Universidad de las Ciencias Informáticas

Facultad 1

Centro de Software Libre

“Herramienta para la sincronización de repositorios en GNU/Linux Nova”

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

Autor: Miguel Luis Hernández Brocat

Tutores:M.Sc. Arianna Rodríguez Jiménez

Ing. Enmanuel Cristian de la Cruz Mora

La Habana, 2020

Declaración de autoría

Declaro por este medio que yo **Miguel Luis Hernández Brocat**, con carné de identidad **96122009027** soy el autor principal del trabajo titulado "**Herramienta para la sincronización de repositorios en GNU/Linux Nova**" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

exclusivo.

Para que así conste firmamos la presente a los _____ días del mes de _____ de _____

Autor: _____

Tutor: _____

Tutor: _____

Dedicatoria

Se lo dedico enespeciala mis padres.

Agradecimientos

Ami padre Juan Miguel Hernández Pérez quien es todo en mi vida, y donde quiera que se encuentre sabrá queeste resultado es para él y gracias a él, atodo su amor y educación.

Ami madre Lourdes Brocat González quien siempre ha sido mi guía y mi apoyo en seguir adelante y esforzarme cada día.

Atoda mi familia, a mi novia, amigos, atodaslaspersonas que hanestado conmigo y me handado su ayuda.

A todos les doy las gracias, sin ellos esto no hubiera sido posible.

Resumen

La sincronización de repositorios cumple un factor importante dentro de la comunidad de usuarios que utilizan los sistemas GNU/Linux. En la Distribución Cubana de GNU/Linux Nova se utiliza la herramienta Rsync para ejecutar esta tarea, sin embargo no cumple con todos las necesidades del usuario puesto las sincronizaciones son lentas, ya sea ocasionado porque debe transferir un elevado volumen de archivos de pequeño tamaño en lugar de un único archivo de gran tamaño lo que incrementa el tiempo total de la copia o debido al poco ancho de banda con el que cuentan las instituciones del país, provocando inconformidad por los clientes que la usan. Por tal motivo el objetivo de la presente investigación es desarrollar una nueva herramienta que permita realizar el proceso de sincronización en menor tiempo del que actualmente demora esta actividad. Para ello se realizó el estudio de las herramientas más utilizadas para la sincronización de repositorios, y se definieron la técnicas y herramientas necesarias para el desarrollo de la solución. Posteriormente se realizaron un conjunto de pruebas para validar que la herramienta desarrollada cumplía con el objetivo trazado y daba solución a la problemática planteada al inicio de la investigación.

Palabras clave: sincronización, repositorios, GNU/Linux Nova

Índice

Índice de Figuras

Índice de Tablas

Introducción

El avance de Tecnologías de la Información y las Comunicaciones (TIC) ha devenido en el desarrollo de lo que se conoce actualmente como sociedad de la información. Representan, además, un factor importante dentro del campo económico y sociocultural de los países convirtiéndose en una de las principales vías de desarrollo de los mismos. En la última década, numerosas tecnologías han aparecido para favorecer el desarrollo de la llamada sociedad de la información. El software libre y de código abierto es, sin lugar a dudas, una de las tecnologías que más influencia está demostrando tener a la hora de favorecer dicho desarrollo.

En 1984, Richard Stallman, desarrollador de software y defensor del software libre, creó el proyecto GNU con la idea de crear un sistema operativo siguiendo los estándares de UNIX. Lo más importante es que un sistema operativo posee un sistema central, más conocido como kernel (núcleo), el cual es más complejo de elaborar. Durante los primeros años de desarrollo se probaron varios núcleos de sistemas, pero ninguno logró satisfacer las necesidades del sistema GNU que se estaba desarrollando (1).

En 1991 sale a la luz un nuevo kernel que marca un precedente en el desarrollo del software libre y de los sistemas operativos. Este kernel se comenzó a desarrollar como un hobby de Linus Torvalds y fue denominado Linux. Este es un sistema operativo totalmente libre y funcional, desarrollado por una comunidad libre. Surge además en busca de una alternativa al software propietario que crea abismos de colaboración entre los desarrolladores de software (1).

Las distribuciones de GNU/Linux, han alcanzado un alto grado de madurez y muchos seguidores al código abierto se han empeñado en lograr sistemas operativos que cumplan con las necesidades y demandas de los usuarios. Con estos sistemas se puede trabajar en diferentes tipos de proyectos, ya que están desarrollados como una robusta y eficaz herramienta de trabajo. Esto ha logrado posicionar a GNU/Linux como uno de los sistemas operativos más elegidos a nivel profesional, por ser realmente estable, confiable y seguro, además de ser libre.

GNU/Linux comenzó a ser optimizado con distintas opciones en las diferentes universidades técnicas y compañías, y a distribuir estas versiones. Esta situación ha devenido en lo que se conoce como distribuciones de GNU/Linux que no son más que versiones del sistema original. Estas distribuciones (abreviada a distros) se diferencian,

entre otras muchas características, por las herramientas de configuración y los sistemas de administración de paquetes de software para la instalación que incluyen (2).

En Cuba se ha iniciado hace unos años un proceso de migración dentro del cual se contempla el desarrollo de una distribución GNU/Linux propia. En la Universidad de Ciencias Informáticas (UCI) en febrero del 2003 surgió un equipo de desarrollo, actualmente devenido en Centro de Soluciones Libres (CESOL), para la creación de una distribución GNU/Linux denominada Nova. Esta tiene como objetivo la migración al código abierto y el software libre en Cuba para afianzar la soberanía tecnológica.

La Distribución Cubana de GNU/Linux Nova al igual que todos los sistemas basados en GNU/Linux usan repositorios de software para instalar los programas necesarios para el usuario. Los repositorios de software son grandes bancos de datos que alojan las aplicaciones que el sistema necesita, entre ellos, paquetes de software y actualizaciones que son instaladas mediante un manejador de paquetes.

A diferencia de otras distros Nova no posee una herramienta para sincronizar sus repositorios, para ello se utiliza Rsync. A pesar de no ser una herramienta diseñada específicamente para estos propósitos permite sincronizar archivos y directorios entre dos máquinas de una red o entre dos ubicaciones en una misma máquina.

Sin embargo, la sincronización de repositorios con Rsync tarda mucho tiempo. Una de las causas que ocasionan esta demora es que debe transferir un elevado volumen de archivos de pequeño tamaño en lugar de un único archivo de gran tamaño. Esto incrementa el tiempo total de la copia puesto que a pesar de que el tiempo de latencia-transferencia del disco es el mismo, el tiempo de búsqueda para la atención de las peticiones al disco duro se incrementa porque el cabezal debe desplazarse continuamente hacia la ubicación de los archivos a copiar.

Otra de las causas es el ancho de banda, si bien no es posible transformar este aspecto debido a las condiciones tecnológicas de la universidad, es un factor determinante a tener en cuenta en las posibles soluciones que se puedan brindar a esta problemática.

El algoritmo base de Rsync forma parte del núcleo del software, sin embargo, la aplicación cuenta con otras funcionalidades que intervienen en el proceso de transferencia entre las que se encuentran la compresión/descompresión de datos y el soporte para protocolos de cifrado como el SSH. A pesar de que la utilización de SSH es buena para la seguridad de la transferencia, enlentece el proceso puesto que es

necesario emplear tiempo para el cifrado y luego para descifrar los datos. Por otro lado, la sincronización se realiza a través de un hilo de copia. Esto dificulta que se ejecuten transferencias de manera simultánea, desaprovechando la posibilidad de realizar diversas tareas de manera concurrente.

Teniendo en cuenta el tiempo que toma el proceso de sincronización de repositorios de la Distribución Cubana de GNU/Linux Nova y luego de analizar los principales aspectos que inciden en esta demora se define como **Problema de investigación:** ¿Cómo disminuir el tiempo de ejecución del actual mecanismo de sincronización de la Distribución Cubana de GNU/Linux Nova?

Se define como **objeto de estudio** el proceso de sincronización de los repositorios de software en los sistemas GNU/Linux, tomando como**campo de acción**el proceso de sincronización de los repositorios de la Distribución Cubana de GNU/Linux Nova. Para dar solución a la problemática planteada se formula como **objetivo general** desarrollar una herramienta informática para la sincronización de repositorios que permita disminuir el tiempo de ejecución de este proceso en la Distribución Cubana de GNU/Linux Nova.

El objetivo general de la investigación se desglosa en los siguientes **objetivos específicos**:

1. Caracterizar el estado actual de los sistemas existentes para la sincronización de repositorios de las distintas distribuciones de GNU/Linux, así como las tecnologías y herramientas utilizadas en su desarrollo.
2. Desarrollar una herramienta para la sincronización de los repositorios de la Distribución Cubana de GNU/Linux Nova
3. Validar la solución propuesta

Como **hipótesis de la investigación** se plantea que el desarrollo de un software específico para la sincronización de los repositorios en la Distribución Cubana de GNU/Linux Nova permitirá disminuir el tiempo que actualmente se dedica a esta tarea.

En la hipótesis anteriormente planteada se define como variable independiente el software, siendo esta de carácter booleano la cual afectará a la variable dependiente tiempo, la cual se expresará en horas.

Para cumplir con los objetivos específicos se proponen las siguientes **tareas de investigación**:

- Análisis de los principales conceptos relacionados con la investigación.
- Investigación sobre los sistemas homólogos y las herramientas a utilizar para el desarrollo de la solución.
- Definición de la metodología de desarrollo, las tecnologías y la arquitectura para la implementación del sistema
- Definición de los requisitos funcionales y no funcionales.
- Implementación de las funcionalidades de la herramienta.
- Diseño y ejecución de las pruebas a la herramienta desarrollada.

Como **resultado** se obtendrá una herramienta para la sincronización de los repositorios de la distribución cubana GNU/Linux Nova que disminuya el tiempo de ejecución con respecto a la utilizada actualmente.

E el proceso de desarrollo de la investigación se emplearon los siguientes **métodos**:

La **tormenta de idea** fue el método empírico utilizado en el proceso de investigación para ladeterminación de las restricciones de software e implementación que se debe tener en cuenta en el proceso de desarrollo de la herramienta.

