga y estrés.
- Ejecutar las pruebas de rendimiento de tipo carga y estrés.

Diseñadores de Pruebas

- Identificar entorno de las pruebas para realizar las pruebas de rendimiento.
- Identificar requisitos de aceptación de las pruebas de rendimiento.
- Diseñar las pruebas para realizar las pruebas de rendimiento.
- Capacitación de los Miembros del Equipo de prueba para ejecutar la herramienta automatizada.

Ingeniero de Componente

- Capacitación de los Miembros del Equipo de prueba para ejecutar la herramienta automatizada.

Ingenieros en Prueba de Sistema

- Validar el entorno de prueba.
- Validar las pruebas de rendimiento de tipo carga y estrés.
- Ejecutar las pruebas de rendimiento de tipo carga y estrés.

2.4.5 Fase para desarrollar el procedimiento

El procedimiento para las pruebas de rendimiento de carga y estrés debe comprender al menos 6 fases para su realización.

- **Planificación de las pruebas**
- **Capacitación para ejecución del servicio**
- **Ejecución de las pruebas**
- **Cierre de las pruebas**

A continuación se dará cada detalle de estas fases:

2.4.5.1 Planificación de las Pruebas

Objetivos

- Recopilar información del proyecto que se le acepta la solicitud para realizar las pruebas de carga y estrés. La información incluye arquitectura del sistema y cómo el usuario interactúa con el sistema.

- Crear una estrategia para las pruebas de carga y estrés.

Descripción de la planificación de las pruebas

Una vez que se acepte la solicitud del proyecto que se le hará las pruebas de carga y estrés, se procede a la planificación de las actividades necesarias para realizar las pruebas de carga y estrés.

En esta etapa participan el Jefe de Proyecto y el Jefe del Laboratorio de Pruebas como miembro de la dirección del GCF. Ambos tienen la responsabilidad de llegar a acuerdos para desarrollar las siguientes actividades:

- Actividad 1. Garantizar la presencia de un representante del Equipo de Desarrollo del proyecto que se va a probar.

El Jefe del Laboratorio de Pruebas tiene la responsabilidad de:

- Actividad 2. Elaborar la estrategia para las pruebas de rendimiento.

Los Miembros del Equipo de Prueba y Diseñadores de Pruebas tienen la responsabilidad de:

- Actividad 3. Identificar entorno de las pruebas para realizar las pruebas de rendimiento.
- Actividad 4. Identificar requisitos de aceptación de las pruebas de rendimiento.
- Actividad 5. Diseñar las pruebas para realizar las pruebas de rendimiento.

Descripción de las actividades:

Actividad 1. Garantizar la presencia de un representante del Equipo de Desarrollo del proyecto que se va a probar

Se hace una reunión de inicio, donde define los objetivos y alcance de las pruebas de rendimiento que realizarán a la aplicación de la solicitud. Tanto Jefe de Proyecto como Jefe del Laboratorio de Pruebas deben garantizar la presencia de un desarrollador del proyecto que se va a probar, para en caso de que los Miembros del Equipo de Prueba tengan alguna duda a la hora de reportar una NC, eviten que estas no procedan.

Actividad 2. Elaborar la estrategia para las pruebas de rendimiento

El Jefe del Laboratorio de Pruebas elabora la propuesta de Estrategia de Prueba que será discutida en la reunión de inicio. En esta Estrategia de Prueba se incluyen todos los factores para crear las condiciones

necesarias para iniciar las pruebas ejemplo: horarios para realizar las pruebas que no afecten las clases de los estudiantes, definir los objetivos de las pruebas de rendimiento y otros elementos organizativos que pueden propiciar un mejor ajuste en la ejecución de las pruebas de rendimiento.

En la reunión de inicio de prueba se debe tener en cuenta, si el laboratorio está en condiciones para ejecutar las pruebas de carga y estrés, recopilar información sobre el proyecto que se esté probando, para saber si se necesita más documentación referente al sistema como: caso de pruebas, si la aplicación funciona correctamente.

Actividad 3. Identificar entorno de las pruebas para realizar las pruebas de rendimiento

Análisis de la documentación:

Antes de empezar a probar la aplicación se debe tener una serie de documentos como: los horarios de producción, la Estrategia de Prueba, el cronograma de prueba, el acta de reunión de inicio de prueba, los requisitos funcionales y no funcionales que debe poseer la aplicación, para que el Equipo de Prueba estudie como debe funcionar el sistema a probar y en base a esto pueden interpretar mejor los resultados e identificar posibles cuellos de botellas que ayuden a los desarrolladores a mejorar la calidad de sus aplicaciones. Además se debe tener en cuenta los casos de pruebas, manual de usuario del software a utilizar para la realización de las pruebas.

Los Diseñadores de Prueba deben identificar el entorno para las pruebas de carga y estrés, deben recoger información sobre el sistema que se somete a prueba.

Para las pruebas de rendimiento es fundamental identificar los recursos que permiten definir funcionalidades del sistema y se puede hacer a través de: entrevistas con las personas interesadas, expectativas del cliente, requisitos y caso de pruebas, aplicaciones similares que se hallan puesto a prueba alguna vez.

Primeramente se deben capturar las funciones de la aplicación o procesos del negocio para identificar requisitos de aceptación de rendimiento. Que servirá para evaluar la carga de trabajo y validar estos requisitos.

Segundo debe capturar las actividades fundamentales de los usuarios, debido a que es imposible simular todas las tareas que el usuario realiza en una aplicación, en la actividad del diseño de prueba se explicará más detalladamente, es necesario identificar cuáles actividades son más importantes para simular, o sea los escenarios de pruebas a utilizar para probar el sistema.

Y por último captura la arquitectura lógica y física del sistema a probar, esto se refiere a la relación que existe entre la aplicación, el hardware y el software.
Para esto el Jefe del Laboratorio de Pruebas debe enviarle al Jefe de Proyecto una plantilla para que exponga los requerimientos necesarios para realizar las pruebas de rendimiento de carga y estrés ejemplo:

Sobre la aplicación que se somete a pruebas:

Hardware:

- Datos sobre la velocidad y la capacidad del servidor de la aplicación.
- Distancia del servidor al laboratorio de pruebas.

Software:

- Datos de software de la aplicación.
- Tipo de conexión y velocidad.

Para interpretar los resultados:

- Cantidad de usuarios concurrentes que necesita soportar la aplicación.
- Tiempo de respuesta en situaciones de carga de una transacción.

Caso de prueba

El proyecto debe realizar el caso de prueba donde describa los escenarios significativos que serán sometidos a la prueba. Las actividades más importantes que se realizan en el negocio de la aplicación, las rutas que más acceden los clientes el sistema. Para más detalles ver la actividad de diseño de prueba.

No necesariamente es imprescindible conocer la arquitectura física del sistema puesto a prueba para realizar una estimación sobre las pruebas de rendimiento y la calidad de las mismas, de manera similar a como se hacen las pruebas de caja negra se pueden hacer las pruebas de rendimiento de carga y estrés desde el punto del usuario y los tiempos de respuesta que estos van a recibir según los casos de pruebas que sean analizados.

Actividad 4. Identificar requisitos de aceptación de las pruebas de rendimiento

Los Miembros del Equipo de Prueba deben identificar y establecer objetivos de las pruebas de rendimiento, identificar y capturar las metas de uso de los recursos y los umbrales que son los valores máximos aceptados para los indicadores identificados para el proyecto, por lo general se especifica en términos de tiempo de respuesta, el rendimiento (transacciones por segundo), y la utilización de los niveles de recursos, los niveles de utilización de recursos incluyen la capacidad de procesador, memoria, y la red que consume una aplicación. Todos los proyectos tienen características diferentes y realizan las pruebas de rendimiento por distintos propósitos.

