Facultad 3

“Herramienta web para la administración remota de repositorios GNU/Linux Nova”

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

Autor: Elena Castrillo Hernández

Tutor: Ing. Javier Piñeiro Cárdenas

Ing. Dargel Veloz Morales

La Habana, junio 2020

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Elena Castrillo Hernández, con carné de identidad 97012708198 soy la autora principal del trabajo titulado Herramienta web para la administración remota de repositorios GNU/Linux Novay 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 firmo la presente a los ____ días del mes de _____ del año _____.

Elena Castrillo Hernández

Autora

Ing. Javier Piñeiro Cárdenas

Tutor

Ing. Dargel Veloz Morales

Tutor

Datos de contacto

Ing. Javier Piñeiro Cárdenas

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo: jpineiro@uci.cu

Ing. Dargel Veloz Morales

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo: dveloz@uci.cu

Agradecimientos:

Dedicatoria:

A mis padres.

Resumen

Cuba se encuentra en un proceso de migración a software libre donde la Distribución Cubana GNU/Linux Nova, es el sistema operativo seleccionado para ser desplegado en los Organismos de Administración Central del Estado. Este sistema cuenta con repositorios de software que representan la vía oficial para que los usuarios puedan instalar un software confiable y recibir actualizaciones. Los repositorios constituyen el punto central del uso y desarrollo de las distribuciones de GNU/Linux, por lo cual la administración de los mismos cumple un factor importante dentro de la comunidad de usuarios que lo utilizan. El objetivo del presente trabajo de diploma es desarrollar una herramienta web que permita contribuir a la eficiencia en la administración remota de repositorios de la Distribución Cubana GNU/Linux Nova, haciendo uso de Aptly, una herramienta que brinda múltiples funcionalidades como la administración de repositorios, la creación de espejos de repositorios remotos ypermite filtrar paquetes con sus dependencias. Para guiar el proceso de construcción de la propuesta de solución se utilizó la metodología de desarrollo de software Variación de AUP para la UCI y en la implementación se utilizaron tecnologías y herramientas de software libre. La evaluación de la propuesta de solución se realizó a partir de la aplicación de técnicas y pruebas que garantizan el correcto funcionamiento de la aplicación y demostraron la satisfacción del cliente hacia el sistema desarrollado. Al culminar la presente investigación se obtuvo una herramienta web que permite la administración de los repositorios de la Distribución Cubana GNU/Linux Nova.

Palabras claves: administración de repositorios, Aptly, GNU/Linux, paquetes.

Índice

Índice de figuras

Índice de tablas

Introducción

El proceso de utilización ordenado y masivo de las Tecnologías de Información y Comunicación (TIC) en Cuba ha tenido un gran auge, con el objetivo de satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad. Dicho 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.

En los últimos años Cuba ha venido realizando una gran labor en el campo de la informatización de la sociedad. La Universidad de las Ciencias Informáticas (UCI), cuenta con un Centro de Soluciones Libres (CESOL) donde se desarrolló la Distribución Cubana de GNU/Linux Nova, que tiene como objetivo la migración al código abierto y el software libre en Cuba para garantizar la independencia tecnológica.

Cada una de las distribuciones GNU/Linux dispone de sus propios repositorios en los que se encuentran nuevas aplicaciones para satisfacer las necesidades de los usuarios, se pueden buscar actualizaciones, obtener correcciones de seguridad y de errores(UBUNTU, 2017).Los repositorios de dichas distribuciones contienen un conjunto de paquetes binarios o de código fuente organizados en una estructura especial, con varios archivos para estandarizar la infraestructura(GNU, 2018).

Los repositorios de la Distribución Cubana GNU/Linux Nova juegan un papel primordial en el ciclo de desarrollo y mantenimiento de sistemas operativos, pues constituyen una vía oficial y confiable para los usuarios. Están firmados digitalmente para que los sistemas instalados puedan identificar cuando el software es procedente de las fuentes oficiales. Aumentando así la confiabilidad del servicio prestado a los Órganos y Organismos de la Administración Central del Estado (OACE) (Fuentes, 2018).

Aparte de los repositorios propios de cada una de las distribuciones, también se pueden añadir y usar repositorios de terceros que contendrán versiones más actuales del software que se tiene instalado o programas que no han incluido los creadores(Carles, 2017). La Distribución Cubana de GNU/Linux Nova usa sus propios repositorios para instalar los programas necesarios para el usuario.

El uso de los repositorios genera ventajas como:

- La fácil recuperación de documentos, pues están almacenados en una misma base de datos.
- La garantía de preservación que supone el depósito en un repositorio es mucho mayor que la que ofrecen las páginas web personales.
- Mayor visibilidad y libre acceso a los contenidos.

- Se utilizan de forma intensiva en los sistemas operativos GNU/Linux, almacenando, en su mayoría, paquetes de software disponibles para su instalación mediante un gestor de paquetes (Dil, 2016).

Existen varias herramientas para la administración de repositorios, la mayoría no están diseñadas para la administración de grandes repositorios, no poseen un mecanismo de base datos que permita múltiples operaciones simultáneas, carecen de la documentación o los requerimientos de seguridad para ser usados (Fuentes, 2018).

Después de varios años de pruebas utilizando diferentes herramientas para realizar esta tarea, CESOL decidió que la más adecuada para la administración de repositorios de la Distribución Cubana GNU/Linux Nova era Aptly.

Aptlyfacilita la labor de administrar repositorios, permitiéndole duplicar repositorios remotos, gestionar repositorios de paquetes locales, tomar instantáneas, obtener nuevas versiones de paquetes junto con dependencias. Está disponible como herramienta de interfaz de línea de comando (CLI) y servicio HTTP REST (APTLY, 2018).

CESOL con el fin de mejorar el proceso de desarrollo de la Distribución Cubana GNU/Linux Nova, implementa un sistema de integración continua. La administración de los repositorios constituye una fase importante y a su vez compleja dentro de este proceso, ya que se cuenta con una estructura de repositorios donde cada paquete es colocado en el repositorio específico que le corresponde atendiendo a su estado, el cual puede ser:

- *Release*: versión funcional de un paquete pendiente a aprobación.
- *Snapshot:* estado momentáneo de un paquete para pruebas.
- Aceptado: paquete aceptado por el probador.
- Rechazado: paquete rechazado por el probador.

Para ello el departamento de Nova cuenta con un *docker*el cual contiene un servidor de Aptly, su interfaz de Programación de Aplicaciones (API) y otra desarrollada por los trabajadores del centro que complementa las funcionalidades que son necesarias para Nova pero que no trae la de Aptly. Sin embargo, no existe en el centro un modo de acceso a las funcionalidades de esas APIs y por tanto se encuentran en desuso.

Actualmente la administración de los repositorios de la Distribución Cubana GNU/Linux Nova se realiza utilizando esta herramienta desde la interfaz de línea de comando, accediendo remotamente al servidor donde se encuentra alojado el repositorio. Lo que trae consigo, demoras en el proceso de administración de los repositorios, pues se hace necesario consultar frecuentemente la documentación de Aptly, debido a la complejidad de la misma. Al realizar este proceso desde la consola no se provee a los administradores una guía o pasos implícitos, que sugieran el orden a

seguir, o simplemente delimiten el ámbito con que se relaciona la acción deseada en el repositorio, generando errores humanos y esto implica que pueden ser eliminados paquetes de código fuente y aplicaciones, incluso repositorios completos; dado que Aptly no hace rutinas de confirmación sobre las acciones irreversibles.

Además, las búsquedas se realizan a través de consultas que en ocasiones pueden ser extensas y complejas. También se debe tener en cuenta que los resultados de las mismas pueden ser muy amplios. Debido a esto, el usuario debe hacer uso de herramientas externas para filtrar los resultados obtenidos. Aptly no está diseñada específicamente para administrar los repositorios de la Distribución Cubana por lo cual el administrador debe tener conocimientos específicos de la estructura y la forma en que están organizados los repositorios de Nova.

Partiendo de esta situación problemática surge el siguiente **problema a resolver:** ¿Cómo contribuir a la eficiencia en la administración remota de repositorios de la Distribución Cubana GNU/Linux Nova?

Se define como **objeto de estudio:** la administración de repositorios de GNU/Linux enmarcado en el **campo de acción:** la administración de repositorios de la Distribución CubanaGNU/Linux Nova.

Por lo que se formula como **objetivo general:** desarrollar una herramienta web para la administración remota de repositorios de la Distribución CubanaGNU/Linux Nova.

Para darle cumplimiento al objetivo general planteado, el mismo se dividió en los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación sobre la administración de repositorios de la Distribución CubanaGNU/Linux Nova.
2. Diseñar una herramienta web para la administración de repositorios de la Distribución CubanaGNU/Linux Nova.
3. Implementar una herramienta web para la administración de repositorios de la Distribución CubanaGNU/Linux Nova.
4. Evaluar la herramienta web para la administración de repositorios de la Distribución CubanaGNU/Linux Nova.

Como **preguntas científicas** se plantean:

- ¿Cuáles son los fundamentos teóricos que sustentan la investigación sobre el proceso de desarrollo de la herramienta web para la administración de repositorios de Nova utilizando Aptly?
- ¿Cuáles son los aspectos a tener en cuenta para realizar el diseño de herramienta web para la administración de repositorios de Nova utilizando Aptly?

- ¿Qué componentes son necesarios implementar en el desarrollo de una herramienta web para la administración de repositorios de Nova utilizando Aptly?
- ¿Qué pruebas de software aplicar para la evaluación de la herramienta web para la administración de repositorios de Nova utilizando Aptly?

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

- Sistematización del marco teórico que sustentan la investigación, relacionados con la administración de repositorios Nova utilizando la herramienta Aptly.
- Análisis y diseño de una herramienta web para la administración de repositorios de Nova utilizando Aptly.
- Fundamentación de la metodología, las herramientas y las tecnologías para el desarrollo de la solución propuesta.
- Implementación de una herramienta web para la administración de repositorios de Nova utilizando Aptly.
- Validación, a través de una estrategia de pruebas, la herramienta web para la administración de repositorios de Nova utilizando Aptly.

En el desarrollo de la investigación se utilizan los siguientes **métodos científicos**:

Los métodos teóricos:

- **Inductivo-Deductivo:** el cual permite extraer patrones de diseños y elegir características para la herramienta web que se va desarrollar.
- **Analítico-sintético:** el cual posibilita el estudio y análisis de fuentes relevantes relacionadas con la administración de repositorios para obtener información y elementos fundamentales para la ejecución de la herramienta.
- **Modelación:** Es empleado en la representación mediante diagramas de las características, procesos y componentes de la solución propuesta, así como la relación existente entre ellos.

Los métodos empíricos:

- **Entrevistas:** se realiza una entrevista al especialista encargado de administrar repositorios en CESOL para conocer las características de este proceso en la actualidad.
- **Encuestas:** se realizaron para obtener información sobre las deficiencias existentes para la administración de repositorios Nova usando Aptly.
- **Observación:** este método se emplea en el análisis del proceso de administración de repositorios. Para su aplicación se utilizó la guía de observación para el proceso de

administración de repositorios Nova.

El presente documento se encuentra dividido en tres capítulos.

El capítulo 1, Fundamentación teórica: este capítulo describe los principales conceptos abordados en la investigación. Se estudiará las características y peculiaridades de sistemas homólogos a la propuesta de solución. Además, se define la metodología de desarrollo de software a emplear y se propondrán los lenguajes de programación, las herramientas y tecnologías a utilizar en el desarrollo de la herramienta web para la administración de los repositorios en la Distribución Cubana GNU/Linux Nova.

El capítulo 2, Descripción de la propuesta de solución: En el capítulo se presenta la propuesta de solución al problema de la investigación. Se definen los requisitos funcionales y no funcionales de la misma. Además, se describen las funcionalidades en las Historias de Usuario. Se explican los patrones arquitectónicos y de diseño que se emplean, y se define la arquitectura del software.

El capítulo 3, Implementación, pruebas y evaluación de la propuesta de solución: este capítulo se enfoca en la construcción del sistema a partir de los resultados del capítulo anterior. En él se elaboran los diagramas pertinentes, se definen los estándares de codificación a utilizar en la implementación de la solución y el código fuente de la misma. Además, se definen y realizanlas pruebas de software requeridas, con el objetivo de descubrir y corregir posibles errores, midiendo así la calidad de la propuesta de solución y constando que el resultado de la investigación satisface las necesidades del cliente.

Capítulo 1: Fundamentación teórica del proceso de administración de repositorios de la Distribución Cubana GNU/Linux Nova

En el presente capítulo se hace importante conocerlas definiciones relacionadas con el proceso de administración de repositorios, el estudio sobre dicho tema y describir la metodología de desarrollo de software a utilizar. También se define y se detallan características del lenguaje y la herramienta de modelado, así como las herramientas y tecnologías para la implementación de la propuesta de solución.

1.1 . Definición de conceptos

A continuación, se muestran un conjunto de conceptos que permiten la comprensión del tema de investigación.

Nova es una Distribución Cubana de GNU/Linux desarrollada en la Universidad de las Ciencias Informáticas. Provee un sistema cómodo, enfocado al usuario final, garantizando una interacción intuitiva que persigue minimizar el cambio brusco al que se enfrentan las personas familiarizadas con sistemas Microsoft Windows. Cuenta con 3 ramas: “Nova Escritorio”, “Nova Ligero” y “Nova Servidor”. Nova se desarrolla para responder a necesidades de varias instituciones del país y apoyar la migración a tecnologías de Software Libre y Código Abierto en Cuba, como parte del proceso de informatización de la sociedad(Humanos).

Dicha distribución cubana cuentas con sus propios**repositorios**que son sitios donde los paquetes son almacenados y mantenidos. Pueden ser locales o remotos, dependiendo de si se encuentran en el sistema de archivos de la máquina del usuario que los utiliza o en una localización remota accesible a través de la red, generalmente a través de los protocolos HTTP (Protocolo de transferencia de hipertexto) o FTP (Protocolo de transferencia de archivos). Cuentan con archivos índices o metadatos que permiten la rápida localización de un paquete en este sitio, así como la rápida obtención de la información del mismo sin tener que descargarlo o procesarlo(Fernández-Sanguino, 2016).

En términos informáticos un **paquete de software** consiste en un archivo comprimido utilizando un algoritmo de compresión conocido, que puede contener los elementos que integran una aplicación, así como los metadatos necesarios para su correcta identificación siguiendo un estándar definido(Fernández-Sanguino, 2016).

En las distribuciones de GNU/Linux los paquetes de software pueden ser de dos tipos: binarios o de código fuente.

Un **paquete de código fuente** contiene todos los elementos necesarios para la construcción de uno o varios paquetes binarios. Dentro de los elementos contenidos en este se encuentra el código fuente de la aplicación, las recetas necesarias para la compilación y construcción de los paquetes binarios.

Así como los recursos necesarios para la ejecución de las aplicaciones contenidas (como configuraciones, imágenes, base de datos(HERTZOG, 2015).

Los **paquetes binarios** son conjunto de ejecutables, bibliotecas, archivos de configuración, documentación y licencias, comprimidos que conforman una aplicación informática. Pueden ser instalados en un ordenador por un tipo especial de herramienta llamada gestor de paquetes. En las distribuciones basadas en Debian un paquete binario es una colección de archivos ejecutables (binarios compatibles con el cargador de arranque de Linux o scripts) y archivos complementarios necesarios para la ejecución de la aplicación(Fuentes, 2018).

Para la administración de los repositorios de la Distribución Cubana GNU/Linux Nova se consumirá servicios de una **API (Interfaz de programación de aplicaciones)** que es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.Permitenhacer uso de funciones ya existentes en otro software (o de la infraestructura ya existente en otras plataformas) para no estar reutilizando así código que se sabe que está probado y que funciona correctamente(Springer, 2015).

Nova cuenta con un **docker** que es una de las plataformas de creación y gestión de paquetes de elementos que permiten crear un entorno en el que las aplicaciones se ejecutan independientemente del sistema operativo, es una herramienta de empaquetado y entrega de software (Docker ecosystem-vulnerability analysis, 2018).

Se utiliza el formato de intercambio **JSON** *(JavaScript Object Notation)* es un formato de codificación de datos eficaz que permite intercambios rápidos de cantidades pequeñas de datos entre los exploradores de cliente (Serialización/Deserialización de objetos y transmisión de datos con JSON, 2016).

1.2. Caracterización de la administración remota de repositorios la Distribución Cubana GNU/Linux Nova

Los repositorios de la Distribución Cubana GNU/Linux Nova brindan actualmente miles de paquetes, en más de diez lenguajes de programación diferentes (Java, C, C++). Tienen la misma estructura que los repositorios de Debian y Ubuntu, pero a diferencia de estas distribuciones cuenta solamente con dos componentes:

- **principal:**en el que se almacena todo el software con soporte oficial de la distribución Nova. El filtro para la selección de los paquetes de este componente es generado a partir de las dependencias y las recomendaciones de los metapaquetes de los principales productos: Nova Escritorio, Nova Ligero y Nova Servidor.

