“Control de acceso a recursos compartidos en el Nautilus”

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

Autor:

Aldy León García

Tutores:

Ing. Juan Manuel Fuentes Rodríguez

Ing. Asistente. Gladys Marsi Peñalver Romero

La Habana, junio del 2017

“Yo creo bastante en la suerte. Y he constatado que, cuanto más duro trabaje, más suerte tengo.”

Thomas Jefferson

Dedicatoria

A mis padres, quienes me han brindado amor incondicional y siempre me han impulsado a superarme profesionalmente y ofreciéndome aliento constante para lograrlo.

A mi hermana y familiares en general, que me apoyan siempre y ayudan a conseguir mis anhelos.

A todos mis profesores y amigos en la UCI.

Agradecimientos

A mis tutores, por hacer de la programación un parque de atracciones.

A todas aquellas personas que de una forma u otra me ayudaron a convertirme en un Ingeniero Informático.

Declaración de autoría

Declaro por este medio que yo Aldy León García, con carné de identidad 9305071147 soy el autor principal del trabajo titulado "*Control de acceso a recursos compartidos en el Nautilus*" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2017.

_____

Aldy León García

_____

Ing. Juan Manuel Fuentes Rodríguez

_____

Ing. Asistente. Gladys Marsi Peñalver Romero

Resumen

Los usuarios de GNU/Linux para compartir sus directorios con usuarios de Microsoft Windows utilizan el servidor de archivos e impresoras Samba. Solo una de las herramientas existentes para gestionar el servidor Samba, el módulo Nautilus Share puede compartir directorios sin privilegios de administración. Nautilus Share, aunque posee deficiencias en el control de los permisos de acceso, destaca por brindar al usuario la posibilidad de compartir un directorio con solo tres pasos: seleccionar un directorio, configurar permisos y compartir. El principal aporte de esta investigación es permitir que los usuarios de la Distribución Cubana GNU/Linux Nova puedan contar con una herramienta para compartir directorios con el gestor de archivos Nautilus con mayor seguridad al gestiona los permisos de lectura y escritura al compartir un directorio.

Palabras clave: *compartir, información, Nautilus, Nautilus Share, Samba.*

Índice

Índice de ilustraciones

Índice de tablas

Introducción

La Universidad de las Ciencias Informáticas (UCI) es una de las instituciones encargadas del proceso de informatización de nuestro país, desempeña un rol especializado en la investigación y desarrollo de aplicaciones informáticas, para satisfacer las necesidades de migración a plataformas de código abierto. La universidad tiene entre sus centros de desarrollo e investigación el Centro de Software Libre (CESOL) encargado del desarrollo, mantenimiento y soporte de la Distribución Cubana de GNU/Linux (Nova) con el objetivo de brindar una distribución del sistema operativo estable y segura.

Nova es un sistema operativo derivado del proyecto Debian, resultado de la creación del núcleo de Linux por parte de Linux Torvalds y el aporte de software libre del proyecto GNU. Brinda soporte a los protocolos de comunicación para compartir directorios con diferentes niveles de seguridad y formas de acceso; ejemplos de ellos son el Protocolo de Transferencia de Archivos (FTP), Sistema de Ficheros en Red (NFS) y Bloque de Mensajes del Servidor (SMB). Los usuarios prefieren los protocolos NFS y SMB por la posibilidad de poder acceder a su información almacenada remotamente en otra computadora como si fuera una unidad de disco duro lógica montada en su sistema.

Nova cuenta con el gestor de ficheros Nautilus perteneciente al proyecto GNOME, el cual posee capacidad de ampliar sus funcionalidades por medio de módulos adicionales. Para compartir directorios con Nautilus se ha desarrollado el módulo Nautilus Share, el cual comparte un directorio seleccionado por el usuario. En un análisis al módulo se obtuvo como resultados la posibilidad de compartir directorios sin privilegios de administración, además de enmascarar el nombre compartido y añadir comentarios. En cuanto al control de los permisos de lectura y escritura de los usuarios solo es posible compartir con todos los usuarios del servidor Samba en conjunto o con el propietario del directorio. Estos resultados revelan que Nautilus Share presenta una gran deficiencia en cuanto al control de los permisos de lectura y escritura de los usuarios. Esta deficiencia trae como consecuencia la modificación y eliminación accidental de la

información sensible ya sea por usuarios malintencionados o no autorizados; e incumplimiento de la seguridad de la información en su carácter obligatorio de la protección de datos personales.

Consecuentemente se formula como **Problema de científico**: ¿Cómo gestionar los permisos de lectura y escritura de los directorios a compartir, para lograr mayor seguridad usando el módulo Nautilus Share del gestor de archivos Nautilus? Lo cual define como **Objeto de investigación**: la gestión de permisos de lectura y escritura de los usuarios para compartir directorios en la distribución cubana de GNU Linux Nova.

Para darle solución al problema planteado, se enuncia como **Objetivo General**: Adicionar al módulo Nautilus Share la funcionalidad de gestionar los permisos de lectura y escritura de los directorios a compartir para lograr mayor seguridad usando el módulo Nautilus Share del gestor de archivos Nautilus. Lo cual delimita como **Campo de acción:** la gestión de permisos de lectura y escritura de los usuarios para compartir directorios con el módulo Nautilus Share.

Para darle cumplimiento al objetivo general se definen los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Diseñar e implementar la solución propuesta.
- ✓ Probar el correcto funcionamiento de la solución.

Para darle cumplimiento a los objetivos específicos planteados se definen las siguientes **tareas de investigación**:

1. Análisis de herramientas homólogas del módulo Nautilus Share para compartir directorios en Nova.
2. Definición de la propuesta de solución para gestionar permisos de lectura y escritura con Nautilus Share.
3. Descripción de la arquitectura de la aplicación propuesta para lograr la gestión de los permisos con Nautilus Share.

4. Codificación de la solución propuesta.
5. Evaluar la solución desarrollada empleando una estrategia de pruebas.

Para desarrollar la investigación se definen las **preguntas científicas** siguientes:

1. ¿Cómo puede contribuir el estudio de los sistemas homólogos a la solución propuesta?
2. ¿Cuáles son las herramientas de software para desarrollar la solución propuesta?
3. ¿Cómo contribuye la etapa de análisis y diseño a la solución propuesta?
4. ¿El uso de estándares de codificación, patrones de diseño y una arquitectura de software favorece la implementación de la solución propuesta?
5. ¿Qué estrategia de pruebas de software garantiza que la solución propuesta es funcional?

Para dar cumplimiento a las tareas de investigación y presupuestos hipotéticos planteados se emplearon los siguientes **métodos científicos**:

Como **método empírico** se utiliza la **observación** para chequear el estado actual de varias soluciones homólogas para reutilizar funcionalidades utilizadas dentro de los mismos.

La **encuesta** permitió obtener criterios y valoraciones por parte de usuarios que trabajan con la distribución cubana GNU/Linux Nova, con respecto a Nautilus Share 0.7.3 antes y después de la presente investigación.

Entre los **métodos teóricos** utilizados en la investigación está el **analítico-sintético** en el estudio del uso y de las características del módulo Nautilus Share y las soluciones homólogas existentes, lo que facilitó la búsqueda de conceptos fundamentales para el desarrollo de la investigación.

La presente tesis está compuesta por: introducción, tres capítulos, conclusiones, recomendaciones y bibliografía.

Capítulo 1: Fundamentación teórica para compartir directorios con Nautilus Share.

En este capítulo se realiza un estudio de las aplicaciones homólogas de Nautilus Share para compartir directorios, así como la metodología, herramientas y lenguajes necesarios para el desarrollo de la solución propuesta. Se muestran los conceptos básicos necesarios para la comprensión del funcionamiento de una aplicación de este tipo, además de todas las facilidades que debe brindarle al usuario.

Capítulo 2: Diseño de la solución propuesta.

Haciendo uso de la metodología seleccionada, se propone una solución al problema planteado, realizando su diseño. Se especifican los requisitos funcionales y no funcionales.

Capítulo 3: Implementación y prueba de la solución propuesta.

Comprende la explicación de los estándares de codificación utilizados en el desarrollo de la solución propuesta, así como el diagrama de componentes que apoya el proceso de implementación. Se explican los distintos tipos de pruebas que se van a utilizar para comprobar el correcto funcionamiento de la solución. Se muestran los resultados de las pruebas y el impacto social de la investigación.

Capítulo 1: Fundamentación teórica para compartir directorios con Nautilus Share.

Introducción

En el presente capítulo se presentan varios elementos para facilitar la comprensión por parte del lector sobre el objetivo de la investigación. Se realiza un estudio de homólogos del módulo Nautilus Share utilizados para compartir directorios en Nova. Se plantean la metodología y las herramientas necesarias para el desarrollo de la investigación.

1.1 Conceptos asociados al tema de investigación

Para apoyar el proceso de investigación es necesario conocer la definición de directorio, los permisos aplicados a un directorio, las características el gestor de archivos Nautilus del entorno de escritorio GNOME, así como a necesidad de cumplir con la Seguridad de la Información.

Directorio

Un archivo es un mecanismo de abstracción para representar datos de una manera simple usando básicamente su nombre y opcionalmente una extensión que indica el tipo de archivo. Un directorio o carpeta es un archivo especial porque no tiene extensión y puede contener otros archivos, sea uno o varios ficheros simples o directorios como muestra la Ilustración 1. [5] Para la presente investigación un directorio se define como un archivo sin extensión que pueda contener otros directorios y archivos.

Ilustración 1: Árbol de ficheros.

Permisos de directorios.

En Linux, todo archivo y directorio tiene tres niveles de permisos de acceso: los que se aplican al propietario del archivo, los que se aplican al grupo que tiene el archivo y los que se aplican a todos los usuarios del sistema. Estos permisos son lectura, escritura y ejecución. [6]

Se muestra en el Anexo 1 cuál es el resultado de listar los archivos contenidos en un directorio, el primer grupo de información nos muestra el tipo de archivo como se puede observar en la Anexo 2 y cuáles son los permisos aplicados para el propietario, grupo u otros en la Anexo 3.

