UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD # 1

Título: Módulo para la configuración del sistema de archivos para NovaRSAT.

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

Autora:

Neysa de la Caridad Martínez Acosta

Tutores:

Ing. Lexys Manuel Díaz Alonso

Ing. Hanny Valdés Hernández

La Habana, diciembre 2021

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Neysa de la Caridad Martínez Acosta, soy la autora del Trabajo de Diploma Módulo configuración del sistema de archivos para NovaRSAT y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración de autoría en La Habana a los del mes del año.

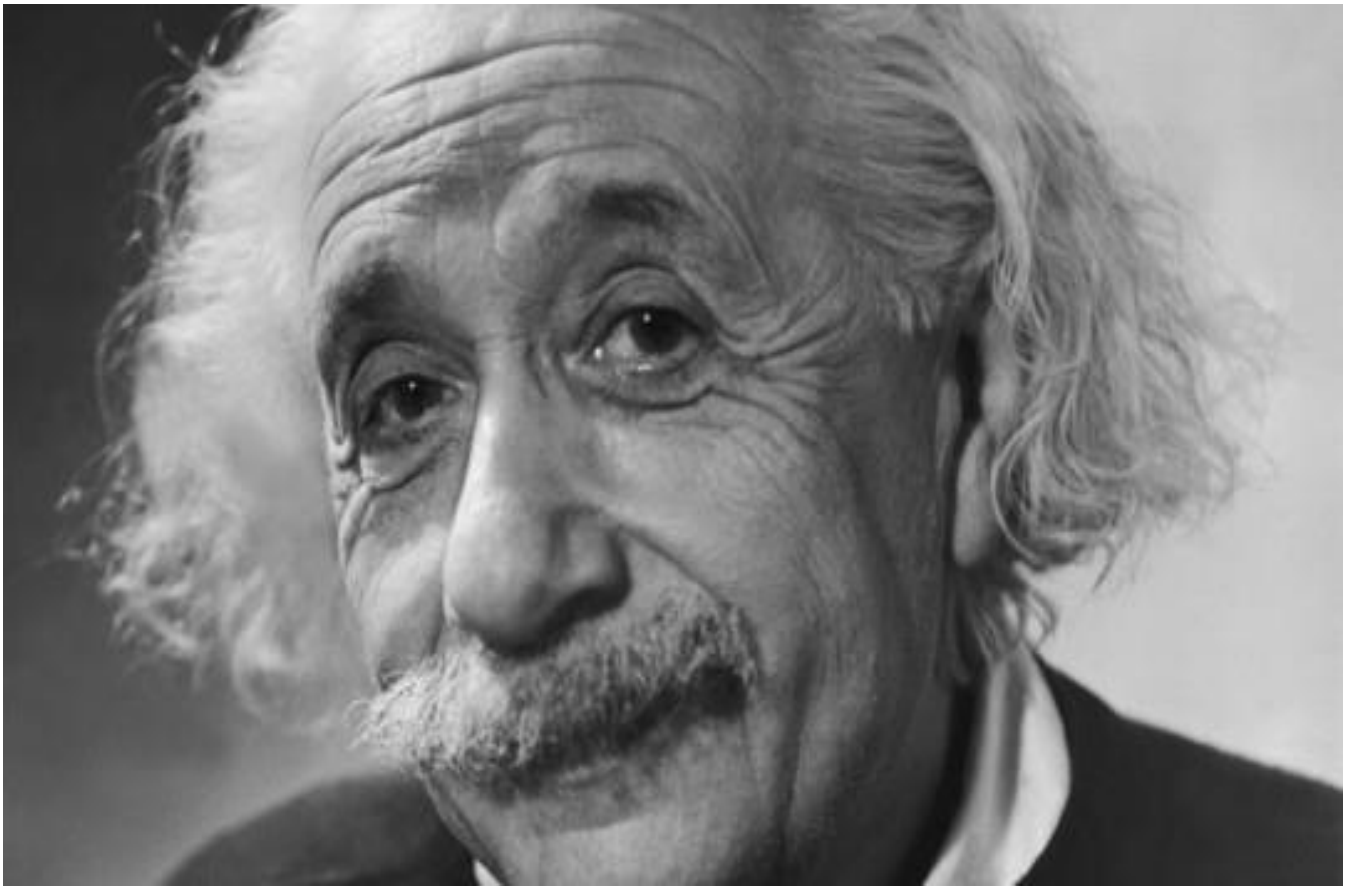
_____

Neysa de la Caridad Martínez Acosta

Autor

_____ _____

Ing. Lexys Manuel Díaz Alonso **Ing. Hanny Valdés Hernández**



"Nunca consideres el estudio como una obligación sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

DATOS DE CONTACTO

Autor:

Neysa de la Caridad Martínez Acosta

Universidad de las Ciencias Informáticas (UCI)

e-mail: ncmartinez@estudiantes.uci.cu

Tutores:

Ing. Lexys Manuel Díaz Alonso

Universidad de las Ciencias Informáticas (UCI)

e-mail: lmdiaz@uci.cu

Ing. Hanny Valdés Hernández

Universidad de las Ciencias Informáticas (UCI)

e-mail: (hvaldes@uci.cu)

DEDICATORIA

*A mi **madre** por ser la persona que siempre me ha apoyado y se ha sacrificado en toda su vida para que hoy pudiera ser la persona que soy.*

*A mi **papá** que aunque no esté presente donde quiera que se encuentre se que cumplí su sueño.*

*A mis **abuelos** que siempre me bridaron su apoyo a lo largo de toda mi carreta para cumplir todas mis metas.*

*A mi **familia** en general por siempre brindarme su apoyo incondicional.*

A ustedes va dedicado con todo el cariño y el amor del mundo.

AGRADECIMIENTOS

Durante estos años de la carrera para convertirme en Ingeniera en Ciencias Informáticas muchas personas han contribuido en la formación de la persona que soy hoy por eso quiero agradecer:

A mi madre por ser la persona que siempre estuvo ahí cuando más lo necesite y dar sus fuerzas y aliento para que yo saliera adelante para cumplir nuestro sueño.

A mis abuelos por ser dos personitas que a pesar de su edad lucharon incansablemente para que yo pudiera salir adelante.

A mi novio por ser la persona que estuvo la mayor parte del tiempo brindándome su apoyo a pesar de tanto estrés y que supo como alentarme en los momentos difíciles.

A mis amigos en especial a mi Samy y Rache por esos momentos de alegría y tristeza que compartimos. Por esas noches en vela estudiando y por siempre estar presente para mí a pesar de todas las dificultades. Les agradezco por ser esas personitas especiales que nunca olvidare a pesar del paso de los años. A todas las personas que me ayudaron a lo largo de mi carrera y aportaron su granito de arena .A aquellas amistades que hoy no estamos juntos pero que dejaron huellas en mi corazón

A mis tutores quienes me apoyaron y brindaron su experiencia para que el resultado de mi tesis fuera el mejor posible.

A cada uno de los profesores que he tenido quienes de ellos he aprendido mas allá de sus conocimientos sus valores . A la UCI y a la revolución cubana por haberme regalado esta posibilidad de superación de forma gratuita.

RESUMEN

En Cuba desde hace varios años se encuentra inmerso en el proceso de migración a software libre. La Universidad de las Ciencias Informáticas(UCI) es una de las instituciones que impulsan este proceso. En ellas se encuentra el proyecto de Servicios Integrales de Migración, Asesoría y Soporte (SIMAYS) donde actualmente se desarrolla una herramienta que permite administrar los servicios telemáticos de forma remota (NovaRSAT).

Actualmente la configuración y administración de la red en modo de producción se realiza de forma manual a través de una interfaz de consola lo que trae consigo una pérdida de tiempo en el despliegue del servicio. Para ello se analizaron herramientas utilizadas actualmente para gestionar el sistema de archivos en las diferentes tecnologías y que pudiesen ser integradas a NovaRSAT en la implementación de un módulo. Debido a esta problemática se realizó la presente investigación con el objetivo de desarrollo una aplicación desktop que permita administrar el servicio de red en los servidores en las instituciones cubanas. Para ello se analizaron herramientas utilizadas actualmente para gestionar el sistema de archivos en las diferentes tecnologías y que pudieses ser integradas en la implementación del un módulo a NovaRSAT. En la presente investigación se exponen las tecnologías, herramientas, lenguajes a utilizar en la construcción del módulo y la definición de los elementos conceptuales necesarios para la mejor construcción del mismo. Para regir el desarrollo de la solución se utilizó la metodología AUP-UCI, adaptada a los procesos productivos de la universidad. A dicho módulo se le fueron realizadas diferentes pruebas de software para verificar su calidad y correcto funcionamiento. Como resultado de la investigación realizada se obtuvo un módulo para la configuración del sistema de archivos para NovaRSAT.

Palabras Clave*:* configuración, sistema de archivos*, NovaRSAT.*

ABSTRACT

In Cuba for several years he has been immersed in the process of migration to free software. The University of Informatics Sciences (UCI) is one of the institutions that promote this process. They include the Comprehensive Migration, Advisory and Support Services project (SIMAYS) where a tool is currently being developed that allows managing telematic services remotely (NovaRSAT).

Currently the configuration and administration of the network in production mode is done manually through a console interface, which entails a loss of time in the deployment of the service. For this, tools currently used to manage the file system in the different technologies were analyzed and that could be integrated into NovaRSAT in the implementation of a module. Due to this problem, the present investigation was carried out with the objective of developing a desktop application that allows managing the network service on the servers in Cuban institutions. Due to this problem, the present investigation was carried out with the objective of developing a desktop application that allows managing the network service on the servers in Cuban institutions. For this, tools currently used to manage the file system in the different technologies were analyzed and that could be integrated in the implementation of a module to NovaRSAT. In this research, the technologies, tools, languages to be used in the construction of the module and the definition of the conceptual elements necessary for the best construction of the module are exposed. To govern the development of the solution, the AUP-UCI methodology was used, adapted to the productive processes of the university. Different software tests were carried out on said module to verify its quality and correct operation. As a result of the research carried out, a module for the configuration of the file system for NovaRSAT was obtained.

Key words: setting, file system, NovaRSAT.

ÍNDICE DE CONTENIDO

INDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

A lo largo de los años los sistemas de archivos surgieron de la necesidad de que varias aplicaciones compartieran el mismo medio de almacenamiento. En la práctica, un sistema de archivos también puede ser utilizado para acceder a datos generados dinámicamente, como los recibidos a través de una conexión de red de computadoras (sin la intervención de un dispositivo de almacenamiento).Los sistemas de archivos proveen métodos para crear, mover, renombrar y eliminar tanto archivos como directorios, pero carecen de métodos para crear, por ejemplo, enlaces adicionales a un directorio o archivo (enlace duro en Unix) o renombrar enlaces padres (".." en Unix).

La mayoría de los sistemas operativos manejan su propio sistema de archivos. A diferencia de Windows, Linux se compone de un sistema de archivos completamente diferente y ajeno a este, aquí no existen letras en las unidades tales como “C:\. D. Este sistema define la estructura de los sistemas de archivos en Linux y otros sistemas operativos UNIX. Sin embargo, el sistema de archivos de Linux también contiene algunos directorios, el cual hasta ahora no han sido definidos de la siguiente manera.(Ebrahim,2017)

GNU/Linux es el sistema operativo que soporta más sistemas de organización que lo convierte en uno de los más versátiles. Además, Linux, implementado en su kernel, admite la administración de manera transparente al usuario de más de 15 tipos diferentes de sistemas de archivos, incluyendo New Technology File System (NTFS) (Microsoft Windows), iso9660, msdos y vfat. La estructura de archivos es una estructura jerárquica en forma de árbol invertido, donde el directorio principal (raíz) es el directorio "/", del que cuelga toda la estructura del sistema. Este sistema de archivos permite al usuario crear, borrar y acceder a los archivos sin necesidad de saber el lugar exacto en el que se encuentran. No existen unidades físicas, sino archivos que hacen referencia a ellas. Consta de tres partes importantes, superbloque, tabla de i-nodos y bloques de datos.

El Sistema de Archivos de Linux o cualquier sistema de archivos, generalmente es una capa bajo el sistema operativo la cual maneja el posicionamiento de tus datos en el almacenamiento, sin este el sistema no puede saber dónde empieza y termina un archivo (Ebrahim, 2017).

En el escenario actual los sistemas GNU/Linux están en constante evolución, todos los días se actualizan, tanto el sistema operativo como el software instalado, pero en ningún momento se exige pagar por actualizaciones ni el software se queda obsoleto y deja de funcionar. En tal sentido la migración al software libre constituye una necesidad para el desarrollo para garantizar la

independencia tecnológica. El avance de la ciencia y la tecnología aplicadas en favor de la sociedad contribuye al desarrollo sostenible de un país.

Desde hace varios años, Cuba está inmersa en una migración al uso del software libre, una tarea que se está realizando en varias instituciones del país. La Universidad de las Ciencias Informáticas (UCI) es una de las instituciones encargadas de impulsar este proceso. Dentro de sus centros productivos está el Centro de Software Libres (CESOL) que tiene como objetivo principal la investigación y el desarrollo de aplicaciones informáticas para satisfacer las necesidades de la migración a plataformas de código abierto en el país.

Uno de los productos de software que se desarrollan en CESOL es Nova Servidores, una variante de la distribución cubana GNU/Linux Nova para la administración de los servicios telemáticos. Esta solución informática se ha sido desplegada por especialistas de CESOL en diferentes Organismos de la Administración Central del Estado (OACE) para mejorar la infraestructura informática de estas entidades y garantizar la soberanía tecnológica.

En una entrevista aplicada a 5 especialistas (Ver anexo 1) de CESOL que han participado en los procesos de migración a software libre de los OACE y desplegado Nova Servidores se pudo conocer que la configuración y administración del sistema de archivos en esta solución informática se realiza de forma manual a través de una interfaz de consola lo que trae como consecuencia:

- Incurrencia en errores humanos debido a que los administradores de red necesitan conocer los comandos necesarios para realizar las configuraciones del sistema de archivos, así como, los ficheros que son necesarios modificar y su ubicación en Nova Servidores.
- Pérdida de tiempo en el despliegue y explotación de Nova Servidores como consecuencias de las demoras en su configuración manual.

Para darle respuestas a los problemas descritos, en CESOL se desarrolla una herramienta que permite administrar los servicios telemáticos de forma remota (NovaRSAT). Uno de los módulos a incluir en este producto es el responsable de la configuración del sistema de archivos en Nova Servidores.

Dada la situación anterior se define como **problema científico**:

¿Cómo facilitarla configuración del sistema de archivos en Nova Servidores de forma remota?

Se determina como **objeto de estudio** el proceso de configuración del sistema de archivos en sistemas operativos; delimitando como **campo de acción** herramientas informáticas para la configuración del sistema de archivos en sistemas operativos.