- **extendido:** componente en el que se almacena todo el software compatible con la distribución, pero que no cuenta con soporte oficial por parte de Nova. En este se encuentran muchas aplicaciones de software provenientes de desarrollos comunitarios o con propósito específico(Fuentes, 2018).

A diferencia de las distribuciones de las que deriva, Nova no clasifica los paquetes debido a su origen, sino a su nivel de soporte. Por esta razón es común localizar en el mismo componente aplicaciones privativas y de licencias abiertas.

La administración de los repositorios de software de las distribuciones de GNU/Linux constituye una piedra angular en sus procesos de desarrollo. Pues estos constituyen el único método oficial para la publicación de software, actualizaciones y correcciones de seguridad para estos sistemas operativos(Fuentes, 2018).

En el proceso de administrar repositorios de la Distribución Cubana GNU/Linux Nova se gestionan paquetes; permitiendo al usuario mover, listar, filtrar y eliminar paquetes. También se muestran informaciones de las distribuciones y de los repositorios que contiene.

Para la gestión de repositorios en Nova se utiliza Aptly, el objetivo de esta herramienta es establecer la repetitividad y los cambios controlados en un entorno centrado en el paquete. Acertadamente permite arreglar un conjunto de paquetes en un repositorio, para que la instalación y actualización del paquete se vuelva determinista. Al mismo tiempo, acertadamente permite realizar un control y mantener los cambios detallados en los contenidos del repositorio para hacer la transición del entorno de su paquete a una nueva versión(APTLY, 2018).

Aptly tiene varias entidades centrales: *mirror* (espejos) del repositorio remoto, consiste en metadatos, lista de paquetes y archivos de paquetes de repositorios locales, se componen de metadatos, paquetes y archivos, los paquetes se pueden agregar y quitar fácilmente. Además, permite al usuario crear su propio repositorio remoto lo cual permite ahorrar el anchode banda si se tiene un gran número de servidores, así como que no dependen de los servicios remotos para funcionar correctamente con el fin de gestionar las piezas críticas de la infraestructura(APTLY, 2018).

Dentro de sus muchas funcionalidades resaltan:

- Creación de réplicas de repositorios remotos.
- Gestión de repositorios personales.
- Creación de copias de seguridad del estado del repositorio.
- Realizar consultas personalizadas de los paquetes.
- Publicar el repositorio en diferentes servicios de almacenamiento.
- Brinda una API pública para la gestión web del servicio e interacción remota con otros servicios de integración continua.

- Soporte para firma digital de los repositorios.

Una de las virtudes de Aptly que lo hacen superior a la hora de realizar las operaciones es que separa el proceso de gestión de los repositorios y réplicas de la publicación de estos. Esto posibilita que los mantenedores de paquetes puedan realizar las operaciones ordinarias sin afectar el servicio de repositorio, aumentando la eficiencia en el trabajo(APTLY, 2018).

1.3. Análisis de sistemas homólogos

El análisis de las particularidades de sistemas similares para administrar los repositorios de los sistemas GNU/Linux, permite conocer características que, por su relevancia, pueden ser fundamentales para la propuesta a desarrollar. En este epígrafe se observarán y analizarán criterios de los sistemas homólogos que son de gran importancia para la propuesta de solución pues pueden servir como objeto de estudio para realizar una selección acertada de estándares, herramientas y tecnologías a utilizar para su construcción. Estos sistemas son Launchpad, OpenSUSEBuildService y Wanna-build utilizados por UBUNTU, OpenSUSE y Debian respectivamente.

1.3.1. Launchpad

Es una plataforma de hospedaje y colaboración de proyectos de software. Brinda un conjunto de herramientas destinadas a ayudar en el desarrollo de estos proyectos. Con Launchpad es fácil compartir código, informar, seguir y resolver errores, ayudar con las traducciones de un proyecto a varios idiomas, y otros muchos más aspectos (Launchpad, 2017).

Este sistema es lanzado en el año 2004 aunque hasta el 2009 no fue liberado, actualmente está bajo la licencia pública GNU Affero General. La estrategia de soporte es web oficial por lo tanto el acceso a dicha plataforma es a través de internet y su dominio es Launchpad.net.

Funciones que realiza:

- permite buscar paquetes, ver la información de ellos y compilar.
- Permite hospedar por completo tu proyecto de software. En este lugar se tienen todas las herramientas necesarias para alojar el código fuente, seguimiento de errores, sugerencias para nuevas versiones, traducciones, así como preguntas y respuestas.
- Permite realizar el seguimiento de errores, compartir informes de errores, el estado o situación en la que se encuentra un determinado error, parches, así como comentarios para facilitar en todo momento la resolución de los mismos.
- Sugerencias: Esta opción de Launchpad, ofrece la posibilidad de que los usuarios de la aplicación, no solo los desarrolladores, puedan aportar sus propias ideas para mejorar las funcionalidades de la aplicación(Launchpad, 2017).

1.3.2. OpenSUSEBuildService

Es una plataforma de desarrollo completo y de código libre (open source) que proporciona la infraestructura para el desarrollo de futuras distribuciones basadas en openSUSE.Es la instancia pública del Open BuildService (OBS) utilizado para ofrecer paquetes de la misma fuente para Fedora, Debian, Ubuntu, SUSE Linux Enterprise y otras distribuciones(opensuse, 2017).

Este sistema tiene un lanzamiento estable en el año 2018, actualmente está bajo la licencia GNU GPL (Licencia pública general de GNU). La estrategia de soporte es web oficial por lo tanto el acceso a dicha plataforma es a través de internet y su dominio es build.opensuse.org.

Funciones que realiza:

- Esta instancia ofrece una interfaz especial de búsqueda de paquetes. Los usuarios de cualquier distribución pueden buscar allí paquetes integrados para su distribución.
- Ofrece a desarrolladores herramientas de compilación, distribución y publicación de proyectos tales como la creación de distribuciones LINUX basadas en openSUSE para diversas arquitecturas de hardware.
- Para los desarrolladores, es un lugar eficiente para crear grupos y trabajar juntos a través de su modelo de proyecto(opensuse, 2017).

1.3.3. Wanna-build

Es una herramienta que ayuda a coordinar la reconstrucción de paquetes a través de una base de datos que mantiene una lista de paquetes y su estado. Hay una base de datos central por arquitectura que almacena los estados, versiones, y alguna otra información de los paquetes(DEBIAN, 2019).

Este sistema está bajo licencia GLP (Licencia pública general de GNU). La estrategia de soporte es web oficial por lo tanto el acceso a dicha plataforma es a través de internet y su dominio es buildd.debian.org donde se encuentran los datos producidos por Wanna-Build.

El conjunto de programas *sbuild (buildd y sbuild*) se utilizan para construir paquetes binarios a partir de paquetes fuente. La base de datos wanna-build rastrea paquetes que requieren construcción. Los horarios de *buildd* funcionan a partir de la información que obtiene de esta y sbuild hace la construcción del paquete real(DEBIAN, 2019).

Una infraestructura Wanna-Build se compone de una base de datos wanna-build relacionada con un repositorio y uno o más servidores de construcción que se comunican con esta base de datos. La base de datos realiza un seguimiento del estado de la construcción de los paquetes de software de Debian desde el repositorio relacionado. Los servidores binarios pueden cargar paquetes binarios correctamente compilados en un repositorio(DEBIAN, 2019).

1.3.4. Resultados del estudio de homólogos.

A continuación, en la tabla 1 se muestra la comparación entre características que presentan estos sistemas homólogos:

Tabla 1. Comparación de sistemas homólogos. (Fuente: Elaboración propia).

Criterios a analizar	**Sistemas Homólogos**		
	Launchpad	**OpenSUSEBuildService**	**Wanna-build**
Madurez del sistema (más de 5 años)	Si	Si	-
Acceso web a través de internet	Si	Si	Si
Estrategia de soporte web oficial	Si	Si	Si
Licencia Pública	Si	Si	Si
Búsqueda de paquetes	Si	Si	Si
Mostrar información de los paquetes	Si	Si	Si
Acceso como administrador	No	No	No
Disponibilidad del código fuente	No	No	No

Después de analizar los sistemas homólogos en la tabla anterior se pudo observar que tanto Launchpad como OpenSUSEBuildService y Wanna-buildcumplen con algunos criterios de selección. Pero estos sistemas no permiten acceder como administrador para realizar funciones claves. Tampoco se encuentra disponible el código fuente para desplegarlo de manera local o realizar modificaciones necesarias para que permita la administración de los repositorios Nova. Como estos sistemas no cumplen con los requerimientos necesarios para darle solución al problema de investigación se decidió desarrollar una nueva solución, aunque se tomarán en cuenta características y funcionalidades de los mismos.

1.4. Metodología de desarrollo de software

La Metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. En un proyecto de desarrollo de software la metodología ayuda a definir: ¿Quién debe hacer, ¿qué, ¿cuándo y cómo debe hacerlo? Es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito(Zúñiga, 2017).

1.4.1. Proceso Unificado Ágil variación para la UCI

La metodología de desarrollo de software que se utilizará es la variación de AUP para la UCI, es una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP("*Proceso Unificado Ágil*) y está definida por la universidad como el documento rector de la actividad productiva(Rodríguez, 2015).

Fases

Está formada por las fases Inicio, Ejecución y Cierre. Tienen como características:

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, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.

Cierre: En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto(Rodríguez, 2015).

Disciplinas

Para el ciclo de vida de los proyectos de la UCI se decidió contar con 8 disciplinas, pero a un nivel más atómico que el definido en AUP, ellas son: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas interna, Prueba de liberación Pruebas de Aceptación y Despliegue(Rodríguez, 2015).

Escenarios

La metodología AUP UCI tiene cuatro escenarios. Para modelar la propuesta de solución se tomará el escenario 4 debido a 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.

1.5. Lenguaje y herramienta para el modelado de la solución

Los lenguajes de modelado sonutilizados para definir un sistema de software, para detallar los artefactos en el sistema, documentar y construir. Las herramientas de modelado son utilizadas para representar mediante diagramas el proceso de desarrollo(Rodríguez, 2017).

En el modelado de la propuesta de solución se utilizará el lenguaje **UML** en su versión **2.0**.

Lenguaje Unificado para el modelado

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. Ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. UML cuenta con un conjunto de notaciones y diagramas para modelar sistemas orientados a objetos y describe la semántica esencial de lo que estos diagramas y símbolos significan (Rumbaugh, 2017).

En el modelado de la propuesta de solución se utilizará la herramienta **VisualParadigm** en su versión **8.0**.

Visual Paradigm

Visual Paradigmas una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad. Brinda la posibilidad de generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software, así como obtener diversos informes a partir de la información introducida en la herramienta (Domingo, 2018).

Visual Paradigm también ofrece (Targetware, 2016):

- Navegación intuitiva entre la escritura del código y su visualización.
- Potente generador de informes en formato PDF/HTML.
- Documentación automática.

- Ambiente visualmente superior de modelado.

1.6. Tecnologías de implementación

Para escoger las tecnologías adecuadas para implementar la propuesta de solución se tuvo en cuenta las características de Aptly, del API que se va a utilizar y el lenguaje en el cual esta implementado. A continuación, se muestran las tecnologías escogidas para implementar la propuesta de solución.

1.6.1. Lenguaje de programación

Es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente(Evaluación de los frameworks en el desarrollo de Aplicaciones Web con Python, 2016).Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana (Olarte, 2018).

En la implementación de la propuesta de solución se utilizará el lenguaje de programación **Python** en su versión **3.5**.

Python es un lenguaje orientado a objetos, potente, es flexible debido a su capacidad para utilizar componentes que fueron diseñados en otra programación. Puede admitir diferentes estilos de programación como estructural. Es de código abierto y como tal cualquiera puede contribuir a su desarrollo y divulgación(Python-The Fastest Growing Programming Language, 2017).

Las principales características que tiene este lenguaje son(lenguajesdeprogramacion.net, 2018):

- Lenguaje de programación multiparadigma (programación orientada a objetos, programación estructurada y programación funcional).
- Utiliza el tipo de dato dinámico para el manejo de memoria.
- Permite la resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa.
- Puede utilizarse como un lenguaje de extensión para módulos y aplicaciones que necesitan de una interfaz programable.

1.6.2. Entorno de desarrollo Integrado

Es un paquete de software que consolida las herramientas básicas necesarias para escribir y probar un software(Rouse, 2018).Es una aplicación destinada a brindar servicios integrales al desarrollador en su trabajo, o dicho más simple, es un programa que permite construir código de una forma más sencilla o didáctica(Blancarte, 2017).

En la implementación de la propuesta de solución se utilizaráel entorno de desarrollo integrado **PyCharm** en su versión**2019.3**.

PyCharm, es unentorno de desarrollo integradomultiplataforma utilizado en el ámbito de la programación. Viene con una consola de Python donde puede escribir los scripts a medida que los ejecuta. Adicionalmente, provee capacidades de alto rango para desarrolladores profesionales de web con el marco de trabajo Django (Naranjo, 2017).

Las principales características que ofrece son (Pythonízame, 2016):

- Editor inteligente.
- Depurador de código gráfico.
- Inspección de código.
- Integración de control de versiones.
- Ejecución de pruebas.
- Soporta los repositorios JIT, Mercurial, Subversion y Github.
- Permite los leguajes XML (Lenguaje de Marcado Extensible) y HTML (Lenguaje de Marcas de Hipertexto).
- Posee una terminal local.

1.6.3. Marcos de trabajo

Un marco de trabajo (framework) es una gran librería o conjunto de librerías donde además de facilitar funciones para su uso, dispone de una sintaxis o metalenguaje específico del marco de trabajo y una forma de organización de su código. Facilita el desarrollo de software y evita los detalles de bajo nivel (Gandarillas, 2017)

En la implementación de la propuesta de solución se utilizaráel marco de trabajo **Django** en su versión**2.0.2**.

Django es un framework web de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Se encarga de gran parte de las complicaciones del desarrollo web y es de código abierto. Está basado en el patrón arquitectónico Modelo-Vista-Plantilla. Apuesta por la sencillez, la rapidez y la reutilización de código. Viene con una base de datos SQLite(Rubio, 2017).

Algunas de las características que ofrece es(developer.mozilla.org, 2019):

- Puede funcionar con cualquier framework en el lado del cliente, y puede devolver contenido en casi cualquier formato.
- Proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando así errores.
- Django está escrito en Python, el cual se ejecuta en muchas plataformas.

1.6.4. Servidor de aplicaciones

Un servidor de aplicaciones es un programa encargado de realizar el despliegue de aplicaciones web. Es un servidor en una red distribuida que proporciona la lógica de negocio para un programa de aplicación(Otalora, 2018).

En la implementación de la propuesta de solución se utilizarán los servidores de aplicaciones **NGINX**en su versión **1.9.15** y **Gunicorn**en su versión **19.6.0**

NGINXes un software de servidor web de código abierto, multiplataforma, sirve como proxy inverso, balanceador de carga HTTP y proxy de correo electrónico y tiene facilidad de expansión y es de alto rendimiento. Puede ejecutarse en muchos sistemas operativos que soportan el sistema Unix(Research on Nginx Dynamic Load Balancing Algorithm, 2020).

Características (Nginx.org, 2017):

- El proxy inverso ayuda con el balanceo de carga distribuyendo las peticiones y almacenando en caché parte del contenido.
- Cuenta con caché para solicitudes HTTP.
- Facilita el uso de URL con posibilidad de usar expresiones regulares.
- Función de rastreo y seguimiento de los usuarios.
- Ofrece tolerancia a fallos.
- Soporta servidores virtuales.

Gunicorn es un ligero y rápido servidor WSGI (Interfaz de Puerta de enlace del Servidor Web), una especificación para la interfaz simple y universal entre servidores web y aplicaciones web o frameworks para Python(Gunicorn.org, 2017).

1.6.5. Servidor de Base de Datos

Un servidor de base de datos es un programa que provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor(Alegsa, 2019).

En la implementación de la propuesta de solución se utilizará el servidor de base de datos **PostgreSQL**.

PostgreSQLes un potente sistema de base de datos relacional de objetos de código abierto que usa y amplía el lenguaje SQL combinado con muchas características que almacenan y escalan de manera segura las cargas de trabajo de datos más complicadas (Postgresql: Extensión del Postgresql para el manejo de datos con frecuencias temporales., 2016)

Ventajas que ofrece dicho servidor de base de datos (postgresql.org, 2019):

- Se ha ganado una sólida reputación por su arquitectura comprobada, confiabilidad, integridad de datos, conjunto de características robustas, extensibilidad y la dedicación de la comunidad

de código abierto detrás del software para ofrecer soluciones innovadoras y de alto rendimiento.

- Se ejecuta en todos los principales sistemas operativos.
- Además de ser gratuito y de código abierto, PostgreSQL es altamente extensible.

1.6.6. Herramienta de control de versiones

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas más adelante(Chacon, 2015).

