Solución para la integración de la Distribución Cubana de GNU/Linux Nova con un dominio de Directorio Activo.

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

Autor

Ronaldo Castellanos Alvarez

Tutores

Ing. Gladys Marsi Peñalver Romero
Ing. Eduardo Alejandro Cuesta Llanes

La Habana, Cuba
"Año 57 de la Revolución"

Me lo contaron y lo olvidé; lo vi y lo entendí; lo hice y lo aprendí.

Confucio

Declaración de Autoría

Declaro ser el único autor de este trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de ___ del año 2015.

_____

Ronaldo Castellanos Alvarez

_____ _____

Ing. Gladys Marsi Peñalver Romero **Ing. Eduardo Alejandro Cuesta Llanes**

Agradecimientos

A mis padres, ya que gracias a ellos soy quien soy.

A mis tutores por su atención y paciencia en cada momento.

A mis compañeros de aula y del proyecto.

A Carlos por su ayuda incondicional.

Al White que solo falla en raras ocasiones.

Dedicatoria

A mi madre que luchó con todas sus fuerzas para hacer que me graduara y que nunca perdiera el camino del saber.

Resumen

Para establecer el control de una red de ordenadores se hace necesario poseer un mecanismo que permita gestionar el control de sus recursos. Los servicios de directorio ofrecen una solución viable a esta problemática, estos proveen a los administradores de redes una alternativa para gestionar el control de los recursos de la red. Por lo que el objetivo de esta investigación es el desarrollo de TodoEnUno, herramienta que permite integrar ordenadores con la Distribución Cubana de GNU/Linux Nova a dominios de Directorios Activos; con el fin de apoyar el proceso de migración que se lleva a cabo en el país, facilitando el trabajo de los especialistas en migración. Permite además, gestionar el acceso de los grupos y usuarios del dominio y logra añadir usuarios al grupo de administradores. Para su desarrollo contó con la guía de la metodología de desarrollo OpenUp, lenguaje de programación Python y el estándar de programación empleado es CamelCase. Aportando para los especialistas en migración una herramienta capaz de simplificar las operaciones que realizan durante el desarrollo del proceso de migración.

Palabras clave: *Distribución Cubana de GNU/Linux Nova, Directorio Activo, dominio, integración, migración.*

Índice de contenido

Introducción

Hoy en día las redes ofrecen múltiples servicios que permiten realizar actividades como: compartir recursos, hacer conexiones remotas, comunicación mediante el correo electrónico, video y entretenimiento interactivo. A medida que aumentan las redes también lo hacen los recursos y usuarios sobre ellas; por lo que se hace necesario almacenar y organizar la información para su utilización, localización y su control de acceso por los usuarios de la red (Paseur, Sarria, 2010).

En la década del 90 se dieron los primeros pasos al respecto con el surgimiento del concepto de servicios de directorio, los cuales representan una base de datos especializada en operaciones de consultas. Almacenan información sobre los recursos presentes en la red, la pone a disposición del administrador del servidor para su gestión y de los usuarios para realizar operaciones de consulta de la información no privada; surgiendo posteriormente las primeras implementaciones (Paseur, Sarria, 2010).

El uso de los servicios de directorio en las instituciones es de vital importancia y se hace casi imprescindible con el desarrollo actual de las tecnologías. Aquellas que poseen un número significativo de ordenadores necesitan que estos se comuniquen entre sí, por lo que forman parte de redes de ordenadores de manera que pueden intercambiar información. Desde el punto de vista de la administración de sistemas, la mejor forma de aprovechar esta característica es con la creación de un dominio[1] mediante el uso de Controladores de Dominio, donde la información administrativa y de seguridad se encuentra distribuida en uno o varios servidores, facilitando así la labor del administrador.

Los Controladores de Dominio son servidores que se ejecutan sobre los sistemas operativos. Son usados como un medio de organizar, controlar y administrar centralizadamente el acceso a los recursos de la red; y permiten el acceso a recursos por todo el dominio usando un único acceso o credencial (Microsoft, 2005).

Dentro de los servicios de directorio se encuentra el Directorio Activo (DA) (Desmond, Allen, 2008), usado actualmente en la Universidad de las Ciencias Informáticas (UCI), para gestionar los recursos de su red de

1 dominio: Nombre único que permite ingresar a un servidor sin saber la dirección IP exacta donde se encuentra (Alegsa, 2015).

trabajo, ya que posee un gran número de ordenadores. Mediante su uso es posible la creación de un dominio que tiene como objetivo controlar los recursos de la red. Por lo que es necesario, cuando un usuario reinstala su ordenador que este se encuentre unido al dominio establecido, medida que se tomó por la universidad con el objetivo de gestionar los recursos de su red de trabajo.

El Centro de *Software* Libre (CESOL), perteneciente a la estructura productiva de la UCI posee el departamento Servicios Integrales de Migración Asesoría y Soporte (SIMAYS), encargado de guiar el proceso de migración hacia *software* libre en el país; tarea que se encuentra en desarrollo hace algunos años con el objetivo de alcanzar la soberanía tecnológica. Durante el proceso de migración, es necesario llevar a cabo un conjunto de tareas tales como: denegar y permitir el acceso a usuarios o grupos del dominio y añadir permisos administrativos a usuarios o grupos del dominio. Tareas que hoy no se integran en su totalidad en una herramienta.

Actualmente para incluir los ordenadores con la Distribución Cubana de GNU/Linux Nova a un dominio de DA durante la migración, se utiliza la herramienta Likewise. Sin embargo esta aplicación carece de una funcionalidad que permita brindar a usuarios o grupos permisos administrativos. Otras como unir o separar el ordenador del dominio se pueden realizar mediante su interfaz gráfica; pero en el caso de denegar y permitir el acceso a usuarios o grupos del dominio es necesario el uso de la Terminal de Comandos, lo cual dificulta su configuración.

Debido a que la herramienta existente para el apoyo a la migración no permite realizar en su totalidad las tareas necesarias; y a la adopción de la Distribución Cubana de GNU/Linux Nova como alternativa para lograr la independencia tecnológica del país, surge la necesidad de desarrollar una herramienta que permita al especialista en migración ejecutar las tareas de integración de ordenadores a un dominio de DA sin hacer uso de la Terminal de Comandos y logre la compatibilidad e interoperabilidad con la distribución antes mencionada.

Por lo anteriormente expuesto se define como **problema de investigación**: ¿Cómo simplificar el proceso de integración de la Distribución Cubana de GNU/Linux Nova con un dominio de DA?

Así mismo se asume como **objeto de estudio** las herramientas para integrar ordenadores con GNU/Linux a un dominio de DA enmarcado en el **campo de acción** las herramientas para integrar ordenadores con la Distribución Cubana de GNU/Linux Nova a un dominio de DA.

El **objetivo general** del presente trabajo es desarrollar una herramienta para simplificar la integración de la Distribución Cubana de GNU/Linux Nova con un dominio de DA.

En el marco de este trabajo se han podido identificar como **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Diseñar la aplicación que permita la integración a dominios de DA.
- Desarrollar y probar las funcionalidades definidas para la propuesta de solución.

Teniendo definidos los objetivos específicos de la investigación, se determina como **idea a defender** que: El desarrollo de una herramienta para la integración de la Distribución Cubana de GNU/Linux Nova con un dominio de DA simplificará las tareas llevadas a cabo por el especialista durante el proceso de migración.

Para dar cumplimiento a los objetivos específicos se hace necesario definir un conjunto de **tareas de investigación** que a continuación se mencionan:

- Análisis de la bibliografía relacionada con la integración a los dominios de los DA.
- Definición de las tecnologías y herramientas para el apoyo del desarrollo de la solución propuesta para la Distribución Cubana de GNU/Linux Nova.
- Análisis y diseño de la solución propuesta para la Distribución Cubana de GNU/Linux Nova.
- Implementación de las funcionalidades definidas para la herramienta que permita la integración de ordenadores con la Distribución Cubana de GNU/Linux Nova a dominios de DA.
- Ejecución de los casos de pruebas de aceptación al sistema para validar su funcionamiento.

Con el objetivo de dar cumplimiento a estas tareas se emplean en la investigación los siguientes métodos teóricos:

El método **Histórico-Lógico** se utilizó para analizar las características de las herramientas que permiten la integración de ordenadores a un dominio de DA que existen en la actualidad, para identificar sus elementos distintivos y las tecnologías a utilizar para proceso de desarrollo.

El método **Analítico-Sintético** permitió analizar toda la información obtenida durante el estudio de las herramientas que permiten la integración a los DA y de ella obtener los rasgos más significativos que aportan a la investigación.

El presente documento se compone por una introducción, tres capítulos, conclusiones generales, referencias bibliográficas, bibliografía y los anexos. La estructura de los capítulos se define a continuación:

Capítulo 1: Fundamentación teórica

En este capítulo se definen los principales conceptos y se realiza un estudio del estado actual de las herramientas y aplicaciones que se usan para unir los ordenadores a dominios de DA. Se definen cuáles son las herramientas y tecnologías que se van a utilizar para el desarrollo de la herramienta así como la metodología que va a guiar el proceso de desarrollo justificando en cada caso la elección tomada.

Capítulo 2: Análisis y diseño

En este capítulo se describe la solución propuesta, se definen los requisitos funcionales y no funcionales que va a poseer la herramienta a desarrollar. Lo que facilita la identificación de las funcionalidades que van a ser descritas a través de casos de uso. Se define todo lo relacionado con el diseño de la herramienta, así como la arquitectura y los patrones de diseño a utilizar.

Capítulo 3: Implementación y Prueba

En este capítulo se definen los aspectos relacionados con la implementación y la ejecución de las pruebas. Se define el estándar de código a utilizar en la implementación de la solución y se realiza la planificación para la implementación de las funcionalidades identificadas para la herramienta. Se define el método de prueba y el tipo de prueba a aplicar para verificar el correcto funcionamiento de la herramienta, a través de los casos de prueba. Por último se argumenta sobre la importancia del desarrollo de la herramienta para la comunidad de *software* libre y el país.

Capítulo 1: Fundamentación Teórica

Durante el presente capítulo se abordan algunos conceptos para lograr un mejor entendimiento de los contenidos que se tratan durante la investigación. Se realiza un estudio sobre las herramientas actuales que permiten integrar ordenadores a dominios de DA, con el objetivo de establecer una comparación entre ellas y de esta forma identificar si alguna puede servir para satisfacer las necesidades del cliente. También se definen las tecnologías que se van a utilizar para llevar a cabo el proceso de construcción de la solución propuesta.

1.1: Principales Conceptos

Antes de comenzar a describir el estado del arte de la investigación es necesario tener claro los conceptos y funciones principales de los servicios de Directorio Activo. Es por eso que hay que conocer los conceptos básicos que se interrelacionan entre sí a la hora de hacerlos funcionar.

Directorio Activo: servicio de directorio que incorporan los entornos Windows. El DA, es un servicio de red, que almacena información acerca de los recursos existentes y controla el acceso de los usuarios (Chico, 2011).

Domain Name System (DNS): es un sistema para asignar nombres a equipos y servicios de red que se organiza en una jerarquía de dominios (Liu, Albitz, 2000).

Dominio de Directorio Activo: es una colección de objetos dentro del directorio que forman un subconjunto administrativo. Pueden existir diferentes dominios dentro de un bosque y para nombrarlos se utiliza el protocolo DNS (Ruíz, 2013).

Bosque: es una colección de dominios dentro de un DA. A efectos se deben reconocer que sea cual sea la cantidad y estructuración de dominios de una organización, todos ellos constituyen un único bosque. Aunque en la organización exista un único dominio, o varios en un único árbol, dicho dominio o árbol constituye por sí solo el bosque de la organización (Terrasa, Ferrer, 2010).

Árbol: es un conjunto de uno o más dominios dentro de un bosque que comparten un espacio de nombres contiguos, es decir comparten un sufijo DNS común (Terrasa, Ferrer, 2010).

Kerberos: es un protocolo de autenticación de red que utiliza una criptografía de llave simétrica para autenticar a los usuarios de los servicios de red (Peñaranda, Avila, 2012).

Servicio de directorio: son almacenes de información acerca de entidades de red, como aplicaciones, archivos, impresoras y usuarios. Estos proporcionan una manera consistente de nombrar, describir, localizar, acceder, administrar y asegurar información acerca de estos recursos. Muchas organizaciones crean almacenes especializados o servicios de directorio dentro de sus aplicaciones para permitir una funcionalidad específica que requieren sus clientes. Los servicios de directorio se manifiestan a través del protocolo *Lightweight Directory Access Protocol (LDAP)* (Desmond, Allen, 2008).

LDAP: 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. Este es el protocolo mediante el cual las aplicaciones acceden para leer o modificar la información existente en la base de datos del directorio (Carter, 2003).

Pluggable Authentication Modules (PAM): *es* un grupo de programas que permiten la autenticación llamados por otros programas como un servicio para delegar la tarea de autenticación (Morgan, 2006).

Name Service Switch (NSS): servicio que permite obtener la información del usuario y del sistema (*passwd, host*, grupo) de diferentes servicios de bases de datos como LDAP (Heslin, 2014).

Después de haber analizado los conceptos fundamentales para entender los principales temas de la investigación; como parte del marco teórico de la investigación se hace necesario realizar un estudio de las herramientas que permiten la integración a un dominio de DA.

1.2: Herramientas para la integración a un dominio de DA

En la actualidad existen varias herramientas que permiten la integración a los dominios de los DA. En el presente documento se exponen las mencionadas en las búsquedas realizadas en Google. Las búsquedas arrojaron además de las herramientas que se mencionan otras que también podrían ser objetivo de investigación como *Vintela Authentication Services o Centrify Express* pero las licencias de estos son privativas y al no poder ser usados de forma libre por los usuarios se omite su investigación. A continuación se muestran las herramientas que fueron investigadas.

