
UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1. CENTRO DE SOFTWARE LIBRE


# PERSONALIZACIÓN DE LA DISTRIBUCIÓN DE GNU/LINUX NOVA SERVIDORES PARA ALTA DISPONIBILIDAD

---

Trabajo final presentado en opción al título de

Ingeniero en Ciencias Informáticas

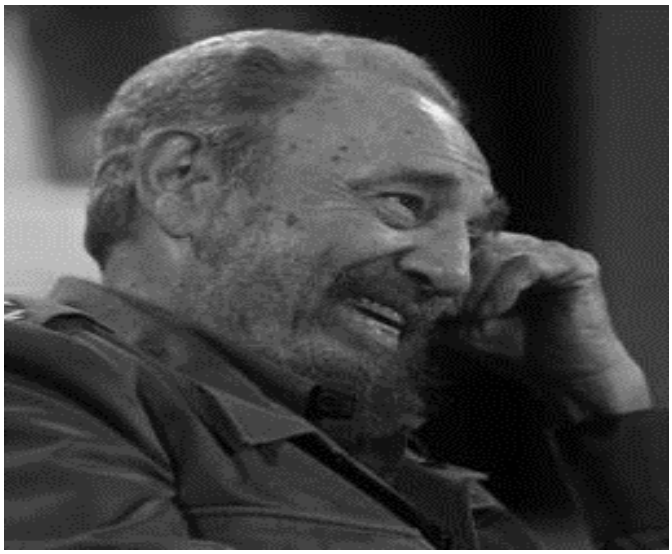
**Autor:** Nestor Llerena Rivera

**Tutores:** Ms. C. Eylín Hernández Luque

Ms. C. Yoandy Pérez Villazón

La Habana, junio 2017

¨Año 59 de la Revolución¨



*¨Y si alguna de las cosas que decimos las explota el enemigo y nos producen profunda vergüenza, ¡¡bienvenida sea la vergüenza!!... ¡¡bienvenida sea la pena!!, si sabemos convertir la vergüenza en fuerza, si sabemos convertir la vergüenza en espíritu de trabajo, si sabemos convertir la vergüenza en dignidad, si sabemos convertir la vergüenza en moral.¨*

Comandante Fidel Castro Ruz

## DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo: Nestor Llerena Rivera, con carné de identidad 93060111983, soy el autor principal del trabajo final de tesis de pregrado que se titula: “Personalización de la distribución de GNU/Linux Nova Servidores para alta disponibilidad”. El cual ha sido desarrollado como parte del trabajo del Centro de Software Libre de la facultad 1 y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los _____ días del mes de _____ del año _____.

_____
**Autor**
Nestor Llerena Rivera

_____
**Tutor**
Ms. C. Eylín Hernández Luque

_____
**Tutor**
Ms. C. Yoandy Pérez Villazón

## DEDICATORIA

*A mi familia por todo su apoyo y comprensión en los momentos más difíciles durante estos cinco años, especialmente a mi mamá, mi papá y hermano que son lo más grande que tengo en esta vida.*

## AGRADECIMIENTOS

*Mi felicidad no estaría completa el día de hoy si me llego a olvidar de esas personas que estuvieron presentes para ayudarme, por eso sería imperdonable dejar de agradecer a todos ellos. Agradezco:*

*A mi madre por sus sacrificios, por ser la luz de mis ojos y la voz de mi conciencia. Por siempre estar ahí sin pedir nada a cambio. Por la preocupación durante todo este tiempo y por siempre estar orgullosa de su hijo.*

*A mi padre por alentarme a continuar creciendo en la vida y aconsejarme a cada momento.*

*A mi hermano por haber venido al mundo a hacerme compañía, brindarme su apoyo y hacer más completa la vida.*

*A mi novia por el apoyo constante en este año y 4 meses, por confiar en mi en todo momento cuando pensaba que no podía. Por su amor inconcidicional , por ser mi amiga y mi compañera inseparable en todo este tiempo.*

*A todos mis amigos y familiares que de alguna forma contribuyeron a materializar este sueño, en especial a mi grupo 1502, nunca me olvidaré de ustedes.*

*A mis tutores por el apoyo y la paciencia brindada durante todo este proceso.*

*Al tribunal por la ayuda y orientación en este último mes.*

*A mis profesores de estos 5 años por ayudar en mi formación como profesional.*

*En general a todas las personas que de una forma u otra hicieron posible cumplir mi sueño de ser ingeniero informático. A todos muchas gracias.*

## RESUMEN

La migración en Cuba hacia plataformas de código abierto, es una de las prioridades del país, por lo que se indicó el desarrollo de un diagnóstico de migración en varios Organismos de la Administración Central del Estado, que permitió obtener información sobre los servidores y servicios telemáticos. En dicho informe, se evidencia una compleja infraestructura de servicios telemáticos, con un alto grado de acceso a los servidores y servicios, así como la necesidad de mantenerlos activos. Los sistemas operativos para servidores, implementan al resolver este problema, mecanismos que garantizan una alta disponibilidad de los servicios. La distribución cubana de GNU/Linux Nova Servidores, actualmente no brinda de forma nativa, el soporte para manejar la alta disponibilidad de los servicios. Tomado como punto de partida esta problemática, se define como objetivo de la investigación desarrollar una personalización de la distribución cubana de GNU/Linux Nova Servidores, mediante herramientas de software libre, que garantice alta disponibilidad en el sistema operativo base, para el proceso de migración en los Organismos de la Administración Central del Estado. En el documento, se muestra la sistematización correspondiente al marco teórico de la investigación, las herramientas y técnicas empleadas, así como los diferentes elementos de la metodología de desarrollo AUP en su variación para la UCI. Los resultados alcanzados contribuyen a una personalización de la distribución de GNU/Linux Nova Servidores y a un módulo clúster para la herramienta Nova-Manager integrado en dicha personalización, con la calidad requerida. La propuesta fue validada por la técnica IADOV.


**Palabras clave:** alta disponibilidad, clúster, Nova Servidores, personalización, sistema operativo.

## ÍNDICE

## ÍNDICE DE ILUSTRACIONES

## ÍNDICE DE TABLAS

## Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) son de suma importancia en la actualidad, ya que permiten aprovechar las nuevas posibilidades didácticas que ofrece este tipo de tecnología. En la actualidad el uso de estas tecnologías ha aumentado considerablemente dado que constituyen un avance considerable en el constante desarrollo del hombre, pues amplía sus capacidades físicas y mentales. El software es un componente fundamental de las TIC este a su vez se clasifica según su creador en privativo o libre de acuerdo a las libertades que tienen los usuarios para trabajar con ellos.

Un programa de software libre según (Stallman, 2004) *"es la libertad de ejecutar el programa sea cual sea el propósito; para modificar el programa para ajustarlo a diferentes necesidades".* Esto significa que los usuarios y la comunidad tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. A veces, el término de software libre se malinterpreta como libertad de precio según la *Free Software Foundation (FSF)* del español Fundación de Software Libre "*el software libre es una cuestión de libertad, no de precio*" (Free Software Foundation, 2016) lo que interesa es la libertad del usuario sobre el mismo.

Dentro del desarrollo de software libre se encuentran, actualmente, un conjunto de proyectos de alta magnitud. Tal es el caso del sistema operativo *GNU/Linux*, cuyo desarrollo es uno de los ejemplos más prominentes del software libre. Todo su código puede ser utilizado, modificado y redistribuido libremente por la comunidad bajo los términos de la Licencia Publica General de *GNU* (*GPL*) y otra serie de licencias libres (Free Software Foundation, 2016).

GNU/Linux, es el término empleado para referirse a la combinación del sistema operativo *GNU*, desarrollado por la *Free Software Fundation*, y el núcleo (*kernel*) Linux, desarrollado por Linus Torvalds y la *Linux Foundation*. La riqueza de GNU/Linux, y la posibilidad de ajustarlo a muy diferentes necesidades y gustos, han favorecido la aparición de las llamadas distribuciones[1]. Actualmente estas distribuciones se han vuelto cada vez más populares entre los usuarios, principalmente por la estabilidad y seguridad que ofrecen. Dentro de las más destacadas se encuentran *Ubuntu*, *Linux Mint*, *PCLinuxOS*, *Slackware Linux*, *Arch Linux*, *FreeBSD*, *openSUSE*, *Fedora*, *Debian GNU/Linux* y *Mageia* (Unsignet Integer Limited, 2017).

[1] **distribución** de software basada en el núcleo Linux que incluye determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios.

En Cuba, con el objetivo de satisfacer la necesidad de migración a plataformas de código abierto, para el logro de la soberanía tecnológica, se desarrolla la Distribución Cubana de GNU/Linux Nova en el Centro de Software Libre (CESOL) de la Facultad 1 de la Universidad de las Ciencias Informáticas. El proyecto Nova provee una línea de productos y servicios de calidad orientado a usuarios nacionales y extranjeros que se desempeñan en el área de las tecnologías de software libre, con el objetivo de alcanzar finalmente un sistema operativo altamente confiable y que cumpla con las necesidades actuales de Cuba.

Nova es una distribución que utiliza el núcleo de Linux e incluye determinados paquetes de aplicaciones informáticas para satisfacer las necesidades de la migración a plataformas de código abierto que experimenta Cuba, como parte del proceso de informatización de la sociedad cubana. Su proceso de construcción, distribución y mantenimiento estará enfocado a alcanzar niveles de excelencia en los siguientes aspectos (Pierra Fuentes, 2011).

- Seguridad: El modelo de desarrollo colaborativo, el acceso al código fuente y el exhaustivo proceso de revisión y auditoría de código garantiza un sistema seguro de virus y sin puertas traseras.
- Soberanía Tecnológica: Mediante la formación de recursos humanos capacitados, será un sistema operativo independiente, con capacidad decisional sobre las tecnologías reutilizadas y desarrolladas.
- Socio-adaptabilidad: Será un sistema operativo hecho por CUBANOS para CUBANOS, alineado a las políticas que orienta la informatización nacional y optimizado para las condiciones tecnológicas del país.
- Sostenibilidad: Mantendrá un proceso flexible y versátil, en constante innovación y consonancia con las nuevas tendencias tecnológicas internacionales, garantizando modelos de comercialización que permitan el ingreso de divisas, por el concepto de exportación de productos y servicios.

La migración en Cuba hacia plataformas de código abierto, es un proceso que retomó fuerza tras el acuerdo del 23 de abril del año 2015, cuando la Comisión de Informatización y Ciberseguridad indicó el desarrollo de un diagnóstico de migración en varios Organismos de la Administración Central del Estado (OACE). El mismo, fue llevado a cabo por el departamento de Servicios Integrales de Migración, Asesoría y Soporte (SIMAYS), y concluyó en diciembre del 2015. Durante este proceso, se obtuvo la información sobre el *hardware* y software de las computadoras institucionales y de los servidores y servicios telemáticos, dicha información dio paso al análisis de la misma y la elaboración de informes de diagnóstico que facilitan el posterior desarrollo de la ejecución del proceso de migración.

Los organismos más grandes que fueron sujetos a este diagnóstico, muestran una compleja infraestructura

de servicios telemáticos con un alto grado de concurrencia en el acceso a los servidores y servicios. La elección del sistema operativo a implantar debe permitir:

- Contar con un mecanismo que permita redundancia en la infraestructura.
- Tolerancia a fallos.
- Automatización del monitoreo de los recursos y las actividades del servidor.
- Disponibilidad y confiabilidad del sistema.

En la actualidad los servicios relacionados con el Internet como son la navegación web, correo electrónico y transferencia de archivos, juegan un papel importante en el desarrollo de las empresas. De esta manera, surge la necesidad de mantener activos estos servicios los siete días de la semana, las veinticuatro horas del día. Para proporcionar un servicio continuo a todos los usuarios que requieran de la utilización de los mismos, y de esta manera agilizar los procesos que dependan de estos servicios en beneficio de la empresa.

Partiendo de que la solución a implantar en los OACE debe ser Nova Servidores se plantea como **problema científico**: ¿Cómo garantizar en la distribución de GNU/Linux Nova Servidores, alta disponibilidad en el sistema operativo base, para el proceso de migración en los Organismos de la Administración Central del Estado?

Se trazó como **objeto de estudio:** alta disponibilidad en sistemas operativos basados en GNU/Linux para servidores. Y como **campo de acción** se plantea: proceso de implementación de alta disponibilidad en la distribución de GNU/Linux Nova Servidores.

Se define como **objetivo general:** Desarrollar una personalización de la distribución de GNU/Linux Nova Servidores, mediante herramientas de software libre, que garantice alta disponibilidad en el sistema operativo base, para el proceso de migración en los Organismos de la Administración Central del Estado.

Para darle cumplimiento al objetivo planteado, se han proyectado los siguientes **objetivos específicos:**

1. Fundamentar las herramientas empleadas en los sistemas basados en la distribución GNU/Linux, para que garantice alta disponibilidad en el sistema operativo base.
2. Desarrollar una personalización de la distribución de GNU/Linux Nova Servidores, que garantice alta disponibilidad en el sistema operativo base, para el proceso de migración en los Organismos de la Administración Central del Estado.

3. Valoración de la solución propuesta, a partir de la simulación de varios entornos de pruebas y mediante la aplicación de un método de consulta a expertos.

El presupuesto hipotético de la investigación se centra en las siguientes **preguntas científicas:**

1. ¿Cuáles son los referentes teóricos que sustentan el desarrollo de la investigación, relacionados con la personalización de una distribución de GNU/Linux Nova Servidores para alta disponibilidad?
2. ¿Qué características cumplen las herramientas, tecnologías, metodologías existentes que permiten la construcción de la personalización de la distribución de GNU/Linux Nova Servidores para alta disponibilidad?
3. ¿Qué elementos deben tenerse en cuenta para realizar el análisis y diseño de la personalización de la distribución de GNU/Linux Nova Servidores para alta disponibilidad?
4. ¿Qué resultados se obtendrán al valorar la personalización de la distribución de GNU/Linux Nova Servidores para alta disponibilidad?

En el proceso investigativo los **métodos teóricos** utilizados fueron:

- El **análisis histórico-lógico:** se empleó con el objetivo de verificar cómo evolucionan teóricamente las aplicaciones empleadas en los sistemas basados en GNU/Linux para lograr una alta disponibilidad y de este modo poder realizar una selección de las técnicas, las herramientas y los algoritmos que se van a utilizar para desarrollar la personalización de Nova servidores que garantice la alta disponibilidad.
- El **inductivo-deductivo**: permitió llegar a proporciones generales a partir de los hechos que conforman la teoría, o a partir de dichas teorías arribar a conclusiones sobre casos particulares que llevaron a la práctica. En la investigación es usado este método para, a partir del análisis de las herramientas existentes para la implementación de un entorno de alta disponibilidad, seleccionar las características que va a poseer la nueva personalización de la distribución de GNU/Linux Nova para Servidores de forma que se garantice la alta disponibilidad.
- La **modelación**: se utilizó para confeccionar los diagramas correspondientes a la representación de la propuesta solución.

Los métodos empíricos utilizados en el proceso investigativo fueron:

- La **Observación**: se utilizó para obtener de forma directa, la información de la realidad objetiva del

comportamiento del proceso de personalización de la distribución GNU/Linux Nova para Servidores, para que garantice alta disponibilidad en el sistema operativo base para el proceso de migración en los Organismos de la Administración Central del Estado.

- El **análisis documental:** se empleó para la revisión de la literatura necesaria durante la investigación sobre el proceso de implantación de entornos de alta disponibilidad para Nova.
- La **tormenta de ideas**: contribuyó para el levantamiento de requisitos, así como para la determinación de las restricciones de software e implementación que deben tenerse en cuenta en el proceso de desarrollo.
- **Consulta a expertos**: se utilizó para evaluar la propuesta que se presenta en la investigación, permitió la valoración de la actualidad, significación de los resultados.

Los **resultados esperados** son:

1. Marco teórico sobre el estado del arte de las herramientas empleadas en los sistemas basados en GNU/Linux, para que garantice alta disponibilidad en el sistema operativo base.
2. Módulo para la herramienta Nova-Manager que permitirá la configuración clúster de alta disponibilidad.
3. Personalización de la distribución de GNU/Linux Nova-Servidor (ISO Nova Servidores), que permita montar un entorno de alta disponibilidad de servidores con configuraciones mínimas.

Para facilitar la comprensión del documento se estructura de la siguiente forma:

En la **Introducción** se realiza la presentación de la investigación, para incluir los antecedentes y la situación problemática, actualidad, importancia y trascendencia del problema planteado. Se precisa el diseño teórico-metodológico, que constituye el punto de referencia para el procesamiento de los resultados y el establecimiento de las correspondientes conclusiones y recomendaciones.

En el **Capítulo uno** se establecen los referentes al proceso de implementación de alta disponibilidad en distribuciones GNU/Linux enfocadas a servidores.

En el **Capítulo dos** se realiza un análisis, que está en correspondencia con las conclusiones del capítulo uno, que fundamenta la solución propuesta. Se exponen los resultados parciales de la aplicación de los métodos empíricos, que son determinantes para identificar la necesidad y comprensión del proceso de personalización de la distribución de GNU/Linux Nova Servidores, para que garantice alta disponibilidad en

el sistema operativo base, para el proceso de migración en los Organismos de la Administración Central del Estado. Por último, se propone el análisis y diseño del módulo para la herramienta Nova-Manager para la creación de un clúster de alta disponibilidad.

En el **Capítulo tres** se presentan los resultados de la validación de la propuesta, a partir de la simulación de varios entornos de pruebas y mediante la aplicación de un método de consulta a expertos.

Posteriormente se muestran las **conclusiones**, **recomendaciones, bibliografía y glosario de términos**; así como los **anexos** que complementan el contenido de la investigación.

# Capítulo 1. Fundamentación Teórica

En este capítulo se realiza una sistematización que permite una mejor comprensión de los fundamentos teóricos sobre las herramientas y técnicas empleadas en los sistemas basados en la distribución GNU/Linux, para que garantice alta disponibilidad en el sistema operativo base. Lo que conlleva a identificar las principales fuentes bibliográficas y áreas de conocimiento que engloba al objeto de estudio y campo de acción.

## 1.2 Sistema operativo

Según la Real Academia de la Lengua Española, un sistema operativo se define como “un programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones”.