Para dar solución al problema científico identificado se determina como **objetivo general de la investigación:** Desarrollar un módulo para NovaRSAT que permita la configuración del sistema de archivos en Nova Servidores de forma remota.

Se determinó como **objetivos específicos**:

1. Elaborar el marco teórico de la investigación sobre el proceso de configuración del sistema de archivos en sistemas operativos.
2. Diseñar un módulo para NovaRSAT que permita la configuración del sistema de archivos en Nova Servidores de forma remota.
3. Implementar un módulo para NovaRSAT que permita la configuración del sistema de archivos en Nova Servidores de forma remota.
4. Evaluar el módulo para NovaRSAT que permita la configuración del sistema de archivos en Nova Servidores de forma remota.

Se definen como preguntas científicas**:**

1. ¿Cuáles son los presupuestos teóricos que fundamentan el proceso de configuración del sistema de archivos en sistemas operativos?
2. ¿Qué elementos se deben tener en el diseño de un módulo para NovaRSAT que permita la configuración del sistema de archivos en Nova Servidores de forma remota?
3. ¿Qué componentes de software son necesarios implementar con el objetivo de obtener un módulo para NovaRSAT que permita la configuración del sistema de archivos en Nova Servidores de forma remota?
4. ¿Qué pruebas de software se deben aplicar en la evaluación de un módulo para NovaRSAT que permita la configuración del sistema de archivos en Nova Servidores de forma remota?

Para el desarrollo de las preguntas científicas se han combinado diferentes métodos teóricos y empíricos de la investigación en la búsqueda y procesamiento de la información, los cuales se describen a continuación.

Métodos teóricos:

- **Analítico – Sintético:** En cuanto se analizó el estudio de las tecnologías existentes para la gestión del sistema de archivos; este método ayudó a analizar cuál de dichas tecnologías era la más apropiada para el desarrollo del módulo para NovaRSAT. Este método fue utilizado durante el estudio y análisis de documentos, libros, artículos además de otras fuentes bibliográficas que permitieron realizar una amplia investigación sobre las tecnologías, herramientas, lenguajes y de los principales gestores de archivos para dar solución al problema planteado.
- **Modelación:** Permitió crear el proceso de diseño mediante la abstracción de sus elementos fundamentales utilizando un lenguaje de modelado y así desarrollar un modelo para la aplicación a desarrollar a partir de la situación problemática planteada

Método empírico:

- **Observación:** Fue utilizado para realizar un análisis de las diferentes soluciones que resuelven de forma parcial el problema de investigación planteado.
- **Entrevista:** Se le realizó una entrevista al Msc. Yasiel Pérez Villazón (especialista involucrado en el desarrollo de NovaRSAT) (ver Anexo 1), en dicha entrevista se identificaron con claridad las funcionalidades que debía proporcionar el módulo para la configuración de archivos en NovaRSAT.

El trabajo que se presenta está estructurado en tres **Capítulos**, los cuales se despliegan a continuación:

Capítulo I. Fundamentación teórica sobre el proceso de configuración del sistema de archivos en sistemas operativos

Se definen los principales conceptos relacionados con el tema de investigación para lograr un mejor entendimiento del problema a resolver, lo que incluye un estudio del proceso de configuración de archivos y un análisis de las soluciones homólogas existentes que aumentan la confiabilidad de los datos informáticos. Se describe la metodología, herramientas y tecnologías a emplear resultando en la selección de una de estas para llevar a cabo el desarrollo del módulo para la configuración del sistema de archivos de NovaRSAT.

Capítulo ll: Análisis y Diseño del módulo para la configuración del sistema de archivos para NovaRSAT:

En este capítulo se describen los principales conceptos asociados al módulo para la configuración de archivos NovaRSAT, la técnica empleada para la captura y validación de requisitos funcionales y no funcionales, diseño arquitectónico que se aplicará al módulo, diagrama de clases del diseño según el marco de trabajo que se utiliza y patrones de diseño.

Capítulo III. Implementación y Pruebas del módulo para la configuración del sistema de archivos para NovaRSAT.

En este capítulo se describe como está implementada la herramienta, a través de los Diagramas de Componentes y el Diagrama de Despliegue. Se diseña un plan de pruebas con el propósito de definir el alcance de la misma y se verifica el funcionamiento de la solución propuesta.

El presente trabajo contiene además **Conclusiones**, **Recomendaciones**, **Referencias Bibliográficas** y **Anexos.**

Capítulo I Fundamentación teórica sobre el proceso de configuración del sistema de archivos en sistemas operativos

Introducción

En el presente capítulo se analizan los conceptos y elementos teóricos relacionados con el desarrollo de la aplicación para la configuración del Sistema de Archivos NovaRSAT. Se plasma además el resultado del estudio realizado a las principales soluciones que resuelven de forma parcial el problema de investigación planteado, estableciendo con estos sistemas homólogos cierta comparación. Por último, se describe la metodología seleccionada para guiar el proceso de desarrollo de software, tecnologías y las herramientas que se seleccionaron para el diseño e implementación de la solución.

1.1. Conceptos fundamentales

Como parte del proceso investigativo se definen una serie de conceptos asociados al tema para comprender el contexto del mismo.

Módulo: En programación, un módulo es una porción de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará, comúnmente, una de dichas tareas (o varias, en algún caso). En general (no necesariamente relacionado con la programación), un módulo recibe como entrada la salida que haya proporcionado otro módulo o los datos de entrada al sistema (programa) si se trata del módulo principal de este; y proporcionará una salida que, a su vez, podrá ser utilizada como entrada de otro módulo o bien contribuirá directamente a la salida final del sistema (programa), si se retorna al módulo principal (Sommerville ,2005).

Configuración Remota: permite configurar su máquina rápidamente aprovechando la facilidad y rapidez de la programación. Al acceder a esta aplicación, las configuraciones de su máquina se descargan automáticamente y se mostrarán en la pantalla de su computadora. Si usted cambia las configuraciones, usted podrá transferirlas automáticamente a la máquina (Brother,2001).

Sistema de archivos: Un sistema de archivos o sistema de ficheros, en informática, es un componente que controla cómo se almacenan y recuperan los datos. Sin un sistema de archivos, los datos colocados en un medio de almacenamiento serían un gran cuerpo de datos sin manera de saber dónde termina un dato y comienza el siguiente. Es el encargado de administrar y facilitar el uso de las memorias periféricas, ya sean secundarias o terciarias. Sus principales funciones son la

asignación de espacio a los archivos, la administración del espacio libre y del acceso a los datos resguardados. Estructuran la información guardada en un dispositivo de almacenamiento de datos o unidad de almacenamiento (normalmente un disco duro de una computadora), que luego será representada ya sea textual o gráficamente utilizando un gestor de archivos (Alegsa, 2016a).

Tipos de sistemas de archivos

- NTFS *(New Technology File System)*, en español Sistema de Archivos de nueva Tecnología
- HPFS (*High Performance File System*), en español Sistema de Archivos de alto Rendimiento
- EXT *(Extended file System),* en español Sistema de Archivos Extendido
- HFS+ *(Hierarchical File System),* en español Sistema de Archivos Jerárquico
- APFS *(Apple File System),* en español Sistema de archivos de Apple
- FAT *(File Allocation Table),* en español Tabla de asignación de archivos
- exFAT *(Extended File Allocation)* en español Asignación de archivos ampliada
- FAT32 *(File Allocation Table)* en español Tabla de asignación de archivos

NTFS

- Teóricamente, NTFS ofrece un tamaño de archivo de 16 EB, que equivalen a 16000000000 GB.
- NTFS está soportado tanto en escritorio, como en sistemas operativos para servidores (Linux, principalmente) (Aller, Ángel. 2020)

Ventajas:

- No tiene límite por archivo.
- Es un sistema ideal para unidades que se utilicen en Windows.
- Desde Mac se puede leer toda la información.
- Es compatible con GNU/Linux.

Desventajas

- En Mac no podremos escribir en un NTFS, ni es un SO compatible con este sistema.
- En TVs antiguas, los USB tenían que ser FAT32, pero es algo casi extinto.

FAT32

- Sistema ideal para dispositivos portátiles porque contiene una limitación de tamaño por archivo de 4 GB.
- Su principal base es la altísima compatibilidad que presenta en cualquier sistema operativo o medios de reproducción.
- No se pueden almacenar archivos individuales de 4 GB.
- Una partición FAT32 no puede ser superior a 8 TB. (Aller, Ángel. 2020)

Ventajas:

- Compatible con todos los SO, videoconsolas, televisiones, etc.
- Perfecto para unidades pequeñas con archivos pequeños.

Desventajas:

- Limitación de 4 GB por archivo.
- Partición inferior a 8 TB.

exFAT

- Es la evolución de FAT32 porque se elimina las limitaciones que tenía FAT32.
- Es compatible con Windows y Mac, por lo que es perfecto para aquellos discos duros externos que queráis utilizar en ambos sistemas operativos. (Aller, Ángel. 2020)

Ventajas:

- Es compatible con Windows y Mac
- Ideal para pendrives y discos duros externos que se usen en ambos SO.
- Se eliminan las limitaciones de FAT32.

Desventajas:

- No es del todo compatible con Linux.
- Puede no ser compatible con ciertos medios de reproducción.

Carpetas o Directorios

Un directorio es una agrupación de archivos de datos, atendiendo a su contenido, a su propósito o a cualquier criterio que decida el usuario. Sirven para organizar mejor los archivos en un medio de almacenamiento como un disco duro, un dispositivo de almacenamiento USB, un CD, etc. Dentro de

un directorio pueden existir también otros directorios, llamados subdirectorios o subcarpetas; de hecho, todos los directorios son subdirectorios del directorio raíz la unidad lógica (Alegsa, 2016b).

Los directorios o carpetas permiten al usuario almacene todo tipo de información dentro de un ordenador, siendo clasificada directamente por el sistema operativo o diciéndole el propio usuario al sistema donde y como se clasifique esa información en el ordenador, para así de este tener todo organizado a su gusto (tecnologicon, 2015).

Tipos de directorios

- Los directorios definidos por el sistema que contienen archivos específicos del mismo.
- El directorio root que se encuentra en la parte superior de la jerarquía del sistema de archivos.
- El directorio raíz que contiene archivos especiales, para iniciar y gestionar el sistema, archivos temporales entre otros.
- Los directorios definidos son creados por el administrador como el caso de inicio de sesión. (Aldeahost, 2020)

Archivos

En informática se conoce como archivo o fichero a un conjunto organizado de unidades de información (bits) almacenados en un dispositivo. Se les denomina de esa manera como metáfora de a partir de los archivos tradicionales de oficina, escritos en papel, y que vendrían a ser su equivalente digital. Cada archivo posee una identificación única o nombre, la cual puede ser modificada o asignada a voluntad del usuario o del programador, y una extensión que determina qué tipo de archivo es y qué funciones cumple (Raffino , 2019). Un archivo tiene generalmente los siguientes atributos: nombre, tipo, ubicación, tamaño, protección, hora, fecha. Estos se pueden estructurar de varias maneras las más comunes son:

Secuencia de bytes: El archivo es una serie no estructurada de bytes, posee máxima flexibilidad y el sistema operativo no sabe que contiene el archivo.

Secuencia de registros: El archivo es una secuencia de registros de longitud fija, cada uno con su propia estructura interna.

Árbol: El archivo consta de un árbol de registros, no necesariamente de la misma longitud, cada registro tiene un campo llamado *key* (llave o clave) en una posición fija del registro. El árbol se ordena mediante el campo clave para permitir una rápida búsqueda de una clave particular (Osnaya, 2012).

1.2 Descripción del proceso de configuración del sistema de archivos

1.2.1 Configuración des sistema de archivos en GNU/Linux

Estructura el sistema de archivos en GNU/Linux

Los sistemas Linux residen bajo un árbol jerárquico de archivos, bastante parecido a como se estructura los sistemas Unix. En sus inicios, ese árbol jerárquico de directorios y archivos no estaban bajo ningún estándar, es decir, existían variaciones entre el de una distribución y otra. Fue esto lo que motivo a grupo de personas a desarrollar, en el año 1993, lo que se conoce como *Filesystem Hierarchy Standard (FHS)* o en español Estándar de Jerarquía de Sistema de Ficheros**.**

FHS

El FHS se define como el estándar que establece y brinda el detalle de los nombres, contenidos, ubicaciones y permisos de los archivos y directorios, en otras palabras, es el conjunto de reglas que determinan una estructuración común de archivos y directorios en los sistemas Linux. Este estándar no es más que un documento de guía, el cual puede ser consultado por los fabricantes y ser aplicado al momento de crear una nueva distribución.

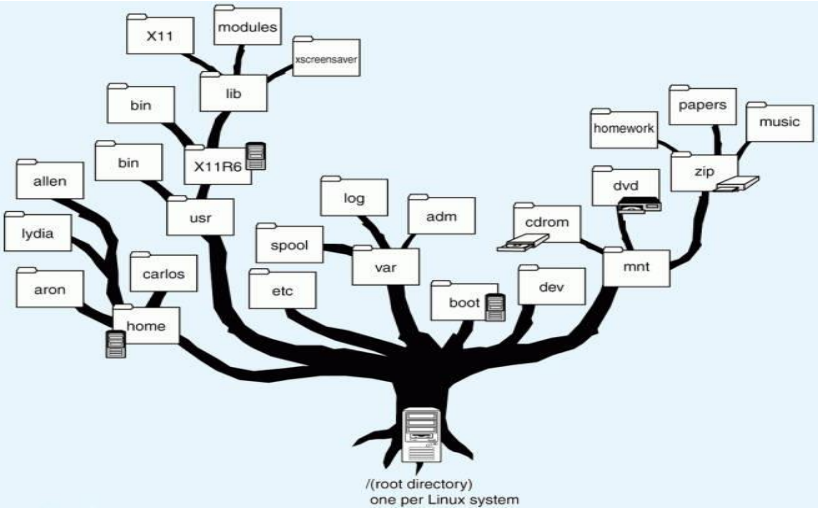


Figura 1 Estructura el sistema de archivos en GNU/Linux

Fuente: Elaboración propia

1.2.2 Configuración des sistema de archivos en Windows

Microsoft sistema operativo Windows utiliza dos sistemas principales de archivos: **FAT** , heredadas del viejo DOS con su posterior extensión **FAT32** , y ampliamente usados **NTFS** sistemas de archivos. Recientemente lanzado **ReFS**sistema de archivos fue desarrollado por Microsoft como una nueva generación de sistema de archivos para Windows 8 Servidores.

La raíz **en S.O. Windows** se representa a través de la barra de directorio: **** (símbolo localizado en la tecla con los caracteres **º** y **ª ;** parte izquierda del teclado), aunque suele ir precedido del identificador de la unidad.