Métodos científicos:

El método **inductivo-deductivo** permite llegar a proposiciones generales a partir de hechos aislados que confirman la teoría o a partir de estas teorías arribar a conclusiones sobre casos particulares que se verifican en la práctica. Este método fue usado para, a partir del análisis de las herramientas existentes que realizan la sincronización de repositorios, seleccionar las características que va a poseer la herramienta a desarrollar de forma tal que realice correctamente la sincronización de los repositorios.

El método **histórico-lógico** plantea que se debe estudiar la trayectoria real de los fenómenos yacontecimientos en el transcurso de una etapa o período analizando las leyes generales delfuncionamiento y desarrollo del fenómeno, estudiando su esencia. Fue empleado con el objetivo de verificar cómo evolucionan teóricamente las herramientas que realizan la sincronización de los repositorios y de este modo poder realizar una selección de las técnicas, herramientas y los algoritmos que se van a utilizar para desarrollar.

El método **analítico-sintético** consiste en la extracción de las partes de un todo, con el objetivo de estudiarlas y examinarlas por separado para luego sintetizarla y tomar los elementos más importantes.Este método permitió el estudio de diferentes fuentes

bibliográficas para extraer los elementos más importantes que se relacionan con las herramientas para la sincronización de los repositorios de distribuciones GNU/Linux.

El método **revisión bibliográfica** permite identificar las fuentes arbitradas, identificar y comprender las ideas principales y sintetizar el argumento central de cada sección a través de la supresión y generalización de ideas principales. Dicho método fue utilizado para determinar las fuentes y referencias bibliografías óptimas y actualizadas para la elaboración de la investigación.

La presente investigación se encuentra estructurada en tres capítulos, el **capítulo 1**, Fundamentación Teórica: describe los principales conceptos asociados al problema de investigación planteado. Se realiza un análisis de las principales características y deficiencias de cada herramienta encontrada y finalmente, se seleccionan las herramientas que van a ser utilizadas para el desarrollo de la solución y se selecciona la metodología de desarrollo que va a guiar todo el proceso de construcción de la herramienta para la sincronización de los repositorios en la Distribución Cubana GNU/Linux Nova.

Capítulo 2, Análisis y Diseño de la Solución: se realizan las fases de inicio y ejecución de lametodología descrita en el capítulo 1, esta última fase continúa en el capítulo 3. Se realiza la extracción de requisitos funcionales y no funcionales de la herramienta a desarrollar, se modela la estructura y funcionamiento de la solución a través de los diagramas de componentes y paquetes. Además, se seleccionan los patrones de diseño que serán usados para resolver la problemática.

Capítulo 3, Implementación y Pruebas de la Solución: se continúa con la fase de ejecución de la metodología escogida en el capítulo 1 y se realiza la tercera fase de esta metodología para el desarrollo de la herramienta, se hace el análisis de las funciones identificadas como vulnerables. Se explican los distintos tipos de pruebas de software que se van a utilizar para comprobar el correcto funcionamiento de la solución, así como los resultados que arrojaron la utilización de las mismas y se muestra el impacto social que va a tener la investigación.

Capítulo 1. Fundamentación Teórica

Introducción

Con el objetivo de facilitar la comprensión del alcance de la investigación, en el presente capítulo se exponen conceptos fundamentales asociados al dominio del problema. Se realiza un análisis de las principales características de las herramientas de sincronización de repositorios para definir los elementos que se van a tener en cuenta para la propuesta de solución. Se seleccionan las herramientas y tecnologías que serán utilizadas para desarrollar la solución y se define la metodología a seguir durante el proceso de construcción de la herramienta para sincronizar repositorios en la distribución cubana GNU/Linux Nova.

1.1 GNU/Linux Nova

Nova es un sistema operativo de propósito general basado en Ubuntu que utiliza el núcleo de Linux e incluye un amplio paquete de aplicaciones informáticas dedicadas a satisfacer los requerimientos derivados del proceso de migración a plataformas y estándares abiertos convirtiéndose en uno de los principales pilares en este proceso de migración a software libre en el país.

Es una distribución de GNU/Linux desarrollada en la UCI con razón de apoyar la migración del país a tecnologías de Software Libre y Código Abierto (5). Nova se basa principalmente en las cuatro libertades del software libre y en las 4 S del Msc. Allan Pierra Fuentes: Seguridad, Soberanía Tecnológica, Socioadaptabilidad y Sostenibilidad(6); las cuales dictan las condiciones políticas, sociales y económicas que regirán el proceso de migración a software libre en Cuba y las cuales se definen de la siguiente manera(6):

- Seguridad: el modelo de desarrollo colaborativo que propone el movimiento de software libre, el acceso al código fuente y el exhaustivo proceso de revisión y auditoría de código garantizará un sistema seguro de ataques y sin puertas traseras.
- Soberanía Tecnológica: es la capacidad del país para desarrollarse en dicho campo en forma autónoma. No supone autarquía (independencia absoluta) sino capacidad de decidir sobre su uso y desarrollo.
- Socio-adaptabilidad: las bases tecnológicas para la informatización de Cuba, deben ser hechas por cubanos y para los cubanos, logrando inigualable adaptabilidad a las condiciones de nuestro País.

- Sostenibilidad: la constante asimilación e investigación de las nuevas tecnologías, la planificación, los modelos novedosos de comercialización y el uso racional de los recursos humanos, materiales y naturales, garantizarán soluciones, vigencia y sostenibilidad a largo plazo.

El modelo de desarrollo colaborativo y el acceso al código utilizado permite un proceso flexible y versátil, en constante innovación y consonancia con las nuevas tendencias tecnológicas internacionales. El amplio espectro y diversidad de las estaciones de trabajo que conviven en el entorno tecnológico cubano crea la necesidad de un sistema capaz de adaptarse a este escenario. Por ello, la distribución cubana de GNU/Linux se divide en 3 productos fundamentales: Nova Ligero, para estaciones de trabajo con bajas prestaciones de hardware (menos de 512Mb de memoria RAM); Nova Escritorio, para estaciones de trabajo con buenas prestaciones de hardware (de 512Mb en adelante) y Nova Servidores para ordenadores que tienen como función el control de redes, centro de datos, entornos empresariales y de desarrollo, entre otros(7).

La distribución brinda una gama de productos y servicios orientados a usuarios nacionales inexpertos en el área de las tecnologías de software libre y que estén experimentado un proceso de migración a las mismas. Al estar equipada con múltiples servicios y con un repositorio bien estructurado es capaz de satisfacer las necesidades de los mismos, lo que la convierte en una opción viable para estos(7).

El sistema se puede dividir en tres paquetes fundamentales (7):

- Gestor de sesiones: permite comenzar una sesión desde la misma u otra computadora. Se presenta al usuario con una pantalla de autenticación (login) que solicita el nombre de usuario y su contraseña. EL Gestor de sesiones a su vez está compuesto por tres paquetes: configuraciones, Entorno de escritorio y las aplicaciones del sistema.
- Sistema operativo base: estructura mínima de interacción con los componentes de funcionamiento del núcleo del sistema y con los componentes de la interfaz de usuario final. Se dedicará al desarrollo de las capas tecnológicas básicas de la distribución, la definición de la arquitectura, la persistencia o gestión de los paquetes de software, así como el desarrollo de tecnologías y servicios para la construcción de los productos asociados a las Líneas de Producción y Sistemas a la medida. Contiene el instalador, el kernel y las bibliotecas del sistema.

- Repositorio: Sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos. Pueden contener los archivos en su servidor o referenciar desde su web al alojamiento originario. Los repositorios pueden ser de acceso público, o pueden estar protegidos y necesitar de una autenticación previa. Posee dos ramas: principal donde se encuentran las aplicaciones fundamentales del sistema y extendido que aloja un conjunto de paquetes, que no dejan de ser importantes pero que sin ellos el sistema puede funcionar, aunque sea con los requerimientos mínimos.

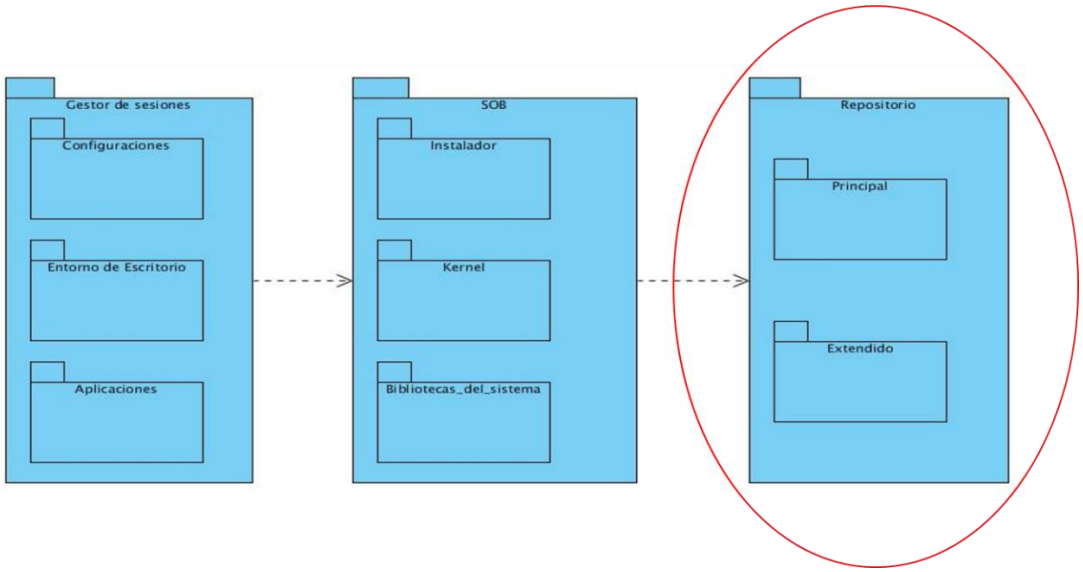

Figura 1 Diagrama de paquetes de Nova.

Adaptado de (7)

La solución de la presente investigación se encuentra enmarcada en los repositorios, cuyas características se abordan a continuación.

1.1 Repositorios en GNU/Linux

En los sistemas operativos GNU/Linux, un repositorio es una colección de paquetesbinarios o de código fuente organizados en una estructura especial que generalmente contiene archivos binarios pre-compilados los cuales pueden ser descargados e instalados por los usuarios de la distribución correspondiente.

