Universidad de las Ciencias Informáticas

Facultad 1

Herramienta de autenticación a un controlador de dominio desde el Centro de Control de la distribución GNU/Linux Nova

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

Autor: Diwel García Valle

Tutores: Ing. Inst. Ivaniet Díaz Romeu

Ing. Haniel Cáceres Navarro

La Habana, junio de 2018

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis que tiene por título: Herramienta de autenticación a un controlador de dominio desde el Centro de Control de la distribución GNU/Linux Nova; y reconozco a la Universidad de Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firman la presente a los ___ días del mes de _____ del año _____.

Diwel García Valle

Firma del Autor

Ing. Inst. Ivaniet Díaz Romeu

Firma del Tutor

Ing. Haniel Cáceres Navarro

Firma del Tutor

DATOS DE CONTACTO

Autor:

Diwel García Valle

dvalle@estudiantes.uci.cu

Tutores:

Ing. Inst. Ivaniet Díaz Romeu

Especialista Centro de *Software* Libre

ivaniet@uci.cu

Ing. Haniel Cáceres Navarro

Especialista Centro de *Software* Libre

hcaceres@uci.cu

AGRADECIMIENTOS

A los linuxeros del Centro de *Software* libre por el apoyo brindado.

A los profes, que sin mirar atrás me apoyaron y me motivaron a seguir adelante, hasta que se concluyó con el desarrollo del trabajo de diploma sin dejar a un lado su labor profesional y su buena práctica como educadores.

A mis compañeros por sus lecciones de compañerismo y humanidad.

A todas mis amistades que durante estos cinco años siempre nos ayudamos y nos mantuvimos unidos para rebasar los retos de la carrera.

A mi papá que me ayudó mucho a mí, a mi mamá, a HIJO y a mi abuela para que pudiera lograr todas mis metas.

A toda mi familia que siempre me apoyó en todo, mis hermanos, mis primos, mis tíos, todos aportaron a mi formación como mejor persona y profesional.

A todos los que de una forma u otra contribuyeron en mi formación profesional, así como en el desarrollo de esta investigación.

DEDICATORIA

A mi mamá por haberme dado la vida y junto a ella toda mi familia, sin dejar de mencionar a mi abuelita por haberme apoyado hasta el último segundo para llegar hasta aquí. Sin el apoyo de estas personas nada de esto hubiera sido posible. No por ser el último es el menos importante mi hijo, el cual me dio las fuerzas suficientes para luchar hasta el final siempre mirado la luz que me desprende para que obtuviera las fuerzas necesarias para cumplir el objetivo.

RESUMEN

En la actualidad la distribución cubana de GNU/Linux Nova hace uso de herramientas de terceros para la autenticación a un controlador de dominio, de las cuales no se posee el código fuente, impidiendo su adaptabilidad a las características de Nova. La presente investigación tiene como objetivo fundamental, desarrollar una herramienta informática que permita la autenticación de la distribución cubana GNU/Linux Nova a un controlador de dominio desde su centro de control. La propuesta de solución estuvo guiada por la metodología de desarrollo de *software* Variación AUP-UCI, como lenguaje de programación se seleccionó Ansi C, como protocolo de autenticación se utilizó LDAP, Visual Paradigm como herramienta CASE. Se identificaron y describieron las funcionalidades requeridas, se definió la arquitectura de *software.* Se obtuvo como resultado el completo desarrollo de la aplicación y su funcionamiento fue evaluado mediante la aplicación de pruebas. El desarrollo de dicha herramienta, permitió informatizar el proceso de autenticación a un controlador de dominio, de tal forma que se controle el acceso de los usuarios y las aplicaciones a los recursos existentes en la red, a través de una interfaz gráfica.

Palabras clave: autenticación, controlador de dominio.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

Introducción

Con el desarrollo y evolución de las Tecnologías de la Informática y las Comunicaciones (TIC), se hace cada vez más necesario la existencia de sistemas de seguridad que permitan al administrador de red brindar un servicio de mayor calidad, con las prestaciones necesarias y que por demás permita el trabajo diario. Siendo la premisa fundamental de este tipo de sistemas, la protección de la información, el control de acceso y los privilegios de usuario en la infraestructura, además de ofrecer al administrador un lugar centralizado para realizar de forma remota las tareas de mantenimiento de la red. Por ello los controladores de dominio constituyen una piedra angular en cualquier infraestructura de red (Fajardo, 2014).

Un controlador de dominio es un servidor de dominio donde se ejecuta el servicio Directorio Activo y que almacena una copia del directorio de dominio. Además, administra los procesos de inicio de sesión, autenticación y búsquedas en directorios de usuarios. Un controlador de dominio solo puede admitir un dominio, por otro lado, un dominio debe disponer de más de un controlador para proporcionar la disponibilidad y la tolerancia a fallos adecuadas (Windows Corporation, 2014).

Dado el nivel de desarrollo alcanzado por el país, se ha logrado avanzar en diferentes sectores. En relación con las tecnologías de la información, la creación de infraestructuras de conectividad, centros de datos, desarrollo de aplicaciones, entre otros, ha requerido el empleo de sistemas de seguridad que garanticen la protección desde la comunicación hasta la información que se procesa. Para lograr este objetivo, existen diversas implementaciones, siendo la más divulgada la propuesta de Microsoft Windows, Active Directory (Directorio Activo) (Fajardo, 2014).

El Directorio Activo, es un servicio de red, que almacena información acerca de los recursos existentes en la red, y controla el acceso de los usuarios y las aplicaciones, a dichos recursos. De este modo, se convierte en un medio de organizar, administrar y controlar centralizadamente, el acceso a los recursos de la red (García, 2011).

Existen gran cantidad de instituciones que para su trabajo diario requieren de esta implementación, pero debido a que requieren de una licencia pagada para su uso, se hace más difícil adquirirlas. En Cuba se ha ido introduciendo poco a poco el uso de los controladores de dominio basado en la integración de herramientas de código abierto, debido a que su licencia es gratuita, es invulnerable a los virus y son menos susceptible a catástrofes. En nuestro país, con el uso de estas herramientas, se ha logrado alcanzar un nivel muy alto de seguridad, adecuados muy a tono con los que el desarrollo de las TIC hoy demanda a nivel internacional (Fajardo, 2014).

Por otra parte, la distribución cubana GNU/Linux Nova es una distribución desarrollada en la Universidad de las Ciencias Informáticas (UCI) por estudiantes y profesores y con la participación de miembros de otras instituciones, con razón de apoyar la migración a tecnologías de software libre y código abierto del país. Nova Escritorio es la edición estándar de GNU/Linux Nova dedicada a los ordenadores personales. Esta versión cuenta con Nova-Shell, un entorno de escritorio que se adapta a las necesidades tanto de los usuarios con experiencia en GNU/Linux así como los nuevos entusiastas acostumbrados al sistema operativo de Microsoft, permitiendo una experiencia más agradable en el proceso de migración (Esteve, 2010).

La distribución cubana GNU/Linux Nova, utiliza el entorno de escritorio Gnome[1] de forma predeterminada, así como su respectivo Centro de Control, el cual brinda acceso a diferentes funcionalidades y configuraciones del sistema. El entorno de escritorio Gnome brinda mejoras para la adaptabilidad del usuario, lo cual se ajusta a sus necesidades a pesar de no contar con una herramienta propia que facilite su autenticación en un controlador de dominio (Heinz, 2007).

Una de las responsabilidades de los controladores de dominio es la autenticación, proceso mediante el cual se garantiza o deniega a un usuario el acceso a recursos compartidos o a otro ordenador de la red, normalmente a través del uso de una contraseña. Cada controlador de dominio usa un security account manager (SAM), o NTDS en Windows 2003 Server (que es la forma promovida de la SAM, al pasar como controlador de dominio), para mantener una lista de pares de nombre de usuario y contraseña. El controlador de dominio entonces crea un repositorio centralizado de contraseñas, que están enlazados a los nombres de usuarios (una clave por usuario), lo cual es más eficiente que mantener en cada máquina cliente centenares de claves para cada recurso de red disponible (Fajardo, 2014).

A través de la observación por parte del autor y encuestas realizadas a especialistas del Centro de Software Libre (CESOL), perteneciente a la UCI se constató que la distribución cubana de GNU/Linux Nova no posee una herramienta nativa para la autenticación a un controlador de dominio, lo que provoca que no se controle el acceso de los usuarios y las aplicaciones, a los recursos existentes en la red. Para ello, hace uso de herramientas de tercero de las cuales no se posee el código fuente, esto impide su adaptabilidad a las características de Nova. Es por ello que la presente investigación plantea la necesidad de desarrollar una herramienta para el centro de control de Nova que permita la autenticación a un controlador de dominio.

[1] Gnome es un entorno gráfico (escritorio de trabajo) para Linux y sistemas tipo Unix, que permite a los usuarios usar y configurar sus ordenadores de una forma sencilla.

Se define como **problema de investigación:** ¿Cómo realizar la autenticación de la distribución cubana GNU/Linux Nova a controladores de dominio desde su centro de control?

Para dar solución al problema antes planteado, se establece como **objetivo general:** Desarrollar una herramienta informática para el centro de control de la distribución cubana GNU/Linux Nova que permita la autenticación a controladores de dominio.

Se define como **objeto de estudio**: los mecanismos de autenticación a controladores de dominio; enmarcado en el **campo de acción**: los mecanismos de autenticación a controladores de dominio en la distribución cubana de GNU/Linux Nova.

Para dar cumplimiento al objetivo general se consideran los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación sobre la autenticación a controladores de dominio desde las distribuciones GNU/Linux, para sentar las bases del desarrollo de la propuesta de solución.
2. Diseñar una herramienta informática que permita la autenticación a un controlador de dominio, desde el centro de control de Nova.
3. Implementar una herramienta informática que permita la autenticación a un controlador de dominio desde el centro de control de Nova.
4. Evaluar la herramienta informática para la autenticación a un controlador de dominio desde el centro de control de Nova.

Para cumplir con los objetivos específicos se proponen las siguientes **tareas de la investigación:**

1. Elaboración del marco teórico a partir del estado del arte de los mecanismos para la autenticación a controladores de dominio en distribuciones de GNU/Linux, que permita analizar los conceptos relacionados a estos y su integración al centro de control de Nova.
2. Definición de la metodología, tecnologías y herramientas de desarrollo para una mejor estructura de la propuesta de solución.
3. Análisis y diseño de las funcionalidades a implementar.
4. Implementación de la solución propuesta.
5. Realización de pruebas de unidad e integración a la solución propuesta para una mejor calidad del producto.

Para llevar a cabo la investigación se hace uso de diversos **métodos científicos:**

Métodos teóricos:

- **Histórico-Lógico:** Este método permitió seguir la evolución del proceso de autenticación a controladores de dominio, lo cual ayudó a profundizar en conocimientos acerca del tema.
- **Analítico-sintético:** Este método permitió el estudio de diferentes fuentes bibliográficas para extraer los elementos más importantes que se relacionan con el proceso de autenticación a controladores de dominio desde distribuciones GNU/Linux.
- **Análisis documental:** Se utiliza en la revisión de la literatura especializada para extraer la información necesaria que permitió realizar el proceso de investigación para desarrollar una herramienta que permita autenticar GNU/Linux a controladores de dominio**.**
- **Análisis comparativo:** Se emplea para detectar similitudes, diferencias e insuficiencias en cuanto a la interpretación que brindan las herramientas existentes para autenticar GNU/Linux a controladores de dominio.

Métodos empíricos:

- **Observación:** Permitió chequear el funcionamiento actual del proceso de autenticación a controladores de dominio con las herramientas usadas para ello.
- **Entrevista:** Se realizó una entrevista al cliente para obtener información acerca de las funcionalidades que debía cumplir la herramienta. Ver anexo 1.
- **Encuesta:** Permitió obtener criterios por parte de especialistas que trabajan con la distribución cubana GNU/Linux Nova con respecto a la autenticación a controladores de dominio antes de la presente investigación. Ver anexo 2.

Este documento está compuesto por introducción, 3 Capítulos, Conclusiones, Recomendaciones, Bibliografía, Referencias Bibliográficas y Anexos. A continuación se describe el contenido de los capítulos.

Capítulo 1: Fundamentación teórica sobre la autenticación a controladores de dominio.

En este capítulo se definen los principales conceptos asociados al tema de investigación, se caracterizan las herramientas de autenticación de los sistemas operativos GNU/Linux a controladores de dominio. Asimismo, se definen la metodología, lenguajes y herramientas para el desarrollo de la propuesta de solución.

Capítulo 2: Análisis y diseño de una herramienta informática para la autenticación a un controlador de dominio desde el centro de control de Nova.

Se realiza el análisis y diseño de la propuesta de solución mediante la definición de los requisitos funcionales y no funcionales que debe cumplir el sistema, así como el trabajo con las herramientas que se utilizan para su implementación. Además, se describe el proceso ágil basado en la metodología AUP[2]-UCI.

Capítulo 3: Implementación y evaluación de la herramienta informática para la autenticación a un controlador de dominio desde el centro de control de Nova.

Este capítulo describe las tareas relacionadas con la fase de implementación del proceso de autenticación a controladores de dominio y las pruebas de *software* que se le realizan a la propuesta de solución.

[2] AUP: Proceso Unificado Ágil, del inglés Agile Unified Process.

Capítulo 1: Fundamentación teórica sobre la autenticación a controladores de dominio

En el presente capítulo se abordan los conceptos fundamentales que sustentan la presente investigación. Se realiza un estudio del proceso de autenticación para finalmente seleccionar los aspectos a tener en cuenta en la propuesta de solución, según el resultado obtenido. Posteriormente se detallan la metodología, herramientas y tecnologías que son empleadas para el desarrollo de la solución propuesta.

1.1. Servicios de Directorio

Desde la primera mitad del siglo pasado, las compañías de telecomunicaciones comenzaron a manejar el concepto de servicio de directorio que posteriormente se introdujo en las Tecnologías de la Información y las Redes de Computadoras; fue una necesidad, en aquel entonces, demandada por los servicios telefónicos para registrar las grandes cantidades de usuarios con que contaban. El tema de los Directorios de Identidades y la autenticación, ha sido identificado como básico para lograr seguridad en las Tecnologías de la Información, conformando estos, los llamados sistemas de seguridad (Fajardo, 2014).

En las redes actuales y sobre todo en aquellas que manejan información sensible, resulta básico mantener un control de acceso basado en identidades reales de los usuarios. La especialidad de "Ingeniería de Identidades" ha cobrado fuerza en los últimos años y se han desarrollado varias plataformas con este objetivo, algunas de carácter libre y otras propietarias, que en su conjunto, posibilitan la implementación de la autenticación y la autorización a los recursos informáticos a través de los sistemas de seguridad (Fajardo, 2014).

Un servicio de directorio puede ser un elemento aglutinador y base para desarrollar aplicaciones que permitan desplegar nuevos servicios basados en la cooperación entre las distintas aplicaciones y el servicio de directorio. Puede actuar como servidor de autenticación, proporcionando una contraseña única para todos los usuarios. Puede actuar Implementación de Controlador de Dominio basado en la integración de herramientas de *software* libre. Puede actuar como repositorio de información de varios servidores donde se almacenaría la información necesaria para darles acceso a los usuarios a determinada información (Vejo, 2008).

La implantación de un servicio de directorio puede jugar un papel fundamental, minimizando los costos de administración en tiempo y esfuerzo y poniendo la información pública a todos aquellos usuarios y aplicaciones que necesiten hacer uso de la misma. Son útiles en redes de gran envergadura que

sufren constantes cambios, objetos de diversos tipos entran y salen de la red, bien por separado o en grupos, la interrelación entre estos objetos puede variar en cualquier momento y su vida útil no es corta. Generalmente, un objeto participará en comunicaciones con mucha mayor frecuencia que la de cambio de su dirección, disponibilidad o ubicación física (Vejo, 2008).

