Universidad de las Ciencias Informáticas

Facultad 1

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

Título:

Módulo para la administración de cortafuegos desde la herramienta para la migración y administración de servicios telemáticos (HMAST).

Autor:

Javier Ramón Vila

Tutores:

MsC. Angel Goñi Oramas

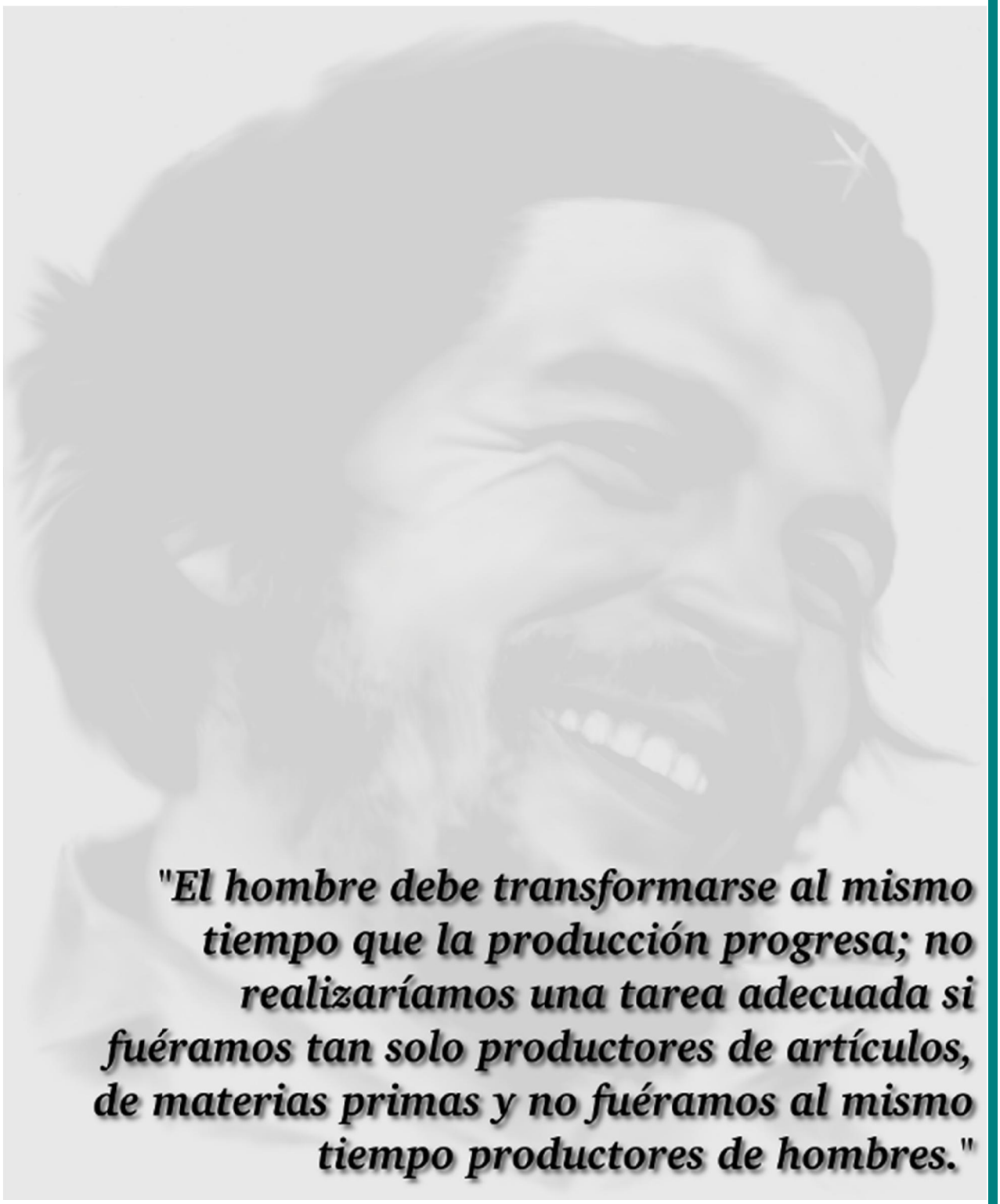
Ing. Reidiel Castillo Arbelo

Consultante:

Ing. Jailen García Gonzalez

Ciudad de la Habana, Cuba.
10 de Junio de 2014.

Pensamiento



Declaración de Autoría

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

Para que así conste firmamos la presente a los 10 días del mes de junio del año 2014.

_____
Javier Ramón Vila

_____ _____
MsC. Angel Goñi Oramas Ing. Reidiel Castillo Arbelo

Agradecimientos

A ti madre *querida, protagonista de mis andanzas y fiel espectadora de mi vida, te agradezco desde lo más profundo de mi corazón, gracias por confiar en mí y siempre estar cuando más te necesito.*

A ti papá*, aunque te encuentres lejos, sigues dándome consejos y apoyándome en todo momento, gracias por darme las fuerzas para ser hoy quien soy.*

A mi hermano*, siempre ejemplo a seguir en todo, siempre trabajando y sacando para adelante todo tipo de cosas, gracias por enseñarme a perseverar.*

A mi abuela*, gracias por tus sabios consejos y por darme todo tu amor siempre.*

A mi Arlette (mi niña)*, siempre pendiente de todo, preocupada por todo, inmersa en muchos problemas pero nunca dejó de pelear las batallas conmigo, gracias por darme los ánimos cuando más los necesitaba, por levantarme cada vez que me caía y por estar a mi lado en cada batalla.*

A mi familión*, gracias por siempre estar pendiente de mí, ofreciendo ayuda incondicional en todo momento.*

A mis amigos *Hanoi y Yosvany, siempre seremos los 3 mosqueteros, gracias por ayudarme en estos 5 años que hemos compartido donde nos hemos vuelto hermanos.*

A mis compañeros de aula *sobre todo a Dany, Gustavo, Jesús, Yamilka, Eva, Isabel y Juan Carlos a los que siempre les di dolores de cabeza, compartí risas, alegrías, tristezas y sobre todo, esos momentos especiales que quedarán atesorados por siempre.*

A mis tutores *Reidiel y Ángel por siempre apoyarme y corregir millones de veces mis equivocaciones, gracias por todos esos momentos que me dedicaron.*

A ***Alexander*** *y a* ***Pablo*** *por ayudarme a salir de ese hueco bien profundo en que se convirtió el programa, gracias por tenderme sus manos y ayudarme a echar para adelante.*

A todas esas personas *que de una forma u otra, en el transcurso de estos cinco años, han vivido momentos tristes y alegres a mi lado. A los que me ayudaron en la realización de este trabajo. A todos gracias por haber contribuido en la formación de la persona que soy hoy.*

Dedicatoria

A mis padres por ser mis mayores amigos, mis cómplices, las personas que me alentaron a seguir cuando ya no tenía fuerzas, por ser los mejores padres de este mundo.

A todas aquellas personas a las que les debo estar aquí.

Resumen

En el año 2012 el departamento de Sistemas Integrales en Migración, Asesoría y Soporte, perteneciente al Centro de Software Libre de la Universidad de las Ciencias Informáticas, desarrolló una herramienta para migrar y administrar servicios telemáticos conocida como HMAST. Esta herramienta cuenta con una versión del módulo para la administración del cortafuego pero presenta errores y carece de funcionalidades que le permitan administrar la tabla Nat presente en dicha tecnología de seguridad. Continuar con el desarrollo de este módulo para aumentar la competitividad de HMAST respecto a sus homólogos es el objetivo del presente trabajo. Para esto se realizó un estudio de las herramientas que administran el cortafuego e implementan las funcionalidades a desarrollar. Además, se documentan las tecnologías, herramientas y lenguajes de programación utilizados, así como los artefactos requeridos por la metodología de desarrollo utilizada: SXP. Como resultado se obtiene un módulo con una interfaz intuitiva y fácil de usar que cuenta con funcionalidades que permiten instalar y desinstalar el software, gestionar todas las tablas de Iptables y administrar las reglas del cortafuego.

Palabras claves: administración, cortafuego, seguridad informática, Iptables.

Índice de Contenido:

Índice de Tablas:

Índice de Ilustraciones:

Introducción

Siendo los ordenadores partícipes activos de los mayores logros alcanzados hoy en día, los hombres se han dado cuenta de la facilidad que tienen para ser explotadas sus vulnerabilidades a pesar de todas las tecnologías implantadas como medio de seguridad. Siendo así que pueden ser usados indebidamente; algunos de estos usos son la falsificación de identidad, sobrecarga de la red, alteración del sistema y desvío de recursos. Se incurre así en una falta de seguridad informática que es la que asegura la estabilidad de los dispositivos y elementos críticos del sistema.

En los servidores GNU/Linux es muy importante el manejo de los paquetes de la red, o sea, se confiere gran importancia a la seguridad mediante el cortafuego. Todavía algunos de estos paquetes se manejan de forma manual, mediante *script* o programas como Iptables, que no es más que la administración de las reglas del cortafuego a nivel del núcleo de Linux. Esto convierte a veces en engorroso su correcta administración ya que se hace muy difícil escribir manualmente cada regla, por lo que se hace necesario para la Herramienta de Migración, Asesoría y Soporte (HMAST) su administración de manera sencilla e intuitiva por medio de una interfaz gráfica. Esta herramienta es la que se utilizará para la migración y administración de los servidores de Windows hacia las plataformas de GNU/Linux en los Órganos y Organismos de la Administración Central del Estado (OACE).

En Cuba desde octubre del 2002 se está llevando a cabo un proceso de migración de sistemas operativos privativos, cuyo objetivo es que todas las entidades cubanas adopten tecnologías de Software Libre y código abierto. Ya existen entidades y organizaciones que han asumido el cambio y en estos momentos utilizan tecnologías libres, entre los cuales se encuentra la Universidad de las Ciencias Informáticas (UCI). Esta institución es uno de los principales actores de esta tarea en todo el país.

En el departamento de Sistemas Integrales de Migración, Asesoría y Soporte (SIMAYS), perteneciente al Centro de Software Libre (CESOL) de la UCI, se encuentra en desarrollo la herramienta HMAST para la migración y administración de las diferentes versiones de Windows Server hacia plataformas libres. Esta herramienta en su primera versión contenía un módulo para la administración del cortafuego el cual brindaba funcionalidades que permiten la gestión de las reglas presentes en la tabla Filter. También, en el módulo desarrollado, se encontraron funcionalidades que presentan deficiencias o que no se encuentran implementadas. Esto lo convierte así en una herramienta poco competitiva con respecto a sus alternativas de Software Libre y código abierto que implementan todas las tablas de una forma más completa, brindando así poca seguridad a la hora de migrar o administrar un servidor al no tener una protección por cortafuego de forma eficaz. Algunas de estas deficiencias son:

- Permite introducir reglas de forma repetida.
- Cambia el bit de reenvío desde el inicio de la aplicación, disminuyendo el nivel de seguridad de los servidores. Además, no permite que se modifique el estado de este bit.
- No elimina una regla que haya sido introducida.
- No gestiona la instalación o desinstalación del cortafuego.
- No realiza un conjunto de validaciones importantes como verificar que los IP, máscaras, protocolos, rangos y puertos introducidos son correctos.
- Carece de funcionalidades que permitan el enrutamiento de paquetes mediante las tablas Nat.

Una vez descrita la problemática anterior se define el siguiente problema científico a resolver: ¿Cómo mejorar la competitividad técnica de HMAST con respecto a las alternativas basadas en tecnologías de Software Libre y código abierto?

El objeto de estudio del presente trabajo se centra en las herramientas de código abierto que administran el cortafuego, enmarcándose el **campo de acción** en el módulo de administración de cortafuego de HMAST.

Por lo que se propone como **objetivo general** desarrollar funcionalidades que permitan la administración de la tabla Nat en el módulo de cortafuego de la herramienta HMAST.

Para un mejor desarrollo del trabajo se ha desglosado el objetivo general en los siguientes **objetivos específicos**:

- Realizar un estudio de los softwares que implementan el cortafuego y las herramientas que los administran.
- Analizar y diseñar las funcionalidades a desarrollar.
- Realizar la implementación y prueba de las funcionalidades requeridas.

Con el propósito de dar cumplimiento a los objetivos planteados se hace necesario realizar las siguientes **tareas de investigación**:

- Revisión y análisis de bibliografía especializada en servicios de cortafuego y las herramientas que lo administran sobre plataformas GNU/Linux.

- Caracterizar módulo de cortafuego HMAST.
- Análisis y diseño de las funcionalidades para la administración del servicio cortafuego.
- Implementación y prueba de las nuevas funcionalidades del módulo definido.

Para la realización de este trabajo se tiene la siguiente idea a defender: El desarrollo de funcionalidades que permitan la administración de la tabla Nat en el módulo de cortafuego de la herramienta HMAST mejorará su competitividad técnica con respecto a las alternativas basadas en tecnologías de Software Libre y código abierto.

En el desarrollo de la investigación es utilizado los siguientes **métodos científicos**:

Analítico – Sintético: Se utiliza en la investigación de las herramientas que administran cortafuego en GNU/Linux. Ayuda a separar los mismos con el objetivo de realizar un estudio del funcionamiento y las configuraciones necesarias de estos, facilitando el desarrollo de las funcionalidades propuestas.

Análisis documental: Se enfoca en la revisión de la literatura especializada en la administración de cortafuego para consultar la información necesaria en el proceso de investigación.

El documento está compuesto por 3 capítulos, las conclusiones generales, la bibliografía general utilizada y el glosario de términos. La estructura de los capítulos se define a continuación:

Capítulo 1: Fundamentación teórica: Se realiza un estudio acerca de las herramientas que administran el cortafuego, así como sus principales características y funcionalidades. Además, se abordan conceptos claves que serán usados durante el desarrollo de la investigación y se describen las tecnologías, lenguajes de programación, metodología y herramientas utilizadas en el desarrollo del sistema.

Capítulo 2: Diseño del sistema: Se presenta la fase de Planificación-Definición definida por la metodología SXP para dar solución al problema científico y se exponen los requisitos funcionales y no funcionales del sistema propuesto. También se realiza la descripción de las Historias de Usuario y se explica la arquitectura del sistema.

Capítulo 3: Implementación y pruebas: Se implementa la solución propuesta. Además, se explican las clases principales del sistema, se confecciona el plan de pruebas y se realizan las pruebas necesarias para validar que las funcionalidades desarrolladas dan cumplimiento a los requisitos planteados.

Capítulo 1: Fundamentación teórica.

En el presente capítulo se realiza un estudio de las herramientas que permiten administrar el cortafuego en los servidores que se encuentran con sistemas operativos GNU/Linux, además de documentarse las principales funcionalidades de los mismos. También, se abordan conceptos claves que serán usados durante el desarrollo de la investigación y se describen las tecnologías, lenguajes de programación, metodología y herramientas utilizadas en el desarrollo del sistema.

1.1 Conceptos asociados a la herramienta.

Para un mejor entendimiento sobre las funcionalidades a implementar y agregar al módulo de cortafuego de la herramienta HMAST, es necesario comprender conceptos importantes asociados a las tecnologías de cortafuego, Netfilter e Iptables los cuales son descritos en los siguientes subepígrafes.

1.1.1 Cortafuegos

Un cortafuego o *firewall* es un sistema que se protege a sí mismo y de otros servidores de los atacantes en redes inseguras, tales como Internet. El cortafuego puede bloquear los paquetes y conexiones basadas en una variedad de criterios, tales como la dirección de origen, dirección de destino, puerto y protocolo. Normalmente, un cortafuego es también un *router*, reenviando los paquetes entre una red local segura y la Internet. Sin embargo, también es posible que un sistema se proteja solo [1].

Controla tanto la comunicación desde el exterior como el tráfico generado desde la propia máquina o red interna. Actúa a base de normas que establece el administrador de seguridad o, en su defecto, el administrador de red o el usuario final. Dichas reglas definen las acciones correspondientes a llevar a cabo cuando se recibe un paquete que cumpla unas determinadas características. A pesar de que existen programas que se venden bajo la denominación de *firewall*, el mismo no es un programa, sino que consiste en un conjunto de medidas de *hardware* y software destinadas a asegurar una infraestructura de red.

En definitiva, se trata de cualquier sistema empleado para "separar" una máquina o una subred del resto, protegiéndola de servicios y protocolos que puedan suponer una amenaza a la seguridad desde el exterior, para mayor entendimiento véase la ilustración 1. El espacio protegido por un cortafuego se denomina perímetro de seguridad, mientras que la red externa recibe el nombre de zona de riesgo.

