Universidad de las Ciencias Informáticas

Facultad 1

Título: Herramienta para consumir paquetes de repositorios comprimidos en la distribución cubana de GNU/Linux Nova.

Autor:

Frank Ernesto Ramos Sendiña.

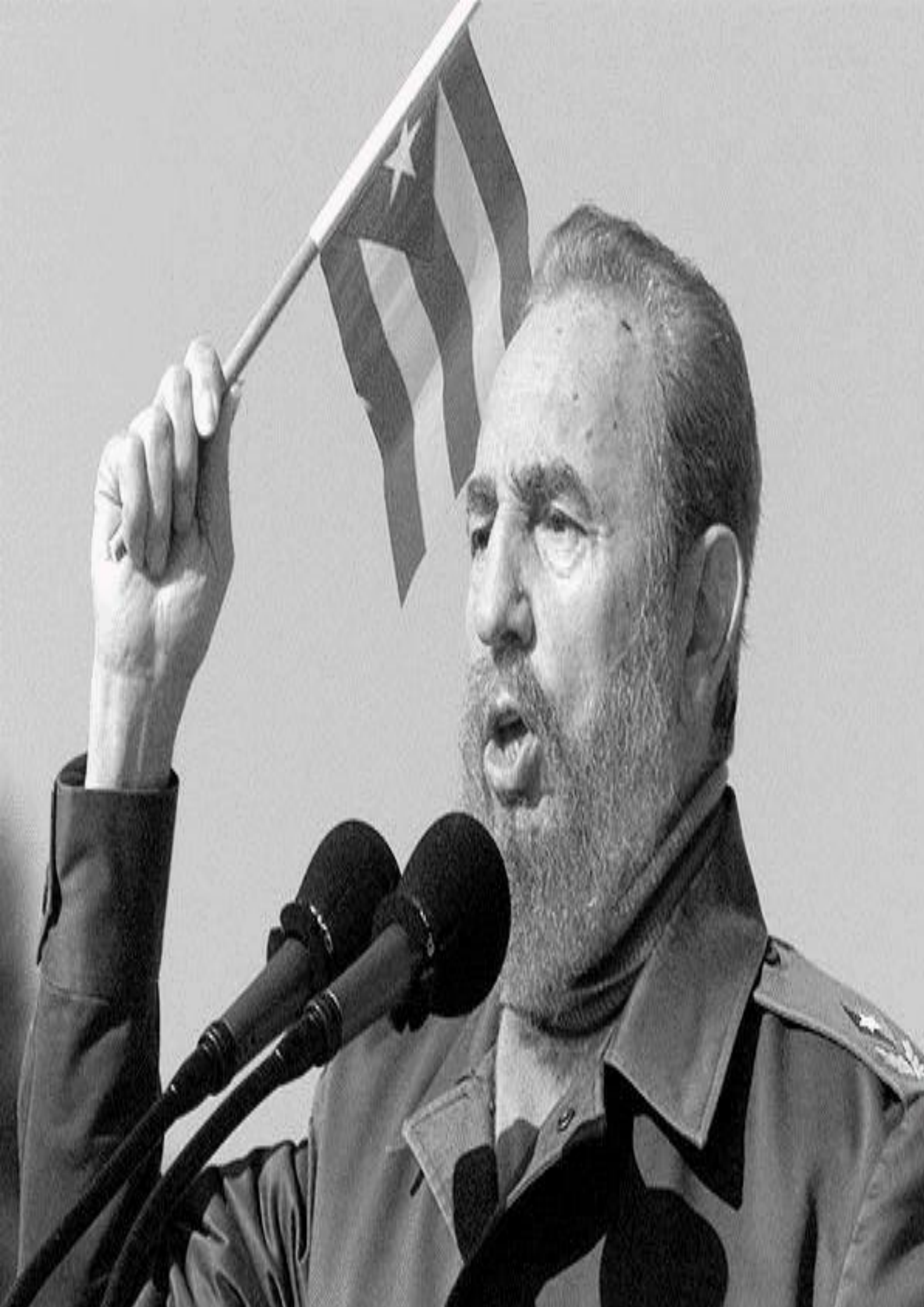
Tutores:

Ing. Gladys Marsi Peñalver Romero

Ing. Manuel Enrique Peiso Cruz

La Habana, Julio de 2017

“Año 59 del Triunfo de la Revolución”



Declaración de Auditoria

Declaro ser el único autor de este trabajo y concedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo. Para que así conste firmo la presente a los ___ días del mes de _____ del año 2017.

_____

Autor: Frank Ernesto Ramos Sendiña

_____

Tutores:

Ing. Gladys Marsi Peñalver Romero

Ing. Manuel Enrique Peiso Cruz

Dedicatoria

A mis padres por siempre estar presente cuando verdaderamente los necesité y a todos mis familiares y amigos.

Al nuestro Comandante en Jefe Fidel Castro Ruz por darme la oportunidad de estudiar esta carrera.

Agradecimientos:

A todas aquellas personas que influyeron de una forma u otra en la realización de mi trabajo de diploma, en especial a mi mamá la persona más importante en mi vida ya que siempre se ha preocupado por mí, por siempre estar apoyándome a pesar de la distancia, por todo el amor que me brinda.

También a mi padre que siempre ha estado presente en todos los momentos.

A mi tía Morchen y a mi prima Chavelys por siempre ayudarme.

A mi novia Liliana por tener tanta paciencia y motivarme cuando en verdad lo necesitaba.

A mi suegra por todo el cariño de madre que me brindó.

Un agradecimiento enorme a mis tutores Manuel Enrique Peiso Cruz y Gladys Marsi Peñalver Romero, gracias a la preocupación y la gran ayuda que me dieron a lo largo de mi tesis.

A toda mi familia y a mis amigos.

Resumen

Actualmente la distribución cubana de GNU/Linux Nova, no cuenta con un procedimiento para la obtención de la información de los repositorios de código fuente y binarios comprimidos en la web, lo que imposibilita su publicación y puesta en funcionamiento. Por lo que el objetivo de la presente investigación consiste en desarrollar una herramienta para consumir paquetes de repositorios comprimidos en la distribución cubana de GNU/Linux Nova. Para ello se realiza el estudio de las herramientas para la obtención de información de los repositorios de código fuente y binarios. Se define como me como metodología de desarrollo la variación de AUP para la UCI la cual guiará el proceso de construcción de la herramienta. Se utilizó el editor de texto Geany y el lenguaje de programación Bash. Se oltuvo como resultado de la investigación una herramienta para consumir paquetes de repositorios comprimidos en la distribución cubana de GNU/Linux Nova.

Palabras claves: paquetes, repositorio, comprimido, Nova.

Índice

Introducción

El surgimiento y la evolución de las tecnologías de la información y las comunicaciones (TIC) crean un escenario de cambio permanente, donde la rápida capacidad de adaptación e innovación es la clave para el éxito de cualquier organización. En los momentos actuales el uso de software libre es uno de los fieles ejemplos de innovación en el campo de la informática, caracterizado por la libertad de usar, copiar, estudiar, modificar y redistribuir las modificaciones del software**,** es por ello que cada vez son más los interesados por este tipo de tecnología ya que de esta forma disminuyen las dependencias tecnológicas que afectan, principalmente, a los países del tercer mundo.

Cuba ha tenido que ejercer una serie de cambios e innovaciones en las tecnologías de la información y las comunicaciones, ya que no puede apostarse por los sistemas operativos privativos como un camino viable para el desarrollo tecnológico, y que a la vez conduzca a la independencia en este mismo ámbito. Factores económicos, de obtención de licencias, de adquisición de software privativo a través de internet y la posibilidad latente de reclamaciones por parte de algunos fabricantes, entre muchas otras complejidades, conducen a optar por otra modalidad de software, menos atado a restricciones legales y más accesible desde el punto de vista económico**.** Por estas razones surge el proceso de migración hacia software libre en el país el cual tiene como pilar fundamental el desarrollo de la distribución cubana de GNU/Linux Nova. Tarea llevada a cabo por el Centro de Soluciones Libres (CESOL) en la Facultad 1. Dicha distribución, para la instalación de los paquetes, usa repositorios de código fuente o binarios que no son más que sitios centralizados donde se almacena y se mantiene información referente a aplicaciones y su código fuente. Estos contienen un gran volumen de información por lo que ocupan mucho espacio, ya que día a día se crean incontables aplicaciones que se almacenan en los mismos; por lo que se hace necesario encontrar una manera viable de hacerlos más pequeños.

El pasado curso 2015-2016 se desarrolló una aplicación para la compresión de los repositorios, destinada a los administradores del repositorios de código fuente y binarios; por lo que se logró reducir el tamaño que los mismos ocupaban. Pero no se definió la manera de publicar la información del repositorio comprimido por los administradores ni un procedimiento para que el cliente pudiera consumir sus aplicaciones. Lo cual tuvo como consecuencia que el resultado práctico antes mencionado, nunca fue puesto en práctica debido a la carencia de una herramienta en la distribución cubana de GNU/Linux Nova que realice estas funciones y se siguen utilizando repositorios de código fuente y binarios de grandes tamaños para la instalación de los paquetes.

Las dificultades anteriormente descritas permiten identificar el siguiente **problema de investigación**: ¿Cómo permitir la utilización de paquetes de repositorios de código fuente y binarios comprimidos en la distribución cubana de GNU/Linux Nova?

