



Facultad 1, Centro de Software Libre

Herramienta de configuración de CNTLM para NOVA

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

Autor: Daymel Feliú Delgado

Tutores: MSc. Yurisbel Vega Ortiz
Ing. Aldy León García

La Habana, junio de 2021

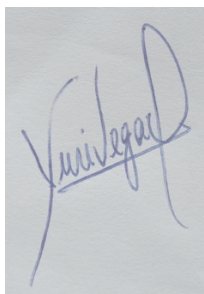
DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo **Daymel Feliú Delgado**, con carné de identidad **98121007622** soy el autor principal del trabajo titulado “Herramienta de configuración de CNTLM para NOVA” 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.

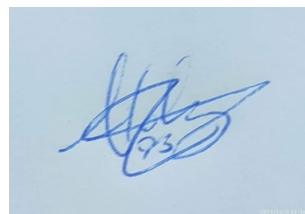
Para que así conste firmamos la presente a los 7 días del mes de diciembre de 2021.



Firma del Autor



Tutor: MSc. Yurisbel Vega Ortiz



Tutor: Ing. Aldy León García

DATOS DE CONTACTO

MSc. Yurisbel Vega Ortiz, Ingeniero Informático graduado en el 2006 en la Universidad de Holguín “Oscar Lucero Moya” (UHo), Máster en Ciencias de la Computación por la Universidad Central de Las Villas (UCLV) en 2013. Profesor Asistente de la disciplina de Programación en el Dpto. de Informática de la Facultad 1. Director del Centro de Software Libre, Facultad 1, UCI desde noviembre de 2018.

Ing. Aldy León García, Ingeniero en Ciencias Informáticas graduado en el año 2017 en la Universidad de las Ciencias Informáticas (UCI). Se desempeña como miembro del equipo de desarrollo del sistema operativo NOVA del Centro de Software Libre (CESOL).

RESUMEN

Las empresas e instituciones gestionan su acceso a Internet generalmente mediante servidores proxy, por razones tan diversas como una mejor administración del ancho de banda, filtrado de contenido, seguridad, anonimato, entre otros factores. Estos servidores proxy generalmente utilizan autenticación NTLM, esta genera una serie de conflictos e inconvenientes con el acceso a Internet de determinadas aplicaciones y sistemas. El Sistema Operativo cubano GNU/Linux Nova, desarrollado en la Universidad de Ciencias Informáticas, enfrenta esta dificultad de autenticación en proxys NTLM así como muchas aplicaciones que corren sobre NOVA, este problema de conectividad se solventa empleando un proxy intermediario, generalmente CNTLM, pero su configuración e instalación puede llegar a ser de gran dificultad para usuarios inexpertos, provocando problemas tan diversos que van desde pérdida del acceso a Internet hasta peligrosas brechas de seguridad. En el presente trabajo de diploma se desarrolla una herramienta para la configuración de CNTLM para NOVA, la herramienta permite crear varios perfiles de configuración de CNTLM, alerta ante posibles fallas de seguridad relacionadas a CNTLM, así como almacena información sensible de manera encriptada, además puede controlar las variables de entorno del Sistema Operativo relacionadas con conexiones proxy, además de tener e iniciar el servicio CNTLM. En el presente trabajo se muestran los resultados de las distintas etapas del proceso de desarrollo, dígase levantamiento de requisitos, análisis, diseño e implementación de la propuesta. Se describen las pruebas que se realizan a la herramienta para validar la propuesta.

Índice

ÍNDICE	V
INTRODUCCIÓN	6
CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA SOBRE EL PROCESO DE CONFIGURACIÓN Y UTILIZACIÓN DE PROXYS PARA ESTABLECER CONEXIÓN CON SERVIDORES NTLM INSTITUCIONALES.	10
1.1 CONCEPTOS FUNDAMENTALES:	11
1.2 Proceso Configuración y ejecución de CNTLM en NOVA	13
1.3 Estudio de Homólogos	14
1.4 Metodología de desarrollo de software	16
1.5 Tecnologías de desarrollo:	18
Conclusiones del capítulo	20
CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO	21
II.1 Representación de la propuesta de solución.	21
II.2 Disciplina de requisitos	22
II.3 Educción de requisitos	22
II.3.1 Descripción de requisitos de software.	24
II.4 DISCIPLINA DE ANÁLISIS Y DISEÑO.	27
II.4.1 Modelo arquitectónico	27
II.4.2 Diagrama de Clases	28
CONCLUSIONES DEL CAPÍTULO	32
CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA	33
III.1 IMPLEMENTACIÓN DEL SISTEMA:	33
III.2 DIAGRAMA DE DESPLIEGUE	38
RECOMENDACIONES	43
BIBLIOGRAFÍA	43
ANEXOS	45

INTRODUCCIÓN

La importancia de las Tecnologías de la Información y la Comunicación (TIC) en la sociedad actual es enorme, estas se encuentran presente prácticamente en todos los sectores de la vida moderna, desde los sectores dedicados a la ciencia e investigación, producción de bienes y servicios, educación, gobierno, al cuidado de la salud, así como nuevas formas de socializar.

Las TIC tienen el potencial de brindar nuevas soluciones a los problemas del desarrollo, en particular en el contexto de la globalización, y pueden promover el crecimiento económico, la competitividad, el acceso a la información y los conocimientos, la erradicación de la pobreza y la inclusión social. (González 2014),

En el informe de 2012 sobre las Tecnologías de la Información y las Comunicaciones (TIC) de la Oficina Nacional de Estadísticas (ONEI) se explicaba que, conceptualmente, la Informatización de la Sociedad se define en Cuba como “el proceso de utilización ordenado y masivo de las TIC para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad. Este proceso busca lograr más eficacia y eficiencia, que permitan una mayor generación de riquezas y hagan sustentable el aumento sistemático de la calidad de vida de los ciudadanos, sobre una política preferentemente orientada al uso social e intensivo de los recursos TIC, para extender sus beneficios a la mayor parte posible de la población y las instituciones. Esta estrategia tiene como centro al ciudadano y busca elevar su calidad de vida en su desempeño familiar, laboral, educacional, cultural y social”.

Como parte del proceso para garantizar la soberanía tecnológica que se lleva a cabo en Cuba, desde el 2004 se desarrolla por estudiantes y profesores de la Universidad de Ciencias Informáticas (UCI), la distribución de GNU/Linux NOVA, liberada por primera vez en 2009. Esta tiene como misión y objetivos, proveer una línea de productos y servicios de calidad orientados a usuarios nacionales inexpertos en el área de las tecnologías de software libre y/o estén experimentado un proceso de migración a las mismas. El sistema operativo NOVA responderá a las necesidades de las instituciones cubanas como parte del proceso de Informatización de la Sociedad de Cuba y promoverá los valores de soberanía e independencia tecnológica.

En muchas instituciones del país, independientemente a que hayan realizado una migración total o parcial, o directamente no haya comenzado su proceso de migración a NOVA, se continúa empleando el sistema NT LAN Manager (NTLM) desarrollado por Microsoft, para garantizar la seguridad y autenticidad de los usuarios que utilizan su red informática para acceder a Internet. Esto provoca que, diversas aplicaciones o sistemas que no son compatibles con este servicio de autenticación, no pueden

acceder a Internet de forma nativa. Para solventar esta dificultad se emplea un sistema intermediario que es capaz de proveer la autenticación requerida por el NTLM. La aplicación más común utilizada para este fin es CNTLM. Esta es capaz de brindar los datos de autenticación necesarios para que dichas aplicaciones accedan a la red de redes.

El Sistema Operativo NOVA en su línea principal no incluye la aplicación CNTLM por defecto, su instalación y configuración queda a disposición del usuario. Esto trae varias dificultades, dentro de ellas la imposibilidad del usuario de acceder a los repositorios oficiales de la distribución desde una institución que utilice el sistema NTLM. El hecho de no acceder a los repositorios de la Distribución representa un problema grave, pues el usuario queda limitado a unas pocas aplicaciones con acceso a Internet (las que son compatibles de manera nativa con NTLM), puede encontrarse incomunicada una aplicación fundamental para la labor que realice en su institución. Además de representar un riesgo para la seguridad informática de su equipo e institución, pues al no tener acceso a los repositorios no podrá recibir las actualizaciones periódicas de seguridad para el sistema operativo y las aplicaciones de terceros. De igual manera, si contase con la aplicación, enfrentará una serie de dificultades asociadas al uso CNTLM, pues para ejecutar CNTLM se debe acceder y editar como superusuario al fichero `cntlm.conf` (archivo de configuración de la aplicación), editar los parámetros de acuerdo al NTLM de su institución, dígase dirección del servidor proxy, puerto, protocolo, excepciones, entre otros parámetros más avanzados como: compartir la autenticación vía proxy. Esta última sección, mal configurada, puede propiciar una brecha de seguridad, pues permitiría que se suplantase la identidad del usuario, por un tercero. Además, un gran inconveniente se encuentra en el factor humano, si este comete algún tipo de error en la edición del archivo `cntlm.conf`, la aplicación puede no ejecutarse correctamente o no ejecutarse. Para iniciar CNTLM es necesario el uso de la CLI del sistema, así como para detenerlo o reiniciarlo. En esta nueva normalidad impuesta por la Covid-19, donde cada vez es más habitual el teletrabajo, estar limitados por deficiencias en la conectividad de un sistema operativo puede marcar la diferencia entre usarlo o no. Estas desventajas e inconvenientes atentan de manera considerable contra las experiencias de usuario y usabilidad, así como la misión de NOVA.