Los sistemas de archivos utilizados por Windows son:

* FAT: FAT12, FAT16 (usados en MS-DOS y Windows 1.0 hasta Windows 95): Para las uni

* FAT32 (estándar para Windows 98 y Windows ME).

* NTFS (estándar para Windows NT, Windows XP, Windows 2000).

* ISO 9660 (soportado desde Windows 95 en adelante): Para los cd-room

* UDF (soportado desde Windows 98 en adelante, con excepción del ME): Es un sistema de archivos con estándar ISO 9660, propiedad de Adaptec, que utiliza las grabadoras de CD/DVD

1.3 Análisis de herramientas informáticas para la configuración del sistema de archivos en sistemas operativos.

1.3.1 Herramientas en Windows

Explorer: Ha sido el administrador de archivos estándar de la familia de sistemas operativos Microsoft Windows y el programa proporciona a los usuarios funciones básicas para la gestión de archivos y directorios, a través de una interfaz de usuario ordenada. La ventana de Explorer está dividida en dos áreas. En el lado izquierdo encontrarás tu directorio de archivos en forma de estructura de árbol. El campo de la derecha se utiliza para mostrar el contenido de la carpeta. Si es necesario, se puede mostrar un tercer campo, la ventana de vista previa. Además, el Explorador de Windows proporciona el escritorio (el escritorio en segundo plano), así como todos los iconos que aparecen en él, la barra de tareas y el menú Inicio.

FreeCommander XE 2017: FreeCommander XE 2017 es la última versión del popular gestor de archivos freeware creado por Marek Jasinski para las versiones XP, Vista, 7, 8 y 10 de Windows. A pesar de su enorme gama de funciones, la interfaz de usuario de FreeCommander es sencilla y ordenada. La disposición de los elementos en dos ventanas permite el trabajo en paralelo. Los archivos y directorios se muestran en una clara estructura de árbol en ambos lados y las ventanas de archivos adicionales se pueden adjuntar como pestañas. De igual forma que con el Explorer de Windows, se puede realizar operaciones con archivos cómodamente mediante la función arrastrar y soltar, la barra de menús o el menú contextual. También se puede controlar el programa a través de accesos directos definidos por el usuario. De este modo, las operaciones estándar pueden llevarse a cabo de forma especialmente eficiente. FreeCommander también ayuda con varias funciones adicionales que aceleran los procesos de trabajo, como renombrar, comparar o sincronizar directorios y filtrar la vista de archivos según criterios definidos. Las carpetas y programas que se utilizan con frecuencia se pueden definir como favoritos si es necesario. El programa incluye un sistema de empaquetado integrado para archivos ZIP, un visor de archivos con vista previa (incluso para los datos archivados) y soporte FTP. Otros formatos de archivo como RAR o 7z se pueden implementar mediante plugins.

Q-Dir

La aplicación gratuita Q-Dir, de Software OK está disponible para usuarios de las versiones de Windows XP, Vista, 7, 8, 8, 8.1 y 10 y para Microsoft Server 2000 hasta 2016. El programa se diferencia del explorador de Windows principalmente por un diseño que le permite administrar tu sistema de archivos en una vista de cuatro ventanas. Una característica distintiva del gestor de archivos de Windows Q-Dir es la disposición de los elementos en cuatro ventanas. En cada ventana del gestor de archivos se puede crear hasta cuatro pestañas y trabajar con hasta 16 carpetas de archivos de forma simultánea. Q-Dir es, por lo tanto, particularmente conveniente para los usuarios que desean realizar operaciones de archivos a través de varios discos duros o medios extraíbles. Al igual que en el Explorador de Windows, el manejo se realiza a través de la barra de menús o cómodamente arrastrando y soltando. Existen combinaciones de teclas y enlaces rápidos para realizar operaciones estándar.

SpeedCommander: El administrador de archivos shareware de SpeedCommander también ofrece una vista clara de dos ventanas y le permite administrar ventanas de carpetas como pestañas. El programa está disponible en 12 diseños diferentes, que se basan en los productos de Microsoft Windows XP, MS Office o Visual Studio. Los usuarios también pueden elegir entre vistas al estilo de

Windows Explorer y al estilo de Norton Commander. El programa soporta 13 formatos de archivo, así como los protocolos FTP y FTP vía SSL. Si es necesario, SpeedCommanderpermite acceder a un servidor FTP y gestionar los archivos de forma remota, así como en el disco duro local. FileSync puede sincronizar carpetas y archivos, mientras que las tareas rutinarias pueden automatizarse con macros. La herramienta viene con un editor para estas macros. Los usuarios que necesiten más funciones tienen la opción de implementar extensiones como servidores COM en proceso a través de una interfaz AddIn integrada.

Además de la versión estándar, el gestor de archivos también se ofrece como SpeedCommander Pro con una amplia gama de funciones por un precio fijo. Esto incluye un programa para grabar CD, DVD y Blu-ray, una herramienta para la administración de archivos de imagen (ISO/BIN), interfaces para los proveedores en la nube Dropbox, Google Drive y OneDrive.

TotalCommander: El gestor de archivos de Windows TotalCommander se distribuye a través de Ghisler Software GmbH como shareware. Como otras alternativas, la herramienta presenta muchas ventajas en funciones frente al Explorador de Windows. Total Commander también utiliza una vista multiventana con la que se pueden mostrar dos ventanas de archivos en paralelo. El diseño de la herramienta se basa en el Explorador de Windows. Para las operaciones de archivo, se puede utilizar la barra de menús, un menú contextual y las funciones de arrastrar y soltar. No se requieren programas adicionales para el utilizar FTP; la herramienta viene con las funciones correspondientes. También proporciona a los usuarios una vista rápida, una función de búsqueda avanzada y funciones para comparar y sincronizar directorios. La opción de favoritos, el historial y la protección con contraseña de los directorios seleccionados completan la gama de funciones (Guide Digital, 2019).

1.3.2 Herramientas en Linux

El gestor de archivos estándar disponible en Linux depende de la distribución y el entorno de escritorio que estés utilizando. La siguiente tabla muestra los entornos más populares para sistemas Linux y el correspondiente gestor de archivos estándar:

Nautilus: Nautilus es el gestor de archivos estándar del entorno de escritorio GNOME y la interfaz de usuario Unity, que se utiliza en Ubuntu hasta la versión 17.0. Además de las funciones básicas de gestión de archivos, Nautilus ofrece pestañas, una vista previa de imágenes, una función de búsqueda y acceso a recursos compartidos remotos a través de Samba, FTP, SFTP, WebDAV o SSH. Los elementos se pueden mostrar como iconos (con vista previa del archivo), en una cuadrícula

o como una lista con detalles. Las extensiones definidas por el usuario se pueden integrar a través de una interfaz a través de un plugin.

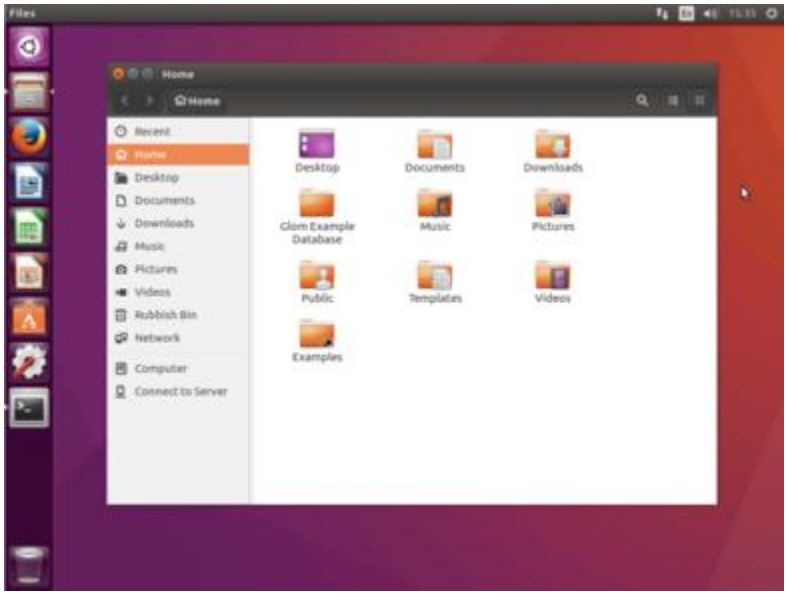


Figura 2 Interfaz de usuario del gestor de archivos para Linux Nautilus

(Guide Digital.2019)

Ventajas

- Pestañas
- Función de compresión de archivos disponible
- Acceso a sistemas de archivos en la red
- Ampliable a través de plugins

Desventajas

- Rango de funciones significativamente reducido desde la versión 3.6
- Sin visor de archivos integrado
- Sin función de sincronización de directorios
- Sin vista multiventana

Dolphin: Dolphin es el gestor de archivos predeterminado del entorno de escritorio KDE. Dolphin se diferencia de Nautilus en que dispone de una barra de navegación estilo breadcrumby una

visualización del espacio de almacenamiento disponible. La vista de la ventana se puede dividir si es necesario. También hay tres modos de visualización para elegir: iconos, detalles y columnas.

Las operaciones de archivo que se pueden realizar con Dolphin incluyen mover, copiar, renombrar, eliminar, administrar propiedades y permisos, y seleccionar archivos. Además, el administrador de archivos proporciona funciones para archivos ocultos, accesos directos y a los servidores de la red (por ejemplo, Samba, FTP, SSH o WebDAV). En caso necesario, se pueden integrar otras funciones a través de un plugin.

Figura 3 Interfaz de usuario de Dolphin File Manager

(Guide Digital, 2019)

Ventajas

- Vista en dos ventanas
- Pestañas
- Barra de navegación estilo breadcrumb
- Acceso a sistemas de archivos en red
- Función de compresión de archivos disponible
- Diversos plugins

Desventajas

- Sin visor de archivos integrado
- Sin función de sincronización de directorios

Thunar: Thunar es el administrador de archivos predeterminado del entorno de escritorio Linux Xfce, similar a Nautilus. Este programa minimalista ofrece funciones básicas para la gestión de archivos y directorios locales, así como acceso remoto a recursos compartidos a través de FTP, SAMBA, WebDAV y otros protocolos. Una característica destacada del software es la opción de menú "Acciones definidas por el usuario", que permite ampliar las funciones Thunar según las necesidades individuales; los usuarios solo añaden las funciones que realmente necesitan.

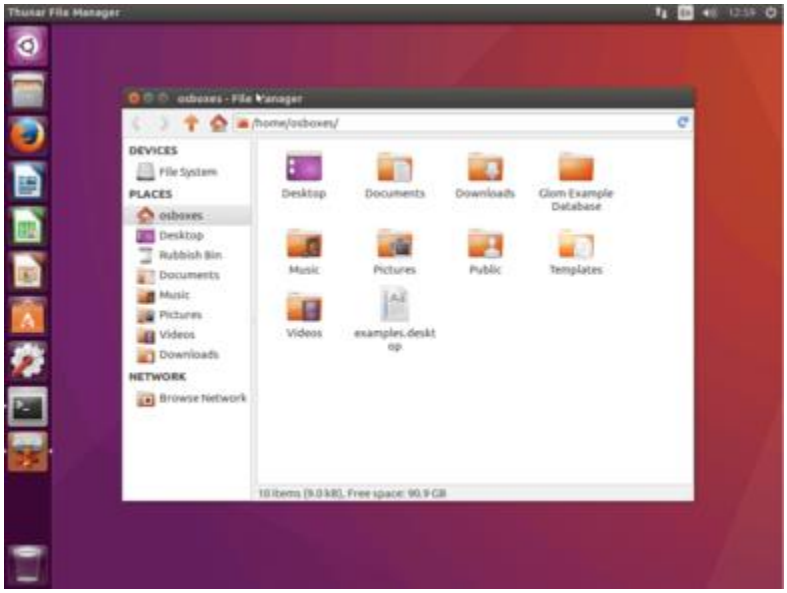


Figura 4 Interfaz de usuario del gestor de archivos de Linux Thunar

(Guide Digital, 2019)

Ventajas

- Pestañas
- Acceso a sistemas de archivos en red
- Diversas ampliaciones mediante "Acciones definidas por el usuario”

Desventajas

- No dispone de vista multiventana
- Función de compresión de archivos disponible solo a través de plugins
- Sin función de sincronización de directorios

PCMan File Manager: PCMan File Manager, el gestor de archivos estándar del entorno de escritorio LXDE, es una alternativa compacta a Nautilus, que principalmente gana a este en su velocidad. El

programa del desarrollador de software taiwanés Hong Yen Jee , ofrece a los usuarios una interfaz con hasta dos ventanas de archivos y navegación por pestañas, en la que se pueden abrir diferentes carpetas de archivos en pestañas separadas, una función de vista previa de imágenes y una vista de directorio como estructura de árbol. Además, PCManFM puede manejar juegos de caracteres no compatibles con UTF-8.

Las funciones para comprimir y descomprimir archivos ya están integradas, al igual que el acceso a sistemas de archivos remotos. PCManFM se distingue de otros gestores de archivos Linux por una interfaz de usuario desarrollada desde cero según criterios de accesibilidad y que, por tanto, tiene en cuenta las necesidades de un gran número de usuarios.

Figura 5. Interfaz de usuario del gestor de archivos de Linux PCManFM

(Guide Digital, 2019)

Ventajas

- Vista en varias ventanas
- Pestañas
- Adaptada para accesibilidad universal
- Soporte FTP
- Ampliable a través de plugins
- Función de compresión de archivos integrada

Desventajas

- Sin función de sincronización de directorios

Nemo: Nemo, el gestor de archivos estándar del entorno de escritorio Cinnamon, es un spin-off de Nautilus 3.4. Tiene disponible una vista de dos ventanas y pestañas para gestionar vistas de directorio separadas. Al igual que Nautilus, Nemo puede ampliarse con numerosos plugins.

Figura 6. Interfaz de usuario del gestor de archivos de Linux Nemo

(Guide Digital, 2019)

Ventajas

- Vista en varias ventanas
- Pestañas
- Barra de navegación estilo breadcrumb
- Acceso a sistemas de archivos en la red
- Función de compresión de archivos integrada
- Ampliable a través de plugins

Desventajas

- Sin función de sincronización de directorios

SpaceFM: SpaceFM es un spin-off del gestor de archivos estándar de LXDE, PCManFM. Desde la versión 0.9.x, SpaceFM se centra en un diseño sencillo para permitir flujos de trabajo rápidos y

eficientes, mientras que la flexibilidad del software y su extensibilidad están en primer plano en SpaceFM.

