Universidad de las Ciencias Informáticas
Facultad Regional Mártires de Artemisa

Administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando JMX.

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

Autor: Lisdey Pérez Hernández

Tutores: MSc. Yamila Vigil Regalado

Lic. Gilberto Ramón Justiniani Fernández

Artemisa, Cuba
Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Facultad Regional “Mártires de Artemisa” de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Lisdey Pérez Hernández

_____

Firma del Autor

Msc. Yamila Vigil Regalado

_____

Firma del Tutor

Lic. Gilberto Ramón Justiniani

_____

Firma del Co-tutor

"Es justamente la posibilidad de realizar un sueño lo que hace que la vida sea interesante."

Paulo Coelho

AGRADECIMIENTOS

Quisiera agradecer en especial, a mis padres pues sin ellos nada de esto habría sido posible; gracias por existir.

A mi novio por estar siempre a mi lado, incluso en los momentos más difíciles, por su apoyo incondicional y todo su cariño.

A todas mis amistades por formar parte de mi vida y siempre creer en mí.

A mis tutores por su orientación y ayuda durante la realización de esta investigación.

A todos los profesores que han contribuido a mi formación profesional.

En general a todos los que de una forma u otra han hecho posible la realización de este sueño.

DEDICATORIA

Dedico este trabajo de diploma a mis padres por su incondicional e infinito amor, por toda la confianza que siempre depositaron en mí, por inspirarme en todo momento y guiarme a ser todo lo que soy.

A mis hermanitos por todo su cariño, por retarme a ser una mejor persona cada día, espero que este trabajo les sirva de inspiración y les demuestre que con voluntad todo es posible.

A mi novio por ser lo mejor que me ha pasado en estos 5 años, por todo su amor y comprensión, por creer en mí y darme fuerzas para seguir adelante.

RESUMEN

La administración de aplicaciones distribuidas y en general de los sistemas en red, ocupa un lugar primordial en el éxito de toda organización. Sin embargo, las crecientes demandas de escalabilidad, modularidad, disponibilidad y alto desempeño requeridas por las aplicaciones empresariales han traído como consecuencia un aumento significativo en el costo y la complejidad de la administración de las mismas.

Hoy en día, el desarrollo de aplicaciones distribuidas en la web está dirigido a lograr mayores niveles de interactividad, eficiencia y tiempo real. Con este objetivo surge el protocolo WebSocket y a su vez un conjunto de tecnologías que lo implementan. Entre estas tecnologías se encuentra el marco de trabajo jWebSocket el cual está diseñado además para funcionar como un servidor web. Actualmente jWebSocket cuenta con una herramienta de administración remota que solo permite realizar operaciones básicas sobre las aplicaciones desarrolladas con este marco de trabajo. Esta situación trae consigo que los administradores no tengan un control eficiente y centralizado de los recursos utilizados por el servidor, ni la certeza de que las aplicaciones se estén ejecutando correctamente. En la mayoría de los casos, esto afecta la disponibilidad de los servicios en general, así como la confiabilidad de los usuarios en estas aplicaciones y en las empresas que las manejan.

Por lo tanto, la presente investigación se centra en el desarrollo del módulo JMXPlugIn, el cual permite la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando para este propósito la tecnología JMX. En el presente documento se describen los principales aspectos del proceso de desarrollo del módulo en cuestión. Además, con el propósito de validar la presente investigación, se trazó una estrategia cuyos resultados demuestran que el módulo JMXPlugIn garantiza un posible aumento en los niveles de disponibilidad y confiabilidad de los servicios brindados por los servidores jWebSocket.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El vertiginoso avance de la informática y las comunicaciones en los últimos años ha propiciado la evolución de los sistemas informáticos. Primeramente surgieron los enormes y lentos sistemas centrales que ejecutaban todas las aplicaciones dentro de sí mismos, haciendo prácticamente imposible la interacción con los usuarios y elevando grandemente los costos de hardware. Con el avance de la tecnología, aparecieron los ordenadores personales que permitían la ejecución de sistemas autónomos más pequeños e independientes, sin embargo, seguían siendo centralizados y por tanto incomunicados entre sí. El desarrollo de las tecnologías de red y la estandarización de sus protocolos hizo posible conectar computadoras permitiendo la transferencia de datos a alta velocidad, provocando un cambio completo de paradigma en la arquitectura de las aplicaciones informáticas. En este contexto aparece el término de aplicaciones distribuidas, convirtiendo en una realidad la comunicación entre aplicaciones y la distribución de procesos entre diferentes equipos. (CABALLÉ, y otros, 2007)

Las aplicaciones distribuidas son aquellas cuyos componentes se ejecutan en entornos separados, normalmente en diferentes plataformas conectadas a través de una red. Estas aplicaciones se pueden clasificar en dos tipos principales: cliente-servidor y aplicaciones de n-capas. Las aplicaciones cliente-servidor cuentan con dos procesos principales. Estos procesos son: el servidor, que se encarga básicamente de enviar, recibir y procesar los datos y el cliente, que se ejecuta en el equipo del usuario siendo sus funciones principales solicitar datos al servidor, presentarlos al usuario y enviar los cambios nuevamente al servidor. Las aplicaciones distribuidas en n-capas cuentan con diferentes procesos distribuidos en capas no solo lógicas, sino también físicas. Los procesos se ejecutan en diferentes equipos, que pueden incluso residir en plataformas o sistemas operativos completamente distintos. (CABALLÉ, y otros, 2007)

La evolución de las aplicaciones distribuidas ha estado fuertemente ligada al desarrollo de las tecnologías de red, en específico, al surgimiento y desarrollo de la World Wide Web o simplemente la Web. La concepción de la Web como plataforma universal para el despliegue de aplicaciones y las crecientes demandas de escalabilidad, modularidad, disponibilidad y alto desempeño requeridas por las aplicaciones empresariales, han

influido enormemente en la adopción de un diseño distribuido. Actualmente, debido al número cada vez mayor de unidades conectadas a las redes y la creciente cantidad de servicios que se brindan, ha aumentado sustancialmente el tamaño y la complejidad de estas aplicaciones. A su vez, la gran importancia que tiene hoy en día el correcto funcionamiento de las aplicaciones distribuidas para lograr la productividad y eficiencia de las organizaciones, hace imprescindible el análisis y monitorización continua de las mismas. Sin embargo, el hecho de interconectar múltiples componentes trabajando de forma independiente a través de la red, trae como consecuencia un aumento significativo en el costo y la complejidad de la administración de estas aplicaciones.

La administración de aplicaciones distribuidas y en general de los sistemas en red, ocupa un lugar primordial en el éxito de toda organización. El hecho de poder controlar la estructura organizativa de las aplicaciones, garantizando la disponibilidad y eficiencia de todos sus módulos y posibilitando la centralización de los procedimientos y configuraciones, hace de la administración un elemento de obligatoria implementación para cualquier sistema en red. La necesidad de estandarización de los protocolos de red fue reconocida desde hace aproximadamente veinte años y desde ese momento han surgido diferentes protocolos que han permitido regular la comunicación entre los componentes de una red informática. Entre los principales estándares que fueron definidos se encuentran: SNMP (*Simple Network Management Protocol*, Protocolo Simple de Administración de Redes) y TMN (*Telecommunication Management Network*, Red de Gestión de Telecomunicaciones). (WITTNER, 2003)

El protocolo de gestión de red SNMP facilita el intercambio de información de administración entre dispositivos de red. Este protocolo fue pensado para ser simple y fácil de aplicar, lo cual ha sido la razón de su éxito y amplia aceptación, pero a su vez ha limitado su uso. Las principales desventajas de SNMP son: alta vulnerabilidad en cuestiones de seguridad y alto consumo de ancho de banda en comparación con otros estándares. Sin embargo, la limitación de seguridad fue solventada con su última versión SNMPv3, la cual además refuerza las prestaciones del protocolo. SNMP es en la actualidad, el protocolo de gestión de red más extendido y conocido. Por otra parte se encuentra el estándar TMN, el cual ha sido diseñado para gestionar redes complejas de telecomunicaciones (WITTNER, 2003). El objetivo de este estándar es proporcionar una

estructura de red organizada para conseguir la interconexión de diversos tipos de sistemas de administración con los equipos de telecomunicación. No obstante, TMN presenta varios inconvenientes, entre los que se destacan: un alto grado de complejidad, no soporta SNMP como protocolo de gestión de red y su funcionamiento se basa en una arquitectura distribuida antigua.

Los protocolos de red han ido evolucionando con el paso de los años. La introducción y desarrollo de las tecnologías web ha traído consigo la creación de nuevas versiones de los protocolos ya existentes y la aparición de nuevos estándares. En este contexto emergen las soluciones web para la administración de sistemas y de redes. Este hecho impulsa el desarrollo de estándares con arquitecturas más abiertas y flexibles dentro de los cuales se destaca JMX (*Java Management Extensions*, Administración de Extensiones de Java). JMX es un estándar del lenguaje de programación Java que define la arquitectura de gestión, los patrones de diseño, y las APIs (*Application Programming Interfaces*, Interfaces de Programación de Aplicaciones) que permiten la creación de soluciones web distribuidas, dinámicas y modulares para la gestión de recursos Java (MAHMOUD, 2004).

Uno de los principales objetivos de JMX es permitir que una amplia gama de aplicaciones y sistemas de gestión sean capaces de acceder y controlar las aplicaciones administradas. JMX logra esta meta con la implementación de una arquitectura en capas que utiliza un conjunto común de componentes y proporciona acceso remoto a través de múltiples protocolos tales como SNMP, TMN, HTTP (*HyperText Transfer Protocol*, Protocolo de Transferencia de Hipertexto), RMI (*Remote Method Invocation*, Invocación Remota de Métodos) y WBEM (*Web-Based Enterprise Management*, Gestión Empresarial Basado en Web). JMX es nativa del lenguaje de programación Java lo que permite una administración natural, eficiente y ligera de las aplicaciones desarrolladas con este lenguaje. Esta tecnología ofrece a los desarrolladores Java los medios para instrumentar soluciones de administración distribuidas de última generación, permitiendo integrarlas fácilmente con los sistemas de administración y monitorización existentes. (*Extensible Markup Language*, Lenguaje de Marcas Extensible) (BOWKER, 2001).

Actualmente la tecnología JMX sigue siendo una de las más usadas por los desarrolladores de aplicaciones Java. Este hecho se demuestra con la larga lista de compañías y/o proyectos que implementan o brindan soporte a esta tecnología, entre los cuales se destaca: los contenedores de servlets Apache Tomcat y Jetty; los servidores de aplicaciones GlassFish, JBoss y WebLogic; las herramientas de administración de sistemas IBM Director, HP Open View e Hyperic y los marcos de trabajo Spring e Hibernate, por citar algunos de los más conocidos. (Java Community Process, 2012) Por todas las prestaciones y ventajas que esta tecnología ofrece y dado el hecho de que JMX es la respuesta oficial de Java para la administración de sistemas, su utilización es una de las mejores soluciones para enfrentar el reto de la administración remota de aplicaciones distribuidas en un entorno en constante evolución.

Hoy en día, el desarrollo de aplicaciones distribuidas en la Web está dirigido a lograr mayores niveles de interactividad, eficiencia y tiempo real. Con este objetivo surge el protocolo WebSocket, el cual proporciona un canal de comunicación bidireccional y *full-duplex*[1] sobre un único *socket*[2] TCP (*Transmission Control Protocol*, Protocolo de Control de Trasmisiones). Este protocolo está diseñado para ser implementado en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. Entre los servidores que soportan Websocket para el desarrollo de aplicaciones hoy día se encuentran, la Pasarela Websocket de Kaazing[3], Jetty WebSocketServlet[4], Socket.IO[5], django-websocket del proyecto Python[6] y jWebsocket[7]. Este último es un marco de trabajo de código abierto para el desarrollo de aplicaciones web estacionarias y móviles basado en Java en el lado del servidor. (SCHULZE, 2011) Además, permite que los clientes puedan ser desarrollados en múltiples lenguajes, tales como: JavaScript, C# y Python.

Este marco de trabajo está diseñado además para funcionar como un servidor web, proporcionando un conjunto importantes de funcionalidades que lo hacen más robusto y

[1] Cualidad de los elementos que permiten la entrada y salida de datos de forma simultánea.
[2] Método para la comunicación entre un programa del cliente y un programa del servidor en una red.
[3] http://kaazing.com/
[4] http://www.eclipse.org/jetty/
[5] http://socket.io/
[6] http://pypi.python.org/pypi/django-websocket
[7] https://jwebsocket.org/

flexible. jWebSocket en su función de servidor web debe ser capaz de controlar de forma eficiente todas las aplicaciones que se estén ejecutando, por lo tanto es necesario que cuente con herramientas de administración que lo permitan. Actualmente jWebSocket cuenta con una herramienta de administración remota que solo permite realizar operaciones básicas sobre las aplicaciones desarrolladas con este marco de trabajo.

Teniendo en cuenta los planteamientos anteriores se define la siguiente **situación problemática:**

La herramienta de administración existente en el marco de trabajo jWebSocket no permite mantener una gestión centralizada y remota de los procesos que realizan las aplicaciones desarrolladas con este marco de trabajo. Además, hoy en día no es posible llevar a cabo la monitorización y control remoto de los servicios, dispositivos y aplicaciones desarrolladas con jWebSocket.

Esta situación trae consigo que los administradores no tengan un control eficiente y centralizado de los recursos utilizados por los servidores de aplicaciones, ni la certeza de que las aplicaciones se estén ejecutando correctamente. En la mayoría de los casos, esto afecta la disponibilidad de los servicios en general y a su vez la confiabilidad de los usuarios en dichas aplicaciones. Esto hecho afecta además la confiabilidad hacia las empresas que manejan estas aplicaciones impactando negativamente en su economía. El hecho de que el marco de trabajo jWebSocket no cuente con una gestión centralizada de sus aplicaciones conlleva a un aumento significativo en la complejidad de la administración de las mismas. De esta forma, se hace mucho más engorroso el trabajo de los administradores al tener un módulo de administración por cada una de las aplicaciones que se están ejecutando.

De la problemática existente surge la siguiente interrogante que define el **Problema Científico:** ¿Cómo garantizar una mejor administración remota de las aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket que permita mayores niveles de disponibilidad y confiabilidad de los servicios?

Por tanto para el desarrollo de la presente investigación se define como **objeto de estudio** la administración remota de aplicaciones distribuidas; delimitándose como

campo de acción la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket.

Para dar solución al problema planteado, se define como **objetivo general** desarrollar un módulo para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando JMX que permita mayores niveles de disponibilidad y confiabilidad de los servicios.

Para dar cumplimiento al objetivo general se definen las siguientes **preguntas científicas**:

- ¿Cuáles son los fundamentos teórico-metodológicos de la administración remota de aplicaciones distribuidas que sustentan la investigación?
- ¿Cuál es la situación actual de la administración remota de aplicaciones distribuidas?
- ¿Cómo desarrollar un módulo para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando JMX que permita mayores niveles de disponibilidad y confiabilidad de los servicios?
- ¿Cómo comprobar la capacidad y potencialidad del módulo propuesto para la administración remota de aplicaciones distribuidas?

Para dar respuesta a las preguntas científicas planteadas se proponen las siguientes **tareas de la investigación**:

1. Fundamentación teórico-metodológica de la administración remota de aplicaciones distribuidas.
2. Análisis de la situación actual de la administración remota de aplicaciones distribuidas.
3. Desarrollo de un módulo para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando JMX que permita mayores niveles de disponibilidad y confiabilidad de los servicios.
4. Comprobación de la capacidad y potencialidad del módulo propuesto para la administración remota de aplicaciones distribuidas.

Los **métodos científicos** utilizados en la presente investigación son:

Teóricos:

- **Método Análisis Documental:** Mediante este método se realiza un estudio de la documentación referente a la administración remota de aplicaciones distribuidas y las herramientas utilizadas actualmente para lograrla, de esta forma se incorporan a la presente investigación las experiencias y sugerencias analizadas.
- **Histórico-Lógico:** Permite analizar la trayectoria completa del proceso de administración de aplicaciones distribuidas, así como el estudio histórico del mismo que permite ver deficiencias y proponer soluciones acorde a las necesidades.
- **Analítico-Sintético:** Este método permite analizar toda la teoría sobre la administración remota de aplicaciones distribuidas que pueda servir para desarrollar mejor el diseño del módulo, y poder aplicar así estos conocimientos en la práctica de manera que se adquiera una mayor preparación sobre el tema en cuestión.
- **Modelación:** Este método permite realizar una representación de la situación que se analiza. Permite obtener mediante diagramas y objetos una mayor compresión del problema, así como desarrollar un modelo para el módulo en cuestión a partir de la situación problemática.