En el libro “Introducción a la Informática” en el capítulo uno los autores Alberto Prieto, Antonio Lloris y Juan Carlos Torres definen un sistema operativo como “Conjunto de programas encargado de controlar los recursos del ordenador” (Prieto, y otros, 2002).

Sistema operativo es el software básico que controla una computadora (Contreras Vásquez, y otros, 2007). Dentro de sus funciones principales se encuentran: establecer la comunicación entre el *hardware* y los usuarios/aplicación y ofrecer una interfaz de usuario que permita ejecutar las aplicaciones.

En la actualidad, no todos los sistemas operativo disponibles pueden ser utilizados para clúster, dado que un sistema operativo para clúster debe ser multiusuario y multitarea (Cáceres, 2012). Como ejemplo de sistema operativo, disponible para clúster, la Distribución Cubana de GNU/Linux Nova Servidores.

### 1.2.1 GNU/Linux y GNU/Linux Nova Servidores

El movimiento del software libre tiene como plataforma el sistema operativo abierto y libre GNU/Linux. Las principales características de este sistema operativo GNU/Linux son (Paredes, 2012):

- Multitarea: se pueden realizar varias actividades a la vez (navegar por Internet, editar un documento, compilar un programa, etc.)
- Multiusuario: varios usuarios pueden trabajar concurrentemente en un único ordenador con varios terminales (teclado y monitor) de forma que tengan la sensación de que es el único que está trabajando en el sistema. Cada usuario almacena sus datos (programas, documentos de

texto e imágenes). Se debe notar que para que sea multiusuario es imprescindible que sea multitarea.

- Multiplataforma: se puede instalar en multitud de dispositivos, desde todo tipo de ordenadores de sobremesa y portátiles y servidores hasta videoconsolas o incluso teléfonos móviles.
- Libre: su código fuente está disponible. La comunidad puede usarlo, modificarlo y distribuir.

La Distribución Cubana de GNU/Linux Nova cuenta actualmente con tres variantes orientadas a diferentes escenarios (García Rivas, y otros, 2016):

- Nova-Escritorio: Sistema operativo para ser utilizado en ordenadores con buenas prestaciones (1 GB de memoria RAM o superior). En consonancia con las nuevas tendencias tecnológicas internacionales en lo que a experiencia de usuario se refiere.
- Nova-Ligero: Sistema operativo para ser utilizado en ordenadores de bajas prestaciones menos de 1GB de memoria RAM. Variante idónea para hacerle frente a las políticas de obsolescencia programada.
- Nova Servidores: Sistema operativo para ser utilizado en servidores de datos, correo, aplicaciones, proxy, ftp, bases de datos y otros servicios telemáticos.

El desarrollo de la distribución de GNU/Linux Nova Servidores se inició en el año 2009 a petición del entonces Ministro de la Informática y las Comunicaciones el Comandante de la Revolución Ramiro Valdés Menéndez. Esta distribución se desarrolló con el objetivo de acercar Linux a las pequeñas y medianas empresas y permitirles aprovechar todo su potencial como servidor de empresa. Es la alternativa en código abierto a los productos *de Microsoft (Windows Small Business Server, Windows Server, Microsoft Exchange, Microsoft Forefront)* basado en la distribución *Ubuntu* (Rosales Rosa, 2011). Nova Servidores permite administrar servicios de una red informática, tales como acceso a i*nternet,* la seguridad de la red y la compartición de recursos a través de una única plataforma.

El marco de esta investigación se centra en el desarrollo de una personalización de la distribución de GNU/Linux Nova Servidores para garantizar la alta disponibilidad de los servicios, dado que es la variante de la Distribución Cubana de GNU/Linux Nova para ser utilizada en servidores y la necesidad de mantener los servicios activos para las empresas e instituciones. Cuenta con una interfaz intuitiva que permite realizar tanto funcionalidades de uso frecuente como de configuraciones avanzadas haciendo énfasis en la

usabilidad del sistema. Todas sus funcionalidades están estrechamente integradas entre sí, por lo que permite automatizar la mayoría de sus tareas y ahorrar tiempo en la administración de sistemas.

### 1.2.2 Personalización de una distribución de GNU/Linux

El uso de una distribución de GNU/Linux permite, la posibilidad de emplearlo o modificarlo para que sea utilizado con un determinado uso u objetivo específico, y esto se debe a que se puede contar con su código fuente. A estas modificaciones que se le realizan a una distribución determinada con el fin de suplir necesidades particulares se le denominan: personalizaciones (Obiol González, y otros, 2015).

A partir de la definición brindada por la Real Academia de la Lengua Española a la palabra personalizar como: “Dar carácter personal a algo” y lo antes analizado es posible establecer como personalización de un sistema operativo al conjunto de pasos y modificaciones realizadas a una distribución para agregar, quitar o modificar características al sistema dependiendo de la necesidad y los usuarios finales.

## 1.3 Clúster

Aunque el concepto de clúster en términos informáticos parece sencillo, realmente trae consigo algunas dificultades a la hora de establecer una definición. Según Gregory F. Pfister en el Libro “*In Search of Clusters*” en la página número 72 luego de presentar problemas a la hora de dar una definición única y exacta del término clúster, llega a determinar que un clúster es “un tipo de sistema distribuido o paralelo conformado por una colección de computadoras interconectadas, funcionando como un único recurso de computación unificado” (Pfister, 1998).

De esta misma manera reconocidos expertos en el tema como son Avi Silberschatz y Peter Galvin en su libro “*Operating System Concepts*” definen un clúster como “un conjunto de máquinas y un servidor dedicado, para realizar los relativamente infrecuentes accesos a los recursos de otros procesos” (Avi Silberschatz, 2005).

Un clúster de ordenadores es, básicamente, un sistema distribuido en paralelo que consiste en dos o más servidores interconectados compartiendo sus recursos y que son vistos como si se tratase de uno solo. Esta medida incrementa enormemente la disponibilidad de un sistema, no solo ante fallos, sino también contemplando las necesarias actualizaciones periódicas del sistema que obligan a sacarlos momentáneamente de producción (Vázquez, 2012).

El análisis documental realizado, permitió establecer de manera general y en términos simples la definición de clúster, el cual se entenderá en esta investigación científica como: un grupo de computadoras interconectadas que comparten sus recursos para trabajar por un servicio de manera coordinada, funcionando como un único recurso de computación de manera invisible para el usuario.

### 1.4 Beneficios de la tecnología de clúster

La tecnología de clúster desempeña un papel decisivo en la solución de problemas de las ciencias, la ingeniería y en la ejecución de aplicaciones comerciales. Con el tiempo ha evolucionado para apoyar actividades como: la supercomputación, el comercio electrónico y las bases de datos de alto rendimiento. Entre sus beneficios se encuentran:

- ✓ Incrementar la capacidad de procesamiento.
- ✓ Automatización del monitoreo de los recursos y las actividades del servidor.
- ✓ Disponibilidad y confiabilidad del sistema.
- ✓ Carga de trabajo repartida por los nodos del clúster.
- ✓ Enfrentar cargas de trabajo cada vez mayores, sin dejar por ello de prestar un rendimiento aceptable.

### 1.5 Tipos de Clúster

Los clústeres se clasifican según las características de *hardware* y sistema operativo de sus nodos en: homogéneo *(*mismo *hardware* y sistema operativo*)*, semi-homogéneo *(*diferentes dispositivos de *hardware* pero mismo sistema operativo*)* y heterogéneo (diferente sistemas operativos y *hardware)* (Matos Faure, y otros, 2013-2014).

Así mismo, existen 3 áreas principales en las que los clústeres tienen su campo de acción y es precisamente de allí de donde toman su nombre (Cáceres, 2012).

1. Clústeres de Alto Rendimiento
2. Clústeres de Balanceo de Carga.
3. Clústeres de Alta Disponibilidad.

### 1.5.1 Clúster de Alto Rendimiento

Un clúster de alto rendimiento surge frente a la necesidad que demandan las aplicaciones que necesitan potencia computacional. Persigue proveer a los sistemas con una alta capacidad de ejecución de procesos para el cómputo de grandes volúmenes de datos (Duran, 2009). Entre las áreas más comunes de clústeres de alto rendimiento se encuentran (Luna, 2009): el cálculo del estado del tiempo, la astronomía, la investigación en criptografía, la simulación militar, la simulación de recombinaciones entre moléculas naturales y el análisis de imágenes (Matos Faure, y otros, 2013-2014). Su principal desventaja es el costo de implementación dado que se requiere de un equipo con altas prestaciones.

### 1.5.2 Clúster de Balanceo de Carga

Un clúster de balance de carga (Luna, 2009) es el que distribuye su trabajo a través de múltiples nodos. Todos los nodos son capaces de responder a peticiones externas al ejecutar los mismos programas y en caso de que se presente la falla de un nodo, las peticiones son distribuidas entre los nodos activos restantes. Este tipo de clúster tiene la ventaja de que los nodos pueden encontrarse en distintos puntos geográficos. En este tipo de clúster, repartir el volumen de trabajo en un conjunto de servidores trae como consecuencia la mejora en el tiempo de acceso de los usuarios y la respuesta del sistema. Además, al ser un conjunto de servidores el que recibe las peticiones, la caída de uno de ellos no ocasiona una caída total del sistema (Matos Faure, y otros, 2013-2014). Para la implementación de esta solución se requiere de un nodo adicional para realizar el balanceo de las peticiones por parte de los usuarios por lo que el tiempo de respuestas se limita al rendimiento del nodo balanceador. No todos los servicios pueden ser incorporados en el clúster y todos los nodos deben poseer una configuración homogénea o sea mismo software y mismo *hardware*.

### 1.5.3 Clúster de Alta Disponibilidad

Un clúster de disponibilidad alta es aquel que brinda recursos de forma ininterrumpida y garantiza un funcionamiento correcto de los servicios proporcionando flexibilidad y robustez al sistema. Tradicionalmente la alta disponibilidad ha sido un requerimiento exigido a aquellos sistemas que realizan misiones críticas. Sin embargo, se hace cada vez más importante esta cualidad en sistemas comerciales y en áreas académicas, donde el objetivo es prestar los servicios ininterrumpidamente (Matos Faure, y otros, 2013-2014).

Un clúster que brinde alta disponibilidad se hace indispensable en ambientes de almacenamiento de datos

sensibles o de acceso constante. Los nodos de un clúster de alta disponibilidad se pueden poner fuera de línea, apagar, actualizar o reparar sin comprometer los servicios que brinda el clúster (Matos Faure, y otros, 2013-2014).

El marco de esta investigación se centra en la implementación de un clúster de alta disponibilidad para la distribución de GNU/Linux Nova Servidores, dado que esta configuración garantiza la alta disponibilidad de los servicios, proporcionar flexibilidad y robustez permitiendo que cada servicio sea parte del clúster, con un costo bajo de implementación. Los nodos que conforman el clúster, no necesitan de una configuración homogénea y garantiza que ante un fallo en el servidor principal los servicios sean migrados automáticamente a un servidor secundario sin que se afecte el uso de los usuarios de estos servicios.

### 1.6 Componentes principales de un clúster

Se muestran en la Ilustración 1, los principales componentes que conforman un clúster: aplicaciones, *middleware*, sistema operativo, protocolos de comunicación y servicios y conexiones de red con sus relaciones de interacción y los fundamentos de cada uno.

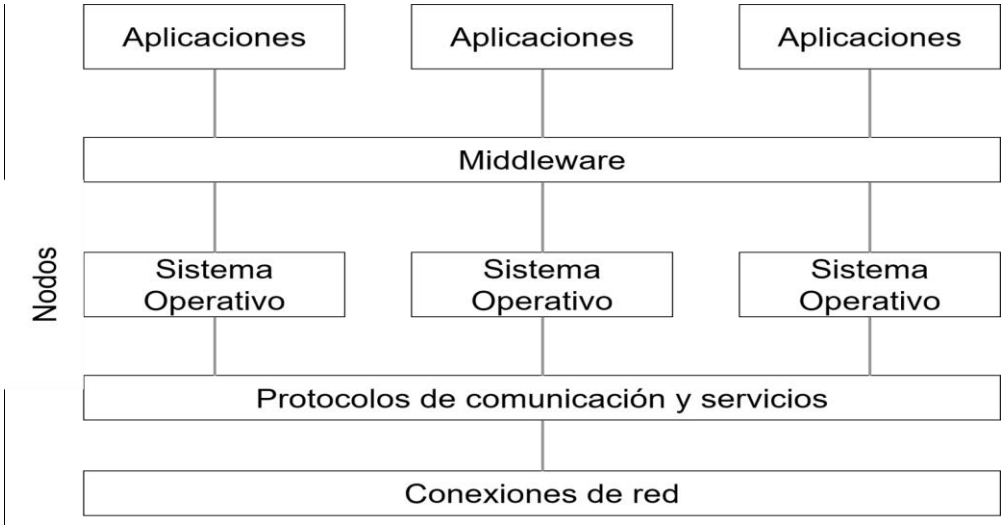



*Ilustración 1: Componentes de un Clúster. Tomado de: (Padilla, 2011).*

#### 1.6.1 Nodos

Un nodo es un computador dentro del clúster, las características de los mismos pueden variar en cuanto a las necesidades para el entorno al que será destinado. Aunque no es estrictamente obligatorio que todos los nodos del clúster sean exactamente iguales es recomendable que estos compartan características

similares de procesamiento y almacenamiento para evitar un comportamiento ineficiente del clúster (Cáceres, 2012).

### 1.6.2 Sistema operativo

Mostrado anteriormente en el presente capítulo en el subepígrafe 1.1 Sistema Operativo.

### 1.6.3 Conexiones de red

Una conexión de red es el medio por el cual, al menos dos computadores, envían y reciben información. Un requisito indispensable para la implementación de un clúster es que todos los nodos sean capaces de comunicarse entre sí a través de una conexión de red. Independientemente de la tecnología seleccionada se recomienda que la velocidad de comunicación entre los nodos sea lo más homogénea posible (Cáceres, 2012).

### 1.6.4 Middleware

En un sistema informático distribuido, *middleware* se define como la capa de software que se encuentra entre el sistema operativo y las aplicaciones en cada sitio del sistema (Sacha Krakowiak, 2007). En otras palabras, es un programa intermediario entre dos programas independientes. El *middleware* es el encargado de monitorear, administrar, distribuir y controlar el comportamiento general del clúster (Cáceres, 2012). Algunos de los *middlewares* de alto nivel más populares para clústeres son:

- Piranha.
- OSCAR.
- OpenMosix.
- Pacemaker.

### 1.6.5 Protocolo de Comunicación y Servicios

Normalmente se cuenta con los protocolos de comunicación *TCP/IP*. Desde el punto de vista de un administrador de sistemas, un servicio o aplicación son piezas de código que se ejecutan en el servidor para cumplir una tarea específica. Sin embargo, desde el punto de vista del clúster, un servicio o aplicación son simples recursos que pueden ser iniciados o detenidos (Cáceres, 2012).

### 1.6.6 Aplicaciones

Son todos aquellos programas que se ejecutan sobre el *middleware*; estos son diseñados para su ejecución en entornos de cómputo paralelo o aprovechamiento del tipo de clúster (Padilla, 2011).

## 1.7 Redundancia y Disponibilidad

El término redundancia se suele interpretar de forma incorrecta como un sinónimo de disponibilidad. Aunque estos conceptos están relacionados, no son equivalentes. La redundancia hace referencia al uso de varios servidores en un entorno con equilibrio de carga para varios fines, como aumentar el rendimiento de la granja de servidores, aumentar la escala para incluir usuarios adicionales y aumentar la disponibilidad (Paredes, 2012).

La disponibilidad es un concepto más especializado que hace referencia a un entorno de varios servidores diseñado para aceptar conexiones y funcionar normalmente incluso si uno o más servidores de la granja no están operativos. Por lo tanto, la disponibilidad implica redundancia e implica además un mecanismo de conmutación por error y algunas otras características posibles. Sin embargo, puede que un sistema redundante no sea de alta disponibilidad (Paredes, 2012).

## 1.8 Alta disponibilidad

En la actualidad las organizaciones dependen cada vez mas de sus sistemas de información, por lo que se desea que estos sean seguros y permanezcan disponibles el mayor tiempo posible. Una interrupción en estos sistemas puede ocasionar efectos como costos directos asociados a la reparación del sistema, pérdida de productividad, pérdida de ingresos. Por lo que la alta disponibilidad de los servicios es una necesidad para cada sistema de información de una organización.

Según el autor Rogel Alfredo Miguez Paredes en su trabajo de investigación "*Desarrollo de una infraestructura de redundancia para servidores Proxy GNU/LINUX en la intranet de la Facultad de Ciencias*" define la alta disponibilidad como "*un sistema cuya implementación asociada asegure un cierto grado absoluto de continuidad operacional durante un período de tiempo dado*" (Miguel, 2012).

La disponibilidad se basa en un modelo matemático que provee el grado en que un sistema de software permanece en una condición operable (Cáceres, 2012).

$$A = \frac{MTTF}{\mathrm{MTTF} + \mathrm{MTTR}}$$

MTTF = Es el tiempo promedio entre fallos.

MTTR = Es el tiempo promedio que tomar reparar el sistema después de un fallo.

Existen 2 métodos para alcanzar una alta disponibilidad.

- ❖ Incrementar el valor del MTTF
- ❖ Reducir el valor del MTTR

### 1.9 Modelos de implementación de un clúster

Los clústeres de alta disponibilidad requieren de al menos dos nodos para proveer redundancia, sin embargo, muchas implementaciones pueden tener n nodos. Tomando en cuenta esta característica los modelos de implementación de un clúster pueden variar de la siguiente manera (Cáceres, 2012).