A partir de la situación problemática anteriormente expresada se identifica como problema de la investigación:

¿Cómo dotar a la distribución cubana GNU/Linux NOVA de una forma simple para gestionar la aplicación CNTLM?

Objeto de estudio:

Proceso de configuración de CNTLM en Sistemas Operativos.

Objetivo general:

Desarrollar una aplicación de escritorio para la distribución cubana GNU/Linux NOVA que permita gestionar de forma simple la aplicación CNTLM.

Campo de acción:

Mecanismos de edición de archivos de configuración y ejecución de scripts relacionados a CNTLM en distribuciones GNU/Linux.

Objetivos específicos:

1. Elaborar el marco teórico de la investigación, conceptos relevantes, características y antecedentes de las aplicaciones para configurar el proxy CNTLM.
2. Diseñar una aplicación de escritorio que permita gestionar el proxy CNTLM en el Sistema Operativo GNU/Linux NOVA.
3. Validar de la propuesta de solución.

Hipótesis de investigación:

El desarrollo de una aplicación de escritorio dotará a la distribución cubana GNU/Linux NOVA de una forma simple para gestionar la aplicación CNTLM.

Métodos Científicos

La investigación se realiza sustentada en el empleo de los siguientes métodos científicos:

Métodos teóricos:

- **Histórico-Lógico:** Es utilizado en el análisis de los sistemas homólogos, de manera que permita buscar elementos que los caractericen y aspectos para fundamentar la propuesta de solución a la problemática planteada. Se analizan los sistemas homólogos de

herramientas para proveer autenticación NTLM en sistemas incompatibles en busca de fortalezas y debilidades.

- **Analítico-sintético:** Es aquel método de investigación que consiste en la desmembración de un todo, descomponiéndolo en sus partes o elementos, para observar las causas, la naturaleza y los efectos. Se aplica al caso de estudio mediante el análisis la autenticación en proxys NTLM usando intermediarios para solventar incompatibilidades.
- **Inductivo-Deductivo:** Se emplea para arribar a razonamientos que puedan ser aplicables al problema a resolver, luego de adquirir una serie de elementos referentes a los mecanismos de configuración y ejecución de CNTLM.

Métodos empíricos:

- **Observación:** Se utilizan a través del estudio a herramientas informáticas, que permiten gestionar conexiones a proxys, para determinar las funcionalidades de interés más comunes.
- **Entrevista:** se realizó una entrevista al Ing. Aldy León García perteneciente al Centro de Software Libre con el objetivo de esclarecer dudas y además obtener requisitos orientados a la solución.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA SOBRE EL PROCESO DE CONFIGURACIÓN Y UTILIZACIÓN DE PROXYS PARA ESTABLECER CONEXIÓN CON SERVIDORES NTLM INSTITUCIONALES.

El presente capítulo aborda aspectos sobre la fundamentación teórica de la investigación. Contiene el estudio del proceso de configuración y utilización de clientes proxy con servidores NTLM, reflejado en un análisis de las soluciones existentes para esta tarea. Se define la metodología de desarrollo de software empleada en la elaboración de la propuesta de solución, así como los lenguajes y herramientas empleados en el modelado y desarrollo de la aplicación de escritorio para la gestión de NTLM en NOVA.

I.1 CONCEPTOS FUNDAMENTALES:

- **Sistema Operativo:**

Un Sistema Operativo es el conjunto software básico de una computadora que provee una interfaz entre el resto de programas del ordenador, los dispositivos de hardware y el usuario. Son programas permiten que el ordenador funcione de forma general, que responda a las instrucciones generales como, leer un disco, imprimir un documento, representar una fotografía, reproducir un sonido o instalar otros programas (De la Cruz,2018).

- **Aplicación de escritorio:**

Una aplicación es un programa informático creado para llevar a cabo o facilitar una tarea en un dispositivo informático.

- **CLI:**

CLI es una interfaz de línea de comandos, por sus siglas en inglés. Es un programa que permite a los usuarios escribir comandos de texto instruyendo a la computadora para que realice tareas específicas.

- **Fichero de configuración:**

Los archivos de configuración, son archivos utilizados para configurar los argumentos(parámetros) y la configuración inicial de algunos programas informáticos. Se usan para aplicaciones de usuario, procesos de servidor y configuraciones de sistema operativo. Existen muchos formatos de archivo de configuración diferentes, con cada aplicación o servicio que potencialmente tiene un formato único, generalmente contiene texto plano editable por humanos, y se usa un formato simple de pares clave-valor común. Las extensiones de nombre de archivo de .cnf, .conf, .cfg, .cf o también .ini y similares se utilizan a menudo.

- **Repositorio de una distribución de GNU/Linux:**

Un repositorio básicamente es un Equipo en Internet, es decir, un servidor que aloja los programas específicos para uno o varios Sistemas Operativos Linux, y por lo general están contruidos para acceder vía gestor de paquetes de consola o gráfico, aunque en otros casos incluye el acceso vía Navegador Web. (Blog Desde Linux, s.f.)

- **Autenticación:**

La identificación es la capacidad de identificar de forma exclusiva a un usuario de un sistema o una aplicación que se está ejecutando en el sistema. La autenticación es la capacidad de demostrar que un usuario o una aplicación es realmente quién dicha persona o aplicación asegura ser (IBM MQ, s.f.).

- **Superusuario:**

En Linux (y Unix en general), existe un superusuario llamado root. El equivalente de root en Windows es el grupo de administradores. El superusuario puede hacer cualquier cosa y todo, por lo que hacer el trabajo diario como superusuario puede ser peligroso. Podría escribir un comando incorrectamente y destruir el sistema. Idealmente, se ejecuta como un usuario que solo tiene los privilegios necesarios para la tarea en cuestión. (UbuntuDocumentation, s.f.)

- **Proxy:**

Los Proxies o Proxy's sirven como intermediarios entre la comunicación entre un servidor que se encuentra en Internet y un Cliente o PC que se encuentra en una Red de Área Local (LAN), es decir, este es el encargado de realizar las peticiones de las PC's que este Proxy tiene en su control, y puede o no, según los permisos establecidos por el administrador aceptar o no las peticiones. (Sánchez, 2007)

- **NTLM:**

En una red de Windows, **NT (New Technology) LAN Manager** (NTLM) es un conjunto de protocolos de seguridad de Microsoft destinados a proporcionar autenticación, integridad y confidencialidad a los usuarios. NTLM es el sucesor del protocolo de autenticación en Microsoft LAN Manager (LANMAN), un producto de Microsoft más antiguo. El conjunto de protocolos NTLM se implementa en un proveedor de soporte de seguridad, que combina el protocolo de

autenticación LAN Manager, los protocolos de sesión NTLMv1, NTLMv2 y NTLM2 en un solo paquete. (Microsoft Docs, s.f.)

- **CNTLM:**

Cntlm es un proxy HTTP de autenticación NTLM / NTLM SR / NTLMv2. Se interpone entre aplicaciones y un proxy corporativo, agregando autenticación NTLM sobre la marcha. (Ubuntu Manuals, 2010)

I.2 PROCESO CONFIGURACIÓN Y EJECUCIÓN DE CNTLM EN NOVA

Para ejecutar un proxy CNTLM en un dispositivo con GNU/Linux se debe contar con un equipo con conectividad a un repositorio de la distribución y seguir los siguientes pasos:

Ejecutar en una CLI con permisos de administrador los siguientes comandos:

```
# sudo apt-get update
# sudo apt-get install cntlm
```

Ya con esto el paquete CNTLM debe encontrarse instalado en el sistema, solo resta su configuración ejecutando un editor de texto como administrador y editando el fichero cntlm.conf que se localiza en la ruta /etc/cntlm.conf, ahí se debe completar las siguientes líneas:

- Username #Usuario NTLM
- Domain #Dominio de red
- Password #Contraseña NTLM
- Proxy #Dirección del servidor Proxy

Una vez configurado se debe reiniciar el servicio de CNTLM ejecutando en un CLI el siguiente comando:

```
# sudo systemctl restart cntlm
```

En caso de precisarse conectividad en el sistema o CLI a través del proxy de CNTLM se debe ejecutar los siguientes comandos en la CLI:

```
# export http_proxy=http://127.0.0.1:3128
```

```
# export https_proxy=http://127.0.0.1:3128
```

```
# export ftp_proxy=http://127.0.0.1:3128
```

Para que el navegador web y aplicaciones accedan a Internet a través de CNTLM se debe apuntar en el apartado de proxy de cada aplicación a la dirección del proxy local <http://127.0.0.1:3128>.

