**Universidad de las Ciencias Informáticas**

Facultad 1

# MÓDULO PARA LA GESTIÓN DE SERVIDORES PROXY SQUID DEL SISTEMA XILEMA SMART KEEPER

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


**Autor:**

Esteban Roberto de León Naranjo

**Tutores:**

MSc. Delly Lien González Hernández

MSc. Hubert Viltres Sala

Ing. Evelyn Labrada Oduardo

Ing. Ramón Morales Álvarez

**La Habana, 2018**

**“Año 60 de la Revolución”**

*"Internet facilita la información adecuada, en el momento adecuado, para el propósito adecuado."*

- Bill Gates

Bill Gates

# Declaración de autoría

Declaro por este medio que yo Esteban Roberto de León Naranjo, con carnet de identidad 93101217202, soy el autor principal del trabajo titulado **"Módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper"** 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.

Declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración de autoría en La Habana a los ___ días del mes de _____ del año 2018.

**Autor:**

_____

Esteban R. de León Naranjo

**Tutores:**

_____ _____

MSc. Delly Lien González Hernández MSc. Hubert Viltres Sala

_____ _____

Ing. Evelyn Labrada Oduardo Ing. Ramón Morales Álvarez

## Resumen

La presente investigación tiene como objetivo desarrollar un módulo que permita de forma remota y centralizada la gestión de servidores *proxy Squid* para el sistema Xilema Smart Keeper y contribuir al proceso de administración y configuración de los servicios del sistema para mejorar el monitoreo y control del acceso al contenido disponible en Internet desde las instituciones. El estudio y análisis del estado del arte permitieron identificar funcionalidades y tecnología para la comprensión y diseño de la solución propuesta. El desarrollo estuvo guiado por la metodología AUP-UCI y se seleccionó como principales tecnologías el marco de trabajo *Symfony*, *PHP* como lenguaje de programación y el entorno integrado de desarrollo *PHPStrom*, para las conexiones remotas se utilizó el protocolo SSH2, como componentes de filtrado los servidores *proxy Squid* y *Dansguardian*. La estrategia de prueba aplicada permitió verificar el cumplimiento de los objetivos trazados y evaluar la calidad del sistema. El método *Delphi* validó que la solución desarrollada tiene un alto valor para los administradores de red al facilitar las tareas de administración y configuración en el sistema Xilema Smart Keeper.

**Palabras claves:** centralizada, gestión*, proxy Squid*, remota, Xilema Smart Keeper.

# Tabla de Contenidos

# Índice de figuras

# Índice de tablas

# INTRODUCCIÓN

Internet crece a un ritmo exponencial, existen más de 3 billones de usuarios que acceden a la información publicada en más de 1 billón de páginas web (Internet Live Stats, 2018). Los usuarios al acceder a la web generan, consultan y comparten información que según Munitich (2013) se puede encontrar en diferentes formatos (videos, imágenes, audios y textos). Esta información aborda diversas temáticas relacionadas con información: científica, de ocio, instrucciones sobre preparación de bombas, producción de drogas ilegales, pornografía, actividades terroristas, instrucciones de hackeo de tarjetas de crédito, contenidos violentos que inciten al odio, a la discriminación y al fraude.

La información disponible en la web, ofrece a los usuarios una fuente de conocimiento que puede ser utilizada para realizar actos ilícitos o potencialmente nocivos a las personas, las instituciones o los gobiernos. Diariamente se realizan más de 62 mil ataques cibernéticos y se desarrollan más de 50 mil virus informáticos y programas maliciosos (Internet Live Stats, 2018) (KasperkyLab, 2017). La actividad delictiva de los usuarios en Internet afecta el orden público, político, comercial y jurídico e influye de forma negativa en la seguridad nacional, en la protección a los menores, la seguridad económica, la propiedad intelectual y la protección a la dignidad humana.

Para limitar que el acceso a una determinada información se convierta en un elemento que afecte los intereses de un individuo o institución, se utilizan soluciones informáticas que permiten controlar el acceso de los usuarios a la información disponible en la web. Según plantean Arab y otros (2015) y Benedico y otros (2017) utilizar servidores *proxy* y filtros de contenido mejora el control y monitoreo del acceso de los usuarios a la información disponible en la web.

En Cuba, la implementación de los Lineamientos del Partido Comunista (PCC) relacionados con el proceso de la informatización de la sociedad, potencia el acceso a Internet (Comité Central del PCC, 2016). Como parte de este proceso se han creado 459 espacios públicos de conexión inalámbrica y 259 salas de navegación que mejoran el acceso de los usuarios a la web (ETECSA, 2017). Con la utilización del Internet en las organizaciones, se ha incrementado la incorrecta gestión de la navegación por la red y se hace cada vez más difícil llevar el control de la misma. En Cuba se desarrollan soluciones para mejorar el control de Internet y el contenido que es accedido desde las instituciones.

La Universidad de las Ciencias Informáticas (UCI) en el cumplimiento de sus objetivos, desarrolla soluciones que contribuyan a un Internet más seguro. Entre estas soluciones se encuentran los filtros de contenido

web, utilizados para restringir los sitios web que puede visitar un usuario desde un dispositivo electrónico (KasperkyLab, 2017). Xilema Smart Keeper es un filtro de contenido, desarrollado por la UCI para controlar la política de uso aceptable en el acceso a Internet de las instituciones (Universidad de las Ciencias Informáticas, 2014). Entre los principales componentes se encuentra un sistema de filtrado y dos analizadores de registros para el servidor *proxy Squid*.

El *proxy Squid* es un tipo de servidor *proxy* para web con *cache*[1], aplica el filtrado de tráfico para Internet y proporciona un alto nivel de seguridad. Este es útil para mejorar el rendimiento de las conexiones a Internet, almacena en la *cache* peticiones recurrentes a servidores web y Servidores de Sistemas de Nombre de Dominio (DNS, por sus siglas en inglés) (Squid Cache, 2015).

La interfaz web de Xilema Smart Keeper permite que los administradores gestionen y monitoricen el acceso de los usuarios a Internet. La misma no dispone de una funcionalidad que permita, de forma remota y centralizada, gestionar los archivos de configuración y administración de los servicios del servidor *proxy Squid*. El administrador para realizar la gestión del control de acceso debe conectarse de manera independiente a cada servidor para realizar las tareas de administración y configuración. Este proceso es lento y engorroso, requiere que el administrador conozca numerosas estructuras de directorios, ficheros y comandos; además, dominar el trabajo en la terminal. Estas herramientas presentan problemas de seguridad al no utilizar una forma segura para la transferencia de información.

Sobre la base de los elementos expuestos y la necesidad de darle solución a esta problemática, se plantea como **problema científico:** ¿Cómo contribuir al proceso de gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper de forma remota y centralizada?

El **objeto de estudio** está definido por el proceso de gestión de los servidores *proxy Squid*, y el **campo de acción** está enmarcado en el proceso de gestión, remota y centralizada, de los servidores *proxy Squid*.

Como **objetivo general** se propone: Desarrollar un módulo que permita gestionar los servidores *proxy Squid* del sistema Xilema Smart Keeper de forma remota y centralizada.

Para alcanzar este objetivo general se han establecido los siguientes **objetivos específicos:**

[1] Memoria utilizada por el microprocesador para reducir el tiempo de acceso a datos ubicados en la memoria principal que se utilizan con más frecuencia.

- Construir los referentes teóricos fundamentales que sustentan la investigación relacionados con el desarrollo de un módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper.
- Diseñar las funcionalidades del módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper.
- Implementar las funcionalidades del módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper.
- Validar las funcionalidades del módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper.

Luego de haber realizado una revisión bibliográfica y desarrollado el marco teórico se plantea como **hipótesis científica:** El desarrollo de un módulo para la gestión los servidores *proxy Squid* del sistema Xilema Smart Keeper, facilitará al trabajo de los administradores del sistema.

Para dar cumplimiento a los objetivos trazados, se hace necesario realizar las siguientes **tareas de la investigación:**

- Estudio de los conceptos asociados al marco teórico de la investigación.
- Caracterización de aplicaciones que permitan administrar y configurar servidores *proxy Squid*.
- Selección de las herramientas, tecnologías y metodología de desarrollo a emplear para la concepción de la solución propuesta.
- Análisis y diseño de la solución propuesta en dependencia de la metodología que se determine utilizar para el desarrollo.
- Implementación de la solución en forma de módulo para ser integrada a la interfaz de administración del sistema Xilema Smart Keeper.
- Integración del módulo al sistema Xilema Smart Keeper.
- Validación de la solución teniendo en cuenta su funcionamiento, seguridad, rendimiento e integración.

Para darle cumplimiento a las tareas de la investigación se utilizaron **métodos** de trabajo científico con el propósito de establecer el camino a seguir por el autor para describir el conocimiento científico sobre la base de la teoría y la práctica para darle cumplimiento al objetivo de la investigación, a continuación, se describen los métodos utilizados:

**Métodos teóricos:**

**Histórico-lógico:** Se utiliza para estudiar y determinar la evolución, comportamiento y tendencias actuales de las tecnologías y herramientas existentes, tomándolas como punto de referencia y comparación de posibles resultados.

**Analítico-Sintético:** Es utilizado para hacer énfasis en el análisis de la documentación existente acerca del tema, con el objetivo de extraer los elementos más importantes para procesar la información y elaborar conclusiones para una mayor utilidad en el desarrollo del trabajo y en el momento de proponer una solución acertada.

**Modelación:** Se utiliza para realizar el modelado del sistema informático a través de los artefactos correspondientes a las fases: Modelo del Negocio, Requisitos, Análisis y Diseño e Implementación.

**Métodos empíricos:**

**Entrevista:** Se emplea en encuentros con el cliente para definir las funcionalidades de la aplicación web, identificando a la vez particularidades de cada usuario y las restricciones que se imponen.

**Observación:** Posibilita obtener conocimiento acerca del funcionamiento de los sistemas existentes en la actualidad para la administración y configuración del *proxy Squid*.

**Encuesta:** Se utilizó para la recopilación de información en la validación de la hipótesis científica, por el método Criterio de Expertos.

La siguiente investigación se desglosa en los siguientes capítulos:

**Capítulo 1 "Gestión de servidores *proxy Squid*":** En este capítulo se puntualizan los principales conceptos asociados al dominio de la investigación y se realiza un estudio del estado del arte de las soluciones existentes a nivel nacional e internacional. Se estudian, además, las herramientas, metodologías y técnicas utilizadas para dar solución al problema planteado.

**Capítulo 2 "Análisis y diseño del módulo de gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper":** En este capítulo se centra el desarrollo la investigación. Se realiza una descripción general de la solución propuesta y su funcionamiento, a partir de la cual se derivan las principales funcionalidades que esta debe contener. También se especifican los patrones del diseño aplicados, así como los artefactos derivados de la metodología de desarrollo de software seleccionada para esta etapa del proceso de

desarrollo del módulo propuesto. Se describe, además, la propuesta de solución, así como los requisitos funcionales y no funcionales que se desempeñan en la misma.

**Capítulo 3 "Implementación y validación del módulo de gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper:** En este capítulo se detalla la propuesta de solución al problema planteado. Se describe la organización del módulo en un diagrama de componentes y se especifican los estándares de codificación a utilizar. Se realizan las estrategias de pruebas definidas para el módulo y se muestran interfaces como parte del resultado final.

# CAPÍTULO 1: "GESTIÓN DE SERVIDORES *PROXY SQUID*"

Xilema Smart Keeper brinda una solución viable para mitigar el riesgo de acceso al contenido de Internet que se clasifique inapropiado. Como núcleo y principal servicio que compone el sistema está el servidor *proxy Squid*. Una correcta administración y configuración de este servicio a través de una interfaz gráfica, le propicia al administrador facilidades de gestión de este proceso. Con el objetivo de lograr una mayor comprensión del alcance de la investigación, en el presente capítulo se exponen los fundamentos teóricos asociados al dominio del problema planteado. Se realiza un estudio y análisis de los principales sistemas para la administración y configuración de los servidores *proxy Squid* identificados, se analizan sus características y procesos, ventajas y desventaja. Se analiza y se selecciona la metodología, tecnologías y herramientas que se utilizan para darle solución al problema.

## 1.1. Servidores *Proxy*

El término "*proxy"* proviene del inglés, traducido como delegado o apoderado (el que tiene poder sobre otro), aunque invariablemente se considera un sinónimo del concepto intermediario (Cambridge, 2017). Un servidor *proxy* se define como una computadora o dispositivo que ofrece un servicio de red que consiste en permitir a los clientes realizar conexiones indirectas hacia otros servicios de red (Barrios, 2014).

Durante el proceso de conexión, ver Figura 1, ocurre lo siguiente:

1. El cliente se conecta a un servidor *proxy*.
2. El cliente solicita una conexión, archivo u otros recursos disponibles en un servidor distinto.
3. El servidor *proxy* proporciona el recurso conectándose hacia el servidor especificado o desde una *cache*.
4. En algunos casos el servidor *proxy* puede alterar la solicitud del cliente o bien la respuesta del servidor para distintos propósitos.

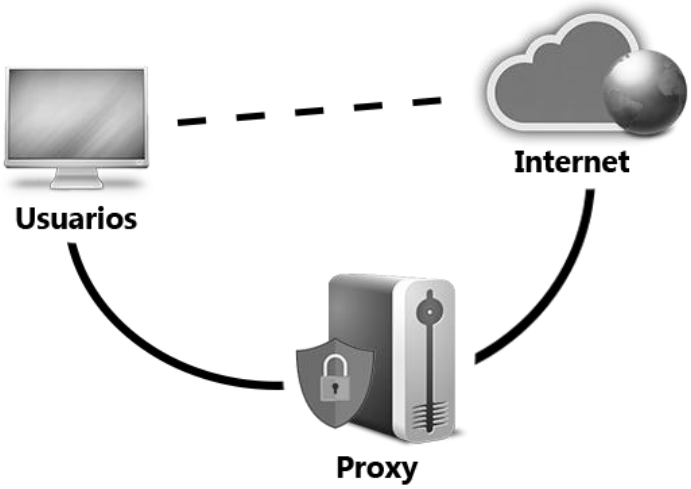



Figura 1. Funcionamiento del *servidor Proxy*

Los servidores *proxy* se pueden emplear como muros cortafuegos para operar en la Capa de Red del Modelo de Interconexión de Sistemas Abiertos (OSI, por sus siglas en inglés) y actuar como un filtro de paquetes o bien operar en la Capa de Aplicaciones para controlar los servicios de *TCP Wrapper*[2] (Beriseño, 2005).

Una aplicación común de los servidores *proxy* es funcionar como *cache* de contenido de red, esto permite a los clientes de la red local acceder de forma rápida y confiable al *proxy*. El contenido de la *cache* es eliminado mediante algoritmos de expiración que analiza antigüedad, tamaño e historial de respuesta de las solicitudes. Estos servidores *proxy* de contenido de red también pueden actuar como filtro de contenido, aplicando políticas de censura de acuerdo a criterios arbitrarios (Barrios, 2017).

## 1.2. *Proxy Squid*

*Squid* es un *proxy cache* para la web con soporte para los protocolos *HTTP*, *HTTPS*, *FTP*, *GOPHER* y *WAIS*. Reduce el consumo de ancho de banda y mejora los tiempos de respuesta en *cache* ya que reutiliza las páginas web solicitadas con frecuencia. *Squid* tiene amplios controles de acceso y funciona como servidor acelerador. Puede ser utilizado en los sistemas operativos más comunes incluyendo *Windows*, aunque su mayor uso es en los sistemas *GNU/Linux* y derivados de *Unix*. Es robusto, confiable y versátil,

[2] Software que actúa de intermediario entre las peticiones de usuarios de servicios y los demonios de los servidores que ofrecen el servicio.

se distribuye bajo los términos de la Licencia Pública General (GNU/GPL) y su código fuente es libre (Squid Cache,2015).

*Squid* consiste en un programa principal como servidor, un programa para la búsqueda en Servidores *DNS*, programas opcionales para reescribir solicitudes y realizar autenticación. Al iniciarse el *proxy Squid* origina un número configurable de procesos de búsqueda en Servidores *DNS*, cada uno realiza una búsqueda única, acelerando el tiempo de respuesta. Las principales funciones del *proxy Squid* son (Squid Cache,2015):

- Permite el acceso web a máquinas privadas (IP privada) que no están conectadas directamente a Internet.
- Controla el acceso web aplicando reglas de navegación.
- Registra el tráfico de Internet desde la red local al exterior.
- Controla el contenido visitado y descargado de Internet.
- Controla la seguridad de la red local ante posibles ataques e intrusiones en el sistema.

La implementación de un *proxy Squid* en una red proporciona grandes ventajas, como: Reducir el tiempo de respuesta, disminuye el tráfico de la red y el consumo del ancho de banda, aumenta la seguridad del usuario respecto a la información acceda. Para mejorar el rendimiento y efectividad del *proxy Squid* existen varios parámetros de configuración (Jara y otros, 2016).

## 1.3. Parámetros básicos configurables en el *proxy Squid*

La configuración del *proxy Squid* se realiza en un único archivo de texto plano generalmente ubicado en */etc/squid/squid.conf* de los sistemas operativos *GNU/Linux*. La estructura del archivo debe comenzar en la primera columna sin dejar espacios en blanco. Las investigaciones de Jara y otros (2016) demuestran que los parámetros básicos para un correcto funcionamiento del servidor *proxy Squid* son los siguientes:

**Parámetro *http_port***

Utilizado para indicar el puerto por el que atiende las peticiones el *proxy Squid*. En caso de no especificarse el *Squid* por defecto utiliza el puerto 3128.

```
http_port 3128
```

**Parámetro *cache_dir***

Utilizado para fijar el directorio y el espacio del disco duro que ocupan los datos para el almacenamiento *cache*. *Squid* usará por defecto 100 MB, dividido en jerarquías de 16 directorios subordinados, hasta 256 niveles cada uno, el tamaño de la *cache* se puede incrementar tanto como lo desee el administrador. Mientras más espacio tenga la *cache* más datos se almacenarán y se consumirá menos ancho de banda.

```
cache_dir ufs /var/cache/squid 100 16 256
```

**Control de acceso**

Para controlar el tráfico de los clientes hacia Internet es necesario establecer Listas de Control de Acceso (ACL, por sus siglas en inglés) que definan una red o varios anfitriones en particular. A cada lista se le asignará una Regla de Control de Acceso que permitirá o denegará el acceso a *Squid*.

**Listas de Control de Acceso:**

Regularmente una ACL se establece siguiendo la siguiente sintaxis:

*acl* [nombre de la lista] *src* [lo que compone la lista]

Por ejemplo, si se desea establecer una ACL que abarque toda la red local, se define la dirección *IP* correspondiente a la red y la máscara de la subred.

```
acl red_local src 10.0.0.0/8
```

También puede definirse una ACL en un archivo de texto plano localizado en cualquier parte del disco duro, este archivo contendrá una lista de direcciones.

```
acl maquinas_permitidas src /etc/squid/listas/permitidas
```

**Reglas de Control de Acceso:**

La sintaxis básica de una regla de control de acceso es:

*http_access* [*deny* | *allow*] [lista de control de acceso]

El siguiente ejemplo se considera una regla que establece acceso permitido a la ACL maquinas_permitidas.

```
http_access allow maquinas_permitidas
```

## 1.4. Principales herramientas de administración y configuración de servidores *proxy Squid*

Las herramientas para la configuración y administración de servidores permiten realizar cambios desde una interfaz de usuario sin necesidad de acceder directamente al fichero de configuración. En la actualidad se han desarrollado distintas herramientas, con el objetivo de identificar ventajas y desventajas de estas, se realiza el siguiente análisis enfatizado en las características y funcionalidades enfocadas en la administración y configuración del *proxy Squid*.

***Webmin***

Es una herramienta para la configuración y administración de servidores con distribuciones *GNU/Linux*, implementada en el lenguaje de programación *Perl*. Su estructura modular permite configurar varios servicios: *Apache*, *FTP*, *MySQL*, *Jabber*, *Samba* y *proxy*, para un total de 27 servicios completamente configurables desde una interfaz web que proporciona esta herramienta (Cameron, 2017). *Webmin* administra los servicios del *proxy* a través de *Squid*, permitiendo:

- Configuración del puerto que atiende las peticiones del *Squid*.
- Gestión de la c*ache*.
- Gestión de las ACL y Reglas de Control de Acceso.
- Sincronización del *Squid* con el cortafuego el sistema.
- Análisis de *logs*[3] que proporciona el *Squid*.
- Reconocimiento inicial y de los cambios que se efectúen el fichero de configuración de forma manual.

