Universidad de las Ciencias Informáticas
Facultad 1

Herramienta para la transferencia de archivos en Nova

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

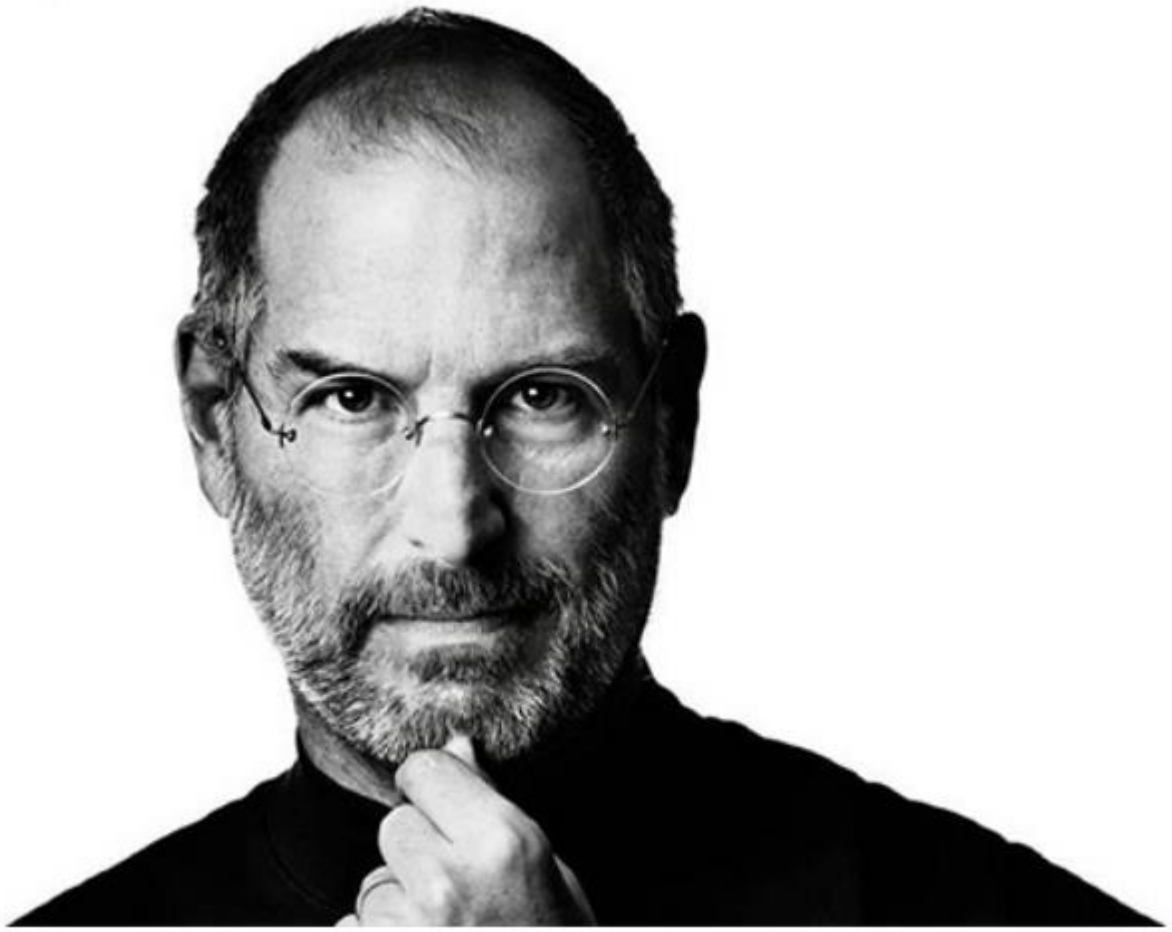
Autor: Leonardo Mastrapa Reynaldo

Tutores: MSc. Ruth Yurina Vega Cutiño

Ing. Javier Piñeiro Cárdena

Ing. Miosotis Toledo Rodríguez

La Habana, Diciembre del 2021



"Tened el valor de seguir vuestro corazón e intuición, porque de alguna manera ya sabéis lo que realmente queréis llegar a ser. Todo lo demás es secundario".

-Steve Jobs

Declaración de autoría

Declaro por este medio que yo Leonardo Mastrapa Reynaldo, con carné de identidad 97091918144 soy el autor principal del trabajo titulado Herramienta para la transferencia de archivos en Nova y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firman la presente a los ____ días del mes de ____ del año 2021.

_____

Leonardo Mastrapa Reynaldo

Autor

_____ _____

MSc. Ruth Yurina Vega Cutiño

Tutor

Ing. Javier Piñeiro Cárdenas

Tutor

_____

Ing. Miosotis Toledo Rodríguez

Tutor

Datos de contacto

MSc. Ruth Yurina Vega Cutiño
Universidad de las Ciencias Informáticas, La Habana, Cuba

Ing. Javier Piñeiro Cárdenas
Universidad de las Ciencias Informáticas, La Habana, Cuba

Ing. Miosotis Toledo Rodríguez
Universidad de las Ciencias Informáticas, La Habana, Cuba

Resumen

Cuba se encuentra en un proceso de migración a software libre donde la Distribución Cubana GNU/Linux Nova, es el sistema operativo seleccionado para ser desplegado en los Organismos de Administración Central del Estado. El objetivo del presente trabajo de diploma es desarrollar una herramienta que permita mejorar y agilizar el proceso de transferencia de archivos en Nova. Para guiar el proceso de construcción de la propuesta de solución se utilizó la metodología de desarrollo de software Variación de AUP para la UCI y en la implementación se emplearon tecnologías de software libre. La evaluación de la propuesta de solución se realizó a partir de la aplicación de técnicas y pruebas que garantizan el correcto funcionamiento de la aplicación y demostraron la satisfacción del cliente hacia el sistema desarrollado. Como resultado de esta investigación se obtuvo una herramienta nativa que permite la transferencia de archivos entre ordenadores conectado en una red LAN (*Local Area Network, Red de Área Local*), de la Distribución Cubana GNU/Linux Nova.

Palabras clave: GNU/Linux, software libre, transferencia de archivos.

Índice

Índice de figuras

Índice de tablas

Introducción

Un sistema operativo (SO) es un conjunto de programas informáticos que permiten la administración eficaz de los recursos de la computadora. Es el software básico de un ordenador que provee una interfaz entre el resto de los programas de la computadora, los dispositivos hardware y el usuario. Son programas que hacen que el ordenador funcione de forma general, es decir que respondan a instrucciones, coordinan y dirigen todos los servicios y aplicaciones que utiliza el usuario en una computadora. Estos tienen como características principales que administran de manera eficiente los recursos del ordenador, juega el papel de intermediario entre el usuario y el hardware, otorga seguridad, protege a los programas y archivos y permiten interactuar con varios dispositivos (Guzmán & Máter, x2018).

La creación, implementación y avance de los sistemas operativos han sido imprescindible para el acercamiento al usuario a productos de gran consumo, como son los ordenadores y los teléfonos móviles. El SO posibilita que cualquier individuo sin conocimiento técnico pueda interactuar con dispositivos electrónicos, siendo relativamente sencillo adaptarse y controlar la interfaz de usuario. En el área de los ordenadores personales, los primeros sistemas operativos desarrollados fueron MS-DOS, Linux, IBM OS/2, MS Windows y el primer Mac OS X. Actualmente uno de los sistemas operativos más usados en el mundo es Linux que se ha vuelto imprescindible en la computación (Guzmán & Máter, x2018).

GNU/Linux es un sistema operativo tipo Unix, desarrollado para explotar al máximo las capacidades de las computadoras basadas en el microprocesador i386 y posteriores. Se distingue por sus características principales como, la estabilidad de operación, su velocidad y la seguridad como pilar fundamental. Se destaca también por su capacidad para la administración eficiente de los recursos. Estas características han hecho que sea diferente a otros sistemas operativos del mundo, pero la más relevante de todas es que su código es abierto, por lo que puede ser modificado y distribuido libremente. Esto ha dado lugar que las distribuciones GNU/Linux hayan ampliado su uso en diferentes esferas de la sociedad (Linux-es, 2015).

Hoy en día, existen muchas distribuciones basadas en GNU/Linux. Están para todo tipo de ordenadores, dispositivos electrónicos y puntos de acceso de redes inalámbricas entre otros. Por su destacada característica de ser un software libre, cualquiera puede tomar el código desarrollado hasta el momento y

adaptarlo a sus propias necesidades, creando sus propias distribuciones evidenciado con mayor fuerza en el sector empresarial.

En la actualidad muchos países están optando por esta alternativa, Cuba es uno de ellos, el cual se encuentra en el proceso de migración a software libre en las instituciones con el despliegue de la distribución cubana de GNU/Linux Nova. Esta distribución es desarrollada por especialistas de la Universidad de las Ciencias Informáticas pertenecientes al centro dedicado al desarrollo de soluciones libres (CESOL).

La distribución cubana GNU/Linux Nova es un sistema operativo que utiliza el núcleo Linux e incluye determinados paquetes de aplicaciones informáticas para satisfacer las necesidades de la migración a plataformas de código abierto que experimenta Cuba (Pierra, 2010). Nova como la mayoría de los sistemas de escritorio cuenta con aplicaciones por defecto, pero esta distribución no cuenta con una herramienta que permita el envío de ficheros a través de la red LAN a la que está conectada la computadora de forma sencilla y rápida, que cualquier tipo de usuario pueda usarla sin preocuparse por permisos, sistemas operativos, protocolos, clientes, servidores y demás. Actualmente en Nova para realizar la transferencia de archivos entre ordenadores, se realiza de diferentes formas, una de estas es mediante la utilización de SAMBA un conjunto de aplicaciones Linux, basadas en el protocolo SMB (*Server Message Block*, *Bloque de Mensaje del Servidor*), que permiten compartir archivos en la red. Este complejiza mucho la simple finalidad de enviar un archivo pues requiere la instalación y configuración del servidor antes de su uso, resultando ser un proceso largo y engorroso y poco factibles para aquellos usuarios con pocos conocimientos sobre el tema.

También se utiliza SSH/SCP (*Secure Copy Protocol*, Protocolo de Copia de Seguridad) 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 puede llegar a ser tedioso en redes que se utilice 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, además no posee una interfaz y todo el proceso se realiza desde la terminal, complejizando su uso.

Uno de los programas más usados por los usuarios de Nova es Dukto, que una vez instalado en ambas computadoras solo necesita ser ejecutarlo para compartir el fichero. Posee una interfaz amigable e intuitiva donde el usuario siempre sabe qué hacer, permitiendo la transferencia desde pequeños hasta grandes volúmenes de contenidos digitales. Su limitación radica en que es un software desarrollado por terceros, no está disponible en los repositorios oficiales, el equipo de desarrollo de Nova no es responsable de su implementación y mantenimiento y no posee su código fuente para su revisión lo que puede representar un problema de seguridad.

Dada la situación problemática anteriormente planteada se define como ***problema de investigación*:** ¿Cómo mejorar la transferencia de archivos en la Distribución Cubana GNU/Linux Nova?

Para guiar la investigación se propone como ***objeto de estudio**:* proceso de transferencia de archivos en los sistemas operativos enmarcado en el ***campo de acción**:* herramientas de transferencias de archivos en los sistemas operativos.

En consecuencia se define como ***objetivo general**:* Desarrollar una herramienta nativa que mejore la transferencia de archivos en la Distribución Cubana GNU/Linux Nova.

A partir del objetivo general se definen los siguientes ***objetivos específicos:***

1. Elaborar el marco teórico de la investigación sobre el proceso de transferencia de archivos en los sistemas operativos.
2. Diseñar una herramienta nativa que mejore la de transferencia de archivos en la Distribución Cubana GNU/Linux Nova.
3. Implementar una herramienta nativa que mejore la transferencia de archivos en la Distribución Cubana GNU/Linux Nova.
4. Evaluar la herramienta nativa que mejore la transferencia de archivos en la Distribución Cubana GNU/Linux Nova.

Para dar cumplimiento a los objetivos específicos planteados se definen las siguientes ***preguntas científicas***:

¿Cuáles son los fundamentos teóricos que sustentan la investigación sobre el proceso de transferencia de archivos en los sistemas operativos?

¿Cuáles son los aspectos a tener en cuenta para realizar el diseño de una herramienta nativa que mejore la de transferencia de archivos en la Distribución Cubana GNU/Linux Nova?

¿Qué componentes son necesarios implementar en el desarrollo de una herramienta nativa que mejore la de transferencia de archivos en la Distribución Cubana GNU/Linux Nova?

¿Qué pruebas de software aplicar para la evaluación de una herramienta nativa que mejore la de transferencia de archivos en la Distribución Cubana GNU/Linux Nova?

En el desarrollo de la investigación se utilizaron métodos de investigación científica. Al emplearlos se pudo comprender de forma lógica y sistemática la esencia del problema que se trata y así obtener resultados satisfactorios al culminar la investigación.

Métodos teóricos

El análisis de los diferentes métodos teóricos permite una mayor comprensión del fenómeno estudiado y proporciona una base teórica para el desarrollo del presente trabajo. El empleo y selección de estos métodos ocasiona un mejor entendimiento de los hechos y procesos estudiados. Uno de los métodos que se utilizó fue el **analítico-sintético** que permitió el estudio de gran parte de la bibliografía existente sobre las aplicaciones de transferencia de archivos. Facilitó la comprensión de las funcionalidades básicas de cada aplicación estudiada. Además ayudó a seleccionar la metodología de desarrollo de software, los lenguajes de programación, las herramientas y tecnologías utilizadas para el desarrollo de la aplicación propuesta.

Otro de los métodos es el de **modelación** el cual se utilizó para el diseño de diagramas y modelos que conforman la aplicación, centrar la idea concreta de programación, así como planificar el cronograma de

ejecución para las fases de desarrollo. Además, para elaborar los diferentes artefactos generados por la metodología que guía y documenta esta investigación.

El método **inductivo-deductivo** se utilizó para realizar la búsqueda de las características y las similitudes que caracterizan al objeto de estudio de la investigación.

Métodos empíricos

Utilizar estos métodos en la investigación permite apoyarse en la experiencia de otros autores y realizar una exploración sobre el tema en cuestión. Se efectúa un análisis preliminar de la información para luego verificar y comprobar las concepciones teóricas. Uno de los métodos empíricos que se utilizó fue la **observación científica** el cual se aplicó durante la investigación para lograr una mayor compresión de las actividades que se efectúan en el desarrollo del proceso (Ver Anexo II).