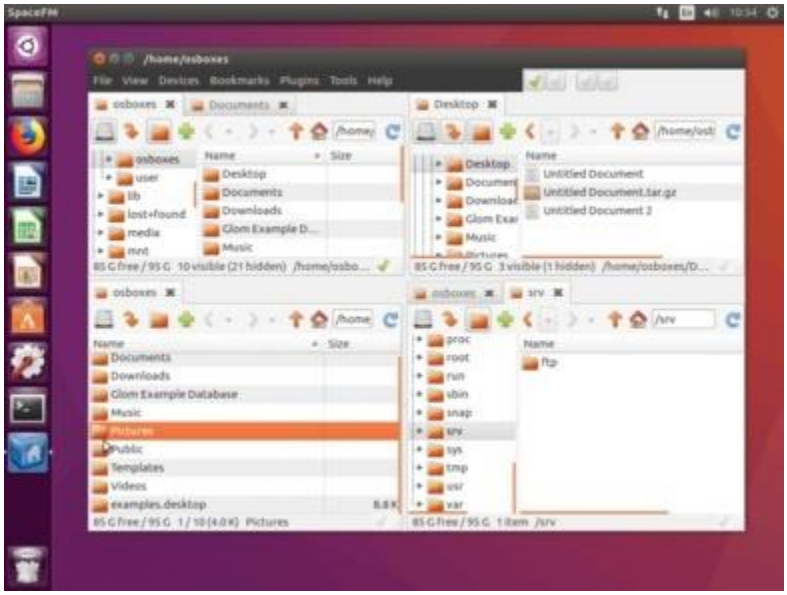


Figura 7. Interfaz de usuario del gestor de archivos de Linux SpaceFM

(Guide Digital, 2019)

Con el administrador de archivos para Linux SpaceFM se puedes gestionar tu sistema de archivos en hasta cuatro ventanas, que pueden configurarse por separado. El programa permite mostrar hasta cuatro carpetas en paralelo gracias a la vista multiventana. Las carpetas de archivos también se pueden adjuntar como fichas. Cada ventana de la interfaz de usuario puede configurarse por separado de las demás. El programa ofrece imágenes de vista previa para archivos de imagen y vídeo, una función de búsqueda avanzada, marcadores y una función de compresión integrada. El menú Marcadores permite enlazar no solo a carpetas y archivos, sino también a comandos personalizados y aplicaciones que se ejecutan con regularidad. Se pueden establecer conexiones a redes remotas a través de NFS, FTP, SMB y SSH.

Ventajas

- Vista en varias ventanas
- Pestañas
- Función de compresión de archivos disponible
- Acceso a sistemas de archivos en red
- Ampliable a través de plugins

- Interfaz de usuario personalizable

Desventajas

- Sin función de sincronización de directorios (se pueden agregar con plugins)

Krusader: Es el gestor de archivos de Linux más potente originalmente desarrollado para KDE, el programa se basa en la disposición de Norton Commander.

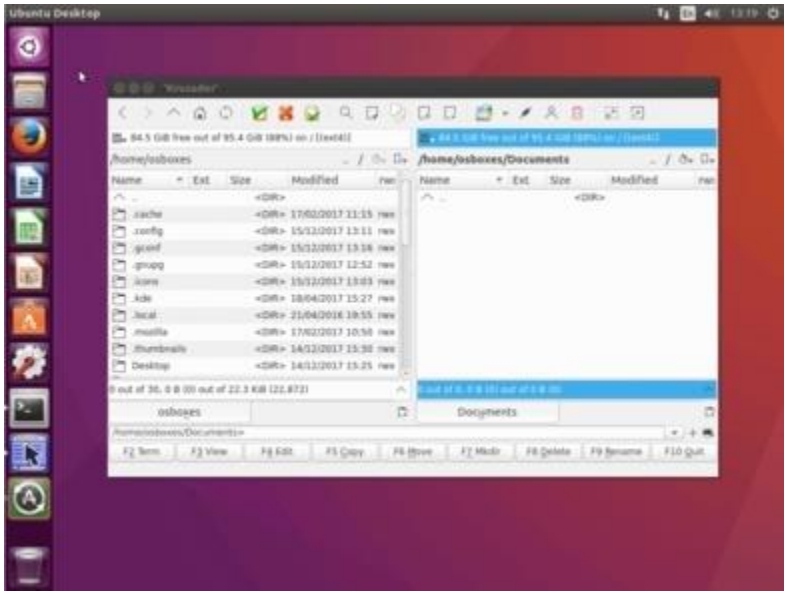


Figura 8. Interfaz de usuario del gestor de archivos de Linux Krusader

(Guide Digital, 2019)

El administrador de archivos le ofrece una vista detallada de dos columnas de su sistema de archivos. Si se desea, se puede pegar directorios en ambas ventanas como pestañas. Como aplicación KDE, Krusader soporta todos los protocolos del sistema de archivos virtuales KDE Input/Output (KIO) - incluyendo FTP, SMB y NFS. Con el gestor de archivos integrado puede empaquetar y descomprimir archivos y directorios en varios formatos de archivo. Además de las operaciones básicas habituales de copiar, cortar, borrar y mover, Krusader ofrece funciones para comparar y sincronizar archivos y directorios, para crear y verificar sumas de comprobación y para renombrar en masa. Una vista previa del archivo, el visor integrado con editor y una función de búsqueda ampliada completan la gama de funciones. Si es necesario, también puede ampliar Krusader con acciones definidas por el usuario.

Ventajas

- Vista en varias ventanas
- Pestañas

- Función de compresión de archivos disponible
- Sincronización de directorios
- Acceso a sistemas de archivos en la red
- Visor y editor de archivos integrado
- Ampliable a través de plugins

Desventajas

- ✓ Sin desventajas significativas

En la siguiente tabla se realiza una comparación entre los diferentes gestores de archivos tanto de Linux como para Windows. Se utiliza una gama de funciones para realizar dicha comparación las cueles fueron elegidas de acuerdo con las funcionalidades requeridas por el cliente para la implementación de la aplicación.

1.4 Comparación de las herramientas

En la siguiente tabla se realiza una comparación de las diferentes herramientas existentes para la configuración del sistema de archivos en los sistemas operativos Linux y Windows.

Tabla 1. Tabla resumen de los gestores de archivos de Windows y Linux

(Guide, Digital, 2019)

Gama de funciones	**Nautilus**	**Dolphin**	**Thunar**	**PCManFM**	**Nemo**	**SpaceFM**	**Krusader**
Crear carpeta/mover/copiar/pegar	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Licencia	GPL	GPL	GPL	GPL	GPL	GPL	GPL
Coste	Gratuito	Gratuito	Gratuito	Gratuito	Gratuito	Gratuito	Gratuito
Versiones del sistema operativo compatibles	Todas las de Linux	Todas las de Linux	Todas las de Linux	Todas las de Linux	Todas las de Linux	Todas las de Linux	Todas las de Linux
Acceso a sistemas de archivos remotos	No	No	No	No	No	No	Sí
Gama de funciones	**FreeComander**	**Q-Dir**	**SpeedCommander**	**TotalComander**			
Crear carpeta/eliminar/mover/copiar	Sí	Sí	Sí	No			
Arrastrar y soltar	Sí	Sí	Sí	Sí			
Pestañas	Sí	Sí	Sí	Sí			
Función de formar paquetes de archivos integrada	Sí	No	Sí	No			

El análisis de los diferentes gestores de archivos existentes para el sistema operativo Linux y Windows permitió conocer la configuración de los sistemas de archivos y cómo funcionan, además de sus características, sin embargo, ninguno de ellos brinda el acceso a sistemas de archivos remotos lo cual es una característica para la realización del módulo de configuración del sistema de archivos en NovaRSAT. Krusader a pesar de poseer funcionalidades similares a las del módulo a realizar no se puede adoptar como solución debido a que no se puede integrar con la herramienta NovaRSAT. TotalComander proporciona un visor de archivos sin embargo no brinda tiene la funcionalidad para crear una carpeta, eliminarla, moverla o copiarla. No obstante estos gestores proporcionan algunas funcionalidades que pueden ser incluidas en la realización de dicho módulo y se obtiene la funcionalidad del gestor FreeComander ya que este proporciona la creación y modificación de carpetas.

1.5 Lenguaje y herramienta para el modelado de la propuesta solución

1.5.1 Herramienta CASE y lenguaje de modelado

Las herramientas CASE *(Computer Aided Software Engineering, por sus siglas en inglés*, o Ingeniería de Software Asistida por Ordenador) son aplicaciones informáticas con el objetivo de aumentar la productividad y la eficiencia en el desarrollo del software, minimizando de esta forma el costo en términos de tiempo y dinero.

La misión de cualquier herramienta CASE utiliza el Lenguaje Unificado de Modelado (UML) como notación para elaborar los modelos, es comunicar, de manera más eficiente posible a todos los agentes del proyecto, todas aquellas decisiones que se toman con respecto a la arquitectura del sistema en discusión y que son determinadas para cumplir con los objetivos del proyecto. Establecer una arquitectura racional y eficiente, acorde a las necesidades de los actores, es la razón fundamental para invertir tiempo y esfuerzos en una herramienta CASE que utiliza la notación UML.

Visual Paradigm v8.0

Herramienta de software diseñada para modelar los sistemas de información empresarial y gestionar el proceso de desarrollo de software. Soporta lenguajes de modelado y estándares claves como UML, SysML (System Modeling Language) en español Lenguaje de modelado del sistema, BPMN (Bussines Process Model and Notation) en español Modelo de proceso empresarial y notación , entre otros. Ofrece además un conjunto completo de herramientas de software para tareas como la captura de requisitos, análisis de procesos, diseño de sistemas y diseño de base de datos (Visual Paradigm Frequently Asked Questions 2017).

UML v2.0

Lenguaje de Modelado Unificado (por sus siglas en inglés UML) es un lenguaje de modelado estandarizado que consiste en un integrado conjunto de diagramas; desarrollados para ayudar a desarrolladores de software y de sistema en la especificación, visualización, construcción u documentación de los artefactos de los sistemas de software, así como también para el modelado de negocio y otros sistemas (What is Unified Moderan Language (UML)? 2017).

1.5.2 Fundamentación de la herramienta y lenguaje de modelado a utilizar

Para el modelado fue seleccionado Visual Paradigm, que tiene características graficas muy cómodas, que facilitan la realización de los diagramas de modelado que siguen el estándar de UML.

También posee cualidades como la generación automática de diagramas a partir de descripciones de casos de usos, permitiendo la agilidad en el trabajo del analista. Además usa un lenguaje estándar común para todo el equipo de desarrollo que facilita la comunicación. Como el lenguaje a utilizar para el modelado se escogió UML, ya que este permite visualizar, construir y documentar los artefactos de un proyecto de software. Además es el lenguaje que utiliza la herramienta escogida para realizar el modelado.

1.6 Metodología de desarrollo

En ingeniería de software una metodología de desarrollo constituye un conjunto de métodos, procedimientos, técnicas, herramientas y soportes documentales utilizados para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información.

Para la adecuada implementación de la solución propuesta es necesario la selección de una metodología que guíe el ciclo de vida del proyecto para asegurar un producto de calidad. Se selecciona en consecuencia la metodología AUP-UCI teniendo en cuenta que es la metodología adaptada al ciclo de vida de los proyectos productivos de la universidad, es ampliamente usada en el área y es extremadamente flexible al proceso de desarrollo de software.

1.6.1 Metodología de desarrollo de software AUP-UCI

AUP-UCI constituye una variante de AUP (Proceso Unificado Ágil, por sus siglas en inglés) surge con el objetivo de ser una metodología que se adapte al ciclo de vida definido por la actividad productiva en la universidad. Se elaboró teniendo en cuenta el Modelo CMMI-DEV v1.3 que constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora, estas prácticas se centran en el desarrollo de productos y servicios de calidad.

Esta metodología propone tres fases para el desarrollo del software (Inicio, Ejecución y Cierre), siete disciplinas (Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación, Pruebas de aceptación. (Ramírez, 2018).

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

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

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

AUP-UCI propone además cuatro escenarios a utilizar para modelar el sistema en los proyectos, de los cuales se escoge:

- **Escenario No 4:** modelar el sistema con HU (Historia de Usuario) cuando no se realice modelado de negocio. (Ramírez, 2018)

Esta metodología se adapta al ambiente de trabajo y le permite al cliente acompañar al equipo de desarrollo para convenir los requisitos y así poder implementarlos. De los escenarios descritos anteriormente se selecciona para el modelado del sistema de la solución a desarrollar el escenario número cuatro .El escenario escogido para esta investigación es el No 4, pues la solución no posee un negocio, este escenario se utiliza para modelar el sistema. Además, se puede concluir que esta metodología se ajusta a cualquier proyecto productivo de la UCI.

1.7 Herramientas, tecnologías y lenguajes para el desarrollo

La selección del lenguaje, tecnologías y herramientas constituye un factor clave en el desarrollo de una solución informática. Esta elección está guiada principalmente por los requerimientos que presenta el software a desarrollar. En la presente sección se definen las tecnologías y herramientas que se escogieron para la implementación.

1.7 Herramientas y tecnologías

Es el conjunto de instrumentos empleados para manejar información por medio de la computadora como el procesador de texto, la base de datos, graficadores, correo electrónico, hojas de cálculo, buscadores, programas de diseño, presentadores, redes de telecomunicaciones (CHALARCA, 2016). La tecnología es la aplicación coordinada de un conjunto de conocimientos (ciencia) y habilidades (técnica) con el fin de crear una solución (tecnológica) que permita al ser humano satisfacer sus necesidades o resolver sus problemas (MASTER MAGAZINE, 2016). A continuación, se presentan las herramientas y tecnologías utilizadas en el desarrollo de la solución propuesta.

1.7.1 Marco de trabajo

Un marco de trabajo no es más que una estructura de soporte definida mediante la cual otro proyecto de software puede ser de desarrollado y organizado. Puede incluir soporte de programas, biblioteca y un lenguaje interpretado entre otros software para ayudar desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela de las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Electron 13.0.1

Electron es una plataforma para construir aplicaciones multiplataforma (que pueden ejecutarse en macOS, Windows y Linux) utilizando tecnologías web (html, javascript y css). Permite que un equipo de trabajo sea más productivo al solo tener que enfocarse en un solo desarrollo que cubra múltiples plataformas, es decir que no se requiere tener un proyecto en desarrollo para cada sistema operativo (Windows, macOS y Linux) sino que al mantener uno solo se pueden realizar las actualizaciones para los tres. Para los desarrolladores de Nodejs, electron es una solución que permite complementar node de forma que no se limite a la línea de comandos, sino que se pueda extender su uso al crear interfaces de usuario sin la necesidad de tener que aprender un nuevo lenguaje de programación. Simultáneamente si se tiene experiencia creando aplicaciones web, Electron te permite utilizar los recursos del sistema operativo, como acceder al sistema de archivos y otras acciones que están restringidas para el navegador. Por ejemplo, es posible crear una aplicación que pueda no solo explorar sino también escribir dentro del sistema de directorios y archivos locales. (Apuntes de Electron, Electron 2020).

