UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

CENTRO DE SOLUCIONES LIBRES DEPARTAMENTO SISTEMA OPERATIVO, FACULTAD 1

Herramienta para la gestión de descargas en Nova

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

Autor: Liz Claudia Reyes Peñate

Tutores Ing. Yileni Hechavarria González

Ing. Enmanuel Cristian de la Cruz Mora

La Habana, noviembre de 2021

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo **Liz Claudia Reyes Peñate**, con carné de identidad **98091309132** soy el autor principal del trabajo titulado ***"Herramienta para la gestión de descargas en Nova"*** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

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

Autor: Liz Claudia Reyes Peñate

Tutor: Ing. Enmanuel Cristian de la Cruz Mora

Tutor: Ing. Yileni Hechavarria González

DATOS DE CONTACTO

Ing. Enmanuel Cristian de la Cruz Mora, graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Forma parte del Centro de Software Libre donde se desempeña como programador del proyecto Nova. (ecdelacruz@uci.cu)

Ing. Yileni Echavarría González, graduada de Ingeniero en Ciencias Informáticas en el 2015 en la Universidad de las Ciencias Informáticas (UCI). Forma parte del Centro de Software Libre (CESOL) donde se desempeña como jefe de proyecto de Nova 8.0. (yileni@uci.cu)

RESUMEN

Con el avance del Internet un mayor número de personas realizan descargas de archivos, videos, documentos. La complejidad de las descargas crece y con ello el consumo de recursos en los ordenadores lo que aumenta la popularidad de los llamados gestores de descarga. En la Distribución cubana de GNU/Linux Nova no existe precedente de una aplicación propia para gestión de descargas. La presente investigación propone desarrollar una herramienta para la gestión de descargas en GNU/Linux Nova. Para ello se realiza un estudio de las herramientas más utilizadas para la gestión de descargas, se define como metodología AUP-UCI, la cual guía el proceso de desarrollo de software. Para la implementación de la solución propuesta se hace uso de Python como lenguaje de programación, como biblioteca de componentes gráficos para el desarrollo de la interfaz de usuario GTK, Visual Studio Code para la edición de códigos y como herramienta de modelado Visual Paradigm. Con todo lo antes planteado se obtuvo una herramienta que permite gestionar las descargas.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las tecnologías de la información y las comunicaciones han tenido un alcance muy importante en la sociedad en los últimos años, estas se encuentran presente prácticamente en todos los sectores de la vida moderna, desde los sectores dedicados a la ciencia e investigación, producción de bienes y servicios, educación, gobierno, al cuidado de la salud, así como nuevas formas de socializar, e incluso representan un factor crucial en el desarrollo de un país tanto así que las grandes potencias mundiales compiten día a día por tratar de tener el control de las mismas.

En Cuba a lo largo de los últimos años se ha realizado una ardua labor para lograr la informatización de la sociedad. Este proceso busca lograr eficiencia y eficacia, que permitan una mayor generación de riquezas y hagan sustentable el aumento sistemático de la calidad de vida de los ciudadanos en su desempeño familiar, laboral, educacional, cultural y social, sobre una política preferentemente orientada al uso social e intensivo de los recursos TIC, para extender sus beneficios a la mayor parte posible de la población y las instituciones.

En el Centro de Soluciones Libres (CESOL) de la Universidad de las Ciencias Informáticas (UCI), desde el 2004 se desarrolla la Distribución Cubana de GNU/Linux Nova, implementada por profesores de la UCI, con la participación de miembros de otras instituciones, para apoyar la migración a tecnologías de Software Libre y Código Abierto en el país. Nova aspira a proveer una línea de productos y servicios de calidad orientados a usuarios nacionales inexpertos en el área de las tecnologías de software libre que experimentan un proceso de migración a las mismas. El sistema operativo responde a las necesidades de las instituciones cubanas como parte del proceso de Informatización de la sociedad de Cuba y provee los valores de soberanía e independencia tecnológica.

Desde sus inicios se ha desplegado este sistema en varios Organismos de Administración Central del Estado y en diversas entidades estatales, pues presenta una serie de aplicaciones útiles tanto desde el punto de vista laboral como de ocio. Debido a estas características el número de usuarios que trabaja con el sistema ha aumentado. Con el acceso masivo a internet que se viene experimentando en Cuba durante los últimos años, el consumo de todo tipo de contenidos multimedia ha aumentado y con ello la necesidad de realizar descargas popularizándose así los llamados Gestores de Descarga.

Un gestor (o administrador o acelerador) de descargas es un programa diseñado para realizar descargas de archivos en Internet, ayudado de distintos medios como algoritmos, para ir pausando y reanudando las descargas de algún servidor FTP (*File Transfer Protocol, Protocolo* de transferencia de archivo) o página de Internet. Estos gestores cuentan con diferentes ventajas que varían según la aplicación que se escoja, pero las funciones principales más comunes que ofrecen son: acelerar descargas, pausarlas y retomarlas, automatizar y programa**r** las descargas o subidas en el horario que más nos interese (González3, 2013).

En el Distribución cubana de GNU/Linux Nova no existe precedente de una aplicación propia para gestión de descargas. Para realizar estas funciones se cuentan con varias opciones: Acudiendo al gestor propio del navegador, que presenta funciones muy básicas que no alcanzan a cubrir sus necesidades, empleando Wget, (que es una pequeña aplicación de terminal que no cuenta con una interfaz gráfica, siendo difícil su uso para los usuarios más inexpertos) o se utilizan aplicaciones de terceros con funcionalidades más avanzadas. Debido a que uno de los pilares fundamentales por los que se rige la Distribución Cubana es la soberanía tecnológica se necesita disponer de un gestor de descargas propio y fácil de usar, que pueda ser mantenido por el equipo de desarrollo.

Teniendo en cuenta lo anteriormente descrito se plantea como **problema de investigación:**

¿Como gestionar las descargas en la Distribución Cubana de GNU/Linux Nova?

Para dar respuesta al problema científico se plantea como **objetivo general**:

Desarrollar una herramienta que permita la gestión de descargas de archivos en la Distribución Cubana de GNU/Linux Nova.

Para guiar la investigación se propone como **objeto de estudio**: Procesos de gestión de descargas enmarcado en el **campo de acción**: Herramientas informáticas para la gestión de descargas.

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

1. Elaborar el marco teórico de la investigación sobre el proceso de gestión de descargas para GNU/Linux Nova.
2. Diseñar una herramienta que permita la gestión de descargas para la Distribución cubana de GNU/Linux Nova
3. Implementar la herramienta de gestión de descargas para la Distribución cubana de GNU/Linux Nova

4. Evaluar el gestor de descargas para la Distribución cubana de GNU/Linux Nova mediante la aplicación de técnicas, métricas y pruebas.

Preguntas científicas:

- ¿Cuáles son los presupuestos teóricos que fundamentan la gestión de descargas en la distribución cubana de GNU/Linux Nova?
- ¿Qué aspectos se deben tener en cuenta para diseñar un gestor de descargas en la distribución cubana de GNU/Linux Nova?
- ¿Cuáles son las tecnologías más adecuadas para implementar el gestor de descargas en la distribución cubana de GNU/Linux Nova?
- ¿Qué técnicas, métricas y pruebas aplicar para la evaluación del gestor de descargas en la distribución cubana de GNU/Linux Nova?

Métodos teóricos:

- **Analítico-Sintético**: es el método de investigación que consiste en la desmembración de un todo, descomponiéndolo en sus partes o elementos, para observar las causas, la naturaleza y los efectos. Se utiliza para descomponer el problema de investigación en los elementos que componen una gestión de descargas con el objetivo de profundizar en su estudio por separado y luego sintetizarlos en la solución propuesta.

- **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 gestión de descargas, identificando elementos que los caractericen y aspectos para fundamentar la propuesta de solución a la problemática planteada.

Métodos empíricos:

- **Observación**: se utilizó a través del estudio realizado a las herramientas informáticas que existen para realizar el proceso de gestión de descargas (Ver anexo 2).

- **Entrevista:** se realizó una entrevista a trabajadores del Centro de Software Libre con el objetivo de esclarecer dudas y obtener requisitos orientados a la solución (Ver anexo 1).

El presente documento se encuentra dividido en tres capítulos.

Capítulo 1: Fundamentos y referentes teórico-metodológicos sobre el proceso de gestión de descargas

En este capítulo queda reflejada la fundamentación teórica de la investigación sobre el proceso de gestión de descargas a través del estudio de sistemas homólogos; además se definen la metodología de desarrollo de software, y los lenguajes y las herramientas para la modelación e implementación del módulo para la gestión de copias de seguridad.

Capítulo 2: Análisis y Diseño de la herramienta para la gestión de descargas

En este capítulo se especifican los requisitos tanto funcionales como no funcionales necesarios para el desarrollo de la solución propuesta. También se describen las historias de usuario correspondientes a los requisitos funcionales, la validación de los requisitos, la arquitectura y los patrones de diseño que se utilizan durante la implementación.

Capítulo 3: implementación y evaluación de la herramienta informática para la gestión de descargas en GNU/Linux Nova

En este capítulo se define los estándares de codificación para el lenguaje de programación Python, así como las estrategias de prueba a seguir para verificar que el sistema realizado cumple con las expectativas del cliente

Capítulo 1: Fundamentos y referentes teórico-metodológicos sobre el proceso de gestión de descargas

Introducción

En este capítulo se explican los principales conceptos que se tratan para lograr un mayor entendimiento del tema tratado. Además, se definen las tecnologías que se utilizaran 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 gestión de descargas.

1.2 Definición de conceptos

El siguiente epígrafe contiene la definición de los conceptos que a criterio de la autora son los principales en el desarrollo del trabajo investigativo. A continuación, se describen dichos conceptos.

Sistema operativo

Es un software que actúa de interfaz entre los dispositivos de hardware y los programas de usuario o el usuario mismo para utilizar un computador. Es responsable de gestionar, coordinar las actividades y llevar a cabo el intercambio de los recursos y actúa como intermediario para las aplicaciones que se ejecuta. (Morales, 2015).

GNU/Linux Nova

Distribución de GNU/Linux desarrollada en la Universidad de las Ciencias Informáticas por estudiantes y profesores con la participación de miembros de otras instituciones, para apoyar la migración del país a tecnologías de Software Libre y Código Abierto.

Descarga

Es la acción informática por la cual un archivo que no reside en la máquina de un usuario pasa a estarlo mediante una transferencia a través de una red desde otra computadora que sí lo alberga. La duración del proceso variará en función del tamaño del fichero, de la velocidad de envío de la máquina que lo alberga y de la velocidad de descarga del que lo recibe (González3, 2013)

Archivo

Se conoce como archivo o fichero a un conjunto organizado de unidades de información (bits) almacenados en un dispositivo (Concepto, 2021).

URL

Se conoce como URL (siglas del inglés: *Uniform Resource Locator*, Localizador Uniforme de Recursos) a la secuencia estándar de caracteres que identifica y permite localizar y recuperar una información determinada en la Internet (Concepto, 2021).

Soberanía Tecnológica

Es la capacidad de nuestro País para desarrollarse en dicho campo en forma autónoma. No supone autarquía (independencia absoluta) sino capacidad decisional sobre su uso y desarrollo (Fuentes, 2011).

Gestor de descargas

Es un programa diseñado para realizar descargas de archivos en Internet, ayudado de distintos medios como algoritmos, para ir pausando y reanudando las descargas de algún servidor FTP (*File Transfer Protocol, Protocolo* de transferencia de archivo) o página de Internet, es decir ,son aplicaciones independientes que te ayudan a tener un mayor control de lo que descargas, así como de las fuentes de donde se descargan los archivos y los ritmos a los que se bajan a tu ordenador. Entre sus principales características se encuentra (González3, 2013):

- Pausar las descargas de archivos muy grandes.
- Reanudar descargas interrumpidas o pausadas (especialmente para archivos muy grandes).
- Transferencias automáticas recursivas (espejos).
- Descargas programadas (incluyendo desconexión y apagado automático).
- Búsquedas de sitios espejos (*mirrors*) y gestión de diferentes conexiones para descargar el mismo archivo más rápidamente.
- Evitar que una descarga sin finalizar se corrompa si hay una desconexión accidental.
- Descargar archivos en conexiones lentas.
- Descargar varios archivos de un sitio automáticamente a través de unas reglas sencillas (tipo de archivos, archivos actualizados).

