Universidad de las Ciencias Informáticas

Facultad 1

“Aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores”

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

Autora:

Anaely Trimiño Haramboure

Tutores:

MSc. Nurisel Palma Pérez

Ing. Zuleimys García Rodríguez

La Habana, diciembre de 2021
“Año 63 de la Revolución”

Agradecimientos

¡A mi mama Iliana porque me enseñó a ser INDEPENDIENTE y volar sola por la vida en busca de cumplir mis sueños, aunque siempre llora cuando me voy de casa! Te amo mami.

A mi papa Pedro por ser el padre que todos y todas sueñan, del cual me siento muy orgullosa y privilegiada de tener, porque la palabra perfecto se queda corta para describirlo, ¡por tanta compresión y tanto amor! Te amo papi.

A mi hermano Yoermes porque a pesar de su carácter difícil ha sabido ser el mejor ejemplo a seguir y por darme a mis sobrinos Yoel y David que son mi vida entera.

A Daylin porque a pesar de somos polos opuestos completamente tenemos una relación de amistad hermosa que espero nos dure toda la vida y por ayudarme en la tesis como si fuera la de ella.

A Arlet por estar conmigo todos estos años llenos de buenos y malos momentos y por tenerme presente como yo siempre la tengo a ella.

A Rocío por ser simplemente mí Rocío y llenarme de alegría con su sonrisa y sus Donu't.

A Rachel por su apoyo incondicional y por estar siempre juntas a pesar de la distancia.

A Yuyu por la complicidad y darme una hermosa sobrina a la cual adorar.

A Gabriel por hacerme sonreír, aunque el momento no lo amerite.

A todos mis compañeros de aula (fueron muchos, he sido siempre una nómada de grupos) que les deseo lo mejor siempre.

A todas las personas que de una forma u otra se han cruzado en mi camino en todos estos años.

A mi tutora Nurisel por exigirme, apoyarme y empujarme en este proceso, por las largas charla de la tesis y la vida, gracias por enseñarme de todo un poco y tratarme con tanto amor.

A mi tutora Zuleimys por correr conmigo a última hora.

A todos los profesores con los que he tropezado a lo largo de la carrera que de una forma u otra me han ayudado.

A mi perrita Mona por ser mi compañera hasta las 5am mientras hacía tesis y por sus lengüetazos que alegran mi existencia.

Y por último a Pedro por los años y años de entrega y amor, porque solo nosotros nos soportamos mutuamente, porque mi tesis lleva su sello en alguna esquinita por así decirlo, por regalarme a Mona y por estar al principio y al final de esta carrera, ¡gracias ratón!

Dedicatoria

A mi madre Iliana por ser luz y a mi padre Pedro por ser felicidad.

Al resto de mi familia los que están todos los días en especial a mi hermano Yoermes y los que están alguna que otra vez.

A Pedro por tantos años de complicidad.

A mis amigas y amigos, porque son lo máximo sin ustedes no respiro chicos.

A los que fueron, los que vienen y en especial, a los que ven más allá de cualquier adversidad…

Declaración de autoría

Declaro por este medio que yo **Anaely Trimiño Haramboure**, con carné de identidad **97051611299** soy la autora principal del trabajo titulado **Aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.
Para que así conste firmamos la presente a los 10 días del mes de diciembre del 2021.

Anaely Trimiño Haramboure

Nombre del autor

MSc. Nurisel Palma Pérez

Tutora

Ing. Zuleimys García Rodríguez

Tutora

Resumen

El presente trabajo tuvo como objetivo desarrollar una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota, facilitando el trabajo de los especialistas en servicios telemáticos durante el proceso de migración a código abierto y de los administradores de redes. Se emplearon los métodos Analítico-Sintético, Entrevista y Observación. Se analizaron las diversas herramientas que permiten la llevar a cabo el desarrollo de la aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores. 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: Visual Studio Code como entorno integrado de desarrollo, HTML5 como lenguaje de programación, ElectronJS como marco de trabajo y Visual Paradigm para el modelado de los procesos. La aplicación debe permitir la configuración de permisos, usuarios y grupos a través de una interfaz web, mediante la cual se agiliza este proceso, evita errores humanos por parte de los administradores y disminuir la complejidad del proceso de migración.

Palabras clave: aplicación, configuración, grupo, usuario, permisos.

Índice de contenidos

Introducción

Con el aumento de uso de las Tecnologías de la Información y las Comunicaciones (TIC), la información fluye sin mesuras, se hace más notorio el impacto de estas en distintas áreas de la sociedad, y crece la necesidad de informatizar diversos sectores, cada día se hace necesario el almacenamiento de datos y su gestión. Cuba no queda fuera de este ámbito, ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las TIC y lograr una cultura digital como una de las características imprescindibles del hombre, lo que facilitaría a la sociedad acercarse más hacia el objetivo de un desarrollo sostenible (RODRÍGUEZ, 2015).

En los últimos años se ha estado llevando a cabo un proceso de informatización de la sociedad, el mismo está presente en todas las actividades que se realizan en las organizaciones, como herramienta necesaria en la gestión y búsqueda de información y como garantía para alcanzar la eficiencia. Buscando la cúspide de una soberanía tecnológica, debido a la necesidad que se presenta en la sociedad cubana de avanzar en los aspectos políticos, económicos, sociales y tecnológicos del país (MIYARES, 2014).

La preparación de las nuevas generaciones en la utilización de las tecnologías de la información y las comunicaciones y el empleo de estas para aumentar la calidad del proceso docente educativo, son elementos que buscan asegurar el futuro del país. La Universidad de las Ciencias Informáticas (UCI) es ejemplo claro de formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática, así como producir aplicaciones y servicios informáticos a partir del vínculo estudio – trabajo como modelo de formación – investigación – producción, sirviendo de soporte a la industria cubana. Por tanto, no ha quedado exenta a los avances tecnológicos de la actualidad persiguiendo una estrategia para informatizar la sociedad cubana (RAMOS et al., 2011)

Para el apoyo de la informatización de la sociedad cubana tenemos el Centro de Software Libre (CESOL) jefe en los procesos de migración hacia tecnologías libres y de código abierto en Cuba. Este ofrece servicios de personalización de sistemas operativos, tanto en el ámbito nacional como de exportación, así como el desarrollo de aplicaciones de escritorio, herramientas de soporte a dicho proceso y la migración de los servicios telemáticos (PÉREZ, GARCÍA y GOÑI, 2015).

Una migración a tecnologías de software libre y código abierto puede definirse como: "Un proceso ordenado en el cual se sustituye, parcial o totalmente, el software existente en la organización por

alternativas liberadas bajo licencias libres o de código abierto. Es recomendado incluir la adopción de estándares abiertos para la documentación" (VIEITES, 2011).

