Universidad de las Ciencias Informáticas

Facultad 1

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

Título:

Herramienta de gestión de copias de seguridad.

Autor:

Mauricio Leal Aguilar

Tutores:

Msc. Máxora Rorayma Castro Pérez

Ing. Asist. Gladys Marsi Peñalver Romero

Ing. Manuel Enrique Peiso Cruz

Declaración Jurada de Autoría

Declaro por este medio que yo, Mauricio Leal Aguilar, con carné de identidad 94081428564, soy el autor principal del presente Trabajo de Diploma “Herramienta de gestión de copias de seguridad” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

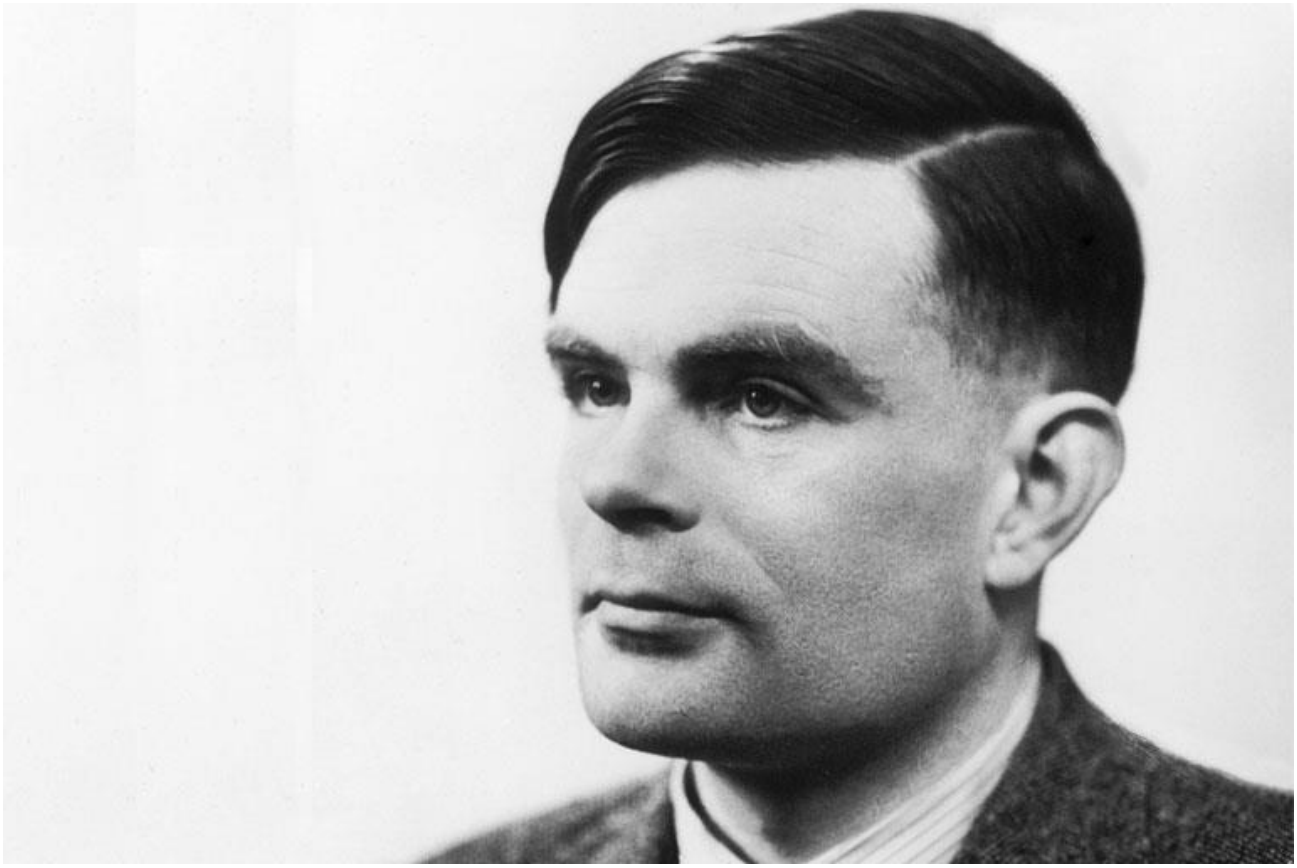
Para que así conste, firman la presente declaración jurada de autoría en La Habana a los ___ días del mes de _____ del año _____.

Mauricio Leal Aguilar

_____

Firma del autor

MCs. Máxora Castro Pérez	**Ing.Asist. Gladys Peñalver Romero**	**Ing. Manuel Peiso Cruz**
_____	_____	_____
Firma de la tutora	Firma de la tutora	Firma del tutor



"Una computadora puede ser llamada inteligente si logra engañar a una persona haciéndole creer que es un humano"
Alan Mathison Turing

Dedicatoria:

A mi familia por el sacrificio que ha hecho para dibujar ante mis ojos la mejor de todas las realidades.

Agradecimientos:

A mi mamá, a mi súper tía Karelia, a la mejor de las abuelas mi abuela Mercedes, a mi papá Leopoldo, a mi papá Jorge Luis, a mis hermanas Idalmis, Karla, Yayi, Mida y Xami.

A mi tío Armando, a mi tío Abdel, a mi tío Gustavo.

A mi eterno amigo, cómplice y consejero Amauri y su esposa Lídice, a mi abuela Myrna y en especial a Grettel por haber compartido a mi lado más que nadie estos duros tiempos.

A mis tutores, por nunca darme la espalda en este camino y sin ustedes no hubiese sido posible lograrlo.

A mis hermanos del Bakano: Juanca, Amado, Carlitos, Ramón, Papi Rauli, Michel, Manu, Llillo, Vlado, Osvaldo, Diwel y David. Son mi nueva familia.

Resumen

Durante el uso y desarrollo de la Distribución Cubana de GNU/Linux Nova, esta puede quedar inoperable o tener un mal funcionamiento debido a errores por mala configuración, modificación de ficheros que forman parte de su funcionamiento, conflicto entre aplicaciones o problemas con el gestor de arranque. En dichos casos la solución es la reinstalación, la cual provoca la pérdida de información y de las configuraciones existentes en el ordenador. Puesto que la distribución no posee una aplicación que permita preservar y restaurar estos datos. El objetivo de la presente investigación se centra en desarrollar una herramienta informática para la gestión de copias de seguridad en la Distribución Cubana de GNU/Linux Nova que influya positivamente en la preservación y restauración de la información. Para cumplir dicho objetivo se empleó Python como lenguaje de programación, PyCharm como entorno de desarrollo integrado y Visual Paradigm como herramienta de modelado. Se obtuvo como resultado el completo desarrollo de la aplicación y su funcionamiento fue evaluado mediante la aplicación de pruebas.

Palabras clave: *gestión, información, pérdida, preservación, restauración*, *Nova* .

Índice general

Índice de tablas

Índice de figuras

Introducción

Dentro del desarrollo tecnológico en el que se encuentra inmersa la sociedad, la información se ha convertido en un elemento importante para cualquier persona, empresa e institución. La disponibilidad es una de sus características fundamentales y su pérdida puede ocasionar grandes problemas, de los cuales es muy difícil recuperarse. Debido a la gran relevancia de la información, en la actualidad las copias de seguridad en los sistemas operativos han pasado a un plano indispensable para el buen funcionamiento de cualquier tipo de empresa, sin tener en cuenta sus dimensiones o perfil de negocio; la capacidad de reanudar sus servicios lo antes posible ante cualquier situación de pérdida o corrupción de la información es de vital importancia (1).

En cualquier situación un escenario de pérdida de datos dará lugar a dos resultados posibles: o bien algunos datos son recuperables o se borran definitivamente. En el entorno actual, con la existencia de numerosas aplicaciones para la preservación de información y las distintas soluciones de recuperación, los usuarios no tienen que presentar situaciones donde los de datos no se puedan recuperar (2).

Una copia de seguridad, funciona como una captura de la estructura y archivos del disco duro de las computadoras, tal como existían cuando se creó, incluyen la configuración y el estado de las preferencias de los usuarios, programas, librerías y demás archivos (3). Su utilidad, es poder devolver a un estado óptimo a los sistemas operativos en caso de que ocurra un percance (4).

Con el propósito de contribuir a la soberanía tecnológica en Cuba, se encuentra en la Universidad de las Ciencias Informáticas (UCI), el Centro de Software Libre (CESOL); encargado del desarrollo de la Distribución Cubana de GNU/Linux Nova en el departamento Sistemas Operativos. Para su funcionamiento utiliza el núcleo de Linux e incluye determinados paquetes de aplicaciones informáticas para satisfacer las necesidades de la migración a plataformas de código abierto que experimenta el país como parte del proceso de informatización de la sociedad. GNU/Linux Nova es la distribución recomendada para ser utilizada como tecnología base para el despliegue y desarrollo de aplicaciones informáticas en los organismos de la administración central del estado (5).

El uso de la Distribución Cubana de GNU/Linux Nova ofrece a los usuarios la posibilidad de contar con una distribución personalizable y a la medida de sus necesidades. Para ello cuenta con un equipo de soporte profesional capacitado y al alcance de la mano; ofreciendo soluciones a las instituciones cubanas, teniendo

en cuenta sus características tecnológicas. Su uso y productos agregados elevan notablemente los niveles de seguridad, soberanía tecnológica, socio-adaptabilidad y sostenibilidad de la informatización de la sociedad cubana actual (5).

Sin embargo, como en toda distribución, el usuario común de la Distribución Cubana de GNU/Linux Nova se encuentra expuesto a diversas situaciones que ocasionarían la reinstalación, citando como ejemplos el borrado accidental de archivos, la eliminación de particiones, el formateo eventual, averías en el sistema de arranque, errores en las aplicaciones y corrupción de datos, por solo mencionar algunas. Cuando la distribución es reinstalada se pierde el entorno de trabajo del usuario; desapareciendo las aplicaciones con sus respectivas configuraciones y la información contenida en la partición que aloja al sistema.

El equipo de desarrollo y personalización de la Distribución Cubana de GNU/Linux Nova desempeña una serie de pruebas que tienen como objetivo demostrar la adaptabilidad y correcta ejecución de este software. En dichas pruebas se realizan cambios importantes en los componentes de la distribución, y en caso de salir defectuosas pueden dañar su funcionamiento, creando conflictos entre aplicaciones o la modificación de archivos esenciales. Dejando como única alternativa a los desarrolladores la reinstalación.

En las versiones anteriores de la Distribución Cubana de GNU/Linux Nova se realizan copias de seguridad con la herramienta rsync que carece de una interfaz gráfica y el trabajo en la misma se hace mediante la consola por medio de comandos. La herramienta permite almacenar todas las configuraciones del usuario, pero no permite la restauración del sistema lo que la hace infructuosa para devolver la información a un estado anterior y de igual manera los usuarios se ven obligados a reinstalar.

Según la problemática descrita, se plantea como **problema científico** de la investigación: ¿Cómo gestionar copias de seguridad en la Distribución Cubana de GNU/Linux Nova que contribuyan a la preservación y restauración de la información?

Como **objeto de estudio** se define el proceso de gestión de copias de seguridad en las distribuciones de GNU/Linux y como **campo de acción** el proceso de gestión de copias de seguridad para la Distribución Cubana de GNU/Linux Nova.

El **objetivo general** de la investigación es desarrollar una herramienta informática para la gestión de copias de seguridad en la Distribución Cubana GNU/Linux Nova que contribuya a la preservación y restauración de la información.

Como **objetivos específicos** fueron establecidos:

- Elaborar el marco teórico de la investigación acerca de las herramientas informáticas de gestión de copias de seguridad.
- Diseñar una herramienta informática para la gestión de copias de seguridad en la Distribución Cubana de GNU/Linux Nova.
- Implementar una herramienta informática para la gestión de copias de seguridad en la Distribución Cubana de GNU/Linux Nova.
- Evaluar el correcto funcionamiento de la solución para la gestión de copias de seguridad en la Distribución Cubana GNU/Linux Nova.

Al presente trabajo se le atribuye como **idea a defender** que el desarrollo de una herramienta informática para la gestión de copias de seguridad en la Distribución Cubana GNU/Linux Nova, puede contribuir a garantizar la preservación y restauración de la información de los usuarios.