En la implementación de la propuesta de solución se utilizarála herramienta de control de versiones **Git**.

Git es una herramienta que realiza una función del control de versiones de código de forma distribuida, es muy potente y rápida, no depende de un repositorio central(Rubio, 2019). No se necesita compartir una misma red con los desarrolladores participantes en el proyecto. Git tiene copias locales del repositorio desde el cual se trabaja directamente(González, 2018).

1.7. Conclusiones del capítulo

En este capítulo se han abordado conceptos relacionados con la administración de repositorios lo que permitió entender mejor este proceso. La comparación de los sistemas homólogos analizados permitió demostrar la necesidad de desarrollar una aplicación web ya que las existentes no permiten la administración de repositorios Nova.

Se identificó 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 IDE PyCharm, como marco de trabajo Django, como servidor de aplicaciones NGINX yGunicorn, como servidor de base de datos PostgreSQL, como herramienta de control de versiones GIT. La herramienta de modelado que se utilizó fue Visual Paradigm y el lenguaje UML.

CAPÍTULO 2: Descripción de la propuesta de solución

El presente capítulo contiene la información relacionada con la descripción de la propuesta de solución tomando como punto de partida los elementos planteados por la metodología Variación de AUP para la UCI en el escenario 4. En él se evidencia la aplicación práctica del conocimiento científico de la ingeniería de software, se reflejan los requisitos funcionales y no funcionales que el sistema debe cumplir y las historias de usuarios para describir los mismos. Se describe la arquitectura a utilizar a través del diagrama de paquetes, también se muestra el diagrama de clases con los patrones de diseño utilizados y el diagrama de datos.

2.1. Aptly y los repositorios de Nova.

En el proceso de desarrollo de la Distribución Cubana GNU/Linux Nova, se implementa un sistema de integración continua donde el manejo de los repositorios, su estructura y organización tienen un papel importante. Los paquetes mantienen un flujo que comienza cuando el desarrollador ejecuta una orden de compilación, luego culminado este proceso el paquete puede ser colocado en dos repositorios Incoming o Unstable. A Incoming irán aquellos paquetes los cuales se le ha realizado un *release*, mientras que aUnstable irán los *snapshot*. Los paquetes en Unstable son los utilizados por los desarrolladores para realizar pruebas de un estado momentáneo de un paquete y luego continuar con el desarrollo.

Una vez concluido el desarrollo y las pruebas de un paquete por parte del desarrollador, este pasaría a Incoming, donde serían los probadores y el personal de calidad los encargados de aprobar el paquete. De ser así el mismo pasa al repositorio Accepted desde el cualpasa más a delante a los repositorios oficiales de la distribución. De ser rechazado el paquete se movería hacia Rejected para comenzar el ciclo nuevamente.

Cada distribución en desarrollo o mantenimiento de Nova debe contar con su repositorio Incoming, Accepted, Rejected, Unstable y cada paquete en el lugar que le corresponde, pues si un paquete que debe ir hacia Accepted de 2017 es pasado por error a Accepted 2019 podría ocasionar graves errores. Por este motivo con el paso del tiempo el número de repositorios que se manejan va en aumento y para facilitar su administración se utiliza Aptly.

El departamento de Nova cuenta con un *docker* el cual contiene un servidor de Aptly, que tiene su propia API y otra desarrollada por los trabajadores del centro que complementa las funcionalidades que son necesarias para Nova pero que no trae la de Aptly.

El API de Aptly permite configurar el puerto de acceso de la siguiente manera:

```
$ aptly api serve -listen=:8080
```

Figura 1.Ejecución de Aptly por el puerto 8080. (Fuente: Aptly.info)

El caso de Nova se utiliza para esto el puerto 10000, permitiendo muchas acciones sobre los paquetes y repositorios como por ejemplo listar o buscar los paquetes de un repositorio.

Mostrar paquetes/Buscar

`GET /api/repos/: name/packages`

Enumerar todos los paquetes en el repositorio local o realizar una búsqueda en el contenido del repositorio y devolver el resultado.

Parámetros de consulta:

Nombre	Parámetros
`q`	**Consulta de paquetes, si faltan se muestran o se devuelven todos los paquetes**
`withDeps`	**Establecer en `1` para incluir dependencias al evaluar la consulta del paquete**
`format`	**Formato de resultado, compacto de forma predeterminada (solo claves de paquete), `details` para devolver información completa sobre cada paquete (puede ser lento en repositorios grandes)**

Figura 2. Mostrar paquetes/Buscar. (Fuente: Aptly.info).

```
$ curl http://localhost:8080/api/repos/aptly-repo/packages
["Pi386 aptly 0.8 966561016b44ed80"]

$ curl http://localhost:8080/api/repos/aptly-repo/packages?q=aptly
["Pi386 aptly 0.8 966561016b44ed80"]

$ curl http://localhost:8080/api/repos/aptly-repo/packages?format=details
[{"Architecture":"i386","Description":" Debian repository management tool\n","Filename":"aptly
_0.8_i386.deb","FilesHash":"966561016b44ed80","Homepage":"http://www.aptly.info/","Installed-S
ize":"11084","Key":"Pi386 aptly 0.8 966561016b44ed80","License":"MIT","MD5sum":"b9be9ed873f1a0
5da103406cc0a6b9d1","Maintainer":"Andrey Smirnov \u003cme@smira.ru\u003e","Package":"aptly","P
riority":"extra","Recommends":"bzip2","SHA1":" 257ab261adcf5dd5bda800976ae606fedb882679","SHA2
56":" 6342804c7f6bd8cb004ca9d19a7e27f492b7e07843b935a4f96a07a254ae6312","Section":"default","S
hortKey":"Pi386 aptly 0.8","Size":"3510032","Vendor":"Andrey Smirnov \u003cme@smira.ru\u003
e","Version":"0.8"}]
```

Figura 3. Ejemplo de Mostrar paquetes/Buscar.(Fuente: Aptly.info).

Aunque acciones claves como copiar, mover o eliminar no están implementadas por sí solas y se hace necesario una combinación de acciones para lograr el objetivo deseado.

- `move`: show packages/search + add packages by key + delete packages by key
- `remove`: show packages/search + delete packages by key
- `copy`: show packages/search + add packages by key

Figura 4. Copiar, mover y eliminar. (Fuente: Aptly.info).

El API de Nova permite mostrar información sobre las distribuciones y los mirrors lo cual no es posible desde el API de Aptly. Para ello se accede a través del puerto 10002.

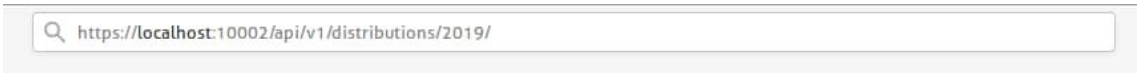

Figura 5. Petición para la información de la Distribución 2019. (Fuente: elaboración propia).

```
GET /api/v1/distributions/2019/

HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "distribution": "2019",
    "prefix": "nova-7",
    "label": "Nova",
    "origin": "Nova",
    "repositories_merge_name": "2019-development",
    "legacy_merge_name": "2019-legacy",
    "updates_name": "2019-updates",
    "incoming_repository": "2019-incoming",
    "accepted_repository": "2019-accepted",
    "rejected_repository": "2019-rejected",
    "unstable_repository": "nova-7.0-unstable",
    "repositories": [
        "nova-7.0-updates",
        "nova-7.0-repository",
        "nova-7.0-upd202002",
        "nova-7.0-upd281019"
    ],
    "components": [
        {
```

Figura 6. Respuesta del API a la petición de la Distribución 2019. (Fuente: elaboración propia).

2.2. Propuesta de solución

Una vez analizadas las dificultades que posee CESOL para administrar sus repositorios y teniendo en cuenta el resultado del análisis de sistemas existentes, la no utilización de las dos APIs que posee y la entrevista con el cliente se propone desarrollar un sistema con las siguientes características.

- Una herramienta web que permite la administración de los repositorios de la Distribución Cubana GNU/Linux Nova, consumiendo los servicios de las APIs existentes y permitiendo el acceso a las funcionalidades de ambas desde una sola herramienta.
- Agrupar las acciones del API de Aptly, copiar, mover y eliminar para que estas se ejecuten de manera secuencial y ordenada desde la interfaz con una sola interacción del usuario logrando el resultado esperado.
- La aplicación ejecutará correctamente, para las acciones de aceptar y rechazar, los movimientos a los repositorios correspondientes en cada caso y para cada distribución. Pidiendo confirmación antes de realizar estas acciones.
- La herramienta contará con una interfaz principal donde el usuario interactúa de manera sencilla con el sistema, el cual se encuentra ordenado de forma intuitiva, a través de un menú. Los paquetes de cada repositorio están listados de forma alfabética y las acciones están bien definidas, no existen ambigüedades.
- El sistema muestra información sobre las distribuciones y de los mirrors, los repositorios que existen, así como los paquetes que contienen y permite realizar modificaciones teniendo en cuenta el nivel de autorización que posea el usuario.

2.3. Descripción del negocio a informatizar

Para comprender el contexto del negocio a informatizar en la aplicación, se elaboró un modelo conceptual, en el que plasmó los conceptos, sus características y las relaciones existentes entre ellos.

El **Modelo conceptual** es una representación visual de los objetos, donde se describen los conceptos significativos, se identifican los atributos y las asociaciones existentes entre ellos (Incorporación del modelo conceptual en las buenas prácticas del Gobierno electrónico, 2018). A continuación, en la figura 7 se muestra el modelo conceptual de la propuesta de solución.

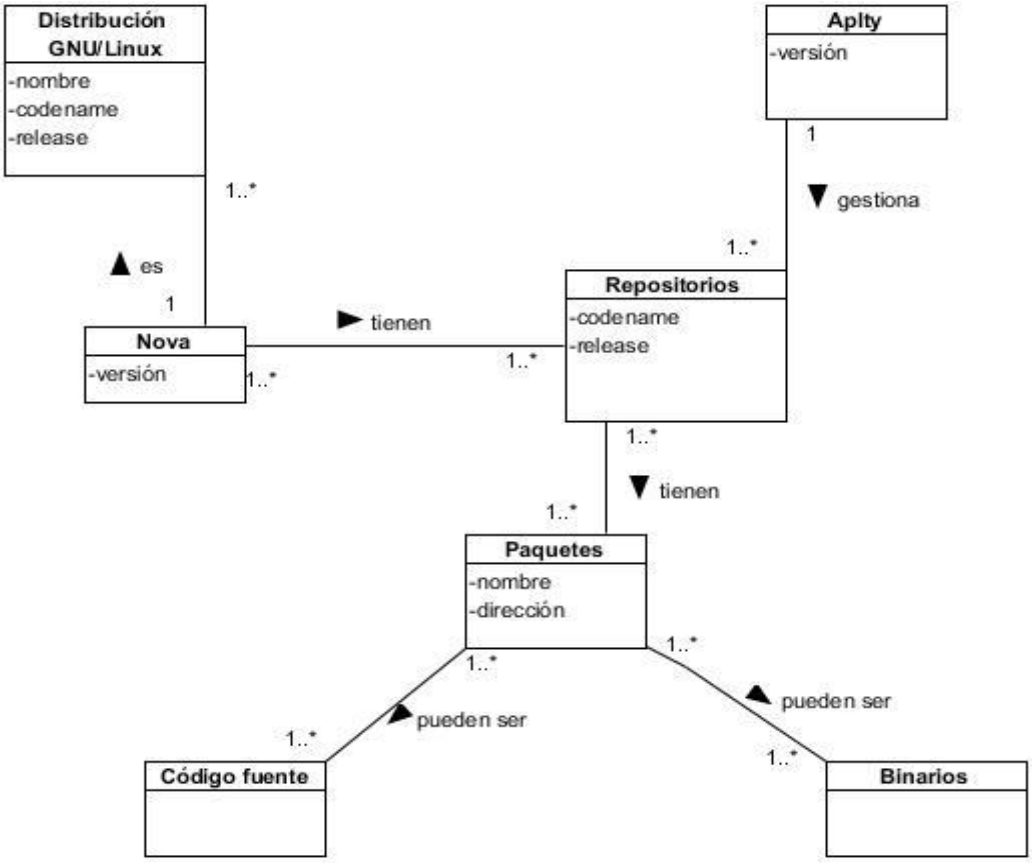

Figura 7. Modelo conceptual.(Fuente: Elaboración propia)

Los conceptos del contexto del negocio anteriormente modelado son los siguientes:

Distribución GNU/Linux: son distribuciones de software libres basadas en Linux.

Nova: es una distribución de GNU/Linux que garantiza la soberanía tecnológica en Cuba.

Repositorios: Es cualquier servidor o lugar en la red donde se encuentran los programas para el sistema operativo que lo está utilizando.

Aptly: es una herramienta que se utiliza para administrar repositorios en la Distribución Cubana GNU/Linux Nova.

Paquetes: es un grupo de uno o más archivos que se encuentran en los repositorios y son necesarios para la ejecución de un programa o para agregar características al mismo.

Código fuente: son paquetes que cuando se compilan generan paquetes binarios.

Binarios: es una colección de archivos ejecutables y complementarios necesarios para la ejecución de la aplicación.

2.4. Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto(Rodríguez, 2015).

Fuentes de obtención de requisitos

Para obtener los requisitos se utilizaron fuentes que proporcionan información y características que debe cumplir la propuesta de solución. Las fuentes utilizadas fueron:

- Especialistas de CESOL.

- Análisis de las herramientas existentes.

2.4.1. Técnicas de identificación de requisitos

Las técnicas de identificación de requisitos de software permiten identificar las necesidades de negocio de los clientes y los usuarios. Son mecanismos que se utilizan para recolectar la información necesaria en la obtención de los requisitos de una aplicación, permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle(Pressman, 2010).A continuación, se especifican las técnicas utilizadas:

Entrevista: La entrevista es de gran utilidad para obtener información cualitativa como opiniones o descripciones subjetivas de actividades. Es una técnica muy utilizada y requiere una mayor preparación y experiencia por parte del analista(Guerra, 2017).

La entrevista fue realizada a la persona que realiza la administración de repositorios en CESOL lo que permitió la obtención de los primeros requisitos (Ver Anexo 1).

Desarrollo de prototipos: los prototipos suelen consistir en vistas reducidas de la aplicación a desarrollar. Permite que el usuario cuente con una visión general de lo que desea y que puedan mejorar las especificaciones de los requerimientos(Guerra, 2017).

2.4.2. Especificación de requisitos de software

El objetivo principal de la Especificación de Requisitos de Software (ERS) es servir como medio de comunicación entre clientes, usuarios, ingenieros de requisitos y desarrolladores. En la ERS deben recogerse tanto las necesidades de clientes y usuarios como los requisitos que debe cumplir el software a desarrollar para satisfacer todas las necesidades(andalucia, 2017). A continuación, se relacionan y describen los requisitos funcionales y no funcionales definidos para la implementación de la propuesta de solución.

Requisitos Funcionales

Los requisitos funcionales (RF) del sistema son aquellos servicios que el usuario espera del sistema, deben ser completos y consistentes*(Prado, 2018)*. Describen todas las interacciones que se prevé que los usuarios tendrán con el software *(Applying an Improving Strategy that embeds Functional and Non-Functional Requierements concepts, 2019)*. En la tabla 2 que aparece a continuación se listan los requisitos funcionales de la propuesta de solución, su descripción y prioridad.

Tabla2.Requisitosfuncionales. (Fuente: Elaboración propia).

No	**Nombre**	Descripción	Prioridad
RF. 1	Autenticar usuario.	El sistema debe permitir autenticarse al usuario.	Baja
RF. 2	Adicionar usuario.	El sistema debe permitir adicionar usuario.	Baja
RF.3	Modificar usuario.	El sistema debe permitir modificar usuario.	Baja
RF.4	Listar usuario.	El sistema debe permitir mostrar usuario.	Media
RF.5	Eliminar usuario.	El sistema debe permitir eliminar usuarios.	Baja
RF.6	Listar paquete.	El sistema debe permitir mostrar un paquete.	Media
RF.7	Buscar paquete.	El sistema debe permitir buscarun paquete.	Alta
RF.8	Eliminar paquete.	El sistema debe permitir eliminarun paquete.	Alta
RF.9	Mover paquete.	El sistema debe permitir moverun paquete.	Alta
RF.10	Listar mirror.	El sistema debe permitir listar mirror.	Media
RF.11	Listar información de los mirror	El sistema debe permitir listar información de los mirror.	Media

RF.12	Listar información de las distribuciones.	El sistema debe permitir mostrar información de las distribuciones.	Media
RF.13	Adicionar grupo.	El sistema debe permitir adicionar un grupo.	Baja
RF.14	Modificar grupo.	El sistema debe permitir modificar un grupo.	Baja
RF.15	Listar grupo.	El sistema debe permitir mostrar un grupo.	Media
RF.16	Eliminar grupo.	El sistema debe permitir eliminar un grupo.	Baja

Requisitos No Funcionales.

