Universidad de las Ciencias Informáticas

Facultad 1

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

Título:

Módulo para administrar el servicio de red desde HMAST

Autor:

Raciel Betancourt Bueno

Tutores:

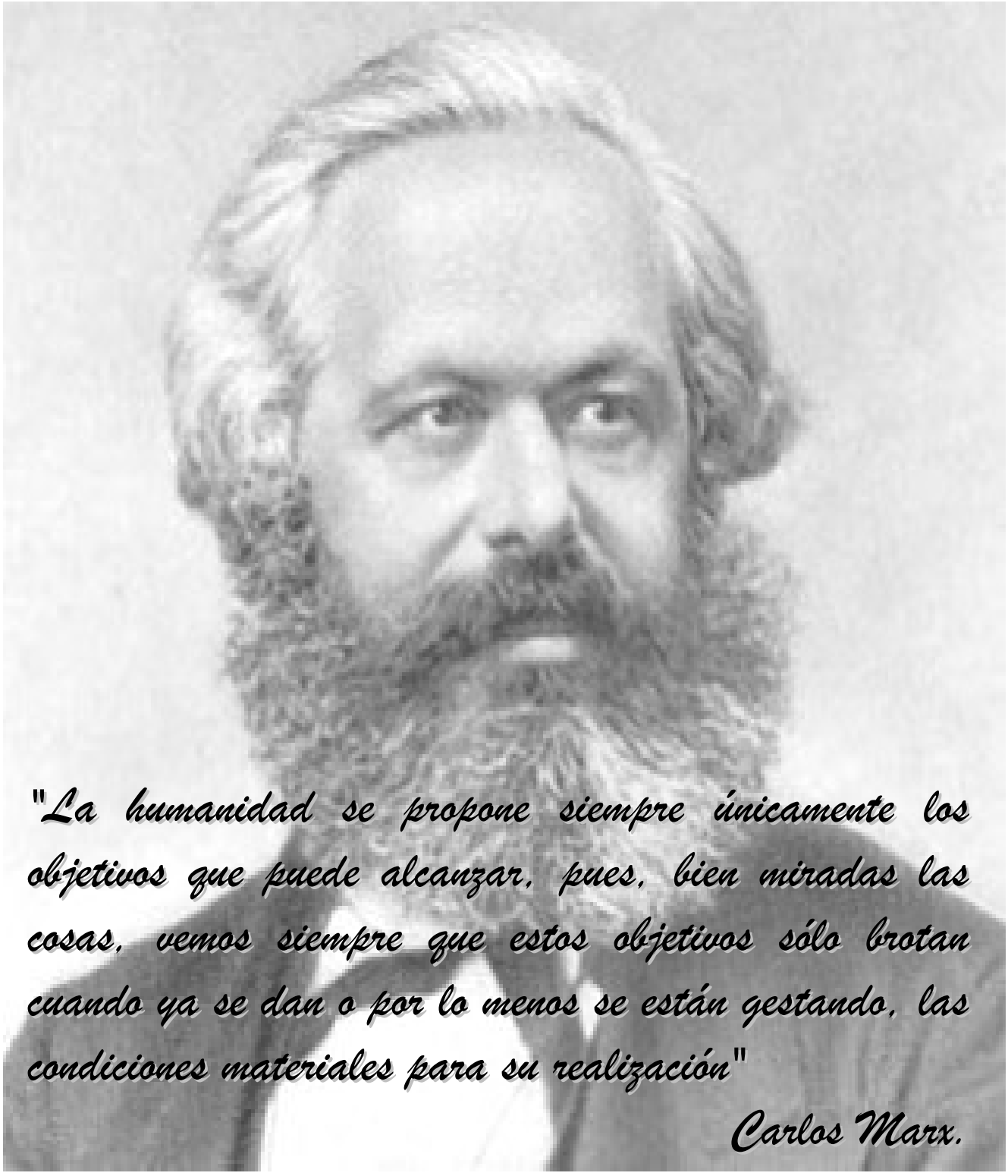
MSc. Osiris Perez Moya

Ing. Abel García Vitier

Ing. Pablo Soria Acosta

La Habana, Cuba.
Junio de 2014.

Pensamiento



Declaración de autoría

Declaro ser el único autor de este trabajo y cedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de ___ del año 2014.

_____	_____
Raciel Betancourt Bueno	MSc. Osiris Perez Moya
_____	_____
Ing. Abel García Vitier	Ing. Pablo Soria Acosta

Agradecimientos

Sinceramente, tengo que agradecer a muchas personas que me han ayudado a lograr este gran sueño que todo universitario quiere alcanzar. A las personas que estuvieron conmigo, tanto en las buenas como en las malas como debe ser si fueron amistades reales, siempre las recordaré por lo que significaron y significan. Los que no estuvieron conmigo porque no pudieron o porque no quisieron cuando yo los necesite, conmigo pueden contar para lo que necesiten.

Quiero agradecer primeramente a todos los compañeros de estudio, a aquellas personas que he conocido en estos años de universidad: estudiantes, profesores y trabajadores en general y a mis compañeros y amistades de toda la vida.

Agradecer a mi gente, a los que desde el comienzo en la UCI estuvieron conmigo, los recuerdo y los quiero de la misma forma y que las cosas vividas en el pasado nunca las voy a olvidar aunque a mucho de ellos no los veo hace mucho tiempo o posiblemente no los vuelva a ver.

Agradecer a mis tutores, Pablo, Abel y Osiris por haberme ayudado en todo lo que necesitaba y compartir su conocimiento conmigo. De forma especial un agradecimiento a Pablo, Reidiel, El Flaco y Yadiel 'El Mellizo' por estar ahí cuando los necesite.

Por ser lo último en mencionar, es lo más importante y a quienes más tengo que agradecer. 'La Familia' mi gran familia, este regalo tan grande que Dios nos hace.

Dedicatoria

Esta trabajo yo se la dedico a:

Mi mamá Nay por traerme al mundo, por darme todo lo suyo, compartir conmigo lo que tiene y hasta lo que no tiene. Por ser tan exigente y al mismo tiempo tan flexible en el momento preciso y exacto. Por comprenderme en cada momento, por darme amor, cariño y todo lo que viene después de esto desde que me nací hasta hoy porque yo se que para ti sigo siendo un niño. Mi hermana Nayla, por darme el amor que otra persona no hubiese podido dar en este mundo, por dormir conmigo cuando menos me lo esperaba, por despertarme a las 7 de la mañana cuando más sueño tenía (...), cuánto me gustaría que supieras y entendieras todo lo que significas para mi. Mi mamá Lourdes por criarme no como su nieto y sí como hijo, por darme lo poco o mucho que ha tenido, por no permitir desde pequeño que le dijera abuela y por aquellos regaños y aquellas palizas cuando me lo merecía.

Mi Tía Neli por ser mi tercera madre, por protegerme y en ocasiones cuidarme más que a su propia hija, por comprenderme y ser tan recta y cuando debías y cuando no debías. A mi prima Giselli, que aunque estas muy lejos de aquí, te agradezco por toda tu ayuda, por darme tanta fuerza, por escucharme y darme tu apoyo, por decirme siempre lo que piensas de las cosas aunque la situación fuera dura y estar conmigo en el momento exacto.

Resumen

La migración[1] de los servicios telemáticos a aplicaciones de código abierto en Cuba, se ve afectada por la carencia de soluciones integradoras que se ajusten a las necesidades tecnológicas de las entidades. El departamento de Servicios Integrales en Migración, Asesoría y Soporte (SIMAYS) comenzó a desarrollar en el 2012 la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST), la cual tiene como objetivo ofrecer una solución a lo antes descrito. Sin embargo, esta herramienta no incorpora funcionalidades para gestionar el servicio de red. El principal resultado de esta investigación, es el desarrollo de un módulo en HMAST para administrar una infraestructura de red, pensado en las facilidades de uso que brinda a los administradores de servicios telemáticos de las entidades del país. Fue llevado a cabo un estudio del estado del arte, en el que se analizaron tecnologías y componentes relacionados con este servicio. A partir de un desarrollo ágil, guiado por la metodología SXP, fueron implementadas funcionalidades que se integran correctamente a la arquitectura de HMAST. Las pruebas realizadas al sistema, arrojaron resultados satisfactorios, evidenciando que las funcionalidades se encuentran correctamente implementadas en un 100%.

Palabras claves: administración, código abierto, configuración, interfaz de red, migración, software libre.

[1] ***Migración***: desplazamiento o traslado de un grupo social desde su lugar de residencia u origen habitual hacia otro donde considere que mejorará su calidad de vida en un entorno social, político y económico diferente.

Índice

Introducción....1
Estructura del documento....7
Capítulo 1: Fundamentación teórica....8
1.1 Servicio de red....8
1.1.1 Interfaces de Red....9
1.1.2 Trabajo con las Interfaces de red....11
1.2 Herramientas homólogas para administrar el servicio de red....14
1.2.1 Herramientas estudiadas....16
Webmin 1.6.30....17
Zentyal 3.2....19
Network Manager....23
1.2.3 Resultado del estudio....26
Herramienta más madura....27
Conclusiones parciales....28
Capítulo 2: Análisis y diseño de la solución propuesta....29
2.1 Propuesta del sistema a desarrollar....29
2.1.1 Descripción del negocio....30
2.2 Requisitos....31
2.2.1 Lista de reserva del producto....31
Requisitos funcionales....32
Requisitos no funcionales....33
2.2.2 Historias de usuario....33
2.3 Arquitectura de HMAST....34
2.3.1 Consideraciones para implementar un módulo para HMAST....36
2.4 Arquitectura del módulo de red para HMAST....36
2.4.1 Tecnologías asociadas al desarrollo del módulo de red....37
2.5 Diseño del módulo Administración del servicio de red y uso de patrones....38
2.5.1 Diagrama de paquetes....38
2.5.2 Diagrama de clases....41
2.5.3 Patrones de diseño utilizados....42
Patrones GRASP....42
Patrones GOF 44
Conclusiones parciales....45
Capítulo 3: Implementación y pruebas....46
3.1 Planificación de Iteraciones....46
3.2 Implementación....47
3.2.1 Principales clases utilizadas....47
3.2.2 Tareas de ingeniería....47
3.2.3 Estándares de codificación....53
3.3 Pruebas de software....54

3.3.1 Pruebas Unitarias....55
3.3.2 Pruebas de Aceptación....57
3.3.3 Resultados de las pruebas....64
3.4 Diagrama de Componentes....64
3.5 Diagrama de Despliegue....64
Conclusiones parciales....65
Conclusiones....66
Recomendaciones....66
Bibliografía referenciada....67
Bibliografía....67
Bibliografía General....69
Glosario de Términos....71
Glosario de Términos....71
Anexo 1: Modelo de dominio....74
Anexo 2: Lista de Reserva del Producto....74
Anexo 3: Historias de Usuario....76
Anexo 4: Diagrama de las clases Entidades....85

Índice de tablas

Tabla 1: Comparación entre herramientas homólogas....26
Tabla 2: Requisitos funcionales....32
Tabla 3: Descripción de Historias de Usuario....34
Tabla 4: Lista de Reserva del Producto....75
Tabla 5: HU_3 Configurar interfaces físicas de red....78
Tabla 6: HU_4 Configurar interfaces de red virtual....80
Tabla 7: HU_1 Reiniciar el servicio de red....81
Tabla 8: HU_2 Mostrar descripción de las tarjetas de red....82
Tabla 9: HU_5 Balancear carga entre interfaces de red física....84

Introducción

Desde el surgimiento de la especie humana, el hombre tuvo la necesidad de comunicarse e intercambiar información para desarrollar habilidades y adquirir experiencias que le garantizarían la superioridad con respecto al resto de los animales. Este intercambio de información en un principio rústico, fue evolucionando y perfeccionándose e innumerables fueron los avances que obtuvo el hombre con el propósito de recopilar y hacer llegar la información a mayor cantidad de personas en el menor tiempo posible. Desde el telégrafo hasta las transmisiones de radio y televisión, desde el papel y las cintas magnéticas hasta las computadoras. Ha sido extraordinario el desarrollo alcanzado en los medios de transmisión, almacenamiento y procesamiento de la información; dividido en dos ramas fundamentales: las telecomunicaciones y la informática.
En sus inicios ambas disciplinas evolucionaron de forma paralela pero distantes entre sí, donde cada una utilizaba los adelantos tecnológicos y científico-técnicos en su propio desarrollo como un ente aislado. Sin embargo, el surgimiento de las redes de computadoras, a finales de los 60 del siglo pasado, dio al traste con este divorcio; dando origen a la telemática[2]. Lo que le propicia cubrir un campo científico y tecnológico de una considerable amplitud, englobando el estudio, diseño, gestión y aplicación de las redes y servicios de comunicaciones, para el transporte, almacenamiento y procesado de cualquier tipo de información (datos, voz, vídeo, etc.), incluyendo el análisis y diseño de tecnologías y sistemas de conmutación.

Comienza entonces a surgir una amplia gama de servicios telemáticos, con el objetivo de procesar e intercambiar información entre computadoras o sistemas informáticos. Entre estos servicios se encuentran: el sistema de nombre de dominio o *DNS*; el protocolo de configuración dinámica de host o *DHCP*; sincronizar relojes de sistemas computarizados con el (protocolo de tiempo en las redes o *NTP*); conectarse a un ordenador desde otro lugar (protocolo de terminales virtuales o *telnet*); transferir ficheros entre una computadora local y una remota (protocolo de transferencia de archivos o *FTP*); leer e interpretar texto, imágenes, sonido y video desde una máquina remota (protocolo de transferencia de hipertexto o *HTTP*); intercambiar mensajes de correo electrónico (*e-mail*); acceso a grupos de noticias y

[2] ***Telemática:*** Disciplina científica y tecnológica, originada por la convergencia entre las tecnologías Telecomunicaciones e Informática.

foros de debates (*news*) así como conversaciones en tiempo real o mensajería instantánea (Chat, *IRC*) entre otros.

A finales de los años 70, las grandes empresas y compañías productoras de software comenzaron a hacer privativos los productos que comercializaban, forzando a los usuarios a aceptar condiciones restrictivas que impedían realizar modificaciones a los mismos (**solucionar errores de código o del software**) o redistribuirlo (Martínez, 2006). Con este antecedente en enero de 1984 Richard Mattews Stallman, anunció como respuesta a las tendencias de licenciamiento de software del momento, la creación de GNU, un acrónimo que significa *GNU is Not Unix*, cuyo objetivo era crear un sistema operativo basado en UNIX pero libre (Stallman, 2010).

El término de libertad no se refiere a que no posee precio, sino a las 4 libertades del programa: libertad de estudiarlo y adaptarlo; libertad de distribuir copias; libertad de mejorar y publicar cambios y libertad de usar el programa con cualquier propósito. Para lograr que el software cumpliera las 4 libertades una vez que estuviese en manos de los usuarios y evitar restricciones posteriores del mismo, en 1985 se crea la a Fundación de Software Libre (en inglés *Free Software Fundation o FSF*), con el objetivo de proveer soportes logísticos, legales y financieros al proyecto GNU (Benjamin, 2011). Para finales de 1980 se crea la Licencia Pública General (GPL) para servir de sostén legal a todo el software que se iba creando, con el término de ***copyleft*** o copia permitida en contrapartida al ***copyright*** (Stallman, 2010). El proyecto GNU protege mediante ***copyleft*** casi todo el software que producen, con el propósito de dar a cada usuario las libertades que el término 'Software Libre[3] (SWL)' implica.

Con un fuerte impacto en el ámbito del aprendizaje de la programación y en su proyección hacia el futuro; la utilización de software libre permite fomentar términos como: ética, no discriminación, colaboración, ciencia, competitividad, seguridad, eficiencia, solidaridad, privacidad y libertad. Representando en sí la no utilización de productos informáticos que demanden la autorización de sus propietarios para su uso. Permite su adaptación a los contextos de aplicación al contar con su código fuente, lo cual garantiza un mayor por ciento de efectividad, además de la corrección de sus errores de código fuente.

El software libre más que una forma de producción y distribución de software es un movimiento tecnológico que se ha extendido por el mundo. Además de lo indicado, se puede ver como un modelo de

[3] ***Software Libre:*** es la denominación del software que respeta la libertad de todos los usuarios que adquirieron el producto y, por tanto, una vez obtenido el mismo puede ser usado, copiado, estudiado, modificado, y redistribuido libremente de varias formas.

negocio para una nueva industria basada en servicios informáticos, más que en productos. Actualmente ha habido una gran inclinación por la utilización de software libre como vía alternativa en la búsqueda de soluciones, como por ejemplo en la administración de los servicios telemáticos.
Para los países subdesarrollados o de bajos ingresos económicos, el software libre es la alternativa ideal para potenciar el desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC). Actualmente en el mundo existe una serie de países en los cuales, sus administraciones públicas, han migrado total o parcialmente sus servidores y sistemas de escritorio; con el objetivo de que el entorno de trabajo debe suplir plenamente las funcionalidades requeridas por los usuarios y remplacen las aplicaciones que actualmente ofrecen los sistemas privativos.

Las estadísticas existentes por zona indican, que aunque en diversas áreas geográficas se ha expandido la migración a software libre, es en Oriente y Europa donde se concentra la mayor cantidad de servidores, computadoras personales y estatales que actualmente la han aplicado (Castillo, 2013). Como ejemplo de lo revelado se tiene que, en el área de América Latina han realizado migración a sus empresas países como: Argentina, Brasil, Chile, Cuba, Ecuador, México, República Dominicana, Perú, Uruguay y Venezuela; en otras regiones del mundo lo han hecho también países como Alemania, China, España y Francia conociéndose experiencias exitosas (cepEF, 2009). Un dato de lo planteado anteriormente, es que en un corto tiempo en Munich, Alemania se han migrado 14.000 computadoras del estado federal, evidenciando esto la preferencia por parte de las instituciones hacia el uso de aplicaciones GNU/Linux[4]. Actualmente, los servidores de red con software libre dominan el 65% del mercado mundial. Según una encuesta realizada recientemente por la *International Business Machines* (IBM), desde el 2007 el 83% de las empresas planean realizar trabajos basados en GNU/Linux (Castillo, 2013).

Por sus condiciones para la administración de los servicios telemáticos, el software libre es de gran ayuda y beneficio por las facilidades que otorga a sus usuarios. Algunas ventajas pueden ser más apreciadas por los usuarios de computadoras de uso doméstico, otras por los equipos de desarrollo, otras por las empresas, organismos e instituciones públicas. Entre las más importantes para los gobiernos y empresas se encuentran, la **independencia tecnológica:** ya que el estado deja de tener sus sistemas controlados por una entidad externa (en ocasiones empresas extranjeras). El **control de la información:** pues al tener la libertad de inspeccionar el mecanismo de funcionamiento del software, la manera en que almacena los

[4] ***GNU/Linux:*** Uno de los términos utilizados para referirse al sistema operativo libre homólogo a Unix que usualmente utiliza herramientas de Sistema GNU.

datos y la posibilidad de modificar estos, queda en manos del estado la llave del acceso a la información en vez de quedar en manos privadas. La **confiabilidad y estabilidad:** el software libre, al ser público, esta sometido a la inspección de una multitud de personas, que pueden buscar problemas, solucionarlos y compartir la solución con los demás. Otra ventaja es la **seguridad*:*** el estado puede fiscalizar que su software no tenga puertas traseras[5], voluntarias o accidentales, y que pueda cerrarlas en caso de encontrarlas y sin menos peso la facilidad **económica:** pues cuenta con normas abiertas para la administración electrónica (Martínez, 2012).

Como el uso de la informática en Cuba ha estado soportado en sistemas operativos Windows y basado en la utilización de tecnologías privativas; por la situación económica y comercial que presenta el país, aspectos políticos, sociales y tecnológicos; la brecha digital, causas sociales y otros factores referentes a la prohibición de su uso, redistribución o modificación, se imposibilita utilizar estas tecnologías, siendo necesario el uso del software libre (Paumier *et al*, 2008).

En la lucha por lograr la soberanía tecnológica, garantizar la democratización y apropiación social de las tecnologías de información; en el 2004, el Consejo de Ministros orienta una migración paulatina de los Órganos y Organismos de la Administración Central del Estado (OACE) hacia tecnologías de código abierto[6]. Pretendiéndose que todos los equipos de la red, tanto clientes como servidores, ejecuten sólo aplicaciones de código abierto, basado en GNU/Linux como sistema operativo base. Sin embargo, durante el proceso de migración de las entidades cubanas, se detectó alta dependencia tecnológica debido a que muchas de estas usan en su infraestructura de servidores, sistemas operativos basados en las diferentes versiones de ***Windows Server***. Además, las herramientas libres existentes para la administración de servicios telemáticos son complejas y menos intuitivas que las privativas. Lo que constituye un reto para los que desarrollan software libre.

Uno de los centros de referencia en la conducción de procesos de migración en las entidades cubanas es la Universidad de las Ciencias Informáticas (UCI), en donde el Centro de Software Libre (CESOL) es el motor impulsor de esta tarea. SIMAYS uno de los departamentos de CESOL tiene como tarea fundamental brindar Servicios Integrales para la Migración Asesoría y Soporte técnico. Por la necesidad de crear un

[5] ***Puerta trasera:*** Líneas de código ocultas en sistemas informáticos que permiten obtener o introducir información con o sin el consentimiento de los usuarios.

[6] ***Código abierto:*** (en inglés ***open source***) término con el que se conoce al software distribuido y desarrollado libremente. Tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas, las cuales destacan en el llamado software libre.

sistema que automatice el proceso de migración que se realiza y facilitar el trabajo de los administradores de servicios telemáticos una vez migrados o instalados estos, se creó la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST).

Actualmente en los OACE, a los administradores de servicios telemáticos se les dificulta la gestión del servicio de red en los servidores cuando necesitan configurar interfaces de red física, interfaces de red virtual y realizar balanceo de carga entre interfaces de red física. Como los servidores no poseen alguna aplicación con interfaz de usuario para gestionar este servicio, las configuraciones mencionadas anteriormente tienen que realizar escribiendo directamente en el fichero (archivo de configuración) correspondiente a este servicio a través de la terminal. Esta situación propicia que el encargado de realizar esta tarea tenga que memorizar una serie de comandos y sentencias. Además, cuando se envían o reciben grandes volúmenes de datos aumenta la carga funcional de las tarjetas de red y en ocasiones atrasos en la ejecución de algunas funciones. El trabajo manual (no automatizado) realizado por los administradores de las entidades del país, por su complejidad es un inconveniente que frena la rapidez con que se gestiona el servicio de red.
Partiendo de que HMAST no ofrece las funcionalidades mencionadas anteriormente para gestionar el servicio de red (Soria y Castillo, 2012) y que actualmente estas son necesitadas en los OACE, se identifica el siguiente **problema a resolver**:

¿Cómo administrar el servicio de red desde HMAST para garantizar una gestión estable de los servicios telemáticos en los OACE ?

El **objeto de estudio** del presente trabajo de investigación se centra en la administración del servicio de red. Enmarcándose en el **campo de acción** administración del servicio de red en GNU/Linux.

