Facultad 1

Herramienta para la transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova

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

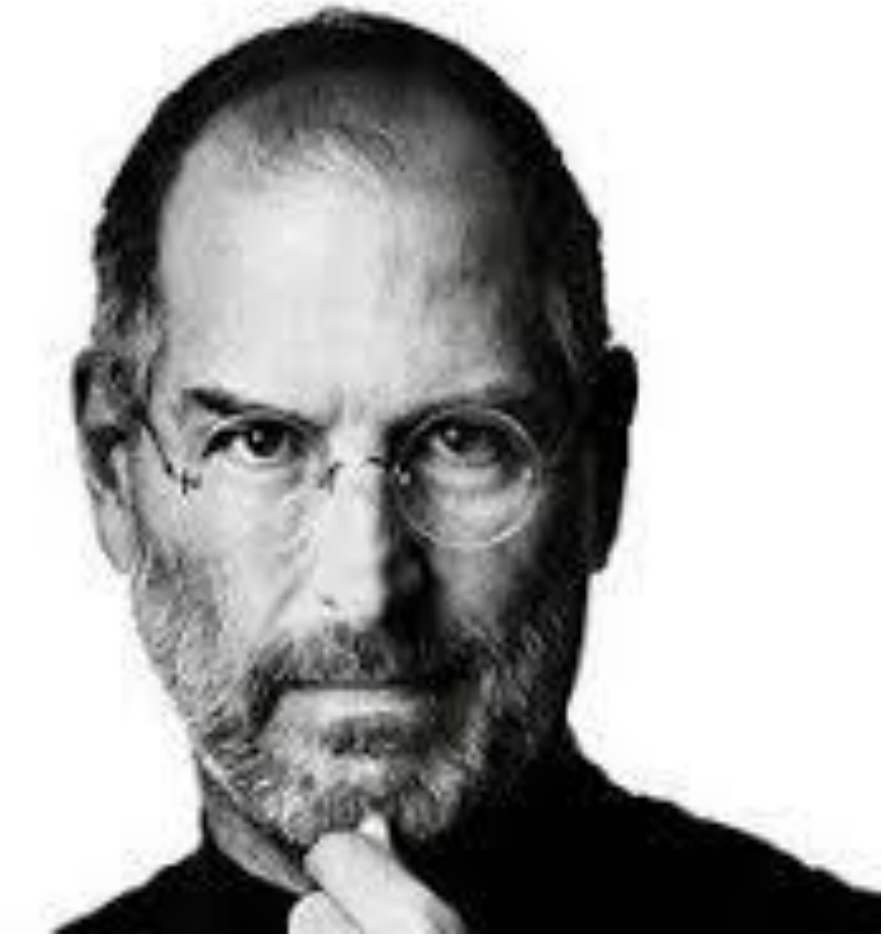
Autor: Daylin Yoennis Contreras Licea

Tutores: M Sc. Ruth Yurina Vega Cutiño

Ing. Enmanuel C. de la Cruz Mora

La Habana, noviembre del 2022

Año 64 de la Revolución



“Tu tiempo es limitado, así que no lo malgastes viviendo la vida de otra persona [...] No dejes que el ruido de las opiniones de otros apague tu propia voz interior”

Steve Jobs

DECLARACIÓN DE AUTORÍA

La autora del trabajo de diploma con título **"Herramienta para la transferencia de ficheros desde el navegador de archivos "Nautilus" en la distribución cubana GNU/Linux Nova 8.0"** concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declaran como únicos autores de su contenido. Para que así conste firman la presente a los 5 días del mes de diciembre del año 2022.

Daylin Yoennis Contreras Licea

_____

Firma del Autor

M Sc. Ruth Yurina Vega Cutiño

_____

Firma del Tutor

Ing. Enmanuel C. de la Cruz Mora

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Enmanuel Cristian de la Cruz Mora, graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2017. Forma parte del Centro de Software Libre donde se desempeña como programador del proyecto Nova. (ecdelacruz@uci.cu)

AGRADECIMIENTOS

Agradezco en primer lugar a toda mi familia. A mi mamá y mi papá por brindarme su apoyo incondicional, por todo el cariño y confianza y por darme fuerzas para vencer cualquier obstáculo durante mi etapa en la universidad, muchas gracias por confiar en mí y creer que podía llegar hasta aquí.

A mi hermano que lo quiero con todo mi corazón, gracias por apoyarme en todo momento y por estar pendiente de todos mis pasos.

A mi prima Osnelvis que la quiero mucho, y que se preocupa mucho por mí.

A mi novio Neiser por todo el tiempo que lleva en mi vida, se ha convertido en alguien muy especial e importante. Gracias por apoyarme y soportar mis malos días.

A mis tutores Ruth y Enmanuel por apoyarme en todo lo que estaba a su alcance, por su ayuda incondicional en la creación de este trabajo, por el tiempo dedicado y la experiencia.

A mis amistades y personas que he conocido aquí, que a muchas no voy a volver a ver nunca, pero que los recordaré siempre.

A mi mejor amigo Yandry que fue un gran apoyo en toda mi etapa universitaria y por estar siempre presente en los buenos y malos momentos

En fin, a todas esas personas y familiares que de una forma u otra me ayudaron y formaron parte de mi vida durante toda mi etapa universitaria.

DEDICATORIA

Dedico este trabajo a mis padres, mi hermano, mi novio y a toda mi familia en general por darme todo el apoyo que necesité durante toda mi etapa universitaria, por todo el amor que me han dado y por confiar en mí y creer que podía llegar hasta aquí.

RESUMEN

Las herramientas de transferencias de archivos son de suma importancia para enviar un archivo de un ordenador a otro. En el gestor de archivos Nautilus, presente en Nova, la transferencia de archivos se realiza a través del cliente de mensajería Evolution. A través de este se pueden enviar ficheros pequeños, tales como archivo de imagen o documentos. El principal aporte de esta investigación es permitir que los usuarios de la Distribución Cubana GNU/Linux Nova puedan contar con una herramienta para transferir archivos en Nautilus de una madera rápida y fácil, pues poseerá una interfaz de usuario muy simple y permitirá el envío de forma simultánea de varios archivos, o incluso carpetas con grandes volúmenes de archivos. La transferencia se realizará a alta velocidad, por lo que el envío de archivos grandes no debería suponer ningún problema.

PALABRAS CLAVE: archivo, Nautilus, ordenador, transferencia

ABSTRACT

File transfer tools are of the utmost importance to send a file from one computer to another. In the Nautilus file manager present in Nova, the file transfer is carried out through the Evolution messaging client. Through it you can send small files, an example of these can be: image files or documents. The main contribution of this research is to allow users of the Cuban GNU/Linux Nova Distribution to have a tool to transfer files in Nautilus quickly and easily, since it has a very simple user interface and It will allow the simultaneous sending of several files, or even folders with large volumes of files. The transfer will be done at high speed, so sending large files should not be a problem.

KEY WORDS: file, Nautilus, computer, transfer

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

OPINIÓN DEL(OS) TUTOR(ES)

<Contenido de la opinión de los tutores>

AVAL DEL CLIENTE

<Contenido del aval del cliente sobre la solución desarrollada>

INTRODUCCIÓN

Los sistemas operativos surgen como una necesidad para poder utilizar máquinas muy complejas en tiempos que se necesitaba personal muy especializado para poder operarlas (*Evolución de los sistemas operativos*, 2022).En la década de los 90, hace su aparición Linux, que posteriormente se uniría al proyecto GNU, un sistema operativo completamente libre, similar a UNIX, al que le faltaba para funcionar un núcleo funcional. Con su fiabilidad, estabilidad y eficiencia, Linux se ha convertido en la mejor solución para servidores de correo electrónico y servidores web (*Linux vs. Windows*, 2022).

Cuba es unos de los países que ha decidido migrar de un sistema operativo privativo a libre. En la actualidad con una distribución propia. La Universidad de las Ciencias Informáticas (UCI), es una de las instituciones encargadas del proceso de informatización del país. Esta desempeña un rol especializado en la investigación y desarrollo de aplicaciones informáticas para satisfacer las necesidades de migración a plataformas de código abierto. Entre los centros de producción que posee se encuentra CESOL[1] encargado del desarrollo, mantenimiento y soporte de la Distribución Cubana de GNU/Linux Nova, que persigue como objetivo brindar una distribución del sistema operativo estable y segura(León García, 2017).

Nova es un sistema operativo que brinda soporte a los protocolos de transferencia para compartir directorios con diferentes niveles de seguridad y formas de acceso. Un protocolo para la transferencia de archivos es una convención o una norma que controla o permite la transferencia de archivos entre dos ordenadores, esto se refiere al acto de transmisión de ficheros a través de una red informática. (Transferencia de archivos, 2022). Ejemplos de ellos son: el Protocolo de Transferencia de Archivos (FTP), Sistema de Ficheros en Red (NFS) y Bloque de Mensajes del Servidor (SMB) (León García, 2017).

En la actualidad, para realizar la transferencia de archivos entre ordenadores, se cuenta con la utilización de SAMBA, un conjunto de aplicaciones Linux, basadas en el protocolo SMB, que permiten compartir archivos en la red. Este complejiza mucho la simple finalidad de enviar un archivo pues

[1] CESOL: Centro de Software Libre.

requiere la instalación y configuración del servidor antes de su uso. Por lo que resulta ser un proceso largo y engorroso y poco factible para aquellos usuarios con pocos conocimientos sobre la transferencia de archivo(*Samba*, 2022).

También es utilizado el protocolo SSH/SCP que garantiza la transferencia segura de datos, pero solo entre dos ordenadores con sistema operativo Linux y que además tengan instalado SSH. Para su uso se necesita conocer la dirección IP, lo cual llega ser tedioso en redes que se utilicen la asignación de direcciones mediante el protocolo DHCP (*Dynamic Host Configuration Protocol*, Protocolo de Configuración Huésped Dinámico). Es preciso tener conocimiento, además, del usuario y contraseña, lo que representa un problema de seguridad a tener en cuenta, también o posee una interfaz y todo el proceso se realiza desde la terminal(Bernal, 2022).

El gestor de ficheros Nautilus con el que cuenta Nova, pertenece al proyecto GNOME, el mismo posee la capacidad de ampliar sus funcionalidades por medio de módulos adicionales, efectúa la transferencia de ficheros solo a través del correo electrónico “Evolution”. Esto trae consigo que para lograr una trasferencia satisfactoria es necesario que los usuarios, tanto el que envía como el que recibe; posean un correo electrónico activo y con espacio suficiente para recibir los archivos a enviar. El tamaño de los ficheros se ve limitado por la configuración del servidor, la trasferencia no se realiza de manera instantánea pues en ocasiones los correos demoran en llegar, el usuario que recibe no es notificado de que se le ha realizado la trasferencia y cualquier falla o error en el servidor del correo provocaría un fallo en la trasferencia. Todo esto conlleva que el usuario cause rechazo a la aplicación decida utilizar alguna otra vía de transferencia de archivos de un ordenador a otro desde la distribución GNU/Linux Nova.

Atendiendo a las deficiencias encontradas se identifica como **Problema de investigación:** ¿Cómo brindar soporte a la transferencia de ficheros entre ordenadores desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova? Para lo que se define como **Objeto de estudio:** el proceso de transferencia de ficheros entre ordenadores. Teniendo como **Campo de acción:** proceso de trasferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova

Para darle solución al problema planteado se enuncia como **Objetivo general:** Desarrollar una herramienta para la transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova.