Los requisitos No Funcionales (RNF) son requerimientos de calidad, que presentan restricciones o las cualidades que el sistema debe tener(Los requisitos no funcionales de software. Una estrategia para su desarrollo en el centro de Informática Médica., 2019). Existen diferentes clasificaciones para los requisitos no funcionales en dependencia de las características de calidad que se definan entre ellos: de fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y seguridad(andalucia, 2017). Con el objetivo de estandarizar la redacción de los requisitos no funcionales de Atributos de calidad se utilizó el ISO 25010. Propuesta en el producto de trabajo Especificación de requisitos de software del expediente de proyecto 4.0. A continuación se muestra la tabla 3 con los requisitos no funcionales y la descripción de los mismos.

Tabla3.Requisitos no funcionales. (Fuente: Elaboración propia).

No	Requisito	Sub-atributo	Descripción
RNF.1	Confiabilidad	Madurez	Evitar un fallo total como resultado de producirse un fallo del software.
RNF.2	Usabilidad	Aprendibilidad	El idioma de todas las interfaces de la aplicación será español. El sistema expondrá el menú general desde cualquiera de sus páginas.
RNF.3	Portabilidad	Adaptabilidad	El servidor estará corriendo sobre el sistema operativo Linux, pero puede correr sobre sistemas Windows también.
RNF.4	Funcionalidad	Preciso	El sistema realizará las operaciones indicadas en cada momento.
RNF.5	Mantenibilidad	Analizabilidad	Se implementará de forma estándar y las funcionalidades serán comentadas.

2.4.3. Descripción de requisitos de software.

Las Historias de Usuario son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones sobre las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea implementar cada funcionalidad(Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software, 2018). A continuación, se muestran las tablas 4 y 5 donde se describen los requisitos Buscar paquete y Mover paquete a través de historias de usuarios.

Tabla 4. Historia de usuario Buscar paquete. (Fuente: Elaboración propia).

<table>
<tr><td colspan="2"></td></tr>
<tr><td>Número: HU7</td><td>Requisito: Buscar paquete.</td></tr>
<tr><td colspan="2">Programador: Elena Castrillo Hernández | Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta | Tiempo Estimado: 72 horas</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Ausencia del desarrollador por enfermedad o pérdida de información imprescindible. | Tiempo Real: 52 horas</td></tr>
<tr><td colspan="2">Descripción: El usuario activa el botón “Buscar paquete” para buscar el paquete, proporciona los datos correspondientes para buscar paquetes.</td></tr>
<tr><td colspan="2">Observaciones: El sistema debe notificar al usuario cuando la operación termine o si dejó de llenar algún campo importante.</td></tr>
<tr><td colspan="2">Prototipo elemental de interfaz gráfica de usuario:
 
</td></tr>
</table>

Tabla 5. Historia de usuario Mover paquete. (Fuente: Elaboración propia).

Número: HU9	Requisito: Mover paquete.
Programador: Elena Castrillo Hernández	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 72 horas
Riesgo en Desarrollo: Ausencia del desarrollador por enfermedad o pérdida de información imprescindible.	Tiempo Real: 52 horas
Descripción: El usuario activa el botón “Mover paquete” para mover el paquete, proporciona los datos correspondientes para mover paquetes.	
Observaciones: El sistema debe notificar al usuario cuando la operación termine o si dejó de llenar algún campo importante.	
Prototipo elemental de interfaz gráfica de usuario: 	

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 disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos(Rodríguez Sánchez, 2015).

2.5.1 Diseño arquitectónico

El diseño arquitectónico es una actividad que permite estructurar de forma general al sistema y su objetivo es tener una visión clara del software a construir.

En el desarrollo de la propuesta de solución se utiliza el patrón arquitectónico Modelo Vista-Plantillautilizado por el marco de trabajo Django.

En este marco, el Modelo es la capa de acceso a la base de datos y contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene y las

relaciones entre los datos. La Vista es la capa de la lógica de negocios incluye la lógica que accede al modelo y la delega a la plantilla apropiada. Es una conexión entre los modelos y las plantillas. La Plantilla es la capa de presentación y comprende las decisiones relacionadas a la presentación: como son mostradas sobre una página web u otro tipo de documento(LibroDjango, 2017). A continuación, se muestra en la figura 8 el diagrama de paquetes donde se representa la arquitectura de la aplicación en el marco de trabajo según el patón arquitectónico.

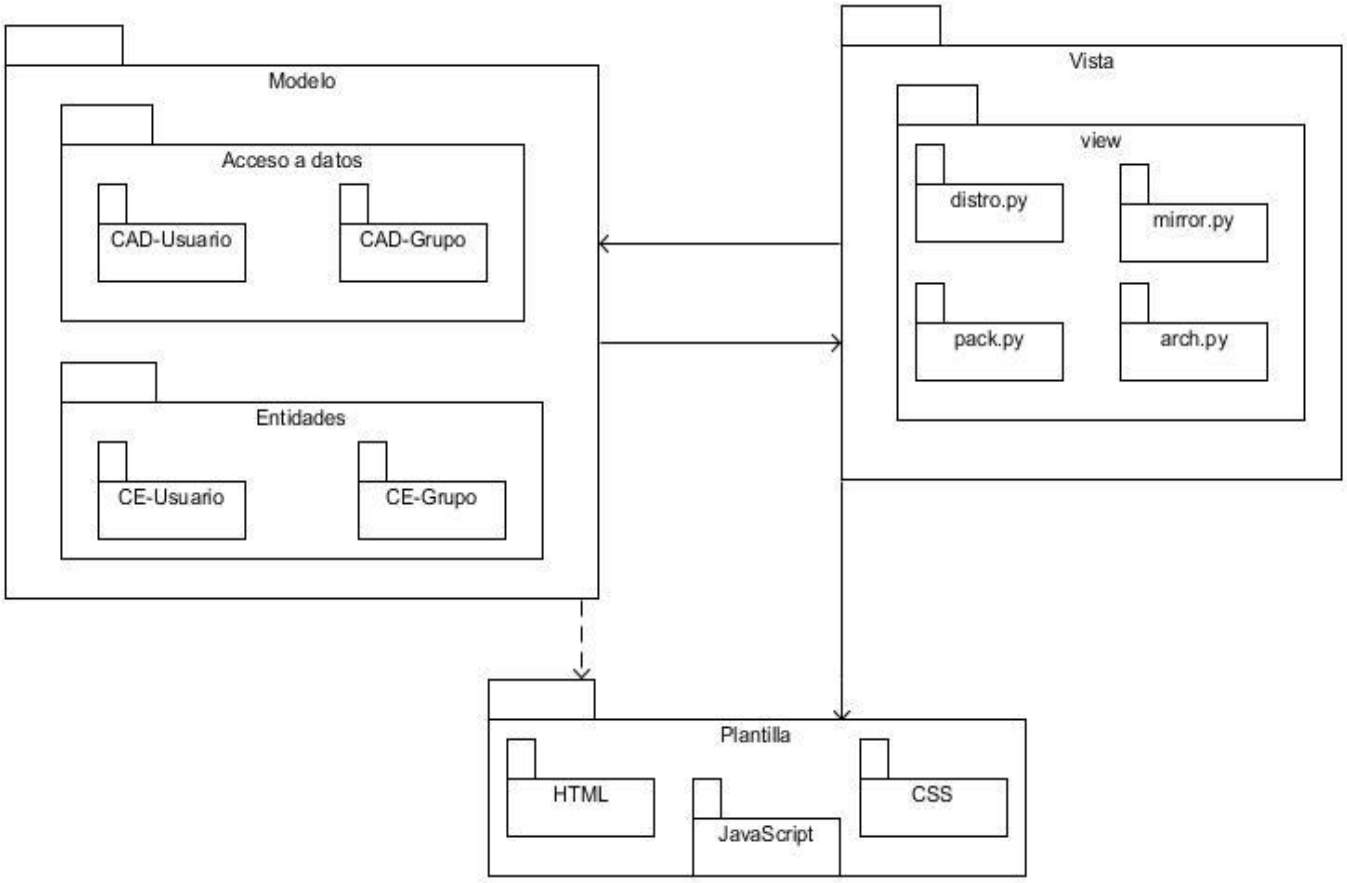

Figura8.Diagrama de paquetes. (Fuente: Elaboración propia).

Descripción

Vista: (View) es la capa que contiene la lógica del negocio. Es la encargada de acceder al modelo y delegar a la plantilla apropiada. En este caso desde esta capa se manejaría además la comunicación con las APIs, a través de los puertos 10000 y 10002 para la de Aptly y la de Nova respectivamente. Utilizando para esto el formato de intercambio JSON.

Modelo:(Model) es la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene y las relaciones entre los datos.

Plantilla: (Template) es la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación, como las cosas son mostradas y la interacción con el usuario.

2.5.2. Modelado de datos

Los modelos de datos definen cómo se modela la estructura lógica de una base de datos. Son entidades fundamentales para introducir la abstracción en una base de datos, definen cómo los datos se conectan entre sí y cómo se procesan y almacenan dentro del sistema(García, 2018). A continuación, la figura 3 muestra el diagrama de modelo de datos que será implementado en la propuesta de solución.

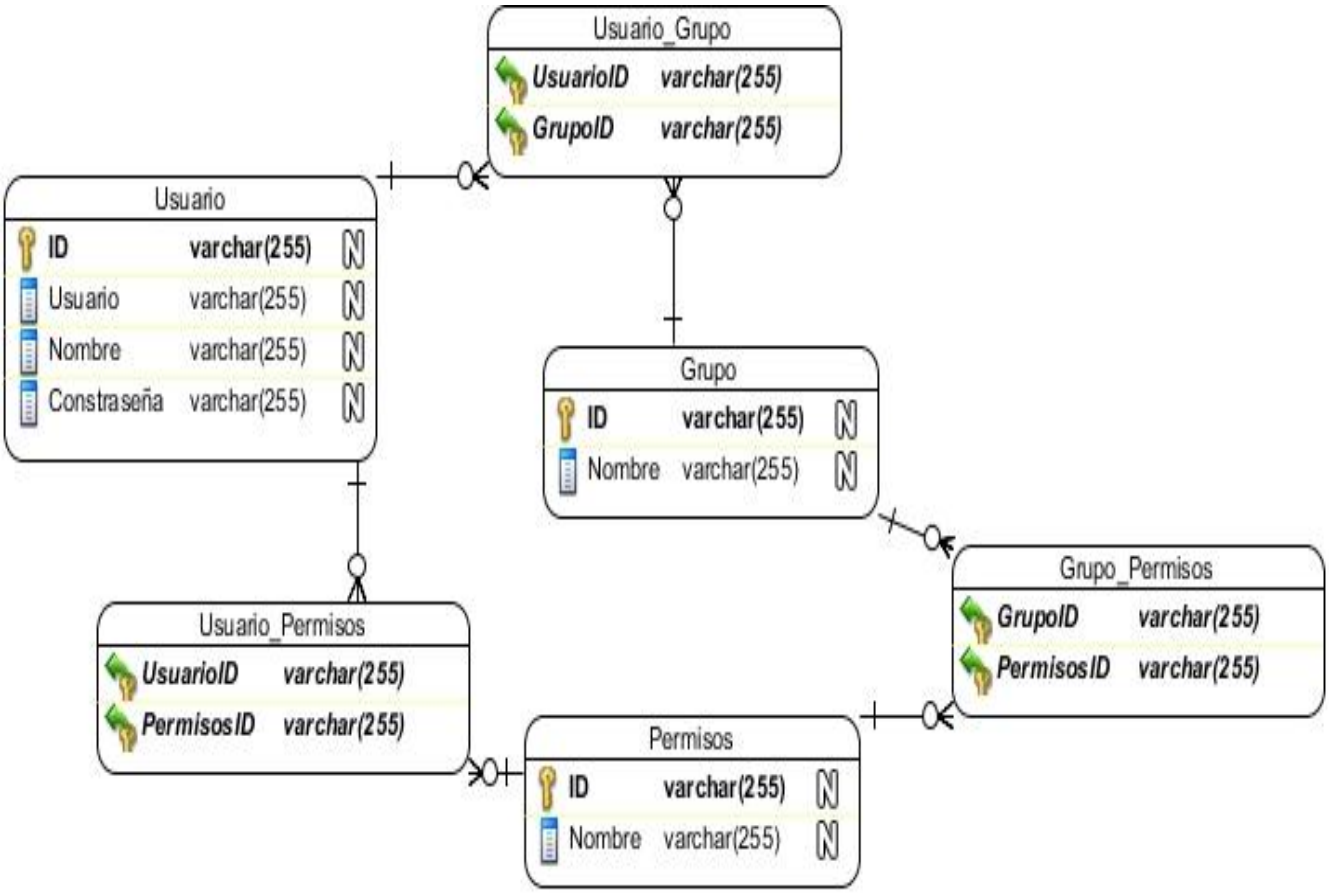

Figura9. Modelo de datos. (Fuente: Elaboración propia).

Este modelo está compuesto por las entidades (tablas) de la base de datos de PostgreSQL, a continuación, se explica el modelo de datos para un mayor entendimiento.

Usuario: se registra de un usuario del sistema el atributo que lo identifica (id_usuario), su usuario uci, la contraseña y su(s) nombre(s).

Grupo: Cada grupo presenta un conjunto de privilegios predeterminados que se conceden al usuario de la base de datos al que se le asigne, este tiene el atributo que lo identifica (id_grupo) y el nombre.

Permisos: Son los privilegios correspondientes a los distintos grupos, este tiene el atributo que lo identifica (id_permiso) y el nombre.

2.5.3. Modelado del diseño.

El diseño es la forma exacta en la que un requisito del cliente se puede convertir en un sistema o producto de software terminado. Crea una representación o modelo del software de forma detallada (Pressman, 2010).

A continuación, se presenta el modelado del diseño de la propuesta de solución.

Diagrama de clases del diseño.

El diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia(Sistemas groupware para el diseño de diagrama de clases uml en ambientes táctiles., 2018). En el caso de las aplicaciones web, el diagrama de clases representa las colaboraciones que ocurren entre las páginas, donde cada página lógica puede ser representada como una clase (Montes de Oca, 2017). A continuación, la figura 10 muestra el diagrama de clases con estereotipos web de la agrupación de requisitos Gestionar Paquete.

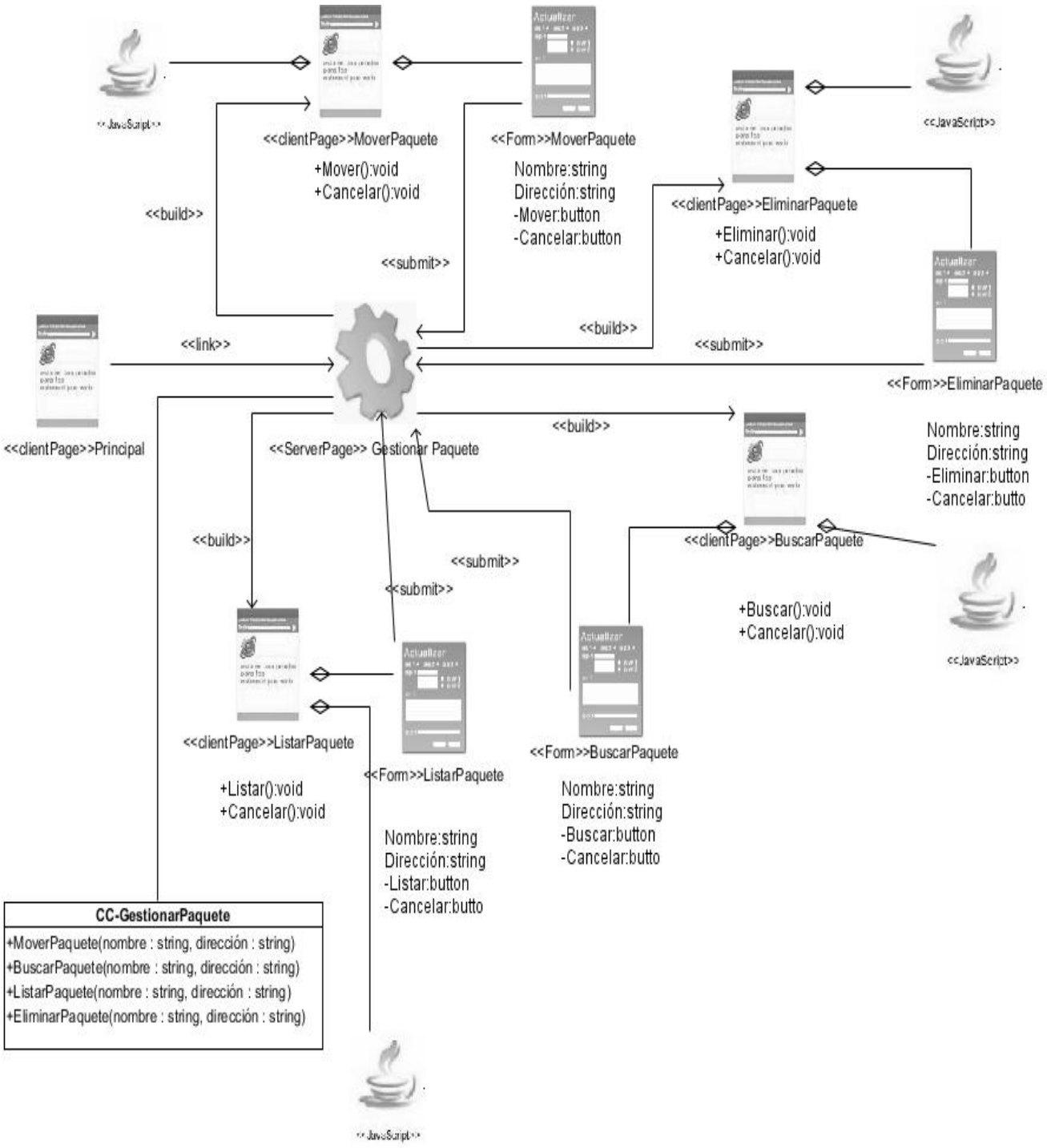

Figura 10. Diagrama de clases. (Fuente: Elaboración propia).

