Universidad de las Ciencias Informáticas
Facultad 1

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas titulado: Módulo de complementos para el servicio proxy de HMAST

Autor:

Alejandro Campos Palacios

Tutores:

Ing. Inst. Yadiel Pérez Villazón

Ing. Miguel Antonio Conde Gutierrez

Ing. Miriela Velázquez Arias

La Habana, junio 2017

Declaración de autoría

Declaro por este medio que yo Alejandro Campos Palacios, con carné de identidad 92122338844 soy el autor principal del trabajo titulado “Módulo de complementos para el servicio proxy de HMAST” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Para que así conste firmo la presente.

_____

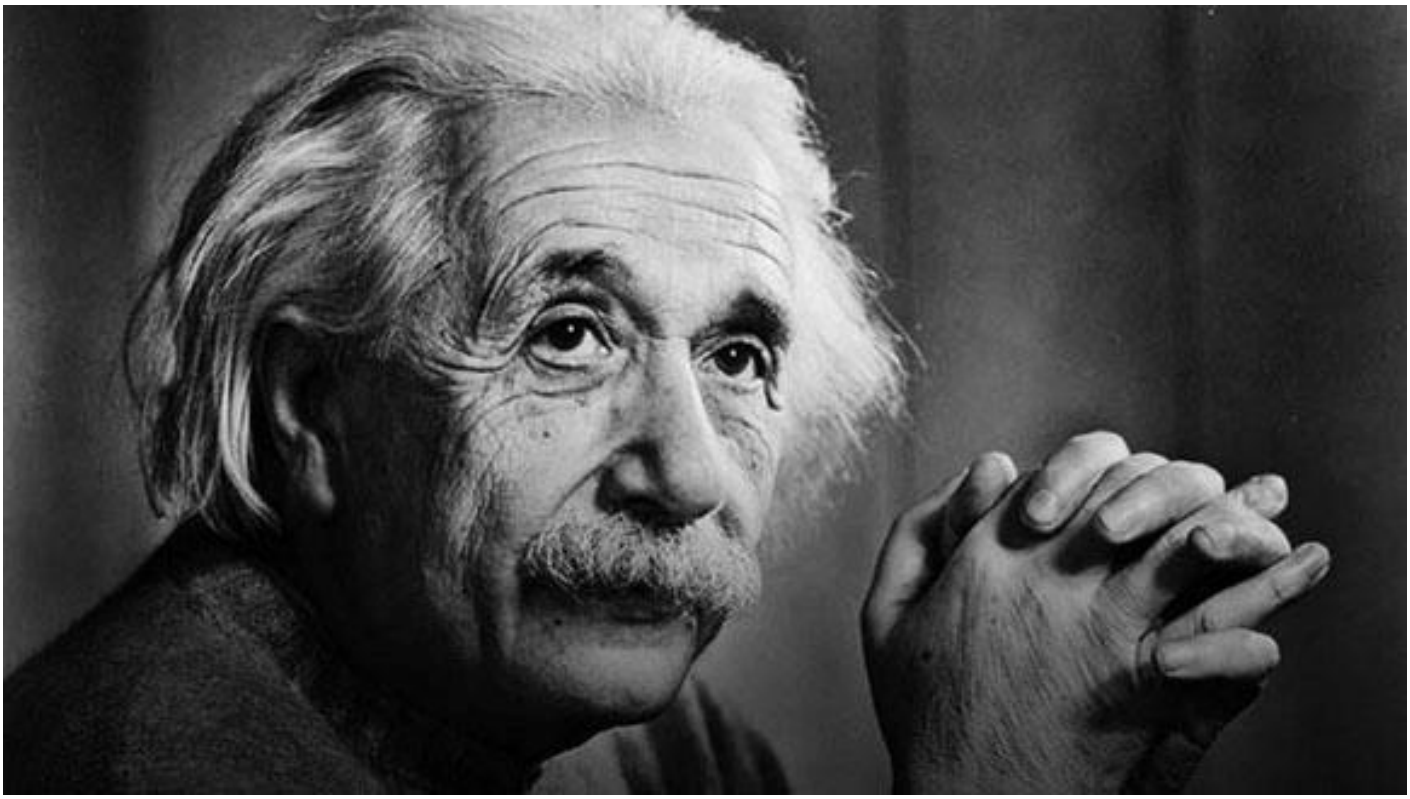
Alejandro Campos Palacios

_____ _____

Yadiel Pérez Villazón Miguel Antonio Conde Gutiérrez

_____

Miriela Velázquez Arias



“El genio se hace con 1% de talento,
y un 99% de trabajo.”

Dedicatoria

Quiero dedicarle el fruto de mi esfuerzo a mi familia, especialmente a mi madre y a mi tía, por ser las mujeres más sabias, fuertes y perseverantes, que he conocido.

Agradecimientos

Es un poco difícil encontrar las palabras adecuadas para agradecer, porque creo que ni todas las palabras del mundo bastarían para agradecerle a mi familia lo que han hecho por mí, pero aquí les dejo mi modesto intento, gracias por todo de verdad, los quiero. También quisiera agradecer a todas aquellas personas que de una forma u otra han formado parte de mi vida, y me han ayudado a llegar hasta aquí, especialmente a mis tutores, gracias por tener tanta paciencia.

Resumen

El presente trabajo se centró en el objetivo de desarrollar funcionalidades que permitan la administración del servicio proxy desde la Herramienta para la Migración y Administración de Servicios Telemáticos en entornos de código abierto. El módulo mencionado surge dada la necesidad de automatizar y centralizar los procedimientos de configuración de un servidor proxy en las empresas e instituciones cubanas. Se empleó el método Analítico-Sintético y el método Análisis Histórico-Lógico. Se documentaron las diferentes tecnologías a usar y la metodología de desarrollo de software Proceso Unificado Ágil en su variante para la Universidad de las Ciencias Informáticas como guía del proceso. Finalmente se obtuvo como resultado un módulo que permite la administración de Squid como servidor proxy, E2guardian como servidor de filtrado de contenido y Squish como herramienta para la gestión de cuotas, obteniendo de esta forma la administración del servicio proxy en diferentes computadoras servidoras de forma remota.

Palabras clave: administración, configuración, E2guardian, servicio proxy, Squid, Squish.

Índice de ilustraciones

Índice de tablas

Introducción

En los últimos tiempos ha resultado notable el desarrollo de las Tecnologías de la Información y la Comunicación (TIC), y como parte de las mismas, es notable además el auge de los servicios telemáticos. Un servicio telemático es aquel que utiliza como soporte servicios básicos, permitiendo el intercambio de información entre terminales con protocolos establecidos para sistemas de interconexión abiertos (1) ; ejemplo de ello lo constituyen: servicio de correo, servicio de nombres de dominio, almacenamiento en la red, y servicio proxy.

El servicio telemático proxy, específicamente, es un grupo de sub-servicios asociados que funcionan de forma conjunta sobre un servidor proxy, este es un ordenador que sirve de intermediario entre un navegador web e internet. Partiendo de que este servicio es el encargado de regular el acceso a internet, su correcta administración resulta estratégica, ya que permite gestionar las cuotas de usuario, reglas de acceso, sitios indebidos, en correspondencia con las políticas de la empresa. En la mayor parte de los Organismos de Administración Central del Estado (OACE) se brinda el servicio proxy, pero en plataformas privativas, en su gran mayoría de la familia Microsoft Windows. Como parte de lo plasmado en el acuerdo 084, el Consejo de Ministros de la República de Cuba establece la migración paulatina de los OACE, teniendo en cuenta que el país aboga por los principios de soberanía tecnológica, y socio adaptabilidad.

Teniendo en cuenta este panorama, el Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI) es el encargado de impulsar la migración a software libre en el país, por lo que sus objetivos van dirigidos al desarrollo de la distribución cubana GNU/Linux Nova, así como la prestación de servicios y desarrollo de soluciones que apoyen esta misión. La Herramienta para la Migración y Administración a Servicios Telemáticos (HMAST) es concebida en el departamento de Servicios Integrales en Migración, Asesoría y Soporte (SIMAYS) de CESOL. Dicha herramienta tiene la finalidad de apoyar la migración de servicios telemáticos a plataformas libres y de código abierto, desde diferentes versiones de Windows Server, así como su posterior administración de forma remota. La última versión liberada es la 2.0 del año 2016. Entre sus módulos se encuentran: SSH[1], APACHE[2], MYSQL[3], DHCP [4]y BACULA[5].

En los sistemas GNU/Linux, la forma básica de administrar el servicio proxy es modificando los ficheros de

[1] *Permite la transferencia de archivos de forma segura.*
[2] *Es un servidor web HTTP de código abierto.*
[3] *Es un sistema de gestión de bases de datos relacional.*
[4] *Es un protocolo cliente/servidor que automáticamente provee direcciones ip.*
[5] *Permite la administración de copias de respaldo.*

configuración, lo cual trae consigo que puedan existir errores humanos, ya que el proceso de configuración es muy engorroso y es de forma manual, siendo en ocasiones imperceptible para el administrador detectar dónde ocurrió el error, dado que cualquier cambio en el fichero principal de configuración podría detener el servicio sin una notificación previa. Además, una administración defectuosa podría atentar contra las políticas de seguridad de la empresa ya que sería posible para los usuarios acceder a sitios no permitidos, u ocupar todo el ancho de banda, afectando de forma significativa el internet de la entidad.

Partiendo de lo anteriormente expuesto se plantea como **problema de la investigación científica:** ¿Cómo realizar una administración del servicio proxy, con gestión de cuotas y control de acceso avanzado para HMAST?

El **objeto de estudio** de la investigación se centra en la administración del servicio proxy para HMAST en GNU/Linux Nova. Por lo que se propone como **objetivo general** desarrollar funcionalidades que permitan la administración del servicio proxy desde HMAST en entornos de código abierto. Se pretende cumplir el objetivo general a partir de los siguientes **objetivos específicos:**

- Caracterizar las tecnologías asociadas a la administración del servicio proxy con el objetivo de adquirir el basamento teórico de la investigación.
- Analizar sistemas informáticos que realizan la administración del servicio proxy.
- Sistematizar la arquitectura de HMAST.
- Definir los requisitos funcionales, no funcionales, así como las historias de usuario.
- Diseñar e implementar las clases y métodos que den solución a los requisitos definidos.
- Diseñar y ejecutar los casos de prueba a las funcionalidades implementadas.

En la presente investigación se plantea como **idea a defender** que el desarrollo del módulo proxy de HMAST facilitará el proceso de administración de los servidores proxy en las instituciones cubanas al tener una administración centralizada con una interfaz de usuario intuitiva.

Para el desarrollo de la presente investigación se tienen en cuenta los siguientes métodos de la investigación científica:

- Dentro de los métodos de nivel teórico, el **Analítico-Sintético** permite realizar la investigación de las herramientas de administración del servicio proxy y un estudio del funcionamiento y las configuraciones necesarias de este servicio, facilitando la implementación de las funcionalidades propuestas.

- También se utilizó el **Análisis Histórico-Lógico,** que posibilita determinar los antecedentes históricos relacionados con las herramientas para la administración del servicio proxy y comprender lógicamente cuáles son las tendencias actuales.

La investigación se ha estructurado para una mejor comprensión en tres capítulos, conclusiones generales y bibliografía. Se cuenta además con los anexos que complementan el trabajo realizado.

Capítulo 1. Fundamentación teórica

En el presente capítulo se ofrece una visión general del servicio proxy y los sub-servicios asociados a este, abordando definiciones y conceptos generales, así como las herramientas de administración. Se describen además las herramientas, tecnologías y metodología de desarrollo definidas en la implementación de HMAST.

1.1 Servicio proxy

Proxy es un compendio de servicios y sub-servicios, los cuales de forma conjunta actúan como intermediarios entre un cliente e internet, de esta forma es evidente que cualquier solicitud hacia la red externa tendrá que pasar a través del proxy, donde será controlado, autorizado o denegado, según las políticas de la empresa (2). Entre los sub-servicios que conforman un proxy se encuentran: los servicios que funcionarán como servidor proxy, servidor de filtrado de contenido y la herramienta para la gestión de cuotas de usuarios. Para una mayor comprensión del tema, a continuación, se especifican y escogen los servicios y herramientas a utilizar.

Servidor Proxy HTTP

Un servidor proxy-http es un sistema informático que está situado entre el cliente que solicita un servicio web y el servidor destino (otro sistema informático) que sirve los recursos solicitados. En su forma más simple, un servidor proxy-http facilita la comunicación entre el cliente y el servidor de destino sin modificar solicitudes o respuestas (3).

Proxy Caché

Un servidor proxy-caché permite incrementar la velocidad de acceso a internet al mantener localmente las páginas más consultadas por los usuarios de una organización, evitando las conexiones directas con los servidores remotos. Los usuarios configuran su navegador para dirigir el acceso al servidor proxy-caché en vez de ir directamente al destino final. El servidor proxy-caché se encarga de proporcionar el recurso solicitado, bien puede ser obteniéndolo de su caché o accediendo directamente al recurso original; al dar servicio a muchos usuarios la caché contendrá muchos recursos almacenados, beneficiándose toda la organización de ello. El uso de un servidor proxy-caché evita transferencias innecesarias y con ello aumenta la velocidad en la carga de las páginas y se reduce el consumo de ancho de banda, ya que no es necesario pedir una página cuando ya esté almacenada en la caché (2).

Proxy Inverso

Un proxy inverso es un servidor proxy-caché "al revés" que, en lugar de permitirles el acceso a internet a usuarios internos, permite a usuarios de internet acceder indirectamente a determinados servidores internos. El servidor proxy inverso es utilizado como un intermediario por los usuarios de internet que desean acceder a un sitio web interno al enviar sus solicitudes indirectamente. Con un proxy inverso, el servidor web está protegido de ataques externos directos, lo cual fortalece la red interna. Además, la función caché de un proxy inverso puede disminuir la carga de trabajo del servidor asignado, razón por la cual se le denomina en ocasiones acelerador de servidor. Finalmente, con algoritmos perfeccionados, el proxy inverso puede distribuir la carga de trabajo mediante la redirección de las solicitudes a otros servidores similares. Este proceso se denomina equilibrio de carga (4) .

Proxy Transparente

Un proxy transparente combina un servicio proxy con *NAT* [6] *(del inglés Network Address Translation),* de manera que las conexiones son enrutadas dentro del proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia. Es utilizado por los proveedores de servicios de internet (ISP). Un servidor proxy transparente es comúnmente instalado únicamente con la finalidad de compartir el internet a varias computadoras, pero normalmente es instalado por los propios usuarios o técnicos sin experiencia en redes, lo cual ocasiona una enorme vulnerabilidad de seguridad a las empresas al no establecer ninguna política de seguridad. Lo cual trae consigo que las computadoras queden expuestas al internet y sean víctimas fáciles de cualquier atacante de la red de redes (5).

1.1.1 Funcionamiento de un servidor proxy caché

Un servidor proxy-caché básicamente funciona como caché de contenido de Red (principalmente HTTP[7] *(del inglés Hypertext Transfer Protocol)*), proporcionando en la proximidad de los clientes un caché de páginas y recursos disponibles a través de la red en servidores HTTP remotos, permitiendo a los clientes de la red local acceder hacia estos de forma más rápida. Cuando se recibe una petición para un recurso de red especificado en una URL (*Uniform Resource Locator*), el servidor a través de las listas de control de acceso (ACL) verifica que el origen y el destino de la petición estén permitidos. (Ver Ilustración 1). Las ACL son reglas que el servidor proxy implementa y son las encargadas de permitir o denegar, respondiendo a las políticas que sean establecidas a través de ellas, la obtención de los contenidos solicitados. Una vez

[6] Es un mecanismo utilizado por routers ip para intercambiar paquetes entre dos redes que asignan mutuamente direcciones incompatibles.

[7] E*s un protocolo de comunicación que permite la transferencia de información red.*

que se verifica que el recurso esté permitido, busca el resultado dentro de su caché, si este es encontrado, responde al cliente proporcionando inmediatamente el contenido solicitado, de lo contrario, lo traerá desde el servidor remoto que contiene el recurso que el cliente solicitó, guardando una copia en la caché. El contenido en la caché es posteriormente eliminado a través de un algoritmo de expiración de acuerdo a la antigüedad, tamaño e historial de respuestas a solicitudes (hits) (6).