Entorno de escritorio GNOME

Es un entorno de escritorio e infraestructura de desarrollo para sistemas operativos UNIX y GNU Linux, BSD o Solaris; compuesto enteramente de Software Libre (SWL). El proyecto fue iniciado por los programadores mexicanos Miguel de Lcaza y Federico Mena, y forma parte oficial del proyecto GNU. Nació como una alternativa a KDE bajo el nombre de GNU Network Object Model Environment (GNOME). El proyecto GNOME, según sus creadores, provee un gestor de ventanas intuitivo y atractivo y una plataforma de desarrollo para crear aplicaciones que se integran con el escritorio. [3]

El proyecto pone un gran énfasis en la simplicidad, usabilidad y eficiencia y tiene como principal objetivo crear un sistema de escritorio para el usuario final que sea completo, libre y fácil de usar. Asimismo, se

pretende que GNOME sea una plataforma muy potente de cara al desarrollador. Otros objetivos del proyecto son: la libertad para crear un entorno de escritorio que siempre tendrá el código fuente disponible para reutilizarse bajo una licencia de software libre; el aseguramiento de la accesibilidad, de modo que pueda ser utilizado por cualquiera, sin importar sus conocimientos técnicos; hacer que esté disponible en muchos idiomas, actualmente está siendo traducido a más de 100 idiomas y por último que tenga un ciclo regular de liberaciones y una estructura de comunidad disciplinada. [3]

Gestor de ficheros Nautilus

Es el administrador de archivos oficial del entorno de escritorio GNOME. Su nombre es un juego de palabras, emulando una Concha, evocando las consolas de los sistemas operativos GNU/Linux. Nos permite realizar las operaciones más comunes con archivos y directorios, tales como copiar, cortar, pegar y eliminar. Los desarrolladores prefieren mantenerlo lo más simple y sencillo posible por lo que dejan a la comunidad de desarrolladores implementar módulos personalizados para agregar funcionalidades. El módulo Nautilus Share para Nautilus permite compartir directorios en red sin privilegios de administración. [4]

Seguridad de la Información

La información es el principal activo de muchas organizaciones por lo que es necesario protegerla adecuadamente frente a amenazas que puedan poner en peligro la continuidad del negocio. La Seguridad de la información tiene como fin la protección de la información y de los sistemas de la información del acceso, uso, divulgación, interrupción o destrucción no autorizada. [10]

Una red informática permite a los usuarios que usan el mismo software o protocolo compartir sus directorios. El intercambio de archivos puede permitir el ahorro de espacio de información duplicada, fomentar el trabajo en equipo y optimiza el flujo de datos. Para compartir información en red hay que tener en cuenta políticas de seguridad de la información para garantizar que cuando se está conectado no permitir que otras personas copien archivos privados, credenciales de acceso, así como descargar un virus o propiciar un problema de seguridad. [9]

Existen varias herramientas para gestionar la seguridad de la información, obligatorias como la Ley Orgánica de Protección de Datos de Carácter Personal y voluntarias como los Sistemas de Gestión de Seguridad de la Información.

1.2 Estado del Arte de Nautilus Share

En Nova, la versión actual del módulo Nautilus Share es la 0.7.3 el cual está integrado al gestor de archivos Nautilus. Es mantenido por proyecto GNOME y su código fuente está programado en C. Su principal funcionalidad es compartir un directorio desde el área de trabajo del Nautilus, con solo desplegar el menú de propiedades. Permite enmascarar el nombre del directorio compartido para personalizar y ocultar la ubicación real del directorio, agregar un comentario para esclarecer el contenido, de forma opcional permitir que los usuarios sin autenticación y los usuarios del sistema acceder al directorio compartido con permisos de lectura y escritura. [17] Se aprecia en la (Ilustación 2) la deficiencia en la gestión del control de acceso de los usuarios, incumpliendo con la seguridad de la información.

Ilustración 2: Interfaz de Nautilus Share.

Para el funcionamiento correcto de Nautilus Share es necesario tener un usuario registrado en el servidor de archivos e impresoras SAMBA dado que comparte los directorios utilizando el protocolo SMB, además de estar autorizado en el grupo sambashare por el administrador del sistema.

Servidor Samba

Compartir directorios en el ordenador para poder acceder a ellas desde otros equipos es una de las formas más cómodas de mantener el acceso a los archivos desde otros ordenadores. Cada vez es más común que en una misma red existan ordenadores con diferentes sistemas operativos, por lo que cada uno de ellos es el encargado de facilitar el medio para compartir los directorios.

En el caso de un ordenador con el sistema operativo Microsoft Windows utiliza el protocolo SMB para compartir archivos en una red informática y su homólogo en GNU / Linux es el servidor Samba que brinda múltiples servicios para el trabajo en redes informáticas. Samba es una implementación libre del protocolo de archivos compartidos de Microsoft Windows para sistemas de tipo UNIX. De esta forma, es posible que ordenadores con GNU/Linux, MacOS X o Unix en general se vean como servidores o actúen como clientes en redes de Windows. Samba también permite servir colas de impresión, directorios compartidos y autentificar con su propio archivo de usuarios. [12]

Samba es un conjunto de aplicaciones que brindan servicios relacionados entre sí. Estos son llamados demonios que son servicios que se ejecuta en segundo plano, fuera del control interactivo de los usuarios del sistema ya que carecen de interfaz con estos. Para interaccionar con los usuarios existen varias aplicaciones basada en texto para realizar configuraciones como se muestra en el Anexo 4. En el Anexo 5 se muestran los comandos para gestionar los usuarios del servidor Samba. [13]

1.3 Estudio de sistemas homólogos al módulo Nautilus Share

A continuación, se realiza un estudio a los sistemas homólogos al módulo Nautilus Share, de los que se citan Gadmin Samba, La herramienta de configuración del servidor Samba y net usershare. Son seleccio-

nados por utilizar el servidor de archivos e impresoras SAMBA y ser los más populares para compartir directorios con el protocolo SMB en Nova. Se enuncian una serie de características y desventajas de las mismas, permitiendo arribar a conclusiones que consoliden el proceso de desarrollo de la herramienta.

Gadmin-Samba

Gadmin-Samba es una aplicación del entorno de escritorio de GNOME, programada en C para administrar el servidor Samba. Administra la configuración principal del servidor ubicado generalmente en "/etc/samba/smb.conf" en forma de formularios agrupados por categorías, ver Ilustración 3. En la categoría de compartidos se muestra es estado del servidor Samba, los directorios compartidos en una lista con los atributos: Nombre Compartido, ubicación del directorio y comentarios. Dispone de un formulario para crear o modificar un directorio compartido con las siguientes opciones: nombre compartido, directorio compartido, comentario, una lista de usuarios o grupos con permisos de acceso, con permisos de escritura, con acceso denegado y propietarios, una máscara de permisos para el directorio, solo lectura, modificable, disponible, navegable, usuario sin autenticación, público e imprimible.

Puede ser instalado utilizando el gestor de paquetes de la distribución o la herramienta apt utilizando el siguiente comando: "sudo apt-get install gadmin-samba" Esta herramienta es recomendada para usuarios expertos que deseen tener un completo control sobre el servidor Samba. Está programado en C utilizando el entorno de trabajo GTK+. Requiere de permisos de administración para configurar el servidor Samba. [15]

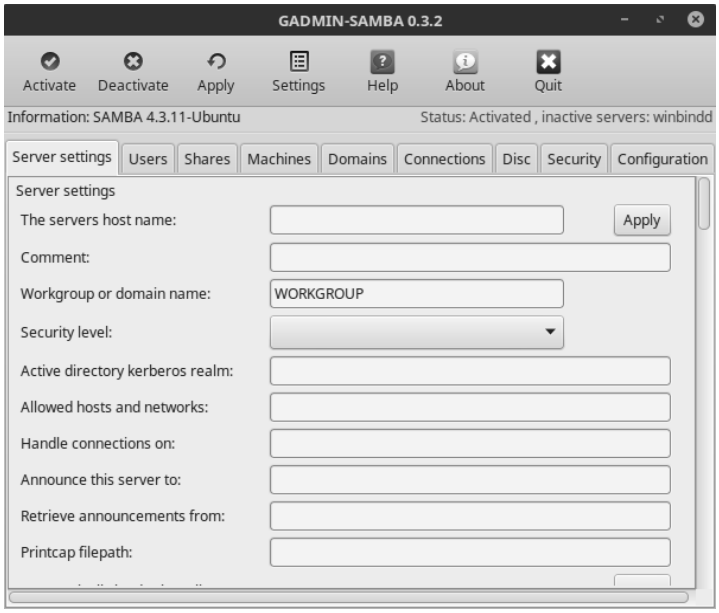


Ilustración 3: Interfaz de Gadmin-Samba.

Herramienta de configuración del servidor Samba

La herramienta de configuración del servidor Samba es programada en Python, para entorno de trabajo GTK+ posee una interfaz en forma de formulario para administrar los recursos compartidos por Samba. Modifica los parámetros de configuración en el archivo de configuración ubicado en "/etc/samba/smb.conf". En pocos pasos se puede compartir un directorio con usuarios no pertenecientes al servidor Samba debido a la funcionalidad de gestionar los usuarios del servidor, así como los permisos de acceso a los directorios compartidos. Se debe tener privilegios de administración para usar la herramienta. [16]

Puede ser instalada utilizando el gestor de paquetes o la herramienta apt utilizando el siguiente comando: **"***sudo apt-get install system-config-samba***".** Esta herramienta es recomendada a usuarios

con privilegios de administración que deseen una fácil gestión de los directorios compartidos y los usuarios del servidor Samba como se puede observar en la Ilustración 3.

Ilustración 4: Herramienta para configurar el servidor Samba.

net usershare

Para administrar el servidor Samba por línea de comandos se utiliza la herramienta “net” que se instala con el paquete del servidor Samba. Entre las funcionalidades de “net” está “net usershare” encargado compartir directorios sin modificar el archivo de configuración de Samba, sino que guarda la información respectiva a los directorios compartidos en la ubicación “/var/lib/samba/usershares”. Un directorio con permisos de lectura para el grupo sambashare y escritura para el usuario creador de la configuración. A continuación, se muestra las funcionalidades de net usershare en la Tabla 1.

Tabla 1: Funcionalidades de net usershare.

Comando:	Descripción
net usershare **add**	Adiciona o modifica el directorio definido por el usuario.
net usershare **delete <sharename>**	Elimina el directorio compartido definido por el usuario.
net usershare **info**	Muestra información de los directorios compartidos.
net usershare **list**	Muestra directorios compartidos.
Ejemplo para añadir un directorio compartido.	
net usershare add [-l|--long] <sharename> <path> [<comment>] [<acl>] [<guest_ok=[y|n]>]	
<sharename>	Nuevo nombre compartido.
<path>	Dirección en el sistema de ficheros local.
<comment>	Comentario acerca del contenido.
<acl>	Añadir una regla opcional para el control de usuarios. En el formato "DOMAIN@name:X,DOMAIN@name:X,…." DOMAIN dominio donde pertenece el usuario, si es local se mantiene DOMAIN. Name es el nombre del usuario o grupo. X es uno de los permisos garantizados. F para lectura y escritura, R solo para Lectura, D ningún permiso.
<guest_ok>	Permite compartir con usuarios sin autenticar.