Patrones del diseño de software.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular (Ingeniería de software, 2018). En el diseño de la propuesta de solución se utilizaron los siguientes patrones.

GRAP.

Los Patrones generales de software para asignar responsabilidades (GRAP) nos dan unos principios generales para asignar responsabilidades y se utiliza sobre todo en la realización de diagramas de interacción. Cada patrón describe un problema, una solución y los beneficios de implementarla (Ingeniería de software, 2018).

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento(Ingeniería de software, 2018).El patrón se evidencia en la clase LoginView la cual tiene la responsabilidad de crear una instancia de AuthenticationLoginForm.

Bajo acoplamiento: Es el patrón que define cómo mantener bajas dependencias, bajo impacto al cambio e incrementar la reutilización. Asignar una responsabilidad de manera que el acoplamiento sea bajo (Ingeniería de software, 2018).Este patrón se utiliza en la clase AuthenticationLoginForm la cual para cumplir con sus funciones solo necesita relacionarse con la clase LoginView.

En la figura 11 se muestra la aplicación de los patrones GRAPS en la propuesta de solución.

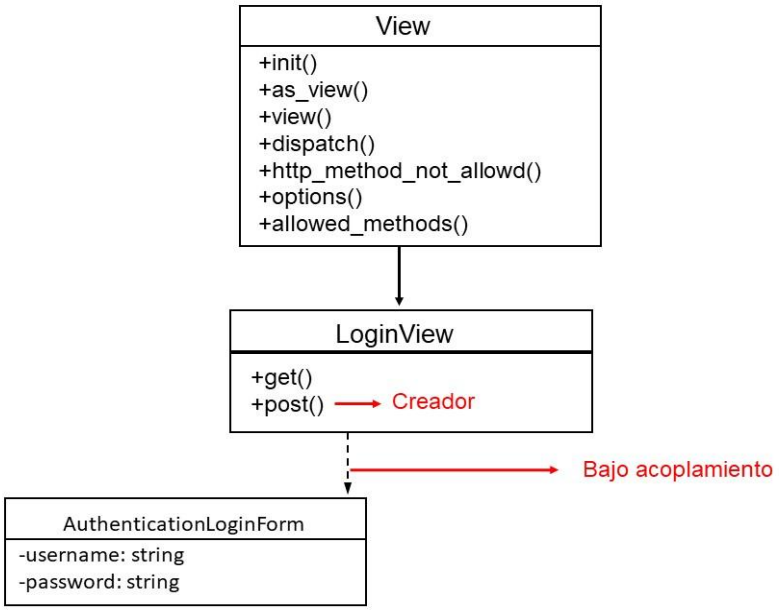

Figura11. Aplicación de los patrones GRAP. (Fuente: Elaboración propia).

GOF.

Los Patrones de la pandilla de los cuatros (GOF) permiten ampliar el lenguaje, aprender nuevos estilos de diseño y describen soluciones simples a problemas específicos en el diseño de software orientado a objetos (Ingeniería de software, 2018).

Decorador: se utilizó el patrón Decorador el cual permitió añadir responsabilidades adicionales a la plantilla base que almacena el código HTML común a todas las páginas de la aplicación. Las demás plantillas pueden heredar el código que esta posee y redefinir su contenido basada en esa estructura heredada.

A continuación, la figura 12muestra un fragmento de un código donde se evidencia la aplicación de los estándares de codificación.

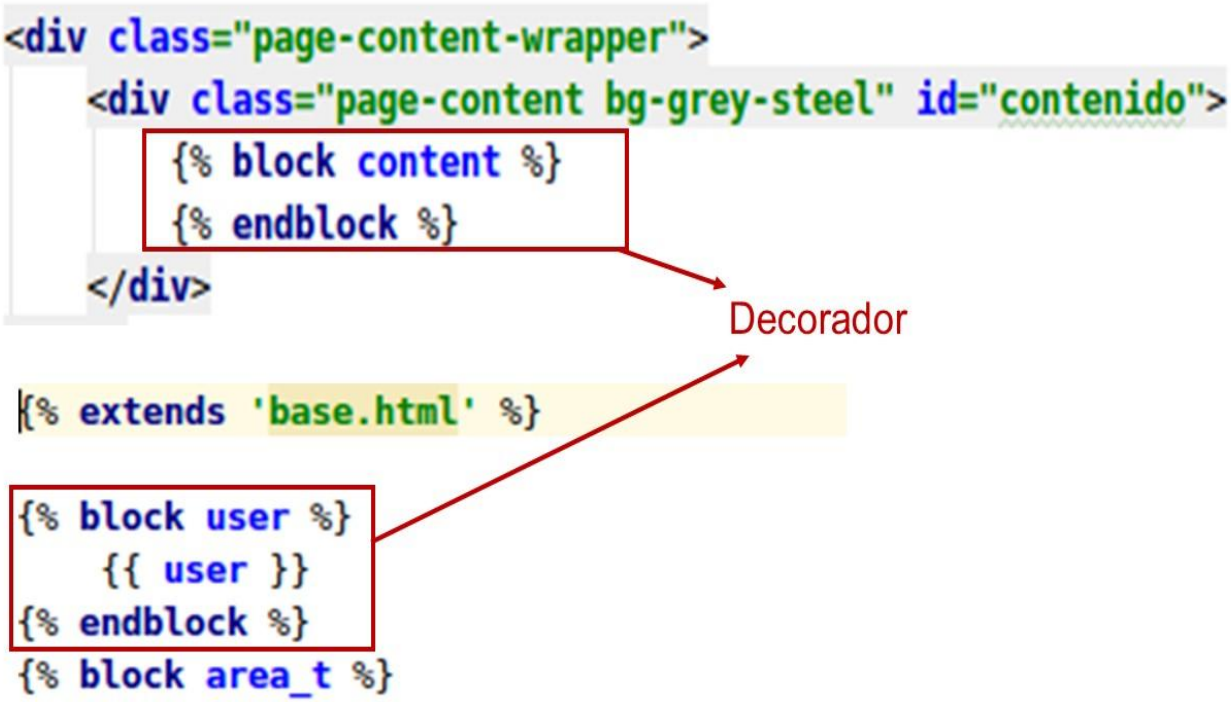

Figura12. Aplicación de los estándares de codificación. (Fuente: Elaboración propia).

2.6. Conclusiones del capítulo

En el presente capítulo se realizó una descripción de las características con las que tiene que contar la herramienta. A través de un modelo conceptual se relacionaron los conceptos fundamentales asociados a la administración de repositorios, se identificaron 16 requisitos funcionales y 5 no funcionales cumpliendo con las necesidades del cliente. Se elaboraron los diagramas de clases del diseño utilizando el patrón arquitectónico Modelo Vista Plantilla y los patrones de diseño GRASP y GOF quedando definida la arquitectura a seguir en la implementación de la herramienta web para administración de repositorios de la Distribución Cubana GNU/Linux Nova.

Capítulo 3: Implementación, pruebas y validación de la propuesta de solución

El presente capítulo se enfoca en la construcción del sistema a partir de los resultados del capítulo anterior. Se elaboran los diagramas de componentes y de despliegue y se define los estándares de codificación a utilizar en la implementación de la solución. Además, se realizan pruebas de software con el objetivo de descubrir y corregir errores y se evalúa la propuesta de solución.

3.1. Implementación

En la disciplina implementación, a partir de los resultados del Análisis y diseño se construye el sistema(Rodríguez, 2015).

3.1.1. Modelo de implementación

El modelo de implementación identifica los componentes físicos de la implementación para que puedan comprenderse y gestionarse mejor. Es el proceso de convertir una especificación del sistema en un sistema ejecutable, también puede incluir un código fuente de nivel bajo y archivos derivados, y sus relaciones con el modelo de diseño.

Diagrama de componentes

Los diagramas de componentes representan la forma en que se organizan los componentes y sus dependencias. Está clasificado como diagrama de estructura y, como tal, representa de forma estática el sistema de información (Trujillo, 2015). A través del diagrama de componente se modela la funcionalidad del software, donde cada componente encapsula código atómico añadiéndole la lógica a la aplicación(Aproximación basada en UML para el diseño y Codificación Automática de plataformas robóticas manipuladoras., 2017). A continuación, la figura 13 muestra el diagrama de componentes para la propuesta de solución.

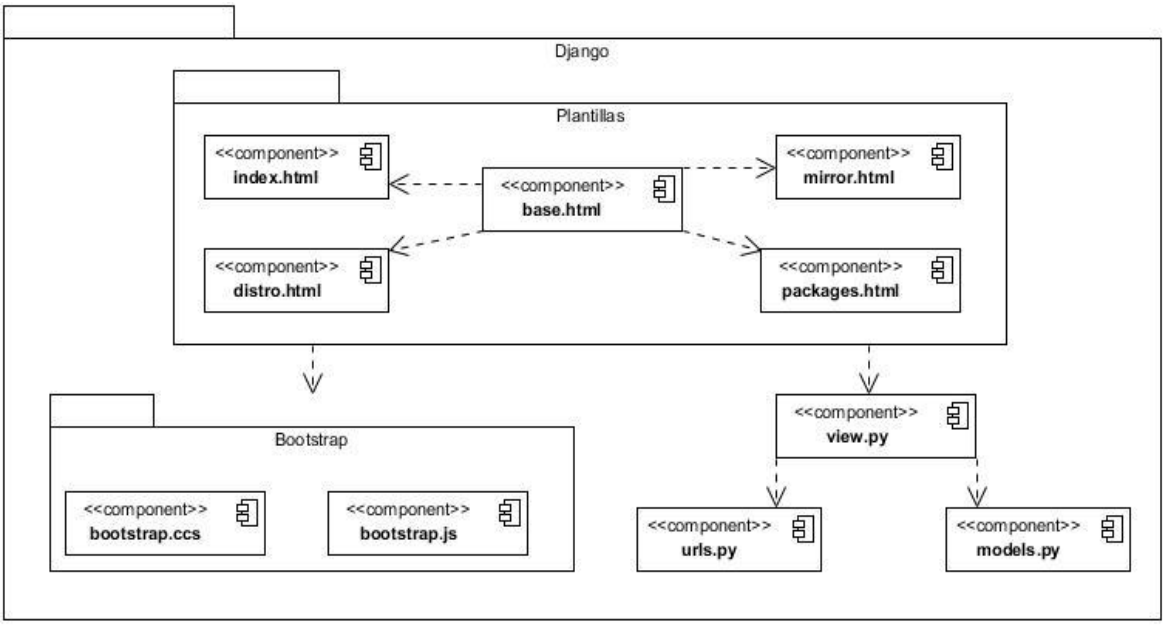

Figura13. Diagrama de componentes del sistema. (Fuente: Elaboración propia).

Descripción de los componentes

Plantillas: Contiene las plantillas o interfaces con las que interactúa el usuario.

Bootstrap: Contiene los archivos .css y .js (extensión de los archivos *JavaScript*) utilizados en las plantillas.

View: Archivo que contiene las views del sistema.

Urls: Archivo que contiene todas las url.

Models: Archivo que contiene todas las clases del modelo.

Diagrama de despliegue

Es utilizado para representar la distribución física (estática) de los componentes software en los distintos nodos físicos de la red. Con este diagrama, es posible modelar muchos de los aspectos hardware de un sistema con el nivel de detalle adecuado para que se pueda especificar la plataforma sobre la que se ejecutará, de tal forma que la frontera software - hardware del sistema sea mejor manejada(AS, 2016).

A continuación, la figura 14 muestra el diagrama de despliegue para la propuesta de solución.

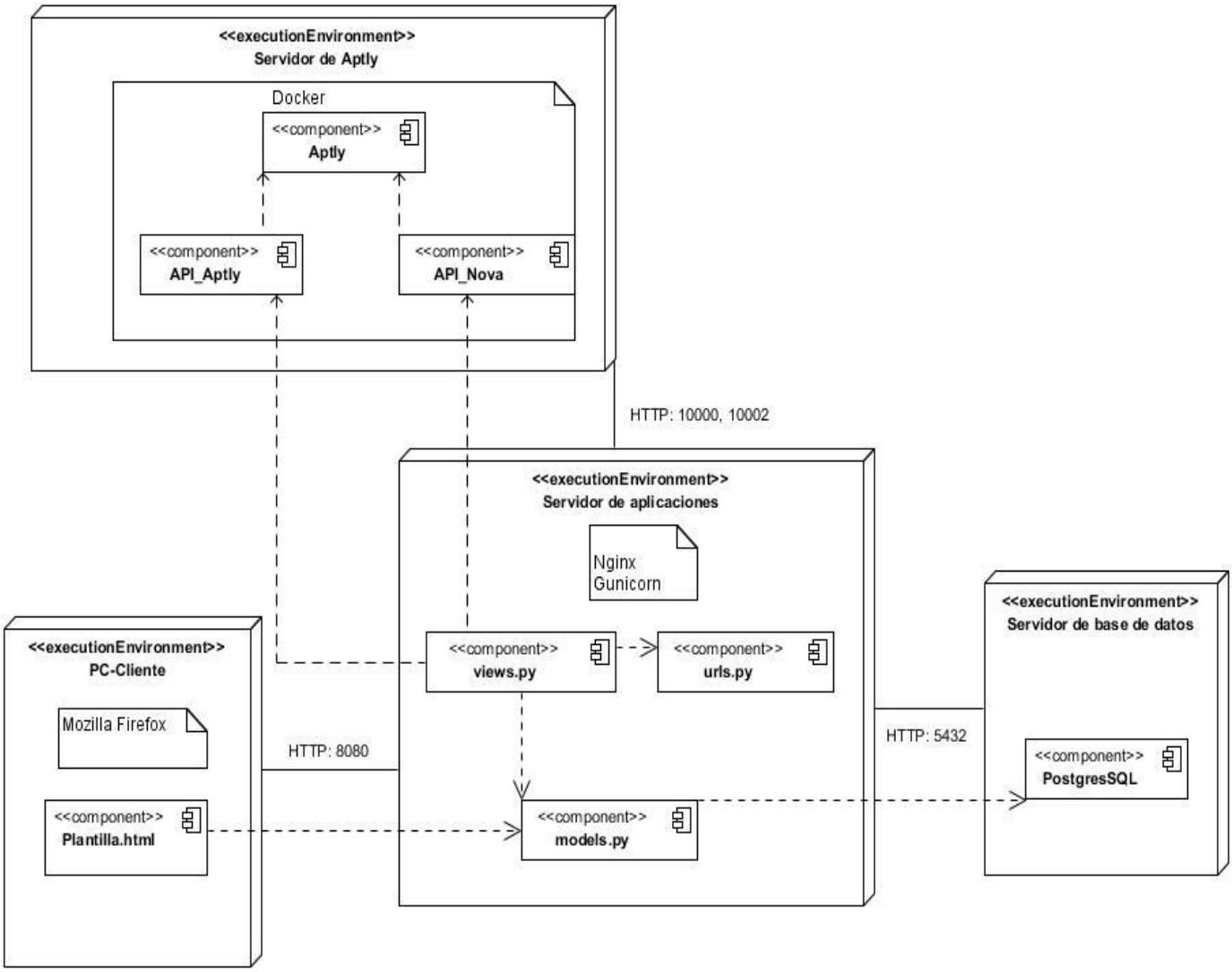

Figura 14. Diagrama de despliegue. (Fuente: Elaboración propia).

PC_Cliente: contendrá el navegador web Mozilla Firefox mediante el cual los usuarios tendrán acceso al sistema y harán uso del mismo.

Servidor de aplicaciones: proporciona los servicios de la aplicación a las computadoras cliente, gestiona las funciones de la lógica de negocio y de acceso a los datos necesarios. Este tendrá corriendo Nginx y Gunicorn.