Lo anteriormente planteado constituye, a modo de resumen, un ejemplo de configuración de CNTLM. Como se puede apreciar puede llegar a ser un proceso confuso y de dificultad para usuarios básicos, al tratarse del empleo excesivo de la CLI y archivos configuración en texto plano.

I.3 ESTUDIO DE HOMÓLOGOS

En la actualidad existen pocas herramientas dedicadas a proveer un proxy de autenticación NTLM para aplicaciones y sistemas no compatibles, entre ellas están: Winfoom, Px Proxy, Corporate Proxy Helper. A continuación, se presenta una breve caracterización de cada una de ellas.

Winfoom:

Winfoom es una fachada de servidor proxy HTTP (s) que permite que las aplicaciones se autenticuen a través de los siguientes proxies:

- Proxy autenticado HTTP NTLM o Kerberos.
- SOCKS versión 4 o 5, con o sin autenticación.
- Archivos de configuración automática de proxy, incluida la extensión Mozilla Firefox que no forma parte de la especificación original de Netscape.

Normalmente se utiliza en entornos corporativos, sin tener que lidiar con la autenticación real.

Muchas aplicaciones de software tienen problemas cuando se trata de un protocolo de servidor proxy autenticado. Winfoom se encuentra entre el proxy corporativo y las aplicaciones y descarga la autenticación y el protocolo del proxy, actuando como una fachada. De esta manera, la aplicación de software solo tendrá que lidiar con un proxy básico sin autenticación. (Covaci, s.f.)

PX Proxy:

Px es un servidor proxy HTTP (s) que permite que las aplicaciones se autentiquen a través de un servidor proxy NTLM o Kerberos, normalmente utilizado en implementaciones corporativas, sin tener que lidiar con el protocolo de enlace real. Está diseñado principalmente para ejecutarse en sistemas Windows y se autentica en nombre de la aplicación utilizando la cuenta de usuario de Windows actualmente registrada.

La ventaja es que Px puede usar las credenciales del usuario actualmente conectado automáticamente sin requerir ninguna credencial proporcionada por el usuario. Esto se logra utilizando Microsoft SSPI para generar los tokens y firmas necesarios para autenticarse con el proxy. (Viswanathan, s.f.)

Corporate Proxy Helper:

Corporate Proxy Helper está diseñado para reducir los problemas de autenticación en proxys 7, normalmente dependen de la funcionalidad específica de los clientes de Windows, como la autenticación NTLM, los scripts de proxy PAC y Kerberos. Además, los certificados SSL inyectados por el proxy corporativo solo se instalan en la tienda de certificados de Windows y solo unas pocas aplicaciones específicas como el navegador web y algunas herramientas de Microsoft pueden "comprender" todo esto. Corporate Proxy Helper funciona como una capa que permite que dichas aplicaciones y servicios se autentiquen mediante NTML, Kerberos o incluso scripts PAC. También resuelve el problema de los errores con los certificados SSL exclusivos de Windows exportándolos a otros formatos que son entendidos por las herramientas de código abierto. (Corporateproxyhelper, s.f.)

QT-Cuota:

Cuota es un software que muestra el consumo de cuenta de internet de personal de la Universidad de Ciencias Informáticas según el servicio web publicado en <https://cuotas.uci.cu/servicios/v1/InetCuotasWS.wsdl>. Contiene una interfaz visual para el proxy CNTLM y está desarrollado con el framework Qt. (Torres, 2015)

A continuación, se muestra una tabla comparativa de los principales aspectos de las herramientas mencionadas anteriormente:

Aspectos	Winfoom	PX Proxy	Corporate Proxy Helper	QT-Cuota
SO Soportados	Unix/Windows	Linux/Windows/Unix	Windows	Linux/Windows
Licencia	Apache 2.0	MIT	Privativa	N/A
Exportar Proxy al Sistema	No	No	No	No
GUI Intuitiva	No	No	Si	Sí
Posibilidad de modificar código Fuente	No	Sí	No	Sí
Documentación disponible	Sí	Sí	No	No

En la información recogida en la tabla anterior se destacan algunas debilidades y aspectos de peso que imposibilitan que sean tomadas como solución del problema de la investigación, en el caso de Winfoom no cuenta con interfaz gráfica, por lo que dificulta su uso y además está registrada bajo licencia Apache 2.0, que a pesar de ser gratuito y de código abierto, no se puede ser modificado para su mejora o adaptarlo a necesidades específicas de los usuarios. En el caso de PX Proxy carece igualmente de interfaz gráfica y además está solo disponible para Windows. De igual manera solo está disponible Corporate Proxy Helper para el sistema operativo de Microsoft, además está registrado bajo licencia privativa por lo que su uso es inviable. La herramienta QT-Cuota es una solución a medida para la Universidad de Ciencias Informáticas, su documentación no está disponible, lo cual dificulta su modificación, además esta desarrollada en QT4(librería gráfica que está quedando en desuso) actualmente puede generar conflicto con versiones más actuales la librería QT. En base a las dificultades de las herramientas mencionadas anteriormente, se propone desarrollar una solución a medida para solventar el problema de investigación.

I.4 METODOLOGÍA DE DESARROLLO DE SOFTWARE

Según Avison y Fitzgerald (1995) una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las

técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo.”

En la actualidad se pueden diferenciar dos grandes grupos de metodologías de desarrollo de software: las ágiles y las tradicionales. Las metodologías de desarrollo de software tradicionales se caracterizan por definir total y rígidamente los requisitos al inicio de los proyectos de ingeniería de software, los ciclos de desarrollo son poco flexibles, no permiten realizar cambios y están pensadas para el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto. Las metodologías ágiles de desarrollo de software son las más utilizadas hoy en día debido a su alta flexibilidad y agilidad. Los equipos de trabajo que las utilizan son mucho más productivos y eficientes, ya que saben lo que tienen que hacer en cada momento. Además, la metodología permite adaptar el software a las necesidades que van surgiendo por el camino, lo que facilita construir aplicaciones más funcionales.

Para el desarrollo de la solución propuesta se selecciona la metodología de desarrollo de software Variación AUP para la UCI, debido a que es una metodología ágil y es una variación de AUP (Agile Unified Process, Proceso Unificado Ágil) que logra estandarizar el proceso de desarrollo de software en los proyectos de la universidad, además de convertirse en la metodología rectora de su desarrollo productivo, ya que se adapta perfectamente al ciclo de vida de la actividad productiva de los diferentes centros de la institución.

Fases de Variación de AUP para la UCI:

La metodología Variación de AUP - UCI está formada por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad, las cuales contienen las características de las cuatro fases (Inicio, Elaboración, Construcción y Transición) propuestas en AUP.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

I.5 TECNOLOGÍAS DE DESARROLLO:

UML (Lenguaje de Modelado Unificado, Unified Modeling Language) en su versión 2.5 es un lenguaje que se centra en la representación gráfica de un sistema, por lo que permite construir, modelar y diseñar dicho sistema. Un modelo UML está compuesto por 3 clases de bloques de construcción: los elementos (representaciones abstractas de cosas reales o ficticias); las relaciones (relacionan los elementos entre sí); y los diagramas (son colecciones de los elementos con sus relaciones) (Lucid Software Inc., 2018).

Visual Paradigm en su versión 16.3, es una herramienta de Ingeniería de Software Asistida por Computadora (CASE por sus siglas en ingles) que utiliza UML para la completa representación de las etapas por las que transita un producto de software. Este permite la realización de una amplia gama de

diagramas como: de despliegue, de clases, modelos conceptuales empleados para representar el problema y la propuesta de solución.

Como editor de código en la presente investigación se selecciona, **Visual Studio Code** este es desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto. (Visual Studio Code, s.f.)

En la presente investigación se selecciona **Python** como lenguaje de programación tomando en cuenta que es muy versátil y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Es un lenguaje ideal para el desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas de acuerdo a su elegante sintaxis y su tipado dinámico, junto con su naturaleza interpretada (VAN ROSSUM 2009). Además, Python cuenta con abundante documentación y una gran comunidad de desarrolladores, además realiza un manejo eficiente de recursos necesarios para su ejecución en el equipo donde se ejecuta, un punto fuerte extra se encuentra en el intérprete de Python preinstalado en Nova siendo así el lenguaje ideal para desarrollar la propuesta de solución.

