UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 1

“Módulo para la administración remota del servicio telemático Proxy en la plataforma Nova-ARST ”

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

Autor : Jordani Yamil Verdecia Lago

Tutor(es) : Msc. Arianna Rodríguez Jiménez
Ing. Magdiel Fernández Santana

La Habana, Diciembre de 2022

<Año 64 del Triunfo de la Revolución>

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo **Jordani Yamil Verdecia Lago**, con carné de identidad **97101306625,** soy el autor principal del trabajo titulado **"Módulo para la administración remota del servicio telemático Proxy en la plataforma Nova-ARST"** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

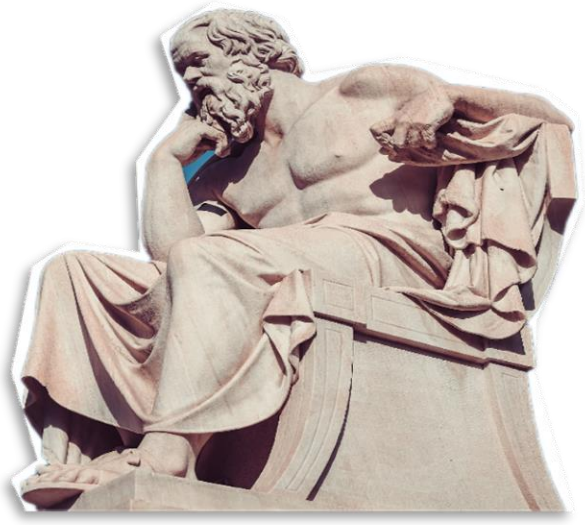
Para que así conste firman la presente a los _____ días del mes _____ del año_____.

_____

Jordani Yamil Verdecia Lago

_____ _____

Ing. Magdiel Fernández Santana **Msc. Arianna Rodríguez Jiménez**



“Solo existe un bien: el conocimiento. Solo hay un mal: la ignorancia”.

Sócrates(469 AC – 399 AC) Filósofo Griego

AGRADECIMIENTOS

Primeramente, agradezco a Dios por su inmenso amor y bondad, por otorgarme una familia maravillosa, por hacerme la persona que soy actualmente y por darme un poco más de sabiduría para enfrentar este trabajo que se tornó algo complejo para mí. Agradezco cada momento cursado en la universidad, pues cada revés sufrido, no fue más que una simple prueba para crecer como persona. Gracias al profesor Jorge Sergio Menéndez, quien no solo me enseñó sobre las particularidades de su asignatura, sino que me educó como un hijo y me inculcó verdaderos valores de un digno profesional. Gracias le doy a Dios por conocer una persona tan excelente, como lo es mi tutor Magdiel Fernández Santana, quien a pesar de tener que lidiar con una persona tan irresponsable como yo, me fue ayudando de a poco y dándome consejos que son de oro para superarme y crecer cada día como Ingeniero en Ciencias Informáticas.

Finalmente agradezco a quien lee este apartado. El desarrollo de esta tesis no lo puedo catalogar como algo sencillo, pero espero que todo el esfuerzo puesto en la investigación para completarla pueda ser de ayuda para alguien más.

DEDICATORIA

Le dedico toda constante solución y fruto de mi esfuerzo a mi familia, especialmente a mi madre que siempre estuvo para apoyarme en cada momento; también he de dedicar estas líneas a mi padre, mis hermanos y mi linda novia, que siempre estuvieron para exigirme un poquito más de esfuerzo para culminar esta investigación. A ellos y a aquellos que me han alentado a ser una mejor persona cada día: MUCHÍSIMAS GRACIAS.

RESUMEN

El presente trabajo se centra en el objetivo de desarrollar funcionalidades que permitan la administración del servicio Proxy desde la Herramienta de administración remota de servicios telemáticos en entornos de código abierto. El módulo mencionado surge para apoyar el procedimiento de instalación y configuración de un servidor proxy en las entidades cubanas. Se emplean los métodos teóricos Analítico-Sintético e Inductivo-Deductivo. Se documentaron las diferentes tecnologías utilizadas y la metodología empleada de desarrollo de software Proceso Unificado Ágil en su variante para la Universidad de las Ciencias Informáticas (AUP-UCI), como guía del proceso. Finalmente se obtiene como resultado un módulo que permite la administración de Squid como servidor proxy, E2guardian como servidor de filtrado de contenido y Squish como herramienta para la gestión de cuotas, obteniendo de esta forma la administración del servicio Proxy en diferentes computadoras servidoras de manera remota.

Palabras clave: administración, configuración, E2guardian, instalación, servicio Proxy, Squid, Squish.

ÍNDICE

INTRODUCCIÓN1

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA SOBRE EL PROCESO DE ADMINISTRACIÓN REMOTA DEL SERVICIO TELEMÁTICO PROXY1

1
3
7
9
10
1.5.1 Lenguaje de modelado de software10

11
11

12

12

12

14

14
CAPÍTULO II: ANÁLISIS Y DISEÑO DE LA HERRAMIENTA INFORMÁTICA PARA LA ADMINISTRACIÓN REMOTA DEL SERVICIO TELEMÁTICO PROXY DESDE LA PLATAFORMA NOVA-ARST1

1

1

1

6

10

11

13

2.4.2 Patrones Gang of Four (GoF)15

17

CAPÍTULO III: IMPLEMENTACIÓN, PRUEBAS Y EVALUACIÓN DE LA HERRAMIENTA INFORMÁTICA PARA LA ADMINISTRACIÓN REMOTA DEL SERVICIO TELEMÁTICO PROXY DESDE LA PLATAFORMA NOVA-ARST 1

3.1 Implementación1

3.1.2 Diagrama de componentes3

3.1.4 Diagrama de Despliegue3

3.2 Pruebas de software4

3.2.1 Tipos de pruebas5

CONCLUSIONES GENERALES 1

RECOMENDACIONES 1

REFERENCIAS BIBLIOGRÁFICAS 1

ANEXOS 1

ÍNDICE DE TABLAS

Tabla 1: Estructura de una ACL 4
Tabla 2: Descripción de tipos de ACL 4
Tabla 3: Comparativa de herramientas homólogas 8
Tabla 4: Listado de Requisitos Funcionales del Módulo Squid 2
Tabla 5: Listado de Requisitos Funcionales del Módulo Squish 3
Tabla 6: Listado de Requisitos Funcionales del Módulo E2guardian 4
Tabla 7: Listado de Requisitos no Funcionales 6
Tabla 8: Historia de Usuario para el requisito funcional 1 Instalar Squid. 8
Tabla 9: Historia de Usuario para el requisito funcional 2 Desinstalar Squid. 9
Tabla 10:Caso de prueba correspondiente al RF 2. Desinstalar Squid 10
Tabla 11: Caminos Básicos y las secuencias 7
Tabla 12 **:**Camino básico para la ruta 1 7
Tabla 13 **:**Camino básico para la ruta 2 7
Tabla 14 **:**Camino básico para la ruta 3 8
Tabla 15 **:**Camino básico para la ruta 4 8
Tabla 16:Descripción de variables para el caso de prueba 2 10
Tabla 17:Descripción de variables para el caso de pruebas 3 11

ÍNDICE DE FIGURAS

Ilustración 1:Prototipo de Interfaz de usuario para el servicio Instalar Squid. 8
Ilustración 2:Prototipo de Interfaz de usuario para el servicio Desinstalar Squid. 9
Ilustración 3:Modelo N-capa definido para la plataforma Nova-ARST. 12
Ilustración 4:Diagrama de Paquete de Squid representado por las 4 capas presentes. 13
Ilustración 5:Ejemplo de Patrón Creador 14
Ilustración 6:Fragmento de ejemplo para el patrón Bajo Acoplamiento 15
Ilustración 7:Ejemplo de patrón GoF 16
Ilustración 8:Ejemplo de patrón decorador 16
Ilustración 9:Diagrama de Clase de Diseño de la configuración de ACL, del módulo Squid. 17
Ilustración 10:Ejemplo de aplicación de los estándares de codificación para React(Iteración en Línea)..... 2
Ilustración 11:Ejemplo de aplicación de los estándares de codificación para Python(Identación)..... 2
Ilustración 12:Diagrama de componente de Squid 3
Ilustración 13:Diagrama de despliegue del módulo de servicio proxy. 4
Ilustración 14:Estrategias de pruebas. 5
Ilustración 15:Grafo de camino.
Ilustración 16:Código representativo para realizar la técnica del Camino Básico. 6
Ilustración 17:Resultados de pruebas funcionales..... 13
Ilustración 18:Diagrama de Clase de Diseño -configurar-reglas de filtrado de contenido de E2Guardian..... 6
Ilustración 19:Diagrama de paquete de Squish 6
Ilustración 20:Diagrama de paquete de E2Guardian 7
Ilustración 21:Módulo Squid en la plataforma Nova-ARST 7

INTRODUCCIÓN

El desarrollo de la sociedad en los últimos tiempos, ha permitido un notable avance en las Tecnologías de la Información y las Comunicaciones (TIC), las cuales favorecen de manera global su accesibilidad y esparcimiento a través del mundo, facilitando la interconexión entre las personas e instituciones a nivel mundial, y eliminando barreras espaciales y temporales. Dentro de las TIC, es notable el auge de los servicios telemáticos. Un servicio telemático es aquel que utiliza como soportes servicios básicos, permitiendo el intercambio de información entre terminales con protocolos establecidos para sistemas de interconexión abiertos(García y Lancharro, 1993); ejemplo de ello lo constituyen: servicio de correo, servicio de gestor de base de datos, servicio de nombres de dominio, almacenamiento en la red y servicio Proxy.

El servicio telemático Proxy, es un grupo de sub-servicios asociados que funcionan de forma conjunta sobre un servidor proxy, este es un ordenador que sirve de intermediario entre un navegador web e internet, con el objetivo de lograr establecer correctas reglas de acceso, predefinir sitios indebidos y gestionar cuotas de usuario, en correspondencia con las políticas de seguridad definidas para cada organismo o institución en el que se emplea; por lo que su correcta administración es de vital importancia.

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs) es un proceso global y constante del que Cuba no está exenta. Aunque existen progresos en estas esferas, concurren limitaciones del ámbito legal que impiden acceder a plenitud a todos los avances presentes en el contexto mundial; hechos que han obligado a la sociedad cubana a buscar soluciones propias para enfrentar estas deficiencias. Dentro de las soluciones creadas se encuentra el sistema operativo cubano (Nova), una distribución de GNU/Linux , con el fin de garantizar soberanía tecnológica; y una de las propuestas de sistemas operativos a emplear durante el proceso de migración hacia plataformas de software libre y código abierto en Cuba.(Carballeira, 2021).

El Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI) es el encargado de impulsar la migración a software libre en el país, por lo que sus objetivos van dirigidos al desarrollo de la distribución cubana GNU/Linux Nova, así como la prestación de servicios y desarrollo de soluciones que apoyen esta misión. La Plataforma Nova-ARST es una herramienta que favorece el proceso migratorio de varios servicios telemáticos a plataformas libres y de código abierto, así como su posterior administración de forma remota. La forma básica de administrar el servicio Proxy es modificando los ficheros a partir de procedimientos manuales en consola, lo que trae consigo lentitud en el proceso de configuración del servicio y sus posteriores tareas, además de errores humanos; ya que el proceso de configuración es engorroso y

complejo; siendo en ocasiones imperceptible para los administradores de red poder detectar dónde ocurrió el error, dado que cualquier cambio en el fichero principal de configuración podría detener el servicio sin una notificación previa. Además, una administración defectuosa podría atentar contra las políticas de seguridad de cada organismo o institución debido a una posible brecha en el sistema que posibilitara acceder a los usuarios a sitios no permitidos, u ocupar todo el ancho de banda, afectando de manera significativa la navegación en la entidad.

Para dar solución a la problemática descrita anteriormente se presenta como **problema de investigación** a resolver:

¿Cómo mejorar la administración del servicio telemático Proxy con control de acceso avanzado y gestión de cuotas de manera remota desde la plataforma Nova-ARST?

Para dar respuesta al problema de investigación se plantea como **Objetivo general**:

Desarrollar un módulo para mejorar la administración del servicio telemático Proxy de manera remota desde la plataforma Nova-ARST.

Objetivos Específicos:

1. Elaborar el marco teórico de la investigación sobre el Servicio telemático Proxy.
2. Diseñar un módulo que permita mejorar la administración del servicio Proxy en la plataforma Nova-ARST.
3. Implementar un módulo que permita mejorar la administración del servicio Proxy de manera remota en la plataforma Nova-ARST.
4. Evaluar el módulo que permita mejorar la administración del servicio Proxy de manera remota en la plataforma Nova-ARST.

Se plantea el siguiente **objeto de estudio:** Administración del servicio telemático Proxy y como **campo de acción:** Administración de manera remota del servicio Proxy en sistemas GNU/Linux Nova.

En aras de encaminar la investigación y lograr el objetivo, se formularon las siguientes **preguntas científicas**:

- ¿Qué elementos debe abordar o tener en cuenta el marco teórico de investigación sobre el servicio telemático Proxy?
- ¿Qué elementos se debe presentar en el diseño de una herramienta informática para mejorar la administración remota del servicio telemático Proxy en la plataforma Nova-ARST?

- ¿Qué componentes de software son necesarios implementar con el objetivo de llevar a cabo una herramienta informática para la administración remota del servicio telemático Proxy en la plataforma Nova-ARST?
- ¿Qué pruebas de software se deben aplicar en la evaluación de la herramienta informática para la administración remota del servicio telemático Proxy en la plataforma Nova-ARST?

Durante el desarrollo de la presente investigación se emplearon los siguientes métodos de investigación:

Métodos Teóricos

Analítico-sintético: Se permite realizar la investigación de las herramientas de administración del servicio proxy y un estudio del funcionamiento y las configuraciones necesarias de este servicio, facilitando la implementación de las funcionalidades propuestas.

Inductivo-deductivo: Este método en la parte inductiva permitió realizar un razonamiento partiendo de lo particular a lo general, reflejando en el resultado final lo común que existe en cada bibliografía estudiada. Por su parte la deducción brindó la posibilidad a través de los conocimientos generales adquiridos durante la investigación, inferir otros conocimientos lógicos como las conclusiones parciales y generales de la tesis.

Métodos Empíricos

Entrevista: La entrevista (Anexo 1) se realizó para conocer las características de la propuesta de solución, la obtención de funcionalidades de software y comprender el contexto de la plataforma Nova-ARST durante el proceso de instalación y configuración del servicio telemático Proxy.

Observación: La observación (Anexo 2) se empleó para registrar el comportamiento del proceso de instalación y configuración del servicio telemático Proxy, así como el análisis de diferentes herramientas informáticas que permiten la administración remota.

El documento consta de 3 capítulos estructurados de la siguiente manera:

CAPÍTULO I: Fundamentación teórica sobre el proceso de administración remota del servicio telemático Proxy: Se definen los fundamentos teóricos de la investigación, exponiéndose los principales conceptos relacionados con el objeto de estudio, para una mejor compresión del tema. Se realiza un análisis sobre las experiencias desarrolladas a nivel nacional e internacional, de las cuales se exponen sus principales características y desventajas que demuestran por qué no son utilizadas, y hacen necesaria la realización de la presente tesis. Además, se describen los elementos fundamentales de la metodología, herramientas y tecnologías a utilizar para el desarrollo de la propuesta de solución.

CAPÍTULO II: Análisis y diseño de una herramienta informática para la administración remota del servicio telemático Proxy: Se describe la solución propuesta y se ofrecen detalles de los principales aspectos relacionados con su diseño. Se muestra el levantamiento de los requisitos funcionales y no funcionales capturados dadas las necesidades existentes para la implementación del módulo, partiendo del estudio del estado del arte realizado en el capítulo anterior . También se definen otros elementos importantes dentro del proceso de desarrollo, tales como: historias de usuario, arquitectura, patrones empleados y elementos de la implementación del mismo.

CAPÍTULO III: Implementación y evaluación de la herramienta informática para la administración remota del servicio telemático Proxy: En este capítulo se describen los elementos relacionados con la implementación, reflejando las prácticas de programación y los estándares de codificación utilizados para el cumplimiento de los requisitos funcionales. Se expone la descripción de los casos de pruebas diseñados para la plataforma Nova-ARST, con el objetivo de evaluar su correcto funcionamiento y su correspondencia con las funcionalidades implementadas, así como los resultados obtenidos en la aplicación de las pruebas.