Otro método utilizado fue la **entrevista**. Se entrevistaron 3 especialistas del centro CESOL y a usuarios de NOVA, en la búsqueda de una solución al problema planteado. Con el fin de comprender el proceso de transferencia de archivos entre varios ordenadores. El tipo de entrevista empleada fue la individual no estructurada, pues no lleva un cuestionario rígido y puede variar de una persona a otra, lo que permite al entrevistado tener mayor libertad.

La presente tesis está estructurada en: resumen, introducción, tres capítulos, conclusiones, referencias bibliográficas y anexos. A continuación, se presenta una breve descripción de los capítulos:

Capítulo 1. Fundamentación teórica sobre el proceso de transferencia de archivos entre ordenadores: en este capítulo queda reflejada la fundamentación teórica de la investigación sobre el proceso de transferencia de archivos a través del estudio de sistemas homólogos; además se definen la metodología de desarrollo de software, y los lenguajes y las herramientas para la modelación e implementación de la aplicación para las transferencias de archivos en el sistema operativo GNU/Linux Nova.

El capítulo 2. Análisis y diseño de una herramienta nativa para la transferencia de archivos en la Distribución Cubana GNU/Linux Nova.: en este capítulo se presenta la propuesta de solución al problema de la investigación. Se definen los requisitos funcionales y no funcionales de la misma. Además,

se describen las funcionalidades en las Historias de Usuario. Se explican los patrones arquitectónicos y de diseño que se emplean, y se define la arquitectura del software.

El capítulo 3. Implementación y evaluación de la herramienta nativa para transferencia de archivos en la Distribución Cubana GNU/Linux Nova: Este capítulo se enfoca en la construcción del sistema a partir de los resultados del capítulo anterior. En él se elaboran los diagramas pertinentes, se definen los estándares de codificación a utilizar en la implementación de la solución y el código fuente de la misma. Además, se definen y realizan las pruebas de software requeridas, con el objetivo de descubrir y corregir posibles errores, midiendo así la calidad de la propuesta de solución y constando que el resultado de la investigación satisface las necesidades del cliente.

Capítulo 1: Fundamentación teórica del proceso de transferencia de archivos en Nova

En el presente capítulo se describen los fundamentos teóricos que sustentan la investigación, en el mismo se definen también los términos utilizados a lo largo de la tesis. Se realiza además un estudio de las aplicaciones de transferencia de archivos existentes. Se define la metodología de desarrollo de software utilizada para la elaboración de la propuesta de solución, también se aborda sobre los lenguajes y herramientas empleados en el diseño e implementación de la aplicación para la transferencia de archivos

1.1 Conceptos generales

A continuación, se muestran un conjunto de conceptos que permiten la comprensión del tema de investigación.

1.1.1 Archivo

En el campo de la informática, se llama “archivo” al elemento de información compuesto por una suma de registros (combinaciones de bytes). Llevan este nombre por ser los equivalentes digitalizados de los archivos físicos. Tanto es así que muchos de los archivos “en papel” se están actualmente digitalizando, para reducir su tamaño físico y facilitar su organización y búsqueda (Raffino, 2020).

1.1.2 Transferencia de archivo

Término genérico para referirse al acto de transmisión de ficheros a través de una red de computadoras. Si bien el término suele estar ligado al Protocolo de Transferencia de Archivos (File Transfer Protocol, FTP), hay muchas formas de transferir archivos a través de una red (Alcantud, 1999).

Existen dos tipos de transferencias de archivos

- Transferencia de archivos «Pull-based»: el receptor inicia una solicitud de transmisión de ficheros.
- Transferencia de archivos «Push-based»: el emisor inicia una solicitud de transmisión de ficheros.

Niveles en los cuales puede tener lugar

- Puede tener lugar sobre una variedad de niveles:
- Transparentes a través sistemas de archivos de red.
- Explícitas desde servicios de transferencia de archivos dedicados, como FTP o HTTP (*Hypertext Transfer Protocol*, Protocolo de Transferencia de Hipertexto).
- Distribuidas entre redes punto a punto.

- En los sistemas de mensajería instantánea.
- Computadoras y dispositivos periféricos.

La transferencia de archivos en una norma que gestiona o permite la transmisión de ficheros a través de redes informáticas o computadoras conectada en una red y los servidores que proporcionan este servicio se llaman servidores de archivos. En la actualidad es de gran utilidad, que a menudo se requiere transferir archivos diversos a diferentes computadoras o redes, y esto hace más fácil y rápida la transmisión de dichos archivos.

1.2 Descripción del proceso de transferencia de archivos

La transferencia de datos es hoy en día uno de los procesos más simples y útiles de realizar con los dispositivos electrónicos y muchos de las posibilidades están especialmente adaptadas para facilitar la tarea al usuario promedio. Esto es uno de los procesos más fáciles que permite realizar una computadora si se tiene las aplicaciones y los medios necesarios para realizar esta tarea. Esta transferencia es siempre de datos y estos pueden estar representados en diferentes estilos ya sea en material multimedia, textos, o software entre otros. De este modo uno puede acceder a diversos tipos de archivos y material en diferentes lugares si se cuenta con los métodos y dispositivos apropiados (Bembibre, 2021).

Normalmente, el proceso de transferencia se puede dar de dos maneras básicas: a través de un sistema en red o a través de un puerto, siendo el más común el conocido puerto USB (*Universal Serial Bus*, Bus de Serie Universal). Dependiendo de la calidad de los dispositivos o del método elegido, la velocidad de la transferencia podrá variar. Por otro lado, los dispositivos involucrados en el proceso deben contar con un mismo lenguaje de protocolo que los haga compatibles. En este sentido poseer computadoras en red permite acceder desde un aparato a información que está guardada en otro aparato. En el caso de los puertos USB, estos son normalmente utilizados con dispositivos externos como celulares, pen drives, impresoras, otras computadoras y dispositivos de memoria.

1.3 Análisis de herramientas de transferencias de archivos en los sistemas operativos

Actualmente existen numerosas herramientas que facilitan la transferencia de archivos, imágenes y documentos entre dos ordenadores conectados en una red LAN. Con estos programas para enviar

archivos se puede compartir documentos de gran tamaño con varias personas a la vez. Esto facilita mucho el trabajo y ahorra tiempo. Sirve especialmente para las personas que realizan labores grupales y necesitan manejar la misma información sobre el tema que están tratando.

Para el desarrollo de la investigación se estudiaron varias aplicaciones que realizan funciones similares a la transferencia de archivos, con el objetivo de obtener características de visualización o implementación en la aplicación propuesta. En este epígrafe se analizarán criterios de las aplicaciones homólogas existentes que son de gran importancia para la propuesta de solución pues pueden servir como objeto de estudio para realizar una selección acertada de estándares, herramientas y tecnologías a utilizar para su construcción. A continuación se muestran algunos programas más utilizados con una descripción de sus características fundamentales.

1.3.1 SAMBA

Es una suite de aplicaciones Unix que implementa el protocolo SMB. Este protocolo es empleado para operaciones cliente-servidor en una red. Por medio del uso de este protocolo Samba le permite a Unix establecen comunicación con productos Microsoft Windows. De esta manera, una máquina Unix con Samba puede ingresar a la red Microsoft, mostrándose como Servidor y brindar los siguientes servicios:

- Compartir diversos sistemas de archivos.
- Compartir impresoras, con instalación en el servidor como en los clientes.
- Proveer un visualizado de clientes en red, lo que facilitara la colaboración con nuestros usuarios.
- Permite realizar verificación de clientes a través de un login contra un dominio Windows.
- Proporcionar o asistir con un servidor de resolución de nombres WINS.

La utilización de un servidor SAMBA, complejiza mucho la simple finalidad de enviar un archivo. Además para enviar un archivo por este servidor es imprescindible realizar los siguientes pasos para que este funcione:

- Instalar dicho servidor en el sistema operativo.
- Configurar el servidor en dependencia de las finalidades que requiramos.
- Añadir usuarios con los cuales se pueda acceder a los recursos compartidos.

- Asignar los permisos adecuados a cada carpeta a compartir.
- Iniciar el servidor SAMBA, para que se pueda acceder a los archivos compartidos en el mismo.

1.3.2 SSH/SCP

El SCP de Linux es un protocolo de transferencia de archivos en red que permite realizar el proceso de forma fácil y segura entre un host remoto y uno local, o entre dos ubicaciones remotas. En esencia, SCP es una mezcla de RCP (Protocolo de Copia Remota) y SSH. Se basa en el primero para realizar operaciones de copia y en el segundo para cifrar la información y autenticar los sistemas remotos.

Aspectos a tener en cuenta

Como SCP utiliza el cifrado SSH, se necesita la contraseña ssh para que se realice la transferencia de archivos. Además, es necesario tener permiso de lectura en la máquina de la que estás a punto de copiar y privilegios de escritura en la(s) máquina(s) a la que copie.

Para la autenticación y la configuración de la conexión, se genera un par de llaves ssh en el terminal mediante comando.

Una vez se haya autenticado en la(s) máquina(s) remota(s), la llave pública se copiará y se estará listo para iniciar las transferencias.

1.3.3 WeTransfer

Este es uno de los programas más conocidos por los usuarios para compartir archivos y se basa en la transferencia por medio de la nube. Es gratuita hasta 2Gb, si el usuario desea transferir archivos que superen esta capacidad, tendrá que realizar una suscripción mensual. No es necesario registrarse con ninguna cuenta de usuario para realizar el envió de archivos, como tampoco la persona que recibe la información no es de carácter obligatorio crearse una cuenta, así se evita los pasos tediosos del registro. Para la trasmisión de fichero utilizando esta aplicación es necesario realizar los siguientes pasos:

1. Acceder a la página web de WeTransfer.
2. Agregue los archivos que quiere enviar.
3. Escriba la dirección de correo electrónico del usuario receptor.
4. Coloca tu correo personal o profesional.

5. Puedes agregar una nota para que el destinatario identifique tus archivos y luego enviar, así tus archivos serán compartidos.

1.3.4 Google Drive

Es una aplicación diseñada para transferir archivos de textos u hojas de cálculo. Permite el acceso a estos archivos en cualquier dispositivo teniendo siempre asociado el mismo correo electrónico. Brinda 15Gb de almacenamiento de forma gratuita para usuarios que no sean Premium. Tiene como requisito fundamental para compartir archivo, realizarlo utilizando obligatoriamente un correo personal de Gmail. Para la trasmisión de fichero utilizando este programa es necesario realizar los siguientes pasos:

1. Acceder a Google Drive e inicia sesión con tu cuenta de Gmail.

2. Seleccionar el archivo que se desea compartir.

3. Seleccionar la opción Compartir dando clic derecho o directamente en el ícono.

4. Agregar el nombre o correo electrónico del o de los destinatarios y asignar los permisos a cada usuario.

5. De forma opcional se puede agregar una nota para que el destinatario identifique los archivos y luego enviar, así los archivos serán compartidos.

1.3.5 Dropbox

Esta herramienta permite tanto almacenar archivos como compartirlos con terceros de una manera fácil, rápida y segura. Dropbox al igual que Google Drive permite acceder a los archivos mediante cualquier dispositivo creando una cuenta o iniciando sesión con Gmail. Posee una versión gratuita que asigna un espacio de 2Gb y otra de pago, la cual otorga más almacenamiento. Cuenta con una aplicación para ser instalada en el ordenador, de esta manera, los cambios que realices en los archivos de la carpeta de Dropbox serán actualizados automáticamente. Para compartir los archivos utilizando este programa es necesario realizar los siguientes pasos:

1. Acceder a Dropbox y crea una cuenta o inicia sesión.
2. Seleccionar el archivo o carpeta que se desea compartir.

3. Hacer clic en el botón “Compartir”.

4. Agregar el correo electrónico del receptor. Si se desea, se puede colocar un mensaje o una nota.

5. Por último, dar clic en “Compartir”, o si se prefiere se puede generar un enlace y a través de este compartir los archivos.

1.3.6 TeamViewer

Es una herramienta de escritorio remoto patentada gratuita (para uso personal). Admite conexiones a computadoras remotas a través de Internet y LAN. Los usuarios a menudo recurren a TeamViewer, especialmente en Linux debido a su fácil configuración y funcionalidades adicionales, como chat de texto incorporado y transferencia de archivos.

Características notables

- Se integra con el sistema systemd init, haciendo que Teamviewer sea fácil de encender y apagar cuando no esté en uso.
- Tiene soporte de transferencia de archivos, chat de video, pizarra y soporte de chat de texto para facilitar mucho la ayuda a los demás.
- Fácil de instalar en Linux, gracias a los paquetes DEB / RPM rápidos y fáciles de descargar.
- El grabador de sesión incorporado permite a los usuarios mantener registros de las sesiones de conexión remota.
- Permite a los usuarios guardar conexiones remotas como «amigos» en una lista de contactos, para facilitar el acceso.

1.3.7 ShareIt

Esta aplicación, disponible para Windows, macOS, Android y iOS, permite transferir archivos entre dispositivos a velocidades altas conectándose de manera directa entre ellos (sin necesidad de Internet o de pasar por un router) alcanzando una velocidad de copia superior a los 20 MB/s, mucho más rápidas que si utilizáramos otros protocolos como, por ejemplo, el Bluetooth. Cuando transferimos archivos mediante este programa, no afectamos la calidad de los archivos, ni su tamaño disminuye, a diferencia de cuando lo hacemos mediante de redes sociales como el WhatsApp o mediante WiFi-Direct.

1.3.8 Dukto, fácil transferencia de archivos.

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. Resulta muy útil a usuarios que comparten archivos con frecuencia, permitiéndolo hacer de forma rápida y sencilla, 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 Wi-Fi o cable. Dukto buscará automáticamente los dispositivos disponibles en la LAN y los mostrará sobre él.

¿Cómo funciona Dukto?

1. Dukto funciona de forma simple, tras instalarlo en los dispositivos deseados, simplemente se debe ejecutar para realizar la transferencia.

2. Al ejecutar Dukto el menú buddies contiene los dispositivos que están ejecutando la aplicación. Si el destinatario no tiene activo el Dukto, este no saldrá en la lista.

3. Para transferir tan solo se debe escoger el equipo al cual desea enviar archivos. El Dukto por defecto identifica las PC por el nombre de las mismas.