1.1.1. Características de los Directorios

Una de las principales características de los directorios, y base fundamental para su diseño, es que están optimizados para las operaciones de lectura y búsqueda de información, siendo las operaciones de inserción, borrado y actualización mucho menos eficientes. La información almacenada en el directorio está definida mediante un esquema que, al contrario que en las bases de datos, es fácilmente extensible y modificable. Esto permite modificar o incorporar atributos a los objetos del directorio sin que esto implique reestructuraciones importantes en su estructura (Pérez, 2008).
El servicio de directorios utiliza un modelo que permite almacenar la información de forma distribuida donde diversas partes del directorio (subárboles) son gestionadas por diferentes servicios de directorio. El modelo también facilita la replicación de información (servidores secundarios) de forma que podemos tener todo o parte del directorio replicado para ganar en eficiencia y escalabilidad. Para alcanzar los altos niveles de eficiencia (principalmente para las lecturas), este esquema normalmente se implementa con características de débil consistencia, permitiéndose inconsistencias temporales mientras se propaga, por ejemplo, una actualización o inserción de registro (Pérez, 2008).

Ejemplos de directorios

Domain Name Sistem (DNS, traducido al español como: Sistema de nombre de dominio), comúnmente utilizado para relacionar nombres de dominio a direcciones IP, es posiblemente el servicio de directorio más extendido en internet. Se trata de una base de datos distribuida y jerárquica que gestiona información relacionada con nombres de dominio (Pérez, 2008).

Network Information Service (NIS, traducido al español como: Servicio de información de red), nos aporta el servicio de configuración de red, originalmente pensado para redes Unix, que permite centralizar información referente a equipos y usuarios de la red. X.500 fue uno de los primeros servicios de directorio propuestos de carácter general. Su protocolo especifica un modelo de conexión de servicios de directorio locales para formar un directorio global distribuido en internet. En la actualidad LDAP[3], que surgió como superación de X.500, es el servicio de directorio de carácter general más extendido. Sus dos implementaciones más extendidas, *OpenLDAP* en los entornos Unix

[3] LDAP Protocolo Ligero de Acceso a Directorios (del inglés Lightweight Directory Access Protocol).

y *Active Directory* en entornos Windows, son en la actualidad las principales herramientas para la configuración y gestión de redes de computadoras (Pérez, 2008).

1.1.2. Definición de los protocolos

Protocolo es el término que se emplea para denominar al conjunto de normas, reglas y pautas que sirven para guiar una conducta o acción. Red, por su parte, es una clase de estructura o sistema que cuenta con un patrón determinado. El concepto de protocolo de red se utiliza en el contexto de la informática para nombrar a las normativas y los criterios que fijan cómo deben comunicarse los diversos componentes de un cierto sistema de interconexión. Esto quiere decir que, a través de este protocolo, los dispositivos que se conectan en red pueden intercambiar datos (Pérez, 2013).

El Protocolo Ligero de Acceso a Directorio (LDAP, del inglés Lightweight Directory Access Protocol) es un protocolo a nivel de aplicación que concede el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. Normalmente se utiliza para acceder a la información almacenada de usuarios, grupos y equipos de una organización (Carter, 2003).

Características de los protocolos

Es un protocolo estándar de la industria, establecido por el Grupo de Trabajo de Ingeniería de Internet (IETF, del inglés Internet Engineering Task Force), que permite a los usuarios consultar y actualizar la información en un servicio de directorio (Microsoft, 2014).

LDAP estructura la información en forma de un árbol, de forma análoga a la estructura de directorio de UNIX. La definición detallada y las especificaciones técnicas de LDAP están disponibles en las RFC[4] 2251 y 3377 respectivamente.

LDAP es el protocolo de acceso principal para Directorio Activo y especifica que un objeto sea representado por una serie de componentes de dominio, unidades organizativas y nombres comunes, creando una ruta de nomenclatura LDAP en el Directorio Activo. Existen varias nomenclaturas para identificar los objetos dentro del Directorio Activo.

Nombre Completo (DN, del inglés Distinguished Name). Es un identificador único de un objeto dentro del Directorio Activo. Identifica el dominio donde se encuentra el objeto y la ruta de acceso completa por la que se llega hasta él.

También conocido como protocolo de comunicación, el protocolo de red establece la semántica y la sintaxis del intercambio de información, algo que constituye un estándar. Las computadoras en red,

[4] RFC: Solicitud de comentarios, del inglés Request For Comments. Documento que contiene una propuesta para una nueva tecnología, información acerca del uso de tecnologías y/o recursos existentes.

de este modo, tienen que actuar de acuerdo a los parámetros y los criterios establecidos por el protocolo en cuestión para lograr comunicarse entre sí y para recuperar datos que, por algún motivo, no hayan llegado a destino. En el protocolo de red se incluyen diversas informaciones que son imprescindibles para la conexión. El protocolo indica cómo se concreta la conexión física, establece la manera en que debe comenzar y terminar la comunicación, determina cómo actuar ante datos corrompidos, protege la información ante el ataque de intrusos, señala el eventual cierre de la transmisión (Pérez, 2013).

1.1.3. Valoración del estudio realizado

El estudio realizado de los servicios de directorios, permitió entender su funcionamiento en cuanto a la centralización y gestión de los recursos, así como la autenticación de usuarios de una red a través del protocolo LDAP. Dicho estudio sentará las bases para el desarrollo de la investigación.

1.2. Controlador de dominio

Controlador de dominio o Directorio Activo, es una gran base de datos con multitud de recursos compartido en la red, que permiten administrar, organizar y controlar dichos recursos desde una misma ubicación. El servicio de Directorio Activo proporciona la capacidad de establecer un único inicio de sesión y un repositorio central de información para toda su infraestructura, lo que simplifica ampliamente la administración de usuarios y equipos, proporcionando además la obtención de un acceso mejorado a los recursos en red. Es un servicio de directorio, en el cual se pueden resolver nombres de URLs o de determinados recursos. Está diseñado para funcionar perfectamente en una instalación de cualquier tamaño, desde sólo un servidor con algunos cientos de objetos, hasta múltiples servidores y millones de objetos (García, 2011).

Las cuentas de usuarios que gestiona Directorio Activo, son almacenadas en la base de datos SAM (Security Accounts Manager), pero directorio activo no sólo almacena información sobre los usuarios, sino que también mantiene información sobre servidores, estaciones de trabajo, recursos, aplicaciones, directivas de seguridad. Las redes de grandes empresas, suelen basarse en este servicio que facilita enormemente la labor del administrador. Servicios de directorio, es una base de datos distribuida que permite almacenar información relativa a los recursos de una red con el fin de facilitar su localización y administración (García, 2011).

Se compone del propio servicio de directorio junto con un servicio secundario que permite el acceso a la base de datos y admite las convenciones de denominación X.500. Se puede consultar el directorio con un nombre de usuario para obtener información como el número de teléfono o la dirección de correo electrónico de ese usuario. Los servicios de directorio también son flexibles. Permiten la

realización de consultas generalizadas para ver una lista resumida de las impresoras o servidores disponibles. Los servicios de directorio también ofrecen la ventaja de suponer un único punto de entrada para los usuarios a la red de toda la empresa. Los usuarios pueden buscar y utilizar recursos en la red sin conocer el nombre o la ubicación exacta del recurso. Igualmente, puede administrar toda la red con una vista lógica y unificada de la organización de la red y sus recursos (García, 2011).

1.2.1. Mecanismos de autenticación a controladores de dominio

Para el desarrollo de la investigación se hizo necesario el análisis de los mecanismos de autenticación a controladores de dominio para conocer las características comunes y las posibles funcionalidades a agregar a la propuesta de solución.

PBIS

El proyecto *Likewise-Open* (en su versión más reciente conocido como *Power Broker Open* o por sus siglas en inglés PBIS) es un programa de código abierto que permite autenticar plataformas distintas a Microsoft Windows con controladores de dominio, fue lanzado en diciembre de 2007. Está disponible bajo una licencia GPL / LGPL v2 (Likewise, 2008).

PBIS es compatible con Linux, Unix y Mac OS X, por lo que puede centralizar la gestión de todos los equipos, autenticar usuarios y autorizar el acceso a los recursos. PBIS es libre de descargar y utilizar de acuerdo con los términos de la Licencia Pública General GNU. El agente Likewise al estar instalado en Linux o UNIX, se autentica con el sistema operativo e implementa la asignación para cualquier aplicación que utilice el servicio de nombres o PAM[5]. Un ejemplo de una aplicación tipo PAM es el proceso de inicio de sesión (Likewise, 2008).

PBIS actúa como un cliente de Kerberos 5[6] para la autenticación y como un cliente LDAP para la autorización (Likewise, 2008). El agente también incluye muchísimas bibliotecas o librerías que utiliza para configurar todos los servicios que necesita para su funcionamiento. Le facilita el trabajo al usuario al no tener que preocuparse por hacer todas esas configuraciones de forma manual.

SSSD

SSSD[7] es un demonio[8] . Su función principal es proporcionar acceso a la identidad y la autenticación

[5] PAM módulo de autenticación conectable(PAM del inglés *Pluggable Authentication Modules*).

[6] Kerberos 5 es un protocolo de autenticación de redes de ordenador que permite a dos ordenadores en una red insegura demostrar su identidad mutuamente de manera segura.

[7] Demonio de servicios de seguridad del sistema (SSSD del inglés System Security Services Daemon).

[8] En sistemas UNIX se conoce como demonio o daemon (Disk And Execution Monitor) a un proceso que se ejecuta en segundo plano del sistema operativo, se ejecuta en todo momento y no posee interacción directa con el usuario, también se

de recurso remoto a través de un marco común que puede proporcionar soporte de almacenamiento en caché y fuera de línea para el sistema. Se proporciona una interfaz de *Name Service Switch* (NSS, traducido al español como: Servicio de nombre Switch) y PAM para el sistema y también una mejor base de datos para almacenar los usuarios locales, así como de datos extendidos de usuario (Petersen, 2004).

SSSD también es extensible y se puede configurar para utilizar nuevas fuentes y mecanismos de autenticación de identidad en caso de que surjan. Además, es totalmente compatible con IPv6 (Petersen, 2004).

Características principales:

- Autenticación fuera de línea: Uno de los principales beneficios de SSSD es la autenticación fuera de línea. Esto resuelve el caso de los usuarios que tienen una cuenta separada corporativa y una de equipo local debido a la exigencia común de implementar una red privada virtual (VPN, del inglés: *Virtual Private Network*) (Petersen, 2004)
- Reducción de carga del servidor: El uso de SSSD también ayuda a reducir la carga de los servidores de identificación. Por ejemplo, al usar nss_ldap[9], cada aplicación cliente que necesita solicitar información sobre el usuario abre su propia conexión con el servidor LDAP. La gestión de estas conexiones múltiples puede dar lugar a una pesada carga en el mismo. En un sistema de cliente SSSD, sólo el proceso SSSD proveedor de datos es quien se comunica con el servidor LDAP, disminuyendo la carga a una conexión por cada sistema cliente (Petersen, 2004).
- Soporte para varios dominios: puede utilizar SSSD para especificar múltiples dominios del mismo tipo, lo cual, comparado con un archivo de configuración nsswitch.conf, con la que sólo se puede solicitar información de los usuarios desde un único servidor de cualquier tipo particular (LDAP, NIS, etc.), resulta superior. Con SSSD, se pueden crear múltiples dominios de la misma, o de diferentes tipos de identidad (Petersen, 2004).
- Diferenciación de los usuarios del mismo nombre: SSSD apoya la diferenciación de los usuarios del mismo nombre en diferentes dominios. Por ejemplo, se puede diferenciar al usuario kate en el dominio de ldap.example.com, del usuario kate en el dominio

le conoce enéricamente como servicio o proceso, del cual no se percibe su ejecución. Un demonio realiza una operación específica en tiempos predefinidos o en respuesta a ciertos eventos del sistema.

[9] Librería o biblioteca del protocolo LDAP, para hacer uso del servicio de nombres NSS.

ldap.myhome.com (Petersen, 2004).

La autenticación con IPA: va más allá de la autenticación fuera de línea, administración de dominio múltiple y otras características ya descritas, SSSD también está diseñado para autenticarse y mejorar la funcionalidad de los clientes IPA. En un entorno con la última versión de la API instalada, SSSD proporciona funcionalidad adicional, incluyendo soporte para actualizaciones de DNS dinámico, basado en el host de control de acceso y contraseña, migración desde un entorno LDAP (sólo en el LDAP / Kerberos 5) ambiente empleado por el IPA (Petersen, 2004).

Winbind

Winbind es un componente de la suite de programas Samba que resuelve los problemas de inicio de sesión unificados. Winbind usa una implementación UNIX de las llamdas RPC de Microsoft, PAM (Pluggable Authentication Modules), y el servico de nombres (NSS, Name Service Switch) para permitir a los usuarios de dominios NT aparecer y operar como usuarios UNIX en una máquina UNIX (Maldonado, 2017).

Winbind proporciona tres funciones separadas:

- Autenticación para credenciales de usuario (vía PAM).
- Resolución de identidad (vía NSS).
- Winbind mantiene una base de datos llamada winbind_idmap.tdb en la cual guarda las asociaciones entre UNIX UIDs / GIDs y NT SIDs. Esta asociación se usa sólo para usuario y grupos que no tienen unos UID/GID locales. Guarda los UID/GID del rango uid/gid de idmap que ha asociado al SID NT. Si se ha especificado el *backend idmap* como ldapsam: url, entonces en lugar de usar de usar la asociación local Winbind la obtendrá de la base de datos LDAP.

Winbind unifica la gestión de cuentas UNIX y Windows NT permitiendo a una máquina Unix volverse un miembro completo de un dominio NT. Una vez hecho, la máquina Unix verá a los usuarios y grupos NT como si fueran usuarios y grupos nativos permitiendo usar el dominio NT usarse de la misma forma que se usa NIS+ en entorno exclusivos UNIX (Maldonado, 2017).

El resultado final es que cuando cualquier programa en la máquina Unix solicita al sistema operativo que busque un nombre de usuario o un grupo, la consulta se resuelve preguntando al controlador de dominio NT del dominio correspondiente. Como winbind enlaza con el sistema operativo a bajo nivel (vía módulo de resolución NSS de la biblioteca C), la redirección al controlador de dominio NT es completamente transparente (Maldonado, 2017).

Los usuarios de la máquina Unix pueden utilizar los nombres de usuarios y grupos NT como si fueran nombres *nativos*. Se pueden cambiar propietarios de ficheros para que sean propiedad de usuarios de usuarios del dominio NT o incluso iniciar una sesión en la máquina Unix y lanzar una sesión X-Window como miembro del dominio (Maldonado, 2017).

La única indicación obvia de que se está usando Winbind que los nombres de usuarios y grupos tienen la forma *DOMAIN\user and DOMAIN\group*. Esto es necesario para permitir a Winbind determinar a qué controlador de dominio hay que re-direccionar para la búsqueda particular y qué dominios de confianza se están referenciando (Maldonado, 2017).

Adicionalmente, Winbind proporciona un servicio de autenticación que engancha con el sistema PAM (*Pluggable Authentication Modules*) (PAM) para proporcionar autenticación vía dominio NT a cualquier aplicación con soporte PAM. Esta capacidad resuelve el problema de sincronizar las contraseñas entre sistema ya que todas las contraseñas se almacenan en una sola ubicación, en el controlador de dominio (Maldonado, 2017).

Forma manual

Esta no es más que un script hecho en el lenguaje de programación BASH, el cual fue publicado por la Dirección de Redes y Seguridad Informática de la Universidad de las Ciencias Informáticas (UCI), el cual trae consigo algunas configuraciones básicas. Esta de todas las formas es la forma más compleja y menos factible para alcanzar el propósito de integrar un ordenador al AD (Arruebo, 2013).

1.2.2. Valoración del estudio realizado

A raíz del estudio realizado a las distintas formas de autenticar un ordenador con sistema operativo GNU/Linux a un controlador de dominio, se obtuvo que; en el Centro de Control de Gnome no existe forma de autenticar Linux a un controlador de dominio de manera visual. Las herramientas estudiadas cuentan con documentación, la cual permitió una mayor comprensión de sus funcionalidades y de las características. Todas poseen licencia libre y son compatibles con plataforma libre. En el caso de SSSD puede ser usado por el cliente y tienen un nivel medio de consumo de recursos. A pesar de estas características favorables solo PBIS puede ser usada para el desarrollo de la solución, debido a que posee las características que más se asemejan a las deseadas por el cliente.