Se trazó como **objeto de estudio**: el proceso de obtención de información a partir de repositorios de código fuente y binarios en la web.

Para el desarrollo de la investigación se concibe como **objetivo general:** Desarrollar una herramienta que permita consumir paquetes de repositorios comprimidos en la web desde la distribución cubana de GNU/Linux Nova, para facilitar la obtención de información a partir de repositorio de código fuente y binarios.

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

1. Caracterizar herramientas de obtención de información a partir de repositorios de código fuente y binarios, protocolos de publicación de información en la web y formatos de compresión de información.
2. Caracterizar la metodología de desarrollo y las tecnologías para la realización de la herramienta para consumir paquetes de repositorios comprimidos de la distribución cubana de GNU/Linux Nova.
3. Analizar y diseñar una herramienta para consumir paquetes de los repositorios comprimidos de la distribución cubana de GNU/Linux Nova
4. Implementar una herramienta para consumir paquetes de los repositorios comprimidos de la distribución cubana de GNU/Linux Nova.
5. Evaluar la solución propuesta.

Se define como **campo de acción:** el proceso de obtención de información a partir de repositorios de código fuente y binarios en la distribución cubana de GNU/Linux Nova.

Preguntas Científicas:

1- ¿Cuáles son las tendencias actuales relacionadas con las herramientas de obtención de información a partir de repositorios de código fuente y binarios, protocolos de publicación de información en la web y formatos de compresión de información?

2- ¿Qué tecnologías y metodología se requieren para implementar la propuesta de solución?

3- ¿Cómo implementar y evaluar la herramienta para consumir paquetes de los repositorios comprimidos de la distribución cubana de GNU/Linux Nova?

Para responder a las preguntas científicas se utilizaron algunos **métodos científicos**:

Métodos teóricos:

Histórico-Lógico: empleado con el objetivo de caracterizar las herramientas para consumir paquetes de repositorios de código fuente y binarios en la web, los protocolos de publicación de información en la web y formatos de compresión de información; obteniendo como resultado el procedimiento para publicar la información en la web así como la manera de consumirla lo que permite obtener los elementos necesarios para el desarrollo de la herramienta para consumir paquetes de los repositorios comprimidos de la distribución cubana de GNU/Linux Nova. Además permite obtener la información útil para la selección de las tecnologías y la metodología de desarrollo a utilizar.

La utilización del método **Analítico-Sintético** posibilita durante toda la investigación el análisis de fuentes relevantes relacionadas con las herramientas para consumir paquetes de repositorios de código fuentes y binarios en la web, los protocolos de publicación de información en la web y formatos de compresión de información; y escoger la información útil para la selección de las tecnologías y la metodología de desarrollo para la realización de la propuesta de solución.

Estructura del Documento:

Queda descrito el contenido de este documento en una introducción, tres capítulos, conclusiones, recomendaciones y referencias bibliográficas. A continuación se describen cada uno de los capítulos:

Capítulo 1 Fundamentación teórica: En este capítulo se identifican los conceptos relacionados con el tema de investigación. Se realiza un análisis de las principales características y deficiencias de las herramientas de obtención en la web, los protocolos de publicación de información para definir los elementos que se van a tener en cuenta para la propuesta de solución. Se seleccionan las herramientas que serán utilizadas para desarrollar la solución y se define la metodología a seguir durante el proceso de construcción de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana GNU/Linux Nova.

Capítulo 2 Análisis y diseño de la aplicación: Se realizan las fases de inicio y de ejecución de la metodología definida. Se describe la propuesta de solución para el desarrollo de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana GNU/Linux Nova. Se identifican las características y cualidades de la herramienta descritas mediante la utilización de las Historias de Usuarios (HU), así como la definición de los elementos de análisis y diseño identificando la arquitectura de software a utilizar.

Capítulo 3 Implementación y pruebas: En el presente capítulo se continúa con la fase de ejecución de la metodología escogida. Se planifica la implementación de las HU definidas donde se describen las funcionalidades de la herramienta, se define los estándares de codificación. Se definen la estrategia de prueba a utilizar, identificando tipos, métodos y niveles de pruebas para verificar el funcionamiento de la solución. Para ello se describen los diseños de casos de pruebas y finalmente se muestran los resultados de las pruebas realizadas y el impacto social de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana GNU/Linux Nova.

Posibles resultados:

1. Formato de compresión para ser utilizado en los repositorios de código fuente y binarios en la distribución cubana de GNU/Linux Nova.
2. Procedimiento para publicar el repositorio de código fuente y binarios comprimidos en la web.
3. Obtención de una herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos en la distribución cubana de GNU/Linux Nova.

Capítulo 1 Fundamentación teórica

Introducción

En este capítulo se identifican los conceptos relacionados con el tema de investigación. Se realiza un análisis de las principales características y deficiencias de las herramientas de obtención en la web, los protocolos de publicación de información para definir los elementos que se van a tener en cuenta para la propuesta de solución. Se seleccionan las herramientas que serán utilizadas para desarrollar la solución y se define la metodología a seguir durante el proceso de construcción de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana GNU/Linux Nova.

1.1 Conceptos Fundamentales

Paquetes de software

Es un grupo de uno o más archivos que son necesarios tanto para la ejecución de un programa de computadora como para agregar características a un programa ya instalado.

Los paquetes de software, pueden estar en un formato estandarizado, que le permite ser instalado por un programa que está integrado en el sistema operativo, o puede ser un instalador autosuficiente (no necesita otros programas), generalmente conocido como "instalador" **[1].**

Repositorio

Un repositorio, es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos, también se suele decir que un repositorio es un lugar en Internet donde se almacena información, en el caso de los repositorios GNU/Linux esta información son programas **[2].**

Repositorio GNU/Linux

Un repositorio de GNU/Linux[1] es una colección de paquetes de programas de una distribución de GNU/Linux específica que generalmente contiene archivos binarios pre compilado que pueden ser descargados e instalados por los usuarios de la distribución correspondiente. Es posible también encontrar paquetes de código fuente **[3].**

Código Binario

Un código binario representa un texto o instrucciones de procesador de computadora usando el sistema de números binarios de dos dígitos: 0 y 1. Un código binario asigna una cadena de bits a cada símbolo (carácter) o a cada instrucción, respectivamente [4].

Código Fuente

En el contexto de la informática, el código fuente se define como el conjunto de líneas de textos, que son las directrices que debe seguir la computadora para realizar dicho programa; por lo que es en el código fuente, donde se encuentra escrito el funcionamiento de la computadora.

El código fuente de un programa está escrito en un lenguaje de programación determinado, sin embargo este tipo de lenguaje no puede ser ejecutado directamente por la computadora, sino que debe ser traducido a otro lenguaje que el ordenador pueda ejecutar más fácilmente. Para esta traducción se emplean los llamados compiladores[2] [5].

Gestión de paquetes

Un sistema de gestión de paquetes, es un grupo de herramientas que se emplean para automatizar el proceso de instalación, actualización, configuración y borrado de paquetes de software. El término se emplea generalmente para hacer referencia a los gestores de paquetes en sistemas basados en Unix, porque estos tipos de sistemas operativos son los que suelen apoyar los sistemas de gestión de paquetes. Son muy útiles para administrar los paquetes, pues estos sistemas operativos libres suelen contar con miles de paquetes de software [6].

---

[1] Sistema operativo libre homólogo a Unix que usualmente utiliza herramientas de Sistema GNU

[2] Es el programa que se encarga de llevar un nivel de programación a código binario para ser ejecutado por el procesador de la computadora.

Compresión de archivos

La compresión de archivos es la manera de reducir el espacio que ocupa un fichero en el disco duro de una computadora, Tablet u otro dispositivo.

1.2 Herramientas de obtención de información a partir de un repositorio de código fuente y código binarios publicado en la web.

Se tuvieron en cuenta dos herramientas para la obtención de información a partir de repositorios: Advanced Packaging Tool (APT) y Yellow dog Updater, Modified (YUM).

Advanced Packaging Tool ((APT) o Herramienta Avanzada de Empaquetado), es un sistema de gestión de paquetes creado por el proyecto Debian y es usado por todas sus distribuciones. APT es una aplicación que se ejecuta en el terminal (o consola). Suele ser necesaria la conexión a internet aunque a veces también funcione sin ella. Las tareas que puede realizar APT son las siguientes:

- Buscar paquetes en internet o localmente
- Solucionar dependencias (algunas veces para que funcione una aplicación se necesitan otras, a esto se le denomina dependencia).
- Descargar de internet aplicaciones, dependencias o paquetes
- Instalar en orden correcto paquetes y dependencias [7].

Yellow dog Updater, Modified (YUM) es un gestor de paquetes RPM que permite la instalación y desinstalación de aplicaciones y actualización del sistema. Asigna a los usuarios normales la capacidad de poder consultar sin modificar, esta posibilidad sola la brinda cuando se es superusuario o root (administrador) del sistema [8].

Es una fácil tarea la instalación de unos o dos paquetes manual, pero enseguida se vuelve una compleja tarea cuando sean un número mayor de paquetes, es por esto que la mayoría de los sistemas operativos libres actuales incluyen un gestor de paquetes (APT o YUM) para la instalación de sus aplicaciones, que proporciona una base común para construir estos programas.