Dentro de los logros más notables de CESOL se encuentra Nova, una distribución de GNU/Linux desarrollada para apoyar la migración del país a tecnologías de software libre y código abierto. Nova permite realizar trabajos de oficina, navegar por Internet, reproducir archivos de música y video, ver fotografías y múltiples aplicaciones útiles para el desempeño laboral. Es un sistema cómodo y enfocado al usuario final. Esta desplegada en cuatro variantes, Nova Escritorio, Nova ligero, NovaDroid y Nova Servidores, este último con el fin de administrar servicios telemáticos. Según la estrategia para la migración a aplicaciones de código abierto, la migración de servicios telemáticos es el primer paso en el proceso de ejecución de esta (PÉREZ, 2015). La migración a Nova Servidores es de suma importancia en este proceso ya que el primer paso para migrar en una institución es la migración de los servicios telemáticos y para esto se necesita instalar Nova Servidores lo cual garantiza cambios y mejoras, así como avances y desarrollo en cuanto al proceso de informatización.

Debido a la importancia de las aplicaciones desplegadas en los servidores y al valor de los datos que en ellos se almacenan, se aplican medidas de control de acceso y autorización al control de acceso en cada computadora como medios de seguridad y respaldo. El control de acceso es un sistema electrónico que restringe o permite el acceso de un usuario o grupo de usuarios a un área específica validando la identificación por medio de diferentes tipos de lectura. Un control de acceso requiere flexibilidad para que no haya limitaciones en la movilidad por cambios que se producen en los permisos. Necesita precisión para que se le asigne el permiso correcto a cada persona y también es necesario que tenga suficiente capacidad para almacenamiento y registro de un mínimo de datos (DE OLIVEIRA, 2020).

La implantación del control de acceso en un sistema informático depende fundamentalmente de la gestión de cuentas de usuarios y de la gestión de permisos y privilegios. El modelo de seguridad aplicado en el control de acceso se basa en la definición y gestión de determinados objetivos lógicos (dispositivos lógicos, ficheros, servicios) y sujetos (usuario y grupo, equipos, procesos, roles) a los que se conceden derechos y privilegios para realizar determinadas operaciones sobre objetos. Estos derechos y privilegios se pueden verificar mediante el proceso de autorización de acceso. Existen dos tipos de control de acceso (RAULT, 2015):

Control de Acceso Obligatorio (MAC, del inglés *Mandatory Access Control*) y Control de Acceso Discrecional (DAC, del inglés *Discrecionary Access Control*). En ambos casos están basados en que los permisos de acceso son definidos por el sistema operativo.

En la seguridad de acceso existen la autenticación y la autorización. La autenticación consiste en aportar a la prueba de que la identidad comunicada es la correcta. En la práctica se hace con contraseña, un token, en certificado, con objetos, biométricos e incluso combinados, entre otro. En cuanto a la autorización, esta interviene justo después de la fase de autenticación con éxito. La autorización permite asignar a una entidad los permisos que le son propios y las credenciales que han sido concedidas (RAULT, 2015).

La UCI ha realizado varios procesos de migración en entidades como: Empresa Telemática del Ministerio de la Industria Pesquera (Telemar), Escuela Superior del Partido Comunista de Cuba “Ñico López”, Centro de Cibernética Aplicada a la Medicina (CECAM), la Empresa Constructora de Obras de Arquitectura No 24 (ECOA24) y la Empresa Constructora de Obras de Arquitectura e Industriales No.3 (ECOAIND3). En el año 2015 se realizó un diagnóstico en 10 OACE (Organismos de la Administración Central del Estado) con vísperas a una futura migración a software libre y plataformas de código abierto, estos son: Ministerio de Educación Superior (MES), Ministerio de Educación (MINED), Ministerio de Industrias (MINDUS), Ministerio de Comunicaciones (MINCOM), Ministerio de la Construcción (MICONS), Banco Central de Cuba (BCC), Oficina Nacional de Estadísticas e Información (ONEI), Ministerio de Relaciones Exteriores (MINREX), Ministerio de Salud Pública (MINSAP) y Ministerio de Comercio Exterior (MINCEX) (PALMA, 2019).

Se realizó una entrevista a 10 especialistas de CESOL que han participado en los procesos de migración en las diferentes instituciones migradas, acerca del proceso de configuración de permisos, usuarios y grupos en Nova Servidores (ver Anexo 1), una vez migrados los servicios telemáticos. Como resultado de la misma se detectó que este es uno de los servicios más críticos para las instituciones debido a la concurrencia de los usuarios en el acceso. Actualmente la configuración de permisos, usuarios y grupos en modo de producción se realiza de forma manual a través de una interfaz de consola, lo que trae como consecuencia:

1. Poco dominio de los comandos a utilizar por parte de los administradores de redes en las instituciones, lo que trae incomodidad y dificultades en el momento de ejecutar cualquier comando necesario.
2. Pérdida de tiempo para realizar las configuraciones, por parte de los especialistas en servicios telemáticos una vez realizada la migración, y además por los administradores de redes cuando necesiten cambiar las mismas.
3. Duplicación de esfuerzos en las instituciones con más de un servidor, donde deben trasladarse de uno a otro realizando las configuraciones.

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

¿Cómo facilitar la configuración de permisos, usuarios y grupos en los servidores GNU/Linux?

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

Desarrollar una aplicación de escritorio para facilitar la configuración remota de permisos, usuarios y grupos en Nova Servidores de forma remota.

Para guiar la investigación se propone como **objeto de estudio**: “proceso de configuración de permisos, usuarios y grupos en servidores GNU/Linux” enmarcado en el **campo de acción**: “proceso de configuración de permisos, usuarios y grupos en Nova Servidores”.

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

- Elaborar el marco teórico referencial de la investigación sobre la configuración de permisos, usuarios y grupos en servidores GNU/Linux.
- Diseñar la aplicación de escritorio para la configuración remota de permisos, usuarios y grupos en Nova Servidores.
- Implementar la aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota.
- Evaluar la aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota.

Se define como **hipótesis:** el desarrollo de una aplicación de escritorio facilitará la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota.

Variable independiente: Aplicación de escritorio.

Variable dependiente: la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota.

Para el desarrollo de la investigación se utilizaron los **métodos científicos**:

Métodos teóricos:

- **Analítico-Sintético:** permitió realizar un análisis de la información y de diferentes fuentes bibliografías referentes a la configuración de permisos, usuarios y grupos en servidores GNU/Linux y a partir de su estudio resumir, sintetizar e identificar los elementos fundamentales.

Métodos empíricos:

- **Entrevista:** se realizaron entrevistas al cliente y a especialistas de CESOL para obtener información más detallada que ayude a resolver la problemática planteada, partiendo de las respuestas expuestas (ver Anexo 1).
- **Observación:** se desempeñó con el objetivo de constatar el estado actual de la problemática mediante la observación al proceso de seguimiento de la configuración de permisos, usuarios y grupos (ver Anexo 3).