1.3. Entornos de Escritorio GNU/Linux

Interfaz gráfica de usuario (en inglés conocida por el acrónimo GUI, de Graphic User Interfase) tipo de visualización que permite al usuario elegir comandos, iniciar programas y ver listas de archivos y otras opciones utilizando las representaciones visuales (iconos) y las listas de elementos del menú. Las

selecciones pueden activarse bien a través del teclado o con el ratón. Para los autores de aplicaciones, las interfaces gráficas de usuario ofrecen un entorno que se encarga de la comunicación con el ordenador o computadora. Esto hace que el programador pueda concentrarse en la funcionalidad, ya que no está sujeto a los detalles de la visualización ni a la entrada a través del ratón o del teclado (Montoro, 2011).

También permite a los programadores crear programas que realicen de la misma forma las tareas más frecuentes, como guardar un archivo, porque la interfaz proporciona mecanismos estándar de control como ventanas y cuadros de diálogo. Otra ventaja es que las aplicaciones escritas para una interfaz gráfica de usuario son independientes de los dispositivos: a medida que la interfaz cambia para permitir el uso de nuevos dispositivos de entrada y salida, como un monitor de pantalla grande o un dispositivo óptico de almacenamiento, las aplicaciones pueden utilizarlos sin necesidad de cambios (Montoro, 2011).

KDE

Inicialmente, sus siglas significaban *Kool Desktop Environment*, pero más tarde esta definición fue abandonada. El proyecto KDE es lanzado en 1998, KDE 1, y desde aquí ha avanzado con los años, y ha pasado de ser un entorno de escritorio bastante arcaico (Ideal para la época) a ser algo mucho más limpio y amigable para el usuario. Actualmente incluye un nuevo diseño de escritorio llamado Plasma, que viene a sustituir los elementos usados antes en KDE. Por supuesto, la personalización final va a depender de cada usuario. A continuación se muestra el entorno de escritorio de KDE. Ver Ilustración 1 Entorno de escritorio KDE.

Ilustración 1 Entorno de escritorio KDE.

Gnome

Este escritorio fue el que estuvo por defecto en Ubuntu hasta la versión 11.04, en donde incluyeron su nuevo entorno de escritorio llamado Unity, que es único de Ubuntu. Actualmente es el escritorio por defecto de Fedora. Muchas de las distribuciones más conocidas permiten elegir *Gnome entre sus entornos de escritorio*, y es uno de los escritorios más limpios, aunque requiere muchos recursos del computador comparado con otros entornos. A continuación se muestra el entorno de escritorio de Gnome. Ver Ilustración 2 Entorno de escritorio Gnome.

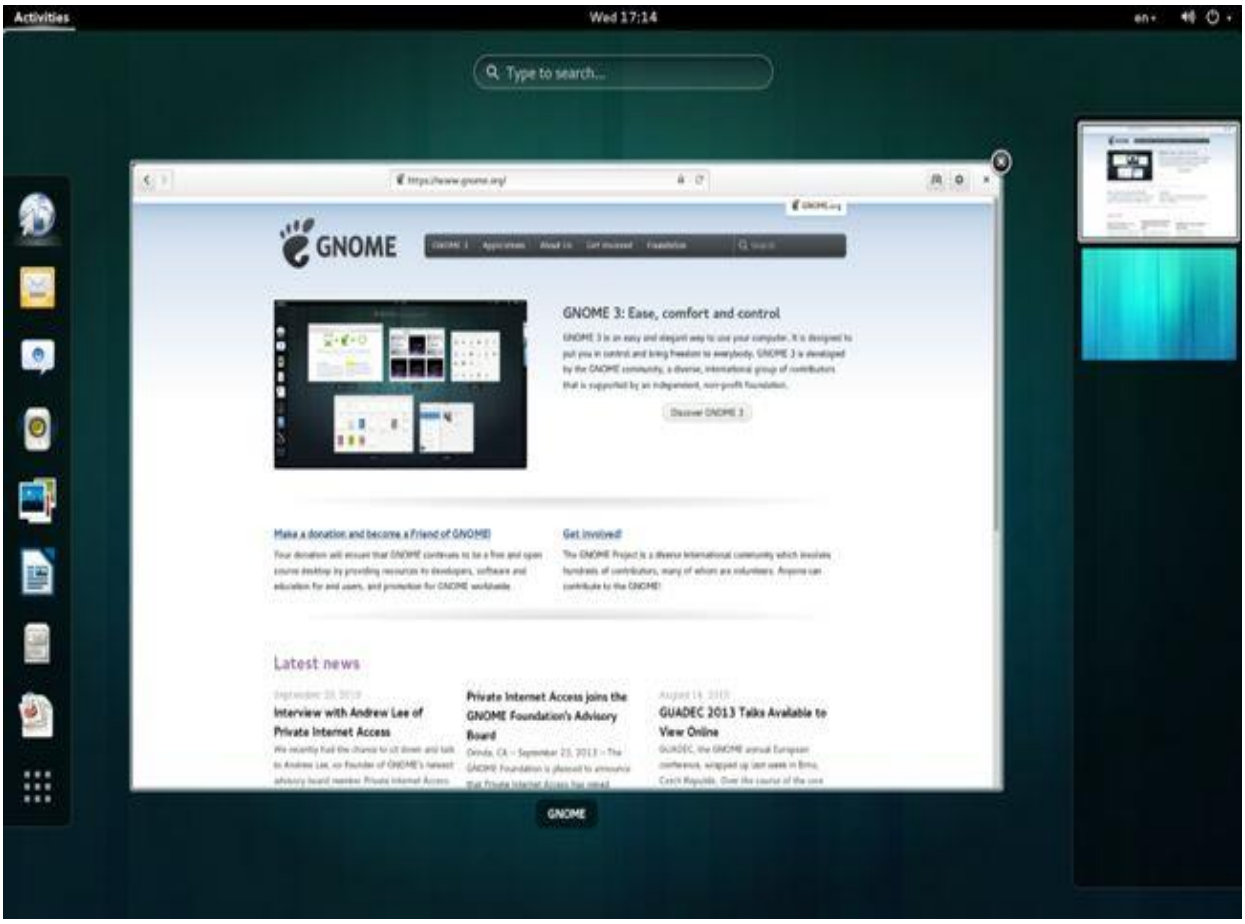

Ilustración 2 Entorno de escritorio Gnome.

LXDE

Para muchos, el entorno de escritorio ideal, y no por su belleza si no por su ligereza. Pide muy pocos recursos para poder funcionar, y aún con eso, no deja de ser agradable a la vista. Por supuesto que no podemos compararlo con Gnome o KDE, pero para consumir tan pocos recursos en el computador, está bastante bien. Excelente para ordenadores antiguos, ya podríamos ponerlos a funcionar y dejar de tenerlos llevando polvo. A continuación se muestra el entorno de escritorio de LXDE. Ver Ilustración 3 Entorno de escritorio LXDE.

Ilustración 3 Entorno de escritorio LXDE.

Cinnamon

El entorno de escritorio por defecto de Linux Mint. Este empezó siendo un Fork de Gnome Shell, pero con el tiempo se ha vuelto tan robusto como cualquier otro. No es un entorno ligero, pero si consume menos recursos que los más famosos como KDE o Gnome. Usa un gestor de ventanas llamado Muffin, que es una bifurcación del gestor de ventanas usado en Gnome 3, Mutter. A continuación se muestra el entorno de escritorio de Cinnamon. Ver Ilustración 4 Entorno de escritorio Cinnamon.

Ilustración 4 Entorno de escritorio Cinnamon.

Valoración del estudio realizado

El estudio realizado de los entornos de escritorio descritos anteriormente, permitió un mejor entendimiento de sus características. Se toma para la investigación el entorno de escritorio Gnome, debido a que es el entorno de escritorio por defecto de la distribución cubana GNU/Linux Nova.

1.3.1. Centros de control en los entornos de escritorio

El centro de control de una distribución GNU/Linux, es la herramienta donde se administran las funcionalidades principales, es la interfaz gráfica para la configuración de los diversos aspectos del sistema de escritorio permitiendo a los usuarios que puedan administrar todo lo relacionado con la configuración de la distribución en cuanto a apariencia, red, antivirus y firewall, periféricos, dispositivos conectados, todo lo referido con el control total de *software* y *hardware*. Desde la creación de los sistemas operativos, sus centros de control han sido parte inherente a ellos (Simón, 2015).

Centro de control de KDE (KControl)

El centro de control KDE, también conocido como KControl, es el administrador de la configuración centralizado para el entorno de escritorio KDE. KControl tiene una arquitectura modular y es parte del paquete kdeadmin. Aunque KControl era instalado por defecto en Kubuntu, esta distribución linux utilizaba una versión no estándar, que fue rediseñada para ser similar al administrador de las configuraciones de Mac OS X. Más tarde este configurador fue integrado en KDE 4 con el nombre de System Settings (Simón, 2015). A continuación se muestra el Centro de control de KDE. Ver Ilustración 5 Centro de control de KDE. Ilustración capturada del entorno de escritorio KDE.

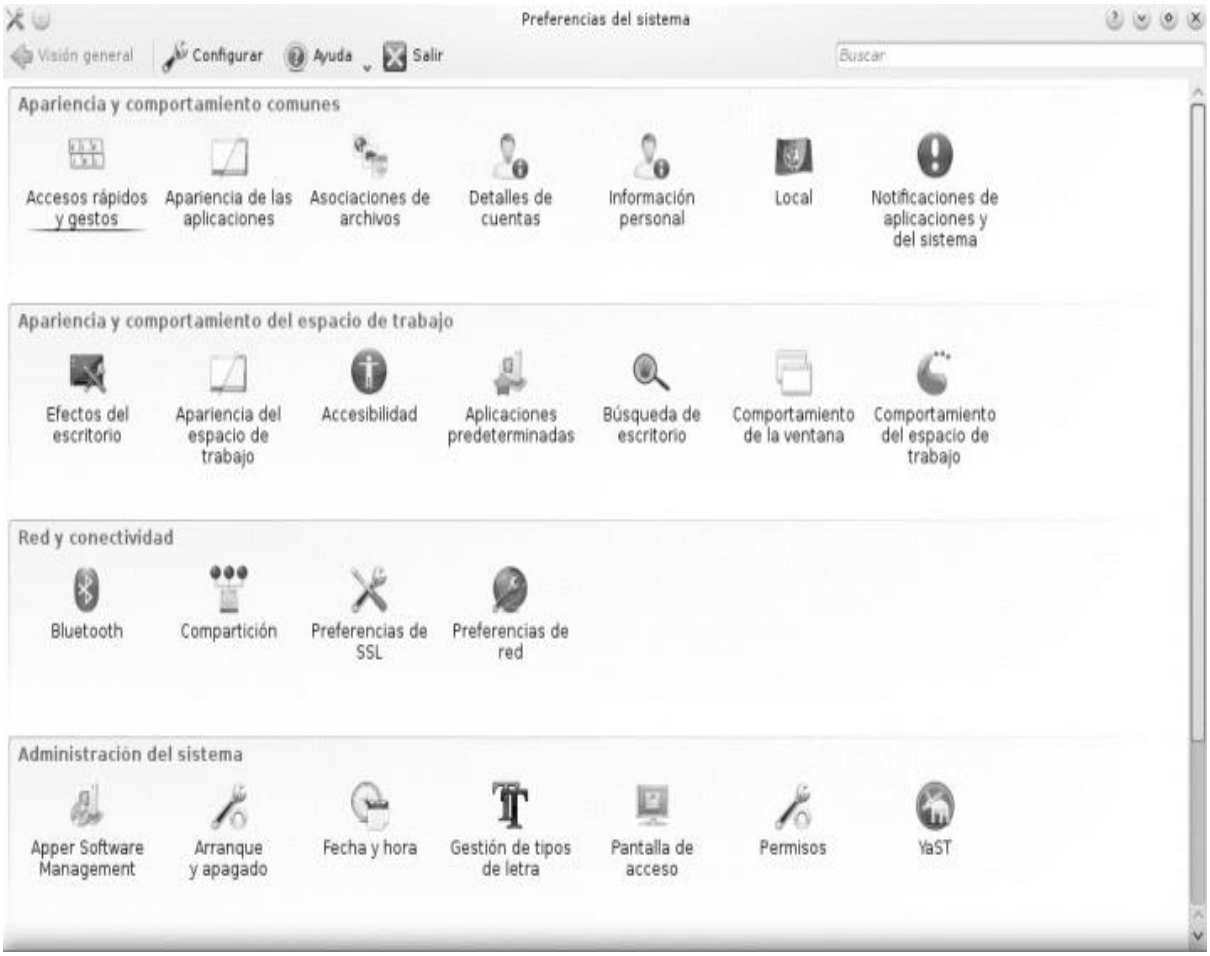

Ilustración 5 Centro de control de KDE. Ilustración capturada del entorno de escritorio KDE.

Centro de control de Gnome

Gnome-control-center 3.12.2 es el administrador de la configuración del entorno de escritorio de Gnome 3.12.2. Está dividido en tres sesiones, con una arquitectura modular, y cada módulo dentro de su respectiva sesión. El mismo está disponible para el trabajo en la configuración de la distribución.

Su código fuente está implementado en el lenguaje ansi C para las funcionalidades, y eXtensible Markup Language (XML) en las interfaces de usuario. A continuación se muestra el Centro de control de Gnome. Ver Ilustración 6 Centro de control de Gnome. Ilustración tomada del entorno de escritorio de Gnome.

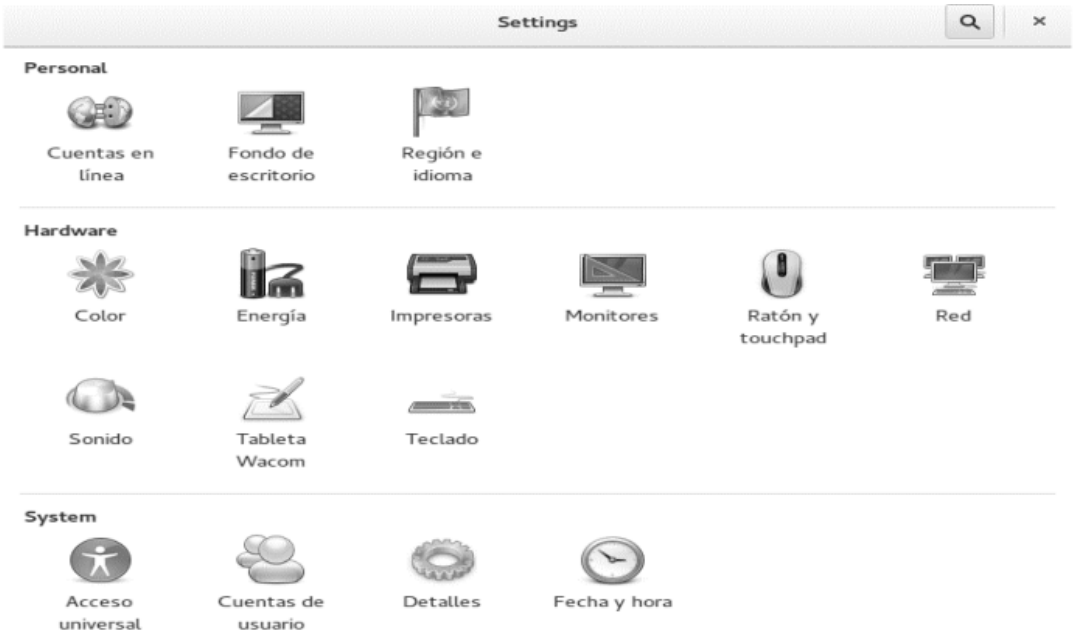

Ilustración 6 Centro de control de Gnome. Ilustración tomada del entorno de escritorio de Gnome.

Centro de configuración de LXDE

El centro de configuración de LXDE es el administrador de la configuración del entorno de escritorio de XLDE. Cuenta de pocas funcionalidades debido a que es el acrónimo inglés de *Lightweight X11 Desktop Environment* (Entorno de Escritorio X11 Liviano), un entorno de escritorio extremadamente rápido, ágil y eficiente en consumo de energía. Las restantes que completan el escritorio se eligen de entre las ofrecidas por terceros. Ver Ilustración 7 Centro de configuración de LKDE. Ilustración tomada del entorno de escritorio de XLDE. A continuación se muestra el Centro de control de LXDE. Ilustración 6 Centro de control de Gnome. Ilustración tomada del entorno de escritorio de Gnome.

