Universidad de las Ciencias Informáticas

Facultad 7

Título: Procedimiento para pruebas de eficiencia bajo carga intensiva en aplicaciones web de salud.

Trabajo de diploma para optar por el título de
Ingeniero en ciencias informáticas

Autor: Yilen Pons Ramírez.

Tutor: Ing. Juan Carlos Pujol García.

Tutor: Ing. Lourdes Escalona Peral.

Asesor: Ing. Yoenny Pérez Romero

Ciudad de La Habana, Junio 2007

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de la presente tesis y reconozco a la Facultad 7 de la Universidad de las Ciencias Informáticas y a Softel los derechos patrimoniales de la misma.
Este trabajo de diploma forma parte integrante de un tema más amplio dirigido por la Empresa Softel.

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

Autora:

_____
Yilen Pons Ramírez

Tutores:

_____ _____
Ing. Juan Carlos Pujol García Ing. Lourdes Escalona Peral

DATOS DE CONTACTO

Juan Carlos Pujol García: Ingeniero en Máquinas Computadoras, ISPJAE-1986, Especialista Superior en Informática de Softel. Profesor Auxiliar de la UCI (Universidad de las Ciencias Informáticas), Maestrante de telemática. Labora actualmente como jefe del proyecto de Servicios Remotos de la empresa Softel, su grupo que se dedica a la administración remota de servidores y aplicaciones informáticas, tanto basadas en plataforma libre como propietaria. Imparte la asignatura Sistemas de Bases de Datos en la UCI. Está interesado en: Técnicas que incrementen la seguridad (en el sentido amplio) de los servidores y aplicaciones informáticas, técnicas de monitoreo de funcionamiento y de eficiencia, técnicas de gestión de ancho de banda, VPN, teleinformática, administración (para alta eficiencia) de servidores de bases de datos.

Datos de contacto:

Empresa: UCI

Dirección: Carretera a San Antonio Km. 2 1/2 Reparto Torrens, Infraestructura productiva de la UCI, Ciudad Habana.

Teléfono: (53-7) 835-8258

e-mail: : juanca@softel.cu, juanca@uci.cu

Lourdes Escalona Peral: Profesor graduado de Ingeniero Informático en el año 2004 en la Universidad de Holguín. Ha impartido las asignaturas ISW 1 y 2 y Seminario de Tesis. Posee la categoría docente de Instructor y cursa la maestría de Ciencias de la Computación en la UCI. Jefe de Dpto. de la Especialidad de la Facultad 7.

Datos de contacto:

Empresa: UCI

Dirección: Carretera a San Antonio Km. 2 1/2 Reparto Torrens, Infraestructura productiva de la UCI, Ciudad Habana.

Teléfono: 835-8131

e-mail: lescalonap@uci.cu.

Ing. Yoenny Perez Romero: Ingeniero Informático graduado en el Instituto Superior Politécnico "José Antonio Echeverría", en el año 2005. Posee 2 años de experiencia laboral, se ha desempeñado como profesor en la Universidad de las Ciencias Informáticas, UCI, vinculado siempre a las asignaturas de la especialidad. Durante este mismo tiempo ha estado desarrollando conjuntamente con un equipo de trabajo de la UCI y de la empresa Softel una solución informática que automatice y gestione los procesos inherentes a la Atención Primaria de la Salud en Cuba, donde ha fungido como Líder de Proyecto.

Datos de contacto:

Empresa: UCI

Dirección: Carretera a San Antonio de los Baños, Km 21/2, Torrens, Boyeros,

Facultad 7, UCI.

Teléfono: (837) -8132

e-mail: yoenny@uci.cu

AGRADECIMIENTOS

Quiero agradecer a todas aquellas personas que de una forma u otra han contribuido al desarrollo de este trabajo de diploma. De manera especial agradecer a mi tutor, el ingeniero Juan Carlos Pujol por su empeño, consagración y disposición en todo momento.

A Yanssel Urquijo Morales quién me brindó su apoyo incondicional para la realización del presente trabajo de diploma.

A mi tutora, la ingeniera Lourdes Escalona Peral, que siempre me ayudó cuando lo necesité.

Así como, a un equipo de especialistas de la empresa SOFTEL: Lic. Regla Silva Calderón, Lic. Rosa Bernaza, Lic. Lucy Cruz, Dr. Dennis Deribet, por su apoyo para la investigación y críticas constructivas para una mejor realización de la tesis.

Además, agradecer a Reinier Alonso, Adrián Peña, Karelys Mesa, Darlen Hornia, Yaimi Márquez, y al resto de mis amistades, por brindarme su ayuda durante la investigación.

También quisiera, agradecer grandemente a mi familia, en especial a mis padres y hermano por apoyarme en todo momento y ayudarme a salir adelante.

De manera muy especial, a nuestro Comandante en Jefe Fidel Castro Ruz y a la Revolución Cubana por darme la oportunidad de estudiar en una escuela con cualidades excepcionales.

DEDICATORIA

Al ingeniero Juan Carlos Pujol: Por su paciencia, dinamismo y ayuda ilimitada durante la confección del presente trabajo de diploma.

A mis padres: Por su amor y su apoyo en todo momento y bajo cualquier circunstancia.

A mi hermano: Por ser mi mano derecha, apoyándome siempre con cariño y respeto.

A mis amigas: Por estar siempre a mi lado, en los buenos y los malos momentos.

Por su cariño y afecto.

A nuestro comandante en jefe Fidel Castro Ruz: Por su empeño en la cultura y superación del hombre. Por crear una escuela en la que he pasado los mejores años de mi vida.

Por "ser" y "estar" para todos los cubanos.

RESUMEN

Para garantizar la calidad en la industria de software se realizan diferentes pruebas, entre ellas están las de eficiencia bajo carga intensiva. Estas necesitan el uso de procedimientos que sirvan de guía a los proyectos productivos para llevarlas a cabo de forma ordenada y concisa.

La investigación tiene como objetivo, desarrollar dos procedimientos para aplicar pruebas de eficiencia bajo carga intensiva en aplicaciones web (uno para la capa de negocio y otro la de acceso a datos) lo que permitirá comprobar los proyectos de software desarrollados por la facultad 7.

En el trabajo se aborda la importancia de las pruebas de calidad en el software haciendo mayor énfasis en las de carga, stress y eficiencia, se comparan las herramientas que pueden ser usadas para estos tipos de pruebas, escogiéndose para la realización de las pruebas de eficiencia bajo carga intensiva, el Apache JMeter, una herramienta que se destaca por su versatilidad, estabilidad, y ser de uso gratuito.

Los procedimientos diseñados para las pruebas de eficiencia bajo carga intensiva en aplicaciones web de salud, se pusieron en práctica en dos proyectos productivos de la facultad 7, obteniéndose resultados satisfactorios, creando lazos de retroalimentación con sus diseñadores, entregándoles un documento donde se recogen los resultados y una lista de los posibles problemas. En el futuro, los procedimientos propuestos se podrán aplicar a los restantes proyectos de la universidad.

PALABRAS CLAVES
Procedimientos, eficiencia, stress, carga, JMeter, test, aplicaciones, pruebas.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La calidad del software es medible y varía de un sistema a otro o de un programa a otro. Un software hecho para ejecutarse una sola vez no requiere gran nivel de calidad; mientras que un producto de software para ser explotado durante un largo período, necesita ser confiable, mantenible y flexible.
(León, D.G 2007)

Es imprescindible tener en cuenta tanto la obtención de la calidad como su aseguramiento durante todas las etapas del ciclo de vida del software, incluyendo sus estados de evolución (especificaciones, diseño, códigos, etc.), y siempre se deben considerar todos los aspectos relacionados con su control (desarrollo de entornos y procedimientos de pruebas, testeos, etc.). Las pruebas representan un elemento crítico para la garantía de calidad y requieren gran cantidad de tiempo y esfuerzo.

Es frecuente encontrarse con el error de afirmar que el objetivo de las pruebas es convencerse de que el programa funciona bien, cuando ese es el objetivo de las fases anteriores; el de las pruebas, es destapar errores.

La fase de pruebas es de vital importancia en todas las etapas del software, para asegurar la calidad del mismo. Sin embargo, la mayoría de las empresas a nivel mundial, a pesar de saber la necesidad de probar sus sistemas, no le están dando la importancia requerida, lo que trae como consecuencia que muchos de ellos presenten fallos. Es por esto que se ha puesto énfasis en el tema de la calidad para una mejor confiabilidad en el software una vez terminados, para su posterior puesta en práctica.

Estos estudios han dado pie a que en nuestro país se comience a profundizar en la realización de prueba al software para que su funcionamiento sea el esperado por el cliente durante su explotación. La Universidad de Ciencias Informáticas es, en estos momentos, una de las proveedoras de software en el país, por lo que también está tomando todas las medidas necesarias para una mejor calidad en sus proyectos productivos. Muchos de los problemas que se presentan, son en cuanto al tiempo de respuesta, cuando muchos usuarios acceden concurrentemente a una información determinada. Para evitar que esto ocurra, es necesario tener conocimiento de unas pruebas específicas, las de eficiencia bajo carga

intensiva. Estas son de importante aplicación en todos los proyectos de software, pero este trabajo está enmarcado solamente en los sistemas web de salud vinculados a la Facultad 7 y Softel de la Universidad de Ciencias Informáticas.

Las pruebas de eficiencia bajo carga intensiva no son más que la unión de pruebas de carga, stress y eficiencia trabajando en conjunto, pero para hablar de ellas, hay que conocer primeramente de las fases de pruebas y a cual pertenecen.

Las fases de prueba son:

Unidades: Las pruebas de unidades es una fase informal antes de entrar en la fase de pruebas propiamente dicha. La fase informal la lleva a cabo el propio codificador en su despacho, y consiste en ir ejecutando el código para convencerse de que "básicamente, funciona". Esta fase suele consistir en pequeños ejemplos que se intentan ejecutar. Si el módulo falla, se suele utilizar un depurador para observar la evolución dinámica del sistema, localizar el fallo, y repararlo. (Mañas, J.A 2007).

Dentro de las pruebas de unidades están las pruebas de caja negra y de caja blanca. Se dice que una prueba es de caja negra cuando prescinde de los detalles del código y se limita a lo que se ve desde el exterior. Intenta descubrir casos y circunstancias en los que el módulo no hace lo que se espera de él.

Las pruebas de caja blanca son el caso contrario de las de caja negra, o sea, lo que se mira con lupa es el código que está ahí escrito y se intenta que falle.

Integración: Las pruebas de integración se centran en probar la coherencia semántica entre los diferentes módulos, tanto de semántica estática (se importan los módulos adecuados; se llama correctamente a los procedimientos proporcionados por cada módulo), como de semántica dinámica (un módulo recibe de otro lo que esperaba). Normalmente estas pruebas se van realizando por etapas, englobando progresivamente más y más módulos en cada prueba. (Mañas, J.A 2007).

Sistema y aceptación: Las pruebas de sistema y de aceptación son las que se plantea el cliente final, que decide qué pruebas va a aplicarle al producto antes de darlo por bueno y pagarlo. El objetivo del que prueba, es encontrar los fallos antes de poner el programa en producción. (Mañas, J.A. 2007).

Las pruebas de aceptación involucran pruebas funcionales y de sistema. En el caso de las pruebas funcionales se realizan pruebas con el método de caja negra utilizando técnicas de participación equivalentes, análisis del valor límite, de comparación, etc.

Las pruebas de aceptación incluyen además pruebas de desempeño, de stress, de seguridad, pruebas para los datos persistentes haciendo en este caso pruebas de frecuencia de uso, restricciones de acceso, restricciones de integridad, requisitos de guarda y retención de datos, etc.

Las pruebas de desempeño permiten analizar y evaluar las características del software relacionadas con el desempeño.

Tipos de pruebas de desempeño

1. **Pruebas de benchmark**.

Los benchmarks son pruebas para medir el rendimiento y poder verificar que el hardware funciona de forma óptima o para comparar distintas configuraciones. (Curiel, M 2005).

2. **Pruebas de stress**.

Las pruebas masivas, conocidas como Stress Test, constituyen un mecanismo de control de calidad mediante el cual se somete el software al funcionamiento constante al límite por un período de tiempo. (Curiel, M 2005).

La idea es que si bajo esta carga de trabajo no presenta fallas, es muy probable que no lo haga por un largo período de tiempo o nunca, pero si las presenta durante la prueba masiva, se evitará entregar un

sistema y que al poco tiempo el usuario descubra defectos en el mismo. Se debe tener en cuenta que estas pruebas no garantizan que los componentes no presenten fallos, sino que reducen la posibilidad que aparezcan al poco tiempo.

Las condiciones de stress no se espera que sucedan en la realidad. Estas pruebas permiten documentar las condiciones bajo las cuales el sistema falla, o sea sus límites. Permiten verificar la aceptabilidad del desempeño del sistema ante condiciones anormales o extremas: como por ejemplo el volumen de usuarios/transacciones extremadamente alto.

3. **Pruebas de carga**:

Estas pruebas son realizadas para garantizar que el funcionamiento de un software será el adecuado cuando tengan acceso a la misma el máximo de usuarios autorizados en el entorno de producción. (Casas, I 2006).
La prueba de carga se debe realizar siempre antes de la implementación de la aplicación en un entorno de producción y permiten verificar y validar el desempeño de un elemento de un sistema bajo diferentes condiciones de carga como son el número de usuarios y de transacciones. Son importantes cuando los sistemas deberán soportar un gran volumen de usuarios o transacciones concurrentes.

4. **Pruebas de eficiencia**:

Estas pruebas están enfocadas a monitorear el comportamiento de una aplicación en ejecución con el fin de conocer dónde invierte su tiempo. (Weyuker, E. J, Vokolos F. I.).

- Acceso a datos
- Llamadas a un procedimiento
- Llamadas al sistema
- Permiten identificar cuellos de botella y procesos ineficientes.

Haciendo las pruebas de eficiencia, se puede ver rápidamente donde se está perdiendo tiempo, tanto en los queries de la base de datos, como en el código de la aplicación. Estas son muy sensibles a las

condiciones sobre las cuales se aplican. Por tanto se debe ser cuidadoso al elegir la carga de trabajo y realizar el análisis de los resultados.

Las pruebas de eficiencia permiten obtener el tiempo de respuesta, u otros parámetros de gasto. Típicamente puede ser motivo de preocupación saber cuánto tiempo le lleva al sistema procesar muchos datos. Para esto, suele ser importante conocer cómo evoluciona al variar la dimensión del problema (por ejemplo, al duplicarse el volumen de datos de entrada).

Dichas pruebas ayudan además a identificar una serie de problemas en las aplicaciones web, ya sea en capa de negocio o en la capa de acceso a datos, la mayoría de estos problemas tiene que ver con la eficiencia de los cuellos de botella que se levantan cuándo múltiples usuarios necesitan acceder un recurso común. Por ejemplo, si la aplicación depende de una única base de datos común, entonces la velocidad de acceso a los datos podría volverse un problema si los queries de la base de datos no son eficaces.

Las pruebas de eficiencia bajo carga intensiva, que son las de interés en este trabajo de diploma, pueden clasificarse como pruebas de sistemas y aceptación, puesto que, las pruebas de aceptación involucran pruebas de desempeño, y estas a su vez, incluyen las pruebas de carga, stress y eficiencia. El objetivo general de las pruebas de aceptación es validar el producto, o sea verificar si este atiende a los requisitos especificados; y uno de ellos seria que las aplicaciones web, respondan a las peticiones que hacen los usuarios, en el menor tiempo posible, que no es más que el tiempo indicado por el grupo de diseñadores de dicho sistema.

Realizar pruebas de eficiencia bajo carga y stress en un software es necesario para saber que cuando esté en producción, será capaz de atender todas las peticiones que le lleguen, o al menos para conocer el umbral a partir del cual dejará de hacerlo.

El tema de las pruebas de eficiencia bajo carga intensiva, surge por la necesidad de su aplicación a los diferentes proyectos de la facultad 7 de la Universidad de Ciencias Informáticas, siendo su objetivo fundamental el mejoramiento de los tiempos de respuesta tanto de aplicaciones web como de bases de

datos de los sistemas de salud. Para la obtención de unas buenas pruebas de eficiencia es necesario la utilización de procedimientos que especifiquen los pasos a seguir para su aplicación, permitiendo uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

La **situación problémica** está enmarcada en los proyectos productivos de la facultad 7 de la Universidad de las Ciencias Informáticas, los cuales están formados por estudiantes y profesores de la facultad. Estos se desempeñan como analistas, desarrolladores, diseñadores, probadores, etc., según el rol que les corresponda jugar en su equipo de trabajo definido en cada etapa de desarrollo del software. El equipo de probadores se encarga de realizar las pruebas al software mientras se está desarrollando y antes de ser entregado al proyecto de calidad de la facultad, el cual comprobará que está listo para ser explotado.

Dentro de las pruebas que deben realizar el equipo de probadores de cada proyecto productivo están las de eficiencia bajo stress, para verificar que los tiempos de respuesta de la aplicación son los correctos.

En la facultad no existe conocimiento de las pruebas de eficiencia bajo carga intensiva, por lo que no se están aplicando a los proyectos productivos, lo que dificulta que tengan un buen rendimiento fundamentalmente cuando existe un número elevado de usuarios conectados a sus aplicaciones.

El **problema científico** que se plantea para la solución de la situación problémica especificada, sería la necesidad de proponer y aplicar dos procedimientos para pruebas de eficiencia bajo carga intensiva en aplicaciones web, uno para la capa de acceso a datos y otro para la capa de negocio, en el tema de informática médica, aplicados a los proyectos de la facultad.

Para dar solución al problema planteado se define como **objeto de estudio**, las pruebas realizadas en aplicaciones web por la facultad 7 y Softel para el tema de la informática médica. Y para enmarcar la investigación en un ambiente más definido se tiene como **campo de acción**, las pruebas de eficiencia bajo carga intensiva en aplicaciones web, realizadas por la facultad 7 y Softel, para la salud.

Para el desarrollo de la investigación se tiene como **objetivo general**, desarrollar dos procedimientos para aplicar pruebas de eficiencia bajo carga intensiva en aplicaciones web, uno para capa de negocio y otro para capa de acceso a datos, desarrolladas por la facultad 7 para el tema de informática médica.

Objetivos específicos:

Las tareas u objetivos específicos que hay que seguir para desarrollar procedimientos para las pruebas de eficiencia bajo carga intensiva, con la calidad requerida, son las siguientes:

1. Analizar pruebas de eficiencia, stress y carga aplicadas a nivel mundial.

2. Comparar conjunto de herramientas de gestión de dichas pruebas, para sugerir y decidir cuáles se adaptan mejor a las necesidades de los proyectos productivos de la facultad.

3. Diseñar dos procedimientos para la gestión de pruebas de eficiencia bajo carga intensiva en aplicaciones web, uno para la capa de acceso a datos y el otro para la capa de negocio.

4. Validar los procedimientos diseñados para realizar las pruebas, sobre al menos un proyecto de la facultad 7, tomando como base la herramienta elegida y establecer lazo de retroalimentación para los desarrolladores de dichos proyectos de software.

El tema de investigación realizado sobre las pruebas de eficiencia bajo carga intensiva en aplicaciones web, comprende tres capítulos. En el capítulo 1 se hace una fundamentación teórica, destacando los antecedentes, tecnologías y tendencias, enfatizando en los motivos del surgimiento del tema de tesis y haciendo una descripción comparativa entre diferentes herramientas que pueden ser usadas. En el capítulo 2, se diseñan dos procedimientos, uno para capa de acceso a datos y otro para capa de negocio de aplicaciones web, con el objetivo de proponer una guía para los diferentes proyectos productivos de la facultad 7 de cómo llevar a cabo las pruebas de eficiencia bajo carga intensiva en sus proyectos de

software. En el capítulo 3 se le aplican varias pruebas de este tipo a dos proyectos que se tomaron como ejemplos en la facultad 7, para poner en práctica los procedimientos diseñados en el capítulo anterior.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Marco conceptual

Ingeniería de Software: Es una tecnología que indica "cómo" construir técnicamente un software: económico, fiable y que funcione eficientemente.

Proceso: es lo que define: "quién", "qué", "cuándo", y "cómo" hay que realizar las cosas para alcanzar un determinado producto de software.

Calidad de software: La calidad del software es el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia.