Por lo que se propone como **objetivo general** desarrollar un módulo en HMAST que permita la administración del servicio de red utilizando tecnologías libres.

En el marco de esta investigación se ha desglosado el objetivo general en los siguientes **objetivos específicos**:

- Caracterizar las herramientas para la administración del servicio de red en GNU/Linux.
- Diseñar la solución propuesta.

- Implementar las funcionalidades diseñadas.
- Evaluar las funcionalidades implementadas.

Para dar cumplimiento a los objetivos específicos se planifican las siguientes **tareas de investigación**:

- Revisión de bibliografía asociada al servicio de red en GNU/Linux y soluciones desarrolladas con tecnologías de código abierto.
- Definición de los requisitos funcionales y no funcionales a desarrollar.
- Análisis de la arquitectura de HMAST.
- Diseño del sistema acoplado a la arquitectura de HMAST.
- Codificación de las funcionalidades en el lenguaje definido.
- Diseño y ejecución de casos de prueba a las funcionalidades implementadas.

En la presente investigación se plantea como **idea a defender** que: la creación de un módulo que gestione el servicio de red desde HMAST permitirá su administración fácil y rápida proporcionando facilidades de uso para los administradores de servicios telemáticos de las instituciones cubanas.

Para el desarrollo de la investigación se tuvieron en cuenta los siguientes **métodos científicos**:

Analítico - Sintético: utilizado en la investigación de los servicios de red existentes y los componentes que estos utilizan; ayudando a escoger de las diversas tecnologías, los aspectos más apropiados para utilizarlos en el desarrollo de la solución, ya que relacionando hechos conocidos aparentemente aislados se formuló una teoría que unificó diversos elementos conocidos, donde se sintetiza lo aprendido para establecer una explicación que se someterá a prueba.

Lógico inductivo: el mismo posibilitó determinar el razonamiento que partiendo de un caso particular dentro del estudio, se elevó y dio paso a conocimientos generales. Uno de los casos, es que cuando se inicia la investigación ya existen conocimientos previos del tema a tratar.

Estructura del documento

El documento se encuentra compuesto por introducción, tres capítulos, conclusiones, bibliografía, glosario de términos y anexos. La estructura de los capítulos se define a continuación:

Capítulo 1. Fundamentación teórica

Marco teórico asociado a la administración del servicio de red, centrado en el estado del arte del objeto de estudio. Se analizan herramientas y aplicaciones que permiten administrar el servicio de red, donde se describen funcionalidades necesarias para lograr un mayor entendimiento del tema. Son tomados en cuenta parámetros específicos para realizar comparaciones entre software de código abierto, y es seleccionado entre estos el que más aporta a la investigación en cuestión.

Capítulo 2. Análisis y diseño de la solución propuesta

Se realiza el análisis y diseño de un módulo para administrar el servicio de red. Se presenta la concepción inicial del módulo, así como la definición de los requisitos de integración, funcionales y no funcionales. Son descritas las Historias de Usuario y a través de estas se confeccionan los diagramas de clases y de paquetes acoplados a la arquitectura del sistema. Se mencionan las herramientas, tecnologías, lenguajes de programación y metodología definidos para el desarrollo.

Capítulo 3. Implementación y pruebas

Son definidas las iteraciones en las que se implementarán las Historias de Usuario, las tareas de ingeniería y la distribución física del sistema. Se describen los casos de pruebas diseñados y se muestran los resultados de las pruebas ejecutadas, validando que las funcionalidades descritas y desarrolladas dan cumplimiento a los requisitos planteados. Además, que el desarrollo de este sistema integrado a HMAST cumple con condiciones que pueden facilitar el trabajo de los administradores de servicios telemáticos en las instituciones cubanas.

Capítulo 1: Fundamentación teórica

En este capítulo se realiza un estudio para obtener conocimientos referentes al servicio de red y sus características particulares. Además, de quedar especificado cómo se gestiona la red en una estación de trabajo o un servidor a través de la configuración de sus interfaces de red. Son documentados a través de un modelo de madurez para calificar software de código abierto, los criterios necesarios para realizar una correcta investigación y comparar herramientas homólogas. Se detalla con claridad características y funcionamiento las de algunos software de código abierto que permiten la administración del servicio de red en servidores soportados por sistemas operativos GNU/Linux. De estos se buscan, procesan y toman en cuenta aspectos positivos que podrían dar una mejor visión de lo que se quiere realizar para solucionar el problema existente; además se elige entre estos cual es el más completo.

1.1 Servicio de red

En la actualidad, con la expansión y desarrollo que han tenido las nuevas tecnologías e Internet, se ha generalizado la utilización de las redes de computadoras[7], siendo muy usadas ya sea en hogares o centros de trabajo. Se hace prácticamente necesario para una institución o entidad, disponer por lo menos de una red interna, que permita procesar información (vídeos, sonido, imágenes, entre otros) o incluso conectarse u ofrecer sus servicios en Internet.

Red de computadoras: llamada red de ordenadores o red informática, es un conjunto de computadoras y/o dispositivos conectados por enlaces de un medio físico (**medios guiados**) o inalámbricos (**medios no guiados**) que comparten información (**archivos**), recursos (**CD-ROM, impresoras, etc.**) y servicios (**e-mail, chat, juegos, etc**) con otros dispositivos (Poratti, 2004).

Las redes, según la propiedad a la que pertenezcan, pueden ser públicas o privadas. La **red pública** es definida como una red que pertenece a organismos estatales y puede ser usada por cualquier persona que lo solicite mediante el correspondiente contrato, no están configuradas con clave de acceso personal.

[7] ***Red de computadoras:*** conjunto de equipos (computadoras y/o dispositivos) conectados por medio de cables, señales, ondas o cualquier otro método de transporte de datos, que comparten Información.

La **red privada** es definida como una red que solo puede ser usada por algunas personas, empresas u organizaciones de índole privada ya que están configuradas con clave de acceso personal.

Las redes según su cobertura o alcance pueden clasificarse en: **red de área personal** (en inglés *Personal Area Network* o PAN), que es una red de ordenadores usada para la comunicación entre los dispositivos de la computadora que se encuentran cerca de una persona; **red de área local** (en inglés *Local Area Network* o LAN) es un tipo de red limitada a un área especial relativamente pequeña; **red de área metropolitana** (en inglés *Metropolitan Area Network* o MAN) es una red que conecta las redes de varias áreas, pero no se extiende más allá de los límites; **red de área amplia** (en inglés *Wide Area Network* o WAN) es una red de comunicaciones de datos que cubre un área geográfica relativamente amplia (de una cuidad a otra, de un país a otro o de un continente a otro); **red de área de almacenamiento** (en inglés *Storage Area Network* o SAN) es una red concebida para conectar servidores. Existen otros tipos de redes como la **red de área local virtual** (en inglés *Virtual LAN* o VLAN) y la **red del área del campus** (en inglés *Campus Area Network* o CAN) (Stalling, 2000).

Para lograr el enlace entre las computadoras y los medios de transmisión (cables de red o medios físicos para redes cableadas e infrarojos ó radiofrecuencias para redes inalámbricas), es necesaria la intervención de una tarjeta de red (en inglés *Network Card Interface* o NIC). Con esta tarjeta se pueden enviar y recibir paquetes de datos desde y hacia otras computadoras, empleando un protocolo para su comunicación. A cada tarjeta de red le es asignado un identificador único por su fabricante, conocido como medio de control de acceso (en inglés *Media Access Control o* MAC), que consta de 48 Bits (6 Bytes). Dicho identificador permite direccionar el tráfico de datos de la red del emisor al receptor adecuados (Poratti, 2004).

1.1.1 Interfaces de Red

Interfaz de red física: Es la vía utilizada para la conexión de una computadora con el medio de transporte de la red. Esto puede ser un módem, una tarjeta de red, un puerto serie, enlace infrarrojo, una conexión inalámbrica, etc.

Las interfaces de red son definidas por el protocolo TCP/IP, a través del cual se accede a la diversidad de dispositivos que pueden usarse en un entorno de red. Ofrecen un conjunto de operaciones que son las mismas para todos los tipos de hardware y básicamente trata con el envío y la recepción de paquetes. Por

cada dispositivo periférico de red tiene que haber una interfaz correspondiente. Cuando se inicia el sistema, estas interfaces se crean automáticamente por cada tarjeta de red que tenga el hardware; ejemplo de esto es el controlador de ***Ethernet*** *que* crea las interfaces ***eth*** (Dawson y Kirch, 2000).

En GNU/Linux existen numerosas interfaces para conectarse a la red, entre las más comunes se encuentran: la interfaz de bucle local o *loopback* denominada ***lo****;* la ***Ethernet*** denominada ***ethx***; la Punto a Punto (en inglés *Point-to-Point Protocol* o PPP) denominada ***pppx***; la Interfaz de Fibra de Datos Distribuido (en inglés *Fiber Distributed Data Interface* o FDDI*)* denominada ***fddix.*** También existen otras como la ***wlanx*** utilizada para conexiones Wi-Fi y la ***hcix*** para conexiones por Bluetooh. Estos nombres son utilizados por los comandos para propósitos de configuración y la asignación del número de identificación u orden '***x'*** es para cuando se quiere especificar un dispositivo físico determinado, que posteriormente permitirá comunicarse con otros dispositivos en la red (Jorba y Suppi, 2004).

loopback *(****lo****):* tiene una dirección IP reservada para el tráfico local; esta dirección es 127.0.0.1. Actúa como un circuito cerrado y tiene como propósito direccionar el tráfico que se origina desde la maquina local, que es además, su destino final. Cualquier paquete IP enviado a esta interfaz por el protocolo de control de transmisión (en inglés *Transmission Control Protocol* o TCP) o el protocolo de datagramas de usuario (en inglés (*User Datagram Protocol* o UDP) le será devuelto simplemente a cualquiera de ellos como si hubiese llegado desde alguna red. Esto permite desarrollar y probar software de red aunque no se esté usando una red real. Toda computadora ejecutando GNU/Linux tiene una interfaz ***lo*** configurada y activa (Jorba y Suppi, 2004).

eth**:** es el tipo más común de interfaz, la tarjeta ***Ethernet*** usada a través de las interfaces **eth** es el tipo de hardware más utilizado en LANs para conectarse a una red. Esto es debido al alto nivel de transmisión de información, que permiten múltiples equipos y son fáciles de implementar (Dawson y Kirch, 2000).

SLIP: son interfaces de líneas seriales, usadas para conectar computadoras aisladas a otras redes mediante un módem; pero pueden además, ser usadas para conectar dos computadoras directamente. Un problema que existe cuando se usan líneas seriales para conectar una red es que la transmisión serial de información no es muy rápida debido a limitaciones de hardware (Dawson y Kirch, 2000).

PPP: es otra interfaz de línea serial que usa el protocolo de punto a punto (ppp). Una interfaz PPP es creada cuando un enlace (***link***) serial es establecido usando ***ppp***. Este tipo de interfaz puede ser usado

para los mismos propósitos que la interfaz SLIP, pero una interfaz PPP es mucho más confiable que una interfaz SLIP y su uso es más popular (Jorba y Suppi, 2004).

Interfaz de red virtual: conocidas como direcciones IP secundarias, es la combinación de los recursos de red del hardware con los recursos de red del software en una única unidad administrativa. El objetivo de hacer virtuales las interfaces de redes consiste en facilitar un uso compartido de recursos de redes eficaz, controlado y seguro para los usuarios y los sistemas.

La interfaz de red virtual es utilizada para asignar otra IP a una tarjeta de red, posibilitando tener múltiples direcciones IP en una misma interfaz. Es decir, que una misma PC podría tener varias direcciones IP o conectarse con una única tarjeta de red a varias redes, ofreciendo servicios de forma independiente por cada una de estas interfaces (Oracle, 2011). Son útiles en los muros cortafuegos donde se pretende que un conjunto de equipos se conecten a Internet enmascarados con una dirección IP (una LAN) y otro conjunto de equipos lo hagan con una dirección IP distinta (Barrios, 2014).

1.1.2 Trabajo con las Interfaces de red

Antes de ser usada en una red TCP/IP, a una interfaz de red se le debe asignar una dirección IP que sirve como su identificador. Esta dirección es distinta al nombre de interfaz mencionado anteriormente (Perpiñan, 2012).

Para crear una infraestructura de red de computadoras se debe tener en cuenta la configuración de las interfaces de red de cada PC. En este proceso se otorgan direcciones lógicas (una o varias) o parámetros a un dispositivo físico o virtual específico, ya sea este estático (de forma manual) o dinámico (de forma automática), este último caso asigna las direcciones IP a través de un servidor DHCP. Además, se realizan configuraciones para obtener servidores de nombre de dominio o servidores de búsqueda respectivamente a través del servidor DNS. Para configurar las interfaces de red, uno de los comandos más utilizados es ***ifconfig***; a través de este se puede obtener la información de cada interfaz disponible (activa o no). En GNU/Linux las configuraciones de las interfaces de red se realizan en el fichero ***etc/network/intefaces,*** con este se pueden comunicar servicios como DNS y DHCP (Barrios, 2014).

Después de configurar las interfaces de red, se deberá comprobar si funcionan adecuadamente. Esto involucra dos tareas: comprobar la conexión a la red con el comando ***ping*** y rastrear las rutas de los paquetes con el comando ***traceroute***.

El comando ***ping*** permite comprobar si existe conexión entre dos máquinas conectadas a la red. Si funciona entonces la configuración (física y lógica) está correcta. Envía paquetes especiales al *host* de destino y espera una respuesta. En un caso más simple se indica el *host* de destino como parámetro del comando ***ping***.

El comando ***traceroute*** se emplea para diagnosticar el encaminamiento entre dos redes. Muestra la ruta que los paquetes de datos han seguido hasta la máquina de destino**.** Para ello ***traceroute*** envía tres paquetes con tiempo de vida de los paquetes (en inglés *Time To Live* o TTL*)* TTL=1 hacia el *host* de destino, incrementa el TTL en 2 y envía tres paquetes más y así sucesivamente. Los TTL se reducen en una unidad cada vez que logran pasar por el *router*. Cuando el TTL alcanza el valor cero, se envía un mensaje al *host* que está enviando los paquetes. De esta manera es posible recopilar información acerca de los *routers* por donde pasa la información hasta alcanzar el *host* destino.

Bonding: este controlador, creado por Donald Becker, está incluido en prácticamente todas las distribuciones de GNU/Linux y es una técnica que consiste básicamente en configurar varias tarjetas de red con la misma IP. Permite sumar las capacidades de varias interfaces físicas de red con el objetivo de crear una interfaz lógica. Al hacer esto, se logra que todas las tarjetas de red trabajen como si fueran una sola produciendo redundancia con balanceo de carga y tolerancia a fallos en la interfaz. Además, se puede utilizar para hacer agregación de enlaces y ancho de banda (Barrios, 2014).

Todas las tarjetas involucradas en el bonding comparten al mismo tiempo la dirección física (MAC), que se extrae de una de estas tarjetas. (Pedrero, 2007). La MAC del ***bond0*** (interfaz del controlador *bonding***)** será la MAC del primer dispositivo esclavo. Para configurar los canales bonding se debe instalar el paquete **ifenslave 2.6** el cual provee las herramientas de usuario para crear las interfaces de canales bonding en GNU/Linux.

El bonding de tarjetas de red es beneficioso en un ambiente LAN donde se necesita un servidor que deba soportar transferencia de grandes volúmenes de datos. Una de las ventajas más importantes es el **balanceo de carga***,* ya que se realizará esta operación entre todas las interfaces reales (por defecto

Round Robin), otro aspecto positivo es la **redundancia** pues si falla una tarjeta de red, los datos automáticamente irán sólo por las que estén en buen estado (Barrios, 2014).

Para realizar su configuración se deben establecer varios parámetros bajo los cuales funcionarán las interfaces en conjunto; estos son (*mode, miimon, downdelay, updelay* y *primary*):

downdelay: cantidad de milisegundos que se espera antes de declarar una interfaz 'caída' y asumir el control otra interfaz.

updelay: cantidad de milisegundos que se espera antes de declarar una interfaz 'activa'.

miimon: especifica cada cuántos milisegundos se debe supervisar el enlace (***link***) de la interfaz medi independiente (en inglés *Media Independent Interface* o **MII**) cuando se necesita alta disponibilidad para verificar si la interfaz está activa y si está conectado a la red.

primary: especifica la interfaz primaria.

mode: establece el modo bajo el cual funcionarán las tarjetas; los diferentes modos de bonding son:

- Modo 0 o **balance-rr**: Utiliza el método Round-Robin entre las tarjetas de red que forman el enlace. Es decir, transmite los paquetes en orden secuencial desde la primera tarjeta esclava hasta la última, y entonces vuelve a empezar por la primera de nuevo. Esta opción ofrece balanceo de carga y tolerancia a fallos.
- Modo 1 o **active-backup**: Sólo una de las tarjetas esclavas está activa. Si la tarjeta activa falla, otra tarjeta se vuelve activa y recibe el tráfico. Esta opción ofrece tolerancia a fallos.
- Modo 2 o **balance-xor**: Se aplica una política XOR basada en *dirección MAC origen* XOR *dirección MAC destino.* De esta forma se selecciona la misma tarjeta esclava para cada MAC destino. Esta opción ofrece balanceo de carga y tolerancia a fallos.
- Modo 3 o **broadcast**: Se retransmiten todos los paquetes a todas las tarjetas esclavas. Esta opción ofrece tolerancia a fallos.
- Modo 4 o **802.3ad**: 802.3ad permite agregar varios enlaces para conseguir un mayor ancho de banda. Todos los enlaces deben tener la misma velocidad y ancho de banda. Es necesario equipamiento de red que soporte 802.3ad. En redes esto se conoce como *Port Trunking* y en Cisco como LACP.
- Modo 5 o **balance-tbl**: El tráfico saliente se distribuye según la carga de trabajo en cada tarjeta esclava.
- Modo 6 o **balance-alb**: Incluye el balanceo del tráfico saliente (*Modo 5*) más el balanceo del tráfico entrante (Pedrero, 2007).

1.2 Herramientas homólogas para administrar el servicio de red

Actualmente existen varias aplicaciones y herramientas que permiten la administración de los servicios telemáticos de forma automatizada, estas pueden ser de tipo web, escritorio y otras que tienen su interfaz desarrollada para el trabajo en la terminal (Ncurse). Muchas permiten facilidades para los usuarios o administradores, pues sería un desafío trabajar con tantos archivos de configuración, comandos, directivas y sentencias. Algunas cuestiones generales a tomar en cuenta para seleccionar el mejor de varios software son las siguientes: **durabilidad del software**; **que nivel de estabilidad tiene o se espera**; **facilidad y factibilidad al agregarle funcionalidades al software** y **si tiene una comunidad activa que lo avale** (Rodríguez y Soria, 2010).

Para establecer una correcta investigación y darle mayor robustez al estudio de las herramientas que fueron definidas, se propone utilizar un ***modelo de madurez*** de proyectos de software de código abierto para definir procesos de evaluación, clasificación y elección final de un software determinado. Existe una amplia gama de ***modelos de madurez***, entre los que se encuentran: OSMM Capgemini, OSMM Navica, BRR y QSOS (Widdows y Duijnhouwer, 2003). Con la ayuda de uno de estos modelos, se podrán examinar las limitaciones y aspectos positivos y negativo que poseen estos sistemas.

OSMM - Capgemini: posee una licencia privada pero con autorización para su distribución. Fue desarrollado para comparar y decidir la utilización de un producto determinando, pudiéndose determinar si este cumple con las necesidades existentes y si permite compararlo con alternativas comerciales. Tiene 27 parámetros (12 características genéricas del producto y 15 características basadas en las necesidades del usuario) La puntuación va de 1 a 5 (Widdows y Duijnhouwer, 2003).

OSMM - Navica**:** posee un método más compacto que el de Capgemini. Consta de 4 fases: la primera fase es para seleccionar el software que se va a evaluar. La segunda fase es para ponderar las categorías de la siguiente forma: *software* (4 ptos), *soporte* (2 ptos), *documentación* (1 pto), *capacitación* (1 pto), *integración* (1 pto) y *servicios de profesionales* (1 pto). La tercera fase se basa en la aplicación de cada una de las plantillas correspondientes a las categorías nombradas en la fase anterior, en las cuales se sugiere qué aspectos dentro de la categoría deben ser evaluados ponderando cada uno de estos. La cuarta fase consiste en multiplicar la puntuación de cada categoría por su ponderación para producir una puntuación final de 0 a 100 (Rodríguez y Soria, 2010).

BRR (*Business Readiness Rating*) fue anunciado en el 2005, este método permite evaluar el software libre dado cuatro pasos:

1. Creación de una lista con el software a evaluar.
2. Clasificación y ponderación de los criterios de selección.
3. Recopilación de datos para cada criterio.
4. Cálculo y publicación de los resultados.

Tiene 12 criterios que pueden ser utilizados para evaluar las aplicaciones: *funcionalidad, usabilidad, calidad, seguridad, rendimiento, escalabilidad, arquitectura, soporte, documentación, adopción, comunidad y profesionalismo.* No tuvo el nivel de adopción (aceptación) que se esperaba ya que no creó una comunidad próspera y en julio de 2010 que se le dio de baja a este proyecto.

El modelo de madurez utilizado en este trabajo para poder elegir un software en dependencia a las características y funcionalidades que posea, es el modelo para **Calificar y Seleccionar Software de Código Abierto** (en inglés *Qualification and Selection of Open Source Software o* ***QSOS***). El mismo presenta cuatro pasos *(Definición, Evaluación, Calificación* y *Selección*), que por su función son todos utilizados en el estudio realizado.

El primer paso es la **definición** de elementos para realizar la clasificación y descripción funcional del software, la clasificación de licencias y la clasificación de la comunidad.

Para la **clasificación y descripción funcional del software**, los elementos tomados en cuenta de los que define este modelo en este paso fueron:

- Sistema operativo: para qué tipo de sistema operativo esta diseñada la herramienta.
- Funcionalidad: la medida en la que el software satisface las necesidades del usuario.
- Usabilidad: la medida en la que el software es intuitivo**/** fácil de instalar**/** fácil de configurar**/** fácil de mantener.
- Calidad: la medida en la que el software está bien diseñado, implementado y probado.
- Seguridad: La capacidad del sistema que indica que está libre de todo peligro, daño o riesgo y que es en cierta medida infalible.
- Rendimiento: el comportamiento del software ante ciertas circunstancias. Efectividad para administrar recursos.
- Documentación: existencia de documentación de buena calidad.

Para la **clasificación de licencias** el elemento definido fue:

- Licencia de software: contrato entre el licenciante y el licenciatario para utilizar el software según las condiciones establecidas por el primero.

Para la **clasificación de la comunidad** los elementos definidos fueron:

- Adopción: el software ha sido adoptado por la comunidad, el mercado, y la industria.
- Comunidad: la comunidad del proyecto está activa.

