Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

EVALUACIÓN DE FRAMEWORKS WEB USANDO TÉCNICAS DE MEDICIÓN DE RENDIMIENTO Y MÉTRICAS DE SOFTWARE APLICADAS SOBRE COMPONENTES DEL MÓDULO DE ADMINISTRACIÓN DEL SERVICIO COMUNITARIO. CASO DE ESTUDIO: JAVA SERVER FACES Y STRUTS 2

Trabajo Especial de Grado
Presentado ante la ilustre
UNIVERSIDAD CENTRAL DE VENEZUELA

Por los Bachilleres:

Laya, Franklin
Ramos, Jonathan

Para optar al Título de
Licenciado en Computación

Tutor:
Andrés Sanoja

Caracas, Marzo de 2009

Resumen

Este trabajo abarca los siguientes tópicos: frameworks, métricas de software y de rendimiento, y así como los pasos o estrategias llevadas a cabo para implementar un conjunto de herramientas que permitan evaluar componentes de software. Los componentes evaluados forman parte del Módulo de Administración de Información del Servicio Comunitario de la Facultad de Ciencias, la cual se desarrolló, en el mismo tenor, en los frameworks Java estudiados en éste trabajo: Java Server Faces (JSF) y Struts 2. Se tomó especial consideración que incluyeran las mismas funcionalidades y se desarrollaran usando la misma lógica de programación. Todo esto con la finalidad de dar respuesta a las siguientes preguntas ¿qué framework es más eficaz o ventajoso, cuando se implementa una aplicación relacionadas con entornos Web de Java?

En nuestro trabajo se tomó en cuenta el framework Struts 2, el cual posee un controlador que viene inserto como un Servlet interno de Struts 2, este último se usó para dirigir la relación existente entre las acciones, la interfaz y los Beans con la finalidad de trabajar la lógica de negocio; simultáneamente las etiquetas de Struts ayudaron a construir la vista y las interfaces en conjunción con la tecnología Ajax.

Otro framework considerado es Java Server Faces, el cual se usó de manera análoga al anterior, para ejecutar la lógica de negocio o acciones del mismo modelo y en el que se contemplaron la interfaz gráfica de usuario, los eventos y sus componentes como si fuesen una aplicación web estándar para diseñar las diferentes interfaces o vistas que luego son mostradas al usuario.

Luego se describe en detalle el subconjunto de métricas seleccionadas, el ambiente y estrategias para realizar la medición y las pruebas de rendimiento y el analisis, diseño e implementación de los componentes de la herramienta Web, de nuestra propia autoría, para el Servicio Comunitario

La evaluación se realizó de dos maneras. Las métricas de software se calcularon por medio de un agregado de software instalado sobre el entorno de desarrollo Eclipse, Metrics 1.3.6, que permitió al usuario exportar los datos obtenidos en un archivo de texto para su posterior procesamiento y visualización. Las métricas de rendimiento fueron obtenidas mediante la herramienta Httperf v9, para plataformas Linux, la cual sometió a stress a los componentes seleccionados como caso de estudio y arrojó una serie de resultados sobre el rendimiento.

Finalmente los resultados capturados son mostrados en una herramienta, llamada Visor, que permite visualizar de manera gráfica los valores de las métricas previamente obtenidas.

Agradecimientos y Dedicatorias

Franklin:

A Dios, por ponerme pruebas muy duras en la vida, pero también por darme cosas buenas.

A mi madre Dora Esperanza Laya, por darme la vida, ser mi amiga, ser mi todo y porque a pesar que Dios me la quitó, con su carácter recio me guió y me enseñó a ser un hombre de bien, este logro era uno de sus más anhelados deseos.

A mis abuelas Luisa Alejandrina Laya, Carmen Emilia Febres de Loreto, a mis tías Gladis Loreto, Cruz Amelia Febres Loreto, a mis tíos maternos y paternos, y a mis primas, por guiarme y orientarme cuando era un estudiante adolescente.

A mi hermana Milagros Loreto, por ser la única hermana en convivir en el día a día conmigo en la Facultad y por estar tan pendiente de mí.

A mi primer tutor José de Sousa, por aceptarme como uno de sus seminaristas.

A mi guía y tutor actual Andrés Sanoja, por trabajar de una manera dura y exigente con nosotros con miras a lograr el trabajo de grado. Mi agradecimiento eterno para él.

A los profesores Robinson Rivas, Marcos Raydán, Eugenio Scalisse, Antonio Leal, Radamés Carmona, Adonahis Alvares, Eleonora Acosta, Paola Saputelli, que actuaron como padres o guías. A Wilfredo Ledesma, Marcos Gollarza y Norelva del niño, por darme su apoyo y brindarme un espacio en los laboratorios de Mefis y el de Sistemas de Información, cuando necesité con urgencia trabajar en este trabajo un equipo de computación.

A Jonathan Ramos, por pelear conmigo cuando había desacuerdos, ser mi compañero de batalla y de lucha en esta tesis.

A mis compañeros-hermanos: Andrés Pedroza, Lenín Torcatt, Antonio Ferrer, Francisco Pantó, Rommel Torres, Luisana Villaroel, Marbett Hedderich., Oscar Cahueñas, por tratarme como a un hermano, dejarme usar sus casas, sus equipos, apoyarme con el problema de mi madre y auxiliarme cuando lo necesité.

A otros compañeros de Estudios y vecinos como Rufino Uribe, Tyrone Valera, Yeslimar Martínez, Jesús infante, Samuel Martínez, Doris Martínez y su familia, Williams Rivas, Marina Valencia y su familia, por estar conmigo en momentos malo y buenos y también por apoyarme cuando tuve el percance con mi madre.

A otros compañeros de pasillo de la Facultad como Alejandro Trejo, Francisco Núñez, Lara Elvys, Jose Osuna, Gabriela Gruber, Neldy Corredor, Ronald, Angela Proaño, Oscar Ruiz, con los cuales estudié y compartí vivencias. A Trino, y a las Secretarias de la Escuela de Computación, por ser mis panas y auxiliarme cuando tenía que imprimir o se me presentaba un problema menor.

Jonathan:

Primero quiero dedicar éste Trabajo Especial de Grado a Dios y a mis padres quienes no imaginan cómo los Amo. A mi madre María Quiñonez por ser el más grande ejemplo de constancia y dedicación que he visto, espero algún día representar tu fuerza y valentía. Mi padre Gerardo Ramos que con el pasar de los años me haz enseñado a reflexionar sobre la vida, sus cosas buenas y sus cosas malas.

A mis hermanos Davidson, Joham y Harley quienes siempre me han brindado su amor y apoyo incondicional.

A mi bella chica Rosmarlyn, siempre haz sido muy importante para mí, gracias por escucharme y darme tu energía, te amo mi princesa.

A todos mis más grandes amigos, Paul, Eunice, Fiorella, Ricardo, Jonathan, Miguel, David, Antonio, Armando, Bárbara y Armandito.

Al pana marcos que dejó de estar con nosotros y nunca tuvimos la satisfacción de verte graduado, se te recuerda con mucho cariño. Este triunfo lo comparto contigo.

Para terminar, mis mas sinceros agradecimientos a los profesores José de Sousa, Eleonora Acosta, Eugenio Scalisse y en especial a nuestro tutor y amigo Andrés Sanoja. Sin ustedes éste sueño nunca habría sido realidad. Gracias, nunca les olvidaré.

Índice de contenidos

Índice de tablas y figuras

Introducción

En la actualidad están disponibles una numerosa gama de marcos de trabajos o frameworks que permiten implementar proyectos o programas bajo la plataforma Web de Java. Cada uno con funciones y características bien específicas creados con la finalidad de poder construir sistemas o aplicaciones bien robustas con poco esfuerzo. Creando una capa de abstracción compleja, en la construcción de aplicaciones Web pero mucho mas fácil de manejar, sobre la cual resulte mas sencilla el desarrollo de nuevos proyectos o programas de aplicación. Esto amerita que las personas que desarrollen estas aplicaciones tengan un grado de conocimiento razonablemente avanzado y de maduración que les permita manejar la complejidad de las mismas, utilidad no presente en desarrolladores novatos que deben adicionar esfuerzo y empaparse de éstos nuevos conocimientos para conocer y manipular de buena y óptima manera la abstracción, complejidad y características presentes en los entornos u ambientes de tales frameworks, con el fin de aprovechar los beneficios, rasgos y atributos que en ellos se es capaz de configurar.

Es importante resaltar que a medida que se analizan los objetos del mundo real éstos se traducen progresivamente en modelos cada vez más complejos. En un dominio cualquiera tratar de representar todos los detalles de los objetos presentes por extensión, puede ser una tarea difícil, o bien poco deseable. Esto conlleva a la construcción de componentes genéricos, reutilizables que pueden ser no tan sencillos de aprender, pero que cuando se hace un esfuerzo para comprenderlos, se hace mucho más llevadero el desarrollo de un proyecto de software. Estos componentes genéricos pueden encontrarse en los ambientes involucrados con los frameworks, en particular todo los relacionados con desarrollo de entornos Web.

Un framework o marco de trabajo puede conceptualizarse como una aplicación genérica incompleta y configurable a la que se puede agregar algunas piezas para construir una aplicación concreta [23]. En este Trabajo Especial de Grado se contemplan dos marcos de trabajos o frameworks de desarrollo. El primero de los frameworks, Struts2 tiene como característica principal un controlador que viene como un Servlet interno de Struts, el cual gobierna, acopla y establece las reglas para conducir las acciones entre la interfaz y el modelo del negocio, esta última trabajada a través de Java Beans y la parte de la vista que es construida por medio de etiquetas de Struts2 que agilizan y facilitan el control y la construcción de ciertas partes de la lógica de negocio, ayudando al diseño de la interfaz de una manera práctica, rápida y entendible al programador.

El segundo framework, Java Server Faces maneja la interfaz gráfica de usuarios, los eventos y sus componentes como sí se tratara de una aplicación de escritorio. Presenta como aspecto fundamental descriptivo la separación entre el comportamiento y la presentación, permitiendo al usuario que utilice el framework para la separación de los roles, al restar la complejidad cuando se implementan interfaces de usuarios.

Sobre ambos frameworks es posible obtener indicadores de rendimiento y medida de la calidad del código escrito al desarrollar una aplicación Web. ¿Pero cuál de estos dos frameworks es más útil?, ¿Cuál es más fácil de manejar? O más aún ¿Cuál framework puede tener mejor desempeño?

Es por eso que éste Trabajo Especial de Grado se presenta una sinopsis detallada de algunas de las características de los frameworks seleccionados, tomando como base el diseño de una herramienta de administración de la información del Servicio Comunitario de la facultad de Ciencias de la Universidad Central de Venezuela. Sobre la cual es posible medir en igualdad de condiciones y bajo un mismo contexto, la calidad y el desempeño de funcionalidades bien específicas implementadas en ambos marcos de trabajo.

El presente Trabajo Especial de Grado se encuentra estructurado de la siguiente manera:

Capítulo I. Problema y Metodología de la investigación: en el cual se realiza la descripción del problema, los antecedentes, los objetivos, la justificación y el alcance del trabajo.

Capítulo II. Marco Conceptual: en donde se expone la teoría de los elementos esenciales manejados en el trabajo, como lo son las métricas de software y rendimiento, así como las características más resaltantes de los frameworks Java Server Faces y Struts2.

Capítulo III. Marco aplicativo: ésta sección del documento expone los elementos configurados en el proceso de desarrollo utilizado, así como el conjunto de herramientas tecnológicas que fueron necesarias, para cumplir con los objetivos propuestos.

Capítulo IV. Casos de estudio: comprende la descripción detallada de los componentes de software seleccionados (Vistas, Java Beans) sobre los cuales fueron llevados a cabo los cálculos de métricas de software y rendimiento.

Capítulo V. Resultados y Conclusiones: La sección de resultados muestra de forma gráfica los resultados obtenidos durante las mediciones realizadas a los componentes descritos en el Capítulo IV, así como también las conclusiones a las cuales se llegaron en el presente Trabajo Especial de Grado.

Capítulo I: Problema y Metodología de la investigación

Éste primer capítulo se realiza la descripción del problema planteado junto a las preguntas de investigación a resolver. Se exponen los objetivos del presente trabajo y se justifica su desarrollo, estableciendo el alcance, antecedentes y especificando el proceso de desarrollo utilizado.

1.1 Planteamiento del problema

Actualmente es amplia la gama de productos que se encuentran disponibles en el mercado para la construcción de sistemas basados en Web y que siguen el patrón de diseño MVC, tal es el caso de los frameworks Struts 2 y Java Server Faces.

Ambos frameworks están centrados en trabajar la construcción de la capa de presentación de aplicaciones Web basadas en plataforma Java, pero implementando su solución de formas diferentes para resolver el problema.

Por un lado se tiene el framework Struts 2, cuyo valor agregado es la experiencia obtenida de su predecesor Struts y su combinación de otro framework llamado WebWork, que unieron esfuerzos y conocimientos para dar paso a la nueva versión de Struts. Por otro lado se cuenta con el framework Java Server Faces, relativamente reciente que afronta el reto de lograr el mismo trabajo bajo una filosofía totalmente distinta a la de Struts y que centra la posibilidad de trabajar la programación en un ambiente Web enfocado desde el punto de vista de una aplicación de escritorio (componentes y eventos), muy convencionales y análogas a una aplicación Swing de Java.

Por tal motivo es posible la elección de cualquiera de los frameworks para el tratamiento de la capa de presentación en la construcción de sistemas Web, sin embargo en nuestra investigación no encontramos una opinión fundamentada o fuertemente razonada acerca de cual de los frameworks es más apropiado para el desarrollo de un determinado proyecto de software.

1.2 Preguntas de investigación a resolver

¿Cuál de los frameworks de la capa de presentación es más eficaz para la implementación de un proyecto, basándose en la evaluación de métricas de software durante el desarrollo y la experiencia del grupo de trabajo?

¿Cuál de los frameworks de la capa de presentación ofrece más utilidades y características ventajosas? y ¿Cuál puede ser el más apropiado para afrontar el desarrollo de una aplicación Web bajo la plataforma Java?

1.3 Objetivo General

Diseñar una herramienta Web en plataforma Java, implementando una funcionalidad bajo los frameworks Struts 2 y Java Server Faces que sirva para la evaluación de métricas de software y de rendimiento en igual cantidad y denominación; siguiendo un mismo proceso de desarrollo y usando la misma estructura lógica del modelo de negocios.

1.4 Objetivos Específicos

- Diseñar una herramienta para administración de la información del Servicio Comunitario sobre la cual realizar una selección de componentes, que se implementarán bajo los frameworks Struts 2 y Java Server Faces a los cuales se les realizará el cálculo de métricas para su posterior análisis

- Identificar los componentes similares existentes en los frameworks Struts 2 y Java Server Faces a partir de los cuales establecer los casos de estudio

- Estructurar un ambiente de trabajo en el entorno de desarrollo Eclipse con lo cual poder llevar a cabo el proceso de medición de los componentes seleccionados

- Estructurar un ambiente de trabajo bajo plataforma Linux sobre el cual llevar a cabo el procedimiento de medición de rendimiento sobre los componentes seleccionados

- Construir una herramienta que permita la lectura de los datos de las mediciones del software realizadas (Componentes de la herramienta de servicio comunitario) para su transformación en un formato óptimo para su mejor comprensión

1.6 Justificación y Antecedentes

Utilizar un framework es por lo general una buena fuente de ayuda para optimizar el desarrollo de un producto de software. Los desarrolladores web apoyan la construcción de sus aplicaciones en la reutilización de estas tecnologías, con la finalidad de reducir tiempos de desarrollo y mantenimiento de software.

Datos sobre las cualidades y rendimiento de un framework son información importante para los desarrolladores. Les permite determinar a priori cual puede desempeñarse en un caso de estudio, ahorrando tiempo y esfuerzo.

Aun cuando existen otros trabajos relacionados con este tópico, los mismos tienen un enfoque con fines comerciales y presentan una visión subjetiva sobre cada uno de los frameworks estudiados. Es importante destacar el trabajo presentado en el TEG de la Lic. Zulma Gonzalez [6] donde se evalúan métricas de software y de rendimiento sobre los frameworks java: Struts 2, Hibernate, y Javamail, así como el framework Ruby-on-Rails. Por otro lado, se tienen trabajos sobre comparación de frameworks Java Server Faces y Struts 2, enfocados a realzar de manera individual las características de los mismos. Por ejemplo los presentados en [24, 25, 26, 27].

En base a lo mencionado anteriormente surge la necesidad de comparar los frameworks y exponer ante los desarrolladores las virtudes y debilidades de cada uno, de la manera más objetiva posible. Forjando las bases sobre la cual justificar la implementación de alguno de ellos para la creación de un producto informático.

Por tal motivo este Trabajo Especial de Grado queda justificado, ya que se enfoca en evaluar rasgos y funcionalidades implementadas en igualdad de condiciones de dos frameworks, en base a un caso de estudio utilizando métricas de software y pruebas de rendimiento.

1.7 Alcance

Cabe destacar que el objetivo esencial del Trabajo Especial de Grado no abarca solamente el desarrollo de una herramienta o aplicación para apoyar al Servicio Comunitario de la Facultad de Ciencias de la UCV, incluye también evaluación de métricas de software y de rendimiento. Para lograr la meta de medición de componentes de tecnologías distintas, se hizo necesaria la construcción de las siguientes aplicaciones:

- **Módulo de servicio comunitario:** Análisis y desarrollo de un módulo para la administración de información del servicio comunitario de la Facultad de Ciencias, sobre el cual sea posible realizar mediciones y cálculos de las métricas seleccionadas en igual número y con igual funcionalidad en ambos frameworks de desarrollo

- **Módulo visualizador de métricas:** Implementación de un analizador de datos de métricas, que pueda convertir los datos contenidos en los archivos generados por un plugin de captura de métricas de software instalado en el entorno de desarrollo Eclipse y la herramienta de cálculo de stress, para su posterior conversión y tratamiento a un formato legible y de fácil compresión para el análisis comparativo de ambas tecnologías usadas en la implementación de la herramienta de servicio comunitario

1.8 Proceso de desarrollo utilizado

"Un proceso de desarrollo de software es un método para organizar las actividades relacionadas con la creación, presentación y mantenimiento de los sistemas de software" [2]. Inicialmente la idea era utilizar el Proceso Unificado Racional (RUP) como proceso de desarrollo en nuestro trabajo. Por otro lado también se tenía contemplado utilizar un proceso bajo un enfoque ágil para ajustarse a las necesidades del equipo de desarrollo, ya que debido a la característica del equipo de trabajo no nos sería posible cumplir con el conjunto de artefactos y entregables que suponía el trabajo con RUP

"El modelado ágil es la presentación de un enfoque para realizar el modelado (diseño) de sistemas de software el cual está basado en un conjunto de valores, principios y buenas prácticas que pueden ser aplicados como complemento a cualquier proceso de desarrollo de un proyecto de software" [3]. Además que brinda libertades al grupo de desarrollo en cuanto a la selección del conjunto de artefactos que son considerados como críticos en un proyecto.

Todo esto motivó a la selección del proceso UP Ágil, que brinda un enfoque ágil aplicado a las disciplinas y a los artefactos establecidos por el Proceso Unificado Racional (RUP) para el desarrollo de software, proceso conocido como (UPA).

Capítulo II: Marco Conceptual

En éste capítulo se realiza un resumen teórico de los elementos manejados en el presente trabajo. Primero se describen las métricas de software y rendimiento, seguido de los frameworks de desarrollo Java Server Faces y Struts2.

2.1 Métricas

El cálculo de métricas es utilizada para tener una certeza de medida, un grado o valor o patrón que da un tamaño de algún objeto, dato, proceso o ente en un caso de estudio, la misma permite valorar la calidad de los productos de ingeniería o los sistemas que se construyen; ésta a su vez proporciona una manera sistemática de valorar la cualidad o calidad basándose en un conjunto de pasos, escala, parámetros, normas o reglas claramente especificadas y bien definidas.

Ejemplos de estas cualidades son medida de la velocidad de un proceso de software o de la aplicación desarrollada, rendimiento en el manejo de recursos, tiempo de respuesta, entre otros. Su uso es de real importancia motivado a que las mismas permiten descubrir y corregir fallas que se presentan cuando modelamos un problema del mundo real. Utilizando los conceptos anteriormente mencionados como indicadores de medida, es posible obtener la calidad del producto informático que se ha desarrollado o se está elaborando.

En términos de beneficios, la medición por sí misma no mejora los procesos, pero su conocimiento permite profundizar en la planificación, control, gestión y mejora. Además de comunicar una medida del progreso y soportar la toma de alguna decisión. Más en detalle estos beneficios pueden ser los siguientes:

- Mejor calidad del producto elaborado
- Es posible incrementar la productividad del equipo de desarrollo
- Se puede traducir en una mejor estimación y planificación del proyecto
- Crea una cultura de calidad del software
- Permite identificar la funcionalidad de los componentes del software

2.1.1 Métricas seleccionadas para aplicación en el proyecto

Previamente en la fase de investigación o seminario se cubrió el concepto de un buen conjunto de métricas, tanto de medición del software como de conceptos involucrados con el rendimiento del mismo.

Una vez establecidos los parámetros y alcances del trabajo se ha seleccionado un conjunto limitado de métricas las cuales serán suficientes para llevar a cabo los análisis detallados, mediciones y comparaciones de módulos, atributos o programas previamente seleccionados en los entornos de los frameworks Struts2 y Java Server Faces. Estas métricas son las que se describen a continuación entre los puntos 2.1.2 y 2.1.5.

2.1.2 Métricas de código fuente

Este tipo de métricas se enfoca en medir las características de mantenimiento del código fuente y son aplicadas sin tomar en cuenta la metodología de programación empleada, éstas son [4]:

- **Total Líneas de Código** (LOC): Totaliza todas las líneas del programa sin tener en cuentan si el mismo contiene código, comentarios o espacios en blanco.

- **Líneas de Comentarios de Código** (CLOC): Totaliza las líneas de comentarios del código.

- **Densidad de Comentarios** (DC): Define el valor de densidad de las líneas comentadas del código, el cual se obtiene de la siguiente manera:

  DC = CLOC / LOC

- **Número de Parámetros** (NP): Indica el número de parámetros presentes en una función o procedimiento. Un gran número de parámetros se traduce en funciones que son complicadas y difíciles de reutilizar porque tienden a ser especializadas.

2.1.3 Métricas para medir la complejidad

Las métricas de complejidad, tratan de evaluar los niveles óptimos con respecto a la característica de software. "pueden emplearse para predecir la información crítica sobre la fiabilidad y mantenimiento de sistemas, software de análisis automáticos de código fuente (o información de diseño de procedimiento)". En este caso se seleccionó la siguiente métrica [5].

- **Complejidad Ciclomática** (Cyclomatic Complexity): Esta métrica, propuesta por Thomas McCabe en 1976, se basa en el diagrama de flujo determinado por las estructuras de control dentro del un código fuente de un programa.

  La complejidad ciclomática calcula el número de caminos de un programa. Se usa para predecir los módulos que son propensos a errores, el esfuerzo en mantenimiento de los mismos y la reutilización de sus módulos.

  La representación de está métrica se basa en el flujo del código fuente del programa también conocida como grafo de flujo, que es una representación compuesta por nodos y enlaces (definidos como aristas).

  La figura 1, muestra la representación de un diagrama de flujo, de un fragmento del código fuente de un programa y su respectivo grafo de flujo asociado. La complejidad del código es obtenida mediante el número ciclomático, calculado por la función: $V(G) = e - n + 2$, donde, e representa el número de arcos y n el número de nodos del Grafo de flujo. [5]

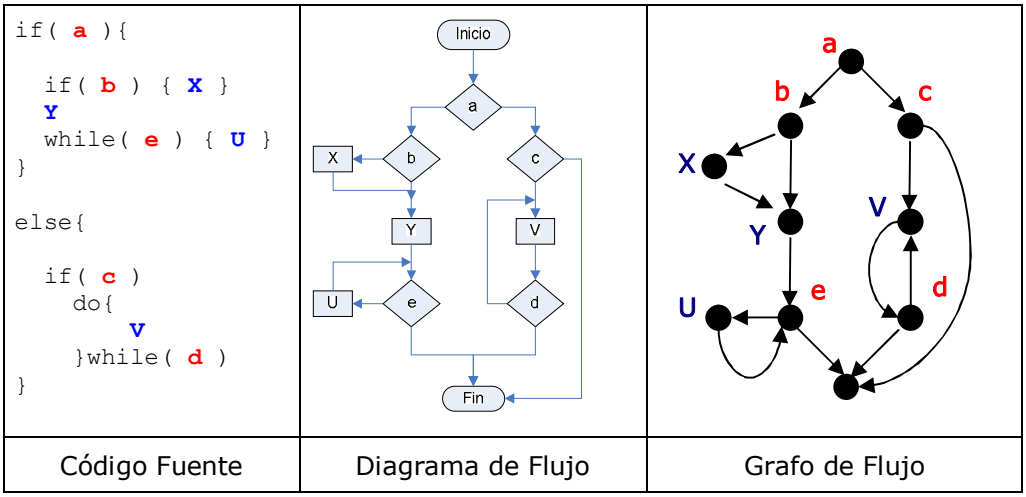

Figura 1: Ejemplo de un Diagrama de flujo con su Grafo de flujo

Una vez calculada la complejidad ciclomática, se puede determinar el riesgo que supone el código utilizando los rangos definidos en la tabla 1:

V(G)	Evaluación del Riesgo
1-10	Programa Simple, sin mucho riesgo (valores ideales)
11-20	Programa Complejo, con riesgo moderado
21-50	Programa Complejo, programa con alto riesgo
> 50	Programa Inestable, riesgo muy alto

Tabla 1: Evaluación de riesgos de la complejidad ciclomática

2.1.4 Métricas para desarrollo de software orientado a objetos

Este tipo de métricas están enfocadas a obtener la ponderación que puede aplicarse a las clases y a las características del diseño (tales como ocultamiento de información, localización, herencia y las técnicas de abstracción de objetos), que definen unívocamente a la clase. Las métricas calculadas son:

- **Falta de Cohesión en los Métodos** (LCOM): Esta métrica indica el grado de cohesión existente entre los métodos que se han definido en una clase. Por medio de la misma se determina a la cohesión como el grado en el cual los métodos en una clase que están relacionados con métodos de otra clase para trabajar juntos y proveer un conjunto de comportamientos semejantes. [6]

- **Acoplamiento hacia el Interior** (CA): Mide el número de clases que se encuentran fuera de un paquete, que dependen de las clases que se encuentran definidas dentro del mismo [6]. Es decir, determina la dependencia entre un paquete y otro.

Para mejorar modularidad y promover el encapsulamiento, este valor no debe ser muy alto. Los valores adecuados de esta métrica para un paquete están comprendidos entre 0 y 500.

- **Acoplamiento hacia el Exterior** (CE): Mide el número de clases dentro de paquete que dependen de clases fuera del mismo [6]. Los valores adecuados de CE para un paquete están comprendidos entre 0 y 20.

- **Inestabilidad** (I): Esta métrica fue propuesta por Robert C. Martin y mide la inestabilidad de los paquetes; donde la inestabilidad es medida calculando el esfuerzo de cambiar un paquete sin afectar a los otros paquetes. El número de dependencias entrantes y salientes de un paquete es un indicador que determina la estabilidad del mismo. El cálculo de inestabilidad se define de la manera siguiente:

  $$I = CE / (CA + CE)$$

  El rango de la inestabilidad es [0,1], donde 1 es el mayor valor de inestabilidad y 0 indica la mayor estabilidad del paquete. Sin embargo es casi imposible que cualquier diseño sea completamente estable o inestable, por lo cual el rango comprendido entre [0.0, 0.3] indica estabilidad, mientras que el rango entre [0.7, 1.0] indica inestabilidad del paquete. [7]

- **Profundidad de la Herencia** (Depth of Inheritance, DIT):
  La profundidad de la herencia se define como la máxima longitud del nodo a la raíz del árbol de herencia [7].

  $$DIT = | \{class\} - \{root\} |$$

  Por ejemplo en Java, como todas las clases heredan de la clase Object, por la tanto el mínimo DIT es 1.

  A medida que el DIT crece, es posible que clases de más bajos niveles hereden muchos métodos. Esto conlleva dificultades potenciales, cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases profunda también conduce a una complejidad de diseño mayor.

  Por el lado positivo, los valores DIT grandes implican un gran número de métodos que se reutilizarán. Mientras más grande sea el valor de DIT las subclases heredan muchos más métodos de las superclases. En algunas situaciones, llega a ser demasiado difícil desarrollar las superclases y las subclases, así que es recomendable mantener en el diseño un valor bajo de DIT. Los valores adecuados de DIT, aplicados para una clase se encuentran en el rango (0, 5) [8].

- **Número de Hijos** (NOC): El número de hijos es el número de subclases de una clase base o superclase. Las subclases inmediatamente subordinadas a una clase en la jerarquía de clases se denominan sus descendientes o hijos. Si se tiene un número muy grande de subclases de una clase base, esto puede ser candidato para que la refactorización cree una jerarquía más sostenible y conservable [4].