Para cumplir con los objetivos mencionados se hace necesaria la utilización de los siguientes **métodos de investigación:**

Métodos teóricos:

Analítico-Sintético: consiste en la extracción de las partes de un todo, con el objetivo de estudiarlas y examinarlas por separado para luego sintetizarlas y tomar los elementos más importantes. Se utilizó en toda la realización de la investigación, para analizar la bibliografía referente al objeto de estudio y sintetizarla en sus elementos fundamentales.

Métodos empíricos:

Entrevista: la técnica entrevista propició la obtención de información a partir de la experiencia de los especialistas involucrados en el proceso de desarrollo de la Distribución Cubana de GNU/Linux Nova sobre cómo se maneja el proceso de gestión de copias de seguridad hoy en día y como fuente para la obtención de los requisitos principales con los que debe contar una herramienta de este tipo.

Estructura del documento:

El presente documento se desglosa, en resumen, introducción, tres capítulos, conclusiones,

recomendaciones, referencias bibliográficas y anexos.

Capítulo 1: Fundamentación teórica de la investigación

Se realiza un análisis de los elementos teóricos que soportan la investigación. Se definen los conceptos fundamentales para la comprensión de la investigación. Se analizan las soluciones existentes que ofrecen algún tipo de solución al problema haciendo especial énfasis en la forma en que se desarrollan. Además, se establecen las tecnologías y la metodología para el desarrollo de software.

Capítulo 2: Análisis y diseño de la herramienta de gestión de copias de seguridad para la Distribución Cubana de GNU/Linux Nova.

Se realiza el análisis y diseño de la propuesta de solución mediante la definición de los requisitos funcionales y no funcionales que debe cumplir el sistema. Se describen a grandes rasgos las características de la solución propuesta a través de la metodología de desarrollo empleada.

Capítulo 3: Implementación y pruebas de la herramienta de gestión de copias de seguridad para la Distribución Cubana de GNU/Linux Nova.

Se realiza la implementación de la herramienta de gestión de copias de seguridad para la Distribución Cubana de GNU/Linux Nova y se especifican las pruebas a realizar a la misma para evaluar su calidad y correcto funcionamiento.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

En el presente capítulo se exponen los fundamentos teóricos asociados al objeto de estudio y al campo de acción y se definen elementos importantes para una mejor comprensión de la investigación. Se realiza un estudio de las principales herramientas informáticas de gestión de copia de seguridad. Se describe la metodología empleada para el proceso de desarrollo del software y se analizan y seleccionan las tecnologías apropiadas para la implementación del sistema.

1.1 Elementos importantes para el contexto de la investigación

Se entiende que una copia de seguridad, es una copia que se realiza de todos los datos y archivos originales de un sistema operativo con el fin de prevenir la pérdida parcial o total de la información del disco duro. Las copias de seguridad se acostumbran a realizar en un soporte de almacenamiento diferente al original como, por ejemplo, en una unidad de almacenamiento externa. De tal manera, en el peor de los casos, no se perderían o dañarían los archivos guardados en el ordenador (6).

1.1.1 *Tipos de copias de seguridad*

Existen diferentes tipos de copias de seguridad según sean las necesidades personales o corporativas, entre ellas:

Copia de seguridad completa: este tipo de copia se realiza en un archivo general, el cual se comprime para ocupar menos espacio y constituye un método de respaldo y recuperación de datos (7).

Copia de seguridad diferencial: luego de realizar una copia de seguridad completa, se puede llevar a cabo una copia de seguridad diferencial, que consiste en copiar los archivos nuevos o las modificaciones de la información ya respaldada, lo que ahorra espacio de almacenamiento y lo hace un método más rápido.

Copias de seguridad incrementales: se asemeja mucho a la copia de seguridad diferencial, pero, en este caso solo se copian los archivos nuevos o las últimas modificaciones partiendo del último respaldo.

Copia de seguridad espejo: se diferencia de la copia de seguridad completa ya que los archivos no son comprimidos ni poseen una clave o contraseña para proteger la información respaldada (8).

En la presente investigación serán realizadas copias de seguridad completas ya que estas permiten preservar la mayor cantidad de datos posibles y permiten la restauración de la información de forma completa. Además, al realizar estas copias en diferentes intervalos de tiempo almacenan varios tipos de

información y brinda al usuario la posibilidad de elegir qué copia desea restaurar atendiendo a sus necesidades.

1.2 Directorios de la Distribución Cubana de GNU/Linux Nova

La Distribución Cubana de GNU/Linux Nova se compone de quince directorios y se hace necesario realizar un estudio sobre los mismos para conocer qué tipo de información almacenan para la realización de la propuesta de solución. Dicha estructura, así como su contenido y funciones**,** viene definida en el denominado *Filesystem Hierarchy Standard* (FHS) que en otras palabras es el estándar de jerarquía para los archivos en sistemas Linux (9).

La jerarquía de directorios según el FHS cuenta con el siguiente sistema de ficheros:

Directorio Raíz o /

La estructura de los directorios de los sistemas basados en UNIX[1] parte de un directorio raíz también llamado directorio *root* y que se simboliza por una barra inclinada /. De tal directorio, es de donde nacen el resto de directorios, independientemente que estén almacenados físicamente en discos o en unidades separadas. Cualquier dirección de archivo o carpeta en Linux empieza por el directorio raíz, seguido de todos los directorios y subdirectorios que lo contienen, separados cada uno de ellos por / (9).

/bin

El directorio */bin* es un directorio estático y es donde se almacenan todos los binarios necesarios para garantizar las funciones básicas a nivel de usuario*. S*olo almacena los ejecutables de usuario, ya que los binarios necesarios para tareas administrativas gestionadas por el usuario *root* o superusuario del sistema se encuentran en el directorio */sbin*. Incluye también los binarios que permiten la ejecución de varias utilidades estándar de la terminal de Linux, concretamente *cat*, *cd*, *cp*, *echo*, *grep*, *gzip*, *kill*, *ls*, *mv*, *rm, ping*, *su*, *ps*, *tar* y *vi* (9).

/boot

Es un directorio estático e incluye todos los ejecutables y archivos que son necesarios en el proceso de arranque del sistema, y que deberán ser utilizados antes que el núcleo empiece a dar las órdenes de

[1] UNIX: Es un sistema operativo portable, multitarea y multiusuario.

ejecución para los diferentes módulos del sistema. Es también donde se encuentra el gestor de arranque *GRand Unified Bootloader* (GRUB) desarrollado por el proyecto GNU, este permite elegir el sistema operativo por el cual arrancar de los que se tengan instalados.

En algunas distribuciones, es común que ese directorio se almacene en su propia partición separada del resto. Esto suele darse sobre todo en el caso de que utilicen *Logical Volume Manager* (LVM), que es la implementación de un administrador de volúmenes lógicos para el núcleo Linux, ya que tradicionalmente el gestor de arranque GRUB (en versiones anteriores a la 2) no podía arrancar desde LVM, por lo que se requería que estuviera en una partición separada.

De hecho, si en una instalación normal de Ubuntu o Debian se opta por utilizar LVM, so nota que el instalador genera un esquema de particiones con el directorio *boot* en una partición aparte**.** En estos casos, en el momento de instalar el sistema es importante prever que el espacio que se le asignará a la partición, ya que, a la larga, con la acumulación de diferentes actualizaciones del núcleo, es común que se quede sin espacio. Si esto sucede, se pueden tener problemas a la hora de instalar futuras actualizaciones del núcleo (9).

/dev

El directorio incluye todos los dispositivos de almacenamiento, en forma de archivos, conectados al sistema, es decir, cualquier disco duro conectado, partición o memoria flash. Siendo así, se observa que la ruta en la que se encuentra cualquier volumen (partición o dispositivo externo) conectado al sistema siempre empieza por */dev* (9). Este es el directorio que contiene, por decirlo de algún modo, la información de cada uno de los volúmenes, a diferencia del directorio ***/****media*, que será analizado más adelante, que contiene solo puntos de montaje, pero no la información real de estos volúmenes.

/etc

Es el encargado de almacenar los archivos de configuración tanto a nivel de componentes del sistema en sí, como de los programas y aplicaciones instaladas a posteriori. Es un directorio que debería contener únicamente ficheros de configuración, y no debería contener binarios (9).

/home

Es el directorio de los usuarios estándar y, por lo tanto, el destinado a almacenar todos los archivos del usuario, como documentos, fotos, vídeos, música, plantillas. También incluye archivos temporales de

aplicaciones ejecutadas en modo usuario, que sirven para guardar las configuraciones de programas. Dentro de */home* se encuentran los directorios personales de todos los usuarios, nombrados según el nombre de usuario utilizado. Así, por ejemplo, si un sistema posee dos usuarios denominados User1 y User2, la estructura será así:

/home/User1

/home/User2

Cada directorio de usuario contiene así mismo diferentes carpetas para ayudarlo a clasificar la información. Generalmente son: */Documentos*, */Imágenes*, */Música*, */Plantillas* y */Vídeos* /, así como otros archivos y carpetas ocultas, que son los encargados de guardar la información de configuraciones de las aplicaciones del usuario (9).

/lib

Incluye las bibliotecas esenciales que son necesarias para que se puedan ejecutar correctamente todos los binarios que se encuentran en los directorios ***/****bin* y ***/****sbin*, así como los módulos del propio núcleo. En los sistemas de 64 bits, además de ***/****lib* existe otro directorio denominado */lib64*, referido a las bibliotecas para aplicaciones de 64 bits (9).

/media

Representa el punto de montaje de todos los volúmenes lógicos que se introducen temporalmente, ya sean unidades externas u otras particiones de disco. En la mayoría de distribuciones GNU/Linux, desde hace ya algún tiempo, cada vez que se monta una unidad externa, esta se monta dentro del directorio */media* y a su vez dentro de un directorio especifico dependiendo del usuario del sistema que monta el volumen (9).

De este modo, si en un sistema hay varios usuarios, ejemplo *User1* y *User2*, los puntos de montaje de los volúmenes se mostraran en directorios separados como tal:

/media/User1

/media/User2

/proc

Este directorio contiene información de los procesos y aplicaciones que se están ejecutando en un momento determinado en el sistema, pero realmente no guarda nada como tal, ya que lo que almacena son archivos virtuales, por lo que el contenido de este directorio es nulo. Básicamente son listas de eventos del sistema que se generan en el momento de acceder a ellos, y que no existen dentro del directorio como tales (9).

/root

Es como el directorio ***/****home* del usuario *root* o superusuario del sistema. A diferencia de los otros usuarios, que se encuentran todos dentro de /home en sus respectivas subcarpetas, el directorio del usuario *root* cuenta con su propia carpeta directamente en la raíz del sistema (9).

/sbin

Si ha dicho que en */bin* se almacenaban los binarios relativos a las funciones normales de usuario, ***/****sbin* hace lo mismo, pero para los binarios relativos a tareas propias del sistema operativo, y que solamente pueden ser gestionadas por el usuario *root*, tales como el arranque del sistema, tareas de restauración y reparación (9).

/srv

Sirve para almacenar archivos y directorios relativos a servidores que se puedan tener instalados dentro del sistema, ya sea un servidor web. Así, por ejemplo, en el caso de tener instalado un servidor de este tipo, es buena idea tener el directorio web público dentro de */srv*, tal como: (9):

/srv/www

/sys

Al igual que ***/****proc,* contiene archivos virtuales que proveen información del núcleo relativa a eventos del sistema. Es en cierto modo una evolución de ***/****proc*, y a diferencia de este último, los archivos se distribuyen de forma jerárquica (9).

/tmp

Como ya da a entender su nombre, sirve para almacenar archivos temporales de todo tipo, ya sean elementos del sistema, o de diferentes aplicaciones a nivel de usuario como puedan ser Firefox o Chrome.

Es un directorio destinado a almacenar contenido de corta duración de ejecución, de hecho, en la gran mayoría de los casos se suele vaciar de forma automática en cada reinicio del sistema. Aun así, se debe borrar su contenido de forma manual, puesto que puede contener archivos necesarios para ciertos programas o procesos que se estén ejecutando (9).