Para la configuración del servidor *proxy Squid, Webmin* implementa múltiples opciones, su interfaz web es poco intuitiva por los numerosos campos de configuración en la misma página. El sistema de ayuda que ofrece a los usuarios es escaso, en él solo se explican los términos poco utilizados, estos representan un bajo por ciento respecto al número de configuraciones que se muestran en la interfaz. *Webmin* tiene que ser instalado en cada servidor de la red, es complejo al tener uno o varios servidores dedicados[4], no permite

[3] Registro oficial de eventos de un sistema o servicio durante un rango de tiempo en particular.
[4] Ordenador que se utiliza para prestar un único servicio, generalmente relacionado con alojamiento web u otros servicios de red.

la configuración de las comunicaciones entre varios servidores *Squid*. Requiere que las conexiones se realicen mediante el usuario *root*[5] del sistema.

***Zentyal***

*Zentyal* es un servidor de red unificada de código abierto, desarrollado en 2004 por la empresa española *eBox Tecnologies S.L*. Esta herramienta integra 30 módulos para la administración y configuración de servicios basados en acceso a Internet, seguridad de la red, compartición de recursos, comunicaciones, unificadas en una sola tecnología (Zentyal Community, 2014). El módulo de *Zentyal* para el *proxy Squid* proporciona un conjunto de funcionalidades para facilitar la administración y configuración de este servicio, de ellas se destacan:

- Configuración de los parámetros básicos: puerto de petición, ACL, *cache*, validación de usuarios.
- Filtrado de páginas web.
- Limitar el uso de ancho de banda que consumen los usuarios.
- Configuración del *proxy Squid* como *proxy* transparente[6].

*Zentyal* enfocándose en la usabilidad, posee una interfaz web desarrollada en el lenguaje *Perl* completamente intuitiva que incluye únicamente aquellas funcionalidades de utilización frecuente (Samueza, 2015). Ofrece un sistema de ayuda sencillo que describe de manera muy general para que se utilizan los módulos, aunque dispone de un sitio oficial con una extensa documentación en español e inglés relacionada con las configuraciones básicas de los servicios de red, con ejemplos, imágenes y video tutoriales.

*Zentyal* administra y configura solo los servicios del servidor donde se instale, requiere de un Servidor web y otro de base de datos *Apache* y *PostgreSQL* respectivamente, esto se presenta como una desventaja cuando se desea tener un servidor dedicado a otro servicio. Este sistema no reconoce y sobrescribe las configuraciones establecidas en el fichero principal del *Squid*, no permite realizar configuraciones que no estén establecidas en el módulo como la comunicación entre *proxy*.

***YAST***

[5] Cuenta de usuario que tiene un control absoluto de todo lo que ocurre en un sistema.
[6] Es una característica que se le puede asociar a un servidor proxy para enrutar conexiones dentro del proxy sin configuración por parte del cliente, y habitualmente sin que el propio cliente conozca de su existencia.

*YAST* (Acrónimo del inglés, *Yet Another Setup Tools*) es una aplicación de escritorio bajo la licencia *GNU/GPL* para la distribución de Linux *OpenSUSE*, implementado en los lenguajes de programación C++ y *Ruby*. Esta herramienta basa su funcionamiento en la gestión de los servicios telemáticos que ofrece *OpenSUSE*, tiene una estructura modular para 16 servicios: correo electrónico, mensajería instantánea, transferencia de archivos, *DNS*, servidores web y *proxy* (SUSE, 2017). El módulo *YAST_SQUID* para la configuración del *proxy* le permite al administrador:

- Iniciar y detener el servicio.
- Configuración del puerto para atender peticiones.
- Gestión de la memoria *cache*.
- Gestión del Control de Acceso.
- Gestión y análisis de *logs* que proporciona el servidor *Squid*.

El sistema de ayuda que posee *YAST* es en forma de pequeños diálogos donde se describen las opciones de la ventana activa en el momento que se solicitó la opción de ayuda. *YAST* se caracteriza por su facilidad de uso y por su atractiva interfaz gráfica. Esta herramienta solo puede ser utilizada en la distribución *OpenSUSE* para la configuración local de los servicios.

**HMAST**

HMAST es un acrónimo de “Herramienta para la migración y administración de servicios telemáticos” desarrollada en el año 2012 por el Centro de Soluciones Libres (CESOL) de la UCI, es una aplicación de escritorio programada en el lenguaje *Python*. Esta herramienta permite migrar servidores de forma remota a plataformas libres, sus funcionalidades están definidas por: administración de usuarios, tareas programadas y servicios a través del protocolo *SSH* (Castillo y otros, 2012).

Su estructura en módulo permitió la incorporación de un módulo para administrar y configurar el *proxy Squid* el cual acogiéndose al plan establecido por el departamento de Servicios Integrales para la Migración, Asesoría y Soporte de CESOL, fue implementado con las siguientes funcionalidades:

- Configuraciones básicas como son: puerto por el que atiende las peticiones, memoria *cache*.
- Gestión de ACL.
- Gestión de políticas de acceso a las ACLs.
- Configuración de autenticación a través del *Squid*.
- Gestión de cuotas de usuario.

- Comunicación entre *proxy*.

### 1.4.1. Análisis del estudio a los sistemas homólogos

El estudio a las herramientas para administrar y configurar un *proxy Squid*, permitió realizar un resumen (Ver Tabla 1) donde se analizaron los siguientes parámetros:

**Lenguaje de Programación:** Se refiere al lenguaje de programación en el que fue desarrollado la herramienta analizada.

**Modo de Ejecución:** Se clasifica en dos categorías:

- Local: Para evaluar que la herramienta analizada solo administra los servicios del *proxy Squid* donde se encuentre instalada.
- Remota: Para evaluar que la herramienta analizada administra los servicios del *proxy Squid* de un servidor diferente.

**Tipo de Aplicación:** Para definir si la herramienta a evaluar es una aplicación web o una aplicación de escritorio (*Desktop*).

**Tipo de Licencia:** Se refiere al estado jurídico de la aplicación definido por su uso, modificación y distribución, esta puede ser pública, en aquellas que no necesitan un pago para ser utilizadas; o privativa, en aquellas que si lo requieren.

Tabla 1. Resumen del análisis de los sistemas homólogos

| Herramienta | Lenguaje de Programación | Modo de Ejecución | Tipo de Aplicación | Tipo de Licencia |
|---|---|---|---|---|
| ***Webmin*** | *Perl* | Local | Web | Pública |
| ***Zentyal*** | *Perl* | Local | Web | Pública |
| ***YAST*** | *C++, Ruby* | Local | *Desktop* | Pública |
| **HMAST** | *Python* | Remota | *Desktop* | Pública |

A partir del análisis realizado, y del presentado, el autor arriba a las siguientes conclusiones:

1. Las herramientas estudiadas basan su implementación en la administración de un único servidor *proxy*.
2. La herramienta HMAST permite administrar de forma remota los servicios del *proxy Squid.* Es una aplicación de escritorio orientada a la migración de servicios telemáticos a plataformas libres. No incluye las funcionalidades requeridas para la administración y configuración del *proxy Squid*.

Las herramientas analizadas no cumplen con el objetivo de la presente investigación, ni constituyen una solución aplicable al sistema Xilema Smart Keeper, toda la administración y configuración de este sistema se realiza mediante una aplicación web. Esto requiere que las funcionalidades que sean añadidas al sistema sean integrables con su interfaz de administración. La gestión de un único servidor *proxy* es una desventaja si la arquitectura de despliegue de Xilema Smart Keeper es distribuida y disponga de varios servidores dedicados al *proxy Squid*. Sin embargo, el análisis realizado posibilitó la identificación de funcionalidades y tecnologías que contribuyen al desarrollo de la propuesta de solución de la investigación.

## 1.5. Metodología de desarrollo de software

La metodología ágil *AUP*, es una versión simplificada del Proceso Unificado de *Rational* (*RUP*). Esta describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en *RUP*. El *AUP* aplica técnicas ágiles incluyendo (Sánchez, 2015):

- Desarrollo Dirigido por Pruebas (*test driven development* - TDD en inglés)
- Modelado ágil
- Gestión de cambios ágil
- Refactorización de Base de Datos para mejorar la productividad

Según Sánchez (2015), al no existir una metodología de software universal, toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.), exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología *AUP*, adaptandola al ciclo de vida definido para la actividad productiva de la UCI. Esta variación está unida al modelo CMMI-DEV v1.3[7], para garantizar las buenas prácticas en función de un software de calidad. El sistema Xilema Smart Keeper,

[7] Colección de buenas prácticas de desarrollo procedentes de la industria y del gobierno. Es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software (SEI, 2010).

al cual debe integrarse el módulo propuesto, fue desarrollada bajo esta metodología. Para el desarrollo de la propuesta de solución, se decide usar la metodología *AUP*, en su variación para la UCI, teniendo en cuenta todo lo antes expuesto.

## 1.6. Herramientas y tecnologías

La propuesta de solución es un módulo para el sistema Xilema Smart Keeper, al analizar la documentación existente relacionada con este, se identifican las bases tecnológicas y las herramientas que fueron empleadas para su desarrollo. Esto garantiza que exista compatibilidad del módulo con la interfaz de administración del sistema para una futura integración.

### 1.6.1. Lenguaje y Herramienta de Modelado

El lenguaje de modelado es un lenguaje informático gráfico o textual que sigue un conjunto semántico de reglas y marcos para el diseño y construcción de estructuras y modelos (Techopedia, 2017). Se utiliza el Lenguaje Unificado de Modelado (UML por sus siglas en inglés) en su versión 2.1, diseñado para visualizar, especificar, construir y documentar software orientados a objetos (Martínez, y otros, 2014).

Dentro de las principales herramientas para el modelado se encuentran herramientas CASE definidas por Araujo y otros (2013) como herramientas informáticas que asisten al diseñador en algunas de las actividades relacionadas con el desarrollo de un sistema (requerimientos, análisis, diseño, codificación y pruebas). *Visual Paradigm* para UML es una herramienta para el modelado de aplicaciones, ideal para ingenieros de software, analistas de sistemas y arquitectos de sistemas, que están interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Para la presente investigación, se utiliza esta herramienta en su versión 8.0.

### 1.6.2. Herramienta para el Control de Versiones

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante (Saldaña, 2017). Para garantizar el control de versiones en el sistema y a su vez, la continua integración del módulo a el sistema Xilema Smart Keeper, se emplea el repositorio *GitLab*, se propone la utilización del cliente *Git* en su versión 2.8, el cual, según Saldaña (2017):

- Es un sistema de control de versiones distribuido
- No depende de acceso a la red o un repositorio central

- Está enfocado a la velocidad, uso práctico y manejo de proyectos grandes.

### 1.6.3. Marco de trabajo

Los marcos de trabajos (*Frameworks)* se pueden definir según (Gómez y otros, 2016) como una plataforma que dependiendo del lenguaje de programación utilizan componentes que facilitan el desarrollo e implementación de aplicaciones. Los marcos de trabajo se pueden considerar una aplicación genérica incompleta y configurable a la que se le añaden las últimas piezas para construir una aplicación completa.

Para el desarrollo del módulo se utiliza el *framework Symfony* en su versión 2.8.2 el cual es un marco de trabajo basado en el patrón Modelo Vista Controlador (MVC) para el desarrollo de aplicaciones web. Implementado en *PHP* (del inglés *Hipertext Pre-procesor*), con él se pueden crear aplicaciones y sitios web rápidos y seguros de una forma profesional (Symfony, 2017).

Para el diseño de la interfaz web de la propuesta de solución se utiliza el *framework CSS* (del inglés *Cascading Style Sheets*) *Boostrap* en su versión 3.3.5, este permite de forma rápida la construcción de interfaces web. Su principal característica radica en la adaptabilidad del diseño web al tamaño de la pantalla del ordenador, *smartphone* o *tablet* (Hidalgo, 2015).

### 1.6.4. Lenguaje de programación del lado servidor

Un lenguaje de programación es un conjunto de reglas semánticas y sintácticas que se utilizan para la codificación de instrucciones de un programa o algoritmo de programación (Álvarez, 2016). Para el desarrollo del módulo se emplea el lenguaje *PHP* en su versión 7.0, es un lenguaje de programación gratuito e independiente de plataforma, rápido, con una gran librería de funciones y un extenso volumen de documentación (PHP, 2017). Según las investigaciones de (Martínez, 2014) se puede afirmar que las principales ventajas de *PHP* son:

- Aplicación de técnicas de la programación orientado a objeto para el desarrollo de aplicaciones web dinámicas.
- Capacidad de expandir su potencial utilizando módulos y extensiones.
- Simplificación de distintas especificaciones, como es el caso de la definición de las variables primitivas.

- Posee una amplia documentación en varios idiomas en su sitio web oficial[8], entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Tiene manejo de excepciones.
- Capacidad de conexión con la mayoría de los gestores de base de datos que se utilizan en la actualidad, destaca su conectividad con *MySQL* y *PostgreSQL*.

### 1.6.5. Lenguaje de desarrollo del lado cliente

*JavaScript* se define como un lenguaje de programación interpretado, no requiere la compilación del código de fuente para ejecutarlo. Este trabaja sobre un enfoque útil y práctico hace que el proceso de desarrollo de aplicaciones web dinámicas sea eficiente, está disponible en cualquier navegador web (Grados, 2016).

Se utiliza el Lenguaje de Marcado de Hipertexto (*HTML* por sus siglas en inglés) en su versión 5.0. *HTML* se utiliza en la traducción y descripción de la estructura y la información en forma de texto, así como para complementar el texto con objetos e imágenes (Abury, 2014). Es un lenguaje básico predefinido en el marco de trabajo seleccionado.

Para definir la apariencia o estilo de las paginas páginas web escritas en *HTML* del módulo propuesto se utiliza *CSS* versión 3.0. Este permite separar el contenido de la forma y proporciona a los diseñadores el mantenimiento de un control más preciso sobre la apariencia de las páginas web. Su novedad más importante consiste en la incorporación de mecanismos para mantener un mayor control sobre el estilo de los elementos que se muestran en la página (Beati, 2016).

### 1.6.6. Librerías

Una librería es un conjunto de recursos (algoritmos) prefabricados, que pueden ser utilizados por el programador para realizar determinadas operaciones. Las librerías utilizadas para el desarrollo de la propuesta de solución se relacionan a continuación:

***jQuery***

*JQuery* es una biblioteca multiplataforma de *JavaScript*, que permite simplificar la manera de interactuar con los documentos *HTML*, manipular el árbol *DOM*[9], manejar eventos, desarrollar animaciones y agregar

[8] Sitio Oficial de PHP: http://www.php.net
[9] Del inglés, *Document Object Model* (Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos).

interacción con la técnica *AJAX*[10] a páginas web. Es además un software libre y de código abierto, permitiendo su uso en proyectos tanto libres como privados. *JQuery*, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio (Beati, 2016). Para la implementación del módulo se emplea la librería *JQuery* en su versión 1.11.2.

**Librerías para la conexión *SSH* a través de *PHP***

- ***libssh2-php*:** Es una librería implementada en el lenguaje de programación *C* para realizar conexiones remotas con *PHP* a través del protocolo *SSH*. Está disponible en todos los repositorios de software de *GNU/Linux*, su documentación está incluida en los manuales oficiales de *PHP* (Troya, 2015). La misma permite:
  - Autenticación utilizando una llave del *host* público, una clave pública o con una contraseña simple.
  - Ejecutar un comando en un servidor remoto.
  - Recuperar una secuencia de datos ampliada y las huellas dactilares del servidor remoto.
  - Añadir una clave pública autorizada.
  - Enviar y obtener fichero mediante *SCP*.
  - Crear enlaces simbólicos y directorios.
  - Renombrar y eliminar ficheros remotos.
  - Abrir túneles a través de un servidor remoto y solicitar un *shell*[11] interactivo.
- ***phpseclib***: Es una librería completamente escrita en *PHP*. Posee implementación para algoritmos criptográficos y contiene funciones para realizar conexiones remotas a través de *SSH*, permite la autenticación mediante claves públicas, ejecutar comandos en el servidor remoto, obtener y enviar ficheros. A esta librería solo se le dio soporte hasta la versión 5 de *PHP* (Wigginton, 2017).

Por el análisis realizado a librerías *SHH* para realizar conexiones remotas a través del lenguaje de programación *PHP* se selecciona ***libssh2-php*** por sus funcionalidades, documentación y disponibilidad.

[10] Del inglés, *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML).
[11] Interprete de comandos que tiene una interfaz de usuario para acceder a los servicios de un sistema operativo (Hanifi, 2016)

### 1.6.7. Entorno Integrado de Desarrollo

*JetBrains PHPStrom* es un Entorno Integrado de Desarrollo (*IDE*, por sus siglas en inglés) de código abierto y multiplataforma creado por la compañía *JetBrains*, es un poderoso y completo editor del lenguaje de programación *PHP* centrado en la productividad del desarrollador. Contiene un editor de código inteligente que analiza sintaxis, configuración de formato del código extendido, comprobación de errores en tiempo real sobre el proceso de implementación, plegado del código en bloques y autocompletado. Brinda soporte a los lenguajes *CSS*, *JavaScript* y *HTML*. Dispone de los sistemas de control de versiones más populares como *Git, Subversion, Mercurial y Perforce*. Ofrece un formato de código acorde a los estándares de codificación de *Symfony* con la integración de un módulo para este *framework*. Para el desarrollo de la propuesta de solución se utiliza *JetBrains PHPStrom* en su versión 2017.1.4 (JetBrains, 2017).

### 1.6.8. Servidor de Base de Datos

Una de las tareas más comunes de las aplicaciones web dinámicas es la persistencia y lectura de la información en las bases de datos. *Symfony* se integra a *Doctrine*, biblioteca que se utiliza para simplificar las operaciones que se realizan en las bases de datos, posee controladores que permiten la compatibilidad con servidores de código abierto como es el caso de *MySQL* (Pacheco, 2013).

***MySQL***

Es un SGBD relacional, su diseño multihilo le permite soportar una gran carga de datos de forma eficiente. *MySQL* es el gestor de base de datos de código abierto más utilizado a nivel mundial por su rapidez y facilidad de uso datos ofrecidos por Oracle (2017). Su utilización está enfocada en aplicaciones web dinámicas escritas en *PHP* por la optimización de consultas sencillas y su compatibilidad con el servidor web *Apache*. Este se caracteriza por su nivel de seguridad, es multiplataforma, admite hasta 32 índices por tablas, la variedad de tipos de datos para las columnas y la utilización de *triggers* (Oracle, 2017).

Se selecciona *MySQL* versión 5.7 como SGBD por su velocidad de procesamiento en operaciones y garantizar un mayor rendimiento cuando se consultan las bases de datos, su bajo consumo ante servidores con bajas prestaciones y su facilidad de uso en la administración.

### *1.6.9.* Servidor Web

Para el hospedaje del sistema se emplea el servidor web *Apache* 2.2, es un software de código abierto desarrollado por la empresa *Apache Software Fundation* en 1996. Es un servidor web flexible, rápido y

eficiente, altamente configurable por su diseño modular, esta característica permite ampliar considerablemente sus capacidades al existir un repositorio extenso completamente gratuito de extensiones y módulos. Es compatible con el lenguaje de programación *PHP,* comparten muchas de sus características (Apache, 2017).

### 1.6.10. Componentes de filtrado del sistema Xilema Smart Keeper

Los componentes de filtrado son elementos de software que actúan en conjunto para evitar que los usuarios accedan a material que se clasifique inadecuado en Internet. Para el desarrollo de la propuesta de solución se utilizan los servidores *proxy Squid* versión 3.0 y *Dansguardian* en su versión 2.10 como componentes de filtrado web para el sistema Xilema Smart Keeper.

**Dansguardian**

Es un filtro de contenido web de código abierto desarrollado en el lenguaje de programación *C++.* Permite el filtrado de Internet en tiempo real en función de métodos como la coincidencia de frases, *URL*, patrones de expresiones regulares y extensiones de archivos. Es flexible y adaptable a las necesidades de filtrado de las instituciones principalmente escuelas, empresas y redes gubernamentales.

# CAPÍTULO 2: "ANÁLISIS Y DISEÑO DEL MÓDULO PARA LA GESTIÓN DE SERVIDORES *PROXY SQUID* DEL SISTEMA XILEMA SMART KEEPER"

En el presente capítulo se describen las principales características de la propuesta de solución y se identifican los requisitos funcionales y no funcionales. Se describe la arquitectura de software, los patrones de diseños y los artefactos generados para darle solución al problema planteado en la investigación.