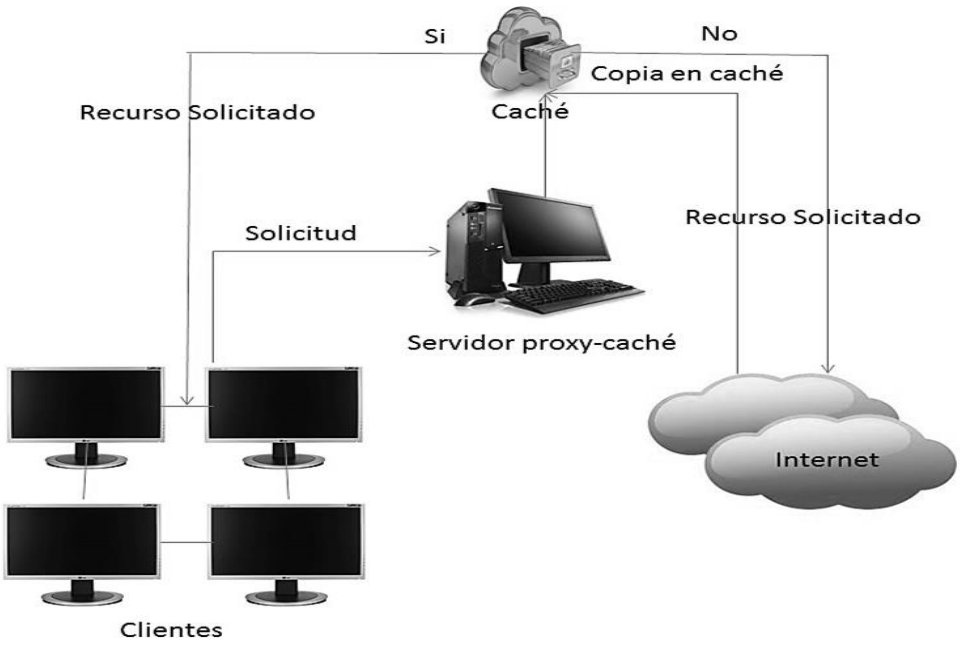


Ilustración 1: Funcionamiento de un proxy-caché.

Una vez analizados los conceptos anteriores se procederá a seleccionar los servicios que se utilizarán como servidor proxy y filtrado de contenido, así como la herramienta para la gestión de cuotas de usuarios.

1.1.2 Selección del servidor proxy

En la actualidad existe una gran variedad de servidores proxy dirigidos a funcionar como intermediarios en sistemas operativos GNU/Linux. A continuación, se especifican las funcionalidades más importantes, así como ventajas y desventajas de cada uno de los servidores y servicios analizados, con el objetivo de definir cuál será utilizado en la administración del servicio proxy desde HMAST.

Varnish web cache

Acelerador web, a veces referido como un acelerador de HTTP o un proxy HTTP inverso, el cual mejora

significativamente el rendimiento de las peticiones de sus clientes. Varnish acelera el cargado de una página web mediante el almacenamiento de una copia de una página que un usuario visita por primera vez. La próxima vez que un usuario solicite la misma página, Varnish servirá la copia en lugar de solicitar la página desde el servidor web. Esto contribuye a que el servidor web tenga que manejar menos tráfico, obteniendo a su vez mayor velocidad en respuesta a la solicitud realizada. La caché hecha por este servidor es a menudo la pieza más crítica de software en un negocio basado en la web (7).

- **Principales características**

Almacena los datos en la memoria virtual y deja la tarea de decidir lo que se almacena en la memoria y lo que se guarda en el disco al sistema operativo. Esto ayuda a evitar la situación en la que el sistema operativo comienza a copiar datos al disco cuando la aplicación está haciendo lo mismo. Una característica significativa de Varnish es que cada conexión cliente se asigna a un subproceso independiente. Cuando se alcanza el límite configurado en el número de subprocesos activos, las conexiones entrantes se colocan en cola y solo cuando esta cola alcanza su límite configurado serán rechazadas las conexiones entrantes. Con el fin de reducir el número de llamadas al sistema a un mínimo, los datos de registro o *logs*[8] se almacenan en la memoria compartida, y la tarea de escribir los registros en el disco se delega a una aplicación separada. Se pueden destacar, las siguientes características de Varnish caché (7):

- Diseño moderno.
- Utiliza *Visual Class Library* (VCL), el cual es un lenguaje de configuración que permite una gran flexibilidad y adaptación. Incluso es posible añadir código C con lo que es posible expandir Varnish sin modificar el código fuente.
- Equilibrio de carga.
- Web GUI (*Beta*), muy útil en caso de configuraciones en clúster.
- Posibilidad de cambiar configuraciones sobre la marcha.
- Reescritura y redireccionamiento de url (no es necesario un programa externo).
- Consulta de estadísticas y logs en tiempo real.

[8] *Base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular.*

Tinyproxy

Proxy ligero HTTP / HTTPS para sistemas operativos POSIX[9]. Diseñado desde cero para ser rápido y pequeño. Tinyproxy es software libre y se distribuye con la licencia GNU GPL (versión 2 o superior) (8).

- **Principales características**

- Tinyproxy posee un tamaño pequeño y requiere de muy pocos recursos del sistema. La huella de memoria tiende a ser alrededor de 2 MB con *glibc*[10], y la carga del CPU aumenta linealmente con el número de conexiones simultáneas (dependiendo de la velocidad de la conexión). Por lo tanto, se puede ejecutar en una máquina de pocas prestaciones, o en un dispositivo de red, como un enrutador de banda ancha basado en Linux, sin ningún impacto notable en el rendimiento.
- Permite el reenvío de las conexiones HTTPS sin modificar el tráfico de cualquier manera a través del método CONNECT.
- Brinda funciones de proxy transparente, de modo que los usuarios puedan navegar sin necesidad de configuración del lado del cliente. También se puede usar como un proxy inverso para sitios web.
- Posee características de privacidad, las cuales permiten configurar qué encabezados HTTP se deben permitir, y cuáles deben ser bloqueados. Esto permite restringir tanto los datos que vienen a su navegador web desde el servidor HTTP, como los que van en sentido contrario.
- Facilita el monitoreo remoto, se puede acceder a las estadísticas del proxy desde cualquier lugar, lo que permite saber exactamente el rendimiento del servidor (8).

Privoxy

Proxy web con capacidades avanzadas de filtrado para proteger la privacidad, filtrar contenido web, gestionar cookies, controlar el acceso, y quitar anuncios, carteles, ventanas emergentes y otra basura de internet. Privoxy tiene una configuración muy flexible y se puede personalizar a las necesidades de la empresa. Privoxy es adecuado tanto para sistemas aislados como para redes multiusuario (9).

- **Principales características**

- Soporta "*Connection*: *keep-alive*". Las conexiones salientes se pueden mantener con vida

[9] *Es el acrónimo de **P**ortable **O**perating **S**ystem **I**nterface, y **X** viene de UNIX como seña de identidad de la API.*
[10] *Es la biblioteca estándar de lenguaje C de GNU. Se distribuye bajo los términos de la licencia GNU LGPL.*

independiente del cliente.

- Compatible con IPv6, siempre que el sistema operativo lo haga también y el script de configuración que lo detecte.
- Soporta el etiquetado que permite cambiar el comportamiento basado en las cabeceras de cliente y servidor.
- Se puede ejecutar como un proxy "interceptador", lo que elimina la necesidad de configurar los navegadores de forma individual.
- Acciones y filtros sofisticados para manipular el servidor y el cliente.
- Puede ser encadenado con otros servidores proxy.
- Configuración modularizada que permite la configuración estándar y configuraciones de usuario a residir en archivos separados, por lo que la instalación de los archivos actualizados de las acciones no sobrescribirá la configuración individual del usuario.
- Filtrado de páginas web.
- Autodetección y relectura de cambios en los archivos de configuración.
- La mayoría de las funciones se pueden controlar en función de cada sitio o por la ubicación (9).

Squid

Squid es un proxy caché para la web con soporte para disímiles protocolos, algunos son HTTP, HTTPS, FTP[11] *(del inglés* File Transfer Protocol). Reduce el ancho de banda y mejora los tiempos de respuesta en caché reutilizando las páginas web solicitadas con frecuencia.

Squid tiene amplios controles de acceso y funciona como un gran servidor acelerador. Se ejecuta en disímiles sistemas operativos, incluyendo *Windows* y está disponible bajo la GNU GPL. Entre otras cosas, Squid puede funcionar como servidor intermediario y caché de contenido de red para los protocolos HTTP, FTP, Proxy de *SSL*[12] *(del inglés* Secure Sockets Layer), caché transparente, aceleración HTTP, caché de

[11] *Es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red, basado en la arquitectura cliente-servidor*.
[12] *Es un protocolo criptográfico que proporciona comunicaciones seguras por una red*.

consultas DNS[13] *(del inglés Domain Name System)* y otras como filtración de contenido y control de acceso por ip y por usuario.

Squid está compuesto por un programa principal como servidor, un programa para búsqueda en servidores DNS, programas opcionales para reescribir solicitudes y realizar autenticación, y algunas herramientas para administración y para clientes. Al iniciar, Squid da origen a un número configurable (5, de modo predeterminado a través de la opción dns_children) de procesos de búsqueda en servidores DNS, cada uno de los cuales realiza una búsqueda única en servidores DNS, reduciendo la cantidad de tiempo de espera para las búsquedas en servidores DNS (10). Las principales funciones de Squid son las siguientes:

- Permite el acceso web a computadoras privadas (IP privada) que no están conectadas directamente a internet.
- Controla el acceso web aplicando reglas.
- Registra el tráfico web desde la red local hacia el exterior.
- Controla el contenido web consultado y descargado.
- Controla la seguridad de la red local ante posibles ataques e intrusiones en el sistema.
- Funciona como una caché de páginas web. Es decir, almacena las páginas web visitadas por los usuarios y de esta manera las puede enviar a otros usuarios sin tener que acceder a internet de nuevo.
- Guarda en caché las peticiones DNS e implementa una caché para las conexiones fallidas.
- Registra *logs* de todas las peticiones cursadas.
- Soporta el protocolo ICP[14] *(del inglés internet Cache Protocol)* que permite integrar cachés que colaboran y permite crear jerarquías de cachés y el intercambio de datos (10).

Como consecuencia de estas funciones, la implantación de un servidor proxy-caché en una red proporciona las siguientes ventajas:

- Reduce los tiempos de respuesta: si la página web que se solicita está en la caché del servidor, esta accede al recurso sin necesidad de acceder de nuevo al servidor original, con lo cual se ahorra tiempo.
- Disminuye el tráfico en la red y el consumo de ancho de banda: si la página web está almacenada en la caché del servidor, la petición no sale de la red local y no será necesario hacer uso de la línea

[13] *Su función más importante es "traducir" nombres inteligibles para las personas en identificadores binarios asociados con los equipos conectados a la red.*
[14] *Su propósito es encontrar la ubicación más apropiada para recuperar un objeto solicitado.*

exterior, consiguiendo así un ahorro en la utilización del ancho de banda.

- Cortafuegos: cuando se utiliza un servidor proxy-caché, este se comunica con el exterior, y puede funcionar como cortafuegos, lo cual aumentará la seguridad del usuario respecto a la información a la que se acceda.
- Filtrado de servicios: es posible configurar el servidor proxy-caché dejando solo disponibles aquellos servicios (HTTP, FTP), sitios, frases, URLs o palabras, que se consideren necesarios, impidiendo la utilización del resto (2).

Selección del servidor

A continuación, se muestra una tabla comparativa donde quedan reflejados aspectos relevantes que caracterizan a cada uno de los servidores proxy analizados anteriormente, así como funcionalidades necesarias en las instituciones cubanas, con el fin de comparar y arribar a conclusiones acertadas que apoyen la selección del que será utilizado para dar solución al problema planteado.

Tabla 1: Comparación de servidores proxy en entornos de código abierto.

Funcionalidades	Squid	Varnish cache	Privoxy	Tiny proxy
Gestión de Listas de control de Acceso (ACL).	Alta	Baja	Alta	Media
Configuración para la comunicación con otros proxy-caché.	Sí	No	Sí	No
Gestión de Cuotas de usuarios.	Sí	No	No	No
Gestión del ancho de banda de la red	Sí	No	No	No
Almacenamiento de las peticiones DNS.	Sí	No	No	No

Una vez analizada la tabla comparativa de los servidores proxy se selecciona Squid como software para administrar el servicio proxy desde HMAST, por resultar ser el más completo de los que fueron analizados.

Este cuenta con un amplio número de ventajas asociadas a las funcionalidades y protocolos que soporta. Entre sus características distintivas se destacan que permite el acceso web a máquinas privadas que no están conectadas directamente a internet, guarda en caché las peticiones DNS e implementa una caché para las conexiones fallidas y soporta el protocolo ICP, el cual permite integrar cachés que colaboran entre sí, así como crear jerarquías de caché e intercambio de datos. Actualmente es el más usado en entornos libres y herramientas reconocidas para la administración de servicios telemáticos como Zentyal y Webmin.

1.1.2.1 Especificaciones técnicas de Squid

A continuación, se referirán elementos técnicos sobre Squid, con el objetivo de lograr una mayor comprensión de las configuraciones básicas para su posterior administración de forma correcta, tomando como referencia el libro Implementación de Servidores con GNU/Linux de Joel Barrios (2).

El archivo principal de configuración de Squid se encontrará localizado en: */etc/squid/squid.conf*. En squid.conf se encontrarán los parámetros de configuración asociados al servicio, a continuación se presentan los principales:

http_port: Esta opción se utiliza con el objetivo de configurar la dirección ip y el puerto por el que el servidor proxy recibirá las peticiones. Por defecto el puerto asignado es el 3128 con ip 0.0.0.0 quedando de la siguiente forma: *http_port* 0.0.0.0:3128.

cache_dir: Esta opción se utiliza con el objetivo de establecer el tamaño deseado que utilice Squid para almacenamiento de caché en el disco duro. De modo predeterminado Squid utilizará el formato *ufs*[15] (*del inglés Unix File System*), para crear en el directorio */var/spool/squid* un caché de 100 MB, dividido en jerarquías de 16 directorios subordinados, hasta 256 niveles cada uno, quedando de la siguiente forma: *cache_dir ufs /var/spool/squid* 100 16 256.

maximum_object_size: Esta opción se utiliza con el objetivo de definir el tamaño máximo de los objetos en el caché, quedando de la siguiente forma: *maximum_object_size* 48 MB.

cache_replacement_policy: Esta opción se utiliza con el objetivo de incluir soporte para los siguientes algoritmos de caché:

[15] *Es un sistema de archivos utilizado por varios sistemas operativos UNIX y POSIX.*

- LRU. Acrónimo de *Least Recently Used* (Menos Usado Recientemente). En este algoritmo los objetos que fueron accedidos hace mucho tiempo, son eliminados primero, manteniendo siempre en caché los objetos recientemente solicitados. Esta política es la utilizada por Squid de modo predeterminado.
- LFUDA. Acrónimo de *Least Frequently Used with Dynamic Aging* (Menos frecuentemente usado con envejecimiento dinámico). En este algoritmo los objetos solicitados permanecen en la caché sin importar su tamaño, optimizando la eficiencia por octetos a expensas de la eficiencia misma, de modo que un objeto grande solicitado con frecuencia impedirá las solicitudes de los objetos pequeños con menor frecuencia.
- GDSF. Acrónimo de *GreedyDual Size Frequency* (Frecuencia de tamaño doble codiciosa). Este algoritmo mantiene en caché los objetos pequeños frecuentemente solicitados.

La opción queda de la siguiente manera: *cache_replacement_policy heap* LFUDA.

cache_mem: Esta opción se utiliza con el objetivo de definir la cantidad ideal de memoria para:

- Objetos en tránsito.
- Objetos frecuentemente utilizados (Hot).
- Objetos negativamente almacenados en el caché.

La opción queda de la siguiente manera: *cache_mem* 48 MB

Las listas de control de acceso

Una de las mayores complejidades de la configuración de un servidor proxy puede estar en las reglas de control de acceso, pues el trabajo principal del proxy es determinar quién puede entrar a determinado lugar, para eso se necesitan listas de control de acceso (acl) con sus respectivas reglas de control de acceso, una infectiva configuración puede permitirle a un usuario acceder a un recurso al cual no está autorizado a hacerlo, o bloquear el acceso a algo a lo cual no tiene por qué tener cerrado el acceso. La sintaxis de una acl puede estar compuesta por estos 4 elementos, donde los 2 primeros son obligatorios y luego puede variar entre si se usa el tercero o el cuarto (ver tabla 2):

Tabla 2: Estructura de una acl.