De acuerdo a un estudio realizado a diferentes referencias, se define para la investigación que los repositorios para distribuciones de GNU/Linux son: “un conjunto de paquetes de software disponibles para una determinada distribución almacenados de forma centralizada, estos se encuentran alojados de forma estructurada y organizados por índices de los paquetes que estén disponibles, asegurando su autenticidad y que no se

encuentren dañados, de manera tal que los usuarios puedan administrarlos usando gestores de paquetes que los transfieren por vía ftp o http". Estos se pueden clasificar en:

- Repositorios institucionales: son los creados por las propias organizaciones para depositar, usar y preservar la producción científica y académica que generan. Supone un compromiso de la institución con el acceso abierto al considerar el conocimiento generado por la institución como un bien que debe estar disponible para toda la sociedad (8).
- Repositorios temáticos: son los creados por un grupo de investigadores, una institución, etc. que reúnen documentos relacionados con un área temática específica (9).
- Repositorios de datos: repositorios que almacenan, conservan y comparten los datos de las investigaciones (4).

Para hacer uso de un repositorio, generalmente se utilizan programas llamados gestores de paquetes y facilitan la gestión de dependencias y las acciones comunes como instalar, actualizar, eliminar y buscar paquetes(4). A diferencia de las computadoras personales, los repositorios suelen contar con sistemas de respaldo (backup) y mantenimiento correctivo y preventivo, lo que hace que la información se pueda recuperar en caso de pérdida de la misma. Otro detalle digno de hacer notar reside en que los paquetes incluidos en los repositorios pueden ser de código fuente o pre-compilados.

Los repositorios basados en Debian sueles ser organizados en dos carpetas principales (10):

- Directorio **dists**: contiene los índices y metadatos de las diferentes versiones de la distribución y sus componentes.
- Directorio **pool**: contiene todos los paquetes del repositorio. Cada paquete se encuentra en una carpeta con la siguiente topología: <a>/<b>/<c>, donde a es el componente a la que pertenece el paquete, b es la letra inicial del nombre del paquete (o las cuatro primeras en caso de que empiece con lib) y c el nombre del paquete.

Un paquete de software es un archivo comprimido y con una estructura establecida que permite ser tratado por herramientas de gestión de software para realizar operaciones como instalar, compilar, eliminar, purgar los archivos de configuración del sistema, actualizar, entre otras, de forma cómoda, segura, estable y centralizada. Se puede compilar si se basa en código fuente o es instalable si lo hace en binarios compilados ya

para una determinada arquitectura o plataforma. Contiene documentación, scripts de pre y post instalación, archivos de configuración iniciales, archivos de recursos binarios o código fuente si está destinado a ser compilado (3).

Cuando un paquete está almacenado conjuntamente con otros, es necesario que sea descrito para facilitar las búsquedas que pudieran tratar de encontrarlo a partir de sus características distintivas. Una forma de localizar los paquetes a través de su descripción es haciendo uso de metadatos.

El concepto de metadatos es análogo al uso de índices para localizar objetos en vez de datos. Los metadatos en etiquetas son un enfoque importante para construir un puente sobre el intervalo semántico, así como para clasificar grandes volúmenes de paquetes (11). Algunas herramientas de sincronización de datos como Rsync, hacen uso de los metadatos para enviar sólo las diferencias existentes entre dos archivos.

1.1.1 Repositorios en GNU/Linux Nova

La creación de la distribución de GNU/Linux Nova se realiza mediante la instalación de un conjunto de paquetes de software definidos en un sistema de archivo virgen (4). Estos paquetes, junto con todo el software compatible con la distribución, está almacenado en una estructura de archivos llamada repositorio. El repositorio de Nova está firmado digitalmente para que los sistemas instalados puedan identificar cuando el software es procedente de las fuentes oficiales.Cada versión cuenta con repositorio destinado en el que son almacenados todos los paquetes compatibles con esta(6).

El repositorio de software de Nova consta con dos ramas oficiales. La primera es principal y cuenta con los paquetes con soporte oficial por la distribución, muchos de los cuales vienen por defecto en cada una de las variantes del producto (Escritorio, Ligero y Servidores). La segunda rama es extendida y contiene paquetes de software que son compatibles con la distribución. Muchos de estos paquetes constituyen desarrollos de comunidades internacionales y otros poseen licencias privativas (10).

Al igual que el de Debian, el repositorio de Nova cuenta con los todos los archivos de infraestructura dentro de la carpeta dists, en donde se encuentra una carpeta por cada distribución. Dentro de cada carpeta son de vital importancia parala seguridad delrepositorio los siguientes archivos:

- Release: información general del repositorio(arquitecturas, componentes, versión, etc).Además, se encuentra una suma de verificación de cada uno de los archivos presentesenel repositorio de la distribución.
- Release.gpg: llave digital (pública) usadapara firmar los índices del repositorio.
- InRelease: contiene la misma informaciónprovista por el archivo Release pero está firmadocon la llave digital disponible en Release.gpg.
- Contents-<arquitectura>: lista de archivoscontenidos dentro de los paquetes delrepositorio. Existe uno por cada arquitectura soportada en el repositorio y generalmentesoncomprimidos debido a su gran tamaño.
- Packages: archivo con toda la informaciónde los paquetes de una rama y arquitecturaespecífica.

1.2 Sincronización de repositorios en GNU/Linux Nova

La sincronización de datos se refiere a la idea de mantener múltiples copias de un conjunto de datos en coherencia entre sí, o para mantener la integridad de los datos. Es el proceso de alineación entre los datos provenientes de diversas fuentes, y su continua armonización en el tiempo (4).

El concepto de sincronización también es extensible a los archivos, y es utilizada para mantener la misma versión de archivos en múltiples dispositivos. Es el proceso de asegurarse de que dos o más ubicaciones contengan las mismas versiones de los archivos. Si se agrega, modifica o elimina un archivo de una ubicación, el proceso de sincronización agregará, modificará o eliminará el mismo archivo en las otras ubicaciones (4).

El proceso de sincronización básicamente consiste en comparar el contenido de un repositorio con el del repositorio que se desea sincronizar, si son iguales significa que el repositorio esta actualizado, de lo contrario se analizan los paquetes y sus dependencias y se procede a la copia. Esta puede ser total o incremental, en dependencia de la herramienta que se utilice para realizarla.

1.3 Herramientas de sincronización

Una de las características principales que ha popularizado a los sistemas operativos Linux, son los diversos esquemas de distribución de software implementados en la web para compartir y distribuir software. Todo esto gracias a la libertad del software libre.

Existen un grupo de herramientas que permiten realizar esta operación, a continuación, se realizará un breve análisis de las características de algunas de ellas.

1.3.1 Rsync

Es una herramienta de copiado y sincronización de archivos relativamente rápida y versátil para el copiado local y remoto de archivos. Ofrece una larga lista de opciones que controlan todos los posibles aspectos de su comportamiento. Rsynccuenta con un algoritmo llamado 'delta-transfer' que reduce la cantidad de datos que se envían vía red al enviar solo las diferencias que hay entre dos archivos a nivel de sus metadatos (permisos, fechas de acceso, etc) y del contenido de sus datos a nivel de bloques en disco (12). Entre sus características se encuentran:

- Es una aplicación libre para sistemas de tipo Unix y Microsoft Windows.
- 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 es que la copia toma lugar con sólo una transmisión en cada dirección.
- Puede copiar o mostrar directorios contenidos y copia de archivos, opcionalmente usando compresión y recursión.
- Actúa como un daemon[1] de servidor, sirviendo archivos en el protocolo nativo Rsync o vía un terminal remoto como RSH o SSH.
- Permite transmitir eficientemente una estructura (como un archivo) a través de un canal de comunicación cuando el receptor ya tiene una versión diferente de la misma estructura.
- Optimiza las transferencias entre dos computadoras sobre TCP/IP,
- Funciona para la compresión y descompresión de los datos bloque por bloque, utilizando zlib al enviar y recibir, y soporte para protocolos de cifrado, tal como

[1] Daemon: servicio o programa residente es un tipo especial de proceso informático no interactivo, es decir, que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario.

SSH, lo que permite transmisión cifrada y eficientemente diferenciada de datos comprimidos.

- Adicionalmente puede utilizarse una aplicación de tunneling también para crear un túnel cifrado que asegure los datos transmitidos.
- Además de archivos, el algoritmo permite copiar directorios, aun recursivamente, así como vínculos, dispositivos, grupos y permisos. No requiere por defecto privilegios de root para su uso.
- Conserva permisos de archivos y propiedad.

1.3.2 Borgbackup

Es un programa de copia de seguridad de duplicación que admite compresión y cifrado autenticado. Su objetivo principal es proporcionar una forma eficiente y segura de hacer copias de seguridad de los datos (13). A continuación, se relacionan algunas de sus características:

- El sistema de «deduplication» permite ahorrar espacio a la hora de crear respaldos de los ficheros.
- Gran velocidad y rendimiento a la hora de crear las copias de seguridad. Esto se debe fundamentalmente al uso de C/Cython para programar las funcionalidades críticas del programa.
- Posibilidad de comprimir los backups con diferentes algoritmos.
- Posibilidad de encriptar los datos para protegerlos y verificar la integridad de los datos.
- Posibilidad de almacenar las copias de seguridad en un servidor remoto, siempre y cuando permita conexiones SSH.
- Los archivos de copia de seguridad creados, pueden ser montados como una unidad de almacenamiento. Esto nos permite examinar los contenidos del backup directamente desde un gestor de archivos.

1.3.3 Unison

Es una herramienta código abierto, bidireccional, lo cual permite mantener dos directorios completamente sincronizados sin importar las modificaciones que se realicen en uno o en otro (14) . Permite que dos réplicas de una colección de archivos y directorios se almacenen en diferentes hosts (o diferentes discos en el mismo host), se modifiquen por separado y luego se actualicen propagando los cambios de cada réplica a otra (15).

- A diferencia de las utilidades simples de duplicación o copia de seguridad, puede ocuparse de las actualizaciones de ambas réplicas de una estructura de directorios distribuida. Las actualizaciones que no entran en conflicto se propagan automáticamente. Se detectan y muestran las actualizaciones en conflicto.
- A diferencia de un sistema de ficheros distribuido, es un programa a nivel de usuario: no hay necesidad de modificar el núcleo ni de tener privilegios de superusuario en ningún host.
- Funciona entre cualquier par de máquinas conectadas a Internet, comunicándose a través de un enlace de conexión directa o de un túnel a través de una conexión SSH cifrada. Es cuidadoso con el ancho de banda de la red, y funciona bien sobre enlaces lentos como las conexiones PPP. Las transferencias de pequeñas actualizaciones a archivos grandes se optimizan utilizando un protocolo de compresión similar a Rsync.
- Es resistente al fracaso. Es cuidadoso de dejar las réplicas y sus propias estructuras privadas en un estado sensible en todo momento, incluso en caso de terminación anormal o fallas de comunicación.
- Tiene una especificación clara y precisa.

