Facultad 3

Herramienta para la instalación y configuración de clústeres de servidores del tipo activo-pasivo utilizando Red Hat Cluster Suite.

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

Autor: Adrian Turiño León

Tutor: Ing. Juniel Tamayo Hernández

Cotutor: Ing. Heiler Fabars Corrales

La Habana

Año 55 de la Revolución

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Adrian Turiño León Ing. Juniel Tamayo Hernández

_____

Ing. Heiler Fabars Corrales

AGRADECIMIENTOS

A Fidel por esta maravillosa idea.

A mi tutor Juniel Hernández por su excelente trabajo.

A mis madres por ser el motor impulsor de cada una de las metas que me trazo.

A mis padres por el apoyo incondicional que siempre me han dado.

A mi hijo por ponerme una bandera en mi vida.

A mi hermanita por tenerme como referente.

A mi hermano por todo el apoyo que me brindo.

A mi novia por esos años de comprensión.

A mis amigos por estar siempre hay cuando los necesitaba y cuando no también.

DEDICATORIA

A María Esther León Jiménez, Esperanza Carballo Gilbert y al pequeño Kevin Adrián Turiño las personas más importante de mi vida.

RESUMEN

Alta disponibilidad es la capacidad de un sistema de mantener de constante la posibilidad del acceso a la información brindada incluso ante la ocurrencia de alguna falla. El método más utilizado para garantizarla es la implementación de clústeres. Red Hat Cluster Suite brinda un conjunto integrado de componentes de software que pueden ser empleados en una variedad de configuraciones para satisfacer necesidades de rendimiento, equilibrio de carga, escalabilidad, compartición de archivo y alta disponibilidad. El proceso de instalación de un clúster utilizando esta tecnología requiere el empleo de un tiempo considerable, y en muchas ocasiones se torna engorroso y complicado.

El objetivo de este trabajo fue desarrollar una herramienta para la instalación y configuración de clústeres de alta disponibilidad de servidores del tipo activo-pasivo utilizando la tecnología Red Hat Clúster Suite. El desarrollo estuvo guiado por la metodología de desarrollo Extreme Programming (XP), definida a partir de las características del proyecto y del equipo de trabajo. El resultado fundamental fue una herramienta funcional cuyo desarrollo está en correspondencia con el estilo arquitectónico establecido y los estándares de codificación definidos y con los requisitos funcionales y no funcionales aprobados.

Finalmente se realizaron pruebas de caja blanca, de caja negra y de aceptación que permitieron constatar la validez de la herramienta desarrollada como alternativa que no solo agiliza los procesos de instalación y configuración de clústeres de alta disponibilidad utilizando la tecnología propuesta, sino que también contribuye a reducir los errores de este.

PALABRAS CLAVES

Clúster, Alta disponibilidad, Software, Herramienta de Instalación, Red Hat Cluster Suite.

Índice de contenidos

ÍNDICE DE TABLAS.

ÍNDICE DE ILUSTRACIONES.

Introducción

El procesamiento de grandes volúmenes de datos o peticiones de servicios, se encuentra limitado por la incapacidad de ser procesados por ordenadores con arquitectura convencional. La adquisición de computadoras potentes en arquitectura de hardware para llevar a cabo tareas de cómputo complejas, unido al costo por mantenimiento una vez adquiridas, constituyen una opción no viable en la mayoría de las situaciones presentadas en Cuba y el resto de los países subdesarrollados. Una solución alternativa ante esta disyuntiva es la utilización de herramientas informáticas capaces de sustituir, mediante otros métodos, este tipo de tareas.

La utilización de clústeres[1] es una alternativa que se ha venido utilizando para lograr la alta disponibilidad[2] y el balanceo de carga[3] en aplicaciones que manejan grandes volúmenes de datos o necesitan prestar determinado servicio a una gama de clientes que puede ir en ascenso y colapsar un servidor convencional.

El término clúster se aplica no sólo a computadoras de alto rendimiento sino también a los conjuntos de computadoras, construidos utilizando componentes de hardware comunes y software libre. (Gómez, 2010) Según este autor un clúster es un conjunto o conglomerado de computadoras construidos mediante la utilización de hardware común y que se comportan como si fuesen una única computadora.

La computación basada en clúster surge gracias a la disponibilidad de microprocesadores de alto rendimiento más económicos y de redes de alta velocidad, y también gracias al desarrollo de herramientas de software para cómputo distribuido de alto rendimiento; todo ello frente a la creciente necesidad de potencia de cómputo para aplicaciones en las ciencias y en el ámbito comercial, así como de disponibilidad permanente para algunos servicios. Por otro lado, la evolución y estabilidad que ha alcanzado el sistema operativo GNU/Linux, ha contribuido de forma importante, al desarrollo de muchas tecnologías nuevas, entre ellas las de clusterización. (Gómez, 2010)

[1] **Clúster**: Grupo, Racimo, etcétera.

[2] Un **clúster de alta disponibilidad** es un conjunto de dos o más máquinas que se caracterizan por mantener una serie de servicios compartidos y por estar constantemente monitorizándose entre sí.

[3] **Balanceo de carga** es repartir las peticiones de servicio que reciba el clúster, entre todos los otros ordenadores que lo forman.

Los clústeres se pueden separar en dos modelos, primero configurando el sistema del tipo Activo-Activo que no es más que dar responsabilidades de ejecución a todos los nodos donde se procesa la información o se le da respuesta a un servicio, además los nodos clusterizados se mantienen monitoreando los servicios para en caso de fallas, asumir esos servicios también. Este modelo es conocido también como doble Activo-Pasivo que es el otro modelo que se implementa en los clústeres. Una posible configuración del Activo-Pasivo es donde las aplicaciones se ejecutan sobre uno de los nodos, el activo, mientras que los nodos restantes actúan como respaldos de este para atender los servicios ofrecidos.

El aumento del flujo de información en la red y el acceso casi masivo de la población a internet y los servicios que está brinda han elevado la necesidad de mantener los sistemas disponibles para su uso todo el tiempo. Para garantizar la alta disponibilidad de los sistemas en la red se han desarrollado varias alternativas que implican la instalación y configuración de clústeres de servidores que garanticen la permanencia de los servicios.

Una de estas alternativas es la propuesta por el sistema operativo Red Hat llamada Red Hat Cluster Suite (RHCS). Esta alternativa consiste en un conjunto integrado de componentes de software que pueden ser empleados en una variedad de configuraciones para satisfacer necesidades de rendimiento, equilibrio de carga, escalabilidad, compartición de archivo y alta disponibilidad. (Red Hat, 2007) Para el uso de esta alternativa con el objetivo de lograr alta disponibilidad de servicios, es necesario realizar la instalación y configuración en cada nodo de los componentes de software necesarios, lo que en muchas ocasiones requiere el empleo de un tiempo considerable, en servidores Linux que generalmente no poseen interfaz gráfica de usuario, este proceso se torna engorroso y complicado. En la actualidad no existe ninguna aplicación que posibilite la instalación autónoma y unificada de los distintos componentes de software necesarios para la puesta en funcionamiento de un clúster de alta disponibilidad del tipo Activo-Pasivo.

A raíz de la situación planteada se define el siguiente problema a resolver: **¿Cómo mejorar el proceso de instalación de clústeres de alta disponibilidad de servidores utilizando la tecnología Red Hat Clúster Suite?** Teniéndose como objeto de estudio la **Clusterización de servidores**. El objetivo general de este trabajo es **desarrollar una herramienta para la instalación y configuración de clústeres de alta disponibilidad de servidores del tipo activo-pasivo utilizando la tecnología Red Hat Clúster Suite** de manera que se mejore este proceso. Enmarcado en el campo de acción del **desarrollo de herramientas de instalación y**

configuración de clústeres de servidores. La idea que se procura defender con este trabajo es que **el desarrollo de una herramienta para la instalación y configuración de clústeres de servidores de alta disponibilidad utilizando la tecnología Red Hat Clúster Suite, contribuirá a mejorar este proceso a la vez que se reducen los errores de este.** Con el fin de darle cumplimiento al objetivo general se desglosó en mismo en los siguientes objetivos específicos:

- Elaborar el marco teórico de la investigación.
- Realizar el diseño del sistema.
- Realizar la implementación del sistema.
- Validar los resultados obtenidos.

Durante la realización de este trabajo se utilizarán diferentes métodos científicos para estudiar las características del objeto de estudio de la investigación:

Métodos Lógicos:

- **Histórico-Lógico:** Para el estudio de las herramientas y tecnologías existentes utilizadas en la instalación y configuración de clústeres.
- **Analítico-sintético:** Para la extracción de las características, rasgos y elementos más importantes para la compresión del funcionamiento de los clústeres.
- **Hipotético-Deductivo:** Para la obtención de la mejor propuesta tecnológica de clusterización que fundamente las necesidades del trabajo.
- **Modelación:** Para la construcción de modelos o diagramas que permiten un mejor entendimiento del proceso de desarrollo de la herramienta y así lograr diseñar e implementar todas las funcionalidades necesarias para el correcto funcionamiento de este proceso.

Métodos Empíricos:

- **Simulación:** Para poder ejecutar el producto desarrollado y así poder crear casos de prueba que permitan medir la correcta implementación de las funcionalidades de la herramienta.
- **Entrevista:** Para la interacción con el cliente con el objetivo de extraer los requisitos necesarios para la puesta en marcha del trabajo.

A continuación se resume el contenido de este trabajo:

El capítulo 1 de este documento hace referencia a la fundamentación teórica del trabajo, describiéndose todos los elementos de la teoría que sostienen el problema científico y los objetivos de la investigación. En este capítulo se realiza un estudio del estado del arte de las herramientas y tecnologías existentes para la instalación y configuración de clústeres de alta disponibilidad utilizando la tecnología Red Hat Clúster Suite. Se fundamentan además las tecnologías, lenguajes de programación, herramientas y metodologías utilizadas durante el proceso de desarrollo del software.

En el capítulo 2 se hará un análisis detallado de la solución que se propone, exponiendo para ello los artefactos generados por la metodología a utilizar.

En el capítulo 3 se expone toda la validación realizada a la aplicación obtenida, así como los casos de pruebas realizados para comprobar que las funcionalidades están de acuerdo con lo que se necesita.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El término clúster se aplica no sólo a computadoras de alto rendimiento sino también a los conjuntos de computadoras, construidos utilizando componentes de hardware comunes y software libre. (Gómez, 2010) Según el autor un clúster es un conjunto o conglomerado de computadoras construidas mediante la utilización de hardware común y que se comportan como si fuesen una única computadora. Los sistemas de clústeres proporcionan fiabilidad, escalabilidad y disponibilidad a servicios de producción crítica. Con un clúster se pueden cubrir necesidades de rendimiento, alta disponibilidad, balance de cargas, escalabilidad y compartición de archivos.

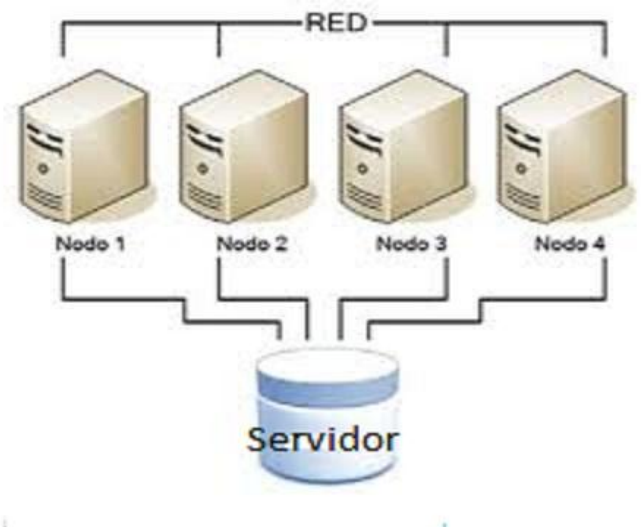

Figura 1. Clúster.

Un clúster está compuesto por dos o más computadoras (*llamados nodos* o *miembros*) que trabajan juntos para ejecutar una tarea. Hay cuatro clases de clústeres:

- Almacenamiento
- Alta disponibilidad
- Balance de carga
- Alto rendimiento

Los clústeres de almacenamiento proporcionan una imagen de sistema de archivos consistente a lo largo de los servidores, permitiendo que estos lean y escriban de forma simultánea a un sistema de archivos compartido. Un clúster de almacenamiento simplifica la administración de la información al limitar la instalación de aplicaciones a un sistema de archivos. Así mismo, con

un sistema de archivos a lo largo de los nodos, un clúster de almacenamiento elimina la necesidad de copias redundantes de los datos de la aplicación y simplifica la creación de copias de seguridad y recuperación contra desastres.

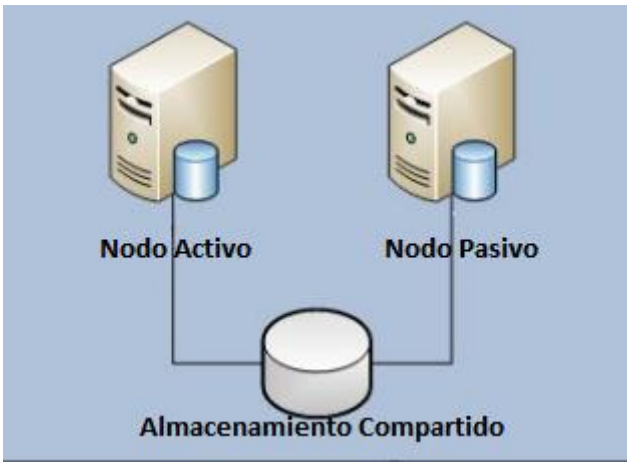

Figura 2. Almacenamiento Compartido.

Los clústeres de alta disponibilidad proporcionan continua disponibilidad de los servicios a través de la eliminación de la falla por un único elemento y a través del proceso de recuperación en contra de fallas al trasladar el servicio desde el nodo de defectuoso a otro nodo completamente funcional. Generalmente, los servicios en los clústeres de alta disponibilidad leen y escriben datos a través de la lectura y escritura a un sistema de archivos montado. Así, un clúster de alta disponibilidad debe mantener la integridad de los datos cuando un nodo recibe el control del servicio desde otro nodo. Los nodos defectuosos no son vistos por los clientes fuera del clúster. Los clústeres de alta disponibilidad son conocidos también como clústeres con recuperación contra fallas.

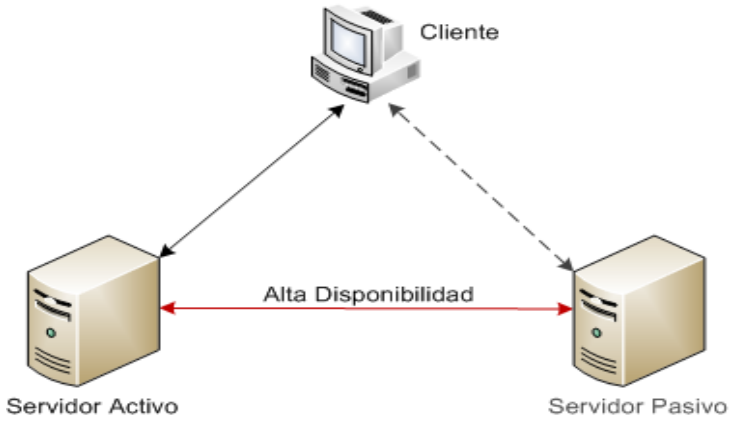

Figura 3. Alta Disponibilidad.