Selección de la herramienta de obtención de información

Definidas las herramientas se llegó a la conclusión que no se escogerá ninguna de las anteriormente descritas, ya que no le dan solución al problema trazado, a pesar de que las mismas realizan todo el proceso de gestión de la paquetería en las máquinas clientes

que consumen los repositorios, no cuentan con ninguna funcionalidad referente a la utilización de repositorios comprimidos. Esto no significa que no sea posible utilizar varias de sus funcionalidades en el desarrollo de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana de GNU/Linux Nova tales como: buscar paquetes en internet o localmente, solucionar dependencias, descargar de internet aplicaciones, dependencias o paquetes e instalar en orden correcto paquetes y dependencias.

1.3 Formatos de Compresión

La distribución cubana de GNU/Linux Nova dispone de varios formatos a la hora de comprimir archivos. Estos formatos reducen el espacio en disco de los ficheros y hacen que su envío sea más cómodo. Los más usados por esta son .zip, tar.bz2, tar.gz, 7zip y .squashfs [8].

.zip

Es un formato de fichero bastante simple, que comprime cada uno de los archivos de forma separada. Comprimir cada archivo independientemente del resto de archivos comprimidos permite recuperar cada uno de los ficheros sin tener que leer el resto, lo que aumenta el rendimiento.

El problema, es que el resultado de agrupar un número grande de pequeños archivos es siempre mayor que agrupar todos los archivos y comprimirlos como si fuera uno sólo. zip soporta un sistema de cifrado simétrico basado en una clave única. Sin embargo, este sistema de cifrado es débil ante ataques de fuerza bruta [10].

.tar.gz

La extensión de archivo tar comprimido con gzip (tar.gz) es similar a un archivo zip tradicional. Sin embargo, este tipo de extensión tiene la habilidad de gestionar archivos de mucho mayor tamaño que el formato zip. Estos archivos también son conocidos como gzip. Debido a su origen, los archivos .tar.gz pueden manejarse con facilidad y extraerse en sistemas operativos Mac o Linux. Sin embargo, Windows no es capaz de gestionar los archivos tar.gz sin ayuda.

Otra diferencia es que los archivos zip pueden contener múltiples archivos en su compresión, mientras que la extensión tar.gz solo permite la compresión de un único archivo de gran tamaño [11].

.tar.bz2

El formato de compresión bz2 o bzip2, desarrollado y mantenido por Julian Seward, utiliza los algoritmos de compresión de Burrows-Wheeler y el algoritmo de codificación de Huffman. Aunque el porcentaje de compresión de los archivos depende del contenido de éstos mismos, resulta una mejor alternativa a zip y gzip, pero con un mayor consumo de memoria y recursos de sistema. De todos los formatos es sin duda el que tiene mayor radio de compresión. Aunque eso le conlleva más lentitud en el proceso y mayor consumo de recursos [12].

.7zip

7-zip es un formato de compresión de archivos que logra el mayor radio o grado de compresión. En GNU/Linux, el paquete p7zip incluye la herramienta 7za. La aplicación soporta los formatos 7z (que implementa el algoritmo de compresión lzma), zip, cab, arj, gzip, bzip2, tar, cpio, rpm y deb. El radio de compresión en el nuevo formato 7z es un 30-50% mejor que en el formato zip [13].

.squashfs

Es un sistema de archivos comprimido de solo lectura para los sistemas basados en Linux. Squashfs comprime archivos, i-nodos y directorios, y soporta tamaños de bloque de hasta 1024 KB para mayor compresión. También es software libre (licenciado como GPL).

Squashfs está pensado para su uso como sistema de archivos genérico de solo lectura y en dispositivos de bloques/sistemas de memoria limitados (por ejemplo, sistemas embebidos), donde se requiere poca sobrecarga. La versión estándar de squashfs utiliza compresión mediante gzip [14].

Selección del formato de compresión

Para seleccionar el formato de compresión más adecuado para comprimir el repositorio de la distribución cubana de GNU/Linux Nova se tuvo en cuenta un aspecto de gran importancia, la máxima reducción del espacio que ocupaban dichos repositorios en el

disco así como la rapidez de montaje de dichos formatos. Para ellos se establece la siguiente función objetivo:

F(x)=min(ELC+VM) donde x es el formato de compresión utilizado, ELC el espacio luego de la compresión y VM la velocidad de montaje.

Formato de compresión	Espacio antes de comprimirlo	Espacio después de comprimirlo	Velocidad De montar la información	F(X)
.tar.gz	64.8 GB	58.4 GB	30min	88.4
.squashfs	64.8 GB	55 GB	0.1min	55.1
.zip	64.8 GB	61.9 GB	10min	71.9
.7zip	64.8 GB	56 GB	4min	60
.tar.bz2	64.8 GB	62 GB	5min	67

TABLA 1: SELECCIÓN DEL FORMATO DE COMPRESIÓN

Cumpliendo este aspecto se arriba a la conclusión que el formato .squashfs es el más convenientes para la selección de compresión del repositorio teniendo en cuenta que es la frontera inferior luego de calculada la función objetivo para cada uno de ellos.

1.4 Protocolos de red para transferencia de archivos

Para el desarrollo de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana de GNU/Linux Nova, se hace necesario encontrar una forma de publicar el repositorio que posibilite una descarga rápida y sin pérdidas de información de los mismos. A continuación se caracterizan los protocolos más usados para publicar información.

File Transfer Protocol (FTP)

Es un protocolo de red para la transferencia de archivos entre sistemas interconectados o enlazados a Internet, basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar o enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

El funcionamiento es relativamente sencillo. Una persona desde su computadora invoca un programa cliente FTP para conectar con otro computador/servidor, que a su vez tiene instalado el programa servidor FTP. Una vez establecida la conexión y debidamente

autenticado el usuario con su contraseña, se pueden empezar a intercambiar archivos de todo tipo.

Los beneficios de contar con un servicio FTP son bastantes, entre otros se encuentran la facilidad para la transferencia de altos volúmenes de información, velocidad y estabilidad de los enlaces, seguridad en la transferencia de información, bajos costos de implementación y flexibilidad en configuración de cuotas, usuarios y permisos de acceso.

Para contar con la posibilidad de manejar servicios FTP se requieren básicamente dos elementos principales, un servidor FTP y un cliente FTP [15].

Network File System (NFS)

Las siglas NFS significan Sistema de Archivos de Red (del inglés Network File System) y fue desarrollado por SUN Microsystems en 1984. Su función en una red es permitir que un equipo GNU/Linux pueda montar y trabajar con un sistema de archivos de otro equipo de la red como si fuera local.

El servicio NFS utiliza las llamadas a procedimientos remotos basadas en el protocolo RPC (del inglés, Remote Procedure Call) que permite desde un equipo (cliente) ejecutar código ubicado en otro equipo remoto (servidor) mediante el establecimiento de sockets (IP + puerto) entre ambas.

Aunque al servicio se le suele conocer con el nombre NFS, realmente NFS es un protocolo de nivel de aplicación y por debajo, el protocolo subyacente que utiliza NFS son las Llamadas a Procedimientos Remotos (RPC) de nivel de sesión, también utiliza TCP/UDP en el nivel Transporte e IP en el nivel de red.

NFS es un protocolo sin estado (state-less) en algunas de sus versiones. Es decir, el servidor no recuerda las solicitudes anteriores. Por tanto, cada llamada a un procedimiento contiene toda la información necesaria para su finalización. Si el servidor NFS falla, el sistema cliente repetirá las solicitudes de NFS hasta que obtenga una respuesta. Además, el servidor no realiza tareas de recuperación frente a fallos [17].

Ventajas

1 Los datos accedidos por todo tipo de usuarios pueden mantenerse en un nodo central, con clientes que montan los directorios en el momento de arrancar.

2 Los datos que consumen grandes cantidades de espacio de disco pueden mantenerse en un nodo.
3 Los datos de administración pueden también mantenerse en un solo nodo. Ya no será necesario usar RPC para instalar el mismo fichero en 20 máquinas distintas [18].
4 Reducen el riesgo de que el fallo de un solo equipo impida acceder a los datos. La mayor parte de los sistemas de archivos en red permiten ingresar en múltiples equipos y acceder a los datos exactamente de la misma forma en todos ellos.
5 Proporcionan ubicaciones centralizadas para los datos que deben o deberían estar compartidas entre todos los usuarios [19].

Desventajas

1. NFSv2 y NFSv3 pueden utilizar UDP como protocolo de transporte que, al ser no orientado a la conexión, minimiza el tráfico de red, pero si el servidor NFS dejara de funcionar por cualquier circunstancia, los clientes NFS seguirían enviando peticiones al servidor produciendo el efecto contrario, que es la saturación de la red.
2. Las versiones 2 y 3 de NFS permiten controlar la exportación y montaje de sistemas de archivos en función del equipo que hace la solicitud, pero no del usuario. Es decir, no se contempla un control de acceso al sistema de archivos por usuario. Sólo para los equipos. Esto implica que, si un sistema de archivos es exportado desde el servidor NFS, cualquier usuario de un equipo remoto cliente NFS podría acceder a él.

