Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Desarrollo de SoftRen v2.0: sistema para realizar pruebas de carga y estrés a aplicaciones cliente servidor utilizando la Plataforma de Tareas Distribuidas T-arenal.

Autor:

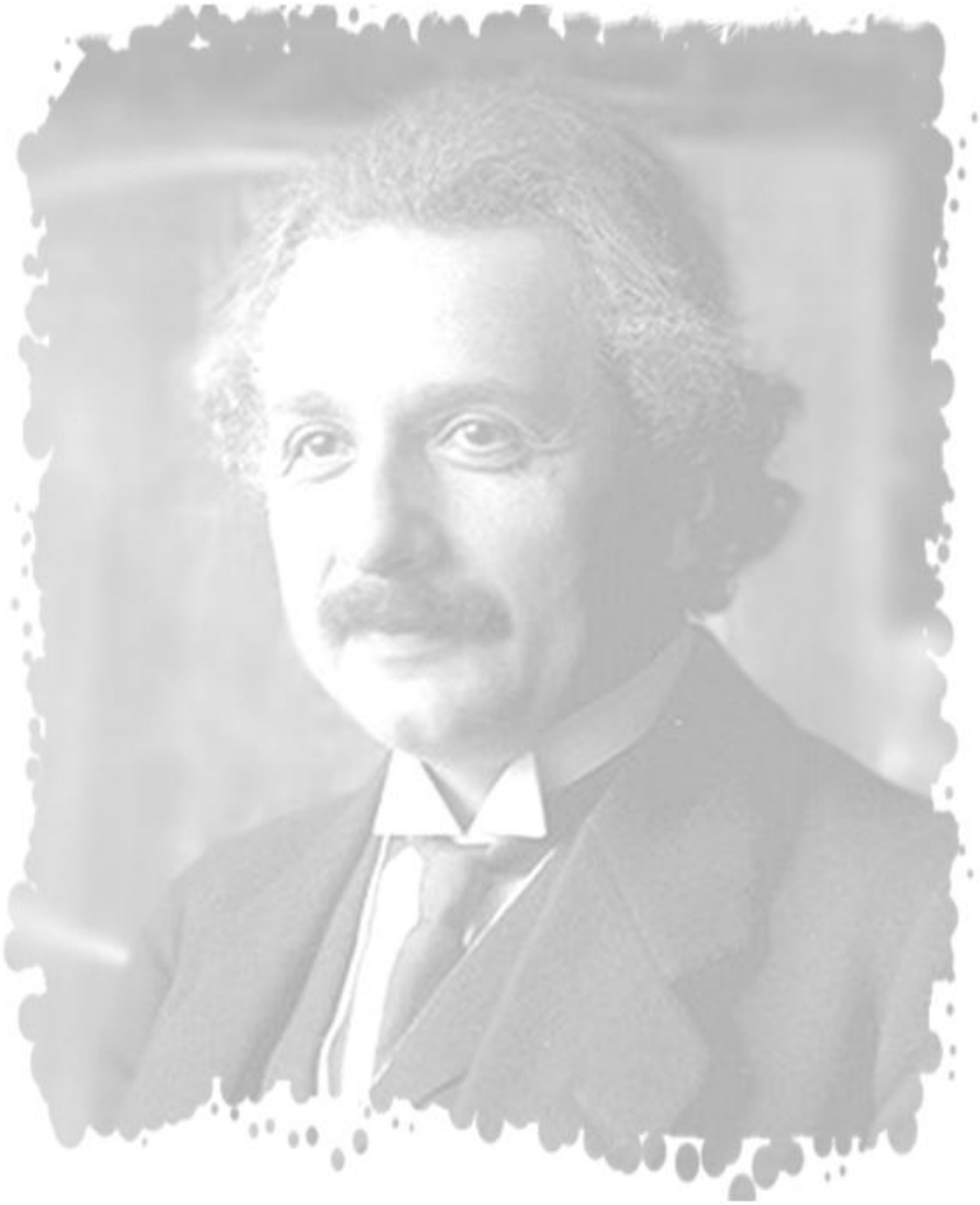
Damián Umpierre Albelo

Tutora:

Ing. Adisley Reyes Crespo

Ciudad de la Habana, junio de 2012

"Año 54 de la Revolución"



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber"

Albert Einstein

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

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

Damián Umpierre Albelo	Adisley Reyes Crespo
_____	_____
Firma del Autor	Firma del Tutor

Datos de contacto

Tutor:

Ing. Adisley Reyes Crespo

Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

areyesc@uci.cu

Agradecimientos

Durante nuestro paso por la universidad, conocemos a muchas personas. Algunas de ellas se quedan en el camino, otras continúan juntas hasta el momento de la graduación. Algunas se convierten en compañeros, otras se convierten en amigos, en familia. Por mi paso por la universidad me acordare de muchas personas que considero compañeros pero nunca podré olvidarme de las más especiales, mi piquete mi grupo mis amigos mi familia en la universidad, ellos son Liset, mi negra bella, mi sacadora de paso Yohana, por fin el novio para cuándo es ?, el más loco de todo el piquete Ernesto, el profe Andry, y por supuesto no puede faltar mi eterna gorda ese eres tu Romy, a todos ustedes gracias por dejarme pertenecer al mejor piquete de la UCI, por supuesto tampoco puedo dejar de mencionar a Rebeca, Yeilen, Sergio, Jorge, Karla, en fin a todo el grupo 6105, a todos ustedes muchísimas gracias por todo.

Tampoco puedo dejar de mencionar a los amigos que hice durante 5 años de carrera, Yuneiry, Leodan, Alberto, Didzan, Leydanis, Audrey, Frank, Yenier, Pedro, mis profes Salvador y Marisel, en fin espero que nadie se ponga bravo si no mencione su nombre pero son muchos y no me alcanzaría el tiempo para nombrarlos a todos, a todos esos que saben los considero amigos muchísimas gracias por estar ahí.

Le agradezco infinitamente, por todo su apoyo, por todos sus concejos, por estar ahí siempre en las buenas y en las malas, a mi hermano Yoanys, gracias pipo por todo lo que has hecho por mí.

No puedo dejar de mencionar por supuesto a mi familia del eléctrico, si porque eso es lo que ya son, parte de mi familia, a la loca de Mariela con la que nunca dejaré de reírme y pasarla bien, a mi querida Yaima que a pesar de que nos conocemos hace menos de un año, sabes que te quiero cantidad, por supuesto que aunque no esté presente no puede faltar el geri, que me dio bastantes concejos de la vida y nada siempre son válidos, y por último y no menos importante a la rubia esa que me acogió como a un hijo y me acompaño hasta el día que discutí mi tesis, a todos ustedes muchísimas gracias.

Por supuesto y no pueden faltar mis agradecimientos a una persona que me aconsejó como a un hermano, que me ayudo a superar cada uno de los obstáculos durante dos años de mi carrera, que me acogió en su familia como a un miembro más a ti David muchísimas gracias.

Y no creas que me he olvidado de ti mi negra linda, mi bella mimi, tu que me acompañaste, me distes ánimo, me aconsejaste en los momentos más difíciles, me apoyaste en cada una de mis decisiones, me distes calor en las noches frías, a ti mi amor muchísimas gracias.

A mi tutora que me acompaño durante el desarrollo de mi tesis, a usted profe Adisley muchísimas gracias.

Siempre dicen que lo que más recuerda alguien cuando lee algo es el principio y el final, tal vez por eso los deje a ustedes para el final porque quiero que se acuerde no solo yo, sino todo el que me está escuchando, de lo agradecido que estoy de tener por tener unos padres como ustedes, que me han apoyado siempre en todas mis decisiones, que no siempre fueron las mejores, pero estaban ahí dispuestos a sacrificar lo que fuera necesario para apoyarme.

Pepe, que es mi papa, no sabes lo orgulloso que estoy de tener un padre como tú, gracias por estar ahí para mí, por aconsejarme en todos los momentos, por estar ahí en todas las etapas de mi vida.

Mami, gracias por ser como eres la mejor madre de todas, por regañarme en los momentos que tocaba, pero gracias a eso soy quien soy hoy en día, une mejor persona, preparada, por guiarme por los caminos donde mejor me podía ir, por quererme tanto, por mimarme tanto, por estar ahí cada vez que necesite mi apoyo, gracias mami.

A mi hermana que ha estado conmigo en cada momento, cuando obtuve 5 en una prueba, como cuando cogí 2 en otras, por estar cuando la necesite, a ti y al amor de mi vida, mi sobrino, gracias por ser el motor impulsor para ser alguien en la vida, para ser lo que soy un ingeniero.

A toda mi familia que me acompaño durante todo el transcurso de mi carrera y que puedo decir con orgullo que se sienten orgullosos de ver lo que he logrado, a toda mi familia, muchas gracias.

Dedicatoria

Dedico este trabajo a dos personas muy especiales en mi vida que aunque no se encuentran entre nosotros fueron un ejemplo a seguir para poder graduarme como ingeniero, a mi abuela Ule por ser la abuela más cariñosa del mundo y a mi tío Buro, que siempre me apoyó durante mi carrera y me guió en todo mi etapa de estudiante.
A ustedes va dedicado este trabajo.

Resumen

SoftRen 2.0 es un sistema que permite realizar pruebas de carga y estrés, utilizando una interfaz propia con todas las funcionalidades necesarias para realizar este tipo de pruebas. Permite al probador la funcionalidad de listar los enlaces de una url definida. Utilizando la Plataforma de Cálculo Distribuido permite realizar simulaciones de gran carga en el servidor o aplicación a probar. La misma disminuye la carga de trabajo en las estaciones de prueba y aumenta la concurrencia en los servidores a probar, ya que en lugar se simular usuarios mediante hilos desde una misma estación de trabajo, distribuye las peticiones entre los clientes de la plataforma, siendo estos los que se conectan al servidor donde se encuentra el sistema (base datos, web) que se desea verificar, lográndose que en las estaciones de trabajo con pocas prestaciones en cuanto al hardware, como los que posee la Universidad, se puedan realizar y concluir las pruebas satisfactoriamente.

Palabras clave

Plataforma de Tareas Distribuida, Pruebas de calidad, Pruebas de rendimiento, T-arenal, pruebas de carga y estrés.

Índice General

Índice de Tablas

Índice de Figuras

Introducción

En un mundo globalizado, en donde las empresas se ven enfrentadas a la competencia a nivel mundial, la calidad de los productos se convierte en un punto diferenciador a tener en cuenta para aumentar la satisfacción general del cliente, los productos o servicios que ostentan certificados de calidad son preferidos por los compradores, porque trasmiten seguridad y confianza. Esto también constituye un atributo de valor para las estrategias de comercialización en el exterior. Si bien la industria del software es nueva, ha tenido que madurar rápidamente, tal como lo exigen los avances tecnológicos y su alta participación al interior de las empresas. Esta industria comparte con las demás el interés por la calidad y la competitividad.

Las pruebas de software, se consideran uno de los procesos más importantes en el diseño y construcción de aplicaciones, ya que constituyen las herramientas que la ingeniería de software emplea para asegurar la calidad de los productos. Estas han evolucionado en el transcurso de los años, no solo permiten depurar y detectar los errores existentes en las aplicaciones, sino que han llegado a convertirse en un proceso integrador, automatizable y administrable que se realiza desde el inicio del desarrollo del software.

Es muy común que en las empresas de desarrollo de software existan equipos de trabajo dedicados a probar las funcionalidades de sus desarrollos, mientras se dedica poco o ningún tiempo a comprobar si estos cumplen con los requisitos mínimos de rendimiento. Estas pruebas son, hoy en día, cada vez más necesarias pues 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.

En la implementación de sistemas informáticos se pueden encontrar diferentes tipos de arquitecturas, como el cliente-servidor, donde un cliente es una estación que realiza peticiones, y el servidor, es el encargado de darle respuesta a dichas peticiones. Las pruebas de rendimiento en este tipo arquitectura es fundamental ya que permiten verificar si la capacidad del sistema es adecuada para la demanda de trabajo que soportará y detectar posibles puntos de ruptura (cuellos de botella) e ineficiencias, proporcionando la información necesaria para un correcto dimensionado.

Cuba no ha estado exenta de este desarrollo tecnológico y en los últimos años ha logrado insertarse tanto en el mercado nacional como en el internacional. Esto implica que cada día las empresas cubanas destinadas al desarrollo de software, tengan como reto brindar soluciones eficientes y con una alta calidad, por lo cual han tenido que perfeccionar los métodos y herramientas para asegurar que los productos que en toda la isla se desarrollan, sean liberados con la garantía de calidad necesaria.

Dentro de las empresas productoras de software en Cuba se encuentran: la Empresa Cubana Nacional de Software (*DeSoft*), la Empresa Cubana Productora de Software para la Técnica Electrónica (*Softel*), la Universidad de las Ciencias Informáticas (*UCI*) entre otras. Esta última cuenta con el Centro de Calidad para Soluciones Informáticas (*CALISOFT*), perteneciente al Ministerio de la Informática y las Comunicaciones (*MIC*). Dicho centro es el encargado de certificar la calidad de los productos que se desarrollan en la UCI, además de facilitar la implementación de mejores prácticas en el proceso de desarrollo y mantenimiento de un software. Para asegurar que el producto está libre de errores y que funciona correctamente según las especificaciones del cliente, CALISOFT cuenta con un Laboratorio Industrial de Pruebas (*LIPS*), que es el encargado de aplicar todos los procedimientos establecidos para verificar y validar la calidad del software que se presenta para su revisión.

En la actualidad existen diferentes tipos de sistemas y herramientas que permiten simular los entornos donde será implantado el software. Ejemplos de estos sistemas son JCrawler, Solex, Qaload y JMeter. El LIPS utiliza el JMeter, herramienta que permite realizar simulaciones de gran carga en el servidor, red, o aplicación para determinar el rendimiento ante diferentes tipos de sobrecarga. Este basa su funcionamiento en la simulación de usuarios conectados a la aplicación a probar, mediante el uso de hilos de ejecución de forma concurrente desde un mismo ordenador. Una de las desventajas que posee JMeter es que necesita estaciones de trabajos con altas prestaciones en cuanto al Hardware.

Las estaciones de trabajo con las que cuenta la Universidad poseen limitaciones dependiendo de sus características físicas, ejemplo de esto es que estas no poseen una memoria RAM alta, además en la UCI no se cuenta con servidores destinados a este tipo de pruebas. Debido a esto, cuando se realizan pruebas con la utilización de JMeter en ordenadores con estas limitaciones, provoca que la computadora disminuya su rendimiento o incluso puede llegar a colapsar, imposibilitando la conclusión satisfactoria de la prueba.