Los clústeres de balance de carga responden a peticiones de servicios de red desde diferentes nodos para balancear las peticiones entre los nodos. El balance de carga proporciona escalabilidad económica porque se puede configurar el número de nodos de acuerdo con los requerimientos de balance de carga. Si un nodo en un clúster de balance de carga falla, el software de balance de carga detecta la falla y asigna las peticiones a otros nodos en el clúster. Los nodos defectuosos en un clúster de balance de carga no son visibles desde los clientes fuera del clúster.

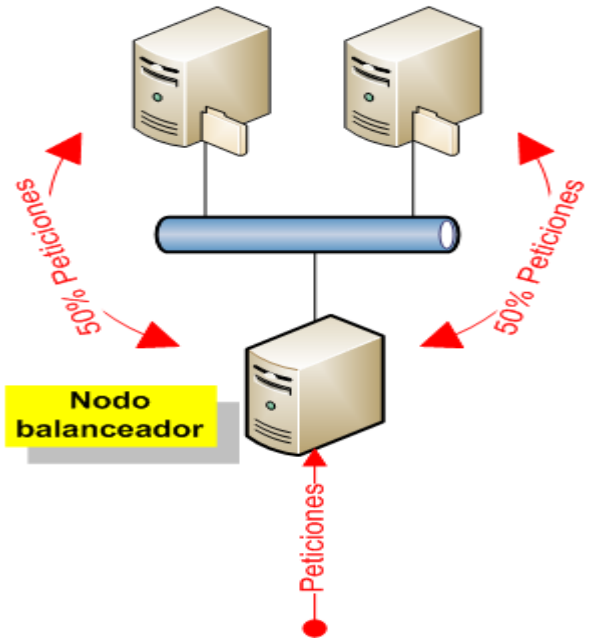

Figura 4. Balance de Carga.

Los clústeres de alto rendimiento utilizan los nodos para ejecutar cálculos simultáneos. Un clúster de alto rendimiento permite que las aplicaciones trabajen de forma paralela, mejorando así el rendimiento de éstas. Los clústeres de alto rendimiento son conocidos como clústeres computacionales o clústeres de computación de red.

Los clústeres se pueden separar en dos modelos, primero configurando el sistema del tipo **Activo-Activo** que no es más que dar responsabilidades de ejecución a todos los nodos donde se procesa la información o se le da respuesta a un servicio. Además los nodos clusterizados se mantienen monitoreando los servicios para en caso de fallas, asumir esos servicios también. Este modelo es conocido también como doble **Activo-Pasivo**. Una posible configuración del modelo Activo-Pasivo es aquel donde las aplicaciones se ejecutan sobre uno de los nodos, el activo, mientras que los nodos restantes actúan como respaldos de este para atender los servicios ofrecidos.

Como se puede apreciar el ámbito de los clústeres es bastante amplio y no existe un mecanismo rígido que los defina, o sea que dada la necesidad de utilizar un clúster se deberá primero analizar las características de los servicios que se ofrecerán y las necesidades específicas de la organización que lo utilizará.

1.1 Tecnología de clusterización Red Hat Clúster Suite

La tecnología clúster permite a las organizaciones incrementar su capacidad de procesamiento usando tecnología estándar, tanto en componentes de hardware como de software que pueden adquirirse a un costo relativamente bajo.

Red Hat Clúster Suite (RHCS) es un conjunto integrado de componentes de software que puede ser implementado en una amplia variedad de configuraciones para cubrir las necesidades de rendimiento, alta disponibilidad, balance de carga, escalabilidad, compartición de archivos y economía de recursos. (Red Hat, 2007)

Infraestructura del clúster

- Proporciona funciones fundamentales para que los nodos trabajen juntos: administración del archivo de configuración, administración de membresías, administración de cierres de exclusión y aislamiento.
- Administración de servicios de alta disponibilidad. Proporciona la transferencia de servicios de un nodo a otro en caso de que uno falle a través del componente de administración de servicios de alta disponibilidad.
- Herramienta de administración de clúster. Herramienta de configuración y administración para configurar y administrar un clúster. Las herramientas se utilizan con los componentes de infraestructura del clúster, los componentes de alta disponibilidad, de administración de servicios y de almacenamiento.
- Red Hat Clúster Suite proporciona balance de carga a través del Servidor Virtual de Linux (LVS). Este software de encaminamiento proporciona balance de carga de IP. LVS se ejecuta en un par de servidores pertinentes que distribuyen las peticiones de los clientes a los servidores reales que están tras los servidores LVS.

RHCS es una suite en la que se incluye una serie de componentes de software que permiten en su conjunto un trabajo multidisciplinario para los sistemas en los que se utiliza. Algunos de estos componentes presentan interfaces graficas que elevan la usabilidad, de forma que se

ubica como una atractiva opción para elevar la gestión en general de las posibilidades que añaden los clústeres. Además permite el monitoreo de una gran cantidad de recursos y con estos crear disimiles servicios los cuales pueden ejecutarse concurrentemente sin comprometerse unos a otros. Sin embargo a pesar de sus buenas prestaciones, esta no presenta una opción que agrupe y controle el proceso de instalación, resaltando la necesidad y factibilidad del desarrollo de una herramienta de software con estos fines.

En el siguiente epígrafe se realiza un estudio de las alternativas existentes que facilitan el control y seguimiento de los procesos de instalación y configuración de clústeres, con el objetivo de identificar elementos comunes necesarios que sirvan de base para la herramienta a desarrollar.

1.2 Herramientas para la instalación y configuración de clústeres.

Durante esta etapa se indagó acerca de la existencia de herramientas de software que brindaran soporte a los procesos de instalación y configuración de clústeres de servidores, a través de interfaces gráficas de usuario que ayudaran a los administradores de servidores a desarrollar estos procesos de manera efectiva.

Durante el estudio se analizaron grandes empresas dedicadas a la producción a gran escala de hardware y software y al soporte a estos, como IBM, DELL, HP o CentOS. Este estudio sirvió para constatar que a pesar de que algunas de estas (IBM, HP) proveen entre los servicios de soporte soluciones de clusterización de alta disponibilidad o rendimiento, ninguna reporta la existencia o utilización de una herramienta que facilite la gestión y seguridad de los procesos de instalación y configuración de clústeres.

Solo las empresas Microsoft y Oracle presentan herramientas que facilitan la instalación y configuración de servicios de alta disponibilidad para soluciones propias.

1.2.1 Microsoft

Microsoft no está exenta de las tecnologías de clusterización y por ende, de las ventajas que se pueden obtener de esta en cuando al logro de la alta disponibilidad que puede requerir determinado servicio. Es por eso que en sistemas operativos como Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows 2000 Advanced, Windows NT 4.0 Enterprise Edition Server se propone el servicio de Microsoft Cluster Service (MSCS). (Microsoft)

El proceso de montaje de un clúster en estos sistemas operativos es guiado por una serie de herramientas que facilitan el trabajo, en cuanto a la abstracción del usuario de cómo se realiza la instalación del sistema. Al leer cualquier manual de los que ofrece Microsoft para el trabajo de los usuarios con los clústeres se puede apreciar que estas herramientas cuentan con interfaces gráficas amigables donde se van conformando cada uno de los parámetros necesarios, aunque presentan como desventaja que no cuentan con una herramienta que unifique todos estos componentes que se necesitan. Además este software al pertenecer a esta compañía se necesita de una licencia para poder utilizarlos por su carácter privativo. A continuación se muestra una de las interfaces de la herramienta para la instalación y configuración de clústeres de alta disponibilidad que propone Microsoft.

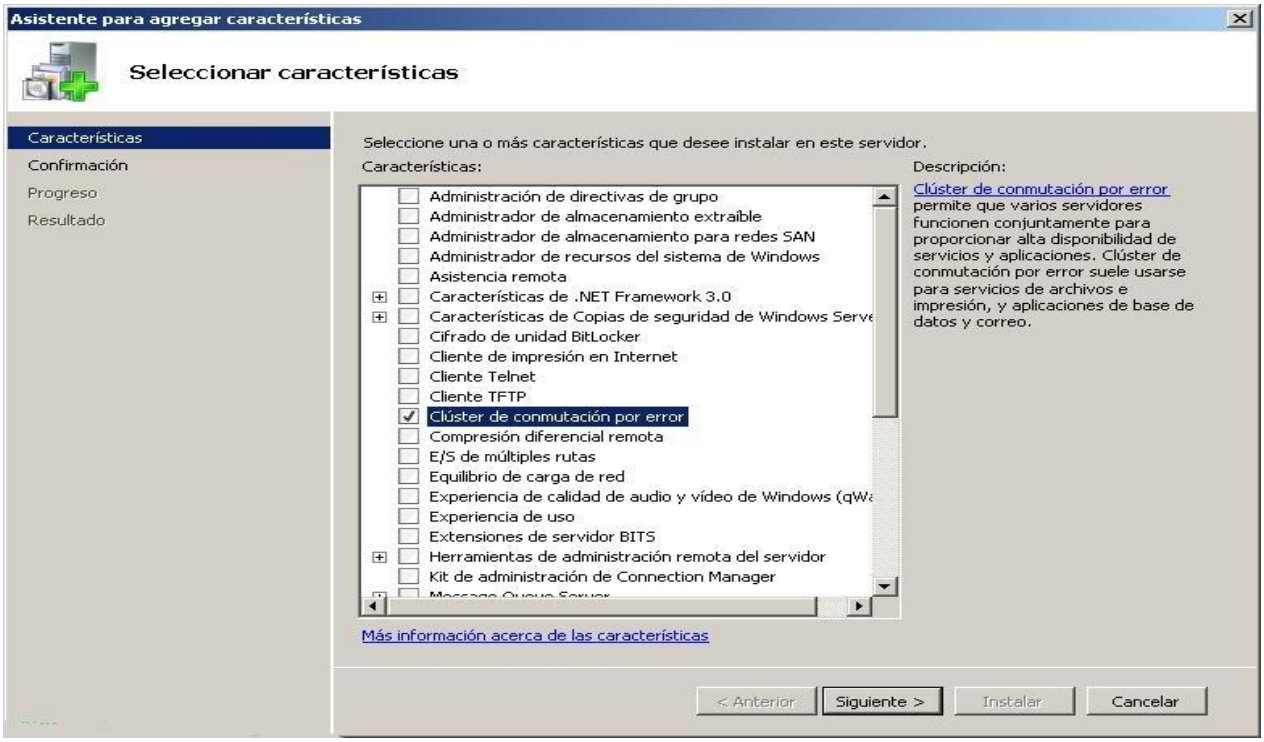

Figura 5. Asistente para agregar características.

1.2.2 Oracle Database 11g

Oracle Database 11g es una herramienta para la instalación y configuración de bases de datos. Oracle Database es líder en el mercado y representa la base de datos preferida por miles de empresas, desarrolladores y administradores de todo el mundo. Con el lanzamiento de esta herramienta se les permite a las empresas adoptar nuevas tecnologías mientras se minimiza el riesgo. Asimismo sobre la base de sus capacidades de autoadministración, ha realizado

importantes avances en el las áreas de capacidad de administración y diagnóstico de fallas. (Jagan R, Athreya, 2007)

Esta herramienta cuenta con una interfaz visual para realizar el proceso de instalación y configuración de distintos tipos de bases de datos. Estas serán desplegadas en la cantidad de nodos que el usuario decida tener. La herramienta realiza la conexión a los nodos usando un protocolo seguro, en este caso SSH. Permite la gestión de los nodos del clúster a la vez que presenta un asistente de ayuda y una excelente navegabilidad dentro de la misma. Otra ventaja de esta herramienta es la centralización de todos los componentes que brinda en una sola aplicación. La siguiente imagen permite observar el diseño de la interfaz principal del sistema donde se podrán apreciar todas estas características.

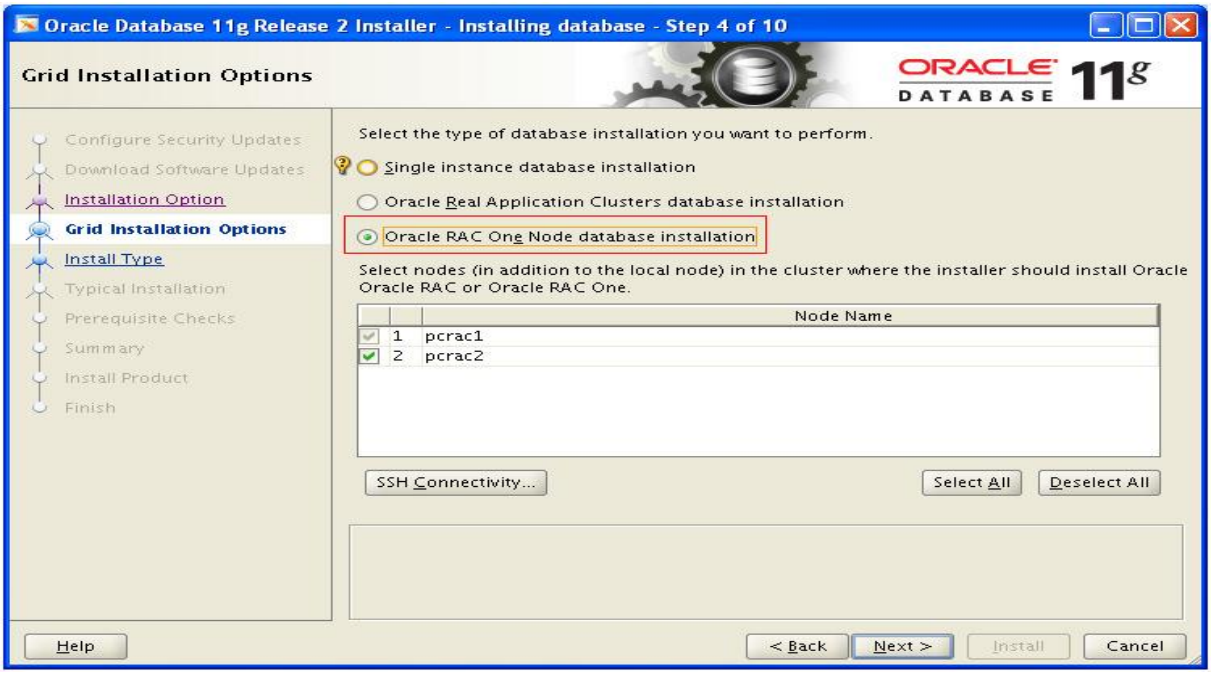

Figura 6. Oracle Database 11g.

Otras de las características que presenta este sistema, las que desafortunadamente no son tan atractivas como las anteriores son que solamente garantiza la disponibilidad de bases de datos y que pertenece a la familia del software propietario, en este caso a la compañía de Oracle por lo que no es una herramienta disponible para todos.

Aunque el estudio de este tipo de herramientas no arrojo una gran cantidad de ejemplos y los analizados tienen un carácter propietario por lo que se hace difícil acceder a muchos términos que serían de gran valor para la investigación, se logró aprender sobre el diseño y organización

de las interfaces presentadas por estas como base para el desarrollo de la herramienta propuesta en este trabajo.

A continuación se definen los elementos ingenieriles, tecnológicos y arquitectónicos que guían el desarrollo de la herramienta en cuestión, dígase metodología de desarrollo de software, herramientas de modelado asistido, lenguaje de programación, entorno de desarrollo integrado y elementos arquitectónicos fundamentales.