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

1. Elaborar el marco teórico – metodológico de la investigación sobre el proceso de transferencia de ficheros entre ordenadores.

2. Diseñar una herramienta para la transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova.

3. Implementar una herramienta para mejorar la transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova.

4. Evaluar la herramienta para mejorar la transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova.

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

1 ¿Cuáles son los fundamentos teóricos sobre el proceso de transferencia de ficheros entre ordenadores?

2 ¿Qué elementos se deben tener en cuenta para diseñar una herramienta para mejorar la transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova?

3 ¿Qué componentes son necesarios implementar para obtener una herramienta que mejore la transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova?

4 ¿Qué pruebas y técnicas aplicar para evaluar la herramienta de transferencia de ficheros desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova?

Para facilitar el cumplimiento del objetivo propuesto de la investigación se usaron los siguientes **métodos científicos**:

Métodos teóricos:

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

Analítico-Sintético: Este método permitió el estudio de diferentes fuentes bibliográficas para extraer los elementos más importantes que se relacionan con las transferencias de archivos con Nautilus, posibilitaron la extracción de los elementos más significativos que sustentan la investigación.

Modelación: se utilizó para modelar los diagramas de requisitos, análisis y diseño e implementación en la construcción de la herramienta para la transferencia de ficheros con Nautilus.

Métodos empíricos:

Observación: Este método permitió analizar el funcionamiento de la transferencia de archivos con Nautilus y herramientas utilizadas dentro de los mismos.

Análisis documental: Es utilizado en la consulta de la literatura especializada publicada a nivel nacional e internacional, para extraer la información necesaria que permita la realización de la herramienta de transferencia de archivos desde el navegador de archivos Nautilus.

El presente trabajo cuenta con tres capítulos en los cuales se recogen los resultados de la investigación realizada:

Capítulo 1: "Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio": Contiene la fundamentación teórica de la investigación, se exponen las técnicas, tecnologías, metodologías y software en los que se apoya la solución propuesta para el problema. Se realiza un análisis crítico y valorativo de las tendencias actuales y el estado del arte de las herramientas de configuración en los entornos de escritorios.

Capítulo 2: "Análisis y diseño de la Herramienta para la transferencia de archivo entre ordenadores": En este capítulo se realiza el desarrollo ágil de la solución. Se explica toda la dinámica del proyecto en forma de historias de usuarios, prototipos de interfaz de usuario y algunos modelos auxiliares además del plan de liberación para las entregas intermedias.

Capítulo 3: "Implementación y pruebas del sistema": En este capítulo se detallan las pruebas realizadas al software, como de tipo caja negra, así como el impacto que tendrá el sistema una vez implantado.

CAPÍTULO I: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO

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 sobre la transferencia de ficheros entre ordenadores desde el navegador de archivos “Nautilus” en la distribución cubana GNU/Linux Nova 8. Se plantean la metodología y las herramientas necesarias para el desarrollo de la investigación.

1.1 Definición de conceptos

Archivo/Fichero:

Archivo o fichero es un conjunto organizado de unidades de información (bits) almacenados en un dispositivo, es un contenedor de información. Este es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene. La mayoría de los archivos que se utilizan contienen información (datos) en un formato determinado: un documento, una hoja de cálculo, un gráfico. El formato es la disposición de los datos dentro del archivo. (Archivo, 2018).

Para el intercambio de información mediante archivo es utilizado:

Transferencia de archivo:

Es la transmisión de un archivo de la computadora a través de un canal de comunicación de un sistema a otro. Normalmente, la transferencia de archivos está mediada por un protocolo de comunicaciones. En la historia de la computación, un gran número de protocolos de transferencia de archivos han sido diseñados para diferentes contextos. (*File Transfer Protocol (FTP) Definition*, 2022).

Para la gestión de archivo son utilizados los gestores de archivos entre los que se encuentra:

Gestor de ficheros Nautilus:

El gestor de archivos Nautilus ofrece una forma sencilla de ver, manipular y personalizar los archivos y carpetas del usuario. Proporciona un punto de acceso integrado a los archivos y aplicaciones. Permite al usuario crear, mostrar, gestionar, personalizar carpetas y documento, además, de navegar por el sistema de archivo (*Gestor de archivos: Nautilus*, 2022).

I.2 Análisis de herramientas de transferencias de archivos entre ordenadores

La transferencia de archivos es el proceso de copiar o mover un archivo de una computadora a otra a través de una red o conexión a Internet. Permite compartir, transferir o transmitir un archivo o un objeto de datos lógicos entre diferentes usuarios o computadoras tanto de forma local como remota (Techopedia, 2021).

1.2.1 Syncthing

Syncthing es un programa gratuito, de código abierto y portátil de sincronización continua de archivos para transferir archivos de forma segura entre dos o más computadoras en tiempo real. Cuenta con una interfaz de usuario, está disponible en todas las plataformas de escritorio, no requiere direcciones IP ni ninguna configuración avanzada y pone énfasis en la privacidad del usuario(*Las mejores herramientas de transferencia de archivos en Linux - HelpFulHelp*, 2022).

Características de **Syncthing**(*Syncthing*, 2022) :

Privado: ninguno de sus datos se almacena en ningún otro lugar que no sea en sus computadoras. No hay un servidor central que pueda verse comprometido, legal o ilegalmente.

Encriptado: toda la comunicación está protegida mediante TLS. El cifrado utilizado incluye el secreto directo perfecto para evitar que cualquier intruso obtenga acceso a sus datos.

Autenticado: cada dispositivo está identificado por un certificado criptográfico fuerte. Solo los dispositivos que ha permitido explícitamente pueden conectarse a sus otros dispositivos.

Poderoso. Sincroniza tantas carpetas como necesites con diferentes personas o solo entre tus propios dispositivos.

- Portátil: se configura y monitorea a través de una interfaz potente y receptiva accesible a través de su navegador. Funciona en Mac OS X, Windows, Linux, FreeBSD, Solaris, OpenBSD y muchos otros. Ejecútelo en sus computadoras de escritorio y sincronícelas con su servidor para realizar copias de seguridad.

- Simple: no necesita direcciones IP o configuración avanzada: simplemente funciona, a través de LAN e Internet. Cada máquina está identificada por un ID. Dale tu ID a tus amigos, comparte una carpeta y mira: UPnP funcionará si no quieres reenviar el puerto o no sabes cómo hacerlo.

1.2.2 EasyJoin

EasyJoin es una aplicación para compartir archivos multiplataforma para enviar mensajes, archivos, carpetas y direcciones URL a cualquier dispositivo conectado sin utilizar una conexión a Internet activa. Cuenta con una interfaz de usuario moderna, filtros de SMS, funcionalidad de dispositivo de control remoto, notificaciones de escritorio y gestión de llamadas(*Las mejores herramientas de transferencia de archivos en Linux - HelpFulHelp*, 2022).

Características de EasyJoin(Damian, 2018):

- EasyJoin es gratis para que todo el que quiera pueda descargarlo y utilizarlo. Cuenta también con una *versión Pro* disponible para aquellos que quieran disfrutar de sus características adicionales, que no son pocas.

- Se puede utilizar en casi cualquier plataforma como Windows, GNU / Linux o Mac, menos en iOS.

- Permite enviar archivos y carpetas a uno o más dispositivos a la velocidad más alta que permita nuestro ancho de banda.

- Podremos enviar mensajes entre nuestros dispositivos ubicados en una misma red, mientras economizamos nuestra tarifa de datos, ya que no requiere acceso a internet.

1.2.3 Rsync

Rsync es una herramienta de línea de comandos gratuita y de código abierto para crear copias de seguridad de forma remota y también funciona como una aplicación para compartir archivos. Con Rsync, los usuarios pueden transferir archivos de forma segura a otros dispositivos Linux y no Linux mediante SSH(*Las mejores herramientas de transferencia de archivos en Linux - HelpFulHelp*, 2022).

Características de Rsync(Rsync, 2022):

- Sincroniza de forma eficiente archivos y directorios entre dos sistemas Linux

- Es más rápido que el comando SCP (Secure Copy Protocol), ya que Rsync utiliza un protocolo de actualización remota que permite copiar únicamente las diferencias entre dos conjuntos de archivos. En la primera ejecución, copia todo el contenido de un archivo o directorio desde origen a destino, pero ejecuciones posteriores copia únicamente los bytes y los bloques que han sido modificados.

- Soporta la copia de enlaces simbólicos, y mantiene permisos, propietarios y grupos.

- Consume menos ancho de banda, ya que realiza la compresión y descompresión de los datos en ambos extremos con cada envío y recepción de los mismos.

1.2.4 Dukto

Programa gratuito diseñado para la transmisión de archivos e información a través de una conexión LAN. Considerado una aplicación multiplataforma pues está disponible en Windows, Linux, OS X, iOS, Android, BlackBerry, Symbian, Open Pandora y Maemo. Permite a los usuarios que sea muy fácil su uso y con mayor rapidez las transferencias de archivos, aunque sean muy pesados gracias a su diseño. Además de compartir archivos, puede incluso enviar mensajes de texto a otros dispositivos. Para que Dukto funcione, todos los dispositivos necesarios deben estar conectados a la misma red mediante WiFi o cable. Dukto buscará automáticamente los dispositivos disponibles en la LAN y los mostrará sobre él(apoklipsix, 2020).

¿Cómo funciona Dukto?

1. Dukto funciona de forma simple, tras instalarlo en los ordenadores deseados tan solo basta con ejecutarlo entre las PC que compartirán el fichero.

2. Al ejecutar Dukto el menú buddies contiene las PC que están ejecutando la aplicación lista para transferir. Si la PC del otro extremo no activa el Dukto, esta no saldrá en la lista.

3. Para transferir tan solo se debe escoger el equipo que desea, el Dukto por defecto identifica las PC por el nombre de las mismas. En el siguiente ejemplo escogimos la PC llamada Pochy.

4. Al seleccionar la PC se habilitará el menú que les permite enviar tanto mensajes, como algún texto que se encuentre en el portapapeles además de archivos independientes o alguna carpeta completa. Además, te permite arrastrar cualquier fichero o carpeta sobre cualquier PC de la lista para enviárselo directamente.

1.2.5 NDrop

Es una herramienta de transferencia de archivos, la cual posee las características (LIU, 2019/2022):

1. Sin autenticación, sin autorización, uso en red confiable.
2. Compatible con "Dukto" y "NitroShare"
3. Admite servidor de archivos HTTP (HFS)
4. Solo modo CLI, sin GUI. Dukto o NitroShare proporcionan una ventana GUI.
5. Transferencia de Archivos y Directorios.

1.3 Análisis de sistemas homólogos de herramientas de transferencias de archivos entre ordenadores

A partir de las herramientas de transferencias de archivos: Syncthing, EasyJoin, Rsync, Dukto y NDrop se realizó un análisis teniendo en cuenta los indicadores como: interfaz gráfica, transferencias sin internet, instalable en Nova, código fuente disponible, documentación abundante, integración con Nautilus, desktop, que nos permitirá conocer si cumplen con las necesidades existentes en la distribución de GNU/Linux Nova para escritorio de transferir ficheros.