Aseguramiento de la calidad: Acciones planificadas y sistemáticas necesarias para proporcionar la confianza adecuada de que un producto o servicio satisfará los requerimientos dados sobre calidad.

Tiempo de Respuesta: Es el intervalo de tiempo que transcurre entre la solicitud de un usuario al sistema y la respuesta de este último.

Capacidad: Máxima cantidad de trabajo útil que se puede realizar por unidad de tiempo.

Fase de pruebas de software: Esta es la fase encargada de añadir valor al producto que se maneja: todos los programas tienen errores y la fase de pruebas los descubre; ese es el valor que añade. El objetivo específico de la fase de pruebas es encontrar cuantos más errores, mejor.

Cuello de botella (Bottleneck): Límite en la capacidad de transferencia de información de un sistema que puede reducir el tráfico en condiciones de sobrecarga. Suele producir una baja del rendimiento y la velocidad general tanto en un sistema como en una conexión. Los cuellos botella (llamados restricciones) condicionan la salida de toda la producción. (Gevert, R 1990).

Pruebas de aguante o stress: Son aquellas que me permiten saber hasta dónde aguantan los sistemas, bien por razones internas (cantidad de datos que puede procesar), o bien por razones externas.

Pruebas de eficiencia o prestaciones: Medir la eficiencia es hacer una métrica específica como tiempo de la contestación y throughput para un número finito de usuarios para un software dado y ambiente del hardware.

Pruebas tensión o carga: Medir la tensión es obtener el número máximo de usuarios que aguanta un servidor de una aplicación antes de que falle.

1.2. Antecedentes, tecnologías y tendencias

1.2.1. Evolución histórica del objeto de investigación.

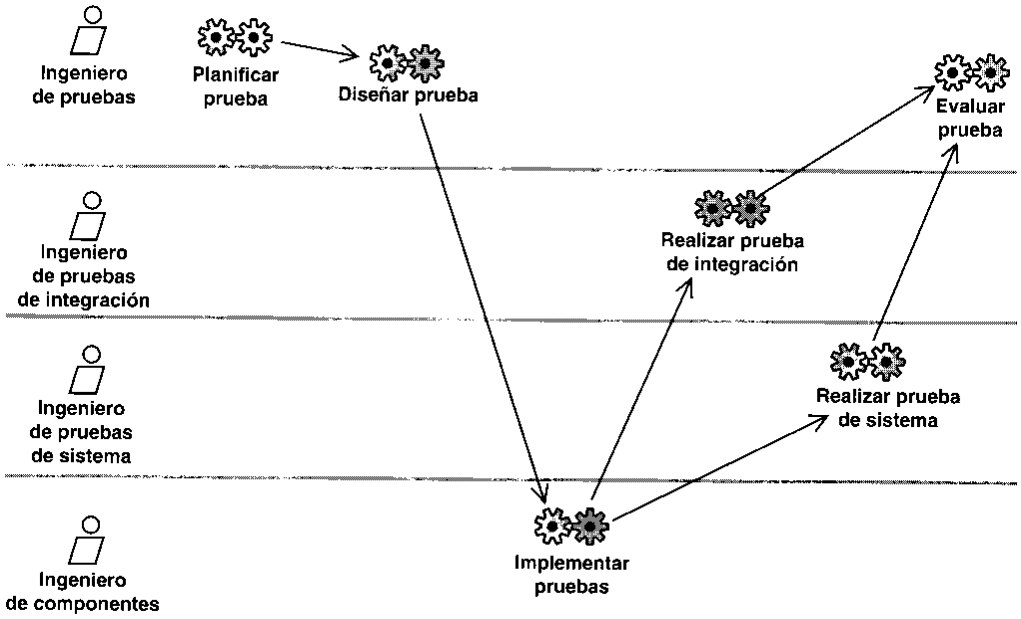
La necesidad de buscar una vía para el desarrollo y mantenimiento de productos de software, hizo que se patentara en la década de 1960, pero no fue hasta el año 1968 que se utilizó por primera vez el término "ingeniería del software", y con él, vías para el aseguramiento de su calidad. Uno de los problemas que se afrontan actualmente en la esfera de la computación es precisamente este de la calidad. Desde la década del 70, ha sido motivo de preocupación para especialistas, ingenieros, investigadores y comercializadores de software. Hoy, el principal desafío es mejorarla y reducir su costo.
Desde el momento en que se utilizó el término calidad, surgió la preocupación de su aseguramiento para lo que fue necesario desarrollar pruebas, las cuales desde esos momentos, fueron la actividad más común de control realizada en los proyectos de desarrollo o mantenimiento de aplicaciones y sistemas.

Las pruebas representan una actividad fundamental en el desarrollo de software y en muchos casos, supone prácticamente el único medio empleado en los proyectos para la verificación y validación del software (Sanz, L.F 2005). Estas se encuentran dentro del flujo de trabajo del RUP, que no es más que una metodología dentro de la ingeniería de software (tecnología multicapa). Esta metodología es la que utiliza la universidad y nuestro país, independientemente que existan otras. Fue creada por Jacobson,

Rumbaugh y Booch, preparado para desarrollar grandes y complejos proyectos y orientado a objetos. Además utiliza el UML como lenguaje de representación visual.

La versión que se estandarizó en el Proceso Unificado de Desarrollo (RUP), surgió en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0, de ahí las siglas con las que se identifica. El propósito de las pruebas dentro del RUP es busca los defectos a los largo del ciclo de vida del software.

Este es el flujo de trabajo durante las pruebas, incluyendo los trabajadores participantes y sus actividades.



Las pruebas del software son el proceso de ejecución de un programa con la intención de descubrir errores, una forma eficiente de hacerlo es diseñar pruebas que sistemáticamente saquen a la luz diferente clases de errores haciéndolo con la menor cantidad de tiempo y esfuerzos. (Pressman, R 2002).

A la hora de hablar de pruebas de software hay que tener en cuenta la relación en las aplicaciones web con las bases de datos.

La conexión de las bases de datos con la web ha ido progresando de una interrelación realizada hasta la situación actual, en la que prácticamente todo SGBD proporciona un módulo o toda una serie de

herramientas para publicar la información de la base de datos en la red, siendo accesible desde cualquier punto, utilizando un navegador. (Feiler, J 1999).

Lo que relaciona a las aplicaciones web con la base de datos, es la capa de acceso a datos, por lo que para llevar a cabo las pruebas de eficiencia bajo carga intensiva a las aplicaciones web, hay que tener en cuenta la relación de las diferentes capas: la capa de presentación que seria sustituida por la herramienta encargada de hacer dichas pruebas, la capa de negocio y la capa de acceso a datos.

1.2.2. Tecnología y tendencias.

"La tecnología por sí misma no proporciona calidad, son unos procedimientos efectivos de pruebas y personal debidamente preparado los que hacen la diferencia. A menos que los responsables de desarrollo, de calidad y de operaciones estén de acuerdo con los procesos de pruebas, los limites organizativos quedarán difuminados y las responsabilidades no estarán claras." (Visitación, M 2007).

Con el acelerado avance tecnológico de la información, la cantidad y la complejidad de los productos de software se están incrementando considerablemente, así como también la exigencia en su funcionalidad y confiabilidad; es por esto que la calidad y la productividad se están constituyendo en las grandes preocupaciones tanto de gestores como para desarrolladores de software.

Por este motivo surge la preocupación de hacer cada vez pruebas más específicas y exactas en los diferentes ciclos de vida del software para verificar que funciona correctamente y corregir los errores cometidos durante su implementación. Una de estas pruebas son las pruebas de eficiencia con las cuales se mide el tiempo de respuesta que demora la aplicación en devolver una información, ya sea por la capa de acceso a datos o por la capa de negocio. Estas pruebas al igual que el resto, son importantes para el aseguramiento de la calidad de software, para su posterior explotación.

La eficiencia de la capa de acceso a datos y la capa de negocio en las aplicaciones web, se pueden medir por separado. Para hacer pruebas bajo es fundamental conocer las capacidades del sistema o

aplicación antes de llevarla a producción, esto ahorrará posibles dolores de cabeza a la hora de poner en producción el sistema y evaluar las capacidades presentes y futuras del mismo.

Para tener una estimación de lo que puede llegar a aguantar y el tiempo que se demora en responder una aplicación, los administradores o creadores del sistema, deben llevar a cabo unas buenas pruebas de eficiencia bajo carga antes de ponerlo en producción.

Las pruebas de eficiencia son recomendables que se lleven a cabo bajo condiciones de carga y stress con el objetivo de asegurar que el software puede responder en tiempo bajo condiciones máximas de aguante. Sin embargo la mayoría de las pruebas de software, requiere para su desenvolvimiento, del uso de herramientas que permitan su automatización, incluso estas.

Con las herramientas de prueba se generan sesiones de usuario que ejecutan las típicas órdenes contra la base de datos que con carácter general realiza un determinado perfil de usuario, recogida de distintos tipos de datos, búsquedas con parámetros, etc. También se establecen diferentes pesos sobre las órdenes que posteriormente serán lanzadas contra el servidor simulando una carga de trabajo realista sobre el mismo.

Para llevar a cabo la ejecución de las pruebas a las aplicaciones se debe primero configurar la carga de usuarios y seleccionar las herramientas de stress adecuadas. Para la configuración de la carga de usuarios hay que establecer el número máximo de usuarios que se espera accedan a la aplicación y luego, aumentarlo. Es probable que una aplicación eficaz sea utilizada por mayor número de usuarios de lo esperado. Además, hay que calcular el tiempo de demora de la aplicación una vez simulado el número de usuarios especificado, determinando la carga que existirá en el servidor durante ese período, que debe ser en el que se pruebe la aplicación. Esta estrategia permite probar el impacto de la carga de usuarios, devolviendo el tiempo de demora para asegurar que la aplicación responde con eficacia durante los momentos de máxima congestión.

A la hora de hacer la selección de las herramientas de stress adecuadas hay que tener en cuenta que debe ser una que pueda simular un elevado número de concurrencia con los subprocesos suficientes para maximizar dichas concurrencias.

1.2.3. En el mundo.

En el momento socio-económico en el que nos encontramos, son cada vez más demandados los servicios informáticos, tanto consultoría, formación, soluciones, etc. Sin embargo en los últimos años se ha vivido el crecimiento y la explosión de una burbuja que ha dado pie a la implantación de proyectos en muchos casos inviables, mal dimensionados, mal desarrollados, encubriendo mediocridad y falta de profesionalidad en todas las áreas implicadas.

Por esta causa es de suma importancia que el software que se liberen para ser explotados tenga la calidad requerida, y para esto es indispensable tener conocimiento de herramientas que permitan hacer pruebas. En este tema de investigación, se profundiza en herramientas para pruebas de stress y carga que permitan medir la eficiencia.

Es muy común en las grandes y medianas empresas que existan equipos de trabajo dedicados a probar la funcionalidad de sus nuevos desarrollos, mientras que dedique poco o ningún tiempo a comprobar si estos cumplen con la eficiencia implantada por el equipo de diseñadores.

A raíz de esos problemas existentes en el mundo, hay empresas que se han preocupado por la demanda del conocimiento sobre pruebas y herramientas para la obtención de calidad. Entre los diferentes servicios que brindan al usuario se encuentran: la documentación sobre los tipos de prueba, su importancia y las herramientas adecuadas para ellas, con el objetivo de enriquecer al usuario con experiencia y visión.

Una de ellas es Autentía. Entre los servicios que brinda, se encuentran las herramientas de prueba (carga, stress, eficiencia), y su utilidad. Esta empresa, entre otras cuestiones ayuda a suplir las necesidades de los clientes a la hora de probar el software para verificar su buen funcionamiento.

Otra empresa que se dedica a brindar a través de la red, servicios que se necesitan para entregar aplicaciones on-line fiables y sin problemas, es Compuware. Según un estudio de Compuware en el año 2005, el 85% de los responsables de tecnologías de la información indican que la calidad de las aplicaciones es crítica o muy crítica para demostrar su valor en la empresa. Ante este problema, el 63% de las empresas ya han comenzado a trabajar en el factor calidad dentro de sus aplicaciones, y dentro de este porcentaje, el 50% han invertido en herramientas de pruebas de calidad para el desarrollo de aplicaciones.

Aunque las empresas entienden la importancia de la calidad y a pesar de las inversiones para conseguirla, no se están consiguiendo los objetivos deseados. Según datos de estudio, solo el 54% de las empresas en el mundo han invertido en soluciones de pruebas de calidad de las aplicaciones.

La relación entre obtención de resultados y uso de metodologías o procedimientos es clara, ya que según el estudio, el 45% de las empresas que aplican un procedimiento para hacer las pruebas, declaran obtener unos resultados excelentes, y un 52% obtienen un alto grado de efectividad en cuanto a eliminar defectos de las aplicaciones antes de su puesta en producción.

En la mayoría de las empresas, las aplicaciones se prueban en producción, por lo que es el usuario final el que define si funcionan o no las aplicaciones. Esto lleva a que una de cada cinco aplicaciones que se ponen en marcha tenga fallos.

Una de las consecuencias de esta situación es que se ralentiza la disponibilidad de las aplicaciones, lo cual afecta directamente un negocio. Por ello, existe necesidad de que los responsables de producción hagan las pruebas de eficiencia pertinentes antes de poner sus sistemas en un entorno real para ver que funcionan correctamente.

Existen serie de herramientas muy útiles para permitir a las empresas asegurar alta fiabilidad y rendimiento de manera sencilla, con productos y servicios que ayudan a los desarrolladores a emplear las mejores prácticas en el proceso de producir aplicaciones de alta calidad. Con estas herramientas se

proporciona un análisis detallado de la calidad y rendimiento de las aplicaciones propiciando a los desarrolladores un experto automatizado capaz de advertirles, localizar y corregir sus problemas.

Las herramientas para el entorno de prueba dan soporte a las pruebas de integración, pruebas de sistema, diagnóstico o afinado de las aplicaciones en los entornos de preproducción o certificación y los de producción, que generalmente son difíciles de realizar de una forma integrada.

Algunas de las herramientas que sirven para pruebas de eficiencia bajo carga intensiva:

- **Application Center Test**

Aplication Center Test (ACT) es una herramienta de Microsoft incluida en Visual Studio .NET Enterprise Developer y Enterprise Architect, diseñada especialmente para realizar pruebas de carga y stress, que permite obtener información para detectar problemas de rendimiento y escalabilidad. A pesar de estar especializada en este tipo de pruebas, ACT también permite realizar pruebas funcionales gracias a los test dinámicos, lo que puede resultar útil para aplicaciones de complejidad media-baja.

El funcionamiento de ACT reside en simular un gran número de usuarios abriendo múltiples conexiones al servidor y enviando peticiones HTTP. Algunas de las características más interesantes de ACT son:
- Permite probar cualquier página Web que responda a una petición HTTP (.asp, .aspx, .php.).
- Graba scripts de pruebas desde una sesión de Internet Explorer.
- Soporta SSL.
- Acumula los resultados de los test para su análisis.
- Gestiona cookies.
- Soporta diferentes tipos de esquemas de autentificación.

Se puede considerar que el predecesor directo de ACT es Web Application Stress Tool (WAST).

- **Herramienta Web Application Stress**

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

Cuando se ejecute la prueba de carga, hay que tener cuidado de no aumentar el nivel de carga en los equipos clientes hasta el punto de que los equipos de prueba pasen más tiempo realizando cambios de contexto entre subprocesos que trabajando. Para asegurar que los subprocesos se distribuyen entre los equipos clientes, utilice varios equipos clientes cuando ejecute una prueba de una aplicación basada en Web, de ese modo disminuirán los límites de recursos de un equipo cliente.

- **Siege**

Útil herramienta para ver de manera rápida y sencilla la carga sobre unas ciertas URLs. Permite enviar peticiones tipo POST, y HTTPS. Siege fue diseñado para permitir a diseñadores web medir la eficiencia de su código bajo la coacción, para ver cómo se levantará al cargar la aplicación web. Permite que el usuario agregue a un servidor web, un número configurable de usuarios simulados concurrentes.

Siege fue escrito en GNU/Linux y se ha puesto en práctica satisfactoriamente en AIX, BSD, HP-UX y Solaris. Debe compilar en las variantes de UNIX y en la mayoría de los más nuevos sistemas de BSD. Siege no correrá en Windows. Pero se puede usar el sitio para probar un servidor http de Windows.

Las herramientas tipo siege se utilizan cuando se quiere comprobar sólo si el servidor va a aguantar tal cantidad de carga en un determinado intervalo de tiempo. Sólo sirven para páginas sencillas, sin autenticación ni cookies.

- **QALoad de Compuware**

Es una herramienta que ayuda a los equipos de probadores, desarrolladores y jefes de proyecto a realizar pruebas de carga efectivas a aplicaciones distribuidas. QALoad es para aquellas aplicaciones que estén sometidas a gran carga de usuarios. Ayuda a alcanzar cargas que simulan el comportamiento real del negocio así como a validar que el sistema cumple aceptablemente con los niveles de servicio.

Algunas características importantes de esta herramienta son:

- ✓ Son pruebas escalables: Se puede emular la carga generada por cientos o miles de usuarios en la aplicación sin requerir la participación de los usuarios finales en sus equipos. Además se puede configurar un escenario de pruebas de carga para controlar las condiciones de las pruebas, crear los usuarios virtuales que se necesitan para simular la carga, iniciar y monitorizar las pruebas y se obtienen los informes de resultados.

- ✓ Fácil desarrollo de los scripts de prueba: Ofrece módulos configurables que facilitan el desarrollo de los scripts de prueba. Estos, llamados EasyScripts, permiten grabar el tráfico que genera la aplicación y convertirlo en un script de prueba. En el resultado se refleja el tráfico real generado por la aplicación y se mide el tiempo empleado para completar las transacciones garantizando que el sistema sometido a las pruebas alcanza las especificaciones para las que ha sido construido.

- ✓ Vista integrada de los recursos del sistema: La integración entre ServerVantage y QALoad permite monitorizar los recursos del sistema mientras se genera una carga realista de la aplicación. Los tiempos de respuesta de la aplicación y la utilización de recursos del sistema durante una prueba de carga, pueden ser analizados conjuntamente lo que permite una identificación rápida de los cuellos de botella.

- **Open Load Tester V5.0 de Open Demand Systems**

Open Load Tester es la primera solución rápida de optimización de rendimiento basada en navegador. Fácil de usar, para las pruebas de carga y stress de aplicaciones.

Impulsado por IBM WebSphere y DB2 base de datos universal, y completamente integrado con el desarrollo de aplicaciones de IBM WebSphere Estudio. El verificador de Carga Open Load Tester V5.0 brinda facilidad de uso, exactitud y escalabilidad de un solo desarrollo integrado y ambiente de la prueba.

Este Minimiza substancialmente el tiempo requerido para las pruebas de stress y pone a punto la eficiencia de la web basado en aplicaciones y servicios por simplificación de procesos, verificando la conducta funcional esperada a través de la regresión automatizada, probando y apuntando con precisión la eficiencia de los cuellos de botella dentro de las aplicaciones web; ofreciendo al desarrollo de aplicaciones y pruebas de equipos, la habilidad de realizar rápida y productivamente la optimización de aplicaciones web.

- **Suite eTest de Empirix**

Es una herramienta profesional de medida de rendimiento de aplicaciones web. Esta permite de un modo sencillo y visual, grabar secuencias de navegación, que después podemos utilizar:
_ Bajo demanda - Prueba de regresión.
_ Periódicamente - Prueba de disponibilidad.
_ De un modo masivo (pero simulando comportamientos reales): Velocidad al aumentar el número de usuarios.
_ De modo distribuido y simulando muchos usuarios concurrentes: prueba de capacidad.
_ Simulando muchos usuarios realmente concurrentes: pruebas de sincronismo.

ETest también nos sirve para:

_ Grabar secuencias y establecer tiempos máximos de carga de página.
_ Utilizar formularios para introducir datos variables.
_ Rellenar estos formularios con datos distintos en cada iteración.

_ Asegurarse que las páginas de retorno contienen componentes concretos.
_ Personalizar el script (que genera la herramienta automáticamente).
_ Controlar de un modo remoto agentes distribuidos para simular un comportamiento real.
_ Seleccionar la velocidad, carencia, ancho de banda de nuestro usuarios objetivo.
_ Analizar el comportamiento de las máquinas (memoria, cpu, red, etc.) según se lanzan los scripts.
_ Verificar el comportamiento de piezas complejas: Servidor de aplicaciones web, base de datos, etc.
_ Generar estadísticas integradas.