Nombre de la acl	Nombre que identificará la acl.
Tipo de acl	El tipo de acl declarada.
Cadena	Cadena de caracteres que se comparará.
Archivo	Archivo que contiene la cadena de caracteres que se comparará.

La opción queda de la siguiente manera: acl *network* src 10.0.0.0/8, donde *network* es el nombre de la acl, *src* el tipo, 10.0.0.0/8 la cadena para comparar. Con la finalidad de comprender mejor las listas de control de acceso se describen varios tipos de acl (ver Tabla 3).

Tabla 3: Descripción de una acl.

Tipo de acl	Descripción	Sintaxis	Ejemplo
src	Para las direcciones ip del cliente o los clientes solicitantes.	acl aclname src dir-ip acl aclname src dir1-dir2	acl direccion src 172.16.1.25/32 acl rango src 172.16.1.25-172.16.1.35/32
dst	Para la o las direcciones ip de destino.	acl aclname dst dir-ip	acl dest dst 14.14.0.0/16
srcdomain	Para el dominio del cliente solicitante.	acl aclname srcdomain nombre_dominio	acl midominio srcdomain .uci.cu
dstdomain	Para el dominio del destino.	acl aclname srcdomain nombre_dominio	acl midominio dstdomain cubadebate.cu
srcdom_regex	Para cuando exista la palabra en el dominio del solicitante.	acl aclname srcdom_regex palabra	acl dominiocontiene srcdom_regex uci
dstdom_regex	Para cuando exista la palabra en el dominio del destino.	acl aclname dstdom_regex palabra	acl dominiocontiene dstdom_regex duolingo
time	Para que la acl sea aplicada en determinado tiempo.	acl aclname time días abreviados H1:M1-H2:M2	acl acltime time W 8:00-16:00

url_regex	Para cuando se quiere controlar a una url determinada.	acl aclname url_regex -i url	acl rechaza_url url_regex –i http://www.xxx.com
urlpath_rege x	Para cuando se quiere controlar todas las url que contengan la palabra introducida.	acl aclname urlpath_regex -i url	acl rechazapalabra urlpath_regex -i
port	Para los puertos de destino de las solicitudes.	acl aclname port numero_del_puerto acl aclname port rango	acl puerto1 port 22
proxy_auth	Para llamar procesos externos de autenticación para los usuarios.	acl aclname proxy_auth usuarios	acl autentica proxy_auth jmengano lfulano acl autentica proxy_auth REQUIRED //para todos los usuarios

Reglas de control de acceso

Las reglas de control de acceso definen qué acl tendrá acceso permitido o denegado, a un recurso determinado, la estructura de las reglas es la siguiente: *http_access allow network, http_access deny network, http_access allow !network.* El signo "!" niega la cadena introducida.

Autenticación con Squid

Una vez conocida la estructura y tipos de las listas de control de acceso es necesario configurar el acceso por autenticación de manera básica, con el objetivo de que el usuario "para acceder a una URL" deba proporcionar un usuario y contraseña.

LDAP

La sintaxis utilizada para basic_ldap_auth es la siguiente: *basic_ldap_auth -b "Directorio-a-utilizar" servidor-ldap-a-utilizar,* Ejemplo: *basic_ldap_auth -b "ou=UCI Domain Users,dc=uci,dc=cu" 10.0.0.87*. Posteriormente es necesario crear una lista de acceso y la regla para habilitar la autenticación: *acl password proxy_auth REQUIRED, http_access allow password.*

NCSA

Squid puede utilizar el módulo ncsa_auth de la NCSA (*National Center for Supercomputing Applications*), y

que ya viene incluido como parte del paquete principal de Squid en la mayoría de las distribuciones actuales. Este módulo provee una autenticación muy sencilla a través de un archivo de texto simple, cuyas contraseñas fueron creadas con htpasswd. Se requerirá la creación previa de un archivo que contendrá los nombres de usuarios y sus correspondientes contraseñas (cifradas). El archivo puede localizarse en cualquier lugar del sistema, con la única condición que sea accesible para el usuario squid. Debe procederse a crear un archivo en /etc/squid/claves: *touch /etc/squid/claves,* como medida de seguridad, este archivo debe tener atributos de lectura y escritura solo para el usuario squid: *chmod 600 /etc/squid/claves, chown squid:squid /etc/squid/claves.* Posteriormente se debe dar de alta las cuentas que sean necesarias, utilizando el comando htpasswd. Ejemplo: *htpasswd /etc/squid/claves usuario.* Una vez realizados estos requerimientos se edita el archivo /etc/squid/squid.conf. con el objetivo de especificar qué programa de autenticación se utilizará.

El comando ***auth_param basic program*** es el encargado de definir el tipo de autenticación que tendrán los usuarios de Squid. Ejemplo: *auth_param basic program/usr/lib/squid/basic_ncsa_auth/etc/squid/claves.*

1.1.3 Servicios asociados

Como servicios asociados al servidor proxy Squid se procede a seleccionar un servidor de filtrado de contenido y una herramienta para la gestión de cuotas de usuarios.

Selección del servicio de filtrado de contenido

Dansguardian

Es un software de filtro de contenido diseñado para controlar el acceso a sitios web. Incluye un filtro de virus, importante en sistemas Windows; es usado principalmente en instituciones de educación, gobierno y empresas. Se caracteriza por su alto grado de flexibilidad y adaptación de la implementación. Dansguardian se instala en un ordenador (servidor) con el sistema operativo GNU/Linux, y filtrará contenidos de webs solicitadas por el resto de ordenadores (independientemente del sistema operativo que tengan instalado). Entre sus principales características se encuentran las listas de control de acceso a contenido, estas se encargan de denegar o permitir el acceso a cierto recurso. En su composición Dansguardian cuenta con 33 listas, estas son:

Tabla 4: Listas de Dansguardian

Authplugins	contentscanners	filtergroupslist

bannedextensionlist	downloadmanagers	greysitelist
bannediplist	exceptionextensionlist	greyurllist
bannedmimetypelist	exceptionfilesitelist	headerregexplist
bannedphraselist	exceptionfileurllist	logregexpurllist
bannedregexpheaderlist	exceptioniplist	logsitelist
bannedregexpurllist	exceptionmimetypelist	logurllist
bannedsitelist	exceptionphraselist	phraselists
bannedurllist	exceptionregexpurllist	pics
blacklists	exceptionsitelist	urlregexplist
contentregexplist	exceptionurllist	weightedphraselist

Por tanto, se puede definir que Dansguardian cuenta con un alto grado de flexibilidad y manejabilidad.

E2guardian

E2guardian es un filtro de contenido web de código abierto que filtra el contenido real de las páginas basadas en diversos métodos, incluyendo frase de correspondencia, el encabezado de la solicitud y filtrado de URL. Entre sus principales características se encuentran las listas de control de acceso a contenido, estas se encargan de denegar o permitir el acceso a cierto recurso. En su composición E2guardian cuenta con 63 listas estas son:

Tabla 5: Listas de E2guardian.

addheaderregexplist	downloadmanagers	localbannedurllist
authexceptionsitelist	embededreferersitelist	localexceptionsitelist

authexceptionurllist	embededrefererurllist	localexceptionurllist
authplugins	exceptionextensionlist	localgreysitelist
bannedextensionlist	exceptionfilesitelist	localgreysslsitelist
bannediplist	exceptionfileurllist	localgreyurllist
bannedmimetypelist	exceptioniplist	logregexpurllist
bannedphraselist	exceptionmimetypelist	logsitelist
bannedphraselist.in	exceptionphraselist	logurllist
bannedregexpheaderlist	exceptionphraselist.in	nocheckcertsitelist
bannedregexpurllist	exceptionregexpurllist	nocheckcertsitelist.in
bannedrooms	exceptionsitelist	phraselists
bannedsearchlist	exceptionurllist	pics
bannedsitelist	filtergroupslist	refererexceptionsitelist
bannedsitelist.in	greysitelist	refererexceptionurllist
bannedsslsitelist	greysslsitelist	searchregexplist
bannedurllist	greyurllist	sslsiteregexplist
bannedurllist.in	headerregexplist	urlredirectregexplist
blacklists	localbannedsearchlist	urlregexplist
contentregexplist	localbannedsitelist	weightedphraselist
contentscanners	localbannedsslsitelist	weightedphraselist.in

Como resultado, se puede apreciar que E2guardian tiene mayor grado de flexibilidad y manejabilidad que

Dansguardian, ya que E2guardian contiene todas las listas de Dansguardian y 30 listas adicionales, por tanto, se puede definir como servicio filtrado de contenido a E2guardian. Entre las principales configuraciones de E2guardian se encuentran:

***language*:** Esta opción establece el lenguaje a utilizar.

language = ´spanish´

loglocation: Esta opción establece la ruta de almacenamiento de los logs generados.

loglocation = '/var/log/e2guardian/access.log'

filterip: Esta opción establece el ip por el que escuchará E2guardian, en caso de que esté en blanco, este escuchará por todos los ips.

filterip = 0.0.0.0

filterport: Esta opción establece el puerto por el que E2guardian escuchará.

filterports = 8080

***proxyip*:** Esta opción establece el ip del servidor proxy.

proxyip = 127.0.0.1

proxyport: Esta opción establece el puerto por el que E2guardian se conectará al proxy.

proxyport = 3128

Una vez configurados los parámetros básicos de E2guardian, se procede a configurar las listas de control de acceso a contenido (LCAC) a continuación se brindarán las sintaxis de las principales LCAC:

bannedextensionlist: En esta lista se incluirán extensiones de los archivos a denegar.

.bat

bannediplist: En esta lista se incluirán ips a denegar. Puede ser por ip específico, por rango o subred.

192.168.0.1

10.0.0.1-10.0.0.3

10.0.0.0/24

bannedphraselist: En esta lista se incluirán palabras o frases a denegar.

Para bloquear cualquier página con palabras que contengan la cadena “sex”.

<sex>

Para bloquear cualquier página con la cadena “sex magazine”.

<sex magazine>

Para bloquear cualquier página que contenga la palabra/cadena “sex” y “fetish”.

<sex>, <fetish>

Para habilitar el bloqueo de frases desde un archivo

.Include</etc/e2guardian/list/phraselists/pornography/banned>

bannedsitelist: En esta lista se incluirán los sitios a denegar. Donde "*domains"*es un archivo que contiene una serie de dominios a denegar.

.Include</etc/e2guardian/list/backlist/adults/domains>

Para banear un sitio por un tiempo determinado

time: <start hour> <start minute> <end hour> <end minute> <days>

Ejemplo:

time: <9> <0> <17> <0> <01234>

bannedurllist: En esta lista se incluirán las url a denegar.

members.home.net/uporn

.Include</etc/e2guardian/list/blacklist/ads/urls>

Las listas de excepciones sobrescriben a las bannedlist, y desbloquean los elementos contenidos en estas. Las sintaxis usadas son las mismas, a continuación, se mencionan las principales.

- *exceptionextensionlist*
- *exceptioniplist*
- *exceptionphraselist*
- *exceptionsitelist*
- *exceptionurllist*

Selección de la herramienta para la gestión de cuotas

Como consecuencia de que Squid no tiene soporte nativamente para cuotas es necesario usar Squish. Esta última es una herramienta de terceros para la gestión de cuotas y optimizar el ancho de banda de internet. Su función principal consiste en mantener un registro con las cuotas asignadas a cada usuario y cada cierto periodo actualizar el consumo de los usuarios; en caso de encontrar alguno que haya sobrepasado el límite de cuota asignado, Squish enviará este usuario al fichero sin navegación. Donde el usuario no tendrá acceso a los servicios de internet hasta que su cuota vuelva a ser reiniciada. La sintaxis para escribir un nuevo usuario es la siguiente:

user amount/period

Donde *user* será el usuario dado, *amount* es la cantidad de cuota que será asignada a este usuario y *period* es el período por el que será válida la cuota. Ejemplo: *apalacios 100mb/day (*El usuario apalacios

tendrá 100 megabytes diarios). *.* 4h/day (*Todos los usuarios tendrán 4 horas diarias).

1.2 Herramienta para la Migración y Administración de Servicios Telemáticos

Herramienta que permite administrar los servidores de forma remota, la cual tiene las funcionalidades necesarias para la gestión de usuarios, tareas programadas y servicios, con el objetivo de hacer posible la administración de conexiones a través del protocolo *SSH*[16] *(del inglés Secure Shell),* y así como la del servicio.

1.2.1 Arquitectura del sistema

La arquitectura que presenta HMAST propone el diseño de una arquitectura N-Capas orientada al dominio, compuesta por cinco capas, como se muestra en la Ilustración 2, las cuales son descritas a continuación.

- **Capa Presentación (presentation).**

Presenta al usuario los conceptos del negocio mediante una interfaz de usuario. Además, facilita la explotación de los procesos de administración que lleva a cabo el usuario, informa sobre la situación de los procesos de negocio e implementación de las reglas de validación de dicha interfaz. En esta capa se realiza el tratamiento a las excepciones lanzadas desde capas inferiores.

- **Capa Aplicación (*application*)**

Es una capa delgada, comúnmente solo realizará llamadas a servicios de capas inferiores (Dominio), los servicios que publica (servicios de aplicación) tienen la responsabilidad, por ejemplo, de adaptar la información que llega a los requerimientos de los servicios de dominio.

- **Capa Dominio (*domain*)**

Constituye el hilo conductor de la aplicación, sus componentes solo dependen de la Capa Infraestructura transversal (*infrastructureCrosscutting*) (están totalmente desacoplados). Implementa la lógica de dominio (reglas de negocio), es responsable de las validaciones. Define las interfaces de persistencia a datos (contratos de repositorio), pero no los implementa. Sus componentes no están ligados a tecnologías específicas.

- **Capa Persistencia (*persistens*)**

[16] *Es un protocolo usado para acceder y controlar máquinas remotas a través de una red.*

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

- **Capa Infraestructura transversal (*infrastructureCrosscutting*)**

Las responsabilidades de esta capa están dadas a promover la reutilización de código, ella albergará las operaciones de seguridad, logging, monitoreo del sistema, mecanismos de persistencia reutilizables, validadores genéricos, o sea, todas aquellas operaciones que sea posible llamar desde otras capas.

Ilustración 2: Arquitectura de HMAST.

1.2.2 Funcionalidades que brinda

- Gestión de servidores lógicos: permite la adición, edición, eliminación de los datos de un servidor lógico, además permite la conexión y desconexión remota a un servidor seleccionado.
- Gestión de servicios telemáticos asociados a un servidor lógico: Permite la adición, edición, eliminación de los datos de un módulo, así como activación y desactivación de los mismos.

- Gestión de las variables de configuración asociadas a un servidor lógico: Permite cargar y salvar las variables de configuración de los servicios telemáticos encontrados en un servidor lógico (ficheros de configuración, nombre de módulos, demonios, entre otros).

1.2.3 Consideraciones para implementar un módulo para HMAST

Para integrar un módulo a HMAST se deben tener en cuenta un conjunto de consideraciones, las cuales se especifican a continuación.

- La lógica de Aplicación no deberá incluir ninguna lógica del Dominio, solo tareas de coordinación relativas a requerimientos técnicos de la aplicación, como conversiones de formatos de datos de entrada a entidades del Dominio y llamadas a componentes de Infraestructura para que realicen tareas complementarias.
- La capa de Presentación no debe tener como entrada o salida, objetos de dominio, sino objetos para la transferencia de datos (DTO).
- Las entidades solo pueden tener dependencias de componentes de la capa de Dominio.
- Las clases de servicios deben ser las únicas responsables de acceder a los repositorios, no se puede implementar código de persistencia a datos en la capa de Dominio.
- Solo se puede acceder a la información almacenada en los servidores haciendo uso de los repositorios.
- El código reutilizable por más de un repositorio debe estar disponible en la capa de Infraestructura Transversal.