Tabla 1 Aplicaciones homólogas (Fuente: Elaboración propia)

Herramientas	Interfaz grafica	Transferencias sin Internet	Instalable en Nova	Código fuente disponible	Documentación abundante	Integración con Nautilus	Desktop
Syncthing	Si	Si	Si	Si	Si	No	No

EasyJoin	No	No	Si	Si	Si	No	No
Rsync	Si	No	Si	Si	Si	No	No
Dukto	Si	Si	Si	No	Si	No	Si
NDrop	No	Si	Si	Si	No	No	Si

Tras el análisis de la tabla anterior, se concluye que ninguna de las aplicaciones existentes es de utilidad para los requisitos planteados, debido a que no se ajustan dado por la no integración con Nautilus, no están disponibles para la aplicación de desktop o no permiten la transferencia sin internet. Estas aplicaciones permitieron obtener las principales funcionalidades que debe contar este tipo de software que debe permitir transferir archivos y carpetas de un gran volumen, conocer los ip conectados a la red y debe poseer una interfaz gráfica.

1.4 Metodología de desarrollo de software

La metodología de desarrollo de software es el conjunto de técnicas y métodos que se utilizan para diseñar una solución de software informático. Es importante señalar que existen varias, de manera que es una decisión de cada equipo. Trabajar con una metodología es imprescindible por una cuestión de organización. No en vano, los factores tienen que estar ordenados y saber cómo se van a utilizar. Por otra parte, las metodologías también sirven para controlar el desarrollo del trabajo. Esto sirve para minimizar los márgenes de errores y anticiparse a esa situación. Otra ventaja de utilizar una metodología es que te hace ahorrar tiempo y gestionar mejor los recursos disponibles. Esto sucede tanto en metodologías a corto como a largo plazo. Cuando te decantes por un sistema, has de tener en cuenta este factor. Al final, uno de los elementos básicos es optimizar los recursos a tu alcance. Finalmente, hay que hacer referencia al valor añadido. Hay metodologías más costosas, efectivamente, pero que facilitan que el resultado final sea mejor(*Metodologías de desarrollo de software | Universitat Carlemany*, 2022). La metodología seleccionada para el desarrollo de la propuesta de solución es Variación de AUP para la UCI.

1.4.1 Variación de AUP para la UCI

La Universidad de las Ciencias Informáticas desarrolló una versión de la metodología de desarrollo de software AUP (Proceso Ágil Unificado), con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la universidad. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, nombrada Ejecución y agregándose también una nueva fase llamada Cierre (Yero Tarancón, 2015).

A continuación, se muestra una tabla con las fases de la metodología AUP-UCI:

Tabla 2 Fases de la variación de AUP para la UCI (Fuente: elaboración propia).

Fases	Descripción de las fases
Inicio	En el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. 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 la ejecución y se realizan las actividades formales de cierre del proyecto.

El presente trabajo de diploma se emplea todas las fases y pasará por las siguientes disciplinas propuestas en la metodología: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de Aceptación. Se escoge el escenario 4 debido a sus características para modelar la propuesta de solución, siendo este aplicable a proyectos que hayan evaluado el negocio a informatizar y logren un negocio bien definido como resultado. El cliente se considera parte del equipo de desarrollo definir los detalles de los requisitos y poder implementarlos satisfactoriamente, ejecutarlos y validarlos.

1.5 Lenguajes y herramientas para el modelado de la solución

A continuación, se muestran todas las herramientas y tecnologías utilizadas en el desarrollo de la solución propuesta, estas fueron seleccionadas por políticas del proyecto al que pertenece el sistema que se pretende desarrollar. Se fundamentan también otras características de importancia.

1.5.1 Lenguaje unificado de modelado

El Lenguaje Unificado de Modelado (UML, del inglés *Unified Modeling Language*) es un lenguaje de modelado visual que permite especificar, construir y documentar artefactos de un software. Se puede usar en las diferentes etapas del ciclo de vida de un proyecto. Incluye conceptos semánticos, notación y principios generales de un sistema. Contiene además construcciones organizativas para agrupar los modelos en paquetes, lo que permite dividir grandes sistemas en piezas de trabajo más simples. Este lenguaje es lo suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo en una empresa, diseño de la estructura de una organización y el diseño del hardware. Un modelo UML está compuesto por tres clases de bloques de construcción(de la O Barrientos, 2019):

- Elementos: los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etcétera).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: son colecciones de elementos con sus relaciones.

1.5.2 Herramientas de modelado

Visual Paradigm ayuda a los equipos de desarrollo de softwares a capturar los requisitos correctos y transformarlos en diseños precisos, lo que ayuda a los desarrolladores a crear el software adecuado según los requisitos. Que tiene como funciones la creación de diagramas y de wireframes. También tiene un editor para arrastrar y soltar, ya que es una herramienta de colaboración(*Visual Paradigm*, 2022).

Visual Paradigm versión 15.1 es una herramienta de modelado que soporta el UML y proporciona asistencia al equipo de desarrollo, durante todo el ciclo de vida de la elaboración de un software: análisis, diseños orientados a objetos, construcción, pruebas y despliegue. Se integra con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad. Genera código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.

Además, posibilita la obtención de diferentes informes a partir de la información introducida en la herramienta. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Agiliza la construcción de aplicaciones con calidad y a un menor costo. Posibilita la generación de bases de datos, transformación de diagramas de entidad-relación en tablas de base de datos, así como ingeniería inversa de bases de datos(de la O Barrientos, 2019).

1.6 Herramienta y lenguaje para la implementación

Una herramienta es un objeto creado para contribuir al desarrollo de software. Existen varias que se dedican a funciones específicas durante la construcción de un producto. En el presente epígrafe se describen las herramientas y tecnologías usadas en la implementación de la aplicación para la transferencia de archivos en el sistema operativo GNU/Linux Nova.

1.6.1 Lenguaje de programación

En la implementación de la propuesta de solución se utilizará el lenguaje de programación Python en su versión 3.10.4.

Python es un lenguaje de programación presente en multitud de aplicaciones y sistemas operativos. Podemos encontrarlo corriendo en servidores, en aplicaciones iOS, Android, Linux, Windows o Mac. Esto es debido a que cuenta con una curva de aprendizaje moderada ya que su filosofía hace hincapié en ofrecer una sintaxis de código legible(miteris, 2021).

Las principales características que tiene este lenguaje son(miteris, 2021):

- Programación Orientado a Objetos (POO): El código se encontrará estructurado por clases y objetos. Representa conceptos y situaciones cotidianas en el programa.
- Lenguaje interpretado: Los intérpretes que se utilizan en este leguaje son capaces de ejecutar programas a través de scripts propios, por lo que no resulta necesario compilar los mismos.
- Multiplataforma: Mientras que se cuente con un intérprete adecuado, Python puede ser capaz de ejecutarse en prácticamente cualquier sistema operativo.
- Lenguaje open source: No necesita de licencias de pago para su uso, puesto que es un lenguaje de código abierto.
- Ampliamente respaldado: Debido a sus interesantes características y funcionalidades posee un gran número de usuarios que emplean el lenguaje, facilitando así encontrar información sobre este.
- Tipado dinámico: Las variables en este lenguaje no necesitan especificar su tipo, las mismas adoptan un tipo automáticamente en función del valor que se les asigne mientras el lenguaje.

1.6.2 Herramienta de diseño de interfaz grafica

Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación(*Visual Studio Code, 2022*).

Características de Visual Studio Code

- **Multiplataforma:** Es una característica importante en cualquier aplicación y más si trata de desarrollo. Visual Studio Code está disponible para Windows, GNU/Linux y macOS.
- **IntelliSense:** permite ser más ágil a la hora de escribir código, ya que proporciona sugerencias de código y terminaciones inteligentes en base a los tipos de variables, funciones, etc. Con la ayuda de extensiones se puede personalizar y conseguir un IntelliSense más completo para cualquier lenguaje.

- **Depuración:** Incluye la función de depuración que ayuda a detectar errores en el código. También es capaz de detectar pequeños errores de forma automática antes de ejecutar el código o la depuración como tal.
- **Uso del control de versiones:** tiene compatibilidad con Git, por lo que puedes revisar diferencias o lo que conocemos con git diff, organizar archivos, realizar commits desde el editor, y hacer push y pull desde cualquier servicio de gestión de código fuente (SMC). Los demás SMC están disponible por medio de extensiones**.**

1.6.3 Herramienta de control de versiones

Un controlador de versiones es un sistema que nos permite guardar un registro de las modificaciones que realizamos sobre un fichero o conjunto de ficheros a lo largo del tiempo de tal manera que sea posible recuperar versiones específicas más adelante. Habitualmente se utiliza en entornos de desarrollo de software, pero puede resultar de gran utilidad para cualquier persona que necesite un control robusto sobre la tarea que está realizando(*5 softwares de control de versiones*, 2021). Algunos de los sistemas de control de versiones más famosos son Subversión (también conocido como SVN), Mercurial, CVS, Monotone y Git, este último es el que utiliza a lo largo de la investigación para el control de versiones. A continuación, se presenta una breve caracterización.

GIT

Git es una herramienta que realiza una función del control de versiones de código de forma distribuid, fue diseñada por Linus Torvalds**,** es muy potente, no depende de un repositorio central, es un software libre. Con ella podemos mantener un historial completo de versiones. Podemos movernos, como si tuviéramos un puntero en el tiempo, por todas las revisiones de código y desplazarnos una manera muy ágil. Es muy rápida. Tiene un sistema de trabajo con ramas que lo hace especialmente potente. En cuanto a la funcionalidad de las ramas, las mismas están destinadas a provocar proyectos divergentes de un proyecto principal, para hacer experimentos o para probar nuevas funcionalidades. Las ramas pueden tener una línea de progreso diferente de la rama principal donde está el core de nuestro desarrollo. En algún momento podemos llegar a probar algunas de esas mejoras o cambios en el código y hacer una fusión a nuestro proyecto principal, ya que todo esto lo maneja Git de una forma muy eficiente(*Qué es GIT y para qué sirve | OpenWebinars*, 2022).

Conclusiones del capítulo

En este capítulo se han abordado los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones: las relaciones existentes entre los principales conceptos asociados al dominio de la presente investigación permitieron una mayor comprensión de la propuesta de solución. El análisis de las diferentes herramientas de transferencias de ficheros, permitió determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución. La selección de la metodología, herramientas y tecnologías posibilitó obtener la base tecnológica de la propuesta de solución. Para guiar el proceso de desarrollo de la herramienta se adoptó como metodología de desarrollo de *software* AUP-UCI. Se definió utilizar la herramienta CASE Visual Paradigm 15.1 para el modelado de los artefactos del análisis y diseño de la solución. Por otra parte, para la implementación se utilizó Python 3.8 como lenguaje de programación.

CAPÍTULO 2: ANALISIS Y DISEÑO DE LA HERRAMIENTA PARA LA TRANSFERENCIA DE ARCHIVO ENTRE ORDENADORES

En el presente capítulo se identifican los principales requisitos funcionales y no funcionales necesarios para el correcto funcionamiento del sistema. Se muestran los artefactos generados por la metodología Variación de AUP para la UCI en el escenario 4 que documenta la investigación. Se aborda los elementos fundamentales referentes a la arquitectura de la aplicación y los patrones de diseño utilizados.