La selección de este IDE se basa principalmente en la experiencia del equipo de desarrollo en su uso, y además en que es el recomendado por Google para el desarrollo de aplicaciones de escritorio para Linux puesto que está implementado con tal propósito.

1.7.2 Lenguaje de programación

Un lenguaje de programación es una interfaz de usuario con la cual los seres humanos pueden elaborar procesos que serán ejecutados por una máquina de cómputo. Los lenguajes de programación tienen reglas y parámetros establecidos para poder realizar diferentes acciones, existen diferentes lenguajes de programación y dependen del tipo de programación (Bellas, Unanue y Fernández 2016).

Para la realización del módulo para la configuración del sistema de archivos NovaRSAT se escoge JavaScript.

JavaScript

JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat. JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo, programación funcional) (Developer Guide Mozilla | JavaScript 2020).

JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos. Es fácil para los humanos leer y escribir. Es fácil para las máquinas analizar y generar. Se basa en un subconjunto del estándar de lenguaje de programación JavaScript ECMA-262 3.ª edición - diciembre de 1999. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son familiares para los programadores de la familia de lenguajes C, incluido C, C ++, C #, Java, JavaScript, Perl, Python y muchos otros. Estas propiedades hacen de JSON un lenguaje de intercambio de datos ideal (Developer Guide Mozilla ,JSON 2020).

HTML

HTML es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de ¨etiquetas¨, rodeadas por corchetes

angulares (<,>), también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

CSS

Son las siglas de *Cascading Style Sheets*, en español Hojas de estilo en Cascada. Es una tecnología que permite crear páginas Web de una manera más exacta. Gracias a este lenguaje se pueden hacer mucho más cosas que antes no se podían hacer solamente HTML, como incluir márgenes, tipos de letra, fondos, colores, definir estilos en un archivo externo a nuestras páginas.

1.7.3 Entorno de Desarrollo Integrado

Es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programacion o bien, poder utilizarse para varios. Un IDE es un entorno de programacion que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz grafica. Los IDE pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

Visual Studio Code

Es un editor de código gratuito, ofrece a los desarrolladores soporte integrado para múltiples idiomas. El editor presenta todas las herramientas estándar de un editor de código moderno, incluyendo resaltado de sintaxis, enlaces de teclado personalizables, coincidencia de corchetes y fragmentos. Control integrado de versiones a través de Git (LARDINOIS, 2015).

1.7.4 Fundamentación de las herramientas, tecnologías y lenguajes a utilizar para el desarrollo de la aplicación

Para el desarrollo del modulo de configuración del sistema de archivos NovaRSAT se propone la utilización como lenguaje de programacion JavaScript por las siguientes razones fundamentales:

- JavaScript tiende a ser muy rápido porque a menudo se ejecuta inmediatamente en el navegador.
- La sintaxis de JavaScript está inspirada por Java y es relativamente sencillo de aprender comparado a otros lenguajes de programación populares como C++.

- JavaScript puede ser usado para crear características como arrastrar y soltar, y componentes tales como las diapositivas, lo cual mejora enormemente la interfaz de usuario y la experiencia del sitio.
- Hay muchos métodos para usar JavaScript mediante servidores Node.js.

Se propone la utilización de Electron debido a que al ser multiplataforma proporciona métodos específicos para integrarse mejor con el sistema operativo de destino. Gracias a las tecnologías en las que se basa Electron, las aplicaciones que lo usan tienden a verse y comportarse de la misma manera, independientemente de la plataforma en la que se estén ejecutando. Sea ese Linux o Mac OS X Yosemite, desde la perspectiva de un usuario de Linux Mac OS X Yosemite, desde la perspectiva de un usuario de Linux, Mac OS X se utiliza como el elemento secundario para una interfaz limpia y elegante.

1.8 Conclusiones parciales

Luego de realizar una minuciosa investigación de las principales características, ventajas y desventajas de cada una de las herramientas para la configuración del sistema de archivos, se pudo concluir que estas herramientas no permiten la configuración del sistema de archivos NovaRSAT. Por lo tanto, no es posible aplicar ninguna de ellas como propuesta de solución, ya que a pesar de las amplias posibilidades que brindan, en su generalidad no son adaptables para NovaRSAT.

Los lenguajes y tecnologías definidas para el desarrollo de la aplicación permiten elaborar un producto con calidad y brindan garantía de contar con librerías del software libre, una amplia documentación y robusta comunidad de desarrollo.

CAPÍTULO II: Análisis y Diseño del módulo para la configuración del sistema de archivos NovaRSAT.

Introducción

En el siguiente capítulo se describen las características del módulo para la configuración del sistema de archivos NovaRSAT. Se especifican las características que debe poseer dicho módulo a desarrollar mediante la definición de requisitos funcionales y no funcionales. Se representa el diseño arquitectónico a emplear durante el desarrollo de dicho módulo, el diagrama de clases del diseño según el marco de trabajo que se utiliza y los patrones de diseño.

2.1 Descripción de la propuesta de solución

A continuación, se representa la propuesta de solución para el módulo para la configuración del sistema de archivos NovaRSAT, con el objetivo de darle respuesta al problema de investigación plateado anteriormente y una descripción del contexto del problema. Actualmente NovaRSAT realiza la configuración del sistema de archivos mediante consola lo que significa una pérdida de tiempo además de la posibilidad que ocurran errores en la configuración de los clientes ligeros a causa de lo alto que es la curva de aprendizaje en el conocimiento de este servicio. Teniendo una aplicación de escritorio que le permita realizar la configuración del sistema de archivos de manera remota además de navegar como administrar la estructura de carpetas y archivos, tanto la del sistema como las que ha creado el usuario el mismo podrá cumplir con las configuraciones en un menor tiempo y de manera más eficiente.

2.1.1 Modelo de dominio

Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se le denomina modelo conceptual, modelo de objetos del dominio y modelo de objetos de análisis. Al utilizar la notación UML, un modelo de dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación. Su principal objetivo es definir las interrelaciones de los objetos más importantes representados mediante clases. Además, desempeña un papel clave en la comprensión del entorno actual (LARMAN, 1999). A continuación, se presenta el modelo de dominio que define a la propuesta de solución como se aprecia en la Figura 8

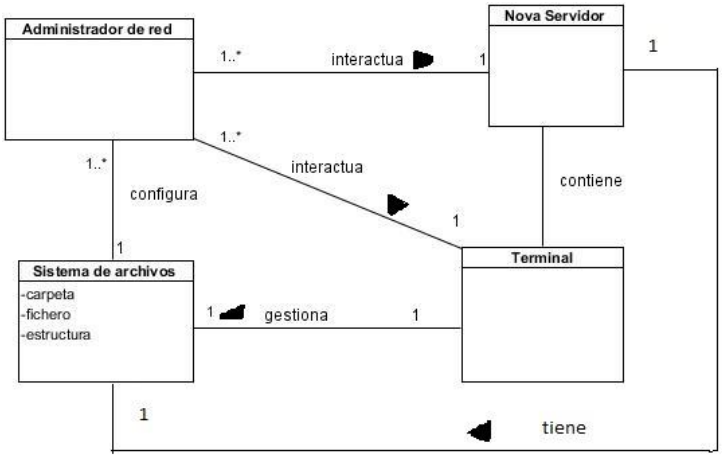


Figura 9. Representación de la Propuesta de Solución

(Fuente: Elaboración Propia)

Descripción de conceptos

Administrador de red: consume un servicio remoto en otro ordenador conocido como servidor, normalmente a través de una red de telecomunicaciones.

Nova servidor: consiste en una Distribución de GNU/Linux orientada a servidores usando estándares abiertos y adaptada a los entornos de las empresas cubanas.

Terminal: forma de acceder al sistema sin utilizar la interfaz gráfica, es decir, realizar todo tipo de tareas en formato texto.

Sistema de archivos: sistema que contiene los archivos los cuales el usuario accede a estos mediante la interfaz de consola.

2.2 Levantamiento de requisitos

Según el estándar 1233 de la IEEE: Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas, un requisito se define como:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Es posible concluir que los requisitos de software son características y funcionalidades que debe poseer un sistema y están enfocados hacia lo que debe hacer el software. Además, pueden ser clasificados en funcionales y no funcionales.

2.2.1 Técnicas para la captura de requisitos

En el proceso de desarrollo de un sistema el equipo de desarrollo siempre se enfrenta al problema de la identificación de requisitos. La definición de estos es un proceso complejo, pues hay que identificar los requisitos que el sistema debe cumplir en orden de satisfacer las necesidades de los usuarios finales y clientes. Para realizar este proceso existen diferentes técnicas, su selección y resultados dependen en gran medida del equipo de desarrollo, como de los propios usuarios o clientes que participen en ellas. Se muestran a continuación las técnicas utilizadas para identificación de los requisitos:

- ✓ **Análisis de sistemas existentes**: Mediante el análisis de sistemas existentes es posible estudiar aplicaciones similares a la que se necesita obtener. Una vez que se tiene la concepción del funcionamiento de un software similar en cuanto a funcionalidades y características es más sencillo identificar los requisitos del sistema que se necesita implementar. Durante la investigación se realizó un estudio de aplicaciones similares a la solución a desarrollar, en las cuales se observaron los diseños de sus interfaces, las funcionalidades que ofrecen, el grado de dificultad a la hora de interactuar con la aplicación, entre otros rasgos importantes que contribuyeran a obtener un producto con la mejor calidad posible (Kotonya, G. y Sommerville, I, 1996).
- ✓ **Entrevista**: se le aplicó al MSc. Yasiel Perez Villazón, especialista involucrado en el proceso de desarrollo de NovaRSAT, lo cual permitió obtener información con la mayor calidad posible e interpretar las necesidades del cliente, identificando así los requisitos funcionales a los que debe responder el módulo. (Ver Anexo#1)

2.2.2 Requisitos funcionales (RF)

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, cómo debe comportarse en situaciones específicas. En algunos casos también pueden plantear explícitamente qué no debe hacer el sistema (Ian Somerville, 2005). A partir de lo antes planteado se definieron los siguientes requisitos funcionales:

:Tabla 1. Descripción de los requisitos funcionales

(Fuente: Elaboración propia)

No.	Requisito Funcional	Descripción	Complejidad	Prioridad para el cliente
RF_1	Crear carpeta.	La aplicación deberá permitir al usuario crear una carpeta la cual inicialmente se crea vacía y con el nombre de “Nueva Carpeta”.	Alta	Alta
RF_2	Crear archivo.	La aplicación deberá permitir al usuario crear un archivo el cual puede ser un texto, un dibujo, un video etc.	Alta	Alta
RF_3	Cambiar el nombre de una carpeta o archivo	La aplicación deberá permitir al usuario renombrar una carpeta o un archivo.	Media	Alta
RF_4	Seleccionar carpetas o archivos.	La aplicación deberá permitir al usuario seleccionar una o varias carpetas, así como uno o varios archivos.	Media	Alta
RF_5	Mover carpetas o archivos.	La aplicación deberá permitir al usuario mover una o varias carpetas, así como uno o varios archivos.	Alta	Alta
RF_6	Copiar carpetas o archivos.	La aplicación deberá permitir al usuario copiar una o varias carpetas, así como uno o varios archivos.	Alta	Alta
RF_7	Eliminar carpetas o archivos.	La aplicación deberá permitir al usuario eliminar carpetas o archivos de forma permanente.	Alta	Alta

2.2.3 Requisitos no funcionales (RNF)

Los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades de este como fiabilidad, tiempo de respuesta y la capacidad de almacenamiento. Incluyen además restricciones de tiempo, sobre el proceso de desarrollo y estándares (SOMMERVILLE, 2011). A continuación, se definen los requisitos no funcionales que debe cumplir la aplicación basándose en los establecido por las normas ISO 25000 Calidad del Producto de Software, específicamente la ISO/IEC 25010 que define las características de calidad que se tienen en cuenta al evaluar las propiedades de un producto de software:

Confiabilidad

RNF1- La aplicación debe ser capaz de manejar los errores y recuperarse.

Usabilidad

RNF2- El sistema debe representar un diseño sencillo, con una interfaz agradable para el cliente y fácil de operar.

Portabilidad

RNF5- La aplicación deberá funcionar en todas las versiones de NovaRSAT.

2.3.4 Historias de usuario

Las historias de usuario (HU) constituyen una forma de administración de requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario son cortas descripciones de una funcionalidad desde la perspectiva de la persona que la desea, usualmente un usuario o cliente Las mismas son escritas utilizando el lenguaje común. Son empleadas en las metodologías de desarrollo ágiles para la especificación de requisitos (COHN, 2018).

En correspondencia con la selección del escenario número cuatro de la metodología empleada se procede a modelar el sistema con historias de usuario, donde se define una por cada requisito funcional. Se muestran a continuación las HU “Crear Carpeta”, “Copiar carpetas y archivos” y “Eliminar carpetas o archivos”.

Tabla 2. Historia de usuario “Crear Carpeta”.

(Fuente: Elaboración propia)

Número: HU – 1	**Nombre del requisito: Crear Carpeta.**
Programador: Neysa Martínez Acosta	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3 semanas
Riesgo en Desarrollo: Medio	**Tiempo Real:** 1 semana
Descripción: Luego de ejecutar la aplicación esta permite al usuario crear carpetas. Se listan a continuación las funcionalidades requeridas: • Permitir al usuario navegar hasta la ubicación donde desea crear la carpeta.	

- Permitir al usuario elegir la opción de “Nueva carpeta”.
- Permitir al usuario introducir el nombre de la carpeta.
- Listar la nueva carpeta.

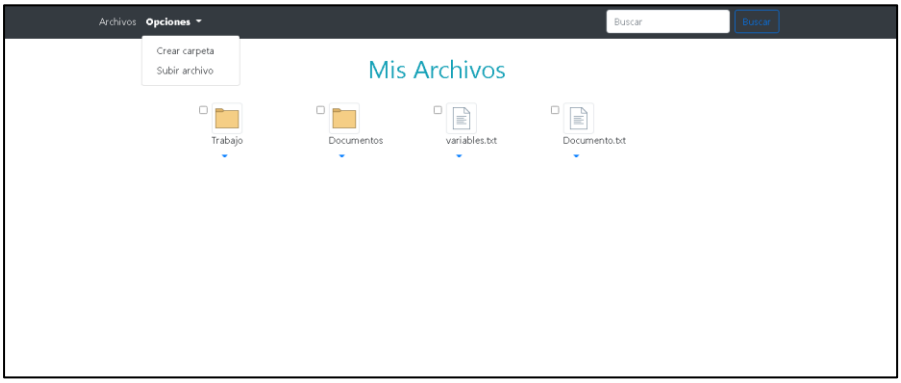


Figura 10 Prototipo interfaz de usuario ¨ Crear carpeta¨

Luego de seleccionar la opción de crear carpeta.