El segundo paso es la **evaluación** donde se colecciona la información general del software, servicios ofrecidos, así como aspectos funcionales y técnicos. Por las necesidades existente planteadas en el problema, los aspectos que se tuvieron en cuenta se mencionan a continuación:

- Interfaz física: gestión de interfaces de red física utilizando método estático y dinámico (dhcp).
- Interfaz virtual: gestión de interfaces de red virtual a partir de interfaces físicas utilizando método estático y dinámico (dhcp).
- Balanceo de carga: realización de balanceo de carga (bonding) entre las interfaces de red física.
- Comprobar conexión: verificar si existe conexión después de configurar una interfaz de red.
- Configuración del DNS: si permite configurar las interfaces a través del servidor DNS y de dominio.

El tercer paso es la **calificación***,* su objetivo es ponderar los criterios definiendo el rango o los valores asignados Para realizar la ponderación de los parámetros estableció: (de 1 a 5 puntos) si son aspectos para clasificar el software o son descripción funcional; (0 o 1 punto) para evaluar el software en dependencia de si ofrece o no servicios y presentan aspectos funcionales o técnicos. De esta forma se puede descartar el software que menos satisface las necesidades que existen hoy en los OACE y HMAST, además de conocer cuál de ellos aporta más en la gestiona de la red como contexto específico.

Por último se realiza la **selección** para identificar el software que cumple con los requisitos del usuario de dos modos posibles: una selección estricta y una selección más amplia. Este paso es el resultado de lo que se hizo en los pasos mencionados anteriormente, ayudando así a un mejor entendimiento de lo que se va hacer y un acercamiento a la solución del problema (Rodríguez y Soria, 2010).

1.2.1 Herramientas estudiadas

A continuación se realiza una descripción de algunas de las herramientas de código abierto que administran el servicio de red. De estos sistemas se detallan características generales, cómo realizan las

configuraciones, sus limitaciones, facilidad de uso, entre otros aspectos. Webmin y Zentyal fueron aplicaciones a comparar definidas por el proyecto HMAST, partiendo del hecho que ya existe un estudio previo de las mismas. Es elegida además por parte del autor NetworkManager, pues está contenida por defecto en la mayoría de los sistemas operativos GNU/Linux y sólo se encarga de gestionar el servicio de red en los sistemas, a diferencia de las anteriores que además administran otros servicios telemáticos. Por tanto en esta investigación se pretende identificar cuál de estas aplicaciones es la más madura según el modelo QSOS y pueda aportar funcionalidades en aras de solucionar el problema existente.

Webmin 1.6.30

Webmin es una herramienta de configuración de sistemas accesible vía web para OpenSolaris, GNU/Linux y otros sistemas Unix. Permite configurar aspectos internos de muchos sistemas operativos, como usuarios, cuotas de espacio, servicios, archivos de configuración, apagado del equipo, así como modificar y controlar muchas aplicaciones libres, tales como el servidor web Apache, PHP, MySQL, DNS, Samba, DHCP, etc.

Funciona a través de una interfaz web, que permite al usuario (autorizado) realizar desde una interfaz gráfica, todo tipo de tareas de administración y configuración de sistemas bajo Unix, Linux o FreeBSD de manera gráfica. Su código está escrito en Perl (versión cinco), ejecutándose como su propio proceso y servidor web. Por defecto se comunica a través del puerto TCP 10000, y puede ser configurado para usar SSL si OpenSSL está instalado con módulos de Perl adicionales requeridos (Webmin, 2013). Todas sus versiones recientes están bajo una licencia BSD, lo que significa que pueden ser libremente distribuidos y modificados para uso comercial y no comercial.

Está construido a partir de módulos entre los cuales se encuentran: Apache WebServer, BIND DNS Server, CVS Server, DHCP Server, FTP, Proxy, Jabber, entre otros, para un total de 27. No limita al usuario poder instalar algún otro módulo que no se encuentre en su configuración inicial. Cuenta con una interfaz para los archivos de configuración y el servidor Webmin, haciendo fácil la adición de nuevas funcionalidades sin mucho esfuerzo. Debido a su diseño modular, es posible para cualquier interesado escribir extensiones. Permite la configuración de la mayoría de los servicios telemáticos posibilitando cambiar el acceso a los archivos de configuración (Webmin, 2013).

Como aspecto destacado de Webmin en su versión1.6.30, se puede mencionar que se le añadió al módulo BIND soporte para el tipo de registro SPF real. Nuevas particiones han sido creadas en el módulo de discos locales para FreeBSD; además de mejorarse el soporte para la configuración de la interfaz y los comandos de inicio, como se utiliza en el módulo de servidor de DHCP en FreeBSD 9. Gestiona la mayoría de los parámetros de configuración y es capaz de integrar los servicios DNS y DHPC.

Administración de red con Webmin

El módulo de configuración de la red se puede encontrar en la categoría ***Redes*** en Webmin. La página principal muestra iconos para cada una de las cuatro categorías de configuración: **Interfaces de red***,* **enrutamiento** *y* **Gateways***,* **Clientes DNS** *y* **Direcciones de host**. Todos los formularios que permiten su edición y las opciones en el módulo están bajo una de estas cuatro categorías. Este módulo fue diseñado principalmente para la configuración de red en sistemas con conexiones permanentes, tales como **Ethernet** (*eth)* o tarjetas **Token Ring** (*tr*).

Webmin permite crear interfaces de red, para utilizarlas momentáneamente o para el futuro, estas pueden visualizarse y ser editadas. Las interfaces existentes se encuentran enumeradas en el sistema en dos categorías, la primera es **'Interfaces activo ahora'***:* son las que están habilitadas actualmente y tiene una dirección IP asignada (*loopback*, *Ethernet* y *PPP*), aunque no todas se puede editar usando Webmin. La segunda categoría es **'Interfaces activado al arrancar'***:* son las que han sido configuradas para ser activadas en el arranque. Las dos listas no serán necesariamente las mismas, ya que algunos tipos de interfaces (como PPP) no se activan durante el arranque y no aparecerán en la segunda lista.

Con Webmin la configuración de las interfaces de red real se pueden realizar de manera **estática***,* en la cual se debe introducir la **dirección IP***,* **máscara de red** y **puerta de enlace** y de forma **dinámica** se obtienen los parámetros mencionados de forma automática a través de algún servidor DHCP en la subred. Además, posibilita gestionar interfaces de red virtual a partir de alguna interfaz real existente, a esto se agrega que se puede configurar el enrutamiento, cambiar la configuración del cliente de nombre o DNS, así como la edición de las direcciones de *host*. El uso de estas funciones evita tener que recordar las ubicaciones de los archivos de configuración (como en /etc), editarlos a mano y detener o iniciar servicios.

Webmin tiene en cuenta cada detalle de configuración del servicio de red, por lo que para usar el mismo es necesario que se tengan conocimientos avanzados en la administración de servidores; por tanto

presenta algunos inconvenientes, que tienen más o menos impacto en dependencia de quién y dónde se utilice. Por ejemplo un aspecto importante a tener en cuenta, es que debe ser instalado en cada servidor de la red, lo cual es innecesario cuando solo se quiere tener un servidor dedicado, como un Proxy o un DNS. También requiere que las conexiones a los mismos se realicen mediante el usuario *root* del sistema y no otro (Soria y Castillo, 2012).

Como se mencionaba anteriormente, el trabajo con este programa se realiza vía web, conectándose al servidor por el puerto 10000 mediante el protocolo SSL. En Cuba, los mecanismos que usan SSL mediante la generación de certificados en un servidor no cuentan con los servicios de una entidad certificadora de prestigio internacional, que genere el certificado necesario para establecer conexiones seguras; por tanto tienen que gestionarlos por sus propios medios.

El autor considera que aunque Webmin en su versión 1.6.30 presenta nuevas opciones y modificaciones a varios de sus módulos, todavía el trabajo con el mismo resulta complejo por la cantidad de configuraciones que tiene en cada interfaz de usuario. Además, aun no se encuentra escrita la ayuda sobre el trabajo con todos sus módulos. Aunque a los efectos de esta investigación, Webmin facilita un mejor entendimiento sobre cómo se gestiona el servicio de red y se realizan las configuraciones.

Zentyal 3.2

Anteriormente conocido como eBox Platform (o una plataforma de red unificada) **Zentyal** es un producto español equivalente a *Windows Small Business Server*. Licenciado bajo GPL, actualmente es una solución consolidada que integra más de 30 herramientas de código abierto para la administración de sistemas y redes en una sola tecnología. Tiene más de 1.000 descargas diarias y dispone de una comunidad activa de más de 5.000 miembros (Zentyal, 2011).

Fue desarrollado con el objetivo de acercar GNU/Linux a las PYMES[8] y otros entornos como centros educativos, administraciones públicas, hospitales o incluso en instituciones de alto prestigio como la propia NASA. Este servidor basado en Ubuntu Server 10.04 proporciona una forma sencilla de configurar

[8] **PYME**: Acrónimo de Pequeñas y Medianas Empresas.

y gestionar infraestructuras de red como puerta de enlace a Internet (***Gateway***) a través de una interfaz web intuitiva[9], que lo distingue de otras distribuciones.

Permite a los informáticos de una institución, administrar todos los servicios telemáticos necesarios en su infraestructura de servidores de forma práctica y a través de una única plataforma. Al igual que ocurre con Webmin, este sistema administra los servicios a través de módulos, permitiendo instalar algún otro si es necesario. Además, permanentemente esta aplicación muestra el estado que presentan sus módulos (**Ejecutándose**, **No creada**, **Deshabilitado** y **No suscrito**) y si algún módulo depende de otro servicio.

El Servidor Zentyal 3.2 ofrece a las PYMES un servidor compatible con Microsoft de forma nativa que se puede configurar en menos de 30 minutos (Zentyal, 2013). La mejora más importante que presenta esta nueva versión es una mayor integración con la tecnología *Samba.* Esto posibilita introducir el servidor Zentyal de forma transparente en un entorno Windows, fácilmente migrar los servicios de red, los usuarios y apagar los servidores Windows innecesarios o no soportados sin causar ninguna molestia a los usuarios. El Servidor Zentyal 3.2 también ofrece soporte a Directivas de Grupo (GPOs) y Unidades Organizativas (OUs) en los módulos más importantes.

Este producto informático tiene un conjunto de funcionalidades entre las que destacan:

- Gestión de red: firewall, servidor DHCP, servidor NTP, servidor DNS, soporte para red privada virtual (en inglés *Virtual Private Network* o VPN), proxy HTTP, entre otros.
- Servidor de correo: POP3 e IMAP con SSL/TLS, filtro antispam y antivirus, webmail, etc.
- Comunicaciones: FTP, servidor VoIP, Voicemail y servidor Jabber de mensajería instantánea.
- Compartición de recursos y trabajo en grupo: servidor de archivos, servidor de impresión y ***groupware*** (agenda, contactos, etc.).
- Gestión centralizada de usuarios mediante LDAP, sincronización con Directorio de Windows, etc.
- Reportes y monitoreo del sistema, además del envío de notificaciones vía correo, RSS o Jabber.
- Respaldos de configuraciones y datos de manera remota.
- Autoridad de certificación.

La interfaz de administración de Zentyal se encuentra dividida en tres partes fundamentales:

[9] **Interfaz Intuitiva:** Es aquella interfaz fácil de trabajar, donde el usuario final sabe que tiene que hacer en cada opción que se muestre, pues generalmente son comunes a todos los usuarios.

Ilustración 1: Menú Izquierdo

◆ **Menú lateral izquierdo,** visto en la *Ilustración 1* contiene los enlaces a todos los **servicios** que se pueden configurar mediante Zentyal, separados por categorías. Cuando se ha seleccionado algún servicio en este menú puede aparecer un submenú para configurar cuestiones particulares del mismo.

◆ **Menú superior,** contiene las acciones de guardar los cambios realizados en el contenido y hacerlos efectivos, así como el cierre de sesión.

◆ **Contenido principal,** ocupa la parte central y comprende uno o varios formularios o tablas con información acerca de la configuración del servicio seleccionado a través del menú lateral izquierdo y sus submenús. En en la parte superior aparecerá una barra de pestañas, en las que cada pestaña representará una subsección diferente dentro de la sección que se ha accedido (Zentyal S.L, 2011).

Administración de red con Zentyal

Algunas de las funciones que Zentyal realiza para la gestión de red:

- **Zentyal gateway**: permite que la red sea más fiable, optimiza el ancho de banda así como ayuda a controlar lo que entra en la red.
- **Control de tráfico:** asegura que el tráfico más crítico se distribuya bien, independientemente de la carga de la red.
- **Balanceo de carga y disponibilidad:** si se tiene más de una conexión a Internet, Zentyal puede distribuir sus clientes de forma transparente y asegurar que estén conectados aunque una de las conexiones esté inactiva.
- **Filtrado de contenido:** restringe o permite el acceso a sitios específicos, imposibilitando la navegación inapropiada y ayudando a cumplir las políticas de uso de Internet.
- ***Zentyal infrastructure*:** gestiona y optimiza el tráfico interno de la red, incluyendo la configuración del servidor de dominio y la gestión de máquinas y certificados digitales.
- **Objetos de red:** se puede elegir hasta qué nivel se quiere gestionar la red con Zentyal. Se puede cambiar la configuración de toda la red, de un departamento o de un solo ordenador.

- **Servidor DNS:** permite establecer los servidores DNS que se encargarán de resolver los nombres, pudiéndose asignar una dirección y un nombre fijo a cada equipo de la red para facilitar la navegación. Por ejemplo, es más fácil recordar uci.cu que 10.0.0.2.

Zentyal a través del parámetro **Red** en la opción **Interfaces** permite configurar la interfaz de cada una de las tarjetas de red detectadas por el sistema. Se puede establecer la dirección de red estática (configurada manualmente) o dinámica (configurada mediante DHCP).
Cuando se configura como cliente de DHCP, no sólo se configurará la dirección IP sino también los servidores DNS y la puerta de enlace. Si se configura la interfaz como estática se debe especificar la dirección IP, la máscara de red y puerta de enlace. Además, se puede asociar una o más **interfaces virtuales** a dicha interfaz real para disponer de direcciones IP adicionales posibilitando que pueda ser utilizado para ofrecer otros servicios.

Con el modo puente o *bridged* Zentyal puede asociar dos interfaces de red físicas de un servidor conectado a dos redes diferentes; por ejemplo, una tarjeta conectada al *router* y otra a la red local. Brinda la opción de resolver nombres de dominio o *DNS*, además propicia integrar las capacidades de este servicio con *DHPC*. Permite realizar el diagnóstico de la red, para comprobar la conexión de una PC con otra. También dispone de la herramienta **traceroute** que se encarga de mostrar la ruta que toman los paquetes hasta llegar a la máquina remota determinada.

A diferencia de Webmin que tenía muchas configuraciones, Zentyal tiene muy pocas con respecto a las funcionalidades que realiza su módulo de configuración de red. Ejemplo de ello es cuando se adiciona un host virtual, no permite cambiar la dirección donde va a estar lo que se quiere compartir. Otro ejemplo es que Zentyal no reconoce en ocasiones los cambios que se realizan directamente en los ficheros, por lo que todo lo que se necesite hacer tiene que ser por medio de su interfaz. En caso de escribir directamente en los ficheros, Zentyal lo reemplaza cuando actualiza sus configuraciones; pudiendo esto causar problemas ya que muchos administradores tienen ficheros de configuración guardados y cuando quieren restablecerlos no pueden. Esto provoca que estos ficheros sean completamente dependientes de este sistema para realizar cualquier cambio.

Debe señalarse entre las desventajas de Zentyal como servidor para gestionar los servicios telemáticos, el hecho de que algunos hardware incompatible con algunos de sus módulos. Además, como todos los

servicios pueden estar en una misma computadora, si existiera por ejemplo, desbordamiento de buffer y hay que reiniciar el servidor, se reinician todos los servicios también. No se permite importar las configuraciones de otros servidores; si se decide migrar a otro servidor se tiene que volver a configurar; si el servidor sufre algún problema hay que reinstalarlo y por ese tiempo están inaccesibles todos los servicios de la red.

Este sistema no permite gestionar correctamente un servidor DHCP porque bloquea algunas opciones que en muchas ocasiones son necesarias. Por ejemplo, no se puede adicionar una subred que no esté enmarcada en la que se instaló en el servidor. Solo permite las actualizaciones automáticas del DHCP con el servidor DNS que él instala y no con otro.

El autor considera, que aunque existen motivos por los cuales se puede recomendar ejecutar la migración con Zentyal en su versión 3.2 en las instituciones cubanas en donde se gestionan los servicios a través de software privativo; existen otros que no favorecen el trabajo de sus administradores. En la gestión de red, se pueden obtener de este sistema varias de sus cualidades, las cuales podrían esclarecer cuestiones sobre este servicio y además brindar conocimiento de las funcionalidades que en esta herramienta se ejecutan en aras de solucionar el problema a través de HMAST.

Network Manager

NetworkManager es un programa que proporciona a los sistemas la detección y configuración automatizada para conectarse a la red desde la interfaz gráfica de usuario. Fue iniciado por Red Hat en 2004 y ahora respaldado por el proyecto GNOME, posee una licencia GPL. Es una aplicación cuyo objetivo es que el usuario nunca tenga que lidiar con la línea de comandos o la edición de ficheros de configuración para manejar sus redes (ya sea cableada o inalámbrica) en GNU/Linux y otros sistemas operativos basados en Unix. Pretende que la detección de dichas redes simplemente funcione tanto como se pueda y que interrumpa lo menos posible el flujo de trabajo del usuario, permitiendo enfrentar con facilidad las necesidades en el uso de redes de computadoras (Barrios, 2014). Funciona para entornos de escritorios como GNOME, KDE, XFCE, entre otros.

Este programa da preferencia a las conexiones por cable antes que las inalámbricas, tiene soporte para conexiones por módem y para ciertos tipos de VPN. Según sea necesario se pide al usuario claves de

Privacidad Equivalente a Cableado (en inglés *Wired Equivalent Privacy o* **WEP**[10]) o Acceso Wi-Fi Protegido (en inglés *Wi-Fi Protected Access o* **WPA**[11]).

La última versión del conocido gestor de conectividad ha llegado con novedades importantes. NetworkManager 0.9.8 y su applet y plugins VPN asociados permiten ahora crear puntos de acceso WiFi si el hardware y los controladores soportan dicha capacidad.

La aplicación de NetworkManager se compone de cuatro partes distintas:

1. *NetworkManager*: un demonio que se encarga de la conexión y la política de la red.
2. *DHCPCD*: un demonio cliente DHCP utilizado por NetworkManager.
3. *NetworkManagerInfo*: programa que recoge y comunica las preferencias del usuario para NetworkManager y da las notificaciones para el usuario.
4. *NetworkManagerNotification*: ícono de notificación situado en el panel y utiliza NetworkManagerInfo para mostrar las posibles conexiones con cable e inalámbricas (RedHat, 2010).

Administración de red con NetworkManager

NetworkManager maneja todos aquellos dispositivos que no estén listados en el fichero /etc/network/interfaces, o aquellos que estén listados en dicho fichero con la opción **auto** o **dhcp**, de esta manera se puede establecer una configuración que sea estática para un dispositivo determinado y este sistema no tratará de sobreescribir dicha configuración. Si se desea administrar todas las interfaces posibles en el sistema, se puede hacer esto escribiendo en el fichero **/etc/network/interfaces** lo siguiente:

auto lo

iface lo inet loopback

Una vez que se ha modificado el fichero mencionao, se reinicia NetworkManager con el comando **service network-manager restart**; después de esto el mismo programa se encargará de detectar las redes inalámbricas disponibles, estas pueden ser vistas en una lista de redes disponibles a través de un icono.

Cuando algún dispositivo con NetworkManager esté en áreas en las cuales ha estado antes, este sistema se conectará automáticamente de forma inalámbica a alguna red donde se haya conectado, pero previamente debe haberse conectado de forma manual al menos una vez. Si se selecciona una red de la lista, automáticamente NetworkManager intentará conectarse; en caso de que esta red requiera una llave

[10] ***WEP:*** Cifra (codifica) los datos de la red mediante una "clave" de forma que sólo el destinatario deseado pueda acceder a ellos.

[11] ***WPA:*** *E*mplea el cifrado de clave dinámico (la clave está cambiando constantemente) y hacen que las incursiones en la red inalámbrica sean más difíciles que con WEP. está considerado como uno de los más altos niveles de seguridad inalámbrica para redes.

de cifrado, se mostrará un cuadro de diálogo en el cual preguntará acerca de ella. Una vez ingresada la llave correcta, la conexión se establecerá.
Para cambiar entre redes, simplemente se debe escoger otra red desde el menú que ofrece el *applet*. Cuando se necesite trabajar con la red cableada, se selecciona una conexión de red comprobando primeramente si es una tarjeta ***Ethernet***; si esto se cumple, cambiará a la red más rápida y confiable. Si no existe tal conexión, el programa intenta obtenerla automáticamente por *DHCF* (Mazzarri, 2013).

Aunque en esta herramienta es muy simple realizar la configuración de la red, tiene muchas diferencias con respecto a las herramientas homólogas Webmin y Zentyal; ejemplo de esto es la posesión de módulos para la administración, las diferentes interfaces de usuario para realizar las configuraciones, la forma de uso y principalmente la forma de gestionar la red.

En ocasiones al ejecutar alguna función, esta aplicación se puede demorar o incluso detener su funcionamiento. Además, presenta un problema general en relación con la ejecución del **script dispatcher,** ya que demora varios segundos antes de ser ejecutado y esto retrasa las ejecuciones internas. También, debido a un error aún no solucionado, al cambiar las conexiones por defecto para la IP estática, el comando **nm-applet** no guarda correctamente el cambio de configuración presentando problemas para obtener una dirección IP mediante DHCP, dando como resultado que para solucionar este problema debe editarse la conexión manualmente en el archivo de configuración **/etc/dhclient**. Se detectó otro problema relacionado con el nombre del equipo *(hostname*), ya que el plugin genérico '**keyfil**' no envía por defecto a un *router* para conectarse, el nombre del equipo en la configuración (ArchLinux, 2013).

El autor de este trabajo considera que aunque NetworkManager presenta facilidades de uso para realizar las configuraciones del las interfaces de red; no cumple con varias de las necesidades actuales de los OACE, pues no realiza el balanceo de carga entre interfaces de red y no posibilita la creación de interfaces virtuales. Por tanto, solo se puede tomar de este, criterios que ayudan a solucionar algunas de las deficiencias existentes en las entidades cubanas y funcionalidades que actualmente HMAST no posee.

Con NetworkManager anticipar y satisfacer las necesidades de gestión de la red se convierte en una tarea fácil. Los usuarios pueden configurar rápidamente las interfaces permitiendo ahorrar tiempo en este proceso. Al no poseer módulos, todas las funcionalidades van encaminadas exclusivamente a la gestión de redes ya sean inalámbricas o cableadas.