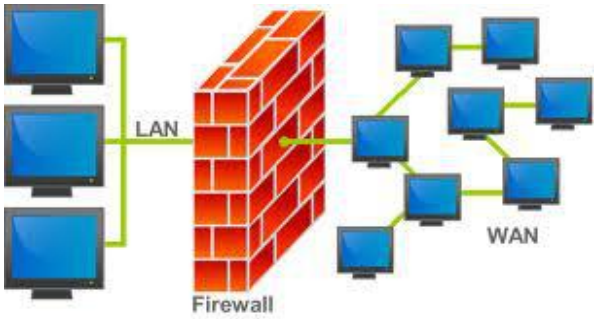


Ilustración 1: Definición Gráfica de Cortafuego

Entre las principales características de la protección cortafuegos [2], destacan las siguientes:

- **Diferentes niveles de protección basados en la ubicación del PC**: Cuando la PC se conecta a una red, la revisión por cortafuegos aplica un nivel de seguridad diferente en función del tipo de red de que se trate. Si se desea modificar el nivel de seguridad asignado inicialmente, se podrá hacer en cualquier momento accediendo a las opciones de configuración del cortafuego.

- **Protección de redes inalámbricas (Wi-Fi)**: Bloquea eficazmente los intentos de intrusión realizadas a través de redes inalámbricas (Wi-Fi). Al producirse una intrusión de este tipo, aparece un aviso emergente que permitirá bloquear la intrusión de inmediato.

- **Accesos a la red y accesos a Internet**: Indica cuáles son los programas instalados en el ordenador que podrán acceder a la red (o a Internet).

- **Protección contra intrusos**: Impide las intrusiones por parte de hackers que intentan acceder al ordenador para realizar en él ciertas acciones.

- **Bloqueos**: El *firewall* permite bloquear los accesos de los programas que se han decidido que no deben acceder a la red local o a Internet. También bloquea los accesos que desde otros ordenadores se realizan para conectarse con programas en el ordenador.

1.1.2 Netfilter

Netfilter es un *framework* disponible en el núcleo GNU/Linux que permite interceptar y manipular paquetes de red. Netfilter es también el nombre que recibe el proyecto que se encarga de ofrecer herramientas libres para cortafuegos basados en GNU/Linux. El componente más popular construido sobre Netfilter es Iptables [3].

1.1.3 Iptables

Iptables es una herramienta de cortafuegos de espacio de usuario[1] que permite no solamente filtrar paquetes, sino también realizar traducción de direcciones de red (NAT)[2] a IPv4[3] y mantener registros (*log)*. Además Iptables permite al administrador, a través de reglas de configuración, definir políticas de seguridad para el tráfico que circula por la red.

Con la aparición del *framework* Netfilter, Iptables mejoró los conceptos asociados a las estructuras de las reglas, se puede citar el ejemplo de Ipchains que incorporó el concepto de cadena e Iptables agregó el concepto de tablas. Otro aspecto a tener en cuenta es las configuraciones de las reglas, que a pesar de ser difíciles de establecer, un usuario con conocimientos avanzados en el tema, puede llegar a crear una sólida barrera protectora.

Antes de la existencia de Iptables, los programas más usados para crear cortafuegos en Linux eran Ipchains en el núcleo Linux 2.2 e Ipfwadm en el núcleo Linux 2.0, que a su vez se basaba en Ipfw de BSD. Tanto Ipchains como Ipfwadm alteran el código de red para poder manipular los paquetes, ya que no existía un *framework* general para el manejo de paquetes hasta la aparición de Netfilter. Iptables mantiene la idea básica introducida en Linux con Ipfwadm: listas de reglas en las que se especifica qué comparar dentro de un paquete y que acciones realizar con ese paquete. Ipchains agrega el concepto de cadenas de reglas (chains) e Iptables extendió este concepto a la idea de tablas: se logró agrupar diferentes funcionalidades en tablas específicas. Adicionalmente, se modificaron los tres puntos en los que se realiza el filtrado en el viaje de un paquete, de modo que los mismos pasen solo por un punto de filtrado.

Mientras que Ipchains e Ipfwadm combinan filtrado de paquetes y NAT (específicamente tres tipos de NAT, llamados enmascaramiento de IP, redireccionamiento de puertos y redirección), Netfilter hace posible por su parte separar las operaciones sobre los paquetes en tres partes: *packet filtering* (filtrado de paquetes), *connection tracking* (seguimiento de conexiones) y *Network Address Translation* (NAT o traducción de direcciones de red). Cada parte se conecta a las herramientas de Netfilter en diferentes puntos para acceder a los paquetes. Los subsistemas de seguimiento de conexiones y NAT son más generales y poderosos que los que realizaban Ipchains e Ipfwadm [4].

Iptables está compuesto por 3 tablas principales, las cuales son:

[1] **Espacio de Usuario:** se refiere a un espacio de aplicación, típicamente en Unix o en sistemas operativos tipo Unix, el cual es externo al núcleo.

[2] **NAT:** es un mecanismo utilizado por *routers* IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles.

[3] **IPv4:** (internet Protocol versión 4 o Protocolo de internet versión 4): es la cuarta versión del protocolo Internet Protocol (IP), y la primera en ser implementada a gran escala.

- **Filter:** esta tabla es usada para filtrar paquetes, por ejemplo con los objetivos ACCEPT (Paquete aceptado), DROP (Paquete rechazado sin notificación) y REJECT (Paquete rechazado y se envía una notificación a través del protocolo ICMP). Cadenas predefinidas: FORWARD, INPUT y OUTPUT.
- **Nat (*Network Address Translation*):** esta tabla permite realizar traducción de direcciones, permitiendo traducir tanto origen como destino. En este caso sólo el primer paquete de la conexión pasa por la tabla, después todos los paquetes de la conexión son tratados de la misma forma. Los objetivos o parámetros válidos en esta tabla son SNAT, DNAT, MASQUERADE (similar a SNAT con IP dinámica) o REDIRECT. Cadenas predefinidas: PREROUTING, POSTROUTING y OUTPUT.
- **SNAT (*Source Network Address Translation o* Traducción de direcciones de red de origen)** para traducir direcciones origen. Su uso común es permitir que una red con IPs privadas pueda acceder a internet a través de una ip pública.
- **DNAT (*Destination Network Address Translation* o traducción de direcciones de red de destino)** está diseñado para convertir direcciones de destino. Este tipo de traducción permite situar distintos servicios en distintos servidores compartiendo una dirección IP.
- **MASQUERADE:** Se utiliza cuando la dirección IP pública que sustituye a la IP de origen es dinámica.
- **REDIRECT:** es utilizado cuando se desea cambiar o redireccionar los puertos.
- **Mangle:** esta tabla está diseñada para manipular los parámetros de los paquetes. Los siguientes parámetros son solo aplicables en esta tabla: TOS[4], TTL[5]. Con ellos se logra modificar los paquetes o marcarlos para luego realizar el control de seguridad. Las cadenas predefinidas en esta tabla son: PREROUTING, POSTROUTING, OUTPUT, INPUT y FORWARD.

Iptables define una cadena como el conjunto de reglas asociados con un determinado tipo de filtro. Iptables también dispone de la posibilidad de que un usuario pueda definir sus propias cadenas y asignarle un nombre. Los nombres de cadenas predefinidas son INPUT, OUTPUT, FORWARD, PREROUTING y POSTROUTING.

- **INPUT** (ENTRADA): Todos los paquetes destinados al cortafuego recorren esta cadena.
- **OUTPUT** (SALIDA): Todos los paquetes creados por el cortafuego recorren esta cadena.
- **FORWARD** (REDIRECCIÓN): Todos los paquetes que meramente pasan por el cortafuego para ser encaminados a sus destinos recorren esta cadena.
- **PREROUTING** (ANTES DE ENRUTADO): No todos los paquetes que arriban desde una interfaz de

[4]**TOS:** Tipo de Servicio que se va a ofrecer (Tipe of Service).
[5]**TTL:** Tiempo de vida de un paquete en la tabla mangle (Time to Live).

red recorren esta cadena, solo lo hacen los de la tabla NAT y Mangle.

- **POSTROUTING** (DESPUÉS DE ENRUTADO): No todos los paquetes que abandonan el cortafuego hacia una interfaz de red recorren esta cadena, solo lo hacen los de la tabla NAT y Mangle.

1.2 Herramientas para la administración del cortafuego.

En este acápite se analizarán varias herramientas para la administración de cortafuegos con el objetivo de seleccionar las características de las mismas que puedan contribuir con el desarrollo de la solución propuesta. Será realizado un análisis enfocado en los aspectos: Interfaz Gráfica, Público objetivo, Principal Funcionalidad y la Licencia que posee.

1.2.1 Shorewall

Es una excelente herramienta para realizar complejas configuraciones para la red, regula los paquetes de entrada y salida de las computadoras que viajan a través de la red. Se define como un cortafuego y a su vez como una puerta de enlace con sus respectivos requisitos de las entradas y salidas de paquetes. A pesar de que Shorewall está basado en Iptables y es muy eficiente comparado con muchas herramientas en cuanto a la configuración detallada de reglas, lo hace a través de archivos de configuración de texto plano y aunque los mismos están bien documentados se necesita un alto nivel de abstracción para realizar una configuración avanzada [5].

- **Público objetivo:** Usuarios expertos.
- **Principal funcionalidad:** la mayor parte de su fuerza reside en su capacidad de trabajar con "zonas", como la DMZ o una zona de red, Nat, filtrado de paquetes.
- **Licencia:** GPLv2 Licencia Pública General versión 2.
- **Interfaz gráfica:** no posee interfaz gráfica.

1.2.2 Zentyal

Zentyal es una alternativa en código abierto a los productos de Microsoft como Windows Small Business Server, Windows Server y Microsoft Exchange, para infraestructura TIC en las pequeñas y medianas empresas (PYMES). El desarrollo de Zentyal se inició en el año 2004 con el nombre de eBox Platform y actualmente es una solución consolidada de reconocido prestigio que integra más de 30 herramientas de código abierto para la administración de sistemas y redes en una sola tecnología. Zentyal está incluido en Ubuntu desde el año 2007, y desde el año 2012 las ediciones comerciales están oficialmente respaldadas por Canonical, empresa encargada del desarrollo y comercialización de Ubuntu. En la actualidad Zentyal tiene más de 1.000 descargas diarias y dispone de una comunidad activa de miles de miembros.

Es un servidor de red integral el cual posee un módulo de cortafuegos muy completo e intuitivo. Permite configurar reglas de Iptables a través de entornos o plantillas predefinidas guiadas por imágenes que orientan a el usuario según sus necesidades, por lo que es un ejemplo a seguir tanto para la configuración de reglas de filtrado de paquete, NAT y tratado de paquetes como para la interacción con su interfaz gráfica. Esto lo hace una herramienta muy potente que puede ser muy útil para tomar ideas en cuanto al diseño y a la configuración de las funcionalidades a agregar al cortafuegos [6].

- **Público objetivo:** Usuarios de bajo, mediano y alto nivel.
- **Principal funcionalidad:** Filtrado de paquetes, NAT, Tráfico de redirección de puertos, Mangle**.**
- **Licencia:** Licencia Publica General de GNU aunque posee algunas licencias privadas**.**
- **Interfaz gráfica:** interfaz gráfica amigable y muy intuitiva.

1.2.3 GUFW (Graphic Uncomplicated Firewall)

El *Firewall* Poco Complicado (UFW, por sus siglas en Inglés) es una adaptación de Iptables y está particularmente bien acondicionado para cortafuegos basados en servidor. UFW proporciona un marco para la gestión de Netfilter, así como una interfaz de línea de comandos para manipular el servidor de seguridad. UFW tiene como objetivo proporcionar una interfaz fácil de usar para las personas no familiarizadas con los conceptos de servidor de seguridad, mientras que al mismo tiempo simplifica complicados comandos de Iptables para facilitarle el trabajo al administrador [7].

- **Público objetivo:** puede ser utilizado por usuarios de bajo nivel.
- **Principal funcionalidad:** gestiona el tráfico entrante y saliente de paquetes, así como configuraciones de rangos de IP y de puertos.
- **Licencia:** Licencia Pública General.
- **Interfaz gráfica:** interfaz gráfica intuitiva.

1.2.4 Webmin

Webmin es una herramienta de configuración de sistemas, accesible vía web para OpenSolaris, GNU/Linux y otros sistemas Unix, la cual permite configurar aspectos internos de muchos sistemas operativos, como gestión de usuarios, cuotas de espacio, servicios, archivos de configuración y apagado del equipo. Está desarrollado en el lenguaje de programación Perl y por defecto se comunica mediante el protocolo TCP a través del puerto 10000.

Mediante cualquier navegador web moderno, se puede configurar Apache, DNS, compartir archivos, manejar

el *firewall* y mucho más. Webmin elimina la necesidad de editar manualmente los archivos de configuración de Unix, permitiendo administrar el firewall desde la consola o remotamente. Esto lo hace una herramienta muy potente que puede ser muy útil para tomar ideas en cuanto al diseño y a la configuración de las funcionalidades a agregar al cortafuegos [8].

- **Tipos de Usuarios:** Usuarios de mediano y alto nivel.
- **Principal funcionalidad:** Filtrado de paquetes, NAT, Mangle**.**
- **Licencia:** Posee licencia de tipo BSD (*Berkeley Software Distribution*).
- **Interfaz gráfica:** interfaz gráfica amigable.

1.2.5 Ventajas obtenidas de las herramientas estudiadas.

Luego del establecido análisis de las herramientas para la administración de cortafuego y partiendo de la necesidad del cliente de aumentar la competitividad técnica con respecto a estas se concluye que:

- Webmin Server y Zentyal son las herramientas con una solución más cercana a las necesidades solicitadas, puesto que presentan unas interfaces muy vistosas e intuitivas donde se puede rápidamente encontrar una manera sencilla de añadir reglas, además poseen una configuración muy exhaustiva de parámetros que le dan ideas al autor de cómo implementar las funcionalidades faltantes al módulo ya anteriormente desarrollado en HMAST.

- Shorewall, es muy difícil de utilizar para un administrador con pocos conocimientos en la seguridad de servicios telemáticos, por lo tanto no se presenta como candidato viable ya que de una manera muy engorrosa, solo apta para los usuarios expertos, configura las reglas. Además, no es adaptable al desarrollo de las funcionalidades propuestas en el módulo ya que la mayor parte de su fuerza reside en su capacidad de trabajar con zonas, pero al tener muchos parámetros configurables abarcadores se toma su trabajo sobre la tabla de NAT como punto de apoyo sobre las funcionalidades a realizar.

- Gufw, como su nombre lo indica, es una herramienta muy intuitiva, por lo tanto las ideas sobre la interfaz, sobre todo la presentación, botones, mensajes y su manera de interactuar con el usuario, se toman de esta herramienta, que de una manera muy sencilla presenta al usuario configuraciones, aunque no una gama muy amplia, de la tabla de filtrado y enrutamiento de paquetes.

1.3 Tecnologías y metodologías utilizadas.

La elección de las tecnologías y la metodología usada para la implementación de las funcionalidades a agregar a la herramienta HMAST fueron las subordinadas a las indicaciones del departamento SIMAYS para

el desarrollo de las soluciones propuestas.

1.3.1 Java (v7.0)

Este lenguaje cuenta con una interfaz orientada a objetos para acceder de un modo portable a cualquier base de datos, promoviendo la portabilidad. Es poseedor de una extensa biblioteca utilitaria y la portabilidad alcanzada es cualitativamente superior a la que se puede obtener con los lenguajes C/C++.

Actualmente, dentro de los lenguajes populares, es uno de los mejores en cuanto a definición debido a que tiene total independencia del implementador del lenguaje y de sus clases auxiliares. Proporciona los tipos de datos primitivos similares a los de C++, solo que carece de punteros y mediante su biblioteca de clases estándar proporciona todas las estructuras contenedoras clásicas. Tiene cuatro niveles de empaquetamiento: variables y funciones, al igual que C++ y otros dos propios de él, denominados: clases y paquetes [9].

- Es un lenguaje compilado, generando ficheros de clases compiladas, las cuales son en realidad, interpretadas por la máquina virtual de java. Siendo esta máquina virtual la que mantiene el control sobre las clases que se estén ejecutando.

- Multiplataforma: El mismo código java que funciona en un sistema operativo, funcionará en cualquier otro que tenga instalada la máquina virtual de java.

- Seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

1.3.2 Framework Spring (v3.1.12)

Es un *framework* de código abierto para el desarrollo de aplicaciones en la plataforma Java. Por su diseño, el *framework* ofrece mucha libertad a los desarrolladores en Java, así como soluciones bien documentadas y fáciles de usar para las prácticas comunes en la industria de la informática [10].

Sus características principales son:

- La inyección de dependencia: El objetivo es lograr un bajo acoplamiento entre los objetos de la aplicación. Con este patrón de diseño, los objetos no crean o buscan sus dependencias con aquellos objetos con los que colabora, sino que éstas son dadas a los mismos. El contenedor (la entidad que coordina cada objeto en el sistema) es el encargado de realizar este trabajo al momento de instanciar el objeto.

- La programación orientada a aspectos: la programación orientada a aspectos (POA) es un paradigma

de programación relativamente reciente que ha logrado un nivel de desarrollo interesante. Su propósito es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos, buscando construir aplicaciones más fáciles de implementar, mantener y extender.