  A medida que el número de hijos crece, la reutilización se incrementa, pero además es cierto que la abstracción representada por la clase predecesora puede diluirse. Esto significa que existe una posibilidad de que algunos descendientes no sean miembros realmente apropiados de la clase predecesora o superclase. Los valores ideales de NOC para una clase están comprendidos entre 0 y 10.

- **Métodos Ponderados por Clase** (WMC): mide el esfuerzo que se requiere para implementar o dar mantenimiento a una determinada clase. Su cálculo se determina mediante la suma de los números ciclomáticos de todos los métodos definidos en la clase.

  WMC (C) = Σ V (m), para todo m

  Donde m Σ V (m) es el conjunto el cual contiene los métodos presentes en la clase C. Si en cada método el número ciclomático es igual a 1, entonces WMC queda igualado con el número de métodos de la clase [8].

  El número de métodos y su complejidad ciclomática son razonablemente evaluadores cualitativos del esfuerzo requerido para implementar y verificar una clase. Se infiere que mientras mayor sea el número de métodos, más complejo se presenta el árbol de herencia, ya que todas las subclases heredan métodos de sus padres originarios. Puntualizando, a medida que se aumenta el número de métodos para una clase en cuestión, es más probable en que se vuelvan más específicos de la aplicación, con lo cual se limita el potencial de reutilización de los mismos. Debido a todos estos argumentos, el valor de WMC debe estar tan bajo como sea razonablemente posible. Según Chidamber y Kemerer los valores usuales ideal para utilizar esta métrica en la implementación de una clase oscilan en el rango 1 y 50 [8].

- **Respuesta para una Clase** (Response For Class, RFC): Esta métrica queda definida con el número total de métodos que pueden ejecutarse en respuesta a un requerimiento de mensaje recibido por una clase. [4]. Se determina esta métrica incluyendo todos los métodos disponibles en la jerarquía de la clase, es decir:

  RFC = NLM + NRM donde:

  NLM: número de Métodos locales en la clase.
  NRM: número de Métodos remotos llamados por métodos locales.

  Si una clase puede ser habilitada para llamar a un gran número de métodos como respuesta a un mensaje, esta acción trae como consecuencia que se dificulte la fase de prueba para todos los posibles resultados, y por ende a medida que se eleva el valor de RFC el esfuerzo necesitado para la comprobación también aumenta, por esta razón el valor de esta métrica no debe ser muy alto. Según Chidamber y Kemerer los valores ideales de RFC para una clase oscilan entre [0,50] [8].

2.1.5 Métricas de rendimiento

Son las medidas que dan como respuesta una parte, un grado, una escala o un porcentaje de medida que involucra al rendimiento. Su objetivo es evaluar el comportamiento del software desde el un punto de vista de un cliente (externo a la aplicación) entre éstas se mencionan [4]:

- **Rendimiento:** Se mide en magnitud, teniendo en cuenta a los siguientes indicadores: la velocidad de procesamiento, el tiempo de respuesta, consumo de recursos, rendimiento efectivo total y eficacia [9].

  En otras palabras, "es el número de solicitudes que una aplicación Web puede atender por unidad de tiempo. Generalmente medido en solicitudes/segundo"

- **Tiempo de Respuesta:** Es el tiempo de espera que transcurre como respuesta inmediata al ejecutar una aplicación mediante la cual un cliente introduce un dato o solicitud requerida. En otras palabras es el "tiempo transcurrido entre la emisión de una solicitud y el primer byte devuelto al cliente desde el servidor" [9]

  Este es el aspecto que mejor puede percibir el usuario, el tiempo de respuesta puede variar independientemente de la tasa de rendimiento

- **Tiempo de Ejecución:** Se define como el tiempo en que tarda en ejecutarse completamente una aplicación o proceso. En otras palabras es el "tiempo en procesarse una solicitud generalmente entre el primer y el último byte devuelto al cliente desde el servidor" [9]

2.2 Frameworks de desarrollo utilizados

En el presente trabajo se han seleccionado dos de las tecnologías más populares para el trabajo con la capa de presentación en el desarrollo de aplicaciones Web basadas en plataforma Java, Sobre las cuales fueron evaluadas usando técnicas de medición.

A continuación las secciones 2.3 y 2.4 describen las características más resaltantes de éstos frameworks de desarrollo seleccionados.

2.3 Framework de desarrollo Struts 2

Este framework es una implementación del modelo MVC (Modelo, Vista, Controlador), donde el controlador de la aplicación es un Servlet interno del framework Struts, éste controlador maneja la relación entre las acciones, la interfaz y los modelos. La parte de la vista se construye mediante etiquetas definidas por el framework que ayudan al diseño de la interfaz y de su interacción con ciertas partes de la lógica del negocio.

"Apache Struts 2 fue conocido originalmente como WebWork1[2]; después de trabajar independientemente por algunos años, las comunidades de WebWork y Struts se unieron aprovechando las ventajas de ambos frameworks para crear Struts 2" [11].

[1] WebWork2 es un framework Web basado en el modelo MVC, que implementa soluciones de IoC, interceptores y el lenguaje de expresión OGNL, con soporte para construcción de plantillas de interfaz de usuario como controles y temas. Se caracteriza por estar construido con interfaces en lugar de clases abstractas.

2.3.1 Arquitectura del Framework Struts 2

Como puede observase en la figura 2, Struts 2 sigue un patrón MVC, donde el controlador del framework se basa en un conjunto de tecnologías, como por ejemplo: Java filters, Java Beans, XML, entre otras.

El modelo se encuentra conformado por las acciones (actions). Opcionalmente el framework puede usar cualquier tecnología de acceso a datos, entre éstas se pueden mencionar JDBC, EJB, Hibernate, entre otras. En cuanto a las vistas es posible la integración con JSP, JTL, JSF, templates, PDF, XSLT y más.

Struts2 introduce un conjunto de características nuevas en esta segunda versión del framework, específicamente componentes tales como los interceptores, manejo de validaciones mediante anotaciones y un poderoso lenguaje OGNL[2] para acceso y búsqueda de objetos.

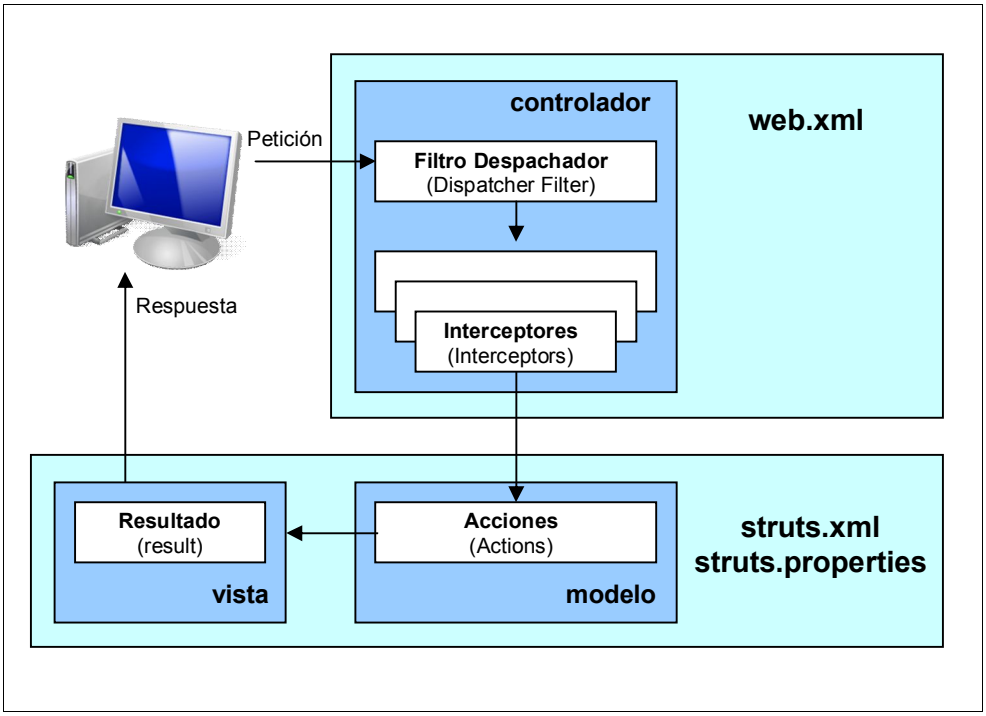

Figura 2: Arquitectura del framework Struts2

Los principales componentes del framework son los siguientes [12]:

- **Filtro Despachador (DispatcherFilter):** Es el punto de entrada del framework, ya que es el primer componente que interactúa en el procesamiento de una petición. A partir de éste se lanza la ejecución del procesamiento para cada petición involucrada con el marco de trabajo, es decir verifica que una petición invoca a

[2] Object-Graph Navigation Language (OGNL), creado por OGNL Technology, es un Lenguaje de Expresiones de código abierto para Java,el cual, mediante el uso de expresiones más simples que el amplio espectro que soporta Java, permite obtener y establecer propiedades (a través de métodos ya definidos getProperty y setProperty similares a los presentes en todos los JavaBeans) y la ejecución de métodos de clases Java.

alguna determinada acción dentro del framework. Además es responsable de comenzar la ejecución en cadena de los interceptores asociados a la petición y limpiar el contexto sobre el cual se ejecutan las acciones.

- **Interceptores (Interceptors):** son componentes que se ejecutan antes y después del procesamiento de una petición. Stuts2 permite utilizar interceptores para las Acciones (Actions), estos como su nombre indica, se encargan de interceptar la invocación a una acción. Los mismos son capaces de realizar operaciones antes y después de la invocación de una acción. Así mismo son capaces incluso de evitar que una acción se ejecute.

- **Acciones (Actions):** Una acción es la unidad básica de trabajo que se puede asociar a una petición del http que viene de un usuario. Las acciones o Actions son los encargados de ejecutar la lógica necesaria para manejar una determinada petición.

- **Resultados (Results):** Son objetos que encapsulan el resultado de una acción. Los Actions de la aplicación simplemente devuelven una cadena después de la ejecución de su o sus métodos, el cual será posteriormente usado para seleccionar un elemento de resultado.

  Un Action puede devolver diferentes tipos de resultados luego de su ejecución. Cada uno de estos resultados es un string que se encuentran previamente configurados en el framework. Una vez obtenido el string resultado, en conjunto con el nombre de la acción, se determina la vista que se retornará al cliente. Los string configurados son:

  ```
  String SUCCESS = "success";
  String NONE    = "none";
  String ERROR   = "error";
  String INPUT   = "input";
  String LOGIN   = "login";
  ```

  Adicionalmente es posible configurar nuevos tipos de resultados para las aplicaciones, en el caso de ser necesario.

2.3.2 Archivos de configuración de struts2

La configuración del framework puede llevarse a cabo a través de los archivos Web.xml, Struts.xml y Struts.properties que se muestran en la figura 2. En más detalle el uso de estos archivos son:

- **Web.xml:** es el descriptor de despliegue de la aplicación, éste se encuentra ubicado dentro del directorio WEB-INF. De forma general, este archivo posee toda la información que el servidor necesite conocer de la herramienta o aplicación para su funcionamiento. En él Struts define su Filtro Despachador (FilterDispatcher), Así mismo puede contener parámetros manejados dentro de la aplicación y la información de cualquiera del resto de los archivos XML necesarios para la configuración y comportamientos del framework.

- **Struts.xml:** Es el archivo de configuración base para el framework, este archivo debe estar ubicado en directorio de clases de la aplicación, generalmente en la ruta "/WEB-INF/classes". Es posible que no sea necesaria la existencia del archivo strust.xml para configurar la aplicación, ya que puede realizarse por completo a

través del archivo de configuración Web.xml. donde se identifica cómo se desarrollan las acciones.

- **Struts.properties:** este archivo es opcional y es un mecanismo para cambiar el comportamiento que maneja el framework por defecto [13]. Estas características pueden también ser configuradas en el archivo Web.xml, a través del uso de la etiqueta "init-init-param" o también mediante la etiqueta "constant" en el archivo Struts.xml.

2.4 Framework de desarrollo Java Server Faces

"La tecnología Java Server Faces (JSF) es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java" [14], basado en la arquitectura MVC, facilita y agiliza el desarrollo de aplicaciones Web. Haciendo que los programadores piensen en términos de componentes de interfaz gráfica de usuario, eventos y las interacciones entre sus componentes.

En este sentido se logra un estilo de programación similar a la programación de aplicaciones de escritorio. Una característica muy importante de JSF, es el hecho de ser una especificación creada por el Java Community Process (JCP)[3], transformándolo en un estándar, haciendo en consecuencia que todas las implementaciones, tanto de código fuente abierto como las comerciales deben respetarla.

2.4.1 Arquitectura del Framework Java Server Faces

Las aplicaciones Java Server Faces se ejecutan en un contenedor de Servlets de Java. La figura 3 muestra los principales componentes del framework Java Server Faces, que intervienen en la construcción de una aplicación Web JEE.

Una aplicación Java Server Faces se ejecuta en un servidor y está integrada con otros subsistemas, como EJB y bases de datos. La misma implementa un controlador central (FrontControler[4]) que se encarga del manejo de todas las peticiones provenientes desde los clientes y gestionar el ciclo de vida de cada petición. Este controlador central es un objeto Servlet que se conoce como Faces Servlet.

[3] **Java Community Process**: establecido en 1998, es un proceso formalizado el cual permite a las partes interesadas a involucrarse en la definición de futuras versiones y características de la plataforma Java. El proceso JCP conlleva el uso de Java Specification Request (JSR), las cuales son documentos formales que describen las especificaciones y tecnologías propuestas para que sean añadidas a la plataforma Java.

[4] **FrontController**: un objeto que acepta todos los requerimientos de un cliente y los direcciona a manejadores apropiados. El patrón Front Controller podría dividir la funcionalidad en dos diferentes objetos, el Front Controller y el Dispatcher. En ese caso, El Front Controller acepta todos los requerimientos de un cliente y realiza la autenticación, y el Dispatcher direcciona los requerimientos a manejadores apropiada.

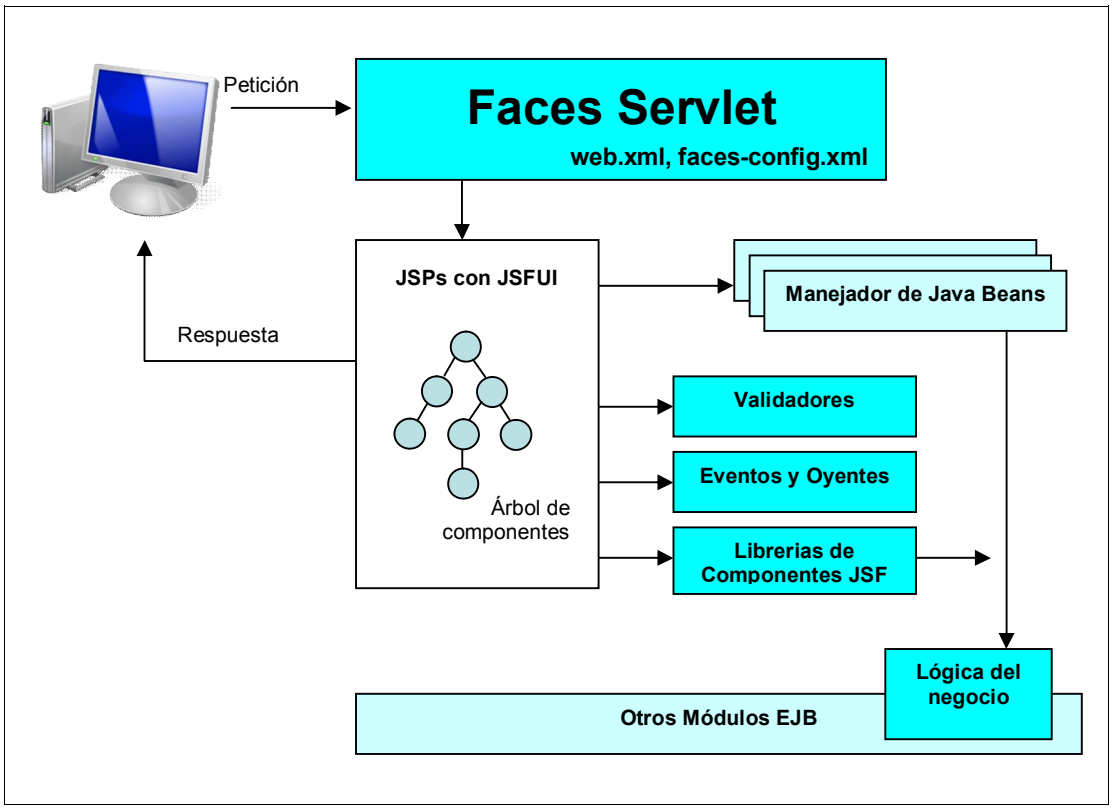

Figura 3: Arquitectura de componentes de una aplicación JSF

Las aplicaciones JSF están compuestas por:

- **Librerías de componentes JSF:** dos librerías básicas para la representación de componentes, manejadores de eventos, validadotes y otras acciones. Las cuales se incluyen en las páginas JSP, declaradas en su encabezado
- **Componentes Java Beans:** también llamados objetos del modelo en la tecnología JSF, Los Beans son clases que se utilizan para implementar cualquier comportamiento dinámico de la aplicación
- **Clases de utilidad del lado del servidor:** conformados por componentes tales como Beans para acceder a las bases de datos y clases que representan bibliotecas de etiquetas de usuario
- **Archivos de configuración:** contiene las reglas de navegación entre páginas, los validadores y los manejadores de Beans (configurados en el Faces-config.xml)
- **Descriptor de la aplicación (Web.xml):** contiene la configuración de la aplicación

2.4.2 Modelo de navegación Java Server Faces

Puede observarse que todas las aplicaciones Web están conformadas por un conjunto de páginas. Uno de los principales problemas de un desarrollador de aplicaciones Web es manejar la navegación entre esas páginas [15]. Con el modelo de navegación de Java Server Faces se facilita la definición de la navegación de páginas y el manejo de cualquier procesamiento adicional necesario para elegir la secuencia en se que enlazan y se cargan.

En muchos casos, no se requiere código para definir la navegación. En su lugar, la navegación se puede definir completamente en el archivo de configuración de la aplicación usando un conjunto de elementos XML.

Las reglas de navegación definen cuál es la siguiente página que se mostrará después de procesar una petición de un usuario. Esta navegación entre páginas se especifica en el archivo de con figuración faces-config.xml.

2.4.3 Características principales del Framework

Las características más resaltantes del framework JSF están definidas por su conjunto de modelos de componentes de interfaz de usuario y el conjunto de validadotes, conversores y eventos que le suelen ser asociados, en más detalle estos componentes se describen a continuación:

- **Modelo de componentes de la interfaz de usuario:** Las aplicaciones Java Server Faces se construyen a partir de componentes de interfaz de usuario. El Framework provee un conjunto de componentes básicos como botones, listas de selección, campos de entrada, entre otros. Estos componentes son orientados a eventos, los cuales son generados del lado del cliente y se procesan del lado del servidor. Además, los componentes proveen facilidades para validación de entradas de datos y conversión de objetos.

  Todas las clases de componentes de interfaz de usuario de Java Server Faces descienden de la clase UIComponentBase, que define el estado y el comportamiento por defecto de un UIComponent. El conjunto de clases de componentes de interfaz de usuario incluidos en la última versión de Java Server Faces son [15]:

  - **UICommand:** Representa un control que dispara actions cuando se activa
  - **UIForm:** Encapsula un grupo de controles que envían datos de la aplicación. Este componente es análogo a la etiqueta form de HTML
  - **UIGraphic:** Muestra una imagen
  - **UIInput:** Toma datos de entrada del usuario
  - **UIOutput:** Muestra la salida de datos en un página
  - **UIPanel:** Muestra una tabla
  - **UIParameter:** Representa la sustitución de parámetros
  - **UISelectItem:** Representa un sólo ítem de un conjunto de ítems
  - **UISelectItems:** Representa un conjunto completo de ítems
  - **UISelectBoolean:** Permite a un usuario seleccionar un valor booleano en un control, seleccionándolo o deseleccionándolo. Esta clase es una subclase de UIInput
  - **UISelectMany:** Permite al usuario seleccionar varios ítems de un grupo de ítems. Esta clase es una subclase de UIInput
  - **UISelectOne:** Permite al usuario seleccionar un ítem de un grupo de ítems. Esta clase es una subclase de UIInput

- **Validadores:** La tecnología Java Server Faces soporta un mecanismo para validar los datos locales de un componente justo antes de actualizar los datos del objeto modelo

  La mayoría de las etiquetas tienen un conjunto de atributos para configurar las propiedades del validador, como por ejemplo los valores máximo y mínimo permitidos para el dato de un componente o dato de entrada. Para este caso existe un atributo en todas las etiquetas de entrada llamado "required", que es del tipo booleano[5], y el cual define sí se debe asignar un validador que obligue a rellenar el campo

- **Convertidores:** Una aplicación Java Server Faces opcionalmente puede asociar un componente con datos del objeto del modelo del lado del servidor. Este objeto del modelo es un componente Java Beans que encapsula los datos de un conjunto de componentes

  Cuando un componente se une a un objeto modelo, la aplicación tiene dos vistas de los datos del componente: la vista modelo y la vista presentación, que representa los datos de un forma que el usuario pueda verlos y modificarlos

  La aplicación JSF debe asegurar que los datos del componente puedan ser convertidos entre la vista del modelo y la vista de presentación. Esta conversión normalmente la realiza automáticamente por el renderizador o constructor del componente

- **Eventos y Oyentes (Listeners):** la tecnología Java Server Faces define las clases Listener y Event que una aplicación puede utilizar para manejar eventos generados por componentes de interfaz de usuario

  Un objeto Event identifica al componente que lo generó y almacena información sobre el propio evento. Una aplicación debe proporcionar una implementación de la clase Listener y registrarla con el componente que genera el evento. Cuando el usuario activa un componente, como cuando pulsa un botón, se dispara un evento. Esto hace que la implementación de Java Server Faces invoque al método oyente que procesa el evento

  Todos los eventos heredan de FacesEvent. Una buena práctica, es que todas las clases de eventos terminen con el sufijo Event. Estos eventos pueden ser personalizados, cada evento debe definir una interfaz oyente personalizada. Dicha interfaz debe heredar de FacesListener. Java Server Faces define sólo dos tipos de evento y sus correspondientes oyentes:

  - **ActionEvent / ActionListener:** Son lanzados por componentes UICommand, que fueron activados por el usuario

  - **ValueChangedEvent / ValueChangeListener:** Son lanzados por componentes UIInput cuyo valor fue modificado

  Cada componente debe tener un método addNombreComponenteListener y otro removeNombreComponenteListener por cada interfaz oyente que pueda notificar.

5 Un atributo booleano es cualquier expresión que se le puede atribuir un valor verdadero o falso.

2.4.4 Archivos de configuración en Java Server Faces

Una aplicación JSF es configurada a través de los archivos de configuración: Web.xml y faces-config.xml, adicionalmente puede agregarse un archivo extra sobre el cual definir cadenas de caracteres, útil para lograr la internacionalización de la aplicación. En más detalle, estos archivos se explican brevemente a continuación:

- **Web.xml:** Es el archivo descriptor de la aplicación, y es similar para todas las aplicaciones JSF, allí se pueden añadir parámetros de contexto, directivas de seguridad, entre otros.

- **Faces-config.xml:** El archivo Faces-config.xml es el archivo de configuración central de Java Server Faces. En el mismo se define el modelo de navegación de la aplicación, el manejo de las clases Beans por Java Server Faces, entre otros.

- **Resource bundles:** en él se pueden definir cadenas de texto para múltiples idiomas útiles en la internacionalización de la aplicación Java Server Faces. Este archivo de configuración debe residir en el directorio classes del WEB-APPS.

Capítulo III: Marco aplicativo

El siguiente capítulo se describe el proceso de desarrollo configurado, la arquitectura del proyecto de medición, el conjunto de herramientas desarrolladas y software utilizado que fueron necesarios para el desarrollo del presente Trabajo Especial de Grado.

3.1 Personalización del proceso de desarrollo

Para llevar a cabo este trabajo se seleccionó como proceso de desarrollo a UP ágil (Proceso Unificado Ágil), Motivado a conservar las cuatro fases del Proceso Unificado pero enfocándonos de una forma ágil al escoger cuidadosamente los artefactos u objetos que son realmente necesarios para los fines del proyecto.

La tabla 2, muestra el resumen de las actividades, artefactos y entregables que fueron necesarios para el desarrollo del proyecto. En este caso se ha llevado a cabo la ejecución de las disciplinas en el orden apreciado en la tabla. Se consideró que para poder efectivamente cambiar de una fase a otra debían desarrollarse por completo cada uno de los artefactos o entregables especificados dentro de la fase.

Fase	Disciplina	Artefactos o Entregables
Inicio	Modelado (Modelo, Análisis y Diseño)	Servicio Comunitario: • Modelo de casos de uso • Modelo lógico de base de datos Visualizador de métricas: • Documento de requerimientos de medición • Arquitectura del proyecto
Elaboración	Implementación	Servicio Comunitario: • Modelo físico de base de datos • Prototipo de interfaz de usuario • Diagramas de clase de diseño
Construcción	Implementación	Servicio Comunitario: • Diagramas de componente (Beans) • Código fuente • Prototipo funcional de la aplicación Visualizador de métricas: • Guía de uso del plugin Metrics 1.3.6 y Httperf • Formato del archivo XML del esquema seleccionado • Código fuente del módulo visor de métricas
Transición	Despliegue	Servicio Comunitario: • Entrega del módulo de Servicio Comunitario • Manual de instalación • Guía del usuario

Tabla 2: Configuración del proceso de desarrollo

3.2 Módulo de Servicio Comunitario

El módulo de Servicio Comunitario es una herramienta desarrollada como parte del Trabajo Especial de Grado. Dispuesto para la extracción de componentes y cálculo de métricas, sobre un producto de software enfocado en modelar brindar una solución a una problemática real, motivo que justificó su construcción.

En éste módulo se contempló el modelado de administración de los siguientes procesos:

- Proyectos
- Comunidades
- Tutores Académicos
- Tutores Comunitarios
- Talleres
- Estudiantes
- Evaluaciones

El diseño de base de datos, casos de uso y capturas de pantalla pueden verse en el apéndice A.

3.3 Requerimientos de medición

La base del experimento es obtener datos y medidas en cuanto a la facilidad de programación y comportamiento de las tecnologías Java Server Faces y Struts2. Para cumplir tal fin se les hará una medición de métricas de software y rendimiento a componentes bien puntuales del Módulo de Administración del Servicio Comunitario especificada anteriormente.

La idea es involucrar los elementos esenciales y más resaltantes de los frameworks para los fines que fueron creados. Para éste caso, el término componente es usado para encapsular una función de la aplicación en la cual intervienen un conjunto de clases o Beans, librerías, vistas, componentes de las vistas y una correspondiente conexión a una base de datos que brinde de dinamismo a los componentes renderizados en las vistas mostradas al usuario.

La estrategia que se plantea es la medición de dos componentes desde dos puntos de vista, un primer punto de vista es la medición interna del componente. Dicho de otra manera la medición de las métricas de software del componente escrito en lenguaje de programación Java.

La segunda perspectiva, que es la percibida por los usuarios cuando a través de un navegador Web interactúan con el componente y donde interesa medir el rendimiento de la aplicación sometida a elevados niveles de stress (conexiones de clientes / usuarios).

Con lo anteriormente expuesto podemos decir que las pruebas de medición se realizan en bajo dos tipos de medición que son:

- La medición interna del componente a través de las métricas de software obtenidas desde el entorno de desarrollo Eclipse, gracias al plugin métrics 1.3.6.

- La medición externa del componente a través de la herramienta de software libre httperf, que simulará un conjunto de conexiones bajo una serie de parámetros no variables sobre una página Web, con la finalidad de someter a determinados niveles de stress el funcionamiento de la aplicación a través del framewok que maneja al componente

3.3.1 Estrategia para implementación del experimento

La estrategia a seguir para llevar a cabo el experimento, consiste principalmente en separar los componentes seleccionados del resto de la aplicación manteniendo la estructura de directorios y paquetes de la aplicación.