PowerBroker Open 6.1

Programa de código abierto anteriormente conocido como *Linkewise Open* que permite integrar plataformas distintas a *Microsoft Windows* con DA, de esta manera se podrá autenticar de forma segura y centralizada con las credenciales de este.

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. Su descarga es libre y se puede utilizar de la misma manera de acuerdo con los términos de la Licencia Pública General GNU[2].

Características:

- Autentica los usuarios con un único nombre de usuario y contraseña en los sistemas no-*Windows*.
- Hace cumplir las políticas de contraseñas a través de sistemas Windows y que no sean *Windows*.
- Caché de credenciales: si se pierde el acceso a la red o el controlador de dominio está caído, se puede seguir trabajando (Power Broken Open Poject, 2014).

***System Security Services Daemon (SSSD)* 1.12.4**

SSSD es un demonio de servicios de sistema de seguridad que ofrece acceso remoto a mecanismos de autenticación e identidad, conocidos como proveedores. Permite a los proveedores conectarse como *backends* SSSD, abstraer las fuentes locales identidad de red y autenticación de red y que cualquier tipo de proveedor de datos de identidad sea conectado.

Demonio[3] cuya función principal es proporcionar acceso a la identidad y la autenticación 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 NSS y PAM para el sistema y también una mejor base de datos para almacenar los usuarios locales, así como de datos extendidos de usuario.

[2] Proyecto iniciado en 1983 por Richard Stallman auxiliado por un número de académicos, programadores voluntarios y formalmente empleados, con el objetivo de crear el primer sistema operativo completamente libre.

[3] 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 (Doc Ubuntu, 2008).

Características:

- Autenticación fuera de línea: cada consulta que se realiza al SSSD queda en caché por un tiempo configurable, sin embargo SSSD nunca invalida el caché cuando el directorio LDAP no está disponible, así que puede trabajar en modo desconectado por tiempo indefinido.
- 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[4], 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.
- Une a los sistemas Linux, Unix y Mac OS para DA en un solo paso a través de una herramienta de interfaz gráfica de usuario o desde la línea de comandos.
- Soporte para varios dominios: SSSD puede utilizar más dominios al mismo tiempo pero al menos uno debe ser configurado o SSSD no se iniciará.
- Obtención de la información de usuarios y grupos Windows NT.
- 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 ronaldo en el dominio de ldap.example.com, del usuario ronaldo en el dominio ldap.myhome.com (Heslin, Pal, 2014).

[4] El módulo nss_ldap es un conjunto de extensiones de biblioteca C que permite a los servidores X.500 y de directorio LDAP ser utilizados como fuente primaria de información del servicio de nombres (PADL, 2014).

Winbind 2.3

Es un componente de la *suite* de programas Samba que resuelve los problemas de inicio de sesión unificados. Provee una solución para el problema de la sesión unificada sin duplicar la información en la máquina Unix y sin crear tareas adicionales de mantenimiento de usuarios y grupos para el administrador. El sistema Winbind proporciona una solución simple y elegante para los tres componentes del problema de la sesión única. Está pensado para organizaciones que tienen una infraestructura existente basada en dominio de *Network (NT[5])* en la cual desean poner estaciones trabajo o servidores de Unix[6].

Características:

- Autenticar usuarios Windows NT.
- Cambio de contraseña para usuarios Windows NT.
- Unifica la gestión de cuentas Unix y Windows NT permitiendo a una máquina Unix volverse un miembro completo de un dominio NT.
- Caché de credenciales: permite el trabajo aunque el controlador de dominio se encuentre caído o se pierda el acceso a la red (Tridgell, 2002).

REALMD 0.16.0

Servicios DBus del sistema, que permiten configurar la autenticación de red y la pertenencia al dominio de una manera estándar. REALMD descubre la información sobre el dominio o dominios de forma automática y no requiere ninguna configuración complicada con el fin de unirse a un dominio o reino.

Características:

- REALMD admite dos tipos de *software* cliente de Active Directory: SSSD y Winbind. Por defecto se utiliza SSSD.

[5] Es una familia de sistemas operativos producidos por Microsoft (Microsoft, 2014).

[6] Es una marca registrada de AT & T Bell Laboratories (Marchall, 1984).

- Como parte de la configuración de un dominio de DA para su uso en el equipo local, REALMD configurará el *software* cliente para habilitar cuentas de dominio para ser usado en el equipo local.
- REALMD se ha integrado en kickstart[7] y en el centro de control de GNOME, y está siendo integrado en otros lugares también (Ballard, Petrová, 2015).

Luego de haber interactuado con cada una de las herramientas antes descritas, se pudieron identificar las funcionalidades que cada una de ellas provee ya sea en cuanto a la restricción del acceso al ordenador a usuarios o grupos del dominio, la inclusión en un dominio existente o incluir usuarios de este dominio al grupo de administradores del ordenador. Por lo que se hace necesario establecer una comparación entre ellas para lograr identificar cuál cumple con los requisitos de la solución que se propone en la investigación. Los parámetros elegidos para esta operación fueron usabilidad que refleja la aceptación del usuario con respecto al trabajo con la herramienta y la funcionabilidad que refleja si la esta ofrece todas las operaciones que necesita el cliente realizar. Otro aspecto a tener en cuenta es si pertenece a licencias de *software* libre o privativo, pero como se menciona anteriormente en el cuerpo del presente documento, para la investigación solo se va a tener en cuenta el uso de las herramientas de licencias libres. A continuación se muestra la Tabla1 que contiene dicha comparación.

Nombre	***Usabilidad***	***Funcionabilidad***
PowerBroker Open(anteriormente Likewise Open)	1. Interfaz pobre. 2. Permite unir ordenadores a dominio de DA pero es necesario apoyarse en el uso de la Terminal de Comandos para lograr otras operaciones.	1. Permite unir y dejar un dominio existente en DA. 2. Mediante el uso de la Terminal de Comandos es posible también denegar o permitir el acceso a usuarios o grupos del dominio al ordenador. 3. No permite incluir usuarios dentro del grupo de administradores.

[7]El archivo Kickstart es un archivo de texto plano que contiene una lista de elementos, cada uno identificado por una clave (Landmann, Cantrell, 2011).

		4. Cuando se incluye un ordenador al dominio en el servidor donde se encuentra el DA , se registra el nombre del ordenador y cuando se deja dicho dominio y se intente incluir nuevamente es necesario cambiar el nombre del cliente ya que la herramienta no lo logra remover del servidor. 5. No permite incluir el ordenador en más de un dominio.
SSSD	1. Integra ordenadores a dominios de DA. 2. No posee ninguna interfaz que controle el proceso, todo se realiza mediante la configuración de ficheros en la Terminal de Comandos.	1. Puede ser configurado en cualquier ordenador con GNU/Linux. 2. Logra realizar un mapeo completo de todos los usuarios y grupos que posee el DA. 3. Permite unir el ordenador a más de un dominio.
Winbind	1. Integra ordenadores a dominios de DA. 2. No posee ninguna interfaz que controle el proceso, todo se realiza mediante la configuración de ficheros en la Terminal de Comandos.	1. Puede ser configurado en cualquier ordenador con GNU/Linux. 2. Soporta múltiples dominios. 3. Provee un inicio de sesión unificado de cuentas de

REALMD	1. Integra fácilmente ordenadores a dominios de DA. 2. Utiliza SSSD para la autenticación. 3. No posee una interfaz visual que gestione todas sus operaciones. Pero haciendo uso de la Terminal de Comandos se entiende fácilmente la operación que se está llevando a cabo.	DA. 1. Puede ser usado en cualquier ordenador con GNU/Linux. 2. Permite realizar todas las operaciones antes descritas en las herramientas anteriores.

Tabla 1: Comparación de herramientas que facilitan la integración con DA.

Como resultado de la comparación realizada entre las herramientas estudiadas se llega a la conclusión, que ninguna de ellas cumple por si solas con los requerimientos de la que se quiere desarrollar. Por lo que para el desarrollo de la solución que da respuesta a la problemática antes expuesta, se utiliza SSSD en su versión 1.12.4 ya que es una extensible alternativa mejorada a Winbind, para adquirir los usuarios y grupos del DA y gestionar la autenticación y REALMD en su versión 0.16.0 para gestionar la unión a los dominios de DA y las políticas de acceso del ordenador. También se encarga de configurar SSSD y que ofrece un servicio *DBus* para el sistema que facilita al programador el desarrollo de herramientas consumiendo sus funcionalidades.

Después de ser analizadas las herramientas que sustentan el estado del arte de la investigación y se definen que elementos se van a tener en cuenta para el desarrollo de la solución a proponer es necesario definir las tecnologías necesarias para su desarrollo.

1.3: Tecnologías para el desarrollo

En el desarrollo de todo sistema informático es de vital importancia la selección de las herramientas, lenguajes y tecnologías a utilizar. A continuación se exponen las principales características de las seleccionadas por el desarrollador.

1.3.1: Herramienta para la gestión de DA

En el presente trabajo se utilizó la herramienta *Manage Your Server* para gestionar y administrar el DA encargado de controlar los recursos de la red que se simuló para apoyar el proceso de desarrollo de la solución.

1.3.2: Metodología de desarrollo

Como metodología de desarrollo se utiliza la metodología ágil OpenUp, ya es la que se encuentra definida por el Departamento de Sistema Operativo (SO) para la realización de las herramientas de apoyo a la Distribución Cubana de GNU/Linux Nova.

Esta metodología constituye una alternativa eficaz para proyectos pequeños y con pocos recursos, en los que se aprecia un desarrollo rápido, iterativo e incremental. Es un proceso ágil y unificado, que contiene el conjunto mínimo de prácticas que ayudan a los equipos a ser más eficaces en el desarrollo de *software*. OpenUP abraza una filosofía pragmática y ágil que se centra en la naturaleza colaborativa de desarrollo de *software*. Es un proceso iterativo que es mínimo, completo y extensible y puede utilizarse tal cual o ampliarse para tratar una amplia variedad de tipos de proyecto. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Está organizada dentro de cuatro áreas principales de contenido: comunicación y colaboración, intención, solución y administración (Yang, Allen, 2015).

En la Imagen 1 se puede observar una imagen que refleja el ciclo de vida de la metodología OpenUp, el tiempo de duración y las fases:

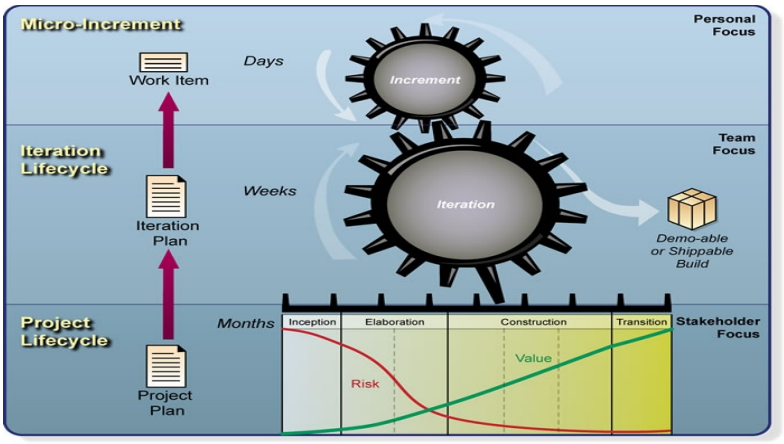


Imagen 1: Metodología Open UP

1.3.3: Herramienta *Computer Aided Software Engineering (CASE)*

Se selecciona como herramienta de modelado Visual Parading en su versión 8.0, porque es la herramienta CASE estándar que utiliza la universidad debido a que tiene una versión libre y evita adquirir gastos en adquisiciones de licencia. Brinda la posibilidad de modelar un sin número de diagramas de clases, facilitando la codificación desde diagramas, la organización y el entendimiento por parte de los desarrolladores. Además, porque contribuye a lograr mayor rapidez en la construcción de aplicaciones informáticas y se encuentra bajo una licencia libre.

Visual Paradigm *for Undefine Model Language (UML)* es una herramienta CASE para modelado UML. Soporta el ciclo de vida completo del desarrollo de *software*, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto y permite control de versiones. También genera informes para una posterior documentación, exporta diagramas como imágenes y posee una alta capacidad de integración con el lenguaje UML que será utilizado durante el modelado de la solución propuesta (Larramendi, 2011).

1.3.4: Lenguaje de modelado

Se selecciona el lenguaje de modelado UML 2.0 ya que es el que se utiliza en la herramienta Visual Paradigm para diseñar y modelar.

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir modelo, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. Incluye conceptos semánticos, notación y principios generales. Se puede aplicar en el desarrollo de *software* entregando gran variedad de formas para dar soporte a una metodología de desarrollo, pero no especifica en sí mismo qué metodología o proceso usar (Fowler, Martin, 1999).

Características:

- El uso de lenguajes visuales permite la asimilación y el entendimiento por parte del equipo de desarrollo.
- Minimiza el tiempo que se emplea en el desarrollo de la arquitectura.
- Ayuda a la detección y resolución de errores.
- Favorece a la organización de la documentación del proyecto (Sierra, 2005).

1.3.5: Lenguaje de programación

El lenguaje de programación seleccionado es Python en su versión 2.7 debido a que es el más conocido y utilizado por el programador, y además como parte de la solución se necesita que la herramienta desarrollada se integre al Panel de Control de la Distribución Cubana de GNU/Linux Nova; el cual es compatible con este lenguaje.

