Facultad 1

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Módulo para la gestión de usuarios, grupos, procesos y repositorios del sistema GNU/Linux desde HMAST

Autora:

Zuleimys García Rodríguez

Tutores:

Ing. María Leisy González Carrera

Ing. Yosel Lázaro Vera González

La Habana, junio 2017

El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento...”

Fidel Castro Ruz

Dedicatoria

El presente trabajo está dedicado a las dos personas más importantes de mi vida, sin las cuales este momento no hubiese sido posible:

Má a ti que has sabido ser una mujer con más decisión y coraje que muchos hombres, que me has dado la vida y las fuerzas para llegar donde estoy hoy. Por ser mi razón de ser.

Papito a ti que has sido siempre el hombre de mi vida, mi primer amor, quien me ha malcriado más que nadie en el mundo y a quien le debo todo.

Agradecimientos

A mis padres, por llevarme siempre de la mano por el camino correcto y ser siempre mi modelo a seguir… cuando sea grande quiero ser como ustedes.

A ti mami que me supiste enseñar que el tiempo si es suficiente cuando la voluntad se impone y que no hay barrera que frene a quien quiere crecer, hacer, o ser…

A ti Papito porque de pequeñita me enseñaste a soñar en grande y ahora no le puedo poner límite. Me enseñaste que a pesar de los fracasos siempre hay que volver a intentar, que el destino o la suerte no es lo que define, sino el empeño y la constancia, y por ello, me esfuerzo cada día.

A mis hermanos por cuidarme siempre, aunque la distancia no haya sido nunca nuestra mejor aliada, por luchar eternamente porque me superara y fuera la mejor versión de mí.

A Miguel Ángel (Tato) por los momentos que hemos vivido, los difíciles, y los muy buenos. Siempre estuve siguiendo tus pasos por el camino de la vida; incluso hoy donde estamos, aún sigo tus pasos, recuerda cuidarlos, que yo te sigo.

A Miguel Ángel (Tata) por estar siempre pendiente de mí. Aunque no hayan sido muchos los momentos, si es inmenso el amor que siento por ti, y mucho más grande el orgullo de ser tu hermanita.

A Hassem, el niño más grande, por dar los primeros pasos y dejar que te siguiéramos.

***A ustedes, muchas gracias, por saber poner el listón bien en alto, y prepararme el camino**.*

A mi abu Amable, por todos los momentos compartidos y las historias contadas, pero más que nada por dejarme conocer lo que se siente tener abuelo.

A mi hermanito de crianza, Yoseilán, por cuidar siempre a mi mamá cuando nosotros no estamos, por ser su ángel.

A mi Papitichi, que me ha sacado más de una sonrisa en momentos difíciles. Por calmarme en tiempo de tormenta y brindarme su hombro cuando lo necesité para dejar mis lágrimas. Por llegar a mi vida en el momento justo: Te amo.

A Dayame y Pedro, que fueron otros padres y me adoptaron como uno más de los suyos, gracias por aguantarme todos estos meses. Sin duda son unas personas maravillosas, los quiero mucho.

Agradecimientos

A mis amigas Stephany y Rachel, por transitar conmigo estos años de la vida. Por todas las fiestas y los momentos buenos (que fueron muchos) y por las lágrimas que me secaron en tiempo malos. Por demostrarme que no debemos compartir equipos en los trabajos y que siempre puedo contar con ustedes, por ser mis amigas… y por las fiestas que nos quedan. Las quiero muchoooo.

A mis tutores, Mari y Yosel, por forzarme intentar las cosas y demostrarme que si puedo, que contrario a lo que yo pensaba, si era capaz. Por los momentos de desacuerdo, por el esfuerzo y el tiempo invertido, por ser ustedes y no otros quienes me guiaron por este proceso, de corazón, gracias.

A Nurisel, otra de mis tutores (ella no lo sabe, pero para mí siempre lo ha sido). Gracias por enseñarme, guiarme, apoyarme y estar ahí cuando te necesité. Porque las mujeres si podemos ser programadoras…

A mis muchachitas Ivet, Anietsy y Anabel, por ser amigas, por compartir juntas incontables momentos y estar a mi lado cuando no lo esperaba. Espero que no acabe aquí.

A mis amigos del aula Liset, Michael, Javier, Asney, Edel, Lexys, Nelson y Yosbel por dejarme contar con ustedes durante estos 5 años, por cuidarme, acogerme y ayudarme, los quiero mucho.

A los muchachos de Cesol Leosdanys y Felipe, que me ayudaron muchísimo durante el desarrollo de la tesis.

A los amigos que encontré durante la carrera, a todos los profesores que me ayudaron a llegar donde estoy.

A la vida.

Declaración de Autoría

Declaro por este medio que yo **Zuleimys García Rodríguez**, con carné de identidad **93121934654** soy la autora principal del trabajo titulado "**Módulo para la gestión de usuarios, grupos, procesos y repositorios del sistema operativo GNU/Linux desde HMAST**" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firman la presente a los ____ días del mes de junio del año 2017.

_____

Zuleimys García Rodríguez

Autora

_____ _____

Ing. María Leisy González Carrera

Tutora

Ing. Yosel Lázaro Vera González

Tutor

Resumen

La presente investigación sostuvo como objetivo el desarrollo de un módulo para la Herramienta de Migración y Administración de Servicios Telemáticos que gestione los usuarios, grupos, procesos y repositorios del sistema operativo GNU/Linux, de forma que facilite el trabajo de los administradores de redes durante el proceso de migración a código abierto. Se emplearon los métodos Analítico-Sintético, Histórico-Lógico, Modelación y Experimental. Se analizaron las diversas herramientas que permiten la gestión del sistema definiéndose la terminal de líneas de comandos para llevar a cabo la gestión de usuarios, grupos, procesos y repositorios del sistema. La metodología de software que se utilizó para el desarrollo de la propuesta de solución fue Proceso Unificado Ágil en su variación para la Universidad de las Ciencias Informáticas y las principales tecnologías empleadas fueron: IntelliJ IDEA como entorno integrado de desarrollo, Java como lenguaje de programación, Spring como marco de trabajo y Visual Paradigm para el modelado de los procesos. El módulo permite la gestión de usuarios, grupos, procesos y repositorios del sistema operativo a través de una interfaz web, mediante la cual se agiliza el proceso de configuración, evita errores humanos por parte de los administradores y disminuye la complejidad del proceso de migración.

Palabras clave: gestión, grupo, proceso, repositorio, usuario.

Índice

FIGURAS Y TABLAS 3

INTRODUCCIÓN 4

CAPÍTULO 1: LA GESTIÓN DE LOS SISTEMAS OPERATIVOS GNU/LINUX. DESCRIPCIÓN DE HMAST. 11

1.1 Gestión de usuarios, grupos, procesos y repositorios desde GNU/Linux 11

1.1.1 Gestión de usuarios 11

1.1.2 Gestión de grupos 12

1.1.3 Gestión de procesos 13

1.1.4 Gestión de repositorios 16

1.2 Herramientas para la gestión de usuarios, grupos, procesos y repositorios 18

1.3 Herramienta de Migración y Administración de Servicios Telemáticos 22

1.3.1 Arquitectura de la herramienta HMAST 22

1.3.2 Aspectos importantes al implementar un módulo para HMAST 24

1.4 Herramientas, tecnologías y metodología de software 25

1.4.1 Herramientas y Tecnologías 25

1.4.2 Metodología de desarrollo de Software 31

CONCLUSIONES PARCIALES 33

CAPÍTULO 2: LA GESTIÓN DE USUARIOS, GRUPOS, PROCESOS Y REPOSITORIOS DEL SISTEMA DESDE HMAST 34

2.1 Descripción general de la propuesta de solución de la investigación 34

2.2 Descripción de requisitos del sistema 34

2.2.1 Requisitos funcionales 34

2.2.2 Requisitos no funcionales 37

2.3 Historias de usuario 38

2.4 Arquitectura del módulo 43

2.5 Diagrama de paquetes 44

2.6 Patrones de diseño 46

2.6.1 Patrones GRASP 47

2.6.2 Patrones GoF 48

CONCLUSIONES PARCIALES 48

Índice

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DE LA GESTIÓN DE USUARIOS, GRUPOS, PROCESOS Y REPOSITORIOS DESDE HMAST. 49

3.1 Estándares de codificación 49

3.2 Pruebas del sistema 50

3.2.1 Prueba de unidad 51

3.2.2 Prueba de integración 58

3.2.3 Prueba de validación 59

3.2.4 Prueba de aceptación 62

3.3 Diagrama de despliegue 63

CONCLUSIONES PARCIALES 63

CONCLUSIONES GENERALES 64

RECOMENDACIONES 65

REFERENCIAS BIBLIOGRÁFICAS 66

ANEXO 1 69

Figuras y Tablas

FIGURA 1 DIRECCIÓN DE REPOSITORIO GNU/LINUX NOVA. 18
FIGURA 2 INTERFAZ DE LA HERRAMIENTA USERS-ADMIN. 19
FIGURA 3 ARQUITECTURA DE HMAST. 23
FIGURA 4 *ARQUITECTURA DEL MÓDULO SYSTEMMANAGER EN LA HERRAMIENTA HMAST.* 44
FIGURA 5 DIAGRAMA DE PAQUETES DEL MÓDULO SYSTEMMANAGER EN LAS CAPAS DE HMAST 45
FIGURA 6 DIAGRAMA DE PAQUETES DE LA CAPA DOMAIN CON LAS ENTIDADES DEL MÓDULO. 46
FIGURA 7 ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO. 69

TABLA 1 COMPARACIÓN DE HERRAMIENTAS DE GESTIÓN DEL SISTEMA 22
TABLA 2 ESPECIFICACIÓN DE REQUISITOS FUNCIONALES. 35
TABLA 3 ESPECIFICACIÓN DE REQUISITOS NO FUNCIONALES. 37
TABLA 4 HISTORIA DE USUARIO ADICIONAR USUARIO 38
TABLA 5 HISTORIA DE USUARIO MODIFICAR GRUPO 40
TABLA 6 HISTORIA DE USUARIO DETENER PROCESO. 41
TABLA 7 HISTORIA DE USUARIO MODIFICAR REPOSITORIO 42
TABLA 8 EJECUCIÓN DE LA TÉCNICA CAMINO MÍNIMO AL MÉTODO LOADSYSTEMPROCESS. 52
TABLA 9 CAMINOS OBTENIDOS DURANTE LA PRUEBA DE UNIDAD. 55
TABLA 10 CASO DE PRUEBA DE UNIDAD PARA EL CAMINO 1. 55
TABLA 11 CASO DE PRUEBA DE UNIDAD PARA EL CAMINO 2. 56
TABLA 12 CASO DE PRUEBA DE UNIDAD PARA EL CAMINO 3. 56
TABLA 13 CASO DE PRUEBA DE UNIDAD PARA EL CAMINO 4. 56
TABLA 14 CASO DE PRUEBA DE UNIDAD PARA EL CAMINO 5. 57
TABLA 15 CASO DE PRUEBA DE UNIDAD PARA EL CAMINO 6. 57
TABLA 16 DESCRIPCIÓN DE LAS VARIABLES PARA LA PRUEBA DE VALIDACIÓN. 59
TABLA 17 ESCENARIO 1.1 ADICIONAR USUARIO CON LOS VALORES OBLIGATORIOS. 60
TABLA 18 ESCENARIO 1.2 ADICIONAR USUARIO CON LOS VALORES OBLIGATORIOS VACÍOS. 61
TABLA 19 ESCENARIO 1.3 ADICIONAR USUARIO CON LOS VALORES OBLIGATORIOS INCORRECTOS. 61
TABLA 20 ESCENARIO 1.4 ADICIONAR USUARIO CON LOS VALORES OPCIONALES INCORRECTOS. 62

Introducción

El término GNU/Linux es utilizado para referirse a la combinación del *kernel*[1] Linux con el sistema GNU (1). Un sistema operativo GNU/Linux está conformado por varios programas que se encargan de comunicar y recibir instrucciones de los usuarios, tales como: leer y escribir datos en el disco duro, controlar el uso de la memoria y ejecutar otros programas. Este sistema está distribuido bajo los términos de la licencia GPL[2], que permite al usuario libertades como: copiar, estudiar, modificar y redistribuir el sistema operativo (2).

GNU/Linux está definido como un sistema multitarea y multiusuario. Multitarea se refiere a que es capaz de ejecutar varios procesos[3] de forma paralela y multiusuario indica la capacidad que posee para combinar varios usuarios al mismo tiempo.

El sistema operativo, además de otras funciones, permite administrar las cuentas de usuarios al delimitarlos en grupos con las mismas libertades. Al mismo tiempo, se encarga de la gestión de los procesos y el manejo de las actualizaciones del sistema mediante la administración de los repositorios.

Un usuario de GNU/Linux debe tener una cuenta de usuario en el sistema que establezca sus privilegios, y les permita ser organizados en grupos de forma que se puedan establecer privilegios a un determinado grupo de trabajo para el acceso a los recursos del sistema (3).

Como parte fundamental de la gestión del sistema, GNU/Linux permite la administración de los procesos, estos son programas independientes que interactúan a través de mecanismos de comunicación dados por el sistema y contienen información de cambios (4). Este sistema cuenta con dos tipos de procesos, los que se ejecutan en modo kernel y en modo usuario y se pueden encontrar en distintos estados, como pueden ser: preparado, ejecución, suspendido, detenido y zombi, y se definen en dependencia del uso del procesador que les sea otorgado. Los procesos del sistema pueden crearse a partir de:

---

[1] Kernel: También conocido como núcleo. Software que constituye parte fundamental del sistema operativo y se ejecuta en modo privilegiado.

[2] GPL: del inglés *General Public License*, Licencia publica General de GNU, su objetivo es declarar que el software cubierto por esta licencia es software libre.

[3] Proceso: Programa o tarea ejecutándose en la computadora para procesar alguna instrucción.

- el arranque del sistema.
- una llamada al sistema realizada por un proceso con el objetivo de instanciar un nuevo proceso.
- una petición deliberada del usuario para crear un proceso.
- el inicio de un trabajo por lotes.

Los repositorios en GNU/Linux son grandes bancos de datos o servidores que alojan las aplicaciones del sistema. Un repositorio de paquetes se encarga de distribuir software y realizar actualizaciones directamente a los usuarios, con el uso de herramientas como apt[4]. Los repositorios pueden ser diseñados de dos tipos, de software privativo o de código abierto, aunque en cualquiera de los casos, deben cumplir con las características de: interoperabilidad, autenticación y autorización de usuarios, interfaz de búsqueda a texto completo y localización permanente de las aplicaciones (5). Los repositorios pueden contener aplicaciones con diferentes fines como son: desarrollo, oficina, internet e instalación y configuración de servicios telemáticos.

Un servicio telemático es una entidad geográficamente distribuida, que provee a un número de personas, un conjunto de facilidades para cubrir un rango de necesidades de información y comunicación, al utilizar los recursos existentes y futuros de las redes de telecomunicaciones. GNU/Linux es uno de los sistemas operativos más utilizados para ofrecer estos servicios por las ventajas que ofrece, sin embargo, la configuración y administración de los servicios telemáticos en GNU/Linux es una tarea compleja (6).

Los servicios telemáticos son herramientas que pueden ser instaladas desde los repositorios públicos de las distribuciones de GNU/Linux, aunque pueden existir algunas que sean distribuidas por diferentes compañías en repositorios propios. *Samba4*[5] tiene como proveedor a la compañía *SerNet*, quien se encarga de construir paquetes para Debian desde el código fuente, pero a partir del lanzamiento de la versión 4.3 decide cambiar su estrategia de distribución y limitar la forma de adquirir estas actualizaciones, al ser posible solo a través de su tienda.

Al encontrarse en ejecución un servicio telemático el sistema operativo lo maneja como un proceso y es necesario gestionarlo para analizar su capacidad de cambiar de estado y prevenir comportamientos que

[4] APT: del inglés *Advanced Packaging Tool*, Herramienta Avanzada de Empaquetado, es un sistema de gestión de paquetes para la instalación y eliminación de paquetes.