- **Apache JMeter**

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.

Esta herramienta se basa en el concepto de “Elemento” (Element), y en una estructura en “Árbol” (Tree). Cualquier parte o rama del árbol, puede ser guardada de forma independiente, para ser reutilizada en otras pruebas. Los “elementos” van a permitir reutilizar y definir el plan de pruebas.

Una característica importante del JMeter es que es extensible y ofrece la posibilidad de que el propio usuario desarrolle en Java un controlador (“Controller”) a medida, cumpliendo una interfaz Java y depositando el .jar correspondiente al desarrollo en el directorio “lib” de JMeter. Además JMeter permite realizar pruebas distribuidas en distintos ordenadores que actúan como clientes.

Puede simular una gran carga en el servidor, HTTP y FTP testing y bases de datos mediante JDBC, multitarea y con grandes facilidades para extender su funcionalidad mediante plugins.

Los elementos jerárquicos del JMeter son:

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

Los Elementos ordenados serían:

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

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 más 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 stress, 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.

Las pruebas de eficiencia bajo carga intensiva, no por ser poco conocidas y aplicadas son de menos importancia que las demás. Tanto a nivel mundial como en nuestro país aún no existe el conocimiento suficiente sobre dichas pruebas en conjunto. Existen muchas empresas que conocen de pruebas de stress y pruebas de carga, pero no de pruebas para medir los tiempos de respuesta de las aplicaciones web bajo stress.

1.2.4. En la Universidad de las Ciencias Informáticas (UCI).

En la universidad tampoco existe conocimiento suficiente de dichas pruebas, por lo que han ocurrido algunos problemas referente a la calidad. Tal es el caso de dos sistemas desplegados por Softel, en los cuales aparecieron, desde el principio de su desarrollo, dificultades durante la concurrencia. Estos sistemas fueron:

1. "Care 2X" : Este es un sistema Open source (download de Internet) para propósitos de gestión de hospitales, desplegado durante la operación Milagro en la Universidad de Ciencias Informáticas (UCI), en el verano del año 2005, para dar servicio al centro de hospitalización que tuvo lugar en la misma universidad. Falló porque a pesar de que la consulta no estaba mal diseñada, la base de datos estaba diseñada de una forma que hacía muy ineficiente esa consulta. Lo que se hizo fue cambiar el query aprovechando que los últimos datos de ubicación del paciente estaban guardados en la tabla master. Por otra parte, la forma en que estaba programada esa opción hacía que el mismo query se ejecutara al menos 2 veces por cada paciente, ya que se ejecutaba cuando cargabas la pantalla por defecto y al pinchar la opción deseada se volvía a ejecutar. Eso impedía que fueran atendidas las solicitudes.

2. Sistema de Registro de profesionales de la Salud. (Softel - MINSAP): Es un sistema para registrar los datos de los profesionales de la Salud en todo el país; el cual fue modificado y convertido temporalmente en censo. Este sistema se cargó por estudiantes durante las Brigadas Estudiantiles de Trabajo (BET) en la Universidad de Ciencias Informáticas (UCI), durante el verano del 2006. El sistema presentó dificultades en el Buscar, donde se cargaban hacia la capa de presentación muchísimos datos de todos los pacientes que cumplían con los criterios de búsqueda, que necesitaban de muchos INNER JOIN y LEFT JOIN. Esta información no se mostraba completamente, pues en una pantalla de resultado de búsqueda no caben ni se necesitan tantas columnas. A partir de esta experiencia se limitó este Buscar y se dejó la consulta grande para mandar a buscar solamente al profesional que había sido solicitado. De esta forma la búsqueda compleja se limitaba a muchos datos, pero de una sola persona y, la búsqueda sencilla, se limitaba

a menos datos de todos los que cumplían con el criterio de búsqueda. El sistema fue posteriormente corregido.

Los problemas surgidos en estos sistemas podrían haberse detectado previamente a la prueba piloto con pruebas de eficiencia bajo carga y stress. Precisamente la versión vieja del segundo de estos dos sistemas, el Sistema de Registro de Profesionales de la Salud (RPS), fue uno de los que se tomaron para probar el procedimiento para capa de acceso a datos en el presente trabajo de diploma.

El resto de los software de la UCI, también presentan problemas con el tiempo de respuesta de sus aplicaciones Web, por lo que se está tratando de enfatizar en la solución de dichos problemas.

La mayoría de los proyectos productivos de la Facultad 7 de la UCI, no poseen equipos de probadores bien definidos, pero se van realizando pruebas al software, internamente en el proyecto, a medida que se termina una etapa en el desarrollo del mismo. Al concluir el software es entregado al proyecto de calidad el cual realiza las pruebas necesarias para validar que está listo para ser liberado. Sin embargo, las pruebas de eficiencia bajo condiciones de stress no se están realizando en la universidad, a pesar de su importancia, por la falta de conocimiento.

Proponer dos procedimientos para hacer las pruebas de eficiencia bajo carga intensiva, en la capa de acceso a datos y en la capa de negocio de las aplicaciones web, seria una guía a aplicar por todos los proyectos de la facultad 7. Esta guía es de interés para el grupo de probadores de cada proyecto productivo, puesto que es una manera organizada de llevar a cabo las pruebas necesarias para conocer el tiempo de respuesta en una aplicación, durante el acceso a los datos en el servidor, o durante la navegación en las paginas de la aplicación, facilitando la corrección de errores, tanto en el código como en los queries con dificultad. En la facultad no se han hecho trabajos de maestrías u otro tipo referente a este tema.

La investigación de este tema y la propuesta de dos procedimientos para hacer pruebas de eficiencia bajo carga intensiva en aplicaciones web, parte de la necesidad de su realización durante el desarrollo de los software que se hacen en la facultad 7 de la Universidad de las Ciencias Informáticas, para la

informatización de la sociedad cubana en las diferentes esferas de la medicina, con el objetivo de lograr mayor calidad en los proyectos que se liberan para su explotación.

Se profundiza en la importancia de las pruebas de calidad de software, haciendo mayor énfasis en las de carga, stress y eficiencia, comparando las herramientas que pueden ser usadas para su desarrollo. En el capítulo 2 se explican detalladamente los dos procedimientos a seguir para llevar a cabo las pruebas de eficiencia bajo carga intensiva en aplicaciones web.

CAPÍTULO 2: PROCEDIMIENTOS PARA PRUEBAS DE EFICIENCIA BAJO CARGA INTENSIVA.

El aseguramiento de la calidad del software, debe incluir los procesos de verificación, en forma de puntos claves en el ciclo de desarrollo donde se han de introducir evaluaciones o actividades de testeo, para garantizar que éste cumple con los requisitos impuestos en la etapa inicial. Dichos requisitos, a su vez, dependerán del grado de calidad que se desee obtener en el producto final y de las exigencias del cliente.

Para que un software tenga la calidad requerida es importante que se le realicen las pruebas debidamente diseñadas, con el objetivo de conocer donde tiene problemas la aplicación. Un fallo, sobre todo de tipo funcional, puede deteriorar la imagen y tener repercusiones económicas negativas para la empresa que aplique dicho software. Sin embargo, las pruebas de eficiencia son, hoy en día, cada vez más necesarias: los tiempos de respuesta por encima de lo aceptable, la excesiva variabilidad de los mismos en función de la carga del sistema y los problemas de fiabilidad o disponibilidad deben de considerarse errores tan graves como los de funcionalidad.

Las pruebas de eficiencia bajo carga intensiva permite comprobar que un sistema está listo, en cuanto a los tiempos de respuestas esperados. Es muy distinto hacer pruebas como si sólo hubiera un usuario accediendo a una información, que hacerlas como si medio millón de usuarios accedieran a la misma información, prácticamente al mismo tiempo. Una aplicación puede comportarse bien ante una carga pequeña, pero hay que tener en cuenta como se comporta cuando es potencialmente grande la carga de usuarios accediendo a ella, por ejemplo: Una que hace uso de una base de datos; con pocos visitantes no hay problema, pero si el número de visitantes supone un número elevado de accesos simultáneos, podría tardar gran cantidad de tiempo en devolver la información que se busca, o sea, que en las aplicaciones web es complicado saber de antemano como va a comportarse ante una carga específica, en cuanto a la repuesta del sistema con respecto a las peticiones del usuario. Los sistemas web actuales son extremadamente complejos y con infinidad de tecnologías implicadas en su funcionamiento. Una forma efectiva de anticipar dicho comportamiento en una aplicación web, es realizando pruebas de eficiencia bajo carga intensiva.

La finalidad de realizar pruebas de eficiencia es simular la normal utilización del sistema antes de su paso a explotación y bajo carga intensiva, o sea, poner el sistema bajo stress, para predecir anticipadamente donde y por que tiene problemas el software en los tiempos de respuestas, para facilitar su corrección.
Realizar unas correctas pruebas de eficiencia que devuelvan resultados veraces y útiles no es una tarea sencilla. Pocas veces se pueden realizar pruebas sobre un entorno idéntico al de producción, hay que tener en cuenta que las maquinas donde se corre la aplicación son de mayor capacidad que donde se hacen las pruebas, además no se podrá simular el comportamiento exacto de un usuario real y casi siempre se manejará un cierto margen de error en los resultados.

Para pruebas de eficiencia no solo hay que tratar de conseguir que un producto o servicio se ajuste a los requisitos establecidos, sino que los procedimientos de pruebas sean permanentes, pues esto permitirá mantener y mejorar la calidad en todos los productos o servicios que se presenten; entendiendo que un procedimiento es un conjunto de tareas y decisiones sucesivas que se siguen para realizar una actividad determinada.

Para la implementación de dichas pruebas, se necesita hacer la revisión del hardware, software y herramienta necesarios. Además se selecciona el equipo de probadores y se capacita al personal de forma tal que no exista problemas para llevar a cabo las pruebas las cuales comenzarán una vez que se tenga un ambiente listo y bien definido.

Para este tema de investigación fue necesario hacer dos procedimientos: uno para capa de negocio (Ver Anexo 1) y el otro para capa de acceso a datos (Ver Anexo 2) de las aplicaciones web. Aunque hay aspectos a tratados en los procedimientos que son comunes para ambos casos y la estructura es la misma par los dos procedimientos (Ver figura 2). En dichos procedimientos se especifica un conjunto de pasos a seguir para hacer las pruebas de eficiencia poniendo a las páginas de las aplicaciones y los queries de la base de datos bajo condiciones de stress con una carga elevada de usuarios.

Con estos procedimientos se pretende definir las acciones o tareas a realizar y los documentos que se van a obtener. Los dos primeros documentos o plantillas son los de cada procedimiento (capa de negocio y capa de acceso a datos). Dentro de ellos se hará alusión a dos plantillas más, una para la planificación

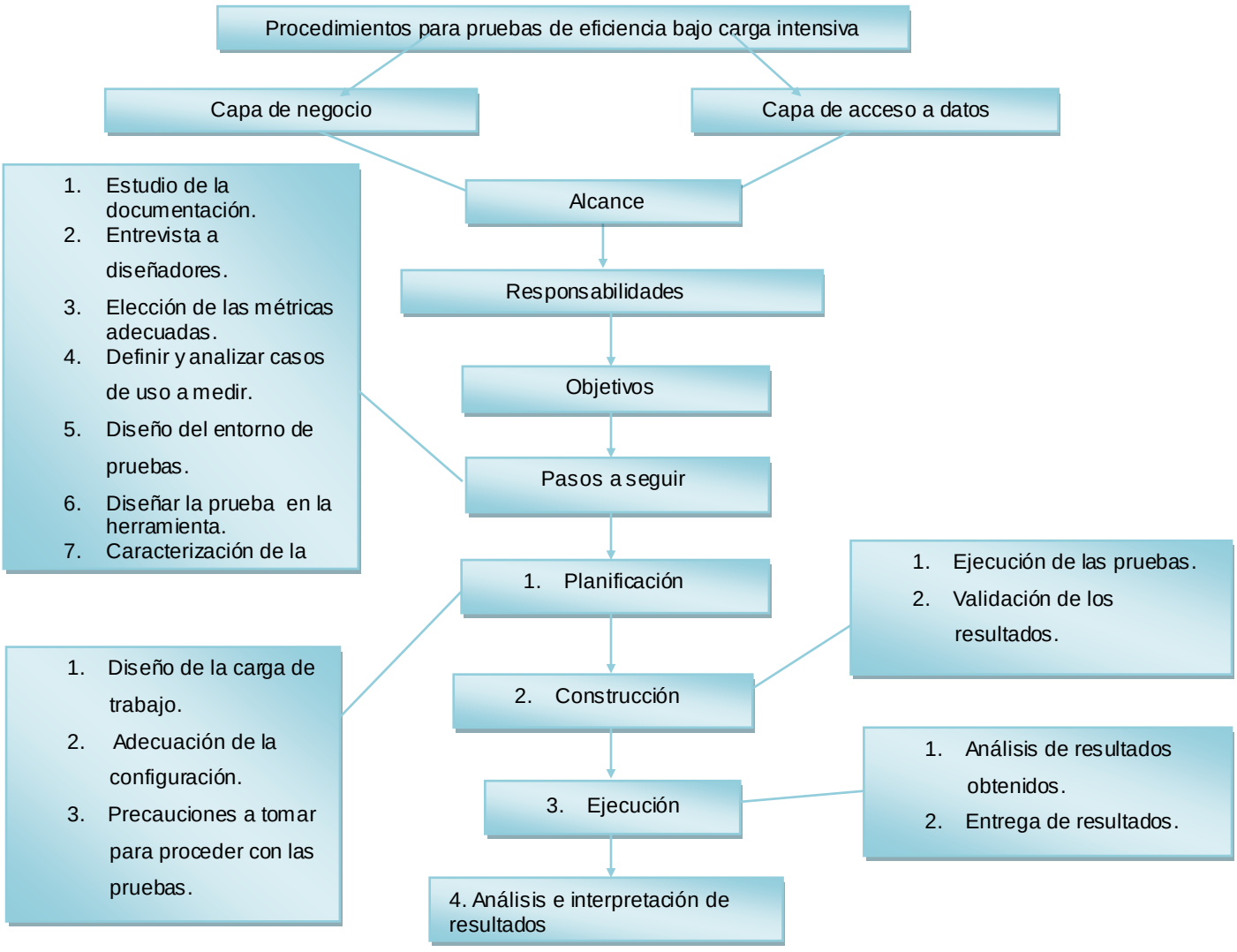
de las pruebas y otra para los resultados obtenidos. Los documentos de resultados son entregados de forma oficial al grupo de diseñadores, mientras que los restantes documentos, se quedan internamente en el grupo de probadores del proyecto.

Los procedimientos diseñados pueden realizarse por separado, pero hay que tener en cuenta que están bastante vinculados. Los tiempos de respuestas en las aplicaciones web están muy relacionados con la complejidad de las queries, o sea, que si una pantalla demora un tiempo determinado, uno de los factores que puede estar provocando la demora es el query al cual se accedió desde la pantalla.

Por eso es recomendable que se realicen primeramente las pruebas a la capa de acceso a datos y cuando se verifique que no tiene dificultades en los tiempos de respuestas, entonces pasar a las pruebas en la capa de negocio.

Para hacer las plantillas de los procedimientos se escogió como guía el documento: “Procedimiento para hacer procedimientos”, realizado en Softel, pero que puede ser utilizado por la facultad 7 de la Universidad de las Ciencias Informáticas.

Figura 2. Estructura de los procedimientos para pruebas de eficiencia bajo carga intensiva.



2.1. Alcance.

La realización de las pruebas de eficiencia bajo carga intensiva es de suma importancia, cuando se habla de garantizar la calidad del software hecho, en cuanto a la medición del tiempo de respuesta. Se espera que el desarrollo de este tema tenga alcance a nivel de facultad y de universidad más adelante. De esta forma, todos los proyectos tendrían en sus manos una guía de cómo hacer las pruebas de eficiencia, para disminuir los problemas en cuanto a la respuesta de los sistemas web.

2.2. Responsabilidades.

Los responsables de hacer las pruebas de eficiencia bajo carga intensiva a las aplicaciones web de la facultad serán: un equipo de probadores de cada proyecto de software seleccionados por el jefe de proyecto, que es el máximo responsable de verificar que se realicen dichas pruebas, junto a los jefes de grupos de trabajo.

1. Procedimiento para medir eficiencia bajo carga intensiva en la capa de negocio de las aplicaciones web.

1.2.3. Objetivos.

En un sentido amplio, las pruebas de eficiencia bajo carga intensiva en la capa de negocio, tienen los siguientes objetivos; teniendo como base para definirlos las necesidades de los proyectos en cuanto al tema de la eficiencia:

- ✓ Verificar que la capa de negocio aguantará la demanda de trabajo que debe soportar, respondiendo a las peticiones de los usuarios en un tiempo menor que el especificado por el grupo de diseñadores.

- ✓ Detectar posibles cuellos de botella e ineficiencias proporcionando la información necesaria para un correcto dimensionado.
- ✓ Hacer una estimación de cuando van a verse afectados los tiempos de respuesta, porque muchas veces no merece la pena llevar un sistema al máximo, si a partir del 60% de carga los tiempos de respuesta se van a varios segundos.

Sin embargo a la hora de enfrentarse a este tipo de pruebas de un sistema concreto se debería de afinar un poco más cuales son los objetivos que se persiguen.

Estos son algunos objetivos específicos que se pueden tener:

- ✓ 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.
- ✓ Identificar las páginas con una mayor desviación típica en sus tiempos de respuesta.

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 los objetivos antes mencionados, siempre pudiendo ser agregados otros en dependencia del interés que se persiga.

1.2.4. Pasos a seguir.

El procedimiento a seguir para la realización de unas pruebas de eficiencia adecuadas para aplicaciones web, debe comprender al menos las siguientes fases:

1. Planificación
2. Construcción

3. Ejecución
4. Análisis

1.2.4.1. Planificación.

El corazón de una correcta planificación de las pruebas de eficiencia bajo carga intensiva y una de las partes fundamentales de las mismas es la correcta caracterización del trabajo a realizar, para esto es necesario hacer énfasis en los aspectos siguientes:

1. Estudio de la documentación.
2. Entrevista a diseñadores.
3. Elección de las métricas adecuadas.
4. Definir y analizar casos de uso a medir.
5. Diseño del entorno de pruebas.
6. Diseño de las pruebas en la herramienta.
7. Caracterización de la carga de trabajo.
8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva correspondiente.

1. Estudio de la documentación:

Cuando una aplicación presenta una desviación con respecto a la funcionalidad que debería tener es porque tiene un error. La funcionalidad debe de estar correctamente detallada en su pliego de requisitos y otros documentos, los cuales debe de estar en poder del personal del departamento de certificación responsable de realizar las pruebas. Muchos de estos datos de la aplicación son de importante conocimiento para una realización satisfactoria de dichas pruebas por parte del equipo encargado hacerlas.

Uno de los documentos más importantes que hay que tener es el expediente del software a probar, según la NC, ISO/ IEC12119, este debe contener:

- La versión del producto liberada.
- Especificación de los casos de uso (CU).
- Documento de especificación de requerimiento.
- Glosario de términos.
- Manual de usuarios.
- Manual de instalación.
- Documentos con los niveles de usuarios, permisos, nombres y los detalles necesarios.
- Requerimiento mínimo de hardware y software para realizar las pruebas.
- Algunos juegos de datos que contienen la base de datos.
- Documento de la arquitectura.

No es necesario tener conocimiento de todos los aspectos antes mencionados, sino definir los necesarios para ayudar a una realización satisfactoria de las pruebas.

2. Entrevista a diseñadores.

Para las pruebas de eficiencia bajo carga intensiva, es necesario hacer entrevista a los diseñadores de la aplicación a probar para poder definir posteriormente los casos de uso a medir y de ellos, que partes se medirán comenzando por las potencialmente problemáticas.

Las entrevistas a los diseñadores se realizan con el objetivo de obtener la mayor cantidad de información posible. En ellas se recogerán informaciones importantes como son: el tiempo estándar que deben demorar las pantallas de la aplicación en responder, la cantidad de usuarios que pueden acceder a la aplicación en horario normal y cuantos en horario pico de forma concurrente, pantallas y pedazos de código problemáticos, etc. Además los diseñadores nos aportarán el conocimiento necesario para comprender la arquitectura del sistema.

3. Métricas.

Para diseñar pruebas de eficiencia bajo carga intensiva para aplicaciones web en la capa de negocio, es de suma importancia tener en cuenta tanto lo que se va a medir, como lo que se va a obtener. A continuación se especifican las métricas que se necesitan.

Variables a generar:

- Niveles de concurrencia

Variables independientes:

- Tiempo de respuesta por fragmentos de código.
- Tiempo de respuesta por navegación del usuario en la aplicación.

El principal problema durante estas pruebas radica en elegir una medida efectiva de la carga que va a ser dada a la aplicación. Algunas de las alternativas a efectuar serian:

- ✓ Número de usuarios: Es la medida más efectiva de la cantidad de carga aplicada, aunque no es muy concreta si no la caracterizamos completamente: existen demasiados parámetros involucrados en los mismos. Por ejemplo: 200 usuarios normales pueden generar la misma carga de trabajo que 20 usuarios que hacen un uso intensivo del sistema (como pulsar el siguiente enlace antes de que la página esté totalmente cargada).
- ✓ Transacciones o ciclos de trabajo: Una posibilidad es definir una transacción o ciclo de trabajo característico y propio del sistema a probar y tomarlo como unidad.

La mejor vía seria usar de forma combinada estas dos de medidas: el número de usuarios trabajando contra el sistema que da una buena idea de la carga aplicada al sistema en cada momento y de como se incrementa, y una caracterización de la misma mediante ciclos de trabajo correctamente definidos que permite conocer su volumen.

4. Casos de uso a medir.

Los casos de uso a medir serian los de las pantallas problemáticas, las cuales se definirán por muestreo, por decisión del grupo de diseñadores, o decisión del grupo encargado de hacer las pruebas, después de haber hecho un estudio de la aplicación a probar.

5. **Diseño del Entorno de pruebas.**

Diseñar y dimensionar correctamente el entorno necesario para las pruebas es un paso previo a la realización de las mismas. El entorno de pruebas no es más que el conjunto de herramientas y máquinas mediante las cuales se generan la carga que va a ser dada a la aplicación donde se va a medir la eficiencia y los servidores sobre los que se efectúan las pruebas, que comúnmente no serán los mismos sobre los cuales se realizará la explotación. Estos elementos se van a explicar de forma independiente, sería:

- Software para generar la carga.
- Hardware de los generadores de carga.
- Servidores.

Software para generar la carga.

Existen diferentes formas de generar la carga de trabajo que se necesitan para realizar las pruebas de eficiencia. La más fácil de implementar es la manual pero también es la más ineficiente la menos flexible.
Un segundo método consistiría en desarrollar generadores de carga a la medida de las necesidades que se tienen. Aunque a simple vista puede parecer que esto reporta un considerable ahorro de dinero, dado el elevado precio de las herramientas comerciales, hay que considerar que las herramientas de generación de carga son enormemente complejas y su desarrollo sería muy costoso en tiempo y dinero salvo que las necesidades se limiten a ciertos requisitos extremadamente rígidos y concretos.

La solución más comúnmente adoptada sería comprar una herramienta de generación de carga, o buscar alguna gratis y que sirva para lo que se quiere lograr, utilizándola desde el propio entorno de pruebas

construido. A la hora de elegir la herramienta adecuada a las necesidades requeridas, sin duda la característica más importante que debemos de buscar es la flexibilidad en su configuración.
La opción más conveniente para la facultad 7, seria encontrar alguna herramienta gratis que se pueda emplear para hacer las pruebas de eficiencia bajo carga, según la complejidad de lo que se quiera lograr.

En este caso la herramienta que se utilizó para aplicar las pruebas de eficiencia bajo carga intensiva entre todas las mencionadas y explicadas en el capítulo anterior fue el JMeter, debido a sus ventajas en relación con el resto. La principal ventaja que posee el JMeter se debe a que, de las herramientas gratis, es la más completa y útil para el tipo de pruebas en cuestión.

Además, el JMeter 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.

Otra de las ventajas que posee el JMeter, es que 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.

Sin duda, este es un producto a tener en cuenta cuando no se dispone de una herramienta comercial más potente como LoadRunner (que no es gratis), pero es una buena alternativa para hacer las pruebas en cuestión.

Hardware de los generadores de carga.

La mayoría de estas herramientas funcionan bajos Windows. El principal recurso que hay que vigilar en las máquinas de las que se disponen es la memoria física que es la que limitará el número de usuarios virtuales a generar. El Sistema operativo y el núcleo de la herramienta deben ocupar entre 32 y 64 Megas

de RAM. La generación los usuario virtuales ocupa otra cantidad más de memoria ya que mientras más usuarios se simulen, mayor carga de procesamiento habrá y por lo tanto más RAM se ocupa.

Evidentemente, existen otros condicionantes a tener en cuenta a la hora de dimensionar la infraestructura hardware: Por ejemplo, es deseable, una utilización de CPU por debajo del 70% y se considera no aceptable una utilización por encima del 90%.

Por otro lado, las herramientas de prueba hacen un uso intensivo de los discos para escribir los datos recogidos durante las pruebas. Se hacen, por tanto, totalmente necesarios un buen espacio de disco en las máquinas de prueba, que han de ser convenientemente mantenidos y saneados para que la carencia de espacio no provoque un error durante las pruebas. La mayoría de las computadoras de la UCI, tienen de 60 a 80 gigas de capacidad en el disco duro, que es un volumen aceptable para hacer las pruebas de eficiencia.

Servidores

Los servidores utilizados en el entorno de pruebas y, sobre todo, la arquitectura del mismo han de ser lo más parecidos posible a los servidores y la arquitectura utilizados finalmente en explotación. Cuanto más similares sean ambos entornos mayor será la exactitud de los resultados obtenidos.

6. **Diseño de pruebas en la herramienta.**

Una vez conocidas las variables independientes y las que se van a generar se procede al diseño de las pruebas en la herramienta especificada. Estas se pueden rediseñar haciendo modificaciones en el código, por ejemplo: anulando captura de datos, permitiendo el acceso concurrente de usuarios con un solo login, etc.

7. **Caracterización de la carga de trabajo.**

La demanda de trabajo a que va a estar sometido un sistema y las características de la misma sólo pueden conocerse con exactitud cuando el sistema ha estado o está sometido a explotación dando un servicio real. Cuando esto no ocurre, la única alternativa sería realizar estimaciones al respecto. La caracterización de la carga de trabajo, sea real o estimada, es un requisito indispensable para plantear las pruebas de eficiencia.

8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de negocio.

El modelo de planificación para las pruebas de eficiencia bajo carga intensiva en la capa de negocio de las aplicaciones web, es un documento donde se recogen todos los datos necesarios para proceder con las pruebas en la herramienta, o sea todos los datos que brindan los aspectos relacionados con la planificación. (Ver Anexo 3).

Para el diseño de las pruebas en la herramienta hay que definir los aspectos siguientes:

1. Nombre del proyecto a medir.
2. Nombre del caso de prueba.
3. Nombre del caso de uso a medir.
4. Número de la prueba (se pueden definir en un mismo caso de prueba más de una prueba; cuando es la misma consulta, cambiando la concurrencia y el número de ciclos o repeticiones).
5. Nivel de concurrencia o número de usuarios (se puede especificar un nivel de concurrencia diferente para cada número de prueba).
6. Transacciones o Ciclos de Trabajo (se puede especificar un ciclo diferente para cada número de prueba).
7. Pantallas a probar.

Además se pueden especificar aspectos como:

8. Tiempos de espera: Tiempo que cada usuario se toma entre el acceso a cada pantalla.

1.2.4.2. Construcción.

Después de planificadas las pruebas de eficiencia, se pasa a una segunda parte del procedimiento especificando las condiciones para construir dichas pruebas; esta segunda parte consta de:

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

1. Diseño de la carga de trabajo.

El diseño de la carga de trabajo es una de las tareas más importantes antes de la realización de las pruebas de eficiencia bajo condiciones de carga. Un diseño erróneo puede llevar a tomar conclusiones equivocadas, positiva o negativamente, respecto a los tiempos de respuestas del sistema. Este puede orientarse de la siguiente forma:

- **Diseño realista de la carga de trabajo.** Persigue realizar un modelado lo más exacto posible de la carga real de trabajo a la que se verá sometida la aplicación para anticipar de la mejor forma posible la respuesta real que tendrá la misma cuando pasé a explotación.

Los principales errores que se deben evitar cometer durante el diseño de la carga de trabajo son los siguientes:

- ✓ Representar sólo la carga media de la aplicación, por ejemplo tomar el número medio de transacciones por día para generar la carga de trabajo durante una hora ignorando los picos temporales.
- ✓ Ignorar los efectos de la caché olvidando personalizar la carga para que los distintos usuarios virtuales involucrados en la prueba accedan a los diferentes datos del sistema.
- ✓ Usar inapropiadamente los controles de los cuales se disponen para variar la carga aplicada al sistema, a saber: el número de usuarios virtuales concurrentes y los tiempos de espera de los mismos. La forma más realista de variar la carga aplicada es mediante el número de usuarios

virtuales concurrentes. Variar los tiempos de espera entre operaciones de los usuarios es el método más sencillo y utilizado, pero hay que tener en cuenta al aplicarlo que la concurrencia simultanea de usuarios es un factor muy importante a la hora de modelar los problemas del entorno real.

La mejor forma de diseñar la carga de trabajo apropiada es indagar en el perfil de los usuarios cuya actividad se quiere reproducir. Las principales variables que definen el perfil de los usuarios son:

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

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

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

Patrones de uso. Tampoco hay que olvidar a la hora de diseñar el modelo de carga la distribución horaria con la que se suelen conectar habitualmente los usuarios del sistema bajo pruebas.

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

Velocidad de acceso. Existe una gran variedad de conexiones diferentes en función de su tecnología, velocidad de acceso y ancho de banda disponible. Esto repercute en la eficiencia.

Evidentemente en la mayoría de los casos de prueba no se puede tener en consideración todos estos factores. Al acometer unas pruebas de eficiencia se tiene que detectar en primer lugar cuáles de ellos son los que en realidad afectan y en una segunda fase cuales de estos son los que se van a utilizar para el modelo a seguir.

2. Adecuación de la configuración.

Si se está interesado en hacer unas duras pruebas de stress sobre el sistema. Lo primero es caracterizar un ciclo de trabajo muy corto, de apenas 10 segundos o menos en vacío, que realice un elevado consumo de recursos en el servidor. Luego se lanzan un número determinado de usuarios virtuales, y se verifica que los tiempos de respuesta se mantengan más o menos constantes al menos durante los primeros minutos. Después se van aumentando los usuarios virtuales hasta obtener la cantidad con los que la aplicación no aguanta.

A veces es preciso modificar ciertos parámetros de la aplicación sólo para poder llevar a cabo con éxito las pruebas de eficiencia. En los sistemas web, es particularmente importante el máximo número de sesiones y el time-out con que caducan las mismas.

3. Precauciones a tener en cuenta para proceder con las pruebas.

Para una realización exitosa de las pruebas de eficiencia bajo carga intensiva, es necesario tomar algunas precauciones como son:

- ✓ La aplicación a la cual se le va a hacer las pruebas debe ser una copia de la que se está desarrollando para ser liberada porque se van a hacer cambios en el código para diseñar casos de pruebas que puedan dar errores en el tiempo de respuesta, o sea, buscar variantes para descubrir cuando la aplicación tiene retrasos en la respuesta a los usuarios que están accediendo a ella.

- ✓ Los jefes de cada proyecto encargados de llevar el control de la realización de las pruebas de eficiencia tienen que tomar medidas de seguridad para que no salga el código de la aplicación del laboratorio de producción donde se están llevando a cabo las pruebas, en caso de que sucediera, seria responsabilidad de los encargados de controlar la realización de las mismas.

1.2.4.3. Ejecución.

La tercera fase en este procedimiento es la de ejecución y como su nombre lo indica, en ella se procede a ejecutar las pruebas en la herramienta especificada, se trataran los siguientes puntos:

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

1. Ejecución de pruebas:

Se pueden comenzar a ejecutar las pruebas de eficiencia bajo carga intensiva en aplicaciones Web una vez hecha la planificación poniendo mayor énfasis en la definición de los siguientes aspectos:

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

Calificación del personal

El personal encargado de hacer las pruebas de eficiencia bajo carga intensiva en aplicaciones web, para comenzar a proceder en la actividad, debe tener conocimiento de la aplicación, HTML, PHP, dominar completamente la herramienta que va a utilizar, y tener alguna forma de acceder a la aplicación.

Además se deben tener los programas necesarios instalados para llevar a cabo dichas pruebas:

- JMeter
- Data Generator, el EMS SQL Manager for MySQL
- MySQL 4.x
- MySQL 5.x

Estas comenzarán una vez que se tenga un ambiente listo, bien definido y estén bien diseñados los casos de prueba, con el objetivo de que no aparezcan problemas como son: demoras por falta de conocimiento de la herramienta, pruebas mal hechas por falta de un buen diseño de los casos de prueba, etc.

2. Validación de los resultados.

Para validar los resultados de las pruebas de eficiencia hay que tener en cuenta la influencia que ejercen algunos parámetros sobre el tiempo de respuesta de la aplicación, introduciendo por tanto un cierto retraso en la generación de dicha carga de trabajo e interfiriendo en los resultados de la prueba.
Algunos de los parámetros son:

Utilización de la CPU. Una utilización por debajo del 70% del tiempo de CPU indica un nivel aceptable mientras que una utilización por encima del 90% indica que la máquina está sobrecargada.

Utilización de memoria. Igualmente, una utilización de memoria por encima del 150% de la física indica que la actividad que está realizando la máquina cliente invalida los resultados de la prueba. Valores por debajo de 125% son aceptables.

Existen otros factores adicionales, que pueden invalidar los resultados de las pruebas de eficiencia:

Ancho de banda consumido. El ancho de banda del que se dispone para las pruebas está limitado por un porcentaje inferior a la capacidad teórica de la red. Si durante las pruebas se supera el ancho de banda de que se dispone, entonces se estará introduciendo un retraso adicional en la respuesta del sistema.

Tasa de errores. Una elevada cantidad de errores durante las pruebas puede significar varias cosas: problemas con los drivers de comunicaciones de las máquinas clientes, un sistema con problemas en su funcionalidad que lo invalida para ser sometido a pruebas de eficiencia o algún error en la introducción de los datos en la herramienta o personalización de la carga a la que se somete al sistema. Por lo general, no se considera admisible un porcentaje de errores superior al 1% del total de transacciones ejecutadas durante las pruebas.

Margen de error. Una vez hechas todas las consideraciones dichas hasta el momento, hay que hacer una preocupante afirmación. No importa lo cuidadoso y detallista que se querrá ser, las pruebas jamás reproducirán con total verosimilitud la carga real a la que se verá sometido el sistema cuando pase a explotación.

El mejor método para calcular el margen de error consiste en revisitar todas las asunciones que se han hecho a la hora de parametrizar la carga de trabajo virtual y construir dos pruebas adicionales: una con los valores más pesimistas posibles y otra con los más optimistas. Contaremos, entonces, con una familia de resultados con la que calcular el margen de error de las pruebas realizadas.

1.2.4.4. Análisis e interpretación de resultados.

Ya en la fase final de este procedimiento, se van a analizar los tiempos de respuesta en función de la carga que se le va a dar a la aplicación, las formas de medirlos según una herramienta determinada poniendo dicha aplicación bajo el stress necesario para encontrar errores y proceder a mejorar su eficiencia.

Luego se registran los análisis y resultados obtenidos en un documento donde se especifiquen el valor de las variables independientes obtenidas, se comparan con los especificados por el diseñador, documentando los criterios de los probadores respecto a los resultados, para dar las pruebas por satisfactorias o para aceptar como bueno el producto, de forma general el criterio del equipo de probadores debe estar enmarcado en que los tiempos de respuesta estén por debajo de los tiempos dados por los diseñadores como aceptables bajo un nivel de concurrencia determinado, por ejemplo, un

sistema que no soporta más de 500 usuarios concurrentes con unos tiempos de respuesta aceptables puede (y debe) certificarse como correcta si los usuarios potenciales nunca llegaran a 200.

Además se debe hacer una lista de posibles problemas en la aplicación que se está probando de forma detallada, entendible y concisa.

Se hará una planilla de resultados por cada caso de prueba. El documento redactado (Ver Anexo 5) se le entregará al grupo de diseñadores del proyecto de software al cual se le aplicaron las pruebas creando lazos de retroalimentación con los integrantes del dicho grupo.

2. Procedimiento para medir eficiencia bajo stress en capa de acceso a datos de una aplicación.

Las pruebas de eficiencia bajo carga intensiva que se realizan durante el acceso a la capa de datos consisten en buscar ineficiencias en la respuesta de las bases de datos, durante la petición de los usuarios de una información determinada.

2.2.3.1. Objetivos:

En un sentido amplio, las pruebas de eficiencia bajo carga intensiva en capa de acceso a datos, tienen como objetivo principal (teniendo como base para definirlo las necesidades de los proyectos en cuanto al tema de la eficiencia):

- Encontrar ineficiencia en el acceso a datos y delimitar problemas específicos proporcionando la información necesaria para que dichos problemas sean corregidos.

Sin embargo a la hora de enfrentarse a este tipo de pruebas de un sistema concreto se debería de afinar un poco más cuales son los objetivos que se persiguen.

Estos son algunos objetivos específicos que se pueden tener:

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

- ✓ Determinar el número máximo de usuarios concurrentes que pueden acceder a las bases de datos, o a un query específico.
- ✓ Identificar los queries que responden más lentos.
- ✓ Determinar el tiempo aleatorio de memoria entre la terminación de un query y el inicio de otro.

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 los objetivos antes mencionados, siempre pudiendo ser agregados otros en dependencia del interés que se persiga.

2.2.4. Pasos a seguir:

El procedimiento a seguir para la realización de unas pruebas de eficiencia adecuadas para la capa de acceso a datos, debe comprender las siguientes fases:

1. Planificación
2. Construcción
3. Ejecución
4. Análisis

2.2.4.1. Planificación.

Para una correcta planificación de las pruebas de eficiencia bajo carga intensiva hay que hacer primeramente una correcta caracterización del trabajo a realizar. Para esto es necesario hacer énfasis en los aspectos siguientes:

1. Estudio de la documentación.
2. Entrevista a diseñadores.
3. Elección de las métricas adecuadas.
4. Definir y analizar casos de uso a medir.
5. Diseño del entorno de pruebas.

6. Diseño de las pruebas en la herramienta.
7. Caracterización de la carga de trabajo.
8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva correspondiente.

1. Estudio de la documentación.

Cuando una aplicación presenta una desviación con respecto a la funcionalidad que debería tener es porque tiene un error. Dicha funcionalidad debe de estar correctamente detallada en su pliego de requisitos y otros documentos, los cuales estarán en poder del personal del departamento de certificación responsable de realizar las pruebas. Muchos de estos datos de la aplicación, son de importante conocimiento para una realización satisfactoria de las pruebas por parte del equipo encargado de hacerlas.

Uno de los documentos más importantes que hay que tener es el expediente del software a probar, según la NC, ISO/ IEC12119, este debe contener:

- La versión del producto liberada.
- Especificación de los casos de uso (CU).
- Documento de especificación de requerimiento.
- Glosario de términos.
- Manual de usuarios.
- Manual de instalación.
- Documentos con los niveles de usuarios, permisos, nombres y los detalles necesarios.
- Requerimiento mínimo de hardware y software para realizar las pruebas.
- Algunos juegos de datos que contienen la base de datos.
- Documento de la arquitectura.

No es necesario tener conocimiento de todos los aspectos antes mencionados, sino definir los necesarios para ayudar a la realización de las pruebas de eficiencia bajo carga intensiva.

2. **Entrevista a diseñadores.**

Para las pruebas de eficiencia bajo carga intensiva a la capa de acceso a datos, se necesitan hacer entrevistas a diseñadores de la aplicación, para poder definir si el tiempo que demoran los queries de los casos de uso a medir, comenzando por los potencialmente problemáticos, es menor que el especificado por el diseñador y la cantidad de usuarios concurrentes que pueden acceder a cada query.