En la mayoría de las distribuciones GNU / Linux la instalación del paquete Samba del repositorio realiza los ajustes necesarios en el sistema para el funcionamiento de la herramienta net usershare. Para configurar manualmente la herramienta en el sistema se debe seguir los siguientes pasos:

1- Modificar /etc/smb.conf

[global]

workgroup = WORKGROUP

usershare path = /var/lib/samba/usershare

usershare max shares = 100

usershare allow guests = yes

usershare owner only = yes

create mask = 0777

2- Crear el directorio:

mkdir /var/lib/samba/usershare

3- Modificar el propietario del directorio

chgrp sambashare /var/lib/samba/usershare

4- Cambiar los permisos del directorio:

chmod 1770 /var/lib/samba/usershare

La herramienta “net usershare” permite configurar los directorios compartidos sin privilegios de administración.

Valoración del estudio de sistemas homólogos al módulo Nautilus Share

Concluido el estudio de los sistemas homólogos: Gadmin Samba, La herramienta de configuración del servidor Samba y “net usershare” para compartir directorios con el servidor Samba. Se determina que todas cuentan con la gestión de los permisos de lectura y escritura sobre los directorios compartidos, utilizando listas de control de acceso. Los elementos más utilizados para compartir directorios son: Nombre

del directorio, dirección real del directorio, comentarios, acceso de usuarios sin autentificación, lista de usuarios con permisos de lectura y escritura. Un factor determinante para valorar estos sistemas fue la necesidad de compartir directorios sin privilegios de administración, dado que la mayoría de los usuarios que utilizan el módulo Nautilus Share no poseen privilegios elevados y solo pueden compartir sus directorios personales. Net usershare es la herramienta seleccionada para permitir que el módulo Nautilus Share pueda compartir directorios con control de acceso a los recursos compartidos y sin privilegios de administración.

Para el desarrollo de la propuesta de solución se determina la metodología de desarrollo y las herramientas necesarias a continuación.

1.4 Metodología de desarrollo

Una metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Es un acumulado de procedimientos, técnicas, artefactos y herramientas que guían a los desarrolladores en el proceso de realización de software. [19]

Se escogió como metodología a utilizar una variación de la metodología “Proceso Unificado Ágil” (AUP) en unión con el modelo CMMI-DEV v 1.3 desarrollada en la UCI denominada AUP-UCI, la cual es idónea para la entrega de productos en cortos períodos de tiempo; además de satisfacer las necesidades del proceso. [19] AUP-UCI es una metodología flexible que no requiere de una gran cantidad de desarrolladores. Es concisa en el aspecto de la documentación, permitiendo generar solo la necesaria y no la especificada para cada flujo de trabajo como lo hace RUP. Está diseñada para trabajar en proyectos pequeños donde la atención se centra en las actividades que realmente son importantes. La metodología contiene un flujo de trabajo que cuenta con tres fases en el ciclo de vida del desarrollo de software. [19] Consta de tres fases en su ciclo de vida: inicio, ejecución y cierre.

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.5 Herramientas de desarrollo utilizadas

Para desarrollar la solución propuesta se realiza la selección de las herramientas necesarias para el diseño, modelado e implementación.

Marco de trabajo

Cómo marco de trabajo se selecciona GTK+ en su versión 3.18 por ser la librería gráfica sobre la que se sustenta todo el interfaz gráfico del proyecto GNOME. Es una librería que contiene todo lo necesario para el desarrollo de interfaces gráficas, permitiendo la posibilidad de crear todo tipo de *widgets*[1], desde los más básicos, como botones, etiquetas, cajas de texto, hasta cosas más complejas como selectores de ficheros, colores, fuentes, cajas de texto con soporte para todo tipo de efectos sobre el texto. [20]

[1] Widgets: En términos de ingeniería del software, un widget es un componente software visible y personalizable. Visible porque está pensado para ser usado en los interfaces gráficos de los programas, y personalizable porque el programador puede cambiar muchas de sus propiedades.

La librería GTK+ sigue el modelo de programación orientado a objetos. La jerarquía de objetos comienza en *GObject* de la librería *Glib* del que hereda *GtkObject*. Todos los widgets heredan de la clase de objetos *GtkWidget*, que a su vez hereda directamente de *GtkObject*. La clase *GtkWidget* contiene las propiedades comunes a todos los *widgets*; cada *widget* particular le añade sus propias propiedades. [20]

Lenguaje de programación.

Los lenguajes de programación son herramientas que nos permiten crear software. Utilizar como marco de trabajo GTK+ permite utilizar gran variedad de lenguajes de programación como Python, C, C++, Vala y Java. [20] El gestor de archivos Nautilus está programado en el lenguaje C, pero brinda un conjunto de interfaces para integrar módulos escritos en Python, Vala y C. El módulo Nautilus Share en su versión 0.7.3 originalmente fue programado en **C**, se continuará con este lenguaje por las ventajas que ofrece.

C es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel y más bien un lenguaje pequeño, sencillo y no está especializado en ningún tipo de aplicación. Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y, sobre todo, se aprende rápidamente. En poco tiempo, un programador puede utilizar la totalidad del lenguaje. [22] Este lenguaje ha sido estrechamente ligado al sistema operativo UNIX, puesto que fueron desarrollados conjuntamente. Sin embargo, este lenguaje no está ligado a ningún sistema operativo ni a ninguna máquina concreta. Se le suele llamar lenguaje de programación de sistemas debido a su utilidad para escribir compiladores y sistemas operativos, aunque de igual forma se puede desarrollar cualquier tipo de aplicación. [22]

Entorno de desarrollo integrado

Para el desarrollo de la aplicación se propone utilizar el Entorno de Desarrollo Integrado (IDE) **Builder** en su versión 3.20.4, el cual incluye funcionalidades y características adaptables a un proyecto desarrollado para el entorno de trabajo GTK+ y el lenguaje de programación C. Builder tiene soporte para la integración

con GNU/Make un conjunto de archivos de configuración que aseguran que el proyecto pueda ser distribuido e instalado en la mayoría de sistemas tipo Unix, especificando las dependencias a bibliotecas compartidas, recursos y un compilador de código. El compilador de código será gcc 5.4.0, que está disponible en la mayoría los sistemas Unix, con características para detectar errores sintácticos y dependencias incumplidas entre clases. [23]

Lenguaje de modelado

El modelado es una parte central de todas las actividades que conducen a la producción de buen software, porque a través de él logramos comprender el sistema a desarrollar. Existen gran variedad de lenguajes de modelado como el SysML, ERD, DFD, BPMN 2.0 y ArchiMate 2.0. Como lenguaje de modelado fue seleccionado el Lenguaje Unificado de Modelado (UML) en su versión 2.1 que permite la modelación de sistemas de software con tecnología orientada a objetos por ser el más conocido y utilizado en la actualidad. Es un lenguaje gráfico que cuenta con un grupo de diagramas, los cuales son utilizados para visualizar, especificar, construir y documentar un sistema de software en cada una de las etapas por las que tiene que pasar. UML indica que es lo que supuestamente hará el sistema, pero no como lo hará. El lenguaje de modelado UML permite generación automática de código además de realizar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. [24]

Herramienta de modelado

Las herramientas para la Ingeniería de Software Asistida por Ordenador (CASE) son las aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. [25] Existen tantas herramien-

tas CASE privativas como de código abierto, multiplataforma y con soporte para el lenguaje UML. Ejemplo de ellas son: Papyrus UML, Violet UML Editor, Lucidchart, draw.io, Dia, Umbrello. **Visual Paradigm** en su versión 8.0 fue la herramienta CASE seleccionada, es una herramienta para el modelado UML profesional que soporta el ciclo de vida completo de desarrollo del software: análisis y diseño, construcción, pruebas y despliegue, con licencia gratuita para proyectos no comerciales. [26]

Herramienta para los prototipos de interface

Para la realización de los prototipos de interfaces se decidió utilizar Glade 3.19 perteneciente al proyecto GNOME. Glade es una herramienta para desarrollar de forma fácil y sencilla as interfaces de usuario del entorno de trabajo GTK+ y el entorno de escritorio GNOME. Las interfaces de usuario generadas en Glade son guardadas en el formato XML, y son usadas por *GtkBuilder* para ser cargadas por la aplicación dinámicamente. Los ficheros XML de Glade pueden ser usados en numerosos lenguajes de programación incluyendo C, C++, C#, Vala, Java, Perl y Python.

1.6 Conclusiones parciales

En el presente capítulo se dio a conocer los elementos necesarios para fundamentar el objetivo de la investigación, se realizó un análisis detallado al módulo Nautilus Share en su versión actual 0.7.3, así como el estudio de los sistemas homólogos del módulo Nautilus Share. La gestión de los permisos de lectura y escritura de los usuarios es posible mediante las listas de control de acceso proporcionada por la herramienta “net usershare”. La utilización de esta herramienta permite a módulo Nautilus Share compartir directorios sin privilegios de administración. Para proponer una mejor propuesta de solución se seleccionaron los elementos fundamentales para compartir directorios: nombre de compartido, dirección real del directorio, comentario y lista de los usuarios con los permisos de lectura y escritura garantizados. La selección de la metodología de desarrollo AUP-UCI posibilitó controlar, planificar y estructurar todo el proceso de desarrollo de la aplicación. La elección de las herramientas de desarrollo como el lenguaje de programación C, el IDE Builder, la plataforma de desarrollo GTK+, la herramienta CASE Visual Paradigm, el len-

guaje de modelado UML y para el prototipo de interfaces la herramienta Glade: permitió que las necesidades técnicas fueran satisfechas para disminuir el tiempo de desarrollo.

Capítulo 2: Diseño de la solución propuesta para el módulo Nautilus Share.

Introducción

En el presente capítulo se realiza un análisis de las características del sistema a desarrollar a partir de la problemática planteada, se describen las principales funcionalidades que brindará el módulo Nautilus Share y los principales elementos que componen la arquitectura y que guían el proceso de construcción del software.