1.2.2 Criterios para comparar las herramientas definidas

A continuación se muestra una tabla con el resultado de cada uno de los criterios tomados en cuenta para realizar la comparación de las herramientas de código abierto estudiadas y descritas anteriormente. Algunos aspectos los plantea el **modelo de madurez** definido y otros fueron los considerados importantes según la necesidad existente, los mismos fueron:

	Webmin	Zentyal	Network Manager
Sistema Operativo	GNU/Linux	Ubuntu	GNU/Linux
Funcionalidad	2	3	4
Usabilidad	3	4	4
Calidad	4	4	3
Seguridad	5	5	5
Rendimiento	3	4	4
Documentación	5	5	3
Licencia	BSD	GPL	GPL
Adopción	5	4	2
Comunidad	5	5	0
Interfaz Física	1	1	1
Interfaz virtual	1	1	0
Balanceo de Carga	0	1	0
Configuración del DNS	1	1	1
Comprobar conexión	1	1	1

Tabla 1: Comparación entre herramientas homólogas

1.2.3 Resultado del estudio

Con el estudio de estas herramientas se obtienen los siguientes resultados:

En Webmin se evidencia que aunque presenta varias facilidades y gestiona los servicios de forma correcta, contiene una interfaz poco intuitiva ya que contiene gran cantidad de campos para insertar datos en una misma interfaz de usuario, sin uso de pestañas u otros componentes que pueden ayudar a organizar los campos de entrada de datos. Cuenta con configuraciones innecesarias, pudiéndose

constatar que existe descentralización de algunos módulos, incluyendo el de red, lo cual se evidencia a partir de la administración independiente de sus componentes.

Después de realizar un estudio del módulo de red de la herramienta Zentyal se ha detectado que la misma sobreescribe los cambios realizados en ficheros de configuración por agentes externos a la aplicación, restringiendo el sistema a las funcionalidades únicas disponibles en esta, cuestión que afecta la flexibilidad. Además, presenta configuraciones que se realizan para el servicio de red que se encuentran en distintas interfaces. También se debe destacar la relación existente entre algunos de los módulos, pues al configurar uno si este requiere de otro, también es configurado. Desde ese punto de vista es una ventaja porque configurando un servicio se tiene parte de la configuración de otro, pero en caso de que no se requiera de esta dependencia habría que realizar el mismo trabajo varias veces.

En Network Manager se encuentra presente una serie de situaciones que no facilitan su uso en los OACE. Aunque con esta aplicación solo se gestiona la red cableada e inalámbrica, hay otros servicios que dependen de sus configuraciones como DHCP y DNS, con los cuales este sistema presenta dificultades cuando necesita escribir en sus ficheros. Además, presenta problemas en el tiempo de su ejecución interna y dificultades en los procedimientos de configuración. La realización del trabajo manual sobre el ficheros de configuración, da paso a demoras e interrupciones en funcionamiento. Esto propicia que este sistema sólo proporcione para la solución un grupo de aspectos, como por ejemplo facilidad en su uso para realizar configuraciones de la red.

Herramienta más madura

Finalmente y dando uso del cuarto paso (**Selección**) del modelo de madurez utilizado (*QSOS);* se elaboró una tabla comparativa para identificar el software que cumple con la mayor cantidad de criterios planteados y definidos.

A partir de esto se determinó que la herramienta más completa (madura) es Zentyal, pero no permite su integración a HMAST por la arquitectura de esta herramienta. Aunque posee aspectos por los cuales no se puede utilizar como vía para solucionar algunos de los problemas existentes en los OACE; contiene aspectos básicos referidos a la gestión de muchos de los servicios telemáticos. Además, realiza la mayoría de las funcionalidades del servicio de red que no presenta actualmente HMAST, con las cuales se

podrían satisfacer varias de las necesidades existentes en las empresas cubanas. Posibilita realizar extensiones entre algunos de los servicios que más se utilizan actualmente. Esto se podría tomar como base para la concepción de un módulo para la gestión de la red. Un claro ejemplo sería: la usabilidad pues se puede crear una interfaz intuitiva que incluya aquellas funcionalidades necesarias y de uso más frecuente que contiene esta herramienta y que son necesarias en los OACE.

Conclusiones parciales

Durante el desarrollo de este capítulo se realizó un estudio referente al servicio de red obteniéndose un mejor entendimiento del mismo. Quedó definido qué son las redes de computadoras y los tipos de estas que existen. Además, qué son las interfaces de red, cuáles son las más comunes hoy en día, el trabajo que estas realizan y como se configuran para que un equipo de cómputo pueda obtener conexión en un red determinada. Fueron estudiadas herramientas de código abierto que permiten gestionar el servicio de red, siendo analizadas sus características más importantes. Teniendo en cuenta aspectos específicos definidos a través del modelo de madurez QSOS se pudo realizar una correcta investigación de estas herramientas y posteriormente la elección de la más completa entre ellas. Se llegó a la conclusión que, entre Webmin, NetworkManager y Zentyal esta última es la más madura y la que más podría aportar a la resolución del problema existente.

Capítulo 2: Análisis y diseño de la solución propuesta

En este capítulo se describe la propuesta del software que se desea implementar para solucionar el problema existente en los OACE. Se realiza el análisis y modela el diseño para desarrollar un módulo que administre el servicio de red. Primeramente, son identificados y expuestos los requisitos funcionales y no funcionales que debe cumplir el sistema, todos agrupados y ordenados por prioridad en el artefacto Lista de Reserva del Producto. Cada una de las funcionalidades se describen y agrupan en Historias de Usuario. Se expone la estructura que tiene HMAST y la arquitectura que debe cumplir el sistema a desarrollar para poder integrarlo a esta herramienta, además se plantean los patrones de diseño utilizados.

La **misión** de este trabajo es proveer a los clientes de un sistema que facilite la gestión del servicio de red en sistemas de software libre, de forma estable y fácil en el menor tiempo posible, con el fin de reemplazar el trabajo no automatizado realizado por los administradores de este servicio en las entidades cubanas.

2.1 Propuesta del sistema a desarrollar

Después de realizar el estudio a las herramientas definidas que administran el servicio de red, de éstas se obtiene un conocimiento sobre cómo gestionan dicho servicio, rendimiento, funcionalidad y facilidades de uso para los usuarios. Dado que HMAST no contiene opciones o funciones cumplan con lo planteado anteriormente para gestionar la red y que estas son necesaria para facilitar el trabajo que realizan los administradores de los OACE. Se concluye con la necesidad de desarrollar una aplicación de código abierto que desde HMAST permita administrar el servicio de red; siendo esta la solución perfecta para solucionar el problema en cuestión. Permitiendo así obtener un sistema que gestione la red, con una interfaz flexible, simple y escalable que vincule los servicios necesarios para obtener una conexión robusta.

Concepción inicial

Se describen los principales procesos de negocio para conformar el módulo de red, proponiéndose para su correcta implementación e integración en la herramienta HMAST, utilizar una arquitectura basada en el estilo *N-capas orientada al dominio.* Simplificando así la comprensión y organización del sistema, reduciendo las dependencias de forma tal que las capas más bajas no sean conscientes de ningún detalle o interfaz de las capas superiores, además de fomentar la re-utilización.

2.1.1 Descripción del negocio

Para poder administrar el servicio de red en servidores desde HMAST posibilitando establecer una conexión estable y agilizar este proceso; se deben implementar las funcionalidades necesarias para de forma automatizada, configurar interfaces de red físicas, interfaces de red virtual y realizar el balanceo de carga. Se desarrollan componentes y crean clases que posibiliten la gestión y comunicación de/entre diferentes servicios telemáticos. Además, las interfaces de usuario deben proveer las opciones necesarias para acceder desde la capa de presentación hasta las configuraciones del servicio.
Para la gestión de algún servicio, HMAST permite seleccionar y agregar el módulo deseado por el usuario, en caso de que sea seleccionado el módulo de red, la aplicación a través de opciones permitirá realizar las configuraciones necesarias a las interfaces de red.

Como por cada tarjeta de red existe una interfaz de red física, a estas sólo se le pueden editar sus parámetros. Las configuraciones se realizarán si el método utilizado es estático, por tanto, los datos deberán ser introducidos por el usuario; si el método es dinámico (*dhcp*) las configuraciones se obtendrán automáticamente a través de un servidor DHCP y se guardarán en un fichero.
Se podrán crear una o varias interfaces virtuales a partir de una interfaz de red física, pudiéndose modificar o eliminar en caso necesario. Los parámetros que se configurarán, son los mismos que los de la interfaz física, pero en este caso se agregará un identificador por cada interfaz virtual creada.
Para realizar el *bonding* (balanceo de carga) entre interfaces, se debe especificar qué interfaces serán esclavas, cuál de estas sera la primaria y el modo que se va establecer. Para obtener la conexión se configuran lo mismo que en el caso de las interfaces de red física y además de esto algunos parámetros propios de esta función, que por defecto estarán en cero si no se configuran.

Con el objetivo de proporcionar la primera visión del sistema a desarrollar, se toma como punto de partida el **modelo de dominio** ver.(*Figura 4: Modelo de dominio)* del Anexo 1.

2.2 Requisitos

Las condiciones que el sistema debe cumplir o capacidad que debe tener con el objetivo de establecer un entendimiento común entre el usuario y el proyecto de software son los requisitos. El propósito de su gestión es establecer un entendimiento común entre el usuario y el desarrollador de software; son clasificados en funcionales y no funcionales.

Los **requisitos funcionales** son la determinación exacta de qué debe ser capaz de hacer el sistema, éstas se corresponden con opciones que ejecutará el software, operaciones realizadas de forma oculta o condiciones extremas a determinar por el sistema (Pressman, 2001).

Los **requisitos no funcionales** (o de calidad) son aquellas propiedades que debe tener la solución, no son funcionalidad, pero influyen directamente en los puntos clave de la arquitectura que sí son funcionalidad del sistema. Se puede decir que los requisitos no funcionales son la especificación de las propiedades del sistema y los puntos claves de la implementación.

Son clasificados en requisitos de Funcionamiento (*Functional*); Facilidad de uso (*Usability*); Fiabilidad (*Reliability*); Rendimiento (*Performance*); Soporte (*Supportability*). Además de otros como: requisitos de implementación; de interfaz; de operaciones; de empaquetamiento y legales (Llorente *et al*, 2010).

2.2.1 Lista de reserva del producto

Durante la definición de los requisitos, estos fueron clasificados en el artefacto Lista de reserva del producto (LRP) (*Anexo 2*) y se organizaron a partir de la prioridad que tenían para el negocio (*Muy Alta, Alta, Media y Baja).* En total fueron definidos un total de 16 requisitos funcionales de ellos 14 con prioridad alta, los dos restantes repartidos con prioridad media y baja. Se incluyen además, 7 requisitos no funcionales, de ellos 6 los establece el sistema raíz (HMAST).

Requisitos funcionales

Partiendo de la solución propuesta y de la visión general del sistema a implementar planteada en el modelo de dominio, son definidos mediante las necesidades del cliente, cuáles serán las características y cualidades a cumplir por la propuesta de solución.

Prioridad	Requisitos
Alta	1. *Mostrar interfaces disponibles.* 2. *Modificar información de interfaz de red física.* 3. *Mostrar interfaz de red física* 4. *Comprobar conexión de la red.* 5. *Crear interfaz de red virtual.* 6. *Mostrar interfaz de red virtual.* 7. *Modificar interfaz de red virtual.* 8. *Eliminar interfaz de red virtual.* 9. *Crear balanceo de carga entre interfaces físicas de red.* 10. *Activar balanceo de carga.* 11. *Desactivar balanceo de carga.* 12. *Mostrar balanceo de carga.* 13. *Editar balanceo de carga.* 14. *Eliminar balanceo de carga.*
Media	15. *Reiniciar servicio de red.*
Baja	16. *Mostrar características de tarjetas de red.*

Tabla 2: Requisitos funcionales

Requisitos no funcionales

Los requisitos no funcionales definidos por el proyecto de desarrollo son clasificados como:

- Requisitos de **interfaz**, de **facilidad de uso** y **de funcionamiento**:
  -Crear interfaz con apariencia sencilla e intuitiva desde la cual se pueda tener acceso a todas las funcionalidades del módulo.
  -La aplicación se va a ejecutar en sistemas operativos libres.
- Requisitos de i**mplementación** (limitación de recursos, lenguajes, herramientas y hardware):
  -Usar como lenguaje de programación Java con el framework de desarrollo Spring versión 3.1.1.
  -Utilizar como entorno de desarrollo integrado NetBeans en su versión 7.3.1.
  -Utilizar como herramienta de modelado a Visual Paradigm en su versión 8.0.
  -Utilizar como servidor web Apache Tomcat en su versión 7.0.4.
  -Disponer en el sistema operativo GNU/Linux del paquete Augeas 1.1.

2.2.2 Historias de usuario

Las Historias de Usuario (HU) son artefactos que constituyen la técnica utilizada en SXP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el Proceso Unificado. Las HU guían la construcción de las pruebas de aceptación y son utilizadas para estimar el tiempo de desarrollo. En este sentido, sólo proveen detalles suficientes para hacer una estimación razonable del tiempo que llevará implementarlas. En el momento de implementar una HU, se debe detallar a través de la comunicación con el cliente, estas son las bases para las pruebas funcionales (Peñalver, 2008).

Se muestran a continuación la descripción de las 5 HU descritas a partir de los requisitos funcionales; la prioridad de estas parten de la prioridad que tienen los requisitos que las componen. Cada HU puede verse en el (Anexo 3).

No	Nombre	Prioridad	Descripción
1	**Reiniciar el servicio de red.**	Media	La aplicación brinda la opción de reiniciar el servicio de red cuando sea necesario.
2	***Mostrar descripción de las tarjetas de red.***	Media	Se muestran características físicas de cada tarjeta de red existente en el sistema.

No	Nombre	Prioridad	Descripción
3	***Configurar interfaces físicas de red.***	Alta	La aplicación brinda la opción de mostrar la información de cada interfaz de red y configurarla a través del método estático o dinámico (dhcp).
4	***Configurar interfaces virtuales de red.***	Alta	La aplicación brinda la opción de crear, modificar y eliminar interfaces de red virtuales, estas son creadas a partir de una interfaz física.
5	***Balancear carga entre inter-faces físicas de red.***	Alta	La aplicación brinda la opción de realizar *bonding* entre interfaces de red física. Se seleccionarán dos o más interfaces de red y se establecerá el balanceo de carga entre ellas.

Tabla 3: Descripción de Historias de Usuario

2.3 Arquitectura de HMAST

HMAST es un sistema base que permite administrar servidores de forma remota, el cual posee las funcionalidades necesarias para administrar usuarios, tareas programadas y diversos servicios telemáticos (Soria y Castillo, 2012).

Arquitectura

La arquitectura que presenta HMAST propone el diseño de una arquitectura N-Capas orientada al dominio compuesta por cinco capas las cuales son descritas a continuación (*Ilustración 2*).

Capa de Presentación (***presentation***): es la que presenta al usuario los conceptos del negocio mediante una o varias interfaces de usuario.

Capa de Aplicación (***application***): realizará llamadas a servicios de la capa inferior (dominio), los servicios que publica (servicios de aplicación) tienen la responsabilidad, por ejemplo de adaptar la información que le llega a los requisitos de los servicios de dominio.

Capa de Dominio (***domain***): constituye el hilo conductor de la aplicación; está compuesta por entidades del dominio que representan objetos del dominio. Sus componentes solo dependen de la capa Infraestructura transversal (*infrastructureCrosscutting*) y están totalmente desacoplados. Implementa la lógica de dominio (reglas de negocio), es responsable de las validaciones. Define las interfaces de

persistencia a datos (contratos de repositorio), pero no los implementa, las operaciones las deben implementar los repositorios.

Capa de Persistencia (***persistences***): es responsable de contener el código necesario para persistir los datos. Los principales componentes que contendrá la capa son los repositorios: que son clases que implementan los contratos de repositorios definidos en la capa de dominio.

Capa Infraestructura transversal (***infrastructureCrosscutting***): tiene responsabilidades de promover la reutilización de código, ella albergará las operaciones seguridad, *logging*, monitoreo del sistema, mecanismos de persistencia reutilizables, validadores genéricos, en fin todas aquellas operaciones que se puedan llamar desde otras capas.

Ilustración 2: Vista lógica de la arquitectura de HMAST

Funcionalidades que brinda:

- Gestión de servidores lógicos: permite la adición, edición y eliminación de los datos de un servidor lógico, además permite la conexión remota y desconexión a un servidor seleccionado.
- Gestión de servicios telemáticos asociados a un servidor lógico: permite la adición, edición y eliminación de los datos de un módulo, así como activación y desactivación de los mismos.
- Gestión de las variables de configuración asociadas a un servidor lógico: permite cargar y salvar las variables de configuración de los servicios telemáticos encontradas en un servidor lógico (ficheros de configuración, nombre de módulos, demonios, etc.).

2.3.1 Consideraciones para implementar un módulo para HMAST

La integración de un módulo a la herramienta HMAST se logra a través de la llamada a funciones que devuelven las conexiones asociadas a servidores lógicos y permiten trabajar con las configuraciones de estos (Pérez y Pérez, 2013).

- La lógica de aplicación no deberá incluir ninguna lógica del dominio, solo tareas de coordinación relativas a requisitos técnicos de la aplicación, como conversiones de formatos de datos de entrada a entidades del dominio, llamadas a componentes de infraestructura para que realicen tareas complementarias.
- Se debe garantizar que no viajen hacia y desde la capa de presentación objetos de dominio, en su lugar deben viajar objetos DTO (*Data Object Transfer*).
- Las entidades solo pueden tener dependencias de componentes de la capa de dominio.
- Las clases de servicios deben ser las únicas responsables (vías de acceso) de acceder a los repositorios, no se puede implementar código de persistencia a datos en la capa de dominio.
- Solo se puede acceder a la información almacenada en los servidores haciendo uso de los repositorios.
- Es importante que todo el código reutilizable por más de un repositorio se ponga a disposición de todos en la capa de infraestructura transversal (Pérez y Pérez, 2013).

2.4 Arquitectura del módulo de red para HMAST

En el proceso de arquitectura, son organizadas o estructuradas las partes más relevantes de lo que se quiere desarrollar, permitiendo tener una visión común entre todos los involucrados (desarrolladores y usuarios) y clara perspectiva del sistema completo, necesarias para controlar su ejecución.

Para lograr que el sistema cumpla con todas las funcionalidades previstas, es imprescindible definir un diseño que incluya una solución simple pero a la vez efectiva, que refleje a través de principios y buenas prácticas la intención de planificación del desarrollador. Para ello es necesario puntualizar la arquitectura de software que adoptará el sistema, la distribución de las clases y paquetes definidos para la misma y por último para el diseño, los Patrones Generales de Software para la Asignación de Responsabilidades (en

inglés *General Responsibility Assignment Software Patterns* o GRASP) y Pandilla de Cuatro (en inglés *Gang of Four* o GOF) que se emplearán en la estructura de la solución.

2.4.1 Tecnologías asociadas al desarrollo del módulo de red

En el proceso de desarrollo de HMAST, previo a su implementación fue estudiado y luego seleccionado un conjunto de tecnologías y herramientas que ayudaron a definir requisitos, en su diseño, modelado e implementación. Estas tecnologías, que se encuentran especificadas en el expediente de proyecto de dicho sistema (Soria y Castillo, 2012), quedaron definidas para utilizarlas en caso de agregarle funcionalidades o realizar futuras modificaciones a esta herramienta. En el proceso para realizar el Módulo de Administración de Red para HMAST, quedaron sujetas a utilizar las tecnologías usadas en el desarrollo de su sistema raíz, las mismas fueron:

- **Java:** lenguaje de programación orientado a objetos usado para la implementación del módulo.
- **Spring framework:** *framework* de Código Abierto que ofrece un modelo de programación y de configuración completa para las aplicaciones empresariales basadas en Java para cualquier tipo de plataforma de despliegue. Para el desarrollo del módulo de red, se usan las características reutilizables que provee este *framework* para la inyección de dependencias con XML para establecer instancias de los objetos.
- **Netbeans:** este entorno de desarrollo integrado, es utilizado para el desarrollo del módulo, su integración con *subversion* permitió gestionar el código fuente en el sistema de control de versiones.
- **Visual Paradigm:** es el software de modelado usado para realizar los diagramas del proceso de desarrollo.
- **Apache Tomcat:** servidor web, utilizado para el despliegue del módulo en el servidor de aplicaciones.
- **RapidSVN:** usado como cliente SVN para la gestión del código fuente de la aplicación en el sistema de control de versiones.
- **Metodología SXP**: es una metodología de desarrollo de software elaborada a partir de las metodologías SCRUM y XP; la misma ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles, que permitan actualizar los procesos de software para el mejoramiento

de la actividad productiva (Peñalver, 2008). Esta metodología es la definida por el proyecto, para guiar todo proceso de desarrollo del software que se realiza en este. Como el equipo de desarrollo es pequeño, el cliente forma parte del mismo y las entregas se realizan en períodos cortos, SXP permitirá agilizar el análisis, diseño e implementación de la solución propuesta. Además, fomentar el desarrollo de la creatividad, el aumento del nivel de preocupación y responsabilidad de los miembros del equipo ayudando al líder del proyecto a tener un mejor control en este.

2.5 Diseño del módulo Administración del servicio de red y uso de patrones

Para realizar el diseño del Módulo para Administrar el servicio de red desde HMAST, se define la utilización de la arquitectura establecida para HMAST con el objetivo de mantener una homogeneidad entre los componentes del sistema. Además, es representado el flujo de información entre las diferentes capas a través del diagrama de paquetes. A continuación se muestra la representación de la arquitectura de la aplicación a desarrollar.

2.5.1 Diagrama de paquetes

Se muestra el diagrama de paquetes en la (*Figura 1*), observándose en este cómo se integra el módulo de red a HMAST. En este diagrama los paquetes exteriores: *presentation*, *application*, *domain*, *persistences* e *infraestructuraCrosscutting* representan lógicamente cada capa de la arquitectura definida en el proceso de desarrollo. Cada paquete contiene otro subpaquete inmediato, que evidencia el servicio que se integra en HMAST; en este caso el paquete que representa el servicio es ***network***; este a la vez contiene otros paquetes que se relacionan entre sí.