1.3.4 Análisis de las herramientas de sincronización

El análisis realizado permitió comparar tres de las herramientas de sincronización más utilizadas en la actualidad, en base a un grupo de características comunes en este tipo de software.

Tabla 1 Comparación entre las herramientas de sincronización

	Rsync	**Borgbackup**	**Unison**
Tipo aplicación	Libre	Libre	Libre
Transmisión de datos	Incremental	Dedebug	
Dirección copia	Unidireccional	Unidireccional	Bidireccional
Sincronización	Deamon	Manual y automático	Manual
Permisos	Conserva	No conserva	No conserva
Interfaz	No	Si	Si
Tiempo de copia para 20 GB de información	3 horas y 20 minutos	5 horas y 48 minutos	6 horas y 15 minutos
Velocidad	14 Mb/s	12 Mb/s	12.6 Mb/s

El resultado del análisis arrojó que Rsync utiliza una transmisión de datos incremental lo que permite copiar solamente los archivos que difieren o que hayan sido modificados. Estodisminuye el tiempo necesario para este proceso en comparación con las demás herramientas estudiadas. Otra característica positiva de Rsync es que conserva los permisos de archivos y de propiedad,algo importante a la hora de trabajar con repositorios, además puede ser ejecutado como deamon por lo que las sincronizaciones serían automatizadas.

A pesar de que esta herramienta cumple con varios de los aspectos a tener en cuenta para la solución de la problemática planteada al inicio de la investigación, la ausencia de interfaz gráfica (todas las operaciones se realizan a través de comandos) y su gran variedad de opciones la hacen compleja de utilizar. Además, al ser una herramienta general incluye opciones, chequeos y protocolos innecesarios que aumentan el tiempo de copia por lo que persiste el problema inicial de la demora en la sincronización, razón por la que se decide desarrollar una nueva herramienta para la sincronización de los repositorios.

1.4 Metodología de desarrollo de software

Según la Real Academia de la Lengua Española una metodología es un conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal. También una metodología de desarrollo de software es un conjunto de pasos y procedimientos que debe seguirse para desarrollar un software. Una metodología está compuesta por varias etapas, las tareas que se realizan en las etapas, las restricciones que deben aplicarse, las técnicas y herramientas que se deben emplear y la forma de controlar y gestionar el proyecto (16).

Las metodologías de desarrollo se software se pueden catalogar en dos grandes grupos: tradicionales o robustas y ágiles. Las metodologías tradicionales o robustas están pensadas para el uso exhaustivo de documentacióndurante todo el ciclo de vida del proyecto(17) y las metodologías ágiles se basan en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala (18).

Retrasar las decisiones y Planificación Adaptativa es el eje en cual gira la metodología ágil, el retrasar las decisiones tan como sea posible de manera responsable será ventajoso tanto para el cliente como para la empresa, lo cual permite siempre mantener

una satisfacción en el cliente y por ende el éxito del producto (18), las principales ventajas de retrasar las decisiones son:

- Reduce el número de decisiones de alta inversión que se toman.
- Reduce el número de cambios necesario en el proyecto.
- Reduce el coste del cambio.

La planificación adaptativa permite estar preparados para el cambio ya que se ha introducido en el proceso de desarrollo del software, además hacer una planificación adaptativa consiste en tomar decisiones a lo largo del proyecto, se estará transformando el proyecto en un conjunto de proyectos pequeños. Esta planificación a corto plazo permitirá tener software disponible para los clientes y además la retroalimentación para hacer la planificación más sensible, ya sea ante inconvenientes que aceleren o retrasen el producto (18). Las ideas fundamentales que presentan las metodologías ágiles son:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

1.4.1 Selección de la metodología de desarrollo

Dentro de las principales metodologías ágiles se encuentran XP (eXtreme Programming), Scrum, AUP y recientemente fue desarrollada en la Universidad de las Ciencias Informáticas (UCI) una metodología basada en AUP nombrada AUP-UCI. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan.

Para el desarrollo de la solución propuesta se seleccionó AUP-UCI debido a que es una metodología ágil y se adapta al ciclo de vida definido para la actividad productiva de la UCI. Además de que está centrada en potenciar las relaciones interpersonales como clave del éxito. A través de su utilización se promueve el trabajo en equipo, predominando el aprendizaje de los desarrolladores y un buen clima de trabajo. Se basa en la realimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida

entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios.

Esta metodología se basa en tres fases las cuales son:

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.

Contempla siete disciplinas adaptadas al ciclo de vida de los proyectos de la UCI. Se consideran los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y Diseño. Se mantiene la disciplina Implementación al igual que en AUP, en el caso de Prueba se divide en tres disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes asociadas a la parte de gestión se cubren con las áreas de procesos que define CMMIDEV v1.319 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (18).

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio, Descripción de Proceso de Negocio o Modelo Conceptual) y existen tres formas de encapsular los requisitos (Casos de Uso del Sistema, Historias de usuario, Descripción de requisitos por proceso), surgen cuatro escenarios para modelar el sistema en los proyectos. Para modelar la propuesta de solución se tomó el **escenario 4** dadas sus características. Es aplicable a 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, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una Historia de Usuario (HU) no debe poseer demasiada información, siendo esta una de las razones por las que se ajusta al desarrollo de la propuesta de solución.

1.6 Tecnologías y herramientas

Para el desarrollo de una herramienta de software resulta de vital importancia la selección de los entornos en los que se realizará la misma. Se debe tener en cuenta el modelado, el lenguaje de programación, los Entornos Integrados de Desarrollo (IDE), entre otros aspectos.

1.6.1 Modelado del sistema

El lenguaje de modelado **UML** en su versión 2.5, permite comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está construyendo y a la vez descubrir oportunidades de simplificación y reutilización. Se basa en el hecho de que un modelo es la simplificación de la realidad(19).

Las funciones de UML se pueden sintetizar en:

- **Visualizar:** Permite expresar de una forma gráfica un sistema, de manera que otra persona lo puedan entender.
- **Especificar:** Permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Visual Paradigm en su versión 16.0, se selecciona como herramienta de modelado profesional, que utiliza UML para la completa representación de las etapas por las que transita un producto de software. Este permite la realización de una amplia gama de diagramas como: casos de uso, de actividades, de despliegue, entre otros, así como la generación de código fuente desde los mismos y la documentación asociada al proceso que esté siendo modelado(20).

1.6.2 Lenguaje de programación

Python 3.8 es uno de los más potentes que hay actualmente, gracias a su simplicidad, es un lenguaje de programacióninterpretado de tipado dinámico cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma y disponible en varias plataformas(21).

- **Interpretado:** Se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución.
- **Multiparadigma:** Soporta programación funcional, programación imperativa y programación orientada a objetos.
- **Tipado dinámico:** Las variables se comprueban en tiempo de ejecución.
- **Multiplataforma:** Disponible para plataformas de Windows, Linux o MAC.
- **Gratuito:** Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1

Python contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.

1.6.3 Entorno Integrado de Desarrollo

Como IDE se selecciona **Visual Studio Code** en su versión 1.44. Esta es una herramienta gratuita y de código abierto. Es un editor de código multiplataforma diseñado para escribir aplicaciones web y en la nube, aunque puede utilizarse para el desarrollo de otras aplicaciones. Puede ser usado en Windows, Linux y Mac y decir de sus desarrolladores es rápido, ligero y poderoso. Incluye soporte para depuración, control de versionesintegrado, completamiento de código y refactorización de código entre otras opciones(22).

Cuenta con extensiones que facilitan el trabajo con diferentes lenguajes de programación, entre ellos Python, esto permite el trabajo en cualquier sistema operativo y con diferentes intérpretes específicamente para este lenguaje.

Conclusiones del capítulo

El análisis de los referentes teóricos metodológicos facilitó la comprensión de diferentes temas asociados al problema de la investigación y el estudio de los conceptos asociados al dominio de la misma aportaron claridad y conocimientos sobre el objeto de estudio. Se realizó un análisis de las herramientas que realizan la sincronización de repositorios y se concluyó que aunque cumplen con algunos de los requisitos para la solución a la problemática, no se resuelve el aspecto de la disminución del tiempo, eje principal de la investigación, por lo que todas son desechadas como posibles soluciones y se decide realizar una nueva herramienta. Para ello se realizó un estudio del estado del arte lo que

permitió identificar la metodología de desarrollo de software y las herramientas que se utilizarán en el desarrollo de la solución que se propone, teniendo como resultado las siguientes: como metodología de desarrollo de software AUP-UCI la cual será la encargada de estructurar, planificar y controlar todo el proceso de desarrollo de la aplicación, como lenguaje de programación se selecciona Python usando Visual Studio Code como IDE de desarrollo; y como herramienta CASE se utilizará Visual Paradigm.

Capítulo 2. Análisis y Diseño

Introducción

En este capítulo se muestran los artefactos correspondientes a las fases de inicio y ejecución de la metodología escogida en el capítulo anterior. Se describe la propuesta de solución para el desarrollo de la herramienta de sincronización de repositorios de GNU/Linux Nova, además se hace un análisis sobre las características y cualidades de la herramienta descritas mediante la utilización de las Historias de Usuarios (HU). Se realiza la definición de los elementos de análisis y diseño identificando la arquitectura de software a utilizar.

2.1 Descripción de la propuesta de solución

Como solución a la problemática planteada al inicio de la investigación se propone la creación de un software para la sincronización de repositorios de la Distribución Cubana GNU/Linux Nova, dicho software sincronizará en el momento en que ocurra algún cambio u/o actualización en el repositorio principal gracias al algoritmo llamado 'delta-transfer', utilizando además el menor tiempo de inconsistencia[2] posible facilitando así el tráfico en la red, igualmente podrá satisfacer las necesidades de los clientes y disminuir el almacenamiento necesario en los repositorios espejos. Para la solución se decidió desarrollar una herramienta multihilos que permita realizar la sincronización de los paquetes de forma concurrente. Debe brindar al usuario la posibilidad de sincronizar el repositorio principal, el extendido o ambos, además de hacerlo a través de SSH, RSH o Socket, podrá ser ejecutado en forma de deamon. Se utilizará para la realización de la misma las tecnologías escogidas en el capítulo anterior.