2.1 Modelo de dominio

El modelo de dominio es el encargado de representar los conceptos fundamentales para el desarrollo de una aplicación y las diferentes relaciones que existen entre ellos. Son presentados como clases los aspectos esenciales y las interrelaciones denotan una estructura de funcionamiento, brindan una mejor comprensión del medio actual para la concepción de un futuro sistema. [20] La siguiente ilustración 5 muestra el modelo de dominio del módulo Nautilus Share:

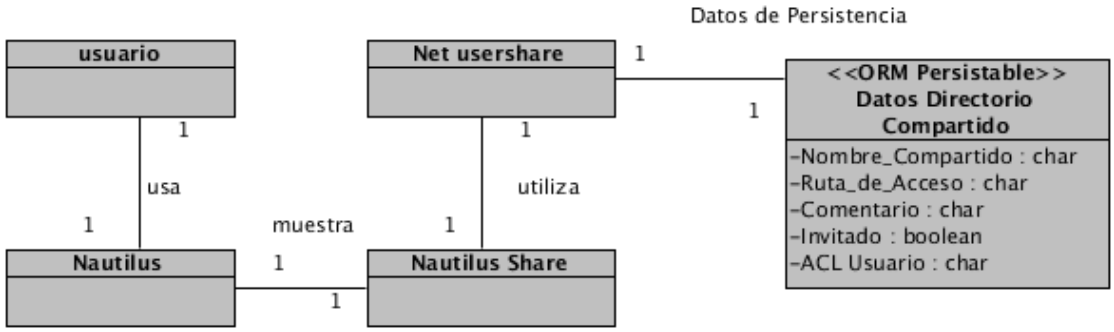


Ilustración 5: Modelo de dominio para la propuesta de solución.

A continuación, se describen cada uno de los objetos o relaciones que intervienen en el modelo de dominio del módulo Nautilus Share.

usuario: Actor que va a compartir un directorio.

Nautilus: Gestor de archivos del entorno de escritorio de GNOME

Nautilus Share: Es la vista de propiedades para compartir un directorio con Nautilus.

Net usershare: Herramienta para compartir un directorio con Samba sin privilegios de administración.

Datos Directorio Compartido: Fichero que guarda la configuración de los directorios compartidos.

2.2 Propuesta de solución

La solución que se propone para el control de acceso a recursos compartidos con el módulo Nautilus Share es el uso de las listas de control de acceso, brindadas por la herramienta “net usershare” para la gestión de los permisos sobre los directorios compartidos. Especificando tres atributos: nombre de usuario o grupo, dominio del usuario o grupo y el permiso garantizado de lectura y escritura. En cuanto a los atributos a los directorios compartidos estarán: nombre del directorio compartido, comentarios y la lista de permisos de acceso.

El módulo recibirá una dirección de un directorio, después validará que pertenece al usuario que ejecuta la acción, para generar un formulario con los atributos: nombre del directorio compartido, comentarios y lista de control de acceso con permisos de lectura y escritura. Una vez terminado de introducir los datos se crea un archivo de persistencia que es procesado por el servidor Samba para ser compartido en la red.

2.3 Requisitos funcionales y no funcionales

El esfuerzo principal en la disciplina requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del

producto. Para la obtención de los requisitos fue aplicada la técnica de tormenta de ideas con los demás programadores del proyecto, las entrevistas al cliente y teniendo en cuenta las experiencias anteriores.

2.3.1 Listado de requisitos funcionales (RF)

Un requisito funcional describe lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios del mismo y del enfoque general tomado por la organización al redactar los requerimientos. Cuando se expresan como requerimientos del usuario, habitualmente se describen de forma abstracta, sin embargo, los requisitos funcionales del sistema describen con detalle la función de este, sus entradas y salidas. [27]

Tabla 2: Requisitos funcionales

Requisitos Funcionales	**Prioridad**	**Descripción**
RF 1. Gestionar los permisos de lectura y escritura a usuarios de directorios compartidos.	Alta	El sistema solicita que sean introducidos los permisos de los usuarios y se valida si los usuarios tienen acceso a los directorios compartidos del servidor Samba.
RF 2. Adicionar permisos de lectura y escritura a usuarios sin autentificar en el sistema.	Alta	Se hablita el usuario invitado, que puede acceder a los directorios compartidos del servidor Samba sin autenticarse.
RF 3. Compartir un directorio.	Alta	Se verifican que los datos introducidos están correctos, se crea un archivo de persistencia para ser procesado por el servidor Samba.
RF 4. Modificar el nombre del directorio a compartir.	Baja	Establecer opcionalmente un nombre diferente al directorio compartido.
RF 5. Agregar comentarios al directorio a compartir.	Baja	Establecer opcionalmente un comentario para especificar el contenido del directorio.

2.3.2 Listado de requisitos no funcionales(RNF)

Un requisito no funcional: Como su nombre indica, son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema. [27]

Software:

RNF 1: Sistema operativo Distribución Cubana de GNU Linux Nova

RNF 2: Gestor de archivos Nautilus >= 2.10

RNF 3: Samba >= 3.0.27

Funcionalidad:

RNF 4: El usuario que va a compartir un directorio debe estar registrado en el servidor Samba.

RNF 5: El usuario que va a compartir un directorio debe ser parte del grupo sambashare.

RNF 6: El atributo nombre del directorio compartido no debe estar vacío.

RNF 7: Los usuarios declarados en la lista de control de acceso deben tener acceso al servidor Samba.

RNF 8: Verificar el estado del servidor Samba.

Seguridad:

RNF 9: Los datos de persistencia almacenados, solo pueden ser alterados por su creador.

Usabilidad:

RNF 10: Usar estándares de diseño de interfaz del proyecto GNOME.

2.4 Historias de usuario (HU)

La Historia de Usuario es la forma que tiene la metodología de desarrollo AUP-UCI en el escenario 4 para especificar los requisitos del software. Estas son escritas por los clientes, en las cuales el cliente describe brevemente las características que el sistema tiene que tener. Las Historias de Usuario deben ser lo suficientemente comprensibles y delimitadas como para que el programador las implemente en unas pocas semanas. [28] Tienen como características:

***Independientes*:** Deben ser atómicas en su definición. Es decir, se debe intentar que no dependa de otras historias para poder completarla.

***Negociables*:** Deben ser ambiguas en su enunciado para poder debatirlas, dejando su concreción a los criterios de aceptación.

***Valoradas*:** Deben ser valoradas por el cliente. Para poder saber cuánto aporta al valor de la aplicación y junto con la estimación convertirse en un criterio de prioridad.

***Estimables*:** Aunque sea siempre un poco como leer de una bola de cristal, deben poder ser estimadas. Tener su alcance lo suficientemente definido como para poder suponer una medida de trabajo en la que pueda ser completarla.

***Pequeñas*:** Para poder realizar una estimación con cierta validez y no perder la visión de la Historia de Usuario, se recomienda que sean mayores de dos días y menores de dos semanas.

***Verificables*:** Este es el gran avance de las Historias de Usuario. Que, junto con el cliente, se acuerdan unos “Criterios de Aceptación” que verifican si se ha cumplido con las funcionalidades descritas y esperadas.

A continuación, se definen las diferentes historias de usuarios que están presentes en el sistema.

Tabla 3 HU-01 Adicionar permisos de lectura y escritura a usuario.

Historia de Usuario	
Número: HU-01	**Nombre Historia de Usuario:** Adicionar permisos de lectura y escritura a usuario.
Modificación de Historia de Usuario: Ninguna	
Usuario: Aldy León García	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: Permitir asignar permiso de escritura y / o lectura a los usuarios.	
Observación: Para adicionar un permiso a un usuario se debe primero activar compartir directorio, luego seleccionar un usuario de la lista, dar clic en adicionar.	
Prototipo de interfaz: 	

Tabla 4 HU-02 Eliminar permisos de lectura y escritura a usuario.

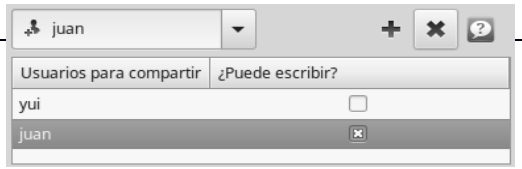
Historias de Usuarios	
Número: HU-02	**Nombre Historia de Usuario:** Eliminar permisos de lectura y escritura a usuario.
Modificación de Historia de Usuario: Ninguna	
Usuario: Aldy León García	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: Permitir eliminar permiso de escritura y / o lectura a un usuario.	
Observación: Para eliminar los permisos de un usuario primero este debe tener una regla anteriormente agregada, luego dar clic sobre el botón eliminar permiso.	
Prototipo de interfaz: 	

Tabla 5 HU-03 Modificar permisos de lectura y escritura a usuario.

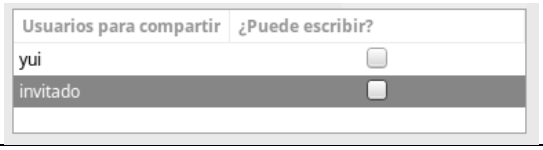
Historias de Usuarios	
Número: HU-03	**Nombre Historia de Usuario:** Modificar permisos de lectura y escritura a usuario.
Modificación de Historia de Usuario: Ninguna	
Usuario: Aldy León García	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: Permitirá modificar permisos de lectura y escritura a un usuario.	
Observación: Para modificar los permisos de un usuario primero este debe tener una regla anteriormente agregada, luego de seleccionar el usuario y modificar el permiso de escritura.	
Prototipo de interfaz: 	

Tabla 6 HU-04 Mostrar permisos de lectura y escritura de los usuarios.

Historias de Usuarios	
Número: HU-04	**Nombre Historia de Usuario:** Mostrar permisos de lectura y escritura de los usuarios.
Modificación de Historia de Usuario: Ninguna	

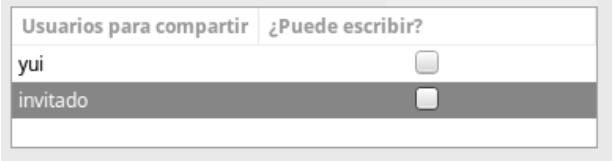
Usuario: Aldy León García	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos Reales:** 1 semana
Descripción: Permitirá mostrar los permisos de lectura y escritura de los usuarios.	
Observación: La aplicación permitirá que los permisos configurados se muestren junto a sus usuarios.	
Prototipo de interfaz: 	

Tabla 7 HU-05 Adicionar permisos de lectura y escritura a usuarios sin autentificar en el sistema.