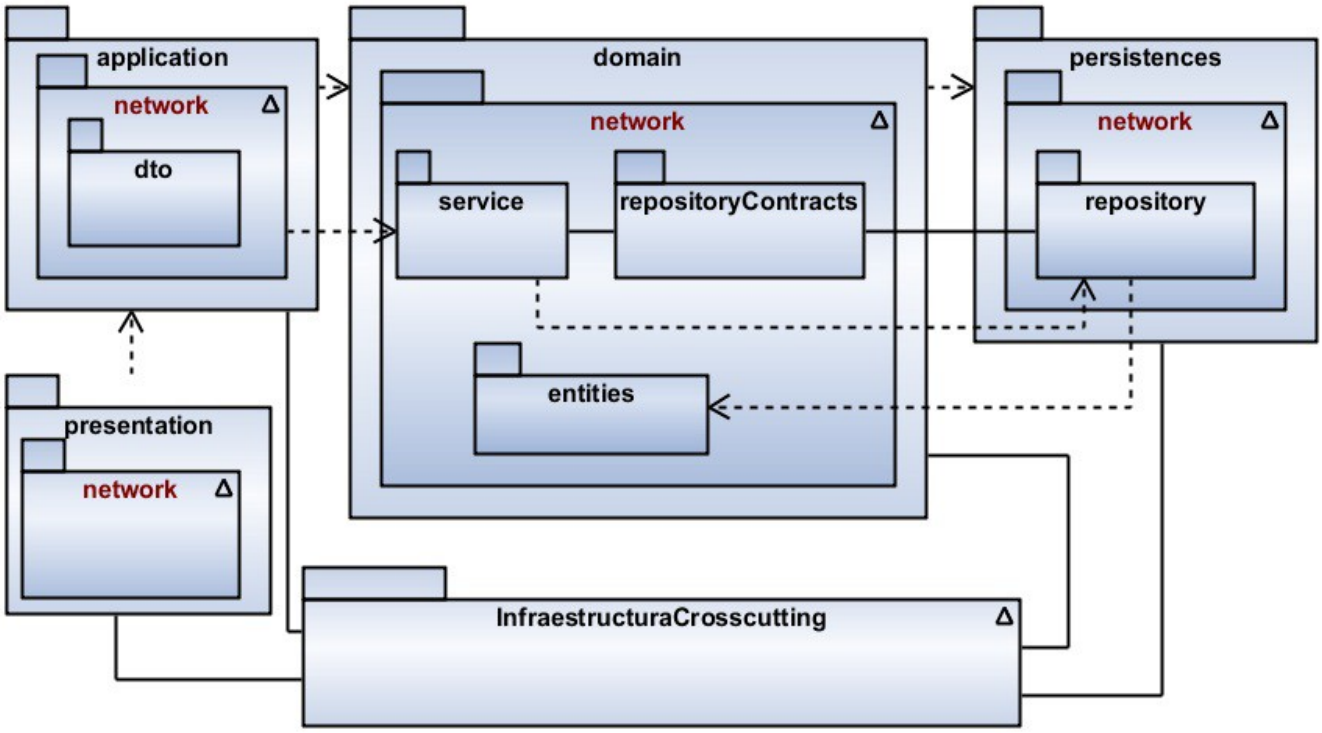


Figura 1: Diagrama de paquetes

Para detallar el contenido de los paquetes de la (*Figura 1)*, a continuación se encuentra una imagen (*Figura 2)* en la cual se muestran las clases por las cuales se relacionan los paquetes y el flujo de información a través de estas. Algunas de las clases representadas contienen atributos a través de los cuales se instancia a las clases interfaces que las implementan. Se refleja como cada paquete de módulo de red (***network***) evidencia el estilo arquitectónico establecido para el desarrollo.

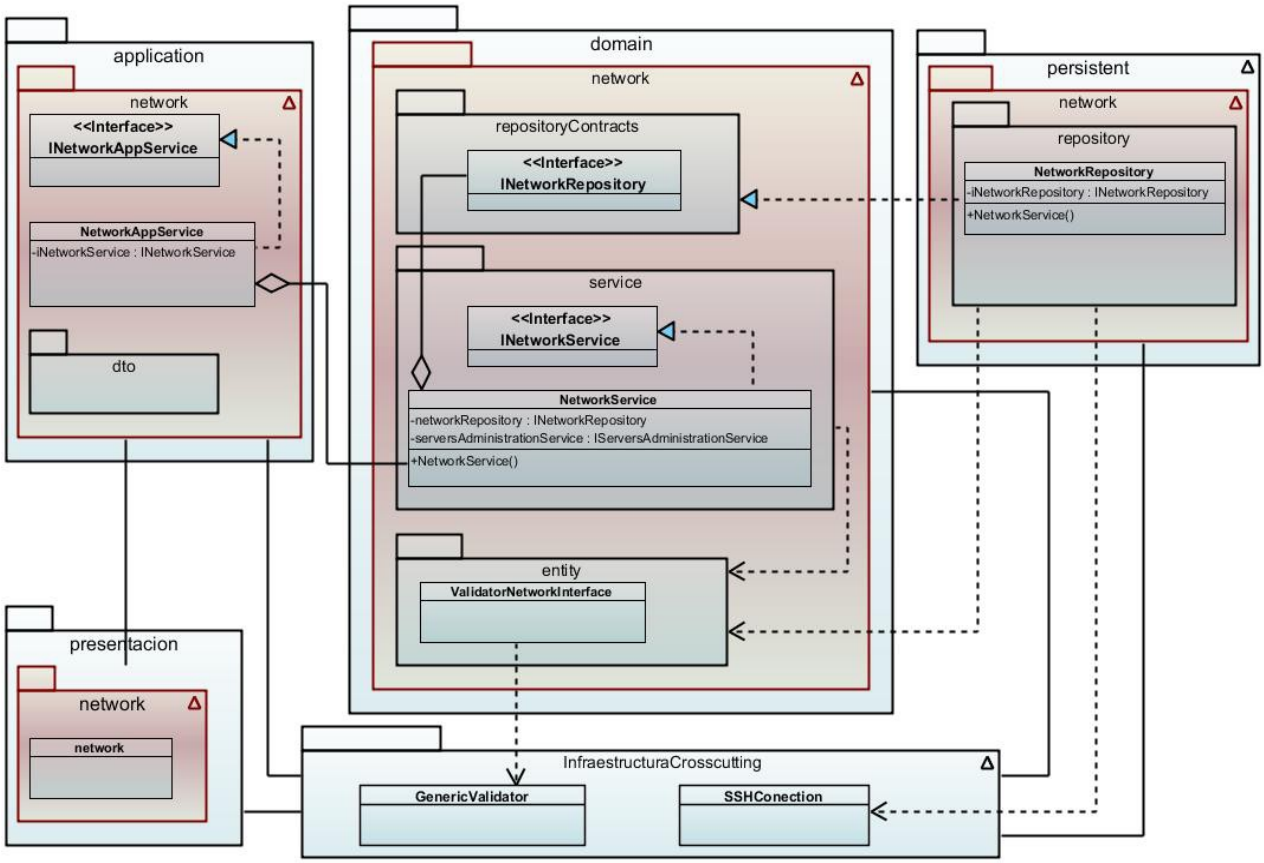


Figura 2: Integración del módulo de red a HMAST

El paquete ***presentation*** representa la capa de presentación, aquí se encuentra la clase controladora *network*, que a través de peticiones construye las interfaces de usuario (que se encuentran dentro de este paquete) y las comunica con la capa de Aplicación.

El paquete ***application*** representa la capa de aplicación, aquí se encuentra la clase interfaz *InetworkAppService, e*sta clase interfaz contiene los métodos que permiten el acceso desde el paquete ***presentation*** y es implementada por la clase *NetworkAppService*. NetworkAppService es responsable de adaptar la información que le llega desde la interfaz de usuario a los requisitos de los servicios del paquete dominio ***domain*** accedidos a través del atributo *INetworkService* de esta clase. En este paquete se encuentra el subpaquete ***DTO*** (en inglés *Data Object Transfe*r) que contiene clases que contienen la información que llega desde la capa de presentación.

El paquete ***domain*** representa la capa de dominio, este contiene tres subpaquetes: ***entities***, ***service*** y ***repositorysContracts***. En el subpaquete ***entities*** se encuentran las clases entidades que son las

contenedoras de toda la información referente al servicio de red. En el subpaquete ***service*** se encuentra la clase interfaz *INetworkService***,** esta contiene métodos que son accedidos desde la capa de aplicación y son implementados por la clase *NetworkService. NetworkService* es la responsable de realizar las validaciones de los datos antes de realizar la escritura o lectura del repositorio. El acceso al subpaquete ***repository*** se realiza a través del atributo de tipo *INetworkRepository* encontrado en esta clase. En el subpaquete ***repositorysContracts*** se encuentra la clase *INetworkRepository* que tiene definidos los métodos que son accedidos por la clase *NetworkService* desde el subpaquete ***service.*** Estos métodos definidos en *INetworkRepository* son implementados por la clase *NetworkRepository* encontrada en el paquete ***persistences*** que representa la capa persistencia; *NetworkRepository t*rabaja directamente con los ficheros y es la clase responsable de cargar y salvar la información de estos en los repositorios.

2.5.2 Diagrama de clases

En la (*Figura 3*) está representado de forma resumida el diagrama de clases, estas son entidades que se encuentran dentro del paquete (***entities***). *NetworkInterface* es la clase principal, sus atributos son objetos donde se instancia cada entidad que la compone. En las clase *InterfazFisica, InterfazVirtual, Bonding y NetworkCard* es donde se encuentra toda la información referentes a la configuración del servicio. A raíz de la clase *InterfazFisica* se crean las interfaces virtuales, por eso contiene un atributo que es una lista de tipo *InterfazVirtual* que instancia esta clase; además es a la que se les realizará el balanceo de carga. *Entre InterfazVirtual* y *Bonding* no existen relaciones de composición hacia *InterfazFisica* porque en la clase *NetworkInterface* se creó un atributo *iFisica* para cumplir con esta *función. TypeMethod* es una clase de tipo ***Enum*** en donde se encuentran los 3 tipos de métodos con que se pueden configurar las interfaces.

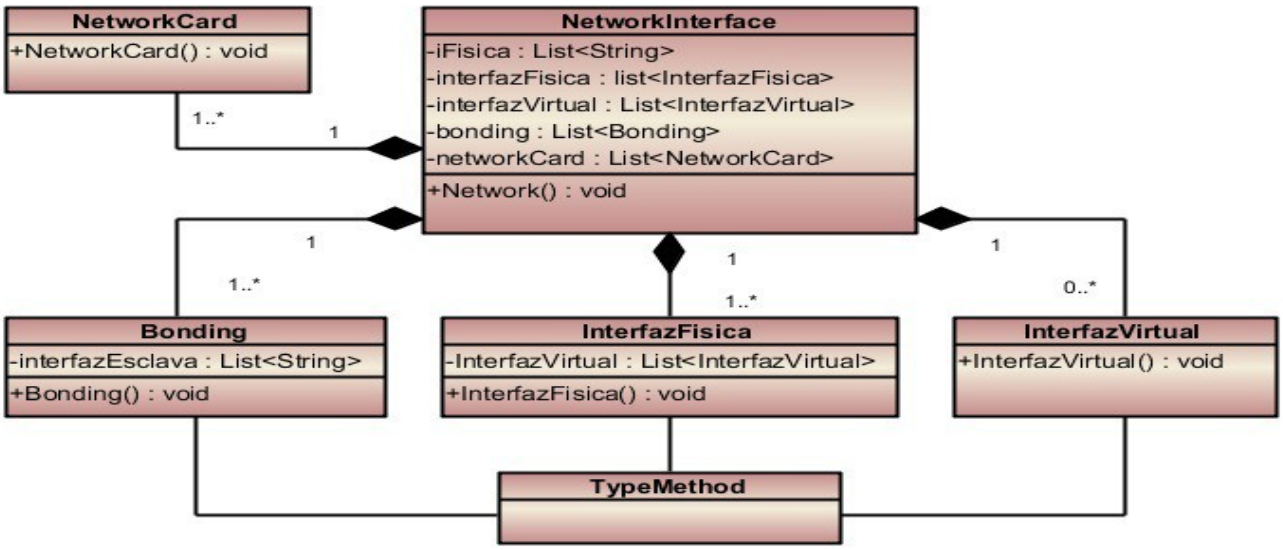


Figura 3: Diagrama de las clases entidad

La imagen que muestra la estructura completa del diagrama de las entidades, donde se pueden apreciar todas las clases, relaciones, atributos y operaciones se encuentra en el (Anexo 4).

2.5.3 Patrones de diseño utilizados

Los patrones de diseño son la base para la búsqueda de soluciones simples y elegantes a problemas específicos y comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Se basan en la experiencia adquirida y la demostración de que funcionan (*Larman*, *1999*).
Solo se aplicaron los patrones, en el caso de tener el mismo problema o uno similar al que soluciona este, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error (Richard y Vlissides, 1995). En el modelo de diseño del software se tuvieron en cuenta los patrones de diseño GRASP y GOF siguientes:

Patrones GRASP

Los patrones GRAP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Las decisiones poco acertadas dan origen a sistemas y

componentes frágiles, difíciles de mantener, entender, reutilizar o extender (Larman, 1999). Por tanto, el uso de los patrones GRASP se hace imprescindible para la elaboración de la solución. En el desarrollo de la misma se aplicaron los patrones GRASP: **Experto**, **Creador**, **Controlador**, **Bajo Acoplamiento** y **Alta Cohesión.**

Experto: es el patrón usado para la asignación de responsabilidades de forma general; es un principio básico muy útil en el diseño orientado a objetos. Con el uso de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Además, se definió en cada clase del módulo, dónde colocar la información que se necesitan de esa funcionalidad. El uso de este patrón, se evidencia en las clases *Entidades* (*Figura 3*), las mismas dada su responsabilidad, contienen toda la información referente al servicio en un aspecto específico.

Creador: este patrón tiene responsabilidad relacionada con la creación de objetos, trata el problema de qué clase debe hacer una instancia de otra. El propósito fundamental del mismo, es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Un ejemplo del uso de este patrón se evidencia en la clase entidad *NetworkInterface* (Patrón Creador), la misma tiene una relación de composición con la clase *InterfazVirtual*, por tanto contiene un atributo que se encarga de crear una instancia de esta clase. En este caso el patrón se fundamenta con la relación A contiene a B.

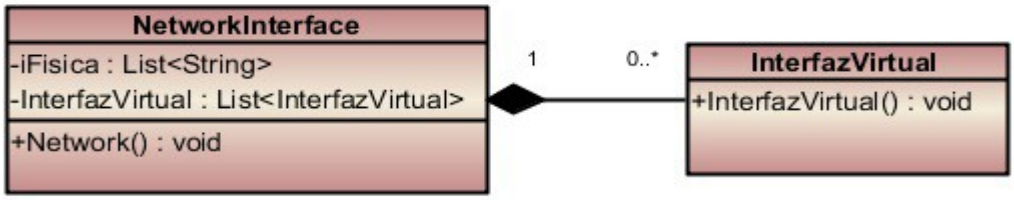


Ilustración 3: Patrón Creador

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras. Recomienda asignar las responsabilidades de tal forma que se le dé soporte a una mayor reutilización y poca dependencia (una clase no dependerá 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. Este patrón se evidencia en la *Ilustración 4*, la clase *NetworkService* se relaciona sólo con las interfaces necesarias para conectarlas a través de inyección de dependencia.

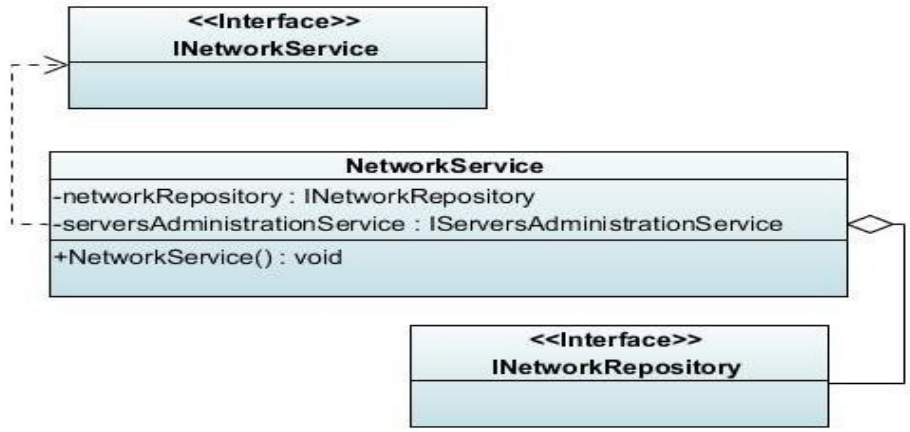


Ilustración 4: Patrón Bajo acoplamiento

Alta cohesión: La cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Se hizo necesaria la utilización de este patrón en el software en cuestión con el fin de controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas, por esta razón, las que se identificaron con una amplia cantidad de funcionalidades, se dividieron en otras clases siguiendo el propósito de distribuir de forma equitativa el peso de la complejidad, manteniendo además, la coherencia entre ellas. Ejemplo del uso de este patrón, se evidencia en la clase *NetworkRepository*, quien es la encargada de implementar métodos específicos, como por ejemplo los ***load*** para leer y escribir en los ficheros o archivos que se ejecutan.

Patrones GOF

Los patrones de diseño GOF constituyen un conjunto de reglas que describen cómo afrontar tareas y solucionar problemas que surgen durante el desarrollo del software. Para la implementación de la solución se empleó el patrón Singleton.

Patrón Solitario (Singleton): pertenece al conjunto de patrones creacionales, encargados de crear objetos, de manera que los desarrolladores no tengan que instanciar directamente, proporcionando a los programas una mayor flexibilidad para decidir qué objetos usar (Larman, 1999).

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (se pueda acceder a ella desde cualquier clase). Es usado debido a la necesidad de trabajar con el mismo objeto en distintos momentos (*Larman*, *1999*).
Este patrón se evidencia dentro la aplicación a través de la clase **SSHConecction**, que define el atributo de tipo SSHConecction para realizar la conexión con el servidor al cual se le administrará el servicio de red, se realiza una única instancia de esta clase, la cual está contenida en el paquete Infraestructura Transversal (*infrastructureCrosscutting*).

Conclusiones parciales

Durante el desarrollo de este capítulo se define la concepción del módulo y la descripción del negocio. Se establece que se implementarán los 16 requisitos funcionales organizados y descritos en 5 HU que incluyen los prototipos de algunas de las interfaces más importantes. La descripción de las HU permite un mejor entendimiento entre el programador y el cliente, llegando a un acuerdo sobre las capacidades con las que el software debe cumplir. Se explica la arquitectura de HMAST y la arquitectura a utilizar para el desarrollo, la cual refleja el intercambio del flujo entre las capas. La arquitectura definida para confeccionar el módulo de administración de red, es la misma con que está estructurado su sistema raíz (HMAST), la cual es flexible ante cualquier cambio en la lógica del negocio. Se describe el diagrama de paquetes y diagrama de clases del módulo, en los cuales se muestra como se integra el módulo de red en HMAST, el correcto uso de los patrones de diseño y cómo se relacionan las clases de las diferentes capas.

Capítulo 3: Implementación y pruebas

Para el desarrollo de esta investigación, en este capítulo se describe el proceso realizado para poder implementar el sistema quedando conformado el mismo. Son planificadas las iteraciones previstas para la implementación de los requisitos definidos y mostradas las clases más importantes, utilizadas en la construcción de funcionalidades relevantes del sistema que se desarrolla. Se describen las Tareas de Ingeniería propuestas para las Historias de Usuario más significativas en la construcción del software, destacándose responsables, descripción y tiempo de duración de cada una. Además, son propuestos los estándares de código a utilizar en la implementación del módulo y expuestas las pruebas realizadas al mismo para ganar en calidad, lo que garantiza que el módulo llegue a manos del cliente con la menor cantidad de errores posibles.

3.1 Planificación de Iteraciones

Después de ser analizadas las Historias de Usuario por parte del cliente, quedando estimado el tiempo y esfuerzo dedicado para desarrollar cada una de ellas, se procede a realizar la planificación de las etapas de implementación del sistema. Este plan concentra las HU por iteraciones, definiendo cuáles serán desarrolladas en cada iteración del proceso de implementación. Teniendo en cuenta lo planteado anteriormente se decide implementar el sistema en dos iteraciones, descritas a continuación:

- Iteración 1: La primera iteración tiene como objetivo dar cumplimiento a las HU (HU_1: *Reiniciar el servicio de red*, HU_3: *Configurar interfaces de red física* y HU_4:*Configurar interfaces de red virtual*)
- Iteración 2: En la segunda iteración se le dará cumplimiento a las HU (HU_2:*Mostrar descripción de las tarjetas de red* y HU_5: *Balancear carga entre interfaces de red física*). Con la implementación de estas funcionalidades se completan todas las configuraciones básicas con las que debe contar el servicio de red para configuras las interfaces de cada tarjeta de red y establecer conexión a la red.

3.2 Implementación

En el flujo de trabajo se empieza con el resultado del diseño y se desarrolla el sistema en términos de componentes; es decir, ficheros de código fuente, *scripts*, ficheros de códigos binarios, ejecutables y similares.

3.2.1 Principales clases utilizadas

Las clases principales que se crearon para desarrollar el sistema son: las clases entidades *NetworkInterface*, *InterfazFisica*, *InterfazVirtual* y *Bonding*, estas se encuentran en el subpaquete ***entities*** y contienen toda la información referente a las configuraciones del servicio de red. La clase *NetworkRepository* que se encuentra en el paquete ***persistence*** es donde se implementan los métodos por los cuales el sistema accede y se comunica con los ficheros necesarios utilizados para la configuración de este servicio. Esta clase es responsable de cargar y salvar la información en los repositorios.

3.2.2 Tareas de ingeniería

Las Tareas de Ingeniería se efectúan para detallar mejor las HU, facilitando con ello, el entendimiento en el proceso de implementación. Cada HU puede contener una o más tareas en caso de necesitarla, explicando paso a paso las acciones que se realizan en la misma. Los puntos de estimación asignados a cada tarea son medidos por días según las características y la complejidad que posea la misma.
Definen cada una de las actividades asociadas a las HU y permiten organizar el proceso de implementación, así como conocer el grado de complejidad de cada HU, teniendo en cuenta la cantidad de tareas asociadas. A continuación se exponen todas las Tareas de Ingeniería pertenecientes a las Historias de Usuario del sistema.

Tareas de Ingeniería para la HU Reiniciar el servicio de red

Tarea de Ingeniería

Número Tarea: 1	**Número Historia de Usuario:** HMAST_Red_1
Nombre Tarea: Estudio de comandos que permiten recargar las configuraciones de la red.	
Tipo de Tarea: Investigación	**Puntos Estimados:** 0.1
Fecha Inicio: 1/2/14	**Fecha Fin:** 3/2/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se realiza un estudio de los comandos ***service network stop*** y ***etc/init.d/networking restart***, que permiten recargar las configuraciones realizadas en el archivo **etc/network/interfaces.** Son ejecutados y se comprueban los resultados obtenidos.	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** HMAST_Red_1
Nombre Tarea: Diseño de opción que permita recargar las configuraciones de la red.	
Tipo de Tarea: Análisis y diseño.	**Puntos Estimados:** 0.1
Fecha Inicio: 2/2/14	**Fecha Fin:** 4/2/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se diseña una interfaz que permita mediante algún funcionalidad visual ejecutar el comando para recargar las configuraciones realizadas a la red.	

	Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** HMAST_Red_1
Nombre Tarea: Crear método que ejecute los comandos para recargar las configuraciones del servicio de red.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 4/2/14	**Fecha Fin:** 7/2/14
Programador Responsable: Raciel Betancourt Bueno:	
Descripción: Se crea un método que ejecute los comandos para recargar las configuraciones realizadas al	

fichero del servicio de red.

Tareas de Ingeniería para la HU Mostrar descripción de las tarjetas de red.