Hyper Text Transfer Protocol (HTTP)

Es un protocolo de transferencia donde se utiliza un sistema mediante el cual se permite la transferencia de información entre diferentes servicios y los clientes que utilizan páginas web. Este sistema fue desarrollado por las instituciones internacionales *World Wide Web Consortium* y la *Internet Engineering Task Force*, finalizado en el año de 1999 [20].

HTTP con una ‘S’ añadida al final, hace referencia a “*Secure Sockets Layer*” otro importante protocolo desarrollado para realizar transferencias de forma segura en Internet usando nuestro navegador [21].

Ventajas

1 Proporcionan un método increíblemente simple para subir archivos a un servidor, con un mínimo de conocimiento sobre transferencias de archivos.
2 Descargar un archivo también es increíblemente fácil.

Desventajas

1 Carencia de potencia cuando se trata de cargar archivos [22].

Selección del protocolo para la publicación de repositorios comprimidos

Para la selección del protocolo a utilizar se tendrán en cuenta, las características antes mencionadas así como la velocidad en que los mismos se demoran para obtener la información del repositorio comprimido con squashfs. A continuación se presentan los datos de la velocidad en forma de tabla comparativa.

Paquete	Protocolo FTP(Duración para montar un fichero)	Protocolo NFS(Duración para montar un fichero)	Protocolo HTTP (Duración en montar el repositorio)
2015.squashfs (55.0 gb)	Aproximadamente 2horas	0.1seg	Aproximadamente1hora

TABLA 2: SELECCIÓN DEL SERVIDOR DE MONTAJE

Después de lo anteriormente planteado se aprecia que el protocolo NFS permite montar la información que posee en la computadora cliente sin necesidad de descargarla, esta es una característica que no posee el resto y que es directamente proporcional a la velocidad en que se obtendrá la información publicada, quedando claro en la tabla antes presentada. Los demás protocolos tienen que realizar una operación de descarga del fichero para dicha utilización lo que provoca que a mayor tamaño de la información, mayor tiempo de descarga de la misma. A pesar de que NFS no posibilita una autenticación a nivel de usuario, la misma no es necesaria para el acceso a la información de un repositorio, por lo que su principal desventaja no da al traste con su utilización. Por todo lo antes planteado se decide escoger el protocolo NFS.

1.5 Tecnologías a utilizar para el desarrollo de la solución propuesta

Modelado

Sommerville define modelado como “Una representación simplificada de un proceso de software, representada desde una perspectiva específica [23].

UML

El Lenguaje de Modelado Unificado (UML Unified Modeling Language por sus siglas en inglés) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Describe el funcionamiento del mismo, sin profundizar en su implementación

Visual Paradigm para UML

Visual Paradigm para UML es una herramienta para el desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos [24]. Se utiliza Visual Paradigm para UML porque ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Documentación automática Ad-hoc.
- Ambiente visualmente superior de modelado.
- Sincronización de código fuente en tiempo real.

Editor de código

Geany

No es un simple editor, es más bien un IDE, como sus autores lo denominan. Carece de creador de interfaces gráficas pero es un entorno de desarrollo de lo más completo. Además de proporcionarnos resaltado de código, atajos de teclado, posibilidad de deshacer/rehacer, sistema de pestaña, opciones que se le pueden exigir a todos los editores actuales [25].

Algunas de las características más destacadas de Geany son:

- Soporte multi-documento.
- Soporte de proyectos.
- Coloreado de sintaxis.
- Emulador de terminal incrustado.

Algunas de las utilidades más conocidas son:

- Compatible con la mayoría de lenguajes.

- Varios paneles para acceder mejor a los datos.
- Herramientas para compilar.
- Buscador integrado.

Ventajas

1. Posibilidad de compilar y ejecutar directamente desde el entorno (en todos los lenguajes orientados a esta labor).
2. Descomposición y representación de las clases y estructuras de nuestro código.
3. Posibilidad de ampliar funcionalidad mediante complementos

Gedit

Es un editor de texto libre oficial del entorno de escritorio de GNOME. Este editor se caracteriza principalmente por su facilidad de uso, conseguida en gran parte gracias a una interfaz gráfica claro y limpio, mostrando únicamente las funcionalidades principales que suelen requerir la mayoría de usuarios.

Características

- Es software libre, de código abierto y gratuito.
- Compatibilidad con textos internacionalizados.
- Coloreado del texto según la sintaxis de varios lenguajes de programación.
- Corrector ortográfico multiidioma.
- Incorporación de plugins para ampliar las funcionalidades básicas del programa.
- Posibilidad de cambiar el color y fuente del texto del editor.
- Numeración de líneas.
- Búsqueda y reemplazo de texto.
- Edición de archivos remotamente.
- Copia de seguridad de los archivos sobre los que se trabaja [26].

Teniendo en cuenta las características planteadas, se decide utilizar Geany por las facilidades para la codificación que el mismo posee, haciendo más fácil el proceso de construcción de la herramienta a desarrollar.

Lenguajes de programación

Lenguaje Bash

Bash es un programa informático cuya función consiste en interpretar órdenes. Está basado en la shell de Unix y es compatible con POSIX. Fue escrito para el proyecto GNU y es el intérprete de comandos por defecto en la mayoría de las distribuciones de Linux. Su nombre es un acrónimo de Bourne-Again Shell, haciendo un juego de palabras (born-again significa renacimiento) sobre el Bourne shell (sh), que fue uno de los primeros intérpretes importantes de Unix [27].

1.6 Metodología de desarrollo de software

Variación Agile Unified Process (AUP) para la UCI

Se decide utilizar la variación de la metodología AUP debido a que es la establecida por la universidad para el proceso productivo. La misma se adapta al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello se apoya en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad [28]. Esta cuenta con 3 fases de vida de los proyectos de la UCI (Inicio, Ejecución, Cierre).

Fases de la metodología AUP

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

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

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

De la metodología seleccionada se escogió el escenario Número 4: Proyectos que no modelen negocio y solo pueden modelar el sistema con HU.

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos y probarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. Todas las disciplinas antes definidas (desde Modelado de negocio hasta Pruebas de Aceptación) se desarrollan en la Fase de Ejecución, de ahí que en la misma se realicen Iteraciones y se obtengan resultados incrementales. En una iteración se repite el flujo de trabajo de las disciplinas, Requisitos, Análisis y diseño, Implementación y Pruebas internas. De esta forma se brinda un resultado más completo para un producto final de manera creciente. Para llegar a lograr esto, cada requisito debe tener un completo desarrollo en una única iteración [28].

La presente investigación utilizará las fases de inicio y ejecución ya que no se realizará el cierre del proyecto, y se decide utilizar el Escenario 4 ya que se adapta a las características de esta teniendo en cuenta que es un proyecto pequeño y de corta duración en el que el cliente estará en constante interacción con los desarrolladores.

Capítulo 2 Análisis y Diseño de la Aplicación

Se realizan las fases de inicio y de ejecución de la metodología definida. Se describe la propuesta de solución para el desarrollo de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana GNU/Linux Nova. Se identifican las características y cualidades de la herramienta descritas mediante la utilización de las Historias de Usuarios (HU), así como la definición de los elementos de análisis y diseño identificando la arquitectura de software a utilizar.

2.1 Propuesta de solución

En el servidor

Primeramente es necesario realizar la compresión del repositorio. Esto se realiza utilizando el formato definido en el capítulo anterior a través del siguiente comando en consola: mksquashfs repositorio/ 2015.squashfs. A continuación es necesario publicar el repositorio comprimido, y para ellos se realiza a través del protocolo NFS, también definido en el pasado capítulo. Se crea el servidor NFS (apt-get install nfs-common nfs-kernel-server), se crea una carpeta de nombre nfs en el directorio /var del sistema de ficheros y dentro se colocan los repositorios comprimidos con squashfs, se comparte la información a través del servidor NFS, esto se realiza configurando el archivo /etc/exports donde se agrega la siguiente línea: /var/nfs *(ro), lo que permite que la información solo esté accesible con permisos de solo lectura desde cualquier ip.

En el cliente

La aplicación estará empaqueteda para poder ser instalada desde el centro de software de la distribución cubana de GNU/Linux Nova. Luego de instalarla esta creará una plantilla vacía del archivo /etc/cmpr/cmpr.conf que es un archivo que la aplicación utiliza para determinar la dirección del repositorio y el *codename* que el mismo posee. Debido a esto el primer paso, luego de instalar la herramienta, es configurar dicho archivo a través de la siguiente línea de comando: cmpr –c, esto abrirá el archivo en el cual primeramente se podrá el ip del servidor, un espacio y luego el *codename* del repositorio. A continuación se listarán las funcionalidades básicas que el sistema realiza:

- cmpr –m: se podrá visualizar el o los repositorios comprimidos que se encuentran publicados en el repositorio configurado.

- cmpr –v: se podrá verificar si realmente el servidor NFS es un servidor correcto.

- cmpr –a: se realizará el proceso de configuración en el sistema del repositorio, este proceso consiste en montar el repositorio en una carpeta local de la máquina, luego los repositorios comprimidos configurados también serán montados en carpetas locales de la máquina con el nombre de dichos repositorios y serán configurados en el archivo /etc/apt/sources.lists de la aplicación apt, como repositorios locales para que puedan ser usados por dicha aplicación. Cada vez que se enciende la máquina si estos repositorios no fueron desmontados la aplicación automáticamente deberá montarlos para poder seguir utilizándolos.