Las aplicaciones programadas para almacenar archivos en este directorio deben asumir que solo serán recuperables en la sesión actual. En ese sentido, hay otro subdirectorio, ***/var/tmp***, dispuesto igualmente para el almacenamiento de archivos temporales, pero cuyo contenido no se borra de forma automática tras el reinicio del sistema (9).

/usr

El directorio ***/usr*** proviene de "*User System Resources*" y sirve para almacenar todos los archivos de solo lectura y relativos a las utilidades de usuario, incluyendo todo el software instalado a través de los gestores de paquetes de cada distribución. Contiene los siguientes subdirectorios:
/usr/bin
/usr/include
/usr/lib
/usr/local
/usr/sbin
/usr/share
/usr/src

Antiguamente */usr* también contenía la carpeta particular de usuario, junto con todos sus documentos, vídeos, fotos, pero más adelante se creó el directorio */home* para este propósito, dejando */usr* reservado para los ficheros relativos a programas (9).

/var

Contiene varios archivos con información del sistema, como archivos de *logs*, emails de los usuarios del sistema, bases de datos, información almacenada en la caché, e información relativa a los paquetes de aplicaciones almacenados en */opt*. En cierto modo se podría decir que actúa a modo de registro del sistema (9).

Del sistema de directorios que presenta la Distribución Cubana de GNU/Linux Nova fueron seleccionados:

/etc, /usr, /home, /bin, /lib, /boot, /root /sbin ya que estos almacenan la información del usuario, programas y sus configuraciones. Los que contienen información temporal o virtual, no se seleccionan puesto que la información que contienen se crea de manera dinámica con el arranque del sistema y se pierde al reinicio de este.

1.3 Análisis de herramientas de copia de seguridad.

Gestión de copias de seguridad en las versiones de la Distribución Cubana de GNU/Linux Nova.

La gestión de copias de seguridad en las versiones de la Distribución Cubana de GNU/Linux Nova hasta el momento se realiza con la herramienta rsync. Esta permite almacenar la información de los usuarios, pero debido a que carece de una interfaz gráfica para su interacción se ejecuta mediante líneas de comandos por consola. Esta razón obliga a conocer dichos comandos y su respectiva sintaxis para realizar cualquier acción, lo que dificulta su uso en usuarios inexpertos y debido a que la distribución tiene como objetivo ser empleada por usuarios de todo el país esto podría constituir un problema para su correcto uso. Por otra parte, la herramienta no cuenta con la funcionalidad de restaurar una copia de seguridad, y la realización de esta de forma manual es un proceso aún más complejo ya que requiere la actualización del GRUB y la modificación del archivo *fstab*, operación que demanda conocimientos avanzados de GNU/Linux.

Para tener una visión sobre las herramientas de gestión de copias de seguridad se analizan las siguientes soluciones similares.

Areca Backup

Areca Backup es un software para la realización de copias de seguridad desarrollado en Java. Incluye un mecanismo de transacción que garantiza la integridad de una copia de seguridad. El software permite a los usuarios seleccionar archivos o directorios a respaldar. Permite filtrar, encriptar y comprimir su contenido, y almacenarlo en una ubicación de respaldo (10).

Esta aplicación está destinada a la copia de seguridad de documentos personales. Admite copias de seguridad incrementales, diferenciales y completas. Permite definir un conjunto de objetivos tales como:

- Qué archivo o directorio serán guardados.
- Establecer un conjunto de filtros de archivos para aplicar durante la copia de seguridad.
- Posibilidad de almacenar los archivos (con o sin compresión y encriptación).

BackupPC

BackupPC es un software de respaldo gratuito, escrito en Perl con una interfaz web. Es un sistema de alto rendimiento y de nivel empresarial para hacer una copia de seguridad de Linux, y computadoras portátiles en el disco de un servidor.

Las características incluyen la agrupación inteligente de archivos idénticos. Es un sistema de copias de seguridad y recuperación totalmente basado en discos de un servidor. BackupPC incorpora un cliente de bloque de mensajes de servidor (Samba) que se puede utilizar para hacer una copia de seguridad de los recursos compartidos de red, en las computadoras. También admite clientes *Dynamic Host Configuration Protocol* (DHCP)*,* protocolo de configuración dinámica de host, siempre que el cliente esté registrado con un servicio de nombres como *Domain Name System* (DNS), es el sistema de nombres de dominio. La herramienta es altamente configurable y fácil de instalar, el software se puede configurar para realizar una copia de seguridad completa a intervalos regulares.

Se puede configurar para mantener un cierto número de copias de seguridad completas. También admite el vencimiento exponencial, lo que permite realizar copias de seguridad completas con varias añadidas (por ejemplo, un número configurable de los totales semanales más recientes, más un número configurable de los anteriores completos que tienen 2, 4, 8 o 16 semanas de diferencia).

Sus parámetros flexibles de configuración permiten qué se realicen múltiples copias de seguridad en paralelo, especificación de qué recursos , que directorios respaldar, varias programaciones para respaldos completos e incrementales y programaciones por recordatorios de correo electrónico a usuarios (11).

Bacula

Bacula está compuesto por un conjunto de programas informáticos para la copia de seguridad, recuperación y verificación de datos informáticos en una red de computadoras de diferentes tipos. Es relativamente fácil de usar y eficiente, ofrece muchas funciones avanzadas de administración de almacenamiento que facilitan la búsqueda y recuperación de archivos perdidos o dañados. Está compuesta de los siguientes componentes o servicios principales: servicios de Director, Consola, Archivo, Almacenamiento y Monitor.

El servicio *Bacula Director* supervisa todas las operaciones de copia de seguridad, restauración, verificación y archivo. El servicio *Bacula Console* permite que el administrador o el usuario se comunique con Bacula Director. Actualmente, *Bacula Console* está disponible en tres versiones: interfaz de consola basada en

texto, interfaz basada en GNOME (GNU *Network Object Model Environment*, es el Entorno de Modelo de Objeto de Red GNU) y una interfaz gráfica *wxWidgets*. El servicio Bacula File (también conocido como el programa Cliente) está instalado en la máquina en que se realizará la copia de seguridad. Los servicios de *Bacula Storage* consisten en los programas de software que realizan el almacenamiento y la recuperación de los atributos y datos de los archivos en los medios o volúmenes físicos de respaldo.

Permite la restauración de uno o más archivos seleccionados interactivamente para la copia de seguridad actual o una copia de seguridad antes de una hora y fecha especificadas. Restauración de un sistema completo. Esto es principalmente automatizado para sistemas Linux y parcialmente automatizado para Solaris (12)

Las características incluyen:

- Copias de respaldo multi-volúmenes compatibles.
- Una completa base de datos estándar de SQL de todos los archivos respaldados. Esto permite la visualización en línea de los archivos guardados en cualquier volumen en particular.
- Poda automática de la base de datos (eliminación de registros antiguos) simplificando así la administración de la base de datos.
- Se puede usar cualquier motor de base de datos SQL, lo que hace que Bacula sea muy flexible.

Fwbackups

Es un programa de copias de seguridad con abundantes características que ofrece una interfaz simple pero potente que le permite realizar dicha tarea con facilidad (13).

Las características incluyen:

- Varios motores de copias de seguridad que incluyen compresión y copias de seguridad incrementales.
- Una configuración de copia flexible: que permite al usuario elegir entre una variedad de modos, incluido el formato de archivo y el modo de copia clonada para recuperar datos de discos dañados.
- Permite que cada usuario del sistema pueda configurar sus propias copias de seguridad.
- Copia de seguridad de archivos en una unidad remota, como un servidor de respaldo.
- Crea una copia de seguridad de toda la computadora.
- Copias de seguridad automatizadas para grupos de archivos y carpetas con un conjunto de respaldo.
- Soporte para realizar copias de seguridad en un host remoto a través de SSH / SFTP.

- Soporte para restaurar cualquiera de las copias de seguridad creadas con Fwbackups.
- Modos de copia de seguridad incremental o diferencial.
- Fwbackups se ocupa de la organización de las copias de seguridad, incluida la eliminación de las expiradas, por lo que no debe preocuparse por organizar las copias de seguridad.

Keep

Keep es un programa de copias de seguridad automático que permite a los usuarios establecer sus parámetros, incluida la frecuencia y la cantidad. Está diseñado para ser un sistema simple.

Hacer una copia de seguridad con esta aplicación es bastante sencillo: selecciona un directorio para hacer una copia de seguridad, selecciona el directorio en el que desea hacer una copia de seguridad, configura algunas opciones (por ejemplo, frecuencia) y listo. Posibilitando agregar más tarde algunos otros directorios (14).

Las características incluyen:

- Interfaz gráfica de usuario simple.
- Vista previa de tamaño de respaldo.
- Registro de respaldo.
- Sistema de carga para reenviar copias de seguridad a dispositivos externos y a través de ftp.

Simple Backup Solution

Simple Backup Solution es un conjunto de *daemons* de respaldo de back-end e interfaces gráficas de usuario (GUI) de GNOME que proporciona una solución de copia de seguridad simple pero poderosa para usuarios de escritorios comunes.

Las copias de seguridad se escriben en servidores remotos utilizando la tecnología GNOME. Un control fino es posible con respecto, a qué carpetas y archivos hacer una copia. Los archivos se pueden excluir incluso con un conjunto de expresiones regulares y se pueden programar copias de seguridad regulares (15).

Las características incluyen:

- Copia de seguridad de cualquier subconjunto de archivos y directorios.
- Excluir archivos y directorios por expresiones.
- Excluir archivos por tipo (extensión).

- Excluir archivos por tamaño máximo de archivo.
- Copia de seguridad en cualquier sistema de archivos remoto compatible con GNOME.
- Copias de seguridad completas e incrementales.
- Programación de copias de seguridad.
- Interfaz de usuario de GNOME para la configuración.
- Copia de seguridad de la lista de paquetes en distribuciones derivadas de Debian.

Afbackup

Es un sistema de copias de seguridad cliente-servidor basado en consola que permite que muchas estaciones de trabajo realicen copias de seguridad en un servidor central (de forma simultánea o en serie). Se utiliza para mantener los archivos comprimidos en un servidor de copia de seguridad o en un archivo.

Las copias de seguridad en los clientes se pueden iniciar automáticamente usando *cron-jobs* en los clientes, pero la solución más inteligente es iniciarlo de forma remota desde un host administrativo central.

La escritura de copias de seguridad normalmente se realiza de forma secuencial: la siguiente escritura en cinta va al final de la escritura anterior sin importar donde haya restaurado hasta el momento (16).

Las características incluyen:

- Sistema Cliente / Servidor.
- La autenticación del cliente se realiza antes de que pueda tomar el control de la aplicación.
- Se pueden configurar varios servidores para cada cliente: el servidor real se elige según la disponibilidad.
- Servidor de flujo múltiple, varios clientes pueden almacenar en un servidor al mismo tiempo.
- Opción de inicio remoto y administración centralizada.
- Restricción de acceso para dispositivos.
- Procesamiento por archivo del lado del cliente. Si el servidor empaquetó y luego procesó los archivos y directorios, un único bit defectuoso en la secuencia procesada haría que el resto de la copia de respaldo no fuera accesible para la restauración.
- Posee un registro de posición de cinta para cada archivo lo que facilita una búsqueda rápida de archivos durante la restauración.
- Manejo de cinta flexible y modo de adición configurable.

- Copias de seguridad completas e incrementales.
- Las particiones sin formato se pueden guardar.

Amanda

Amanda, es un software basado en consola como utilizado como archivador de discos de red automático avanzado, es un sistema de respaldo que permite al administrador configurar un solo servidor maestro de respaldo para hacer copias de seguridad de múltiples hosts a través de la red en unidades de cintas cambiadores, discos o medios ópticos.

Amanda usa volcado nativo e instalaciones GNU/Linux y puede realizar copias de seguridad de una gran cantidad de estaciones de trabajo que ejecutan múltiples versiones de Linux (17).