1.3 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software son “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”. (PIATTINI 1996). “Son todas las actividades necesarias para transformar los requisitos de un usuario en un sistema de software”. (Jacobson, y otros, 2000). Las metodologías de desarrollo de software imponen un proceso a cumplir disciplinadamente para el desarrollo de un software cualquiera, con el fin de hacerlo más predecible y eficiente. Además tienen como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. Las metodologías de desarrollo se dividen en dos grupos, las metodologías ágiles y las robustas.

1.3.1 Metodologías de desarrollo robustas.

Las metodologías robustas están guiadas por una fuerte planificación. Centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo y en cumplir con un plan de proyecto, definido en la fase inicial del mismo. Estas están caracterizadas por:

- Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
- Cierta resistencia a los cambios.
- Proceso mucho más controlado, con numerosas políticas o normas.
- Existe un contrato prefijado.
- El cliente interactúa con el equipo de desarrollo mediante reuniones.
- Grupos grandes y posiblemente distribuidos.
- Gran cantidad de artefactos.
- Disimiles roles.
- La arquitectura del software es esencial y se expresa mediante modelos.

1.3.2 Metodologías de desarrollo ágil

Las metodologías ágiles se caracterizan por tener un desarrollo incremental para producir tempranamente pequeñas entregas en ciclos rápidos y disposición para el cambio y adaptación continua. (Claro Sánchez, y otros, 2011) Además por hacer énfasis en la comunicación cara a cara, es decir, se basan en una fuerte y constante interacción, donde clientes y desarrolladores trabajan constantemente juntos, estableciéndose así una estrecha comunicación. Estas metodologías están orientadas al resultado del producto y no a la documentación; exige que el proceso sea adaptable, permitiendo realizar cambios de último momento.

La siguiente tabla muestra una comparación entre algunos de los aspectos más importantes que definen estos dos tipos de metodologías. Esta información podría ser de gran ayuda para hacer la selección de una metodología en un determinado sistema.

Tabla I. Comparación entre metodologías ágiles y robustas (Pardillo Castañeda, 2007).

Metodologías robustas	**Metodologías Ágiles**
Más Roles, más específicos	Pocos Roles, más genéricos y flexibles
Más Artefactos. El modelado es esencial, mantenimiento de modelos	Pocos Artefactos. El modelado es prescindible, modelos desechables.
Existe un contrato prefijado	No existe un contrato tradicional o al menos es bastante flexible.
La arquitectura es esencial: Se promueve que la arquitectura se defina tempranamente en el proyecto.	Menos énfasis en la arquitectura: La arquitectura se va definiendo y mejorando a lo largo del proyecto.
Cierta resistencia a los cambios (estos pueden provocar grandes costos).	Sujeta a cambios en cualquier etapa del proyecto.
Dirigidas al proceso: roles, actividades y artefactos.	Dirigidas a las personas: el individuo y el trabajo en equipo.
El cliente interactúa con el equipo de desarrollo mediante reuniones.	El cliente es parte del equipo de desarrollo.

Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas en proyectos grandes y con equipos posiblemente dispersos.	Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños y trabajando en el mismo sitio.
Proceso mucho más controlado, con numerosas políticas o normas.	Proceso menos controlado, con pocos principios.
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.

Teniendo en cuenta las particularidades presentes en cada una de estas metodologías y atendiendo a las necesidades y condiciones del sistema que se desea implementar tales como la cercanía del cliente con el equipo de desarrollo, la inclinación del sistema hacia obtener un resultado funcional y no una documentación detallada, por el periodo de tiempo con que se dispone para la realización del trabajo se decide utilizar una metodología ágil para guiar el proceso de desarrollo del software. A continuación se hace un análisis entre algunas de las metodologías ágiles más conocidas para definir cuál se adapta más a las características del sistema.

Scrum

SCRUM es una forma de gestionar proyectos de software. No es una metodología de análisis, ni de diseño, como podría ser RUP (Rational Unified Process - Proceso Racional Unificado), es una metodología de gestión del trabajo. Una de las características más importantes es que es muy fácil de explicar y de entender, lo que ayuda mucho a su implantación. (Pardillo Castañeda, 2007) Por otra parte SCRUM puede ser aplicado a distintos modelos de calidad puesto que estos definen qué se tiene que hacer, es decir, definen que se tiene que gestionar el proyecto, pero no definen cómo. Ahí es donde entra SCRUM como modelo de gestión de proyectos. Scrum se define como un proceso de gestión y control que implementa técnicas de control de procesos; se lo puede considerar un conjunto de patrones organizacionales. La dimensión del equipo total de Scrum no debería ser superior a diez ingenieros. El número ideal es siete. Si hay más, lo más recomendable es formar varios equipos. (Pardillo Castañeda, 2007)

No hay una ingeniería del software prescripta para Scrum; cada quien puede escoger entonces las prácticas de automatización, inspección de código, pruebas unitarias, análisis o programación en pares que le resulten adecuadas. Es habitual que Scrum se complemente con XP (eXtreme Programming); en estos casos, Scrum suministra un marco de gestión basado en patrones organizacionales, mientras XP constituye la práctica de programación, usualmente orientada a objetos y con fuerte uso de patrones de diseño. Uno de los nombres que se utiliza para esta alianza es XP@Scrum. (Pardillo Castañeda, 2007)

XP (eXtreme Programming)

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, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo.

XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

La programación extrema se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del software**.** (Corbea & Pérez, 2007)

Características de la Metodología XP

• Los diseñadores y programadores se comunican efectivamente con el cliente y entre ellos mismos.

• Los diseños del software se mantienen sencillos y libres de complejidad o pretensiones excesivas.

• Se obtiene retroalimentación de usuarios y clientes desde el primer día gracias a las baterías de pruebas.

• El software es liberado en entregas frecuentes tan pronto como sea posible.

• Los cambios se implementan rápidamente tal y como fueron sugeridos.

• Las metas en características, tiempos y costos son reajustadas permanentemente en función del avance real obtenido.

• Empieza en pequeño y añade funcionalidad con retroalimentación continua.

• El manejo del cambio se convierte en parte sustantiva del proceso.

• El costo del cambio no depende de la fase o etapa.

• No introduce funcionalidades antes que sean necesarias.

• El cliente o el usuario se convierte en miembro del equipo.

El cliente siempre está disponible para resolver dudas y para decidir qué y qué no se hace en cada momento, en función de los intereses del negocio. Debido a su inmersión dentro del equipo junto a las pruebas que verifican si la funcionalidad es la correcta y deseada, el cliente obtiene información absolutamente realista del estado del proyecto. (Corbea & Pérez, 2007)

Agile Unified Process (AUP)

Agile Unified Process (AUP), es una versión simplificada del Proceso Unificado de Rational (RUP). Descrito en una forma simple, fácil de entender y brinda un enfoque de desarrollo de software utilizando técnicas ágiles y conceptos de RUP. El proceso unificado (Unified Process o UP) es un marco de desarrollo de software iterativo e incremental. AUP abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente. El modelado agrupa los tres primeros flujos de RUP (Modelamiento del negocio, Requerimientos y Análisis y Diseño). Dispone de cuatro fases igual que RUP: Creación, Elaboración, Construcción y Transición. (Ortiz Fidalgo, y otros, 2010)

El ciclo de vida de AUP se puede ver en sus cuatro fases Inicio, Elaboración, Construcción y Transición. Su naturaleza iterativa se expresa en sus siete disciplinas que son: Modelado, Implementación, Prueba, Despliegue, Gestión de Configuración, Gestión de Proyecto y Ambiente. (Ortiz Fidalgo, y otros, 2010)

Analizadas las características de las anteriores metodologías de desarrollo de software, es seleccionada **XP** como la candidata para guiar la implementación resultante de la investigación, ya que se ajusta al ambiente que se presenta. XP es una metodología incluida dentro de las denominadas ágiles, se encamina más a lograr entregas desde el principio basadas en la comunicación e interacción directa que debe existir entre el equipo de desarrollo y el cliente, siendo este último parte del mismo, además, existe poca disponibilidad de personal dentro del grupo de trabajo.

1.4 Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos, caracteres y reglas que son entendibles y ejecutables por un computador. Tienen un conjunto de instrucciones que permiten realizar operaciones de entrada/salida, manipulación de textos, lógica/comparación y almacenamiento/recuperación.

Para la implementación de la herramienta se realiza un análisis de los lenguajes de programación más utilizados en el desarrollo de aplicaciones de escritorio de este tipo tales como, Python, C++ y Java.

Python

El objetivo de Python es la sencillez, código sencillo y entendible. El Python es un lenguaje interpretado, eso significa que no se compila ni se enlaza, sino que el mismo código fuente hace las veces de ejecutable y para eso siempre se necesita tener instalada la plataforma del intérprete. Se le pasa de parámetro al intérprete el código en un archivo *.py. En general los lenguajes interpretados son sencillos, pero tienen un rendimiento mucho más bajo por ello, las bibliotecas de Python están programadas en C++. (Martínez Prieto, 2012)

C++

El lenguaje C++ no es recomendable porque la programación se realiza a bajo nivel, implicando un importante incremento de la dificultad en cuanto a la declaración de eventos, visualización, punteros, control de la memoria y una disminución de la productividad. Aunque es muy bueno en cuanto al rendimiento y la protección del código presenta una limitada portabilidad en diferentes sistemas operativos porque depende de bibliotecas que en ocasiones son dependientes del sistema operativo. (Martínez Prieto, 2012)

Java

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Java soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza

cálculos. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red. (Marañón, 1999) Además ofrece una gran cantidad de bibliotecas como Swing para la realización de las interfaces gráficas y Jsch para las conexiones usando protocolo seguro en este caso SSH, las cuales son aspectos fundamentales a tener en cuenta.

Además de las características mencionadas anteriormente, fue valorada la experiencia y los conocimientos del equipo de trabajo con los diferentes lenguajes propuestos, decidiéndose utilizar Java como lenguaje para la implementación del sistema.

Lenguaje unificado de Modelado (UML)

El Lenguaje de Modelado (UML) permite la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo, siendo utilizado para la modelación de sistemas, con características orientado a objetos. Facilita a los integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente, promueve la reutilización e incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. (Vera, 2008)

1.5 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering, o como se le conoce en español Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. Existen muchísimas herramientas CASE que responden a diferentes necesidades de los clientes que las usan; entre ellas se pueden encontrar Visual Paradigm[4], Rational Rose Enterprise Suite[5], Erwin[6], EasyCASE[7], Oracle Designer[8], System Architect[9] por solo citar algunas. (Vera, 2008)

[4] Visual Paradigm: Proporciona un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

[5] Rational Rose Enterprise Suite: permite crear los diagramas que se van generando durante el proceso de Ingeniería en el Desarrollo del Software.

Rational Rose Enterprise Suite

Rational Rose, es una herramienta CASE desarrollada por Rational Corporation, basada en UML, que permite crear los diagramas que se van generando durante el proceso de Ingeniería en el desarrollo del software. Las personas que desarrollaron Rational Unified Process (RUP) son miembros de Rational Corporation, por lo que el mismo es completamente compatible con esta metodología, brinda muchas facilidades en la generación de la documentación del software que se está desarrollando, además posee un gran número de estereotipos predefinidos que facilitan el proceso de modelación del software. Dicha herramienta es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño, pero tiene la limitación de que aún hay varios lenguajes de programación que no soporta o que solo lo hace a medias. Por otra parte, una vez que se tiene el diagrama de clases persistentes a partir del cual se genera la base de datos del sistema, no existe la posibilidad de exportar ese modelo hacia algún sistema gestor de bases de datos. (Quintana Santisteban, 2008)

Visual Paradigm

Visual Paradigm es una herramienta CASE que ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (la versión Community Edition, ya que existe la Enterprise, Professional, etc.). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones. También permite la reutilización del software, portabilidad y estandarización de la documentación,

[6] Erwin: herramienta de diseño de base de datos. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones.
[7] EasyCASE: Herramienta que permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación.
[8] Oracle Designer: Herramienta de software para analizar los requerimientos de negocios y para diseñar y generar sistemas cliente/servidor que satisfagan tales requerimientos.
[9] System Architect: Provee soporte para técnicas variadas para el desarrollo de sistemas de información. Permite generar automáticamente plantillas de código en varios lenguajes de programación y también esquemas de implementación para gestores de bases de datos relacionales.

además del uso de las distintas metodologías propias de la Ingeniería del Software. (Zuzel, y otros, 2009)

Características de Visual Paradigm

- Soporta aplicaciones Web y de escritorio.
- Genera código y base de datos a partir de los diagramas UML realizados.
- Permite la realización de Ingeniería Inversa.
- Permite la generación automática de informes en diferentes formatos.
- Provee abundantes tutoriales, demostraciones interactivas y proyectos UML.
- Facilita la integración con herramientas como Eclipse[10], NetBeans[11], Oracle JDeveloper [12]entre otros.
- Destaca además por su robustez, usabilidad y portabilidad.

Tabla II. Comparación de las herramientas CASE.

Aspecto de Valoración.	**Rational Rose Enterprise Suite**	**Visual Paradigm**
Desventajas	• No soporta varios lenguajes de programacion. • No existe la posibilidad de exportar el modelo de datos de las clases persistentes hacia algún sistema gestor de bases de datos. • Propietario	• Las imágenes y reportes generados, no son de muy buena calidad.

[10] Eclipse: Es una potente y completa plataforma de programación, desarrollo y compilación.
[11] NetBeans: Es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java.
[12] JDeveloper: Es un entorno de desarrollo integrado desarrollado por Oracle Corporation para muchos lenguajes. Es un software propietario pero gratuito desde 2005.

Atendiendo a las particularidades de cada una de estas herramientas y las desventajas que presentan se define usar para esta versión del sistema el Visual Paradigm como herramienta CASE que permitirá la creación de los diagramas de clases y de despliegue necesarios para esta primera versión del sistema. Además se puede resaltar que en la universidad se cuenta con la licencia de uso de ella por lo que se puede publicar este trabajo y sus resultados si incurrir en problemas legales asociados a los derechos de autor.

1.6 Entornos de desarrollo.

Un entorno de desarrollo integrado, por sus siglas en inglés (IDE), consiste en un software cuyo principal objetivo es el desarrollo de otro software. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas, que incluyen varias funcionalidades en una misma suite, un editor de código, un constructor de interfaz gráfica, administración de proyectos y archivos, un compilador y depurador de código e integración con sistemas controladores de versiones o repositorios. Pueden especializarse en el trabajo con un solo lenguaje de programación o con varios de ellos. (Netbeans, 2012)

Eclipse

En noviembre de 1998, una de las transnacionales más grandes del mundo IBM, otorgó la tarea a uno de sus grupos de desarrollo de software, denominado Object Technology International (OTI), de crear una plataforma de desarrollo común para confeccionar sus productos, basados en el lenguaje de programación Java, uno de los más populares en ese momento; surgiendo de esta manera el denominado proyecto Eclipse. (Mariño Echemend, y otros, 2010)

Cuando la plataforma Eclipse fue donada a la comunidad open source por IBM en el año 2001, la noticia no fue notable en el mundo por la enorme suma de dinero que IBM declaró que se había gastado en el desarrollo de la misma (40 millones de dólares); sino debido al resultado obtenido con esta millonaria inversión: un producto inigualable por su arquitectura madura, bien diseñada, extensible y basada en complementos. (Mariño Echemend, y otros, 2010)

NetBeans 7.2

NetBeans es un proyecto de código abierto para desarrolladores de software. Puede obtener todas las herramientas que necesite para crear aplicaciones profesionales para el escritorio, la empresa, la web y equipos móviles con el lenguaje Java, C/C++, y Ruby. NetBeans IDE es fácil

de instalar y de uso instantáneo y se ejecuta en varias plataformas incluyendo Windows, Linux y Mac OS X y Solaris. (Gutiérrez, y otros, 2012)

Otras características que hacen interesante la elección de NetBeans son las siguientes:

- Los proyectos desarrollados no dejan de ser multiplataforma y poseen lanzadores para cada plataforma.
- Sistema de ventanas práctico para desarrollar las interfaces de usuario.
- Su licencia permite construir aplicaciones open source y comerciales.