1.3 Estudio de elementos homólogos

Han sido desarrolladas aplicaciones para la administración de servidores de red. Estas aplicaciones facilitan el trabajo de los administradores en gran medida pues no hay necesidad de ir directamente al archivo de configuración, este puede ser configurado desde una interfaz. Algunas de estas son:

Zentyal

Zentyal (anteriormente conocido como *eBox Platform*) es un servidor de red unificada de código abierto (o una plataforma de red unificada) para las *PYMEs*[17]. Zentyal puede actuar gestionando la infraestructura de red como puerta de enlace a internet (*Gateway*), gestionando las amenazas de seguridad (UTM), como servidor de oficina, como servidor de comunicaciones unificadas o una combinación de estas. Además,

[17] *PYMEs: Acrónimo de Pequeñas y Medianas Empresas.*

Zentyal incluye un marco de desarrollo (un framework) para facilitar el desarrollo de nuevos servicios basados en Unix (11).

Para la administración del servicio proxy este servidor utiliza Squid, permitiendo las siguientes configuraciones:

- Definir si el proxy funcionará en modo transparente.
- Filtrado de páginas web según su contenido.
- Implementar un límite flexible para controlar el ancho de banda que consumen los usuarios.
- Configuración de los parámetros básicos de administración del servicio proxy como puerto de escucha, caché, ACL, validación de usuario.

El módulo de Zentyal para la administración del servicio proxy presenta, como todos los demás módulos, una interfaz intuitiva. Sin embargo, este servidor no reconoce y sobrescribe las configuraciones establecidas en el fichero principal de configuración de Squid, de manera que no permitirá realizar configuraciones que no estén establecidas en el módulo como las comunicaciones entre proxy (11).

Webmin

Webmin es una interfaz web escrita en Perl para administrar un servidor. Con esta herramienta se pueden configurar los permisos para usuarios y grupos o configurar el funcionamiento del servidor Mysql Server, cuotas de espacio, servicios, archivos de configuración, apagado del equipo. Webmin tiene una estructura por módulos para administrar una amplia variedad de servicios como Apache, PHP, MySQL, DNS, Samba, DHCP, Proxy, entre otros, además de ser completamente configurables y de poder crear nuevos módulos (12). Este servidor administra el servicio proxy a través de Squid, permitiendo las siguientes configuraciones:

- Establecer las configuraciones básicas y avanzadas para el correcto funcionamiento del servicio como puerto de escucha, parámetros de la caché reglas de acceso.
- Sincronización del servicio con el cortafuego del sistema.
- Administrar opciones de registros del sistema.
- Reconoce los cambios realizados en el fichero principal de configuración.

El módulo de administración del servicio proxy de Webmin cuenta con múltiples opciones de configuración; sin embargo, está diseñado con una interfaz poco intuitiva por los numerosos campos de configuración que presenta en una misma página. Otro elemento a destacar es que no permite la configuración del sistema para que funcione como proxy transparente ni proxy inverso, además de no permitir la configuración de la comunicación entre proxy (12).

Smart Keeper

Smart Keeper es un filtro de contenido web que permite aplicar la política de uso aceptable de internet (AUP) de una institución para regular el acceso de los usuarios a la red de redes. Está basado en el uso e integración del servidor proxy caché Squid, el servidor web Apache2, el servidor de base de datos PostgreSQL y otros componentes libres (13) . Este filtro de contenido web "entre sus características" posibilita:

- Instalar y configurar servidor de antivirus Clamav.
- Instalar y configurar PostgreSQL y crear la base de datos del sistema.
- Instalar y configurar Squid3, la ACL externa redir y el servidor de logs syslog-ng.
- Instalar y configurar el servidor ICAP GreasySpoon (13).

Smart Keeper cuenta con múltiples opciones de configuración; sin embargo, la administración del filtro es a través de un servidor de logs.

Cada uno de los elementos analizados cuenta con importantes características para la administración de un proxy, estas características serán usadas para la obtención de requisitos y el diseño de funcionalidades similares; sin embargo, estos no pueden ser integrados a HMAST debido a su arquitectura.

1.4 Metodología, herramientas y tecnologías

Teniendo en cuenta que el módulo que se va a desarrollar debe ser integrado a una herramienta (HMAST), la metodología, las herramientas y las tecnologías empleadas se corresponden con las mismas de la herramienta principal.

Metodología de desarrollo de software

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado. Estas se clasifican en dos tipos: ágiles y tradicionales (14). Para el desarrollo del módulo proxy de HMAST se emplea la metodología AUP-UCI, resultante de una variación de la metodología ágil AUP[18] *(del inglés Agile Unified Process)*.

La metodología AUP-UCI define 3 fases, 7 disciplinas y 11 roles. Las fases son: inicio, ejecución y cierre.

[18]*AUP: Proceso Unificado Ágil.*

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planificación. En la segunda fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. En la fase de cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades normales de cierre del proyecto. Los roles definidos por esta metodología son: jefe de proyecto, planificador, analista, arquitecto de información (opcional), desarrollador, administrador de la configuración, cliente/proveedor de requisitos, administrador de calidad, probador, arquitecto de sistema y administrador de base de datos. Además, AUP-UCI define cuatro escenarios para modelar el sistema. Se emplea el escenario número 4, el cual indica que proyectos que no modelen negocio solo pueden modelar el sistema con HU (14).

Lenguaje de programación

Los lenguajes de programación son un conjunto de reglas, herramientas y condiciones que permiten crear programas o aplicaciones dentro de un ordenador. Permiten además ordenar distintas acciones a la computadora en un lenguaje entendible por esta. Cada lenguaje posee su parte sintáctica y semántica, que no son más que el conjunto de reglas acerca de cómo se deben escribir las instrucciones y de qué forma (15).

Java

Java es un lenguaje de programación creado por Sun Microsystems basado en C y C++, aunque es un lenguaje mucho más sencillo y simple. El concepto de objeto es parte integral de Java, convirtiéndolo en un lenguaje orientado a objetos. Estos últimos combinan los valores con las funciones que operan sobre ellos. Posee una arquitectura neutra y funciona con uniformidad en diferentes sistemas operativos. La precisión y los formatos para los tipos de datos primitivos se especifican por completo, para asegurar que los programas de Java funcionen igual en todas las implantaciones. También posee otras características: el compilado genera ficheros de clases compiladas, pero estas clases compiladas son en realidad interpretadas por la máquina virtual de Java, siendo la máquina virtual de Java 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 sistema operativo que tenga instalada la máquina virtual 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 (16).

Framework Spring

Un *framework* (marco de trabajo) es un conjunto de bibliotecas, módulos o artefactos reutilizables que establecen una estructura que brinda soporte a los desarrolladores. Provee las piezas necesarias, las

cuales deben ser personalizadas y conectadas para desarrollar la aplicación. Representa un conjunto de consideraciones especiales que garantizan la calidad del diseño del mismo (17).

Spring Framework ofrece una programación completa y el modelo de configuración para las modernas aplicaciones empresariales basadas en Java en cualquier tipo de plataforma de despliegue. Un elemento clave de *Spring* es el apoyo de infraestructura a nivel de aplicación, este *framework* se centra en la "fontanería" de las aplicaciones empresariales a fin de que los equipos pueden centrarse en la lógica de negocio a nivel de aplicación, sin ataduras innecesarias a los entornos de despliegue específico.

La Inversión de Control (IoC del inglés *Inversion of Control*) y la Inyección de Dependencias (DI del inglés *Dependency Injection*) se encuentran entre las principales características de *Spring,* y están estrechamente relacionadas entre ellas. IoC es un patrón de diseño de software en el cual un ensamblador, en este caso *Spring Framework*, realiza el acoplamiento entre objetos en tiempo de ejecución en vez de en tiempo de compilación. Permite que los desarrolladores programen un conjunto de interfaces, las cuales puedan ser intercambiadas en diferentes ambientes y sin que sea necesario volver a compilar el código. La DI es la técnica más común para realizar este proceso. Usando DI, una porción del programa, es decir una clase, se declara que depende de una interfaz y en tiempo de ejecución *Spring Framework* es el encargado de realizar la inyección de la instancia de esa dependencia (17).

Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE) es un programa que permite a los desarrolladores escribir y ejecutar código a través de un lenguaje de programación y un conjunto de herramientas. Estos pueden soportar más de un lenguaje de programación o uno específicamente.

- **IntelliJ Idea**

Es un IDE para el lenguaje de programación Java. Es ligero en el diseño y ofrece características útiles como pruebas de *JUnit*[19], depuración, inspección de código y soporte para múltiple refactorización. IntelliJ Idea se encuentra enfocado en elevar la productividad de los desarrolladores, por lo que ofrece soporte avanzado para los *frameworks* y estándares más importantes en el desarrollo web como son Spring Framework, Vaadin, Play, Grails, Web Services y Struts. Además, incluye la asistencia de código para lenguajes como HTML, CSS, JavaScript, Node.js y TypeScript. Favorece la integración que tiene el IDE con herramientas para la construcción de proyectos como Maven, Ant y Gradle; y para el control de versiones como Git, SVN y Mercurial. Incorpora también un editor de base de datos SQL con soporte completo SQL

[19] *Es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.*

para Oracle, PostgreSQL, MySQL y SQL Server (18).

Herramienta de Ingeniería de Software Asociada por Computadora

La ingeniería de software asistida por computadora (o CASE) es un conjunto de herramientas de programación que utilizan una interfaz común para diseñar, desarrollar y depurar software. Por lo tanto, un entorno CASE consta de herramientas que proveen un modelo visual de una aplicación, herramientas que crean un código a través de interfaces visuales y finalmente un depurador para probar el código final (19). Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para ingenieros de software, analistas y arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos.

Visual Paradigm también ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Ambiente visualmente superior de modelado.
- Sofisticado diagramador automáticamente de layout.
- Sincronización de código fuente en tiempo real u *on deman* (19).

Lenguaje Unificado de Modelado

UML son las siglas de *"Unified Modeling Language"* o "Lenguaje Unificado de Modelado". Es uno de los estándares *Object Management Group* (OMG) más usados y se ha adoptado a nivel internacional por numerosos organismos. Diseñado para visualizar, especificar, construir y documentar software orientado a objetos. UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, por lo que cuenta con reglas para combinar tales elementos (20).

Herramienta para la gestión y construcción de proyectos

Maven es una herramienta de software de código abierto para la gestión y construcción de proyectos basada en estándares. Permite gestionar el ciclo de vida de un proyecto desde su creación hasta la generación de un binario que pueda distribuirse con este. Ofrece facilidades a los desarrolladores como la sencilla y ágil creación de proyectos o módulos, además de aplicar una estandarización de la estructura y organización del proyecto. También dispone de un mecanismo de gestión de dependencias de un proyecto sobre las bibliotecas propias o de terceros y mantiene disponible para los desarrolladores un repositorio de

bibliotecas de código abierto en constante actualización.

Sistema de Control de Versiones

Un sistema de control de versiones es una herramienta que registra todos los cambios que han tenido lugar en uno o más proyectos, guardando así versiones del producto en todas sus fases del desarrollo. Las versiones son como fotografías que registran su estado en ese momento del tiempo y se van guardando a medida que se hacen modificaciones al código fuente. En el proyecto en el cual se desarrollará el módulo se utiliza la herramienta Git. Esta herramienta fue lanzada en el año 2005. Git modela sus datos como un conjunto de instantáneas de un mini sistema de archivos (21).

Conclusiones parciales

El estudio realizado en el capítulo permitió definir la metodología, herramientas y los lenguajes que se utilizaran en el desarrollo de la propuesta de solución, por lo que se puede concluir lo siguiente:

- El estudio del estado del arte de los sistemas homólogos existentes relacionados con la propuesta de solución evidenció que no cumplían con las necesidades requeridas para ser integradas con HMAST, pero si se pueden utilizar para la captura de requisitos.
- El estudio de los servicios y sub-servicios asociados a un proxy permitió definir que se utilizará Squid como servidor proxy, E2guardian como servidor de filtrado de contenido y Squish como herramienta para la gestión de cuotas de usuarios.
- La selección y estudio de la base tecnológica para el desarrollo de la propuesta de solución permitió definir AUP-UCI como metodología de desarrollo de software para guiar este proceso. Se seleccionaron como herramientas a utilizar para el desarrollo de la propuesta de solución IntelliJIDEA en su versión 14.0.3 como entorno de desarrollo integrado, Visual Paradigm 8.0 como herramienta CASE para el modelado de los artefactos del análisis, UML como lenguaje de modelado, Spring en su versión 3.2.3 como framework de desarrollo, Java 1.8 como lenguaje de programación, Maven como herramienta para la construcción de proyectos, y Git para el control de versiones.

Capítulo 2. Análisis y diseño

Tomando como punto de partida las características del sistema base HMAST y los elementos planteados por la metodología AUP-UCI, se describe la propuesta de diseño del módulo a desarrollar. Para ello, se especifican las funcionalidades del sistema mediante los requisitos funcionales y las historias de Usuario. Además, se describe la arquitectura y los patrones de diseño que se emplean.

2.1 Propuesta de solución

En la presente investigación se propone el desarrollo de un módulo con las funcionalidades que permitan, desde HMAST, administrar el servicio proxy de forma centralizada. El módulo a implementar contendrá 3 sub-módulos, los cuales permitirán la administración del servicio, estableciendo las principales configuraciones generales y avanzadas, utilizando E2guardian como tecnología para el filtrado de contenido, y Squish para una correcta gestión de las cuotas de usuario, estas tecnologías serán integradas sobre Squid como servidor proxy. Resulta importante esclarecer, que los servicios antes mencionados no necesitan estar instalados en la misma PC, pero en caso de que esto suceda, es de vital importancia para el correcto funcionamiento del proxy establecer las configuraciones necesarias para la comunicación entre ellos. En la sección 1.1 se explican elementos conceptuales y técnicos necesarios para la comprensión y la correcta realización de los módulos especificados.

2.2 Artefactos generados

El proceso de desarrollo es guiado por la metodología AUP-UCI. Teniendo en cuenta que no se modela el negocio y se ajusta por tanto al escenario 4 que establece esta metodología, las funcionalidades se describen en el documento de Especificación de Requisitos de Software. Estos requisitos son encapsulados mediante Historias de Usuario, lo cual constituye el principal artefacto generado durante el diseño del módulo.

2.2.1 Especificación de requisitos

El Glosario de Terminología Estándar de Ingeniería de Software[20] define al requisito como una condición que necesita un usuario para resolver un problema o lograr un objetivo. También como una capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato,

[20] *Es una asociación mundial de ingenieros dedicada a la estandarización y el desarrollo en áreas técnicas.*

estándar, u otro documento impuesto formalmente (22). Estos se clasifican en funcionales y no funcionales. Los requisitos funcionales describen las funciones que el software debe ejecutar, con el objetivo de establecer un entendimiento común entre el usuario y el proyecto de software. La siguiente tabla muestra el listado de los requisitos funcionales del módulo, clasificados según su prioridad en alta, media y baja.

Tabla 6: Listado de requisitos funcionales del módulo Squid.

No.	Nombre del requisito funcional	Descripción	Prioridad
RF1	Instalar servicio para la navegación en internet	Permite instalar el servidor proxy a partir de la herramienta Squid.	Alta
RF2	Desinstalar servicio de navegación en internet	Permite desinstalar el servicio de proxy a partir de la herramienta Squid.	Alta
RF3	Mostrar configuración general	Permite mostrar la configuración general con los parámetros básicos para un correcto funcionamiento del servidor proxy.	Alta
RF4	Modificar configuración general	Permite modificar la configuración general del servidor proxy.	Alta
RF5	Mostrar listas de control de acceso	Muestra las ACLs del archivo squid.conf.	Media
RF6	Adicionar lista de control de acceso	Permite adicionar ACL al archivo squid.conf.	Alta
RF7	Modificar lista de control de acceso	Permite modificar ACL del archivo squid.conf.	Media
RF8	Eliminar lista de control de acceso	Permite eliminar ACL del archivo squid.conf.	Baja
RF9	Mostrar listado de reglas de control de acceso	Permite mostrar un listado con las reglas de control de acceso del archivo squid.conf.	Media
RF10	Adicionar regla de control de acceso	Permite adicionar regla al archivo squid.conf.	Alta
RF11	Modificar regla de control de acceso	Permite modificar regla del archivo squid.conf.	Media
RF12	Eliminar regla de control de acceso	Permite eliminar regla del archivo squid.conf.	Baja
RF13	Modificar configuración para la autenticación	Permite configurar autenticación por directorio activo del archivo squid.conf.	Media
RF14	Iniciar servicio para la navegación en internet	Permite iniciar el servicio para la navegación en internet usando la herramienta Squid.	Media
RF15	Reiniciar servicio para la navegación en internet	Permite reiniciar el servicio para la navegación en internet usando la herramienta Squid.	Media

RF16	Detener servicio para la navegación en internet	Permite detener el servicio para la navegación en internet usando la herramienta Squid.	Media
RF17	Aplicar cambios	Permite aplicar los cambios realizados en el servidor.	Alta
RF18	Descartar cambios	Permite descartar los cambios realizados.	Alta

Tabla 7: Listado de requisitos funcionales del módulo Squish.