Es un lenguaje potente, muy fácil de entender y de aprender. Posee un enfoque simple pero efectivo a la programación orientada a objetos gracias a su estructura eficiente de datos y de su alto nivel. Es un lenguaje de tipado dinámico, con una sintaxis limpia y elegante que favorece tener un código legible. Su naturaleza interpretada provee que se ejecute gracias a un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje de máquina que pueda entender y ejecutar directamente una computadora, lo cual permite el desarrollo rápido de aplicaciones sobre la mayoría de las plataformas (Van Rossum, 2009).

Sus características más notables son:

- La sencillez y simplicidad de su sintaxis favorece mucho al programador.
- El tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, hacen que desarrollar una aplicación en Python sea sencillo y rápido.
- Es orientado a objetos, pero soporta también los estilos de programación procedural y funcional.
- Corre en múltiples plataformas, incluyendo Windows, Mac OS y Linux.
- Es adecuado tanto para programar scripts como aplicaciones de gran tamaño.
- Permite separar un programa en módulos y luego usarse en otros programas.
- Cuenta con administración automática de memoria a través de recolección de basura.
- Incluye una poderosa y extensa biblioteca de clases.
- Cuenta con una gran comunidad que se dedica a promover su desarrollo y adopción.
- Por su naturaleza interactiva, resulta ideal para llevar a cabo programación experimental y desarrollo rápido.
- Es un lenguaje interpretado lo cual ahorra tiempo durante el desarrollo (Campos, Vargas, 2015).

1.3.6: Plataforma de desarrollo

Como el lenguaje de programación es Python es necesario utilizar como plataforma de desarrollo el *Intergrated Development Evironment* (IDE) Geany en su versión 2.7, debido a que es un entorno fácil de usar y el desarrollador está muy familiarizado con su funcionamiento.

Geany es un pequeño y ligero entorno de desarrollo integrado. Fue desarrollado para proporcionar un pequeño y rápido IDE, que tiene solo unas pocas dependencias de otros paquetes. Es compatible con la mayoría de los lenguajes, soporta auto-completado, coloreado de sintaxis, varios paneles para acceder mejor a los datos, un buscador integrado y la posibilidad de compilar y ejecutar directamente desde el entorno (en todos los lenguajes orientados a esta labor).

Características:

- Auto completado.
- Soporte multi-documento.

- Soporte de proyectos.
- Coloreado de sintaxis.
- Emulador de terminal incrustado.
- Compatible con la mayoría de los lenguajes.
- Varios paneles para acceder mejor a los datos.
- Herramientas para compilar.
- Buscador integrado.
- Posibilidad de compilar y ejecutar directamente desde el entorno (en todos los lenguajes orientados a esta labor).
- Descomposición y representación de las clases y estructuras del código.
- Posibilidad de ampliar funcionalidades mediante complementos (Brush, 2010).

1.3.7: Herramienta para el diseño de la solución

Para el diseño de la herramienta se utiliza el *framework* QT en su versión 4, ya que posee alta compatibilidad con el lenguaje de programación Python y una buena documentación. Tiene un apoyo técnico de alta calidad y sigue el principio de reutilizar el código para crear más y hacer despliegues sin importar el lugar.

Es un marco de trabajo de código abierto con licencia General Public License (GPL) elegido por el desarrollador para crear las interfaces. Permite desarrollar aplicaciones e interfaces multiplataforma. Es una tecnología que proporciona un conjunto de elementos gráficos para la creación de interfaces y aplicaciones en diferentes plataformas. Actualmente cuenta con gran éxito y una gran implantación en diferentes ámbitos, que van desde las aplicaciones de escritorio hasta los sistemas electrónicos industriales. En la última década, Qt pasa de ser un producto usado por unos pocos desarrolladores especializados, a un producto usado por miles de desarrolladores de código abierto en todo el mundo, por lo que el futuro de esta tecnología es hoy día muy prometedor.

Características:

- Compatibilidad multiplataforma con un solo código fuente.
- Disponibilidad del código fuente.
- Diseñador de interfaces gráficas: Qt Designer.
- Soporte para XML y conexión a bases de datos (Lara, 2015).

Para el diseño de las interfaces se utilizó el programa que forma parte del conjunto de programas que ofrece Qt para el marco de trabajo Qt Designer, el cual permite crear la interfaz de la herramienta para luego exportar a código el diseño y de esta forma utilizarlo en el desarrollo de la herramienta. Logrando obtener de manera fácil las vistas de la herramienta, para posteriormente solo editarlas y ajustarlas acorde a los requerimientos del cliente.

Qt Designer 4

Es un programa (parte del conjunto de programas para el desarrollo de aplicaciones para el marco de trabajo Qt) para diseñar y crear interfaces gráficas de usuario usando los componentes de Qt, genera un archivo XML cuyo contenido se puede convertir con los programas pertinentes a múltiples lenguaje de programación.

Características:

- Contiene un depurador visual.
- Resaltado y auto-completado de código.
- Soporta los lenguajes: C#, Java, Python, Perl y *Hypertext Pre-processor (PHP).*
- Exporta interfaces como XML (Gutiérrez, 2009).

1.3.8: *Software* necesarios

Para realizar la tarea de unir el ordenador correctamente al dominio, requisito fundamental de la herramienta se utiliza una compilación realizada por el usuario de REALMD en su versión 0.16.0 para Ubuntu 14.04 basado en la arquitectura de 32bits, debido a que la existente en los repositorios de la universidad presentaba problemas de funcionamiento y no se instalaban correctamente arrojando errores de dependencias o paquetes rotos. Además para gestionar el acceso de los usuarios del dominio al

ordenador se utiliza SSSD en su versión 1.12.4 con la misma arquitectura, también compilado por el desarrollador ya que presentaba el mismo problema que el REALM. Para realizar ambas compilaciones es necesario compilar también ding-libs en su versión 0.4.0, dependencia necesaria para realizar las compilaciones anteriormente mencionadas. Las versiones utilizadas son las últimas liberaciones hasta el momento.

1.4: Conclusiones parciales

En el presente capítulo se estudiaron las diferentes herramientas que permiten integrar ordenadores con GNU/Linux a dominios de DA donde se concluyó que las mismas no satisfacen las necesidades actuales para llevar a cabo satisfactoriamente el proceso de migración a *software* libre en el país. Por tanto se decide desarrollar un *software* capaz de; unir los ordenadores a dominios de DA, gestionar el acceso de los grupos y usuarios del dominio y gestionar los permisos administrativos de los usuarios o grupos del dominio. Para la realización de estas funcionalidades se utilizó REALMD en su versión 0.16.0 y SSSD en su versión 1.12.4. La metodología de desarrollo de *software* que se decide utilizar es OpenUP y como herramienta CASE para el modelado Visual Paradigm. El lenguaje UML para el modelado y Python como lenguaje de programación. Como marco de trabajo y librería gráfica se utiliza Qt teniendo en cuenta que presenta un entorno gráfico ligero.

Capítulo 2: Análisis y Diseño

En el presente capítulo, como establece el proceso de desarrollo de *software*, antes de comenzar a implementar un producto es necesario realizar un correcto análisis y diseño que garantice identificar y documentar los requisitos del mismo, después de tener bien definido que es lo que se quiere hacer. Además se describen los casos de uso, se define la arquitectura del sistema y los patrones de diseño a utilizar.

A continuación se expone la propuesta de solución de la herramienta TodoEnUno, donde se describen las principales funcionalidades que la misma va a poseer y la forma en que se les va a lograr dar solución.

2.1: Propuesta de Solución

La herramienta que lleva por nombre TodoEnUno provee a los especialistas en migración de una herramienta capaz de simplificar el proceso de integración de los ordenadores con la Distribución Cubana de GNU/Linux Nova a los DA. Debe permitir a los usuarios de una entidad unirse al dominio que la misma tenga definido, otorgarle permisos administrativos a grupos o usuarios del dominio y restringir el acceso a usuarios o grupos del dominio. Estas operaciones se logran utilizando las herramientas REALMD 0.16.0 y SSSD 1.12.4, la primera se encarga de gestionar la unión al dominio del DA y de las políticas de acceso, con la segunda se listan los usuarios y grupos del DA y se gestiona la autenticación. Además debe permitir gestionar los permisos administrativos de los usuarios y grupos, tarea que se realiza modificando el fichero sudores del ordenador. La interfaz es fácil de usar por el usuario ya que le permite utilizar todas sus funcionalidades sin complejidad alguna, solo es necesario elegir la funcionalidad que este desea y entrar los datos requeridos.

En la Imagen 2 se muestra el modelo de dominio de la solución antes propuesta, quedando identificadas las principales clases conceptuales que se relacionan en el desarrollo de la misma y la forma en que estas interactúan entre sí.

Modelo del Dominio

Un modelo de dominio se utiliza como fuente de inspiración para el diseño de los objetos del *software*. Muestra a los modeladores clases conceptuales significativas en el dominio del problema. Se representa

como un diagrama de clases o un conjunto de estos. Se utiliza para comprender los conceptos relacionados con el desarrollo y puesta en práctica de la aplicación (Larman, 2003).

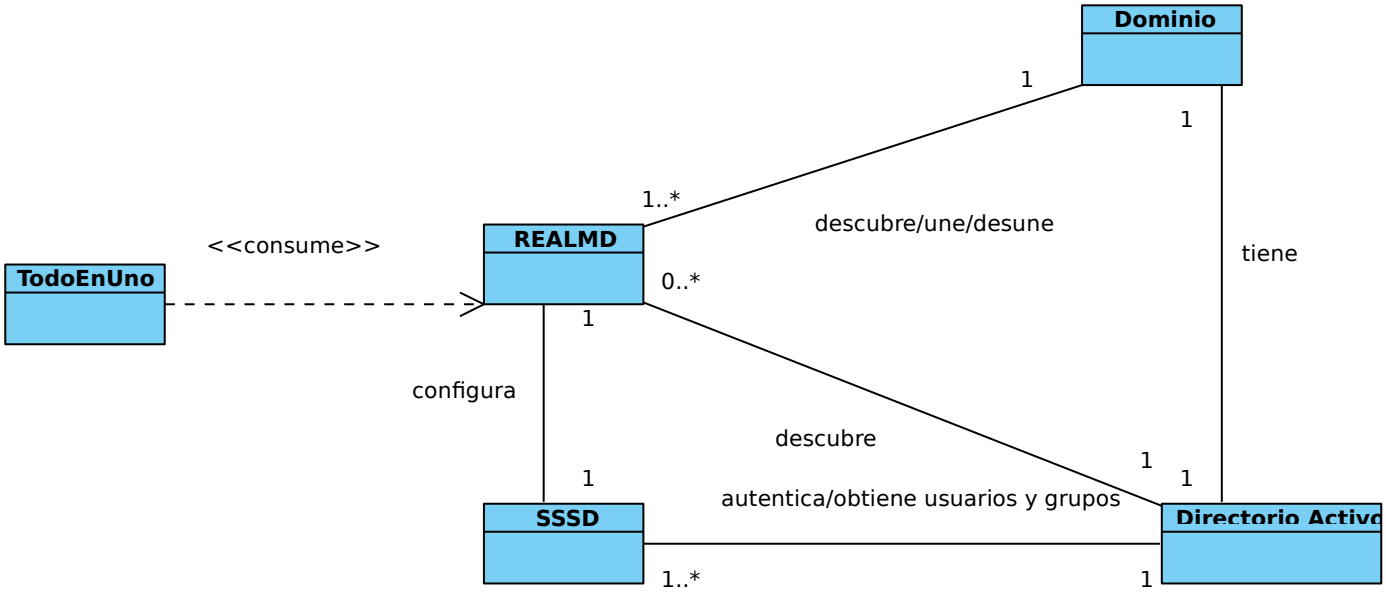


Imagen 2: Modelo de Dominio

A continuación se describen las clases conceptuales del Modelo de Dominio y sus relaciones:

REALMD: herramienta que utiliza TodoEnUno para realizar las operaciones de unir, desunir y gestionar el acceso de usuarios y grupos.

SSSD: herramienta que autentica y obtiene los usuarios y grupos del Directorio Activo.

Directorio activo: contiene el dominio y los usuarios y grupos que a este pertenecen.

Dominio: nombre único definido por la institución que permita acceder a su servidor sin conocer su dirección.

La herramienta TodoEnUno consume el REALMD que se encuentra instalado en el ordenador, el cual se encarga de descubrir los DA los cuales pueden o no existir y los dominios que estos puedan poseer. Además se encarga de las operaciones de unir o desunir a los dominios que descubra y de configurar el SSSD para que este se encargue de obtener los usuarios y grupos del DA y de gestionar la autenticación.

2.2: Características y cualidades del sistema

Las características y cualidades del sistema, constituyen los requisitos funcionales (RF) y no funcionales (RNF) del sistema a implementar. En la solución propuesta se identificaron 8 requisitos funcionales y 4 requisitos no funcionales. Los cuales se muestran a continuación:

2.2.1: Requisitos Funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En estos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Sommerville, 2005).

Para la realización de la implementación del sistema se identifican sus RF por prioridad.

Prioridad Alta

- ✔ **RF_1** Unir Ordenador al dominio.
- ✔ **RF_2** Eliminar Ordenador del dominio.
- ✔ **RF_3** Autenticar contra el dominio.
- ✔ **RF_4** Autenticar contra el dominio fuera de línea.

Prioridad media

- ✔ **RF_5** Restringir el acceso a determinados usuarios o grupos del dominio.
- ✔ **RF_6** Permitir el acceso a determinados usuarios o grupos del dominio.

Prioridad baja

- ✔ **RF_7** Dar permisos de administración a determinados usuarios o grupos del dominio.
- ✔ **RF_8** Retirar permisos de administración a determinados usuarios o grupos del dominio.