- Activo/Activo: Este modelo también se conoce como simétrico y consiste en reenviar el tráfico dirigido al nodo con problemas a otro nodo del clúster. Para implementar este modelo es necesario que todos los nodos tengan una configuración homogénea.
- Activo/Pasivo: Este modelo también se conoce como asimétrico. Cada nodo tiene un nodo de respaldo. Un clúster de alta disponibilidad en una configuración activo/pasivo, consiste en un servidor que posee los recursos del clúster y otros servidores que son capaces de acceder a esos recursos, pero no los activan hasta que el propietario de los recursos ya no esté disponible. Las ventajas de la configuración activo/pasivo son que no hay degradación de servicio y que los servicios sólo se reinician cuando el servidor activo deja de responder.
- 1-a-N: Un solo nodo del clúster provee todos los servicios y cuando esta falla, esos servicios son transferidos temporalmente al resto de nodos que normalmente son nodos más pequeños o con menos recursos.
- N-a-1: Solo un nodo del clúster toma temporalmente el lugar del nodo con problemas.
- N-a-N: Varios o todos los nodos del clúster proveen servicios y cuando uno de ellos falla, sus servicios se transfieren temporalmente a cualquier otro nodo.

Analizado los diferentes modelos de implementación de un clúster se decidió por parte del autor de esta investigación científica utilizar para el desarrollo de la solución el modelo Activo/Pasivo. Dado que es la configuración utilizada para la implementación de clúster de alta disponibilidad. Garantiza que el servicio

permanezca activo para los usuarios ya que estos pueden ser migrados al ocurrir un fallo del servidor principal hacia un servidor secundario y de esta manera garantizar continuidad operacional de los mismos.

### 1.10 Otras definiciones asociadas a la investigación

A continuación, se muestran un conjunto de definiciones asociadas a la investigación como son: granja de servidores, *failover*, membrecía y mensajería.

#### 1.10.1 Granja de servidores del inglés (*server farm*)

Una granja de servidores es un grupo de servidores, normalmente mantenidos por una empresa o universidad para ejecutar tareas que van más allá de la capacidad de una sola máquina corriente, como alternativa generalmente más económica, a un superordenador (Paredes, 2012). Se muestra en la Ilustración 2, la estructura de una granja de servidores donde se pueden apreciar los componentes informáticos y los servicios que pueden ser brindado por el mismo.

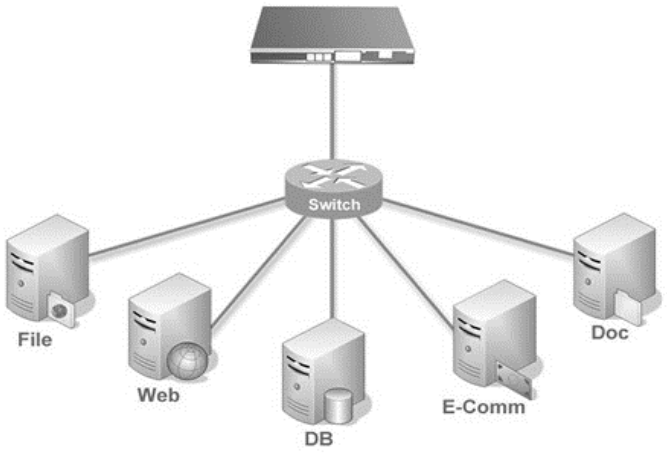



*Ilustración 2: Ejemplo de granja de servidores. Tomado de: (AceNet Technology, 2009).*

#### 1.10.2 Failover

Se conoce como *Failover* o conmutación por error al proceso de migración de un servicio o aplicación desde un nodo del clúster a otro. Este proceso se puede realizar manualmente, sin embargo, en condiciones ideales se realiza automáticamente. La personalización de la distribución de GNU/Linux Nova Servidores para alta disponibilidad permitirá que este proceso sea realizado automáticamente tras ocurrir un fallo en el

servidor principal y de manera manual para realizar actualizaciones o la desconexión de un nodo, migrando los servicios hacia otro nodo del clúster.

### 1.10.3 Membrecía

Proceso mediante el cual un nodo obtiene el acceso a un clúster. Este proceso debe ser realizado de forma segura por lo que la solución constará con mecanismos para la autenticación de los nodos del clúster, mediante clave de autenticación encriptada sin la cual un nodo no pasará a formar parte del clúster.

### 1.10.4 Mensajería

La mensajería es el proceso mediante el cual los nodos se comunican entre sí dentro de un clúster. Mediante este proceso se detecta cuando un nodo deja de estar en una condición operable.

## 1.11 Ejemplos de Sistemas de Clúster Implementados para GNU/Linux

### Beowulf

En el verano de 1994, Thomas Sterling y Don Becker, trabajando en el *CESDIS* (*Center of Excellence in Space Data and Information Sciences*) bajo la tutela del proyecto *ESS* (*Earth and Space Sciences*), construyeron un clúster computacional consistente en procesadores de tipo x86 comerciales conectados por una red *Ethernet* de 10Mb. Llamaron a su máquina *Beowulf* (Hurtado Cuellar, 2011).

Un clúster de tipo *Beowulf* es una colección de computadoras personales interconectadas por medio de una red privada de alta velocidad, corriendo un sistema operativo libre. Los nodos en el clúster están dedicados exclusivamente a ejecutar tareas asignadas al clúster (Hurtado Cuellar, 2011).

### MareNostrum

En julio de 2004 se creó el Centro de Supercomputación de Barcelona (*BSC*) de la Universidad Politécnica de Cataluña, España. El *BSC* creó el clúster *MareNostrum* el cual, en noviembre de 2004, se ubicó en el Top 500 como el primer clúster más veloz y el cuarto sistema más rápido del mundo; sin embargo, para julio de 2005 se ubicó en la quinta posición. Está conformado por 3.564 procesadores PowerPC970 de 2.2 GHz, utiliza una red *Myrinet*[2] y su rendimiento es de 20.53 *TFlops* (Giancarlo Ruiz, y otros, 2012).

[2] Myrinet es una red de interconexión de clúster de altas prestaciones

El marco de esta investigación se centra en el desarrollo de una personalización de la distribución de GNU/Linux Nova Servidores. La cual permitirá la administración y configuración de un clúster de alta disponibilidad de tipo *Beowulf,* que cosiste en un nodo principal el cual será el encargado de ofrecer los servicios y uno o varios nodos esclavos, para retomar estos servicios en caso de ocurrir un fallo en el servidor principal, utilizando el modelo de implementación Activo/Pasivo.

## 1.12 Metodologías y herramientas para el desarrollo de la personalización de la distribución de GNU/Linux Nova servidores para alta disponibilidad.

Para el desarrollo de la personalización de la distribución de GNU/Linux Nova Servidores y el módulo clúster para la herramienta Nova-Manager se hace necesario el uso de tecnologías de software libre como *Corosync* para la comunicación entre los nodos del clúster, *Pacemaker* como administrador de recursos del clúster, *Genisoimage* para la creación del ISO, *Python* como lenguaje de programación, *PyCharm* como Entorno de Desarrollo Integrado, *Visual Paradigm* para UML como herramienta CASE y la metodología de desarrollo AUP versión UCI.

### 1.12.1 Metodología

El desarrollo del software depende de un sin número de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el éxito del producto. El papel preponderante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo (G Figueroa, y otros, 2007).

Existes varias clasificaciones para estas metodologías dentro de las cuales se destacan (G Figueroa, y otros, 2007):

- **Metodologías tradicionales:** Conocidas también como formales se enfocan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones). Otra de las características de este enfoque es el alto costo de realizar un cambio y al no ofrecer una buena solución a proyectos donde el entorno es volátil.
- **Metodologías ágiles:** Utilizadas para dar respuesta a los problemas derivados de las metodologías tradicionales.

El autor de la presente investigación decidió utilizar una metodología ágil para aprovechar las facilidades que brinda en cuanto a requerir poco cúmulo de documentación, adaptable fácilmente a un entorno cambiante y se cuenta con poco tiempo para el desarrollo; estas facilidades son de gran importancia porque para la implementación del sistema se cuenta con un solo desarrollador, por lo que es más importante una obtención adecuada y funcional del sistema que lo documentado que este pueda estar. Para una mejor comprensión en la Tabla 1 se muestra una comparación entre ambas metodologías.

*Tabla 1: Comparación entre Metodologías Tradicionales y Metodologías Ágiles. Tomado de: (G Figueroa, y otros, 2007).*

| Metodologías Tradicionales | Metodologías Ágiles |
|---|---|
| Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo | Basadas en heurísticas provenientes de prácticas de producción de código |
| Cierta resistencia a los cambios | Especialmente preparados para cambios durante el proyecto |
| Impuestas externamente | Impuestas internamente (por el equipo) |
| Proceso mucho más controlado, con numerosas políticas/normas | Proceso menos controlado, con pocos principios. |
| El cliente interactúa con el equipo de desarrollo mediante reuniones | El cliente es parte del equipo de desarrollo |
| Más artefactos | Pocos artefactos |
| Más roles | Pocos roles |
| Grupos grandes y posiblemente distribuidos | Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio |
| La arquitectura del software es esencial y se expresa mediante modelos | Menos énfasis en la arquitectura del software |
| Existe un contrato prefijado | No existe contrato tradicional o al menos es bastante flexible |

Dentro de las metodologías de desarrollo ágil como son XP(*Extreme Programming*) , OpenUp(*Open Unified Process*), Scrum , AUP(*Agile_Unified Process*) se decidió utilizar la metodología AUP versión UCI, es una metodología ágil y una variación de la metodología AUP. Dado que no existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos) se decide en la Universidad de las Ciencias Informáticas (UCI) hacer esta variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Es una metodología que se apoya en el Modelo CMMI-DEV v1.3 el cual constituye una guía para

aplicar las mejores prácticas en una entidad desarrolladora, centradas en el desarrollo de productos y servicios de calidad.

Dicha metodología cuenta con 3 fases y 7 disciplinas (Sánchez, 2015):

**Fases:** Inicio, Ejecución y Cierre.

**Disciplinas:** Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas interna, Pruebas de liberación, Pruebas de Aceptación, áreas de procesos de gestión y soporte que propone DEV v1.3 (Planeación de proyecto, Monitoreo y control de proyecto y Gestión de configuración).

### 1.12.2 Herramienta para la creación del ISO

***Genisoimage*** *(versión 1.1.11)*

Antes llamado mkisofs; cambió de nombre a partir de la versión 4 *Etch* de *Debian*. Acrónimo de *Generate ISO Image*, es un programa para crear imágenes de sistemas de archivos ISO-9660 para CD-ROM, que pueden grabarse en CD o DVD. Es capaz de generar el protocolo de sistema de uso compartido de registros (SUSP), especificados por el protocolo de intercambio *Rock Ridge*. Esto se utiliza para describir aún más los ficheros en el sistema de archivos ISO-9660 a un *Host UNIX* y proporciona información como nombres de archivos largos, permisos POSIX[3], enlaces simbólicos y dispositivo de bloque. *Genisoimage* incluye herramientas útiles para trabajar con las imágenes ISO:

- **mkzftree**, crea la imagen ISO-9660 con contenidos comprimidos.
- **dirsplit**, separa fácilmente grandes contenidos de un directorio en discos de un tamaño predefinido

### 1.12.3 Herramientas para la alta disponibilidad

***Heartbeat*** *(versión 3.0.0)*

*Heartbeat* es un producto del proyecto *Linux High Availability* que persigue proporcionar una herramienta que garantice la fiabilidad, disponibilidad y calidad de servicio a nivel de mensajes entre los nodos del clúster. Es exclusivo para plataformas *GNU/Linux* bajo licencia *GPL*. Esta herramienta permite implementar clústeres de control descentralizado *(*más de un maestro*)* siendo estable, flexible y eficiente. Requiere dos

[3] Interfaz Portable de Sistema Operativo. Permite asignar permisos, o derechos de acceso, a los archivos para determinados usuarios o grupos.

máquinas como mínimo para su implementación y su unidad de trabajo es el recurso *(por ejemplo: IP)*. *Heartbeat* monitorea el nodo para el que fue configurado y es capaz de detectar su caída. Cuando transcurre un tiempo y el servidor maestro no responde, *Heartbeat* determina que está fuera de servicio o inactivo y automáticamente activa el servidor secundario que pasa a ser maestro asumiendo las peticiones de los clientes (Matos Faure, et al., 2013-2014).

***Corosync*** *(versión 2.3.5)*

*Corosync* es un proyecto de código abierto bajo la nueva licencia *BSD* y derivado del proyecto *OpenAIS*. La misión de *Corosync* es desarrollar una solución de alta disponibilidad para clústeres tanto comerciales como de código abierto.

*Corosync* proporciona una capa de comunicaciones fiable para clústeres de alta disponibilidad. Se asegura de que los nodos del clúster pueden enviar y recibir mensajes a través de múltiples vías de comunicación redundantes. *Corosync* es compatible con múltiples medios de transporte, tales como la multidifusión *UDP*, *UDP unicast* y *InfiniBand* nativo (Matos Faure, et al., 2013-2014).

*Corosync* es una versión derivada de OpenAIS y está respaldado tanto por *Red Hat* como por *Novell/Suse*. Por otro lado contiene los protocolos de red necesarios para la intercomunicación de los procesos y para la implementación general del clúster de alta disponibilidad sin que tenga dependencia con alguna distribución específica de Linux (Cáceres, 2012).

Algunas de las características principales de *Corosync* son:

- La tecnología usada por *Corosync* se basa en más de 20 años de investigación en computación distribuida.
- Soporte para encriptación y autenticación para una comunicación segura.
- Soportado por la mayoría de distribuciones de GNU/Linux.
- Diseño compacto con menos de 60 mil líneas de código.
- Diseñado optimizado para minimizar las copias a memoria y los intercambios de contexto.
- Alto rendimiento en redes *Ethernet* y 10G *InfiniBand*.
- Disponible para Linux, Solaris, BSD, Darwin.

*Corosync* aborda la alta disponibilidad asegurando que cada servidor redundante en el sistema mantiene una copia redundante de información que se usa para tomar decisiones acerca de las aplicaciones. Por otro

lado, en lugar de enviar llamadas a funciones como se hace en un enfoque de estado de máquina típico, Corosync envía estados de máquina a todos los nodos del clúster de manera ordenada y consistente (Cáceres, 2012).

*Corosync* es un proyecto que da continuidad a *Heartbeat*. Dado que la comunidad ha decidido darle más soporte y mantenimiento a *Corosync* incluyéndole un conjunto de funcionalidades mayor a las que posee actualmente *Heartbeat*. Luego del análisis realizado se selecciona la herramienta *Corosync* como herramienta para la comunicación segura entre los nodos del clúster.

### 1.12.4 Middleware

***Pacemaker*** *(versión 1.1.14)*

*Pacemaker* es un administrador de recursos del clúster orientado a obtener la máxima disponibilidad de sus recursos mediante la detección, recuperación de nodos y los fallos a nivel de recursos, haciendo uso de las capacidades de mensajería y la pertenencia a la infraestructura de clúster proporcionada por *Corosync* o *Heartbeat*. Independiente de la distribución de GNU/Linux (Rubio Sapiña, 2012).

Puede hacer esto para los clústeres de prácticamente cualquier tamaño y viene con un modelo de dependencia de gran alcance que permite al administrador expresar exactamente las relaciones (ordenar y localización) entre los recursos del clúster. Prácticamente todo lo que se puede programar puede ser administrado como parte de un clúster mediante *Pacemaker*.

Entre las principales características destacan:

- Detección y recuperación de errores de nodo y a nivel de servicio.
- Independiente de almacenamiento, sin necesidad de almacenamiento compartido.
- Independiente de recursos.
- Soporta grandes y pequeños clúster.
- Soporta prácticamente cualquier configuración de redundancia.
- La actualización de la configuración se replica automáticamente desde cualquier nodo.
- Mecanismo para determinar si el clúster puede o no continuar ofreciendo los servicios.
- Soporte para especificar las aplicaciones que deben ejecutarse en un mismo nodo y el orden de ejecución.
- Máximo de nodos no establecidos.

***Piranha***

Una de las herramientas más usadas es *Piranha* de la empresa *Red Hat*, que incorpora balance de carga mediante direcciones IP y también hace la inclusión de código del proyecto *Linux Virtual Server*, en esta herramienta *Red Hat* incorporó mejoras; para poder hacer uso eficiente de *Piranha* es recomendado el uso de *Red Hat Enterprise Linux* 4 o posterior, su sencillez en instalación y amplio soporte por parte de dicha empresa brinda satisfacción al hacer una implementación con esta. Fuera del pago de la licencia para el sistema operativo el software de *Piranha* está disponible de manera gratuita para usuarios registrados de esta distribución (Padilla, 2011).

Un clúster *Piranha* se compone de los siguientes elementos:

- El parche *IPVS* para el *kernel.*
- El demonio *LVS* para manejar las tablas *IPVS* a través de *ipvsadm*.
- El demonio nanny para monitorizar servicios y servidores.
- El demonio pulse para controlar el estado del resto de demonios del clúster y la entrada en funcionamiento del nodo de balance de carga de respaldo en caso de fallo del primario.
- La interfaz gráfica *Piranha* para administrar el clúster.

***Oscar***

La herramienta *Oscar* es una colección de software de código abierto para crear un clúster sobre GNU/Linux desarrollada por el Grupo de Clúster Abiertos (*OCG –Open Clúster Group*). El objetivo primario del grupo de trabajo *Oscar* es conseguir que la instalación, la configuración y la administración de un clúster de tamaño modesto, sea fácil. Cualquier cosa necesaria para un clúster como instalación y configuración, mantenimiento, programación (paralela), sistemas de encolamiento, programación de tareas, está incluida en *Oscar*. Su principal labor es para cómputo de alto rendimiento (Padilla, 2011).

***OpenMosix***

*OpenMosix* es un software para construir clústeres en GNU/Linux, migrando los procesos de forma dinámica. Consiste en algoritmos de compartición de recursos adaptativos a nivel de *kernel*, que están enfocados a conseguir alto rendimiento, escalabilidad con baja sobrecarga y un clúster fácil de utilizar. La idea es que los procesos colaboren de forma que parezca que están en un mismo nodo. Los algoritmos de *OpenMosix* son dinámicos lo que contrasta y es una fuerte ventaja frente a los algoritmos estáticos,

responden a las variaciones en el uso de los recursos entre los nodos migrando procesos de un nodo a otro, de forma transparente para el proceso, para balancear la carga y para evitar falta de memoria en un nodo. *OpenMosix* puede balancear una aplicación si está dividida en procesos, lo que ocurre en un gran número de aplicaciones. Y también puede balancear las aplicaciones entre sí, lo que balancea *OpenMosix* son procesos. Cuando un nodo está muy cargado por sus procesos y otro no, se migran procesos del primer nodo al segundo (Peres Echebarria, y otros, 2008).