Consideraciones para determinar los objetivos pruebas de rendimiento:

- Los objetivos de las pruebas de rendimiento representan el punto de partida de la validación de resultados y actividades de verificación.
- Los objetivos se relacionan con las necesidades del negocio, por lo que representan escenarios reales de negocios de cliente reales.

Los objetivos de las pruebas de rendimiento se suele especificarse en términos de tiempos de respuesta, el rendimiento (transacciones por segundo), y la utilización de los niveles de recursos y por lo general se centran en los indicadores que pueden estar directamente relacionados con la satisfacción del usuario.
Antes de determinar los objetivos para los cuales se le hacen las pruebas de rendimiento a una aplicación los Miembros del Equipo de Prueba deben determinar los escenarios críticos, o sea los de uso más frecuente, y así determinar qué requisitos se debe evaluar durante la ejecución de la prueba.
Los Miembros del Equipo de Prueba deben tener bien claro los umbrales que como se describe anteriormente son los valores máximos aceptados para los indicadores identificados para el proyecto.
Ejemplo de umbrales en cuanto a los recursos durante la prueba son:
El rendimiento de un servidor se degrada, si el uso del procesador regularmente supera el 80 por ciento.
Los Miembros del Equipo de Prueba deben tener bien claro los tiempos de respuesta de usuarios finales, que es la medida de respuesta que le da la aplicación puesta a prueba a una solicitud de un cliente, o sea, el tiempo que pasa entre una petición inicial y la descarga completa de la respuesta. Los objetivos de rendimiento son criterios que los Miembros del Equipo de Prueba deben conocer para poder reportar las NC antes de la liberación del producto.
Ejemplo de tiempo de respuesta:

Si el objetivo de un tiempo de respuesta de determinada transacción es de 3 segundos, pero el tiempo de respuesta real es de 3,3 segundos, esto se reportaría como una NC y por tanto retrasaría la liberación del producto.
Algunos aspectos a tener en cuenta con respecto a los recursos utilizados durante la realización de las pruebas:

Configuración de la red

La infraestructura de la red debe de soportar el tráfico generado durante la simulación para no desvirtuar el resultado de las pruebas ejemplo tener en cuenta el ancho de banda, el número de ordenadores conectados en la red y los switch.

Hardware

La generación de cada usuario virtual ocupa una cantidad de memoria que viene determinada por la herramienta en sí, el tamaño del script de pruebas y por el protocolo que se usa en emulación de carga del Software. En el caso de esta investigación las pruebas se realizan con la herramienta Jmeter, que para poder ejecutar esta herramienta se debe instalar una Máquina Virtual Java 1.3 o superior.

Servidores

Los servidores utilizados en el entorno de pruebas han de ser lo más parecidos posible a los servidores y la arquitectura utilizados finalmente en el lugar donde se encuentra el cliente. Cuanto más similares sean, mayor será la exactitud de los resultados obtenidos.
Los tiempos de respuestas tienen mucho que ver con los servidores que se utilizan, para ver cuánto se tarda una petición en ser atendida según el servidor, se deben hacer pruebas con éste, pero con diferentes cargas, pueden ser primero con una carga baja del servidor. Es decir, se verá cuánto tarda en responder cuando no existen demasiadas peticiones en el servidor.
Puede ser con una carga media del servidor. Aquí se pone una carga al límite de la capacidad del servidor y se verá como es ahora el tiempo medido. Y por último será sobrecargando el servidor con más peticiones de las que puede servir. Se verá cómo se disparan las peticiones erróneas ya que algunas conexiones no se podrán establecer.

A modo de resumen los criterios de aceptación se basan en las siguientes actividades:

- investigar las necesidades del usuario final. Que el usuario se sienta cómodo con la aplicación, a

este no le interesa las pruebas de rendimiento, ni la cantidad de datos que puede procesar la aplicación o que tan eficiente es el uso de sus recursos, si no en cuán rápido responde el sistema ante una transacción o si el acceso a sus páginas le resultan lentas o no. O sea, representa un reto complacer al usuario y con las pruebas de rendimiento puede lograrse.

- Recoger los requisitos del negocio. La lógica del negocio, que actividades se debe tener en cuenta como: si en un lanzamiento inicial a un número determinado de usuarios es aceptable.
- Determinar los requisitos técnicos. Es los que se refiere al uso de la tecnología como: hardware, software y servidores.

Para hacer Casos de pruebas con la herramienta de automatización. Jmeter

Ejemplo la plantilla desarrollada para casos de prueba de rendimiento usando la herramienta Jmeter. (Ver Anexo 1)

Actividad 5. Diseño de las pruebas para realizar las pruebas de rendimiento.

El principal objetivo de esta actividad para el diseño de las pruebas de rendimiento, es simular la navegación del usuario en la aplicación lo más real posible. Esta actividad la realizan los Miembros del Equipo de Prueba y Diseñadores de pruebas.

Las cargas de trabajo que se determinen para probar la aplicación deben representar un escenario de producción del mundo real, y para esto se debe tener en cuenta:

Los requisitos de aceptación de las pruebas de rendimiento como se menciona en la actividad anterior, se basan en los requisitos de usuario, del negocio y los técnicos. Los escenarios más importantes dentro del caso de prueba pueden ser: los casos de pruebas críticos dentro del negocio, los de uso de interés técnico que hagan que se utilicen más los recursos tecnológicos del sistema, el número de usuarios concurrentes, que realicen mayor cantidad de peticiones sobre este caso de prueba, la cantidad de transacciones por unidad de tiempo que se realizan al servidor. Para determinar estos escenarios se debe tener en cuenta contratos, caso de uso y experiencia de otras aplicaciones similares que hayan sido sometidas a estas pruebas.

Se debe determinar los principales escenarios dentro de un caso de prueba para la generación de las pruebas de carga y estrés, ya que no se puede simular todas las actividades realizadas por los usuarios debido a que mientras mayor sea las rutas de navegación, mayor será las peticiones al servidor y el script

generado por la herramienta Jmeter, lo que dificulta la observación y el análisis de los resultados obtenidos durante las pruebas.

Determinar rutas de navegación de los escenarios seleccionados, esta es una tarea difícil ya que no se puede predecir todas las rutas que los usuarios frecuentan en una aplicación, los que se debe hacer es determinar la rutas que afecten más el rendimiento y que se encuentren en varios escenarios claves.

Utilizar los registros (logs) para realizar una estimación sobre el tráfico de un sitio. La demanda de trabajo bajo la cual va a estar sometido el sistema y las características de la misma sólo puede conocerse con exactitud cuando el sistema ha estado o está sometido a explotación dando un servicio real. Cuando esto no ocurre se debe realizar estimaciones al respecto o recurrir a los datos proporcionados por algún sistema anterior o lo suficiente para que estos datos así obtenidos nos resulten útiles. La caracterización de la carga de trabajo, sea esta real o estimada, es un requisito indispensable para plantear las pruebas de rendimiento.

Cuando se dispone de datos procedentes de una anterior instancia de la aplicación que ha pasado por un período real de explotación. Para conocer la información referente a esta aplicación se debe usar los ficheros de logs dejados por el sistema y las numerosas utilidades existentes para procesarlos y extraer datos útiles a partir de los mismos. Datos sobre las peticiones de cada página, desglosadas por fecha y hora del día, la direcciones IP de los visitantes, sistemas operativos utilizados, navegadores, páginas visitadas por períodos que incluye los tipos de archivos (JPG, CCS, PDF etc.), el abandono por parte del usuario de una página, tiene que ver con el tiempo que esperan en que esta sea cargada y el la retirada del sitio, esto trae como consecuencia negativas en los resultados de las pruebas de cargas.