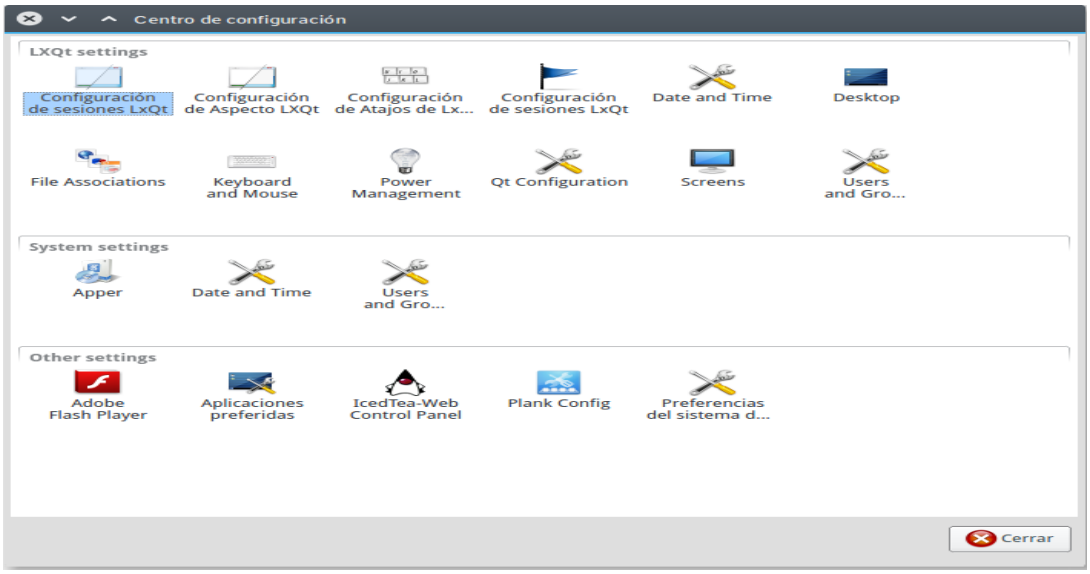

Ilustración 7 Centro de configuración de LKDE. Ilustración tomada del entorno de escritorio de XLDE.

Centro de control de Cinnamon (Configuración del sistema)

El botón “Configuración del Sistema” inicia el Centro de Control Cinnamon. Esta aplicación es el administrador de la configuración centralizado para el entorno de escritorio Cinnamon. Te permite configurar todos los aspectos del escritorio Cinnamon y de la computadora en general. A continuación se muestra el Centro de control de CinnamonVer Ilustración 8 Configuración del sistema de Cinnamon. Ilustración tomada del entorno de escritorio de Cinnamon.

Ilustración 8 Configuración del sistema de Cinnamon. Ilustración tomada del entorno de escritorio de Cinnamon.

1.3.2. Centro de control de Nova

El Centro de control de GNU/Linux Nova posee varios paneles encargados de adaptar el entorno de trabajo a gusto del usuario. El entorno de escritorio Gnome es bastante configurable: se pueden configurar los menús, los iconos, las tipografías, el fondo, el protector de pantalla, el tema, el administrador de ventanas, sonido, la interacción con las ventanas y muchos otros detalles de acuerdo al gusto del usuario. El centro de control de Gnome, conocido como el paquete gnome-control-center, es la aplicación instalada por defecto en el entorno de escritorio Gnome, instalado en las distribuciones de Nova Escritorio, su objetivo es administrar la configuración centralizada del entorno de Gnome (Simón, 2015). A continuación se muestra el Centro de control de Nova. Ver Ilustración 9 Centro de control de Nova 6. Ilustración tomada de Nova.

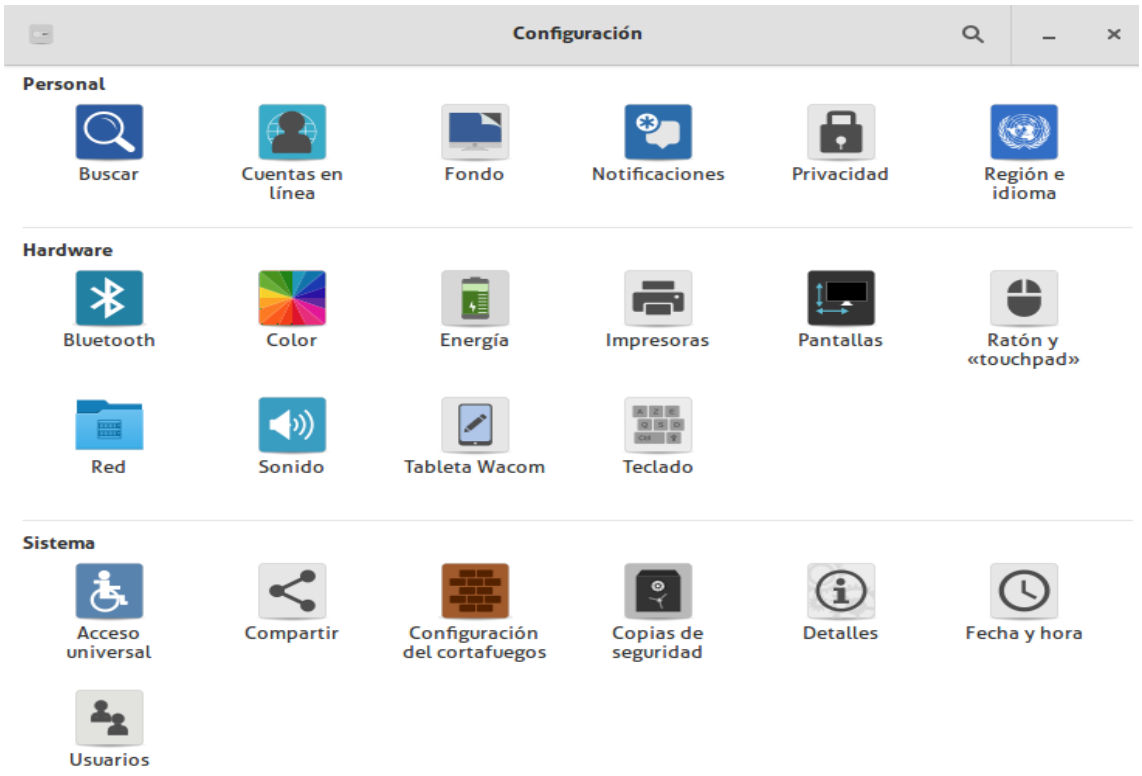

Ilustración 9 Centro de control de Nova 6. Ilustración tomada de Nova.

1.4. Metodología de desarrollo de *software*

La metodología para el desarrollo de *software* es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de *software* comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de *software* desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado (INTECO, 2009).

Para el desarrollo de la herramienta de autenticación de la distribución GNU/Linux Nova a un controlador de dominio desde su centro de control, se emplea la metodología AUP-UCI, resultante de una variación de la metodología ágil AUP.

AUP-UCI. Dispone de tres fases, estas son: inicio, ejecución y cierre. Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación. En la segunda fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. En la fase de cierre se analizan tanto los resultados del

proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto **(UCI, 2015)**. El desarrollo de la presente investigación se centra en la fase de ejecución.

De los escenarios que define AUP-UCI para modelar el sistema, dependiendo de la complejidad y características del proceso de desarrollo, el escenario cuatro se adapta al desarrollo de la herramienta informática propuesta. Las disciplinas de la metodología que serán abordadas en la presente investigación son: Requisitos, Análisis y diseño, implementación y Pruebas interna.

1.5. Entorno de desarrollo

En la actualidad los proyectos productivos son regidos por un conjunto de tecnologías que facilita el trabajo del grupo de desarrollo. En este epígrafe quedan plasmadas las tecnologías a utilizar en el desarrollo de la propuesta de solución, además de definir la metodología de desarrollo y caracterizar cada una de las herramientas.

1.5.1. Herramientas de modelado de software

Como herramienta de modelado se propone utilizar Visual Paradigm, la cual permite la representación gráfica en lenguaje UML[10] del ciclo de vida completo del desarrollo de *software*, características que podemos encontrar en herramientas similares a esta como: Rational Rose y ArgoUML ambas herramientas privativas; Visual Paradigm posee la característica de ser una aplicación multiplataforma de probada utilidad para el analista y diseñada para la construcción de sistemas de forma confiable a través de la utilización de un enfoque orientado a objetos (Arruebo, 2013).

Visual Paradigm 8.0

Es una herramienta diseñada para una amplia gama de usuarios, incluyendo ingenieros de *software*, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquier persona que esté interesada en la construcción de forma fiable de los sistemas de *software* a gran escala con un enfoque orientado a objetos. Ofrece soporte para varias versiones de Lenguaje Unificado de Modelado (UML) y para operaciones de ingeniería inversa de código a modelo o diagrama y de modelo o diagrama a código. Permite el trabajo colaborativo, es decir, varias personas pueden trabajar sobre un mismo proyecto (Hernández, 2009).

[10] Acrónimo de “Unified Modeling Language”, traducido al español como “Lenguaje de Modelado Unificado”. Junto al proceso de desarrollo de *software* RUP, constituyen la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

1.5.2. Lenguaje de modelado de software

UML

Para el modelado de la herramienta informática fue utilizado el lenguaje UML que constituye el estándar para representación del desarrollo de *software*, ilustrando los procesos, objetos y entidades de un *software*, así como sus relaciones (Hernández, 2009).

Las funciones de UML se pueden sintetizar en (Hernández, 2009):

- **Visualizar:** Permite expresar de una forma gráfica un sistema, de manera que otra persona lo puedan entender.
- **Especificar:** Permite especificar cuáles son las características de un sistema antes de su construcción.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.5.3. Lenguaje de programación y principales bibliotecas

En este sub-epígrafe después de un estudio realizado queda plasmado, el lenguaje de programación y las principales bibliotecas a utilizar en la propuesta de solución. Además se dará una descripción de cada uno de ellos.

ANSI C

ANSI C, o simplemente C, es un lenguaje de programación creado en 1972 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de SO, concretamente Unix. C es un lenguaje de programación de nivel medio ya que combina los elementos del lenguaje de alto nivel con la funcionalidad del ensamblador, pero es muy versátil y eficiente y revolucionó las técnicas y estilo de programación (Basurto, 2000).

GTK+

GTK+ es una librería para la creación de interfaces gráficas de usuario. Su licencia permite desarrollar programas libres así como aplicaciones comerciales. Todo esto sin tener que pagar licencias o regalías para su uso. GTK es el acrónimo de GIMP Toolkit debido a que originalmente fue escrito para desarrollar el Programa de manipulación de imágenes de GNU (GNU Image Manipulation Program, GIMP). Desde el punto de vista de GTK+ una aplicación gráfica se construye mediante un conjunto

de objetos que se comunican mediante eventos y retro llamadas. Su diseño, en constante mejora, le permite lograr una gran velocidad de ejecución, interactuar con diferentes lenguajes de programación como C, C++, Python, C#, Perl o Ada, y ser un ambiente de desarrollo multiplataforma (UNIX, Linux, Windows TM, MacOS X TM). GTK es la parte gráfica que interactúa con el usuario. GTK nos ofrece todo lo necesario para el desarrollo de interfaces gráficas de usuario como botones, cajas de texto, menús, ventanas, etc. Otras formas o widgets mucho más complejos pueden ser creados para satisfacer nuestras necesidades (Nieto, 2017).

1.5.4. Entorno de desarrollo de software

Para el desarrollo de la solución propuesta se realizó un estudio en cuanto a las herramientas a utilizar. Teniendo en cuenta las ventajas de cada una y que cumplan con las condiciones del sistema a desarrollar. Se decide hacer uso de NetBeans 8.2 como el IDE de programación y Glade 3.16.1 como herramienta para el desarrollo de interfaces gráficas de usuario.

NetBeans 8.2

NetBeans IDE es un ambiente integrado de desarrollo de código abierto para desarrolladores de *software* que intentan construir aplicaciones usando mayormente Java, o algún otro lenguaje de programación popular como C/C++, PHP, Python, Groovy, Ruby, y otros. Tomando ventaja de casi 10 años de desarrollo constante, la plataforma NetBeans logró ser creada con la contribución de la comunidad de código abierto, siendo un paquete IDE bien diseñado que puede ser usado para la creación de cualquier tipo de aplicaciones de escritorio, web y móvil (Cumbal, y otros, 2009).

También hace foco en permitir el desarrollo de aplicaciones usando conjuntos de componentes modulares de *software* que pueden ser creados tanto por los propietarios de NetBeans, la Corporación Oracle, como por terceros desarrolladores que han logrado expandir sus funcionalidades con numerosos plugins. Tres módulos principales de NetBeans IDE son NetBeans Profiler (que puede monitorear aplicaciones, reportar problemas, dificultades de performance y más), el editor de JavaScript NetBeans y la herramienta de diseño GUI (Cumbal, y otros, 2009).

Glade 3.16.1

Glade es una herramienta que permite rápida y eficientemente el diseño de interfaces gráficas de usuario. Las interfaces de usuario se guardan como un archivo XML que describe la estructura del widget y sus propiedades y estos son asociado con cada uno de los manejadores de señales. El paquete LibGlade puede cargar el archivo de interfaz de usuario con el fin de construir dinámicamente

las aplicaciones. Esto permite modificar la interfaz de usuario estéticamente sin necesidad de recompilar la aplicación (Krause, 2007).

Glade se utiliza para diseñar la interfaz de usuario de una aplicación, configurar las señales que serán asociados con funciones de retro llamada implementadas en el código y cuidar las propiedades comunes del widget. Sin embargo, Glade no es un editor de código o un entorno de desarrollo integrado, imprime los archivos que deben ser cargados por su aplicación, y debe implementar todas las funciones de devolución de llamada en el código. Glade sólo está destinado a simplificar el proceso de inicialización de interfaz gráfica de usuario de la aplicación y la conexión de las señales (Krause, 2007).

1.8. Conclusiones parciales

La sistematización en los conceptos, características y funciones del servicio de directorio permitió una mejor comprensión de su estructura y funcionamiento. También se realizó un estudio de controlador de dominio y sus mecanismos de autenticación a los mismos, arrojando como resultado el análisis de sistemas similares para determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución. La elección de la metodología, herramientas y tecnologías posibilitó obtener las tecnológicas a utilizar para la propuesta de solución. Para guiar el proceso de desarrollo de la herramienta se afilió como metodología de desarrollo de *software* la Variación de AUP para la UCI.

Capítulo 2: Análisis y diseño de una herramienta informática para la autenticación a un controlador de dominio desde el centro de control de Nova

En el presente capítulo se realiza una descripción detallada de la solución propuesta, exponiendo sus características. Además, se explica el modo en que se hace uso de la metodología de desarrollo de *software* AUP-UCI, que guía los esfuerzos para el diseño exitoso de la propuesta solución. Como parte de este proceso de desarrollo, está previsto sobre la marcha generar los artefactos propuestos por la metodología.

2.1. Descripción de la propuesta de solución

La presente investigación tiene como objetivo: desarrollar una herramienta informática para el centro de control de la distribución cubana GNU/Linux Nova, que permita la autenticación a controladores de dominio. La solución cuenta con una interfaz gráfica, esta contiene la funcionalidad de añadir un ordenador al dominio, mostrando un diálogo donde se solicita: nombre del dominio, nombre del ordenador (este campo muestra por defecto el nombre del ordenador, este puede ser modificado en caso de ser necesario), usuario y contraseña del dominio.

Una vez insertados los datos, el usuario puede decidir si cancelar la operación o autenticar al controlador de dominio, en caso de este último, se solicita la contraseña del ordenador haciendo uso de *Policy Kit* para autenticar al usuario como administrador.

Otra de las funcionalidades presentes en la interfaz es: eliminar la autenticación del ordenador en el controlador de dominio, este también hace uso de *Policy Kit* para autenticar al usuario como administrador y poder realizar la operación.

2.2. Modelo conceptual

Para lograr una mayor comprensión del contexto en que se ubica la solución, se recurre al empleo del modelo conceptual. El modelo de dominio es el encargado de representar los conceptos fundamentales para el desarrollo de una aplicación y las diferentes relaciones que existen entre ellos. Son presentados como conceptos los aspectos esenciales y las interrelaciones denotan una estructura de funcionamiento, brindan una mejor comprensión del medio actual para la concepción de un futuro sistema (Pressman, 2001). A continuación presenta el Modelo conceptual que describe el contexto del negocio de la propuesta de solución. Ver Ilustración 10 Modelo conceptual.