## 2.1. Descripción de la propuesta de solución

El módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper tiene como principal característica, administrar y configurar los servicios de forma remota y centralizada del *proxy Squid*. El módulo propuesto (ver Figura 2), se integra a la interfaz de administración del sistema Xilema Smart Keeper y permite a los administradores consultar el estado actual del *proxy Squid* (Iniciado, Detenido, En espera y Sin conexión), cambiar su estado (Iniciar, Detener y Reiniciar) y administrar el fichero de configuración a través de formularios. El módulo permite configurar los componentes de filtrado (*Dansguardian*) que serán asignados a cada *Squid* y gestiona las reglas para el filtrado que establezca la institución.

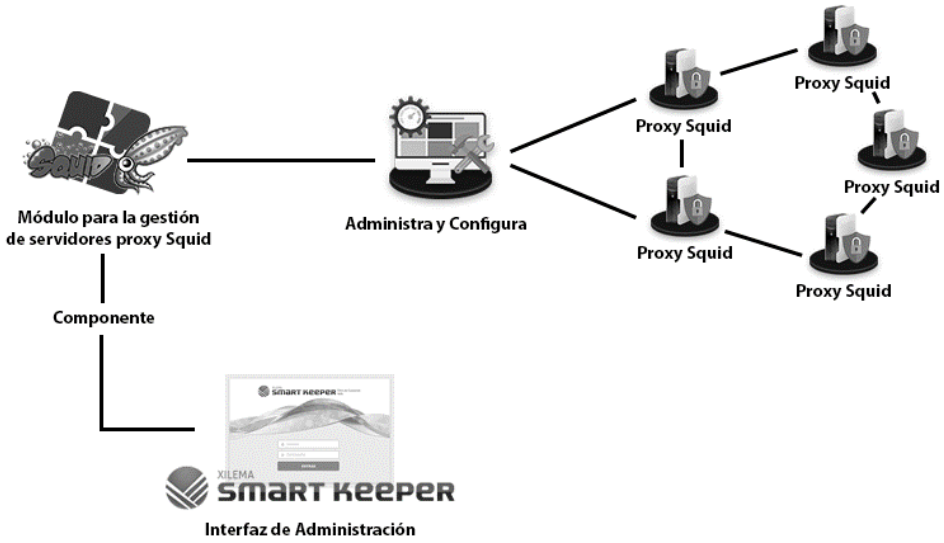



Figura 2. Propuesta de solución

## 2.2. Modelo conceptual

A partir de un análisis a los procesos que requieren ser informatizados, se realizó un modelo conceptual (Figura 3), este describe los procesos importantes del contexto del módulo y las relaciones entre ellos. En el modelo se describe al Administrador como principal actor del proceso. Este mediante la interfaz de administración gestiona los servicios y sus ficheros de configuración. Los servicios son definidos como los componentes de filtrado *Dansguardian* y los servidores *proxy Squid*. A continuación, se describen las clases que componen el modelo conceptual:

- **Administrador:** Usuario con máximo privilegios encargado de realizar todas las acciones en el módulo.
- **Interfaz de Administración:** Interfaz gráfica del sistema Xilema Smart Keeper, donde interactúa el Administrador y realiza todas las acciones.
- **Servicios:** Representa los servicios que gestiona el Administrador en el módulo.
- ***Proxy Squid*:** Tipo de servicio que representa a los servidores *proxy Squid* instalados en el sistema.
- ***Dansguardian*:** Tipo de servicio que representa a los componentes de filtrado instalados en el sistema.
- **Ficheros de Configuración:** Archivos que representan las configuraciones y los comportamientos de los servicios.

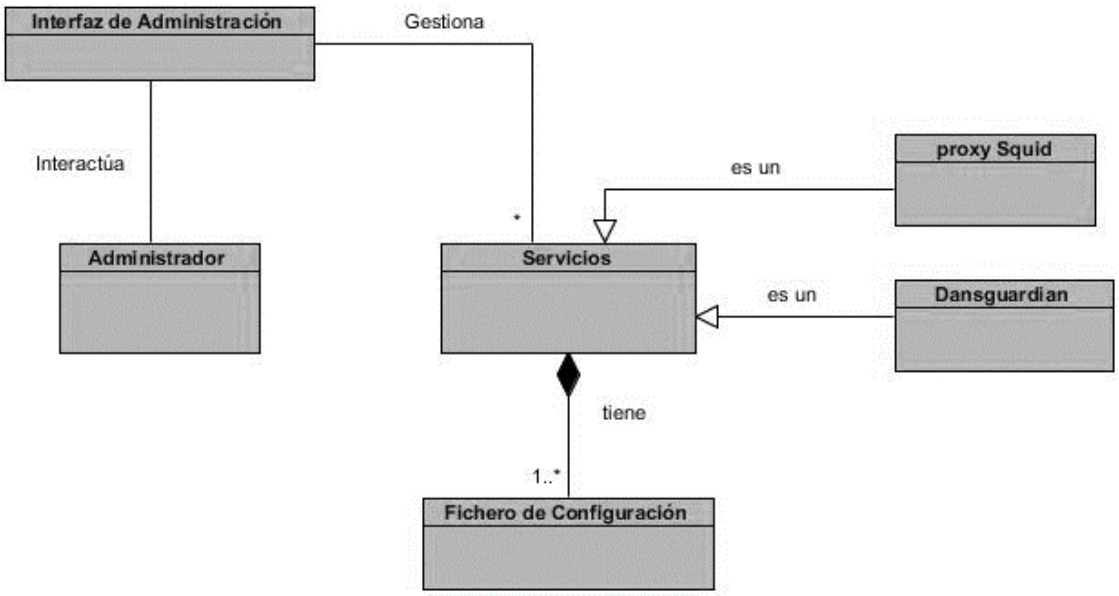



Figura 3. Modelo Conceptual

## 2.3. Requisitos de la propuesta de solución

La especificación de los requisitos de software establece una base sólida para su diseño e implementación. Estos definen las necesidades, describen los escenarios de uso, delinean las funciones y características e identifican las restricciones del proyecto (Pressman, 2010). Los requisitos se pueden clasificar en funcionales y no funcionales.

### 2.3.1. Requisitos funcionales

Según Ramos y otros (2015), los requisitos funcionales (RF) son aquellos directamente relacionados con las funciones y las reacciones que el sistema debe proporcionar. Son asignados directamente a elementos o características del sistema de software. Los mismos se especifican en la Tabla 2.

Tabla 2. Requisitos funcionales del módulo

| Requisitos | Descripción | Prioridad |
|---|---|---|
| **RF1:** Insertar *proxy Squid* | El sistema debe permitir insertar un *proxy Squid* | Alta |
| **RF2:** Mostrar *proxy Squid* | El sistema debe permitir mostrar detalles de los *proxy Squid* | Media |
| **RF3:** Modificar *proxy Squid* | El sistema debe permitir modificar los *proxy Squid* | Media |
| **RF4:** Configurar servicios del *proxy Squid* | El sistema debe permitir configurar los servicios del *proxy Squid* | Alta |
| **RF5:** Eliminar *proxy Squid* | El sistema debe permitir eliminar los *proxy Squid* | Media |
| **RF6:** Comprobar conexión del servidor *proxy Squid* | El sistema debe permitir comprobar la conexión al servidor *proxy Squid* que se está insertando | Baja |
| **RF7:** Consultar estado del servicio del *proxy Squid* | El sistema debe permitir mostrar la información del estado de los *proxy Squid* (Sin Conexión, Detenido, En Espera, Iniciado) | Baja |
| **RF 8:** Buscar *proxy Squid* | El sistema debe permitir realizar una búsqueda simple de los servidores *proxy Squid* | Baja |
| **RF 9:** Configurar servicio del *Dansguardian* | El sistema debe permitir configurar los servicios del *Dansguardian* relacionados con el *proxy Squid* asociado | Alta |
| **RF 10:** Iniciar servicio del *proxy Squid* | El sistema debe permitir iniciar los servicios del *proxy Squid* | Alta |
| **RF 11:** Detener servicio del *proxy Squid* | El sistema debe permitir detener los servicios del *proxy Squid* | Alta |

| | | |
|---|---|---|
| **RF 12:** Reiniciar servicio del *proxy Squid* | El sistema debe permitir reiniciar los servicios del *proxy Squid* | Alta |
| **RF 13:** Configurar reglas de filtrado | El sistema debe permitir configurar reglas de filtrado del *Dansguardian* asociado a un *proxy Squid* | Alta |

### 2.3.2. Requisitos no funcionales

Los requisitos no funcionales (RnF) son restricciones impuestas al funcionamiento del sistema, relacionadas con el tiempo, presupuesto, proceso de desarrollo, políticas de la organización y normas que deben adoptarse (Ramos y otros, 2015). A partir de un análisis se identificaron nueve (9) RnF, distribuidos en especificaciones de usabilidad, hardware, software, seguridad y apariencia, los cuales se relacionan a continuación:

- **Usabilidad:**

  **RnF 1:** Debe ser intuitivo, los requisitos funcionales son agrupados por responsabilidad, para permitir a los administradores de red de poca experiencia, hacer uso del mismo.

  **RnF 2:** Debe disponer de un sistema de ayuda, el cual explicará con detalles como utilizar el módulo y los principales conceptos asociados a las configuraciones del *proxy Squid*.

- **Hardware:**

  **RnF 3:** El servidor donde se instale el módulo debe tener una memoria *RAM* de 2GB o superior y un procesador a una velocidad 2.10 GHz o superior.

  **RnF 4:** El servidor de base de datos *MySQL* debe cumplir con los requerimientos mínimos (80GB de disco duro) que define el sistema Xilema Smart Keeper.

- **Software:**

  **RnF 5:** El ordenador donde se ejecute el módulo debe tener instalada la distribución de *GNU/Linux Ubuntu* 14.04 o superior.

  **RnF 6:** El ordenador donde se ejecute el módulo requiere de las librerías *ssh*, *openssl* y un servidor de base de datos *MySQL.*

- **Seguridad:**

**RnF 7:** La información en el módulo debe estar protegida de acceso no autorizados, se utilizan los mecanismos de autenticación del sistema Xilema Smart Keeper para el control de acceso.

**RnF 8:** El sistema utilizará mecanismos de encriptación para las contraseñas almacenadas en la base de datos.

- **Apariencia o interfaz externa:**

**RnF 9:** El módulo debe mantener las pautas de diseño que establece la estrategia marcaria de la UCI para productos comerciales.

## 2.4. Historias de usuario

La metodología *AUP*-UCI, en su escenario 4 para la disciplina Requisitos, genera como artefacto a las Historias de Usuario (HU) (Sánchez, 2015), estas se utilizan para especificar los requisitos de software en las aplicaciones. Las HU se descomponen en tareas de programación y describen las características que el sistema debe cumplir, están escritas en un formato legible por el cliente, sin necesidad de sintaxis técnicas (Ordoñez y otros, 2015).

A continuación, se muestran dos (2) HU de las trece (13) que fueron generadas, pertenecientes a los RF 4 y RF 13 respectivamente, el resto puede ser consultada en el Anexo 1 de la presente investigación.

Tabla 3. HU_4 Configurar servicios del *proxy Squid*

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_4 | **Nombre:** Configurar servicios del *proxy Squid* |
| **Programador responsable:** Esteban R. de León Naranjo | **Iteración asignada:** 1 |
| **Prioridad:** Alta | |
| **Tiempo estimado:** 12 horas | **Tiempo real:** 12 horas |
| **Descripción:** Los administradores pueden configurar a través de un formulario los servicios de *proxy Squid*. | |
| **Observaciones:** Un *proxy Squid* puede ser configurado si el estado es distinto a Sin Conexión. Si los datos son introducidos correctamente el sistema mostrará un mensaje de notificación “El *proxy Squid* <nombre del *proxy*> fue configurado correctamente” y automáticamente será reiniciado para aplicar los cambios. Si el usuario introduce los datos incorrectamente el sistema mostrará un mensaje de error y señalará en color rojo aquellos campos que su validación fue incorrecta. | |

**Prototipo de Interfaz:**


Tabla 4. HU_13 Configurar reglas de filtrado

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_13 | **Nombre:** Configurar reglas de filtrado |
| **Programador responsable:** Esteban R. de León Naranjo | **Iteración asignada:** 1 |
| **Prioridad:** Alta | |
| **Tiempo estimado:** 8 horas | **Tiempo real:** 8 horas |
| **Descripción:** Permite configurar las reglas de filtrado de un *Dansguardian* asociado a un *proxy Squid*. Se elige el tipo de regla a configurar y los parámetros de filtrado para la regla seleccionada. | |
| **Observaciones:** Una regla de filtrado puede ser configurada si el estado de los servicios asociadas a esta es distinto a Sin Conexión. Si los datos son introducidos correctamente el sistema mostrará un mensaje de notificación “La regla fue aplicada con éxito” y automáticamente será reiniciado el servicio *Dansguardian* para aplicar los cambios. Si el usuario introduce los datos incorrectamente el sistema mostrará un mensaje de error y se señalará en color rojo aquellos campos que su validación fue incorrecta. | |

**Prototipo de Interfaz:**


## 2.5. Arquitectura de Software

La arquitectura de software es un conjunto de patrones que definen la organización de un sistema, sus componentes, el ambiente, y los principios que orientan su diseño y evolución. Los patrones arquitectónicos es la estructura del sistema que comprende a los elementos de software, las propiedades visibles externamente de dichos elementos y las relaciones entre ellos (Rodríguez y otros, 2016).

### Modelo – Vista – Controlador

Symfony basa su flujo de trabajo en el patrón arquitectónico Modelo – Vista – Controlador, implementada de forma que el desarrollo de aplicaciones sea rápido y sencillo. Este patrón está formado por tres niveles:

- **Modelo:** Representa la información con la que trabaja el sistema, su lógica de negocio. El modelo está representado en los archivos del directorio: *Entity.*
- **Vista:** Transforma el modelo en una página web y permite al usuario interactuar con este. Las vistas del módulo están agrupadas en el directorio: *app/Resources/view.*
- **Controlador:** Se encarga de procesar las interacciones del usuario, y realiza los cambios apropiados en el modelo o la vista. Las clases controladoras de la propuesta de solución están situadas en el directorio: *Controller.*

En el controlador se encuentran las acciones que permiten configurar el *proxy Squid*, gestionar el estado de los servicios y las reglas de filtrado, son el núcleo del módulo y contienen la lógica de la aplicación. Estas acciones utilizan el modelo y envían las variables a la vista, que es la encargada de originar las páginas que son mostradas como resultado de las acciones. En el modelo se encuentra la clase relacionada con el *proxy Squid* para el acceso y la manipulación de los datos, realizado mediante objetos. El controlador es la capa intermediaria entre la Vista y el Modelo.

## 2.6. Patrones de diseño

Según Pressman (2010), un patrón de diseño se caracteriza como una regla de tres (3) partes que expresa una relación entre cierto contexto, un problema y una solución. Para el diseño de software, el contexto permite entender el ambiente del problema y la solución apropiada. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas que influyen en la interpretación del contexto del problema y cómo podría aplicarse con eficacia la solución. Durante la implementación de la propuesta de solución se utilizaron los siguientes patrones de diseño:

### 2.6.1. Patrones Generales de Software para la Asignación de Responsabilidades (GRASP)

Los patrones GRASP, fomentan una serie de buenas prácticas para el diseño de software (Cárdenas, 2016). De esta familia, para el desarrollo del módulo se utilizaron los siguientes patrones (Hamon, 2014):

**Experto:** Asigna las responsabilidades a aquellos objetos que disponen de la información para hacerlo. En el módulo, se evidencia el uso de este patrón, al realizar cambios de estado en un determinado *proxy Squid.* La clase entidad ***Squid*** es la que contiene los atributos necesarios para realizar la conexión remota al servidor y cambiar su estado, esta clase es la experta en información.

**Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos para lograr menor dependencia y mayor oportunidad en la reutilización del código. En el módulo este patrón se utiliza, por ejemplo, en la función ***EncryptAction*** encargada de aplicar algoritmos de encriptación a la contraseña obtenida por parámetro en texto plano. Este retorna a la función que realizó la petición, una instancia del objeto contraseña encriptada.

**Alta Cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Las mismas poseen un número relativamente pequeño de responsabilidades, las clases realizan solo las funciones para las cuales fueron creadas, esto conlleva a un bajo acoplamiento y fomenta la reutilización. En el módulo, este patrón se manifiesta en la clase controladora ***SquidController***, colabora

y delega responsabilidades a la clase ***SquidType*** para la creación de formularios relacionados con la entidad ***Squid***.

**Bajo Acoplamiento:** Es una medida de la fuerza en las conexiones de una clase con otras, la recurrencia y las conexiones a ella. Una clase con bajo acoplamiento no depende de muchas clases. Su utilización se evidencia en que las clases controladoras del sistema no se relacionan entre sí, lo que disminuye las dependencias entre las mismas.

**Controlador:** Un Controlador es un objeto de interfaz no destinado al usuario, se encarga de manejar un evento del sistema a una clase que represente el sistema global. En el módulo, los eventos generados por el usuario son redirigidos a una clase controladora que realiza las operaciones solicitadas.

### 2.6.2. Patrones Gang of Four (GOF)

Describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Gestionan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas (Craig, 2003). Los patrones GOF utilizados en el módulo se relacionan a continuación:

**Estructurales:**

***Decorator*** **(Decorador):** Permite añadir responsabilidades adicionales a un objeto dinámicamente para proporcionar una alternativa flexible a la especialización mediante herencia si se añaden nuevas funcionalidades. En el módulo se observa este patrón en la vista ***base.html.twig*** de ella heredan el resto de las páginas de la aplicación. Esta vista contiene los elementos que son comunes para el diseño y estructura del módulo.

**Otros:**

***Registry:*** Este patrón es útil para los desarrolladores en la Programación Orientada a Objetos. Es un medio sencillo y eficiente de compartir datos y objetos en la aplicación, sin la necesidad de conservar numerosos parámetros o utilizar variables globales. Se aplica en la clase de configuración (*config.yml*) que es la encargada de guardar las variables globales del módulo, como *drivers* de conexión a la base de datos y rutas bases del proyecto.

## 2.7. Diagramas de clases del diseño

Un Diagrama de Clases del Diseño (DCD) muestra la especificación de las clases de una aplicación, sus asociaciones, atributos y métodos, interfaces, navegabilidad y dependencias. Las clases de diseño de los DCD definen entidades y no conceptos del mundo real (UNAD, 2016). Para la presente investigación se generaron un total de siete (7) DCD relacionados con los requisitos de prioridad alta, de ellos se exponen dos (2) correspondientes a los HU_4 y HU_13 respectivamente, el resto de los DCD puede ser consultado en el Anexo 2.