Servidor de base de datos: Almacena toda la información referente a los usuarios, los grupos y los permisos concebidos a los mismos. La comunicación con el servidor de aplicaciones es a través del puerto 5432.

Servidor de Aptly: proporciona los servicios de API_Aptly y API_Nova. Contiene un docker en el que está Aptly, su API a la que se accede a través del puerto 10000 y el API desarrollada por Nova con acceso por el puerto 10002.

3.1.2. Estándares de implementación

Los estándares de implementación, son parte de las llamadas buenas prácticas, son un conjunto de convenciones establecidas de ante mano para la escritura de código. Estos estándares varían dependiendo del lenguaje de programación elegido(Merkury, 2017). Para la codificación de la propuesta de solución se utiliza el estándar de codificación para Python:

- **Indentación**: se utilizaron 4 espacios por cada nivel de indentación.
- **Tabuladoreso espacios**: no se mezclaron tabuladores y espacios en la codificación. Las formas más populares deindentar en Python son utilizando solo espacios o solo tabuladores, el código indentadocon una mezcla de tabuladores y espacios se reformateó y se usaron espacios exclusivamente.
- **Tamaño máximo de línea**: se limitaron todas las líneas a un máximo de 79 caracteres.
- **Convenciones de nombres**: no se utilizaron los caracteres 'l' (letra ele minúscula), 'O' (letra o mayúscula) o 'I' (letra i mayúscula) como nombres de variables de un solo caracter debido a que, en algunas fuentes, estos caracteres son indistinguibles de los numerales uno y cero.
  Se utilizó CapWords (palabras que comienzan con mayúsculas) para los nombres de las clases.
- **Otras consideraciones**: se rodearon siempre los siguientes operadores binarios con un espacio en cada lado: asignación (=), asignación aumentada (+=, -=, *=, /=), comparación (==, <, >, =, <>, <=, >=, in, not in, is, isnot), booleanos (and, or, not).
  Se utilizaron espacios alrededor de los operadores aritméticos.

A continuación, la figura 15muestra una demostración de la aplicación de los estándares de codificación.

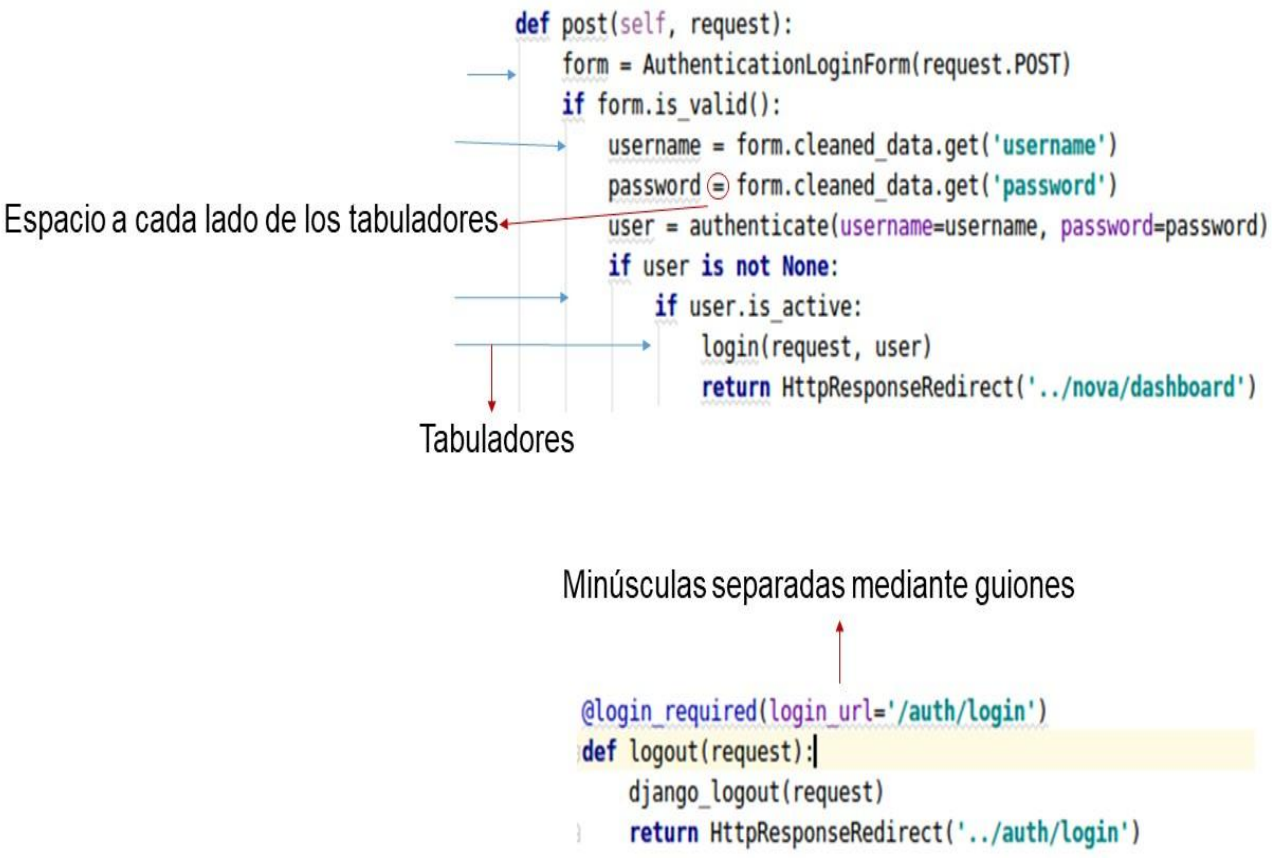

Figura15. Aplicación de los estándares de codificación. (Fuente: Elaboración propia).

3.1.3. Interfaz gráfica de usuario

La interfaz gráfica de usuario es parte fundamental de cualquier aplicación, permite representar las acciones y la información disponible utilizando un conjunto de imágenes y objetos gráficos. Su función principal consiste en facilitar un entorno visual sencillo que permita la comunicación con el ordenador(Albornoz, y otros, 2017)

A continuación, las figuras 16 y 17muestran algunas interfaces de la propuesta de solución.

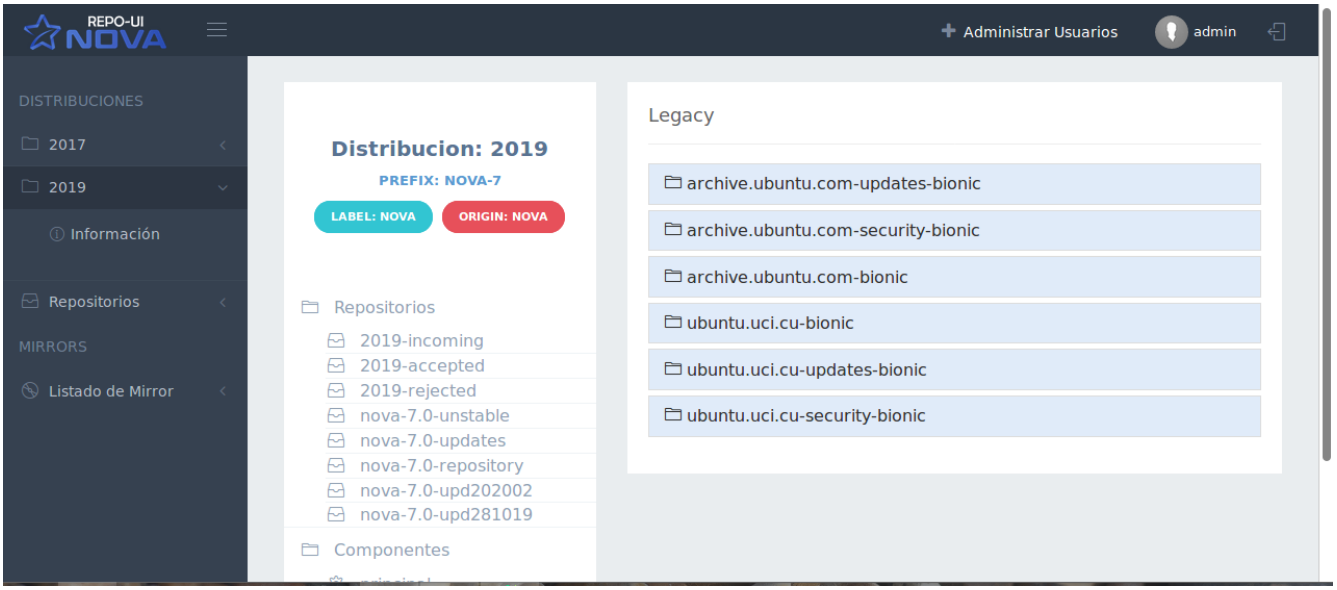

Figura16. Interfaz de la distribución 2019. (Fuente: Elaboración propia).

Figura17. Interfaz de los paquetes de ANDROID-ANTMAVEN. (Fuente: Elaboración propia).

3.2. Pruebas de software

Para asegurar un producto con calidad, se deben realizar una serie de pruebas que garanticen su óptimo funcionamiento antes de entregar el software al usuario final(VS. Nett Add-on for optimal definition of black box software testing using covering arrays, 2018). Según(Ares, 2017) las pruebas de software comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo.

3.2.1 Pruebas a realizar

Las pruebas que se le realizaron al software son las pruebas Unitarias y las Funcionales.

Pruebas unitarias

Son pruebas automatizadas que verifican la funcionalidad en el componente, clase, método o nivel de propiedad. El objetivo principal de las pruebas unitarias es tomar la pieza más pequeña de software comprobable en la aplicación, aislarla del resto del código y determinar si se comporta exactamente como esperamos. Cada unidad se prueba por separado antes de integrarlas en los componentes para probar las interfaces entre las unidades (Quijano, 2018).

Pruebas funcionales

Las pruebas funcionales se encargan principalmente de verificar que las funcionalidades desarrolladas en el sistema cumplan con sus especificaciones. Son pruebas automatizadas o manuales que prueban las funcionalidades de la aplicación o módulo construidos desde el punto de vista del usuario final, con sus diferentes roles, para validar que el software hace lo que debe y, sobre todo, lo que se ha especificado(Análisis de la aplicación de pruebas funcionales y pruebas de usabilidad de software en el desarrollo de sistemas web., 2019).

3.2.2. Métodos de pruebas

Los métodos de pruebas son procedimientos definitivos que producen un resultado de una prueba. Se centran en la búsqueda de errores y en comprobar el funcionamiento del sistema (Rodríguez, 2018).Las pruebas funcionales realizadas utilizan el método de caja negra y las unitarias cajas blancas.

Caja negra es un método de pruebas de software que verifica el correcto manejo de funciones externas soportadas por el software. Verifica que el comportamiento observado se responda a las especificaciones del producto y a las expectativas del usuario(Rodríguez, 2018).

Caja blanca este método verifica la correcta implementación de las pruebas internas y hace énfasis en la reducción de errores internos. Se basa en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa (Hoogenraad, 2018).

3.2.3. Técnicas de prueba

Las técnicas de pruebas tienen el objetivo de identificar condiciones de las pruebas, casos de pruebas y datos de pruebas para que se tenga la mayor probabilidad de encontrar errores reduciendo tiempo y esfuerzo(Sánchez, 2015).Las técnicas que se utilizaron fueron partición de equivalencia para las pruebas funcionales y camino básico para las pruebas unitarias.

Partición de equivalencia esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software (Garrido, 2016). Se trata de los valores de entrada del programa o del sistema que se dividen en grupos que tengan comportamiento similar, de manera que puedan ser procesados de la misma forma (Mendoza, 2019).

Camino básico consiste en verificar el código del sistema de manera que se compruebe que todo funciona correctamente, es decir, se verifica que todas las instrucciones del programa se ejecutan por lo menos una vez (Mause, 2017).

3.3. Aplicación de las pruebas de software

Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando los productos de trabajo implementados, a través de instrumentos de prueba como: diseños de casos de prueba y listas de chequeo (Rodríguez, 2015).

3.3.1 Pruebas unitarias

Las pruebas unitarias se desarrollaron utilizando la técnica del camino básico del método de prueba caja blanca. En esta técnica se utilizó la métrica del software complejidad ciclomática que proporciona una medida cuantitativa de la complejidad lógica de un programa o procedimiento. Cuando se utiliza en el contexto de prueba el valor calculado mediante la complejidad ciclomática define el número de caminos independientes en el conjunto básico de un programa o procedimiento y proporciona un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia almenas una vez (Pressman, 2010). A continuación, la figura 18representa el procedimiento Autenticar usuario.

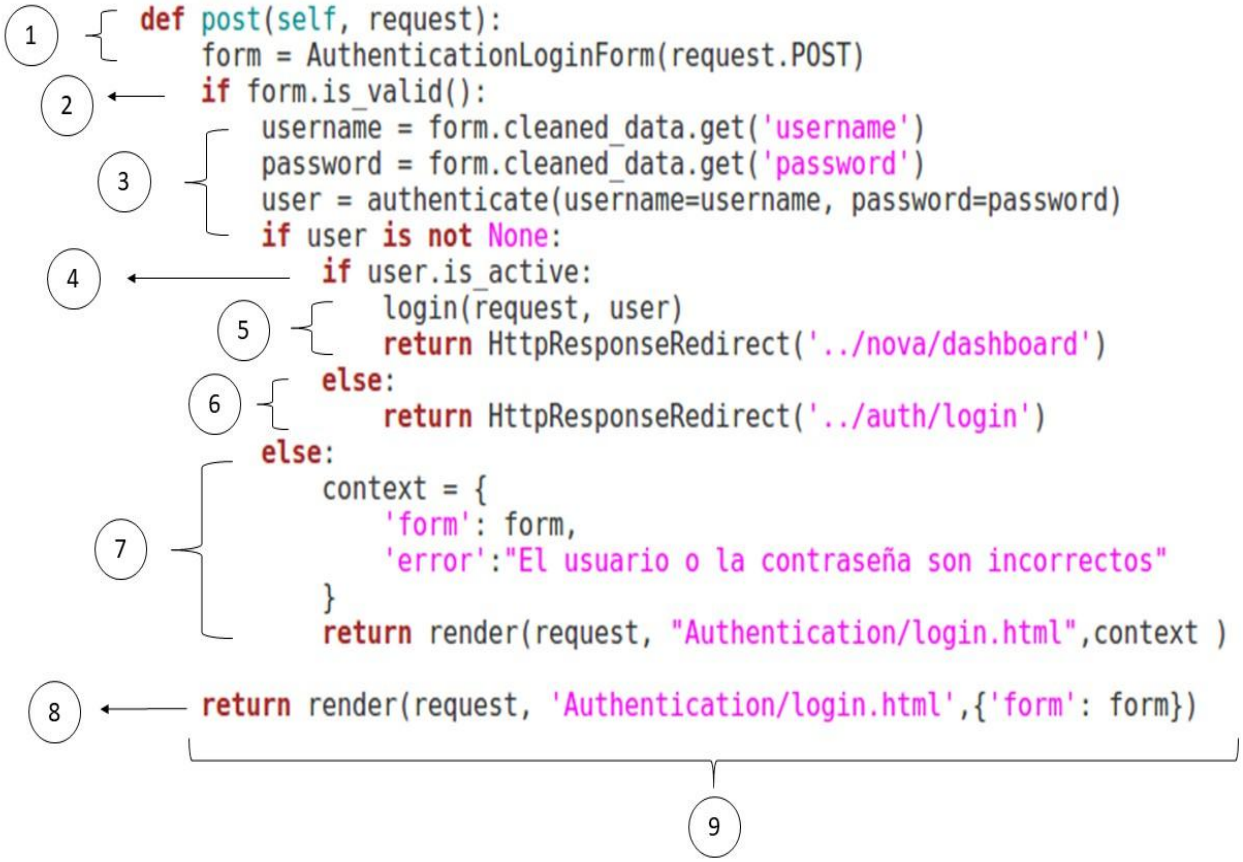

Figura 18. Procedimiento para Autenticar usuario. (Fuente: Elaboración propia).

Paso 1. Dibujar el grafo de flujo de la funcionalidad o procedimiento a analizar

En la figura 19se utilizó la notación de grafo de flujo para representar en un grafo de flujo el código del procedimiento a probar.

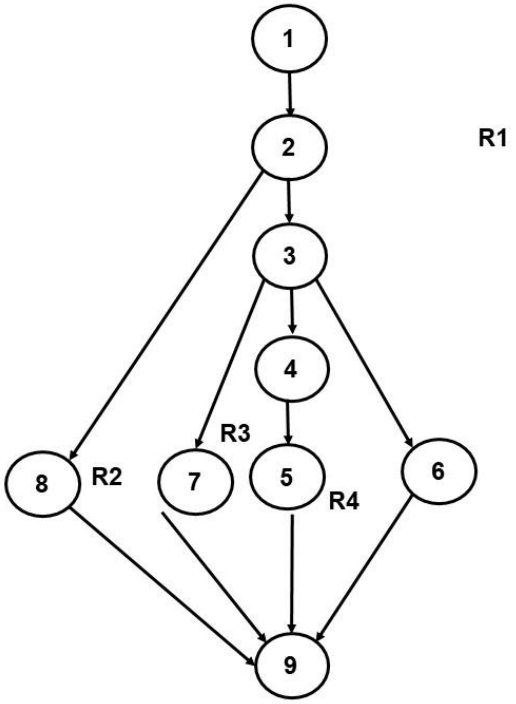

Figura 19. Grafo de flujo.(Fuente: Elaboración propia).