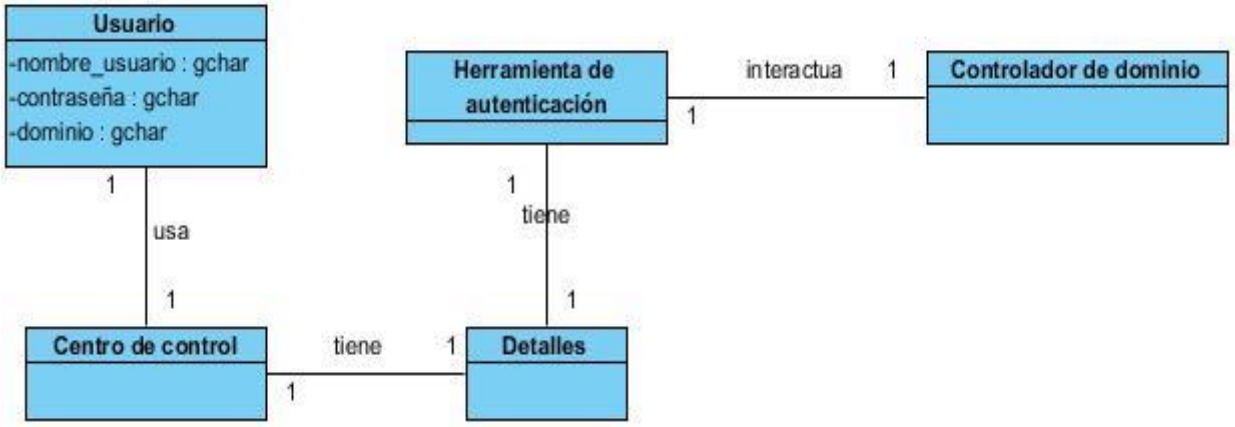

Ilustración 10 Modelo conceptual.

Conceptos	Definiciones
Usuario	Persona que utiliza el centro de control para autenticar el ordenador con distribución GNU/Linux Nova a un controlador de dominio.
Centro de control	Es el administrador de la configuración centralizada de la distribución cubana GNU/Linux Nova.
Herramienta de autenticación	Es la herramienta encargada de realizar la autenticación del ordenador a un controlador de dominio.
Controlador de dominio	Servidor de dominio donde se ejecuta el servicio de Directorio Activo mediante el cual se autenticarán los ordenadores para formar parte de un controlador de dominio, este permite la gestión centralizada de los recursos compartidos en una misma red.

Tabla 1 Definición de los principales conceptos del modelo de dominio.

2.3. Especificación de requisitos de software

El Glosario de Terminología Estándar de Ingeniería de *Software*[11] define al requisito como una condición que necesita un usuario para resolver un problema o lograr un objetivo. También como una capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente (Sommerville, 2005). Los requisitos que se obtienen en este proceso pueden ser funcionales o no funcionales.

[11] IEEE: *Standard Glossary of Software Engineering Terminology*, en Español Tecnología estándar de ingeniería de *software*.

2.3.1. Requisitos funcionales

El análisis de requisitos es una tarea de ingeniería del software que cubre el hueco entre la definición del software a nivel de sistema y el diseño de software. Este análisis permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software (Pressman, 2005).

Listado de requisitos funcionales:

A través de la entrevista aplicada al cliente se determinaron los siguientes requisitos funcionales:

- **RF_1** Autenticar usuario como administrador en el ordenador.
- **RF_2** Modificar nombre ordenador.
- **RF_3** Autenticar ordenador a un controlador de dominio.
- **RF_4** Eliminar la autenticación del ordenador en el controlador de dominio.

2.3.2. Requerimientos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estas se refieren a las características que hacen al producto atractivo, usable, rápido o confiable. Los requisitos no funcionales a menudo se aplican a todo el sistema. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2005).

Apariencia o interfaz externa:

- La aplicación debe poseer una apariencia acorde con los estándares establecidos por el proyecto Gnome. Los componentes visuales deben comportarse de manera consistente y predecible, o sea el comportamiento debe ser el mismo independientemente del entorno en que sea utilizado.

Usabilidad

- El sistema debe ser fácil de usar por las personas con poca experiencia en ordenadores con la distribución cubana GNU/Linux Nova instalada.
- Se utiliza el idioma español para todo el sistema.

Hardware

- El sistema debe funcionar en computadoras con las mismas prestaciones que requiere Nova 6.0 (mayor o igual que 1 GB (gigabyte) de memoria RAM (*Random Access Memory*).

Seguridad

- Solo el usuario root[12] o un usuario con permisos de administración podrán hacer uso de la aplicación.

Software

- La aplicación debe funcionar sobre los sistemas operativos GNU/Linux.

2.4. Historias de usuarios

La metodología AUP-UCI, en su escenario cuatro para la disciplina Requisitos, genera como uno de sus artefactos las Historias de Usuario (HU), que consiste en una técnica para encapsular los requisitos del *software*, a través de un conjunto de tablas que describen brevemente las características que desea el cliente para el sistema a desarrollar. Estas son lo suficientemente comprensibles y delimitadas para que los programadores puedan implementarlas (Rodríguez, 2015).

2.4.1. Descripción de historias de usuarios

A continuación, se muestran las Historias de Usuario:

- Autenticar usuario como administrador en el ordenador.
- Autenticar ordenador a un controlador de dominio.
- Modificar Nombre ordenador.
- Eliminar la autenticación del ordenador en el controlador de dominio.

Historia de usuario	
Número: HU_1	**Nombre del requisito:** Autenticar usuario como administrador en el ordenador.
Programador: Diwel Garcia Valle	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 120 h
Riesgo en desarrollo: No aplica	**Tiempo real:** 123 h

[12] Nombre de usuario o la cuenta que por defecto tiene acceso a todos los comandos y archivos en un sistema operativo Linux o Unix. También se le conoce como la cuenta de root, usuario root y el superusuario.

Descripción: Esta funcionalidad permite la autenticación del usuario como administrador, introduciendo la contraseña de administrador del ordenador. Se realizara haciendo uso de Policy Kit, herramienta de autenticación la cual le muestra al usuario una interfaz donde le solicita la contraseña de administrador. Una vez ingresada la contraseña esta herramienta verifica que esté correcta y permite el acceso al dialogo de autenticación, de no ser así deniega el permiso.
Observación: Para ejecutar la herramienta informática de Autenticación es necesario ser administrador del ordenador.

Tabla 2 Historia de usuario: Autenticar usuario como administrador en el ordenador.

Historia de usuario	
Número: HU_2	**Nombre del requisito:** Autenticar ordenador a un controlador de dominio.
Programador: Diwel Garcia Valle	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 90 h
Riesgo en desarrollo: No aplica	**Tiempo real:** 96 h
Descripción: Esta funcionalidad permite la autenticación del ordenador a un controlador de dominio, describiendo dentro del campo dominio a cual se quiere autenticar, el nombre del ordenador puede ser modificado en caso de que el usuario desee modificarlo, el nombre de usuario y contraseña registrado en el controlador de dominio. En esta funcionalidad se consume los recursos de PBIS para solicitar la autenticación al dominio ejecutando el comando domainjoin-cli join, este recibe los parámetros introducidos por el usuario en el dialogo.	
Observación: En la funcionalidad de autenticación se le notifica al usuario si existe algún error, este puede ser producto a: El nombre del ordenador ya existe en el controlador de dominio, el dominio especificado no es correcto, el nombre de usuario o contraseña no son correctos. También notificara en caso de que algún campo está vacío, cual falta por datos	

Tabla 3 Historia de usuario: Autenticar ordenador a un controlador de dominio.

Historia de usuario	
Número: HU_3	**Nombre del requisito:** Modificar nombre del ordenador.

Programador: Diwel Garcia Valle	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 95h
Riesgo en desarrollo: No aplica	**Tiempo real:** 87 h
Descripción: Esta funcionalidad permite al usuario modificar el nombre del ordenador en caso de que este ya exista en el controlador de dominio. Para modificar el nombre del ordenador se accede al fichero hostname que se encuentra en la dirección /etc/ hostname y en la dirección /proc/sys/kernel/hostname, una vez modificado este fichero el nombre del ordenador es el especificado por el usuario.	
Observación: Esto permitirá que en caso de que exista algún ordenador autenticado en el controlador de dominio con el mismo nombre que el ordenador en cuestión, este pueda ser cambiado por el nombre que el usuario desee.	

Tabla 4 Historia de usuario: Modificar nombre del ordenador.

Historia de usuario	
Número: HU_4	**Nombre del requisito:** Eliminar la autenticación del ordenador en el controlador de dominio.
Programador: Diwel Garcia Valle	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 90 h
Riesgo en desarrollo: No aplica	**Tiempo real:** 83 h
Descripción: Esta funcionalidad permite al usuario en caso que no desee continuar autenticado al controlador de dominio, sacar su ordenador del mismo. Esto se realiza una vez que el usuario selecciona el botón eliminar, se consume los recursos de la herramienta PBI, se ejecuta el comando domainjoin-cli leave.	
Observación: Se le notifica al usuario se tuvo éxito o no a través de un mensaje. En caso de tener éxito estos cambios serán tomados una vez que sea reiniciado el ordenador.	

Tabla 5 Historia de usuario: Eliminar la autenticación del ordenador en el controlador de dominio.

2.5. Análisis y diseño

En este epígrafe se hace alusión a los diferentes estilos y patrones del diseño arquitectónico empleados como buenas prácticas para el desarrollo de la herramienta de Autenticación al Dominio.

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, además de una descripción que sea fácil de mantener y ayude a la estructuración del sistema. En esta disciplina también se compone el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Buschmann, 2001).

Diseñar un sistema es crear la estructura interna y el comportamiento de tal forma que el mismo sea robusto, fácil de extender y de alta calidad. Un buen diseño mejora la calidad y hace que el sistema sea fácil de mantener y extender; por el contrario, un diseño pobre puede aumentar de forma significativa los costos asociados con la producción y el mantenimiento del *software* (Pressman, 2001).

El diseño es una abstracción del código que presenta el sistema desde una perspectiva que hace más sencillo manejar la estructura y el comportamiento del *software*. Los diseños pueden ser modelos visuales, bosquejos simples, descripciones tipo texto, etc. Un buen diseño describe claramente cómo los diferentes elementos del sistema interaccionan para cubrir los requerimientos (Pressman, 2005).

2.5.1. Arquitectura del sistema

La arquitectura del sistema; es el diseño de *software* (con un mayor nivel de abstracción) que ofrece soluciones a problemas de arquitectura de *software* en ingeniería de *software*. Brindan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un correcto diseño arquitectónico expresa un esquema de organización estructural esencial para un sistema de *software*, que consta de subsistemas, sus responsabilidades e interrelaciones (Pressman, 2001).

Según las características de la herramienta informática, se propone el uso del estilo arquitectónico por capas. Esta arquitectura tiene como objetivo principal el de separar los diferentes aspectos del desarrollo en que está conformado el sistema y permite la construcción de sistemas débilmente acoplados. Al dividir un sistema en capas, cada una puede tratarse de forma independiente, sin tener que conocer los detalles de las demás.

La organización de las capas en la herramienta informática quedaría agrupada en dos, dividiéndose en una capa Presentación y una Controladora. La capa de presentación es la que muestra el sistema al usuario, le comunica la información y la captura cuando este la introduce, en un mínimo de procesos.

Esta capa se comunica únicamente con la capa controladora. A continuación se muestra una ilustración de la arquitectura en capas. Ver Ilustración 11 Arquitectura en dos capas de la herramienta informática.

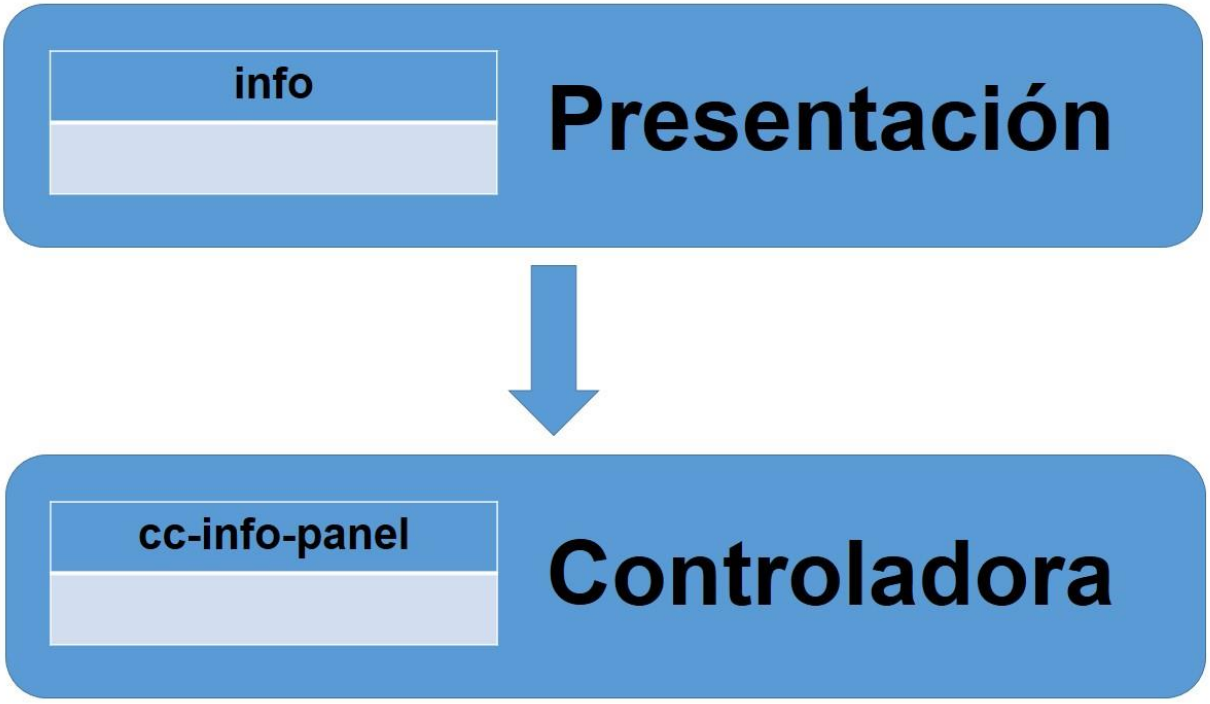

Ilustración 11 Arquitectura en dos capas de la herramienta informática.

Para lograr una mayor comprensión de la estructura de la herramienta informática, se evidencia la arquitectura seleccionada. En el diagrama se muestra el paquete Vista que encierra todos los componentes visuales de la aplicación y permite al usuario la selección de operaciones. En el paquete Funcionalidad se encuentra la implementación de todas las operaciones que puede realizar el usuario donde las peticiones realizadas en la Vista son resueltas por sus respectivas instrucciones del paquete Funcionalidad.

2.5.2. Diseño de la estructura de las funcionalidades

En el presente sub-epígrafe se describe el contenido de la estructura presente en la herramienta de autenticación a un controlador de dominio. Dicha estructura, está conformada por las cabeceras gtk, config, glib, gio y glibtop. Estas cabeceras hacen referencia a bibliotecas de las cuales se consume servicios para el desarrollo de la herramienta informática. Además, esta estructura contiene los métodos necesarios para la autenticación a un controlador de dominio. Ver Ilustración 12 Métodos de la estructura.

```
static void info_panel_setup_domain_app (CcInfoPanel *self);

static void add_domain_clicked (GtkButton *add_domain, CcInfoPanel *self);

void eliminar_domain (GtkButton *eliminar_dom, gpointer *label2);

static void authenticate_domain(GtkButton *autentic, CcInfoPanel *self);

static void cerrar (GtkButton *cancelar1, GtkWidget *dialog);
```

Ilustración 12 Métodos de la estructura.

2.5.3. Patrones de diseño

El objetivo de los patrones es crear un ambiente que ayude a los desarrolladores de *software* a resolver problemas recurrentes que surgen a lo largo del desarrollo. Los patrones ayudan a crear un lenguaje compartido para comunicar perspectiva y experiencia acerca de dichos patrones y sus soluciones. La codificación formal de estas soluciones y sus relaciones permite acumular con éxito el cuerpo de conocimientos que define nuestra comprensión de las buenas arquitecturas que satisfacen las necesidades de sus usuarios (Pressman, 2001).