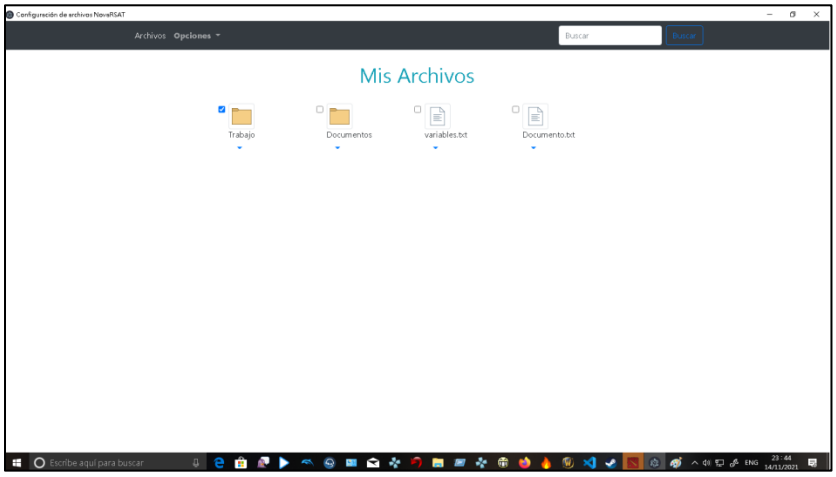


Figura 10 Prototipo interfaz de usuario ¨ Crear carpeta¨

(Fuente: Elaboración propia)

Tabla 3. Historia de usuario “Copiar carpetas o archivos”.

(Fuente: Elaboración propia)

Número: HU – 6	**Nombre del requisito: Copiar carpetas o archivos.**
Programador: Neysa Martínez Acosta	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:**2
Riesgo en Desarrollo: Medio	**Tiempo Real:** 2

Descripción: Luego de ejecutar la aplicación esta permite al usuario copiar carpetas o archivos desde una ubicación de origen a una ubicación de destino.

Se listan a continuación las funcionalidades requeridas:

- Permitir al usuario navegar hasta la ubicación de donde desea copiar la carpeta o el archivo.
- Posibilitar al usuario seleccionar la carpeta o el archivo que desea copiar en otra ubicación.
- Permitir mostrar la opción de copiar al dar clic derecho encima de la carpeta que se desea copiar.
- Permitir al usuario elegir en el menú la opción “Copiar”.
- Permitir al usuario navegar hasta la unidad de almacenamiento o la carpeta donde desea copiar el elemento seleccionado.
- Permitir al usuario elegir en el menú la opción “Pegar”.

Al seleccionar la carpeta que el usuario desea copiar.

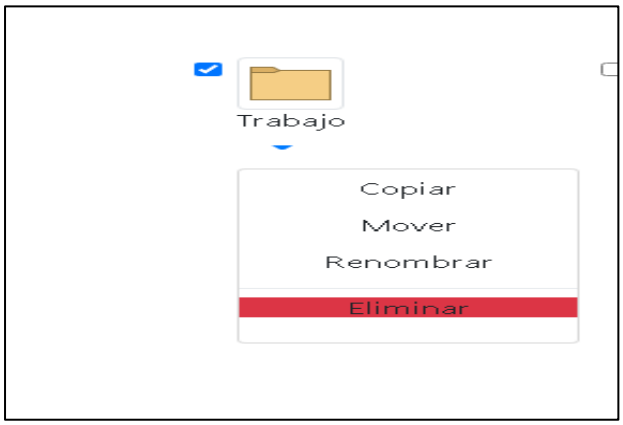


Figura 11 Prototipo de interfaz de usuario ¨Copiar carpeta¨

(Fuente: Elaboración propia)

Técnicas de validación de requisitos

Es muy importante asegurar la validez de los requisitos previamente a comenzar un desarrollo de software. Para ello debe de hacerse una comprobación de la correspondencia entre la descripciones iniciales y si el modelo es capaz de responder al planteamiento inicial. Para llevar a cabo esto, se suele realizar comprobando que el modelo obtenido responde de la misma forma deseada que la que el cliente pide por un lado, y por otro a la inversa si otras respuestas del modelo convencen al cliente. La validación de los requisitos, obviamente tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva. A continuación se hace un análisis de estas técnicas y como fueron aplicadas en la realización del modulo para la configuración del sistema de archivos para NovaRSAT.

Inspecciones del software: analizan y comprueban las representaciones del sistema como el documento de requerimientos, los diagramas de diseño y y el código fuente del programa. Se aplica a todas las etapas del proceso de desarrollo. Las inspecciones se complementan con algún tipo de análisis automático del texto fuente o de los documentos asociados. Las inspecciones del software y los análisis automatizados son técnicas de verificación y validación estáticas puesto que no requieren que el sistema se ejecute. Los requisitos fueron analizados sistemáticamente por el desarrollador y el cliente en busca de anomalías y/u omisiones.

Diseños de casos de prueba: tiene un único objetivo tener la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo posible. Se realizó un caso de prueba para cada requisito, lo que permitió detectar errores o defectos relacionados con las funcionalidades del módulo además de controlar la veracidad del código (SOMMERVILLE, 2011).

2.4 Análisis y diseño

El diseño tiene como principal objetivo la traducción de los requisitos a una especificación que describe cómo implementar el sistema.

Sus objetivos son:

- ✓ Transformar los requisitos al diseño del futuro sistema.
- ✓ Desarrollar una arquitectura para el sistema.
- ✓ Adaptar el diseño para que este sea consistente con el entorno de implementación.

En este método de análisis y diseño se crea un conjunto de modelos utilizando una notación acordada como el Lenguaje Unificado de Modelado (UML). El diseño es el lugar en el que se

establece la calidad del software (Pressman 2005a). Esta actividad del ciclo de vida de ingeniería de software debe describir la arquitectura de construcción (Luca Crisan, 2019).

2.4.1 Descripción de la arquitectura

El diseño arquitectónico es un proceso creativo en el que se intenta establecer una organización del sistema que satisfaga los requisitos funcionales y no funcionales del propio sistema. Debido a que es un proceso creativo, las actividades dentro del proceso difieren radicalmente dependiendo del tipo de sistema a desarrollar, el conocimiento y la experiencia del arquitecto del sistema, y los requisitos específicos del mismo (Sommerville 2005).

Es decir, la arquitectura de software es una forma de representar sistemas mediante el uso de la abstracción, de forma que aporte el más alto nivel de comprensión de los mismos. Esta representación incluye los componentes fundamentales del software, su comportamiento y formas de interacción para satisfacer los requisitos del sistema. Según Roger Pressman: “En su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan” (Pressman, 2005a).

Para el desarrollo de la solución propuesta en el presente trabajo de diploma se propone una arquitectura Modelo-Vista-Controlador (MVC, Model-View-Controler) teniendo en cuenta principalmente su bien definida separación de los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

Al utilizar el framework Electron se realiza un diseño orientado a web por la posibilidad que brida este permite el desarrollo de aplicaciones enriquecidas de escritorio mediante el uso de tecnologías web

Se presenta a continuación una vista lógica de esta arquitectura utilizando el diagrama de paquetes proporcionado por UML.

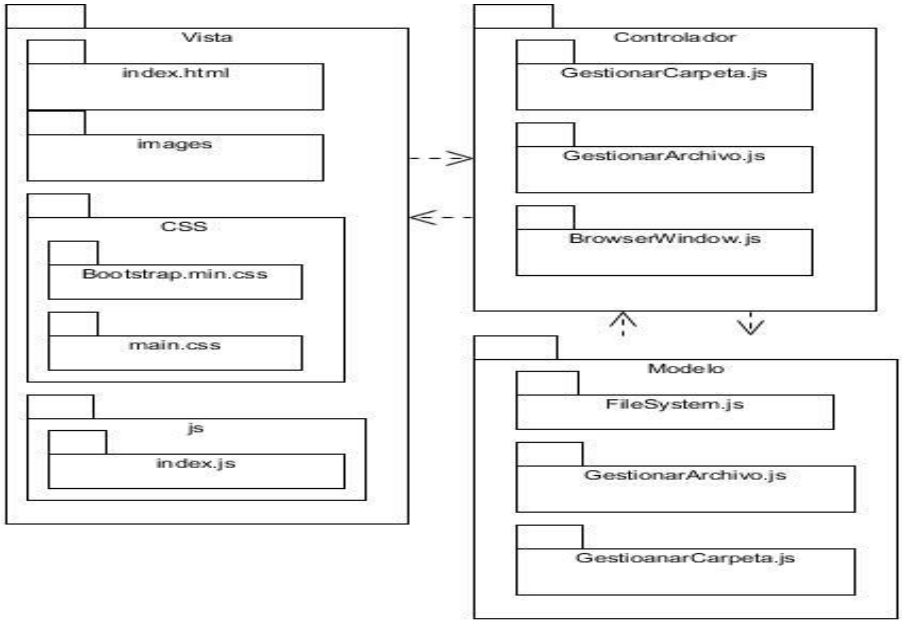


Figura 12. Vista lógica de la arquitectura MVC

(Fuente: Elaboración propia).

Como se observa en la figura el sistema se estructura en tres componentes que interactúan entre sí. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, sobre multitud de lenguajes y plataformas de desarrollo.

- El Modelo que contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia.
- La Vista, o la interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con este.
- El Controlador que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Diagrama de clases de diseño

El diagrama de clases con estereotipos web o diagrama de clases del diseño muestra la especificación para las clases de software de la aplicación. Incluye información como clases, asociaciones, atributos, interfaces con sus operaciones y constantes, métodos, dependencias, muestra definiciones de entidades de software más que conceptos del mundo real, se añaden los detalles referentes al lenguaje de programación que se vaya a usar.

Para la representación se utilizan estereotipos tales como: las *Server Page* las cuales permiten construir y controlar la información de las páginas clientes llamadas *Client Page* y estas a su vez estas muestran información en el cliente manejándolos datos del formulario que contiene toda la información. Las Clases Controladoras controlan todo el funcionamiento de la aplicación y las Clases Entidades que contienen todos los datos necesarios de las entidades existentes. A continuación se representa el diagrama de clases de diseño con estereotipos web realizado.

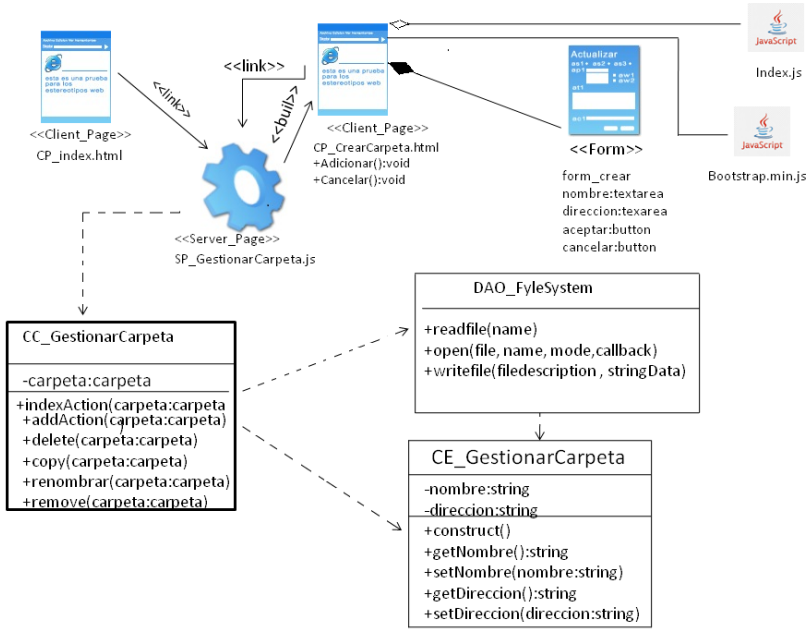


Figura 13. Diagrama de clases del diseño

(Fuente: Elaboración Propia)

2.5 Patrones de diseño

Los patrones de diseño están relacionados con el diseño de los objetos y frameworks de pequeña y mediana escala. Aplicables al diseño de una solución para conectar los elementos de gran escala que se definen mediante los patrones de arquitectura, y durante el trabajo de diseño detallado para cualquier aspecto del diseño local (Larman 2003). Estos refinan los componentes basados en la experiencia obtenida. Son una solución estándar para un problema común de programación, ya que ayudan a mantener un código reutilizable y tener mayor control sobre los problemas recurrentes, por eso, son muy utilizados en el desarrollo de en múltiples aplicaciones (Larman 2003).

Patrones GRASP

Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a uno a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades (Larman y Applying, 2004).

Experto: se usa más que cualquier otro al asignar responsabilidades, es un principio básico que suele utilizarse en el diseño orientado a objetos (Larman y Applying 2004). Consiste en la asignación de una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo. El uso de este patrón da pie a un bajo acoplamiento y una alta cohesión, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos

Controlador: Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. El mismo asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente el “sistema” global.

Este patrón se pone de manifiesto en todo el módulo debido a que cada uno de los eventos generados por el usuario es redirigido a una clase controladora que realiza las operaciones solicitadas, manteniendo siempre la alta cohesión.

Bajo acoplamiento: Este patrón es una medida de la fuerza con que una clase está conectada a otras, las conoce y recurre a ellas (Larman, 2004). En el diseño de la propuesta de solución existen clases con un bajo acoplamiento debido a que no dependen de muchas otras clases; una clase solo depende de otra, como máximo, para realizar sus funciones.

Alta cohesión: el patrón de alta cohesión asigna responsabilidades de manera que una clase no tenga muchas funcionalidades no relacionadas o no realice un trabajo excesivo (Larman, 2004). Este patrón se evidencia debido a la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. A cada una de las clases se le asignaron responsabilidades de tal forma, que están estrechamente relacionadas entre sí y no realizan un trabajo excesivo.