La estructura del documento de investigación consta de introducción, tres capítulos, conclusiones generales, referencias bibliográficas y anexos**.**

Estructura del contenido:

Capítulo 1: “Fundamentación teórica sobre la configuración de permisos, usuarios y grupos en servidores GNU/Linux”: en el presente capítulo se abordan los principales conceptos y definiciones asociados a la configuración de permisos, usuarios y grupos en servidores GNU/Linux y específicamente en Nova Servidores. Se realiza un estudio de herramientas homólogas en busca de una solución para desarrollar una aplicación de escrito para configurar permisos, usuarios y grupos en Nova Servidores. Se describe la metodología de desarrollo de software, las tecnologías, lenguajes y herramientas para el modelado e implementación de la solución.

Capítulo 2: “Análisis y diseño de la propuesta de una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota”: se plantea una propuesta de solución al problema antes expuesto, se definen requisitos funcionales y no funcionales de la aplicación. Más tarde se realiza la descripción de los requisitos funcionales mediante las Historias de Usuario, los prototipos de interfaz y los Casos de Pruebas, la arquitectura y los patrones de diseño.

Capítulo 3: “Implementación y pruebas de la propuesta de una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota”: se implementa la solución propuesta, se plantean el estándar de codificación y el diagrama de despliegue. Además, se confecciona el plan de pruebas y se realizan las pruebas necesarias para verificar que las funcionalidades implementadas dan solución a los requisitos planteados. Finalmente se realiza un análisis crítico sobre la importancia y aporte de la aplicación.

Capítulo 1: Fundamentación teórica sobre la configuración de permisos, usuarios y grupos en servidores GNU/Linux

En el presente capítulo se contempla la fundamentación teórica de la investigación y se abordan los conceptos fundamentales que tienen relación con el objeto de estudio. Se valoran los sistemas existentes relacionados al dominio de la problemática y se realiza una breve descripción de las tecnologías a utilizar definidas por CESOL.

1.1 Conceptos principales

A continuación, aparecen los conceptos principales asociados al tema de investigación:

Permiso: tener el control completo de un sistema, lo que permite realizar operaciones sobre este (RAULT, 2015).

Permiso de acceso: también llamados permisos o privilegios, hace referencia a la cantidad de dispositivos electrónicos que existen en el mercado para autorizar la entrada únicamente a personas previamente identificadas a zonas restringidas. La implantación del control de acceso en un sistema informático depende fundamentalmente de la gestión de cuentas de usuarios y de la gestión de permisos y privilegios. El modelo de seguridad aplicado en el Control de acceso se basa en la definición y gestión de determinados objetivos lógicos (dispositivos lógicos, ficheros, servicios) y sujetos (usuario y grupo, equipos, procesos, roles) a los que se conceden derechos y privilegios para realizar determinadas operaciones sobre objetos (RAULT, 2015).

Autorización: permite asignar a una entidad los permisos que le son propios y las credenciales que le han sido concedidas (RAULT, 2015).

Usuario: un usuario es aquel individuo que utiliza de manera habitual un producto, o servicio. Es un concepto muy utilizado en el sector informático y digital. Los usuarios pueden distinguirse teniendo en cuenta los servicios de los que hagan uso. Por ejemplo, un usuario de un establecimiento deportivo, se refiere a aquel que asiste habitualmente a un centro para practicar algún tipo de deporte. En cambio, un usuario informático, es el que utiliza diversos programas, o navega en Internet (PEIRO, 2020).

Grupo: un grupo se refiere a un conjunto de seres, individuos, o incluso cosas, que se pueden identificar por su proximidad, sus características comunes, o cualquier otra característica que permite diferenciarlos. No sólo se habla de grupos de individuos, sino que este concepto se extiende a casi todos los ámbitos. Cuando de informática se habla, un grupo es una recopilación de usuarios que pueden compartir archivos y otros recursos del sistema. Por ejemplo, usuarios que trabajan en el mismo proyecto podrían formarse en un grupo. Cada grupo debe tener un nombre, un número de identificación de grupo (del inglés GID *group identifier*) y una lista de nombres de usuario que pertenecen al grupo. Un número GID identifica el grupo internamente para el sistema. Los dos tipos de grupos al que un usuario puede pertenecer son los siguientes (RAFFINO, 2020):

- **Grupo primario –** Especifica un grupo que el sistema operativo asigna a archivos creados por los usuarios. Cada usuario debe pertenecer a un grupo primario.
- **Grupos secundarios –** Especifica uno o más grupos a los que los usuarios pueden pertenecer. Los usuarios pueden pertenecer a hasta 15 grupos secundarios.

1.2 Caracterización de sistema de archivos en Linux

Los permisos de Linux básicamente se aplican sobre archivos y directorios, por lo que se hace necesario primeramente conocer la estructura de estos en el sistema. En un sistema Linux, los archivos se ubican en unas carpetas y subcarpetas determinadas, facilitando así que las bibliotecas instaladas (funciones ya programadas) sean compartidas por diversas aplicaciones. Esta estructura se desarrolló en el proyecto FHS (Estándar de Jerarquía de Sistema de Archivos, del inglés *Filesystem Hierarchy Standard*), que especifica los nombres, ubicaciones, contenidos y permisos de un sistema de archivos en sistemas Linux. Esto implica que, salvo ligeras diferencias, todas las distribuciones sitúan los mismos archivos de configuración en las mismas carpetas (TORRES y PIZARRO, 2016).

En un sistema Linux, se encuentran los directorios que se muestran en la Tabla 01:

Tabla 01: Estructura del sistema de ficheros en Linux

(Fuente: TORRES y PIZARRO, 2016).

Directorio	Descripción

/	Directorio raíz, que contiene el resto de los directorios.
/bin	Archivos binarios, ejecutables por el usuario.
/boot	Directorio que contiene los archivos necesarios para arrancar el sistema.
/dev	Archivos de dispositivos.
/etc	Archivos de configuración y arranque del sistema, para las aplicaciones instaladas y los servicios de red. Se agrupan en subdirectorios.
/home	Contiene subdirectorios para cada uno de los usuarios del sistema.
/lib	Directorio de las bibliotecas compartidas, utilizadas por los ejecutables del sistema y por los programas instalados.
/media	Puntos de montaje temporal para medios extraíbles como CD/DVD o USB.
/mnt	Punto de montaje temporal para sistemas de archivos externos.
/opt	Complementos de programas. Se agrupan en subdirectorios.
/proc	Directorio virtual de procesos. Consiste en un sistema virtual de archivos mediante el que el *kernel* se comunica con los usuarios y les informa de los procesos que está desarrollando.
/root	Directorio del superusuario, separado de los restantes usuarios por seguridad. Solo pueden acceder los usuarios administradores del sistema.
/srv	Servicios ofrecidos por el sistema
/sys	Información del sistema
/sbin	Contiene los ejecutables que son fundamentales para el funcionamiento del sistema, los que permiten su arranque y recuperación.
/tmp	Directorio donde se ubican temporalmente los archivos.
/usr	Directorio donde se incluyen los programas instalados por los usuarios, aunque también existen otras aplicaciones que son propias del sistema.
/var	Directorio que contiene los archivos variables y temporales. Se refiere a los datos que varían conforme se hace uso del sistema.