1.3.3 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlos, o sea, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios [11].

1.3.4 HTML

El HTML, acrónimo inglés de *HyperText Markup Language* (lenguaje marcado de hipertexto), es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web. Está compuesto por dos componentes: el contenido, que es el texto que se verá en la pantalla de un ordenador, y las etiquetas y atributos que estructuran el texto de la página web en encabezados, párrafos, listas y enlaces. Gracias a Internet y a los navegadores Web del tipo Internet Explorer, Opera, Firefox o Netscape. El HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos y también de los más fáciles de aprender [12].

1.3.5 Netbeans (v8.0)

NetBeans IDE es un entorno de desarrollo; una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso, fundado por Sun MicroSystems en junio de 2000 y actualmente continúa siendo el patrocinador principal de los proyectos.

La plataforma Netbeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software [13].

1.3.6 Visual Paradigm (v8.0)

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a la construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML.

Visual Paradigm soporta un conjunto de lenguajes, tanto en la generación de código como en la ingeniería inversa. Puede generar código a partir de los modelos y viceversa. Cualquiera de los cambios que se realicen en el código existente puede reflejarse en el modelo. Esta herramienta visual permite construir la aplicación con mayor rapidez, mayor exactitud, mejor trabajo en equipo y fácil de utilizar, además de que aumenta las expectativas mediante la interfaz gráfica [14].

1.3.7 RapidSVN (v1.12)

Es un cliente de interfaz gráfica para la comunicación con servidores Subversion, que se distribuye bajo la Licencia Pública General de GNU, conocida como licencia GPL. Está completamente escrito en C++. Facilita el mantenimiento de las diferentes versiones de ficheros, mediante una interfaz sencilla e intuitiva. Es una herramienta rápida y eficiente que se puede ejecutar en plataformas Windows, Linux, MAC OS X y Solaris [15].

1.3.8 Augeas (v0.4)

Augeas es un editor que analiza los archivos en sus formatos nativos y los transforma en un tipo abstracto de datos llamado árbol, y viceversa. La manipulación de la configuración la realiza mediante este árbol, y los cambios que sean aplicados a él se guardan de forma nativa en dichos archivos de configuración. Para su implementación utiliza unas estructuras llamadas Lenses (en español lentes), que permiten que Augeas pueda entender cualquier archivo de configuración. Es una herramienta modular, construida con el lenguaje de programación C, brindando rapidez y eficiencia desde el punto de vista del uso de recursos computacionales. Además, cuenta con una API de no más de una docena de funcionalidades, que le brindan a los desarrolladores la posibilidad de interactuar con la herramienta. También brinda implementaciones en Python, Ocaml, Ruby, Perl, Haskell, Java, entre otros. Permite además la interacción mediante la consola de Linux, a través de la herramienta Augtool [16].

1.3.9 Metodología SXP

Es una metodología ágil desarrollada en la UCI y surgió de la unión de SCRUM y XP. Con el uso de SCRUM

para la planificación de los proyectos, SXP logra una eficiente gestión de proyectos de software y con la toma de las mejores prácticas de XP, proporciona una mejor calidad en el producto a entregar. Esta metodología utiliza métodos ágiles con el fin de dar solución a muchos de los problemas que afectan el desarrollo y adecuada ejecución de los proyectos, entrega de los mismos, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo. Involucra a los miembros del equipo de desarrollo en las decisiones sobre el proyecto.

SXP está específicamente recomendada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad [17].

Consta de 4 fases:

- **Planificación-Definición:** donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- **Desarrollo:** es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
- **Entrega:** puesta en marcha.
- **Mantenimiento:** donde se realiza el soporte para el cliente.

Cada una de estas fases está compuesta por una serie de actividades que son las que generan los artefactos que quedan incluidos en el nuevo expediente de proyecto, estas actividades están recogidas en el guión de la metodología.

Conclusiones del capítulo

Con el estudio realizado sobre las herramientas de administración de cortafuego existentes se adquirió la base teórica necesaria para lograr los objetivos trazados en el desarrollo de la solución propuesta y se identificó la necesidad de implementarle al módulo funcionalidades que le permitieran administrar la tabla NAT para que fuera más competitivo respecto a sus homólogos.

Además, fueron detalladas cada una de las herramientas, lenguajes de programación, y tecnologías utilizadas en el desarrollo de la primera versión del módulo de cortafuegos, las cuales serán las mismas a utilizar en el presente trabajo; estando guiado el proceso de desarrollo por la metodología SXP que permitió

obtener de forma ágil la documentación necesaria para el mismo.

Capítulo 2: Análisis y Diseño.

En este capítulo se exponen las funcionalidades que debe cumplir el sistema, las cuales están descritas mediante las Historias de Usuario. Además se expone la arquitectura y los patrones de diseño que se utilizarán y la estructura que deben cumplir sus módulos.

2.1 Propuesta de solución.

El módulo permitirá gestionar las reglas para la administración del servicio de cortafuegos al añadirle el soporte para la tabla Nat y para algunas funcionalidades de la tabla Mangle al sistema anteriormente creado, el cual solo soportaba la interacción con la tabla Filter.

- Como funcionalidades complementarias permitirá configurar la aplicación de acuerdo con las preferencias del administrador, donde se podrá escoger la ruta donde se guarda el archivo de configuración, además se podrá escoger la ruta donde se carga.

- Buscando la flexibilidad en la configuración del sistema, la propuesta hereda de la aplicación anteriormente desarrollada una arquitectura en n- capas orientada al dominio como está definida en la concepción HMAST, lo cual simplifica la comprensión y organización del sistema, reduciendo las dependencias de forma que las capas más bajas no sean conscientes de ningún detalle o interfaz de las superiores.

2.2 Herramienta para la migración y administración de servicios telemáticos (HMAST).

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea, así como sus interfaces y la comunicación entre ellos. Debido a esto para desarrollar las nuevas funcionalidades se hereda la estructura del módulo ya implementado con anterioridad, por lo tanto la aplicación se distribuye en 5 paquetes, véase la ilustración 2. Un aspecto a tener en cuenta es que la interacción entre capas se realizará a través de interfaces utilizando inyección de dependencia. *S*e hace una breve descripción de los componentes a continuación.

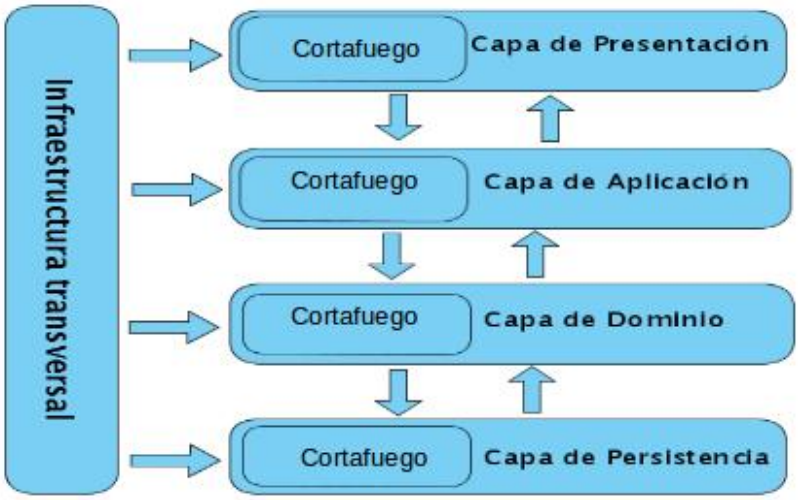


Ilustración 2: Arquitectura del Módulo

Capa de presentación (presentation): la responsabilidad de la capa de presentación es presentar al usuario los conceptos de negocio mediante una interfaz de usuario (*User Interface* UI, en inglés), facilitar la explotación de dichos procesos de negocio, informar sobre la situación de los procesos de negocio e implementación de las reglas de validación de dicha interfaz. Desde el punto de vista del usuario final, esta capa es la "Aplicación" en sí.

Capa de aplicación (application): es una capa delgada, comúnmente solo realizará llamadas a servicios de la capa de dominio, teniendo la posibilidad de que los servicios de aplicación adapten la información que le llega a los requerimientos de los servicios de dominio.

Capa de dominio (domain): constituye el hilo conductor de la aplicación, sus componentes solo dependen de la Capa de infraestructura transversal los cuales están totalmente desacoplados. Esta capa 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. Sus componentes no están ligados a tecnologías específicas.

Capa de persistencia (persistence): la capa 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.

Los repositorios se diferencian de las clases de persistencia de datos tradicionales en que los primeros

almacenan la información que se está procesando en memoria, además solo persisten los datos realmente cuando se les ordena explícitamente.

Capa infraestructura transversal (infrastructureCrosscutting): la responsabilidad de la capa de infraestructura transversal está dada a promover la reutilización de código.

Estilos arquitectónicos: En el diseño de HMAST quedó definida la utilización de una arquitectura **N-Capas orientada al Dominio**, este estilo tiene como objetivo estructurar de forma concreta la complejidad de una aplicación empresarial basada en las diferentes capas de la arquitectura . Teniendo en cuenta que el módulo ya desarrollado está integrado a dicha herramienta y persiguiendo el propósito de mantener una homogeneidad entre los componentes del sistema, se decide incluir las funcionalidades a implementar para el módulo de administración del servicio Iptables en cada una de las capas, considerando las responsabilidades que cada una establezca [18].

2.3 Roles

Es responsabilidad de los miembros del equipo, cumplir con el rol asignado para que se obtenga un producto con la calidad requerida y en el tiempo establecido.

Tabla 1 Roles

Rol	Responsabilidad	Nombre y Apellidos
Gerente	Es el responsable de tomar las decisiones finales, participa en la definición de objetivos y requerimientos. Tiene la responsabilidad de controlar el progreso del software.	Javier Ramón Vila
Cliente	Asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración, participa en la concepción inicial del sistema. Contribuye a definir las historias de usuario y los casos de prueba de aceptación.	Departamento de Sistemas Integrales de Migración, Asesoría y Soporte.
Miembros del Equipo		
Programador	Define las tareas de ingeniería. Produce el código del sistema. Selecciona el estándar de programación. Confecciona los Manuales de Usuario y de desarrollo.	Javier Ramón Vila
Analista	Escribe la concepción del sistema y las historias de usuario. Crea el Modelo de historia de usuario del negocio y la LRP. Asigna la prioridad a las Historias de Usuario.	Javier Ramón Vila

Diseñador de interfaz	Encargado del diseño del sistema, así como el de los prototipos de interfaces, supervisa el proceso de construcción.	Javier Ramón Vila
Probador	Escribe los casos de prueba de aceptación. Ejecuta las pruebas, es responsable de las herramientas de soporte para pruebas.	Javier Ramón Vila
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas, además aportan ideas y experiencias para el beneficio del sistema en desarrollo.	Ing. Jailen García Gonzalez

2.4 Artefactos generados en el desarrollo

A continuación se describen los artefactos generados durante el desarrollo: Lista de Reserva del Producto, Historias de Usuario, Modelo de Dominio y Diagrama de Paquetes.

2.4.1 Requerimientos funcionales

Los requerimientos son aquellas 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 [19]. El propósito de su gestión es establecer un entendimiento común entre el usuario y el desarrollador de software. Se clasifican en requerimientos funcionales y no funcionales. Los funcionales son la determinación exacta de qué debe ser capaz de hacer el sistema, estas se corresponden con opciones que ejecutará el software, operaciones realizadas de forma oculta o condiciones extremas a determinar por el sistema.

Durante la captura de los requerimientos se confecciona la Lista de Reserva del Producto (LRP), en la cual se definen las funcionalidades que tendrá el producto en forma de requisitos técnicos y de negocio. La LRP es una lista priorizada y garantiza la organización de los requisitos funcionales y no funcionales, a partir de la prioridad que tengan para la implementación del sistema. Para el desarrollo de esta investigación se planifica una iteración, siendo identificados 14 requisitos de prioridad muy alta y alta. A continuación se muestra la tabla de requisitos contenida en la LRP.

Tabla 2 Lista de Reserva del Producto

Asignado a	**Ítem**	**Descripción**	**Estimación**	**Estimado por**
Prioridad: Muy Alta				

Programador	1	Añadir regla NAT.	0.3	Javier Ramón Vila
Programador	2	Eliminar regla NAT.	0.3	Javier Ramón Vila
Programador	3	Modificar reglas NAT.	0.3	Javier Ramón Vila
Programador	4	Mostrar reglas NAT.	0.3	Javier Ramón Vila
Programador	5	Salvar reglas NAT en un directorio.	0.3	Javier Ramón Vila
Programador	6	Restaurar reglas NAT desde un directorio.	0.3	Javier Ramón Vila
Programador	7	Añadir una regla Mangle.	0.3	Javier Ramón Vila
Programador	8	Eliminar una regla Mangle.	0.3	Javier Ramón Vila
Programador	9	Modificar una regla Mangle.	0.3	Javier Ramón Vila
Programador	10	Mostrar las reglas Mangle.	0.3	Javier Ramón Vila
Programador	11	Salvar las reglas Mangle en un directorio.	0.3	Javier Ramón Vila
Programador	12	Restaurar las reglas Mangles desde un directorio.	0.3	Javier Ramón Vila
Alta				
Programador	13	Instalar el servicio de cortafuego.	0.3	Javier Ramón Vila
Programador	14	Desinstalar el servicio de cortafuego.	0.3	Javier Ramón Vila
Requisitos No Funcionales				

	1	Crear interfaz sencilla e intuitiva desde la cual se acceda a todas las funcionalidades del sistema.		
	2	Usar como lenguaje de programación Java usando el framework Spring.		
	3	La herramienta de desarrollo a utilizar será Neatbeans en su versión 8.0.		
	4	La aplicación se ejecutará en sistemas operativos libres.		
	5	El sistema contará con una ayuda que explicará cómo trabajar con el sistema.		
	6	Usar como herramienta de modelado el Visual Paradigm 8.0.		

2.4.2 Historias de Usuario (HU)

Las HU constituyen la técnica utilizada en SXP para especificar los requisitos del software. Se recogen los 14 requisitos funcionales antes expuestos en 4 HU. Las HU guían la construcción de las Pruebas de Aceptación y son utilizadas para estimar tiempos de desarrollo; en este sentido, proveen los detalles suficientes para hacer una estimación razonable del tiempo que llevará implementarlas siendo detallada a través de la comunicación con el cliente. A continuación se exponen las tablas de las HU.

Tabla 3 HU Gestionar estado del firewall

Historia de Usuario	
Número: HMAST_Firewall_1	**Nombre Historia de Usuario:** Gestionar estado del firewall.
Modificación de Historia de Usuario: 0	

Usuario: Javier Ramón Vila	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 3.7 semanas
Riesgo en desarrollo: Alta	**Puntos Reales:** 3.7 semanas
Descripción: Permitirá gestionar el servicio del firewall; se refiere al proceso de instalación y desinstalación.	
Observación: **Instalar el servicio cortafuego:** El usuario tiene la posibilidad de poder instalar el servicio de cortafuegos en caso de no estar instalado en el servidor, para esto solo es necesario seleccionar la opción correspondiente a dicha funcionalidad. El comando a ejecutar en la consola sería "*apt-get install iptables*". **Desinstalar el servicio cortafuego:** El usuario tiene la posibilidad de poder desinstalar el servicio de cortafuegos en caso de estar instalado en el servidor, para esto solo es necesario seleccionar la opción correspondiente con dicha funcionalidad. El comando a ejecutar en la consola sería "*apt-get purge iptables*".	

Tabla 4 HU Gestionar reglas NAT

Historia de Usuario	
Número: HMAST_Firewall_2	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Modificación de Historia de Usuario: 0	
Usuario: Javier Ramón Vila	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 5.2 semanas
Riesgo en desarrollo: Muy Alta	**Puntos Reales:** 5.2 semanas
Descripción: Permitirá gestionar las reglas Nat (traducción de direcciones); se refiere al tráfico de los paquetes en la cadena de Prerouting, Postrouting y Output, donde se podrá añadir, eliminar, modificar y mostrar las reglas correspondientes a esta tabla.	

Observación:

Adicionar regla de traducción de red: La adición de reglas de traducción de red se realiza en dependencia de la cadena que se quiera filtrar, siendo diferentes las mismas en cada caso.

- **Adicionar reglas la cadena de Prerouting:** se mostrará un asistente de configuración, en el cual el usuario introduce los parámetros para adicionar dicha regla NAT. Estos parámetros son:

  Tipo de regla: el usuario selecciona el tipo de regla que desea, en este caso se refiere al destino que tendrá el paquete si coincide con los parámetros de configuración. Es obligatorio seleccionar una de estas opciones.