1.3 Tipos de gestores de descarga

1.3.1 Gestores de descargas continuas

Son los más comunes para la descarga de archivos de Internet. Emplea características avanzadas, entre las cuales está la recuperación de ficheros con errores producidos durante la descarga, resumen de descargas interrumpidas y un listado de últimas descargas. Estos programas cuentan con la capacidad de continuar la descarga de archivos, aunque se hayan producidos errores ya sea por la suspensión repentina del suministro de energía eléctrica o la desconexión del módem y se les puede configurar la fecha y hora en la que se debe confirmar la descarga de algún archivo en forma automática sin importar si se cambia de servidor, esto se debe a que el servidor original se encuentra ocupado en el momento. También se le puede indicar que apague la computadora cuando termine la descarga o cuando se requiera (González3, 2013).

1.3.2 Gestores de descargas por categorías

Estos programas funcionan junto al navegador y su principal labor es acelerar la descarga de los archivos de Internet, contiene funciones tales como: definición de categorías tales por el usuario, conexión y desconexión automática, agenda de descargas, además comprueba los virus automáticamente y permite establecer vínculos con los lugares de los cuales se descarga la información y ordenar los archivos que se descargan para posteriormente poder administrarlo en forma similar a como se hace con las ventanas de exploración. Antes y durante la descarga se pueden observar valores aproximados del tamaño de los archivos, el tiempo que tarda la descarga y la velocidad de transferencia (González3, 2013).

1.3.3 Gestores de descargas fragmentadas

Estos fragmentan los archivos en partes cuando el tamaño es establecido por el usuario en la configuración. Después de descargar los fragmentos los integra para conformar el archivo nuevamente. Son programas ideales para la descarga de archivos grandes cuando se integran a navegadores con servidores tanto HTTP (Hypertext Transfer Protocol) y FTP (File Transfer Protocol**)** (González3, 2013)**.**

1.4 Estudio de Homólogos

El análisis de sistemas similares para la descarga de archivos permite conocer más en profundidad características imprescindibles y posibles deficiencias a tener en cuenta para el desarrollo de la propuesta de solución. Según los sitios web Adslzone y Softzone entre los gestores de descarga más utilizados se encuentran:

Free Download Manager (FDM)

Gestor de Descargas Libre (Free Download Manager) es un gestor de descargas muy eficiente y fácil de utilizar que no solo posee una amplia variedad de herramientas y servicios fáciles de acceder, sino que también posee una interfaz eficaz que puede ser utilizada por usuarios de diferentes niveles de conocimiento técnico. FDM puede impulsar todas sus descargas hasta 10 veces, procesar archivos de medios de varios formatos populares, arrastrar y soltar los archivos directamente desde un navegador web, así como descargar simultáneamente múltiples archivos. El programa soporta protocolos HTTP/HTTPS/FTP/BitTorrent**.** Además, FDM le permite ajustar el uso de tráfico, organizar descargas, controlar las prioridades de los archivos Torrents, descargar archivos de gran tamaño eficientemente y reanudar descargas canceladas (Free Download Manager, s.f.).

Xtreme Download Manager (XDM)

Entre las principales características del Gestor de Descargas Extremo (*Xtreme Download Manager)* se encuentra un sofisticado algoritmo de segmentación dinámica, compresión de datos y reutilización de las conexiones para dotar de más velocidad al proceso de descarga. El programa soporta protocolos HTTP, HTTPS y FTP, además de cortafuegos, servidores *proxy*, redirección de archivos, *cookies* y mucho más. Otra de las funciones más interesantes que dispone el programa es la posibilidad de descargar vídeos de YouTube**,** algo que gusta a muchos usuarios. El programador inteligente nos permite limitar la velocidad y descargas en cola. Con ello se puede seguir navegando con fluidez durante la descarga. XDM permite conectarse a Internet a una hora determinada, descargar los archivos que nos interesen y al finalizar desconectar o apagar el equipo cuando termine (Xtreme Download Manager, s.f.).

Internet Download Manager (IDM)

Herramienta que permite aumentar la velocidad de descarga hasta 5 veces, reanudar y programar descargas. La recuperación integral de errores y la capacidad de reanudación reiniciarán las descargas interrumpidas o interrumpidas debido a conexiones perdidas, problemas de red, apagados de la computadora o cortes de energía inesperados. La interfaz gráfica de usuario simple hace que IDM sea fácil de usar y fácil de usar (Internet Download Manager, s.f.).

.

Internet Download Manager tiene un acelerador lógico de descarga inteligente que presenta una segmentación de archivos dinámica inteligente y una tecnología de descarga segura de varias partes para acelerar sus descargas. A diferencia de otros administradores y aceleradores de descargas, IDM segmenta los archivos descargados dinámicamente durante el proceso de descarga y reutiliza las conexiones disponibles sin etapas adicionales de conexión e inicio de sesión para lograr el mejor rendimiento de aceleración (Internet Download Manager, s.f.).

Admite servidores proxy, protocolos ftp y http, firewalls, redireccionamientos, cookies, autorización, procesamiento de contenido de audio y video. También puede arrastrar y soltar archivos o utilizar Internet Download Manager desde la línea de comandos. *Internet Download Manager* puede marcar su módem a la hora establecida, descargar los archivos que desee, luego colgar o incluso apagar su computadora cuando haya terminado (Internet Download Manager, s.f.).

uGet Download Manager (uGetDM)

uGet es un administrador de descargas de código abierto para Linux bastante conocido. Este administrador de descargas puede ser organizado de manera eficiente y altamente configurable. Entre las características principales de uGet podemos encontrar la capacidad de poner en cola, pausar y reanudar descargas, conexión múltiple, soporte para espejos, soporte multiprotocolo (incluidos HTTP, HTTPS, FTP, BitTorrent y Metalinks), categorización avanzada, monitor del portapapeles, descargas por lotes, ajustes de categoría individualizados, limitación de las velocidades de descarga, control de descargas activas totales y muchas otras. Ofrece la capacidad de descargar videos de YouTube usando la API de YouTube, por lo que no es necesario administrar las dependencias, además presenta una interfaz intuitiva, sencilla y fácil de utilizar (Uget Download Manager , s.f.).

Luego de estudiar individualmente las características más importantes que presentan algunos de los principales gestores de descargas que se utilizan a día de hoy se realiza una tabla comparativa en busca de la mejor solución para dar respuesta al problema de investigación.

Tabla 1: Comparación de herramientas homólogos.

	Sistemas informáticos para la gestión de descargas

Criterios de análisis	FDM	**XDM**	IDM	**uGetDM**
Interfaz intuitiva, sencilla y fácil de utilizar	Sí	Sí	Sí	Sí
Posibilidad de reanudar descargas	Sí	Sí	Sí	Sí
Multilenguaje (Principalmente español)	Sí	Sí	Sí	Sí
Soporte para proxy	Sí	Sí	Sí	Sí
Disponibilidad para Linux	Sí	Sí	No	Sí
Licencia / Precio	Licencia publica GPL/ gratis	Licencia publica / gratis	Propietario licencia shareware /24.95	Licencia publica LGPL / gratis
Desarrollado en Cuba	No	No	No	No

Como se observa en la taba ya existen diferentes soluciones para la gestión de descarga de archivos. IDM cuenta con innumerables funcionalidades, pero no es una opción a tener en cuenta para dar solución al problema de investigación pues solo se encuentra disponible para el sistema operativo Windows, además es un software de pago y con licencia shareware donde los autores de la aplicación conservan sus derechos de autor sobre los contenidos y no se permite modificar dichos programas ni distribuir copias modificadas. En el caso de FDM, XDM y **uGetDM** a diferencia de IDM si se encuentran disponibles para Linux gratuitamente y cuentan con gran variedad de funcionalidades, pero al no ser desarrollados por el centro CESOL las actualizaciones se realizan a voluntad de terceros y muchas veces con ellas se deja de dar soporte a algunas tecnologías empleadas por GNU/Linux Nova, teniendo que readaptar o modificar el sistema fuera de los tiempos establecidos, además que no cumplen con los valores de soberanía e independencia tecnológica que se promueven en Cuba a día de hoy. En base a las problemáticas planteadas anteriormente es necesario desarrollar una solución a medida para dar solución al problema de investigación.

1.5 Metodología de desarrollo de software

Según Avison y Fitzgerald (1995) una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo."

Para el desarrollo de la solución propuesta se selecciona la metodología de desarrollo de software Variación AUP para la UCI ,debido a que es una metodología ágil y es una variación de AUP (*Agile Unified Process*, Proceso Unificado Ágil) que logra estandarizar el proceso de desarrollo de software en los proyectos de la universidad, además de convertirse en la metodología rectora de su desarrollo productivo, ya que se adapta perfectamente al ciclo de vida de la actividad productiva de los diferentes centros de la institución (Rodríguez, 2015).

1.5.1 Fases de Variación de AUP para la UCI

La metodología Variación de AUP-UCI está forma por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad las cuales contienen las características de las cuatro fases (Inicio, Elaboración, Construcción y Transición) propuestas en AUP.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	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.
Elaboración Construcción Transición	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.

Ilustración 1: Metodología AUP variación UCI

En la elaboración de la herramienta informática propuesta, se emplea el escenario 4 de la metodología, modelando el sistema con historias de usuario, empleadas para la descripción de requisitos ya que el negocio está bien definido y el cliente estará siempre acompañando al equipo de desarrollo para validar y probar los requisitos, además de no ser un proyecto muy extenso.

1.6 Tecnologías de desarrollo

UML

(Lenguaje de Modelado Unificado, *Unified Modeling Language*) en su version 2.5 es un lenguaje que se centra en la representación gráfica de un sistema, por lo que permite construir, modelar y diseñar dicho sistema. Un modelo UML está compuesto por 3 clases de bloques de construcción: los elementos (representaciones abstractas de cosas reales o ficticias); las relaciones (relacionan los elementos entre sí); y los diagramas (son colecciones de los elementos con sus relaciones) [CITATION UML \l 3082].

Visual Paradigm

Se utiliza Visual Paradigm en su versión 8.0 como herramienta UML CASE de modelado profesional, que utiliza UML para la completa representación de las etapas por las que transita un producto de software, para la realización de diferentes diagramas entre los que se encuentran diagrama de clases, modelos conceptuales, diagrama de despliegue, prototipo de interfaz de usuarios, entre otros (Licidchart, 2017).

Visual Studio Code

Se selecciona como editor de texto Visual Studio Code el cual es desarrollado por Microsoft para Windows, Linux y macOS pues es intuitivo y fácil de utilizar ,incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que el usuario puede cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto (Visual Studio Code, s.f.)

Python

Para la realización de la propuesta de solución se selecciona como lenguaje de programación Python pues es fácil de aprender, cuenta con gran cantidad de bibliografía oficial y una amplia comunidad de desarrolladores. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Es un lenguaje ideal para el desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas de acuerdo a su elegante sintaxis y su tipado dinámico, junto con su naturaleza interpretada (Russum, 2018) .

GTK

Para la implementación de la interfaz de usuario se selecciona GTK(conocido hasta febrero de 2019 como GTK+) o *The GIMP Toolkit* que no es más que una biblioteca de componentes gráficos multiplataforma para desarrollar interfaces gráficas de usuario (GUI). Es uno de los kit de herramientas de widgets más popular para el sistema operativo GNU/Linux, teniendo un amplio soporte para Wayland y XOrg además se usa en gran cantidad de proyectos destacando entre los más famosos : GIMP, Inkscape, Pitivi, Pidgin, XChat, LibreOffice, VLC y Elementary OS (GTK, s.f.).