Las entrevistas a los diseñadores de las bases de datos, se realizan con el objetivo de obtener la mayor cantidad de información posible. Además los diseñadores nos aportarán el conocimiento necesario para comprender la arquitectura del sistema.

3. **Métricas.**

Para diseñar pruebas de eficiencia bajo carga intensiva para aplicaciones web en la capa de acceso a datos, es de suma importancia tener en cuenta tanto lo que se va a medir, como lo que se va a obtener. A continuación se especifican las métricas que se necesitan.

Variables a generar:

- Niveles de concurrencia

Variables independientes:

- Tiempo de transición entre la petición a un query y la respuesta de la base de datos.
- Tiempo de respuesta de un query con diferentes usuarios concurrentes bajo diferentes situaciones de stress.
- Tiempo de respuesta de la capa de acceso a datos dando un tiempo aleatorio entre peticiones a los queries de las bases de datos, y parametrizándolos.

Uno de los problemas principales durante las pruebas radica en elegir una medida efectiva de la carga que va a ser dada a la aplicación. Algunas de las alternativas a efectuar serian:

- ✓ Número de usuarios: Es la medida más efectiva de la cantidad de carga aplicada, aunque no es muy concreta si no la caracterizamos completamente: existen demasiados parámetros involucrados en los mismos. Por ejemplo: 200 usuarios normales pueden generar la misma carga de trabajo que 20 usuarios que hacen un uso intensivo del sistema (como pulsar el siguiente enlace antes de que la página esté totalmente cargada).

- ✓ Transacciones o Ciclos de Trabajo: Una posibilidad es definir una transacción o ciclo de trabajo característico y propio del sistema a probar y tomarlo como unidad.

La mejor vía sería usar de forma combinada estas dos de medidas: el número de usuarios trabajando contra el sistema que da una buena idea de la carga aplicada en cada momento y de como esta se incrementa, y una caracterización de la misma mediante ciclos de trabajo correctamente definidos que permite conocer su volumen.

4. Casos de uso a medir.

Los queries de los casos de uso a medir son los potencialmente problemáticos. Estos se definirán por decisión del grupo de diseñadores, o decisión del grupo encargado de hacer las pruebas, después de haber hecho un estudio de la capa de acceso a datos en las cuales se efectuarán.

En caso de que los queries a medir los defina el grupo encargado de hacer las pruebas, pueden ser extraídos por diferentes variantes:

a) Sacarlos de la base de datos y agruparlos según el nivel de complejidad.

b) Navegar por la aplicación generando al mismo tiempo trazas en el servidor de bases de datos, sincronizando las horas de los servidores con las de la computadora donde se está llevando a cabo la navegación, antes de activar las trazas. Luego extraer los queries de las trazas generadas y agruparlos.

5. Diseño del entorno de pruebas.

Diseñar y dimensionar correctamente el entorno necesario para las pruebas es un paso previo a la realización de las mismas. El entorno de pruebas no es más que el conjunto de herramientas y máquinas mediante las cuales se generan la carga que va a ser dada a la aplicación donde se va a medir la eficiencia y el conjunto de servidores sobre los que se efectúan las pruebas y que comúnmente no serán los mismos sobre los cuales se realizará la explotación. Estos elementos se van a explicar de forma independiente, sería:

- Software para generar la carga.
- Hardware de los generadores de carga.
- Servidores.

Software para generar la carga.

Existen diferentes formas de generar la carga de trabajo que se necesitan para realizar las pruebas de eficiencia. La más fácil de implementar es la manual pero también es la más ineficiente y la menos flexible.

Un segundo método consistiría en desarrollar generadores de carga a la medida de las necesidades que se tienen. Aunque a simple vista puede parecer que esto reporta un considerable ahorro, dado el elevado precio de las herramientas comerciales, hay que considerar que las herramientas de generación de carga son enormemente complejas y su desarrollo sería muy costoso en tiempo y dinero salvo que las necesidades se limiten a ciertos requisitos extremadamente rígidos y concretos.

La solución más comúnmente adoptada sería comprar una herramienta de generación de carga, o buscar alguna gratis y que sirva para lo que se quiere lograr, utilizándola desde el propio entorno de pruebas construido. A la hora de elegir la herramienta adecuada a las necesidades requeridas, sin duda la característica más importante que debemos de buscar es la flexibilidad en su configuración.

La opción más conveniente para en la facultad 7 de la universidad, sería encontrar alguna herramienta gratis que pueda ser empleada para hacer las pruebas de eficiencia bajo carga intensiva, según la complejidad de lo que se quiera lograr.

En este caso la herramienta que se utilizó para aplicar dichas pruebas, entre todas las mencionadas y explicadas en el capítulo anterior fue el JMeter, debido a sus ventajas en relación con el resto. La principal ventaja que posee se debe a que, de las herramientas gratis, es la más completa y útil para el tipo de pruebas en cuestión.

Además, el JMeter es una herramienta que sirve para realizar pruebas funcionales, pero también 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.

Otra de las ventajas que posee el JMeter, es que 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.

Sin duda este es un producto a tener en cuenta si no se dispone de una herramienta comercial más potente como LoadRunner (que no es gratis), pero que es una buena alternativa para la realización de las pruebas en cuestión.

Hardware de los generadores de carga.

La mayoría de estas herramientas funcionan bajos Windows. El principal recurso que hay que vigilar en las máquinas de las que se disponen es la memoria física que es la que limitará el número de usuarios virtuales a generar. El sistema operativo y el núcleo de la herramienta deben ocupar entre 32 y 64 Megas de RAM. La generación los usuario virtuales ocupa otra cantidad más de memoria ya que mientras más usuarios se simulen, mayor carga de procesamiento habrá y por lo tanto más RAM se ocupa.

Evidentemente, existen otros condicionantes a tener en cuenta a la hora de dimensionar la infraestructura hardware: Por ejemplo, es deseable, una utilización de CPU por debajo del 70% y se considera no aceptable una utilización por encima del 90%.

Por otro lado, las herramientas de prueba hacen un uso intensivo de los discos para escribir los datos recogidos durante las pruebas. Se hacen, por tanto, totalmente necesarios un buen espacio de disco en las máquinas de prueba que han de ser convenientemente mantenidos y saneados para que la carencia de espacio no provoque un error durante las pruebas. La mayoría de las computadoras de la UCI, tienen de 60 a 80 gigas de capacidad en el disco duro, que es un volumen aceptable para hacer las pruebas en cuestión.

Servidores

Los servidores utilizados en el entorno de pruebas y, sobre todo, la arquitectura del mismo han de ser lo más parecidos posible a los servidores y la arquitectura utilizados finalmente en explotación. Cuanto más similares sean ambos entornos mayor será la exactitud de los resultados obtenidos.

6. Diseño de pruebas en la herramienta.

Después de conocidas las variables independientes y las que se van a generar se procede al diseño de las pruebas en la herramienta especificada. Antes de realizar las pruebas se necesitan preparar las condiciones en los servidores.

Para preparar las condiciones de los servidores hay que tener en cuenta que muchas veces los diseñadores dan las copias de las bases de datos sin la cantidad de datos que tienen los servidores reales, además para las pruebas de eficiencia bajo carga intensiva se necesita tener la base de datos lo más poblada posible para tener mayor probabilidad de encontrar errores, por este motivo es necesario utilizar alguna herramienta que me permita cargarla.

La herramienta a utilizar para llenar las tablas es el EMS SQL Manager 2005 de MySQL, a través de la cual se pueden cargar varias tablas con elementos ficticios del mismo tipo, de esa forma cuando se configure el número de usuarios a conectarse al mismo query, va a estar más cargado el servidor, lo que permitirá que la prueba sea más efectiva.

El MySQL Manager posee una opción que es de suma importancia para hacer las pruebas bajo carga, el Plugins, el cual me permite poblar o llenar las tablas de la base de datos con parámetros falsos, repetidos la cantidad de veces que se desee en cualquiera de las tablas existentes. Como la opción de Plugins no está activada cuando se abre por primera vez el MySQL Manager, debemos activarla. Una vez activada la opción, se busca el Test Data Generator en el cual se pueden especificar algunos campos como son: la base de datos en la cual se quiere trabajar, la tabla específica que se quiere llenar, y otros parámetros de entrada según el tipo de datos que posee la tabla. Se debe además especificar que tipo o tipos de datos queremos reproducir en la tabla y cuantas veces lo queremos reproducir. Al generar los datos ficticios, el servidor va a estar cargado, lo que va a ser muy útil para crear mayor stress a la hora de hacer las pruebas.

7. Caracterización de la carga de trabajo.

La demanda de trabajo a que va a estar sometido un sistema y las características del mismo sólo pueden conocerse con exactitud cuando el sistema ha estado o está sometido a explotación dando un servicio real. Cuando esto no ocurre, la única alternativa sería realizar estimaciones al respecto. La caracterización de la carga de trabajo, sea real o estimada, es un requisito indispensable para plantear las pruebas de eficiencia.

Cuando se disponen de datos procedentes de la aplicación que ha pasado por un periodo real de explotación. El principal aliado serán los ficheros de Log dejados por el sistema y las numerosas utilidades existentes para procesarlos y extraer datos útiles a partir de los mismos, ellos son:

a) Fichero de lentitud (Slow query log).
b) Fichero de registro (log) de errores.

c) El registro general de queries (General log).

a) El fichero de lentitud, es una traza que se genera en el My SQL, que registra todas las sentencias que tardaron un tiempo excesivo en responder a la hora de ejecutarse. O sea, MySQL escribe un archivo de registro que contiene todas las sentencias SQL que se llevaron más tiempo (en segundos) del planificado para ejecutarse completamente.

b) Otro es el fichero de registro de errores, el cual registra problemas encontrados cuando se está iniciando, ejecutando o parando el MySQL. Este contiene información que indica cuando se ha iniciado y parado MySQL y si ha ocurrido algún error critico mientras el servidor se estaba ejecutando. A través de este fichero podemos ver si algunos de los queries que se guardaron en el "slow query log" o registro de lentitud, está dentro de los que se registraron con error, y si las fechas y horas en que se llevo a cabo el error, coinciden con el momento de la ejecución de alguna de las pruebas a realizar.

c) El registro general de queries registra todas las conexiones y sentencias a un archivo. Este registro es útil cuando se quiera saber exactamente que envió el cliente a MySQL. En este fichero MySQL escribe las sentencias al registro de queries en el orden que los recibe. Este orden puede ser distinto al orden de ejecución de dichos queries. Para hacer las pruebas de eficiencia es útil porque permite conocer la hora exacta entre el acceso a un query y otro, y de ese modo poder medir el tiempo de respuesta.

Los ficheros mencionados anteriormente y que son generados por el MySQL, sirven para facilitar las pruebas de eficiencia aplicadas a la capa de acceso a datos.

8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de acceso a datos.

El modelo de planificación para las pruebas de eficiencia bajo carga intensiva para la capa de acceso a datos, es un documento donde se recogen todos los datos necesarios para proceder con dichas pruebas

en la herramienta, o sea todos los datos que brindan los aspectos relacionados con la planificación (Ver Anexo 4).

Para el diseño de las pruebas en la herramienta hay que definir primero los aspectos siguientes:

- Nombre del proyecto a medir.
- Nombre del caso de prueba.
- Nombre del caso de uso a medir.
- Número de la prueba (se pueden definir en un mismo caso de prueba más de una prueba; cuando es la misma consulta, cambiando la concurrencia y el número de ciclos o repeticiones).
- Nivel de concurrencia o número de usuarios (se puede especificar un nivel de concurrencia diferente para cada número de prueba).
- Transacciones o Ciclos de Trabajo (se puede especificar un ciclo diferente para cada número de prueba).
- Consultas a probar de dichos casos de uso.

Además se pueden especificar aspectos como:

- Tiempos de espera: tiempo que cada usuario se toma entre el acceso a cada consulta de la base de datos.
- Puntos de sincronización: forma de sincronizar la actuación de los usuarios virtuales en un determinado punto de su ejecución.

2.2.4.2. Construcción.

Después de planificadas las pruebas de eficiencia, se pasa a una segunda parte del procedimiento especificando las condiciones para construir dichas pruebas; esta segunda parte consta de:

1. Diseño de la carga de trabajo.
2. Adecuación de la configuración.

3. Precauciones a tomar para proceder con las pruebas.

1. Diseño de la carga de trabajo.

El diseño de la carga de trabajo es una de las tareas más importantes antes de la realización de las pruebas de eficiencia bajo condiciones de carga. Un diseño erróneo puede llevar a tomar conclusiones equivocadas, positiva o negativamente, respecto a los tiempos de respuestas del sistema. Este puede orientarse de la siguiente forma:

- **Diseño realista de la carga de trabajo.** Persigue realizar un modelado lo más exacto posible de la carga real de trabajo a la que se verá sometida la aplicación para anticipar de la mejor forma posible la respuesta real que tendrá la misma cuando pasé a explotación.

Los principales errores que se deben evitar cometer durante el diseño de la carga de trabajo son los siguientes:

- ✓ Representar sólo la carga media del sistema, por ejemplo tomar el número medio de transacciones por día para generar la carga de trabajo durante una hora ignorando los picos temporales de trabajo.
- ✓ Ignorar los efectos de la caché olvidando personalizar la carga para que los distintos usuarios virtuales involucrados en la prueba accedan a los diferentes datos del sistema.

Para el diseño de la carga de trabajo es necesario tener conocimiento del por que es importante quitar el query caché en el servidor de bases de datos antes de comenzar a ejecutar las pruebas a la capa de acceso a datos de la aplicación web:

La importancia de quitarle el query caché a la base de datos viene dada porque la query caché almacena el texto de un query junto con el resultado que se le envió al cliente. Si se recibe un query idéntico posteriormente, el servidor devuelve el resultado de la caché en lugar de parsear y ejecutar el query de nuevo. O sea que es muy útil en un entorno donde se tienen tablas que no cambian frecuentemente y donde el servidor recibe muchos queries idénticos, pero no a la hora de hacer las pruebas de eficiencia

bajo carga intensiva; ya que estas pruebas son precisamente para medir el tiempo que demora el servidor en dar una respuesta simulando que hay muchos usuarios a la vez accediendo a la mismo query, por lo que es indispensable que se acceda al servidor cada vez que se hace un llamado a un query de la base de datos, así sea el mismo muchas veces repetido.

Además para llevar a cabo dichas pruebas se necesitan hacer varios cambios en las bases de datos, introducir datos ficticios, cambiar algunos valores de queries para hacerlos fijos, etc.

La caché no devuelve datos antiguos. No funciona en un entorno donde se tienen muchos servidores MySQL actualizando las mismas tablas, como es el caso de las pruebas a realizar. Por esto es necesario desactivar la caché al arrancar el servidor, se hace poniendo la variable de sistema query_cache_size=0 en el my.cnf.

Al desactivar el código de caché, se podrán hacer las pruebas de eficiencia sin ningún tipo de dificultad.

- ✓ Usar inapropiadamente los controles de los cuales se disponen para variar la carga aplicada al sistema, a saber: el número de usuarios virtuales concurrentes y los tiempos de espera de los mismos. La forma más realista de variar la carga aplicada es mediante el número de usuarios virtuales concurrentes. Variar los tiempos de espera entre operaciones de los usuarios es el método más sencillo y utilizado, pero hay que tener en cuenta al aplicarlo, que la concurrencia simultanea de usuarios es un factor muy importante a la hora de modelar los problemas del entorno real.

La mejor forma de diseñar la carga de trabajo apropiada es indagar en el perfil de los usuarios cuya actividad se quiere reproducir. Las principales variables que definen el perfil de los usuarios son:

Tiempos de espera. El tiempo que cada usuario se toma entre cada actuación en el sistema es significativo para la eficiencia del mismo. Evidentemente, no provoca la misma carga el usuario que conoce a la perfección nuestro sistema y pulsa compulsivamente los enlaces antes de que las páginas se

carguen por completo, que el usuario que mantiene la aplicación en segundo plano y actualiza casualmente.

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

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

Patrones de uso. Tampoco hay que olvidar a la hora de diseñar el modelo de carga la distribución horaria con la que se suelen conectar habitualmente los usuarios del sistema bajo pruebas.

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

Velocidad de acceso. Existe una gran variedad de conexiones diferentes en función de su tecnología, velocidad de acceso y ancho de banda disponible. Esto repercute en la eficiencia.

Evidentemente en la mayoría de los casos de prueba no se puede tener en consideración todos estos factores. Al acometer unas pruebas de eficiencia se tiene que detectar en primer lugar cuáles de ellos son los que en realidad afectan y en una segunda fase cuales de estos son los que se van a utilizar para el modelo a seguir.

2. Adecuación de la configuración.

Si se está interesado en hacer unas duras pruebas de stress sobre el sistema. Lo primero es caracterizar un ciclo de trabajo muy corto, de apenas 10 segundos o menos en vacío, que realice un elevado consumo de recursos en el servidor. Luego se lanzan un número determinado de usuarios virtuales, y se verifica que los tiempos de respuesta se mantengan más o menos constantes al menos durante los primeros minutos. Luego se van aumentando los usuarios, hasta obtener la cantidad de usuarios con los que la aplicación no aguanta.

A veces es preciso modificar ciertos parámetros de los queries sólo para poder llevar a cabo con éxito las pruebas de eficiencia.

3. Precauciones a tener en cuenta para proceder con las pruebas.

Para una realización exitosa de las pruebas de eficiencia bajo carga intensiva, es necesario tomar algunas precauciones como son:

- ✓ Las pruebas de eficiencia bajo stress se deben hacer en una copia de la base de datos para no estorbar el trabajo del resto del personal que está desarrollando dicha aplicación para ser liberada.
- ✓ Los jefes de cada proyecto encargados de llevar el control de la realización de las pruebas de eficiencia tienen que:
    - Verificar que todos los programas necesarios para hacer las pruebas de eficiencia se encuentren bien instalados en las máquinas.
    - Quitar el query caché del servidor al cual se le van a hacer las pruebas.
    - Tomar medidas de seguridad necesarias para que no salga el código de la aplicación fuera del laboratorio como son:
    - Implantar acceso limitado.
    - Dar a conocer una lista de personal con acceso.
    - Dar a conocer a los encargados de hacer estas pruebas una política de salvas, la cual se recibe por escrito del equipo diseñador de la aplicación.

2.2.4.3. Ejecución.

La tercera fase en este procedimiento es la de ejecución y como su nombre lo indica, en ella se procede a ejecutar las pruebas en la herramienta especificada, se trataran los siguientes puntos:

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

1. Ejecución de pruebas

Se pueden comenzar a ejecutar las pruebas de eficiencia bajo carga intensiva en la capa de acceso a datos, cuando estén definidos los siguientes aspectos:

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

Calificación del personal.

El personal encargado de hacer las pruebas de software para comenzar a proceder en la actividad deben tener conocimiento de:

- Las bases de datos a probar.
- Base de datos.
- Ip, usuario y contraseña para acceder a las bases de datos.
- La herramienta que se va a usar para hacer las pruebas de eficiencia.

Las pruebas a las bases de datos comenzarán una vez que se tenga un ambiente listo, bien definido y estén bien diseñados los casos de prueba, con el objetivo de que no aparezcan problemas como son:

demoras por falta de conocimiento de la herramienta, pruebas mal hechas por falta de un buen diseño de los casos de prueba, etc.

2. Validación de los resultados.

En general, los resultados de las pruebas de eficiencia en bases de datos son válidos cuando no están influenciados por factores que posibiliten un cierto retraso en la generación de dicha carga de trabajo e interfirieran en los resultados de la prueba. Para esto hay que tener en cuenta:

Utilización de la CPU. Una utilización por debajo del 70% del tiempo de CPU indica un nivel aceptable mientras que una utilización por encima del 90% indica que la máquina está sobrecargada.

Utilización de memoria. Igualmente, una utilización de memoria por encima del 150% de la memoria física nos indica que la actividad deswapping que está realizando la máquina cliente invalida los resultados de la prueba. Valores por debajo de 125% son aceptables.