  - **DNAT:** Traducción de la dirección de red de destino. Esta opción consiste en cambiar la dirección de destino ya predefinida por una nueva dirección de destino. Para esto al seleccionar la opción DNAT, el usuario podrá introducir una nueva dirección o rango de IP. En este caso el usuario puede introducir una rango de IP separado por **"-"** o una dirección IP. Para este caso se tiene en cuenta que el IP será válido si solo se adicionan cuatro números separados por **"."** y si cada número es menor o igual que 255. Ejemplo de una regla de este tipo seria: "iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to-destination 192.168.1.80".
  - **Redireccionar:** Esta opción permite al usuario cambiar la dirección de un paquete, es similar a la opción DNAT, excepto que en este caso solo se pueden redireccionar puertos. En este caso el usuario solo podrá introducir un puerto simple. Ejemplo de una regla de este tipo sería: "iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8888".

  Dirección de origen: en este caso el usuario tiene tres opciones para definir su dirección de origen o en el mejor de los casos, puede seleccionar las tres opciones, las cuales son:

  - **Interfaz de entrada:** Interfaz de red de entrada del servidor (eth0, eth1). Este campo solo acepta las interfaces que se definan en el servidor, es decir, en caso de introducir una interfaz que no sea las que posee el servidor, se mostrará un mensaje de error.
  - **Dirección Ip/máscara:** Dirección IP y máscara del dispositivo que originó la conexión**.** La versión del IP usada es la IPv4. En caso de la máscara, solo el usuario tendrá la posibilidad de seleccionar el número.

- **Puerto origen:** El usuario tiene la posibilidad de introducir un puerto simple o un rango de puertos por **":"**. En este caso solo son válidos los puertos que se encuentran en el rango de 0-65535, no se permiten letras en este campo.
- **Mac:** Dirección MAC del dispositivo que originó la conexión. En este campo el usuario solo podrá adicionar seis pares de letra y dígitos separados por **":"**, las letras solo pueden ser de la (a – f).

Dirección de destino: en este caso el usuario tiene una opción para definir su dirección de destino la cual es:

- **Dirección IP/Máscara:** el mismo caso del campo (Dirección IP/Máscara) de la opción Dirección de origen.
- **Protocolo:** el usuario puede seleccionar el protocolo que se definió ya en el sistema. En este campo solo son válidos los protocolos **TCP**, **UDP**, **ICMP** y **ALL**, para especificar todos los protocolos.
- **Puerto destino:** En este caso solo son válidos los puertos que se encuentran en el rango de 0-65535, no se permiten letras en este campo.

- **Adicionar regla en la cadena de Postrouting:** se mostrará un asistente de configuración, en el cual el usuario introduce los parámetros para adicionar dicha regla NAT. Estos parámetros son:

  Tipo de regla: el usuario selecciona el tipo de regla que desea, en este caso se refiere al destino que tendrá el paquete si coincide con los parámetros de configuración. Es obligatorio seleccionar una de estas opciones.

  - **SNAT:** Traducción de la dirección de red de origen. Esta opción consiste en cambiar la dirección de origen ya predefinida, por una nueva dirección de origen. Para esto al seleccionar la opción SNAT, el usuario podrá introducir una nueva dirección IP. En este caso el usuario puede introducir un rango de IP separado por **"-"** o una dirección IP. Para este caso se tiene en cuenta que el IP será válido si adicionan cuatro número separados por **"."** y si cada número es menor o igual que 255. Ejemplo de una regla de este tipo sería: "iptables -t nat -A POSTROUTING -o eth1 -j SNAT –to-source 192.168.1.89".

- **Enmascarar:** Esta opción es similar a **SNAT**, solo que esta es utilizada, cuando no se conoce exactamente la dirección de origen. Esto se pone en evidencia cuando se tiene direcciones dinámicas es decir servicio DHCP y por lo tanto no necesita que se le especifique una nueva dirección. No posee Interfaz de entrada. Ejemplo de una regla de este tipo sería:"iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE".

Interfaz de salida: Interfaz de red de salida del servidor (eth0, eth1). Este campo solo acepta las interfaces que se definan en el servidor, es decir, en caso de introducir una interfaz que no sea las que posee el servidor, se mostrará un mensaje de error.

Dirección de origen: en este caso el usuario tiene tres opciones para definir su dirección de origen o en el mejor de los casos, puede seleccionar las tres opciones, las cuales son:

- **Dirección Ip/máscara:** Dirección IP y máscara del dispositivo que originó la conexión**.** La versión del IP usada es la IPv4. En caso de la máscara, solo el usuario tendrá la posibilidad de seleccionar el número.
- **Puerto origen:** El usuario tiene la posibilidad de introducir un puerto simple o un rangos de puertos por **":"**. En este caso solo son válidos los puertos que se encuentran en el rango de 0-65535, no se permiten letras en este campo.

Dirección de destino: en este caso el usuario tiene una opción para definir su dirección de destino la cual es:

- **Dirección IP/Máscara:** el mismo caso del campo (Dirección IP/Máscara) de la opción Dirección de origen.
- **Protocolo:** el usuario puede seleccionar el protocolo que se definió ya en el sistema. En este campo solo son válidos los protocolos **TCP**, **UDP**, **ICMP** y **ALL**, para especificar todos los protocolos.
- **Puerto destino:** El usuario tiene la posibilidad de introducir un puerto simple o un rangos de puertos por **":"**. En este caso solo son válidos los puertos que se encuentran en el rango de 0-65535, no se permiten letras en este campo.

- **Adicionar reglas la cadena de Output:** se mostrará un asistente de configuración, en el cual el usuario introduce los parámetros para adicionar dicha regla NAT. Estos parámetros son:

Tipo de regla: el usuario selecciona el tipo de regla que desea, en este caso se refiere al destino que tendrá el paquete si coincide con los parámetros de configuración. Es obligatorio seleccionar una de estas opciones.

- **DNAT:** Traducción de la dirección de red de destino. Esta opción consiste en cambiar la dirección de destino ya predefinida por una nueva dirección de destino. Para esto al seleccionar la opción DNAT, el usuario podrá introducir una nueva dirección o rango de IP. En este caso el usuario puede introducir una rango de IP separado por **"-"** o una dirección IP. Para este caso se tiene en cuenta que el IP será válido si solo se adicionan cuatro números separados por **"."** y si cada número es menor o igual que 255. Ejemplo de una regla de este tipo seria: "iptables -t nat -A OUTPUT -i eth0 -p tcp --dport 80 -j DNAT --to-destination 192.168.1.80".
- **Redireccionar:** Esta opción permite al usuario cambiar la dirección de un paquete, es similar a la opción DNAT, excepto que en este caso solo se pueden redireccionar puertos. En este caso el usuario solo podrá introducir un puerto. Ejemplo de una regla de este tipo sería: "iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 8888".

Dirección de origen: en este caso el usuario tiene tres opciones para definir su dirección de origen o en el mejor de los casos, puede seleccionar las tres opciones, las cuales son:

- **Interfaz de entrada:** Interfaz de red de entrada del servidor (eth0, eth1). Este campo solo acepta las interfaces que se definan en el servidor, es decir, en caso de introducir una interfaz que no sea las que posee el servidor, se mostrará un mensaje de error.
- **Dirección Ip/máscara:** Dirección IP y máscara del dispositivo que originó la conexión**.** La versión del IP usada es la IPv4. En caso de la máscara, solo el usuario tendrá la posibilidad de seleccionar el número.
- **Mac:** Dirección MAC del dispositivo que originó la conexión. En este campo el usuario solo podrá adicionar seis pares de letra y dígitos separados por **":"**, las letras solo pueden ser de la (a – f).

Dirección de destino: en este caso el usuario tiene una opción para definir su dirección de destino la cual es:

- **Dirección IP/Máscara:** el mismo caso del campo (Dirección IP/Máscara) de la opción Dirección de origen.
- **Protocolo:** el usuario puede seleccionar el protocolo que se definió ya en el sistema. En este campo solo son válidos los protocolos **TCP**, **UDP**, **ICMP** y **ALL**, para especificar todos los protocolos.
- **Puerto origen:** El usuario tiene la posibilidad de introducir un puerto simple o un rangos de puertos por **":"**. En este caso solo son válidos los puertos que se encuentran en el rango de 0-65535, no se permiten letras en este campo.
- **Puerto destino:** En este caso solo son válidos los puertos que se encuentran en el rango de 0-65535, no se permiten letras en este campo.

Listar reglas de traducción de red: En la vista se mostrará el campo correspondiente a la lista donde se mostrarán los datos previamente adicionados en forma de tabla, mostrando exactamente los mismos datos que el usuario introduce en el asistente.

Modificar reglas de traducción de red: El usuario selecciona en la lista, la regla que será modificada, y luego de esto selecciona el botón correspondiente a dicha funcionalidad. Seguidamente se muestra el asistente de configuración correspondiente con el escenario perteneciente a la regla seleccionada.

Eliminar regla de traducción de red: El usuario marca en la lista las reglas que desea eliminar y pasado esto seleccionar el botón correspondiente a dicha funcionalidad.

Prototipos:

A continuación se muestra el prototipo de Listar Regla, los demás se encuentran en los anexos.

Ilustración 3: Listar Reglas Nat

Tabla 5 HU Gestionar reglas Mangle

Historia de Usuario	
Número: HMAST_Firewall_3	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Modificación de Historia de Usuario: 0	
Usuario: Javier Ramón Vila	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 5.2 semanas
Riesgo en desarrollo: Muy Alta	**Puntos Reales:** 5.2 semanas

Descripción: Permitirá gestionar las reglas Mangle; se refiere al modificación de los paquetes en la cadena de Prerouting, Postrouting, Input, Output y Forward, donde se podrá añadir, eliminar, modificar y mostrar conjunto de reglas.

Observación:

Adicionar reglas Mangle: para adicionar una cadena se le pasan los siguientes parámetros:

- **Cadena:** puede ser una de las siguientes opciones: Prerouting, Postrouting, Input, Output y Fordward.
- **TTL:** Esta opción permite al usuario cambiar el tiempo de vida que tiene el paquete (TTL) el cual se encuentra en la cabecera del paquete. Para cambiarlo se debe entrar el nuevo valor para el TTL y seleccionar la opción deseada que puede ser una de las siguientes:
    - **1er caso**: comparar con los paquetes que tengan mayor TTL.
    - **2do caso**: comparar con los paquetes que tengan menor TTL.
    - **3er caso**: comparar con los paquetes que tengan igual TTL.
- **MARK:** Esta opción permite marcar (MARK) el paquete con un número para su fácil tratamiento posterior. En este caso el usuario entra un número cualquiera mientras este entre el rango de 1 a 100.
- **TOS:** Esta opción permite comparar el tipo de servicio (TOS) que se encuentra en la cabecera del IP con la entrada por el usuario. Para realizar esta opción se selecciona una de las siguientes opciones:
    - #0: Servicio Normal.
    - #2: Selecciona el camino de menor costo monetario.
    - #4: Selecciona el camino de mayor confianza.
    - #8: Selecciona el camino con el rendimiento máximo.
    - #16: Selecciona el camino con el menor retraso.

Listar reglas mangle: En la vista se mostrará el campo correspondiente a la lista donde se mostrarán los datos previamente adicionados en forma de tabla, mostrando exactamente los mismos datos que el usuario introduce en el asistente.

Modificar reglas mangle: El usuario selecciona en la lista, la regla que será modificada, y luego de esto selecciona el botón correspondiente a dicha funcionalidad. Seguidamente se muestra el asistente de configuración correspondiente con el escenario perteneciente a la regla seleccionada.

Eliminar regla mangle: El usuario marca en la lista las reglas que desea eliminar y pasado esto selecciona el botón correspondiente a dicha funcionalidad.

Prototipos:

A continuación se muestra el prototipo de Listar Regla, los demás se encuentran en los anexos.

Ilustración 4: Listar Regla Mangle

Tabla 6 HU Acciones Adicionales sobre reglas

Historia de Usuario	
Número: HMAST_Firewall_4	**Nombre Historia de Usuario:** Acciones adicionales sobre las reglas.
Modificación de Historia de Usuario: 0	

Usuario: Javier Ramón Vila	**Iteración Asignada:** 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 0,5 semanas
Riesgo en desarrollo: Muy Alta	**Puntos Reales:** 0,5 semanas
Descripción: Permitirá realizar acciones adicionales sobre la tabla de las reglas como son la realización de salvas y la restauración de salvas.	
Observación: **Salvar reglas:** Al usuario se le mostrará un botón donde podrá salvar las reglas en el servidor de destino en /etc/iptables-save, además se salvan en la carpeta local de HMAST en la máquina local si se desea. **Restaurar reglas:** Al usuario después de entrar al módulo de Iptables se le mostrará un listado con todas las reglas restauradas.	

2.4.3 Diagrama de Paquetes

El Diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas reflejando las dependencias entre las mismas. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. A continuación se desglosa el diagrama de paquetes por capas para un mejor entendimiento.

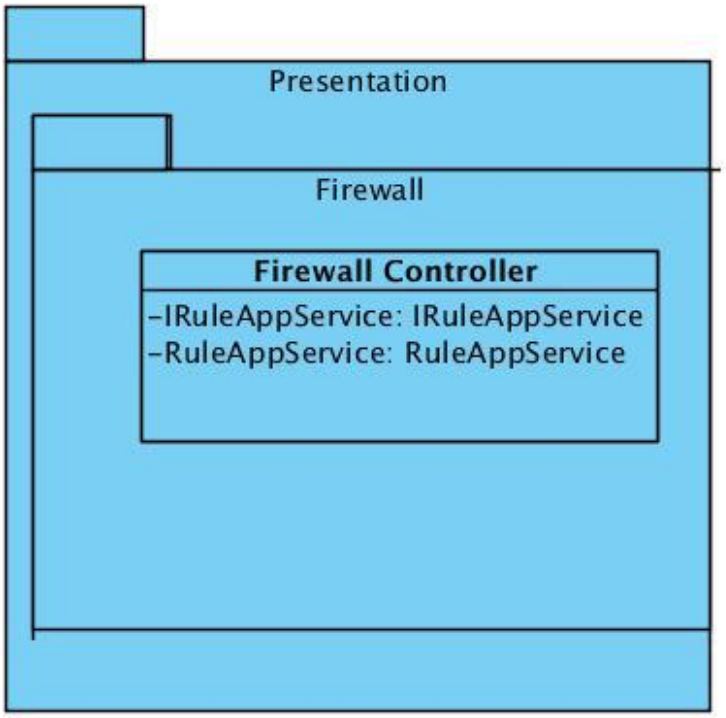


Ilustración 5: Capa de Presentación del Diagrama de Paquete

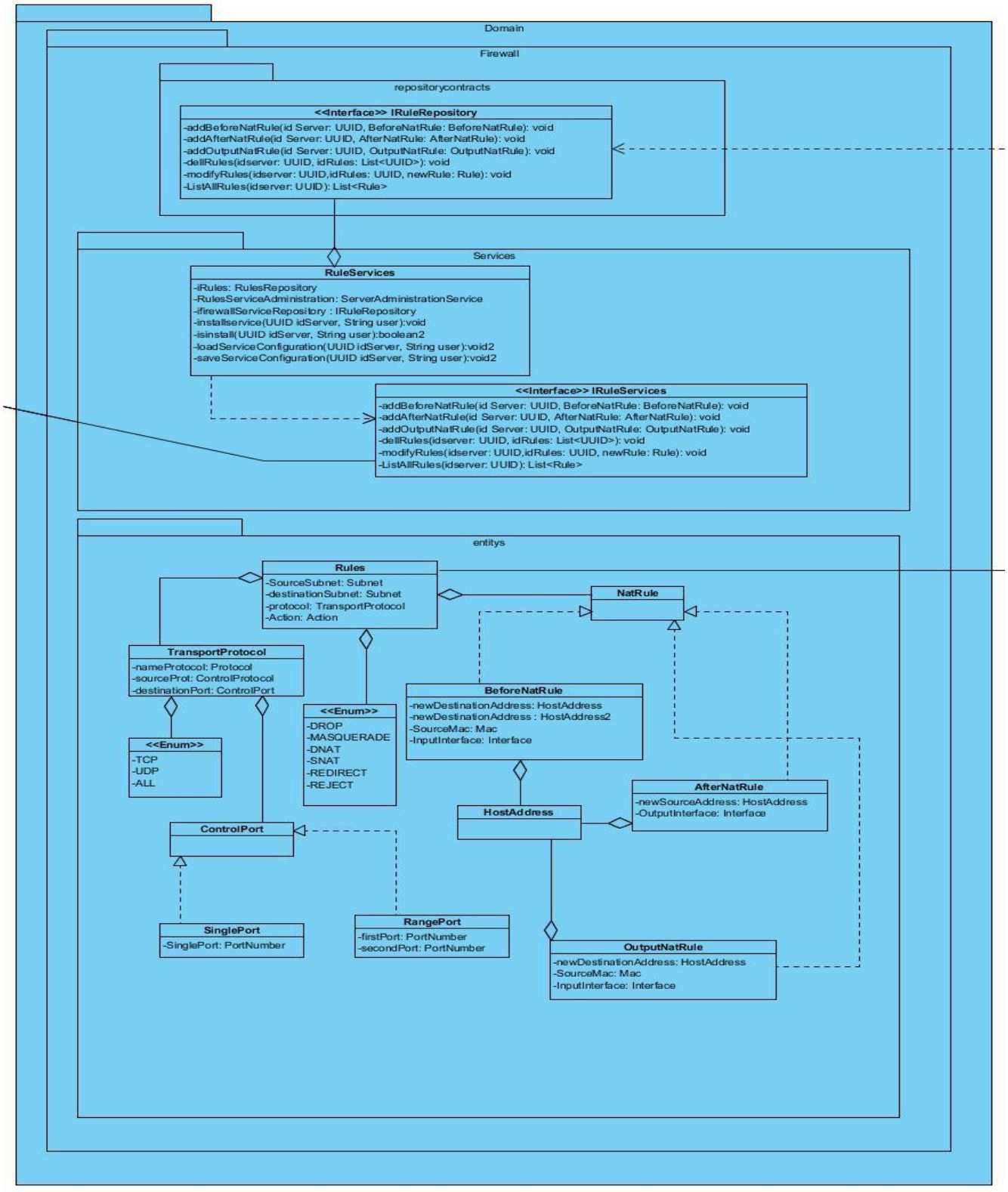


Ilustración 6: Capa de Dominio del Diagrama de Paquetes.