[5] Samba4: es una herramienta utilizada para permitir la interoperabilidad entre servidores GNU/Linux y clientes Windows.

provoquen el bloqueo del sistema. Además de conocer el uso de los recursos del sistema operativo, se necesita visualizar los procesos que están en ejecución, cuántos están activos, los recursos que consume un servicio, el espacio en memoria que utiliza y qué usuario está maneja cada proceso. Por otro lado, los usuarios y grupos del sistema operativo pueden poseer diferentes permisos sobre determinados procesos del sistema, lo que hace necesario definir cuál de ellos tiene acceso en el sistema a determinado servicio para no comprometer la seguridad.

En Cuba, la soberanía tecnológica representa un imperativo ético y moral para el desarrollo económico y social, en este sentido el país trabaja en la migración hacia los sistemas de plataformas libres cuyo objetivo primario es garantizar la disponibilidad, fiabilidad y seguridad de los datos, así como lograr una mayor estabilidad y reducción de costos por licencias. Esta línea de trabajo es legitimada gubernamentalmente en el acuerdo 084 del Consejo de Ministros de la República de Cuba en el 2004, en él se estipula una migración paulatina de los Organismos de la Administración Central del Estado (OACE) hacia aplicaciones de código abierto (7).

La Universidad de las Ciencias Informáticas (UCI) cuenta con centros de desarrollo de servicios y aplicaciones para contribuir al soporte de la industria informática en el país, y vincula simultáneamente a los estudiantes en el proceso de formación y producción. El Centro de Software Libre (CESOL) es responsable de los procesos de migración hacia soluciones libres y de código abierto de los sistemas operativos utilizados en los OACE. El departamento de Servicios Integrales en Migración, Asesoría y Soporte (SIMAYS), perteneciente a CESOL, está especializado en ofrecer servicios de migración, asesoría y soporte a tecnologías de software libre y código abierto.

Con el fin de apoyar a los especialistas en los procesos de migración a código abierto y a los administradores de redes de los OACE, se desarrolla la Herramienta de Migración y Administración de Servicios Telemáticos (HMAST). HMAST tiene como objetivo administrar los servicios telemáticos en sistemas GNU/Linux y está conformada por los módulos Apache[6], Bácula[7], DHCP[8], MySQL[9] y SSH[10]. Esta

[6] Apache: servidor *web* HTTP de código abierto para plataformas Unix.

[7] Bácula: Colección de herramientas de respaldo de equipos en redes IP.

[8] DHCP: del inglés *Dynamic Host Configuration Protocol,* protocolo de configuración dinámica de host.

[99] MySQL: Sistema de gestión de bases de datos relacional.

herramienta permite la administración de servidores distribuidos por módulos que agrupan uno o varios servicios telemáticos. La administración de servidores permite crear una conexión segura con un usuario del sistema operativo con los privilegios suficientes para administrar un determinado servicio. Durante la puesta en práctica del proceso de migración se pudo comprobar que era necesario que HMAST permitiera:

- La gestión de usuarios y grupos del sistema operativo, debido a que la conexión que se realiza desde la herramienta al servidor tiene que ser desde un usuario creado con anterioridad en el sistema.
- Gestionar los procesos del sistema operativo para evitar un fallo del mediante el desempeño incorrecto de alguno de ellos.
- Mantener un control de los repositorios del sistema operativo, al permitir que el administrador de la herramienta pueda añadir, modificar o eliminar los repositorios.

En el proceso de migración, la gestión de usuarios, grupos, procesos y repositorios del sistema operativo se realiza de forma manual a través de una terminal y directamente en el servidor, trayendo consigo deficiencias tales como: demoras en el proceso de configuración, introducción de errores humanos por desconocimiento de los sistemas GNU/Linux por parte de los administradores, lentitud durante la ejecución de las tareas y, por consiguiente, un aumento de la complejidad del proceso de migración.

Debido a la problemática existente se define como problema científico de la investigación ¿Cómo garantizar la gestión de los usuarios, grupos, procesos y repositorios en los sistemas GNU/Linux durante el proceso de migración de los servicios telemáticos, desde HMAST?

Enfocándose en la solución al problema planteado, el objeto de estudio se define como la gestión de usuarios, grupos, procesos y repositorios en los sistemas operativos GNU/Linux, enmarcándose en el campo de acción la gestión de usuarios, grupos, procesos y repositorios en los sistemas operativos GNU/Linux desde HMAST.

[10] SSH: del inglés *Secure Shell*, Intérprete de órdenes seguro.

El objetivo general de la presente investigación es: Desarrollar un módulo para HMAST que permita la gestión de usuarios, grupos, procesos y repositorios del sistema operativo GNU/Linux y facilite el proceso de migración de servicios telemáticos en las organizaciones del país. A partir de esto se plantean los siguientes objetivos específicos:

1. Sistematizar en los sistemas operativos GNU/Linux, la gestión de usuarios, grupos, procesos y repositorios.
2. Diseñar las funcionalidades que permitan la gestión de usuarios, grupos, procesos y repositorios.
3. Implementar la solución propuesta.
4. Realizar las pruebas al módulo implementado.

Como vía al cumplimiento de estos objetivos, se definen las siguientes tareas de investigación:

1. Caracterización de los referentes teóricos que sustentan la gestión de los usuarios, grupos, procesos y repositorios en los sistemas GNU/Linux para analizar su ejecución en la propuesta de solución.
2. Análisis de sistemas informáticos que realizan la gestión del sistema en GNU/Linux para adquirir conocimiento respecto al manejo de los usuarios, grupos, procesos y repositorios del sistema operativo.
3. Sistematización de la arquitectura HMAST para permitir la integración del módulo.
4. Levantamiento y descripción de los requisitos para garantizar el cumplimiento de las necesidades de los clientes.
5. Implementación del módulo para la gestión de usuarios, grupos, procesos y repositorios.
6. Ejecución de las pruebas funcionales.

A partir del desarrollo de la investigación surgen una serie de preguntas científicas que ayudan en el proceso de desarrollo:

- ¿Cuáles son los referentes teóricos que sustentan el desarrollo de la gestión de usuarios, grupos, procesos y repositorios en los sistemas operativos GNU/Linux?
- ¿Cuáles son las características y componentes de las herramientas, tecnologías y metodologías existentes que permiten la construcción del módulo para la gestión de usuarios, grupos, procesos y repositorios en los sistemas operativos GNU/Linux en HMAST?

- ¿Cuál es la estructura que poseen los elementos que deben tenerse en cuenta para realizar el análisis y diseño del módulo de gestión de usuarios, grupos, procesos y repositorios en los sistemas operativos GNU/Linux desde HMAST?

- ¿Cómo contribuye la propuesta para la gestión de usuarios, grupos, procesos y repositorios en los sistemas operativos GNU/Linux desde HMAST a facilitar el proceso de migración de servicios telemáticos en los organismos nacionales?

Durante el desarrollo de la investigación se utilizan los métodos teóricos y empíricos de la investigación científica que se exponen a continuación:

Analítico-Sintético: se pone en práctica durante el estudio de los elementos del problema planteado, para comparar verdades de razón y verdades de hecho[11] a través de diferentes fuentes bibliográficas para extraer valoraciones concernientes a la gestión de usuarios, grupos, procesos y repositorios en sistemas GNU/Linux, además de la realización de resúmenes que tributan al desarrollo de la investigación.

Histórico-Lógico: se utiliza para el estudio de los elementos relacionados con la gestión de usuarios, grupos, procesos y repositorios desde herramientas desarrolladas, y a partir de los resultados identificar los elementos que inciden en los procesos de gestión.

Modelación: se emplea al crear abstracciones para representar la realidad y elaborar representaciones acerca de la administración de usuarios, grupos, procesos y repositorios desde la herramienta HMAST mediante el diseño de diagramas de clases.

Experimental: se utiliza para comprobar la integridad de la gestión de los usuarios, grupos, procesos y repositorios desde la herramienta HMAST una vez desarrollado el módulo.

El presente trabajo está estructurado en introducción, tres capítulos, conclusiones generales y referencias bibliográficas. El primer capítulo “La gestión de los sistemas operativos GNU/Linux. Descripción de HMAST” presenta la fundamentación teórica de la investigación a través del estudio del

[11] Gottfried Leibnitz refiriéndose a la distinción del método analítico-sintético.

estado del arte. Se analiza además la bibliografía correspondiente a partir de los métodos expuestos, se realiza una caracterización de la estructura de la herramienta HMAST y la metodología de desarrollo a utilizar.

En el segundo capítulo “Gestión de usuarios, grupos, procesos y repositorios del sistema desde HMAST” se especifican las características de la propuesta de solución y se definen las funcionalidades a desarrollar. Además, se presentan las historias de usuario, los patrones de diseño utilizados durante la implementación y se define la arquitectura del sistema.

En el tercer capítulo “Implementación y pruebas de la gestión de usuarios, grupos, procesos y repositorios desde HMAST” se documentan artefactos asociados a la implementación del módulo y se describen, diseñan, realizan y controlan los casos de prueba del sistema.

Capítulo 1: La gestión de los sistemas operativos GNU/Linux. Descripción de HMAST.

En el presente capítulo se realiza un análisis de cómo los sistemas GNU/Linux llevan a cabo la administración de usuarios, grupos, procesos y repositorios. Además, se analizan herramientas que se encargan de la gestión del sistema operativo. Se describen características de la herramienta HMAST tales como: arquitectura y aspectos más significativos a tener en cuenta para la integración del módulo. Como parte de la solución propuesta se definen las herramientas y tecnologías que se emplean durante el proceso de desarrollo.

1.1 Gestión de usuarios, grupos, procesos y repositorios desde GNU/Linux

La gestión de usuarios, grupos, procesos y repositorios de un sistema GNU/Linux necesita ser analizada de forma diferenciada, debido a que el sistema gestiona los usuarios y los grupos de forma similar, y trata a los procesos y repositorios de forma independiente. La gestión de estos elementos solo puede ser realizada por un usuario administrador, a través de la utilización de distintos comandos (8).

1.1.1 Gestión de usuarios

GNU/Linux es un sistema multiusuario, por lo que la tarea de añadir, modificar, eliminar usuarios se convierte en algo importante. Esta gestión se considera como un elemento de seguridad, que mal administrado o tomado a la ligera, puede convertirse en una gran falla de seguridad.

Los usuarios en GNU/Linux se identifican por un número, (*User* ID, UID) y pertenecen a un grupo principal de usuario, así como a otros grupos secundarios, identificados también por un número único de grupo, (*Group* ID, GID). El usuario puede pertenecer a más grupos además del principal, y como parte de las instrucciones que se pueden utilizar para la gestión de estos se encuentran:

Para adicionar un usuario se utiliza el comando **useradd,** que permite añadir un usuario y definir como parámetros la información del mismo, además de que permite crear el usuario en una misma línea de comando, su sintaxis es:

Crear usuario: **useradd [opciones] nombre-usuario**

Entre las **opciones** más destacadas se tienen:

-u: asigna un identificador al usuario.

-g: especifica el grupo principal que se quiere que tenga el usuario.

-G: asigna grupos secundarios a los que puede pertenecer el usuario.

-d: crea la carpeta home del usuario, si no se especifica suele ser */home/nombre-usuario*.

-s: asigna un intérprete de comandos (*shell* o consola) del usuario, si no se define otro el valor es */bin/bash*.

-c: permite añadir un comentario o descripción del perfil.

-r: permite crear un usuario del sistema con identificador menor de 500.

-p: establece la contraseña del usuario.

Para modificar un usuario se utiliza el comando **usermod,** que permite cambiar los parámetros que fueron definidos con anterioridad o asignar nuevos valores no definidos. Esta instrucción se ejecuta junto al nombre de usuario que se desea modificar, su sintaxis es:

Modificar usuario: **usermod [opciones] nombre-usuario**

Para eliminar un usuario se utiliza el comando **userdel** seguido del nombre del usuario. Esta instrucción junto a la opción **–r** indica que se elimina también su carpeta home, siempre que el usuario no se encuentre activo en el sistema durante el proceso de eliminación (9). Este comando se puede ejecutar además con la opción –f, que garantiza la eliminación del usuario, aunque este ejecute alguna acción en el sistema, su sintaxis es:

Eliminar usuario: **userdel [opciones] nombre-usuario**

1.1.2 Gestión de grupos

Los grupos son expresiones lógicas de organización que reúnen usuarios para un propósito común. Los usuarios dentro de un mismo grupo pueden leer, escribir o ejecutar archivos que pertenecen a ese grupo. En GNU/Linux, los grupos se gestionan a partir de los siguientes comandos:

groupadd: permite añadir un grupo y definir como parámetro el nombre del grupo.

groupmod: permite modificar los parámetros de un grupo.

groupdel: permite eliminar el grupo, (si un usuario tiene dicho grupo como grupo primario, este comando no eliminará el grupo).

Para añadir un usuario a un grupo se utiliza el comando **useradd** seguido del nombre del usuario y del nombre del grupo al que se quiere añadir.

Para eliminar un usuario de un grupo se utiliza el comando **userdel** seguido del nombre del usuario y el nombre del grupo del que se desea eliminar (9).

1.1.3 Gestión de procesos

Un proceso es un programa o un comando en curso de ejecución en un sistema operativo. Los sistemas como GNU/Linux pueden llevar a cabo varios procesos al mismo tiempo y son conocidos como multitarea o multiproceso. A cada proceso el *kernel* del sistema operativo le asigna un identificador (PID[12]) y los almacena en una tabla de procesos, estos procesos se relacionan de forma jerárquica al formar un árbol de padre a hijo. Todos los procesos son hijos de otro proceso excepto el proceso *init*, con identificador 1, que no tiene padre y es el primer proceso que se crea al inicio del sistema (10). Además del identificador, el sistema operativo se encarga de determinar la prioridad inicial del proceso, asignarle recursos e insertarlo en la cola de procesos.

En el caso de la jerarquía de los procesos, se puede asumir de dos formas: a) el proceso que se crea hereda el entorno de ejecución de su padre o b) este se ejecuta independientemente, con un entorno totalmente diferente. En cuanto a los recursos físicos y lógicos asignados a los procesos se tiene una relación en la que:

- padre e hijo comparten todos sus recursos
- el padre comparte un subconjunto de recursos a sus hijos
- padre e hijo no comparten recursos

[12] PID: Identificador único del proceso.

En el entorno de ejecución el padre puede asignarles a sus hijos, además de sus recursos, parámetros iniciales que le ofrezcan ayuda en la realización de sus tareas. Para esto el proceso padre pude realizar sus tareas de forma concurrente con su proceso hijo o esperar a que sus hijos terminen para poder continuar.

Algunas de las instrucciones que se pueden utilizar para trabajar con los procesos en GNU/Linux son:

- ps aux: visualiza los procesos.
- ps –u *root*: visualiza los procesos del usuario root.
- ps –ef: para conocer el proceso padre de un proceso.
- ptrace: se encarga de proporcionar un medio por el que un proceso padre puede observar y controlar la ejecución de otro proceso.

Un proceso finaliza cuando termina de ejecutar su última declaración, luego se borra de las listas de procesos, y los recursos reutilizables de los que disponía vuelven al sistema, así como su espacio en memoria. Para la eliminación de los procesos del sistema se utilizan varias instrucciones:

- **exit**: termina un proceso y permite que los recursos asignados se liberen. Cuando un proceso padre llama la función **exit,** puede esperar la terminación de sus hijos al utilizar la instrucción **wait**, que muestra el identificador de sus hijos completados. Si el proceso padre ha terminado, a todos sus procesos hijos se les asigna un proceso **init** como nuevo padre, para evitar la eliminación en cascada.
- **Kill, killall, pkill:** se encarga de terminar la ejecución de un proceso, aunque no siempre es utilizado para ello, en otras ocasiones es empleado para enviar mensajes sencillos a los procesos ejecutándose en el sistema. Por defecto el mensaje que se envía es una señal de terminación (SIGTERM[13]) que solicita al proceso limpiar su estado y salir. La diferencia entre estas instrucciones radica en el argumento que se utiliza para ejecutar el comando, en el caso de kill se

[13] SIGTERM: Compuesto por el termino SIG (prefijo utilizado para nombres de señales), TERM (abreviación para terminar).

utiliza junto a una señal de finalización y el identificador del proceso que se desea eliminar. En el caso de killall, esta instrucción permite eliminar un proceso en base a su nombre, en lugar de su PID. Por ultimo pkill es utilizado tanto para finalizar un proceso por su identificador como por su nombre, esta instrucción permite utilizar expresiones regulares[14] para la terminación de un proceso, por lo que se puede utilizar el nombre completo o una parte de este.