4. Al seleccionar el dispositivo se habilitará el menú que les permite enviar mensajes de textos, además de archivos independientes o alguna carpeta completa. Por otro lado, brinda la posibilidad de arrastrar cualquier fichero o carpeta sobre cualquier dispositivo de la lista para enviárselo directamente.

1.4 Análisis comparativo de aplicaciones para la transferencia de archivos

A continuación, se hace un análisis de las herramientas de transferencia de archivos con el objetivo de conocer si cumplen con las necesidades existentes en la distribución de GNU/Linux Nova para la transferencia de archivos. En la comparación se utilizaron seis criterios de análisis definidos a partir de las deficiencias identificadas en la situación problemática y las características del contexto del negocio y de la propuesta de solución.

Criterios de análisis:

Uso imple

Herramienta nativa de la distribución GNU/Linux Nova

Código fuente disponible

Interfaz gráfica

Compatible con Linux

Transferencia sin internet

Tabla 1. Resultado del análisis de las herramientas de transferencia de archivos

(Fuente: elaboración propia)

Criterios de análisis	**Herramientas seleccionadas**							
	SAMBA	**SSH/SCP**	**WeTransfer**	**Google Drive**	**Dropbox**	**TeamViewer**	**ShareIt**	**Dukto**
Uso simple	No	No	No	No	No	No	No	Sí
Herramienta nativa	No	No	No	No	No	No	No	No
Código fuente disponible	Sí	Sí	No	No	No	No	No	No
Interfaz Gráfica	No	No	Sí	Sí	Sí	Sí	Sí	Sí
Instalable en Linux	Sí	Sí	Sí	Sí	Sí	Sí	No	Sí
Transferencia sin Internet	Sí	Sí	No	No	No	Sí	Sí	Sí

El análisis anterior de los sistemas homólogos demuestra la necesidad de implementar una nueva solución ya que estas herramientas no resuelven las dificultades que se presentan para transferir archivos en el sistema operativo Nova. Además sirvió para tomar experiencias y tener en cuenta características y funcionalidades de los mismos para la propuesta a implementar.

1.5 Metodología de desarrollo de software

Una metodología de desarrollo de software se refiere al entorno que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de informático. Suele estar documentada y promovida por algún tipo de organización ya sea esta pública o privada. Tienen como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, exigiéndose así que el proceso sea configurable. Se clasifican en dos clases: las metodologías tradicionales o robustas y las ágiles o ligeras (Pressman, 2010). La metodología seleccionada para el desarrollo de la propuesta de solución es **Variación de AUP para la UCI**

1.5.1 Variación de AUP para la UCI

La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP (*Agile Unified Process*, Proceso Unificado Ágil), 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(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 utiliza la fase de Ejecución y transitó las disciplinas propuestas en la metodología: 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

1.6 Lenguajes y herramientas para el modelado de la solución

Se utilizaron los siguientes lenguajes y herramientas para el modelado de la propuesta de solución.

1.6.1 Lenguaje Unificado de Modelado

UML (*Unified Modeling Language*, Lenguaje de Modelado Unificado) es el estándar industrial de la notación de modelado para sistemas orientados a objetos, constituye la plataforma inicial para el desarrollo rápido de aplicaciones. Es un lenguaje gráfico para visualizar, especificar, construir y documentar artefactos de un sistema de software (Xiao He, Zhiyi Ma, Weizhong Shao, 2017). Representar mediante diagramas los diferentes procesos de un sistema, permite que los programadores entiendan de forma clara lo que deben hacer. El lenguaje que se usa para modelar los diagramas del sistema es UML en su versión 2.5 debido a que proporciona amplias posibilidades de representar clases y procesos. Todo software debe tener una estructura definida según patrones, modelos y diagramas. Se le nombra arquitectura del sistema a esta estructura.

- Puede usarse en las diferentes etapas del ciclo de vida del desarrollo de sistemas.
- Es independiente del proceso o metodología de desarrollo y del lenguaje de implementación.
- Permite crear y modificar modelos expresivos mediante la combinación de los 13 tipos de diagramas que suministra en esta versión.
- Es posible extender la funcionalidad de la notación gráfica mediante estereotipos y proveer una base formal para los diagramas (Xiao He, Zhiyi Ma, Weizhong Shao, 2017).

1.6.2 Herramientas de modelado

Visual Paradigm es una herramienta CASE (*Computer Aided Sotware Engineering*, Ingeniería de software Asistida por computadora) que soporta el lenguaje UML. La herramienta está diseñada para una amplia gama de usuarios, incluidos los ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquier persona que esté interesada en la construcción de sistemas de software fiable a gran escala con un enfoque orientado a objetos. Además, soporta los últimos estándares de la notación UML (CNET, 2019).

La herramienta Visual Paradigm es compatible con las metodologías existentes para el desarrollo de software. Posee características gráficas cómodas que facilitan la realización de los diagramas de modelado que sigue el estándar de UML (UML CASE Tool for Software Development, 2021). La elección de la herramienta Visual Paradigm en su versión 8.1 está determinada por su condición de multiplataforma, resulta fácil de usar, instalar y actualizar; permite el diseño y modelado de los diagramas generados durante el ciclo de vida del desarrollo del software.

1.7 Tecnologías de implementación

Una herramienta es un objeto creado para contribuir al desarrollo de software. 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.

1.7.1 Lenguaje de programación

En la implementación de la propuesta de solución se utilizará el lenguaje de programación **Python** en su versión **3.5**.

Python es un lenguaje orientado a objetos, potente, es flexible debido a su capacidad para utilizar componentes que fueron diseñados en otra programación. Puede admitir diferentes estilos de programación como estructural. Es de código abierto y como tal cualquiera puede contribuir a su desarrollo y divulgación(“Python-The Fastest Growing Programming Language,” 2017).

Las principales características que tiene este lenguaje son (*Lenguajesdeprogramacion.Net.*, 2018):

- Lenguaje de programación multiparadigma (programación orientada a objetos, programación estructurada y programación funcional).

- Utiliza el tipo de dato dinámico para el manejo de memoria.
- Permite la resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa.
- Puede utilizarse como un lenguaje de extensión para módulos y aplicaciones que necesitan de una interfaz programable.

1.7.2 Entorno de desarrollo Integrado

Es un paquete de software que consolida las herramientas básicas necesarias para escribir y probar un software (Rouse, 2018). Es una aplicación destinada a brindar servicios integrales al desarrollador en su trabajo, o dicho más simple, es un programa que permite construir código de una forma más sencilla o didáctica (Blancarte, 2021).

En la implementación de la propuesta de solución se utilizará el entorno de desarrollo integrado **PyCharm** en su versión **2019.3**.

PyCharm, es un entorno de desarrollo integrado **multiplataforma** utilizado en el ámbito de la programación. Viene con una consola de Python donde puede escribir los scripts a medida que los ejecuta. Adicionalmente, provee capacidades de alto rango para desarrolladores profesionales de web con el marco de trabajo Django (Naranjo, 2019).

Las principales características que ofrece son (Pythonízame, 2016):

- Editor inteligente.
- Depurador de código gráfico.
- Inspección de código.
- Integración de control de versiones.
- Ejecución de pruebas.
- Soporta los repositorios JIT, Mercurial, Subversión y Github.
- Permite los leguajes XML (Lenguaje de Marcado Extensible) y HTML (Lenguaje de Marcas de Hipertexto).
- Posee una terminal local.

1.7.3 Herramienta de diseño de interfaz gráfica

GTK (Kit de herramientas de GIMP) es gratis y de código abierto multiplataforma kit de herramientas de widgets, es utilizado para crear interfaces gráficas de usuario (GUI), de los entornos de escritorio y de la totalidad de programas que se ejecutan sobre el sistema operativo. Se ha diseñado para permitir programar con lenguajes como C, C++, C#, Java, Ruby, Perl, PHP o Python. Licenciado bajo los términos de LGPL(*Lesser General Public License*, Licencia Pública General Reducida). Se utiliza en su versión 4.2.1 para el diseño de la interfaz gráfica de la propuesta de solución.

1.7.4 Herramienta de control de versiones

Un sistema de control de versiones registra todos los cambios hechos en uno o más proyectos, guardando así versiones del producto en todas sus fases del desarrollo. Las versiones son como fotografías que registran su estado en ese momento del tiempo y se van guardando a medida que se hacen modificaciones al código fuente (Chacon, 2015). Algunos de los sistemas de control de versiones más famosos son Subversión (también conocido como Svn), Mercurial 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 fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida. La rapidez en la gestión distribuida, la gestión de las ramas, la gestión eficiente de proyectos grandes y el realmacenamiento periódico de paquetes son algunas de sus características más importantes (Chacon, 2015).

Conclusiones del capítulo

El análisis del marco teórico sobre el proceso de transferencias de archivos en Nova y el estudio de los principales conceptos asociados al problema planteado, permitieron establecer las bases para el desarrollo de la investigación y conocer las características del objeto de estudio. A partir de la caracterización y comparación de varios programas para la transferencia de archivos, se demostró la necesidad de desarrollar una aplicación que aproveche las mejores funcionalidades que estos ofrecen, pues no resuelven la totalidad de las insuficiencias presentes en Nova. En el desarrollo de la propuesta de

solución se definió el empleo de herramientas informáticas en su mayoría libres para garantizar la soberanía tecnológica, y la utilización de la metodología de desarrollo de software Variación de AUP para la UCI.

Capítulo 2: Análisis y diseño de una herramienta nativa para la transferencia de archivos en la Distribución Cubana GNU/Linux Nova

En el presente capítulo se identifican los requisitos funcionales y no funcionales necesarios para el correcto funcionamiento de la herramienta para la transferencia de archivos. 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 Descripción de la propuesta de solución

Luego de identificadas las dificultades que existen para la transferencia de archivos entre dispositivos que posean el SO Nova y Windows, considerando además el resultado del análisis de aplicaciones homólogas y la entrevista con el cliente 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.
- Brindará posibilidad de enviar y recibir fragmentos de texto.
- Posibilitará abrir archivos recibidos directamente desde la aplicación. También proporcionará la opción de elegir la carpeta en la que se quiere guardar la información recibida.
- Mostrará las direcciones IP que se estén utilizando.
- El código fuente estará disponible en los repositorios de nova

2.2. Descripción del negocio a informatizar

Se elaboró un modelo conceptual con el objetivo de lograr una mejor comprensión del contexto del negocio a informatizar en la aplicación, abordando los conceptos tratados, sus principales atributos y las relaciones entre ellos.

2.2.1 Modelo conceptual

Un modelo conceptual de negocio conocido también como modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades (Francisco José García Peñalvo, 2018). Su utilidad radica en ser una forma de "inspiración" para el diseño de los objetos software. Es entrada para muchos de los artefactos que se construyen en un proceso software. Un modelo de dominio muestra las clases conceptuales significativas en un dominio del problema Se centra en las abstracciones relevantes, vocabulario del dominio e información del dominio. Es el artefacto clave del análisis orientado a objetos. En UML se utilizan los diagramas de clases para representar los modelos de dominio. A continuación, en la figura 1 se muestra el modelo conceptual de la propuesta de solución.

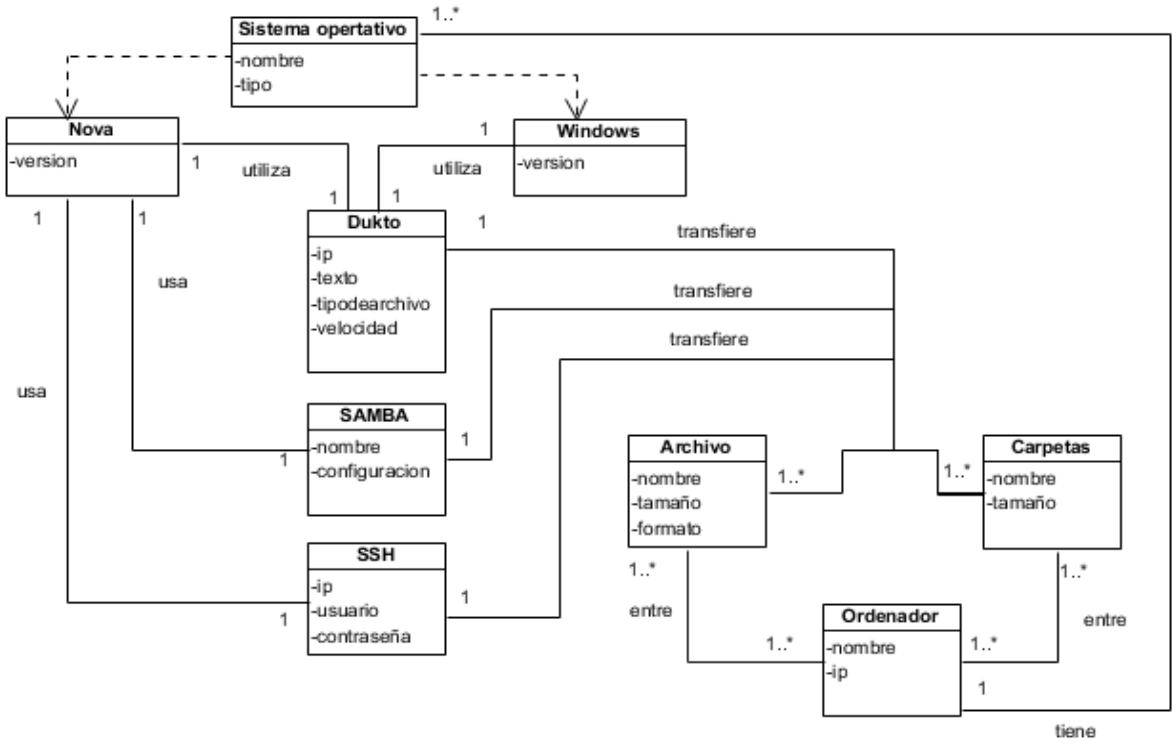


Figura 1 Modelo conceptual

(Fuente: elaboración propia)

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.

Nova: es una distribución cubana de GNU/Linux desarrollada en la Universidad de las Ciencias Informáticas.

Windows: familia de distribuciones de software para PC, servidores, sistemas empotrados, desarrollados y vendidos por Microsoft.

Dukto: es una aplicación de escritorio para la transferencia de archivo entre ordenadores conectados en una misma red de área local.

SAMBA: es un proyecto de software libre que implementa el protocolo de archivos compartidos de Windows para Sistemas operativos de tipo UNIX**.**

SSH: es un protocolo de administración remota, que les permite a los usuarios controlar y modificar sus servidores remotos. Además permite copiar datos de forma segura.