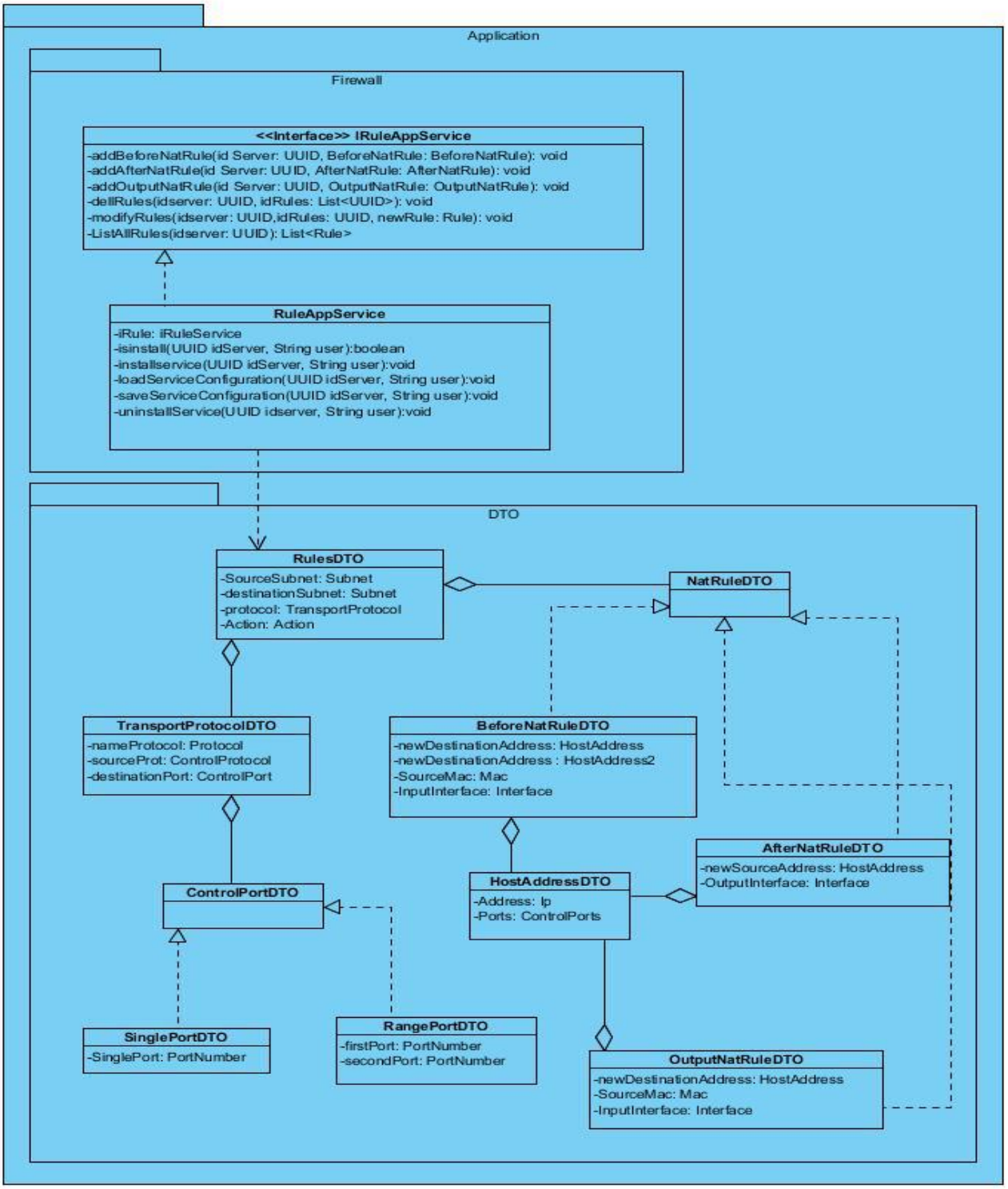


Ilustración 7: Capa de Aplicación del Diagrama de Paquetes

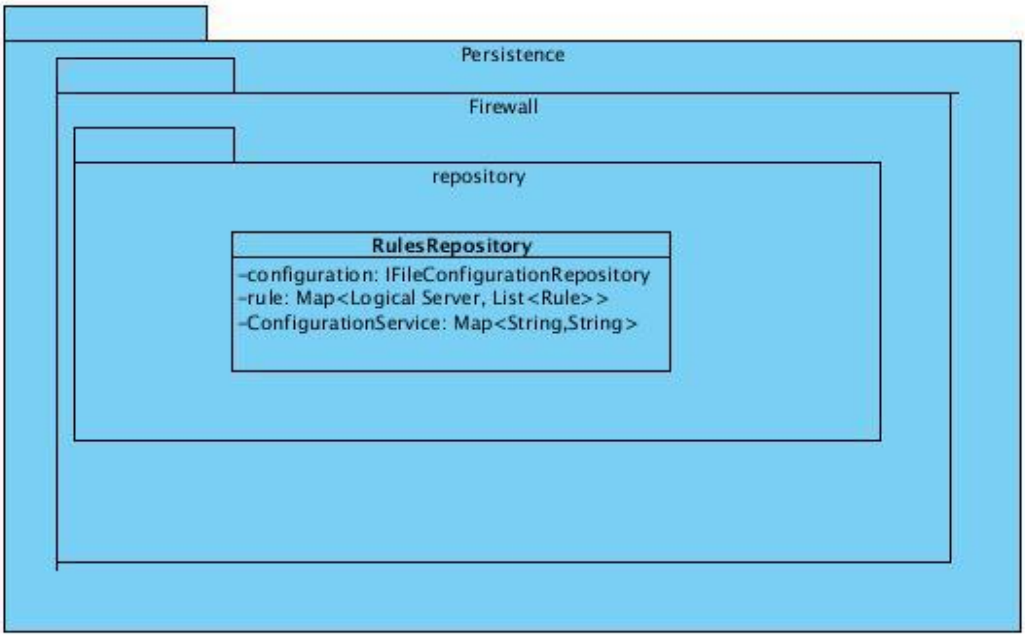


Ilustración 8: Capa de Persistencia del Diagrama de Paquetes

Los diagramas expuestos anteriormente representan la distribución de las clases para la implementación de la historia de usuario gestionar tablas Nat. En la capa de aplicación (*application*) se encuentra el paquete firewall que presenta la interfaz IRuleAppService. Esta contiene métodos que permiten el acceso desde la capa de presentación, los cuales son implementados por la clase RuleAppService. Esta última es responsable de adaptar la información que le llega desde la interfaz de usuario a los requerimientos de los servicios de dominio, estos servicios son accedidos a través del atributo RuleServices contenido en la propia clase.

Esta capa además contiene el paquete DTO (Data Object Transfer), el mismo almacena clases que representan la información que llega desde la capa de presentación. La capa de dominio (*domain*) contiene los subpaquetes Entidades (*entitys*), Servicio (*service*) y Contratos de repositorios (*repositoryscontrats*). En Entidades se encuentra las clases Rule y Nat, quienes representan a las reglas de las tablas Nat a configurar. El subpaquete Servicio contiene la clase IRuleServices que define métodos que son accedidos desde la capa de aplicación, implementados por la clase RuleService, es la encargada de realizar las validaciones de los datos antes de realizar las operaciones en el repositorio. El acceso al repositorio se realiza a través del atributo iRules encontrado en esta clase. En el subpaquete Contratos de repositorios se encuentra la clase IRuleRepository, la cual tiene definidos los métodos que son accedidos desde el subpaquete Servicio y son implementados por la clase RulesRepository encontrada en la capa de persistencia de los datos (persistens).

Los diagramas de paquetes que representan el resto de las HU presentan el mismo flujo de comunicación entre clases que se describe anteriormente independientemente de las funcionalidades que estas tengan.

2.4.4 Modelo de Dominio

El Modelo de Dominio se representa como uno o un conjunto de diagramas de clases y engloba los conceptos de la propia realidad física en la que ejerce el sistema. Con la construcción del modelo de dominio se persigue plasmar el conocimiento adquirido. Es utilizado como una útil herramienta para comprender el sector al cual el sistema va a servir. Una vez completado este paso es probable que se desarrolle una amplia visión del sistema y el entorno en que este será desplegado posteriormente [20].

A continuación se muestra el modelo de dominio del módulo de HMAST para administrar el servicio de cortafuegos:

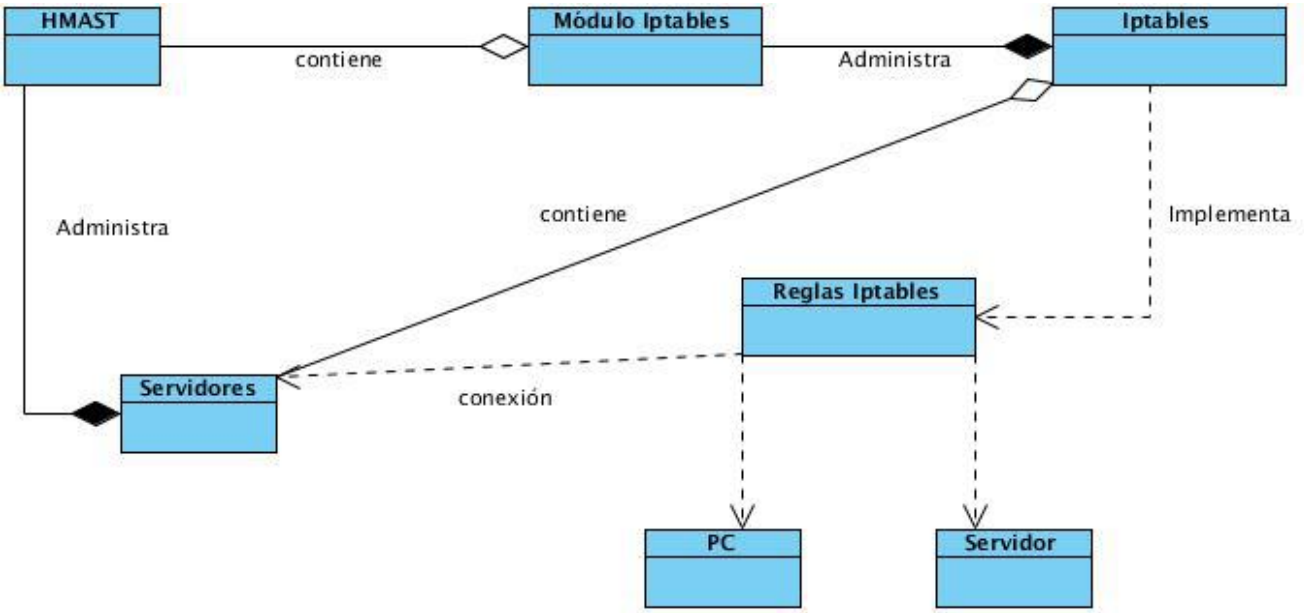


Ilustración 9: Modelo de Dominio

Como se muestra en la figura anterior un usuario administra HMAST que a su vez contará con el módulo de administración de cortafuegos, el mismo estará compuesto por el servicio de Iptables, los servidores que administra HMAST, las reglas de Iptables, las computadoras y servidores que se administran. Seguidamente se detallan dichas clases:

- **Iptables**: representa el servicio que administra el módulo como tal.

- **Reglas Iptables:** representa las reglas generadas por el servicio administrado, las cuales permiten o deniegan el acceso desde las PC o Servidores administrados hacia o desde los Servidores que administra directamente HMAST.

2.5 Patrones de diseño

Un patrón es una pareja de problema/solución y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y después describe la esencia de la solución a dicho problema, de tal forma que puedas usar dicha solución reiteradamente de distintas maneras [21].

2.5.1 Patrones Grasp

Patrones generales de software para asignación de responsabilidades (*General Responsibility Assignment Software Patterns* en inglés), no son considerados solo patrones propiamente dichos, sino buenas prácticas de aplicación recomendable en el diseño de software [22].

- **Experto:** A modo de ejemplo, este patrón indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). Con la utilización de este patrón, se hizo posible definir dónde colocar, en cada clase, las funcionalidades que necesitan de esa información, dicha clase sería el experto en información. En la aplicación se necesita adicionar, eliminar y modificar un objeto de tipo NatRule que define la estructura de ese tipo de regla. En este caso, la clase Firewall que contiene una lista de NatRule es la experta en información y por tanto, la encargada de realizar las operaciones anteriormente mencionadas sobre la lista.

- **Creador:** El patrón creador ayuda a identificar quien debe ser el responsable de la creación de instancias o de nuevos objetos o clases. La instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, o usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase. Para las relaciones entre clase del desarrollo de las funcionalidades del módulo propuesto, se utiliza el patrón creador teniendo en cuenta que el mismo indica que clase tendrá un objeto o instancia de otra clase. Este patrón se utilizó en la Capa de Aplicación cuando para la conversión de clases de tipo DTO a entidad, se crean instancias de las entidades, debido a que las clases DTO contienen los datos de inicialización de estas.

- **Bajo acoplamiento:** Se basa en la idea de tener las clases lo menos entrelazadas posible. De tal

forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. En el desarrollo de las funcionalidades del módulo propuesto el bajo acoplamiento se logra a través de las inyecciones de dependencia y mediante el uso de interfaces. 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. También las inyecciones de dependencia proporcionadas por el framework Spring, permiten el uso de este patrón en el sistema.

- **Alta cohesión:** este patrón se basa en que la información que almacena una clase debe ser coherente y debe estar en la medida de lo posible relacionada con la clase. Se hizo necesario la utilización de este patrón en el módulo 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.

2.5.2 Patrones GoF

Los patrones GoF cuentan con tres clasificaciones según su propósito: Creación, Estructurales y de Comportamiento. En el desarrollo de las funcionalidades del módulo propuesto se utilizó el patrón solitario (*singlenton*) el cual permite hacer una única instancia para una clase, garantizando el acceso global a la misma con el mismo objeto en varios momentos y lugares [23]. Es utilizado debido a la necesidad de trabajar con el mismo objeto en distintos momentos, se puede evidenciar en la conexión a un servidor ya que con una única instancia de este objeto se realizan todas las operaciones sobre el servidor.

Conclusiones del capítulo

Una vez terminada la fase de Planeación-Definición establecida por la metodología SXP queda propuesto el desarrollo de un módulo para HMAST con 14 requisitos funcionales descritos en 4 HU. Además, la descripción de las HU permitió un mejor entendimiento entre el programador y el cliente, llegando a un acuerdo sobre las capacidades y cualidades con las que el software debe cumplir. También, la correcta definición de los patrones de arquitectura y los de diseño, permitieron que se obtuviera una base del sistema capaz de soportar posteriores cambios en los requerimientos.

El estudio realizado en este capítulo ha facilitado un entendimiento de la dinámica y estructura de la aplicación, contribuyendo a una mejor comprensión del problema que las funcionalidades a desarrollar deben resolver y estableciendo las bases para la fase de implementación.

Capítulo 3: Implementación y pruebas del sistema.

En el presente capítulo se refleja el flujo de actividades que conforman la implementación y pruebas de la solución propuesta, además se describe los estándares de código utilizados.

3.1 Estándar de código.

Un estándar de codificación comprende todos los aspectos de la generación de código. Los estándares de codificación permiten una mejor integración entre las líneas de producción y establece pautas que conllevan a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su capacidad de mantenimiento a lo largo del tiempo.

Nomenclatura de las clases.

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación **PascalCasing**. Con solo leerlo se reconoce el propósito de la misma, pues cada palabra comienza con letra mayúscula.