CAPÍTULO I: Fundamentación teórica sobre el proceso de administración remota del servicio telemático Proxy

Introducción

En este capítulo se presenta una descripción general del servicio Proxy y los servicios asociados a este, abordando un estudio de sistemas homólogos, conceptos generales, la metodología de desarrollo de software, así como los lenguajes y herramientas empleadas en el modelado y desarrollo de Nova-ARST.

1.1 Definición de conceptos

Módulo

Los módulos permiten enriquecer las páginas de su Sitio y agregarles más funcionalidad. Son programas web que se componen de dos piezas de software: Una que pasa a formar parte de las páginas del sitio web, y otra que permite administrar los datos mediante usuario y clave en una página de Administración. Un módulo web se crea ensamblando servlets, archivos JPS (*Java Server Pages*, por sus siglas en inglés) y contenido estático como paginas HTML (*Hypertext Markup Language,* por sus siglas en inglés) en una única desplegable. Los módulos web se almacenan en archivos WAR (*Web Application Archive*, por sus siglas en inglés), que son archivos de archivado Java estándar.(Wright, 2022)

Administración remota

Es la funcionalidad de acceder y controlar, total o parcialmente, una serie de programas que permiten realizar ciertos tipos de acciones desde un equipo local, ejecutando diferentes actividades deseadas entre un usuario y otro a través de Internet o una red local.(Wallen, 2010)

Herramientas de administración remota

Son programas usados para conectarse a un ordenador remoto a través de una red y ejecutar las actividades deseadas.(Wallen, 2010)

Servicio Proxy

Proxy es un compendio de servicios y sub-servicios, los cuales de forma conjunta actúan como intermediarios entre un cliente e internet, de esta forma es evidente que cualquier solicitud hacia la red externa tendrá que pasar a través del proxy, donde será controlado, autorizado o denegado, según las políticas de la empresa(*Tech*, 2022). Entre los sub-servicios que conforman un proxy se encuentran: los servicios que

funcionarán como servidor proxy, servidor de filtrado de contenido y la herramienta para la gestión de cuotas de usuarios.

Servidor Proxy

Es una tecnología donde toda la información pasa primero por él, este es el encargado de enviarlo al lugar de destino, impidiendo toda comunicación directa entre nuestro ordenador destino e Internet.(*Tech*, 2022)

Proxy-Caché

Un servidor proxy-caché permite incrementar la velocidad de acceso a internet al mantener localmente las páginas más consultadas por los usuarios de una organización, evitando las conexiones directas con los servidores remotos. Los usuarios configuran su navegador para dirigir el acceso al servidor proxy-caché en vez de ir directamente al destino final. El servidor proxy-caché se encarga de proporcionar el recurso solicitado, bien puede ser obteniéndolo de su caché o accediendo directamente al recurso original; al dar servicio a muchos usuarios la caché contendrá muchos recursos almacenados, beneficiándose toda la organización de ello. El uso de un servidor proxy-caché evita transferencias innecesarias y con ello aumenta la velocidad en la carga de las páginas y se reduce el consumo de ancho de banda, ya que no es necesario pedir una página cuando ya esté almacenada en la caché.(Squid, 2019)

Proxy transparente

Un proxy transparente combina un servicio proxy con NAT (*Network Address Translation*, por sus siglas en inglés) de manera que las conexiones son enrutadas dentro del proxy sin configuración por parte del cliente y habitualmente sin que el propio cliente conozca de su existencia. Es utilizado por los proveedores de servicios de internet. Un servidor proxy transparente es comúnmente instalado únicamente con la finalidad de compartir el internet a varias computadoras, pero normalmente es instalado por los propios usuarios o técnicos sin experiencia en redes, ya que ayudan a centralizar, supervisar y administrar la misma debido a que no es necesario realizar ninguna modificación en la petición, lo cual ocasiona una enorme vulnerabilidad de seguridad a las empresas al no establecer ninguna política de seguridad. Lo cual trae consigo que las computadoras queden expuestas al internet y sean víctimas fáciles de cualquier atacante de la red de redes.(*Taller Digital*, 2017)

Proxy inverso

Es un servidor proxy-caché "al revés" especialmente diseñado para recuperar información de un servidor o varios servidores que tenga asociados. Son los encargados de gestionar de forma transparente todas las solicitudes de recursos en los servidores de destino sin requerir ninguna acción por parte del solicitante. (*Zscaler*, 2020)

1.2 Descripción del proceso de administración remota del Proxy

1.2.1 Configuración básica de Squid como base

Para lograr un mayor entendimiento sobre las configuraciones básicas sobre Squid, se hace referencia a sus principales elementos técnicos.(Barrios, 2017)

Ubicaciones Importantes de archivos de Squid:

- El archivo de configuración de Squid se encontrará localizado en: **/etc/squid/squid.conf.** En la última ramificación squid.conf es donde encontramos algunas directivas asociadas al servicio.
- El archivo de registro de acceso de Squid se encontrará localizado en: **/var/log/squid/access.log.** En la última ramificación Access.log es donde encontramos el archivo asociado al acceso.
- El archivo de registro de caché de Squid se encontrará localizado en: **/var/log/squid/cache.log.** En cache.log encontramos el fichero a lo que tiene que ver con los datos cache.

Principales parámetros alojados en el archivo de configuración squid.conf, asociados a servicio:

http_port: Esta opción se utiliza con el objetivo de configurar la dirección ip y el puerto para que el servidor Proxy reciba las peticiones, por defecto es 3128,por lo que termina quedando de la siguiente forma: *http_port* 0.0.0.0:3128. El mismo puede ser cambiado por cualquier otro puerto y se le puede añadir la etiqueta **“transparente”** al final de la línea : **http_port 8080 transparente** para hacer que Squid proxy actúe como un proxy transparente.

cache_dir: Esta opción se utiliza con el objetivo de establecer un tamaño para el almacenamiento en caché en el disco duro, aunque Squid emplea el formato ufs[2] (*Unix File System,* por sus siglas en inglés)
para crear en el directorio */var/spool/squid* un caché de 100 MB de manera predeterminada.

Listas de control de acceso

Son listas de condiciones que permiten designar permisos de acceso a cualquier elemento del sistema o directorio en general. En función de estas condiciones, se concede o deniega el acceso a la modificación de las propiedades de estos elementos a los diferentes usuarios o procesos del sistema. La sintaxis de una ACL (*Access Control List*, por sus siglas en inglés) está compuesta por 4 elementos, donde los 2 primeros son obligatorios y luego puede variar entre si se usa el tercero o el cuarto (ver tabla 1).

Tabla 1: Estructura de una ACL

Nombre de la ACL	Nombre que identificará a la lista.
Tipo de ACL	El tipo de acl declarada.(src,dst,port)
Cadena	Cadena de caracteres que se comparará.
Archivo	Archivo que contiene la cadena de caracteres que se comparará.

Sintaxis de ACL

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 localnet src 172.16.100.0/24,siendo localnet el nombre de la ACL, src el tipo de ACL, 172.16.100.0/24 la cadena para comparar.(ver tabla 1). Con la finalidad de comprender mejor las listas de control de acceso se muestran descripciones acerca de varios tipos de ACL(ver tabla 2).

Tabla 2: Descripción de tipos de ACL

Tipo de acl	Descripción	Sintaxis	Ejemplo
src	Para las direcciones ip del cliente o varios clientes según los solicitantes.	acl aclname src dir_ip acl aclname src dir_ip1-dir_ip2	acl centro src 172.16.1.26/32 acl range src 172.16.1.26-172.16.1.42/32
dst	Para la o las direcciones ip de destino.	acl aclname dst dir-ip	acl destino dst 10.8.123.10/80

srcdomain	Para el dominio del cliente solicitante.	acl aclname srcdomain nombre_dominio	acl nueva srcdomain .uci.cu
dstdomain	Para el dominio del destino.	acl aclname srcdomain nombre_dominio	acl nueva dstdomain picta.cu
port	Para los puertos de destino de las solicitudes.	acl aclname port numero_del_puerto acl aclname port rango	acl puertos port 32

Permiso de acceso

Estas definen las ACL que tendrán accesos permitido o denegado a un determinado recurso. La estructura de las reglas son las siguientes:

- http_access allow localnet
- http_access deny localnet

1.2.2 Servicios Asociados

1.2.2.1 Configuración de la herramienta E2guardian para el filtrado de contenido

Es un filtro de contenido web de código abierto que filtra el contenido real de las páginas basadas en diversos métodos:

- frase de correspondencia
- el encabezado de la solicitud
- filtrado de URL

Entre las principales configuraciones de E2guardian se encuentran:

language: Se establece el lenguaje a emplear en el trabajo con la herramienta.

language = ´spanish´

filterip: Se establece el ip por el que se guiará.

filterip = 10.8.1.0

filterport: Se establece el puerto por el que E2guardian escuchará.

filterports = 8080

proxyip: Se establece el ip del servidor proxy.

proxyip = 127.0.0.1

proxyport: Esta opción establece el puerto por el que E2guardian se conectará al proxy.
proxyport = 3128
bannedextensionlist: Se incluyen extensiones de los archivos a denegar.
.bat
bannedsitelist: Se incluirán los sitios a denegar. Estableciéndose un archivo llamado: “domains”, en el que se encuentran una serie de dominios a denegar.
Include</etc/e2guardian/list/blacklist/sex/domains>
Para excluir un sitio por un tiempo determinado:
time:<starthour><startminute><endhour><endminute><days>
Ejemplo:
time:<9><0><17><0><01234>
bannediplist: Se incluyen la dirección o direcciones ip a denegar. Puede ser por ip específico, por un rango determinado o por una determinada subred.
127.0.0.1
10.0.0.1-10.0.0.8
10.0.0.0/24
bannedphraselist: Se incluyen palabras o frases a denegar. Para bloquear cualquier página con palabras que contengan la cadena “eua”.
<eua>
Para bloquear cualquier página con la cadena “política”.
<política>
Para bloquear cualquier página que contenga la palabra/cadena “sex” y “política”.
<sex>, <política>
Para habilitar el bloqueo de frases desde un archivo
Include</etc/e2guardian/list/phraselists/pornography/banned>

1.2.2.2 Configuración de la herramienta para la gestión de cuotas Squish

Es una herramienta que se emplea para garantizar la gestión de cuotas y poder optimizar el ancho de banda de internet. Su función principal consiste en mantener un registro con las cuotas asignadas a cada usuario y cada cierto periodo actualizar el consumo de los usuarios; en caso de encontrar alguno que haya sobrepasado el límite de cuota asignado, Squish enviará este usuario al fichero sin navegación.

La sintaxis para crear un nuevo usuario es la siguiente: *newuser amount/period*, donde *newuser* será el usuario dado, *amount* es la cantidad de cuota que se le asignará a ese nuevo usuario y *period* es el período por el que será válida la cuota. Ejemplo: jyverdecia 140mb/day.

1.3 Análisis descriptivo de herramientas informáticas para administración remota del Proxy

En el siguiente epígrafe se efectúa un análisis comparativo de algunas herramientas informáticas que hacen uso del proceso de instalación y configuración del servicio telemático Proxy en sistemas de código abierto.

Webmin: Es una herramienta de configuración de sistemas accesible vía web para OpenSolaris y GNU/Linux. Permite configurar los permisos para usuarios y grupos o configurar el funcionamiento del servidor Mysql Server, cuotas de espacio, servicios, archivos de configuración y apagado del equipo. Esta herramienta consta con una estructura bien definida por módulos para administrar una amplia variedad de servicios como Apache, PHP, MySQL, DNS, Samba, DHCP, Proxy, entre otros, además de ser completamente configurables y de poder crear nuevos módulos.

Zentyal: Es un servidor de red unificado de código abierto que puede actuar gestionando la infraestructura de red como puerta de enlace a internet (*Gateway*), gestionando las amenazas de seguridad, como servidor de oficina, como servidor de comunicaciones unificadas o una combinación de estas. Para la administración del servicio Proxy este servidor utiliza Squid.(*Zentyal 7.0*, 2004)

Nova_AST: Es una herramienta de código abierto para sistemas GNU/Linux y desarrollada por el Centro de Software Libre (CESOL). Esta plataforma cuenta con una estructura bien definida por módulos para administrar de manera centralizada diversos servicios como Proxy, Postrgres SQL, Apache, DNS, entre otros. Al igual que otras plataformas nos permite administrar el servicio telemático Proxy empleando Squid.

HMAST: Es un acrónimo de “Herramienta para la migración y administración de servicios telemáticos” desarrollada en el año 2012 por el Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas, en forma de una aplicación de escritorio. 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 Soria, 2012)

1.3.1 Análisis comparativo de herramientas informáticas para la instalación y configuración del servicio telemático Proxy

En la (tabla 3) se presenta una comparación de las herramientas estudiadas basada en los criterios de análisis definidos en correspondencia a las deficiencias relacionadas en la situación problemática, las

características del contexto de negocio y la propuesta de solución. En ella se muestra el análisis realizado con los siguientes parámetros:

Lenguaje de programación: Se refiere al lenguaje de programación en el que fue desarrollada la herramienta analizada, el criterio es de Cumple en caso de ser desarrollado a través del lenguaje Python, ya que se pretende desarrollar una herramienta que utilice este lenguaje de programación como base; en caso contrario el criterio es de No Cumple ya que no aporta al desarrollo de esta característica para una nueva herramienta.

Interfaz gráfica de usuario: Para evaluar que la herramienta analizada presenta una interfaz gráfica para la interacción con el usuario.

Administración remota: Para evaluar que la herramienta analizada administra los servicios del proxy Squid de manera remota.

Solución basada en web: Para definir si la herramienta a evaluar es una aplicación web.

Administración centralizada: Para definir si la herramienta a evaluar se basa en una plataforma, capaz de administrar cada uno de los servicios a través de un menú centralizado.

Tabla 3: Comparativa de herramientas homólogas

Fuente: Elaboración propia

Aspectos Comparativos	Zentyal	WebMin	Nova_AST	HMAST
Lenguaje de programación	No Cumple	No Cumple	No Cumple	No Cumple
Interfaz gráfica de usuario	Cumple	Cumple	Cumple	Cumple
Administración remota	No Cumple	No Cumple	No Cumple	Cumple

Solución basada en web	Cumple	Cumple	Cumple	No Cumple
Administración centralizada	Cumple	Cumple	Cumple	Cumple

A partir del análisis realizado, y del presentado, se arriba a la conclusión:

- Ninguna de las herramientas estudiadas emplea el lenguaje Python como base de desarrollo para la gestión o administración de módulos.
- Tres de las cuatro herramientas estudiadas no emplean la administración remota para el manejo de datos a distancia.
- La herramienta HMAST no cumple con el objetivo principal planteado al principio de la investigación, el cual es el desarrollo de un módulo vinculado a una plataforma web.

Al HMAST ser una aplicación de escritorio y aunque permita la administración de sus servicios de manera remota, no es posible desarrollar el módulo para la administración remota del servicio telemático proxy en esta herramienta, por lo que se plantea una alternativa similar pero cumpliendo con el requisito de aplicación web y que este sea capaz de lograr integrar el módulo, además de que permita la administración de manera remota, siendo programado en el lenguaje Python y que pueda integrarse al proceso de Migración del Sistema Operativo NOVA.

1.4 Metodología de desarrollo de software (AUP-UCI)

En el desarrollo de software, la necesidad de organizar o estructurar de forma correcta y disciplinada es uno de los factores más importantes para evitar pérdidas de tiempo y recursos. Con el fin de prevenir tales errores es preciso definir una estrategia que ordene las posibles tareas a desarrollar, así como llevar a cabo una guía de cómo efectuar las actividades, procedimientos y pasos que se deben de seguir para el desarrollo de un software (T. Rodríguez, 2014).

La metodología Variación de AUP para la UCI es una variante de la metodología AUP (*Agile Unfied Process*, Proceso Unificado Ágil). La misma fue definida por la universidad para guiar el proceso de desarrollo de software en la institución ya que se adapta al ciclo de vida definido para la actividad productiva, se ajusta a las características de cada proyecto y logra una mayor homogeneidad entre los

procesos de desarrollo de todos los centros productivos de la UCI(T. Rodríguez, 2014). Esta metodología contiene tres fases: Inicio, Ejecución y Cierre. El desarrollo de la propuesta de solución se realiza empleando la fase de Ejecución ya que es donde se desarrollan las actividades para especificar los requisitos de software, diseñar, implementar y probar la solución.

En la elaboración del módulo informático propuesto, se emplea el escenario 4 de la metodología, para la descripción de los requisitos ya que el mismo se desarrolla dentro del Centro de Software Libre (CESOL), quienes aplican este escenario para los proyectos que no tienen el negocio bien definido y dónde el cliente acompaña al equipo de desarrollo para convenir los detalles de los requisitos, modelando solamente el sistema con historias de usuario.