No.	Nombre del requisito funcional	Descripción	Prioridad
RF19	Instalar el complemento para la gestión de cuotas	Permite instalar el complemento para la gestión de cuotas a partir de la herramienta Squish.	Alta
RF20	Desinstalar el complemento para la gestión de cuotas	Permite desinstalar el complemento para la gestión de cuotas a partir de la herramienta Squish.	Alta
RF21	Mostrar listado de cuotas de usuario	Permite mostrar las cuotas de usuarios en el archivo de configuración squish.conf.	Media
RF22	Adicionar cuota a un usuario	Permite adicionar cuota a un usuario determinado en el archivo de configuración squish.conf.	Alta
RF23	Modificar cuota de un usuario	Permite modificar cuota de un usuario determinado en el archivo de configuración squish.conf.	Media
RF24	Eliminar cuota de un usuario	Permite eliminar cuota de un usuario determinado en el archivo de configuración squish.conf.	Baja
RF25	Establecer período de actualización de cuota	Permite establecer el período de actualización de cuotas en el archivo de sistema crontab.	Baja
RF26	Aplicar cambios	Permite aplicar los cambios realizados al servidor.	Alta
RF27	Descartar cambios	Permite descartar los cambios realizados.	Alta

Tabla 8: Listado de requisitos funcionales del módulo E2guardian.

No.	Nombre del requisito funcional	Descripción	Prioridad
RF28	Instalar servicio de filtrado de contenido	Permite instalar el servicio de filtrado de contenidos a partir de la herramienta E2guardian.	Alta
RF29	Desinstalar servicio de filtrado de contenido	Permite desinstalar servicio de filtrado de contenido a partir de la herramienta E2guardian.	Alta
RF30	Modificar configuración del servicio de filtrado de contenido	Permite configurar la red y el idioma	Alta

		ubicados en el archivo de configuración *E2guardian.conf*.	
RF31	Mostrar listado de extensión de archivo a excluir	Permite mostrar un listado con la extensión del archivo a excluir en el archivo *exceptionextensionlist*.	Alta
RF32	Adicionar extensión de archivo a excluir	Permite adicionar extensión de archivo a excluir en el archivo *exceptionextensionlist*.	Media
RF33	Eliminar extensión de archivo a excluir	Permite eliminar la extensión del archivo a excluir en el archivo *exceptionextensionlist*.	Baja
RF34	Mostrar listado de sitio excluir	Permite mostrar un listado de los sitios a excluir en el archivo *exceptionsitelist*.	Alta
RF35	Adicionar sitio a excluir	Permite adicionar sitio a excluir en el archivo *exceptionsitelist.*	Media
RF36	Eliminar sitio a excluir	Permite eliminar sitio a excluir en el archivo *exceptionsitelist.*	Baja
RF37	Mostrar listado de dirección ip a excluir	Permite mostrar un listado de las direcciones ip a excluir del archivo *exceptioniplist.*	Alta
RF38	Adicionar dirección ip a excluir	Permite adicionar dirección ip a excluir en el archivo *exceptioniplist*.	Media
RF39	Eliminar dirección ip a excluir	Permite eliminar dirección ip a excluir en el archivo *exceptioniplist*.	Baja
RF40	Mostar listado de frases a excluir	Permite mostrar un listado de frases a excluir en el archivo *exceptionphraselist.*	Alta
RF41	Adicionar frase a excluir	Permite adicionar frase a excluir en el archivo *exceptionphraselist.*	Media
RF42	Eliminar frase a excluir	Permite eliminar frase a excluir en el archivo *exceptionphraselist.*	Baja
RF43	Mostrar listado de extensión de archivo a denegar	Permite mostar un listado de extensiones de archivo a denegar en el archivo *bannedextensionlist.*	Alta
RF44	Adicionar extensión de archivo a denegar	Permite adicionar extensión de archivo a denegar en el archivo *bannedextensionlist.*	Media
RF45	Eliminar extensión de archivo a denegar	Permite eliminar extensión de archivo a denegar en el archivo *bannedextensionlist.*	Baja
RF46	Mostrar listado de sitio a denegar	Permite mostrar un listado de sitios a denegar en el archivo *bannedsitelist*.	Alta
RF47	Adicionar sitio a denegar	Permite adicionar sitio a denegar en el archivo *bannedsitelist*.	Media
RF48	Eliminar sitio a denegar	Permite eliminar sitio a denegar en el archivo *bannedsitelist*.	Baja
RF49	Mostar listado de dirección ip a denegar	Permite mostar un listado de dirección ip a denegar en el archivo *bannediplist.*	Alta

RF50	Adicionar dirección ip a denegar	Permite adicionar dirección ip a denegar en el archivo *bannediplist.*	Media
RF51	Eliminar dirección ip a denegar	Permite eliminar dirección ip a denegar en el archivo *bannediplist.*	Baja
RF52	Mostar listado de frase a denegar	Permite mostar un listado de frase a denegar en el archivo *bannedphraselist*.	Alta
RF53	Adicionar frase a denegar	Permite adicionar frase a denegar en el archivo *bannedphraselist*.	Media
RF54	Eliminar frase a denegar	Permite eliminar frase a denegar en el archivo *bannedphraselist*.	Baja
RF55	Iniciar servicio de filtrado de contenidos	Permite iniciar el servicio de filtrado de contenidos a partir de la herramienta E2guardian.	Alta
RF56	Reiniciar servicio de filtrado de contenidos	Permite reiniciar el servicio de filtrado de contenidos a partir de la herramienta E2guardian.	Media
RF57	Detener servicio de filtrado de contenidos	Permite detener el servicio de filtrado de contenidos a partir de la herramienta E2guardian.	Baja
RF58	Aplicar cambios	Permite aplicar los cambios realizados al servidor.	Alta
RF59	Descartar cambios	Permite descartar los cambios realizados.	Alta

Los requisitos no funcionales se refieren a cualidades que imponen restricciones en el diseño y la implementación (22). La metodología AUP-UCI propone una taxonomía partiendo de la ISO 25010 donde se asocian estos requisitos a atributos de calidad. Diversos requisitos no funcionales son heredados de la herramienta HMAST, puesto que condicionan el funcionamiento del módulo para lograr una correcta integración con el sistema base. A continuación, se muestra el listado de los requisitos no funcionales.

Tabla 9: Listado de requisitos no funcionales.

No.	Nombre del requisito no funcional	Atributo de calidad	Descripción
RNF1	Emplear como lenguaje de programación Java	Funcionalidad	Estas son restricciones heredadas de HMAST, el módulo debe cumplirlas para poder integrarse correctamente.
RNF2	El módulo se ejecutará sobre el sistema operativo GNU/Linux Nova Servidores.		
RNF3	Disponer en el sistema operativo GNU/Linux de los siguientes paquetes: augeas-tools, libjna-java, openjdk-7-jdk.		
RNF4	Proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o modificar los mismos, y las personas o sistemas autorizados tengan el acceso a ellos.	Seguridad (Acceso restringido)	Todos los datos manejados por el módulo estarán encriptados; tanto los que se transfieren entre la estación cliente y el

			servidor HMAST como los que se almacenan temporalmente en el servidor.
RNF5	Permitir al usuario aprender su aplicación.	Usabilidad	Internacionalizar la información que se muestra, en los idiomas español e inglés.
RNF6	El módulo debe mantener un nivel de ejecución o desempeño especificado en caso de fallos del software o de infracción de su interfaz especificada.	Confiabilidad (Tolerancia a fallos)	Ante el fallo de una funcionalidad del sistema el resto de las funcionalidades que no dependen de esta, deberán seguir funcionando.

2.2.2 Historias de Usuario

Las Historias de Usuario especifican las tareas que debe realizar el sistema, lo que equivale a los casos de uso en el proceso unificado. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Además, guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo (22).

A continuación, se muestran algunas Historias de Usuario con gran relevancia, las demás se pueden encontrar en los anexos:

1. Instalar servicio para la navegación en internet.
2. Desinstalar servicio para la navegación en internet.
3. Modificar configuración general.

Tabla 10: Instalar servicio para la navegación en internet.

Número: HU_1	**Nombre del Requisito:** Instalar servicio para la navegación en internet
Programador: Alejandro Campos Palacios **Prioridad:** Alta	**Iteración Asignada:** 1 **Tiempo Estimado:** 5 horas
Riesgo en desarrollo: Inexperiencia del equipo técnico y de trabajo en el desarrollo e implementación del proyecto. Afectaciones en el rendimiento de los recursos humanos debido a problemas de salud. Caída de los servidores y estaciones de trabajo por fallas eléctricas.	**Tiempo Real:** 5 horas
Descripción: La funcionalidad comienza cuando el usuario selecciona en el menú de módulos la opción Administrar, posteriormente se comprueba que el servicio no esté previamente instalado, de no estarlo es redireccionado a otra página donde hay una breve descripción del servicio y en la esquina inferior derecha se encuentra la opción de instalar. Al terminar el proceso el servicio se instala y termina el proceso de instalación.	

Observaciones:
1. Es necesario antes de instalar realizar una actualización mediante la ejecución del comando: *apt update*.
2. Si el servicio no se encuentra instalado se podrá instalar, en caso contrario se permitirá desinstalar.
3. La acción de instalar el servicio Squid ejecutará el comando: *apt install squid.*
4. Por defecto se crearán configuraciones y estas quedarán guardadas en el fichero *squid.conf* situado en el directorio /etc/squid/ .
5. Por defecto se crearán configuraciones en el archivo de logs de acceso *access.log* que quedará alojado en el directorio /var/log/squid/.
6. También por defecto se definirá el puerto de escucha 3128, posteriormente este puerto podrá ser modificado.
7. Al finalizar el proceso de instalación se mostrará el mensaje de información: "El servicio ha sido instalado satisfactoriamente.". En caso contrario, el sistema mostrará un mensaje notificando al usuario los detalles del error o los errores que tuvieron lugar durante el proceso.

Prototipo de interfaz:

Tabla 11: Desinstalar servicio para la navegación en internet.

Número: HU_2	**Nombre del Requisito:** Desinstalar servicio para la navegación en internet
Programador: Alejandro Campos Palacios	**Iteración Asignada:** 1
Prioridad: Alta **Riesgo en desarrollo:** Inexperiencia del equipo técnico y de trabajo en el desarrollo e implementación del proyecto. Afectaciones en el rendimiento de los recursos humanos debido a problemas de salud. Caída de los servidores y estaciones de trabajo por fallas eléctricas.	**Tiempo Estimado:** 5 horas **Tiempo Real:** 5 horas

Descripción:
La funcionalidad comienza cuando el usuario selecciona en el menú de módulos la opción Administrar, esta opción despliega un submenú en el que se ha verificado con anterioridad si el servicio está previamente instalado, de estarlo se habilita la opción desinstalar servicio, y se procede a hacer la desinstalación, el servicio se desinstala y termina el proceso de desinstalación.

Observaciones:
1. El servicio Squid deberá estar instalado para poder desinstalarlo.
2. Al seleccionar la opción de desinstalar el servicio se mostrará el mensaje de advertencia: "¿Está seguro que desea desinstalar el servicio?".
3. La acción de desinstalar ejecutará el comando: *apt-get purge squid.*
4. Al desinstalar el servicio se borran los ficheros de configuración del directorio */etc/squid*.
5. Se notifica mediante un mensaje de información: "El servicio ha sido desinstalado satisfactoriamente.". En caso contrario, el sistema mostrará un mensaje notificando al usuario los detalles del error o los errores que tuvieron lugar durante el proceso.

Prototipo de interfaz:

Tabla 12: Modificar configuración general.

Número: HU_3	**Nombre del Requisito:** Modificar configuración general
Programador: Alejandro Campos Palacios	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 6 horas
Riesgo en desarrollo: Inexperiencia del equipo técnico y de trabajo en el desarrollo e implementación del proyecto. Afectaciones en el rendimiento de los recursos humanos debido a problemas de salud. Caída de los servidores y estaciones de trabajo por fallas eléctricas.	**Tiempo Real:** 6 horas

Descripción:

La funcionalidad comienza cuando el usuario selecciona en el menú de módulos la opción Administrar, posteriormente es redireccionado hacia la página principal, donde se encuentran las configuraciones básicas de red del servicio Squid, los parámetros que intervienen son:

Dirección IP*: Es un campo de texto obligatorio que se utiliza para identificar un dispositivo en una red. Estos dispositivos al formar parte de una red serán identificados mediante un número IP único en esa red, por ejemplo, una dirección IP puede ser 192.168.66.254. Los valores permitidos son cuatro números entre 0 y 255 separados por puntos. La dirección IP no viene definida por defecto, se establece en caso de ser necesario por cuestiones de seguridad.

Puerto*: Es uncampo de texto obligatorio que se utiliza para el acceso entre equipos para el uso de servicios y flujo de datos entre ellos. Squid, al ser instalado, por defecto utilizará el puerto 3128 para atender peticiones y no podrá utilizar puertos en uso, sin embargo, se puede especificar que lo haga en cualquier otro puerto o bien que lo haga en varios puertos a la vez. El puerto puede ser un número entre 1 y 65535, incluyendo ambos.

Formato de caché*: Define el formato que tendrá la caché que son [UFS] o [AUFS]. El formato de caché ufs puede llegar a bloquear el proceso principal de Squid en operaciones de entrada/salida sobre el sistema de archivos cuando hay muchos clientes conectados. Para evitar que esto ocurra, se recomienda utilizar aufs, que utiliza el mismo formato de ufs, pero funciona de manera asincrónica.

Directorio de caché*: Es un campo para definir la dirección donde se desea almacenar la caché. Debe permitir seleccionar el directorio donde se almacenará la caché.

Tamaño del directorio*: Es un campo de texto que se utiliza para definir el tamaño que tendrá el directorio donde se encontrará la caché (en megabytesMB). Se puede incrementar el tamaño del caché en el directorio hasta donde se desee. Mientras más grande sea el caché, más objetos se almacenarán en éste y por lo tanto se consumirá menos el ancho de banda. Es aconsejable que el tamaño de la caché tenga como valor mínimo de 256 y el valor máximo la mitad de la RAM de la máquina que utiliza el usuario.

Directorios subordinados *: Es un campo de texto que se utiliza para definir el número de subdirectorios que se creará bajo el directorio de memoria caché. El valor predeterminado es de 16 directorios subordinados en un primer nivel. Solo permitirá la entrada de números. Por defecto se establecen 16 directorios. Solo permitirá la entrada de números entre 1 y 64, incluyendo ambos.

Niveles*: Es un campo de texto que se utiliza para definir el número de niveles posibles dentro de cada uno de los subdirectorios que serán creados en cada uno de los niveles del directorio. Por defecto se establecen 256. Solo permitirá la entrada de números entre 1 y 1024, incluyendo ambos.

Memoria en caché*: Es un campo de texto que se utiliza para definir el tamaño que tendrán los objetos en caché.

Tamaño máximo de los objetos en caché*: Es un campo de texto que se utiliza para definir el tamaño máximo de los objetos en la caché (en megabytesMB), se recomienda establecerla en escenarios con alta carga de trabajo, puesto que permite evitar desperdiciar recursos de sistema almacenando en la caché, objetos de gran tamaño que probablemente sólo sean aprovechados por unos pocos usuarios. Por defecto se establecen 8 Mb. Solo permitirá la entrada de números entre 1 y 512, incluyendo ambos.