Se definen las siguientes **variables de la investigación:**

- Variable Independiente:
  - Módulo para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando JMX.
- Variables Dependientes:
  - Grado de disponibilidad de los servicios del servidor jWebSocket.
  - Grado de confiabilidad en los servicios del servidor jWebSocket.

Teniendo en cuenta el cumplimiento de las tareas de la investigación propuestas se esperan como **posible resultado:**

- Módulo para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando JMX.

Para una mejor comprensión de la investigación, el contenido ha sido desglosado en tres capítulos, además de las conclusiones generales, recomendaciones, referencias bibliográficas y los anexos que complementan el trabajo realizado. Los capítulos han sido estructurados de la siguiente manera:

Capítulo 1. Fundamentación Teórica: Se realiza la fundamentación teórico-metodológica de la investigación. Se expone un estudio del estado del arte sobre el proceso actual de administración remota de aplicaciones distribuidas así como un estudio de las metodologías, tecnologías y herramientas a tener en cuenta para el desarrollo de la solución propuesta.

Capítulo 2. Características, Análisis y Diseño del Sistema: Brinda una fundamentación de la solución propuesta, a partir de la cual se describen las actividades de análisis de la solución, seguidas por la descripción de los procesos del módulo y de la etapa de diseño.

Capítulo 3. Implementación y Validación del Sistema: Se describe la etapa de implementación que conlleva a la obtención del módulo. Se establece una estrategia de validación para certificar la capacidad y potencialidad del módulo para la administración remota de aplicaciones distribuidas. Además, se exponen los resultados alcanzados y el aporte tanto social como económico del módulo desarrollado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se elabora la base teórico-metodológica que sustenta y respalda la presente investigación. Con este objetivo se exponen los principales conceptos asociados al dominio del problema permitiendo una mejor comprensión del objeto de estudio y campo de acción en cuestión. Se realiza además un análisis de soluciones existentes con características similares y un estudio de las metodologías, tecnologías y herramientas a tener en cuenta para el desarrollo de la solución propuesta.

1.1. Conceptos Asociados al Dominio del Problema

La administración de aplicaciones distribuidas y en general de los sistemas en red, ocupa un lugar primordial en el éxito de toda organización. El objeto de estudio y el campo de acción que guían la presente investigación están enmarcados en este tema. Por lo tanto, los conceptos fundamentales en los cuales se sustenta el presente trabajo de diploma se refieren a las aplicaciones distribuidas, la administración remota y la tecnología JMX.

La evolución de las tecnologías de redes marca el surgimiento del término aplicaciones distribuidas. En la actualidad, este concepto es ampliamente utilizado y por tanto ha sido definido por varios autores. Entre estos conceptos se encuentra el que define a las aplicaciones distribuidas como una aplicación de software que se distingue por los siguientes tres aspectos:

1. Sus funcionalidades se han separado (distribuidos) en un conjunto de unidades funcionales que se comunican y cooperan entre sí. Cada unidad funcional tiene su estado y sus operaciones que permiten manipular el estado.
2. Las unidades funcionales se pueden asignar a diferentes máquinas; una máquina por su parte puede albergar más de una unidad funcional al mismo tiempo.
3. Las unidades funcionales se comunican entre sí (por ejemplo a través de invocación remota de procedimientos -*remote procedure call*-, paso de mensajes, entre otros). (CABALLÉ, y otros, 2007)

Esta definición tiene en cuenta elementos importantes de las aplicaciones distribuidas, ya que las caracteriza principalmente como un conjunto de unidades funcionales que se comunican y cooperan entre sí. Sin embargo, otros autores plantean que las aplicaciones distribuidas son en particular aquellas aplicaciones que involucran a múltiples usuarios cooperando juntos en un entorno distribuido. (BALTER, et al., 2009) Al analizar este concepto se observa un enfoque diferente con respecto a las aplicaciones distribuidas, resaltando la utilidad de las mismas para la interacción conjunta de múltiples usuarios.

A su vez, otro de los conceptos de aplicaciones distribuidas plantea que son sistemas de software que se ejecutan en dos o más equipos conectados por una red de ordenadores. (RUSSELL, y otros, 2009) Este último enfoque aporta nuevos elementos a la definición general ya que asume que los componentes de una aplicación distribuida se comunican a través de una red de computadores. Teniendo en cuenta las definiciones mencionadas anteriormente y dadas las características de la presente investigación, el autor asume como concepto de aplicaciones distribuidas a aquellas aplicaciones cuyos componentes se ejecutan en unidades funcionales separadas, los cuales se comunican y cooperan entre sí a través de una red de ordenadores.

Con el objetivo de garantizar el correcto funcionamiento de las aplicaciones distribuidas en las cuales se basa la presente investigación se hace imprescindible lograr una administración eficiente de las mismas. Por lo tanto se hace necesario la definición del concepto de administración remota de aplicaciones distribuidas, para lo cual fueron valorados diferentes términos asociados a este concepto.

La palabra administración proviene etimológicamente del latín "*administratĭo*", conceptualizado por la Real Academia de la Lengua Española como "graduar o dosificar el uso de algo, para obtener mayor rendimiento de ello o para que produzca mejor efecto". En la actualidad la administración se ha convertido en todo un proceso, por lo que ha adquirido nuevas perspectivas y ha sido definida como: "el proceso de planear, organizar, dirigir y controlar el uso de los recursos para lograr los objetivos organizacionales". (CHIAVENATO, 2004)

La necesidad de una correcta administración no se reduce solamente a los sistemas convencionales. La creciente evolución de los sistemas informáticos en red ha propiciado un nuevo enfoque de este término. Es el caso de la definición de administración de sistemas en red la cual "…se refiere a las actividades, métodos, procedimientos y herramientas que pertenecen a la operación, administración, mantenimiento y aprovisionamiento de los sistemas en red…" (PRIETO, 2008). Otro de los enfoques de este término plantea que la administración de sistemas en red comprende todas las medidas necesarias para garantizar el funcionamiento eficaz y eficiente de un sistema y sus recursos de conformidad con los objetivos de una organización. (LATASA, 2009) A pesar de que las definiciones mencionadas anteriormente caracterizan correctamente la administración de sistemas en red, no resaltan la importancia del término remoto para este proceso en las aplicaciones distribuidas en general.

La administración de las aplicaciones distribuidas se convierte en un proceso extremadamente complicado al realizarse directamente en cada una de las unidades funcionales que las componen. El hecho de que los elementos de estas aplicaciones se encuentren dispersos en varios equipos de cómputo a lo largo de la red provoca que la administración en la mayoría de los casos sea realizada remotamente. La palabra remoto es definida por la Real Academia de la Lengua Española como "distante, apartado, distante en el espacio o en el tiempo". Este término es inherente al concepto de redes de computadoras en sí, ya que la gran mayoría de los elementos de estas redes se encuentran separados físicamente.

Atendiendo al concepto de aplicación distribuida definida para la presente investigación y teniendo en cuenta los conceptos mencionados anteriormente, el autor asume que la administración remota de aplicaciones distribuidas es el conjunto de actividades, métodos, procedimientos y herramientas que garantizan el funcionamiento eficaz y eficiente de un sistema en red, evitando el despliegue de personal o dispositivos en la localización de los componentes o recursos distribuidos.

El constante crecimiento y complejidad de las redes informáticas y de telecomunicaciones ha traído consigo la evolución de la administración remota. Este hecho ha propiciado a su vez el surgimiento de estándares de administración de redes

cada vez más eficientes. En la actualidad algunos de los más utilizados son SNMP, TNM, WBEM y JMX. Este último es un estándar del lenguaje de programación Java para la gestión y monitorización de recursos, el cual ha sido definido para el desarrollo de la presente investigación.

Diferentes autores han caracterizado el estándar de administración de redes JMX. Entre estas definiciones se encuentra la que conceptualiza a JMX como una tecnología Java que proporciona herramientas para la gestión y monitorización de aplicaciones, objetos del sistema, dispositivos y redes orientadas a servicios. (GUI, 2011) Este concepto abarca aspectos determinantes de esta tecnología. Sin embargo, otros autores se refieren a JMX de una forma mucho más amplia, definiéndolo como un estándar que especifica la arquitectura de gestión, los patrones de diseño, las APIs y los servicios para la gestión de aplicaciones y redes en el lenguaje de programación Java. (MARON, 2010) De esta forma se ofrece una caracterización más profunda de JMX reflejándolo como un estándar robusto, eficiente y claramente definido para la administración remota.

Por último es imprescindible mencionar el concepto dado por Éamonn McManus, líder técnico del equipo de desarrollo de esta tecnología en Sun MicroSystems, el cual define que JMX proporciona las herramientas para crear soluciones distribuidas, modulares y dinámicas para la gestión y monitorización de dispositivos, aplicaciones y redes. La API JMX define la noción de MBeans[8] u objetos manejables, los cuales exponen los atributos y operaciones de manera tal que permite el acceso de las aplicaciones de administración remota. (MCMANUS, y otros, 2006)

Teniendo en cuenta las características de la presente investigación, así como los diferentes enfoques mencionados anteriormente sobre esta tecnología, el autor define como JMX al estándar que proporciona las herramientas para crear soluciones distribuidas, modulares y dinámicas para la administración y monitorización remota de servicios, dispositivos y aplicaciones. La tecnología JMX es nativa del lenguaje de programación Java, por lo tanto permite la creación de soluciones naturales, eficientes y ligeras. Además presenta una arquitectura altamente escalable que permite la gestión de recursos con un mínimo impacto en el diseño de las aplicaciones.

[8] Objeto Java que representa un objeto gestionable.

La arquitectura de JMX está dividida en tres niveles: capa de Instrumentación, capa de Agente y capa de Gestión de Servicios Distribuidos. El nivel de Instrumentación provee una especificación para la implementación de recursos JMX manejables llamados MBeans, los cuales pueden ser, aplicaciones, la implementación de un servicio, un dispositivo, un usuario, entre otros. El nivel de Agente provee una especificación que permite controlar directamente los MBeans llamado MBeanServer, exportándolos para el acceso remoto de las aplicaciones de administración. La capa de Gestión de Servicios Distribuidos define las interfaces de gestión y los componentes que permiten operar sobre los agentes o sobre jerarquías de agentes. Esta capa proporciona acceso remoto, seguridad y otras funcionalidades de los MBeans. La combinación de estas tres capas proporciona la arquitectura para el diseño e implementación de soluciones de administración completamente funcionales. (MARON, 2010)

Los MBeans de la capa de Instrumentación se encuentran divididos en cuatro tipos:

- Standard MBeans: son los más simples de diseñar e implementar, usan el estilo de convenciones de nombres de JavaBeans para especificar estáticamente su interfaz de gestión.
- Dynamic MBeans: deben implementar una interfaz de programación específica. Para mayor flexibilidad exportan su interfaz de gestión en tiempo de ejecución.
- Open MBeans: son una implementación de los MBeans dinámicos. Se basan en tipos de datos básicos, de fácil uso y que se auto describen para lograr una administración universal.
- Model MBeans: son una implementación de los MBeans dinámicos que son altamente configurables y que se auto describen en tiempo de ejecución. Proporcionan un tipo de MBean genérico con un comportamiento por defecto para la instrumentación dinámica de recursos. (Sun Microsystems Inc., 2006)

1.2. Análisis de Soluciones Existentes

Los sistemas de administración de redes han ido evolucionando con el paso de los años. Desde las crípticas consolas de administración de los primeros dispositivos de red hasta los grandes sistemas de administración modernos capaces de gestionar redes dinámicas remotamente. En la actualidad, tras el vertiginoso avance de Internet y la aparición de las

aplicaciones distribuidas, los sistemas de gestión deben ser capaces de controlar y monitorizar servidores de base de datos, servidores web, servidores de aplicaciones y redes de almacenamiento alrededor de mundo. A su vez, los usuarios de estos sistemas demandan modernas interfaces web de administración que posibiliten la gestión y monitorización remota de recursos.

Hoy en día, han sido desarrolladas varias tecnologías que permiten la creación de sistemas de gestión de redes que cumplan con estos requerimientos. Entre estas se destaca la tecnología JMX de la plataforma Java. Esta plataforma incluye varias características que permiten la implementación de complejas soluciones de administración de red. Algunas de estas características son: multiplataforma, el trabajo con la red es parte del núcleo de Java y la carga de clases a través de la red es de forma dinámica y segura. La tecnología JMX es considerada un paso clave para lograr la administración universal de redes dinámicas con completa compatibilidad con versiones anteriores y con soporte a las infraestructuras de sistemas de gestión de redes existentes. (LI, 2002)

JMX es una de las tecnologías más usadas por los desarrolladores de aplicaciones Java. Este hecho se demuestra con la larga lista de compañías y/o proyectos que implementan o brindan soporte a esta tecnología. (Java Community Process, 2012) Entre estos proyectos se pueden mencionar los servidores de aplicaciones GlassFish, jBoss y WebLogic; los contenedores de servlets Tomcat y Jetty; el marco de trabajo Spring, la pasarela WebSocket Kaazing, entre otros. A continuación se realiza una descripción detallada del soporte brindado por cada uno de estos proyectos a la tecnología JMX, finalizando con un análisis general en el cual se reflejan las pautas que marcan en la actualidad el desarrollo de soluciones de administración usando esta tecnología.

1.2.1. Servidores de Aplicaciones

GlassFish[9] es un servidor de aplicaciones de código abierto iniciado por Sun Microsystems para la plataforma Java EE (Enterprise Edition, Edición Empresarial) y actualmente es patrocinado por Oracle Corporation. El servidor GlassFish aprovecha completamente las ventajas que brinda la tecnología JMX. Con el objetivo de simplificar

[9] http://glassfish.java.net/

la gestión de recursos del servidor, GlassFish proporciona la API AMX (AppServer Management eXtension, Gestión de Extensiones del Servidor de Aplicaciones) basada en la tecnología JMX. Esta API permite la implementación de MBeans para la configuración y monitorización del servidor. De esta forma es posible la utilización de diferentes herramientas para la administración de GlassFish sin la utilización directa de JMX. Sin embargo, es posible acceder a los MBeans creados por la API AMX a través de aplicaciones de administración basadas en JMX. (KOU, 2010)

AMX es una librería propia de GlassFish y proporciona una abstracción de la especificación JSR-77[10] de JMX. Esta librería proporciona una poderosa herramienta de monitorización. De esta forma los desarrolladores y administradores son capaces de controlar determinados aspectos de GlassFish mediante código Java, sin la necesidad de entender la API de JMX o la complejidad de los factores de monitorización y recopilación de estadísticas. (CHAMBERS, 2006) Además, GlassFish proporciona un mecanismo de autogestión, en el cual las reglas de administración son construidas con JMX. El modelo de eventos presente en estas reglas está basado en el mecanismo de notificaciones de JMX. (KOU, 2010)

JBoss[11] es un servidor de aplicaciones J2EE de código abierto implementado completamente en Java. El núcleo de su arquitectura es llamado JBoss Microcontainer. Este componente está integrado con la tecnología JMX permitiendo una fácil gestión del servidor y de las aplicaciones desplegadas sobre el mismo. La implementación de JMX en JBoss forma parte de su columna vertebral de integración donde los componentes son conectados. De esta forma estos componentes son administrados fácilmente utilizando JMX. JBoss utiliza JMX para cargar los servicios como MBeans. El patrón de MBeans en este servidor consiste en una serie de funcionalidades las cuales presentan un mecanismo de notificaciones JMX. Este mecanismo envía notificaciones cuando los MBeans son creados, iniciados, detenidos y eliminados. (JBoss Community , 2008)

JBoss implementa una versión personalizada de los Model MBeans de JMX, llamados XMBean. Este tipo de MBeans permiten especificar la interfaz de administración de los componentes a través de un descriptor XML. Los XMBeans, proveen un mecanismo

[10] http://jcp.org/en/jsr/detail?id=77
[11] http://www.jboss.org/

sencillo para la especificación de los metadatos requeridos por los MBeans dinámicos, incluyendo a su vez las notificaciones de eventos de JMX. Por otra parte JBoss implementa adaptadores que permiten el remoto acceso al MBeanServer de JMX. Los adaptadores actuales incluyen HTML, una interfaz RMI y EJB (Enterprise JavaBeans, JavaBeans Empresariales). La seguridad para el acceso remoto a la consola JMX es implementado usando la seguridad basada en roles estándar de J2EE. (JBoss Community , 2008)

WebLogic[12] es un servidor de aplicaciones J2EE desarrollado por BEA Systems y posteriormente adquirido por Oracle Corporation. WebLogic Server implementa la especificación de JMX con el objetivo de proporcionar una administración de servicios extensible y abierto. Todos los recursos del servidor pueden ser gestionados utilizando los servicios basados en JMX. A su vez, las aplicaciones y servicios de terceros que se ejecutan dentro de WebLogic Server pueden ser administradas a través de JMX. Algunos de los recursos que pueden ser administrados a través de los servicios JMX son el proveedor JMS (Java Message Service, Servicio de Mensajes de Java) y el contenedor JDBC (Java Database Connectivity).