1.5 Lenguajes y herramientas para el modelado de la solución

1.5.1 Lenguaje de modelado de software

Lenguaje Unificado de Modelado(UML 2.0)

El Lenguaje Unificado de Modelado (UML *Unified Modeling Language* por sus siglas en inglés) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente enriquecedor para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. Permite a los desarrolladores especificar, visualizar, construir y documentar artefactos de un sistema de software. UML tiene aplicaciones más allá del desarrollo de software, por ejemplo, en el flujo de procesos en la fabricación. En este se describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene. Se caracteriza por guardar una relación directa con el análisis y el diseño orientados a objetos.

1.5.2 Herramienta para el modelado de la solución

Las herramientas de modelado se emplean para la creación de modelos de sistemas, estas garantizan un mínimo nivel de calidad en los modelos y minimizan los riesgos, dado que los cambios que se deban realizar por errores o cambios en los requerimientos, son más factibles de realizar sobre el modelo que sobre el propio sistema ya implementado; permitiendo así la creación de un simulacro del sistema.

Visual Paradigm V15. 1

Las herramientas CASE son herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (o mantenimiento). Permiten la automatización de las actividades de gestión de proyectos, la gestión de los productos de los trabajos elaborados a través del proceso de desarrollo y ayudan a los ingenieros en el trabajo de análisis, diseño y codificación. Visual Paradigm es una herramienta CASE para el desarrollo de aplicaciones que utiliza el lenguaje UML, ofreciendo confiabilidad y estabilidad en el proceso de desarrollo. Soporta el ciclo de vida completo del software: análisis y desarrollo orientados a objetos, construcción, prueba y despliegue y brinda un diseño centrado en casos de uso y enfocado al negocio, lo que genera un software de mayor calidad, además permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y la generación de documentación.(*Visual Paradigm, 2019*).Para ayudar a la hora de desarrollar el módulo del servicio Proxy, se hace empleo de esta herramienta para el modelado de los diagramas de prototipado web, los diagramas de componente y de despliegue respectivamente.

1.6 Tecnologías de implementación

En el desarrollo de la propuesta de solución se utilizaron las siguientes tecnologías:

1.6.1 Lenguajes de programación

Python V3.10

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. Administrado por Python Software Foundation, posee una licencia de código abierto, denominada Python Software Foundation License y se logra clasificar constantemente como uno de los lenguajes de programación más populares(*Python*, 2021).

JavaScript V1.5

JavaScript es un lenguaje interpretado usado para múltiples propósitos, pero solo considerado como un complemento hasta ahora. Es utilizado para crear páginas web dinámicas con el objetivo de incorporar efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos, pudiéndose probar directamente en cualquier navegador sin necesidad de procesos intermedios.(*JavaScript , 2022*)

1.6.2 Lenguaje de marcado

HTML 5

HTML (*Hypertext Markup Language*, por sus siglas en inglés) es un lenguaje de marcado usado para la creación de sitios web. Consiste en un conjunto de códigos cortos, que se clasifican como archivos de textos en las etiquetas. Dicho de otra manera, el texto se guarda en un archivo llamado HTML doctype que se puede encontrar a través de los buscadores. Cada una de las etiquetas generadas tiene diferentes funciones. De forma breve, sirve para describir el contenido de un sitio web, como la información estructurada de párrafos e imágenes. De acuerdo a ello, podemos clasificarla como una de las habilidades indispensables para un desarrollador web.(*HTML*, 2022)

1.6.3 Hojas de estilos

CSS

Es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo y bordes. El propósito del desarrollo de CSS es separar la estructura y el contenido de la presentación estética en un documento.(*CSS*, 2022). Se hace empleo a la hora de diseñar la interfaz web de la propuesta de solución.

1.6.4 Marcos de trabajo

Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software. Son usados por programadores**,** ya que permiten acelerar el trabajo y favorecer que este sea colaborativo, logran reducir errores y muestran un mayor resultado en cuanto a la calidad**.** Un framework nos aporta una mayor ganancia de tiempo al desarrollar un proyecto y produce un código ya prediseñado de forma limpia y consistente de manera eficaz. Nos ofrece una estructura base permitiendo complementar o modificar según los objetivos propuestos a la hora de desarrollar un sistema web.(Alegsa, 2016)

Django V 3.2

Django es un framework para el desarrollo web escrito en el lenguaje Python, el mismo permite construir y mantener aplicaciones web de alta calidad con el mínimo esfuerzo posible. Django permite a los desarrolladores concentrarse en el código de la aplicación, manejando tareas repetitivas y complejas propias de la programación web. De esta forma, provee un alto nivel de abstracción para patrones comunes en el desarrollo web, incrementando la calidad de las soluciones, aumentando la productividad y disminuyendo los errores en el código.(*Django, 2022*)

Django REST Framework

Django REST Framework, es un framework que nos permite el fácil desarrollo de una API REST en el lenguaje Python.(*Django REST framework*, 2022)

1.6.5 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (IDE *Integrated Development Environment*, por sus siglas en inglés) es un sistema de software para el diseño de aplicaciones que combina herramientas comunes para desarrolladores en una sola interfaz de usuario gráfica (GUI *graphical user interface*, por sus siglas en inglés). Generalmente, un IDE se características por ser un editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico para el lenguaje y la comprobación de errores a medida que se escribe el código. Posee herramientas que automatizan tareas sencillas y repetitivas como parte de la creación de una compilación local del software para su uso por parte del desarrollador, como la compilación del código fuente de la computadora en un código binario, el empaquetado de ese código y la ejecución de pruebas automatizadas.

Visual Studio Code V1.69

El Entorno de Desarrollo Integrado utilizado es el Visual Studio Code. Este entorno es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS que permite trabajar con diversos lenguajes de programación, admite gestionar tus propios atajos de teclado y refactorizar el código. Es gratuito, de código abierto y nos proporciona una utilidad para descargar y gestionar extensiones con las que podemos

personalizar y potenciar esta herramienta. Incluye soporte para depuración, control de Git integrado, resaltado de sintaxis, finalización de código inteligente, fragmentos de código y refactorización de código. También es personalizable, de modo que los usuarios pueden cambiar el tema del editor, los métodos abreviados de teclado y las preferencias.(*Visual Studio Code ,2018*)

1.6.7 Librerías

React V 17.01

React es una librería frontend para el desarrollo de componentes, que permite crear también aplicaciones Javascript modernas con una experiencia de desarrollo amistosa. React está pensada para el desarrollo en base a componentes y la creación de interfaces de usuario dinámicas y avanzadas.(*React, 2021*)

Paramiko V3.2

Es una biblioteca de Python que realiza una conexión con un dispositivo remoto a través de SSH. Paramiko está usando SSH2 como reemplazo de SSL para hacer una conexión segura entre dos dispositivos. También es compatible con el modelo de servidor y cliente SFTP. Es compatible con Linux, Solaris, BSD, MacOS X, Windows y otras plataformas para conectarse de una plataforma a otra. Con este módulo, la conexión SSH y el protocolo SFTP se pueden utilizar para la transferencia de archivos SFTP cómodamente. Permitirá la conexión del SSH garantizando la integridad de datos e el sistema de configuración (*Python*, 2021).

Conclusiones del capítulo

Con la elaboración y análisis del marco teórico sobre el proceso de instalación y configuración del servicio telemático Proxy de manera remota desde la plataforma Nova-ARST y el estudio de los principales conceptos asociados al problema planteado, se pudo:

- Establecer las bases para el desarrollo de la investigación.
- Conocer las características del objeto de estudio.

Además, con la caracterización y comparación realizada de varias herramientas para la administración remota del servicio telemático Proxy se demostró:

- La necesidad de desarrollar un módulo que mejore el proceso de instalación y configuración del servicio telemático Proxy desde la plataforma Nova-ARST.

Empleándose como metodología de desarrollo de software : Variación de AUP para la UCI, lo que permitió:

- La ejecución de las tareas y el desarrollo del producto de trabajo propuesto.
- Definir el empleo de tecnologías libres para garantizar la soberanía tecnológica de la solución.

CAPÍTULO II: Análisis y Diseño de la herramienta informática para la administración remota del servicio telemático Proxy desde la plataforma Nova-ARST

Introducción

En el presente capítulo se explican las diferentes características que presenta el sistema propuesto y se definen los principales elementos a tener en cuenta para la realización del mismo, tal es el caso de los requisitos funcionales y no funcionales, historias de usuario, diseño de la arquitectura y los patrones de diseño que se emplean.

2.1 Propuesta de Solución

En el presente trabajo investigativo, se propone realizar un módulo que permita gestionar de manera remota la instalación y configuración del servicio telemático Proxy, el cual estará constituido por 3 submódulos. Para mantener el filtrado de contenido se hace empleo de E2guardian y para establecer una correcta gestión de cuotas de usuario se utiliza el Squish; ambas integradas sobre Squid como servidor Proxy. Para tener una idea más clara sobre las configuraciones necesarias, en la sección 1.2 se esclarecen elementos conceptuales y técnicos para establecer la correcta realización de las configuraciones en el módulo.

2.2 Artefactos generados en el proceso de desarrollo

El proceso de desarrollo se encuentra enmarcado por el empleo de la metodología AUP-UCI. Teniendo en cuenta que no se modela el negocio y el mismo se ajusta al escenario 4 de esta metodología, las funcionalidades de la herramienta se describen en cada especificación del levantamiento de requisitos, en las historias de usuario y en los diagramas de clases de prototipado web, lo que compone el principal artefacto generado durante el diseño del módulo.

2.2.1 Especificación de Requisitos

Los requisitos de un sistema según el autor Ian Sommerville son la descripción de los servicios proporcionados por el mismo y sus restricciones operativas(Sommerville, 2011). Para un mejor entendimiento de las necesidades del cliente y del sistema en cuestión, fue necesario definir un conjunto de requisitos funcionales y no funcionales mostrados en el apartado de este epígrafe.

Requisitos Funcionales

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer en cuanto a funcionamiento, entradas, salidas y excepciones(Sommerville, 2011). A través de las técnicas de entrevistas con el cliente y la observación del comportamiento de las soluciones existentes para el desarrollo del módulo del servicio Proxy, se obtuvieron un total de 43 requisitos funcionales para el desarrollo del sistema, los cuales se listan en las (tabla 4, tabla 5 y tabla 6) respectivamente :

Tabla 4: Listado de Requisitos Funcionales del Módulo Squid

No.	Nombre del requisito funcional	Descripción	Prioridad
RF1	Instalar Squid	Permite la instalación del servidor Proxy.	Alta
RF2	Desinstalar Squid	Permite desinstalar el servidor Proxy.	Alta
RF3	Mostrar listas de control de acceso	Muestra las ACL del archivo squid.conf.	Media
RF4	Adicionar lista de control de acceso	Permite adicionar ACL al archivo squid.conf.	Alta
RF5	Modificar lista de control de acceso	Permite modificar ACL del archivo squid.conf.	Media
RF6	Eliminar lista de control de acceso	Permite eliminar ACL del archivo squid.conf.	Baja
RF7	Mostrar listado de reglas de control de acceso	Permite mostrar un listado con las reglas de control de acceso del archivo squid.conf.	Media
RF8	Adicionar regla de control de acceso	Permite adicionar regla al archivo squid.conf.	Alta
RF9	Modificar regla de control de acceso	Permite modificar regla al archivo squid.conf.	Media
RF10	Eliminar regla de control de acceso	Permite eliminar regla al archivo squid.conf.	Baja
RF11	Iniciar servicio de Squid	Permite iniciar el servicio del servidor proxy.	Media
RF12	Detener servicio de Squid	Permite detener el servicio del servidor proxy.	Media

RF13	Aplicar cambios	Permite aplicar los cambios realizados en el servidor.	Alta
RF14	Cancelar acción	Permite cancelar los cambios realizados en el servidor.	Alta

Tabla 5: Listado de Requisitos Funcionales del Módulo Squish

No.	Nombre del requisito funcional	Descripción	Prioridad
RF15	Instalar Squish	Permite instalar el complemento para la gestión de cuotas Squish.	Alta
RF16	Desinstalar Squish	Permite desinstalar el complemento para la gestión de cuotas a partir de la herramienta Squish.	Alta
RF17	Mostrar listado de cuotas de usuario	Permite mostrar las cuotas de usuarios en el archivo de configuración squish.conf.	Media
RF18	Adicionar cuota a usuario	Permite adicionar cuota a un usuario determinado en el archivo de configuración squish.conf.	Alta
RF19	Modificar cuota de usuario	Permite modificar cuota de un usuario determinado en el archivo de configuración squish.conf.	Media
RF20	Eliminar cuota de usuario	Permite eliminar cuota de un usuario determinado en el archivo de configuración squish.conf.	Baja
RF21	Aplicar cambios	Permite aplicar los cambios realizados al servidor.	Alta
RF22	Cancelar cambios	Permite descartar los cambios realizados.	Alta

Tabla 6: Listado de Requisitos Funcionales del Módulo E2guardian

No.	Nombre del requisito funcional	Descripción	Prioridad
RF23	Instalar E2guardian	Permite instalar el servicio de filtrado de contenidos.	Alta
RF24	Desinstalar E2guardian	Permite desinstalar servicio de filtrado de contenido.	Alta
RF25	Mostrar listado de sitio excluir	Permite mostrar un listado de los sitios a excluir en el archivo *exceptionsitelist*.	Alta
RF26	Adicionar sitio a excluir	Permite adicionar sitio a excluir en el archivo *exceptionsitelist.*	Media
RF27	Eliminar sitio a excluir	Permite eliminar sitio a excluir en el archivo *exceptionsitelist.*	Baja
RF28	Mostrar listado de dirección ip a excluir	Permite mostrar un listado de las direcciones ip a excluir del archivo *exceptioniplist.*	Alta
RF29	Adicionar dirección ip a excluir	Permite adicionar dirección ip a excluir en el archivo *exceptioniplist*.	Media
RF30	Eliminar dirección ip a excluir	Permite eliminar dirección ip a excluir en el archivo *exceptioniplist*.	Baja
RF31	Mostar listado de frases a excluir	Permite mostrar un listado de frases a excluir en el archivo *exceptionphraselist.*	Alta
RF32	Adicionar frase a excluir	Permite adicionar frase a excluir en el archivo *exceptionphraselist.*	Media
RF33	Eliminar frase a excluir	Permite eliminar frase a excluir en el archivo *exceptionphraselist.*	Baja
RF34	Mostrar listado de sitio a denegar	Permite mostrar un listado de sitios a denegar en el archivo *bannedsitelist*.	Alta

RF35	Adicionar sitio a denegar	Permite adicionar sitio a denegar en el archivo *bannedsitelist*.	Media
RF36	Eliminar sitio a denegar	Permite eliminar sitio a denegar en el archivo *bannedsitelist*.	Baja
RF37	Mostar listado de dirección ip a denegar	Permite mostrar un listado de dirección ip a denegar en el archivo *bannediplist.*	Alta
RF38	Adicionar dirección ip a denegar	Permite adicionar dirección ip a denegar en el archivo *bannediplist.*	Media
RF39	Eliminar dirección ip a denegar	Permite eliminar dirección IP a denegar en el archivo *bannediplist.*	Baja
RF40	Iniciar servicio	Permite iniciar el servicio de filtrado de contenidos.	Alta
RF41	Detener servicio	Permite detener el servicio de filtrado de contenidos.	Baja
RF42	Aplicar cambios	Permite aplicar los cambios realizados al servidor.	Alta
RF43	Cancelar cambios	Permite descartar los cambios realizados.	Alta

Requisitos No Funcionales

Los requisitos no funcionales son restricciones de los servicios del sistema, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes y restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema(Sommerville, 2011). Para el desarrollo de la presente investigación se definieron un total de 7 requisitos no funcionales, los cuales se clasifican en usabilidad, software, hardware y de seguridad. A continuación, se expresan los requisitos no funcionales definidos para un mejor desarrollo del sistema:

Tabla 7: Listado de Requisitos no Funcionales

No.	Nombre del requisito no funcional	Atributo de calidad
RNF1	El módulo debe poseer interfaces gráficas bien formadas y sencillas, en idioma español, para un mejor manejo y entendimiento por parte del usuario.	Usabilidad
RNF2	El sistema debe permitir en el 80% de las veces que con un máximo de 5 clicks sea suficiente para llegar a la información deseada.	Usabilidad
RNF3	El sistema deberá utilizar como sistema de gestión de bases de datos Postgres SQL.	Restricción del Sistema (Software)
RNF4	El módulo se ejecutará sobre el sistema operativo GNU/Linux Nova en su variante Escritorio, con versión 8.	Restricción del Sistema (Software)
RNF5	El ordenador donde se ejecute el módulo requiere de las librerías Python Paramiko V3.2, React V17.01.	Restricción del Sistema (Software)
RNF6	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.	Restricción del Sistema (Hardware)
RNF7	La herramienta brinda la posibilidad de establecer permisos sobre acciones, garantizando que solo acceda quien esté autorizado.	Seguridad (Acceso restringido)

2.2.2 Historias de Usuario

Las historias de usuario (HU) son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes, por lo que una historia de usuario puede tener varios cambios a lo largo de un desarrollo sin afectarse el tiempo (Francino, 2019).