NetBeans IDE 7.2 es una herramienta de programación integrada. Está enfocado entre otros al lenguaje de Programación Java, lenguaje definido para la confección del trabajo y es el entorno de desarrollo con el que más experiencia se cuenta en el equipo. Además pertenece a la familia del software libre, posee un sistema para hacer un reconocimiento y carga de clases, métodos y objetos, permite la creación de aplicaciones de escritorio como la que se necesita para el desarrollo de la herramienta.

1.7 Conclusiones

RHCS no cuenta con una interfaz gráfica para apoyar el control en la instalación y configuración de los componentes de software que utiliza por lo que se hace necesaria la implementación de una herramienta donde se unifiquen los componentes de software propuestos y se limiten las desventajas que se puede encontrar en ella.

Las características que tienen la herramienta a desarrollar y el grupo de trabajo guiaron el proceso de selección de la metodología de desarrollo, entorno de trabajo y herramientas necesarias para la implementación de la aplicación.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Este capítulo está enmarcado en las dos primeras fases definidas en la metodología XP, planificación y diseño. Se abordan temas relacionados con el funcionamiento del sistema, se muestran las historias de usuarios, se definen las iteraciones por los programadores, construyéndose además, el plan de entrega y las tarjetas CRC (Cargo o Clase, Responsabilidad y Colaboración).

2.1 Propuesta de solución

Para lograr alta disponibilidad de los servicios que brindan los sistemas, se propone la utilización de clústeres. El uso de esta alternativa trae consigo algunas desventajas como son el no tener un control de la instalación de los componentes y el alto costo en tiempo que se debe dedicar actualmente para su puesta en funcionamiento, por lo que se resuelve la implementación de una herramienta multiplataforma donde se pueda integrar la paquetería mínima que se necesita para la creación de un clúster utilizando la propuesta que hace RHCS y así de manera global poder garantizar la alta disponibilidad de sistemas sin tener que incurrir en el empleo de un tiempo excesivo y además controlar que el trabajo final es satisfactorio.

2.2 Fase de planificación

La metodología de desarrollo XP comienza con la fase de planificación. En esta fase los clientes plantean a grandes rasgos las historias de usuarios que son de interés para la primera entrega del producto. Al mismo tiempo, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas historias de usuarios. Los programadores estiman los tiempos de desarrollo sobre la base de esta información.

2.2.1 Historias de Usuario. Requisitos funcionales y no funcionales.

Las Historias de Usuario es la técnica utilizada para especificar los requisitos del software tal y como define la metodología XP. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Para la implementación de la herramienta se definieron cuatro historias de usuario que reúnen a diecisiete requisitos (ver anexo 1) independientes del sistema. Seguidamente se representa gráficamente estos elementos:

Historias de Usuario definidas:

- **Conexión remota usando protocolo ssh.** Donde el usuario deberá especificar los datos de conexión, probar la conexión y establecer la conexión.

- **Configurar parámetros del clúster.** Aquí el usuario deberá seleccionar su nodo activo, los servicios a clusterizar y configurar sus parámetros, además decidir si va a instalar y en qué nodo la herramienta de administración web y también deberá verificar la disponibilidad de los paquetes.

- **Instalar y configurar componentes de software.** Aquí se deberá copiar la paquetería correspondiente a cada nodo y ejecutarla, en caso de fallo restablecer el estado inicial del nodo. Además el usuario podrá configurar los parámetros del clúster y actualizar los mismos antes poner a funcionar el clúster.

- **Consultar ayuda de usuario.** El usuario podrá contar con una ayuda, para facilitar el trabajo con la herramienta y con los clústeres en general y así poder hacer un uso de la misma de manera más rápida.

De estas historias de usuario se extrajeron los requisitos funcionales y no funcionales que modelan las características propias que deben estar presentes en la herramienta. Seguidamente se muestran los requisitos funcionales y no funcionales que describen a la historia de usuario número uno.

Requisitos funcionales extraídos de las historias de usuario

Tabla III. Especificar datos de conexión.

Historia de Usuario Conexión remota usando protocolo ssh.	
Número: 1	Nombre del requisito: Especificar datos de conexión como son: dirección del IP destino, usuario administrador, contraseña del usuario administrador.

Programador: Adrian Turiño León	Iteración Asignada: 1
Prioridad en el negocio: Alta	Tiempo Estimado: 1 semanas
Nivel de Complejidad: Medio	Tiempo Real: 5 días
Descripción: Permitirá al usuario especificar los datos para establecer la conexión con la computadora deseada dentro del clúster.	
Observaciones:	

Tabla IV. Establecer conexión

Historia de Usuario Conexión remota usando protocolo ssh.	
Número: 3	Nombre del requisito: Establecer conexión.
Programador: Adrian Turiño León	Iteración Asignada: 1
Prioridad en el Negocio: Alta	Tiempo Estimado: 2 semana
Nivel de Complejidad: Alto	Tiempo Real: 10 días
Descripción: Permitirá establecer las conexiones a las computadoras por un medio seguro en este caso ssh.	
Observaciones:	

Requisitos no funcionales

Los requisitos no funcionales, son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o Estándares de Calidad. Son propiedades o cualidades que el producto debe tener. (Sommerville, 2005)

Entre los requerimientos no funcionales del sistema propuesto se encuentran los siguientes:

Usabilidad:

La herramienta debe brindar una interfaz sencilla y profesional, sin muchas animaciones o imágenes. Debe ser interactiva, intuitiva y de fácil comprensión para el usuario, facilitando en todo momento la interacción de este con el sistema.

Software:

Para hacer uso del sistema se necesita tener instalada la máquina virtual de java 6.25 en la estación de trabajo del cliente.

Los nodos del clúster deben tener como sistema operativo Red Hat 6.2 o superior, además el usuario deberá contar con un repositorio de dicho sistema operativo para la instalación de componentes de software desde el repositorio.

Seguridad:

La conexión entre el sistema y los nodos del clúster, debe realizarse por medio de un protocolo seguro en este caso se propone ssh para garantizar la seguridad de la herramienta.

Rendimiento:

El sistema debe garantizar un rendimiento adecuado, para optimizar el tiempo de instalación de los nodos. El trabajo con la programación multihilos hace posible lograr este importante atributo.

Hardware:

El clúster deberá contar con al menos dos nodos aunque puede trabajar con hasta cuatro nodos, los cuales deben contar como mínimo 1 GB de memoria RAM y dos interfaces de red, una Ethernet[13] para la comunicación entre los nodos del clúster usando un Switch[14] y otra conexión Ethernet para que los clientes puedan acceder a los servicios del clúster.

2.2.2 Validación de Requisitos

Para la validación de requisitos se utilizan diversas técnicas, generalmente se realiza una reunión con el cliente para revisar los modelos y detectar inconsistencias o errores. La validación de los requisitos es un proceso que tiene como misión principal asegurar que los requisitos definidos son realmente las funcionalidades que el cliente solicitó y que los mismos

[13] Ethernet es un estándar de redes de área local para computadores con acceso al medio por contienda.
[14] Switch: Dispositivo digital lógico de interconexión de redes de computadoras.

tengan una descripción correcta, para lograr un entendimiento al pasar a etapas de diseño e implementación de los mismos. Los requisitos definen realmente qué debe realizar el sistema para satisfacer las necesidades del cliente. (Zamora Abreu, 2010)

Para la validación de los requisitos del sistema se realizaron tres técnicas, las cuales se muestran seguidamente y se expresan los resultados obtenidos a través de las mismas.

Revisión de requisitos

Las revisiones de requisitos consisten en una o varias reuniones planificadas, donde se intenta confirmar que los requisitos poseen los atributos de calidad deseados. Estas reuniones son llevadas a cabo por el analista principal del proyecto y la representación del cliente. El resultado final de las reuniones de revisión es un documento que contiene la lista de defectos localizados y una lista de acciones recomendadas. (Zamora Abreu, 2010)

Luego de la revision de los requisitos del sistema se detectaron errores en cuatro requisitos los cuales fueron corregidos, obteniendose una lista conformada por diescisiete requisitos funcionales, que obtuvieron la confirmación del cliente y del miembro del equipo de trabajo participantes en la revisión.

Estabilidad de los requisitos

El objetivo de esta técnica es medir cuan estables son los requisitos para asegurar su adecuación antes de pasar a la próxima disciplina. Se considera que los requerimientos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. La estabilidad de los requerimientos se calcula como:

$$ETR = \left[\frac{RT - RM}{RT} \right] * 100$$

Donde:

ETR: Estabilidad de los requisitos.

RT: Total de requisitos definidos [17]

RM: Cantidad de requisitos modificados [0]

Luego de aplicada esta técnica a los requisitos identificados de la aplicación arrojo como resultado que la estabilidad de los mismos es de un 100% lo que demuestra que no se están

realizando cambios sobre los requisitos, que son estables y, por tanto, es confiable trabajar el análisis y diseño sobre ellos.

Validación de requisitos mediante prototipos y escenarios

Los prototipos de interfaz de usuario ayudan a identificar, comunicar y mostrar un producto antes de crearlo. Estos ilustran cómo se desea que quede el producto final y pautan la línea a seguir para el desarrollo. Con esta técnica de validación de requisitos se pudo presentar al usuario una imagen que aunque no define la imagen final del sistema, si puntualiza los resultados esperados del sistema y a su vez descritos en los requisitos funcionales.

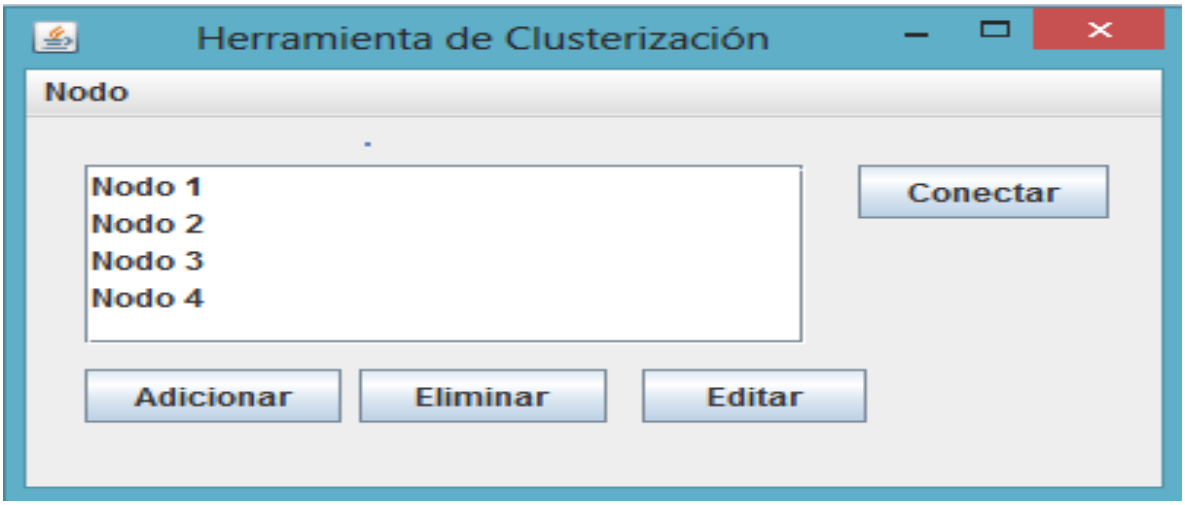

Figura 7. Interfaz principal.

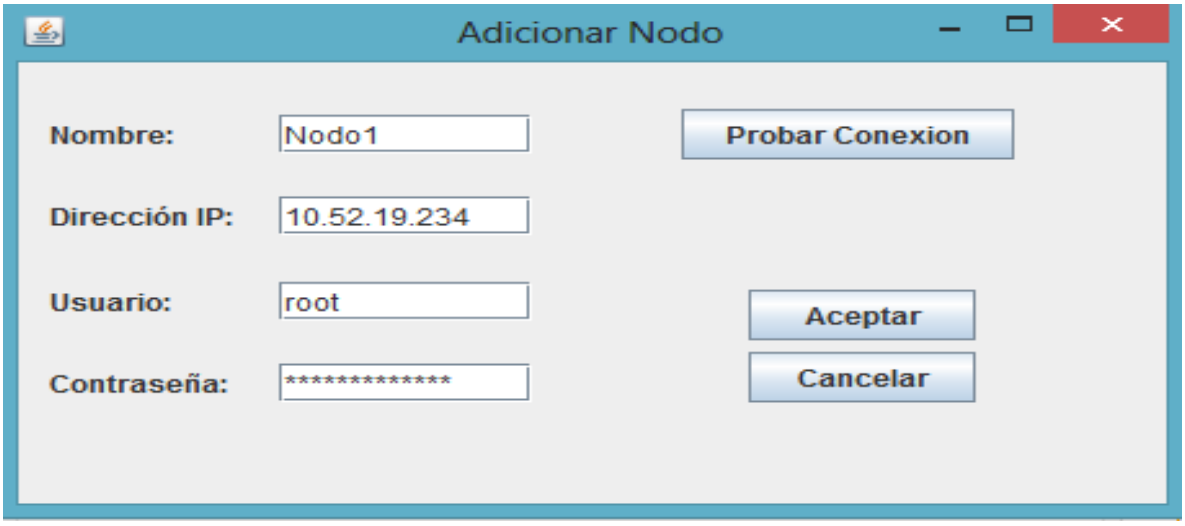

Figura 8. Adicionar nodo.

2.2.3 Actor del sistema

El actor del sistema es un usuario fuera del sistema que interactúa con él. A continuación se muestra su justificación.

- **Usuario del sistema:** El sistema contará con un solo actor el cual podrá acceder a todas las opciones que brinda el sistema como son, establecer conexión con los nodos, definir el nodo activo dentro del clúster, definir los servicios que desea monitorear y los parámetros de los mismos.

2.2.4 Plan de iteraciones

El plan de iteraciones está compuesto por 4 iteraciones, tomando como base cada una de las historias de usuarios y el esfuerzo que se requiere para el desarrollo de estas, se procede a fragmentar el trabajo en distintas iteraciones obteniendo un trabajo incremental. Al final de la última iteración el sistema estará listo para entrar en actividad.

Tabla V. Plan de iteraciones.