Los recursos administrados de WebLogic Server proporcionan además métricas de desempeño y otras informaciones sobre su estado de ejecución a través de MBeans. Entre las funcionalidades adicionales que permite la implementación de JMX en WebLogic Server se encuentra la configuración y gestión de la seguridad del servidor, la creación de escuchadores de eventos que envían notificaciones bajo determinadas circunstancias y la creación de servicios de monitorización a los diferentes MBeans que son exportados. (Oracle WebLogic Server, 2012)

En general, los servidores de aplicaciones analizados presentan una tendencia a implementar una infraestructura de administración basada en la tecnología JMX. GlassFish implementa una librería propia que proporciona una abstracción de JMX. De esta forma la infraestructura de administración se acopla perfectamente a la arquitectura del servidor proporcionando una gestión y monitorización de los recursos más eficiente. Por otra parte, el servidor de aplicaciones JBoss integra JMX al núcleo de su

[12] http://www.oracle.com/us/products/middleware/application-server/index.html

arquitectura, permitiendo que todos sus componentes sean fácilmente administrados. A su vez la solución de administración del servidor WebLogic permite la gestión y monitorización a través de JMX de todos recursos del servidor, así como de las aplicaciones y servicios de terceros que se ejecuten en el mismo.

1.2.2. Contenedores de Servlets

Tomcat [13] es un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Todas las versiones de Tomcat 5.x o superior soportan de forma nativa JMX. Esto significa que una vez que se establezcan las conexiones JMX, se obtiene acceso inmediatamente a determinados componentes internos de Tomcat. La herramienta de administración de Tomcat incluye un ligero proxy HTTP de JMX, llamado JMXProxy, que permite la ejecución de comandos simples a través de solicitudes de URI (Uniform Resource Identifier, Identificar Uniforme de Recurso). Este proxy permite realizar tres acciones principales: obtener información, modificar información e invocar métodos de los MBeans. Sin embargo, el hecho de realizar acciones a través de solicitudes URI es muy complicado. Para utilizar JMXProxy es necesario poseer un alto conocimiento tanto de la tecnología JMX como de las funcionalidades para la administración de Tomcat. (Apache Tomcat 7, 2012)

Jetty[14] es un servidor web, un cliente-servidor WebSocket y un contenedor de servlets implementado completamente en Java. Es un proyecto de código abierto que forma parte de la comunidad *Eclipse Foundation*. La integración de Jetty con JMX utiliza la implementación del MBeanServer proporcionado por la máquina virtual de Java (JVM). Esta integración se basa en una implementación propia del Dynamic MBean, llamada Object MBean. Esta implementación permite envolver un POJO (Plain Old Java Object) arbitrario en un MBean permitiendo su exportación. La creación de MBeans está coordinada por la implementación de la clase MBeanContainer. El servidor Jetty y sus componentes utilizan una clase Container para mantener un árbol de contención de componentes permitiendo la notificación de cambios en ese árbol. La clase

[13] http://tomcat.apache.org/
[14] http://www.eclipse.org/jetty/

MBeanContainer escucha los eventos de los contenedores de MBeans, creándolos y eliminándolos según sea necesario. (Eclipse Foundation, 2012)

A modo de resumen, las soluciones de administración implementadas en los contenedores de servlets analizados presentan diferentes enfoques. La implementación de la tecnología JMX en Tomcat es bastante sencilla, cuenta con un ligero proxy JMX para HTTP que permite ejecutar comando a través de soluciones URI. Esta solución es poco amigable y bastante complicada de utilizar ya que es necesario poseer un alto conocimiento sobre las funcionalidades que permiten administrar el servidor. Por otra parte Jetty implementa una especificación personalizada de JMX. Este servidor utiliza el MBeanServer que provee la JVM. Sin embargo, implementa un tipo específico de objeto JMX manejable, los Object MBeans, los cuales se adaptan perfectamente a los recursos que gestiona el servidor. De esta forma Jetty crea un mecanismo de administración eficiente y fácil de utilizar.

1.2.3. Marco de Trabajo Spring[15]

Es el marco de trabajo de código abierto más popular para el desarrollo de aplicaciones Java. El soporte para JMX en Spring provee las funcionalidades necesarias para integrar de forma fácil y sencilla las aplicaciones desarrolladas con este marco de trabajo en una infraestructura JMX. Spring permite el registro automático de cualquier clase como un MBean de JMX a través de ficheros de configuración. Ofrece la posibilidad de crear un nuevo MBeanServer para controlar los recursos a exportar o simplemente utilizar uno ya en funcionamiento.

Este marco de trabajo proporciona un mecanismo flexible para controlar las interfaces de gestión de las clases a exportar. Este mecanismo permite crear las interfaces de gestión básicamente de dos formas: utilizando anotaciones de Java o a través de interfaces de programación. Además, Spring permite exportar remotamente los MBeans utilizando los conectores definidos en la especificación JSR-160[16] para lo cual implementa un conjunto de clases que funcionan como un contenedor para la API de JMX. Este hecho hace

[15] http://www.springsource.org/
[16] http://jcp.org/en/jsr/detail?id=160

posible que no sea necesario que los desarrolladores posean un alto conocimiento sobre esta tecnología.

Otra de las funcionalidades que brinda Spring es la creación de *proxies* que permitan redireccionar las llamadas a MBeans registrados en MBeanServers locales o remotos. Además, la implementación de JMX de este marco de trabajo ofrece soporte para el envío de notificaciones JMX. Todas las funcionalidades mencionadas anteriormente están diseñadas para funcionar evitando el acoplamiento de los componentes de las aplicaciones a gestionar con las interfaces o clases tanto de Spring como de JMX. (JOHNSON, y otros, 2010)

En resumen, la solución de administración proporcionada por el marco de trabajo Spring es altamente flexible. Este marco de trabajo permite a los desarrolladores integrar fácilmente sus aplicaciones a una infraestructura JMX. Una de las características más relevantes de esta integración es el mecanismo que permite especificar las interfaces de gestión de las clases a exportar. Este mecanismo tiene como objetivo controlar los atributos y métodos que serán exportados de cada una de las clases. Sin embargo, presenta ciertas desventajas.

Las anotaciones Java son altamente flexibles. Sin embargo, su utilización implica realizarle grandes cambios al código fuente de las aplicaciones, afectando la limpieza del mismo y convirtiendo su mantenimiento en una tarea complicada. Las interfaces de programación evitarían todos los problemas mencionados anteriormente. Sin embargo, su uso imposibilita la personalización de los metadatos de las clases a exportar. Este hecho trae como consecuencia que sea necesario un alto conocimiento de las funcionalidades y atributos de estas clases por parte de los administradores.

1.2.4. Pasarela WebSocket Kaazing[17]

Esta pasarela permite una comunicación *full-duplex* y de alto rendimiento a través de Internet utilizando el protocolo WebSocket de HTML5. Además proporciona la gestión y monitorización remota de sus servicios utilizando la tecnología JMX. Actualmente el servicio que puede ser monitorizado utilizando JMX es el manejo interno de sesiones de

[17] http://kaazing.com/

los usuarios conectados a la pasarela. A través de este servicio es posible controlar las actividades de los usuarios y cerrar las sesiones de la pasarela. Como valor agregado a este servicio es posible recibir notificaciones JMX al crearse y cerrarse una sesión. (Kaazing Corporation, 2012)

En síntesis, se observa que la solución de administración implementada por la pasarela de comunicación Kaazing es muy pobre. A pesar de que implementa la tecnología JMX, solamente permite la gestión y monitorización del servicio para el manejo interno de sesiones de los usuarios conectados a la pasarela. De esta forma se desaprovecha la variedad de funcionalidades que brinda la tecnología JMX para la creación de una solución de administración eficiente y robusta.

1.2.5. Análisis General

En resumen, la amplia aceptación y utilización de JMX por parte de la mayoría de los proyectos Java más relevantes, demuestra que esta tecnología es uno de los estándares más eficientes para la creación de soluciones de administración. Luego del análisis de las soluciones anteriores se observa que el uso de JMX varía en dependencia de determinados factores. Algunos de estos factores son: el tipo de infraestructura que se desea implementar, la arquitectura donde se implantará la solución de administración, los tipos de recursos que serán gestionados, los usuario a los cuales va dirigida esta solución, entre otros. En general, se observa una tendencia a crear una infraestructura JMX basada en las características específicas de cada proyecto proporcionando de esta forma una gestión y monitorización de recursos más eficiente. Sin embargo, la solución de administración brindada por el marco de trabajo Spring es una de las más genéricas y eficientes de todas las analizadas.

La presente investigación tiene como objetivo principal desarrollar un módulo para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket. Es importante destacar que este marco de trabajo está diseñado además para funcionar como un servidor web y que a su vez promueve la utilización del marco de trabajo Spring para el desarrollo de aplicaciones. Actualmente, jWebSocket integra el *IoC Container* (*Inversion of Control Container,* Contenedor de Invesión de Control) de Spring así como lo módulos de Autenticación y Validación. Por lo tanto, en la presente

investigación es seleccionada la solución de administración brindada por Spring para formar parte del módulo a desarrollar. Sin embargo, el mecanismo utilizado por Spring para controlar las interfaces de gestión de las clases que serán exportadas, presenta ciertas desventajas. En consecuencia, se hace necesario el diseño e implementación de una solución de administración utilizando la tecnología JMX, la cual ofrezca una mejora al mecanismo de Spring y a su vez responda a las características propias de jWebSocket.

1.3. Metodología para el Desarrollo de Software

Para el desarrollo de la solución propuesta se hace necesaria la selección de una metodología de desarrollo acorde a las características propias del proyecto y que a su vez garantice la eficiencia y calidad del proceso de desarrollo de software. Con este objetivo fue analizada la metodología robusta RUP en representación de los métodos tradicionales, así como las metodologías ágiles SCRUM, XP y SXP. De cada una de estas se exponen sus características principales, lo cual permite adquirir los elementos necesarios para determinar cuál es la más adecuada para guiar el proceso de desarrollo de software de la solución propuesta.

RUP es una propuesta de proceso para el desarrollo de software orientado a objetos que utiliza el Lenguaje Unificado de Modelado (UML, *Unified Model Language*) para describir todo el proceso. Consta de 4 fases principales: Concepción, Elaboración, Construcción y Transición.

Esta metodología está basada en componentes, lo cual quiere decir que el sistema en construcción está formado por componentes interconectados a través de interfaces bien definidas. Sus características principales son:

1. Guiado/Manejado por casos de uso.
2. Centrado en arquitectura.
3. Iterativo e Incremental.
4. Desarrollo basado en componentes.
5. Utilización de un único lenguaje de modelación.
6. Proceso Integrado. (JACOBSON, et al., 2000)

Las metodologías robustas centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto. Otra característica importante dentro de este enfoque son los altos costos al implementar un cambio, por lo cual no ofrecen una buena solución para proyectos donde el entorno es volátil. (FIGUEROA, y otros, 2011)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software promoviendo el trabajo en equipo. XP se basa en una retroalimentación continua entre el cliente y el equipo de desarrollo, una comunicación fluida entre todos los participantes, así como la simplicidad en las soluciones implementadas. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. (BECK, 2002). El ciclo de vida ideal de esta metodología consta de 6 fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software, más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de programadores. Scrum plantea el desarrollo en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nuevas funcionalidades. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de *software* como RUP o Extreme Programming. (FIGUEROA, y otros, 2011). Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características son: desarrollo mediante iteraciones y reuniones sistemáticas a lo largo del proyecto. (SCHWABER, y otros, 2011)

SXP es una metodología de desarrollo de software compuesta por las metodologías SCRUM, para la gestión de proyectos de forma eficiente, y XP, para la ingeniería de software, donde el cliente es parte del equipo de trabajo hasta llegar al éxito del producto. Consta de 4 fases principales:

- Planificación-Definición: se establece la visión del proyecto, se fijan las expectativas y se asegura el financiamiento del mismo.
- Desarrollo: se realiza la implementación del sistema en iteraciones.
- Entrega: es la puesta en marcha del producto, en esta etapa se realiza la integración, se entregan las pruebas del sistema y la documentación en general.
- Mantenimiento: fase de soporte del producto para el cliente.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, requisitos muy cambiantes o imprecisos, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ayuda al equipo de desarrollo a trabajar juntos en la misma dirección, permitiendo además visualizar de forma clara el avance de las tareas a realizar. (PEÑALVER, 2008)

Teniendo en cuenta las características de la metodología SXP y dadas las particularidades de la presente investigación, la cual es considerada un proyecto de corta duración, con un entorno en constante cambio y con escaso personal participante, se selecciona esta metodología para el desarrollo de la solución propuesta.

1.4. Herramientas y Tecnologías

1.4.1. Entorno Integrado de Desarrollo (IDE)

Un IDE (acrónimo en inglés de *Integrated Development Environment*) es un entorno de programación que integra varias herramientas con el objetivo de facilitar el desarrollo de software sobre uno o varios lenguajes de programación. Por lo tanto, se hace necesaria la selección de un IDE acorde al lenguaje de programación a utilizar y que a su vez facilite el desarrollo del módulo propuesto. A continuación se realiza una caracterización de varios IDEs que permitirá adquirir los elementos necesarios para determinar cuál es el más adecuado para el desarrollo de solución propuesta.

Eclipse SDK es un entorno integrado de desarrollo de código abierto y multiplataforma. En un principio Eclipse fue desarrollado por IBM y posteriormente su desarrollo fue llevado a cabo por Eclipse Foundation, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto. Eclipse basa su funcionalidad en módulos (en inglés plug-in) que se adaptan a las necesidades del programador. Este

mecanismo de módulos es una plataforma ligera para componentes de software que permite el uso de diferentes lenguajes de programación, entre los cuales se destaca Java. (Eclipse, 2011). Es considerado uno de los IDE más completos para el desarrollo de aplicaciones Java.

NetBeans IDE 7.0.1 Es una herramienta desarrollada por Sun Microsystems. Permite el desarrollo de aplicaciones de escritorio, web y móviles. Brinda soporte a varios lenguajes de programación entre los cuales se destaca Java. Es un producto libre y gratuito sin restricciones de uso. Su misión consiste en evitar tareas repetitivas, facilitar la escritura correcta de código, disminuir el tiempo de depuración e incrementar la productividad del desarrollador. Cuenta con un depurador, perfilador de integración, herramientas para refactorizaciones, completamiento de código y control de versiones de archivos. (NetBeans, 2011).

NetBeans es considerado a su vez uno de los IDE más completos y potentes para el desarrollo de aplicaciones Java, siendo además uno de los más utilizados. Teniendo en cuenta todas sus características y dado que se tiene una mayor experiencia y familiarización con esta herramienta por parte del equipo de trabajo, es seleccionado para el desarrollo de la solución propuesta en la presente investigación.

1.4.2 Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo costos en términos de tiempo y dinero. Teniendo en cuenta que la metodología de desarrollo de software SXP, la cual ha sido seleccionada para el desarrollo de la solución propuesta, define a UML (*Unified Modeling Language*) como lenguaje de modelado, se hace necesaria la selección de una herramienta CASE compatible con este lenguaje. A continuación se caracterizan algunas de estas herramientas más representativas del lenguaje UML, lo cual permitirá adquirir los elementos necesarios para determinar cuál es la más adecuada para especificar y diseñar la solución propuesta.