Posteriormente e individualmente son calculadas las métricas para cada uno de los componentes. La figura 5 muestra los pasos de la estrategia que se utilizó para llevar a cabo el experimento de medición. Estos pasos se describen a continuación:

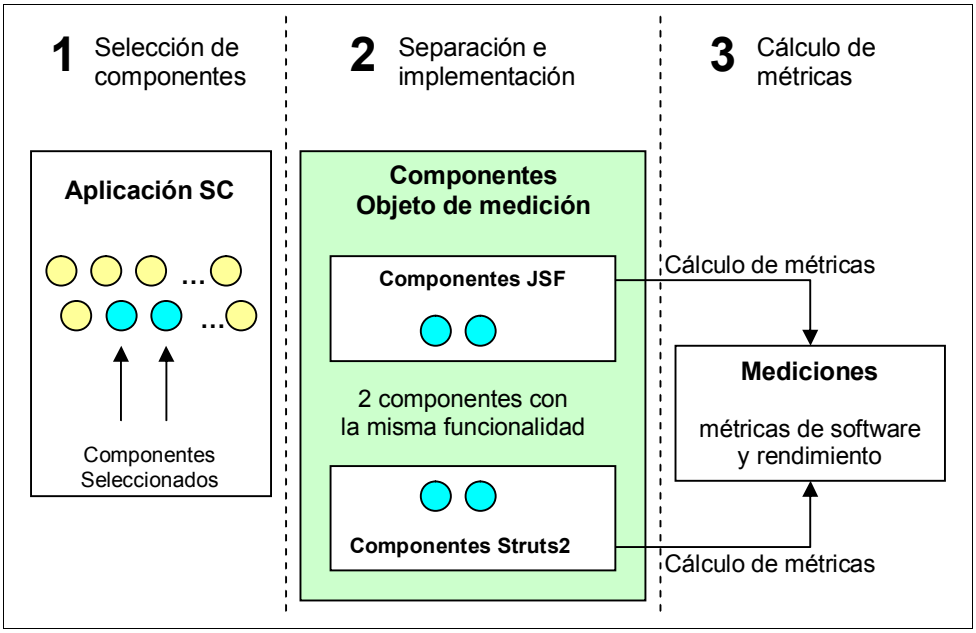

Figura 5: Pasos para la implementación del experimento

1. Selección de componentes: para el cálculo de métricas se han seleccionado como caso de pruebas dos componentes pertenecientes a la sección del sistema que involucra el manejo de la información de los estudiantes registrados. Más en específico estos componentes son:

- **Componente no1:** El primer componente seleccionado para el cálculo de métricas, es el formulario de modificación de los datos de un estudiante. El formulario seleccionado es uno de los más básicos de la aplicación.

  El criterio de selección de éste formulario estuvo orientado a evaluar el comportamiento del framework frente al renderizado de los componentes más simples de interfaz de usuario, tales como pueden ser un textfield, select o checkbox, entre otros.

- **Componente no2:** El segundo componente seleccionado para el cálculo de métricas, fue la página inicial de la sección de estudiantes dentro del Módulo de Administración del Servicio Comunitario.

En ésta vista se muestra el listado de los estudiantes registrados a través de una tabla de datos la cual es renderizada desde una colección de objetos o Beans que modelan a cada uno de los estudiantes registrados en el sistema. En éste caso se pretende evaluar el comportamiento del framework frente al renderizado de componentes complejos en la de interfaz de usuario.

Las tablas de datos son los elementos de interfaz de usuario más utilizados e incluso más pesados en cuanto a la cantidad de información manejada. En el caso del módulo desarrollado, estas representan un gran valor debido a que son utilizadas como el punto de partida para las operaciones sobre los datos visualizados en las mismas, operaciones tales como editar o eliminar, entre otras.

2. Separación e implementación: Los componentes de la herramienta del servicio comunitario, fueron separados del resto de la aplicación y se les realizó una pequeña modificación a nivel del código fuente, con la finalidad de crear la instancia del objeto de conexión a la base de datos, una vez se que se hiciera referencia a la ejecución del componente. Es decir simular que previamente un usuario realizó su autenticación en la aplicación y se inicializó su información para el libre uso de las opciones y operaciones presentes en la herramienta.

3. Cálculo de las métricas: es el paso más importante del experimento ya que es aquí donde se obtienen los resultados que serán posteriormente presentados. Estos resultados obtenidos dependen de los tipos de métricas calculadas, las cuales son:

- **Métricas de software:** el cálculo de estas métricas es obtenido directamente desde el entorno de desarrollo Eclipse, desde donde fueron escritos los componentes a medir

- **Métricas de rendimiento:** los datos de la medición de rendimiento son obtenidos a través de una herramienta de software libre desarrollada para sistemas linux, utilizada en la medición de funcionamiento de sitios Web bajo línea de comandos. La herramienta es capaz de simular múltiples sesiones y peticiones a determinados recursos contenidos en un sitio.

  Motivado a que los datos obtenidos después de la medición son un conjunto de valores totales (Tiempos de respuesta, Tiempo de conexión, Errores) se decidió realizar un conjunto de 22 experimentos, a su vez en cada uno de ellos se incrementó el numero de clientes realizando peticiones sobre el mismo recurso.

  Para ello fue necesario un shell script[6] que en cada ejecución, realizaba una llamada a la aplicación de medición de estadísticas de rendimiento bajo un conjunto fijo de parámetros previamente configurados dentro del script.

3.4 Eclipse Metrics 1.3.6

Un plugin o complemento es una aplicación que se relaciona con otra para aportar a esta última una nueva función y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.[7] [16]

[6] Un shell script es una serie de comandos escritos en un archivo de texto plano. como su nombre lo indica cada uno de estos comandos son interpretados y ejecutados en una consola o shell.

Metrics 1.3.6 es un plugin para el Entorno de desarrollo Eclipse, capaz de calcular varias métricas del código fuente durante la construcción de sistemas, y así proveer al desarrollador la forma de observar los problemas relacionados con valores fuera de rango para cada métrica. Permitiendo que constantemente se mantenga la correcta programación del código de la aplicación. La forma de uso de éste plugin puede observarse con detenimiento en el Apéndice B del presente trabajo.

Aunque el cálculo de las métricas puede ser útil en un esfuerzo de desarrollo de software, también es cierto que no tienen como objetivo sustituir la experiencia. Las métricas son más útiles como indicadoras de la calidad del software que se escribe, y por consiguiente, un código base con muchas advertencias de rangos de violación es probablemente una señal de que el código necesita ser refactorizado, lo cual necesariamente no indica que el código escrito es bueno.

En tal sentido, las métricas de software son un mecanismo útil para diagnosticar y darle más sentido y calidad al código que está siendo desarrollado por el programador o analista y en ello radica el beneficio de uso de este complemento (plugin). Por tal motivo se avala un estudio de uso de esta herramienta con la finalidad de adaptar su uso y de esta forma obtener el cálculo de las métricas, enfocadas a medir una aplicación propia o personalizada creada por un usuario particular.

3.4.1 Métricas calculadas por el plugin Metrics 1.3.6

A continuación se presenta una lista y una breve descripción de las métricas que pueden ser calculadas directamente por el plugin Metrics 1.3.6. Éstas son:

- **Líneas del código (LOC):** Líneas de código totales en la aplicación
- **Número de los métodos estáticos (NSM)**: Número total de métodos estáticos en la clase, presentes en una aplicación
- **Acoplador aferente (CA):** Número de clases fuera de un paquete que dependen de clases dentro del paquete el cual se está usando o al cual se hace referencia
- **Acoplador eferente (CE):** Número de clases dentro de un paquete que dependen de clases fuera del paquete
- **Abstractividad (RMA):** Número de clases abstractas dividido por el número total de tipos en un paquete
- **Distancia normalizada (RMD):** | RMA + RMI – 1 |, un valor muy cercano a cero indica el buen diseño de un paquete
- **Número de las clases (NOC):** Número total de clases en la aplicación o programa seleccionado
- **Índice de la especialización (SEISES):** Promedio del índice de la especialización, definido como NORMA * DIT/NOM

[7] Una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- **Inestabilidad (RMI):** CE/(CA + CE)
- **Número de los atributos (NOF):** Número total de atributos en la clase
- **Número de los paquetes (NOP):** Número total de paquetes en el alcance seleccionado
- **Líneas de código por método (MLOC):** Número total de líneas de código dentro de cuerpos del método, excepto líneas en blanco y comentarios
- **Métodos cargados por la clase (WMC):** Suma de la complejidad de ciclomática de McCabe para todos los métodos en una clase
- **Número de métodos principales (NORM):** Número total de métodos en una clase que se eliminan de una clase padre
- **Número de atributos estáticos (NSF):** número de atributos estáticos seleccionados en la clase de una aplicación
- **Profundidad de bloque jerarquizada (NBD):** La profundidad de bloques jerarquizados de código
- **Número de los métodos (NOM):** Número total de métodos definidos en la clase de una aplicación
- **Falta de cohesión de los métodos (LCOM):** Una medida para la cohesividad de una clase
- **Complejidad Ciclomática de McCabe (VG):** Cuenta el número de caminos que puede atravesar un fragmento de código. Esta métrica es calculada solamente para los métodos
- **Número de parámetros (NP):** Número total de parámetros en la clase de una aplicación
- **Número de los interfaces (NOI):** Número total de interfaces en la clase de una aplicación
- **Número de hijos (NSC):** Número total de subclases directas de una clase
- **Profundidad del árbol de la herencia (DIT):** Distancia del objeto de la clase en jerarquía de la herencia

A pesar de la gran variedad de métricas de software que es capaz de calcular este plugin, se han seleccionado sólo seis de las métricas anteriormente descritas como punto de comparación en el desarrollo del software del caso de estudio.

El criterio de selección estuvo centrado en la evaluar la sencilla estructura de los Beans que conformaron la lógica de negocios de aplicación, seleccionándose de ésta manera las métricas comunes que ayuden a la medición de pequeñas porciones del software escrito. Estas son:

- Número de las clases (NOC)
- Total de líneas de código (TLOC)

- Líneas de código por método (MLOC)
- Falta de cohesión de los métodos (LCOM)
- Complejidad Ciclomática de McCabe (VG)
- Métodos cargados por la clase (WMC)

Excluyéndose de esta manera las métricas relacionadas al calculo de jerarquías, paquetes, herencias, parámetros, atributos o interfaces.

Se incorpora la métrica de complejidad con la finalidad de determinar que el desarrollo de las interfaces sean simples, es decir no deben ser complejas; una interfaz no debe contener lógica de control. Se espera que V(G) tenga un valor entre 1 y 10.

3.4.2 Entorno de desarrollo al que se integra

Un IDE o Entorno de Desarrollo Integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica de usuario.

En el caso de construcción de aplicaciones Java, el IDE seleccionado para desarrollar el proyecto fue Eclipse, principalmente debido a que el paquete de métricas Metrics 1.3.6 fue desarrollado como un agregado para este entorno de desarrollo.

Una vez instalado y habilitada la ejecución del plugin, no es necesario modificar los valores predefinidos en el mismo, sin embargo los usuarios pueden reconfigurarlo modificado sólo las siguientes preferencias:

- Habilitar el cálculo de las métricas justo después de la compilación del código
- Establecer los colores de las métricas calculadas (valores dentro y fuera del rango), así como los colores de visualización de las gráficas de dependencia de paquetes
- Establecer los preferentes valores máximos para cada una de las métricas disponibles
- Seleccionar la estructura del formato XML en el caso de requerir exportar los datos obtenidos

3.4.3 Archivos de valores de métricas

Las métricas calculadas pueden exportarse a un archivo con formato XML. Sólo basta seleccionar la ruta o alcance (proyecto, paquete o clase) que se quiere mostrar u observar en la vista. Entonces se utiliza la barra de herramientas para seleccionar la función de la exportación.

Para exportar el archivo de métricas, sólo basta con hacer clic sobre el botón exportar XML, ubicado en el borde derecho del marco de la ventana visualizadora de métricas, el cuál abre una ventana donde se elige el directorio al cuál se guardará el archivo de generado. Posteriormente se hace clic sobre el botón guardar.

3.4.4 Esquemas de archivos de valores de métricas

El plugin Metrics 1.3.6 es capaz de exportar los resultados obtenidos de las métricas calculadas en un proyecto, paquete o archivo a un documento con formato XML.

El archivo resultado puede exportarse en dos esquemas diferentes los cuales son:

- **Esquema No 1:** es el archivo con la estructura más sencilla, el cual contiene organizados los resultados según las métricas que han sido calculadas por el plugin

- **Esquema No 2:** El segundo esquema supone una mayor complejidad, ya que ordena los resultados, en dos niveles, el primer nivel contiene los paquetes de la aplicación y las métricas asociadas a cada uno de los mismos, el segundo nivel contiene los archivos incluidos en cada uno de los paquetes y sobre los cuales son calculadas el resto de las métricas de software

A manera de ilustrar los esquemas anteriormente planteados, supongamos un proyecto organizado de la siguiente manera:

- Dos paquetes, llamados paquete1 y paquete2

- Tres archivos Java (Archivo1, Archivo2 y Archivo3), a los cuales se les cálculo las métricas de software

- Dentro del paquete1, se encuentran los archivos Java llamados Archivo1 y Archivo2

- Dentro del paquete2 se encuentra el tercer archivo Java llamado Archivo3

Al exportar los datos de las métricas calculadas, se puede generar un archivo XML con cualquiera de las estructuras de árbol, correspondiente a los esquemas número uno y número dos descritos anteriormente. Estos esquemas son ilustrados en la figura 8 como una estructura de árbol.

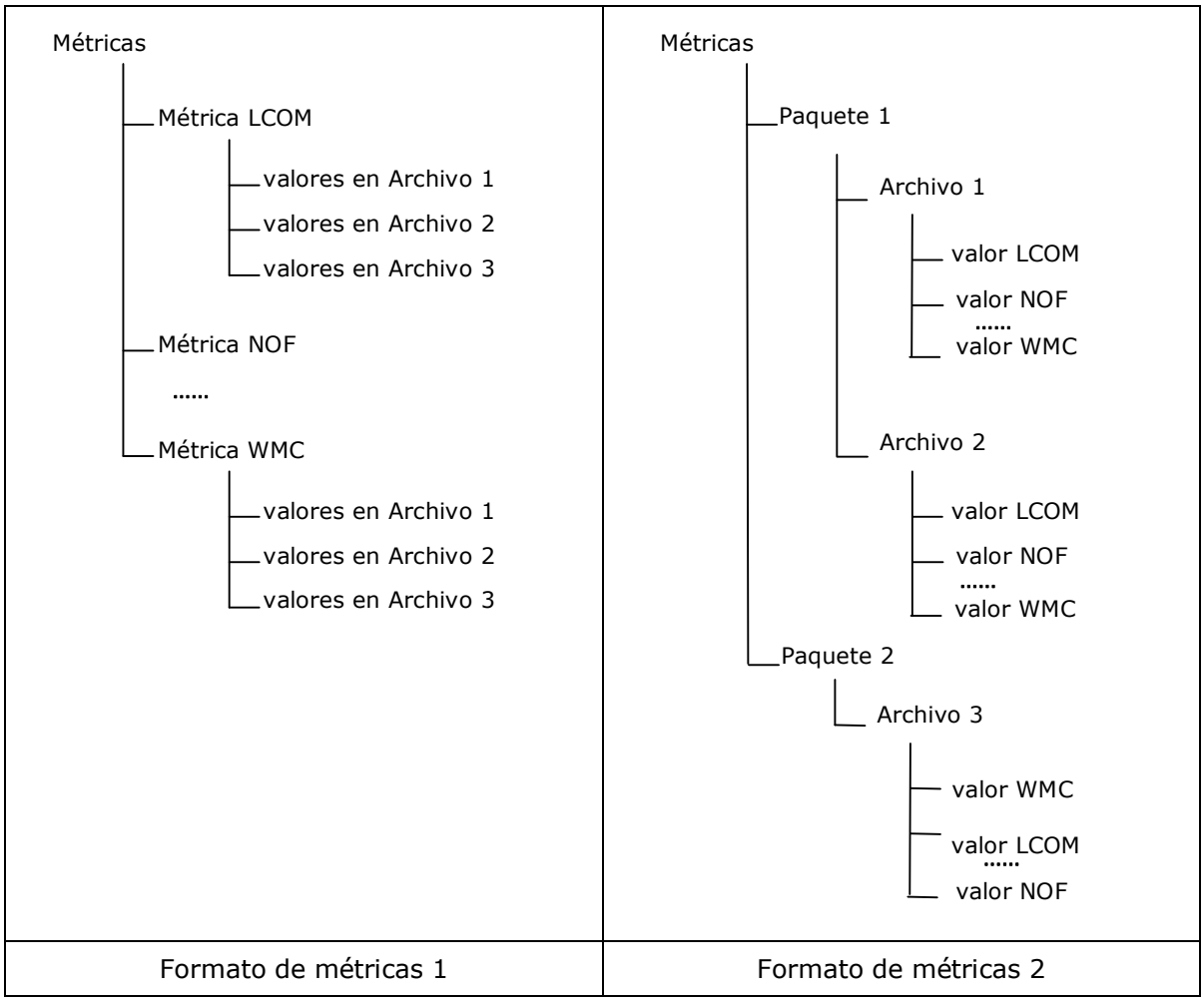

Figura 8: Estructura de los documentos XML generados por el plugin Metrics 1.3.6

3.5 Httperf

Httperf es una herramienta para medir el funcionamiento de un servidor Web. Proporcionando una manera fácil y flexible al generar varias cargas de trabajo con la cuál obtener una medición del funcionamiento del servidor. El foco de la herramienta no está en la ejecución de una prueba o patrón particular sino en el abastecimiento de una herramienta robusta, de alto rendimiento que facilite la construcción de las pruebas patrones micro y del macro nivel [20].

Httperf funciona bajo línea de comandos, en el Anexo I, se muestra una traducción al español de su manual, Cabe destacar que la traducción fue realizada para el presente trabajo motivado a que sólo existía en la red su versión en idioma inglés.

Los estadísticas generadas por el comando httperf fueron redireccionadas a archivos de texto para su posterior tratamiento en una herramienta de visualización. El Apéndice D describe en detalle la estructura de las estadísticas retornadas tras la ejecución del comando httperf.

3.6 Herramienta visualizadora de métricas

Con el fin de obtener una clara visión de los datos obtenidos, se implementó una herramienta Web en plataforma php5, la cual a transforma los archivos obtenidos en las mediciones realizadas a los componentes, a un formato adecuado para su presentación.

Este módulo visualizador está compuesto por las tres opciones que se describen a continuación:

1. **Métricas de software:** se encuentra enfocada a la visualización y graficación de las métricas de software calculadas por el plugin Metrics 1.3.6 del entorno de desarrollo Eclipse. En esta opción de despliega un formulario para la selección de dos archivos de métricas de software calculadas.

   Al seleccionar los archivos y hacer clic sobre el botón generar gráficos, la aplicación se encarga de subir los dos archivos de datos hacia el servidor para su tratamiento y desde allí realizar la respectiva graficación, tal como se muestra en la figura 9.

   Acompañada de cada gráfica se tiene la opción de ver en detalle los cálculos realizados por el plugin de Eclipse.

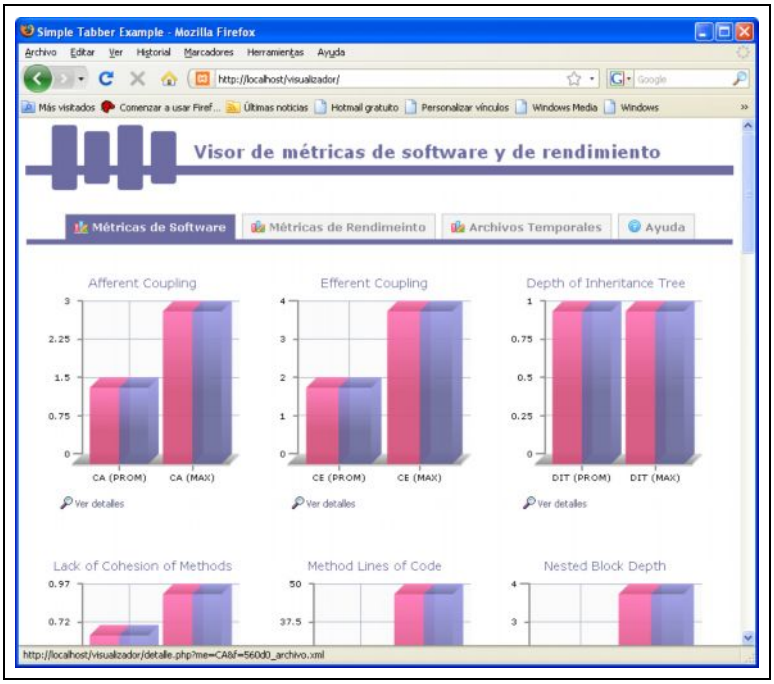

Figura 9: Visualizador: Sección de métricas de software

2. **Métricas de rendimiento:** esta opción está orientada a la lectura de los datos de rendimiento calculados por httperf. En esta opción se despliega un formulario para selección de un archivo de texto plano, que contiene los datos de las mediciones de rendimiento obtenidos por la herramienta. Posteriormente estos datos son transformados en información presentada con un formato sencillo para mayor compresión de los mismos. La figura 10 muestra como son graficados los datos de rendimiento en ésta sección.

Figura 10: Visualizador: Sección de métricas de rendimiento

3. **Archivos temporales:** para la lectura y manejo de los datos, se hace necesario subir los archivos hacia el servidor donde se encuentra alojada la aplicación de visualización de métricas, Motivado al problema de acumulación de archivos temporales del lado del servidor se hace necesaria esta opción, lo cual permite al usuario eliminar el conjunto de archivos temporales en un momento determinado.

3.7 Otras herramientas utilizadas

Esta sección describe el conjunto de aplicaciones complementarias que fueron utilizadas a lo largo del desarrollo del presente trabajo, estas herramientas son:

3.7.1 Embarcadero Estudio

Embarcadero ER/Studio, es una herramienta utilizada en el diseño bases de datos. "Los arquitectos de datos poseen la capacidad para analizar e implementar bases de datos de alta calidad que reflejen necesidades comerciales. El formato visual altamente legible optimiza la comunicación entre funciones de trabajo y unidades comerciales. Las capacidades de gestión de modelos envolventes y el soporte de integración de meta datos simplifican la construcción y el mantenimiento de modelos empresariales complejos" [17].

Embarcadero ER fue una herramienta muy útil durante el proceso de desarrollo del modelo de la base del Módulo de Servicio Comunitario. Al permitir desarrollar un modelo lógico que posteriormente fue transformado en un modelo físico, y de forma automática por la herramienta. A continuación la figura 11 muestra el ambiente de trabajo del Embarcadero Estudio ER.

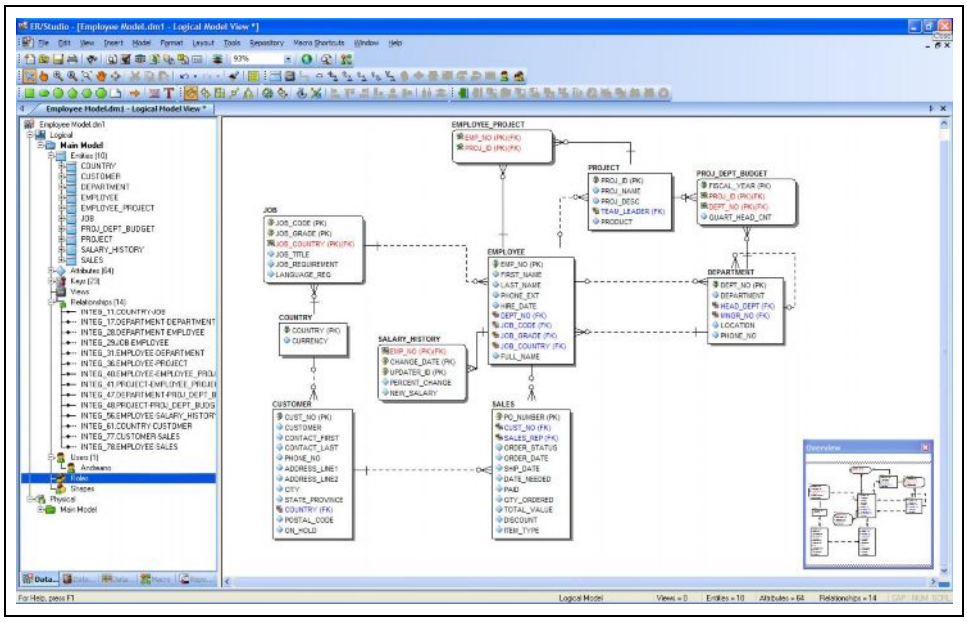

Figura 11: Ambiente de trabajo Embarcadero ER

3.7.2 ArgoUML

ArgoUML es una herramienta UML (Lenguaje de Modelado Unificado). Está escrito en lenguaje de programación Java y utiliza Java Web Start[8], con lo cual es fácil de operar y utilizar sobre cualquier plataforma. Posee soporte completo para UML 1.4 estándar y provee funciones para diseño y manipulación de diagramas en UML. Al igual que muchas otras herramientas de UML ofrece generación de código a partir de los modelos realizados.

Los siguientes lenguajes de programación son soportados para la generación de códigos: Java, C++, C# y PHP. El soporte de Java es mejor ya que puede ser usada la funcionalidad de ingeniería inversa. ArgoUML además posee soporte parcial para modelo de usuarios tales como modelos de decisión, modelo de objetivos, entre otros. Finalmente los diagramas pueden ser exportados en muchos formatos gráficos, inicialmente GIF, PNG, PS y SVG. [18]

ArgoUML fue una muy buena opción al momento del desarrollo de los modelos de caso de uso, del Módulo de Servicio Comunitario y el Módulo Visualizador de Métricas.

3.7.3 Entorno de desarrollo Eclipse

Eclipse es una plataforma de desarrollo de código abierto basada en Java, cuyo código fuente fue puesto a disposición de los usuarios. En si mismo Eclipse es un marco y un

[8] Java Web Start es un componente del entorno de ejecución de Java (JRE) que se instala con el JRE. Cuando se descarga por primera vez una aplicación que utiliza la tecnología Java Web Start, el software se ejecuta automáticamente y guarda la aplicación localmente en la memoria caché del equipo. De este modo, las subsiguientes ejecuciones son prácticamente instantáneas, Cada vez que se inicia la aplicación, se comprueba si en la sede Web de la aplicación hay una nueva versión disponible; en el caso de existir, se descarga y se ejecuta de forma automática.

conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (plugins).

Cuenta con un editor de texto donde puede verse el contenido de los archivos que se están trabajando, una lista de tareas, y otros módulos similares. Si bien las funciones de Eclipse son más bien de carácter general, las características del programa se pueden ampliar y mejorar mediante el uso de plugins. La instalación y ejecución es muy sencilla, sólo basta con descargar el archivo comprimidos de la última actualización del entorno de desarrollo Eclipse desde su página Web, una vez realizado este proceso, se descomprime en cualquier directorio en donde se ejecuta haciendo clic sobre el icono de aplicación Eclipse.

El IDE Eclipse se consideró la herramienta principal durante el desarrollo del presente trabajo, gracias a éste potente editor fue sencillo construir el Módulo de Servicio Comunitario y cada uno de los componentes que formaron los casos de estudio. La figura 12 muestra el entorno de desarrollo Eclipse.

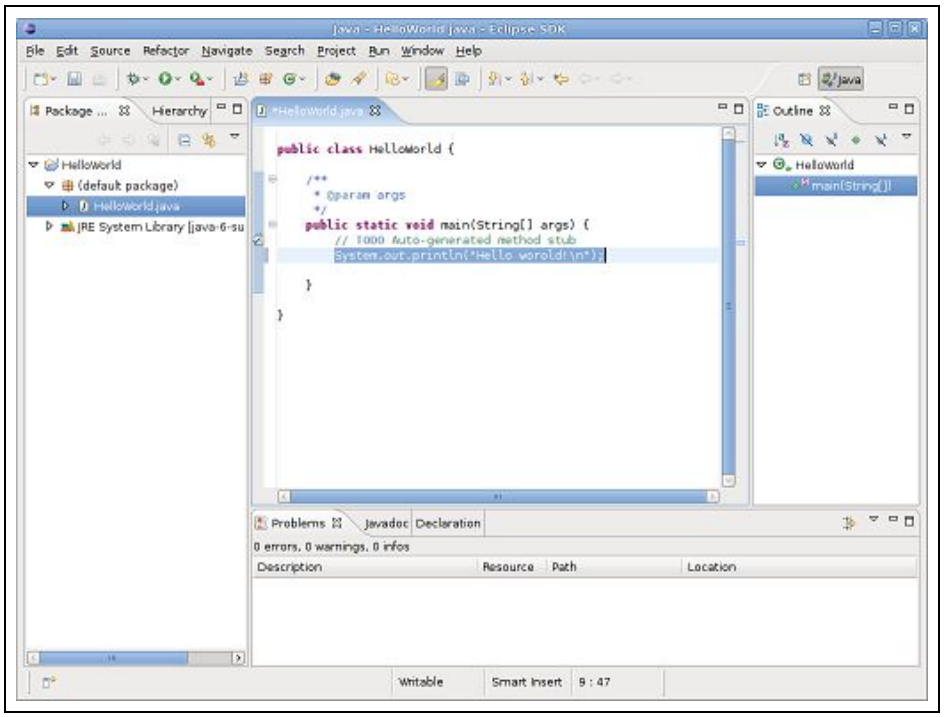

Figura 12: Ambiente de trabajo del entorno Eclipse

3.7.4 Entorno de diseño Dreamweaver

Dreamweaver es una herramienta avanzada para el diseño de páginas Web, el cual soporta una gran cantidad de tecnologías. Permite al usuario utilizar la mayoría de los navegadores Web instalados en su computadora para previsualizar las páginas Web diseñadas. También dispone de herramientas de administración de sitios dirigidas a principiantes como un aspecto de alta consideración de esta herramienta es su arquitectura extensible. La cual permite el uso de extensiones. Las extensiones son pequeños programas, que cualquier desarrollador Web puede escribir y que cualquiera puede descargar e instalar. Cabe destacar que la herramienta dreamweaver fue necesaria para la construcción de las

interfaces de usuario del Módulo de Servicio Comunitario a través de un sistema de plantillas.

Entre las funciones más atractivas de dreamweaver se encuentran:

- Hojas de estilo en cascada y capas
- JavaScript para crear efectos e interactividades
- Inserción de archivos multimedia
- Uso de plantillas y administración de sitios

La figura 13, muestra una captura de pantalla del entorno de diseño de dreamweaver.