Las características incluyen:

- Tiene opciones de configuración para controlar casi todos los aspectos de la operación de copia de seguridad y proporciona varios métodos de programación.
- Diseñado para manejar una gran cantidad de clientes y datos, es razonablemente fácil de instalar y de operar.
- Realiza una copia de seguridad de varias máquinas en paralelo a un disco de retención.
- Admite una amplia gama de dispositivos de almacenamiento en cinta.
- Comunicación segura entre el servidor y el cliente.
- El cliente o el servidor de cintas pueden hacer compresión de software o se puede usar compresión de hardware. En el lado del cliente, la compresión del software reduce el tráfico de la red. En el lado del servidor, reduce la carga de la CPU del cliente.
- Soporte para múltiples escrituras simultáneas en dispositivos de almacenamiento.
- Incluye un programa de comprobador de ejecución previa, que realiza comprobaciones de cordura tanto en el host del servidor de cintas como en todos los hosts del cliente (en paralelo), y enviará un informe por correo electrónico de cualquier problema que pueda causar copias de seguridad.

Cedar Backup

Cedar Backup es un paquete de software basado en consola diseñado para administrar copias de seguridad del sistema para un conjunto de máquinas remotas. Entiende cómo realizar copias de seguridad de los datos del sistema de archivos, así como de las bases de datos MySQL y PostgreSQL. También se puede

ampliar fácilmente para admitir otros tipos de fuentes de datos.

Cedar Backup se centra en las copias de seguridad semanales en un único dispositivo externo, con la expectativa de que el disco se cambie o sobrescriba al comienzo de cada semana. Puede escribir discos multisesión, permitiendo a los usuarios agregar datos incrementales a un disco diariamente (18).

Las características incluyen:

- Comunicación segura entre el servidor y el cliente.
- Diseñado para ser flexible. Permite a los usuarios decidir qué pasos de copia de seguridad ejecutar (y cuándo ejecutarlos), en función de la situación y las prioridades del individuo.
- Variedad de fuentes de respaldo (sistema de archivos, repositorios de bases de datos, repositorios de control de revisiones).
- Admite el concepto de clientes administrados. Los clientes administrados tienen todo su proceso de respaldo administrado por el maestro a través de un *shell* remoto. Las mismas acciones se ejecutan como parte del proceso de copia de seguridad, pero el maestro controla cuando las acciones se ejecutan en los clientes en lugar de los clientes que lo controlan por sí mismos.
- División de datos entre múltiples discos.

Duplicity

Duplicity realiza copias de seguridad incrementales de archivos y directorios encriptando volúmenes de formato de tar con GNU y cargándolos a un servidor de archivos remoto. Para transmitir datos, puede usar el acceso a archivos locales, rsync y ftp.

Debido a que utiliza libsync, los archivos incrementales son eficientes desde el punto de vista del espacio y solo registran las partes de los archivos que han cambiado desde la última copia de seguridad. Como el software usa GnuPG para encriptar y / o firmar estos archivos, estarán a salvo de espionaje y / o modificación por parte del servidor.

El paquete también incluye la utilidad rdiffdir que es una extensión del rdiff de librsync a los directorios; se puede usar para producir firmas y deltas de directorios, así como archivos regulares (19).

Las características incluyen:

- Archivos encriptados y firmados.
- Ancho de banda y espacio eficiente, utilizando el algoritmo rsync.

- Formato de archivo estándar.
- Elección de protocolo remoto.

Tar

Es un software basado en consola para empaquetar un conjunto de archivos como un archivo único en formato tar. Se puede hacer un archivo tar en una unidad de cinta, sin embargo, también es común escribir un archivo tar en un archivo normal.

Es muy utilizado por el sistema de gestión de paquetes Debian, y es útil para realizar copias de seguridad del sistema e intercambiar conjuntos de archivos con otros.

Inicialmente, los archivos tar se usaban para almacenar archivos convenientemente en cinta magnética. El nombre "Tar" proviene de este uso; es sinónimo de archivador de cinta. A pesar del nombre de la utilidad, tar puede dirigir su salida a dispositivos, archivos u otros programas disponibles (utilizando tuberías), incluso puede acceder a dispositivos o archivos remotos (20).

Las características incluyen:

- Almacenamiento remoto de archivos.
- Extracción de archivos.
- Manipulación de archivos.

Rsync

Es una aplicación de software para sistemas de tipo Unix que ofrece transmisión eficiente de datos incrementales, que opera también con datos comprimidos y cifrados. Mediante una técnica de *delta encoding*, permite sincronizar archivos y directorios entre dos máquinas de una red o entre dos ubicaciones en una misma máquina, minimizando el volumen de datos transferidos. Una característica importante de rsync no encontrada en la mayoría de programas o protocolos es que la copia toma lugar con sólo una transmisión en cada dirección. rsync puede copiar o mostrar directorios contenidos y copia de archivos, opcionalmente usando compresión y recursión (21).

En la siguiente tabla se muestra la comparación de las herramientas estudiadas donde los parámetros a cumplir responden a las características fundamentales que debe tener una herramienta que gestione copias de seguridad para la Distribución Cubana de GNU/Linux Nova.

Tabla1: Comparación de las herramientas

Herramientas	***Copias de seguridad locales***	***Almacenamiento de la configuración de usuario***	***Interfaz gráfica***
Areca Backup		X	X
Backup PC		X	X
Bacula		X	X
Fwbackups		X	X
Keep		X	X
Simple Backup Solution		X	X
Afbackup		X	
Amanda		X	
Cedar Backup		X	
Duplicity		X	
Tar		X	
Rsync	X	X	

Las aplicaciones analizadas demuestran que “Crear” y “Restaurar” son las funcionalidades principales con las que debe contar una herramienta de este tipo. La comparación anterior arroja que ninguna de estas herramientas cumple con los parámetros para la ejecución de los usuarios de la distribución, puesto que la funcionalidad de restauración que implementan consiste en comparar los archivos que se encuentran en la copia del servidor con los que se tienen almacenados en disco y en caso que falte alguno, reponerlo. Dicha restauración puede copiar los directorios necesarios para el funcionamiento de la distribución, pero en caso de querer utilizar una copia de seguridad en otro ordenador o que se pierda información debido a la ruptura del disco duro estas aplicaciones no solapan la actualización de GRUB ni la modificación del fichero *fstab* ya que este almacena el identificador de cada disco y al momento de restaurar una copia en un nuevo hardware es necesaria la modificación de este. También carecen de interfaz gráfica que permita al usuario de manera sencilla la gestión de copias de seguridad o solicitan almacenar la copia en un servidor remoto, tecnología con la que no se cuenta en el mayor de los casos. Debido al análisis realizado se evidencia la necesidad de desarrollar una herramienta que cumpla con todos los parámetros establecidos.

1.4 Tecnologías

Para el desarrollo de la herramienta informática de gestión de copias de seguridad, se hace imprescindible contar con un conjunto de tecnologías que permitan su desarrollo, las seleccionadas fueron:

Lenguaje para la creación de interfaces

Se utilizó en la presente investigación GIMP Toolkit (GTK) y sus librerías para la creación de las interfaces de usuario, debido a que el entorno de escritorio de la Distribución Cubana de GNU/Linux Nova se integra correctamente con el código GTK.

GTK, es un conjunto de herramientas multiplataforma para crear interfaces gráficas de usuario. Al ofrecer un conjunto completo de *widgets*, GTK es adecuado para proyectos que van desde pequeñas herramientas únicas hasta completas suites de aplicaciones.

GTK es multiplataforma y fácil de usar, lo que acelera el tiempo de desarrollo, está escrito en C, pero se ha diseñado desde cero para admitir una amplia gama de idiomas, no solo C / C ++. El uso de GTK va desde lenguajes como Perl y Python (especialmente en combinación con el constructor Glade) proporcionando un método efectivo de desarrollo de interfaces de usuario (22).

Herramienta para la creación de interfaces

En la investigación para la creación de interfaces gráficas de usuario se utilizó Glade por su integración con código GTK. y por las facilidades que brinda para un sencillo diseño y creación de interfaces de usuario. Las interfaces de usuario se guardan como un archivo que describe la estructura del *widget* (artefacto) y sus propiedades, estos son asociados con cada uno de los manejadores de señales. El paquete LibGlade puede cargar el archivo de interfaz de usuario con el fin de construir dinámicamente las aplicaciones. Esto permite modificar la interfaz de usuario estéticamente sin necesidad de recompilar la aplicación. Glade se utiliza para diseñar la interfaz de usuario de una aplicación, configurar las señales que serán asociados con funciones de retro llamada implementadas en el código y cuidar las propiedades comunes del widget. Sin embargo, Glade no es un editor de código o un entorno de desarrollo integrado, imprime los archivos que deben ser cargados por su aplicación, y debe implementar todas las funciones de devolución de llamada en el código. Glade sólo está destinado a simplificar el proceso de inicialización de interfaz gráfica de usuario de la aplicación y la conexión de las señales (23).

Lenguaje de programación

Para el desarrollo de la solución el lenguaje de programación seleccionado es Python en su versión 2.7.12, ya que permite un manejo fácil de procesos y subprocesos dentro de una misma aplicación. Python tiene una correcta integración con Glade que es la herramienta seleccionada para la creación de las interfaces de usuario.

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones (24).

Python usa tipado dinámico y conteo de referencias para la administración de memoria. Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos) (25).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable. Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación

funcional tradicional ya que existen bastantes analogías entre Python y otros lenguajes (25).

Entorno de desarrollo integrado

Se selecciona PyCharm como entorno de desarrollo integrado ya que la herramienta está diseñada para el desarrollo de aplicaciones en lenguaje Python. Cuenta con una versión gratuita llamada "*Community Edition*" liberada bajo la licencia Apache que presenta entre sus características: autocompletado, depuración de código gráfico, refactorización, inspección de código, integración de control de versiones, ejecución de *tests*, soporte para el desarrollo web con Django además de contar con un editor inteligente y una consola Python integrada. La versión de PyCharm seleccionada es la v2018.1 pues incluye mejoras para reducir el tiempo de espera entre procesos, mayor velocidad y facilidad de configuración del intérprete (26).

Lenguaje de modelado UML 2.0

UML (Lenguaje de Modelado Unificado) es un lenguaje de modelado estandarizado que consiste en un conjunto de diagramas desarrollados para ayudar a los desarrolladores de software y de sistema en la especificación, visualización, construcción y documentación de los artefactos de los sistemas de software, así como también para el modelado de negocio y otros sistemas (27). Para modelar los artefactos generados se utilizará UML en su versión 2.0 pues el desarrollador posee experiencia en el uso de este lenguaje.

Herramienta de modelado

En la presente investigación se utiliza como herramienta para el modelado Visual Paradigm ya que es la herramienta orientada por la universidad. Es una herramienta de Ingeniería de Software asistida por computación, la misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm está concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Fue diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (28).

Visual Paradigm ofrece, además ingeniería inversa código a modelo, código a diagrama. Generación de código, modelo a código, diagrama a código, incluyendo la especificación del modelo general y de las descripciones de los casos de uso, diagramas de flujo de datos (29).

Sistema de control de versiones

Se selecciona Git como herramienta para el control de versiones ya que es la utilizada en centro CESOL y se opta por la utilización de una herramienta de este tipo debido a que se hace necesario mantener un registro sobre los cambios realizados al proyecto y por su integración con Pycharm.

Git es un sistema de control de versiones diseñado para detectar cambios en un conjunto de ficheros y para realizar trabajo en paralelo sobre estos entre varias personas. Es usado principalmente para la gestión de código fuente en el desarrollo de software, pero puede ser utilizado para mantener un registro de cambios sobre cualquier conjunto de ficheros (30).

1.5 Metodología de desarrollo de software

Para el desarrollo del presente trabajo se hace uso de la metodología de desarrollo AUP variación para la UCI, puesto que es la seleccionada en la universidad para emplearse en los proyectos productivos.

AUP variación para la UCI define cuatro escenarios para modelar el negocio. En la presente investigación se utiliza el escenario cuatro, ya que este aplica a los proyectos que evalúan el negocio a informatizar y como resultado se obtiene un negocio muy bien definido (31).

La metodología cuenta con 3 fases, las cuales se explican de forma breve a continuación:

1. **Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En dicha 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.

2. **Ejecución:** es la fase, en que 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.

3. **Cierre:** se analizan tanto los resultados del proyecto como su ejecución.