Para la implementación de la interfaz gráfica de la propuesta de solución se selecciona GTK (conocido hasta febrero de 2019 como GTK+) o The GIMP Toolkit, esta es una biblioteca de componentes gráficos multiplataforma para desarrollar interfaces gráficas de usuario (GUI). Es uno de los kit de herramientas de widgets más popular para el sistema operativo GNU/Linux, teniendo un amplio soporte para Wayland y XOrg además se usa en gran cantidad de proyectos destacando entre los más famosos : GIMP, Inkscape, Pitivi, Pidgin, XChat, LibreOffice, VLC y Elementary OS. (GTK, s.f.)

Para diseñar la interfaz de usuario de la propuesta de solución se utiliza Glade. Esta es una herramienta que permite un desarrollo rápido y sencillo de interfaces de usuario para el kit de herramientas GTK y el entorno de escritorio GNOME. Las interfaces de usuario diseñadas en Glade se guardan como XML y, al usar el objeto GTK de GtkBuilder, las aplicaciones pueden cargarlas dinámicamente según sea necesario. (Glade - A User Interface Designer, s.f.). Este es un elemento muy importante a tener en cuenta pues permite integrar el código escrito en Python con la interfaz GTK, de manera simple y eficiente, permitiendo elaborar interfaces complejas en un menor tiempo.

CONCLUSIONES DEL CAPÍTULO

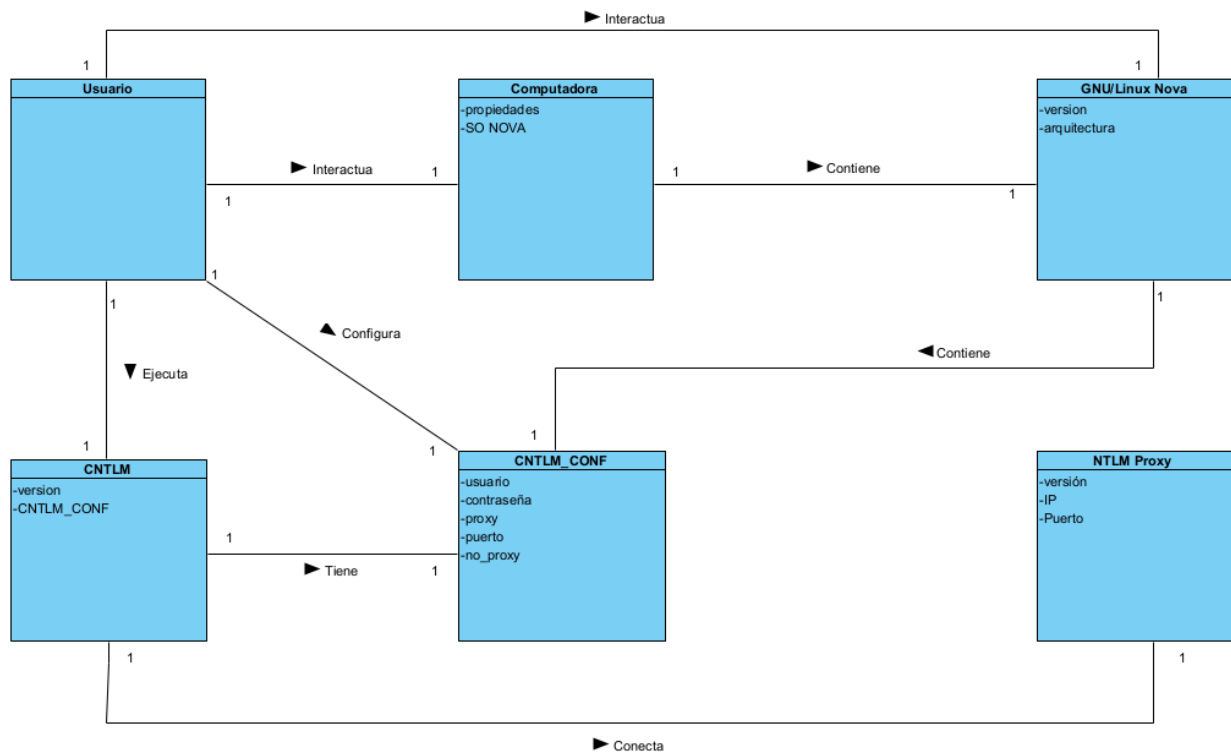
La definición de los conceptos principales que permite una mejor comprensión del objeto de estudio de la investigación, así como análisis de las funcionalidades que brindan algunos de los sistemas homólogos que permiten la comunicación entre sistemas que no soportan la autenticación entre NTLM y sistemas proxy con dicha autenticación, junto a la entrevista realizada permite identificar requerimientos imprescindibles a tener en cuenta en la solución propuesta. El estudio del arte realizado permitió identificar la metodología de desarrollo de software y herramientas que se utilizarán en el desarrollo de la solución que se propone, teniendo como resultado las siguientes: como metodología de desarrollo de software AUP-UCI, como lenguaje de programación se utiliza Python, como biblioteca de componentes gráficos para el desarrollo de la interfaz gráficas de usuario GTK y Visual Studio Code para la edición de códigos.

CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

En el presente capítulo se describen las características de la Herramienta de configuración de CNTLM para NOVA. Se documentan las disciplinas de requisitos, de análisis y de diseño, teniendo en cuenta, la metodología de desarrollo de software AUP-UCI.

II.1 REPRESENTACIÓN DE LA PROPUESTA DE SOLUCIÓN.

Para la representación de la propuesta de solución se identifican los conceptos significativos en el problema, identificando los atributos y las asociaciones entre ellos.



Usuario: Es el usuario final del sistema e interactúa con la computadora.

Computadora: Es un dispositivo informático que es capaz de recibir, almacenar y procesar información de una forma útil.

GNU/LINUX NOVA: Sistema Operativo que utiliza la computadora.

CNTLM: Es un servicio de proxy intermediario entre aplicaciones y proxys NTLM.

CNTLM_CONF: Es un fichero de configuración necesario para la ejecución del servicio CNTLM
NTLM Proxy: Proxy institucional con autenticación NTLM.

II.2 DISCIPLINA DE REQUISITOS

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Para la especificación de los requisitos funcionales se utilizan historias de usuario teniendo en cuenta que el escenario elegido es el número 4. Para la identificación de los requisitos funcionales y no funcionales se aplica actividad de la ingeniería de requisitos definidas por Pressman (2010)

II.3 EDUCCIÓN DE REQUISITOS

En la educción de requisitos se tomaron como fuentes los sistemas homólogos analizados en el capítulo anterior, la entrevista realizada al Ing. Aldy León García, así como la investigación empírica. La información recopilada se transformó en requisitos apropiados para el diseño de la herramienta para configurar CNTLM en NOVA. Los requisitos obtenidos se han clasificado en funcionales y no funcionales.

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que prestará la herramienta, en la forma en que reaccionará a determinados insumos. Pueden ser interacciones con usuarios, sistemas, respuestas automáticas, procesos predefinidos. En algunos casos, los requisitos funcionales de los sistemas también establecen explícitamente lo que el sistema no debe hacer. La prioridad de los requisitos se establece a partir de un proceso de decisión tendiente a establecer su orden de implementación a partir del factor importancia. Este criterio de priorización se refiere al proceso en que los stakeholders manifiestan para cada requerimiento en particular un criterio de prioridad (Alta, Media, Baja) según el nivel de importancia que le confieren. (Somerville,2011)

Se identificaron los siguientes requisitos funcionales:

Tabla #1 Descripción de Requisitos Funcionales.

Identificador	Requisito	Descripción	Prioridad
RF1	Controlar Servicio CNTLM	La herramienta debe controlar el estado del servicio CNTLM en el Sistema Operativo	Alta
RF2	Editar Configuración de CNTLM	La herramienta debe poder cambiar los parámetros de configuración del servicio CNTLM, en el archivo cntlm.conf	Alta
RF3	Manejar Perfiles	La herramienta debe permitir intercambiar entre uno o varios perfiles de configuración.	Media
RF4	Gestionar Configuraciones	La herramienta debe permitir crear, modificar y cargar los ficheros de configuración para CNTLM	Media
RF5	Exportar Proxy a Variables de Sistema	La herramienta debe permitir exportar hacia las variables de sistema el proxy creado por CNTLM	Media

Requisitos no funcionales

Se trata de requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad. En palabras más sencillas, no hablan de “lo que” hace el sistema, sino de “cómo” lo hace. Alternativamente, definen restricciones del sistema tales como la capacidad de los dispositivos de entrada/salida y la representación de los datos utilizados en la interfaz del sistema. (Somerville,2011)

Se identificaron los siguientes no requisitos funcionales:

Tabla #2 Descripción de los requisitos no funcionales.