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

HU-1	
Número: RF1	**Nombre del requisito:** Instalar Squid

Programador: Jordani Y. Verdecia Lago.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 10 horas
Riesgo en Desarrollo: Existencia de un fallo en los servidores a causa de fallas eléctricas e inexperiencia del equipo de trabajo en el desarrollo e implementación del proyecto	**Tiempo Real:** 36 horas

Descripción:

La funcionalidad comienza cuando el usuario selecciona en el menú de módulos la opción Servicio Proxy y luego acceder al apartado Servicio Squid, posteriormente se comprueba que el servicio no esté previamente instalado, de no estarlo es redireccionado a otra página donde hay una breve descripción del servicio y en la esquina inferior derecha se encuentra la opción de instalar. Al terminar el proceso el servicio se instala y concluye el proceso de instalación.

Observaciones:

1. Es necesario antes de instalar realizar una actualización mediante la ejecución del comando: apt upgrade.
2. Si el servicio no se encuentra instalado se podrá instalar, en caso contrario se permitirá desinstalar.
3. La acción de instalar el servicio Squid ejecutará el comando: apt install squid.
4. Se crearán configuraciones por defecto y estas estarán alojadas en el fichero squid.conf situado en el directorio /etc/squid/ .
5. Se crearán configuraciones por defecto en el archivo de logs de acceso access.log que quedará alojado en el directorio /var/log/squid/.
6. También por defecto se definirá el puerto de escucha 3128, posteriormente este puerto podrá ser modificado.
7. Al finalizar el proceso de instalación se mostrará el mensaje de información:

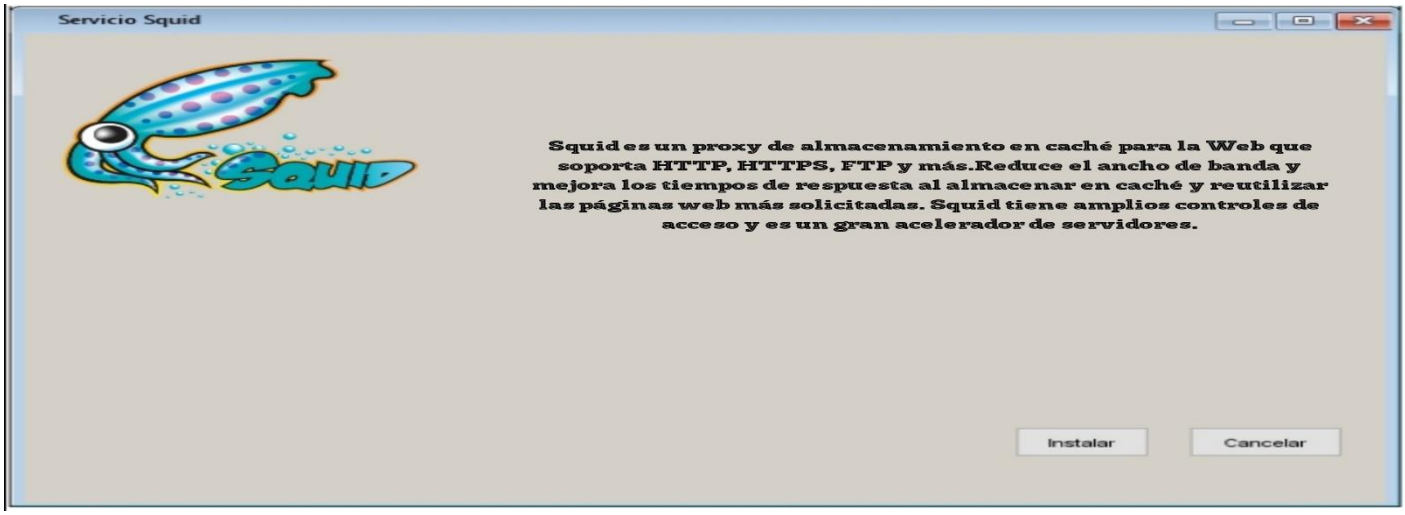
"El servicio ha sido instalado satisfactoriamente.". En caso contrario, el sistema mostrará un mensaje notificando al usuario los detalles del error durante el proceso.
Prototipo elemental de interfaz gráfica de usuario:  *Ilustración 1:*Prototipo de Interfaz de usuario para el servicio Instalar Squid. Fuente: Elaboración Propia

Tabla 8: Historia de Usuario para el requisito funcional 1 Instalar Squid.

HU-2	
Número: RF2	**Nombre del requisito:** Desinstalar Squid
Programador: Jordani Y. Verdecia Lago.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 10 horas
Riesgo en Desarrollo: Existencia de un fallo en los servidores a causa de fallas eléctricas e inexperiencia del equipo de trabajo en el desarrollo e implementación del proyecto	**Tiempo Real:** 24 horas
Descripción: La funcionalidad comienza cuando el usuario selecciona en el menú de módulos la opción Servicio Proxy y luego acceder al apartado Servicio Squid, posteriormente se comprueba si el servicio ya está instalado, de	

estarlo se habilita la opción desinstalar servicio, se procede a desinstalar el servicio instalado, terminando así el proceso de desinstalación.

Observaciones:

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

Prototipo elemental de interfaz gráfica de usuario:

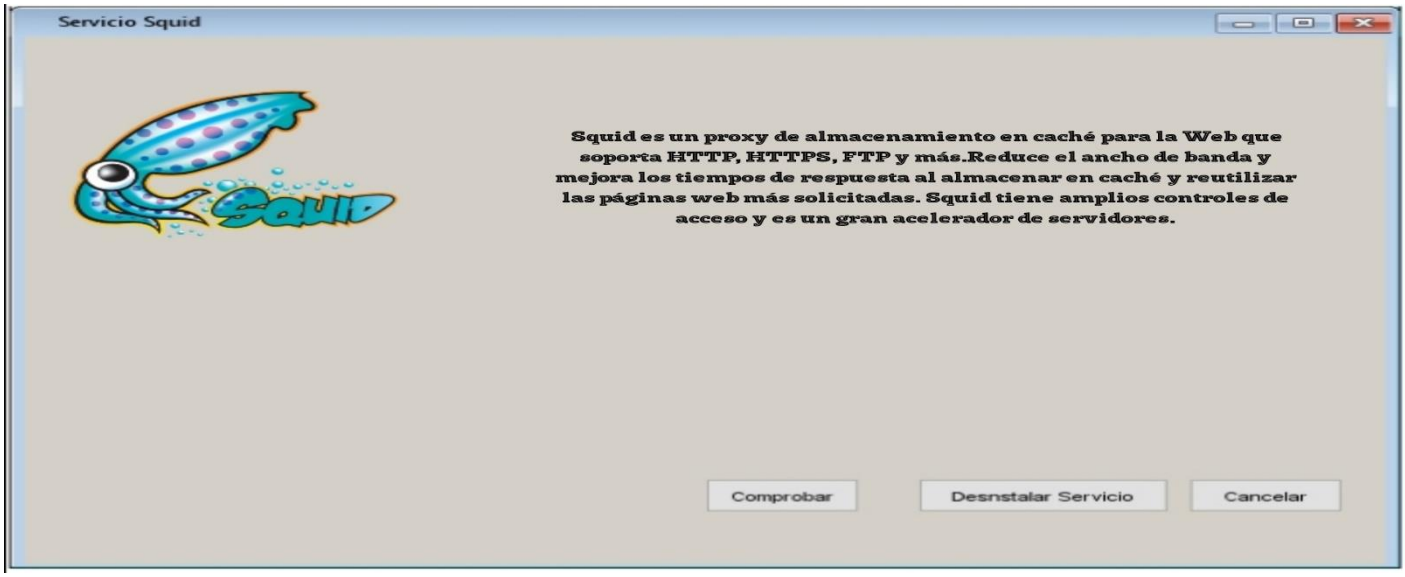


*Ilustración 2:*Prototipo de Interfaz de usuario para el servicio Desinstalar Squid. Fuente: Elaboración Propia

Tabla 9: Historia de Usuario para el requisito funcional 2 Desinstalar Squid.

2.2.3 Validación de Requisitos

El objetivo de la validación de requisitos es asegurar que todos los requisitos del software que se han establecido se correspondan con las necesidades del negocio, de los clientes y de los usuarios, que se han detectado las inconsistencias, omisiones y errores y que estos han sido corregidos(Molina, 2020).Para validar los requisitos obtenidos se aplicaron las técnicas de prototipado de interfaz de usuario y diseños de caso de prueba, los cuales se describen a continuación.

Prototipado de interfaz de usuario: el prototipado de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un software que permite a los clientes, usuarios finales y equipo de desarrollo explorar un diseño de interfaz de usuario alcanzable y adecuado que cumpla los requisitos, ayudando a reducir las distancias entre lo que es necesario y lo que es factible. El objetivo principal de la creación de un prototipo de interfaz de usuario es poder probar el diseño de estas interfaces, incluyendo la capacidad de utilización antes de que empiece el desarrollo real. De este modo, se puede garantizar que se está construyendo el sistema correcto, antes de dedicar demasiado tiempo y recursos al desarrollo de la solución informática.(*IBM Corp ,2018*)

En el prototipado de interfaz de usuario se observan las diferentes configuraciones del servicio Proxy en las variantes que se pueden realizar, ya sea filtrado de datos y un correcto empleo del filtrado de datos. Para ello se utilizaron capturas de pantallas de la implementación de la herramienta.

Diseño de caso de prueba (DCP): son una serie de acciones que se realizan para determinar una función o funcionalidad particular de su aplicación, su objetivo principal es crear casos de prueba para probar los posibles defectos que pudiera tener el sistema y demostrar que se satisfacen los requisitos obtenidos (Cillero, 2020).

La generación de casos de prueba permitió especificar los posibles datos de entrada, así como las salidas de los requisitos funcionales del módulo para la configuración remota del Servicio Proxy. A continuación, se presenta el DCP que corresponde al RF2 Desinstalar Squid.

*Tabla 10:**Caso de prueba correspondiente al RF 2. Desinstalar Squid***

Descripción general: El módulo permite al usuario desinstalar el servidor proxy Squid.

Precondiciones: El servidor proxy debe estar instalado.

Escenario	Descripción	Respuesta del sistema	Flujo central
			Vía 1:

EC 1.1 Verificar si el servidor proxy se encuentra instalado y seleccionar opción “Comprobar”.	Permite comprobar si Squid se encuentra instalado.	Se comprueba si el servidor ya está instalado y muestra como mensaje: “El servidor Proxy se encuentra instalado”. O “No se cuenta con un servidor proxy”	1- Seleccionar la opción que permite comprobar si Squid se encuentra instalado. 2- Seleccionar la opción “Comprobar“. 3- Seleccionar la opción “Desinstalar”. **Vía 2:** 1- Seleccionar la opción que permite salir de la ventana de desinstalación 2- 4- Seleccionar la opción “Cancelar“.
EC 1.2 Cancelar Desinstalar Squid y seleccionar opción “Cancelar”.	Permite Cancelar Desinstalar Squid.	Se permitirá regresar a la vista anterior y cancelar la operación.	**Vía 1:** 1- Seleccionar la opción que permite cancelar la desinstalación de Squid. 2- Seleccionar la opción “Cancelar“.
EC 1.3 Desinstalar Squid y seleccionar opción “Desinstalar”.	Permite Desinstalar Squid.	Se procede a desinstalar Squid, de no estar instalado el servidor proxy, se muestra como mensaje: “No se cuenta con un servidor proxy”	**Vía 1:** 1- Seleccionar la opción que permite la desinstalación de Squid. 2- Seleccionar la opción “Desinstalar“. **Vía 2:** 1- Seleccionar la opción que permite salir de la ventana de desinstalación 2- Seleccionar la opción “Cancelar“.

2.3 Arquitectura del módulo

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(A. Rodríguez y Silva, 2016). Se determina como arquitectura del módulo, la establecida para la plataforma Nova-ARS, que es la herramienta base del sistema (ver Ilustración 3). Se emplea una arquitectura N-Capas orientada al Dominio, la cual tiene como objetivo estructurar de forma clara la complejidad de una aplicación empresarial basada en las diferentes capas de la arquitectura siguiendo el patrón N-Capas y las tendencias de arquitecturas orientadas al dominio.

Ilustración 3: Modelo N-capa definido para la plataforma Nova-ARST (Fuente: Visual Paradigm).

Las entidades de dominio se refieren a la presencia de cada submódulo asociado al servicio Proxy, presente en cada una de las capas de la arquitectura(Squid, Squish, E2guardian).

La capa de Presentación: Se encuentra compuesta por el paquete SquidAppService, es la que contiene las vistas de las clases implementadas al usuario, en ella se encuentran todos los componentes de tipo jsx trabajados con React.

La capa de Aplicación: Se encuentra compuesta por el paquete Squid y dentro de este el paquete DTO. En Squid se encuentra la interfaz ISquidAppService, en la que se pueden encontrar los métodos que serán implementados en la clase SquidAppService; a su vez el paquete DTO es el encargado de transferir los datos entre la capa de Aplicación y la capa de Dominio.

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

La capa de Persistencia: Se encuentra el paquete Squid y dentro de este el paquete Repository, en el que se encuentra la clase SquidRepository, la cual implementa a la clase ISquidRepository definidas en el subpaquete RepositoryContrats en el paquete domain, esta capa trabaja directamente con los ficheros, contiene el código necesario para persistir los datos, es decir, que los cambios se muestren al usuario en el propio momento en que se realizan al mostrar la tabla contenedora en dependencia del servicio a elegir.

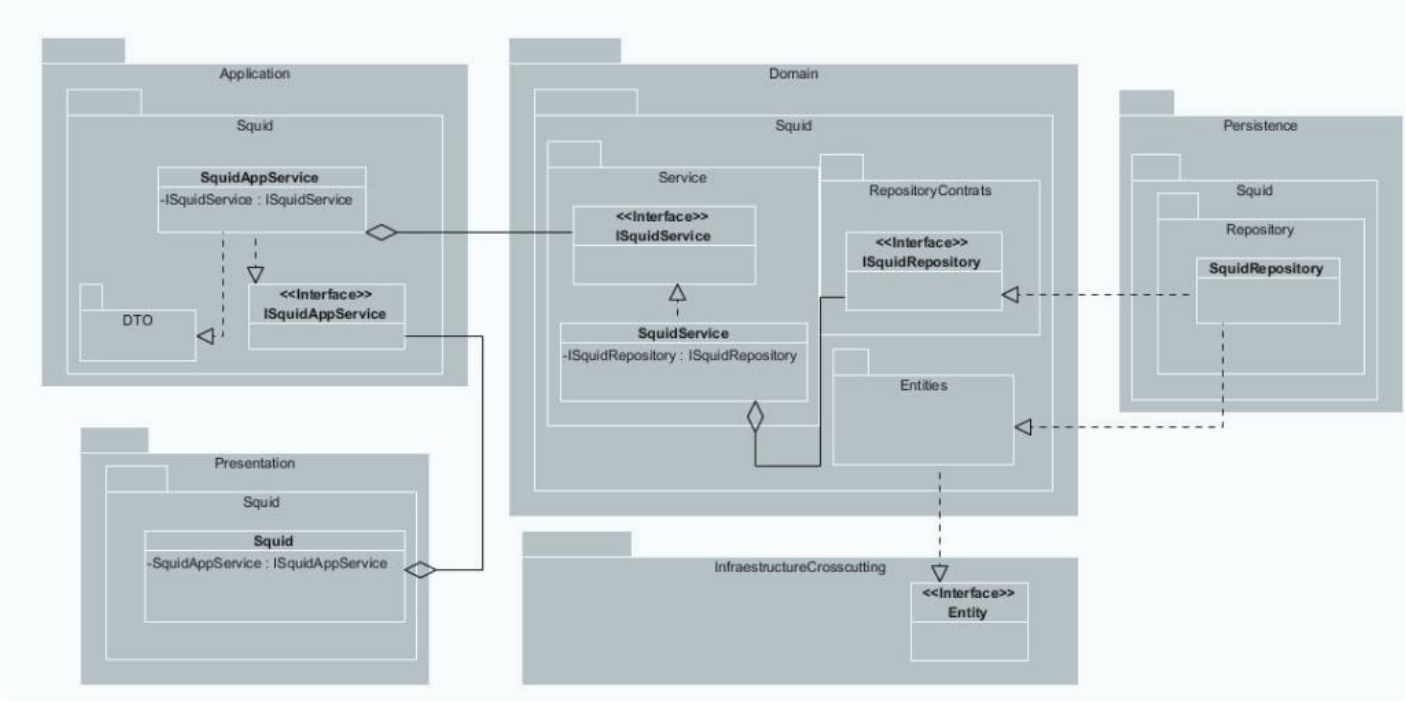


*Ilustración 4 :*Diagrama de Paquete de Squid representado por las 4 capas presentes.