Para contrarrestar esta deficiencia que posee JMeter, en la Universidad, específicamente en el departamento de Bioinformática de la facultad 6, se desarrolló un sistema que utilizando las ventajas de la Plataforma de Cálculo Distribuido T-arenal (implementada en ese departamento) lograra realizar estas pruebas utilizando al máximo los recursos de computo que posee el LIPS. Esta herramienta no se encuentra en funcionamiento en estos momentos, ya que aún no cumple con todos los requisitos que el centro necesita para realizar las pruebas, ejemplo de esto es que no posee una interfaz propia, sino que utiliza la propia interfaz de T-arenal.

Otros factores que se deben tener en cuenta es que, en la actualidad, se realizan aplicaciones más complejas, que deben tener un rendimiento alto ante gran número de peticiones y con altos volúmenes de procesamiento de datos y además el LIPS recibe

muchas solicitudes de revisión, por lo que no se le pueden destinar todos los recursos a una sola aplicación.

Por todo lo antes expuesto se define como **problema de la investigación**:

¿Cómo potenciar la realización de las pruebas de carga y estrés que se realizan en la Universidad por el Laboratorio Industrial de Pruebas?

La presente investigación tiene como **objeto de estudio** el proceso de pruebas de calidad de software enmarcando el **campo de acción** en el proceso de pruebas de carga y estrés a servidores de aplicación.

Para dar solución al problema de la investigación se define como **objetivo general**: Desarrollar la versión 2.0 de SoftRen, que mediante la computación distribuida, logre disminuir la carga de trabajo en los servidores de prueba y aumente el nivel de concurrencia durante una prueba de carga y estrés en un servidor de aplicaciones.

Objetivos específicos:

- ✓ Construir el marco teórico sobre las principales herramientas que permiten realizar pruebas de carga y estrés.
- ✓ Definir los requisitos funcionales y no funcionales que tendrá la herramienta para realizar pruebas de carga y estrés a aplicaciones cliente-servidor.
- ✓ Diseñar el sistema analizado.
- ✓ Implementar el sistema diseñado.
- ✓ Realizar pruebas funcionales al sistema para validar su correcto funcionamiento.

El presente trabajo se desglosa en tres capítulos fundamentales:

Capítulo 1. Fundamentación teórica del sistema para realizar pruebas de carga y estrés a aplicaciones cliente servidor utilizando la Plataforma de Tareas Distribuidas T-arenal, en este capítulo se realiza un estudio sobre la calidad del software y sus conceptos más significativos, los tipos de pruebas que se realizan en los sistemas informáticos y los sistemas que se utilizan para medir el rendimiento de una aplicación cliente servidor. Se explican además, las tecnologías y herramientas utilizadas para dar solución al problema planteado.

Capítulo 2. Análisis y diseño del sistema para realizar pruebas de carga y estrés a aplicaciones cliente servidor utilizando la Plataforma de Tareas Distribuidas T-arenal, en este capítulo se definen los requisitos funcionales y no funcionales de la herramienta, se presenta el diagrama de caso de uso del sistema, así como las descripciones textuales de los casos de uso para comprender mejor el funcionamiento del sistema. Se expone la descripción de los patrones usados, tanto de diseño como arquitectónicos, así como los diagramas de clases del diseño; vista de casos de uso, vista lógica y el diagrama de despliegue.

Capítulo 3.Implementación y validación del sistema para realizar pruebas de carga y estrés a aplicaciones cliente servidor utilizando la Plataforma de Tareas Distribuidas T-arenal, en este capítulo se describen los elementos del diseño en términos de componentes reflejados en el diagrama de componentes. Se describe el código fuente fundamental de la herramienta y posteriormente se realizan pruebas de validación para verificar el correcto funcionamiento de las funcionalidades del sistema.

Capítulo 1: Fundamentación teórica del sistema para realizar pruebas de carga y estrés a aplicaciones cliente servidor utilizando la Plataforma de Tareas Distribuidas T-arenal.

En este capítulo se realiza un estudio sobre la calidad del software y sus conceptos más significativos, los tipos de pruebas que se realizan en los sistemas informáticos y los sistemas que se utilizan para medir el rendimiento de una aplicación cliente servidor. Se explica además, las tecnologías y herramientas a utilizar para dar solución al problema planteado.

1.1 Calidad de software

En la actualidad uno de los problemas fundamentales en la esfera de la informática, es la calidad del software. La obtención de un software con calidad implica la utilización de metodologías o procedimientos a lo largo del desarrollo del mismo en vista a lograr un producto con eficiencia.

La calidad puede verse dentro de un proceso de producción de software como el conjunto de metodologías y herramientas de control aplicadas durante todas las fases de elaboración con el fin de lograr los objetivos establecidos inicialmente, identificar aquellos objetivos que no han sido conseguidos y establecer mecanismos de corrección para estos mismos.

La calidad debe hacerse extensible al cliente y al usuario de la aplicación y debe ser partícipe en ella de la misma forma que lo hace en la fase de definición de requisitos. El objetivo es involucrar a todas las partes participantes del proyecto con la misión de establecer la calidad no sólo como una herramienta de construcción sino como un sello de garantía de objetivos cumplidos y trabajo bien hecho.

Una de las primeras definiciones de calidad de software aseguraba que “la calidad de un programa o sistema se evaluaba de acuerdo al número de defectos por cada mil líneas de código.” (2)

La definición de la calidad del software según la IEEE, Std. 610-1990, es “el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario” (3).

Según Pressman, la calidad de software no es más que la concordancia con los requisitos funcionales y de rendimiento explícitamente establecida con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo producto desarrollado profesionalmente. (4)

Por lo que se puede concluir que la calidad de software es el desarrollo de software basado en estándares con la funcionalidad y rendimiento total que satisfacen los requerimientos del cliente.

1.2 Pruebas de software

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de pruebas, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra el comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador probando el comportamiento del mismo. (5)

Las pruebas de software: "son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas." (3)

Las "pruebas del software" son un eslabón fundamental en el proceso de desarrollo del software. Permiten evaluar el producto para conocer en qué condiciones se encuentra el mismo y de esta forma detectar errores que pudieran generar comportamientos erróneos durante su ejecución.

Por todo lo antes planteado las pruebas de software tienen entre sus objetivos principales:

- Detectar defectos en el software.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregidos antes de entregar el software al cliente.
- Diseñar casos de pruebas que sistemáticamente saquen a la luz las diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

1.2.1 Tipos de pruebas

En la etapa de prueba del software se crean una serie de Casos de Pruebas que intentan "destruir" la aplicación desarrollada. La prueba requiere que se descarten ideas preconcebidas sobre la "calidad o corrección" del software desarrollado. (16)

Un buen Caso de Pruebas es:

- ✓ Un proceso de ejecución de un programa con la intención de descubrir un error.
- ✓ Es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

- ✓ Tiene éxito si descubre un error no detectado hasta entonces.

Existen diferentes tipos de pruebas que se le realizan a los software para evaluar la capacidad de su funcionamiento en los diferentes entornos donde serán implantados, ejemplo de estas pruebas son las de funcionalidad, usabilidad, comunicación, volumen, etc. Otro tipo de prueba que se realiza son las de rendimiento que permiten determinar si los tiempos de respuesta tanto en condiciones normales como especiales, se encuentran dentro de los límites predefinidos.

1.2.1.1 Pruebas de rendimiento

Son pruebas dirigidas a evaluar la conformidad de un sistema o componente con requerimientos de desempeño específicos. Normalmente esto se lleva a cabo usando una herramienta de prueba automática que permite simular un gran número de usuarios, carga y volumen de información además de monitorear el desempeño del hardware, dentro de las pruebas de rendimiento se pueden encontrar las pruebas de resistencia (SOAK), las pruebas de picos (Spike testing), las pruebas de carga y las pruebas de estrés donde:

Entre los tipos de pruebas de rendimiento se encuentran:

- ✓ *Pruebas de carga:* Este es el tipo más sencillo de pruebas de rendimiento. Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Si la base de datos, el servidor de aplicaciones, también se monitorizan, entonces esta prueba puede mostrar el cuello de botella en la aplicación.
- ✓ *Pruebas de estrés:* Esta prueba se utiliza normalmente para romper la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

Las pruebas de carga y estrés en general permiten al probador determinar el punto de ruptura[1] de la aplicación, obteniendo así posibles errores de disponibilidad en estas. Dentro de las distintas ventajas que nos brindan las pruebas de carga y estrés

[1] Cantidad de Usuarios máximo que puede soportar la aplicación

es que posibilita obtener el tiempo de respuesta de una aplicación ante determinados números de usuarios conectados concurrentemente.

1.2.2 Tiempo de respuesta de las aplicaciones

El tiempo de respuesta de un sistema se puede definir como el tiempo definido por una solicitud del cliente y la respuesta final del servidor. En una aplicación web el tiempo de respuesta se define por el tiempo de respuesta de la red (N1+N2+N3+N4) y el tiempo de la aplicación (A1+A2+A3) (ver Figura 1). El tiempo de respuesta es la clave del rendimiento del software. (6) Para obtener este rendimiento no existe una fórmula que le permita asegurar a un probador el rendimiento del sistema ante diferentes tipos de sobre cargas.

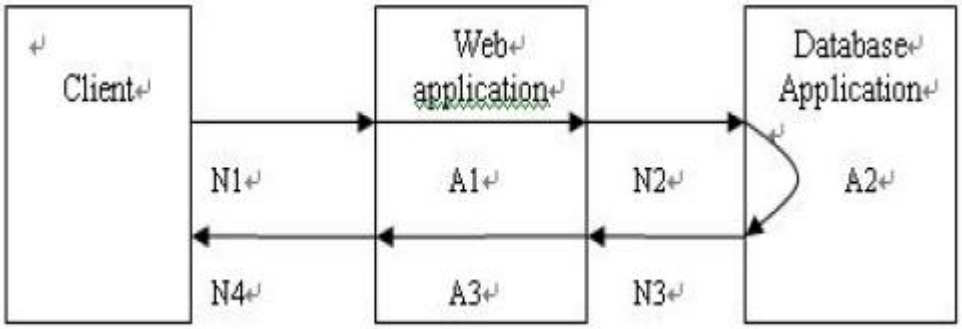


Figura 1: Tiempo de respuesta de las páginas de una aplicación web

En las pruebas prácticas, la ingeniería presta más atención a la concurrencia de los usuarios en el negocio, es decir cuántos usuarios estarán concurrentemente disponibles en el sistema. En la fórmula (1) N es la cantidad de usuarios que utilizan la aplicación, L la duración de empleo de la aplicación y T es el tiempo de inspección.

$$C = N * L / T \quad (1)$$

Para obtener un rendimiento de una aplicación se propone el estudio del siguiente caso de estudio:

En una empresa se posee un sistema informático y se desea calcular el rendimiento de dicha aplicación, específicamente en la sección de inicio de sección. Se sabe que en la empresa existen aproximadamente 3000 usuarios y que conectados concurrentemente lo visitan 400, utilizándolo 4 horas de las 8 que se trabajan en el día.

Si el inicio de sección está condicionado en una distribución de Poisson Cp proporciona los usuarios concurrentes en el sistema.

$$Cp = C + 3 * \sqrt{C} \quad (2)$$

Utilizando estas dos fórmulas se obtiene que el número de usuarios conectados concurrentemente en el sistema probabilísticamente sea:

$$C = 400 * 4 /8 = 200$$

$$Cp = 200 + 3 * \sqrt{200} = 242 \quad (6)$$

En la fórmula 3 se puede obtener el rendimiento de la aplicación (F) como cantidad de usuarios virtuales (Nuv) * golpes enviados por el usuario[2] (R) / tiempo de respuesta del sistema (T).

F = Nuv * R / T (3) (6)

Como bien se ha expresado antes no se posee una fórmula específica para calcular el rendimiento de las aplicaciones, por lo cual se determina emplear esta fórmula para el cálculo. Esta se selecciona debido a que es la forma más probable de obtener el rendimiento cuantitativo de los sistemas que se deseen probar.

Llevando esta fórmula a la situación problemática que se posee se obtiene:

Rend = Client_utiliz * Cant_conex / Tiemp_resp

Esta fórmula se puede interpretar de varias maneras siempre teniendo en cuanta el tiempo de respuesta de la aplicación. El resultado obtenido de esta fórmula esta expresado en transacciones por segundo, ejemplo de esto es el siguiente caso:

Una prueba realizada a la aplicación[3] arrojó los siguientes resultados:

Tabla 1: Datos de la prueba realizada

Usuarios a simular	Tiempo de respuesta	Rendimiento Obtenido
200	0,459 seg	0,434829 trans/seg
300	0,24627 seg	1,2181 trans/seg

En la tabla se muestra él comporta miento de la aplicación ante diferentes tipos de cargas, obteniéndose que para menos de un segundo se pudieron simular 200 usuarios satisfactoriamente, obteniéndose un rendimiento de 0,4 transacciones por segundo, lo que significa que el rendimiento de la aplicación es *normal.*

Sin embargo para un total de 300 usuarios el tiempo de respuesta fue de 0,2 segundos y el rendimiento obtenido fue de 1,2 transacciones por segundo, esto muestra que la segunda prueba aunque se ejecutó con más usuarios a simular que la primera, el rendimiento fue mucho *mayor*.

En el cálculo del rendimiento existen también varios factores a tener en cuenta como por ejemplo:

[2] Cantidad de conexiones establecidas

[3] 10.54.18.5/publica

- ✓ La velocidad de la red: si la velocidad de conexión es baja entonces el tiempo de respuesta de la aplicación es mucho mayor por lo que el rendimiento debe de ser bajo.
- ✓ La disponibilidad de clientes en T-arenal: si en el momento de la prueba T-arenal tiene pocos clientes disponibles entonces el tiempo de respuesta seria mayor y el rendimiento disminuiría.
- ✓ Los errores obtenidos: si en el transcurso de la prueba se detectan errores entonces ya el rendimiento por muy alto que sea no es válido debido a que la aplicación presenta errores a la hora de su trabajar cierta cantidad de usuarios con ella.

De este caso de estudio se puede concluir que si el tiempo de respuesta de la aplicación es alto entonces se obtendría un bajo rendimiento de la misma.

1.3 Herramientas que se utilizan para pruebas de rendimiento.

Existen diferentes tipos de herramientas que nos permiten medir el rendimiento de un sistema. A continuación se describen algunas de estas herramientas:

1.3.1 JMeter

JMeter es una herramienta de código abierto realizada en Java que se utiliza para realizar pruebas de rendimiento, normalmente contra aplicaciones web.

Permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar y analizar el rendimiento ante diferentes tipos de sobrecarga. Dentro de sus principales características se pueden encontrar:

- ✓ Es un sistema de código abierto, con licencia Freeware-Opensource.
- ✓ Es multiplataforma.
- ✓ Permite realizar pruebas de rendimiento a servidores de base de datos, LDAP y ftp.
- ✓ Multihilo: permite realizar testeos de forma concurrente. (7)

1.3.2 JCrawler

Aplicación de código abierto para realizar test de estrés a aplicaciones web. Se introduce la URL y se puede realizar la navegación web. Esta herramienta es apropiada para portales complejos, ya que su funcionamiento está basado en probar todas las páginas del portal y no solo algunas Urls.

JCrawler está basado en ataques/segundo y no en “X” hilos atacando una web (ya que en estos últimos casos puede que realmente se estén dando por ejemplo solo 2 ataques por segundo aun teniendo 200 hilos.

Entre sus características más significativa se tienen:

- **Fácil de configurar**. Toda la configuración se reduce a un fichero de configuración XML.
- **Independiente** del Sistema Operativo.
- **Código Abierto**: ofrece confianza, ya que en cuanto se detecte un error, será solucionado en un muy breve período de tiempo.

1.3.3 Solex

Solex es una herramienta de prueba de aplicaciones web de código abierto, fue construido como un plugins para el IDE de Eclipse. Proporciona funciones para grabar una sesión de cliente, de acuerdo a diversos parámetros y reproducirlo más tarde por lo general con el fin de garantizar la no regresión del comportamiento de la aplicación. Otra de las funcionalidades que brinda es que permite realizar pruebas de rendimiento y estrés contra la aplicación web. (8)

1.3.4 VPerformer

VPerformer es un producto de pruebas de rendimiento y de carga, que puede ser usado para evaluar el rendimiento y escalabilidad de las aplicaciones web. Permite evaluar cómo responden las aplicaciones web cuando son accedidas concurrentemente por un gran número de usuarios. Puede medir las características de rendimiento de la aplicación, grabando y repitiendo scripts (archivos de secuencia de instrucciones) automatizados que simulan una gran cantidad de usuarios virtuales concurrentes.

Además de mostrarte las características de rendimiento de tu aplicación, vPerformer permite monitorear el estado de cualquier componente externo que influye en el comportamiento de la aplicación. Estos componentes incluyen sistemas operativos, servidores web, servidores de aplicación, servidores de base de datos, dispositivos de red, e implementaciones SNMP de diversos distribuidores. Esto permite identificar cuellos de botellas en cuanto a rendimiento y determinar el origen del problema.

Beneficios de vPerformer

- Una sencilla interfaz de usuario gráfica que puede registrar interacciones web basadas en operaciones de "señalar y hacer clic".
- No requiere una programación de fondo. Para usuarios que desean programas sus scripts, vPerformer utiliza el lenguaje básico de la industria JavaScript.
- Es capaz de simular un gran número de usuarios virtuales mientras ubica su insignificante carga en el hardware subyacente.
- Desarrolla scripts reutilizables, parametrizados, y con manejo de datos.

- Se pueden utilizar múltiples máquinas como agentes repetidores a través de un único punto de control.
- Ingenieros de quality assurance (garantía de calidad) pueden probar fácilmente el rendimiento y escalabilidad de las aplicaciones web antes del despliegue

1.3.5 SoftRen 1.0

SoftRen es una herramienta que permite realizar pruebas de carga y estrés a aplicaciones web y base de datos. Esta herramienta utilizando la computación distribuida, permite disminuir la carga de trabajo en los servidores de prueba y aumenta el nivel de concurrencia durante una prueba de rendimiento en un servidor de aplicaciones. Para su funcionamiento SoftRen se basa en la creación de hilos y realiza pruebas concurrentemente.

Ventajas de SoftRen:

- Utilizando los sistemas distribuidos, las tareas (realizar pruebas de carga y estrés) se distribuyen a varios clientes (PC que realizan pruebas) y así disminuye la carga de trabajo de cada una de las maquinas que realizan estas pruebas.
- Es un sistema multiplataforma.
- Posee una sencilla interfaz de usuario.
- Los ingenieros en calidad pueden probar fácilmente el rendimiento de una aplicación web o una base de datos.

Para la realización de pruebas de carga y estrés las herramientas antes mencionadas son factibles, pero tienen como principales desventajas:

- VPerformer posee licencia propietaria.
- JCrawler se limita a aplicaciones web solamente.
- JMeter necesita estaciones de trabajo con altas prestaciones en cuanto al hardware.
- SoftRen 1.0 es una herramienta que aún no posee todas las funcionalidades que el LIPS necesita para realizar las pruebas de carga y estrés como por ejemplo:
  - ✓ Utiliza la interfaz de T-arenal para la realización de la prueba.
  - ✓ No brinda un reporte de la prueba que se realizó a los probadores.
  - ✓ La graficación de la prueba no es la que se desea en el LIPS ya que no posee ningún valor agregado.
  - ✓ No le permite al probador la selección de los enlaces que desea probar y se limita solamente al enlace que le sea definido en los parámetros.

Por todo lo antes expresado es que surge la necesidad de desarrollar la versión 2.0 de SoftRen, que permita realizar todas las funcionalidades a través de una interfaz propia, que permita dado un enlace listar todas las url contenidas en el mismo, que permita generar un reporte con los resultados de la prueba, que permita descargar la solución de la Plataforma de Cálculo Distribuido y que mediante la computación distribuida, logre disminuir la carga de trabajo en los servidores de prueba y aumente el nivel de concurrencia durante una prueba de carga y estrés en un servidor de aplicaciones.

1.4 Sistemas Distribuidos

Los sistemas distribuidos son sistemas cuyos componentes (hardware y software) están en ordenadores conectados en red, estos se comunican y coordinan sus acciones mediante el paso de mensajes para lograr un objetivo.

1.4.1 Ventajas de los sistemas distribuidos:

- ✓ Abaratamiento de red y maquinas disponibles
- ✓ Compartición de recursos
- ✓ Los que permite el trabajo cooperativo
- ✓ Escalabilidad
- ✓ Tolerancia a fallos

1.4.2 Desventajas de los sistemas distribuidos:

- ✓ Programación compleja
- ✓ Inexistencia de reloj global(en ocasiones)
- ✓ Fallos independientes: el sistema puede ser más robusto, pero aparecen nuevos modos de fallos
- ✓ Inseguridad(pero también al revés)
- ✓ Redes conectadas mediante IP
- ✓ Comunicación mediante paso de mensajes

1.4.3 Aspectos fundamentales de un sistema de cómputo:

- ✓ ***Transparencia:*** Estos sistemas deben ocultar la naturaleza distribuida permitiendo que los usuarios interactúen con una aplicación de Escritorio y trabajen como si se tratara de una sola PC. Los programadores o desarrolladores no tienen que encargarse de repartir el trabajo entre los nodos del sistema, solamente tienen que especificar cómo dividir el problema y cómo integrar las soluciones parciales. El trabajo “sucio” de distribuir físicamente las

unidades más pequeñas del problema y llevar el control de que estas se realicen queda a cargo del sistema.

- ✓ ***Eficiencia:*** La solución a los problemas se obtiene más rápidamente haciendo uso de la Plataforma respecto a la respuesta que daría una computadora.
- ✓ ***Flexibilidad:*** Un proyecto debe estar abierto a modificaciones que mejoren su funcionamiento. Los sistemas tienen que ser lo suficientemente flexible para que cualquier cambio a realizar no requiera la parada de todo el sistema y la recompilación de todo el código. Si se realizan cambios en el módulo del cliente, cambiarlo es tan simple como colocarlo en el servidor y a medida que cada elemento de cómputo realiza peticiones de trabajo, se le envía la nueva versión para su actualización automática.
- ✓ ***Escalabilidad:*** El tamaño de una red varía en dependencia de la institución que la mantenga, el sistema debe de comportarse estable tanto en redes pequeñas como en grandes. También debe garantizar un adecuado mantenimiento y actualización, empleando el mínimo de personal. Ampliar el sistema en cuanto a unidades de cómputos, es tan sencillo como instalar el módulo cliente en una PC y configurarle la dirección física del servidor al cual debe reportarse.
- ✓ ***Fiabilidad:*** Deben asegura al 100% que el problema del usuario será resuelto independientemente de los problemas ajenos que existan, ya sean fallos de red, electricidad, apagado de máquinas, entre otros. Si un elemento de cómputo se desconecta la unidad en la que estaba trabajando no se pierde, ya que se almacena una copia de la misma en el servidor y luego se envía a otro cliente pasado un tiempo. (9)

1.4.4 Plataforma de Cálculo Distribuido T-arenal

La Plataforma de Tareas Distribuidas (T-arenal) es un producto informático que ofrece una alternativa de cómputo y que aglutina en un solo conjunto varias estaciones de trabajo sin intentar eliminar o pretender aminorar las amplias posibilidades de aplicación de los modelos paralelos, sino de complementar todos los medios disponibles en una gran “supercomputadora virtual”.

T-arenal v2.0 fue implementada con una arquitectura de varios servidores para una mejor configuración y uso más equitativo de los clientes. Esto es posible debido a que los elementos de cómputo estarán distribuidos por diferentes servidores de acuerdo a las prestaciones que cada uno tenga. La plataforma será vista por el usuario final

como un solo ordenador y todos los servidores serán gestionados a través de un servidor central.

La Plataforma de Cálculo Distribuido T-arenal se encuentra dividida en tres partes fundamentales:

Servidor Central: Su principal función es la de chequear y planificar la asignación de las diferentes tareas a los servidores de peticiones.

Servidor de Peticiones: Tienen asignado uno o varios clientes para realizar una ejecución determinada. Es el responsable de solicitar una tarea al servidor central, así como atender y controlar la realización de la misma. La tarea a ser atendida, será dividida en pequeñas subtareas denominadas unidades de trabajo. Este reparte estas unidades por cada uno de los clientes que tiene asignado, luego procesa el resultado de cada una de las subtareas generadas, para finalmente construir el resultado de la tarea original.

Cliente: Es el que realiza una solicitud de una unidad de trabajo al servidor de peticiones correspondiente. Realiza el procesamiento de la unidad de trabajo y posteriormente retorna el resultado obtenido, y así sucesivamente. Múltiples clientes pueden realizar peticiones de trabajo al servidor de peticiones. (9)

T-arenal es un sistema que es extensible, lo que posibilita que con solo redefinir sus dos clases principales, DataManager, (que es la que se ejecuta en el servidor de peticiones) y la clase Task, (que es la que se ejecuta en los clientes), se puede realizar cualquier problema, utilizando y especificando siempre las librerías o archivos necesarios para realizar la tarea.

Se determinó utilizar T-arenal como plataforma de cálculo distribuido ya que es desarrollada en la universidad, posee un grupo de trabajo y está desplegado en la misma, además permite aprovechar al máximo los recursos disponibles en la universidad.

1.5 Patrones

Un patrón se define como una solución probada con éxito que aparece una y otra vez ante determinado tipo de problema en un contexto dado. Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación. Este define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones. (10)

1.5.1 Patrones de diseño

Un patrón de diseño constituye una solución estándar para un problema común de programación en el desarrollo del software. Además es una técnica muy eficaz para flexibilizar el código haciéndolo satisfacer ciertos criterios.

1.5.1.1 Patrones GRASP

Representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones.

- ✓ Experto: Se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad
- ✓ Creador: es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.
- ✓ Controlador: Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.
- ✓ Bajo acoplamiento: Este patrón es el encargado de asignar una responsabilidad para conservar bajo acoplamiento.
- ✓ Alta cohesión: Asigna una responsabilidad de forma tal que la cohesión siga siendo alta.

1.5.1.2 Patrones GOF

Se clasifican en 3 categorías basadas en su propósito.

- ✓ Creacionales: abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.
- ✓ Estructurales: se ocupan de como las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- ✓ Comportamiento: están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.

1.5.2 Patrones arquitectónicos

Los patrones de arquitectura de software son patrones de diseño de software que constituyen una vía en la solución de problemas de arquitectura de software. Los mismos poseen un nivel de abstracción mucho mayor que los patrones de diseño. Además brindan una descripción de los elementos y el tipo de relación que tienen, así como las restricciones a para su uso.

1.5.2.1 Patrón cliente-servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios (servidores), y los demandantes (clientes).

Un cliente realiza peticiones a otro programa, el servidor, es el encargado de darle respuesta a dichas peticiones. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La utilización de T-arenal condiciona el uso del patrón ya que para integrarse a ella la herramienta tiene que tener el mismo patrón.

1.5.2.1.1 Ventajas de la arquitectura cliente-servidor

Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos.

Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

Por todo lo antes expuesto, y porque T-arenal cuenta con esta misma arquitectura, es que se decide utilizar el cliente servidor como arquitectura de SoftRen.

1.6 Metodología, tecnologías y herramientas

Para la realización de una aplicación de software siempre se debe definir los tipos de tecnologías a utilizar así como las herramientas y metodologías que serán de mayor utilidad para su implementación. A continuación se explica en detalle cada una de las tecnologías que fueron seleccionadas para llevar a cabo la implementación y

documentación de la aplicación que se desarrollará siguiendo los estándares empleados en la plataforma.

1.6.1 Metodología de desarrollo de software: OpenUp/Basic

OpenUp es un marco de trabajo para procesos de desarrollo de software. Fue liberado por el Eclipse Process Framework, construido sobre una donación realizada por IBM del Basic Unified Process. Fue entregada a Eclipse en 2005 y renombrada como OpenUp en 2006.

Como principales características posee que se basa en un desarrollo iterativo incremental, dirigidos por casos de uso y centrado en la arquitectura. Es una metodología para proyectos pequeños.

1.6.2 Herramienta para la modelación visual del sistema: Visual Paradigm 8.0

Para la modelación visual del sistema se emplea Visual Paradigm para UML en su versión 6.4. Es una herramienta Case profesional que utiliza “UML”, ayudando a construir aplicaciones rápidamente, mejor y económicamente, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue, además de ser una herramienta fácil de usar e instalar.

Permite modelar todos los tipos de diagramas de clases, código inverso, generar código fuente desde diagramas y documentación en HTML/PDF. En el mismo se pueden modelar los prototipos de interfaz de usuarios, dando al desarrollador una vista de lo que será el sistema. (11)

1.6.3 Lenguaje de Modelado: UML 2.1

El lenguaje unificado de modelado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación y esquemas de bases de datos. Su principal objetivo es representar el conocimiento acerca de los sistemas que se pretenden construir y las decisiones tomadas durante su desarrollo.