Historias de Usuarios	
Número: HU-05	**Nombre Historia de Usuario:** Adicionar permisos de lectura y escritura a usuarios sin autentificar en el sistema.
Modificación de Historia de Usuario: Ninguna	
Usuario: Aldy León García	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en desarrollo: Baja	**Puntos Reales:** 1 semana
Descripción: Permite el acceso de usuarios sin autenticar.	
Observación: En la lista de usuarios estará el usuario invitado. Al seleccionar esta opción cualquier usuario no autorizado podrá leer y / o modificar los directorios compartidos.	
Prototipo de interfaz: 	

Tabla 8 HU-06 Compartir un directorio.

Historias de usuarios	
Número: HU-06	**Nombre Historia de Usuario:** Compartir un directorio.
Modificación de Historia de Usuario: Ninguna	
Usuario: Aldy León García	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 1 semana
Riesgo en desarrollo: Baja	**Puntos Reales:** 1 semana
Descripción: Modifica el estado compartido de un directorio.	
Observación: **Estado 1: Desactivado.** El directorio no ha sido compartido o se desea eliminar todas las modificaciones ocurridas sobre él. **Estado 2: Activado.** El directorio ha sido compartido o se desea aplicar modificaciones para que sea compartido.	
Prototipo de interfaz: 	

Tabla 9 HU-07 Modificar el nombre del directorio a compartir.

Historias de Usuarios	
Número: HU-07	**Nombre Historia de Usuario:** Modificar el nombre del directorio a compartir.
Modificación de Historia de Usuario: Ninguna	
Usuario: Aldy León García	**Iteración Asignada:** 2
Prioridad en Negocio: Baja	**Puntos Estimados: 1** semana
Riesgo en desarrollo: Baja	**Puntos Reales: 1** semana
Descripción: Permite establecer un nombre para el nuevo directorio compartido.	
Observación: Para modificar el nombre se debe seleccionar el campo de texto correspondiente a nombre del directorio compartido.	

Prototipo de interfaz:	

Tabla 10 HU-08 Adicionar un comentario al directorio compartido.

Historias de Usuarios	
Número: HU-08	**Nombre Historia de Usuario:** Adicionar un comentario al directorio compartido.
Modificación de Historia de Usuario: Ninguna	
Usuario: Aldy León García	**Iteración Asignada:** 2
Prioridad en Negocio: Baja	**Puntos Estimados: 1** semana
Riesgo en desarrollo: Baja	**Puntos Reales: 1** semana
Descripción: Permite establecer un comentario opcional para el nuevo directorio compartido.	
Observación: Para insertar un comentario se debe seleccionar y editar el campo de comentario.	
Prototipo de interfaz:	

2.5 Diseño del sistema propuesto.

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

epígrafe se explicará los principales elementos del análisis y el diseño tomados en cuenta para la implementación del sistema.

2.5.1 Arquitectura de software

De acuerdo al Software Engineering Institute (SEI), la arquitectura de software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos. Generalmente, en el proceso de desarrollo de una aplicación, suelen emplearse varios estilos arquitectónicos. Esto permite dar una mejor solución mediante el empleo de las ventajas que proporcionan cada uno de ellos.

Mediante el estudio del gestor de ficheros Nautilus, se determinó que no existe referencia bibliográfica que describiese el estilo arquitectónico usado en el gestor de ficheros. El análisis y el estudio de diferentes estilos arquitectónicos se evidencia el uso de la arquitectura estilos de descomposición modular donde el software se estructura en grupos funcionales muy acoplados. No existe una distinción rígida entre la organización del sistema y la descomposición modular, los componentes de los módulos son normalmente más pequeños que los subsistemas, lo cual permite usar estilos alternativos de descomposición. El diseño modular propone dividir el sistema en partes diferenciadas y definir sus interfaces. Una descomposición modular posee cualidades como independencia funcional, acoplamiento, cohesión, compresibilidad y adaptabilidad. El desarrollo del módulo Nautilus Share deberá regirse por la arquitectura definida en el gestor de ficheros Nautilus como se observa en la Ilustración 6.

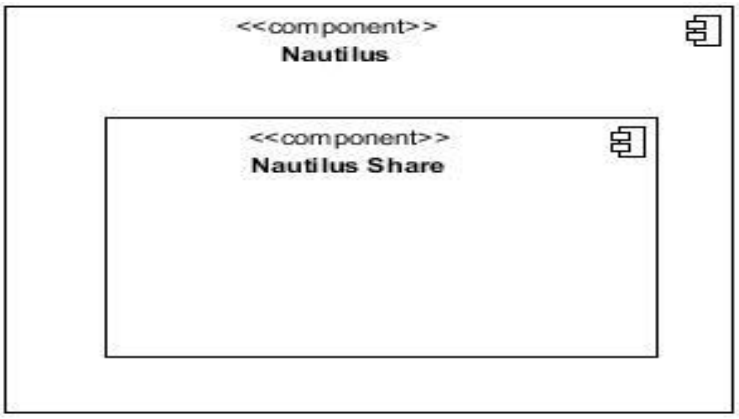


Ilustración 6: Descomposición Modular.

2.5.2 Patrones de diseño

Los patrones **GRASP** describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

A continuación, se explican los patrones utilizados:

Experto en información: El GRASP de experto en información es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).

Ejemplo de ello es la clase property-page.

PropertyPage *page;

page = g_new0 (PropertyPage, 1);

Controlador: El patrón controlador es un patrón que sirve como intermediario entre determinada interfaz y el algoritmo que la implementa. En el desarrollo de la propuesta de solución este patrón se evidencia en la clase share.

Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Un ejemplo de lo antes expuesto lo constituye la clase share que constituye un módulo sin dependencia a las demás clases del proyecto.

Los patrones **GOF,** acrónimo de Gang o Four, Pandilla de los Cuatro, en su traducción al español. Los patrones de diseño, conocidos como GoF se clasifican en tres grandes categorías basadas en su propósito: creadores, estructurales y de comportamiento. A continuación, se describen los que fueron utilizados en el desarrollo de la propuesta de solución.

Patrones de Creación: El objetivo de estos patrones es de abstraer el proceso de instanciar y ocultar los detalles de cómo son creados e inicializados los objetos.

Prototipo (Prototype): permite la creación de nuevos objetos al clonarlos de una instancia ya existente copiando el prototipo de esta clase. Este patrón se evidencia en la estructura de datos ShareInfo de la clase share.

Patrón Solitario (Singleton): Este patrón garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Es usado debido a la necesidad de tra-

bajar con el mismo objeto en distintos momentos. Este patrón se evidencia en la clase property-page-class.

2.6 Conclusiones parciales

En apoyo a la propuesta de solución se realizaron las historias de usuario definidas en 2 iteraciones permitiendo mayor control del tiempo y las funcionalidades con las que debe cumplir la aplicación. El uso de la arquitectura de descomposición modular y los patrones de diseño permite desarrollar la herramienta utilizando buenas prácticas de programación. El análisis y diseño de las funcionalidades propuestas ha permitido la obtención de artefactos como los requisitos funcionales, historias de usuario y el modelo de dominio para una mayor comprensión en la etapa de implementación.

Capítulo 3: Implementación y pruebas

Introducción

La etapa de las pruebas es de vital importancia en la implementación de un software. Según la metodología escogida para realizar la presente investigación es necesario una etapa de pruebas en cada una de las iteraciones en el desarrollo del software. En este capítulo se realizan las tareas de ingeniería y se exponen los estándares de código para ser implementada la solución propuesta. Se realizan pruebas internas de unidad, de integración y de aceptación para la demostración del funcionamiento de la aplicación, permitiendo evaluar la calidad del producto desarrollado y garantizar que cumpla con los requisitos definidos.

3.1 Plan de iteraciones

El plan de liberaciones contiene las historias de usuarios por iteraciones, especificando cuales de ellas serán implementadas en cada iteración del proceso. Como resultado del artefacto plan de liberación se encuentra la especificación de las entregas al cliente. El desarrollo se realiza en tres iteraciones según la prioridad definida en las historias de usuario.

Iteración 1: Se da cumplimiento a las historias de usuario con prioridad Alta (01, 02, 03, 04,05,06) que definen como la aplicación gestionará los permisos de lectura y escritura a los usuarios y el modo de activar o desactivar el directorio compartido con Nautilus Share, así como recuperar la información persistente del directorio compartido.

Iteración 2: Se da cumplimiento a las historias de usuario con prioridad Baja (07,08). El desarrollo de estas historias de usuario permite añadir personalización a la configuración para compartir directorios como el nombre del directorio y los comentarios.

Tabla 11: Plan de iteraciones.

Iteración	Descripción	Orden	Duración total de la iteración
1	Historias de Usuario de prioridad alta.	HU-01, HU-02, HU-03, HU-04, HU-05, HU-06,	6 semanas
2	Historias de Usuario de prioridad media.	HU-07, HU-8	2 semanas

3.2 Tareas de ingeniería

A continuación, se relacionan algunas de las tareas de ingenierías correspondientes a las historias de usuario, basándose en la prioridad que tienen.

Tabla 12 Tarea de Ingeniería 1 HU-01

Tarea de Ingeniería	
Número de la tarea: 1	Nombre de la historia de usuario: HU-01
Nombre Tarea: Estudiar el proceso para compartir directorios con net usershare.	
Tipo de Tarea: Investigación	Puntos Estimados: 5
Fecha Inicio: 03/02/2017	Fecha Fin: 10/02/2017
Programador Responsable: Aldy León García	
Descripción: Hacer un resumen de las funcionalidades de net usershare.	

Tabla 13 Tarea de Ingeniería 2 HU-01

Tarea de Ingeniería	
Número de la tarea: 2	Nombre de la historia de usuario: HU-01
Nombre Tarea: Programar el botón adicionar permiso de acceso a usuarios.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 2
Fecha Inicio: 10/02/2017	Fecha Fin: 12/02/2017
Programador Responsable: Aldy León García	
Descripción: Funcionalidad para agregar permisos de lectura y escritura a los usuarios.	

Tabla 14 Tarea de Ingeniería 3 HU-02

Tarea de Ingeniería	
Número de la tarea: 3	Nombre de la historia de usuario: HU-02
Nombre Tarea: Programar el botón eliminar permiso de acceso a usuarios.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 4
Fecha Inicio: 12/02/2017	Fecha Fin: 16/02/2017
Programador Responsable: Aldy León García	
Descripción: Funcionalidad para eliminar permisos de lectura y escritura a los usuarios.	

Tabla 15 Tarea de Ingeniería 4 HU-03

Tarea de Ingeniería	
Número de la tarea: 4	Nombre de la historia de usuario: HU-03
Nombre Tarea: Desarrollar la funcionalidad modificar permiso de acceso a usuarios.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 2
Fecha Inicio: 16/02/2017	Fecha Fin: 18/02/2017
Programador Responsable: Aldy León García	
Descripción: Funcionalidad para modificar permisos de lectura y escritura a los usuarios.	