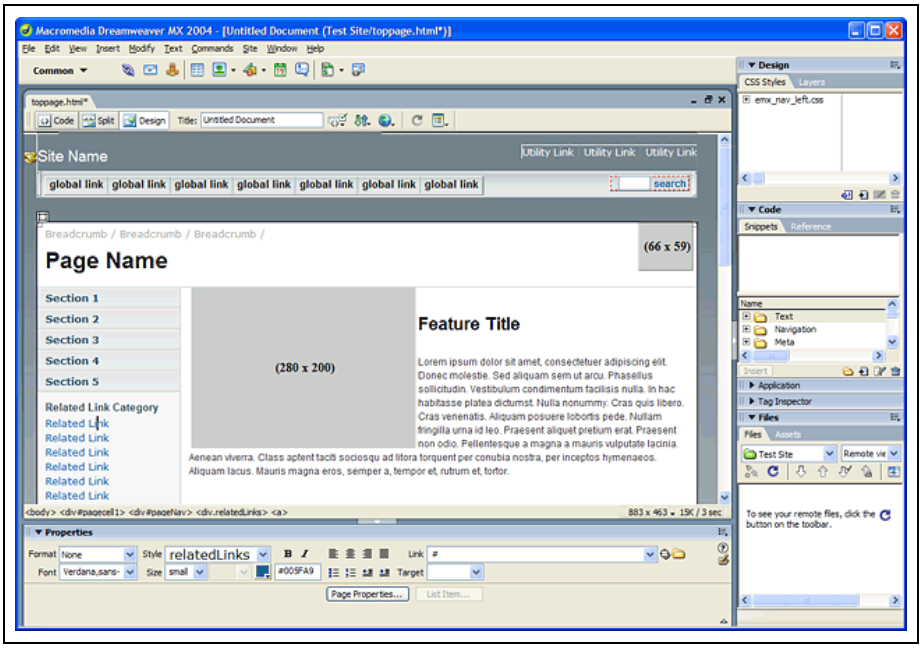

Figura 13: Ambiente de trabajo Dreamweaver

3.8 Arquitectura del proyecto de medición

Para llevar a cabo los datos de medición de los frameworks de desarrollo Java, se dispone de la arquitectura de medición que se muestra en la figura 14. Estos datos de medición no se evalúan sobre la aplicación completa, por el contrario son evaluados a componentes específicos como se mencionó anteriormente en el documento.

Los componentes son primeramente evaluados por el plugin Metrics del entorno de desarrollo Eclipse para obtener las métricas de software y segundo por la herramienta httperf para obtener la de medición de rendimiento.

Los resultados obtenidos de ambos casos son exportados en archivos para su posterior procesamiento por una herramienta Web de visualización de los datos, llamada 'visualizador de métricas'. Cuya su finalidad es la presentación de los datos en un formato mucho más entendible al usuario. Incluso con la capacidad de generar gráficos comparativos de los datos obtenidos de la mediciones hechas a cada uno de los componentes.

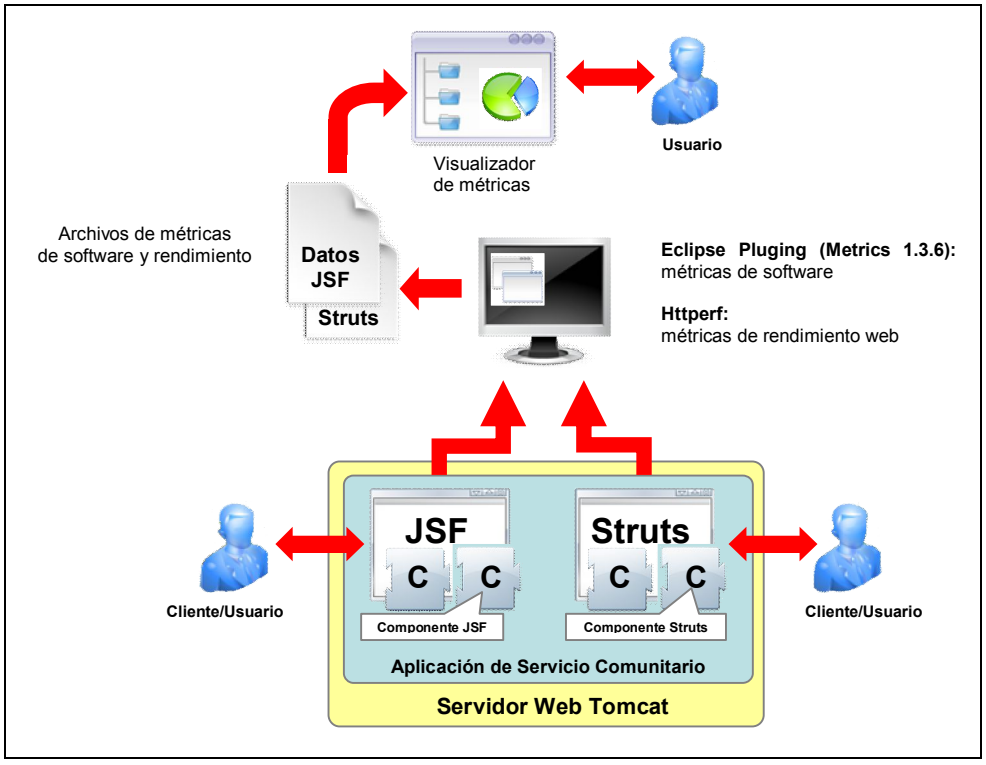

Figura 14: Arquitectura del proyecto de medición

3.9 Ambiente del experimento

Con el fin de obtener datos confiables en cuanto a la medición del rendimiento de los frameworks, bajo igualdad de condiciones, se dispuso de una plataforma para el experimento de medición con la siguiente configuración de hardware y software:

Hardware:
Servidor HP Proliant DL380
Procesador Intel Xeon Quad Core X5460 (3,6 Ghz) x 2
Bus frontal: 800 Mhz
Memoria ram: 6.592 mb

Software:
Sistema Operativo: Distribución linux debian ETCH
Servidor web: Tomcat versión 5.5
Plataforma Java: Java 1.5.0
Manejador de base de datos: MySQL 5

Cabe nuevamente destacar, que los datos obtenidos en las pruebas de rendimiento son calculados desde otra máquina y no por el mismo servidor donde se encuentran alojadas las aplicaciones.

Por otro lado se tiene una máquina cliente en la cual se encuentra la aplicación que se encargará de generar las peticiones al servidor web, específicamente a la aplicación a medir, para así obtener los datos de rendimiento. Las características de hardware y de software de éste equipo son las siguientes:

Hardware:
PC Dell Optiplex gx520
Procesador dual core 3.2 Ghz
Bus 800 Mhz
Memoria ram: 1024 mb

Software:
Sistema operativo: Distribución ubuntu linux 8.10
Httperf v9

La figura 15 muestra la conexión física entre el servidor de aplicaciones dispuesto para las pruebas de rendimiento de los frameworks y la máquina cliente desde la cuál se simularon las peticiones concurrentes de usuarios.

Un cliente httperf realiza conexiones simultáneas sobre uno de los componentes Struts o Java Server Faces dispuestos en el servidor de aplicaciones.

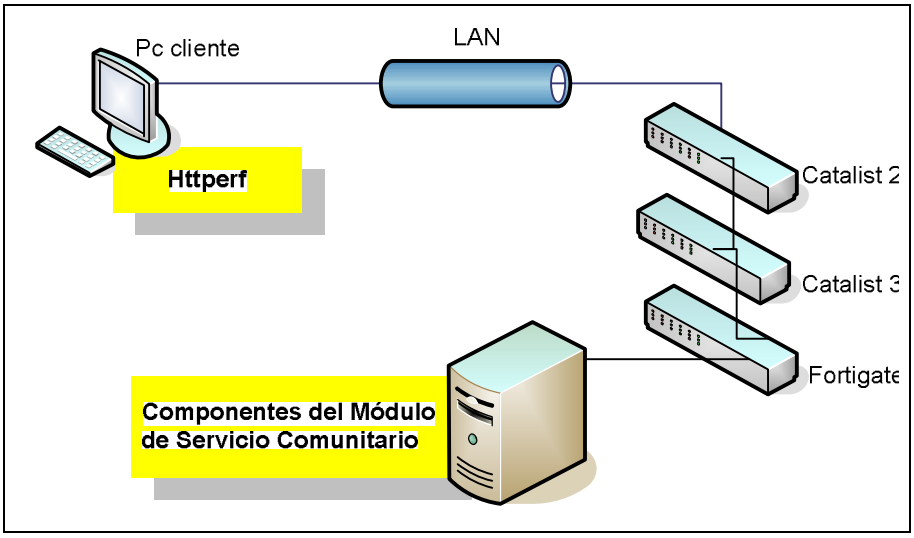

Figura 15: Diagrama de conexión física entre cliente y servidor de aplicaciones

Capítulo IV: Casos de estudio

En este capítulo se realiza una descripción detallada de los componentes de software (Páginas JSP y Beans) que fueron seleccionados como casos de estudio, para realizar el proceso de medición del software.

En la sección 3.4, se describe la arquitectura del proyecto de medición, se mencionó la medición de componentes específicos de la aplicación, en otras palabras, se seleccionaron dos funcionalidades sencillas, ya que sólo involucran la petición de una vista o página JSP, de todo el gran conjunto de vistas o páginas que conforman el Módulo de Administración del Servicio Comunitario.

A pesar de lo sencillo que pueda parecer las funcionalidades seleccionadas, su propósito final es comprender a fondo el desempeño de cada framework, conociendo la estructura interna y entendiendo la cadena de acciones que son desencadenadas internamente a nivel de los objetos del componente y de sus relaciones, las cuales hacen posible mostrar un determinado resultado al usuario.

El conjunto de páginas JSP, Beans y archivos de configuración dispuestos para el cálculo de métricas, fueron separados del resto de los archivos del Módulo de Administración de Servicio Comunitario para su proceso de medición, el cual fue llevado a cabo de la siguiente manera:

- Para el cálculo de las métricas de software, se seleccionó el proyecto dentro del entorno de desarrollo Eclipse y fue compilado para obtener los valores de las métricas. Ésta primera medición es la que arroja los archivos de métricas de software en formato XML

- Se realizó una comparación del código incrustado en las vistas, código el cual es necesario para el renderizado de los componentes de interfaz a mostrar (select, checkbox, radibutton, entre otros)

- Las mediciones de rendimiento se realizaron con ayuda de la herramienta httperf, desde una máquina cliente. Donde la herramienta actúa como un conjunto de clientes accediendo constantemente a la vista especificada y con los parámetros asignados, sobre las cuales se renderizan los componentes de interfaz de usuario (Java Server Faces o Struts2).

Cabe destacar que para el caso de escritura de los componentes basados en Java Server Faces, se utilizó parte de la implementación con la especificación RichFaces[9] para el manejo de las tablas de datos. De forma análoga las tablas de datos generadas por el framework Struts2 son construidas a través de la librería de etiquetas DisplayTag[10] desarrollado para struts.

A continuación se describen los dos casos de estudio.

[9] RichFaces: es una implementación JSF de código abierto que añade capacidad Ajax dentro de las aplicaciones existentes sin recurrir a JavaScript. Rich Faces incluye ciclo de vida, validaciones, conversores y la gestión de recursos estáticos y dinámicos. Los componentes de Rich Faces están construidos con soporte Ajax y un alto grado de personalización del "look-and-feel" que puede ser fácilmente incorporado dentro de las aplicaciones JSF

[10] DisplayTag: es una suite de código abierto compuesto por un conjunto de etiquetas personalizadas, creadas para la su incorporación en el framework struts2, permitiendo la visualización de colecciones de objetos dentro de un tabla de datos. Incorpornado un mecanismo de ordenamiento, exportación y personalización mediante hojas de estilos

4.1 Caso de estudio 1 (CE1): Renderizado de componentes básicos de interfaz de usuario

El primer caso de estudio considerado evalúa el mecanismo utilizado por cada framework para la construcción o renderizado de los componentes simples de interfaz de usuario a partir de los datos contenidos en los objetos o Beans responsables de manejar la lógica del negocio de la aplicación.

Como fue mencionado anteriormente se seleccionó un formulario básico, correspondiente a la vista de modificación de los estudiantes registrados en la base de datos de la herramienta de servicio comunitario. Este formulario es cargado con los datos de un estudiante, una vez que se activa la opción de establecer el contenido del Bean, es decir que los atributos del mismo son mapeados a la vista sobre la cual es realizada la petición Web. Lo cual indica que internamente el framework es responsable de la instanciación de la clase y cargar el formulario con el valor de los atributos privados contenidos en el Java Bean a través de los métodos públicos get().

4.1.1 Interfaz de usuario

La figura 16 muestra el diseño o la apariencia con la cual lucen las vistas, que fueron el punto de entrada para la ejecución de las pruebas de rendimiento, A partir de la construcción de las mismas, se desencadenan el conjunto de eventos internos del framework. Como es posible observar, ambas interfaces poseen exactamente los mismos componentes de interfaz. Su exacta apariencia es lograda a través del uso de hojas de estilo en cascada.

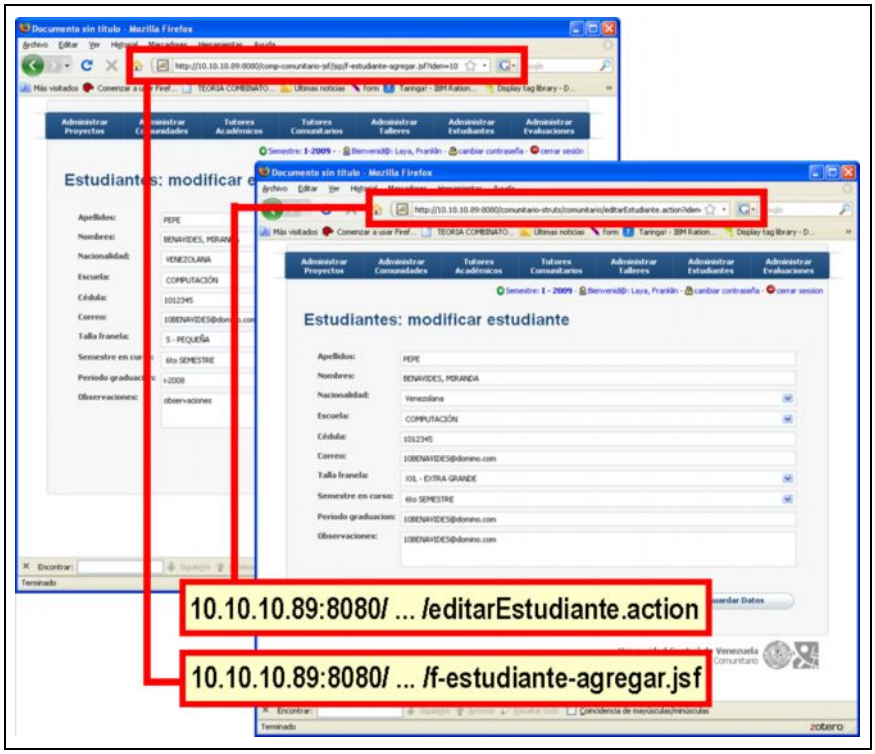

Figura 16: Capturas de pantalla de la interfaz de usuario del caso de estudio no 1

4.1.2 Diagrama de componentes

Como se observa en el diagrama de componentes de la figura 17, la implementación del caso de estudio no 1, es exactamente igual en cuanto a la estructura de sus componentes internos, específicamente de las clases o Java Beans, así de como también, la forma en como se relacionan e interactúan entre ellas, para producir una salida en las vistas tal y como se muestra en la figura 15.

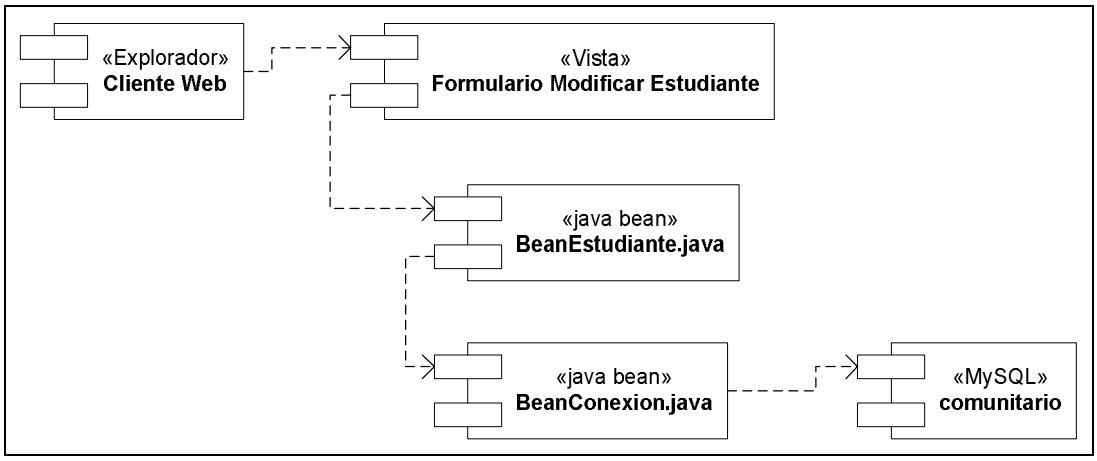

Figura 17: Diagrama de componentes del caso de estudio no 1

4.1.3 Código fuente

La estructura interna de los Java Beans (atributos y métodos) utilizados para este caso se mantiene excepto para los tipos de datos utilizados y retornados que posteriormente serán renderizados a la vista del usuario. Es decir, para los elementos tipo select, checkbox, radiobuttons, es necesaria una estructura que permita el manejo de elementos del tipo clave-etiqueta. En el caso de Java Server Faces la estructura manejada es la de SelectItem, mientras que para el caso de Struts2 la estructura manejada es un HashMap de valores.

Adicionalmente, los métodos accedidos a los Beans escritos para JSF, tienen como parámetro un actionEvent, mientras que para Struts2, los métodos son sólo accedidos si la clase o Bean hereda desde ActionSuport.

Cómo se observa en el diagrama de componentes de la figura 16, el Java Bean BeanEstudiante es el que interactúa o se mapea con la vista del usuario. A continuación la figura 18 muestra un par de fragmentos del código Java empleado con los cuales es posible recuperar la información contenida en el Bean y renderizarla en la página Web mostrada al usuario.

```
public class BeanEstudiante {


... Atributos ...
... Métodos ...

private SelectItem[] listaEscuelas;

public SelectItem[] getListaEscuelas() {

   BeanEstudianteLista b;
   b = new BeanEstudianteLista();
   this.ListaEscuelas = b.getListaEscuelas('t');
   return listaEscuelas;
}

public void setListaEscuelas
(SelectItem[] listaEscuelas) {

   this.listaEscuelas = listaEscuelas;
}

public void setearEstudiante(ActionEvent event){

   … cuerpo del método …
}

}
```

Java Bean Java Server Faces

```
public class BeanEstudiante extends ActionSupport {

... Atributos ...
... Métodos ...

private Map<String, String> listaEscuelas;

public Map<String, String> getListaEscuelas() {

   BeanEstudianteLista b;
   B = new BeanEstudianteLista();
   this.listaEscuelas = b.getListaEscuelas('t');
   return listaEscuelas;
}

public void setListaEscuelas
(Map<String, String> listaEscuelas) {

   this.listaEscuelas = listaEscuelas;
}

public void setearEstudiante(){

    … cuerpo del método …
}

public String execute() throws Exception {
   return SUCCESS;
}
}
```

Java Bean Struts2

Figura 18: Fragmentos del código fuente del Bean BeanEstudiante

4.1.4. Etiquetas empleadas en las vistas de usuario

Implementado el código Java necesario en los Java Beans para su interacción con los componentes de la interfaz de usuario, es escrito su correspondiente etiquetado en las vistas JSP.

La figura 19 muestra parte del etiquetado Java Server Faces y Struts2 que fue necesario escribir para que los atributos del Bean fuesen mapeados a sus correspondientes componentes en la interfaz de usuario, como lo son para éste caso en particular, las listas de selección simple y los campos de texto.

Como puede observarse la estructura del etiquetado usado en cada implementación es muy semejante, salvo que en el caso de Java Server Faces puede notarse que los componentes son un poco más elaborados en comparación a los componentes de Struts 2.

Una característica que resalta a simple vista en los formularios mostrados en la figura 18, está directamente relacionado con el botón de submit, que en el caso de Java Server Faces hace referencia a un método definido en un Java Bean llamado BeanEstudiante mientras que en Struts 2 se hace referencia a una acción definida dentro de la configuración de la aplicación y que es llamada salvarEstudiante.action.

<h:form id="form1"> <h:inputText value="#{BeanEstudiante.apellidos}" styleClass="required" /> <h:inputText value="#{BeanEstudiante.nombres}" styleClass="required" /> <h:selectOneMenu styleClass="validate-selection" value="#{BeanEstudiante.nacionalidad}" > <f:selectItems id="listaNacionalidad" value="#{BeanEstudiante.listaNacionalidad}" /> </h:selectOneMenu> <h:selectOneMenu styleClass="validate-selection2" value="#{BeanEstudiante.dependencia}"> <f:selectItems id="listaEscuelas" value="#{BeanEstudiante.listaEscuelas}" /> </h:selectOneMenu> … <h:commandButton value="Guarda Datos" action="#{BeanEstudiante.**salvarDatos**}" styleClass="boton" /> </h:form>	<s:form id="form1" name="form1" theme="simple" action="**salvarEstudiante.action**"> <s:textfield name="apellidos" theme="simple" cssClass="required" /> <s:textfield name="nombres" theme="simple" cssClass="required" /> <s:select name="nacionalidad" theme="simple" cssClass=" validate-selection " list="listaNacionalida"/> <s:select name="id_dependencia" theme="simple" cssClass="required" list="listaEscuelas" value="%{dependencia}" /> … <s:submit value="Guardar Datos" theme="simple" cssClass="boton" /> </s:form>
Java Bean Java Server Faces	Java Bean Struts2

Figura 19: Fragmentos del etiquetado JSf y Struts2 en las vistas del caso de estudio 1

4.2 Caso de estudio 2 (CE2): Renderizado de tablas de datos

El segundo caso de estudio tiene como finalidad evaluar el mecanismo empleado por cada framework para la construcción o renderizado de un componente de interfaz de usuario de mayor complejidad como lo es una tabla de datos.

Una tabla de datos es un elemento construido a partir de un atributo perteneciente al Bean contra el cual ha sido mapeada la vista. El cual consiste en una colección de objetos que manejan un determinado grado de complejidad, definido por sus atributos y la complejidad de sus operaciones o métodos.

La vista seleccionada para este caso de estudio, corresponde a la tabla de datos o página inicial de la sección de estudiantes de la herramienta de servicio comunitario.

4.2.1 Interfaz de usuario

En la figura 20 se muestra la apariencia con la cual lucen las vistas o páginas Web que contienen la tabla de datos de estudiantes. El listado o tabla de datos es renderizadada por el framework desde la colección de estudiantes que es consultada a las bases de datos de la aplicación de servicio comunitario.

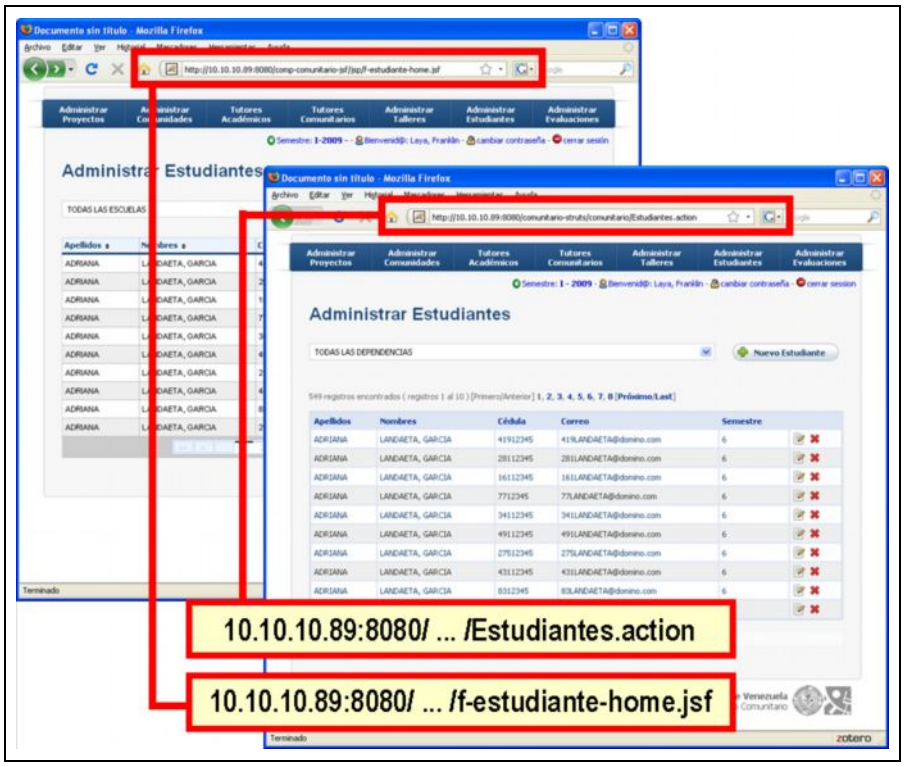

Figura 20: Capturas de pantalla de la interfaz de usuario del caso de estudio 2

4.2.2 Diagrama de componentes

La figura 21 muestra el diagrama de componentes asociado al segundo caso de estudio. Cabe destacar el diagrama es el mismo para cada implementación (Struts2 o Java Server Faces). Un mismo modelo para ambos componentes evaluados es posible gracias a la implementación de la lógica de negocio a través de Beans.