En los siguientes epígrafes se describe cómo se realiza la configuración de permisos, usuarios y grupos.

1.3 Configuración de usuarios

GNU/Linux es un sistema multiusuario en el que conviven simultáneamente diferentes cuentas, desempeñando roles muy diversos. Las características que definen la cuenta de un usuario son (GÓMEZ, 2014):

- Tiene un nombre y un identificador de usuario (UID) únicos en el sistema.
- Pertenece a un grupo principal.
- Puede pertenecer a otros grupos de usuarios.
- Puede definirse una información asociada con la persona propietaria de la cuenta.
- Tiene asociado un directorio personal para los datos del usuario.
- El usuario utiliza en su conexión un determinado intérprete de mandatos, donde podrá ejecutar sus aplicaciones y las utilidades del sistema operativo.
- Debe contar con una clave de acceso personal y difícil de averiguar por parte de un impostor.
- Tiene un perfil de entrada propio, donde se definen las características iniciales de su entorno de operación.
- Puede tener una fecha de caducidad.
- Pueden definirse cuotas de disco para cada sistema de archivos.
- Es posible contar con un sistema de auditoria que registre las operaciones realizadas por el usuario.

1.3.1 Clasificación de usuarios

Un usuario es el conjunto de privilegios, permisos, recursos o dispositivos, a los cuales se tiene acceso el cual puede ser tanto una persona, maquina u otro programa (GÓMEZ, 2011).

Root o superusuario: (UID = 0), que es el único que tiene privilegios sobre todo el sistema, y el responsable de las tareas de administración del sistema, tales como la instalación y desinstalación de

software, entre otras muchas. Para cualquier acción que necesite permisos de superusuario, el sistema requerirá las credenciales de root.

Usuarios de sistema: son los que van vinculados a ciertos servicios, y que pueden asumir ciertos privilegios relativos a este servicio. Se crean automáticamente en la instalación del sistema o de ciertas aplicaciones, como es el caso del antivirus ClamAv.

Algunos ejemplos de estos usuarios son bin, mail, apache, clamav, pulse, avahi, syslog, colord, etcétera, pero realmente hay muchos más en cualquier instalación estándar.

Usuarios estándar: puede haber tantos como se requiera. Cada usuario estándar posee su directorio personal dentro de la */home*. Allí se almacenan todos sus archivos personales, además de las preferencias de varias aplicaciones, archivos temporales, etcétera.

1.3.2 Comandos utilizados

A continuación, se describen los comandos utilizados para la configuración de usuarios (TORRES y PIZARRO, 2016):

1.3.2.1 adduser

Crea un usuario nuevo con su directorio /home. Por ejemplo: *adduser amidala* crearía el usuario amidala junto con el directorio */home/amidala*. Este comando pide información sobre el nuevo usuario que se crea y su contraseña. Además, añade los archivos de configuración que se localizan en el directorio */etc/skel*.

También puede ejecutar el comando useradd nuevousuario.

Para ver las opciones de este comando puede ejecutar: *man adduser*

Por defecto, al crearse un usuario se crea un grupo con su mismo nombre, es decir que el identificador de usuario UID y el identificador de grupo GID serán iguales. Las opciones de configuración de este comando se encuentran en el archivo */etc/adduser.conf*. Se puede usar también para añadir un usuario ya creado a un grupo ya existente: *adduser usuario grupo.*

Con la opción: *adduser --group nombregrupo,* se añade al sistema un grupo de usuarios vacío. Esta opción equivale al comando addgroup.

Otra opción útil de este comando es cuando se especifica un directorio principal distinto:

useradd -d /home/otrodirectorio

Para llevar un control más claro de cada usuario, y debido a que el nombre de usuario podría no proporcionar información suficiente, puede especificar el nombre completo del usuario usando la opción -c, por ejemplo:

useradd -c "Julia Schneider" julia

1.3.2.2 deluser

Elimina un usuario determinado. Se utiliza de la forma: *deluser nombreusuario*. Este comando no borra el directorio /home/usuario. Para borrarlo hay que añadir:

deluser nombreusuario --remove-home

Para quitar de un grupo a un usuario se utiliza:

deluser nombreusuario nombregrupo

1.3.2.3 passwd

Una vez que se crea un nuevo usuario (o en cualquier momento que se considere necesario), se debe crear una contraseña asociada a dicho usuario. Siempre se recomienda usar contraseñas con un nivel de seguridad alto:

- Combine letras mayúsculas, minúsculas y números.
- No use secuencias de números.
- Agregue signos como /, *, -, +, _, ?, ..

El comando utilizado para establecer contraseñas es passwd seguido por el nombre del usuario. Por ejemplo, para establecer la contraseña del usuario julia se utiliza:

passwd julia

Su shell le pedirá que indique la contraseña y que luego la confirme. Por seguridad Linux no muestra su contraseña ni caracteres ocultos mientras escribe, así que debe hacerlo con cuidado. A continuación, un ejemplo de la salida en su terminal al usar el comando anterior:

Changing password for user julia.

New password:

Retype new password:

passwd: all authentication tokens updated successfully.

Existen opciones que puede utilizar junto con el comando passwd para bloquear la contraseña (-l o --lock), desbloquearla (u o --unlock) e incluso para establecer el tiempo máximo de vida de la contraseña: --maximum=DÍAS.

Por ejemplo, si se desea que la contraseña del usuario julia expire en 90 días:

passwd --maximum=90 julia

Se sabe si el procedimiento fue exitoso si el mensaje de salida es similar al siguiente:

Adjusting aging data for user julia.

passwd: Success

1.3.2.4 usermod

Con el comando usermod se pueden realizar varios cambios en la configuración de los usuarios del sistema. Algunos ejemplos a continuación.

Para cambiar el nombre de inicio de sesión de un usuario:

usermod -l nombreantiguo nombrenuevo

Para cambiar el directorio /home de un usuario se utiliza:

usermod -d /nuevo/directorio -m nombredeusuario

Para agregar el usuario a otros grupos suplementarios:

usermod -G grupo1,grupo2,grupo3

Use el parámetro --expiredate seguido por la fecha en formato AAAA-MM-DD para establecer la fecha de expiración de la cuenta de usuario:

usermod --expiredate 2017-01-02 nombredeusuario

Para bloquear la contraseña de un usuario:

usermod --lock nombredeusuario

Para desbloquear la contraseña de un usuario:

usermod --unlock nombredeusuario

1.3.2 chage

El comando chage (**ch**ange **age**) cambia el número de días entre los cuales debe cambiar la contraseña. Cuando el usuario inicie sesión, aparecerá un mensaje indicando que debe cambiar la contraseña antes de que expire.