Wget

Se selecciona pues es una herramienta libre que mediante líneas de comandos permite la descarga de contenidos desde servidores web de una forma simple. Actualmente admite descargas mediante los protocolos HTTP, HTTPS y FTP. Entre las características más destacadas que ofrece Wget está la posibilidad de pausar las descargas en curso, reanudar descargas abortadas, eliminar archivos descargados, fácil descarga de sitios espejo complejos de forma recursiva, conversión de enlaces para la visualización de contenidos HTML localmente, soporte para *proxies* y *cookies*, entre otros.

Glade

Para realizar el diseño de la interfaz de usuario de la propuesta de solución se utiliza Glade pues es una herramienta que permite un desarrollo rápido y sencillo de interfaces de usuario para el kit de herramientas GTK y el entorno de escritorio GNOME. Las interfaces de usuario diseñadas en Glade se guardan como XML y, al usar el objeto GTK de GtkBuilder, las aplicaciones pueden cargarlas dinámicamente según sea necesario. (Glade - A User Interface Designer, s.f.). Este es un elemento muy importante a tener en cuenta pues permite integrar el código escrito en Python con la interfaz GTK, de manera simple y eficiente, permitiendo elaborar interfaces complejas en un menor tiempo.

Conclusiones del capítulo

El análisis de las funcionalidades que brindan algunos de los sistemas que permiten la gestión de descargas en el ámbito internacional, arrojó que ninguno de estos responde al objetivo general de la investigación y permitió identificar requerimientos imprescindibles a tener en cuenta en la implementación. El estudio del arte realizado permitió identificar la metodología de desarrollo de software y 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, como lenguaje de programación se utilizó Python ,como biblioteca de componentes gráficos para el desarrollo de la interfaz gráficas de usuario GTK y Visual Studio Code para la edición e códigos .

Capítulo 2: Análisis y diseño de la herramienta de gestión de descarga

Introducción

En el presente capítulo se propone una solución al problema planteado. Se presenta la descripción del negocio, los requisitos funcionales y no funcionales y la validación de estos. En el mismo se elaboran las historias de usuario necesarias para la solución, la arquitectura a utilizar y se modela el diagrama de clases.

2.2 Propuesta de solución

Analizando la situación problemática planteada se propone como solución la creación de un gestor de descargas utilizando Wget lo cual permitirá la administración de descargas en la Distribución Cubana GNU/Linux Nova. Este software al ser desarrollado en nuestro país para GNU/Linux Nova contribuye a desarrollar los valores de soberanía tecnologías que persigue nuestro país. Se utilizará para la realización de la misma las tecnologías escogidas en el capítulo anterior.

Condiciones para el funcionamiento

- Poseer un sistema operativo compatible (en este caso Nova 7).
- Estar conectado a una red.
- Tener espacio en disco para almacenar el archivo a descargar.

2.3 Representación del contexto del dominio del negocio

Para la representación del contexto del negocio a informatizar se identificaron los conceptos significativos en el problema, identificando los atributos y las asociaciones entre ellos.

Descripción de conceptos:

Usuario: es el usuario final del sistema y quien interactúa con la PC.

PC: máquina electrónica capaz de realizar un tratamiento automático de la información y de resolver con gran rapidez problemas matemáticos y lógicos mediante programas informáticos

Nova: sistema operativo que contiene el ordenador del usuario

Gestión de descarga: se realiza el proceso de descargas de los archivos desde el servidor

Servidor: contiene los archivos a descargar.

Archivo: conjunto organizado de unidades de información.

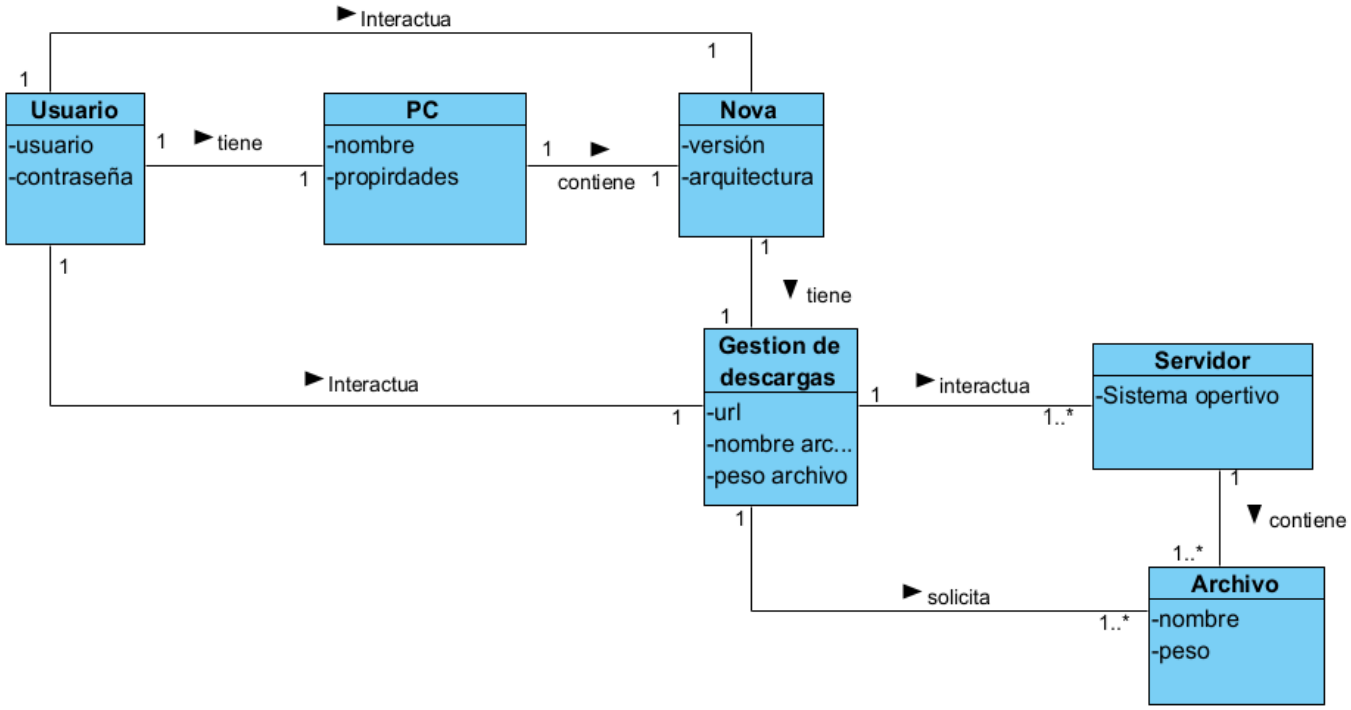

Ilustración 2 Representación del contexto del dominio del negocio

2.3.1 Requisitos

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan las necesidades de los clientes del software, lo que permite que exista una mayor comunicación entre el cliente y el equipo de desarrollo Fuente especificada no válida.. Según Pressman la ingeniería de requisitos, es un conjunto de procesos, tareas y técnicas que permiten la definición y gestión de los requisitos de un producto. Facilita la adecuada comprensión de las necesidades del cliente, confirmando su viabilidad, negociando una solución razonable, especificando la

solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional.

2.3.1 Fuentes para la obtención de requisitos:

Entre los elementos más importantes del proceso de desarrollo del software se encuentra la obtención de los requisitos, debido a que ayuda a conciliar conflictos de intereses entre los involucrados, y determinar qué tipo de software se desea desarrolla. En este proceso intervienen diferentes fuentes que permiten identificar los requisitos que forman parte de una aplicación informática. Durante esta etapa de la investigación se tuvieron en cuenta como fuentes de obtención de requisitos:

- Análisis de la representación del contexto del dominio del negocio a informatizar (Ver epígrafe 2.3)
- Análisis de las herramientas existentes (Ver epígrafe 1.2).
- Especialistas de CESOL.

2.3.2 Técnicas de identificación de requisitos

Entrevista

Se utilizó como técnica de obtención de requisitos la entrevista, que es la más tradicional de las técnicas de obtención y consiste en reuniones analista-interesado en las cuales suceden preguntas y respuestas para extraer el dominio de la aplicación. La técnica se aplicó a través de una entrevista realizada a los directivos del centro CESOL (clientes) para obtener y comprender los requisitos a tener en cuenta.

Tormenta de ideas

Consiste en reuniones entre un grupo de personas donde como primer paso sugieren toda clase de ideas sin juzgar su validez, y después de recopilar todas las ideas se realiza un análisis detallado de cada propuesta (Guerra, 2018) Se realizó una tormenta de ideas con los miembros del equipo de desarrollo de Nova, con el objetivo de concretar cuáles son las funcionalidades que no deben faltar en la propuesta de solución.

2.3.3 Requisitos funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos,

los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. Describen con detalle la función de éste, sus entradas y salidas (Sommerville, 2011).

Luego de aplicar las técnicas y procesar sus resultados se identificaron los siguientes requisitos funcionales:

Tabla 2: Requisitos Funcionales

Identificador	Requisito	Descripción	Prioridad
RF1	Descargar archivo	Debe permitir que una vez añadida la *url* del archivo el software pueda iniciar su descarga	Alta
RF2	Pausar descarga	El software debe permitir detener las descargas en curso, para ser reanudadas en otro momento	Media
RF3	Reanudar Descarga	El software debe permitir reanudar las descargas interrumpidas siempre que sea posible	Media
RF4	Listar descargas	Debe mostrar una lista con las descargas realizadas	Baja
RF5	Eliminar descarga	El software debe permitir eliminar las descargas realizadas y los archivos asociados a estas.	Baja

2.3.4 Requisitos no funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo,

sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2011).

Entre los requisitos no funcionales identificados están los que se muestran a continuación:

Tabla 3: Requisitos no funcionales

Identificador	Clasificación	Requisito
RnF1	Usabilidad	La aplicación debe ser intuitiva y fácil de utilizar.
RnF2	Implementación	La aplicación debe ser escalable
RnF3	Restricciones del diseño de implementación	La interfaz visual debe mantener el estilo empleado en la capa de personalización, empleada en Nova GNU/Linux.
RnF4	Mantenibilidad	La aplicación debe ser capaz de usarse en otras versiones de Nova

2.3.5 Descripción de requisitos

Para realizar la descripción de requisitos que permitirá la gestión de descargas en GNU/Linux Nova se hace uso de las Historias de Usuarios (HU), técnica utilizada para encapsular los requisitos funcionales del sistema propuesta en el escenario 4 de la metodología AUP-UCI.

Una HU es una descripción breve y en lenguaje sencillo de lo que se espera como salida de la implementación, y de cómo se ve beneficiado el usuario final. Con las historias de usuario se da al equipo de desarrollo el contexto y el porqué de lo que están creando. A continuación, se muestra la historia de usuario correspondientes a los requisitos funcionales.