Existen dos factores adicionales, que pueden invalidar los resultados de las pruebas de eficiencia:

Tasa de errores. Una elevada cantidad de errores durante las pruebas puede significar varias cosas: problemas con los drivers de comunicaciones de las máquinas clientes, un sistema con problemas en su funcionalidad que lo invalida para ser sometido a pruebas de eficiencia o algún error en la grabación o personalización de la carga a la que se somete al sistema. Por lo general, no se considera admisible un porcentaje de errores superior al 1% del total de transacciones ejecutadas durante las pruebas.

Margen de error. Una vez hechas todas las consideraciones dichas hasta el momento, hay que hacer una preocupante afirmación. No importa lo cuidadoso y detallista que se querrá ser, las pruebas jamás reproducirán con total verosimilitud la carga real a la que se verá sometido el sistema cuando pase a explotación.

El mejor método para calcular el margen de error consiste en volver a verificar las precauciones tomadas a la hora de parametrizar la carga de trabajo virtual y construir dos pruebas adicionales: una con los valores más pesimistas posibles y otra con los más optimistas. Contaremos, entonces, con una familia de resultados con la que calcular el margen de error de las pruebas realizadas.

2.2.4.4. Análisis e interpretación de resultados.

Ya en la fase final de este procedimiento, se van a analizar los resultados en función de la carga que se le va a dar a la capa de acceso a datos, las formas de medir los tiempos de respuestas según una herramienta determinada poniendo los queries bajo el stress necesario para encontrar errores y proceder a mejorar su eficiencia.

Luego se registran dichos análisis y los resultados obtenidos en un documento donde se especifiquen el valor de las variables independientes que se obtuvo , compararlos con los especificados por el diseñador, documentando en el mismo los criterios de los probadores respecto a los resultados, para dar las pruebas por satisfactorias o para aceptar como bueno el producto, de forma general el criterio del equipo de probadores debe estar enmarcado en que los tiempos de respuesta de los queries estén por debajo de los tiempos dados por los diseñadores como aceptables bajo un nivel de concurrencia determinado. Además se debe hacer una lista de posibles problemas en la aplicación que se está probando de forma detallada, entendible y concisa.

Se hará una planilla de resultados por cada caso de prueba. El documento redactado (Ver Anexo 6) se le entregará al grupo de diseñadores del proyecto de software al cual se le aplicaron las pruebas creando lazos de retroalimentación con los integrantes del dicho grupo.

En este capítulo se proponen dos procedimientos a seguir para lograr buenas pruebas de eficiencia bajo carga intensiva, uno para aplicaciones web y otro para bases de datos, partiendo de estudios realizados sobre las pruebas de carga, stress y eficiencia, tratando de brindar conocimiento de cómo hacer dichas pruebas a los software de los proyectos productivos de la facultad, logrando una integración de

conocimiento y de habilidades de las pruebas en cuestión y la herramienta usada a través de la investigación.

Este proceso de pruebas no se ha usado puesto que en la facultad 7 no se habían hecho pruebas de eficiencia bajo carga intensiva con anterioridad. Sin embargo se espera que tenga resultados satisfactorio en la facultad y en la universidad y que sea reconocido y bien recibido por parte de integrantes del grupo de probadores de los diferentes proyectos productivos y los equipos de desarrollo.

CAPÍTULO 3: PRUEBAS DE EFICIENCIA BAJO CARGA INTENSIVA SIGUIENDO LOS PROCEDIMIENTOS REALIZADOS, A LOS PROYECTOS DE EJEMPLO.

El objetivo de este capítulo es aplicar los dos procedimientos diseñados en el capítulo anterior y exponer los resultados obtenidos.

Para probar el procedimiento para capa de negocio se escogió el proyecto Sistema de Gestión de Información en el Proceso de Formación de Recursos Humanos en Salud, en el cual se van a aplicar las pruebas solamente al módulo Matrícula del Estudiante. Y para probar el procedimiento para capa de acceso a datos se tienen las bases de datos ofuscadas de tres de los módulos del proyecto Atención Primaria de la Salud (APS): Ciudadano, Personal de Salud (rpsdb), y Unidad de Salud.

3.1. Breve descripción de la aplicación web donde se va a probar la capa de negocio.

La aplicación web a probar es un Sistema de Gestión de Información en el Proceso de Formación de Recursos Humanos en Salud que se desarrolló para la automatización de los procesos de gestión docente en las universidades médicas cubanas. El sistema está compuesto por 8 módulos: datos de la planilla Matrícula del Estudiante, Curso Premédico, Gestión Académica, Diagnóstico Estado de Salud, Administración, Informes, y los módulos Movimientos y Documentación se incluyen tanto en el Curso Premédico como en el de Gestión Académica.

Los usuarios del sistema poseen roles, asignados por el administrador del sistema, que garantizan la autenticación de los mismos y el acceso solo a información limitada según el rol que desempeñen.

Datos de la planilla Matrícula del Estudiante.

En este módulo se podrá insertar un nuevo estudiante al sistema llenando los datos generales del mismo, datos personales, familiares, datos de la carrera que estudiará y de su procedencia escolar en general. Una vez el estudiante se encuentre en el sistema se pueden modificar estos datos.

Los usuarios que pueden acceder a este módulo son: el editor nacional de Premédico que es quien debe introducir estos datos cuando el estudiante llega a nuestro país, la primera etapa de su formación es precisamente el curso Premédico, el editor nacional de Ciencias Médicas y el administrador nacional.

Para insertar un nuevo estudiante al sistema, se deberá llenar los formularios que van apareciendo en páginas sucesivas: página de los datos generales del estudiante, dos páginas de sus datos personales, página de datos familiares y página de datos referentes a la carrera. Si se han dejado datos sin llenar o ha cometido algún error al llenarlos, el sistema mostrará un mensaje con el error cometido.

Luego de haberse insertado el estudiante con todos sus datos en el sistema, se crea un código de identificación para el estudiante, que se mostrará inmediatamente en una página con los datos más generales de este, confirmando así que el estudiante fue insertado satisfactoriamente.

En caso de que el estudiante a insertar ya se encontrara en el sistema, se mostrará un mensaje informando la situación y no se podrá realizar la inserción.

3.2. Breve descripción la aplicación donde se va a probar la capa de acceso a datos.

El sistema donde se va a probar la capa de acceso a datos es Atención Primaria de la Salud (APS), que tiene como objetivo principal informatizar la salud en Cuba. Dicho sistema posee varios módulos, de los cuales hay 5 terminados, 2 en construcción, 4 en mantenimiento y otros que todavía están por hacer. Este sistema es bastante complejo por lo que se ha invertido en él gran cantidad de tiempo y esfuerzo. Es un sistema que se va a ir aplicando poco a poco en todos los policlínicos del país, a medida que van siendo liberados los diferentes módulos que lo componen.

El sistema posee una base de datos para cada módulo y en muchas de ellas se hacen llamadas a datos de otros servidores, por lo que los queries están muy relacionadas entre si.

Del proyecto APS solo se van a utilizar las bases de datos ofuscadas de tres de sus módulos: Registro de Ciudadanos, Registro de Unidad de Salud y Registro de Personal de Salud (rps).

3.3. Pruebas de eficiencia bajo carga intensiva para capa de acceso a datos.

Para aplicar los procedimientos diseñados en los proyectos mencionados anteriormente, se llevarán a cabo diferentes casos de pruebas, los cuales se realizarán de la siguiente forma:

3.3.1. Caso de prueba 1: Base de datos ofuscada del módulo Unidad de Salud de la aplicación del proyecto Atención Primaria de la Salud (APS).

Objetivo: Encontrar ineficiencia en el acceso a los queries de la base de datos ofuscadas de los módulos a probar del proyecto APS y delimitar problemas específicos proporcionando la información necesaria para que dichos problemas sean corregidos.

Pasos a seguir

1. Planificación

1.1. Documentación.

La aplicación del proyecto APS donde se van a realizar las pruebas siguiendo el procedimiento hecho, tiene del expediente de software los siguientes aspectos documentados: la versión del producto liberada, la especificación de los casos de uso, glosario de términos, el manual de usuario, loa juegos de datos de la base de datos, el documento de especificación de requerimientos.

1.2. Entrevista a diseñadores.

De las entrevistas a los diseñadores se definieron:

- ✓ Las pantallas que tenían acceso a los queries problemáticos de la base de datos del módulo Unidad de Salud.
- ✓ Tiempo estándar que deben demorar los queries de la dicha base de datos en responder: no debe exceder los 1000 ms.

- ✓ Cantidad de usuarios que pueden acceder a los queries en cuestión de forma concurrente: menos de 100 usuarios.

1.3. Definir y analizar casos de uso a medir.

Se van a medir los queries que son accedidos desde los casos de uso insertar y buscar profesional de salud.
Los queries a probar fueron definidos por el encargado de hacer las pruebas. Su obtención se lleva a cabo a través de la navegación en la aplicación de los módulos especificados, apuntando los tiempos que se demora la misma en responder cuando se llama a alguna consulta, generando al mismo tiempo trazas en el servidor de bases de datos, sincronizando las horas del servidor con las computadoras donde se está llevando a cabo la navegación, antes de activar las trazas. Luego se extraen los queries de las trazas generadas y se agrupan.

1.4. Diseño del entorno de pruebas.

Software para generar la carga: La herramienta a usar para generar la carga necesaria y medir los tiempos de respuesta en los queries de la base de datos del módulo Unidad de Salud es el JMeter.

Hardware de los generadores de carga:

- Sistema Operativo: Windows XP Profesional.
- Memoria RAM: 512Mbytes
- Disco duro: 80 Gigas.

Servidores: Se buscó que los servidores utilizados en el entorno de pruebas y la arquitectura del mismo fueran lo más parecidos posible a los servidores y la arquitectura utilizados finalmente en explotación para lograr exactitud en los resultados.

1.6. Diseño de las pruebas en la herramienta.

Para diseñar la prueba en el JMeter que es la herramienta a utilizar se procede de la siguiente forma:

El componente principal del JMeter es denominado Plan de Prueba o Test Plan, en él se definen todos los aspectos relacionados con una prueba de carga, como : parámetros empleados, tipo de reportes a generarse con los resultados obtenidos, la posible reutilización de requisiciones compuestas por usuarios, entre otros aspectos.

El plan de prueba sale por defecto cuando se abre la herramienta.

Seguidamente se muestra paso a paso el Plan de Prueba utilizado para este primer caso de prueba.

Lo primero es crear un Thread Group o Grupo de Hilos, un "Thread Group" es considerado como el grupo de usuarios que se desea simular para la aplicación o la base de datos. Una vez creado el grupo de hilos, se llenan las siguientes opciones:

- Name: Se utiliza para definir un nombre más descriptivo sobre el grupo de usuarios, en este caso de pruebas no se especifico ningún nombre.

- Number of Threads (users): Equivale al número de usuarios que se desean simular.

- Ramp-Up Period: Es el lapso de tiempo en segundos que se desea tener entre cada grupo de usuarios ("Thread Group"), se usará en este caso 0 segundo, o sea que se quiere que los usuarios repitan el acceso a las queries la cantidad de veces especificadas de forma seguida.

- Loop Count o Forever: Se utiliza para indicar si la simulación para grupos de hilos ("Thread Group") será llevada acabo infinitamente, o por un ciclo determinado de veces. Esto es, si se selecciona la opción Forever indica que se desea simular la cantidad de usuarios especificada, por segundo, de forma infinita. Para esta prueba se va a definir un número determinado de ciclos.

Una vez definidas las características del Grupo de Hilo o Thread Group, se pasa a generar las JDBC Connection Configuration (se crea una para cada base de datos a probar). Las JDBC Connection

Configuration se usan para configurar las conexiones de las bases de datos a probar. Esta posee los siguientes aspectos:

- Nombre de la variable (Variable Name): En este caso se pone BD_UnidadesSalud
- Database URL: Se pone la url de la base de datos a acceder: jdbc:mysql://10.128.3.30:3306/ BD_UnidadesSalud
- JDBC Driver class: Siempre se pone: com.mysql.jdbc.Driver
- El usuario y contraseña de la base de datos.

El resto de los campos que aparecen están predefinidos por defecto.
Luego se generan las JDBC Request utilizadas para definir las requisiciones de simulación, en la cual aparecerán las siguientes opciones:

- Name: Se define el nombre que se le quiera dar a la request. Se crea una JDBC Request para cada query. En el caso de estudio se le puso nombre JDBC Request 1, 2, 3, etc.

- Variable Name: Se le pone el mismo nombre que se le puso a Variable Name de la JDBC Connection Configuration. En este caso se hicieron 51 JDBC Request, una de cada consulta de la base de datos.

- Query Type: Se selecciona el tipo de query, Select Statement en el caso de que se está probando puesto que son queries de tipo Select los tres que se van a probar.

- Query: En este campo se escribe la consulta.

Luego se hace un Agregate Graph y un View Results in Table, para ver los resultados. También se pueden generar otros Listener que me generan gráficas de resultados, sumarios, etc.
Finalmente se guarda el plan de prueba y se ejecuta.

1.7. Caracterización de la carga de trabajo.

La carga de trabajo a la cual va a estar sometido el sistema durante las pruebas va a ser similar a la carga que tendrá durante su explotación.

1.8. Modelo de planificación.

- **Planificación de pruebas de eficiencia bajo carga intensiva en la capa de acceso a datos.**

- **Planificación de prueba 1 del caso de prueba 1.**

Nombre del proyecto a medir: Atención Primaria de la Salud (APS).
Nombre del caso de prueba: 1.
Nombre del caso de uso a medir: Unidad de Salud.
Número de la prueba: 1
Nivel de concurrencia: 10
Ciclos de trabajo: 1
Queries a probar: Se van a probar 51 queries juntos, unos más problemáticos que otros. Uno de los de mayor dificultad es el primero:

```
SELECT distinct t_unidad.id_unidad,
t_unidad.nombre_unidad,
t_unidad.id_tipou,
c_tipo_unidad.nombre_tipou,
t_unidad.codigo_unidad,
t_unidad.direccion,
t_unidad.ubicacion,
t_unidad.codigo_reeup,
t_unidad.subordinacion,
t_unidad.tipicidad,
t_unidad.id_organismo,
c_organismos.nombre_organismo,
t_unidad.id_grupo,
```

```
c_grupos.nombre_grupo,
t_unidad.id_municipio,
t_unidad.activa,
t_unidad.observacion,
t_unidad.localidad,
t_rel_provmun.id_provincia
FROM t_unidad
LEFT JOIN c_tipo_unidad on t_unidad.id_tipou=c_tipo_unidad.id_tipou
LEFT JOIN c_grupos on t_unidad.id_grupo=c_grupos.id_grupo
LEFT JOIN c_organismos ON t_unidad.id_organismo=c_organismos.id_organismo
LEFT JOIN t_val_desgatribcual ON t_unidad.id_unidad = t_val_desgatribcual.id_unidad
LEFT JOIN tvalor_cap ON t_unidad.id_unidad = tvalor_cap.id_unidad
LEFT JOIN t_rel_provmun ON t_unidad.id_municipio = t_rel_provmun.id_municipio WHERE activa=1
Order by nombre_unidad
```

- **Planificación de prueba 2 del caso de prueba 1.**

Nombre del proyecto a medir: Atención Primaria de la Salud (APS).
Nombre del caso de prueba: 1
Nombre del caso de uso a medir: Unidad de Salud.
Número de la prueba: 2
Nivel de concurrencia: 50
Ciclos de trabajo: 1
Queries a probar: Se van a probar 51 queries juntos, unos más problemáticos que otros. Otro de los de mayor dificultad es el segundo:

```
SELECT c_atribcual.id_atribcual, c_atribcual.nombre_atribcual,
t_desg_atribcual.id_desglose_atribcual, t_desg_atribcual.nombre_desgcualt,
t_unidad.id_unidad, t_unidad.nombre_unidad
FROM t_unidad INNER JOIN t_val_desgatribcual ON t_unidad.id_unidad = t_val_desgatribcual.id_unidad
```

INNER JOIN t_desg_atribcual ON t_val_desgatribcual.id_desglose_atribcual = t_desg_atribcual.id_desglose_atribcual
INNER JOIN c_atribcual ON t_desg_atribcual.id_atribcual = c_atribcual.id_atribcual
WHERE t_unidad.id_unidad

Todos los datos especificados anteriormente se van a recoger en la planilla de planificación de procedimientos para capa de acceso a datos (Ver Anexo 4).

2. Construcción.

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

2.1. Diseño de la carga de trabajo.

Tomar todas las medidas necesarias para no cometer ninguno de los errores especificados durante el diseño de la carga de trabajo.

2.2. Adecuación de la configuración.

Para esta prueba no serán pasados parámetros a los queries, ni se definirán tiempos aleatorios entre ellos.

2.3. Precauciones a tomar.

- La base de datos a probar es una copia ofuscada de la base de datos original.
- Se llevará un control exhaustivo durante la realización de las pruebas por parte de los jefes de proyecto.

3. Ejecución.

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

3.1. Ejecución de las pruebas.

Se pueden ejecutar las pruebas una vez que esté todo listo y bien definido y el personal esté capacitado par proceder con las mismas.

3.2. Validación de los resultados.

Para la validación de los resultados hay que tener en cuenta:

- Utilización de la CPU: Menor de 70%
- Utilización de la memoria: Menor de 125%
- Tasa de errores: Menor de 1%

4. Análisis y resultados de las pruebas para capa de acceso a datos.

Se tiene como inconveniente que dicho proceso se pudo hacer solamente una vez, por lo que no es segura la precisión de las horas tomadas, además que no se pudieron acceder a todas las páginas previstas porque hubo problemas con las bases de datos ofuscadas, y en muchos de los casos daba error al hacer algún pedido al servidor. No obstante con la información que se obtuvo fue suficiente para hacer las pruebas requeridas, aunque no se pudieron hacer pruebas de mucha complejidad. Los detalles de dichas pruebas con los resultados obtenidos se especifican a continuación.

Resultados obtenidos en prueba 1:

- El tiempo máximo de respuesta por query (especificando solamente el tiempo de los más problemáticos):

Consulta1: 6403 ms

Consulta2: 1295 ms

El tiempo del resto de los queries está entre 15ms y 1000 ms, dependiendo de la complejidad de los queries:

- % de error: En todos los queries probados hubo en 0 % de error, según la herramienta especificada.

Resultados obtenidos en prueba 2:

- El tiempo máximo de respuesta por query (especificando fundamentalmente el tiempo de los más problemáticos):

Consulta1: 34438 ms
Consulta2: 10187 ms

El resto de los queries oscilan en un tiempo entre 100 ms y 2000 ms, de acuerdo con su complejidad:

- % de error.

Consulta1: 80%
Consulta2: 78%
El resto de los queries no dieron error.

Criterios de probadores en cuanto a:

Una vez realizada la prueba en la herramienta JMeter, se pudo observar que con 10 usuarios concurrentes y con un ciclo de trabajo igual a 1, para los 51 queries, o sea, 510 repeticiones, la base de datos de forma satisfactoria, sin generar ningún margen de error; sin embargo para 50 usuarios concurrentes, y el resto de los datos igual, la respuesta de la base de datos es dada con un % de error inaceptable en los dos primeros queries que son los de mayor dificultad, llevándose a cabo 2550 repeticiones en la herramienta. En las tablas de resultados se puede ver como estos errores en los dos

primeros queries, provocan que algunas de los queries posteriores no se ejecuten, o que se atrasen en su ejecución.

Se compararon los resultados obtenidos con los especificados por grupo de diseñadores y se llegó a la conclusión que los tiempos de respuesta de los queries problemáticos están por encima de los tiempos dados por los diseñadores como aceptables bajo un nivel de concurrencia determinado.
Las pruebas realizadas tuvieron éxito, encontrando retrasos en los tiempos de respuesta y errores provocados por la concurrencia en los queries cargados.

Lista de posibles problemas en la aplicación.

- Problemas en la respuesta de los queries debido a su complejidad.

- Error en los queries problemáticos debido a la concurrencia, o sea, el sistema no soporta la cantidad de concurrencia que le fue asignada para la prueba, por lo que explotaron los queries de mayor dificultad, afectando la respuesta de algunos queries posteriores.

- Error en el sistema debido a que la propia herramienta JMeter se estresa cuando se pasa de unos niveles determinados de concurrencia, por tanto hay que ejecutarla en computadoras rápidas y de mucha RAM. O utilizar una facilidad que posee la herramienta de carga distribuida que se está siendo estudiada actualmente.