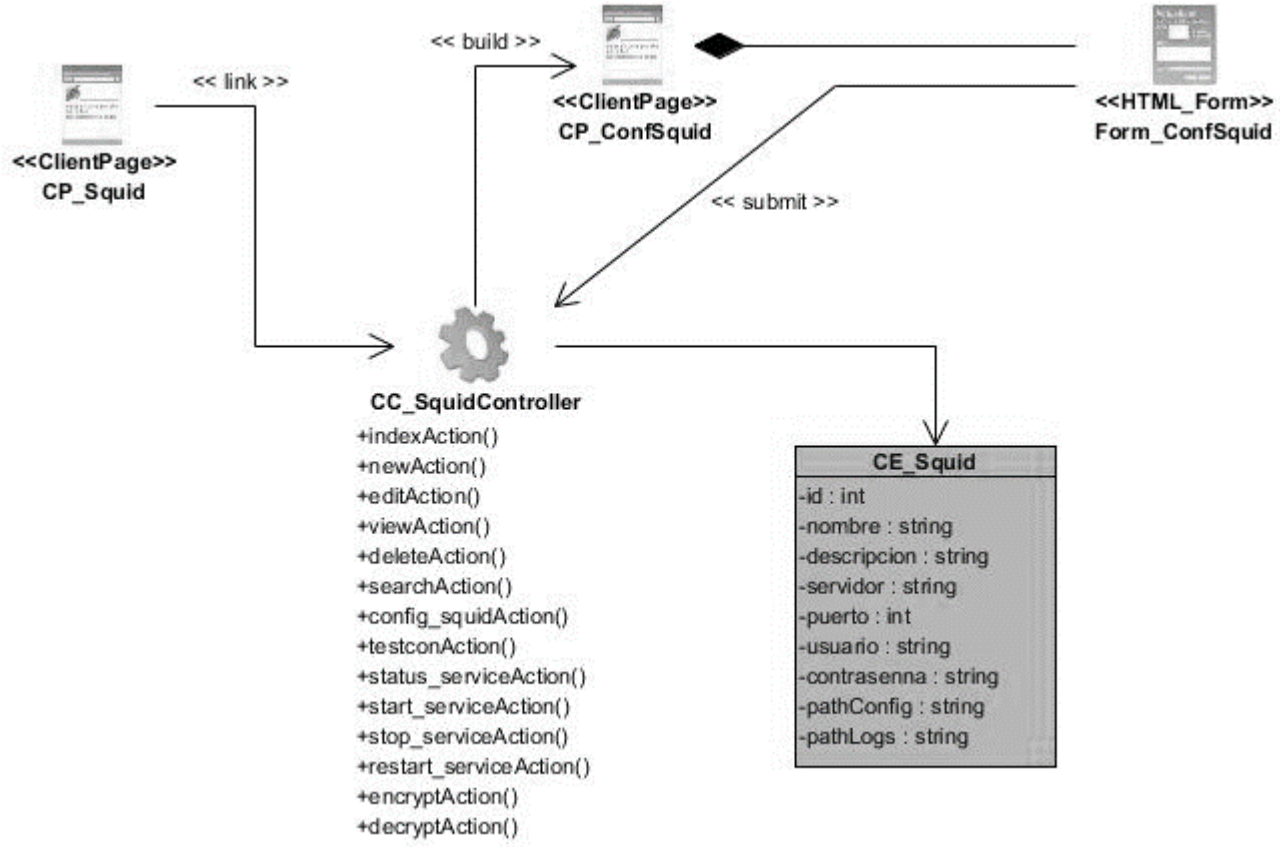



Figura 4. Diagrama de clases del diseño HU_4 Configurar servicios del *proxy Squid*

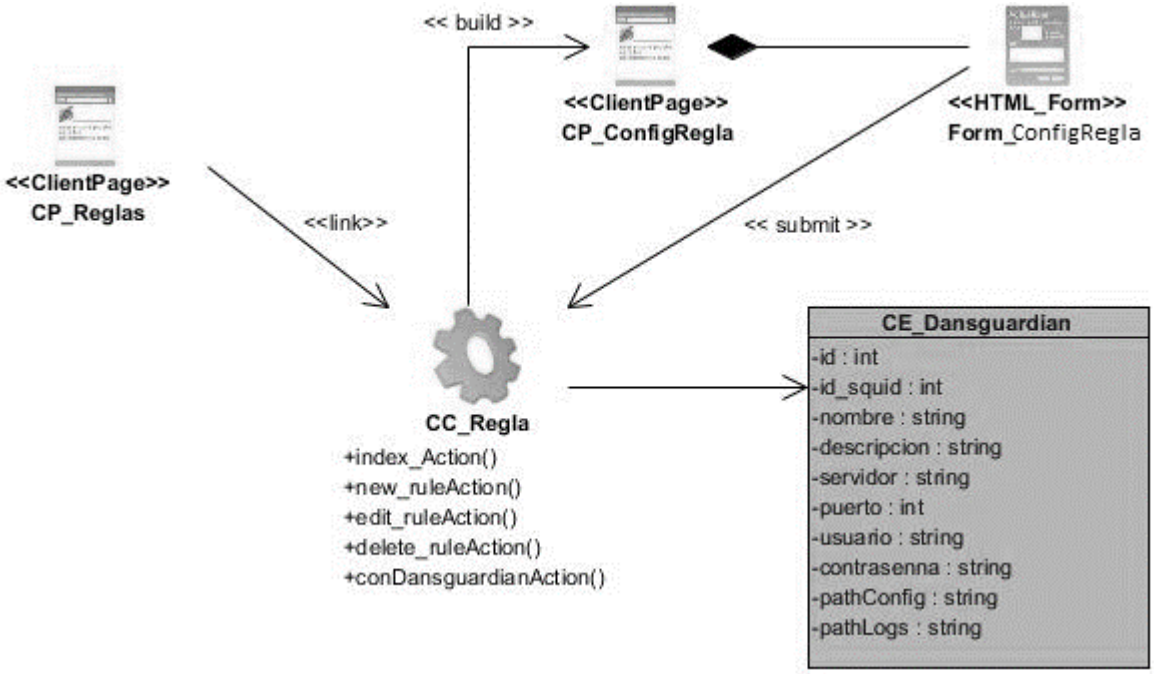



Figura 5. Diagrama de clases del diseño HU_13 Configurar reglas de filtrado

## 2.8. Diagramas de secuencia

Los diagramas de secuencia (DS) en el UML se utilizan principalmente para modelar las interacciones entre los actores y los objetos en un sistema, así como las interacciones entre los objetos (Sommerville, 2011). Para la presente investigación se generaron un total de siete (7) DS relacionados con los requisitos funcionales de prioridad alta, definidos en el subepígrafe 2.3.1. A continuación se muestran dos (2) relacionados con los RF 4 y RF 13 respectivamente, el resto puede ser consultado en el Anexo 3.

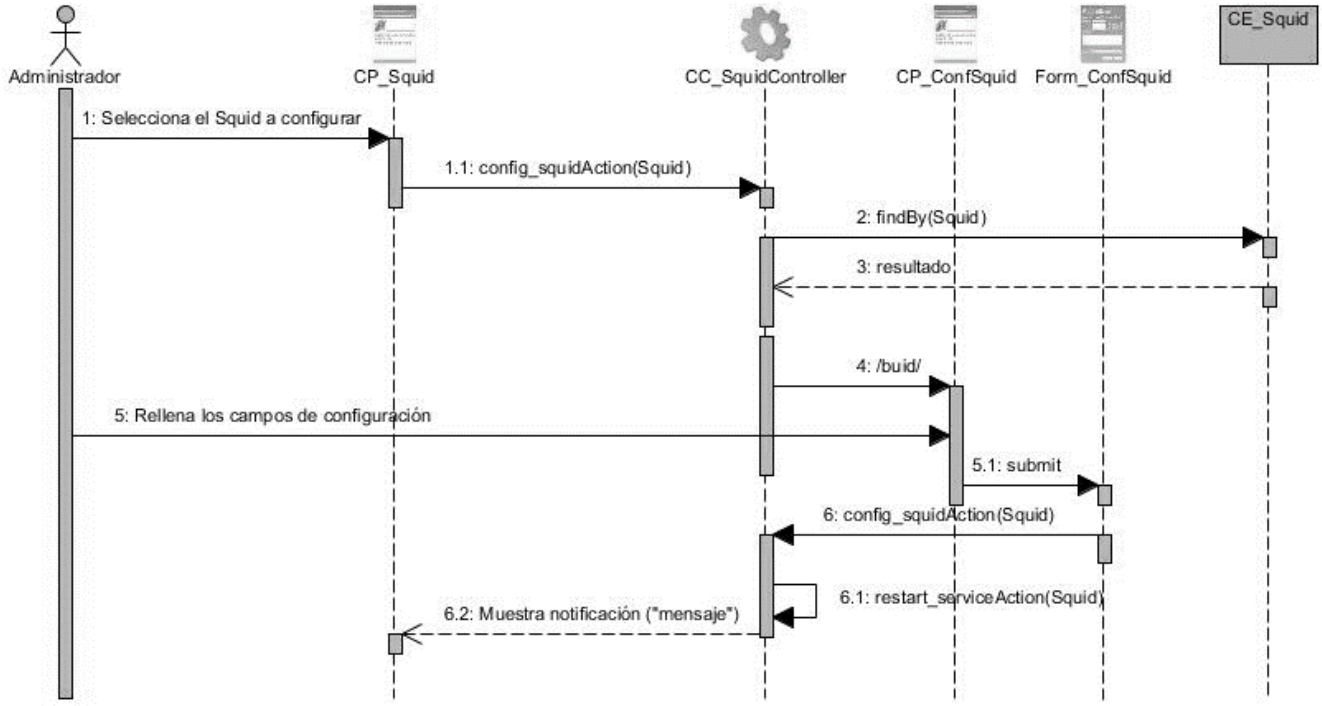



Figura 6. Diagrama de secuencia HU Configurar servicios del *proxy Squid*

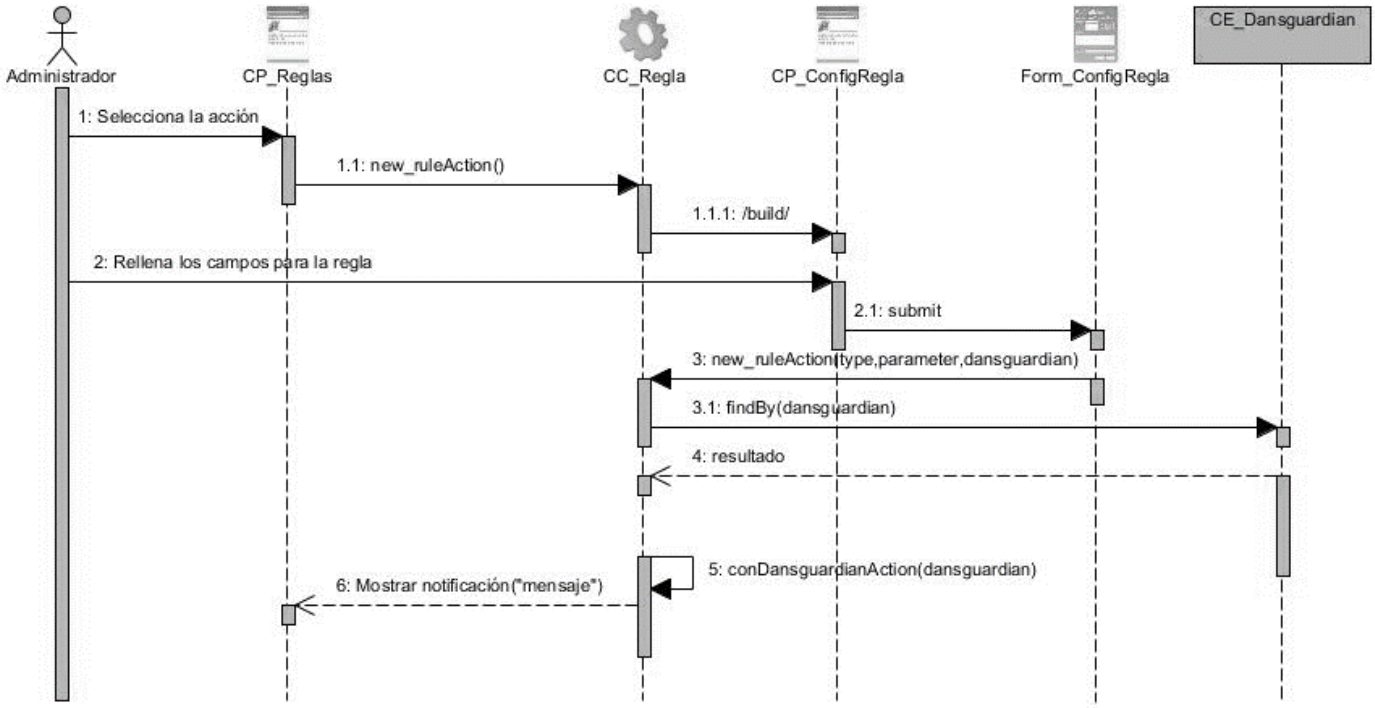



Figura 7. Diagrama de secuencia HU Configurar reglas de filtrado (escenario crear regla)

## 2.9. Modelo de datos

Un modelo de datos consiste en un conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los modelos de datos comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones (Zorrilla, 2011). El modelo de datos generado para la propuesta de solución de la presente investigación, como se observa en la Figura 8, está estructurado en tres (3) tablas, estas se explican a continuación:

**Usuario:** Registra la información de los usuarios del sistema. Se almacena el atributo que lo identifica (id), nombre(s), apellidos, usuario, contraseña, semilla para la encriptación de la contraseña (salt), correo electrónico, si usuario es activo o no (is_active) y su avatar.

**Squid:** Almacena la información que necesita el sistema Xilema Smart Keeper para realizar las conexiones remotas a un servidor *proxy Squid*. Se registra el identificador (id), nombre, descripción, servidor, puerto, usuario, contraseña, ruta del archivo de configuración (pathConf) y ruta de los ficheros *logs* (pathLogs).

**Dansguardian:** Almacena la información que necesita el sistema Xilema Smart Keeper para realizar las conexiones remotas al servidor donde se encuentre instalado el filtro de contenido asociado a un *proxy Squid*. De este se almacena su identificador (id), *proxy Squid* asociado (id_squid), nombre, descripción, servidor, puerto, usuario, contraseña, ruta del archivo de configuración (pathConf) y la ruta de los ficheros *logs* (pathLogs).

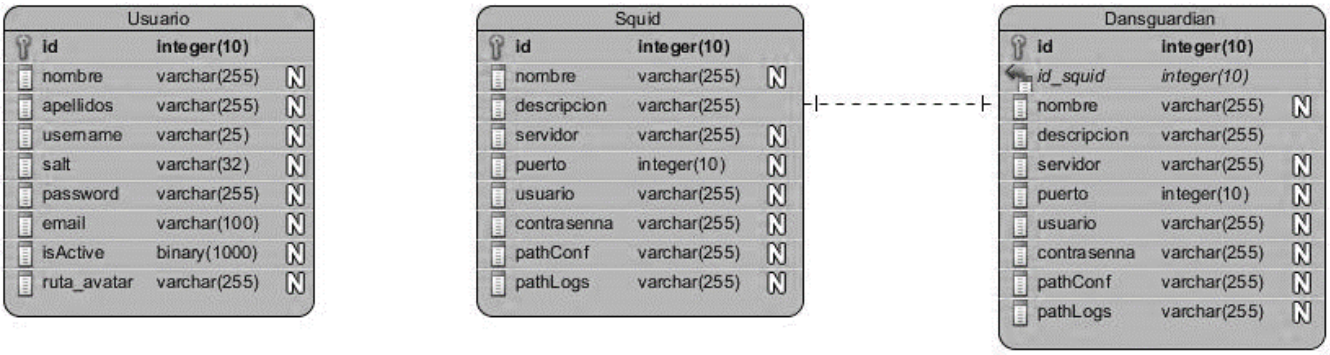



Figura 8. Modelo de datos de la propuesta de solución

## 2.10. Modelo de despliegue

Según Sarmiento (2016), un modelo de despliegue es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista de distribución de los artefactos del software en los destinos de

despliegue; se definen los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. La Figura 9 muestra el despliegue propuesto para el módulo:

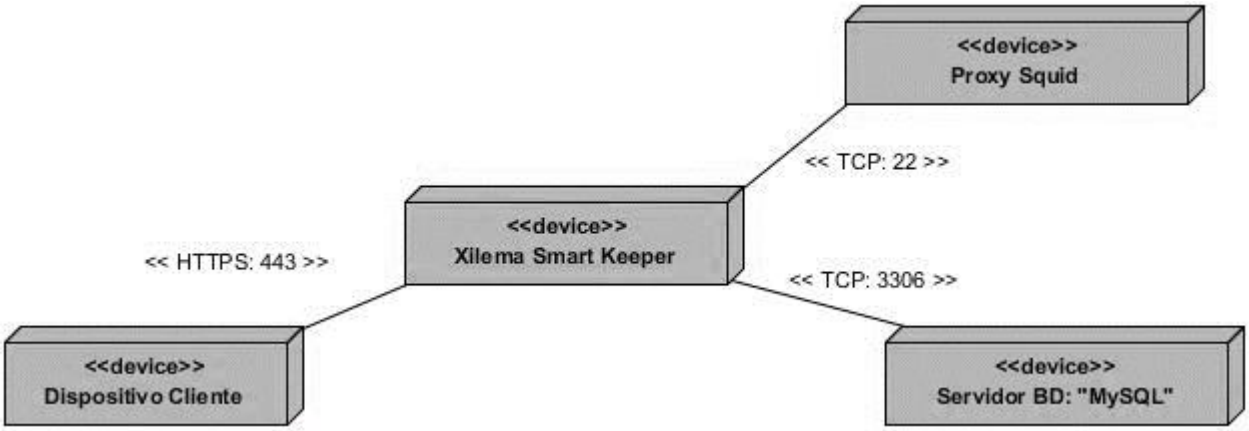



Figura 9. Modelo de despliegue

En este diagrama se define el nodo Dispositivo Cliente desde donde se originan las peticiones mediante el protocolo *HTTPS* al sistema Xilema Smart Keeper alojado en el servidor de aplicaciones web. Este servidor debe utilizar el sistema operativo Ubuntu 14.04 o superior y cuyas propiedades mínimas sean un procesador de 2.10 GHz y 2GB de memoria *RAM*. El sistema mantendrá una comunicación con los servidores *proxy Squid* por el protocolo *TCP* y el puerto 22 para las conexiones seguras del protocolo *SSH* y el servidor de Base de Datos a través del protocolo *TCP* por el puerto 3306, este utilizará el SGBD *MySQL* con una capacidad mínima de 80 GB de disco duro.

# CAPÍTULO 3: "IMPLEMENTACIÓN Y VALIDACIÓN DEL MÓDULO DE GESTIÓN DE SERVIDORES *PROXY SQUID* DEL SISTEMA XILEMA SMART KEEPER"

Con el objetivo de asegurar que el módulo propuesto funciona de acuerdo a los requisitos definidos y materializar en forma de componentes la fase de análisis y diseño, en el presente capítulo se describe la etapa de implementación y codificación del módulo. Se documentan los resultados obtenidos al aplicar las estrategias de pruebas, utilizadas para comprobar la calidad del software durante la etapa de validación.

## 3.1. Diagrama de Componentes

El diagrama de componentes proporciona una visión física de la construcción del sistema. Muestra la organización de los componentes software, sus interfaces y las dependencias entre ellos (Ramos, 2016). Basándose en la arquitectura de software que propone el marco de trabajo Symfony, seleccionada en la fase de análisis y diseño, la Figura 10 muestra el diagrama de componentes del módulo para la gestión de servidores proxy Squid del sistema Xilema Smart Keeper, a continuación, se describen los elementos que componen el diagrama Tabla 5.