- cmpr –d: se utilizará para desmontar los repositorios montados.

- cmpr –i: se utilizará para realizar proceso de instalación, desinstalación, descarga de código fuente.

2.2 Características y cualidades de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana GNU/Linux Nova

Los **requisitos de un sistema** son declaraciones de servicio que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares como se debe comportar en situaciones particulares. También puede verse como una declaración abstracta de alto nivel de un servicio que el sistema debe proporcionar.

Los **requisitos funcionales**, son los que definen qué debe hacer un sistema y expresan la naturaleza de su funcionamiento (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento) [29].

Nombre de Requisito Funcional	Requisitos Funcionales	Complejidad	Descripción	Prioridad para el cliente
RF_1	Instalar de paquetes desde un repositorio comprimido.	Alta	Instala en el sistema operativo la aplicación seleccionada así como las dependencias necesarias para su correcto funcionamiento.	Alta
RF_2	Montar el repositorio comprimido	Alta	Monta el servidor NFS en una carpeta local del sistema y luego monta los	Alta

	en una carpeta local.		repositorios configurados en el archivo /etc/cmpr/cmpr.conf.	
RF_3	Descargar el código fuente de un paquete del repositorio comprimido.	Media	Descarga el código fuente de una aplicación seleccionada por el usuario.	Media
RF_4	Desmontar el repositorio comprimido	Media	Desmontar los repositorios montados anteriormente.	Media
RF_5	Reinstalar paquetes desde un repositorio comprimido.,	Mala	Desinstala y luego instala una aplicación seleccionada por el usuario en el sistema operativo.	Mala
RF_6	Modificar la dirección del repositorio comprimido	Mala	Modificar en el archivo /etc/cmpr/cmpr.conf el ip del servidor NFS utilizado.	Mala

TABLA 3: LISTADO DE REQUISITOS FUNCIONALES

Un **requisito no funcional** especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requisitos funcionales. Sirven de apoyo a los requisitos funcionales**.**

Requisitos no funcionales de software:

RNF 1: Sistema operativo Distribución Cubana de GNU Linux Nova 5.0.

RNF 2: Uso de la herramienta de código abierto Geany en su versión 7.3.3.

RNF 3: Debe existir un servidor de repositorio para publicar el repositorio comprimido.

Requisitos no funcionales de implementación:

RNF 4. Utilizar como lenguaje de programación Bash.

Requisitos no funcionales de seguridad:

RNF 5: Se publicará la información en el repositorio NFS con permisos de solo lectura.

2.3 Descripción de los requisitos de software

Las Historias de Usuarios (HU) sirven para registrar los requerimientos de los clientes según el negocio y son utilizadas para poder realizar la estimación de cada una de las iteraciones durante la fase de planificación. Las HU son escritas por el equipo de trabajo en conjunto con los clientes en base a lo que se estima que es necesario para el sistema. Están escritas en un formato de oraciones en la terminología del cliente, sin necesidad de sintaxis técnicas.

También son utilizadas para poder crear las pruebas de aceptación. Las HU solo proveen suficiente detalle para poder realizar la estimación de cuánto tardará en ser implementada dicha funcionalidad. Una gran diferencia entre las HU y los documentos tradicionales es que se centran en lo que el cliente necesita.

Las HU utilizan las siguientes variables para describir los requisitos, a continuación se describe qué significa cada una de ellas para una mayor comprensión.

Prioridad en negocio: Prioridad que se le asigna a la historia de usuario en el negocio. Si es importante debe ser implementada lo antes posible. El tipo de prioridad se clasifica en:

- ✓ Alta: Se les otorga a las historias de usuarios que resultan funcionalidades fundamentales en el desarrollo del sistema, las que el cliente define como principales para el control integral del sistema.
- ✓ Media: Se les otorga a las historias de usuarios que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- ✓ Baja: Se les otorga a las historias de usuarios que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

Riesgo en desarrollo: Riesgo que representa para el desarrollo la historia de usuario. En dependencia del riesgo que represente se le asigna un valor que puede ser:

- ✓ Alta: Cuando en la implementación de las historias de usuarios se consideran la posible existencia de errores que lleven la inoperatividad del código.
- ✓ Media: Cuando pueden aparecer errores en la implementación de las historias de usuarios que puedan retrasar la entrega de la versión.

- ✓ Baja: Cuando pueden aparecer errores que serán tratados con relativa facilidad, sin que traigan perjuicios para el desarrollo del proyecto

Tiempo estimado: Cantidad de días que se estima culminada la HU.

Tiempo Real: Cantidad de días en la que se realizó la HU.

<table>
<tr><th colspan="2">Historias de usuarios</th></tr>
<tr><td>Numero: HU-01</td><td>Nombre de la HU: Instalar de paquetes desde un repositorio comprimido.</td></tr>
<tr><td colspan="2">Modificación de la historia de usuario: Ninguna</td></tr>
<tr><td>Programador: Frank E. Ramos Sendiña</td><td>Iteración asignada: 1</td></tr>
<tr><td>Prioridad en el Negocio: Baja

Riesgo en desarrollo: Baja</td><td>Tiempo estimado: 1 día

Tiempo real: 1 día</td></tr>
<tr><td colspan="2">Descripción: El usuario introduce el nombre del paquete que desea instalar, seguido la herramienta verificará la disponibilidad del paquete seleccionado por el usuario en el repositorio.

Escenario 1: Está disponible el paquete seleccionado

Se procede con la instalación del paquete seleccionado por el usuario.

Escenario 2: No está disponible el paquete seleccionado

Se notifica el suceso al usuario.</td></tr>
<tr><td colspan="2">Observaciones:</td></tr>
<tr><td colspan="2">Prototipo de interfaz:</td></tr>
</table>

TABLA 4: HU-01: INSTALAR DE PAQUETES DESDE UN REPOSITORIO COMPRIMIDO.

Numero: HU-02	**Nombre de la HU:** Montar el repositorio comprimido en una carpeta local.
Modificación de la historia de usuario: Ninguna	
Programador: Frank E. Ramos Sendiña	**Iteración asignada:** 1
Prioridad en el Negocio: Alta	**Tiempo estimado:** 3 día
Riesgo en desarrollo: Alta	**Tiempo real:** 2 día
Descripción: El usuario introduce, en el archivo /etc/cmpr/cmpr.conf, el ip y el *codename* del repositorio que desea montar. Escenario 1: El ip especificado es un servidor NFS y exista el repositorio con el *codename* especificado. Se monta el repositorio con el *codename* especificado por el usuario. Escenario 2 El ip especificado es un servidor NFS pero no existe el repositorio con el *codename* especificado. Se le notifica al usuario lo sucedido. Escenario 3: El ip especificado no es un servidor. Se le notifica al usuario lo sucedido.	
Observaciones:	
Prototipo de interfaz:	

TABLA 5: HU-02: MONTAR EL REPOSITORIO COMPRIMIDO EN UNA CARPETA LOCAL.

Historias de usuarios

Numero: HU-03	**Nombre de la HU:** Descargar el código fuente de un paquete del repositorio comprimido.
Modificación de la historia de usuario: Ninguna	
Programador: Frank E. Ramos Sendiña	**Iteración asignada:** 1
Prioridad en el Negocio: Media	**Tiempo estimado:** 1 día
Riesgo en desarrollo: Media	**Tiempo real:** 1 día
Descripción: El usuario introduce el nombre del paquete del cual desea descargar el código fuente. A continuación la herramienta verificará la disponibilidad del paquete seleccionado por el usuario en el repositorio. Escenario 1: Está disponible el paquete seleccionado Se procede con la descarga del código fuente del paquete seleccionado por el usuario. Escenario 2: No está disponible el paquete seleccionado Se notifica el suceso al usuario.	
Observaciones:	
Prototipo de interfaz:	

TABLA 6: HU-03: DESCARGAR EL CÓDIGO FUENTE DE UN PAQUETE DEL REPOSITORIO COMPRIMIDO.

Numero: HU-04	**Nombre de la HU:** Desmontar el repositorio comprimido.
Modificación de la historia de usuario: Ninguna	
Programador: Frank E. Ramos Sendiña	**Iteración asignada:** 1

Prioridad en el Negocio: Alta	**Tiempo estimado:** 2 día
Riesgo en desarrollo: Alta	**Tiempo real:** 2 día
Descripción: El usuario selecciona la opción desmontar repositorio. Escenario 1: Exista un repositorio montado en una carpeta local. Se desmonta el repositorio. Escenario 2: No existe el repositorio montado en la carpeta local. Se le notifica al usuario lo sucedido.	
Observaciones:	
Prototipo de interfaz:	

TABLA 7: HU-04: DESMONTAR EL REPOSITORIO COMPRIMIDO.

Historias de usuarios	
Numero: HU-05	**Nombre de la HU:** Reinstalar paquetes desde un repositorio comprimido.
Modificación de la historia de usuario: Ninguna	
Programador: Frank E. Ramos Sendiña	**Iteración asignada:** 1
Prioridad en el Negocio: Baja	**Tiempo estimado:** 1 día
Riesgo en desarrollo: Baja	**Tiempo real:** 1 día
Descripción: El usuario introduce el nombre del paquete que desea reinstalar. A continuación la herramienta verificará la disponibilidad del paquete seleccionado por el usuario en el repositorio. Escenario 1: Está disponible el paquete seleccionado.	