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.

Ordenadores: son máquinas que almacena y automatiza la información a través de programas informáticos diseñados específicamente para esta tarea.

2.3. Requisitos

Identificar los requisitos funcionales y no funcionales es de suma importancia cuando se va a desarrollar un software. La correcta obtención de los requisitos es uno de los aspectos más críticos de un proyecto software, independientemente del tipo de proyecto que se trate, dado que una mala captura de los mismos es la causa de la mayor parte de los problemas que surgen a lo largo del ciclo de vida(José & Peñalvo,

2018). 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 (Sánchez, 2015).

2.3.1 Fuentes de obtención de requisitos

Las fuentes que proporcionaron la información, que permitieron obtener las características y requisitos que debe tener la propuesta de solución son los siguientes:

- Análisis de los sistemas homólogos existentes.
- Especialistas de CESOL.
- Modelo conceptual elaborado en la descripción del contexto del negocio.

2.3.1. Técnicas de identificación de requisitos

En los proyectos de ingeniería de software una etapa elemental es la identificación y documentación de los requerimientos del futuro sistema al comienzo de este, pues pueden prevenirse errores que conllevan al fracaso del proyecto. Para el adecuado levantamiento de los requerimientos de un sistema se usan técnicas de identificación de requisitos, las cuales permiten identificar las necesidades de negocio de los clientes y los 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 (Pressman, 2010). A continuación, se especifican las técnicas usadas:

Entrevista: es una técnica de recogida de información que además de ser una de las estrategias utilizadas en procesos de investigación, tiene ya un valor en sí misma. Es de gran utilidad para adquirir información cualitativa como opiniones o descripciones subjetivas de actividades (Folgueiras, n.d.). La entrevista fue realizada a un usuario del sistema operativo en CESOL lo que permitió la obtención de los primeros requisitos (Ver Anexo 1).

Observación: Es una técnica útil cuando se está documentando la situación actual de procesos de negocio. Consiste en estudiar el entorno de trabajo de los usuarios, clientes e interesados de proyecto. Esta permite observar la manera en que se llevan a cabo los procesos y verificar que realmente se sigan todos los pasos especificados. 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, lo que permitió refinar e incorporar nuevos requisitos a los existentes.

Desarrollo de prototipos: los prototipos suelen consistir en vistas reducidas de la aplicación a desarrollar. Permiten a los usuarios experimentar para ver cómo este ayuda a su trabajo, fomentan el desarrollo de ideas que desembocan en requerimientos, mejoran las especificaciones de requerimientos, identifican discrepancias entre los desarrolladores y los usuarios y permite formalizar la aceptación previa por parte del cliente de los requisitos del proyecto (Guerra, 2017)

Con los requisitos obtenidos mediante la aplicación de la Entrevista y la Observación se realizaron los prototipos de las interfaces lo que posibilitó verificar la consistencia y completitud de los mismos. Además, permitió la identificación del resto de los requisitos, una vez que el programador, la analista principal y el jefe de proyecto de Nova interactuaron con estos. La aplicación de estas técnicas propició la identificación de los requisitos que se describen a continuación.

.

2.3.2. Especificación de requisitos de software

La especificación de requisitos de software (ERS) es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de casos de uso que describe todas las interacciones que tendrán los usuarios con el software. Sirve como medio de comunicación entre clientes, usuarios, ingenieros de requisitos y desarrolladores. Es la etapa donde se recoge las necesidades de clientes y usuarios, permite obtener los requisitos que debe cumplir el software a desarrollar para satisfacer todas las necesidades (andalucia, 2017). A continuación se relacionan y describen los requisitos funcionales y no funcionales definidos para la implementación de la propuesta de solución.

Requisitos Funcionales

Los requisitos funcionales (RF) son las funcionalidades con las que va a contar el sistema, o sea son la parte esencial a la hora de desarrollar el software. Dan una perspectiva eficaz de las necesidades que tiene el cliente y que se necesitan implementar. Los requisitos funcionales (RF) del sistema son aquellos servicios que el usuario espera del sistema, deben ser completos y consistentes (Prado, 2018). A continuación se muestran los requisitos funcionales de la propuesta de solución, su descripción y prioridad.

RF 1 Listar dispositivos: permite la visualización de la lista de los dispositivos conectados a la misma subred.

RF 2 Insertar IP: permite insertar de forma manual el IP de la computadora la que se desea enviar algún archivo.

RF 3 Enviar texto: posibilita enviar mensajes de textos al dispositivo de destino.

RF 4 Transferir archivos: permite transferir pequeños o grandes ficheros.

RF 5 Transferir carpeta: permite enviar una carpeta con gran cantidad de información.

RF 6 Mostrar Historial: permite visualizar el historial de la información recibida.

RF 7 Modificar ajustes: esta funcionalidad le posibilita al usuario, modificar el color de la interfaz y la dirección de destino de los archivos recibidos, acceder a la lista de todas las direcciones de IP de los adaptadores de red, permite ir a la ubicación de destino mediante un acceso directo.

Requisitos no funcionales

Los requisitos no funcionales (RNF) son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Casi nunca se aplican a características o servicios individuales del sistema (Pressman, 2010).

RNF 1 Apariencia o interfaz externa: el diseño de la interfaz debe ser simple, sencillo y ligero. El sistema debe contar con una interfaz de fácil navegación, amigable, que permita al usuario trabajar de forma sencilla y cómoda. La distribución de colores debe de ser agradable a la vista.

RNF 2 Portabilidad: la herramienta estará ejecutándose sobre Nova pero será capaz trasferir archivos hacia otros sistemas.

RNF 3 Confiabilidad: la herramienta debe funcionar sin que se produzcan errores o con el mínimo posible que no afecte el correcto funcionamiento de esta.

RNF 4 Usabilidad: el idioma de todas las interfaces de la herramienta será español, expondrá el menú general desde cualquiera de sus interfaces.

RNF 5 Funcionalidad: la herramienta ejecutará las operaciones indicadas en cada momento.

RNF 6 Requerimientos de software: un ordenador con Nova.

2.3.3. Descripción de requisitos de software.

Una historia de usuario (HU) es la representación de un requisito funcional escrito en el lenguaje natural. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea implementar cada funcionalidad (Montero, 2018). 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 insertar IP y enviar texto.

Tabla 3. Historia de usuario Insertar IP.

(Fuente: elaboración propia)

Número: HU2	Requisito: ***Insertar IP***	
Programador: Leonardo Mastrapa Reynaldo		Iteración Asignada: 1
Prioridad: Alta		Tiempo Estimado: 72 horas
Riesgo en Desarrollo: Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.		Tiempo Real: 52 horas
Descripción: El usuario debe ingresar el IP al que desea realizar la transferencia, luego selecciona la opción deseada, como transferir texto, transferir archivo o transferir carpeta.		
Observaciones: El IP ingresado debe ser válido y encontrarse en la misma red.		
Prototipo elemental de interfaz gráfica de usuario:		

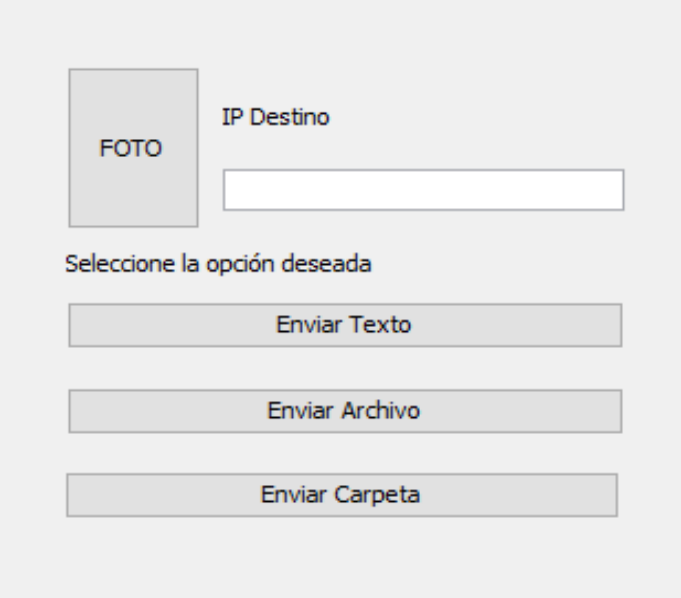


Figura 2. Prototipo de RF Insertar IP

(Fuente: elaboración propia)

Tabla 4. Historia de usuario del RF Enviar Texto.

(Fuente: elaboración propia)

Número: HU3	Requisito: *Enviar texo*
Programador: Leonardo Mastrapa Reynaldo	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 72 horas
Riesgo en Desarrollo: Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.	Tiempo Real: 52 horas
Descripción: El usuario ingresa el texto que desea enviar y selecciona el botón enviar.	

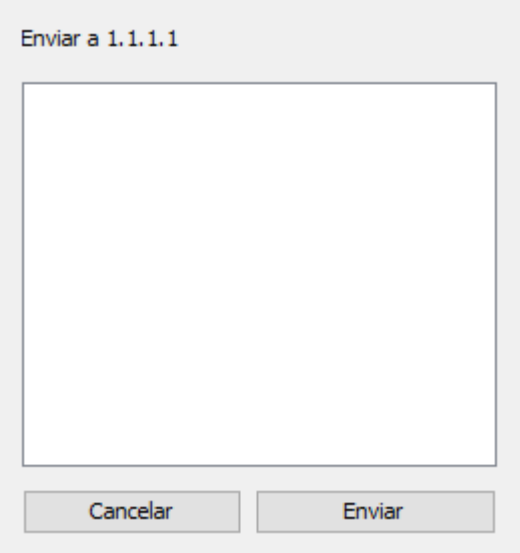
Observaciones: En el texto ingresado por el usuario pueden existir letras, números y caracteres especiales.
Prototipo elemental de interfaz gráfica de usuario: 

Figura 3. Prototipo de RF Insertar IP

(Fuente: elaboración propia)

2.4. 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 (Sánchez, 2015).

2.4.1 Diseño arquitectónico

Se define la arquitectura de software como la estructura del sistema, que comprende elementos de software, las propiedades de esos elementos visibles y las relaciones entre ellos. Esta conforma el esqueleto de cualquier sistema, y es la principal responsable de los atributos de calidad. Una arquitectura

adecuada, diseñada de forma correcta, documentada y evaluada, constituye la base para que un proyecto finalice con éxito (Len Bass, Paul Clements, y Rick Kazman, 2019).

Los patrones arquitectónicos definen la estructura de un sistema, a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes, con el objetivo de facilitar la tarea del diseño del sistema (Reynoso, 2021).

La arquitectura en **capas** es una de las más utilizadas, no solo por su simplicidad, sino porque también es utilizada por defecto cuando no estamos seguros que arquitectura debemos de utilizar para nuestra aplicación. Consta en dividir la aplicación en capas, con la intención de que cada capa tenga un rol muy definido, como podría ser, una capa de presentación, una capa de reglas de negocio y una capa de acceso a datos, sin embargo, este estilo arquitectónico no define cuantas capas debe de tener la aplicación, sino más bien, se centra en la separación de la aplicación en capas (Oscar Blancarte, 2018).

- La capa de persistencia es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar otros nuevos. Encapsula los datos y las funcionalidades.
- La capa de presentación es por lo general la interfaz de usuario, es decir muestra la información al mismo. Presenta el modelo en un formato adecuado para interactuar.
- La capa de negocio responde a eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, es decir acciones del usuario e invoca cambios en el modelo y en la vista.

En la Figura 2 se muestran la sucesión de procesos que realiza el estilo en capas.

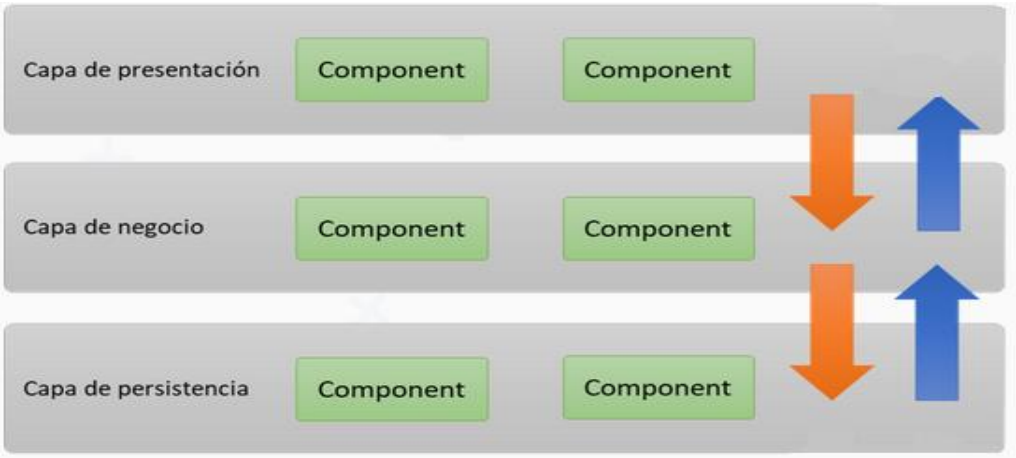


Figura 4. Ciclo del patrón en Capas

(Fuente: reactiveprogramming.io)

A continuación, se muestra en la figura 3 el diagrama de paquetes donde se representa la arquitectura de la herramienta según el estilo arquitectónico.