Algoritmo de caché*: Se utiliza para cuando se acaba el espacio de la caché de disco, el sistema debe decidir qué objetos de la caché desea eliminar para dar paso a los nuevos objetos entrantes. En Squid se utilizarán 3 políticas de reemplazo: LRU, LFUDA, GDSF. Squid utiliza por defecto el algoritmo LRU si no se indica lo contrario. Se debe seleccionar uno de los algoritmos siguientes: LRU: Acrónimo de Least Recently Used, que traduce como Menos Recientemente Utilizado. En este algoritmo los objetos que fueron accedidos hace mucho tiempo, son eliminados primero, manteniendo siempre en el caché a los objetos más recientemente solicitados. LFUDA: Least Frequently Used with Dynamic Aging, que se traduce como Menos Frecuentemente Utilizado con Envejecimiento Dinámico. Se trata de una modalidad en la que prima la cantidad de bytes de tráfico ahorrados a los clientes sobre la cantidad de hits sobre la caché. Se trata de un algoritmo que tiene en cuenta dos factores:

• Frecuencia de uso: A uso menos frecuente de un objeto web, más posibilidades de que este sea eliminado de la caché.

• Tamaño: A objeto de mayor tamaño, menos posibilidades de que este sea eliminado de la caché.

GDSF: Acrónimo de Greedy Dual Size Frequency. Se trata de un algoritmo de reemplazamiento donde prima la importancia sobre

la cantidad de objetos cargados directamente de la caché. Suele preferir almacenar gran cantidad de objetos pequeños en la caché que pocos y grandes, haciendo que el % de posibilidades de que el objeto deseado esté en la caché sea mayor.

Observaciones:

1. El servicio Squid deberá estar instalado para poder realizar las configuraciones de la red.
2. En el archivo de configuración squid.conf la directiva que se modifica para configurar la red es:

http_port <Dirección IP: # Puerto>. Ejemplo: http_port 192.168.1.254:8080. Debe considerarse que el puerto se pone en función de sus necesidades, por tanto, es importante un previo conocimiento de los puertos que son utilizados por los diferentes servicios.

3. Se deberá añadir una regla de iptables al Cortafuegos que permita el correcto funcionamiento del servicio, para esto se proveerán elementos como dirección de origen y destino, puertos de origen y destino, interfaz de entrada, de salida y protocolo. En este caso el IP y el Puerto definido es abierto para tráfico entrante y saliente. Estos datos se pasarán al módulo de cortafuegos que se encargará de generar la regla necesaria para que el servicio permita el acceso desde y hacia el mismo. En caso de cambiar algunos de estos campos se remplazará la regla antigua por una nueva con los campos correspondientes.
4. Si se dejan campos obligatorios vacíos se mostrará un mensaje de error indicando los campos que se dejaron vacíos.
5. Si se introduce algún parámetro en algún campo que no sea el permitido se mostrará un mensaje de error indicando los valores permitidos.
6. En el archivo de configuración squid.conf las directivas que se modifican para configurar la caché son:

cache_dir <Formato> <Directorio> <Tamaño> <Cantidad de directorios> <Subdirectorios>. Ejemplo: cache_dir aufs /var/spool/squid 100 16 256.

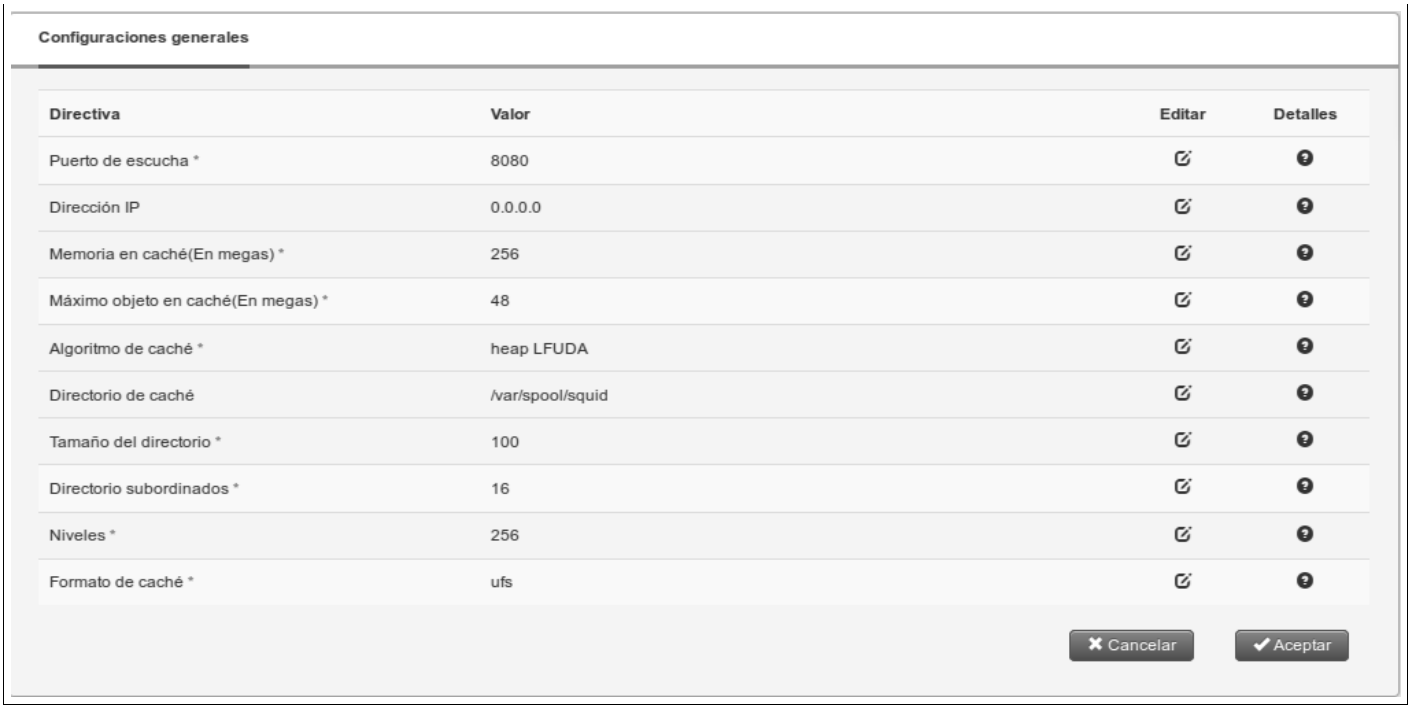
cache_mem <Tamaño de objetos en la cache>. Ejemplo: cache_mem 8 MB. Si se posee un servidor con al menos 128 MB de RAM, establezca 16 MB como valor para este parámetro.

maximum_object_size <Tamaño máximo de los objetos en la caché> Ejemplo: máximum_object_size 48MB.

memory_replacement_policy <Políticas de reemplazo>. Ejemplo: memory_replacement_policy LRU.

7.Para el parámetro Directorio, Squid utilizará el directorio por defecto /var/spool/squid. Este puede ser cambiado si se desea.
8. Para los parámetros Cantidad de directorios y Subdirectorios los números 16 y 256.

Prototipo de Interfaz de usuario:



2.3 Arquitectura del módulo

El proceso de diseño de la arquitectura debe decidir cuál funcionalidad es la más importante a desarrollar. Además, define cuáles son los componentes más básicos del sistema y cómo se relacionan entre ellos para implementar la funcionalidad (23). Se determina como arquitectura del módulo, la establecida para la herramienta base HMAST (ver Ilustración 3), para lograr la consistencia con los componentes del sistema. Se emplea una arquitectura N-Capas orientada al Dominio, la cual tiene como objetivo estructurar de forma clara la complejidad de una aplicación empresarial basada en las diferentes capas de la arquitectura siguiendo el patrón N-Capas y las tendencias de arquitecturas orientadas al dominio (23). En cada una de las capas: Persistencia, Dominio, Aplicación, y Presentación, se inserta un paquete de proxy que dentro contendrá 3 nuevos módulos los cuales son Squid, Squish y E2guardian. A continuación, se puede apreciar de manera más clara, en la ilustración 3.

Ilustración 3: Arquitectura de los módulos.

2.4 Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que se pueden encontrar a la hora de desarrollar una aplicación. Aunque una aplicación sea única, tendrá partes comunes con otras aplicaciones: acceso a datos, creación de objetos, operaciones entre sistemas. Se pueden solucionar problemas utilizando algún patrón, ya que son soluciones probadas y documentadas por multitud de programadores. Para la implementación del módulo a desarrollar se hizo uso de los patrones GRASP[21] *(del inglés General Responsibility Assignment Software Patterns)* y GoF[22] (24).

[21] *Patrones Generales de Software para Asignar Responsabilidades.*
[22] *Patrones de diseño de software, acrónimo de Gang o Four, Pandilla de los Cuatro, en su traducción al español*.

2.4.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales para asignar responsabilidades a los objetos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de *software* (25). A continuación, se describen los patrones que fueron utilizados en el desarrollo del módulo propuesto.

- **Experto en información o experto:** se utiliza para la asignación de responsabilidades relacionadas con la obtención de información. Conduce a diseños donde los objetos del *software* realizan aquellas operaciones que normalmente se hacen a los objetos inanimados del mundo real que representan. Se mantiene el encapsulamiento de la información logrando un bajo acoplamiento entre los objetos y se distribuye el comportamiento entre las clases, lo que estimula las definiciones de clases más cohesivas.
- **Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento. Su utilización se evidencia cuando en la clase SquidAppService “de la capa de Aplicación” se realiza la creación de los DTO utilizando para ello las entidades necesarias del dominio. Esta clase es la encargada de crear un DTO a partir de una entidad o crear la entidad a partir de un DTO.
- **Bajo acoplamiento:** consiste en asignar responsabilidades de manera que el acoplamiento permanezca bajo. El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos. El uso de este patrón permite la reutilización de las clases y que no se afecten por cambios que se realicen en otros componentes. Este patrón se emplea en las distintas capas del módulo mediante el uso de interfaces que relacionan una capa con otra de forma que dichas relaciones no se establezcan directamente hacia las clases. Las conexiones se realizan a través del mecanismo de inyección de dependencias. Esto se evidencia en el paquete RepositoryContrats, donde se define la interfaz ISquidRepository, y la implementación de sus métodos es realizada por la clase SquidRepository en la capa de Persistencia.
- **Alta Cohesión:** la cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Las clases tienen una responsabilidad moderada en un área funcional y colaboran con otras clases para llevar a cabo una tarea determinada. Permite que las clases sean fáciles de entender, mantener y reutilizar. Se emplea en la clase SquidRepository, que tiene la responsabilidad de interactuar con los objetos del fichero de configuración

y realizar operaciones de creación, lectura, actualización y eliminación. Para el cumplimiento de estas tareas utiliza los métodos loadLocalConfiguration, saveGeneralFormOptions, saveACL, saveRules, delegando en estos las operaciones de conversión de los atributos.

2.4.2 Patrones GoF

Los patrones GoF son patrones de diseño de software que solucionan problemas de creación de instancias, estos ayudan a encapsular y abstraer dicha creación.

- **Patrón Solitario (*Singleton*):** Garantiza la existencia de una única instancia para una clase. Es usado debido a la necesidad de trabajar con el mismo objeto en distintos momentos. Se hace uso del mismo en la aplicación para establecer la conexión con el servidor que se desee administrar, por lo que se hace una única instancia del objeto SSHConection, presente en el paquete infrastructure-Crosscutting.

2.5 Diagrama de clases por paquetes

El diagrama de clases por paquetes muestra las agrupaciones lógicas en que está dividido el sistema, así como las dependencias entre dichas agrupaciones. Para el diseño del diagrama de clases por paquetes se toma como referencia la arquitectura anteriormente propuesta, de modo que existe un paquete por cada capa, y dentro de cada uno de estos paquetes otro, llamado Squid, Squish, y E2guardian que contienen todas las clases pertenecientes al módulo. A continuación, se puede apreciar el diagrama de clases por paquete Squid, Squish, E2guardian.

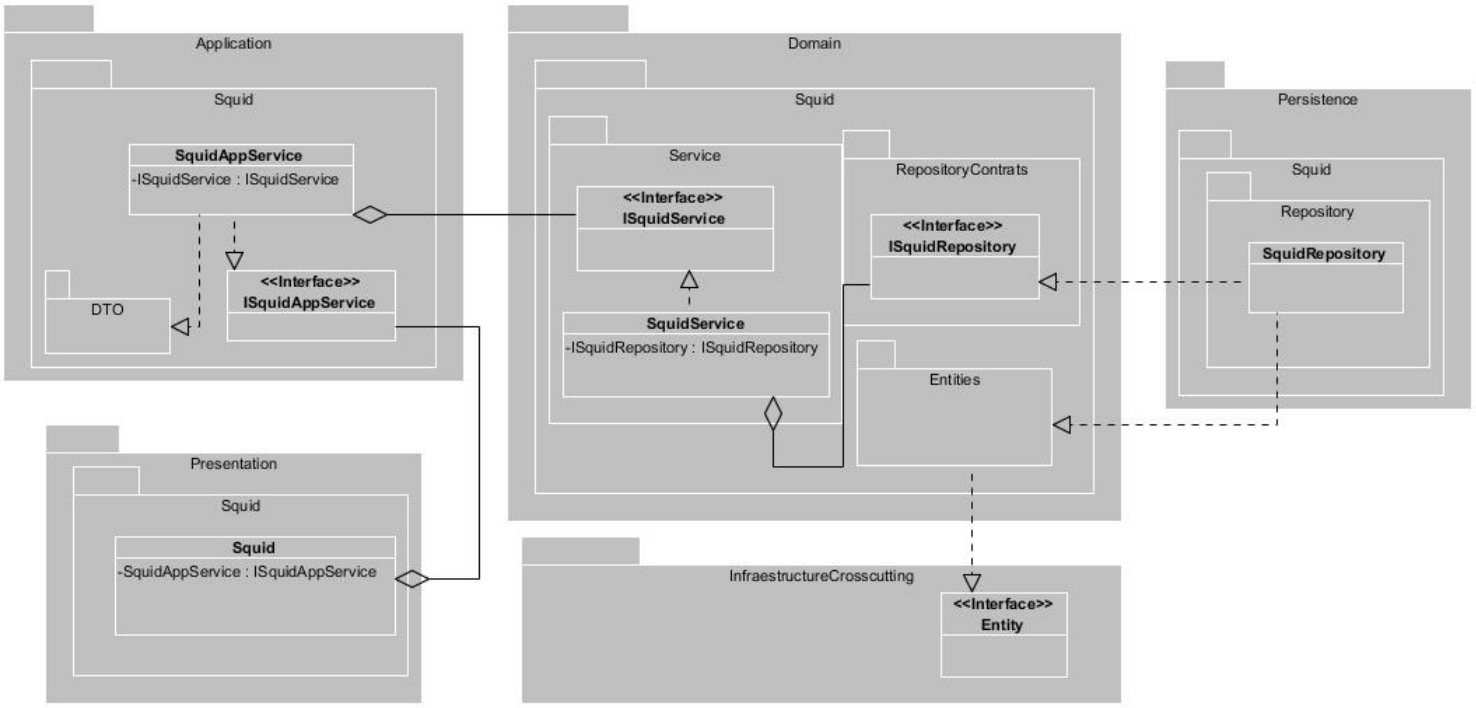


Ilustración 4: Diagrama de clases por paquetes Squid.

En la ilustración 5 se puede ver el diagrama de clases por paquete Squish.