Tabla 5. Descripción de los componentes del Diagrama de Componentes

| Componentes | | | | Descripción |
|---|---|---|---|---|
| Xilema Smart Keeper | Módulo de Gestión del *proxy Squid* | Modelo | Squid.php | Clase entidad que almacena los datos de los servidores *proxy Squid* para realizar las conexiones remotas y las configuraciones. |
| | | | Dansguardian.php | Clase entidad que almacena los datos de los componentes de filtrado *Dansguardian* para realizar las conexiones remotas al servidor donde este se encuentre instalado y realizar las configuraciones. |
| | | Controlador | SquidController.php | Clase controladora que contiene las acciones relacionadas con los servidores *proxy Squid*. Realiza acciones generales de gestión (insertar, eliminar, mostrar, editar y buscar), consulta y cambio de estado (iniciar, detener, reiniciar), conexión, validación y configuración del servicio. |
| | | | DansguardianController.php | Clase controladora que gestiona la conexión, administración y configuración del servicio Dansguardian. |

| | | | | | |
|---|---|---|---|---|---|
| | | | ReglaController.php | | Clase controladora encargada de gestionar las reglas de filtrado. |
| | | | Formularios | SquidType.php | Clase responsable de construir los formularios relacionados con los servidores *proxy Squid*. |
| | | | | DansguardianType.php | Clase responsable de construir los formularios relacionados con los componentes de filtrado *Dansguardian.* |
| | | Vista | conf_squid | index.html.twig | Muestra un listado de los servidores *proxy Squid* registrados en el sistema. |
| | | | | new_squid.html.twig | Permite registrar un *proxy Squid* en el sistema a través de un formulario |
| | | | | edit_squid.html.twig | Permite editar un *proxy Squid* a través de un formulario. |
| | | | | conf_squid.html.twig | Permite gestionar y modificar el fichero de configuración del *proxy Squid* mediante formularios. |
| | | | conf_ds | index.html.twig | Muestra un listado de los componentes de filtrado *Dansguardian* registrados en el sistema. |
| | | | | conf_ds.html.twig | Permite configurar y asociar un servidor *proxy Squid* a un componente de filtrado *Dansguardian*. |
| | | | conf_regla | index.html.twig | Muestra un listado con las reglas de filtrado registradas en el sistema |
| | | | | new_regla.html.twig | Permite registrar una regla de filtrado en el sistema. |
| | | | | edit_regla.html.twig | Permite editar un regla de filtrado en el sistema. |
| | | routing.yml | | | Contiene el mapa de rutas *URL* que se enlazan a las acciones de los controladores del módulo. |
| | | libssh2 | | | Librería *PHP* que permite realizar conexiones remotas a través del protocolo *SSH.* |
| | config.yml | | | | Contiene las configuraciones generales del sistema Xilema Smart Keeper. |

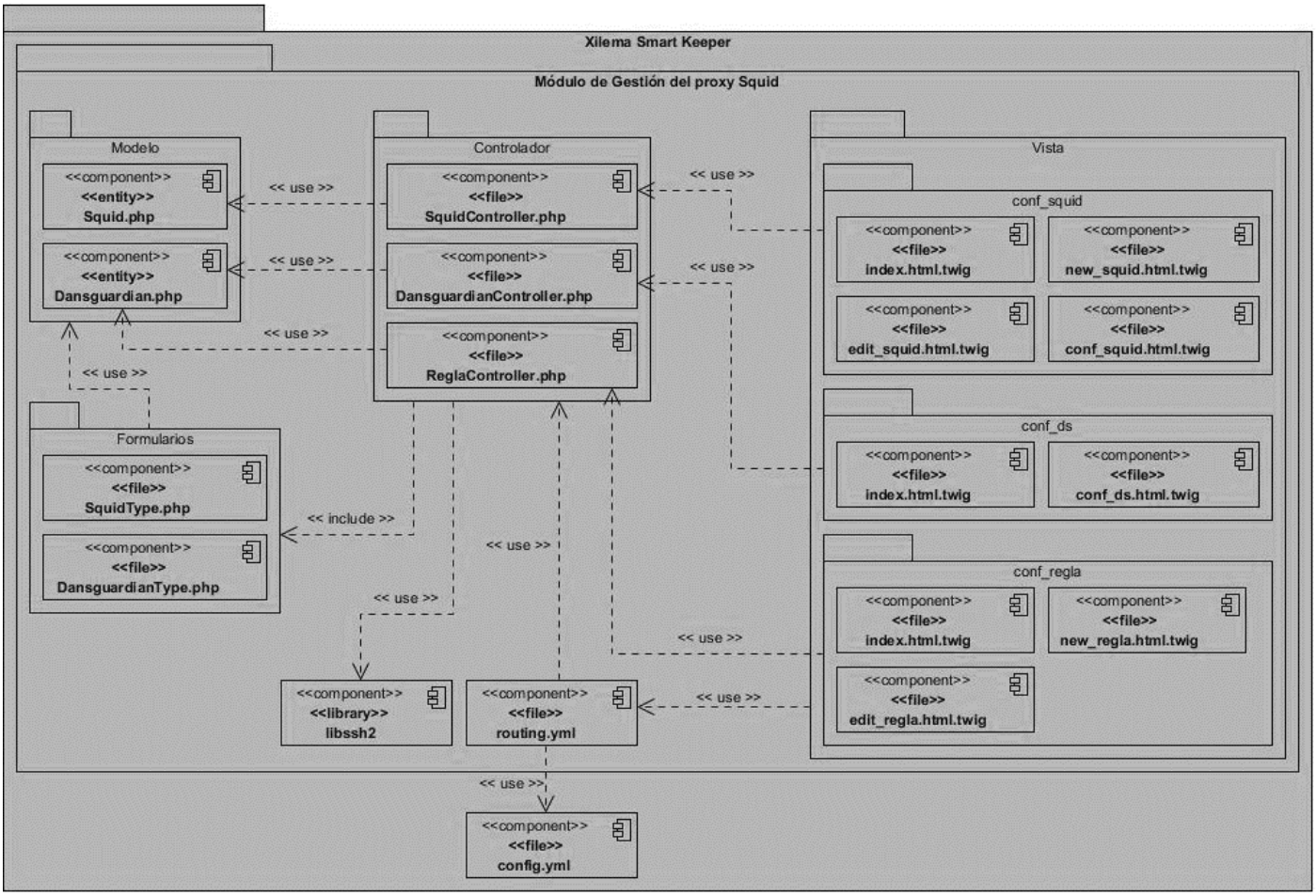
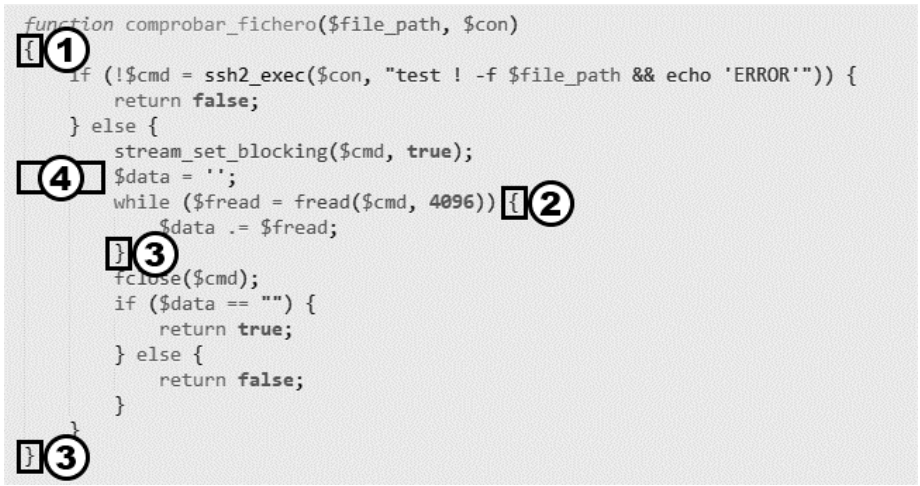


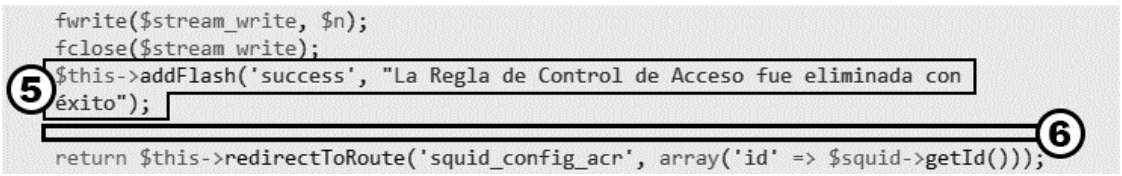

Figura 10. Diagrama de componentes

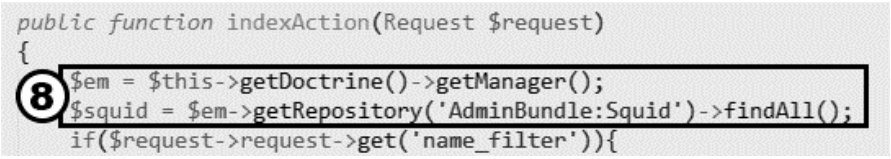
## 3.2. Estándares de Codificación

Los estándares de codificación son un conjunto de convenciones para lograr uniformidad en el código de fuente de un software. Permiten una mayor comprensión, modificación, calidad y mantenibilidad del código generado independientemente de su autor. A continuación, de definen los estándares de codificación utilizados durante la implementación del módulo, basándose en los que propone el lenguaje *PHP* para el marco de trabajo *Symfony* (PHP-Fig, 2018).

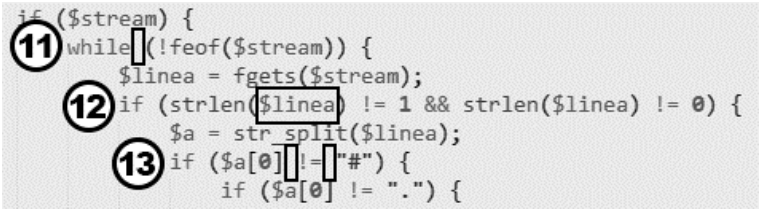
Tabla 6. Estándares de codificación

<table>
<tr><th>Tipo de Estándar</th><th>Descripción</th></tr>
<tr><td>Organización del código</td><td>
1. La apertura de llaves para clases y funciones deben estar en la próxima línea después de su declaración.<br>
2. Las aperturas de llaves en las estructuras de control se colocan en la misma línea.<br>
3. El cierre de llaves para clases, funciones y estructuras de control deben estar en la próxima línea después del bloque de código.<br>
4. La identación se realiza con tabulación y no con 4 espacios en blanco.<br>


</td></tr>
<tr><td>Líneas</td><td>
5. El tamaño máximo de una línea no excederá de ochenta (80) caracteres.<br>
6. Se deberá agregar una línea en blanco delante de la sentencia return.<br>


7. Se puede agregar líneas en blanco para mejorar la legibilidad e indicar bloques de códigos relacionados.<br>
8. Debe haber una declaración por línea.<br>


</td></tr>
</table>

| | |
|---|---|
| Codificación | 9. Los archivos *PHP* deben utilizar solo las etiquetas de apertura *<?php* omitiendo las etiquetas de cierre *?>*<br>10. Utilizar la codificación *UTF-8.* |
| Espacios en blanco en expresiones y sentencias | 11. Se debe utilizar un espacio en blanco después de la palabra clave en las estructuras de control.<br>12. No deben existir espacios en la apertura y cierre de paréntesis.<br>13. Debe usarse un espacio en blanco entre los operadores lógicos, aritméticos, de comparación y asignación<br>  |
| Convenciones de Nombramientos | 14. Se debe utilizar el estilo de escritura *camelCase* para la declaración de variables y no guiones bajos para separar palabras.<br>15. Utilizar letras mayúsculas sostenida para declarar contantes y separar las palabras con guiones bajos.<br>16. Se debe usar el estilo de escritura *StudlyCaps* para los nombres de clases y funciones.<br>  |

## 3.3. Validación de la propuesta de solución

Según la ISO/IEC/IEEE International Standard (2017), la fase de validación es el proceso de evaluar la calidad del software, con el objetivo de detectar posibles errores y corregirlos enfocándose en la lógica interna y los requerimientos especificados. A continuación, se documentan los resultados de las pruebas realizadas al Módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper, con el objetivo de evaluar su calidad y correcto funcionamiento.

### 3.3.1. Pruebas funcionales

Las pruebas funcionales tienen como objetivo validar que las funcionalidades implementadas cumplan con las especificaciones de los requisitos definidos. Para esta prueba, el autor selecciona la técnica de Caja

Negra, que según Pressman (2010) permite derivar un conjunto de condiciones de entrada, que revisarán por completo todos los requisitos funcionales de una aplicación con el objetivo encontrar errores de funciones incorrectas o ausentes, errores de interfaz, de comportamiento o rendimiento. Para aplicar las pruebas funcionales al módulo se diseñaron siete (7) Casos de Prueba (CP) que corresponden a los requisitos funcionales de prioridad alta. A continuación, se muestran dos (2) CP correspondientes a los RF4 y RF20 respectivamente, el resto puede ser consultada en el Anexo 4.

Tabla 7. Descripción de las variables para el Caso de Prueba 1

| Nº | Nombre del campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Puerto de *proxy* | Campo numérico | No | Longitud mínima 2 dígitos, longitud máxima 4 dígitos. Permite solo caracteres numéricos. |
| 2 | Límite de uso de memoria | Campo compuesto (campo numérico y campo de selección) | No | El campo numérico: rango mínimo 1. Permite solo caracteres numéricos. El campo de selección: Puede tomar uno de los siguientes valores KB, MB y GB. |
| 3 | Directorio | Campo de texto | No | Permite todos los caracteres. |
| 4 | Tipo | Campo de selección | No | Puede tomar uno de los siguientes valores UFS, DISKD, UFS Asíncrono, COSS. |
| 5 | Tamaño (MB) | Campo numérico | No | Rango mínimo 100 MB. Permite solo caracteres numéricos. |
| 6 | Directorios de 1er Nivel | Campo numérico | No | Rango mínimo 16 directorios. Permite solo caracteres numéricos. |
| 7 | Directorios de 2do Nivel | Campo numérico | No | Rango mínimo 256 directorios. Permite solo caracteres numéricos. |

Tabla 8. Caso de Prueba del RF4_Configurar servicios del *proxy Squid*

| **Caso de Prueba 1:** SC RF4_Configurar servicios del *proxy Squid.* | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema y debe existir al menos un *proxy Squid* disponible (estado distinto a "Sin Conexión"). | | | | | | | | | | |
| **Escenario** | **Descripción** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **Respuesta del Sistema** | **Flujo Central** |
| EC 1.1 Configurar aspectos generales del *proxy Squid* de forma correcta. | El módulo configura un proxy Squid de forma correcta. | V<br>8080 | V<br>256 KB | V<br>*/var/spool/squid* | V<br>UFS | V<br>512 | V<br>28 | V<br>512 | Muestra un mensaje de confirmación "El proxy Squid fue configurado con éxito". | En el menú superior del sistema:<br>1. El usuario accede a la opción Configuración.<br>2. El usuario selecciona la opción *Proxy Squid y* el módulo muestra una lista de los servidores *proxy Squid* registrados.<br>3. El usuario selecciona la opción de Configuraciones (columna Opciones 3er botón) de un proxy Squid disponible.<br>4. El sistema muestra un formulario con la información de la configuración actual del proxy Squid seleccionado.<br>5. El usuario introduce los nuevos valores y presiona el botón Guardar y Salir. |
| EC 1.2 Configurar aspectos generales del *proxy Squid* de forma incorrecta. | El módulo no configura un *proxy Squid* de forma incorrecta. | I<br>35697 | I<br>0 GB | V<br>*/var/spool/squid* | V<br>UFS | I<br>90 | I<br>10 | I<br>200 | Muestra un mensaje de error "No se pudieron realizar los cambios, revise los parámetros introducidos", señala en rojo para cada caso los campos incorrectos y muestra los siguientes errores. Puerto de *proxy*: "Este campo no debe tener más de 4 dígitos". Límite de uso de memoria: "El uso de la memoria debe ser mayor a 1 unidad". Tamaño: "El tamaño debe ser superior a 100 MB". Directorios de 1er nivel: "Deben ser más | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | de 16 directorios". Directorios de 2do nivel: "Deben ser más de 256 directorios. | 1. El usuario accede a la opción Configuración.<br>2. El usuario selecciona la opción *Proxy Squid y* el módulo muestra una lista de los servidores *proxy Squid* registrados.<br>3. El usuario selecciona la opción de Configuraciones (columna Opciones 3er botón) de un proxy Squid disponible.<br>4. El sistema muestra un formulario con la información de la configuración actual del proxy Squid seleccionado.<br>5. El usuario introduce los nuevos valores y presiona el botón Guardar y Salir. |
| EC 1.3 Configurar aspectos generales del *proxy Squid* con campos vacíos. | El módulo no configura un servidor *proxy Squid* con campos obligatorios vacíos. | I | I | I | V | I | I | I | Muestra un mensaje de error "No se pudieron realizar los cambios, revise los parámetros introducidos", señala en rojo para cada caso los campos obligatorios vacíos y muestra debajo del campo el siguiente mensaje "Este campo es requerido" | |
| | | | MB | | UFS | | | | | |

Tabla 9. Descripción de las variables para el Caso de Prueba 2

| Nº | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | Frase | Campo de texto | No | Permite todos los caracteres. |
| 2 | Servidor | Campo de selección | No | Permite seleccionar uno de los servidores activos. |

Tabla 10. Caso de Prueba del RF13_Configurar reglas de filtrado

| **Caso de Prueba 2:** SC RF13_Configurar reglas de filtrado. | | | | | |
|---|---|---|---|---|---|
| **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema y debe existir al menos un sistema de filtrado *Dansguardian* disponible (estado distinto a "Sin Conexión"). | | | | | |
| **Escenario** | **Descripción** | **1** | **2** | **Respuesta del sistema** | **Flujo Central** |
| EC 2.1 Crear una regla de filtrado por frases de forma correcta. | Inserta una nueva regla de filtrado de forma correcta. | V<br>hacer,bomba | V<br>Seleccionar cualquier servidor activo | Muestra un mensaje de confirmación "La frase se adicionó con éxito" | En el menú superior del sistema:<br>1. El usuario accede a la opción Reglas de Filtrado.<br>2. El usuario selecciona la opción Frases *y*<br>el módulo muestra dos paneles uno a la derecha y otro a la izquierda. El de la izquierda muestra una lista con las frases registradas en el sistema y en el de la derecha un formulario para insertar nuevas frases.<br>3. El usuario introduce los valores de las reglas y presiona el botón Adicionar. |
| EC 2.2 Crear una regla de filtrado por frases con campos vacíos. | El módulo no inserta una regla de filtrado con campos obligatorios vacíos. | I | I | Muestra un mensaje de error "Ocurrió un error al adicionar la frase" | |

**Resultado de las pruebas funcionales**

Se obtuvo en una primera iteración un total de ocho (8) no conformidades, divididas en cinco (5) de ortografía, dos (2) de funcionalidad y una de excepción. Las deficiencias identificadas fueron resueltas en la primera iteración. Para una segunda iteración no se identificaron nuevas no conformidades obteniendo resultados satisfactorios, estos son expuestos en la Figura 11.

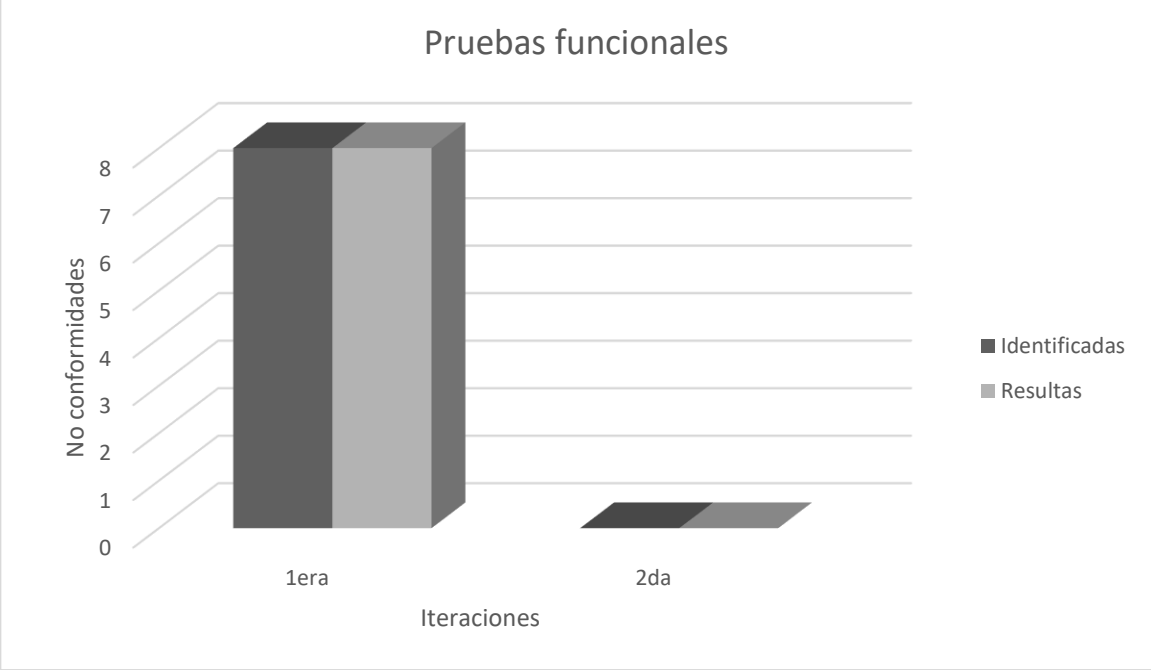



*Figura 11. Resultado de las pruebas funcionales*

Las no conformidades funcionales, están relacionadas la validación de los formularios al no mostrar la notificación de error especificada en las respuestas del sistema. La presencia de esta no conformidad se origina en la ausencia de estas notificaciones en los métodos de las clases controladoras, para resolverlo fueron incluidos en el código de fuente. La no conformidad de excepción se obtenía por la validación del campo Servidor del CP2, al dejar este campo en blanco lanzaba un error fatal que interrumpía el flujo normal del módulo y hacia vulnerable su seguridad, fue resulta al añadir una aserción para el campo Servidor, donde especifica que no puede ser nulo. Las no conformidades de ortografía estaban relacionadas con problemas de acentuación en las etiquetas de los campos y en las notificaciones de usuario. Resueltas las insuficiencias detectadas, se obtiene un módulo funcional que cumple con los requisitos especificados.

### 3.3.2. Pruebas de seguridad

Las pruebas de seguridad miden la confidencialidad, integridad y disponibilidad de la información, con el objetivo de identificar amenazas, riesgos y la probabilidad de enfrentarse a ataques informáticos que descubran y exploten las vulnerabilidades de la aplicación (Quality, 2016). Para realizar las pruebas de seguridad al módulo, se utiliza la herramienta *Acunetix Web Vulnerability Scanner,* en una primera iteración, la herramienta realiza treinta mil seiscientos dieciséis (30 616) ataques informáticos y se generan setenta y dos (72) alertas de vulnerabilidad, divididas en sesenta (60) de nivel medio y doce (12) de carácter informacional. Las alertas identificadas fueron corregidas en la primera iteración, para una segunda iteración, se igualaron la cantidad de ataques a la primera iteración y no se generan nuevas alertas.