Artefactos que se generan:

Los casos de pruebas, manual de usuarios sobre la herramienta Jmeter, plantilla de requerimiento de la aplicación que se somete a prueba, los caso de pruebas definidos para probar.

2.4.5.2 Capacitación para ejecución del servicio de prueba de rendimiento

Objetivos

- Preparar a los miembros del Equipo de Prueba para ejecutar la herramienta de automatización de prueba de rendimiento.

Descripción capacitación para ejecución del servicio de prueba de rendimiento

Una vez que se tenga bien claro cuáles son los escenarios principales que se van a probar, o sea, las páginas principales que visitan los usuarios en el sistema que se probará, con la herramienta de automatización Jmeter se graban los escenarios, y esta a su vez generan los script que serán la base para observar los resultado con todos los parámetros que se definieron en la fase de planificación.

En esta etapa participan el Jefe del Laboratorio de Pruebas**,** el Diseñador de Pruebas o el Ingeniero de Componentes, tienen la responsabilidad preparar a los Miembros del Equipo de Prueba para ejecutar la herramienta automatizada que ejecutará las pruebas de rendimiento esto se refleja en la siguiente actividad:

- Actividad 6. Capacitar los Miembros del Equipo de prueba para ejecutar la herramienta automatizada.

Descripción de la actividad:

Actividad 6. Capacitación de los Miembros del Equipo de Prueba para ejecutar la herramienta automatizada.

La capacitación será impartida por el Jefe del Laboratorio de Pruebas, el Diseñador de Pruebas o el Ingeniero de Componentes para que los Miembros del Equipo de Prueba estén preparados para ejecutar la herramienta Jmeter y se realice las pruebas de manera eficiente y con la calidad requerida.

JMeter es una herramienta de software libre que ofrece la posibilidad de realizar pruebas de carga sobre diferentes aspectos de una aplicación web. JMeter es fácilmente configurable y permite conocer los tiempos de respuesta experimentados por una aplicación cuando se tiene un número de usuarios y el número real de transacciones procesadas por unidad de tiempo. Los resultados pueden ser visualizados en diferentes formatos, lo cual facilita las labores de análisis por los Miembros del Equipo de Prueba.

2.1Tabla de resultados generales

Petición	# Muestra	Media	Mediana	Linea de 90%	Mix	Max	%Error	Rendimiento	KB/S

La herramienta jmeter tiene muchas variedades para mostrar los resultados de las pruebas, pero el informe agregado es uno de los elementos que esta posee, que permite interpretar, analizar y comparar los valores obtenidos. Las conclusiones sobre los resultados se obtendrán con el # de muestras que representan grupo de hilos que se quiera conectar a la aplicación, es decir la representación de la

cantidad de usuarios conectados, el % de error que representa las páginas que no se cargaron satisfactoriamente, y el rendimiento que representa el tiempo total que demoraron en cargarse todas las peticiones (páginas hilos) al servidor que se le realiza la prueba. El conjunto de estas variables muestran cómo se comporta la aplicación con respecto al rendimiento total de todas las peticiones dígase todo los escenarios que se introducen en la prueba que generan un determinado script.

El plan de pruebas que se crea en la herramienta Jmeter 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.

Con los Samplers se pueden realizar peticiones a determinado servidor, también pueden ser configurados a través de los elementos de configuración.

A continuación se hará referencia a algunos los elementos de configuración:

- IfController: permite decidir si realizar o no una petición http en función de una condición.
- Grupo de Hilos: definen la cantidad de usuarios que se van a simular en la prueba.
- Período de subida (en segundos): donde se especifica el periodo intermedio en segundos en los cuales va a ir iniciándose cada uno de los usuarios.
- 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.

Para obtener información más detallada sobre estos elementos que se utilizan como parámetro en la ejecución de la herramienta Jmeter ver el tutorial “Cómo realizar pruebas de carga y estrés con Jmeter”.

2.4.5.3 Ejecución de las pruebas

Objetivos

- Ejecutar los scripts que se generaron de acuerdo a los escenarios seleccionados.
- Reportar las No conformidades encontradas durante la ejecución de las pruebas de carga y estrés.

Descripción de la ejecución de las pruebas de rendimiento

La ejecución de las pruebas la realizan los Ingenieros en Prueba de Sistema y Miembros del Equipo de prueba esta actividad se produce con la ayuda de la herramienta de automatización de Jmeter, que es la que genera los scripts de acuerdo a los escenarios especificados en la fase de planificación.
Los Miembros del Equipo de prueba realizan la actividad de identificar las NC encontradas en el funcionamiento de aplicación. Después de reportadas todas las no conformidades el Jefe del Laboratorio de Pruebas debe revisar y confirmar las NC elaboradas y elabora un documento llamado parte diario donde hace un resumen del estado de las pruebas. El Jefe del Laboratorio de Pruebas comunica al Jefe de Proyecto si la prueba se realizó o no de acuerdo con los criterios de criticidad establecidos por Calisoft.
En caso de haber fallado la prueba el Equipo de Desarrollo debe realizar nuevamente la solicitud de la prueba, de lo contrario deben darle respuesta a las NC identificadas para dar comienzo a la otra iteración.
Durante la actividad de respuesta a las NC el Jefe del Laboratorio de Pruebas verifica la corrección de las NC, y mediante una reunión se analizan las NC que no procedieron, se ratifican y se modifican los objetivos de las pruebas. Una vez realizado este proceso se pasa a la siguiente iteración.
Actividades que se realizan durante la ejecución de rendimiento:

- Actividad 7. Validar el entorno de prueba.
- Actividad 8. Validar las pruebas de rendimiento de tipo carga y estrés.
- Actividad 9. Ejecutar las pruebas de rendimiento de tipo carga y estrés.

Descripción de las actividades:

Actividad 7.Validar el entorno de prueba

Antes de ejecutar las pruebas es importante que los Miembros del Equipo de Pruebas verifiquen, que el entorno de las pruebas coincide con la configuración y el diseño de la prueba que se realizó en la fase de planificación.

Actividad 8. Validar las pruebas de rendimiento de tipo Carga y Estrés

Los Miembros del Equipo de Pruebas deben verificar que la simulación de la carga refleje con precisión el diseño de la prueba.

Actividad 9. Ejecutar las pruebas

Aunque el proceso de prueba es totalmente dependiente de una herramienta, del entorno y del contexto del proyecto se debe tener en cuenta algunas tareas generales: Después de cada prueba, los Miembros del Equipo de Prueba deben subir al Redmine las NC y agregar comentarios donde se incluyen fallos parte del equipo de prueba, fallos por parte de la aplicación, problemas en la red.

Después de la ejecución final de las pruebas Miembros del Equipo de Prueba deben comprobar haber reportado correctamente todas las NC correctamente antes de desmontar todo el entorno de las pruebas.

Para ejecutar las pruebas de carga y estrés se debe tener bien claro:

- Cuáles son los requisitos funcionales y no funcionales de la aplicación y bajo qué condiciones se ejecuta la aplicación.

Elegir las variables de entrada y salida con las que se va a probar el sistema, para poder interpretar posibles resultados con los obtenidos y poder dar una evaluación. Las variables son número de usuarios, número de transacciones, tiempo de respuesta y otras variables que se van definiendo a medida que hagan falta cuando se esté haciendo la prueba con la herramienta Jmeter.