[2] Inconsistencia: consiste en la diferencia existente entre los datos que deberían ser iguales después de realizada la sincronización.

2.2 Requisitos del sistema

Unos de los puntos fundamentales en el desarrollo de sistemas son la correcta obtención de los requisitos o ingeniería de requerimientos, la cual trata de establecer las funcionalidades fundamentales que debe cumplir el sistema, así como sus restricciones. Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de realizar un sistema que ayude a resolver algún problema como el control de un dispositivo, hacer un pedido o encontrar información(23).

2.2.1 Requisitos funcionales

Los requisitos funcionales explican en detalle las tareas que el sistema debe ser capaz de realizar y expresan la naturaleza de su funcionamiento (cómo interacciona el sistema con su entorno y cuáles van a ser su estado y funcionamiento).Se determinó por parte del equipo de desarrollo en relación directa con las especificaciones del cliente definir un requisito funcional de alto nivel que se desglosa en los requisitos funcionales de la aplicación que se muestran a continuación.

- **RF1:** Sincronizar repositorios. (**Alto nivel**)
- **RF1.1:**Sincronización inicial de los repositorios
- **RF1.2:**Sincronización incremental de los repositorios
- **RF1.3:** Restaurar Backup.
- **RF1.4:** Información de las sincronizaciones.
- **RF1.5:** Notificaciones del proceso.

2.2.2 Requisitos no 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. Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo sobre el proceso de desarrollo y estándares.

- **Requisitos de software:**
    - **RNF 1:** Sistema operativo Distribución libre GNU/Linux Nova.
    - **RNF 2:** Debe existir un servidor de repositorio backup para sincronizar el repositorio.
- **Requisitos de implementación:**
    - **RNF 3:** Se trabajará usando la arquitectura Tuberías y Filtros.

- **Requerimientos de rendimiento:**
    - **RNF 4:** El servidor debe contar como mínimo con un procesador DualCore y 4 gb de RAM.
    - **RNF 5:** La herramienta debe ser escalable, permitiendo incorporarle nuevas funcionalidades sin afectar las existentes.

2.2.4 Validación de requisitos

La validación de los requisitos, obviamente tiene como objetivo comprobar que estos son correctos. Esta fase debe realizarse o de lo contrario se corre el riesgo de implementar una mala especificación, con el costo que eso conlleva. Los parámetros a validar en los requisitos son:

- **Validez:** no basta con preguntar a un usuario, todos los potenciales usuarios pueden tener puntos de vista distintos y necesitar otros requisitos.
- **Consistencia:** no debe haber contradicciones entre unos requisitos y otros.
- **Completitud:** deben estar todos los requisitos. Esto es imposible en un desarrollo iterativo, pero, al menos, deben estar disponibles todos los requisitos de la iteración en curso.
- **Realismo:** se pueden implementar con la tecnología actual.
- **Verificabilidad:** tiene que existir alguna forma de comprobar que cada requisito se cumple.

Se escogió para la validación de los requisitos de la herramienta la técnica prototipo de interfaz de usuario. Esta es una técnica de representación aproximada de la interfaz de usuario de un sistema software que permite a clientes y usuarios entender más fácilmente la propuesta de los ingenieros de requisitos para resolver sus problemas de negocio. Los dos tipos principales de prototipos de interfaz de usuario son:

- **Desechables:** se utilizan sólo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en software.
- **Evolutivos:** una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final.

A continuación, se muestra la Figura 2 con el prototipo de interfaz de usuario del tipo Desechable.

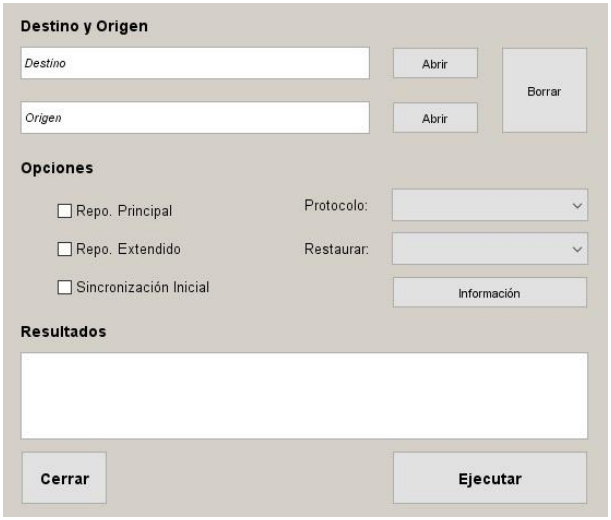

Figura 2 Prototipo de interfaz de usuario para la validación de requisitos

2.3 Historias de Usuario

Las historias de usuarios conforman la parte central del Escenario 4 de la metodología AUP-UCI pues definen lo que se debe construir en el proyecto de software. Tienen una prioridad (Alta, Media, Baja) asociada, definida por el cliente y la importancia que tiene para el funcionamiento de la herramienta las funcionalidades antes mencionadas.

El riesgo de desarrollo está dado por la ausencia del desarrollador por enfermedad o por pérdida de información imprescindible. El tiempo de cada funcionalidad será estimado por el desarrollador. La estimación del tiempo está dada de acuerdo al tiempo que toma la tarea en ser desarrollada. A continuación, se muestran las historias de usuario definidas para el desarrollo de la herramienta.

Tabla 2. HU1 Sincronización inicial de los repositorios

Historia de usuario	
Numero: HU1	**Usuarios:** Administradores
Nombre de la HU: Sincronización inicial de los repositorios.	
Prioridad: Alta	**Riesgo en Desarrollo:** Media. Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.
Tiempo estimado: Se estima para su implementación 40 días.	
Programador responsable: Miguel Luis Hernández Brocat	
Descripción: Se hace una copia completa del repositorio de archivos hacia el servidor de copias configurado para la aplicación, iniciando de esta manera el primer backup	

completo para luego a partir de este hacer backup diferenciales. El sistema debe brindar la posibilidad de elegir el repositorio que se desea actualizar (principal, extendido o ambos), así como la opción de seleccionar el protocolo a utilizar para la transferencia de datos.
Observaciones:

Tabla 3. HU2 Sincronización incremental de los repositorios

Historia de usuario	
Numero: HU2	**Usuarios:** Administradores
Nombre de la HU: Sincronización incremental de los repositorios	
Prioridad: Alta	**Riesgo en Desarrollo:** Media. Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.
Tiempo estimado: Depende de la terminación de las principales partes del software debido a que es el requisito funcional principal. Se estima alrededor de 60 días.	
Programador responsable: Miguel Luis Hernández Brocat	
Descripción: Se analizan los archivos del repositorio buscando los cambios realizados en el mismo con respecto a su última sincronización, como la eliminación, modificación, y agregación de archivos. A partir de haber analizado los cambios se procede a hacer una copia de los ficheros hacia el servidor remoto en dependencia al tipo de backup seleccionado.El sistema debe brindar la posibilidad de elegir el repositorio que se desea actualizar (principal, extendido o ambos), así como la opción de seleccionar el protocolo a utilizar para la transferencia de datos.	
Observaciones: El tipo de backup implementado es solamente el completo.	

Tabla 4. HU3 Restaurar backup

Historia de usuario	
Numero: HU3	**Usuarios:** Administradores
Nombre de la HU: Restaurar Backup.	
Prioridad: Alta	**Riesgo en Desarrollo:** Media. Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.

Tiempo estimado: Se estima para su implementación 20 días.
Programador responsable: Miguel Luis Hernández Brocat
Descripción: Se restaura la información del servidor de backup, dejando al repositorio principal en el estado de su última sincronización o a otra anterior especificada.
Observaciones: Inicialmente solo será posible restaurar a la última sincronización, luego se dará la opción de restaurar a un backup específico.

Tabla 5. HU4 Información de las sincronizaciones

Historia de usuario	
Numero: HU4	**Usuarios:** Administradores
Nombre de la HU: Información de las sincronizaciones.	
Prioridad: Alta	**Riesgo en Desarrollo:** Media. Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.
Tiempo estimado: Se estima para su implementación 20 días.	
Programador responsable: Miguel Luis Hernández Brocat	
Descripción: Se muestra un historial de sincronizaciones hechas en los últimos 2 meses. Dentro de la información se encuentra la identificación y la fecha en que fue hecho le backup, así como el tamaño que ocupa en el servidor de backup.	
Observaciones:	

Tabla 6. HU5 Notificaciones del proceso

Historia de usuario	
Numero: HU5	**Usuarios:** Administradores
Nombre de la HU: Notificaciones del proceso	
Prioridad: Alta	**Riesgo en Desarrollo:** Media. Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.
Tiempo estimado: Se estima para su implementación 35 días.	
Programador responsable: Miguel Luis Hernández Brocat	
Descripción: Muestra el progreso de los procesos de “Restaurar Backup”, “Sincronización inicial de los repositorios” y “Sincronizar repositorio principal”, y ofrece	

una opción de envío de correo al administrador del sistema para notificar cuando el proceso haya concluido.
Observaciones:

2.4 Estimación de esfuerzo por historias de usuario

Las historias de usuarios seleccionadas para entregar en cada iteración, sondesarrolladasy probadas en su ciclo correspondiente, teniendo en cuenta el ordenpreestablecido. Alcomienzo de cada ciclo, se realiza una reunión de planificación de laiteración. Acontinuación, en la tabla se resume la estimación del esfuerzo realizada por el desarrollador.

Tabla 7 Estimación de esfuerzo por historias de usuario

Historia de Usuario	Puntos Estimados (Semanas)
HU 1 Sincronizar repositorio principal	**8.5**
HU 2 Sincronización inicial de los repositorios	**5.5**
HU 3 Restaurar Backup	**2.5**
HU 4 Información de las sincronizaciones	**2.5**
HU 5 Notificaciones del proceso	**5.0**

2.5 Plan de iteraciones

Como resultado de determinar dicho flujo, aparece la plantilla plan de iteraciones, en la que se recogen las iteraciones a realizar con sus características, además del orden de las historias de usuario con su planificación estimada para ser implementadas. En la tabla plan de iteraciones, se incluyen los siguientes campos: **Número de iteración:** que contiene el identificador de la iteración que se va a desarrollar; **Descripción de la iteración:** breve descripción del objetivo de la iteración; **Orden:** contiene los identificadores de las HU a implementar en la iteración, en el mismo orden en que se deben realizar; y **Duración total:** cantidad de semanas que durará realizar la iteración, la que depende del tiempo estimado de las HU propuestas.