2.1 Propuesta de solución

Luego de identificadas las dificultades que existen para la transferencia de archivos entre dispositivos que posean el sistema operativo Nova, considerando además el resultado del análisis de herramientas homólogas se propone desarrollar una herramienta con las siguientes características:

- Una herramienta de escritorio que permita la transferencia de archivos entre computadoras, de una manera fácil y rápida.
- La aplicación tendrá una interfaz de usuario muy simple, donde el funcionamiento es sencillo y no deja lugar a errores. El usuario será capaz de saber en todo momento qué es lo que debe hacer. En el momento de realizar envíos, el usuario podrá añadirlos y guardarlos en un historial, para no repetir la información cada vez que se realice un envío.
- El programa a desarrollar no necesitará ninguna configuración, tampoco es necesario estar conectado a Internet, solo que los ordenadores entre los que se va a transferir archivos estén ubicados en la misma red de área local.
- Detectará automáticamente los clientes dentro de la red local en la que se está trabajando.
- Permitirá el envío de forma simultánea de varios archivos, o incluso carpetas con grandes volúmenes de archivos. La transferencia se realizará a alta velocidad, por lo que el envío de archivos grandes no debería suponer ningún problema.
- Mostrará las direcciones IP que se estén utilizando.
- El código fuente estará disponible en los repositorios de nova

2.1.1 Mapa conceptual

El mapa conceptual es un diagrama que ayuda a entender un tema en específico al visualizar las relaciones entre las ideas y conceptos. Por lo general, las ideas son representadas en nodos

estructurados jerárquicamente y se conectan con palabras de enlace sobre las líneas para explicar las relaciones (Lucidchart 2020).

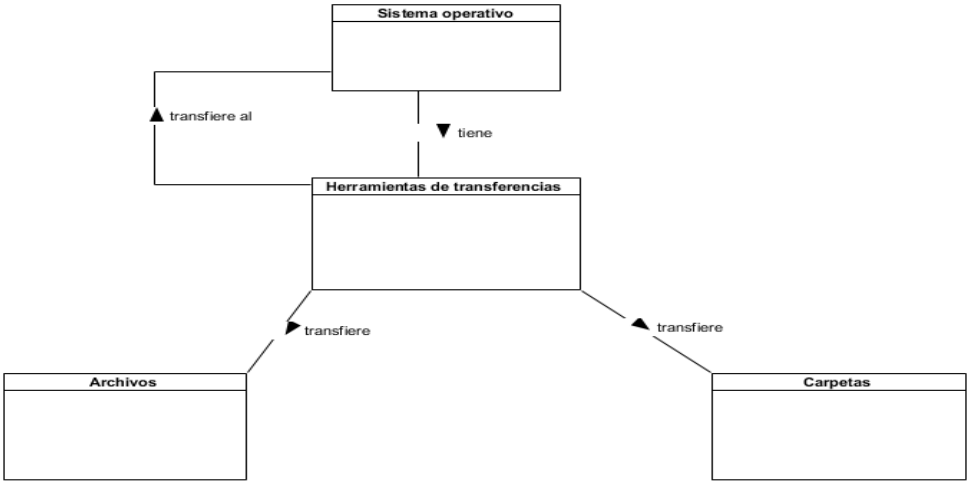


Figura 1 Mapa conceptual

Conceptos

Sistema operativo: es el software básico de una computadora que brinda una interfaz entre el resto de los programas del ordenador, los dispositivos hardware y el usuario.

Herramientas de transferencia: son las aplicaciones que se utilizan para la transmisión de un archivo del ordenador a través de un canal de comunicación de un sistema a otros.

Archivos: son un grupo de datos estructurados que son almacenados en algún medio y pueden ser usados por las aplicaciones.

Carpeta: es un medio para organizar programas y documentos en un disco y puede contener archivos y otras carpetas.

2.2 Requisitos

Los requisitos de software son la descripción de las características y las funcionalidades del sistema. Los requisitos nos comunican las expectativas de los consumidores de productos de software. Los requisitos pueden ser obvios o estar ocultos, conocidos o desconocidos, esperados o inesperados, desde el punto de vista del cliente(Roger S. Pressman, 2010).

2.2.1 Técnica de obtención de requisitos

Las técnicas de identificación de requisitos de software permiten identificar las necesidades de negocio de clientes y usuarios. Son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos de una aplicación, permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle, requieren ser adecuadamente orientadas para cubrir la información que se requiere capturar(Roger S. Pressman, 2010). A continuación, se especifican las técnicas usadas:

Tormenta de ideas

Se usó como técnica de extracción de requisitos la tormenta de ideas, que es una de las técnicas de obtención de requisitos más tradicional y consiste en una sesión de trabajo estructurada orientada para obtener la mayor cantidad de ideas posibles. Dicha tormenta de ideas se realizó con el autor de la investigación y especialistas de CESOL.

Observación

Este método permite observar la forma en que se llevan a cabo los procesos y, por otro, verificar que realmente se sigan todos los pasos especificados. Como sabemos, en muchos casos los procesos son una cosa en papel y otra muy diferente en la práctica. Los observadores experimentados saben qué buscar y cómo evaluar la relevancia de lo que observan(Guerra, 2017). Se observó el proceso de trasferencia de archivos llevado a cabo por los especialistas de CESOL y usuarios del sistema lo que permitió refinar e incorporar nuevos requisitos a los existentes.

2.2.2 Requisitos funcionales

Los requerimientos funcionales de un sistema, son aquellos que describen cualquier actividad que este deba realizar. Por lo general, estos deben incluir funciones desempeñadas por pantallas específicas, descripciones de los flujos de trabajo a ser desempeñados por el sistema y otros requerimientos de negocio, cumplimiento, seguridad u otra índole(pmoinformatica.com, 2017).

RF 1 Listar dispositivo: permite la visualización de la lista de los dispositivos conectados a la misma subred.
RF 2 Enviar por ip: permite enviar un archivo o una carpeta usando un ip.

RF 3 Enviar por local: permite enviar un archivo o una carpeta de forma local.
RF 4 Mostrar historial: permite previsualizar el historial de archivos enviados y recibidos.

2.2.3 Requisitos no funcionales

Los requerimientos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Suelen presentar dificultades en su definición dado que su conformidad o no conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición con criterios de aceptación que se puedan medir(pmoinformatica.com, 2015). Para lograr la satisfacción del cliente, y una buena calidad del sistema, se definieron los siguientes requisitos no funcionales, basado en lo establecido por la norma ISO 25000 Calidad del Producto de Software, específicamente la ISO/IEC 25010, la cual define las características de calidad que se tienen en cuenta al evaluar las propiedades de un producto de software:
RNF 1 Apariencia o interfaz externa: la aplicación mostrara un resaltado de sintaxis para facilitar las operaciones.
RNF 2 Portabilidad: la aplicación se ejecutará en consola por lo que será capaz de trasferir archivos hacia otros sistemas.
RNF 3 Confiabilidad: el sistema debe funcionar sin que se produzcan errores o con el mínimo posible que no afecte el correcto funcionamiento de la aplicación.
RNF 4 Usabilidad: el idioma de todas las interfaces de la aplicación será español.
RNF 5 Funcionalidad: El sistema ejecutará las operaciones indicadas en cada momento.
RNF 6 Software: un ordenador con Sistema Operativo GNU/Linux Nova.

2.2.4 Descripción de requisitos de software mediante Historia de Usuarios

Para la aplicación que se desea desarrollar se definió una historia de usuario por cada requisito funcional. A continuación, se define el diagrama de HU correspondientes a los requisitos funcionales.

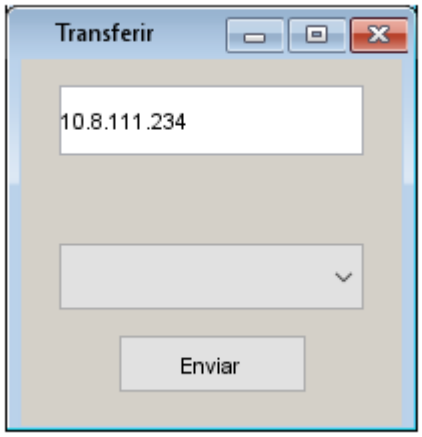
Tabla 3 Historia de Usuario 1 (Fuente: Elaboración Propia)

Historia de Usuario

Número: HU_1	**Requisito:** Listar dispositivo
Programador: Daylin Yoennis Contreras Licea	**Iteración Asignada:** NA
Prioridad: Media	**Tiempo Estimado:** 60 h
Riesgo en Desarrollo: La no visualización de los dispositivos conectados a la subred.	**Tiempo Real:** 25 h
Descripción: Luego de seleccionar el archivo o carpeta a enviar y buscar la opción de enviar a submenu1 se debe visualizar la interfaz con la opción se seleccionar los ip conectados a la subred.	
Prototipo de interfaz:	

Tabla 4 Historia de Usuario 2 (Fuente: Elaboración Propia)

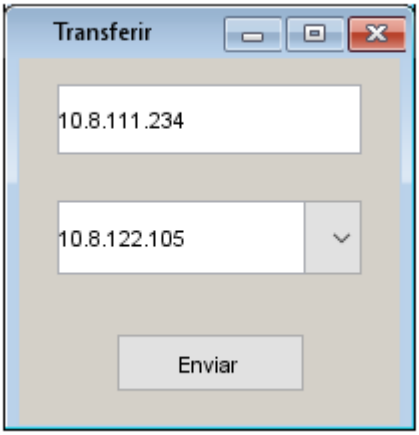
Historia de Usuario	
Número: HU_2	**Requisito:** Enviar por ip
Programador: Daylin Yoennis Contreras Licea	**Iteración Asignada:** NA
Prioridad: Alta	**Tiempo Estimado:** 60 h
Riesgo en Desarrollo: Que el sistema no permita introducir el ip.	**Tiempo Real:** 30 h
Descripción: Luego de seleccionar el archivo o carpeta a enviar y buscar la opción de enviar a submenu1 se debe visualizar la interfaz con la opción de introducir el ip.	



Prototipo de interfaz:

Tabla 5 Historia de Usuario 3 (Fuente: Elaboración Propia)

Historia de Usuario	
Número: HU_3	**Requisito:** Enviar por local
Programador: Daylin Yoennis Contreras Licea	**Iteración Asignada:** NA
Prioridad: Alta	**Tiempo Estimado:** 60 h
Riesgo en Desarrollo: El sistema no permita la selección de los ip conectados a la subred	**Tiempo Real:** 30 h
Descripción: Luego de seleccionar el archivo o carpeta a enviar, buscar la opción de enviar a submenu1 se debe visualizar la interfaz con la opción se seleccionar los ip conectados a la subred, luego seleccionar el ip deseado y enviar.	



Prototipo de interfaz:

Tabla 6 Historia de Usuario 4 (Fuente: Elaboración Propia)

Historia de Usuario	
Número: HU_4	**Requisito:** Mostrar Historial
Programador: Daylin Yoennis Contreras Licea	**Iteración Asignada:** NA
Prioridad: Baja	**Tiempo Estimado:** 60 h
Riesgo en Desarrollo: El sistema no permita la visualización de los archivos o carpetas enviadas o recibidas.	**Tiempo Real:** 15 h
Descripción: Luego de a ver enviado o recibido los archivos o carpetas debe mostrar el historial de los archivos o carpetas enviadas y recibidas.	
Prototipo de interfaz:	

2.3 Análisis y diseño

El análisis y el diseño es una etapa muy importante dentro del proceso de construcción de un software ya que permite una mejor compresión del problema, generación de conceptos de solución y selección y desarrollo de la mejor alternativa. Es un momento donde 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 para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos (Análisis y diseño, 2019).