Tabla 4: Historia de usuario #1 Descargar archivo

Historia de Usuario	
Número: 1	**Nombre:** Descargar archivo
Programador: Liz Claudia Reyes Peñate	**Iteración Asignada:**1ra
Prioridad: Alta	**Tiempo Estimado:** *7* días
Riesgo en Desarrollo: N/A	
Descripción: Debe permitir iniciar el proceso de descargas del fichero del cual se proporcionó la url **Pasos:** 1-El usuario accede al software y selecciona el botón añadir 2-Intoduce la url. 3-Selecciona el botón descargar 4-Inicia la descarga	
Observaciones: Se debe verificar que el enlace del archivo sea válido y este activo	
Prototipo elemental de interfaz gráfica de usuario:	

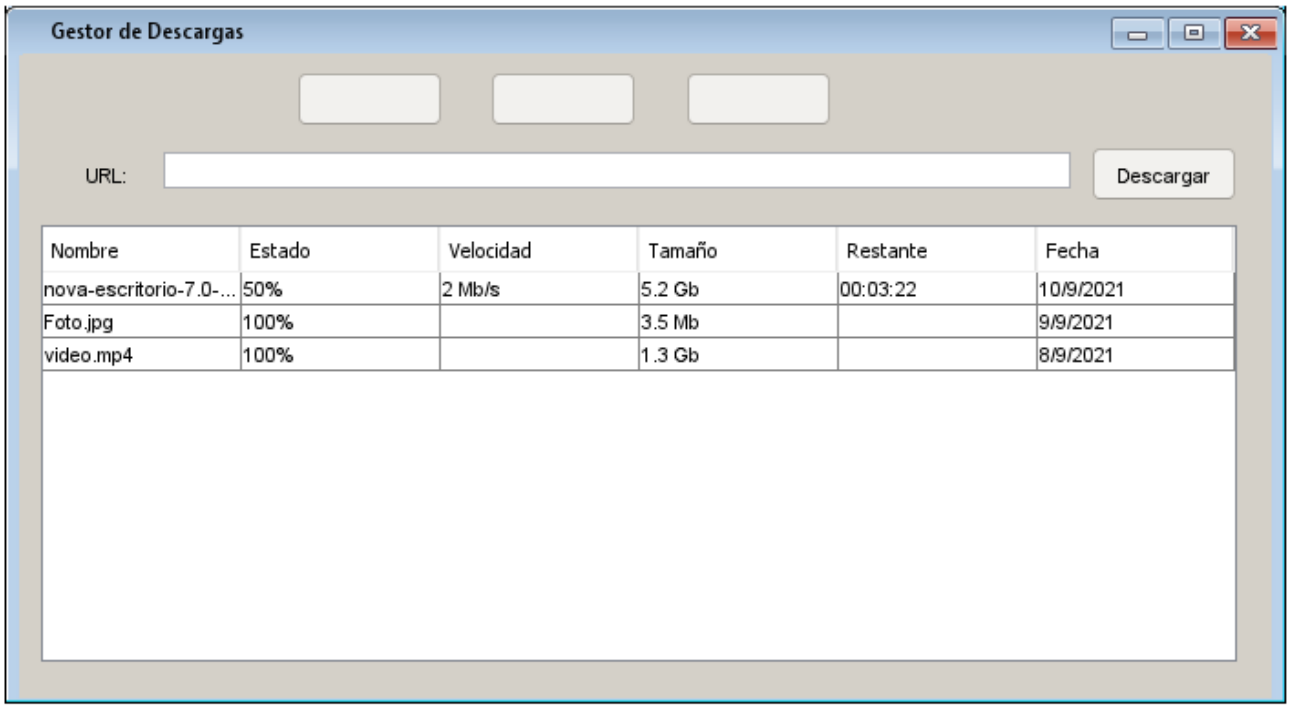

Ilustración 3: Prototipo de interfaz de usuario RF Descargas archivo

Tabla 5: Historia de usuario #2 Eliminar descarga

Historia de Usuario	
Número: 2	**Nombre:** Eliminar descarga
Programador: Liz Claudia Reyes Peñate	**Iteración Asignada:** 1ra
Prioridad: Baja	**Tiempo Estimado:** 2 días
Riesgo en Desarrollo: N/A	
Descripción: EL software debe permitir a los usuarios eliminar una descarga de la lista de descargas **Pasos:** 1-El usuario accede al software y selecciona la descarga a eliminar	

2- Selecciona el botón eliminar

3-Se elimina la descarga

Observaciones: Al eliminar la descarga se debe eliminar también el archivo asociado

Prototipo elemental de interfaz gráfica de usuario:

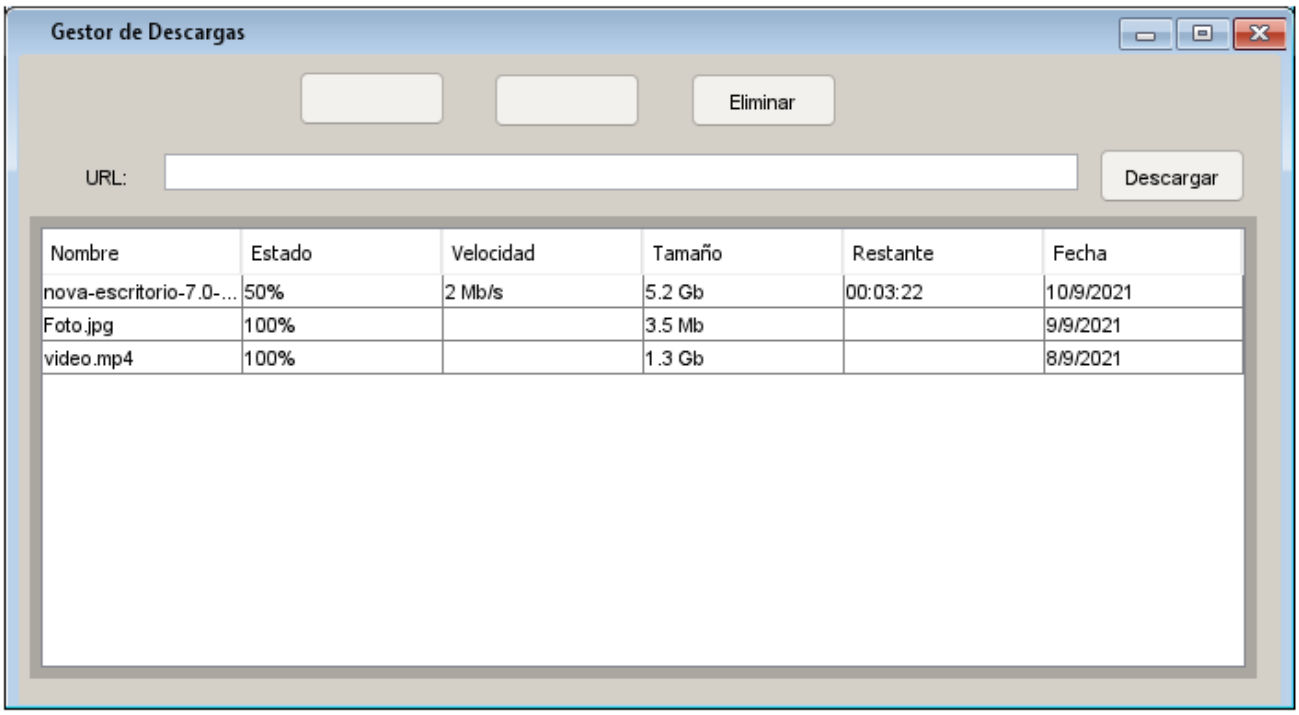

Ilustración 4: Prototipo de interfaz de usuario RF Eliminar descarga

Tabla 6: Historia de usuario #3 Pausar descarga

Historia de Usuario	
Número: 3	**Nombre:** Pausar descarga
Programador: Liz Claudia Reyes Peñate	**Iteración Asignada:**1ra
Prioridad: media	**Tiempo Estimado:** *5* días
Riesgo en Desarrollo: N/A	
Descripción: EL software debe permitir a los usuarios detener la descarga en curso al seleccionar el botón pausar **Pasos:** 1-El usuario accede al software y selecciona la descarga 2- Selecciona el botón pausar 3-Se detiene la descarga	
Observaciones: En caso de ser un sitio donde no se permite pausar las descargas debe alertar al usuario que se perderá el contenido de la misma	
Prototipo elemental de interfaz gráfica de usuario:	

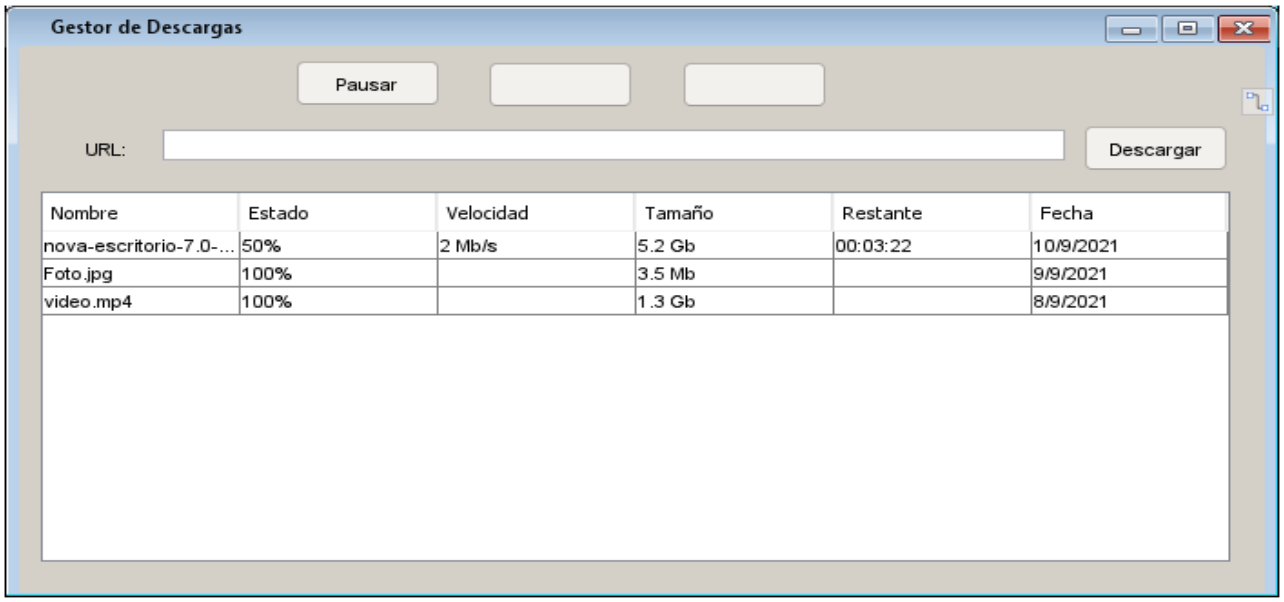

Ilustración 5: Prototipo de interfaz de usuario RF Pausar descarga

Tabla 7: Historia de usuario #4 Reanudar descarga

Historia de Usuario	
Número: 4	**Nombre:** Reanudar descarga
Programador: Liz Claudia Reyes Peñate	**Iteración Asignada:**1ra
Prioridad: media	**Tiempo Estimado:** *5* días
Riesgo en Desarrollo: N/A	
Descripción: EL software debe permitir a los usuarios reanudar una descarga previamente seleccionada que se encuentra pausada al seleccionar el botón reanudar.	
Observaciones: En caso de ser un sitio donde no se permite reanudar las descargas debe alertar al usuario que no es posible reanudar	

Pasos:

1-El usuario accede al software y selecciona la descarga que desea reanudar

2- Selecciona el botón reanudar

3-Se reanuda la descarga

Prototipo elemental de interfaz gráfica de usuario:

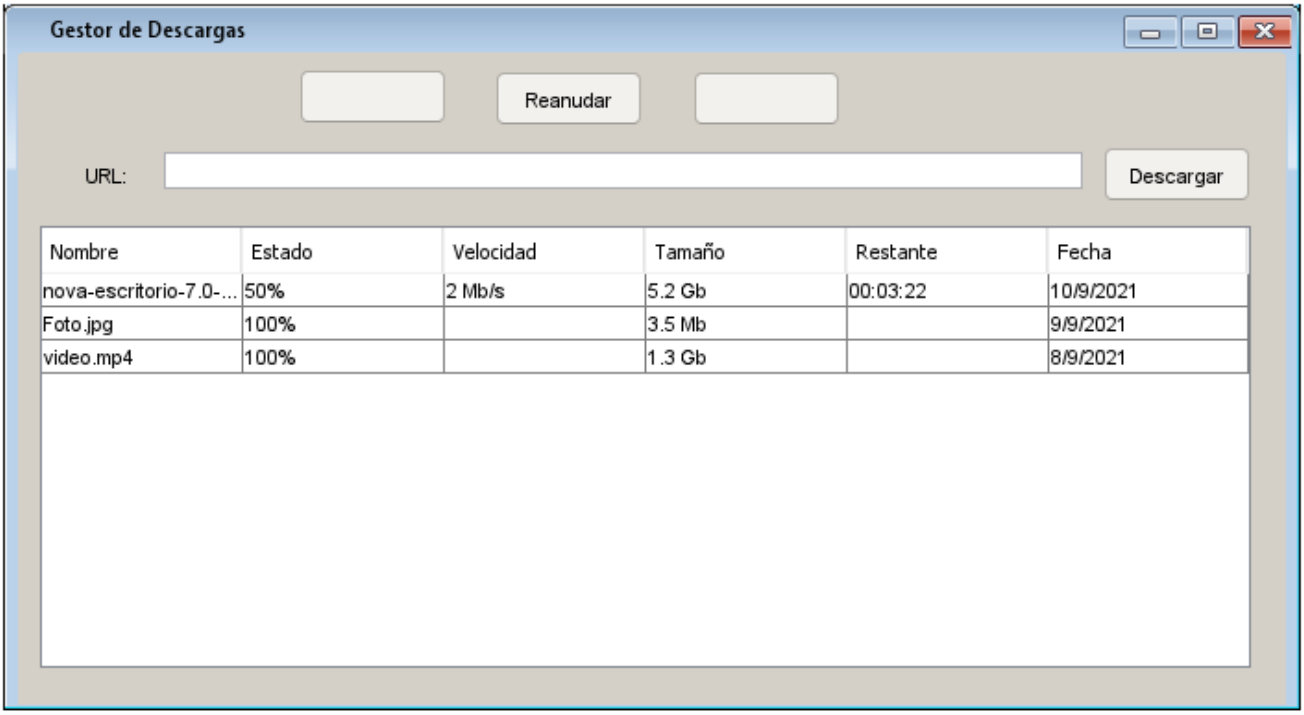

Ilustración 6: Prototipo de interfaz de usuario RF Reanudar descarga

Tabla 8: Historia de usuario #5 Listar descarga

Historia de Usuario

Número: 5	**Nombre:** Listar descarga
Programador: Liz Claudia Reyes Peñate	**Iteración Asignada:**1ra
Prioridad: baja	**Tiempo Estimado:** *4* días
Riesgo en Desarrollo: N/A	
Descripción: EL software debe mostrar una tabla compuesta por 5 columnas especificando nombre, estado, velocidad, tamaño, tiempo restante y fecha de las descargas realizadas y en curso.	
Observaciones: El nombre incluye la extensión del archivo.	

Prototipo elemental de interfaz gráfica de usuario:

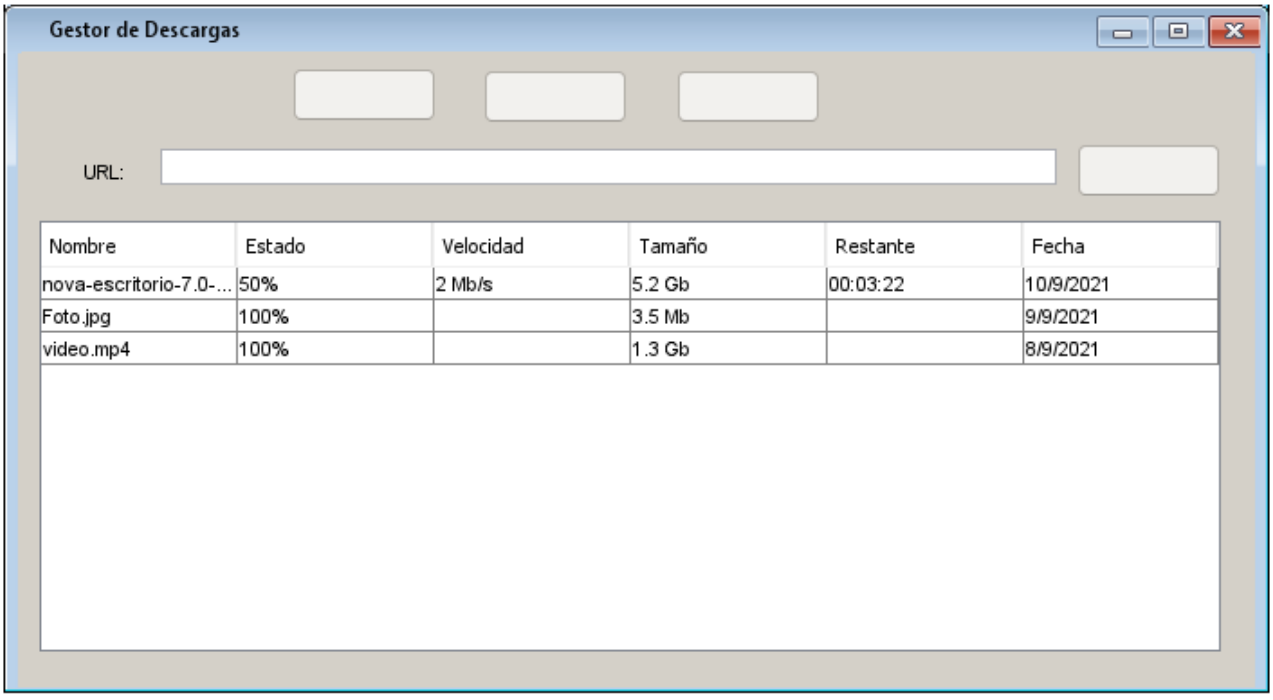

Ilustración 7: Prototipo de interfaz de usuario RF Listar descarga

2.4 Validación de requisitos de software

La calidad de los productos del trabajo que se generan como consecuencia de la ingeniería de los requerimientos se evalúa durante el paso de validación. La validación de los requerimientos analiza la especificación a fin de garantizar que todos ellos han sido enunciados sin ambigüedades; que se detectaron y corrigieron las inconsistencias, las omisiones y los errores, y que los productos del trabajo se presentan conforme a los estándares establecidos para el proceso, el proyecto y el producto (Pressman, 2010).

2.4.1 Prototipado de interfaz de usuario

El prototipado de interfaz de usuario es una técnica de representación aproximada de la interfaz de usuario de un software que permite a los clientes, usuarios finales y equipo de desarrollo explorar un diseño de interfaz de usuario alcanzable y adecuado que cumpla los requisitos, ayudando a reducir las distancias entre lo que es necesario y lo que es factible. El objetivo principal de la creación de un prototipo de interfaz de usuario es poder probar el diseño de estas interfaces, incluyendo la capacidad de utilización antes de que empiece el desarrollo real. De este modo, se puede garantizar que se está construyendo el sistema correcto, antes de dedicar demasiado tiempo y recursos al desarrollo de la solución informática (IBM MQ, s.f.) (Ver II.1.2)

Una vez diseñados los prototipos de interfaz grafía de usuarios se procedió a presentárselos al cliente obteniendo una respuesta satisfactoria por parte del mismo.

2.4.2 Diseño de caso de prueba (DCP)

El diseño de casos de prueba es una parte de las pruebas de componentes y sistemas en las que se diseñan los casos de prueba (entradas y salidas esperadas) para probar el sistema. El objetivo del proceso de diseño de casos de prueba es crear un conjunto de casos de prueba que sean efectivos descubriendo defectos en los programas y muestren que el sistema satisface sus requerimientos (Pressman, 2010).

Tabla 9: Diseño de casos de prueba RF Descargar archivo

Diseño de caso de prueba				
Escenario	Descripción	URL	Respuesta del sistema	Flujo central

EC 1.1 Descargar archivo	Debe permitir al usuario realizar la descarga del archivo.	V http://repo.nova.cu/isos/nova-escritorio-7.0-amd64.iso	Descarga completada	1-El usuario accede Al software y selecciona el botón añadir 2-Intoduce la url. 3-Selecciona el botón descargar 4-Inicia la descarga 5-El sistema envía la alerta
EC 1.2 No se crea correctamente el archivo	No permite al usuario descargar el archivo correctamente.	I NA	Introduzca la url	1-El usuario accede Al software y selecciona el botón añadir 2-El usuario no Introduce la url. 3-Selecciona el botón descargar 4-El sistema envía la alerta

Tabla 10: Diseño de casos de prueba RF2 Eliminar descarga

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar	Debe permitir al	La descarga fue	1-El usuario accede al software y

descarga	usuario eliminar una descarga	eliminada con éxito	selecciona la descarga a eliminar 2- Selecciona el botón eliminar 3-El sistema envía la alerta
EC 1.2 No se elimina correctamente la descarga	No permite al usuario eliminar una descarga.	La descarga no fue eliminada	1-El usuario accede al software y no selecciona la descarga a eliminar 2- Selecciona el botón eliminar 3-El sistema envía la alerta

2.5 Análisis y Diseño

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos (Rodríguez, 2015)

2.5.1 Modelo arquitectónico

El diseño arquitectónico representa la estructura de los datos y de los componentes del programa que se requieren para construir un sistema basado en computadora. Considera el estilo de arquitectura que adoptará el sistema, la estructura y las propiedades de los componentes que lo constituyen y las interrelaciones que ocurren entre sus componentes arquitectónicas. Es una representación que permite analizar la efectividad del diseño para cumplir los requerimientos establecidos, considerar alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil y reducir los riesgos asociados con la construcción del software (Pressman, 2010).

Modelo N-capas

El estilo arquitectural en capas (*N-Layer*), es un sub-estilo dentro del estilo llamada y retorno que se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva

de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan (Llorente, 2010).

Para la realización de la propuesta de solución se emplea la arquitectura **N-capas** pues distribuyendo las capas entre múltiples sistemas (físicos) puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento, permite realizar actualizaciones en el interior de las capas sin que esto afecte al resto del sistema, además, muestra una vista completa del modelo y a la vez proporciona suficientes detalles para entender las relaciones entre capas, etc. En este caso se definen **3 capas**: Presentación, Lógica de negocio y Almacenamiento (variante tres capas).

Presentación: es la capa diseñada para recibir solicitudes de almacenamiento o recuperación de información desde la capa de lógica de negocio. Es la encargada de presentar al usuario los conceptos de negocio mediante una interfaz de usuario (UI, del inglés *User Interface*), facilitar la explotación de dichos procesos, informar sobre la situación de los procesos de negocio e implementación de las reglas de validación de dicha interfaz. Es quien permite interactuar con la aplicación y se comunica con la capa de Lógica de Negocio.

Lógica de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

Almacenamiento: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. Existen muchos casos que usan bases de datos; sin embargo, también existen ocasiones en que se puede almacenar una cantidad de datos en un archivo local simple, en vez de una base de datos relacional. La regla común es: si solo se necesita almacenar datos, y recuperarlos por nombre, un sistema de archivos es suficiente. Pero si se requieren búsquedas sobre la información, entonces es necesaria una base de datos, especialmente si las búsquedas incluyen varios criterios.

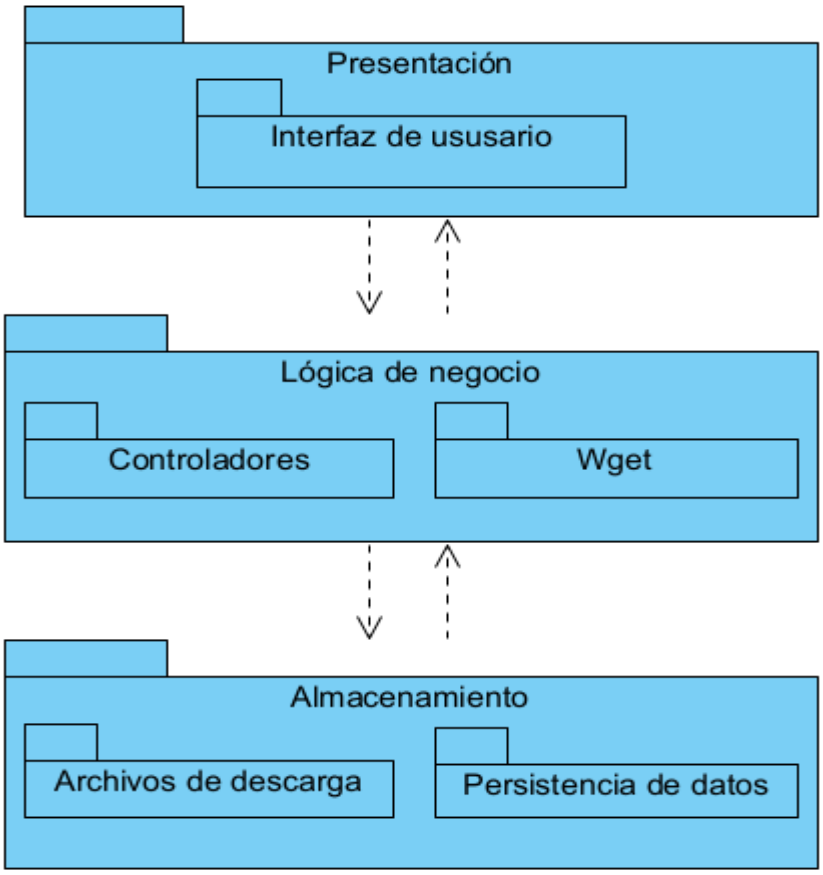

Ilustración 8: Diseño arquitectónico de la propuesta de solución

La persistencia de datos se trabaja a través de un sistema de archivos donde se especifica el nombre y la ruta de la descarga pues solo se necesita almacenar datos, y recuperarlos por nombre.