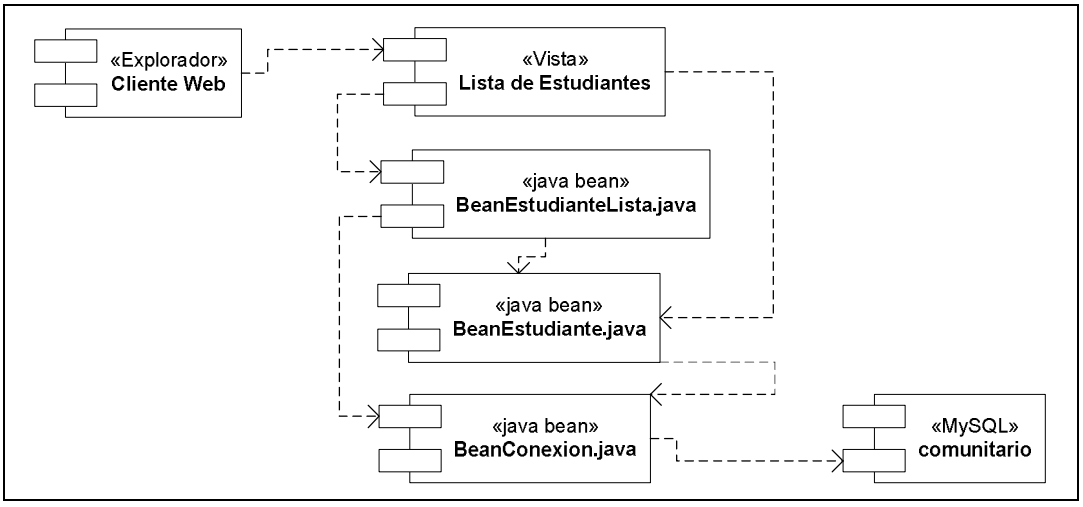

Figura 21: Diagrama de componentes del caso de estudio no 2

4.2.3 Código fuente de los Beans

La figura 22 muestra un fragmento del código Java implementado en los Beans BeanEstudianteLista, usado por los frameworks Java Server Faces y Struts2 respectivamente.

```
public class BeanEstudianteLista{

public BeanEstudianteLista(){}
ArrayList<BeanEstudiante> arreglo;

// retorna el listado de estudiantes
public ArrayList<BeanEstudiante> getArreglo() {

sesionConexion();

arreglo = new ArrayList<BeanEstudiante>();
BeanConexion conexion = (BeanConexion)
FacesContext.getCurrentInstance().
getExternalContext().getSessionMap().
get("conexion");

  Connection conn = null;
  PreparedStatement pstmt = null;
  ResultSet rset = null;
  …

  String sql= "SELECT * " +
  "FROM estudiante " + opcionBusqueda +
  "ORDER BY apellidos_estudiante,
  nombres_estudiante;";

    try {
        conn = conexion.conectar();
        pstmt = conn.prepareStatement(sql);
        rset = pstmt.executeQuery();

        while (rset.next()) {

         BeanEstudiante e = new BeanEstudiante();
         e.setEstudiante(e, rset);
         arreglo.add(e);
        }

    } catch (Exception e) { e.printStackTrace(); }
    finally {
      try {
          if (rset != null) rset.close();
          if (pstmt != null)pstmt.close();
          if (conn != null) conn.close();
      } catch (Exception e) { e.printStackTrace(); }}

return arreglo;
}}
```

Java Bean Java Server Faces

```
public class BeanEstudianteLista extends
ActionSupport {

public BeanEstudianteLista(){}
ArrayList<BeanEstudiante> arreglo;

public String execute() throws Exception {
        setArreglo( getArreglo() );
        return SUCCESS;
}

// retorna el listado de estudiantes
public ArrayList<BeanEstudiante> getArreglo() {

arreglo = new ArrayList<BeanEstudiante>();
BeanConexion conexion = (BeanConexion)
ActionContext.getContext().getSession()
.get("conexion");

  Connection conn = null;
  PreparedStatement pstmt = null;
  ResultSet rset = null;
  …

  String sql= "SELECT * FROM estudiante " +
  opcionBusqueda + "ORDER BY
  apellidos_estudiante, nombres_estudiante;";

    try {
        conn = conexion.conectar();
        pstmt = conn.prepareStatement(sql);
        rset = pstmt.executeQuery();

        while (rset.next()) {
        BeanEstudiante e = new  BeanEstudiante();
        e.setEstudiante(e, rset);
        arreglo.add(e);
        }

    } catch (Exception e) { e.printStackTrace();
    } finally {
    try {
        if (rset != null) rset.close();
        if (pstmt != null)pstmt.close();
        if (conn != null) conn.close();
        } catch (Exception e)
    { e.printStackTrace(); }}

    Map<String, ArrayList> session2 =
    ActionContext.getContext().getSession();
    session2.put("listadoEstudiantes", arreglo);
    return arreglo;
}}
```

Java Bean Struts2

Figura 22: Fragmentos del código fuente del Bean BeanEstudianteLista

4.2.4. Componentes Struts y Java Server Faces en las vistas de usuario

La figura 23, muestra un fragmento del etiquetado Java Server Faces y Struts2 utilizado en las vistas o archivos JSP para renderizado de los Beans de tipo estudiantes a la tabla de datos mostrada en la figura 20, donde se muestra la interfaz de usuario.

```
<rich:dataTable
value="#{BeanEstudianteLista.arreglo}"
var="estudiante"
styleClass="marco_tablas"
rowClasses="fila-uno, fila-dos"
headerClass="titulos_tablas"
border="0"
cellpadding="5"
rows="10"
reRender="ds">

<rich:column headerClass="titulos_tablas"
  sortBy="#{estudiante.apellidos}">
  <f:facet name="header">
    <h:outputText value="Apellidos" />
  </f:facet>
  <h:outputText value="#{estudiante.apellidos}" />
</rich:column>

... Otras columnas ...

<rich:column id="column2" headerClass="titulos_tablas">
  <f:facet name="header">
  <h:outputText value=""> </h:outputText>
  </f:facet>
  <h:commandLink id="Edit" action="editar"
  actionListener  ="#{BeanEstudiante.setearEstudiante}"
  title="editar">

    <h:graphicImage  value="../css/iconos/edit.png">
    </h:graphicImage>
    <f:param id="editId" name="id"
    value="#{estudiante.id_estudiante}" />

  </h:commandLink>
</rich:column>

<f:facet name="footer">
  <rich:datascroller id="ds"> </rich:datascroller>
</f:facet>

</rich:dataTable>
```

Java Bean Java Server Faces

```
<display:table
name="sessionScope.listadoEstudiantes"
id="row"
requestURI="#"
sort="page"
pagesize="10"
export="true"
class="marco_tablas"
cellpadding="4"  >

  <display:column property="apellidos"
    title="Apellidos" sortable="true"  />

  <display:column property="nombres"
    title="Nombres" sortable="true" />

  <display:column
   property="cedula_estudiante"
   title="Cédula" sortable="true" />

  <display:column
   property="email_estudiante"
   title="Correo" sortable="true" />

  <display:column
   property="semestre_en_curso"
   title="Semestre" sortable="true" />

  <display:column
   property="acciones_Bean" />

</display:table>
```

Java Bean Struts2

Figura 23: Fragmentos del etiquetado JSf y Struts2 en las vistas del caso de estudio 2

Capítulo V: Resultados y Conclusiones

En éste capítulo se presenta el conjunto de resultados obtenidos tras la medición de métricas de software y rendimiento realizados a los casos de estudio anteriormente descritos y visualizados a través de la herramienta de visualización de métricas. Así mismo emitiendo las respectivas opiniones y conclusiones sobre los resultados obtenidos.

5.1 Resultados

La sección de resultados está dividida en dos secciones, es decir tomando en cuenta el tipo de métricas calculadas, la sección 5.1.1 son los resultados de las métricas de software. La sección 5.1.2 muestra los resultados de las métricas obtenidas tras las pruebas de rendimiento.

5.1.1 Resultados de las métricas de software

Haciendo un repaso al capítulo IV, se listan a continuación el conjunto de métricas de software que fueron consideradas para el experimento de medición, éstas son:

- Número de las clases (NOC)
- Total de líneas de código (TLOC)
- Líneas del código por método (MLOC)
- Falta de cohesión de los métodos (LCOM)
- Complejidad Ciclomática de McCabe (VG)
- Métodos cargados por la clase (WMC)

La gráfica 24 es una captura de pantalla del módulo visor de métricas, específicamente su sección de visualización de métricas de software. En esta se muestra los resultados obtenidos en la medición del conjunto de clases que fueron seleccionadas como objeto de pruebas para el experimento de medición. En este experimento se evaluó el conjunto completo de todas cuatro clases que conforman la lógica de negocio de los casos de estudio.

Como puede apreciarse en la figura 24, cada gráfica está compuesta por dos pares de barras, donde el primer par representa los valores promedios obtenidos y en donde el segundo par representa el valor máximo alcanzado en el cálculo de la métrica.

Cada par de barras es representado en dos colores diferentes. El color rosado está asociado a los valores contenidos en el primer archivo seleccionado para la medición, mientras que su representación en color morado nos indica la asociación de dicho valor con un segundo archivo seleccionado.

Para este caso en particular, el color rosa y púrpura han sido asociados respectivamente a las mediciones realizadas al componente JSF y Struts2.

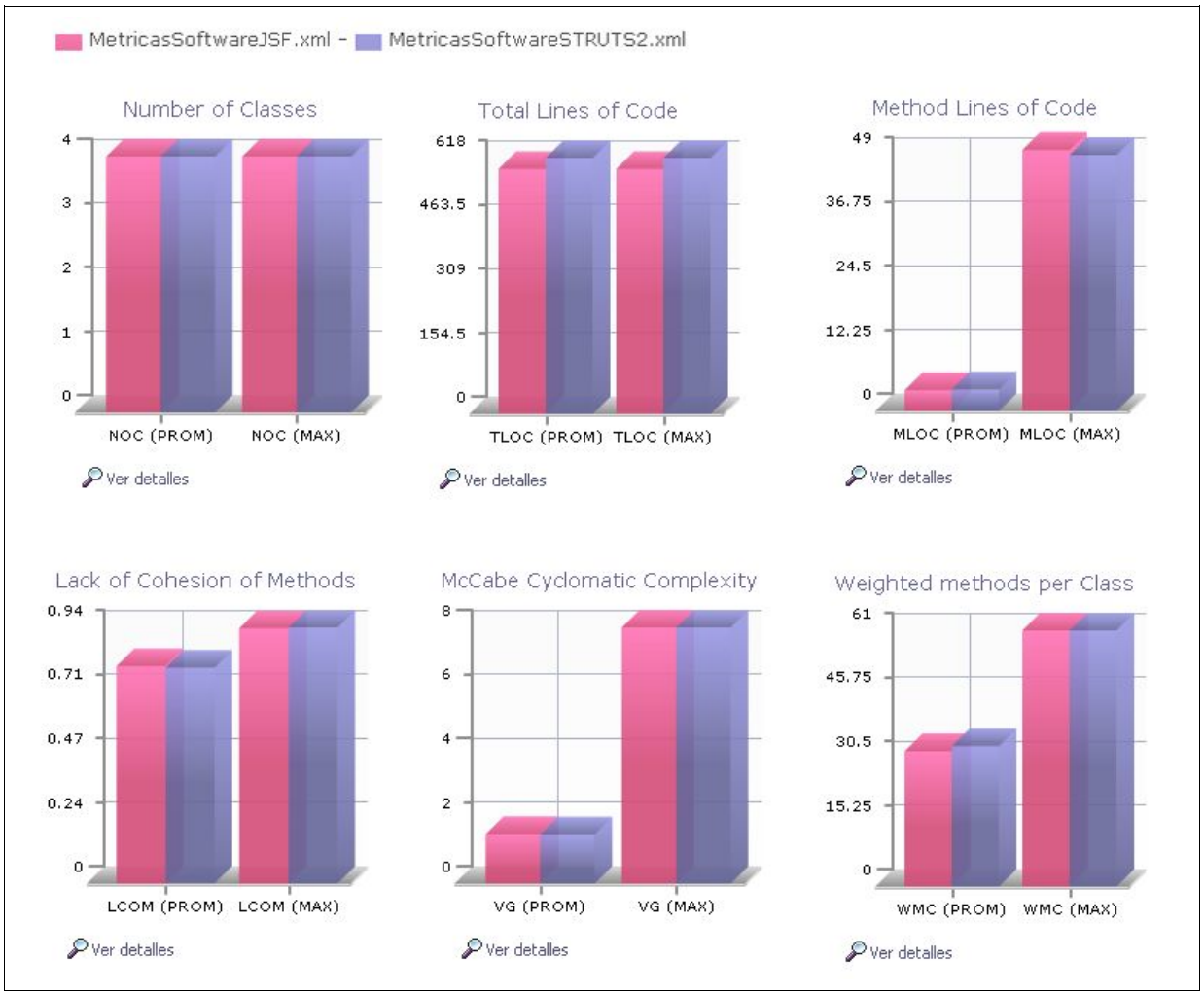

Figura 24: Gráficas de resultados de métricas de software

Es importante destacar que en los valores de la métrica de complejidad se mantienen dentro del rango esperado, por lo cual no permite asegurar que las interfaces son simples y corresponde con las buenas prácticas de programación.

5.1.2 Resultados de las métricas de rendimiento

La sección que sigue a continuación está enfocada en mostrar los resultados obtenidos y de forma breve conclusión al someter las páginas solicitadas frente a una gran cantidad de peticiones http, y de esta forma tener conocimiento del las capacidades de respuestas y manejo de peticiones por parte de cada framework.

Análogo a los casos se estudio, la sección de resultados se presenta también a su vez en dos secciones, una por cada caso de estudio. Primero se presentan los resultados del caso de estudio no 1 y caso de estudio no 2.

Tal y como se especificó en la sección 3.3.1, donde se mencionan las características de la estrategia para implementación del experimento, se escribió un shell script con el cual

ejecutar la herramienta de cálculo de rendimiento httperf y redireccionar los datos obtenidos a un archivo de texto plano, el cual con la ayuda de la herramienta de visualización de métricas transformará los datos obtenidos en gráficas de fácil comprensión para el usuario.

La figura 25 muestra el conjunto de comandos y variables que componen el shell script utilizado para llevar a cabo el experimento de medición de rendimiento.

```
# Script de calculo de métricas de rendimiento
# $1 = servidor
# $2 = uri
# $3 = archivo de salida
# $4 = numero de conexiones

peticiones=5      # no de peticiones
intervalo=0       # intervalo de llegada de conexiones
puerto=8080       # numero de iteraciones del experimento

echo "Resultados de medicion de rendimiento" >> $3
echo "Archivo resultado:" $3 >> $3

  # Calcula no. de conexiones
  conexiones=$4
  echo "------------------------------------" >> $3
  echo "-------> Comienza medición con: $conexiones conexiones"

START=$(date +%s)

# Calculo de rendimiento
httperf --hog --server=$1 --uri=$2 --port=$puerto
--wsess=$conexiones, $peticiones, $intervalo
--burst-length=$peticiones --timeout 4 --recv-buffer=1638400
--period=u0.1,0.3 >> $3

END=$(date +%s)
DIFF=$(( $END - $START ))
echo "    |----> tiempo de ejecucion: $DIFF segundos"
```

Figura 25: Contenido del shell script utilizado para el experimento de rendimiento

Como se puede observar en la figura 24, el script está compuesto por los parámetros que se describen a continuación:

- $1: indica la dirección ip del servidor
- $2: indica la uri o la dirección del recurso a medir
- $3: indica el nombre del archivo resultado en donde se guardarán los resultados obtenidos tras la ejecución del comando httperf
- $4: indica el número de conexiones serán utilizadas para llevar a cabo el experimento

5.1.3 Resultados del caso de estudio No 1. Componentes básicos de UI.

Los resultados aquí expuestos fueron obtenidos tras la medición realizada al caso de estudio No 1. Para cada gráfica la línea púrpura indica los datos obtenidos por el framework Java Server Faces y la línea verde los datos obtenidos con el framework Struts2.

Los datos se muestran en las siguientes secciones:

Conexiones: La sección de conexiones está conformada conforma por las dos gráficas mostradas en la figura 26, en ellas se expresa la siguiente información:

- La primera gráfica muestra el promedio de conexiones atendidas por segundo
- La segunda gráfica muestra el tiempo promedio de vida de duración de las conexiones creadas por cada usuario que accede al recurso

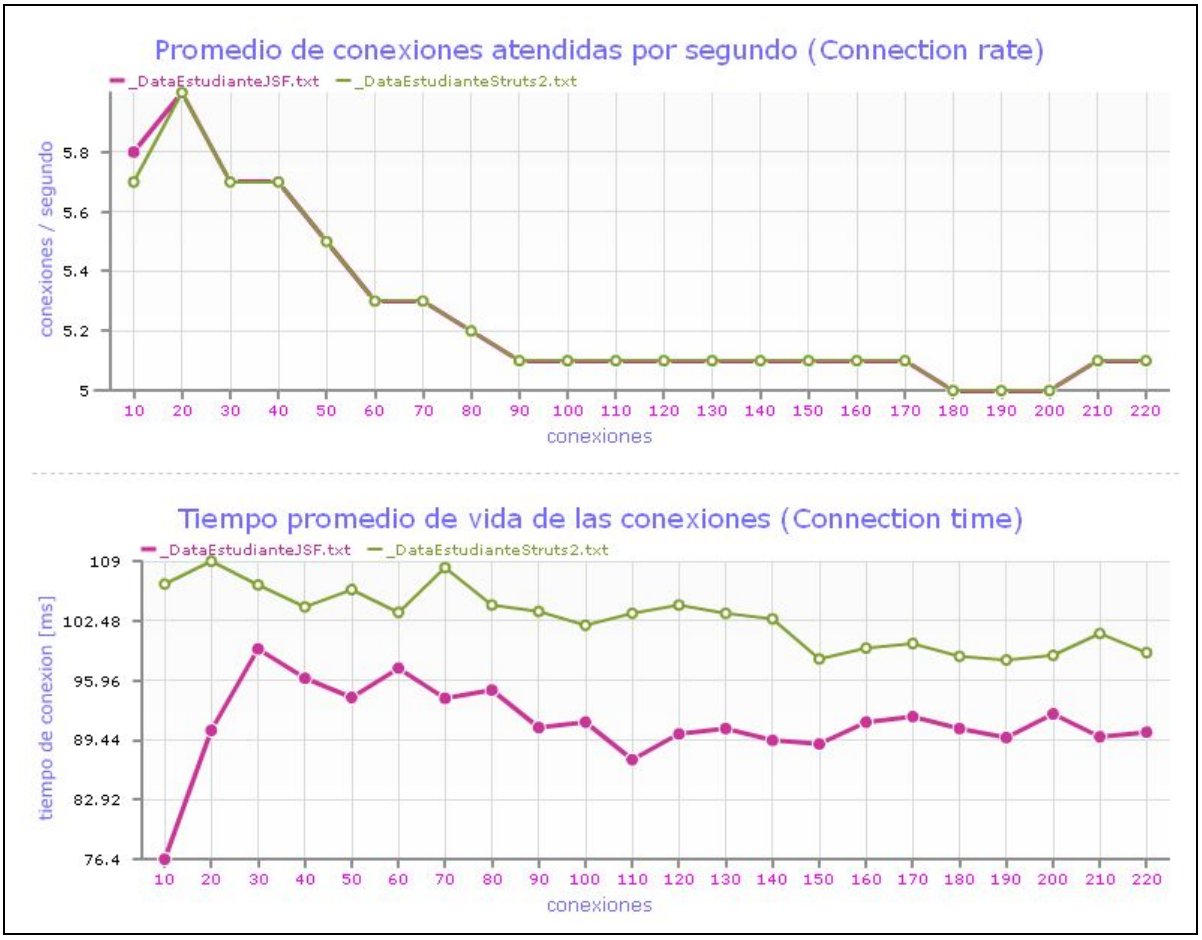

Figura 26: Gráficas de resultados de conexiones del CE1

Peticiones: La figura 27 es la gráfica del número de peticiones emitidas por segundo desde la aplicación httperf para el proceso de medición de rendimiento.

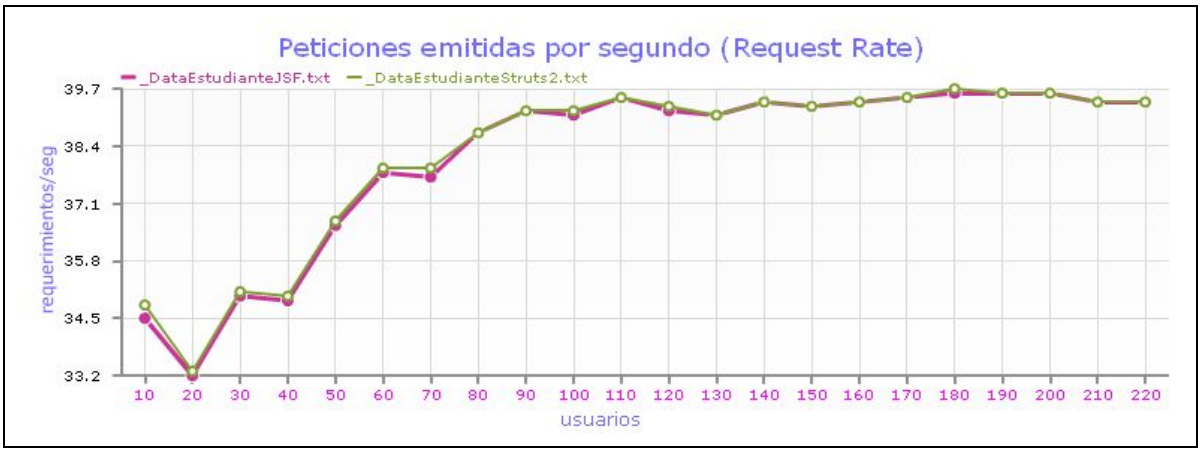

Figura 27: Gráfica de peticiones emitidas por segundo del CE1

Respuestas: La figura 28 muestra los datos obtenidos en la sección de respuestas, en esta gráfica se visualiza el tiempo promedio empleado en respuesta a las peticiones realizadas.

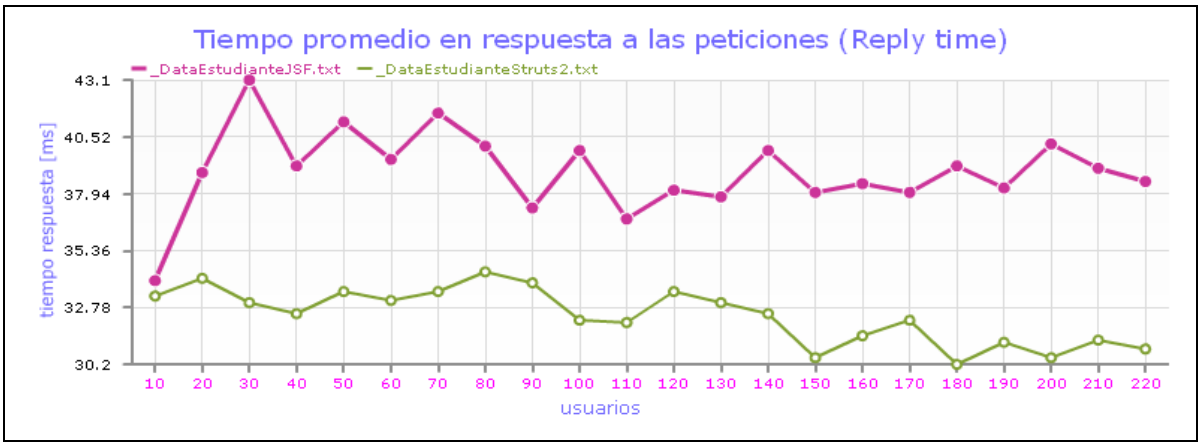

Figura 28: Gráfica de tiempo promedio de respuesta a las peticiones del CE1

Uso de cpu: La gráfica de uso de cpu, mostrada en la figura 29 es de gran utilidad para indicar la confiabilidad de los datos, los valores cercanos al 100% son indicio de una gran precisión durante la medición de rendimiento.

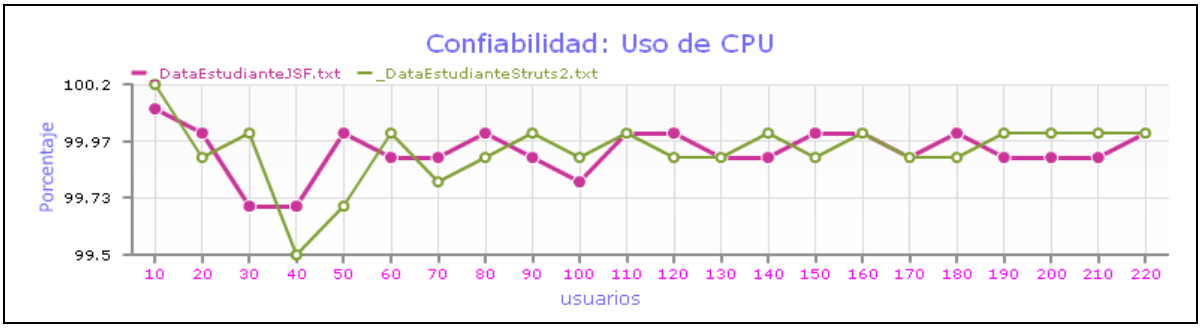

Figura 29: Gráfica de resultados de uso de cpu de los datos obtenidos en el CE1

Con respecto a los resultados obtenidos y graficados, principalmente sobre los datos mostrados en la figura 28, sobre el primer caso de estudio, se puede afirmar que el funcionamiento del framework Struts 2 presentó un mejor rendimiento que el framework Java Server Faces.

5.1.4 Resultados del caso de estudio No 2. Tablas de datos

Los resultados que son expuestos a continuación fueron obtenidos tras la medición de rendimiento realizada al caso de estudio no 2. Análogo a los resultados del caso de estudio anterior, para cada gráfica presentada, la línea púrpura indica los datos obtenidos tras medir el framework Java Server Faces y la línea verde los datos obtenidos con al medir el framework Struts2.

Los datos se muestran en las siguientes secciones:

Conexiones: La sección de conexiones está conformada por las dos gráficas mostradas en la figura 30, en ellas se expresa la siguiente información:

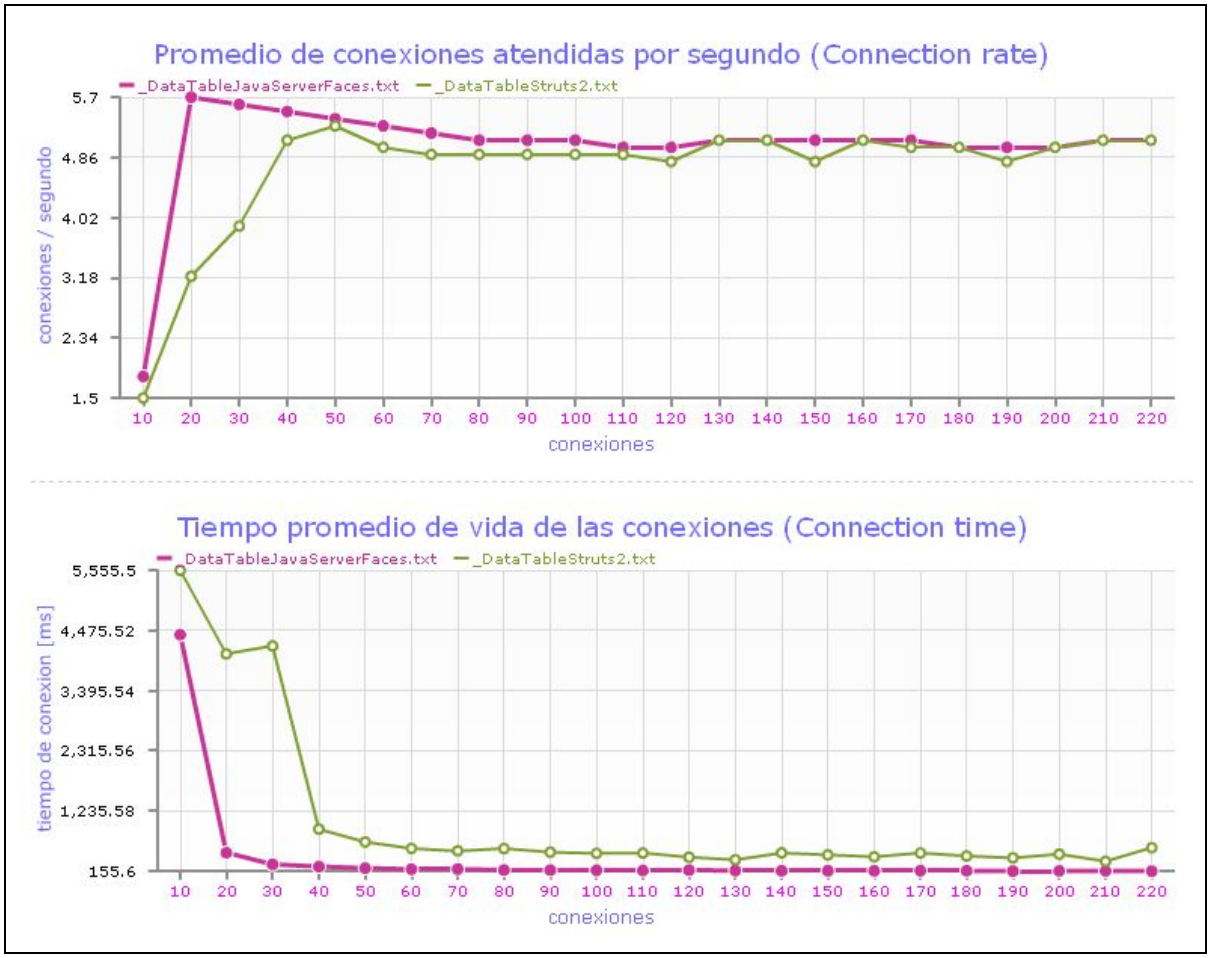

Figura 30: Gráficas de resultados de conexiones del CE2

Peticiones: La figura 31 es la gráfica del número de peticiones emitidas por segundo desde la aplicación httperf para el proceso de medición de rendimiento.

Figura 31: Gráficas de resultados de conexiones del CE2

Respuestas: La figura 32 muestra los datos obtenidos de las secciones de respuestas. Las respuestas están conformadas por dos gráficas, en ellas se visualiza la siguiente información:

- La primera gráfica muestra el promedio de conexiones atendidas por segundo
- La segunda gráfica muestra el tiempo promedio de duración de las sesiones

Figura 32: Gráficas de resultados de respuestas CE2

Uso de cpu: La gráfica de uso de cpu, mostrada en la figura 33 es de gran utilidad para indicar la confiabilidad de los datos obtenidos, a medida que los valores se acercan al 100% se tiene la certeza de ola precisión de los datos obtenidos durante esa simulación.

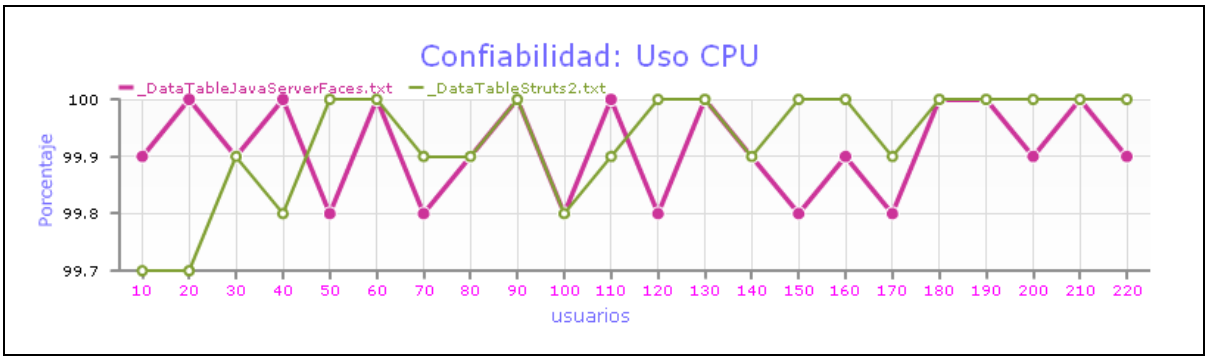

Figura 33: Gráficas de resultados del número de errores

Con respecto a los resultados obtenidos y graficados, principalmente sobre los datos mostrados en la figura 32, sobre el primer caso de estudio, se puede afirmar que el funcionamiento del framework Struts 2 presentó un mejor rendimiento que el framework Java Server Faces.