2.3.1 Diseño Arquitectónico

El estándar IEEE[2] define la arquitectura de software como la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (Garlan, y otros, 1993). Según Roger Pressman: “En su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan” (Pressman, 2010).

Para la realización de esta herramienta se propone el patrón arquitectónico n-capas. La arquitectura en capas consta en dividir la aplicación en capas, con la intención de que cada capa tenga un rol muy definido(*Arquitectura en Capas*, 2021). Esta herramienta va a estar conformada por dos capas, la capa controladora que es la que crea la interfaz y gestiona los procesos y la capa vista que es la que envía los datos del ip a la capa controladora para realizar la transferencia.

2.3.2 Modelo de datos

El modelado de datos es el proceso de documentar un diseño de sistema de software complejo como un diagrama de fácil comprensión, usando texto y símbolos para representar la forma en que los datos necesitan fluir. El diagrama se puede utilizar como un mapa para la construcción de un nuevo software o para la reingeniería de una aplicación antigua(*¿Qué es Modelado de datos?*, 2022).

---

[2] IEEE: Del inglés Institute of Electrical and Electronics Engineers. Traducido al español Instituto de Ingeniería Eléctrica y Electrónica

2.3.3 Modelado del diseño

El diseño es la forma exacta en la que un requisito del cliente se puede convertir en un sistema o producto de software terminado. Crea una representación o modelo del software de forma detallada(Roger S. Pressman, 2010).

Diagrama de clase de diseño

Un diagrama de clase es un tipo de diagrama UML que describe un sistema visualizando los diferentes tipos de objetos dentro de un sistema y los tipos de relaciones estáticas que existen entre ellos. También ilustra las operaciones y atributos de las clases(Diagramas de Clase, 2020). A continuación, se muestra el diagrama de clase.

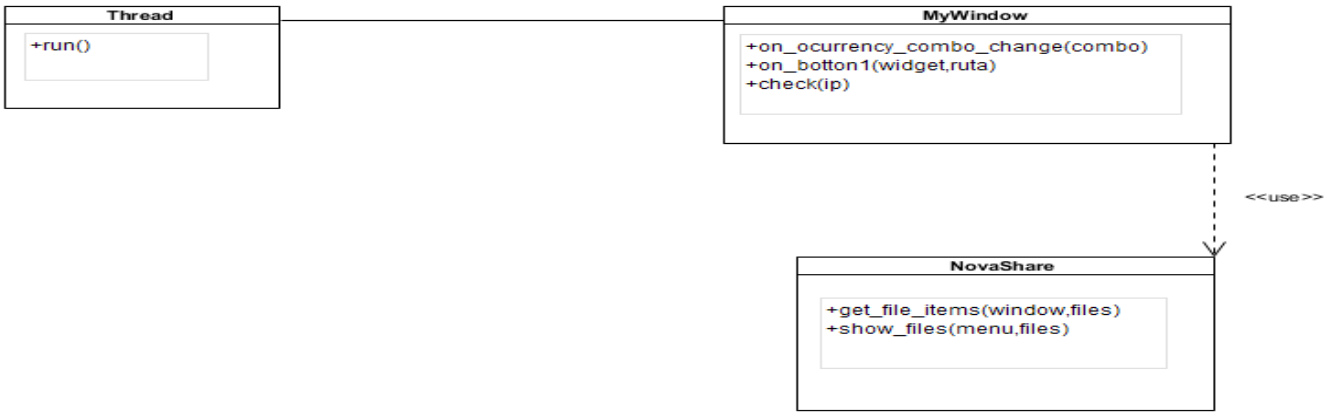


Figura 2 Diagrama de Clase (Fuente: Elaboración Propia)

Patrones del diseño de software

Los patrones de diseño o design patterns, son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error(Miriam, 2020)**.**

Los patrones de diseño a utilizar para el desarrollo del sistema, son los generales de software para la asignación de responsabilidades por sus siglas en inglés GRASP (General Responsibility Assignment Software Patterns, Patrones de Software de Asignación de Responsabilidad General) son guías o principios que sirven para asignar responsabilidades a las clases(Patrones GRASP, 2022).

- Experto:

El GRASP de experto en información es el principio básico de asignación de responsabilidades. Nos 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 obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento)(Patrones GRASP, 2022).Este patrón se utiliza en la clase MyWindows, que es la clase necesaria para mostrar los datos.

- Bajo acoplamiento:

Es el patrón que define cómo mantener bajas dependencias, bajo impacto al cambio e incrementar la reutilización. Asignar una responsabilidad de manera que el acoplamiento sea bajo. Un componente con bajo acoplamiento no mantiene fuertes dependencias con otros componentes(Patrones GRASP, 2022).Este patrón se utiliza en la clase Hilo, la cual se relaciona con la clase MyWindows para cumplir con sus funciones necesarias.

- Creador:

El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases(Patrones GRASP, 2022). El patrón se evidencia en la clase MyWindows, la cual tiene responsabilidad de crear instancia de la clase Hilo.

Patrones GoF

Gang-of-Four (GoF), también conocidos como la ‘pandilla de los cuatro’ son patrones de diseño utilizados en situaciones muy frecuentes debido a que se basan en la experiencia acumulada al resolver problemas reiterativos. Además, favorecen la reutilización del código(de la O Barrientos, 2019). A continuación, se describe el patrón GoF utilizado en la solución propuesta:

- **Abstract Factory**

En este patrón, una interfaz crea conjuntos o familias de objetos relacionados sin especificar el nombre de la clase(Miriam, 2020). En la figura 3 se visualiza la creación de dos objetos que son el box y el progressbar, donde el box lo q hace es importar de la librería GtK y agregar la funcionalidad

pack_stort que lo que hace es transferir paquete dentro del cual se le pasa atributo que son el objeto progressbar y dos bulianas.

```
class MyWindow(Gtk.Window):
  def __init__(self,files):
    super().__init__(title="Enviamos")

    self.box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
    self.add(self.box)

    self.progressbar = Gtk.ProgressBar()
    self.box.pack_start(self.progressbar, True, True,1)
```

Figura 3 Patrón Abstract Factory (Fuente: Elaboración Propia)

- Singleton:

Un patrón Singleton en python es un patrón de diseño que le permite crear solo una instancia de una clase, a lo largo de la vida útil de un programa. El uso de un patrón singleton tiene muchos beneficios (Miriam, 2020).En la figura 4 se visualiza donde el método *show_files* crea una instancia de la clase *MyWindows*.

```
def show_files(self,menu,files):
  win = MyWindow(files)
  win.connect("destroy", Gtk.main_quit)
  win.show_all()
  Gtk.main()
```

Figura 4 Patrón Singleton (Fuente: Elaboración Propia)

Conclusiones del capítulo

En este capítulo se han abordado los elementos del análisis y diseño de la herramienta de transferencia desde el navegador de archivos Nautilos, arribando a las siguientes conclusiones: los requisitos funcionales y no funcionales obtenidos a partir del proceso de identificación de los requisitos, sirvieron de guía para desarrollar las distintas funcionalidades y de este modo satisfacer las necesidades detectadas. La utilización de los patrones de diseño permitió identificar aspectos importantes de la estructura del diseño de la herramienta propuesta, lo que garantizó una mayor organización e hizo el código más legible. Las descripciones de las historias de usuario y la elaboración de diagrama de clases del diseño posibilitaron una mejor comprensión del funcionamiento de la propuesta de solución. Adoptar la arquitectura de software n-capas con 2 capas, permitió una propicia organización del sistema a implementar.

CAPÍTULO 3: IMPLEMENTACION Y PRUEBAS DEL SISTEMA

El presente capítulo se enfoca en la implementación de la herramienta a partir de los resultados del Análisis y diseño en el capítulo anterior. Se elaboran los diagramas de componentes y de despliegue y se define los estándares de codificación a utilizar en la construcción de la solución. Se realizan pruebas de software con el objetivo de descubrir y corregir errores y se evalúa la propuesta de solución.

3.1 Implementación

Se implementaron las Historias de Usuario en tres iteraciones, repasando y corrigiendo los detalles con cada una. Al culminar el proceso se realizó la revisión del plan de iteraciones y se corrigieron los cambios identificados. Específicamente las pruebas de software permiten evaluar las soluciones y determinar el nivel de calidad que poseen, por lo que se debe definir un proceso que se pueda emplear en el entorno de desarrollo de aplicaciones informáticas en la universidad.

- **Iteración No.1:** Se implementó la HU Listar Dispositivo.
- **Iteración No.2:** Se implemento la HU Enviar por Ip y Enviar por Local.

Estándar de codificación

Para Python existen normas para la mejora de su codificación, los PEPs (Python Enhancement Proposals). De esta estos estándares el PEP8, es “Guía de estilo para el código Python” (PEP 8 – Style Guide for Python Code | peps.python.org).

1. Sangría

- Use 4 espacios por nivel de sangría.

2. Espacios

- Los espacios son el método de sangría preferido.
- Las pestañas deben usarse únicamente para mantener la coherencia con el código que es ya sangrada con tabulaciones.
- Python no permite mezclar tabulaciones y espacios para la sangría.

3. Codificación del archivo origen
    - El código en la distribución central de Python siempre debe usar UTF-8 y no debe tener una declaración de codificación.
4. Importaciones
    - Las importaciones generalmente deben estar en líneas separadas
    - Las importaciones deben agruparse en el siguiente orden:
    - Importaciones de biblioteca estándar.

- Importaciones de terceros relacionados.
- Importaciones específicas de bibliotecas/aplicaciones locales.

3.2 Diagrama de componentes

Los diagramas de componentes muestran las interacciones y relaciones de los componentes de un modelo. Entendiéndose como componente a una clase de uso específico, que puede ser implementada desde un entorno de desarrollo, ya sea de código fuente, binario o ejecutable, dichos componentes poseen tipo, y están unidos mediante relaciones de dependencia (Roger. S. Pressman, 2010). A continuación, se presenta el diagrama de componentes del sistema.

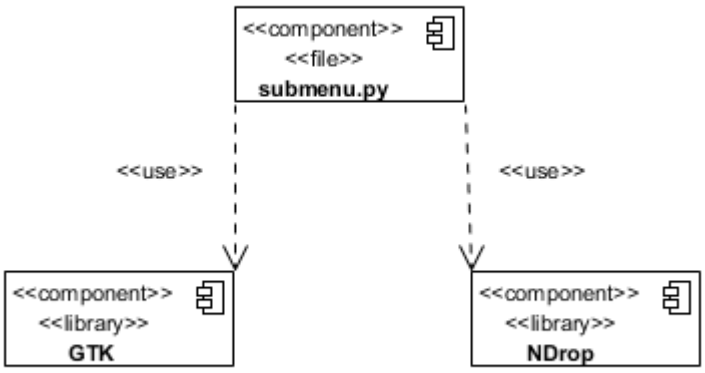


Figura 5 Diagrama de Componentes (Fuente: Elaboración Propia)

3.3 Diagrama de despliegue

El modelo de despliegue se realiza como parte de la implementación para describir la distribución física del sistema. Establece la correspondencia entre la arquitectura lógica, los procesos y nodos. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. Los nodos poseen relaciones que representan medios de comunicación entre ellos (Roger. S. Pressman, 2010). Se refleja en este artefacto los protocolos de comunicación mediante los cuales se comunican los nodos respectivos.