Tabla 16 Tarea de Ingeniería 5 HU-04

Tarea de Ingeniería	
Número de la tarea: 5	Nombre de la historia de usuario: HU-04
Nombre Tarea: Desarrollar la funcionalidad mostrar los permisos de acceso a usuarios.	
Tipo de Tarea: Desarrollo	Puntos Estimados:1
Fecha Inicio: 18/02/2017	Fecha Fin: 19/02/2017
Programador Responsable: Aldy León García	
Descripción: Funcionalidad para mostrar los permisos de acceso a los usuarios.	

Tabla 17 Tarea de Ingeniería 6 HU-05

Tarea de Ingeniería	
Número de la tarea: 6	Nombre de la historia de usuario: HU-05
Nombre Tarea: Desarrollar la funcionalidad para compartir con usuarios sin autenticar.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 19/02/2017	Fecha Fin: 23/02/2017
Programador Responsable: Aldy León García	
Descripción: Funcionalidad que permite compartir con usuarios sin autenticar en el servidor Samba.	

3.3 Diagrama de Componentes

Un componente representa una parte de un sistema modular, desplegable y reemplazable, que encapsula la implementación y expone un conjunto de interfaces [24]. Podría ser, por ejemplo, código fuente, binario o ejecutable.

En la siguiente ilustración 7 se muestra la relación que posee el componente property-page.c con el resto de los componentes, para mostrar la interfaz a los usuarios y llevar el proceso de compartir un directorio. Para integrar el módulo Nautilus Share a Nautilus se utiliza el componente nautilus-extension.c, para obtener la información de los directorios se utiliza nautilus-info-provider.c y sus permisos con el componente nautilus-permissions.c que determina si el directorio se le deben aplicar nuevos permisos para ser compartido. Nautilus Share puede detectar dependencias incumplidas para su funcionamiento correcto, el módulo installSamba.c se encarga de esta tarea. Para completar el proceso de compartir un directorio el componente share.c se encarga de gestionar la información de los directorios y comunicar las acciones del usuario al servidor Samba.

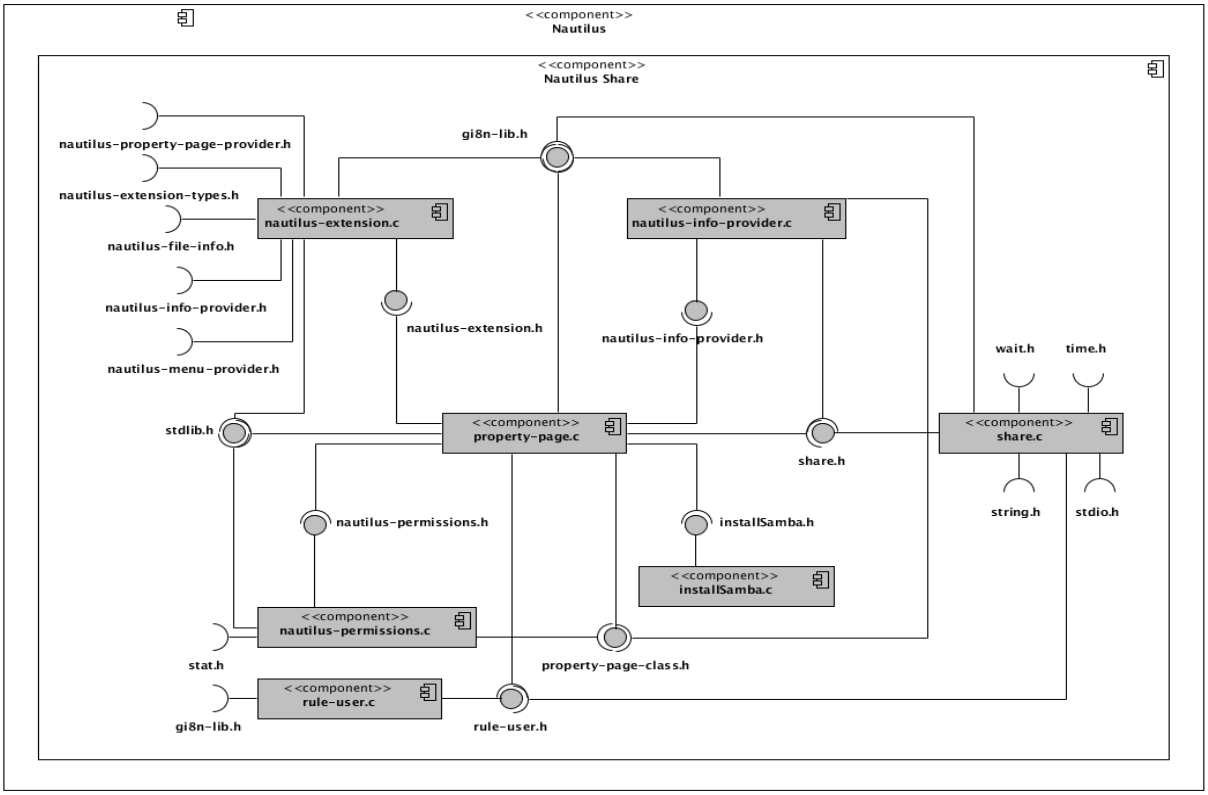


Ilustración 7: Diagrama de componentes de la solución propuesta.

3.4 Estándar de Codificación en C

Un estándar de codificación en informática son reglas que se rigen la escritura del código fuente; con el fin de homogeneizar la manera de codificar para que otros puedan entenderlo sin muchas dificultades. Parte esencial en cada estándar lo establecen las formas de escribir clases, métodos y variables. La propuesta de solución a ser implementada debe guiarse por las normas de codificación del proyecto GNOME dado que será implementando un módulo para el gestor de ficheros Nautilus. Estas indicaciones de codificación son inspiradas por los documentos *GTK's CODING-STYLE*, *The Linux Kernel's CodingStyle*, y el *GNU Coding Standards.* Cada uno de ellos es una variación de otros documentos con modificaciones particula-

res para cada proyecto y cultura. Una codificación basada en estándares permite el soporte a largo plazo del proyecto. [29]

La regla más importante: Cuando se escribe código se debe preservar el estilo del código original.

Usar **líneas de código** con 80 y 120 caracteres de longitud para permitir su visualización en la mayoría de los monitores con una fuente de texto legible.

Sangría del código: Estilo GNU. Cada nuevo nivel se le añade de sangría 2 espacios y las líneas que están en el mismo nivel comparten la misma sangría.

Funciones: El tipo de dato de retorno de la función debe ser puesto en una línea antes.

Parámetros en funciones: Se deben escribir separados por coma cada uno en una nueva línea.

Condicionales: No se debe comparar las variables booleanas con valores TRUE o FALSE usar siempre if (variable_condicional). En caso de varias variables en la condición poner cada una seguida de la operación lógica en líneas nuevas.

Llaves: Siempre que se escriba una llave adicionar un espacio antes nunca después.

3.5 Diagrama de Despliegue

Los diagramas de despliegue visualizan la distribución de los componentes de software en los nodos físicos. En él se especifican tres elementos fundamentales: Nodos, Dispositivos y Conectores. Los nodos describen los elementos de procesamiento con al menos un procesador, memoria y cualquier otro dispositivo. Los dispositivos son nodos que no tienen capacidad de procesamiento cuando se modela. Los conectores expresan el tipo de protocolo utilizado en el resto de los elementos del modelo. En el diagrama de despliegue de la solución propuesta (Ilustración 8) se aprecia como el módulo Nautilus Share comparte los directorios utilizando el servidor Samba por medio del protocolo TCP/IP.

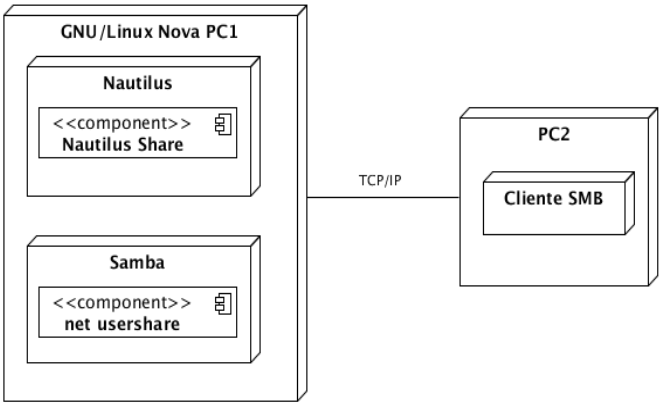


Ilustración 8: Diagrama de despliegue de la solución propuesta.

3.6 Estrategia para la prueba de software

La metodología de desarrollo AUP-UCI propone que el flujo de trabajo de las disciplinas, Requisitos, Análisis y Diseño, Implementación y Pruebas internas sean de forma iterativa. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración. [19]

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 evaluando los resultados. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad. [25]

3.6.1 Niveles de Pruebas

Cuando se le van a aplicar pruebas a un software, se tienen en cuenta una serie de objetivos en diferentes escenarios y niveles de trabajo, debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo. [31]

Pruebas unitarias: Se focaliza en ejecutar cada módulo para ser probado lo que provee un mejor modo de manejar la integración de las unidades en componentes mayores. Cada módulo puede ser un método, una clase o unidad mínima del código.

Pruebas de integración: Se encargan de identificar errores introducidos por la combinación de programas probados unitariamente. Verifica que las interfaces entre las entidades externas (usuarios) y las aplicaciones funcionan correctamente, las especificaciones de diseño sean alcanzadas y determina el enfoque para avanzar desde un nivel de integración de las componentes al siguiente.

Pruebas del sistema: Se focaliza en asegurar la apropiada navegación dentro del sistema, ingreso de datos, procesamiento y recuperación. Las pruebas del sistema deben enfocarse en requisitos que puedan ser tomados directamente de casos de uso o historias de usuario. El objetivo de estas pruebas es verificar el ingreso, procesamiento y recuperación apropiado de datos, y la implementación apropiada de las reglas de negocios.

Prueba de aceptación: La prueba de aceptación es ejecutada antes de que la aplicación sea instalada dentro de un ambiente de producción. La prueba de aceptación es generalmente desarrollada y ejecutada por el cliente o un especialista de la aplicación y es conducida a determinar como el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio. Con las mismas se pretende encontrar estos tipos de errores: funciones incorrectas o ausentes, errores en la interfaz, errores de rendimiento, errores de inicialización y de terminación, errores en estructuras de datos.