Finalmente y basados en los resultados obtenidos en ambos casos de estudio podemos afirmar que no se cumple un patrón de rendimiento en el funcionamiento de cada uno de los frameworks, por consiguiente no podemos concluir que una tecnología tenga un mejor desempeño que otra.

5.2 Conclusiones

Los frameworks de desarrollo para la capa de presentación Struts 2 y Java Server Faces utilizados en la elaboración de este trabajo de grado, son en la actualidad dos de las herramientas más populares entre los programadores al momento de desarrollar aplicaciones basadas en tecnología Java. Como cualquier otra herramienta de software pueden utilizarse métricas, como indicadores de medida de la calidad del software, dándole valor agregado y mejoras al producto desarrollado; aportando practicidad y experiencia al usuario que los utilice para desarrollar aplicaciones Web. Quizás Java Server Faces presente una leve ventaja por ser un marco de trabajo mucho más manejable, por tener una sintaxis mucho más sencilla, por ser más fácil de configurar y por consiguiente también por ser mucho más fácil de aprender, a diferencia de Struts 2; énfasis hecho en base a la experiencia propia que aplicamos en los resultados obtenidos y en la referida por usuarios externos, los cuales usaron parte de la aplicación que se elaboró o que trabajaron en el desarrollo de aplicaciones relacionadas con estos dos frameworks (Ver apéndice F).

Las métricas de software fueron de gran ayuda y una herramienta útil para el momento en el cual surgió la necesidad de medir el código elaborado; mientras que las medición de rendimiento ayudo a conocer estadísticas del producto informático que se estaba desarrollando. Este último fue sometido a distintas pruebas tales como niveles máximos de usuarios, conexiones o sesiones. Estas estadísticas develaron resultados que nos indicaron de alguna manera el comportamiento de la aplicación que se estaba desarrollando.

Finalmente respondiendo a las preguntas ¿Cuál de los dos frameworks es más eficaz para su implementación de un proyecto, basándose en la definición de métricas durante su desarrollo y la experiencia del grupo de trabajo?, ¿Cuál de los dos frameworks de la capa de presentación ofrece más ventajas y puede ser el más idóneo para afrontar el desarrollo de una aplicación Web, bajo la plataforma Java?

En base a los resultados obtenidos opinamos que la escogencia de alguno de las dos tecnologías, sea Java Server Faces o Struts 2, puede depender en gran parte de la experiencia y de la comodidad que sienta el desarrollador o grupo de desarrollo al momento de escribir y configurar una aplicación bajo alguno de los frameworks mencionados.

En nuestro caso para comprobar el funcionamiento y desempeño de ambos frameworks se recurrió a la medición de componentes previamente seleccionados, en igual número y funcionalmente equivalentes, sin embargo ambos marcos trabajados presentaron resultados muy parecidos durante el experimento de medición de nuestros casos de estudio, lo cual nos lleva a no poder afirmar en nuestro trabajo, cual de los dos frameworks es más óptimo para desarrollar la aplicación Web en estudio.

Analizando las métricas de software, observamos que son bastante similares. Se presenta -en promedio- una diferencia casi apreciable en los valores obtenidos. En cuanto al rendimiento en el caso de estudio 1 JavaServer Faces fue superior, mientras que en el segundo Struts funcionó mejor.

Es decir se puede afirmar con base a nuestros resultados que los frameworks Strust 2 y Java Server Faces, no son equiparables, en nuesta experiencia, pues funcionalmente se comportaron de manera equivalente; esto es, argumentando, que no se consigue una linealidad o un posible modelo o patrón comparativo de rasgos o atributos presentes en los dos Marcos de Trabajo que indiquen que un framework presenta una característica de superioridad o distintiva con respecto al otro en los casos de estudio presentados.

Recomendamos como trabajos futuros:

1. Continuar el esquema presentado en este Trabajo Especial de Grado, incorporando otros Frameworks

2. Traducción de la salida de las herramientas al idioma castellano

3. Desarrollo de una herramienta de medición de métricas de rendimiento en plataforma Linux que se integre con la herramienta de medición httperf y el entorno del Servidor Web para realización automatizada de pruebas de rendimiento por lotes

4. Continuar con el desarrollo del Módulo de Administración del Servicio Comunitario, contemplando las opciones de: registro, impresión y entrega de certificados. Así como crear un mecanismo de discriminación para la asistencia entre talleres y curso introductoria con el fin de automatizar el procedimiento de aprobación por asistencia.

Referencias bibliográficas

[1] CAVANESS, CHUCK. *Jakarta Struts, Desarrollo de aplicaciones Web con servlets y JSP*. Editorial Anaya Multimedia. O`reilly. Madrid 2007. Pp. 544; fuente original: www.AnayaMultimedia.es, sección "Atención al cliente", opción complementos" ó http://examples.oreilly.com/0596006519,

[2] LARMAN, Craig. (1999). *UML y Patrones*. Upper Saddle River, Nueva Jersey, Estados Unido: Prentice Hall.

[3] Ambler Scott. *EFFECTIVE PRACTICES FOR MODELING AND DOCUMENTATION.*, 2009, from http://www.agilemodeling.com/.

[4] *ESTUDIO DE MÉTRICAS DE SOFTWARE. Métricas.* http://www.lsi.us.es/docencia/get.php?id=1474.

[5] McCabe, T. and A. Watson. (1994). Software complexity.

[6] *Métricas de cohesion* http://www.aivosto.com/project/help/pm-oo-cohesion.Htm

[7] GONZÁLEZ RODRÍGUEZ, Z. (2008). *Estudio de métricas de software: Caso ruby on rails y frameworks en Java*

[8] AQUILINO Adolfo, J. F. *Metodología para el diseño de métricas en tiempo de ejecución.*http://www.di.uniovi.es/~cueva/investigacion/tesis/Aquilino.pdf

[9] CUEVA COELLE, Juan. Manuel. *Metricas de usabilidad en la Web.* http://www.di.uniovi.es/~cueva/asignaturas/doctorado/2004/MetricasUsabilidad.pdf.

[10] Pressman, R. (2002). *Ingeniería del software, un enfoque práctico* (5ta ed.) McGraw-Hill.

[11] *CORE DEVELOPERS GUIDE.* (2008) , 2008, from http://struts.apache.org/2.x/docs/core-developers-guide.html

[12] *STRUTS 2 TUTORIAL, Struts2.* http://www.roseindia.net/Struts/Struts2/

[13] ROUGHLEY, Ian. *Starting struts 2.0.*http://www.infoq.com/minibooks/starting-Struts2

[14] Sun Microsystems. (2009). *JAVASERVER FACES TECHNOLOGY.*http://Java.sun.com/Javaee/JavaServerFaces/download.Htm

[15] *INTRODUCION LA TECNOLOGÍA JAVA SERVER FACES, Java en castellano.* http://www.programacion.com/Java/tutorial/jsf_intro/

[16] *Wikipedia, ¿Qué son plugins o agregados?* http://es.wikipedia.org/wiki/Add-on

[17] *Embarcadero.* http://www.embarcadero.com/products/erstudio/index.html

[18] *ArgoUml.* http://argouml.tigris.org/

[19] *Desarrollo Web.* http://www.desarrolloweb.com/articulos/332.php

[20] *Httperf, una herramienta para muestreo de rendimiento.* (2008). http://www.hpl.hp.com/research/linux/httperf/

[21] CHAD MICHAEL, D. (2008). In Manning Publications (Ed.), *Struts 2 in action*

[22] MOSELEY Ralhp. (2008). Desarrollo de aplicaciones Web

[23] RODRÍGUEZ DE LA FUENTE, Santiago. (2008). In Thomson Editores. Paraninfo. S. A. (Ed.), *Programación de aplicaciones Web*

[24] Roland Barcia. *Faces (JSF) vs struts A brief comparison.*, 2004, http://homepage1.nifty.com/algafield/barcia.html

[25] Craigh McClanahan. *JavaServer faces and struts: Competition or coexistence?.* , 2003, http://www.baychi.org/calendar/files/Struts-And-Faces/Struts-And-Faces.pdf

[26] Gerber Fuentes Torrico, Lic. Lineth Cintya Camacho Cabrera. *Benchmark sobre tecnologías faces.*, 2007, http://www.postgradoinformatica.edu.bo/enlaces/investigacion/pdf/INGSW3_39.pdf?PHPSESSID=7c920e13611d0a050d88bb251e83df8

[27] Nicolas Brasseur. *Coexistence of JSF and struts.* , 2004 http://www.baychi.org/calendar/files/Struts-And-Faces/Struts-And-Faces.pdf

Apéndices

Apéndice A: Módulo de Administración de Servicio Comunitario

El módulo de Administración de Servicio Comunitario es una herramienta desarrollada para ser utilizada como caso de estudio, además de plantearse como un modelo genérico para modelar los procesos involucrados con la Unidad de Servicio Comunitario de la Facultada de Ciencias. La descripción del modelos de casos de uso y diseño de base de datos son descritas a continuación.

Modelo de casos de uso

Durante el desarrollo del Módulo de Servicio Comunitario, sólo se consideró el perfil de usuario administrador, es decir, el único actor en interactuar con la aplicación, sin embargo durante la etapa de diseño se dejó abierta la posibilidad de incluir nuevos actores con prioridades, pero que no forman parte del plan de trabajo planteado.

El módulo construido fue estructurado en ocho casos de uso principales, tal y como se muestran en la figura A1.

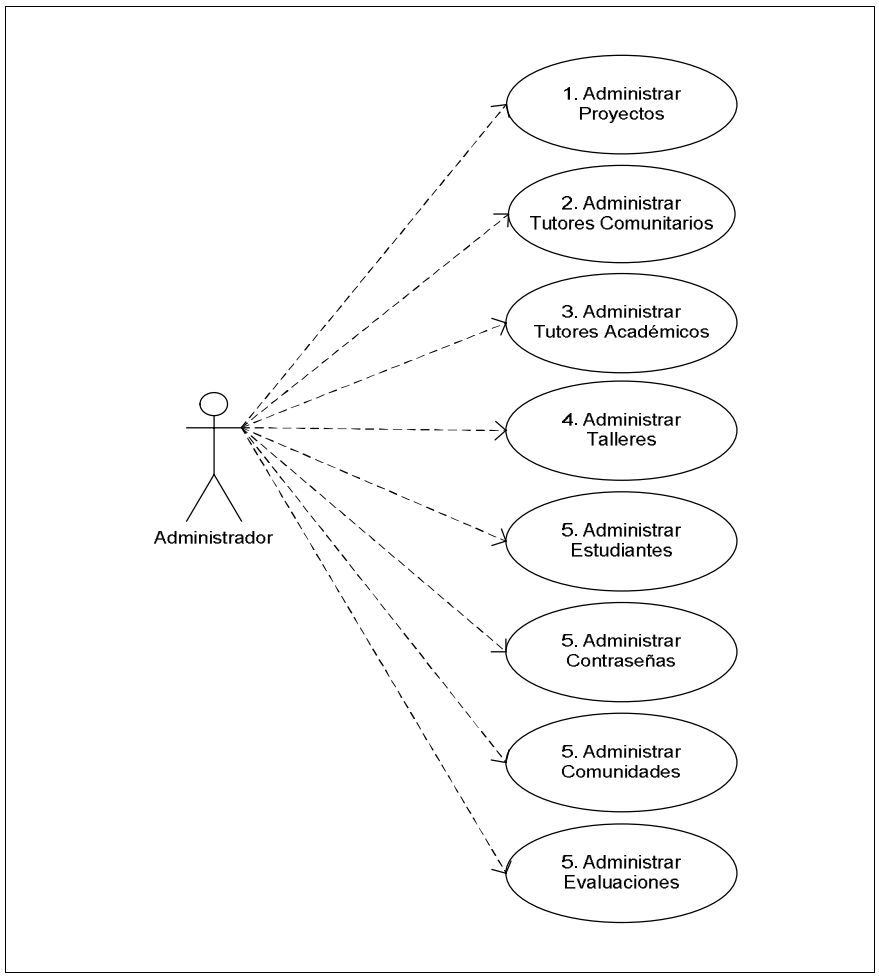

Figura A1: Casos de uso del módulo de servicio comunitario

Los casos de uso son representados en la figura A1 son descritos a continuación:

Identificador:	1
Nombre:	Administrar Proyectos
Actor Principal:	Administrador
Descripción:	Permite al usuario administrador manejar las opciones básicas tales como agregar, modificar o eliminar proyectos, los cuales son posteriormente asignados a los estudiantes para prestar su servicio comunitario

Identificador:	2
Nombre:	Administrar Tutores Comunitarios
Actor Principal:	Administrador
Descripción:	Permite al usuario administrador manejar opciones básicas tales como agregar, modificar o eliminar tutores comunitarios, así como vincular su información directamente a la comunidad donde pertenezcan

Identificador:	3
Nombre:	Administrar Tutores Académicos
Actor Principal:	Administrador
Descripción:	Permite que el usuario administrador maneje opciones básicas tales como agregar, modificar la información de tutores académicos, así mismo como inhabilitarlos para que puedan o no prestar tutoría sobre algún estudiante de la facultad. Conservando así un repositorio con los datos de todos los profesores de la Facultad de Ciencias

Identificador:	4
Nombre:	Administrar Talleres
Actor Principal:	Administrador
Descripción:	Permite al usuario administrador manejar las opciones básicas tales como agregar, modificar o eliminar la información de los talleres o cursos introductorios a los cuales deben asistir y aprobar los estudiantes (previo a la prestación del servicio comunitario). Además poseen la opción de asignar el estado de asistencia por cada estudiante

Identificador:	5
Nombre:	Administrar Estudiantes
Actor Principal:	Administrador
Descripción:	Permite al usuario administrador manejar las opciones básicas tales como visualizar, agregar, modificar o eliminar la información de los estudiantes registrados en la base de datos

Identificador:	6
Nombre:	Administrar Contraseñas
Actor Principal:	Administrador
Descripción:	Permite al usuario modificar su contraseña de ingreso al sistema

Identificador:	7
Nombre:	Administrar Comunidades
Actor Principal:	Administrador
Descripción:	Permite que el usuario administrador maneje opciones básicas tales como agregar, modificar o eliminar la información de comunidades en las cuales es prestado el servicio comunitario de los estudiantes de la Facultad de Ciencias. Así mismo puede consultar la información de la comunidad sus tutores comunitarios y los estudiantes que se encuentran prestando servicio comunitario

Identificador:	8
Nombre:	Administrar Evaluaciones
Actor Principal:	Administrador
Descripción:	Permite que el usuario administrador maneje opciones básicas tales como agregar, modificar o eliminar información sobre las posibles evaluaciones a las cuales pueden ser sometidos los estudiantes que se encuentran realizando el curso introductorio del servicio comunitario. También tiene la opción de asignar las notas obtenidas por los estudiantes, durante las evaluaciones.

Diseño de base de datos

A partir de la actividad de levantamiento de información fue posible desarrollar un diseño básico de la base de datos para el Módulo de Servicio Comunitario, el cual permitiría modelar el comportamiento deseado de los procedimientos llevados a diario por la Unidad de Servicio Comunitario, en conjunto con algunos requerimientos propiamente solicitados.

Éste diseño de base de datos se encuentra conformado por 21 tablas en las que se maneja la información de los entes involucrados, tales como: estudiantes, talleres, tareas, escuelas, tutores comunitarios, comunidades, tutores académicos, entre otros.

La figura A2 muestra el diseño lógico de la base de datos, indicando exclusivamente en la gráfica las claves primarias y foráneas de cada tabla entidad o relación.

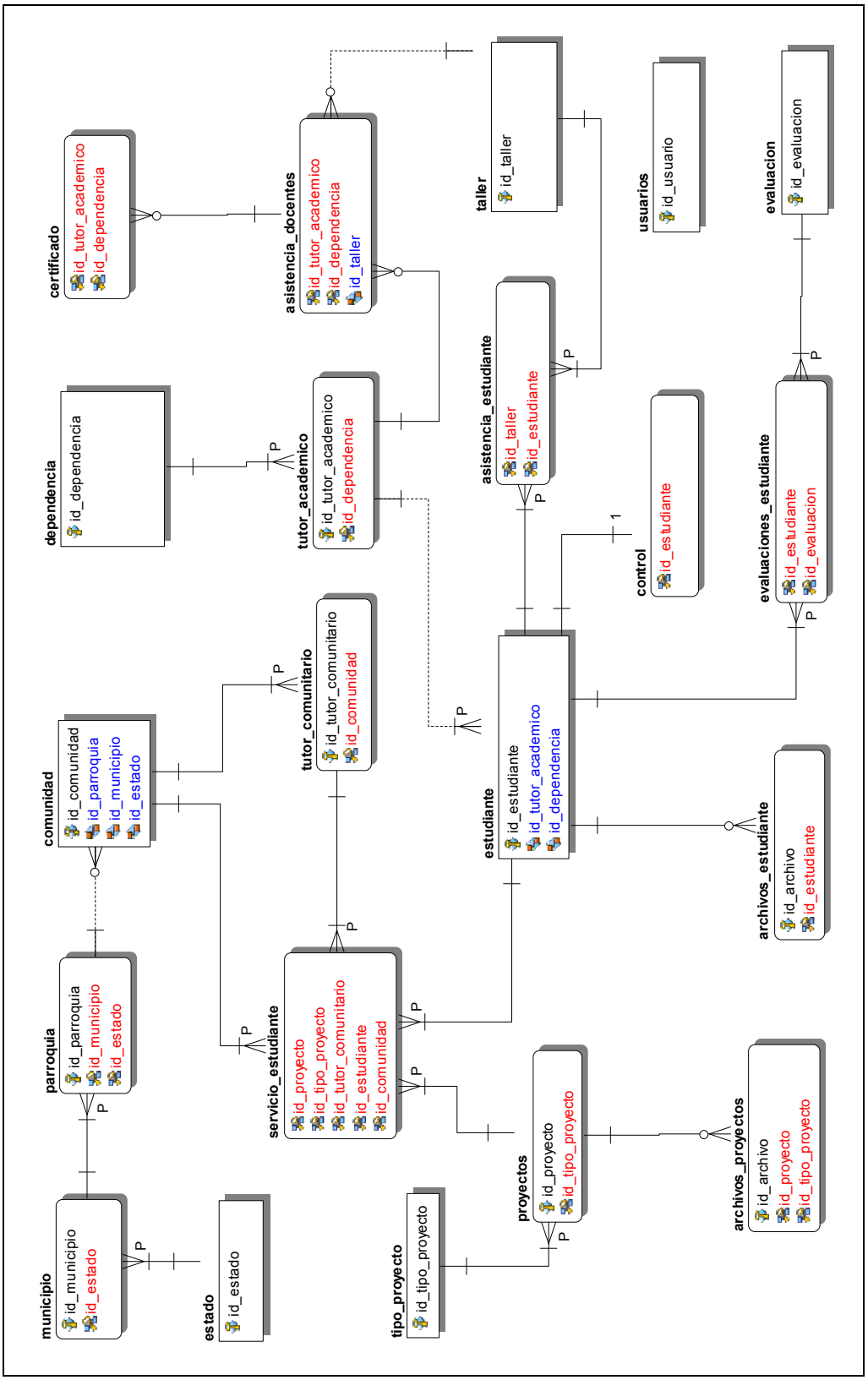

Figura A2: Diseño lógico de base de datos de los procesos de servicio comunitario

A continuación se presenta el diccionario de datos de la base de datos mostrada en la figura A2, en el cual se especifican los atributos, referencias y la descripción de cada una de las tablas que componen su diseño físico.

Tabla: archivos_estudiante

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_archivo	int(11)	Not Null		Si
id_estudiante	int(11)	Not Null		Si
descripcion_archivo	text			
tipo_mime	char(50)	Not Null		
archivo_alumno	longblob	Not Null		
fecha_agregado	datetime	Not Null		

Índices

Nombre del indice	En el campo	Único	
PRIMARY	`id_archivo`, `id_estudiante`	Yes	
Ref187	`id_estudiante`		

Descripción:

Tabla que almacena cualquier material, tal como presentaciones, informes, entre otros que sustenten la labor desempeñada por el estudiante en la comunidad donde prestó sus servicios.

Tabla: archivos_proyectos

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_archivo	int(11)	Not Null		Si
id_proyecto	int(11)	Not Null		Si
id_tipo_proyecto	int(11)	Not Null		Si
tipo_mime	char(50)	Not Null		
archivo_proyecto	longblob	Not Null		
fecha_agregado	datetime	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_archivo`, `id_proyecto`, `id_tipo_proyecto`	Si	
Ref22101	`id_proyecto`, `id_tipo_proyecto`		

Descripción:

Tabla que sirve para guardar cualquier material, tal como imágenes, mapas, archivos de texto, entre otros, que brinden una información complementaria al proyecto.

Tabla: asistencia_docentes

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_tutor_academico	int(11)	Not Null		Si
id_dependencia	int(11)	Not Null		Si
id_taller	int(11)	Not Null		
nota_asistencia	text			
tipo_asistencia	char(1)	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_tutor_academico`, `id_dependencia`	Si	
Ref25	`id_dependencia`, `id_tutor_academico`		
Ref1286	`id_taller`		

Descripción:

Tabla que maneja el registro de las asistencias de los docentes a los talleres registrados en la tabla "taller"

Tabla: asistencia_estudiante

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_taller	int(11)	Not Null		Si
id_estudiante	int(11)	Not Null		Si
tipo_asistencia	char(1)			
observacion_asistencia	text			

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_taller`, `id_estudiante`	Si	
Ref146	`id_estudiante`		
Ref1222	`id_taller`		

Descripción:

Tabla que almacena el registro de las asistencias de los estudiantes a los talleres registrados en la tabla "taller"

Tabla: certificado

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_tutor_academico	int(11)	Not Null		Si
id_dependencia	int(11)	Not Null		Si
status_certificado	char(1)			
semestre	char(1)	Not Null		
ano	int(11)	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_tutor_academico`, `id_dependencia`	Si	
Ref5125	`id_dependencia`, `id_tutor_academico`		

Descripción:

Tabla que almacena el control de los certificados entregados por la unidad de servicio comunitario a los docentes

Tabla: comunidad

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_comunidad	int(11)	Not Null		Si
id_parroquia	int(11)	Not Null		
nombre_comunidad	char(100)	Not Null		
persona_contacto	char(40)			
telefono_comunidad	char(12)			
email_comunidad	char(40)			
id_municipio	int(11)	Not Null		
id_estado	int(11)	Not Null		
direccion_comunidad	text	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_comunidad`	Si	
Ref914	`id_municipio`, `id_parroquia`, `id_estado`		
Refparroquia14	`id_parroquia`, `id_municipio`, `id_estado`		

Descripción:

Tabla que almacena la lista de las comunidades a las cuales se les brindará el servicio

Tabla: control

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_estudiante	int(11)	Not Null		Si
fecha_entrega_informe	date	Not Null		
fecha_presentacion	datetime	Not Null		
status_presentacion	char(3)	Not Null		
semestre	char(1)	Not Null		
ano	int(11)	Not Null		
status_certificado	char(2)	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_estudiante`	Si	
Ref139	`id_estudiante`		

Descripción:

Tabla que almacena el registro de entregas del informe y presentación a la unidad de servicio comunitario, entrega de certificado, así como su aprobación y fecha de exposición de su presentación

Tabla: dependencia

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_dependencia	int(11)	Not Null		Si
tipo_dependencia	char(1)	Not Null		
nombre_dependencia	text	Not Null		
descripcion_dependencia	text	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_dependencia`	Si	

Descripción:

Tabla que almacena la lista de las dependencias o escuelas de la faculta de ciencias

Tabla: estado

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_estado	int(11)	Not Null		Si
nombre_estado	text	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_estado`	Si	

Descripción:

Tabla que almacena el listado de los estados del territorio nacional

Tabla: estudiante

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_estudiante	int(11)	Not Null		Si
status	char(1)			
id_tutor_academico	int(11)	Not Null		
id_dependencia	int(11)	Not Null		
apellidos_estudiante	char(40)	Not Null		
nombres_estudiante	char(40)	Not Null		
nacionalidad	char(1)	Not Null		
cedula_estudiante	int(11)	Not Null		
numero_planilla	varchar(11)	Not Null		
semestre_en_curso	int(11)			
periodo_graduacion	char(10)			
email_estudiante	char(40)			
talla_franela	char(4)			
observaciones_estudiante	char(200)			
aprobado	char(1)			

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_estudiante`	Si	
Ref282	`id_dependencia`, `id_tutor_academico`		
Reftutor_academico82	`id_tutor_academico`, `id_dependencia`		

Descripción:

Tabla que almacena el registro de los estudiantes (responsable de brindar el servicio)

Tabla: evaluacion

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_evaluacion	int(11)	Not Null		Si
tipo_evaluacion	char(40)	Not Null		
Semestre	char(1)	Not Null		
fecha_evaluacion	date	Not Null		
descripcion_evaluacion	text	Not Null		
Ano	int(11)	Not Null		

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_evaluacion`	Si	

Descripción:

Tabla que almacena la información de las evaluaciones creadas y que se realizarán a los estudiantes durante el semestre o curso de introducción al servicio comunitario. Indicando el tipo de evaluación, fecha, semestre, evaluación, ponderación y año

Tabla: evaluaciones_estudiante

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_estudiante	int(11)	Not Null		Si
id_evaluacion	int(11)	Not Null		Si
fecha_evaluado	date	Not Null		
nota_evaluacion	int(11)			
observacion_evaluacion	text			

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_estudiante`, `id_evaluacion`	Si	
Ref188	`id_estudiante`		
Ref2589	`id_evaluacion`		

Descripción:

Tabla que almacena el registro de las evaluaciones realizadas a los estudiantes durante la etapa de curso introductorio. Nota: la fecha de la evaluación no necesariamente debe coincidir con la fecha de registro de la evaluación (rezagados)

Tabla: municipio

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_municipio	int(11)	Not Null		Si
id_estado	int(11)	Not Null		Si
nombre_municipio	text	Not Null		

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_municipio`, `id_estado`	Si	
Ref33108	`id_estado`		

Descripción:

Tabla que almacena el listado de los municipios pertenecientes a los estados del territorio nacional

Tabla: parroquia

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_parroquia	int(11)	Not Null		Si
id_municipio	int(11)	Not Null		Si
id_estado	int(11)	Not Null		Si
nombre_parroquia	char(40)	Not Null		

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_parroquia`, `id_municipio`, `id_estado`	Si	
Ref34109	`id_estado`, `id_municipio`		
Refmunicipio109	`id_municipio`, `id_estado`		

Descripción:

Tabla que almacena el listado de las parroquias pertenecientes a los municipios del territorio nacional

Tabla: proyectos

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_proyecto	int(11)	Not Null		Si
id_tipo_proyecto	int(11)	Not Null		Si
titulo_proyecto	char(100)	Not Null		
descripcion_proyecto	text	Not Null		
objetivos_proyecto	text	Not Null		
org_promueve	text			
org_financian	text			
observaciones	text			
status_aprobacion	char(10)			
status_proyecto	char(2)	Not Null		

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_proyecto`, `id_tipo_proyecto`	Si	
Ref38113	`id_tipo_proyecto`		

Descripción:

Tabla que almacena la lista de proyectos específicos en los cuales puede un estudiantes prestar su servicio comunitario (manejado como macroproyecto en la unidad de servicio comunitario)

Tabla: servicio_estudiante

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_proyecto	int(11)	Not Null		Si
id_tipo_proyecto	int(11)	Not Null		Si
id_tutor_comunitario	int(11)	Not Null		Si
id_estudiante	int(11)	Not Null		Si
id_comunidad	int(11)	Not Null		Si
horas_actividad	int(11)	Not Null		
personas_atendidas	int(11)	Not Null		
status	char(1)			

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_proyecto`, `id_tipo_proyecto`, `id_tutor_comunitario`, `id_estudiante`,	Si	

	`id_comunidad`		
Ref2297	`id_proyecto`, `id_tipo_proyecto`		
Ref3117	`id_tutor_comunitario`, `id_comunidad`		
Ref1121	`id_estudiante`		
Refcomunidad128	`id_comunidad`		

Descripción:

Tabla que vincula la información del proyecto y comunidad donde presta sus servicios un estudiante, así como su tutor comunitario, sus horas de actividad y personas atendidas. El campo estatus de la tabla sirve para llevar un histórico de las comunidades donde ha prestado su servicio un estudiante. Brindando la posibilidad que un estudiante concluya su servicio comunitario en otra comunidad distinta en la cual comenzó su trabajo, para así completar el total de horas exigidos por ley

Tabla: taller

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_taller	int(11)	Not Null		Si
tipo_taller	char(10)	Not Null		
fecha_taller	Date	Not Null		
hora_taller	Time	Not Null		
duracion_taller	int(11)	Not Null		
lugar_taller	Text	Not Null		
actividades_taller	varchar(150)			

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_taller`	Si	

Descripción:

Tabla que almacena la información de los talleres que serán impartidos por la unidad de servicio comunitario a los tutores académicos o estudiantes

Tabla: tipo_proyecto

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_tipo_proyecto	int(11)	Not Null		Si
tipo	text	Not Null		

Índices

Nombre del índice	En el campo	Único	
PRIMARY	`id_tipo_proyecto`	Si	

Descripción:

Tabla que almacena la clasificación en el entorno o ambiente en la cual se enmarcan los microproyectos donde los estudiantes prestan su servicio comunitario (Es la clasificación seleccionada por el estudiante vía Web, durante el proceso de inscripción ante la unidad de servicio comunitario)

Tabla: tutor_academico

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_tutor_academico	int(11)	Not Null		Si
id_dependencia	int(11)	Not Null		Si
cedula	int(11)	Not Null		
nombres_tutor	char(50)	Not Null		
apellidos_tutor	char(50)	Not Null		
cargo_tutor	text	Not Null		
ubicacion_tutor	text	Not Null		
telefono_tutor	char(12)	Not Null		
email_tutor	text			
status_tutor	char(1)	Not Null		

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_tutor_academico`, `id_dependencia`	Si	
Ref44	`id_dependencia`		

Descripción:

Tabla que almacena la información del tutor (docente) asignado al estudiante, cuya función es brindar su apoyo o tutoría dentro de la facultad de ciencias

Tabla: tutor_comunitario

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_tutor_comunitario	int(11)	Not Null		Si
id_comunidad	int(11)	Not Null		Si
cedula	int(11)	Not Null		
nombres_comunitario	char(50)	Not Null		
apellidos_comunitario	char(50)	Not Null		
cargo_comunitario	text	Not Null		
telefono_comunitario	char(12)	Not Null		
email_comunitario	text			
status_tutor	char(1)	Not Null		

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_tutor_comunitario`, `id_comunidad`	Si	
Ref818	`id_comunidad`		

Descripción:

Tabla que almacena la información del tutor asignado a un estudiante dentro de una comunidad (Tutor Comunitario)

Tabla: usuarios

Atributos de la tabla

Nombre	Tipo	Not Null	Único	P/K
id_usuario	int(11)	Not Null		Si
login	Char(20)	Not Null		
contrasena	Char(32)	Not Null		
nombres	Char(20)	Not Null		
apellidos	Char(30)	Not Null		
permisos	int(11)	Not Null		
tipo_usuario	Char(2)	Not Null		

Indices

Nombre del índice	En el campo	Único	
PRIMARY	`id_usuario`	Si	

Descripción:

Tabla que almacena a los usuarios que interactuarán con la herramienta de servicio comunitario, en este caso el usuario administrador de la aplicación.