- **sigint –2**: interrumpe un proceso, equivale a pulsar Ctrl+C.
- **sigstop –19**: detener un proceso, es igual que pulsar Ctrl+Z.

Un proceso suspendido o bloqueado no puede continuar su ejecución hasta ser reanudado por otro proceso. El sistema operativo se encarga de eliminar temporalmente los procesos con el objetivo de reducir la carga del sistema durante alguna situación crítica. Normalmente los periodos de suspensión de un proceso son pequeños, aunque también pueden existir suspensiones largas en las que es necesario liberar los recursos del proceso para poder utilizarlos. Reanudar un proceso implica reiniciarlo a partir del punto en el que se suspendió. Las instrucciones que se utilizan para suspender o reanudar un proceso son:

- **sleep:** suspende la ejecución de un proceso por un tiempo determinado.
- **renice**: cambia la prioridad de un proceso.
- **sigcount –18**: reanuda un proceso que se halla parado por ejemplo con sigstop (11).

Como parte de los estados que poseen los procesos se encuentran:

Ejecución: es un proceso que tiene el uso del procesador y que se encuentra activo en el sistema.

Bloqueado: es un proceso que se encuentra listo para usarse y solo está en espera de que el sistema le asigne tiempo del procesador.

[14] Expresión regular: forma de representar los lenguajes, construida utilizando caracteres del alfabeto sobre el que se define el lenguaje.

Zombi: es el proceso que terminó su ejecución, pero retiene aun recursos del sistema. Al terminar su ejecución envía una señal al proceso padre para que elimine su entrada de la tabla de procesos, pero si por alguna razón el proceso padre no recibe la comunicación este continua activo en el sistema.

Detenido: un proceso que se encuentra detenido en el sistema es aquel que no entra en el reparto del procesador, y solo pasa a bloqueado en el momento que recibe alguna señal que le permite continuar.

Suspendido: los procesos se encuentran en estado suspendido cuando se hallan en espera por algún tipo de evento que le permite continuar, en el momento que ocurre este evento el proceso pasa a formar parte del conjunto de procesos bloqueados (11).

GNU/Linux es un sistema multiproceso y se encarga de repartir de forma ordenada el uso del procesador. A partir de los estados anteriores, el sistema detalla fases que permiten definir prioridad respecto a los procesos que se encuentran en un mismo estado. Esta prioridad es definida por el sistema para los estados **suspendido** y **ejecución**, y permite garantizar una prioridad interna para los procesos de un mismo estado, además de permitir que se definan para un mismo proceso distintas combinaciones de estas fases.

s: denota los procesos que son líder de la sesión.

<: indica que un proceso tiene prioridad mayor a la normal.

L: refiere que el proceso tiene páginas bloqueadas en memoria.

l: define a los procesos que son multihilos.

N: indica que el proceso posee prioridad menor a la normal.

+: refiere que el proceso se ejecuta en primer plano.

1.1.4 Gestión de repositorios

Además de las aplicaciones que conforman al sistema GNU/Linux, se pueden instalar otras que permiten aumentar las funcionalidades. Para facilitar la instalación y administración de nuevas aplicaciones existen los paquetes, que representan una pieza de software que cumple determinada funcionalidad y pueden ser instalados de diversas formas.

Capítulo 1: La gestión de los sistemas operativos GNU/Linux. Descripción de HMAST.

Los repositorios de GNU/Linux son una colección de paquetes de programas de una distribución de GNU/Linux específica que contiene archivos binarios precompilados y paquetes de código fuente, que pueden ser descargados e instalados por los usuarios de la distribución correspondiente (12).

Los sistemas GNU/Linux tienen un archivo en el que se almacena la lista de repositorios a los que se puede acceder a través del gestor de paquetes[15] para instalar el software. Este archivo se encuentra en la ubicación */etc/nombredelgestor/*, donde nombre del gestor se refiere al gestor de paquetes del sistema. Por ejemplo, en Ubuntu se encuentra en */etc/apt/sources.list*, donde se puede añadir o eliminar repositorios manualmente. También se pueden manejar mediante interfaces gráficas como el centro de software de Ubuntu o Synaptic[16]. Para instalar un programa desde la línea de comandos de Ubuntu utilizamos: **sudo apt-get install nombre_paquete** (13).

La mayoría de las aplicaciones disponen de versiones **deb** y **rmp**:

- **deb**: se denomina a la extensión del formato de paquetes de software de la distribución de GNU/Linux Debian.
- **rmp**: se refiere al formato de paquetes de la herramienta de administración *Red Hat Package Manager*. Diseñado para instalar, actualizar, desinstalar, verificar y solicitar paquetes en la distribución *Red Hat Linux*.

Para la gestión de repositorios se debe editar el archivo **/etc/apt/sources.list,** que posee los componentes de los repositorios. Otro componente importante en los repositorios es **deb-src**, que permite el acceso a los paquetes de código fuente de las aplicaciones. Para el desarrollo de la presente investigación se utiliza la distribución Nova en su versión 5.0, la que posee la siguiente dirección de repositorio:

[15] Gestor de paquetes: herramienta que ayuda con la instalación, actualización, configuración y eliminación de paquetes en el sistema.

[16] Synaptic: gestor de paquetes de GNU/Linux.

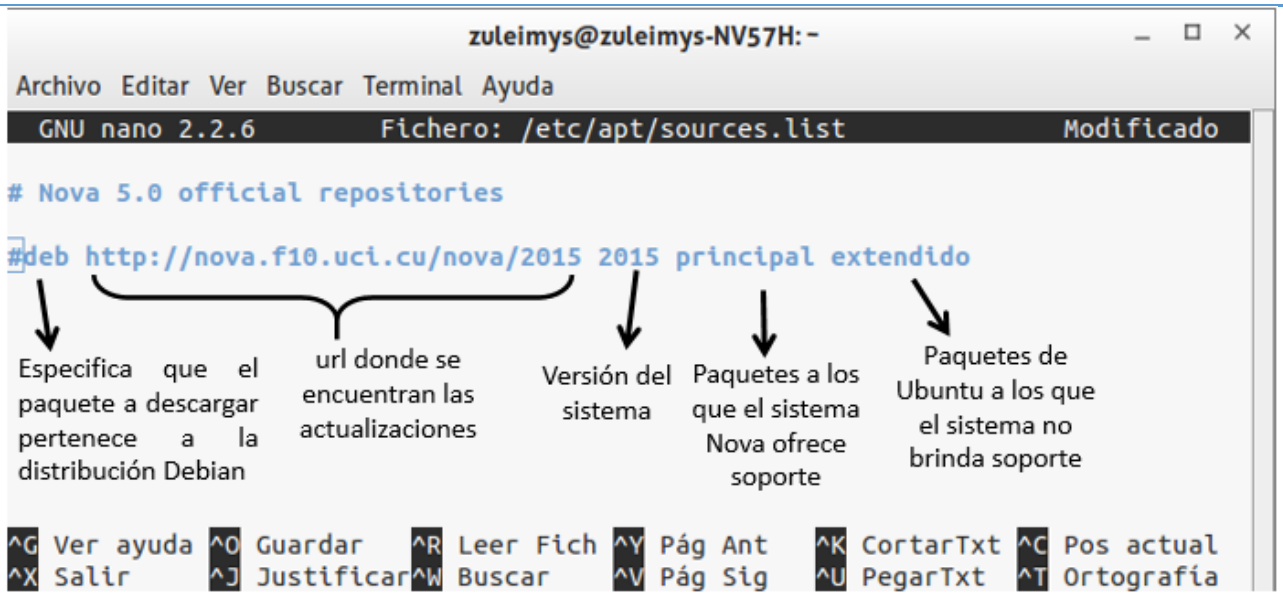


Figura 1 Dirección de repositorio GNU/Linux Nova.

1.2 Herramientas para la gestión de usuarios, grupos, procesos y repositorios

GNU/Linux como el resto de los sistemas operativos, necesita de herramientas que se encarguen de la gestión del sistema operativo, y a su vez de la gestión de los usuarios, grupos, procesos y repositorios que contiene. A continuación, se presentan algunas de esas herramientas:

users-admin

Para la administración de usuarios, GNU/Linux dispone de la herramienta gráfica users-admin, para acceder a la herramienta se ejecuta en la consola de root la instrucción users-admin, o si se inicia sesión como root se puede pulsar Alt+F2 y ejecutar users-admin. Esta herramienta presenta también una pestaña para gestionar grupos.

Figura 2 Interfaz de la herramienta users-admin.

cPanel

Panel de control utilizado para la administración de servidores de alojamiento web que proveen herramientas de automatización y una interfaz gráfica basada en páginas web. Es un software propietario desarrollado para ser compatible con la mayoría de las distribuciones de GNU/Linux. Fue concebido para ser instalado en un servidor dedicado[17] por el alto grado de integración con el sistema operativo, es válido resaltar que los desarrolladores no recomiendan desinstalar el programa, sino formatear el servidor (14). Cuenta entre sus funciones con un panel para la administración de usuarios y grupos del sistema operativo.

Plesk

Panel de control para la administración de sistemas, que se instala en los servidores para facilitar las tareas de gestión y administración de una cuenta. Permite a los usuarios con poca experiencia en gestión de servidores llevar a cabo tareas tales como: crear y modificar cuentas de correo electrónico, crear y administrar bases de datos, gestionar cuentas ftp, subir, modificar y eliminar archivos, crear carpetas,

[17] Servidor dedicado: PC que se utiliza para prestar servicios dedicados, generalmente relacionados con el alojamiento web y otros servicios en red.

entre otros. Su interfaz es muy sencilla y permite realizar cualquier tarea de gestión sin necesidad de ingresar una línea de código (15).

YaST

Herramienta de instalación y configuración de *openSUSE* y de la distribución *SUSE Linux Enterprise*. Es popular por su facilidad de uso, la interfaz gráfica atractiva y la capacidad de personalizar su sistema de forma rápida durante y después de la instalación. YaST es un acrónimo del inglés Yet another Setup Tool, lo que se podría traducir como: otra herramienta de configuración. Se puede utilizar para la configuración de hardware, red, servicios del sistema y el ajuste de las configuraciones de seguridad. Permite además la gestión de las cuentas de usuarios y grupos del sistema operativo a través del centro de control de YaST (16).

Zentyal

Se desarrolló con el objetivo de acercar Linux a las pymes[18] y permitirles aprovechar todo su potencial como servidor de empresa. Es la alternativa de código abierto a los productos de Microsoft para infraestructura TIC[19] en las pymes (Windows Small Business Server, Windows Server, Microsoft Exchange, Microsoft Forefront) y está basado en la distribución Ubuntu. Zentyal permite a profesionales de la informática y las comunicaciones, administrar todos los servicios de una red informática, tales como el acceso a Internet, la seguridad de la red, la compartición de recursos, la infraestructura de la red o las comunicaciones de forma sencilla y a través de una única plataforma. Posibilita la gestión de las cuentas de usuarios y grupos del sistema operativo. Posee una interfaz intuitiva que incluye únicamente aquellas funcionalidades de uso más frecuente, aunque también dispone de los medios necesarios para realizar toda clase de configuraciones avanzadas y todas sus funcionalidades están estrechamente integradas entre sí, al automatizar la mayoría de las tareas y ahorrar tiempo en la administración de sistemas (17).

Webmin

[18] Pymes: plural del acrónimo de Pequeña y Mediana Empresa (pyme).

[19] TIC: Tecnologías de la Información y las Comunicaciones.

Herramienta de configuración de sistemas accesible vía web para OpenSolaris, GNU/Linux y otros sistemas Unix. Permite configurar aspectos internos de muchos sistemas operativos, como: usuarios, cuotas de espacio, servicios, archivos de configuración, apagado del equipo, así como modificar y controlar muchas aplicaciones libres, tales como el servidor web Apache, PHP, MySQL, DNS, Samba, DHCP, entre otros.

Está construido a partir de módulos, los cuales tienen en una interfaz los archivos de configuración, esto hace fácil la adición de nuevas funcionalidades. Debido al diseño modular de Webmin, es posible para cualquier interesado escribir extensiones para configuración de escritorio. También permite controlar varias máquinas a través de una interfaz simple, o iniciar sesión en otros servidores Webmin de la misma subred o red de área local y permite la administración de las cuentas de usuarios y grupos del sistema operativo, así como la administración de los procesos (18).

Interfaz de línea de comandos

La terminal de línea de comandos (CLI por sus siglas en inglés) es una herramienta de configuración de sistemas accesible vía desktop. Permite a los usuarios dar instrucciones a algún programa mediante una línea de texto simple. Esta herramienta puede utilizarse interactivamente o a través de un archivo *batch*, leyendo ordenes desde un archivo de scripts. Aparece en todas las interfaces de escritorio como un método de ejecutar aplicaciones rápidamente. Las CLI son usadas por muchos programadores y administradores de sistemas como herramienta primaria de trabajo, especialmente en sistemas operativos basados en Unix; en entornos científicos y de ingeniería, y un subconjunto más pequeño de usuarios domésticos avanzados (19).

A partir del estudio de las herramientas antes mencionadas se puede definir que solo la Interfaz de línea de comandos permite la gestión de usuarios, grupos procesos y repositorios, por lo que el análisis de las restantes herramientas sirvió solo como referente teórico para conocer como llevan a cabo la gestión de los elementos del sistema. Con el objetivo de resumir la comparación de las herramientas, a continuación, se presenta una tabla que relaciona cada una de ellas con el componente a gestionar, y se especifica si gestiona o no cada elemento:

Tabla 1 Comparación de herramientas de gestión del sistema.

Herramienta	Usuarios	Grupos	Procesos	Repositorios
users-admin	✔	✔	✖	✖
cPanel	✔	✔	✔	✖
Plesk	✔	✔	✖	✖
YaST	✔	✔	✖	✖
Zentyal	✔	✔	✖	✖
Webmin	✔	✔	✔	✖
Interfaz de línea de comandos	✔	✔	✔	✔

1.3 Herramienta de Migración y Administración de Servicios Telemáticos

HMAST es una aplicación web encargada de la administración de servidores[20], esta herramienta hace posible la gestión de forma local o remota de los servicios telemáticos Apache, Bácula, DHCP, SSH y MySQL. Estos servicios se encuentran alojados en los servidores de los organismos nacionales, y se gestionan a través de conexiones seguras mediante el uso del protocolo SSH.

1.3.1 Arquitectura de la herramienta HMAST

La arquitectura de HMAST presenta un estilo N-Capas orientada al Dominio y su estructura se basa en cinco capas, las que interactúan a través de interfaces y utilizan inyección de dependencias (ver figura 3). Las características y principales responsabilidades de cada una de estas capas se describen a continuación:

[20] Servidor: PC que posee un programa informático que procesa una aplicación, realiza conexiones bidireccionales y/o unidireccionales y genera una respuesta en cualquier lenguaje o Aplicación que puede ser accedida por otra PC, conocida como Cliente.

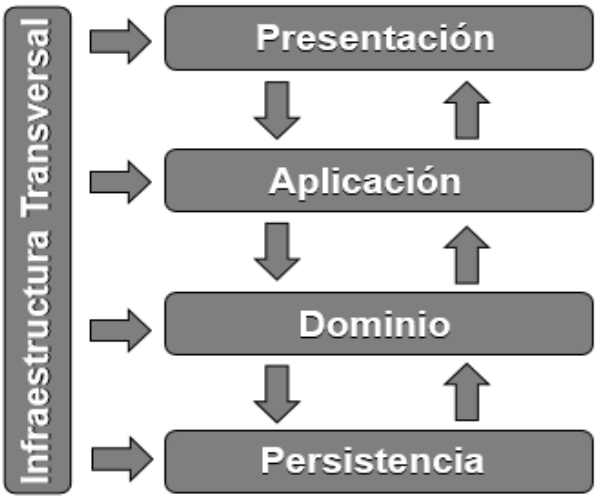


Figura 3 Arquitectura de HMAST.