Características de *OpenMosix*:

**Ventajas**

- No se requieren paquetes extras.
- No son necesarias modificaciones en el código.

**Desventajas**

- Es dependiente del *kernel.*
- No migran los procesos siempre, tiene limitaciones de funcionamiento.
- Problemas con memoria compartida.

En la tabla 2 se muestra la comparación referente a las herramientas analizadas anteriormente.

*Tabla 2: Comparación entre middleware. Elaboración propia.*

| Herramienta | Formato de distribución | Distribuciones soportadas | Balanceo de carga y alta disponibilidad | Monitorización | Migración de servicios | Comunidad activa | Licencia |
|---|---|---|---|---|---|---|---|
| **Pacemaker** | Deb y código fuente | Todas | Alta disponibilidad | si | Manual y automático | si | free |
| **openMosix** | RPM y código fuente | Todas(Parche al kernel de GNU/Linux) | Balanceo de carga de procesos sin | si | Manual y automático | si | free |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | alta disponibilidad | | | | |
| **Oscar** | RPM y código fuente | Red Hat y basadas en esta. | Balanceo de carga de procesos sin alta disponibilidad | si | si | si | free |
| **Piranha** | RPM | Red Hat Enterprise 4 o superior | Posee herramientas propias para ambos aspectos | si | si | si | Red Hat |

La herramienta seleccionada para la administración de recursos del clúster será *Pacemaker* dado que no es dependiente de ninguna distribución de GNU/Linux y garantiza la alta disponibilidad de los servicios. Permite una configuración de clúster con cantidad de nodos indefinidos, por lo que se pone de manifiesto la escalabilidad de la solución a realizar. Esta herramienta integra con *Corosync* proveyendo una solución con comunicación segura entre los nodos del clúster y la migración dinámica o automática de los servicios.

### 1.12.5 Herramienta de Virtualización

**Oracle Vm VirtualBox** *(versión 4.2.8)*

VirtualBox es un software de virtualización disponible en arquitecturas x86 y x64. No solo es extremadamente rico en características de alto rendimiento para clientes empresariales, sino que es también una solución profesional que está libremente disponible como software de código abierto bajo los términos de la Licencia Pública General de GNU (GPL) versión 2. VirtualBox está siendo desarrollado activamente con los lanzamientos frecuentes y tiene una lista creciente de características, con el apoyo de sistemas operativos que virtualizan y las plataformas que soporta. VirtualBox es un esfuerzo de la comunidad respaldada por una empresa donde Oracle garantiza que se cumpla con los criterios de calidad profesional (Matos Faure, y otros, 2013-2014).

Esta herramienta se utilizó para la simulación de servidores reales para la configuración y administración

del clúster de alta disponibilidad.

### 1.12.6 Servicios

**Sshpass:** servicio para conectar por ssh sin usar claves privadas. Este servicio permitirá la copia desde los nodos esclavos de la clave de autenticación de nodos generada por *Corosync* en el nodo maestro de modo no interactiva.

**Crmsh:** este componente de *Pacemaker* provee una interfaz de línea de comandos avanzada para la administración de clústeres de alta disponibilidad en GNU/Linux. Ofrece facilidad para configurar, administrar y solucionar problemas de los clústeres desde la línea de comandos. Crmsh también ofrece funciones avanzadas como configuración de clúster de bajo nivel, secuencias de comandos de clúster y administración de paquetes, así como herramientas de exploración de historiales que le ofrecen una vista instantánea de lo que está haciendo su clúster (Wilson Armando, 2016).

**OpenSSH-server**: OpenSSH es un conjunto gratuito de código abierto de herramientas informáticas que se utiliza para proporcionar una comunicación segura y encriptada a través de una red informática utilizando el protocolo ssh. Muchas personas, nuevas en las computadoras y los protocolos, crean un concepto erróneo acerca de OpenSSH, creen que es un protocolo, pero no lo es, es un conjunto de programas informáticos que utilizan el protocolo ssh (Red Hat, 2012).

**Ssh**: SSH es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un *host* remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas (Red Hat, 2012).

### 1.12.7 Lenguaje de programación

**Python** *(versión 2.7.12)*

Es un lenguaje de programación de alto nivel creado por Guido van Rossum en el año 1991, la extensión de los proyectos creados en él es .py. Actualmente Python se desarrolla como un proyecto de código abierto, administrado por la *Python Software Fundation*. Posee una sintaxis muy limpia y un código legible. Es interpretado o de script, esto quiere decir que se ejecuta utilizando un programa intermedio llamado intérprete, con tipado dinámico, es decir, no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que

se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo; fuertemente tipado, ya que no se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario primero convertir de forma explícita dicha variable al nuevo tipo; multiplataforma y orientado a objetos o sea que los elementos del problema se tratan como clases y objetos, y la ejecución del programa consiste en la interacción entre estos, también permite la programación imperativa, programación funcional y programación orientada a aspectos (González Duque, 2008).

### 1.12.8 IDE de desarrollo a utilizar

**PyCharm** *(versión 2016.3.2)*

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Comunidad de Software Libre que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional (BBVA, 2015).

Algunas de sus características fundamentales son (BBVA, 2015):

- Integración con *frameworks* como Django, Flask, Pyramid o Web2Py.

- Autocompletado.

- Resaltador de sintaxis.

- Herramienta de análisis.

- Refactorización.

- Depurador avanzado de Python y JavaScript.

- Sistema de control de versiones como *Git, CVS* y *Mercurial*.

### 1.12.9 Herramienta de Modelado

**Visual Paradigm** *(versión 8.0)*

Es una herramienta *CASE* para desarrollo de aplicaciones utilizando el lenguaje de modelado *UML* que permite la representación de todo tipo de diagramas (Procesos de Negocio, Decisión, Actor de negocio,

Documento). 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. Se encuentra disponible para varias plataformas y cuenta con múltiples versiones con diferentes especificaciones. Entre los elementos que ofrece Visual Paradigm se encuentran (Obiol González, y otros, 2015):

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato *PDF/HTML*.
- Documentación automática *Ad-hoc*
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador automático de *Layout*.

**Conclusiones Parciales**

- ✓ La sistematización realizada a las herramientas para, la configuración de clúster de alta disponibilidad, permitió identificar que la mejor opción para la solución a implementar, será la integración de las herramientas Corosync, para la comunicación entre los nodos del clúster y Pacemaker como administrador de recursos del clúster.
- ✓ La actual situación de la distribución de GNU/Linux Nova Servidores, de no poseer la capacidad de brindar un servicio de alta disponibilidad activo para los usuarios todos los días del año, las veinticuatro horas del día, propicia el desarrollo de una estrategia de clúster de alta disponibilidad que gestione de forma eficiente los servicios a brindar, sobre las premisas de rendimiento, redundancia y disponibilidad.
- ✓ El marco teórico realizado en el capítulo, permitió identificar las principales herramientas para la creación de un clúster de alta disponibilidad, así como la metodología y aspectos como componentes de un clúster, modelos de implementación, tipos de clúster, demostrando así que la implementación de una estrategia de clúster es una solución factible para la distribución GNU/Linux Nova Servidores.

## Capítulo 2. Análisis y diseño

En el capítulo se determinaron los elementos teóricos a tener en cuenta para, crear una personalización de la distribución de GNU/Linux Nova Servidores y el módulo clúster para la herramienta Nova-Manager.

### 2.1 Propuesta del sistema

Para dar solución a la problemática planteada se propone la realización de una personalización de GNU/Linux Nova Servidores, para la creación de un clúster de alta disponibilidad. Esta personalización utilizará el núcleo de Linux e incluirá un conjunto de aplicaciones necesarias para la creación del clúster, así como un módulo para la herramienta Nova-Manager mediante la cual se podrá realizar la configuración y administración del clúster de alta disponibilidad, así como la monitorización los nodos del clúster y los servicios para obtener información sobre el estado de los mismos.

El módulo clúster para la herramienta Nova-Manager permitirá, la creación de un clúster de alta disponibilidad de tipo *Beowulf* que, consistirá en un nodo maestro que será el encargado de administrar el clúster y uno o varios nodos esclavos que serán los encargados de retomar los servicios en caso de caída del nodo maestro. Este módulo, permitirá opciones de configuración mediante las cuáles se podrán migrar servicios, mostrar configuraciones realizadas, mostrar el estado del clúster de forma que se tendrá un control total sobre la creación y administración del sistema.

Se muestra en la Ilustración 3, el modelo de relación entre los paquetes del análisis del sistema, donde sus principales paquetes son: sistema operativo base que contiene los paquetes (instalador, kernel y biblioteca), Nova Servidores 6.0 que contiene los paquetes (aplicaciones (Nova-Manager (clúster)), sesión de trabajo), repositorio de código fuente, repositorio de paquetes. La relación de dependencia de cada uno de estos paquetes y la descripción de los mismos.

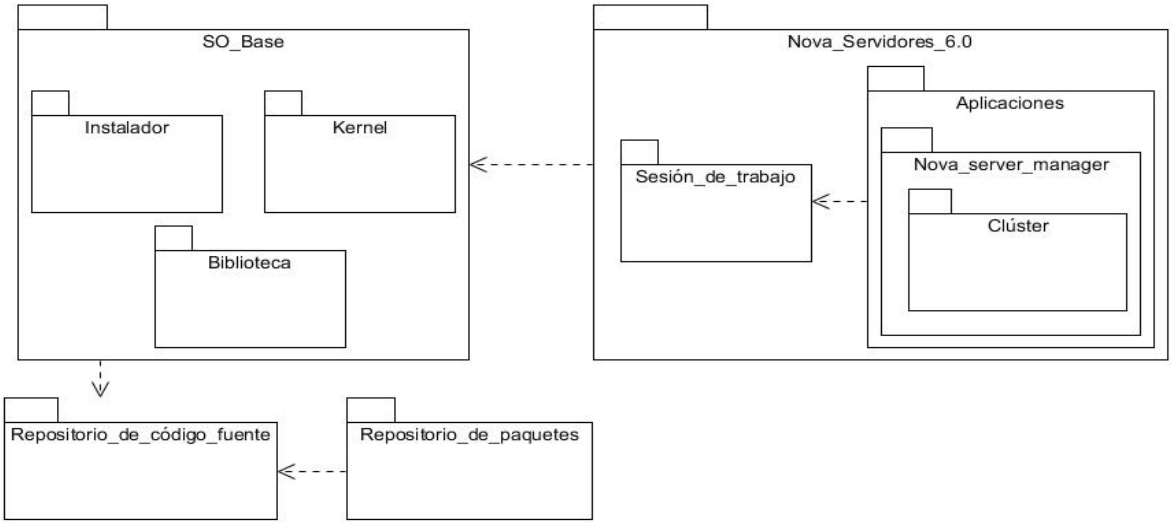



*Ilustración 3: Diagrama de paquetes de análisis del diseño de Nova Servidores 6.0. Elaboración propia.*

Para una descripción más detallada del paquete clúster se muestra en la Ilustración 4 el modelo de relación entre los paquetes del análisis del mismo, sus relaciones de importaciones(*import*) y la relación de usabilidad(*usa*) entre el paquete *frontend* y el paquete local para la internacionalización del sistema, así como sus principales fundamentos.

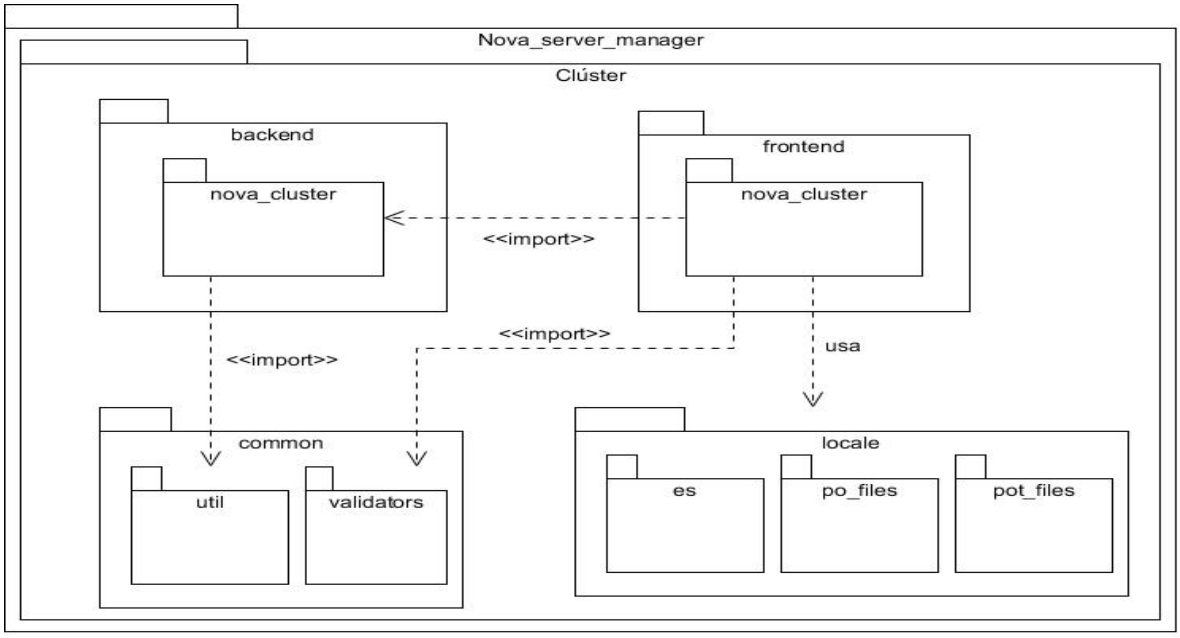



*Ilustración 4: Diagrama de paquetes del análisis del diseño del módulo clúster. Elaboración propia.*

**Descripción de los componentes del sistema**

A continuación se muestra la descripción de los componentes del sistema para la personalización de la distribución de GNU/Linux Nova Servidores(Aguilar López, 2017)**.**

**Repositorio_de_codigo_fuente:** colección de paquetes de programas en Nova que contienen archivos de código fuente que pueden ser descargados, modificados, analizados y reutilizado.

**Repositorio_de_paquetes:** es una colección de paquetes de programas de la distribución Nova que contiene archivos binarios pre compilado que pueden ser descargados e instalados por los usuarios.

**SO_Base:** estructura mínima de interacción con los componentes de funcionamiento del núcleo del sistema y con los componentes de la interfaz de usuario final.

- ***Instalador****:* herramienta que contiene módulos de gestión del meta-paquete *debian-installer* donde se modifican los paquetes necesarios para la versión de Nova Servidores 6.0.
- ***Kernel****:* es el componente central de un sistema operativo GNU/Linux. Se encarga de manejar los recursos hardware como la CPU, la memoria y los discos duros, y proporciona abstracciones que le dan a las aplicaciones una visión consistente de esos recursos.
- ***Bibliotecas_bases****:* conjunto de archivos y programas en forma de biblioteca que brindan las herramientas y funcionalidades básicas para la construcción de una distribución GNU/Linux.

**Nova Servidores 6.0:** Aplicación desarrollada para Servidores donde el trabajo en consola con la ayuda de interfaces gráficas ofrece al usuario una interacción cómoda con facilidades de acceso y configuración.

- ***Sesión_de_trabajo****:* es una colección de información que indica al sistema los archivos y carpetas a los que puede obtener acceso.
- ***Aplicaciones****:* conjunto de aplicaciones comunes en las variantes de Nova-Base y Nova-Servidores.
- ***Nova-Manager****:* es el gestor de administración de Nova Servidores. Se encarga de la instalación, configuración y desinstalación de los servicios para servidores con que cuenta Nova Servidores 6.0. Posee varios paquetes de servicios, los cuales abarcan las funcionalidades que posibilitan el uso de los servicios para los distintos tipos de servidores dentro de la arquitectura del sistema.
- ***Clúster****:* módulo para la creación del clúster de alta disponibilidad. Permitirá la configuración y

administración de clúster.

- ***Frontend**:* se encontrarán definidas todas las clases de las vistas del clúster.
- ***Backend**:* es el paquete donde se encontrarán definidas las clases controladoras del clúster.
- ***Locale**:* paquete de internacionalización del sistema.
- ***Commnon**:* se encuentran definidos los modelos de datos y los validadores.

### 2.3 Requisitos del software

Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema (Sommerville, 2005).

#### 2.3.1 Requisitos funcionales del sistema

Expresan las capacidades que debe poseer la solución, se identifican a partir de necesidades de negocio o necesidades de los clientes y usuarios. Estos requisitos describen beneficios que recibirán la organización y/o sus clientes con el desarrollo del sistema. Por lo general agrupan el comportamiento de varios requisitos funcionales de usuario (Universidad de las Ciencias Informáticas).

*Tabla 3: Requisitos funcionales del sistema. Elaboración propia.*

| Código | Descripción (Requisitos Funcionales) | Prioridad |
|---|---|---|
| **RF1** | Instalar corosync, pacemaker, sshpas y crmsh | Alta |
| **RF2** | Desinstalar corosync, pacemaker, sshpas y crmsh | Baja |
| **RF3** | Iniciar servicio corosync, pacemaker, sshpas y crmsh | Alta |
| **RF4** | Detener servicio corosync, pacemaker, sshpas y crmsh | Baja |
| **RF5** | Reiniciar servicio corosync, pacemaker, sshpas y crmsh | Baja |
| **RF6** | Cambiar propietario del clúster | Media |
| **RF7** | Generar clave de autenticación de corosync en nodo maestro | Alta |
| **RF8** | Restringir permisos de la clave generada por corosync | Media |
| **RF9** | Permitir que solo usuarios root puedan acceder a la clave de autenticación | Media |

| | | |
|---|---|---|
| **RF10** | Copiar clave de autenticación del nodo maestro a los nodos esclavos | Alta |
| **RF11** | Configurar corosync.conf en los nodos del clúster. | Alta |
| **RF12** | Configurar auto-inicio de corosync. | Baja |
| **RF13** | Configurar auto-inicio de pacemaker después del inicio de corosync | Baja |
| **RF14** | Comprobar funcionamiento del clúster. | Media |
| **RF15** | Configurar mecanismo para determinar a qué nodo migrarán los servicios. | Baja |
| **RF16** | Iniciar servicio de ip flotante en alta disponibilidad. | Alta |
| **RF17** | Configurar quorum. | Alta |
| **RF18** | Migrar servicios entre los diferentes nodos del clúster | Baja |
| **RF19** | Obtener interfaz de red. | Baja |
| **RF20** | Mostrar configuración del clúster | Baja |
| **RF21** | Detener servicios en alta disponibilidad | Baja |
| **RF22** | Eliminar servicios en alta disponibilidad | Baja |
| **RF23** | Iniciar servicios en alta disponibilidad | Alta |
| **RF24** | Permitir la instalación del sistema desde un CD, DVD o dispositivo USB | Alta |
| **RF25** | Iniciar consola para la administración. | Alta |

### 2.3.2 Requisitos no funcionales del sistema

Son restricciones de los servicios o funciones ofrecidas por la solución informática (tiempo, proceso o estándares del dominio de negocio). Normalmente afectan al sistema en su totalidad más que una característica o servicio en particular (Universidad de las Ciencias Informáticas).