A modo de resumen la ejecución de rendimiento prueba consiste en actividades tales como la validación de entornos de prueba, ejecución de scripts generados por los escenarios definidos en el diseño de las pruebas, ejecutar la prueba, y la generación de los resultados de la prueba.

Artefactos generados:

Parte diario, NC, Casos de Pruebas.

2.4.5.4 Cierre de las pruebas

En esta última fase se hace el análisis y la interpretación de los resultados una vez ejecutado los casos de pruebas y la herramienta automatizada, se analizan tanto las NC que proceden como las que no proceden, lo que permite determinar si el trabajo se hizo con calidad. Se hace el cierre de las pruebas donde el Jefe del Laboratorio de Pruebas emite un acta de liberación por parte de FORTES que es recibida por Jefe de Proyecto para que pueda realizar la solicitud de prueba a Calisoft.

Artefactos generados:

Acta de liberación

2.5 Conclusiones parciales del capítulo

En este capítulo se describió de forma detallada la propuesta de procedimiento para la ejecución de las pruebas de rendimiento de carga y estrés con la mayor calidad posible. Cada actividad que se define durante las fases del procedimiento, sirve de guía a los estudiantes del GCF, para realizar pruebas de forma estructurada a las aplicaciones que realicen una solicitud. En estas actividades se detalla todas las características referentes a las pruebas de rendimiento de tipo carga y estrés mostrando cuan efectiva e importante puede ser para una aplicación web realizarle las pruebas no funcionales. Además este procedimiento ofrece una organización en cuanto a los responsables que realizan cada actividad que se describe en el mismo, lo que ayuda a adoptar paulatinamente las pruebas de rendimiento de carga y estrés al GCF y al ciclo de vida de desarrollo de la aplicación puesta a prueba, contribuyendo así con la mejorar las capacidades de la aplicación en cuanto al uso de recursos y cantidad de conectados concurrentemente. En el procedimiento se hace referencia a requisitos que se deben tener en cuenta a la hora de interpretar los resultados, lo cual sirve de base para el siguiente capítulo donde se aplica el procedimiento a un proyecto.

CAPÍTULO3 Aplicación del Procedimiento para las Pruebas de Carga y Estrés

3.1 Introducción

En este capítulo se aplica el procedimiento descrito anteriormente, observando y evaluando los resultados obtenidos durante la ejecución de las pruebas de rendimiento de tipo carga y estrés con la herramienta automatizada Jmeter. Con la ayuda de los diseños de casos de pruebas y los requisitos establecidos de la aplicación web desarrollada en el proyecto Alfaomega, se obtendrá una serie de datos necesarios para identificar, interpretar e informar el rendimiento de dicha aplicación.

3.2 Aplicación del procedimiento para las pruebas de rendimiento de carga y estrés en el proyecto Alfaomega.

Este procedimiento se le aplica a la plataforma ZERA específicamente a un componente que permite gestionar y supervisar las tareas de los estudiantes, de manera que el docente pueda darle seguimiento al estudiante en el proceso de enseñanza – aprendizaje.

3.2.1 Objetivo del procedimiento

Identificar las No Conformidades (NC) que se observen durante la ejecución de las fases de este procedimiento en la aplicación del proyecto Alfaomega. Detectar errores en el rendimiento de dicha aplicación con respecto a las expectativas del cliente final (en este caso son las escuelas que usen este software), con las características del proyecto y los recursos que se usen en la ejecución de las pruebas de carga y estrés.

3.2.2 Alcance del procedimiento

Se realizan las pruebas de rendimiento de carga y estrés a un componente del proyecto Alfaomega desarrollada en symfony. Se observa y se identifican fallos en el rendimiento de dicha aplicación con la ayuda de los principales escenarios de los casos de pruebas que se definan como críticos y la ejecución de la herramienta automatizada Jmeter.

3.2.3 Fase para aplicar el procedimiento

Para aplicar el procedimiento a este proyecto se basa en la fase de planificación de las pruebas, ejecución de las pruebas y cierre de las pruebas, donde se describe todos los elementos necesarios de las

actividades que se definieron en el capítulo anterior para lograr ejecutar las pruebas de carga y estrés de manera satisfactoria.

3.2.3.1 Planificación de las pruebas

Actividad 1. Identificar el entorno de las pruebas

El proyecto Alfaomega contiene los documentos necesarios para ejecutar las pruebas de carga y estrés como: manuales de usuarios, casos de uso, casos de pruebas, un manual de ayuda para navegar por el sistema y documentos de especificación de requerimientos**.**

Primeramente se capturan las funciones de la aplicación o procesos del proyecto para identificar criterios de aceptación de rendimiento, que se hace sobre la base de toda la documentación de este proyecto.

Segundo obtener una visión rápida de lo que hace el módulo que se va a probar:

Este componente o este módulo, está inmerso dentro de la Plataforma Educativa ZERA, tienen como principal objetivo permitirles a los profesores de una forma fácil y flexible, la gestión, asignación y visualización de tareas docentes a estudiantes, así como la satisfacción de que los estudiantes resuelvan la misma sin ningún problema al interactuar con dichos componentes.

Se escoge las principales actividades que el usuario realiza en el sistema, debido a que es imposible simular todas las tareas que realiza el usuario en la aplicación, ya que es recomendable hacer las pruebas sin mucha navegación del sistema, ya que se hace tedioso analizar las peticiones que se le hacen al servidor, además que la navegación en la plataforma ZERA es muy complicada, por lo que se escoge el siguiente caso de pruebas:

Caso de prueba: Gestionar Softarea (Incluir Softarea)

Para identificar la arquitectura lógica y física se llena la plantilla con los siguientes datos:

Sobre la aplicación que se somete a pruebas:

Hardware:

Datos sobre la velocidad y la capacidad del servidor de la aplicación.

Servidores Locales

En las escuelas en dependencia de la cantidad de usuarios pudiese elegirse usar un solo servidor para la web, Base de Datos, o de forma separada en dos: 1 para la web y otro para la BD. En este caso las características del servidor en la UCI es la siguiente:

Tabla 3.1. Datos sobre el hardware de la aplicación que somete a prueba.

Tipo de Servidor	Cantidad	Especificaciones
Servidores Web	1	1 Procesador CoreDuo 2GB de RAM 120 GB de HDD mínimo, aunque puede ser configurable de acuerdo a la cantidad de contenidos que se vayan contratando.
Servidor de Base de Datos	1	1 Procesador CoreDuo 2GB de RAM 120 GB de HDD mínimo, aunque puede ser configurable de acuerdo a la cantidad de contenidos que se vayan contratando.
Ancho de banda	10/100 Mbps UTP 5 LAM	

Distancia del servidor al laboratorio de pruebas es de 1000m.

Software:

- Datos de software de la aplicación.
- Tipo de conexión y velocidad.

Tabla 3.2. Datos sobre el software de la aplicación que somete a prueba.

Software	Datos

Sistema Operativo	Windows XP Servi Pack 2 o Linux.
Gestor de Base de Datos	PostgreSQL 8.4.
Antivirus	Kaspersky Workstation 6.0
Máquina Virtual Java	Virtual Machine 1.3 o superior

Para interpretar los resultados:

- Cantidad de usuarios concurrentes que necesita soportar la aplicación:

  Entre 50 y 100 usuarios.

- Tiempo de respuesta en situaciones de carga de una transacción:

  Entre 2 y 5 segundo.

Caso de prueba

Para realizar las pruebas de carga y estrés el desarrollador del módulo Gestionar Softarea (Incluir Softarea) del proyecto Alfaomega, define como escenarios más críticos: EC1, EC2 y EC6.