Iteraciones	Historias de usuarios	Duración total de iteraciones
Iteración 1	Conexión remota usando protocolo ssh. Donde el usuario deberá especificar los datos de conexión, probar la conexión y establecer la conexión.	2 semanas
Iteración 2	Configurar parámetros del clúster. Aquí el usuario deberá seleccionar el nodo activo, los servicios a clusterizar y configurar sus parámetros, además decidir si va a instalar y en que nodo la herramienta de administración web y también deberá verificar la disponibilidad de los paquetes.	3 semanas
Iteración 3	Instalar y configurar componentes de software. Aquí el sistema deberá copiar la paquetería correspondiente a cada nodo y ejecutarla, en caso de fallo restablecer el estado inicial del nodo. Además el usuario podrá configurar los parámetros del clúster y actualizar los mismos antes poner a funcionar el clúster.	4 semanas

Iteración 4	Consultar ayuda de usuario. El usuario podrá contar con una ayuda, para facilitar el trabajo con la herramienta y con los clústeres en general y así pueda hacer un uso de la misma de manera más rápida.	2 semanas
Total 12 semanas		

Observaciones:

En cada iteración se deberá atender cada uno de los requisitos que responden a la historia de usuario que se esté trabajando sin importar el nivel de prioridad que estos presenten, al final de esta iteración se deberán hacer validaciones del trabajo, para poder pasar a la siguiente iteración. Logrando un trabajo incremental en el menor tiempo posible de desarrollo, en este caso el trabajo fue planificado para un tiempo 12 semanas.

2.2.5 Plan de entregas

El cronograma de entrega establece que las historias de usuario serán agrupadas para conformar una entrega, donde el cliente ordenará y agrupará según sus prioridades las historias de usuarios.

Tabla VI. Plan de entregas.

Iteración	Iteración 1	Iteración 2	Iteración 3	Iteración 4
Cantidad de historias de usuarios	1	1	1	1
Fecha de entrega	15 de marzo de 2013	14 de abril de 2013	15 de mayo de 2013	21 de mayo de 2013

2.3 Fase de diseño

La metodología de desarrollo XP establece prácticas especializadas, que inciden directamente en la realización y elaboración del diseño de un software, sin embargo no requiere que la representación del sistema sea mediante diagramas de clases basados en UML, sino que pueden emplearse indistintamente sencillos esquemas descritos en pizarras u otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). (Letelier, y otros, 2006)

2.3.1 Patrones de Diseño

En la actualidad es muy común el uso de patrones en el desarrollo del software, los mismos proveen a los equipos de desarrollo de un mecanismo que les permitirá diseñar aplicaciones de alta calidad en el menor tiempo posible, debido a que constituyen soluciones a problemas específicos del diseño, promueven la reutilización del código y la agilización del proceso de desarrollo de software.

Un patrón de diseño es una solución a un problema de diseño. Se consideran como procedimientos para solucionar diversos problemas del mismo tipo y son la base para la búsqueda de soluciones a dificultades comunes en el desarrollo de software. (Larman, 2003)

Los patrones de diseño se dividen en dos grupos principales, los patrones GRASP (patrones para la asignación de responsabilidades) y los patrones GOF, estos últimos se clasifican de la siguiente forma:

- Patrones Creacionales: inicialización y configuración de objetos.
- Patrones Estructurales: separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan para formar estructuras más grandes.
- Patrones de Comportamiento: describen la comunicación entre objetos o clases.

Patrones GRASP empleados:

Controlador

El patrón Controlador aumenta el potencial de reutilización y asegura que la lógica de la aplicación no se maneje en la capa de presentación. Un controlador es un objeto que no pertenece a la interfaz de usuario y es responsable de recibir o manejar un evento del sistema.

El desempeño de este patrón consiste en asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

Este patron es evidenciado en el sistema el la clase Cluster.java, clase que controla todo el flujo de eventos entre la capa de presentación, la capa de negocio.

```
35    public class Cluster {
36
37        private LinkedHashSet<Server> servers;
38        private LinkedHashSet<Resource> resources;
39        private LinkedHashSet<Service> services;
40        private final LinkedHashSet<Message> messages;
41        private XML xml;
42        private File localStorage;
```

Figura 9. Fragmento de código de la clase Cluster.

Experto

Consiste en asignar una responsabilidad al experto en información, o sea, a la clase que tiene la información necesaria para realizar la responsabilidad. Un modelo de diseño podría definir miles de clases y una aplicación podría requerir que se realicen gran cantidad de responsabilidades. Si se aplican de la forma correcta, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, existiendo más oportunidades para reutilizar componentes en futuras aplicaciones.

Con el uso de este patrón se mantiene el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas. Es evidente este patrón en la aplicación en la clase Server.java ya que es la responsable de establecer la conexión, instalar y desintalar ademas de otras acciones propias de los servidores.

```
42   public void connect() throws JSchException, UnknownHostException, IOException {...}
47
48   public void disConnect() {...}
51
52   public boolean install(String pathRepo) throws JSchException, IOException {...}
76
77   public boolean unInstall() throws JSchException, IOException {...}
90
91   public boolean installToolAdministration(String pathRepo) throws JSchException, IOException {...}
109
110  public boolean unInstallToolAdministration() throws JSchException, IOException {...}
119
120  public boolean copyConfig(String config) throws JSchException, IOException {...}
128
129  public void removeConfig() throws JSchException, IOException {...}
132
133  private boolean checkInstall(String packages) throws JSchException, IOException {...}
149
150  private boolean checkInstalled(String packages) throws JSchException, IOException {...}
162
163  private String execYum(String options, String packages) throws JSchException, IOException {...}
173
174  private boolean copyRepo(String pathRepo) throws JSchException, IOException {...}
```

Figura 10. Fragmento de código de la clase Server.

Creador

El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creadora. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización.

Este patrón se puede evidenciar en las clases AddServer y AddServices las cuales tienen la responsabilidad de crear los objetos servidor y servicio respectivamente, ya que conocen todos los atributos para la creación de los mismos.

2.3.2 Definición de la Arquitectura

El estilo arquitectónico en capas está compuesto principalmente por una capa de presentación, una de negocio y una de persistencia o acceso a datos, además de una capa de datos y otras capas horizontales o verticales utilizadas en dependencia de las características y necesidades específicas que debe satisfacer el producto.

La capa de presentación es la capa superior en esta arquitectura, representa la interfaz de usuario y es la encargada de interactuar con este. Esta capa es la encargada de la codificación y el control de las interfaces de usuario y de las formas de navegación del usuario estas.

A la capa de negocio se le conoce también como capa intermedia y es donde se localiza la lógica del negocio. Esta es responsable de implementar las reglas del negocio, las validaciones y los cálculos, recibe las peticiones del usuario a través de la capa de presentación y se encarga de darles respuesta. En algunos sistemas esta capa tiene su propia representación interna de las entidades del dominio del negocio, en otros utiliza el modelo definido por la capa de persistencia.

La capa de persistencia no es más que un grupo de clases y de componentes responsables del almacenamiento de los datos. Esta incluye necesariamente un modelo de las entidades del dominio del negocio.

La capa de datos, es la representación real de los datos, y representa además la persistencia del estado del sistema. En esta capa se van a encontrar los datos, o sea la base de datos. Aquí se representan los procedimientos almacenados y el esquema de los datos.

Además cada sistema tiene una infraestructura de clases de ayuda o utilidad, que son usadas por cada capa de la aplicación, estos elementos infraestructurales no forman una capa puesto que no obedecen las reglas para la dependencia de la capa intermediaria de una arquitectura en capas. (Galbán Izquierdo, 2007)

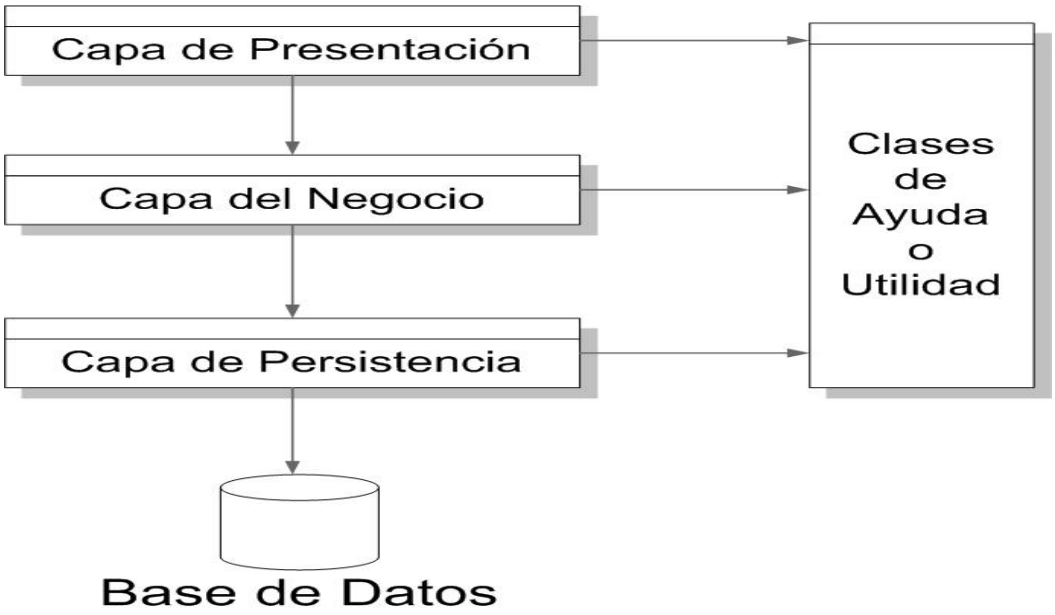

Figura 11. Representación de la arquitectura en tres capas.

Como se puede apreciar el estilo arquitectónico en capas es capaz de organizar todas las clases en diferentes paquetes, estructuras o niveles, lo que ayudaría a mantener la organización de las clases implementadas y además ante un cambio hacer la búsqueda del mismo en menor tiempo y sin tener la necesidad de correr el riesgo de comprometer otras clases. Estas características influenciaron en la decisión de usar esta arquitectura para la construcción de la herramienta, aunque no se necesita hacer uso de todas ellas, pues el sistema no requiere del uso de bases de datos, lo que simplifica el modelo. En la capa presentación se encuentran todas las interfaces de usuario de la herramienta, en la de negocio se maneja todo lo referente a las funcionalidades propias de la herramienta como es la creación de conexiones y la ejecución de acciones remotas y por último la de acceso a datos se encuentra la clase conexión, los registros log y el xml. La capa de clases de ayuda o utilidad contiene clases para el trabajo con ficheros comprimidos, las bibliotecas utilizadas para la gestión remota del proceso o para la presentación de la aplicación de forma independiente del sistema operativo donde se ejecute, entre otras. Gráficamente el modelo arquitectónico queda como se aprecia en la figura 12.

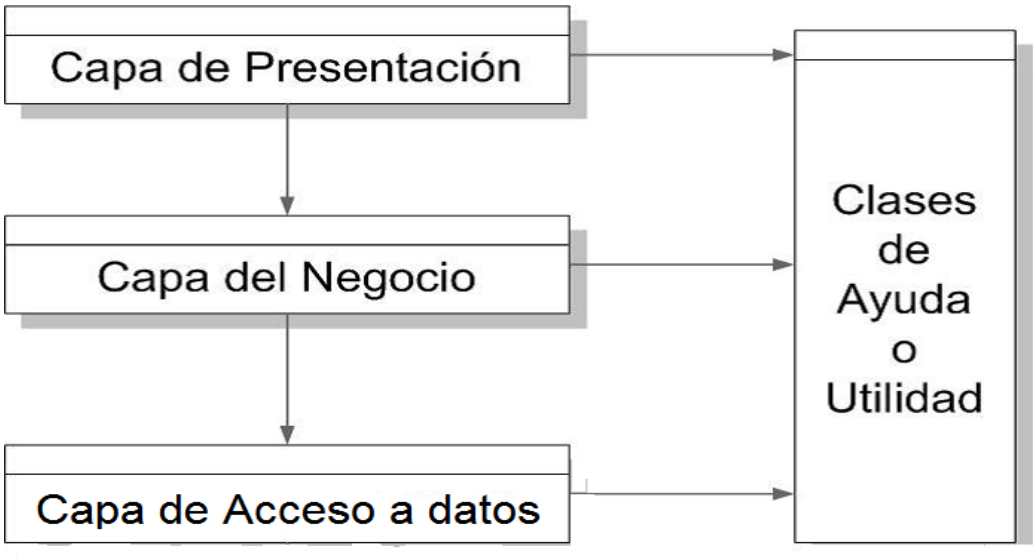

Figura 12. Arquitectura definida para la herramienta.

2.3.3 Diagrama de clases del Diseño

Un diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema y los componentes que se encargarán del funcionamiento y la relación entre uno y otro. El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. A continuación se muestra el diagrama de clases del diseño representando las clases más significativas del sistema.

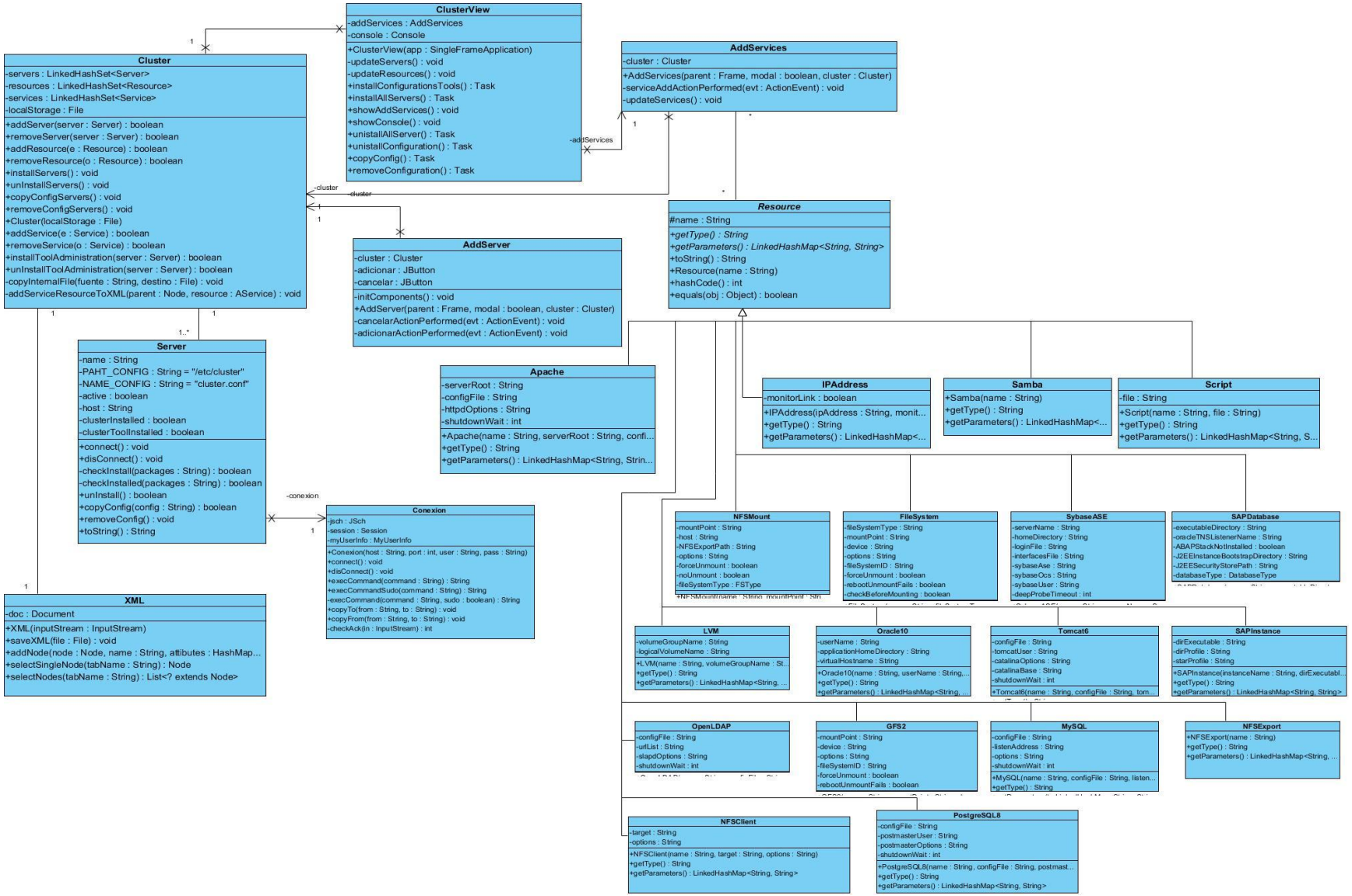

Figura 13. Diagrama de Clases del diseño.