2.2.2: Requisitos no Funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2005).

Los requisitos no funcionales identificados para el desarrollo de la herramienta son:

Usabilidad

1. La solución se utiliza fácilmente por cualquier usuario, para el logro de sus funcionalidades solo es necesario acceder a la funcionalidad, proporcionar los datos requeridos y confirmar la operación.

Rendimiento

1. El tiempo de respuesta de la aplicación tiene que ser de 25 segundos, permitiendo al usuario hacer sus operaciones de manera rápida.

Software

1. Sobre la distribución o la plataforma en la que tiene que funcionar es en la Distribución Cubana de GNU/Linux Nova.

Hardware

1. Microprocesador de 1,7 GHz o superior, 1GB mínimo de memoria RAM y una tarjeta de red.

Después de ser identificadas las características y las cualidades que va a tener la solución para la integración de la Distribución Cubana de GNU/Linux Nova con un Dominio de DA, TodoEnUno. Es necesario definir las funcionalidades con las que va a contar el desarrollador para la implementación.

2.3: Funcionalidades de TodoEnUno

Un caso de uso es una secuencia de interacciones que se desarrollan entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los casos de uso pretenden ser herramientas simples para describir el comportamiento del *software* o de los sistemas, no describen ninguna funcionalidad interna (oculta al exterior) del sistema, ni explican cómo se implementarán. Simplemente muestran los pasos que el actor sigue para realizar una tarea (Alarcon, 2000).

El conjunto de características que hacen que TodoEnUno sea práctico y utilitario, se engloban en 4 Casos de Uso (CU) que a continuación se mencionan:

1. -CU1 Gestionar unión al dominio: contiene a los requisitos funcionales RF_1, RF_2.
2. -CU2 Autenticar: contiene a los requisitos funcionales RF_3, RF_4.
3. -CU3 Definir políticas de acceso: contiene a los requisitos funcionales RF5, RF_6.
4. -CU4 Gestionar permisos administrativos: contiene a los requisitos funcionales RF_7, RF_8.

Diagrama de Casos de Uso

Los diagramas de CU sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. El diagrama de CU documenta el comportamiento de un sistema desde el punto de vista del usuario. Representan las funciones que un sistema puede ejecutar (Alarcón, 2000).

La Imagen 4 muestra las funcionalidades de la herramienta a desarrollar y posteriormente se describen las mismas para una mejor comprensión.

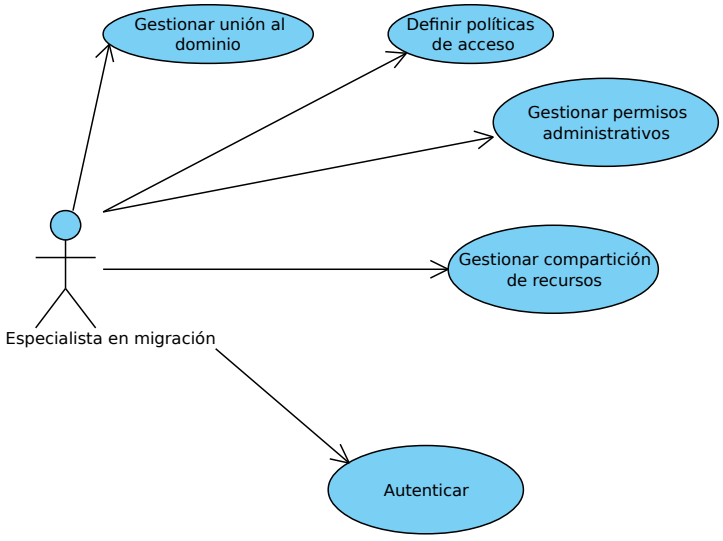


Imagen 3: Diagrama de Casos de Uso del sistema.

El especialista en migración es aquel para el que la herramienta se desarrolla, este accede a la herramienta, para lograr el acceso debe antes autenticarse en el sistema. Es el encargado de realizar todas las operaciones que esta permite y de utilizarla durante la migración; puede gestionar la unión a dominios, definir las políticas de acceso y gestionar los permisos administrativos de usuarios o grupos del dominio.

Descripción de los actores del sistema

Actores	Responsabilidad
Especialista en migración	Es el usuario que se autentica en la herramienta y tiene el máximo de privilegios para realizar todas las operaciones de la herramienta.

Tabla 2: Descripción de los actores del sistema.

Después de ser identificados los CU y los actores del sistema es necesario realizar una descripción de los mismos para comprender su funcionamiento. A continuación se muestran las descripciones de 2 de los más importantes, para consultar los restantes diríjase a las tablas (Tabla5, Tabla6).

Descripción de los casos de uso del sistema

Caso de Uso:	Gestionar unión al dominio
Actores:	Especialista en migración
Resumen:	El caso de uso inicia cuando el Actor presiona la opción “Gestión de Dominio” en el panel de funcionalidades de la herramienta.
Pre-condiciones:	El usuario debe estar autenticado en el sistema.
Flujo básico: Gestionar unión al dominio	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Gestión de Dominio”.	1.1 Muestra una ventana con las opciones: a) Adicionar dominio. Ver sección (a) b) Eliminar del dominio. Ver sección (b)
Sección a: Adicionar dominio	

Flujo básico: Adicionar dominio	
Acción del Actor	**Respuesta del Sistema**
2. El usuario ingresa los datos para unirse al dominio (nombre del dominio, nombre de usuario del dominio y contraseña del mismo).	2.1. El sistema realiza las configuraciones necesarias para unir el ordenador al dominio. 2.2. El sistema muestra al usuario un mensaje anunciando el éxito de la operación.
Flujo Alterno	
2.1 Si no se puede unir al dominio el sistema lanza una notificación al usuario.	
Sección b: Eliminar dominio	
Flujo básico: Eliminar dominio	
Acción del Actor	**Respuesta del Sistema**
3. El usuario ingresa los datos para eliminar del dominio (nombre de usuario del dominio y contraseña del mismo).	3.1 Realiza los cambios necesarios para eliminar el ordenador del dominio. 3.2 Elimina el dominio y muestra un mensaje anunciando el éxito de la operación.
Flujo alterno	
3.1 Si no se elimina el ordenador del dominio se muestra un mensaje	

Tabla 3: Descripción del CU Gestionar unión al dominio.

Caso de Uso:	Autenticar
Actores:	Usuario del SO, Especialista en migración
Resumen:	El caso de uso inicia cuando el Actor ha unido el ordenador al dominio e intenta autenticarse por un usuario del dominio.
Pre-condiciones:	El usuario debe haber unido el ordenador al dominio.
Flujo básico: Autenticar	

Acción del Actor	Respuesta del Sistema
1. El autor une el ordenador al dominio sale de la sesión e intenta autenticarse.	1.1 Permite dos tipos de autenticación: a) Autenticarse online. Ver sección (a) b) Autenticarse offline. Ver sección (b)
Sección a: Autenticarse online	
Flujo básico: Autenticarse online	
Acción del Actor	**Respuesta del Sistema**
2. Introduce datos de autenticación para entrar a la sesión.	2.1 Valida las credenciales introducidas y permite el inicio de sesión.
Flujo Alterno	
2.1 Si el usuario introduce datos inválidos no puede acceder a la sesión.	
Sección b: Autenticarse offline	
Flujo básico: Eliminar del dominio	
Acción del Actor	**Respuesta del Sistema**
3. Verifica que el ordenador no se encuentre conectado a la red e introduce las credenciales.	3.1 Valida las credenciales introducidas y permite el inicio de sesión.
Flujo alterno	
3.1 Si el usuario introduce datos inválidos o SSSD no se encuentra correctamente configurado no puede acceder a la sesión	

Tabla 3: Descripción del CU Autenticar.

2.4: Arquitectura de TodoEnUno

La arquitectura de un *software* es la representación de la estructura de los componentes del mismo, sus propiedades e interacciones. Permite la comunicación entre todas las partes participantes en el desarrollo de un sistema de cómputo. Destaca las actividades relacionadas con el diseño y que a su vez tendrán un gran impacto durante todo el trabajo de ingeniería y sobre todo en el resultado final. Modela cómo se estructura el sistema y la relación existente entre sus componentes (Valencia, Ferro, 2011).

Todo *software* creado para sistemas de cómputo muestra uno o varios estilos arquitectónicos. Cada estilo describe una categoría de sistema que abarca un conjunto de componentes encargados de realizar alguna función requerida por el sistema, y muestra además cómo se realiza la integración o comunicación entre dichos componentes, permitiendo al diseñador las propiedades generales del sistema (Pressman, 2005).

Arquitectura

Se elige para el desarrollo de la herramienta una variante de la arquitectura N-Capas. Este patrón arquitectónico distingue diferentes capas internas de una aplicación, delimitando la situación de los diferentes componentes por su tipología. Esta arquitectura concreta es personalizable según las necesidades de cada proyecto y preferencias de Arquitectura (Llorente, Plain, 2010).

A continuación se presenta la Imagen 4 que representa los componentes de la variante de la arquitectura diseñada para el sistema a desarrollar esta arquitectura:

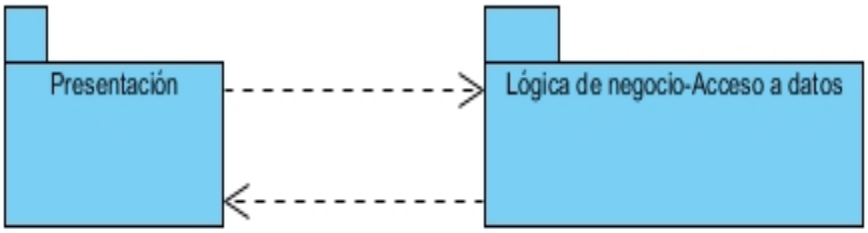


Imagen 4: Representación de las Capas de la variante utilizada

Presentación: esta capa contiene las interfaces externas de la herramienta.

Lógica de negocio- Acceso a datos: esta capa contiene las clases que definen las funcionalidades de la herramienta, ellas se encargan del manejo de las vistas y de utilizar métodos implementados en otras clases para modificar los ficheros sudores y sssd.conf.

Para conocer como es utilizada esta arquitectura en el desarrollo de la herramienta es necesario comprender el funcionamiento de la misma, a continuación se realiza una descripción de dicho funcionamiento:

Mediante los componentes de la capa **Presentación** el usuario introduce los datos que son necesarios para realizar las funcionalidades de la herramienta, enviando estos a la capa **Lógica de negocio- Acceso a datos.** Esta capa se encarga de procesar las funcionalidades , de llamar las funcionalidades implementadas que se encargan de dar solución a la seleccionada por el usuario y de dar respuesta a los usuarios modificando las vistas.

Estilo arquitectónico

Todo el diseño arquitectónico antes presentado a pesar de sus particularidades por las características propias del proyecto responde al estilo arquitectónico en capas. Pressman define que todo lo que se divide lógicamente por capas es un estilo en capas (Pressman, 2010b).

2.5: Diseño de la herramienta TodoEnUno

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. 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*. Si bien es cierto que desde el código también se pueden definir los elementos estructurales y de comportamiento, es mucho más complicado tratar de afrontarlos a través de esta vía. Los diseños pueden ser modelos visuales, bosquejos simples y descripciones tipo texto. En general un buen diseño describe claramente cómo los diferentes elementos del sistema interaccionan para cubrir los requisitos (Pressman, 2010).

Diagrama de clases de TodoEnUno

En la Imagen 5 puede apreciarse la distribución de las clases que van a componer la herramienta, el diagrama se encuentra basado en la arquitectura del sistema por lo que cada clase ha sido englobada en la capa que pertenece:

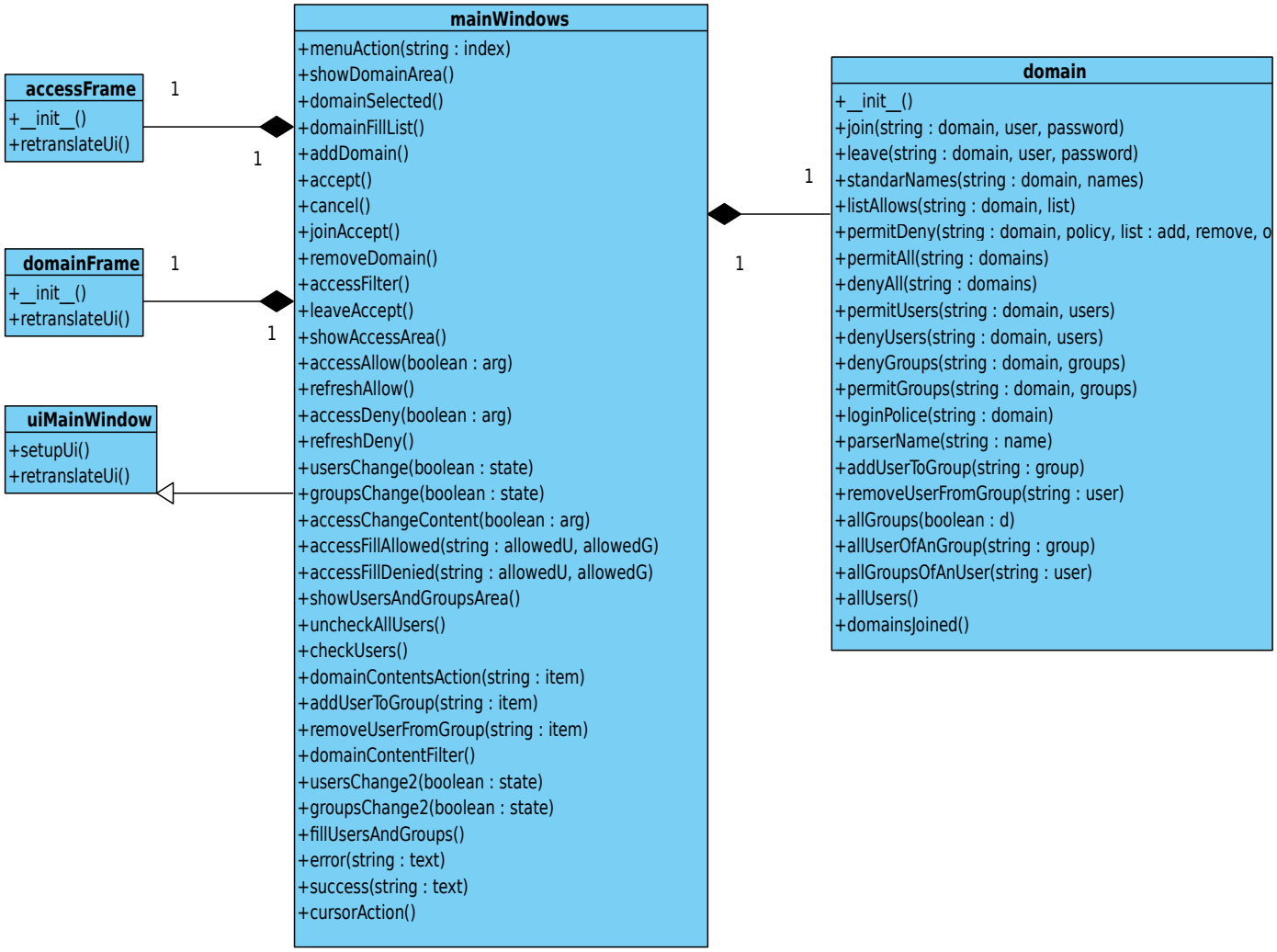


Imagen 5: Diagrama de clases del sistema.