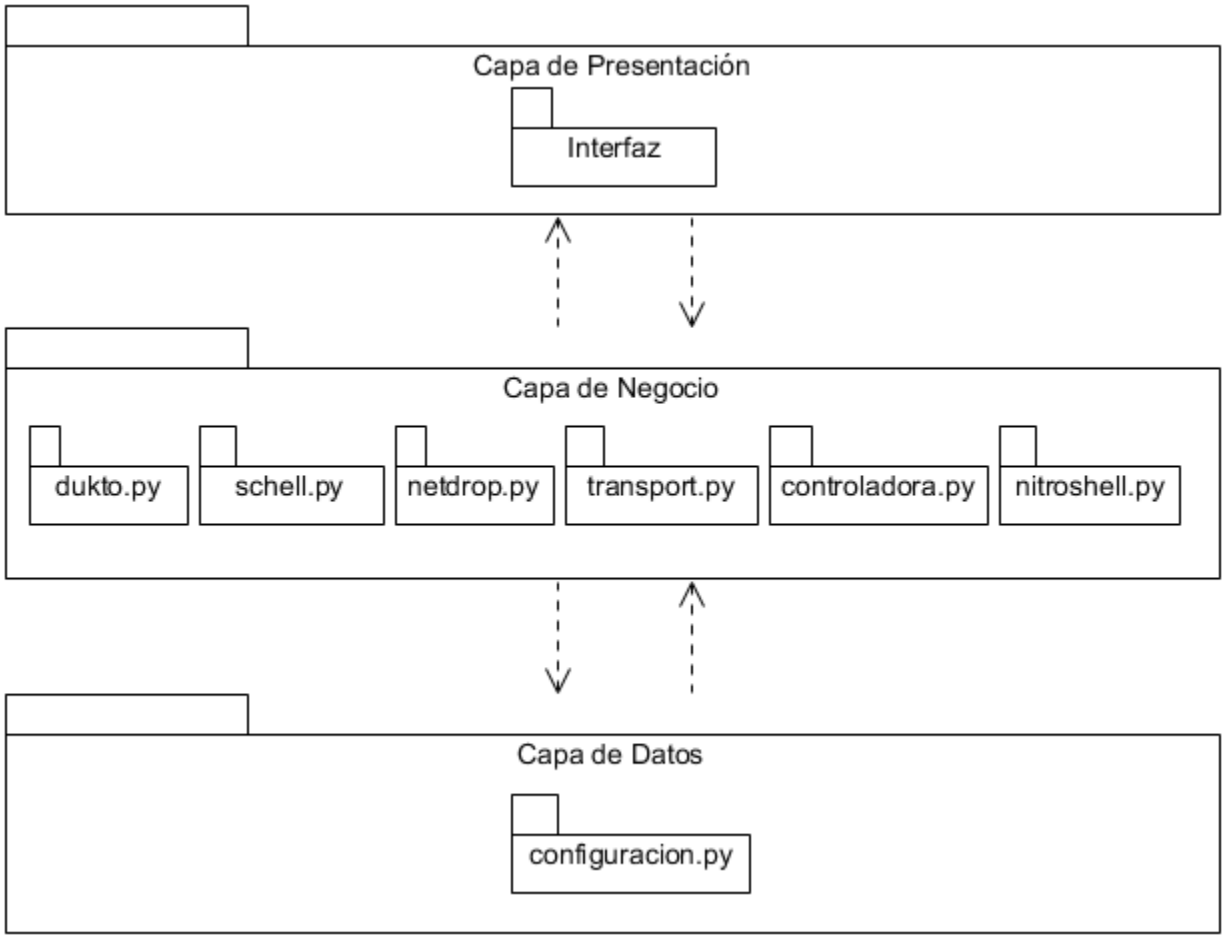


Figura 5. Diagrama de paquete

(Fuente: elaboración propia)

2.4.2 Modelo de datos

Un modelo de datos es un conjunto de conceptos y reglas que permiten estructurar los datos resultantes de la observación de la realidad, de forma que pueden ser representadas todas sus propiedades, tanto estáticas como dinámicas. Los modelos de datos de alto nivel o conceptuales utilizan conceptos muy cercanos a la forma en que los usuarios perciben los datos, mientras que los de bajo nivel o físicos describen detalles de cómo se almacenan los datos en la computadora (Gómez, 2013).En la tabla 5 se presentan los atributos que persistirán en el fichero de pre configuración del ISO.

Tabla 5. Persistencia de los datos

(Fuente: elaboración propia)

Atributo	Tipo de dato	Estructura de persistencia
Carpeta de destino	String	*/etc/apt/duktonova*
Color del tema	String	*F4A12*
Nombre en la red	String	*NovaPC*

Carpeta de destino: Es la carpeta donde el usuario desea que se guarden los archivos recibidos.

Color del tema: Color deseado por el usuario para la interfaz.

Nombre en la red: Nombre el cual los demás usuarios visualizan al dispositivo.

2.4.3. 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 (Pressman, 2010). Seguidamente, se presenta el modelado del diseño de la propuesta de solución.

Diagrama de clases del diseño

El diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia (Sistemas groupware para el diseño de diagrama de clases UML en ambientes táctiles., 2018). Además se considera como la presentación gráfica de la vista estática, que muestra una colección de elementos declarativos (estáticos) del modelo, como clases, tipos, contenidos y relaciones. Contiene ciertos elementos materializados de comportamiento, como operaciones, pero cuya dinámica se puede representar en otros diagramas, como diagramas de estados o diagramas de colaboración (Ferrer, 2016). A continuación, la figura 4 muestra el diagrama de clase.

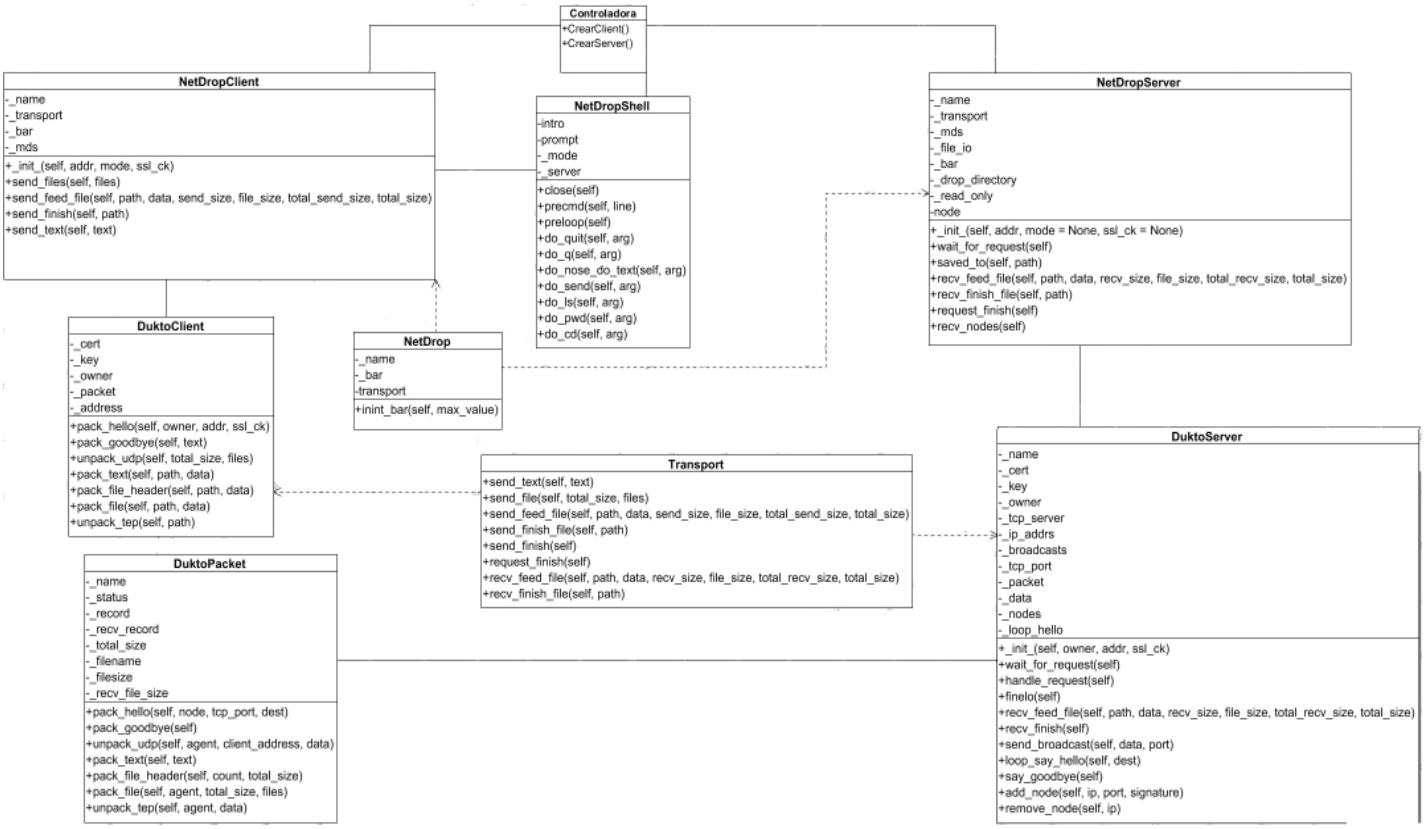


Figura 6. Diagrama de clase

(Fuente: Elaboración propia

Patrones del diseño de software

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Estas soluciones son basadas en la experiencia y se ha demostrado que funcionan. Estos en esencia no son fáciles de entender pero, una vez entendido su funcionamiento permiten obtener diseños mucho más flexibles, modulares y reutilizables (Gracia, 2019).

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) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (*Ingeniería de Software. UCI*, 2018).

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 (Peréz, 2013). Este patrón es usado de forma general en el proyecto implementado ya que ninguna de las clases tiene muchas dependencias de otras, la máxima relación entre clases es tres. Un ejemplo es en la clase NetDrop la cual para cumplir con sus funciones solo necesita relacionarse con las clases NetDropClient y NetDropShell.

Experto: su misión es asignar una responsabilidad a la clase que cuenta con la información necesaria para cumplir la tarea. Este patrón se evidencia en la aplicación, debido a que las clases se diseñan con el objetivo de que las mismas tengan la mayor información referente al método a desarrollar. Se utiliza en la clase Controladora ya que es la que contiene toda la información y las funcionalidades necesarias para crear un cliente o servidor.

Creador: Es el patrón que define quién debe ser el responsable de crear una nueva instancia de la clase. Asignar a la clase B la responsabilidad de crear una instancia de la clase A (Peréz, 2013). El patrón se evidencia en la clase Controladora la cual tiene responsabilidad de crear instancias de NetDropClient y NetDropServer.

En la figura 5 se muestra la aplicación de los patrones GRAPS en la propuesta de solución.

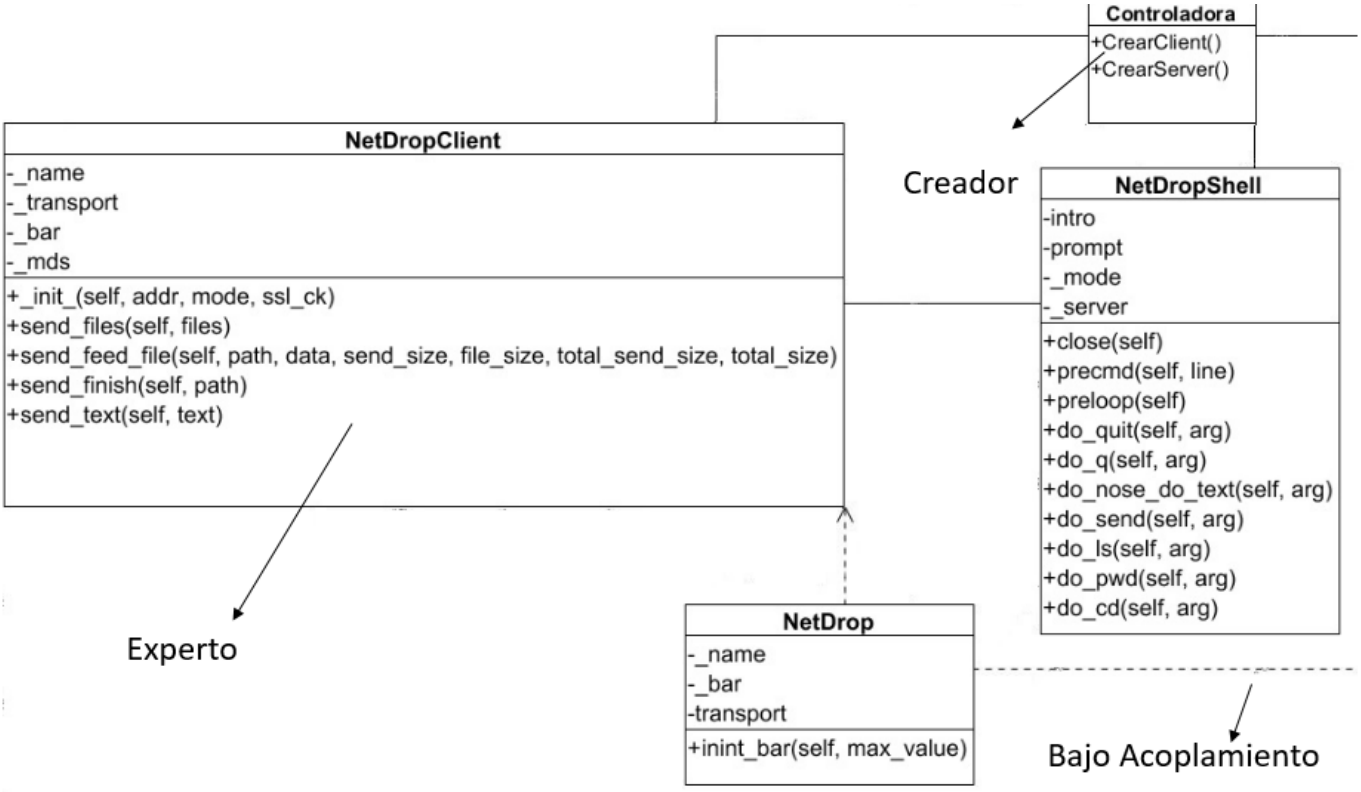


Figura 7. Aplicación de los patrones GRAPS

(Fuente: elaboración propia)

Conclusiones del capítulo

En el desarrollo del capítulo se transitó por las disciplinas, Requisitos y Análisis y diseño. Se relacionaron los conceptos fundamentales asociados a la transferencia de archivos a través de un modelo conceptual que permitió conocer las características del contexto del negocio informatizado. Se identificaron 7 requisitos funcionales y 6 no funcionales que garantizó el cumplimiento de las necesidades del cliente. Se elaboró el diagrama de clases empleando los patrones de diseño GRASP y se modeló la arquitectura basada en el estilo en capas que garantizaron el cumplimiento y la calidad del diseño de la herramienta para la transferencia de archivos de la Distribución Cubana GNU/Linux Nova.

Capítulo 3: Implementación, pruebas, validación y evaluación de la herramienta nativa para la transferencia de archivos en Nova.

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

En la disciplina implementación, a partir de los resultados del Análisis y diseño se construye el sistema (Sánchez, 2015).