**Requerimientos de software**

RNF1: Se requiere de la distribución de GNU/Linux Nova para Servidores en su versión 6.0.

**Requerimientos de hardware**

RNF2: 2 PC con 4gb de memoria RAM o superior.

RNF3: Conexión de red.

**Requerimientos de diseño e implementación**

RNF4: Utilizar como lenguaje de programación Python.

RNF5: El estilo de programación debe ser el estándar de codificación del lenguaje Python[4].

**Requerimientos de interfaz de usuario**

RNF6: Se requiere una interfaz visualmente agradable y que posibilite al usuario una navegabilidad intuitiva por las funcionalidades del sistema.

**Requerimientos de licencia**

RNF7: Se requiere del uso de herramientas y recursos de software libre, las cuáles se podrán usar, modificar y distribuir libremente.

### 2.4 Historias de usuario (HU)

La metodología empleada propone tres variantes a utilizar en los proyectos (CUN, DPN o MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP) (Sánchez, 2015). En el caso de la solución propuesta, se cuenta con un negocio muy bien definido y con la compañía del cliente con el equipo de desarrollo para convenir los detalles de los requisitos para la posterior implementación, prueba y validación. Teniendo en cuenta las anteriores características se decide adoptar por el escenario número cuatro que emplea las historias de usuarios (HU) para encapsular los requisitos. Las HU constituyen una forma de administrar los

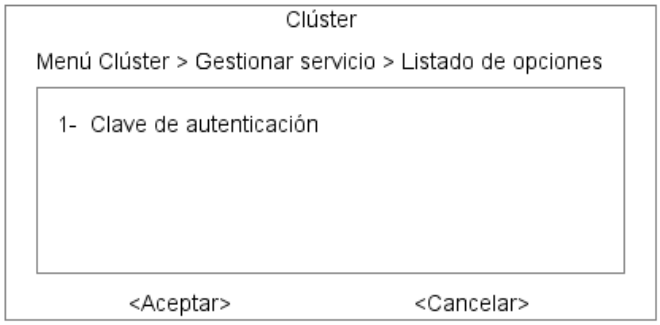
los requisitos sin la elaboración de gran cantidad de documentación y sin requerir mucho tiempo para administrarlos (Cohn, 2005).

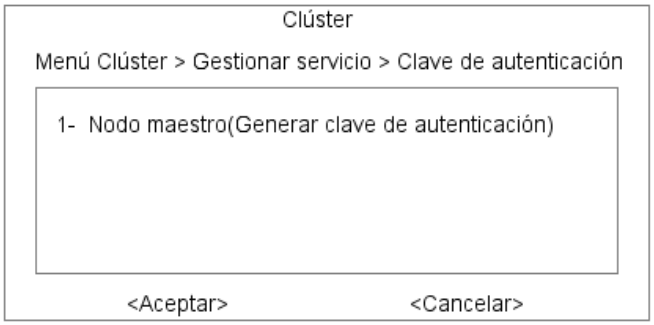
Para la descripción de los requisitos funcionales para la propuesta de solución se definió una HU para cada uno de ellos para un total de 25 HU. A continuación, se muestran la HU “Generar clave de autenticación de corosync en nodo maestro”, el resto de las historias de usuarios se encuentran definidas en el Anexo 1: Historias de usuarios.

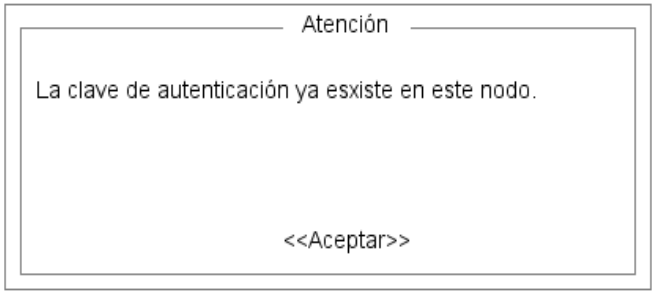
[4] Disponible en: http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf

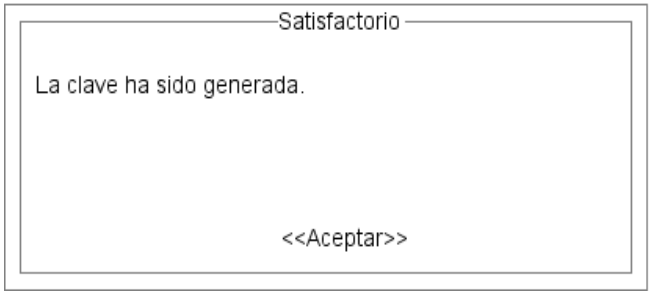
*Tabla 4: Historia de usuario "Generar clave de autenticación de corosync en el nodo maestro". Elaboración propia.*

| **Historia de usuario** | |
|---|---|
| **Número:** HU_7 | **Nombre:** Generar clave de autenticación de corosync en nodo maestro |
| **Prioridad en negocio:** Alta | **Iteración Asignada:** 7 |
| **Programador:** Nestor Llerena Rivera | **Tiempo Estimado:** 4 días |
| **Riesgo en Desarrollo:**<br>- Fallo de conexión<br>- Fallo del servidor<br>- Interrupción del fluido eléctrico | **Tiempo Real:** 5 días |
| **Descripción:** Será generada la clave de autenticación de corosync que permitirá la autenticación de los nodos, esta será generada en el nodo maestro para ser copiada hacia los nodos esclavos. Sin esta clave de autenticación los nodos no pasarán a formar parte del clúster. Esta opción deberá permitir seleccionar como tipo de nodo maestro que será el encargado de generar la clave. Si la clave ya ha sido generada se mostrará una alerta que indicará que ya fue generada en este nodo. | |
| **Prototipo:** | |



## 2.5 Diseño

El papel del diseño en el ciclo de vida de un software es facilitar la comprensión de su funcionamiento y proveer una representación o modelo del mismo con el propósito de definirlo con los suficientes detalles como para permitir su realización física. El modelo de diseño provee una representación arquitectónica del software que sirva de punto de partida para las tareas de implementación, dando al traste con los requisitos del sistema.

### 2.5.1 Descripción de la arquitectura

El estándar IEEE[5] define la arquitectura de software como la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (Garlan, y otros, 1993). Según Roger Pressman: “En su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan” (Pressman, 2009).

Para el presente trabajo de diploma se propone el patrón arquitectónico Modelo-Vista-Controlador (MVC) para el módulo clúster para la herramienta Nova-Manager ya que es la arquitectura que define este sistema para cada uno de sus diferentes módulos.

Este patrón arquitectónico separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí. El componente Modelo maneja los datos asociados al sistema y las operaciones asociadas a esos datos. El componente Vista define y gestiona como se presentan esos datos al usuario. El componente Controlador dirige la interacción del usuario y pasa estas interacciones a Vista y Modelo (Sommerville, 2005).

La solución propuesta constará con dos componentes fundamentales Vista (paquete ***Frontend***) y Controlador (paquete ***Backend***) como se muestra en la Ilustración 4 en el apéndice 2.1, dado que no se maneja una base de datos y el manejo de datos se realiza mediante la persistencia a datos en ficheros. Para un mejor entendimiento de cómo será manejada la persistencia a datos en ficheros ver la tabla 5 que se muestra a continuación.

[5] IEEE: Del inglés Institute of Electrical and Electronics Engineers. Traducido al español Instituto de Ingeniería Eléctrica y Electrónica

*Tabla 5: Persistencia a datos en ficheros. Elaboración propia.*

| Nombre | Tipo | Archivo | Forma en que se guarda |
|---|---|---|---|
| bindnetaddr | string | /etc/corosync/corosync.conf | bindnetaddr: 10.53.1.0 |
| START | string | /etc/default/corosync | START=yes |
| pcmk | string | /etc/corosync/service.d/pcmk | service {<br>name: pacemaker<br>ver: 1<br>} |

### 2.5.2 Diagrama de despliegue

El autor de la presente investigación coincide con (Sarmiento, 2016) cuando plantea que un diagrama de despliegue consiste en un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue o distribución de los artefactos del software en los destinos de despliegue; definiendo a los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Ejemplos de artefactos son archivos ejecutables, bibliotecas, archivos, esquemas de BD, archivos de configuración, etc. Cuando se refiere a destino de despliegue se hace referencia a un nodo que es o bien un dispositivo de *hardware* o bien un entorno de ejecución de software.

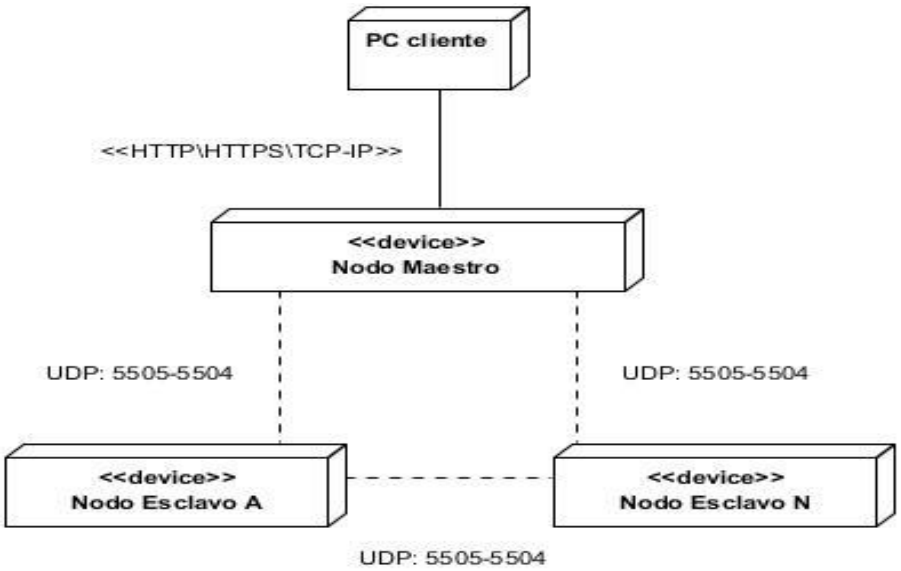



*Ilustración 5: Diagrama de despliegue. Elaboración propia.*

### 2.5.3 Patrones de diseño

Un patrón de diseño se caracteriza como “una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución”. Para el diseño de software, el con-texto permite al lector entender el ambiente en el que reside el problema y qué solución sería apropiada en dicho ambiente. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas que influyen en la manera en la que puede interpretarse el problema en este contexto y en cómo podría aplicarse con eficacia la solución (Pressman, 2009).

**Patrones Grasp[6]**

*Patrones Generales de* Software para Asignar Responsabilidades llamados comúnmente GRASP, del acrónimo *de General Responsibility Assignment Software Patterns*.

Los utilizados para el módulo son:

**Bajo acoplamiento:** El patrón Bajo acoplamiento es una medida de la fuerza con que una clase se relaciona con otras, porque las conoce y recurre a ellas; una clase con bajo acoplamiento no depende de muchas otras. El uso de este patrón permite que las clases no se afecten por cambios de otros componentes, haciendo posible que sean fáciles de entender y de reutilizar (Cabeza Chávez, 2015). Este patrón se evidencia en la clase *Main* en el paquete *frontend* donde se encuentra definida la interfaz y la implementación de sus métodos se realiza en la clase *Main* en el paquete *backend.*

**Alta Cohesión:** Se aplica para realizar un diseño que evite contener clases con un alto grado de abstracción, delegando las responsabilidades muy complejas a otros objetos. Este patrón se evidencia en las validaciones.

**Patrones GOF**

Los patrones de diseño GOF son patrones de asignación de responsabilidades, estos patrones se clasifican según su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos.

**Patrón solitario (Singleton):** es un patrón de tipo creacional que tiene como objetivo garantizar la

[6]Grasp: Patrones Generales de Software para Asignar Responsabilidades, del *inglés General Responsibility Assignment Software Patterns.*

existencia de una única instancia para una clase y posibilitar el acceso global a dicha instancia. Este patrón se evidencia en la clase *Main* en el paquete *Backend* para realizar la copia de la clave de autenticación de corosync generada en el nodo maestro hacia el nodo esclavo mediante la conexión SSH en el método slaveKey () donde se envía la dirección IP (*Internet Protocol*, Protocolo de Internet), el nombre de usuario y la contraseña del nodo del cual será copiada la clave.

### 2.5.4 Diagrama de clases del diseño del módulo clúster

Describen gráficamente las especificaciones de las clases del software y de las interfaces en una aplicación. En su diseño contiene como información (clases, métodos, información sobre los tipos de los atributos, dependencias) (Pressman, 2002). Se muestra en la Ilustración 6 el diagrama de clases del diseño del módulo clúster, que cuanta con dos clases principales, *Main* en el paquete *fontend* que proporciona la vista del sistema y su relación de importación con la clase *Main* en el paquete *backend* que es la clase controladora del sistema.

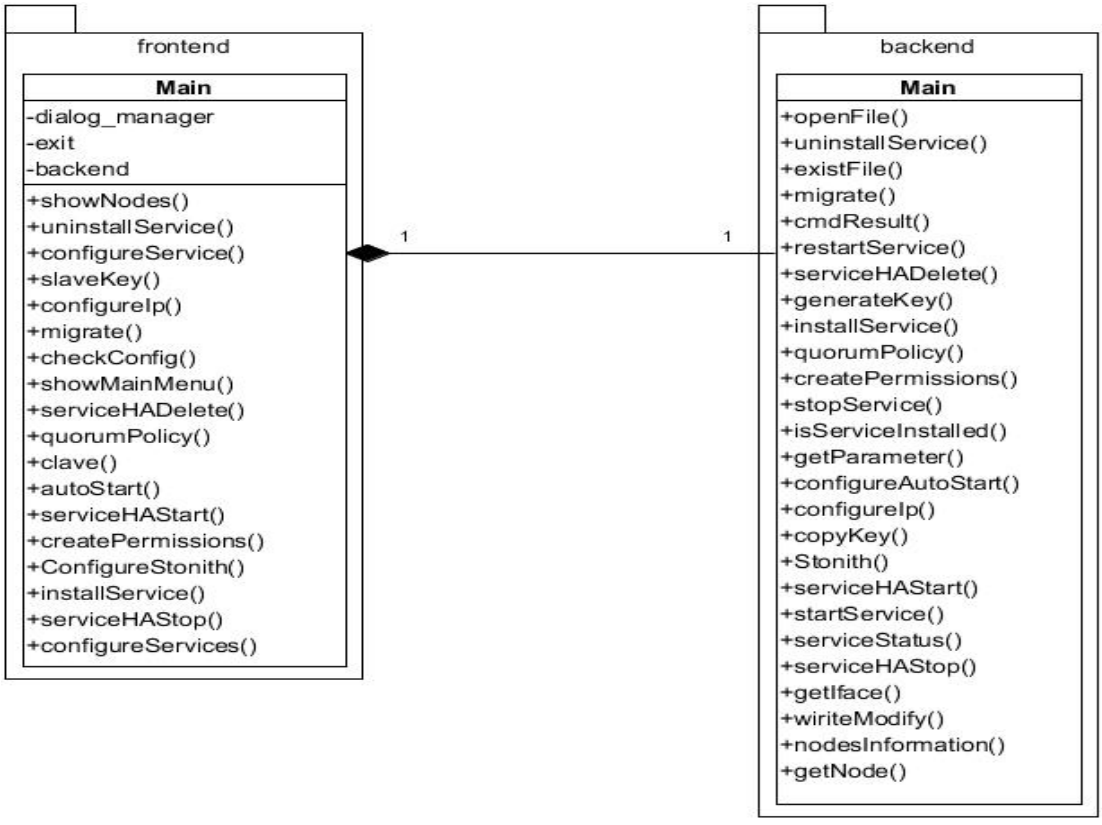



*Ilustración 6: Diagrama de clases del diseño del módulo clúster. Elaboración propia.*

## Conclusiones parciales

- ✓ En el capítulo se definió el proceso de construcción de la personalización de la distribución de GNU/Linux Nova para servidores y el módulo clúster para la herramienta Nova-Manager, que incorporará dicho sistema y permitirá la configuración y administración de un clúster de alta disponibilidad, para establecer las bases de la implementación y comprensión del mismo.
- ✓ La definición de los requisitos funcionales y no funcionales que debe cumplir el sistema, permitió realizar un correcto análisis y diseño del mismo mediante el uso del patrón arquitectónico MVC, así como los patrones GRASP y GOF.
- ✓ La definición y descripción de los requisitos funcionales y no funcionales trajo como resultado las pautas a seguir para la implementación de los mismos durante la fase de desarrollo.
- ✓ La arquitectura propuesta es la MVC, dado que es la arquitectura que define Nova-Manager para cada uno de sus módulos, lo que permitió mantener la homogeneidad con la herramienta principal. Además, fueron definidos los patrones de diseños que favorecieron la reutilización y un mejor entendimiento del código.

## Capítulo 3. Implementación y Prueba

Este capítulo centra la atención en la descripción del proceso de construcción de la distribución GNU/Linux Nova para servidores y del módulo clúster para la herramienta Nova-Manager, además de las pruebas realizadas a los mismos. Se exponen los estándares de codificación empleados para el desarrollo del módulo clúster para la herramienta Nova-Manager y se establece la estrategia de pruebas para verificar la calidad del módulo implementado, así como la documentación referente a las pruebas seleccionadas. Además de la aplicación de la técnica IADOV para evaluar la propuesta de solución.