Conclusiones del capítulo

El estudio sobre de las herramientas de gestión de copias de seguridad arrojó que las existentes no responden a las necesidades de los usuarios de la Distribución Cubana de GNU/Linux Nova, por lo que se

hace necesario el desarrollo de una nueva solución, permitiendo identificar las funcionalidades básicas de las herramientas de este tipo. El análisis de los directorios de la distribución determinó que /etc, /usr, /home, /bin, /lib, /root, /boot y /sbin son los directorios principales para la realización de la copia de seguridad. El estudio de las tecnologías, así como del entorno en el que desarrollar la solución determinó la utilización de: Visual Paradigm en su versión 8.0 para el modelado, Python 2.7.12 como lenguaje de programación, PyCharm 2018.1 y como metodología de desarrollo de software AUP variación para la UCI.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA HERRAMIENTA DE GESTIÓN DE COPIAS DE SEGURIDAD PARA LA DISTRIBUCIÓN CUBANA DE GNU/LINUX NOVA.

Una vez identificadas las herramientas a utilizar para el desarrollo de la aplicación en el presente capítulo se describe la propuesta de solución. Además, se identifican los requisitos funcionales y los requisitos no funcionales de la solución, especificados en historias de usuario. Se determinan la arquitectura y los patrones de diseño para implementación.

2.1 Descripción de la propuesta de solución

La presente investigación tiene como propuesta de solución una aplicación de escritorio, que se encuentre disponible en el servidor de aplicaciones de la Distribución Cubana de GNU/Linux Nova que le permita al usuario preservar los programas instalados y sus respectivas configuraciones, como también los archivos contenidos en la partición donde se ubica el sistema realizando una copia de seguridad completa a la distribución. Para ello el usuario tendrá la opción de seleccionar previamente los directorios que desee salvar. Dicha copia puede ser creada en cualquier directorio del ordenador incluyendo dispositivos extraíbles. La herramienta permitirá mostrar las copias creadas y de igual forma ofrece la posibilidad de eliminarlas. La aplicación permitirá la restauración de las copias de seguridad creadas por el usuario y se encargará de manera automática de la actualización del GRUB para que una vez culminado el proceso dicha copia se encuentre en el menú de selección.

Para una mejor comprensión de la propuesta de solución se muestra el siguiente mapa conceptual.

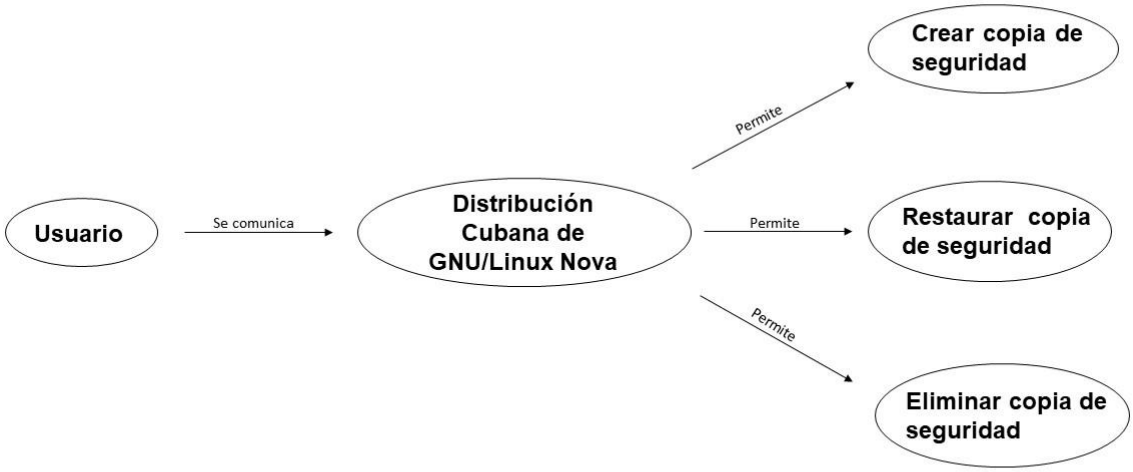


Figura 1. Mapa conceptual

(Fuente: Elaboración propia)

El usuario es quien utiliza la Distribución Cubana de GNU/Linux Nova la cual permite Crear, Restaurar y Eliminar copias de seguridad

2.2 Características y cualidades de la propuesta de solución

Requisitos

Los requisitos de software son las cualidades del sistema, se definen y describen de forma clara, consistente, compacta, y sin ambigüedades. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir; los no funcionales son propiedades o cualidades del producto. Los requisitos forman parte del modelo del sistema que se desarrolla (32).

Técnica de identificación de requisitos

Existen varias técnicas que permiten establecer comunicación clara entre los interesados en el producto y el equipo de trabajo, pues facilitan indagar en lo que los usuarios realmente necesitan. Dichas técnicas se centran principalmente en el descubrimiento de los requisitos del sistema. Por lo que es necesario trabajar

con el cliente debido a que sus requisitos abarcan las tareas que se necesitan para llevar a cabo el desarrollo (33). La identificación de requisitos fue realizada mediante la técnica:

Entrevista: se utilizó con el fin de obtener diversas opiniones y recomendaciones en un intercambio mediante preguntas con los especialistas de desarrollo de la Distribución Cubana de GNU/Linux Nova, para esclarecer con precisión las funcionalidades principales del sistema, ver **Anexo 1**.

Especificación de requisitos de software

La especificación de requisitos del software es un proceso de descubrimiento, refinamiento de requisitos. Se refinan en detalle los requisitos del sistema y el papel asignado al software. Tanto el desarrollador como el cliente tienen un papel activo en la ingeniería de requisitos siendo un conjunto de actividades que son denominadas análisis (32). El cliente intenta replantear un sistema confuso, a nivel de descripción de datos, funciones y comportamiento, en detalles concretos. El desarrollador actúa como interrogador, como consultor, como persona que resuelve problemas y como negociador.

El análisis de requisitos es una tarea de ingeniería del software que cubre el hueco entre la definición del software a nivel sistema y el diseño de software. Este análisis permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software (32).

La propuesta de solución cuenta con cuatro requisitos funcionales presentados en la siguiente tabla:

Tabla 2: Requisitos funcionales

No.	***Nombre***	***Descripción***	***Prioridad***	***Complejidad***
RF_1	Crear copia de seguridad	Se crea una copia de seguridad de la distribución donde se contienen los directorios seleccionados por el usuario en la ruta especificada por él.	Alta	Alta
RF_2	Restaurar copia de seguridad	El sistema se restaura a partir de la selección previa de una copia de seguridad creada por el usuario descomprimiéndola en una ruta especificada por él.	Alta	Media

RF_3	Mostrar copia de seguridad	Se muestran las copias de seguridad creadas.	Media	Media
RF_4	Eliminar copia de seguridad	Se elimina la copia de seguridad que el usuario desee.	Media	Media

Requisitos no funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema informático (34).

A continuación, se listan los requisitos no funcionales de la propuesta de solución:

Requisitos no funcionales de usabilidad:

RNF _1: La interfaz visual en sus botones y vistas debe propiciar que el usuario intuya cómo operar cada funcionalidad.

RNF_2: Las vistas de la herramienta deben mostrar en cada momento la acción que se está realizando.

RNF_3: Los íconos deben estar representados por una imagen acorde a la acción que se realiza.

Requisitos no funcionales de hardware:

RNF_4: PC de 512 Mb de memoria RAM o superior.

RNF_5: El espacio en disco deberá ser superior al tamaño del sistema.

Requisito no funcional de software:

RNF_6: La herramienta está diseñada para operar en la Distribución Cubana de GNU/Linux Nova.

Requisito no funcional de disponibilidad:

RNF_7: La herramienta debe estar disponible en el repositorio de aplicaciones.

Historias de usuario

Las historias de usuario guían la construcción de las pruebas de aceptación (deben generarse una o más pruebas para verificar que la historia ha sido correctamente implementada) y son utilizadas para estimar tiempos de desarrollo. En este sentido, solo proveen detalles suficientes para hacer una estimación razonable del tiempo que llevará implementarlas. En el momento de implementar una historia de usuario, se debe detallar a través de la comunicación con el cliente. Son las bases para las pruebas funcionales (35). La presente investigación cuenta con cuatro historias de usuario. A continuación, en las siguientes tablas se muestran las historias de usuario asociadas a los requisitos funcionales.

Tabla 3: Historia de usuario 1

Número: HU_1	**Nombre del requisito:** Crear copia de seguridad.
Programador: Mauricio Leal Aguilar	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 20 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 20 horas
Descripción: Se listan a continuación el proceso de creación de una copia de seguridad con la herramienta: Permitir que el usuario seleccione a qué directorios desea realizar la copia, dicha selección se realiza mediante *checkbox* los cuales indican los directorios. Permitir que el usuario acepte o cancele la operación mediante botones. Una vez aceptada la operación permitir que el usuario seleccione dónde desea almacenar la copia de seguridad, esta selección se realiza mediante una estructura de *filechooser*. En el proceso de creación de la copia de seguridad para ahorrar espacio en disco se comprimen los directorios seleccionados en formato *squashfs.*	
Observaciones: La copia de seguridad se almacena automáticamente después de creada.	

Tabla 4: Historia de usuario 2

Número: HU_2	**Nombre del requisito:** Restaurar copia de seguridad.
Programador: Mauricio Leal Aguilar	**Iteración Asignada:** 1

Prioridad: Alta	**Tiempo Estimado:** 20 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 20 horas
Descripción: Permitir al usuario seleccionar la copia de seguridad que desee restaurar y dicha selección se realizada mediante la estructura *filechooser*. Se selecciona la ruta donde será restaurada la información y dicha selección es realizada mediante la estructura *filechooser*. Una vez seleccionada la ruta donde será restaurada la copia, se descomprimen sus datos con el comando *unsquashfs*. Actualizar el GRUB	
Observaciones: La copia de seguridad debe de estar creada para ser restaurada.	

Tabla 5: Historia de usuario 3

Número: HU_3	**Nombre del requisito:** Eliminar copia de seguridad.
Programador: Mauricio Leal Aguilar	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 10 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 10 horas
Descripción: Permitir al usuario la eliminación de una copia de seguridad, la copia que se desea eliminar se selecciona mediante una estructura *filechooser*. Una vez seleccionada la copia de seguridad se elimina mediante la función *rmthree* la cual recibe por parámetro la dirección del archivo seleccionado en el *filechooser*.	
Observaciones: Las copias de seguridad deben estar creadas para poder ser eliminadas.	

Tabla 6: Historia de usuario 4

Número: HU_4	**Nombre del requisito:** Mostrar copia de seguridad.

Programador: Mauricio Leal Aguilar	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:** 20 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 20 horas
Descripción: Permitir mostrar al usuario las copas de seguridad creadas dicha operación se realiza mediante la estructura *filechooser*.	
Observaciones: La copia de seguridad debe de estar creada para ser mostrada.	

2.3 Análisis y diseño de la propuesta de solución.

Definición de la arquitectura

La arquitectura del software de un programa es la estructura del sistema, que incluye los componentes del software, sus propiedades visibles externamente y las relaciones entre ellos (34). En la presente investigación para la propuesta de solución se selecciona la arquitectura N-capas, con un estilo arquitectónico dos capas ya que su estructura permite separar de forma clara los componentes de cada capa y a su vez permite integrarlos entre sí para la ejecución de las tareas.

Para lograr una mayor comprensión de la estructura de la herramienta, se muestra a continuación un diagrama de paquetes, donde se evidencia la arquitectura seleccionada. En el diagrama se muestra el paquete Vista que encierra todos los componentes visuales de la aplicación y permite al usuario la selección de operaciones. En el paquete Funcionalidad se encuentra la implementación de todas las operaciones que puede realizar el usuario donde las peticiones realizadas en la Vista son resueltas por sus respectivas instrucciones del paquete Funcionalidad.