Seguridad	RNF 1. Validación de datos en la aplicación, para evitar estados inconsistentes en la información. RNF 2. Almacenar información sensible de manera encriptada. RNF 3. La herramienta debe configurar el modo <i>Gateway</i> en desactivado por defecto. RNF 4. La herramienta debe enviar una notificación cada vez que el servicio CNTLM sea ejecutado en modo <i>Gateway</i> .
Usabilidad	RNF 5. La herramienta debe ser usable usuarios con poca experiencia en el manejo de aplicaciones de configuración. RNF 6. La herramienta debe mostrar el estado del servicio CNTLM en todo momento.
Implementación	RNF 7. La herramienta debe ser escalable.
Restricciones del diseño de implementación	RNF 8. La interfaz visual debe mantener el estilo empleado en la capa de personalización de NOVA.

II.3.1 Descripción de requisitos de software.

La descripción de los requisitos funcionales de la propuesta de solución se realiza mediante las historias usuario, técnica que propone el escenario 4 de la metodología AUP-UCI para encapsular los **requisitos**

funcionales del sistema. Este producto de trabajo permite describir las funcionalidades que la herramienta debe poseer, sean requisitos funcionales o no funcionales. A continuación, se muestra la historia de usuario correspondientes al requisito funcional Controlar Servicio CNTLM.

Tabla #3 Historia de Usuario Controlar Servicio CNTLM.

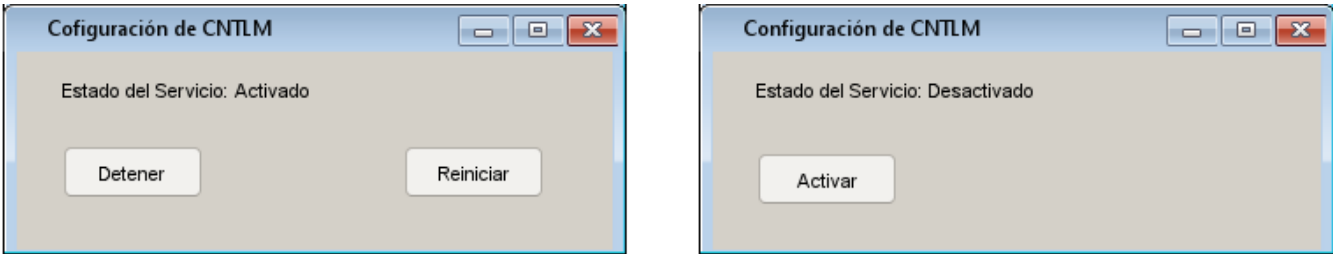
Número: 1	Nombre del requisito: Controlar Servicio CNTLM	
Programador: Daymel Feliú Delgado		Iteración Asignada: 1ra
Prioridad: Alta		Tiempo Estimado: 2 días
Riesgo en Desarrollo: N/A		
Descripción: La herramienta debe poder controlar el estado de servicio de la CNTLM mediante el <code>systemctl</code> de Linux.		
Observaciones: EL usuario debe disponer de botones visibles que permitan la verificación del estado del Servicio CNTLM y mediante la interacción con los mismos pueda cambiar dicho estado, (Activo, Detenido, Reiniciar)		
Prototipo:		
		

Tabla #4 Historia de Usuario Múltiple configuración.

Número: 2	Nombre del requisito: Múltiple configuración
Programador: Daymel Feliú Delgado	Iteración Asignada: 1ra
Prioridad: Media	Tiempo Estimado: 1 días
Riesgo en Desarrollo: N/A	

Descripción: La herramienta debe poder crear, editar, eliminar, cambiar de perfiles de configuración de CNTLM.

Observaciones: Los perfiles deben almacenarse cifrados por seguridad.

Prototipo:



II.4 DISCIPLINA DE ANÁLISIS Y DISEÑO

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis (Sánchez).

II.4.1 Modelo arquitectónico

La arquitectura no es el software operativo. Es una representación que permite analizar la efectividad del diseño para cumplir los requerimientos establecidos, considerar alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil y reducir los riesgos asociados con la construcción del software (Pressman, 2010).

La arquitectura seleccionada para la propuesta de solución es “Arquitectura Tubos y Filtros”:

“El patrón de arquitectura Tuberías y Filtros proporciona una estructura para los sistemas de flujo de proceso de datos. Cada paso del proceso se encapsula en un componente de filtro. Los datos se pasan a través de tubos entre filtros adyacentes. Recombinar filtros le permite construir familias de sistemas relacionados “(Rivera,2010).

Al CNTLM ser una aplicación de terceros está sujeta a modificaciones que pudieran llegar a generar incompatibilidades con la herramienta a desarrollar, por lo que se necesita que la herramienta propuesta en esta investigación, sea fácil de mantener y modificar, para ser adaptada a posibles nuevas necesidades; al estar encapsulados en filtros cada componente es reemplazable e independiente, por lo que realizar modificaciones no representará una dificultad grave.

La arquitectura de Tuberías y Filtros puede tener los siguientes elementos:

Bomba (Entrada): También llamado productor es el origen de datos. Pueden ser archivos, bases de datos, o dispositivos de entrada.

Tubo: Es el conector que pasa los datos de un filtro al siguiente. Se trata de un flujo unidireccional de datos, que suele ejecutarse por un buffer de datos para almacenar todos los datos; hasta que el siguiente filtro tiene tiempo para procesarlo.

Filtro: Es el encargado de filtrar o transformar los datos que recibe a través de las tuberías.

Sumidero (Salida): Se conoce también como consumidor es el objetivo de los datos, puede ser a archivos, bases de datos o dispositivos de salida.

La figura siguiente es una ilustración de la arquitectura de Tuberías y Filtros, aplicada al desarrollo de la Herramienta de configuración para CNTLM en NOVA.