Se procede con la reinstalación del paquete seleccionado por el usuario. Escenario 2: No está disponible el paquete seleccionado. Se notifica el suceso al usuario.
Observaciones:
Prototipo de interfaz:

TABLA 8: HU-05: REINSTALAR PAQUETES DESDE UN REPOSITORIO COMPRIMIDO.

Numero: HU-06	**Nombre de la HU:** Modificar la dirección del repositorio comprimido.
Modificación de la historia de usuario: Ninguna	
Programador: Frank E. Ramos Sendiña	**Iteración asignada:** 1
Prioridad en el Negocio: Baja	**Tiempo estimado:** 1 día
Riesgo en desarrollo: Baja	**Tiempo real:** 1 día
Descripción: El usuario modifica el archivo /etc/cmpr/cmpr.conf con el nuevo repositorio.	
Observaciones:	
Prototipo de interfaz:	

TABLA 9: HU-06: MODIFICAR LA DIRECCIÓN DEL REPOSITORIO COMPRIMIDO.

2.4 Selección de la arquitectura

En la investigación se seleccionó la arquitectura Sistemas Basados en Flujos de Datos. En la misma, todo el sistema de software es visto como una serie de transformaciones en piezas consecutivas o conjunto de datos de entrada, donde los datos y las

operaciones son independientes entre sí. Los datos se pueden lanzar en la topología gráfico con los ciclos, en una estructura lineal sin ciclos, o en una estructura de tipo árbol. El objetivo principal de este enfoque es lograr las cualidades de re utilización y modificación. Es adecuada para aplicaciones que involucran una serie bien definida de las transformaciones de datos independientes o cálculos en la entrada y la salida ordenada definida como compiladores.

Existen tres tipos de secuencias, Tuberías y filtros, Secuencial por lotes y Control de procesos. Para la implementación de la herramienta se selecciona el tipo Tuberías y filtros ya que este enfoque pone énfasis en la transformación gradual de los datos por componentes sucesivos. El flujo de datos es conducido por datos y todo el sistema se descompone en componentes de origen de datos, filtros, tuberías, y los sumideros de datos. Las conexiones entre los módulos son de flujo de que puede ser flujo de bytes, caracteres o cualquier otro tipo de este tipo. La característica principal de esta arquitectura es su ejecución concurrente [30].

Los llamados filtros no realizan forzosamente tareas de filtrado, como ser eliminación de campos o registros, sino que ejecutan formas variables de transformación, una de las cuales puede ser el filtrado.

Una tubería (pipeline) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salida. Las entradas de datos para las funcionalidades que se desean implementar pasan por filtros que determinan la salida o salidas que deben tener. Esto se evidencia en el ejemplo siguiente.

Un paquete trae implícito varias dependencias que resultan salidas para la instalación del paquete pero a su vez entradas que pasan por el filtro para determinar si deben ser instaladas antes de instalar el paquete.

Capítulo 3: Implementación y Prueba

En el presente capítulo se continúa con la fase de ejecución de la metodología escogida. Se planifica la implementación de las HU definidas donde se describen las funcionalidades de la herramienta, se define los estándares de codificación. Se definen la estrategia de prueba a utilizar, identificando tipos, métodos y niveles de pruebas para verificar el funcionamiento de la solución. Para ello se describen los diseños de casos de pruebas y finalmente se muestran los resultados de las pruebas realizadas y el impacto social de la herramienta para consumir paquetes de repositorios de código fuente y binarios comprimidos de la distribución cubana GNU/Linux Nova.

3.1 Planificación de la implementación

En la planificación de la implementación se tienen en cuenta las seis historias de usuarios generadas a partir de los requisitos obtenidos. Se tendrán en cuenta para la misma 2 iteraciones, la primera con los requisitos de prioridad alta y otra con los de prioridad media y baja. La característica que mayor peso tiene es la prioridad, ya que define la importancia para el desarrollo de la herramienta, se puede medir por el avance en cuanto a la implementación que defina el desarrollador. Los requisitos de prioridad alta son montar el repositorio comprimido en una carpeta local e instalar paquetes desde un repositorio comprimido.

3.2 Estándares de Codificación

El estándar de codificación empleado para la investigación es el Estándar de codificación para Bash. El mismo define parámetros a seguir para llevar a cabo la implementación del script que da lugar a la investigación. Estos parámetros son de gran importancia porque contribuyen a lograr uniformidad del código y volver en un futuro el código entendible para otros desarrolladores [31]. A continuación se explican e identifican los parámetros empleados para la implementación de la solución propuesta.

Formateo del código: ofrece características que debe cumplir el código en cuanto a la identación, el tamaño máximo de línea y las líneas en blanco.

Indentación

1. El código debe contener 4 espacios entre por cada por cada nivel de indentación como se muestra en la figura.

```
while [ $VAR != $MY_VAR ];
    do

echo "Presione q para Salir y seguido 'enter' "

echo "Presione d para desmontar las Carpetas(Si ya desmonto no presione mas ' d ')"

read VAR_NEW

if [ $VAR_NEW = $MY_VAR ]
    then
```

ILUSTRACIÓN 1: INDENTACIÓN

Tamaño máximo de línea

1. El tamaño máximo de cada línea de código no debe exceder a los 79 caracteres como se muestra en la figura.

```
elif [ $VAR = $MY_VAR_SE ]
    then
sudo umount $DP/MOUT_PKT
sudo umount $DP/REPO
sudo rmdir $DP/MOUT_PKT
sudo rmdir $DP/REPO
```

ILUSTRACIÓN 2: TAMAÑO MÁXIMO DE LÍNEA

Líneas en blanco

1. Las funciones no anidadas y las definiciones de clases se separaran con dos líneas en blanco. Mientras las definiciones de métodos dentro de una misma clase solo se separaran con una línea en blanco.

```
#Se monta el servidor en la carpeta deifida
sudo archivemount $nom $DP/MOUT_PKT
echo "Presione q para Salir y seguido 'enter'
echo "Presione d para desmontar las Carpetas"
```

ILUSTRACIÓN 3: LÍNEAS EN BLANCO

Uso de variables

1. Para variables importantes, nombres autodocumentados deben ser utilizados (**como inputfile**). Los nombres largos deben ser separados con "_" para mejorar la legibilidad como se muestra en la figura.

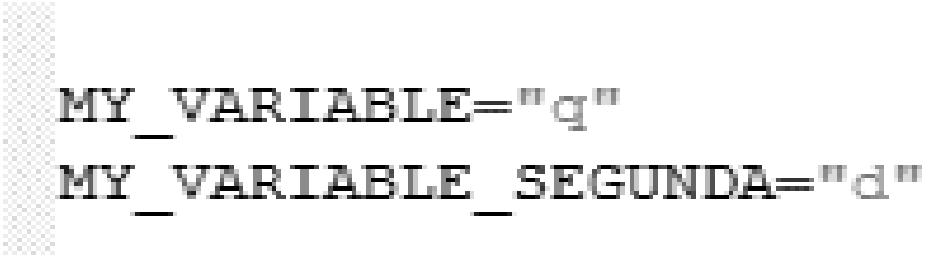


ILUSTRACIÓN 4: USO DE VARIABLES

3.2 Pruebas de software

Las pruebas de software no son más que un proceso de análisis de un sistema que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador, probando el comportamiento del mismo.

Para la investigación se definió la estrategia de pruebas que encierran diferentes tipos, niveles y métodos de pruebas que servirán de guía para el correcto funcionamiento de la solución. Los niveles de pruebas que se tuvieron en cuenta según el desarrollo de la solución fueron funcionales, de aceptación y de integración. Dentro de las pruebas unitarias se empleó el método caja blanca y la técnica utilizada fue camino básico en el caso de las pruebas de aceptación se utilizó el método caja negra y la técnica clases de equivalencia.

Niveles de prueba

Pruebas Funcionales:

Las pruebas funcionales son un proceso de control de calidad que consiste en asegurar el cumplimiento de un sistema o componente con requerimientos funcionales.

Estas pruebas pueden realizarse durante la fase de desarrollo, individualmente para secciones específicas desarrolladas por su equipo, al final del desarrollo de su proyecto, cuando las diferentes secciones de su proyecto están unidas. [32].

El objetivo principal de las pruebas funcionales es analizar el producto terminado y determinar si hace todo lo que debería hacer y si lo hace correctamente.

Pruebas de Aceptación:

Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. Una historia de usuario no es aceptada hasta que haya pasado su prueba de aceptación.

Las pruebas de aceptación verifican que el sistema que recibe el usuario funciona y lo hace de acuerdo con las especificaciones. Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcionad pactada previamente con el cliente [33].

Pruebas de integración:

Las pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos. Las pruebas de integración (algunas veces llamadas integración y testeo) es la fase del prueba de software en la cual módulos individuales de software son combinados y probados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden a las pruebas del sistema [34].