### 3.1 Estándar de codificación

**Identación:** Emplear 4 espacios por cada nivel de identación.

```
# Instala los servicios corosync, pacemaker sshpass y crmsh para
# la configuración y administración del cluster

def installService(self):
    return subprocessCommandReal('sudo apt-get install corosync pacemaker sshpass crmsh --yes')
    subprocessCommand('apt-get update')
```

*Ilustración 7: Identación. Elaboración propia.*

**Tabuladores y espacios:** No mezclar tabulaciones con espacios, el método de identación más popular en Python es el uso de espacios y el segundo más utilizado es con tabulaciones, pero no deben ser mezclados.

```
def getIface(self):
    subprocessCommand('ifconfig -s >> /etc/corosync/iface')
    fileWr = open('/etc/corosync/iface', "r")
    lista = []
    for line in fileWr:
        lista.append(line)
    fileWr.close()
    os.remove('/etc/corosync/iface')
    return lista
```

*Ilustración 8: Tabuladores y espacios. Elaboración propia.*

**Importaciones:** Las importaciones deben estar en líneas por separado

```
import subprocess
import os.path
from common.util.commands import subprocessCommand
from common.util.commands import subprocessCommandReal
```

*Ilustración 9: Importaciones. Elaboración propia.*

**Declaraciones:** Se deben declarar cada variable en una línea diferente y de esta forma se puede comentar

cada variable por separado.

```
else:
    # Iniciar servicio corosync para comunicación del cluster
    output = self.backend.startService('corosync')
    # Iniciar servicio pacemaker para administración del cluster
    output1 = self.backend.startService('pacemaker')
    if output[1] or output1[1]:
```

*Ilustración 10: Declaraciones de variables. Elaboración propia.*

**Tamaño máximo de líneas:** Limita todas las líneas a un máximo de 79 caracteres. Esto puede ser realizado mediante el uso de paréntesis de forma implícita o mediante el uso de la barra invertida.

```
def copyKey(self,ip,user,password):
    cmd ='sshpass -p "'+password+'" scp -v -o StrictHostKeyChecking=no -o ' \
    'UserKnownHostsFile=/dev/null '+user+'@'+ip+':/etc/corosync/authkey /etc/corosync'
    return subprocessCommandReal(cmd)
```

*Ilustración 11: Tamaño máximo de líneas. Elaboración propia.*

**Comentarios:** Los comentarios deben añadir claridad al código, un comentario que contradiga el código es peor que no tener ningún comentario.

```
# Devuelve si un servicio esta instalado o no, se utiliza para determinar
# cuando se puede realizar una operación que requiera de estos servicios.

def isServiceInstalled(self,service):
    result=False
    corosync = False
    pacemaker=False
    cmd = subprocess.Popen("dpkg -s "+service+" | grep Status"\
    , shell=True, stdout=subprocess.PIPE)
    out = cmd.stdout.readlines()
    for line in out:
        if 'install ok installed' in line:
            corosync= True
    if corosync==True:
        result=True
    return result
```

*Ilustración 12: Comentarios. Elaboración propia.*

**Asignación de nombres:** Evitar nombres largos y que difieran en una letra o en el uso de mayúsculas. Para nombrar funciones y variables se describe con la primera palabra en minúscula en caso de ser

compuesta se utiliza la notación *CamelCase*[7].

Para nombres simples:

```
def migrate(self,name,service):
    return subprocessCommandReal('crm resource migrate '+service+' '+name)
```

*Ilustración 13: Asignación de nombres simples. Elaboración propia.*

Para nombres compuestos:

```
def configureAutoStart(self):
    fileWr = self.openFile('/etc/default/corosync')
    fileW = open('/etc/default/auxiliar', "w+")
```

*Ilustración 14: Asignación de nombres compuestos. Elaboración propia.*

**Líneas en blanco:** Las definiciones de métodos dentro de una clase son separados por una línea en blanco.

```
def getParameter(self, file, parameter):
    fileWr = self.openFile(file)
    for line in fileWr:
        if parameter in line and not '#' + parameter in line:
            return line
    return 'not configured'

# Configura el fichero de configuración /etc/default/corosync
# para habilitar el autoinicio de corosync.

def configureAutoStart(self):
    fileWr = self.openFile('/etc/default/corosync')
    fileW = open('/etc/default/auxiliar', "w+")
```

*Ilustración 15: Líneas en blanco. Elaboración propia.*

### 3.2 Descripción del proceso de construcción

El proceso de construcción del sistema se dividió en dos fases:

- **Primera fase:** Construcción del módulo clúster para la herramienta Nova-Manager.
- **Segunda fase:** Construcción de la personalización de la distribución de GNU/Linux Nova Servidores.

[7] *CamelCase* es la práctica de escribir frases o palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra.

### 3.2.1 Primera fase: Construcción del módulo clúster para la herramienta Nova-Manager

Para la construcción del módulo clúster para la herramienta Nova-Manager que permitirá la configuración y administración de un clúster de alta disponibilidad primeramente se hizo necesario la configuración manual del clúster mediante algunas de las principales configuraciones y comandos necesarios para la configuración y administración del clúster. Este proceso permitió adquirir los conocimientos y las habilidades necesarias para la posterior automatización del proceso. A continuación, se describen los principales pasos y configuraciones para la configuración del clúster con dos nodos y comprobar su funcionamiento.

**Paso 1: Crear dos máquinas virtuales con Nova para servidores.**

Primeramente, fueron creadas dos máquinas virtuales con la distribución Nova Servidores 6.0 instalada mediante la herramienta VirtualBox.

Máquina uno:

```
nodoa@nodoa:~$
```

Máquina dos:

```
nodob@nodob:~$
```

Autenticarse como usuario root en ambos nodos:

```
#sudo su
```

**Paso 2: Instalar corosync, pacemaker, crmsh y ssh.**

Se instalarán corosync para la comunicación entre los nodos, pacemaker como administrador de recursos, ssh para realizar la copia de la clave de autenticación de corosync entre los nodos y crmsh como componente de pacemaker para administrar y monitorizar el clúster.

Primero se actualizan los repositorios mediante los siguientes comandos:

```
#apt-get update
#apt-get upgrade
```

Para instalar las herramientas será ejecutado el siguiente comando:

```
#apt-get install corosync pacemaker crmsh ssh
```

**Paso 3: Generar clave de autenticación en el nodoa.**

Será generada la clave de autenticación mediante la cual serán reconocidos los nodos que formarán parte del clúster. Sin esta clave un nuevo nodo no podrá integrase al clúster.

```
root@nodoa:/home/nodoa# corosync-keygen
```

**Paso 4: Copiar clave de autenticación al nodob.**

```
#root@nodoa:/etc/corosync#:scp authkey nodob@10.53.2.206:/home/nodob

#root@nodob:/home/nodob#:cp authkey /etc/corosync
```

**Paso 5: Asignar permiso a la clave de autenticación en ambos nodos.**

Estos permisos deberán ser asignados a la clave en ambos nodos para que la clave sea iniciada correctamente.

```
#chmod 400 /etc/corosync/authkey
```

Asegurar que solo el usuario root pueda leer la clave:

```
#chown root:root /etc/corosync/authkey
```

**Paso 6: Cambiar propietario del clúster en ambos nodos.**

```
#chwn –R hacluster:haclient /var/lib/heartbeat
```

**Paso 7: Editar fichero /etc/corosync/corosync.conf en ambos nodos.**

Será modificado el fichero conrosync.conf donde se modificará el parámetro brindnetaddr que será por donde se comunicarán los nodos del clúster. Este parámetro puede ser la dirección IP del nodoa o la dirección de red del área local terminado en cero.

```
#nano /etc/corosync/corosync.conf
```

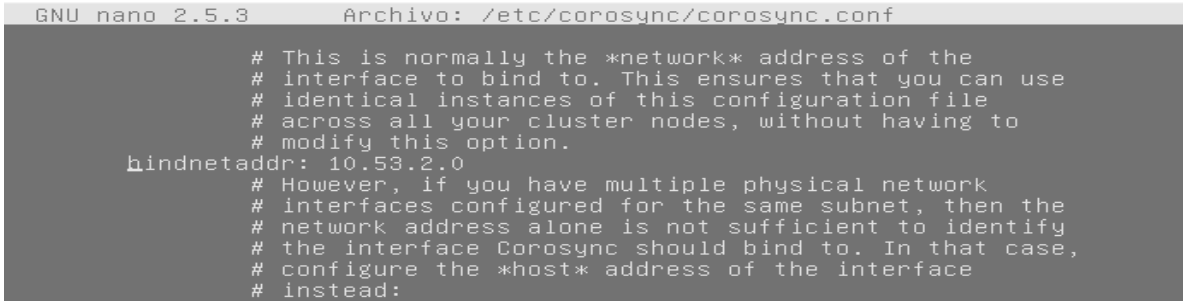



*Ilustración 16: Editar fichero corosync.conf. Elaboración propia.*

**Paso 8: Editar fichero /etc/default/corosync en ambos nodos.**

Será configurado corosync para que inicie automáticamente una vez iniciado el nodo. Cambiando el valor *Start* a *yes*.

```
#nano /etc/defaul/corosync
```

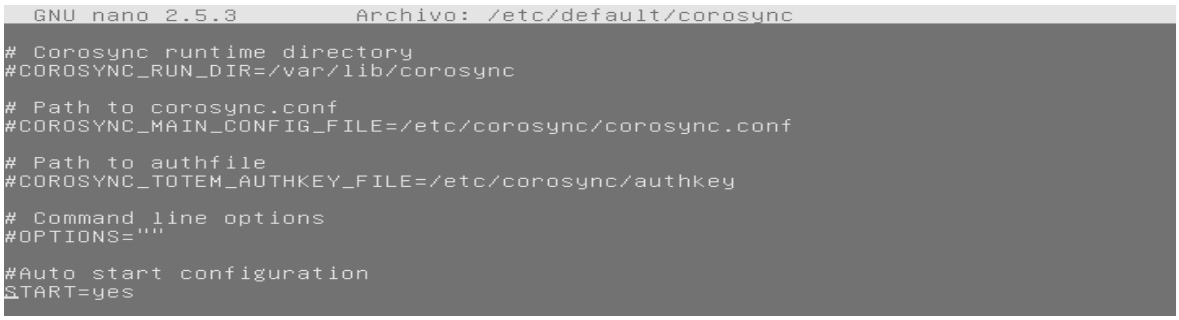



*Ilustración 17: Habilitar inicio automático de corosync. Elaboración propia.*

**Paso 9: Habilitar auto-inicio de pacemaker.**

Habilitar el auto-inicio de pacemaker estableciendo la prioridad de servicio a 20 (la prioridad de corosync es 19) esto garantiza que pacemaker inicie un segundo después de iniciado corosync.

```
#update-rc.d pacemaker defaults 20 01
```

**Paso 10: Iniciar servicios corosync y pacemaker.**

Serán iniciados corosync y pacemaker en ambos nodos para realizar las posteriores configuraciones.

```
#service corosync start

#service pacemaker start
```

**Paso 11: Configurar stonith.**

Stonith es el mecanismo que determina hacia que nodo debe migrar un servicio tras ocurrir un fallo en el nodo que este está siendo brindado.

```
#crm configure property stonith-enabled=false
```

**Paso 12: Configurar políticas quorum.**

Las políticas *quorum* son las encargadas de determinar cuando el clúster debe continuar ofreciendo los servicios. Un clúster tiene *quorum* cuando más de la mitad de los nodos se encuentran activos por lo que las políticas serán ignoradas dado que el objetivo principal es brindar los servicios, aunque solo exista un nodo activo.

```
#crm configure property no-quorum-policy=ignore
```

**Paso 13: Reiniciar los servicios corosync y pacemaker.**

```
#service corosync restart

#service pacemaker restart
```

**Paso 14: Comprobar funcionamiento del clúster.**

Mediante el comando crm_mon se muestra el estado actual de los nodos y los servicios configurados. De esta forma determinar si algún nodo no se encuentra activo o si las configuraciones no han sido realizadas correctamente.

```
#crm_mon
```

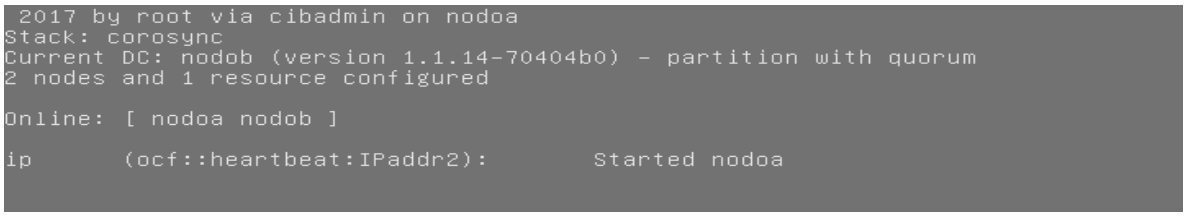



*Ilustración 18: Comprobar funcionamiento del clúster. Elaboración propia.*

Luego de ejecutado el comando se puede comprobar el estado actual de los nodos y el estado actual de los servicios.

**Paso 15: Configurar corosync para permitir servicio pacemaker.**

En ambos nodos se crea el archivo pcmk en el directorio de corosync.

```
#nano /etc/corosync/service.d/pcmk
```

El archivo debe quedar configurado de la siguiente manera:

```
service {
  name: pacemaker
}
```

**Paso 16: Configurar IP flotante.**

Mediante esta dirección IP flotante el clúster brindará los servicios a los usuarios de manera que todos los nodos del clúster responderán por una única dirección IP.

```
#crm configure primitive FAILOVER-ADDR ocf:heartbeat:IPaddr2 params ip=”10.53.1.100” nic=”eth0” op monitor interval=”10s” meta is-managed=”true”
```

Donde FAILOVER-ADDR es el nombre del servicio en alta disponibilidad, ocf la clase que provee este servicio, heartbeat el proveedor del servicio, IPaddr2 el servicio, IP la dirección por la que responderá el clúster nic la interfaz de red del nodo, op monitor interval define con qué frecuencia (en segundos) realiza la operación un valor de 0 significa nunca, un valor positivo define una acción recurrente, que se suele utilizar con monitor.

**Paso 17: Migrar servicios.**

Para migrar un servicio es necesario ejecutar el siguiente comando:

```
#crm resource migrate FAILOVER-ADDR nodob
```

Donde FAILOVER-ADDR es el nombre del servicio y nodob el nombre del nodo al que se desea migrar el servicio.

**Paso 18: Iniciar, detener y eliminar un servicio en alta disponibilidad.**

Para iniciar, detener y eliminar los servicios en alta disponibilidad es necesario ejecutar los siguientes comandos donde FAILOVER-ADDR es el nombre del servicio.

**Iniciar:**

```
#crm resource start FAILOVER-ADDR
```

**Detener:**

```
#crm resource stop FAILOVER-ADDR
```

**Eliminar:**

```
#crm configure configure delete FAILOVER-ADDR
```

Para agregar un tercer nodo al clúster deberán ser ejecutados los pasos realizados anteriormente al nodob.

Luego de realizados los pasos anteriores se hizo posible la implementación del módulo clúster para la herramienta Nova-Manager. Para realizar este proceso se hizo necesario la implementación de métodos que permitirán llevar a cabo la configuración del clúster como por ejemplo el método *getIface (self)* en la clase controladora para obtener la interfaz de red del nodo para la configuración de la dirección IP flotante como se muestra a continuación:

```
def getIface(self):
    subprocessCommand('ifconfig -s >> /etc/corosync/iface')
    fileWr = open('/etc/corosync/iface', "r")
    lista = []
    for line in fileWr:
        lista.append(line)
    fileWr.close()
    os.remove('/etc/corosync/iface')
    return lista
```

*Ilustración 19: Método para obtener interfaz de red. Elaboración propia.*

Los siguientes pasos fueron realizados a partir de la distribución de GNU/Linux Nova Servidores 6.0

### 3.2.2 Segunda fase: Construcción de la personalización de la distribución de GNU/Linux Nova Servidores.

**Paso 1: Modificar la base del sistema (filesystem. squashfs)**

El sistema de ficheros en las distribuciones de GNU/Linux se almacena en un fichero comprimido tipo squashfs, típicamente: filesystem.squashfs. Para realizar modificaciones del filesystem.squashfs es necesario la instalación del paquete squashfs-tools:

```
#apt-get install squashfs-tools
```

Luego de instalada la herramienta se crea el directorio filesystem donde será descomprimido el archivo filesystem.sqashf y se ejecuta el siguiente comando:

```
#unsquashfs -d filesystem/ -f filesystem.squashfs
```

Una vez descomprimido el archivo se realizan las modificaciones necesarias, en este caso será actualizado el directorio /usr/share/nova_manager/ donde serán copiados los paquetes del módulo clúster que serán incluidos en la herramienta Nova-Manager.

Luego de realizadas las modificaciones necesarias se crea nuevamente el fichero filesystem.squashfs ejecutando el siguiente comando:

```
# mksquashfs filesystem/ filesystem. squashfs
```

**Paso 2: Crear estructura del repositorio**

Se crea la estructura del repositorio igual a la estructura del repositorio http://nova.f10.uci.cu/ donde serán copiados los siguientes paquetes con todas sus dependencias.