mainWindow: es la clase que se encarga de manejar toda la lógica de la herramienta.

domainFeame: vista que maneja la unión a los dominios.

Accesframe: clase que controla la vista del panel de opciones.

uiMainWindow: clase que contiene la interfaz inicial de la herramienta.

domain: clase que contiene todos los métodos que realizan todas las funciones que se requieren de la herramienta.

La clase mainWindow se encarga de manejar las vistas que se van a mostrar en la herramienta de acorde a la funcionalidad que sea requerida, las vistas van a estar alojadas en las clases domainFrame, uiMainWindow y accesFrame y la implementación de todas las funcionalidades que la herramienta posee se encuentra en la clase dominio. En la clase mainWindow se crea un objeto de la clase dominio, para usar todos los métodos implementados en ella y controlar el flujo de las acciones.

Patrones de diseño usados

Los patrones de diseño se utilizan para describir la solución de un problema particular dentro de un contexto determinado mediante una estructura de diseño. Cada patrón describe un problema que ocurre una y otra vez en el entorno, y después describe la esencia de la solución a dicho problema, de tal forma que se pueda usar esta solución un millón de veces más, sin nunca hacerlo dos veces de la misma forma (Pressman, 2005).

Para el desarrollo de la herramienta es necesario hacer uso de los patrones de diseño que a continuación se exponen.

Patrones Generales de *Software* para la Asignación de Responsabilidades (GRASP)

Este nombre surgió para indicar la trascendencia de captar estos principios, si se quiere diseñar un *software*. Estos patrones son los encargados de describir los principios fundamentales de la asignación de responsabilidades (Boteros, 2011).

Los patrones GRASP que se emplean en el desarrollo de la herramienta son el Creador, el Controlador y el Observador. El Creador se utiliza para definir en el desarrollo de la herramienta la jerarquía de clases, el

Controlador se utiliza para asignar a una clase la responsabilidad de realizar todas las funcionalidades de la herramienta y de manejar el flujo de la misma y el Observer se utilizará para identificar dependencias entre clases. A continuación se describe cada uno de ellos y se expone un ejemplo de cómo es utilizado en el desarrollo de la herramienta.

- **Creador:** el patrón Creador soluciona el problema de seleccionar cual clase debe ser la encargada de instanciar a otra. Propicia el principio para la creación de objetos. Es el que crea y guía la asignación de responsabilidades relacionadas con la creación de objetos, encontrando un creador que debemos encadenar con el objeto originado en cualquier evento (Pressman, 2005).

En el diseño de la herramienta se puede apreciar el uso de este patrón. Un ejemplo de ello se evidencia cuando la clase **mainWindow** que es la contenedora de todas las funcionalidades de la aplicación TodoEnUno instancia a la clase dominio.

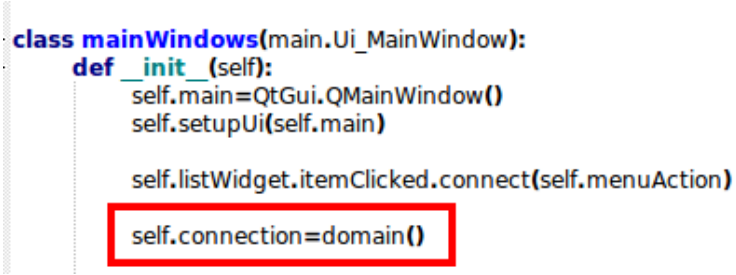


Imagen 6: Ejemplo de código aplicando el patrón creador.

- **Controlador:** un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Consiste en asignar la responsabilidad del manejo de los eventos de un sistema a una única clase (Larman, 2003).

En la solución propuesta se evidencia en la clase **mainWindow**, la cual se encarga de gestionar los procesos en la herramienta.

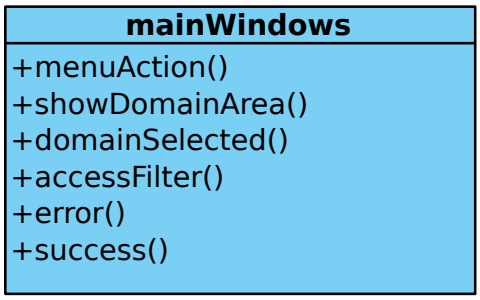


Imagen 7: Ejemplo representativo del patrón controlador

- ***Observador*:** permite a los objetos captar dinámicamente las dependencias entre objetos, de tal forma que un objeto notifica a los objetos dependientes a él cuando cambia su estado, siendo actualizados automáticamente (Martínez, 2000)**.**

En la solución propuesta, este patrón se utiliza al importar en la clase mainWindow clases que se encuentran en otro directorio diferente al suyo, cualquier cambio en alguna de estas clases es actualizado en la clase mainWindow.

```
from ui import main
from ui.domain import domainFrame
from ui.access import accessFrame
```

Imagen 8: Ejemplo de código usando el patrón observer

Patrones GoF

Los patrones “Banda de los Cuatro” (GoF, por sus iniciales en inglés *Gang of Four*), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Existen tres tipos de patrones: los de **creación** que abstraen el proceso de creación de instancias, **estructurales** que se ocupan de verificar como las clases y objetos son utilizados para componer

estructuras de mayor tamaño y de **comportamiento** que se refieren a los algoritmos y a la asignación de responsabilidades entre objetos (Gamma, Helm, 1995).

- **Decorador:** es un patrón de tipo estructural encargado de asociar responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Brinda una mayor flexibilidad que la herencia estática, permitiendo añadir una funcionalidad dos o más veces. Propicia concentrar en lo alto de la jerarquía de clases guiadas por las responsabilidades. De esta forma las nuevas funcionalidades se componen de piezas simples que se crean y se combinan con facilidad, independientemente de los objetos cuyo comportamiento extienden (Wesley, 2006).

El uso de este patrón en la aplicación se evidencia en el uso del decorador **cursorAction** el cual es el encargado de poner el cursor en modo de Cargando y retornarlo al modo normal cuando se ejecutan métodos que requieren algún tiempo de procesamiento visible al usuario.

```
def cursorAction():
    def decorador(fun):
        def interna(self,*arg):
            self.main.setCursor(QtCore.Qt.WaitCursor)
            try:
                fun(self,*arg)
            except Exception, err:
                self.error(err)
            self.main.setCursor(QtCore.Qt.ArrowCursor)
        return interna
    return decorador
```

Imagen 9: Ejemplo del patron decorador

El uso de los patrones GRASP y Gof en el diseño del sistema permite al desarrollador obtener una herramienta con el código debidamente estructurado permitiendo futuros cambios y facilitando la comprensión de otros desarrolladores.

2.6: Conclusiones parciales

En este capítulo se identifican los requisitos del sistema, 8 funcionales y 4 no funcionales que fueron agrupados en 4 CU, **CU Gestionar unión al dominio**, **CU Autenticar**, **CU Definir políticas de acceso** y **CU Gestionar permisos administrativos**. Además se define una variación de la arquitectura N-Capas para la construcción de la solución basada en el estilo arquitectónico en capas y se seleccionan los patrones de diseño GRASP Controlador, Observador y Creador y el patrón GoF Decorador.

Capítulo 3: Implementación y prueba

En el presente capítulo se analizan los elementos relacionados con la implementación del sistema y las pruebas correspondientes que se realizan para evaluar su funcionamiento. Se muestran además elementos referentes a los estándares del código utilizado en el desarrollo del sistema. Se analiza el resultado de las pruebas para verificar el correcto funcionamiento de la herramienta y se expone el impacto que va a causar su implementación.

3.1: Implementación

Partiendo de los CU descritos anteriormente en el Capítulo 2 se realiza la implementación de cada una de las funcionalidades definidas cuyo objetivo se encamina a desarrollar de forma iterativa e incremental un producto completo, listo para ser utilizado. Para la obtención del resultado esperado, se planificaron 3 iteraciones, una primera con duración de 5 semanas donde el desarrollador obtuvo los métodos referentes a las funcionalidades que requiere la herramienta, una segunda iteración con duración de 4 semanas más donde se realiza la interfaz visual de la funcionalidad Gestionar acceso al dominio. Finalmente en la tercera iteración se añade a la interfaz la segunda funcionalidad referente a las políticas de acceso y la funcionalidad que permite gestionar los permisos administrativos. Además se realiza una mejora del diseño. A continuación se muestran estas descripciones detalladamente:

Iteración	Descripción	CU Implementado	Duración
1	- Desarrollo de todas las funcionalidades que van a componer el sistema.	CU1, CU2, CU3, CU4.	5 semanas
2	-Desarrollo de la interfaz visual con la funcionalidad de Gestionar acceso al dominio.	CU1, CU2	4 semanas
3	-Se añade a la interfaz la segunda funcionalidad referente a las políticas de acceso. -Se añade a la interfaz la funcionalidad de gestionar los permisos administrativos. -Se realiza una mejora del diseño.	CU3, CU4.	4 semanas

	- Se realiza mejora al diseño.		

Tabla 7: Planificación de la implementación por iteraciones de TodoEnUno.

Diríjase a la sección de los anexos para observar en las imágenes (**Imagen 14 ,** Imagen 15, Imagen 16, Imagen17) los resultados obtenidos en las iteraciones antes mencionadas.

3.1.1: Diagrama de componentes

Un Diagrama de Componentes representa como un sistema de *software* es dividido en componentes y muestra las dependencias entre estos. Los Diagramas de Componentes prevalecen en el campo de la arquitectura de *software* pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. Estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes (Roger S. Pressman, 2005).

A continuación se muestra el Diagrama de Componentes de la herramienta TodoEnUno, donde queda reflejada la forma en que el proyecto se encuentra estructurado.

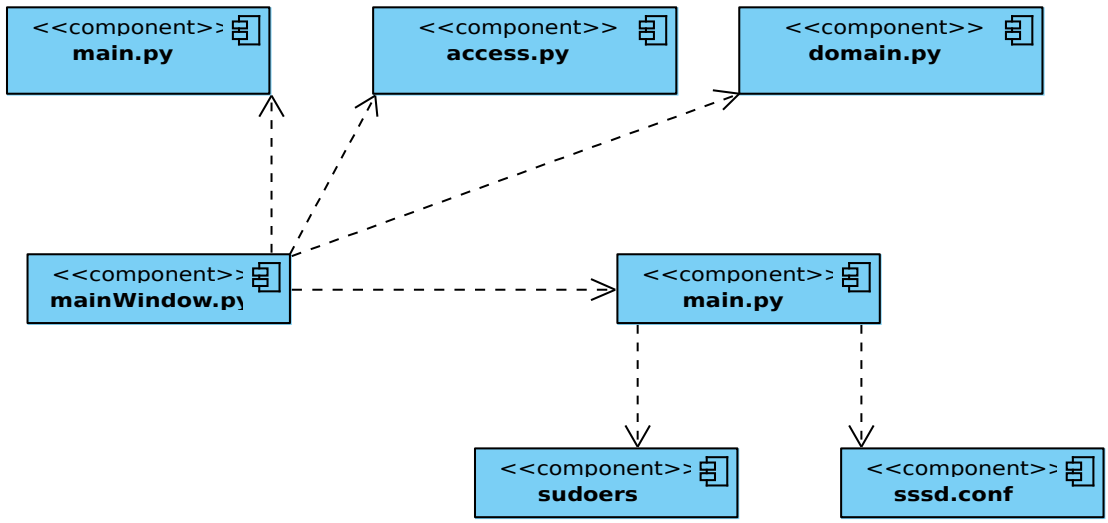


Imagen 10: Diagrama de componentes de TodoEnUno

El componente **mainWindow.py** es el encargado de gestionar las vistas de la herramienta y de recibir los datos que el usuario envía a través de ellas para la realización de las funcionalidades seleccionadas. Para gestionar las vistas este se relaciona con los componentes **mainWindow.py, access.py y domain.py** y para lograr dar solución las funcionalidades utiliza los métodos implementados en el componente main.py, dentro del cual se encuentran aquellos que modifican los ficheros sudoers y sssd.con.