3.1.1 Modelo de implementación

El Modelo de implementación es el proceso de convertir una especificación del sistema en un sistema ejecutable. Describe cómo los elementos de diseño se implementan en componentes, estos componentes incluyen: ficheros ejecutables, ficheros de código fuente y otros tipos de ficheros necesarios para la implementación y el despliegue del sistema. En este modelo se describen las relaciones entre los paquetes y las clases el diseño con los diferentes subsistemas y componentes físicos (Cardoso, 2014).

Diagrama de componentes

Un diagrama de componentes muestra la organización, relaciones y dependencias entre los componentes de un sistema (partes modulares, desplegables y reemplazables de un sistema que encapsula implementación y expone un conjunto de interfaces) (Cardoso, 2014). A continuación se presenta el diagrama de componentes para la propuesta de solución.

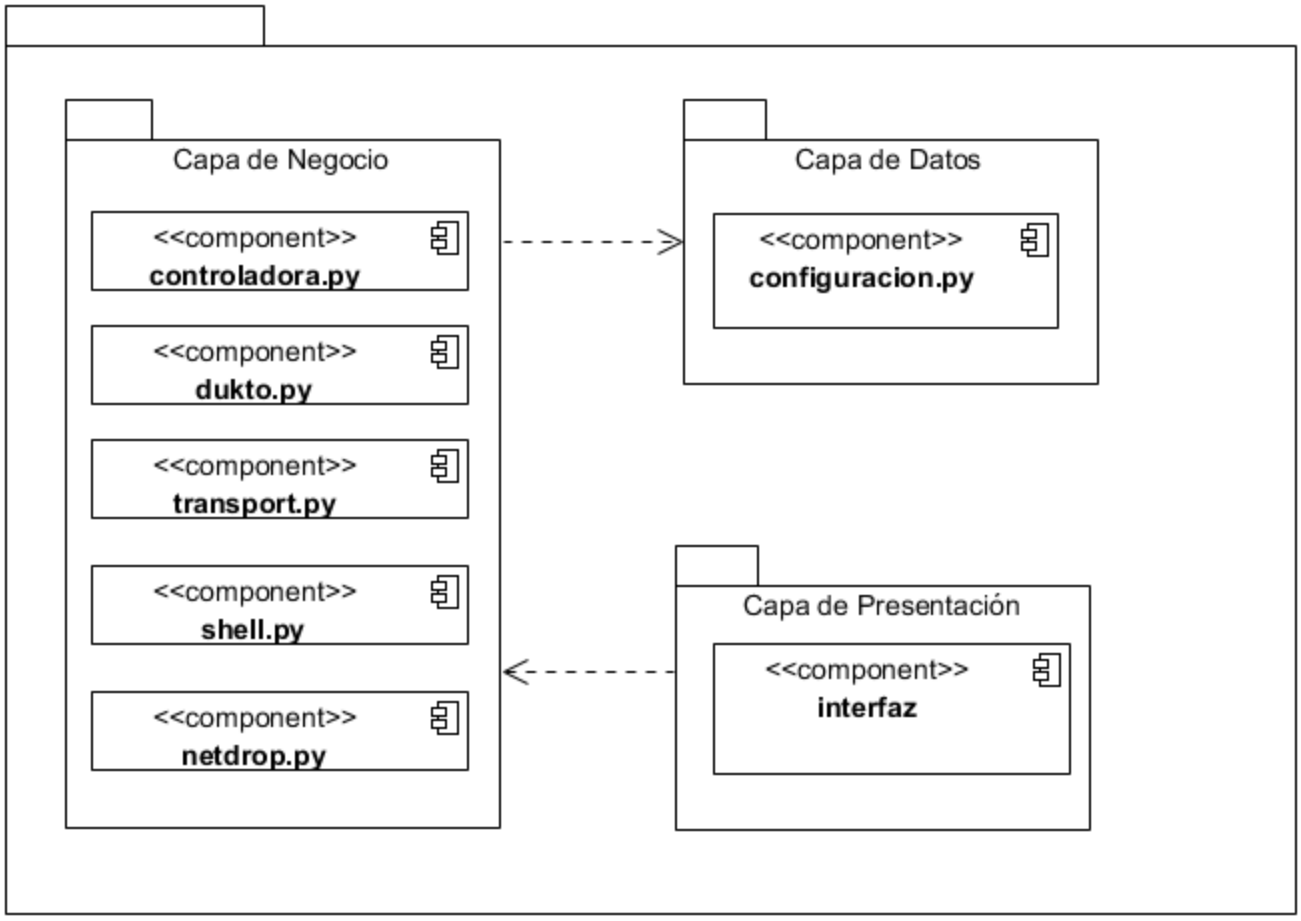


Figura 8. Diagrama de componente

(Fuente: elaboración propia)

Diagrama de despliegue

El diagrama de despliegue permite modelar la disposición física de un sistema. Muestra el hardware usado y los componentes instalados en el hardware, además de las conexiones físicas entre este y las relaciones entre componentes. Es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. El mismo está compuesto por: nodos, dispositivos y conectores (Acosta y otros, 2013). A continuación se presenta el diagrama de despliegue para la propuesta de solución.

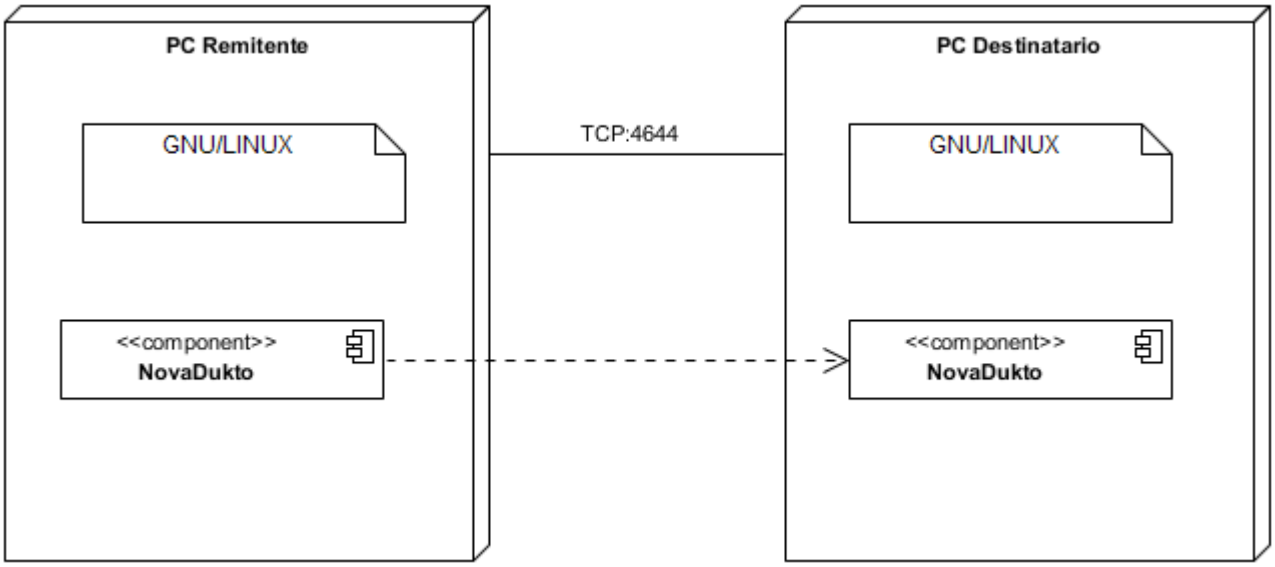


Figura 9. Diagrama de despliegue

(Fuente: elaboración propia)

Descripción de los nodos

PC_Remitente: Tendrá el NovaDukto para enviar los archivos deseados hacia la PC de destino.

PC_Destinatario: Tendrá el NovaDukto para recibir los archivos enviados por la otra PC.

NovaDukto: Herramienta para la transferencia de archivos.

TCP *(Transmission Control Protocol***,** Protocolo de Control de Transmisión): Protocolo de red que permite la transferencia de archivos entre las dos PC.

3.1.2. Estándares de implementación

Los estándares de implementación, son parte de las llamadas buenas prácticas, son un conjunto de convenciones establecidas de ante mano para la escritura de código. Estos estándares varían dependiendo del lenguaje de programación elegido (Merkury, 2017). Para la codificación de la propuesta de solución se utiliza el estándar de codificación para Python:

- **Indentación**: se utilizaron 5 espacios por cada nivel de indentación.
- **Tabuladoreso espacios**: no se mezclaron tabuladores y espacios en la codificación. Las formas más populares de indentar en Python son usados solo espacios o solo tabuladores, el código

indentado con una mezcla de tabuladores y espacios se reformateó y se usaron espacios exclusivamente.

- **Tamaño máximo de línea**: se limitaron todas las líneas a un máximo de 94 caracteres.
- **Convenciones de nombres** No se utilizaron los caracteres 'l' (letra ele minúscula), 'O' (letra o mayúscula) o 'I' (letra i mayúscula) como nombres de variables de un solo caracter debido a que en algunas fuentes, estos caracteres son indistinguibles de los numerales uno y cero.
- **Otros elementos que se tuvieron en cuenta**: .se rodearon siempre los siguientes operadores binarios con un espacio en cada lado: asignación (=), asignación aumentada (+=, -=, *=, /=), comparación (==, <, >, =, <>, <=, >=, in, not in, is, isnot), booleanos (and, or, not).
  Se utilizaron espacios alrededor de los operadores aritméticos.

A continuación, la figura muestra una demostración de la aplicación de los estándares de codificación.

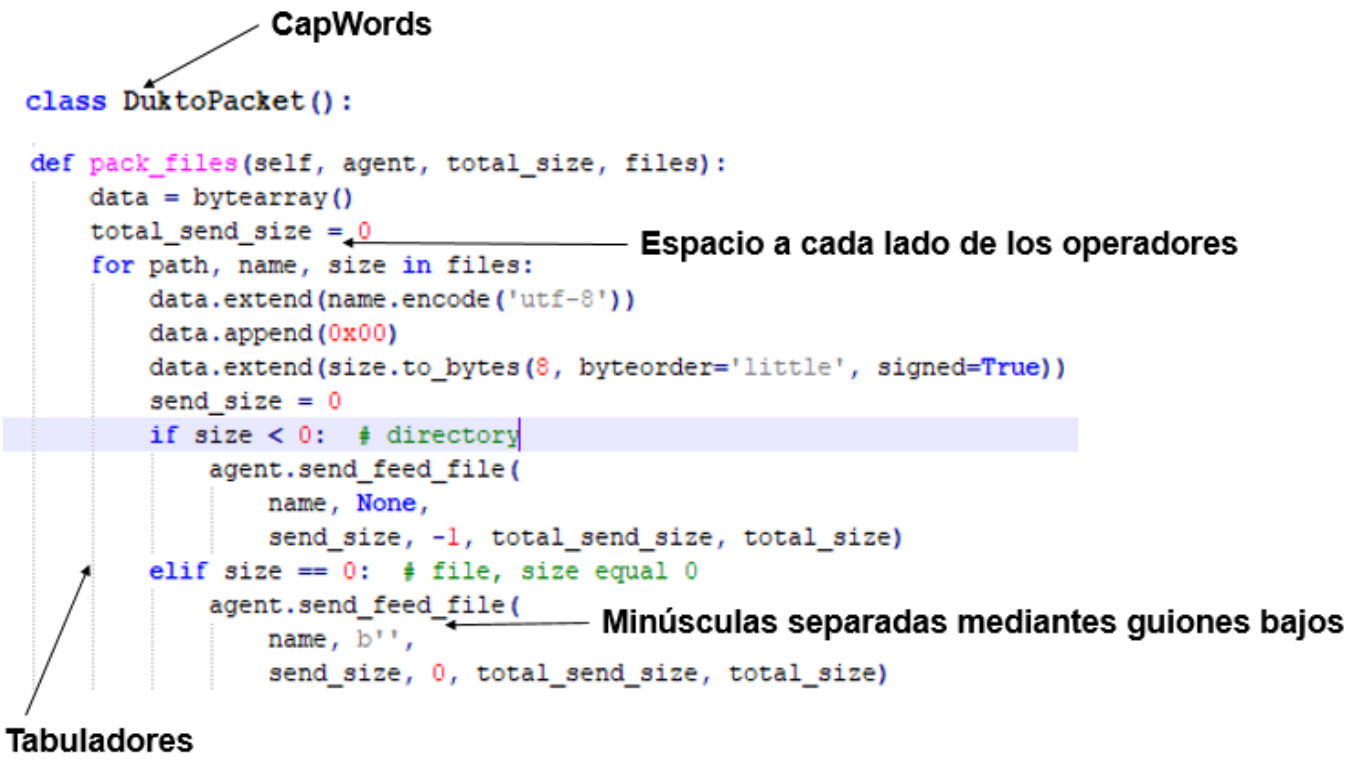


Figura 10. Aplicación de los estándares de codificación

(Fuente: elaboración propia)

3.1.3. Interfaz gráfica de usuario

La interfaz gráfica de usuario es esencial en cualquier aplicación, permitiendo representar las acciones y la información, usando un conjunto de objetos gráficos e imágenes. Tiene como función principal facilitar un entorno visual sencillo que permita la comunicación con el ordenador (Albornoz, y otros, 2017). Se aplicaron las siguientes reglas (Pressman, 2010) en el diseño de las interfaces gráficas de usuario de la propuesta de solución.

Reglas de oro:

- Dar control al usuario: Definir los modos de interacción de forma que el usuario no realice acciones innecesarias o indeseables, proporcionar una interfaz flexible, incluir opciones de interrumpir y deshacer la interacción del usuario, ocultar al usuario ocasional los elementos técnicos internos, diseñar interacción directa con los objetos que aparecen en pantalla.
- Reducir la carga en memoria del usuario: Reducir la demanda de memoria a corto plazo, definir valores por defecto que tengan significado, definir accesos directos intuitivos, el formato visual de la interfaz debe basarse en la metáfora tomada de la realidad, desglosar la información de manera progresiva.
- Lograr que la interfaz sea consistente: Permitir que el usuario incluya la tarea actual en el contexto que tenga algún significado.