*Tabla 6: Paquetes a incluir en el repositorio base. Elaboración propia.*

| Paquete | Descripción |
|---|---|
| corosync | Para la comunicación entre los nodos del clúster. |
| pacemaker | Para la administración de recursos del clúster. |
| crmsh | Componente de pacemaker para el control de los servicios. |
| ssh-pass | Para la comunicación sin utilizar claves privadas. |
| ssh | Protocolo para la comunicación segura. |
| openssh-server | Para la comunicación mediante el protocolo ssh. |

A continuación, se muestra como queda la estructura del repositorio:

```
/dist/2017/principal/binary-amd64

/dists/2017/extendido/binary-amd64
```

```
/pool/principal

/pool/extendido
```

**Paso 3: Generar *packages* del repositorio**

En estos *packages* se encuentra la información de los paquetes y sus dependencias. Para generar los *packages* se ejecutan los siguientes comandos:

```
dpkg-scanpackages pool/principal /dev/null > dists/2017/principal/binary-amd64/Packages.gz

dpkg-scanpackages pool/extendido /dev/null > dists/2017/extendido/binary-amd64/Packages.gz
```

**Paso 4: Crear el archivo Release**

Primeramente, se crea un archivo release.conf con los siguientes valores:

```
APT::FTPArchive::Release::Origin "Nova";

APT::FTPArchive::Release::Label "Nova";

APT::FTPArchive::Release::Suite "2017";

APT::FTPArchive::Release::Codename "2017";

APT::FTPArchive::Release::Version "6.0";

APT::FTPArchive::Release::Architectures "amd64";

APT::FTPArchive::Release::Components "principal extendido";

APT::FTPArchive::Release::Description "Nova 2017 6.0";
```

A partir de este archivo se genera el archivo Release ejecutando el siguiente comando:

```
apt-ftparchive -c /home/nesty/Escritorio/release.conf release dists/2017/> dists/2017/Release
```

**Paso 5: Generar suma hash con md5**

Dentro de la carpeta donde se encuentra el ISO a crear se ejecuta el siguiente comando para generar la suma md5:

```
find  -type f -exec md5sum '{}' \; > md5sum.txt
```

**Paso 6: Modificar fichero *preseed***

Se modifica el fichero *preseed* que es el fichero para automatizar el proceso de instalación a partir de este. Fue modificado el archivo *preseed* disponible en la distribución de GNU/Linux Nova Servidores donde fueron agregadas las siguientes líneas donde una vez iniciado el sistema operativo tendrá instaladas las herramientas pacemaker, ssh, sshpass, openssh-server y crmsh:

```
tasksel    tasksel/first          multiselect server standar
d-i        pkgsel/include string corosync pacemaker ssh sshpass openssh-server crmsh
d-i finish-install/reboot_in_progress note
```

**Paso 7: Crear el ISO**

Una vez realizado los pasos anteriores se procede a la creación del ISO de Nova Servidores para alta disponibilidad ejecutando el siguiente comando:

```
mkisofs -r -V "Nova para Servidores" -cache-inodes -J -l -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o /home/Nesty/Escritorio/Nova_Servidores_AltaDisponibilidadHA.iso /home/Nesty/Escritorio/Nova_Servidores
```

donde /home/Nesty/Escritorio/Nova_Servidores_AltaDisponibilidadHA.iso es la dirección y el nombre con el que será generado el ISO y /home/Nesty/Escritorio/Nova_Servidores la carpeta a partir de la cual este se genera.

### 3.3 Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia y el desempeño de un software, involucra las operaciones del sistema bajo condiciones controladas y evaluando los resultados (Pressman, 2002).

#### 3.3.1 Pruebas a realizar

**Funcionales**

Se escoge este nivel de prueba ya que lo que se desea es la verificación de las funcionalidades del sistema a través de las interfaces y su relación con los demás componentes del sistema.

Las pruebas de sistema tienen como objetivo ejercitar profundamente el sistema comprobando la

integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica (Pressman, 2002). En lugar de caer en este absurdo, el ingeniero del software debe de anticiparse a posibles problemas de la interfaz (Cabeza Chávez, 2015):

- ✓ Diseñar ruta de manejo de errores que prueben toda la información proveniente de otros elementos del sistema.
- ✓ Aplicar una serie de pruebas que simulen datos incorrectos u otros posibles errores en la interfaz de software.
- ✓ Registrar los resultados de la prueba como evidencia en el caso de que se culpe.

**Unitarias**

La prueba de unidad se concentra en el esfuerzo de verificación de la unidad más pequeña del diseño del software: el componente o módulo de software. Tomando como guía la descripción del diseño a nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites del módulo. Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos dentro de los límites de un componente (Pressman, 2009).

**Integración**

La prueba de integración es una técnica simétrica para construir la arquitectura del software mientras, al mismo tiempo, se aplican las pruebas para descubrir errores asociados con la interfaz. El objetivo de tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño (Pressman, 2009). La estrategia para realizar la prueba de integración seleccionada por el autor de la investigación científica es la integración ascendente.

La estrategia de integración ascendente como su nombre lo indica, empieza la construcción y la prueba con componentes de los niveles más bajos de la estructura del programa (Pressman, 2009).

### 3.3.2 Método de prueba

**Caja Negra**

Se escoge el método de caja negra ya que los niveles de pruebas escogidos pretenden probar el funcionamiento de la interfaz y de las funcionalidades del sistema, para esto es necesario este método ya

que permite comprobar el comportamiento del software a través de los requisitos funcionales (Pressman, 2002), obviando el funcionamiento interno y la estructura del programa.

Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y de terminación.


*Ilustración 20: Esquema del Método de Caja Negra.*

**Caja Blanca**

La prueba de caja blanca, en ocasiones llamada prueba de caja de cristal, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Al emplear los métodos de prueba de caja blanca, el ingeniero del software podrá derivar casos de prueba que (Pressman, 2009):

1. Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
2. Ejerciten los lados verdadero y falso de todas las decisiones lógicas.
3. Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
4. Ejerciten estructuras de datos internos para asegurar su validez.

### 3.3.3 Técnica de prueba

**Partición equivalente**

**Características**

- Divide el dominio de entrada de un programa en clases de datos, a partir de las cuales pueden derivarse casos de prueba.
- Descubre clases de errores, que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general.
- Reducir al máximo el total de casos de prueba que deben desarrollarse (Cabeza Chávez, 2015).

**El diseño consiste:**

- Identificar clases de equivalencia. Las clases de equivalencia son un conjunto de estados válidos o inválidos para condiciones de entrada.
- Crear los casos de prueba.

**Camino básico**

La prueba de camino básico es una técnica de prueba de caja blanca. Este método permite que el diseñador de casos de pruebas obtenga una medida de complejidad lógica de un diseño procedimental y que use esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para ejercitar el conjunto básico deben garantizar que se ejecutan cada instrucción del programa por lo menos una vez durante la prueba (Pressman, 2009).

## 3.4 Aplicación de las pruebas

### 3.4.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas (Sánchez, 2015).

**Pruebas unitarias**

Las pruebas unitarias fueron realizadas mediante el método de caja blanca mediante la técnica camino básico. Las unidades de prueba más pequeña son las operaciones dentro de la clase. A continuación, se realiza la técnica del camino básico, al método *configureCorosync ()* en la clase *Main* en el paquete *backend*.

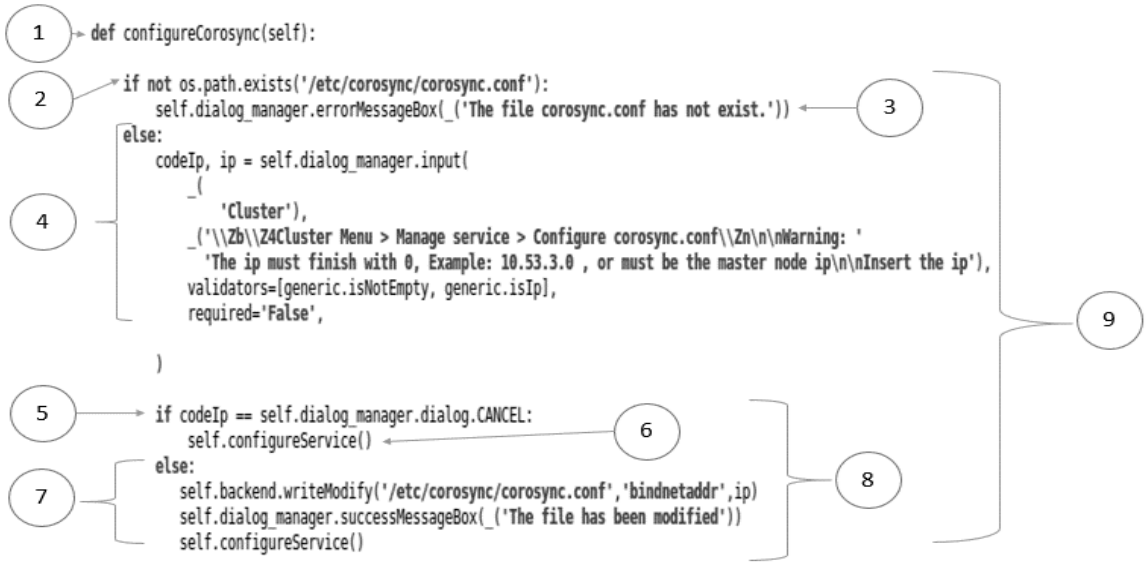



*Ilustración 21: Enumeración de las líneas de código. Elaboración propia.*

Luego de enumerar las líneas de código, se diseña la gráfica del programa que describe el flujo de control lógico empleando nodos y aristas.

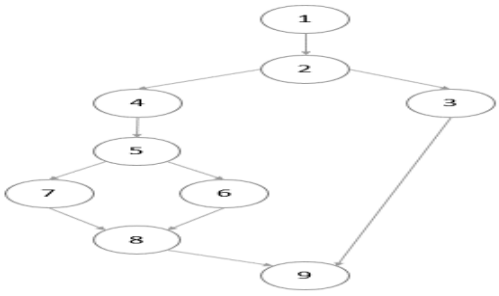



*Ilustración 22: Grafo de flujo. Elaboración propia.*

A partir del grafo obtenido con 9 nodos y 10 aristas, se calcula la complejidad dicromática *V(G)* la cual constituye una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa.

**V(G)** = cantidad_aristas – cantidad_nodos + 2

**V(G)** = 10 – 9 + 2 = 3

Un camino independiente es cualquier camino del programa que introduce al menos un nuevo conjunto de sentencias de proceso o una nueva condición (Pressman, 2009). La cantidad de caminos independientes se establece por la complejidad ciclomática, por tanto, se identifican 3 caminos como se muestra en la siguiente tabla.

*Tabla 7: Listado de caminos independientes. Elaboración propia.*

| No. | Camino |
|---|---|
| 1 | 1-2-3-9 |
| 2 | 1-2-4-5-7-8-9 |
| 3 | 1-2-4-5-6-8-9 |

El valor de la complejidad ciclomática ofrece además un límite superior para la cantidad de pruebas que se deben diseñar y ejecutar para garantizar que se cumplen todas las sentencias del programa (Pressman, 2009). A continuación, se muestran los casos de pruebas para cada camino independiente.

*Tabla 8: Caso de prueba de unidad para el camino 1. Elaboración propia.*

| **Caso de prueba de unidad** | |
|---|---|
| **No. Camino:** 1 | **Camino:** 1-2-3-9 |
| **Nombre de la persona que realiza la prueba:** Nestor Llerena Rivera | |
| **Descripción de la prueba:** Verificar configuración de *corosync.conf* en los nodos del clúster. | |
| **Entrada:** No | |
| **Resultado esperado:** Se muestra el mensaje “El fichero *conrosync.conf* no existe” | |
| **Evaluación de la prueba:** Satisfactoria. Se mostró el mensaje correctamente. | |

*Tabla 9: Caso de prueba de unidad para el camino 2. Elaboración propia.*

| **Caso de prueba de unidad** | |
|---|---|
| **No. Camino:** 2 | **Camino:** 1-2-4-5-7-8-9 |

| |
|---|
| **Nombre de la persona que realiza la prueba:** Nestor Llerena Rivera |
| **Descripción de la prueba:** Verificar configuración de *corosync.conf* en los nodos del clúster. |
| **Entrada:** Dirección IP: 10.53.3.0 |
| **Resultado esperado:** Se modifica el parámetro *bindnetaddr* : 10.0.0.0 con la nueva dirección IP y muestra el menú principal. |
| **Evaluación de la prueba:** Satisfactoria. Fue modificado el fichero en el parámetro *bindnetaddr* con la nueva dirección IP. |

*Tabla 10: Caso de prueba de unidad para el camino 3. Elaboración propia.*

| **Caso de prueba de unidad** | |
|---|---|
| **No. Camino:** 3 | **Camino:** 1-2-4-5-6-8-9 |
| **Nombre de la persona que realiza la prueba:** Nestor Llerena Rivera | |
| **Descripción de la prueba:** Verificar configuración de *corosync.conf* en los nodos del clúster. | |
| **Entrada:** El usuario presiona cancelar. | |
| **Resultado esperado:** No se modifica el parámetro *bindnetaddr* y se muestra el menú de opciones. | |
| **Evaluación de la prueba:** Satisfactoria. | |

### Resultado de las pruebas unitarias

Fueron realizadas tres iteraciones mediante el método de caja blanca mediante la técnica de camino básico donde fue encontrado un error de validación para la escritura en fichero y un error de llamada a métodos.

### Pruebas Funcionales

Las pruebas funcionales fueron realizadas mediante en el método de caja negra mediante la técnica partición equivalente. A continuación, se muestra el diseño de Casos de Pruebas basados en Historias de Usuarios para el caso de prueba “Iniciar servicio de IP flotante en alta disponibilidad” el resto de los casos de pruebas se encuentran definidos en el Anexo 2.

**Caso de Prueba 2. Iniciar servicio de IP flotante en alta disponibilidad.**

**Descripción de las variables**

*Tabla 11: Descripción de las variables del caso de prueba "Iniciar servicio ip flotante en alta disponibilidad.".*

| No | Nombre de Campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Introduzca el IP. | String | no | Debe ser una dirección IP válido, no debe estar vacío. |
| 2 | Introduzca el nombre del servicio. | String | no | Acepta cualquier carácter alfanumérico no debe estar vacío. |

*Tabla 12: Descripción de los escenarios del caso de prueba " Iniciar servicio de IP flotante en alta disponibilidad".*

| Escenario | Descripción | V1 Dirección IP del servicio IP flotante | V2 Nombre del servicio | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|
| EC 1.1 Configuración del servicio IP flotante. | Se configura el servicio IP flotante en alta disponibilidad y el clúster responde por esta dirección. | V<br>10.53.1.100 | V<br>FAILOVERADDR | El sistema muestra un mensaje de confirmación "IP flotante ha sido configurado satisfactoriamente." | 1. El usuario selecciona la opción gestionar servicio y presiona aceptar.<br>2. El usuario selecciona la opción configurar servicios y presiona aceptar.<br>3. El usuario selecciona la opción IP flotante y presiona aceptar.<br>4. El usuario introduce la |
| EC 1.2 Configuración del servicio IP flotante con campos incorrectos. | No se configura el servicio IP flotante en alta disponibilidad y el clúster no responde por esta dirección. | I<br>asd | V<br>FAILOVERADDR | El sistema muestra "Debe ser una dirección IP." | |
| | | V<br>10.53.3.100 | I<br>FAILOVER-ADDR | El sistema muestra "El valor debe contener solo letras y números." | |

| | | | | | |
|---|---|---|---|---|---|
| EC 1.3 Configuración del servicio IP flotante con campos vacíos. | No se configura el servicio IP flotante en alta disponibilidad y el clúster no responde por esta dirección. | I<br>(vacío) | V<br>FAILOVERADDR | El sistema muestra "No debe estar vacío." | dirección IP para la IP flotante y presiona aceptar.<br>5. El usuario introduce el nombre del servicio y presiona aceptar. |
| | | V<br>10.53.3.100 | I<br>(vacío) | El sistema muestra "No debe estar vacío." | |
| EC 1.3 Configuración del servicio IP flotante con campos incorrectos. | No se configura el servicio IP flotante en alta disponibilidad y el clúster no responde por esta dirección. | I<br>asd | I<br>Failover@- | El sistema muestra un mensaje de Error "La dirección IP flotante no ha sido configurada" | |

**Resultados de los casos de pruebas**

Se utilizó el método de caja negra para realizar las pruebas sobre la interfaz del módulo clúster para la herramienta Nova-Manager. Donde fueron encontradas 8 no conformidades en 3 iteraciones. De estas no conformidades 3 fueron de errores ortográficos y 5 de validación incorrecta. En la Ilustración 23 se muestran las iteraciones realizadas y las no conformidades detectadas, las resueltas y las pendientes.

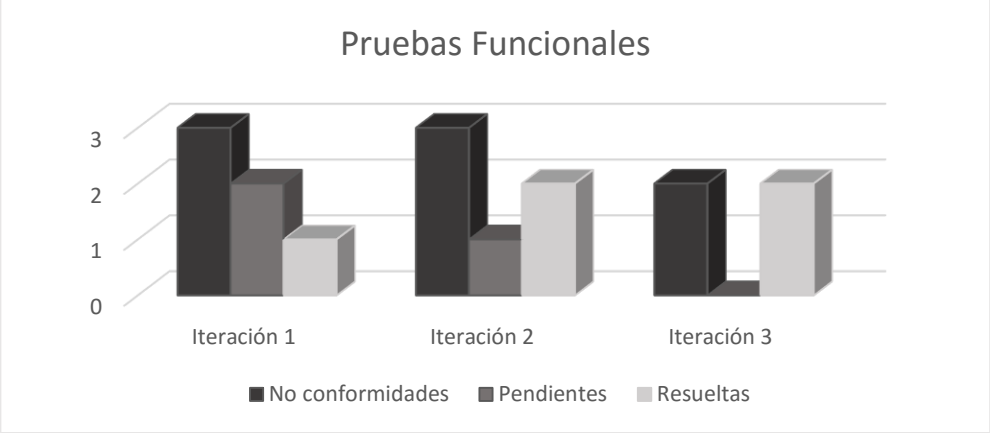



*Ilustración 23: Estadísticas de las pruebas. Elaboración propia.*

**Integración**

Las pruebas de integración fueron realizadas mediante la técnica integración ascendente. En la siguiente tabla se muestran los componentes en nivel ascendente y la integración con el componente en el nivel superior.

*Tabla 13: Integración entre los componentes. Elaboración propia.*

| Nivel | Componente | Descripción | Integración | Resultado |
|---|---|---|---|---|
| 1 | Módulo Clúster | Módulo para la configuración y administración de un clúster de alta disponibilidad. | Herramienta Nova-Manager | Satisfactorio |
| 2 | Herramienta Nova-Manager | Gestor de administración de nova servidores. Se encarga de la instalación, configuración y desinstalación de los servicios para servidores con que cuenta Nova Servidores 6.0. | Personalización de la distribución de GNU/Linux Nova Servidores 6.0 para alta disponibilidad. | Satisfactorio |
| 3 | Personalización de la distribución de GNU/Linux Nova Servidores 6.0 para alta disponibilidad. | Aplicación desarrollada para Servidores donde el trabajo en consola con la ayuda de interfaces gráficas ofrece al usuario una interacción cómoda con facilidades de acceso y configuración | | |

**Resultado de las pruebas de integración**

Mediante las pruebas de integración realizadas se puedo comprobar la integración de forma ascendente de los componentes de la solución donde los resultados obtenidos fueron satisfactorios luego de realizadas las pruebas.