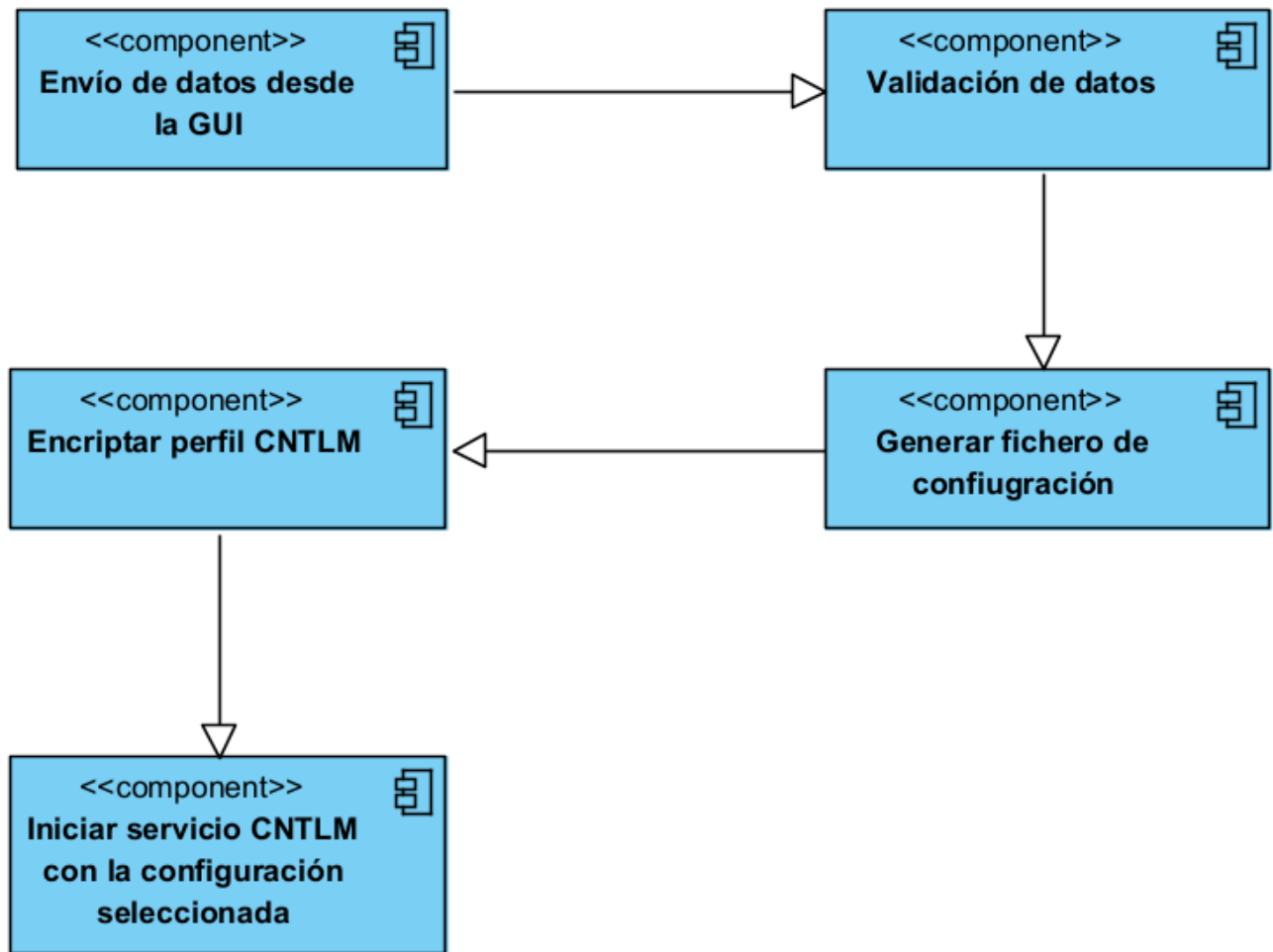


Figura #1 Arquitectura de la Herramienta de configuración Para CNTLM en NOVA

II.4.2 Diagrama de Clases

Un diagrama de clases permite representar gráficamente y de manera estática la estructura general de un sistema, mostrando cada una de las clases y sus interacciones (como herencias, asociaciones, etc.), representadas en forma de bloques, los cuales son unidos mediante líneas y arcos. Los diagramas de

clases son el pilar fundamental del modelado con UML, siendo ampliamente utilizados tanto para análisis como para diseño de sistemas y software en general.

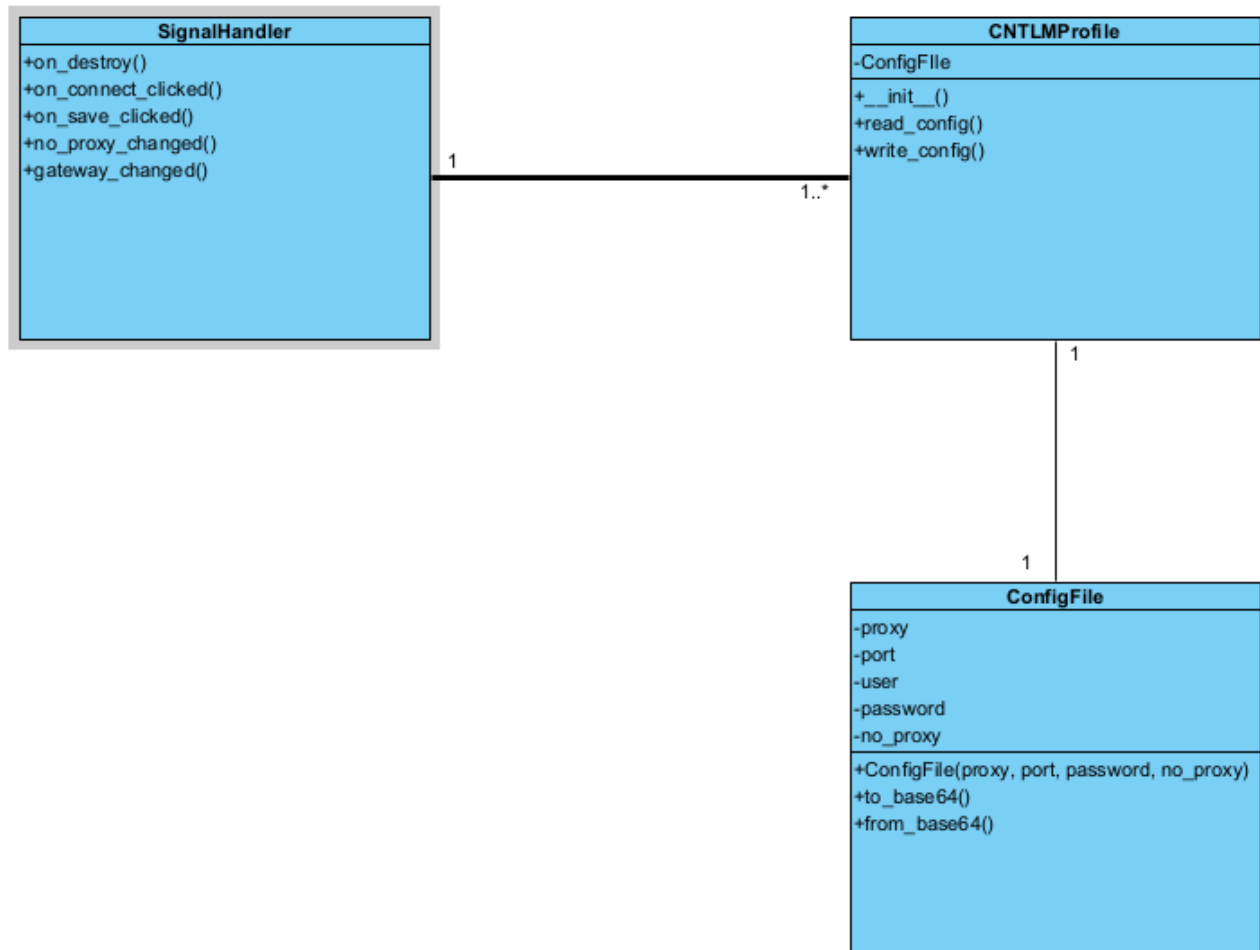


Figura #2 Diagrama de Clases

II.5 Patrones de diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico y entre “fuerzas” que afectan la manera en la que se aplica y en la que se utiliza dicho patrón.

El objetivo de cada patrón de diseño es proporcionar una descripción que permita a un diseñador determinar (Pressman, 2010):

- si el patrón es aplicable al trabajo en cuestión
- si puede volverse a usar (con lo que se ahorra tiempo de diseño)

si sirve como guía para desarrollar un patrón distinto en funciones o estructura.

Los siguientes patrones son empleados en el desarrollo de la herramienta:

Patrones de asignación de responsabilidad general:

Patrones de asignación de responsabilidad general (GRASP por sus siglas en inglés), son una serie de patrones que describen los principios fundamentales de la asignación de responsabilidades a objetos y son considerados una serie de buenas prácticas en el diseño de software (Prácticas de software 2016).

Experto:

Experto es un patrón que suele utilizarse en el diseño orientado a objetos. Este patrón sugiere asignar responsabilidad al objeto que posea la información necesaria para desempeñarla. Con la utilización se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener. Este patrón se emplea en la clase SignalHandler, esta se especializa en el manejo de las señales enviadas por la interfaz GTK, al usuario interactuar con la misma, cuentan entre sus atributos con toda la información necesaria para tales efectos.

Alta cohesión:

Alta cohesión asigna responsabilidades de manera tal que la cohesión siga siendo alta, o sea que las funcionalidades de las clases estén altamente relacionadas de forma tal que exista una colaboración entre ellas para compartir el esfuerzo y no caiga todo el peso sobre una única clase. Usar este patrón simplifica el mantenimiento y favorece el bajo acoplamiento.

Bajo acoplamiento: Plantea que se debe poder reutilizar las funcionalidades de las distintas clases, con un nivel de dependencia mínima. Este patrón es de gran utilidad pues facilita hacer cambios a partes del código sin repercusión en otras funciones de la aplicación, en la aplicación se mantiene un número bajo de relaciones entre clases, como generalmente relaciones entre 1 o 2 clases.

Patrones GOF:

Gang-of-Four (GoF), también conocidos como la 'pandilla de los cuatro' son patrones de diseño utilizados en situaciones muy frecuentes debido a que se basan en la experiencia acumulada al resolver problemas reiterativos. Además, favorecen la reutilización del código (Prácticas de software 2016). A continuación, se describe el patrón GoF utilizado en la solución propuesta:

Singleton:

El patrón de diseño Singleton recibe su nombre debido a que solo se puede tener una única instancia para toda la aplicación de una determinada clase. La intención de este patrón es garantizar que solamente pueda existir una única instancia de una determinada clase y que exista una referencia global en toda la aplicación. La clase CNTLM_CONF contiene la única configuración activa de CNTLM, pues en cualquier método donde se necesite acceder a la configuración de CNTLM se necesita que se devuelva la configuración seleccionada o activa.