Para ver información con respecto a la contraseña de un usuario ejecute:

chage --list nombredeusuario

Para establecer el número de días máximo para la expiración de la contraseña se utiliza la opción -M seguido del número de días. Por ejemplo: para que la contraseña expire cada 90 días:

chage -M 90 nombredeusuario

1.4 Configuración de grupos

Para simplificar la gestión de permisos entre tantas cuentas, Linux agrupa todos los usuarios dentro de grupos. De este modo, a la hora de especificar ciertos permisos sobre un conjunto de cuentas, se pueden establecer los permisos directamente sobre el grupo. Cada grupo se identifica por un Group ID o GUID.

Cada usuario pertenece por defecto a un grupo con el mismo nombre, que se conoce como grupo principal de usuario o grupo primario. Además, un mismo usuario puede estar dentro de otros grupos, que serían los secundarios.

1.4.1 Comandos utilizados

A continuación, se describen los comandos utilizados para la configuración de grupos (TORRES y PIZARRO, 2016):

1.4.1.1 groupadd

Con el comando *groupadd* se añade un grupo. Se utiliza de la forma:

groupadd nombregrupo

Para crear un nuevo grupo del sistema añada el parámetro -r al comando groupadd:

groupadd -r nombredelgrupo

1.4.1.2 groupdel

Este comando elimina un grupo de usuarios que esté vacío. Se utiliza de la forma:

groupdel nombregrupo

También puede ejecutar:

delgroup nombredelgrupo

1.4.1.3 groups/id

Se pueden usar los siguientes comandos para identificar los grupos a los cuales pertenece un usuario en particular:

groups nombredeusuario

id nombredeusuario

1.4.1.4 gpasswd

Se pueden agregar usuarios a un grupo utilizando:

gpasswd -a nombredeusuario nombredegrupo

Para remover un usuario de un grupo al cual pertenece:

gpasswd -d nombredeusuario nombredegrupo

1.4.1.5 groupmod

El comando groupmod permite hacer algunos cambios a los grupos disponibles.

Para cambiar el nombre de un grupo se utiliza:

groupmod -n gruponuevo grupoantiguo

1.5 Ficheros asociados a la configuración de usuarios y grupos

La configuración de usuarios y grupos en Linux esencialmente se maneja en los tres siguientes ficheros.

1.5.1 /etc/passwd

Contiene información de las cuentas de usuarios y sus características, como se aprecia en la Tabla 2.

Tabla 2: Campos del fichero /etc/passwd

(Fuente: TORRES y PIZARRO, 2016).

Campo	**Descripción**
Login	Nombre del usuario
Password	El password del usuario en texto plano, o un asterisco * o una X si está encriptado.
UID	User ID. Número de identificación único de usuario. Los usuarios pueden cambiar muchos prámetros, incluso su name, pero el UID no lo deben cambiar nunca. El UID del root es 0. Las cuentas de servicios y demonios tienen los números más bajos, mientras que las de usuarios finales comienzan en el valor definido en UID_MIN en el fichero /etc/login.defs.
GID	Gruop ID. Número de identificador único de grupo. Varios usuarios pueden tener el mismo grupo, aunque al crear un usuario se crea un grupo con ese mismo nombre por defecto salvo que se indique lo contrario. Los datos del grupo aparecen en /etc/group.
GECOS	Campo de comentarios que incluye información extra sobre el usuario (nombre real, dirección…) Informalmente se le llama información finger.
directory	Home directory. Directorio de inicio del usuario. Los usuarios finales se suelen situar bajo /home.
Shell	La shell que utiliza por defecto el usuario (en muchos casos es /bin/bash). Si el usuario tiene /sbin/nologin o /usr/bin/false, significa que no tiene permiso para loguearse en el sistema, lo cual es común en daemons como medida de seguridad.

1.5.2 /etc/shadow

Contiene información sobre contraseñas de los usuarios, almacenada de manera cifrada, como se precia en la Tabla 3.

Tabla 3: Campos del fichero /etc/shadow

(Fuente: TORRES y PIZARRO, 2016).

Campo	Descripción
Login	Nombre del usuario
Password	Password encriptado. La función hash usada para encriptar el password se indica al comienzo.
Lastmod	Tiempo transcurrido desde el último cambio de clave.
Min	Número mínimo de días hasta que se puede volver a cambiar la contraseña.
Max	Número máximo de días hasta que el sistema obliga a cambiar la contraseña del usuario.
Aviso	Número de días previos al Max en los que el usuario es avisado de su obligado cambio de contraseña.
Inactividad	Número de días entre el vencimiento de la contraseña y el bloqueo de la cuenta.
Expiración	Fecha en la que la cuenta se deshabilita. Si se deja en blanco, la cuenta nunca expira.
Reservado	Campo reservado para futuros usos.

1.5.3 /etc/group

Este archivo contiene información sobre los grupos del sistema. Su estructura es similar a la de los archivos passwd y shadow, contando con los campos que se muestran en la Tabla 4.

Tabla 4: Campos del fichero /etc/group.

(Fuente: TORRES y PIZARRO, 2016).

Campo	Descripción
Grupo	Nombre del grupo.
passwd	Password que permite a un usuario cambiar de grupo. Si está vacío no requiere contraseña, y una x significa que se gestiona mediante el archivo /etc/gshadow.
GID	Group ID. Identificador único (numérico) para el grupo.
Miembros	Lista separada por comas con los nombres de usuario que pertenecen a ese grupo.

1.6 Configuración de permisos

Los permisos de Linux básicamente se aplican sobre archivos y sobre directorios, van asociados a usuarios o grupos, y pueden ser de lectura, de escritura o de ejecución. Dicho de otro modo, todos los archivos y directorios en Linux tienen asociado un conjunto de permisos que debe definir las posibilidades de lectura, escritura y ejecución que se aplican al usuario propietario del archivo, al grupo de usuarios al que pertenece, y al resto del mundo (PANEQUE, 2013).

1.6.1 Tipos de permisos a asignar

- **Lectura(r)**: representan la posibilidad de poder acceder a un archivo o carpeta y leer su contenido. En un directorio hacen referencia a la posibilidad de visualizar su contenido.
- **Escritura(w)**: definen la posibilidad de acceder y modificar el contenido de un archivo, o en caso de un directorio, la capacidad de borrar o añadir archivos dentro de él.
- **Ejecución(x)**: estos son los más críticos. Indican la posibilidad de ejecutar un determinado archivo en el sistema. En el caso de un directorio, los derechos de ejecución representan la capacidad de entrar dentro del directorio.
- **-**: el guion indica que el archivo o directorio carece del permiso correspondiente en dicha posición.