Tabla 8 Plan de iteraciones

Número de Iteración	Descripción de la Iteración	Orden	Duración Total

Iteración 1	En esta iteración de desarrollan las funcionalidades priorizadas como altas, es decir, las HU que se consideran con mayor complejidad de desarrollo	HU1 HU2 HU3 HU4	19
Iteración 2	En esta iteración de desarrollan las funcionalidades priorizadas como media, es decir, las HU que se consideran con unacomplejidad de desarrollo menor.	HU5	5

2.6 Descripción de la arquitectura del sistema

Para el desarrollo de la arquitectura, se tuvo en cuenta el estilo arquitectónico Tuberías y filtros debido a que facilita en gran medida el desarrollo de la aplicación. En este estilo arquitectónico los componentes, llamados filtros, leen y procesan sus entradas para producir un flujo de salida. Existen diferentes tipos de filtros:

- Filtro fuente o source filter: producen un flujo de salida sin recibir ninguno de entrada.
- Filtro transformador o transform filter: recibe un flujo de entrada, lo procesa y genera un flujo de salida.
- Filtro consumidor o sink filter: recibe un flujo de entrada y no produce un flujo de salida.

Los conectores de procesos son las tuberías que llevan la salida de un filtro hacia la entrada del otro. Este estilo arquitectónico tiene muchas ventajas como la facilidad de ser descrito, entendido e implementado y su modularidad pues resultan irrelevantes los detalles de la implementación de cada uno de los filtros siempre que estén bien descritas sus entradas, salidas y objetivos.

2.6.1 Patrones

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular(17). Para el desarrollo de la propuesta de solución se utilizarán los siguientes patrones:

- Patrones GRASP: los Patrones de Principios Generales para Asignar Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

- Experto en información: las responsabilidades deben ser asignadas a las clases que poseen la información para realizar dicha responsabilidad.
- Alta cohesión: una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el resto de las clases.
- Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- Controlador: es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones. Se evidencia el uso de este patrón en la aplicación, ya que para cada petición o evento que se genere en el mismo, existe un controlador con la responsabilidad de obtenerla y devolver una respuesta. La respuesta puede ser mostrar una vista, ejecutar un método, devolver un mensaje, etc.

Conclusiones del capítulo

En este capítulo fueron descritos los procesos que intervienen en el negocio. Además, se presentó la propuesta de solución para darle respuesta a la problemática planteada al inicio del trabajo.Como parte del proceso de análisis y diseño de la solución propuesta se realizó la extracción de los requisitos funcionales y no funcionales de la herramienta a desarrollar, que fueron posteriormente descritos en las historias de usuarios que permitieron desarrollar las distintas funcionalidades que debe cumplir el sistema para solucionar las necesidades detectadas.Se diseñó una aplicación, utilizando el estilo arquitectónico Tuberías y filtros, para que ejecute el procedimiento definido previamente, dicha arquitectura garantizará una mayor organización, reutilización de funciones y un código más legible.Al concluir el presente capítulo, se han creado las condiciones para efectuar la implementación de la herramienta de sincronización de repositorios para la distribución cubana de GNU\Linux Nova.

Capítulo 3.Implementación y Pruebas

Introducción

En el actual capítulo se describen los resultados obtenidos durante la ejecución de las pruebas, teniendo en cuenta el flujo de implementación del sistema, con el objetivo de definir la organización del código, la implementación de los elementos de diseño y la integración de los diferentes componentes, generando un ejecutable entregable o producto final. Cabe señalar que luego de su culminación, este debe cumplir con todos los requisitos funcionales y no funcionales del software.

3.1 Implementación

La implementación constituye una de las fases más importantes del desarrollo de software. En ella se toman como punto de partida los resultados obtenidos en el diseño, implementándose el sistema en términos de componentes como ficheros de código binario, código fuente, scripts y ejecutables. Su importancia reside en que se obtiene como consecuencia un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de software.

3.1.1 Tareas de ingeniería o programación

Para alcanzar los objetivos de una iteración es necesario completar las HU que están presentes en estas, por lo que se hace necesario saber cuáles son las tareas de ingeniería que componen cada una, las cuales harán posible el cumplimiento de los objetivos de cada HU. En ellas están presentes los siguientes campos: **Número de tarea:** que contiene un número consecutivo en base a la historia de usuario correspondiente; **Número historia de usuario:** que contiene el identificador de la HU a la que pertenece esta tarea; **Nombre tarea:** contiene un nombre que identifica a la tarea; **Tipo de tarea:** contiene el tipo de tarea, que puedes ser de desarrollo, corrección, mejora, o la especificación de otra; **Puntos estimados:** estimación en semanas de la duración de la tarea; **Fecha inicio:** contiene la fecha de inicio de la tarea; **Fecha fin:** contiene la fecha de fin de la tarea y **Descripción:** que contiene la descripción de la tarea.

Tabla 9. Tarea de desarrollo 1. Sincronización inicial de los repositorios

Tarea	
Número de tarea: 1	**Número de historia de usuario:** HU1
Nombre de la tarea: Sincronización inicial de los repositorios.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 5.5
Fecha de inicio: 13 enero	**Fecha fin:** 28 febrero
Descripción: Esta tarea realiza una copia completa de cada fichero del repositorio seleccionado hacia un servidor de Backup externo, iniciando así la primera sincronización completa.	

Tabla 10. Tarea de desarrollo 2. Sincronización incremental de los repositorios

Tarea	
Número de tarea: 2	**Número de historia de usuario:** HU2
Nombre de la tarea: Sincronización incremental de los repositorios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 8.5
Fecha de inicio: 1 noviembre	**Fecha fin:** 12 enero
Descripción: Esta tarea realiza una comparación entre los ficheros existentes en el repositorio y en el servidor a actualizar y realiza una copia de aquellos que han sido modificados desde la última sincronización.	

Tabla 11. Tarea de desarrollo 3. Restaurar backup

Tarea	
Número de tarea: 3	**Número de historia de usuario:** HU3
Nombre de la tarea: Restaurar Backup	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.5
Fecha de inicio: 1 marzo	**Fecha fin:** 22 marzo
Descripción: El resultado de realizar esta tarea es que restaura el estado de los repositorios a la última sincronización realizada o a una anterior elegida.	

Tabla 12 Tarea de desarrollo 4. Información de las sincronizaciones

Tarea	
Número de tarea: 4	**Número de historia de usuario:** HU4
Nombre de la tarea: Información de las sincronizaciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2.5
Fecha de inicio: 23 marzo	**Fecha fin:** 10 abril
Descripción: Muestra una lista de información acerca de las ultimas	

sincronizaciones realizadas, así como la fecha y su identificación.

Tabla 13 Tarea de desarrollo 5. Notificaciones del proceso

Tarea	
Número de tarea: 10	**Número de historia de usuario:** HU5
Nombre de la tarea: Notificaciones del proceso	
Tipo de tarea: Desarrollo	**Puntos estimados:** 5.0
Fecha de inicio: 11 abril	**Fecha fin:** 20 mayo
Descripción: Realiza el envío de un correo notificando de la finalización de los procesos de sincronización, además muestra un progreso del estado y porciento del proceso de sincronización.	

3.1.2 Estándar de código

Un estándar de programación es una forma de "normalizar" la programación para que, al trabajar en un proyecto cualquiera, las personas involucradas en el mismo tengan un acceso rápido y una correcta comprensión del código. Un estándar nos define la escritura y organización del código fuente de un programa. Ventajas que brinda:

- Facilitan el mantenimiento de una aplicación. Dicho mantenimiento constituye el 80% del coste del ciclo de vida de la aplicación.
- Permite que cualquier programador entienda y pueda mantener la aplicación. En muy raras ocasiones una misma aplicación es mantenida por su autor original.
- Los estándares de programación mejoran la legibilidad del código, al mismo tiempo que permiten su compresión rápida.

Para implementar el presente sistema se utilizó el estándar de codificación establecido por el lenguaje de programación Python(24).

3.1.3 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes: hardware y software, en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

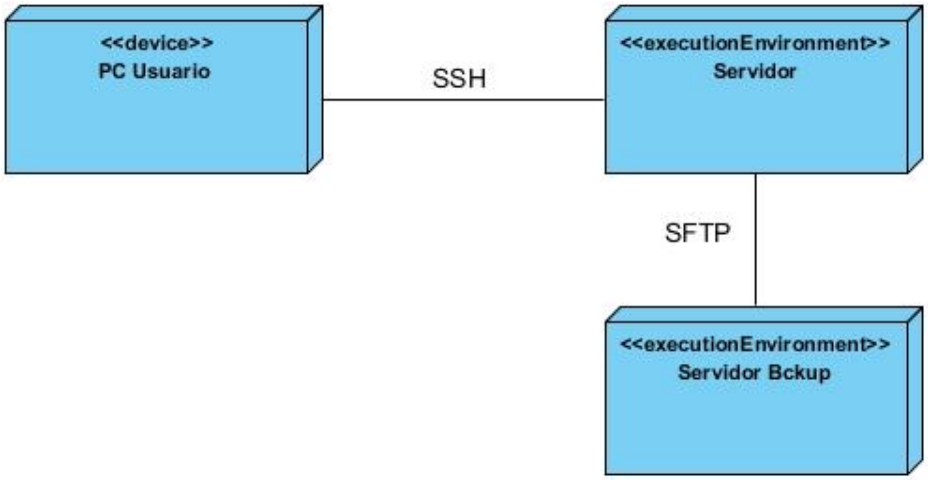

Figura 3 Diagrama de despliegue de la solución

3.2 Pruebas

La prueba es un proceso de ejecución de un programa con la intención de descubrir un error, no puede asegurar la ausencia de defectos, sino demostrar la presencia de los mismos en el software, por lo que constituye el único instrumento adecuado para determinar el status de la calidad de un producto. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema en su totalidad, con el objetivo de medir el grado en que este cumple con los requerimientos. En ellas se usan casos de prueba, especificados de forma estructurada mediante diferentes técnicas. Entre sus objetivos están:

- Verificar que todos los requisitos se han implementado correctamente.
- Detectar defectos en el software.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo en el menor tiempo y esfuerzo posible.