2.3.4 Tarjetas CRC (Cargo o Clase, Responsabilidad y Colaboración)

Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Las tarjetas CRC representan objetos; la clase a la que pertenece el objeto se escribe en la parte de arriba de la tarjeta, a modo de título, en una columna a la izquierda se escriben las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

A continuación se describen las tarjetas CRC correspondientes a la historia de usuario Conexión remota usando protocolo ssh. (Ver anexo 2)

Tabla VII. Gestionar servidores.

Tarjeta CRC	
Clase: ClusterView.java	
Responsabilidades	**Colaboraciones**
Gestionar los servidores. **Gestionar los recursos.**	**AddServer.java** **AddServices.java**

Tabla VIII. Adicionar servidores.

Tarjeta CRC	
Clase: AddServer.java	
Responsabilidades	**Colaboraciones**
Agregar servidores.	**Server.java** **Conexion.java**

Tabla IX. Servidor.

Tarjeta CRC

Clase: Server.java	
Responsabilidades	**Colaboraciones**
Instanciar los servidores.	**Conexion.java**

2.3.5 Validación del diseño

Luego de la realización del diseño se procede a la validación del mismo, mediante la utilización de métricas como Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC).

Tamaño Operacional de Clase (TOC)

Consiste en medir el tamaño general de una clase tomando el valor de la cantidad de operaciones. Si el resultado obtenido indica valores grandes, significa que la clase posee un alto grado de responsabilidad, debido a esto se reducirá la reutilización, se hará mucho más difícil la implementación y la realización de pruebas de dicha clase. Por tanto mientras menor sea el valor para el TOC se hará mucho más fácil la reutilización de dicha clase dentro del sistema.

Tabla X. Relación de números de procedimiento por clase

Clase	Cantidad de funciones
AddServer	2
ClusterAboutBox	1
ClusterApp	4
RadioListRenderer	1
XML	4
Server	19
Cluster	16

OpenLDAP	3
NFSExport	5
NFSClient	5
MySQL	5
IPAddress	5
FileSystem	5
Apache	5
Conexion	11
MyUserInfo	6

Resultado de la aplicación de la métrica Tamaño Operacional de Clase (TOC)

Figura 14. Resultado de aplicar la métrica TOC en el atributo responsabilidad.

Figura 15. Resultado de aplicar la métrica TOC en el atributo complejidad.

Figura 16. Resultado de aplicar la métrica TOC en el atributo reutilización.

Relaciones entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas:

Tabla XI. Relaciones de uso entre clases.

Clase	Cantidad de Relaciones de Uso
AddServer	2
ClusterAboutBox	3
ClusterApp	2
RadioListRenderer	1
XML	1
Server	1
Cluster	1
OpenLDAP	1
NFSExport	1
NFSClient	1
MySQL	1

IPAddress	1
FileSystem	1
Apache	1
Conexion	3
MyUserInfo	2

Resultado de la aplicación de la métrica Relación entre Clases (RC)

La métrica de diseño RC fue aplicada a una muestra de 16 clases seleccionadas aleatoriamente, la **Tabla IX** ilustra la cantidad de dependencias y el porciento que representan de la muestra seleccionada. Las figuras 17,18, 19 y 20 muestran los resultados que arrojó la aplicación de la métrica en los atributos de calidad acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas.

Figura 17. Resultado de aplicar la métrica RC en el atributo acoplamiento.

Figura 18. Resultado de aplicar la métrica RC en el atributo complejidad de mantenimiento.

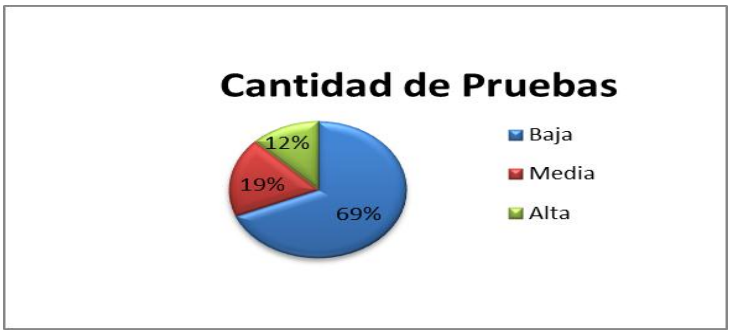

Figura 19. Resultado de aplicar la métrica RC en el atributo cantidad de pruebas.

Figura 20. Resultado de aplicar la métrica RC en el atributo reutilización.

2.4 Implementación

Durante esta etapa se definió el diagrama de despliegue de la aplicación como representación de la distribución de los distintos componentes de hardware y software y el modo de conexión entre ellos. Además se establecieron los estándares a emplear durante la codificación de los requisitos funcionales aprobados, tarea culminante de la etapa.

2.4.1 Diagrama de Despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema, este muestra la configuración de los elementos de hardware y muestra cómo los elementos y artefactos del software se relacionan. A continuación se muestra el diagrama de despliegue de la aplicación, para apoyar la etapa de pruebas a la que será sometido el sistema una vez implementado.

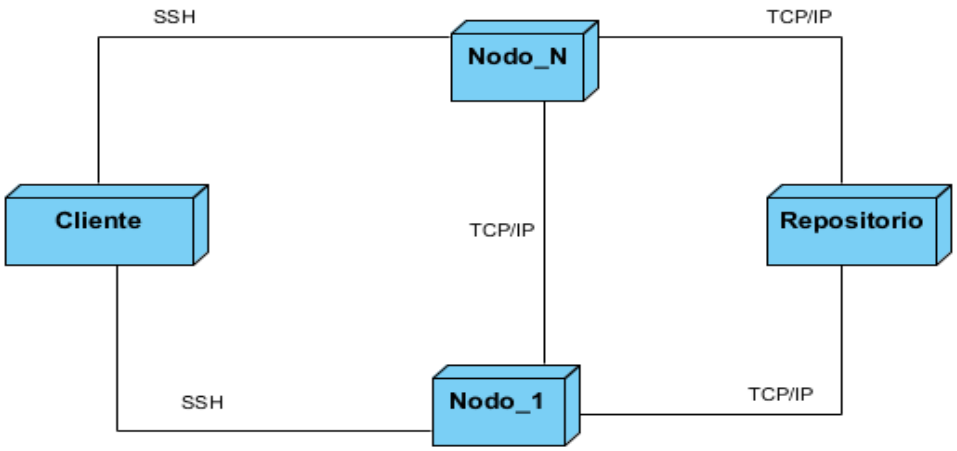

Figura 21. Diagrama de Despliegue.

2.4.2 Estándares de codificación

Los estándares de código son modelos de programación que no están encaminados a la lógica del programa, sino a su organización y aspecto físico con el objetivo de proporcionar la lectura, comprensión y mantenimiento del código durante el proceso de desarrollo del software. Estos estándares de código establecen las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que pueda aumentar su mantenibilidad a lo largo del tiempo.

A continuación se definen 3 partes primordiales dentro de un estándar de programación:

- ✓ Convención de nomenclatura: es la forma de nombrar las variables, funciones y clases.
- ✓ Convenciones de legibilidad de código: es la forma de organizar el código.
- ✓ Convenciones de documentación: es la manera de establecer comentarios, la ayuda, entre otros.

2.4.2.1. Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación *PascalCasing*, esta notación define que los nombres de las variables, métodos, identificadores y clases que están compuestos por varias palabras, donde cada una inicia en mayúscula (Sommerville, 2005). Con sólo leerlo se reconoce el propósito de la misma. Ejemplo: ServiceResource.java.

2.4.2.2. Nomenclatura de las funcionalidades

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación *CamelCasing* (Sommerville, 2005), y con sólo leerlo se reconoce el propósito de la misma. Esta notación es parecida a *PascalCasing,* solo que la primera letra es diferente (minúscula). Ejemplo: installServers

2.4.2.3. Nomenclatura de las variables

El nombre a emplear para las variables se escribe en minúscula. Las variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas. Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales. La siguiente figura muestra un ejemplo de la nomenclatura.

```
public class Cluster {

    private LinkedHashSet<Server> servers;
    private LinkedHashSet<Resource> resources;
    private LinkedHashSet<Service> services;
    private XML xml;
    private File localStorage;
```

Figura 22. Nomenclatura de las variables.

2.4.2.4. Nomenclatura de los comentarios

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

2.4.3 Implementación de las funcionalidades

Esta etapa es crucial en el desarrollo del sistema. A partir de los requisitos aprobados se implementan las funcionalidades que permitan cumplirlos en su totalidad, siguiendo los estándares definidos como una buena práctica de programación y aprovechando el estilo arquitectónico seleccionado.

A continuación se muestra un grupo de figuras que representan algunas de las interfaces gráficas de la aplicación desarrollada.

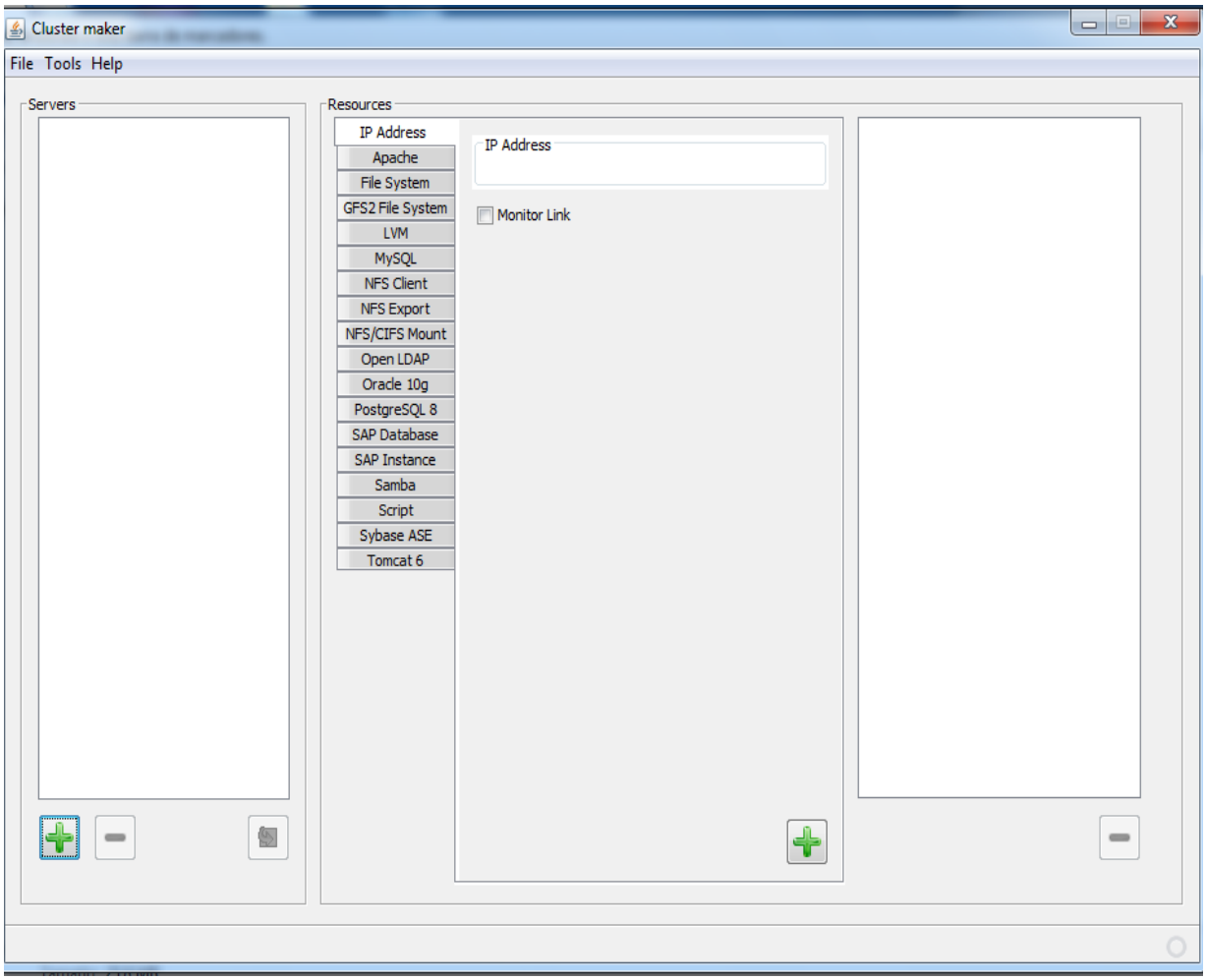

Figura 23. Interfaz principal de la herramienta.

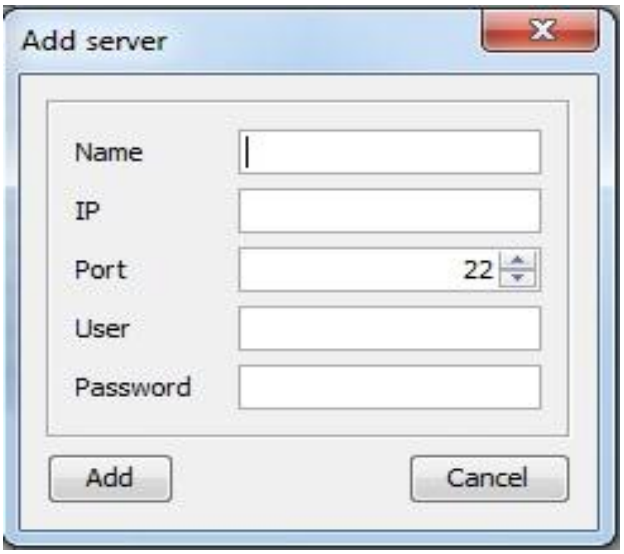

Figura 24. Interfaz para agregar los nodos al clúster.