Prueba de usabilidad: Determina cuán bien el usuario podrá usar y entender la aplicación. Identifica las áreas de diseño que hacen al sistema de difícil uso para el usuario. La prueba de usabilidad detecta problemas relacionados con la conveniencia y practicidad del sistema desde el punto de vista del usuario.

3.6.2 Método de prueba

El método de **caja blanca** permite realizar verificaciones y validaciones directamente con el código fuente, para ello las **pruebas unitarias** son realizadas por el programador cada vez agrega una funcionalidad o método para asegurar la calidad del código. Las **pruebas de integración** se realizan para permitir que los datos son procesados correctamente y mostrados de la manera esperada. Es utilizada la herramienta de depuración *Valgrind,* un marco de instrumentación para la construcción de herramientas de análisis dinámico para detectar fugas de memoria y punteros no liberados.

El método de **caja negra** son las pruebas que se le realizan a la aplicación sin acceso al código fuente. Las **pruebas del sistema** se realizan en un primer momento en el entorno de desarrollo para detectar y corregir errores, en un segundo momento en el entorno de producción para verificar el correcto funcionamiento de todas las funcionalidades mostradas en las historias de usuario. Las **pruebas de usabilidad** verifican que la aplicación no presenta los siguientes problemas de usabilidad típicos:

1. El sistema es demasiado complejo y difícil de usar.
2. Es difícil instalar y entender el sistema.
3. La recuperación de errores es pobre y los mensajes de error no tienen significado.
4. Los procedimientos no son simples ni obvios.
5. La lógica y conveniencia de los botones, campos de texto y mensajes de ayuda deben ser testeados.

Para validar que el sistema cumple con el funcionamiento esperado en la metodología AUP-UCI se utilizan los casos de **prueba de aceptación**. Estos son chequeados por el cliente que dicta el grado de aceptación que tiene con respecto a las funcionalidades y el rendimiento del producto. Para la validación del módulo Nautilus Share del gestor de archivos Nautilus de la distribución cubana GNU/Linux Nova 6.0 se definieron los casos de prueba para las diferentes historias de usuario.

Tabla 18: Caso de prueba de aceptación 01.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 01	**Nombre Historia de Usuario:** Adicionar permisos de lectura y escritura a usuario.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Permite al usuario establecer un permiso de escritura o lectura a un directorio compartido.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba.	
Pasos de ejecución: 1. Seleccionar el directorio a compartir en el Nautilus. 2. Clic secundario, seleccionar la opción Compartir con usuarios Windows. 3. Activar compartir directorio. 4. Adicionar un usuario. 5. Se mantiene el checkbox sin marcar en la columna “Puede escribir?” Para permiso de Lectura. 6. Se marca el checkbox en la columna “Puede escribir?” Para permiso de Escritura. 7. Clic en el botón Crear compartido	
Resultado Esperado: El acceso al directorio compartido presenta los permisos garantizados.	
Evaluación de la Prueba:	

Tabla 19: Caso de prueba de aceptación 02.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 02	**Nombre Historia de Usuario:** Eliminar permisos de lectura y escritura a usuario.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Permite al usuario eliminar un permiso de escritura o lectura a un directorio compartido.	

Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba. Tener dos o más usuarios en la lista de acceso.
Pasos de ejecución: 1. Seleccionar el directorio a compartir en el Nautilus. 2. Clic secundario, seleccionar la opción Compartir con usuarios Windows. 3. Activar compartir directorio. 4. Adicionar un usuario. 5. Se mantiene el checkbox sin marcar en la columna “Puede escribir?” Para deshabilitar la Escritura. 6. Se selecciona un usuario y se da clic en eliminar para deshabilitar la Lectura. 7. Clic en el botón Crear compartido o Modificar compartido
Resultado Esperado: El acceso al directorio compartido presenta los permisos garantizados.
Evaluación de la Prueba:

Tabla 20: Caso de prueba de aceptación 03.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 03	**Nombre Historia de Usuario:** Modificar permisos de lectura y escritura a usuario.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Permite al usuario modificar los permisos de escritura o lectura a un directorio compartido.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba. Debe estar compartido el directorio con anterioridad.	
Pasos de ejecución: 1. Seleccionar el directorio a compartir en el Nautilus. 2. Clic secundario, seleccionar la opción Compartir con usuarios Windows.	

3. Activar compartir directorio. 4. Adicionar un usuario. 5. Se modifica el checkbox en la columna Puede escribir? Para modificar el permiso. 6. Clic en el botón Modificar compartido.
Resultado Esperado: El acceso al directorio compartido presenta los permisos garantizados.
Evaluación de la Prueba:

Tabla 21: Caso de prueba de aceptación 04.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 04	**Nombre Historia de Usuario:** Mostrar permisos de lectura y escritura de los usuarios.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Mostrar a los usuarios los permisos asignados con anterioridad.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba. Debe estar compartido el directorio con anterioridad.	
Pasos de ejecución: 1. Seleccionar el directorio a compartir en el Nautilus. 2. Clic secundario, seleccionar la opción Compartir con usuarios Windows.	
Resultado Esperado: Se recupera la lista de permisos.	
Evaluación de la Prueba:	

Tabla 22: Caso de prueba de aceptación 05.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 05	**Nombre Historia de Usuario:** Adicionar permisos de lectura y escritura a usuarios sin autentificar en el sistema.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Comprobar si es posible compartir con usuarios sin autenticación el directorio compartido.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba.	
Pasos de ejecución: 1. Seleccionar el directorio a compartir en el Nautilus. 2. Clic secundario, seleccionar la opción Compartir con usuarios Windows. 3. Activar compartir directorio. 4. Adicionar el usuario invitado. 5. Establecer permisos. 6. Clic en el botón Crear compartido	
Resultado Esperado: El acceso al directorio compartido se realiza sin autenticarse.	
Evaluación de la Prueba:	

Tabla 23: Caso de prueba de aceptación 06.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 06	**Nombre Historia de Usuario:** Compartir un directorio.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Comprobar si un usuario puede compartir un directorio.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba.	
Pasos de ejecución: 1- Seleccionar el directorio a compartir en el Nautilus. 2- Clic secundario, seleccionar la opción Compartir con usuarios Windows. 3- Establecer un nombre. 4- Añadir usuarios y permisos. 5- Añadir un comentario. 4- Clic en el botón Crear compartido	
Resultado Esperado: El directorio es compartido satisfactoriamente.	
Evaluación de la Prueba:	

Tabla 24: Caso de prueba de aceptación 07.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 07	**Nombre Historia de Usuario:** Recuperar el estado y configuración de un directorio compartido.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Recuperar la información de un directorio compartido.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba. Debe estar compartido el directorio con anterioridad.	

Pasos de ejecución: 1- Seleccionar el directorio a compartir en el Nautilus. 2- Clic secundario, seleccionar la opción Compartir con usuarios Windows.
Resultado Esperado: Se muestra el nombre del directorio compartido, los permisos y los comentarios correctamente.
Evaluación de la Prueba:

Tabla 25: Caso de prueba de aceptación 08.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 08	**Nombre Historia de Usuario:** Modificar el nombre del directorio a compartir.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Modificar el nombre del directorio a compartir.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba.	
Pasos de ejecución: 1- Seleccionar el directorio a compartir en el Nautilus. 2- Clic secundario, seleccionar la opción Compartir con usuarios Windows. 3- Establecer un nombre. 4- Clic en el botón Crear compartido	
Resultado Esperado: El nuevo nombre para el directorio compartido se muestra correctamente en el cliente del servidor Samba.	
Evaluación de la Prueba:	

Tabla 26: Caso de prueba de aceptación 09.

Casos de Pruebas de Aceptación	
Código Caso de Prueba: 09	**Nombre Historia de Usuario:** Adicionar un comentario al directorio compartido.
Nombre de la persona que realiza la prueba: Liss Betty López Rodríguez	
Descripción de la prueba: Adicionar comentarios al directorio compartido.	
Condiciones de ejecución: Se debe tener al menos un usuario en el servidor Samba.	
Pasos de ejecución: 1- Seleccionar el directorio a compartir en el Nautilus. 2- Clic secundario, seleccionar la opción Compartir con usuarios Windows. 3- Establecer un comentario. 4- Clic en el botón Crear compartido	
Resultado Esperado: El comentario se muestra correctamente en el cliente del servidor Samba.	
Evaluación de la Prueba:	

3.7 Resultados obtenidos en las pruebas de aceptación.

Las pruebas se centraron en el cumplimiento de las funcionalidades de todas las historias de usuario implementadas. Las No Conformidades (NC) se clasificaron en Alta (A) y Baja (B) en dependencia del impacto que tuvieran, generalmente las altas responden a errores técnicos relacionados directamente con la funcionalidad interna de la Historia de Usuario y las bajas tienden a ser errores ortográficos, validaciones entre otros errores de bajo impacto.

Se realizaron 3 iteraciones durante la implementación de la solución propuesta donde se obtuvieron varias no conformidades. En la primera iteración se detectaron errores en el momento de establecer y recuperar los permisos de los usuarios por el mal manejo de las estructuras de datos de las cuales 3 fueron de prioridad alta y 1 de prioridad baja las que se resolvieron completamente en las iteraciones siguientes. En la

segunda iteración fue detectada una no conformidad, al compartir un directorio el servidor Samba no permitía el acceso a ningún usuario desde el cliente; considerada de prioridad alta por ser objetivo principal de la historia de usuario. En la tercera iteración las no conformidades detectadas fueron de prioridad baja con respecto a la longitud de un nombre de un directorio para que fuera compatible con la mayoría de los clientes, además de la longitud máxima de texto en el área del comentario.

Fueron resueltas todas las no conformidades y para asegurar que el producto sea completamente funcional se implementaron una serie de validaciones para evitar la introducción de datos incorrectos.

3.8 Resultados de las pruebas de usabilidad.

Se le aplicó la lista de chequeo UCI al módulo Nautilus Share de Nautilus. Dicha lista está desarrollada por los especialistas del centro de calidad de la UCI. En la tabla 27 se evidencia que la implementación del módulo cumple con los 144 indicadores de usabilidad que presenta la lista de chequeo para la calidad del software.