Al comenzar un proyecto de *software*, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del *software* y para obtener un buen rendimiento. Además, si se aplica de forma continua un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y posteriormente, se efectúan revisiones de rutinas, caben muchas posibilidades de que un proyecto de *software* se convierta en un sistema fácil de comprender y de mantener.

3.1.2: Estándar de Codificación

Una de las buenas prácticas utilizadas en la construcción de la solución planteada lo constituye la adopción del estándar de codificación ***CamelCase***, asegurando que el código exprese claramente el propósito del mismo y agilice el proceso de refactorización, que sea legible, entendible y refleje un estilo armonioso. Se elige por ser el utilizado por el desarrollador desde sus inicios en la materia del desarrollo de *software* y para mantener el código de la herramienta organizado.

CamelCase es un término que se refiere a la práctica de escribir palabras compuestas en la primera letra del identificador en minúscula y la primera letra de cada palabra concatenada subsiguiente se escribe con mayúscula. Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en *CamelCase* se asemejan a las jorobas de un camello. El nombre *CamelCase* se podría traducir como Mayúsculas/Minúsculas Camello. El término *case* se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula y tiene su origen en la disposición de los tipos móviles en casilleros o cajas (IEEE, 2013).

A continuación se muestra como se utiliza el estándar definido para organizar el código de la solución propuesta, TodoEnUno.

```
def showDomainArea(self):
    self.frame = domainFrame(self.centralwidget)
    self.frame.show()
    self.frame.pushButton.clicked.connect(self.addDomain)
    self.frame.pushButton_2.clicked.connect(self.removeDomain)
    self.frame.listWidget.itemSelectionChanged.connect(self.domainSelected)
    self.domainFillList()
```

Imagen 11: Ejemplo de código utilizando el estándar Camelcase

Los ficheros que contienen el código de la herramienta se encuentran estructurados siguiendo las pautas que a continuación se mencionan y ejemplifican.

Organización de los ficheros: Se evitan los ficheros de gran tamaño que contengan más de 1000 líneas de código, en ocasiones el tamaño excesivo provoca que la clase no encapsule un comportamiento claramente definido, albergando una gran cantidad de métodos que realizan tareas funcional o conceptualmente heterogéneas.

```
def listAllows(self,domain,list):
        """list=PermittedLogins for users, PermittedGroups for groups"""
        try:
                realm=self.auth(domain)
        except Exception, err:
                raise RuntimeError(str(err))
        realmProps = dbus.Interface(realm, FREEDESKTOP_IFACE_PROPERTIES)
        return [ str(i).split('@')[0] for i in realmProps.Get(REALMD_IFACE_REALM, list)]
```

Imagen 12: Ejemplo de código demostrando la organización de código en los ficheros

Estructuras de control: Las estructuras de control utilizan siempre llaves de apertura y cierre “{}”, incluso en situaciones en las que técnicamente son opcionales. Con esto se aumenta la legibilidad del código y se disminuye la probabilidad de errores lógicos. Incluyen *if, for, foreach, while y switch.*

```
if not self.frame.treeWidget_2.selectedItems():
        self.error('Debe selecccionar un usuario/grupo para permitir')
        return False
for i in self.frame.treeWidget_2.selectedItems():
        if i.whatsThis(0)=='usuario' :
                moveU.append(str(i.text(0).toUtf8().data()))
        else:
                moveG.append(str(i.text(0).toUtf8().data()))
try:
        domain=str(self.frame.comboBox.currentText().toUtf8().data())
        self.connection.permitUsers(domain,moveU)
        self.connection.permitGroups(domain,moveG)
```

Imagen 13: Ejemplo de código de las estructuras If y for

Al finalizar la implementación de todas las funcionalidades definidas en las iteraciones de la planificación para el desarrollo de la herramienta es necesario realizar pruebas sobre esta para validar su correcto funcionamiento.

3.2: Pruebas *software*

Las pruebas de *software* son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia y el desempeño de un *software*. Involucra las operaciones del sistema bajo condiciones controladas y evalúa los resultados. Las técnicas para encontrar problemas en un programa son extensamente variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad. Una vez completada la implementación, es necesario probar el *software* con el objetivo de descubrir y corregir la mayor cantidad de errores posibles antes de entregárselo al cliente (Pressman 2005).

La metodología OpenUP define las pruebas funcionales basadas en casos de prueba, razón por la cual una vez finalizada la fase de implementación se llevan a cabo las pruebas para validar su correcto funcionamiento, siguiendo el **método de caja negra** ya que estas se enfocan a probar la interfaz de la herramienta. El tipo de prueba a seguir seleccionado es las Pruebas de Aceptación, sirviendo para validar si el comportamiento observado del *software* cumple o no con sus especificaciones llevadas a cabo por el cliente.

Pruebas de Caja Negra

Se define utilizar el método de prueba de caja negra ya que son las adecuadas a utilizar cuando se conoce la función específica para la que se diseñó el producto; y consiste en aplicar pruebas que demuestren que cada función es plenamente operacional, mientras que se buscan los errores de cada función. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Los casos de prueba son escenarios donde se espera un resultado, se introducen los datos a validar y se obtiene el resultado de las mismas. Son representaciones de las pruebas para guiar al probador.

Tipo de Prueba

Las Pruebas de Aceptación son pruebas funcionales, pero basadas en el cliente de la aplicación, en aras de demostrar al cliente que la funcionalidad se encuentra terminada y funciona correctamente.

Casos de prueba de Aceptación

En correspondencia con los casos de uso definidos se diseñaron los casos de prueba al **CU Gestionar Unión al Dominio** y al **CU Definir Políticas de Acceso** por ser las principales funcionalidades del sistema**,** basados en escenarios como se observa en las tablas 8 y 9 que se muestran a continuación:

Pruebas de Aceptación al CU Gestionar Unión al Dominio.

Id del Escenario	Escenario	Descripción	Resultado esperado	Respuesta del Sistema	Resultado de la Prueba
EC1	Escenario 1: Entrar al apartado perteneciente a la gestión de dominio.	El usuario inicia la aplicación y en el panel de funcionalidades oprime la nombrada Gestionar Dominio.	El sistema debe mostrar una lista con los dominios a	Muestra una lista de los dominios en blanco.	No satisfactoria

			los que se encuentra unido.		
EC2	Escenario 2: Petición de Credenciales.	El usuario selecciona la opción de añadir nuevo dominio.	El sistema debe pedir credenciales de autenticación y nombre del dominio a añadir.	Muestra una ventana pidiendo credenciales de autenticación y nombre del dominio a añadir.	Satisfactoria
EC3	Escenario 3: Dejar dominio sin credenciales.	El usuario selecciona un dominio para dejar.	El sistema debe pedir credenciales de autenticación y nombre del dominio a añadir.	Muestra una ventana pidiendo credenciales de autenticación.	Satisfactoria
EC4	Escenario 4: Introducir nombre del dominio falso para unirse a él.	El usuario selecciona la opción de añadir un dominio y entra un dominio que no existe.	El sistema debe lanzar una notificación informando que el dominio entrado no existe.	No sucede nada	No satisfactoria
EC5	Escenario 4: Introducir	El usuario selecciona la	El sistema	No muestra la	No satisfactoria

	nombre del dominio falso al querer dejar un dominio	opción de dejar un dominio y entra un dominio que no existe.	debe lanzar una notificación informando que el dominio entrado no existe.	notificación.	

Tabla 8: Casos de prueba al CU Gestionar Unión al dominio.

Pruebas de Aceptación al CU Definir Políticas de Acceso

Id del Escenario	Escenario	Descripción	Resultado esperado	Respuesta del Sistema	Resultado de la Prueba
EC1	Escenario 1: Entrar al apartado perteneciente a la Definición de las políticas de acceso.	El usuario inicia la aplicación y en el panel de funcionalidades oprime la nombrada Políticas de Acceso y adiciona un usuario o grupo a la lista de permitidos.	El sistema debe mostrar un mensaje anunciando que el acceso ha sido garantizado.	Muestra un mensaje anunciando que el acceso ha sido garantizado.	Ortografía
EC2	Escenario 2: Entrar al apartado perteneciente a la Definición de las políticas de acceso.	El usuario inicia la aplicación y en el panel de funcionalidades oprime la nombrada Políticas de Acceso y adiciona un usuario o grupo a la lista de denegados.	El sistema debe denegar el acceso al usuario o grupo seleccionado y mostrar un mensaje anunciando que el acceso ha sido denegado.	Muestra un mensaje anunciando que el acceso ha sido denegado.	Satisfactoria

EC3	Escenario 3: Añadir más de un usuario y grupo a la lista de permitidos.	El usuario inicia la aplicación y en el panel de funcionalidades oprime la nombrada Políticas de Acceso, selecciona más de un elemento en la lista de denegados para adicionarla a la de permitidos.	El sistema debe garantizar el acceso al usuario o grupo seleccionado y mostrar un mensaje anunciando que el acceso ha sido garantizado.	No muestra la notificación esperada, sin embargo garantiza el acceso al usuario o grupo.	No Satisfactoria
EC4	Escenario 4: Filtrar usuarios.	El usuario inicia la aplicación y en el panel de funcionalidades oprime la nombrada Políticas de Acceso y entran datos al filtro.	El sistema debe mostrar los usuarios que contienen nombres con los elementos entrados.	No muestra usuarios.	No satisfactoria
	Escenario 4: Mostrar solo usuarios.	El usuario inicia la aplicación en el panel de funcionalidades oprime la nombrada Políticas de Acceso y selecciona la opción que permite solo mostrar usuarios.	El sistema solo debe mostrar en las listas de permitidos y denegados los usuarios.	Muestra usuarios y grupos	No satisfactoria

Tabla 9: Casos de prueba al CU Gestionar Políticas de Acceso.

3.2.1: Resultados de las pruebas realizadas

Las Pruebas de Aceptación fueron realizadas en dos iteraciones, llevadas a cabo luego de concluido el desarrollo de la herramienta. Al finalizar la primera iteración se obtuvo como resultado un total 7 No Conformidades (NC), de ellas 3 fueron clasificadas de Significativas (S), 4 de No Significativas (NS). La segunda iteración finalizó con 4 NC, de ellas 3 de tipo S, 1 de tipo NS. Ya en la iteración final no se obtuvo

ninguna NC, lo que demuestra que el *software* funciona correctamente y que se encuentra listo para ser utilizado. La explicación anterior se puede ver reflejada en el Diagrama 1.

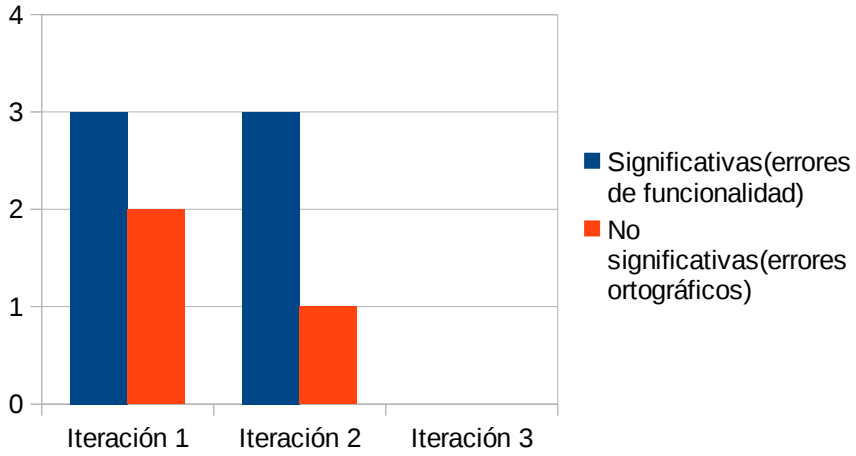


Diagrama 1: Resultado de las Pruebas de Aceptación

3.3: Aportes de la herramienta Todo En Uno

TodoEnUno, va a ser la herramienta de apoyo para los especialistas en migración durante el desarrollo del proceso que se lleva a cabo en el país. Con el desarrollo de la misma se agrega una aplicación más a las desarrolladas para la Distribución Cubana de GNU/Linux Nova para lograr que sus usuarios ya no queden aislados a los de *Windows*, debido a la existencia de una herramienta para integrar ordenadores con Nova a redes creadas en estos. Mediante su uso, TodoEnUno simplifica las tareas a desarrollar durante el proceso de migración que anteriormente se realizaban sin orden alguno, utilizando varias tecnologías. Esta herramienta evita que el especialista en migración necesite poseer amplios conocimientos para lograr integrar un ordenador a un dominio de DA, realizando esta operación de forma sencilla con solo ejecutar la funcionalidad y entrando los datos requeridos. Permite además gestionar los privilegios administrativos sin necesidad de acceder a ficheros de configuración del sistema o ejecutar comandos en la Terminal. El desarrollo de esta herramienta se realiza sin gasto alguno y no es necesario adquirir licencias para su uso debido a que se le otorgan los derechos de autoría a la UCI, ahorrando los gastos monetarios al país.

3.4: Conclusiones parciales

En este capítulo se describen las 3 iteraciones que fue necesario realizar hasta llegar a la solución óptima. Se define el estándar de codificación CamelCase, lo que permitió desarrollar un código entendible por parte del equipo de desarrollo para su futura modificación o estudio. Además se decide utilizar las pruebas de Aceptación para verificar el correcto funcionamiento de la herramienta las cuales arrojaron un total de 11 NC, de ellas 6 de tipo S y 5 de tipo NS, siendo todas resueltas. Esto garantiza que la versión implementada en la iteración final cumpla con las necesidades del cliente, garantizando así la calidad y efectividad.