A continuación la figura A3 muestra una captura de pantalla de la sección de administrar talleres del Módulo de Servicio Comunitario, específicamente la opción de agregar un nuevo taller.

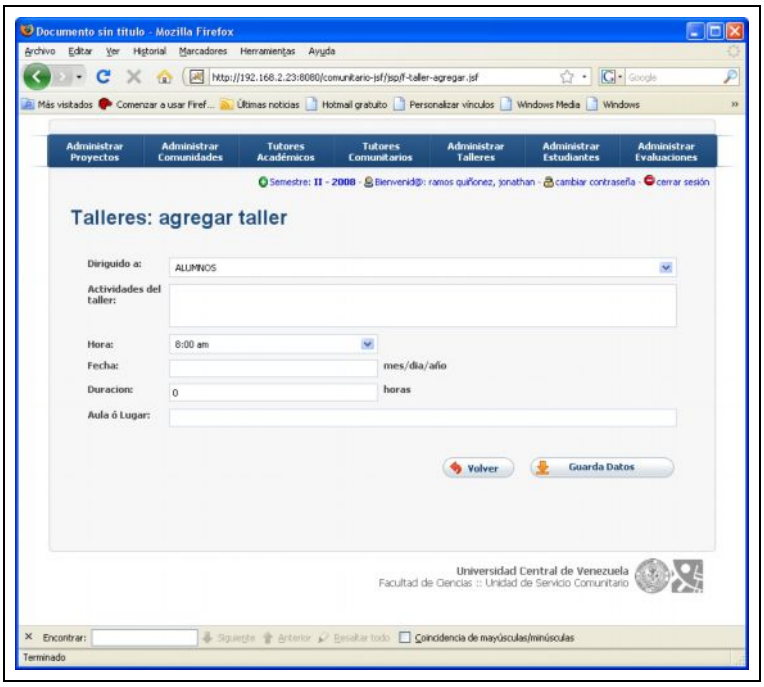

Figura A3: Módulo de Servicio Comunitario: Agregar nuevo taller

La figura A4 muestra una captura de pantalla de la opción cambiar contraseña del Módulo de Servicio Comunitario.

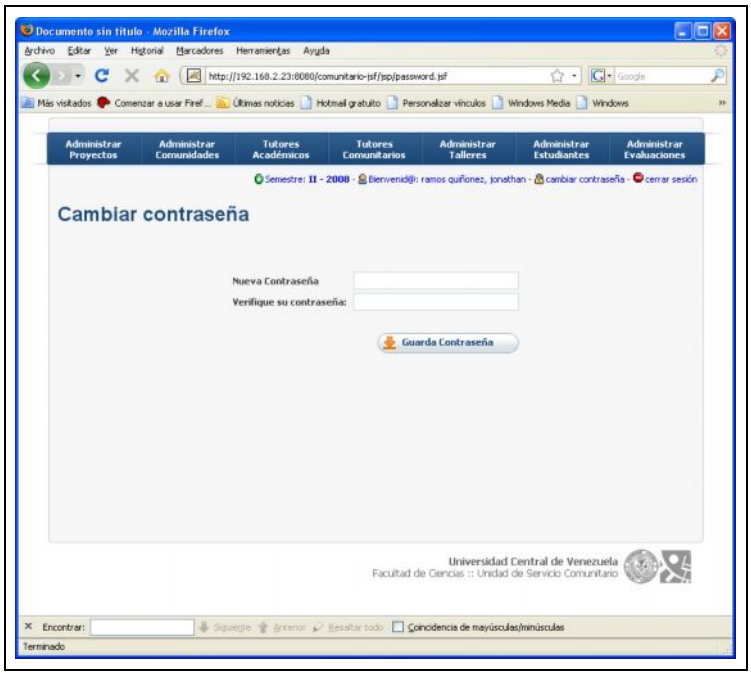

Figura A4: Módulo de Servicio Comunitario: Cambiar contraseña de usuario

Apéndice B: Cómo utilizar el plugin Metrics 1.3.6

Para utilizar el plugin de métricas es necesario tener un conocimiento básico del manejo del lenguaje de programación Java y el desarrollo en el entorno Eclipse. Primero debe seleccionarse el proyecto en el panel izquierdo y ubicar la opción propiedades, la cual despliega una ventana en donde se seleccionará la opción "Metrics" la cual muestra un checkbox que permite habilitar o deshabilitar el cálculo de las métricas en el proyecto.

Posteriormente para poder visualizar las métricas del código de una aplicación o programa en el entorno de desarrollo Eclipse, debe activarse la pestaña de visualización de las mismas navegando a través del menú en "Window -> show view" y seleccionando la opción other; una vez realizado estos pasos se muestra una ventana en donde seleccionamos el visor de las métricas, como se muestra en la figura B1.

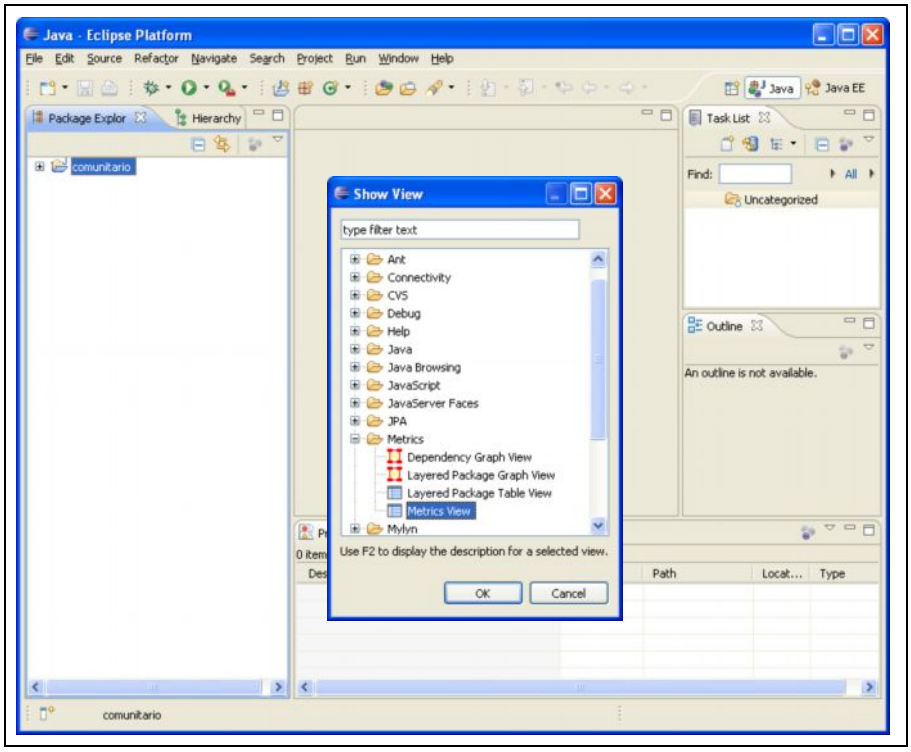

Figura B1: Pantalla de selección del plugin Metrics 1.3.6

El cálculo de las métricas de software será mostrado sólo cuando se haya compilado un proyecto el cual fue previamente seleccionado y habilitado para el cálculo de las métricas. Una vez llevado a cabo todo el proceso de cálculo de las métricas, son mostrados los resultados en la pestaña del editor Eclipse que ha sido habilitada para ello.

La figura B2 muestra el aspecto de la pestaña con los valores de las métricas calculadas, una tipografía en azul indica que la métrica calculada se encuentra dentro del rango normal de los valores esperados.

Sólo en el caso de métricas cuyo cálculo se encuentran por fuera de los valores normales, el color de la tipografía en la lista de métricas calculadas es resaltado en color rojo, anunciando una alerta sobre él código fuente escrito.

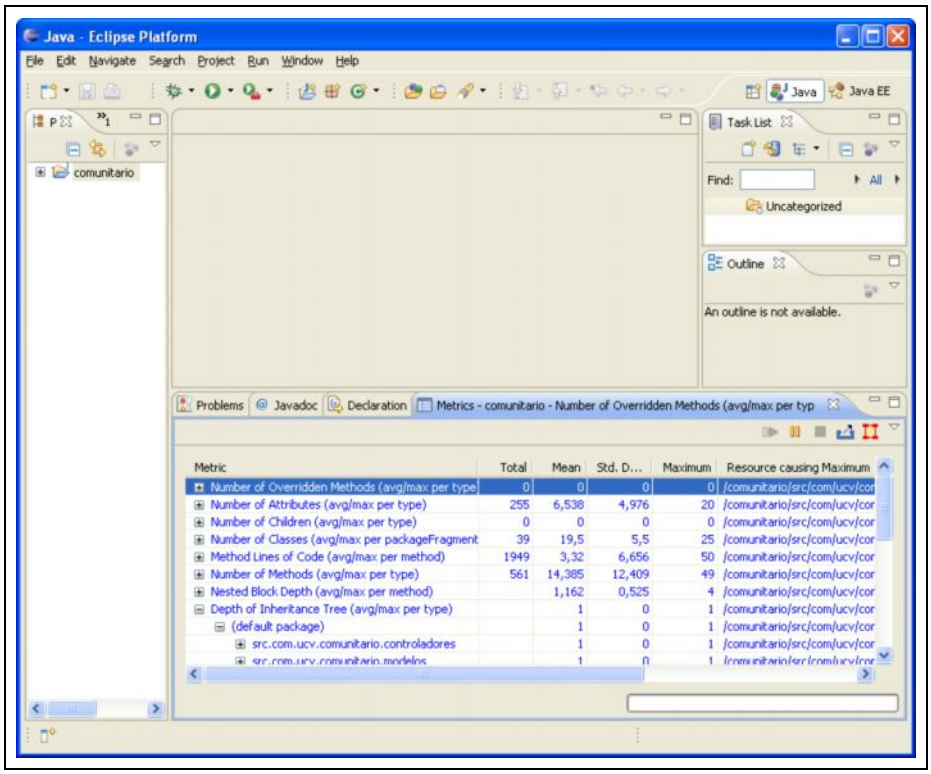

Figura B2: Panel de visualización de métricas

La tabla resultado dentro de la pestaña de cálculo de métricas se visualiza como una especie de árbol ramificado que muestras características que relacionan al paquete con las diferentes clases, métodos y métricas con los cuales puede estar relacionado dentro de la estructura de la aplicación.

Configuración de las Preferencias del plugin

Como se muestra en la figura B3, en la opción preferencias, se tiene opción de modificar el orden de visualización de las métricas calculadas por el plugin.

En la aplicación o proyecto puede ahora accionarse la métrica y las advertencias que aparecen en la vista o descripción de la tarea involucrada así como de los indicadores en donde se observa los métodos y los tipos para los cuales se están violando los rangos de la métrica.

Los valores mínimos y los máximos para cada métrica también se pueden fijar en la ventana de preferencias. Esta característica mencionada es inhabilitada por defecto, pero puede ser nuevamente habilitada en la sección de preferencias tal y como se muestra en la figura B4.

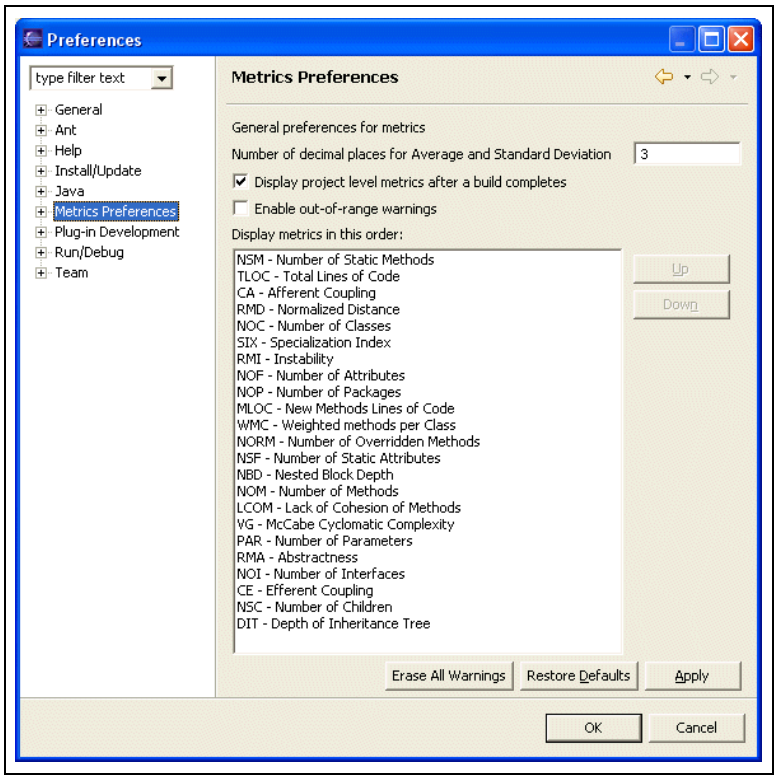

Figura B3: Opciones de preferencia para el plugin Metrics 1.3.6

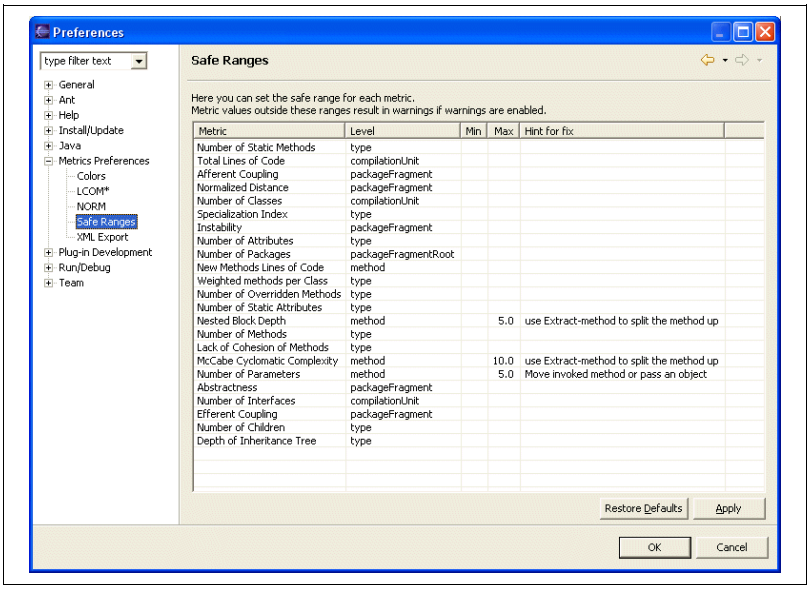

Figura B4: Preferencias personalización del rango de valores del plugin Metrics 1.3.6

Visualización mediante colores

La visualización de las métricas, como se ha mencionado anteriormente se resaltada en dos colores por defecto. El azul para los valores de la métrica dentro de los parámetros normales, el color rojo para las métricas cuyos valores se encuentran fuera de rango.

Cualquier color para visualización de los valores calculados de las métricas ó colores para las gráficas de dependencia pueden ser modificados también a través de la opción de preferencias del plugin como es mostrado en la figura B5.

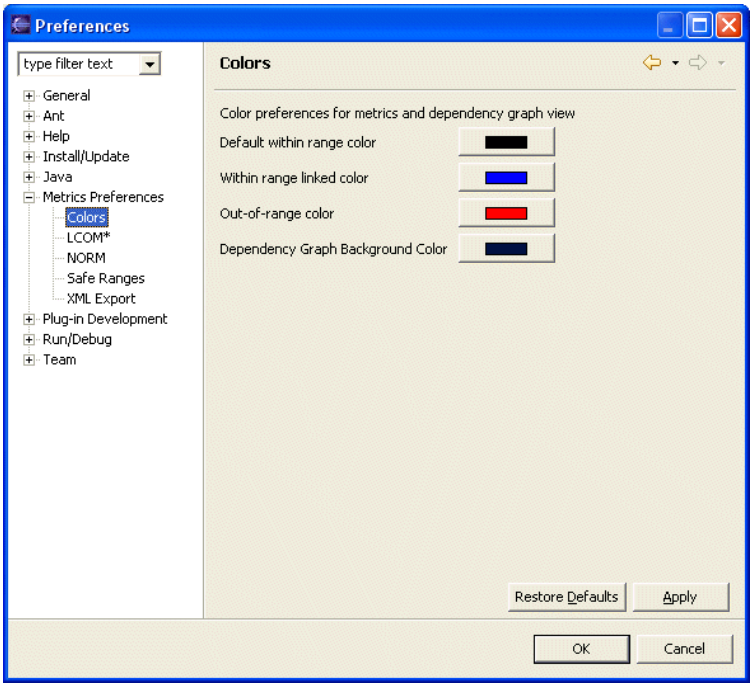

Figura B5: Pantalla configuración de colores en el plugin Metrics 1.3.6

Análisis de la dependencia de paquetes

El plugin incluye un analizador gráfico de la dependencia de paquetes de la aplicación. Para su visualización debe haber un proyecto previamente seleccionado.

Se puede ver un gráfico de las dependencias del paquete cuyas partes pueden ser enfocados y girados. Utilizando los botones de radio y Scrollbar para manipular el gráfico. Los rectángulos rojos y azules representan los paquetes, y las líneas que los unen sus dependencias. La figura B6 muestra un ejemplo de una gráfica de dependencia.

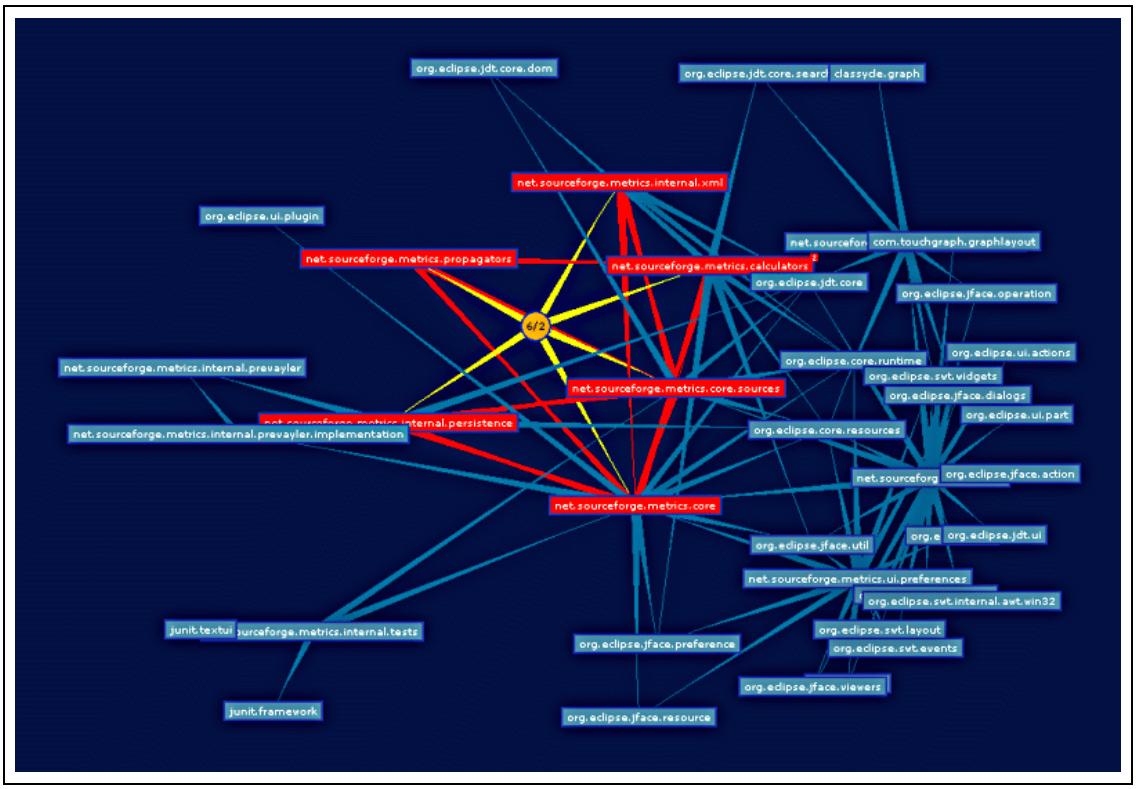

Figura B6: Ejemplo de gráfica de dependencia de una aplicación bajo Eclipse

La gráfica de dependencia muestra un análisis de los paquetes o componentes de la aplicación. Aquellos componentes que son accesibles unos con otros y que conforman un ciclo son coloreados en rojo, lo cual indica una fuerte conexión entre ellos. Mientras que aquellos paquetes que no intervienen en ningún ciclo son coloreados en azul.

Apéndice C: Estructura del documento XML de métricas de software

El plugin Metrics 1.3.6, tiene la opción de exportar los cálculos de métricas realizados a un archivo con formato XML como el mostrado en la figura C1 y que cumple con la siguiente estructura:

```
<?XML version="1.0" encoding="UTF-8"?>

<Metrics
       scope="SCOPE"
       type="Project"
       date="FECHA_ARCHIVO_XML"
       XMLns="http://Metrics.sourceforge.net/2003/Metrics-First-Flat">

   <Metric id="ABREVIARURA " description="NOMBRE ">

      <Values
         per = "TIPO"
         total = "TOTAL" avg="VALOR_MEDIO"
         stddev = "DEVIACION_ESTANDARD"
         max = "VALOR_MAXIMO_ENCONTRADO">

             <Value
                name = "NOMBRE_DEL_TIPO_MEDIDO"
                source = "NOMBRE_DEL_ARCHIVO"
                package = "NOMBRE_DEL_PAQUETE"
                value = "VALOR_DE_METRICA"/>

      </Values>

   </Metric>

</Metrics>
```

Figura C1: Estructura del archivo XML de cálculo métricas

Como puede apreciarse en la figura C1, el formato XML del archivo exportado por el plugin Metrics se encuentra conformado por cuatro sencillas etiquetas que se describen a continuación:

Metrics: es la etiqueta raíz del documento, esta etiqueta posee tres atributos clave:

- Scope: indica el nombre del objetivo sobre el cual se han realizados los cálculos de las métricas
- Type: indica el tipo del objetivo indicado por el atributo Scope, el mismo puede ser un proyecto, paquete o archivo
- Date: indica la fecha en la cual fue generado el archivo

Metric: es la etiqueta que indica la métrica calculada, a su vez posee dos atributos que son:

- Id: indica el nombre abreviado de la métrica
- Description: indica el nombre completo de la métrica

Values: ésta etiqueta contiene como atributos los valores globales de la métrica calculada, estos atributos son:

- Per: indica el tipo sobre el cual fue llevado el cálculo de la métrica, es decir, supongamos por ejemplo el cálculo de las métricas Número de Paquetes (NP) y Número de Parámetros (NOP), en cuyo caso sus respectivos tipos son "paquetes" y "métodos"
- Total: indica el valor total y general de la métrica
- Avg: indica el valor promedio calculado para la métrica
- Stddev: indica la desviación estándar de la métrica calculada
- Max: indica el valor máximo calculado para la métrica

Value: contiene el valor desagregado de la métrica calculada, es decir, el valor contenido dentro de la etiqueta values es la sumatoria del conjunto de todos los valores desagregados que se encuentran expresados en las etiquetas value. A su vez la etiqueta cuenta con los siguientes atributos:

- Name: indica el nombre del tipo medido, es decir el método, archivo o paquete sobre el cual se realizó el cálculo de la métrica
- Source: indica la ruta completa del paquete o archivo, el cual fue objeto de la medición
- Package: indica sólo el nombre del paquete donde se encuentra el objeto que fue sometido a medición
- Value: contiene el valor calculado de la métrica, identificado por los tres atributos anteriormente descritos

Apéndice D: Estructura del documento de resultados de httperf

La figura D1 ejemplifica los resultados obtenidos después de la ejecución de la herramienta de pruebas de rendimiento httperf sobre una página Web.

```
Total: connections 5000 requests 19417 replies 19222 test-duration 101.964 s

Connection rate: 49.0 conn/s (20.4 ms/conn, <=94 concurrent connections)
Connection time [ms]: min 1.6 avg 1030.3 max 3105.8 median 2003.5 stddev 1012.2

Request rate: 190.4 req/s (5.3 ms/req)
Request size [B]: 80.0

Reply rate [replies/s]: min 34.6 avg 190.7 max 200.0 stddev 37.0 (20 samples)
Reply time [ms]: response 7.1 transfer 0.5

CPU time [s]: user 17.43 system 81.37 (user 17.1% system 79.8% total 96.9%)
Net I/O: 17556.9 KB/s (143.8*10^6 bps)

Errors: total 195 client-timo 195 sockettimo 0 connrefused 0 connreset 0
```

Figura D1: Resultados de la ejecución de la herramienta httperf

Como se aprecia en la figura D1, los resultados obtenidos por la herramienta, se encuentran divididos en seis secciones. Cada línea de cada sección se describe a continuación:

Sección: Total de datos utilizados en el experimento

Total: connections 5000 requests 19417 replies 19222 test-duration 101.964 s

Ésta línea del archivo nos indica:

- Número de conexiones TCP realizadas
- Número de peticiones
- Número de respuestas
- Tiempo de duración del experimento o prueba

Sección: Resultado de conexiones

Connection rate: 49.0 conn/s (20.4 ms/conn, <=94 concurrent connections)
Connection time [ms]: min 1.6 avg 1030.3 max 3105.8 median 2003.5 stddev 1012.2

- La primera línea número promedio de conexiones simultáneas, así como el intervalo de tiempo entre conexiones y número máximo de conexiones simultáneas

- La segunda línea indica las estadísticas del tiempo de vida para las conexiones acertadas o exitosas

Sección: Resultados de peticiones

Request rate: 190.4 req/s (5.3 ms/req)
Request size [B]: 80.0

- La primera línea muestra el número de peticiones emitidas por segundo, así como el periodo de tiempo con el cuál las solicitudes http fueron emitidas. Sin conexiones persistentes, los resultados son similares a los resultados de conexión

- La segunda línea indica en promedio el tamaño de la petición http, expresada en bytes

Sección: Resultados de respuestas

Reply rate [replies/s]: min 34.6 avg 190.7 max 200.0 stddev 37.0 (20 samples)
Reply time [ms]: response 7.1 transfer 0.5
Reply size [B]: header 234.0 content 95050.0 footer 0.0 (total 95284.0)
Reply status: 1xx=0 2xx=19222 3xx=0 4xx=0 5xx=0

- La primera línea indica el valor mínimo, promedio, máximo y desviación estándar de respuestas por segundo, así como el número de muestras usadas para los cálculos en el experimento

- La segunda línea indica el tiempo promedio utilizado en respuesta a una petición y el tiempo promedio utilizado en recibirse totalmente la respuesta

- La tercera línea muestra en promedio la longitud de la respuesta expresada en bytes. La respuesta incluye cabeceras, contenido y pies de página

- La cuarta línea muestra un histograma de los códigos de estado recibidos

Sección: Uso de CPU

CPU time [s]: user 17.43 system 81.37 (user 17.1% system 79.8% total 96.9%)
Net I/O: 17556.9 KB/s (143.8*10^6 bps)

- La primera línea resume las estadísticas de CPU sobre el curso del experimento. Nótese que el total de utilización de CPU no esta cercano al 100%, por consiguiente los resultados obtenidos no pueden ser bien confiables

- La segunda línea muestra la entrada y salida de red usada para transferir y recibir datos a lo largo del experimento

Sección: Errores

Errors: total 195 client-timo 195 sockettimo 0 connrefused 0 connreset 0

Ésta línea del archivo indica:

- Número total de errores
- Número de clientes con timeout
- Número de clientes con socket timeout
- Número de conexiones rechazadas
- Número de conexiones reseteadas

Apéndice E: Módulo visor de métricas

Modelo de casos de uso

La etapa de diseño abarcó dos herramientas, una orientada a la administración del servicio comunitario la cual será el objeto de pruebas y una segunda herramienta para la visualización y comprensión de los datos obtenidos en la medición del objeto de pruebas.