2.4 Patrones de Diseño

En el diseño de las clases del sistema propuesto se aplicaron patrones del grupo GRASP Y GoF. A continuación, se describen los mismos.

2.4.1 Patrones GRASP

Los patrones GRASP, constituyen buenas prácticas en el diseño de software. Estos guardan relación directa con la creación y asignación de responsabilidades a los objetos(Botero, 2019). Para el diseño del módulo se emplearon los siguientes patrones GRASP:

Experto en información o experto: Se utiliza para la asignación de responsabilidades relacionadas con la obtención de información. Conduce a diseños donde los objetos del software realizan aquellas operaciones que normalmente se hacen a los objetos inanimados del mundo real que representan. Se mantiene el

encapsulamiento de la información logrando un bajo acoplamiento entre los objetos y se distribuye el comportamiento entre las clases, lo que estimula las definiciones de clases más cohesivas.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento. Este patrón se utilizó para identificar qué clase A debe crear elementos de una clase B, apoyándose en que la clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B.

En el módulo, la utilización de este patrón es evidenciado cuando las clases presentes en la capa Aplicación crean instancia de las entidades para la conversión de los objetos DTO a objetos del dominio. Un ejemplo es cuando la clase ACLAppService crean un nuevo objeto de tipo ACL para enviarlo a la capa de dominio para su posterior validación.

```
def ACLAppService(clave, valor):

    parser = ConfigParser()
    parser.read(FILE)
    parser.set(CONFIG, clave, valor)
    with open(FILE, 'w') as configfile:
        parser.write(configfile)

    return {'msg': 'Valores creados'}
```

Ilustración 5: Ejemplo de Patrón Creador

Bajo acoplamiento: Consiste en asignar responsabilidades de manera que el acoplamiento permanezca bajo. El uso de este patrón permite la reutilización de las clases y que no se afecten por cambios que se realicen en otros componentes. Este patrón se emplea en las distintas capas del módulo mediante el uso de interfaces que relacionan una capa con otra de forma que dichas relaciones no se establezcan directamente hacia las clases. Las conexiones se realizan a través del mecanismo de inyección de dependencias. Esto se evidencia en el paquete RepositoryContrats, donde se define la interfaz ISquidRepository, y la implementación de sus métodos es realizada por la clase SquidConfRepository en la capa de Persistencia.

```
def SquidConfRepository(host, user, password):
    file_dir = Path(__file__).resolve().parent.parent
    file_dir = str(file_dir) + '/modules/Proxy/Scripts/squid.conf'

    temp = Path(__file__).resolve().parent.parent
    temp = str(temp) + '/modules/Proxy/Scripts/Temp/squid.conf'

    remote_path = '/home/jordi/Escritorio/'
    comando = 'sudo cp squid.conf  /etc/squid/squid.conf'
```

Ilustración 6: Fragmento de ejemplo para el patrón Bajo Acoplamiento

Alta Cohesión: Es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Las clases tienen una responsabilidad moderada en un área funcional y colaboran con otras clases para llevar a cabo una tarea determinada. Permite que las clases sean fáciles de entender, mantener y reutilizar. Se emplea en la clase SquidConfRepository, que tiene la responsabilidad de interactuar con los objetos del fichero de configuración y realizar operaciones de creación, lectura, actualización y eliminación. Para el cumplimiento de estas tareas utiliza los métodos loadLocalConfiguration, saveGeneralFormOptions, saveACL, saveRules, delegando en estos las operaciones de conversión de los atributos.

2.4.2 Patrones Gang of Four (GoF)

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

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

```
1 import paramiko
2 ssh = paramiko.SSHClient()
3 ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
4  ssh.connect ("una dirección IP", 22, "nombre de usuario", "contraseña")
```

*Ilustración 7:*Ejemplo de patrón GoF

Patrón Decorador: Es un patrón estructural que extiende la funcionalidad de un objeto dinámicamente de tal modo que es transparente a sus clientes, utilizando una instancia de una subclase de la clase original que delega las operaciones al objeto original. Provee una alternativa muy flexible para agregar funcionalidad a una clase. Uno de los decoradores empleados de la herramienta fue @action utilizado para las acciones de inicio del contenido de filtrado E2Guardian.

```
@action(methods=['get'], detail=False, url_path='iniciare', url_name='iniciare')
def servicesSSHIniciarE(self, request, pk=None):

        respuesta = estadoe2(ip,user,password, 'sudo systemctl start e2guardian')

        print("Respuesta ", respuesta)
        return Response(respuesta)
```

Ilustración 8: Ejemplo de patrón decorador

2.5 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 2 DCD relacionados con la configuración del filtrado de contenido del submódulo E2Guardian y la configuración general de las listas de control de acceso ACL.

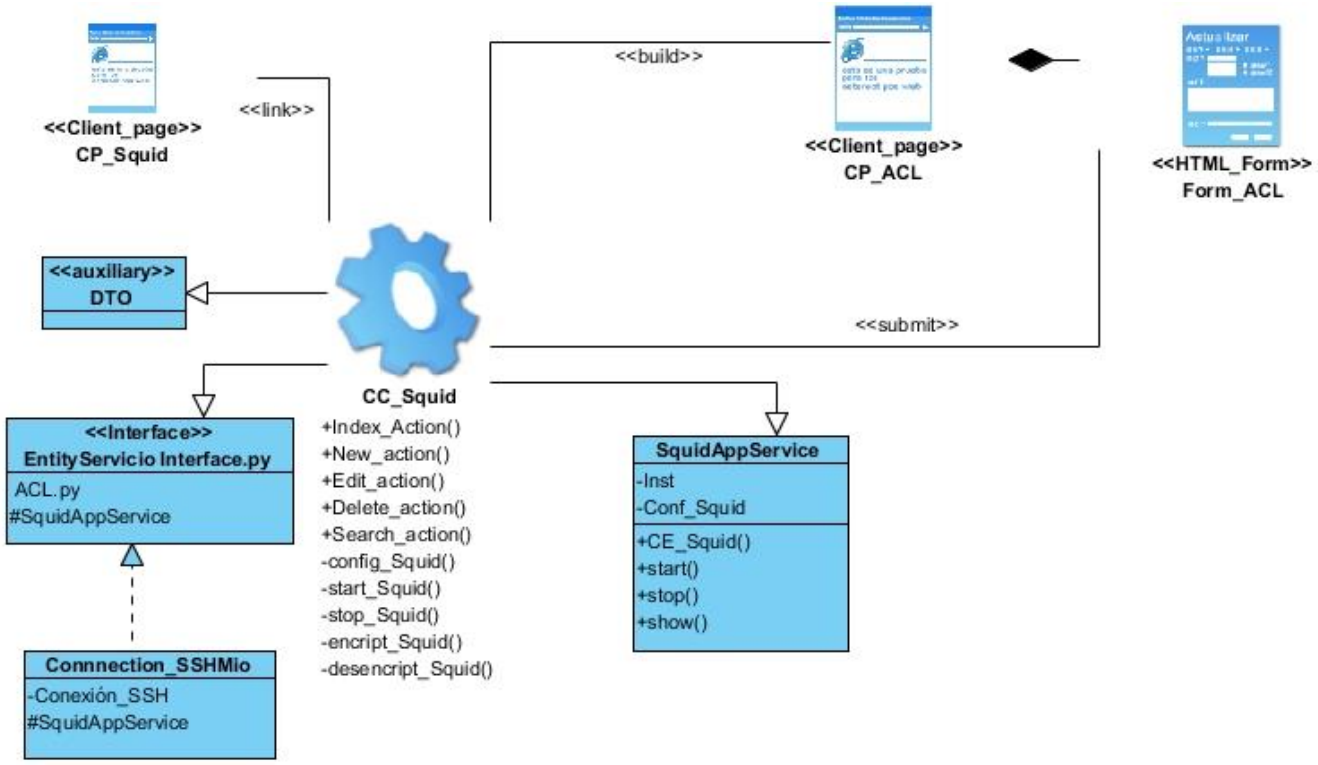


Ilustración 9:Diagrama de Clase de Diseño de la configuración de ACL, del módulo Squid.

Conclusiones del Capítulo

A partir del estudio realizado enfocado en el servicio proxy y el análisis de los elementos homólogos se han logrado definir:

- Los requisitos asociados al módulo, obteniéndose un total de 50 requisitos, de ellos 43 funcionales y 7 no funcionales asociados a la herramienta Nova-ARST, los cuales fueron encapsulados en sus respectivas historias de usuario según plantea la metodología AUP-UCI.
- El estilo arquitectónico en capas, posibilitando definir la organización y la comunicación entre los diferentes componentes de la propuesta de solución y garantizando la integridad de los datos.
- Los patrones de diseño empleados en la propuesta de solución para cumplir con la reutilidad del código.
- Complementando los diagramas de clases de prototipado web correspondientes al módulo Proxy.

CAPÍTULO III: Implementación, pruebas y evaluación de la herramienta informática para la administración remota del servicio telemático Proxy desde la plataforma Nova-ARST

Introducción

En el presente capítulo se muestra el estándar de codificación empleado para el desarrollo del módulo con el objetivo de conservar la analogía de la codificación de los restantes módulos que forman parte de la plataforma, se establecen las estrategias de pruebas, se presentan los estándares de codificación utilizados durante la implementación de la solución y se muestran los diagramas de componentes y despliegue para tener una mejor compresión de la composición física de la plataforma.

3.1 Implementación

La implementación constituye la realización de determinados procesos y estructuras en un sistema. Representa así la capa más baja en el proceso de paso de una capa abstracta a una capa más concreta (Hernández, 2016). Durante la implementación se emplearon los estándares de codificación descritos a continuación, permitiendo al programador mejor legibilidad y testabilidad del código implementado.

3.1.1Estándares de codificación

Los estándares de codificación incorporan principios de ingeniería sólidos para la programación en sus respectivos lenguajes y forman la base de cualquier enfoque preventivo. Su objetivo es inculcar prácticas de programación probadas que conduzcan a un código seguro, confiable, comprobable y mantenible.(Hiken, 2020). Los estándares que se emplean en el módulo se ajustan a los ya establecidos para la plataforma Nova-ARST.

React:

- Iteración en Línea
- Los atributos JSX y HTML utilizan comillas dobles y los de JavaScript utilizan comillas simples

Python:

- **Asignación de nombres:** Emplear descriptores en inglés. Evitar nombres largos y que difieran en una letra o en el uso de mayúsculas. Para nombrar las funciones y variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se utiliza la notación CamelCase.
- **Indentación:** Se utilizan 4 espacios sin caracteres de tabulación o espacios en blanco.

- **Declaraciones:** Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.
- **Tabuladores o Espacios**: Las estructuras de control deben tener un espacio entre la palabra reservada de la estructura y el signo de apertura de paréntesis para distinguir entre las llamadas de las funciones. Las funciones son llamadas sin espacio entre el nombre de la función, el signo del paréntesis y el primer parámetro; espacios entre cada coma por parámetro y sin espacios entre el ultimo paréntesis, el signo de paréntesis cerrado y el signo de punto y coma (;).
- **Longitud de la línea:** Limitar todas las líneas a un máximo de 120 caracteres.

```
return (
  <div>
    {this.props.data.map(function(data, i) {
      return (<Component data={data} key={i} />)
    })}
    </div>
);
```

Ilustración 10: Ejemplo de aplicación de los estándares de codificación para React(Iteración en Línea)

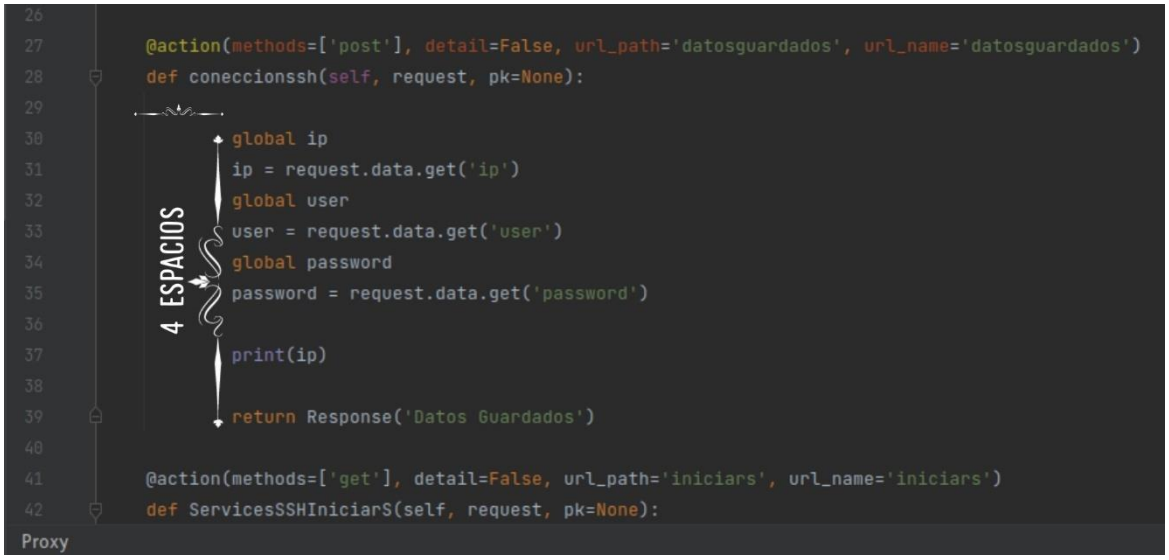


Ilustración 11: Ejemplo de aplicación de los estándares de codificación para Python(Identación)

Fuente: Elaboración propia

3.1.2 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean éstos de código fuente, binarios o ejecutables. Muestra las clases del diseño del software y las relaciones existentes entre ellos en términos de componentes o unidades físicas de implementación con interfaces bien definidas y pensada para ser utilizada como parte reemplazable de un sistema informático(*IONOS*, 2020) . En la Ilustración 9 se presenta el diagrama de componentes de la propuesta de solución. Contiene una representación de las diferentes clases de la herramienta para la administración del servicio proxy vistas como componentes.