Para la aplicación del procedimiento de pruebas de rendimiento diseñado, en él GCF se identifica un entorno lo más similar posible al laboratorio de producción del proyecto Alfaomega. Con las siguientes características:

Tabla 3.3. Características del Hardware del laboratorio de prueba.

Hardware	**Datos**
microprocesador	Intel(R) Core(TM)2 Duo CPU E4500 a 2.20 GHz
Memoria RAM	1 GB (PC Clientes) 1GB (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)

Tabla 3.4. Características del Software del laboratorio de prueba.

Software	**Datos**

Sistema Operativo	Windows XP Servi Pack 2 o Linux.
Gestor de Base de Datos	PostgreSQL 8.4.
Antivirus	Kaspersky Workstation 6.0
Máquina Virtual Java	Virtual Machine 1.3 o superior
Software de prueba	JMeter Versión 2.3.4

Actividad 2. Requisitos de aceptación de rendimiento

Los requisitos de rendimiento suelen identificarse por las características del proyecto y las necesidades del usuario. Para esta aplicación se definen los siguientes requisitos:
La máxima carga de usuarios que debe soportar la aplicación es en dependencia de la escuela donde se instale la plataforma ZERA, pero en la UCI de soportar entre 50 y 100 usuarios conectados.
Para realizar esta prueba se toma como muestra 50 usuarios conectados concurrentemente. Con 50 usuarios conectados, la aplicación debe mantener su integridad, no se debe distorsionar ninguna imagen, gráfico y que se deje de ver algún párrafo.

Tiempos de respuesta. Ejemplo. Para EC2 Selecciona un Recorrido Dirigido, el sistema debe mostrar las listas de evidencias y ejercicios en 3 segundos.

Utilización de recursos. La cantidad de recursos que la aplicación está consumiendo en términos de procesador, memoria, disco de entrada y salida y la red. Ejemplo el rendimiento de un servidor se degrada, si el uso del procesador regularmente supera el 80 %.

Actividad 3. Diseño de las pruebas

De acuerdo como se escogió en la actividad de identificar los casos de pruebas a simular, se debe especificar los escenarios que contienen las rutas de navegación más frecuentadas.
Para determinar estas rutas se le preguntó al desarrollador del módulo de Softarea cuáles escenarios consideraba que fueran más usadas por los clientes. Y como respuesta se obtuvo el EC1, EC2 y EC6 del caso de prueba Gestionar Softarea (Incluir Softarea). (Ver anexo 2)

Para calcular el total de los resultados esperados de cada transacción que se hace en los escenarios del caso de prueba Gestionar Softarea (Incluir Softarea) se descarga el pluying firebug 1.6.1 que representa una extensión del Mozilla Firefox, que analiza el código fuente de las aplicaciones que se ejecuten en este navegador y muestra el tiempo de respuesta de determinada petición que se haga al servidor.
Para las siguientes pruebas se analiza el parámetro de rendimiento que ofrece la herramienta Jmeter. Y se tendrá en cuenta la capacidad del procesador de la máquina donde se realiza la prueba.

Tabla 3.5. Carga de trabajo y resultados esperados de los escenarios

Identificador del escenario	Escenarios de la sección	Carga de trabajo	Descripción	Resultados esperados	Resultado obtenidos de la prueba
EC1	Seleccionar la opción incluir una nueva tareas	50	Incluir una nueva tarea dirección del módulo http://10.128.50.59:5800/bachelor.php/eTask/home/tabs/1	La aplicación debe responder en un tiempo menor de 23,2 seg.	La aplicación responde en un tiempo (parámetro de rendimiento de Jmeter) de 22.3 seg.
EC2	Seleccionar un Recorrido dirigido	50	Selecciona el o los estudiantes involucrados.	La aplicación debe responder en un tiempo menor a 23,3 seg.	La aplicación responde en un tiempo (parámetro de rendimiento de Jmeter) de 22.3 seg.
EC6	Selecciona la opción de Crear Recorrido Dirigido.	50	Tiene la opción de incluir, ver, modificar y eliminar un Recorrido Dirigido.	La aplicación debe responder en un tiempo menor a 27,1 seg.	La aplicación responde en un tiempo (parámetro de rendimiento de Jmeter) de 22.3 seg.

3.2.3.2 Ejecución de las pruebas

Una vez definido los casos de prueba con los escenarios y rutas de navegación principales se procede a ejecutar la herramienta Jmeter la cual genera los script que permiten mostrar toda la información referente al rendimiento de la aplicación del proyecto Alfaomega.
Se ejecuta la herramienta Jmeter grabando los EC1, EC2 y EC6 del caso de prueba Gestionar Softarea (Incluir Softarea). (Ver anexo 2), mediante peticiones http, por el puerto 8080 usando el navegador Mozilla Firefox, con la siguiente dirección url: http:// 10.128.50.59:5800

Análisis e interpretación de los resultados

Con respecto al rendimiento de la aplicación para los EC1, EC2 y EC6 no se encontraron ninguna NC, esto se evidencia en las siguientes tablas que contienen los resultados generados por el informe agregado que se escoge como plan de prueba de la herramienta Jmeter para mostrar los valores de las pruebas.

Tabla 3.6. Resultados EC1: Selecciona la opción de incluir una nueva tarea.

Petición	# Muestra	Media	Mediana	Linea de 90%	Mix	Max	%Error	Rendimiento	KB/S
# de usuarios	50	9	0	30	0	63	0.0%	1,5 sec	190,7
total	1700	2013	15	1245	0	28745	7.4	22,3	30246,6

Tabla 3.7. Resultados EC2: Seleccionar un Recorrido dirigido.

Petición	# Muestra	Media	Mediana	Linea de 90%	Mix	Max	%Error	Rendimiento	KB/S
# de usuarios	50	9	0	30	0	63	0.0%	1,5 sec	192,7
total	1700	2013	15	1245	0	28745	7.4	22,3	30246,6

Tabla 3.8.Resultados EC6: Selecciona la opción de Crear Recorrido Dirigido.

Petición	# Muestra	Media	Mediana	Linea de 90%	Mix	Max	%Error	Rendimiento	KB/S
# de usuarios	50	8	0	60	0	64	0.0%	1,5 sec	193,7
total	1700	2013	15	1245	0	28745	7.4	22,3	30246,6

Significado de los elementos del informe agregado

- # Muestras: Cantidad de páginas (hilos) que simulan la cantidad de usuarios, que están interactuando con el sistema desde la misma URL.
- Media: Media de páginas que se cargaron satisfactoriamente.

- Mediana: Tiempo promedio que ha tardado en cargarse las páginas.
- Línea de 90 %: El 90 % de las páginas que se cargaron de manera satisfactoria.
- Mix: Tiempo mínimo en que ha demorado en cargarse una página.
- Max: Tiempo máximo en que ha tardado en cargarse una página.
- % Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
- Rendimiento: Tiempo total que demoró en cargarse la cantidad de peticiones (páginas hilos) de la prueba.
- KB/S: Tiempo total que demoró en cargarse la cantidad de peticiones (páginas hilos) de la prueba en Kbyte por segundo.

Caso de prueba Gestionar Softarea (Incluir Softarea)

Como se puede ver el rendimiento, o sea, el tiempo que responde la aplicación ante las transacciones de 50 usuarios conectados concurrentemente es menor que los resultados esperados para los tres escenarios de la prueba que se muestra en la tabla 3.5.
En un ambiente con un servidor de 1 GB de RAM, un disco duro con una capacidad de 160 GB y con un ancho de banda de 10/100 Mbps UTP 5 LAM bps, con la ejecución de la herramienta automatizada Jmeter en el sistema operativo Linux, se obtuvo que para un total de 1700 páginas la aplicación generó un total de 30246,6 KB/s de transferencia de datos, lo que incurrió en un rendimiento de 22.3 segundos. Se demuestra que la aplicación es estable aunque la tasa de error es de 7.4 % debido a que en la aplicación existen páginas en Java Script, y como Jmeter no simula un navegador, por lo tanto ciertas características de la página web no pueden ser ejecutadas, por ejemplo Java Script, Applets de Java, animaciones o aplicaciones que usen flash. [22]

Realizar la prueba de Estrés.

Esta prueba busca el punto donde la aplicación empieza a dejar de funcionar y se torna lenta.
En cuanto a la utilización de los recursos en el EC6, el procesador tiene una capacidad entre 60 % y 70% para 50 usuarios simulados por la herramienta, pero a partir de 1200 usuarios tiene un 80 % de capacidad por lo que el sistema empieza a degradarse, por lo que la aplicación se torna más lenta siendo esto, una de las principales causa de abandono de páginas web por parte de los usuarios.

Esta aparece debido a que durante la grabación de la carga, se observa que se graba en la herramienta Jmeter todas las peticiones HTTP que hace constantemente el Jabber interno de la aplicación al servidor, por lo que interfiere en los resultados de la prueba.

3.2.3.3 Cierre de las prueba

La aplicación de este procedimiento con la herramienta Jmeter a pesar de que no se encuentran ninguna NC se realiza de manera satisfactoria, pues le permite a los administradores de Alfaomega conocer el punto donde la plataforma ZERA comienza a tener problema con determinado número de usuarios, lo cual resulta necesario, para tomar ciertas precauciones como: revisar el código de los programas, los tiempos de respuestas de los servidores.

La aplicación de las pruebas en general tiene una estructura que facilita comprender en la práctica cada actividad que se describe en el procedimiento. Aunque la arquitectura lógica y física del laboratorio de pruebas de GCF no posee las mismas características que las de Alfaomega:

- Se puede determinar un estimado sobre cómo responde la aplicación ante una carga de trabajo en específica.
- Técnicamente el resultado de estas pruebas muestra que los requisitos no funcionales que se recogen cumplen con las expectativas del módulo de Gestión de Softarea.
- Se estima que la aplicación funcione correctamente en las escuelas para 50 usuarios no siendo así para 1200 usuarios.
- Indica cómo va a reaccionar el sistema cuando excede el límite de su funcionamiento, en este caso el exceso es de sobre el número de usuarios conectados concurrentemente a la aplicación web.

3.3 Conclusiones parciales del capítulo

En este capítulo se aplicó el procedimiento descrito anteriormente al proyecto de Alfaomega en la plataforma ZERA específicamente al módulo de Gestión de Softarea, donde se precisa que el funcionamiento de la aplicación es estable en cuanto al uso de los recursos y rendimiento, se observa que las páginas de Java script no son generadas por la herramienta Jmeter y que para 1200 usuarios conectados a la aplicación hace que se degrade el servidor y se torne más lenta la aplicación. La

aplicación de este procedimiento permite observar el éxito de las fases y las actividades definidas durante la ejecución de las pruebas en el módulo de Gestión de Softarea.

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 durante la etapa de prueba.

Al realizar este procedimiento se llegaron a las siguientes conclusiones:

- Realizarle las pruebas a los productos en su etapa de desarrollo constituye un factor fundamental en la calidad del mismo y por consiguiente en la aceptación del producto por parte de los clientes y usuarios.
- Se realizó un estudio de las pruebas de rendimiento que se realizan en la UCI y en el mundo para definir las actividades que se deben realizar en el procedimiento para en Grupo de Calidad FORTES.
- Con la elaboración de este procedimiento se logra introducir las pruebas de rendimiento de carga y estrés a los servicios que brinda el GCF, de acuerdo con las características del mismo.
- La aplicación de este procedimiento servirá para la corrección de errores en el funcionamiento de las aplicaciones web que se realicen en FORTES o en cualquier otro centro que solicite los servicios del GCF.
- Se define todas las actividades relacionadas con la ejecución de las pruebas de carga y estrés, las responsabilidades individuales para cada tarea, los recursos y los requisitos que deben ser considerados para realizar estas pruebas.
- Se desarrolló y se aplicó el procedimiento para las pruebas de rendimiento de acuerdo con las características del GCF, lo cual contribuyó con la calidad de la plataforma ZERA específicamente en el módulo Softarea.

Recomendaciones

- Extender la aplicación de las pruebas de carga y estrés a los demás casos de pruebas críticos, ya que solo fue aplicado a un solo caso de prueba y a tres escenarios de la aplicación desarrollada en el proyecto Alfaomega.
- Estudiar sobre otras herramientas que se puedan utilizar y modificar el procedimiento.
- Estudiar sobre los otros tipos de pruebas de rendimiento para poder crear algún procedimiento que se pueda utilizar en un futuro.
- Analizar y profundizar la interpretación de los resultados que muestra la herramienta Jmeter.
- Al realizar las pruebas, se debe comenzar con un número pequeño de usuarios e ir aumentando la carga de trabajo gradualmente. Siempre se debe dar tiempo a que el sistema se estabilice entre aumentos de carga.
- Para mejor información sobre los escenarios de pruebas más utilizados por los usuarios, se debe realizar una explotación del sistema durante un tiempo para observar los ficheros logs de la aplicación para tener datos más concretos.

Referencias Bibliográfica

[1] Universidad Tecnológica de Pereira. Introducción a la ingeniería de software. [En línea] 2008. [Citado el: 14 de enero de 2011.] http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/111233326-331.pdf

[2] Mi Tecnológico. Temario para las nuevas materias del plan de estudio 2010. [Citado el: 1 junio de 2011.] http://www.mitecnologico.com/Main/DefinicionCalidadDeSoftware

[3] R. S. Pressman. Ingeniería del software. Un enfoque práctico.5taEdición. McGrawHill.

[4] 2011EcuRedhttp://www.ecured.cu/index.php/Pruebas_de_Calidad_de_Software.

[5] Ivette Carolina Martínez. Universidad Simón Bolívar. http://ldc.usb.ve/~martinez/cursos/ci3715/clase11_AJ2010.pdf

[6] It-Mentor. Capacitación y guía para el desarrollo del software. http://materias.fi.uba.ar/7548/Pruebas-Intro.pdf

[7] 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: 13 de enero de 2011.]http://is.ls.fi.upm.es/docencia/erdsi/Documentacion_Evaluacion_7.pdf

[8] María Eudenia Valencia de Avadía. Procedimiento para Prueba de Software. [Citado el: 10 de febrero de2011.]http://dspace.icesi.edu.co/dspace/bitstream/item/2645/1/Procedimientos_para_prueba_software.pdf

[9] 2007 Johanna Rojas y Emilio Barrioshttp://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html

[10] Conferencia de Ingeniería de software. Perfil de calidad. Pruebas y evaluación del software 2008/2009.

[11] Testing en español. Artículos y Herramientas de testing. 2009-2011 http://josepablosarco.wordpress.com/performance-testing/

[12] Testhouse. Pruebas de Rendimiento. [En línea] 2010. [Citado el: 13 de Enero de 2011.]http://www.es.testhouse.net/pruebas-de-rendimiento

[13]Portal Corporación Cybven C. A. 2011.http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246

[14] Gustavo Vázques. Testing de Performance aplicando metodologías de ingeniería del Software y experiencias de su aplicación en Uruguay. Ensayos de Software, Facultad de Ingeniería de la Universidad de la República, Montevideo, Uruguay www.ces.com.uy/documentos/imasd/CES-Performance.pdf

[15] Microsoft patterns&practice: Performance Testing Guidanse for Web Applications. 2006-2011. http://perftestingguide.codeplex.com/

[16] Msdn. *Microsoft.* [En línea] [Citado el: 14 de enero de 2011.]http://msdn.microsoft.com/es-es/goglobal/bb688152

[17] Ruvalcaba, Manuel. 2011**.** Manuel Ruvalcaba. [En línea] 2011. [Citado el: 14 de enero de 2011.]http://www.manuelruvalcaba.com/tag/loadrunner/

[18] Articoloz, Directorio de artículos gratishttp://www.articuloz.com/software-articulos/herramienta-para-la-ejecucion-de-las-pruebas-886032.html#sdfootnote1anc

[19] Testing calidad de software. [En línea] [Citado el: 13 de Enero de 2011.]http://www.ivanfl.com/testingcalidad/index.php/pruebas-de-carga/107-jmeter/99-1-primer-contacto.html

[20] 2011Paesslerhttp://www.paessler.com/webstress/features

[21] Grupo ARQUISOFT - Johanna Rojas - Emilio Barrios – 2007.http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node68.html

[22] Copyright is for losers desarrollado por Wikidot.com. [En línea] [Citado el: 23 de Mayo de 2011.]http://carloszuluaga.wikidot.com/pruebascarga:grabar-script-web

Bibliografías

Universidad de Morón - Facultad de Informática, Ciencias De la Comunicación y Técnicas Especiales. Calidad de software. http://perftestingguide.codeplex.com/

Microsoft patterns&practice: Performance Testing Guidanse for Web Applications. 2006-2011. http://perftestingguide.codeplex.com/

EJIE S.A.SociedadInforática del Gobierno Vasco. Tipo de Pruebas.2008 www.ejie.net/documentos/Anexos_PBT/Indicadores_NAC.Desarrollo.pdf

F. Javier Díaz, Claudia M. TzancoffBanchoff, Anahí S. Rodríguez, Valeria Soria, Usando Jmeter para pruebas de rendimiento *ww.linti.unlp.edu.ar/.../usando_jmeter_para_pruebas_de_rendimiento.pdf*

Liudmila Sánchez Almenares. Cómo realizar pruebas de carga y estrés con la herramienta Jmeter.2008

Capítulo12: Pruebas. Introducción al Jmeter.*bibing.us.es/.../TOMO+II%252F12+Capitulo+12+Pruebas.pdf*

Junta de Andalucia [En línea] 1999-2010, [Citado el: 1 de marzo de 2011.]http://www.juntadeandalucia.es/xwiki/bin/view/MADEJA/JMeter

Portal de Corporación Cybven, C.A. Integración Tecnológica.2011 http://www.corporacionsybven.com/portal/index.php?option=com_content&view=article&id=246

Juan Oliver Navarro. Estado del arte de métodos, tipos de testing y herramientas para aplicar pruebas de rendimiento. Trabajo presentado como requisito para optar por el Título de Ingeniero de Sistema. Fundación Universitaria Tecnológica Comfenalco. Cartagena de India. 2010 http://es.scribd.com/doc/37584872/ESTADO-DEL-ARTE-DE-METODOS-TIPOS-DE-TESTING-Y-HERRAMIENTAS-PARA-APLICAR-PRUEBAS-DE-RENDIMIENTO

Gustavo Vázques. Testing de Performance aplicando metodologías de ingeniería del Software y experiencias de su aplicación en Uruguay. Ensayos de Software, Facultad de Ingeniería de la Universidad de la República, Montevideo, Uruguay www.ces.com.uy/documentos/imasd/CES-Performance.pdf

Estadísticas basadas en logs. [En línea] 2000 -2011. [Citado el: 2 de marzo de 2011.]http://www.entraenlared.com/estadisticas/logs.asp

The Apache Jakarta Proyect *1999-2011,* The Apache Software Foundation. http://jakarta.apache.org/jmeter/index.html

Nazareno Marín. GrupoSolución. Servicios de Marketing en línea [En línea] 10 de junio de 2011, [Citado el: 1 de junio de 2011.] http://www.gruposolucion.com/gs/?page_id=16

E. J.Weyuker, F. I. Vokolos. *Experience with Performance Testing of Software Systems: Issues, an Approach, and Case Study.* IEEE Transactions on Software Engineering.http://www.cc.gatech.edu/classes/AY2005/cs4803epr_spring/CaseStudies/CaseStudy_TSEperformanTS.pdf

Softonic.Programas gratis.Descargas seguras [En línea] 2011, [Citado el: 1 de junio de 2011.] http://firebug.softonic.com/descargar

Anexos

Anexo 1: Plantilla de caso de prueba

<Nombre del Proyecto>

Módulo:<Nombre del Módulo>

<Versión del proyecto>

Diseño de Casos de Pruebas de Carga y Estrés

<Nombre del Caso de Uso>

<Versión del CP >

Control de versiones:

Fecha	Versión	Descripción	Autor
<dd/mmm/yy>	<x.x>	<detalles>	<nombre>

Tabla de Contenido

Muestra todo el contenido del caso de uso con sus respectivas páginas.

1. **Descripción General.**

<Resumen del CU.>

2. **Condiciones de Ejecución.**

<Precondiciones del CU.>

3. **Secciones a probar en el Caso de Prueba.**

<Para cada sección los escenarios van a ser flujo básico + los alternativos.>

<Para cada sección los escenarios van a ser flujo básico + los alternativos.>

ID del escenario	Escenarios de la sección	Carga de Trabajo	Descripción	Resultado esperado	Resultado de la prueba
	EC 1.2: Nombre del Escenario.		*<Descripción de la Funcionalidad.>*		

	EC 1.n: Nombre del Escenario.		*<Descripción de la Funcionalidad.>*		

4. Registro de defectos y dificultades detectados.

Elemento	No	No conformidad	Aspecto correspondiente	Etapa de detención	Significativa	No significativa	Estado de la NC	Respuesta de Equipo de desarrollo
<Nombre del Elemento>	*< 1>*	*<Descripción de la No Conformidad>*	*<Descripción del Aspecto correspondiente>*	*<Etapa de detección del error>*			*<Se coloca el estado de la NC y la fecha, cada vez que se revise se deja el estado anterior y se coloca el nuevo con la fecha en que se revisó.> RA: Resuelta PD: Pendiente NP: No Procede*	*<Esta columna se comienza a llenar a partir de la 2da iteración, y es responsabilidad del equipo de desarrollo, quien especifica la conformidad con lo encontrado o no y en caso de no proceder la no conformidad explica por qué.>*

En la tabla de Registro de defectos y dificultades detectados se reportan las NC que se muestran en la interfaz de las aplicaciones mientras se hace la prueba ejemplo una imagen que no se muestra con determinada cantidad de usuarios conectados.