2.5.2 Diagrama de clases

Para modelar clases, incluidos sus atributos, operaciones, relaciones y asociaciones con otras clases, el UML proporciona un diagrama de clase, que aporta una visión estática o de estructura de un sistema mediante la representación gráfica, sin mostrar la naturaleza dinámica de las comunicaciones entre los objetos de las clases. Se puede decir que existen tres perspectivas diferentes desde las cuales se pueden utilizar los diagramas de clase (Peñalvo, 2015).

• **Conceptual:** El diagrama de clase representa los conceptos en el dominio del problema que se está estudiando. Este modelo debe crearse con la mayor independencia posible de la implementación final del sistema.

• **Especificación:** El diagrama de clase refleja las interfaces de las clases, pero no su implementación. Aquí las clases aparecen más cercanas a los tipos de datos, ya que un tipo representa una interfaz que puede tener muchas implementaciones diferentes.

• **Implementación:** Esta vista representa las clases tal cual aparecen en el entorno de implementación

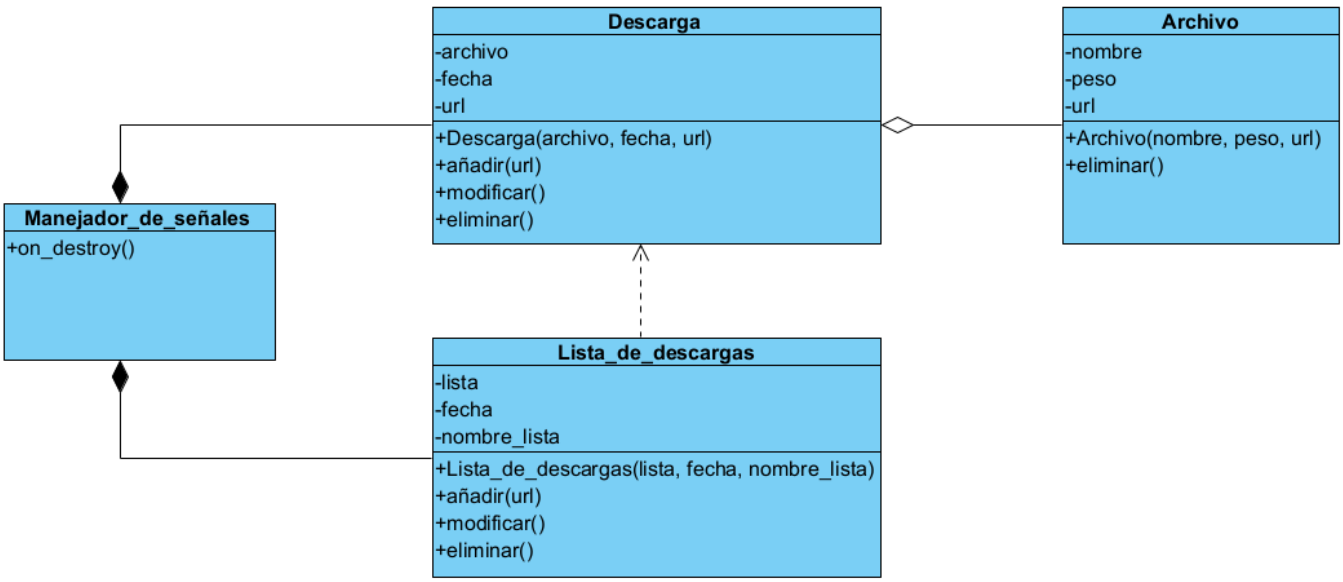

Ilustración 9: Diagrama de clases

Descarga: Representación de las tareas a descargar o descargados

Archivo: Representación del fichero guardado en disco

Lista _ de_ descargas: Lista de descargas a realizar

Manejo_ de _ señales: Se encarga de procesar todas las señales que envía la interfaz grafica

2.5.3 Patrones de diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico y entre “fuerzas” que afectan la manera en la que se aplica y en la que se utiliza dicho patrón.

El objetivo de cada patrón de diseño es proporcionar una descripción que permita a un diseñador determinar (Pressman, 2010):

- si el patrón es aplicable al trabajo en cuestión
- se puede volverse a usar (con lo que se ahorra tiempo de diseño)
- si sirve como guía para desarrollar un patrón distinto en funciones o estructura.

Patrones GRASP

Los Patrones Generales de Software para Asignar Responsabilidad (GRASP, por sus siglas en ingles), son una serie de patrones que describen los principios fundamentales de la asignación de responsabilidades a objetos y son considerados una serie de buenas prácticas en el diseño de software (Larman, 1999).

Experto: es un patrón que suele utilizarse en el diseño orientado a objetos. Este patrón sugiere asignar responsabilidad al objeto que posea la información necesaria para desempeñarla. Con la utilización se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener (Larman, 1999).. Este patrón se evidencia en la clase descarga pues contiene toda la información que necesita para realizar todas sus funciones.

- **Bajo acoplamiento:** el objetivo de este patrón es lograr la menor dependencia entre clases, de manera que las modificaciones realizadas en algunas de ellas afecten lo menos posible el funcionamiento del sistema. Soporta un diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. Este patrón se evidencia en el código pues se mantiene un numero bajo de relaciones entre clases, generalmente una clase está relacionada con un máximo de dos clases.

Conclusiones del capítulo

El análisis y diseño permitió extraer mediante la técnica de entrevista y tormenta de ideas 5 requisitos funcionales y 4 requisitos no funcionales lo cual permitió definir con mayor claridad las características de la propuesta de solución. Se realizó la representación de la propuesta de solución y las descripciones de las historias de usuario además de la realización del diagrama de clases, lo que permitió tener una guía de la situación real de la propuesta de solución. El uso de una arquitectura N-capas en su variante 3-capas y el empleo de patrones de diseño, garantizan obtener una solución de software con poca dependencia entre clases, flexible al mantenimiento y a la aceptación de cambios

Capítulo 3: Implementación y evaluación de la herramienta informática para la gestión de descargas en GNU/Linux Nova

Introducción

En el presente capítulo se presenta la especificación de los estándares de codificación utilizados durante la implementación de la solución y se abordan los temas relacionados con el flujo de trabajo correspondiente a las pruebas. Además, se especifican las pruebas que se le realizaron al Gestor de Descargas, para validar una correcta implementación de los requisitos de la herramienta y ofrecer un producto de calidad.

3.2 Implementación

El objetivo de esta disciplina es construir el sistema informático, a partir de los resultados del Análisis y Diseño (Rodríguez, 2015). La etapa de implementación del desarrollo de software es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre implica los procesos de diseño y programación de software, pero, si se utiliza un enfoque evolutivo de desarrollo, también puede implicar un refinamiento de la especificación del software (Sommerville, 2011).

3.2.1 Estándares de codificación

Las guías de código (también llamadas estándares de código *o* estilos de programación) es el nombre que se le da al conjunto de normas usadas para escribir código fuente, estas son regularmente dependientes del lenguaje de programación que se haya elegido. Un buen estilo para programar debe tener una estructura de código fácil de entender no solo para sí mismo sino también para otra gente y aportar eficiencia al proceso de desarrollo, logrando que los programas sean más robustos y comprensibles (codigofacilito, 2017)..

Indentación

- Usa cuatro espacios por cada nivel de codificación.
- Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).

- El método de indotación más popular en Python es con espacios. El segundo más popular con tabulaciones, sin mezclar unos con otros. Para proyectos nuevos, únicamente espacios son preferibles y recomendado antes que tabulaciones.

Máxima longitud de las líneas

- Las líneas deben estar limitadas a un máximo de 79 caracteres.
- Utilizar la continuación implícita para cortar las líneas largas dentro de paréntesis, corchetes
- llaves.
- En cualquier circunstancia se puede utilizar el caracter “\” para cortar líneas largas.

Líneas en blanco

Se utiliza dos líneas en blanco para separar funciones de alto nivel y definiciones de clase

Definiciones de métodos dentro de una clase son separadas por una línea en blanco

Importación

- Las importaciones deben estar en líneas separadas
- Las importaciones siempre se colocan al comienzo del archivo, simplemente luego de cualquier comentario o documentación del módulo, y antes de globales y constantes.
- Deben quedar agrupadas de la siguiente forma:
    1. Importaciones de la librería estándar.
    2. Importaciones terceras relacionadas.
    3. Importaciones locales de la aplicación/librerías.
- Cada grupo de importaciones debe de estar separado por una línea en blanco

Espacios en blanco en expresiones y sentencias

- Evitar usar espacios en blanco en las siguientes situaciones:
    1. Inmediatamente dentro de paréntesis, corchetes o llaves.

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

3. Inmediatamente antes de un corchete que empieza una indexación.

4. Más de un espacio alrededor de un operador de asignación u otro para alinearlo con otro.

□□Deben rodearse de espacios en blanco los siguientes operadores:

1. Asignación (“=”).

2. Asignación de aumentación (“+=”, “-=”, “*=”, “/=”).

3. Comparación (“==”, “”, “>=”, “<=”, ¡“! =”, “<>”, “in”, “*not in*”, “*is not*”).

4. Expresiones lógicas

□ Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad.

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

Comentarios

□ Los comentarios deben ser oraciones completas.

□ Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula.

□ Si un comentario es corto el punto final puede omitirse.

□ Los comentarios de una línea para aclaraciones del código aparecerán seguidos de los caracteres “#” y deben ubicarse en la misma línea que se desea comentar.

□ Los comentarios de varias líneas para organización del código aparecerán dentro de los caracteres “‘…’”.

Convecciones de nombramiento

- Nunca se deben utilizar como simples caracteres para nombres de variables los caracteres ele minúscula “l”, o mayúscula “O”, ele mayúscula “L” ya que en algunas fuentes son indistinguibles de los números uno (1) y cero (0). • Los módulos deben tener un nombre corto y en minúscula.

- Los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúscula).
- Los nombres de las funciones deben estar escritos en minúscula separando las palabras con un guion bajo (_).
- Las constantes deben quedar escritas con letras mayúsculas separando las palabras con un guion bajo (_).

La siguiente figura representa un ejemplo donde se muestra la aplicación de algunos estándares decodificación.