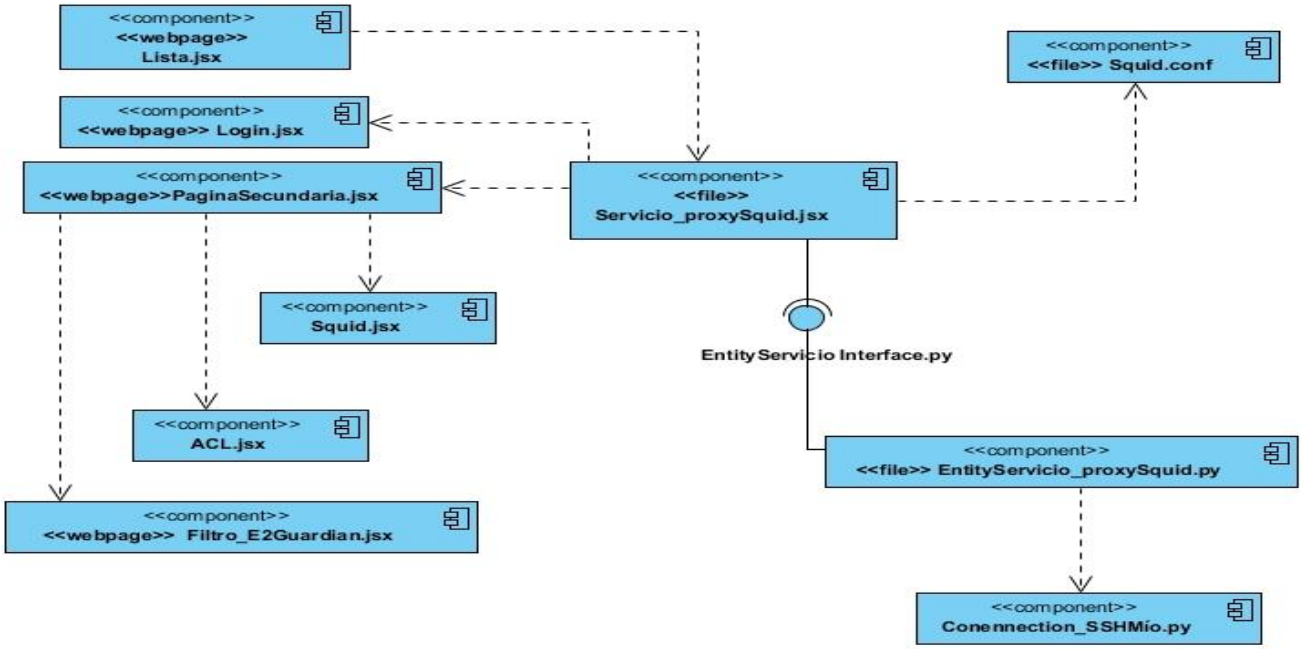


Ilustración 12: Diagrama de componente de Squid

3.1.4 Diagrama de Despliegue

El diagrama de despliegue es aquel que muestra la arquitectura de ejecución de un sistema, incluyendo nodos como entornos de ejecución de hardware o software, estos son utilizados normalmente para visualizar el hardware y el software físico de un sistema(*Visual Paradigm*,2022). En la Ilustración 10 se representa el diagrama de despliegue de la propuesta de solución, el cual muestra los recursos de hardware necesarios para la ejecución de la herramienta.

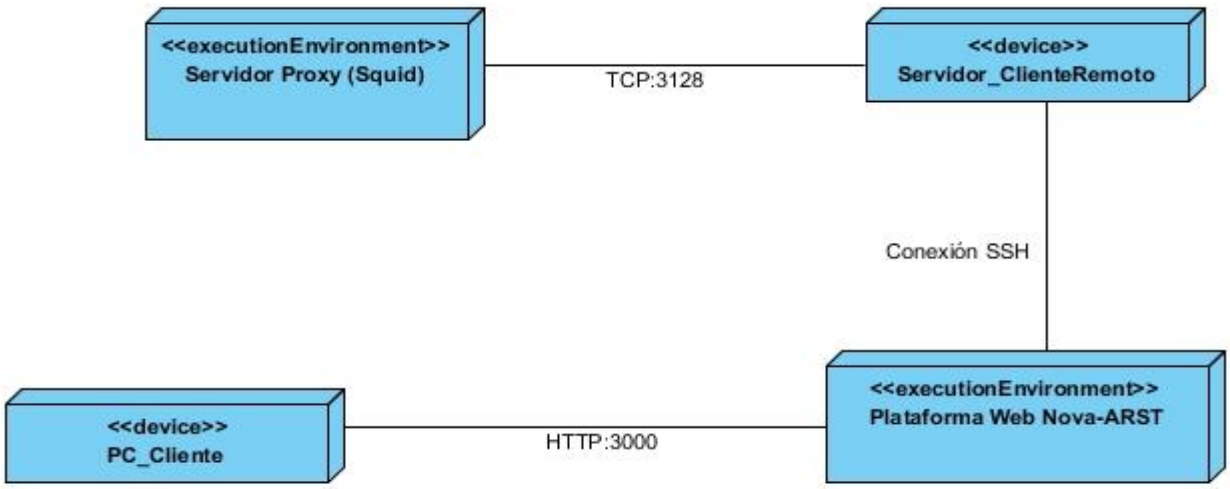


Ilustración 13: Diagrama de despliegue del módulo de servicio proxy.

Fuente: Elaboración propia

Los nodos que integran el diagrama de despliegue son:

PC_Cliente: consume los servicios ofrecidos por la plataforma Web Nova-ARST, permitiendo al personal encargado de la administración de diversos servicios como servicio proxy.

Servidor Proxy(Squid): Es el módulo instalado en el servidor cliente que gestiona todo lo referente con el servicio proxy, lista de control de acceso, filtrado de contenido y gestión de cuotas.

Plataforma Web Nova-ARST: Es la plataforma web a través de la cual el personal se encarga de gestionar diversos servicios, entre ellos el servicio proxy.

Servidor_ClienteRemoto: Es el servidor cliente, donde se instala el servidor proxy de manera remota y se ejecuta de manera segura el servicio proxy.

3.2 Pruebas de software

Las pruebas de software son un conjunto de acciones que pueden ser planificadas con antelación y ejecutarse sistemáticamente durante la implementación o al finalizar el desarrollo del software, estas comprenden un conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación. Las pruebas de software se ejecutan a partir de la aplicación de métodos y técnicas(Pressman, 2010). Para la evaluación de la herramienta propuesta se

definieron las pruebas a partir de las disciplinas establecidas por la metodología AUP-UCI, descritas a continuación.

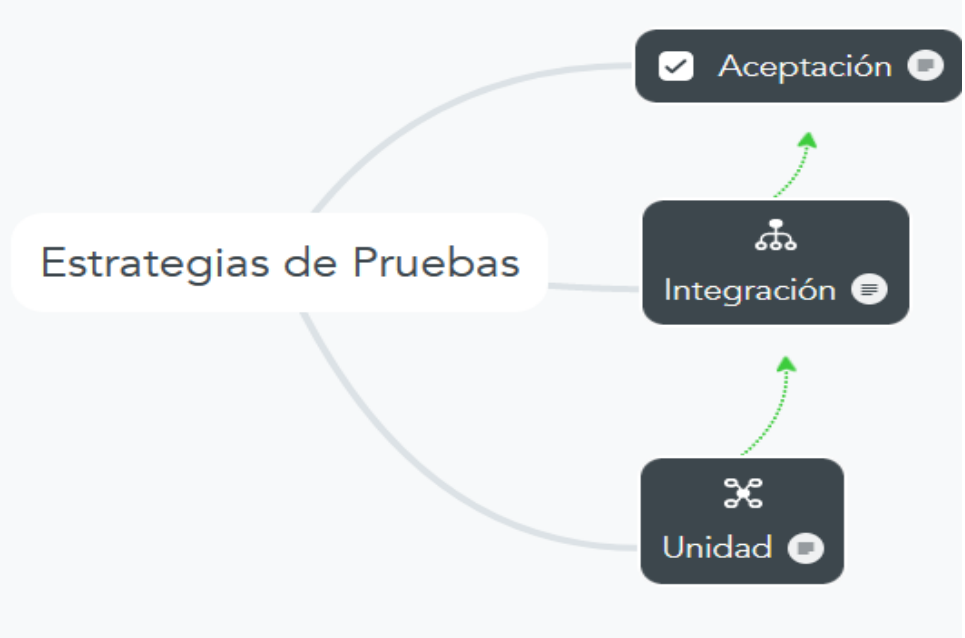


Ilustración 14: Estrategias de pruebas.

3.2.1 Tipos de pruebas

Pruebas Unitarias

Se concentran en el esfuerzo de verificación de la unidad más pequeña del diseño, el componente o módulo de software. Tomando como guía la descripción del diseño a nivel de componente, se prueban importantes caminos de control para describir errores dentro de los límites de la herramienta. El restringido que se ha determinado para las pruebas de unidad limita la relativa complejidad de las pruebas y los errores que éstas revelan(Pressman, 2010). En el caso de la presente investigación en este nivel de prueba se decide aplicar el método de caja blanca y la técnica de camino básico.

El método de Caja blanca se enfoca en probar el sistema teniendo en cuenta la estructura interna del mismo. Verifica la correcta implementación de las unidades internas, las estructuras y sus relaciones y hacen énfasis en la reducción de errores internos(Pressman, 2010). En el presente trabajo de investigación el método fue ejecutado aplicando la técnica de camino básico.

La técnica del camino básico tiene como objetivo comprobar que cada camino se ejecute de manera independiente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño.

Esta técnica debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, al tratar de confirmar que cada camino independiente sea ejecutado al menos una vez en el sistema (Sommerville, 2011).

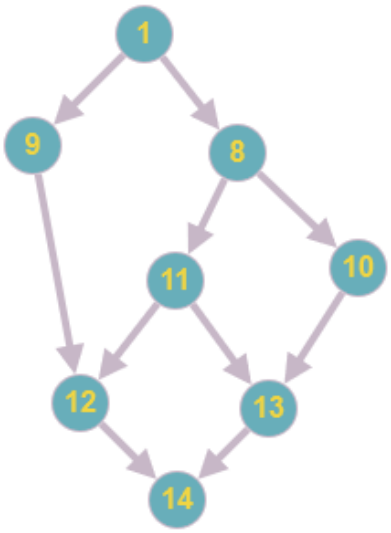


Ilustración 15: Grafo de camino.

```
330
331  def EnviarFicheroSquishConf(host, user, password):
332  1     file_dir = Path(__file__).resolve().parent.parent
333  1     file_dir = str(file_dir) + '/modules/Proxy/Scripts/squish.conf'
334
335  2     temp = Path(__file__).resolve().parent.parent
336  2     temp = str(temp) + '/modules/Proxy/Scripts/Temp/squish.conf'
337
338  3     remote_path = '/home/jordi/Escritorio/'
339  3     comando = 'sudo cp squish.conf  /etc/squid/squish.conf'
340
341  4     print(dir)
342  5     client = paramiko.SSHClient()
343  6     client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
344  7     client.connect(host, username=user, password=password)
345
346  8     sftp_client = client.open_sftp()
```

Ilustración 16: Código representativo para realizar la técnica del Camino Básico.

Técnica del Camino Básico

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

V(g) = a - n + 2

V(g) = 10 - 8 + 2

V(g) = 4

Siendo (n) el número de nodos predicados, se define como la complejidad Ciclomática (V(g)) como:

V(g) = n + 1

V(g) = 3+1

V(g) = 4

Siendo (a) el número de regiones, se tiene como la complejidad Ciclomática (V(g)) como:

V(g) = a

V(g) = 4

La complejidad Ciclomática es 4, por lo que como mínimo se deberán diseñar 4 casos de prueba. Los caminos de pruebas elegidos son los siguientes:

Tabla 11: Caminos Básicos y las secuencias

Caminos	Secuencia
Camino básico 1	**1 – 9 – 12 – 14**
Camino básico 2	**1 – 8 – 10 – 13 – 14**
Camino básico 3	**1 – 8 – 11 – 12 – 14**
Camino básico 4	**1 – 8 – 11 – 13 – 14**

Tabla 12 **:**Camino básico para la ruta 1

Caso de Prueba para el Camino básico 1	
Nombre de quien realiza la prueba: Jordani Yamil Verdecia Lago	**Estado:** Satisfactoria
Descripción: Para la ejecución de esta prueba se debe introducir el id de un LogicalServer y un objeto de tipo ACL, en caso que no exista un Objeto de tipo LogicalServer con el id pasado por parámetro.	
Entradas:	User, password, host
Condición de Ejecución: El valor de la variable idServer no se corresponde con ningún id de ninguno de los objetos de tipo LogicalServer.	
Resultado: Se verifican los parámetros entrados y si no se corresponden, no se adiciona la nueva ACL.	

Tabla 13 **:**Camino básico para la ruta 2

Caso de Prueba para el Camino básico 2	
Nombre de quien realiza la prueba: Jordani Yamil Verdecia Lago	**Estado:** Satisfactoria
Descripción: Para la ejecución de esta prueba se debe introducir el id de un LogicalServer y un objeto de tipo ACL, una vez comprobado que el id sea válido, se verifica la existencia de una ACL con el mismo nombre. En caso de no existir, se adiciona la nueva ACL al listado de ACL del servidor en el archivo de configuración	
Entradas:	idServer,newACL()
Condición de Ejecución:	

El valor de la variable idServer se corresponde con el id de alguno de los objetos de tipo LogicalServer y el nombre de la nueva ACL a adicionar no se corresponde con ninguna de las ya existentes.	
Resultado: Se adiciona la nueva ACL.	

Tabla 14 **:**Camino básico para la ruta 3

Caso de Prueba para el Camino básico 3	
Nombre de quien realiza la prueba: Jordani Yamil Verdecia Lago	**Estado:** Satisfactoria
Descripción: Para la ejecución de esta prueba se debe introducir el id de un LogicalServer y un objeto de tipo ACL, una vez comprobado que el id sea válido, se verifica la existencia de una ACL con el mismo nombre. En caso de existir alguna y esta posea el tipo de ACL igual que la nueva, los valores de la nueva ACL son adicionados a la ya existente.	
Entradas:	idServer,newACL()
Condición de Ejecución: El valor de la variable idServer se corresponde con el id de alguno de los objetos de tipo LogicalServer y el nombre y el tipo de la nueva ACL a adicionar se corresponde con alguna de las ya existentes.	
Resultado: Los valores de la nueva ACL son adicionados a la ACL existente con el mismo nombre y tipo.	

Tabla 15 **:**Camino básico para la ruta 4

Caso de Prueba para el Camino básico 4	
Nombre de quien realiza la prueba: Jordani Yamil Verdecia Lago	**Estado:** Satisfactoria
Descripción: Para la ejecución de esta prueba se debe introducir el id de un LogicalServer y un objeto de tipo ACL, una vez comprobado que el id sea válido, se verifica la existencia de una ACL con el mismo nombre. En caso	

de existir, y posea un tipo diferente a la nueva, se lanza la excepción de tipo ExceptionInvalidACL.	
Entradas:	idServer,newACL()
Condición de Ejecución: El valor de la variable idServer se corresponde con el id de alguno de los objetos de tipo LogicalServer, además el nombre se corresponda con alguna de las ya existentes, pero el tipo sea diferente.	
Resultado: La excepción de tipo ExceptionInvalidACL es lanzada y no se adiciona la nueva ACL.	

Se realizaron dos iteraciones de la prueba unitaria. En la primera iteración se detectaron 2 no conformidades, las cuales fueron resueltas para la segunda iteración. Las no conformidades detectadas estaban asociadas a errores ortográficos y errores de validación.

Pruebas de Integración

Es una técnica para construir la arquitectura del software, cuyo objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que posibilite determinar el diseño(Pressman, 2010). En nuestra investigación, asumiendo la arquitectura por capas empleada y que el módulo debe complementarse con la herramienta base, se emplea una estrategia de integración ascendente, donde los componentes se integran de manera escalonada ascendentemente, realizándose también la prueba de regresión, la cual permite que se vuelva a ejecutar el mismo conjunto de código probado para asegurar que los cambios realizados no han propagado efectos colaterales.

Caso de Prueba 1: Comprobar conexión del componente de filtrado E2Guardian.			
Condiciones de ejecución: Debe estar instalado el servidor proxy Squid y tener disponible un filtro de contenido E2Guardian (estado distinto a “No se cuenta con un servidor Proxy”).			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Probar la conexión del componente de filtrado E2Guardian de manera correcta.	El sistema comprueba la conexión al componente de filtrado de E2Guardian de manera correcta, luego de haber validado los campos del formulario para insertar	Muestra un mensaje de confirmación “La conexión fue exitosa”.	En el menú superior del sistema: 1. El usuario accede a la opción de Configuración. 2. El usuario selecciona la opción E2Guardian y el módulo muestra una lista de los componentes de filtrados de contenidos registrados. 3. El usuario selecciona la opción Adicionar en la parte superior derecha del panel principal. 4. El sistema muestra un formulario para adicionar un nuevo filtro de contenido.

	un elemento de filtrado.		5. El usuario introduce valores válidos y presiona el botón Probar Conexión.

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

Pruebas de Aceptación

Se realizan con el objetivo de validar que el sistema cumple con el funcionamiento esperado y permitir que el usuario determine su aceptación desde el punto de vista de su funcionalidad y rendimiento(Pressman, 2010).Tomándose en cuenta las Historias de Usuario de mayor criticidad, se presentan los casos de pruebas correspondientes, a partir de los cuales el cliente ejecutó las Pruebas.

Para aplicar esta prueba al módulo se diseñaron cuatro(4) Casos de Prueba (CP) que corresponden a los requisitos funcionales RF4,RF11,RF12 y RF32 respectivamente. A continuación, se muestran dos (2) CP correspondientes a los RF4 y RF32, el resto puede ser consultada en el Anexo.

*Tabla 16:*Descripción de variables para el caso de prueba 2.