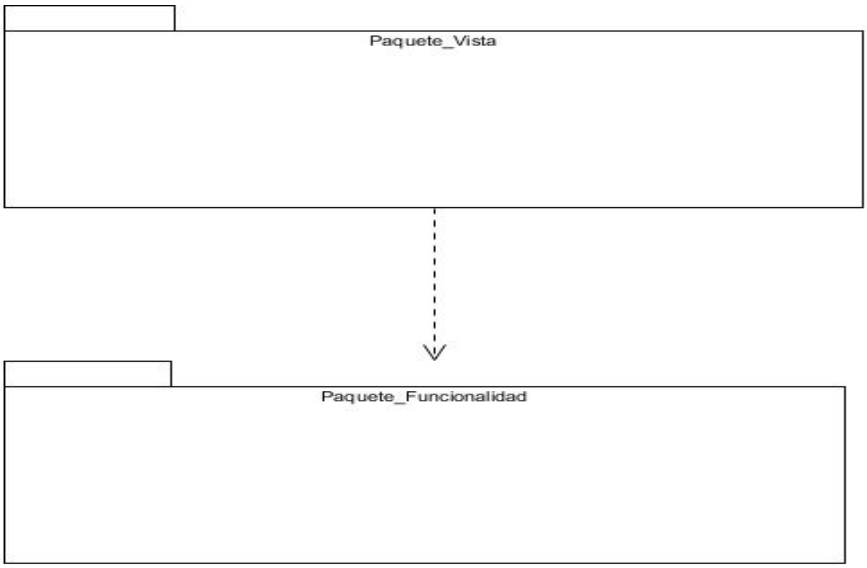


Figura 2: Diagrama de paquetes

(Fuente de elaboración propia)

Diagrama de clases de diseño

Los diagramas de clase permiten modelar y describir los tipos de clases de un sistema, así como los distintos tipos de relaciones que pueden existir entre ellas; se considera la técnica más potente para el modelado de un software, la cual suele recoger los conceptos claves del modelo de objetos subyacente al método orientado a objetos (36). A continuación, se muestra el diagrama de clases para la propuesta de solución, ver figura 2.

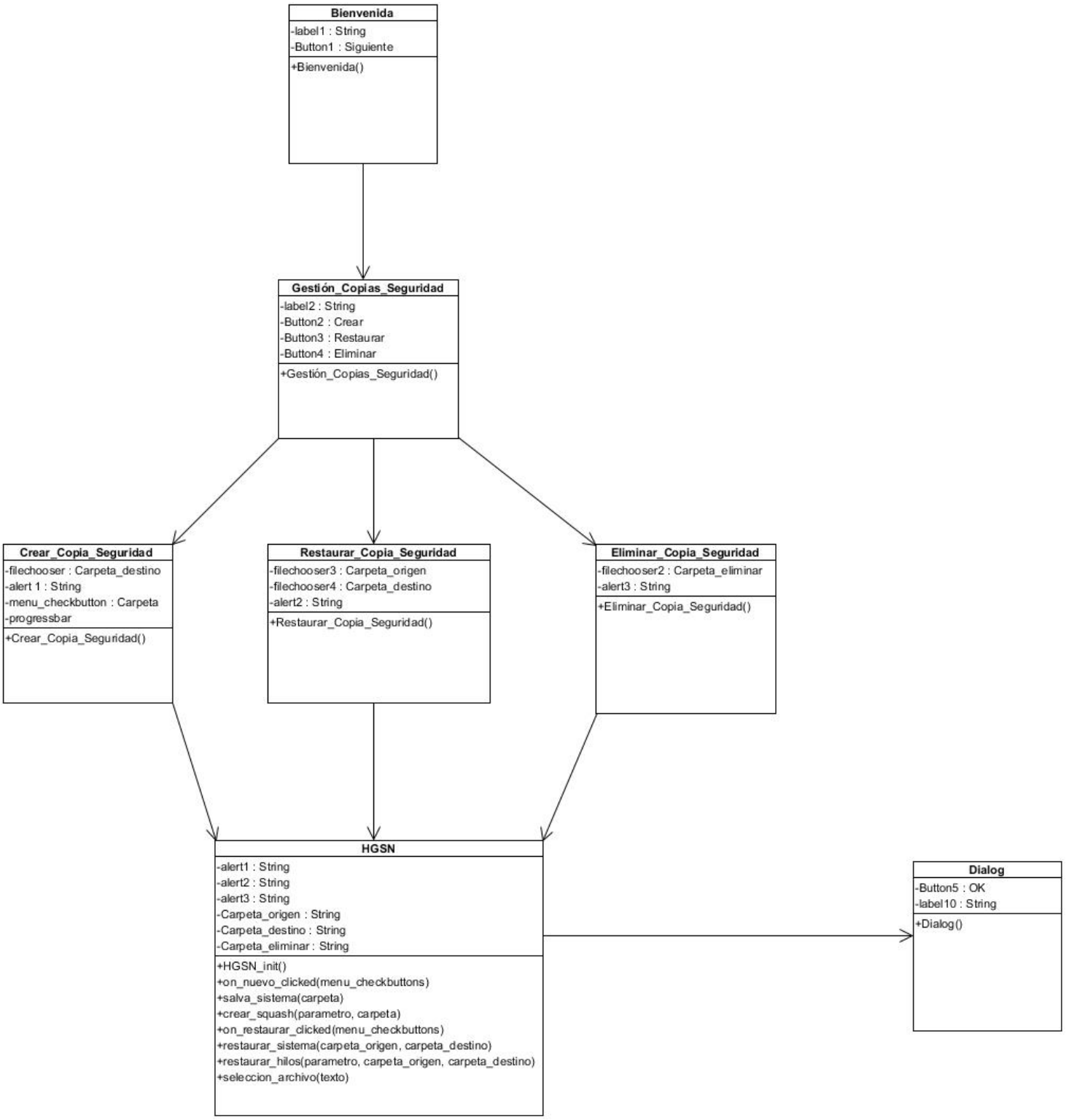


Figura 3: Diagrama de clases

En el diagrama de la figura 2 se muestran las clases utilizadas en la implementación de la solución propuesta siendo la clase HGSN la clase principal, donde se encuentran implementadas todas las funcionalidades de la herramienta. Las clases Restaurar_Copia_de_Seguridad, Crear_Copia_de_Seguridad y Eliminar_Copia_de_Seguridad muestran las respectivas interfaces para la ejecución de sus funcionalidades en la clase HGSN. En la clase Dialog se encuentra una vista intermedia tipo diálogo, para comprobar que el usuario está seguro de la operación a realizar, después de aceptar comienza la actividad seleccionada.

Patrones de diseño

Los diseñadores expertos en orientación a objetos han formado un amplio repertorio de principios generales y expresiones que los guía a crear el software. A estos se le asigna el nombre de patrones, si se codifican en un formato estructurado que describe el problema y su solución, y si se le asigna un nombre (37).

Se ha hecho uso de patrones de los principios generales para asignar responsabilidades GRASP[2], entre los que se encuentran:

Experto: es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos realizan cosas relacionadas con la información que poseen. Ofrece una analogía con el mundo real. El método que se muestra a continuación pertenece a la clase HGSN, dicha clase tiene asignada y bien definida sus responsabilidades dentro de la aplicación.

[2] *General Responsibility Assignment Software Patterns*. Traducido al español Patrones Generales de Software para Asignar Responsabilidades.

```
def restaurar_hilos(self, carpeta_origen, carpeta_destino):
  for i in range(len(self.arreglo)):
    existe = os.path.exists(carpeta_destino+"/"+self.arreglo[i])
    if existe == True:
      shutil.rmtree(carpeta_destino+"/"+self.arreglo[i])
    parametro = self.arreglo[i]
    existe = os.path.exists(carpeta_origen+"/"+parametro+"/"+parametro+".squashfs")
    if existe == True:
      os.system("unsquashfs -d "+carpeta_destino+"/"+parametro+" "+carpeta_origen+"/"+parametro+".squashfs")
      if parametro == 'etc':
        comando =  "blkid | grep `df -h "+carpeta_destino+" | cut -d ' ' -f1 | grep /`"
        tuberia = os.popen(comando)
        salida_estandar = tuberia.readlines()
        tuberia.close()
```

Figura 4: Método de la clase HGSN

Creador: el patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento, se debe buscar una clase de objeto que agregue, contenga y realice otras operaciones sobre este tipo de instancias (27). Se evidencia en la clase HGSN en la creación de instancias de la clase Dialog, cuando se lanzan las alertas para la confirmación de las operaciones.

```
def Alerta(self, texto):
  dialog = UnSoloBoton(self.window, texto)
  response = dialog.run()
  dialog.destroy()
  return response
```

Figura 5: Método de la clase Dialog

Bajo acoplamiento: el bajo acoplamiento es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento (27). Se evidencia en la reutilización de componentes entre la clase HGSN y la clase Dialog.

Controlador: es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos (27).Se utiliza en la herramienta para garantizar que los procesos funcionales sean manejados por la capa Funcionalidad y no por ninguna de las vistas.

Conclusiones del capítulo

La captura de los requisitos y la elaboración de las historias de usuario correspondientes permitió comprender e identificar las funcionalidades de la aplicación que se desea desarrollar, el comportamiento de la misma y la secuencia de actividades que debe de realizar el usuario para lograr su objetivo. La utilización de la arquitectura N capas y los patrones de diseño GRASP contribuyeron 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 Se identificaron 4 requisitos funcionales y 7 requisitos no funcionales.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DE LA HERRAMIENTA DE GESTIÓN DE COPIAS DE SEGURIDAD PARA LA DISTRIBUCIÓN CUBANA DE GNU/LINUX NOVA.

En el presente capítulo se describe la implementación del software, fase donde se materializa el producto y cumple con los requisitos obtenidos al inicio de la investigación. Se definen los estándares de codificación que debe cumplir, así como los métodos y técnicas para la realización de las pruebas.

3.1 Estándar de codificación

Se establecen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea legible (38). La codificación de las aplicaciones fue realizada siguiendo los estándares que se explican a continuación.

- **Código identado:** Emplear 4 espacios por cada nivel de identación.
- **Tabuladores y espacios:** No mezclar las tabulaciones con los espacios.
- **Tamaño máximo de líneas:** Se limitan todas las líneas a un máximo de 50 caracteres. Esto puede ser realizado mediante el uso de paréntesis de forma implícita o mediante el uso de la barra invertida.
- **Nomenclatura de las variables:** en el caso de que sea una palabra compuesta será separada mediante guiones bajos según sea necesario para mejorar la legibilidad (39)

3.2 Pruebas de software

Las pruebas de software comprenden un conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de una aplicación, por medio de pruebas sobre al comportamiento de la misma ya que los sistemas informáticos, programas y aplicaciones han crecido a niveles inimaginables en complejidad e interoperabilidad, con lo cual también se han incrementado las posibilidades de defectos (40).

En principio a la herramienta le fueron aplicadas pruebas de unidad, con el método de caja blanca y la técnica de camino básico para comprobar la correcta ejecución de sus principales funcionalidades. Posteriormente se aplicaron pruebas funcionales con el método de caja negra utilizando la técnica partición de equivalencia donde los requisitos descritos en las historias de usuario fueron evaluados.

Pruebas unitarias

Las pruebas unitarias le fueron aplicadas a la herramienta ya que constituyen el proceso de probar componentes del programa tales como métodos o clases de objetos. Las funciones o los métodos individuales son el tipo más simple de componente. Las pruebas deben llamarse para dichas rutinas con diferentes parámetros de entrada (34).

Para el diseño de los casos de prueba fue utilizado el método de caja blanca, donde las pruebas se enfocan en la estructura de control del programa. Los casos de prueba se derivan para asegurar que todos los enunciados en el programa se ejecutaron al menos una vez durante las pruebas y que todas las condiciones lógicas se revisaron (32).

La técnica de prueba de caja blanca utilizada en la investigación es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (32).

En la siguiente figura se muestra el procedimiento de selección de archivos de una copia de seguridad para durante el proceso de restauración.

Figura 6: Proceso de selección de copia de seguridad.

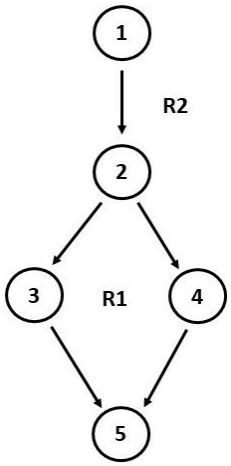


Figura 7: Grafo de flujo y cálculo de caminos linealmente independientes

Grafo de flujo y cálculo de caminos linealmente independientes:

Camino 1: 1,2,3,5

Camino 2: 1,2,4,5

Fórmula para calcular la cantidad de caminos:

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

P: es el número de nodos predicados (nodos con más de una arista de salida) contenidos en el grafo.