Conclusiones Generales

A partir de los objetivos planteados y el trabajo realizado en esta investigación donde se desarrolló una herramienta para simplificar el proceso de integración de la Distribución Cubana de GNU/Linux Nova a dominios de DA se arribó a los siguientes resultados:

1. Se realizó un estudio detallado de las aplicaciones utilizadas para la integración de ordenadores con la Distribución Cubana de GNU/Linux Nova a dominios de DA, así como sus características y funcionamiento.
2. El establecimiento de OpenUP como metodología para el desarrollo de la aplicación y el uso de las tecnologías seleccionadas; permitieron analizar, describir e implementar las funcionalidades definidas, materializando así una solución acorde a los requerimientos del cliente.
3. Las pruebas realizadas a la solución demostraron que la misma está lista para ser usada en un entorno de trabajo real.
4. La implementación de la herramienta TodoEnUno, permitió que el proceso de integración a dominios existentes en un DA fuera simplificado.

Recomendaciones

Se recomienda:

- Traducir la herramienta a otros idiomas para lograr su internacionalización.
- Adicionar a la herramienta la funcionalidad de compartir recursos como carpetas e impresoras a usuarios o grupos del dominio.
- Incluir la herramienta al Panel de Control de la Distribución Cubana de GNU/Linux Nova.
- Implementar la ayuda de la herramienta.

Referencias Bibliográficas

[1]. Alarcon, 2000. *Diseño orientado a objetos con UML*. S.l.: s.n.

[2]. Alegsa, 2015. ¿Cuál es la definición de Dominio? [en línea]. [Consulta: 21 mayo 2015]. Disponible en: http://www.alegsa.com.ar/Dic/dominio.php.

[3]. Ballard, Petrová, 2015. *Red Hat Enterprise linux 7 Windows Integration Guide*. 2015. S.l.: s.n.

[4]. Boteros, 2011. *Patrones GRASP y anti-patrones: Un enfoque orientado a objetos desde la lógica de programación*. 2011. S.l.: s.n.

[5]. Brush, 2010. Geany : About. [en línea]. [Consulta: 12 mayo 2015]. Disponible en: http://www.geany.org/Main/About.

[6]. Campos y Vargas, 2015. *Revista de Ciencias, Vol. 6, No. 1*. 2015. S.l.: s.n.

[7]. Carter, 2003. *LDAP System Administration*. S.l.: s.n.

[8]. Chico, 2011. *Estudio de Viabilidad de Directorio Activo en Linux*. 2011. S.l.: s.n.

[9]. Desmond, Allen, 2008. *Active Directory, 4th Edition*. S.l.: s.n.

[10]. Doc Ubuntu, 2008. Daemon - doc.ubuntu-es. [en línea]. [Consulta: 21 mayo 2015]. Disponible en: http://doc.ubuntu-es.org/Daemon.

[11]. Fowler, Martin, 1999. *UML Gota a Gota*. 1999. S.l.: s.n.

[12]. Gutierrez, 2009. *Tutorial de QtDesigner y QtDevelop*. 2009. S.l.: s.n.

[13]. Heslin, 2014. *Integrating Red Hat Enterprise Linux 6 with Active Directory*. 2014. S.l.: s.n.

[14]. Heslin, Pal, 2014. *Interoperability Update: Red Hat Enterprise Linux 7 beta and Microsoft Windows*. 2014. S.l.: s.n.

[15]. Hull, 2005. *Requirements Engineering Second Edition*. S.l.: s.n.

[16]. IEEE, 2013. User Evaluations for *Software* Engineering Researchers (USER). *2nd International Workshop* [en línea]. San Francisco, CA: s.n., pp. 13 - 15. Disponible en: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6603079&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6603079. 13778784

[17]. Landmann, Cantrell, 2011. *Guía de instalación de Red Hat Enterprise*. 2011. S.l.: s.n.

[18]. Lara, 2015. Programando en QT I: primera GUI | AlvaroLara.com. [en línea]. [Consulta: 4 junio 2015]. Disponible en: http://www.alvarolara.com/2012/06/10/programando-en-qt-i-primera-gui/.

[19]. Larman, 2003. *UML y Patrones Introducción al análisis y diseño orientado a objetos. Segunda edición*. S.l.: s.n.

[20]. Larramendi, 2011. *Plan de desarrollo de la Ingeniería de Requisitos del Proyecto de Informatización de los TPC*. 2011. S.l.: s.n.

[21]. Liu, Albitz, 2000. *OReilly DNS and BIND*. S.l.: s.n.

[22]. LLorente, Plain, 2010. *Guia_Arquitectura_NCapas DDD NET 4*. 2010. S.l.: s.n.

[23]. Marshall, 1984. *A Fast File System for UNIX*. 1984. S.l.: s.n.

[24]. Microsoft, 2005. Controladores de dominio. [en línea]. [Consulta: 29 abril 2015]. Disponible en: https://msdn.microsoft.com/es-es/library/cc759623(v=ws.10).aspx.

[25]. Microsoft, 2014. Windows NT 4. [en línea]. [Consulta: 22 mayo 2015]. Disponible en: http://toastytech.com/guis/nt4.html.

[26]. Morgan, 2006. *PAM Pluggable Authentication Modules*. 2006. S.l.: s.n.

[27]. PADL 2014. nss_ldap. [en línea]. [Consulta: 22 mayo 2015]. Disponible en: http://www.padl.com/OSS/nss_ldap.html.

[28]. Paseur, Sarria, 2010. *Plataforma de Gestión de Servicios Telemáticos en GNU/Linux. Módulo de Directorio v2.0*. 2010. S.l.: s.n.

[29]. Peñaranda, Avila, 2012. *Kerberos*. 2012. S.l.: s.n.

[30]. Power Boken Open Project, 2014. PowerBroker Open - Likewise Open - Active Directory *Software* Integration. [en línea]. [Consulta: 19 mayo 2015]. Disponible en: http://www.powerbrokeropen.org/.

[31]. Pressman, 2005. *Pressman Cap 09 Ingeniería del diseño*. S.l.: s.n.

[32]. Pressman, 2010a. *7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman*. S.l.: s.n.

[33]. Pressman, 2010b. *Capítulo 9 Arquitectura de Diseño*. S.l.: s.n.

[34]. Pressman, 2005. *Ingeniería de Software un enfoque práctico*. S.l.: s.n.

[35]. Roger S. Pressman, 2005. *Ingeniería del Software*. 6. México: s.n.

[36]. Ruíz, 2013. 3.2. Conceptos básicos en una estructura de Directorio Activo. [en línea]. [Consulta: 22 mayo 2015]. Disponible en: http://somebooks.es/?p=3375.

[37]. Santos, 2009. OpenUP: Un proceso ágil. [en línea]. [Consulta: 21 mayo 2015]. Disponible en: http://www.ibm.com/developerworks/br/rational/local/open_up/.

[38]. Sierra, 2005. *Trabajando con Visual Paradigm for UML*. 2005. S.l.: s.n.

[39]. Terrasa, Ferrer, 2010. *Que es el directorio Activo*. 2010. S.l.: s.n.

[40]. Tridgell, 2002. Manual de referencia. [en línea]. [Consulta: 4 junio 2015]. Disponible en: file:///home/hmast/.mozilla/firefox/krr72u6l.default/ScrapBook/data/20150603155300/index.html.

[41]. Valencia, A.M. y Ferro, 2011. Documentación *y análisis crítico de algunas arquitecturas de software en aplicaciones empresariales*. 2011. S.l.: s.n.

[42]. Van Rossum, 2009. *El tutorial de Python* [en línea]. 2009. S.l.: s.n. Disponible en: http://python.org.ar/pyar/Tutorial.

[43]. Wesley, 2006. *Patrones de diseño*. 2006. S.l.: s.n.

[44]. Wiegers, Beatty, 2013. *Software Requirements, Third Edition*. S.l.: s.n.

[45]. Yang, Allen, 2015. OpenUP. [en línea]. [Consulta: 4 junio 2015]. Disponible en: http://epf.eclipse.org/wikis/openup/index.htm.

[46]. Alarcon, 2000. *Diseño orientado a objetos con UML*. S.l.: s.n.

[47]. Sommerville, 2005. *Somerville Parte 2 Requerimientos* [en línea]. 2005. S.l.: s.n. Disponible en: http://eva.uci.cu/file.php/158/Documentos/Bibliografia_general/Textos_Complementarios/Ediciones_del_Sommerville/Sommerville_7ma_edicion/Sommerville_Parte_II_Requerimientos.pdf.

Bibliografía

Carter, Gerald. 2003. *LDAP System Administration.* s.l.: O'Reilly, 2003. 1-56592-491-6.

Desmond, Brian. 2009. *Active Directory, Fourth Edition.* United States of America: Appingo, Inc., 2009. 978-0-596-52059-5.

Espinosa Peña, Viviana Isabel. 2012. *Instalación, Configuración y Pruebas de Funcionamiento del Servicio de Samba en Fedora 17 Linux-Unix.* s.l.: Universidad Francisco de Paula Santander, 2012. 1150017.

Eyes, David. 2006. *Extending Native Active Directory Capabilities to Unix and Linux.* U.S. y Canada: World Headquarters, 2006. 949.754.8000.

Hull, Elizabeth. 2005. *Requirements Engineering Second Edition.* United States of America: s.n., 2005. 1-85233-879-2.

Llorente, César de la Torrente. 2010. *Guía de arquitectura N-Capas orientada al dominio .Net (beta).* España: Krasis Consulting, 2010. 978-84-936696-3-8.

Luna Toro, Aparicio Ernesto. 2003. *Informe Final Práctica Académica Modalidad Práctica Empresarial.* Medellín: Intergrupo S.A, 2003. 8.028.831.

R. Stanek, William. 2012. *Window Server 2012.* United States of America. : Microsoft Press, 2012. 978-0-7356-6633-7.

Red Hat. **Heslin, Mark. 2012.** USA: Varsity Drive, 2012, Vol. Version 1.5.

S.Pressman, Roger. 2010. *7th ed software engineering a practitioners approach.* New York: The McGraw-Hill Companies, 2010. 978–0–07–337597–7.

Sunpy - Python para física solar: una implementación para seguimiento de correlaciones locales. **Campos Rozo, José, Iván. 2014.** No.1, México: Publicado Online, 2014, Vol. Vol. 6 . 2256-3830.

Wiegers, Karl. 2013. *Software Requirements, 3rd Edition.* Washington: Microsoft Press, 2013. 978-0-7356-7966-5.

Somerville, Ian. 2005. Ingeniería del Software, Séptima Edición. España: Pearson Educación, SA, 2005. 84-7829-074-5.

Schmuller, Joseph, 2000. UML en 24 Horas. México: Pearson Educación, 2000. 968-444-463-X.

H.Howes, Timothy, 2003. Understanding and deploying LDAP Directory Services, Second Edition. Boston: Pearson Education, Inc, 2003. 0-672-32316-8.

Somerville, Ian. 2007. Software Engineering , Eighth Edition. Republic of China: Pearson Education Limited. 2007. 978-0-321-31379-9.

Anexos

Anexo 1: Descripción del CU Definir Políticas de Acceso.

Caso de Uso:	Definir políticas de acceso
Actores:	Especialista en migración
Resumen:	El caso de uso inicia cuando el Actor ha unido el ordenador al dominio y selecciona la opción ¨Políticas de acceso¨ en el panel de opciones de la herramienta.
Pre-condiciones:	El usuario debe haber unido el ordenador al dominio.
Complejidad	Alta
Prioridad	Media

Flujo Normal de Eventos	
Flujo básico: Definir políticas de acceso	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción “Políticas de Acceso”.	2. Muestra una ventana con dos tablas que contienen los usuarios y los grupos permitidos y denegados y las opciones: 1. Añadir a permitidos. Ver sección (1) 2. Añadir a denegados. Ver sección (2)
Sección 1: Añadir a permitidos	
Flujo básico: Añadir a permitidos	
Acción del Actor	**Respuesta del Sistema**

1. Selecciona la opción “Políticas de Acceso”.	2. Muestra una ventana con dos tablas que contienen los usuarios y los grupos permitidos y denegados y dos botones que indican las acciones de añadir a permitidos y añadir a denegados.
3. Selecciona un usuario o grupo y presiona la opción que indica “Añadir a permitidos”.	4. Hace las configuraciones necesarias e incluye el usuario o el grupo a la lista. 5. Termina el CU.
Flujo Alterno	
3.1 No se selecciona ningún usuario o grupo	
Acción del Actor	**Respuesta del Sistema**
	Muestra un mensaje notificando que es necesario seleccionar algún usuario o grupo.
Sección 2: Añadir a denegados	
Flujo básico: Añadir a denegados	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción “Políticas de Acceso”.	2. Muestra una ventana con dos tablas que contienen los usuarios y los grupos permitidos y denegados y dos botones que indican las acciones de añadir a permitidos y añadir a denegados.
3. Selecciona un usuario o grupo y presiona la opción que indica “Añadir a denegados”.	4. Hace las configuraciones necesarias e incluye el usuario o el grupo a la lista. 5. Termina el CU.
Flujo alterno	

3.1 No se selecciona ningún usuario o grupo	
Acción del Actor	**Respuesta del Sistema**
	Muestra un mensaje notificando que es necesario seleccionar algún usuario o grupo.
Pos-condiciones	Queda establecida la política de seguridad

Tabla 5: Descripción del CU Definir Políticas de Acceso.