Figura #3 Ejemplo de Patrón Singleton en Python:

```
1 class CNTLM(object):
2
3     class __CNTLM:
4         def __init__(self):
5             self.user = None
6             self.password = None
7             self.proxy = None
8             self.port = None
9             self.gateway = None
10            self.gateway_config = None
11        def __str__(self):
12            return `self` + ' ' + self.user
13
14        instance = None
15
16        def __new__(cls):
17            if not CNTLM.instance:
18                CNTLM.instance = CNTLM.__CNTLM()
19            return CNTLM.instance
20
21        def __getattr__(self, user):
22            return getattr(self.instance, user,password,proxy,port,gateway,gateway_config)
23
24        def __setattr__(self, , user,password,proxy,port,gateway,gateway_config, value):
25            return setattr(self.instance, user,password,proxy,port,gateway,gateway_config, value)
```

CONCLUSIONES DEL CAPÍTULO

La identificación de 5 requisitos funcionales y 7 no funcionales, permite definir con claridad las funcionalidades del sistema, así como uso de los patrones de diseño facilita el desarrollo de software y posibilita escribir código de calidad, así como su mantenibilidad. Queda definida la arquitectura de desarrollo de software Tuberías y Filtros, así como Python como lenguaje de programación junto a herramientas tecnológicas como GTK, a emplear durante el proceso de desarrollo.

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se desarrolla una descripción de la implementación y se abordan los temas relacionados con el flujo de trabajo correspondiente a las pruebas. Además, se especifican las pruebas que se le realizaron al módulo para la Herramienta de Configuración CNTLM para NOVA, para validar una correcta implementación de los requisitos del sistema y ofrecer un producto de calidad.

Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código el cual refleja un estilo armonioso, como si un único programador hubiese escrito todo el código de una sola vez. Cuando el proyecto de software incorpora código fuente previo, o cuando realiza el mantenimiento de un sistema de software el estándar de codificación debería establecer cómo operar con la base de código existente (Sommerville, 2007).

III.1 Implementación del sistema:

Luego de los resultados del análisis y diseño, se comienza la implementación de la herramienta. A continuación, se describen los estándares de codificación que se utilizan en el desarrollo de la herramienta para la configuración de CNTLM en NOVA.

Para la codificación de la propuesta de solución se utiliza el estándar de codificación para Python basado en la guía de estilo del código Python por Guido Van Rossum y Barry Warsaw. En este documento se listan convenciones utilizadas en el código Python comprendido en la librería estándar de la distribución principal de Python.

Indentación:

- Usa cuatro espacios por cada nivel de codificación.
- Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).

- No se mezclarán tabuladores y espacios en la codificación. El método de indentación más popular en Python es con espacios. El segundo más popular con tabulaciones, sin mezclar unos con otros. Cualquier código indentado con una mezcla de espacios y tabulaciones debe ser convertido a espacios exclusivamente.

Máxima longitud de las líneas:

- Todas las líneas deben estar limitadas a un máximo de 79 caracteres.
- Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas.
- En cualquier circunstancia se puede utilizar el caracter “\” para cortar líneas largas.

Líneas en blanco:

- Utilizar la codificación UTF-8.
- Se pueden incluir caracteres que no correspondan a esta codificación utilizando “\x”, “\u”, “\U” para cadenas (strings).

Importación:

- Las importaciones deben estar en líneas separadas.
- Las importaciones siempre se colocan al comienzo del archivo, simplemente luego de cualquier comentario o documentación del módulo, y antes de globales y constantes.
- Deben quedar agrupadas de la siguiente forma:
 1. Importaciones de la librería estándar.
 2. Importaciones terceras relacionadas.
 3. Importaciones locales de la aplicación/librerías.
- Cada grupo de importaciones debe de estar separado por una línea en blanco.

Espacios en blanco en expresiones y sentencias:

- Evitar usar espacios en blanco en las siguientes situaciones:
 1. Inmediatamente dentro de paréntesis, corchetes o llaves.

2. Inmediatamente antes de una coma, un punto y coma o dos puntos.
3. Inmediatamente antes de un corchete que empieza una indexación.
4. Más de un espacio alrededor de un operador de asignación u otro para alinearlo con otro.

- Deben rodearse de espacios en blanco los siguientes operadores:

1. **Asignación** (“=”).

2. **Asignación de aumentación** (“+=”, “-=”, “*=”, “/=”).

3. **Comparación** (“==”, “”, “>=”, “<=”, “! =”, “<>”, “in”, “not in”, “is not”).

4. **Expresiones lógicas.**

- Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad.
- No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.

Comentarios

- Los comentarios deben ser oraciones completas.
- Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula.
- Si un comentario es corto el punto final puede omitirse.

Convecciones de nombramiento

- Nunca se deben utilizar como simples caracteres para nombres de variables los caracteres ele minúscula “l”, o mayúscula “O”, ele mayúscula “L” ya que en algunas fuentes son indistinguibles de los números uno (1) y cero (0). • Los módulos deben tener un nombre corto y en minúscula.
- Los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúscula).

- Los nombres de las funciones deben estar escritos en minúscula separando las palabras con un guion bajo (_).
- Las constantes deben quedar escritas con letras mayúsculas separando las palabras con un guion bajo (_)

La siguiente figura representa un ejemplo donde se muestra la aplicación de algunos estándares de codificación.

```

class SignalHandler:
    def on_destroy(self, *args):
        Gtk.main_quit()

    def on_connect_clicked1(self, button, parent=None,):
        proxy_ip = builder.get_object('ip1').get_text()
        port = builder.get_object('port1').get_text()
        username = builder.get_object('username1').get_text()
        password = builder.get_object('password1').get_text()
        localPort1 = builder.get_object('localPort1').get_text()
        domain = builder.get_object('domain1').get_text()
        allowedIP1 = builder.get_object('allowedIP1').get_text()
        denyIP1 = builder.get_object('denyIP1').get_text()
        no_proxy = builder.get_object('no_proxy1').get_text()

        write_config(1, username, password, domain, proxy_ip, port,
                    builder.get_object('np1').get_active(), no_proxy,
                    builder.get_object('gateway1').get_active(), localPort1,
                    allowedIP1, denyIP1)
        print(os.getcwd())
        service_controller.start_cntlm(os.getcwd() + '/cntlm1.conf')

    def on_save1_clicked(self, button):
        proxy_ip = builder.get_object('ip1').get_text()
        port = builder.get_object('port1').get_text()
        username = builder.get_object('username1').get_text()
        password = builder.get_object('password1').get_text()
        localPort1 = builder.get_object('localPort1').get_text()
        domain = builder.get_object('domain1').get_text()
        allowedIP1 = builder.get_object('allowedIP1').get_text()
        denyIP1 = builder.get_object('denyIP1').get_text()
        no_proxy = builder.get_object('no_proxy1').get_text()

        write_config(1, username, password, domain, proxy_ip, port,
                    builder.get_object('np1').get_active(), no_proxy,
                    builder.get_object('gateway1').get_active(), localPort1,allowedIP1, denyIP1)

    def no_proxy_changed1(self, checkbutton):
        if checkbutton.get_active():
            builder.get_object('no_proxy1').set_sensitive(True)
        else:
            builder.get_object('no_proxy1').set_sensitive(False)

```

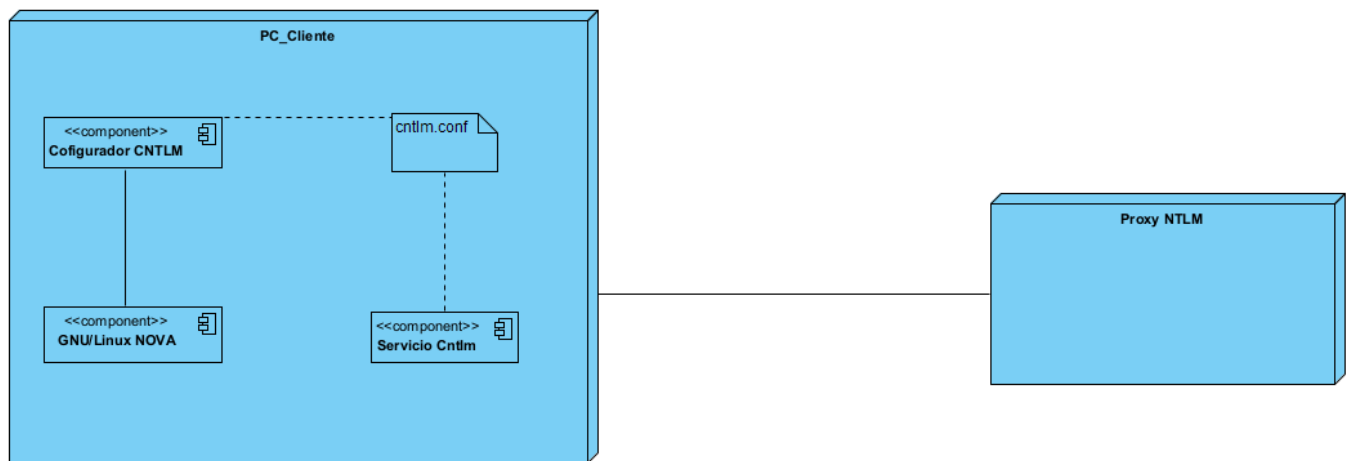
CapWords

Tabuladores

Espacios cada lado

III.2 DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue es una estructura que muestra la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos de despliegue. En él están representados nodos que corresponden con los dispositivos de hardware o con algún entorno de ejecución de software. Estos nodos pueden ser conectados a través de vías de comunicación para crear sistemas en red de complejidad arbitraria (Larman, 2010). En la figura siguiente se presenta el diagrama de despliegue elaborado para la propuesta de solución:



A continuación, se describen los nodos, componentes y protocolos de comunicación representados en el diagrama anterior.