Tabla 27. Resultados de prueba de usabilidad utilizando lista de chequeo

Categoría de Indicadores	**Indicadores**	**Correctos**
Visibilidad del sistema	17	17
Lenguaje común entre sistema y usuario	11	11
Libertad y control por parte del usuario	29	29
Consistencia y estándares	33	37
Estética y diseño minimalista	18	18
Prevención de errores	8	8

Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores	11	11
Ayuda y documentación	11	11
Flexibilidad y eficiencia de uso	6	6
Total	**144**	**144**

3.9 Conclusiones del capítulo

El estilo de codificación utilizado en la implementación de la aplicación, permitió mantener una uniformidad en la codificación y mayor entendimiento del código en futuras revisiones del mismo. La descripción de las tareas de la ingeniería y el plan de iteraciones a seguir en el desarrollo de la solución propuesta permitió reflejar el estudio realizado para poder cumplir cada una de las Historias de Usuario. Las pruebas de software realizadas permitieron evaluar el rendimiento de la aplicación, la conformidad con los requisitos definidos y la eliminación de las no conformidades encontradas en las iteraciones, comprobando de esta forma el correcto funcionamiento de la propuesta de solución.

Conclusiones

Con la culminación del presente trabajo de diploma se cumplieron cada uno de los objetivos trazados, distinguiéndose de manera general los siguientes aspectos:

- El estudio de los sistemas homólogos aportó las tendencias para compartir directorios como un nombre, un comentario, usuarios autorizados, permisos de acceso y uso de credenciales sin autenticación permitiendo plantear una propuesta de solución.
- La utilización de la herramienta net usershare para configurar el servidor Samba permitió compartir directorios sin privilegios de administración y gestionar los permisos de lectura y escritura de los usuarios.
- El desarrollo módulo Nautilus Share facilitó que los usuarios del sistema operativo Nova 6.0 puedan compartir directorios desde el gestor de archivos Nautilus de forma sencilla y eficiente.

Por todo lo antes mencionado se concluye que el trabajo desarrollado ha cumplido con los objetivos planteados al inicio de la investigación. Pues logró desarrollarse las funcionalidades para la herramienta Nautilus Share de manera satisfactoria, lo que constituye la manera más efectiva de darle solución al problema planteado.

Recomendaciones

Adicionar un módulo en el panel de control del sistema operativo cubano GNU / Linux Nova 6.0 para la gestión de los directorios compartidos con la herramienta Nautilus Share y gestionar los usuarios del servidor Samba.

Referencias bibliográficas

[1] A. Perpiñan, «GNU/Linux Básicamente». Fundación de código libre, 25-mar-2003.

[2] G. Valdés Lozano, «Software Libre». 28-nov-2007.

[3] «GNOME – An easy and elegant way to use your computer, GNOME 3 is designed to put you in control and get things done.» [En línea]. Disponible en: https://www.gnome.org/. [Accedido: 27-abr-2017].

[4] «Nautilus - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Nautilus. [Accedido: 27-abr-2017].

[5] A. S. Tanenbaum y A. S. Woodhull, *Operating systems: design and implementation: [the MINIX book]*, 3. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2006.

[6] «Permisos de archivos y directorios». [En línea]. Disponible en: https://www.linuxtotal.com.mx/index.php?cont=info_admon_011. [Accedido: 27-abr-2017].

[7] «Server Message Block Overview». [En línea]. Disponible en: https://msdn.microsoft.com/es-es/library/hh831795(v=ws.11).aspx. [Accedido: 13-abr-2017].

[8] «El protocolo SMB». [En línea]. Disponible en: http://webcache.googleusercontent.com/search?q=cache:3QoYYPRjeWcJ:ftp://169.158.189.34/pub/Sitio%2520Linux/Todo%2520linux/Linux-Windows/ch04s02.html+&cd=11&hl=es-419&ct=clnk&gl=cu. [Accedido: 13-abr-2017].

[9] «6. Compartir informacion. - PCPI 12/13 INFORMATICA MARIO». [En línea]. Disponible en: https://sites.google.com/site/pcpi1213informaticamario/home/modulos/3tic/05-redes-de-datos/6-compartir-informacion. [Accedido: 13-abr-2017].

[10] AEC, «Seguridad de la información». [En línea]. Disponible en: https://www.aec.es/c/document_library/get_file?uuid=b4fcf82c-5056-4ac4-aa15-915e74891bb2&groupId=10128. [Accedido: 13-abr-2017].

[11] «ISO 27001: ¿Qué significa la Seguridad de la Información?» [En línea]. Disponible en: http://www.pmg-ssi.com/2015/05/iso-27001-que-significa-la-seguridad-de-la-informacion/. [Accedido: 13-abr-2017].

[12] «Samba - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Samba. [Accedido: 12-abr-2017].

[13] «Servicios y demonios en Linux - Jesús Torres». [En línea]. Disponible en: https://jmtorres.webs.ull.es/me/2013/05/servicios-y-demonios-en-linux/. [Accedido: 12-abr-2017].

[14] «Funcionamiento de Samba». [En línea]. Disponible en: http://persoal.citius.usc.es/tf.pena/ASR/Tema_4html/node20.html. [Accedido: 12-abr-2017].

[15] «Gadmin-Samba - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Gadmin-Samba. [Accedido: 12-abr-2017].

[16] «Samba GUI Information». [En línea]. Disponible en: https://www.samba.org/samba/GUI/. [Accedido: 12-abr-2017].

[17] «[ubuntu] Shared folder confusion: nautilus share vs samba». [En línea]. Disponible en: https://ubuntuforums.org/showthread.php?t=1129647. [Accedido: 12-abr-2017].

[18] «net». [En línea]. Disponible en: https://www.samba.org/samba/docs/man/manpages-3/net.8.html. [Accedido: 12-abr-2017].

[19] T. Sánchez Rodríguez, «Metódología de desarrollo para la actividad productiva de la UCI.» UCI, 2015.

[20] «The GTK+ Project». [En línea]. Disponible en: https://www.gtk.org/. [Accedido: 12-abr-2017].

[21] «GtkWidget: GTK+ 3 Reference Manual». [En línea]. Disponible en: https://developer.gnome.org/gtk3/stable/GtkWidget.html. [Accedido: 12-abr-2017].

[22] rsequera20 Monografias.com, «Lenguaje C - Monografias.com». [En línea]. Disponible en: http://www.monografias.com/trabajos4/lenguajec/lenguajec.shtml. [Accedido: 12-abr-2017].

[23] «Apps/Builder - GNOME Wiki!» [En línea]. Disponible en: https://wiki.gnome.org/Apps/Builder. [Accedido: 12-abr-2017].

[24] E. Hernández Orallo, «El Lenguaje Unificado de Modelado (UML)». [En línea]. Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF. [Accedido: 13-abr-2017].

[25] R. Pressman, *S. 2002, Ingeniería de Software: Un enfoque práctico*. Madrid: McGraw Hill.

[26] «Software Design Tools for Agile Teams, with UML, BPMN and More». [En línea]. Disponible en: https://www.visual-paradigm.com/. [Accedido: 13-abr-2017].

[27] «Requerimientos funcionales y no funcionales», *Scribd*. [En línea]. Disponible en: https://www.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales. [Accedido: 13-abr-2017].

[28] J. Quijano, «Historias de usuario, una forma natural de análisis funcional», *Genbeta Dev*, 28-feb-2012. [En línea]. Disponible en: https://www.genbetadev.com/metodologias-de-programacion/historias-de-usuario-una-forma-natural-de-analisis-funcional. [Accedido: 13-abr-2017].

[29] «C Coding Style». [En línea]. Disponible en: https://developer.gnome.org/programming-guidelines/3.22/c-coding-style.html.en. [Accedido: 04-may-2017].

[30] «Interfaz de usuario - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Interfaz_de_usuario. [Accedido: 14-may-2017].

[31] «Niveles de prueba de software - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Niveles_de_prueba_de_software. [Accedido: 14-may-2017].

Anexos

Anexo 1 Permisos al listar un directorio con ls -l

Tipo de archivo y permiso	Numero de enlaces	Propietario	Grupo	Tamaño y Fecha de modificación	Nombre del archivo o directorio
drwxr-xr-x	3	yui	yui	4096 abr 11 18:57	vpworkspace
-rwxrwxr-x	1	yui	yui	17767 jun 27 2016	wihost

Anexo 2: Tipo de archivo, Primer carácter al extremo izquierdo

Carácter	Tipo de archivo
-	un guion representa un archivo común de texto o estructurado
d	representa un directorio
l	link, es un enlace o acceso directo
b	binario, un archivo generalmente ejecutable

Anexo 3: Representación de los permisos de archivo

Permisos de Archivo								
Propietario			Grupo			Otros		
rwx			rwx			r-x		
Read Lectura	Write Escritura	Execution Ejecución	Read Lectura	Write Escritura	Execution Ejecución	Read Lectura	Nada	Execution Ejecución

Anexo 4: Servidor Samba

Samba ofrece servicio de acceso remoto a ficheros e impresoras, autenticación y autorización de usuarios y servicio de resolución de nombres de dominio. [14] Cuenta con dos demonios principales:

smdb permite la compartición de archivos e impresoras sobre una red SMB, y proporciona autentificación y autorización de acceso para clientes SMB.

nmbd permite que el sistema Unix participe en los mecanismos de resolución de nombres propios de Windows (WINS), lo que incluye: anuncio en el grupo de trabajo, gestión de la lista de ordenadores del grupo de trabajo, contestación a peticiones de resolución de nombres, anuncio de los recursos compartidos.

Adicionalmente a los dos programas anteriores, Samba ofrece varias utilidades; algunas de las más relevantes son:

smbclient: Interfaz que permite a un usuario de un sistema Unix conectarse a recursos SMB y listar, transferir y enviar ficheros.

testparm: Verifica un error de sintaxis en los archivos de configuración

smbpasswd: Con este comando se administran los usuarios y contraseñas que podrán acceder al servidor Samba. Es necesario que los usuarios existan en el sistema y una vez añadidos con este comando pueden compartir los directorios que le pertenezcan. Su utilización se muestra en la Tabla 4.

pdbedit: Este comando acompañado de sus parámetros -L y -w permite mostrar los usuarios enumerados (-L) y sus campos en un formato compatible con smbpasswd (-w).

net: Utilidad de administración local y remota del servidor Samba usando la terminal de comandos.

Anexo 5: Gestionar usuarios del servidor Samba

Gestionar usuarios del Servidor Samba	
Acción	**Comando**
Añadir usuario	sudo smbpasswd -a <usuario>
Añadir usuario con campo de contraseña vacío	sudo smbpasswd -n <usuario>
Eliminar usuario	sudo smbpasswd -x <usuario>
Deshabilitar un usuario	sudo smbpasswd -d <usuario>
Habilitar un usuario	sudo smbpasswd -e <usuario>