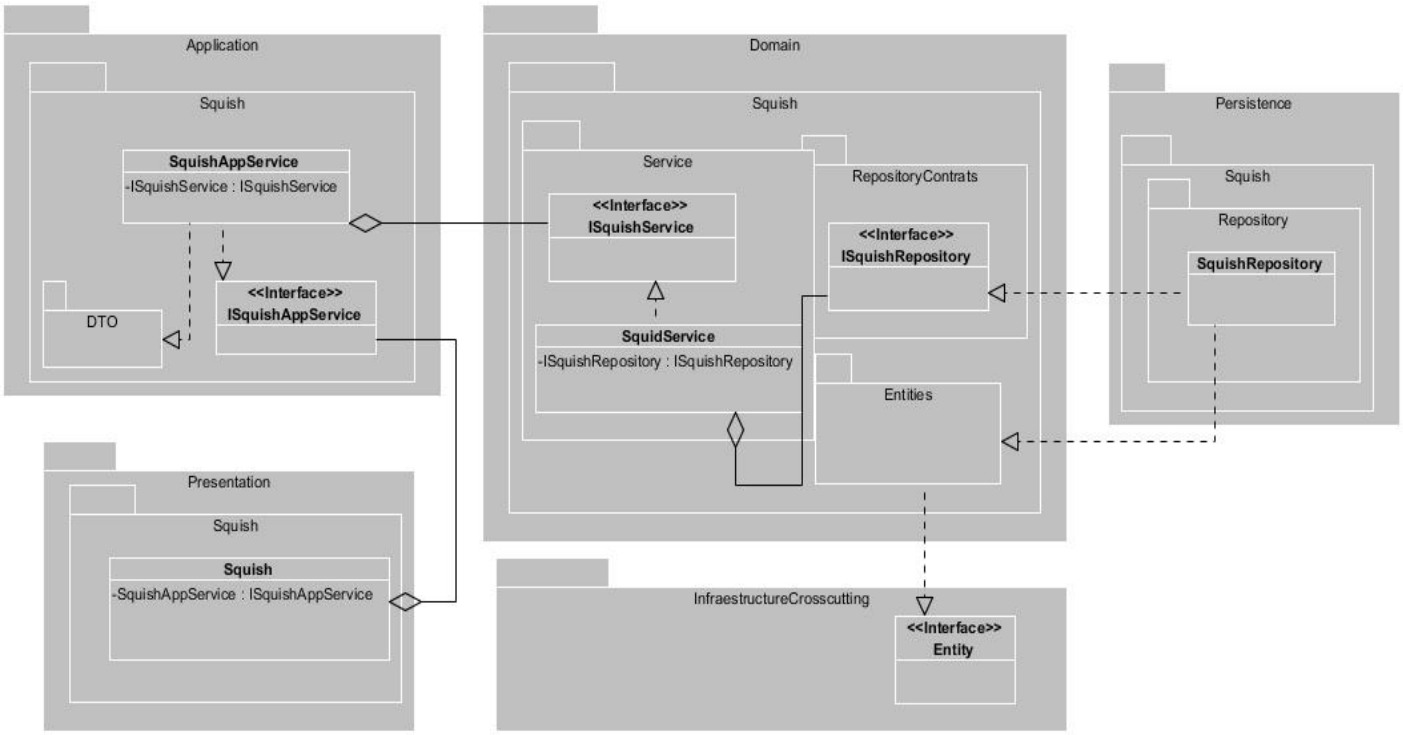


Ilustración 5: Diagrama de clases por paquetes Squish.

En la siguiente ilustración se puede ver el diagrama de clases por paquetes E2guardian.

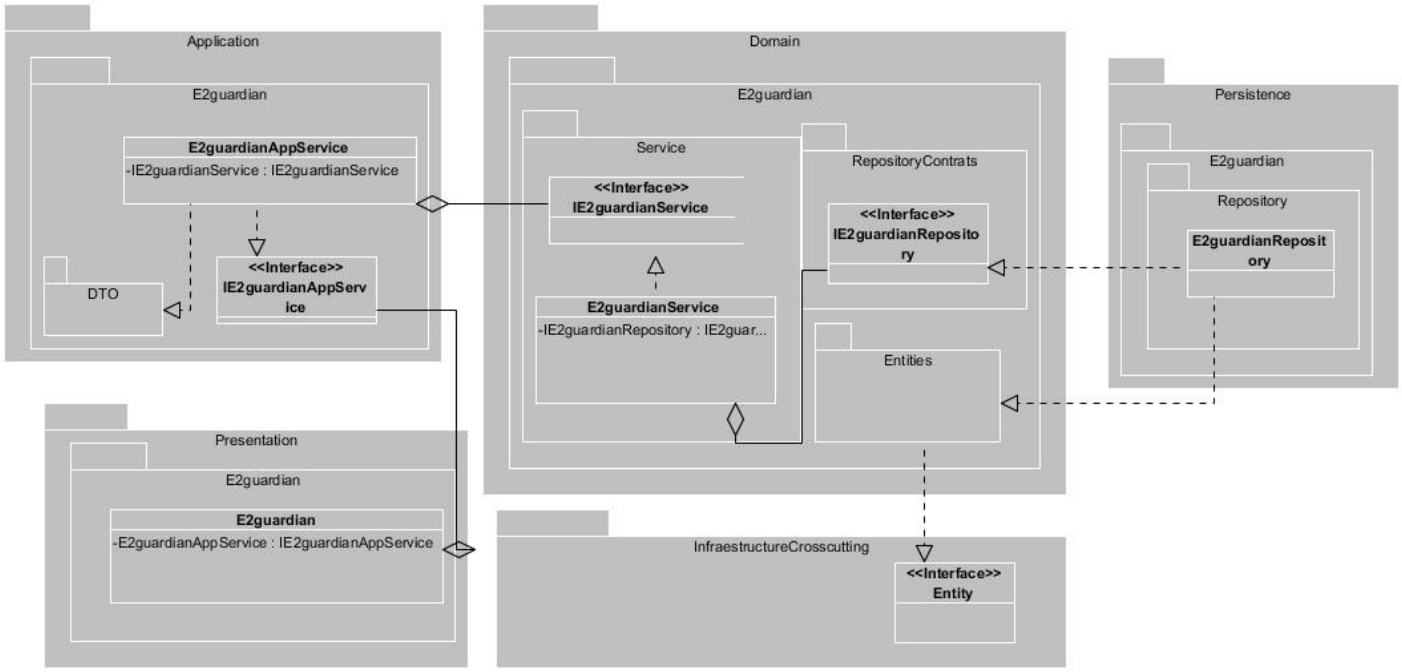


Ilustración 6: Diagrama de clases por paquetes E2guardian.

A continuación, se realiza una breve descripción del funcionamiento del módulo a través de los paquetes que este presenta y la organización de sus elementos. La capa de Aplicación (Application) está compuesta por el paquete Squid y dentro de este el paquete DTO.

En Squid se encuentra la interfaz ISquidAppService, en esta se encuentran los métodos que serán implementados en la clase SquidAppService. El paquete DTO se encarga de la transferencia de datos entre la capa de Aplicación y la capa de Dominio. La siguiente capa es la de Dominio (Domain), en esta capa se encuentra como subpaquete Squid, la que contiene a su vez tres paquetes nombrados Entities, Services y RepositoryContrats.

En el subpaquete Entities se encuentran las clases entidades (Entity) que son las contenedoras de toda la información referente al servicio Squid. El paquete Services contiene la clase ISquidService, que define métodos que son accedidos desde la capa de Aplicación, implementados por la clase SquidService, la cual realiza las validaciones de los datos antes de realizar las operaciones en el repositorio. Se encuentra, el paquete RepositoryContrats, en el que se definen los contratos de repositorios en ISquidRepository, pero no realiza su implementación.

La capa de Persistencia (Persistens) es la siguiente, en la que se encuentra el paquete Squid y dentro el

paquete Repository, en el que se encuentra la clase SquidRepository, la cual implementa a la clase ISquidRepository definidas en el subpaquete RepositoryContrats en el paquete domain, esta capa trabaja directamente con los ficheros, contiene el código necesario para persistir los datos, es decir, que los cambios se muestren al usuario en el propio momento en que se realizan.

Conclusiones parciales

A partir del estudio de los servicios proxy, y el análisis de los elementos homólogos se lograron definir:

- Los requisitos asociados al módulo, obteniéndose un total de 65 requisitos, de ellos 59 funcionales y 6 no funcionales asociados a la herramienta HMAST, los cuales fueron encapsulados en sus respectivas historias de Usuario según plantea la metodología AUP-UCI.
- Se definió la arquitectura, los patrones a utilizar y se conformaron los diagramas correspondientes al módulo.

Capítulo 3. Implementación y prueba

En este capítulo se muestra el estándar de codificación empleado para el desarrollo del módulo con el objetivo de conservar la analogía de la codificación de los restantes módulos que ya presenta la herramienta. Se establece la estrategia de pruebas y se documenta la realización de las pruebas seleccionadas. Se elabora, documentan y llevan a cabo los casos de prueba que se aplicarán sobre el módulo. Para comprender la distribución física de los componentes que conforman el sistema, también se modela el diagrama de despliegue.

3.1 Estándar de codificación establecido por HMAST

El estándar de codificación es utilizado para asegurar que todos los programadores del proyecto trabajen de forma coordinada (26). El estándar que se emplea en el módulo se ajusta a las pautas definidas en el expediente de proyecto de HMAST.

- **Asignación de nombres:** Emplear descriptores en inglés. Evitar nombres largos y que difieran en una letra o en el uso de mayúsculas. Para nombrar las funciones y variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se utiliza la notación CamelCase[23].
- **Ficheros de código fuente:** Cada fichero contiene una única clase o interfaz. Si hay una clase privada o una interfaz asociada a una clase pública se puede poner en el mismo fichero. La clase pública debe ser la primera.
- **Indentación:** La unidad de bloques de sentencias son 4 espacios.
- **Comentarios:** Los comentarios deben añadir claridad al código. Deben contar el por qué y no el cómo. Deben ser concisos.
- **Declaraciones:** Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.
- **Continuidad de las líneas largas:** Cuando una sentencia no quepa en una única línea se debe fraccionar después de una coma, después de un operador y alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea anterior.

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

- **Longitud de la línea:** Limitar todas las líneas a un máximo de 120 caracteres.
- **Nombres de componentes:** Todos los paquetes comienzan con cu.uci.hmast.xxx.yyy.zzz.kkk

xxx →presentation, application, domain, persistence.

yyy →nombre del módulo (squid).

zzz→elementos que pueden contener los componentes verticales (*entitys, repositorys*).

kkk → clases o subpaquetes.

3.2 Estrategia de pruebas

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba debidamente seleccionados, de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo (22).

La metodología AUP-UCI desagrega las pruebas en tres disciplinas: internas, liberación y aceptación. Las pruebas de liberación son diseñadas y ejecutadas por una entidad certificadora externa, por lo que en el presente trabajo no son analizadas. La estrategia de pruebas de software que se emplea propone pruebas de bajo nivel que incluyen todos los componentes del módulo. Esta estrategia tiene un enfoque incremental y se representa mediante una espiral que analiza el código y los requisitos, para lo cual se emplean pruebas de unidad, de integración, y las de aceptación. En la Ilustración 7 se representa la estrategia de pruebas descrita.

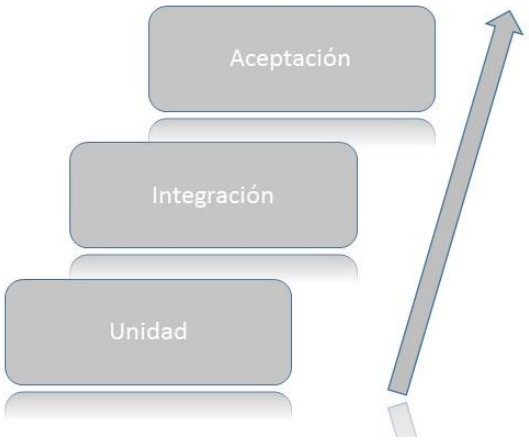


Ilustración 7: Estrategia de pruebas.

La prueba de unidad analiza cada módulo individualmente, asegurando que funciona adecuadamente

como una unidad, haciendo un uso intensivo de las pruebas de caja blanca, utilizando la técnica del camino básico para asegurar un alcance completo y una detección máxima de errores. La prueba de integración se enfoca al diseño y la construcción de la arquitectura del software, y las de aceptación son las pruebas realizadas por el cliente.

3.2.1 Prueba de unidad

Las pruebas de unidad son los procedimientos de pruebas locales a un módulo del sistema. Por definición dichas pruebas cubren la funcionalidad propia del módulo tanto con una perspectiva de caja blanca como de caja negra (22). La prueba de unidad que se realiza hace uso del método de caja blanca y de la técnica del camino básico. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (22).

La técnica se aplica a todos los métodos, pero a continuación solo se muestra la aplicación de la técnica al método *loadLocalOption.*

Enumeración del código.

1	public GeneralConfiguration loadLocalOption(LogicalServer server) throws EInvalidPath, ENotFoundUrlLocal, ENotFoundUrlRemoto, IOException, JSchException, NoSuchAlgorithmException, SQLException, SftpException, InterruptedException, EEntityExist, EInvalidEntity, EFileWithIncorrectSettingsForAugeas, EPortOutOfRange, EAddressOutOFRange {
2	IdentityGenerator id = new IdentityGenerator();
3	UUID idSquid = id.getNewUUID();
4	String baseDir = this.createRootFolderConfigurationModuleIfNotExist(server).getPath();
5	this.createPrincipalFileConfigIfNotExist(server);
6	Augeas augeas = augeasSingleton.getInstanceAugeas(baseDir);
7	String pathBase = "/files" + serversAdministrationService.getConfigurationValueByKey(server.getId(),canonicalName,"rootFolderConfiguration") + serversAdministrationService.getConfigurationValueByKey(server.getId(),canonicalName,"nameFilePrincipalConfig");
8	if(!augeas.exists(pathBase)){
9	Throw new EFileWithIncorrectSettingsForAugeas("squid.errors.found.when.loading.file.incorrect.settings.for.augeas",server

```
    sAdministrationService.getConfigurationValueByKey(server.getId(),canonicalName,"rootFolderConfiguration")  +
    serversAdministrationService.getConfigurationValueByKey(server.getId(),canonicalName,"nameFilePrincipalConf
    ig")) ;
10  }
11  String cacheMem = "256" ;
12  String cacheMemUnit = "MB" ;
13  String cacheMem_ = augeas.get(pathBase + "/cache_mem");
14   if(cacheMem_ != null
15  && !cacheMem_.equals("")){
16  cacheMem = cacheMem_.split(" ")[0] ;
17  cacheMemUnit = cacheMem_.split(" ")[1] ;
18  }
19  CacheMem cacheMemory = new CacheMem(Integer.parseInt(cacheMem), cacheMemUnit);
20  String maxObjSize = "48" ;
21  String maxObjectSizeUnit = "MB" ;
22  String maxObjSize_ = augeas.get(pathBase + "/maximum_object_size");
23  if(maxObjSize_ != null
24  && !maxObjSize_.equals("")){
25  maxObjSize = maxObjSize_.split(" ")[0] ;
26  maxObjectSizeUnit = maxObjSize_.split(" ")[1] ;
27  }
28  CacheMaxObjSize cacheMaxObjSize = new CacheMaxObjSize(Integer.parseInt(maxObjSize), maxObjectSizeUnit);
2   String port_ = augeas.get(pathBase + "/http_port");
```

```
29
30  String port="3128" ;
31  String localAddres = "0.0.0.0" ;
32  if (port_ != null
33  && !port_.equals("")) {
34  String[] temp = port_.split(":");
35  if (temp.length > 1) {
36  localAddres=temp[0];
37  port=temp[1];
38  }
39  else {
40  port=temp[0];
41  }
42  }
43  HttpPort httpPort = new HttpPort(new PortNumber(Integer.parseInt(port)),new
    Ip(localAddres),UUID.randomUUID()) ;
44  String cachepolicy = "heap LFUDA";
45  String cachepolicy_ = augeas.get(pathBase + "/cache_replacement_policy");
46  if (cachepolicy_ != null
47  && !cachepolicy_.equals("")) {
49  cachepolicy = cachepolicy_;
50  }
5   GeneralConfiguration config = new GeneralConfiguration(idSquid, httpPort, null, null, null, cacheMemory,
```

1	cachepolicy, cacheMaxObjSize);
52	return config;
53	}

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

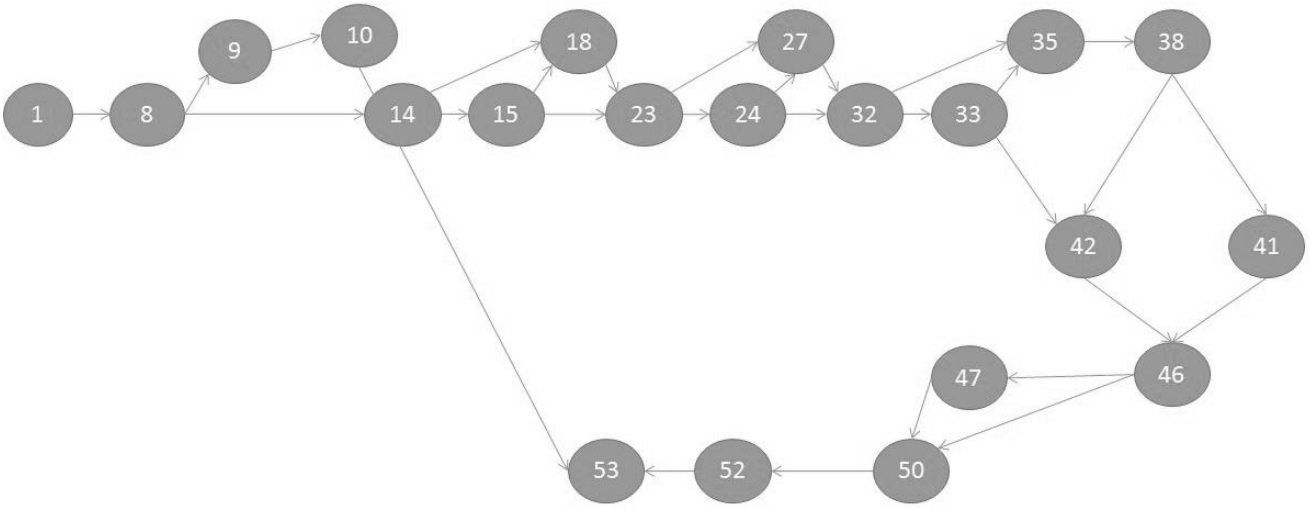


Ilustración 8: Diagrama de flujo de control.