Para evaluar la calidad de la solución propuesta se realizarán pruebas unitarias y pruebas de aceptación. Las pruebas unitarias son creadas por los desarrolladores, son una técnica de caja blanca y consisten en comprobar la lógica interna del software(23). Estas se fueron aplicando luego de concluir cada tarea de implementación para asegurar que no se arrastraran errores de codificación a la próxima etapa.

3.2.1 Pruebas de Aceptación

Las pruebas de aceptación son pruebas funcionales, pero vistas directamente desde el cliente, demostrándole que la funcionalidad está terminada y correcta, es decir, se realizan con el fin de verificar si el producto ha sido desarrollado de acuerdo con las normas o criterios establecidos y que cumplen con todos los requisitos especificados por el cliente.

Uno de los métodos utilizados en la realización de esta prueba es el de caja negra, el cual se centra en lo que se espera de un sistema, intentando encontrar casos en que el sistema no se atiene a su especificación. El probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo la herramienta internamente. A continuación se muestran algunos de los casos de prueba desarrollados para la ejecución de las pruebas de aceptación, los restantes se encuentran en los anexos del documento:

Tabla 14. Caso de Prueba 1. Sincronización inicial de los repositorios

Caso de prueba de aceptación	
Código de la prueba: CP1_P1	**Nombre de la historia de usuario:** HU1 Sincronización inicial de los repositorios
Descripción de la prueba: Se ejecuta el proceso de sincronización inicial del repositorio principal.	
Condiciones de ejecución: Estar conectado al repositorio y seleccionar a través de qué protocolo se desea hacer la transferencia de datos.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción “Sincronización inicial” seleccionado y empieza a sincronizarse dicho repositorio.	
Resultados esperados: El usuario realiza exitosamente la sincronización inicialhacia el repositorio extendido.	
Clasificación de la prueba: Satisfactoria	

Tabla 15. Caso de Prueba 2. Sincronización incremental de los repositorios

Caso de prueba de aceptación	
Código de la prueba: CP2_P1	**Nombre de la historia de usuario:** HU2 Sincronización incremental de los repositorios

Descripción de la prueba: Se ejecuta el proceso de sincronización incremental del repositorio principal.
Condiciones de ejecución: Estar conectado al repositorio y seleccionar a través de qué protocolo se desea hacer la transferencia de datos.
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción “Repo Principal” seleccionado y empieza a sincronizarse dicho repositorio.
Resultados esperados: El usuario sincroniza exitosamente el repositorio principal. Solo se copiarán los archivos que fueron modificados desde la última sincronización.
Clasificación de la prueba: Satisfactoria

Tabla 16. Caso de Prueba 3. Restaurar Backup

Caso de prueba de aceptación	
Código de la prueba: CP3_P1	**Nombre de la historia de usuario:** HU3 Restaurar Backup
Descripción de la prueba: Se ejecuta el proceso de restauracióndel repositorio principal.	
Condiciones de ejecución: Estar conectado al repositorio. Seleccionar el Backup que se desea restaurar.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción de la lista desplegable “Restaurar” seleccionada y empieza a sincronizarse dicho repositorio.	
Resultados esperados: El usuario restaura exitosamente el repositorio.	
Clasificación de la prueba: Satisfactoria	

Tabla 17. Caso de Prueba 4. Información de las sincronizaciones

Caso de prueba de aceptación	
Código de la prueba: CP4_P1	**Nombre de la historia de usuario:** HU4 Información de las sincronizaciones
Descripción de la prueba: Se muestra informacióndel proceso de sincronización de repositorios en las fechas seleccionadas por el usuario.	
Condiciones de ejecución: Existir al menos una sincronización previa	
Pasos de ejecución: El usuario presiona en el botón “Información”.	

Resultados esperados: Se le muestraal usuario la información correspondiente a las sincronizaciones de los repositorios realizadas en las fechas seleccionadas.
Clasificación de la prueba: Satisfactoria

Tabla 18. Caso de Prueba 5. Notificaciones del proceso

Caso de prueba de aceptación	
Código de la prueba: CP5_P1	**Nombre de la historia de usuario:** HU5. Notificaciones del proceso
Descripción de la prueba: Una vez realizado el proceso de sincronización de repositorios se le muestra al usuario las notificaciones correspondientes.	
Condiciones de ejecución: Estar conectado al repositorio	
Pasos de ejecución: -	
Resultados esperados: Se le envíaexitosamente al usuario la notificación correspondiente alaculminación del proceso de sincronizaciónactual o la ocurrencia de un error y el progreso actual de dicha sincronización.	
Clasificación de la prueba: Satisfactoria	

3.2.2 Validación de la propuesta de solución

Debido a la situación actual ocasionada por la Covid-19 no ha resultado posible realizar las pruebas al sistema en un entorno real, donde influyen varios parámetros, entre ellos el ancho de banda y la velocidad de tráfico en la red. Sin embargo, se han ejecutado pruebas locales para verificar el funcionamiento de la aplicación.

Para ello se han realizado sincronizaciones de datos (1 GB) como backup completoinicialmente a otra carpeta local en la misma máquina, posteriormente se hicieron cambios en ficheros de textos, así como la eliminación y creación de nuevos ficheros, para hacer Backus diferenciales. Esta operación se realizó con éxito y no arrojó no conformidades al respecto.

Luego se repitió la misma prueba con un mayor volumen de ficheros (5 GB), pero esta vez se usó como servidor externo para almacenar los Backus una máquina virtual con Ubuntu Server 18.04, esta se encontraba de manera local en la misma máquina y se utilizó una red virtual para la comunicación. En este caso también se obtuvieron resultados satisfactorios, solo se observó que el proceso de copia podía mejorarse al copiar más de

un fichero simultáneamente cuando se trata de archivos más pequeños en tamaño, para aprovechar al máximo las capacidades de flujo de información de la red y los discos.

Posteriormente se repitieron las pruebas utilizando la herramienta Rsync para la sincronización, lo cual permite crear una comparación entre los tiempos de ambas para ejecutar la misma tarea. Se observó que la herramienta propuesta iguala el tiempo de Rsync en el caso de la primera prueba y presenta una disminución en el tiempo de sincronización del segundo escenario.

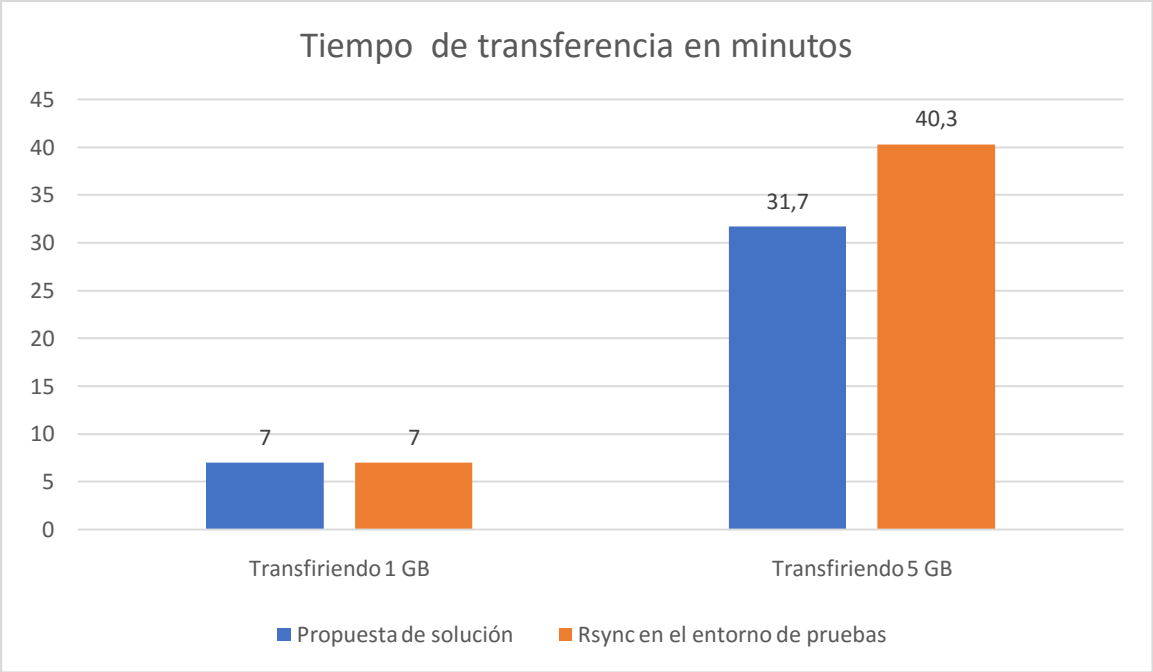

Figura 4 Comparaciones entre los tiempos de sincronización de Rsync y la solución propuesta

Estableciendo una relación entre el tiempo de sincronización de la propuesta de solución y Rsync en el entorno de pruebas se puede decir que la herramienta desarrollada es 1.27 veces, aproximadamente, más rápida que Rsync. Por tanto si se conoce que esta última tarda aproximadamente 3.19 horas para copiar 20 GB de datos en un entorno real, se puede estimar que el tiempo para realizar la misma operación con la herramienta desarrollada como propuesta de solución a la problemática planteada es aproximadamente 2.37 horas.

Conclusiones del capítulo

En el desarrollo del presente capítulo:

- Se materializó la propuesta de solución en una herramienta de sincronización de repositorios especificándose el uso de los estándares de codificación para lograr obtener claridad y organización en el código fuente de la solución.
- Se realizó el diagrama de despliegue lo cual permitió conocer la distribución física del sistema sobre una arquitectura de hardware y los protocolos de comunicación entre ellos.
- Para realizar la verificación del sistema se decidió utilizar las técnicas de caja blanca y pruebas de aceptación, para darle un mejor acabado al producto que se deseaba obtener siguiendo los requerimientos del sistema.
- La ejecución de pruebas a la herramienta de sincronización posibilitó determinar que luego de la aplicación de la solución propuesta fue posible realizar la sincronización correcta de archivos en los directorios especificados, esto solo fue probado de manera local debido a la incapacidad de probarlo en red con los repositorios oficiales de GNU/LINUX Nova.