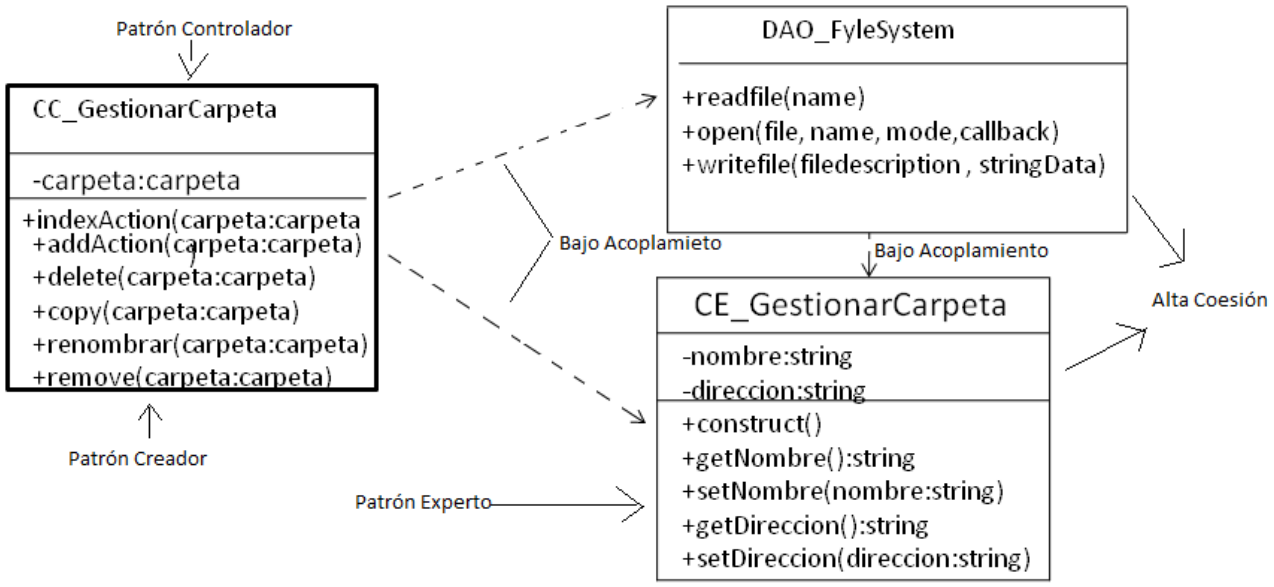


Figura 14 Representación de los patrones de diseños utilizados

Fuente: Elaboración propia

2.5.1 Conclusiones parciales

A partir del desarrollo del presente capítulo se arribó a las siguientes conclusiones:

- La captura de los requisitos y la elaboración de las historias de usuario correspondientes permitió comprender mejor las funcionalidades de la aplicación que se desea desarrollar y el comportamiento de la misma.
- La utilización de la arquitectura MVC y los patrones de diseño contribuyó al diseño de la aplicación, proporcionando una estructura para la misma y posibilitando el empleo de buenas prácticas de programación y la reutilización de código.
- Como resultado principal de este capítulo quedó plasmada la propuesta de solución para la problemática planteada, detallada a través de los requisitos funcionales y no funcionales, las historias de usuario, la arquitectura y el uso de patrones de diseño.

CAPÍTULO lll. Implementación y prueba del módulo para la configuración del sistema de archivos en NovaRSAT.

3.1. Introducción

La disciplina de implementación en el desarrollo de un producto de software, es el mecanismo donde se ponen en práctica todas las descripciones y arquitecturas propuestas en las fases de análisis y diseño, es el complemento del trabajo de las fases que lo preceden dentro del proceso de desarrollo de software. Para un despliegue exitoso del navegador primero este debe pasar por un conjunto de pruebas las cuales permiten validar el correcto funcionamiento. En el presente capítulo se exponen las especificaciones asociadas a la implementación del módulo, se describen los estándares de codificación empleados, como también se describe el diseño de las pruebas realizadas a la aplicación y los resultados obtenidos durante cada iteración. Además, se explican de manera breve los resultados arrojados por las pruebas realizadas a las diferentes funcionalidades.

3.2 Implementación del sistema

Luego del resultado obtenido del análisis y diseño se da paso a la implementación del módulo, logrando concebir la arquitectura y el sistema como un todo. A continuación se describen los estándares de codificación que se utilizan en el desarrollo del módulo para la configuración de sistema de archivos para NovaRSAT.

Diagrama de componentes

Los diagramas de componentes UML representan las relaciones entre los componentes individuales del sistema mediante una vista de diseño estática. Pueden ilustrar aspectos de modelado lógico y físico. En el contexto del UML, los componentes son partes modulares de un sistema independientes entre sí, que pueden reemplazarse con componentes equivalentes. Son auto contenidos y encapsulan estructuras de cualquier grado de complejidad. Los elementos encapsulados solo se comunican con los otros a través de interfaces. Los componentes no solo pueden proporcionar sus propias interfaces, sino que también pueden utilizar las interfaces de otros componentes, por ejemplo, para acceder a sus funciones y servicios. A su vez, las interfaces de un diagrama de componentes documentan las relaciones y dependencias en una arquitectura de software.

Un diagrama de componentes proporciona una visión general del sistema y documenta la organización de los componentes del sistema y sus relaciones y dependencias mutuas. Los diagramas de componentes proporcionan una visión orientada a la ejecución, es decir, dan al

desarrollador información sobre si un sistema funciona de forma coherente y cumple sus tareas y objetivos (Guide Digital, 2019).

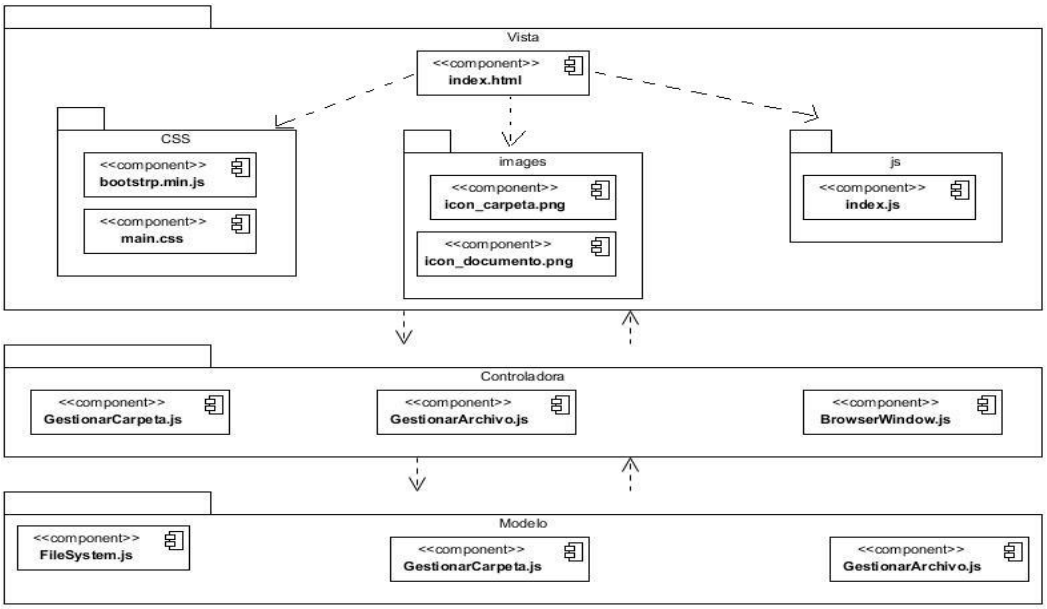


Figura 15 Diagrama de componentes

(Fuente: Elaboración Propia)

Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez(Arevalo Lizardo, 2012).

Para facilitar el entendimiento del código y fijar un modelo a seguir, se establecieron estándares de codificación. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

Reglas a seguir para el trabajo con las clases del módulo

- Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con mayúsculas. Los nombres serán simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa (URL, HTTP, entre otras.).
- Los nombres de las clases se escriben con la primera letra de cada palabra que lo compone en mayúscula
- Se utilizan los comentarios en caso de ser necesario para explicar la utilidad de la clase.

Reglas a seguir para nombrar las variables y métodos

- Los nombres deben ser descriptivos y concisos. No usar grandes frases y solo abreviaturas pequeñas.
- La palabra reservada var solamente se debe indicar al definir por primera vez la variable (lo que se denomina declarar una variable).
- Los nombres deben ser descriptivos y concisos. No usar grandes frases y solo abreviaturas pequeñas.
- Al definir una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
- El identificador para las variables y los parámetros serán con letras en minúsculas y en caso de ser un nombre compuesto se divide cada palabra con mayúscula inicial.
- Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas.

Reglas a seguir para la declaración de clases / interfaces

- Durante el desarrollo de clases / interfaces se deben seguir las siguientes reglas de formateo: No incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.
- El carácter inicio de bloque ("{") debe aparecer al final de la línea que contiene la sentencia de declaración.
- El carácter fin de bloque ("}") se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto cuando la sentencia sea nula, en tal caso se situará detrás de "{".

- Los métodos se separarán entre sí mediante una línea en blanco.

```
const { app, BrowserWindow } = require("electron");
const path = require("path");

function createWindow() {
  const win = new BrowserWindow({
    width: 800,
    height: 600,
    autoHideMenuBar: true,
    webPreferences: {
      preload: path.join(__dirname, "preload.js"),
    },
  });

  win.loadFile("src/views/index2.html");
}

app.whenReady().then(() => {
  createWindow();

  app.on("activate", () => {
    if (BrowserWindow.getAllWindows().length === 0) {
      createWindow();
    }
  });
});
```

Figura 16. Reglas para declarar clases

(Fuente: Elaboración Propia)

Reglas a seguir para los espacios en blanco en expresiones y sentencias

• Evitar utilizar espacios en blanco en las siguientes situaciones:

-Inmediatamente dentro de paréntesis, corchetes y llaves

- Inmediatamente antes de una coma, un punto y coma o dos puntos

- Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función

• Deben rodearse con exactamente un espacio los siguientes operadores binarios:

- Asignación (=)

- Asignación de aumentación (+=, -=, etc.

-Comparación (==, <, >, >=, <=, !=, <>, in, not in, is, is not)

- Expresiones lógicas (and, or, not)

• No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.

Reglas a seguir para comentarios

- Los comentarios deben ser oraciones completas
- Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula
- Si un comentario es corto el punto final puede omitirse
- Los comentarios de una línea para aclaraciones del código aparecerán seguidos de los caracteres "//" en caso de código Javascript.

3.3 Diagrama de despliegue

El modelo de despliegue consiste en un modelo de objetos que tiene como objetivo describir la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre las estaciones de trabajo (denominados nodos). La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Los nodos representan objetos físicos existentes en tiempo de ejecución, sirven para modelar recursos que tiene memoria y capacidad de proceso y puede ser tanto ordenadores como dispositivos, memoria o personas (Sommerville 2011).

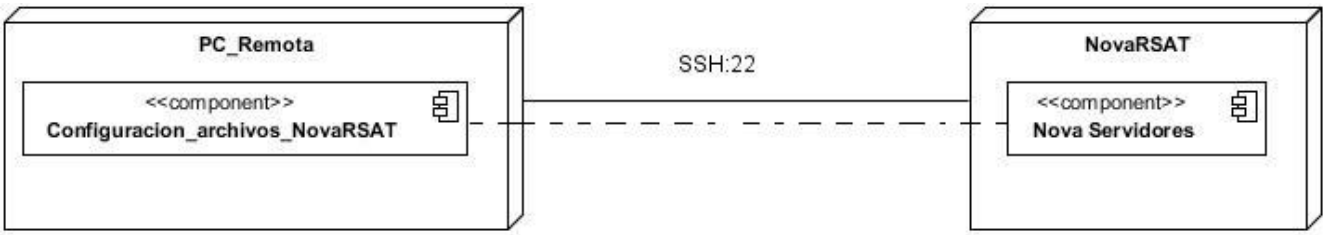


Figura 17. Diagrama de despliegue

(Fuente: Elaboración Propia)

A continuación se muestra una descripción del diagrama de despliegue propuesto para el módulo.

PC Remota: Esta computadora tiene la capacidad de acceder a un ordenador o dispositivo desde cualquier ubicación remota. Esta PC de escritorio contiene el componente (Configuración de archivos para NovaRSAT) el cual es la aplicación de escritorio que permite realizar las configuraciones del sistema de archivos de manera remota utilizando SSH (Secure Shell),en español cubierta segura.

SSH es un protocolo que trabaja como el parámetro de seguridad que procura la administración de los aspectos relacionados a la conexión entre redes**.** Es bien conocido ya que a través de los servidores se realiza una constante transmisión de información desde un computador a otro. Desde SSH se pueden realizar copias seguras de los archivos que se transmiten a través de la dirección seleccionada, en caso de necesitarlas. Por defecto SSH utiliza el protocolo TCP de la capa de transporte, y el número de puerto es el 22.

NovaRSAT: Consiste en una herramienta que permite la configuración remota del servidor

Nova Servidores: Distribución de GNU/Linux orientada a servidores usando estándares abiertos y adaptada a los entornos de las empresas cubanas. Entre sus características se encuentran la configuración fácil e intuitiva a través de la herramienta nova-manager, destinada a la administración de los servicios telemáticos y la compatibilidad con hardware obsoleto en el entorno empresarial.

3.4. Validación de la propuesta de solución

Durante la etapa de implementación, pueden cometerse algunos errores y pueden pasarse por alto algunos elementos que son importantes para el correcto funcionamiento del sistema. Por tal motivo, es esencial llevar a cabo la fase de validación, en la cual, a través de varios tipos y métodos de pruebas de software (estrategia de pruebas), se pretende comprobar el cumplimiento de las especificaciones del diseño y de la codificación, identificar los posibles errores cometidos y validar la solución propuesta en los capítulos anteriores (Labrada, y otros, 2013).

El proceso de validación del módulo para la configuración del sistema de archivos NovaRSAT se realiza mediante la estrategia de pruebas. La estrategia de pruebas de software proporciona un mapa que describe los pasos que se darán para realizarlas, indica cuándo se planea y se darán dichos pasos, además cuánto tiempo, esfuerzo y recursos consumirán. Un software se prueba para descubrir los errores cometidos, si se realiza sin ningún plan seguramente se desperdicia tiempo y esfuerzo.

Pruebas de integración: Abordan los conflictos asociados con los problemas duales de verificación y construcción de programas. Durante la integración, se usan más las técnicas de diseño de casos de prueba que se enfocan en entradas y salidas (Pressman, 2002a).

Prueba funcional: Las pruebas de caja negra o funcional tienen el objetivo de verificar el correcto manejo de las funciones externas provistas o soportadas por el software y que el comportamiento observado se corresponda con las especificaciones del producto y a las expectativas del usuario (Pressman, 2002a).