	Tarea de Ingeniería
Número Tarea: 4	**Número Historia de Usuario:** HMAST_Red_2
Nombre Tarea: Estudiar comandos que permitan mostrar características de las tarjetas de red del sistema.	
Tipo de Tarea: Investigación.	**Puntos Estimados:** 0.2
Fecha Inicio: 7/5/14	**Fecha Fin:** 9/5/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se realizará el estudio del comando ***lshw -c network***; que permite mostrar aspectos físicos de cada tarjeta de red del sistema, se ejecutará y se comprobarán los resultados.	

	Tarea de Ingeniería
Número Tarea: 5	**Número Historia de Usuario:** HMAST_Red_2
Nombre Tarea: Diseño de interfaz que muestre las características de las tarjeta de red del sistema.	
Tipo de Tarea: Análisis y diseño.	**Puntos Estimados:** 0.1
Fecha Inicio: 9/2/14	**Fecha Fin:** 11/2/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se diseña una interfaz que permita mostrar aspectos o parámetros físicos de la tarjeta de red, de forma básica y de una forma más detallada.	

	Tarea de Ingeniería
Número Tarea: 6	**Número Historia de Usuario:** HMAST_Red_2
Nombre Tarea: Crear método que ejecute el comando que muestra características físicas de cada tarjeta de red del sistema.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 9/5/14	**Fecha Fin:** 12/5/14
Programador Responsable: Raciel Betancourt Bueno.	

Descripción: Se crea un método que ejecute la funcionalidad: Mostrar características de las tarjetas de red ejecutando el comando estudiado.

Tareas de Ingeniería para la HU Configurar interfaces de red física.

		Tarea de Ingeniería
Número Tarea: 7	**Número Historia de Usuario:** HMAST_Red_3	
Nombre Tarea: Estudio de comandos para configurar interfaces de red física.		
Tipo de Tarea: Investigación		**Puntos Estimados:** 0.2
Fecha Inicio: 13/2/14		**Fecha Fin:** 16/2/14
Programador Responsable: Raciel Betancourt Bueno.		
Descripción: Se realizará el estudio de los comandos que permiten la configuración de las interfaces de red física, a través del método estático o dhcp. Los mismos se ejecutan para comprobar su correcto funcionamiento, estos comandos son: ***auto eth0:[0...n]:*** para declarar la interfaz **i*face eth0 inet static*	*iface eth0 inet dhcp***: para asignar el tipo de método. ***address*	*netmask*	*network*	*broadcast*	*gateway*	*dns-nameservers*	*domain:*** para la configuración.	

		Tarea de Ingeniería
Número Tarea: 8	**Número Historia de Usuario:** HMAST_Red_3	
Nombre Tarea: Diseño de interfaces que permitan configurar interfaces de red física.		
Tipo de Tarea: Análisis y diseño.		**Puntos Estimados:** 0.1
Fecha Inicio: 16/2/14		**Fecha Fin:** 18/2/14
Programador Responsable: Raciel Betancourt Bueno.		
Descripción: Se diseñan interfaces de usuario que permitan realizar todas las configuraciones que requieren las interfaces de red física. La configuración debe realizarse a través de los métodos estático o *dhcp*.		

Tarea de Ingeniería

Número Tarea: 9	**Número Historia de Usuario:** HMAST_Red_3
Nombre Tarea: Crear funciones que ejecuten los comandos que permiten configurar interfaces de red física.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 18/4/14	**Fecha Fin:** 25/4/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se implementan funciones capaces de ejecutar los comandos que permiten configurar interfaces de red física. Las mismas deben comunicar las capas y gestionar todos los parámetros que requiere su configuración a través de las interfaces de usuario	

Tareas de Ingeniería para la HU Configurar interfaces de red virtual

	Tarea de Ingeniería
Número Tarea: 10	**Número Historia de Usuario:** HMAST_Red_4
Nombre Tarea: Estudio de comandos para configurar interfaces de red virtual.	
Tipo de Tarea: Investigación	**Puntos Estimados:** 0.2
Fecha Inicio: 13/2/14	**Fecha Fin:** 17/2/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se realizará el estudio de los comandos que permiten configurar interfaces de red virtual a través del método *estático* o *dhcp.* Los mismos se ejecutarán para comprobar su correcto funcionamiento, los comandos son: ***auto eth0:[0...n]:*** para declarar la interfaz **i*face eth0 inet static*	*iface eth0 inet dhcp***: para asignar el tipo de método. ***address*	*netmask*	*network*	*broadcast*	*gateway*	*dns-nameservers*	*domain:*** para la configuración.	

	Tarea de Ingeniería
Número Tarea: 11	**Número Historia de Usuario:** HMAST_Red_4
Nombre Tarea: Diseño de interfaces que permitan configurar interfaces virtuales de red.	
Tipo de Tarea: Análisis y diseño.	**Puntos Estimados:** 0.1

Fecha Inicio: 21/22/14	**Fecha Fin:** 23/2/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se diseñan interfaces de usuario que permitan crear, listar y mostrar, modificar y eliminar interfaces de red virtual creadas a partir de una interfaz física. Para la configuración se realizará a través de los métodos estático o *dhcp*, los parámetros a configurar son los mismos que los de las interfaces físicas.	

	Tarea de Ingeniería
Número Tarea: 12	**Número Historia de Usuario:** HMAST_Red_4
Nombre Tarea: Crear funciones que ejecuten los comandos que permiten configurar interfaces de red virtual.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 28/2/14	**Fecha Fin:** 4/3/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se implementan funciones capaces de ejecutar los comandos que permiten configurar interfaces de red virtual a partir de alguna interfaz de red física existente. Las mismas deben comunicar las capas y gestionar todos los parámetros que requiere este tipo de configuración a través de las interfaces de usuario.	

Tareas de Ingeniería para la HU Realizar balanceo de carga entre interfaces físicas de red.

	Tarea de Ingeniería
Número Tarea: 13	**Número Historia de Usuario:** HMAST_Red_5
Nombre Tarea: Estudio de los comandos que permiten realizar bonding entre interfaces de red física.	
Tipo de Tarea: Investigación	**Puntos Estimados:** 0.2
Fecha Inicio: 5/4/14	**Fecha Fin:** 9/4/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se realizará el estudio de los comandos que permiten realizar el bonding entre interfaces de red física. Los comandos son: ***auto*** *bond0****:[0...n]:*** para declarar el bonding que se crea.	

iface bond0 inet static: el tipo de método asignado siempre va ser estático. ***address*** **|** ***netmask*** **|** ***network*** **|** ***broadcast*** **|** ***gateway*** **|** ***dns-nameservers*** **|** ***domain:*** para la configuración. ***slaves*** **|** ***mode*** **|** ***miimon*** **|** ***downdelay*** **|*****primary:*** configuraciones propias del bonding.

	Tarea de Ingeniería
Número Tarea: 14	**Número Historia de Usuario:** HMAST_Red_5
Nombre Tarea: Diseñar interfaz que permita realizar bonding entre interfaces de red física.	
Tipo de Tarea: Análisis y diseño.	**Puntos Estimados:** 0.2
Fecha Inicio: 10/4/14	**Fecha Fin:** 14/4/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se diseña una interfaz que permitirá crear el balanceo de carga entre interfaces de red física. Se crearán opciones para seleccionaran que interfaces serán declaradas esclavas del bonding, además de asignarle un conjunto de parámetros mencionados anteriormente para su configuración.	

	Tarea de Ingeniería
Número Tarea: 15	**Número Historia de Usuario:** HMAST_Red_5
Nombre Tarea: Crear funciones que ejecuten los comandos que permiten realizar el balanceo de carga entre interfaces de red física.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio: 14/05/14	**Fecha Fin:** 21/05/14
Programador Responsable: Raciel Betancourt Bueno.	
Descripción: Se implementan funciones capaces de ejecutar los comandos que permiten realizar el balanceo de carga entre interfaces físicas. Las mismas deben comunicar las capas y gestionar todos los parámetros que requiere este tipo de configuración a través de las interfaces de usuario.	

3.2.3 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, este debe tender siempre a lo

práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez (msdn, 2009).

Estándar para nombrar las clases:
Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto se pondrá con minúscula, cuando sea un nombre compuesto se utilizará la notación PascalCase.
Ejemplo: *InterfazFisica*

Estándar para nombrar las funciones:
El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se utiliza la notación CamelCase, y con solo leerlo se reconoce el propósito de la misma. Ejemplo: *augeasEditVirtualInterface*

Estándar para nombrar las variables:
El nombre de las variables se escribe: primera palabra con minúscula, si es un nombre compuesto se utilizara notación CamelCase. Ejemplo: *address*

Estándar para nombrar los componentes:
Todos los paquetes comienzan con cu.uci.hmast.xxx.yyy.zzz.kkk
xxx →presentation, application, domain, persistence.
yyy →nombre del modulo (network).
zzz →elementos que pueden contener los componentes verticales (*entities, repositorys*).
kkk → clases o subpaquetes.
Ejemplo: *cu.uci.hmast.domain.network.entitys.NetworkInterface*

3.3 Pruebas de software

El único instrumento adecuado para determinar el estado de la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema en su totalidad, con el objetivo de medir el grado en que la aplicación desarrollada cumple con los requisitos. Entre los métodos más efectivos para realizarse pruebas al software se destacan el método de Caja Blanca y el de Caja Negra.

Al sistema implementado se le realizarán un conjunto de pruebas para verificar que en su funcionamiento se cumple con todos los parámetros definidos. Para esto se establece realizar pruebas unitarias utilizando el método de Caja Blanca y pruebas de aceptación por parte del cliente utilizando el método de Caja Negra. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados, y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

3.3.1 Pruebas Unitarias

Las Pruebas Unitarias se realizan desde el comienzo de la implementación, ya que cada método o funcionalidad desarrollada ha sido probada a través de la clase ***main***. Para realizar las pruebas unitarias se comprueban los caminos lógicos, bucles y condiciones examinando así el estado del programa para informar de la situación real de la calidad del software. Se tienen en cuenta las características que este ofrece, utilizándolo para el diseño de casos de pruebas que se basan en un análisis detallado de la estructura del código.

En este tipo de pruebas se utilizan diferentes métodos con el objetivo de analizar el código por diferentes criterios como son:

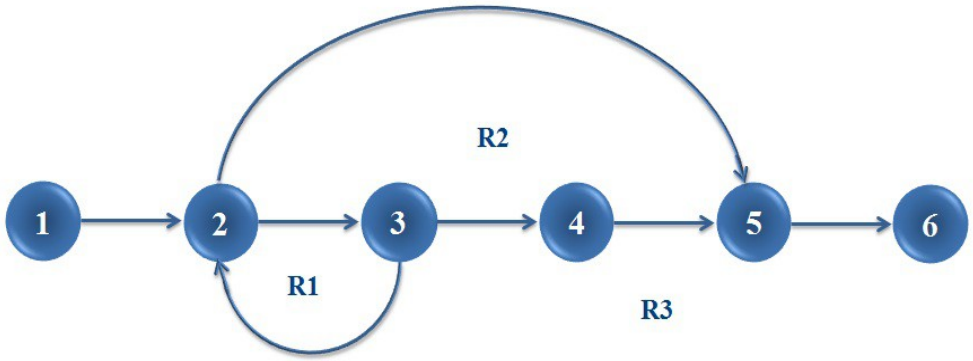
- Pruebas de camino básico.
- Pruebas de condición.
- Pruebas de bucles.

Para el desarrollo de las pruebas, se utilizará la técnica del **camino básico**. Este proceso de pruebas se realizó a lo largo de toda la implementación (desde el comienzo hasta el final del desarrollo) a todas las funcionalidades. A continuación se describe un ejemplo, realizado a una funcionalidad implementada:

Primeramente se procede a enumerar el código que se desea probar a través de regiones.

```
public void updateNetworkInterface(LogicalServer server, InterfazFisica interfaz) {
        NetworkInterface network = listNetworkServer.get(server.getId());
        for (int i = 0; i < network.getInterfazFisica().size(); i++) {
            InterfazFisica interfazFisica1 =network.getInterfazFisica().get(i);
            if (interfazFisica1.getId().equals(interfaz.getId())) {
                network.getInterfazFisica().set(i, interfaz);
                break;
            }
        }
        augeasEditPhisicalInterface(server, interfaz);
}
```

Una vez realizada la enumeración, se construye el grafo de flujo, teniendo en cuenta la notación para cada una de las instrucciones del código.



Luego se calcula la complejidad ciclomática mediante alguna de las siguientes formulas:

- V(G) = Aristas - Nodos + 2 = **[** 7-6+2=**3]**
- V(G) = Nodos predicados + 1 = **[** 2+1=3 **]**
- V(G) = Número de regiones = **[** 3 **]**

El cálculo efectuado mediante las tres fórmulas ha dado como resultado el mismo valor, por lo que se puede afirmar que la complejidad ciclomática del código es 3. Esto acota el límite superior para el número de caminos independientes que componen el conjunto básico y consecuentemente, un valor límite superior para el número de pruebas que se deben diseñar y ejecutar con el propósito de garantizar que se cubran todas las sentencias de los procedimientos.

Caminos	Secuencia
Camino básico 1	1-2-3-4-5-6
Camino básico 2	1-2-5-6
Camino básico 3	1-2-3-2-3-4-5-6
Camino básico 4	1-2-3-2-5-6

A continuación se describe un caso de prueba para uno de los caminos identificados.

Caso de Prueba para el Camino básico 1	
Nombre de quien realiza la prueba: Raciel Betancourt Bueno	**Estado:** Satisfactoria
Descripción: Para ejecutar esta prueba se necesitan como parámetros los atributos: *server* de tipo *LogicalServer* e *interfaz* de tipo *InterfazFisica.* Luego se crea el objeto *network* de tipo ***NetworkInterface*** y se recorre este desde la posición cero hasta el final. A continuación se crea el objeto *interfazFisica1* de tipo ***InterfazFisica*** y en caso que este tenga el mismo id que el parámetro *interfaz*, se obtiene la interfaz de la posición especificada en el objeto *network.* Luego *se* realiza una llamada al método *augeasEditPhisicalInterface* el cual edita el parámetro *interfaz*.	
Entradas: *server, interfaz*	
Condición de Ejecución: El id del objeto *interfazFisica1* y del parámetro *interfaz* deben ser iguales.	
Resultado: Se edita la interfaz de red física existente seleccionada a través de la lista de interfaces dado un servidor.	

3.3.2 Pruebas de Aceptación

Con el fin de demostrar el cumplimiento del objetivo planteado y la satisfacción del cliente con el resultado obtenido, se realizaron pruebas de aceptación para validar la calidad del módulo implementado. La metodología SXP en el proceso de pruebas de aceptación, propone realizar pruebas constantemente tanto como sea posible.

Con las pruebas de aceptación se verifica cada funcionalidad de cada Historia de Usuario, estas son redactadas en colaboración con el cliente para mejor entendimiento. A continuación se describe el proceso de pruebas realizado a las Historias de Usuarios más significativas para el producto. Debido al extenso número de pruebas realizadas, solo se exponen aquellas en las que todos los datos son correctos. Las demás pruebas pueden verse en el expediente de proyecto.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU3_3	**Nombre Historia de Usuario:** Configurar interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se muestran las interfaces física existentes.	
Condiciones de Ejecución: el sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú de configuraciones: Interfaz Física.	
Resultado Esperado: Se muestran las interfaces existentes y sus configuraciones.	
Evaluación de la Prueba: Satisfactorio	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU3_5	**Nombre Historia de Usuario:** Configurar interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se puede configurar una interfaz de red física estática con datos obligatorios y opcionales.	
Condiciones de Ejecución: el sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Física. Marcar la interfaz que se va a modificar. Ejecutar la opción de *Editar*. Seleccionar el método: Estático. *Se llenan los datos obligatorios Dirección IP: 10.53.3.200; Máscara: 255.255.255.0; Puerta de enlace: 10.53.3.254; Servidor(es) DNS: 10.0.0.3; Dominio: uci.cu y opcionales Red: 10.53.3.1 y Dir. de Difusión: 10.53.3.255* Salvar la configuración.	
Resultado Esperado: Se configura la interfaz con el método y datos introducidos.	
Evaluación de la Prueba: Satisfactorio	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU3_6	**Nombre Historia de Usuario:** Configurar interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se puede configurar una interfaz de red física dinámica.	
Condiciones de Ejecución: el sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Física. Marcar la interfaz que se va a modificar. Ejecutar la opción de *Editar*. Seleccionar el método: Dinámico (dhcp). Los datos obligatorios de la red, se llenarán automáticamente. Salvar la configuración.	
Resultado Esperado: Se configura la interfaz automáticamente con el método introducido.	
Evaluación de la Prueba: Satisfactorio	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU4_8	**Nombre Historia de Usuario:** Configurar interfaces virtuales de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se puede crear una interfaz de red virtual estática con datos obligatorios y opcionales.	
Condiciones de Ejecución: El sistema debe tener al menos una interfaz de red física configurada.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Virtual. Ejecutar la opción de *Crear*. Seleccionar el método: Estático. Seleccionar el Id: 2. Se llenan los datos obligatorios Dirección IP: *10.53.3.200*; Máscara: *255.255.255.0*; Puerta de enlace: *10.53.3.254;* Servidor(es) DNS: *10.0.0.3*; Dominio: *uci.cu* y opcionales Red: *10.53.3.1* y Dir. de Difusión: *10.53.3.255.*	

Salvar la configuración.
Resultado Esperado: Se configura la interfaz con el método introducido.
Evaluación de la Prueba: Satisfactorio

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU4_11	**Nombre Historia de Usuario:** Configurar interfaces virtuales de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se puede editar una interfaz de red virtual de forma estática.	
Condiciones de Ejecución: El sistema debe tener al menos una interfaz virtual.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Virtual. Marcar la interfaz que se va a modificar. Ejecutar la opción de *Editar*. Seleccionar el método: Estático. Se llenan los datos obligatorios Dirección IP: *10.53.3.200*; Máscara: *255.255.255.0*; Puerta de enlace: *10.53.3.254*; Servidor(es) DNS: *10.0.0.3*; Dominio: *uci.cu* y si se desea los datos opcionales Red: 10.53.3.1 y Dir. de Difusión: 10.53.3.255 Salvar la configuración.	
Resultado Esperado: Se edita la interfaz con el método introducido.	
Evaluación de la Prueba: Satisfactorio	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU4_13	**Nombre Historia de Usuario:** Configurar interfaces virtuales de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se puede eliminar una interfaz virtual estática ó dinámica (dhcp).	
Condiciones de Ejecución: El sistema debe tener al menos una interfaz virtual.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Virtual. Marcar la interfaz que se va a eliminar.	

Ejecutar la opción de *Eliminar*. Salvar la configuración.
Resultado Esperado: Se elimina la interfaz y sus configuraciones.
Evaluación de la Prueba: Satisfactorio

A continuación se muestran algunas excepciones que deben ser lanzadas a partir de acciones que no se realizan:

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU3_18	**Nombre Historia de Usuario:** Configurar interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se lanza excepción si hay al menos un campo obligatorio vacío.	
Condiciones de Ejecución: El sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Física. Marcar la interfaz que se va a modificar. Seleccionar la opción: *Editar*. Llenar los datos de la red obligatorios Dirección IP:____;Máscara: 255.255.255.0; Puerta de enlace: 10.53.3.254; Servidor(es) DNS: 10.0.0.3; Dominio: uci.cu. Salvar la configuración.	
Resultado Esperado: Se lanza la excepción indicando que existen datos obligatorios vacíos.	
Evaluación de la Prueba: Satisfactorio	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU3_20	**Nombre Historia de Usuario:** Configurar interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se lanza excepción si la Dirección IP está mal cuando se configura una red estática.	
Condiciones de Ejecución: el sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo.	

Seleccionar en el menú: Interfaz Física. Marcar la interfaz que se va a modificar. Seleccionar la opción: *Editar*. Se llenan los datos Dirección IP: ***0.53.3.57*** | ***10,53,3,57*** | ***10 53 3 57*** | ***10-53-3-57*** | ***.53.3.257*** | ***10.53.3.257***; Máscara: 255.255.255.0; Puerta de enlace: *10.53.3.254; Servidor(es) DNS: 10.0.0.3* y Dominio*: uci.cu*. Salvar la configuración.
Resultado Esperado: Se lanza la excepción indicando que existen datos inválidos.
Evaluación de la Prueba: Satisfactorio

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU3_21	**Nombre Historia de Usuario:** Configurar interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se lanza una excepción cuando se quiere editar una interfaz y la misma no está seleccionada.	
Condiciones de Ejecución: El sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Física. Seleccionar la opción: *Editar*.	
Resultado Esperado: Se lanza la excepción indicando que se debe seleccionar un elemento.	
Evaluación de la Prueba: Satisfactorio	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU4_23	**Nombre Historia de Usuario:** Configurar interfaces de red virtuale.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se lanza una excepción cuando se quiere eliminar una interfaz y la misma no está seleccionada.	
Condiciones de Ejecución: El sistema debe tener al menos una interfaz virtual.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Interfaz Virtual. Seleccionar la opción: *Eliminar*.	

Resultado Esperado: Se lanza la excepción indicando que se debe seleccionar un elemento.
Evaluación de la Prueba: Satisfactorio

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU5_24	**Nombre Historia de Usuario:** Balancear carga entre interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se lanza una excepción cuando se quiere editar el bonding y este no está seleccionado.	
Condiciones de Ejecución: el sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Bonding. Seleccionar la opción: *Editar*.	
Resultado Esperado: Se lanza la excepción indicando que se debe seleccionar un elemento.	
Evaluación de la Prueba: Satisfactorio	

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST_HU5_25	**Nombre Historia de Usuario:** Balancear carga entre interfaces físicas de red.
Nombre de la persona que realiza la prueba: Pablo Soria Acosta	
Descripción de la Prueba: Verificar que se lanza una excepción cuando se quiere eliminar el bonding y este no está seleccionado.	
Condiciones de Ejecución: El sistema debe tener al menos una interfaz física.	
Entrada / Pasos de ejecución: Abrir el módulo. Seleccionar en el menú: Bonding. Seleccionar la opción: *Eliminar*.	
Resultado Esperado: Se lanza la excepción indicando que se debe seleccionar un elemento.	
Evaluación de la Prueba: Satisfactorio	

3.3.3 Resultados de las pruebas

Luego de realizarse todas las pruebas por parte del cliente al sistema desarrollado se concluye que, de los 25 Casos de Pruebas realizados al finalizar la primera iteración de la implementación fueron detectadas 9 no conformidades. Estas inconformidades estaban relacionadas con errores de validación de entidades, todas fueron corregidas y solucionadas en su totalidad, quedando el cliente satisfecho con los resultados obtenidos para dar paso así a la segunda etapa de desarrollo.

Al concluir la segunda etapa del desarrollo, se realizaron un total de 22 Casos de Pruebas, detectándose 7 no conformidades, algunas asociadas con errores de sintaxis y las otras validaciones de entidades; estas no conformidades fueron en su totalidad resueltas obteniéndose buenos resultados y la satisfacción por parte del cliente. Un factor importante a tener en cuenta, es que durante el proceso de desarrollo fueron aplicadas pruebas a nivel de programador, lo cual permitió que en las pruebas finales de cada iteración de desarrollo, se obtuvieran un menor número de errores. Además, en las pruebas realizadas por el cliente se comprobó que el módulo cuenta con los requisitos establecidos por el mismo y que estos funcionan correctamente.