### 3.4.2 Pruebas de Aceptación

Se escoge realizar la prueba de aceptación ya que es necesario para la validación de la solución que el cliente este de acuerdo con el funcionamiento del sistema desarrollado y así pueda emitir la carta de aceptación donde se demuestra la conformidad del cliente con la solución.

Como técnica se escoge las pruebas alfas esta se lleva a cabo por un cliente en el lugar de desarrollo. Estas pruebas se llevan a cabo en un entorno controlado y se usa el software de forma natural con el desarrollador como observador del usuario.

Para que tengan validez se debe crear un ambiente con las mismas condiciones que se encontrarán en las instalaciones del cliente. Una vez logrado esto se procede a realizar las pruebas y a documentar los resultados. El resultado de las pruebas de aceptación se muestra en el Anexo 3, donde fue emitida el acta de aceptación con el cliente.

### 3.5 Evaluación del objetivo general de la investigación

La técnica de IADOV es utilizada para determinar el nivel de satisfacción individual y grupal de los usuarios a partir de una encuesta elaborada según las exigencias pertinentes. En la presente investigación fue aplicada a cinco especialistas, objeto de estudio. La encuesta realizada se muestra en el Anexo 4.

La técnica de IADOV constituye una vía indirecta para el estudio de satisfacción, ya que los criterios que se utilizan se fundamentan en las relaciones que se establecen entre las tres preguntas cerradas, que se intercalan dentro de un cuestionario y cuya relación el encuestado desconoce. En este caso, de la encuesta, son las preguntas 2,3 y 4 se relacionan a través de lo que se denomina el “Cuadro Lógico de IADOV” que se muestra en la tabla 14.

*Tabla 14: Cuadro lógico de IADOV.*

| 4. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la solución desarrollada. | 2. ¿Considera usted correcta la forma en que se brindan los servicios actualmente a los usuarios? | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **No** | | | **No sé** | | | **Sí** | | |
| | 3. ¿Considera usted factible la creación de una personalización de la distribución de GN/Linux Nova Servidores para Alta Disponibilidad? | | | | | | | | |
| | **Sí** | **No sé** | **No** | **Sí** | **No sé** | **No** | **Sí** | **No sé** | **No** |
| Me gusta mucho | 1 | 2 | 6 | 2 | 2 | 6 | 6 | 6 | 6 |
| Me gusta más de lo que me disgusta | 2 | 2 | 3 | 2 | 3 | 3 | 6 | 3 | 6 |
| Me da lo mismo | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Me disgusta más de lo que me gusta | 6 | 3 | 6 | 3 | 4 | 4 | 3 | 3 | 4 |
| No me gusta nada | 6 | 6 | 6 | 6 | 4 | 4 | 6 | 6 | 5 |
| No sé decir | 2 | 3 | 6 | 3 | 3 | 3 | 6 | 6 | 4 |

La forma de utilizar la tabla es la siguiente: Cada encuestado recibe una evaluación individual en dependencia de las respuestas que dé a las preguntas cerradas. Para facilitar el procesamiento posterior, en el diseño de la encuesta se debe tener en cuenta que a estas preguntas sólo se responda de la forma prevista en el cuadro lógico de IADOV. Las respuestas a las preguntas 2 y 3 pueden ser SI, NO, NO SÉ, y a las preguntas 4, “Me gusta mucho”, “Me gusta más de lo que me disgusta”, “Me da lo mismo”, “Me disgusta más de lo que me gusta”, “No me gusta nada”, o “No sé qué decir”.

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y – 1. El número resultante de la interrelación de las tres preguntas indica la posición de cada encuestado en la siguiente escala de satisfacción: clara satisfacción +1, más satisfecho que insatisfecho 0.5, no definido y contradictorio 0, más insatisfecho que satisfecho -0.5 y clara insatisfacción -1

El índice de satisfacción grupal (ISG) se expresa en una escala numérica que va desde 1 (máxima satisfacción), hasta -1 (máxima insatisfacción). El ISG se calcula mediante la siguiente fórmula:

$$ISG = \frac{\mathrm{A}\,(+1)\ +\ \mathrm{B}\,(+0.5)\ +\ \mathrm{C}\,(0)\ +\ \mathrm{D}\,(-0.5)\ +\ \mathrm{E}\,(-1)}{\mathrm{N}}$$

En esta fórmula A, B, C, D, E, representan la cantidad de encuestados colocados respectivamente en las posiciones de satisfacción 1; 2; 3 o 6; 4; 5 y donde N representa la cantidad total de encuestados

Por ejemplo, si 4 especialistas presentan máxima satisfacción, 1 están más satisfechos que insatisfechos, el índice de satisfacción se calcularía de la siguiente forma:

$$ISG = \frac{4\,(+1) +\ 1(+0.5)\ +\ 0\,(0)\ +\ 0\,(-0.5)\ +\ 0\,(-1)}{5}$$

**ISG**=0.9


*Ilustración 24: Ubicación del índice de satisfacción Grupal. Elaboración propia.*

Los valores de ISG que se encuentran comprendidos entre -1 y -0,5 indican insatisfacción; los comprendidos entre -0,49 y +0,49 evidencian contradicción y los que caen entre 0,5 y 1 indican que existe satisfacción.

El proceso de evaluación del objetivo de la investigación mediante la técnica de IADOV confirmó su factibilidad de uso, expresando cuantitativamente en el alto ISG (0.9) y cualitativamente en los criterios emitidos en el centro CESOL, lo que refleja la aceptación de la propuesta y el reconocimiento a su utilidad.

**Conclusiones parciales**

- ✓ La realización de las dos fases definidas permitió obtener la imagen de instalación de la personalización de la distribución de GNU/Linux Nova Servidores, que tendrá incorporado un módulo para la herramienta Nova-Manager para la configuración de un clúster de alta disponibilidad, siendo la misma el producto que se propone como solución de la presente investigación.
- ✓ Los diseños de los casos de pruebas, creados a partir de los requisitos definidos del sistema permitieron validar la propuesta de solución, luego de tres iteraciones de pruebas donde fueron corregidas las no conformidades detectadas.
- ✓ La prueba de aceptación realizada permitió evaluar la propuesta de solución por parte del cliente y su satisfacción con el producto final.
- ✓ La aplicación de la técnica de IADOV permitió evaluar de manera satisfactoria la propuesta de solución demostrando la satisfacción de los potenciales usuario hacia el sistema desarrollado.

**Conclusiones Generales**

- ✓ El análisis y la fundamentación teórico-metodológico de los principales conceptos asociados a la investigación científica, permitió la conceptualización de los componentes de un clúster de alta disponibilidad, así como los tipos de clúster y por último los modelos de implementación de un clúster.
- ✓ La sistematización del marco teórico de la investigación científica, posibilitó la selección de la metodología, herramientas y tecnologías para la implementación de la solución propuesta, las que se enmarcan en AUP en su versión para la UCI como metodología de desarrollo, Genisoimage como herramienta para la creación del ISO, Corosync como herramienta para la comunicación entre los nodos del clúster, Pacemaker como administrador de recursos del clúster, crmsh como componente de Pacemaker para la monitorización de los servicios, Python como lenguaje de desarrollo, PyCharm como entorno de desarrollo integrado, Visual Paradigm para UML como herramienta para modelado.
- ✓ El uso del patrón arquitectónico MVC unido al empleo de los patrones de diseño: alta cohesión y bajo acoplamiento, permitió desarrollar el módulo clúster para la distribución de GNU/Linux Nova Servidores de manera que fue posible integrar su funcionamiento a la herramienta Nova-Manager.
- ✓ El desarrollo de la personalización de la distribución GNU/Linux Nova para Servidores, mediante el uso de herramientas de software libre, permitió garantizar la alta disponibilidad en el sistema operativo base para el proceso de migración en los Organismos de la Administración Central del Estado.
- ✓ La aplicación de las pruebas, métodos y técnicas demostraron el correcto funcionamiento del módulo clúster de la herramienta Nova-Manager para la distribución de GNU/Linux Nova Servidores, la satisfacción del cliente y los potenciales usuarios hacia la solución desarrollada.

**Recomendaciones**

Como resultado de la investigación el autor recomienda:

- Añadir las funcionalidades para la integración de los tipos de clúster: clúster de balanceo de carga y clúster de alto rendimiento mediante el uso de nuevas herramientas y tecnologías.
- Desplegar la solución en un entorno real para comprobar el funcionamiento de la solución bajo estas condiciones.

**Referencias Bibliográficas**

**Abraira, José Angel Alvarez. 2016.** *Sistema interactivo - experimental para el proceso de enseñansa - aprendizaje de la Matemática Discreta en la carrera Ingeniería en Ciencias Informáticas.* La Habana : s.n., 2016.

**AceNet Technology. 2009.** AceNet. [En línea] 2009. [Citado el: 3 de 12 de 2016.] http://acenettech.com/solutions/securitymanagement.html.

**Aguilar López, Oneisy. 2017.** *Modelo de diseño Proyecto Nova-Servidores 6.0.* La Habana : s.n., 2017.

**Avi Silberschatz, Peter Galvin. 2005.** *Operating System Concepts.* s.l. : 7ma edicion, 2005.

**BBVA. 2015.** BBVA. [En línea] 8 de 5 de 2015. [Citado el: 22 de 3 de 2017.] https://bbvaopen4u.com/es/actualidad/herramientas-basicas-para-los-desarrolladores-en-python.

**Cabeza Chávez, Yanet. 2015.** *Módulo de JAVA para la herramienta Auditoría de Código Fuente.* La Habana : s.n., 2015. Trabajos de diplomas.

**Cáceres, Gerson. 2012.** *Estrategia de implementación de un clúster de alta disponibilidad de N nodos sobre linux usando sofware libre.* Quito : s.n., 2012.

**Cohn, Mike. 2005.** *Agile Estimating and Planing.* 2005. ISBN 0-13-147941-5.

**Contreras Vásquez, Gilberto, y otros. 2007.** *Informática 1.* México : Tercera edición, 2007.

**Duque, Raúl gonzález. 2012.** *Python para todos.* s.l. : vol.2, 2012.

**Duran, Cristian. 2009.** SlideShare. [En línea] 27 de 10 de 2009. [Citado el: 27 de 11 de 2016.] http://www.slideshare.net/cduranmardones/trabajo-clusters.

**Free Software Foundation. 2016.** gnu.org. [En línea] Free Software Foundation, 25 de 8 de 2016. [Citado el: 15 de 11 de 2016.] https://www.gnu.org/philosophy/free-sw.es.html.

**G Figueroa, Robert, A Cabrera, Armando y J Solís, Camilo. 2007.** *Metodologías Tradicionales vs Metodologías Ágiles.* 2007.

**García Rivas, Dairelis, Fuentes Rodríguez, Juan Manuel y Rabelo Pérez, Jesús. 2016.** *Nova 5.0.* La Habana : Fórum de Ciencia y Técnica 2016, 2016.

**Garlan, David y Shaw, Mary. 1993.** *An introduction to software architecture. Advances in Software Engineering and Knowledge Engineering. .* 1993. Vol. vol1.

**Giancarlo Ruiz, Erick, Marcela Diaz, Tania y Sinisterra, Maria Mercedes. 2012.** Dialnet. [En línea] 4 de 9 de 2012. [Citado el: 28 de 11 de 2016.] https://dialnet.unirioja.es/descarga/articulo/4364562.pf.

**Gómez, Rosa María Yáñes. 2011.** *Introducción a las tecnologías de Clustering en GNU/Linux.* s.l. : Version 1.0, 2011.

**González Duque, Raúl . 2008.** *Python para todos.* España : s.n., 2008. Tutorial.

**Gutiérrez Rendón, Erick Samuel. 2010.** *Propuesta de implementación de un clúster de altas prestaciones.* 2010. Tesis.

**Hurtado Cuellar, Alberto. 2011.** *Propuesta de un servidor Web Apache 2 sobre un Cluster en Linux.* Santa Clara : s.n., 2011. Trabajo de Diploma.

**Kai Hwang, Khiwei Xu. 1998.** *Scalable Parallel Computing.* 1998.

**Luna, Denisse Aidee Ibañes. 2009.** [En línea] 2009. [Citado el: 27 de 11 de 2016.] www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/1182/Tesis.pdf?seq.

**Matos Faure, Yanislei y Feitas Cruz, Arel Luis. 2013-2014.** Repositorio Digital. [En línea] 2013-2014. [Citado el: 27 de 11 de 2016.] https://repositorio.uci.cu/jspui/handle/ident/9166.

**Miguel, Rogel Alfredo. 2012.** *Desarrollo de una infraestructura de redundancia para servidores Proxy GNU/LINUX en la intranet de la Facultad de Ciencias.* Riobamba-Ecuador : s.n., 2012. Tesis .

**Miranda Domínguez, Laura Elida, y otros. 2014.** Repositorio digital. [En línea] junio de 2014. https://repositorio.uci.cu/jspui/handle/ident/8887.

**Obiol González, Yisel y Ferrer Durruthy, Dayrol Lázaro. 2015.** Repositorio Digital. [En línea] 2015. [Citado el: 24 de 2 de 2017.] https://repositorio.uci.cu/jspui/handle/123456789/7062.

**Padilla, Flavio Mauricio Gallardo. 2011.** Scribd. [En línea] 5 de 2011. [Citado el: 5 de 12 de 2016.] https://es.scribd.com/doc/57937293/Balanceo-de-Carga-en-Centos.

**Palma Pérez, Nurisel. 2013.** Repositorio digital. [En línea] 2013. [Citado el: 24 de 11 de 2016.] https://repositorio.uci.cu/jspui/handle/ident/8429.

**Paredes, Rogel Alfredo Migues. 2012.** *Desarrollo de una infraestructura de redundancia para servidores Proxy GNU/LINUX en la intranet de la Facultad de Ciencias.* Riobamba-Ecuador : s.n., 2012.

**Peres Echebarria, Adan y Norberto Garces, Jose Raul. 2008.** Repositorio digital de la Universidad Veracruzana. [En línea] 2008. http://cdigital.uv.mx/handle/123456789/40290.

**Pfister, Gregory F. 1998.** *In search of clusters.* Prentice-Hall : 2da edicion, 1998.

**Pierra Fuentes, Allan. 2011.** *NOVA, DISTRIBUCIÓN CUBANA DE GNU/LINUX. REESTRUCTURACIÓN ESTRATÉGICA DE SU PROCESO DE DESARROLLO.* La Habana : s.n., 2011. Trabajo final presentado en opción al título de Máster en Informática Aplicada.

**Pressman, Roger. 2002.** *Ingeniería del sofware, un enfoque práctico.* 2002.

**Pressman, Roger S. 2009.** *Software Engineering. A practitioner´s Approach.* New York : McGraw Hill, 2009. ISBN 978-0-07-337597-7.

**Prieto, Alberto, Lloris, Antonio y Torres, Juan Carlos. 2002.** *Introducción a la Informática.* McGraw-Hill Interamericana de España : 3era edición, 2002.

**Red Hat. 2012.** Red Hat. [En línea] Linux Fundations, 2012. [Citado el: 12 de 2 de 2017.] http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html.

**Rosales Rosa, Eugenio. 2011.** Manual de usuario de Nova para Servidores 2011. *Humanos.* [En línea] 9 de 2011. [Citado el: 15 de 2 de 2016.] http://humanos.uci.cu/download/manual-de-usuario-de-nova-para-servidores-2013/.

**Rubio Sapiña, Aurelio. 2012.** *Alta disponibilidad en servidores y optimización de recursos de hardware a bajo coste.* Valencia : s.n., 2012. Tesis de Maestría.

**Sacha Krakowiak. 2007.** ObjectWeb - What's Middleware. *ObjectWeb.* [En línea] 2007. [Citado el: 17 de 11

de 2016.] http://middleware.objectweb.org/index.html.

**Sánchez, Tamara Rodríguez. 2015.** *Metodología de desarrollo para la.* La Habana : versión 1.2, 2015.

**Sarmiento, Johana. 2016.** UML: Diagrama de despliegue. *Visión general de los diagrama de despliegue.* [En línea] 16 de Febrero de 2016. [Citado el: 4 de 1 de 2017.] http://umldiagramadespliegue.blogspot.com/.

**Sommerville, Ian. 2005.** SlideShare. [En línea] 2005. [Citado el: 25 de 2 de 2017.] https://es.slideshare.net/jasc_584/ingenieriadesoftware-iansommerville7maedicion-9417118.

**Stallman, Richar M. 2004.** *Software libre para una sociedad libre.* Madrid : Traficantes de Sueños, 2004.

**Tramullas, Jesús y Garrido, Piedad. 2006.** Fundamentos. [aut. libro] Piedad Garrido Jesús Tramullas. *Software libre para servicios de información digital.* Madrid : Pearson Prentice Hall, 2006.

**Universidad de las Ciencias Informáticas.** Mejora de Procesos de Software. [En línea] Universidad de las Ciencias Informáticas. [Citado el: 6 de 2 de 2017.] http://mejoras.prod.uci.cu/.

**Unsignet Integer Limited. 2017.** DistroWatch.com. [En línea] 2017. [Citado el: 27 de 5 de 2017.] http://distrowatch.com/dwres.php?resource=major.

**Vázquez, José María Morales. 2012.** *Diseñando Sistemas de Alta Disponibilidad y Tolerantes a Fallos.* Madrid : s.n., 2012.

**Wilson Armando, Acero Quilumbaquín. 2016.** *DISEÑO E IMPLEMENTACIÓN DE UN CLUSTER DE ALTA DISPONIBILIDAD PARA VIRTUALIZAR LOS SERVIDORES DE ADQUISICIÓN Y PROCESAMIENTO DE DATOS DEL INSTITUTO GEOFÍSICO.* Quito : s.n., 2016. PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN .