

Procedimiento para la Aplicación de Pruebas de Carga y Rendimiento
Mediante una Herramienta Tecnológica en Aplicativo Web para el CIADTI

Danilo Jherson Saed Flórez Tunaroz

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERIAS Y ARQUITECTURA
DEPARTAMENTO DE ELECTRONICA, ELECTRICA, TELECOMUNICACIONES Y
SISTEMAS
PROGRAMA INGENIERIA DE SISTEMAS
PAMPLONA
2016

PROCEDIMIENTO PARA LA APLICACION DE PRUEBAS DE CARGA Y
RENDIMIENTO MEDIANTE UNA HERRAMIENTA TECNOLOGICA EN
APLICATIVO WEB PARA EL CIADTI.

DANILO JHESRON SAED FLOREZ TUNAROZA

TRABAJO DE GRADO PARA OPTAR POR EL TITULO DE INGENIERIO DE
SISTEMAS

DIRECTOR: MS. LUIS ALBERTO ESTEBAN VILLAMIZAR

UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERIAS Y ARQUITECTURA

DEPARTAMENTO DE ELECTRONICA, ELECTRICA, TELECOMUNICACIONES Y
SISTEMAS

PROGRAMA INGENIERIA DE SISTEMAS

PAMPLONA

2016

DEDICATORIA

Dedico este trabajo principalmente a Dios por permitir culminar este proyecto. Porque me ha dado todo lo que tengo en la vida: salud, fuerza, sabiduría y entendimiento.

Con ayuda de él emprendo una nueva etapa de mi vida, para ser mejor profesional y una persona íntegra para la sociedad.

Sin desconocer el apoyo, ejemplo y amor de mis Padres, Abuelos, Tíos y demás familiares quienes me acompañaron y orientaron en diferentes momentos de este camino hacia el éxito profesional.

AGRADECIMIENTOS

Agradezco a Dios por haberme permitido contar con una gran familia, que ha creído siempre en mí, dándome ejemplo de superación, humildad y sacrificio. Enseñándome a valorar todo lo que tengo. A mi padre Oscar Flórez Zúñiga, a mi abuela Gladys Mogollón, Dary Tunarosa y Víctor Manuel Celis por su apoyo, y fomentando en mí el deseo de superación. Mi madre Erika Tunarosa Mogollón por su incondicional apoyo y mi mayor motivación para lograr el presente trabajo.

A todos los docentes que contribuyeron en mi formación como profesional y como persona: Ailin Orjuela, Carlos Parra, Edgar Albornoz, Sergio Peñaloza, Omar Portilla, Luis Armando Portilla, y de manera especial a los docentes Luis Esteban, Nelson Fernández y Orlando Maldonado por su acompañamiento y conocimientos brindados.

A las personas que me brindaron su amistad y que contribuyeron a mi formación.

Tabla de contenidos

1	Introducción	2
1.1	Objetivos.....	4
1.1.1	Objetivo General.....	4
1.1.2	Objetivos Específicos	4
1.2	Planteamiento del Problema	4
1.3	Justificación	5
1.4	Metodología de trabajo	6
2	Marco teórico	8
2.1	Pruebas de Software.....	8
2.2	Estrategias de Pruebas de Software.	9
2.2.1	Estrategia de Prueba de Software Para Arquitecturas Convencionales.....	10
2.2.2	Estrategia de prueba de Software en el Ambiente del aplicativo web.....	12
2.3	Planeación de Pruebas.	14
2.3.1	Documentación de las Pruebas.....	15
2.4	Técnicas de Prueba de Software.	16
2.4.1	Casos de Pruebas	17
2.4.2	Actividades de las Pruebas.	18
2.5	Pruebas de Desempeño.	19
2.5.1	Pruebas de Carga.	19
2.5.2	Pruebas de Tensión.....	20
2.6	Arquitectura de Aplicativo Web	21
2.6.1	Arquitectura del Aplicativo Web (CIADTI)	22

2.6.2	Academusoft - Campus Académico	24
2.6.3	Gestasoft – Campus Administrativo.....	25
2.7	Descripción del software JMeter	26
2.7.1	Componentes del Plan de Pruebas.....	28
2.7.2	Grupo de Usuarios	28
2.7.3	Controladores	28
2.7.4	Fragmentos de Prueba	29
2.7.5	Receptores	29
2.7.6	Temporizadores	29
2.7.7	Afirmaciones	30
2.7.8	Elementos de configuración	30
3	Procedimiento propuesto para pruebas de Rendimiento.....	32
3.1	Descripción del procedimiento	33
3.1.1	Descripción del Software a probar	34
3.1.2	Preparación de Infraestructura.....	34
3.1.3	Diseño Caso de Prueba.....	36
3.1.4	Ejecución de la prueba.....	37
3.1.5	Análisis de Resultados.....	37
3.2	Aplicación del procedimiento propuesto	38
3.2.1	Descripción del Software a Probar	38
3.2.2	Preparación de infraestructura	41
3.2.3	Diseño caso de prueba	43
3.2.4	Ejecución de la Prueba	47
3.2.5	Análisis de Resultados.....	47
4	Conclusiones	52

5	Bibliografía.....	53
6	Anexos.....	56

Tabla de figuras

Figura 1 Pasos de Pruebas de Software	9
Figura 2 Estrategia de Prueba de Software.....	10
Figura 3 Estrategia de Prueba de Software Convencional	12
Figura 4 Proceso de Prueba en Ambiente de el aplicativo web.....	13
Figura 5 Olsina “Specifying Quality Characteristics and attributes for Web Sites.....	15
Figura 6 Jacyntho,D, "An Architecture for Structuring Complex Web Applications"	22
Figura 7 Fuente. Documento de Infraestructura y soporte tecnológico (CIADTI.....	23
Figura 8 Aplicativo Web v.3.2 (Academusoft)	39
Figura 9 Aplicativo Web v.4.0 (Academusoft)	40
Figura 10 Configuración Grupo de Usuarios	44
Figura 11 Configuración proxy del navegador Firefox	45
Figura 12 Ajuste de Receptores de la Prueba.....	48
Figura 13 Resultados de la ejecución de la Prueba	49
Figura 14 Resultados Petición de usuario	50
Figura 15 Arbol de Respuesta (Tipo de Petición)	50

Lista de tablas

Tabla 1 Caso de Prueba [IEEE Std. 829-1991]	16
Tabla 2 Elementos de Caso de Pruebas	18
Tabla 3 Sistemas Operativos Compatibles para JMeter	27
Tabla 4 Software.....	41
Tabla 5 Material Bibliográfico	42
Tabla 6 computador de escritorio (Equipo para uso)	42
Tabla 7 Computador Portátil	42
Tabla 8 Servicios	43

RESUMEN

Este documento describe un procedimiento para la realización de pruebas de carga y rendimiento a aplicaciones web mediante el uso de la herramienta Jmeter. El procedimiento fue diseñado de acuerdo a las necesidades de pruebas de este tipo en el CIADTI donde se desarrolló la pasantía como modalidad de trabajo de grado.

Palabras claves: Aplicativo web, pruebas de software, ingeniería del software, CIADTI, herramienta tecnológica, aplicación web.

1 Introducción

Las pruebas de software se pueden definir como el intento sistemático de localizar errores en forma planeada en el software implementado, de ahí la importancia de seleccionar una adecuada técnica de pruebas en el aplicativo web para garantizar su correcto funcionamiento. En un proceso de pruebas se pueden identificar tres tareas principales: “desarrollo de los casos de prueba, ejecución de los casos de prueba y análisis de los resultados” (Burnstein, 2003).

Este documento describe un procedimiento para la aplicación de pruebas de software (carga y rendimiento), adaptado a las funcionalidades del aplicativo web CIADTI, creando un mapa que describe los pasos que hay que llevar a cabo como: cuándo se deben planificar y realizar, cuánto esfuerzo y recursos van a requerir incorporando planificación, diseño y ejecución de las pruebas y la agrupación y evaluación de los datos resultantes haciendo uso de la herramienta tecnológica.

La principal idea es ofrecer una respuesta sistemática a la pregunta: ¿Cómo desarrollar un caso de prueba que permita la verificación del comportamiento del aplicativo web, respecto a rendimiento y carga? La cual se responderá concretamente en las siguientes páginas apoyado por una técnica de la ingeniería del software y la herramienta tecnológica.

Una característica de la aplicación de pruebas de rendimiento o desempeño, es simular situaciones de carga en el mundo real conforme crece el número de usuarios simultáneos de la aplicación web, el número de transacciones en línea, o se incrementa la cantidad de datos (descargados o cargados) buscando el quebrantamiento del software. De esta manera los desarrolladores identifican el defecto que causa la falla observada; se procede hacer una corrección que es un cambio a un componente con el propósito de eliminar o reparar los defectos del software. El objetivo de las pruebas es descubrir la mayor cantidad de fallas con el menor esfuerzo posible, para que puedan repararse antes de la entrega del software.

En este caso se utilizaron Pruebas de Desempeño que permite evaluar el comportamiento de una aplicación ante condiciones extremas operacionales, tales como picos de usuarios, y con base en la información obtenida después de ejercitar la funcionalidad bajo prueba, el

software bajo los parámetros y requerimientos establecidos, realizar ajustes para mejorar su rendimiento.

Con las Pruebas de Desempeño se puede detectar el número de usuarios y la cantidad de datos, transacciones, peticiones que la aplicación puede soportar sin que su rendimiento se vea degradado. Pero si ese número de usuarios no es el requerido, con estas pruebas se puede obtener información para detectar los segmentos de la aplicación que ocasionan bajo rendimiento o hasta llegar al punto del colapso, de manera que puedan ser corregidos, para que luego se pueda verificar si el rendimiento de la aplicación web es el esperado.

Este documento está organizado en cuatro capítulos:

En el primer capítulo se describe los alcances del trabajo de grado en término de los objetivos, del planteamiento del problema y la justificación, posteriormente se encuentra el marco teórico y los métodos, por último se encuentra el desarrollo del proyecto el cronograma, las conclusiones, recomendaciones y la bibliografía.

En el segundo capítulo se presenta los conceptos necesarios mínimos para la definición del procedimiento, estos temas se agrupan bajo el título de marco teórico.

El tercer capítulo presenta el procedimiento adaptado a la medida de las necesidades del CIADTI.

El cuarto capítulo presenta las conclusiones y recomendaciones. Finalmente se encuentra un listado de referencias que se usaron durante todo el documento

1.1 Objetivos

1.1.1 Objetivo General

Desarrollar un procedimiento para el proceso de pruebas en el aplicativo web CIADTI a través de la utilización de una herramienta tecnológica.

1.1.2 Objetivos Específicos

- ✓ Analizar un modelo de pruebas que requiera el aplicativo web para caracterizar el procedimiento y el uso de la herramienta tecnológica.
- ✓ Diseñar un procedimiento y/o estrategia para el manejo de la herramienta tecnológica utilizando como parámetros la estructura del aplicativo web.
- ✓ Realizar y aplicar sesiones de pruebas en funcionalidades del aplicativo web CIADTI haciendo uso de la herramienta tecnológica.

1.2 Planteamiento del Problema

Las pruebas de desempeño forman parte del ciclo de vida de desarrollo software y permiten verificar el buen funcionamiento de la aplicación. Según (Dijkstra, 1972), las pruebas de software pueden demostrar la presencia de errores pero no la ausencia de ellos.

Por ello resulta necesario realizar pruebas de software haciendo énfasis en pruebas de desempeño, en el desarrollo de funcionalidades del aplicativo web CIADTI, facilitando al programador el cambio de código para mejorar su estructura (proceso llamado refactorización) y a su vez haciendo un software más robusto (es la capacidad de respuesta del software ante condiciones excepcionales tales como entradas inválidas o maliciosas, asegurando la protección de los datos y del software), permitiendo hacer pruebas sobre los cambios y así asegurando que los nuevos cambios no han introducido errores. Para esto hay que tener en cuenta que el ciclo de vida de las pruebas nunca finaliza ya que cada interacción de los usuarios finales con la aplicación, es el inicio de una nueva prueba con nuevos datos y configuraciones de usuarios.

Se hace necesario también como parte de las pruebas identificar, contextualizar, definir y relacionar todos los elementos que conforman el modelo del aplicativo web en especial las

funcionalidades a probar para llevar a cabo un procedimiento adecuado, integrando las soluciones de tecnologías de la información y pautas de la ingeniería del software, las aplicaciones, los procesos, y así cumplir con los requerimientos que se desean, de esta manera alcanzar sus objetivos de forma efectiva y eficiente, logrando competitividad y un software más confiable.

1.3 Justificación

Hoy en día las organizaciones tienen como necesidad aumentar su competitividad adaptándose al cambio en las tecnologías, sistemas de información, de servicios y productos. En este contexto se hace necesario establecer mecanismos y estrategias de pruebas de software que permita generar confiabilidad del software, siendo esta una medida del éxito donde el comportamiento del software se apega a condiciones especificadas.

Algunos tipos de pruebas, una aplicación web es vital conocer la capacidad del software en cuanto a número de usuarios y concurrencia de los mismos. Por ello resulta necesario simular un enjambre de usuarios que ponga a prueba la aplicación para poder valorar su rendimiento, o que realice en ella ciertas operaciones que se quieren poner a prueba.