Las alertas de nivel medio estuvieron relacionadas con cincuenta y siete (57) mensajes de error provocados por el modo de ejecución *DEV* para el desarrollo de aplicaciones en el marco de trabajo Symfony, de los que se puede obtener información sensible de los registros *DEBUG* del módulo. Se identificaron tres (3) formularios *HTML* vulnerables a la falsificación de petición en sitios cruzados (CSFR, por sus siglas en inglés). Los formularios fueron comprobados de forma manual como recomienda la herramienta *Acunetix* y se identificó que no pueden ser utilizados por el atacante para alterar el sistema ni obtener información, se denomina estas vulnerabilidades como falsos positivos.

En las alertas de carácter informacional se identificaron once (11) enlaces rotos, estos fueron eliminados por pertenecer a la estructura original de marco de trabajo Symfony y no ser utilizados en el módulo. Se obtuvo una alerta relacionada con el fichero *parameters.yml* que contiene información de usuario y contraseña para conectar con el servidor de base de datos *MySQL*, esta vulnerabilidad fue solucionada limitando el acceso a este fichero, en las listas de control de acceso del módulo. Los resultados obtenidos fueron satisfactorios y se obtiene un módulo que cumple con los requisitos de seguridad avalados por la Dirección de Calidad de la UCI (Carrazana, 2017).

### 3.3.3. Pruebas de integración

Las pruebas de integración aseguran que los distintos componentes de un software interactúan entre sí de forma correcta. Con el objetivo de validar la compatibilidad y el funcionamiento de las distintas interfaces que componen al Módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper se toman acciones relacionadas con:

- Validación de conexión a servidores *proxy Squid* (Ver Tabla 11).

- Validación de conexión a componentes de filtrado *Dansguardian* para la gestión de las reglas de filtrado (Ver Tabla 12).

Tabla 11. Caso de Prueba 3 Comprobar conexión del servidor *proxy Squid*

| **Caso de Prueba 3:** SC RF6_Comprobar conexión del servidor *proxy Squid.* | | | |
|---|---|---|---|
| **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema y debe haber llenado correctamente el formulario para Insertar un nuevo *proxy Squid*. | | | |
| **Escenario** | **Descripción** | **Respuesta del sistema** | **Flujo Central** |
| EC 3.1. Probar la conexión del servidor *proxy Squid* de forma correcta | El sistema comprueba la conexión al servidor *proxy Squid* de forma correcta, luego de haber validado los campos del formulario para insertar un *proxy Squid.* | Muestra un mensaje de confirmación: "La conexión fue exitosa". | En el menú superior del sistema:<br>1. El usuario accede a la opción Configuración.<br>2. El usuario selecciona la opción *Proxy Squid y* el módulo muestra una lista de los servidores *proxy Squid* registrados.<br>3. El usuario selecciona la opción Adicionar en la parte superior derecha del panel principal.<br>4. El sistema muestra un formulario para adicionar un nuevo *proxy Squid*.<br>5. El usuario introduce valores válidos y presiona el botón Probar Conexión. |

Tabla 12. Caso de Prueba 4: Comprobar conexión con el componente de filtrado *Dansguardian*

| **Caso de Prueba 4:** Comprobar conexión del componente de filtrado Dansguardian. | | | |
|---|---|---|---|
| **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema y debe haber llenado correctamente el formulario para Insertar un nuevo *Dansguardian*. | | | |
| **Escenario** | **Descripción** | **Respuesta del sistema** | **Flujo Central** |

| | | | |
|---|---|---|---|
| EC 3.1. Probar la conexión del componente de filtrado *Dansguardian* de forma correcta | El sistema comprueba la conexión al componente de filtrado *Dansguardian* de forma correcta, luego de haber validado los campos del formulario para insertar un *Dansguardian.* | Muestra un mensaje de confirmación: "La conexión fue exitosa". | En el menú superior del sistema:<br>1. El usuario accede a la opción Configuración.<br>2. El usuario selecciona la opción *Dansguardian y*<br>el módulo muestra una lista de los componentes de filtrados *Dansguardian* registrados.<br>3. El usuario selecciona la opción Adicionar en la parte superior derecha del panel principal.<br>4. El sistema muestra un formulario para adicionar un nuevo *Dansguardian*.<br>5. El usuario introduce valores válidos y presiona el botón Probar Conexión. |

En la aplicación de estas pruebas no se identificaron no conformidades, lo que valida que existe una correcta integración de los componentes internos del módulo.

### 3.3.4. Validación de la hipótesis científica de la investigación

Con el objetivo de evaluar la fiabilidad de la hipótesis científica de la investigación se aplica el Criterio de Expertos mediante el método Delphi. Este método consiste en una técnica de obtención de información, basada en la consulta de expertos en un área de estudio, con el fin de obtener la opinión de consenso más fiable proveniente del conocimiento y la experiencia de los participantes del grupo (Reguant y otros, 2016).

De la hipótesis científica planteada en la presente investigación, se define como variable independiente: el módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper, esta consiste en un componente que permite administrar y configurar de forma remota y centralizada los servicios de servidores *proxy Squid*. Como variable dependiente: facilitar el trabajo a los administradores del sistema, refiriéndose a la optimización del tiempo, esfuerzo y facilidad en las tareas de gestión de los servidores *proxy Squid* del sistema.

Para aplicación del Método Delphi se emplearán los siguientes pasos:

- Identificación y selección de posibles expertos.
- Aplicación de encuestas a expertos.
- Valoración de la información obtenida.

Identificación y selección de posibles expertos

Se identifican cinco (5) posibles expertos en materias relacionadas con la administración y configuración de servidores *proxy Squid*. Con el objetivo de valorar el nivel de experiencia que poseen, se aplica un cuestionario de autovaloración (ver Anexo 5) de los niveles de información y argumentación que tienen sobre el tema en cuestión. En la Tabla 13 se resumen los resultados obtenidos.

Tabla 13. Resultado de niveles de conocimiento de posibles expertos

| Expertos | Nivel de conocimiento o información del tema | | | | | | | | | | Kc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| Miguel Ángel Chávez Alfonso | | | | | | | | X | | | 0.8 |
| Rubén Reynaldo Bonachea | | | | | | | | | X | | 0.9 |
| Yadier Perdomo Cuevas | | | | | | | | X | | | 0.8 |
| Joelsy Porven Rubier | | | | | | | | X | | | 0.8 |
| Adrián Hernández Yeja | | | | | | | | | X | | 0.8 |

Para calcular el Coeficiente de Conocimiento o Información (Kc) del experto se utiliza ecuación 3.1:

$Kc = n(0.1)$ (3.1)

Donde *n* es el nivel de conocimiento o información seleccionado por el experto. Para determinar, además, el coeficiente de argumentación o fundamentación de cada experto es necesario utilizar como factores, los que aparecen en la Tabla 14.

Tabla 14. Patrón para el cálculo de Coeficiente de Argumentación o Fundamentación

| Fuentes de Argumentación | Alto | Medio | Bajo |
|---|---|---|---|
| Análisis teóricos realizados por usted | 0.3 | 0.2 | 0.1 |
| Experiencia obtenida | 0.5 | 0.4 | 0.2 |
| Trabajos de autores nacionales | 0.05 | 0.05 | 0.05 |
| Trabajos de autores extranjeros | 0.05 | 0.05 | 0.05 |
| Su conocimiento del estado del problema en el extranjero | 0.05 | 0.05 | 0.05 |
| Su Intuición | 0.05 | 0.05 | 0.05 |

El Coeficiente de Argumentación (Ka) de cada experto se calcula con la ecuación 3.2:

$Ka = \sum_{i=1}^{n} n_i$ (3.2)

Donde $n_i$ es el valor obtenido en la fuente de argumentación *i* (de 1 hasta n) y *n* es la cantidad de fuentes de argumentación. Al aplicar la ecuación se obtienen los siguientes resultados:

- Miguel Ángel Chávez Alfonso. $Ka = 0.8$
- Rubén Reynaldo Bonachea. $Ka = 0.9$
- Yadier Perdomo Cuevas. $Ka = 0.8$
- Joelsy Porven Rubier. $Ka = 0.8$
- Adrián Hernández Yeja. $Ka = 0.9$

Con los Coeficientes de Conocimiento (Kc) y Argumentación (Ka) se puede obtener finalmente el Coeficiente de Competencia (K) para determinar que expertos se toman en consideración para trabajar en la investigación. Este es calculado con la ecuación 3.3:

$K = 0.5(Kc + Ka)$ (3.3)

El resultado obtenido se muestra en la Tabla 15 y es valorado de la siguiente manera:

- El coeficiente es alto si $0.8 < K < 1$
- El coeficiente es medio si $0.5 < K < 0.8$
- El coeficiente es bajo si $K < 0.5$

Tabla 15. Resultado de niveles de competencia de posibles expertos

| Expertos | K | Valoración |
|---|---|---|
| Miguel Ángel Chávez Alfonso | 0.8 | Alto |
| Rubén Reynaldo Bonachea | 0.9 | Alto |
| Yadier Perdomo Cuevas | 0.8 | Alto |
| Joelsy Porven Rubier | 0.8 | Alto |
| Adrián Hernández Yeja | 0.9 | Alto |

Fueron seleccionados todos los expertos al ser su nivel de competencia alto, finalmente el panel de expertos quedó conformado de la siguiente manera (Ver Tabla 16):

Tabla 16. Expertos seleccionados para la validación de la hipótesis científica

| Nombre y Apellidos | Entidad | Años de Experiencia |
|---|---|---|
| Joelsy Porven Rubier | Dirección de redes y servicios telemáticos (UCI) | 15 |
| Yadier Perdomo Cuevas | Dirección de redes y servicios telemáticos (UCI) | 13 |
| Adrián Hernández Yeja | Dirección de redes y servicios telemáticos (UCI) | 9 |
| Miguel Ángel Chávez Alfonso | Centro de Ideoinformática (Facultad 1) | 8 |
| Rubén Reynaldo Bonachea | Centro de Ideoinformática (Facultad 1) | 6 |

Aplicación de encuestas a expertos:

Seleccionado el panel de expertos que trabajará en la validación de la hipótesis científica de la investigación, se somete a su consideración una encuesta (Ver Anexo 6), para evaluar el Módulo de gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper. La Tabla 17 resume el resultado de los juicios emitidos por los expertos, de acuerdo a las siguientes sentencias:

1. Facilidad de administración y configuración de servidores *proxy Squid*.
2. Administración y configuración remota de los servicios del servidor *proxy Squid*.
3. Facilidad en la gestión de las reglas de filtrado.
4. Rapidez en la respuesta de peticiones.
5. Presentación de una interfaz agradable e intuitiva para el usuario.

Para el procesamiento y análisis de la información obtenida se clasificaron las respuestas en las siguientes categorías evaluativas: muy adecuado (MA), adecuado (A), poco adecuado (PA), inadecuado (I).

Tabla 17. Resultado de las encuestas realizadas a los expertos

| Sentencias | Categorías Evaluativas | | | | Total |
|---|---|---|---|---|---|
| | MA | A | PA | I | |
| 1 | 5 | 0 | 0 | 0 | 5 |
| 2 | 4 | 1 | 0 | 0 | 5 |
| 3 | 2 | 3 | 0 | 0 | 5 |
| 4 | 3 | 2 | 0 | 0 | 5 |
| 5 | 5 | 0 | 0 | 0 | 5 |

Expuestos los criterios de los expertos en cada categoría evaluativa para las sentencias de la encuesta, se aplican los siguientes pasos hasta llegar a la conclusión de valoración de cada sentencia:

- Obtención de la tabla de frecuencia acumulada (Ver Tabla 18).
- Obtención de la tabla de frecuencia relativa acumulativa (Ver Tabla 19).
- Asignación del valor de la imagen que corresponde a cada frecuencia relativa acumulativa, por la inversa de la curva normal (Tabla Z de la distribución normal) y obtención de puntos mediante el cálculo N-P para el promedio relativo (Ver Tabla 20).

Tabla 18. Frecuencia acumulada de los datos primarios obtenidos

| Sentencias | Categorías Evaluativas | | | |
|---|---|---|---|---|
| | MA | A | PA | I |
| 1 | 5 | 5 | 5 | 5 |
| 2 | 4 | 5 | 5 | 5 |
| 3 | 2 | 5 | 5 | 5 |
| 4 | 3 | 5 | 5 | 5 |
| 5 | 5 | 5 | 5 | 5 |

Tabla 19. Frecuencia relativa acumulativa de los datos primarios obtenidos

| Sentencias | Categorías Evaluativas | |
|---|---|---|
| | MA | A |
| 1 | 1 | 1 |
| 2 | 0.8 | 1 |
| 3 | 0.4 | 1 |
| 4 | 0.6 | 1 |
| 5 | 1 | 1 |

Tabla 20. Imagen de la frecuencia relativa acumulativa de los datos primarios obtenidos

| Sentencias | Categorías Evaluativas | | Suma | Promedio | Promedio relativo |
|---|---|---|---|---|---|
| | MA | A | | | |
| 1 | 3.49 | 3.49 | 6.98 | 3.49 | -0.96 |
| 2 | 0.85 | 3.49 | 4.34 | 2.17 | 0.36 |
| 3 | -0.25 | 3.49 | 3.24 | 1.62 | 0.91 |
| 4 | 0.27 | 3.49 | 3.73 | 1.87 | 0.66 |
| 5 | 3.49 | 3.49 | 6.98 | 3.49 | -0.96 |
| Puntos de Corte | 1.57 | 3.49 | 25.27 | | |

Al analizar los promedios relativos (N-P) de las sentencias, se puede observar que ninguno excede el límite superior o punto de corte de la categoría evaluativa MA. Esto demuestra que los expertos con un nivel de concordancia del 100% clasifican con el más alto nivel de adecuación (Muy Adecuado) las sentencias evaluadas, lo que apoya la hipótesis científica planteada en la presente investigación. Todos los indicadores fueron evaluados satisfactoriamente lo que evidencia la calidad y el alto valor del Módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper.

# CONCLUSIONES

- El estudio y análisis del estado del arte identificó los parámetros necesarios para la configuración de servidores *proxy Squid* del sistema Xilema Smart Keeper.
- La selección de la librería SSH2-PHP para las conexiones remotas, garantizan seguridad en la transferencia de información y estabilidad en las conexiones a los servidores *proxy Squid* por el protocolo SSH.
- La utilización de las herramientas y tecnologías identificadas y la puesta en práctica de los estándares de codificación definidos, permitieron implementar el módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper.
- Las pruebas de software solucionaron los errores detectados en el módulo y demuestran que es una solución funcional, integrable y segura.
- El criterio de expertos demostró con un alto nivel de concordancia que la propuesta desarrollada facilita el trabajo a los Administradores de Red por la centralización de los servicios en tareas de administración y configuración del servidor *proxy Squid*.

# RECOMENDACIONES

Para el desarrollo de futuras investigaciones relacionadas con la presente, se recomienda:

- Permitir la gestión de las reglas de filtrado si los servicios no se encuentran operativos.
- Adicionar una funcionalidad que permita la autenticación de las conexiones remota a través de llaves públicas.

# BIBLIOGRAFÍA

**ABURY, Christopher.** HTML5 y CSS3: para sitios con diseño web responsive. Barcelona: Ediciones ENI, 2014. pp. 326. ISBN: 978-2-7460-9290-7.

**ÁLVAREZ, Sara.** Conceptos básicos y definiciones sobre programación [En línea]. 2016 [Fecha de consulta: 14 noviembre 2017]. Disponible en: https://desarrolloweb.com/articulos/2357.php.

**APACHE.** What is the Apache HTTP Server Project? [En línea]. Apache Software Foundation, 2017 [Fecha de consulta: 24 de enero 2018]. Disponible en: https://httpd.apache.org/ABOUT_APACHE.html.

**ARAB, Elías y DÍAZ, Alejandra.** Impacto de las redes sociales e internet en la adolescencia: aspectos positivos y negativos. Revista Médica Clínica Las Condes [En línea]. 2015. vol. 26, no. 1, pp. 7-13. [Fecha de consulta: 25 noviembre 2017]. ISSN: 0716-8640. Disponible en: https://www.sciencedirect.com/science/article/pii/S0716864015000048.

**ARIAS, Ángel.** Bases de Datos con MySQL. 2.ª ed. IT Campus Academy, 2015. pp. 196. ISBN: 978-1515194392.

**BARRIOS, Joel.** Configuración de Servidores con GNU/Linux [En línea]. México, D.F: Alcance Libre, 2017 [Fecha de consulta: 12 febrero 2018]. Disponible en: http://www.alcancelibre.org/filemgmt/visit.php?lid=1.

**BEATI, Hernan.** HTML5 y CSS3-Para diseñadores. Buenos Aires: Alfaomega Grupo Editor, 2016. pp. 348. ISBN: 978-987-1609-67-3.

**BENEDICO, Yoel y CRESPO, Yankiel.** Sistema para el control de trazas de un servidor proxy. Revista Universidad&Ciencia [En línea]. 2017. vol. 6, no. 2. pp. 1-16. [Fecha de consulta: 15 octubre 2017]. ISSN: 2227-2690. Disponible en: http://revistas.unica.cu/index.php/uciencia/article/view/547

**BERISEÑO, Jose.** Transmisión de Datos [En línea]. 3.ª ed. Mérida: Universidad de los Andes, 2005 [Fecha de consulta: 14 diciembre 2017]. ISBN: 9802215392. Disponible en: https://www.elsolucionario.org/transmision-de-datos-jose-e-briceno-marquez-3ed/.

**CAMBRIDGE Dictionary.** Traducción de "proxy" - Diccionario Inglés-Español [En línea]. Cambridge University Press, 2017 [Fecha de consulta: 28 enero 2018]. Disponible en: https://dictionary.cambridge.org/es/diccionario/ingles-espanol/proxy.

**CAMERON, Jaime.** Webmin Documentation [En línea]. WebminGroup, 2017 [Fecha de consulta: 16 enero 2018]. Disponible en: https://doxfer.webmin.com/Webmin/Main_Page.

**CÁRDENAS, Lain.** Aplicación de Patrones de Diseño para Garantizar Alta Flexibilidad en el Software. Universidad César Vallejo [En línea]. 2016. vol. 7, no. 1. pp. 77-82. [Fecha de consulta: 15 marzo 2018]. ISSN: 1819-4575. Disponible en: http://revistas.ucv.edu.pe/index.php/RTD/article/view/696

**CARRAZANA, Deilis.** Programa de mejora: Procedimiento para pruebas de seguridad [En línea]. La Habana: Universidad de las Ciencias Informáticas, 2017. [Fecha de consulta: 26 de marzo de 2018]. Disponible en: http://mejoras.prod.uci.cu/proceso_desarrollo_produccion/guidances/whitepapers/resources/Procedimiento%20de%20Pruebas%20de%20Seguridad.pdf.

**CASTILLO, Reidel y SORIA, Pablo.** Herramienta para la administración y configuración de servidores (HMAS). Tesis (Ingeniero en Ciencias Informáticas). La Habana: Universidad de las Ciencias Informáticas, 2012.

**COMITÉ CENTRAL DEL PCC.** Lineamientos de la política económica y social del Partido y la Revolución. [En línea]. (13 de septiembre de 2016). [Fecha de consulta: 15 de octubre de 2017]. Disponible en : http://www.cubadebate.cu/especiales/2016/09/13/vea-el-texto-integro-de-la-actualizacion-de-los-lineamientos-para-el-periodo-2016-2021-pdf/.

**CRAIG, Larman.** UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2.ª ed. Madrid: Pearson Educación, 2003. pp. 590. ISBN: 84-205-3438-2.

**DÍAZ-CANEL, Miguel.** Existe la voluntad de poner la Informatización y la Internet al servicio de todos. Clausura: I Taller Nacional de Informatización y Ciberseguridad. La Habana, 2015.

**ETECSA.** Internet y conectividad [En línea]. La Habana: ETECSA, 2017 [Fecha de consulta: 19 septiembre 2017]. Disponible en: http://www.etecsa.cu/internet_conectividad/.

**FONDO, Pablo.** Proxy HTTP para visualización de tráfico web en tránsito. Tesis (Ingeniería en Sistemas). Vigo: Universidad de Vigo, 2016.

**GÓMEZ, Ezequiel, SARTORIO, Alejandro y VAQUERO, Marcelo.** EasyCard.js Framework. Tesis (Ingeniería en Sistemas Informáticos). Santa Fe: Universidad Abierta Interamericana, 2016.

**GRADOS, Julio.** *JavaScript convive con HTML* [Artículo de un blog]. (2016). [Fecha de consulta: 8 de noviembre de 2017]. Recuperado de: https://devcode.la/blog/que-es-javascript/.

**HAMON, Hugo.** Applying Design Patterns to Symfony [En línea]. Estambul, Turquía: SensioLabs, 2014 [Fecha de consulta: 12 septiembre 2017]. Disponible en: https://speakerdeck.com/hhamon/applying-design-patterns-to-symfony.