N: son las regiones, áreas delimitadas por los nodos y aristas del grafo.

V(G) = A-N+2 **V(G) = P+1** **V(G) = R**

V(G) = 5-5+2 **V(G)** = 1+1 **V(G)** = 2

V(G) = 2 **V(G)** = 2

Tabla 7: Caso de prueba para el camino 1

Diseño de caso de prueba para el camino 1	
Descripción	Selección del directorio donde se encuentra almacenada la copia de seguridad.
Condición de ejecución	Parte del proceso de restauración de la copia de seguridad.
Entrada	Gtk.ResponseType.OK: confirma que el usuario ha seleccionado el directorio.

Resultado esperado	Se almacena en la variable aux la dirección del directorio seleccionado y se cierra el cuadro de diálogo.

Tabla 8: Caso de prueba para el camino 2

Diseño de caso de prueba para el camino 2	
Descripción	No se selecciona el archivo de copia de seguridad.
Condición de ejecución	Parte del proceso de restauración de la copia de seguridad
Entrada	Gtk.ResponseType.CANCEL confirma que el usuario a cancelado la operación
Resultado esperado	Se cierra el diálogo si la selección del archivo.

Para una primera iteración de la prueba fueron detectadas dos no conformidades explicadas a continuación:

- Una vez confirmado el directorio de la copia de seguridad la dirección de este no se guarda en la variable aux.
- El cuadro de diálogo no se cierra después de la ejecución de la aceptación o cancelación de la operación.

Durante una segunda iteración de la prueba no fueron encontradas no conformidades, por lo cual no fue necesaria la ejecución de una tercera iteración.

Pruebas funcionales

Fueron realizadas pruebas funcionales para verificar la apropiada aceptación de datos y el procesamiento, recuperación e implementación adecuada de las reglas del negocio (32).

Como método de prueba se utilizó caja negra que se enfoca en probar el sistema sin tomar en cuenta la estructura interna del mismo, su objetivo es validar que las salidas sean las esperadas. Se centra en encontrar las circunstancias en las que el sistema no se comporta conforme a las especificaciones establecidas (32).

La técnica de prueba utilizada fue partición de equivalencia que divide el dominio de entrada de un programa en clases de equivalencia, a partir de las cuales pueden derivarse casos de prueba. Además, descubre clases de errores, que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general. Mediante su empleo se puede reducir al máximo el total de casos de prueba que deben desarrollarse (34).

Se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. Su objetivo

es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos. Para su diseño, se selecciona una característica del sistema o componente que se está probando, un conjunto de entradas que ejecutan dicha característica, se documentan las salidas esperadas o rasgos de salida y donde sea posible se diseña una prueba automatizada que demuestre que las salidas reales y las esperadas son las mismas (41).

Los casos de prueba fueron diseñados teniendo en cuenta las funcionalidades descritas en las historias de usuario permitiendo una mejor compresión de las especificaciones con las que la solución debe cumplir. A continuación, se muestran los casos de prueba correspondientes a las historias de usuario “Crear copia de seguridad”, “Restaurar copia de seguridad” y “Eliminar copia de seguridad” ya que representan las historias de usuario principales en la solución.

Tabla 9: Caso de Prueba "Restaurar copia de seguridad"

Nombre del requisito	Carpeta Origen	Carpeta destino	Descripción general	Escenarios de prueba	Flujo del escenario
Restaurar copia de seguridad	V	V	La herramienta debe permitir la restauración de las copias de seguridad.	EP 1.1: Restaurar copia de seguridad correctamente.	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación restaurar copia de seguridad. 3. Se selecciona el directorio donde se encuentra la copia de seguridad que se desea restaurar. 4. Se selecciona el directorio donde será restaurada la copia de seguridad. 5.Se realiza la restauración de la copia de seguridad. 6. La herramienta muestra un mensaje para informar que la operación se ha realizado correctamente.
	I	**V**	La herramienta debe emitir un mensaje de error cuando no es introducido	EP 1.2: Restaurar la copia de seguridad sin especificar el directorio donde se	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación restaurar copia de seguridad.

			el directorio que almacena la copia de seguridad	encuentra la copia de seguridad.	3. No se selecciona el directorio donde la copia de seguridad está almacenada. 4. Se selecciona el directorio donde se realizará la restauración. 5. La herramienta muestra un mensaje para informar que la operación no se ha podido realizar.
	V	**I**	La herramienta debe emitir un mensaje de error cuando no es introducido el directorio en que será realizada la restauración.	EP 1.3 Restaurar copia de seguridad sin especificar en qué directorio se va a restaurar.	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación restaurar copia de seguridad. 3. Se selecciona el directorio donde se encuentra la copia de seguridad. 4. No se selecciona el directorio donde se va a realizar la restauración. 5. Se realiza la restauración en la carpeta donde se encuentra la aplicación. 6. La herramienta muestra un mensaje para informar que la operación se ha realizado correctamente.

Tabla 10: Caso de Prueba “Crear copia de seguridad”

Nombre del requisito	Seleccionar directorios	Selección del directorio donde almacenar la copia	Descripción general	Escenarios de prueba	Flujo del escenario
Crear copia de seguridad	**V**	**V**	La herramienta debe permitir crear las copia de seguridad.	EP 1.1: Crear copia de seguridad correctamente.	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación crear copia de seguridad. 3. Se seleccionan los directorios a copia de seguridad. 4. Se selecciona el directorio donde almacenar la copia de seguridad. 5.Se realiza la copia de seguridad. 6. La herramienta muestra un mensaje para informar que la operación se ha realizado correctamente.
	V	**I**	La herramienta debe permitir crear la copia de seguridad en el directorio donde se encuentra la aplicación	EP 1.2: Crear copia de seguridad en un directorio no especificado	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación crear copia de seguridad. 3. Se seleccionan los directorios a copiar. 4. No se selecciona el directorio donde almacenar la copia de seguridad.

					5. Se realiza la copia de seguridad en la carpeta donde se encuentra la aplicación. 6. La herramienta muestra un mensaje para informar que la operación se ha realizado correctamente.
	I	**V**	La herramienta debe emitir un mensaje de error notificando que la operación no puede ser realizada.	EP 1.3 Crear una copia de seguridad sin especificar los directorios a realizar la copia de seguridad.	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación crear copia de seguridad. 3. No se seleccionan los directorios a copiar. 4. Se muestra un mensaje de error informando que no se seleccionó ningún directorio.

Tabla 11: Caso de Prueba "Eliminar copia de seguridad"

Nombre del requisito	Directorio a eliminar	Descripción general	Escenarios de prueba	Flujo del escenario
Eliminar copia de seguridad	**V**	La herramienta debe permitir la eliminación de copias de seguridad.	EP 1.1: Eliminar copia de seguridad correctamente.	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación eliminar copia de seguridad. 3. Se selecciona el directorio donde se encuentra la copia de seguridad que se desea eliminar. 4. Se realiza la eliminación de la copia de seguridad. 5. La herramienta muestra un mensaje para informar que la operación se ha realizado correctamente.
	I		EP 1.2: Eliminar copia de seguridad, sin especificar el directorio donde se encuentra.	1. Se muestra la interfaz para la selección de operación. 2. Se selecciona la operación eliminar copia de seguridad. 3. No se selecciona el directorio se encuentra la copia de seguridad. 4. La herramienta muestra un mensaje para informar que la operación no se ha podido realizar.

Según muestra la siguiente gráfica fueron realizadas tres iteraciones de pruebas funcionales, en la primera se encontraron 9 no conformidades, de ellas 2 de ortografía y 7 de funcionalidad. Para la segunda iteración fueron corregidas las no conformidades de ortografía y persistieron 3 no conformidades de funcionalidad. Durante la tercera no fueron encontradas no conformidades.

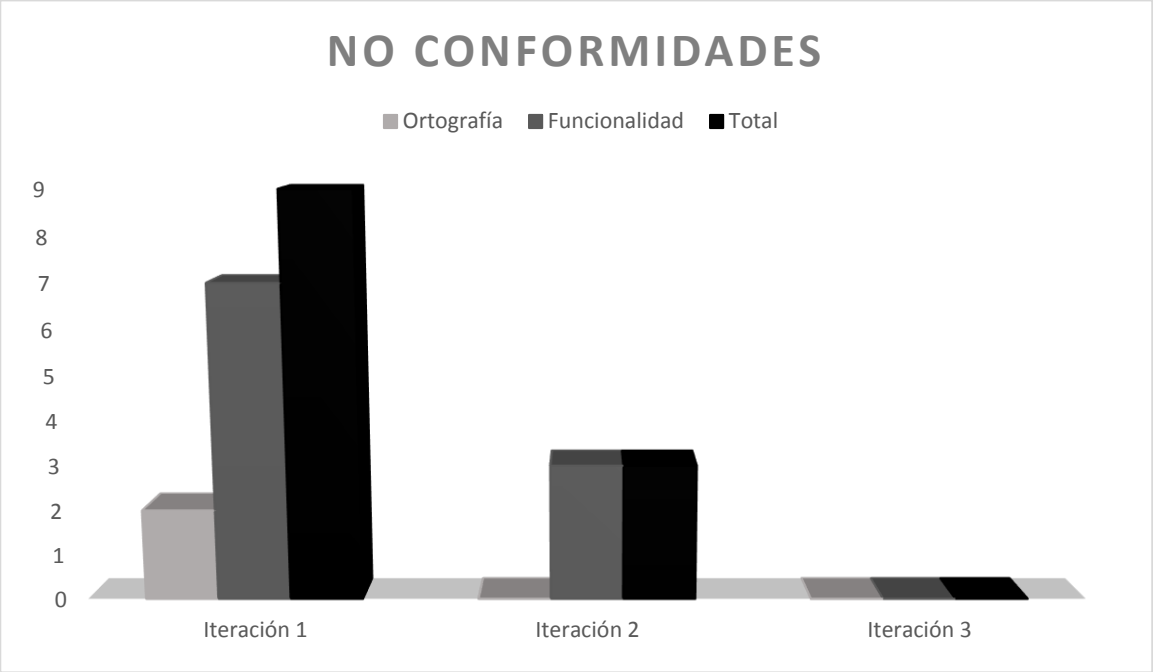


Figura 8: Resultados de las iteraciones de las pruebas funcionales

3.3 Evaluación del índice de satisfacción de usuarios hacia la propuesta de solución

La técnica de IADOV es utilizada para determinar el nivel de satisfacción individual y grupal de los usuarios a partir de una encuesta elaborada. Dicha encuesta fue aplicada a 10 especialistas como objeto de estudio del centro CESOL, ver **Anexo 3**.

La aplicación de la técnica es una vía indirecta para el estudio de satisfacción, los criterios que se utilizan se fundamentan en las relaciones que se establecen entre las tres preguntas cerradas, que se intercalan dentro de un cuestionario y cuya relación el encuestado desconoce. En este caso, en la encuesta, son las preguntas 2, 3 y 4 que se relacionan a través de lo que se denomina el “Cuadro Lógico de Iadov” que se muestra en el **Anexo 2**.

El número resultante de la interrelación de las tres preguntas indica la posición en la escala de satisfacción

siguiente: clara satisfacción (A), más satisfecho que insatisfecho (B), no definida (C), más insatisfecho que satisfecho(D), clara insatisfacción (E) y contradictoria (C).

A partir de la cantidad de respuestas por categoría es posible calcular el Índice de Satisfacción Grupal (ISG) siguiendo la siguiente fórmula:

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

Donde N es la cantidad total de respuestas

Figura 9: Fórmula del Índice de Satisfacción Grupal

El valor del ISG permite identificar las siguientes categorías grupales:

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

El índice general arroja valores entre + 1 y - 1. Los valores que se encuentran comprendidos entre -1 y - 0,5 indican insatisfacción; los comprendidos entre - 0,49 y + 0,49 evidencian contradicción y los que están entre 0,5 y 1 indican que existe satisfacción (42).