Ejemplo: BeforeNatRule

Nomenclatura de 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 empleará notación **CamelCasing**, y con solo leerlo se reconoce el propósito de la misma.

Ejemplo: addRule

Estándar para nombrar las variables.

El nombre de las variables se escribe con primera palabra en minúscula, si es un nombre compuesto se utilizará notación **CamelCasing**.

Ejemplo: newDestinationAddress

Nomenclatura de los componentes:

Todos los paquetes comienzan con cu.uci.hmast.xxx.yyy.zzz

xxx → Capa donde se encuentra (presentación, aplicación, dominio, persistencia).

yyy →Nombre del módulo (postfix, dhcp3, mmias, etc).

zzz →Elementos que pueden contener los componentes verticales (Entitys, repositorys, etc).

Ejemplo: cu.uci.hmast.domain.Iptables.service

Nomenclatura de los comentarios.

Los comentarios deben ser claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. En caso de ser una función complicada se comentaría dentro de la misma para lograr una mejor comprensión del código.

Ejemplo:

/** Permite adicionar una regla al servidor que se corresponda con el identificador que se pasa por parámetro

* @param identificador de un servidor

* @param una regla

* @throws JSchException

* @throws IOException

* @throws EDuplicateObject si la regla ya existe

* @throws ELogicalServerNotExist

* @throws EInvalidRule si lo que se recibe por parámetro no es una regla

* @throws EPaperInvalid si el archivo es incorrecto.

* @throws EInvalidObject si el servidor que se recibe por parámetro no existe

* @throws EInvalidPath si la ruta no es valida

* @throws EInvalidSymbol si el símbolo recibido no se corresponde con los permitidos en las reglas

* @throws EExistingItem

*/

void addRule (UUID server, RuleDTO rule) throws JSchException, IOException, InterruptedException, EDuplicateObject, ELogicalServerNotExist, EInvalidRule, EPaperInvalid,

EInvalidObject, EInvalidPath, EInvalidSymbol, EExistingItem;

3.2 Tareas de Ingeniería

Las tareas de ingeniería son las que guían el trabajo de las iteraciones, las cuales se realizan para especificar las acciones llevadas a cabo por los programadores en cada HU, ya que estas no ofrecen el nivel de detalle requerido para saber qué implementar. Según el Plan de Release las HU se agruparon en una iteración. A continuación se muestra las tareas de ingeniería derivadas de cada HU.

Tabla 7 Tarea de Ingeniería 1

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** HMAST_Firewall_1
Nombre Tarea: Prueba comando para realizar la instalación y desinstalación del servicio.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.50
Fecha Inicio: 03/02/14	**Fecha Fin:** 07/02/14
Programador Responsable: Javier Ramón Vila	
Descripción: Se prueban para instalar el comando apt-get install Iptables y para desinstalar el comando apt-get purge Iptables.	

Tabla 8 Tarea de Ingeniería 2

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** HMAST_Firewall_1
Nombre Tarea: Diseño de una interfaz que permita la instalación y desinstalación del servicio.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.50
Fecha Inicio: 07/02/14	**Fecha Fin:** 12/02/14
Programador Responsable: Javier Ramón Vila	
Descripción: Se diseña una interfaz que muestra los botones para poder realizar las acciones de instalar o parar el servicio.	

Tabla 9 Tarea de Ingeniería 3

	Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** HMAST_Firewall_2
Nombre Tarea: Prueba de la configuración de la tabla Nat.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 0.3
Fecha Inicio: 12/02/14	**Fecha Fin:** 15/02/14
Programador Responsable: Javier Ramón Vila	
Descripción: Se prueban la configuración y los diferentes componentes de la tabla Nat.	

Tabla 10 Tarea de Ingeniería 4

	Tarea de Ingeniería
Número Tarea: 4	**Número Historia de Usuario:** HMAST_Firewall_2
Nombre Tarea: Diseño de una interfaz que permita la administración de la tabla Nat.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.50
Fecha Inicio: 15/02/14	**Fecha Fin:** 2002/14
Programador Responsable: Javier Ramón Vila	
Descripción: Se diseña una interfaz que le hará escoger el tipo de cadena a la que a hacer referencia, dependiendo de la cadena son las opciones que saldrán, éstas incluyen varios campos de texto, uno para modificar el puerto de entrada. Se encuentra también un campo para establecer el ip al cual se le dirigirá o entrará la conexión, además se podrá escoger el protocolo.	

Tabla 11 Tarea de Ingeniería 5

	Tarea de Ingeniería
Número Tarea: 5	**Número Historia de Usuario:** HMAST_Firewall_3
Nombre Tarea: Prueba de la configuración de la tabla Mangle.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 0.50
Fecha Inicio: 20/02/14	**Fecha Fin:** 25/02/14
Programador Responsable: Javier Ramón Vila	

Descripción: Se prueban la configuración y los diferentes componentes de la tabla Mangle.

Tabla 12 Tarea de Ingeniería 6

Tarea de Ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** HMAST_Firewall_3
Nombre Tarea: Diseño de una interfaz que permita la administración de la tabla Mangle.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.50
Fecha Inicio: 25/02/14	**Fecha Fin:** 30/02/14
Programador Responsable: Javier Ramón Vila	
Descripción: Se diseña una interfaz que le hará escoger el tipo de cadena a la que a hacer referencia, estas incluyen varios campos de texto, uno para modificar el tiempo de vida del paquete, otro para modificar el tipo de servicio y por último uno para establecer una marca.	

Tabla 13 Tarea de Ingeniería 7

Tarea de Ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** HMAST_Firewall_4
Nombre Tarea: Prueba de los métodos para salvar, cargar y el filtrado de las reglas por cadenas.	
Tipo de Tarea : Investigación	**Puntos Estimados:** 0.50
Fecha Inicio: 30/02/14	**Fecha Fin:** 5/03/14
Programador Responsable: Javier Ramón Vila	
Descripción: Se prueban los métodos de salvado, cargado y filtrado de las reglas según las cadenas.	

Tabla 14 Tarea de Ingeniería 8

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** HMAST_Firewall_4
Nombre Tarea: Diseño de una interfaz que permita salvar, cargar o filtrar reglas por cadenas.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.50

Fecha Inicio: 5/03/14	**Fecha Fin:** 10/03/14
Programador Responsable: Javier Ramón Vila	
Descripción: Se diseña una interfaz que muestra los botones para poder realizar las acciones de salvar o cargar el servicio, además de poder filtrar las reglas según las cadenas a las que estén asociadas.	

3.3 Implementación de la solución

Luego de ser analizadas las HU por parte del cliente, queda estimado el tiempo y esfuerzo dedicado para desarrollar cada una de ellas, realizándose 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. A continuación se muestra la tabla que recoge esta información en el Plan de Release del módulo propuesto:

Tabla 15 Plan de Release

Release	Descripción de la iteración	Orden de las HU a implementar	Duración total
1	Dar cumplimiento a las HU que describen los requisitos funcionales de prioridad Muy Alta.	HMAST_Firewall_2, HMAST_Firewall_3, HMAST_Firewall_4.	10 semanas
2	Dar cumplimiento a las HU que describen los requisitos funcionales de prioridad *Alta*.	HMAST_Firewall_1.	8 semanas

3.4 Pruebas de software.

El instrumento adecuado para determinar el estado de la calidad de un producto de software es el proceso de pruebas. En el mismo se ejecutan pruebas dirigidas a componentes del sistema de software en su totalidad, con el objetivo de medir el grado en que la aplicación cumple con los requerimientos.

Este no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. La creciente inclusión del software como un elemento más de muchos sistemas y la importancia de los costos asociados a un fallo del mismo han motivado la creación de pruebas más minuciosas y bien planificadas [24].

Las pruebas permiten 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. Las propuestas por SXP son las Pruebas de Aceptación destinadas a evaluar si al final de una iteración se consiguió la funcionalidad

requerida diseñadas por el cliente final.

3.4.1 Aplicación de Pruebas de Aceptación.

Las Pruebas de Aceptación también conocidas como pruebas del cliente, son utilizadas para probar que las HU han sido implementadas de forma correcta al final de cada iteración y por lo tanto comprueban que las funcionalidades del sistema se encuentran en relación con las HU definidas. A continuación se muestran los casos de pruebas correspondientes con la HU Gestionar reglas NAT.

Tabla 16 Caso de Prueba 1

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-4*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se puede adicionar una cadena prerouting cuando el tipo de acción es DNAT.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Escoger la opción de añadir una regla NAT de cadena Prerouting con el tipo de acción DNAT. 3-Llenar los datos de la regla con la siguiente información: IP de entrada, IP de salida, puerto de entrada, puerto de salida, mac de entrada, interfaz de entrada, protocolo, **DNAT**. 4- Seleccionar la opción de salvar.	
Resultado Esperado: Se añade la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 17 Caso de Prueba 2

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-5*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se puede adicionar una cadena prerouting cuando el tipo de acción es Redirect.	

Condiciones de Ejecución: Estar conectado a un servidor.
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Escoger la opción de añadir una regla NAT de cadena prerouting con el tipo de acción es Redirect. 3- Llenar los datos de la regla con la siguiente información: puerto de entrada,puerto de salida,ip de destino,ip de origen,interfaz de entrada,máscara ,protocolo,**REDIRECT** y puerto(s) destino. 4- Seleccionar la opción de salvar.
Resultado Esperado: Se añade la regla.
Evaluación de la Prueba: Satisfactoria

Tabla 18 Caso de Prueba 3

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-6*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se puede adicionar una cadena output cuando el tipo de acción es Redirect.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1- Abrir el módulo de Iptables. 2- Escoger la opción de adicionar una cadena output cuando el tipo de acción es Redirect. 3- Llenar los datos de la regla con la siguiente información: puerto de entrada,puerto de salida,ip de destino,ip de origen,interfaz de salida,protocolo,**REDIRECT** y puerto(s) destino. 4- Seleccionar la opción de salvar.	
Resultado Esperado: Se adiciona la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 19 Caso de Prueba 4

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-*	**Nombre Historia de Usuario:** Gestionar reglas NAT.

Firewall_2-7	
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se puede adicionar una cadena output cuando el tipo de acción es DNAT.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1- Abrir el módulo de Iptables. 2- Escoger la opción de adicionar una cadena output cuando el tipo de acción es Redirect. 3- Llenar los datos de la regla con la siguiente información: puerto de entrada, puerto de salida, IP de destino, IP de origen, interfaz de salida, protocolo, **DNAT** y IP(s) destino. 4- Seleccionar la opción de salvar.	
Resultado Esperado: Se añade la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 20 Caso de Prueba 5

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-8*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se puede adicionar una cadena Postrouting cuando el tipo de acción es SNAT.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1- Abrir el módulo de Iptables. 2- Escoger la opción de adicionar una cadena Postrouting cuando el tipo de acción es SNAT. 3- Llenar los datos de la regla con la siguiente información: puerto de entrada,puerto de salida,ip de destino,ip de origen,interfaz de salida,protocolo,**SNAT** y ip(s) origen. 4- Seleccionar la opción de salvar.	
Resultado Esperado: Se añade la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 21 Caso de Prueba 6

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-9*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se puede adicionar una cadena Postrouting cuando el tipo de acción es Masquerade.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1- Abrir el módulo de Iptables. 2- Escoger la opción de adicionar una cadena Postrouting cuando el tipo de acción es Masquerade. 3- Llenar los datos de la regla con la siguiente información: puerto de entrada,puerto de salida,ip de destino,ip de origen,interfaz de salida,protocolo,**MASQUERADE** y puerto(s) a enmascarar. 4- Seleccionar la opción de salvar.	
Resultado Esperado: Se añade la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 22 Caso de Prueba 7

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-10*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se puede adicionar una cadena Postrouting cuando el tipo de acción es Masquerade.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1- Abrir el módulo de Iptables. 2- En la selección de la interfaz verificar que se muestra las interfaces de la máquina.	
Resultado Esperado: Se muestran las interfaces del servidor.	

Evaluación de la Prueba: Satisfactoria

Tabla 23 Caso de Prueba 8

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-11*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que los protocolos solo pueden ser TCP, UDP, ICMP y ALL.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Verificar que en la opción del protocolo solo se puede escoger TCP, UDP, ICMP y ALL.	
Resultado Esperado: Solo se pasan las opciones que se indicaron.	
Evaluación de la Prueba: Satisfactoria	

Tabla 24 Caso de Prueba 9

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-12*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se adiciona una regla cuando el puerto de destino es menor que 0 o mayor que 65535.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Adicionar la regla poniendo el puerto 5555. 3-Salvar al configuración.	
Resultado Esperado: Se adiciona la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 25 Caso de Prueba 10

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-13*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se lanza excepción cuando el puerto de destino es menor que 0 o mayor que 65535.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Adicionar la regla poniendo el puerto 55555.	
Resultado Esperado: Se lanza excepción indicando que el puerto esta erróneo.	
Evaluación de la Prueba: Satisfactoria	

Tabla 26 Caso de Prueba 11

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-14*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se adiciona una regla cuando el puerto de origen es mayor que 0 o menor que 65535	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Adicionar la regla poniendo todas las opciones bien. Las opciones escogidas son: puerto de salida,ip de destino,ip de origen,interfaz de salida,protocolo,tipo de acción. 3- Salvar la configuración.	
Resultado Esperado: Se adiciona la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 27 Caso de Prueba 12

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-15*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se lanza excepción cuando el puerto de origen es menor que 0 o mayor que 65535.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Adicionar todas las opciones de la regla bien excepto las del puerto.	
Resultado Esperado: Se lanza excepción indicando que el puerto esta erróneo.	
Evaluación de la Prueba: Satisfactoria	

Tabla 28 Caso de Prueba 13

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-16*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se adiciona una regla cuando IP de origen es incorrecto.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Adicionar la regla poniendo todas las opciones bien excepto el IP de origen. Las opciones escogidas son: puerto de salida,puerto de entrada,ip de destino,interfaz de salida,protocolo,tipo de acción. 3- Salvar la configuración.	
Resultado Esperado: Se adiciona la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 29 Caso de Prueba 14

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-17*	**Nombre Historia de Usuario:** Gestionar reglas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que no se adiciona una regla cuando la máscara de origen es incorrecta.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Adicionar la regla poniendo todas las opciones bien excepto la máscara de origen. Las opciones escogidas son: puerto de salida, puerto de entrada, IP de destino, IP de origen, interfaz de salida, protocolo, tipo de acción. 3- Salvar la configuración.	
Resultado Esperado: Se adiciona la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 30 Caso de Prueba 15

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_2-33*	**Nombre Historia de Usuario:** Gestionar políticas NAT.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se listan las políticas de tipo NAT.	
Condiciones de Ejecución: Estar conectado a un servidor. Tener Iptables instalado.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Se debe escoger la opción de reglas de tipo NAT y debe salir un listado con las políticas ya introducidas.	
Resultado Esperado: Se listan las políticas de tipo NAT.	
Evaluación de la Prueba: Satisfactoria	

Conclusiones del Capítulo

Al finalizar el presente capítulo se concluye que la definición de dos iteraciones en el Plan de Release, permitió una mejor organización en el desarrollo de los requerimientos funcionales ya que se implementaron en una primera iteración los de mayor prioridad y complejidad, y en una segunda iteración los menos importantes. Además, se describieron las Tareas de Ingeniería, las cuales permitieron comprobar que los requerimientos fueran correctos y facilitaron el desarrollo de la solución, permitiendo un ahorro considerable de tiempo ya que indicaban el camino a seguir para darle cumplimiento a las HU.

La ejecución de las Pruebas de Aceptación permitió identificar que el software cumple con todos los requisitos funcionales definidos al principio de la investigación, por lo que el módulo está apto para ser usado en HMAST.

Conclusiones Generales

Una vez cumplidos los objetivos se arribó a las siguientes conclusiones:

- El estudio realizado sobre las herramientas de código abierto para la administración de cortafuegos permitió determinar las funcionalidades a implementar y la forma más adecuada para realizar esta acción en HMAST.

- Las funcionalidades desarrolladas facilitan la administración y configuración del módulo de cortafuegos anteriormente desarrollado. Por lo que con el módulo desarrollado la herramienta HMAST es más competitivo con respecto a las herramientas existentes para la administración de cortafuegos basados en Software Libre y código abierto.

- La ejecución de pruebas realizadas al culminar cada nueva funcionalidad y cada iteración garantizó el desarrollo de un módulo con la calidad requerida por el cliente.

Recomendaciones

Para el desarrollo de futuras versiones se recomienda implementar funcionalidades que provean al módulo de un asistente de configuración de una infraestructura de red.

Referencia Bibliográfica

1. ZIEGLER, Robert. *Guía Avanzada Firewalls Linux*. 1era Edición. Prentice Hall PTR. ISBN 8420529494.

2. Características de la protección firewall. [en línea]. [Consultado 11 Febrero 2014]. Disponible en: http://www.pandasecurity.com/homeusers/downloads/docs/product/help/ap/2014/sp/530.htm

3. GAMMA, Erich, HELM, Richard, JONSON, Ralph and VLISSIDES, John. *"Design Patterns—Elements of Reusable Software*. 1st. ISBN 078-5342633610.