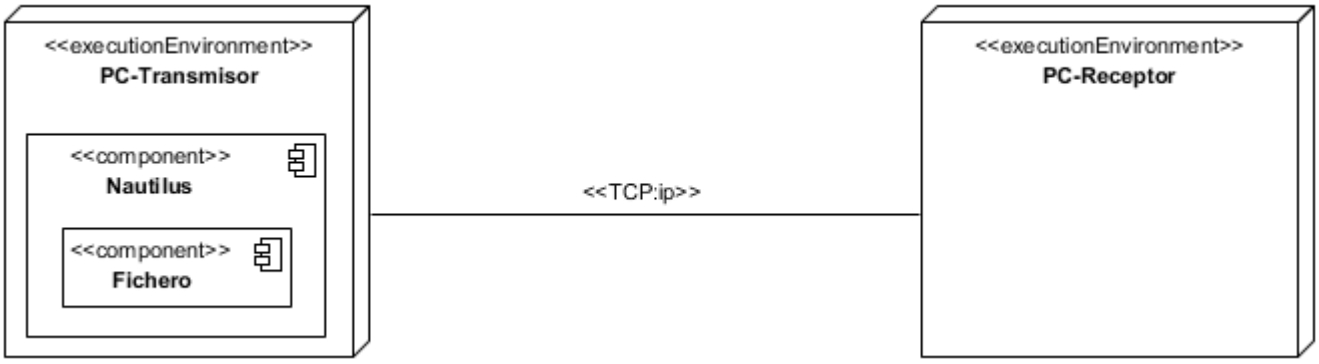


Figura 6 Diagrama de despliegue (Fuente: Elaboración Propia)

3.4 Interfaz gráfica de usuario

A continuación, se muestra una representación de las interfaces graficas que se tuvo como resultado en el proceso de desarrollo del software, las demás se muestran en el Anexo 2.

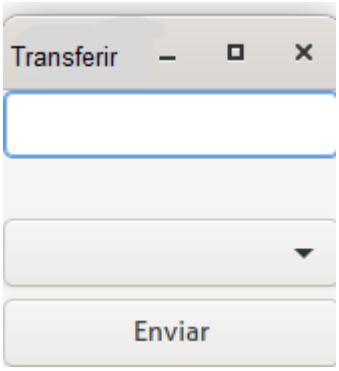


Figura 7 Interfaz Gráfica 1 (Fuente: Elaboración Propia)

3.5 Pruebas de software

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas. Es un proceso de ejecución de un programa con la intención de descubrir un error, no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

Las pruebas de software comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo.

3.5.1 Estrategias de prueba

Una estrategia de prueba del software integra los métodos de diseño de casos de prueba en una serie bien planteada de pasos que va a converger en la eficaz construcción del software. Dicha estrategia debe ser lo suficientemente flexible como para promover un enfoque personalizado, y al mismo tiempo lo adecuadamente rígido como para originar una planeación razonable y un seguimiento administrativo del avance del producto (Pressman 2010).

Pressman plantea las siguientes estrategias de prueba:

- Pruebas de unidad
- Pruebas de integración
- Pruebas de validación
- Pruebas de sistema

Las pruebas de unidad se centran en cada unidad del software, tal como se implementó en el código fuente. La prueba de integración se enfoca al diseño y la construcción de la arquitectura del software. La prueba de validación permite validar los requisitos establecidos como parte del análisis de requisitos de software, comparándolos con el sistema que ha sido construido. Finalmente se prueba el software como un todo empleando la prueba del sistema (Pressman 2010).

3.5.1.1 Pruebas de unidad

El objetivo de estas pruebas es ejecutar un código fuente al llamar directamente a los métodos de una clase pasándole a estos los parámetros adecuados.

Las pruebas de unidad descomponen las funciones del programa en comportamientos comprobables discretos que se pueden probar como unidades individuales. Están destinadas a verificar las unidades más pequeñas del software. Se aplican a las funcionalidades para verificar que los flujos de control y datos están cubiertos y funcionan tal como se espera. La prueba de unidad siempre está orientada a caja blanca (Pressman 2010).

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que (Pressman 2010):

- Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

La prueba de unidad que se realiza hace uso del método de caja blanca y de la técnica del camino básico. El método del camino básico permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño procedural y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

A continuación, se realiza la técnica del camino básico a un fragmento del código.

```
def on_enviar(self, widget,ruta):
   #subprocess.run(["ndrop", "--mode", "dukto", "--send", "10.53.4.75", "/home/nova/pepe"])
   ip =str(self.entry.get_text())
   resultado = self.check(ip)
   if resultado == 1:
        self.label.set_text("Ip Correcto")
        running = True
        while running:
            subprocess.run(["ndrop", "--mode", "dukto", "--send", ip, ruta])
            activo.clear()
            break
   else:
        self.label.set_text("IP Incorrecto")
```

Figura 8 Fragmento de Código de la Prueba de Unidad (Fuente: Elaboración Propia)

Primero se enumera las líneas de código para luego realizar el grafo de flujo que describe el flujo de control lógico empleando nodos y aristas, como se muestra en la siguiente figura.

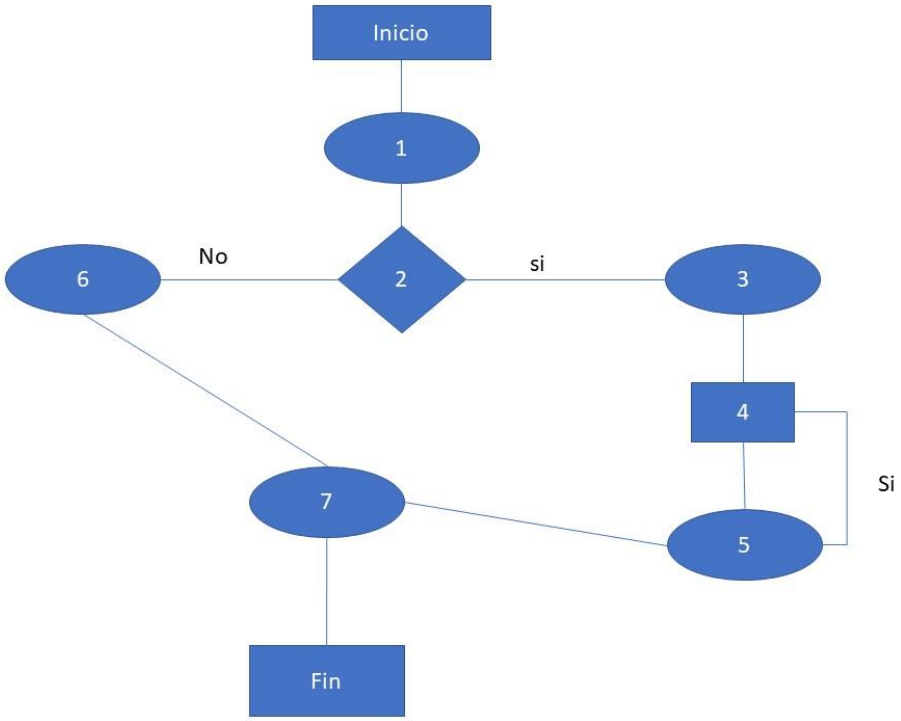


Figura 9 Grafo de Flujo (Fuente: Elaboración Propia)

A partir del grafo obtenido con 7 nodos y 8 aristas se calcula la complejidad ciclomática V(G), la cual constituye una métrica de software que proporciona una medida cuantitativa de la complejidad lógica del programa (Roger. S. Pressman, 2010).

V(G)= (cantidad_aristas – cantidad_nodos) + 2

V(G)= (8 – 7) + 2 = 3

Una ruta independiente es cualquier ruta del programa que ingrese al menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición (Roger. S. Pressman, 2010). La cantidad de rutas independientes son establecidas por la complejidad ciclomática, por tanto, se identifican 3 caminos, tal y como se muestra a continuación.

Caminos:

- 1-2-3-4-5-7
- 1-2-6-7

Resultado de la prueba de unidad

Se realizaron tres iteraciones de la prueba unitaria. En la primera iteración se detectaron 2 no conformidades; en la segunda, 1 no conformidad la cual fue resuelta para la tercera iteración. Las no conformidades detectadas estaban asociadas a errores de validación del ip.

3.5.1.2 Prueba de integración

La prueba de integración es una técnica para construir la arquitectura del software. El objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño (Roger. S. Pressman, 2010). La herramienta debe incorporarse al sistema operativo base de la distribución de GNU/Linux Nova 8.0.

La herramienta de transferencias de archivos con Nautilus de la distribución cubana de GNU/Linux Nova 8.0 se expuso a pruebas de integración para verificar la compatibilidad y el funcionamiento de las interfaces, lo que arrojó como resultado que cada uno de los componentes compilados de la aplicación, se integran correctamente.

Las pruebas de integración arrojaron como resultado que existe una correcta integración entre el Nautilus y la aplicación.

3.5.1.3 Prueba de validación

Las pruebas de validación consisten en realizar acciones visibles para el usuario y en la salida que el sistema tiene ante cada una de estas acciones. La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos funcionales. Las pruebas de caja negra se realizan sobre la interfaz del software, sin tener en cuenta el funcionamiento interno del sistema, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas (Roger. S. Pressman, 2010).

A continuación, se presentan los casos de prueba correspondientes a las historias de usuario Listar dispositivo, Enviar por ip, Enviar por local.

Tabla 7Prueba de Aceptación Listar Dispositivo (Fuente: Elaboración Propia)

Caso de Prueba de Aceptación	
Código Caso de Prueba: Transferencia de archivo Nova 8.0	Nombre Historia de Usuario: Listar dispositivo
Nombre de la persona que realiza la prueba: Cristhiam Duvier López Núñez	
Descripción de la prueba: El sistema permitirá listar los dispositivos dándole clic derecho en el archivo o carpeta a enviar y se le debe mostrar una interfaz con los ip de los dispositivos conectados a la subred.	
Condiciones de Ejecución: El usuario debe de tener instalada la herramienta con la integración con Nautilus para la distribución cubana GNU/Linux Nova 8.0	
Entrada/Pasos de Ejecución: 1.Acceder al archivo o carpeta 2.Clic derecho encima del archivo o carpeta deseada. 3.Elegir opción “Enviar a submenu1”. 4.Seleccionar el ip del dispositivo a enviar.	
Resultado esperado: El sistema muestra los ip conectados a la subred.	
Evaluación de la Prueba: Satisfactorio	

Tabla 8 Prueba de Aceptación Enviar por Ip (Fuente: Elaboración Propia)

Caso de Prueba de Aceptación	
Código Caso de Prueba: Transferencia de archivo Nova 8.0	Nombre Historia de Usuario: Enviar por ip
Nombre de la persona que realiza la prueba: Cristhiam Duvier López Núñez	
Descripción de la prueba: El sistema permitirá enviar por ip dando clic derecho al archivo o carpeta y seleccionar la opción de enviar a submenu1, luego se debe mostrar la interfaz con la opción de introducir el ip y enviar.	
Condiciones de Ejecución: El usuario debe de tener instalada la herramienta con la integración con Nautilus para la distribución cubana GNU/Linux Nova 8.0	
Entrada/Pasos de Ejecución: 1.Acceder al archivo o carpeta 2.Clic derecho encima del archivo o carpeta deseada. 3.Elegir opción “Enviar a submenu1”. 4.Introducir el ip deseado. 5. clic en enviar.	
Resultado esperado: El sistema deja introducir el ip y envía el archivo o la carpeta.	
Evaluación de la Prueba: Satisfactorio	