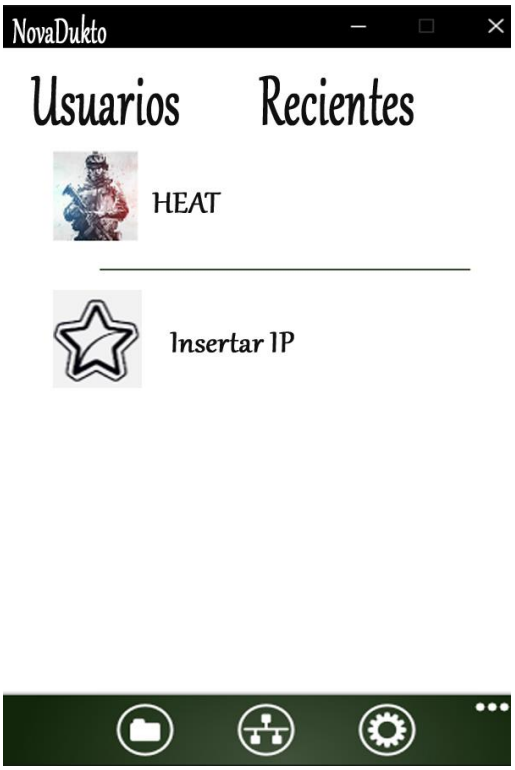


Figura 11. Interfaz de usuario

(Fuente: elaboración propia)

3.2 Pruebas de software

Las pruebas de software son un conjunto de actividades que se realizan con la finalidad de detectar errores y evaluar las características del software, regulan la ejecución de los proyectos, garantizando la calidad del producto desarrollado. Están conformadas por técnicas y métodos específicos del diseño de casos de prueba (Pressman, 2010). Consisten en la verificación dinámica del comportamiento de un programa en un conjunto finito de casos de prueba, adecuadamente seleccionado de los posibles escenarios del sistema, para asegurarse que arroja el resultado definido en la especificación de requisitos (Granda, 2013). Para obtener un producto de calidad, es necesario realizar un conjunto de pruebas al sistema con el objetivo de garantizar un correcto funcionamiento antes de entregar el software al cliente. Las pruebas tienen como objetivo identificar posibles fallos de funcionamiento, configuración o usabilidad (Ares, 2017).

3.2.1 Pruebas a realizar

Pruebas unitarias

Las pruebas unitarias se concentran en el esfuerzo de verificación de la unidad más pequeña del software, el componente o módulo de software. Tomando como guía de partida la descripción del diseño a nivel de componentes, se prueban importantes caminos de control para descubrir errores dentro de los límites del módulo. Centran su actividad en verificar la funcionalidad y la estructura (lógica interna) de cada elemento individualmente, una vez que ha sido codificado (Pressman, 2010).

Pruebas funcionales

Las pruebas funcionales se realizan de forma automatizadas o manuales que prueban las funcionalidades de la aplicación o módulo construidos desde el punto de vista del usuario final, con sus diferentes roles, para validar que el software hace lo que debe y, sobre todo, lo que se ha especificado. Se encargan principalmente de verificar que las funcionalidades desarrolladas en el sistema cumplan con sus especificaciones (Larrea, 2019).

3.2.2 Métodos de prueba

Caja negra este método se enfoca en el correcto manejo de funciones externas soportadas por el sistema sin tomar en cuenta la estructura interna del mismo. Se centra en buscar resultados no esperados, es decir que el sistema no responda a las especificaciones establecidas (Rodríguez, 2018).

Caja blanca este método verifica la correcta implementación de las pruebas internas y hace énfasis en la reducción de errores internos. (Hoogenraad, 2018).

3.2.3 Técnicas de prueba

Las técnicas de pruebas tienen el objetivo de identificar condiciones de las pruebas, casos de pruebas y datos de pruebas para que se tenga la mayor probabilidad de encontrar errores reduciendo tiempo y esfuerzo (Sánchez, 2015). Las técnicas que se utilizaron fueron partición de equivalencia para las pruebas funcionales y camino básico para las pruebas unitarias.

Partición de equivalencia esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software (Garrido, 2016). Se trata de los valores de entrada del

programa o del sistema que se dividen en grupos que tengan comportamiento similar, de manera que puedan ser procesados de la misma forma (Mendoza, 2019).

Camino básico consiste en verificar el código del sistema de manera que se compruebe que todo funciona correctamente, es decir, se verifica que todas las instrucciones del programa se ejecutan por lo menos una vez (Mause, 2017).

3.3. Aplicación de las pruebas de software

3.3.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando los productos de trabajo implementados, a través de instrumentos de prueba como: diseños de casos de prueba y listas de chequeo (Rodríguez, 2015).

Pruebas unitarias

Las pruebas unitarias se desarrollaron utilizando la técnica del camino básico del método de prueba caja blanca. En esta técnica se utilizó la métrica del software complejidad ciclomática que proporciona una medida cuantitativa de la complejidad lógica de un programa o procedimiento. Cuando se utiliza en el contexto de prueba el valor calculado mediante la complejidad ciclomática define el número de caminos independientes en el conjunto básico de un programa o procedimiento y proporciona un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2010). En la figura 9 se presenta el procedimiento para enviar texto del RF3.

```
1  def send_text(self, text):
       sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
2      if self._cert and self._key:
3          ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
           ssl_context.check_hostname = False
           ssl_context.verify_mode = ssl.CERT_NONE
           sock = ssl_context.wrap_socket(sock, server_side=False)
4      sock.connect(self._address)
       data = self._packet.pack_text(text)
       try:
           sock.sendall(data)
5      except KeyboardInterrupt:
           pass
       except Exception:
           pass
6      sock.close()
       self.send_finish()
```

Figura 12. Procedimiento para enviar texto del RF 3.

(Fuente: elaboración propia)

1. Dibujar el grafo de flujo de la funcionalidad o procedimiento a analizar

En la figura 10 se utilizó la notación de grafo de flujo para representar en este el código del procedimiento a probar.

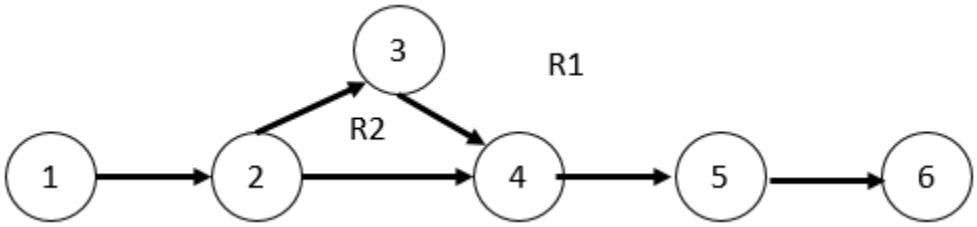


Figura 13. Grafo de flujo

(Fuente: elaboración propia)

2. Determinar la complejidad ciclomática del grafo

La complejidad ciclomática del grafo V (G) se puede calcular de las tres formas siguientes:

V(G) = A - N + 2

V(G) = 6 – 6 + 2 = 2.

Dónde: A es el número de aristas del grafo de flujo y N es el número de nodos del grafo.

V(G) = P + 1 V(G) = 1 + 1 = 2	Dónde: P es el número de nodos predicados (nodos con más de una arista de salida) contenidos en el grafo.
V(G) = R V(G) = 2	Dónde: R son las regiones, áreas delimitadas por nodos y aristas en el grafo.

3. Determinar los caminos linealmente independientes

- Camino básico 1: 1, 2, 4, 5, 6
- Camino básico 2: 1, 2, 3, 4, 5, 6

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

En el diseño de los casos de prueba se debe especificar los siguientes elementos:

- ✓ **Descripción:** contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.
- ✓ **Condición de ejecución:** se especifican los parámetros que debe poseer el caso de prueba para que se cumpla una condición deseada como respuesta del funcionamiento del procedimiento.
- ✓ **Entrada:** se muestran los parámetros de entrada al procedimiento.
- ✓ **Resultados esperados:** se explica el resultado esperado de la ejecución del procedimiento.

En la siguiente tabla muestra el diseño de caso de prueba para el camino 1 del conjunto básico de caminos linealmente independientes, correspondiente a la funcionalidad Enviar texto.

Tabla 6. Listado Caso de prueba para el camino 1 del procedimiento Enviar texto

(Fuente: elaboración propia)

Diseño de caso de pruebas para el camino 1	
Descripción	El texto no puede ser nulo.
Condición de ejecución	La dirección de IP a la que se desea enviar el texto tiene que ser válida y encontrarse en la misma red LAN.
Entrada	Self : text

Resultado esperado	Se envía el texto de forma satisfactoria.

Pruebas funcionales

Se realizaron pruebas funcionales, a través del método de caja negra y la técnica de partición de equivalencia. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada.

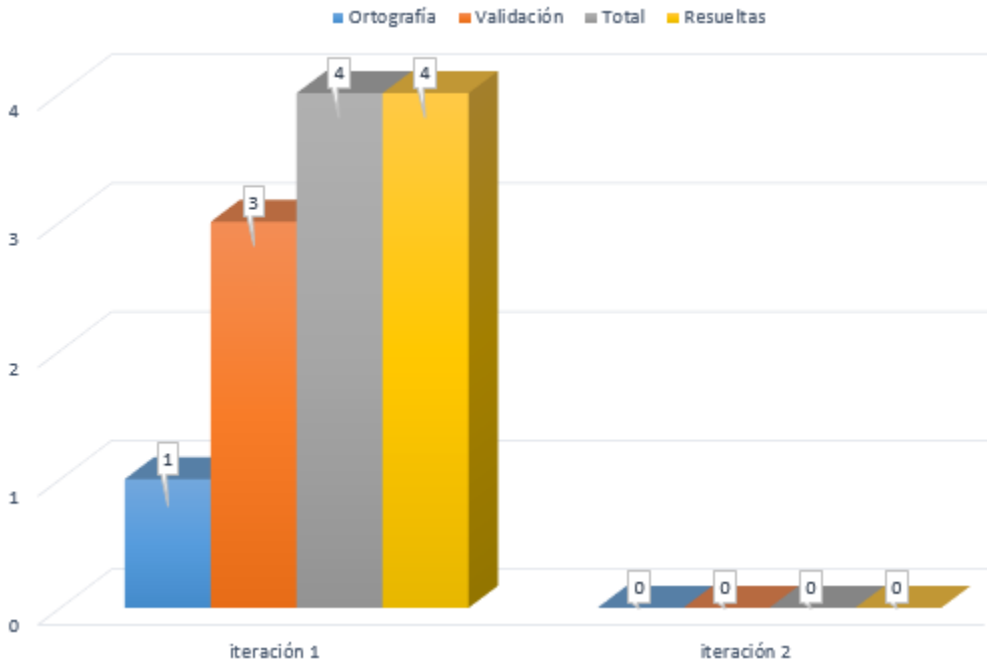


Figura 14. Resultado de las pruebas funcionales

(Fuente: elaboración propia)

Las pruebas funcionales se realizaron en dos iteraciones, se encontraron 4 no conformidades en la primera iteración, de ellas 3 de validación y 1 de ortografía, en la segunda iteración no se encontraron no conformidades.

3.3.2 Pruebas de aceptación

Es la prueba final que se realiza antes del despliegue del sistema. Su principal objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Sánchez, 2015). El cliente realizó pruebas funcionales a través

del método de caja negra y la técnica de partición de equivalencia, en las cuales no se detectaron no conformidades. A continuación, la tabla 7 muestra el Diseño de Caso de Prueba del requisito funcional Insertar IP.

Tabla 7. RF 4. Caso de prueba del requisito funcional Insertar IP.

(Fuente: elaboración propia)

Escenarios	**Descripción**	**Dirección IP**	**Respuesta del sistema**	**Flujo central**
EC.1.1Insertar correctamente la dirección ip	El sistema permite seleccionar la opción deseada, como transferir archivo o enviar texto	(Válido) 10.28.27.21	El sistema activa las opciones que puede seleccionar el usuario	1. El usuario ingresa la dirección de forma correcta.
EC1.2 Insertar dirección invalida	El sistema no permite seleccionar la opción deseada, como transferir archivo o enviar texto	(Inválido) 256.0.0.0	El sistema no activa las opciones que pueda seleccionar el usuario	1. El usuario ingresa la dirección de forma incorrecta.
EC1.3 No insertar dirección IP	El sistema no permite seleccionar la opción deseada, como transferir archivo o enviar texto	(Inválido) –·–·–·–	El sistema no activa las opciones que pueda seleccionar el usuario	1. El usuario no ingresa la dirección.

3.4 Evaluación del objetivo de la investigación

Siempre que se implementa una propuesta de un producto, 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 (Silega, 2014).

Iadov es una técnica eficaz 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 la validación por el usuario de indicadores definidos, mediante la respuesta a cinco preguntas, de ella tres cerradas (1, 3 y 5) y dos abiertas (2 y 4), cuya relación ignora el sujeto. Estas tres preguntas cerradas se relacionan a través del “Cuadro lógico de Iadov”, el cual se presenta adaptado a la presente investigación en la Tabla 8.

Para aplicar esta técnica se usa el "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. (Ruiz, 2016).

Tabla 88. Cuadro Lógico de Iadov

(Fuente: elaboración propia)

5. Luego de conocer la herramienta para la transferencia de archivos en Nova, refleje en qué medida le gusta la solución implementada.	1. ¿La forma en que se realizan las trasferencias de los archivos permite satisfacer todas las necesidades existentes?								
	No			**No sé**			**Sí**		
	3. ¿Considera usted que sería factible contar con una herramienta que mejora la transferencia de archivos en Nova?								
	Sí	**No sé**	**No**	**Sí**	**No sé**	**No**	**Sí**	**No sé**	**No**
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	3	3	6	3	6
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	6	5
No sé decir	2	3	6	3	3	3	6	6	4

El número resultante de la interrelación de las tres preguntas nos indica la posición de cada sujeto en la escala de satisfacción, o sea su satisfacción individual. La escala de satisfacción utilizada es la siguiente:

1. Clara satisfacción
2. Más satisfecho que insatisfecho
3. No definida
4. Más insatisfecho que satisfecho
5. Clara insatisfacción
6. Contradictoria

Esta técnica también permite obtener el índice de satisfacción grupal (ISG), para lo cual se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y - 1 de la siguiente forma:

$$I = \frac{a(+1)+b(+0,5)+c(0)+d(-0,5)+e(-1)}{N}$$

En esta fórmula a, b, c, d y e representan el número de sujetos en indicar 1, 2, 3, 4, 5 y 6, y N el número de sujetos.