Existen varios patrones de diseño utilizados en el proceso de desarrollo del *software* que describen los principios fundamentales del diseño de objetos. Estos son agrupados fundamentalmente por dos grandes grupos conocidos como Patrones de *Software* para la Asignación General de Responsabilidad (GRASP - General Responsibility Assignment *Software* Patterns).

Patrones GRASP

Los patrones GRASP constituyen un apoyo para entender el diseño de objetos y aplican el razonamiento para el diseño de una forma sistemática, racional y explicable. En consecuencia, los patrones GRASP son una codificación de principios básicos ampliamente utilizados (Larman, 2003). Los patrones GRASP empleados en el desarrollo de la herramienta informática son descritos a continuación.

Experto

La responsabilidad de asignar una labor a la estructura que tiene o puede tener los datos necesarios para cumplir determinada responsabilidad, es la solución que pretende dar este patrón ante el problema de cómo realizar la asignación de la forma más eficiente posible. Si estas asignaciones de responsabilidades se hacen adecuadamente, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, lo que nos ofrece la garantía de poder reutilizar los componentes en futuras

aplicaciones. Se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos (Larman, 2003). A continuación se evidencia el uso del patrón experto. Ver Ilustración 13 Código que pone en práctica el patrón Experto.

```
self->priv->dominio_data = GTK_ENTRY (gtk_builder_get_object (self->priv->builder, "dom_data"));

self->priv->usuario = GTK_ENTRY (gtk_builder_get_object (self->priv->builder, "user"));
self->priv->label1 = GTK_LABEL (gtk_builder_get_object(self->priv->builder, "label"));
self->priv->password = GTK_ENTRY (gtk_builder_get_object (self->priv->builder, "passw"));
gtk_entry_set_visibility(self->priv->password, FALSE );
```

Ilustración 13 Código que pone en práctica el patrón Experto.

Le asigna a los Gtk_Entry dominio_data, usuario y password, la responsabilidad para la obtención de información de la estructura que cuenta con la información necesaria para cumplir con dicha responsabilidad.

Creador

Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Con esto se favorece el bajo acoplamiento (Larman, 2003). A continuación se evidencia el uso del patrón creador. Ver Ilustración 14 Código que pone en práctica el patrón creador.

```
GtkWidget *dialog = self->priv->options_dialog;
```

Ilustración 14 Código que pone en práctica el patrón creador.

Asigna la responsabilidad de crear una instancia de dialog en la estructura.

Bajo acoplamiento

Este patrón propone la asignación de responsabilidades de manera tal que la dependencia entre una estructura y otra sea la menor posible, de tal forma que se potencie la reutilización y se mitiguen los efectos que puedan producir en una, la realización de cambios en la otra. Lo cual reduce el impacto del cambio (Larman, 2003). A continuación se evidencia el uso del patrón bajo acoplamiento. Ver Ilustración 15 Código que pone en práctica el patrón Bajo acoplamiento.

```
GtkWidget *dialog = self->priv->options_dialog;
```

Ilustración 15 Código que pone en práctica el patrón Bajo acoplamiento.

Asignar una responsabilidad para mantener un bajo acoplamiento entre las estructuras.

Controlador

Es el encargado de asignar la responsabilidad del manejo de uno de los eventos del sistema, a una estructura facultada de atender determinada funcionalidad, dándole solución así al problema fundamental de este patrón, al saber quién debería ocuparse de dicho evento. En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada (Larman, 2003). A continuación se evidencia el uso del patrón controlador. Ver Ilustración 16 Código que pone en práctica el patrón Controlador.

```
add_domain = GTK_WIDGET (gtk_builder_get_object (builder, "add_domain"));

g_signal_connect (add_domain, "clicked",
                  G_CALLBACK(add_domain_clicked),
                  self);
```

Ilustración 16 Código que pone en práctica el patrón Controlador.

Es el encargado de asignar la responsabilidad a la funcionalidad add_domain para el manejo de un mensaje en los eventos de un sistema.

2.5.4. Diagrama de componentes

El diagrama de componentes describe los elementos físicos del sistema y sus relaciones. En él se muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, entre otros (Simón, 2015).

Este diagrama ilustra la relación existente entre los componentes de *software*, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. Dado que estos forman parte de la propuesta de solución. Para un mejor entendimiento, a continuación se muestra el diagrama de componentes. Ver Ilustración 17 Diagrama de componentes..

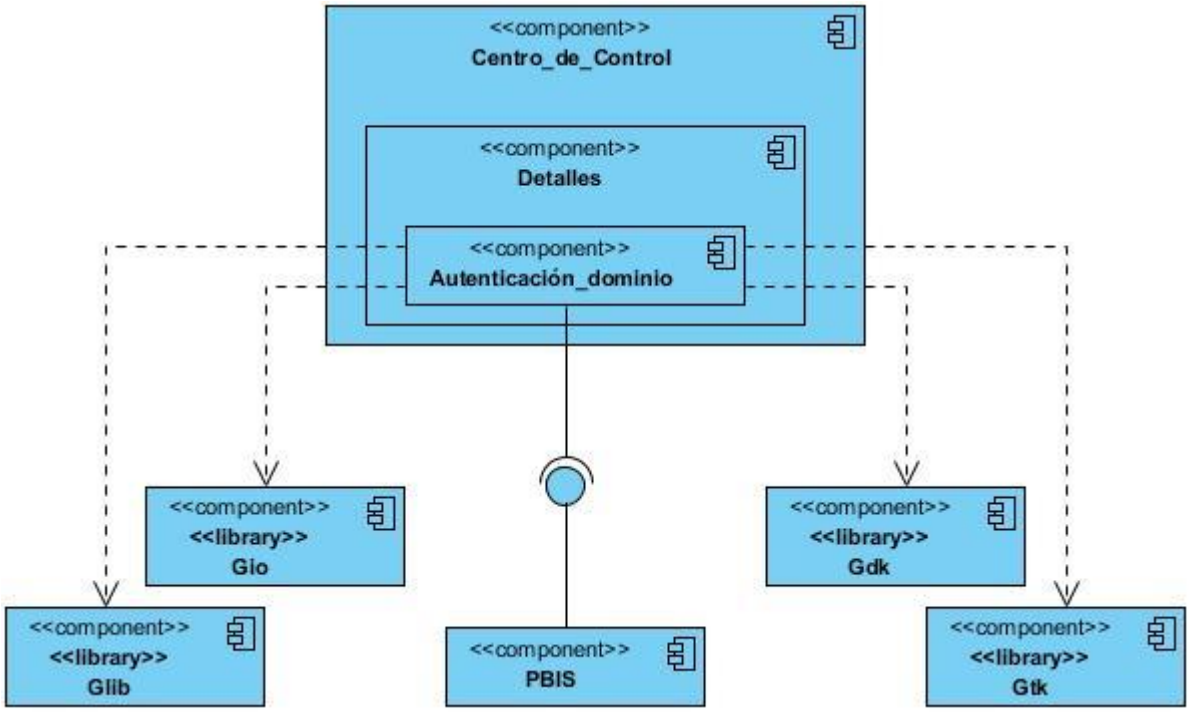

Ilustración 17 Diagrama de componentes.

Descripción de componentes

Los componentes que conforman el diagrama son: Centro de Control, este contiene varios componentes entre los cuales se encuentra Detalles, este contendrá el componente de Autenticación al dominio el cual consume recursos de las bibliotecas glib, gio, gdk y gtk para llevar a cabo su desarrollo, también se comunica con el componente PBIS a través de una interfaz interna de la cual consume los recursos que brinda dicho componente.

2.5.5. Diagrama de despliegue

En un diagrama de despliegue se muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre ellos. Los nodos no son más que elementos físicos que existen en tiempo de ejecución y representan un recurso computacional que generalmente poseen una asignación de memoria y capacidad de procesamiento. El diagrama de despliegue servirá para modelar la topología de *hardware* sobre la cual se ejecutará el sistema (Petitpierre, 2006). A continuación, se muestra el diagrama de despliegue. Ver **¡Error! No se encuentra el origen de la r eferencia.**.

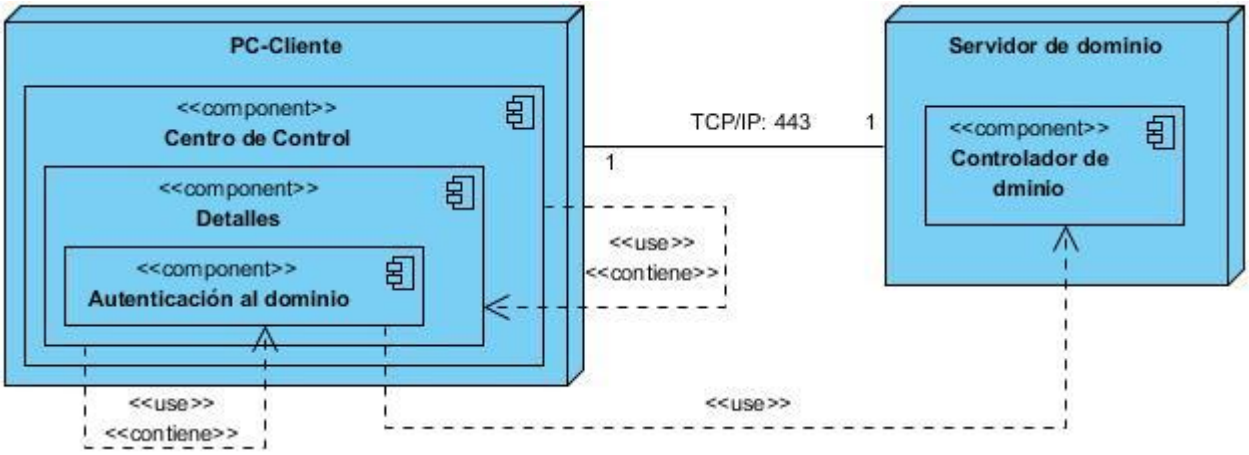

Ilustración 17 Diagrama de despliegue de la herramienta informática.

Descripción del diagrama de despliegue

El diagrama de despliegue de la herramienta informática está compuesto por los nodos PC-Cliente y Servidor de dominio, estos se comunican a través del protocolo TCP/IP por el puerto 443. El nodo PC-Cliente contendrá el centro de control, desde el cual se podrá ejecutar el componente Detalles, este contiene la herramienta de autenticación al dominio. El usuario accede a la herramienta informática, esta se ejecuta sobre ordenadores clientes que hacen peticiones al servidor de dominio para autenticar el ordenador al mismo y este le ofrece una respuesta.

2.6. Conclusiones parciales

Haciendo uso de la metodología ágil AUP-UCI en su escenario cuatro, para lograr flexibilidad y rapidez, se llevó a cabo el proceso ingenieril de la herramienta informática para autenticar a un controlador de dominio. Se obtuvieron 4 requisitos funcionales, 2 de complejidad alta, 1 de complejidad media y 1 de complejidad baja y 6 requisitos no funcionales. Se definió además, la arquitectura n-capas en su variante 2 capas y los patrones GRASP a utilizar. Esto permitió cumplir con el objetivo específico propuesto para el presente capítulo.

Capítulo 3: Implementación y evaluación de la herramienta informática para la autenticación a un controlador de dominio desde el centro de control de Nova

El presente capítulo se lleva a cabo la implementación y ejecución de las pruebas. Las actividades de implementación se contemplan en el desarrollo del sistema que se necesita y las pruebas que son un elemento vital para garantizar la calidad del software y representan una revisión final de las especificaciones, del diseño y la codificación. Además, se definen las pruebas que se van a realizar al sistema para verificar que las funcionalidades estén correctamente implementadas, con el fin de asegurar la calidad de la misma y poder entregar al cliente un mejor producto. Esta etapa de pruebas constituye una actividad en la cual, un sistema y componentes es ejecutado bajo condiciones o requerimientos específicos y los resultados son observados y registrados para su posterior evaluación o corrección.

3.1. Implementación

El proceso de implementación consiste en desarrollar y organizar los componentes basándose en las especificaciones del diseño. Se lleva a cabo mediante la realización de las especificaciones técnicas o algoritmos como un programa, componente de *software*, u otro sistema de cómputo (Petitpierre, 2006). La Implementación comienza con el resultado obtenido del diseño detallado. El objetivo principal de este es desarrollar el diseño de la arquitectura propuesta y el sistema como un todo.

3.1.1. Estándar de codificación

Los estándares de codificación contribuyen a una mejor compresión del código fuente (Abela, 2002). La legibilidad del código fuente, influye directamente en el entendimiento que pueda tener otro programador del mismo, aspecto vital ya que todo *software* tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades, por lo que en la implementación de la herramienta informática para la autenticación a un controlador de dominio se utilizaron los siguientes estándares de implementación:

- **Asignación de nombres.** Evitar nombres largos y que difieran en una letra o en el uso de mayúsculas.
- **Comentarios.** Los comentarios deben añadir claridad al código. Deben contar el por qué y no el cómo. Deben ser concisos.

- **Declaraciones.** Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado.
- **Continuidad de las líneas largas.** Cuando una sentencia no quepa en una única línea se debe fraccionar después de una coma, después de un operador y alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea anterior.
- **Estructura.** Utiliza un solo espacio después de cada delimitador (*coma*)**.** Añade un solo espacio alrededor de los operadores (==, &&, ||).
- **Longitud de la línea.** Limitar todas las líneas a un máximo de 120 caracteres.

3.2. Pruebas de software

El interés por la calidad crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades. Es por esto que es necesario realizar pruebas al sistema a medida que se implementan las funcionalidades (Swebok, 2004).

Las pruebas del software son la actividad más común de control de la calidad realizada en los proyectos para asegurar el correcto funcionamiento del software. Tienen como objetivos la verificación de la correcta implementación de los requisitos explícitamente establecidos, la adecuada integración de los componentes que conforman el sistema y la ejecución de casos de prueba que permitan detectar el mayor número de no conformidades y corregirlas antes de la entrega del software al cliente. Es importante destacar que las pruebas reducen la probabilidad de que aparezcan defectos ocultos en el software, pero incluso si no se encuentra ningún defecto, nunca será una garantía para la perfección del funcionamiento del software (Swebok, 2004).

3.2.1. Pruebas a aplicar

La metodología AUP-UCI desagrega las pruebas en tres disciplinas: internas, liberación y aceptación. Las pruebas de liberación son diseñadas y ejecutadas por una entidad certificadora de calidad externa, por lo que en el presente trabajo no son analizadas, tampoco se realizan pruebas de aceptación. Las pruebas de software aplicadas permiten evaluar las funcionalidades de la herramienta informática. Comprenden un conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de una aplicación, por medio de pruebas sobre al comportamiento de la misma ya que los sistemas informáticos, programas y aplicaciones han crecido a niveles inimaginables en complejidad e interoperabilidad, con lo cual también se han incrementado las posibilidades de defectos (Fernández, 2005).

En principio a la herramienta le fueron aplicadas pruebas de unidad, con el método de caja blanca y la técnica de camino básico para comprobar la correcta ejecución de sus principales funcionalidades. Posteriormente se aplicaron pruebas funcionales con el método de caja negra utilizando la técnica partición de equivalencia donde los requisitos descritos en las historias de usuario fueron evaluados.

3.2.2. Pruebas unitarias

La prueba de unidad analiza cada módulo individualmente, asegurando que funciona adecuadamente como una unidad, haciendo un uso intensivo del método de prueba de caja blanca y ejercitando caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores (Baray, y otros, 2006).

Para el diseño de los casos de prueba fue utilizado el método de caja blanca, donde las pruebas se enfocan en la estructura de control del programa. Los casos de prueba se derivan para asegurar que todos los enunciados en el programa se ejecutaron al menos una vez durante las pruebas y que todas las condiciones lógicas se revisaron (Pressman, 2001).

La técnica de prueba de caja blanca utilizada en la investigación es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Pressman, 2001).

A continuación, se realiza la técnica del camino básico a la funcionalidad eliminar del dominio que permite sacar un ordenador del Directorio Activo. Ver Ilustración 18 Método eliminar_domain.