Presentación: presenta al usuario los conceptos del negocio mediante una interfaz de usuario. Además, facilita la explotación de dichos procesos, informa sobre la situación del negocio y la implementación de las reglas de validación de dicha interfaz. En esta capa se realiza el tratamiento a las excepciones lanzadas desde capas inferiores.

Aplicación: realiza las llamadas a servicios de la capa inferior y tiene la responsabilidad de adaptar la información que recibe a los requisitos de los servicios de dominio. Está compuesta por los servicios de aplicación y los objetos para la transferencia de datos.

Dominio: implementa la lógica de dominio, o sea, las reglas del negocio. Define las interfaces de persistencia a datos, conocidos como contratos a repositorios, pero no los implementa. Es responsable de realizar las validaciones y sus componentes solo dependen de la capa **Infraestructura Transversal**. Esta capa está compuesta por las entidades del dominio, los servicios de dominio y los contratos de repositorio. Las entidades representan objetos del dominio y están definidas fundamentalmente por su identidad y continuidad en el tiempo. Los servicios de dominio contienen la lógica que trata a las entidades como un

todo y los contratos de repositorios son interfaces que especifican las operaciones que deben implementar los repositorios.

Persistencia: contiene el código necesario para persistir los datos. Se compone fundamentalmente por los repositorios, que son clases que implementan los contratos de repositorios definidos en la capa **Dominio**.

Infraestructura Transversal: promueve la reutilización de código. Contiene las operaciones de seguridad, inicio de sesión, monitoreo del sistema, mecanismos de persistencia reutilizables, validadores genéricos y todas aquellas operaciones que se puedan utilizar desde otras capas.

1.3.2 Aspectos importantes al implementar un módulo para HMAST

Las siguientes consideraciones se deben tener en cuenta para la integración de un módulo a la herramienta HMAST:

- La lógica de la capa **Aplicación** no debe incluir ninguna lógica de la capa **Dominio**, solo tareas de coordinación relativas a requerimientos técnicos de la aplicación, como conversiones de formatos de datos de entrada a entidades del **Dominio** y llamadas a componentes de **Infraestructura Transversal** para que realicen tareas complementarias.

- La capa **Presentación** no debe tener como entrada o salida objetos de dominio, sino objetos para la transferencia de datos.

- Las entidades solo pueden tener dependencias de los componentes de la capa **Dominio**.

- Las clases de servicios deben ser las únicas responsables de acceder a los repositorios, no se puede implementar código de persistencia a datos en la capa **Dominio**.

- Solo se puede acceder a la información almacenada en los servidores a través del uso de los repositorios.

- El código reutilizado por más de un repositorio debe estar disponible en la capa de **Infraestructura Transversal** (20).

1.4 Herramientas, tecnologías y metodología de software

Para el desarrollo del sistema se necesita definir una serie de elementos significativos, enfocados en estructurar y orientar el proceso de desarrollo. La metodología de software se define como un marco de trabajo utilizado para planificar y controlar el desarrollo del sistema. La utilización de herramientas y tecnologías permite ejecutar un proceso de desarrollo de software de calidad, basado en la innovación y la solución de necesidades.

1.4.1 Herramientas y tecnologías

Como solución al problema planteado en la introducción se hace necesario el estudio de las diferentes herramientas y tecnologías utilizadas en HMAST puesto que el presente trabajo constituye un módulo de dicha herramienta.

Framework

Un *framework* o marco de trabajo es una estructura conceptual y tecnológica que ofrece soporte, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio (20).

Spring

Spring Framework es un *framework* de código abierto para el desarrollo de aplicaciones Java, aunque cuenta también con una plataforma .NET. Por su diseño ofrece libertades a los desarrolladores y soluciones bien documentadas y fáciles de usar para las prácticas comunes. Cuenta como características principales las inyecciones de dependencias y la programación orientada a aspectos.

La inyección de dependencias es un patrón de diseño orientado a objetos en el que se suministran objetos a una clase en lugar de ser creados en la propia clase, tiene como objetivo lograr un bajo acoplamiento entre los objetos de la aplicación y permite que la comunicación de los componentes sea realizada mediante parámetros o paso de mensajes. La Programación Orientada a Aspectos (AOP) es un ejemplo de programación relativamente reciente ya que permite una adecuada modularización de las aplicaciones y posibilita una mejor separación de conceptos. Permite capturar los diferentes conceptos que componen una aplicación y separarlos en entidades bien definidas, elimina las dependencias entre cada uno de los módulos, así como la dispersión del código, por lo que sus implementaciones resultan más comprensibles, adaptables y reusables (21).

Bootstrap

Bootstrap es un framework desarrollado y liberado por Twitter que tiene como objetivo facilitar el diseño web. Permite crear de forma sencilla webs de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla y siempre se vean igual de bien. Es *Open Source* o código abierto, por lo que se puede usar de forma gratuita y sin restricciones (22).

Lenguaje de programación

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un programa informático. Permite a un programador especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser almacenados y transmitidos estos, además de qué acciones se debe tomar bajo una variada gama de circunstancias (23).

Java

Java fue diseñado como un lenguaje orientado a objetos, los cuales agrupan en estructuras encapsuladas tanto sus datos como las funcionalidades. Proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, al facilitar la creación de aplicaciones distribuidas. Es interpretado y compilado a la vez. Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de

memoria liberan a los programadores de una familia entera de errores. Soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas (24).

Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE[21]) es un programa que permite a los desarrolladores escribir y ejecutar código a través de un lenguaje de programación y un conjunto de herramientas. Estos pueden soportar más de un lenguaje de programación o uno específicamente (20).

IntelliJ IDEA

Es un IDE para el lenguaje de programación Java. Es ligero en el diseño y ofrece características útiles como pruebas de Junit, depuración, inspección de código y soporte para múltiple refactorización. IntelliJ Idea se encuentra enfocado en elevar la productividad de los desarrolladores por lo que ofrece soporte avanzado para los frameworks y estándares más importantes en el desarrollo web como son Spring Framework, Vaadin, Play, Grails, Web Services y Struts. Además, incluye la asistencia de código para lenguajes como HTML, CSS, JavaScript, Node.js y TypeScript. Favorece la integración que tiene el IDE con herramientas para la construcción de proyectos como Maven, Ant y Gradle; y para el control de versiones como Git, SVN y Mercurial. Incorpora también un editor de base de datos SQL con soporte completo SQL para Oracle, PostgreSQL, MySQL y SQL Server. (25)

Herramientas CASE

Las herramientas CASE[22] son aplicaciones informáticas que tienen el propósito de aumentar la productividad en el desarrollo de software, disminuyendo el costo de las mismas en términos de dinero y tiempo. Pueden contribuir en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el cálculo de costos, la compilación automática, el proceso de realizar un diseño del proyecto, la

[21] IDE: del inglés *Integrated Development Environment*, Entorno de Desarrollo Integrado.

[22] CASE: Del inglés *Computer Aided Software Engineering*, Herramienta de Ingeniería de Software Asistida por Computadora.

implementación de parte del código automáticamente con el diseño dado, la documentación o la detección de errores, entre otras (26).

Visual Paradigm

Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software. Permite el modelado visual UML y propicia la construcción de aplicaciones de calidad. Es una herramienta colaborativa potente y fácil de usar, pues soporta múltiples usuarios que trabajan en el mismo proyecto. Con el uso de esta herramienta se pueden generar todo tipo de diagramas, el código a partir de estos diagramas, así como la documentación.

Esta herramienta se caracteriza por: ofrecer soporte para varias versiones de UML, una versión gratuita y otra comercial, permitir la ingeniería inversa (código a modelo, código a diagrama), permitir la generación de código (modelo a código, diagrama a código), permitir exportar el diseño de la base de datos para varios gestores y ser un producto multiplataforma (27).

Lenguaje Unificado de Modelado

UML es un lenguaje estándar utilizado para escribir planos de software. Puede utilizarse para visualizar, especificar, construir y documentar los artefactos y las relaciones entre ellos, que se crean durante el desarrollo del software. Este lenguaje se emplea para el diseño del módulo, específicamente para la construcción de los diagramas de clases por paquetes y el diagrama de despliegue (28).

ForeUI

ForeUI es una herramienta de creación de prototipos de interfaz de usuario diseñada para crear *mockup*[23], *wireframe*[24] o prototipos para cualquier aplicación o sitio web. Con ForeUI, un prototipo de proyecto es

[23] Mockup: Se refiere al bosquejo que permite a los diseñadores mostrar como quedaran sus diseños.

[24] Wireframe: Es un esquema de página o plano de pantalla que representa en esqueleto de un sitio web.

skinnable[25], ya que puede cambiar fácilmente su estilo simplemente al editar el tema de la interfaz de usuario. Permite diseñar el comportamiento de prototipos y define diagramas de flujo intuitivos para manejar eventos específicos. Su prototipo puede ser exportado a imágenes *wireframe*, documentos PDF o simulación HTML5. Es posible utilizar ForeUI como IDE para desarrollar front-end[26] de aplicaciones web reales. ForeUI funciona en sistemas Windows, Mac OS X y Linux (29).

Sistema de Control de versiones

Un sistema de control de versiones es una combinación de tecnologías y prácticas para seguir y controlar los cambios realizados en los ficheros del proyecto, en particular en el código fuente, en la documentación y en las páginas web (30).

Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensado para la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones que poseen un gran número de archivos de código fuente. Entre sus características más relevantes se encuentran: la rapidez en la gestión de ramas y mezclado de versiones, la publicación de los almacenes de información por HTTP, FTP, *rsinc* o mediante algún protocolo nativo, ya sea una conexión TCP/IP simple o a través de cifrado SSH, además de poseer una gestión eficiente de proyectos grandes y un realmacenamiento periódico en paquetes (31).

Maven

Maven es una herramienta para la gestión y construcción de proyectos Java, con un modelo de configuración de construcción simple basado en un formato XML[27], perteneciente a la Apache Software Fundation, liberado bajo la licencia Apache 2.0. Maven utiliza un Project Object Model (POM) para

[26] Skinnable: Se refiere a los estilos visuales, es además un paquete de apariencia gráfica personalizado mediante el uso de una interfaz gráfica de usuario (GUI).

[26] Front-end: Se refiere a la parte que interactúa con los usuarios, es visual de las aplicaciones.

[27] XML: del inglés eXtensible Markup Language, traducido como Lenguaje de Marcado Extensible, utilizado para almacenar datos de forma legible.

describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos *Open Source* en Java, de Apache y otras organizaciones y desarrolladores. Maven provee soporte no solo para obtener archivos de su repositorio, sino también para subir artefactos al repositorio al final de la construcción de la aplicación, dejándola al acceso de todos los usuarios (32).

JQuery

JQuery es una biblioteca de JavaScript que permite simplificar la forma de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y utilizar Ajax para agregar interacción. Con una combinación de versatilidad y extensibilidad (33).

HTML

HTML, "Lenguaje de Marcado de Hipertexto" por sus siglas en inglés *HyperText Markup Language*, es un lenguaje que pertenece a la familia de los "lenguajes de marcado" y es utilizado para la elaboración de páginas web. El estándar HTML lo define la W3C (*World Wide Web Consortium*) y actualmente HTML se encuentra en su versión 5. HTML no es un lenguaje de programación ya que no cuenta con funciones aritméticas, variables o estructuras de control propias de los lenguajes de programación, por lo que HTML genera únicamente páginas web estáticas, sin embargo, HTML se puede usar en conjunto con diversos lenguajes de programación para la creación de páginas web dinámicas (34).

Augeas

Augeas es una herramienta de edición de configuración. Analiza archivos de configuración en sus formatos nativos y los transforma en un árbol. Los cambios de configuración se realizan al manipular este árbol y guardarlo en archivos de configuración nativos.

Augeas es un API proporcionado por una biblioteca C, además de una herramienta de línea de comandos para manipular la configuración desde el *Shell* que cuenta con representaciones de árboles canónicos de archivos de configuración comunes y posee un lenguaje específico de dominio para describir los formatos de archivo de configuración. Tiene como objetivos manipular los archivos de configuración de forma

segura, proporcionar una API de configuración local para Linux y facilitar la integración de nuevos archivos de configuración en el árbol Augeas (35).

1.4.2 Metodología de desarrollo de Software

Una metodología de desarrollo de software es el entorno que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de software. Todos los proyectos cuentan con una metodología de software, pero no todas las metodologías son aplicables a cualquier tipo de proyectos, por lo que cada proyecto se adapta a una metodología específica (36).

Metodología de Software AUP[28]

AUP es una versión simplificada del Proceso Unificado Racional (RUP por sus siglas en inglés). Describe de manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio, al utilizar técnicas agiles y conceptos que aún se mantienen en RUP. AUP aplica técnicas agiles que incluyen el desarrollo dirigido por pruebas, el modelado ágil, la gestión de cambios ágil y la refactorización de Base de datos para mejorar la productividad.

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el Modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. Como las metodologías de desarrollo se adaptan a las características de cada proyecto, se decide realizar una variación de la metodología AUP para la actividad productiva de la UCI.

Variación de AUP para la UCI

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio y modificar el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola llamada Ejecución y se agrega la fase de Cierre.

- **Primera fase**: Inicio, donde se llevan a cabo las actividades relacionadas con la planeación.

- **Segunda fase**: Ejecución, aquí se confeccionan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto, tiene en cuenta, además, los requisitos y la arquitectura.

- **Tercera fase**: Cierre, donde se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales del cierre del proyecto.

AUP define 7 disciplinas:

1. **Modelado del negocio**. El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se aborda en el proyecto, y determinar una solución viable para resolver el problema de dominio. Agrupa los flujos de trabajos de Modelado de negocio, Requisitos y Análisis y Diseño.

2. **Implementación**. El objetivo de esta disciplina es transformar su modelo en código ejecutable.

3. **Pruebas**. El objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificar que se cumplan los requisitos.
4. **Despliegue**. El objetivo de esta disciplina es la prestación y ejecución del sistema y que el mismo este a disposición de los usuarios finales.

5. **Gestión de configuración**. El objetivo de esta disciplina es la gestión de acceso a herramientas de su proyecto. Esto incluye no sólo el seguimiento de las versiones con el tiempo, sino también el control y gestión del cambio para ellos.

6. **Gestión de proyectos**. El objetivo de esta disciplina es dirigir las actividades que se lleva a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc.), coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que es entregado a tiempo y dentro del presupuesto.

---

[28] AUP: del inglés Agile Unified Process, Proceso Unificado Ágil.

7. **Entorno**. El objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario.

El proceso de desarrollo del módulo es guiado por la metodología AUP en su variación para la UCI. A partir de que no se realiza un modelado del negocio, se utiliza el escenario 4 que establece esta metodología y se describen las funcionalidades a desarrollar en un documento de Especificación de Requisitos de Software (36).

Conclusiones parciales

El estudio de los conceptos y características de la gestión de usuarios, grupos, procesos y repositorios del sistema permitió comprender como funcionan y que estructura poseen estos elementos en los sistemas GNU/Linux. Luego de caracterizar las herramientas que se encargan de la gestión del sistema, se selecciona la Terminal de líneas de comandos de GNU/Linux y sus herramientas para llevar a cabo el desarrollo del módulo propuesto. Se define la arquitectura, metodología, tecnologías y herramientas a emplear durante el diseño e implementación a partir de los elementos empleados en la HMAST para garantizar la integración del módulo.

Capítulo 2: La gestión de usuarios, grupos, procesos y repositorios del sistema desde HMAST

A partir de las herramientas, metodologías y tecnologías definidas con anterioridad, en el presente capítulo se describe la propuesta de solución. A continuación, se especifican las características y funcionalidades del módulo a desarrollar a través de las historias de usuario.

2.1 Descripción general de la propuesta de solución de la investigación

Se propone el desarrollo de un módulo que realice una gestión básica del sistema operativo centrándose en la gestión de usuarios, grupos, procesos y repositorios del servidor desde la herramienta HMAST. El desarrollo del módulo se propone para facilitar el manejo de estos elementos por parte de los administradores de red. La aplicación permite, mediante una conexión SSH, gestionar servidores remotos y proporciona las funcionalidades de gestión del sistema a partir de una interfaz gráfica. Las funcionalidades permiten gestionar las cuentas de usuarios y grupos a los que pertenecen, posibilitan manejar los procesos del sistema y definir los repositorios para las actualizaciones, tanto del sistema operativo como de los servicios telemáticos.

2.2 Descripción de requisitos del sistema

Los requisitos se definen como una capacidad que tiene que poseer un sistema o componente de este para satisfacer un contrato, estándar u otro documento impuesto formalmente, además de ser una condición que necesita un usuario para resolver un problema o lograr un objetivo. Estos requisitos pueden ser clasificados en funcionales y no funcionales (37).

2.2.1 Requisitos funcionales

Los requisitos funcionales detallan las funciones que el software debe ejecutar para lograr la interacción entre el usuario y el proyecto de software. A continuación, se muestra el listado de requisitos, donde se presentan 29 requisitos funcionales definidos para la implementación del módulo. Estos son clasificados a partir de su prioridad en alta, media y baja. Como el cliente no define una prioridad de ejecución específica

para los requisitos, el analista se encarga de establecer un orden de ejecución para estos, en función de la necesidad y la dependencia entre ellos.

Tabla 2 Especificación de requisitos funcionales.

Prioridad		**Alta**
No.	**Nombre del requisito**	**Descripción**
RF1	Adicionar un usuario al sistema operativo	Permite adicionar un usuario al sistema, y tiene en cuenta: nombre, identificador, contraseña, entre otros.
RF2	Modificar un usuario del sistema operativo	Permite modificar un usuario creado previamente en el sistema y tiene en cuenta: nombre, identificador, contraseña, entre otros.
RF3	Eliminar usuario del sistema operativo	Permite eliminar un usuario creado previamente en el sistema.
RF4	Adicionar un grupo al sistema operativo	Permite adicionar un grupo al sistema y tiene en cuenta: identificador, nombre, entre otros.
RF5	Modificar grupo del sistema operativo	Permite modificar los atributos de un grupo que ya ha sido creado en el sistema operativo.
RF6	Eliminar grupo del sistema operativo	Permite eliminar un grupo del sistema operativo.
RF7	Reiniciar proceso	Permite reiniciar un proceso del sistema operativo.
RF8	Detener un proceso	Permite detener un proceso que se encuentre en ejecución.
RF9	Eliminar un proceso	Permite eliminar un proceso de la lista de procesos.
RF10	Añadir repositorio al sistema operativo	Permite añadir un repositorio al sistema y tiene en cuenta: tipo de paquete, url, versión del sistema, entre otros.
RF11	Modificar repositorio del sistema	Permite modificar los repositorios que posee el sistema operativo.
RF12	Eliminar repositorio del sistema operativo	Permite eliminar los repositorios que sean seleccionados por el usuario.

Prioridad		**Media**
RF13	Adicionar un usuario a grupos	Permite adicionar el usuario a uno o varios grupos.
RF14	Modificar grupos del usuario	Permite adicionar el usuario a uno o varios grupos nuevos.
RF15	Eliminar grupos del usuario	Permite eliminar el usuario de uno o varios grupos.
RF16	Adicionar usuarios a un grupo	Permite adicionar usuarios a un grupo del sistema.
RF17	Modificar usuarios de un grupo	Permite adicionarle al grupo uno o varios usuarios nuevos.
RF18	Eliminar usuarios de un grupo	Permite eliminar usuarios del grupo.
RF19	Comprobar cambios locales respecto al servidor	Permite comprobar si existen cambios locales en la base de datos respecto al servidor.
RF20	Aplicar cambios en el servidor	Permite aplicar en el servidor los cambios que se han realizado en la base de datos.
RF21	Descartar cambios locales	Permite descartar los cambios locales realizados en la herramienta.
RF22	Mostrar listado de usuarios del sistema operativo	Permite visualizar el listado de los usuarios del sistema operativo.
RF23	Mostrar listado de grupos del sistema operativo	Permite visualizar el listado de los grupos del sistema operativo.
RF24	Mostrar listado de procesos	Permite visualizar el listado de los procesos del sistema operativo.
RF25	Mostrar listado de repositorios del sistema operativo	Permite visualizar el listado de los repositorios que se encuentren en el sistema operativo.
Prioridad		**Baja**
RF26	Buscar un usuario del sistema operativo	Permite buscar un usuario en el listado de usuarios del sistema.
RF27	Buscar un grupo del sistema operativo	Permite buscar un grupo del listado de grupos del sistema operativo.

RF28	Buscar un proceso del sistema	Permite buscar un proceso en el listado de procesos del sistema operativo.
RF29	Buscar un repositorio del sistema	Permite buscar un repositorio del sistema operativo.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son cualidades que se le imponen al sistema en cuanto al diseño y la implementación. La metodología AUP en su variación para la UCI, propone en la ISO 25010, una asociación de estos requisitos a atributos de calidad (20). Algunos de estos requisitos no funcionales son definidos por la herramienta HMAST, debido a que condicionan el funcionamiento de la aplicación para lograr una correcta integración con el sistema base. Como requisitos no funcionales del sistema propuesto se encuentran:

Tabla 3 Especificación de requisitos no funcionales.

No.	Nombre del requisito	Atributo de calidad	Descripción
1	Utilizar Java como lenguaje de programación.	Funcionalidad	Restricciones que han sido heredadas del sistema HMAST y el módulo debe cumplir para lograr una correcta integración.
2	EL módulo debe ser ejecutado sobre el sistema operativo GNU/Linux Nova Servidores.		
3	Disponer en el sistema operativo GNU/Linux de los siguientes paquetes: libjna-java, openjdk-7-jdk, augeas, augeas-tools.		
4	EL módulo debe ser capaz de recuperarse en caso de fallos de ejecución a través de mecanismos especificados.	Confiabilidad	Durante el fallo de alguna funcionalidad del módulo, el resto de estas debe continuar su ejecución.
5	Proteger la información para que personas o sistemas no puedan leer o	Seguridad	Los datos que se transfieren entre la herramienta y el sistema

	modificar las contraseñas de los usuarios del sistema operativo sin autorización y permitir que las personas con acceso puedan manejarlas.		operativo estarán protegidos, así como los que sean almacenados temporalmente en el servidor.

2.3 Historias de usuario

Las historias de usuario especifican las tareas que debe realizar el sistema. Estas son escritas en lenguaje natural, sin formato predefinido, con una extensión de solo unas líneas. Las historias de usuario son utilizadas para estimar tiempos de desarrollo además de guiar la construcción de las pruebas de aceptación. Las historias de usuario que se describen a continuación son algunas que representan a varios requisitos de mayor prioridad en el desarrollo del módulo:

- Adicionar usuario.
- Editar grupo.
- Detener proceso.
- Editar repositorio.

Tabla 4 Historia de usuario Adicionar usuario.

Numero:1	**Nombre del requisito**: Adicionar usuario.
Programador: Zuleimys García Rodríguez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 20
Riesgo en desarrollo: Disponibilidad de la red.	**Tiempo real:** 20
Descripción: La funcionalidad comienza cuando usuario de HMAST selecciona la opción *Adicionar usuario,* esta despliega en una interfaz los campos donde se añaden los atributos con los que se crea el usuario. La instrucción que se ejecuta en la terminal es **useradd**. En la especificación de los atributos que posee el nuevo usuario se encuentran los parámetros: En la pestaña *Principal*:	

Nombre de usuario (obligatorio): el nombre de usuario puede contener hasta 32 caracteres, incluyendo mayúsculas y minúsculas, no puede contener espacios en blanco o alguno de los caracteres: ", + * ; :\ = ¿'><.

Identificador(opcional): este campo es un valor numérico, que puede ser definido por el usuario, en caso de que no ocurra el sistema añade un identificador por defecto, que corresponde al último identificador asignado más uno. En dependencia del número definido, se especifica si es un usuario con permisos administrativos o no. Si el valor del identificador es menor que 1000, es un usuario del sistema.

Descripción de la cuenta(opcional): establece una breve descripción del perfil, donde se añade el nombre completo o las características del puesto de trabajo. El parámetro que registra el identificador del usuario es **-u** seguido por la descripción.

Contraseña (obligatorio): este atributo establece una contraseña, para ello se registran los valores que introduce el usuario siempre que cumplan con la complejidad establecida en el sistema operativo. Para esto se ejecuta la instrucción -**p** seguido del nombre de usuario.

Confirmar contraseña (obligatorio): es un campo de texto en el que el usuario debe repetir la contraseña anterior y es necesario que coincidan los valores registrados.

En la pestaña *Adicional*:

Terminal (opcional), (por defecto: /bin/bash): es un botón que define si el usuario va a especificar alguna terminal o utiliza la que le asigna el sistema por defecto. Si se activa se habilita el atributo *Dirección de la terminal.*

Dirección de la terminal (obligatorio, si el botón terminal está seleccionado): es una cadena que especifica la terminal que tendrá el usuario. El parámetro que permite adicionar esta cadena es **–s.**

Crear *home* (opcional): (por defecto: home/nombre_usuario): es un botón que define si el usuario va a especificar alguna dirección de home o utiliza la que le asigna el sistema por defecto. Si desea que el usuario reciba una dirección de home por defecto, no debe marcar la opción.

Dirección del *home* (obligatorio si el botón crear home esta seleccionado): es una cadena que define la dirección del home del usuario si la dirección especificada es una ruta válida en el sistema. La instrucción que ejecuta las modificaciones a la carpeta home es **–m** seguido de la nueva dirección del home.

Observaciones: El usuario es adicionado al sistema operativo solo si se ejecuta la opción que aplica los cambios en el servidor.

Prototipo elemental de interfaz gráfica de usuario:

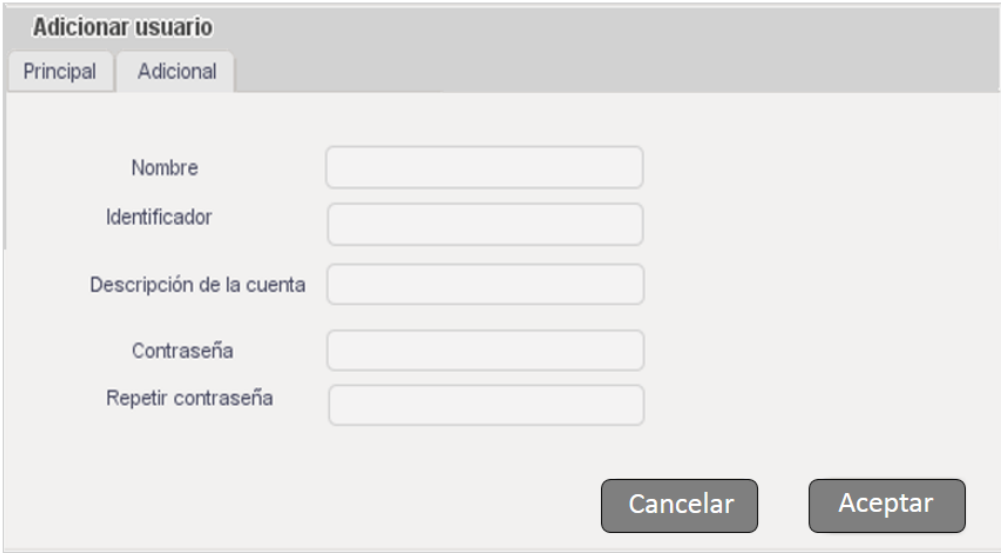


Tabla 5 Historia de usuario Modificar grupo.

Numero: 5	**Nombre del requisito**: Modificar grupo.
Programador: Zuleimys García Rodríguez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 10
Riesgo en desarrollo: Disponibilidad de la red.	**Tiempo real:** 10
Descripción: La funcionalidad comienza cuando usuario de HMAST selecciona en Gestión de grupos la opción *Editar,* la que despliega en una interfaz los atributos con los que el grupo será editado en el sistema. Los atributos que posee son: **Nombre del grupo** (opcional): permite definir cuál es el nuevo nombre del grupo. **Identificador** (opcional): permite asignar un nuevo valor numérico que será con el que se identifica el grupo. El atributo que modifica el identificador del grupo es **–g**. La instrucción que se ejecuta en la terminal es **groupmod –g** seguido por nombre del grupo.	

Usuarios del grupo: es un campo de selección donde se visualizan todos los usuarios del sistema y los usuarios que posee el grupo. Si desea que el grupo que se modifica contenga uno o varios usuarios nuevos deberá seleccionarlos de la lista de usuarios y enviarlos hacia la lista de Usuarios del grupo. Si desea que el usuario no contenga algún usuario de los que posee actualmente, debe seleccionarlos y acceder al botón eliminar que se encuentra en la lista de Usuarios del Grupo.
Observaciones: El grupo es modificado en el sistema operativo solo si se ejecuta la opción que aplica los cambios en el servidor.
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 6 Historia de usuario Detener proceso.

Numero: 8	**Nombre del requisito**: Detener proceso.
Programador: Zuleimys García Rodríguez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 15
Riesgo en desarrollo: Disponibilidad de la red.	**Tiempo real:** 15
Descripción: La funcionalidad comienza cuando usuario HMAST selecciona en el menú lateral la opción	

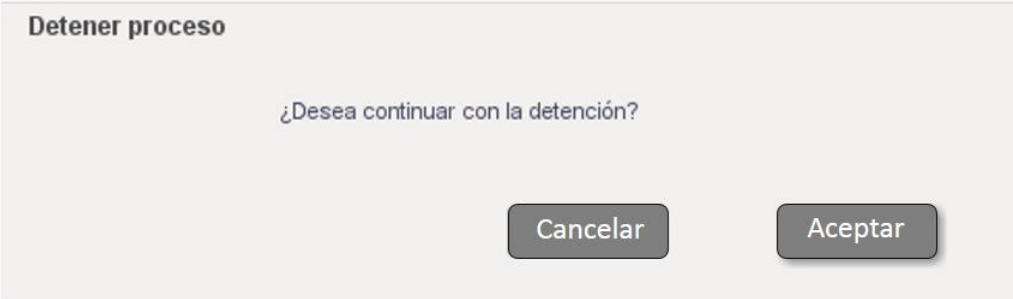
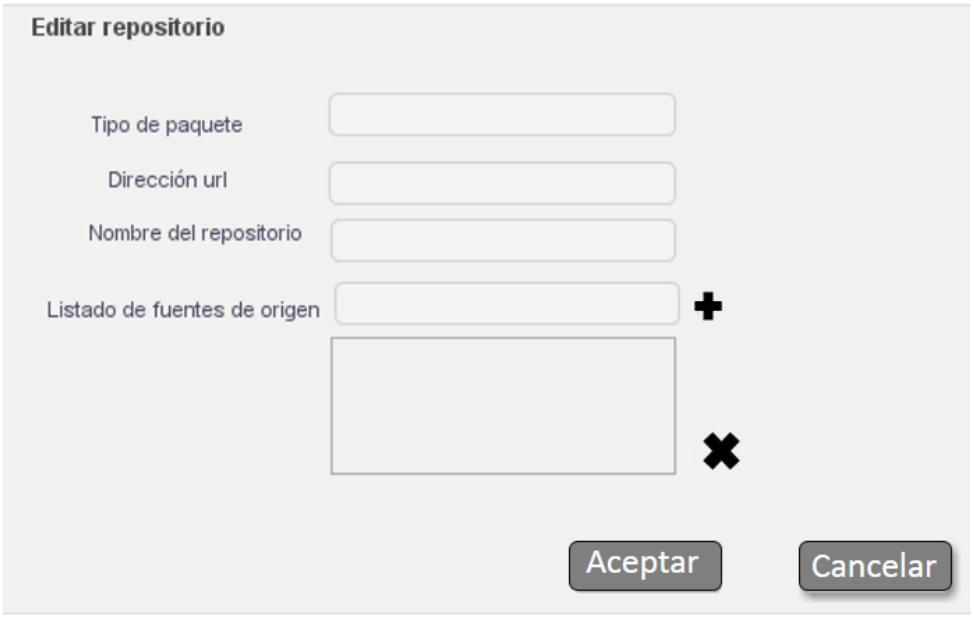
Gestión de procesos, donde escoge un proceso y ejecuta la opción *Detener*. Esta funcionalidad muestra una interfaz donde se puede aceptar o cancelar, y permite o no, detener la ejecución del proceso y liberar los recursos que utiliza.
Observaciones: El proceso es detenido en el sistema operativo solo si se ejecuta la opción que aplica los cambios en el servidor.
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 7 Historia de usuario Modificar repositorio.