- Error en el sistema debido a que la herramienta es gratis, por lo que tiene sus dificultades y todavía no se domina a profundidad

Recomendaciones:

- Se recomienda optimizar los queries 1 y 2 que son las más grandes para que respondan en el tiempo correcto.

- Se recomienda seguir estudiando y profundizando en el uso de la herramienta para lograr unas pruebas de eficiencia efectivas.

3.3.2. Caso de prueba 2: Pantallas del módulo Matrícula del Estudiante de la aplicación del proyecto Sistema de Gestión de Información en el Proceso de Formación de Recursos Humanos en Salud.

Objetivos:

- Verificar que la capa de negocio aguantará la demanda de trabajo que debe soportar, respondiendo a las peticiones de los usuarios en un tiempo menor que el especificado por el grupo de diseñadores.

- Detectar posibles cuellos de botella e ineficiencias proporcionando la información necesaria para un correcto dimensionado.

Pasos a seguir

5. **Planificación**

1.3. Documentación.

La aplicación del proyecto en cuestión, tiene del expediente del software los siguientes aspectos documentados: la versión del producto liberada, la especificación de los casos de uso, requerimientos funcionales y no funcionales, glosario de términos, el manual de instalación, loa juegos de datos de la base de datos.

1.4. Entrevista a diseñadores.

De las entrevistas a los diseñadores se definieron:

- Tiempo estándar que deben demorar las pantallas de la aplicación en responder: menos de 500 ms.
- Cantidad de usuarios que pueden acceder a la aplicación en horario normal de forma concurrente: 50 usuarios.
- Cantidad de usuarios que pueden acceder a la aplicación en horario pico de forma concurrente: 100 usuarios.
- Pantallas y fragmentos de código problemáticos.

Definir y analizar casos de uso a medir.

Se van a medir las pantallas del caso de uso insertar estudiante.
Las pantallas a probar fueron definidas el grupo de diseñadores de la aplicación.

1.5. Diseño del entorno de pruebas.

Software para generar la carga: La herramienta a usar para generar la carga necesaria y medir los tiempos de respuesta en los queries de la base de datos del módulo Unidad de Salud es el JMeter.

Hardware de los generadores de carga:

- Sistema Operativo: Windows XP Profesional.
- Memoria RAM: 512Megas
- Disco Duro: 80 Gigas.

Servidores: Se buscó que los servidores utilizados en el entorno de pruebas y la arquitectura del mismo fueran lo más parecidos posible a los servidores y la arquitectura utilizados finalmente en explotación para lograr exactitud en los resultados.

1.6. Diseño de las pruebas en la herramienta.

Para diseñar la prueba en el JMeter que es la herramienta a utilizar se procede de la siguiente forma:

Una vía que tiene el JMeter y que se usó para hacer este caso de prueba, es generar dicho caso de prueba a través de una navegación de usuario, para ello hay que indicar que JMeter actúe como proxy, para capturar la navegación del caso de prueba. El objetivo seria medir los tiempos entre páginas por las que se navegó y compararlas en cuanto al tiempo de respuesta.

Se le puede poner un nombre al HTTP Proxy Server, lo que es ente caso no se hizo. Además se puede indicar que almacene solo la primera request de cada petición de la navegación, de este modo se descartan las imágenes.

- Patterns to Include: Se escribe el parámetro que queremos que se incluya en la navegación de la siguiente forma:

- .*\.php
- .*\.http

Patterns to Exclude: Se escribe el parámetro que queremos que se excluya en la navegación, o sea que no ponga dichas Request, esto se hace de la misma forma que en el de inclusión, En el caso de prueba que se muestra se excluyeron las fotos:

- .*\.jpg
- .*\.gif
- .*\.gif
- .*\.png

Una vez configurado el HTTP Proxy Server, se arranca el servidor Proxy de JMeter. Luego se va al navegador y se configura el proxy (localhost, 8080). Y después, nuevamente en la herramienta, se teclea la URL de la web a probar, para el caso de prueba fue la del módulo Matricula del estudiante:

/TesisYilen/Gestion_Academica_Cubanos/Paginas/Datos_Matricula_Cubanos/Planilla_Inicial.php.

Automáticamente, JMeter va capturando las Request que el navegador realiza al proxy. Después de parar el proxy, se pueden eliminar los elementos que no interesan y reorganizarlos en el árbol, para ello se crea un Thread Group y un Simple Controller dentro del Grupo de Hilos y se mueven las HTTP Request al Simple Controller. El Simple Controller es un controlador simple, que se usa para agrupar una serie de peticiones.

Las HTTP Request que el navegador realiza al proxy y que captura el JMeter, aparecen en la herramienta con el nombre igual al camino o Path que recorrió: /TesisYilen/Gestion_Academica_Cubanos/Paginas/Datos_Matricula_Cubanos/Planilla_Inicial.php.

En este caso se dejó con ese mismo nombre pero puede ser cambiado. También aparecen en las Request que se generan, el resto de las informaciones más importantes completadas, como son el ip, el número de puerto, etc.

Luego se eliminan los “http Head Manager” de cada request, ya que de lo contrario el test puede que no funcione, al tener grabado en la cabecera valores de otra sesión. También es recomendable añadir un “http Cookie Manager” delante del controlador. Después se agrega un Listener Aggregate Report dentro del Simple Controller, para ver los resultados de la prueba, y por ultimo se ejecuta la prueba. Por último se guarda el plan de prueba y se ejecuta obteniendo una serie de resultados en el Agregate Report.

1.8. Caracterización de la carga de trabajo.

La carga de trabajo a la cual va a estar sometido el sistema para la prueba, será mayor que la especificada por el diseñador.

1.8. Modelo de planificación.

- **Planificación de pruebas de eficiencia bajo carga intensiva en la capa de negocio.**

- **Planificación de prueba 1 del caso de prueba 2.**

Nombre del proyecto a medir: Sistema de Gestión de Información en el Proceso de Formación de Recursos Humanos en Salud.
Nombre del caso de prueba: 2.
Nombre del caso de uso a medir: Matricula del Estudiante.
Número de la prueba: 1
Nivel de concurrencia: 40
Ciclos de trabajo: 2
Pantallas a probar: Se van a probar 3 pantallas del módulo Matricula de Estudiante, módulo de mayor dificultad según la opinión del diseñador.

Las pantallas a probar son:
- Insertar datos generales del estudiante.
- Insertar datos personales1.
- Insertar datos personales 2.

- **Planificación de prueba 2 del caso de prueba 2.**

Nombre del proyecto a medir: Sistema de Gestión de Información en el Proceso de Formación de Recursos Humanos en Salud.
Nombre del caso de prueba: 2.
Nombre del caso de uso a medir: Matricula del Estudiante.
Número de la prueba: 2
Nivel de concurrencia: 100
Ciclos de trabajo: 1
Pantallas a probar: Se van a probar 3 pantallas del módulo Matricula de Estudiante, módulo de mayor dificultad según la opinión del diseñador.

Las pantallas a probar son:
- Insertar datos generales del estudiante.
- Insertar datos personales1.
- Insertar datos personales 2.

Todos los datos especificados anteriormente se van a recoger en la planilla de planificación de procedimientos para capa de negocio (Ver Anexo 3).

2. Construcción.

4. Diseño de la carga de trabajo.
5. Adecuación de la configuración.
6. Precauciones a tomar para proceder con las pruebas.

2.1. Diseño de la carga de trabajo.

Hay que tomar todas las medidas necesarias para no cometer ninguno de los errores especificados durante el diseño de la carga de trabajo.

2.2. Adecuación de la configuración.

Para esta prueba se modificaran algunos parámetros para anular la captura de datos, y para permitir la concurrencia con un solo login de un usuario.

2.3. Precauciones a tomar.

- La aplicación a la cual se le va a hacer las pruebas debe es copia de la que se está desarrollando debido a los cambios que se van a hacer en el código.
- Se llevará un control exhaustivo durante la realización de las pruebas por parte de los jefes de proyecto.

3. Ejecución.

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

3.1. Ejecución de las pruebas.

Se pueden ejecutar las pruebas una vez que esté todo listo y bien definido y el personal esté capacitado par proceder con las mismas.

3.2. Validación de los resultados.

Para la validación de los resultados hay que tener en cuenta:

- Utilización de la CPU: Menor de 70%
- Utilización de la memoria: Menor de 125%
- Tasa de errores: Menor de 1%

4. Análisis y resultados de las pruebas para capa de negocio.

Resultados obtenidos en prueba 1:

- El tiempo máximo de respuesta por pantallas.
    - Insertar datos generales del estudiante: 305 ms.
    - Insertar datos personales1: 92 ms.
    - Insertar datos personales2: 58 ms.
- % de error: 0% en las tres pantallas probadas.
- Tiempo de respuesta total de las pantallas:

Resultados obtenidos en prueba 2:

- El tiempo máximo de respuesta por pantallas.
    - Insertar datos generales del estudiante: 359 ms.

- Insertar datos personales1: 150 ms.
- Insertar datos personales2: 88 ms.

- % de error: o% en las tres pantallas.

Criterios de probadores en cuanto a:

Una vez realizada la prueba en la herramienta JMeter, se pudo observar que con 40 usuarios concurrentes y repitiendo el ciclo de trabajo dos veces, para las 3 pantallas, efectuándose un total de 240 repeticiones, la aplicación responde de forma satisfactoria, sin generar ningún margen de error; y para 100 usuarios concurrentes, haciéndose el ciclo de trabajo solo una vez, para las 3 pantallas, efectuándose en total 300 repeticiones, la aplicación responde de forma correcta igualmente, por lo que se puede aceptar como buena la aplicación. La aplicación probada no posee pantallas con gran nivel de complejidad.

Se compararon los resultados obtenidos con los especificados por grupo de diseñadores y se llegó a la conclusión que los tiempos de respuesta de las pantallas están por debajo de los tiempos dados por los diseñadores como aceptables bajo un nivel de concurrencia determinado.
Las pruebas realizadas no tuvieron éxito, porque no se encontraron retrasos en los tiempos de respuesta.

Se aplicaron los procedimientos en los proyectos escogidos para hacer las pruebas, diseñando pruebas en el JMeter, recogiendo los resultados obtenidos para ser entregados al grupo de diseñadores. De esta manera se llevó a cabo un lazo de retroalimentación con los mismos, con el objetivo de lograr mayor calidad en dichos proyectos, siempre a partir del conocimiento de la capa de acceso a datos y la capa de negocio de las aplicaciones de muestra.

CONCLUSIONES

A partir de la investigación realizada sobre las pruebas de software en la facultad 7 de la Universidad de las Ciencias Informáticas, profundizando en las pruebas de eficiencia bajo carga intensiva, se llegó a la conclusión que se necesitan dos procedimientos para la realización de dichas pruebas en aplicaciones web.

Por este motivo se estudiaron las pruebas de eficiencia, stress y carga aplicadas a nivel mundial. Se compararon un conjunto de herramientas de gestión de dichas pruebas, para sugerir y decidir cuáles se adaptan mejor a las necesidades de los proyectos productivos de la facultad, diseñando dos procedimientos para la gestión de pruebas de eficiencia bajo carga intensiva en aplicaciones web, uno para la capa de acceso a datos y el otro para la capa de negocio.

Por último se validaron los procedimientos diseñados para realizar las pruebas, sobre dos proyectos de la facultad 7, tomando como base la herramienta elegida, estableciendo lazos de retroalimentación para los desarrolladores de dichos proyectos de software.

Realizar dichas pruebas sirvió para mejorar los procedimientos iterativamente.

RECOMENDACIONES

- ✓ Evaluar el procedimiento propuesto con la finalidad de tener resultados prácticos del mismo.

- ✓ Utilizar el Procedimiento propuesto para las pruebas de eficiencia bajo carga intensiva en aplicaciones web, en los diferentes proyectos productivos de la Facultad 7 y hacerlo extensivo al resto de la Universidad en los casos que sea posible.

- ✓ Seguir validando y perfeccionando este procedimiento de pruebas de eficiencia bajo carga intensiva, poniendo hincapié en el uso de la herramienta JMeter.

REFERENCIAS BIBLIOGRÁFICAS

- Curiel, M. (2005). "Introducción a la Evaluación del Desempeño de Sistemas Informáticos".
- Casas, I. (2006) Técnicas de Pruebas de Cargas. http://cursos.puc.cl/html/iic3532-1/almacen/20050222173726_Trans7x1BN_sec1.PDF.
- Feiler, J. (1999). Database-Driven Web Sites, Morgan Kaufmann.
- Gevert, R. (1990) Siguiendo la pista a cada cuello de botella. www.gestiopolis.com.
- Mañas, J.A. (2007) Pruebas de Programas. http://www.cienciasmisticas.com.ar/informatica.
- León, D.G. (2007) La Calidad Total - Sistemas de Aseguramiento de la Calidad. http://www.iaf.es/webiaf.nsf.
- Pressman, R. (2002). "Ingeniería de Software, Un enfoque práctico", Quinta edición, McGraw-Hill.
- Sanz, L..F. (2005) Calidad e Ingeniería del Software, Revista Española de Innovación Vol.1, No. 2.
- Visitación, M. (2007). "Métodos y Tecnología de Sistemas y Procesos (MTP)."
- Weyuker, E. J, Vokolos F. I. "Experience with Performance Testing of Software Systems": Issues, an Approach, and Case Study. IEEE Transactions on Software.

BIBLIOGRAFÍA

- A. Sabiola. Web Load Test Planning. Predicting how your Web Site will respond to stress. STQE Magazine.
- A. Sabiola. Response Time. Understanding and measuring performance test results. STQE Magazine.
- Booch, G.: Rumbaugh, J. y Jacobson, I.; "El Lenguaje Unificado de Modelado". 2000. Addison-Wesley.
- Dr. D. Luís Fernández Sanz, Dr. D. Juan José Cuadrado-Gallego. Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS).2005.
- Dr. José A. Gutiérrez. Elaboración de Procedimientos. 2006.
- E. Ambichi. Very Large Scale Load Testing with SilkPerformer. Segue Press.
- E. J. Weyuker, F. I. Vokolos. Experience with Performance Testing of Software Systems: Issues, an Approach, and Case Study. IEEE Transactions on Software Engineering.
- E. Metz. Efficient Instrumentation for Performance Profiling. http://www.cs.nmsu.edu/~jcook/woda2003/papers/Metz.pdf.
- EMS SQL Manager 2005 de MySQL.
- G. Conde. Introducción a SilkPerformer y Pruebas de carga. MTP.
- Juan Angel Martínez Párraga. Planificación y Gestión de Sistemas de Información. Estándar IEEE 1219 de mantenimiento del software. Mayo, 1999.
- José María Morales Vázquez. Métodos y Tecnología de Sistemas y Procesos (MTP).2005.
- Jean Pierre Norguet. http://jakarta.apache.org/jmeter/. Enero, 2006.
- Jacobson, I.; Booch, G. y Rumbaugh, J.; "El Proceso Unificado de Desarrollo de software". 2000. Addison-Wesley.
- K. Johnson. Mining Gold from Server Logs. Using existing site data to improve Web Testing. STQE Magazine.
- Lloyd G. Williams y Connie U. Smith. Five Steps to Solving Software Performance Problems. Junio, 2002.
- Lidia Fuentes, José M. Troya y Antonio Vallecillo. Desarrollo de Software Basado en Componentes. 2006.

- Lisa Crispin, Tip House. Testing Extreme Programming. Octubre 25, 2002.
- Luis Sacristán. http://sentidoweb.com/2006/10/20/lista-de-herramientas-para-testeo-de-aplicaciones-web.php. Octubre, 2006.
- MSc. Yamilis Fernández Pérez. Proceso de pruebas de aceptación para un software de gestión.2006.
- Manual de referencia de MySQL 5.0. agosto, 2006.
- Prof. Mariela Curiel. Introducción a la Evaluación del Desempeño de Sistemas Informáticos.
- Prof. Ignacio Casas. Técnicas de Pruebas de Cargas. http://cursos.puc.cl/html/iic3532-1/almacen/20050222173726_Trans7x1BN_sec1.PDF
- Pressman, Roger; Ingeniería de software. Un enfoque práctico. 2002.
- Practicas actuales de garantía de calidad. www.compuware.com. 2005.
- RUP, Ayuda de Rational Unified Process, 2003.
- Roberto Canales Mora. Tutoriales. Medidas de Rendimiento de Aplicativos Web. www.autentia.com. 2003-2005.
- Roberto Canales Mora. JMeter.abril, 2005.
- S. Splaine y S. P. Jaskiel. The Web Testing Handbook. STQE Publishing.
- Vincent Massol. JUnit in Action. Julio 08, 2003.
- .http://www.als-es.com/home.php?location=herramientas. 2007.

ANEXOS

Anexo1. Procedimiento para pruebas de eficiencia bajo carga intensiva en capa de negocio de aplicaciones web.

TITULO: Pruebas de eficiencia bajo carga intensiva para capa de negocio en aplicaciones web.

COPIA CONTROLADA N^{O} : **1** **ASIGNADA A :**	
REDACTADO POR : (*) **Yilen Pons Ramirez**	
REVISADO POR : (*)	
APROBADO POR: (*)	**FECHA DE ENTRADA EN VIGOR :**

(*) NOMBRE(S), FIRMA(S), FECHA(S)

RELACION DE REVISIONES

REV.	FECHA	PAGINAS AFECTADAS	REDACTADO	REVISADO	APROBADO

INDICE

1. Alcance.

Aplicar a los proyectos productivos de la facultad 7 y posteriormente a los proyectos de la Universidad de Ciencias Informáticas.

2. Responsabilidades.

Los responsables de hacer dichas pruebas so el equipo de probadores de cada proyecto de software.
Los responsables de verificar que se realicen dichas pruebas son los jefes de proyecto y jefes de grupos de trabajo.

3. **Objetivos.**

3.1. Objetivos generales.

- Verificar que la capa de negocio aguantará la demanda de trabajo que debe soportar, respondiendo a las peticiones de los usuarios en un tiempo menor que el especificado por el grupo de diseñadores.
- Detectar posibles cuellos de botella e ineficiencias proporcionando la información necesaria para un correcto dimensionado.
- Hacer una estimación de cuando van a verse afectados los tiempos de respuesta.

3.2. Objetivos específicos.

- 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.

- Identificar las páginas con una mayor desviación típica en sus tiempos de respuesta.

4. Pasos a seguir.

1. Planificación.
2. Construcción.
3. Ejecución.
4. Análisis.

4.1. Planificación.

1. Estudio de la documentación.
2. Entrevista a diseñadores.
3. Elección de las métricas adecuadas.
4. Definir y analizar casos de uso a medir.
5. Diseño del entorno de pruebas.
6. Diseño de las pruebas en la herramienta.
7. Caracterización de la carga de trabajo.
8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva correspondiente.

4.1.1. Estudio de la documentación.

Lo más importante que se debe tener es el expediente del software a probar, según la NC, ISO/ IEC12119, este debe contener:

- La versión del producto liberada.
- Especificación de los casos de uso (CU).
- Documento de especificación de requerimientos.
- Glosario de términos.
- Manual de usuarios.
- Manual de instalación.
- Documentos con los niveles de usuarios, permisos, nombres y los detalles necesarios.

- Requerimiento mínimo de hardware y software para realizar las pruebas.
- Algunos juegos de datos que contienen la BD.
- Documento de la Arquitectura.

No es necesario tener conocimiento de todos los aspectos antes mencionados, pero si de los necesarios para una realización satisfactoria de las pruebas.

4.1.2. Entrevista a diseñadores.

Las entrevistas con los diseñadores deben brindar información referente a:

- Tiempo estándar que deben demorar las pantallas de la aplicación en responder.
- Cantidad de usuarios que pueden acceder a la aplicación en horario normal de forma concurrente.
- Cantidad de usuarios que pueden acceder a la aplicación en horario pico de forma concurrente.
- Pantallas y pedazos de código problemáticos.

4.1.3. Métricas.

Variables a generar:

- Niveles de concurrencia

Variables independientes:

- Tiempo de respuesta por fragmentos de código.
- Tiempo de respuesta por navegación del usuario en la aplicación.

4.1.4. Casos de uso a medir.

Escoger los casos de uso de las pantallas problemáticas para sacar los lugares críticos y de esa forma que exista mayor posibilidad de encontrar errores.