Paso 2. Determinar la complejidad ciclomática del grafo

La complejidad ciclomática del grafo V (G) se puede calcular de las tres formas siguientes:

V (G)= A-N+2 Dónde: A es el número de aristas del grafo de flujo y N es el número de nodos del grafo.

V (G)=11-9+2=4

V (G)=P+1 Dónde: P es el número de nodos predicados (nodos con más de una arista de salida) contenidos en el grafo.

V (G)=3+1=4

V (G)=R Dónde: R son las regiones, áreas delimitadas por nodos y aristas en el grafo.

V (G)=4

Paso 3. Determinar los caminos linealmente independientes

- **Camino básico 1:** 1,2,8,9
- **Camino básico 2:** 1,2,3,7,9
- **Camino básico 3:** 1,2,3,4,5,9
- **Camino básico 4:** 1,2,3,4,6,9

En el diseño de los casos de prueba se debe especificar los siguientes elementos:

- **Descripción:** contiene una descripción sobre las restricciones de los datos de entrada que debe tener el caso de prueba.
- **Condición de ejecución:** se especifican los parámetros que debe poseer el caso de prueba para que se cumpla una condición deseada como respuesta del funcionamiento del procedimiento.
- **Entrada:** se muestran los parámetros de entrada al procedimiento.
- **Resultados esperados:** se explica el resultado esperado de la ejecución del procedimiento.

Las tablas 6 y 7muestran los diseños de casos de pruebas para el camino 2y el camino 3 del conjunto básico de caminos linealmente independientes, correspondiente a la funcionalidad Autenticar.

Tabla 6. Caso de prueba para el camino 2.(Fuente: Elaboración propia).

	Diseño de caso de pruebas para el camino 2
Descripción	El usuario se autentica correctamente.
Conditioned ejecución	El usuario admin no ha sido creado con anterioridad.
Entrada	Usuario: admin Password: admin
Resultados esperados	El sistema muestra un mensaje: "El usuario o la contraseña son incorrectos".
Evaluación de la prueba	Satisfactorio. En este camino se prueba que el sistema no debe autenticar cuando el usuario o la contraseña son incorrectos.

Tabla 7. Casos de prueba para el camino 3. (Fuente: Elaboración propia).

	Diseño de caso de pruebas para el camino 3
Descripción	El usuario se autentica correctamente.
Condición de ejecución	El usuario admin ha sido creado con anterioridad.
Entrada	Usuario: admin Password: admin
Resultados esperados	El usuario se autentica satisfactoriamente y accede a la página principal.
Evaluación de la prueba	Satisfactorio. En este camino se prueba el autenticar un usuario de forma satisfactoria.

3.3.2. Pruebas funcionales

Se realizaron pruebas funcionales, a través del método de caja negra y la técnica de partición de equivalencia. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. A continuación, la tabla 8 muestra el Diseño de Caso de Prueba del requisito funcional Mover paquete.

Tabla 8. Caso de prueba del requisito funcional Mover paquete. (Fuente: Elaboración propia).

Escenarios	Descripción	Nombre	Dirección	Respuesta del sistema	Flujo central
EC.1.1Mover un paquete para un repositorio.	El sistema permite mover el paquete en caso de que no exista ningún error.	(Válido) ace-link	(Válido) documentos/paquete	El paquete (nombre del paquete) ha sido movido satisfactoriamente.	1.1El usuario selecciona el paquete. 1.2El usuario da clic en mover. 1.3El usuario selecciona el repositorio y da clic en aceptar.
EC1.2 Mover un paquete para un repositorio y que ya exista en él.	El sistema permite verificar si ya existe el paquete en el repositorio para el cual se desea mover.	(Inválido) iu-len	(Válido) documentos/paquete	El sistema muestra un mensaje “El paquete que desea mover ya existe.	1.1El usuario selecciona el paquete. 1.2El usuario da clic en mover. 1.3 El usuario selecciona el repositorio y da clic en aceptar.
EC1.3 Mover un paquete para un repositorio que no exista.	El sistema permite verificar si existe el repositorio para el cual se desea mover el paquete.	(Válido) ace-link	(Inválido) Vacío	1.1 El sistema muestra un mensaje “El repositorio no existe”,	1.1El usuario selecciona el paquete. 1.2El usuario da clic en mover. 1.3 El usuario selecciona el repositorio y da clic en aceptar.

Descripción de las variables

Tabla 9. Descripción de variables. (Fuente: Elaboración propia).

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Caracteres alfanuméricos y caracteres especiales.
2	Dirección	Campo de texto	No	Caracteres alfanuméricos

Las pruebas funcionales se realizaron en dos iteraciones, en la primera se encontraron 8 no conformidades, de ellas 5 de validación y 3 de ortografía, en la segunda iteración no se

encontraron no conformidades. En la Figura 20 se presenta un gráfico con los resultados de las pruebas.

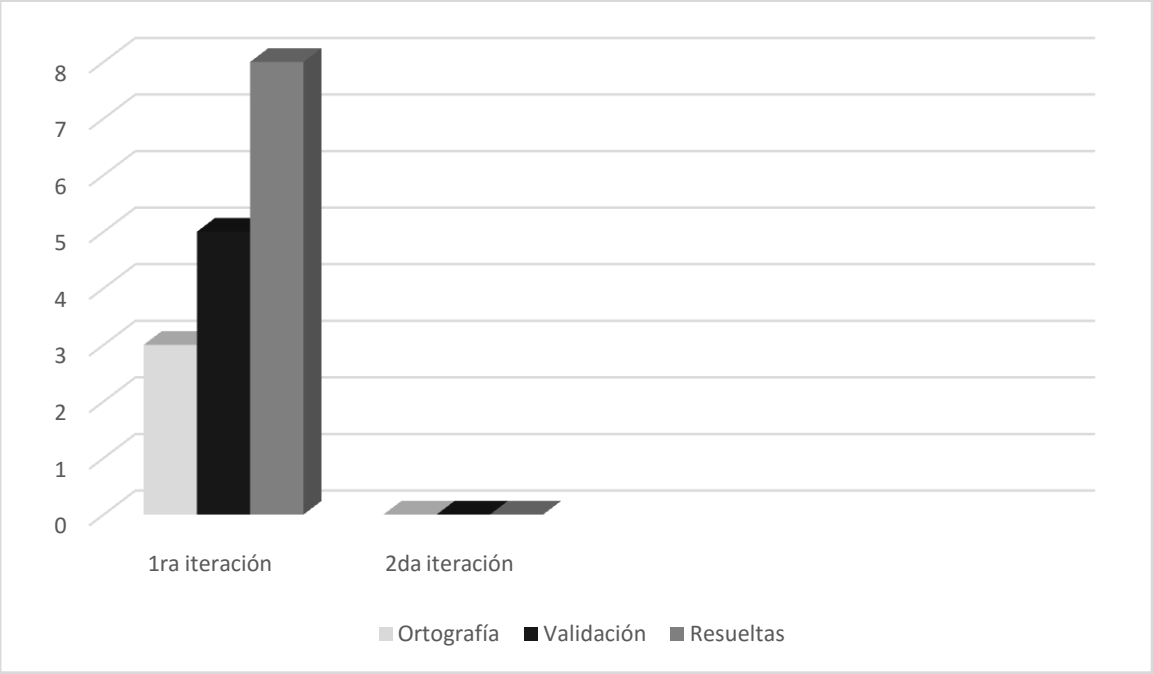

Figura 20. Resultados de las pruebas funcionales.(Fuente: Elaboración propia).

3.3.3. Evaluación de la solución propuesta

Con el objetivo de comprobar si la solución realmente contribuye a la eficiencia en la administración remota de repositorios de la Distribución Cubana GNU/Linux Nova se realizó una comparación entra la forma que existía para la administración remota de los mismos y la solución desarrollada. Para la comparación se utilizó la acción de aceptar el paquete firefox-locale-bg de la versión 2019.

Procedimiento existente para aceptar de forma remota el paquete firefox-locale-bg de la versión 2019

1. Realizar una búsqueda del paquete firefox-locale-bg en el repositorio 2019-incoming.

```
obel@obel-Inspiron-7370:~/Documentos/api/nova-ci$ curl http://localhost:10000/ap
i/repos/2019-incoming/packages?q=firefox-locale-bg
["Pamd64 firefox-locale-bg 69.0.1+build1-0ubuntu0.18.04.1.nova2 6564e57a6f64594d
```

Figura 21. Buscar el paquete firefox-locale-bg. (Fuente: Elaboración propia).

2. Copiar el paquete hacia el repositorio 2019-accepted utilizando para esto la clave del paquete obtenida en el paso anterior.

```
"]obel@obel-Inspiron-7370:~/Documentos/api/nova-ci$ curl -X POST -H 'Content-Typ
e: application/json' --data '{"PackageRefs": ["Pamd64 firefox-locale-bg 69.0.1+b
uild1-0ubuntu0.18.04.1.nova2 6564e57a6f64594d"]}' http://localhost:10000/api/rep
os/2019-accepted/packages
{"Name":"2019-accepted","Comment":"Development repository for 2019 distribution"
```

Figura 22. Copiar el paquete hacia el repositorio 2019-accepted. (Fuente: Elaboración propia).

3. Eliminar el paquete del repositorio 2019-incoming utilizando la clave del paquete.

```
obel@obel-Inspiron-7370:~/Documentos/api/nova-ci$ curl -X DELETE -H 'Content-Typ
epe: application/json' --data '{"PackageRefs": ["Pamd64 firefox-locale-bg 69.0.1
+build1-0ubuntu0.18.04.1.nova2 6564e57a6f64594d"]}' http://localhost:10000/api/r
epos/2019-incoming/packages
{"Name":"2019-incoming","Comment":"Development repository for 2019 distribution"
,"DefaultDistribution":"2019","DefaultComponent":"principal"}obel@obel-Inspiron-
```

Figura 23. Eliminar el paquete del repositorio 2019-incoming. (Fuente: Elaboración propia).

Procedimiento para aceptar de forma remota el paquete firefox-locale-bg de la versión 2019 con la solución desarrollada

1. Buscar el paquete firefox-locale-bg en el repositorio 2019-incoming y realizar la acción aceptar.

Arquitectura	Nombre	Verisión	Acciones
amd64	firefox-locale-bg	69.0.1+build1-0ubuntu0.18.04.01.nova2	Aceptar | Rechazar

Figura 24. Buscar el paquete firefox-locale-bg en el repositorio 2019-incoming. (Fuente: Elaboración propia).

La forma existente para administrar remotamente los repositorios en Nova lleva consigo que el usuario tenga la necesidad de poseer un amplio conocimiento de la estructura, organización y flujo de los paquetes en estos, para lograr que siempre vallan al repositorio adecuado. Además, se necesita un mayor número de acciones de más complejidad, provocando en ocasiones errores humanos, y el empleo de mayor tiempo. Lo que provoca que no se utilice esta vía con frecuencia, realizándose las acciones directamente en el servidor Aptly.

Con la herramienta desarrollada se simplifican las acciones y se abstrae al usuario de a dónde debe ir cada paquete, de cuáles se pueden eliminar y cuáles no, permitiéndose solo las acciones necesarias en cada caso. Evitando la introducción de errores humanos y reduciendo el tiempo de las operaciones. Por todo lo antes se puede concluir que la solución desarrollada contribuye de manera significativa a la eficiencia en la administración remota de repositorios de la Distribución Cubana GNU/Linux Nova.

3.4. Evaluación del objetivo de la investigación

Cuando se realiza una propuesta, es recomendable retroalimentarse con la opinión de los usuarios potenciales. Esta información es útil para conocer las debilidades de la propuesta y profundizar en sus fortalezas. En ese sentido, la técnica de Iadov es un instrumento que ayuda a conocer el grado de satisfacción de los potenciales usuarios.

Iadov es una técnica efectiva para el estudio del nivel de satisfacción de los participantes a través de la consulta a un panel de experto. Este método calcula el Índice de Satisfacción Grupal (ISG) se

implementa mediante un cuestionario (Ver anexo 3) en el cual se le incluyen tres preguntas cerradas que se intercalan dentro de un cuestionario de cinco preguntas y cuya relación el encuestado desconoce (Ruiz, 2016).

Estas tres preguntas se relacionan a través del "Cuadro Lógico de Iadov" el cual permite ubicar a cada encuestado, según el cuadro lógico en una escala de satisfacción, para luego calcular el ISG. La escala de satisfacción la cual toma valores de 1 a 6 es la siguiente 1-Clara satisfacción, 2-Más satisfecho que insatisfecho, 3-No definida, 4-Más insatisfecho que satisfecho, 5-Clara insatisfacción y 6-Contradictoria (Ruiz, 2016).

Para la aplicación de esta técnica se les realizó una encuesta a 10 desarrolladores del departamento de Nova y a 2 administradores del repositorio de Nova.

Tabla 10 Cuadro Lógico de Iadov. (Fuente: Elaboración propia)

5. Luego de haber visto la herramienta web para la administración de los repositorios de Nova, refleje en qué medida le gusta la solución desarrollada.	2. ¿La forma en que se realizaba la administración remota de repositorios en el departamento de NOVA, permitía satisfacer las necesidades existentes de manera eficiente?								
	No			**No sé**			**Si**		
	3. ¿Considera usted que la herramienta web permite contribuir a la eficiencia (tiempo empleado, un ámbito específico para cada función a realizar, entorno amigable para representar los resultados de búsqueda) de la administración remota de los repositorios de Nova?								
	Sí	**No sé**	**No**	**Sí**	**No sé**	**No**	**Sí**	**No sé**	**No**
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	3	4
No me gusta nada	6	6	6	6	4	4	6	6	5
No sé decir	2	3	6	3	3	3	6	6	4

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

- clara satisfacción (A)
- más satisfecho que insatisfecho (B)
- no definida (C)

- más insatisfecho que satisfecho(D)
- clara insatisfacción (E)
- contradictoria (C)

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

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

Donde N es la cantidad total de respuestas.

El valor del ISG permite identificar las siguientes categorías grupales:

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

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

Resultados obtenidos

1. Los resultados obtenidos de la aplicación de la encuesta se presentan en la Tabla 11.

Tabla 11. Resultados de la escala de satisfacción. (Fuente: Elaboración propia)

Categorías grupales de satisfacción	**N=12**	**Escala**
Clara satisfacción	10	A
Más insatisfecho que satisfecho	2	B
No definido	0	C
Más satisfecho que insatisfecho	0	D
Máximo de satisfacción	0	E
Contradictorio	0	C

2. Cálculo del Índice de Satisfacción Grupal

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

ISG= (10(+1) +2(+0.5)) /5 = 0.91

3. Interpretación del resultado del ISG.

El valor obtenido del ISG fue 0.91 lo que indica máxima satisfacción de los usuarios con respecto a la herramienta web para la administración de los repositorios de la Distribución Cubana GNU/Linux Nova. Se puede afirmar que se cumplió el objetivo de la investigación. Las respuestas a las preguntas abiertas brindadas por los encuestados reafirman los beneficios que traerá la utilización del sistema propuesto.

3.5. Conclusiones del capítulo

En el desarrollo del capítulo se transitó por las disciplinas, Implementación, Pruebas internas y Pruebas de aceptación. La implementación permitió obtener una herramienta web que permite contribuye a la eficiencia de la administración de los repositorios de la Distribución Cubana GNU/Linux Nova cumpliendo con los estándares de codificación definidos. Las pruebas unitarias comprobaron que el flujo de trabajo de las funcionalidades es correcto y las pruebas funcionales contribuyeron a identificar y corregir 8 no conformidades. Se evidenció la satisfacción del cliente a través de las pruebas de aceptación y la aplicación de la técnica de Iadov la cual propició la evaluación satisfactoria de la herramienta web para la administración de repositorios.