Rational Rose es una herramienta CASE que da soporte al modelado visual con UML cubriendo todo el ciclo de vida de un proyecto. Es compatible con la metodología RUP

permitiendo crear los diagramas que son generados durante el proceso de ingeniería en el desarrollo del software. Permite la autogeneración de código a partir de modelos y viceversa para lenguajes como Java, así como el desarrollo multiusuario. El sistema operativo admitido para esta herramienta es Windows y posee licencia privativa. (IBM,2012)

Visual Paradigm 6.4 es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Proporciona excelentes facilidades de interoperabilidad con otras aplicaciones y compatibilidad entre versiones. Además, permite realizar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta posee licencia gratuita y comercial, es multiplataforma, posibilita la integración con NetBeans IDE y posee soporte para el Rational Rose. (Visual Paradigm, 2012)

Teniendo en cuenta las políticas de software libre establecidas por la Universidad de las Ciencias Informáticas se decide utilizar para la especificación y diseño del presente trabajo diploma, la herramienta Visual Paradigm 6.4 Enterprise Edition. Esta herramienta es multiplataforma y presenta una licencia comercial la cual posee la Universidad, siendo de esta forma la más adecuada para el modelado de la presente investigación.

1.4.3 Herramienta de Control de Versiones

Con el objetivo de garantizar una gestión de cambios eficiente y segura durante el desarrollo de la solución propuesta se hace necesaria la selección de una herramienta de control de versiones que permita el trabajo colaborativo entre varios puestos de trabajo y a su vez que proporcione una mayor seguridad y disponibilidad de los datos. Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal característica es mantener el historial de cambios y modificaciones que se han realizado sobre dichos archivos. Entre estos sistemas se destaca **Subversion**, el cual es una herramienta de control de versiones de código abierto y multiplataforma basada en un repositorio cuyo funcionamiento se asemeja al de un sistema de ficheros.

Esta herramienta cuenta con varias características entre las cuales se debe mencionar por su importancia para la presente investigación: el soporte para desarrolladores que posibilita el acceso a todas las funcionalidades de Subversion desde diferentes IDEs entre los cuales se destaca NetBeans. Además Subversion permite el desarrollo paralelo permitiendo que miembros individuales del equipo de trabajo puedan completar diferentes partes y versiones de un proyecto al mismo tiempo. Otras características relevantes de esta herramienta son:

- Los directorios son versionados.
- Resolución de conflictos de forma interactiva.
- Gestiona de manera eficaz los archivos binarios.
- Bloqueo de archivos.
- Vinculaciones para lenguajes de programación.
- Vinculación de varios repositorios. (Apache Software Foundation, 2011)

Considerando las características de Subversion, la experiencia que se posee con esta herramienta por parte del equipo de trabajo y las políticas de software libre establecidas por la Universidad de las Ciencias Informáticas, se decide su utilización para garantizar una mayor seguridad y disponibilidad de los datos en la presente investigación.

1.4.4 Cliente de Control de Versiones

Con el objetivo de interactuar de forma sencilla y eficiente con la herramienta de control de versiones seleccionada en la presente investigación, se hace necesaria la selección de un cliente de control de versiones compatible con SubVersion. A continuación se realiza una caracterización de algunas de estos clientes que permitirá adquirir los elementos necesarios para determinar cuál es el más adecuado para interactuar con el servidor de SubVersion.

TortoiseSVN es un cliente gratuito de código abierto bajo licencia GPL para el sistema de control de versiones Subversion. Una de sus características más importantes es que está diseñado como una extensión de *shell* de Windows, lo cual permite su fácil integración con Windows Explorer. Esta herramienta provee un menú contextual ampliado en el explorador de Windows que permite ejecutar fácilmente los comandos de Subversion. (KUNG, 2011) TortoiseSVN es considerado uno de los clientes de control de

versiones más completos y flexibles, sin embargo, presenta como desventaja que solo puede ser utilizado con la plataforma Windows.

RapidSVN es una plataforma de interfaz gráfica de usuario gratuito, de código abierto, bajo licencia GPL y escrito en C++ para el sistema de control de versiones Subversion. Esta herramienta permite ejecutar fácilmente los comandos de Subversion a través de la interfaz gráfica. Sus características principales son: de fácil uso, eficiente, multiplataforma y rápido. Además podemos mencionar que ha sido traducido a varios lenguajes y proporciona un soporte completo para Unicode. (RapidSVN, 2011)

La presente investigación es desarrollada teniendo en cuenta las políticas de software libre establecidas por la Universidad de las Ciencias Informáticas por lo tanto se selecciona RapidSVN como cliente de control de versiones.

1.4.5 Herramienta para la construcción de proyectos

Maven es una herramienta para la gestión y construcción de proyectos Java. Permite compilar, ejecutar pruebas o realizar distribuciones tratando de forma automática las dependencias del proyecto. Una de sus características más importantes es la actualización en línea a través de repositorios. Maven es capaz de descargar nuevas actualizaciones de las librerías de las que depende un proyecto dado y de igual manera subir una nueva distribución a un repositorio de versiones, permitiendo el acceso de todos los usuarios. Esta herramienta posee una licencia de software libre y es utilizada para compilar y construir los módulos de la solución propuesta en la presente investigación.

1.4.6 Marcos de Trabajo

Para el desarrollo de la solución propuesta se hace uso de los marcos de trabajo jWebSocket y Spring. El marco de trabajo jWebSocket forma parte del campo de acción en el cual se basa la presente investigación. Por lo tanto es considerado una tecnología imprescindible para el desarrollo del módulo propuesto y a su vez se convierte en el objeto al cual está dirigido el aporte del presente trabajo de diploma. Por otra parte, el marco de trabajo Spring brinda un soporte a la tecnología JMX altamente flexible y genérico que permite integrar las aplicaciones desarrolladas con este marco de trabajo

en una infraestructura JMX. Por lo tanto, es utilizado en la presente investigación para el trabajo con esta tecnología. A continuación se realiza una caracterización detallada de cada uno de estos marcos de trabajo para un mayor entendimiento de su función y utilización en el desarrollo de la solución propuesta.

jWebSocket es un marco de trabajo de código abierto para el desarrollo de aplicaciones web estacionarias y móviles basado en Java en el lado del servidor. Además, permite que los clientes puedan ser desarrollados en múltiples lenguajes, tales como: JavaScript, C# y Python. Es una nueva tecnología orientada al desarrollo de aplicaciones basadas en WebSocket que proporcionan altos niveles de velocidad, escalabilidad, seguridad y trabajo en tiempo real. El servidor jWebSocket está diseñado para funcionar como servidor de comunicaciones o como servidor web, brindando total flexibilidad. jWebsocket como servidor web proporciona un conjunto importantes de funcionalidades. Además su arquitectura extensible mediante plug-in permite añadir fácilmente características adicionales a un sistema independiente. Todas las características de este marco de trabajo muestran su fortaleza y flexibilidad para el desarrollo de aplicaciones web estacionarias y móviles, multiplataforma, multisectorial y compatible con todos los navegadores. (SCHULZE, 2011)

Actualmente un grupo de estudiantes de la Facultad Regional de la UCI “Mártires de Artemisa” integran el equipo de desarrollo del marco de trabajo jWebSocket, del cual constituyen aproximadamente el 50%. La presente investigación forma parte del proyecto jWebSocket por lo cual se define la utilización de este marco de trabajo para el desarrollo de la solución propuesta.

Spring es una plataforma Java que proporciona una infraestructura integral para el desarrollo de aplicaciones Java. Este marco de trabajo es una solución ligera y potente para la construcción de aplicaciones empresariales. Integra varias tecnologías de Java Enterprise Edition, entre las cuales se destaca el soporte para JMX. Esta integración brinda las funcionalidades necesarias para integrar de forma fácil y sencilla las aplicaciones desarrolladas con este marco de trabajo en una infraestructura JMX. Por lo tanto es utilizado en la presente investigación debido a las potencialidades que brinda para el trabajo con esta tecnología. En el epígrafe 1.2.3 se ofrece una explicación más

detallada sobre las funcionalidades que brinda el marco de trabajo Spring para el trabajo con la tecnología JMX.

1.4.7 Lenguajes usados para el desarrollo de la solución

Java es el lenguaje de programación que se utiliza en el marco de trabajo jWebSocket para el desarrollo de aplicaciones del lado del servidor, el cual ha sido seleccionado para el desarrollo de la presente investigación. Java es un lenguaje moderno, de alto nivel y orientado a objetos. Actualmente Java es el lenguaje de programación más utilizado[18] para el desarrollo de aplicaciones por las ventajas que brinda, entre estas se encuentra:

- Interpretado y compilado a la vez: genera ficheros de clases compiladas, siendo luego interpretadas por la Máquina Virtual de Java, la cual mantiene el control sobre las clases que se estén ejecutando.
- Multiplataforma: el código Java puede ejecutarse en cualquier sistema operativo que tenga instalada la Máquina Virtual de Java. Esta es una de las principales características que ha favorecido el crecimiento y difusión de este lenguaje.
- Seguro: la Máquina Virtual, al ejecutar el código java, realiza comprobaciones de seguridad.
- Distribuido: proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir *sockets*, establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.
- Multiproceso: soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. (GOSLING, y otros, 2005)

Lenguaje de Modelado Unificado (UML)

Teniendo en cuenta que la metodología de desarrollo de software SXP, seleccionada en la presente investigación, define a **UML** como lenguaje de modelado, este es asumido para modelar y diseñar la solución propuesta. El Lenguaje Unificado de Modelado (UML en sus siglas en inglés) es una técnica para la especificación de sistemas informáticos en todas sus fases. Es el lenguaje de modelado de sistemas de software más conocido y

[18] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. (RUMBAUGH, y otros, 2007).

Lenguaje de Marcas Extensible (XML)

XML se utiliza específicamente para la estructuración de los ficheros de configuración de la solución propuesta en la presente investigación. Es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es muy similar a HTML pero su función principal es describir datos, permitiendo su la lectura a través de diferentes aplicaciones. Es utilizada para estructurar, almacenar e intercambiar información. XML posibilita la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. (VAN DER VLIST, 2002).

1.4.8 Tecnología JMX

JMX es utilizada en la presente investigación con el objetivo de lograr la administración remota de las aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket. Esta tecnología es nativa de Java por lo cual permite una gestión eficiente, natural y ligera de las aplicaciones desarrolladas con este lenguaje. Además presenta una arquitectura en tres capas, altamente escalable que permite la administración de aplicaciones con un impacto mínimo en su diseño.

JMX es una tecnología que define una arquitectura de gestión, los patrones de diseño, la API y los servicios para la monitorización y administración de aplicaciones desarrolladas con Java. Esta tecnología proporciona una manera simple y estándar de gestionar recursos, tales como aplicaciones, dispositivos y servicios. (Sun Microsystems Inc., 2006) Esta tecnología ofrece a los desarrolladores de Java los medios para instrumentar soluciones de administración distribuidas de última generación, permitiendo integrarlas fácilmente con los sistemas de administración y monitorización existentes. Los usos típicos de JMX son: consultar y cambiar las configuraciones de las aplicaciones, recopilar estadísticas sobre el comportamiento de las aplicaciones, así como notificar cambios de estado y condiciones erróneas. (MAHMOUD, 2004)

Conclusiones Parciales

En el presente capítulo se analizaron los conceptos indispensables para la comprensión del proceso de administración remota de aplicaciones distribuidas utilizando la tecnología JMX. Se realizó además un estudio de soluciones existentes con características similares. El cual arrojó como resultado la necesidad de diseñar e implementar una solución de administración utilizando el mecanismo brindado por Spring para el trabajo con la tecnología JMX y que a su vez responda a las características propias del marco de trabajo jWebSocket. Además fueron investigadas las tendencias actuales en cuanto a metodologías, herramientas y tecnologías para el desarrollo de software. De esta forma fue posible definir las más adecuadas para darle solución a la problemática planteada, teniendo en cuenta las características de la presente investigación y partiendo del concepto de migración a software libre que sigue el país actualmente.

CAPÍTULO 2. CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se realiza la caracterización del módulo a desarrollar teniendo en cuenta la metodología de desarrollo SXP. Con este objetivo, se desarrolla el modelado del negocio para lograr una mejor comprensión del contexto del problema y se realiza la propuesta de solución del módulo, describiendo cómo debe funcionar y destacando sus características distintivas. Además se especifican los Requisitos Funcionales y No Funcionales del módulo; se elaboran las historias de usuarios y las tareas de ingeniería asociadas a las mismas, a su vez se realizan el modelado del diseño utilizando metáforas y el diagrama de componentes del módulo.

2.1 Propuesta de Solución

El módulo a desarrollar, denominado JMXPlugIn, debe permitir la administración remota de las aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket. Con este objetivo, el módulo permite a los administradores invocar remotamente las funcionalidades de las aplicaciones que se estén ejecutando en un servidor jWebSocket determinado. Este hecho posibilita mantener el control y monitorización centralizado de los recursos manejados por el servidor, simplificando enormemente la complejidad en la administración del mismo. Por otra parte, a través de JMXPlugIn es posible administrar remotamente determinadas funcionalidades del servidor jWebSocket, tales como: mostrar información sobre las conexiones existentes, así como sobre los plugins y filtros cargados en el servidor; exportar las características de los servidores en funcionamiento, del motor en el cual se ejecutan estos servidores y del nodo donde se ejecuta el servidor jWebSocket.

Además, con el objetivo de ofrecer una mayor flexibilidad a los desarrolladores que utilicen el marco de trabajo jWebSocket, JMXPlugIn posee un mecanismo sencillo para integrar los plugins o clases desarrolladas a la infraestructura de administración del módulo. Para lograr esta integración se utilizan ficheros de configuración, aportando gran flexibilidad a la solución. En general, este mecanismo permite administrar atributos y operaciones tanto de plugins como de clases así como brinda la oportunidad de

configurar notificaciones de eventos para cada uno de los elementos que son administrados. Esta funcionalidad posibilita centralizar la administración de las aplicaciones desarrolladas utilizando el marco de trabajo jWebSocket, aportando flexibilidad y simplicidad al proceso.

A su vez, JMXPlugIn permite el acceso remoto a todas sus funcionalidades a través de los protocolos RMI y HTTP, posibilitando una mayor comodidad al administrador así como brindando mayor flexibilidad y usabilidad a la solución. Además, con el objetivo de garantizar la seguridad del módulo se cuenta con un componente para la autenticación el cual utilizará MD5[19] como algoritmo de encriptación y establece la comunicación a través de un protocolo seguro. En general, las funcionalidades descriptas anteriormente influyen positivamente en los niveles de disponibilidad y confiabilidad de los servicios al garantizar una administración centralizada y remota de las aplicaciones que se están ejecutando en los servidores jWebSocket.

2.2 Planificación del Proyecto por Roles

Para lograr una mayor organización y eficiencia en el desarrollo de la solución propuesta se hace necesario la definición de los diferentes roles que intervienen en el proceso de desarrollo de software. De esta forma se le asigna a cada integrante del proyecto una responsabilidad con el objetivo de coordinar e integrar sus esfuerzos para lograr un objetivo común. A continuación en la **Tabla # 1** se muestra la asignación de roles pertenecientes al proyecto, así como las principales responsabilidades de cada uno de estos.

Tabla # 1 Planificación del proyecto por roles.

Rol	Responsabilidad	Nombre
Líder del Proyecto	Es el responsable de que el proyecto se está llevando a cabo de acuerdo con las prácticas establecidas y que todo funciona según lo planeado. Su principal trabajo es remover impedimentos y reducir riesgos del producto. Asegura	Msc. Yamila Vigil Regalado

[19] Message-Digest Algorithm

	que se consiguen los objetivos de cada iteración.	
Gerente	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la selección de objetivos y requerimientos. Tiene la responsabilidad de controlar el progreso y trabaja junto con el Jefe de Proyecto en la reducción de la Lista de Reserva del Producto.	Alexander Schulze
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, aportan ideas y experiencias para el beneficio del sistema en desarrollo.	Alexander Schulze, Rebecca Schulze
Cliente	Participa en las tareas que involucran la Lista de Reserva del Producto.	Alexander Schulze
Programador	Elabora el código de las nuevas funcionalidades a implementar. Escribe las pruebas unitarias.	Lisdey Pérez Hernández
Analista	Escribe las Historias de Usuario y las pruebas funcionales para validar su implementación.	Lisdey Pérez Hernández
Encargado de Pruebas	Es el encargado de ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.	Lisdey Pérez Hernández
Arquitecto	Se vincula con el analista y el diseñador	Lisdey Pérez Hernández

	ya que su trabajo tiene que ver con la estructura y el diseño del sistema. Ayuda en el diseño de las metáforas.	