**HANIFI, Nadin.** Diseño de Terminal para Conexión desde el Laboratorio de Ingeniería Eléctrica a UPVNET basado en Raspebrry PI. Tesis (Ingeniería en Tecnologías Industriales). Valencia: Universidad Politécnica de Valencia, 2016.

**HIDALGO, Chistrian.** Configuración de Servicios de Red y Portal Web para la Preinscripción a Cursos. Tesis (Licenciatura en Análisis de Sistemas). El Oro: Unidad Académica de Ingeniería Civil, 2015.

**INTERNET LIVE STATS.** Worldometers [En línea]. W3C, 2018 [Fecha de consulta: 2 mayo 2018]. Disponible en : http://www.internetlivestats.com/.

**ISO/IEC/IEEE INTERNATIONAL STANDARD** (EEUU). ISO/IEC/IEEE 24765, of. 2017: Systems and software engineering-Vocabulary. New York, agosto 28 de 2017.

**JARA, Andrés, PINTADO, Juan y SERRANO, Marcelo.** Configuración de un Servidor Proxy utilizando Squid. Trabajo de Curso. Cuenca: Universidad del Azuay, 2016.

**JETBRAIN.** Enjoy Productive PHP [En línea]. JetBrains s.r.o, 2017 [Fecha de consulta: 6 febrero 2018]. Disponible en: https://www.jetbrains.com/phpstorm/.

**KASPERKYLAB.** ¿Qué es un filtro web? | Definición de filtro web [En línea]. AO Kaspersky Lab, 2017 [Fecha de consulta: 5 octubre 2017]. Disponible en: https://latam.kaspersky.com/resource-center/definitions/web-filter.

**MARTÍNEZ, Erick.** Lenguajes de Programación del lado servidor [En línea]. 2014 [Fecha de consulta: 7 noviembre 2017]. Disponible en: http://michelletorres.mx/lenguajes-de-programacion-del-lado-servidor/.

**MARTÍNEZ, Alejandro y MARTÍNEZ, Raúl.** Guía a Rational Unified Process [En línea]. Albacete: Escuela Politécnica Superior de Albacete. Universidad de Castilla la Mancha, 2014 [Fecha de consulta: 12 febrero 2018]. Disponible en: https://www.researchgate.net/publication/268005509_Guia_a_Rational_Unified_Process

**MUNITICH, Martín.** Internet, fuente general de información: Uso y soporte como referencia bibliográfica de trabajos prácticos. Reflexión Académica en Diseño & Comunicación. 2013. vol. 14, no.21, 2013. pp. 68-70. ISSN: 1668-1673.

**ORACLE.** Getting Started with MySQL [En línea]. Oracle Corporation, 2017 [Fecha de consulta: 22 enero 2017]. Disponible en: https://dev.mysql.com/doc/mysql-getting-started/en/.

**ORDOÑEZ, Hugo.** Business Processes as a Strategy to Improve Requirements Elicitation in Extreme Programming. Colombia, Popayán: Memorias del VII Congreso Iberoamericano de Telemática CITA, 2015.

**PACHECO, Nacho.** Bases de datos y Doctrine [En línea]. 2013 [Fecha de consulta: 14 noviembre 2017]. Disponible en: http://gitnacho.github.io/symfony-docs-es/book/doctrine.html.

**PERL.** About Perl. What is Perl? Features and History [En línea]. Perl, 2017 [Fecha de consulta: 11 enero 2018]. Disponible en https://www.perl.org/about.html.

**PHP.** ¿Qué es PHP? [En línea]. PHP Fundation, 2017 [Fecha de consulta: 12 enero 2018]. Disponible en: http://php.net/manual/es/intro-whatis.php.

**PHP-FIG.** PHP Standards Recommendation [En línea]. PHP Framework Interop Group, 2018 [Fecha de consulta: 14 abril 2018]. Disponible en: https://www.php-fig.org/psr/

**POSTGRESQL.** What is PostgreSQL? [En linea]. PostgreSQL Global Development Group, 2017 [Fecha de consulta: 25 enero 2018]. Disponible en: https://www.postgresql.org/docs/9.4/static/intro-whatis.html.

**PRESSMAN, Roger.** Ingeniería del Software. Un enfoque práctico. 7ª ed. México, D.F: The MacGraw-Hill Companies Inc., 2010. ISBN: 978-607-15-0314-5.

**QUALITY.** Pruebas de Seguridad [En línea]. Colombia: V&V Quality, 2016 [Fecha de consulta: 12 abril 2018]. Disponible en: http://vyvquality.com/pruebas-seguridad/

**RAMOS, Daniel.** Curso de Ingeniería de Software. 2.ª ed. IT Campus Academy, 2015. pp. 340. ISBN: 978-1544132532

**RAMOS, Daniel.** Desarrollo de Software: Estimación Requisitos y Análisis. 2.ª ed. IT Campus Academy, 2016. pp. 125. ISBN: 9781530088614.

**REGUANT, Mercedes y TORRADO, Mercedes.** El método Delphi. Revista d'Innovació i Recerca en Educació. 2016. vol. 9, no. 1. pp. 87-102. ISSN:2013-2255.

**RODRÍGUEZ, Alina y SILVA, Luis.** Arquitectura de software para el sistema de visualización médica Vismedic. Revista Cubana de Informática Médica. 2016. vol. 8, no. 1. pp. 75-86. ISSN: 1684-1859.

**SALDAÑA, Gabriel.** Nethazard [Artículo de un blog]. 2017. [Fecha de consulta: 30 de abril de 2017]. Recuperado de: http://blog.nethazard.net.

**HERNANDEZ, Roberto, FERNANDEZ, Carlos y BAPTISTA, María.** *Metodología de la investigación* [En línea]. 2010. ISBN 9786071502919. Disponible en: http://www.casadellibro.com/libro-metodologia-de-la-investigacion-5-ed-incluye-cd-rom/9786071502919/1960006

**SAMUEZA, Fernanda.** *Que es Zentyal* [En línea]. Quito: Prezi Inc, 2015 [Fecha de consulta: 11 enero 2018]. Disponible en: https://prezi.com/oxocahx2zcj_/que-es-el-zentyal/.

**SÁNCHEZ, Tamara.** Metodología de desarrollo para la Actividad productiva de la UCI v1.2. La Habana: Universidad de las Ciencias Informáticas, 6 de marzo de 2015.

**SARMIENTO, Julio.** UML: Diagrama de despliegue. Obtenido de Visión general del diagrama de despliegue [Artículo de blog]. 2016. Disponible en: http://umldiagramadespliegue.blogspot.com/.

**SEI.** Mejora de los procesos para el desarrollo de mejores productos y servicios. Software Engineering Institute. Technical Report, 2010.

**SOMMERVILLE, Ian.** Ingeniería del software. 7ª ed. Madrid: Pearson Educación S.A, 2011. ISBN: 84-7829-074-5.

**SQUID CACHE.** What is Squid? [En línea]. Squid Software Fundation, 2015 [Fecha de consulta: 12 septiembre 2018]. Disponible en: http://www.squid-cache.org/Intro/.

**SUSE.** Portal: YAST [En línea]. SUSE LLC, 2017 [Fecha de consulta: 26 de enero 2018]. Disponible en: https://es.opensuse.org/Portal:YaST.

**SYMFONY.** What is Symfony? Symfony para programadores [En línea]. Symfony SAS, 2017 [Fecha de consulta: 8 noviembre 2017]. Disponible en: http://symfony.com/what-is-symfony.

**TECHOPEDIA.** Modeling Lenguage [En línea]. Techopedia INC, 2017 [Fecha de consulta: 12 abril 2018]. Disponible en: https://www.techopedia.com/definition/20810/modeling-language.

**TROYA, Javier.** Creando conexiones SSH con PHP [Articulo de blog]. (20 de noviembre de 2015). [Fecha de consulta: 14 de noviembre de 2017]. Recuperado de: https://rootstack.com/es/blog/creando-conexiones-ssh-con-php.

**UNAD.** Lenguaje Unificado de Modelado UML. Diagrama de Clases de Diseño [En línea]. Universidad Nacional Abierta y a Distancia, 2016 [Fecha de consulta: 1 de marzo de 2018]. Disponible en: http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html.

**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.** Manual de usuario Smart Keeper. La Habana: Universidad de las Ciencias Informáticas, 2014.

**WIGGINTON, Jim.** phpseclib - PHP Secure Communications Library README [En línea]. GitHub Social Coding, 2017 [Fecha de consulta: 17 de noviembre de 2017]. Disponible en: https://github.com/phpseclib/phpseclib.

**ZENTYAL Community.** Las pymes y las TICs. Zentyal: servidor Linux para pymes [En línea]. Zentyal Comnunity, 2014 [Fecha de consulta: 5 de noviembre de 2017]. Disponible en: https://wiki.zentyal.org/wiki/Es/3.5/Presentacion.

**ZORRILLA, Marta.** Modelo de datos [En línea]. Cantabria: Universidad de Cantabria, 2011 [Fecha de consulta: 24 marzo 2018]. Disponible en: https://es.scribd.com/document/73596837/02-Modelos-de-Datos-ER-UML-Relacional.

# GLOSARIO DE TÉRMINOS

**FTP:** Protocolo de transferencia de archivos, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red *TCP*, basado en la arquitectura cliente-servidor.

**GNU:** Es un acrónimo recursivo que significa "*GNU* no es *Unix*" (*GNU is Not Unix*). Este proyecto fue iniciado por *Richard Stallman*, y anunciado el 27 de septiembre de 1983, con el objetivo de crear un sistema operativo completamente libre.

**GOPHER:** Es un servicio de Internet consistente en el acceso a la información a través de menús. La información se organiza en forma de árbol: sólo los *nodos* contienen menús de acceso a otros menús o a *hojas*, mientras que las hojas contienen simplemente información textual. En cierto modo es considerado un predecesor de la web, aunque sólo se permiten enlaces des de nodos-menús hasta otros nodos-menús o a hojas, y las hojas no tienen ningún tipo de hiperenlaces.

**HTTP:** Son las siglas de **"***Hypertext Transfer Protocol*" es un protocolo de transferencia donde se utiliza un sistema mediante el cual se permite la transferencia de información entre diferentes servicios y los clientes que utilizan páginas web.

**HTTPS:** Es un protocolo de aplicación basado en el protocolo *HTTP*, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de *HTTP*.

**SCP:** Son las siglas de *Simple Communication Protocol* es un medio de transferencia segura de archivos informáticos entre un host local y otro remoto o entre dos hosts remotos, usando el protocolo *Secure Shell* (SSH).

**SSH:** *Secure Shell* en español Intérprete de Órdenes Seguro protocolo que sirve para acceder servidores privados a través de una puerta trasera. Permite manejar por completo el servidor mediante un intérprete de comandos, y también puede redirigir el tráfico de un servidor.

**TCP:** Protocolo de control de transmisión, permite a dos anfitriones establecer una conexión e intercambiar datos. El *TCP* garantiza la entrega de datos y que estos no se pierdan durante la transmisión, también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados.

**WAIS:** Es un sistema de búsqueda de texto distribuido para buscar bases de datos indexadas en computadoras remotas.

# ANEXOS

## Anexo 1. Historias de Usuario

Tabla 21. HU_1 Insertar *proxy Squid*

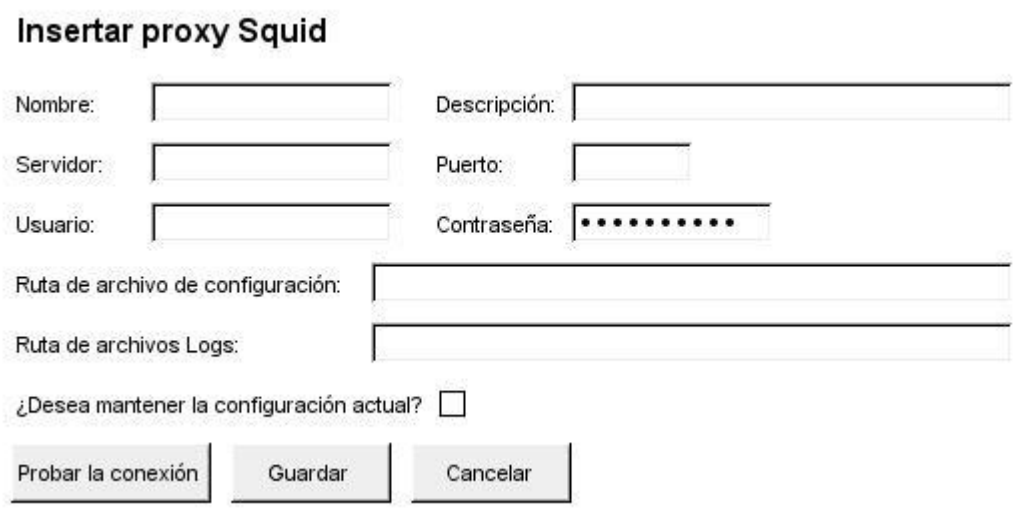
| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_1 | **Nombre:** Insertar *proxy Squid* | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Alta | | |
| **Tiempo estimado:** 2 horas | **Tiempo real:** 2 horas | |
| **Descripción:** El sistema debe permitir insertar un *proxy Squid* | | |
| **Observaciones:** Al insertar un proxy Squid, este debe tener un nombre, una descripción, la dirección IP o el nombre del servidor, puerto por el que se realizarán las conexiones *SSH*, usuario *root* del servidor, así como su contraseña, las rutas del fichero de configuración y de los archivos *logs* del servidor *proxy*. El administrador podrá decidir si desea mantener las configuraciones actuales del *proxy Squid* o establecer algunos de los perfiles de configuración que establece el sistema Xilema Smart Keeper. | | |
| **Prototipo de interfaz:**   | | |

Tabla 22. HU_2 Mostrar proxy Squid

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_2 | **Nombre:** Mostrar *proxy Squid* | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Media | | |
| **Tiempo estimado:** 2 horas | **Tiempo real:** 2 horas | |
| **Descripción:** El sistema debe permitir mostrar detalles de los *proxy Squid* | | |
| **Observaciones:** Sin observaciones | | |
| **Prototipo de interfaz:** No aplica | | |

Tabla 23. HU_3 Modificar proxy Squid

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_3 | **Nombre:** Modificar *proxy Squid* | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Media | | |
| **Tiempo estimado:** 2 horas | **Tiempo real:** 2 horas | |
| **Descripción:** El sistema debe permitir modificar los *proxy Squid* | | |
| **Observaciones:** Para modificar un proxy Squid, este debe tener un nombre, una descripción, la dirección IP o el nombre del servidor, puerto por el que se realizarán las conexiones *SSH*, usuario *root* del servidor, así como su contraseña, las rutas del fichero de configuración y de los archivos *logs* del servidor *proxy*. | | |
| **Prototipo de interfaz:** No aplica | | |

Tabla 24. HU_5 Eliminar proxy Squid

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_5 | **Nombre:** Eliminar *proxy Squid* | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Media | | |

| | |
|---|---|
| **Tiempo estimado:** 2 horas | **Tiempo real:** 2 horas |
| **Descripción:** El sistema debe permitir eliminar los *proxy Squid* | |
| **Observaciones:** Sin Observaciones | |
| **Prototipo de interfaz:** No aplica | |

Tabla 25. HU_6 Comprobar conexión del servidor proxy Squid

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_6 | **Nombre:** Comprobar conexión del servidor *proxy Squid* | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Baja | | |
| **Tiempo estimado:** 1 hora | **Tiempo real:** 1 hora | |
| **Descripción:** El sistema debe permitir comprobar la conexión al servidor *proxy Squid* que se está insertando | | |
| **Observaciones:** Si los parámetros de conexión son introducidos correctamente el sistema mostrara un mensaje notificando al usuario “La conexión fue exitosa” en caso contrario mostrara una error notificando la causa del fallo de conexión. | | |
| **Prototipo de interfaz:** No aplica | | |

Tabla 26. HU_8 Buscar proxy Squid

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_8 | **Nombre:** Buscar *proxy Squid* | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Baja | | |
| **Tiempo estimado:** 2 hora | **Tiempo real:** 2 hora | |
| **Descripción:** El sistema debe permitir realizar una búsqueda simple de los servidores *proxy Squid* | | |
| **Observaciones:** El sistema permite filtrar la búsqueda por el nombre del *proxy Squid* o por el estado actual de los servicios | | |
| **Prototipo de interfaz:** | | |


Tabla 27. HU_7 Consultar estado del servicio del proxy Squid

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_7 | **Nombre:** Consultar estado del servicio del *proxy Squid* |
| **Programador responsable:** Esteban R. de León Naranjo | **Iteración asignada:** 1 |
| **Prioridad:** Baja | |
| **Tiempo estimado:** 2 horas | **Tiempo real:** 2 horas |
| **Descripción:** El sistema debe permitir mostrar la información del estado de los *proxy Squid* | |
| **Observaciones:** Los estados del *proxy Squid* son:<br>• **Sin Conexión:** No se ha podido establecer una conexión con el *proxy Squid*.<br>• **Iniciado:** El servicio del *proxy Squid* esta iniciado.<br>• **Detenido:** El servicio del *proxy Squid* está detenido.<br>• **En Espera:** El servicio del *proxy Squid* está realizando un cambio de estado en el momento de su consulta. | |
| **Prototipo de interfaz:** No aplica | |

Tabla 28. HU_9 Configurar servicio del *Dansguardian*

| **Historia de Usuario** | |
|---|---|
| **Número:** HU_9 | **Nombre:** Configurar servicio del *Dansguardian* |
| **Programador responsable:** Esteban R. de León Naranjo | **Iteración asignada:** 1 |
| **Prioridad:** Alta | |
| **Tiempo estimado:** 4 horas | **Tiempo real:** 4 horas |

| |
|---|
| **Descripción:** Los administradores pueden configurar a través de un formulario los servicios del componente de filtrado *Dansguardian* |
| **Observaciones:** Un componente de filtrado *Dansguardian* puede ser configurado si el estado de los servicios asociadas a este es distinto a Sin Conexión. Si los datos son introducidos correctamente el sistema mostrará un mensaje notificando que “El Dansguardian ha sido configurado con éxito” y automáticamente será reiniciado el servicio *Dansguardian* para aplicar los cambios. Si el usuario introduce los datos incorrectamente el sistema mostrará un mensaje de error y señalará en color rojo aquellos campos que su validación fue incorrecta. |
| **Prototipo de interfaz:** |

Tabla 29. HU_10 Iniciar servicio del *proxy Squid*

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_10 | **Nombre:** Iniciar servicio del *proxy Squid* | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Alta | | |
| **Tiempo estimado:** 1 hora | **Tiempo real:** 1 hora | |
| **Descripción:** El sistema debe permitir iniciar los servicios del servidor *proxy Squid* | | |
| **Observaciones:** Un servicio puede ser iniciado si su estado es distinto a Sin conexión o no estar iniciado. Si el servicio es iniciado correctamente el módulo enviará una notificación al usuario “El servicio <nombre de servicio> se ha iniciado correctamente” y cambiará su estado actual a Iniciado. Si no se puede iniciar el servicio mostrará un mensaje notificando al usuario porque no pudo ser iniciado, y mantendrá su estado actual. | | |
| **Prototipo de interfaz:** No aplica. | | |

Tabla 30. HU_11 Detener servicio del *proxy Squid*

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_11 | **Nombre:** Detener servicio del proxy Squid | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Alta | | |
| **Tiempo estimado:** 1 hora | **Tiempo real:** 1 hora | |
| **Descripción:** El sistema debe permitir detener los servicios del servidor *proxy Squid* | | |

| |
|---|
| **Observaciones:** Un servicio puede ser detenido si su estado actual es distinto a Sin conexión o no estar detenido. Si el servicio es detenido correctamente el módulo enviará una notificación al usuario “El servicio <nombre de servicio> se ha detenido correctamente” y cambiará su estado actual a Detenido. Si no se puede detener el servicio mostrará un mensaje notificando al usuario porque no pudo ser detenido, y mantendrá su estado actual. |
| **Prototipo de interfaz:** No aplica. |

Tabla 31. HU_12 Reiniciar servicio del *proxy Squid*

| **Historia de Usuario** | | |
|---|---|---|
| **Número:** HU_12 | **Nombre:** Reiniciar servicio del proxy Squid | |
| **Programador responsable:** Esteban R. de León Naranjo | | **Iteración asignada:** 1 |
| **Prioridad:** Alta | | |
| **Tiempo estimado:** 1 hora | **Tiempo real:** 1 hora | |
| **Descripción:** El sistema debe permitir reiniciar los servicios del servidor *proxy Squid* | | |
| **Observaciones:** Un servicio puede ser reiniciado si su estado actual es distinto a Sin conexión. Si el servicio es reiniciado correctamente el módulo enviará una notificación al usuario “El servicio <nombre de servicio> se ha reiniciado correctamente” y mantendrá su estado actual. Si no se puede reiniciar el servicio mostrará un mensaje notificando al usuario porque no pudo ser reiniciado, y mantendrá su estado actual. | | |
| **Prototipo de Interfaz:** No aplica. | | |