4. PURDY, Gregor. *Linux Iptables Pocket Reference* [en línea]. First Edition. O'Reilly, 2004. Disponible en: safari.oreilly.com

5. *Iptables - Wikiunix* [en línea]. [Consultado 9 Mayo 2014]. Disponible en: http://osl2.uca.es/wikiunix/index.php/Iptables

6. Iptables. [en línea]. [Consultado 9 Mayo 2014]. Disponible en: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-iptables.html

7. Shorewall 4.4/4.5/4.6 Documentation. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://shorewall.net/Documentation_Index.html

8. Zentyal 3.4 Documentación Oficial — Documentación de Zentyal 3.4. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://doc.zentyal.org/es/

9. Uncomplicated Firewall. [en línea]. [Consultado 11 Febrero 2014]. Disponible en: https://wiki.ubuntu.com/UncomplicatedFirewall

10. Webmin. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://www.webmin.com/

11. SIERRA, Kathy and BART, Bates. *Head First Java*. 2da. O'Reilly. 2012

12. SEGURA SALAZAR, Juan. *Características de JAVA* [en línea]. Disponible en: http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm

13. MUDUNURI, Srinivas. *Spring Framework: A Step by Step Approach for Learning Spring Framework*. 1era. 2013. ISBN 978-1482395983.

14. MORRISON, Michael. *Head First JavaScript*. O'Reilly, 2008. ISBN 978-0-596-52774-7.

15. CASTRO, Elizabeth. *HTML 4 for the World Wide Web: VQS*. 3ra. ISBN 978-0201696967.

16. NetBeans IDE - Overview. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: https://netbeans.org/features/index.html

17. LARMAN, Craig. *UML y Patrones,Introducción al análisis y diseño orientado a objetos*. ISBN 970-17-0261-1.

18. RapidSVN. [en línea]. [Consultado 11 Febrero 2014]. Disponible en: http://www.rapidsvn.org/

19. Augeas — Main. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://augeas.net/

20. PEÑALVER ROMERO, GLADYS MARSI. *Metodología ágil para proyectos de Software Libre.* Habana, Cuba: Universidad de las Ciencias Informáticas, 2008.

21. CASTRO, Cesar, RAMOS, Miguel Angel and CALVARRO, Javier. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0*. Unai de la Torre Llorente. ISBN 978-84-936696-3-8.

22. BRUEGGE, Bernd and DUTOIT, Allen. *Object Oriented Software Engineering*. Prentice Hall. 2013

23. GUTIERREZ, Demian. *UML Diagramas de Paquetes* [en línea]. 2009. Universidad de los Andes. Disponible en: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf

24. PRESSMAN, Roger. *Ingeniería del Software, Un enfoque práctico*. 6ta. ISBN 9701054733.

Bibliografía

ZIEGLER, Robert. *Guia Avanzada Firewalls Linux*. 1era Edición. Prentice Hall PTR. ISBN 8420529494.

Características de la protección firewall. [en línea]. [Consultado 11 Febrero 2014]. Disponible en: http://www.pandasecurity.com/homeusers/downloads/docs/product/help/ap/2014/sp/530.htm

GAMMA, Erich, HELM, Richard, JONSON, Ralph and VLISSIDES, John. *"Design Patterns—Elements of Reusable Software*. 1st. ISBN 078-5342633610.

PURDY, Gregor. *Linux Iptables Pocket Reference* [en línea]. First Edition. O'Reilly, 2004. Disponible en: safari.oreilly.com

Iptables - Wikiunix [en línea]. [Consultado 9 Mayo 2014]. Disponible en: http://osl2.uca.es/wikiunix/index.php/Iptables

iptables. [en línea]. [Consultado 9 Mayo 2014]. Disponible en: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-iptables.html

Shorewall 4.4/4.5/4.6 Documentation. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://shorewall.net/Documentation_Index.html

Zentyal 3.4 Documentación Oficial — Documentación de Zentyal 3.4. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://doc.zentyal.org/es/

Uncomplicated Firewall. [en línea]. [Consultado 11 Febrero 2014]. Disponible en: https://wiki.ubuntu.com/UncomplicatedFirewall

Webmin. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://www.webmin.com/

SIERRA, Kathy and BART, Bates. *Head First Java*. 2da. O'Reilly. 2012

SEGURA SALAZAR, Juan. *Características de JAVA* [en línea]. Disponible en: http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm

MUDUNURI, Srinivas. *Spring Framework: A Step by Step Approach for Learning Spring Framework*. 1era. 2013. ISBN 978-1482395983.

MORRISON, Michael. *Head First JavaScript*. O'Reilly, 2008. ISBN 978-0-596-52774-7.

CASTRO, Elizabeth. *HTML 4 for the World Wide Web: VQS*. 3ra. ISBN 978-0201696967.

LARMAN, Craig. *UML y Patrones,Introducción al análisis y diseño orientado a objetos*. ISBN 970-17-0261-1.

RapidSVN. [en línea]. [Consultado 11 Febrero 2014]. Disponible en: http://www.rapidsvn.org/

Augeas — Main. [en línea]. [Consultado 8 Mayo 2014]. Disponible en: http://augeas.net/

CASTRO, Cesar, RAMOS, Miguel Angel and CALVARRO, Javier. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0*. Unai de la Torre Llorente. ISBN 978-84-936696-3-8.

BRUEGGE, Bernd and DUTOIT, Allen. *Object Oriented Software Engineering*. Prentice Hall. 2013

GUTIERREZ, Demian. *UML Diagramas de Paquetes* [en línea]. 2009. Universidad de los Andes. Disponible en: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf

PRESSMAN, Roger. *Ingeniería del Software, Un enfoque práctico*. 6ta. ISBN 9701054733.

FOWLER, Martin. *Analysis Patterns* .Booch Jacobson Rumbaugh.2012

LADRON DE GUEVARA, Jorge Martinez. *Fundamentos de Programación en Java*. EME. ISBN 978-84-96285-36-2

SHALLOWAY, Alan and TROTT, James. *Design Patterns Explained*. Modeller. 2012

FREEMAN, Elizabeh and Eric, *Head First Design Patterns.* O'REILLY. 2004. ISBN: 0596007124.

MCLAUGHLIN, Bratt, POLLICE,Gary and WEST,David. *Head First Object-Oriented Design and Analysis.* O'REILLY. ISBN: 978-0-596-00867-3.

ORAMAS GOÑI, Angel. *Metodología para la gestión de proyectos de Consultoría en Migración a*

Tecnologías de Software Libre y Código Abierto. 2012*.*

CASTILLO ARBELO, Reidiel and SORIA ACOSTA, Pablo. *Herramienta para la Migración y Administración de Servicios Telemáticos.*2012

BARRIOS DUEÑAS, Joel. *Configuración de Servidores con GNU/Linux*. 2014

Glosario de Términos

1. **Dirección IP:** Dirección de un ordenador dentro de una red con protocolo TCP/IP.

2. ***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.

3. ***Gateway*:** Dispositivo dedicado a intercomunicar sistemas con protocolos incompatibles. Puerta de enlace, acceso, pasarela. Nodo en una red informática que sirve de punto de acceso a otra red.

4. **GPL:** es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está 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.

5. **Historias de Usuario (HU):** Secuencias de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de las secuencias.

6. **IDE:** Entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE'). Es un programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.

7. **Instalar:** Incorporar a la computadora un programa o dispositivo para ser utilizado.

8. **IPV4:** Dirección IP versión 4, direcciones de 32 bits empleadas por las computadoras para el acceso a la red.

9. ***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.

10. **MAC:** Es un identificador hexadecimal de 48 bits que se corresponde de forma única con una tarjeta o interfaz de red.

11. **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.

12. **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.

13. **SXP:** Metodo*logí*a compuesta por las metodo*logí*as 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.

14. **TCP/IP**: Transfer Control Protocol / Internet Protocol. Protocolos que se utiliza en Internet para transmitir datos. El TCP está orientado a la conexión que establece una línea de diá*log*o entre el emisor y el receptor antes de que se transfieran los datos.

15. **Fichero**: archivo o documento que se encuentra en un sistema de cómputo, puede ser de distintos tipos (ejemplo de texto, configuración, imagen, etc.) según el fin con el que fue creado.

16. **VLAN**: redes lógicas independientes dentro de una misma red física.

17. **DMZ:** diseño conceptual de red donde los servidores de acceso público se colocan en un segmento separado aislado de la red.

18. **Espacio de Usuario:** se refiere a un espacio de aplicación, típicamente en Unix o en sistemas operativos tipo Unix, el cual es externo al núcleo.

19. **NAT**: es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles.

20. **TTL:** Tiempo de vida de un paquete en la tabla mangle (*Time to Live*).

21. **TOS:** Tipo de Servicio que se va a ofrecer (*Tipe of Service*).

Anexos

Tabla 31 HMAST-Firewall_1-2

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_1-2*	**Nombre Historia de Usuario:** Gestionar estado del firewall.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que no se instala si ya se encuentra instalado el servicio Iptables.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Verificar que el modulo se abre directamente porque está el servicio instalado.	
Resultado Esperado: Se abre el modulo porque el servicio está instalado.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 32 HMAST-Firewall_1-3

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_1-3*	**Nombre Historia de Usuario:** Gestionar estado del Firewall.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Probar que el servicio Iptables se desinstala.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Seleccionar la opción de desinstalar el servicio.	
Resultado Esperado: Iptables se desinstala.	
Evaluación de la Prueba: Satisfactoria	

Tabla 33 HMAST-Firewall_3-18

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_3-18*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se lanza excepción cuando se edita una regla de tipo mangle y no se encuentra ninguna seleccionada.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Seleccionar la opción de editar una regla sin tener la regla seleccionada.	
Resultado Esperado: Se lanza una excepción indicando que la regla debe ser seleccionada.	
Evaluación de la Prueba: Satisfactoria	

Tabla 34 HMAST-Firewall_3-20

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_3-20*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se pueden editar reglas MANGLE.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Seleccionar la regla mangle que se desea editar. 3-Editar la regla. 4-Salvar la configuración.	
Resultado Esperado: Se edita la regla mangle.	
Evaluación de la Prueba: Satisfactoria	

Tabla 35 HMAST-Firewall_3-23

Caso de Prueba de Aceptación

Código Caso de Prueba: *HMAST-Firewall_3-23*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se pueden eliminar reglas MANGLES.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Seleccionar la regla MANGLE que se desea eliminar. 3-Seleccionar la opción de eliminar regla. 4-Salvar la configuración.	
Resultado Esperado: Se elimina la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 36 HMAST-Firewall_3-24

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_3-24*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se pueden adicionar reglas mangles.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Seleccionar la opción de adicionar regla mangle. 3-Llenar los campos de la regla con todos los datos bien. Los datos seleccionado son: tipo de ttl,valor del ttl,tos o mark. 4-Salvar la configuración.	
Resultado Esperado: Se adiciona la regla.	
Evaluación de la Prueba: Satisfactoria	

Tabla 37 HMAST-Firewall_3-25

Caso de Prueba de Aceptación

Código Caso de Prueba: *HMAST-Firewall_3-25*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que la cadena de la regla MANGLE solo pueden ser prerouting, postrouting, input, output y fordward.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Seleccionar la opción de adicionar regla. 3-Verificar que el tipo de cadena que se puede adicionar son: prerouting, postrouting, input, output y fordward.	
Resultado Esperado: Solo se pueden adicionar las relgas de los tipos establecidos.	
Evaluación de la Prueba: Satisfactoria	

Tabla 38 HMAST-Firewall_3-26

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_3-26*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se lanza excepción cuando no se selecciona una de las opciones del TTL y se hace una comparación.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Hacer una comparación sin estar un TTL seleccionado.	
Resultado Esperado: Se lanza excepción indicando que se debe seleccionar un TTL.	
Evaluación de la Prueba: Satisfactoria	

Tabla 39 HMAST-Firewall_3-27

Caso de Prueba de Aceptación

Código Caso de Prueba: *HMAST-Firewall_3-27*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se lanza excepción cuando se escoge que se va a modificar con TOS y no se selecciona una de las opciones del TOS.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Escoger la opción de añadir una regla de tipo mangle 3-Modificar un TOS sin estar seleccionado alguna de las opciones del TOS	
Resultado Esperado: Se lanza excepción indicando que se debe seleccionar un TOS.	
Evaluación de la Prueba: Satisfactoria	

Tabla 40 HMAST-Firewall_3-28

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_3-28*	**Nombre Historia de Usuario:** Gestionar reglas Mangle.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se lanza excepción cuando el valor al MARK pasado es Mayoor que 100 o menor que 0.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Adicionar una regla MANGLE con todos los datos bien, excepto el campo MARK. Los valores seleccionado son: tipo de ttl, valor del ttl, tos. 3-Salvar la configuración.	
Resultado Esperado: Se debe lanzar una excepción cuando el valor al MARK pasado es mayor que 100 o menor que 0.	
Evaluación de la Prueba: Satisfactoria	

Tabla 41 HMAST-Firewall_4-29

Caso de Prueba de Aceptación

Código Caso de Prueba: *HMAST-Firewall_4-29*	**Nombre Historia de Usuario:** Acciones adicionales sobre las reglas.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Se pueden realizar salvas de las reglas. Tener Iptables configurado con al menos una regla.	
Condiciones de Ejecución: Estar conectado a un servidor.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Escoger la opción de salvar la regla en el servidor 3-Seleccionar la dirección de donde se quiere guardar la salva. 4-Salvar la configuración.	
Resultado Esperado: Se deben salvar las reglas.	
Evaluación de la Prueba: Satisfactoria	

Tabla 42 HMAST-Firewall_4-30

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_4-30*	**Nombre Historia de Usuario:** Acciones adicionales sobre las reglas.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Se pueden cargar de las reglas que estaban salvadas.	
Condiciones de Ejecución: Estar conectado a un servidor. Tener Iptables instalado.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Seleccionar la opción de restaurar reglas. 3-Seleccionar fichero con las reglas guardadas. 4-Salvar la configuración.	
Resultado Esperado: Se muestran las reglas salvadas.	
Evaluación de la Prueba: Satisfactoria	

Tabla 43 HMAST-Firewall_3-32

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_3-32*	**Nombre Historia de Usuario:** Gestionar reglas MANGLE.

Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo
Descripción de la Prueba: Verificar que se listan las reglas de tipo MANGLE.
Condiciones de Ejecución: Estar conectado a un servidor. Tener Iptables instalado.
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Se debe escoger la opción de reglas de tipo MANGLE y debe salir un listado con las reglas ya introducidas.
Resultado Esperado: Se listan las reglas de tipo MANGLE.
Evaluación de la Prueba: Satisfactoria

Tabla 44 HMAST-Firewall_3-34

Caso de Prueba de Aceptación	
Código Caso de Prueba: *HMAST-Firewall_3-34*	**Nombre Historia de Usuario:** Gestionar políticas MANGLE.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se listan las políticas de tipo MANGLE.	
Condiciones de Ejecución: Estar conectado a un servidor. Tener Iptables instalado.	
Entrada / Pasos de ejecución: 1-Abrir el módulo de Iptables. 2-Se debe escoger la opción de reglas de tipo MANGLE y debe salir un listado con las políticas ya introducidas.	
Resultado Esperado: Se listan las políticas de tipo MANGLE.	
Evaluación de la Prueba: Satisfactoria	

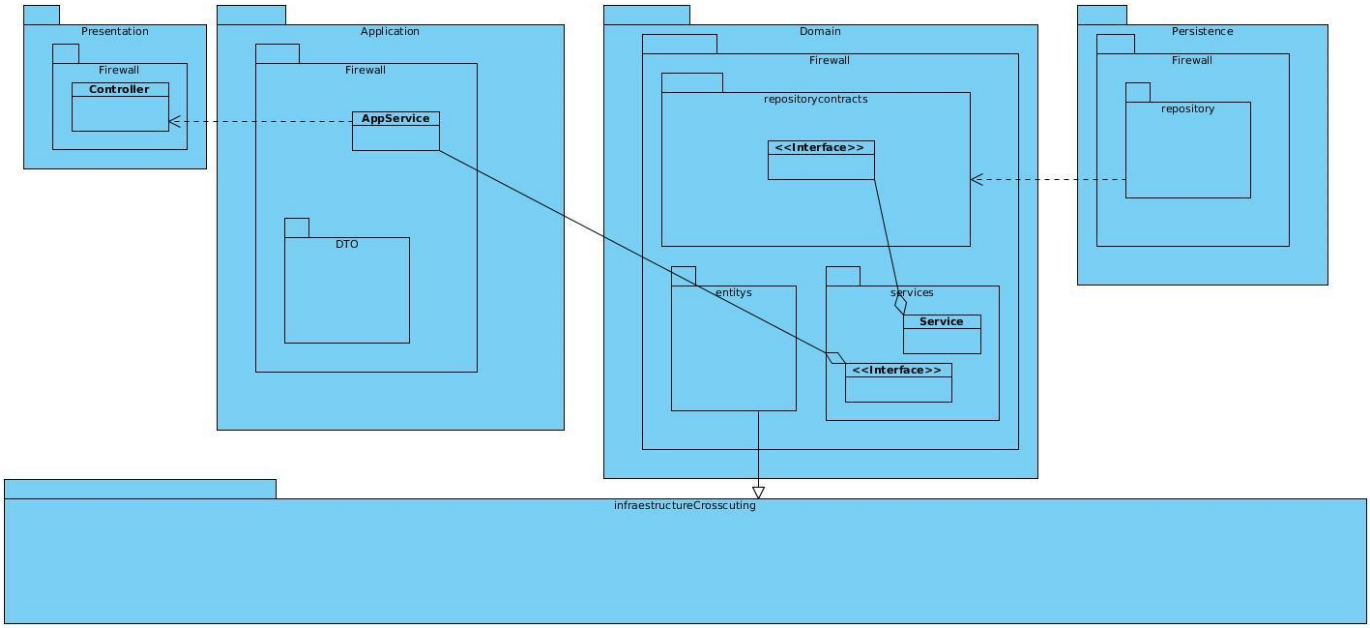


Ilustración 10: Diagrama de Paquetes

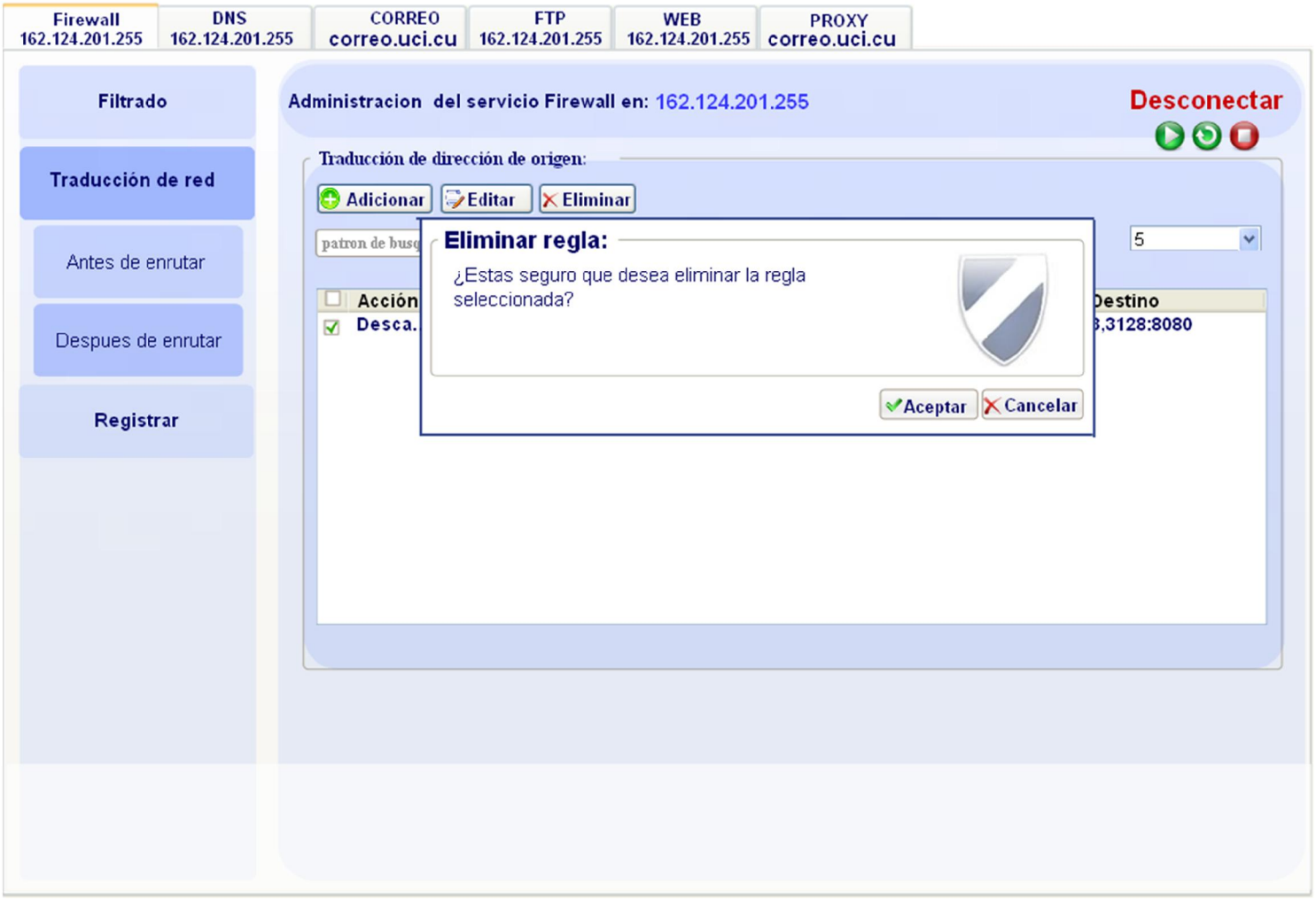


Ilustración 13: Prototipo de Eliminar Regla

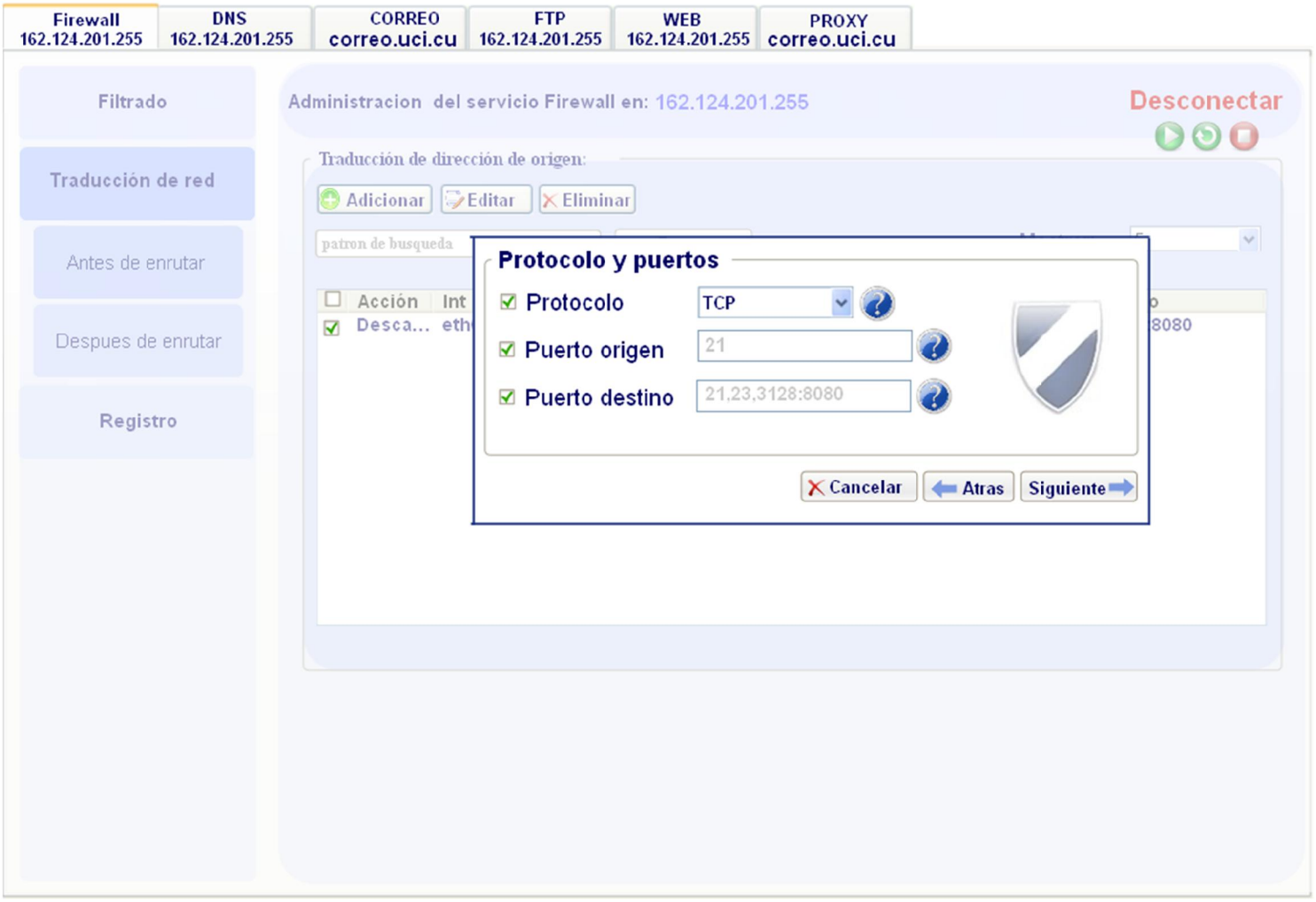


Ilustración 14: Prototipo de Añadir Regla Nat 1

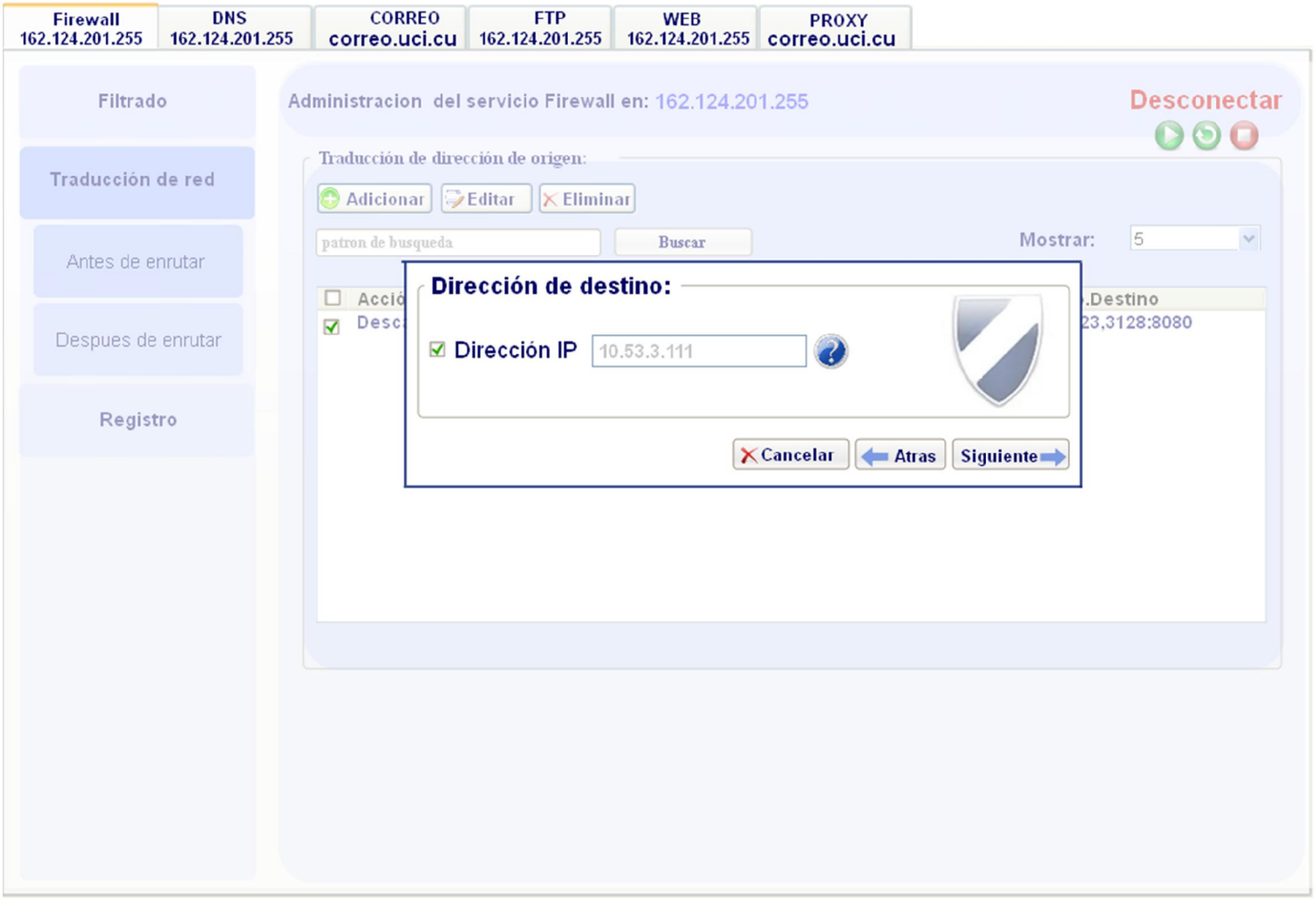


Ilustración 15: Prototipo de Añadir Regla Nat 2

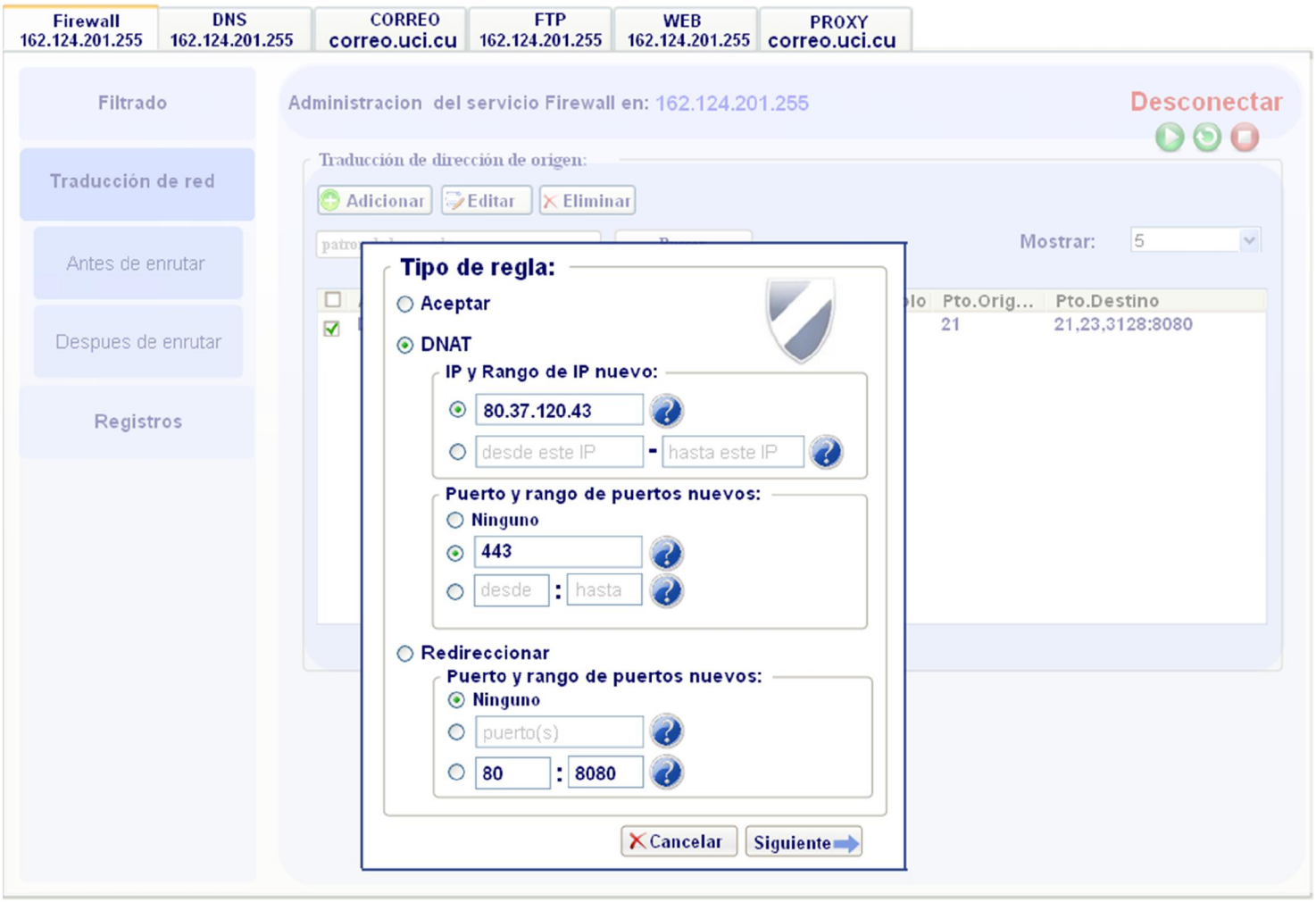


Ilustración 16: Prototipo de Añadir Regla Nat 3

Ilustración 17: Prototipo de Añadir Regla Mangle