La figura E!, muestra los casos de uso que fueron considerados para desarrollo de la herramienta de visualización y graficación de los datos de las mediciones del software.

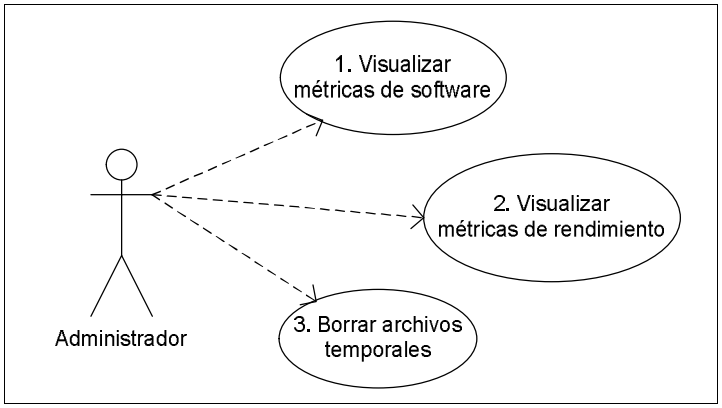

Figura E1: Casos de uso de la herramienta de visualización de métricas

Los casos de uso del módulo visor de métricas se describen a continuación:

Identificador:	1
Nombre:	Visualizar métricas de software
Actor Principal:	Usuario
Descripción:	Permite al usuario seleccionar dos archivos de métricas de software y visualizar la información obtenidas a través de gráficas comparativas de los archivos seleccionados. A la vez que permite ver el detalle de los datos con los cuales fueron graficados los resultados

Identificador:	2
Nombre:	Visualizar métricas rendimiento
Actor Principal:	Usuario
Descripción:	Permite seleccionar un archivo generado por la herramienta de muestreo de rendimiento a partir de la cual se obtiene la sumatoria de los datos estadísticos contenidos en el mismo para su adecuada presentación y posterior análisis por parte del usuario

Identificador:	3
Nombre:	Borrar archivos temporales
Actor Principal:	Usuario
Descripción:	Provee al usuario la facilidad de eliminar los archivos que se han subido al servidor de aplicación, durante el proceso de tratamiento de los documentos de métricas

Apéndice F: Testimonios

Además de la experiencia propia obtenida con el estudio y uso de los frameworks, el grupo tuvo la oportunidad de compartir también la experiencia de terceros, nos referimos a un par de desarrolladores que casualmente, en la actualidad están iniciándose en la aplicación de las tecnologías Java Server Faces y Struts 2, el primero de estos desarrolladores es David Besson, analista del departamento de tecnología del Servicio Nacional de Contrataciones, cuyo testimonio sigue a continuación:

"Soy programador con 8 años de experiencia en lenguajes como C++, C++Builder, Visual Basic, PHP, ASP y otros. Mis conocimientos de Java eran nulos. Yo había ingresado a mi actual trabajo para el desarrollo de un nuevo sistema donde tentativamente se desarrollaría bajo PHP. Durante esos días había una necesidad de un report Web Server, que exportara sus reportes tanto en formato PDF o XLS, y me llevó a una búsqueda de herramientas donde tomé a Jasper Report como solución, los cual tuve que investigar información relacionada con Java y como consecuencia termine topándome con un framework llamado Java Server Faces.

Estuve leyendo la documentación y me pareció muy interesante y hasta cierto punto intuitivo, razón por la que me anime a realizar un pequeño proyecto de prueba cuyo tiempo de desarrollo e investigación del JSF duro mas o menos el termino de una semana, con óptimos resultados ya que había desarrollado una pequeña aplicación que hace las cuatro operaciones básicas de base de datos.

Lo que me impresiono de este framework fue que el desarrollo sus aplicaciones Web se asemeja mucho al desarrollo de aplicaciones tradicionales cliente servidor de escritorio que se solía hacer en VisualBasic y C++Builder donde te puedes abstraer bastante de muchos operaciones típicas de la programación Web que para entonces yo conocía, esto te permite reducir los tiempos de desarrollo o reducir tu plantilla de programadores, lo que me pareció estupendo ya que actualmente en el SNC son mas los sistemas a programar y mantener que las personas disponibles a esa tarea. Otra característica que me gusto es su corta curva de aprendizaje, teniendo encuesta que todo lo relacionado a Java siempre tiene un buen grado de dificultad ya que el es un mundo aparte dentro de la programación, con sus propias normas y conceptos, yo he visto a JSF como un camino fácil y rápido para ir aprendiendo de la tecnología Java, así como una verdadera herramienta RAD para el desarrollo de aplicaciones Web."

Anexos

Anexo I: Manual de uso de httperf (línea de comandos)

El presente manual de uso de la herramienta httperf, fue traducida al español durante el desarrollo del presente Trabajo Especial de Grado.

Sinopsis

httperf [--add-header S] [--burst-length N] [--client I/N] [--close-with-reset]
[-d|--debug N] [--failure-status N] [-h|--help] [--hog] [--http-version S]
[--max-connections N] [--max-piped-calls N] [--method S] [--no-host-hdr]
[--num-calls N] [--num-conns N] [--period [d|u|e]T1[,T2]] [--port N]
[--print-reply [header|body]] [--print-request [header|body]] [--rate X]
[--recv-buffer N] [--retry-on-failure] [--send-buffer N] [--server S]
[--server-name S] [--session-cookie] [--ssl] [--ssl-ciphers L] [--ssl-no-reuse]
[--think-timeout X] [--timeout X] [--uri S] [-v|--verbose] [-V|--version]
[--wlog y|n,F] [--wsess N,N,X] [--wsesslog N,X,F] [--wset N,X]

Descripción

Httperf, es una herramienta utilizada para medir el rendimiento del servidor de Web. Este habla con el protocolo de HTTP tanto en su versión HTTP/1.0 como su versión HTTP/1.1, ofreciendo una variedad de trabajos generados. Mientras corre esta herramienta va guardando las trazas de un número de métricas de rendimiento que son agregadas en formas estáticas e impresas al final del trabajo o tarea que se ha ejecutado.

La operación más básica de httperf es generar un número determinado de peticiones o HTTP GET request y medir cuántas réplicas (respuestas) regresan al servidor y en que rata es que llegan las respuestas.

Importante para obtener correctos resultados es necesario correr más de un proceso por cliente/máquina. También es posible tener con pocos procesos, un background del servidor cliente y del cliente máquina. A continuación un conjunto de ejemplos de uso de la herramienta:

httperf --hog --server www

Este comando causa que httperf cree una conexión al host mediante www, enviando un requerimiento al documento raíz (en http://www), recibe la replica, cierra la conexión y entonces imprime una muestra o un aparte del rendimiento estático.

httperf --hog --server www --num-conn 100 --rate 10 --timeout 5

Inicialmente un total de 100 conexiones son creadas mediante este commando, y las conexiones son creadas en una rata de 10 conexiones por segundo, este comando acepta la opción abreviada de ”ra” por “rate”

httperf --hog --server=www --wsess=10,5,2 --rate 1 --timeout 5

Causa que el httperf genere un total de 10 sesiones en una rata de una sesión por segundo. Cada sesión consiste de 5 llamadas que son espaciadas por 2 segundos.

httperf --hog --server=www --wsess=10,5,2 --rate=1 --timeout=5 –ssl

Semejante al ejemplo anterior, excepto que el httperf contacta al servidor para conectarse por defecto en el puerto 443 (el puerto por defecto para la conexión SSL).

httperf --hog --server www --wsess=10,5,2 --rate=1 --timeout=5--ssl --ssl-ciphers=EXP-RC4-MD5: EXP-RC2-CBC-MD5--ssl-no-reuse—http version=1.0

Similar, httperf informará la excepción al servidor que solamente se puede seleccionar de dos suite de cifras (EXP-RC4-MD5 o EXP-RC2-CBC-MD5); además httperf usará la versión http 1.0, la cual requiere una nueva conexión TCP para cada requerimiento. También para los ids de sesión de SSL que no son reusados, como también el entero SSL para el establecimiento de la conexión de proceso (conocido como el apretón de manos de SSL) ocurrido para cada conexión.

Parámetros

La operación de httperf puede ser controlada a través de un número de opciones. La herramienta soporta ambas opciones de nombre de caracteres largos (arbitraria-longitud) y cortos (un-carácter). La opción corta es antepuesta con un simple guión (-), la opción larga con un doble guión (--). Múltiples cortas opciones pueden ser agrupadas. Por ejemplo '-vV' es equivalente a '-v –V', y las opciones largas también pueden ser agrupadas juntas en una única. Los parámetros para opción pueden ser especificados de manera siguiente, con un signo igual, y el parámetro valor, separando el nombre de la opción y el valor con un espacio en blanco.

--add-header=S

Especifica el String S a incluir como un requerimiento adicional del encabezado. Este es necesario para especificar el retorno de línea carro de la secuencia explicita en estudio. Este puede ser hecho usando la secuencia escape "\n". Esto hace posible incluir múltiples requerimientos en el encabezado.

--burst-length=N

Especifica la longitud del estallido. Cada estallido consiste en N llamadas al servidor. El exacto significado del parámetro depende del trabajo de carga generado. Para regular el trabajo de carga, ver la opción –wsess.

--no-host-hdr

Especifica aquel host del encabezador que puede no ser incluido cuando es emitido un requerimiento http.

--num-calls

Para sesiones orientadas al trabajo de carga, ver la descripción del a opción –wsess.

--client=I/N

Especifica aquella máquina donde httperf está corriendo sobre un cliente de Entrada-Salida de un total de N clientes. Debería estar en el rango de 0 a -1. Algunos de los trabajos de carga generados (por ejemplo –wset) usa la identidad cliente como un valor de para asegurar que no todos los clientes generan trabajo de carga perfectamente idénticos. Cuando desarrollas un trabajo que involucra cliente y máquina, esto es generalmente una buena idea para especificar esta opción.

--close-with-reset

Requiere que httperf cierre la conexión TCP enviando un RESET en vez de una conexión normal cuando se termina el abrazo o enlace. Al encender esta opción se puede tener infortunados efectos tales como una corrupción de datos, los bloques de control de TCP atascado, o malos resultados. Por esta razón la opción no puede ser usada al menos que sea absolutamente necesario.

-d=N ó --debug=N

Establece el nivel de depuración N de R. de los valores mas grandes que resultaran en mayor cantidad de producto.

--failure-status=N

Especifica que un estado de código de N respuestas http debe ser tratado como un fracaso (por ejemplo, una respuesta que retorne un código 504 sea considerado como un status de error al momento de la realización del experimento). Esta opción es actualmente soportada para trabajos de solamente carga. (ver las opciones wseess y wsesslog).

-h ó --help

Imprime un resumen de las opciones disponibles y sus parámetros.

--hog

Esta opción de muchos puertos TCP como necesarios, sin esta opción httperf es típicamente limitado a usar los puertos efímeros (en el rango de 1024 a 5000). Este limitado rango de puerto puede transformarse en un problema de cuello de botella, es por eso que es una buena idea especificar esta opción. Para trabajos correctos esta alternativa debe ser especificada cuando mides desde servidores NT, ya que así se evita una incompatibilidad de protocolo TCP entre NT y las máquinas UNIX.

--http-version=S

Especifica el String versión que debería ser incluido en el requerimiento que se envía al servidor. Esta opción puede ser establecida a "1.0" para forzar la generación de pedidos http/1.0. Establecer esta opción para algún valor aparte de "1.0" o "1.1". Puede resultar en un comportamiento inesperado.

--max-connections=N

Especifica que al menos N conexiones estén abiertas para cada sesión. Esta alternativa es significativa en conjución con las opciones –wsess y –wsesslog solamente.

--max-piped-calls=N

Especifica que al menos N llamadas realizadas en un túnel o Pipe, son hechas públicas en cada conexión. Esta alternativa es significativa en conjunción con las opciones –wsess y –wsesslog solamente.

--method=S

Especifica el método que debe ser usado cuando se hace una petición http. Si esta opción no es especificada, el método GET es usado. El método S puede ser una cadena arbitraria, pero es usualmente un GET; HEAD; PUT o POST.

--num-calls=N

Esta opción para los pedidos orientados a trabajos de carga. Esto especifica el total de número de llamadas a asumir en cada conexión antes antes de cerrar esta. Si el N es mas grande que el 1, el servidor debe soportar conexiones persistente. El valor por defecto para esta opción es 1. Si la longitud del estallido es establecida a R B, entonces las llamadas que están en el estallido de B son entonelizadas en cada llamada. Entonces, el total del número de llamadas será N/B (por conexión).

--num-conns=N

Esta alternativa es significativa para las peticiones sólo orientadas a trabajos de carga. Ella especifica el total de número de conexiones a crear: sobre cada conexión las llamadas son hechas públicas y especificada por opciones "número de llamadas" y "longitud del estallido".

--period=[D]T1[,T2]

Especifica el intervalo de tiempo entre la creación de conexiones o sesiones. Las conexiones son creadas por defecto en las opciones de sesiones wsess y wsesslog. Esta conexión/sesión que "interarrivan" pueden ser especificadas por "la rata de la opción", aunque más flexible, disponible para –period está el parámetro "d", el cual especifica el tiempo de interarrivo de la distribución. Si se omite o establece "d", a un periodo determinístico (por Ejemplo un arreglo de valores) "period", este es usado para especificar el parámetro interno R de T1 unidades usada sen ese momento. Si D es establecida con "e", una distribución exponencial (por ejemplo una distribución Poisson) es usada una vez como la media del tiempo de interarrivo de R. Si D es establecida a "u", una distribución uniforme sobre el intervalo [T1, T2] es usado para tiempo de interarrivo. Finalmente, si D es establecida a "v", un número de ratas puede ser especificado como sigue –period = vT1, D1, T2, D2…Tn, Dn Donde n es igual al NUM_RATAS en el httperf y Ti representan el período de tiempo (en consecuencia 1/rata) y la duración a mantener en aquella rata (y entonces el períod=v 1,2,0,5,4 generará 1 pedido sobre segundos para cada 2 segundos y entonces 2 pedidos/segundos se ejecutan en 4 segundos). En todos los casos un período de 0 resulta en la conexiones o sesiones que son generadas secuencialmente (una nueva conexión/sesión es iniciada tan pronto como una previa es completada. El valor por defecto para esta opción es 0. Note la especificación para el ejemplo, con una –rata=5 es equivalente a especificar un período --period= d=0.2. por especificción –period=? 1,3 el tiempo de interarrivo estará aleatoriamente elegido del intervalo entre 1 y 3 segundos. La específica secuencia (pseudo-) aleatoria de tiempo de interarrivo son idénticas de una corrida de Httperf a otra tan larga con los valores para el período –period y la opción para el ciente, --client.

--port=N

Esta opción especifica el número de puerto N sobre el cual el servidor Web esta escuchando un pedido de http. Por defecto, el httperf usa el puerto número 80.

--print-reply[=[header|body]]

Imprime una replica y resumen de los encabezados y cuerpo de las peticiones. La salida es directamente la salida estándar. Las líneas del encabezado son antepuestas por "RH", las líneas del cuerpo son antepuestas por "RB", y una replica de tamaño de resumen es prefijada por "RS". El prefijo es seguido por un número

serial que únicamente identifica la llamada la cual es una replica del número de líneas (".") El carácter que marca el comienzo de la actual replica de líneas. Para imprimir sólo replica de los encabezados, se pasa el argumento "header" a esta opción. Para imprimir sólo la replica del cuerpo, se pasa el argumento "body" a esta opción.

--print-request[=[header|body]]
Imprime los pedidos de la solicitud, cuerpo (si está presente), y un resumen. La salida es directamente la salida estándar. Las líneas de encabezado en las peticiones son antepuesta por "SH", las líneas del cuerpo del a solicitud son antepuestas por "SB" y el resumen de la solicitud es prefijado por "SS". El prefijo es seguido por un número serial que identifica únicamente la llamada a la cual es una replica del número de líneas. El caracter (".") marca el comienzo de la replica actual de las líneas. Para imprimir sólo el encabezado de una petición, se pasa el argumento "header" a esta opción. Para imprimir sólo el cuerpo de la petición, se pasa el argumento "body" a esta opción.

--rate=X
Especifica la rata fija en la cual las conexiones o sesiones son creadas. Las conexiones son creadas por defecto, si la opcion –wsess or –wsesslog ha sido especificadas. En ambos casos una rata de 0 resulta en una conexión o sesión que ha sido generada secuencialmente (una nueva conexión/sesión es iniciada tan pronto como una previa es completada. El valor por defecto para esta opción es 0).

--recv-buffer=N
Especifica el máximo tamaño de socket recibido por el buffers usado para recibir las respuestas HTTP. Por defecto el límite es 16KB. Un menor valor puede ayudar a los clientes con limitaciones de memoria mientras que un valor más grande puede ser necesario cuando se comunica con un servidor sobre un alto ancho de banda, y alta latencia de conexión.

--retry-on-failure
Esta opción es significativa para sesiones de trabajo de solamente carga (ver las opciones wsess y el –wseeelog).

--send-buffer=N
Especifica el tamaño máximo del socket enviado a los buffers. Por defecto, el límite es de 4KB. Un valor más pequeño puede ayudar a los clientes con limitaciones de memoria mientras que un valor más grande puede ser necesario cuando se generan de grandes pedidos a un servidor conectado a través de un gran ancho de banda, y de alta latencia de conexión.

--server=S
Especifica la IP del hostname o nombre de host del servidor. Por defecto, el nombre de host usado es "localhost". Esta opción debe ser siempre especificada; como esto es en general, no es una buena idea ejecutar el cliente y el servidor en la misma máquina.

--server-name=S
Especifica el nombre del servidor que aparece en el "Host:" del encabezado de todas las solicitudes enviadas por httperf. Sin esta opción, el nombre del host es especificado por la opción -server.

--session-cookie
Cuando esta opción está encendida, la gestión de cookies es activada por/sobre una

sesión base. Lo que esto significa es que si una respuesta a una solicitud que se generó por un período de sesiones contiene RX cookies y, a continuación, todas las solicitudes enviadas por X período de sesiones incluirá esta cookie también. En la actualidad, el manejador de cookies httperf apoya únicamente a una cookie por sesión. Si una segunda cookie es recibida, la nueva cookie se sobrescribe a la existente y un mensaje de advertencia es impreso si" - debug 1''está encendido.

--ssl

Especifica que todas las comunicaciones entre el servidor y httperf debería utilizar el protocolo Secure Sockets Layer (SSL). Esta opción sólo está disponible si httperf fue compilado con soporte SSL activado.

--ssl-ciphers=L

Esta opción sólo significativa si se está en uso de SSL (véase - ssl opción). Esta opción especifica la lista L, de las suites de cifrado que httperf puede utilizar en la negociación de una conexión segura con el servidor. Si la lista contiene más de una suite de cifrado, las cifras deben estar separados por dos puntos. Si el servidor no acepta alguna de las suites de cifrado, la conexión fallará y el establecimiento httperf saldrá inmediatamente. Si esta opción no se especifica cuando la - ssl opción está presente entonces httperf usará todas las suites de cifrado sslv3 proporcionadas por las biblioteca SSL.

--ssl-no-reuse

Esta opción sólo tiene significado si SSL y las sesiones están en uso (véase - ssl, - wsess, - wsesslog). Cuando se establece una conexión SSL el cliente recibe un identificador de sesión (Session ID) del servidor. En posteriores conexiones SSL, el cliente normalmente reutiliza Id este período de sesiones, a fin de evitar los gastos de repetir el (lento) establecimiento de la conexión o Handshake de SSL para establecer una nueva sesión SSL y obtener otro período de sesiones de identificación (aun cuando el cliente intenta volver a utilizar un identificador de sesión , el servidor puede obligar al cliente a volver a negociar un período de sesiones). Por defecto httperf reutiliza el período de sesiones de identificación a través de todas las conexiones en un período de sesiones. Si la opción ssl-no-reutilizable está en efecto, entonces httperf no reutiliza el identificador de sesión, y todo el establecimiento de conexión de SSL o Handshake de SSL, se realizará para cada nueva conexión en una de sesión.

--think-timeout=X

Especifica el tiempo máximo que el servidor puede tener para iniciar el envío de la respuesta de una determinada solicitud. Tenga en cuenta que este valor de tiempo se añade al valor de tiempo normal. Al acceder a contenido Web estático, por lo general no es necesario especificar esta opción. Sin embargo, al realizar ensayos de larga duración con los Scripts CGI, puede ser necesario usar esta opción para permitir mayores tiempos de respuesta. El valor por defecto para esta opción es de cero segundos, lo que significa que el servidor tiene que ser capaz de responder en el valor de tiempo normal.

--timeout=X

Especifica la cantidad de tiempo X que httperf está dispuesto a esperar por una reacción por parte del servidor. El tiempo se especifica en segundos y puede ser un número fraccionario (por ejemplo un tiempo de 3,5 segundos). Este valor de tiempo se utiliza cuando se establece una conexión TCP, al enviar una solicitud, para la espera de una respuesta, y al recibir una respuesta.

Si durante activación de una solicitud cualquiera para que no avance en el tiempo asignado, httperf considera que la petición que ha muerto, se cierra la conexión o valores asociados a la misma y se aumenta el período de sesiones de tiempo-cliente de error que se ha tomado en cuenta. El verdadero valor de tiempo para la espera de una respuesta es la suma de este tiempo.

--uri=S

URI S especifica que debe ser visitada en el servidor. Para algunos de los generadores de trabajo (por ejemplo, - WSET), esta opción especifica al prefijo de la URI que se accede.

--use-timer-cache

Esta característica permite al usuario especificar si necesita caché de tiempo o no. El tiempo no está en la caché por defecto, pero el usuario puede habilitar la caché si un mayor rendimiento es más importante que el tiempo exacto. Para los pequeños tamaños de respuesta, desactivar el temporizador de caché reduce el rendimiento de httperf en aproximadamente un 10%; para grandes tamaños de respuesta había poco o ningún efecto.

-v o --verbose

En este modo, la producción adicional, como la tasa de respuesta de las muestras individuales y conexión son impresas.

--version

Imprime la versión de httperf

--wlog=B,F

Esta opción puede utilizarse para generar una secuencia específica de la URI de accesos. Esto es útil para reproducir los accesos registrados en un archivo de registro del servidor, por ejemplo. El parámetro F es el nombre de un archivo que contiene el ASCII NUL o lista separada de las URIs que deben ser accesadas. Si el parámetro B se establece en "y", httperf retorna al comienzo del archivo cuando se alcanza el final de la lista (para la lista de URIs que es accedida repetidamente). Con B establecido a "n", la prueba se detendrá a más tardar cuando se alcance el final de la lista de URIs.

--wsess=N1,N2,X

Solicita la generación y medición de sesiones en vez de pedidos individuales. Una sesión consiste en una secuencia de estallidos los cuales son espaciados externament. Cada estallido consiste de un número prefijado de L llamadas al servidor (L es especificado por la opción burst-lenght). Las llamadas en el estallido son hechas públicas como sigue: En principio, una sola llamada es hecha pública. Una vez que la replica de la primera llamada ha sido completamente recibida, todas las demás llamadas que permanecen en el estallido son hechas públicas concurrentemente. Las llamadas simultáneas son hechas públicas otras llamadas son canalizadas sobre una conexión persistente o como una llamada individual sobre conexiones distintas. Una conexión persistente es usada si el servidor responde al primer llamado con una réplica que incluye una "línea de encabezamiento de conexión: fin". Si tal línea está presente son usadas distintas conexiones.

La opción especifica los siguientes parámetros:

- N1 es el número total del as sesiones para generar.
- N2 es el número de llamadas para la sesión y X es el tiempo de reflexión de usuario (en segundos) que separa consecutivamente cada estallido.

Por ejemplo las opciones "--wsess=100,50,10 -burst-lenght=5" resultaría en 100 sesiones con un total de 50 llamadas cada una. Ya que cada estallido tiene una longitud de 5 llamadas. Un total de 10 estallidos por llamada sería generado por cada sesión. El tiempo entre los estallidos de cada llamada sería de 10 segundos. Note que el tiempo de reflexión de usuario X denota el tiempo entre la recepción de la última réplica del a previa llamada del estallido y el envío de la primera solicitud del próximo estallido.

Un aprueba que involucra que las sesiones finalizan tan pronto como el número de solicitud N1 de sesiones ha fallado o ha sido completado. Una sesión es considerada a haber fallado si alguna operación en la sesión toma más largamente tiempo de espera que el especificado para estas opciones, se involucran en adición al tiempo de espera y al tiempo de reflexión para usuario; una sesión también falla si el servidor retorna una replicación con un código de estado que hace un único matching con la alternativa especificada para dicha opción, en la cual se indica --estado-fracasado o failure status.

--wsesslog=N,X,F

Este especifica un período de sesiones de trabajo similar al generador - wsess (lease la descripción anterior). Con - wsesslog sin embargo, muchos aspectos de las sesiones de usuario, incluyendo el número y secuencia de la URI, métodos de petición, tiempo de reflexión y los parámetros de longitud de estallido, pueden ser especificados en un archivo de entrada F. Los otros dos parámetros se mantienen de - wsess , renombrando a N, el número de sesiones para iniciar, y a X, el tiempo-de-reflexión-de-usuario (nota que esto se convierta en un tiempo predeterminado desde el archivo de entrada F también puede especificar un tiempo de reflexión para usuario sobre o por estallido base). Un pequeño ejemplo a continuación de un archivo de entrada puede mostrar más fácilmente los parámetros establecidos:

```
# session 1 definition (this is a comment)
/foo.html think=2.0
      /pict1.gif
      /pict2.gif
/foo2.html method=POST contents='Post data'
      /pict3.gif
      /pict4.gif

# session 2 definition
/foo3.html method=POST contents="Multiline\ndata"
/foo4.html method=HEAD
```

La descripción anterior se especifica 2 períodos de sesiones. La primera sesión se iniciará con una solicitud de / foo.html. Cuando la respuesta / foo.html retorna, una ráfaga de 2 solicitudes siguen (/ pict1.gif y / pict2.gif). Cuando la última de esas respuestas se reciben, un tiempo de reflexión usuario de aproximadamente dos segundos de tiempo, se inserta antes de la próxima petición de / foo2.html, la cual se publicará en breve. Esta solicitud se envía como un POST. El envió de datos puede contener entre uno o dos comillas. Saltos de línea pueden aparecer en los datos enviados como `` \ n''o `` como \ <CR>''. La respuesta / foo2.html es seguida por una ráfaga de solicitud / pict3.gif y / pict4.gif, que llegan a la conclusión de este período de sesiones. La segunda sesión se ha iniciado algún tiempo después de la primera, según lo especificado por la - tasa – o período de opciones.

La segunda sesión consta de 2 solicitudes separadas por defecto por el tiempo de reflexión asignado a usuario especificado por el parámetro de la opción X en - wsesslog. Si el parámetro de la N - wsesslog es mayor que el número de sesiones

se define en el archivo de entrada de RF, la definición de períodos de sesiones se utiliza varias veces hasta que N sesiones se hayan creado (es decir, las sesiones definidas son usadas en un round-robin).
Uno debe evitar usar --Wsesslog in conjunción con otras opciones de httperf que también controlan el comportamiento de sesión y el trabajo de carga de las URIs denominados concretamente –duración-de- longitud-del- estallido o burst-lenght, - wsess, - wlog, y - wset.

--wset=N, X

Esta opción puede ser utilizada para navegar a través de una lista de URIs en una determinada rata. El parámetro N especifica el número de URIs que deben ser generados y X especifica la rata en que los nuevos URIs son accesados. Una tasa de 0,25 significa algún URI puede ser accesado cuatro veces en una fila antes de pasar al próximo URI. Este tipo de patrón de acceso es útil en la generación de un trabajo de carga que induce una cantidad relativamente previsibles de tráfico en el disco de entrada/salida (E / S) del subsistema del servidor (Asumiendo que N y los archivos de acceso son lo suficientemente grandes como para superar el buffer caché del servidor). Los URIs generados son de la forma R prefijo / path. html, en donde el prefijo es el prefijo URI especificado por la opción - uri y el path o camino se genera de la siguiente manera: para las actividades de I i-ésimo archivo de trabajo establecido, se escribe i en decimal , anteponiendo el número con muchos ceros tantos sean necesario para obtener una cadena que tiene el mayor número de dígitos como R N -1. A continuación se introduce una barra de caracteres entre cada dígito. Por ejemplo, el 103 º en un archivo de conjunto de trabajo que consta de 1024 archivos daría lugar a un Path o camino de "0/1/0/3 ". Por lo tanto, si en la URI- el prefijo es / wset1024, entonces la URI estaría siendo accedida en / wset1024/0/1/0/3.html. En otras palabras, los archivos en el servidor tienen que ser organizado como un árbol decinario (10ary).