1.6.2 Niveles de privilegios

En cualquier fichero o carpeta hay tres tipos o niveles de privilegio, los que se aplican al propietario del archivo o carpeta, los que se aplican a todos los miembros del grupo al que pertenece el propietario, y los que se aplican a todos los demás. A continuación, se describe cada uno:

- **Permisos del Usuario:** es el primer nivel de privilegios. Básicamente representa los permisos que se aplican al propietario de un determinado fichero o directorio.
- **Permisos del Grupo:** el segundo nivel de privilegios, que define los derechos de lectura, escritura y ejecución que se aplican solo a aquellos usuarios que pertenecen al mismo grupo que el propietario del fichero.
- **Permisos de Otros:** este es el último nivel. Representan los privilegios de lectura, escritura y ejecución por parte del resto de usuarios que no entran en ninguno de los niveles anteriores (PONS, 2016).

1.6.3 Comandos utilizados

A continuación, se describen los comandos utilizados para la configuración de permisos (TORRES y PIZARRO, 2016):

1.6.3.1 chmod

Se utiliza para cambiar los permisos de archivos y directorios de los que se es propietario. Solo root puede modificar cualquier archivo. Se utiliza de la forma (TORRES y PIZARRO, 2016):

chmod [opciones] <modo de cambio> <archivo|directorio>

Las opciones más comunes son -R y -v, para cambiar todo el contenido de un directorio y para mostrar las acciones efectuadas en cada archivo, respectivamente. El modo de cambio de permisos puede ser octal o simbólico (TORRES y PIZARRO, 2016).

- **Permisos en el modo octal**

En el modo octal los permisos se expresan con tres números, el primero corresponde al usuario dueño, el segundo al del grupo principal al que pertenece el usuario y el tercero a “otros”, es decir, a los restantes usuarios del sistema. Cada número de usuario se obtiene sumando 4 (si hay permiso de lectura), 2 (si hay permiso de escritura) y 1 (si hay permiso de ejecución). Por ejemplo, si un archivo tiene permisos 740, el usuario los tiene todos (4+2+1), el grupo tiene permisos de lectura (4) y el resto no tiene ninguno (0) (TORRES y PIZARRO, 2016). Por ejemplo:

chmod 0777 archivo.txt	Asigna todos los permisos al archivo archivo.txt
chmod 0666 *	Asigna permisos de lectura y escritura, no de ejecución a todos los archivos y directorios del directorio donde se ejecuta el comando.
chmod -R 0644 *	Asigna permisos a todos los archivos y directorios del directorio donde se invoca el comando y de todos los directorios que están dentro de él. Los permisos asignados son de lectura a todos los usuarios, de escritura sólo al dueño del archivo y de ejecución a nadie.

- **Permisos en el modo simbólico**

En el modo simbólico se especifica si el permiso de lectura (r), escritura (w) y/o ejecución (x), se concede (+) o se quita (-) a un usuario (u), grupo (g), otros (o) o a todos (a). Por ejemplo (TORRES y PIZARRO, 2016):

chmod g -w	Quita el permiso de escritura al grupo al que pertenece el usuario.
chmod o -x, og +w archivo	Quita el permiso de ejecución a otros usuarios, y concede permisos de escritura a otros y al grupo.

1.6.3.2 chgrp

El comando **chgrp** permite cambiar el grupo de usuarios de un archivo o directorio en sistemas tipo UNIX (del inglés *Uniplexed Information and Computing Service*). Cada archivo de Unix tiene un identificador de usuario (UID) y un identificador de grupo (GID), que se corresponden con el usuario y el grupo de quien lo creó.

$ chgrp nuevogrp archivo1 [archivo2 archivo3...]	Cambia a cualquier archivo el grupo.
$ chgrp -R nuevogrp directorio	Cambia el grupo de archivo1 archivo2, etc. que pasará a ser nuevogrp.

Todos y cada uno de los ficheros en Linux son propiedad de un usuario y de un grupo, y sobre cada una de ellos recaen unos privilegios relativos al propietario, al grupo, y al resto del mundo (TORRES y PIZARRO, 2016).

1.7 Estudio de sistemas homólogos para el desarrollo de una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores

Se realiza un estudio de herramientas homólogas en busca de una solución para desarrollar una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota. 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. Ejemplo de estas herramientas tenemos:

1.7.1 users-admin

Para la administración de usuarios, GNU/Linux dispone de la herramienta gráfica users-admin, como se aprecia en la Figura 1. 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 (GARCIA, 2017).

Figura 1: Interfaz de la herramienta users-admin

Fuente:(GARCIA, 2017)

1.7.2 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 (PC que se utiliza para prestar servicios dedicados, generalmente relacionados con el alojamiento web y otros servicios en red.) 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. Cuenta entre sus funciones con un panel para la administración de usuarios y grupos del sistema operativo (DOCUMENTACION cPANEL, 2021).