Métodos de pruebas

Técnica de diseño de las pruebas de Caja Negra

Las pruebas de caja Negra o pruebas funcionales son aquellas que recibe ciertas entradas y produce salidas o respuestas, sin tener en cuenta su funcionamiento interno, desde un punto de vista es saber lo que hace, sin dar importancia a como lo hace. Esta se centra en los requisitos funcionales del software. Es por ello que se le denominan

pruebas funcionales; ya que el probador se limita a suministrarle datos de entrada y estudiar lo que da de salida. Dichas pruebas se les aplicaran a las distintas historias de usuarios para verificar su correcto funcionamiento, se realizará una prueba a cada servicio con datos correctos e incorrectos. En el caso de datos correctos se mostrará el resultado esperado en caso contrario se notificará el error deseado.

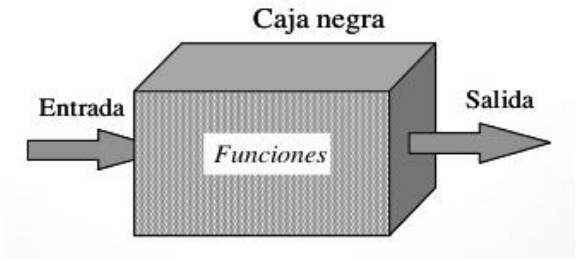


ILUSTRACIÓN 5: TÉCNICA DE DISEÑO DE LAS PRUEBAS DE CAJA NEGRA

Las pruebas de caja negra pretenden encontrar estos tipos de errores

1. Funciones incorrectas o ausentes.
2. Errores en la interfaz.
3. Errores en estructuras de datos o en accesos a bases de datos externas.
4. Errores de comportamiento o desempeño.
5. Errores de inicialización y de terminación [35].

Clases de Equivalencias

Una partición equivalente es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia [36].

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.

Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Las clases de

equivalencia se pueden definir de acuerdo con las siguientes directrices: Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

- Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Aplicando estas directrices se ejecutan casos de pruebas para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez**.**

A continuación, se presentan los casos de prueba correspondientes a la historia de usuario.

3.2.4 Diseño de Casos de Prueba

Caso de Prueba HU-01 "Instalar de paquetes desde un repositorio comprimido"				
Escenario	**Descripción**	**Variable (Nombre del paquete)**	**Respuesta del sistema**	**Flujo Central**
EC 1.1	Se instala el paquete correctamente.	V Cualquier valor alfanumérico igual al nombre de los paquetes existente en el repositorio	Se muestra la opción finalizada para la instalación del paquete	1. El usuario entra el paquete que desea instalar. 2. La herramienta finaliza el proceso instalando el paquete seleccionado por el usuario.
EC 1.2	No se instala el paquete.	I Cualquier valor alfanumérico distinto de los nombre	Se muestra una notificación con lo sucedido	1. El usuario entra el paquete que desea instalar. 2. La herramienta finaliza el proceso mostrando una notificación de error.

		existentes en el repositorio		

TABLA 10: CASO DE PRUEBA HU-01

Caso de Prueba HU-02 “Descargar el código fuente de un paquete del repositorio comprimido”				
Escenario	**Descripción**	**Variable (Nombre del paquete)**	**Respuesta del sistema**	**Flujo Central**
EC 2.1	Se descarga el código fuente del paquete.	V Cualquier valor alfanumérico igual al nombre de los paquetes existente en el repositorio	Se muestra la opción finalizada para la descarga del código fuente del paquete	1. El usuario selecciona el paquete que desea descargar su código fuente. 2. La herramienta finaliza el proceso de descarga de código fuente del paquete seleccionado por el usuario.
EC 2.2	No se descarga el código fuente del paquete.	I Cualquier valor alfanumérico distinto de los nombre existentes en el repositorio	Se muestra una notificación con lo sucedido	1. El usuario entra el paquete que desea descargar su código fuente. 2. La herramienta finaliza el proceso mostrando una notificación de error.

TABLA 11: CASO DE PRUEBA HU-02

Caso de Prueba HU-03 “Reinstalar paquetes desde un repositorio comprimido”				
Escenario	**Descripción**	**Variable (Nombre del paquete)**	**Respuesta del sistema**	**Flujo Central**

EC 3.1	Se reinstala el paquete seleccionado por el usuario.	V Cualquier valor alfanumérico igual al nombre de los paquetes existente en el repositorio	Se muestra la opción finalizada para la descarga del código fuente del paquete	1. El usuario selecciona el paquete que desea reinstalar. 2. La herramienta finaliza el proceso de reinstalación del paquete seleccionado por el usuario.
EC 3.2	No se reinstala el paquete ya que no coincide el nombre entrado por el usuario.	I Cualquier valor alfanumérico distinto de los nombre existentes en el repositorio	Se muestra una notificación con lo sucedido	1. El usuario entra el paquete que desea reinstalar. 2. La herramienta finaliza el proceso mostrando una notificación de error.
EC 3.3	No se reinstala el paquete porque no se encuentra instalado	I Cualquier valor alfanumérico distinto de los nombre existentes en el repositorio	Se muestra una notificación con lo sucedido	1. El usuario entra el paquete que desea reinstalar. 2. La herramienta finaliza el proceso mostrando una notificación de error

TABLA 12: CASO DE PRUEBA HU-03

Caso de Prueba HU-04 "Montar el repositorio comprimido en una carpeta local"				
Escenario	**Descripción**	**Variable (Nombre del repositorio)**	**Respuesta del sistema**	**Flujo Central**
EC 4.1	Se monta el repositorio correctamente.	V Cualquier valor alfanumérico correspondientes a los repositorios existentes en el servidor NFS	Se muestra la operación finalizada.	1. El usuario se conecta al servidor NFS. 2. El usuario introduce el nombre del repositorio que desea montar.

				3. La herramienta finaliza con el montaje del repositorio.
EC 4.2	No se monta el repositorio.	I Cualquier valor alfanumérico correspondientes a los repositorios existentes en el servidor NFS	Se muestra una notificación con lo sucedido	1. El usuario se conecta al servidor NFS. 2. El usuario introduce el nombre del repositorio que desea montar. 3. La herramienta finaliza con una notificación de error.

TABLA 13: CASO DE PRUEBA HU-04

Caso de Prueba HU-05 "Desmontar el repositorio comprimido"				
Escenario	**Descripción**	**Variable (N/A)**	**Respuesta del sistema**	**Flujo Central**
EC 5.1	Se desmonta el repositorio correctamente.	N/A	Se muestra la operación de desmontaje finalizada.	1. El usuario solicita la opción de desmontar repositorio comprimido. 2. La herramienta verifica si hay repositorios montados en una carpeta local. 3. La herramienta finaliza con el desmontaje del repositorio.
EC 5.2	No se desmonta el repositorio.	N/A	Se muestra una notificación con lo sucedido	1. El usuario solicita la opción de desmontar repositorio comprimido.

				2. La herramienta verifica si hay repositorios montados en una carpeta local. 3. La herramienta finaliza con una notificación de error.

TABLA 14: CASO DE PRUEBA HU-05

Caso de Prueba HU-06 "Modificar la dirección del repositorio comprimido"				
Escenario	**Descripción**	**Variable (codename y ramas)**	**Respuesta del sistema**	**Flujo Central**
EC 6.1	Se modifica la dirección del repositorio.	V Cualquier valor alfanumérico que concuerde con el codename y las ramas del repositorio que se desea modificar la dirección.	Se muestra la operación de modificación de la dirección del repositorio finalizada.	1. El usuario solicita la opción de modificar la dirección del repositorio comprimido. 2. La herramienta verifica si el codename y las ramas concuerdan con las del repositorio comprimido. 3. La herramienta finaliza con la modificación de la dirección del repositorio.
EC 6.2	No se modifica la dirección del repositorio.	I Cualquier valor alfanumérico que no concuerde con el codename y las ramas del repositorio que se desea modificar la dirección.	Se muestra una notificación con lo sucedido	1. El usuario solicita la opción de modificar la dirección del repositorio comprimido. 2. La herramienta verifica si el codename y las ramas concuerdan con las del repositorio comprimido. 3. La herramienta finaliza con una notificación de error.

TABLA 15: CASO DE PRUEBA HU-06

3.3 Resultados obtenidos de los casos de prueba

Se utilizó el método de caja negra para realizar las pruebas sobre la interfaz de la herramienta. Donde se encontraron 10 no conformidades, en 4 iteraciones. De estas no conformidades 2 fueron de errores ortográficos y 8 de validación incorrecta. En la siguiente gráfica se muestra las iteraciones realizadas y las no conformidades detectadas, las resueltas y las pendientes.