2.3 Modelo de Historias de Usuario del Negocio

Una de las actividades más importantes definidas en la metodología SXP es la confección del Modelo de Historias de Usuario del Negocio. En este artefacto se realiza una descripción precisa del negocio en cuestión, especificando los usuarios y trabajadores que intervienen así como su interacción con las historias de usuario. A continuación en la **Fig. 1** se muestra el diagrama de historias de usuario del negocio correspondiente al módulo a desarrollar, el cual describe las actividades principales que conforman la definición de administración remota de aplicaciones distribuidas en la presente investigación.

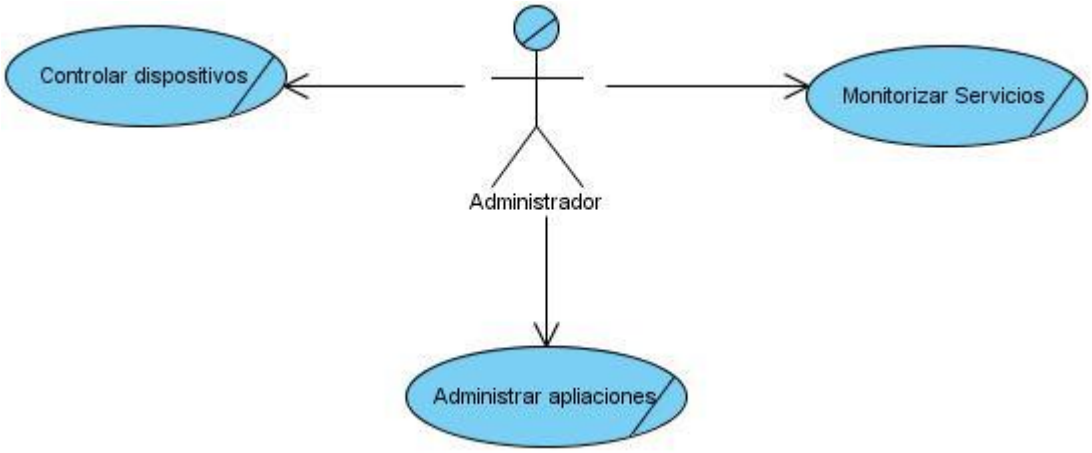

Fig. 1 Modelo de historias de usuario del negocio del módulo a desarrollar.

A continuación se describen las responsabilidades del actor del negocio asociado al Modelo de Historias de Usuario del Negocio del módulo a desarrollar:

- Administrador: Es quien inicializa y para el cual están dirigidas todas las historias de usuario del negocio del módulo a desarrollar.

2.4 Lista de Reserva del Producto (LRP)

En la etapa de planificación, una de las principales actividades definidas por la metodología SXP es la creación de la Lista de Reserva del Producto (LRP). Este artefacto contiene una lista priorizada que define el trabajo a desarrollar durante el

proyecto. El objetivo principal que se persigue durante la etapa de definición del LRP es asegurar que el producto final cumpla en todos los aspectos con las necesidades del cliente. A continuación en la **Tabla # 1** se presenta la lista priorizada contenida en el LRP perteneciente a la solución propuesta.

Tabla # 2 Lista de Reserva del Producto para el módulo a desarrollar.

Prioridad	Ítem*	Descripción	Estimación	Estimado por
Alta				
	1	Invocar funcionalidades remotamente.	18 días	Analista
	2	Mostrar información sobre las funcionalidades que pueden ser invocadas remotamente.	2 días	Analista
	3	Exportar información sobre las conexiones al servidor.	2 días	Analista
	4	Exportar información sobre los servidores en funcionamiento.	1 días	Analista
	5	Exportar información sobre los plugins cargados.	2 días	Analista
	6	Exportar información sobre los filtros cargados.	2 días	Analista
	7	Exportar información sobre el motor en funcionamiento.	1 días	Analista
	8	Exportar información sobre el nodo donde se ejecuta el servidor jWebSocket.	2 días	Analista
	9	Exportar funcionalidades de las aplicaciones desarrolladas con jWebSocket.	20 días	Analista
Media				
	10	Controlar el acceso remoto de los usuarios al módulo.	10 días	Analista
RNF (Requisitos No Funcionales)				
	11	El módulo está dirigido a usuarios con elevados conocimientos de informática y un correcto dominio del negocio		

		en cuestión por lo cual debe estar diseñado para ser robusto y de fácil uso por el cliente.		
	12	Se debe garantizar la protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.		
	13	Se debe realizar el módulo de forma versionable que permita darle mantenimiento a fin de aumentar las funcionalidades y/o corregir los errores del mismo a través de versiones posteriores.		
	14	Para el desarrollo del módulo se han establecido pautas para la codificación que permitan mantener un código uniforme a la hora de realizar modificaciones en el mismo.		
	15	Se documentará el módulo con diferentes manuales con el objetivo de explicar su uso y garantizar de esta forma el soporte del mismo.		
	16	El lenguaje de programación a utilizar para el desarrollo del módulo es Java empleando el Framewok jWebSocket y el IDE NetBeans 7.0.1		
	17	La metodología de desarrollo a seguir es SXP y para la modelación se utilizará la herramienta Visual Paradigm 3.4.		
	18	Todos los textos y mensajes en pantalla aparecerán en idioma inglés.		
	19	Los requisitos mínimos de hardware para el correcto funcionamiento del módulo son: 512 MB de memoria RAM, microprocesador Pentium IV,		

		tarjeta red 100 mbps.		
	20	Para el funcionamiento del módulo es necesario tener instalado un cliente estándar de JMX o un navegador web.		
	21	El servidor de aplicaciones debe tener instalado la Máquina Virtual de Java OpenJDK 7 y el servidor de jWebSocket.		
	22	Los datos del módulo se trasmitirán a través de un canal seguro.		
	23	El código del módulo será liberado bajo la Licencia Pública General Reducida de GNU (LGPL).		
	24	La comunicación entre el cliente y el servidor será a través de los protocolos HTTP y/o RMI indistintamente.		

2.5 Historias de Usuario y Tareas de Ingeniería

En la metodología SXP, las Historias de Usuario son equivalentes a los casos de uso en RUP. Las mismas son definidas por el cliente y describen las tareas que el módulo debe realizar. Las Historias de Usuario constituyen una guía para la construcción de las pruebas de aceptación y son utilizadas además para estimar tiempos de desarrollo. Estas son la base necesaria para el desarrollo eficiente y organizado de un módulo que cumpla con las especificaciones del cliente. A continuación se muestran las Historias de Usuarios definidas para la solución propuesta, así como las tareas de ingeniería asociadas a cada una de estas.

Tabla # 3 Descripción de la HU Invocar funcionalidades remotamente.

Historia de Usuario	
Número: HU_1	**Nombre Historia de Usuario:** Invocar funcionalidades remotamente.
Modificación de Historia de Usuario Número: Ninguna	

Usuario: Lisdey Pérez Hernández	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 4
Riesgo en Desarrollo: Alto	**Puntos Reales:** 4
Descripción: La presente historia de usuario tiene como objetivo invocar de forma remota funcionalidades de las aplicaciones Java que se estén ejecutando en un servidor jWebSocket determinado. El desarrollador de estas aplicaciones es el responsable de determinar las funcionalidades que serán invocadas. Esta historia de usuario a su vez permite mostrar información sobre las funcionalidades que pueden ser invocadas remotamente. Además, brinda la posibilidad de ejecutar estas funcionalidades de forma remota utilizando el protocolo RMI a través de un cliente estándar de JMX o el protocolo HTTP a través de un navegador.	
Observaciones: Para ejecutar estas acciones el usuario debe autenticarse.	
Prototipo de interfaz: Ninguno	

Con el objetivo de desarrollar correctamente la HU# 1 descrita anteriormente, son diseñadas un conjunto de tareas de ingeniería. A continuación se muestran en las Tablas # 4-8 las descripciones de algunas de estas tareas. En el **Anexo # 1** se encuentran las restantes tareas de ingeniería especificadas para esta Historia de Usuario.

Tabla # 4 Descripción de la Tarea de Ingeniería #1.1.

Tarea de Ingeniería	
Número Tarea: 1.1	**Número Historia de Usuario:** HU_1
Nombre Tarea: Investigar sobre cómo crear una infraestructura JMX.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 2 días
Fecha Inicio: 01/11/2011	**Fecha Fin:** 03/11/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Efectuar un estudio sobre la tecnología JMX que permita obtener el conocimiento necesario sobre el proceso de creación de una infraestructura JMX.	

Tabla # 5 Descripción de la Tarea de Ingeniería #1.2.

Tarea de Ingeniería	
Número Tarea: 1.2	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementación de una infraestructura JMX.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 3 días
Fecha Inicio: 03/11/2011	**Fecha Fin:** 08/11/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: A partir de la investigación realizada se sientan las bases para desarrollar una infraestructura JMX que permita exportar todas las funcionalidades de JMXPlugIn utilizando esta tecnología.	

Tabla # 6 Descripción de la Tarea de Ingeniería #1.3.

Tarea de Ingeniería	
Número Tarea: 1.3	**Número Historia de Usuario:** HU_1
Nombre Tarea: Investigar cómo se crea una API en jWebSocket para la comunicación entre plugins.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 2 día
Fecha Inicio: 08/11/2011	**Fecha Fin:** 10/11/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Efectuar un estudio para determinar una forma óptima de desarrollar una API para jWebSocket que permita desde un plugin invocar las funcionalidades de los demás plugins desarrollados.	

Tabla # 7 Descripción de la Tarea de Ingeniería #1.4.

Tarea de Ingeniería	
Número Tarea: 1.4	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementación de una API en jWebSocket para la comunicación entre plugins.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 día
Fecha Inicio:10/11/2011	**Fecha Fin:** 11/11/2011

Programador Responsable: Lisdey Pérez Hernández
Descripción: A partir de la investigación realizada se sientan las bases para desarrollar una API en jWebSocket que permita ejecutar desde JMXPlugIn las funcionalidades de los demás plugins que se están ejecutando en el servidor.

Tabla # 8 Descripción de la Tarea de Ingeniería #1.5.

Tarea de Ingeniería	
Número Tarea: 1.5	**Número Historia de Usuario:** HU_1
Nombre Tarea: Implementación de la funcionalidad para invocar funcionalidades remotamente.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 día
Fecha Inicio: 11/11/2011	**Fecha Fin:** 14/11/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Implementar una funcionalidad que permita invocar remotamente las funcionalidades de los plugins que se están ejecutando en servidor jWebSocket. Esta comunicación se establece utilizando la API desarrollada con este objetivo.	

A continuación en la **Tabla # 9** se muestra la descripción correspondiente a la HU # 2 definida para el módulo a desarrollar.

Tabla # 9 Descripción de la HU Exportar funcionalidades del servidor jWebSocket.

Historia de Usuario	
Número: HU_2	**Nombre Historia de Usuario:** Exportar funcionalidades del servidor jWebSocket.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisdey Pérez Hernández	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2
Descripción: La presente historia de usuario tiene como objetivo facilitar la administración de determinadas funcionalidades del servidor de jWebSocket. Permite mostrar información sobre las conexiones al servidor jWebSocket, así como	

de los plugins y filtros que están cargados en el servidor. Además posibilita exportar características de los servidores que están en funcionamiento, del motor con el cual se ejecutan estos servidores y sobre el nodo donde se ejecuta el servidor jWebSocket. Brinda la posibilidad de administrar estas funcionalidades de forma remota utilizando el protocolo RMI a través de un cliente estándar de JMX o el protocolo HTTP a través de un navegador.
Observaciones: Para ejecutar estas acciones el usuario debe autenticarse.
Prototipo de interfaz: Ninguno

Con el objetivo de desarrollar correctamente la HU# 2 descrita anteriormente, son diseñadas un conjunto de tareas de ingeniería. A continuación se muestran en las Tablas # 10-13 las descripciones de algunas de estas tareas. En el **Anexo # 2** se encuentran las descripciones de las restantes tareas de ingeniería especificadas para esta Historia de Usuario.

Tabla # 10 Descripción de la Tarea de Ingeniería #2.1.

Tarea de Ingeniería	
Número Tarea: 2.1	**Número Historia de Usuario:** HU_2
Nombre Tarea: Investigar los tipos de funcionalidades del servidor jWebSocket que pueden ser exportadas.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 2 días
Fecha Inicio: 29/11/2011	**Fecha Fin:** 30/11/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Efectuar un estudio sobre los tipos de funcionalidades que son comúnmente exportadas utilizando la tecnología JMX por otros servidores. Esta investigación tiene como objetivo identificar los tipos de funcionalidades del servidor jWebSocket que pueden ser exportadas.	

Tabla # 11 Descripción de la Tarea de Ingeniería #2.2.

Tarea de Ingeniería	
Número Tarea: 2.2	**Número Historia de Usuario:** HU_2

Nombre Tarea: Implementación de la funcionalidad para exportar información sobre las conexiones al servidor.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 días
Fecha Inicio: 30/11/2011	**Fecha Fin:** 05/12/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Implementar un *wrapper* para la funcionalidad getAllConnectors del servidor jWebSocket, que permita exportarla utilizando la infraestructura de JMX del módulo. Este *wrapper* permite mostrar la información correspondiente a todas las conexiones existentes con el servidor jWebSocket.	

Tabla # 12 Descripción de la Tarea de Ingeniería #2.3.

Tarea de Ingeniería	
Número Tarea: 2.3	**Número Historia de Usuario:** HU_2
Nombre Tarea: Implementación de la funcionalidad para exportar información sobre los servidores en funcionamiento.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 días
Fecha Inicio: 05/12/2011	**Fecha Fin:** 06/12/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Implementar un *wrapper* para la funcionalidad getServers del servidor jWebSocket, que permita exportarla utilizando la infraestructura de JMX del módulo. Este *wrapper* permite mostrar la información correspondiente a los servidores en funcionamiento en una determinada instancia de jWebSocket.	

Tabla # 13 Descripción de la Tarea de Ingeniería #2.4.

Tarea de Ingeniería	
Número Tarea: 2.4	**Número Historia de Usuario:** HU_2
Nombre Tarea: Implementación de la funcionalidad para exportar información sobre los plugins cargados.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1 días
Fecha Inicio: 06/12/2011	**Fecha Fin:** 07/12/2011
Programador Responsable: Lisdey Pérez Hernández	

Descripción: Implementar un *wrapper* para la funcionalidad getPlugIns del servidor jWebSocket, que permita exportarla utilizando la infraestructura de JMX del módulo. Este *wrapper* permite mostrar la información correspondiente a los plugins que se están cargados en el PlugInChain de jWebSocket.

A continuación en la **Tabla # 14** se muestra la descripción correspondiente a la HU # 3 definida para el módulo a desarrollar.

Tabla # 14 Descripción de la HU Exportar funcionalidades de las aplicaciones desarrolladas con jWebSocket.

Historia de Usuario	
Número: HU_3	**Nombre Historia de Usuario:** Exportar funcionalidades de las aplicaciones desarrolladas con jWebSocket.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisdey Pérez Hernández	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 4
Riesgo en Desarrollo: Medio	**Puntos Reales:** 4
Descripción: La presente historia de usuario tiene como objetivo que los desarrolladores de aplicaciones Java que utilicen el marco de trabajo jWebSocket sean capaces de administrar de forma centralizada determinadas funcionalidades de sus aplicaciones. Con este objetivo provee un mecanismo que permite integrar los plugins o clases desarrollados a la infraestructura de administración del módulo. Brinda además la posibilidad de administrar estas funcionalidades de forma remota utilizando el protocolo RMI a través de un cliente estándar de JMX o el protocolo HTTP a través de un navegador.	
Observaciones: Para ejecutar estas acciones el usuario debe autenticarse.	
Prototipo de interfaz: Ninguno	

Con el objetivo de desarrollar correctamente la HU# 3 descrita anteriormente, son diseñadas un conjunto de tareas de ingeniería. A continuación se muestran en las Tablas # 15-18 las descripciones de algunas de estas tareas. En el **Anexo # 3** se encuentran las

descripciones de las restantes tareas de ingeniería especificadas para esta Historia de Usuario.