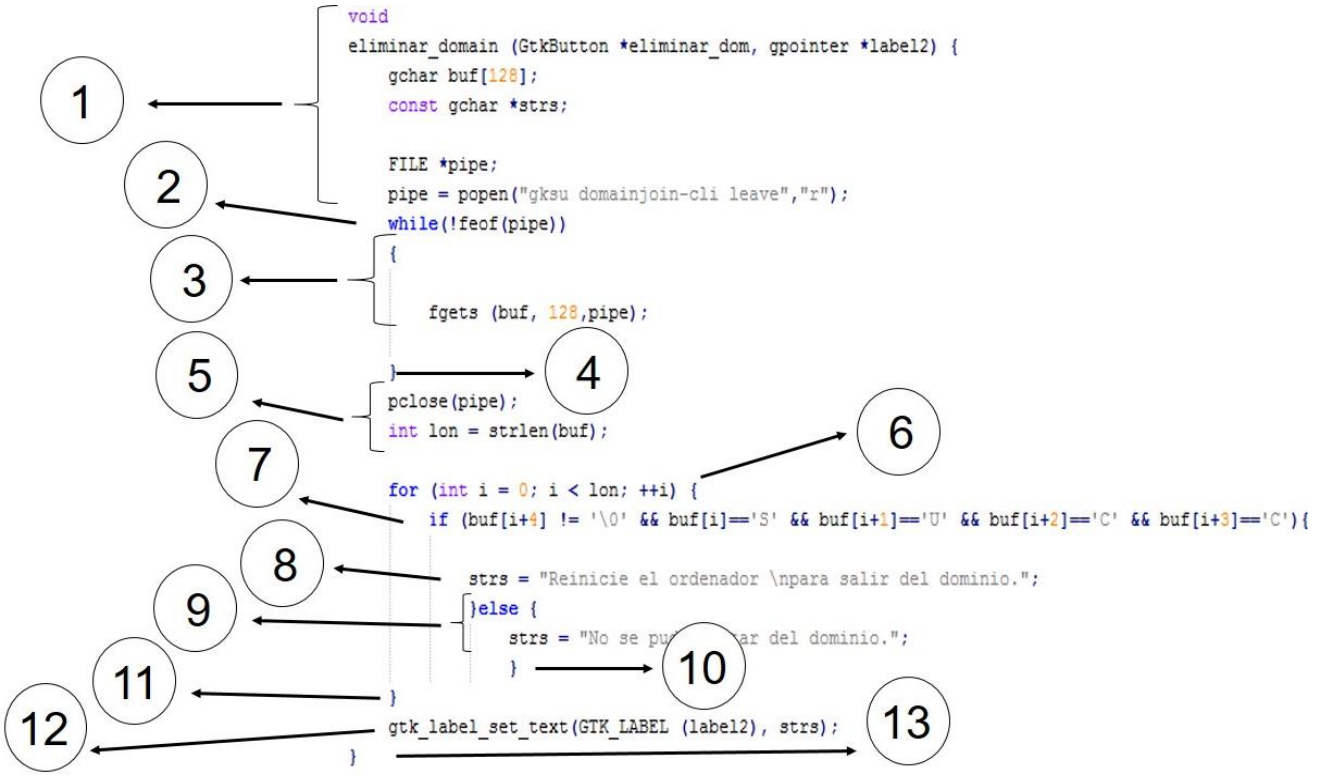

Ilustración 18 Método eliminar_domain.

1. Dibujar el grafo de flujo de la funcionalidad eliminar_domain.

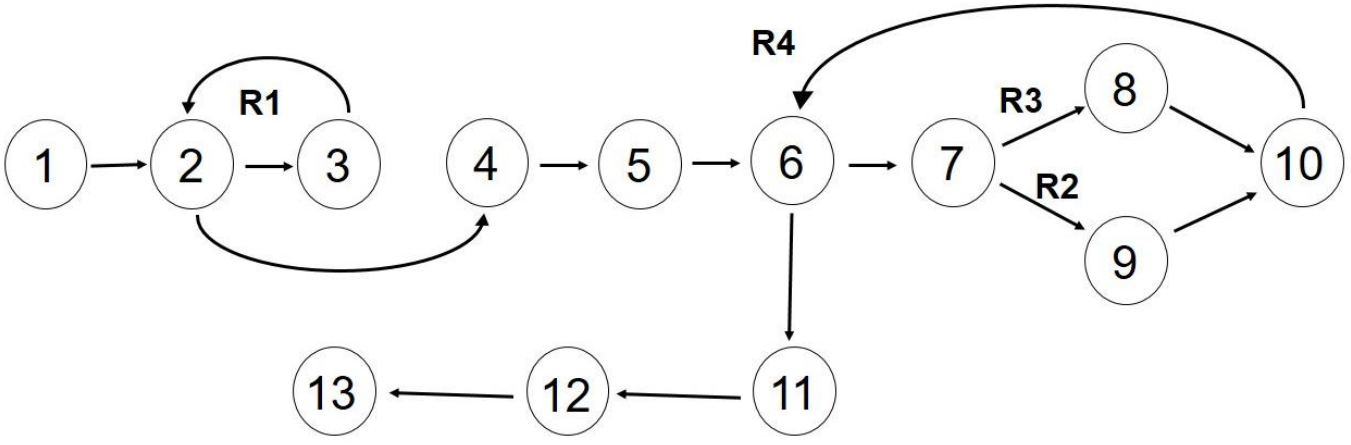

Ilustración 19 Grafo de flujo del método eliminar_domain.

2. Determinar la complejidad ciclomática del grafo

La complejidad ciclomática del grafo se puede calcular de tres formas siguientes:

(1) V(G) = A - N + 2 Donde: A es el número de aristas del grafo de flujo y N es el número de nodos.

V(G) = 15 – 13 +2

V(G) = 4

(2) V(G) = R Donde: R son las regiones, áreas delimitadas por nodos y aristas en el grafo.

V(G) = 4

(3) V(G) = P + 1 Donde: P es el número de nodos predicados (nodos con más de una arista de salida) contenidos en el grafo.

V(G) = 3+1

V(G) = 4

3. Determinar los caminos independientes

Camino 1: 1, 2, 4, 5, 6, 7, 8, 10, 6, 11, 12, 13

Camino 2: 1, 2, 4, 5, 6, 7, 9, 10, 6, 11, 12, 13

Camino 3: 1, 2, 3, 2, 4, 5, 6, 7, 8, 10, 6, 11, 12, 13

Camino 4: 1, 2, 3, 2, 4, 5, 6, 7, 9, 10, 6, 11, 12, 13

4. Definir los casos de pruebas para comprobar la ejecución de cada camino

Diseño de caso de prueba para el camino 3	
Descripción	Método para sacar un ordenador del dominio.
Condición de ejecución	El usuario selecciona la opción de eliminar el ordenador del dominio.
Entrada	GtkButton *eliminar_domain, gpointer *label2: confirma que el usuario ha seleccionado el botón eliminar.
Resultado esperado	Se almacena en la variable buf la salida de pipe, verifica si la salida es correcta y muestra el dialogo en el label.

Tabla 6 Diseño de caso de prueba para el camino 3.

Resultados de la prueba

Se realizaron 2 iteraciones de la prueba unitaria. Para una primera iteración de la prueba fueron detectadas dos no conformidades explicadas a continuación:

- Una vez seleccionada la opción de eliminar no guardaba en la variable buf la salida de pipe.
- El texto esperado en el label no se muestra para notificar al usuario el resultado de la ejecución.

Durante una segunda iteración de la prueba no fueron encontradas no conformidades por lo que los

resultados arrojados fueron satisfactorios.

3.3.1. Pruebas de integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción (Pressman, 2005).

En la evaluación de la solución se probó la integración de la herramienta de autenticación con el centro de control. La siguiente tabla muestra los casos de prueba de integración realizado. Ver Tabla 7 Integración entre el centro de control y la herramienta de autenticación al dominio. (Elaboración propia).

Módulo al cual se integra	Centro de Control
Condiciones de Ejecución	El Centro de Control se encuentra instalado en la PC-Cliente, este debe contener la herramienta para la autenticación al dominio en el panel Detalles.
Descripción de la prueba	Desde el Centro de Control el usuario accede al panel Detalles, el cual contiene la herramienta de autenticación al dominio. Aquí se Comprueba que la herramienta informatica es capaz de realizar el acceso a las funcionalidades.
Entradas/Pasos de ejecución	La herramienta de autenticación recibe los datos a través del dialogo que muestra el Centro de Control para verificarlos y hacer la solicitud de autenticación o de salir del dominio haciendo uso de las funcionalidades de PBIS.
Resultado esperado	Los usuarios tienen acceso a las funcionalidades de siempre que sea administrador del ordenador.
Evaluación	Prueba satisfactoria.

Tabla 7 Integración entre el centro de control y la herramienta de autenticación al dominio. (Elaboración propia).

Al finalizar las pruebas de integración no se detectaron errores asociados a la interacción entre la herramienta de autenticación y el centro de control de la distribución cubana GNU/Linux Nova.

3.3.2. Pruebas funcionales

Una prueba funcional es una prueba basada en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Las pruebas funcionales se hacen mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el

paquete informático. Dicho de otro modo son pruebas específicas, concretas y exhaustivas para probar y validar que el software hace lo que debe y sobre todo, lo que se ha especificado (Pressman, 2011).

Como método de prueba se utilizó caja negra que se enfoca en probar el sistema sin tomar en cuenta la estructura interna del mismo, su objetivo es validar que las salidas sean las esperadas. Se centra en encontrar las circunstancias en las que el sistema no se comporta conforme a las especificaciones establecidas (Pressman, 2005).

La técnica de prueba utilizada fue partición de equivalencia que divide el dominio de entrada de un programa en clases de equivalencia, a partir de las cuales pueden derivarse casos de prueba. Además, descubre clases de errores, que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general. Mediante su empleo se puede reducir al máximo el total de casos de prueba que deben desarrollarse (Sommerville, 2005).

Método de caja negra

Es denominada prueba de comportamiento, ya que se centra en los requisitos funcionales del software. Permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa y está referida a las pruebas que se llevan a cabo sobre la interfaz del software, examinando algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software (código fuente de la aplicación).Para llevar a cabo las mismas se diseñan casos de prueba para estudiar las salidas del sistema y observar si concuerdan con la salida esperada (Pressman, 2011).

Los casos de prueba fueron diseñados teniendo en cuenta las funcionalidades descritas en las historias de usuario permitiendo una mejor compresión de las especificaciones con las que la solución debe cumplir. A continuación, se muestran los casos de prueba correspondientes a las historias de usuario “Autenticar usuario como administrador en el ordenador”, “Autenticar ordenador a un controlador de dominio”, “Modificar nombre del ordenador” y “Eliminar la autenticación del ordenador en el controlador de dominio” ya que representan las historias de usuario de la solución.

Casos de pruebas del sistema

Nombre del requisito	Descripción general	Escenarios de prueba	Flujo del escenario
Autenticar usuario como administrador en el ordenador.	La herramienta debe permitir la autenticación del usuario como administrador.	EP 1.1: Autenticar usuario como	1. Clic en la opción “Autenticar”. 2. Se muestra la interfaz para autenticarse como administrador.

		administrador en el ordenador con éxito.	3. Se introduce la contraseña de administrador correcta.
	La herramienta debe emitir un mensaje de error cuando no es introducida la contraseña correcta.	EP 1.2 Autenticar usuario como administrador en el ordenador insertando la contraseña incorrecta.	1. Clic en la opción “Autenticar”. 2. Se muestra la interfaz para autenticarse como administrador. 3. Se introduce la contraseña de administrador incorrecta. 4. La herramienta muestra un mensaje para informar que la contraseña es incorrecta. .
	La herramienta debe emitir un mensaje de error cuando no es introducida la contraseña.	EP 1.3 Autenticar usuario como administrador en el ordenador sin insertar la contraseña.	1. Clic en la opción “Autenticar”. 2. Se muestra la interfaz para autenticarse como administrador. 3. Se deja el campo contraseña de administrador vacío. 4. La herramienta muestra un mensaje para informar que el campo contraseña está vacío.

Tabla 8 Sección "Autenticar usuario como administrador del ordenador".

Nombre del requisito	Descripción general	Escenarios de prueba	Flujo del escenario
Autenticar ordenador a un controlador de dominio.	La herramienta debe permitir la autenticación del ordenador a un controlador de dominio.	EP 2.1: Autenticar ordenador a un controlador de dominio con éxito.	1. Clic en la opción “Añadir”. 2. Se muestra la interfaz para introducir los datos esperados. 3. Se introducen los datos esperados correctamente. 4. Se solicita la autenticación del ordenador al controlador de dominio haciendo clic en la opción “Autenticar”. 5. Se muestra una notificación informando que tuvo éxito.

	La herramienta debe emitir un mensaje de error cuando existe algún campo vacío.	EP 2.2 Autenticar ordenador a un controlador de dominio dejando campos por llenar.	1. Clic en la opción “Añadir”. 2. Se muestra la interfaz para introducir los datos esperados. 3. Se introducen algunos datos esperados correctamente, se dejan campos vacíos. 4. Se solicita la autenticación del ordenador al controlador de dominio haciendo clic en la opción “Autenticar”. 5. Se muestra una notificación informando que faltan campos por llenar.
	La herramienta debe emitir un mensaje de error cuando el dominio especificado no es encontrado (no existe).	EP 2.3 Autenticar ordenador a un controlador de dominio poniendo el campo dominio incorrecto.	1. Clic en la opción “Añadir”. 2. Se muestra la interfaz para introducir los datos esperados. 3. Se introducen los datos esperados correctamente, menos el campo dominio, este se introduce incorrectamente. 4. Se solicita la autenticación del ordenador al controlador de dominio haciendo clic en la opción “Autenticar”. 5. Se muestra una notificación informando que el campo dominio es incorrecto.
	La herramienta debe emitir un mensaje de error cuando el nombre de usuario o contraseña son incorrectos.	EP 2.4 Autenticar ordenador a un controlador de dominio poniendo los campos “Usuario o Contraseña” incorrectos.	1. Clic en la opción “Añadir”. 2. Se muestra la interfaz para introducir los datos esperados. 3. Se introducen los datos “Dominio y Nombre PC” correctamente, poniendo los campos “Usuario o Contraseña” incorrectos. 4. Se solicita la autenticación del ordenador al controlador de dominio haciendo clic en la opción “Autenticar”. 5. Se muestra una notificación informando que los campos “Usuario o Contraseña” son incorrectos.

	La herramienta debe emitir un mensaje de error cuando el nombre del ordenador ya exista en el controlador de dominio.	EP 2.5 Autenticar ordenador a un controlador de dominio asignándole un “Nombre PC” existente en el mismo.	1. Clic en la opción “Añadir”. 2. Se muestra la interfaz para introducir los datos esperados. 3. Se introducen los datos “Dominio, Usuario y Contraseña” correctamente, asignándole un “Nombre PC” existente en el mismo. 4. Se solicita la autenticación del ordenador al controlador de dominio haciendo clic en la opción “Autenticar”. 5. Se muestra una notificación informando que ya existe un ordenador con ese nombre en el controlador de dominio.

Tabla 9 Sección "Autenticar ordenador a un controlador de dominio".

Nombre del requisito	Descripción general	Escenarios de prueba	Flujo del escenario
Modificar Nombre ordenador.	La herramienta debe permitir modificar el nombre del ordenador.	EP 3.1: Modificar nombre del ordenador con éxito.	1. Clic en la opción “Añadir”. 2. Se muestra la interfaz para introducir los datos esperados. 3. Se introducen los datos correctamente, se modifica el campo “Nombre PC” existente en el mismo. 4. Se solicita la autenticación del ordenador al controlador de dominio haciendo clic en la opción “Autenticar”. 5. Se modifica el nombre del ordenador y se realiza la autenticación.
	La herramienta debe emitir un mensaje de error cuando no se puede modificar el nombre del ordenador.	EP 3.2 Modificar nombre del ordenador sin permiso de administración.	1. Clic en la opción “Añadir”. 2. Se muestra la interfaz para introducir los datos esperados. 3. Se introducen los datos correctamente, se modifica el campo “Nombre PC” existente en el mismo.

			4. Se solicita la autenticación del ordenador al controlador de dominio haciendo clic en la opción “Autenticar”. 5. Se notifica al usuario que no tiene permiso de administrador para realizar la modificación.

Tabla 10 Sección "Modificar Nombre ordenador".