Anexo 2: Coso de prueba del módulo de Gestión de Softarea

ID del scenario	Escenario	RD	Est	Ev	Ej	FE	HC	N	P	Respuesta del sistema
EC 1	Selecciona la opción de incluir una nueva tarea.	V	V							Muestra los tipos de elementos asociados a la tarea y permite seleccionar: • Recorrido Dirigido. • Estudiantes y\o grupos de ellos. Y permite además: • Crear Recorrido Dirigido. • Actualizar Listado RD. • eleccionar todos los estudiantes. • Desmarcar todos los estudiantes. • Ir al botón Siguiente. • Salir de la Gestión de la Softarea.
EC 2	Selecciona un RD. Selecciona el o los estudiantes involucrados. Selecciona el botón Siguiente.			V	N\A					Muestra una lista de: • Evidencias. • Ejercicios. Y permite seleccionar al menos uno de estos elementos. Permite: • Crear Evidencia. • Actualizar Listado E. • Ir al botón Anterior. • Ir al botón Siguiente. • Salir de la Gestión de la Softarea.
				N\A	V					

EC 3	Selecciona una evidencia y uno o más ejercicios y/o Selecciona el botón Siguiente.					V	V	V	V	Permite seleccionar: • Fecha de entrega. • Hora en que cierra. • E introducir: • Nombre (Identificador de la tarea). • Propósito. • Acciones a llevar a cabo. • Y permite: • Ir al botón Anterior. • Ir al botón Siguiente. • Salir de la Gestión de la Softarea.
EC 4	Selecciona la fecha de inicio, fecha de entrega, la hora en que cierra, e introduce el nombre de la Softarea, propósito y acciones a llevar a cabo. Selecciona el botón Siguiente.									Muestra un resumen de la tarea y permite: • Ir al botón Anterior. • Finalizar y asignar. • Salir de la Gestión de la Softarea.
EC 5	El actor selecciona la opción de Finalizar y asignar.									*Valida los datos.* *Crea la tarea.* Finaliza la tarea y la envía a los estudiantes seleccionados.
										La asignación de la tarea se notifica a la mensajería del estudiante. *Se crea un evento en la agenda, con la fecha definida en la tarea.* El caso de uso termina.
EC 6	Selecciona la opción de Crear Recorrido Dirigido.									Muestra el *CU Gestionar Recorrido Dirigido.*

EC 7	Selecciona la opción de Actualizar Listado RD.									Se actualiza el listado de recorridos dirigidos. Regresa al paso 3 del flujo básico.
EC 8	Selecciona la opción de Desmarcar todos los estudiantes.									Se desmarcaran todos los estudiantes.
EC 9	Selecciona la opción de Crear Evidencia.									Muestra el *CU Gestionar Evidencia.*
EC 10	Selecciona la opción de Actualizar Listado E.									Se actualiza el listado de las evidencias. Regresa al paso 6 del flujo básico.
EC 11	Selecciona el botón Anterior.									Regresa a la pantalla anterior.
EC 12	Existen Campo incompletos.	*V*	*V*	*V*	*V*	*V*	*V*	*V*	*V*	Se crea la tarea exitosamente. Sobre cada campo incompleto muestra un mensaje de información.
		V	*V*	*N\A*	*V*	*V*	*V*	*V*	*V*	
		V	*V*	*V*	*N\A*	*V*	*V*	*V*	*V*	
		I	*V*	*I*	*I*	*V*	*V*	*V*	*V*	
		V	*I*	*I*	*I*	*V*	*V*	*V*	*V*	
		V	*V*	*V*	*N\A*	*I*	*V*	*V*	*V*	
		V	*V*	*N\A*	*V*	*V*	*I*	*V*	*V*	
		V	*V*	*I*	*I*	*V*	*V*	*I*	*V*	
		V	*V*	*I*	*I*	*V*	*V*	*V*	*I*	
		V	*V*	*I*	*I*	*V*	*V*	*V*	*V*	

EC 13	Existen datos incorrectos	*N\A*	*N\A*	*N\A*	*N\A*	*N\A*	*N\A*	*I*	*V*	Sobre cada campo incorrecto muestra un mensaje de información.
		N\A	*N\A*	*N\A*	*N\A*	*N\A*	*N\A*	*V*	*I*	
		N\A	*N\A*	*N\A*	*N\A*	*N\A*	*N\A*	*V*	*V*	
EC 14	Selecciona la opción de Cancelar de la Gestión de la Softarea.									Muestra un mensaje de información. Permite: Aceptar. Cancelar.
EC 15	Selecciona la opción Guardar sin terminar.									Sale de la gestión de la Softarea.

Glosario de términos

Disponibilidad: El grado en que un sistema es funcional, estable y eficiente. Se mide por el tiempo de funcionamiento (el tiempo entre fallos).

Fiabilidad: La capacidad de un sistema para ser funcional, estable y eficiente, tanto en circunstancias de rutina como en condiciones adversas.

Escalabilidad: Se refiere a la aplicación de la capacidad para manejar una carga de trabajo adicional, sin afectar negativamente el rendimiento, mediante la adición de recursos tales como el procesador, la memoria y la capacidad de almacenamiento.

Rendimiento: La tasa de operaciones por unidad de tiempo.

Usuario Virtual: Un usuario creado para aplicar la carga a un sistema. Cada usuario virtual simula las acciones reales de los usuarios mediante el envío de peticiones HTTP a un servidor de la misma manera que un cliente.

Cuello de botella: Punto donde la congestión y los retrasos se producen en una aplicación, ralentizando la tramitación de las solicitudes y causando que los usuarios experimenten retrasos inaceptables en el servicio. Puntos débiles de una aplicación. Obstáculos.

Fallo: La incapacidad de un sistema o de alguno de sus componentes para realizar las funciones de requeridas dentro de los requisitos de rendimientos especificados.

Error: Acción humana que conduce a un resultado incorrecto.

Capacidad: Es la carga total que un sistema puede aguantar sin violar criterios de aceptación de rendimiento.

Tiempo de respuesta: Es una medida de respuesta que le da la aplicación a una solicitud de un cliente. El tiempo que pasa entre la petición inicial y la descarga completa de la respuesta.

Saturación: Se refiere al punto en el que ha llegado a una plena utilización de los recursos.

Escenarios: En el contexto de rendimiento es una secuencia de paso que puede realizar el usuario en la aplicación, Un escenario puede representar un caso de uso, caso de prueba o una función de negocios como la búsqueda de un catálogo de productos, añadiendo un elemento a una cesta de la compra, o realizar un pedido.

Estabilidad: En el contexto de las pruebas de rendimiento, estabilidad se refiere a fiabilidad general, la solidez, y la integridad de datos funcionales, la disponibilidad, y la coherencia de la respuesta de un sistema bajo una variedad de condiciones.

Rendimiento: Es el número de unidad de trabajo que puede ser manejado por unidad de tiempo, por ejemplo las solicitudes por segundo, las llamadas por día.

Utilización: En el contexto de pruebas de rendimiento, la utilización es un porcentaje de tiempo que un recurso está siendo ocupado por el servicio que solicita un usuario.

Componente de prueba: Es cualquier prueba de rendimiento que se dirige a cualquier elemento arquitectónico de la aplicación, que pueden ser servidores, base de datos, redes, dispositivos de almacenamiento.

Carga de trabajo: Es el estímulo aplicado a una aplicación o componente, para simular un patrón de uso, en lo que respeta concurrencia o entrada de datos. La carga de trabajo incluye el número total de usuarios activos simultáneos, volúmenes de datos y volúmenes de transacción, junto con la mezcla de transacción. Para el modelado de rendimiento, una carga de trabajo asociada con un escenario particular.

Logs: Los ficheros logs son una grabación de la actividad del sitio web a lo largo de un período de tiempo en particular.

Symfony: Symfony es un framework muy completo, diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web

ZERA: Software educativo destinada a apoyar el proceso de enseñanza – aprendizaje.

Validación: Comprobación de que se está construyendo el producto correcto.

Verificación: Comprobación de que se está construyendo el producto correctamente.

Artefactos: Cualquier elemento que resulte del proceso de desarrollo de software; por ejemplo: documentos de requisitos, especificaciones, diseños de interfaces, caso de pruebas, etc.

Servidor: Es una computadora que, formando parte de una red, provee servicios a otras computadoras llamadas clientes.

Servidor web o servidor HTTP: Es un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web.

Aplicaciones web: Son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador. Es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.