Tabla # 15 Descripción de la Tarea de Ingeniería #3.1.

Tarea de Ingeniería	
Número Tarea: 3.1	**Número Historia de Usuario:** HU_3
Nombre Tarea: Investigar sobre el mecanismo existente en Spring para exportar clases.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 3 días
Fecha Inicio: 14/12/2011	**Fecha Fin:** 19/12/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Efectuar un estudio sobre el mecanismo utilizado por el marco de trabajo Spring para la exportación de clases. Esta investigación tiene como objetivo caracterizar este mecanismo para ser capaz de modificarlo de acorde a las necesidades del módulo a desarrollar.	

Tabla # 16 Descripción de la Tarea de Ingeniería #3.2.

Tarea de Ingeniería	
Número Tarea: 3.2	**Número Historia de Usuario:** HU_3
Nombre Tarea: Implementación de clases que permitan definir los ficheros de configuración del módulo.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 4 días
Fecha Inicio: 19/12/2011	**Fecha Fin:** 22/12/2011
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Implementar un conjunto de clases que permitan definir los ficheros de configuración que contendrán las clases y plugins a exportar. Este conjunto de clases es parte esencial del nuevo mecanismo de exportación del módulo.	

Tabla # 17 Descripción de la Tarea de Ingeniería #3.3.

Tarea de Ingeniería	
Número Tarea: 3.3	**Número Historia de Usuario:** HU_3

Nombre Tarea: Implementación de un mecanismo para definir las descripciones y nombres de los recursos a exportar.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 3 días
Fecha Inicio: 10/01/2012	**Fecha Fin:** 13/01/2012
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Implementar un conjunto de clases que permitan personalizar en los recursos a exportar las descripciones de las clases, los atributos y las operaciones, así como los nombres de los parámetros.	

Tabla # 18 Descripción de la Tarea de Ingeniería #3.4.

Tarea de Ingeniería	
Número Tarea: 3.4	**Número Historia de Usuario:** HU_3
Nombre Tarea: Implementación de un mecanismo que permita el envío de notificaciones.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 días
Fecha Inicio: 13/01/2012	**Fecha Fin:** 17/01/2012
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Implementar un conjunto de clases que permitan enviar notificaciones dadas determinadas circunstancias en los recursos exportados. En este caso las notificaciones serán enviadas al cambiar el valor de un atributo, así como antes y después de ejecutar una operación.	

A continuación en la **Tabla # 19** se muestra la descripción correspondiente a la HU # 4 definida para el módulo a desarrollar.

Tabla # 19 Descripción de la HU Autenticar Usuario.

Historia de Usuario	
Número: HU_4	**Nombre Historia de Usuario:** Autenticar usuario.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lisdey Pérez Hernández	**Iteración Asignada:** 3

Prioridad en Negocio: Media	**Puntos Estimados:** 2
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2
Descripción: La presente historia de usuario tiene como objetivo permitir la autenticación de los usuarios para el acceso remoto al módulo.	
Observaciones:	
Prototipo de interfaz: Ninguno	

Con el objetivo de desarrollar correctamente la HU# 4 descrita anteriormente, son diseñadas un conjunto de tareas de ingeniería. A continuación se muestran en las Tablas # 20-21 las descripciones de algunas de estas tareas. En el **Anexo # 4** se encuentran las descripciones de las restantes tareas de ingeniería especificadas para esta Historia de Usuario.

Tabla # 20 Descripción de la Tarea de Ingeniería #4.1.

Tarea de Ingeniería	
Número Tarea: 4.1	**Número Historia de Usuario:** HU_4
Nombre Tarea: Investigación sobre los mecanismo de autenticación utilizados para la tecnología JMX.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 2 días
Fecha Inicio: 27/01/2012	**Fecha Fin:** 31/01/2012
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Efectuar un estudio sobre los diferentes mecanismo de autenticación utilizados para JMX con el objetivo de seleccionar el más adecuado teniendo en cuenta las características del módulo.	

Tabla # 21 Descripción de la Tarea de Ingeniería #4.2.

Tarea de Ingeniería	
Número Tarea: 4.2	**Número Historia de Usuario:** HU_4
Nombre Tarea: Implementación del mecanismo de autenticación para el acceso remoto utilizando el protocolo RMI.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 2 días

Fecha Inicio: 31/01/2012	**Fecha Fin:** 02/02/2012
Programador Responsable: Lisdey Pérez Hernández	
Descripción: Implementar las clases que permitan la autenticación para el acceso remoto utilizando el protocolo RMI. El método de autenticación a implementar será el seleccionado durante la investigación como el más adecuado para el módulo.	

2.6 Plan de Release

El artefacto Plan de Release define las iteraciones a realizar durante el desarrollo de la solución propuesta, de esta forma se especifican las entregas finales e intermedias. Este artefacto tiene como entrada el conjunto de Historias de Usuario definidas anteriormente, de las cuales se tiene en cuenta la prioridad definida por el usuario para establecer el orden de su implementación. Como resultado de priorizar las Historias de Usuario definidas para el desarrollo del módulo propuesto se obtiene la planificación que se muestra en la **Tabla # 22**.

Tabla # 22 Plan de Release.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
Iteración 2	En esta iteración se desarrollarán las historias de usuario con categoría alta. Esta es la iteración principal del módulo, en la cual se genera una primera versión estable del producto.	HU_1, HU_2, HU_3	10 semanas
Iteración 3	En esta iteración se desarrollará la historia de usuario con categoría media integrándola con las historias de usuario implementadas. El objetivo principal de esta iteración es adicionar los mecanismos de seguridad al módulo.	HU_4	2 semanas

2.7 Arquitectura de Software

La arquitectura de software es considerada la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, así como los principios que orientan su diseño y evolución. El módulo JMXPlugIn está basado en la tecnología nativa de Java, JMX. Por lo tanto, implementa la arquitectura especificada para esta tecnología, la cual está dividida en tres capas principales: Instrumentación, Agente y Servicios Distribuidos.

En correspondencia con esta arquitectura, el módulo a desarrollar cuenta con una capa de Instrumentación la cual contiene los componentes que encapsulan los objetos que serán administrados por el módulo, es decir agrupa los llamados MBeans u objetos manejables. Por otra parte, JMXPlugIn presenta una capa de Agente en la cual están agrupados todos los componentes que permiten controlar los MBeans pertenecientes a la capa de Instrumentación y hacerlos disponibles para su acceso remoto a través de los conectores y adaptadores de la capa de Servicios Distribuidos. Esta es la capa principal del módulo, la cual permite crear la infraestructura JMX para el servidor jWebSocket. Por último, la capa de Servicios Distribuidos contiene los elementos que posibilitan el acceso remoto a las funcionalidades del módulo. En este caso en específico, el acceso se realiza a través del conector RMI o del adaptador HTTP. Además, es la capa que se encarga de la seguridad del módulo. Es importante resaltar que esta capa contiene además las aplicaciones clientes compatibles con JMX que son usadas para el acceso al módulo. Sin embargo, estas no forman parte del alcance de la presente investigación, por lo cual no están reflejadas en la estructura de JMXPlugIn. A continuación se muestra en la **Fig. 2** el diagrama que representa la arquitectura del módulo JMXPlugIn descrita anteriormente.

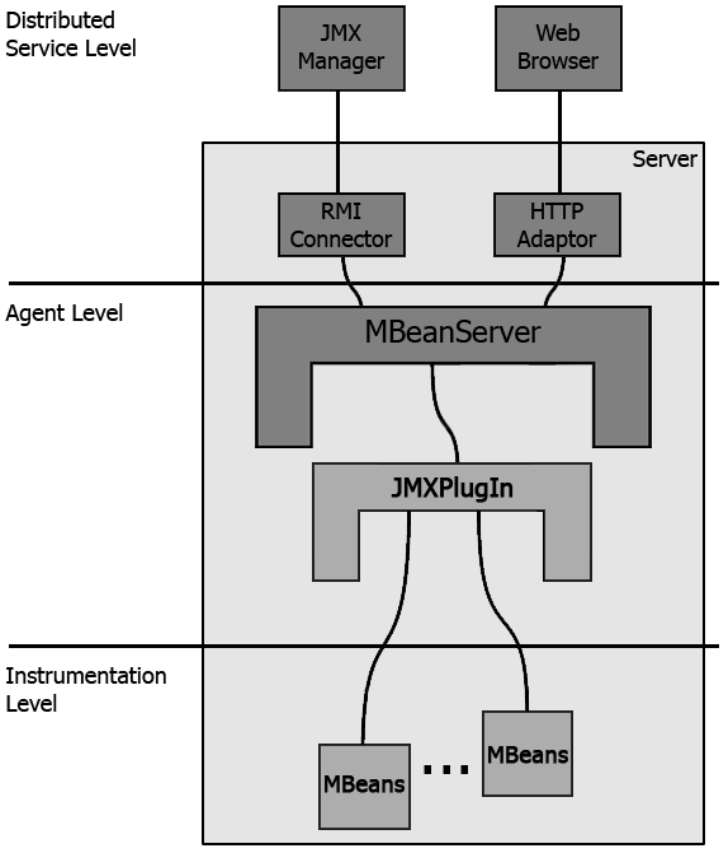

Fig. 2 Arquitectura de software del módulo JMXPlugIn.

2.8 Diseño con Metáforas

La metodología SXP asume las fortalezas tanto de SCRUM como de XP. Esta última metodología define el término metáforas, el cual es considerado una historia compartida que describe cómo debería funcionar el sistema. La práctica de las metáforas consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. (FOWLER, 2004)

El Diseño con Metáforas constituye el diseño más simple de la solución propuesta que pueda ser implementado en un momento dado del proyecto. El mismo genera el artefacto Modelo de Diseño, el cual contiene el diagrama de paquetes del módulo a desarrollar. A continuación en la **Fig. 3** se muestra el diagrama de paquetes perteneciente al módulo propuesto así como una descripción general del mismo.

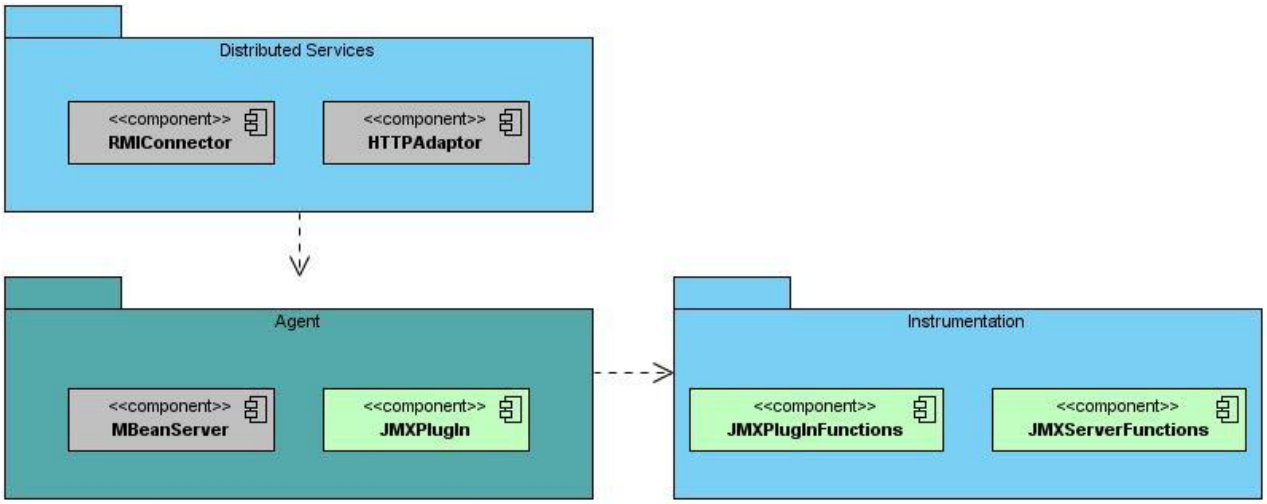

Fig. 3 Diagrama de paquetes del módulo a desarrollar.

La **capa Distributed Services** contiene los componentes RMIConnector y HTTPAdaptor los cuales permiten el acceso remoto al módulo a través de los protocolos RMI y HTTP respectivamente. Esta capa se comunica directamente con la **capa Agent,** la cual agrupa un conjunto de componentes encargados de controlar los recursos contenidos en la capa Instrumentation**,** haciéndolos disponibles para su acceso remoto a través de la capa Distributed Services. Por último, la **capa Instrumentation** contiene los componentes que agrupan las funcionalidades principales que serán gestionadas en el módulo a desarrollar.

2.9 Diagrama de Componentes

El diagrama de componentes describe los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de software incluidos en el desarrollo de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, entre otros. De esta forma se tiene un mejor entendimiento por parte de los integrantes del proyecto de los elementos a tener en cuenta durante el desarrollo de la solución propuesta. A continuación en la **Fig. 4** se presenta el diagrama de componentes correspondiente al módulo a desarrollar.

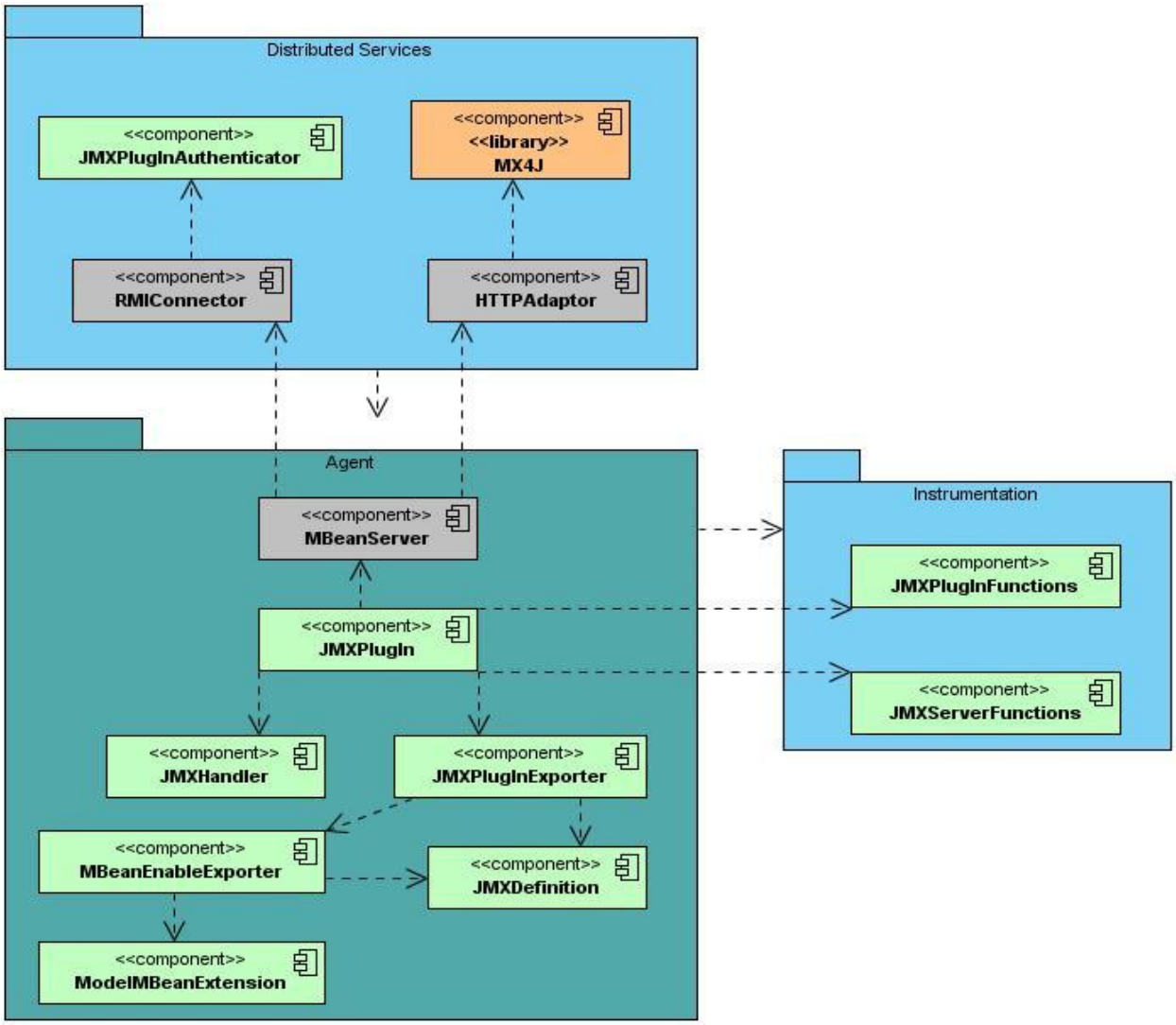

Fig. 4 Diagrama de componentes del módulo a desarrollar.