Cálculo de la complejidad ciclomática

Siendo (a) el número de aristas del grafo de flujo y (n) el número de nodos del mismo, se define como la complejidad ciclomática (V(g)) como:

V(g) = a - n + 2

V(g) = 29 - 21 + 2

V(g) = 10

La complejidad ciclomática es 10, por lo que como mínimo se deberán diseñar 10 casos de prueba. Los caminos de pruebas elegidos son los siguientes:

Camino 1: [1-8-9-10-53]

Camino 2: [1-8-14-15-18-23-24-27-32-33-35-38-42-46-47-50-52-53]

Camino 3: [1-8-14-15-23-24-27-32-33-35-38-42-46-47-50-52-53]

Camino 4: [1-8-14-15-18-23-24-32-33-35-38-42-46-47-50-52-53]

Camino 5: [1-8-14-15-18-23-24-27-32-33-42-46-47-50-52-53]

Camino 6: [1-8-14-15-18-23-24-27-32-33-35-38-42-46-50-52-53]

Camino 7: [1-8-14-15-18-23-24-27-32-33-35-38-41-46-47-50-52-53]

Camino 8: [1-9-10-14-18-23-24-32-33-42-46-47-50-52-53]

Camino 9: [1-9-10-14-18-23-27-32-33-42-46-47-50-52-53]

Camino 10: [1-8-14-15-18-23-24-32-33-35-38-41-46-50-52-53]

Tabla 13: Ruta 1.

Caso de Prueba para el Camino: 1	
No. Ruta: 1	**Ruta:** 1-8-9-10-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: Si el fichero principal squid.conf no se encuentra en la carpeta /var/hmast/usersData/admin/nova/squid sistema lanza una excepción y finaliza la ejecución.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 14: Ruta 2.

Caso de Prueba para el camino: 2	
No. Ruta: 2	**Ruta:** 1-8-14-15-18-23-24-27-32-33-35-38-42-46-47-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema debe cargar todas las configuraciones del archivo de configuración.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 15: Ruta 3.

Caso de Prueba para el Camino: 3	
No. Ruta: 3	**Ruta:** 1-8-14-15-23-24-27-32-33-35-38-42-46-47-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema no debe cargar la configuración de memoria en cache, las restantes configuraciones son cargadas correctamente.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 16: Ruta 4.

Caso de Prueba para el Camino: 4	
No. Ruta: 4	**Ruta:** 1-8-14-15-18-23-24-32-33-35-38-42-46-47-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema no debe cargar la configuración de objeto máximo ni la unidad, las restantes configuraciones son cargadas correctamente.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 17: Ruta 5.

Caso de Prueba para el Camino: 5	
No. Ruta: 5	**Ruta:** 1-8-14-15-18-23-24-27-32-33-42-46-47-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema no debe cargar la configuración de puerto de escucha ni la dirección ip, las restantes configuraciones son cargadas correctamente.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 18: Ruta 6.

Caso de Prueba para el Camino: 6	
No. Ruta: 6	**Ruta:** 1-8-14-15-18-23-24-27-32-33-35-38-42-46-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema no debe cargar la configuración de política de caché, las restantes configuraciones son cargadas correctamente.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 19: Ruta 7

Caso de Prueba para el camino: 7	
No. Ruta: 7	**Ruta:** 1-8-14-15-18-23-24-27-32-33-35-38-41-46-47-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema no debe cargar la configuración de política de caché, las restantes configuraciones son cargadas correctamente.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 20: Ruta 8.

Caso de Prueba para el camino: 8	
No. Ruta: 8	**Ruta:** 1-8-14-15-18-23-24-27-32-33-35-38-41-46-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	

Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.
Resultado esperado: El sistema no debe cargar la configuración de política de caché, además, carga el puerto por defecto no el del archivo de configuración, las restantes configuraciones son cargadas correctamente.
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.

Tabla 21: Ruta 9.

Caso de Prueba para el Camino: 9	
No. Ruta: 9	**Ruta:** 1-8-14-15-23-24-32-33-42-46-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema no debe cargar la configuración de memoria en caché, objeto máximo, puerto de escucha, dirección ip, política de cache, las restantes configuraciones son cargadas correctamente.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Tabla 22: Ruta 10.

Caso de Prueba para el Camino: 10	
No. Ruta: 10	**Ruta:** 1-8-14-15-18-23-24-32-33-35-38-41-46-50-52-53
Nombre de la persona que realiza la prueba: Alejandro Campos Palacios	
Descripción de la prueba: Leer las configuraciones básicas del servicio Squid.	
Entrada: Se envía como parámetro un logical server, al cual se le quiere cargar la configuración de Squid.	
Resultado esperado: El sistema no debe cargar la configuración de algoritmo de cache, las restantes configuraciones son cargadas correctamente.	
Evaluación de la Prueba: La prueba fue satisfactoria ya que se obtuvo el resultado esperado.	

Se realizaron tres iteraciones de la prueba unitaria. En la primera iteración se detectaron 3 no conformidades; en la segunda 2 no conformidades, las cuales fueron resueltas para la tercera iteración. Las no conformidades detectadas estaban asociadas a errores ortográficos, errores de validación y al tratamiento de las excepciones desde el código.

3.2.2 Prueba de integración

La prueba de integración es una técnica para construir la arquitectura del software. El objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño (22). Asumiendo la arquitectura por capas empleada y que el módulo debe incorporarse a una herramienta base, se emplea una estrategia de integración ascendente, donde los componentes se integran de abajo hacia arriba. En el contexto de la prueba de integración ascendente se realiza la prueba de regresión que permite ejecutar nuevamente el mismo subconjunto de pruebas que ya

se ha aplicado para asegurar que los cambios no han propagado efectos colaterales indeseables (22).

3.2.3 Prueba de aceptación

Con el objetivo de validar que el sistema cumple con el funcionamiento esperado y permitir que el usuario determine su aceptación desde el punto de vista de su funcionalidad y rendimiento, se realizan pruebas de aceptación (22). Para ello se tuvieron en cuenta las Historias de Usuario de mayor criticidad. A continuación, se presentan los casos de prueba correspondientes a las Historias de Usuario *Modificar configuración general, Adicionar lista de control de acceso* y *Adicionar regla de control de acceso*, a partir de los cuales el cliente ejecutó las pruebas.

Tabla 23: Caso de prueba de aceptación. Modificar configuración general.

Caso de prueba de aceptación	
Código caso de prueba: HMAST-Squid-1	**Nombre Historia de Usuario:** Modificar configuración general.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Editar los parámetros de la configuración general.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema HMAST. Debe adicionar un servidor lógico y activar el módulo Squid.	
Entrada/Pasos de ejecución: 1. Acceder al módulo Squid. 2. Opción “Configuración general” en el menú lateral izquierdo. 3. Seleccionar una opción. 4. Seleccionar el botón “Editar”. 5. Aceptar los cambios. 6. Aplicar los cambios en el servidor.	
Resultado esperado: Los parámetros de la opción seleccionada debe tomar los nuevos valores y muestra un mensaje notificándolo.	
Evaluación de la prueba: Satisfactoria.	

Tabla 24: Caso de prueba de aceptación. Adicionar ACL.

Caso de prueba de aceptación	
Código caso de prueba: HMAST-Squid-2	**Nombre Historia de Usuario:** Adicionar lista de control de acceso.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Adicionar una nueva lista de control de acceso al servidor.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema HMAST. Debe adicionar un servidor lógico y activar el módulo Squid.	
Entrada/Pasos de ejecución: 1. Acceder al módulo Squid. 2. Opción “Lista de control de acceso” en el menú lateral izquierdo. 3. Seleccionar el botón “Adicionar”. 4. LLenar los parámetros establecidos para una nueva ACL. 5. Aceptar los cambios. 6. Aplicar los cambios en el servidor.	
Resultado esperado: La nueva ACL es adicionada al servidor y muestra un mensaje notificándolo.	
Evaluación de la prueba: Satisfactoria.	

Tabla 25: Caso de prueba de aceptación. Adicionar regla de control de acceso.

Caso de prueba de aceptación	
Código caso de prueba: HMAST-Squid-2	**Nombre Historia de Usuario:** Adicionar regla de control de acceso.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Adicionar una nueva regla de control de acceso al servidor.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema HMAST. Debe adicionar un servidor lógico y activar el módulo Squid.	
Entrada/Pasos de ejecución: 1. Acceder al módulo Squid.	

2. Opción “Regla de control de acceso” en el menú lateral izquierdo. 3. Seleccionar el botón “Adicionar”. 4. LLenar los parámetros establecidos para una nueva regla. 5. Aceptar los cambios. 6. Aplicar los cambios en el servidor.
Resultado esperado: La nueva regla es adicionada al servidor y muestra un mensaje notificándolo.
Evaluación de la prueba: Satisfactoria.

Como parte de las pruebas de aceptación ejecutadas se realizaron 3 iteraciones, detectando nueve, tres y cero no conformidades en la primera, la segunda y la tercera iteración respectivamente.

Conclusiones parciales

El estudio del estándar de codificación definido en el capítulo permitió:

- Una codificación limpia, acorde a las pautas de la herramienta HMAST.
- Además, una vez desarrollada las funcionalidades del software, se procedió a la documentación y ejecución de pruebas. Las pruebas Unitarias realizadas a cada una de las funcionalidades del software, permitieron detectar errores en la implementación del código y validación existentes; por otra parte, con las pruebas de aceptación por parte del cliente se verificó la calidad del producto. De esta forma se detectaron no conformidades que fueron erradicadas en su totalidad; luego de solucionadas estas, se obtiene un sistema que cumple con las funcionalidades definidas al inicio de la investigación. Luego de concluir el proceso de implementación y pruebas realizadas al módulo, el mismo se encuentra integrado a HMAST y en condiciones de ser aplicado donde se necesite.

Conclusiones generales

Al finalizar la investigación y con los resultados obtenidos, relacionados con el desarrollo de funcionalidades que permitan desde HMAST la administración del servicio proxy, se concluye que:

- Entre las herramientas estudiadas Webmin, Zentyal y SmartKeeper; aportan para la solución del problema, pero no permite su integración a HMAST por la arquitectura de esta herramienta.
- La correcta utilización de las herramientas, lenguajes y tecnologías descritas, hicieron posible la obtención de un diseño e implementación acertados para el módulo desarrollado, proceso que estuvo guiado sobre la metodología AUP-UCI.
- La solución propuesta cumple con la totalidad de los requisitos funcionales definidos por el cliente, lo que se pudo constatar con el resultado satisfactorio de las pruebas de aceptación realizadas.

Recomendaciones

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

- Aumentar en el módulo la cantidad de elementos configurables que puedan ser manejados desde HMAST.
- Mejorar el manejo de elementos a bloquear y excluir en el control de acceso.

Referencias

1. Garcia Tomas, Jesus y Alcalde Lancharro, Eduardo. *Introducción a la Teleinformática.* Madrid : MCGRAW-HILL/INTERAMERICANA, 1993.

2. Barrios Dueñas, Joel. *Implementación De Servidores Con GNU/Linux .* México : s.n., 2015.

3. Pascual Viñé, Jose Carlos. *Introducción a la Telemática y a las Redes de Datos.* Madrid : Dirección de Servicios de Formación de Telefónica de España, 2000. RIS-220-01.

4. IBM. IBM Knowledge Center. [En línea] [Citado el: 24 de 05 de 2017.] https://www.ibm.com/support/knowledgecenter/es/SSKTXQ_8.5.0/com.ibm.help.sametime.v85.doc/config/st_adm_port_rvprxy_overview_c.html.

5. Taller Digital VW. *Servicio Web Hosting.* [En línea] [Citado el: 24 de 05 de 2017.] http://tallerdigitalvw.com/servidores/servidores-proxy-transparentes.

6. Universidad de Jaén. *Universidad de Jaén.* [En línea] 22 de 09 de 2005. [Citado el: 24 de 05 de 2017.] http://www.ujaen.es/sci/redes/proxy/.

7. Velázquez, Francisco, y otros. *The Varnish Book.* 2016.

8. GitHub Pages. *Tinyproxy.* [En línea] [Citado el: 24 de 05 de 2017.] https://tinyproxy.github.io/.

9. Debian.org. *Packages.* [En línea] [Citado el: 24 de 05 de 2017.] https://packages.debian.org/es/sid/privoxy.

10. Squid. *Squid-cache.org.* [En línea] 09 de 05 de 2013. [Citado el: 24 de 05 de 2017.] http://www.squid-cache.org/Support/.

11. Zentyal.org. Zentyal.org. [Online] Zentyal S.L., 2004. [Cited: 05 27, 2017.] http://www.zentyal.org/.

12. Webmin. Webmin. [Online] Webmin, 2006. [Cited: 05 27, 2017.] http://www.webmin.com/.

13. Universidad de las Ciencias Informáticas. *Manual de usuario Smart Keeper.* Habana : s.n., 2014.

14. Sánchez, Tamara Rodríguez. excriba.prod.uci.cu. [En línea] 2015. [Citado el: 27 de 05 de 2017.] http://excriba.uci.cu.

15. *Lenguajes de Programación.* Almagro, Carlos Ureña. Madrid : s.n., 2011.

16. López, Angel. *Java la programación del futuro.* Buenos Aires, Argentina : Rosgal S.A., 1997. 987-9131-38.

17. Van de Velde, Thomas, y otros. *Spring Framework.* Indianapolis, Indiana : Wiley Publishing, Inc, 2007. 978-0-471-10161-2.

18. Jet Brains. IntelliJ IDEA. [Online] JetBrains s.r.o, 2000. [Cited: 05 28, 2017.] https://www.jetbrains.com/idea/.

19. Visual Paradigm. Visual Paradigm. [Online] Visual Paradigm. [Cited: 05 28, 2017.] https://www.visual-paradigm.com/.

20. Booch, Gandy, Rumbaugh, Jim y Jacobobson, Ivar. UML. Lenguaje Unificado de modelado.

21. Straub, Ben y Chacon, Scoot. Pro Git. 2. 2014. ISBN-13: 978-1484200773.

22. Pressman, Roger S. Ingeniería del software. Un enfoque práctico. México : McGraw-Hill, 2010. 978-607-15-0314-5.

23. Zorrilla Castro, César. Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0. 2010. 978-84-936696-3-8.

24. GENBETA: dev. [En línea] 14 de julio de 2014. [Citado el: 22 de febrero de 2017.] https://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos.

25. Usaola, Macario Polo. Patrones GRASP.

26. Microsoft. msdn.microsoft.com. [En línea] [Citado el: 19 de marzo de 2017.] http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx..

27. CCM. [En línea] diciembre de 2016. [Citado el: 5 de diciembre de 2016.] http://es.ccm.net/contents/222-ingenieria-de-software-asistida-por-ordenador-case.

28. Software.com.ar. [En línea] [Citado el: 5 de diciembre de 2016.] http://www.software.com.ar/p/visual-paradigm-para-uml#product-description.

29. Aprenderaproramar.com. [En línea] 2016. [Citado el: 8 de diciembre de 2016.] http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163.

30. Diagramas UML. Catedra de Proyecto. [En línea] www.teatroabadia.com/es/uploads/documentos/iagramas_del_uml.pdf.

31. Gutiérrez, Javier. Introducción a el proceso de pruebas. España : s.n.

32. Tecnónología y Synergix. [En línea] 15 de marzo de 2008. [Citado el: 23 de marzo de 2017.] https://synergix.wordpress.com/2008/03/15/definimos-pruebas-de-unidad-como/.