Numero: 11	**Nombre del requisito**: Modificar repositorio.
Programador: Zuleimys García Rodríguez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 8
Riesgo en desarrollo: Disponibilidad de la red.	**Tiempo real:** 8
Descripción: La funcionalidad comienza cuando usuario HMAST selecciona en el menú lateral la opción Gestión de repositorios y accede al icono de *Editar*. Esta opción despliega una interfaz que contiene los valores con los que el repositorio seleccionado fue añadido al sistema y permite modificar los valores que posee: **Tipo de paquete**: ofrece la opción de decidir si el repositorio contendrá los paquetes deb o deb-src. **Url del repositorio**: permite modificar la dirección url del repositorio. **Nombre del repositorio**: ofrece la posibilidad de cambiar el nombre del repositorio que fue seleccionado.	

Origen: permite editar las fuentes de origen establecidas para los paquetes del repositorio. Estos valores son adicionados y mostrados en una lista para que el administrador decida si desea eliminar alguno de los valores que han sido establecidos o adicionar algunos nuevos.
Observaciones: El repositorio es modificado en el sistema operativo solo si se ejecuta la opción que aplica los cambios en el servidor.
Prototipo elemental de interfaz gráfica de usuario: 

2.4 Arquitectura del módulo

La arquitectura de software se encarga del diseño y especificación de la estructura global del sistema (38). Como arquitectura para el módulo se emplea la que define la herramienta HMAST (ver figura 4), con el objetivo de lograr una consistencia entre los componentes del sistema. Se utiliza una arquitectura N-Capas orientada al dominio, la que se encarga de estructurar la complejidad de la aplicación basada en las diferentes capas de la arquitectura guiada por los patrones y las tendencias de arquitecturas orientadas al dominio.

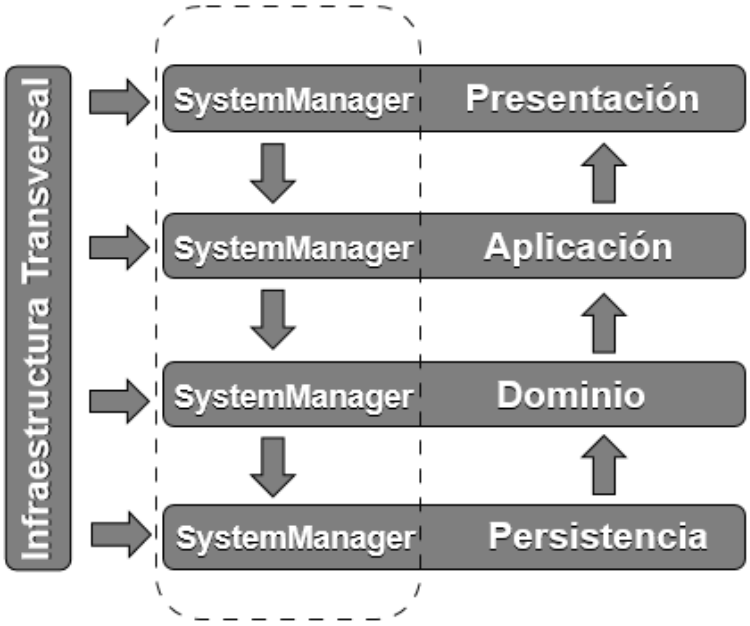


Figura 4 *Arquitectura del módulo SystemManager en la Herramienta HMAST.*

2.5 Diagrama de paquetes

El diagrama de paquetes muestra las agrupaciones lógicas que dividen al sistema y sus dependencias. Para el desarrollo del diagrama de paquetes se toma como referencia la arquitectura propuesta por HMAST, de modo que existe un paquete *systemmanager* por cada capa, y este a su vez contiene las clases del módulo (ver Figura 5).

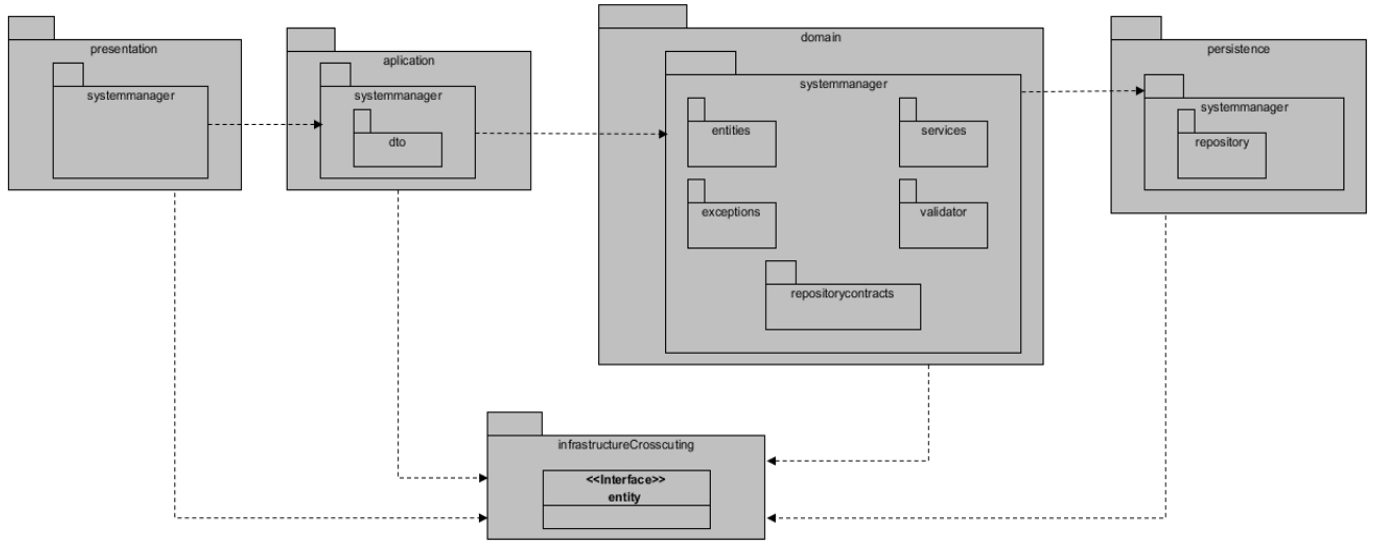


Figura 5 Diagrama de paquetes del módulo SystemManager en las capas de HMAST.

La capa de Dominio se relaciona con la capa de Aplicación, a través de la relación de agregación con la interfaz ISystemManagerRepository mediante las inyecciones de dependencias. Esta capa contiene dentro el paquete systemmanager y sus paquetes repositorycontrats, exceptions, validator y entitys. En el paquete entitys se definen las clases que se encargan de representar los objetos del Dominio y están definidas por su identidad y continuidad en el tiempo (ver Figura 6).

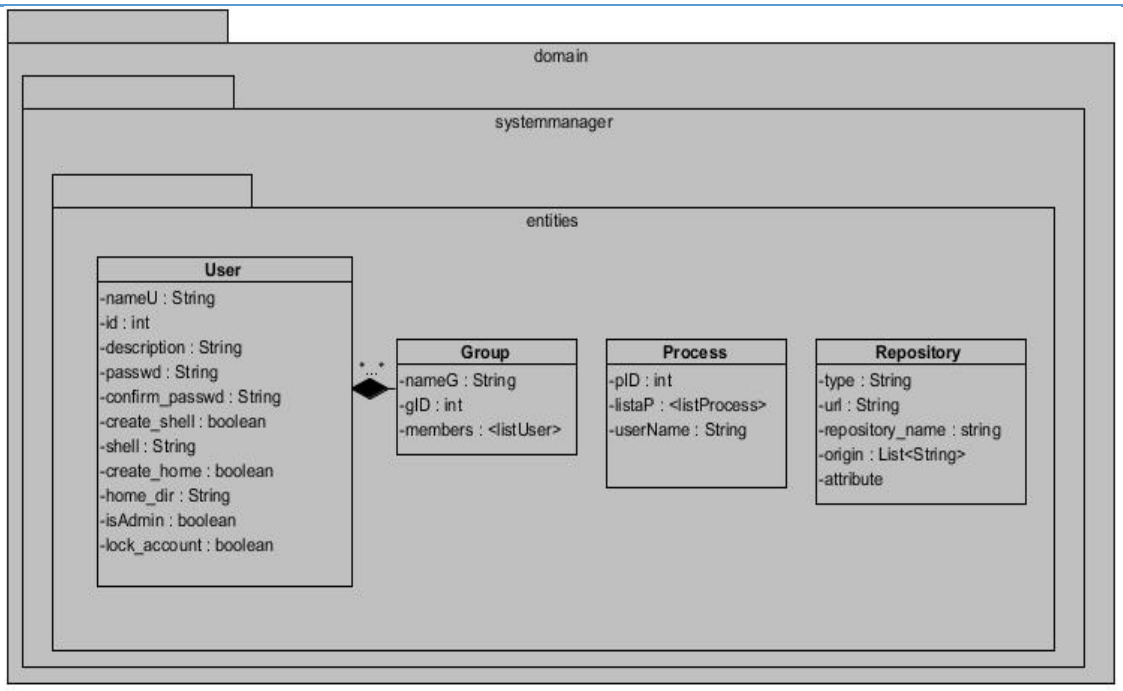


Figura 6 Diagrama de paquetes de la capa domain con las entidades del módulo.

2.6 Patrones de diseño

Los patrones de diseño son una solución a un problema de diseño. Estos patrones pretenden proporcionar catálogos de elementos reusables en el diseño de sistemas, evitar la reiteración de búsqueda de soluciones a problemas ya conocidos y solucionados con anterioridad, y estandarizar el modo en que se realiza el diseño (39). En el modelo de diseño del módulo se aplican patrones GRASP[29] y patrones GoF[30], los que favorecen la estructuración y entendimiento del diseño.

[29] GRASP: del inglés General Responsibility Assignment Software Patterns, Patrones Generales de Software para Asignar Responsabilidades.

[30] GoF: del inglés Gang of Four, La Banda de los Cuatro. Nombre con el que se conoce a los autores del libro Design Patterns.

2.6.1 Patrones GRASP

Describen las bases fundamentales de la asignación de responsabilidades y los principios del diseño de objetos expresados como patrones. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo, obtener una información encapsulada y un diseño con mayor cohesión (39).

Experto: empleado en la asignación de responsabilidades referentes a la obtención de información. Este patrón guía hacia diseños en los que los objetos del sistema ejecutan las operaciones realizadas para representar objetos inanimados del mundo real. Se mantiene el encapsulamiento de la información al permitir que se distribuya el comportamiento entre las clases, así como un bajo acoplamiento entre los objetos, lo que estimula las definiciones de clases más cohesivas. Este patrón es utilizado en la clase *SystemManagerRepository.*

Creador: permite conocer quién es el encargado de la creación o instanciación de las nuevas clases u objetos. Una de las actividades más comunes en un sistema orientado a objetos es la creación de instancias, por lo que si se asignan correctamente las responsabilidades el diseño puede soportar un bajo acoplamiento y mayor cohesión. Este patrón es empleado en la clase *SystemManagerAppService* de la capa de aplicación, a través de la creación de los Objetos para la Transferencia de Datos (DTO) entre la capa *Presentación* y *Aplicación* al utilizar las entidades del dominio que sean necesarias. En esta clase se crea un DTO a partir de una entidad o se crea la entidad a partir del DTO.

Bajo acoplamiento: a partir del framework *Spring* se obtienen inyecciones de dependencias, por lo que es necesario el uso de este patrón en el sistema. Los objetos no buscan o crean dependencias, sino que son dadas al objeto. Este patrón es utilizado a través del uso de las distintas interfaces que relacionan unas clases con otras de forma que las relaciones no se establezcan directamente hacia las clases. En la interfaz *ISystemManagerRepository* perteneciente al paquete *repositorycontrats* se evidencia el uso de este patrón, ya que quien realiza la implementación de sus métodos es la clase *SystemManagerRepository*, de la capa *Persistencia*.

Alta cohesión: se refiere a la medida en que se relacionan los elementos de un sistema, así como la centralización de sus responsabilidades. Las clases colaboran con otras para realizar determinadas

actividades y tienen una responsabilidad contenida. Garantiza que las clases sean posibles de entender, reutilizar y de fácil mantenimiento. Se evidencia en la clase *SystemManagerRepository*, ya que es quien interactúa con los objetos y realiza las operaciones de adición, modificación, actualización y eliminación de las entidades en el sistema.

2.6.2 Patrones GoF

Los patrones GoF son patrones de asignación de responsabilidades y se pueden categorizar en tres grupos a partir de su propósito: creacionales, estructurales y de comportamiento (39).

Solitario: es un patrón de tipo creacional, garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto. En el caso del módulo el patrón se utiliza para establecer una conexión SSH a un servidor, debido a que con una sola instancia de la conexión se realizan las operaciones en el sistema.

Conclusiones parciales

A partir del estudio realizado en el Capítulo 1 sobre la gestión de usuarios, grupos, procesos y repositorios, se definen las funcionalidades y las características del módulo propuesto, donde se obtienen 29 requisitos funcionales y 5 requisitos no funcionales. Se utilizó la arquitectura definida por la herramienta base, N-Capas orientada al Dominio, para mantener la homogeneidad del módulo con HMAST.

Capítulo 3: Implementación y pruebas de la gestión de usuarios, grupos, procesos y repositorios desde HMAST.

En el presente capítulo se especifican los estándares de codificación utilizados durante el desarrollo de la solución para lograr una uniformidad respecto a los demás módulos. Se definen y se llevan a cabo pruebas al sistema, de las que se documentan los resultados. Se realiza además el modelado del diagrama de despliegue, incluyendo la distribución de los componentes del sistema.

3.1 Estándares de codificación

A partir de las pautas definidas en el expediente de proyecto de HMAST se definen como estándares empleados para el desarrollo del módulo propuesto:

- **Asignación de nombres**: emplear descriptores en inglés. Evitar nombres largos que difieran en una letra o mediante el uso de mayúsculas. En el caso de los nombres de variables y funciones se escriben con la primera letra en minúsculas, en caso de que el nombre sea compuesto utilizar la notación CamelCase[31].
- **Ficheros de código fuente:** para cada fichero existe una única clase o interfaz, aunque si existe una clase privada o una interfaz asociada a una clase pública se pueden colocar en el mismo fichero, donde aparece primero la clase pública.
- **Identación**: la unidad de identación de bloques de sentencias son 4 espacios.
- **Comentarios**: deben añadir claridad al código, contar el por qué y no el cómo y ser concisos.
- **Declaraciones**: se declara cada variable en una línea distinta para permitir que cada una se comente por separado.
- **Continuidad de las líneas largas**: en el momento en que una sentencia no quepa en una única línea se debe fraccionar después de una coma u operador, y alinear a la nueva línea al mismo nivel con el comienzo de la expresión de la línea anterior.
- **Longitud de la línea**: Limitar todas las líneas a un máximo de 120 caracteres.

[31] CamelCase: Estilo de escritura que se aplica a frases o palabras, este estilo elimina espacios y define la primera letra de cada palabra en mayúsculas.

