Universidad de las Ciencias Informáticas

FACULTAD 7

Herramienta para la administración de cluster en servidores de aplicaciones JBoss.

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

Autores: Reynier Manuel Germán Agüero

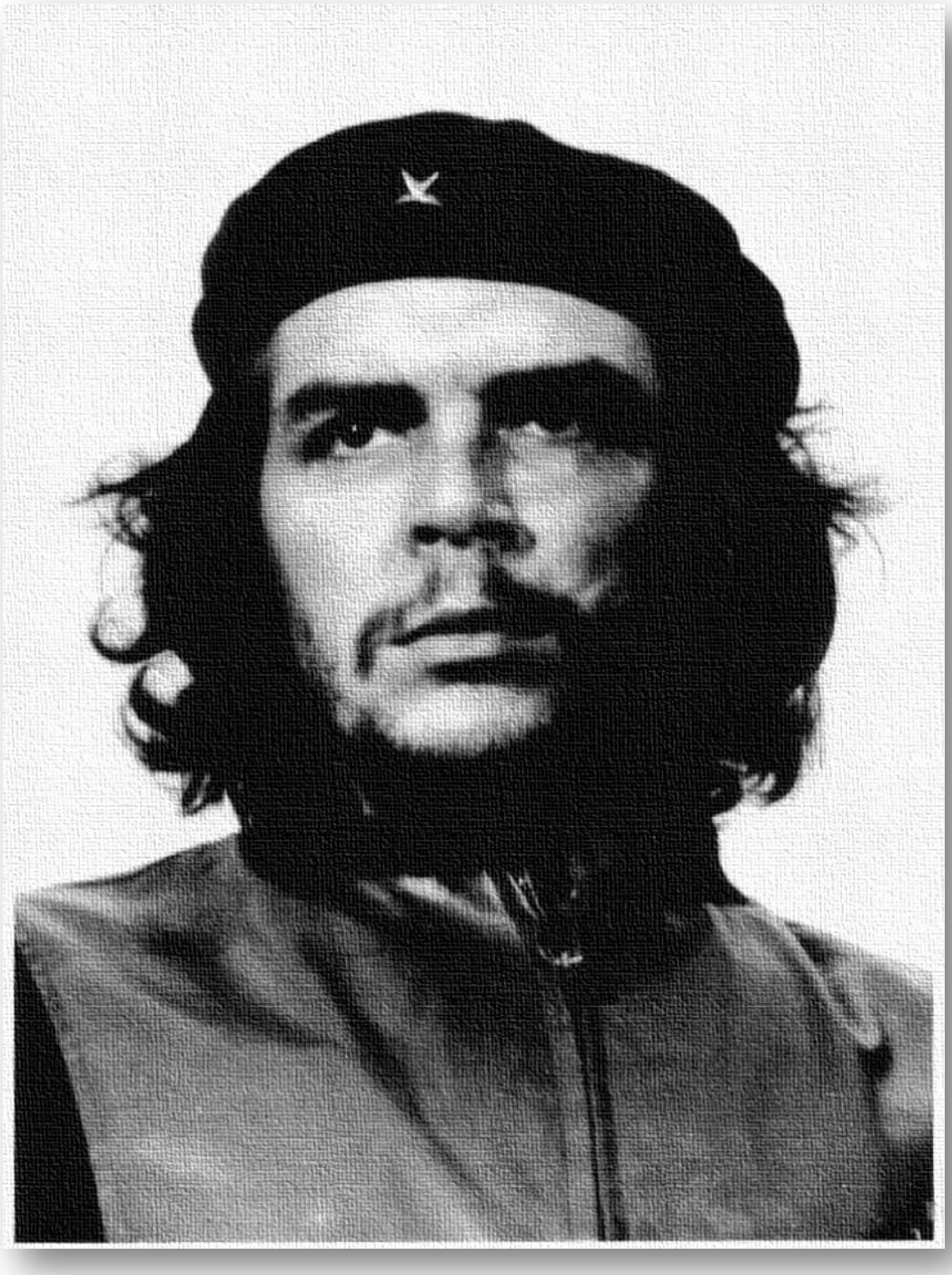
José Manuel Suau Lovelle

Tutor: MSc. Yovannys Sánchez Corales

Co-tutor: Ing. Néstor Brito Medina

La Habana, Mayo 2013

"Año 55 del Triunfo de la Revolución"



"La revolución es algo que se lleva en el alma, no en la boca para vivir de ella."

DATOS DE CONTACTOS

MSc. Yovannys Sánchez Corales: Profesor graduado de Ingeniero en Ciencias Informáticas en el año 2005 en la CUJAE. Categoría docente: Asistente. Ha impartido las asignaturas Programación III, Inteligencia Artificial, Arquitectura de Software y Administración de Servidores Linux. Se desempeña como arquitecto principal del Sistema Integral de Atención Primaria para la Salud (alas SIAPS) perteneciente al Centro de Informática Médica (CESIM).

Correo electrónico: yscorales@uci.cu

Ing. Néstor Brito Medina: Graduado de Ingeniero en Ciencias Informáticas en el año 2010 en la UCI. Recién Graduado en Adiestramiento.

Correo electrónico: nbrito@uci.cu

DEDICATORIA

De José

A mis padres y hermana por su apoyo en las buenas y en las malas en todos estos años, los quiero.

A mi tía Teresa, que no se encuentra entre nosotros, pero donde quiera que esté siempre la llevo en mi corazón. Por su cariño y consejos.

De Reynier

A mis padres, mi abuela y a mi hermana por haberme apoyado en estos años de estudio.

AGRADECIMIENTO

Agradecemos a todas a aquellas personas que nos han ayudado en el transcurso de este trabajo y a las que no también, por enseñarnos a que no todo es a pedir de boca. A los compañeros de cuarto y proyecto, por ayudar con su granito de arena en el desarrollo de esta tesis. Al tutor por exigirnos cada vez más.

De José

A mis padres por estar pendiente en todo momento de mis éxitos y fracasos, por ser un ejemplo para mí, y por amarme, los quiero. A mi hermana por existir.

A mis compañeros de cuarto y de batalla, Darian, Jandro y Alejandro, por aguantarme todos estos años, gracias. A mis amigos de Bayamo por estar siempre presente.

A mi compañero de tesis, por hacer más fácil la realización de la misma, siempre solventando las dificultades.

De Reynier

A mis padres por estar siempre aconsejándome en los momentos buenos y malos de la vida, también por haber confiado en mí en todos estos años de profesión. A mi abuela y mi hermana por ayudarme y apoyarme en cada año de la carrera.

A mi familia de la Habana, por haber estado presente en todos estos años de estudio y servicio militar,

A mis compañeros de apartamento, por haber compartido todos años.

A mi compañero de tesis, que desde el primer momento confió en mi para la realización de la misma.

A esta Revolución por darnos la oportunidad de estudiar y llegar a ser un profesional.

RESUMEN

El servidor de aplicaciones JBoss, es el primer servidor de código abierto, preparado para aplicaciones de negocios y combinado con una arquitectura orientada a servicios. Entre sus funcionalidades principales está el soporte de agrupamiento o *cluster*, esto permite la conexión de varios servidores, lo que facilita una mejor disponibilidad de los servicios.

El Sistema Integral para la Atención Primaria de Salud (alas SIAPS) utiliza un servidor de aplicaciones JBoss como parte de su funcionamiento, el mismo se encarga de administrar todas las solicitudes realizadas por los usuarios al sistema. Aunque el servidor está preparado para este tipo de peticiones se puede ver afectado debido a la cantidad de conexiones realizadas concurrentemente. Igualmente pueden existir fallos en el sistema alas SIAPS por ataques informáticos realizados a dicho servidor. Por estos problemas se decide realizar un *cluster* de servidores de aplicaciones JBoss para el sistema alas SIAPS.

Configurar un *cluster* con servidores de aplicaciones JBoss se convierte en una tarea tediosa a medida que aumenta la cantidad de servidores, pues de manera proporcional se debe comprobar el correcto funcionamiento de cada servidor. Es por ello que el objetivo de esta investigación es desarrollar una herramienta que administre el despliegue y la configuración de un *cluster* en el servidor de aplicaciones JBoss para el sistema alas SIAPS.

La herramienta es validada a través de la técnica Partición de Equivalencias, mientras que se utilizó la aplicación Jmeter para comprobar el correcto funcionamiento del *cluster*, así como, la disponibilidad y escalabilidad del sistema alas SIAPS.

Palabras claves: *cluster*, servidores de aplicaciones, JBoss, herramienta de administración.

ÍNDICE

ÍNDICE

INTRODUCCIÓN

En la actualidad, las Tecnologías de la Informática y las Comunicaciones (en lo adelante, TICs) (1) gestionan y monitorizan los procesos sociales y económicos más comunes de la vida. Dichas tecnologías han revolucionado el mundo contemporáneo con el objetivo de transformar los principales sectores de la sociedad, entre los que se encuentran la educación, las comunicaciones y la salud. La incorporación de las TICs en el campo de la medicina constituye un avance para mejorar la calidad de vida de los ciudadanos.

Los servidores de aplicaciones (AS por sus siglas en inglés) (2) constituyen una de las tecnologías más utilizadas por desarrolladores y empresas. A diferencia de los servidores web, los AS se distinguen por el uso extensivo de contenido dinámico y frecuente integración con bases de datos. Entre los principales beneficios se destacan la centralización y la disminución de la complejidad en el desarrollo de los sistemas.

El servidor de aplicaciones JBoss (en lo adelante, JBoss AS) es desarrollado por la empresa JBoss Inc y constituye el primer servidor de aplicaciones de código abierto (3). Entre sus funcionalidades (4) se destaca el soporte para el agrupamiento o *cluster* (5) (6). Dicha funcionalidad permite la conexión entre varios servidores de aplicaciones para facilitar una alta disponibilidad de los sistemas, así como mejorar la tolerancia al fallo de los mismos. Además permite una cantidad superior de conexiones que las soportadas por un solo servidor de aplicaciones, sin que se interrumpa el flujo de información.

En Cuba, desde hace algunos años se lleva a cabo una revolución informática para apoyar el desarrollo de la sociedad cubana. La Universidad de las Ciencias Informáticas (en lo adelante, UCI) constituye una de las principales instituciones encaminadas para este propósito. En ella se desarrollan varios programas que utilizan servidores de aplicaciones como Tomcat AS y el JBoss AS para mejorar su funcionamiento y despliegue.

El Sistema Integral para la Atención Primaria de Salud (en lo adelante, alas SIAPS) incluye el JBoss AS como parte de su funcionamiento. El mismo, tiene la misión de informatizar los procesos fundamentales relacionados con los pacientes en la atención primaria de salud. Dicho sistema es desarrollado por el Centro de Informática Medica (CESIM) perteneciente a la UCI.

En la actualidad, el sistema alas SIAPS utiliza una estrategia de despliegue conformada por dos nodos y dos niveles mediante una arquitectura cliente-servidor, en la que el cliente se conecta a un JBoss AS y la aplicación a su vez accede a otro servidor de base de datos. Esta infraestructura aunque simplifica el despliegue y el mantenimiento del sistema alas SIAPS, necesita la puesta en marcha y

disponibilidad permanente de dicho servidor, por lo que no soporta la tolerancia al fallo de conexiones o alguna deficiencia en el funcionamiento del mismo. Lo antes mencionado trae consigo que el sistema alas SIAPS pueda sufrir problemas relacionados con la disponibilidad.

Un mal funcionamiento del sistema alas SIAPS también puede estar condicionado por problemas en la cantidad de conexiones hacia dicha aplicación o por cambios realizados en la configuración que impliquen el reinicio del servidor de aplicaciones, lo que trae como consecuencia la pérdida de la sesiones de trabajo abiertas por los usuarios en ese momento. Además, al no existir un componente intermediario (*Middleware*) entre las máquinas clientes y el JBoss AS, todas las peticiones realizadas por los usuarios inciden directamente en el servidor de aplicaciones, que aunque soporta este tipo de solicitudes puede que el rendimiento y/o escalabilidad del sistema se vea afectado, debido al aumento de las conexiones realizadas.

Realizar la configuración de un *cluster* con varios JBoss AS constituye una labor complicada debido a que son muchos y variados los archivos de configuración. También aumentan las tareas de mantenimiento y comprobación de estado de cada JBoss AS del *cluster*. A su vez se necesita de un mecanismo para visualizar indicadores, como el estado de los servidores y el despliegue en tiempo real del sistema alas SIAPS.

De forma adicional se plantea que gran parte de la seguridad del sistema alas SIAPS incurre sobre el servidor de aplicaciones, lo que facilita la práctica de diferentes ataques informáticos como Denegación de Servicio (7), Cross Site Scripting (8) (9), entre otros. Por tanto la información sensible manipulada por el sistema puede estar comprometida.

La situación anterior arroja como **problema a resolver:** ¿Cómo mejorar la plataforma de despliegue del Sistema Integral para la Atención Primaria de Salud?

El **objeto de estudio** se centra en el: proceso de despliegue y configuración del *cluster*.

Como **objetivo general** se tiene**:** desarrollar una herramienta que administre el despliegue y la configuración de un *cluster* de servidor de aplicaciones JBoss en el Sistema Integral para la Atención Primaria de Salud.

El **campo de acción** lo constituye la: herramienta para el despliegue y configuración del *cluster* en servidores de aplicaciones JBoss.

Para lograr el objetivo propuesto se le dará cumplimiento a las siguientes tareas:

1. Identificación de las técnicas de *cluster* y balanceo de carga de peticiones entre servidores de aplicaciones.
2. Análisis de las distintas configuraciones del servidor JBoss.
3. Creación de un *cluster* de alta disponibilidad mediante servidores de aplicaciones JBoss.
4. Creación de una plataforma de virtualización que brinde soporte al *cluster*.
5. Conexión de un balanceador como punto de acceso al *cluster*.
6. Desarrollo de una herramienta informática que permita monitorear el *cluster*.
7. Validación del correcto funcionamiento de la herramienta desarrollada.

Los métodos de investigación utilizados se exponen a continuación:

Métodos Teóricos

- ✓ Histórico-lógico para investigar y analizar los orígenes y evolución de las herramientas para administrar *cluster*, con el objetivo de lograr una mejor comprensión del objeto y el campo de estudio propuesto.
- ✓ Análisis-sintáctico en el estudio de las principales técnicas y herramientas existentes para la creación de un sistema que administre los servidores.
- ✓ Inducción-deducción para el análisis y la definición de la estrategia a utilizar para el desarrollo de la herramienta a partir de los problemas detectados.

Método Empírico

- ✓ Observación directa del uso y administración del servidor en el sistema, y los resultados que esto arrojó, permitió tener un análisis del problema en cuestión.

El presente documento sigue una estructura como se muestra a continuación:

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS: contiene los aspectos esenciales para entender el entorno del problema a resolver. Se describen los conceptos fundamentales asociados al dominio del problema, sistemas similares existentes vinculados a la administración de *cluster* de servidores de aplicación, así como las tendencias y las tecnologías más usadas.

CAPÍTULO 2. CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS: se explican las configuraciones a realizar en el *cluster* del JBoss AS y las del servidor web Apache como balance de carga de peticiones para dicho agrupamiento.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS: se define el modelo de domino y los diagramas a realizar para un entendimiento mejor del sistema. También se argumenta la estructura del sistema, sus clases y como quedará el producto terminado.

CAPÍTULO 4. VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA: se argumentan los pasos a seguir para lograr que la herramienta sea desarrollada con la calidad requerida y con un mínimo margen de error, además que ofrezca un manejo fácil para el usuario.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS METODOLÓGICOS RELACIONADOS CON ADMINISTRACIÓN DE CLUSTER JBOSS.

Para la realización de esta investigación se exponen los fundamentos teóricos de los principales conceptos a tratar, dentro de los que se destacan los servidores web y de aplicación, las herramientas a utilizar en el desarrollo del sistema, así como la guía metodológica para lograr un mejor proceso de la presente investigación. Además, se especifican los tipos de despliegue y cómo se pueden utilizar. A continuación se detallan cada uno de los fundamentos teóricos relacionado con este trabajo.

1.1 Servidores web y de aplicaciones

Servidor web

Los servidores web (10) a través de una arquitectura cliente-servidor (11) apoyan el funcionamiento de la Red Informática Mundial (WWW - *World Wide Web*) (12). Por su parte los navegadores web se encargan de utilizar esta red, para proporcionar los recursos que almacena el servidor web solicitados por los usuarios mediante el Protocolo de Transferencia de Hipertexto (HTTP - *Hypertext Transfer Protocol*) o el Protocolo Seguro de Transferencia de Hipertexto (HTTPS - *Hypertext Transfer Protocol Secure*), que incluye un esquema de funcionamiento muy simple como se muestra a continuación:

1. Espera peticiones por el Protocolo de Control de Trasmisión (TCP – *Transmission Control Protocol*) indicado, el estándar por defecto para HTTP es el 80.
2. Recibe una petición.
3. Busca el recurso.
4. Envía el recurso para utilizar la misma conexión por la que recibió la petición.
5. Vuelve al segundo punto.

El último informe de la compañía de servicios de internet Netcraft (13), en abril del 2013, señaló que el número total de sitios que utilizan servidores web asciende a 649.072.682, de ellos se encuentran activos 186.986.142. Entre los servidores web más utilizados para el despliegue de sitios y portales web se destacan Apache, Microsoft, Nginx y Google según muestra la figura 1.1.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

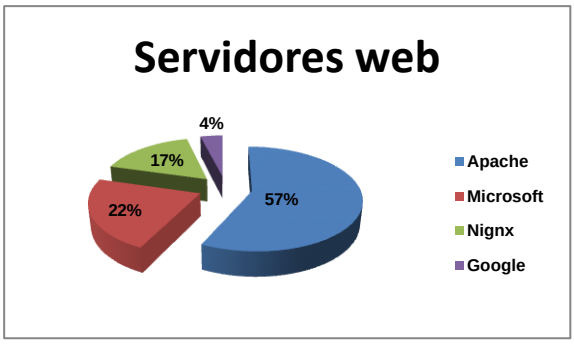


Figura 1.1: Porcentaje de los servidores web más utilizados en internet

Fuente: Elaboración propia.

Servidor de aplicaciones

Debido al éxito que ha tenido el lenguaje de programación Java, el término servidor de aplicaciones usualmente hace referencia a los servidores de aplicaciones de Edición Empresarial para Plataforma Java (Java EE- *Java Platform Enterprise Edition*) (14) (15). Dentro de esta gama de servidores se destacan los privativos WebLogic de Oracle (16) y WebSphere de IBM (17). Entre los servidores de aplicaciones libres más conocidos están TomEE de Apache (18), GlassFish de Oracle (19), Tomcat de Apache (20) y JBoss AS de JBoss Inc.

Los servidores de aplicaciones se distinguen por el uso extensivo de contenido dinámico y por su frecuente integración con bases de datos. Dichos servidores gestionan la mayor parte de las funciones de lógica de negocio y acceso a los datos de las aplicaciones, además que funcionan como un intermediario para la seguridad y el mantenimiento de las mismas. Presenta funciones multiproceso y multihilo para atender a distintas peticiones, además de enviar solicitudes a diferentes equipos en función de la carga y la disponibilidad.

Según Plumbr (21), el programa encargado de realizar estadísticas entre diferentes sistemas y aplicaciones, arrojó en marzo del 2013 un informe con alrededor de 1024 expedientes, de ellos 676 confirmaban la utilización de los servidores de aplicaciones Java EE en sus negocios, como se evidencia en la siguiente gráfica.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

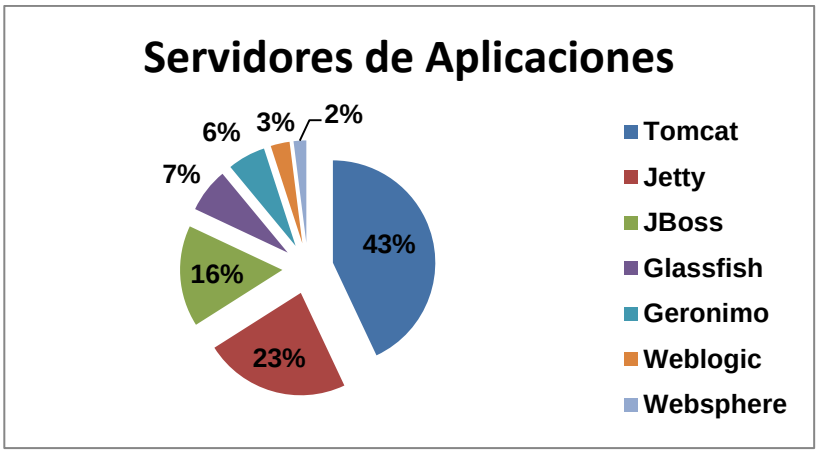


Figura 1.2: Porcentaje de los servidores de aplicaciones más utilizados

Fuente: Elaboración propia.

1.2 Breve reseña de la técnica de cluster.

El origen del término *cluster* y el uso de este tipo de tecnología son desconocidos pero se puede considerar que comenzó a finales de los años 50 y principios de los 60. La base formal de los *cluster* es el trabajo en paralelo que fue posiblemente inventado por Gene Amdahl de IBM (22), que en 1967 publicó lo que ha llegado a ser considerado como el papel inicial de procesamiento en paralelo. Las redes de conmutación de paquetes (23) fueron conceptualmente inventados por la corporación RAND en 1962 (24).

También se puede definir como un sistema de procesamiento en paralelo o distribuido (25). Consta de un conjunto de computadoras independientes, interconectadas entre sí, de tal manera que funcionan como un solo recurso computacional. A cada uno de los elementos del *cluster* se le conoce como nodo. En este trabajo el *cluster* se conforma de una plataforma de virtualización con tres nodos, cada uno con su respectivo JBoss AS configurado en *cluster* mediante el protocolo TCP.

Con la utilización del concepto de una red de conmutación de paquetes el proyecto ARPANET (26) logró crear en 1969 lo que fue considerado la primera red de computadoras básicas, basadas en el *cluster* de computadoras por cuatro tipos de centros informáticos.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

Formas de administrar un cluster

Desde el surgimiento de los *cluster* el hombre ha mejorado la disponibilidad y seguridad de los servicios que estos sostienen, a su vez se han creado herramientas para monitorear y brindar mantenimiento a los *cluster* con el objetivo de mejorar su funcionamiento y control. Estas tareas pueden ser administradas por algún programa como por ejemplo *Ganglia* (27), herramienta de monitoreo para *cluster*. El primer prototipo aparece con la versión 1.0 en el 2000, entre sus principales características se encuentra el almacenamiento de los datos resultantes de la comunicación *multicast* (28) en memoria que son trasmitidos a través de conexiones lógicas (*sockets*) (29) (30). La versión 2.x aparece en el año 2001, fue realizado en un lenguaje más potente y eficaz lo que le garantizó mayor estabilidad con respecto a la versión anterior.

Para la XIV Jornada de Paralelismo (31) en Leganés, España, en septiembre del 2003 se presentó un trabajo titulado “Estrategia de Instalación y Gestión de un Clusters” con software libre, donde se aborda la utilización de un sistema para gestionar y monitorear un *cluster* a través de algunos códigos (*script)*. Las principales funciones eran iniciar y apagar el *cluster* como si fuese un solo equipo a través de una Imagen Única de Sistema (SSI - *Single System Image*) mediante una petición del administrador y desde el servidor, el *script* apaga todos los nodos y luego apaga el servidor. Una vez que el servidor está listo y tiene en marcha todos los servicios necesarios, otro *script* se encarga de encender los nodos sin intervención de un operador.

Por otro lado se encuentra el sistema *Portable Batch System* (27) (PBS por sus siglas en inglés), es un sistema flexible para el balance de carga de peticiones y planificación de tareas. Creado por la compañía Veridian para la Administración Nacional de Aeronáutica y del Espacio (NASA - *National Aeronautics and Space Administration*) con el objetivo de solucionar la imposibilidad de manipular requerimientos complejos de administración de recursos de un sistema ya existente en la institución llamado *Network Queueing System* (NQS por sus singlas en inglés). El sistema se creó en marzo del 2003 con el estándar POSIX 1003.2d del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE - *Institute of Electrical and Electronics Engineers*).

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

1.3 Elementos esenciales para configuración de cluster JBoss AS

Para la realización de la presente investigación se estudiaron algunos servicios pertenecientes a la configuración de la técnica de *cluster* en los JBoss AS**.** En las siguientes sesiones se describen cada uno de estos servicios.

Servicio JNDI en cluster

La Interfaz de Nombrado y Directorio Java (JNDI - *Java Naming and Directory Interface*) (32) es una Interfaz de Programación de Aplicaciones (API - *Application Programming Interface*) de java para servicios de directorio. Permite descubrir y buscar objetos por un nombre a través de la red. El API presenta las siguientes características:

- ✓ Balanceo de carga de operaciones de nomenclatura.
- ✓ Descubrimiento automático de clientes de los servidores HA-JNDI[1] a través del servicio de red *multicast*.
- ✓ Búsqueda a través del *cluster*.

Replicación de sesiones

La replicación de sesiones HTTP es utilizada para replicar el estado asociado con la sesión del usuario a otro nodo del *cluster*, es decir, si el servidor donde se creó la sesión del usuario se detiene por alguna razón, dicha sesión será enviada a los demás nodos del *cluster*. La replicación es directamente manejada por JBoss AS, esto puede ocurrir en caso de iniciar el servidor de aplicación en la configuración "*all"* que se encuentra en el directorio "*server"* dentro del JBoss AS para habilitar el soporte de *clustering*. La figura 1.3 muestra lo antes mencionado.

[1] Alta disponibilidad de JNDI

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

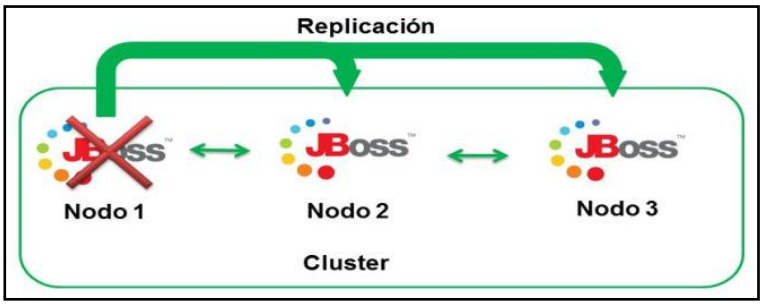


Figura 1.3: Replicación de sesiones.

Fuente: Elaboración propia.

Según la figura 1.3, el nodo 1 ha fallado y las sesiones de trabajo de los distintos usuarios se replican hacia los nodos 2 y 3 respectivamente.

Existe además la replicación de amigos (*buddy replication*) (33) en la que la sesión del usuario es replicada a solo un conjunto de nodos en vez de a todos. Lo anterior le permite al *cluster* escalar fácilmente sin un impacto extra en la memoria o en el tráfico de red con cada nodo añadido.

1.3.1 Sesión de Beans EJB en cluster

El JBoss AS soporta sesiones de *beans* (34) EJB (por su nombre en inglés *Enterprise JavaBeans*) en *cluster*, por lo que las peticiones para un *beans* son balanceadas por todo el *cluster*. Para los estados de *beans* se hace una copia de seguridad que se mantiene en uno o varios nodos del *cluster* proporcionando alta disponibilidad, en caso de que el nodo que maneja la sesión en particular falle o sea detenido. Para utilizar la sesión de *beans* EJB 3 solo se tiene que agregar a la clase de *beans* la anotación “@Clustered” como se muestra en la figura 1.4.

```
@org.jboss.ejb.annotation.Clustered
public class MyBean implements MySessionInt{
    public void test(){
        //Aquí la implementación de la clase
    }
}
```

Figura 1.4: Anotación @Clustered

Fuente: Elaboración propia.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

1.4 Tipos de despliegue del cluster

Con el propósito de realizar un despliegue de una aplicación web en un *cluster*, en la presente investigación se exploró los diferentes procedimientos para realizar dicho despliegue. Que a continuación se describen.

Servicios Singleton en cluster

Un servidor configurado en *cluster* de tipo *singleton* (también conocido como alta disponibilidad solitario o HA *singleton*) es un servicio que es desplegado en múltiples nodos en *cluster*, el nodo que inicia el servicio *singleton* es llamado nodo maestro y es el único que se mantiene activo dentro del *cluster*, mientras que los demás están a la espera. Cuando el nodo maestro falla o es apagado, otro nodo es seleccionado desde los nodos restantes y ocupa el lugar del nodo maestro, así el servicio es reiniciado en el nuevo nodo maestro. Esto es posible mediante las clases "*HASingletonMBeanExample"* y *"HASingletonController"* que posee cada nodo del *cluster*, las mismas están compuestas por los métodos "*startSingleton()*" y "*stopSingleton()"* así como una variable de control (verdadero o falso), que indica cuando un nodo es maestro. Con dichos métodos los nodos tienen la capacidad de elegir un nodo maestro dentro del *cluster*. De esta manera el servicio está siempre iniciado en un solo nodo como se muestra en la figura 1.5.

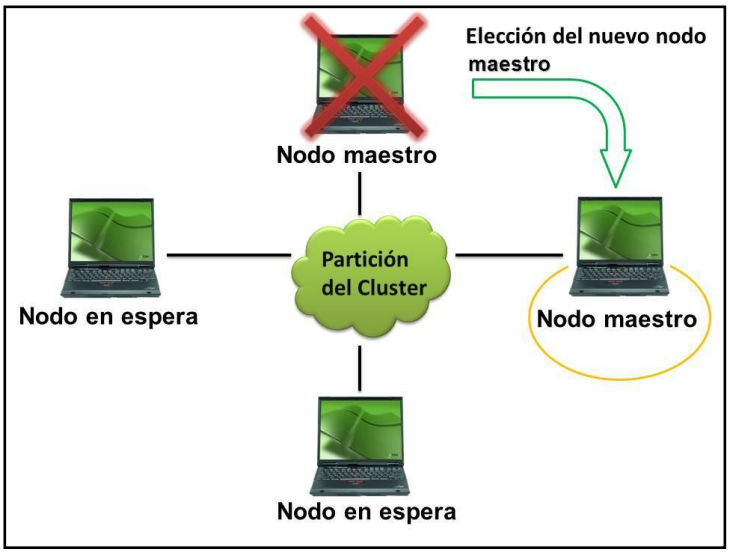


Figura 1.5: Tipología, fallo del nodo Maestro.

Fuente: Elaboración propia.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

Una estrategia utilizada para realizar un despliegue HA singleton es obtener un empaquetado ordinario con las extensiones (war, ear, jar) y desplegarlo en el directorio raíz donde se ubicó el JBoss AS, como se muestra a continuación.

- ✓ "$JBOSS_HOME" es la variable que indica donde se localiza el JBoss, dentro del mismo se accede a la configuración "*all*" para todas las funcionalidades del *cluster*.

Desde la carpeta "*all*" se accede al directorio "*deploy-hasingleton"* que se encarga de realizar el despliegue del tipo HA *singleton*. El contenido de "*deploy-hasingleton"* no es desplegado automáticamente cuando un JBoss AS inicia. La responsabilidad de desplegar el contenido de este directorio incurre en el *beans HASingletonDeployer*, su configuración se ubica en el archivo "*deploy-hasingleton-jboss-beans.xml*" que está contenido en el directorio "*deploy"*.

Farming deployment

Una manera simple de desplegar una aplicación en el *cluster* es usar el servicio de "*farming o farm"* (despliegue en granja). Al usar este servicio, se puede desplegar un empaquetado ordinario a la carpeta "*farm"* que se ubica dentro del directorio "*deploy"* de cualquier miembro del *cluster* y la aplicación automáticamente se replicará a través de todos los nodos. Si un nodo es agregado posteriormente, este iniciará las aplicaciones desplegadas en el *cluster*. Si la aplicación es eliminada desde el directorio "*farm"* en un nodo, el despliegue de la aplicación será removida localmente y luego eliminado de los restantes nodos.

1.5 Aplicaciones informáticas para administrar cluster

Las herramientas para administrar servidores se han convertido en un poderoso instrumento para muchos de los administradores de sistemas y arquitectos de software, debido a que ofrecen facilidad de manejo y control de los recursos. Lo anterior permite que los servidores administrados tengan un seguimiento según la carga y rendimiento en tiempo real.

En este epígrafe se hace un estudio de las aplicaciones informáticas existentes tanto a nivel nacional como internacional relacionada con la administración del JBoss AS. Se hace énfasis en la forma de despliegue y configuración del mismo.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

Herramienta para la instalación y configuración de clusters del Contenedor de Servlets Apache Tomcat

Esta herramienta (35) fue creada por la UCI en el año 2009 con el objetivo de garantizar la administración de los *clusters* establecidos para Tomcat y Apache 2. El programa se creó en el marco del convenio Cuba-Venezuela con el fin de automatizar todo el proceso de instalación y configuración que realiza el personal encargado de dar soporte a los proyectos productivos desplegados en estos servidores.

Dentro de las principales características que presenta son:

- ✓ Gestionar perfil de administración.
- ✓ Instalar y configuración servicios en los servidores.
- ✓ Gestionar instalación de la aplicación java.
- ✓ Gestionar seguridad de la aplicación web desplegada.

La herramienta se centra solamente en el *cluster* Tomcat AS, por lo que no es factible su utilización en la presente investigación, puesto que el objetivo se enfoca en el JBoss AS. Además no permite monitorear dichos servidores, así como la administración del funcionamiento de los mismos, dígase, iniciar, reiniciar, apagar, entre otras acciones.

Nagios XI

La herramienta informática Nagios XI (36) desarrollada en los Estados Unidos por la empresa que lleva el mismo nombre del producto, proporciona una monitorización completa del JBoss AS lo que le permite un controlar de sus atributos, además, del uso de memoria y el estado de los hilos.

Dentro de las principales características que presenta están:

- ✓ Disponibilidad incrementada de servidores, servicios y aplicaciones.
- ✓ Detección rápida de las interrupciones de red, fallas de protocolo, procesos fallidos, servicios y trabajos por lotes.

Nagios XI cumple parcialmente con el objetivo propuesto en la presente investigación, pues aunque se puede establecer un monitoreo constante del JBoss AS y servicios que este presta, no es posible

administrar el *cluster* de JBoss AS. Además no se puede adquirir gratuitamente puesto que requiere un costo de adquisición.

JBoss Operations Network.

Es una herramienta informática (37) desarrollada por la empresa RedHat, proporciona un control centralizado, ya sea despliegue, gestión o supervisión de las aplicaciones y servicios que se encuentran en el *JBoss Enterprise Middleware* (38). Con la utilización de JBoss Operations Network (a partir de ahora JBoss ON) se pueden visualizar todos sus ambientes además de ver el estado de las aplicaciones y servicios que opera. También brinda la posibilidad de analizar componentes específicos del mismo y solucionar problemas que se detecten.

La aplicación presenta un costo adicional para su adquisición, debido a sus restricciones y al alto precio de su licencia.

1.6 Herramientas y tecnologías

Ambiente integrado de desarrollo

Los Entornos de Desarrollos Integrados (IDE - *Integrated Development Environment*) facilitan el desarrollo ágil de programas informáticos para la web o sistemas de escritorios. Entre la gama de entornos de desarrollo existentes se destaca el sistema privativo Visual Studio (39) (40), por ser uno de los más utilizados, es de fácil manipulación y posee un lenguaje de programación entendible para cualquier persona. También se destaca el Eclipse (41), por su robustez, consistencia e integración con diferentes marcos de trabajos (*framerwork)* como por ejemplo JBoss Seam (42).

Por otro parte NetBeans (43) (44) (45) es un software libre y gratuito sin restricciones de uso, hecho principalmente para el lenguaje de programación Java. Entre sus características se tiene que es multilenguaje, multiplataforma, permite desarrollar casi todo tipo de aplicaciones java como pueden ser aplicaciones móviles. Es una plataforma para construir aplicaciones completas para el cliente, además permite crear ventanas, menús y barras de herramientas fácilmente, así como facilita el diseño de las interfaces del sistema.

Características principales:

- ✓ Mejoras en el editor de código.

- ✓ Instalación y actualización más simple.
- ✓ Características visuales para el desarrollo web.
- ✓ Soporte para PHP.

Es seleccionado NetBeans para el desarrollo de la herramienta propuesta en esta investigación debido al fácil manejo de las interfaces con la librería *swing* (46). Entre los componentes visuales que incluye, se tienen las plantillas de interfaz de usuario (*look and feel*) que se ajustan al entorno del sistema operativo donde se encuentre instalado, siendo de esta manera más amigable para el usuario. Además, posee una base de datos embebida que facilita el desarrollo de la herramienta, puesto que no necesita un sistema de administración para su utilización.

Herramienta de modelado

Las herramientas de modelado de sistemas se utilizan para la visualización en diagramas de los productos que se van a desarrollar. Entre dichas herramientas se destacan Enterprise Architect (47), la misma facilita la realización de las etapas del ciclo de vida de un programa informático tales como análisis, diseño, pruebas y mantenimiento. Rational Rose (48) es otra de las herramientas destinadas al modelado de sistemas, con su utilización se puede generar código en diferentes lenguajes de programación (Java, C++ y C#) desde un diagrama realizado. También brinda la posibilidad de generar documentación referente al sistema informático, para así obtener una ayuda en cuanto a la implementación del mismo. Además, al obtener el código fuente de un producto informático es posible adquirir los diagramas mediante la ingeniería inversa.

Visual Paradigm (49) es una herramienta de Ingeniería de Software Asistida por Computadora (CASE - *Computer Aided Software Engineering*) que soporta el Lenguaje Unificado de Modelado (UML - *Unified Modeling Language*) y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un sistema. Las ventajas que proporciona Visual Paradigm por UML son:

- ✓ Dibujo: Facilita el modelado de UML, puesto que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, porque la misma se ajusta al estándar soportado por la herramienta.
- ✓ Corrección sintáctica: Evita errores de sintaxis en el modelado.
- ✓ Coherencia entre diagramas: Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, para evitar duplicidades.

- ✓ Integración con otras aplicaciones: Permite integrase con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.

La herramienta Visual Paradigm por UML es la que más se ajusta al perfil de trabajo para darle solución al problema planteado por las ventajas antes mencionadas, además de ser libre de distribución por lo que los elementos como costo de desarrollo y mantenimiento pueden ser reducidos.

Sistema de gestión de bases de datos

Un sistema gestor de base de datos (SGBD) es el encargado de almacenar, modificar y extraer información de una base de datos. Además, brinda herramientas para insertar, editar y eliminar datos, así como proporciona métodos para mantener la integridad de los mismos. Entre la variedad de SGBD existentes se pueden mencionar SQL Server (50), Mysql (51) y Postgresql (52), siendo este último de código abierto y uno de los más utilizados. Postgresql utiliza una arquitectura cliente/servidor con multiprocesos que garantizan la estabilidad del sistema, esto implica que si ocurre algún fallo en un proceso no se afecten los demás.

Apache Derby (53) es un SGBD de código abierto implementado en su totalidad en Java. Puede ser embebido en aplicaciones de este tipo de lenguaje y está disponible bajo la licencia Apache en su versión 2.0. Algunas de las ventajas clave incluyen:

- ✓ Tiene un tamaño reducido de 2,6 megabytes para el motor de base y el controlador integrado llamado Conexión Java a Base de Datos (JDBC - *Java Database Connectivity*).
- ✓ Está basado en el Java, JDBC y SQL estándares.
- ✓ Ofrece un control integrado JDBC que permite incorporar Derby en cualquier solución basada en Java.
- ✓ Es compatible con el más familiar modo cliente-servidor con el *Derby Network Client* y Server JDBC de *Derby Network*.
- ✓ Es fácil de instalar, implementar y utilizar.

El gestor de bases de datos Apache Derby resulta ser el adecuado para almacenar los datos manipulados por la herramienta, debido a que se corresponden a las configuraciones del *cluster* y el acceso al sistema. La base de datos se encuentra embebida en la herramienta y no requiere una herramienta para administración gráfica.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO METODOLÓGICOS RELACIONADOS CON LA ADMINISTRACIÓN DE CLUSTER JBOSS

Lenguajes de programación de alto nivel

Los lenguajes de programación de alto nivel se caracterizan por expresar los algoritmos de una manera más fácil y entendible para el usuario. Dentro de estos se destaca el lenguaje C Sharp (C#) (54) (55), el mismo está desarrollado y estandarizado por Microsoft como parte de su plataforma .NET para el desarrollo de aplicaciones. Este lenguaje incorpora el paradigma de la programación (56) orientado a objetos para el desarrollo de aplicaciones y programas informáticos más eficientes.

Java es un lenguaje de programación de la compañía Sun Microsystems actualmente perteneciente a Oracle Corporation. Es la tecnología subyacente que permite el uso de programas punteros como herramientas, juegos y aplicaciones de negocios. Es utilizado por los entornos de desarrollo integrado Netbeans y *Eclipse* entre otros. La plataforma Java EE utiliza este lenguaje de programación para sus proyectos, enmarcándose en la creación e implementación de aplicaciones empresariales. Además, consta de un conjunto de servicios, API y protocolos que proporcionan la funcionalidad necesaria para desarrollar aplicaciones basadas en web en varios niveles.

Herramienta para realizar pruebas de rendimiento

Con el objetivo de liberar un sistema se le deben realizar pruebas para comprobar su correcto funcionamiento. Para esto existen herramientas que se encargan de monitorear diferentes estados y configuraciones del producto a examinar, de esta manera se pueden arrojar resultados con los que el cliente puede determinar si cumple con los resultados esperados. Entre las herramientas existentes se encuentran Visual Studio Test Professional (57), el mismo es un conjunto de herramientas especializadas para equipos de control de calidad. Funciona conjuntamente con el software Visual Studio para desarrolladores, lo que permite una colaboración efectivo entre programadores y evaluadores durante todo el ciclo de vida del desarrollo de la aplicación. Entre sus característica se encuentra la posibilidad de probar y depurar a fondo el código de una aplicación.

También existe la herramienta Selenium (58), es un entorno de pruebas de software para aplicaciones basadas en la web. Permite grabar y reproducir lo que posibilita crear pruebas sin usar un lenguaje de programación. Además, realiza pruebas en diferentes lenguaje entre los que se encuentra java, C# y Python.

Por último la herramienta Jmeter (59) escogida para comprobar el correcto funcionamiento del *cluster*. La misma permite realizar pruebas de carga para analizar y medir el desempeño de las aplicaciones

web. Dicha herramienta posibilita verificar el rendimiento en recursos estáticos y dinámicos como pudieran ser archivos y bases de datos bajo diferentes tipos de carga concurrentemente, con la posibilidad de obtener una conclusión más acertada. También muestra las operaciones fallidas durante el proceso de pruebas. Además, posee un funcionamiento de hilos que permite realizar varias operaciones al mismo tiempo, lo que brinda diferentes estadísticas de la prueba que se realiza.

Plataforma de virtualización

La virtualización es la forma de crear a través de un software una versión virtual de algún recurso tecnológico, dígase una plataforma de *hardware*, un sistema operativo, un dispositivo de almacenamiento o recursos de red. También se puede definir como una abstracción de los recursos de una computadora llamado Monitoreo de Máquinas Virtuales (VMM - *Virtual Machine Monitor*). En la actualidad existen distintos sistemas para virtualizar entre los que se encuentran Oracle VM Virtual Box (60), Windows Virtual PC (61) y OpenVZ (62).

Virtual Box es una herramienta para arquitecturas de x86/amde64 e Intel64. Permite instalar sistemas operativos adicionales como GNU/Linux, Mac OS X (63) y Microsoft Windows. Por otro lado Windows Virtual PC emula un *hardware* sobre el cual funcione un determinado sistema operativo. También se caracteriza por tener licencia privada lo que dificulta su utilización en los distintos entornos. Estas herramientas se caracterizan por consumir grandes recursos de las máquinas donde fueron instaladas, por lo que se requiere de un *hardware* potente para virtualizar varios sistemas operativos.

Open VZ constituye una de las mejores opciones para virtualizar en los sistemas operativos Linux. Utiliza una arquitectura de paravirtualización lo que permite realizar la virtualización a nivel de sistema operativo. Ofrece más rapidez en su funcionamiento que otros sistemas debido a que no simula completamente el *hardware*. Comparte el mismo núcleo que el sistema operativo anfitrión, por lo que permite un mayor ahorro de recursos, y consume solo el uno o dos por ciento de recurso del CPU. Por estas razones se escoge esta herramienta para el despliegue de las máquinas virtuales que contendrán los JBoss AS.

1.7 Guía para el desarrollo de software

Una metodología de desarrollo de software (64) es un procedimiento, una técnica, la cual ayuda a estructurar, planear y controlar el proceso de desarrollo de productos de software. Cuando se utiliza una buena metodología de software se garantiza la reducción de los costos y la eliminación de retrasos

en los proyectos, para conducir al mejoramiento de la calidad del producto. Entre las principales metodologías se destaca el Proceso Unificado de Rational (RUP - *Rational Unified Process*) (65), Programación Extrema (XP - *eXtreme Programming*) (66) y Scrum (67).

Con el objetivo de alcanzar el nivel 2 del Modelo de Madurez de Capacidad Integrado (CMMI - *Capability Madurity Model Integration*) la UCI llevó a cabo un proceso de mejora, que se utiliza como guía de apoyo para la metodología utilizada en del software. CMMI es un modelo para la mejora y evaluación de procesos enmarcado en el desarrollo, mantenimiento y operación de sistemas de software.

En la UCI, y específicamente en el proyecto Sistema Integral para la Atención Primaria de Salud, para llevar a cabo el proceso de gestión de todos los requisitos del proyecto es utilizado el documento IPP-3510:2009 Libro de Proceso para la Administración de Requisitos perteneciente a una de las áreas del proceso de mejoras, cuyo objetivo es definir el proceso de administración de requisitos. Este documento establece el ciclo de vida a seguir asociado a los proyectos involucrados en el proceso de mejora y consta de 9 fases (Estudio Preliminar, Modelación del Negocio, Requisitos, Análisis y Diseño, Pruebas Internas, Pruebas de Liberación, Despliegue y Soporte). En este se establece por cada fase la relación con los subprocesos descritos en el libro de procesos específico para el área de Administración de Requisitos.

Consideraciones del capítulo

El tipo de despliegue en granja (*farming*), resultó ser el más conveniente para esta investigación, pues la distribución de alas SIAPS en todos los nodos al mismo tiempo evita pausas extensas y por ende cuellos de botellas.

La réplica de sesiones posibilita mantener las sesiones activas de los usuarios, por lo tanto es evitable el reinicio innecesario del servidor de aplicaciones así como las pérdidas de los trabajos realizados por los usuarios.

Las herramientas y el lenguaje escogido para la construcción del sistema de administración de *cluster* JBoss AS resultan idóneos, pues favorecen las etapas pactadas en la guía definida para los proyectos que están involucrados en el proceso de mejoras llevado a cabo en la Universidad, cuya definición se basa en lo estipulado por CMMI en su nivel dos y por la experiencia de utilización de la metodología de desarrollo RUP.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

Para la realización del *cluster* entre los JBoss AS en la presente investigación se dispone de tres servidores lógicos, además un tercer servidor web que utiliza el módulo jk para balancear la carga de solicitudes entre los JBoss AS. A continuación se detallan los pasos para configurar un *cluster* con JBoss AS y un servidor web Apache.

2.1 Esquema de integración general

La integración del *cluster* formado por JBoss AS con el servidor web Apache sigue un esquema que admite una adecuada gestión en las conexiones de los clientes y mayor seguridad en las transacciones del sistema. El mismo está conformado por una plataforma de virtualización Open VZ con tres nodos, cada uno de los cuales contiene un JBoss AS configurado en *cluster*, los mismos están conectados a un servidor web Apache que funciona como punto de acceso y balance de peticiones. Finalmente se posee una herramienta capaz de administrar de forma remota el *cluster* previamente creado. A continuación la figura 2.1 muestra el esquema de integración general administrado por la herramienta desarrollada.

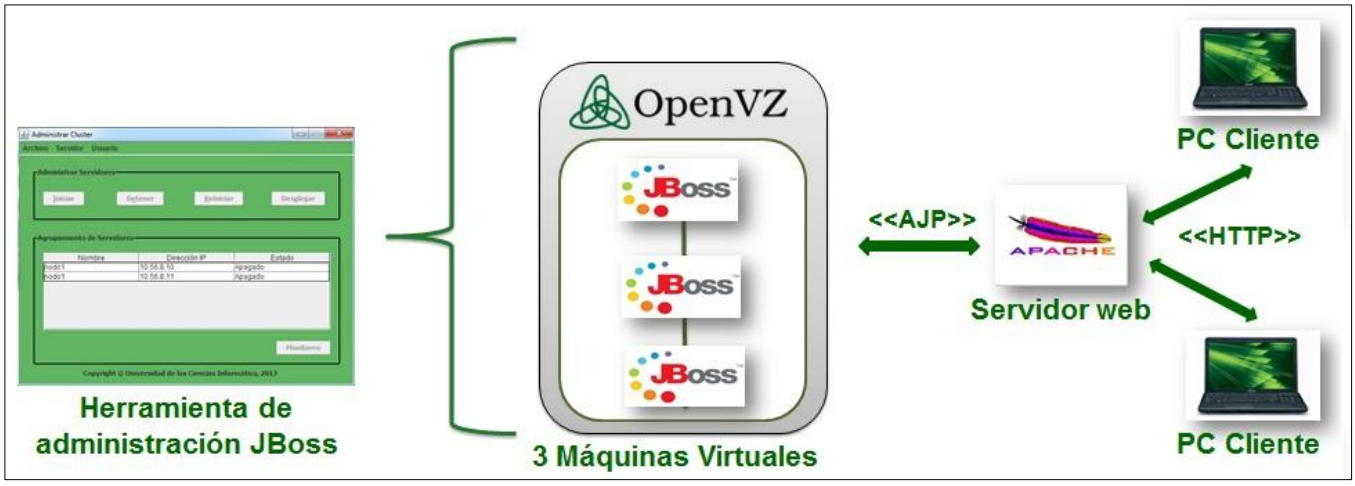


Figura 2.1: Esquema de integración del cluster de JBoss AS y el servidor web Apache.
Fuente: Elaboración propia.

Para la configuración del *cluster* de JBoss AS existen escenarios establecidos para aplicarse en algún contexto según sus características. En las siguientes secciones se explican detalladamente los elementos presentes durante la conformación de este esquema para un mejor entendimiento.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

2.2 Escenarios para el cluster

Existen varios escenarios para la conformación de un *cluster* con JBoss AS. Cada uno describe los elementos a tener presente para la configuración del *cluster* según el escenario escogido. Se puede configurar el *cluster* con la utilización del JBoss AS en servidores físicos distintos, así como en un mismo servidor físico con más de una tarjeta de red o con una sola tarjeta de red.

Para cada instancia de JBoss AS que se inicia, ya fuese en *cluster* o no, es necesario indicarle al servidor de aplicaciones en qué dirección debe escuchar el *socket* para el tráfico de datagramas IP, por defecto el JBoss AS escucha en la dirección 127.0.0.1. En caso que se inicie el JBoss AS en *cluster* no es recomendable utilizar esta dirección, puesto que la comunicación entre los mismos debe ser especificada con un nombre único así como una dirección IP de la misma subred a la que pertenecen los JBoss AS. Los escenarios son explicados con los nodos o JBoss AS utilizados en el *cluster* configurado en este trabajo.

Nodo 1: 10.56.5.201

Nodo 2: 10.56.5.202

Nodo 3: 10.56.5.203

Por defecto, el servidor establece como nombre del *cluster* "*DefaultParttion*", y puede ser utilizado cuando se posee un solo *cluster*. Este nombre es manejado por el "*MBean HAPartition*" para identificar el conjunto de nodos destinados a unirse dentro del mismo *cluster* y definido por el parámetro "-g". El nombre debe ser corto, puesto que éste se incluye en cada mensaje enviado a todo el *cluster*. Por tal motivo se define como nombre de *cluster* "*CAPSPartition*". La sintaxis de ejecución es la siguiente.

run.sh -c <nombre_instancia> -g <nombre del cluster>

El comando utilizado para la ejecución de los JBoss AS en el *cluster* definido se muestra a continuación:

run.sh -c all -g CAPSPartition

La elección del escenario a implementar será basada en las características del sistema y de la entidad responsable. A continuación se describen cada uno de los escenarios anteriormente mencionados.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

Escenario 1: Nodos en máquinas separadas

Este es el escenario más utilizado por desarrolladores para áreas de producción de software con el objetivo de mejorar la escalabilidad y disponibilidad de los servidores, debido a que los nodos están separados en máquinas diferentes. Al tener dos o más nodos, garantiza la disponibilidad de los servicios que estos prestan en caso de que alguno de ellos se detenga. A continuación se muestran los pasos a seguir para la configuración de dicho escenario.

Paso 1: Ejecutar las siguientes líneas de comando, en el nodo 1:

cd /opt/jboss_nodo1/bin

run.sh -c nodo1 -g CAPSPartition –u 239.255.100.100 -b 10.56.5.201

Paso 2: Ejecutar las siguientes líneas de comando, en el nodo 2:

cd /opt/jboss_nodo2/bin

run.sh -c nodo2 -g CAPSPartition –u 239.255.100.100 -b 10.56.5.202

Paso 3: Ejecutar las siguientes líneas de comando, en el nodo 3:

cd /opt/jboss_nodo3/bin

run.sh -c nodo3 -g CAPSPartition –u 239.255.100.100 -b 10.56.5.203

Las opciones mostradas anteriormente se detallan a continuación:

-c con esta opción se especifica que configuración debe ejecutar el servidor de aplicaciones (all/default/minimal/nodo1/nodo2).

-g con este parámetro se establece el nombre del *cluster*.

-u con esta opción se especifica la dirección multicast.

-b especifica la dirección de red que utilizará el nodo.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

Escenario 2: Dos nodos en un solo servidor con más de una tarjeta de red

En este escenario se ejecutan múltiples nodos en la misma máquina y se utilizan diferentes tarjetas de red (multitarjeta) para cada instancia de JBoss AS. Este contexto es muy común en ambiente de desarrollo y usado en la producción combinado con el escenario 1. Al efectuar estas instancias en el servidor en áreas de producción, no resulta muy recomendado debido a que llega a ser un punto de falla en el *hardware*. Cada instancia de JBoss AS está enlazada a una dirección de red diferente para evitar un conflicto de puertos cuando los nodos soliciten abrir los *sockets*. Las líneas de comando a ejecutar son las mismas que en las del escenario 1.

Escenario 3: Dos nodos en un solo servidor

Este escenario es similar al segundo, con la diferencia de que el servidor solo tiene una dirección IP disponible, es decir, solo tiene una tarjeta de red (sin multitarjeta). Dos procesos no pueden enlazar los *sockets* a la misma dirección y puerto, por lo cual se debe especificar que el JBoss AS use diferentes puertos para las dos instancias. Todo esto es posible por el servicio "*ServiceBindingManager*", que es el encargado de administrar toda la información referente a los puertos que son abiertos y cerrados por el JBoss AS, este servicio viene activado por defecto en los JBoss AS 5 y desactivado en la versión JBoss AS 4.x. Para activar este servicio en la versión 4.x, hay que deshabilitar las configuraciones del archivo "*jboss-service.xml*" localizado en el directorio "*conf*", así como mover los puertos en los parámetros de inicio del JBoss AS; a continuación se muestran los pasos a realizar en cada instancia.

Efectuar las siguientes líneas de comando para la primera instancia:

cd /opt/jboss_nodo1/bin

run.sh -c nodo1 -g CAPSPartition -b 10.56.5.201 -Djboss.service.binding.set=ports-defaults

Para la segunda instancia:

cd /opt/jboss_nodo2/bin

run.sh -c nodo2 -g CAPSPartition -b 10.56.5.202 -Djboss.service.binding.set=ports-01

Para la tercera instancia:

cd /opt/jboss_nodo3/bin

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

run.sh -c nodo3 -g CAPSPartition -b 10.56.5.203 -Djboss.service.binding.set=ports-01

La propiedad del sistema "*jboss.service.binding.set*" le indica al "*ServiceBindingManager*" que conjunto de puertos debe utilizar. Por lo tanto el nodo1 utiliza el conjunto de puertos predeterminados del JBoss AS, al contrario el nodo2 utiliza "ports-01", esto le permite a cada puerto un desplazamiento de 100 números desde el número del puerto definido en el nodo1.

Por lo general esta configuración no se recomienda para uso en producción, debido a que incrementa la complejidad de la administración que se produce con el uso de diferentes puertos, además una futura actualización seguramente cambiaría la configuración asignada en cada instancia.

Para la realización del presente trabajo se escoge el escenario 1. Este escenario evita el acceso directo al JBoss AS por estar distribuido en diferentes máquinas, además permite a la aplicación que se encuentra desplegada brindar un mejor servicio debido a la continuidad de los servidores, independientemente de si ocurre algún tipo de fallo en el sistema.

2.3 Cluster entre JBoss AS

Para comenzar con las configuraciones del *cluster* entre los JBoss AS utilizados en esta investigación, se edita el fichero "*cluster-service.xml*", ubicado en el directorio "*deploy*" de la carpeta "*all*". En este fichero se encuentran todas las configuraciones principales del servicio del *cluster*. Luego para lograr un enlace entre diferentes JBoss AS y enviar información entre ellos es necesario establecer un nombre para el *cluster*, una dirección IP y un protocolo de transporte como el Protocolo de Datagrama de Usuario (UDP – *User Datagram Protocol*) y TCP.

Aunque el JBoss AS tiene por defecto configurado el protocolo de transporte UDP en el servicio de *clustering* para el envío de datagramas IP a través de la red, este no tiene confirmación ni control de la entrega o recepción de los paquetes. Sin embargo, el protocolo TCP establece una conexión más confiable en cuanto al envió de datos, debido a que entrega los datos a un receptor específico y a la vez envía un mensaje de recepción al emisor, si los datos recibidos son corruptos, el protocolo TCP permite que los destinatarios soliciten al emisor que vuelvan a enviar los datos.

Por lo planteado en el párrafo anterior se escoge en este trabajo el protocolo TCP como transporte de paquetes en los JBoss AS. Durante la especificación de dicho protocolo se debe modificar el fichero "*cluster-service.xml*", habilitar la configuración TCP y deshabilitar la UDP como se muestra en la figura 2.2.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

```
<Config>
    <TCP bind_addr="10.56.5.201" start_port="7800" loopback="true"
         tcp_nodelay="true"
         recv_buf_size="20000000"
         send_buf_size="640000"
         discard_incompatible_packets="true"
         enable_bundling="false"
         max_bundle_size="64000"
         max_bundle_timeout="30"
         use_incoming_packet_handler="true"
         use_outgoing_packet_handler="false"
         down_thread="false" up_thread="false"
         use_send_queues="false"
         sock_conn_timeout="300"
         skip_suspected_members="true"/>
    <TCPPING initial_hosts="10.56.5.201[7800],10.56.5.202[7800],10.56.5.203[7800]" port_range="3"
             timeout="3000"
             down_thread="false" up_thread="false"
             num_initial_members="3"/>
```

Figura 2.2: Fichero de servicio de *cluster* (cluster-service.xml).

Fuente: Elaboración propia.

A continuación se muestran los atributos fundamentales a configurar en la sección “<Config> TCP”:

- ✓ **bind_addr:** Se especifica la dirección IP de la PC donde se encuentra el JBoss AS.
- ✓ **Initial_hosts:** Se especifican las direcciones IP de todas la PC que contemplan JBoss AS en *cluster*.

Conector HTTP

Debido a que el servidor web Apache se encarga de balancear las cargas de peticiones entre los JBoss AS, se deshabilita el contenedor HTTP con el puerto 8080 de los diferentes nodos incluidos en el *cluster*. Dicho contenedor se localiza en el fichero “*server.xml*” disponible en el directorio “*jboss-web.deployer*” de “*deploy*”. Esta configuración permite las conexiones directas al Apache. A su vez se redirecciona la petición al JBoss AS con menor transacciones. La figura 2.3 muestra la configuración descrita en ambos nodos o JBoss AS.

```
<Connector port="8080" address="${jboss.bind.address}"
           maxThreads="250" maxHttpHeaderSize="8192"
           emptySessionPath="true" protocol="HTTP/1.1"
           enableLookups="false" redirectPort="8443" acceptCount="100"
           connectionTimeout="20000" disableUploadTimeout="true" />
```

Figura 2.3: Contenedor HTTP del fichero Tomcat de JBoss (server.xml).

Fuente: Elaboración propia.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

Como siguiente paso se habilita en los diferentes nodos el Protocolo JServer de Apache (AJP - *Apache JServ Protocol*) (68). Dicho protocolo se encarga de una constante comunicación entre el servidor web y los servidores de aplicaciones, mediante el envío de solicitudes entre los servidores de aplicaciones para examinar el estado de los mismos. La figura 2.4 ilustra la configuración realizada en ambos nodos.

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" address="${jboss.bind.address}" protocol="AJP/1.3"
    emptySessionPath="true" enableLookups="false" redirectPort="8443" />
```

Figura 2.4: Conector AJP del fichero Tomcat de JBoss (server.xml).

Fuente: Elaboración propia.

2.4 Configuración de la aplicación web

Cuando se realiza una aplicación web con la tecnología Java EE, se crea el descriptor de despliegue "*web.xml*", ubicado en el directorio "WEB-INF" de la carpeta donde se almacena todo el contenido web (WebContent). El fichero "*web.xml*" proporciona la configuración y el despliegue de la información para los componentes web que conforman la aplicación. A dicho archivo se le agrega la etiqueta "<*distributable*/>" para indicar a la aplicación web que trabaje en un ambiente de *cluster*. La figura 2.5 muestra la especificación en el descriptor de despliegue del sistema alas SIAPS.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <distributable/>
</web-app>
```

Figura 2.5: Fichero descriptor de despliegue de la aplicación web (server.xml).

Fuente: Elaboración propia.

Al realizar esta especificación, el JBoss AS puede traer la sesión web al contexto del *cluster*, elemento fundamental para mantener las sesiones de los usuarios activas y seguras.

Para completar la configuración de la aplicación web es necesario editar el fichero "*jboss-web.xml*". Este fichero permite agregar información específica de JBoss AS que no se debe modificar en "*web.xml*". La figura 2.6 refleja las utilidades más comunes de dicho fichero.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

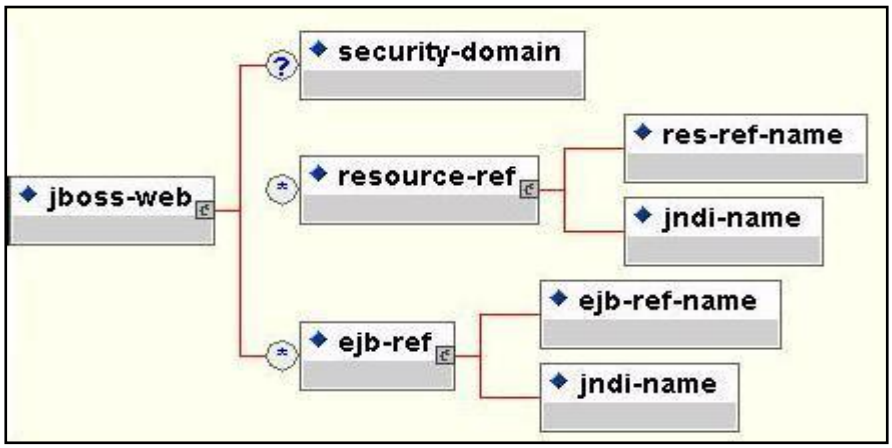


Figura 2.6: Usos comunes del fichero jboss-web.xml.

Fuente: Elaborado por Scott Stark (69).

A continuación se muestra en la figura 2.7 las configuraciones realizadas en el fichero "*jboss-web.xml*" del sistema alas SIAPS, es destacada en un cuadro rojo las transformaciones principales.

```
<!DOCTYPE jboss-web PUBLIC
    "-//JBoss//DTD Web Application 4.2//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-web_4_2.dtd">

<jboss-web>
    <class-loading java2ClassLoadingCompliance="false">
        <loader-repository>
            seam.jboss.org:loader=si
            <loader-repository-config>java2ParentDelegation=false</loader-repository-config>
        </loader-repository>
    </class-loading>
    <replication-config>
      <replication-trigger>SET_AND_NON_PRIMITIVE_GET</replication-trigger>
      <replication-granularity>SESSION</replication-granularity>
      <replication-field-batch-mode>true</replication-field-batch-mode>
    </replication-config>
</jboss-web>
```

Figura 2.7: Replicación de sesión en la aplicación web (jboss-web.xml).

Fuente: Elaboración propia.

Luego de reflejar las configuraciones del fichero "*jboss-web.xml"* se muestra el significado de cada elemento del mismo, para comprender mejor lo expuesto en la figura 2.6. El elemento "*<replication-trigger>*" es el encargado de determinar cuándo es necesario replicar los datos de sesión dentro del *cluster*. Este elemento se compone por las opciones; *SET, SET_AND_GET, ACCESS* y *SET_AND_NON_PRIMITIVE_GET*. La última opción realiza la replicación de la sesión de todos los componentes solo una vez, si el atributo recibido no forma parte de los tipos de datos primitivos (*string*, *integer*, *long*, entre otros).

Otro elemento es "<*replication-granularity*>", que controla el tamaño de las unidades que se van a replicar. Posee tres opciones a configurar; *ATTRIBUTE*, *FIELD* y *SESSION*. La configuración *SESSION* permite replicar el objeto entero de la sesión, en caso de que sea modificado cualquier atributo. Además la sesión completa es serializada en un solo nodo, y luego hace referencia a objetos compartidos en los demás nodos del *cluster*. Por último el elemento "<*replication-field-batch-mode*>" indica si los mensajes de replicación asociados con la solicitud se procesan por lotes en un solo mensaje, el valor predeterminado es verdadero (*true*).

Finalmente, es necesario serializar (70) todas las clases que se requiera replicar dentro del *cluster* cuando esté desplegada la aplicación web. La serialización es un proceso donde el estado de un objeto y sus metadatos (como el nombre de la clase del objeto y los nombres de sus atributos) se almacenan en un formato binario, lo que permite ser enviado por la red a otra máquina. En la figura 2.8 se muestra la serialización de la clase *Authenticator* del sistema alas SIAPS.

```
@Name("authenticator")
public class Authenticator implements Serializable
{
    /**
     *
     */
    private static final long serialVersionUID = 1L;
```

Figura 2.8: Serialización de la clase authenticator del sistema alas SIAPS.

Fuente: Elaboración propia.

2.5 Instalación y configuración del servidor web Apache como balanceador de carga

El servidor web Apache configurado como balanceador de carga de peticiones utiliza la propiedad "*sticky-session*" para mantener los usuarios autenticados en el mismo JBoss AS accedido. Esta propiedad permite que todas las sesiones de los usuarios sean almacenadas en el servidor al cual el Apache envió a los usuarios. Una vez que una sesión es creada en un nodo, cada petición futura también será procesada por el mismo nodo.

Configuración del servidor web Apache

Como paso inicial se habilita el módulo jk en el fichero "httpd.conf" que se encuentra dentro del sub-directorio "conf" y este a su vez se localiza en la dirección de instalación del servidor web Apache. Dentro de este fichero se incluyen todos los módulos a utilizar por el servidor, es por ello que se debe

habilitar la configuración del módulo jk para su funcionamiento posteriormente. A continuación se muestran los pasos seguidos durante las especificaciones pertinentes en el fichero:

Paso 1: Agregar la siguiente sentencia al final del fichero para que el Apache lea la configuración del módulo jk:

Include conf/mod-jk.conf

Paso 2: Seguidamente se crea el fichero "*mod-jk.conf*", en el sub-directorio "*conf*". En este archivo se localizan las configuraciones para el correcto funcionamiento del módulo dentro del Apache. El fichero configurado para este trabajo se muestra en la figura 2.9.

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info

# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
#JkMount /tesis/* loadbalancer
#JkMount / jkmanager /* jkstatus

# You can use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties


# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
#JkShmFile run/jk.shm

# Add jkstatus for managing runtime data

<Location /jkstatus/>
    JkMount status
    Order deny,allow
    Deny from all
    Allow from 10.56.5.14
</Location>
```

Figura 2.9: Fichero para configurar mod_jk (mod-jk.conf).

Fuente: Elaboración propia.

La directiva "*LoadModule*" debe hacer referencia a la librería donde se encuentra el módulo jk. En el directorio "*modules*" se encuentran todos los módulos utilizados por el servidor web Apache, es por ello que se debe especificar la ruta en donde radica este módulo. La directiva "*JkMountFile*" permite leer el fichero "*uriworkermap.properties*" localizado en el directorio "*conf*". En el fichero "*uriworkermap.properties*" se especifican las aplicaciones que se deseen balancear; este parámetro permite adicionarle varias aplicaciones al Apache.

Las especificaciones del archivo "*uriworkermap.properties*" utilizan el formato: "*/url=loadbalancer*" o "*/url/*=loadbalancer*", donde la "url" es el nombre de la aplicación y "/*" sugiere que todo el contenido de la aplicación sea balanceado. Por último la directiva "*loadbalancer*" define el nombre del balanceador incluido en el fichero "*mod-jk.conf*". A continuación se muestra la configuración del archivo "*uriworkermap.properties*" utilizado en la presente investigación.

```
# Simple worker configuration file
# Mount the Servlet context to the ajp13 worker
/alasSIAPS=loadbalancer
/alasSIAPS/*=loadbalancer
```

Figura 2.10: Aplicaciones a balancear por el servidor web Apache (uriworkermap.properties).

Fuente: Elaboración propia.

La directiva "*JkWorkersFile*" especifica el fichero "*workers.properties*", que se ubica en la carpeta "*conf*", en mismo se definen todos los nodos del JBoss AS que estarán enlazados al servidor web Apache. Es incluido una vez en el archivo de configuración "*mod-jk.conf*" debido a que se detallan las direcciones donde se encuentran los JBoss AS. La figura 2.11 muestra el contenido del fichero "*workers.properties*" utilizado.

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

```
# for the mod_jk version 1.2.18 and later
worker.list=loadbalancer,status

# Definimos el nodo Nodo1
# Puerto del conector ajp de nuestro tomcat (JBoss)
worker.nodo1.port=8009
worker.nodo1.host=10.56.5.201
worker.nodo1.type=ajp13
# Peso de nuestro nodo. A más peso, más peticiones recibe.
worker.nodo1.lbfactor=1
worker.nodo1.cachesize=10
worker.nodo1.ping_mode=A
#esto es en caso de fallo del nodo 2
worker.nodo1.redirect=nodo2
#worker.nodo1.socket_timeout=20
#worker.nodo1.recycle_timeout=40
#worker.nodo1.cachesize=100

# Definimos el nodo Nodo2
worker.nodo2.port=8009
worker.nodo2.host=10.56.5.202
worker.nodo2.type=ajp13
worker.nodo2.lbfactor=1
worker.nodo2.cachesize=10
worker.nodo2.ping_mode=A
#esto es en caso de fallo del nodo 1
worker.nodo2.redirect=nodo3
#worker.nodo2.socket_timeout=20
#worker.nodo2.recycle_timeout=40
#worker.nodo2.cachesize=100
# Definimos el nodo Nodo3
worker.nodo2.port=8009
worker.nodo2.host=10.56.5.203
worker.nodo2.type=ajp13
worker.nodo2.lbfactor=1
worker.nodo2.cachesize=10
worker.nodo2.ping_mode=A
#esto es en caso de fallo del nodo 1
worker.nodo2.redirect=nodo1
#worker.nodo2.socket_timeout=20
#worker.nodo2.recycle_timeout=40
#worker.nodo2.cachesize=100

# Comportamiento del loadbalancer
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=nodo1,nodo2,nodo3
worker.loadbalancer.sticky_session=1
```

Figura 2.11: Archivo de configuración de los nodos (workers.properties).

Fuente: Elaboración propia.

En cada nodo del *cluster* se debe especificar el nombre de la instancia del JBoss AS para que puedan esperar el reenvío de peticiones desde el balanceador de carga. Los nodos del *cluster* tienen el mismo nombre definido en el fichero "*workers.properties*" de modo que el servidor web Apache pueda asociar el nombre y la dirección IP de cada uno reflejado en cada instancia del JBoss AS. Para definir el nombre de la instancia de un nodo se edita el fichero "*server.xml*" que se localiza en el directorio "*jboss-web.deployer*" dentro de la carpeta "*deploy*".

CAPÍTULO 2: CONFIGURACIÓN DEL CLUSTER ENTRE SERVIDORES DE APLICACIONES JBOSS.

Dentro del archivo "*server.xml*" se localiza el elemento "<*Engine*>" para agregar el atributo "*jvmRoute*". Esta condición permite añadirle el mismo nombre que se definió en el fichero "*workers.properties*" para cada nodo. A continuación se describen los nombres definidos en la propiedad "*jvmRoute*" del archivo "*server.xml*" en cada uno de los nodos utilizados en el presente trabajo.

Nodo 1:

<Engine name="jboss.web" defaultHost="localhost**" jvmRoute="nodo1**"/>

Nodo 2:

<Engine name="jboss.web" defaultHost="localhost" **jvmRoute="nodo2**"/>

Nodo 3:

<Engine name="jboss.web" defaultHost="localhost" **jvmRoute="nodo3**"/>

2.6 Configuración de la conexión segura

El protocolo HTTPS utiliza la Capa de Conexión Segura (SSL - *Secure Sockets Layer*) para realizar encriptaciones para proporcionar comunicación fiable en la red. Las aplicaciones que utilizan el protocolo SSL saben cómo brindar y recibir claves de cifrado con otras aplicaciones, además de cifrar y descifrar los datos enviados entre el receptor y el emisor a través de certificados que contienen las claves. De modo que la información se transmita encriptada para evitar que pueda ser leída por *sniffers* (71) (72) u otros recursos.

Para el servidor web Apache se habilito la configuración SSL debido a que constituye el punto de entrada al *cluster* y por ende a la aplicación desplegada. En el anexo 1 se describen los pasos seguidos para la configuración del mismo.

2.7 Virtualización mediante OpenVZ

Para la virtualización utilizada en este trabajo se emplearon tres nodos o contenedores con la distribución GNU/Linux Debian. Los pasos seguidos durante la configuración de la virtualización se aprecian en los anexos 2, 3, 4 y 5. En cada uno de ellos se encuentra una instancia del JBoss AS que a su vez incluyen el despliegue de sistema alas SIAPS. Cada nodo posee las siguientes prestaciones:

- ✓ **Memoria RAM:** 1 GB

- ✓ **Almacenamiento:** 10 GB
- ✓ **Memoria de intercambio:** 2 GB

Se programaron tres procedimientos (*scripts*) para la automatización de las configuraciones de la red de cada nodo, así como el enlace (*bridge*) con la máquina virtual. Es optimizado el Contexto de Ejecución de Java (*JRE – Java Runtime Enviroment*) para que utilice la memoria de intercambio. Los procedimientos programados se aprecian en los anexos 6, 7 y 8.

Consideraciones del capítulo

El escenario para implantar la técnica de *cluster* en los servidores JBoss AS escogido demuestra ser aceptable para el despliegue del sistema alas SIAPS.

Las configuraciones realizadas correspondientes a las conexiones seguras en el servidor Apache favorecen los niveles de confiabilidad e integridad de los datos que se gestionan en la aplicación alas SIAPS.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS.

Se detallan los conceptos fundamentales que sirven de guía para confeccionar el modelo de dominio. Este modelo permite especificar las condiciones, capacidades y cualidades que la herramienta debe tener, además la definición de las actividades que constituyen objeto de automatización. También se plantean los requerimientos funcionales y no funcionales de la herramienta de administración. Todos estos aspectos quedan reflejados en los distintos artefactos identificados en cada una de las fases de la guía que rige el ciclo de vida de desarrollo.

3.1 Caracterización del entorno.

El inicio del JBoss AS para el despliegue del sistema alas SIAPS puede ser mediante sentencias de comandos Linux (terminal) de manera local o remota a través del protocolo de Intérprete de Ordenes Segura (SSH - Secure SHell). El administrador de sistemas habilita el historial de comandos en su ordenador personal para facilitar la labor de despliegue y actualización. La detección de un fallo en el JBoss AS y consecuentemente la no disponibilidad del sistema puede durar horas; para evitar estos fallos el administrador chequea el JBoss AS mediante soluciones como la ejecución pings, diagnóstico físico, entre otros.

3.2 Herramienta propuesta.

La herramienta propuesta es una aplicación de escritorio portable basada en software libre y multiplataforma, por lo que se puede utilizar en cualquier sistema operativo que utilice la máquina virtual de java (JVM-Java Virtual Machine) (73) para ejecutar aplicaciones realizadas en el lenguaje de programación java.

La herramienta informática registra todos los JBoss AS que se encuentren en el *cluster* definido en el capítulo 2, lo que le permite ejecutar las funciones principales de cada uno de ellos (iniciar, detener, reiniciar) y de realizar el despliegue del sistema alas SIAPS. La autenticación de los usuarios es a través del dominio UCI. La base de datos es embebida y se almacenan los usuarios que tienen privilegios de modificar determinadas opciones que son fundamentales en la herramienta. A continuación se describen algunas interfaces de usuario de la herramienta desarrollada.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

La interfaz de autenticación es la encargada de permitir el acceso a los usuarios registrados en la base de datos (Local) y los del dominio (uci.cu). Todos los usuarios registrados en la base de datos local tendrán privilegios administrativos, en consecuencia con el objetivo de la herramienta. La figura 3.1 refleja la pantalla antes descrita.

Figura 3.1: Interfaz autenticación.

Fuente: Elaboración propia.

Por otro lado se encuentra la pantalla principal de la herramienta Administrar Cluster. La misma posee un listado de todos los nodos que conforman el *cluster* que dispone de las funcionalidades Iniciar, Detener y Reiniciar, encargadas de cambiar el estado de los JBoss AS. El despliegue de una aplicación en cualquiera de los nodos es otra de las funcionalidades importantes que se puede realizar desde esta interfaz. A continuación se muestra la pantalla correspondiente a la descripción anterior.

Figura 3.2: Interfaz administrar *cluster*.

Fuente: Elaboración propia.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

Para realizar un despliegue de una aplicación en uno de los nodos del *cluster* se utiliza la figura 3.3. En la misma se verifica la búsqueda del archivo “alassiaps.ear”, el cual contiene todo el sistema a desplegar.

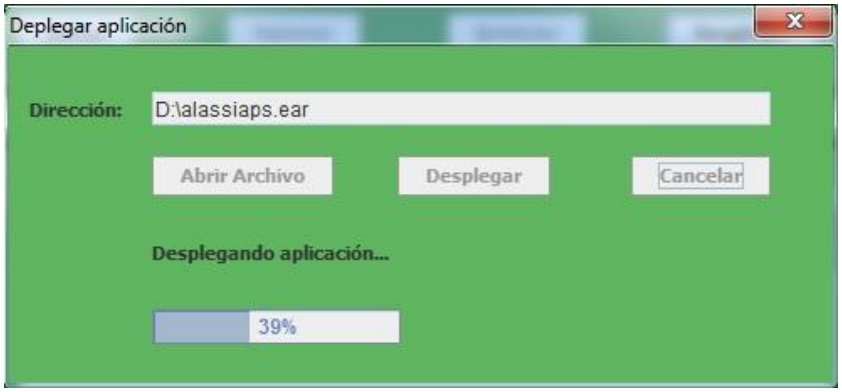


Figura 3.3: Interfaz para desplegar aplicación web.
Fuente: Elaboración propia.

A continuación se describen en las siguientes secciones los elementos presentes durante la etapa de análisis y diseño de la herramienta desarrollada.

3.2.1 Especificación de requerimiento.

Según Pressman (74) los requisitos constituyen funcionalidades o condiciones presentes en un producto para que este sea válido, es decir, debe regirse por un contrato que exprese las necesidades del cliente a cumplir para que el sistema a implementar tenga éxito. Los requisitos de software se clasifican en varias categorías como los requisitos funcionales y los requisitos no funcionales.

En los requisitos funcionales se describen todas las condiciones que el sistema debe cumplir, dígase entradas, salidas de datos y excepciones. Mientras que los requisitos no funcionales son las propiedades o cualidades que el producto debe contener, para ser usable, rápido, atractivo y confiable.

Requisitos funcionales

RF1. Iniciar Servidor

Este requisito es el encargado de iniciar un JBoss AS en la máquina donde se encuentre el mismo.

RF2. Detener Servidor.

Detiene el nodo seleccionado por el administrador del sistema.

RF3. Reiniciar Servidor.

Permite reiniciar el JBoss AS que fue seleccionado por el administrador del sistema.

RF4. Desplegar aplicación en un servidor.

Con esta funcionalidad se puede desplegar una aplicación web en cualquier nodo del *cluster*, el cual se encargaría de desplegar dicha aplicación en los demás nodos.

RF5. Buscar Servidor.

Esta funcionalidad permite buscar un nodo determinado dentro de la base de datos.

RF6. Autenticación.

Es el encargado de validar el acceso a la aplicación por dos tipos de dominio (Local y uci.cu).

RF7. Agregar Servidor.

Permite adicionar un nuevo JBoss AS a la herramienta para su posterior control.

RF8. Eliminar Servidor.

Ofrece la posibilidad de eliminar un JBoss AS de la base de datos del sistema.

RF9. Editar Servidor.

Editar Servidor posibilita modificar los datos reflejados en un nodo del *cluster.*

RF10. Adicionar Usuario.

Esta funcionalidad permite adicionar un nuevo usuario a la base de datos de dicho sistema.

Requisitos no funcionales

Usabilidad.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

RnF1. El sistema estará diseñado de manera que los usuarios adquieran las habilidades necesarias para explotarlo en un tiempo reducido:

Usuarios normales: 15 días

Usuarios avanzados: 30 días

RnF2. Cumplir con las pautas de diseño de las interfaces.

Fiabilidad

RnF3. En los servidores de los policlínicos y en el Centro de Datos Nacional se garantizará una arquitectura de máxima disponibilidad, tanto de servidores de aplicación como de base de datos.

Portabilidad.

RnF4. La herramienta está construida para operar en sistemas operativos que tenga la JVM instalada, está realizada con un lenguaje libre y sin necesidad de instalación.

Confiabilidad.

RnF5. Se garantizará la consistencia de los datos, al realizar comprobaciones y validaciones cada vez que sea necesario.

Seguridad.

RnF6. Se mantendrá seguridad y control a nivel de usuario, lo que garantiza el acceso de los mismos a los niveles establecidos de acuerdo al dominio que utilicen para su autenticación.

Hardware.

RnF7. Mínimo para la herramienta: PC con procesador Intel Pentium IV 2.8 GHz, 1 GB de RAM, 3 GB de disco duro.

RnF8. Mínimo para el *cluster*: PC con procesador Intel Dual-Core 2.8 GHz, 2 GB de RAM, 30 GB de disco duro.

RnF9. La comunicación de las terminales clientes con el servidor será a través de conexiones a una velocidad constante de 10/100 Mbps.

Software.

RnF10. Es obligatorio tener instalado la JVM para el correcto funcionamiento de la aplicación.

3.2.2 Modelo de dominio

En el modelo de dominio se detallan los términos que utilizan los usuarios y desarrolladores para poder comprender el sistema a utilizar, al igual que se describen los conceptos que deberá trabajar la herramienta. El principal objetivo del modelo de dominio es comprender y describir las clases más importantes dentro del contexto del sistema, es decir, contribuir a una mejor comprensión del problema que el sistema resuelve en relación a su contenido. En la figura 3.4 se muestra el diagrama del modelo de domino de la herramienta informática para administrar el *cluster* de JBoss AS.

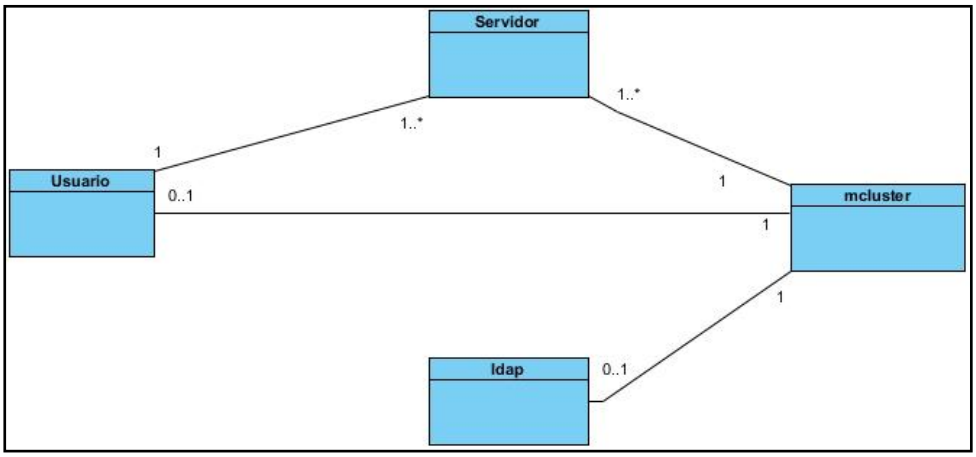


Figura 3.4: Modelo del dominio.
Fuente: Elaboración propia.

Cada uno de los conceptos o términos del diagrama anterior se describen a continuación:

- ✓ Usuario: El concepto hace referencia a los usuarios que tienen acceso a la herramienta informática.
- ✓ Servidor: El concepto hace referencia a los servidores que están asociados al *cluster* creado.
- ✓ *Cluster*: El concepto hace referencia al *cluster* creado por los servidores asociados y los usuarios registrados.

- ✓ Ldap: El concepto hace referencia a la propiedad de obtener todos los usuarios del dominio UCI.

3.2.3 Arquitectura del sistema

Según Kruchten (75) la arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema.

Existen algunos estilos arquitectónicos como el Modelo Vista Controlador (MVC) (76), encargado de separar los datos y la lógica de negocio de la interfaz gráfica de la aplicación. Este patrón se basa en las ideas de reutilización de código y la separación de conceptos, lo que facilita el trabajo de los desarrolladores.

La arquitectura Cliente – Servidor (77) es otro estilo utilizado comúnmente para distribuir el trabajo en una aplicación. Este modelo comparte las tareas entre los proveedores de servicios (servidores) y los demandantes (clientes). Además, ofrece un fácil mantenimiento puesto que distribuye las funciones y responsabilidades entre varios ordenadores independientes, brinda la posibilidad de realizar cambios en los servidores sin impedir la dinámica de trabajo de los clientes.

Por otro lado existe la arquitectura en capas (78), la misma posibilita la distribución del sistema en diferentes capas. Dichas capas agrupan el contenido de las aplicaciones, lo que posibilita utilizar tantas capas como se necesiten para el desarrollo de un sistema. La arquitectura en tres capas es la más utilizada dentro de esta gama, la misma está compuesta por la capa diseño, lógica de negocio y de acceso a datos.

En la presente investigación se escoge el estilo en capas debido a que se ajusta mejor a las características de la herramienta al distribuir en capas los componentes utilizados en el desarrollo de la misma. Las capas utilizadas se separan según su responsabilidad: la capa de lógica de negocio de la presentación y la de datos.

La capa de presentación, está compuesta por las interfaces y las imágenes de la herramienta desarrollada. La capa de negocio comprende las clases del negocio:

- ✓ *Ldap,* para la comunicación con del dominio.
- ✓ *Servidor*, para la gestión de los datos relacionados con los JBoss AS.
- ✓ *Usuario,* para la gestión de los datos relacionados con los usuarios.
- ✓ *General,* para la gestión de los datos globales como validaciones y algoritmos comunes.

La capa de acceso a datos encargada de realizar las conexiones de la capa de negocio a la base de datos. La capa de datos se compone de la base de datos Apache Derby. En la figura 3.5 se ilustra las diferentes capas así como la interrelación entre ellas.

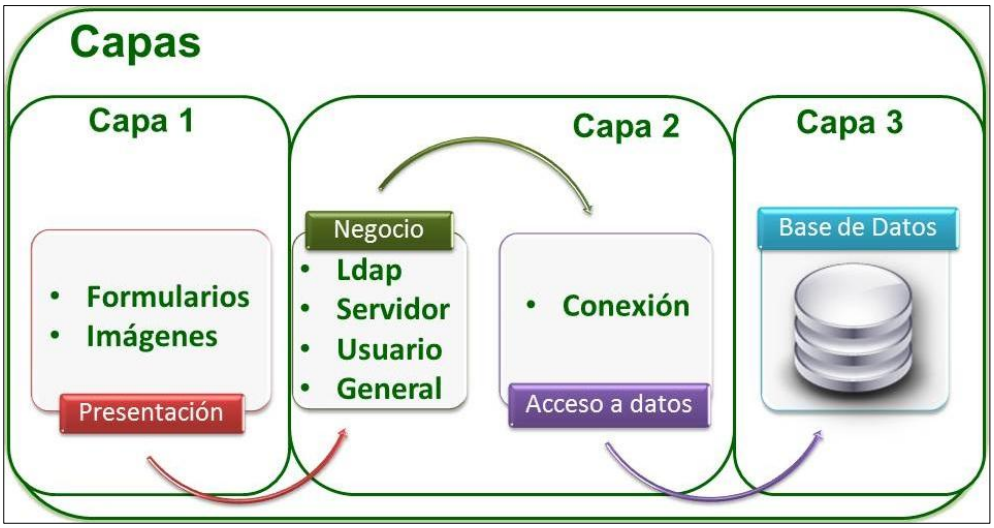


Figura 3.5: Arquitectura en capas de la herramienta desarrollada.
Fuente: Elaboración propia.

3.2.4 Modelo de datos

En el modelo de datos se ilustra toda la estructura de la base de datos utilizada en el producto informático desarrollado. También dentro del mismo se detallan las restricciones necesarias para que los datos reflejen la realidad deseada y las principales operaciones sobre los mismos. En la figura 3.6 se muestra el modelo de datos definido para la herramienta informática deseada. Consta de dos tablas relacionadas para el almacenamiento de los datos, usuarios y las configuraciones del *cluster*.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

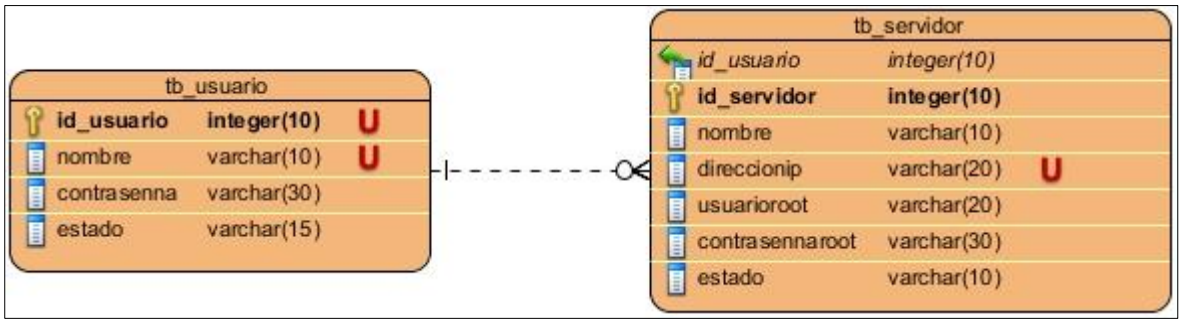


Figura 3.6: Modelo de datos de la herramienta propuesta.

Fuente: Elaboración propia.

A continuación se describe las estructuras de las tablas utilizadas en el modelo de datos correspondientes a la herramienta desarrollada.

Descripción de las tablas de la base de datos

Esquema de administración del *cluster*.

Tabla 1: Esquemas de tablas

Esquemas	
Nombre	**Descripción**
tb_usuario	Contiene los usuarios administradores que tienen acceso a la herramienta.
tb_servidor	Contiene todos los servidores que conforman el *cluster*.

A continuación se presenta el esquema de la tabla tb_usuario. La misma muestra los atributos que contiene, además de su descripción y tipo de dato.

Tabla 2: Tabla Usuario

Estuctrura tb_usuario				
Nombre	**Tipo de Dato**	**PK/FK**	**Nullable**	**Descripción**
id_usuario	Integer	Llave primaria (PK)	Not null	Identificador del usuario
nombre	varchar(10)		Not null	Nombre del usuario
contrasenna	varchar(30)		Not null	Contraseña del usuario administrador
estado	varchar(15)		Not null	Estado del usuario administrador, si está conectado o desconectado

También se muestra el esquema de la tabla tb_servidor. En la misma se puede visualizar los atributos que contiene y sus características principales.

Tabla 3: Tabla Servidor

Estructura tb_servidor				
Nombre	**Tipo de Dato**	**PK/FK**	**Nullable**	**Descripción**
id_usuario	Integer	Llave foránea (FK)	Not null	Identificador del usuario asociado a los servidores
id_servidor	Integer	Llave primaria (PK)	Not null	Identificador del servidor en la tabla
nombre	varchar(10)		Not null	Nombre del servidor en la

				tabla, es único
direccionip	varchar(20)		Not null	Dirección IP de las PC donde se encuentran los servidores.
estado	varchar(10)		Not null	Indica el estado de los servidores (Encendido o Apagado)
usuario	varchar(15)		Not null	Administrador de la PC donde se encuentra el servidor
contrasenna	varchar(30)		Not null	Contraseña del super-usuario de la PC donde se encuentra el servidor

3.2.5 Estándar de codificación

Los estándares de codificación constituyen un conjunto de normas que guían a los desarrolladores para escribir un código fuente comprensible de acuerdo lo establecido por el proyecto. En la presente investigación se utilizó la convención de java como estándar de codificación (79). Algunos elementos del estándar se describen a continuación:

Comentarios

Los programas escritos en el lenguaje java pueden tener dos tipos de comentarios, comentarios de implementación y comentarios de documentación. Los comentarios de implementación se delimitan por ***/*...*/***, y ***//***, además se utilizan para describir todo el código. Los comentarios de documentación presentan una visión de más alto nivel para personas y desarrolladores que no tienen el código y desean reutilizarlo.

Comentarios de una Línea

Pueden aparecer comentarios cortos de una línea al nivel del código que siguen. Si un comentario no se puede escribir en una línea, debe seguir el formato de los comentarios de bloque. Un comentario de una sola línea debe ir precedido de una línea en blanco. La figura 3.7 muestra un fragmento de código en el que se evidencia un comentario de línea utilizado en la herramienta.

```
/**
 * @retorna la conexion de la base de datos
 */
public Conexion getConexionDb() throws ClassNotFoundException,
                                       InstantiationException,
                                       IllegalAccessException{

    if (cdb == null) {
        String nombreClase=System.getProperty("databaseclass");

        cdb=(Conexion)Class.forName(nombreClase).newInstance();
    }

    return cdb;
}
```

Figura 3.7: Comentarios de una línea en la clase de acceso a datos.

Fuente: Elaboración propia.

Comentarios fin de Línea

El delimitador de comentario ***//*** puede convertir en comentario una línea completa o una parte de una línea. No debe ser usado para hacer comentarios de varias líneas consecutivas; sin embargo, puede usarse en líneas consecutivas para comentar secciones de código, como se muestra en la figura 3.8.

```
public boolean Comprobardireccionip(String dir){
    //inicializando lista servidor y la variable boolean var
    List<servidorClase> listaServ=con.obtenerDatosServidores();
    boolean var=false;

    //haciendo una busqueda
    for (int i = 0; i < listaServ.size(); i++) {
        //si la dirección entrada por parámetro
        //es igual a alguna dirección de la base de datos
        if (listaServ.get(i).getDireccionIP().equals(dir)) {
            var=true;
            //detener busqueda cuando la variable var sea verdadera
            break;
        }
    }
    return var;
}
```

Figura 3.8: Comentarios de fin de línea.

Fuente: Elaboración propia.

Declaraciones y cantidad de líneas

Se utiliza una declaración por línea, ya que facilita los comentarios. En la figura 3.9 se observa un fragmento de esta declaración correspondiente a la clase “servidorClase”.

```
private String Nombre;
private String DireccionIP;
private String Estado;
private String Usuario;
private String Contrasenna;
```

Figura 3.9: Declaración y cantidad de líneas de la clase “servidorClase”.

Fuente: Elaboración propia.

Líneas y espacios en blanco

Las líneas en blanco mejoran la facilidad de lectura porque separa secciones de código que están lógicamente relacionadas. Se deben utilizar siempre dos líneas en blanco en las siguientes circunstancias:

- ✓ Entre las secciones de un fichero fuente.
- ✓ Entre las definiciones de clases e interfaces.

Se debe usar siempre una línea en blanco en las siguientes circunstancias:

- ✓ Entre métodos.
- ✓ Entre las variables locales de un método y su primera sentencia.
- ✓ Antes de un comentario de bloque o de un comentario de una línea.
- ✓ Entre las distintas secciones lógicas de un método para facilitar la lectura.

Se deben usar espacios en blanco en las siguientes circunstancias:

- ✓ Una palabra clave del lenguaje seguida por un paréntesis debe separarse por un espacio.

Variables, constantes, clases y métodos

- ✓ Todas las instancias y variables de clases o métodos empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúscula. Los

nombres de variables no deben empezar con los caracteres: "_" o "$", aunque ambos están permitidos por el lenguaje.

- ✓ Los nombres de las variables deben ser cortos pero con significado. La elección del nombre de una variable debe ser un mnemónico, designado para indicar a un observador casual su función. Los nombres de variables de un solo caracter se deben evitar, excepto para variables índices temporales.

- ✓ Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas separadas por el sub-guión "_". Las constantes ANSI se deben evitar, para facilitar su depuración.

- ✓ Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas.

- ✓ Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.

3.2.6 Esquema de despliegue

El despliegue de la herramienta es sobre los ordenadores de los administradores de sistema con funcionamiento local, aunque también puede ser desplegada mediante la tecnología Java Web Start (*JavaWS*) ***(80)*** incorporada en el entorno de ejecución de java (JRE) ***(81)***. La opción *JavaWS* permite descargar y ejecutar aplicaciones desde un navegador web siguiendo el formato establecido. Al ejecutar la herramienta, se comprueba previamente la existencia de alguna actualización antes de poner en marcha la aplicación. La principal potencialidad de este esquema es la reducción de los costos de implantación y mantenimiento, pues estas tareas se realizan de forma centralizada. La figura 3.10 representa el diagrama de despliegue de la aplicación informática en el servidor web Apache.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

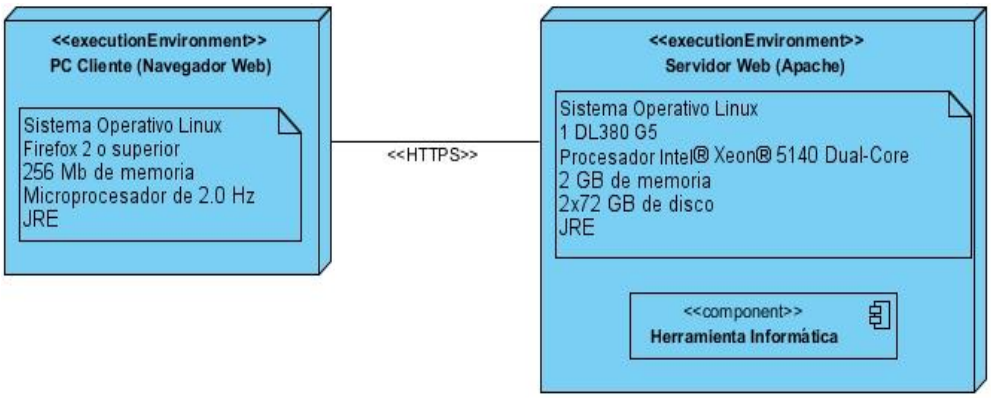


Figura 3.10: Diagrama de despliegue de la herramienta propuesta.

Fuente: Elaboración propia

3.2.7 Seguridad

La seguridad en un sistema de información es de vital importancia debido a que protege todos los datos manejados y almacenados por el personal encargado de efectuar las tareas de gestión de dicha información. De manera que se garantiza una mayor integridad, confidencialidad y disponibilidad de la información en aras de prevenir la pérdida de información. Al enlazar todas estas características se crea un punto de acceso, que impide que se modifiquen valores en los que el personal no está autorizado.

La herramienta incluye un control de acceso basado en una política de usuarios y contraseñas, lo que posibilita que cada usuario acceda solo a las funcionalidades establecidas de acuerdo a la función que realizan. El acceso a las funcionalidades es centralizado mediante el Protocolo Ligero de Acceso a Directorios (LDAP - *Lightweight Directory Acces Protocol*) de modo que la autenticación es única. También brinda la posibilidad de una autenticación local para los casos en que el directorio no esté disponible.

Al autenticarse el usuario, primero se comprueba su validez en el dominio y luego de manera local. Solo los usuarios que aparezcan en la base de datos local podrán gestionar un *cluster* de JBoss AS para así evitar que otros usuarios registrados en el directorio sean administradores de la herramienta y por ende del *cluster*.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

3.2.8 Internacionalización

El proceso de internacionalización (82) (generalmente acortada en i18n) consiste en confeccionar un sistema de manera tal que se ajuste a diferentes idiomas y regiones sin la necesidad de modificar el código fuente. Esto permite que cada elemento (texto, formato de fecha, colores e imágenes) puedan depender del lenguaje y la cultura del usuario final.

La internacionalización se realiza mediante archivos de propiedades (*properties*), en estos se representan los datos en la forma clave/valor. La clave es el identificador utilizado por el programa para recuperar el texto, y el valor es el texto real. Se crea un archivo de propiedades para cada idioma en el que se traduce el programa. Las claves son las mismas en cada lugar, los únicos que cambian son los valores para cada idioma. Fueron internacionalizados todos los componentes visuales y los mensajes tanto de error como de información. En la figura 3.11 se representa el archivo "*properties*" donde se almacenan los lenguajes español e inglés en caso de que el sistema operativo utilice alguno de estos idiomas.

Bundle.properties

Key	default language	en_US - inglés (Estados Unidos)
adminCluster.jMenuItem4.text	Eliminar	Delete
adminCluster.jButton3.text	Reiniciar	Restart
adminCluster.jButton1.text	Detener	Stop
adminCluster.jPanel2.border.title	Agrupamiento de Servidores	Cluster Server
adminCluster.title	Administrar Cluster	Cluster Admin
adminCluster.jPanel1.border.title	Administrar Servidores	Server Admin
adminCluster.jButton2.text	Iniciar	Start
adminCluster.jMenuItem7.text	Eliminar	Delete
adminCluster.jMenuItem10.text	Contraseña	Password
adminCluster.jMenuItem9.text	Nombre	Name
adminCluster.jMenu5.text	Editar	Edit
adminCluster.jMenu4.text	Usuario	User
adminCluster.jMenuItem5.text	Agregar	Add
adminCluster.jMenu1.text	Archivo	File
adminCluster.jMenu2.text	Servidor	Server
adminCluster.jMenuItem1.text	Salir	Exit
adminCluster.jMenu6.text	Editar	Edit
adminCluster.jMenuItem2.text	Agregar	Add
adminCluster.jMenuItem11.text	Usuario Root	Root User
adminCluster.jMenuItem6.text	Servidor	Server
adicionarUsuario.jLabel2.text	Contraseña:	Password:
adicionarUsuario.jLabel1.text	Nombre:	Firs Name:
adicionarUsuario.title	Adicionar Usuario	Add User
adicionarUsuario.jButton2.text	Cancelar	Cancel
adicionarUsuario.jButton1.text	Adicionar	Add

Figura 3.11: Internacionalización de los interfaces visuales.

Fuente: Elaboración propia.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA PARA LA ADMINISTRACIÓN DE CLUSTER JBOSS

Consideraciones del capítulo

Mediante el trabajo realizado en este capítulo se pudo abordar las siguientes conclusiones:

El estilo de arquitectura escogido para la herramienta de administración de *cluster* JBoss posibilita la estructura lógica en capas compuestas por los distintos archivos incluidos en el espacio de trabajo (*workspaces*).

La realización del diseño a través del modelo de dominio facilita el desarrollo de la herramienta propuesta debido a sus características y contexto en que se utiliza.

La seguridad de la herramienta resultó ser configurable y adaptable a otros ambientes, debido a la utilización de una autenticación centralizada basada en dominio o local para contextos que lo requieran.

CAPÍTULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

CAPITULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

En la confección de un producto informático es imprescindible realizar pruebas de calidad en todo el ciclo de vida. En el presente capítulo se describen las pruebas y validaciones realizadas a la herramienta de administración, así como al *cluster* configurado.

La validación está conformada por el despliegue del sistema alas SIAPS en el *cluster* para comprobar el rendimiento y la disponibilidad del mismo. Se efectúan pruebas de estrés para chequear el comportamiento de algunas funcionalidades del módulo configuración perteneciente al sistema alas SIAPS. Por último, se aplican pruebas de caja negra a través de diseños de casos de prueba para validar la aplicación informática desarrollada.

4.1 Despliegue del módulo de configuración

Se realizó un despliegue del sistema alas SIAPS en el *cluster* configurado para comprobar el correcto funcionamiento del mismo basado en indicadores como disponibilidad y estrés. A continuación se describen cada uno de los indicadores mencionados, así como el análisis comparativo arrojado.

4.1.1 Disponibilidad

La disponibilidad en un sistema informático permite otorgar un continuo funcionamiento de los servicios que se brindan. El *cluster* posee esta posibilidad, de modo que, si un nodo es detenido por alguna razón ajeno a su funcionamiento otros estarán disponibles para que el servicio no se pierda.

En la figura 4.1 se aprecia la unión de tres nodos en *cluster.* En dicha imagen el nodo uno con dirección IP 10.56.5.201 detecta la existencia de un nuevo miembro, de este modo se conforma el *cluster* que tiene como nombre "*DefaultPartition*".

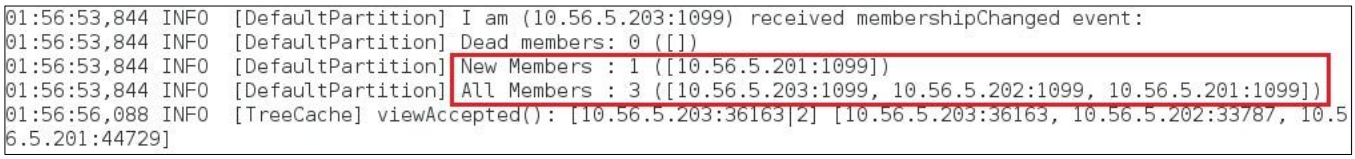


Figura 4.1: Unión de dos nodos en cluster.

Fuente: Elaboración propia.

El *cluster* habilita el despliegue del sistema alas SIAPS con el fin de brindar un mejor servicio al sistema. La figura 4.2 proporciona la vista del módulo configuración del sistema desplegado.

CAPÍTULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

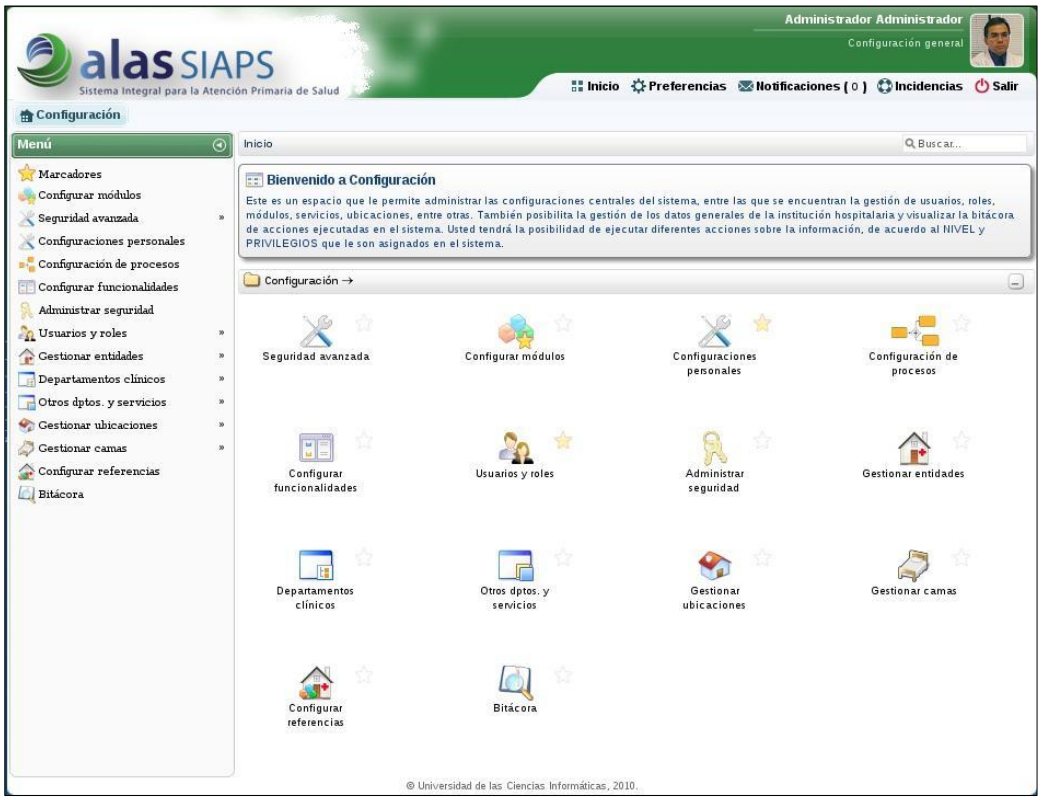


Figura 4.2: Interfaz de configuración. Sistema alas SIAPS.

Fuente: Elaboración propia.

Al ocurrir un error en el nodo dos, ya sea por mal funcionamiento o rotura, el mismo es desconectado del *cluster*. Los nodos restantes asumen la responsabilidad de mantener el sistema alas SIAPS operativo. En la figura 4.3 se aprecia la separación de uno de los nodos que conforman el *cluster*.

```
02:05:10,203 INFO  [DefaultPartition] I am (10.56.5.201:1099) received membershipChanged event:
02:05:10,203 INFO  [DefaultPartition] Dead members: 1 ([10.56.5.202:1099])
02:05:10,203 INFO  [DefaultPartition] New Members : 0 ([])
02:05:10,203 INFO  [DefaultPartition] All Members : 2 ([10.56.5.203:1099, 10.56.5.201:1099])
```

Figura 4.3: Salida de un nodo.

Fuente: Elaboración propia.

4.2 Pruebas de rendimiento

Las pruebas de rendimiento o de estrés son las pruebas que se realizan desde una perspectiva para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Existen dos tipos de pruebas de rendimiento: pruebas de carga y de estrés.

CAPÍTULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

Las pruebas de carga se ejecutan para comprender el comportamiento de una aplicación ante una carga determinada. La carga puede estar compuesta por el número de usuarios esperado en producción o un número de transacciones durante un tiempo determinado. El resultado de esta prueba nos ofrece el tiempo de respuesta de todas las transacciones críticas.

Las pruebas de estrés son utilizadas normalmente para someter a la aplicación al límite de su funcionamiento mediante la ejecución de un número de usuarios muy superior al esperado, o bien median la substracción de recursos (también conocidas como pruebas negativas donde se simula por ejemplo el fallo de un servidor en *cluster*). Este "test de stress" tiene como finalidad determinar la robustez de una aplicación cuando la carga es extrema, y ayuda a los administradores a comprobar la configuración de las alarmas del sistema entre otras cosas. En este tipo de pruebas los tiempos de respuesta de la aplicación no son importantes y tienden a ser ignorados. Otro posible objetivo de este tipo de pruebas es determinar el límite real de la aplicación en cuanto a número de usuarios concurrentes, número de transacciones por segundo, entre otros.

Para asegurar y garantizar el correcto funcionamiento del módulo de configuración, se realizaron pruebas para determinar la capacidad de procesamiento, estabilidad y rapidez del mismo con el apoyo del programa Jmeter. Las pruebas se aplicaron a las funcionalidades “Agregar Usuario” y “Modificar Usuario” pertenecientes al módulo Configuración del sistema alas SIAPS en dos escenarios.

- ✓ **Escenario 1**: Pruebas aplicadas al sistema con un JBoss AS.
- ✓ **Escenario 2**: Pruebas aplicadas al sistema en *cluster* JBoss configurado.

La división en estos escenarios se realiza con el objetivo de establecer análisis comparativos así como distinguir claramente los diferentes parámetros medidos. Se muestra a continuación los análisis hechos por cada escenario. El flujo simulado por la herramienta Jmeter es el siguiente:

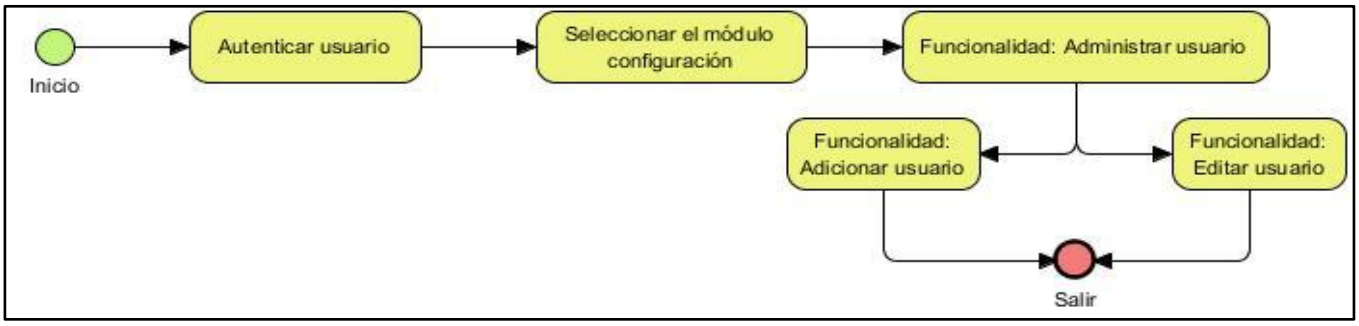


Figura 4.4: Flujo de acceso simulado por la herramienta Jmeter.

Fuente: Elaboración propia.

CAPÍTULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

Escenario 1:

La figura 4.5 muestra las pruebas realizadas perteneciente a la funcionalidad Agregar Usuario.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/gehos	30	2613	2119	4236	2112	4239	0,00%	6,7/sec	12,7
/gehos/	30	2119	2120	2130	2107	2132	0,00%	7,1/sec	11,5
/gehos/mod...	90	2998	2750	5509	329	5997	0,00%	2,4/sec	11,4
/gehos/mod...	30	4646	4545	4990	4393	5032	0,00%	3,9/sec	19,6
/gehos/mod...	60	3493	2560	4894	2186	5197	0,00%	5,8/sec	16,6
/gehos/mod...	60	3770	3605	5989	261	6808	0,00%	4,9/sec	13,9
/gehos/mod...	60	1813	829	5072	160	6173	0,00%	7,4/sec	21,2
/gehos/mod...	30	2600	2574	3438	1328	4089	0,00%	3,8/sec	18,7
/gehos/inde...	30	1950	1688	2852	1026	3111	0,00%	3,9/sec	19,5
Total	420	2934	2536	5034	160	6808	0,00%	9,9/sec	35,3

Figura 4.5: Pruebas de estrés realizado a la funcionalidad “Agregar Usuario” con un JBoss AS.

Fuente: Elaboración propia.

En la imagen anterior se visualiza la prueba efectuada para simular 30 usuarios conectados. En la siguiente tabla se detallan los campos más importantes de la figura anterior. Además se efectúan pruebas con diferentes cantidades de solicitudes para realizar una comparación en cuanto al rendimiento, tiempo de respuesta y tasa de error.

Tabla 4.1: Resumen de las pruebas aplicada a la funcionalidad “Agregar Usuario”.

Actividad: Agregar Usuario				
No. de solicitudes	**Tiempo de respuesta (milisegundos)**		**% Error**	**Rendimiento**
	Mínimo	**Máximo**		
10	120	3945	0	4,8/segundos
30	160	6808	0	9,9**/**segundos

Como se puede observar se simularon 2 experimentos con 10 y 30 usuarios respectivamente. En el primer experimento se obtuvo un procesamiento de 4,8 peticiones por segundo, mientras que el segundo experimento arrojó 9,9 peticiones por segundo. Se comprueba entonces que entre mayor sea la cantidad de peticiones, más tiempo demora el sistema en brindar una respuesta a los usuarios.

Para la funcionalidad “Modificar Usuario” se aplicó el mismo procedimiento. A continuación se muestra en la figura 4.6 el informe que brindado por la herramienta Jmeter.

CAPÍTULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/gehos	30	139	45	157	13	2234	0,00%	13,4/sec	25,4
/gehos/modC...	90	2417	2588	3332	912	3957	0,00%	3,5/sec	16,7
/gehos/modC...	30	2396	2377	2566	2059	2752	0,00%	5,1/sec	25,2
/gehos/modC...	60	1338	543	2473	162	2739	0,00%	8,6/sec	24,6
/gehos/modC...	60	1554	574	2838	265	3002	0,00%	8,0/sec	22,7
/gehos/modC...	90	1185	429	2902	104	3222	0,00%	6,2/sec	13,2
/gehos/modC...	30	2630	2665	2870	2073	3044	0,00%	4,6/sec	23,0
/gehos/index....	30	2107	2007	2506	1629	3297	0,00%	4,2/sec	21,1
Total	420	1704	2151	2917	13	3957	0,00%	16,0/sec	56,2

Figura 4.6: Pruebas de estrés realizado a la funcionalidad “Modificar Usuario” en un JBoss AS.

Fuente: Elaboración propia.

Posteriormente se visualiza la tabla 4.2 con los datos más significativos. En dicha tabla se refleja el comportamiento de la funcionalidad “Modificar Usuario” con diferentes volúmenes de usuarios conectados concurrentemente.

Tabla 4.2: Resumen de las pruebas aplicada a la funcionalidad “Modificar Usuario”

Actividad: Modificar Usuario				
No. de solicitudes	**Tiempo de respuesta (milisegundos)**		**% Error**	**Rendimiento**
	Mínimo	**Máximo**		
10	15	2139	0	8,3/segundos
30	13	3957	0	16,0**/**segundos

La tabla 4.2 refleja para 10 y 30 usuarios un procesamiento de 8,3 y 16,0 peticiones por segundos respectivamente.

Escenario 2:

Luego de haber realizado las pruebas con un JBoss AS, se procede a efectuar un chequeo del rendimiento, tiempo de respuesta y tasa de error del *cluster* configurado. La tabla 4.3 refleja los parámetros antes mencionados.

CAPÍTULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/gehos	30	53	26	193	13	237	0,00%	25,9/sec	48,9
/gehos/mod...	90	1391	1099	3329	102	4168	0,00%	5,6/sec	27,1
/gehos/mod...	30	1839	642	3623	250	3928	0,00%	3,0/sec	15,1
/gehos/mod...	60	994	369	3390	82	3921	0,00%	5,0/sec	14,1
/gehos/mod...	60	773	354	2488	86	3807	0,00%	5,2/sec	14,9
/gehos/mod...	60	706	303	2429	92	3952	0,00%	5,3/sec	15,1
/gehos/mod...	30	1539	1301	2951	199	3788	0,00%	2,8/sec	14,1
/gehos/inde...	30	1003	465	2368	172	3468	0,00%	3,0/sec	14,7
Total	390	1043	406	3205	13	4168	0,00%	24,4/sec	90,7

Figura 4.7: Pruebas de estrés realizado a la funcionalidad “Adicionar Usuario” con el cluster.

Fuente: Elaboración propia.

A continuación se presenta la tabla 4.3 con un resumen de los datos mostrados en la figura anterior.

Tabla 4.3: Prueba de estrés para el *cluster*.

Actividad: Agregar Usuario				
No. de solicitudes	**Tiempo de respuesta (milisegundos)**		**% Error**	**Rendimiento**
	Mínimo	**Máximo**		
10	14	935	0	22.5/segundos
30	13	4168	0	24,4/segundos

La tabla 4.3 refleja para 10 y 30 usuarios un procesamiento de 22,5 y 24,4 peticiones por segundos respectivamente.

La siguiente figura representa las pruebas realizas con 30 muestras de la funcionalidad “Modificar Usuario”, realizadas con el *cluster*.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/gehos	30	85	28	277	12	335	0,00%	25,6/sec	48,5
/gehos/mod...	90	1283	1123	2980	120	3637	0,00%	5,4/sec	26,0
/gehos/mod...	30	1897	2580	3075	203	3495	0,00%	3,6/sec	18,1
/gehos/mod...	60	1124	424	3413	83	3868	0,00%	5,8/sec	16,6
/gehos/mod...	60	1042	455	3029	86	4035	0,00%	5,4/sec	15,4
/gehos/mod...	90	654	343	1642	77	3725	0,00%	7,1/sec	15,1
/gehos/mod...	30	1207	896	2008	203	3695	0,00%	3,3/sec	16,5
/gehos/inde...	30	1286	964	2881	208	3959	0,00%	3,3/sec	16,7
Total	420	1044	509	2965	12	4035	0,00%	25,0/sec	87,7

Figura 4.8: Pruebas de estrés realizado a la funcionalidad “Modificar Usuario” con el cluster.

Fuente: Elaboración propia.

La tabla 4.4 representa el resumen de las pruebas realizadas a la funcionalidad “Modificar Usuario”.

Tabla 4.4: Prueba de estrés para el *cluster*.

Actividad: Modificar Usuario				
No. de solicitudes	**Tiempo de respuesta (milisegundos)**		**% Error**	**Rendimiento**
	Mínimo	**Máximo**		
10	14	939	0	23.5/segundos
30	12	4035	0	25,0/segundos

La tabla 4.4 refleja para 10 y 30 usuarios un procesamiento de 23,5 y 25,0 peticiones por segundos respectivamente.

Las pruebas realizadas al *cluster* fueron en general mejores que las arrojadas por las de un JBoss AS. Los tiempos de respuestas son ligeramente más pequeños para cada muestra escogida, además, mejora la cantidad de peticiones por segundo llegando a un máxima de 25. Por lo tanto queda demostrada la factibilidad del mismo.

Es importante señalar que el rendimiento del sistema está estrechamente vinculado con el *hardware* del balanceador de carga, base de datos y los nodos del *cluster*, ya que estos determinan el desempeño del mismo. En la presente investigación se utilizó para realizar las pruebas de estrés dos máquinas para los JBoss AS y otra para el servidor web Apache. A continuación se muestran las especificaciones de dichas máquinas.

Nodo 1, Nodo 2 y Nodo 3 (JBoss AS): procesador Intel(R) i3 a 3.10 GHz con 2 GB de RAM.

Balanceador de carga (Apache): procesador Pentium(R) Dual-Core(R) a 2.60 GHz con 1 GB de RAM.

4.3 Pruebas de caja negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, y es completamente indiferente al comportamiento interno y la estructura del programa. Estas pruebas se realizan con el objetivo de detectar errores tales como: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y terminación. Algunas técnicas se mencionan a continuación:

✓ *Partición de equivalencia*: La técnica de partición de equivalencia divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de pruebas. El diseño de estos

casos de pruebas para esta partición se basa en la evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada y regularmente estas condiciones pueden ser un valor numérico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

✓ *Análisis de valores límites*: Los casos de pruebas que exploran las condiciones límites producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. La técnica del análisis de valores límites complementa a la de partición equivalente, pues en lugar de centrarse solamente en las condiciones de entrada, deriva los casos de pruebas también para el campo de salida.

✓ *Grafos de causa-efecto*: En este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. En este método:

- ✓ Se crea un grafo de objetos importantes y sus relaciones.
- ✓ Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

Los diseños de casos de prueba se diseñan según la plantilla elaborada por la entidad CaliSof radicada en la UCI. Estos documentos forman parte del expediente de proyecto universitario y se encuentra disponible en el sitio web oficial de la entidad. En la siguiente sección se muestran los diseños de casos de pruebas elaborados.

4.3.1 Diseños de casos de pruebas

Tabla 4.5: Diseño de casos de prueba “Adicionar Usuario”

Escenario	Descripción	Desde	Hasta	Respuesta del sistema	Flujo central
EC 1.1 Adicionar usuario con campos correctos.	*Para adicionar un usuario se comienza llenando los campos con datos correctos.*	*V*	*V*	*El sistema muestra un mensaje "Usuario adicionado correctamente"*	*El usuario selecciona en el menú Usuario la opción Agregar. El sistema muestra una interfaz para introducir los datos del usuario administrador que se quiere crear: nombre y contraseña; así como las opciones de Agregar y Cancelar. El usuario entra los datos que pide la interfaz y presiona el botón Agregar. El sistema agrega el nuevo usuario con todos los datos y muestra un mensaje que indica que el usuario administrador se añadió correctamente. El usuario puede añadir otro administrador o puede salir de la interfaz mediante el botón cancelar.*

EC 1.2 Adicionar usuario con los campos incorrectos	*Se presiona el botón agregar*	*V*	*V*	*El sistema muestra un mensaje "Por favor, llene todos los campos"*	*El usuario selecciona en el menú Usuario la opción Agregar. El sistema muestra una interfaz para introducir los datos del usuario administrador que se quiere crear: nombre y contraseña; así como las opciones de Agregar y Cancelar. El usuario presiona el botón Agregar. El sistema muestra un mensaje de error que indica que los campos tienen que estar llenos. El usuario puede intentar adicionar otro administrador o puede salir de la interfaz mediante el botón cancelar.*
EC 1.3 Adicionar usuario con un usuario existente	*Se llenan todos los campos de la interfaz*	*V*	*V*	*El sistema muestra un mensaje "El usuario ya existe"*	*El usuario selecciona en el menú Usuario la opción Agregar. El sistema muestra una interfaz para introducir los datos del usuario administrador que se quiere crear: nombre y contraseña; así como las opciones de Agregar y Cancelar. El usuario introduce un nombre que ya existe y una contraseña, luego presiona el botón Agregar. El sistema muestra un mensaje indicando que el usuario ya existe. El usuario puede intentarlo añadir otro usuario administrador o puede salir de la interfaz mediante el botón cancelar.*
EC 1.3 Cancelar	*Se escoge la opción Cancelar*	*NA*	*NA*	*El sistema regresa a la interfaz anterior*	*El usuario escoge la opción Cancelar. El sistema regresa a la interfaz anterior.*

Tabla 4.6: Diseño de casos de prueba “Autenticación Usuario”

Escenario	Descripción	Desde	Hasta	Respuesta del sistema	Flujo central
EC 1.1 Autenticación con todo los campos correctos	*El usuario debe seleccionar un dominio para su autenticación e introduce nombre de usuario y contraseña*	*V*	*V*	*El sistema permite el acceso a la herramienta*	*El usuario introduce nombre y contraseña de acceso en la interfaz de autenticación, además de escoger el dominio por donde desea acceder a la herramienta. Luego el usuario presiona el botón aceptar. El sistema permite el acceso a la herramienta*
EC 1.2 Autenticación con los componentes vacíos	*El usuario debe seleccionar un dominio para su autenticación*	*V*	*V*	*El sistema muestra un mensaje "Usuario y contraseña incorrecto"*	*El usuario presiona el botón aceptar y el sistema deniega el acceso y muestra un mensaje, luego brinda la posibilidad de autenticarse*
EC 1.3 Autenticación con los componentes incorrectos	*El usuario debe seleccionar un dominio para su autenticación*	*V*	*V*	*El sistema muestra un mensaje "Credenciales de acceso incorrectas"*	*El sistema muestra una interfaz de autenticación, el usuario introduce los datos nombre y contraseña, luego presiona el botón aceptar. El sistema muestra un mensaje indicando que no puede acceder a la herramienta por que las credenciales son incorrectas, el sistema permite reintentar el acceso*
EC 1.3 Cancelar	*Se escoge la opción Cancelar*	*NA*	*NA*	*El sistema regresa a la interfaz anterior*	*El usuario escoge la opción Cancelar. El sistema termina su funcionamiento.*

CAPÍTULO 4: VALIDACIÓN DE LA HERRAMIENTA DESARROLLADA

Luego de realizar las prueba por el administrador de la calidad del departamento grupo de calidad interno, a partir de los diseños de casos de pruebas y usando la técnica de partición de equivalencias, se lograron detectar 3 no conformidades de la cuales ninguna eran de impacto alto. Las mismas se relacionan con:

- ✓ Navegabilidad en el redireccionamiento de las interfaces.
- ✓ La funcionalidad de inserción y la adquisición de datos (opción agregar servidor).
- ✓ La notificación de las acciones agregar servidor y desplegar proyecto.

En un tiempo breve, fueron solucionadas las no conformidades detectadas y se conciliaron con los desarrolladores antes de proceder a la solución de las mismas. Cumplido este tiempo se realizaron nuevas pruebas de verificación en materia del cumplimiento de estas no conformidades, obteniendo como resultado la calidad necesaria para ser explotada.

Consideraciones parciales

En este capítulo se realizaron las validaciones al producto desarrollado, las mismas arrojaron las siguientes conclusiones:

Se comprobó la eficiencia del *cluster* con respecto a un solo JBoss AS, arrojando rápidas respuestas para la cantidad de usuarios estimados.

Con los diseños de casos de pruebas se comprobó el buen funcionamiento de la herramienta, puesto que los valores de salida se corresponden con las variables de entrada.

CONCLUSIONES GENERALES

El análisis de los distintos escenarios de aplicación de la técnica *cluster* JBoss AS permitió escoger el de tipo 1, el cual se adecua a las características de despliegue del sistema alas SIAPS.

La virtualización basada en OpenVZ escogida se integra fácilmente con el *cluster* JBoss AS configurado, pues ofrece elementos de calidad sistémica como escalabilidad, rendimiento y facilidad de administración.

El acceso centralizado a la herramienta a través del dominio o directorio favorece el proceso de autenticación en entornos donde el rol de administrador de sistemas puede variar.

Los resultados obtenidos durante la validación conformada demostraron el correcto funcionamiento del *cluster* y de la herramienta de administración.

RECOMENDACIONES

- ✓ Configurar un *cluster* para la base de datos del sistema alas SIAPS que funcione de manera integrada al *cluster* JBoss AS.

REFERENCIAS BIBLIOGRÁFICAS

1. *Ventajas del uso de las TICs en el proceso de enseñanza-aprendizaje desde la óptica de los docentes universitarios españoles.* **Ferro Soto, Carlos A., Martínez Senra, Ana Isabel y Otero Neira, María del Carmen.** 29, s.l. : Edutec: Revista electrónica de tecnología educativa, 2009.

2. **Selley Rojas, Héctor Julián.** *Monitoreo del comportamiento de servidores de aplicaciones.* Ciudad de México : s.n., 2008. A060432.

3. **Miguel Armendáriz, Luis.** Código abierto (Open Source). [En línea] 2006. [Citado el: 21 de Marzo de 2013.]

4. **Miranda en Prezi, Daniela.** *JBoss.* [En línea] [Citado el: 1 de Mayo de 2013.] http://prezi.com/gp4lqgcmtu4s/jboss/.

5. **Rodas Gálvez, Carlos Enrique.** Universidad de San Carlos de Guatemala. *Cluster.* [En línea] 2002. [Citado el: 3 de Febrero de 2013.] http://carlos8rg.files.wordpress.com/2008/08/onto_cluster.pdf.

6. **Correa Baez, Edit Marili y Morejó Abril, Marianela del Pilar.** *Estudio y diseño de cluster Beowulf bajo la plataforma Linux para la elaboración de un Webcluster.* s.l. : Ambato, 2005.

7. Portal de Seguridad Informática. *¿Cómo es el ataque de Denegación de Servicio o DDoS?* [En línea] [Citado el: 14 de Febrero de 2013.] https://seguridad.uci.cu/content/%C2%BFc%C3%B3mo-es-el-ataque-de-denegaci%C3%B3n-de-servicio-o-ddos..

8. *Test de penetración y gestión de vulnerabilidades, estrategias clave para evaluar la seguridad de red.* **AREITIO, J.** 653, s.l. : Española de Electrónica, 2009.

9. OWASP. *The Open Web Application Security Proyect. Cross-site Scripting (XSS).* [En línea] 12 de Agosto de 2011. [Citado el: 6 de Febrero de 2013.] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

10. **Álvarez S., Nicolás y Monsalve Z., Juan.** *Instalación y configuración de un servidor web.* Santa Maria : s.n., 2008.

11. **Universida de las Américas Puebla.** Universida de las Américas Puebla. *Cliente - Servidor.* [En línea] [Citado el: 6 de Febrero de 2013.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf.

12. **Bassi, Roxana.** *Navegación en la World Wide Web.* Argentina : s.n., 2001.

13. *Web Server Survey | Netcraft.* [En línea] [Citado el: 20 de Abril de 2013.] http://news.netcraft.com/archives/category/web-server-survey/.

14. Java. *¿Qué es la tecnología Java y por qué lo necesito?* [En línea] [Citado el: 7 de Febrero de 2013.] http://www.java.com/es/download/faq/whatis_java.xml.

15. Java. *¿Qué es Java Enterprise Edition (Java EE)?* [En línea] [Citado el: 8 de Febrero de 2013.] http://www.java.com/es/download/faq/techinfo.xml.

16. Oracle. *Productos Oracle WebLogic.* [En línea] [Citado el: 24 de Abril de 2013.] http://www.oracle.com/es/products/middleware/appserver/index.html.

17. IBM developerWorks. *Introducción a WebSphere.* [En línea] [Citado el: 24 de Abril de 2013.] http://www.ibm.com/developerworks/ssa/websphere/newto/.

18. Apache TomEE. [En línea] [Citado el: 24 de Abril de 2013.] http://tomee.apache.org/apache-tomee.html.

19. GlassFish Server. [En línea] [Citado el: 24 de Abril de 2013.] http://www.oracle.com/technetwork/middleware/glassfish/overview/index.html.

20. Apache Tomcat - Welcome! [En línea] [Citado el: 24 de Abril de 2013.] http://tomcat.apache.org/.

21. Plumbr. *Most popular application servers.* [En línea] [Citado el: 24 de Abril de 2013.] http://plumbr.eu/blog/most-popular-application-servers.

22. *IBM - España.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.ibm.com/es/es/.

23. *Conmutación de Paquetes.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.uazuay.edu.ec/estudios/sistemas/teleproceso/apuntes_1/conmutacion_paquetes.htm.

24. RAND. *Corporation Provides Objective Research Services and Public Policy Analysis.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.rand.org/.

25. **Nudelman, Gustavo.** Universidad Tecnológica Nacional. [En línea] [Citado el: 13 de Mayo de 2013.] http://www.electron.frba.utn.edu.ar/materias/95-0419/archivos/20100614_Procesamiento_Paralelo.pdf.

26. Internet Society. *Brief History of the Internet - Internet Timeline.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet.

27. Clusters. *Administración del Cluster.* [En línea] [Citado el: 23 de Abril de 2013.] http://clusterfie.epn.edu.ec/clusters/Definiciones/definiciones4.html.

28. **Almeroth, K. y Quinn, B.** fags. *IP Multicast Applications: Challenges and Solutions.* [En línea] 9 de 2001. [Citado el: 13 de Junio de 2013.] http://www.faqs.org/rfcs/rfc3170.html.

29. Ejemplos de Java y C/Linux. *Programación de sockets en C de Unix/Linux.* [En línea] 4 de 2 de 2007. [Citado el: 13 de Junio de 2013.] http://www.chuidiang.com/clinux/sockets/sockets_simp.php#sockets.

30. **Márquez García, Francisco Manuel.** *UNIX, Programación avanzada.* s.l. : RA-MA EDITORIAL, 2004. 978-84-7897-603-4.

31. **Ridruejo, Francisco Javier, Agirre, Jon y Miguel-Alonso, José.** XIV Jornada de Paralelismo. *Estrategias de instalación y gestión de clusters con software libre.* [En línea] Septiembre de 2003. [Citado el: 15 de 6 de 2013.] http://www.sc.ehu.es/acwmialj/papers/jornadas03_a.pdf.

32. **Oracle.** Oracle Technology Network. *JNDI.* [En línea] [Citado el: 14 de 6 de 2013.] http://www.oracle.com/technetwork/java/overview-142035.html.

33. Red Hat Customer Portal. *Buddy Replication.* [En línea] [Citado el: 21 de Mayo de 2013.] https://access.redhat.com/site/documentation/en-US/JBoss_Enterprise_Application_Platform/4.3/html/Cache_Tree_Cache_Guide/Clustered_Cache___Using_Replication-Buddy_Replication.html.

34. **Oracle.** Oracle Technology Network. *JavaBeans FAQ: General Questions.* [En línea] [Citado el: 12 de Junio de 2013.] http://www.oracle.com/technetwork/java/javase/faq-135947.html.

35. **La Rosa Agramonte, Alberto Jesús y Del Castillo Rodríguez, Hiran.** *Herramienta para la instalación y configuración de clústeres del Contenedor de Servlets.* La Habana : s.n., 2009.

36. Nagios. *Nagios XI.* [En línea] [Citado el: 13 de Mayo de 2013.] http://www.nagios.com/products/nagiosxi/.

37. Red Hat. *JBoss Operations Network.* [En línea] [Citado el: 24 de Abril de 2013.] http://www.redhat.com/products/jbossenterprisemiddleware/operations-network/.

38. Red Hat JBoss Enterprise Middleware. *Open source middleware.* [En línea] [Citado el: 1 de Mayo de 2013.] http://www.redhat.com/products/jbossenterprisemiddleware/.

39. MSDN Microsoft. *Visual Studio.* [En línea] [Citado el: 16 de Junio de 2013.] http://msdn.microsoft.com/es-es/library/ms235632%28v=vs.80%29.aspx.

40. MSDN Microsft. *Visual Studio Resources.* [En línea] [Citado el: 16 de Junio de 2013.] http://msdn.microsoft.com/en-us/vstudio/cc136611.aspx.

41. **Rivillo Matía, Norberto J.** *Control mediante IDE Eclipse, de una placa de entradas-salidas multiples conectada al microcontrolador STM32L discovery.* 2012.

42. **María del Carmen, Tellería Prieto.** *SISTEMA DE INFORMACIÓN CLÍNICA.* s.l. : En Informática Salud 2013, 2012. SLD107 BEHIQUE-SIC.

43. Netbeans. [En línea] [Citado el: 11 de Febrero de 2013.] http://netbeans.org/.

44. **Cerda, Felipe.** *NetBeans.* [En línea] [Citado el: 11 de Febrero de 2013.] http://api.ning.com/files/PcRFEwSyAz6w2k4r-KsPqXzHa3EJ6JtOUDOCxjxcpwJJFWH8XaIgnfUFGtj2TURFJc3LgS7W9tCq-NsM68o3ZBHIMDWV12-W/netbeans65es_cl.pdf.

45. **Tituaña Cumbal, Walter Celiano y Torres Cañizares, Edwin Jesús.** *Elaboración de un manual de la plataforma Netbeans Ide para la Disicom.* s.l. : LATACUNGA, 2009.

46. **Oracle.** Document Oracle. *Project Swing.* [En línea] 2010. [Citado el: 14 de Junio de 2013.] http://docs.oracle.com/javase/1.5.0/docs/guide/swing/.

47. Enterprise Architect. *Herramienta de diseño UML y herramienta CASE UML para desarrollo de software.* [En línea] [Citado el: 19 de Mayo de 2013.] http://sparxsystems.com.ar/products/ea.html.

48. Rational Rose Enterprise. *Paquete Rational Rose.* [En línea] [Citado el: 19 de Mayo de 2013.] http://www.rational.com.ar/herramientas/roseenterprise.html.

49. *Guión Visual Paradigm for UML.* [En línea] 2011. [Citado el: 7 de Febrero de 2013.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

50. *Adapting microsoft SQL server for cloud computing.* **Bernstein, Philip A.** 12005313, s.l. : Data Engineering (ICDE), 2011 IEEE 27th International Conference on, 2011. 1063-6382.

51. MySQL. *MySQL Customers.* [En línea] [Citado el: 20 de Mayo de 2013.] http://www.mysql.com/customers/.

52. PostgreSQL. *About postgres.* [En línea] [Citado el: 20 de Mayo de 2013.] http://www.postgresql.org/about/.

53. Apache DB Proyect. *What is Apache Derby?* [En línea] [Citado el: 7 de Febrero de 2013.] http://db.apache.org/derby/.

54. *C# Station.* [En línea] [Citado el: 19 de Mayo de 2013.] http://csharp-station.com/.

55. **Rodríguez del Toro, Tamara.** *Herramienta de autoría de material educativo en lenguaje C#.* [En línea] 2011. [Citado el: 15 de Junio de 2013.]

56. **Tymoschuk, Jorge.** Universidad Tecnológica Nacional. *Paradigmas de Programación.* [En línea] 2009. [Citado el: 12 de Junio de 2013.] http://labsys.frc.utn.edu.ar/ppr-2009/Unidad%20I%20-%20POO%20Avanzada/Apunte/Unidad%20I%20-%20POO%20Avanzada.pdf.

57. **Microsoft.** Microsoft. *Visual Studio Test Professional.* [En línea] 2012. [Citado el: 15 de Junio de 2013.] http://www.microsoft.com/visualstudio/esn/products/visual-studio-test-professional-2012#product-edition-testpro.

58. **Selenium.** Document Selenium. *What is Selenium.* [En línea] [Citado el: 15 de Junio de 2013.] http://docs.seleniumhq.org/.

59. **Apache.** Apache. *Apache Jmeter.* [En línea] [Citado el: 15 de Junio de 2013.] http://jmeter.apache.org/.

60. *Oracle VM VirtualBox.* [En línea] [Citado el: 5 de Junio de 2013.] https://www.virtualbox.org/.

61. Microsoft. *Windows Virtual PC.* [En línea] [Citado el: 5 de Junio de 2013.] http://www.microsoft.com/es-ES/download/details.aspx?id=3702.

62. *OpenVZ Linux Containers Wiki.* [En línea] [Citado el: 5 de Junio de 2013.] http://openvz.org/Main_Page.

63. Apple. *Todo lo que necesitas saber sobre OS X.* [En línea] [Citado el: 5 de Junio de 2013.] http://www.apple.com/es/osx/what-is.

64. **Brito Jiménez, Arístides Lescay y Estrada, Pedro Manuel.** *Componentes web Psicología y Psiquiatría del Sistema Integral para la Atención Primaria de Salud.* La Habana : s.n., 2012.

65. **Jacobson, y otros, y otros.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000. 84-7829-036-2.

66. *Ingeniería de software.* **Sommerville, Ian.** Madrid : Pearson Educación, 2005, Vol. Séptima edición. 84-7829.

67. *Sistema de Información Medioambiental.* **Taboada González, José A. y Cotos Yáñez, José Manuel.** Madrid : Carlos Iglesia, 2005. 84-9754-056-6.

68. The Apache Tomcat Connector. *AJP Protocol Reference - AJPv13.* [En línea] [Citado el: 24 de Abril de 2013.] http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html.

69. Web container configuration. *Use of jboss-web.xml.* [En línea] [Citado el: 15 de Mayo de 2013.] http://www.jboss.ru/translate/jboss24/ch05s22.html.

70. MSDN Microsoft. *Serialización.* [En línea] [Citado el: 15 de Junio de 2013.] http://msdn.microsoft.com/es-es/library/7ay27kt9%28v=vs.80%29.aspx.

71. *Network traffic analysis and intrusion detection using packet sniffer.* **Qadeer, Mohammed Abdul, y otros, y otros.** Singapore : Communication Software and Networks, 2010. ICCSN '10. Second International Conference on , 2010. Vols. En Communication Software and Networks, 2010. 978-1-4244-5726-7.

72. *Assessing the Fidelity of COTS 802.11 Sniffers.* **Serrano, Pablo, Zink, Michael y Kurose, Jim.** Rio de Janeiro : INFOCOM 2009, IEEE, 2009. 978-1-4244-3512-8.

73. *Virtual machine showdown: Stack versus registers.* **SHI, Yunhe.** 4, s.l. : ACM Transactions on Architecture and Code Optimization (TACO), 2008, Vol. 4.

74. **Pressman, Roger S.** *Ingeniería de software un enfoque práctico.* Madrid : Madrid. MacGraw-Hill, 2002.

75. *Planos Arquitectónicos: El Modelo de "4+ 1" Vistas de la Arquitectura del Software.* **Kruchten, Philippe.** 6, s.l. : IEEE Software, 1995, Vol. 12.

76. ComuSOFT. *Modelo Vista Controlador – Definición y Características.* [En línea] [Citado el: 19 de Mayo de 2013.] http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

77. *Arquitectura cliente-servidor.* [En línea] [Citado el: 19 de Mayo de 2013.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

78. **González, Enrique.** *Arquitectura en capas. Un camino hacia los procesos distribuidos.* Chile : s.n., 2006.

79. **García Llinás, Luis Fernandeo.** *Programación orienta a objetos en Java.* Bogotá, Colombia : Universidad del Norte de la U, 2010. 9789587410624.

80. *¿Qué es Java Web Start y cómo se ejecuta?* [En línea] [Citado el: 16 de Mayo de 2013.] http://www.java.com/es/download/faq/java_webstart.xml.

81. **Java.** *Java Runtime Environment.* [En línea] [Citado el: 5 de Junio de 2013.] http://es.download.cnet.com/Java-Runtime-Environment-JRE-64-Bit/3000-2378_4-75317067.html.

82. Internationalizing a GUI. *NetBeans IDE Tutorial.* [En línea] [Citado el: 15 de Mayo de 2013.] https://netbeans.org/kb/docs/java/gui-automatic-i18n.html.

BIBLIOGRAFÍA

1. *Ventajas del uso de las TICs en el proceso de enseñanza-aprendizaje desde la óptica de los docentes universitarios españoles.* **Ferro Soto, Carlos A., Martínez Senra, Ana Isabel y Otero Neira, María del Carmen.** 29, s.l. : Edutec: Revista electrónica de tecnología educativa, 2009.

2. **Selley Rojas, Héctor Julián.** *Monitoreo del comportamiento de servidores de aplicaciones.* Ciudad de México : s.n., 2008. A060432.

3. **Miguel Armendáriz, Luis.** Código abierto (Open Source). [En línea] 2006. [Citado el: 21 de Marzo de 2013.]

4. **Miranda en Prezi, Daniela.** *JBoss.* [En línea] [Citado el: 1 de Mayo de 2013.] http://prezi.com/gp4lqgcmtu4s/jboss/.

5. **Rodas Gálvez, Carlos Enrique.** Universidad de San Carlos de Guatemala. *Cluster.* [En línea] 2002. [Citado el: 3 de Febrero de 2013.] http://carlos8rg.files.wordpress.com/2008/08/onto_cluster.pdf.

6. **Correa Baez, Edit Marili y Morejó Abril, Marianela del Pilar.** *Estudio y diseño de cluster Beowulf bajo la plataforma Linux para la elaboración de un Webcluster.* s.l. : Ambato, 2005.

7. Portal de Seguridad Informática. *¿Cómo es el ataque de Denegación de Servicio o DDoS?* [En línea] [Citado el: 14 de Febrero de 2013.] https://seguridad.uci.cu/content/%C2%BFc%C3%B3mo-es-el-ataque-de-denegaci%C3%B3n-de-servicio-o-ddos..

8. *Test de penetración y gestión de vulnerabilidades, estrategias clave para evaluar la seguridad de red.* **AREITIO, J.** 653, s.l. : Española de Electrónica, 2009.

9. OWASP. *The Open Web Application Security Proyect. Cross-site Scripting (XSS).* [En línea] 12 de Agosto de 2011. [Citado el: 6 de Febrero de 2013.] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

10. **Álvarez S., Nicolás y Monsalve Z., Juan.** *Instalación y configuración de un servidor web.* Santa Maria : s.n., 2008.

11. **Universida de las Américas Puebla.** Universida de las Américas Puebla. *Cliente - Servidor.* [En línea] [Citado el: 6 de Febrero de 2013.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf.

12. **Bassi, Roxana.** *Navegación en la World Wide Web.* Argentina : s.n., 2001.

13. *Web Server Survey | Netcraft.* [En línea] [Citado el: 20 de Abril de 2013.] http://news.netcraft.com/archives/category/web-server-survey/.

14. Java. *¿Qué es la tecnología Java y por qué lo necesito?* [En línea] [Citado el: 7 de Febrero de 2013.] http://www.java.com/es/download/faq/whatis_java.xml.

15. Java. *¿Qué es Java Enterprise Edition (Java EE)?* [En línea] [Citado el: 8 de Febrero de 2013.] http://www.java.com/es/download/faq/techinfo.xml.

16. Oracle. *Productos Oracle WebLogic.* [En línea] [Citado el: 24 de Abril de 2013.] http://www.oracle.com/es/products/middleware/appserver/index.html.

17. IBM developerWorks. *Introducción a WebSphere.* [En línea] [Citado el: 24 de Abril de 2013.] http://www.ibm.com/developerworks/ssa/websphere/newto/.

18. Apache TomEE. [En línea] [Citado el: 24 de Abril de 2013.] http://tomee.apache.org/apache-tomee.html.

19. GlassFish Server. [En línea] [Citado el: 24 de Abril de 2013.] http://www.oracle.com/technetwork/middleware/glassfish/overview/index.html.

20. Apache Tomcat - Welcome! [En línea] [Citado el: 24 de Abril de 2013.] http://tomcat.apache.org/.

21. Plumbr. *Most popular application servers.* [En línea] [Citado el: 24 de Abril de 2013.] http://plumbr.eu/blog/most-popular-application-servers.

22. *IBM - España.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.ibm.com/es/es/.

23. *Conmutación de Paquetes.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.uazuay.edu.ec/estudios/sistemas/teleproceso/apuntes_1/conmutacion_paquetes.htm.

24. RAND. *Corporation Provides Objective Research Services and Public Policy Analysis.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.rand.org/.

25. **Nudelman, Gustavo.** Universidad Tecnológica Nacional. [En línea] [Citado el: 13 de Mayo de 2013.] http://www.electron.frba.utn.edu.ar/materias/95-0419/archivos/20100614_Procesamiento_Paralelo.pdf.

26. Internet Society. *Brief History of the Internet - Internet Timeline.* [En línea] [Citado el: 10 de Febrero de 2013.] http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet.

27. Clusters. *Administración del Cluster.* [En línea] [Citado el: 23 de Abril de 2013.] http://clusterfie.epn.edu.ec/clusters/Definiciones/definiciones4.html.

28. **Almeroth, K. y Quinn, B.** fags. *IP Multicast Applications: Challenges and Solutions.* [En línea] 9 de 2001. [Citado el: 13 de Junio de 2013.] http://www.faqs.org/rfcs/rfc3170.html.

29. Ejemplos de Java y C/Linux. *Programación de sockets en C de Unix/Linux.* [En línea] 4 de 2 de 2007. [Citado el: 13 de Junio de 2013.] http://www.chuidiang.com/clinux/sockets/sockets_simp.php#sockets.

30. **Márquez García, Francisco Manuel.** *UNIX, Programación avanzada.* s.l. : RA-MA EDITORIAL, 2004. 978-84-7897-603-4.

31. **Ridruejo, Francisco Javier, Agirre, Jon y Miguel-Alonso, José.** XIV Jornada de Paralelismo. *Estrategias de instalación y gestión de clusters con software libre.* [En línea] Septiembre de 2003. [Citado el: 15 de 6 de 2013.] http://www.sc.ehu.es/acwmialj/papers/jornadas03_a.pdf.

32. **Oracle.** Oracle Technology Network. *JNDI.* [En línea] [Citado el: 14 de 6 de 2013.] http://www.oracle.com/technetwork/java/overview-142035.html.

33. Red Hat Customer Portal. *Buddy Replication.* [En línea] [Citado el: 21 de Mayo de 2013.] https://access.redhat.com/site/documentation/en-US/JBoss_Enterprise_Application_Platform/4.3/html/Cache_Tree_Cache_Guide/Clustered_Cache___Using_Replication-Buddy_Replication.html.

34. **Oracle.** Oracle Technology Network. *JavaBeans FAQ: General Questions.* [En línea] [Citado el: 12 de Junio de 2013.] http://www.oracle.com/technetwork/java/javase/faq-135947.html.

35. **La Rosa Agramonte, Alberto Jesús y Del Castillo Rodríguez, Hiran.** *Herramienta para la instalación y configuración de clústeres del Contenedor de Servlets.* La Habana : s.n., 2009.

36. Nagios. *Nagios XI.* [En línea] [Citado el: 13 de Mayo de 2013.] http://www.nagios.com/products/nagiosxi/.

37. Red Hat. *JBoss Operations Network.* [En línea] [Citado el: 24 de Abril de 2013.] http://www.redhat.com/products/jbossenterprisemiddleware/operations-network/.

38. Red Hat JBoss Enterprise Middleware. *Open source middleware.* [En línea] [Citado el: 1 de Mayo de 2013.] http://www.redhat.com/products/jbossenterprisemiddleware/.

39. MSDN Microsoft. *Visual Studio.* [En línea] [Citado el: 16 de Junio de 2013.] http://msdn.microsoft.com/es-es/library/ms235632%28v=vs.80%29.aspx.

40. MSDN Microsft. *Visual Studio Resources.* [En línea] [Citado el: 16 de Junio de 2013.] http://msdn.microsoft.com/en-us/vstudio/cc136611.aspx.

41. **Rivillo Matía, Norberto J.** *Control mediante IDE Eclipse, de una placa de entradas-salidas multiples conectada al microcontrolador STM32L discovery.* 2012.

42. **María del Carmen, Tellería Prieto.** *SISTEMA DE INFORMACIÓN CLÍNICA.* s.l. : En Informática Salud 2013, 2012. SLD107 BEHIQUE-SIC.

43. Netbeans. [En línea] [Citado el: 11 de Febrero de 2013.] http://netbeans.org/.

44. **Cerda, Felipe.** *NetBeans.* [En línea] [Citado el: 11 de Febrero de 2013.] http://api.ning.com/files/PcRFEwSyAz6w2k4r-KsPqXzHa3EJ6JtOUDOCxjxcpwJJFWH8XaIgnfUFGtj2TURFJc3LgS7W9tCq-NsM68o3ZBHIMDWV12-W/netbeans65es_cl.pdf.

45. **Tituaña Cumbal, Walter Celiano y Torres Cañizares, Edwin Jesús.** *Elaboración de un manual de la plataforma Netbeans Ide para la Disicom.* s.l. : LATACUNGA, 2009.

46. **Oracle.** Document Oracle. *Project Swing.* [En línea] 2010. [Citado el: 14 de Junio de 2013.] http://docs.oracle.com/javase/1.5.0/docs/guide/swing/.

47. Enterprise Architect. *Herramienta de diseño UML y herramienta CASE UML para desarrollo de software.* [En línea] [Citado el: 19 de Mayo de 2013.] http://sparxsystems.com.ar/products/ea.html.

48. Rational Rose Enterprise. *Paquete Rational Rose.* [En línea] [Citado el: 19 de Mayo de 2013.] http://www.rational.com.ar/herramientas/roseenterprise.html.

49. *Guión Visual Paradigm for UML.* [En línea] 2011. [Citado el: 7 de Febrero de 2013.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

50. *Adapting microsoft SQL server for cloud computing.* **Bernstein, Philip A.** 12005313, s.l. : Data Engineering (ICDE), 2011 IEEE 27th International Conference on, 2011. 1063-6382.

51. MySQL. *MySQL Customers.* [En línea] [Citado el: 20 de Mayo de 2013.] http://www.mysql.com/customers/.

52. PostgreSQL. *About postgres.* [En línea] [Citado el: 20 de Mayo de 2013.] http://www.postgresql.org/about/.

53. Apache DB Proyect. *What is Apache Derby?* [En línea] [Citado el: 7 de Febrero de 2013.] http://db.apache.org/derby/.

54. *C# Station.* [En línea] [Citado el: 19 de Mayo de 2013.] http://csharp-station.com/.

55. **Rodríguez del Toro, Tamara.** *Herramienta de autoría de material educativo en lenguaje C#.* [En línea] 2011. [Citado el: 15 de Junio de 2013.]

56. **Tymoschuk, Jorge.** Universidad Tecnológica Nacional. *Paradigmas de Programación.* [En línea] 2009. [Citado el: 12 de Junio de 2013.] http://labsys.frc.utn.edu.ar/ppr-2009/Unidad%20I%20-%20POO%20Avanzada/Apunte/Unidad%20I%20-%20POO%20Avanzada.pdf.

57. **Microsoft.** Microsoft. *Visual Studio Test Professional.* [En línea] 2012. [Citado el: 15 de Junio de 2013.] http://www.microsoft.com/visualstudio/esn/products/visual-studio-test-professional-2012#product-edition-testpro.

58. **Selenium.** Document Selenium. *What is Selenium.* [En línea] [Citado el: 15 de Junio de 2013.] http://docs.seleniumhq.org/.

59. **Apache.** Apache. *Apache Jmeter.* [En línea] [Citado el: 15 de Junio de 2013.] http://jmeter.apache.org/.

60. *Oracle VM VirtualBox.* [En línea] [Citado el: 5 de Junio de 2013.] https://www.virtualbox.org/.

61. Microsoft. *Windows Virtual PC.* [En línea] [Citado el: 5 de Junio de 2013.] http://www.microsoft.com/es-ES/download/details.aspx?id=3702.

62. *OpenVZ Linux Containers Wiki.* [En línea] [Citado el: 5 de Junio de 2013.] http://openvz.org/Main_Page.

63. Apple. *Todo lo que necesitas saber sobre OS X.* [En línea] [Citado el: 5 de Junio de 2013.] http://www.apple.com/es/osx/what-is.

64. **Brito Jiménez, Arístides Lescay y Estrada, Pedro Manuel.** *Componentes web Psicología y Psiquiatría del Sistema Integral para la Atención Primaria de Salud.* La Habana : s.n., 2012.

65. **Jacobson, y otros, y otros.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000. 84-7829-036-2.

66. *Ingeniería de software.* **Sommerville, Ian.** Madrid : Pearson Educación, 2005, Vol. Séptima edición. 84-7829.

67. *Sistema de Información Medioambiental.* **Taboada González, José A. y Cotos Yáñez, José Manuel.** Madrid : Carlos Iglesia, 2005. 84-9754-056-6.

68. The Apache Tomcat Connector. *AJP Protocol Reference - AJPv13.* [En línea] [Citado el: 24 de Abril de 2013.] http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html.

69. Web container configuration. *Use of jboss-web.xml.* [En línea] [Citado el: 15 de Mayo de 2013.] http://www.jboss.ru/translate/jboss24/ch05s22.html.

70. MSDN Microsoft. *Serialización.* [En línea] [Citado el: 15 de Junio de 2013.] http://msdn.microsoft.com/es-es/library/7ay27kt9%28v=vs.80%29.aspx.

71. *Network traffic analysis and intrusion detection using packet sniffer.* **Qadeer, Mohammed Abdul, y otros, y otros.** Singapore : Communication Software and Networks, 2010. ICCSN '10. Second International Conference on , 2010. Vols. En Communication Software and Networks, 2010. 978-1-4244-5726-7.

72. *Assessing the Fidelity of COTS 802.11 Sniffers.* **Serrano, Pablo, Zink, Michael y Kurose, Jim.** Rio de Janeiro : INFOCOM 2009, IEEE, 2009. 978-1-4244-3512-8.

73. *Virtual machine showdown: Stack versus registers.* **SHI, Yunhe.** 4, s.l. : ACM Transactions on Architecture and Code Optimization (TACO), 2008, Vol. 4.

74. **Pressman, Roger S.** *Ingeniería de software un enfoque práctico.* Madrid : Madrid. MacGraw-Hill, 2002.

75. *Planos Arquitectónicos: El Modelo de "4+ 1" Vistas de la Arquitectura del Software.* **Kruchten, Philippe.** 6, s.l. : IEEE Software, 1995, Vol. 12.

76. ComuSOFT. *Modelo Vista Controlador – Definición y Características.* [En línea] [Citado el: 19 de Mayo de 2013.] http://www.comusoft.com/modelo-vista-controlador-definicion-y-caracteristicas.

77. *Arquitectura cliente-servidor.* [En línea] [Citado el: 19 de Mayo de 2013.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

78. **González, Enrique.** *Arquitectura en capas. Un camino hacia los procesos distribuidos.* Chile : s.n., 2006.

79. **García Llinás, Luis Fernandeo.** *Programación orienta a objetos en Java.* Bogotá, Colombia : Universidad del Norte de la U, 2010. 9789587410624.

80. *¿Qué es Java Web Start y cómo se ejecuta?* [En línea] [Citado el: 16 de Mayo de 2013.] http://www.java.com/es/download/faq/java_webstart.xml.

81. **Java.** *Java Runtime Environment.* [En línea] [Citado el: 5 de Junio de 2013.] http://es.download.cnet.com/Java-Runtime-Environment-JRE-64-Bit/3000-2378_4-75317067.html.

82. Internationalizing a GUI. *NetBeans IDE Tutorial.* [En línea] [Citado el: 15 de Mayo de 2013.] https://netbeans.org/kb/docs/java/gui-automatic-i18n.html.