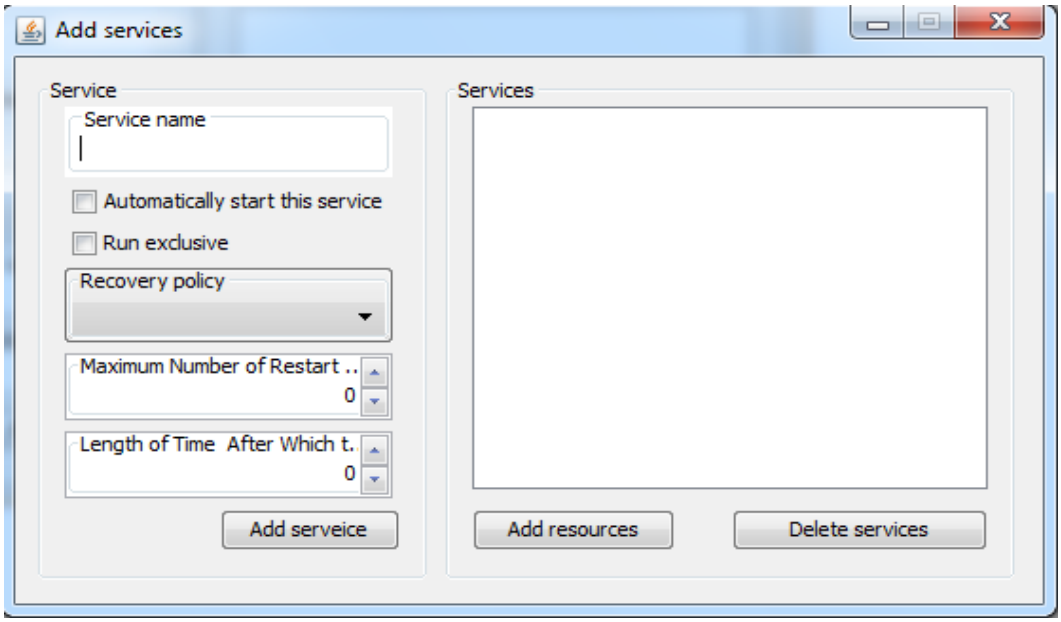

Figura 25. Interfaz para crear los servicios.

2.5 Conclusiones

Siguiendo lo definido por la metodología de desarrollo seleccionada se ejecutaron las tareas de diseño e implementación de la aplicación de manera que se culminaran estas etapas contando con una herramienta funcional cuyo desarrollo está en correspondencia con el estilo arquitectónico establecido y los estándares de codificación definidos. Con esto la aplicación está lista para ser probada.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

Uno de los pilares de la metodología XP es el proceso de pruebas, el cual anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. (J. J. Gutiérrez)

En este capítulo se exponen las validaciones y las pruebas que se le realizan al sistema con el objetivo de comprobar la completitud de la implementación y el cumplimiento de los objetivos planteados.

3.1 Pruebas

En XP se definen las pruebas unitarias y las pruebas de aceptación. Las pruebas unitarias son aquellas que los desarrolladores ejecutan con el objetico de validar que el código sea funcional. Por otra parte, el objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario determinar su aceptación, desde el punto de vista de su funcionalidad y rendimiento. Las pruebas de aceptación son definidas por el usuario del sistema y preparadas por el equipo de desarrollo, aunque la ejecución y aprobación final corresponde al usuario. Una historia de usuario sólo se considerará terminada cuando haya pasado correctamente todas las pruebas de aceptación.

3.1.1 Prueba de Caja Blanca

Durante el proceso de desarrollo de software se hace necesaria la comprobación no solo de la funcionalidad de este, sino también de la eficiencia de su código fuente. Con el objetivo de garantizar esto se realizan las pruebas de caja blanca, las cuales se basan en un examen minucioso de los detalles procedimentales, logrando examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. (Leonard, 2012)

Pruebas Unitarias

JUnit es un marco de trabajo el cual provee un conjunto de clases que permite realizar la ejecución de código Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de las clases se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de

que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

En esta versión del sistema se realizaron pruebas a los algoritmos que componen el código de una de las clases más importantes que interfieren en el funcionamiento del sistema, en esta ocasión todos los resultados obtenidos por JUnit fueron satisfactorios para el sistema.

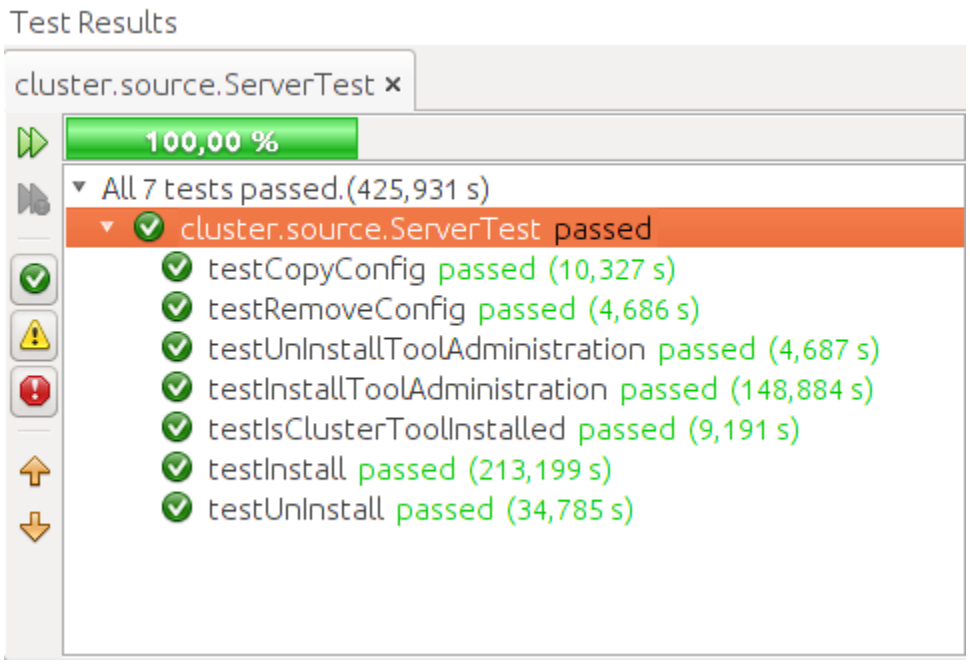

Figura 26. Prueba con Junit.

3.1.2 Pruebas de Caja Negra

Las pruebas de caja negra se centran en los requisitos funcionales, permitiendo al ingeniero del software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. (Leonard, 2012)

Las pruebas de caja negra se llevan a cabo sobre la interfaz de usuario, se centran principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa, se trata de verificar que el sistema “haga lo que tiene que hacer”. Se examinan aspectos del modelo, principalmente del sistema, sin tener en cuenta la estructura interna del software.

3.1.3 Pruebas de aceptación

El uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece sobre la forma que hasta ese momento se realizaba el proceso automatizado. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada prueba de aceptación. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. Para la puesta en práctica de este tipo de pruebas es recomendable no implicar a los miembros del equipo de trabajo, y que preferentemente sea el propio cliente, pues es quien más relacionado está con lo esperado del funcionamiento del sistema.

A continuación se muestran los resultados obtenidos de las pruebas de aceptación realizada al sistema en su versión 1.0, estas pruebas se fueron aplicando a medida que se terminaba cada iteración cumpliendo con uno de los principios que define XP.

Tabla XII. Caso de prueba 1

Caso de Prueba de Aceptación	
Código Caso de Prueba: PA1-HU1	**Nombre Historia de Usuario:** Conexión remota usando protocolo ssh.
Nombre de la persona que realiza la prueba: Juniel Tamayo Hernández	
Descripción de la Prueba: Esta prueba tiene como objetivo verificar si desde la aplicación se puede hacer una conexión remota a los diferentes nodos del clúster, esta conexión se hace de manera individual para cada nodo, especificando los parámetros de conexión: dirección IP del nodo destino, usuario administrador y contraseña del usuario administrador.	
Condiciones de Ejecución: Para la ejecución de esta prueba es necesario que en el nodo este instalado el servidor SSH.	
Escenario número 1	
Entrada / Pasos de ejecución: El usuario desde la interfaz principal decide agregar los nodos a su clúster haciendo clic en el botón adicionar, este evento le muestra otra interfaz	

donde debe ingresar los datos de la conexión. El usuario en este escenario introducirá datos incorrectos en los campos para establecer la conexión.
Resultado Esperado: El sistema debe mostrar un mensaje de error.
Escenario número 2
Entrada / Pasos de ejecución: El usuario desde la interfaz principal decide agregar los nodos a su clúster haciendo clic en el botón adicionar, este evento le muestra otra interfaz donde debe ingresar los datos de la conexión. El usuario en este escenario introducirá en el sistema datos correctos para establecer la conexión.
Resultado Esperado: El sistema debe mostrar un mensaje de conexión satisfactoria, y adicionar el nodo a la lista de nodos del clúster.
Evaluación de la Prueba: Satisfactoria

Los resultados obtenidos después de la realización de esta prueba, en cada escenario de los descritos se muestran en las siguientes figuras.

Resultados del Escenario 1

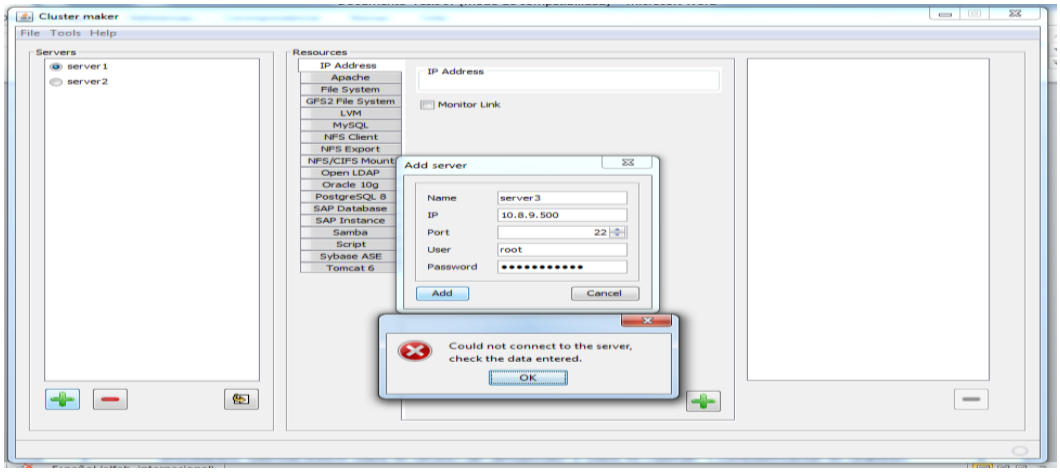

Figura 27. Prueba historia usuario 1 datos erróneos.

En este escenario, se insertaron datos erroneos en diferentes campos como el ejemplo de la figura anterior donde la dirección del nodo no existe, otros datos erroneos probados fueron el usuario de conexión, contraseña y el puerto por el que se debía conectar. En cada uno de estos casos se lograron los resultados esperados tal y como se muestra anteriormente.

Resultados del Escenario 2

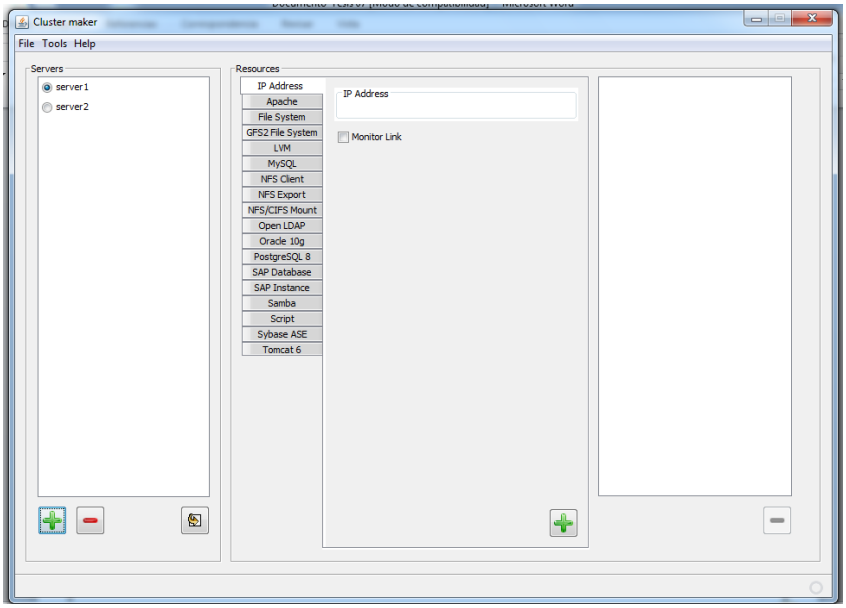

Figura 28. Prueba historia usuario 1 datos correctos.

Para este escenario los datos fueron insertados correctamente. En el panel de la izquierda se puede apreciar como los servidores fueron adicionados a la lista.

Después de añadir los nodos, ya se podrá realizar la instalción de los mismo, en ese tiempo el usuario podrá configurar los recursos que necesite y además controlar el proceso tanto por los mensajes como por las consolas de cada uno de los nodos disponible en el menú de las herramientas. Como se ha podido apreciar los resultados obtenidos hasta el momento son satisfactorios. A continuación se muestran otras tablas e imágenes que pertenecen a las pruebas realizadas a las otras historias de usuario donde se muestra el cumplimiento del objetivo de la herramienta.

Tabla XIII. Caso de prueba 2

Fecha	Versión	Descripción	Autor
<14/04/2013>	<1.0>	Prueba de aceptación para la	<Adrian Turiño León

		historia de usuario número 2	>
Caso de Prueba de Aceptación			
Código Caso de Prueba: PA2-HU2		**Nombre Historia de Usuario:** Configurar parámetros del clúster.	
Nombre de la persona que realiza la prueba: Juniel Tamayo Hernández			
Descripción de la Prueba: En esta prueba el usuario deberá configurar los parámetros de los servicios que decidió monitorear con el clúster, donde se comprobara el tipo da datos insertados en los campos de configuración de los recursos señalado.			
Condiciones de Ejecución: El usuario debe tener seleccionado el recurso para poder configurarlo.			
Escenario número 1			
Entrada / Pasos de ejecución: El usuario selecciona un recurso, automáticamente se debe activar los campos correspondientes para editar los parámetros del recurso señalado. Definidos los recursos podrá conformar los servicios, en el menú de las herramientas del sistema. En este escenario se introducirán tipos de datos incorrectos en los campos de configuración.			
Resultado Esperado: El sistema debe mostrar un mensaje de error.			
Escenario número 2			
Entrada / Pasos de ejecución: El usuario selecciona un recurso, automáticamente se debe activar los campos correspondientes para editar los parámetros del recurso señalado. Definidos los recursos podrá conformar los servicios, en el menú de las herramientas del sistema.			