Conclusiones

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

- El análisis de los referentes teóricos y de los sistemas informáticos estudiados evidencian limitaciones y fortalezas, que se tienen en cuentaen el desarrollode la herramienta webpara la administración de repositorios de la Distribución Cubana GNU/Linux Nova.
- La herramienta para la administración de los repositorios de la Distribución Cubana de GNU/Linux Nova,permite a los administradores de repositorios de NOVA realizar las operaciones en menores intervalos de tiempo.
- La herramienta para la administración de los repositorios de la Distribución Cubana de GNU/Linux Nova reduce significativamente la introducción de errores humanos en el proceso, facilitando de forma simplificada el ámbito adecuado para cada funcionalidad.
- La herramienta para la administración de los repositorios de la Distribución Cubana de GNU/Linux Nova, representa en un ambiente visual personalizado los resultados de las búsquedas.
- Existe dependencia significativa entre Aptly y la herramienta para la administración de los repositorios de la Distribución Cubana de GNU/Linux Nova. Las actualizaciones de Aptly a futuras versiones superiores evoca cambios que afectan la API desarrollada por NOVA, y por ende el funcionamiento de la herramienta web.
- La herramienta para la administración de los repositorios de la Distribución Cubana de GNU/Linux Nova, es un resultado factible para los administradores de repositorios de NOVA, tal como evidencia la aplicación de técnicas y pruebasque garantizan el correcto funcionamiento de la aplicación, y demostraron la satisfacción del cliente hacia el sistema desarrollado.

Recomendaciones

Para futuras investigaciones relacionadas con la administración de repositorios de la Distribución Cubana GNU/Linux Nova se recomienda lo siguiente:

- Incorporar otras funcionalidades, como mostrar los logs de compilación de los paquetes y la posibilidad de revertir acciones realizadas, en caso de que la funcionalidad llevada a cabo lo permita.
- Establecer acciones de soporte a la herramienta para la administración de los repositorios de la Distribución Cubana de GNU/Linux Nova y toda la infraestructura de la cual es dependiente, en caso de modificación en cualquiera de ellos, migración o incorporación de elementos tecnológicos.
- Realizar periódicamente análisis del estado del arte de sistemas homólogos a Aptly, para considerar el uso de la herramienta web para la administración de los repositorios de la Distribución Cubana de GNU/Linux Nova, en caso de que faciliten una API adecuada.

Referencias

Albornoz, María Claudia, Berón, Mario y Montejano, Germán Antonio. 2017. Repositorio institucional de la UNLP. [En línea] abril de 2017. [Citado el: 16 de marzo de 2020.] http://sedici.unlp.edu.ar/handle/10915/62078.

Alegsa, Leandro. 2019. alegsa. [En línea] octubre de 2019. [Citado el: 20 de octubre de 2019.] http://www.alegsa.com.ar/Dic/servidor_de_base_de_datos.php.

Análisis de la aplicación de pruebas funcionales y pruebas de usabilidad de software en el desarrollo de sistemas web. **Larrea, Natalia. 2019.** 3.4, s.l. : Ciencia Digital, 2019, Vol. 3.

andalucia, Junta de. 2017. juntadeandalucia.es. [En línea] 2017. [Citado el: 22 de enero de 2020.] www.juntadeandalucia.es/servicios/madeja/contenido/recurso/407.

Applying an Improving Strategy that embeds Functional and Non-Functional Requierements concepts. **Becker, Pablo. 2019.** 2, s.l. : Journal of Computer Science & Technology, 2019, Vol. 19.

Aproximación basada en UML para el diseño y Codificación Automática de plataformas robóticas manipuladoras. **Estévez, E. 2017.** 1, s.l. : Revista Iberoamericana de Automática e Informática Industrial, 2017, Vol. 14.

APTLY. 2018. aptly.info. [En línea] 2018. [Citado el: 1 de octubre de 2019.] https://www.aptly.info.

Ares, Laura. 2017. visual-engin. [En línea] 2017. [Citado el: 16 de marzo de 2020.] http://visual-engin.com/2017/10/26/importancia-pruebas-de-software-testing/.

AS, Elvis. 2016. lideshare. [En línea] febrero de 2016. [Citado el: 27 de febrero de 2020.] https://es.slideshare.net/ElvisAR/diagrama-de-despliegue-58634994.

Blancarte, Oscar. 2017. oscarblancarteblog. [En línea] 16 de octubre de 2017. [Citado el: 10 de octubre de 2019.] https://www.oscarblancarteblog.com/2017/10/26/ide.

Carles, Joan. 2017. geekland. [En línea] 2017. [Citado el: 29 de septiembre de 2019.] https://geekland.eu/que-son-los-repositorios-en-linux/.

Chacon, Scott. 2015.*Pro Git Book.* 2015.

DEBIAN. 2019. debian.org. [En línea] 2019. [Citado el: 6 de febrero de 2020.] https://www.debian.org/devel/buildd/index.es.html.

developer.mozilla.org. 2019. developer.mozilla.org. [En línea] septiembre de 2019. [Citado el: 27 de enero de 2020.] https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducci%C3%B3n.

Dil. 2016. repositoriodig. [En línea] 5 de abril de 2016. [Citado el: 29 de septiembre de 2019.] http://repositoriodig.blogspot.com.

Docker ecosystem-vulnerability analysis. **Martin, Antony. 2018.** s.l. : Computer Communications, 2018, Vol. 122.

Domingo, I., Rius, G., y Cuenca L. 2018. Una revisión sobre el estado del arte en herramientas de modelado basado en UML. 6th. [En línea] julio de 2018. [Citado el: 20 de enero de 2020.]

Evaluación de los frameworks en el desarrollo de Aplicaciones Web con Python. **Ríos, Jimmy Rolando. 2016.** 4, s.l. : Revista latinoamericana de Ingeniería de Software, 2016, Vol. 4.

Fernández-Sanguino, Javier. 2016. Debian. [En línea] 2016. [Citado el: 18 de enero de 2020.] https://www.debian.org/doc/manuals/debian-faq/ch-pkg_basics.en.html#s-package.

Fuentes Rodríguez, Juan Manuel. 2018.*PROCEDIMIENTO DE GESTIÓN DE LOS REPOSITORIOS DE LA DISTRIBUCIÓN CUBANA DE GNU/LINUX NOVA .* 2018.

Fuentes, Juan Manuel. 2018.*GESTIÓN DE REPOSITORIOS DE PAQUETES DE NOVA CON APTLY.* 2018.

Gandarillas, Aurelio. 2017. metodologia.es. [En línea] 25 de junio de 2017. [Citado el: 21 de enero de 2020.] https://metodologia.es/marco-de-trabajo/.

García, Dia. 2018. tecnologias-informacion. [En línea] 2018. [Citado el: 29 de enero de 2020.] https://www.tecnologias-informacion.com/modelos-datos.html.

Garrido Tejero, A. 2016. riunet.upv. [En línea] 2016. [Citado el: 17 de marzo de 2020.] https://riunet.upv.es/handle/10251/63579.

GNU. 2018. gnu. [En línea] 2018. [Citado el: 29 de enero de 2020.] https://www.gnu.org.

González, Lucía. 2018. dinahosting. [En línea] 6 de septiembre de 2018. [Citado el: 20 de octubre de 2019.] https://blog.dinahosting.com/herramientas-de-control-de-versiones/.

Guerra, Cesar. 2017. Obtención de Requerimientos. Técnicas y Estrategia. [En línea] 2017. [Citado el: 6 de febrero de 2020.] https://sg.com.mx/17/obtencion-requerimientos-tecnicas-yestrategia..

Gunicorn.org. 2017. Gunicorn.org. [En línea] 2017. [Citado el: 27 de febrero de 2020.] https://gunicorn.org/.

HERTZOG, Raphaël and MAS, Roland. 2015.*The Debian Administrator's Handbook .* 2015.

Hoogenraad, Wim. 2018. es.itpedia. [En línea] febrero de 2018. [Citado el: 17 de marzo de 2020.] https://es.itpedia.nl/2018/02/05/white-box-testing-onder-de-loep/.

Humanos. Humanos. [En línea] UCI. http://www.humanos.uci.cu.

Incorporación del modelo conceptual en las buenas prácticas del Gobierno electrónico. **Medina, Oscar. 2018.** 31, s.l. : Revista Tecnología y Ciencia, 2018.

Ingeniería de software. **UCI. 2018.** 2018.

Launchpad. 2017. launchpad.net. [En línea] 2017. [Citado el: 1 de octubre de 2019.] https://launchpad.net.

lenguajesdeprogramacion.net. 2018. lenguajesdeprogramacion.net. [En línea] 2018. [Citado el: 20 de enero de 2020.] https://lenguajesdeprogramacion.net/python/.

LibroDjango. 2017. uniwebsidad. [En línea] 2017. [Citado el: 29 de enero de 2020.] https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv.

Los requisitos no funcionales de software. Una estrategia para su desarrollo en el centro de Informática Médica. **Hernández Molina, Yenisel. 2019.** 2, s.l. : Revista cubana de Ciencias Informáticas, 2019, Vol. 13.

Mause, JC. 2017. c-mouse. [En línea] octubre de 2017. [Citado el: 18 de marzo de 2020.] http://www.jc-mouse.net/ingenieria-de-sistemas/caja-blanca-prueba-del-camino-basico.

Mendoza, Jhonny. 2019.*Análisis del sistema informático de la unidad de tránsito en el municipio del cantón Las Naves.* s.l. : Tesis de licenciatura, 2019.

Merkury. 2017. ohmyroot. [En línea] 12 de enero de 2017. [Citado el: 16 de marzo de 2020.] https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/.

Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. **Montero, Bryan. 2018.** 17, s.l. : Espirales revista multidisciplinaria de investigación., 2018, Vol. 2.

Montes de Oca, Yanirys. 2017. enciclopedia virtual. [En línea] 2017. [Citado el: 29 de enero de 2020.] http://www.eumed.net/libros-gratis/1232/diagrama-clases-diseno.html.

Naranjo, David. 2017. desdelinux. [En línea] 2017. [Citado el: 10 de octubre de 2019.] https://blog.desdelinux.net/pycharm-un-entorno-de-desarrollo-para-python/.

Nginx.org. 2017. Nginx.org. [En línea] 2017. [Citado el: 10 de enero de 2020.] https://nginx.org/en/. .

Olarte, Luis. 2018. conogasi. [En línea] abri de 2018. http://conogasi.org/articulos/lenguaje-de-programacion/.

opensuse. 2017. opensuse. [En línea] 2017. [Citado el: 3 de octubre de 2019.] https://es.opensuse.org/Build_Service.

Otalora, Cindy. 2018.*Servidor de aplicaciones.* s.l. : Nuevas tecnologías de desarrollo, 2018.

postgresql.org. 2019. postgresql.org. [En línea] 2019. [Citado el: 3 de febrero de 2020.] https://www.postgresql.org/about/.

Postgresql: Extensión del Postgresql para el manejo de datos con frecuencias temporales. **Aguilera, Ana. 2016.** 4, s.l. : Tekné, 2016, Vol. 1.

Prado, Luis. 2018. mindmeister. [En línea] 2018. [Citado el: 22 de enero de 2020.] www.mindmeister.com/es/ingenieria-de-requisitos.

Pressman, R.S. 2010.*Software engineering: a practitioner's approach.* New York : 7th ed, 2010.

Pythonízame. 2016. PyCharm Community Edition. [En línea] 2016. [Citado el: 7 de enero de 2020.] http://pythoniza.me/pycharm-community-edition. .

Python-The Fastest Growing Programming Language. **Srinath, KR. 2017.** 12, s.l. : International Research Journal of Engineering and Technology (INJERT), 2017, Vol. 4.

Quijano, Juan. 2018. genbeta. [En línea] octubre de 2018. [Citado el: 17 de marzo de 2020.] https://www.genbeta.com/desarrollo/que-pruebas-debemos-hacerle-a-nuestro-software-y-para-que.

Research on Nginx Dynamic Load Balancing Algorithm. **Qin, E. 2020.** 2020.

Rodríguez, Eduardo. 2018.*Estrategia y técnicas de pruebas de software.* 2018.

Rodríguez, Rolando. 2017. gestiopolis. [En línea] 2017. [Citado el: 4 de octubre de 2019.] gestiopolis.com..

Rodríguez, Tamara. 2015.*Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

Rouse. 2018. searchSoftwareQuality. [En línea] 6 de septiembre de 2018. [Citado el: 10 de septiembre de 2020.] https://www.searchsoftwarequality.techtarget.com.

Rubio, Daniel. 2017.*Beginning Django: Web Application Development and Deployment with Python.* s.l. : Apress, 2017.

Rubio, Juan Carlos. 2019. openwebinars. [En línea] 25 de febrero de 2019. [Citado el: 21 de octubre de 2019.] https://openwebinars.net/blog/que-es-git-y-para-que-sirve/.

Ruiz, María de los Ángeles. 2016. Modelo para la implementación de la gestión de documentos en el sistema empresarial cubano. [En línea] 2016. [Citado el: 20 de marzo de 2020.] http://www.congreso-info.cu/index.php/info/2016/paper/viewFile/287/40.

Rumbaugh, J., Jacobson, Ivar., y Booch, Grady.,. 2017. El Lenguaje Unificado de Modelado. [En línea] 2017. [Citado el: 1 de febrero de 2020.] https://juanantonioleonlopez.files.wordpress.com/2017/05/el-lenguaje-unificado-de-modelado-manual-de-referencia.pdf.

Sánchez, José Manuel. 2015.*Pruebas de software. Fundamentos y técnicas.* 2015.

Serialización/Deserialización de objetos y transmisión de datos con JSON. **Mora, Juan Antonio. 2016.** 1, s.l. : Revista Tecnología en Marcha, 2016, Vol. 29.

Sistemas groupware para el diseño de diagrama de clases uml en ambientes táctiles. **Hernández, Nancy. 2018.** 127, s.l. : Pistas Educativas, 2018, Vol. 39.

Springer, Alex. 2015. ticbeat.c. [En línea] 2015. [Citado el: 27 de enero de 2020.] https://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/.

Targetware. 2016. software. [En línea] 29 de octubre de 2016. [Citado el: 10 de octubre de 2019.] //www.software.com.ar/p/visual-paradigm-para-uml..

Trujillo, Estalin. 2015. slideshare. [En línea] 2015. [Citado el: 16 de marzo de 2020.] https://es.slideshare.net/omar1023/diagrama-de-componentes-43490802.

VS. Nett Add-on for optimal definition of black box software testing using covering arrays. **Meneses, Herney. 2018.** 33, s.l. : Revista Ingenierías Universidad de Medellín, 2018, Vol. 17.

UBUNTU. 2017. Repositories/Ubuntu. [En línea] 2017. https://help.ubuntu.com/community/Repositories/Ubuntu.

Zúñiga, Albert. 2017.*Metodología de desarrollo de software.* Perú : s.n., 2017.

ANEXOS

Anexo 1: Entrevista realizada al especialista que tiene la tarea de administrar repositorios en CESOL.

1. ¿Cree usted que es importante administrar los repositorios de la Distribución Cubana GNU/Linux Nova? ¿Por qué?

2. ¿Cuáles son los aspectos fundamentales a tener en cuenta cuando se realiza una administración de los repositorios?

3. ¿Por qué se encuentra en desuso el API desarrollado en CESOL?

4. ¿Cree necesario la creación de una herramienta web que permita realizar la administración de estos repositorios? ¿Por qué?

5. ¿Cuáles serían las características que tendría esta herramienta?

Anexo 2: Guía de observación para el proceso de administración de los repositorios de la Distribución Cubana GNU/Linux Nova.

Observador: Elena Castrillo Hernández

Lugar: Laboratorio del proyecto de Nova

Objetivo: Identificar los requisitos fundamentales para la realización del proceso de construcción de la herramienta web para la administración de los repositorios.

1- Datos de identificación del proceso
   - Nombre del proceso
   - Especialista que ejecuta el proceso
2- Características del espacio donde se desarrolla el proceso
   - ¿Dónde se debe realizar?
   - ¿Bajo qué condiciones se debe realizar?
   - ¿Qué se necesita para iniciar el proceso?
3- Características del proceso.
   - ¿Cómo es el proceso?
   - ¿Qué herramientas se emplean?
   - ¿Cuáles son los pasos a seguir?
   - ¿Cuál es el resultado?

Anexo 3: Encuesta a especialistas de CESOL de la Universidad de las Ciencias Informáticas.

Especialista, le invito a responder la siguiente encuesta, con el fin de conseguir su colaboración en la presente investigación. Solicito que exprese en sus respuestas, criterios verídicos que guíen al autor del trabajo. Marque en cada pregunta con una X en una sola opción y en el caso de la 5 responda brevemente. Por el tiempo brindado, muchas gracias.

1. ¿Considera que la administración de los repositorios de la Distribución Cubana GNU/Linux Nova son necesarias en la universidad?

   Sí ___ No ___ No sé___

2. ¿La forma en que se realizaba la administración remota de repositorios en el departamento de NOVA, permitía satisfacer las necesidades existentes de manera eficiente?

   Sí ___ No ___ No sé ___

3. ¿Considera usted que la herramienta web permite contribuir a la eficiencia (tiempo empleado, un ámbito específico para cada función a realizar, entorno amigable para representar los resultados de búsqueda) de la administración remota de los repositorios de Nova?

   Sí___ No___ No sé___

4. ¿Qué beneficios puede traer a la universidad utilizar la herramienta desarrolladaen función de la eficiencia en la administración de los repositorios de Nova?
5. Luego de haber visto la herramienta web para la administración de los repositorios de Nova, refleje en qué medida le gusta la solución desarrollada.

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