- **Nombre de componentes:** Todos los paquetes comienzan con cu.uci.hmast.xxx.yyy.zzz.kkk.

  xxx → presentation, aplication, domain, persistence.

  yyy → nombre del módulo (systemmanager).

  zzz → elementos que pueden contener los componentes verticales (entities, repositorys).

  kkk → clases o subpaquetes.

3.2 Pruebas del sistema

La realización de pruebas se lleva a cabo para demostrar que el sistema ejecuta las funcionalidades para las que fue creado, y están orientadas a demostrar que existen defectos en el software. El diseño de casos de prueba se encarga de realizar pruebas que puedan encontrar la mayor cantidad de no conformidades con el menor esfuerzo y tiempo posible.

La metodología de software variación de AUP para la UCI establece 3 disciplinas para la ejecución de pruebas: internas, liberación y aceptación. Como parte de las pruebas realizadas al módulo implementado se decide la realización de las pruebas internas y de aceptación. En las pruebas internas se propone verificar el resultado de la aplicación, al probar cada construcción, así como las versiones finales del módulo. En el caso de las pruebas de aceptación, se llevan a cabo para verificar que el software está listo y que puede ser utilizado por usuarios finales, para ejecutar las tareas y funciones para las que fue construido.

Para la ejecución de las pruebas internas se define una estrategia de prueba con un enfoque incremental, al partir del análisis de la codificación, se realizan pruebas de integración basándose en el diseño y por último se ejecuta la prueba de validación centrándose en el análisis de los requisitos.

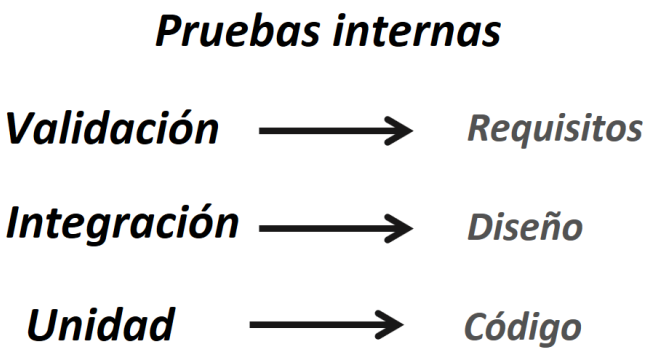


Figura 7 Etapas en el proceso de prueba del software.

En la Figura 7 se puede apreciar cómo está conformado el plan de pruebas a partir de la estrategia planteada. Para esta estrategia se define el análisis de código para comprobar que el sistema funciona correctamente, al realizar el método de prueba de caja blanca y ejercitar los caminos específicos para lograr una detección máxima de errores durante la ejecución de la prueba de unidad. La prueba de integración permite analizar el diseño y la construcción de la arquitectura de software. Durante la prueba de validación se valida el cumplimiento de los requisitos establecidos durante la etapa de Análisis y Diseño (40).

3.2.1 Prueba de unidad

Según Pressman, la prueba de unidad se lleva a cabo mediante la realización del método de caja blanca y la técnica de camino básico. Esta prueba tiene como objetivo verificar la funcionalidad y estructura de cada componente individualmente una vez que ha sido desarrollado. Es un proceso para probar los subprogramas, subrutinas, procedimientos individuales o las clases en un programa (40).

Para la ejecución de las pruebas se analizan las operaciones pertenecientes a las clases del módulo, y se aplica la técnica de camino básico a todos los métodos del sistema. A continuación, se presenta la ejecución de la prueba a la operación **loadSystemProcess** de la clase **SystemManagerRepository**.

Tabla 8 Ejecución de la técnica camino mínimo al método loadSystemProcess.

(1)	public List<ProcessSystem> loadSystemProcess (LogicalServer logicalServer) throws IOException, JSchException, InterruptedException, EErrorMessageOperatingSystem {
(2)	List<ProcessSystem> result = entityManager.createQuery("select process from ProcessSystem process where process.idServer =: idServer", ProcessSystem.class).setParameter("idServer", logicalServer.getId()).getResultList();
(3)	if (!result.isEmpty())
(4)	return result;
(5)	IPersist sshConnection = logicalServer.getPersistens();
(6)	String commmand = "ps -aux";
(7)	PersistOutput output= sshConnection.executeComandInServerWithCodeOutput (commmand);
(8)	if (output.getCode() != 0)
(9)	throw new EErrorMessageOperatingSystem(output.getCode(), commmand, output.getStandardOutputErrors().toString());
(10)	String[] processLines = output.getStandarOutput().split("\n");
(11)	this.createRootFolderConfigurationIfNotExist(logicalServer);
(12)	for (int i = 1; i < processLines.length; i++) {
(13)	String[] arr = processLines[i].split(" ");
(14)	List<String> lista = new LinkedList<>();
(15)	for (int j = 0; j < arr.length; j++){
(16)	if (!arr[j].isEmpty())

```
(17)  lista.add(arr[j]);
(18)  }
(19)  ProcessSystem process= new ProcessSystem();
(20)  process.setIdServer(logicalServer.getId());
(21)  process.setId(UUID.randomUUID());
(22)  process.setUser(lista.get(0));
(23)  process.setIdProcessSystem(lista.get(1));
(24)  process.setCpu(lista.get(2));
(25)  process.setMemoria(lista.get(3));
(26)  process.setVsz(lista.get(4));
(27)  process.setRss(lista.get(5));
(28)  process.setTty(lista.get(6));
(29)  process.setStat(lista.get(7));
(30)  process.setStart(lista.get(8));
(31)  process.setTime(lista.get(9));
(32)  process.setCommand(lista.get(10));
(33)  entityManager.persist(process);
(34)  result.add(process);
(35)  }
(36)  return result;
(37)  }
```

El método de camino básico permite obtener una medida de la complejidad lógica de un diseño procedimental, y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Después de enumerar las sentencias del código se diseña la gráfica que describe el flujo de control lógico mediante nodos y aristas.

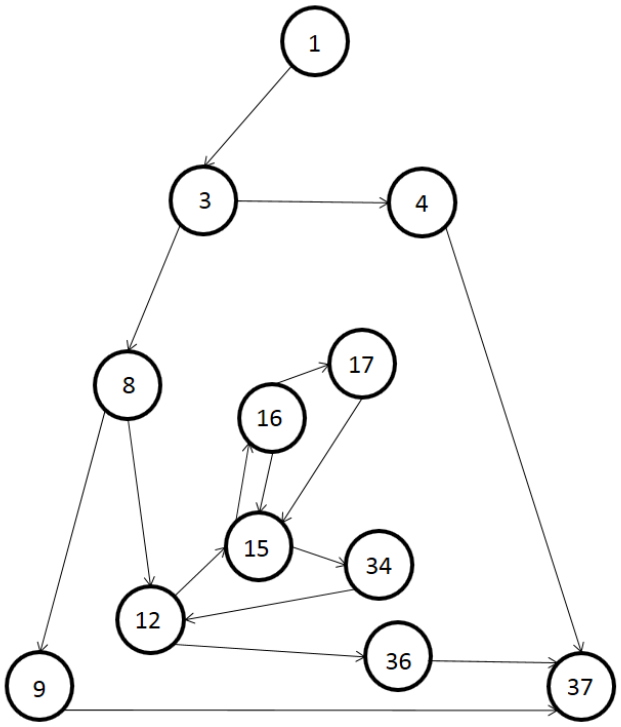


Figura 8: Grafo de flujo.

El grafo de flujo de la instrucción anterior define 12 nodos y 16 aristas. Para estos valores se calcula la complejidad ciclomática V(G), que representa la métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Esta complejidad lógica está compuesta por un conjunto de caminos independientes que nos ofrece un límite superior para el número de pruebas que se deben realizar para comprobar que se ejecuta cada sentencia al menos una vez. La complejidad para este grafo es:

V(G) = cantidad_aristas – cantidad_nodos + 2

V(G) = 16 – 12 + 2 = 6

Tabla 9 Caminos obtenidos durante la prueba de unidad.

Camino 1	***1 – 3 –*** 4 ***–37***
Camino 2	***1 –3 –8 –9 –37***
Camino 3	***1 – 3 – 8 – 12 – 36 –37***
Camino 4	***1 – 3 – 8 – 12 – 15 – 34 – 12 – 36 –37***
Camino 5	***1 – 3 – 8 – 12 – 15 – 16 – 15 – 12 – 36 –37***
Camino 6	***1 – 3 – 8 – 12 – 15 – 16 – 17 – 15 – 12 – 36 –37***

A partir de los caminos encontrados en la complejidad ciclomática se diseñan y ejecutan los casos de prueba que son aplicados a cada ruta independiente. A continuación, se muestran los resultados de recorrer los caminos:

Tabla 10 Caso de prueba de unidad para el camino 1.

Caso de Prueba de Unidad	
Camino 1	***Ruta: 1 – 3 – 4 – 37***
Nombre de la persona que realiza la prueba: Zuleimys García Rodríguez	
Descripción de la prueba: Mostrar el listado de procesos del sistema que se encuentra en la base de datos.	
Entrada: Recibe como parámetro el identificador del servidor lógico del que se van a listar los procesos.	
Resultado esperado: El sistema muestra el listado de procesos de la base de datos.	
Evaluación de la prueba: Satisfactorio.	

Tabla 11 Caso de prueba de unidad para el camino 2.

Caso de Prueba de Unidad	
Camino 2	***Ruta: 1 –3 –8 –9 –37***
Nombre de la persona que realiza la prueba: Zuleimys García Rodríguez	
Descripción de la prueba*: Mostrar un mensaje de notificación que informa que no existen procesos en la base de datos*.	
Entrada: Recibe como parámetro el identificador del servidor lógico del que se van a listar los procesos.	
Resultado esperado: El sistema muestra un mensaje notificando que no existen procesos en *la base de datos*.	
Evaluación de la prueba: Satisfactorio.	

Tabla 12 Caso de prueba de unidad para el camino 3.

Caso de Prueba de Unidad	
Camino 3	***Ruta: 1 – 3 – 8 – 12 – 36 –37***
Nombre de la persona que realiza la prueba: Zuleimys García Rodríguez	
Descripción de la prueba: Mostrar el listado de procesos del sistema con la información incompleta.	
Entrada: Recibe como parámetro el identificador del servidor lógico del que se van a listar los procesos.	
Resultado esperado: El sistema muestra el listado de procesos del sistema con la información incompleta.	
Evaluación de la prueba: Satisfactorio.	

Tabla 13 Caso de prueba de unidad para el camino 4.

Caso de Prueba de Unidad	
Camino 4	***Ruta: 1 – 3 – 8 – 12 – 15 – 34 – 12 – 36 –37***

Nombre de la persona que realiza la prueba: Zuleimys García Rodríguez
Descripción de la prueba: Mostrar el listado de procesos del sistema vacío.
Entrada: Recibe como parámetro el identificador del servidor lógico del que se van a listar los procesos.
Resultado esperado: El sistema muestra el listado de procesos del sistema vacío.
Evaluación de la prueba: Satisfactorio.

Tabla 14 Caso de prueba de unidad para el camino 5.

Caso de Prueba de Unidad	
Camino 5	***Ruta: 1 – 3 – 8 – 12 – 15 – 16 – 15 – 12 – 36 –37***
Nombre de la persona que realiza la prueba: Zuleimys García Rodríguez	
Descripción de la prueba: Mostrar el listado de procesos del sistema con la información de un proceso.	
Entrada: Recibe como parámetro el identificador del servidor lógico del que se van a listar los procesos.	
Resultado esperado: El sistema muestra el listado de procesos del sistema con la información de un proceso.	
Evaluación de la prueba: Satisfactorio.	

Tabla 15 Caso de prueba de unidad para el camino 6.

Caso de Prueba de Unidad	
Camino 6	***Ruta: 1 – 3 – 8 – 12 – 15 – 16 – 17 – 15 – 12 – 36 –37***
Nombre de la persona que realiza la prueba: Zuleimys García Rodríguez	
Descripción de la prueba: Mostrar el listado de procesos del sistema que se encuentran en el servidor lógico.	

Entrada: Recibe como parámetro el identificador del servidor lógico del que se van a listar los procesos.
Resultado esperado: El sistema muestra el listado de procesos del sistema que se encuentran en el servidor lógico.
Evaluación de la prueba: Satisfactorio.

Para la ejecución de la prueba de unidad se tuvo en cuenta como se llevó a cabo el proceso de desarrollo del sistema, el que se realizó de forma vertical, al desarrollar todos los requisitos de cada elemento a gestionar antes de pasar al siguiente, ejecutándose un total de 4 iteraciones.

A partir de la realización de la prueba para la gestión de usuarios se analizaron 7 casos de prueba con un total de 9 no conformidades. Para la gestión de grupos se definieron 7 casos de prueba, donde se encontraron 5 no conformidades. Durante la prueba a la gestión de procesos se llevaron a cabo 6 casos de prueba para un total de 3 no conformidades, y por ultimo para la gestión de repositorios se analizaron 6 casos de prueba, done se encontraron 2 no conformidades. Las no conformidades detectadas se encontraban asociadas a errores de validación, tratamiento de excepciones desde el código y la redacción de mensajes de confirmación o error del sistema.

3.2.2 Prueba de integración

La prueba de integración es una técnica para construir la estructura del programa con el fin de detectar errores asociados con la interacción (36). Como parte de la prueba de integración se realiza la prueba de regresión, al ejecutar el mismo subconjunto de pruebas que ya se han aplicado para asegurar que los cambios no han propagado efectos colaterales no deseados (40).

La integración del módulo es un elemento clave tenido en cuenta durante todo el proceso de desarrollo. Debido a que la propuesta de solución constituye un módulo que debe ser integrado a la herramienta HMAST, se definen desde el inicio de la etapa de desarrollo algunos elementos como el lenguaje de programación y la utilización de librerías y frameworks comunes. A partir de la arquitectura definida para el sistema se emplea una estrategia de integración ascendente, al probar el programa en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir.

3.2.3 Prueba de validación

La prueba de validación se ejecuta para comprobar el funcionamiento del sistema, demostrar que cumple con lo que se espera y permitir que el usuario compruebe su funcionalidad y rendimiento. La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. A continuación, se presentan los casos de prueba que corresponden a la historia de usuario *Adicionar usuario*.

Tabla 16 Descripción de las variables para la prueba de validación.

No.	Nombre de campo	Clasificación	Valor nulo	Descripción
1	**Nombre**	Campo de texto	No	Debe ser una palabra de 32 caracteres como máximo, no debe contener espacios ni los caracteres especiales /*+-;´´><
2	**Identificador**	Campo de texto	Si	Debe ser un número mayor o igual que 1000.
3	**Descripción**	Campo de texto	Si	Puede tener una longitud de 255 caracteres y debe describir el nombre completo del usuario o algún dato específico de la cuenta.
4	**Contraseña**	Campo de texto	No	Debe contener una cadena de caracteres con un mínimo de 8 elementos, debe contener mayúsculas, minúsculas, números y caracteres especiales.
5	**Confirmar contraseña**	Campo de texto	No	Debe contener una cadena de caracteres que coincida exactamente con el campo Contraseña.
6	**Habilitar**	Campo de selección	Si	Se habilita si el usuario no va a utilizar

	home			la dirección de home por defecto y desea definir otra dirección de home para la cuenta, y activa el campo **Dirección del home.**
7	**Dirección del home**	Campo de texto	No	Si el campo de selección **Habilitar home** esta seleccionado, el home debe existir y ser una ruta válida en el sistema.
8	**Habilitar terminal**	Campo de selección	Si	Se habilita si el usuario desea definir una terminal de comandos diferente a la que define el sistema. Esta selección activa el campo de texto **Dirección de la terminal**.
9	**Dirección de la terminal**	Campo de texto	No	Si el campo de selección Habilitar terminal esta seleccionado, la terminal debe existir y debe ser una terminal válida.

Escenario:1.1 Adicionar usuario con los valores obligatorios.

Descripción: El sistema debe permitir adicionar el usuario con los valores definidos en los campos no nulos.

Flujo central: Al acceder al icono adicionar, se muestra una ventana donde se introducen los valores válidos, luego se accede al botón aceptar.