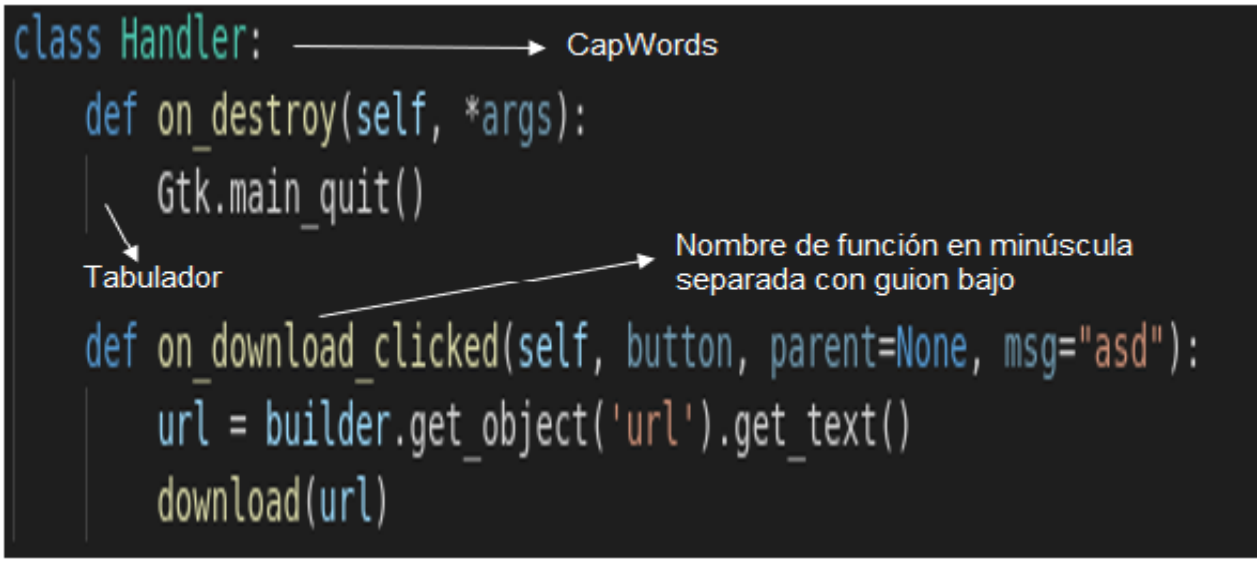

Ilustración 10: Aplicación de los estándares de codificación

3.3 Diagrama de despliegue

El diagrama de despliegue es una estructura que muestra la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos de despliegue. En él están representados nodos que corresponden con los dispositivos de hardware o con algún entorno de ejecución de software.

Estos nodos pueden ser conectados a través de vías de comunicación para crear sistemas en red de complejidad arbitraria (Larman, 2010). En la figura siguiente se presenta el diagrama de despliegue elaborado para la propuesta de solución:

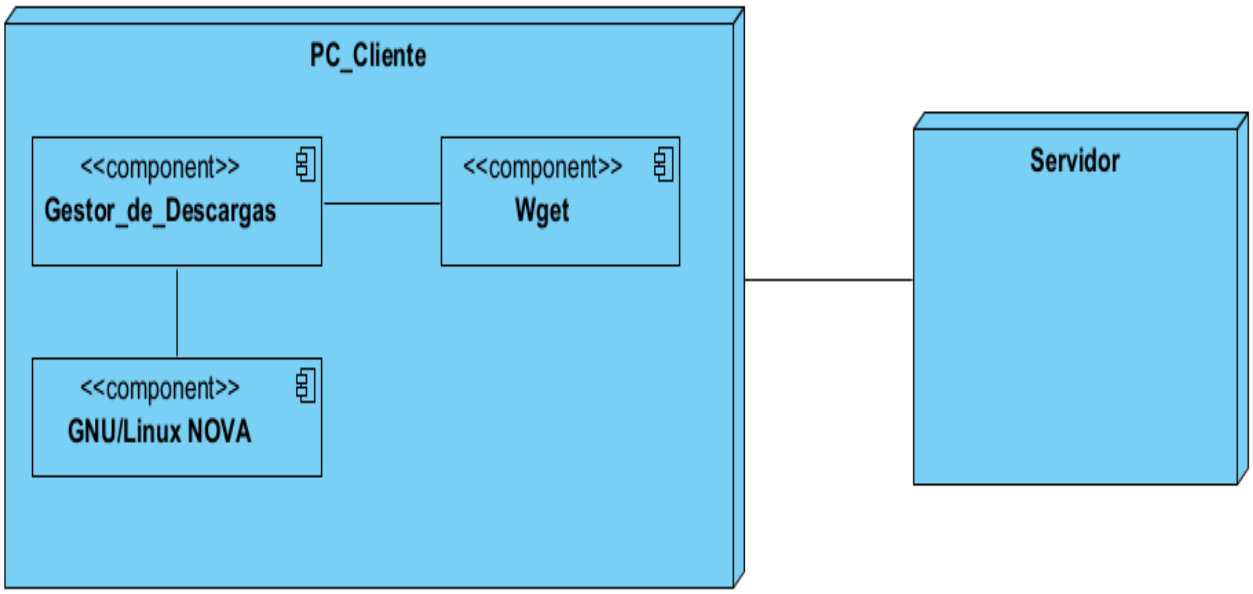

Ilustración 11: Diagrama de despliegue

A continuación, se describen los nodos, componentes y protocolos de comunicación representados en el diagrama anterior.

Descripción de los nodos:

PC Cliente: estación de trabajo desde la cual se gestionan las descargas.

Servidor: dispositivo informático que almacena, distribuye y suministra recursos informáticos.

Descripción de los componentes:

Nova Escritorio: distribución cubana de GNU/Linux Nova, que se encuentra como sistema operativo en las estaciones de trabajo (computadoras).

Wget: herramienta libre que mediante líneas de comandos permite la descarga de contenidos.

Gestor_ de_ Descargas: programa diseñado para realizar descargas de archivos en Internet.

3.4 Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia, el desempeño de un software, involucra las operaciones del sistema bajo condiciones controladas y evalúa los resultados. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad (Pressman, 2010).El principal objetivo de estas pruebas es diseñar pruebas que detecten la mayor cantidad de errores posibles y de la manera más óptima.

Pruebas unitarias

Las pruebas unitarias tienen como objetivo, permitirle verificar al desarrollador que los componentes unitarios estén codificados bajo condiciones de robustez, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Para que una prueba de unidad sea buena debe ser independiente, completa y reutilizable donde no se requiera de una intervención manual (GRAHAM, 2008). Estas pruebas deben ser realizados por el desarrollador ya que es recomendable conocer el código del programa en profundidad analizando el código para comprobar que cumple con las especificaciones requeridas.

Las pruebas unitarias se realizaron a través del método de prueba caja blanca y la técnica del camino básico.

Método de prueba: Caja blanca

Las pruebas de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba (Pressman,2010). Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:

1) Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.

2) Revisen todas las decisiones lógicas en sus lados verdadero y falso.

3) Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas.

4) Revisen estructuras de datos internas para garantizar su validez.

Técnica de prueba: Camino básico

La técnica del Camino básico permite obtener una medida de la complejidad lógica de la codificación de software y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución independiente en un componente o programa. Un camino o ruta es una vía por la cual procede la ejecución a través de una función desde su inicio hasta el fin (PRESSMAN, 2010).

Resultados de la prueba unitarias:

```
1  def download(url):
2      if is_url_valid(url):
3          downloads_dir = GLib.get_user_special_dir(GLib.UserDirectory.DIRECTORY_DOWNLOAD)
4          filename = wget.download(url, out=downloads_dir, bar=bar_progress)

5          subprocess.call(['notify-send', 'Descargas NOVA ', 'La descarga ha finalizado.'])
6          tview = builder.get_object("liststore1")
7          a = [1, filename.split('/')[len(filename.split('/'))-1], "100%", "2 Mb/s", "0:0:0",filename]
8          tview.append(a)
       else:
9          print("URL no valida")
```

Ilustración 12: Código de implementación para el método del camino básico

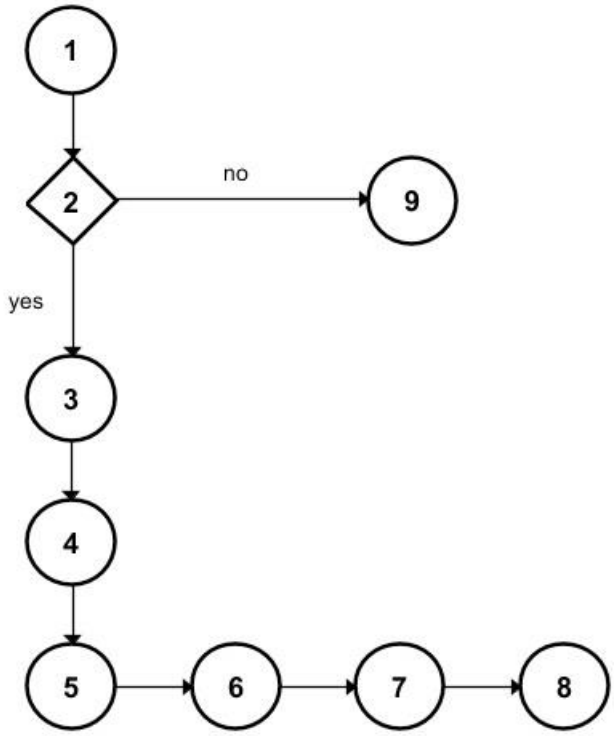

Ilustración 13: Grafo de flujo

A partir del grafo obtenido con 9 nodos y 9 aristas se calcula la complejidad ciclomática (V(G)). La complejidad ciclomática se basa en el recuento del número de caminos lógicos individuales contenidos en un programa. Para hallar la complejidad ciclomática, el programa se representa como un grafo, y cada instrucción que contiene, un nodo del grafo. Las posibles vías de ejecución a partir de una instrucción (nodo) se representan en el grafo como aristas.

V(G) = cantidad_aristas – cantidad_nodos + 2

V(G) = 9 – 9 + 2 = 2

Una ruta independiente es cualquier trayecto del programa que ingrese al menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición (Pressman, 2006). La cantidad de rutas independientes se establecen por la complejidad ciclomática; fueron identificadas 2 rutas, tal y como se muestra en la Tabla 11:

Tabla 11: Listado de caminos independientes.

No. Ruta	**Caminos**
1	1-2-9
2	1-2-3-4-5-6-7-8

El valor de V(G) ofrece además un límite superior del número de pruebas que debe diseñarse y ejecutarse para garantizar la cobertura de todas las instrucciones (Pressman, 2010). A continuación, se diseñan casos de pruebas para ser aplicados a cada ruta independiente:

Tabla 12: Caso de prueba unitaria. Ruta 1.

Caso de Prueba de Unidad	
No. Ruta: 1	**Ruta:** 1-2-9
Descripción de la prueba: Descargar el archivo que indica la URL.	
Entrada: Se recibe como parámetro una URL.	
Resultado esperado: Devolver una alerta. “URL no válida”	
Evaluación de la prueba: Satisfactorio. Se obtiene la alerta “URL no válida”.	

Tabla 13: Caso de prueba unitaria. Ruta 2.

Caso de Prueba de Unidad	
No. Ruta: 2	**Ruta:** 1-2-3-4-5-6-7-8
Descripción de la prueba: Descargar el archivo que indica la URL.	
Entrada: Se recibe como parámetro una URL.	
Resultado esperado: Se descarga el archivo de la dirección URL.	
Evaluación de la prueba: Satisfactorio. Se obtiene el archivo indicado en la URL y se muestra la alerta.	

Pruebas de integración

Una vez que todos los módulos han sido probados se suele pensar que si todos ellos funcionan por separado tienen que funcionar juntos, pero lamentablemente, aunque una aplicación funcione por sí sola,

nada garantiza que funcione junto con el resto de las aplicaciones que componen el ecosistema. En la realidad los datos pueden perderse a través de una interfaz; un componente puede tener un inadvertido efecto adverso sobre otro, las subfunciones, cuando se combinan, pueden no producir la función principal deseada, las estructuras de datos globales pueden presentar problemas (Pressman, 2010).

Teniendo en cuenta que la herramienta desarrollada debe integrase como aplicación de sistema. Se emplea una estrategia de integración ascendente, donde los componentes se integran de abajo hacia arriba. En esta prueba de integración ascendente se realiza la prueba de regresión que permite ejecutar nuevamente el mismo subconjunto de pruebas que ya se ha aplicado para asegurar que los cambios no han propagado efectos colaterales indeseables.