1.6.4 Entorno de Desarrollo Integrado: NetBeans 7.1

NetBeans IDE es un entorno de desarrollo integrado (IDE), modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. El proyecto

NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación.

Provee varias características y mejoras, tales como características de edición JavaScript, soporte para usar estructuras Spring de soporte web y una mejor manera de compartir librerías entre proyectos dependientes. El aclamado soporte para Ruby/JRuby ha sido mejorado con un nuevo editor de soluciones rápidas (Quick Fix), un administrador para la plataforma Ruby, soporte para depuración rápida en JRuby y otras características y soluciones.

1.6.5 Tecnología utilizada para la programación: Java 1.6

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principio de los años 90. Esta empresa lo describe como “simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico”.

Además de estas características el lenguaje de programación java constituye un lenguaje “simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico”. (13)

Se define este lenguaje ya que se usó para implementar la plataforma T-arenal, donde dicha plataforma brinda una librería que facilita el trabajo con la misma.

1.7 Conclusiones del capítulo

Luego de realizado el estudio de los principales factores que hacen posible el desarrollo del software, se obtuvo un mejor dominio del problema. Al analizar las distintas desventajas que posee JMeter y SoftRen 1.0 se determina que los mismos no cumplen con las necesidades del LIPS, por lo que se deriva la necesidad de realizar el análisis, diseño e implementación de la versión 2.0 de SoftRen que solucione los problemas detectados en la versión 1.0 del mismo y que aprovechando al máximo las estaciones de trabajo de la Universidad de las Ciencias Informáticas, permitan realizar pruebas de carga y estrés a servidores de aplicación (web o base de datos). Se seleccionaron como herramientas de desarrollo: OpenUp/Basic como metodología a seguir, Visual Paradigm en su versión 8.0 como herramienta CASE, NetBeans 7.1 como IDE de desarrollo, como lenguaje de programación java en su versión 1.6 y como Plataforma de Tareas Distribuidas T-arenal 2.0.

Capítulo 2: Análisis y diseño del sistema para realizar pruebas de carga y estrés a aplicaciones cliente servidor utilizando la Plataforma de Tareas Distribuidas T-arenal.

En este capítulo se definen los requisitos funcionales y no funcionales de la herramienta, se presenta el diagrama de caso de uso del sistema, así como las descripciones textuales de los casos de uso para comprender mejor el funcionamiento del sistema. Se expone la descripción de los patrones usados, tanto de diseño como arquitectónicos, así como los diagramas de clases del diseño; vista de casos de uso, vista lógica y el diagrama de despliegue.

2.1 Propuesta del sistema

En la actualidad en el LIPS se realizan pruebas de rendimiento al software enfocadas en carga y estrés. Para la realización de estas pruebas, se utiliza la herramienta JMeter. La aplicación de estos tipos de pruebas utilizando la herramienta se hace desde un único ordenador generando conexiones en hilos. La creación de hilos de ejecución consume recursos de la computadora, como la memoria RAM, de ahí que físicamente exista un límite máximo de hilos y conexiones posibles a crear en un determinado intervalo de tiempo.

Por todo lo antes planteado, se define la propuesta de una herramienta que permita obtener el rendimiento cuantitativo de las aplicaciones a las cuales se les apliquen pruebas de carga y estrés y que logre además aumentar el nivel de concurrencia en el servidor para llegar más rápido al punto de ruptura de la aplicación a probar, aprovechando al máximo los recursos de cómputo que posee la universidad. En esta propuesta se utilizarán varias estaciones de trabajo generando hilos de ejecución sobre el servidor. En lugar de ser X hilos de ejecución en un mismo ordenador cada cierto tiempo, contar con más estaciones de trabajo propiciaría que se pueda distribuir esta carga de peticiones entre ellas. De esta forma la prueba podría realizarse en menos tiempo ya que sería distribuida y se llegaría más rápido al punto de ruptura de la aplicación a probar. ¿De qué forma sería posible esto? Utilizando la Plataforma de Tareas Distribuidas T-arenal y aprovechando su ambiente distribuido explicado anteriormente. (17)

2.1.1 Ejemplo de la solución propuesta

Se desea realizar una prueba de carga y estrés a una aplicación web, el LIPS cuenta con 1 servidor central de T-arenal, 1 servidor de peticiones y 3 clientes (estaciones de trabajo). El probador a través de SoftRen especifica los datos de la prueba que se desea realizar (simular 300 usuarios conectados al servidor). Al ejecutarse esta tarea se almacena la prueba en el servidor central de T-arenal. Esta tarea es enviada al servidor de peticiones, que es el encargado de dividir el problema (prueba) en varias subtareas y distribuye estos problemas en los 3 clientes.

Cada cliente ejecuta su tarea y cuando concluye envía los resultados (tiempo de respuesta, cantidad de errores, errores obtenidos) al servidor de peticiones que es el encargado de construir el resultado final (que obtiene luego de que todos los clientes culminen la ejecución) y se lo envía a el servidor central que a través de la conexión establecida le permite a el probador conocer el resultado de la prueba.

Un ejemplo de esta solución es la siguiente:

Se desea realizar la prueba simulando 300 usuarios y el servidor de peticiones posee 3 clientes pidiendo tareas esta tarea se divida en 3 pequeños problemas (100*3 = 300), estas ejecuciones ya no se realizarían desde un mismo ordenador, por lo que se ganaría en tiempo de ejecución y aumentaría el nivel de concurrencia en el servidor a probar, además de que las estaciones de trabajo no sufrirán degradación en su sistema.

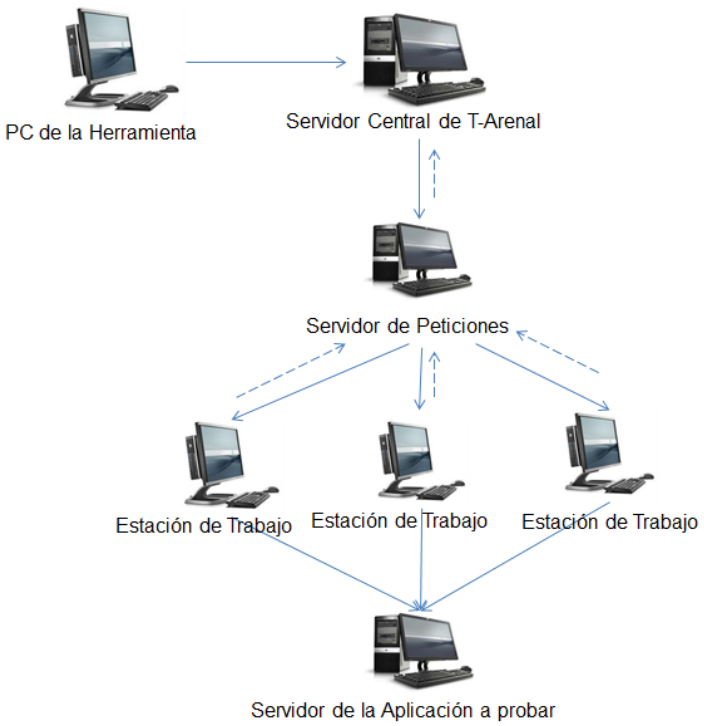


Figura 2: Herramienta (SoftRen) integrada con T-arenal

2.2 Especificación de los requisitos de software

El flujo de trabajo de requerimientos es uno de los más importantes, porque en él se establece qué es lo que tiene que hacer exactamente el sistema que se construya, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que se especifiquen. Se dividen en dos grupos: los requisitos funcionales y los requisitos no funcionales.

2.2.1 Requisitos funcionales

Los requisitos funcionales deben comprenderlo tanto los desarrolladores como los usuarios, ya que son las funcionalidades que el sistema debe cumplir para su funcionamiento.

- ✓ ***RF 1:*** Autenticar Usuario.
  El sistema debe permitir al usuario autenticarse.
- ✓ ***RF 2:*** Definir pruebas para aplicaciones Web.
  El sistema debe permitirle al probador definir la prueba para una aplicación web.
- ✓ ***RF 3:*** Definir pruebas para Base de Datos.
  El sistema debe permitirle al probador definir la prueba para una Base de Datos.
- ✓ ***RF 4:*** Ejecutar prueba.

El sistema debe permitirle al probador ejecutar la prueba definida.

- ✓ ***RF 5:*** Generar reporte

  El sistema debe permitirle al probador exportar los resultados de las pruebas generadas.

- ✓ ***RF 6:*** Graficar prueba

  El sistema debe permitirle al probador graficar los resultados de las pruebas generadas

- ✓ ***RF 7:*** Eliminar Pruebas.

  El sistema debe permitirle al probador eliminar una prueba.

- ✓ ***RF 8:*** Calcular Rendimiento.

  El sistema debe permitirle al probador obtener el rendimiento cuantitativo de una aplicación.

- ✓ ***RF 9:*** Descargar Solución

  El sistema debe permitirle al probador descargar la solución de la prueba realizada.

- ✓ **RF 10:** Modificar Prueba

  El sistema debe permitirle al probador modificar una prueba definida

2.2.2 Requisitos no funcionales

Una vez analizados los requisitos funcionales, se hace necesario analizar los requisitos no funcionales, que no son más que las propiedades o cualidades que el producto debe tener. Entre los requisitos no funcionales del sistema se encuentran:

Apariencia o interfaz externa: El sistema debe contar con una interfaz amigable, donde el usuario pueda orientarse fácilmente.

Usabilidad: La herramienta estará dirigida a usuarios con un nivel medio o superior de informática. La herramienta tendrá un ambiente sencillo y será fácil de manejar para el usuario.

Fiabilidad: La interacción con el sistema, estará sometida a un proceso de autenticación del usuario. La plataforma T-arenal guarda constantemente los cambios, garantizando que si hay fallos ya sean fallos de red, electricidad, apagado de máquinas, entre otros que puedan surgir; no ocurra nada con el seguimiento de la tarea.

Seguridad:

• Confidencialidad: Se requiere de usuario y contraseña para poder acceder a la información de las pruebas.

• Disponibilidad: En caso de tener el usuario y la contraseña, se le garantiza poder acceder a la herramienta en todo momento dependiendo del permiso que este tenga en la plataforma.

Portabilidad: El sistema es multiplataforma, razón por la cual podrá ser utilizado tanto en Windows como en Linux.

Software: Debe estar instalada la máquina virtual de Java SDK 1.5.x o superior. El servidor central de T-arenal, debe estar disponible, así como los clientes de prueba. Por consiguiente la plataforma T-arenal debe estar disponible conjuntamente con el servidor central.

Hardware: El ordenador donde se utilizará la herramienta debe tener como mínimo de memoria RAM 256 MB.

2.3 Actores del Sistema

Los actores representan a terceros fuera del sistema que interactúan con él. En el sistema que se describe se identifica el siguiente actor:

Tabla 2: Descripción de los actores del sistema

Actores	Descripción
Probador	Representa el usuario que define y ejecuta las pruebas.

2.4 Casos de usos del sistema

Los casos de uso del sistema que a continuación se presentan tienen como objetivo satisfacer los requisitos funcionales descritos con anterioridad.

- ✓ CU Autenticar usuario
- ✓ CU Definir prueba para una aplicación web
- ✓ CU Definir prueba para una base de datos
- ✓ CU Eliminar prueba
- ✓ CU Ejecutar prueba
- ✓ CU Obtener resultados
- ✓ CU Calcular rendimiento

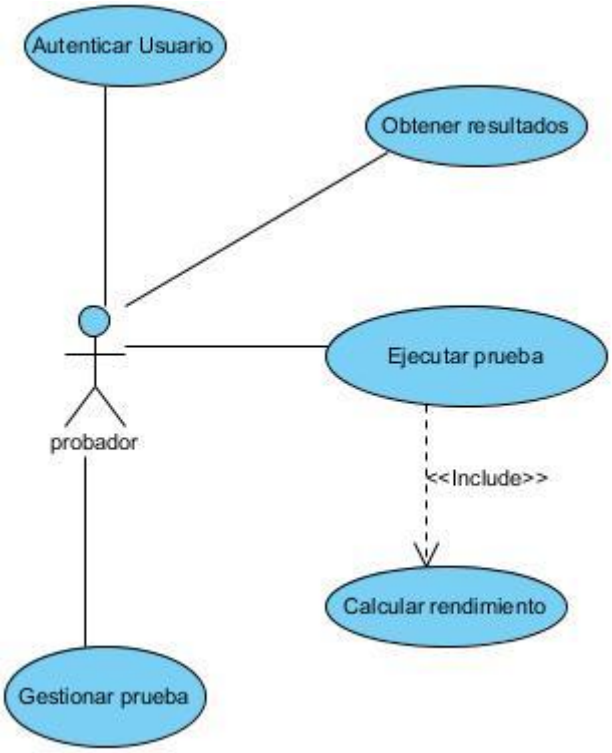


Figura 3: Diagrama de Casos de Usos del Sistema

2.5 Descripción de los Casos de Uso del Sistema

A continuación se realizan la descripción textual de los Casos de Uso del Sistema.

Tabla 3: Descripción Caso de Uso *Autenticar usuario*

Objetivo	Este caso de uso permite al usuario poder acceder al sistema.
Actores	Probador
Resumen	Se inicia cuando el actor ejecuta la aplicación y se requiere de usuario, contraseña, servidor y puerto para autenticarse. El caso de uso termina cuando el usuario accede a la aplicación una vez autenticado.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El usuario debe estar adicionado en la Plataforma de Cálculo Distribuido
Postcondiciones	El usuario se autentica en el sistema.
Flujo de eventos	
Flujo básico Autenticar Usuario	

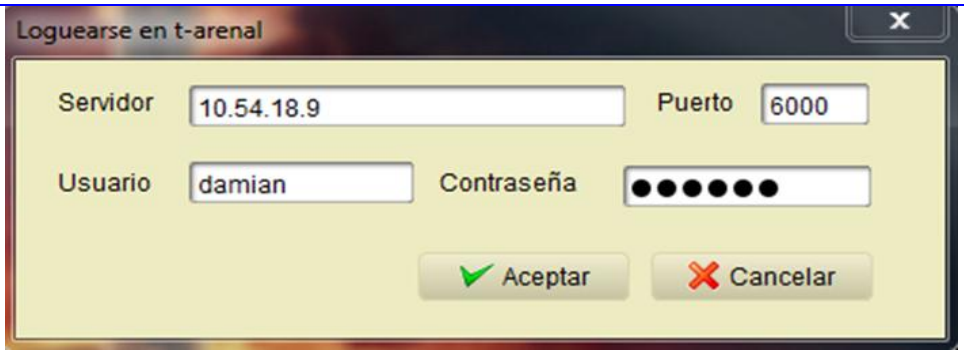
	Actor	Sistema
1.	El usuario interactúa con el sistema.	
2.		Muestra un formulario para introducir los datos • usuario • contraseña • servidor • puerto
3.	Introduce los datos requeridos y presiona el botón aceptar.	
4.		Valida los datos introducidos.
5.		Otorga los permisos necesarios para acceder a la aplicación, terminando así el caso de uso.
Flujos alternos		
Nº1. Los datos son incorrectos.		
	Actor	**Sistema**
1.		4.1 El sistema muestra un mensaje de error y regresa al punto 2 del flujo normal de eventos.
2.		
Requisitos funcionales	RF 1	
Prototipo Interfaz de Usuario		

Tabla 4: Descripción Caso de Uso *Gestionar Prueba*

Objetivo	El caso de uso permite al probador gestionar una prueba en el sistema.
Actores	Probador

Resumen	Se inicia cuando el administrador accede al sistema para gestionar una prueba. El caso de uso termina cuando la prueba es eliminada, creada o modificada por el probador en el sistema.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El probador ha sido autenticado.
Postcondiciones	Se actualiza, elimina o crea una prueba.