Tabla 17 Escenario 1.1 Adicionar usuario con los valores obligatorios.

V1	V2	V3	V4	V5	V6	V7	V8	V9
V	N/A	V	V	N/A	N/A	N/A	N/A	N/A
pepe		**12345678**	**12345678**					

V	N/A	V	V	V	V	V	V	N/A
maría		**12345678**	**12345678**	**True**	**/home/pepe**	**True**	**/bin/sh**	

Respuesta del sistema: El sistema adiciona el usuario y muestra el mensaje: El usuario ha sido adicionado.

Escenario 1.2 Adicionar usuario con los valores obligatorios vacíos.

Descripción: El sistema no debe adicionar el usuario con los campos obligatorios vacíos.

Flujo central: Al acceder al icono adicionar, se muestra la ventana adicionar usuario, donde se introducen los valores especificados se accede al botón aceptar.

Tabla 18 Escenario 1.2 Adicionar usuario con los valores obligatorios vacíos.

V1	V2	V3	V4	V5	V6	V7	V8	V9
I	N/A	V	V	N/A	N/A	N/A	N/A	N/A
(vacío)		**123qaz,-**	**123qaz,-**					
V	N/A	V	V	N/A	N/A	N/A	N/A	N/A
cesol		**1qazxsw2..**	**(vacío)**					

Respuesta del sistema: El sistema muestra un mensaje de error con un aviso de que el usuario no se ha podido adicionar y muestra el parámetro que contiene vacío.

Escenario 1.3 Adicionar usuario con valores obligatorios incorrectos.

Descripción: El sistema no debe permitir adicionar un usuario con valores incorrectos.

Flujo central: Al acceder a la ventana de adicionar, se introducen los valores especificados y se accede al botón aceptar.

Tabla 19 Escenario 1.3 Adicionar usuario con los valores obligatorios incorrectos.

V1	V2	V3	V4	V5	V6	V7	V8	V9
V	N/A	V	V	N/A	N/A	N/A	N/A	N/A
José Pérez		Zaq12wsx,,	Zaq12wsx,,					
V	N/A	V	V	N/A	N/A	N/A	N/A	N/A
carlos		qazxcv123.	QAZXCV123.					

Respuesta del sistema: El sistema muestra un mensaje de error notificando que el usuario a adicionar contiene valores incorrectos y especifica el o los parámetros que contienen error.

Escenario 1.4 Adicionar usuario con valores opcionales incorrectos.

Descripción: El sistema no debe permitir adicionar un usuario con valores incorrectos.

Flujo central: Al acceder a la ventana de adicionar y llenar los campos con los valores especificados y se accede al botón aceptar.

Tabla 20 Escenario 1.4 Adicionar usuario con los valores opcionales incorrectos.

V1	V2	V3	V4	V5	V6	V7	V8	V9
V	I	V	V	N/A	N/A	N/A	N/A	N/A
jperez	100	Zaq12wsx,,	Zaq12wsx,,					

Respuesta del sistema: El sistema muestra un mensaje de error que informa que el usuario a adicionar contiene valores incorrectos y especifica el parámetro que contiene el error.

3.2.4 Prueba de aceptación

Las pruebas de aceptación son ejecutadas para validar la conformidad del cliente de acuerdo con las especificaciones definidas. Para el módulo de gestión del sistema, la variable de mayor relevancia es la integridad de los datos, por lo que se define un diseño que garantiza la validación de esta variable. Según la norma ISO/IEC 9126 de la Oficina Nacional de Normalización, el modelo de calidad está definido por categorías, que validan los atributos a partir de seis características: funcionalidad, confiabilidad, usabilidad, mantenibilidad, eficiencia y portabilidad. De estos atributos se realizó la evaluación de la

prueba de funcionalidad como parte de las pruebas de aceptación establecidas por la metodología de software. A partir de la ejecución de las pruebas el cliente pudo examinar cada una de las funcionalidades del sistema, al comprobar el correcto funcionamiento del mismo y determinar que el módulo si facilita el trabajo de los administradores de los OACE. Para que así conste, el cliente firma un Acta de aceptación de productos de trabajo, la que figura en los anexos del presente trabajo.

3.3 Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Muestra las relaciones físicas entre los componentes de *hardware* y *software* en el sistema final (41). La herramienta HMAST se ejecuta en un servidor web *Tomcat* sobre un sistema operativo GNU/Linux Nova Servidores. El usuario accede a HMAST mediante un navegador web que emplea conexiones seguras con el protocolo HTTPS. A través del módulo Gestión del sistema, el usuario accede al servidor y realiza las operaciones de gestión de usuarios, grupos, procesos y repositorios del sistema. A continuación, se muestra la distribución física del módulo:

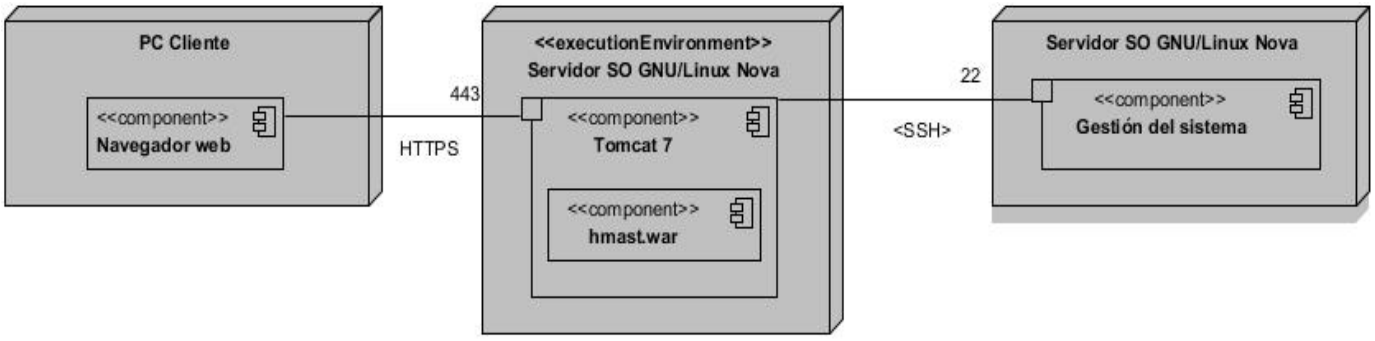


Figura 9: Diagrama de despliegue del sistema.

Conclusiones parciales

Mediante la especificación de los estándares de codificación se pudo realizar una aplicación que permitiera la reutilización y la comprensión de su código, además de mantener una uniformidad respecto a los restantes módulos. La definición y puesta en práctica de una estrategia de prueba que contara con un enfoque incremental posibilitó la evaluación del módulo, así como la revisión de la correcta ejecución de las instrucciones, la evaluación de la integración y la validación del cumplimiento de los requisitos del sistema.

Conclusiones generales

La realización de la presente investigación permitió desarrollar un módulo para la herramienta HMAST que gestiona usuarios, grupos procesos y repositorios del sistema operativo GNU/Linux, al dar respuesta a los planteamientos propuestos en las preguntas científicas descritas al inicio del trabajo. A partir de lo antes mencionado se concluye que:

- El estudio de la gestión de usuarios, grupos, procesos y repositorios en los sistemas operativos GNU/Linux permitió comprender su funcionamiento y estructura.
- A partir del análisis de las herramientas homólogas se logró identificar las funcionalidades a desarrollar en la solución propuesta.
- Mediante el diseño e implementación del módulo se logró un sistema informático que permite gestionar los usuarios, grupos, procesos y repositorios del sistema operativo.
- A partir de la ejecución de las pruebas se logró validar el correcto funcionamiento del sistema, además del cumplimiento de los requisitos definidos.

Recomendaciones

Una vez concluido el presente trabajo de diploma se recomienda, en función de ampliar las tareas de gestión de usuarios, grupos, procesos y repositorios:

- Definir un modo de iniciar los procesos del sistema desde la herramienta HMAST.
- Determinar un mecanismo para evitar dese HMAST la eliminación en cascada de los procesos, al eliminar un proceso que posea hijos en ejecución.

Referencias Bibliográficas

1. gnu.org. [En línea] 2016. [Citado el: 14 de noviembre de 2016.] https://www.gnu.org/gnu/linux-and-gnu.es.html.

2. ¿Qué es GNU/Linux? [En línea] [Citado el: 14 de noviembre de 2016.] https://www.debian.org/releases/stable/mips/ch01s02.html.es.

3. Canonical Ltd. Ubuntu. Administración de usuarios en Ubuntu. [En línea] 2017. [Citado el: 13 de noviembre de 2016.] http://www.ubuntu-es.org/node/173046.

4. Stallman,Richard.cyta.com.ar.[En línea]2015.http://www.cyta.com.ar/biblioteca/bddoc/020301/v2n3a1.htm.

5. Canonical Ltd. Ubuntu. [En línea] 2017. [Citado el: 20 de noviembre de 2016.] https://help.ubuntu.com/lts/serverguide/configuration.html.

6. Plataforma de gestión de servicios telemáticos en GNU/Linux. Ramón Alexander Anglada Martínez, Alain Abel Garófalo Hernandez. 3, La Habana : UCI, 2012, Vol. 11.

7. Villazon Perez, Yoandy. Metodologia para la Migracion a Software Libre de las Universidades del Ministerio de Educacion Superior. Tesis de diploma. Universidad de las Ciencias Informaticas, Facultad 10, La Habana, Cuba, 2008.

8. Ubuntu guia. Gestionar usuarios y grupos en Ubuntu. [En línea] 19 de abril de 2014. [Citado el: 4 de diciembre de 2016.] http://www.ubuntu-guia.com/2009/09/gestion-de-usuarios-y-grupos-en-ubuntu.html.

9. humanos.uci.cu. Curso Linux: Administración de Sistema y Servicios - Administracion_Linux_2.pdf. [En línea] [Citado el: 1 de diciembre de 2016.] https://humanos.uci.cu/wp-content/uploads/2010/09/Administracion_Linux_2.pdf.

10. Stallman, Richard M. Software libre para una sociedad libre. s.l. : Traficantes de Sueños, 2004.

11. humanos.uci.cu. [En línea] 2011-2012. [Citado el: 30 de noviembre de 2016.] https://humanos.uci.cu/2012/10/gestion-de-procesos-en-linux-fg-jobs-presentacion-en-pdf/.

12 . Guía Ubuntu. [En línea] 2012. [Citado el: 26 de diciembre de 2016.] http://www.guia-ubuntu.com/index.php/A%C3%B1adir_repositorios_externos.

13. Ubuntu. [En línea] 2015. [Citado el: 28 de diciembre de 2016.] https://lists.ubuntu.com/archives/ubuntu-ar/2015-September/049202.html.

14. documentación cPanel. [En línea] 22 de octubre de 2014. [Citado el: 30 de noviembre de 2016.] https://documentation.cpanel.net/display/ALD/Installation+FAQ.

15. Plesk International GmbH. Plesk, the power to simplify. [En línea] 2017. [Citado el: 30 de diciembre de 2016.] https://www.plesk.com/about-us.

16. YaST/Introducción - OpenSUSE. [En línea] [Citado el: 4 de diciembre de 2016.] https://es.opensuse.org/Portal:YaST/Introducci%C3%B3n.

17. Zentyal Linux Small Business Server. [En línea] 2017. [Citado el: 30 de diciembre de 2016.] https://www.zentyal.com.es.

18. ubuntu es.¿Qué es WebMin? [En línea] 6 de marzo de 2010. [Citado el: 4 de diciembre de 2016.] http://www.ubuntu-es.org/node/129411.

19. Stephenson, Neal (1999). En el principio... fue la línea de comandos. Consultado el 4 de diciembre de 2016.

20. Santiago, Felipe González. Módulo de HMAST para la administración y migración hacia Samba4 del servicio de Directorio Activo. s.l. : UCI, 2016.

21. springLa. [En línea] 2015. [Citado el: 4 de diciembre de 2016.] https://springla.io/spring/spring-framework/.

22. Bootstrap [En línea] 2017 [Citado el: 4 de diciembre de 2016.] http://getbootstrap.com/.

23. Savit, Jeff, Wilcox, Sean y Jayaraman, Bhuvana. JAVA para la empresa. s.l. : McGraw-Hill, 2000.

24. Anón. Características del lenguaje Java. [En línea] [Citado el: 30 de noviembre de 2016.] <http://www.iec.csic.es/criptonomicon/java/quesjava.html>.

25. IntelliJ IDEA. [En línea] 2000-2017. [Citado el: 4 de diciembre de 2016.] https://www.jetbrains.com/idea/.

26. Contreras Martínez, José Luís, Ana García Domenech, Daniel Martínez Espadas, y Begoña Morillas. Práctica 4 - Herramienta CASE: Umbrello. [En línea] [Citado el: 30 de noviembre de 2016.] <http://es.scribd.com/doc/86321835/Practica-4-Herramienta-CASE-Umbrello>.

27. Visual Paradigm. Visual Paradigm. [En línea] 2017. [Citado el: 17 de enero de 2017.] https://www.visual-paradigm.com/aboutus/.

28. Larman, Craig. UML y patrones, 2da Edición.

29. Solution, E. (2009-2015). ForeUI. Obtenido de http://www.foreui.com.

30. SVN Pilone, Dan y Miles, Russ. Head First Software Development. O'Reilly, 26 de marzo de 2008.

31. GitLab. GitLab.com. [En línea] 2017. [Citado el: 2017 de 1 de 14.] https://about.gitlab.com/about/.

32. Apache Maven. [En línea] 2002-2017. [Citado el: 14 de 1 de 2017.] https://maven.apache.org/.

33. jquery.org, jQuery Foundation. ¿Qué es jQuery? [En línea] [Citado el: 3 de diciembre de 2016.] https://translate.googleusercontent.com/translate_c?depth=1&https://jquery.com/&usg=ALkJrhjTwLmwogYRh4E2DtJlzW96vIjFjw.

34. W3C. [En línea] 2017. [Citado el: 10 de enero de 2017.] https://www.w3.org/html/.

35. Augeas- Main. [En línea] [Citado el: 3 de diciembre de 2016.] http://augeas.net/.

36. Palma, Nurisel. Módulo para la administración de los servidores web en HMAST. Habana : UCI, 2013.

37. ISTR-Ingeniería de Software y tiempo real. [En línea] Universidad de Cantabria. [Citado el: 17 de enero de 2017.] http://www.ctr.unican.es/asignaturas/is1/IEEE830_esp.pdf.

38. David Garlan, M. S. (1994). An introduction to Software Architecture. Pittsburgh.

39.Erich Gamma, R. H.-A. (1995-2005). Design Patterns. Elements of Reusable Object-Oriented Software.

40. Pressman, Roger S. Ingeniería de Software Un enfoque práctico. Quinta edicion.

41. OMG Unified Modeling Language(OMG UML), Superstructure, V2.1.2. 2007. formal/2007-11-02.

Anexo 1

Acta de aceptación del producto.

Universidad de las Ciencias Informáticas

Acta de aceptación de productos de trabajo

ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO

En cumplimiento del **Convenio de colaboración** establecido entre el **Centro de Software Libre (CESOL)** y la estudiante **Zuleimys García Rodríguez** de la Facultad 1 de la Universidad de las Ciencias Informáticas y en función de la ejecución del proyecto: **Módulo para la gestión de usuarios, grupos, procesos y repositorios del sistema GNU/Linux desde HMAST**, se hace entrega del producto que se relaciona a continuación:

- Módulo para la gestión de usuarios, grupos, procesos y repositorios del sistema GNU/Linux desde HMAST.

La parte Cliente, luego de haber revisado el producto de trabajo relacionado anteriormente procede a firmar la aceptación de los mismos en total conformidad.

Entrega	Recibe
Nombre y apellidos: Zuleimys García Rodríguez Villazón	**Nombre y apellidos:** Yoandy Pérez
Cargo: Estudiante Facultad 1	**Cargo:** Director de CESOL
Firma:	**Firma:**

Universidad de las Ciencias Informáticas
CESOL Facultad 1

Fecha: 31/05/2017

Figura 7 Acta de aceptación de productos de trabajo.