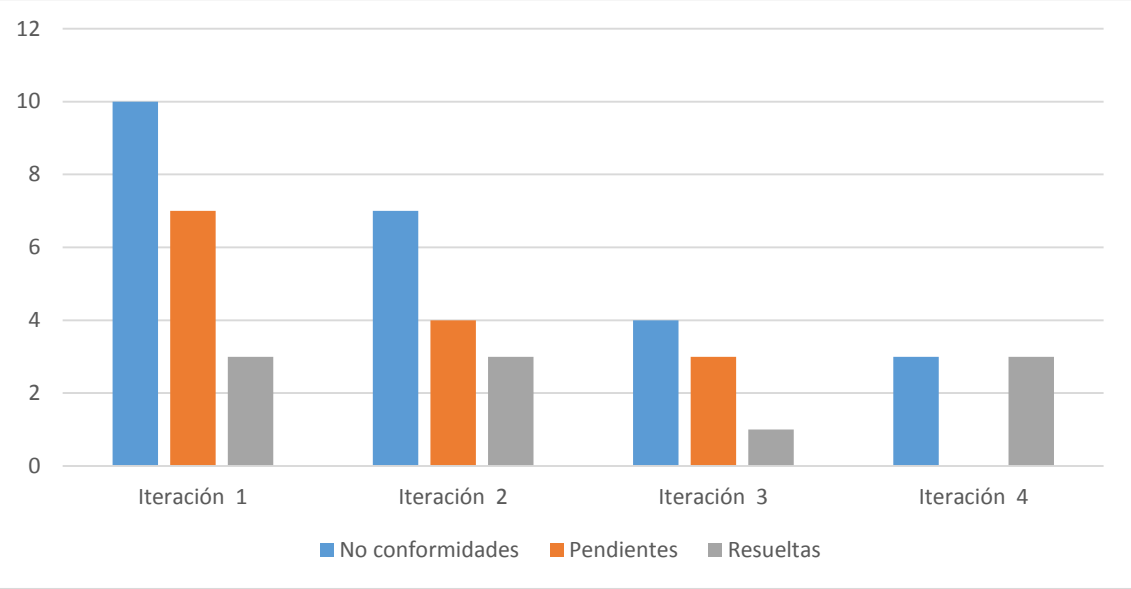


ILUSTRACIÓN 6: RESULTADOS OBTENIDOS DE LOS CASOS DE PRUEBA

Pruebas de integración

Para garantizar la correcta integración de la herramienta con la distribución cubana de GNU/Linux Nova se procede a la construcción de un paquete DEBIAN, que sea instalable en el mismo a través de centro de software. Esto permitió copiar los componentes de la herramienta en los directorios necesarios para su correcto funcionamiento y permitió verificar que la herramienta es instalable y se integra correctamente con dicha distribución.

Conclusiones

Con el desarrollo de una herramienta para consumir paquetes de repositorios comprimidos de la distribución cubana de GNU/Linux Nova se da cumplimiento al objetivo general planteado. Para llegar a este resultado se concluye lo siguiente

• Se realizó el estudio de las principales herramientas de obtención de información, formatos de compresión y protocolos para publicarla, quedaron seleccionados .squashfs como formato de compresión y NFS como protocolo para publicar la información.

• Se realizó el análisis, diseño e implementación de la herramienta para consumir paquetes de un repositorio de la Distribución Cubana de GNU/Linux Nova, obteniendo así una herramienta que cumple con las necesidades.

• Se diseñaron y se realizaron las pruebas de software, permitiendo la correcta comprobación del funcionamiento de la solución, dando la posibilidad de realizar el proceso de entrega del software a consideración del cliente.

Recomendaciones

Tras finalizada la investigación se recomienda:

- ✓ Incluir la herramienta en las versiones de la distribución cubana de GNU/Linux Nova para dicha utilización en el futuro.

- ✓ Desarrollar una interfaz visual para la herramienta que facilite un sencillo y mejor uso de la misma

Referencias bibliográficas

1 Alegsa, L. (5 de diciembre de 2010). Diccionario de Informática y Tecnologia. Obtenido de Definicion de Paquete de software: http://alegsa.com.ar/Dic/paquete_de_software.php

2 Aiser. (13 de marzo de 2014). Ovtoaster Tecnología Libre para mentes curiosas. Obtenido de http://ovtoaster.com/repositorio-linux

3 Wordpress. (4 de novienbre de 2012). ¿Que es un repositorio? . Obtenido de http://softwarelibreecuador.wordpress.com/2012/11

4 Alegsa, L. (26 de agosto de 2010). Diccionario de Informática y Tecnologia. Obtenido de Definicion de codígo binario: http://alegsa.com.ar/Dic/codigo_binario.php

5 Definista. (1 de junio de 2016). Obtenido de Concepto Relacion: http://conceptodefinicion.de/codigo-fuente

6 Alegsa, L. (5 de diciembre de 2010). Diccionario de Informática y Tecnologia. Obtenido de Definicion de Sistema de Gestion de Paquetes: http://alegsa.com.ar/Dic/sistema_de_gestion_de_paquetes.php

7 Todo y Mas. (28 de septiembre de 2009). Obtenido de Manual Basico sobre uso de APT en linux: http://todoymas.wordpress.com/2009/09/28/manual-basico-sobre-el-uso-de-apt-en-linux

8 Ognio, A. (17 de abril de 2010). Gestor de paquetes YUM. Obtenido de Gestor de paquetes YUM: https://es.slideshare.net/gnfran/gestor-de-paquetes-yum

9 Herredia, M. L. (9 de mayo de 2014). Extensiones de archivos en GNU/linux mas comunes. Obtenido de http://www.laguialinux.es/extensiones-de-archivos-mas-comunes

10 Comprimidor de Archivos. (17 de junio de 2010). Obtenido de http://compiar.blogspot/2016/06/comprimidor-de-archivos_17.html

11 TGZ, GZIpped Tar File. (2 de marzo de 2012). Obtenido de http:online-convert.com/es/formato-de-archivo/tgz

12 alcanceLibre. (20). Obtenido de Comprension y descomprension de archivos: http://alcancelibre.org/staticpages/index.php/comprension-descompresion-archivos

13 Meyer, B. (27 de noviembre de 2012). Linuxito. Obtenido de Como descomprimir archivos 7zip: http://linuxito.com.gnu-linux/nivel-basico/125-como-descomprimir-archivos-7zip

14 Campos, J. I. (2013). Squshfs.

15 Que es un servicio FTP- File Tranfer Protocol. (1 de mayo de 2011). Obtenido de http://intentaya/que-es-el-servicio-ftp-file-tranfer-protocol

16 Sandoval, J. (9 de enero de 2016). Ventajas y desventajas del Servidor FTP. Obtenido de http://ventajasftp.blogspot.com

17 NfS : sistemas de archivos de red. (25 de agosto de 2015). Obtenido de http://recursostic.educacion.es/observatorio/web/gl/software/software-general/733-nfs-sistema-de-archivos-de-red

18 JCL, G. (22 de julio de 2011). Servidores NFS. Obtenido de Ventajas del Servidor NFS : http://servidornfs.blogspot.com/2011/07/ventajas-del-servidor-nfs.html

19 Fonseca, G. (31 de marzo de 2011). NFS – NETWORK FILE SYSTEM. Obtenido de https://guillermofonseca.wordpress.com/2011/03/31/nfs-network-file-system/

20 Definista. (18 de abril de 2014). Definición de Http. Obtenido de http://conceptodefinicion.de/http/

21 Copyright. (12 de octubre de 2010). ¿Qué es https y para que sirve? Obtenido de http://www.ordenadores-y-portatiles.com/https.html

22 Darrington, J. (1 de diciembre de 2009). Transferencia de archivos HTTP versus FTP . Obtenido de http://www.ehowenespanol.com/transferencia-archivos-http-versus-ftp-sobre_117577/

23 Sommerville, I. (2005). Ingeneria de Software. Departamento Ciencia de La computacion e Inteligencia Artificial Universidad de Alicante.

24 Copyright. (12 de abril de 2010). software.com.ar. Obtenido de Visual Paradigm para UML: http://www.software.com.ar/p/visual-paradigm-para-uml#product-description

25 Gil, J. C. (2 de febrero de 2010). Geany, algo más que un editor de código para Gnome. Obtenido de http://www.linuxhispano.net/2010/02/02/geany-algo-mas-que-un-editor-de-codigo-para-gnome/

26 Gedit. (4 de novienbre de 2007). Obtenido de http://guia-ubuntu.com/index.php/gedit

27 Cetred, C. C. (27 de enero de 2011). Fartarneo GNU/Linux . Obtenido de http://fraterneo.blogspot.com/2011/01/que-es-bash.html

28 Sánchez, T. R. (2015). Metodología de desarrollo para la Actividad productiva de la UCI. La Habana, Cuba.

29 Sommerville, I. (2005). ingeniería de Software. Departamento Ciencia de La computación e Inteligencia Artificial Universidad de Alicante.

30 CESOL_Nova 6.0_Estandares de codificación para Bash.doc

31 Pressman, R (2002). Ingeniería del Software, un enfoque práctico, Oc-Graw Hill

32 ¿Qué son pruebas funcionales? (12 de agosto de 2011). Obtenido de https://crowdsourcedtesting.com/es/pruebas-funcionales

33 Freepik. (8 de agosto de 2016). PMOinformatica.com. Obtenido de La oficina de proyectos de informática: http://www.pmoinformatica.com/2016/08/pruebas-aceptacion-software-istqb.html

34 Cillero, M. (12 de mayo de 2011). Pruebas de Integración. Obtenido de https://manuel.cillero.es/doc/metrica-3/tecnicas/pruebas/integracion/

35 Carmona, C. R. (2 de julio de 2013). Prueba caja blanca. Obtenido de https://es.slideshare.net/cristalramirezcarmona/prueba-caja-blanca

36 Pruebas de software "Caja Blanca y Caja Negra". (30 de noviembre de 2008). Obtenido de http://angelmolsoftware.blogspot.com/2008/11/pruebas-del-software-caja-blanca-y-caja.html