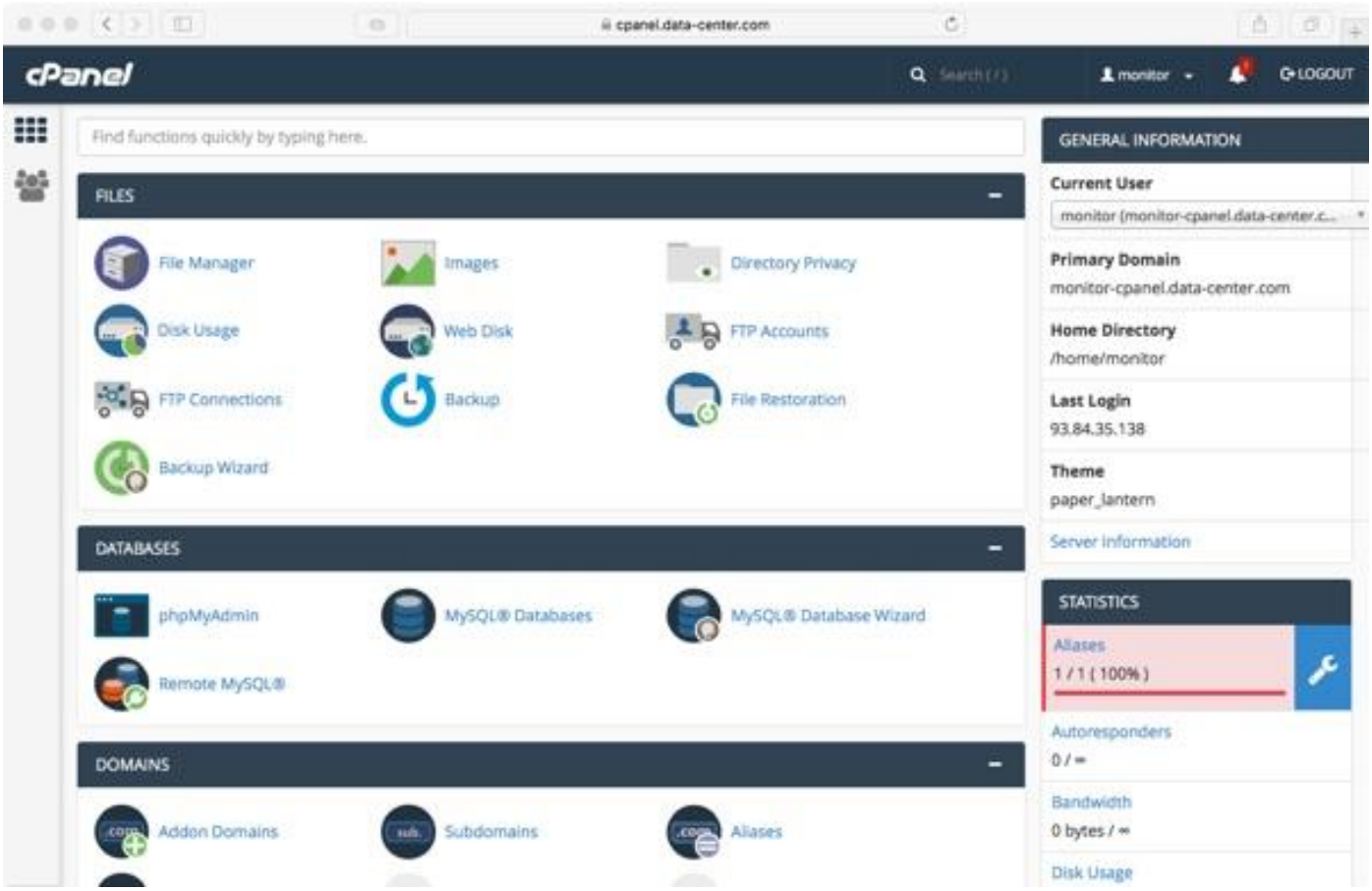

Figura 2: Interfaz de la herramienta cPanel

Fuente: (DOCUMENTACION cPANEL, 2021)

1.7.3 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,

entre otros. Su interfaz es muy sencilla y permite realizar cualquier tarea de gestión sin necesidad de ingresar una línea de código (PLESK INTERNATIONAL GMBH, 2021).

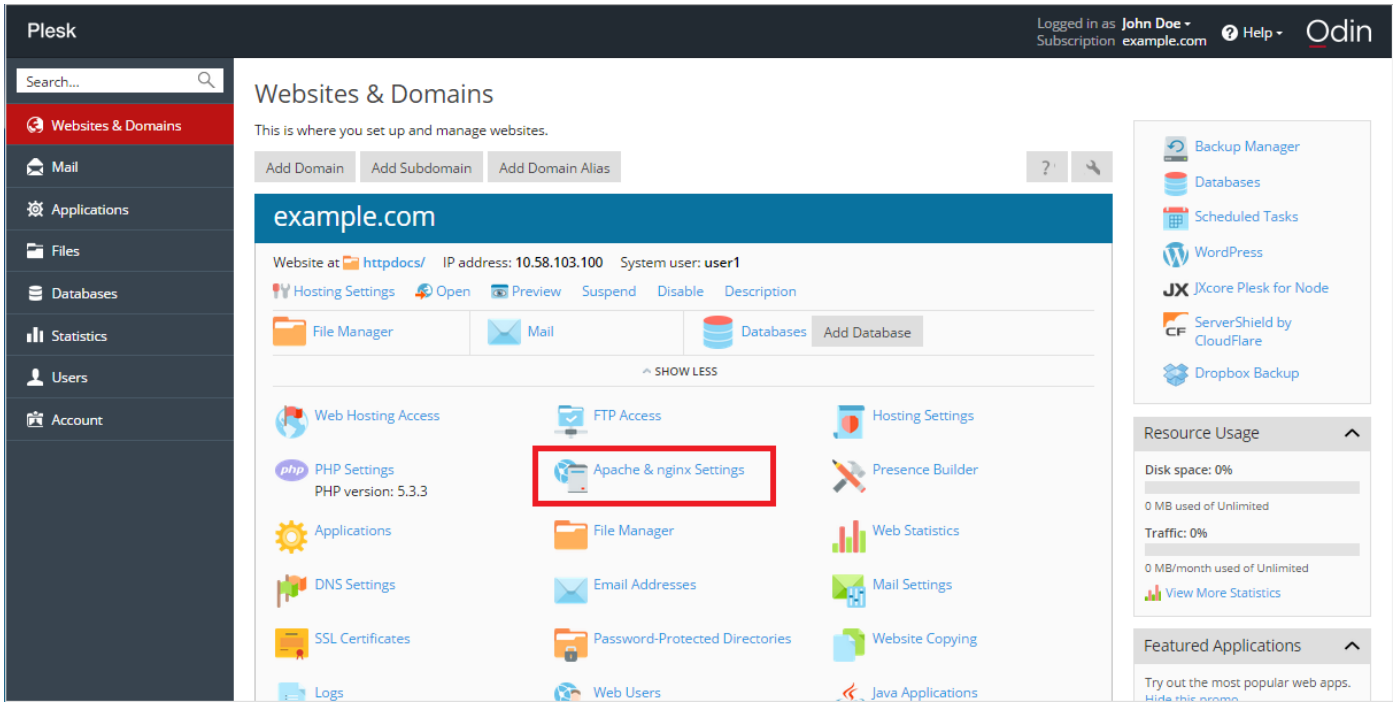

Figura 3: Interfaz de la herramienta pLESK

Fuente: (PLESK INTERNATIONAL GMBH, 2021)

1.7.4 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 (YaST/INTRODUCCION – OPENSUSE, 2021).

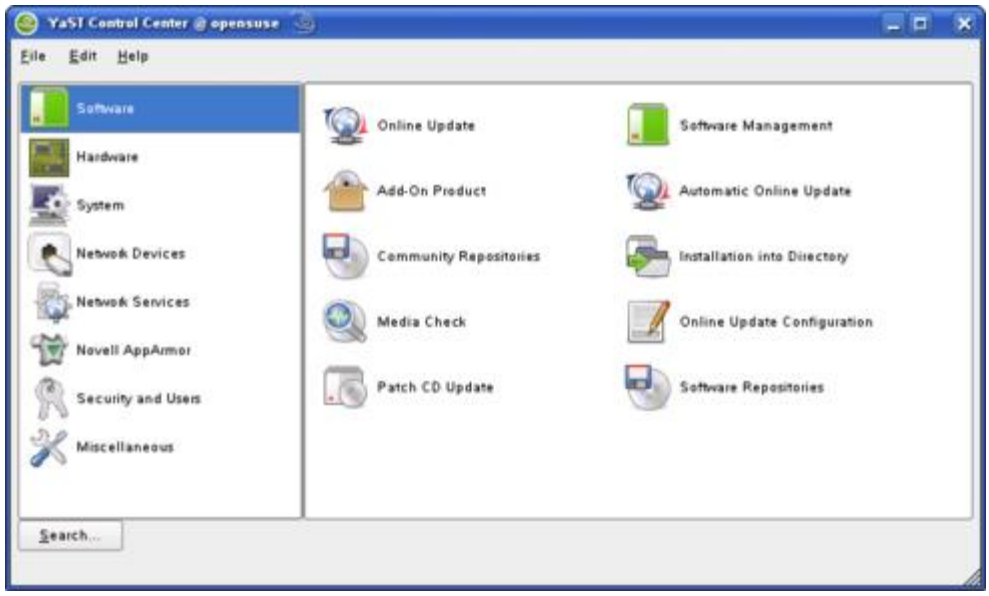

Figura 4: Interfaz de la herramienta YaST

Fuente: (YaST/INTRODUCCION – OPENSUSE, 2021)