El valor del ISG permite identificar las siguientes categorías grupales:

- Máxima insatisfacción: desde -1 hasta -0,49
- Más insatisfecho que satisfecho: desde -0,5 hasta -0,1
- No definido y contradictorio: 0
- Más satisfecho que insatisfecho: desde 0,1 hasta 0,49
- Máximo de satisfacción: desde 0,5 hasta 1

El índice general arroja valores entre + 1 y - 1. Los valores que se encuentran comprendidos entre -1 y -0,5 indican insatisfacción; los comprendidos entre - 0,49 y + 0,49 evidencian contradicción y los que están entre 0,5 y 1 indican que existe satisfacción.

Resultados obtenidos

1. Los resultados obtenidos de la aplicación de la encuesta se presentan en la Tabla 9.

Tabla 9. Resultado de la escala de satisfacción

(Fuente: elaboración propia)

Categorías grupales de satisfacción	N=7	Escala
Clara satisfacción	6	a
Más insatisfecho que satisfecho	1	b
No definido	0	c

Más satisfecho que insatisfecho	0	d
Máximo de satisfacción	0	e
Contradictorio	0	c

2. Cálculo del Índice de Satisfacción Grupal

ISG=a(+1)+b(+0.5)/N

ISG=(6(+1)+1(+0.5))/7 = 0.92

3. Interpretación del resultado del ISG.

Se obtuvo como resultado del ISG el valor 0.92 lo que indica máxima satisfacción de los usuarios con respecto a la herramienta para la transferencia de archivo en Nova. Se puede afirmar que se cumplió el objetivo de la investigación.

Conclusiones del capítulo

En el desarrollo del capítulo se transitó por las disciplinas, Implementación, Pruebas internas y Pruebas de aceptación. La implementación arrojó como resultado una herramienta que permite mejorar el proceso de transferencia de archivos en Nova. El producto desarrollado cumple con los estándares de codificación definidos. Las pruebas unitarias comprobaron que el flujo de trabajo de las funcionalidades es correcto y las pruebas funcionales contribuyeron a identificar y corregir 4 no conformidades. Se evidenció la satisfacción del cliente a través de las pruebas de aceptación y la aplicación de la técnica de Iadov la cual propició la evaluación satisfactoria de la herramienta.

Conclusiones

La investigación realizada cumple con los objetivos planteados mediante el desarrollo de la herramienta nativa para la transferencia de archivo en Nova y se arriba a las siguientes conclusiones:

- El análisis de los referentes teóricos y de los sistemas informáticos existentes estudiados demostró la necesidad de desarrollar una herramienta para mejorar el proceso de transferencia de archivo en Nova.
- Se obtuvo una herramienta nativa que permite la transferencia de archivos en Nova cumpliendo con los requisitos definidos por el cliente.
- La herramienta para la transferencia de archivos en la Distribución Cubana de GNU/Linux Nova, es un resultado factible para los usuarios de este sistema operativo, tal como evidencia la aplicación de técnicas y pruebas que garantizan el correcto funcionamiento de la aplicación, y demostraron la satisfacción del cliente hacia el sistema desarrollado.

Recomendaciones

Para futuras investigaciones relacionadas con la transferencia de archivo en la Distribución Cubana GNU/Linux Nova se recomienda lo siguiente:

- Se recomienda incluir a la herramienta nativa para la transferencia de archivo para otras variantes, versiones y arquitecturas de Nova.
- Establecer acciones de soporte a la herramienta para la transferencia de archivo en la Distribución Cubana de GNU/Linux Nova y toda la infraestructura de la cual es dependiente, en caso de modificación en cualquiera de ellos, migración o incorporación de elementos tecnológicos

Referencias bibliográficas

Acosta, Tomás y Hernández Rodríguez, Nodelvis. 2013. Módulo de reportes webmétricos para el motor de búsqueda Orión. Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas. [En línea] 2013. [Citado el: 22 de octubre de 2021.] https://repositorio_institucional.uci.cu/jspui/handle/ident/8296.

Albornoz, María Claudia, Berón, Mario y Montejano, Germán Antonio. 2017. Repositorio institucional de la UNLP. [En línea] abril de 2017. [Citado el: 16 de noviembre de 2021] http://sedici.unlp.edu.ar/handle/10915/62078.

Alcantud, M. 1999. Teleformación. In Universidad de Valencia.

Ares, Laura. 2017. visual-engin. [En línea] 2017. [Citado el: 16 de marzo de 2020.] http://visual-engin.com/2017/10/26/importancia-pruebas-de-software-testing/.

Ayuntamiento de Gijón,2011. Nuevos mecanismos de transferencia de tecnología. Disponible en: Transferencia de tecnología (Formato pdf). Consultado: 10 noviembre del 2021.

Larrea, Natalia. 2019. Análisis de la aplicación de pruebas funcionales y pruebas de usabilidad de software en el desarrollo de sistemas web. 3.4, s.l. : Ciencia Digital, 2019, Vol. 3.

Andalucia, Junta de. 2017. juntadeandalucia.es. [En línea] 2017. [Citado el: 22 de mayo de 2021.] ww.juntadeandalucia.es/servicios/madeja/contenido/recurso/407.

Aviztar. 2017. Curso práctico de Modelado de Negocios con BPMN y UML. [En línea] 2017. [Citado el: 22 de mayo de 2021] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.

Bembibre, Victoria .Sitio: Definición ABC. Título: Transferenci. [Citado el: 1 de diciembre de 2021.] | URL: https://www.definicionabc.com/tecnologia/transferencia.php

Blancarte, Oscar. 2017. oscarblancarteblog. [En línea] 16 de octubre de 2017. [Citado el: 22 de mayo de 2021.] https://www.oscarblancarteblog.com/2017/10/26/ide.

Cardoso, Dayana y González Iznaga, Jorge Luis. 2014. Componente para la Digitalización de Contornos de Estructuras Geológicas. [En línea] 2014. [Citado el: 25 de octubre de 2021.] https://repositorio_institucional.uci.cu/jspui/handle/ident/9119.

Chacon, Scott. 2015. Pro Git Book. 2015. [En línea] 2015. [Citado el: 16 de junio de 2021] https://git-scm.com/book/es/v2

CNET. 2019. Visual paradigm for UML - free download and software reviews. Visual Paradigm for UML - Free Download and Software Reviews CNET,. http://download.cnet.com/%0AVisual-Paradigm-for-UML/3000-2247_4-42700.html

Pierra, Allan, 2010. Nova, distribución cubana de gnu/linux. reestructuración estatégica de su proceso de desarrollo. [En línea] 2017. [Citado el: 16 de junio de 2021. https://www.uci.cu/universidad/claustro/allan-pierra-fuentes

E. Gamma, R. Helm, R. Johnson, y J. V. 1995. Design Patterns Elements of Reusable Object-Oriented Software.

Ferrer, Lautaro. 2016. Propuesta de conceptualización de requisitos para proyectos software basados en formalismo de ingeniería de conocimiento. [En línea] 2016. [Citado el: 26 de septiembre de 2021.] http://revistas.unla.edu.ar /software/article/view/1279/1114.

Folgueiras, P. 2016 Técnica de recogida de información: La entrevista. Garrido Tejero, A. 2016. riunet.upv. [En línea] 2016. [Citado el: 27 de noviembre de 2021.] https://riunet.upv.es/handle/10251/63579.

Granda, Ailec. 2013. Modelo didáctico para el uso de comunidades virtuales en el proceso de enseñanza aprendizaje de la Disciplina Ingeniería y Gestión de Software en la Universidad de las Ciencias Informáticas. Tesis para obtener el grado de doctora en tecnología educativa. [En línea] 2013. [Citado el: 3 de noviembre de 2021.] http://www.tesisenred.net/bitstream/handle/10803/127226/tagd1de1.pdf?sequence=1.

Gómez, Maria del Carmen. 2013. Notas del curso Base de Datos. MÉXICO, D.F : Casa abierta al tiempo, 2013.

Gracia, Joaquín. 2019. Patrones de diseño: análisis y diseño. Ingeniería del software.

Guerra, César Arturo. 2017. SG Buzz. Obtención de Requerimientos. Técnicas y Estrategia. [En línea] 2017. [Citado el: 26 de septiembre de 2021.] https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

Guzmán, E, & Máter, A. 2018. SISTEMAS OPERATIVOS Concepto y definición de Sistemas Operativos ,

historia de los Sistemas Operativos , características de los Sistemas Operativos , clasificación de los sistemas operativos , Aplicaciones . De la Cruz Barboza , Lisdney Concepto y definic.

Hoogenraad, Wim. 2018. es.itpedia. [En línea] febrero de 2018. [Citado el: 17 de noviembe de 2021.] https://es.itpedia.nl/2018/02/05/white-box-testing-onder-de-loep/.

José, F., & Peñalvo, G. 2018. Ingeniería de software i. Departamento de Informática y Automática Universidad de Salamanca

Graig Larman, 1999. UML y Patrones, introducción al análisis y diseño orientado a objetos. México: Prentice Hall, 1999].

Len Bass, Paul Clements, y Rick Kazman. 2019. Software architecture in practice. SEI series in software engineering, (2nd edición).

lenguajesdeprogramacion.net. 2018. lenguajesdeprogramacion.net. [En línea] 2018. [Citado el: 20 de junio de 2021.] https://lenguajesdeprogramacion.net/python/.

Linux-es. 2015 Sobre Linux | El rincón de Linux. [En línea] 2015. [Citado el: 28 de mayo de 2021.] http://www.linux-es.org/sobre_linux.

Mause, JC. 2017. c-mouse. [En línea] octubre de 2017. [Citado el: 18 de marzo de 2020.] http://www.jc-mouse.net/ingenieria-de-sistemas/caja-blanca-prueba-del-camino-basico.

Mendoza, Jhonny. 2019. Análisis del sistema informático de la unidad de tránsito en el municipio del cantón Las Naves. s.l. : Tesis de licenciatura, 2019.

Merkury. 2017. ohmyroot. [En línea] 12 de enero de 2017. [Citado el: 12 de noviembre de 2021.] https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/.

Montero, Bryan. 2018. Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. Espirales revista multidisciplinaria de investigación., 2018, Vol. 2.

Naranjo, David. 2019. desdelinux. [En línea] 2017. [Citado el: 10 de octubre de 2021.] https://blog.desdelinux.net/pycharm-un-entorno-de-desarrollo-para-python/.

Peréz, S.L Peréz. 2013. Fundamentos de Ingeniería de Software. Patrones de diseño. Universidad Autónoma de México. [En línea] 2013. [Citado el: 5 de noviembre de 2021.] http://computacion.cs.cinvestav.mx/~sperez/cursos/fis/PatronesDiseno.pdf.

Pressman, R. S. 2010. Software engineering: a practitioner's approach. 7th ed. New York : EUA, 2010. ISBN:978-0-07-337597.

Python-The Fastest Growing Programming Language. 2017. 12, s.l. : International Research Journal of Engineering and Technology (INJERT), 2017, Vol. 4.

Pythonízame. 2016. PyCharm Community Edition. [En línea] 2016. [Citado el: 7 de junio de 2020.] http://pythoniza.me/pycharm-community-edition. .

Raffino, María Estela. 2020. Archivo - Concepto, características y archivo informático. [En línea] 2019. [Citado el: 28 de septiembre de 2021.] https://concepto.de/archivo/#ixzz6zQlWSUJH

Reynoso, Carlos. 2021. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Universidad de Buenos Aires

Rouse. 2018. searchSoftwareQuality. [En línea] 6 de septiembre de 2018. [Citado el: 23 de junio de 2021.] https://www.searchsoftwarequality.techtarget.com.

Ruiz, María de los Ángeles. 2016. MOPIGD: Modelo para la implementación de la gestión de documentos en el sistema empresarial cubano. [En línea] 2016. [Citado el: 12 de nobiembre de 2021.] http://www.congreso-info.cu/index.php/info/2016/paper/viewFile/287/40.

Sánchez,Tamara Rodríguez. 2015. Metodología de desarrollo para la actividad productiva de la UCI. Universidad de las Ciencias Informáticas. La Habana. Cuba : s.n., 2015.

Software, D. E. 2018. Modelo de dominio. Departamento de Informática y Automática Universidad de Salamanca

Silega, Nemury. 2014. Método para la trasformación automatizada del modelo de procesos de negocio a modelo de componentes para sistemas de gestión empresarial. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas. [En línea] 2014. [Citado el: 20 de nobiembre de 2021.] https://repositorio_institucional.uci.cu/jspui/handle/ident/8584.

Steve Burbeck. 1992. Applications programming in smalltalk- 80 (tm): How to use modelview-controller

(mvc).

Tarancón, Y. Y. 2015. Xabal Excriba. Gestor de Documentos Administrativos. Citado el: 23 de junio de 2021.] http://excriba.prod.uci.cu/proxy/alfresco/api/node/content/workspace/SpacesStore/a622adab-eac5-4fb3-ba08-a266767fff5f/Metodologia UCI.pdf?a=true

Xiao He, Zhiyi Ma, Weizhong Shao, y G. L. 2017. A metamodel for the notation of graphical modeling languages. In IEEE edición (Vol. 1).

Anexos

Anexo I:

Anexo I: Entrevista realizada al jefe de proyecto de Nova en el centro CESOL.

1. ¿Cree usted que es importante contar con una aplicación nativa de Nova para la transferencia de archivos? ¿Por qué?
2. ¿Cuáles son los aspectos fundamentales a tener en cuenta actualmente para realizar transferencia de archivos?
3. ¿Cuáles son los pasos a seguir para realizar una transferencia de archivo? ¿Qué vía usan actualmente para transferir archivos?
4. ¿Cree necesario la creación de una herramienta que permita realizar transferencias de archivos nativa de Nova? ¿Por qué?
5. ¿Cuáles serían las características que tendría esta herramienta?

Anexo II: Guía de observación para el proceso de transferencia de datos de Nova.

Observador: Leonardo Mastrapa Reynaldo

Lugar: Laboratorio del proyecto Nova

Objetivo: Identificar los requisitos fundamentales para la realización del proceso de transferencia de archivos Nova.

1- Datos de identificación del proceso
    - Nombre del proceso
    - Especialista que ejecuta el proceso
2- Características del espacio donde se desarrolla el proceso
    - ¿Dónde se debe realizar?
    - ¿Bajo qué condiciones se debe realizar?
    - ¿Qué se necesita para iniciar el proceso?
3- Características del proceso.
    - ¿Cómo es el proceso?
    - ¿Qué herramientas se emplean?
    - ¿Cuáles son los pasos a seguir?
    - ¿Cuál es el resultado?