Durante muchos años la única defensa contra los errores de programación era un cuidadoso diseño y la propia inteligencia y habilidad del programador, a través de los años esto ha cambiado, en la actualidad se cuenta con técnicas modernas de pruebas y herramientas tecnológicas que permiten reducir el número de errores que se encuentren en el código. Así probando el software antes de su entrega final y ofrecer un software de calidad, cumpliendo con los requerimientos y asegurando el correcto funcionamiento del aplicativo frente a peticiones del usuario.

Sin embargo el uso de una herramienta tecnológica de pruebas en particular, no asegura el 100% de la ausencia de errores ya que estas mismas herramientas cuentan con actualizaciones y versiones o parches que subsanan brechas contenidas en dichas herramientas que se utilizan para realizar las pruebas.

1.4 Metodología de trabajo

Para iniciar el desarrollo del procedimiento, se tomaron las funcionalidades orientadas a la inscripción de grados en instituciones de educación superior, del sistema de información académico. Para conocer el proceso de grados, fue necesario leer los manuales existentes y realizar algunas pruebas funcionales que permitieron establecer cómo debe ser el comportamiento de la aplicación con respecto a los usuarios y los resultados que de ella se esperan. Una vez identificadas las funcionalidades más complejas dentro del proceso en cuanto a la importancia y concurrencia por parte de los usuarios, se seleccionan para proceder a realizar las respectivas pruebas de carga y rendimiento.

Los resultados se obtuvieron con soporte en la aplicación web desarrollada en el centro de investigación aplicada, previamente codificada y con un proceso de pruebas realizado según los estándares y procedimientos que allí se utilizan. Según las métricas que se realizan al interior proceso de pruebas, se hallaron un total de 127 funcionalidades evaluadas para la última versión probada. De ellas 27 presentaron algún error de tipo funcional. Por tanto, se ejecutaron las pruebas no funcionales faltantes y relacionadas con la concurrencia de la aplicación. Lo anterior sirvió como base para iniciar el desarrollo del procedimiento al que este documento hace referencia; es decir los pasos para llevar a cabo un procedimiento de pruebas de carga y rendimiento en un aplicativo web del CIADTI. El objetivo buscó de forma esquemática, planeada y organizada identificar errores y encontrar los defectos antes de la liberación del software, y demostrar que la aplicación web satisface los requerimientos del proceso de grados.

Las variables de tiempo y costo fueron especialmente considerados, en contraste con el personal limitado para la realización de probar todas las funcionalidades del aplicativo web. Estas son las bases para la creación de una guía que indique los pasos correspondientes para la aplicación de pruebas de carga y rendimiento en el aplicativo web que incluya la planeación, técnicas de pruebas, metodología y una herramienta tecnológica que aportara rapidez en la ejecución de las pruebas.

Es importante tener en cuenta que este tipo de pruebas solo serán relevantes para aquellas funcionalidades que requieran alta demanda tanto individual como colectiva, lo que limita la cantidad de funcionalidades probadas que se deben aplicar. Con ello se logra la

reducción en el costo de las pruebas y el tiempo necesario para su terminación. Lo anterior son los requisitos para dar como resultado un Procedimiento para la aplicación de pruebas de carga y rendimiento mediante una herramienta tecnológica para el CIADTI.

Este proceso se llevó a cabo en diferentes funcionalidades lo que permitió establecer un estándar que permitió hacer un checklist de los pasos necesarios para generar un procedimiento de pruebas relacionado con el rendimiento de la aplicación. Esta lista de chequeo se puede conocer en la sección 3.

2 Marco teórico

En esta sección se explica los conceptos básicos referentes a Pruebas de Software, sus beneficios, tipos de pruebas que existen actualmente y sus ventajas. También hace parte de esta sección una descripción del proceso de pruebas, una clasificación breve de las más utilizados y destacadas en la actualidad para evaluar un aplicativo web.

2.1 Pruebas de Software.

“Las pruebas de software consisten en la verificación dinámica del comportamiento de un programa sobre una cantidad finita de casos de prueba, seleccionados cuidadosamente dentro de los infinitos casos posibles del dominio, contra un comportamiento esperado” (IEEE, 2004).

El término pruebas de software se puede definir como el proceso de encontrar diferencias entre el comportamiento esperado, especificado por el modelo del sistema y el comportamiento observado del software. Es un proceso en el que se revisa el software a probar en este caso a un aplicativo web bajo condiciones definidas explícitamente, y se le aplica, eventualmente con apoyo de software especializado, un conjunto de estímulos (los casos de prueba) diseñados de manera sistemática utilizando técnicas apropiadas, con el objetivo de detectar niveles inadecuados de calidad. Este proceso debe llevarse a cabo disciplinadamente y respaldarse en métricas bien definidas. Todas estas actividades y sus resultados son documentados, en especial las fallas detectadas.

El proceso de someter a prueba el aplicativo web es una suma de actividades relacionadas con una sola meta: descubrir errores en el contenido, la función, la facilidad de uso, la navegabilidad, el desempeño, la capacidad y la seguridad del aplicativo web (Miller E. , 2000). Esto se logra mediante la aplicación de una estrategia de prueba que abarca tanto revisiones como pruebas ejecutables. Los ingenieros a cargo del desarrollo y otros participantes del proyecto (gerentes, clientes, usuarios finales) toman parte en el proceso de probar el aplicativo web.

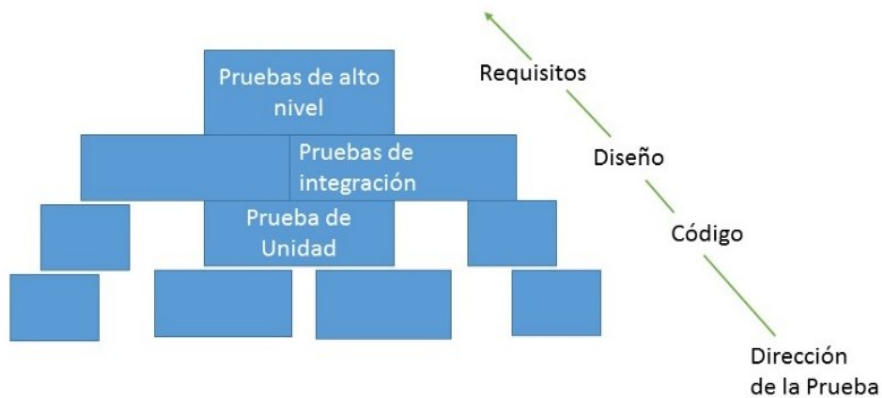


Figura 1 Pasos de Pruebas de Software

La figura anterior representa los diferentes tipos de pruebas aplicados secuencialmente: el inicio de la prueba se concentra en cada componente individual, asegurando que funcione de manera correcta como unidad. La prueba de integración trata todos los aspectos referentes a la verificación y construcción del programa, haciendo uso de técnicas de diseño de casos de prueba que se concentran en las entradas y las salidas de información para luego ser evaluadas. Una vez validado, el software debe combinarse con otros elementos del software (hardware, sistema operativo, personas, bases de datos), por consiguiente asegurando que el software cumple con todos los requisitos funcionales, de comportamiento y desempeño (Gil, 1995). De esta manera culminando las pruebas de alto nivel.

2.2 Estrategias de Pruebas de Software.

Las pruebas son el último escalón para la calidad y descubrimiento de errores. Pero las pruebas no deben considerarse como una red de seguridad. “No es posible probar la calidad, si carece antes de que empiece las pruebas de software, no estará cuando se termine” (Miller, 1977). La calidad del software se incorpora en todo el proceso de desarrollo con la aplicación de métodos y herramientas apropiadas. De tal manera las pruebas son un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática, por lo cual se debe especificar una plantilla para las pruebas de

software (pasos a seguir que contengan técnicas y métodos específicos del diseño de caso de pruebas). Todas tienen las siguientes características genéricas.

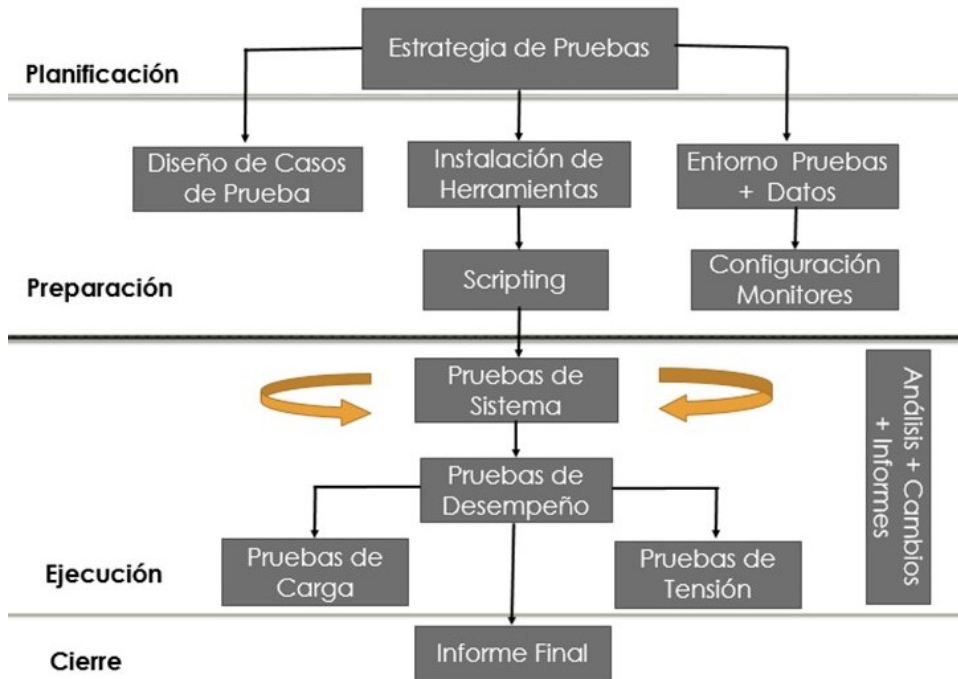


Figura 2 Estrategia de Prueba de Software

Para realizar pruebas efectivas un equipo de software debe llevar a cabo revisiones técnicas formales y efectivas. Esto eliminara muchos errores antes de que empiece la prueba. Las pruebas comienzan a nivel de componentes y trabaja de adentro hacia fuera, hacia la integración de todo el software. Es apropiado utilizar diferentes técnicas de pruebas en diferentes momentos. La prueba la dirige el desarrollador de software.

La prueba y la depuración son dos actividades diferentes, pero la segunda debe incluirse en cualquier estrategia de prueba. A continuación en esta sección se describirán algunas estrategias típicas de las Pruebas de Software más utilizadas actualmente.

2.2.1 Estrategia de Prueba de Software Para Arquitecturas Convencionales.

(Gil, 1995) Plantea que debe abordar los siguientes puntos si se desea implementar con éxito una estrategia de prueba de software.

- ✓ Especificar los requisitos del producto de manera cuantificable antes de que comiencen las pruebas. Como transportabilidad, facilidad de mantenimiento y uso,

siendo características de calidad de manera que sean medibles y no generen resultados ambiguos.

- ✓ Establecer los objetivos de la prueba de software de manera explicita como la efectividad de la prueba, la cobertura de la prueba, tiempo medio de fallo, el coste para encontrar y arreglar errores, densidad de fallos frecuencia de ocurrencia.
- ✓ Desarrollar un plan de prueba que pueda reducir el costo de las pruebas y el tiempo para su terminación.

Una estrategia también posible en forma de espiral (figura 3), la prueba de unidad empieza en el vértice de del espiral se concentra en cada componente del software, mediante la inspección manual del código fuente.

La prueba avanza hacia afuera llegando a la prueba de integración donde se examina el diseño y la construcción de la arquitectura del software. Luego de superar la anterior prueba ahora se encuentra la prueba de validación donde se validan los requisitos establecidos como parte del análisis de los requerimientos del software, comparándolos con el software creado. Por último se llega a la prueba de sistema donde se enfoca el software completo, asegurando que el software se apege a sus requerimientos funcionales y no funcionales.

La prueba del sistema verifica que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento total del software.



Figura 3 Estrategia de Prueba de Software Convencional

2.2.2 Estrategia de prueba de Software en el Ambiente del aplicativo web

Las estrategias de pruebas para una aplicación web adoptan los principios básicos para todas las pruebas de software. Los siguientes pasos resumen el enfoque:

- ✓ Se revisa el modelo contenido de la aplicación web para descubrir errores.
- ✓ Se revisa el modelo de la interfaz para asegurarse que todos los casos de uso puedan acomodarse.
- ✓ Se revisa el modelo de diseño de la aplicación web para descubrir errores de navegación.
- ✓ Se prueba la interfaz del usuario para descubrir errores en la presentación o los mecanismos de navegación.
- ✓ Componentes funcionales seleccionados se prueban en forma individual.
- ✓ Se prueba la navegación a través de toda la arquitectura.

- ✓ La aplicación web se implementa en diversas configuraciones ambientales y se prueba su compatibilidad con cada configuración.
- ✓ Se ejecutan pruebas de seguridad con el objetivo de explotar vulnerabilidades en la aplicación web dentro de su ambiente.
- ✓ Se llevan a cabo pruebas de desempeño.
- ✓ La aplicación web se prueba en un pequeño conjunto seleccionado de clientes/usuarios evaluando usabilidad, contenido y navegación, compatibilidad, confiabilidad y el desempeño de la aplicación web.