Flujo de eventos

Flujo básico Gestionar usuario

	Actor	Sistema
1.	Selecciona la opción ¨Nueva prueba¨. Selecciona la opción ¨Eliminar¨. Selecciona la opción ¨Modificar¨.	
2.		Para Nueva prueba: Ver sección 1. Para Eliminar prueba: Ver sección 2. Para Modificar prueba: Ver sección 3.
3.		Terminando así el caso de uso.

Sección 1: “Nueva Prueba”

Flujo básico “Gestionar usuario”

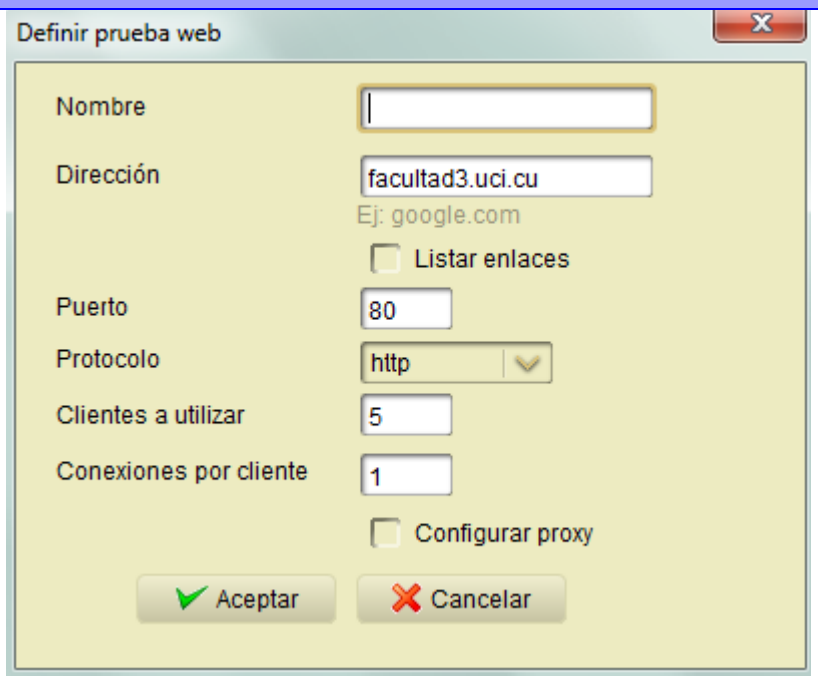
	Actor	Sistema
1.	Selecciona la opción Nueva prueba web. Selecciona la opción Nueva prueba base de datos	
2.		Para nueva prueba web: Ver sección 1 Para nueva prueba base de datos: Ver sección 2
3.		Terminando así el caso de uso

Sección 1: Definir prueba web

Flujo Básico “Definir prueba”

	Actor	Sistema
1.	Accede a la opción Prueba/Nueva/Nueva prueba web.	
2.		Muestra un formulario con los datos a insertar
3.	Define los datos de la prueba web y presiona el botón aceptar	
4.		Valida los datos
5.		Adiciona la prueba al panel de pruebas y desactiva la opción Prueba.
Flujo Alterno		
Nº 1 Los datos son incorrectos		
	Actor	**Sistema**
1.		**4.1** El sistema muestra un mensaje de error y regresa al punto 2 del flujo normal de eventos.
Flujo Alterno		
Nº 2 Existen campos en blanco		
	Actor	**Sistema**
1.		**4.1** El sistema muestra un mensaje de error y regresa al punto 2 del flujo normal de eventos.
Sección 2: Definir prueba para base de datos		
Flujo Básico "Definir prueba"		
	Actor	**Sistema**
1.	Accede a la opción Prueba/Nueva/Nueva prueba base de datos.	
2.		Muestra un formulario con los datos a insertar.
3.	Define los datos de la prueba a la base de datos y presiona el botón aceptar.	
4.		Adiciona la prueba al panel de pruebas

		y desactiva la opción Prueba.
Flujo alterno		
Nº 1 Existen campos en blanco		
	Actor	**Sistema**
1.		**4.1** El sistema muestra un mensaje de error y regresa al punto 2 del flujo normal de eventos.
Sección 2: "Eliminar prueba"		
Flujo básico "Gestionar prueba"		
	Actor	**Sistema**
1.	Selecciona la prueba que desea eliminar	
2.		Activa la opción eliminar
3.	Accede a la opción eliminar.	
4.		Elimina la prueba del sistema y del panel de pruebas.
Flujo Alterno		
Nº 1 La prueba no ha concluido su ejecución		
	Actor	**Sistema**
1.		**4.1** El sistema muestra un mensaje de error.
Sección 3: "Modificar prueba"		
Flujo básico "Gestionar prueba"		
	Actor	**Sistema**
1.	Selecciona la prueba que desea modificar	
2.		Activa la opción Modificar
3.	Accede a la opción Modificar prueba.	
4.		
5.		Muestra el formulario con los datos de la prueba

6.	Modifica los datos de la prueba	
7.		Actualiza los datos de la prueba.
Flujo alterno		
Nº 1 La prueba no ha concluido su ejecución		
1.		**7.1** El sistema muestra un mensaje de error.
Requisitos funcionales	RF 2, RF 3, Rf 7, RF 10	
Prototipo Interfaz de Usuario		
Definir prueba web		

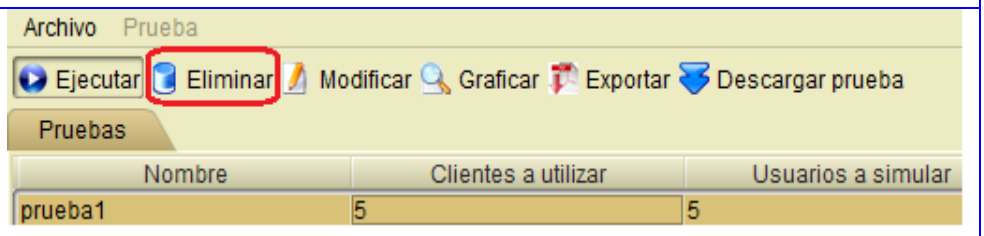
Definir prueba base de datos	
Eliminar prueba	
Modificar prueba	

Tabla 5: Descripción Caso de Uso *Ejecutar prueba*

Objetivo	Este caso de uso permite al usuario poder ejecutar una prueba definida en el sistema.
Actores	Probador
Resumen	Se inicia cuando el actor adiciona una prueba en el sistema y selecciona la opción Ejecutar.
Complejidad	Alta
Prioridad	Crítico

Precondiciones	El usuario ha adicionado una prueba
Postcondiciones	La prueba comienza su ejecución.

Flujo de eventos

Flujo básico Ejecutar prueba

	Actor	Sistema
1.	El usuario selecciona la prueba que desea ejecutar	
2.		Activa la opción ejecutar
3.	Presiona el botón ejecutar.	
4.		El sistema muestra un mensaje con el estado de la prueba. • Cola • Ejecución

Flujos alternos

Nº1. Existe una prueba en el sistema con el mismo nombre.

	Actor	Sistema
3.		4.1 El sistema muestra un mensaje de error, elimina la prueba del panel de pruebas y activa la opción Prueba

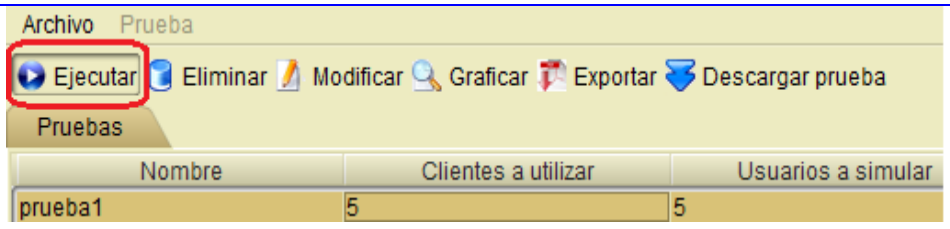
Requisitos funcionales	RF 4
Prototipo de Interfaz de Usuario	

Tabla 6: Descripción de Caso de Uso *Obtener resultados*

Objetivo	El caso de uso permite al probador seleccionar el tipo resultados que desea obtener
Actores	Probador
Resumen	Se inicia cuando el probador adicione una prueba al sistema.

Complejidad	Alta
Prioridad	Crítico
Precondiciones	El probador ha adicionado una prueba.
Postcondiciones	Se genera el reporte, se muestra la gráfica o se descarga la solución de la prueba.

Flujo de eventos

Flujo básico Obtener resultados

	Actor	Sistema
1.	Selecciona la opción ¨Generar reporte¨. Selecciona la opción ¨Graficar ¨. Selecciona la opción ¨Descargar prueba¨	
2.		Para Generar reporte: Ver sección 1. Para Graficar: Ver sección 2. Para Descargar prueba: Ver sección 3
3.		Terminando así el caso de uso.

Sección 1: "Generar reporte"

Flujo básico "Obtener resultados"

	Actor	Sistema
1.	Accede a la opción Generar reporte.	
2.		Muestra una ventana para que el usuario seleccione la dirección donde se guardará el reporte.
3.	Entra el nombre del archivo que desea guardar y oprime el botón aceptar.	
4.		Guarda el archivo con los datos de la prueba en la dirección especificada.
5.		Muestra un mensaje al usuario.

Flujos Alternos

Nº 1 La prueba no ha concluido su ejecución

	Actor	Sistema
1.		**2.1** El sistema muestra un mensaje de error.
Sección 2: "Graficar"		
Flujo básico "Obtener resultados"		
	Actor	**Sistema**
1.	Accede a la opción graficar.	
2.		Muestra una interfaz con la gráfica de la prueba seleccionada.
Flujos Alternos		
Nº 1 La prueba no ha concluido su ejecución		
	Actor	**Sistema**
1.		**2.1** El sistema muestra un mensaje de error.
Sección 3: Descargar solución		
Flujo básico "Obtener resultados"		
1.	Accede a la opción descargar solución.	
2.		Muestra una ventana para que el usuario seleccione la dirección donde se guardará la solución.
3.	Entra el nombre del archivo que desea guardar y oprime el botón aceptar.	
4.		Guarda el archivo con los datos de la prueba en la dirección especificada.
5.		Muestra un mensaje al usuario.
Flujos Alternos		
Nº 1 La prueba no ha concluido su ejecución		
	Actor	**Sistema**
1.		**2.1** El sistema muestra un mensaje de error.
Requisitos	RF 5, RF 6, RF 9	

funcionales	
Prototipo Interfaz de Usuario	

2.6 Diagrama de Clases del Diseño

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso. (14)

Como bien se puede observar en la figura 4 el sistema está compuesto de 4 clases principales, a continuación se detallan sus funcionalidades y se describen sus principales acciones:

- ✓ Control: Es la clase encargada de realizar todas las funcionalidades del sistema:
  - ➢ *Ejecutar prueba:* permite subir un problema (prueba de carga y estrés) a la plataforma T-arenal y comenzar su ejecución.
  - ➢ *Eliminar prueba:* permite dado la llave (id de la prueba) eliminar una prueba en la plataforma.
  - ➢ *Loguin:* permite al usuario autenticarse en el sistema.
  - ➢ *Estado:* permite obtener el estado de un problema en la plataforma.

- *Graficar:* dado los resultados obtenidos en la prueba permite la graficación de la misma.

✓ VisualPrincipal: es la clase que permite la interacción entre el sistema y T-arenal:

- *Update:* que es el método que permite mantener toda la aplicación actualizada a cada uno de los cambios que se realicen en el servidor.

✓ ServerLink: es la clase que permite generar la unidad de trabajo para cada una de las estaciones de trabajo que le solicitan, sus principales funcionalidades son:

- *GenerateWorkUnit:* este método es el encargado de crear todas las pruebas que se realizarán.
- *getStatus*: este método es el que permite obtener en la plataforma todos los resultados de las pruebas, los cuales se obtendrán en la clase control.
- *processResult:* es el encargado de procesar el resultado por cada uno de los clientes que culminan su ejecución y el que construye el resultado final de la prueba que se realiza.

✓ TestClient: es la clase encargada de crear la tarea que le es enviada por el ServerLink, sus principales funcionalidades son:

- *processUnit:* es método es el que permite procesar una unidad de trabajo enviada específica.
- *RealizeLinkConnection:* esta funcionalidad es la encargada de comenzar la ejecución y de realizar todas las conexiones concurrentemente.

2.7 Patrones de diseño

Como patrón de diseño su utiliza el patrón GRASP. Este se evidencia en el sistema de la siguiente manera:

✓ Controlador: Este patrón se evidencia en la herramienta con la clase Control que es la encargada de realizar todas las funciones necesarias para trabajar con el sistema.