N.º	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	**Nombre de la ACL**	Campo de texto	No	Nombre que identificará a la lista, longitud mínima 2 caracteres, longitud máxima 16 caracteres. Permite solo caracteres de letras.
2	**Tipo de ACL**	Campo de texto	No	El tipo de acl declarada. Ejemplo de rangos válidos:(src,dst,port)
3	**Cadena**	Campo compuesto(números y puntos)	No	Cadena de caracteres que se comparará. Rango máximo 4 caracteres numéricos separados por punto.

Caso de Prueba 2: SC RF4_Adicionar Lista de Control de Acceso						
Condiciones de ejecución: Debe estar instalado el servidor proxy Squid (estado distinto a “No se cuenta con un servidor Proxy”).						
Escenario	**Descripción**	**1**	**2**	**3**	**Respuesta del sistema**	**Flujo central**
		V	**V**	**V**		

EC 1.1 Adicionar aspectos generales de una lista de control de acceso de manera correcta.	El módulo añade una lista de control de acceso de manera correcta.	localnet	src	10.8.10.4	Muestra un mensaje de notificación: “ACL añadida satisfactoriamente”.	En el menú superior del sistema: 1. El usuario selecciona la opción Proxy Squid *y* el módulo muestra un menú interno. 2. El usuario selecciona la opción de ACL (columna Opciones 2do botón) de Proxy Squid. 3. El sistema muestra un formulario con la información de las ACL actuales del proxy Squid. 4. El usuario introduce los nuevos valores y presiona el botón Guardar y Salir.
EC 1.2 Adicionar aspectos generales de una lista de control de acceso de manera incorrecta.	El módulo añade una lista de control de acceso de manera incorrecta.	**I** Loc3lnet	**V** dst	**I** 255.255.255.0	Muestra un mensaje de error: “no se pudo adicionar la lista, revise los parámetros introducidos”.	
EC 1.3 Adicionar aspectos generales de una lista de control de acceso con campos vacíos.	El módulo no añade una lista de control de acceso con campos vacíos.	**I**	**I**	**V** 10.8.7.2	Muestra un mensaje de error: “no se pudo adicionar la lista, revise los parámetros introducidos, estos campos son requeridos”.	

Tabla 17:Descripción de variables para el caso de pruebas 3

N.º	Nombre del campo	Clasificación	Valor Nulo	Descripción

1	**Frase**	Campo de texto	No	Permite todos los caracteres
2	**Servidor**	Campo de selección	No	Permite seleccionar el servidor

Caso de Prueba 3: SC RF32_Adicionar frase en el filtrado de contenido					
Condiciones de ejecución: Debe estar instalado el servidor proxy Squid y tener disponible un filtro de contenido E2Guardian (estado distinto a “No se cuenta con un servidor Proxy”).					
Escenario	**Descripción**	**1**	**2**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crea una frase de filtrado de manera correcta.	Se inserta una regla de filtrado de manera correcta.	**V** Salida, ilegal	**V** Seleccionar servidor activo	Muestra un mensaje de confirmación “La frase se adicionó con éxito.	En el menú superior del sistema: 1. El usuario accede a la opción Reglas de Filtrado. 2. El usuario selecciona la opción Frases y 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. 3. El usuario introduce los valores de las reglas y presiona el botón Adicionar
EC 1.2 Adicionar frase de filtrado con campos vacíos.	El módulo no inserta una frase de filtrado con campos vacíos obligatorios.	**I**	**I**	Muestra un mensaje de error: “Ocurrió un error al adicionar la frase”.	

Resultados de las pruebas funcionales

Se obtuvo en una primera iteración un total de diez (10) no conformidades, divididas en tres (3) de ortografía y siete (7) de funcionalidad. 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 Ilustración 12.

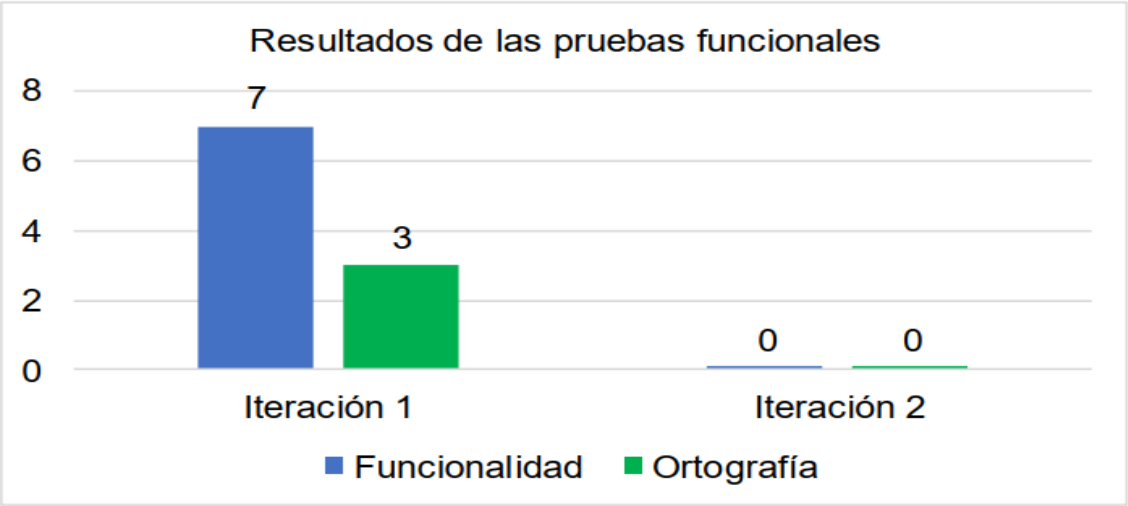


*Ilustración 17:*Resultados de pruebas funcionales

La ejecución de las pruebas funcionales permitió identificar y corregir las deficiencias presentes en la herramienta para la administración remota del servicio telemático proxy.

Las no conformidades funcionales, estaban relacionadas a la validación de los formularios al no poder mostrar la notificación de error especificada en las respuestas del sistema y 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, garantizando la obtención de un producto que responde al problema de investigación con la calidad requerida.

Conclusiones del Capítulo

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

- Una codificación transparente, acorde a las pautas de la herramienta Nova-ARST.
- Una vez desarrollada las funcionalidades del software, se procedió a la documentación y ejecución de pruebas, donde:
    - Las pruebas Unitarias realizadas a cada una de las funcionalidades del software, permitieron comprobar que su flujo es el correcto, pues cada sentencia de código se ejecuta al menos una vez.
    - Las pruebas de aceptación por parte del cliente llegaron a verificar la calidad del módulo implementado. De esta forma se detectaron no conformidades que fueron erradicadas en su totalidad.
- Luego de concluir el proceso de implementación y pruebas realizadas al módulo Proxy integrado a la plataforma Nova-ARST, se obtiene un sistema que cumple con las funcionalidades definidas al inicio de la investigación y se encuentra en condiciones de ser aplicado.

Conclusiones Generales

El presente trabajo de diploma trae como resultado el desarrollo de funcionalidades que permitan desde la plataforma Nova-ARST la administración del servicio Proxy, permitiendo concluir que:

- La correcta utilización de las herramientas, lenguajes y tecnologías descritas, hicieron posible la obtención de un diseño e implementación acertados para el módulo desarrollado, proceso que estuvo guiado sobre la metodología AUP-UCI.
- Las herramientas estudiadas Webmin, Zentyal, Nova-AST y HMAST aportan para la solución del problema, pero no permite su integración a Nova-ARST por la arquitectura que esta herramienta presenta y por la disposición de obtener una administración del módulo Proxy de manera remota.
- Los 43 requisitos funcionales y los 7 no funcionales definidos permitieron cumplir con las necesidades propuestas para el módulo perteneciente a la herramienta Nova-ARST. El diseño propició el modelado de un módulo para la administración remota del servicio telemático Proxy. El uso de los patrones de diseño GRASP y el estilo arquitectónico brindó una mayor calidad al software desarrollado.
- La aplicación de las pruebas de software posibilitó la evaluación de la solución desarrollada y garantizar así, el correcto funcionamiento del módulo implementado. Permitiendo así comprobar la satisfacción de las deficiencias planteadas en la situación problemática y mejorar el proceso de administración de manera general del servicio Proxy en la plataforma Nova-ARST.

RECOMENDACIONES

Con la realización de la presente investigación se lograron desarrollar diversas funcionalidades para la plataforma NOVA-ARST, las cuales permiten la administración de manera remota del servicio Proxy. Para futuras investigaciones y desarrollos se recomienda:

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

REFERENCIAS BIBLIOGRÁFICAS

1. Alegsa, L. (2016). *▷ Definición de Framework de desarrollo (informática)*. https://www.alegsa.com.ar/Dic/framework.php
2. Barrios, J. (2017). *Configuración de Servidores en GNU/Linux* (Enero 2017).
3. Botero, T. (2019). *Patrones Grasp y Anti-Patrones: Un Enfoque Orientado a Objetos desde Lógica de Programación.*
4. Carballeira, F. G. (2021). *Sistemas Operativos: Una Visión Aplicada.: Vol. Volumen II*. Independently published.
5. Castillo, R., & Soria, P. (2012). *Herramienta para la administración y configuración de servidores (HMAS)*. Universidad de las Ciencias Informáticas.
6. Cillero, M. (2020). Diagrama de Clases. *manuel.cillero.es*. https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-clases/
7. *CSS Tutorial*. (s. f.). Recuperado 17 de septiembre de 2022, de https://www.w3schools.com/css/
8. *Diagrama de Clases de Diseño | LENGUAJE DE MODELADO UNIFICADO - UML*. (2016). UNAD. http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html
9. *Diagrama de componentes: Modelado eficiente de sistemas con módulos de software*. (s. f.). IONOS Digital Guide. Recuperado 28 de octubre de 2022, de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-componentes/
10. Francino. (2019). *What is a User Story? Definition from WhatIs.com*. SearchSoftwareQuality. https://www.techtarget.com/searchsoftwarequality/definition/user-story
11. García, J., & Lancharro, E. (1993). *Introducción a la teleinformática* (4º menor. XIV). McGraw-Hill.
12. Hernández, W. G. (2016). La implementación de procesos de informatización en organizaciones como competencia en la formación de profesionales en informática. *E-Ciencias de la Información*, *6*(2), 30-49.
13. Hiken, A. (2020, abril 24). *Estándares de codificación de software y pautas de programación*. Parasoft. https://es.parasoft.com/blog/an-ounce-of-prevention-software-safety-security-through-coding-standards/

14. *Home—Django REST framework*. (s. f.). Recuperado 17 de septiembre de 2022, de https://www.django-rest-framework.org/
15. *HTML: Lenguaje de etiquetas de hipertexto | MDN*. (s. f.). Recuperado 17 de septiembre de 2022, de https://developer.mozilla.org/es/docs/Web/HTML
16. *IBM International Business Machines Corporation Annual 2018*. (s. f.).
17. *Instalación—Documentación de Zentyal 7.0*. (s. f.). Recuperado 17 de septiembre de 2022, de https://doc.zentyal.org/es/installation.html
18. *Introducción a Django—Aprende sobre desarrollo web | MDN*. (s. f.). Recuperado 17 de septiembre de 2022, de https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction
19. *JavaScript | MDN*. (s. f.). Recuperado 17 de septiembre de 2022, de https://developer.mozilla.org/es/docs/Web/JavaScript
20. Molina, Y., Granda, A., & Velázquez, A. (2020). Estrategia de desarrollo de requisitos no funcionales en aplicaciones para la salud. *Revista Cubana de Informática Médica 2020:12(1)92-107 http://scielo.sld.cu*.
21. Pressman, R. (2010). *Ingeniería de Software. Un enfoque práctico.* (Séptima Edición).
22. *React – Una biblioteca de JavaScript para construir interfaces de usuario*. (s. f.). Recuperado 17 de septiembre de 2022, de https://es.reactjs.org/
23. Rodríguez, A., & Silva, L. (2016). Arquitectura de software para el sistema de visualización médica Vismedic. *Revista Cubana de Informática Médica:2016*, *8*.
24. Rodríguez, T. (2014). *Metodología de desarrollo para la Actividad Productiva de la UCI.*
25. *Servidores Proxy Transparentes*. (2017). https://tallerdigitalvw.com/servidores-proxy-transparentes/
26. Sommerville, I. (s. f.). *Software Engineering* (8th edition).
27. Squid, S. P. O. encargado de almacenar en caché todos los accesos y peticiones que se realizan desde su subred hacia el exterior y mantener estas peticiones almacenadas con el fin de que la próxima petición obtenga la respuesta de manera más rápida V., Windows, Ccp. S. O. compatibles M., Propietario, L. L., & GNU/GPL. (s. f.). *Proxy-caché—EcuRed*. Recuperado 12 de septiembre de 2022, de https://www.ecured.cu/Proxy-cach%C3%A9
28. *Visual Paradigm—EcuRed*. (s. f.). Recuperado 17 de septiembre de 2022, de https://www.ecured.cu/Visual_Paradigm

29. *Visual Studio Code—Code Editing. Redefined*. (s. f.). Recuperado 17 de septiembre de 2022, de https://code.visualstudio.com/
30. Wallen, J. (2010, agosto 25). Remote Administration with Linux. *Linux.Com*. https://www.linux.com/training-tutorials/remote-administration-linux/
31. *Welcome to Python.org*. (s. f.). Python.Org. Recuperado 17 de septiembre de 2022, de https://www.python.org/
32. *What is a Proxy Service? - Definition from Techopedia*. (s. f.). Techopedia.Com. Recuperado 12 de septiembre de 2022, de http://www.techopedia.com/definition/31705/proxy-service
33. *What Is a Reverse Proxy?* | *Core Concepts and Definition*. (s. f.). Zscaler. Recuperado 12 de septiembre de 2022, de https://www.zscaler.com/resources/security-terms-glossary/what-is-reverse-proxy
34. Wright, G. (s. f.). *What is a module in software, hardware and programming?* WhatIs.Com. https://www.techtarget.com/whatis/definition/module

35. Cillero, M. (2020). Diagrama de Clases. *manuel.cillero.es*. https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-clases/
36. *CSS Tutorial*. (s. f.). Recuperado 17 de septiembre de 2022, de https://www.w3schools.com/css/
37. Francino. (2019). *What is a User Story? Definition from WhatIs.com*. SearchSoftwareQuality. https://www.techtarget.com/searchsoftwarequality/definition/user-story
38. *Home—Django REST framework*. (s. f.). Recuperado 17 de septiembre de 2022, de https://www.django-rest-framework.org/
39. *HTML: Lenguaje de etiquetas de hipertexto* | *MDN*. (s. f.). Recuperado 17 de septiembre de 2022, de https://developer.mozilla.org/es/docs/Web/HTML
40. *Introducción a Django—Aprende sobre desarrollo web* | *MDN*. (s. f.). Recuperado 17 de septiembre de 2022, de https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction
41. *JavaScript* | *MDN*. (s. f.). Recuperado 17 de septiembre de 2022, de https://developer.mozilla.org/es/docs/Web/JavaScript
42. *React – Una biblioteca de JavaScript para construir interfaces de usuario*. (s. f.). Recuperado 17 de septiembre de 2022, de https://es.reactjs.org/
43. Rodríguez, T. (2014). *Metodología de desarrollo para la Actividad Productiva de la UCI.*

44. Squid, S. P. O. encargado de almacenar en caché todos los accesos y peticiones que se realizan desde su subred hacia el exterior y mantener estas peticiones almacenadas con el fin de que la próxima petición obtenga la respuesta de manera más rápida V., Windows, Ccp. S. O. compatibles M., Propietario, L. L., & GNU/GPL. (s. f.). *Proxy-caché—EcuRed*. Recuperado 12 de septiembre de 2022, de https://www.ecured.cu/Proxy-cach%C3%A9
45. *Visual Studio Code—Code Editing. Redefined*. (s. f.). Recuperado 17 de septiembre de 2022, de https://code.visualstudio.com/
46. Wallen, J. (2010, agosto 25). Remote Administration with Linux. *Linux.Com*. https://www.linux.com/training-tutorials/remote-administration-linux/
47. *Welcome to Python.org*. (s. f.). Python.Org. Recuperado 17 de septiembre de 2022, de https://www.python.org/

ANEXOS

Anexo 1: Entrevista realizada a 3 especialistas de CESOL relacionados con el proceso de migración hacia tecnologías libres y el uso de la plataforma Nova-ARST

Objetivo: Conocer las características del proceso de configuración de los diversos Servicios telemáticos en la plataforma Nova-ARST, deficiencias que se presentan en la ejecución de este proceso y posibles soluciones a dichas deficiencias.