3.5 Interfaces principales

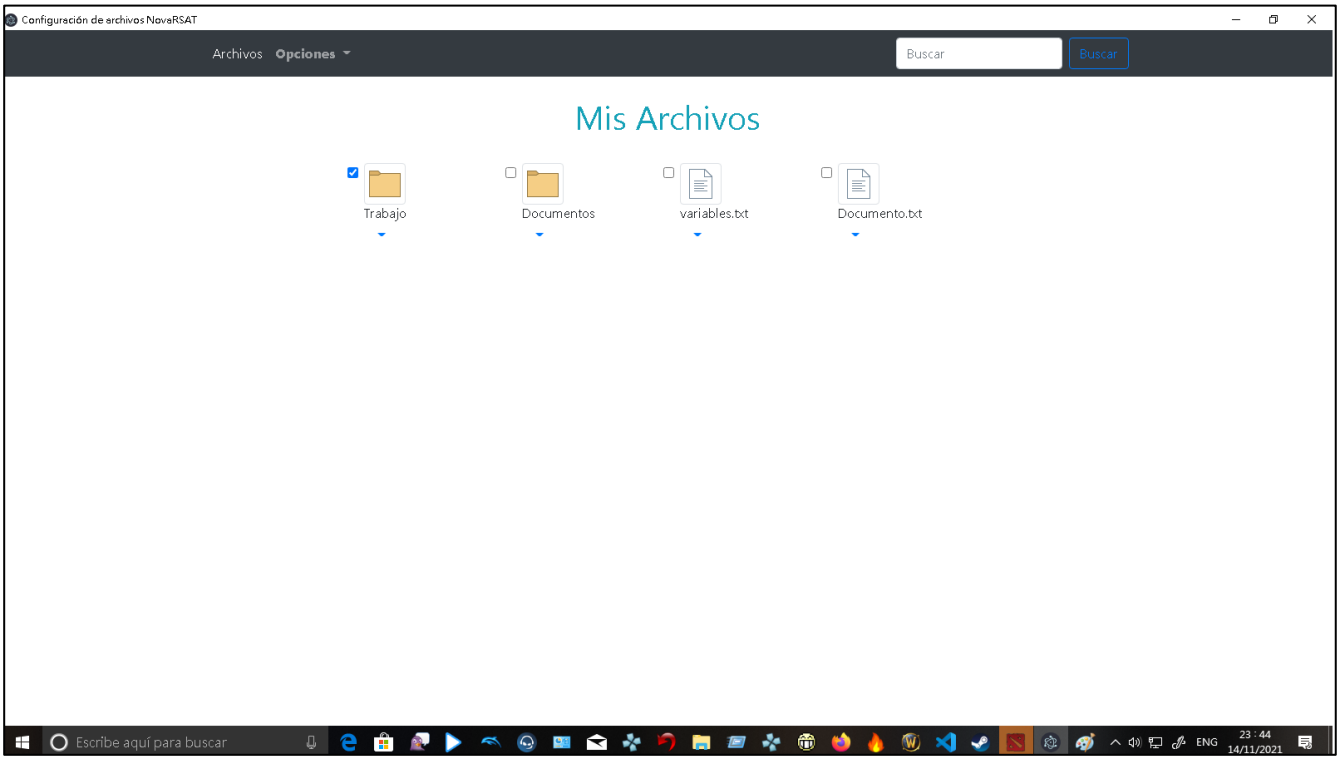


Figura 2. Interfaz principal

(Fuente: Elaboración propia)

3.6 Conclusiones parciales

Durante el desarrollo del capítulo se realizó la validación del sistema a través de los tipos de pruebas realizadas, obteniendo las siguientes conclusiones:

- La implementación de las Historias de Usuario teniendo en cuenta los estándares de codificación, permitió la elaboración de la aplicación la cual responde a las necesidades del cliente.
- Al aplicar los estándares de codificación se logró una mayor legibilidad, limpieza y organización del código.
- Al realizar el diagrama de despliegue se obtuvo una mejor la vista de implementación del sistema.

Conclusiones generales

Con el desarrollo de esta investigación para el desarrollo del módulo para la configuración del sistema de archivos para NovaRSAT, se arriban a las siguientes conclusiones:

El estudio de los conceptos asociados a la investigación, el análisis de las diferentes aplicaciones informáticas que realizan la configuración de sistemas de archivos, el estudio de la herramienta NovaRSAT y la realización del estudio de caso, permitieron aclarecer las características de las funcionalidades del módulo desarrollado.

- El análisis de la situación problemática existente demostró la necesidad de desarrollar un módulo para la configuración del sistema de archivos NovaRSAT.
- La implementación de las Historias de Usuario teniendo en cuenta los estándares de codificación, permitió la elaboración de la aplicación.
- La correcta selección de la metodología, las herramientas y tecnologías para el desarrollo de la solución, permitió la implementación del módulo.

Referencias Bibliográficas

ALEGSA, L., 2016a. Definición de Administrador de archivos (aplicación). [en línea]. [Consulta: 30 de junio 2021]. Disponible en: http://www.alegsa.com.ar/Dic/administrador_de_archivos.php

ALEGSA, L., 2016b. Definición de Directorio (informática). [en línea]. [Consulta: 30 junio 2021]. Disponible en: http://www.alegsa.com.ar/Dic/directorio.php.

ALLER, ANGEL, 2020. Profesional review: Qué es el sistema de archivos: ventajas y desventajas de cada uno. [en línea]. [Consulta: 2 julio 2021]. Disponible en:https://www.profesionalreview.com/2020/06/20/que-es-el-sistema-de-archivos/

AREVALO LIZARDO, M.E., 2012. Propuesta de Estándar de desarrollo o codificación (Primera Entrega) #programacion – Maria Eugenia Arevalo Lizardo. [en línea]. [Consulta: 13 de noviembre de 2021]. Disponible en: https://arevalomaria.wordpress.com/2012/11/02/propuesta-de-estandar-de-desarrollo-o-codificacion-primera-entrega-programacion/.

APUNTES DE ELECTRON | ELECTRON, 2020.Apuntes de Electron [en línea]. [Consulta: 4 de julio 2021]. Disponible en: https://apuntes.de/electronjs/introduccion-al-uso-de-electron/#gsc.tab=0

BELLAS, F.G., UNANUE, R.M. y FERNÁNDEZ, V.D.F., 2016. *Lenguajes de programación y procesadores*. S.l.: Editorial Centro de Estudios Ramon Areces SA.

Brother, 2001 . Preguntas frecuentes y Solución de problemas [en línea]. [Consulta: 4 de julio 2021] Disponible en: https://support.brother.com/g/b/faqend.aspx?c=us&lang=es&prod=ads1200_all&faqid=faq00100531_000

COHN, 2018. Amazon.com: User Stories Applied: For Agile Software Development (Addison-Wesley Signature Series (Beck)) eBook: Mike Cohn: Kindle Store. [en línea]. [Consulta: 29 enero 2021]. Disponible en: https://www.amazon.com/-/es/User-Stories-Applied-Development-Addison-Wesley-ebook-dp-B0054KOL74/dp/B0054KOL74/ref=mt_kindle?_encoding=UTF8&me=&qid=

DEVELOPER GUIDE MOZILA | JAVASCRIPT, 2020.Tecnología para desarrolladores web [en línea]. [Consulta: 4 de julio 2021]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript

Ebrahim, Mokhta (2017). Que es el sistema de archivos de Linux. [en línea]. [Consulta: 30 junio 2021]. Disponible en: https://likegeeks-com.cdn.ampproject.org/v/s/likegeeks.com/es/sistema-de-archivos-de linux/amp/?amp_js_v=a6&amp_gsa=1&usqp=mq331AQKKAFQArABIIACAw%3D%3D#aoh=1625095

0869068&referrer=https%3A%2F%2Fwww.google.com&amp_tf=De%20%251%24s&ampshare=https%3A%2F%2Flikegeeks.com%2Fes%2Fsistema-de-archivos-de-linux%2F

GUIDE DIGITAL,2019. Digital Guide: Los mejores gestores de archivos para Windows, Linux y Mac [en línea]. [Consulta: 2 julio 2021]. Disponible en: https://www.ionos.es/digitalguide/servidores/herrramientas/gestores-de-archivos-para-windows-linux-y-mac/

IAN SOMERVILLE, 2005. *Ingeniería de Software*. S.l.: s.n.

Juncos, Raúl (21 de enero de 2008). Sistema de ficheros GNU/Linux» (html). Ministerio de Educación, Cultura y Deporte (España) [en línea]. [Consulta: 30 junio 2021]. Disponible en: https://web.archive.org/web/20081214104329/http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=549

Kotonya, G. y Sommerville, I. (1996). "Requirements Engineering with viewpoints". BCS/IEE Software Engineering J [en línea]. [Consulta: 30 junio 2021]. Disponible en: https://digital-library.theiet.org/content/journals/10.1049/sej.1996.0002

LARDINOIS, Frederic. Microsoft shocks the world with visual studio code a free code editor for OSX Linux and windows. [En línea]. 2015. [Consultado el: 14 de noviembre de 2021]. Disponible en: [https://www.techcrunch.com/2016/04/14/microsoft-visual-studio-code-for-os-x-linux-and-windows]

Labrada, Evelyn y Aragón, Yaniel. 2013.*Desarrollo del Módulo de Gestión de Reportes Estadísticos para el sistema AiresProxyAudit.* La Habana : s.n., 2013

LARMAN, C., 2003. *UML y PATRONES. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. (Begoña, M., Trad.)*. S.l.: Madrid: Pearson Educación, SA (Original en inglés publicado en 2002).

LARMAN, C. y APPLYING, U.M.L., 2004.*Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development , Chapter 6*. S.l.: Prentice Hall. October.

LARMAN, Craig. UML y patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Segunda edición. 1999. México. Prentice Hall. ISBN 970-17-0261-1. p.164-238

LUCA CRISAN, A., 2019. *A study of Kotlin's: conciseness, safety and interoperability.*B.S. thesis. S.l.: UniversitatPolitècnica de Catalunya.

MASTER MAGAZINE. Definición de Herramienta- Significado y definición de herramienta. [En línea].

2016. [Consultado el: 11 de octubre 2021]. Disponible en: [https://www.mastermagazine.info/termino/5234.php]

OSNAYA, E., 2012. Administración de Archivos. Administración de Archivos -eosnaya [en línea]. [Consulta: 20 de junio 2021]. Disponible en: https://eosnaya.wordpress.com/1-conceptos-basicos-de-archivos/.

PRESSMAN, R.S., 2005a. *Software engineering: a practitioner's approach*. S.l.: Palgravemacmillan.

Pressman, R. S. (2002a). *Ingeniería de software: Un enfoque práctico*

Pressman, R. S. (2002b). *Ingeniería de software: Un enfoque práctico 5ta edicion* (M. Hill Ed. 5ta ed.). Nueva York, Estados Unidos.

RAMÍREZ, F.J., 2018. Metodologías para el desarrollo de software. [en línea], [Consulta: 4 Julio 2021]. Disponible en: https://www.academia.edu/9953322/Metodologias_para_el_desarrollo_de_software.

RAFFINO, M.E., 2019. Archivo en Informática: Concepto, Características y Formato. [en línea]. [Consulta: 25 junio 2021]. Disponible en: https://concepto.de/archivo-informatico/.

SOMMERVILLE, I., 2011. SOMMERVILLE, I., 2011. Ingeniería del Software. 9na. Mexico: Addison Wesley. Pearson Education, Inc. ISBN 978-607-32-0603-7 - Buscar con Google. [en línea]. [Consulta: 28 octubre 2021]. Disponible en: https://www.google.com/search?source=hp&ei=IdIwXo70N82E5wK58rCQDw&q=SOMMERVILLE%2C+I.%2C+2011.+Ingenier%C3%ADa+del+Software.+9na.+Mexico%3A+Addison+Wesley.+Pearson+Education%2C+Inc.+ISBN+978-607-32-0603-7&oq=SOMMERVILLE%2C+I.%2C+2011.+Ingenier%C3%ADa+del+Software.+9na.+Mexico%3A+Addison+Wesley.+Pearson+Education%2C+Inc.+ISBN+978-607-32-0603-7&gs_l=psy-ab.3...9870.9870..10741...0.0..0.0.0.....0....2j1..gws-wiz.divSIJBALvo&ved=0ahUKEwjOiPiSx6fnAhVNwlkKHTk5DPIQ4dUDCAU&uact=5.

SOMMERVILLE, I., 2005. *Ingeniería del software*. S.l.: Pearson educación.

STAFF, ALDEAHOST, 2020. AldeaHost: Que es un directorio en informática y para qué sirve [en línea]. [Consulta: 2 julio 2021]. Disponible en: https://aldeahost.com.mx/ que-es-un-directorio-en-informatica/

TECNOLOGICON, 2015. Definición De Directorio (Informática). Tecnologicon [en línea]. [Consulta: 25 junio 2021]. Disponible en: https://tecnologicon.com/definicion-de-directorio-informatica/.

WHAT IS UNIFIED MODELING LANGUAGE (UML)? 2017. What is Unified Modeling Language (UML)? [enlínea]. [Consulta: 4 julio2021]. Disponible en: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/. VISUAL PARADIGM FREQUENTLY ASKED QUESTIONS, 2017. Visual Paradigm Frequently Asked Questions. [en línea]. [Consulta: 4 julio 2021]. Disponible en: https://www.visual-paradigm.com/support/faq.jsp.

Anexos

Anexo #1. Entrevista realizada al MSc. Yasiel Pérez Villazón

Estimado(a) compañero(a):

La presente entrevista tiene como objetivo conocer, analizar cómo se realiza la configuración de archivos en NovaRSAT. El resultado de esta entrevista resulta de vital importancia para la presente investigación debido a que contribuirá a la correcta elaboración de la propuesta solución.

Datos generales del entrevistado:

Título universitario: _____

Categoría científica: _____

Categoría docente: _____

Instrucciones:

1. ¿ A que tipo de clientes (usuarios) va dirigido el producto NovaRSAT?. ¿ En que entorno es desplegado NovaRSAT?

_____

2. ¿Que servicios ofrece NovaRSAT?

_____

3. ¿Que otras características considera que debe presentar el sistema, en cuanto a la usabilidad, seguridad, interfaz u otro aspecto que garantice su calidad?

_____

Anexo # 2. Historia de usuario : Eliminar carpetas o archivos

Número: HU – 7	**Nombre del requisito: Eliminar carpetas o archivos.**
Programador: Neysa Martínez Acosta	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:**3
Riesgo en Desarrollo: Medio	**Tiempo Real:** 2

Descripción: Luego de ejecutar la aplicación esta permite al usuario eliminar carpetas o archivos del dispositivo de forma momentánea.

Se listan a continuación las funcionalidades requeridas:

- Permitir al usuario navegar hasta la ubicación de donde desea eliminar la carpeta o el archivo.
- Posibilitar al usuario seleccionar la carpeta o el archivo que desea eliminar.
- Permitir dar clic derecho encima de la carpeta que se desea eliminar
- Listar opción de eliminar
- Permitir al usuario elegir en el menú la opción “Eliminar”.
- Mostrar una caja de diálogo para confirmar la acción, esta confirmación significa una eliminación momentánea para los elementos del disco duro.

Luego de seleccionar la carpeta y hacer clic en la opción eliminar.