✓ Alta cohesión: El patrón Alta Cohesión permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Ejemplos de ello son las clases Control y VisualPrincipal, las cuales están formadas por varias funcionalidades relacionadas, siendo las responsables de definir las acciones y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a sus propiedades. Cada

una de ellas contiene solamente operaciones correspondientes con su responsabilidad

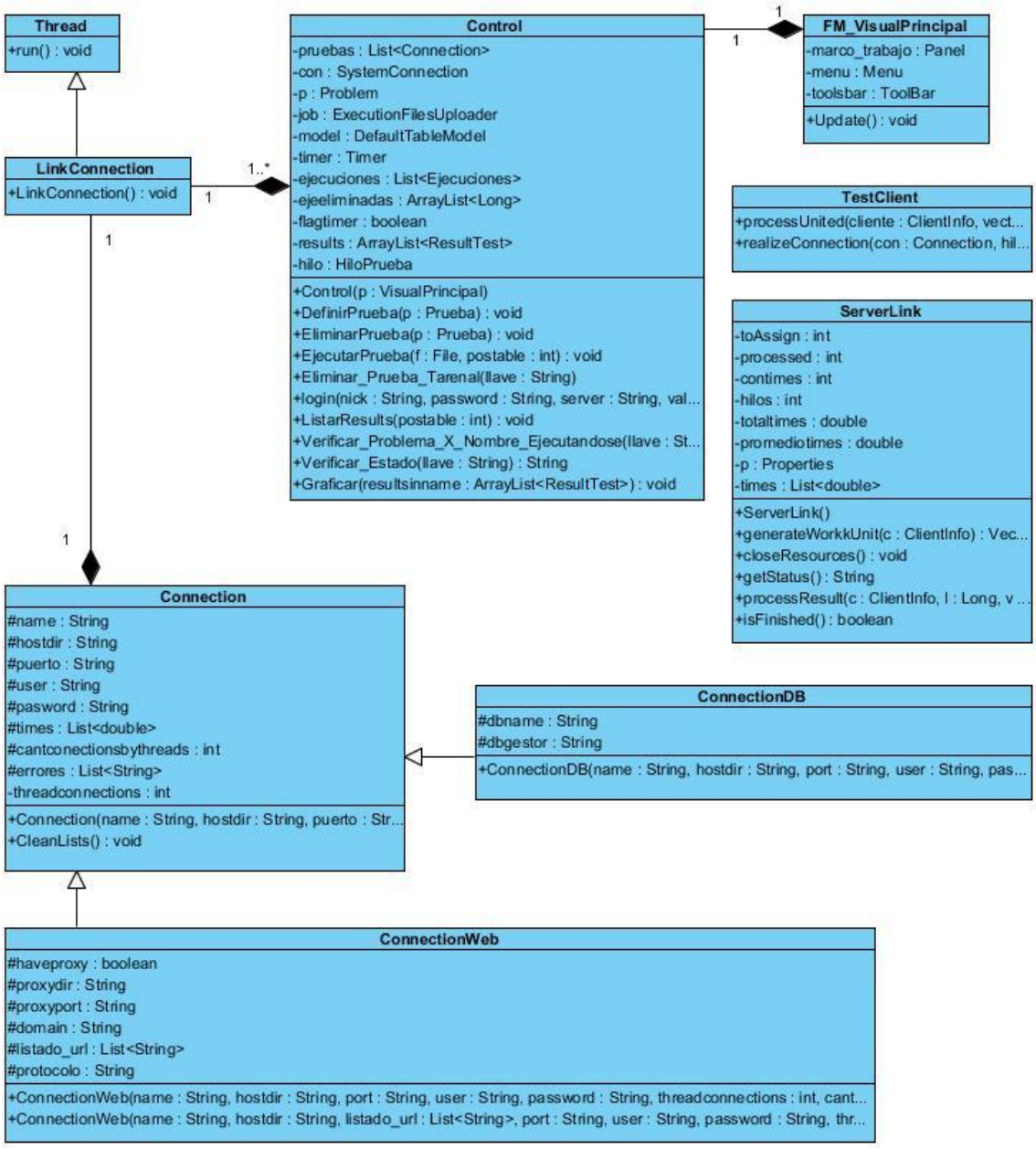


Figura 4: Diagrama de Clases del Diseño

- ✓ Bajo acoplamiento: La aplicación de este patrón permite asignar una responsabilidad de modo que no se incremente el acoplamiento y, por tanto, no produzcan los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes y reutilizables, reduciéndose el impacto de los cambios. Debe considerarse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades. En la implementación del sistema es aplicado este patrón en la mayoría de las clases, propiciando que

los componentes sean fáciles de entender por separado y de reutilizar, y que no sean afectados por cambios en otros componentes.

2.8 Patrón Arquitectónico Cliente-Servidor

En la figura 5, se muestra el patrón utilizado cliente-servidor entre la herramienta y el sistema distribuido. *La PC_herramienta y los Clientes_T-arenal* actúan como clientes.

El *Servidor T-arenal* siempre se comporta como servidor, en el momento de enviarle la respuesta de la tarea a la PC_herramienta y para darle una tarea al *Servidor de Peticiones*. El Servidor de peticiones se comporta de las dos formas (cliente y servidor). Como cliente al solicitar una tarea al *Servidor_T-arenal (Servidor Central)* y como servidor cuando distribuye la tarea a los *Clientes_T-arenal* que solicitan la tarea.

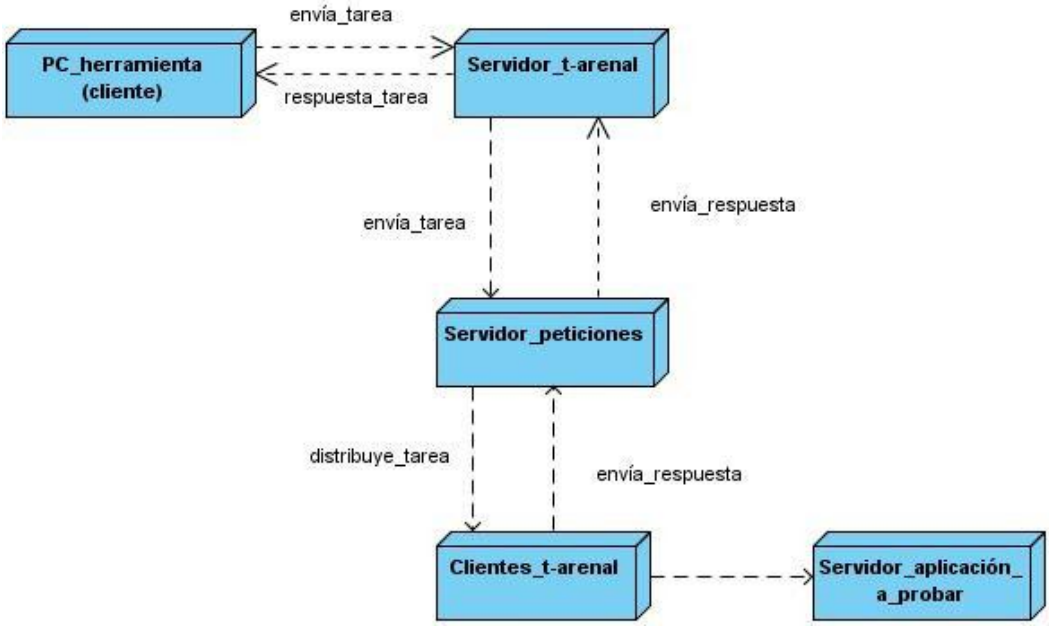


Figura 5: Patrón arquitectónico cliente-servidor aplicado al sistema

2.9 Descripción de la vista lógica

La vista lógica, contiene las clases del diseño más importantes, organizadas por paquetes y subsistemas en capas de trabajo. Esta vista describe los paquetes en los que se dividirá la herramienta. En la figura siguiente, se muestra la vista lógica de la herramienta.

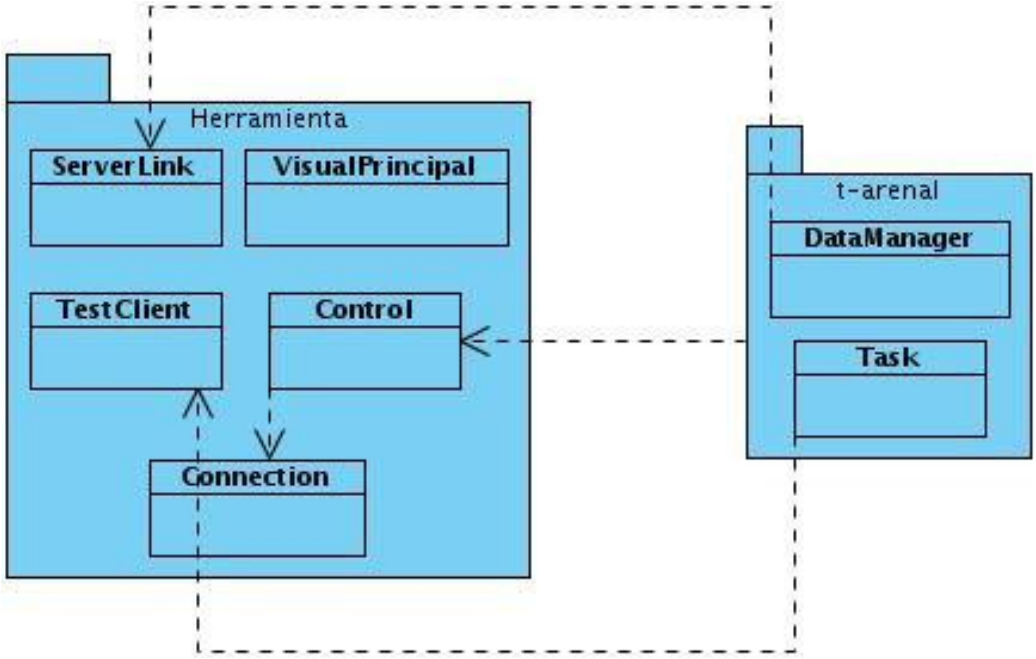


Figura 6: Vista lógica del sistema

Como bien se muestra en la figura el paquete T-arenal (que no es más que un subsistema que utiliza la aplicación) posee dos archivos fundamentales el DataManager (es el archivo que le permite a T-arenal saber cómo se ejecutan las acciones y como procesar el resultado) y el archivo Task (es el que define como se realiza la tarea).

La herramienta posee dos archivos que heredan de estas clases ServerLink (hereda de DataManager) y TestClient (hereda de Task), en estos dos archivos es donde se define como se ejecuta la prueba de carga y estrés para las distintas aplicaciones (web y base de datos). Cuando se ejecuta una prueba en el VisualPrincipal esta clase realiza la acción de subir a T-arenal un problema (que está compuesto por un DataManager, Task y las clases o librerías necesarias para su ejecución). T-arenal toma estos archivos y ejecuta la tarea asignada, obteniendo un resultado que es el que la clase control toma para su posterior visualización en la interfaz de SoftRen, de ahí que los datos en esta interfaz se actualicen solos.

2.10 Descripción de la vista de despliegue

La vista de despliegue describe los principales nodos físicos que se necesitan para desplegar el sistema y la relación entre ellos mediante los protocolos de comunicación. En la figura 6, se muestra una propuesta de la vista de despliegue de la herramienta. (Ver figura 8)

- ✓ **PC Herramienta:** Se refiere a la computadora donde se encuentra la herramienta.

- ✓ **Servidor central:** Se refiere al servidor central de T-arenal, el cual realizaría las acciones para distribuir las tareas a los servidores de peticiones.
- ✓ **Servidor de peticiones:** Servidores encargados de distribuir las tareas a los clientes de prueba.
- ✓ **Cliente prueba:** Clientes en los cuales se realizan las tareas enviadas desde el servidor (en este caso las pruebas).
- ✓ **Aplicación a probar:** Aplicación Cliente-Servidor (Web o Base de Datos), a la cual se le realizan las pruebas.

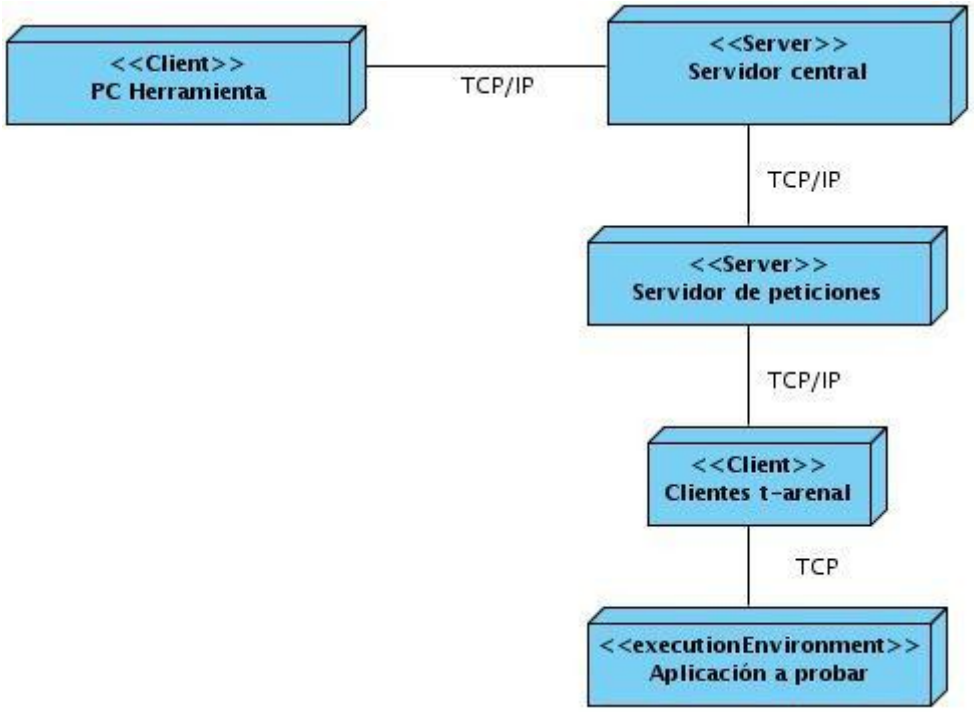


Figura 7: vista de despliegue

2.11 Conclusiones del capítulo

En el capítulo concluido se inició el desarrollo de un sistema que aprovechando al máximo las estaciones de trabajo de la Universidad de las Ciencias Informáticas permita realizar pruebas de carga y estrés a servidores de aplicaciones. Se definieron los requisitos funcionales y no funcionales mediante el levantamiento de requisitos, lo que permitió obtener el diseño y realización de la propuesta de solución a la problemática planteada. Estas acciones facilitarán el desarrollo de un sistema que utilizando al máximo las estaciones de trabajo de la UCI, mediante la Plataforma de Cálculo Distribuido T-arenal, permita realizar pruebas de carga y estrés a servidores de aplicación (web y base de datos).

Capítulo 3: Implementación y validación del sistema para realizar pruebas de carga y estrés a aplicaciones cliente servidor utilizando la Plataforma de Tareas Distribuidas T-arenal.