El proceso de prueba de la aplicación web con su diseño en pirámide, conforme se desarrolló el flujo de pruebas de izquierda a derecha y de arriba abajo, los elementos de la aplicación web visibles para el usuario (elementos superiores de la pirámide) se prueban primero, seguidos por los elementos de diseño de infraestructura.

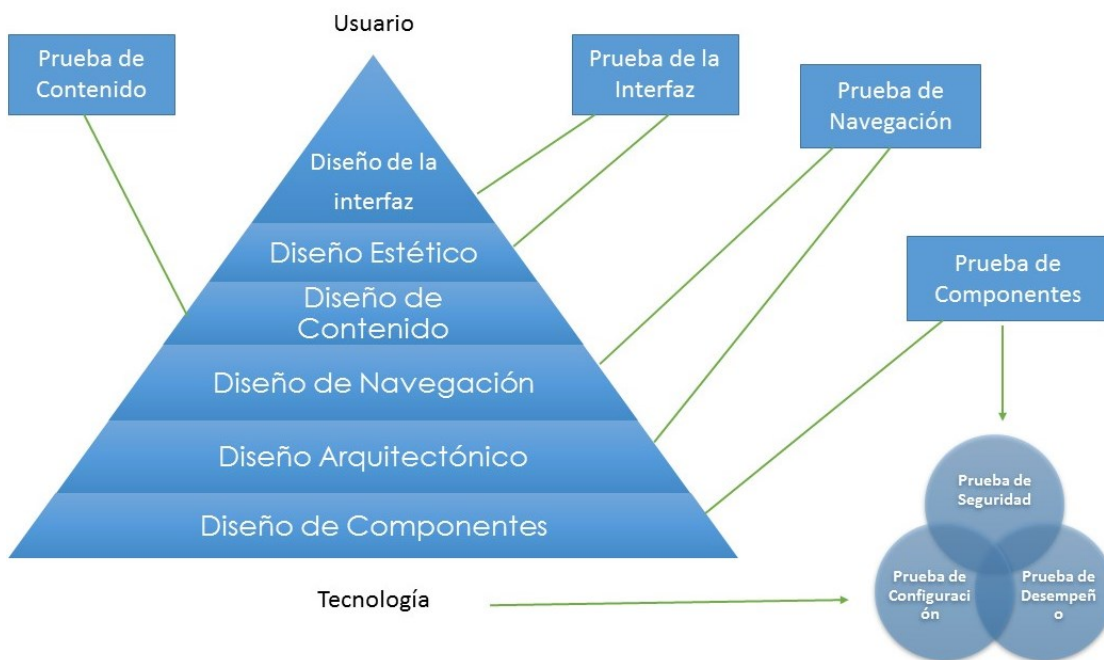


Figura 4 Proceso de Prueba en Ambiente de el aplicativo web

Las siguientes pruebas examinan ciertas dimensiones de calidad según, (Miller E. , 2000) (Nguyen, 2001) La prueba de contenido para descubrir errores tipográficos, errores en la consistencia del contenido, inexactitudes en las representaciones gráficas y fallas en las referencias cruzadas y en el contenido estático. La prueba de interfaz ejercita los mecanismos de interacción y valida los aspectos estéticos de la interfaz del usuario, hallando debilidades en la implementación de interacción.

La prueba de navegación evalúa el flujo de navegación entre los objetos y el contenido para todas las funciones de la aplicación web. Cada página web encapsula contenido vínculos de navegación y elementos de procesamiento (estilos, formatos, applets), una “unidad” dentro de la arquitectura de la aplicación web puede ser un componente funcional que proporciona servicio directamente a un usuario final, en la misma forma que se prueba un módulo individual en el software convencional, además las pruebas de navegación y componentes se maneja como pruebas de integración. La prueba de componentes evalúa el contenido y las unidades funcionales dentro del aplicativo web. Las pruebas de configuración intentan descubrir los errores respecto a un cliente o un ambiente de servidor en particular, se crea una matriz de referencia cruzada que se encamina a descubrir errores asociados en cada posible configuración teniendo en cuenta como parámetros todos los posibles sistemas operativos, plataformas de hardware, navegadores y protocolos de navegación.

La prueba de seguridad trata de encontrar fallas de seguridad en el software, diseñadas para colapsar las vulnerabilidades del aplicativo web y su ambiente.

La prueba de desempeño abarca requerimientos no funcionales como también pruebas diseñadas para valorar el aumento de trafico de usuarios tiempo respuesta y confiabilidad de la web, que componentes son responsables del colapso o limitación del software y de qué manera impacta los requisitos globales del aplicativo web.

2.3 Planeación de Pruebas.

Dos elementos claves son comenzar pronto la selección de los casos de prueba y hacer las pruebas en forma paralela. Hay que reconocer que una buena y cuidadosa planeación de

pruebas puede reducir el costo de las pruebas y el tiempo necesario para su terminación. La prueba (una actividad sistemática y planeada) requiere el mayor porcentaje del esfuerzo técnico en el proceso del software. A continuación se mostrará un conjunto de atributos técnicos de calidad que desea que contenga la aplicación web.

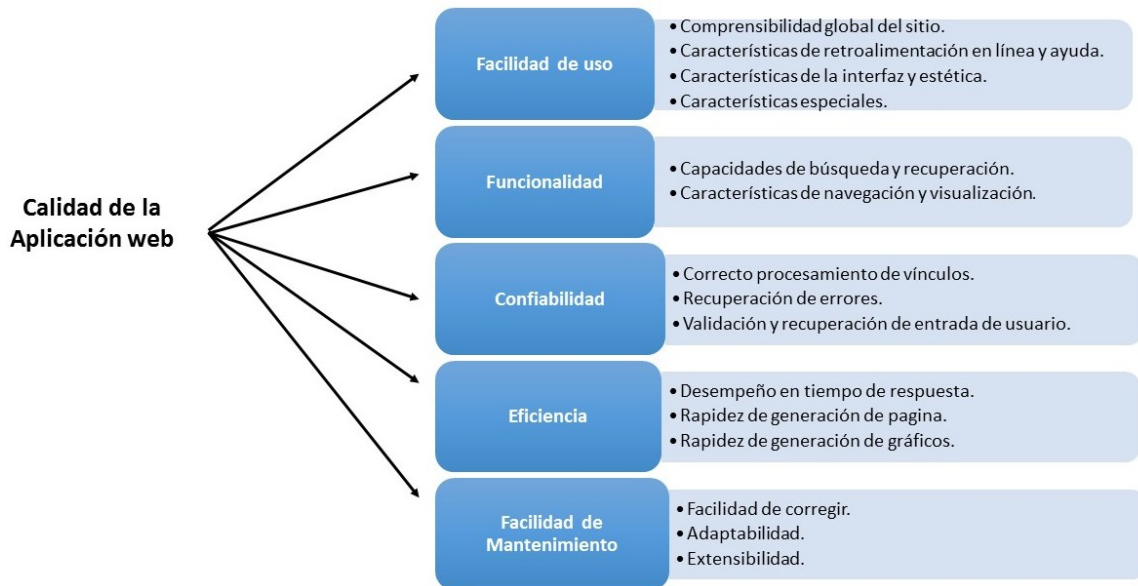


Figura 5 Olsina "Specifying Quality Characteristics and attributes for Web Sites"

La figura anterior según (Offutt, Marzo,2002) (Olsina, 1999) representa un árbol de requisitos de calidad proporcionando una base útil para valorar la calidad de un aplicativo web, y que conducen a un software de gran calidad, bajo un conjunto de atributos técnicos que son universales para todo el software.

2.3.1 Documentación de las Pruebas.

Las actividades de las pruebas se documentan en cuatro tipos de artefactos. Plan de pruebas, Especificaciones de casos de prueba, Reportes de incidentes de pruebas y Reporte de resumen de pruebas:

Plan de Pruebas: enfoca aspectos administrativos. Documentación del alcance, enfoque, recurso y calendarización de las actividades de pruebas. En este documento se especifica los requerimientos y componentes o funcionalidades a probar.

Especificaciones de casos de prueba: Este documento contiene todas las tareas a realizar, también contiene las entradas, manejadores, y las salidas que se esperan de las pruebas. Cada prueba se documenta con una especificación de caso de prueba.

Reportes de incidentes de pruebas: Se registran los datos reales obtenidos en las pruebas y las diferencias con respecto a la salida esperada. Es decir cada ejecución de la prueba se documenta.

Reportes de resumen de pruebas: Contiene todas las fallas que se descubrieron durante las pruebas y que necesitan una verificación. A partir de este documento se analiza y se asigna prioridad a cada falla y se planea los cambios al software y modelos.

Especificación del Caso de Prueba
1. Identificador del caso de pruebas.
2. Conceptos a probar.
3. Especificaciones de entrada.
4. Especificaciones de salida.
5. Necesidades ambientales.
6. Requerimientos procedurales ambientales
7. Dependencia entre casos

Tabla 1 Caso de Prueba [IEEE Std. 829-1991]

Los documentos descritos en esta sección sobre documentación de pruebas se basan en el estándar (IEEE Standar for Software Test Documentation, 1997).

2.4 Técnicas de Prueba de Software.

Una técnica de prueba define la estrategia para la selección de los casos de prueba, en esta técnica se diseña, se documenta un conjunto de casos de prueba que tenga una alta probabilidad de encontrar errores, para comprobar la lógica interna, las interfaces, los requisitos internos, relaciones entre componentes, se definen los resultados esperados y se registran los resultados reales obtenidos. NO solamente es suficiente lo anterior sino

también realizar Revisiones Técnicas Formales para evaluar la estrategia de prueba y los propios casos de prueba, para descubrir inconsistencias, omisiones y errores evidentes en el enfoque de la prueba. Por lo tanto, las revisiones ahorran tiempo, reducen la cantidad de esfuerzo que requiere las pruebas para producir un software de calidad (Beizer, 1990).

Existen dos maneras de probar cualquier software: las pruebas de caja negra y pruebas de caja blanca. Las pruebas de caja negra o también llamadas pruebas de comportamiento son las que se aplican a la interfaz del software enfocada en el comportamiento de entrada/salida del software o componente; las pruebas de caja negra no maneja aspectos internos del componente o software. Las pruebas de caja blanca o pruebas de caja de cristal se enfocan en la estructura interna del componente, se prueba cada estado del componente y la interacción entre sus objetos, es decir, que las operaciones internas funcionan de acuerdo con las especificaciones (Myers, 1979).

2.4.1 Casos de Pruebas

Los casos de prueba es un conjunto de datos de entrada y resultados esperados que prueban a un componente con el propósito de causar fallas y detectar defectos. Un caso de prueba consta de cinco elementos: nombre, ubicación, entrada, oráculo y bitácora (Binder, 1999).

Nombre: Permite que quien haga la prueba la distinga de otros casos de prueba existentes (*Prueba_Loguin*).

Ubicación: indica donde puede encontrarse al caso de prueba, puede ser una ruta o un URL hacia el ejecutable de la prueba y datos de entrada.

Entrada: conjunto de datos de entrada o comandos específicos para el caso de prueba.

Oráculo: Describe el comportamiento esperado.

Bitácora: Es la documentación de los resultados obtenidos tras haber ejecutado la prueba, comportamiento observado contra el esperado.

<i>Atributos</i>	<i>Descripción</i>
Nombre	Nombre del caso de prueba

Ubicación	Nombre de ruta completo del ejecutable
Entrada	Datos de entrada o comandos
Oráculo	Resultados esperados de la prueba contra los resultados de salida de la prueba
Bitácora	Salida producida por la prueba

Tabla 2 Elementos de Caso de Pruebas

2.4.2 Actividades de las Pruebas.

Inspección de componentes: Las inspecciones localiza fallas en un componente mediante la inspección de su código fuente, también se pueden realizar las inspecciones antes o después de las pruebas unitarias (Fagan, 1976).

Pruebas unitarias: Encuentra defectos aislando los componentes del software ejercitando cada componente con un caso de prueba. Hay tres características en el enfoque de los componentes, primero permite enfocarse en unidades más pequeñas del software, es decir, la prueba unitaria reduce la complejidad de las actividades de pruebas generales. Segundo facilita encontrar y corregir defectos ya que se examina pocos componentes. Tercero cada componente puede probarse en forma independiente de los demás, es decir, permite ejecutar actividades de prueba en paralelo (Myers, 1979).

Pruebas de integración: Una vez eliminado los defectos de cada componente se da por culminado las pruebas unitarias, ahora sus componentes están listos para su integración en subsistemas más grandes. Dos o más componentes se integran y se prueban, ante la usencia de un nuevo defecto se añaden nuevos componentes, permitiendo probar partes cada vez más complejas del software (Myers, 1979).

Pruebas del sistema: Este tipo de prueba asegura que el software completo se apege a los requerimientos funcionales y no funcionales del sistema. Al final el software se incorpora a otros elementos del sistema como (hardware, personas, información) es el objetivo de esta prueba ejercitar profundamente la totalidad del software para verificar que se hayan

integrado adecuadamente todos los elementos del sistema y que realicen las funciones apropiadas (Beizer B. , 1984).

2.5 Pruebas de Desempeño.

Este tipo de pruebas se diseñan con el fin de simular situaciones de carga y tensión del mundo real, para descubrir problemas como falta de recursos en el lado del servidor, base de datos, debilidades del sistema operativo, funcionalidades mal diseñadas en aplicativos web y otros conflictos de hardware y software que reducen el desempeño. También comprender como responde a la carga (número de transacciones o volumen de datos) para mejorar su desempeño (Pressman, 2005).

2.5.1 Pruebas de Carga.

El objetivo es determinar como el aplicativo web y el ambiente del lado del servidor responde a ciertas condiciones de carga, simulando una carga de trabajo similar y superior a la que tendrá cuando el sitio esté funcionando. Las siguientes variables definen un conjunto de condiciones de prueba dentro de los límites operativos normales del aplicativo web (Pressman, 2005):

N, es el número de usuarios concurrentes.

T, es el número de transacciones en línea por usuario por unidad de tiempo.

D, es la carga de datos procesada por el servidor por transacción.

Con cada condición de prueba ejecutada se puede extraer la siguiente información: la respuesta de usuario promedio, el tiempo promedio para descargar una unidad de datos estandarizada, o el tiempo promedio para atender una transacción. Valorar la velocidad de conexión requerida o recomendada para los usuarios de la aplicación web también hace parte de los objetivos de las pruebas de carga.

P, la cantidad de información global procesada en unidad de tiempo. Se calcula de la siguiente forma:

$$P=N*T*D$$

Como ejemplo, considere un sitio educativo de una universidad, en un determinado momento 20.000 usuarios concurrentes realizan una solicitud (una transacción T) una vez cada dos minutos en promedio. Cada transacción requiere que el aplicativo web descargue un nuevo contenido que promedia los 4 Kbyte de longitud. La cantidad de información procesada en una unidad de tiempo se puede calcular como:

$$P = [20000 * 0.5 * 4 \text{ Kb}] / 60$$

$$P = 500 \frac{\text{Kb}}{\text{seg}}$$

$$P = 0.5 \frac{\text{Mb}}{\text{seg}}$$

La conexión de red para el servidor tendría que soportar la tasa calculada de datos y se debe probar para garantizar cumple con lo requerido.

2.5.2 Pruebas de Tensión.

Según (Splaine & Jaskiel, 2001), la intención de estas pruebas es comprender de una mejor forma como falla un software conforme se presiona más allá de sus capacidades operativas, este tipo de pruebas se lleva hasta un punto máximo de capacidad.

También pueden ser consideradas una extensión de las pruebas de carga, donde se fuerzan el número de usuarios concurrentes, el número de transacciones en línea por usuario por unidad de tiempo, y la carga de datos procesada por el servidor por transacción, que corresponden a las variables N , T , D para hacer colapsar el software, y obtener información relevante como (Pressman, 2005):

- El servidor se desconecta cuando rebasa su capacidad.
- El servidor genera mensajes de alerta.
- Que valores para N , T , D se requieren para colapsar el ambiente del servidor. ¿Las transacciones se pierden conforme se rebasa la capacidad? La integridad de los datos se afecta cuando colapsa la capacidad del software. Si el software falla, ¿cuánto tiempo toma para estar en línea de nuevo?

Esto determinara la solidez de la aplicación en los momentos de carga extrema y ayuda a determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

2.6 Arquitectura de Aplicativo Web

Cuando se construye un aplicativo web se debe tener algunas consideraciones, como los datos de la aplicación se deben separar de los contenidos de la página, y dichos contenidos, deben estar claramente separados de la apariencia y de la interfaz gráfica (paginas). Se diseña una arquitectura que desacople la interfaz de la navegación y del comportamiento de la aplicación. La arquitectura modelo-vista-controlador desacopla la interfaz del usuario de la funcionalidad del aplicativo Web y del contenido de información, simplifica la implementación y mejora significativamente la reutilización. Donde la vista es la página HTML el esqueleto que soporta todas las funciones específicas de la interfaz y la presentación de contenido. El modelo contiene todo el contenido específico de la aplicación y objetos de contenidos, el modelo es el sistema de gestión de bases de datos y el acceso a fuentes de información externas que requiere la aplicación. El controlador gestiona el acceso al modelo y a la vista coordinando el flujo de datos entre ellos, los datos y las solicitudes se manejan mediante el controlador y de ahí se determina el tipo de solicitud, se transmite una solicitud de comportamiento al modelo, y el modelo accede a los datos almacenados que se necesiten. En otras palabras el controlador hace de intermediario entre *la vista y el modelo*.

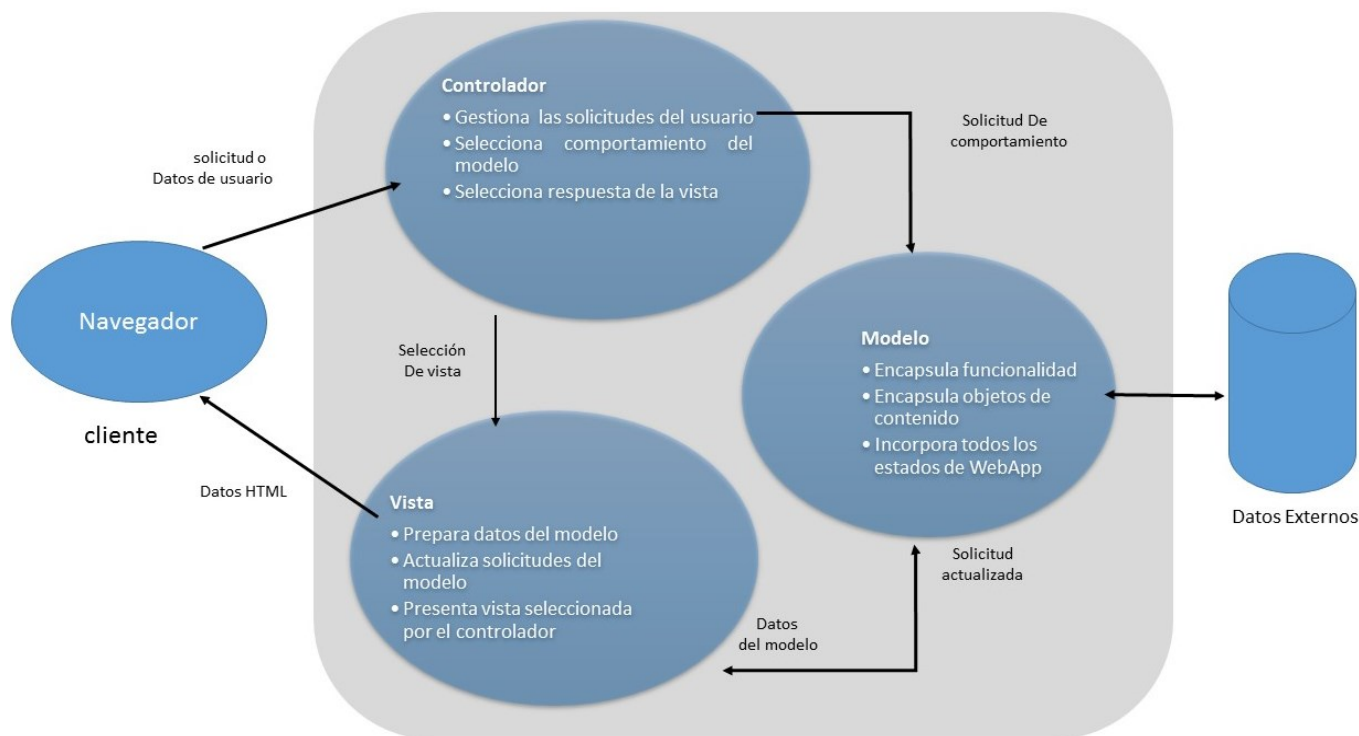


Figura 6 Jacyntho,D, "An Architecture for Structuring Complex Web Applications"

Se debe resaltar que la arquitectura de modelo-vista-controlador (Krasner, 1988) es un patrón de diseño arquitectónico desarrollado originalmente para el ambiente Smalltalk, para infraestructuras de aplicación web.

2.6.1 Arquitectura del Aplicativo Web (CIADTI)

La Universidad de Pamplona apoya sus procesos de negocio haciendo uso de la aplicación ACADEMUSOFT 3.2 esta a su vez integra un conjunto de aplicaciones y módulos para facilitar la realización de las actividades académicas, administrativas y financieras logrando eficiencia y eficacia a la hora de alcanzar los objetivos del macro proceso de Gestión Académica, haciendo uso de las TI que se encuentran disponibles actualmente y en constante evolución. Estas aplicaciones son desarrolladas por el Centro de Investigación Aplicada y Desarrollo en Tecnologías de Información (CIADTI), además de desarrollarlas,

el CIADTI brinda soporte técnico y el servicio de infraestructura tecnológica para dichas aplicaciones.

Academusoft “es una EAS (Enterprise Applications Solutions) para las Instituciones de Educación Superior, que ofrece una alternativa de alto nivel para el ingreso, organización, gestión y administración de la información en cada uno de los procesos Académicos y necesidades de negocio generadas por la Institución Educativa. Busca optimizar la generación de la información y apoyar a la alta gerencia en la toma de decisiones.” (Universidad de Pamplona, 2015)

Academusoft integra tres módulos que abarcan en su totalidad todas las funcionalidades que son utilizadas por los diferentes procesos y actividades en la gestión académica.

Hermesoft – Campus Colaborativo.

Academusoft - Campus Académico.

Gestasoft – Campus Administrativo.



Figura 7 Fuente. Documento de Infraestructura y soporte tecnológico (CIADTI)

2.6.2 Academusoft - Campus Académico

Academusoft – Campus Académico es un módulo dentro de la suite de Academusoft y es el encargado del apoyo a los procesos académicos como tal, Academusoft “apoya los procesos académicos de las Instituciones de Educación Superior desde el instante en que el aspirante expresa sus deseos de ingresar a la Universidad, la admisión como estudiante de la misma, como también todos los procesos académicos internos de cada institución, que se ajustan a la normatividad y autonomía de cada entidad educativa hasta la obtención del título como profesional.” (Universidad de Pamplona, 2015)

Academusoft-Campus Académico integra módulos que apoyan la gestión académica desde la inscripción del aspirante hasta su respectivo grado y seguimiento como egresado, está contenida de la siguiente manera:

- Admisiones
 - Inscripción
 - Selección
 - Inscripción en línea
- Registro Académico
 - Hoja de vida estudiante
 - Matricula Financiera
 - Horarios
 - Liquidación
 - Matricula Académica
 - Calificaciones
 - Grados
 - Pruebas de Ingles
 - Trabajo Social
 - Ecaes
 - App Estudiante
- Recursos Académicos
 - Carga Administrativa
 - Estructura Curricular

- Responsabilidad Académica
- Calendario Académico
- Recursos Físicos
- Documentos

2.6.3 Gestasoft – Campus Administrativo

Gestasoft – Campus Administrativo “es una excelente solución integral creada para administrar cuentas, gestionar información financiera, de proveedores y clientes, ordenar datos de inventarios, distribución y logística de cualquier organización de carácter público o privado, facilitando el manejo administrativo, financiero de las Instituciones de Educación Superior” (Universidad de Pamplona, 2015).

Gestasoft integra una serie de funcionalidades que apoyan los procesos de administración que abarca del macro proceso de gestión académica, a continuación se listan las funcionalidades que incorpora Gestasoft – Campus Administrativo.

- Gestión Talento Humano
 - Parámetros Generales
 - Nómina
 - Contratación
 - Talento Humano
- Gestión Financiera
 - Contabilidad
 - Presupuesto
 - Facturación y Cartera
 - Pagaduría y Tesorería
- Gestión de Servicios
 - Servicios Generales
 - Almacén e Inventario
 - Gestión Documental

2.7 Descripción del software JMeter

Apache JMeter es una aplicación Open Source Java diseñada para realizar pruebas funcionales medir el rendimiento y comportamiento en aplicaciones web, también se utiliza para probar el rendimiento tanto de los recursos estáticos y dinámicos (archivos, Servlets, scripts de Perl, objetos Java, bases de datos, servidores FTP). Con dicha herramienta se puede simular una carga en un servidor, la red o un objeto para poner a prueba su capacidad de resistencia y analizar el rendimiento global con diferentes tipos de carga. JMeter almacena los planes de pruebas en formato XML. Esto quiere decir que se puede generar un plan de pruebas utilizando un editor de texto.

Características de JMeter.

- Multiplataforma
- Licencia Open Source
- Extensible
- Marco de Trabajo Multi-hilo
- Visualización de resultados
- Soporte Multi Protocolo

Soporte Servidor/ Protocolo

- Web –HTTP,HTTPS
- SOAP / REST
- FTP
- Bases de Datos vía JDBC
- LDAP
- Correo –SMTP(S), POP3(S), IMAP(S)
- MongoDB (NoSQL)
- TCP

Sistema Operativos Compatibles.

Sistema Operativo	Java	Arquitectura
Free BSD 9.0	Open JDK 6	Amd64
Linux 2.4, 3.0, 3,2	Sun JDK 5,6,7 and Open JDK 6, Open JDK 7, Oracle JDK 7	i386, amd64
Mac OS	Oracle JDK 6, Oracle JDK 7, Apple JDK	64 bits
Windows 7	Oracle JDK 6, Oracle JDK 7	32/64 bits
Windows 8.1	Oracle JDK 7	64 bits

Tabla 3 Sistemas Operativos Compatibles para JMeter

Para instalar una versión de la herramienta Apache JMeter, simplemente descomprimir el archivo zip / tar en el directorio donde quiere que sea guardado JMeter. Tener JRE / JDK correctamente instalado junto con las variables de entorno JAVA_HOME.

Descomprimir el archivo, estructura de carpetas:

- /bin: Contiene scripts para iniciar JMeter.
- /docs: Documentación JMeter.
- /lib/: Contiene las librerías Java requeridas por JMeter.
- /lib/ext: Contiene los archivos jar para los protocolos JMeter.
- /lib/junit: Contiene la librería JUnit usada por JMeter.

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

- jmeter | jmeter.bat: Corresponde a los ejecutables de la interface principal para plataformas Linux y Windows.
- jmeter-server | jmeter-server.bat: Representan los ejecutables para el emulador de Servidor JMeter para plataformas Linux y Windows.
- jmeter.properties: Contiene propiedades de arranque para JMeter que son utilizadas por cualquiera de sus ejecutables.

- `jmeter.log`: Representa los registros ("logs") generados al ejecutar JMeter.
- `users.xml` | `users.dtd`: Un archivo XML y su correspondiente DTD, empleados para definir características y configuraciones de usuarios que serán simulados por JMeter.

2.7.1 Componentes del Plan de Pruebas

Un Plan de Pruebas (Test Plan), es el componente principal de JMeter en él se definen todos los aspectos relacionados con la prueba de carga, tales como: el tipo de reportes que se desea generar, parámetros empleados por requisición. Consta de los siguientes ítems:

- Threads users (Hilos Usuarios) Thread Groups (Grupo de Hilos).
- Login controllers (Controladores lógicos).
- Listeners (Receptores)
- Timers (Temporizadores)
- Assertions (Afirmaciones - Aserciones)
- Configuration elements (Elementos de configuración)

2.7.2 Grupo de Usuarios

Representa el conjunto de usuarios que se desea simular en el Plan de Pruebas (Test- Plan) estos son los que realizan las peticiones al aplicativo web, es decir desde el componente Grupo de Hilos se controla las características de los hilos tales como: la cantidad de hilos que se van a ejecutar en la prueba (número de usuarios), establecer el periodo de tiempo, establecer el número de iteraciones que se desea ejecutar la prueba, y bajo que especificaciones.

2.7.3 Controladores

JMeter cuenta con dos tipos de controladores: Muestreador y Controladores lógicos

Muestreadores: Permiten enviar determinados tipos de peticiones a un servidor determinado, simulando una petición de usuario al aplicativo web alojado en el servidor destino; los muestreadores incluyen:

- Solicitud de FTP.
- Solicitud HTTP.

- Solicitud JDBC.
- Java Object Request.
- Solicitud de LDAP.
- Solicitud SOAP / XML-RPC
- Solicitud Webservice (SOAP)

Como también indicar información asociada al servidor como nombre, puertos, rutas y protocolos.

- Controladores Lógicos: Permite personalizar la lógica que JMeter utiliza para decidir cuándo se envía las solicitudes, establecer condiciones, intervalos, controlador de tiempo de ejecución, controlador de rendimiento y repeticiones.

2.7.4 Fragmentos de Prueba

Se distingue del grupo de hilos en que no se ejecuta a menos que se adicione el elemento Include Controller. Es un tipo de prueba especial de controlador, su objetivo principal es la reutilización de código dentro del plan de pruebas.

2.7.5 Receptores

Estos componentes se pueden añadir en cualquier lugar de la prueba, recogen todos los datos de las pruebas únicamente de elementos iguales o inferiores a su nivel. Muestra los detalles de las solicitudes, de todas las respuestas de los servidores, sus tiempos, puede mostrar los datos obtenidos, en gráficas, en tablas, documentos HTML y XML, así como dirigir los datos a un archivo plano para su uso posterior.

2.7.6 Temporizadores

Los hilos de JMeter envían solicitudes sin hacer pausa, es decir simula usuarios simultáneos enviando peticiones sin pausa entre cada solicitud, esto se puede controlar con este componente, se agrega un temporizador que permite definir un periodo de espera entre cada solicitud dependiendo los requerimientos de la prueba. A continuación los temporizadores que ofrece JMeter:

- Temporizador constante.
- Gaussiano temporizador aleatorio.

- Uniforme temporizador aleatorio.
- Rendimiento constante temporizador.
- Sincronización temporizador.
- Tiempo JSR223.
- Tiempo BeanShell.
- Tiempo BSF.
- Poisson hora aleatoria.

2.7.7 Afirmaciones

Generalmente se le añade al muestreador, se utiliza para confirmar que la aplicación devuelve los resultados que el ingeniero a cargo de la prueba espera, recibir la validación acerca de las respuestas recibidas desde el servidor que se está probando. A continuación las Afirmaciones con las que cuenta JMeter.

- Beanshell aserción.
- BSF aserción.
- Comparar aserción.
- JSR223 aserción.
- Respuesta de aserción.
- Duración de aserción.
- Tamaño de aserción.
- Afirmación XML.
- BeanShell aserción.
- MD5Hex aserción.
- HTML aserción.
- XPath aserción.
- Esquema XML aserción.

2.7.8 Elementos de configuración

Permite crear valores por defecto y las variables que se utilizaran por los elementos de configuración modificando las peticiones que se desean ejecutar. Tiene prioridad sobre un

muestreador, un elemento de configuración se accede solo desde el interior de la rama en la que se colocó. A continuación los elementos de Configuración con las que cuenta JMeter:

- Configuración del almacén de claves.
- Configuración de la conexión JDBC.
- configuración de conjunto de datos CSV.
- Contador.
- La petición FTP predeterminados.
- Administrador de autorización HTTP.
- Administrador de caché HTTP.
- Administrador de cookies HTTP.
- Servidor proxy HTTP.
- Valores por defecto HTTP.
- Administrador de encabezado HTTP.
- Solicitar Java predeterminados.
- Solicitud LDAP predeterminados.
- Elemento configuración simple.
- Valores por defecto LDAP.
- TCP Sampler Config.
- Las variables definidas por el usuario.
- Elemento de configuración sencilla.
- Variable aleatoria.

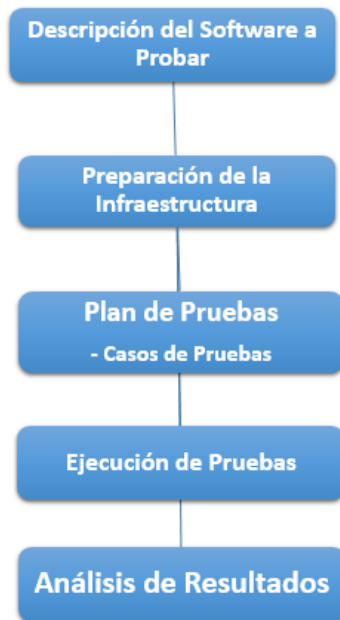
3 Procedimiento propuesto para pruebas de Rendimiento

Con el siguiente procedimiento se pretende documentar los pasos necesarios que se deben aplicar en un proceso de pruebas, de manera que se mida la capacidad de respuesta del sistema cuando es sometido a condiciones extremas operacionales, tales como picos de usuarios, o flujo de datos, manteniendo el desempeño esperado para el que fue diseñado, garantizando la confiabilidad del sistema en un ambiente de producción.

El procedimiento se diseñó para evaluar la capacidad y estado actual de gestión de una de las aplicaciones utilizadas en la Universidad de Pamplona, seguidamente se presentó una propuesta de un modelo de pruebas y por último se hizo la validación y ejecución de ese modelo aplicando herramientas tecnológicas de pruebas de desempeño (carga tensión) JMeter.

Se tomó una funcionalidad dentro del aplicativo web seleccionad, con el fin de recopilar información de su desempeño, fallas, puntos críticos y cómo se comporta frente al colapso y de qué manera se puede corregir los defectos encontrados y mejorar el desempeño de la funcionalidad. El análisis de los datos obtenidos es de suma importancia para saber qué puntos a favor y fallas el software manifiesta, en caso tal empezar un ágil proceso de depuración.

3.1 Descripción del procedimiento



Para iniciar el desarrollo del procedimiento, se toma funcionalidades orientadas a gran concurrencia de datos, en instituciones de educación superior. Para conocer el proceso, es necesario leer documentos, manuales y realizar algunas pruebas funcionales que permite visualizar la respectiva orientación de cómo debe ser el comportamiento de la aplicación con respecto a los usuarios y los resultados que de ella se esperan. Una vez identificada las funcionalidades más complejas dentro del proceso en cuanto a importancia y concurrencia por parte de los usuarios, se seleccionan para proceder a realizar las respectivas pruebas de carga y rendimiento.

Se parametriza el software de gestión que se usa para llevar a cabo dicho proceso de pruebas, teniendo en cuenta que se debe crear un plan de pruebas, que consiste en identificar las funcionalidades a probar, establecer los tipos de pruebas (desempeño, carga y tiempo de respuesta) que se realizan, las capacidades físicas del software y las fechas en las que se aplicarán. Para cada una de las funcionalidades seleccionadas se debe documentar cada uno de los casos de pruebas, teniendo como base los datos de entrada y los resultados esperados y obtenidos que hacen referencia al rendimiento del software. Para cada funcionalidad se estableció como valor mínimo dos casos de prueba. Este se ingresa a la herramienta. Posteriormente se define la cantidad de usuarios que van a acceder a la

aplicación Web de manera simultánea y el lapso de tiempo que van a hacer llevadas a cabo las solicitudes de los usuarios, esto se ingresa en la herramienta. Estos parámetros permiten por cada tipo de prueba que se aplique a cada una de las funcionalidades definir el umbral que indica si una funcionalidad alcanza el nivel de aceptación, teniendo en cuenta la unidad de medida para cada una de ellas. El nivel del umbral se toma de la población total de usuarios accediendo simultáneamente a la aplicación, si el porcentaje definido de las peticiones son atendidas significa que se logra el nivel de aceptación. Después se ingresa la ruta donde se encuentra la funcionalidad a probar, definida previamente en el plan de pruebas, esta se ingresa a la herramienta a esta petición se le configura los casos de excepción y se configuran los parámetros que requiera la funcionalidad para operar. Se añade el tipo de formato que se quiere para el informe final de los resultados y se procede a ejecutar en la herramienta.

De los datos obtenidos en la herramienta se descarga el informe y se procede a su respectivo análisis, verificando tiempos de respuesta, tamaño de la cabecera en bytes, los estados de la solicitud, analizando la media de respuesta que arroja la herramienta respecto a tiempo vs cantidad de usuarios.

3.1.1 Descripción del Software a probar

Para configurar los componentes necesarios para llevar a cabo un plan de pruebas y posteriormente su respectiva ejecución a una funcionalidad específica del aplicativo web. Simulando el comportamiento que puede tener determinado cliente o usuario, también es posible realizar requisiciones para que cada solicitud contenga parámetros únicos por usuario, de esta manera permitiendo simular clientes específicos. Se puede obtener los resultados de manera gráfica teniendo como parámetros de entrada tiempo vs número de peticiones, o también datos de estado y componentes, y en forma de tablas y datos estadísticos.

3.1.2 Preparación de Infraestructura

Se indicara los pasos más relevantes para la configuración y uso herramienta tecnológica, y componentes para llevar a cabo un plan de pruebas.

Como toda aplicación WEB, antes de iniciar su operación, se hace necesario definir el espacio donde esta se va a almacenar, razón por la cual se debe realizar la respectiva preparación en cuanto a recursos físicos se requiere con el fin de obtener un espacio y puerto asignado dentro de los lineamientos establecidos por la institución en su infraestructura tecnológica. La preparación de la infraestructura del servidor donde estará instalado el software bajo prueba comprende:

Solicitud de Espacio en un servidor determinado: Máquina física donde se almacena la aplicación, con dirección IP como resultado.

Solicitud del puerto de publicación: Puerto que se usa dentro de dicha máquina para realizar la publicación.

Identificar las ubicaciones de los archivos: Ruta donde se van a colocar los componentes desarrollados que van a ser probados.

Almacenar los Archivos: Copiar los archivos correspondientes a los componentes desarrollados y que están listos para pruebas en la ruta donde se va a alojar la aplicación a probar.

Realizar configuraciones del Publicador: Realizar las respectivas configuraciones técnicas necesarias para que el publicador quede funcionando, incluye modificaciones de los recursos y direcciones de acceso según indicaciones de desarrollo en el server.xml

Una vez se han establecido y configurado todos los pasos necesarios para la preparación de la infraestructura requerida para el alojamiento de la aplicación, se debe informar al personal que realiza las pruebas las direcciones y puertos por los que se debe acceder.

Infraestructura del cliente: Equipo de Pruebas debe contar con unos requisitos mínimos como memoria RAM de 2Gb, Procesador no menor a un dual Core, espacio en disco duro como mínimo 120 Mb para la instalación de la herramienta tecnológica que va llevar a cabo

el proceso de pruebas de rendimiento y el Sistema Operativo que sea compatible con dicha herramienta:

La herramienta no necesariamente se debe instalar al lado del servidor si no en un equipo remoto siempre y cuando tenga acceso al aplicativo web, este se debe configurar para permitir el acceso y el que va utilizar, se debe configurar el navegador con el cual se desea acceder al aplicativo, se configura el proxy HTTP, puerto, para ejecutar las pruebas de carga y rendimiento. También es necesario la base de datos, el servidor de aplicaciones y el aplicativo web que se va a poner a prueba.

3.1.3 Diseño Caso de Prueba

Para dar inicio a la ejecución del plan de pruebas y de esta manera determinar si el sistema es operable bajo un conjunto de condiciones establecidas, se debe establecer inicialmente cuales son los casos de pruebas que aplican a la funcionalidad a probar, para ello se pueden definir diferentes escenarios que permitan validar todos los posibles caminos que conllevarían a un resultado esperado dentro del proceso de pruebas.

Para lograr el objetivo anterior se establece un proceso para la definición del caso o los casos de pruebas que permita verificar los diferentes escenarios a los que puede ser sometida la funcionalidad, para ello debemos saber que el diseño del caso de pruebas debe contemplar:

- 1. Código del caso de prueba:** número asignado que identifica unívocamente cada uno de los casos planteados, dicha codificación puede estar sujeta a verificación se sistema de aseguramiento de la calidad;
- 2. Nombre o Funcionalidad a probar:** Permite identificar el caso e indica a cual módulo de la aplicación se le aplicará;
- 3. Ruta de la funcionalidad a probar:** Recorrido que debe realizar la persona dentro de la aplicación para acceder a la funcionalidad;
- 4. Descripción de la funcionalidad:** Permite dar una visión global de lo que hace el modulo a revisar, su contenido base debe estar sustentado en el requerimiento realizado por el cliente;

5. Descripción del Caso: Indica cómo se va a aplicar la prueba, estableciendo el paso a paso de la prueba desde el ingreso de los datos de entrada hasta la obtención del resultado final;

6. Parámetros de entrada: Hace referencia a la información que requiere la funcionalidad para operar;

7. Precondiciones: Información previa que debe estar gestionada en la aplicación para que el caso se puede aplicar según las especificaciones dadas en el requerimiento;

8. Resultado de la prueba: Indica si el caso de prueba es exitoso o no y que sucede con el sistema después que el caso se aplica.

Generalmente el diseño de estos tipos de casos, son diseños conceptuales que permiten establecer la información básica necesaria para poder ejecutar una prueba específica, razón por la cual se incluye dentro de la plantilla que se plantea en este trabajo.

3.1.4 Ejecución de la prueba

Para llevar a cabo la ejecución de las pruebas se requiere completar todos los requerimientos necesarios antes mencionados para que no queden datos faltantes ya que esto puede generar errores futuros en la obtención de los resultados de las pruebas, comienza con la preparación del ambiente de prueba se define un ambiente controlado para la ejecución de las pruebas, se procede a la iteración de las pruebas, ejecución de los requerimientos de la prueba, un registro de hallazgos, se lleva un control y seguimiento de los indicadores de las pruebas un informe de avance de las pruebas y un informe de cierre de las pruebas para una retroalimentación y su posterior análisis de resultados.

3.1.5 Análisis de Resultados

Indicadores de rendimiento como son: el tiempo de respuesta, estado de la petición, el porcentaje de transacciones fallidas o el número de usuarios concurrentes, la tasa de operaciones/seg, el tipo de peticiones, son indicadores que deben ser integrados para

estudiar la capacidad del software. Con ayuda de la herramienta se puede generar un informe de pruebas de resultados de manera automática, como graficas de tiempo, tablas de rendimiento que permiten evaluar diferentes indicadores de las pruebas, para identificar los posibles puntos de saturación y detectar "cuellos de botella".

3.2 Aplicación del procedimiento propuesto

El propósito de este procedimiento es la comparación de dos versiones diferentes de una misma aplicación software, por lo tanto los resultados de las pruebas mostrara cuál de las dos versiones es más estable y tiene un mejor desempeño cuando se fuerza al quebrantamiento bajo ciertos parámetros.

Para el desarrollo del proyecto, se utilizaron diferentes materiales, libros, software libre, herramientas tecnológicas y computadores de escritorio a través de los cuales se diseñó planifico y ejecuto los casos de pruebas en funcionalidades del aplicativo web (CIADTI) de la Universidad de Pamplona.

3.2.1 Descripción del Software a Probar

El aplicativo web Academusoft contiene todos los procesos académicos involucrando estudiantes y egresados, gestionando procesos desde la inscripción del aspirante hasta su respectivo grado y seguimiento como egresado, actualmente la versión que se encuentra disponible al público es la 3.2, con la que los usuario pueden interactuar, sin embargo ya existe una versión más actual que corresponde a la 4.0. Con la ayuda de la herramienta tecnológica para el diseño y ejecución de pruebas de carga y rendimiento se valida el funcionamiento de dichas versiones de Academusoft bajo ciertas condiciones, simulando un enjambre de usuarios realizando peticiones al aplicativo web. A continuación se mostrara la interfaz gráfica de las dos versiones (3.2 y 4.0) las cuales componen Academusoft, estas son:

Académico AcademuSoft

AcademuSoft 3.2, la función que intenta ejecutar esta deshabilitada OBJETOS

Evite que esta página cree registros de diseño adicionales.

Lista de Grado Central

Unidades Regionales

<input type="radio"/> UNPAMPLONA REGIONAL 1 (ME)	RIOHACHA	Aceptar
<input type="radio"/> UNPAMPLONA REGIONAL 2 (ME)	VALLEDUPAR	
<input checked="" type="radio"/> UNIVERSIDAD DE PAMPLONA	PAMPLONA	
<input type="radio"/> UNIVERSIDAD NORTESANTANDEREANA	PAMPLONA	
<input type="radio"/> URIBIA	URIBIA	
<input type="radio"/> VALERA	VALERA	
<input type="radio"/> VALLEDUPAR	VALLEDUPAR	
<input type="radio"/> VIRMI	VIRMI	

Periodos Académicos

Año	Periodo	Fecha Inicio	Fecha Fin	Tipo Periodo Académico
Lista de Periodos: 01 - 15 de 58				
<input type="radio"/> 2016	IT	11-01-2016	24-12-2016	ANUAL
<input type="radio"/> 2016	1	24-01-2016	30-06-2016	SEMESTRAL
<input checked="" type="radio"/> 2015	1	01-06-2015	19-12-2015	SEMESTRAL
<input type="radio"/> 2014	2	03-02-2014	31-05-2015	SEMESTRAL
<input type="radio"/> 2014	1	20-01-2014	20-12-2014	ANUAL
<input type="radio"/> 2014	1	07-07-2014	26-12-2014	SEMESTRAL
<input type="radio"/> 2013	IT	01-01-2013	15-09-2013	ANUAL
<input type="radio"/> 2013	1	02-01-2013	30-12-2013	SEMESTRAL
<input type="radio"/> 2012	2	09-07-2012	31-08-2012	SEMESTRAL

Ceremonias de Grado

Número del Acto	Lugar de Ceremonia	Fecha y Hora de Inicio	Fecha y Hora de Fin
<input checked="" type="radio"/> 82015	AULAS - II 201	15-11-2015 11:09:27	15-12-2015 11:09:32

Agregar Continuar

Volver Ayuda Menú Menú Volver Salir de la

Figura 8 Aplicativo Web v.3.2 (AcademuSoft)

Academusoft Académico 4.0 - Administrador AcademuSoft®

Bienvenido, ALBERTO CAMILO MARTINEZ GALARZA, es_CO

Inicio Datos de Gra...

- Liquidaciones
- Proyectar Demanda
- Horarios
- Matrícula Académica
- Calificaciones
- Gestionar Sanciones
- Estímulos
- Grados
 - Plantilla Acta de Grado
 - Verificar Requisitos de Grado
 - Gestionar Grados
 - Datos de Grado General
 - Firmas de Acta de Grado General
 - Datos de Grado por Programa
 - Datos de Grado Estudiante
 - Resolución de Grado Automática
 - Graduar Estudiantes
 - Acta de Grado General
 - Acta de Grado Estudiante
 - Reversar Proceso de Graduación
- Parametrizar Rango de Registros
- Gestionar Porcentaje de Aprobación
- Activar Estudiante
- Gestionar Convocatoria
- ???functionality.02_11_08???
- ???functionality.02_11_09???
- ???functionality.02_11_10???
- ???functionality.02_11_11???
- ???functionality.02_11_12???

Datos de Grado General

Unidades Regionales

Mostrar: 10 registros Buscar: PAMPLONA

	Nombre	Ubicación
<input type="radio"/>	CENTRO DE EDUCACIÓN VIRTUAL Y A DISTANCIA	PAMPLONA
<input type="radio"/>	INSTITUTO NACIONAL DE ENSEÑANZA MEDIA	PAMPLONA
<input type="radio"/>	PAMPLONA	PAMPLONA
<input type="radio"/>	UNIDAD GENERAL UNIVERSIDAD DE PAMPLONA	PAMPLONA
<input type="radio"/>	UNIPAMPLONA REGIONAL 1 (ME)	RIOHACHA
<input type="radio"/>	UNIPAMPLONA REGIONAL 2 (ME)	VALLEDUPAR
<input checked="" type="radio"/>	UNIVERSIDAD DE PAMPLONA	PAMPLONA
<input type="radio"/>	UNIVERSIDAD NORTESANTANDEREANA	PAMPLONA

Mostrando registros del 1 al 8 de un total de 8 registros (filtrado de un total de 148 registros) Anterior 1 Siguiente

Períodos Académicos

<input type="radio"/>	2008	2	15-07-2008	30-12-2008	SEMESTRAL
<input type="radio"/>	2009	11	15-02-2009	29-07-2009	CICLO
<input type="radio"/>	2010	1	15-02-2010	30-06-2010	ANUAL
<input type="radio"/>	2013	1T	01-01-2013	15-09-2013	ANUAL
<input type="radio"/>	2014	2	03-02-2014	31-05-2015	SEMESTRAL
<input checked="" type="radio"/>	2015	1	01-06-2015	19-12-2015	SEMESTRAL

Mostrando registros del 1 al 6 de un total de 6 registros (filtrado de un total de 58 registros) Anterior 1 Siguiente

Universidad de Pamplona - Centro de Investigación y Desarrollo en Tecnologías Aplicadas - Todos los Derechos Reservados © 2013

Figura 9 Aplicativo Web v.4.0 (Academusoft)

Datos de Grado General es una de las funcionalidades que se pone a prueba, la figura 15 que corresponde a la versión 3.2 y la figura 16 a la versión 4.0 dichas funcionalidades está contenidas en Academusoft.

3.2.1.1 Interacción de Componentes en el Aplicativo Web

1. El usuario ingresa a la dirección de la aplicación Web.
2. Interactúa con la interfaz (pulsar un botón, enlace) o realiza una petición.
3. El controlador recibe la notificación de la acción (por medio de los objetos interfaz- vista), el controlador gestiona la transacción solicitada por el usuario normalmente a través de un gestor de eventos (handler) o callback.

4. El controlador accede al modelo, se actualiza dependiendo la solicitud del usuario, encapsula las acciones y simplifica su extensión.
5. La vista obtiene los datos del modelo para generar la interfaz actualizada para el usuario, donde se verán los cambios reflejados.
6. Termina la transacción o solicitud del usuario siendo reflejada en la interfaz.
7. La interfaz de usuario está a la espera de nuevas interacciones por parte del usuario.
8. el ciclo comenzara nuevamente.

Cada vez que un nuevo usuario interactúa con el software o aplicativo web es el inicio de una prueba más, ya que con la entrada de nuevos datos de usuarios y peticiones es una configuración nueva y se pone a prueba el comportamiento del software.

3.2.2 Preparación de infraestructura

Los materiales usados se describen a continuación en la siguiente tabla.

#	Nombre	Justificación
1	JMeter	Software utilizado para realizar pruebas de carga y rendimiento, con resultados estadísticos.
2	Test Studio	Software Utilizado para realizar pruebas de rendimiento, carga y pruebas automáticas.

Tabla 4 Software

#	Nombre	Justificación
1	Miller, E. "WebSite Testing", 2000	Documentación sobre pruebas en aplicativos web
2	Nguyen, H. "Testing Web-Based Applications", 2000	Documentación sobre pruebas en aplicativos web
3	Nguyen, H. " Testing Applications on the Web "	Documentación sobre pruebas en aplicativos web
4	http://jmeter.apache.org/	Documentación y descarga de la Herramienta tecnológica para la

		ejecución de pruebas
5	http://www.telerik.com/teststudio	Documentación y descarga de la Herramienta tecnológica para la ejecución de pruebas

Tabla 5 Material Bibliográfico

<p>Procesador: Intel Core i5 de tercera generación</p> <p>RAM: 6 GB (Crecimiento a 12 GB)</p> <p>Una salida de audio, una entrada para micrófono, una entrada para parlantes</p> <p>Garantía 3 años</p>

Tabla 6 computador de escritorio (Equipo para uso)

Documento de infraestructura y soporte tecnológico para procesos de renovación y registros de programas académicos. CIADTI [docx]. p.14

<p>Processor: Intel® Core™ i5-3210M Processor 2.50 / 3.1 GHz (3MB L3 Cache, Chipset Mobile Intel HM76)</p> <p>Pantalla: 14.0" LED HD Antirreflejo 1366 x 768</p> <p>RAM: 6GB DDR3 1333 MHz (1DIMM), max a 8 GB</p> <p>Disco duro: 500 GB 5400 rpm</p> <p>Quemador: DVD RW</p> <p>Tarjeta de red: (10/100/1000 NIC)</p> <p>Conectividad adicional: Bluetooth</p> <p>Batería: 6-cell (47 WHr) Li-Ion battery, Hasta 4.5 hrs.</p> <p>Garantía de 3 años</p>
--

Tabla 7 Computador Portátil

Servicios
Tomcat
Virtualizadores VMWare
Apache
FTP
Proxy
SMB
DNS
NTP
DHCP
NFS
NAT
Sendmail (backups)
Openfire
Office 365
Moodle
Virtual Host
Ice cast
Livezilla
Open Jurnal
Enmascaramientos
Portal Cautivo

Tabla 8 Servicios

3.2.3 Diseño caso de prueba

La construcción del plan de pruebas que describe los pasos que se ejecutara con la herramienta son los siguientes:

1. Caso de Prueba.
2. Nombre de la funcionalidad.

3. Ruta de la funcionalidad.
4. Añadir y eliminar elementos de prueba.
5. Configuración de los elementos, lo que permite configurar el comportamiento de un determinado elemento de prueba.
6. Guardar el plan de pruebas antes de ejecutarlo.
7. Ejecución del plan de pruebas.
8. Detención de una prueba.
9. Informe de errores: este informe muestra las advertencias y los errores encontrados en el archivo jmeter.log, información sobre la prueba de funcionamiento. Además, se puede realizar un análisis gráfico de rendimiento o probar el comportamiento de un objeto, secuencia de comandos o servidor, bajo carga simultánea.

Se inicia la interfaz gráfica, se añade el primer y principal elemento que es el plan de pruebas, se le asigna un nombre relacionado al tipo de prueba. A continuación se crea el Plan de Pruebas y la opción añadir usuarios y configuración de la cantidad de hilos, va a ser de 100 usuarios, y el periodo de subida 10 segundos, esto significa 10 usuarios por segundo, también se habilita la opción “continuar” en caso de un error de muestreador, esto es cuando una solicitud de un hilo o usuario no logra ser atendida que proceda con la siguiente solicitud en cola, es decir que continúe la simulación. También se selecciona la opción contador de bucle esto es para indicar cuantas veces quiere que se repita la simulación del grupo de hilos en este caso solo serán dos iteraciones.

Grupo de Hilos	
Nombre:	Grupo de Hilos
Comentarios	
Acción a tomar después de un error de Muestreador	
<input checked="" type="radio"/> Continuar <input type="radio"/> Comenzar siguiente iteración <input type="radio"/> Parar Hilo <input type="radio"/> Parar Test <input type="radio"/> Parar test ahora	
Propiedades de Hilo	
Número de Hilos	100
Periodo de Subida (en segundos):	10
Contador del bucle: <input type="checkbox"/> Sin fin	2

Figura 10 Configuración Grupo de Usuarios

Se Añade el elemento de configuración de Valores por defecto para petición HHTP, cuando se comparte la misma url y configuración de las funcionalidades que se van a poner a prueba, en este caso se utilizara, se procede añadir.

Se Agrega la Petición HHTTP que se desea realizar, en el campo Ruta (Path) indicar la ruta o la dirección de la funcionalidad a probar. Se agrega la url donde se encuentra alojada la funcionalidad del aplicativo web a probar, también el parámetro puerto (port) indica el puerto en el que escuchará las peticiones a grabar, en este caso será el 8081 (posteriormente se configura el navegador que navegue a través de ese puerto), se selecciona la implementación HHTTP:HttpClient4 que es la versión que proporciona apoyo para la comunicación del robusto protocolo HTTP, HttpClient que proporciona un amplio y elegante Framework para realizar las tareas de comunicación HTTP en el lado cliente. Se hará una petición GET. Se configura el navegador que se desee para que navegue a través de un proxy, por ejemplo para FireFox, deberá dirigirse al menú Herramientas – Opciones – Avanzado – pestaña Red:

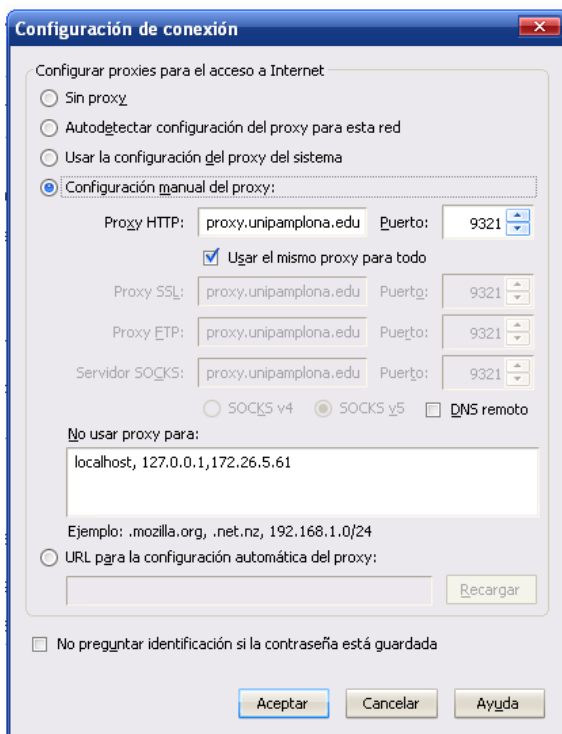


Figura 11 Configuración proxy del navegador Firefox

Banco de Trabajo se añade los Elementos NodePrueba y el Servidor Proxy HHTP. Puerto que va a capturar para realizar la grabación (ingresamos el Puerto de la aplicación). Controlador Objetivo: maneja el destino de todos los HTTP Request que se crearan durante la grabación del escenario. La opción: Plan de Pruebas y Grupo de Hilos, determina como quedaran agrupados los HTTP Request que se crearan durante la grabación. Simplemente

proporciona un lugar para almacenar temporalmente los elementos de prueba mientras no esté en uso, con fines de copiar / pegar, o cualquier otro propósito que se desee.

Como se está probando la aplicación en una máquina local, también es necesario usar el proxy, por lo que deberás borrar el contenido del cuadro de texto No usar proxy para.

Los resultados obtenidos durante y después de ser ejecutada la prueba de carga sobre el Grupo de Usuarios receptores hasta este punto arrojará varias opciones, en este caso se utilizó:

- Ver Árbol de Resultados: Muestra un árbol de todas las respuestas de la muestra, permite ver el estado de la petición y sus atributos, información de la cabecera de respuesta del muestreador. Además se puede ver una descripción de la estructura en HTML de la composición de la página de la funcionalidad que se puso a prueba en la pestaña Datos de Respuesta.
- Reporte Resumen: Crea una tabla y una fila para cada solicitud de forma diferente, en su prueba se muestra mínimo, máximo, promedio, la tasa de errores, rendimiento aproximado (petición / segundo) y kilobytes por segundo, el rendimiento es real a durante el transcurso de toda la prueba.
- Ver resultados en Árbol: Este receptor crea una fila por cada resultado de la muestra, se visualiza el estado de la prueba, el tamaño del hilo en bytes, el nombre, la hora en que comienza cada hilo, y lo que duro el tiempo de la muestra en milisegundos (msg) y la cantidad de hilos.
- El Gráfico Resultados: Genera un gráfico que traza todos los tiempos de muestreo. A lo largo de la prueba, la muestra actual color (negro), el promedio actual de todas las muestras color (azul), la desviación estándar actual color (rojo), y la tasa de rendimiento actual color (verde) se muestran en milisegundos.

El número de rendimiento representa el número real de solicitudes / minuto el servidor.

3.2.4 Ejecución de la Prueba

Para realizar la ejecución de las pruebas es necesario completar los requerimientos tanto de infraestructura tecnológica y los requerimientos no funcionales que se desean alcanzar de la aplicación web mencionados anteriormente, de esta manera garantizar correctamente los resultados de las pruebas, con la preparación lista del ambiente de prueba donde se encuentra alojada la aplicación web y configurado el equipo de pruebas que es el responsable de dicho proceso, una vez establecido el ambiente controlado para la ejecución de las pruebas, el tester o ingeniero a cargo procede a la ejecución de las pruebas utilizando la herramienta tecnológica, teniendo en cuenta como parámetros de entrada la ruta donde se encuentra alojada la funcionalidad, la cantidad de usuarios establecidos, el tiempo(sg) que va a durar la prueba, la versión de la aplicación, se lleva un control y seguimiento de los datos obtenidos de las pruebas, un informe para una retroalimentación y su posterior análisis de resultados.

3.2.5 Análisis de Resultados

En la institución se ofrecen 56 carreras en pregrado, 15 especializaciones, y 11 maestrías. Cada uno de estos programas se agrupa en una facultad. La institución cuenta en total, con siete: Artes y Humanidades, Ciencias Agrarias, Ciencias Básicas, Ciencias de la Salud, Ciencias Económicas y Empresariales, Ciencias de la Educación, Jurisprudencia y Ciencias Políticas, y por último, Ingeniería y Arquitectura.

Población Estudiantil (%)		
Pregrado	28.370	94,35(%)
Postgrado	1696	5,64(%)
Total	30066	100(%)

Se tuvo en cuenta estudiantes presenciales y a distancia, de las distintas sedes que hacen parte de la universidad de Pamplona.

Para seleccionar la cantidad de usuarios que se desea simular para la ejecución de pruebas se consultó los datos más recientes de los graduandos de los periodos comprendidos entre el 2015 y 2016. En el segundo periodo del año 2015 hubo setecientos treinta y dos (732) graduandos en modalidad presencial, y para el primer periodo del año 2016 quinientos setenta y siete (577) graduandos.

Se recopila la información de la ejecución del Plan de Pruebas por medio de la herramienta contenida, utilizando Receptores los cuales muestra los detalles obtenidos durante ejecución del plan de pruebas de todas las respuestas de la solicitudes de los servidores, estos resultados obtenidos se pueden mostrar en forma de:

- Árbol.
- Archivos de Registros.
- Gráficos.
- Tablas.

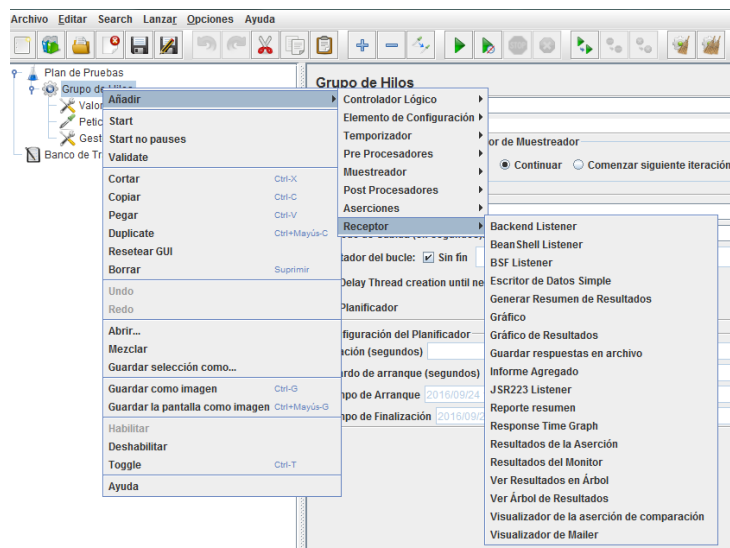


Figura 12 Ajuste de Receptores de la Prueba

Luego de terminar de configurar el Plan de Pruebas, se guarda y se procede a ejecutar dicho plan, los receptores proporcionan un recurso para medir y analizar el desempeño del

aplicativo Web detectando posibles errores y problemas en el software. A continuación se mostrara los resultados obtenidos de una prueba de carga simulando 500 usuarios en 10 segundos a una funcionalidad gestionar grados, gestionar datos de grados de estudiante.

Ver Resultados en Árbol

Nombre: Ver Resultados en Árbol

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo Navegar... Log/Mostrar sólo: Escribir en Log Sólo Errores Éxitos Configurar

Muestra #	Tiempo de comien...	Nombre del hilo	Etiqueta	Tiempo de Muestra (ms)	Estado	Bytes	Latency	Connect Time(ms)
1	22.07.17.834	Grupo de Hilos 1-3	Petición HTTP	582	✓	49434	255	0
2	22.07.17.790	Grupo de Hilos 1-1	Petición HTTP	635	✓	49434	319	0
3	22.07.17.790	Grupo de Hilos 1-2	Petición HTTP	645	✓	49434	339	0
4	22.07.17.936	Grupo de Hilos 1-4	Petición HTTP	536	✓	49434	213	0
5	22.07.18.139	Grupo de Hilos 1-6	Petición HTTP	526	✓	49434	210	0
6	22.07.18.040	Grupo de Hilos 1-5	Petición HTTP	646	✓	49434	224	0
7	22.07.18.240	Grupo de Hilos 1-7	Petición HTTP	522	✓	49434	233	0
8	22.07.18.340	Grupo de Hilos 1-8	Petición HTTP	431	✓	49434	180	0
9	22.07.18.441	Grupo de Hilos 1-9	Petición HTTP	452	✓	49434	195	0
10	22.07.18.542	Grupo de Hilos 1-	Petición HTTP	408	✓	49434	176	0
11	22.07.18.641	Grupo de Hilos 1-	Petición HTTP	444	✓	49434	191	0
12	22.07.18.741	Grupo de Hilos 1-	Petición HTTP	413	✓	49434	160	0
13	22.07.18.841	Grupo de Hilos 1-	Petición HTTP	415	✓	49434	181	0
14	22.07.18.941	Grupo de Hilos 1-	Petición HTTP	453	✓	49434	198	0
15	22.07.19.042	Grupo de Hilos 1-	Petición HTTP	413	✓	49434	163	0
16	22.07.19.143	Grupo de Hilos 1-	Petición HTTP	423	✓	49434	172	0
17	22.07.19.243	Grupo de Hilos 1-	Petición HTTP	457	✓	49434	193	0

Figura 13 Resultados de la ejecución de la Prueba

La figura 12 muestra usuario por usuario, el estado, es decir, si la petición fue atendida, su tamaño en bytes, la latencia; anexo a esto también realiza un análisis de la estructura o arquitectura del aplicativo Web esta opción se puede obtener añadiendo dentro del plan de pruebas el receptor “ver Árbol de Resultados”.

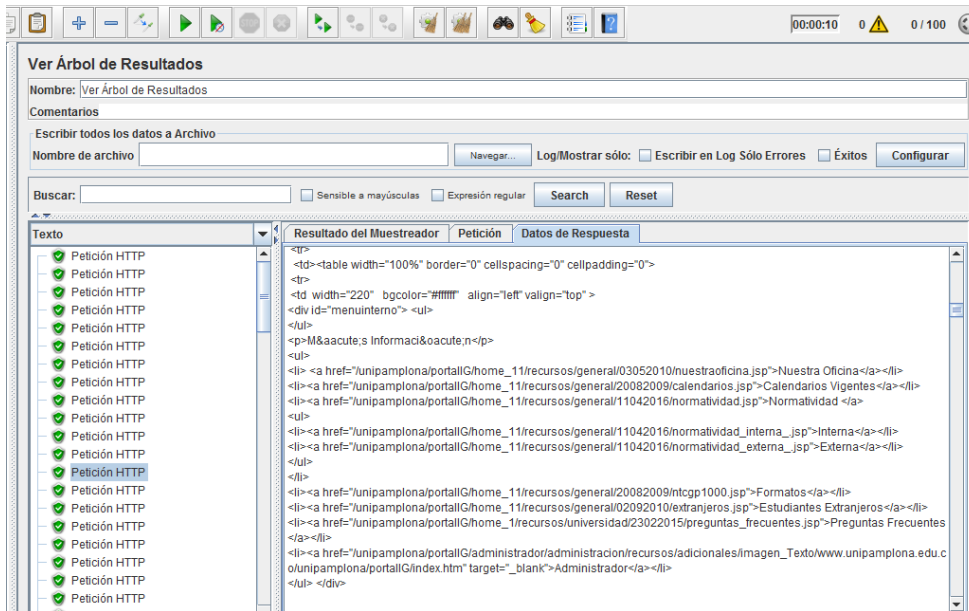


Figura 14 Resultados Petición de usuario

En la pestaña resultado de muestreador del anterior receptor se obtiene información del tipo de petición que ejecuto el usuario en este caso es Petición HTTP validando el estado y la dirección web donde se encuentra alojada la funcionalidad a probar.

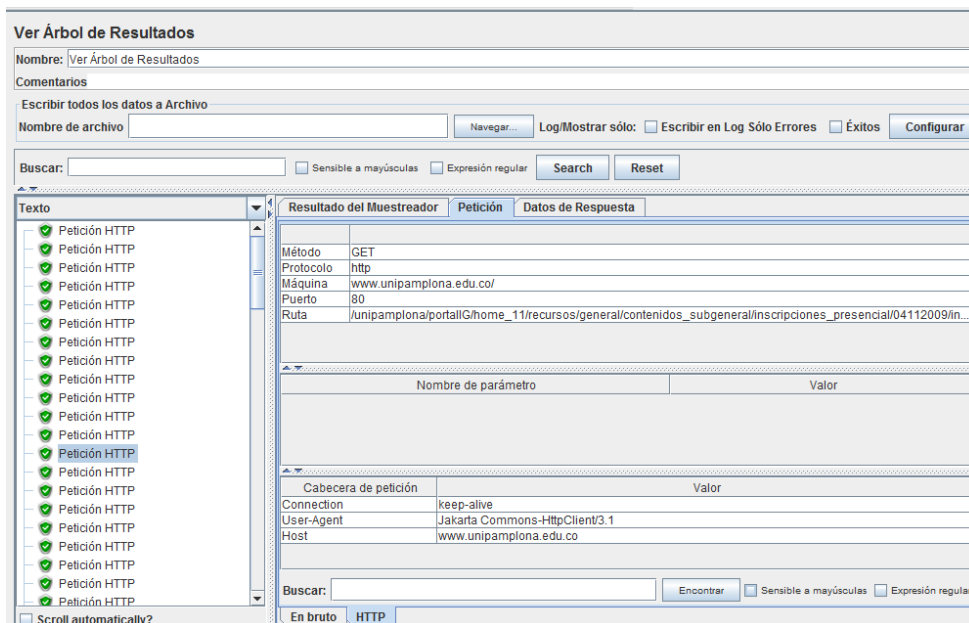


Figura 15 Arbol de Respuesta (Tipo de Petición)

Se puede concluir después de ejecutar una prueba de carga básica a un aplicativo web determinado, simulando 500 usuarios en 10 segundos es decir 50 usuarios por segundo

solicitando una petición HTTP en este caso en gestionar los datos personales de cada estudiante para grado, que su rendimiento fue óptimo con un 7% margen de error correspondiente a las solicitudes no atendidas, ya que se logra atender la mayor parte de las solicitudes de los usuarios satisfactoriamente sin llegar a un detrimento del aplicativo web o al colapso del servidor y el tiempo requerido para atender cada petición el cual corresponde al tiempo de latencia, desde que se envía el request hasta que se recibe el primer byte de respuesta de cada usuario, como también parte de la estructura en HTML de la página del aplicativo web. El estado de la solicitud que envía el código 200 Haciendo referencia a que fue atendida. En uno de los casos que no fue atendida una solicitud de un usuario la latencia sería cero porque no hubo respuesta del request, pero sin embargo se puso en espera y finalmente fue rechazada en promedio tarda 13000 (ms), obteniendo como mensaje de respuesta “NO HHTP response message: Connection reset” Esto se puede observar en las gráficas de árbol de resultados en la parte izquierda representado con un icono verde.

La versión del aplicativo web que se encuentra actualmente en uso corresponde a la versión 3.2, alcanzo un 33,4 % margen de error llegando hacer afectado el rendimiento del aplicativo, sin lograr atender todas las peticiones simuladas con un tiempo de 3100(ms) en promedio, en algunas peticiones se halló un tiempo de cero 0 (ms) esto significa un respuesta de rechazo expulsándolos de sesion. Alcanzando 150 peticiones de usuarios rechazadas con un rendimiento de 17,5 usuarios/seg.

Los datos obtenidos y el anterior análisis corresponden a una sesión de pruebas a la versión 4.0 y 3.2 del aplicativo, garantizando que la nueva versión da resultados óptimos de carga y rendimiento respecto a su versión anterior la 3.2.

4 Conclusiones

Las pruebas son necesarias para demostrar al desarrollador y al cliente que el software cumple con los requerimientos funcionales y no funcionales se cumplan, y que el código desarrollado funcione correctamente, y para descubrir: Defectos, fallas o errores y comportamientos incorrectos o indeseables. Por tal motivo es una tarea de vital importancia para dar confiabilidad y calidad al software y evitar fallas futuras por tal motivo se implementa un procedimiento para la aplicación de pruebas de carga con apoyo de la herramienta tecnológica para evaluar el aplicativo web.

Gracias a la utilización de la herramienta tecnológica se logró verificar que el rendimiento de la aplicación es óptimo ante situaciones de carga y estrés en componentes típicos de Internet (HTTP), se establece cuatro tipos de pruebas de rendimiento: rendimiento, estrés, sostenidas y picos. Mediante la ejecución de un Plan de Pruebas (Testplan) configurado para lograr cada objetivo. Como resultado de la obtención y procesamiento de la información, utilizando conceptos estadísticos que arroja la herramienta después de la ejecución de las pruebas se procede a una medición del rendimiento del aplicativo web para lograr los objetivos propuestos, así desarrollar un aplicativo web de calidad, y a su vez generar confiabilidad en el software.

5 Bibliografía

Universidad de Pamplona. (20 de MAYO de 2015). Obtenido de Recursos-Gestasoft:
http://www.unipamplona.edu.co/unipamplona/portallIG/home_38/recursos/01_general/28072014/gestasoft.jsp

Universidad de Pamplona. (20 de mayo de 2015). Obtenido de Recursos-Academusoft:
http://www.unipamplona.edu.co/unipamplona/hermesoft/portallIG/home_28/recursos/academusoft/14042008/academusoft_inicio.jsp

Universidad de Pamplona. (20 de mayo de 2015). Obtenido de Recursos-Academusoft:
http://www.unipamplona.edu.co/unipamplona/portallIG/home_38/recursos/01_general/28072014/academusoft.jsp

GIL, T. (1995). What We Fail To Do In Our Current en Testing Culture. En *Testing Techniques Newsletter*.

IEEE. (2004). Pruebas de Software. En S. E. Committee, *Guide to the Software Engineering Body of Knowledge*.

KRASNER, G. y. (1988). Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, 26-29.

MILLER, E. (1977). Program Testing Techniques. En *The Philosophy of Testing* (págs. 1-3).

MILLER, E. (2000). *WebSite Testing*. Obtenido de <http://www.soft.com/eValid/Technology/White.Papers/website.testing.html>

Myers, G. (1979). The Art of Software Testing. En G. Myers, *The Art of Software Testing*.

NGUYEN, H. (2001). *Testing Applications on the Web*.

OFFUTT, J. (Marzo,2002). Quality Attributes of Web Software Applications. En *IEEE Software* (págs. 25-32).

OLSINA. (1999). Specifying Quality Characteristics And Attributes for Web Sites.
Workshop on Web Engineering.

SPLAINE, S., & JASKIEL, S. (2001). *The Web Testing Handbook.*

(Universidad de Pamplona, 2015)

Universidad de Pamplona. Recursos-Gestasoft.[en línea] <
http://www.unipamplona.edu.co/unipamplona/portaIG/home_38/recursos/01_general/28072014/gestasoft.jsp>[citado el 20 de mayo de 2015]

Universidad de Pamplona. Recursos-Academusoft.[en línea] <
http://www.unipamplona.edu.co/unipamplona/hermesoft/portaIG/home_28/recursos/academusoft/14042008/academusoft_inicio.jsp>[citado el 20 de mayo de 2015]

Universidad de Pamplona. Recursos-Academusoft.[en línea] <
http://www.unipamplona.edu.co/unipamplona/portaIG/home_38/recursos/01_general/28072014/academusoft.jsp>[citado el 20 de mayo de 2015]

Universidad de Pamplona. Recursos-Gestasoft.[en línea] <
http://www.unipamplona.edu.co/unipamplona/portaIG/home_38/recursos/01_general/28072014/gestasoft.jsp>[citado el 20 de mayo de 2015]


Documento de infraestructura y soporte tecnológico para procesos de renovación y registros de programas académicos.CIADTI [docx]. p. 6

Universidad de Pamplona. Recursos-Academusoft. [En línea] <
http://www.unipamplona.edu.co/unipamplona/hermesoft/portaIIG/home_28/recursos/academusoft/14042008/academusoft_inicio.jsp>[citado el 20 de mayo de 2015]

Universidad de Pamplona. Sistema Integrado de Gestión [en línea]
<http://www.unipamplona.edu.co/unipamplona/portaIIG/home_13/recursos/01_general/10022014/mapa_procesos_2014.jsp> [citado el 4 de mayo del 2015].

http://www.unipamplona.edu.co/unipamplona/portaIIG/home_11/recursos/general/contenidos_subgeneral/inscripciones_presencial/21042014/ofertaacademica_2016.jsp

6 Anexos


Universidad de Pamplona
Pamplona - Norte de Santander - Colombia
Tel: (7) 5883403 - 5883404 - 5883405 Fax: 5882730 - www.unipamplona.edu.co

GA 190.60.30.FO20.1213-004

Pamplona, 24 de Junio de 2016

**CENTRO DE INVESTIGACION APLICADA Y DESARROLLO
DE TECNOLOGIAS DE LA INFORMACION**


CERTIFICACION DE PASANTIA

Mediante la presente Acta se Certifica que el estudiante **DANILO JHERSON SAED FLOREZ TUNAROZA**, identificado con la cedula de ciudadanía No. 1.094.268.196 expedida en Pamplona, estudiante del programa de **INGENIERIA DE SISTEMAS** de la Universidad de Pamplona. Cumplió satisfactoriamente con las prácticas programadas en el **Primer Periodo académico** del año 2016 en el horario acordado, durante los cuatro (4) meses desempeño y desarrolló las Actividades y Tareas programadas en su plan de Actividades durante su Pasantía en esta Empresa, efectuada y concluida en el periodo comprendido desde el día 24 de febrero hasta el 24 de junio.

Atentamente,

Raúl Hacip Contreras Jaime
INGENIERO
RAUL HACIP CONTRERAS JAIMES
Coordinador Técnico de Desarrollo (CIADTI)

Danielo Jheron Saed Florez Tunaroz


Una universidad innovante y comprometida con el desarrollo integral