Resultados de la prueba de integración:

Con la realización de esta prueba para la integración de la aplicación con el sistema se pudo identificar una falla en la comunicación con Wget pues este no recibía correctamente los parámetros. Luego de solucionar esta no conformidad, la herramienta para la gestión de descargas para Nova se integró correctamente con el sistema operativo.

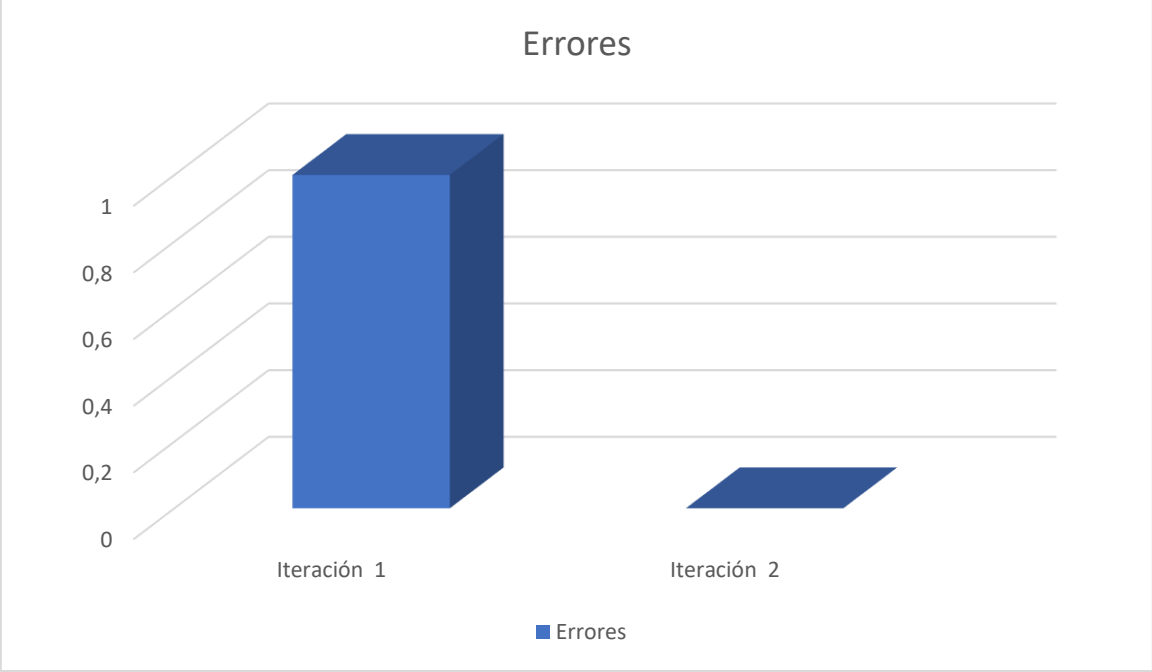

Ilustración 14: Gráfica resultados prueba de integración

Pruebas de aceptación

Visualmente resulta poco probable que un desarrollador de software prevea como utilizara el cliente final el programa. Las instrucciones para usarlo pueden malinterpretarse; regularmente pueden usarse combinaciones extrañas de datos; la salida que parecía clara a quien realizó la prueba puede ser ininteligible para un usuario. Debido a esto cuando se construye un software a la medida para un cliente, se realiza una serie de pruebas de aceptación a fin de permitir al mismo, validar todos los requerimientos y se centran en las características y funcionalidades generales que son visibles y revisables para garantizar que el sistema cumpla con lo que el cliente espera de él. Las pruebas de aceptación se derivan de las Historias de usuario que se han implementado como parte de la liberación del software (Pressman, 2010).

A continuación, se muestran los casos de prueba a realizar de algunas Historias de Usuario (HU).

Tabla 14: Caso de prueba de aceptación para la historia de usuario Descargar archivo

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Descargar archivo	
Descripción: Prueba realizada a la funcionalidad: Descargar archivo.	
Condiciones de ejecución: La computadora debe tener acceso mediante red al servidor donde se pretende descargar.	
Pasos de ejecución: **1.** El usuario introduce la url. **2.** Presiona el botón descargar. **3.** Comienza la descarga.	
Resultados esperados: Satisfactorio	

Tabla 15: Caso de prueba de aceptación para la historia de usuario Eliminar archivo

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Eliminar archivo	
Descripción: Prueba realizada a la funcionalidad: Eliminar archivo.	
Condiciones de ejecución: El archivo a eliminar debe estar seleccionado	

Pasos de ejecución: **1.** El usuario selecciona el archivo a eliminar. **2.** Presiona el botón eliminar. **3.** Se elimina el archivo.
Resultados esperados: Satisfactorio

Como resultado de las pruebas se obtuvo una evaluación satisfactoria por parte del cliente. Lo que representa el cumplimiento de los 9 requisitos solicitados y los objetivos propuestos para esta prueba.

Conclusiones del capítulo

Con el uso de los estándares de codificación se logró obtener claridad y organización en el código fuente de la solución. La aplicación de las pruebas unitarias, de integración y de aceptación permitieron detectar errores que afectaban el funcionamiento del software, lo que posibilitó corregirlos a tiempo para que el mismo cumpliese con los requisitos especificados.

CONCLUSIONES FINALES

- El análisis de los referentes teóricos y de las herramientas informáticas para la gestión de descargas estudiadas evidenció la necesidad de desarrollar un gestor de descargas para en la distribución cubana de GNU/Linux Nova
- La correcta selección de la metodología, herramientas, lenguajes y tecnologías permitió la implementación del gestor de descargas para GNU/Linux Nova.
- Se realizó el análisis, diseño e implementación de la herramienta para la gestión de descargas en la Distribución Cubana de GNU/Linux Nova, obteniendo así una herramienta que cumple con las necesidades del cliente.
- La aplicación de las pruebas de software permitió evaluar la solución desarrollada, dando la posibilidad de realizar el proceso de entrega del software a consideración del cliente.

RECOMENDACIONES

- Integrar la herramienta desarrollada a futuras versiones de Nova.

Bibliografía

Acosta, E., Álvarez, J. A., & Gordillo, A. (2006). Arquitecturas en n-Capas: Un Sistema Adaptivo. *Polibits*, *34*. https://www.redalyc.org/articulo.oa?id=402640447007

Avison, D. and G. Fitzgerald, (1995). Information Systems Development: Methodologies, Techniques, and Tools. McGraw-Hill.

Agudo, S. (2016, 11 enero). *uGet, el gestor de descargas para Ubuntu que no te puedes perder*. Ubunlog. Recuperado 15 de septiembre de 2021, de https://ubunlog.com/uget-el-gestor-de-descargas-para-ubuntu-que-no-te-puedes-perder/

Equipo editorial, Etecé. (2021, 6 agosto). *URL - Concepto, usos, partes y características*. Concepto. https://concepto.de/url/

Equipo editorial Etecé. (2017, 9 enero). ¿Qué es una guía de código? CódigoFacilito. https://codigofacilito.com/articulos/guia_codigo

Free Download Manager features. (s. f.). Free Download Manager. Recuperado 15 de septiembre de 2021, de https://www.freedownloadmanager.org/es/features.htm

Gestores de descarga: SoftwareUsco. (2018, 4 febrero). SoftwareUsco. Recuperado 20 de septiembre de 2021, de https://compusoftwareusco.webnode.com.co/novedades/software/gestores-de-descarga/

Guerra, C. A. (2018). *Obtención de Requerimientos. Técnicas y Estrategia*. SG Buzz. Recuperado 1 de octubre de 2021, de https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia

Glade - A User Interface Designer. (s.f.). Obtenido de Glade - A User Interface Designer: https://glade.gnome.org/

Hernández Mendoza, Y., Martin Jaime, E. M., & Martínez González, M. (2013). SERVICIO DE DESCARGA CENTRALIZADA PARA UNA RED UNIVERSITARIA. *3 Ciencias*. Published. https://www.3ciencias.com/

Internet Download Manager: The fastest download accelerator. (s. f.). Internet Download Manager. Recuperado 15 de septiembre de 2021, de https://www.internetdownloadmanager.com/#features

Introducción — documentación de Programación de Interfaces Gráficas de Usuario con GTK+ 3 - 1. (s. f.). Documentación de Programación de Interfaces Gráficas de Usuario Con GTK+. Recuperado 11 de septiembre de 2021, de https://programacion-de-interfases-graficas-de-usuario-con-gtk-3.readthedocs.io/001-intro.html

Iglesias, M. (2021, 14 septiembre). *Descarga más rápido de Internet con estos programas*. ADSLZone. https://www.adslzone.net/listas/mejores-programas/mejores-gestores-descargas/

Morales Jaramillo, C. O. (2015, enero). *Compilación unidad temática: Sistema Operativo*.Universidad del Amazonia.http://www.udla.edu.co/documentos/docs/Programas%20Academicos/Tecnologia%20en%20Informatica%20y%20sistemas/Compilados/Compilado%20Sistemas%20Operativos.pdf

Larman, C. (1999). UML Y PATRON. Cámara Nacional de la Industria Editorial Mexicana Num. 1524
https://upload.wikimedia.org/wikipedia/commons/a/a8/UML_y_Patrones__Larman_2_Edicion.pdf

Llorente, C. d. l. T. (2010). Guía de arquitectura N-capas.

Pérez, O. (2017). *Técnicas de obtención de requisitos*.

Pressman, R. S. (2010). *Ingeniería De Software Un enfoque práctico* (7.ª ed.). MCGRAW HILL EDDUCATION.

Qué es el lenguaje unificado de modelado (UML). (2017, 10 mayo). Lucidchart. Recuperado 20 de agosto de 2021, de https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml

UbuntuDocumentation. (s.f.). Obtenido de https://help.ubuntu.com/community/RootSudo

Rodríguez, T. S. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI.* Universidad de las Ciencias Informáticas.

Rossum, V. G & Python Development Team. (2018). *Python Tutorial: Release 3.6.4*. 12th Media Services.

Sommerville, I. (2011). *Ingeniería De Software* (9.ª ed.). Pearson Education.

uGet Download Manager. (s. f.). *uGet Features ·*. Recuperado 15 de septiembre de 2021, de https://ugetdm.com/features/.

Velasco, R. (2021, 25 octubre). *Mejores gestores de descarga*. SoftZone. https://www.softzone.es/programas/utilidades/mejores-programas-descargar-archivos/

Xtreme Download Manager | XDMAN | XDM Home. (s. f.). Xtreme Download Manager. Recuperado 15 de septiembre de 2021, de https://xtremedownloadmanager.com/#features

ANEXOS

Anexo 1: Entrevista realizada a Ing. Enmanuel Cristian de la Cruz Mora e Ing. Yileni Echavarría González

Datos generales del entrevistado:

Nombre y apellidos: _____

Cargo: _____

Categoría científica: _____

Años de experiencia: _____

Preguntas:

¿Como se realizan las descargas actualmente en GNU/LINUX NOVA?

¿Qué problemas trae consigo la forma actual de realizar descargar en GNU/LINUX NOVA?

¿Considera necesario la realización de un gestor de descargas propio para NOVA?

¿Qué características considera fundamentales en un gestor de descargas?

Anexo 2: Guía de observación para verificar la gestión de descargas en aplicaciones informáticas

Observadora: Liz Claudia Reyes Peñate

Objetivo: Identificar los elementos fundamentales para la gestión de descargas en herramientas informáticas.

1. Datos de las herramientas informáticas que permiten la gestión de descargas:
    - Nombre de las aplicaciones
    - Tipo de licencia
    - Compatibilidad con Linux
2. Características de las herramientas informáticas que permiten la gestión de descargas:
    - ¿Cómo son las herramientas informáticas que permiten la gestión de descargas?
    - ¿Cuáles son los pasos a seguir para instalar estas herramientas?
3. Impacto social de las herramientas informáticas que permiten la gestión de descargas.

Anexo 3: Interfaz de usuario