En este capítulo se describen los elementos del diseño en términos de componentes reflejados en el diagrama de componentes. Se describe el código fuente fundamental de la herramienta y posteriormente se realizan pruebas de validación para verificar el correcto funcionamiento de las funcionalidades del sistema.

3.1 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

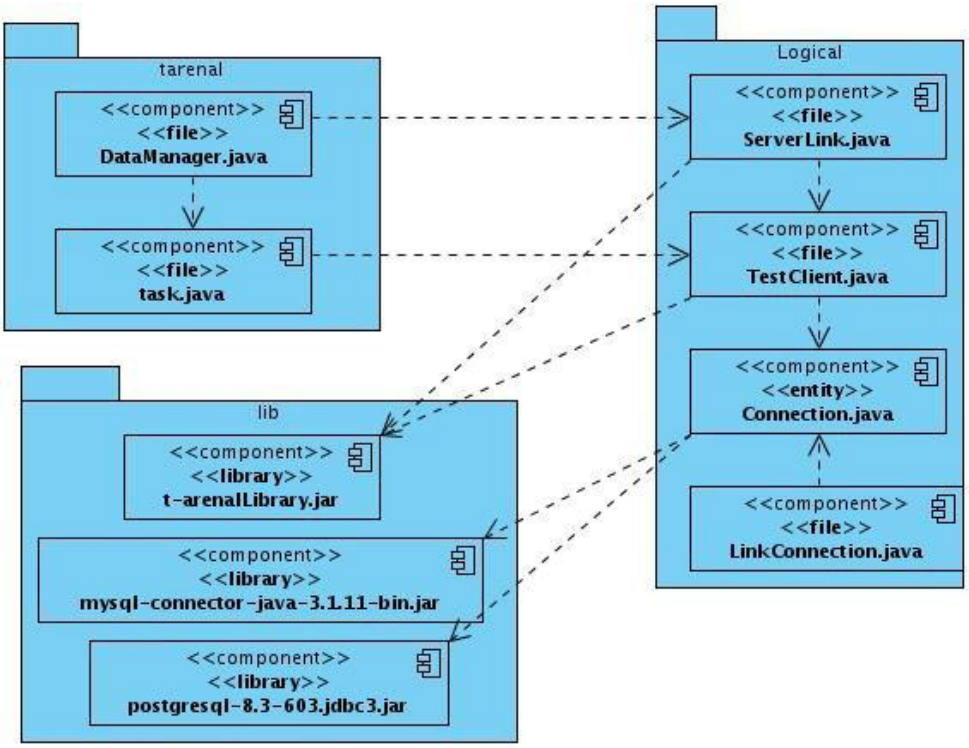


Figura 8: Diagrama de componentes

Como bien se puede observar en la figura 8 la herramienta para su funcionamiento necesita de varios componentes que la integran a continuación se explica cómo se realiza el funcionamiento y las dependencias de cada uno de estos componentes.

- ✓ *Paquete Logical:* este paquete posee 1 componentes principales, que es el que permite definir la tarea (prueba de carga y estrés) que se desea ejecutar.
    - ➢ ServerLink: es la encargada de definir las pruebas, para definir estas pruebas se necesita el archivo TestClient, el archivo Connection, LinkConnection, la librería de T-arenal, y las librerías que permiten la conexión a las base de datos (mysql, postgresql). Este componente es heredado del paquete T-arenal, y se tiene que re implementar de acuerdo a las necesidades de cada tarea que se desea ejecutar.

3.2 Estándar de codificación

Un estándar de codificación es un conjunto de políticas por las que se tiene que regir un programador a la hora de desarrollar un programa e indican la estructura del código.

Declaraciones:

- ✓ Se recomienda una declaración por línea, ya que facilita los comentarios.
- ✓ No poner diferentes tipos en la misma línea.

Inicializaciones

- ✓ Intentar inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.
- ✓ Poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{" y "}".) No esperar al primer uso para declararlas; puede confundir a programadores no preavisados y limitar la portabilidad del código dentro de su ámbito de visibilidad.

Líneas en blanco

- ✓ Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas.
- ✓ Se deben usar siempre dos líneas en blanco en las siguientes circunstancias:
    - ➢ Entre las secciones de un fichero fuente
    - ➢ Entre las definiciones de clases e interfaces.
- ✓ Se debe usar siempre una línea en blanco en las siguientes circunstancias:
    - ➢ Entre métodos
    - ➢ Entre las variables locales de un método y su primera sentencia
    - ➢ Antes de un comentario de bloque o de un comentario de una línea
    - ➢ Entre las distintas secciones lógicas de un método para facilitar la lectura.

3.3 Pruebas funcionales

El software desarrollado en las empresas destinadas a este tipo de producción es refinado a través de iteraciones en su ciclo de vida. Este ciclo se beneficia siguiendo un proceso iterativo equivalente donde en cada iteración el equipo de desarrollo produce uno o más casos de pruebas, cada uno de estos casos de pruebas son candidatos potenciales a probar. (15)

Para probar que el software funciona correctamente existen dos tipos de métodos de pruebas:

- ✓ *Caja blanca:* Esta prueba examina el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.
- ✓ *Pruebas de caja negra:* Esta pruebas son las que se llevan a cabo sobre la interfaz del software.

3.3.1 Pruebas de caja negra

Las pruebas de Caja Negra se centran principalmente en los requisitos funcionales del software, obteniéndose un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa. En ellas se ignora la estructura de control. Este método de prueba es un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca, entre los que se pueden citar los siguientes: [18]

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

Para preparar los casos de pruebas es necesario contar con un conjunto de datos, válidos o inválidos, que ayuden a la ejecución de estos casos y que permitan que el sistema se ejecute en todas sus variantes. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa.

Dentro del método de Caja Negra la técnica de partición de equivalencia es una de las más efectivas, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software y descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. Esta técnica fue la utilizada para diseñar los casos de prueba de caja negra aplicados a la extensión, dividiendo el campo de entrada en variables de equivalencia con juegos de datos de entrada y salida. Dichas variables se clasifican en dos tipos: las *válidas*, que representan entradas válidas al programa, y las *no válidas*, que

representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

3.3.1.1 Escenario 1: Caso de prueba Adicionar prueba web

Descripción: El usuario selecciona la opción Prueba/Nueva/Prueba web

Condiciones: Debe haberse autenticado en el sistema.

Tabla 7: Caso de prueba Adicionar prueba web

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Adicionar prueba con datos válidos	Se adiciona la prueba correctamente en el sistema	Se muestra en el panel de trabajo la prueba adicionada.	Se selecciona en el menú principal la opción *Adicionar Prueba Web*, se rellena el formulario y se oprime el botón **Aceptar**
EC 1.2 Adicionar prueba con datos no válidos en el campo nombre	No se adiciona la prueba en el sistema y muestra un mensaje de error	El sistema muestra el mensaje *"El campo nombre no debe comenzar con números ni contener caracteres especiales",* luego vacía el campo nombre.	Se selecciona en el menú principal la opción *Adicionar Prueba Web*, se rellena el formulario y se oprime el botón **Aceptar**
EC 1.3 Adicionar prueba con datos no válidos en el campo dirección	No se adiciona la prueba en el sistema y muestra un mensaje de error	El sistema muestra el mensaje *"La dirección... es invalida",* luego vacía el campo dirección.	Se selecciona en el menú principal la opción *Adicionar Prueba Web*, se rellena el formulario y se oprime el botón **Aceptar**
EC 1.4 Adicionar prueba con datos no válidos en los campos numéricos	No se adiciona la prueba en el sistema y muestra un mensaje de error	El sistema no permite que se escriban estos caracteres en los campos numéricos.	Se selecciona en el menú principal la opción *Adicionar Prueba Web*, se rellena el formulario y se oprime el botón **Aceptar**
EC 1.5 Adicionar prueba con campos en blanco	No se adiciona la prueba en el sistema y muestra un mensaje de error	El sistema muestra el mensaje *"No deben existir campos en blanco".*	Se selecciona en el menú principal la opción *Adicionar Prueba Web*, se rellena el formulario y se oprime el botón **Aceptar**

3.3.1.2 Escenario 2: Caso de prueba Adicionar prueba base de datos

Descripción: El usuario selecciona la opción Prueba/Nueva/Prueba web

Condiciones: Debe haberse autenticado en el sistema.

Tabla 8: Caso de prueba Adicionar prueba base de datos

Escenario	Descripción	Respuesta del sistema	Flujo central

EC 2.1 Adicionar prueba con datos válidos	Se adiciona la prueba correctamente en el sistema	Se muestra en el panel de trabajo la prueba adicionada.	Se selecciona en el menú principal la opción *Adicionar Prueba base datos*, se rellena el formulario y se oprime el botón **Aceptar**
EC 2.2 Adicionar prueba con datos no válidos en el campo nombre	No se adiciona la prueba en el sistema y muestra un mensaje de error	El sistema muestra el mensaje *"El campo nombre no debe comenzar con números ni contener caracteres especiales",* luego vacía el campo nombre.	Se selecciona en el menú principal la opción *Adicionar Prueba base datos*, se rellena el formulario y se oprime el botón **Aceptar**
EC 2.3 Adicionar prueba con datos no válidos en los campos numéricos	No se adiciona la prueba en el sistema y muestra un mensaje de error	El sistema no permite que se escriban estos caracteres en los campos numéricos.	Se selecciona en el menú principal la opción *Adicionar Prueba base datos*, se rellena el formulario y se oprime el botón **Aceptar**
EC 2.4 Adicionar prueba con campos en blanco	No se adiciona la prueba en el sistema y muestra un mensaje de error	El sistema muestra el mensaje *"No deben existir campos en blanco".*	Se selecciona en el menú principal la opción *Adicionar Prueba base datos*, se rellena el formulario y se oprime el botón **Aceptar**

3.3.1.3 - Escenario 3: Caso de prueba Eliminar prueba

Descripción: El usuario selecciona la prueba que desea eliminar y oprime el botón Eliminar que se encuentra en ubicado en la parte superior izquierda del sistema.

Condiciones: Debe haberse adicionado al menos una prueba y debe haber concluido su ejecución.

Tabla 9: Caso de prueba Eliminar prueba

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 3.1 Eliminar una prueba que ha concluido su ejecución.	Se elimina correctamente la prueba del sistema y de la Plataforma de Tareas Distribuida T-arenal.	Muestra un mensaje de confirmación *"¿Está seguro de eliminar esta prueba?"* y la respuesta del usuario es OK.	Se selecciona la fila de la prueba y se oprime el botón eliminar.
EC 3.2 Eliminar un prueba que no ha concluido su ejecución.	El sistema muestra un mensaje de Información.	El sistema muestra el mensaje *"La prueba no puede eliminarse ya que se encuentra en ejecución".*	Se selecciona la fila de la prueba y se oprime el botón eliminar.

3.3.1.4 - Escenario 4: Caso de prueba Ejecutar prueba

Descripción: El usuario selecciona la prueba que desea ejecutar y oprime el botón Ejecutar que se encuentra en ubicado en la parte superior izquierda del sistema.

Condiciones: Debe haberse adicionado al menos una prueba.

Tabla 10: Caso de prueba Ejecutar prueba

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 4.1 Ejecuta una prueba en T-arenal	Se comienza la ejecución de la prueba en la Plataforma de Tareas Distribuidas T-arenal.	Muestra un mensaje con el estado de la prueba (en ejecución, en cola, concluido).	Se selecciona la fila de la prueba y se oprime el botón ejecutar.
EC 4.2 Ejecutar una prueba que ya se está ejecutando en T-arenal	El sistema muestra un mensaje de Información.	El sistema muestra el mensaje *"La prueba se encuentra en ejecución en el sistema".*	Se selecciona la fila de la prueba y se oprime el botón eliminar.
EC 4.3 Ejecutar una prueba que ya se está encuentra adicionada en T-arenal	El sistema muestra un mensaje de error.	El sistema muestra el mensaje *"Ya existe una prueba con ese nombre en el sistema".*	Se selecciona la fila de la prueba y se oprime el botón eliminar.

3.3.1.5 - Escenario 5: Caso de prueba Generar reporte

Descripción: El usuario selecciona la prueba que desea obtener el reporte y oprime el botón Exportar que se encuentra en ubicado en la parte superior izquierda del sistema.

Condiciones: Debe haberse adicionado al menos una prueba y debe haber concluido su ejecución

Tabla 11: Caso de prueba Generar reporte

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 5.1 Generar un reporte de una prueba que no ha	El sistema muestra el informe (generado en la extensión .pdf)	Muestra el informe generado por el sistema.	Se selecciona la fila de la prueba y se oprime el botón exportar.

concluido su ejecución.			
EC 5.1 Generar un reporte de una prueba que ha concluido su ejecución.	El sistema muestra un mensaje de Información.	El sistema muestra el mensaje *"No se puede generar el informe hasta que la prueba haya concluido su ejecución".*	Se selecciona la fila de la prueba y se oprime el botón exportar.

3.4 Resultados de las pruebas

La figura 9 muestra que en la primera iteración se detectaron 10 no conformidades de las cuales fueron resueltas las 9, en la segunda iteración se detectaron 5 no conformidades (incluyendo la que quedaba pendiente de la primera iteración) y se resolvieron las 5, en la tercera iteración se detectaron 1 no conformidad de las cuales fueron resueltas 1 y en la cuarta iteración no se encontraron no conformidades. Cada una de las no conformidades detectas fueron resueltas inmediatamente por el equipo de desarrollo (exceptuando 1 de la primera iteración ya que no se podía solucionar en el momento). La solución de las no conformidades contribuyó a que la aplicación alcanzara una mejor calidad y cumpliera con los requisitos planteados en las Historias de Usuarios. Las No Conformidades por cada una de las iteraciones se especifican en el documento complementario "NC_Sistema para Pruebas de Rendimiento SoftRen 2.0".

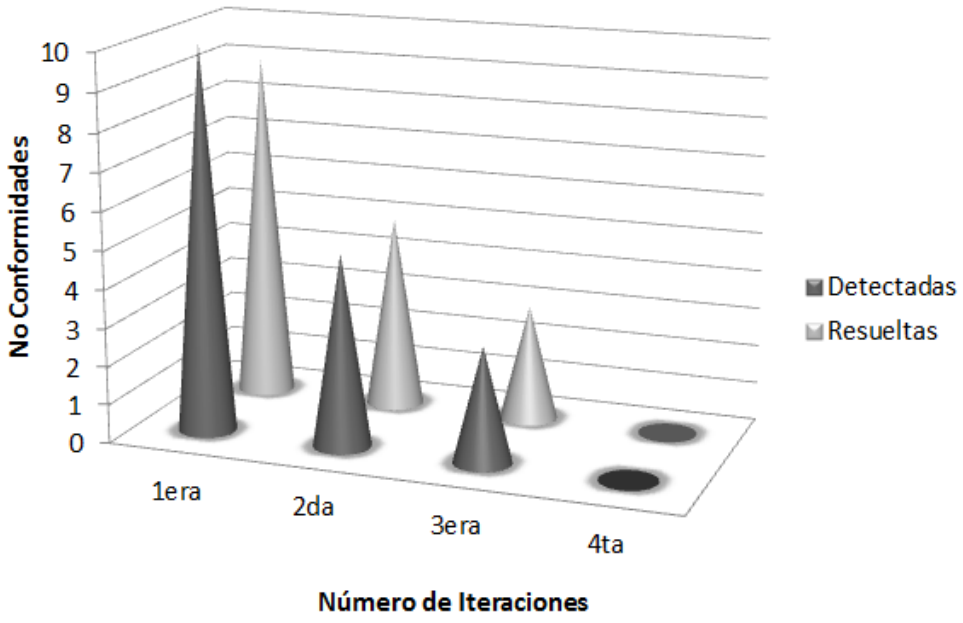


Figura 9: Gráfico de No Conformidades por Iteración

3.5 Pruebas de validación

Una vez terminada la implementación, se realizaron pruebas de validación para verificar que las funcionalidades del sistema satisfacen las necesidades del cliente y que el mismo realiza las pruebas de forma satisfactoria.

Se tomó de muestra como servidor web el *Sistema para la publicación y el intercambio de imágenes en la red social de la Universidad de las Ciencias Informáticas* alojado en el servidor *10.54.18.177/publica.*

Como servidor de base de datos se tomó *10.54.18.177:5432/base_datos_salvador* que como principales características, como gestor de base de datos Postgreslq 8.0, con 5 dimensiones , 2 hechos y 6 tablas de meta dato.

3.5.1 Pruebas a la aplicación web

Para realizar la prueba a este servidor se realizaron 10 pruebas incrementando la cantidad de los usuarios en escala de 100.

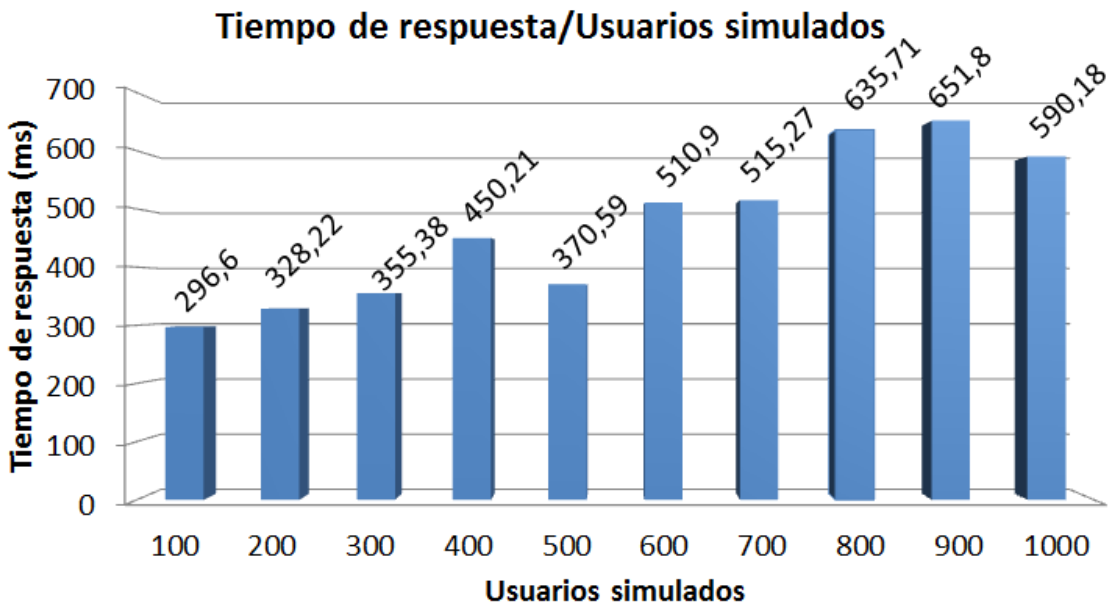


Figura 10: Resultados gráficos de la prueba (escala de 100)

3.5.2 Pruebas a la base de datos

Para realizar la prueba a este servidor se realizaron 10 pruebas incrementando la cantidad de los usuarios en escala de 10.

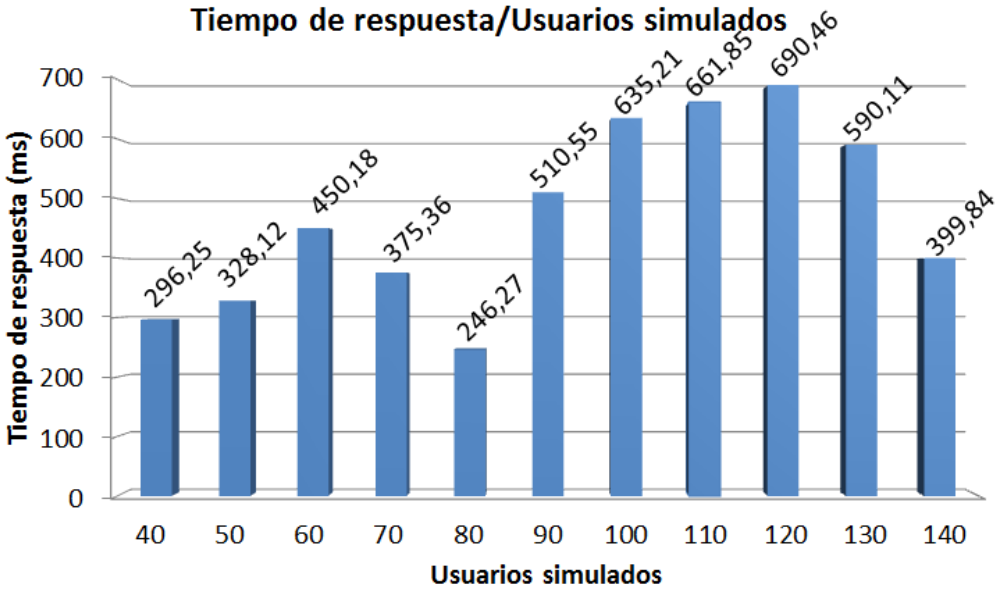


Figura 11: Resultados gráficos de la prueba (escala de 10)

Al concluir las pruebas al servidor web y la base de datos se verificó el funcionamiento del sistema (que estableciera la conexión con los servidores) a través de los log de los sistemas probados, verificándose el buen funcionamiento de la herramienta.

3.6 Conclusiones del capítulo

En el desarrollo del capítulo concluido se elaboró el diagrama de componentes que permitió organizar y definir las dependencias entre ejecutables, códigos fuentes y librerías que se utilizarán en el sistema. Se define el estándar de programación, los cuales permiten lograr una mejor organización del código fuente. Se describieron las principales pruebas realizadas al sistema, indicando para cada una la respuesta de la aplicación. De forma general cada una de las pruebas arrojaron resultados satisfactorios, ya que como resultado se obtuvo un número reducido de no conformidades, las cuales fueron solucionadas de forma rápida por el equipo de desarrollo, contribuyendo a obtener un producto de mayor calidad.

Conclusiones Generales

La presente investigación tiene un papel fundamental en la definición e implementación de un sistema informático que permita realizar pruebas de carga y estrés a aplicaciones web y base de datos:

- ✓ A partir del estudio de las diferentes herramientas y sistemas que se utilizan en la Universidad de las Ciencias Informáticas para realizar pruebas de carga y estrés, se definió el marco teórico conceptual de la investigación, permitió identificar la problemática y definir las bases para analizar, diseñar e implementar un sistema que aprovechando los sistemas distribuidos, logre disminuir la carga de trabajo en los servidores de pruebas y aproveche al máximo las estaciones de trabajo de la Universidad de las Ciencias Informáticas..
- ✓ La definición de los requisitos funcionales y no funcionales permitió obtener el diseño del sistema, con el cual se logró un mejor entendimiento de las principales funcionalidades del Sistema para realizar pruebas de carga y estrés a aplicaciones cliente-servidor utilizando la Plataforma de Cálculo Distribuido T-arenal.
- ✓ Aplicadas las técnicas de validación a la herramienta mostró el buen funcionamiento de la misma en distintos escenarios de pruebas, comprobando así su funcionamiento de acuerdo a los requisitos planteados.

Al finalizar se obtiene una investigación sustentada en elementos teóricos, todos los artefactos definidos dentro del proceso de desarrollo de *software* seleccionado y un sistema que utilizando las estaciones de trabajo con que cuenta la Universidad de las Ciencias Informáticas permite realizar pruebas de carga y estrés a servidores de aplicación (web o base de datos), cumpliéndose con el objetivo principal del trabajo, obteniéndose los resultados esperados.

Recomendaciones

1. Implementar las funcionalidades que permitan realizar pruebas a las funciones y consultas de las base de datos.
2. Implementar las funcionalidades que permitan obtener los distintos tipos de errores de las aplicaciones web.
3. Implementar las funcionalidades que permitan graficar los resultados de las pruebas en función del rendimiento.

Bibliografía Citada

1. **Gonzalez, J.** *Las normas de la calidad del software.* España : Addison-Wesley. Iberoamericana.

2. Pruebas. [En línea] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php.

3. **IEEE.** *Standard Glossary of Software Engineering Terminology.* s.l. : IEEE, 1990. STD 610-1990.

4. **Pressman, Roger.** *Ingienería de Software 3ra Ed.* McGraw Hill : s.n., 1993.

5. **Pressman, R.** *Ingeniería del Software: Un enfoque práctico.* McGraw Hill : s.n., 2002.

6. **Yunming, Pu y Mingna, Xu.** Load Testing for Web Applications.

7. **Foundation, Apache Software.** Apache JMeter. [En línea] http://jmeter.apache.org/usemanual/index.html.

8. Solex. Web application testing with Eclipse. [En línea] http://solex.sourceforge.net.

9. **Facultad6, Grupo Bioinformática.** *T-arenal v2.0.1-Manual de Usuario.*

10. **Billy Reynoso, Carlos.** *Introducción a la Arquitectura de Software.* 2004.

11. **Baeza, Pablo Nicolas.** *Visual Paradigm DB Visual Architect SQL. D y TSystem.*

12. Sitio Oficial de Eclipse. [En línea] http://www.eclipse.org.

13. **García Jalón, Javier.** *Aprende java como si estuvieras en primero.*

14. Ecured. [En línea] http://www.ecured.cu/index.php/Diagrama_de_Clase.

15. Ecured. [En línea] http://www.ecured.cu/index.php/Flujo_de_pruebas_de_un_software#cite_ref-0.

16. **Terna, Carolina.** Pruebas de Funcionalidad. [En línea] 02 de 05 de 2005. http://carolina.tema.net/ingsw3/datos/Pruebas_Funcionales.pdf.9.

17. **Noda Castillo, Eugenio.** *Herramienta informática para realizar pruebas de rendimiento a aplicaciones cliente/servidor.* [Tesis de Diploma] La Habana : s.n., 2011.

18. Pruebas de Caja Negra. *Pruebas de Caja Negra.* [En línea] www.ecured.cu/index.php/Pruebas_de_caja_negra..

Bibliografía Consultada

1. **Gonzalez, J.** *Las normas de la calidad del software.* España : Addison-Wesley. Iberoamericana.

2. Pruebas. [En línea] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php.

3. **IEEE.** *Standard Glossary of Software Engineering Terminology.* s.l. : IEEE, 1990. STD 610-1990.

4. **Pressman, Roger.** *Ingienería de Software 3ra Ed.* McGraw Hill : s.n., 1993.

5. **Pressman, R.** *Ingeniería del Software: Un enfoque práctico.* McGraw Hill : s.n., 2002.

6. **Foundation, Apache Software.** Apache JMeter. [En línea] http://jmeter.apache.org/usemanual/index.html.

7. Solex. Web application testing with Eclipse. [En línea] http://solex.sourceforge.net.

8. **Facultad6, Grupo Bioinformática.** *T-arenal v2.0.1-Manual de Usuario.*

9. **Billy Reynoso, Carlos.** *Introducción a la Arquitectura de Software.* 2004.

10. **Baeza, Pablo Nicolas.** *Visual Paradigm DB Visual Architect SQL. D y TSystem.*

11. Sitio Oficial de Eclipse. [En línea] http://www.eclipse.org.

12. **García Jalón, Javier.** *Aprende java como si estuvieras en primero.*

13. Ecured. [En línea] http://www.ecured.cu/index.php/Diagrama_de_Clase.

14. Ecured. [En línea] http://www.ecured.cu/index.php/Flujo_de_pruebas_de_un_software#cite_ref-0.

15. **C. Jorgensen, Paul.** *Software Testing: A Craftsman Approach .* s.l. : CRC, 2002.

16. **Duan, Nian.** *Software performance testing and practical.* s.l. : Tsinghua university Press, 2006.

17. **LoadRunner.** *LoadRunner 8.0 user Manual.* s.l. : Mercury Interactive, 2005.

18. **U. Smith, Connie y Lioyed G., Williams.** *Performance solutions a practical guide to creating responsive scalable software.* s.l. : China Machine Press, 2003.

19. **Musa, John D.** *Software Realibility Engineering.* s.l. : McGraw Hill Education, 1999.

20. **C. Kallepalli, J. Tian.** *Measuring and modeling usage and realibility for statical web testing.* s.l. : IEEE Trans on Software Engineering, 2001.

21. **Xiaopeng, Deng.** *Progressing in Testing for web Aplicattions.* s.l. : Journal of Computer Research and Development, 2007.

22. **Terna, Carolina.** Pruebas de Funcionalidad. [En línea] 02 de 05 de 2005. http://carolina.tema.net/ingsw3/datos/Pruebas_Funcionales.pdf.9.

23. **Yunming, Pu y Mingna, Xu.** Load Testing for Web Applications.

24. NetBeans. [En línea] [Citado el: 2012 de 06 de 13.] http://netbeans.org/community/releases/61/index_es.html.

25. **Noda Castillo, Eugenio.** *Herramienta informática para realizar pruebas de rendimiento a aplicaciones cliente/servidor.* [Tesis de Diploma] La Habana : s.n., 2011.

26. Pruebas de Caja Negra. *Pruebas de Caja Negra.* [En línea] www.ecured.cu/index.php/Pruebas_de_caja_negra..