3.4 Diagrama de Componentes

La descripción y realización del diagrama de componentes se encuentran en el expediente de proyecto del sistema.

3.5 Diagrama de Despliegue

Teniendo en cuenta que el módulo de red es una aplicación que se integró al sistema HMAST; para este software no se aplica un despliegue por separado; ya que esto solo lo realiza el sistema raíz. El módulo se verá desplegado cuando se despliegue HMAST y todos sus componentes. Es recomendable cuando se despliegue HMAST con el módulo de red integrado, colocar los servidores que forman parte de la infraestructura, en una zona protegida para aumentar el grado de seguridad.

Conclusiones parciales

En este capítulo, fueron definidas las iteraciones en las que se implementaron las HU en dependencia a su importancia (prioridad) para el negocio, posibilitando así establecer un orden para este proceso. Quedaron definidas y descritas las Tareas de Ingeniería propuestas para darle cumplimiento a las HU, facilitando el trabajo del desarrollador y permitiendo un considerable ahorro de tiempo. Una vez desarrollada las funcionalidades del software, se procedió a la documentación y ejecución de pruebas. Las pruebas Unitarias realizadas a cada una de las funcionalidades del software, permitieron detectar errores en la implementación del código y validación existentes; por otra parte con las pruebas de Aceptación por parte del cliente se verificó la calidad del producto. De esta forma se detectaron no conformidades que fueron erradicadas en su totalidad; luego de solucionadas estas, se obtiene un sistema que cumple con las funcionalidades definidas al inicio de la investigación. Luego de concluir el proceso de implementación y pruebas realizadas al módulo, el mismo se encuentra integrado a HMAST y en condiciones de ser aplicado donde se necesite.

Conclusiones

Al culminar esta investigación se arribó a las siguientes conclusiones:

- Entre las herramientas estudiadas Webmin, Zentyal y NetworkManager; Zentyal es la que más aspectos positivos aporta para la solución del problema, pero no permite su integración a HMAST por la arquitectura de esta herramienta.
- El estudio de herramientas que administran el servicio de red permitió identificar características y funcionalidades ajustables a las necesidades existentes en los OACE.
- La implementación del diseño de clases para la solución propuesta, acopladas a la arquitectura de HMAST cubre el 100% de las funcionalidades definidas.
- El desarrollo de este software resuelve las deficiencias existentes en HMAST para administrar el servicio de red, siendo favorable su utilización en los OACE.

Recomendaciones

Con la realización del presente trabajo se desarrollaron funcionalidades para HMAST que permitirán la administración del servicio de red. Para futuras investigaciones y desarrollos se recomienda:

- Agregarle al módulo funcionalidades para gestionar la red de otro tipo de interfaz como las ***wlan,*** para poder obtener red de forma inalámbrica en caso de que el hardware posea este tipo de tarjeta.

Bibliografía referenciada

1. ArchLinux. NetworkManager (Español). 2013. *[Consultado: Febrero 2014]. Disponible en:* https://wiki.archlinux.org/index.php/NetworkManager_(Espa%C3%B1ol).
2. *Barrios Dueñas, Joel. Configuración De Servidores* Con GNU/Linux. México, Alcance Libre, 2014. *784p. Disponible en:* http://www.alcancelibre.org/staticpages/index.php/suscripciones
3. *Benjamin, MH. Free software fundation, working together for free software. (2011). Disponible en:* http://www.fsf.org.
4. *Carlos RM. Copyleft - software libre: libertad de conocimiento, libertad de creación. (2008). Disponible en:* http://www.softwarelibre.cl/drupal//?q=node/28.
5. *Castillo Arbelo, Reidiel; Soria Acosta, Pablo. Herramienta para la Migración y Administración de Servidores (HMAS). Pregrado, Universidad de las Ciencias Informáticas, la Habana, 2012.*
6. *Castillo, Nelson. Más sobre ventajas y desventajas de GNU/Linux y Windows. 2013.* Disponible en: http://www.somoslibres.org/modules.php?name=News&file=article&sid=1678
7. *cepEP. El Software Libre en Venezuela y la Soberanía Tecnológica.* 2009. Disponible en: http://www.cepep.org.ve.
8. Coronado Zuñiga, Juan Manuel; Hernández Pino, Ulises. Implementación de Servicios Telematicos con SW Libre en proyectos de impacto social. 2004. Disponible en: http://www.iered.org/joiner/charlas/2004-05-06_Implementacion-SwLibre.pdf
9. *Dawson, Terry; Kirch, Olaf. Guía de Administración de Redes con GNU/Linux. 2000. Disponible en:* http://www.linuxdoc.org/LDP/nag/nag.html.
10. *De la Torre Llorente, César; Zorrilla Castro, Unai; Ramos Barroso, Miguel Angel; Calvarro Nelson, Javier. Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0. (Beta), Krasis pres. 2010. [Consultado Enero 2014]. Disponible en: http://www.campusmvp.com/catalogo/Product-Gu%EF%BF%BDa-de-Arquitectura-N-Capas-orientada-al-Dominio-con-.NET-4.0---Microsoft_109.aspx.*
11. *Ivar Jacobson, G. B. El proceso unificado de desarrollo de software. Habana, Félix Varela,*

2004.

12. *Jorba Esteve, Josep y Suppi Boldrito, Remo. Administración Avanzada de GNU/Linux. Universidad de Cataluña, Marzo 2004. 472 p. ISBN: 84-9788-116-8*
13. *Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos s.l: Patience Hall, Hispanoamerica, 1999. 272-279p*
14. *Martinez Soler, Rafael. Software libre, historia y evolución. Tipos de licencia. Software libre en la administración pública. 2006.* Disponible en: http://www.martinezsoler.es/web/es/swl.
15. *Martínez, Rafael. Ventajas y desventajas del software libre. 2012. [Consultado: Enero 2014]. Disponible en:* http://www.linux-es.org/node/2132.
16. *Mattews Stallman, Richard. Visión general del sistema gnu. (2010).* Disponible en: http://www.gnu.org/gnu/gnu-history.es.html.
17. *Mazzarri, Milton. Tecnología, Internet, Software Libre y GNU/Linux. Network Manager: Facilitando el manejo de redes inalámbricas. 2010 [Consultado: Febrero 2014]. Disponible en:* http://milmazz.com.ve/archivos/2007/04/23/network-manager-facilitando-el-manejo-de-redes-inalambricas/.
18. *Oracle. Administración de Oracle Solaris: interfaces y virtualización de redes. Oracle Solaris 11. 2011. Disponible en:* http://docs.oracle.com/cd/E26921_01/html/
19. *Paumier Samón, Ramón; Pérez Villazón,Y; Meneses Abad, A. Guía Cubana de Migración a Software Libre. Cuba. (2008). [Consultado: Octubre 2013]. Disponible en:* <http://es.scribd.com/doc/17779155/Guia-Cubana-Migracion-a-Software-Libre>
20. *Pedrero Ortega, Francisco J. Bonding. 2007. Disponible en: http://es.opensuse.org/Bonding.*
21. *Peñalver Romero, Gladis Marsi. Metodología ágil para proyectos de software libre [online]. S.l, Universidad de las Ciencias Informáticas, La Habana, 2008.*
22. *Pérez Villazón, Yasiel; Pérez Villazón, Yadiel. Módulo de administración de correo electrónico. Universidad de las Ciencias Informáticas, La Habana, 2013.*
23. *Perpiñan, Antonio. Administracion de redes gnu linux. Interfaces de red. 2012, Disponible en: http://www.codigolibre.org.*
24. *Pressman, Roger S. Ingeniería del Software un enfoque practico. 5ta Edición. Madrid, IEEE Software, 2001, 614p. Disponible en: http://www.pressman5.com.*
25. *Poratti, Gustavo Gabriel. Redes la guía de referencia actual y definitiva. Buenos Aires, MP*

Ediciones SA, 2004. 309p. Disponible en: www.tecnimes.com

26. *Fernández Calvo, Rafael. Glosario básico inglés-español para usuarios de Internet. Disponible en:* http://www.ati.es.
27. *Red Hat. NetworkManager. [Consultado: Febrero 2014]. Disponible en:* http://www.redhat.com/magazine/003jan05/features/networkmanager/.
28. *Reynoso, Carlos; Kiccillof, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. 0.1. Disponible en:* <http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp>.
29. *Richard Helm, Ralph Johnson y John Vlissides, Erich Gamma. Design Patterns: Elements of Reusable Object-Oriented Software. ISBN: 0-201-63361-2. 1995.*
30. *Stallman, Richard. Free Software Foundation. El proyecto GNU – Fundación para el software libre. Disponible en:* http://www.gnu.org.
31. *Rodríguez, Anahí; Soria, Valeria. VIII Jornada de software Libre. No todo es codificar – Modelos de Madurez en SL.*
32. *Seoane ED, José A. Redes de Computadoras. 2008. Disponible en: http://www.econ.uba.ar/www/departamentos/*
33. *Widdows, Chris; Duijnhouwer, Frans-Willem. Open Source Maturity Model. 2003. Disponible en:* www.Seriouslyopen.org.
34. *William, Stalling. Comunicaciones y redes de computadores. 6Ta Edición. 2000. Prentice Hall, 734p. Disponible en: http://www.shore.net/- ws/COASe.html*

Bibliografía General

- *Características de Java como Lenguaje de programación. [Consultado: Noviembre 2013]. Disponible en:* http://java.sun.com.
- *El servidor Apache Tomcat | Linux, Java y programación. [Online]. [Consultado: Noviembre 2013]. Disponible en la Web:* http://casidiablo.net/el-servidor-apache-tomcat/.
- *IDE de Programación [Consultado: Noviembre 2013]. Disponible en:* http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

- *RapidSVN. In: [online]. [Consultado Noviembre 2013]. Disponibles en:* http://www.rapidsvn.org/.
- *SpringSource. Spring Framework. In: Springsource [online]. [Consultado: Noviembre 2013]. Disponible en:* http://www.springsource.org/spring-framework.
- *Users List. Visual Paradigm for UML. In: Visual Paradigm [online]. [Consultado 20 Noviembre 2013]. Disponible en:* http://www.visual-paradigm.com/.
- *Welcome to NetBeans. [Consultado: Noviembre 2013]. Disponible en:* http://netbeans.org/.
- *Webmin. NetworkConfiguration. [Consultado: Octubre 2013]. Disponible en:* http://www.webmin.com/; http://doxfer.webmin.com/Webmin/NetworkConfiguration.
- *Zentyal. Zentyal para administradores de redes. Servicio de Red. (2011). [Consultado: Octubre 2013]. Disponible en:* http://www.zentyal.com/es/, http://www.zentyal.com/es/for/-smbs/deploy-network-infraestructure.
- *Zentyal S.L. Zentyal 3.3 Documentación Oficial. Primeros pasos con Zentyal. La interfaz web de administración de Zentyal. [Consultado: Diciembre 2013]. Disponible en:* http://doc.zentyal.org/es/index.html.
- Msdn. Revisiones de código y estándares de codificación, 2009 *[Consultado: Abril 2014]*. Disponible en: http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.
- Las pruebas de software forman el único instrumento adecuado para determinar la calidad de software, 2010. [Consultado: Abril 2014]. Disponible en: http://pruebasdesoftware.com/laspruebasdesoftware.htm.

Glosario de Términos

1. **Copyright**: derecho que tiene un autor, incluido el autor de un programa informático, sobre todas y cada una de sus obras y que le permite decidir en qué condiciones han de ser éstas reproducidas y distribuidas. Aunque este derecho es legalmente irrenunciable puede ser ejercido de forma tan restrictiva o tan generosa como el autor decida. El símbolo de este derecho es ©.
2. **Copyleft**: términos de distribución de un programa, sometido a derechos de autor, que otorgan a los demás los derechos a utilizar, modificar y redistribuir el código del programa o cualquier programa derivado del mismo, pero sólo si los términos de distribución no son cambiados a fin de garantizar que el código y las libertades se hacen legalmente inseparables.
3. **DBS:** licencia que permite el uso del código fuente en software no libre. El autor bajo esta licencia, mantiene la protección de copyright únicamente para la renuncia de garantía y para requerir la adecuada atribución de la autoría en trabajos derivados, pero permite la libre redistribución y modificación. Tiene menos restricciones en comparación con otras.
4. **DHCP:** Protocolo de Configuración Dinámica del Anfitrión (Dynamic Host Configuration Protocol), usado para proporcionar automáticamente información como direcciones IP, máscaras de subred e información de encaminamiento (enrutamientos) entre computadoras. Si se usa en la red DHCP, se necesitará un cliente DHCP para poder conectarse a ella mediante este protocolo.
5. **Dirección IP:** dirección de un ordenador dentro de una red con protocolo TCP/IP.
6. **Framework:** es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
7. **GPL:** licencia creada por la Free Software Foundation en 1989 (la primera versión) orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.
8. **Historias de Usuario (HU):** secuencias de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de las secuencias.

9. **HU significativos para el negocio:** HU que representa procesos de gran importancia en la línea del negocio.
10. **IDE:** Entorno de desarrollo integrado en inglés (*Integrated Development Environment)*. Es un programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.
11. **Instalar:** incorporar a la computadora un programa o dispositivo para ser utilizado.
12. **IPV4:** Dirección IP versión 4, direcciones de 32 bits empleadas por las computadoras para el acceso a la red.
13. **Kernel:** Núcleo. Parte esencial de un sistema operativo que provee los servicios más básicos del sistema. Se encarga de gestionar los recursos como el acceso seguro al hardware de la computadora, determina qué programa accederá a un determinado hardware, si dos o más programas quieren usarlo al mismo tiempo, entre otras.
14. **Licencia**: contrato entre el desarrollador de un software sometido a derechos de autor y el usuario, en el cual se definen con precisión los derechos y deberes de ambas partes. Es el desarrollador, o aquel a quien haya cedido los derechos de explotación, quien elige la licencia según la cual distribuye el software.
15. **LDAP:** módulo que realiza las tareas básicas para la gestión de un servicio de directorio, tales como la vista general del árbol de directorio, la gestión de entradas, como por ejemplo, cuentas de usuario y correo electrónico, alias de correo, grupo, unidad organizativa, entre otros.
16. **MAC:** es un identificador hexadecimal de 48 bits que se corresponde de forma única con una tarjeta o interfaz de red.
17. **Máscara de red:** es una combinación de bits que sirve para delimitar el ámbito de una red de computadoras. Su función es indicar a los dispositivos qué parte de la dirección IP es el número de la red, incluyendo la subred, y qué parte es la correspondiente al host.
18. **Privativo:** el software no libre (también llamado software propietario, software privativo, software privado, software con propietario o software de propiedad) se refiere a cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo, cuyo código fuente no está disponible.
19. **Servidor**: computadora central de un sistema de red que provee servicios y recursos (programas, comunicaciones, archivos, etc.) a otras computadoras (clientes) conectadas a ella.

20. **SXP:** metodología compuesta por las metodologías SCRUM y XP, especialmente indicada para proyectos pequeños, rápido cambio de requisitos donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad.
21. **TCP/IP**: Transfer Control Protocol / Internet Protocol. Protocolo que se utiliza en Internet para transmitir datos. El TCP está orientado a la conexión que establece una línea de diálogo entre el emisor y el receptor antes de que se transfieran los datos. IP, define el protocolo que permite identificar las redes y establecer los caminos entre los diferentes ordenadores.

Anexo 1: Modelo de dominio

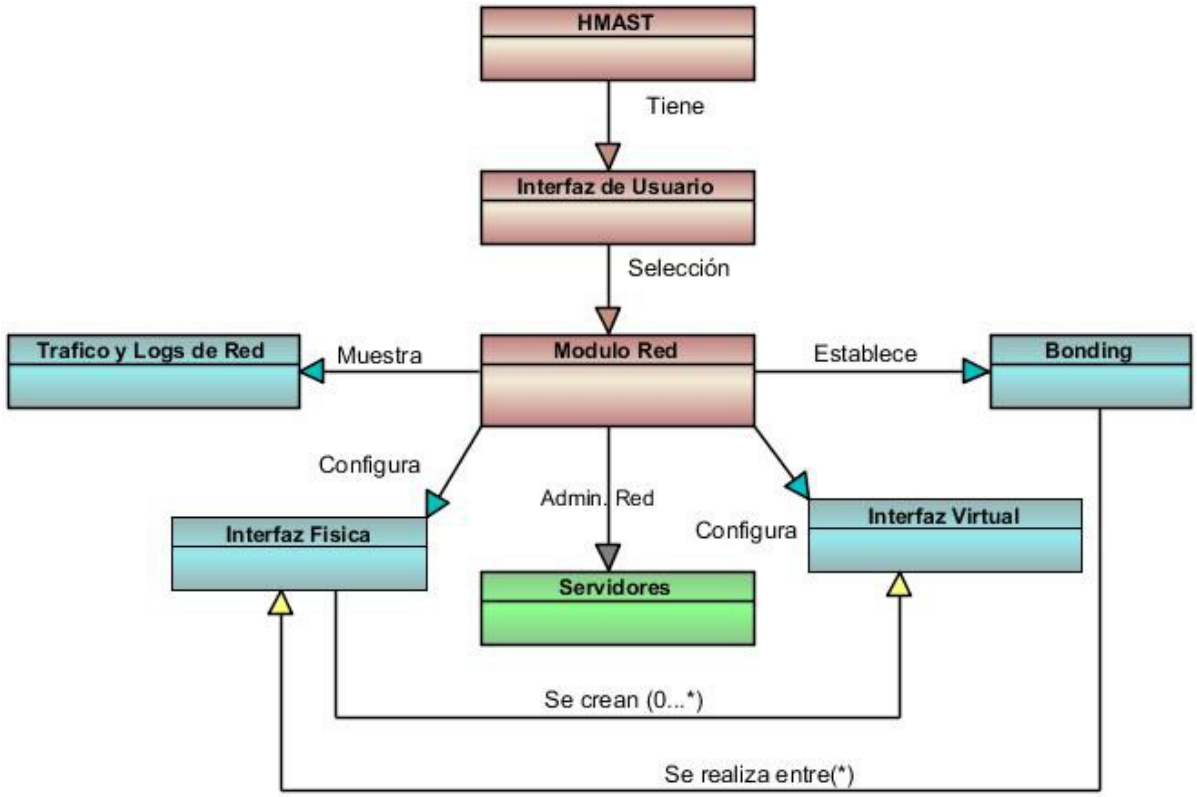


Figura 4: Modelo de dominio

Anexo 2: Lista de Reserva del Producto

Asignado a	Ítem *	Descripción	Estimación	Estimado por
Prioridad			Alta	
Desarrollador	1	*Mostrar interfaces disponibles.*		Raciel Betancourt
Desarrollador	2	*Modificar información de interfaz de red física.*		Raciel Betancourt
Desarrollador	3	*Mostrar interfaz de red física.*		Raciel Betancourt
Desarrollador	4	*Comprobar conexión de la red.*		Raciel Betancourt

Desarrollador	5	*Crear interfaz de red virtual.*		Raciel Betancourt
Desarrollador	6	*Mostrar interfaz de red virtual.*		Raciel Betancourt
Desarrollador	7	*Modificar interfaz de red virtual.*		Raciel Betancourt
Desarrollador	8	*Eliminar interfaz de red virtual.*		Raciel Betancourt
Desarrollador	9	*Crear balanceo de carga entre interfaces físicas de red.*		Raciel Betancourt
Desarrollador	10	*Activar balanceo de carga.*		Raciel Betancourt
Desarrollador	11	*Desactivar balanceo de carga.*		Raciel Betancourt
Desarrollador	12	*Mostrar balanceo de carga.*		Raciel Betancourt
Desarrollador	13	*Editar balanceo de carga.*		Raciel Betancourt
Desarrollador	14	*Eliminar balanceo de carga.*		Raciel Betancourt
Prioridad		Media		
Desarrollador	15	*Reiniciar servicio de red.*		Raciel Betancourt
Prioridad		Baja		
	16	*Mostrar características de las tarjeta de red.*		
RNF (Requisitos No Funcionales)				
Desarrollador	1	*Crear interfaz con apariencia sencilla e intuitiva desde la cual se pueda tener acceso a todas las funcionalidades del módulo.*		Raciel Betancourt
Desarrollador	2	*La aplicación se va a ejecutar en sistemas operativos libres.*		Raciel Betancourt
Desarrollador	3	*Usar como lenguaje de programación Java con el framework de desarrollo Spring.*		Raciel Betancourt
Desarrollador	4	*Utilizar como entorno de desarrollo integrado Net Beans en su versión 7.3.1.*		Raciel Betancourt
Desarrollador	5	*Utilizar como herramienta de modelado a Visual Paradigm en su versión 8.0.*		Raciel Betancourt
Desarrollador	6	*Utilizar como servidor web Apache Tomcat.*		Raciel Betancourt
Desarrollador	7	Disponer en el sistema operativo GNU/Linux del paquete Augeas.		Raciel Betancourt

Tabla 4: Lista de Reserva del Producto.

Anexo 3: Historias de Usuario

Historias de Usuario de la 1ra iteración.

Historia de Usuario	
Número: HMAST_Red_3	**Nombre Historia de Usuario:** *Configurar interfaces de red física.*
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Raciel Betancourt Bueno	**Iteración Asignada:** 1ra
Prioridad en Negocio: Alta	**Puntos Estimados:** 3 Semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 3 Semanas
Descripción: La aplicación brinda la opción de mostrar y configurar la información de las interfaces de red física de forma estática o automática.	

Observaciones: La aplicación, permitirá poder realizar configuraciones a interfaces de red física. Para ello, las directivas, sentencias, comandos y datos (*separados por espacio cada uno*) se escribirán y guardarán en el sistema en el archivo de configuración **/etc/network/interfaces**. Antes de guardar los datos introducidos se verificará que estos sean correctos.

Mostrar interfaces disponibles: Se mostrará al usuario el estado de cada interfaz de red física disponible en el sistema. A través de ***ifconfig*** se muestra la información de las interfaces de red activas y con ***ifconfig -a*** se muestra el estado de todas las interfaces activas o no. Ejemplo de esto seria:

eth0 Link encap:Ethernet direcciónHW 00:1c:c0:29:36:1a
Direc. inet:10.53.3.219 Difus.:10.53.3.255 Másc:255.255.255.0
Dirección inet6: fe80::21c:c0ff:fe29:361a/64 Alcance:Enlace
ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1
Paquetes RX:491774 errores:0 perdidos:0 overruns:0 frame:0
Paquetes TX:234816 errores:0 perdidos:0 overruns:0 carrier:0
colisiones:0 long.colaTX:1000
Bytes RX:521598089 (521.5 MB) TX bytes:23374511 (23.3 MB)
Interrupción:20 Memoria:90300000-90320000

Modificar información de interfaz de red física.*:* La aplicación permitirá configurar interfaces de red

física a través del método estático o dinámico (*DHCP*). La configuración en el fichero se realizará de la siguiente forma: primeramente debe introducirse la sentencia ***auto*** seguida de ***eth***[0..**n**], ***n*** es el número de la interfaz que se va a configurar. Luego, en la siguiente línea se establece el método escribibiéndose la sentencia: ***iface eth0 inet,*** seguido de: ***static*** si es IP estático o ***dhcp*** si es IP dinámico.