4.1.5. Diseño del entorno de pruebas.

Es el conjunto de herramientas y máquinas mediante las cuales se generan la carga que va a ser dada a la aplicación donde se va a medir la eficiencia y el conjunto de servidores sobre los que se efectúan las pruebas y que comúnmente no serán los mismos sobre los cuales se realizará la explotación, estos elementos serían:

- **Software para generar la carga:** herramienta para hacer las pruebas de eficiencia bajo carga. JMeter.
- **Hardware de los generadores de carga:** Tener en cuenta:

- Sistema Operativo.
- Memoria física.
- Memoria RAM.
- Utilización de la CPU.

- **Servidores:** Los servidores utilizados en el entorno de pruebas y, sobre todo, la arquitectura del mismo han de ser lo más parecidos posible a los servidores y la arquitectura utilizados finalmente en explotación para lograr exactitud en los resultados.

4.1.6. Diseño de las pruebas en la herramienta.

Tratar de rediseñar las pruebas haciendo modificaciones en el código:

- Anulando captura de datos.
- Permitiendo el acceso concurrente de usuarios con un solo login.
- etc.

4.1.7. Caracterización de la carga de trabajo.

Estimar cual será la carga de trabajo a la cual va a estar sometido el sistema, y las características del mismo.

4.1.8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de acceso a

datos en las aplicaciones web.

Es un documento donde se recogen todos los datos necesarios para proceder con las pruebas en la herramienta, o sea todos los datos que brindan los aspectos relacionados con la planificación. (Ver Anexo 3).

4.2. Construcción.

Este paso consta de:

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

4.2.1. Diseño de la carga de trabajo.

Persigue realizar un modelado lo más exacto posible de la carga real de trabajo a la que se verá sometida la aplicación para anticipar de la mejor forma posible la respuesta real que tendrá la misma cuando pasé a explotación.

Los principales errores que se deben evitar cometer durante el diseño de la carga de trabajo son los siguientes:

- Representar sólo la carga media de la aplicación, por ejemplo tomar el número medio de transacciones por día para generar la carga de trabajo durante una hora ignorando los picos temporales de trabajo.
- Ignorar los efectos de la caché olvidando personalizar la carga para que los distintos usuarios virtuales involucrados en la prueba accedan a los diferentes datos del sistema.
- Usar inapropiadamente los controles de los cuales se disponen para variar la carga aplicada al sistema, a saber: el número de usuarios virtuales concurrentes y los tiempos de espera de los mismos.

4.2.2. Adecuación de la configuración.

- Caracterizar un ciclo de trabajo muy corto, de apenas 10 segundos o menos en vacío, que realice un elevado consumo de recursos en el servidor.
- Lanzar un número determinado de usuarios virtuales y se verifica que los tiempos de respuesta se mantienen más o menos constantes al menos durante los primeros minutos.
- Lanzar un número de usuarios mayor a anterior y así sucesivamente hasta obtener la cantidad de usuarios con los que la aplicación no aguanta.

Modificar parámetros de la aplicación:

- El máximo número de sesiones.
- El *time-out* con que caducan las mismas.

4.2.3. Precauciones a tomar para proceder con las pruebas.

- La aplicación a la cual se le va a hacer las pruebas debe ser una copia de la que se está desarrollando para ser liberada porque se van a hacer cambios en el código para diseñar casos de pruebas que puedan dar errores en el tiempo de respuesta, o sea, buscar variantes para descubrir cuando la aplicación tiene retrasos en la respuesta a los usuarios que están accediendo a ella.

- Los jefes de cada proyecto encargados de llevar el control de la realización de las pruebas de eficiencia tienen que tomar medidas de seguridad para que no salga el código de la aplicación del laboratorio de producción donde se están llevando a cabo las pruebas, en caso de que sucediera, seria responsabilidad de los encargados de controlar la realización de las mismas.

4.3. Ejecución.

En la fase de ejecución de este procedimiento, se trataran los siguientes puntos:

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

4.3.1. Ejecución de las pruebas.

Se pueden comenzar a ejecutar las pruebas cuando están definidos los siguientes aspectos:

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

Calificación del personal.

El personal encargado de hacer las pruebas de software para comenzar a proceder en la actividad deben tener conocimiento de:

- la aplicación Web a probar.
- HTML.
- PHP.
- La herramienta que se va a usar para hacer las pruebas de eficiencia.
- Alguna forma de acceder a la aplicación.

Además debe tener los programas necesarios instalados con la herramienta que se va a usar para llevar a cabo dichas pruebas.

- JMeter
- Data Generator el EMS SQL Manager for MySQL
- MySQL 4.x
- MySQL 5.x

Las pruebas a la aplicación comenzarán una vez que se tenga un ambiente listo y bien definido.

4.3.2. Validación de los resultados.

Hay que tener en cuenta la influencia que ejercen algunos parámetros sobre el tiempo de respuesta de la aplicación, introduciendo por tanto un cierto retraso en la generación de dicha carga de trabajo e interfiriendo en los resultados de la prueba.

Algunos de los parámetros son:

- **Utilización de la CPU.** Una utilización por debajo del 70% del tiempo de CPU indica un nivel aceptable mientras que una utilización por encima del 90% indica que la máquina está sobrecargada.
-
- **Utilización de memoria.** Una utilización de memoria por encima del 150% de la memoria física nos indica que la actividad que está realizando la máquina cliente invalida los resultados de la prueba. Valores por debajo de 125% son aceptables.

Tener en cuenta otros factores adicionales, que pueden invalidar los resultados de las pruebas de eficiencia:

- **Ancho de banda consumido.**
- **Tasa de errores elevada y su significado.** Por lo general, no se considera admisible un porcentaje de errores superior al 1% del total de transacciones ejecutadas durante las pruebas.

4.4. Análisis e interpretación de los resultados.

- Analizar los tiempos de respuesta en función de la carga que se le va a dar a la aplicación.
- Registrar dichos análisis y los resultados obtenidos en un documento donde se especifiquen el valor de las variables independientes que se obtuvo, documentando los criterios de los probadores respecto a los resultados, para dar las pruebas por satisfactorias o para aceptar como bueno el producto.
- Hacer una lista de posibles problemas en la aplicación que se está probando de forma detallada, entendible y concisa.

El documento redactado con los resultados (ver Anexo 5) se le entregara al grupo de diseñadores del proyecto de software al cual se le aplicaron las pruebas.

5. Documentos de referencia.

- Elaboración de Procedimientos realizado por la empresa Softel.

Anexo2. Procedimiento para pruebas de eficiencia bajo carga intensiva en capa de acceso a datos de aplicaciones web.

TITULO: Pruebas de eficiencia bajo carga intensiva para capa de acceso a datos en aplicaciones web.

COPIA CONTROLADA N^{O} : **ASIGNADA A :**	
REDACTADO POR : (*)	**Yilen Pons Ramírez**
REVISADO POR : (*)	
APROBADO POR: (*)	**FECHA DE ENTRADA EN VIGOR :**

(*) NOMBRE(S), FIRMA(S), FECHA(S)

RELACION DE REVISIONES

REV.	FECHA	PAGINAS AFECTADAS	REDACTADO	REVISADO	APROBADO

INDICE

1. Alcance.

Aplicar a los proyectos productivos de la facultad 7 y posteriormente a los proyectos de la Universidad de Ciencias Informáticas.

2. Responsabilidades.

Los responsables de hacer dichas pruebas so el equipo de probadores de cada proyecto de software.

Los responsables de verificar que se realicen dichas pruebas son los jefes de proyecto y jefes de grupos de trabajo.

3. Objetivos.

3.1. Objetivos generales.

- Encontrar ineficiencia en el acceso a datos y delimitar problemas específicos proporcionando la información necesaria para que dichos problemas sean corregidos.

3.2. Objetivos específicos.

- Determinar el tiempo medio de respuesta que obtendrá el usuario.
- Determinar el número máximo de usuarios concurrentes que pueden acceder a las bases de datos, o a un query específico.
- Identificar los queries que responden más lentos.
- Determinar el tiempo aleatorio de memoria entre la terminación de un query y el inicio de otro.

4. Pasos a seguir.

1. Planificación.
2. Construcción.
3. Ejecución.
4. Análisis.

4.1. Planificación.

1. Estudio de la documentación.
2. Entrevista a diseñadores.

3. Elección de las métricas adecuadas.
4. Definir y analizar casos de uso a medir.
5. Diseño del entorno de pruebas.
6. Diseño de las pruebas en la herramienta.
7. Caracterización de la carga de trabajo.
8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva correspondiente.

4.1.1. Estudio de la documentación.

Lo más importante que se debe tener es el expediente del software a probar, según la NC, ISO/ IEC12119, este debe contener:

- La versión del producto liberada.
- Especificación de los casos de uso (CU).
- Documento de especificación de requerimientos.
- Glosario de términos.
- Manual de usuarios.
- Manual de instalación.
- Documentos con los niveles de usuarios, permisos, nombres y los detalles necesarios.
- Requerimiento mínimo de hardware y software para realizar las pruebas.
- Algunos juegos de datos que contienen la base de datos.
- Documento de la arquitectura.

No es necesario tener conocimiento de todos los aspectos antes mencionados, pero si de los necesarios para una realización satisfactoria de las pruebas.

4.1.2. Entrevista a diseñadores.

Las entrevistas con los diseñadores deben brindar información referente a:

- Tiempo estándar que deben demorar los queries de la aplicación en responder.

- Cantidad de usuarios que pueden acceder a los queries de forma concurrente.
- Queries problemáticos.

4.1.3. Métricas.

Variables a generar:

- Niveles de concurrencia

Variables independientes:

- Tiempo de transición entre la petición a un query y la respuesta de la base de datos.
- Tiempo de respuesta de un query con diferentes usuarios concurrentes bajo diferentes situaciones de stress.
- Tiempo de respuesta de la capa de acceso a datos dando un tiempo aleatorio entre peticiones a los queries de las bases de datos, y parametrizándolos.

4.1.4. Casos de uso a medir.

Escoger los casos de uso que posean queries potencialmente problemáticos.
Estos van a ser definidos por:

- Decisión del grupo de diseñadores.
- Decisión del grupo encargado de hacer las pruebas, después de haber hecho un estudio de la capa de acceso a datos en las cuales se efectuarán las pruebas.

En caso de que los queries a medir los defina el grupo encargado de hacer las pruebas, pueden ser extraídos por diferentes variantes:

- Sacar los queries agrupándolos según nivel de complejidad.
- Navegar por la aplicación generando al mismo tiempo trazas en el servidor de bases de datos, sincronizando las horas del servidor con las computadoras donde se está llevando a cabo la

navegación, antes de activar las trazas. Luego extraer los queries de las trazas generadas y agruparlos.

4.1.5. Diseño del entorno de pruebas.

Es el conjunto de herramientas y máquinas mediante las cuales se generan la carga que va a ser dada a la aplicación donde se va a medir la eficiencia y el conjunto de servidores sobre los que se efectúan las pruebas y que comúnmente no serán los mismos sobre los cuales se realizará la explotación, estos elementos serían:

- **Software para generar la carga:** herramienta para hacer las pruebas de eficiencia bajo carga. JMeter.
- **Hardware de los generadores de carga:** Tener en cuenta:

- Sistema Operativo.
- Memoria RAM.
- Utilización de la CPU.

- **Servidores:** Los servidores utilizados en el entorno de pruebas y, sobre todo, la arquitectura del mismo han de ser lo más parecidos posible a los servidores y la arquitectura utilizados finalmente en explotación para lograr exactitud en los resultados.

4.1.6. Diseño de las pruebas en la herramienta.

- Preparar las condiciones en los servidores. Se va a utilizar la herramienta MySQL Manager 2005 de MySQL.

4.1.7. Caracterización de la carga de trabajo.

Estimar cual será la carga de trabajo a la cual va a estar sometido el sistema, y las características del mismo.

Pueden servir de ayuda:

- Fichero de lentitud (Slow query log).

- Fichero de registro (log) de errores.
- El registro general de queries (General log).

4.1.8. Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de acceso a datos en las aplicaciones web.

Es un documento donde se recogen todos los datos necesarios para proceder con las pruebas en la herramienta, o sea todos los datos que brindan los aspectos relacionados con la planificación (Ver Anexo 4).

4.2. Construcción.

Este paso consta de:

1. Diseño de la carga de trabajo.
4. Adecuación de la configuración.
5. Precauciones a tomar para proceder con las pruebas.

Diseño de la carga de trabajo:

Persigue realizar un modelado lo más exacto posible de la carga real de trabajo a la que se verá sometida la aplicación para anticipar de la mejor forma posible la respuesta real que tendrá la misma cuando pasé a explotación.

Los principales errores que se deben evitar cometer durante el diseño de la carga de trabajo son los siguientes:

- Representar sólo la carga media de la aplicación, por ejemplo tomar el número medio de transacciones por día para generar la carga de trabajo durante una hora ignorando los picos temporales de trabajo.
- Ignorar los efectos de la caché olvidando personalizar la carga para que los distintos usuarios virtuales involucrados en la prueba accedan a los diferentes datos del sistema.

- Usar inapropiadamente los controles de los cuales se disponen para variar la carga aplicada al sistema, a saber: el número de usuarios virtuales concurrentes y los tiempos de espera de los mismos.

Adecuación de la configuración.

- Caracterizar un ciclo de trabajo muy corto, de apenas 10 segundos o menos en vacío, que realice un elevado consumo de recursos en el servidor.
- Lanzar un número determinado de usuarios virtuales y se verifica que los tiempos de respuesta se mantienen más o menos constantes al menos durante los primeros minutos.
- Lanzar un número de usuarios mayor a anterior y así sucesivamente hasta obtener la cantidad de usuarios con los que la aplicación no aguanta.

Modificar parámetros de los queries:

- Dar Tiempos aleatorios entre queries.
- Pasar parámetros.

Precauciones a tomar para proceder con las pruebas.

- Las pruebas de eficiencia bajo stress se deben hacer en una copia de la base de datos para no estorbar el trabajo del resto del personal que está desarrollando dicha aplicación para ser liberada.
- Los jefes de cada proyecto encargados de llevar el control de la realización de las pruebas de eficiencia tienen que:

- Verificar que todos los programas necesarios para hacer las pruebas de eficiencia se encuentren bien instalados en las máquinas.
- Quitar el query caché del servidor al cual se le van a hacer las pruebas.
- Tomar medidas de seguridad necesarias para que no salga el código de la aplicación fuera del laboratorio como son:

- Implantar acceso limitado.
- Dar a conocer una lista de personal con acceso.

- Dar a conocer a los encargados de hacer estas pruebas una política de salvas, la cual se recibe por escrito del equipo diseñador de la aplicación.

4.3. Ejecución.

En la fase de ejecución de este procedimiento, se trataran los siguientes puntos:

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

4.3.1. Ejecución de las pruebas.

Se pueden comenzar a ejecutar las pruebas cuando están definidos los siguientes aspectos:

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

Calificación del personal.

El personal encargado de hacer las pruebas al software, para comenzar a proceder en la actividad deben tener conocimiento de:

- Las bases de datos a probar.
- Base de datos.
- Ip, usuario y contraseña para acceder a las bases de datos.
- La herramienta que se va a usar para hacer las pruebas de eficiencia.

Además debe tener los programas necesarios instalados con la herramienta que se va a usar para llevar a cabo dichas pruebas.

- JMeter
- Data Generator el EMS SQL Manager for MySQL
- MySQL 4.x
- MySQL 5.x

Las pruebas a la aplicación comenzarán una vez que se tenga un ambiente listo y bien definido.

4.3.2. Validación de los resultados.

Hay que tener en cuenta la influencia que ejercen algunos parámetros sobre el tiempo de respuesta de la aplicación, introduciendo por tanto un cierto retraso en la generación de dicha carga de trabajo e interfiriendo en los resultados de la prueba.

Algunos de los parámetros son:

- **Utilización de la CPU.** Una utilización por debajo del 70% del tiempo de CPU indica un nivel aceptable mientras que una utilización por encima del 90% indica que la máquina está sobrecargada.
- **Utilización de memoria.** Una utilización de memoria por encima del 150% de la memoria física nos indica que la actividad que está realizando la máquina cliente invalida los resultados de la prueba. Valores por debajo de 125% son aceptables.

Tener en cuenta otros factores adicionales, que pueden invalidar los resultados de las pruebas de eficiencia:

- **Ancho de banda consumido.**
- **Tasa de errores elevada y su significado**. Por lo general, no se considera admisible un porcentaje de errores superior al 1% del total de transacciones ejecutadas durante las pruebas.

4.4. Análisis e interpretación de los resultados.

- Analizar los tiempos de respuesta en función de la carga que se le va a dar a la aplicación.
- Registrar dichos análisis y los resultados obtenidos en un documento donde se especifiquen el valor de las variables independientes que se obtuvo, documentando los criterios de los probadores respecto a los resultados, para dar las pruebas por satisfactorias o para aceptar como bueno el producto.
- Hacer una lista de posibles problemas en la aplicación que se está probando de forma detallada, entendible y concisa.

El documento redactado con los resultados (ver Anexo 6) se le entregara al grupo de diseñadores del proyecto de software al cual se le aplicaron las pruebas.

5. Documentos de referencia.

- Elaboración de Procedimientos realizado por la empresa Softel.

Anexo3. Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de negocio en aplicaciones web.

softel soluciones informáticas	UCI Universidad de las Ciencias Informáticas				
Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de negocio.					
Título del procedimiento: Pruebas de eficiencia bajo carga intensiva para capa de negocio.					
Nombre del proyecto a medir:					
Nombre del caso de uso a medir:					
Nombre del caso de Prueba:					
Número de prueba: (según las que se realicen):	**1**	**2**	**3**	**4**	**Etc...**

Nivel de concurrencia o número de usuarios por prueba:					
Transacciones o ciclos de trabajo por prueba:					
Tiempos de espera por prueba (opcional):					
Pantallas a probar					
1.					
2.					
3.					
Etc.…					

Anexo 4. Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de acceso a datos.

softel soluciones informáticas	UCI Universidad de las Ciencias Informáticas
Modelo de planificación de pruebas de eficiencia bajo carga intensiva para capa de acceso a datos.	
Título del procedimiento: Pruebas de eficiencia bajo carga intensiva para capa de acceso a datos.	
Nombre del proyecto a medir:	
Nombre del caso de uso a medir:	
Nombre del caso de Prueba:	

Número de prueba: (según las que se realicen):	1	2	3	4	Etc.…
Nivel de concurrencia o número de usuarios por prueba:					
Transacciones o ciclos de trabajo por prueba:					
Tiempos de espera por prueba (opcional):					
Puntos de sincronización por prueba (opcional):					
Queries a probar					
1.					
2.					
3.					
4.					
5.					
Etc.…					

Anexo 5. Análisis y resultados de las pruebas de eficiencia bajo carga intensiva para capa de negocio.

softel soluciones informáticas	UCI Universidad de las Ciencias Informáticas
Análisis e interpretación de los resultados de las pruebas de eficiencia bajo carga intensiva en capa de negocio.	
Título del procedimiento:	Pruebas de eficiencia bajo carga intensiva para capa de negocio.

Nombre del proyecto probado:				
Nombre del caso de uso probado:				
Nombre del caso de Prueba:				
Resultados obtenidos por cada prueba realizada.				
Número de prueba:	1	2	3	Etc.
Tiempo máximo de respuesta por pantalla:				
% de error:				
Criterio de los probadores.				
Lista de problemas en la aplicación.				
1.				
2.				
3.				
Etc.....				

Anexo 6. Análisis y resultados de las pruebas de eficiencia bajo carga intensiva para capa de acceso a datos.

softel soluciones informáticas	UCI Universidad de las Ciencias Informáticas
Análisis e interpretación de los resultados de las pruebas de eficiencia bajo carga intensiva en capa de acceso a datos.	

Título del procedimiento:	Pruebas de eficiencia bajo carga intensiva para capa de acceso a datos.			
Nombre del proyecto probado:				
Nombre del caso de uso probado:				
Nombre del caso de Prueba:				
Resultados obtenidos por cada prueba realizada.				
Número de prueba:	1	2	3	Etc.
Tiempo máximo de respuesta por consulta:				
% de error:				
Criterio de los probadores.				
Lista de problemas en la aplicación.				
1.				
2.				
3.				
4.				