Descripción de los nodos:

PC_Cliente: estación de trabajo desde la cual se gestiona el servicio CNTLM desde la aplicación Configurator CNTLM.

Descripción de los componentes:

Nova Escritorio: distribución cubana de GNU/Linux Nova, que se encuentra como sistema operativo en las estaciones de trabajo (computadoras).

Servicio CNTLM: Servicio CNTLM que corre sobre Nova Escritorio para proveer autenticación en proxys NTLM.

Configurador NTLM: Aplicación que corre sobre Nova Escritorio para gestionar los perfiles de configuración de CNTLM.

Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia, el desempeño de un software, involucra las operaciones del sistema bajo condiciones controladas y evalúa los resultados. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad (PRESSMAN, 2010).

Una vez que se implementa el código del sistema es necesario realizar pruebas para encontrar y enmendar la mayor cantidad de errores antes de entregarlo al cliente. El objetivo de este proceso es diseñar casos de pruebas que tengan una alta probabilidad de encontrar errores. Para alcanzar este objetivo existen técnicas de pruebas de software, que contienen directrices sistemáticas para comprobar la lógica interna y de interfaces de los componentes del sistema, así como los dominios de entrada y salida para identificar errores funcionales y de desempeño. A continuación, se describen los tipos de pruebas de software aplicadas, así como los métodos y técnicas empleadas para la evaluación de la herramienta de configuración de CNTLM para NOVA.

Tipos de pruebas de software

Técnicas de Prueba:

Pruebas de aceptación:

Visualmente resulta poco probable que un desarrollador de software prevea como utilizara el cliente final el programa. Las instrucciones para usarlo pueden malinterpretarse; regularmente pueden usarse combinaciones extrañas de datos; la salida que parecía clara a quien realizó la prueba puede ser ininteligible para un usuario. Debido a esto cuando se construye un software a la medida para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al mismo, validar todos los requerimientos y se centran en las características y funcionalidades generales que son visibles y revisables para garantizar que el sistema cumpla con lo que el cliente espera de él. Las pruebas de aceptación se

derivan de las Historias de usuario que se han implementado como parte de la liberación del software (Pressman, 2010)

Caso de prueba de aceptación	
Código: HU1_P1	Historia de usuario: 1
Nombre: Controlar Servicio CNTLM	
Descripción: La herramienta debe poder controlar el estado de servicio de la CNTLM mediante el <code>systemctl</code> de Linux.	
Condiciones de ejecución: La computadora debe tener CNTLM instalado y un archivo de configuración válido.	
Pasos de ejecución: <ol style="list-style-type: none">1. El usuario introduce la configuración de CNTLM.2. Presiona el botón iniciar.3. El sistema ejecuta el servicio CNTLM.	
Resultados esperados: El sistema inicia el servicio CNTLM con la configuración seleccionada.	
Evaluación de la prueba: Satisfactoria.	

Caso de prueba de aceptación	
Código: HU2_P1	Historia de usuario: 2
Nombre: Múltiple configuración	
Descripción: La herramienta debe poder crear perfiles de configuración de CNTLM.	
Condiciones de ejecución: El servicio CNTLM debe estar detenido.	
Pasos de ejecución: <ol style="list-style-type: none">1. El usuario debe introducir los parámetros de configuración: proxy, puerto, usuario, contraseña y puerto de escucha.2. La herramienta valida los datos introducidos.3. La herramienta cifra los datos introducidos.4. La herramienta Reemplaza o crea a el fichero de configuración según sea necesario.5. Modificar el expediente laboral del trabajador. Seleccionar la opción Guardar.	
Resultados esperados: El sistema guarda un archivo <code>.conf</code> con la configuración introducida.	
Evaluación de la prueba: Satisfactoria	

Pruebas de Integración

La prueba de integración es una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera independiente y construir una estructura de programa que determine el diseño (PRESSMAN, 2010) Es una forma de verificar la correcta interrelación de los distintos componentes del sistema, en el caso de la solución desarrollada es la verificación de una correcta interoperabilidad entre la herramienta desarrollada, CNTLM, aplicaciones y NOVA.

Teniendo en cuenta que la herramienta desarrollada debe integrarse como aplicación de sistema. Se emplea una estrategia de integración ascendente, donde los componentes se integran de abajo hacia arriba. En esta prueba de integración ascendente se realiza la prueba de regresión que permite ejecutar nuevamente el mismo subconjunto de pruebas que ya se ha aplicado para asegurar que los cambios no han propagado efectos colaterales indeseables.

Resultados de la prueba de integración:

Las pruebas de integración realizadas permitieron identificar una falla en el control del servicio CNTLM. Luego de solucionar esta no conformidad, la herramienta de configuración de CNTLM para Nova se integró correctamente con el sistema operativo y el resto de aplicaciones.

CONCLUSIONES FINALES

Para concluir este Trabajo de Diploma se desarrolló una Herramienta para la configuración de CNTLM en NOVA, para ello:

- El análisis de los principales conceptos asociados a la investigación permitió determinar el alcance de la aplicación.
- El estudio de las herramientas homólogas permitió identificar funcionalidades que no deben faltar en la aplicación.
- El estudio de metodologías y tecnologías contribuyó a seleccionar la más acorde para obtener una herramienta con la calidad requerida.
- Vincular distintas áreas del conocimiento como lo son ingeniería y programación fue necesario para el diseño e implementación de la propuesta de solución con la calidad requerida.
- Con el diseño de la estrategia de pruebas se determinó el cronograma a seguir para realizar las pruebas y cuáles se ejecutarán.

RECOMENDACIONES

- Integrar la herramienta desarrollada a futuras versiones de NOVA.

BIBLIOGRAFÍA

- Blog Desde Linux.* (s.f.). Obtenido de Blog Desde Linux: <https://blog.desdelinux.net/repositorios-de-distribuciones-gnu-linux/>
- Corporateproxyhelper.* (s.f.). Obtenido de Corporateproxyhelper: <https://www.corporateproxyhelper.com/>
- Covaci, E. (s.f.). *Github.* Obtenido de Github: <https://github.com/ecovaci/winfoo>
- Glade - A User Interface Designer.* (s.f.). Obtenido de Glade - A User Interface Designer: <https://glade.gnome.org/>
- GTK.* (s.f.). Obtenido de GTK: <https://www.gtk.org/docs/>
- IBM MQ.* (s.f.). Obtenido de IBM MQ: <https://www.ibm.com/docs/es/ibm-mq/7.5?topic=ssfskj-7-5-0-com-ibm-mq-sec-doc-q009740--htm>
- Microsoft Docs.* (s.f.). Obtenido de Microsoft Docs: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-nntp/7a61abbf-9909-4a54-8d9e-984cc0a36b55
- Pressman. (2010). *Ingeniería De Software Un enfoque práctico (7.a ed.). MCGRAW HILL EDUCATION.* MCGRAW HILL EDUCATION.
- RootSudo.* (s.f.). Obtenido de RootSudo: <https://help.ubuntu.com/community/RootSudo>
- Torres, V. (2015). *HumanOS.* Obtenido de HumanOS: <https://humanos.uci.cu/2015/11/04/compartetusoftware-aplicacion-para-monitorizar-el-estado-de-la-cuenta-cntlm/>
- Ubuntu Manuals.* (2010). Obtenido de Ubuntu Manuals: <https://manpages.ubuntu.com/manpages/precise/en/man1/cntlm.1.html>
- UbuntuDocumentation.* (s.f.). Obtenido de <https://help.ubuntu.com/community/RootSudo>
- Visual Studio Code.* (s.f.). Obtenido de Visual Studio Code: <https://code.visualstudio.com/docs/supporting/faq>
- Viswanathan, G. (s.f.). *Github.* Obtenido de Github: <https://github.com/genotrance/px>

ANEXOS

Anexo #1:

Entrevista al Ing. Aldy León García:

Objetivo: Conocer, analizar y obtener información del proceso de configuración de sistemas intermediarios entre proxy en Nova.

4. ¿Nova trae alguna herramienta preinstalada para la gestión de proxys intermediarios?
5. ¿Se emplea alguna aplicación de terceros para ello?
 - 2.1 ¿Principales funcionalidades de interés?
 - 2.2 ¿Inconvenientes?