En este escenario se introducirán tipos de datos correctos en los campos de configuración.
Resultado Esperado: El sistema debe actualizar el panel de la izquierda, por lo que los datos serán evaluados como correctos para el sistema. Una vez terminado este proceso estos datos se guardan en el fichero de configuración para que puedan ser copiados a los nodos.
Evaluación de la Prueba: Satisfactoria

Tabla XIV. Caso de prueba 3

Fecha	**Versión**	**Descripción**	**Autor**
<15/05/2013 >	<1.0>	Prueba de aceptación para la historia de usuario número 3	<Adrian Turiño León >
Caso de Prueba de Aceptación			
Código Caso de Prueba: PA3-HU3		**Nombre Historia de Usuario:** Instalar y configurar componentes de software	
Nombre de la persona que realiza la prueba: Juniel Tamayo Hernández			
Descripción de la Prueba: En esta prueba cuando el usuario este configurando los servicios y sus parámetros, el sistema deberá ir instalando los paquetes de software en los nodos.			
Condiciones de Ejecución: El usuario debe tener por lo menos dos nodos en su lista de nodos agregados al clúster y estos nodos tengan instalado el sistema operativo Red Hat 6 o superior, además deberían tener conexión a un repositorio del mismo sistema operativo aunque no es de carácter obligatorio.			
Escenario número 1			
Entrada / Pasos de ejecución: El usuario escoge la opción para la instalación .En			

este escenario los nodos no tendrán conexión con el repositorio. Por lo que se probará si el sistema es capaz de instalar desde su propio repositorio.
Resultado Esperado: Se espera un mensaje de la instalación satisfactoria y al lado de cada nodo se muestre una figura que identifique que el nodo se instaló.
Escenario número 2
Entrada / Pasos de ejecución El usuario escoge la opción para la instalación .En este escenario los nodos tendrán conexión con el repositorio.
Resultado Esperado: Se espera un mensaje de la instalación satisfactoria y al lado de cada nodo se muestre una figura que identifique que el nodo se instaló.
Evaluación de la Prueba: Satisfactoria

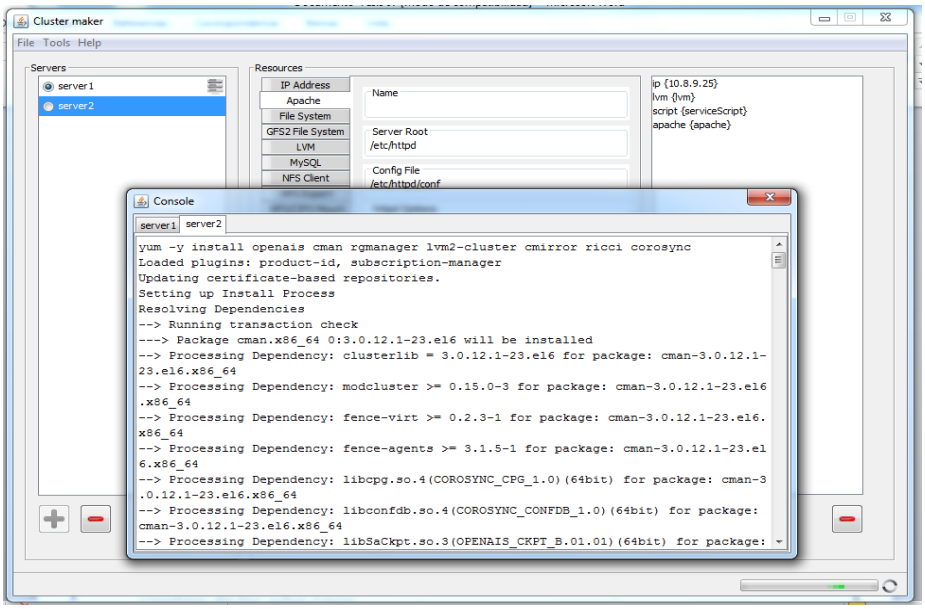

Figura 29. Prueba historia instalar servidores.

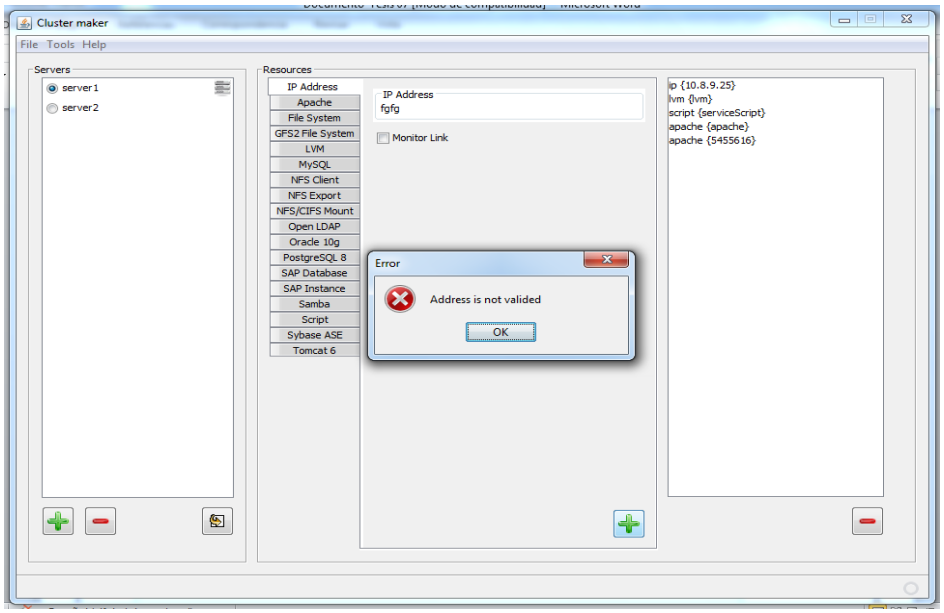

Figura 30. Prueba adicionar recursos (valores incorrectos).

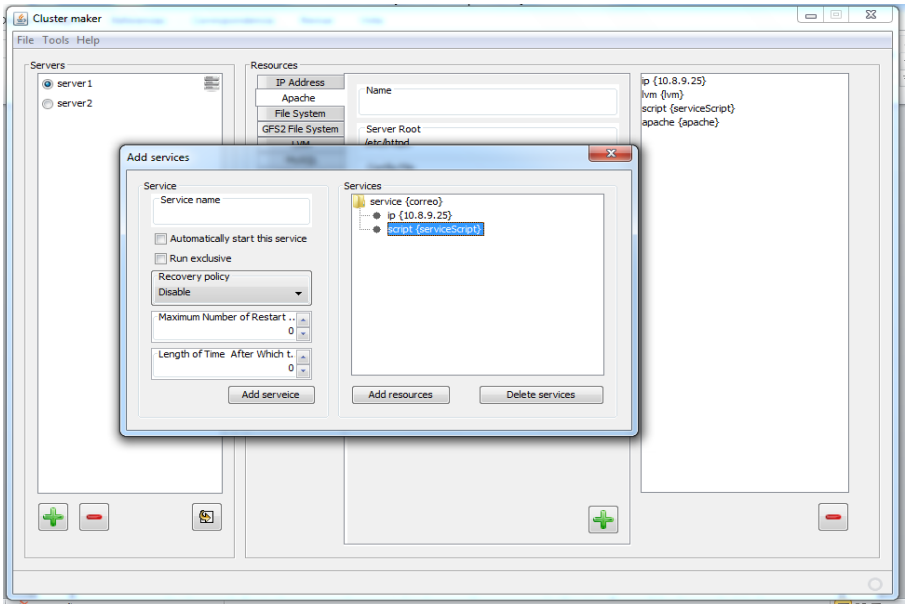

Figura 31. Prueba configurar servicio.

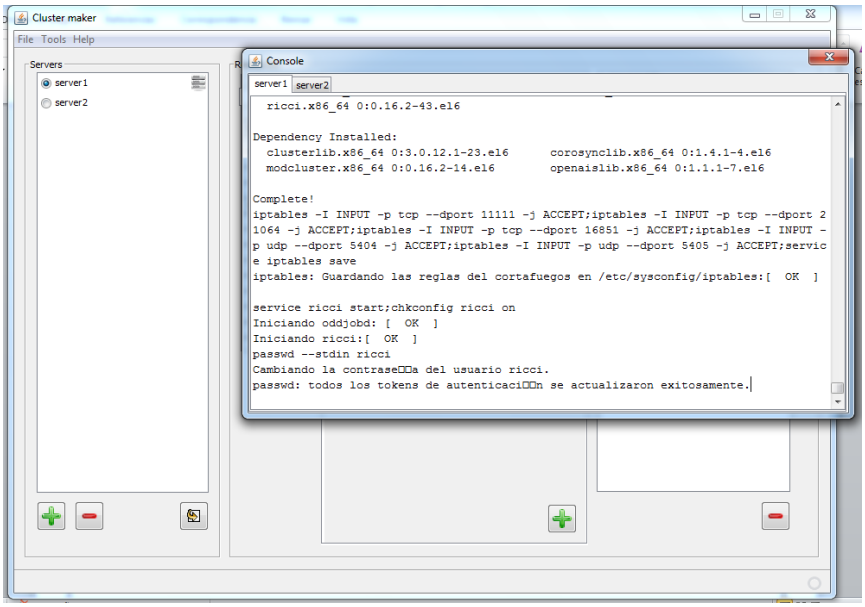

Figura 32. Prueba instalar herramienta de administración (ricci).

Al finalizar este periodo de pruebas el cliente encontró 14 no conformidades (NC) con la herramienta. Entre las que se pueden encontrar por ejemplo:

- Una ventana sin el nombre que la identifica.
- Un mensaje de error en idioma español.
- La herramienta no contaba de un nombre que la identificara.
- No se actualizaba la imagen en la instalación de los servidores.
- La ayuda en el botón de adicionar los nodos no se correspondía.
- Algunas faltas de ortografía.

Todas estas NC fueron de complejidad baja, las cuales fueron corregidas satisfactoriamente en un tiempo relativamente corto gracias a la arquitectura utilizada. Además esta tarea se pudo llevar a cabo sin comprometer el funcionamiento del sistema, ya que ninguna de las NC detectadas era en las funcionalidades.

3.2 Conclusiones

En el desarrollo de este capítulo se pudo obtener una valoración sobre el resultado del trabajo realizado en el desarrollo de la herramienta. Tanto las pruebas unitarias realizadas al código como las pruebas de aceptación realizadas al funcionamiento en general del sistema revelaron resultados satisfactorios para el grupo de desarrollo y para el cliente. De esta manera se puede

afirmar que el componente desarrollado mejora el proceso de instalación y configuración de clústeres de servidores del tipo Activo-Pasivo utilizando la tecnología de Red Hat Cluster Suite.

CONCLUSIONES GENERALES

Se logró establecer el marco teórico de la investigación a través de un estudio de las diferentes herramientas utilizadas actualmente para la instalación y configuración de clústeres de servidores.

Con el análisis y diseño de la herramienta se logró obtener los principales artefactos que tributan a un entendimiento y garantizan el proceso de seguimiento de la misma, así como se realizó la selección de la arquitectura en capas para garantizar una correcta estructuración del sistema.

Se realizó la implementación de una herramienta que permite mejorar el proceso de instalación y configuración de clústeres de servidores del tipo activo-pasivo.

Se aplicaron pruebas de caja negra y caja blanca para la validación del sistema, arrojando resultados positivos en cada caso, demostrándose así el cumplimiento de los objetivos planteados.

RECOMENDACIONES

Hacer un seguimiento del trabajo propuesto con el objetivo de ampliar el campo de acción de la herramienta.

Certificar la herramienta por las entidades competentes de la universidad.

Implementar la internacionalización del producto.

BIBLIOGRAFÍA

Claro Sánchez, Ana Margarita and Rodríguez Abreu, Ariel. 2011. *Módulo "Cuento" del producto Mis Mejores Cuentos .* La Habana : Univercidad de las Ciencias informaticas, 2011.

Corbea, Maite Rodríguez and Pérez, Meylin Ordóñez. 2007. *LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA.* La Habana : UCI, 2007.

Díaz Pérez, Zuzel and González Cárdenas, Adony. 2009. *Sistema Gestor de Trabajos de Diplomas.* Habana : Univercidad de las Ciencias Informaticas, 2009.

Galbán Izquierdo, Yusmary. 2007. *Arquitectura de los Módulos Entrada de Datos y Generador de Modelos.* La Habana : UCI, 2007.

Gómez, Rosa María Yáñez. 2010. *Introducción a las tecnologías de clustering en GNU/Linux.* 2010.

Gutiérrez, Limonta and Dayron. 2012. *Extensión del IDE NetBeans para el desarrollo de aplicaciones empleando el marco de trabajo Sauxe.* La Habana : Univercidad de las Ciencias Informaticas, 2012.

IBM. IBM. [Online] [Cited: Junio 12, 2013.] http://www-03.ibm.com/systems/mx/x/hardware/largescale/cluster/.

Agustín, Candia. 2002. 2002.

Information, Communication and Automation Technologies. **Castaos, I. 2009.** Bilbao, Epaña : s.n., 2009.

J. J. Gutiérrez, M. J. Escalona, M. Mejías,J. Torres. *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.* España : University of Sevilla .

Jagan R, Athreya. 2007. Oracle. [Online] junio 2007. [Cited: junio 13, 2013.] http://www.oracle.com/technetwork/es/documentation/317546-esa.pdf.

Larman, Craig. 2003. *UML y Patrones. Seguna edicion.* Madrid : Pearson Educación. S.A, 2003. 978-84-832-2927-9.

Leonard, Yaima Hernández. 2012. *Evaluación estática del código para la capa de presentación de los proyectos desarrollados bajo tecnología Sauxe en el CEIGE.* La Habana : UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS, 2012.

Letelier, Patricio and Penadés, Mª Carmen. 2006. *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia, 2006.

Marañón, Gonzalo Álvarez. 1999. *¿Qué es Java?* 1999.

Mariño Echemend, Reynaldo and Yandy, González Hervis. 2010. *Diseño e implementación de un plugin de Eclipse que agilice el desarrollo de aplicaciones Web que utilicen la arquitectura única Dalas.* La Habana : Univercidad de las Ciencias Informaticas, 2010.

Martínez Prieto, Ana Belén. 2012. Programar Sencillo . *¿Qué lenguaje de programación escoger?* [Online] Enero 15, 2012. [Cited: junio 11, 2013.] http://programarsencillo.blogspot.com/2012/01/que-lenguaje-de-programacion-escoger.html.

Microsoft. Microsoft . [Online] [Cited: junio 13, 2012.] http://support.microsoft.com/kb/259267/es#appliesto.

Netbeans. 2012. 2012.

Olano Trejos, Germán Andres. 2010. *DISEÑO DE UN CLÚSTER CON COMPUTADORAS DADAS DE BAJA.* s.l. : versidad Tecnológica de Pereira, 2010.

Ortiz Fidalgo, Jackeline and Rios Morales, Fredy Hector. 2010. *Desarrollo de un componente de software para importar reportes desde documentos Excel a Sistemas Gestores de Bases de Datos en el CEINPET.* La Habana : Univercidad de las Ciencias Informaticas, 2010.

Pardillo Castañeda, Martha Karina. 2007. *Referencia para la aplicación de la metodología ágil Programación Extrema a proyectos DESOFT SS.* La habana : Univercidad de las Ciencias Informaticas, 2007.

Quintana Santisteban, Luis Angel. 2008. *ANÁLISIS Y DISEÑO DE UNA HERRAMIENTA PARA MODELACION DE BASES DE DATOS.* La Habana : Univercidad de las Ciencias Informaticas, 2008.

Red Hat. 2007. *Red Hat Cluster Suite Overview provides an overview of Red Hat Cluster Suite for Red Hat En-.* USA : Red Hat, 2007.

Sommerville. 2005. *Ingeniería de Software.* Mexico : DF, 2005.

Vera, Lázaro Campoalegre. 2008. *Herramienta de Generación de Código Mediante Sistema Experto.* Ciudad de la Habana : Univercidad de las Ciencias Informaticas, 2008.

Yusmary Galbán Izquierdo, Ridosbey Milian Iglesias. 2007. *Arquitectura de los Módulos Entrada de Datos y Generador de Modelos.* La habana : UCI, 2007.

Zamora Abreu, Arelis Nelsa. 2010. *Validación de Requisitos con Especialistas Funcionales para Proyectos de Desarrollo de Software en la UCI. .* CIUDAD DE LA HABANA : Universidad de las Ciencias Informáticas, 2010.