Tabla 9 Prueba de Aceptación Enviar por Local (Fuente: Elaboración Propia)

Caso de Prueba de Aceptación	
Código Caso de Prueba: Transferencia de archivo Nova 8.0	Nombre Historia de Usuario: Enviar por local

Nombre de la persona que realiza la prueba: Cristhiam Duvier López Núñez
Descripción de la prueba: El sistema permitirá enviar por local dando clic derecho al archivo o carpeta y seleccionar la opción de enviar a submenu1, luego se debe mostrar la interfaz con la opción de seleccionar el ip del dispositivo deseado y enviar.
Condiciones de Ejecución: El usuario debe de tener instalada la herramienta con la integración con Nautilus para la distribución cubana GNU/Linux Nova 8.0
Entrada/Pasos de Ejecución: 1.Acceder al archivo o carpeta 2.Clic derecho encima del archivo o carpeta deseada. 3.Elegir opción “Enviar a submenu1”. 4.Seleccionar el ip del dispositivo y enviar.
Resultado esperado: El sistema deja seleccionar los ip conectados a la subred y enviar.
Evaluación de la Prueba: Satisfactorio

Resultado de la prueba de validación

Las pruebas de validación se ejecutaron tras concluir cada una de las tres iteraciones de implementación, de manera general en la primera iteración se detectaron 5 no conformidades, agrupadas en no conformidades de código, interfaz y ortografía, de ellas fueron solucionadas 3. En la segunda no se detectaron no conformidades solo las 2 no resueltas en la iteración anterior, aquí se resolvieron las 2 no conformidades. Para una tercera iteración, no se encontró ninguna no conformidad. Como se muestra en la figura.

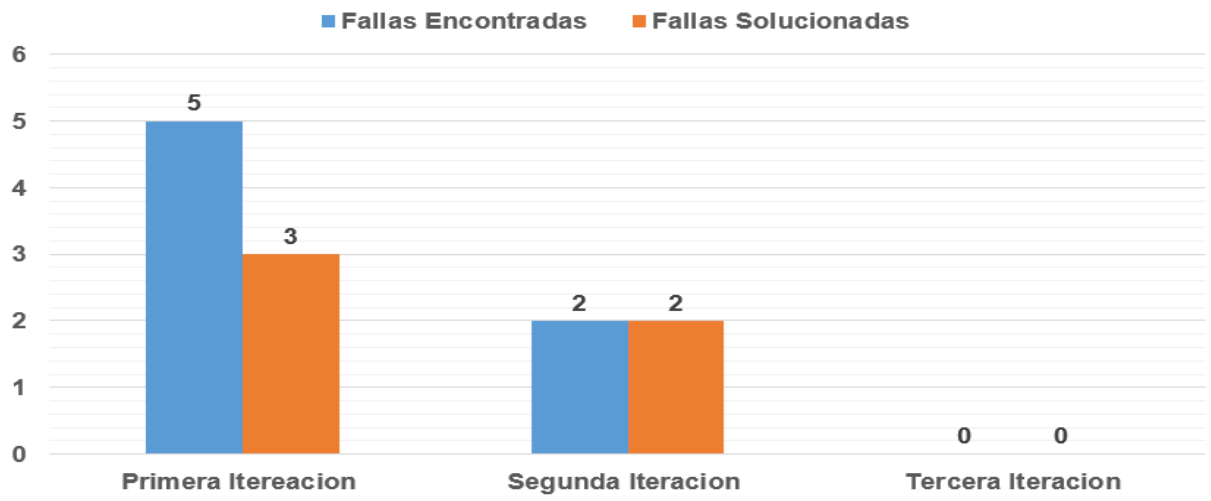


Figura 10 Resultado de la Prueba de Validación (Fuente: Elaboración Propia)

3.5.1.4 Prueba de aceptación por historia de usuario

La técnica de Caja Negra puede utilizarse para lograr objetivos de cobertura de entrada y salida, con entradas humanas, vía interfaces a un sistema, o parámetros de interfaz de las pruebas de integración". En esta técnica es importante identificar las clases de equivalencia, por ejemplo, rango de valores entre 1 y 10 serán las clases de equivalencia, es decir que todo valor menor a 1 y todo valor mayor a 10 serán valores inválidos. Luego se generan los casos de prueba con diferentes valores para asegurar que la aplicación solo acepte valores entre 1 y 10 (Paz, 2016).

Se emplearon las pruebas de caja negra con el objetivo de encontrar errores en las siguientes categorías:

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en las estructuras de datos.
- Errores de comportamiento o rendimiento.

Para desarrollar el método de caja negra se utilizan las técnicas (Pressman, 2010):

- Partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

- Análisis de valor de frontera: prueba la habilidad del programa para manejar datos que se encuentran en los límites o fronteras aceptables.

Para poner en práctica este método de pruebas, según las técnicas que se describieron anteriormente, se toma en cuenta los Diseños de Casos de Pruebas (DCP). Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Por su parte, el análisis de valor de frontera es una técnica de diseño de casos de prueba que complementan la partición de equivalencia. En lugar de enfocarse exclusivamente en las condiciones de entrada, también deriva casos de prueba a partir del dominio de salida (Pressman, 2010).

Luego de realizar el diseño de casos de prueba se completó la tabla de control de documento y control de cambios, se tomó para esta prueba la Historia de Usuario Enviar por Ip.

Descripción General: El Sistema debe enviar archivo por Ip

Condiciones de ejecución: El sistema debe estar conectado a una red local o externa

Tabla 10 Prueba, Historia de Usuario (Fuente: Elaboración propia)

Escenario	Descripción	Ip	Respuesta del sistema	Flujo central
El ip es correcto	El sistema valida un ip correctamente	10.8.111.03	Se notifica que el Ip es válido y se envía la información	Se introduce el ip o se selecciona del listado de ip disponibles deseado, se da clic en enviar
El ip es incorrecto	El sistema no valida un ip incorrectamente	111.2. 3333.a	Se notifica que el ip es incorrecto	
	El sistema valida que el ip es vacío		Se notifica que está vacío	

3.5.1.5 Pruebas de regresión

Las pruebas de regresión son pruebas de un programa previamente probado que ha sufrido modificaciones, para asegurarse que no se han introducido o descubierto defectos en áreas del software que no han sido modificadas como resultado de los cambios realizados. Se realiza cuando el software o su entorno han sido modificados.

En el caso de la aplicación se realizaron las pruebas de regresión descritas al requisito funcional enviar por ip y se arrojaron distintos resultados en tres distintas iteraciones:

Tabla 11 Prueba de Regresión (Fuente: Elaboración Propia)

Iteración	Error a Resolver	Problemas Encontrados
Iteración 1	No era posible enviar por Ip	Al enviar por el Ip no se conectaba al Ip ejemplo de esto es debido a falta de conexión, fuera de servicio o que este apagado.
Iteración 2	Se corrigió el error de la Iteración 1	El archivo no se envía porque no tiene conexión.
Iteración 3	Se corrigió el error de la Iteración 2	No se encontraron problemas.

3.5.1.6 Evaluación del objetivo general de la investigación

Cuando se realiza una propuesta, es recomendable retroalimentarse con la opinión de los usuarios potenciales. Esta información es útil para conocer las debilidades de la propuesta y profundizar en sus fortalezas. En ese sentido, la técnica de Iadov es un instrumento que ayuda a conocer el grado de satisfacción de los potenciales usuarios (Piñeiro Cárdenas, 2017).

Iadov es una técnica efectiva para el estudio del nivel de satisfacción de los participantes a través de la consulta a un panel de experto. Este método calcula el Índice de Satisfacción Grupal (ISG) se implementa mediante un cuestionario (Ver anexo 2) en el cual se le incluyen tres preguntas cerradas que se intercalan dentro de un cuestionario de cinco preguntas y cuya relación el encuestado desconoce (Piñeiro Cárdenas, 2017). La encuesta elaborada para evaluar el índice de satisfacción de los usuarios potenciales de la propuesta de solución fue aplicada a 7 especialistas del proyecto Nova.

Estas tres preguntas se relacionan a través del "Cuadro Lógico de Iadov" el cual permite ubicar a cada encuestado, según el cuadro lógico en una escala de satisfacción, para luego calcular el ISG. La escala de satisfacción la cual toma valores de 1 a 6 es la siguiente 1-Clara satisfacción, 2-Más satisfecho que insatisfecho, 3-No definida, 4-Más insatisfecho que satisfecho, 5-Clara insatisfacción y 6-Contradictoria (Piñeiro Cárdenas, 2017).

Tabla 12 Cuadro lógico de IADOV (Fuente: Elaboración Propia).

4. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la solución desarrollada.	2. ¿La forma en que se realiza la transferencia de archivo con el gestor del fichero nautilus en la universidad permite satisfacer todas las necesidades y exigencias del usuario?								
	No			No sé			Si		
	3. ¿Considera usted factible la implementación de la herramienta de transferencia de fichero con el gestor de fichero nautilus para distribución cubana de GNU/Linux Nova 8.0?								
	Si	No sé	No	Si	No sé	No	Si	No sé	No
Me gusta mucho	1	2	6	2	6	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	3	3	6	3	3
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	3	4
No me gusta nada	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

El número resultante de la interrelación de las tres preguntas indica la posición en la escala de satisfacción siguiente: clara satisfacción (A), más satisfecho que insatisfecho (B), no definida (C), más insatisfecho que satisfecho(D), clara insatisfacción (E) y contradictoria (C).

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y – 1. El número resultante de la interrelación de las tres preguntas que indica la posición de cada encuestado en la siguiente escala de satisfacción:

1. Clara satisfacción +1
2. Más satisfecho que insatisfecho 0.5
3. No definido y contradictorio 0
4. Más insatisfecho que satisfecho -0.5
5. Clara insatisfacción -1

El índice de satisfacción grupal (ISG) se expresa en una escala numérica que va desde 1 (máxima satisfacción), hasta -1 (máxima insatisfacción). El ISG se calcula mediante la siguiente fórmula:

$$ISG = \frac{A(+1) + B(+0.5) + C(0) + D(-0.5) + E(-1)}{N}$$

En esta fórmula A, B, C, D, E, representan la cantidad de encuestados colocados respectivamente en las posiciones de satisfacción 1; 2; 3 u 6; 4; 5 y donde N representa la cantidad total de encuestados.

Resultados obtenidos

Los resultados obtenidos de la herramienta de la encuesta se presentan en la siguiente tabla:

Tabla 13 Resultados obtenidos de los encuestados (Fuente: Elaboración Propia)

Categorías grupales de satis-	N = 7	Escala

facción		
Clara satisfacción	4	A
Más satisfecho que insatisfecho	2	B
No definido	0	C
Más insatisfecho que satisfecho	0	D
Clara insatisfacción	0	E
Contradictorio	1	C

2. Cálculo del ISG

ISG=A (+1) +B (+0.5) + C (0) /N

ISG= (4(+1) +2(+0.5)) +1(0) /7=0.71

3. Interpretación del resultado del ISG

El valor obtenido del ISG fue 0.71 lo que indica máxima satisfacción de los usuarios con respecto a la herramienta de transferencia de archivo con el gestor de fichero nautilus de la distribución cubana de GNU/Linux Nova 8.0. Se puede afirmar que se cumplió el objetivo de la investigación. Las respuestas a las preguntas abiertas brindadas por los encuestados reafirman los beneficios que traerá la utilización de la herramienta propuesta.