Nombre del requisito	Descripción general	Escenarios de prueba	Flujo del escenario
Eliminar la autenticación del ordenador en el controlador de dominio.	La herramienta debe permitir eliminar la autenticación del ordenador en el controlador de dominio.	EP 4.1: Eliminar la autenticación del ordenador en el controlador de dominio con éxito.	1. Clic en la opción “Eliminar”. 2. Se solicita eliminar la autenticación del ordenador en el controlador de dominio. 3. Se modifica al usuario que tuvo éxito para salir del dominio.
	La herramienta debe emitir un mensaje de error cuando no se puede eliminar la autenticación del ordenador en el controlador de dominio.	EP 4.2 Eliminar la autenticación del ordenador en el controlador de dominio sin permiso de administración.	1. Clic en la opción “Eliminar”. 2. Se solicita eliminar la autenticación del ordenador en el controlador de dominio. 3. Se modifica al usuario que no tiene permiso de administrador en el ordenador.

Tabla 11 Sección "Eliminar la autenticación del ordenador en el controlador de dominio".

Resultado de las pruebas funcionales

Se realizaron tres iteraciones del método de caja negra, en la primera se encontraron 9 no conformidades, de ellas 2 de ortografía y 7 de funcionalidad. Para la segunda iteración fueron corregidas las no conformidades de ortografía y persistieron 3 no conformidades de funcionalidad. Durante la tercera no se detectaron no conformidades. Ver Ilustración 20 Resultados de las iteraciones del método de caja negra.

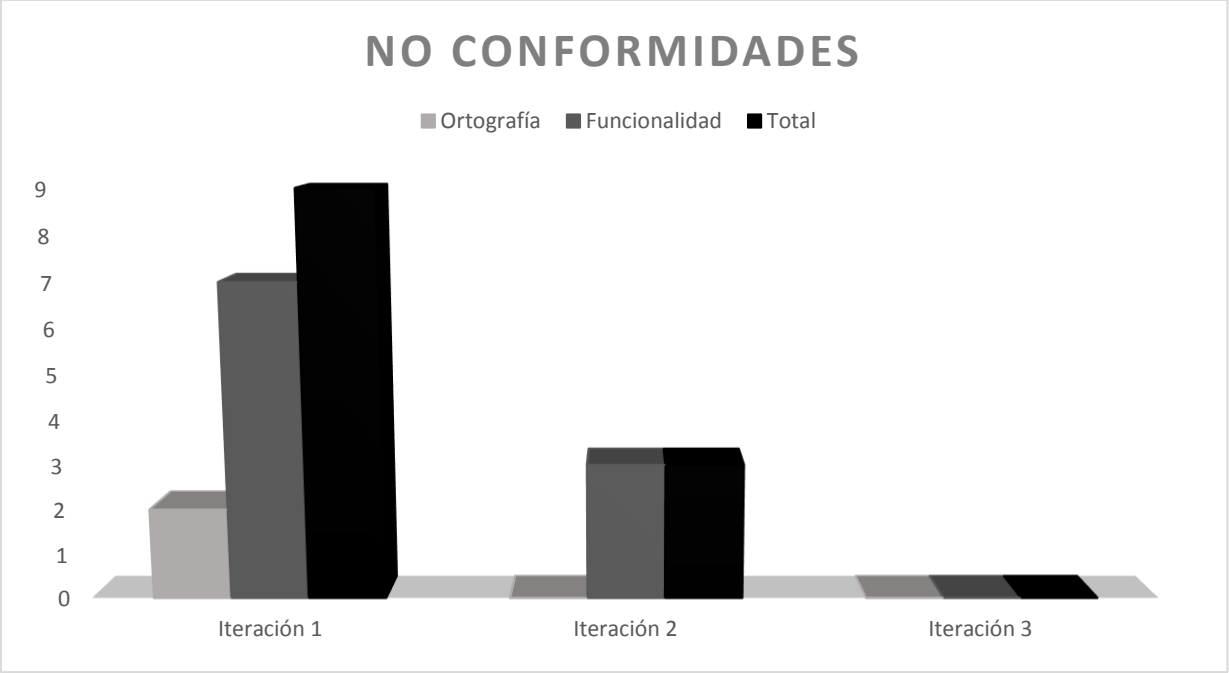

Ilustración 20 Resultados de las iteraciones del método de caja negra.

3.4. Conclusiones parciales

En este capítulo se han abordado los elementos de la implementación de la herramienta de autenticación a un controlador de dominio, así como las pruebas realizadas a dicha herramienta informática y los resultados obtenidos; arribando a las siguientes conclusiones:

El correcto uso de los estándares de codificación permitió que el código del sistema desarrollado fuera legible para lograr una fácil y mejor comprensión del mismo, la cual es de utilidad para el mantenimiento del sistema. La aplicación de pruebas de unidad, integración y pruebas funcionales a la solución desarrollada permitió encontrar errores que afectaban el correcto funcionamiento del sistema, lo que posibilitó corregirlos en un mínimo de tiempo, para que el mismo cumpliera totalmente con los requisitos funcionales definidos en la etapa de análisis.

Conclusiones

A partir de la investigación realizada y los resultados obtenidos referentes a la herramienta de autenticación a un controlador de dominio, se arriban a las siguientes conclusiones:

1. Todo el proceso de desarrollo fue guiado por la metodología AUP-UCI. Se seleccionó PBIS como herramienta base para la autenticación a un controlador de dominio. Asimismo se seleccionaron las tecnologías y herramientas para el desarrollo de la propuesta de solución.
2. A partir de la entrevista realizada al cliente se identificaron cuatro requisitos funcionales y seis no funcionales, descritos a través de cuatro historias de usuario, lo que facilito la descripción, comprensión y posterior codificación de la solución.
3. La implementación de la herramienta informática que permite autenticar un ordenador con distribución cubana GNU/Linux Nova a un controlador de dominio, permitió darle respuesta a los requisitos definidos y de esa manera responder al problema científico planteado.
4. Se evaluó la herramienta informática a través de diferentes pruebas, lo que permitió corregir las no conformidades y obtener un producto de calidad.

Recomendaciones

El autor del presente trabajo recomienda el estudio de otros módulos de los centros de control de sistemas GNU/Linux con el objetivo de continuar centralizando configuraciones para la distribución GNU/Linux Nova.

Bibliografía

Arruebo, Pedro Yaisel Sosa. 2013. *Desarrollo de herramienta de integración con Directorio Activo para Nova.* 2013.

Baray, Héctor Luis Ávila y Coll, Juan Carlos Martínez. 2006. *Introducción a la metodología de la investigación.* 2006.

Basurto, Marco A. Peña y Espín, José M. Cela. 2000. *Introducción a la programación en C.* 2000.

Buschmann, Frank. 2001. *Pattern-oriented software architecture: a system of patterns.* New York : John Wiley & Sons, 2001. Part II.

Carter, Gerald. 2003. *LDAP system administration.* 2003.

Cumbal, Titуaña y Celiano, Walter y Torres Cañizares, Edwin Jesús. 2009. *Elaboración de un manual de la plataforma Netbeans Ide para la Disicom. .* 2009.

Fajardo, Ernesto Pérez y Domínguez Suárez, José Antonio. 2014. *Implementación de Controlador de Dominio basado en la integración de herramientas de software libre.* 2014.

Fernández, Sanz Luis. 2005. *Un sondeo sobre la práctica actual de pruebas de software en España.* 2005. Vol1.

García, Joanna Chirino. 2011. *Estudio de viabilidad de Directorio Activo en Linux.* 2011.

González, Pascual Lopéz, González, Ana Amelia Lopéz y Gallud, José Antonio Lázaro. 1995. *Herramientas CASE. ¿Cómo incorporarlas con éxito en nuestra organización?* 1995.

Guillermo, Lazaro Valdés. 2007. *Software libre.* 2007.

Hernández, Enrique Orallo. 2009. *El lenguaje Unificado de Modelado (UML).* 2009.

INTECO, Laboratorio Nacional de Calidad del Software. 2009. *Ingeniería del Software: Metodologías y ciclos de vida. España.* 2009.

Krause, Andrew. 2007. *Build sophisticated graphical applications using one of the world's most powerful cross-platform toolkits! Foundations of GTK+ Development.* 2007.

Larman, Craig. 2003. *UML y Patrones.* 2003. Segunda edición.

Likewise, Software. 2008. *Likewise Open Installation And Administration Guide.* 2008.

Maldonado, Alex Xavier Rodríguez. 2017. *Análisis e implementación de soluciones de infraestructura que permita la ejecución de aplicativos y escritorios virtuales de manera remota manejando múltiples sesiones simultáneas en servidores.* 2017.

Microsoft, Corporation. 2014. *Windows Protocols Master Glossary.* 2014.

Montoro, Arturo Fernández. 2011. *Cámbiate a LINUX.* 2011.

Nieto, Noe Misael Arroyo. 2017. *Programación de Interfaces Gráficas deUsuario con GTK+ 3 Documentation.* 2017.

Osorio, Wendy Natalia Sabogal. 2015. *Sistema de información de tutor en línea para los estudiantes de la universidad libre de Pereira.* 2015.

Pérez, Fancisco Maciá. 2008. *Administración de servicios de Internet: De la teoría a la práctica.* 2008.

Pérez, Julián Porto y Ana Gardey. 2013. *Definición de protocolo de red.* 2013.

Pérez, Yoandy Villazón. 2013. *El proceso de migración a aplicaciones de código abierto en Cuba desde un enfoque metodológico.* 2013.

Petersen, Richard y Haddad, Ibrahim. 2004. *Red Hat Linux Pocket Administrator.* 2004.

Petitpierre, Claude. 2006. *Software Engineering: The Implementation Phase.* 2006.

Pierra, Allan Fuentes. 2010. *Nova, distribución cubana de GNU/Linux.* 2010.

Pressman, Roger. 2001. *Ingeniería de software. Un enfoque práctico.* 2001.

Pressman, Roger S. 2011. *Software Engineering, a practitioner's approach.* s.l. : McGraw-Hill, 2011. 7th.

Pressman, Roger. 2005. *Software Engineering. A practitioner´s Approach. .* 2005. 7ma edición.

Rodríguez, Tamara Sánchez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Sención, Antonio Perpiñan. 2003. *Fundación de código libre.* 2003.

Simón, Angel Ismael Van Brakle. 2015. *Centro de Control de Nova Escritorio 5.0.* 2015.

Sommerville, Ian. 2005. *Ingeniería del software.* 2005.

Stallman, Richard Matthew. 2004. *Software libre para una sociedad libre.* 2004.

Swebok, IEEE. 2004. *Guide to the Software engineering body of knowledge.* 2004.

UCI, Universidad de las Ciencias Informáticas. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Vejo, Bislandry Paula y Gómez, Carlos Antonio Brisuela. 2008. *El servicio de directorio como medio de integraciónde aplicaciones de red.* 2008.

Windows Corporation. 2014. *Active Directory Technical Specification.* 2014.

Referencias Bibliográficas

Arruebo, Pedro Yaisel Sosa. 2013. *Desarrollo de herramienta de integración con Directorio Activo para Nova.* 2013.

Baray, Héctor Luis Ávila y Coll, Juan Carlos Martínez. 2006. *Introducción a la metodología de la investigación.* 2006.

Basurto, Marco A. Peña y Espín, José M. Cela. 2000. *Introducción a la programación en C.* 2000.

Buschmann, Frank. 2001. *Pattern-oriented software architecture: a system of patterns.* New York : John Wiley & Sons, 2001. Part II.

Carter, Gerald. 2003. *LDAP system administration.* 2003.

Cumbal, Tituaña y Celiano, Walter y Torres Cañizares, Edwin Jesús. 2009. *Elaboración de un manual de la plataforma Netbeans Ide para la Disicom. .* 2009.

Fajardo, Ernesto Pérez y Domínguez Suárez, José Antonio. 2014. *Implementación de Controlador de Dominio basado en la integración de herramientas de software libre.* 2014.

Fernández, Sanz Luis. 2005. *Un sondeo sobre la práctica actual de pruebas de software en España.* 2005. Vol1.

García, Joanna Chirino. 2011. *Estudio de viabilidad de Directorio Activo en Linux.* 2011.

González, Pascual Lopéz, González, Ana Amelia Lopéz y Gallud, José Antonio Lázaro. 1995. *Herramientas CASE. ¿Cómo incorporarlas con éxito en nuestra organización?* 1995.

Guillermo, Lazaro Valdés. 2007. *Software libre.* 2007.

Hernández, Enrique Orallo. 2009. *El lenguaje Unificado de Modelado (UML).* 2009.

INTECO, Laboratorio Nacional de Calidad del Software. 2009. *Ingeniería del Software: Metodologías y ciclos de vida. España.* 2009.

Krause, Andrew. 2007. *Build sophisticated graphical applications using one of the world's most powerful cross-platform toolkits! Foundations of GTK+ Development.* 2007.

Larman, Craig. 2003. *UML y Patrones.* 2003. Segunda edición.

Likewise, Software. 2008. *Likewise Open Installation And Administration Guide.* 2008.

Maldonado, Alex Xavier Rodríguez. 2017. *Análisis e implementación de soluciones de infraestructura que permita la ejecución de aplicativos y escritorios virtuales de manera remota manejando múltiples sesiones simultáneas en servidores.* 2017.

Microsoft, Corporation. 2014. *Windows Protocols Master Glossary.* 2014.

Montoro, Arturo Fernández. 2011. *Cámbiate a LINUX.* 2011.

Nieto, Noe Misael Arroyo. 2017. *Programación de Interfaces Gráficas deUsuario con GTK+ 3 Documentation.* 2017.

Osorio, Wendy Natalia Sabogal. 2015. *Sistema de información de tutor en línea para los estudiantes de la universidad libre de Pereira.* 2015.

Pérez, Fancisco Maciá. 2008. *Administración de servicios de Internet: De la teoría a la práctica.* 2008.

Pérez, Julián Porto y Ana Gardey. 2013. *Definición de protocolo de red.* 2013.

Pérez, Yoandy Villazón. 2013. *El proceso de migración a aplicaciones de código abierto en Cuba desde un enfoque metodológico.* 2013.

Petersen, Richard y Haddad, Ibrahim. 2004. *Red Hat Linux Pocket Administrator.* 2004.

Petitpierre, Claude. 2006. *Software Engineering: The Implementation Phase.* 2006.

Pierra, Allan Fuentes. 2010. *Nova, distribución cubana de GNU/Linux.* 2010.

Pressman, Roger. 2001. *Ingeniería de software. Un enfoque práctico.* 2001.

Pressman, Roger S. 2011. *Software Engineering, a practitioner's approach.* s.l. : McGraw-Hill, 2011. 7th.

Pressman, Roger. 2005. *Software Engineering. A practitioner´s Approach. .* 2005. 7ma edición.

Rodríguez, Tamara Sánchez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Sención, Antonio Perpiñan. 2003. *Fundación de código libre.* 2003.

Simón, Angel Ismael Van Brakle. 2015. *Centro de Control de Nova Escritorio 5.0.* 2015.

Sommerville, Ian. 2005. *Ingeniería del software.* 2005.

Stallman, Richard Matthew. 2004. *Software libre para una sociedad libre.* 2004.

Swebok, IEEE. 2004. *Guide to the Software engineering body of knowledge.* 2004.

UCI, Universidad de las Ciencias Informáticas. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Vejo, Bislandry Paula y Gómez, Carlos Antonio Brisuela. 2008. *El servicio de directorio como medio de integraciónde aplicaciones de red.* 2008.

Windows Corporation. 2014. *Active Directory Technical Specification.* 2014.

Anexos

1. Entrevistas realizada al cliente para conocer las funcionalidades que debía cumplir la herramienta

1. ¿Existe alguna forma para realizar la autenticación de la distribución cubana GNU/Linux Nova a un controlador de dominio en la actualidad?
2. ¿Con qué herramientas lo hace?
3. ¿Por qué usted considera necesario la autenticación a un controlador de dominio?
4. ¿Qué facilidades quisiera usted que le brindara la herramienta a desarrollar?

2. Encuestas realizada a especialistas que trabajan con la distribución cubana GNU/Linux Nova para conocer el estado de la autenticación a un controlador de dominio antes de la presente investigación.

- ¿Cómo se realiza el proceso de autenticación a un controlador de dominio?
- ¿Qué herramientas utilizan para este proceso de autenticación?
- ¿Cree usted conveniente tener una herramienta en el centro de control para autenticar GNU/Linux Nuca a un controlador de dominio?
- ¿Por qué?

3. Prototipos de interfaz