Conclusiones

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

1. Se realizó un estudio de las principales herramientas que realizan la gestión de repositorios en los sistemas GNU/Linux y de las tecnologías existenteslo que aportó la base tecnológica para el desarrollo de la solución propuesta. Este análisis permitió identificar las características de dichas herramientas que resultaran de utilidad en el desarrollo de la solución para el entorno de sincronización de repositorios en GNU/Linux Nova.
2. La solución implementada se construyó siguiendo las bases de la arquitectura Tuberías y Filtros, la cual permitió definir características y bases importantes a la misma.
3. Se realizó el análisis, diseño e implementación de la herramienta para sincronizar repositorios en la Distribución Cubana de GNU/Linux Nova, obteniendo así una herramienta que da solución a la problemática planteada.
4. Las pruebas realizadas al producto implementado garantizaron la funcionalidad del mismo y evidenciaron que cumple con los requerimientos definidos y el objetivo trazado al inicio de la investigación.

Recomendaciones

Para futuras investigaciones y desarrollo de la herramienta se recomienda:

- Incorporar el mecanismo de encriptar los archivos en el servidor de Backup con el fin de lograr más seguridad y eficiencia

Bibliografía

1. **Free Software Foundation, Inc.** GNU. *El sistema operativo GNU.* [Online] 01 04, 2020. https://www.gnu.org/gnu/gnu.html.

2. **Pascual, Juan Antonio.** ComputerHoy. *Qué es una distribución Linux, en qué se diferencian y cómo elegir una.* [Online] 01 21, 2017. https://computerhoy.com/noticias/software/que-es-distribucion-linux-que-diferencian-como-elegir-54784.

3. **Ferrer Martínez, Juan.** Componentes de una aplicación. Empaquetado. *Desarrollo de interfaces.* Madrid, España : RA-MA, SA, 2015.

4. **Pérez Herrera, Dariem.***Sistema distribuido para automatizar la construcción de repositorios de paquetes binarios de la distribución cubana GNU/Linux Nova.* Universidad Central ¨Marta Abreu¨. Facultad de Matemática, Físicas y Computación. Departamento Ciencias de la Computación. Santa Clara : s.n., 2013. Tesis doctoral.

5. *Buenas prácticas para la migración a software libre y código abierto.* **Pérez Villazón, Yoandy, Peñalver Romero, Gladys Marsi and Veliz Pedraza, Nelio.** 4, La Habana : s.n., 2017, Serie Científica. Universidad de las Ciencias Informáticas, Vol. 10. 2306-2495.

6. **Pierra Fuentes, Allan.***Nova, distribución cubana de GNU/Linux: re-estructuración estratégica de su proceso de desarrollo.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2011. Tesis de Maestría.

7. *Distribución cubana de GNU/Linux Nova 6.0.* **Hechavarría González, Yileni, et al.** 3, La Habana : s.n., Marzo 2018, Serie Científica de la Universidad de las Ciencias Informáticas, Vol. 11, pp. 71-76. 2306-2495.

8. **Universidad del Rey Juan Carlos.** Repositorios institucionales. [Online] 2019. https://urjconline.atavist.com/repositorios-institucionales-2019.

9. *Evolución de repositorios temáticos y megarrevistas: visión 2018.* **López Borrul, Alexandre.** Barcelona : s.n., 2018, Anuario ThinkEPI, Vol. 12, pp. 316-320.

10. *GESTIÓN DE REPOSITORIOS DE PAQUETES DE NOVA CON APTLY.* **Funtes Rodríguez, Juan Manuel, Blanco Sánchez, Yarilaisi and Sierra Corredera, Luis Daniel.** La Habana : s.n., 2018. XVII Convención y Feria Internacional Informática 2018.

11. **OBS Business School .***Qué son los metadatos y por qué son tan importantes.* [Online] 2017. https://obsbusiness.school/es/blog-investigacion/direccion-general/que-son-los-metadatos-y-por-que-son-tan-importantes.

12. **González Durán, Sergio.** Linux Total. *Rsync. Manuel de uso.* [Online] 2018. https://www.linuxtotal.com.mx/index.php?cont=Rsync-manual-de-uso.

13. **The Borg Collective.** Borg Documentation. [Online] 2019. https://borgbackup.readthedocs.io/en/stable/.

14. **González Platas, Rubén.** Sincronización de ficheros bidireccional con Unison. [Online] marzo 27, 2017. https://winamic.es/blog/sincronizacion-ficheros-bidireccional-unison/.

15. **DAVIDOCHOBITS.** Unison: Sincronización y backups en Linux. *ochobitshacenunbyte.* [Online] Agosto 2019. https://www.ochobitshacenunbyte.com/2019/08/19/unison-sincronización-y-backups-en-linux.

16. **García Peñalvo, Francisco José.** Ingeniería del Software. *Ingeniería del Software.* Salamanca : Grupo GRIAL, 2018, pp. 277-388.

17. **Pressman, Roger S.***Ingeniería de software. Un enfoque práctico.* Séptima Edición. s.l. : McGraw Hill, 2010. 978-607-15-0314-5.

18. **Rodríguez Sánchez, Tamara.***Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

19. **Debrawuer, Laurent and Van der Heyde, Fien.***UML 2.5: iniciación, ejemplos y ejercicios corregidos.* s.l. : Ediciones ENI, 2016.

20. **Visual Paradigm.** Visual Paradigm 16.0 Released. *Visual Paradigm.* [Online] Julio 2019. https://www.visual-paradigm.com/features.

21. **Python.** Python 3.8 Release. *Python.* [Online] Octubre 2019. https://www.python.org/downloads/release/python-380.

22. **Visual Studio Code.** Visual Studio Code March 2020. *Visual Studio Code.* [Online] Marzo 2020. https://code.visualstudio.com/updates/v1_44.

23. **Sommerville, Ian.***Ingeniería de Software.* Novena. México : Pearson Education, 2011.

24. **Recursos Python. [Online] https://recursospyhton.com/pep8es.pdf&ved=2ahUKEwiB99nUkbTrAhWIB80KHQ2fC5QQFjABegQIDBAG&usg=AOvVaw2M3yJYnkRofKEJG2qUfHhJ.**

Anexos

Tabla 19Caso de Prueba 6. Sincronización inicial de los repositorios.

Caso de prueba de aceptación	
Código de la prueba: HU1_P2	**Nombre de la historia de usuario:** HU1 Sincronización inicial de los repositorios
Descripción de la prueba: Se ejecuta el proceso de sincronización inicial del repositorio extendido.	
Condiciones de ejecución: Estar conectado al repositorio y seleccionar a través de qué protocolo se desea hacer la transferencia de datos.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción “Sincronización inicial” seleccionado y empieza a sincronizarse dicho repositorio.	
Resultados esperados: El usuario realiza exitosamente la sincronización inicialhacia el repositorio extendido.	
Clasificación de la prueba: Satisfactoria	

Tabla 20Caso de Prueba 7. Sincronización inicial de los repositorios.

Caso de prueba de aceptación	
Código de la prueba: HU1_P3	**Nombre de la historia de usuario:** HU1 Sincronización inicial de los repositorios
Descripción de la prueba: Se ejecuta el proceso de sincronización inicial del repositorio principal y el repositorio extendido a la vez.	
Condiciones de ejecución: Estar conectado al repositorio y seleccionar a través de qué protocolo se desea hacer la transferencia de datos.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción “Sincronización inicial” seleccionado y empiezan a sincronizarse los repositorios.	
Resultados esperados: El usuario realiza exitosamente la sincronización inicialde ambos repositorios.	

Clasificación de la prueba: Satisfactoria

Tabla 21Caso de Prueba 8. Sincronización incremental de los repositorios.

Caso de prueba de aceptación	
Código de la prueba: CP2_P2	**Nombre de la historia de usuario:** HU2 Sincronización incremental de los repositorios
Descripción de la prueba: Se ejecuta el proceso de sincronización incremental del repositorio extendido.	
Condiciones de ejecución: Estar conectado al repositorio y seleccionar a través de qué protocolo se desea hacer la transferencia de datos.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción “Repo Extendido” seleccionado y empieza a sincronizarse dicho repositorio.	
Resultados esperados: El usuario sincroniza exitosamente el repositorio extendido. Solo se copiarán los archivos que fueron modificados desde la última sincronización.	
Clasificación de la prueba: Satisfactoria	

Tabla 22Caso de Prueba 9. Sincronización incremental de los repositorios.

Caso de prueba de aceptación	
Código de la prueba: CP2_P3	**Nombre de la historia de usuario:** HU2 Sincronización incremental de los repositorios
Descripción de la prueba: Se ejecuta el proceso de sincronización incremental del repositorio principal y el extendido.	
Condiciones de ejecución: Estar conectado al repositorio y seleccionar a través de qué protocolo se desea hacer la transferencia de datos.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción “Ambos Repos” seleccionado y empieza a sincronizarse dicho repositorio.	
Resultados esperados: El usuario sincroniza exitosamente ambos repositorios a la vez. Solo se copiarán los archivos que fueron modificados desde la última sincronización.	
Clasificación de la prueba: Satisfactoria	

Tabla 23Caso de Prueba 10. Restaurar Backup.

Caso de prueba de aceptación	
Código de la prueba: CP3_P2	**Nombre de la historia de usuario:** HU3 Restaurar Backup
Descripción de la prueba: Se ejecuta el proceso de restauracióndel repositorio extendido.	
Condiciones de ejecución: Estar conectado al repositorio. Seleccionar el Backup que se desea restaurar.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción de la lista desplegable “Restaurar” seleccionada y empieza a sincronizarse dicho repositorio.	
Resultados esperados: El usuario restaura exitosamente el repositorio.	
Clasificación de la prueba: Satisfactoria	

Tabla 24Caso de Prueba 11. Restaurar Backup.

Caso de prueba de aceptación	
Código de la prueba: CP3_P2	**Nombre de la historia de usuario:** HU3 Restaurar Backup
Descripción de la prueba: Se ejecuta el proceso de restauraciónde los repositorios principal y extendido.	
Condiciones de ejecución: Estar conectado al repositorio. Seleccionar el Backup que se desea restaurar.	
Pasos de ejecución: El usuario presiona el botón “Ejecutar” con la opción de la lista desplegable “Restaurar” seleccionada y empiezan a sincronizarse los repositorios.	
Resultados esperados: El usuario restaura exitosamente ambos repositorios.	
Clasificación de la prueba: Satisfactoria	