1. Los resultados obtenidos de la aplicación de la encuesta se presentan en la Tabla 12.

Tabla 12: Resultados de la escala de satisfacción

(Fuente: elaboración propia)

Categorías grupales de satisfacción	N=10	Escala
1. Clara satisfacción	7	A
2. Más satisfecho que insatisfecho	2	B
3. No definido	0	C

4. Más insatisfecho que satisfecho	1	D
5. Máximo de insatisfacción	0	E
6. Contradictorio	0	C

2. Cálculo del Índice de Satisfacción Grupal

ISG=A (+1) +B (+0.5) +D (-0.5) /N

ISG= (8(+1) +2(+0.5) +1(-0.5)) /10 = 8 + 1 - 0.5/10 = 0.85

3. Interpretación del resultado del ISG.

El valor obtenido del ISG fue 0.85 lo que indica máxima satisfacción de los usuarios con respecto a la herramienta de gestión de copias de seguridad en la Distribución Cubana GNU/Linux Nova. Se puede afirmar que se cumplió el objetivo general de la investigación. Las respuestas a las preguntas abiertas brindadas por los encuestados reafirman los beneficios que traerá la utilización de la herramienta propuesta.

3.4 Aporte de la investigación

La herramienta desarrollada constituye un aporte a la Distribución Cubana de GNU/Linux Nova, ya que ofrece a los usuarios la posibilidad de gestionar un aspecto tan importante como las copias de seguridad, estrechamente relacionadas con la disponibilidad de la información. Este valor agregado permite a los usuarios una rápida recuperación de sus datos ante fallos del sistema o del hardware.

Es decir, la herramienta enriquece la gama de prestaciones que ofrece la Distribución Cubana de GNU/Linux Nova, que a su vez contribuye al proceso de informatización de la sociedad, debido a que se utiliza como la tecnología base para el despliegue y desarrollo de aplicaciones informáticas en los organismos de la administración central del estado. Además, es el software que se instala en las computadoras que se ensamblan en la empresa cubana para la Informática, las comunicaciones y la electrónica, por lo que de esta forma ayuda con lo plateado en los lineamientos 108 y 188 del Partido Comunista de Cuba (PCC), referidos a la soberanía tecnológica y al desarrollo de la industria electrónica y automática respectivamente.

4 Conclusiones del capítulo

La descripción del proceso de implementación de la aplicación, a través de la definición de los estándares de codificación posibilitó una mejor legibilidad del código, haciéndolo más comprensible y estandarizado.

La aplicación de las pruebas unitaria y funcionales permitieron comprobar el correcto funcionamiento del código de la solución, validar la completitud de los requisitos.

Conclusiones

Dándole cumplimiento al objetivo general se concluye:

- El estudio de las herramientas de gestión de copias de seguridad y los diferentes conceptos permitieron establecer las bases teóricas para el posterior desarrollo de la solución e identificar la necesidad de desarrollar una herramienta de gestión de copias de seguridad para la Distribución Cubana de GNU/Linux Nova.
- La identificación de 4 requisitos funcionales y 7 no funcionales, descritos a través de 4 historias de usuario, facilitó significativamente la posterior codificación de la solución.
- La implementación de una herramienta de gestión de copias de seguridad para la Distribución Cubana de GNU/Linux Nova permitió dar respuesta a los requisitos definidos y de esa manera resolver el problema de investigación planteado.
- La implementación de la Herramienta de gestión de copias de seguridad permitió dar respuesta al problema de investigación planteado.
- La aplicación de pruebas a la solución posibilitó la identificación de No Conformidades que fueron resueltas satisfactoriamente, logrando así su correcto funcionamiento y la aceptación por parte del cliente.

Recomendaciones

La herramienta de gestión de copias de seguridad para la Distribución Cubana de GNU/Linux Nova dispone de las funcionalidades necesarias para la creación de copias en cuanto a programas y sus respectivas configuraciones, también para archivos que se ubiquen en la partición donde se encuentra instalada la distribución por lo que se sugieren las siguientes recomendaciones:

- Realizar copias de seguridad incrementales.
- Ofrecer al usuario la posibilidad de encriptar las copias de seguridad.

Referencias bibliográficas

1. **Maxim Y**. The History of Backup [Internet]. [citado 9 de enero de 2018]. Disponible en: http://www.backuphistory.com/
2. **Abadal E, Codina L**. Recuperación de información. Bases Datos Doc Características Funciones Método. 2005;29-92.
3. **Arango R**. Soluciones de backup: diseña tu sistema. Todo Linux Rev Mens Para Entusiastas GNULINUX. 2010;(119):38-41.
4. **Significado de Copia de seguridad** [Internet]. Significados. [citado 25 de mayo de 2018]. Disponible en: http://www.significados.com/copia-de-seguridad/
5. **Acerca de | Portal web para Nova** [Internet]. [citado 24 de mayo de 2018]. Disponible en: https://www.nova.cu/es/contenido/acerca-de
6. **Copias de seguridad y mucho más** [Internet]. 2017 [citado 23 de noviembre de 2017]. Disponible en: https://ubunlog.com/systemback-otra-herramienta-util-para-copias-de-seguridad-y-mas/
7. **Gómez Pinzón EE, Rodriguez Barreto FJ**. Diseño de un tutor para generar copias de seguridad y políticas de backup y recuperación RMAN (recovery Manager). 2016;
8. **Dordoigne J. Redes informáticas**-Nociones fundamentales (5ª edición):(Protocolos, Arquitecturas, Redes inalámbricas, Virtualización, Seguridad, IP v6...). Ediciones ENI; 2015.
9. **El Árbol de Directorios de Linux al Detalle. Principales Carpetas y Sus Funciones** [Internet]. ComputerNewAge. 2015 [citado 27 de mayo de 2018]. Disponible en: https://computernewage.com/2015/06/14/el-arbol-de-directorios-de-linux-al-detalle-que-contiene-cada-carpeta/
10. **Areca Backup** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/arecabackup/
11. **BackupPC** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/backuppc/
12. **Bacula** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/bacula/
13. **fwbackups** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/fwbackups/
14. **Keep** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en:

https://www.linuxlinks.com/keep/
15. **Simple Backup Solution** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/simplebackupsolution/
16. **afbackup** - LinuxLinks [Internet]. [citado 31 de mayo de 2018]. Disponible en: http://www.linuxlinks.com/afbackup/
17. **Amanda** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/amanda/
18. **Cedar Backup** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/cedarbackup/
19. **Duplicity** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/duplicity/
20. **tar** [Internet]. LinuxLinks. [citado 31 de mayo de 2018]. Disponible en: https://www.linuxlinks.com/tar/
21. **Rsync** - EcuRed [Internet]. [citado 14 de junio de 2018]. Disponible en: https://www.ecured.cu/Rsync
22. **The GTK+ Project** [Internet]. [citado 14 de mayo de 2018]. Disponible en: https://www.gtk.org/
23. **Glade** - A User Interface Designer [Internet]. [citado 14 de mayo de 2018]. Disponible en: https://glade.gnome.org/
24. **Van Rossum G, Drake FL**. Python language reference manual. Network Theory; 2003.
25. **BeginnersGuide** - Python Wiki [Internet]. [citado 14 de mayo de 2018]. Disponible en: https://wiki.python.org/moin/BeginnersGuide
26. **Features - PyCharm** [Internet]. [citado 14 de mayo de 2018]. Disponible en: https://www.jetbrains.com/pycharm/features/
27. **Larman C. UML y Patrones**. Pearson Educación ^ eMadrid Madrid; 2003.
28. **Visual Paradigm - UML, Agile, PMBOK, TOGAF, BPMN and More!** [Internet]. [citado 14 de mayo de 2018]. Disponible en: https://www.visual-paradigm.com/features/
29. **Visual Paradigm Product Overview** [Internet]. [citado 14 de mayo de 2018]. Disponible en : https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html
30. **Git** [Internet]. [citado 14 de mayo de 2018]. Disponible en: https://git-scm.com/
31. **eXcriba »** Detalles del documento [Internet]. [citado 31 de mayo de 2018]. Disponible en: https://excriba.prod.uci.cu/page/context/shared/document-details?nodeRef=workspace://SpacesStore/a622adab-eac5-4fb3-ba08-a266767fff5f

32. **Pressman, R.S. Software Engineering**. A practitioner´s Approach. 7th ed. New York: McGraw Hill; 2010.
33. **Vélez Mejía LJ**. Estudio empírico sobre el proceso y la productividad de la ingeniería de requisitos en las empresas antioqueñas de software. Universidad EAFIT; 2014.
34. **Sommerville, I**. Ingeniería del Software. 9na ed. Mexico: Addison Wesley. Pearson Education, Inc.; 2011.
35. **Canós JH, Letelier MCPP**. Metodologías ágiles en el desarrollo de software. 2012;
36. **Pardo Aguilar C, García Peñalvo** FJ. Diagramas de Clase en UML 1.1. 1988;
37. **Cosmin PI**. Patrones de Diseño. MoleQla Rev Cienc Univ Pablo Olavide. 2016;(23):36.
38. **Van Rossum G, Warsaw B, Coghlan N**. PEP 8: style guide for Python code. Python Org. 2001;
39. **Zhang C, Feng W, Steffens E**, de Landaluce A, Kleinman S, LeBlanc MD. Lexos 2017: building reliable software in python. J Comput Sci Coll. 2018;33(6):124-34.
40. **Fernández Sanz L.** Un sondeo sobre la práctica actual de pruebas de software en España. REICIS Rev Esp Innov Calid E Ing Softw. 2005;1(2).
41. **Ochoa F, Camilo J**. Metodología para testing de software basado en componentes. 2010;
42. **López Rodríguez A, Maura VG**. La técnica de Iadov. Una Apl Para El Estud Satisf Los Alumnos Por Las Cl De. 2002;

Anexos

Anexo 1: Plan de preguntas realizadas a los especialistas de CESOL.

No.	Preguntas
1	¿Cómo se puede recuperar correctamente la información?
2	¿Cuáles son las funcionalidades principales con las que debe contar una herramienta de copias de seguridad?
3	¿Debería elegir el usuario la ruta donde almacenar la copia de seguridad?
4	¿Porqué el GRUB durante la restauración debe ser actualizado?
5	¿Cómo se realizan las copia de seguridad actualmente en la distribución?

Anexo 2: Tabla de IADOV.

4. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la solución desarrollada.	2. ¿Actualmente se puede gestionar copia de seguridad en la Distribución Cubana GNU/Linux Nova?								
	No			**No sé**			**Sí**		
	3. ¿Considera usted factible la implementación de una herramienta de gestión de copias de seguridad en la Distribución Cubana GNU/Linux Nova?								
	Sí	**No sé**	**No**	**Sí**	**No sé**	**No**	**Sí**	**No sé**	**No**
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	3	4
No me gusta nada	6	6	6	6	4	4	6	6	5
No sé decir	2	3	6	3	3	3	6	6	4

Anexo 3: Encuesta de satisfacción.

Especialista, le invito a responder el siguiente cuestionario con el fin de conseguir su colaboración para la presente investigación, solicito que exprese en sus respuestas criterios verídicos que guíen la investigación. Marque con una X en una sola opción y en el caso de la 5 responda brevemente. Por el tiempo brindado, muchas gracias.

1. ¿Considera usted apropiado poder gestionar copias de seguridad en la Distribución Cubana GNU/Linux Nova?

   Sí ___ No ___ No sé___

2. ¿Actualmente se puede gestionar copias de seguridad en la Distribución Cubana GNU/Linux Nova?

   Sí ___ No ___ No sé___

3. ¿Considera usted factible la implementación de una herramienta de gestión de copias de seguridad en la Distribución Cubana GNU/Linux Nova?

   Sí ___ No ___ No sé___

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

__Me gusta mucho __Me disgusta más de lo que me gusta __Me gusta más de lo que me disgusta __No me gusta nada __Me da lo mismo __No sé decir

5. ¿Qué opina usted acerca de los beneficios que traería para la universidad disponer de una herramienta de gestión de copias de seguridad en la Distribución Cubana GNU/Linux Nova?