Conclusiones parciales del Capítulo

En este capítulo se han abordado los elementos de la implementación de la herramienta de transferencia de archivos desde el navegador de archivo Nautilus en tiempo real de los clientes ligeros desde GNU/Linux Nova, así como las pruebas realizadas al mismo y los resultados obtenidos; arribando a las siguientes conclusiones: la elaboración de diagrama de componente, permitió una mejor comprensión de la estructura de los componentes del sistema implementado. La aplicación de las pruebas de unidad, integración, validación y rendimiento a la solución desarrollada permitió encontrar errores que afectaban el funcionamiento del sistema, lo que posibilitó corregirlos a tiempo para que el mismo cumpliese totalmente con los requisitos funcionales definidos en la etapa de análisis.

CONCLUSIONES FINALES

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

- Las relaciones existentes entre los principales conceptos asociados al dominio de la presente investigación, permitieron una mayor comprensión de la propuesta de solución.
- El análisis de las diferentes herramientas para la transferencia de archivos permitió determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución.
- La elaboración de los artefactos propuestos por la metodología de desarrollo y la especificación de requisitos permitieron un mejor entendimiento de la herramienta que se propone desarrollar, así como las características del mismo.
- Como resultado de la implementación se obtuvo la herramienta para la transferencia de archivo desde el navegador de archivo Nautilus en la distribución cubana GNU/Linux Nova 8.0, que cumple con los 4 requerimientos funcionales identificados en la fase de ejecución.
- Las pruebas diseñadas y ejecutadas arrojaron como resultado que el sistema implementado responde a los requerimientos definidos por el cliente.

RECOMENDACIONES

Una vez concluida la investigación y el desarrollo de la propuesta de solución, el autor del presente trabajo recomienda:

- Realizar un estudio sobre que es mas factible si vincular NDrop a nautilus que ya posee una interfaz gráfica o mejorar la interfaz, hacer las notificaciones y la barra de progreso.

REFERENCIAS BIBLIOGRÁFICAS

5 softwares de control de versiones. (s. f.). Recuperado 5 de junio de 2022, de https://www.drauta.com/5-softwares-de-control-de-versiones

A, D. (2018, enero 11). EasyJoin, envía archivos entre tu teléfono y tu PC sin Internet. *Ubunlog*. https://ubunlog.com/easyjoin-envios-entre-telefono-pc/

Analisis y diseño. (2019). Applicatta. https://www.applicatta.cl/index.php/soluciones/metodologia-applicatta/analisis-y-diseno

apoklipsix. (s. f.). *Dukto, fácil transferencia de archivos. – Inforksol*. Recuperado 28 de mayo de 2022, de https://inforksol.cubava.cu/dukto-facil-transferencia-de-archivos/

Archivo en Informática—Concepto, características y formato. (s. f.). *Concepto*. Recuperado 9 de junio de 2022, de https://concepto.de/archivo-informatico/

Arquitectura en Capas. (s. f.). Recuperado 15 de septiembre de 2022, de https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas

Bernal, L. (2022). *Protocolos SSH, SFTP, SCP*. prezi.com. https://prezi.com/m_2rqhwglbm4/protocolos-ssh-sftp-scp/

de la O Barrientos, M. (2019). *Módulo para la implementación de mecanismos de bloqueo a conexiones remotas que intentan accesos por fuerza bruta para NovaLTSP*. UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS.

El Ultimo Tutorial de Diagramas de Clase Para Ayudar a Modelar sus Sistemas Facilmente. (2020, octubre 25). *Blog de Creately*. https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-clases/

Evolución de los sistemas operativos. (s. f.). Recuperado 9 de junio de 2022, de http://iic2333.ing.puc.cl/activities/history.html

File Transfer Protocol (FTP) Definition. (s. f.). Investopedia. Recuperado 20 de mayo de 2022, de https://www.investopedia.com/terms/f/ftp-file-transfer-protocol.asp

Gestor de archivos: Nautilus. (s. f.). Recuperado 7 de julio de 2022, de http://cefire.edu.gva.es/file.php/1/LLiurex_pera_la_tasca_docent/Unidad_2/gestor_de_archivos_nautilus.html

Guerra, C. A. (2017). *Obtención de Requerimientos. Técnicas y Estrategia*. SG Buzz. https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia

Historia de los sistemas operativos. (s. f.). Recuperado 22 de junio de 2022, de https://www.fib.upc.edu/retro-informatica/historia/so.html

Las mejores herramientas de transferencia de archivos en Linux—HelpFulHelp. (s. f.). Recuperado 6 de junio de 2022, de https://helpfulhelp.net/es/las-mejores-herramientas-de-transferencia-de-archivos-en-linux

León García, A. (2017). *Control de acceso a recursos compartidos en el Nautilus*.

Linux vs. Windows: Soluciones de alojamiento web. (s. f.). IONOS Digitalguide. Recuperado 8 de junio de 2022, de https://www.ionos.es/digitalguide/servidores/know-how/linux-vs-windows-el-gran-cuadro-comparativo/

LIU, Y. (2022). *NDrop* [Python]. https://github.com/liuyug/ndrop (Original work published 2019)

Metodologías de desarrollo de software | Universitat Carlemany. (s. f.). Recuperado 1 de junio de 2022, de https://www.universitatcarlemany.com/actualidad/metodologias-de-desarrollo-de-software

Miriam. (2020, junio 24). Qué son los Patrones de Diseño de software / Design Patterns. *Profile Software Services*. https://profile.es/blog/patrones-de-diseno-de-software/

miteris. (2021, diciembre 16). ¿Qué es Python? Características y librerías. *Miteris*. https://www.miteris.com/blog/que-es-python-caracteristicas-y-librerias/

Patrones GRASP. Patrones GoF. Diferencia entre GRASP y GoF - TuxNots. (s. f.). Recuperado 16 de septiembre de 2022, de http://tuxnots.com.ar/materias-de-la-facu/metodologia-de-sistemas/patronesgrasppatronesgofdiferenciaentregraspygof

Paz, J. (2016). Análisis del proceso de pruebas de calidad de software. *Ingeniería solidaria*, *12*, 163. https://doi.org/10.16925/in.v12i20.1482

Piñeiro Cárdenas, J. (2017). *Herramienta web para la creación de personalizaciones de Nova Servidores* [Trabajo de diploma por titulo de ingeniero]. Universidad de las Ciencias Informaticas.

pmoinformatica.com, P. por. (2015, mayo 6). *Requerimientos no funcionales: Ejemplos*. http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html

pmoinformatica.com, P. por. (2017, febrero 6). Requerimientos funcionales: Ejemplos. *requisitos funcionales*. http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html

Pressman, Roger S. (2002). *Ingeniería de Software, un enfoque práctico* (Quinta edición).

Pressman, R. S. (2010). Ingeniería de software enfoque practico.7ed.Pressman.PDF. *Ingeniería del software, un enfoque práctico*. https://www.academia.edu/44770344/Ingenieria_de_software_enfoque_practico_7ed_Pressman_PDF

Qué es GIT y para qué sirve | OpenWebinars. (s. f.). Recuperado 5 de junio de 2022, de https://openwebinars.net/blog/que-es-git-y-para-que-sirve/

¿Qué es Modelado de datos? - Definición en WhatIs.com. (s. f.). ComputerWeekly.es. Recuperado 14 de septiembre de 2022, de https://www.computerweekly.com/es/definicion/Modelado-de-datos

¿Qué es un mapa conceptual? | *Lucidchart*. (s. f.). Recuperado 21 de octubre de 2022, de https://www.lucidchart.com/pages/es/que-es-un-mapa-conceptual

Qué es Visual Studio Code y qué ventajas ofrece | *OpenWebinars*. (s. f.). Recuperado 10 de junio de 2022, de https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/

Rsync: 10 ejemplos prácticos de comandos Rsync. (s. f.). *ProxAdmin Blog*. Recuperado 6 de junio de 2022, de https://www.proxadmin.es/blog/rsync-10-ejemplos-practicos-de-comandos-rsync/

Samba. (2022). https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-samba.html

Software—Requisitos. (s. f.). Recuperado 10 de septiembre de 2022, de https://www.tutorialspoint.com/es/software_engineering/software_requirements.htm

Syncthing. (s. f.). Recuperado 6 de junio de 2022, de https://syncthing.net/

Visual Paradigm. (s. f.). Capterra. Recuperado 2 de junio de 2022, de https://www.capterra.es/software/145716/visual-paradigm

ANEXOS

Anexo 1 Interfaces Graficas

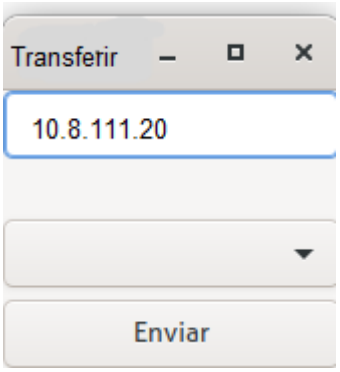


Figura 11 Interfaz Gráfica 1(Fuente: Elaboración Propia)

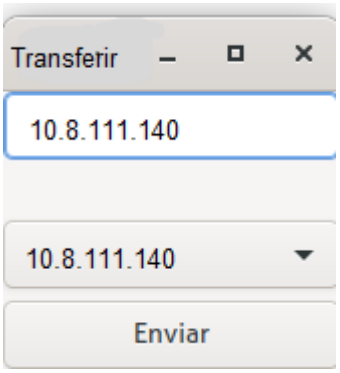


Figura 12 Interfaz Gráfica 2 (Fuente: Elaboración Propia)

Anexo 2 Encuesta realizada a especialistas del Centro de Software Libre (CESOL)

Objetivo: Evaluar la propuesta de solución desarrollada.

Especialista, le invito a responder el siguiente cuestionario con el fin de conseguir su colaboración en la presente investigación, solicito que exprese en sus respuestas criterios verídicos que guíen al autor de la investigación:

1. ¿Considera importante la implementación de una herramienta para la transferencia de archivo para la distribución cubana de GNU/Linux Nova 8.0?

___Sí ___No ___ No sé

2. ¿La forma en que se realiza la transferencia de archivo con el gestor del fichero nautilus en la universidad permite satisfacer todas las necesidades y exigencias de usuario?

___Sí ___No ___ No sé

3. ¿Considera usted factible la implementación de la herramienta de transferencia de fichero con el gestor de fichero nautilus para distribución cubana de GNU/Linux Nova 8.0?

___Sí ___No ___ No sé

4. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la solución desarrollada.

___me gusta mucho ___me disgusta más de lo que me gusta

___me gusta más de lo que me disgusta ___no me gusta nada ___me da lo mismo

___no sé decir

5. ¿Qué opina usted acerca de los beneficios que traería para la universidad disponer de la personalización propuesta para la distribución cubana de GNU/Linux Nova 9.0?