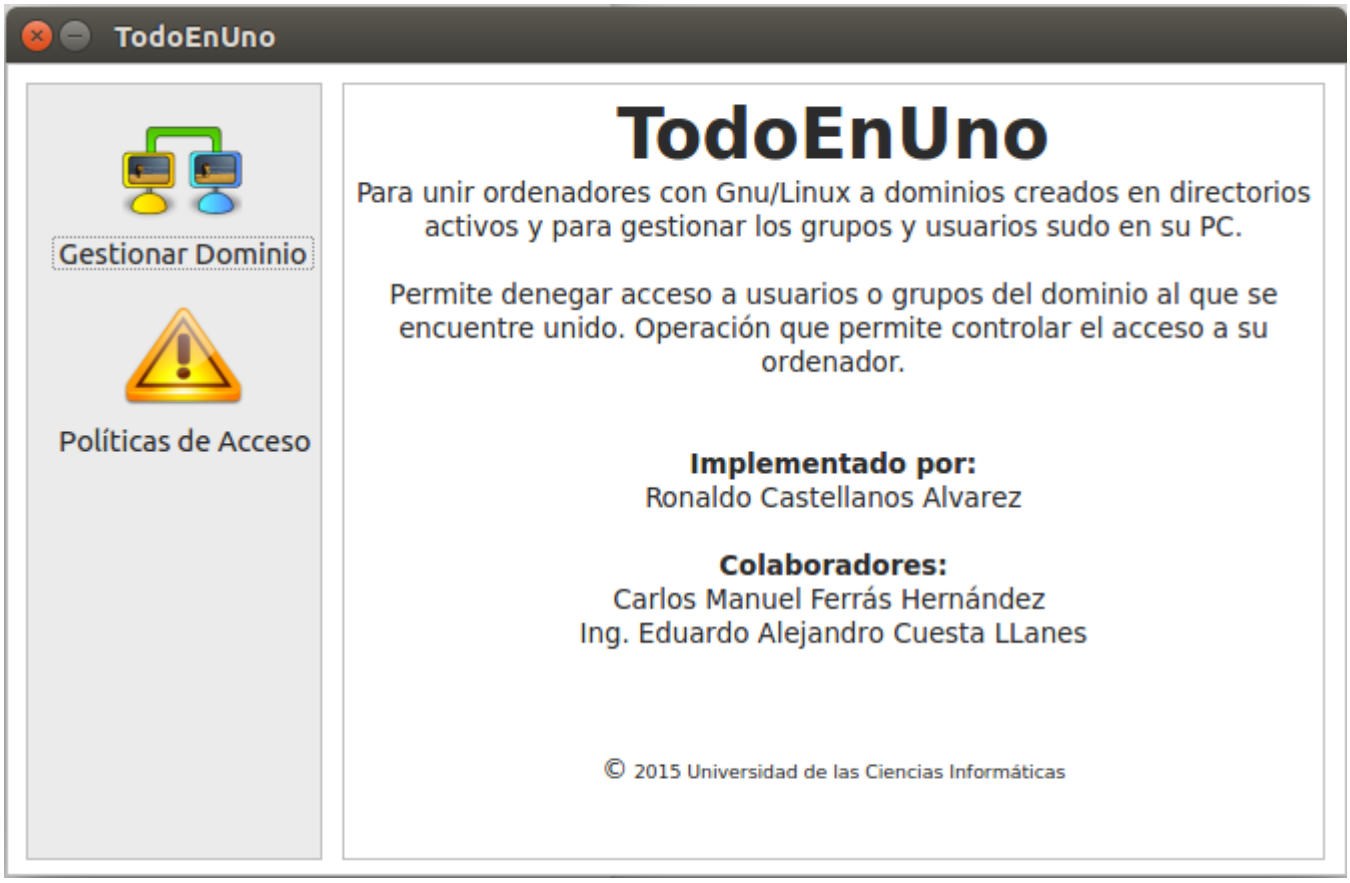
Anexo 2: Descripción del CU Gestionar permisos administrativos.

Caso de Uso:	Gestionar los permisos administrativos
Actores:	Especialista en migración
Resumen:	El caso de uso inicia cuando el Actor ha unido el ordenador al dominio y selecciona la opción “Gestionar políticas de acceso administrativos” en el panel de opciones de la herramienta.
Pre-condiciones:	El usuario debe haber unido el ordenador al dominio.
Flujo Normal de Eventos	
Flujo básico: Permisos administrativos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción “Gestionar políticas de acceso administrativos”.	1.1 Muestra una ventana con dos tablas que contienen los usuarios y grupos permitidos y los denegados. Los elementos de la tabla de los permitidos pueden ser marcados. Opciones a realizar: a) Seleccionar para añadir al grupo sudo. Ver sección (a) b) Quitar selección para eliminar del grupo de administradores. Ver sección (b)
Sección a: Añadir a grupo sudo	
Flujo básico: Añadir a grupo	

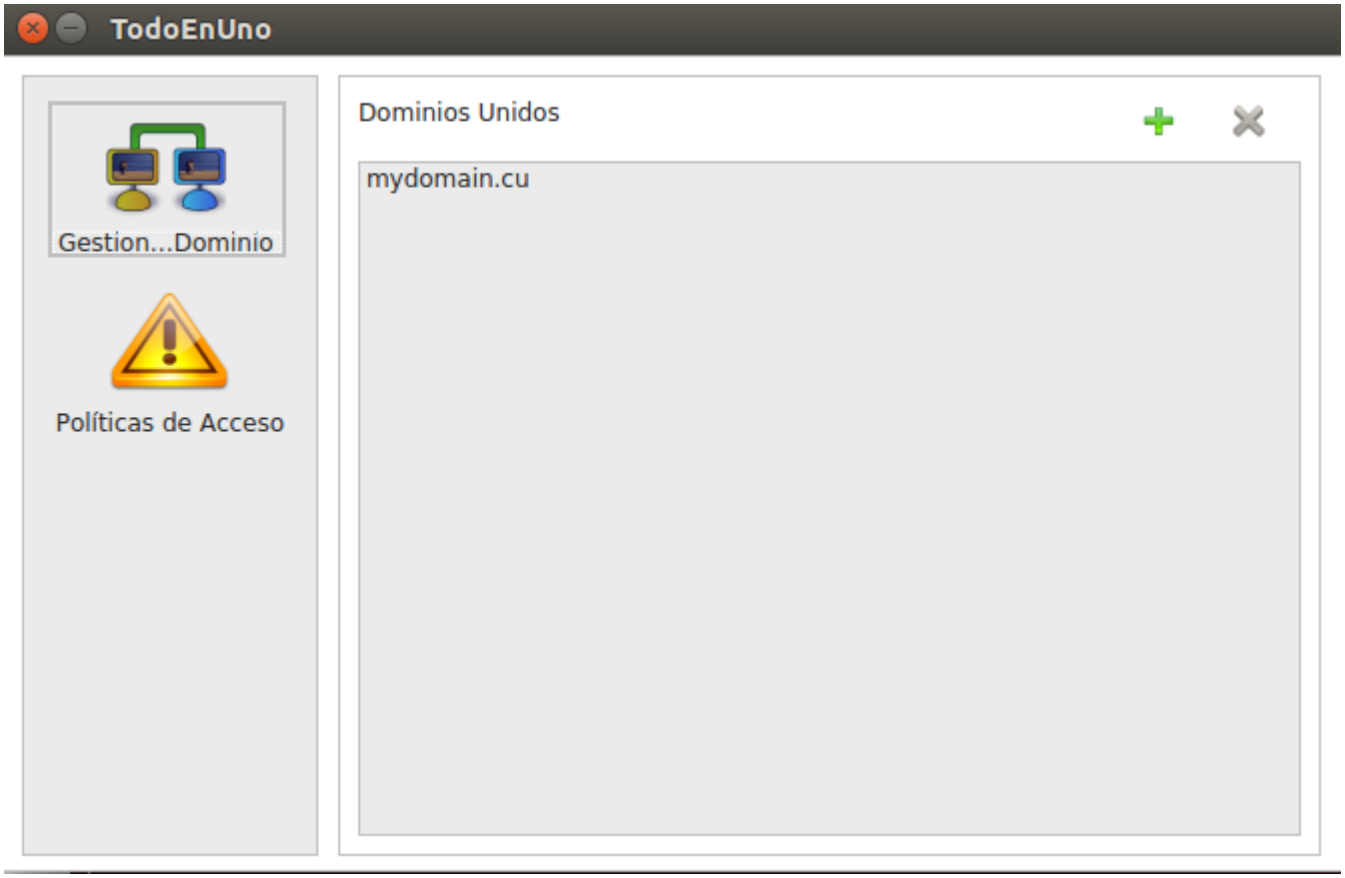
Acción del Actor	Respuesta del Sistema
2. Selecciona un grupo o usuario para otorgarle permisos administrativos.	2.1 El sistema realiza las configuraciones necesarias para otorgarle los permisos al usuario o grupo.
Flujo Alterno	
2.1 No se agregan los permisos administrativos al usuario o grupo seleccionado.	
Sección b: Eliminar del grupo	
Flujo básico: Eliminar del grupo	
Acción del Actor	**Respuesta del Sistema**
3. Selecciona un grupo o usuario que se encuentre ya marcado.	3.1 El sistema realiza las configuraciones necesarias para eliminar los permisos administrativos al usuario o grupo.
Flujo Alterno	
3.1 No se eliminan los permisos administrativos del usuario o grupo seleccionado.	
Pos-condiciones	Queda establecida la política de seguridad

Tabla 6: Descripción del CU Gestionar permisos administrativos.

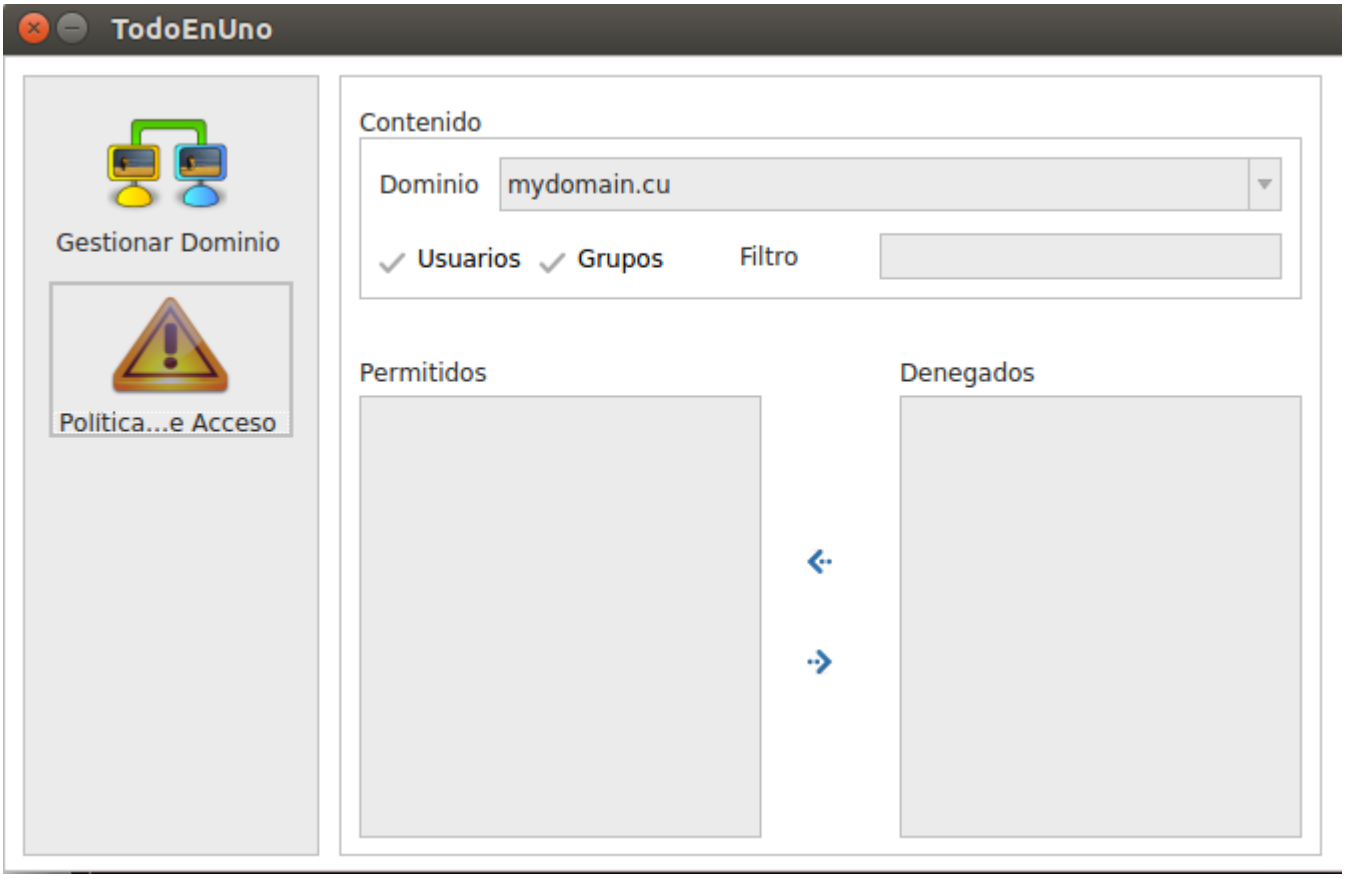
Anexo 3: Ventana Principal TodoEnUno



Anexo 4: Funcionalidad de Gestionar Dominio de TodoEnUno



Anexo 5: Ventana de gestionar políticas de seguridad de TodoEnUno



Anexo 6: Imagen que representa la estructura del Directorio Activo.