-*Dirección IP estática*. En el sistema por defecto este modo estará activado. El usuario deberá establecer la configuración de forma manual introduciendo en los campos disponibles los datos necesarios para conectarse a una red determinada. La sentencia para establecer este tipo de configuración es ***iface eth0 inet static; etho*** es la interfaz de tipo Ethenet**.** Los campos habilitados para introducir los datos son los de las sentencias siguientes: ***address* IPv4 (***dirección IP*); ***netmask* IPv4 (***máscara de red*) y ***gateway* IPv4 (***puerta de enlace o pasarela*), se deberá realizar de forma obligatoria. Además, de forma opcional se introducirán ***network* IPv4** (*red*), ***broadcast* IPv4** (*dirección de difusión*), ***domain* IPv4 o alfanumérica** (*dominio*) y al menos un ***dns-nameservers* IPv4** o **alfanumérica (***servidor de nombres DNS*), si son varios servidores, cada uno debe escribirse separado por espacio o por coma.

-*Dirección IP dinámico (DHCP).* Este tipo de configuración se realizá a través de las sentencias: ***iface eth0 inet dhcp*** permitiendo que el sistema obtenga la configuración de la interfaz de forma automática mediante un servidor DHCP**.** Cuando se seleccione este modo el sistema deshabilitará los campos obligatorios (*dirección IP, máscara de red, puerta de enlace),* los campos opcionales (*red* y *dirección de difusión)*, los servidores de nombre (DNS) y el dominio respectivamente.

-A través de un tercer método (No configurado) la aplicación permitirá limpiar los datos de la configuración automáticamente, borrando toda la información de cada campo.

Para poder editar, tiene que seleccionarse primeramente la interfaz deseada en la lista de interfaces. Luego de ser editados los datos en uno, varios o todos los campos, debe haber un botón para salvar los cambios realizados en el fichero del sistema. También habrá un botón *Cancelar*, para deshacer los datos introducidos, dejando la configuración como se encontraba inicialmente. Ejemplo de lo mencionado anteriormente para configurar las interfaces físicas:

```
auto eth0
iface eth0 inet static
address 10.53.3. 219
netmask 255.255.255.0
network 10.53.3.0
broadcast 10.53.3.255
gateway 10.53.3.254
dns-nameservers 10.0.0.3 10.0.0.4
dns-search uci.cu
```

Mostrar interfaces de red física: Se mostrarán de las interfaces de red física creadas las configuraciones: dispositivo (interfaz), método, dirección IP, máscara de red, puerta de enlace, dirección de difusión, dominio y servidores DNS.

Comprobar conexión de la red: Luego de haber configurado la interfaz de red, se puede verificar si hay realmente conexión haciendo una comprobación ejecutando el mandato ***ping*** hacia cualquier dirección de la red local o el propio host. Ejemplo de esto sería:

ping 10.53.3.219
PING 10.53.3.219 (10.53.3.219) 56(84) bytes of data.
64 bytes from 10.53.3.219: icmp_req=1 ttl=64 time=0.029 ms
64 bytes from 10.53.3.219: icmp_req=2 ttl=64 time=0.033 ms

Prototipo de interfaz:

Tabla 5: HU_3 Configurar interfaces físicas de red.

Historia de Usuario	
Número: HMAST_Red_4	**Nombre Historia de Usuario:** *Configurar interfaces de red virtual.*
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Raciel Betancourt Bueno	**Iteración Asignada:** 1ra
Prioridad en Negocio: Alta	**Puntos Estimados:** 3 Semanas

Riesgo en Desarrollo: Medio	**Puntos Reales:** 3 Semanas

Descripción: La aplicación brinda la opción de crear, modificar y eliminar interfaces de red virtual, estas son creadas a partir de una interfaz de red física existente.

Observaciones: La aplicación, permitirá poder crear una interfaz de red virtual a partir de alguna interfaz de red física existente, es decir que para crear una virtual tiene que existir una física. Como se mencionaba anteriormente para realizar la configuración, las directivas, sentencias, comandos y datos (*separados por espacio cada uno*) se escribirán y guardarán en el archivo de configuración **/etc/network/interfaces**. Antes de guardar los datos introducidos se verificará que los mismos sean correctos.

Crear interfaz de red virtual: La aplicación mediante una opción ejecutará esta acción permitiendo poder crear hasta 9 interfaces virtuales por cada interfaz de red física existente. Para crear la interfaz de red virtual debe introducirse en el fichero lo siguiente: ***auto*** seguido de ***eth***[0..**n**] (***n*** es el número de la interfaz de red física por la cual se va a crear), dos puntos '**:**' y el número de la interfaz virtual (número de un dígito que comienza por cero '0' que indica de que interfaz virtual se trata), quedando de la forma **eth0:0.** Si ha sido creada otra interfaz virtual anteriormente, el número '***n'*** de la nueva interfaz virtual debe ser consecutivo al de esta. Después de esto se tiene que establecer uno de los dos métodos para la configuración introduciendo las sentencias: ***iface eth0:0 inet*** seguido de: ***static*** si es IP estático o ***dhcp*** si es IP dinámico, el resto de la configuración se realiza de la misma manera que en las interfaces físicas.

Cuando se introduzcan los datos deseados en los campos, debe haber un botón para salvar los cambios realizados en el fichero del sistema También habrá un botón *Cancelar*, para deshacer los datos introducidos, dejando la configuración como se encontraba inicialmente. Ejemplo de lo mencionado anteriormente para crear y configurar las interfaces virtuales sería:

```
iface eth0:0 inet static
address 10.53.3.56
netmask 255.255.255.0
network 10.53.3.0
broadcast 10.53.3.255
gateway 10.53.3.254
dns-nameservers 10.0.0.3 10.0.0.4
dns-search uci.cu
```

Mostrar interfaces de red virtual configuradas: Se mostrarán de las interfaces de red virtual creadas las configuraciones siguientes: dispositivo (interfaz física por la cual creada), Id de interfaz, método, dirección IP, máscara de red, puerta de enlace, dirección de difusión, dominio y servidores DNS.

Modificar interfaz de red virtual: Para modificar una interfaz de red virtual, se debe seleccionar esta en la lista de interfaces virtuales y luego accionar un botón para ejecutar esta acción. Se permitirá modifi-

car los datos de los campos que estaban habilitados cuando se creó, la interfaz de red física por la cual esta basada no se podrá editar. Luego de ser editados los datos en uno, varios o todos los campos, debe haber un botón para salvar los cambios realizados en el fichero del sistema. Si en algún campo existen datos inválidos o en caso de que sean campos obligatorios estos se encuentren vacíos no se podrá salvar la configuración realizada. También se tendrá un botón Cancelar, para deshacer los datos introducidos, dejándola con su configuración inicial.

Eliminar interfaz de red virtual: Para eliminar la interfaz de red virtual, la aplicación debe permitir seleccionar esta en la lista de interfaces y luego a través de un botón se ejecutará esta acción eliminándola del archivo en donde se encontraba.

Prototipo de interfaz:

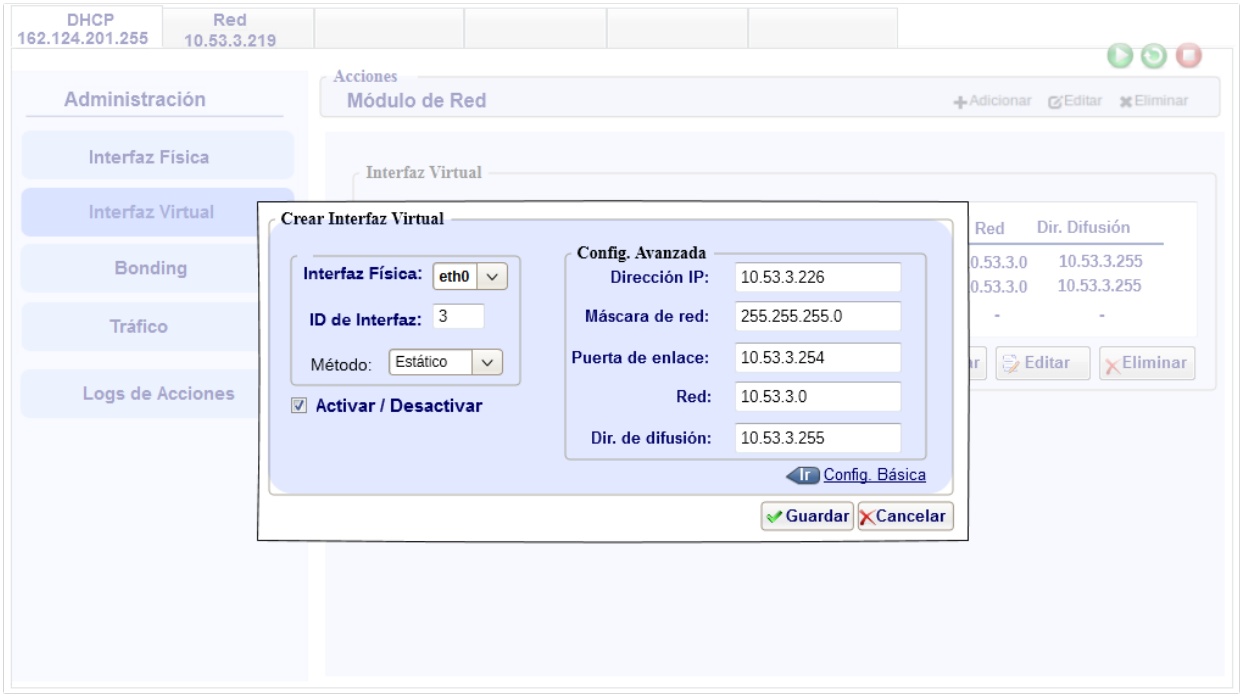


Tabla 6: HU_4 Configurar interfaces de red virtual

Historia de Usuario	
Número: HMAST_Red_1	**Nombre Historia de Usuario:** Reiniciar el servicio de red.
Modificación de Historia de Usuario Número: Ninguna	

Usuario: Raciel Betancourt Bueno	**Iteración Asignada:** 1ra
Prioridad en Negocio: Medio	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1 Semana
Descripción: Se podrá reiniciar el servicio de red cuando se desee por parte del administrador.	
Observaciones: El software brindará la opción de reiniciar servicio de red cuando sea necesario. **Reiniciar servicio de red:** El software debe tener una opción que permita *Reiniciar* el servicio después de hacer algun tipo de configuración, permitiendo al sistema cargar y establecer la nueva información introducida. Para esta operación se deberá ejecutar en el sistema las sentencias siguientes: ***/etc/init.d/networking restart*** ó ***service network restart.***	
Prototipo de interfaz: 	

Tabla 7: HU_1 Reiniciar el servicio de red

Historias de Usuario de la 2da iteración.

Historia de Usuario	
Número: HMAST_Red_2	**Nombre Historia de Usuario:** *Mostrar descripción de las tarjetas de red.*
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Raciel Betancourt Bueno	**Iteración Asignada:** 2da
Prioridad en Negocio: Bajo	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Bajo	**Puntos Reales:** 1 Semana
Descripción: Se muestran los parámetros físicos de cada tarjeta de red existente en el sistema.	
Observaciones: **Mostrar características de las tarjetas de red:** La aplicación debe mostrar aspectos o características físicas de cada tarjeta de red que tenga el sistema, sin importar de que tipo sea esta. Para estos parámetros de cada tarjeta se deberá ejecutar las sentencias: ***lshw -c network***. De la información que devuelve la petición realizada, se tomarán y mostrarán específicamente lo necesario para el usuario. Ejemplo de esto sería:	

lshw -c network

PCI (sysfs)
  *-network
      descripción: Ethernet interface
      producto: 82566DC Gigabit Network Connection
      fabricante: Intel Corporation
      id físico: 19
      información del bus: pci@0000:00:19.0
      nombre lógico: eth0
      versión: 02
      serie: 00:1c:c0:29:36:1a

Prototipo de interfaz:

Tabla 8: HU_2 Mostrar descripción de las tarjetas de red

Historia de Usuario	
Número: HMAST_Red_5	**Nombre Historia de Usuario:** *Balancear carga entre interfaces físicas de red.*
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Raciel Betancourt Bueno	**Iteración Asignada:** 2da
Prioridad en Negocio: Alta	**Puntos Estimados:** 3 Semanas
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2.5 Semanas
Descripción: La aplicación brinda la opción de realizar balanceo de carga (bonding) a tarjetas de red.	

Observaciones: La aplicación permitirá seleccionar interfaces físicas de la lista de interfaces para poder crear bonding entre estas. Se establece el ***bonding*** a través de la interfaz ***bond*****,** este dispositivo creará un puente entre las interfaces seleccionadas estableciendo una sola dirección Ip y dirección física (MAC) para todas. La MAC del ***bond[0...n]*** será la MAC del primer dispositivo esclavo (interfaz). Para configurar los *bonding* se debe instalar el paquete **ifenslave** el cual provee las herramientas necesarias para realizar esta acción. Como se mencionaba anteriormente para la configuración, las directivas, sentencias, comandos y datos (*separados por espacio cada uno*) se escribirán y guardarán en el archivo de configuración **/etc/network/interfaces**. Antes de guardar los datos introducidos se verificará que los mismos sean correctos.

Crear balanceo de carga entre interfaces físicas de red: La aplicación mediante una opción permitirá realizar el balanceo de carga entre tarjetas de red. Sólo se podrá realizar esta acción si existen 2 o más interfaces fisicas de red, se escribirá en el fichero el siguiente contenido: **auto bond0,** seguido en la

siguiente línea las sentencias ***iface bond0 inet static.*** El método estático es el único que se podrá usar para este tipo de configuración ya que los parámetros propios del bonding (*primary mode, miimon, downdelay, updelay*) que son números enteros (cero por defecto) no se podrán obtener a través de un servidor DHCP. Para obtener conexión a la red, la configuración a realizar es la misma que las de las interfaces físicas y virtuales. Luego de esto se deben seleccionar las interfaces físicas que se declaran como esclavas a través del parámetro ***slaves.*** Considerando que se tiene las interfaces eth0 y eth1, el contenido de esta configuración sería:

```
auto bond0
iface bond0 inet static
address 10.3.10.41
netmask 255.255.255.0
gateway 10.3.10.254
slaves eth0 eth1
bond_mode 0
bond_miimon 10
bond_downdelay 150
```

Activar balanceo de carga: La aplicación permitirá activar las interfaces del bonding configurado introduciendo las sentencias ***up /sbin/ifenslave bond0 eth0 eth1.*** En caso de tener más interfaces que estas (**eth0 eth1**), se escribirán a continuación de estas separadas por espacio**.** Después de reiniciar el sistema, se puede verificar usando el comando ***dmesg***, si el canal bonding fue creado exitosamente.

Desctivar balanceo de carga: La aplicación permitirá desactivar las interfaces del bonding configurado introduciendo las sentencias ***down /sbin/ifenslave -d bond0 eth0 eth1*****.** En caso de tener más interfaces que estas (**eth0 eth1**), se escribirán a continuación de estas y separadas por espacio**.** Después de reiniciar el sistema, se puede verificar usando el comando ***dmesg***, si el canal bonding fué desactivado exitosamente.

Mostrar bonding configurados: Se mostrarán las configuraciones de cada bonding configurado, mostrándose de cada uno: dispositivo (***bond[0...n]***), método, dirección IP, máscara de red, puerta de enlace, interfaces esclavas, interfaz primaria y modo.

Editar balanceo de carga: Para modificar un bonding entre interfaces físicas de red, se debe seleccionar el mismo en la lista que se muestra de bonding configurados. La aplicación da la oportunidad de modificar todos los datos disponibles cuando se crea el bonding. Permitiendo quitar o agregar otras interfaces al mismo.

Eliminar balanceo de carga: Se podrá eliminar el bonding configurado, la aplicación debe permitir seleccionar este de la lista de bonding y luego a través de un botón se ejecutará esta acción eliminándolo del archivo en donde se encontraba.

Prototipo de interfaz:

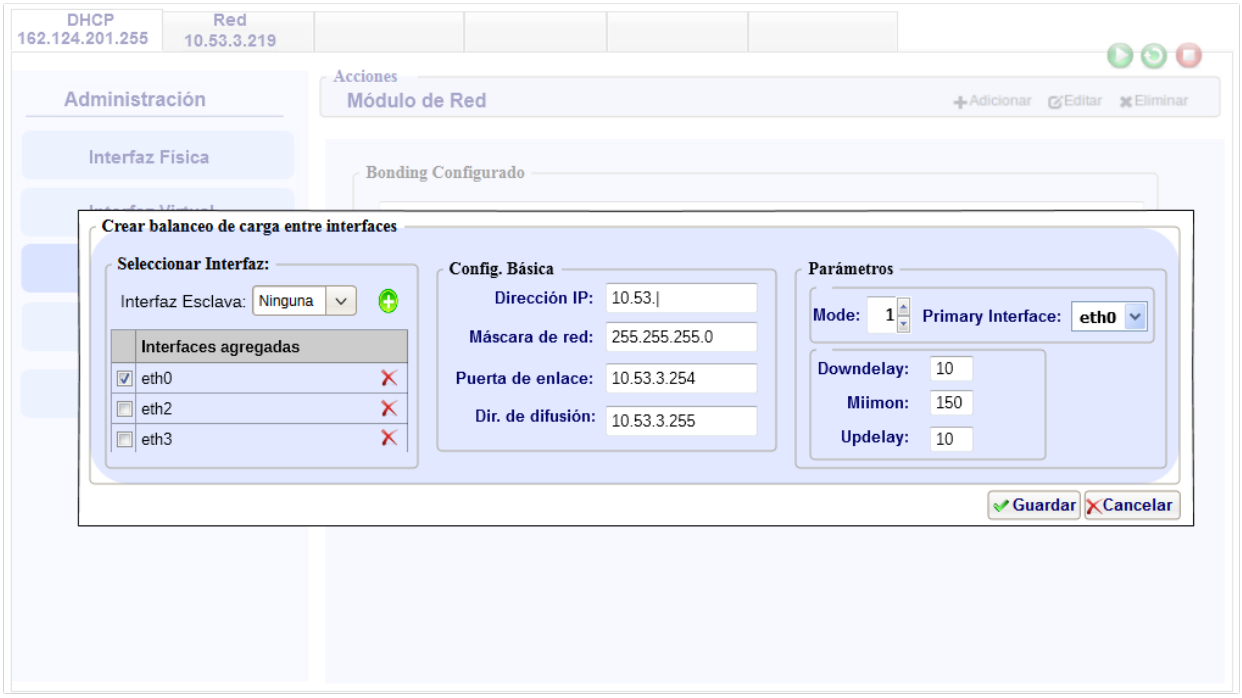


Tabla 9: HU_5 Balancear carga entre interfaces de red física.

Anexo 4: Diagrama de las clases Entidades

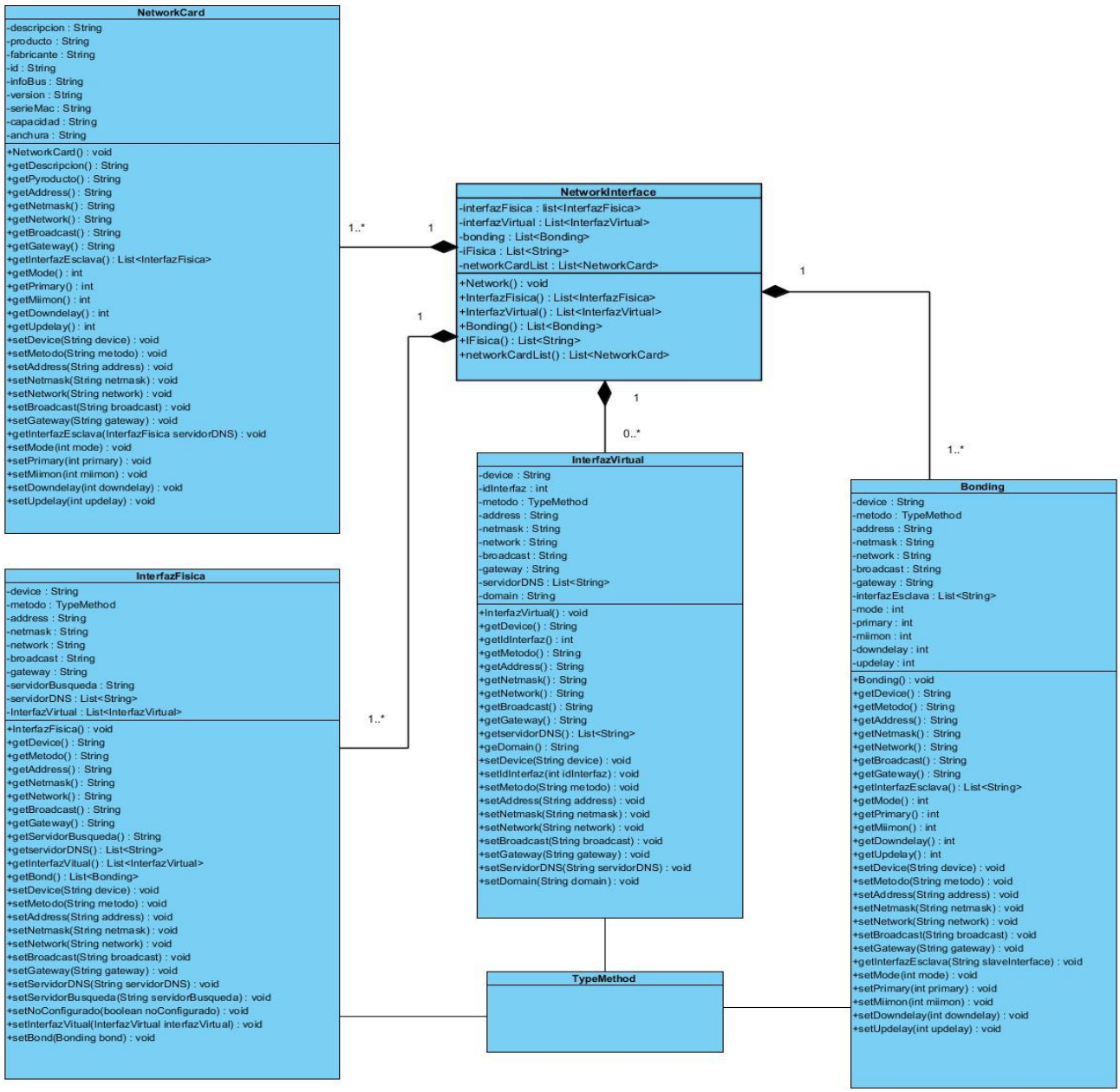


Figura 5: Diagrama de las clases Entidad (Atributos y Operaciones)