Para una mejor comprensión del diagrama de componentes del módulo a desarrollar, se realiza a continuación una descripción general de cada uno de estos:

- **RMIConnector:** permite el acceso remoto al módulo a través del protocolo RMI, utiliza el componente **JMXPlugInAuthenticator** para garantizar la seguridad del mismo.
- **HTTPAdaptor:** permite el acceso remoto al módulo a través del protocolo HTTP, utiliza la librería **MX4J** con este objetivo.
- **MBeanServer:** se encarga de controlar los recursos que serán gestionados en el módulo y hacerlos disponibles para su acceso remoto.

- **JMXPlugIn:** es la clase principal del módulo que interactúa con todos los demás componentes que lo integran.
- **JMXPlugInExporter:** clase principal del mecanismo que permite integrar los plugins y clases desarrolladas con el marco de trabajo jWebSocket a la infraestructura JMX del módulo. Se encarga de leer todos los ficheros de configuración creados con este propósito convertirlos a objetos e integrarlos con el mecanismo de exportación de Spring.
- **JMXHandler:** permite convertir dinámicamente del tipo de datos *Map*, al tipo de datos *CompositeData*, el cual es utilizado por la tecnología JMX para el trabajo con tipos de datos complejos es decir, objetos, ya que por defecto, esta tecnología solo es capaz de manejar los tipos de datos simples.
- **JMXPlugInFunctions:** permite invocar las funcionalidades de las aplicaciones que se están ejecutando en el servidor jWebSocket, a través de la infraestructura JMX creada.
- **JMXServerFunctions:** es la clase encargada de encapsular determinadas funcionalidades del servidor jWebSocket permitiendo gestionarlas a través de la infraestructura JMX creada.

Conclusiones Parciales

En el presente capítulo fueron definidas las características y principales funcionalidades del módulo JMXPlugIn. De esta forma queda especificada la propuesta de solución del presente trabajo de diploma para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket. Además, se presentan los Requisitos Funcionales y No Funcionales necesarios para la obtención de un módulo eficiente. Se describieron las Historias de Usuario y sus correspondientes tareas de ingeniería, así como el plan de *release* del proyecto definiéndose claramente el cronograma de trabajo y las tareas que el módulo debe realizar. Se realizó además, el modelado del diagrama de paquetes y de componentes que permiten un mejor entendimiento de la estructura del módulo a desarrollar.

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

Introducción

En el presente capítulo se describe la etapa de implementación de la solución propuesta en el capítulo anterior, modelándose el diagrama de despliegue del módulo desarrollado. Se establece una estrategia de validación para certificar la capacidad y potencialidad del módulo desarrollado para la administración remota de aplicaciones distribuidas. Además, se exponen los resultados alcanzados hasta el momento y el aporte tanto social como económico del módulo desarrollado.

3.1 Diagrama de Despliegue

En el diagrama de despliegue se modela la arquitectura en tiempo de ejecución del módulo, indicándose la situación física de los componentes lógicos desarrollados. A continuación en la **Fig. 5** se presenta el diagrama de despliegue que representa la distribución física del módulo desarrollado. Además se realiza una breve descripción de cada uno de los nodos que integran este diagrama.

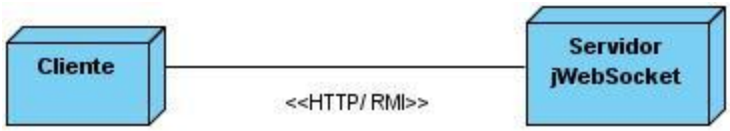

Fig. 5 Diagrama de despliegue del módulo desarrollado.

La conexión del cliente con el servidor de jWebSocket puede realizarse a través de los protocolos HTTP o RMI indistintamente. Para la conexión con HTTP el cliente solo necesitará un navegador web sobre cualquier sistema operativo. En el caso de la conexión a través del protocolo RMI, el cliente debe contar con una aplicación de administración compatible con JMX.

3.2 Validación de la Solución

Para realizar la validación de la presente investigación, la cual tiene como objetivo desarrollar un módulo para la administración remota de las aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket usando JMX, se decide trazar una estrategia de validación basada en las siguientes etapas. La primera etapa consiste en diseñar y ejecutar un conjunto de casos de pruebas funcionales a las Historias de

Usuario definidas para el módulo desarrollado con el objetivo de verificar su correcto funcionamiento. La siguiente etapa consiste en un proceso de certificación de la calidad realizado por el grupo de calidad del Centro de Desarrollo de la Facultad Regional "Mártires de Artemisa". De esta forma es posible validar la funcionalidad, estandarización y limpieza del código así como la calidad de los artefactos documentales generados teniendo en cuenta la metodología de desarrollo SXP seleccionada en la presente investigación.

Por último, se somete el módulo desarrollado a una valoración por parte de Alexander Schulze, líder de la comunidad de jWebSocket internacional y arquitecto principal del proyecto. En esta etapa de validación se certifica la capacidad de integración de la solución desarrollada con el marco de trabajo jWebSocket a nivel internacional, así como su usabilidad real por los clientes potenciales y desarrolladores de esta comunidad. A continuación se realiza una descripción más detallada de cada una de las etapas de la estrategia de validación trazada a las cuales fue sometido el módulo desarrollado.

3.2.1. Casos de Pruebas Funcionales

Los casos de pruebas funcionales son definidos teniendo en cuenta las Historias de Usuario especificadas para el desarrollo de la solución propuesta. Su utilización permite medir principalmente la calidad del trabajo realizado así como garantizar la entrega de un producto eficiente y en correspondencia con las necesidades del cliente. Durante el desarrollo del módulo JMXPlugIn se diseñaron un conjunto de pruebas funcionales con el objetivo de comprobar el correcto funcionamiento del mismo. A continuación en las Tablas # 23-27 se muestran las descripciones de algunos de los casos de prueba más relevantes realizados a cada una de las Historias de Usuario con las que cuenta el módulo desarrollado. En el **Anexo # 5** se encuentran los restantes casos de prueba realizados a las funcionalidades desarrolladas.

Tabla # 23 Descripción del caso de prueba #3 para la HU #1.

Caso de Prueba Funcional	
Código Caso de Prueba: JWS-01-3	**Nombre Historia de Usuario:** Invocar funcionalidades remotamente.

Nombre de la persona que realiza la prueba: Lisdey Pérez Hernández
Descripción de la Prueba: Esta prueba consiste en invocar una funcionalidad determinada de una aplicación que esté cargada en el servidor jWebSocket. En este caso serán ejecutadas las siguientes pruebas: Se introducen los datos correctos para comprobar que el módulo es capaz de invocar remotamente funcionalidades. Se introducen los datos incorrectos para verificar que la operación está validada correctamente.
Condiciones de Ejecución: El usuario debe acceder al módulo ya sea por el protocolo HTTP o RMI y autenticarse correctamente. Además el usuario debe tener conocimiento de los servidores jWebScoket que están en funcionamiento, de los plugins que están cargados y de los métodos que pueden ser invocados (toda esta información se encuentra en el atributo *InformationOfRunningServers*).
Entrada / Pasos de ejecución: Se accede a la sección de *MBeans*, específicamente se selecciona el *MBeanServer* llamado *jWebSocketServer* correspondiente al módulo. Se selecciona el *MBean* llamado *JMXPlugInFunctions*, se accede a la sección de operaciones pertenecientes a este *MBean*. Se introduce el id del servidor en el cual se está ejecutando el plugin deseado, se introduce el id del plugin deseado, se especifica el nombre del método que se desea invocar y se introducen los parámetros necesarios para ejecutar dicho método en caso de que sea necesario. Por último se presiona el botón correspondiente a la operación InvokePlugInOperations.
Resultado Esperado: Se muestran los resultados de invocar la operación en caso de que los parámetros sean correctos y que la operación especificada esté disponible para su acceso desde el módulo. En el caso contrario se lanza una excepción.
Evaluación de la Prueba: Satisfactoria

Tabla # 24 Descripción del caso de prueba #1 para la HU #2.

Caso de Prueba Funcional

Código Caso de Prueba: JWS-02-1	**Nombre Historia de Usuario:** Exportar funcionalidades del servidor jWebSocket.
Nombre de la persona que realiza la prueba: Lisdey Pérez Hernández	
Descripción de la Prueba: Esta prueba consiste en mostrar información sobre todas las conexiones de un servidor jWebSocket determinado. En este caso serán ejecutadas las siguientes pruebas: Se introducen los datos correctos para comprobar que el módulo es capaz mostrar las conexiones de un servidor dado. Se introducen los datos incorrectos para verificar que la operación está validada correctamente.	
Condiciones de Ejecución: El usuario debe acceder al módulo ya sea por el protocolo HTTP o RMI y autenticarse correctamente. Además, el usuario debe tener conocimiento de los servidores jWebSocket que están en funcionamiento en ese momento.	
Entrada / Pasos de ejecución: Se accede a la sección de *MBeans*, específicamente se selecciona el *MBeanServer* llamado *jWebSocketServer* correspondiente al módulo. Se selecciona el *MBean* llamado *JMXServerFunctions*, se accede a la sección de operaciones y se selecciona la operación *getAllConnectors*. Se introduce el id del servidor del cual se desea conocer sus conexiones y se presiona el botón correspondiente a dicha operación.	
Resultado Esperado: Se muestra información sobre las conexiones del servidor especificado, en caso de que no sea especificado correctamente el parámetro requerido para ejecutar esta operación se lanza una excepción.	
Evaluación de la Prueba: Satisfactoria	

Tabla # 25 Descripción del caso de prueba #1 para la HU #3.

Caso de Prueba Funcional	
Código Caso de Prueba: JWS-03-1	**Nombre Historia de Usuario:** Exportar funcionalidades de las aplicaciones desarrolladas con

	jWebSocket.
Nombre de la persona que realiza la prueba: Lisdey Pérez Hernández	
Descripción de la Prueba: Esta prueba consiste en adicionar una clase desarrollada con jWebSocket a la infraestructura JMX del módulo, a través del fichero de configuración creado con este propósito.	
Condiciones de Ejecución: El usuario debe crear un fichero de configuración siguiendo las normas establecidas con este objetivo.	
Entrada / Pasos de ejecución: Se debe crear un fichero de configuración en el cual se especifica los atributos, operaciones y constructores de la clase que se desea integrar a la infraestructura JMX. Además es posible definir notificaciones de eventos que serían lanzados cuando cambia el valor de un atributo, antes de ejecutar una operación o luego de ejecutarla. Para comprobar que la clase ha sido integrada correctamente, el usuario debe acceder al módulo ya sea por el protocolo HTTP o RMI y autenticarse correctamente. A continuación debe acceder a la sección de *MBeans*, específicamente se debe seleccionar el *MBeanServer* llamado *jWebSocketServer* correspondiente al módulo en cuestión. Por último se selecciona el nuevo *MBean* agregado y se comprueba que contiene todos los elementos que fueron definidos en el fichero de configuración correspondiente.	
Resultado Esperado: Si la clase ha sido integrada con éxito se muestra un nuevo *MBean* con el nombre especificado en el fichero de configuración.	
Evaluación de la Prueba: Satisfactoria	

Tabla # 26 Descripción del caso de prueba #2 para la HU #4.

Caso de Prueba Funcional	
Código Caso de Prueba: JWS-04-2	**Nombre Historia de Usuario:** Autenticar usuario.
Nombre de la persona que realiza la prueba: Lisdey Pérez Hernández	
Descripción de la Prueba: Esta prueba consiste en conectarse remotamente al módulo a través del protocolo RMI. En este caso serán realizadas dos pruebas:	

Se introducirá el usuario y la contraseña correcta para comprobar que es posible el acceso al módulo con las credenciales adecuadas. Se introducirá el usuario y la contraseña incorrecta para comprobar que no se pueden conectar usuarios no deseados.
Condiciones de Ejecución: Se debe tener instalada una aplicación de administración compatible con JMX.
Entrada / Pasos de ejecución: Se selecciona la pestaña de conexión remota de la ventana inicial de la aplicación de administración compatible con JMX. Se introduce la URL correspondiente al servicio RMI del módulo. Se introduce el usuario y la contraseña correspondiente. Por último, se presiona el botón Conectar para acceder al módulo.
Resultado Esperado: Acceso remoto al módulo permitido a través del protocolo RMI, especificando el usuario y la contraseña correcta.
Evaluación de la Prueba: Satisfactoria

Tabla # 27 Descripción del caso de prueba #3 para la HU #4.

Caso de Prueba Funcional	
Código Caso de Prueba: JWS-04-3	**Nombre Historia de Usuario:** Autenticar usuario.
Nombre de la persona que realiza la prueba: Lisdey Pérez Hernández	
Descripción de la Prueba: Esta prueba consiste en conectarse remotamente al módulo a través del protocolo HTTP. En este caso serán realizadas dos pruebas: Se introducirá el usuario y la contraseña correcta para comprobar que es posible el acceso al módulo con las credenciales adecuadas. Se introducirá el usuario y la contraseña incorrecta para comprobar que no se pueden conectar usuarios no deseados.	
Condiciones de Ejecución: Se debe tener instalado un navegador web.	
Entrada / Pasos de ejecución: Se introduce la URL con el dominio en el cual se está ejecutando el módulo. Se introduce el usuario y la contraseña correspondiente.	

Por último, se presiona el botón Aceptar para acceder al módulo.
Resultado Esperado: Acceso remoto al módulo permitido a través del protocolo HTTP, especificando el usuario y la contraseña correcta.
Evaluación de la Prueba: Satisfactoria

3.2.2. Certificación de Calidad de Software

El grupo de calidad del Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa” es el encargado de realizar el proceso de certificación de calidad tanto del código fuente del módulo desarrollado como de la documentación generada teniendo en cuenta la metodología de desarrollo SXP seleccionada en la presente investigación. La revisión del código fuente es llevada a cabo por el Ing. Domma Moreno Dager, Asesor de Tecnología del Centro. Durante esta revisión se evalúa la estandarización del código fuente basándose principalmente en la Plantilla Estándar de Código generada con este propósito. El hecho de que el módulo desarrollado cumpla estrictamente con los estándares definidos para el proyecto facilita el mantenimiento posterior del código, garantizando a su vez la calidad y limpieza del mismo. Por otra parte, durante el proceso de revisión del código fuente es analizado el funcionamiento del módulo desarrollado teniendo en cuenta los requisitos funcionales definidos por el cliente. Con este objetivo se ejecutan un conjunto de pruebas funcionales basadas en las Historias de Usuarios definidas para este módulo. Estas pruebas funcionales permiten verificar la calidad y el correcto funcionamiento de la solución.

El proceso de revisión de la documentación generada para el módulo desarrollado es llevado a cabo por la Ing. Maidel Ojeda Castro, Asesora de Calidad del Centro de Desarrollo. La documentación a evaluar está compuesta por los siguientes artefactos:

- Plantilla Modelo de Historia de Usuario del Negocio
- Plantilla lista de reserva del producto (LRP)
- Plantilla de Historia de Usuario
- Plantilla de Arquitectura de Software SXP
- Plantilla Tarea de Ingeniería

- Plantilla de Releases
- Plantilla Estándar de Código
- Plantilla Caso de Prueba de Aceptación
- Plantilla Manual de Usuario
- Plantilla Manual de Administración
- Plantilla Manual de Desarrollo

Estos artefactos se generan teniendo en cuenta la metodología SXP y describen los principales aspectos del proceso de desarrollo de software del módulo. Durante el proceso de revisión de esta documentación se evalúa el cumplimiento con las plantillas definidas por la metodología SXP para cada uno de estos artefactos. Además se verifica la claridad en la redacción de toda la documentación de forma tal que facilite el entendimiento de la misma por parte de otros desarrolladores o usuarios. A su vez estos artefactos deben cumplir con los estándares de calidad establecidos por la Universidad de la Ciencias Informáticas para garantizar de esta forma la continuidad del proyecto.

3.2.3. Valoración del cliente

El módulo desarrollado en la presente investigación fue sometido al criterio de Alexander Schulze, fundador y principal arquitecto del proyecto jWebSocket, así como líder de la comunidad internacional del mismo. Para su evaluación se almacena en el repositorio internacional de jWebSocket el código fuente del módulo desarrollado y una documentación en inglés que posibilita un mayor entendimiento de la solución. El análisis del código fuente tiene como objetivo principal verificar la capacidad de integración del módulo con el marco de trabajo jWebSocket a nivel internacional. Con este fin se evalúa la limpieza del código fuente y los estándares de codificación establecidos para este marco de trabajo. De esta forma se garantizan la calidad de la solución así como su correcto funcionamiento en correspondencia con jWebSocket. A su vez, se evalúa la usabilidad real que presenta el módulo desarrollado para la administración remota de aplicaciones distribuidas tanto para clientes potenciales como para desarrolladores de esta comunidad.

La documentación en inglés asociada al módulo desarrollado está compuesta por los manuales de Usuario, Desarrollador y Administración. Esta documentación no solo es utilizada para lograr un mayor entendimiento de la solución sino que a su vez es evaluada exhaustivamente con el objetivo de garantizar su calidad. De esta forma estos manuales pueden ser usados para facilitar el soporte del módulo desarrollado una vez que este sea liberado e integrado al marco de trabajo jWebSocket.

3.3 Resultados Obtenidos

El proceso de validación al cual fue sometido el módulo desarrollado arrojó los siguientes resultados:

- Los casos de pruebas funcionales realizados fueron satisfactorios, garantizando el correcto funcionamiento del módulo desarrollado y el cumplimiento de los requisitos funcionales definidos por el cliente.
- El grupo de calidad del Centro de Desarrollo de la Facultad Regional “Mártires de Artemisa” emitió un aval que certifica la funcionalidad y estandarización del módulo desarrollado, así como la calidad de la documentación que permite a la solución ser usable y generalizada para otros usuarios. (Ver **Anexo # 6**)
- La valoración por parte de Alexander Schulze certificó satisfactoriamente el módulo desarrollado, asegurando de esta forma la correcta integración de la solución al marco de trabajo jWebSocket a nivel internacional, así como su usabilidad real por los clientes potenciales y desarrolladores de esta comunidad. (Ver **Anexo # 7**)

Teniendo en cuenta los resultados satisfactorios de cada una de las etapas de la estrategia de validación asumida en la presente investigación, queda disponible la versión 1.0 del módulo JMXPlugIn. De esta forma se obtiene una solución para la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket que garantiza un posible un aumento en los niveles de disponibilidad y confiabilidad de los servicios.

3.4 Funcionalidades Obtenidas

Las principales funcionalidades que posee el módulo JMXPlugIn en su versión 1.0 con:

- Invocar remotamente las funcionalidades de las aplicaciones que se estén ejecutando en un servidor jWebSocket determinado.
- Mostrar información sobre las funcionalidades que pueden ser invocadas remotamente.
- Administrar remotamente determinadas funcionalidades del servidor jWebSocket tales como:
    - Mostrar información sobre las conexiones existentes con el servidor.
    - Mostrar información sobre los plugins que se están ejecutando en un servidor jWebSocket determinado.
    - Mostrar información sobre los filtros cargados en un servidor jWebSocket determinado.
    - Exportar las características de los servidores jWebSocket en funcionamiento.
    - Exportar las características del motor con el cual se ejecuta el servidor jWebSocket.
    - Exportar las características del nodo donde se ejecuta el servidor jWebSocket.
- Integrar los plugins y clases desarrollados con jWebSocket a la infraestructura JMX creada.
- Acceder remotamente a todas las funcionalidades del módulo a través de los protocolos RMI y HTTP.
- Controlar el acceso remoto de los usuarios al módulo.

3.5 Aporte Social y Económico

El módulo JMXPlugIn permite la administración remota de aplicaciones distribuidas desarrollados con el marco de trabajo jWebSocket. JMXPlugIn provee a los administradores con una herramienta flexible y eficiente que permite mantener un control y monitorización constante de los recursos de los servidores jWebSocket. Además, permite la gestión centralizada de las aplicaciones desarrolladas con este marco de trabajo, simplificando la complejidad de la administración de las mismas. De esta forma,

se hace mucho más sencillo el trabajo de los administradores al contar con un módulo que permite el manejo centralizado las aplicaciones que se están ejecutando.

Por otra parte, el uso de JMXPlugIn como herramienta de administración remota para jWebSocket posibilita que las empresas que utilizan este marco de trabajo y/o servidor de aplicaciones cuenten con un control eficiente y centralizado de sus recursos. En la mayoría de los casos, esto garantiza la disponibilidad de los servicios en general, aumentando significativamente la confiabilidad de los usuarios en dichas aplicaciones. Este hecho influye positivamente en la economía de estas empresas, atrayendo a una mayor cantidad de usuarios debido a la estabilidad y eficiencia de los servicios brindados. Por otra parte, al ser mucho más simple el proceso de administración de los servidores jWebSocket se hace posible la reducción de los costos destinados a este proceso en dichas empresas.

Conclusiones Parciales

En el presente capítulo fue definida la situación física de los componentes lógicos desarrollados, modelándose con este objetivo el diagrama de despliegue del módulo en cuestión. Además, fue establecida una estrategia de validación que permitió certificar el correcto funcionamiento del módulo desarrollado, demostrando que sus funcionalidades satisfacen las necesidades del cliente y concuerdan con los requisitos definidos en la etapa inicial del proyecto. Como resultado final de la presente investigación se obtuvo la versión 1.0 del módulo JMXPlugIn el cual permite la administración remota de aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket.

CONCLUSIONES GENERALES

Con la realización del presente trabajo de diploma se dio respuesta a cada una de las preguntas científicas planteadas, destacando de manera general las siguientes conclusiones:

- Se realizó la fundamentación teórico-metodológica de la investigación, permitiendo conceptualizar el proceso de administración remota de aplicaciones distribuidas, el cual comprende todas las medidas necesarias para garantizar el funcionamiento eficaz y eficiente de un sistema en red.
- Se realizó un estudio de las soluciones actuales que utilizan la tecnología JMX permitiendo identificar una tendencia a crear una infraestructura de administración teniendo en cuenta las características propias de cada proyecto.
- El módulo JMXPlugIn desarrollado permite la administración remota de las aplicaciones distribuidas desarrolladas con el marco de trabajo jWebSocket y a su vez garantiza un posible aumento de los niveles de disponibilidad y confiabilidad de los servicios.
- Se comprobó la capacidad y potencialidad del módulo JMXPlugIn para la administración remota de aplicaciones distribuidas a través de la estrategia de validación trazada en la presente investigación, garantizando que su código fuente y documentación poseen la calidad requerida, además puede ser integrada al entorno de desarrollo de jWebSocket y utilizada por su comunidad.

RECOMENDACIONES

Se recomienda para versiones posteriores del módulo JMXPlugIn:

- Modificar la librería MX4J utilizada para el acceso remoto al módulo a través del protocolo HTTP con el objetivo de mejorar los mecanismos de autenticación, así como las vistas utilizadas para mostrar los tipos de datos complejos (*CompositeData*).

- Aumentar el número de funcionalidades del servidor de jWebSocket que puedan ser administradas remotamente a través del módulo.

BIBLIOGRAFÍA REFERENCIADA

Apache Tomcat 7. 2012. *The Apache Software Foundation.* [En línea] 2012. [Citado el: 21 de 2 de 2012.] http://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html#Using_the_JMX_Proxy_Servlet.

BALTER, R, y otros. 2009. *Architecturing and configuring distributed application with Olan.* London : Springer-Verlag London, 2009. ISBN:1-85233-088-0.

BECK, Kent. 2002. *Una explicación de la programación extrema: Aceptar el cambio.* Madrid : Addison-Wesley, 2002. 9788478290550.

BOWKER, Todd. 2001. *JavaWorld.* [En línea] Infoworld, 8 de Junio de 2001. http://www.javaworld.com/javaworld/jw-06-2001/jw-0608-jmx.html.

CABALLÉ, Santi y XHAFA, Fatos. 2007. *Aplicaciones Distribuidas en Java con tecnología RMI.* s.l. : Delta Publicaciones Universitarias, S.L., 2007. ISBN-8496477959.

CHAMBERS, Lloyd. 2006. *GlassFish Community.* [En línea] 1 de 8 de 2006. [Citado el: 21 de 2 de 2012.] http://glassfish.java.net/javaee5/amx/docs/amx.html.

CHIAVENATO, Idalberto. 2004. *Introducción a la Teoría General de la Administración .* s.l. : McGraw-Hill 7ma Edición, 2004. ISBN-9701055004 .

Eclipse. 2011. *The Eclipse Foundation Open Source Community Website.* [En línea] Eclipse Foundation, 2011. http://www.eclipse.org/.

Eclipse Foundation. 2012. *Eclipse Wiki.* [En línea] Eclipse Foundation, 2012. [Citado el: 16 de 2 de 2012.] http://wiki.eclipse.org/Jetty.

FIGUEROA, Roberth, SOLÍS, Camilo y CABRERA, Armando. 2011. *Metodologías tradicionales vs. Metodologías ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación, 2011.

FOWLER, Martin. 2004. Is Design Dead? [En línea] 5 de 2004. [Citado el: 27 de 3 de 2012.] http://www.martinfowler.com/articles/designDead.html.

GOSLING, James, y otros. 2005. *The Java language specification.* s.l. : Addison Wesley, 2005. 3ra Edition.

GUI, Ning. 2011. *Middleware-based adaption evolution with reusable adaption components.* s.l. : University of Antwerp, 2011.

IBM. 2012. IBM-Rational Rose Enterprise. [En línea] IBM. [Citado el: 15 de 2 de 2012.] http://www-142.ibm.com/software/products/es/es/enterprise/.

JACOBSON, Ivar, GRADY, Booch y RUMBAUGH, James. 2000. *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2000. 84-7829-036-2.

Java Community Process. 2012. *Oracle Sun Developer Network (SDN).* [En línea] Sun MicroSystem, 2012. [Citado el: 20 de 2 de 2012.] http://www.oracle.com/technetwork/java/javase/tech/jmxadoption-jsp-142249.html.

JBoss Community . 2008. *JBoss Application Server 5.0.0 Administration And Development Guide.* s.l. : JBoss Community , 2008.

JOHNSON, Rod, HOELLER, Juergen y DONALD, Keith. 2010. *Spring Reference Documentation 3.0.* 2010.

Kaazing Corporation. 2012. *Kaazing WebSocket Gateway.* [En línea] Kaazing Corporation, 2012. [Citado el: 21 de 2 de 2012.] http://tech.kaazing.com/documentation/html5/howto-admin-monitor.html.

KOU, Xuekun. 2010. *GlassFish Administration.* s.l. : Packt Publishing, 2010. ISBN-1847196500.

KUNG, Onken & Large. 2011. *TortoiseSVN: Un cliente de Subversion para Windows.* s.l. : Autoedición, 2011.

LATASA, Félix Cuadrado. 2009. *A Proposal for Model-Based Automation of Enterprise Service Change Management Processes.* Madrid : Universidad Politécnica de Madrid, 2009.

LI, Sing. 2002. *From black boxes to enterprises, Part 1: Management, JMX 1.1 style.* s.l. : IBM Developer Works, 2002. http://www.ibm.com/developerworks/java/library/j-jmx1/.

MAHMOUD, Qusay H. 2004. *Oracle Sun Developer Network (SDN).* [En línea] Sun MicroSytems, 6 de Enero de 2004. http://java.sun.com/developer/technicalArticles/J2SE/jmx.html.

MARON, Jonathan. 2010. *Application performance tuning server-side component. 7827535* USA, 10 de 12 de 2010.

MCMANUS, Éamonn y HEISS, Janice J. 2006. *MXBeans in Java SE 6: Bundling Values Without Special JMX Client Configurations.* s.l. : Sun MicroSystems, 2006. http://java.sun.com/developer/technicalArticles/J2SE/mxbeans/.

NetBeans. 2011. *NetBeans.* [En línea] Oracle Corporation, 2011. http://netbeans.org/features/index.html.

Oracle WebLogic Server. 2012. *Oracle Coporation.* [En línea] 2012. [Citado el: 21 de 2 de 2012.] http://docs.oracle.com/cd/E24329_01/web.1211/e24415/understandwls.htm#JMXCU112.

PEÑALVER, Gladis. 2008. *Metodología ágil para proyectos de software libre.* Cuidad de La Habana, Cuba : Universidad de las Ciencias Informáticas, 2008.

PRIETO, Alberto González. 2008. *Adaptive Real-time Monitoring for Large-scale Networked Systems.* Stockholm, Sweden : School of Electrical Engineering KTH, 2008.

RapidSVN. 2011. RapidSVN. [En línea] 2011. [Citado el: 13 de 12 de 2011.] http://www.rapidsvn.org/.

RUMBAUGH, James y JACOBSON, Ivar. 2007. *El lenguaje Unificado de Modelado, Manual de Referencia.* Madrid : Addison-Wesley, 2007. ISBN-9788478290871.

RUSSELL, Ethan George, y otros. 2009. *Method and system for monitoring the perfonrmance of a distributed application . 7600014* USA, 6 de 10 de 2009.

SCHULZE, Alexander. 2011. *Framework Approach for WebSockets.* s.l. : Web Technologies & Internet Applications (WebTech 2011), 2011. http://dl.globalstf.org/index.php?page=shop.product_details&flypage=flypage_images.tpl&product_id=528&category_id=42&option=com_virtuemart&Itemid=4&vmcchk=1&Itemid=4..

SCHWABER, Ken y BEEDLE, Mike. 2011. *Agile Software Development with SCRUM.* s.l. : Prentice Hall, 2011. 0130676349.

Sun Microsystems, Inc. 2006. *Java Management Extensions (JMX), Specification, version 1.4.* California : Sun Microsystems, Inc., 2006.

VAN DER VLIST, Eric. 2002. *XML Schema: The W3C´s Object-Oriented Description for XML.* s.l. : O´Reilly Media, 2002.

Visual Paradigm. 2012. Visual Paradigm for UML. [En línea] [Citado el: 15 de 2 de 2012.] http://www.visual-paradigm.com.

WITTNER, Otto. 2003. *Emergent behavior based implements for distributed network management.* s.l. : Department of Telematics Norwegian University of Science and Technology, 2003.

BIBLIOGRAFÍA CONSULTADA

Apache Software Foundation. 2011. Apache Subversion. *Apache Subversion.* [En línea] 10 de 12 de 2011. [Citado el: 10 de 12 de 2011.] http://subversion.apache.org/.

BALOS, Kazimierz, y otros. 2008. *Open interface for autonomic management of virtualized resources in complex systems - construction methodology.* 5, Kraków, Poland : Future Generation Computer Systems, 2008, Vol. 24.

CLEMM, Alexander. 2006. *Network Management Fundamentals.* s.l. : Cisco Press, 2006. ISBN-1587201372 .

DUGUAY, Claude. 2005. Extending Spring JMX support Customize resource management with the Spring framework. *IBM-DeveloperWorks.* [En línea] 01 de 11 de 2005. http://www.ibm.com/developerworks/java/library/j-springjmx/.

MOS, Adrian. 2004. *A Framework for Adaptive Monitoring and Performance Management of Component-Based Enterprise Applications.* Dublin : s.n., 2004.

MILLÁN Tejedor, Ramón Jesús. 2004. *Tendencias en gestión de red.* Comunicaciones World nº 189, s.l. : IDG Communications S.A, 2004.

NOURIE, Dana. 2005. *Oracle Sun Developer Network (SDN).* [En línea] Oracle Corporation, 24 de Mayo de 2005. http://java.sun.com/developer/technicalArticles/tools/intro.html.

PERRY, Steven J. 2002. *Java Management Extensions.* s.l. : O´Reilly, 2002. ISBN-0-596-00245-9.

Real Academia Española. *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición.* [En línea] Real Academia Española. [Citado el: 17 de 2 de 2012.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=administrar.

Real Academia Española. *DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición.* [En línea] Real Academia Española. [Citado el: 17 de 2 de 2012.] http://buscon.rae.es/draeI/SrvltGUIBusUsual?LEMA=remota&TIPO_HTML=2

THOMAS, Tony G. 2002. JMX Unlocks Application Management Potential. [En línea] AdventNet, 06 de 03 de 2002. http://www.ebizq.net/topics/systems_management/features/1738.html.