1. ¿Qué características tiene el proceso de configuración del Servicio telemático Proxy?
2. ¿Cuáles son los comandos que deben dominar los especialistas para realizar la configuración del Servicio telemático Proxy en la plataforma Nova-ARST?
3. ¿Qué ficheros son necesarios modificar en el proceso de instalación y configuración del Servicio telemático Proxy en la plataforma Nova-ARST? ¿Cuál es su ubicación?
4. ¿Existen deficiencias durante el proceso de instalación y configuración del Servicio telemático Proxy en el proceso de migración del Sistema Operativo Nova?
5. ¿Cuáles son estas deficiencias?
6. ¿Cuáles son las más comunes?
7. ¿Cómo usted las solucionaría?
8. ¿Cree usted que es importante el uso de una nueva plataforma para de manera remota poder administrar el proceso de instalación y configuración de varios Servicios telemáticos, entre ellos el Servicio Proxy? ¿Por qué?
9. Considera que sería factible administrar de forma remota la configuración del Servicio telemático Proxy en la plataforma Nova-ARST. ¿Por qué?

Anexo 2: Guía de observación para verificar el proceso de instalación y configuración de un Servicio telemático en la plataforma Nova-ARST

Observador: Jordani Yamil Verdecia Lago

Lugar: Laboratorio 103 del Centro de Software Libre (CESOL)

Objetivo: Identificar los elementos fundamentales para la administración de manera remota del proceso de instalación y configuración del Servicio telemático Proxy en la plataforma Nova-ARST.

1. Datos de ejemplos del Servidor Proxy en el que se basan las herramientas informáticas para permitir la administración de manera remota en cuanto a la instalación y configuración del Servicio telemático Proxy : • Nombre de las aplicaciones o Servidores Proxy
• Tipo de licencia
• Cuenta con interfaz gráfica de usuario
• Compatibilidad con diferentes sistemas operativos
2. Características de las herramientas informáticas que permiten la instalación y configuración del Servicio telemático Proxy :
• ¿Cómo son las herramientas informáticas que permiten la instalación y configuración del Servicio telemático Proxy ?
• ¿Cuáles son los pasos a seguir para instalar y configurar estas herramientas?
3. Impacto social de las herramientas informáticas que utilizamos para brindar el proceso de instalación y configuración del Servicio telemático Proxy.

Caso de Prueba 4: SC RF11_Iniciar servicio del proxy Squid

Condiciones de ejecución: Debe estar instalado el servidor proxy Squid (estado distinto a “No se cuenta con un servidor Proxy”).

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 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: 1. El usuario accede a la opción Configuración. 2. El usuario selecciona la opción Proxy Squid y el módulo muestra una lista de los servidores proxy Squid registrados. 3. El usuario selecciona la opción Iniciar proxy (columna Servicio 1er botón) de un proxy Squid disponible
EC 1.2 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	

Caso de Prueba 5: SC RF12_Detener servicio del proxy Squid

Condiciones de ejecución: Debe estar instalado el servidor proxy Squid (estado distinto a “No se cuenta con un servidor Proxy”).

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Detener un proxy correctamente.	El sistema permite detener 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: 1. El usuario accede a la opción Configuración. 2. El usuario selecciona la opción Proxy Squid y el módulo muestra una lista de los servidores proxy Squid registrados. 3. El usuario selecciona la opción Detener proxy (columna Servicio 2do botón) de un proxy Squid disponible.
EC 1.2 Detener un proxy que ya está iniciado	El sistema no permite iniciar un proxy que ya este detenido	El sistema deshabilita el botón que permite detener el proxy Squid	

Número: HU_11	**Nombre del requisito:** Iniciar servicio de Squid.
Programador: Jordani Yamil Verdecia Lago	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1 hora
Riesgo en Desarrollo:	**Tiempo Real:** 2 horas
Descripción: El sistema debe permitir iniciar los servicios del servidor 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 ha 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 elemental de interfaz gráfica de usuario: N/A	

Número: HU_12	**Nombre del requisito:** Detener servicio de Squid.
Programador: Jordani Yamil Verdecia Lago	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 1 hora
Riesgo en Desarrollo:	**Tiempo Real:** 2 horas
Descripción: El sistema debe permitir iniciar los servicios del servidor 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 elemental de interfaz gráfica de usuario: N/A	

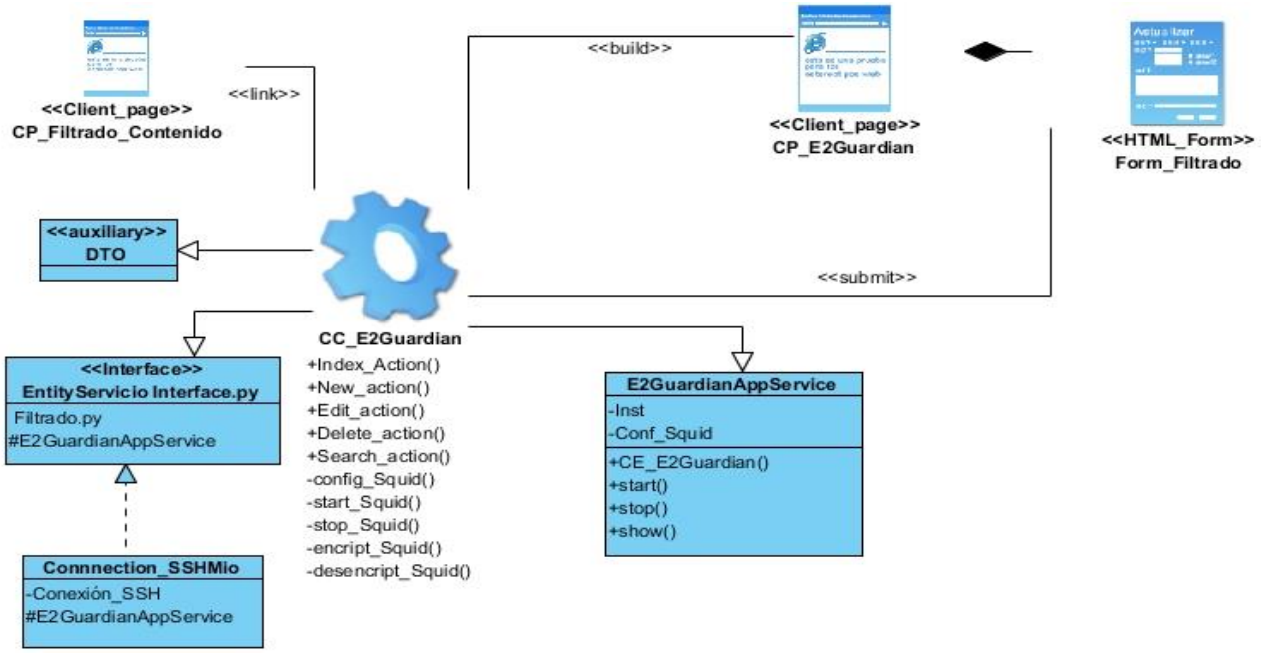


Ilustración 18:*Diagrama de Clase de Diseño -configurar-reglas de filtrado de contenido de E2Guardian.*

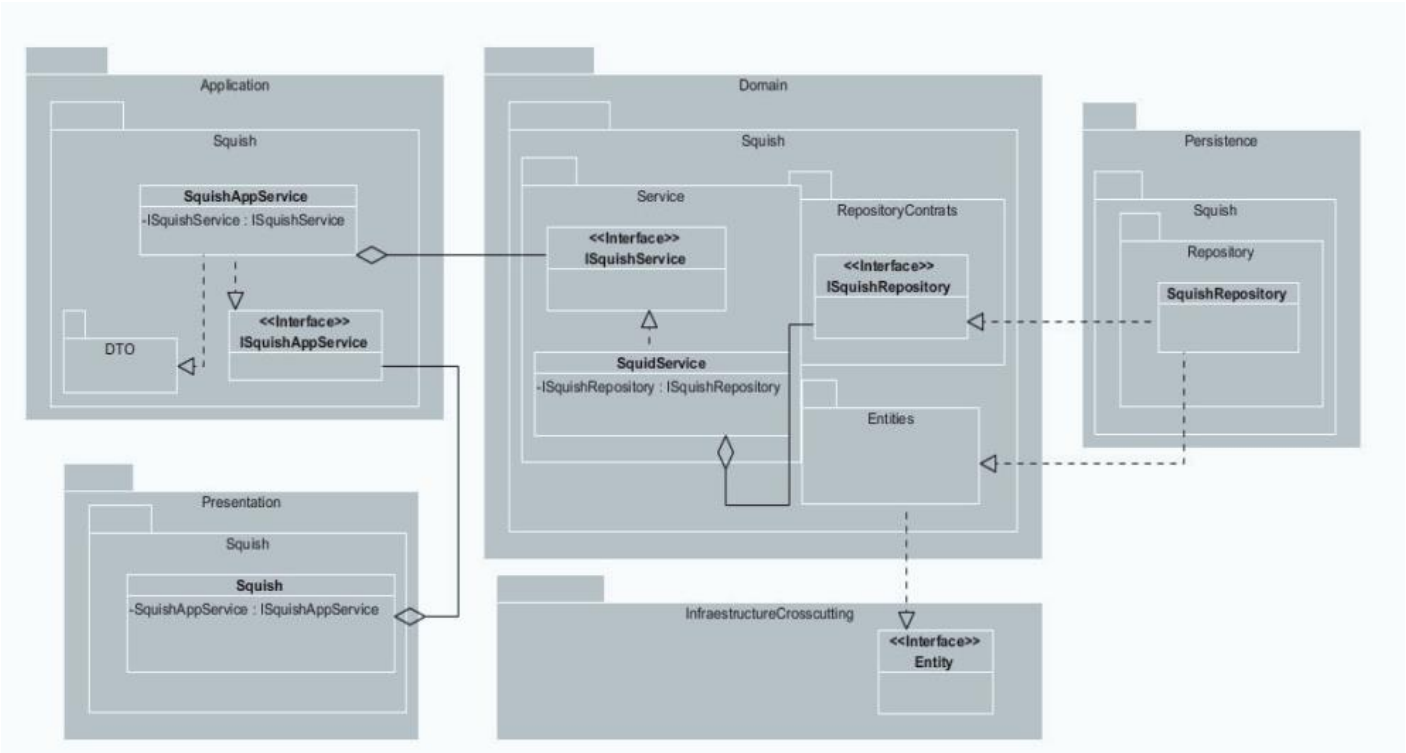


Ilustración 19 :*Diagrama de paquete de Squish*

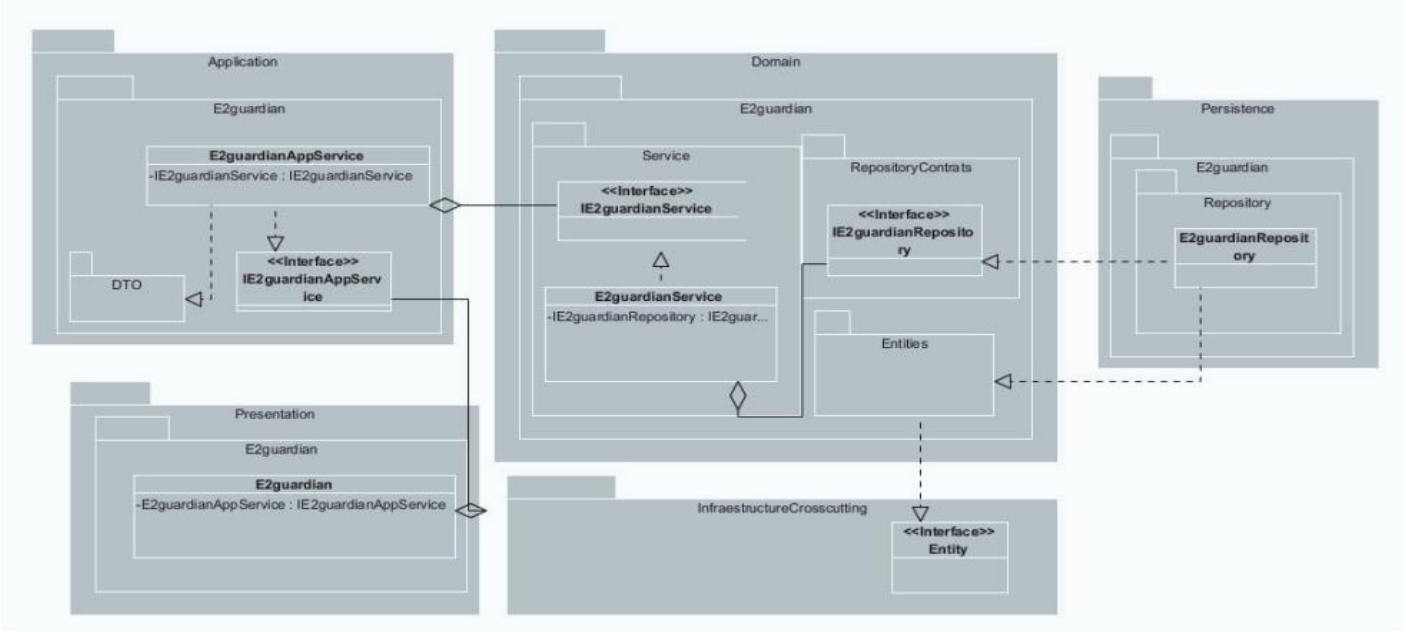


Ilustración 20 :*Diagrama de paquete de E2Guardian*

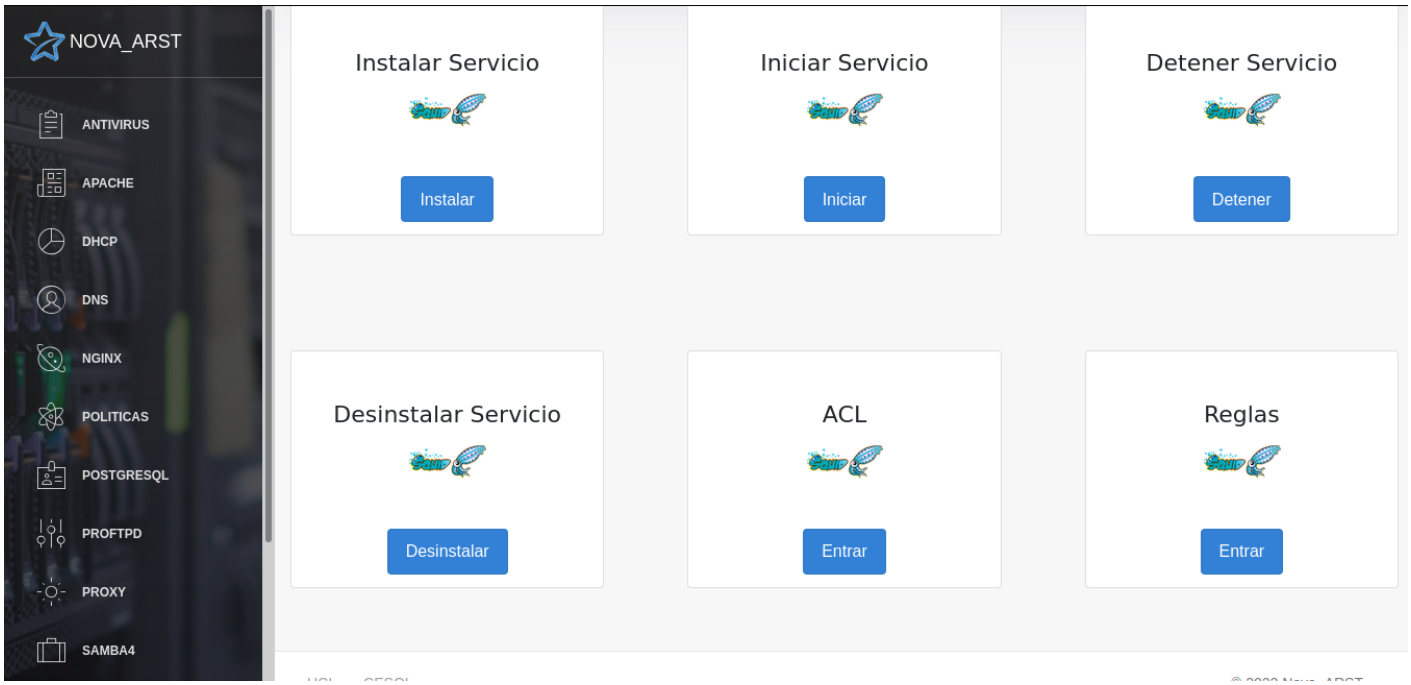


Ilustración 21: Módulo Squid en la plataforma Nova-ARST