# Anexo 2. Diagramas de clases de diseño

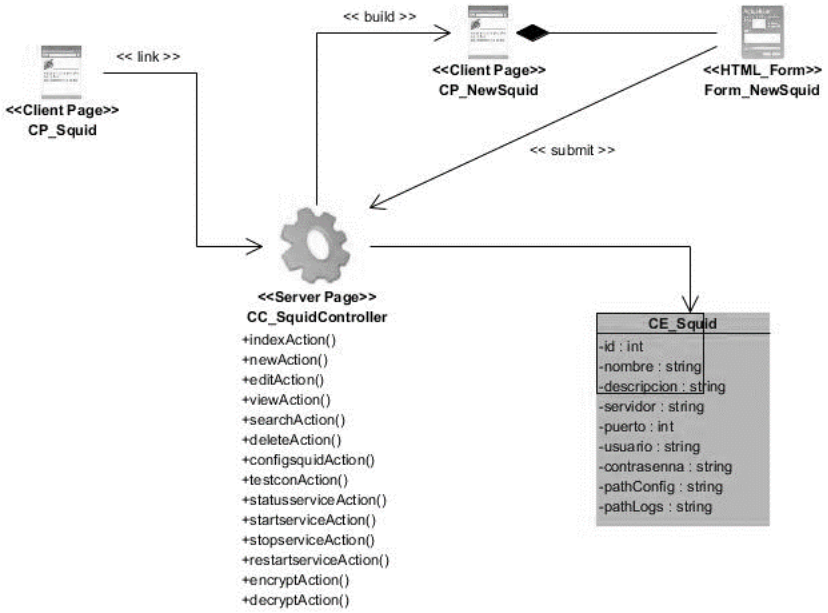



Figura 12. Diagrama de clases del diseño HU Insertar *proxy Squid*

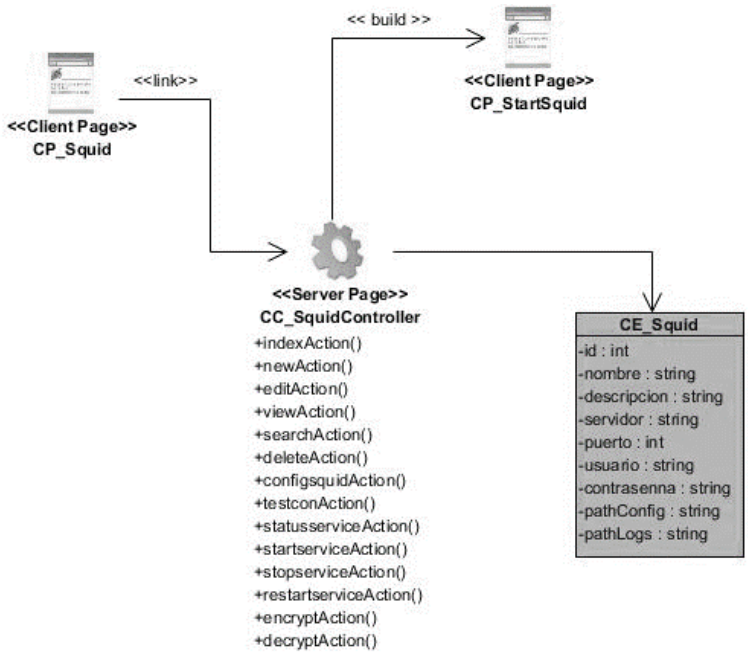



Figura 13. Diagrama de clases del diseño HU Iniciar servicio del *proxy Squid*

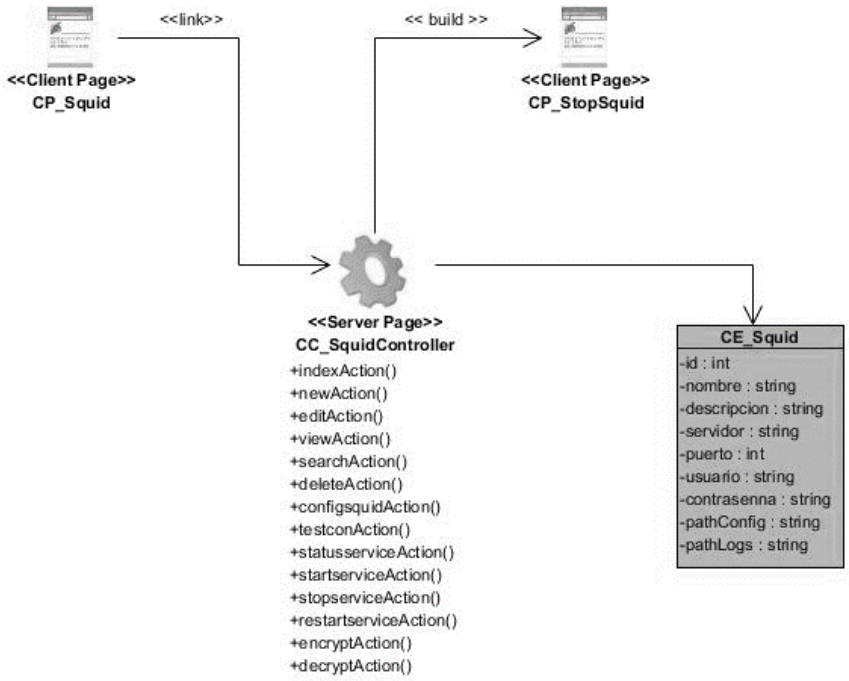



Figura 14. Diagrama de clases del diseño HU Detener servicio del proxy Squid

# Anexo 3. Diagramas de secuencia

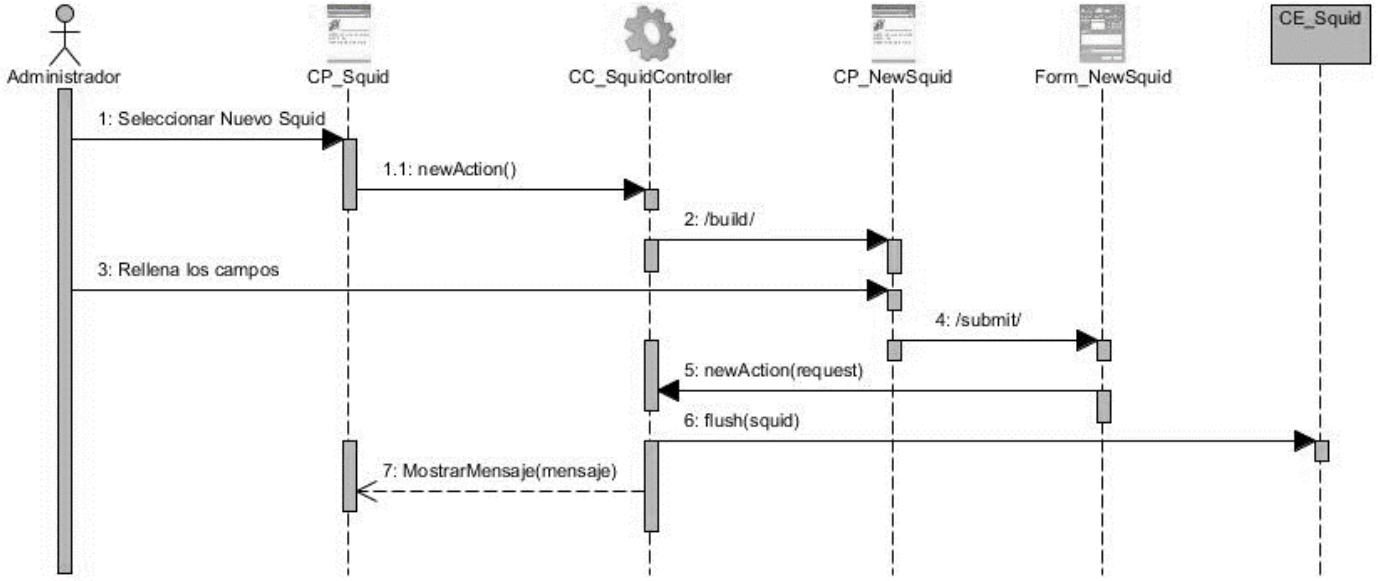



Figura 15. Diagrama de secuencia HU Insertar *proxy Squid*

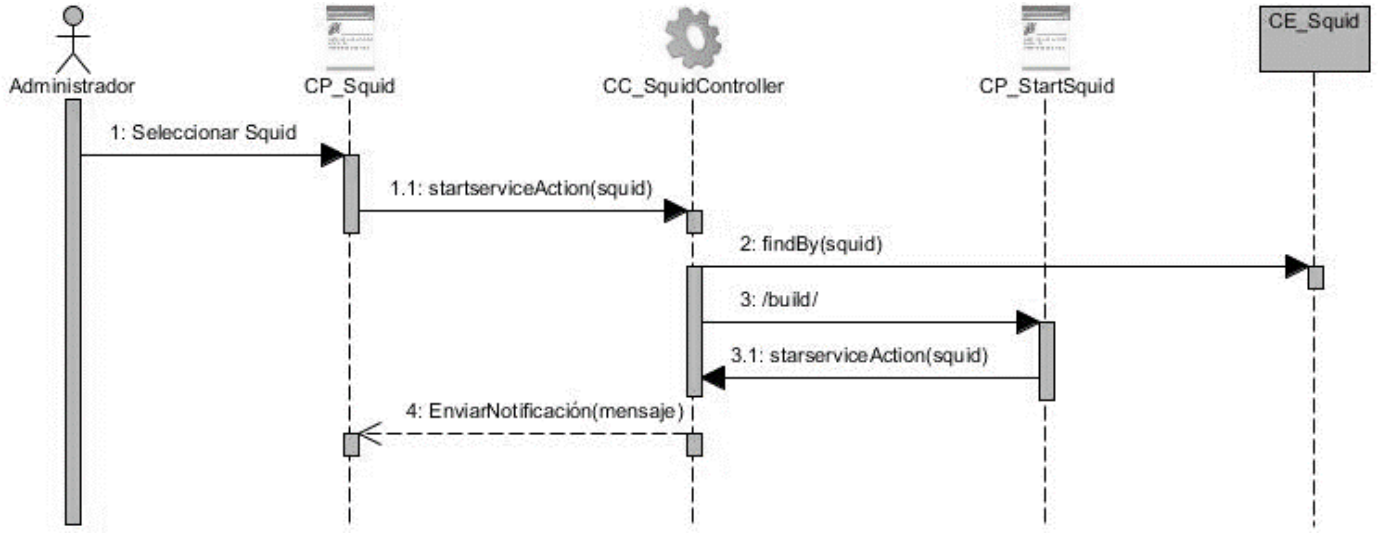



Figura 16. Diagrama de secuencia HU Iniciar servicio del *proxy Squid*

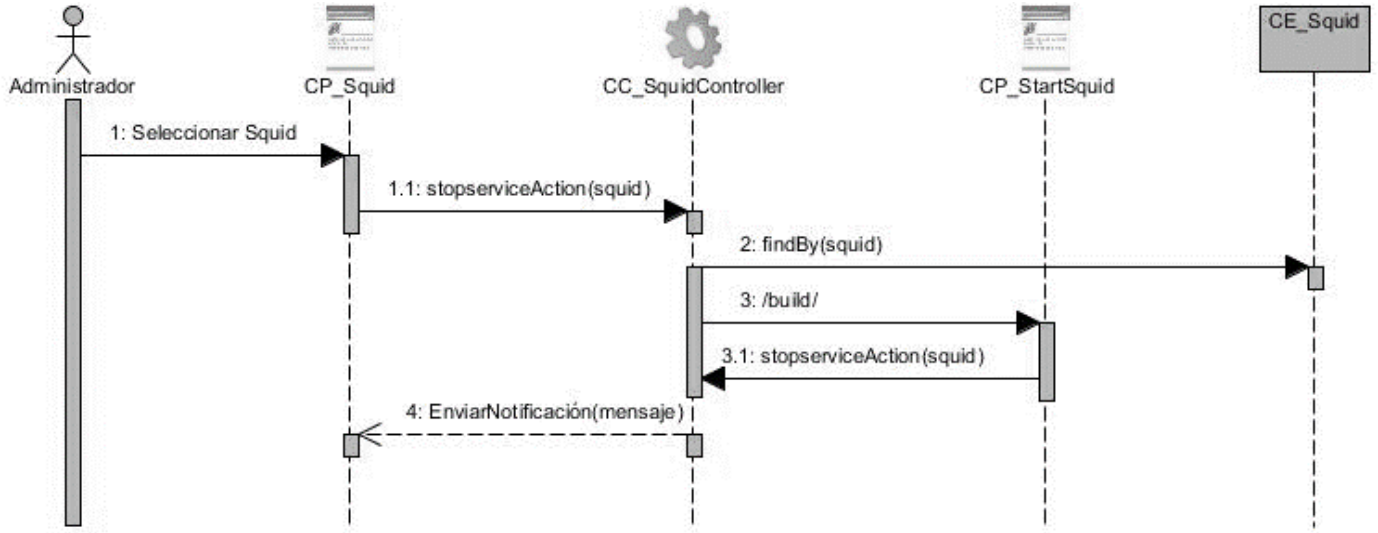



Figura 17. Figura 16. Diagrama de secuencia HU Detener servicio del *proxy Squid*

## Anexo 4. Casos de Prueba

Tabla 32. Caso de Prueba del RF10_Iniciar servicio del proxy Squid

| **Caso de Prueba 3:** SC RF10_Iniciar servicio del *proxy Squid* | | | |
|---|---|---|---|
| **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema y debe existir al menos un *proxy Squid* disponible (estado distinto a "Sin Conexión"). | | | |
| **Escenario** | **Descripción** | **Respuesta del Sistema** | **Flujo Central** |
| EC 3.1 Iniciar un *proxy* correctamente | El sistema permite iniciar un *proxy* que no esté iniciado de forma correcta | El sistema muestra una notificación al usuario "El *proxy* <nombre> se ha iniciado correctamente" | En el menú superior del sistema:<br>1. El usuario accede a la opción Configuración.<br>2. El usuario selecciona la opción *Proxy Squid y* el módulo muestra una lista de los servidores *proxy Squid* registrados.<br>3. El usuario selecciona la opción Iniciar *proxy* (columna Servicio 1er botón) de un *proxy Squid* disponible. |
| EC 3.2 Iniciar un *proxy* que ya está iniciado | El sistema no permite iniciar un *proxy* que ya este iniciado | El sistema deshabilita el botón que permite iniciar el *proxy Squid* | |

Tabla 33. Caso de Prueba del RF11_Detener servicio del proxy Squid

| **Caso de Prueba 4:** SC RF11_Detener servicio del *proxy Squid* | | | |
|---|---|---|---|
| **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema y debe existir al menos un *proxy Squid* disponible (estado distinto a "Sin Conexión"). | | | |
| **Escenario** | **Descripción** | **Respuesta del Sistema** | **Flujo Central** |
| EC 3.1 Detener un *proxy* correctamente | El sistema permite iniciar un *proxy* que no esté detenido de forma correcta | El sistema muestra una notificación al usuario "El *proxy* <nombre> se ha detenido correctamente" | En el menú superior del sistema:<br>1. El usuario accede a la opción Configuración.<br>2. El usuario selecciona la opción *Proxy Squid y* el módulo muestra una lista de los servidores *proxy Squid* registrados.<br>3. El usuario selecciona la opción Detener *proxy* (columna Servicio 2do |
| EC 3.2 Detener un *proxy* que ya está detenido | El sistema no permite detener un *proxy* que ya este detenido | El sistema deshabilita el botón que permite detener el *proxy Squid* | |

| | | | |
|---|---|---|---|
| | | | botón) de un *proxy Squid* disponible. |

Tabla 34. Caso de Prueba del RF12_Reiniciar servicio del proxy Squid

| **Caso de Prueba 5:** SC RF12_Reiniciar servicio del *proxy Squid* | | | |
|---|---|---|---|
| **Condiciones de ejecución:** El usuario debe estar autenticado en el sistema y debe existir al menos un *proxy Squid* disponible (estado distinto a "Sin Conexión"). | | | |
| **Escenario** | **Descripción** | **Respuesta del Sistema** | **Flujo Central** |
| EC 3.1 Reiniciar un *proxy* correctamente | El sistema permite reiniciar un *proxy* forma correcta | El sistema muestra una notificación al usuario "El *proxy* <nombre> se ha reiniciado correctamente" | En el menú superior del sistema:<br>1. El usuario accede a la opción Configuración.<br>2. El usuario selecciona la opción *Proxy Squid y* el módulo muestra una lista de los servidores *proxy Squid* registrados.<br>3. El usuario selecciona la opción Reiniciar *proxy* (columna Servicio 3do botón) de un *proxy Squid* disponible. |

## Anexo 5. Encuesta para la identificación de posibles expertos

Este material constituye un instrumento para la investigación del trabajo de diploma en opción al título de Ingeniero en Ciencias Informáticas: **Módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper** del autor: **Esteban R. de León Naranjo**, para el Centro de Ideoinformática (CIDI) de la Facultad 1. Usted ha sido identificado como posible experto en el tema y su colaboración es importante para la investigación.

**1.-** Nombre y Apellidos: _____

**2.-** Cargo que ocupa: _____

**3.-** Años de experiencia: ___

**4.-** Marque con una cruz (X):

Nivel Escolar: ___ Técnico Medio ___ Enseñanza Media Superior ___ Enseñanza Superior

Grado Científico: ___ Master en Ciencias ___ Doctor en Ciencias ___ Ninguno

**5-.** Valore en la siguiente tabla con una escala del 1 al 10, el grado de conocimiento que usted posee sobre cada uno de los temas. Considere que la escala es ascendente donde el 1 es que no posee ningún conocimiento y 10 posee el máximo conocimiento del tema:

| Tema | Escala de conocimiento |
|---|---|
| Administración de Redes de Datos | |
| Configuración y Administración de servidores *proxy Squid* | |
| Filtrado de contenido web | |
| Xilema Smart Keeper | |

**6.-** Realice una autoevaluación del grado de influencia que cada una de las fuentes, que se le presenta a continuación, ha tenido en su conocimiento y criterio para la administración y configuración de servidores *proxy Squid*. Para ello marque con una cruz (X) según corresponda:

| Fuentes de Argumentación | Alto | Medio | Bajo |
|---|---|---|---|
| Análisis teóricos realizados por usted | | | |
| Experiencia obtenida | | | |
| Trabajos de autores nacionales | | | |
| Trabajos de autores extranjeros | | | |

| | | | |
|---|---|---|---|
| Su conocimiento del estado del problema en el extranjero | | | |
| Su Intuición | | | |

Gracias por su colaboración.

## Anexo 6. Encuesta a expertos

Co.:

Usted fue seleccionado como miembro del panel de experto para validar la **hipótesis científica:** El desarrollo de un módulo para la gestión los servidores *proxy Squid* del sistema Xilema Smart Keeper, facilitará el trabajo a los administradores del sistema. En aplicación del modelo matemático Torgerson del método Delphi, se le solicita que responda el siguiente cuestionario según su nivel de valoración.

Respecto al Módulo para la gestión de servidores *proxy Squid* del sistema Xilema Smart Keeper emita una valoración según las siguientes categorías de evaluación: **MA** si considera que la sentencia es Muy Adecuada, **A** si la considera adecuada, **PA** si la valora Poco Adecuada e **I** si es Inadecuada. Maque con una cruz (X):

1. Facilidad de administración y configuración de servidores *proxy Squid*.
2. Administración y configuración remota de los servicios del servidor *proxy Squid*.
3. Facilidad en la gestión de las reglas de filtrado.
4. Rapidez en la respuesta de peticiones.
5. Presentación de una interfaz agradable e intuitiva para el usuario.

| Nº | Sentencia | Categorías Evaluativas | | | |
|---|---|---|---|---|---|
| | | MA | A | PA | I |
| 1 | Facilidad de administración y configuración de servidores *proxy Squid.* | | | | |
| 2 | Administración y configuración remota de los servicios del servidor *proxy Squid.* | | | | |
| 3 | Facilidad en la gestión de las reglas de filtrado. | | | | |
| 4 | Rapidez en la respuesta de peticiones. | | | | |
| 5 | Presentación de una interfaz agradable e intuitiva para el usuario. | | | | |

Gracias por su colaboración.

Atentamente: Esteban R. de León Naranjo