1.7.5 Zentyal

Se desarrolló con el objetivo de acercar Linux a las pequeñas y medianas empresas y permitirles aprovechar todo su potencial como servidor de empresa. Es la alternativa de código abierto a los productos de Microsoft para infraestructuras como las que utilizan 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 (ZENTYAL LINUX SMALL BUSINESS SERVER, 2021).

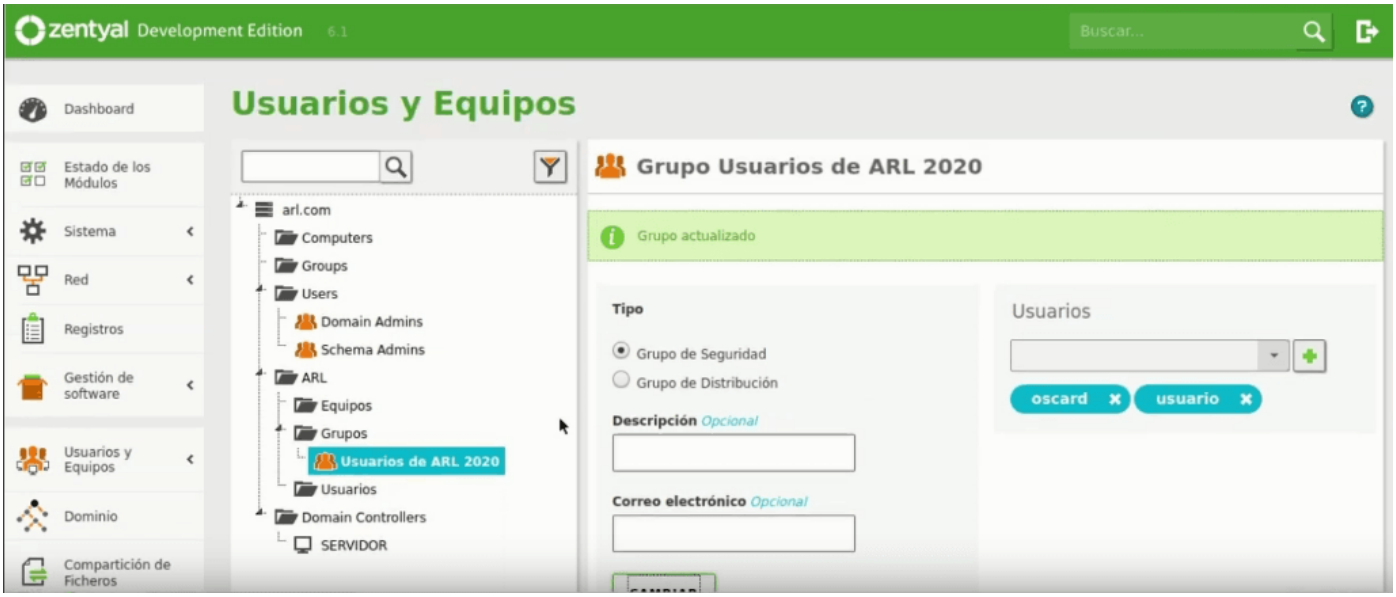

Figura 5: Interfaz de la herramienta Zentyal

Fuente: (ZENTYAL LINUX SMALL BUSINESS SERVER, 2021)

1.7.6 Webmin

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 (del inglés *Hypertext Preprocessor*), MySQL (del inglés *My Structured Query Lenguaje*), DNS (del inglés *Domain Name System*), Samba, DHCP (del inglés *Dynamic Host Configuration Protocol*), 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 (UBUNTU, 2021).

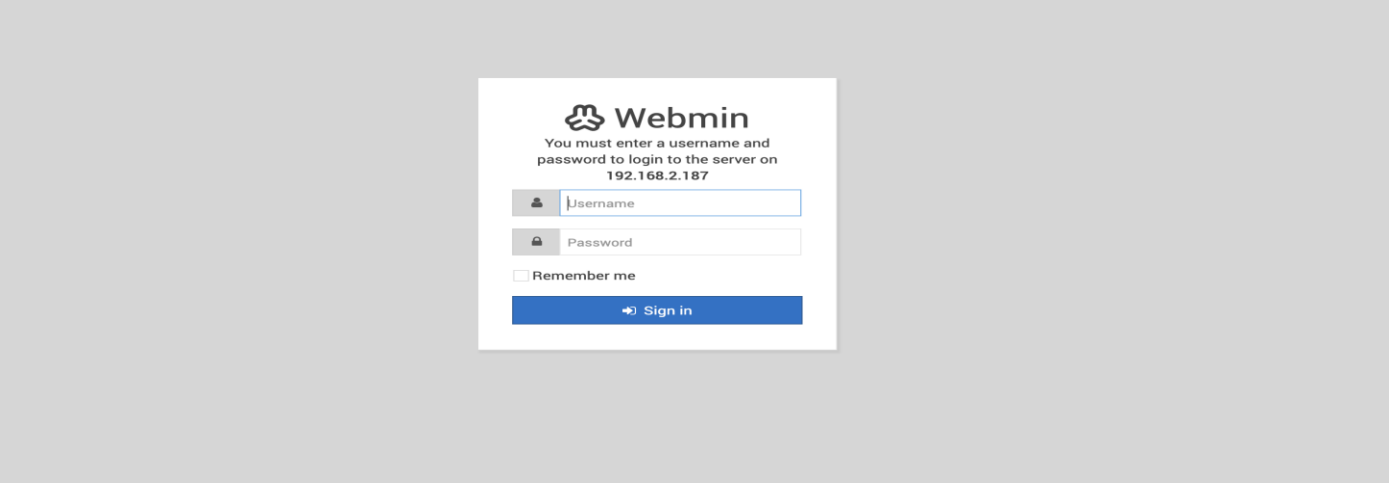

Figura 6: Interfaz de la herramienta Webmin

Fuente:(UBUNTU, 2021)

1.7.7 Resultado del estudio de herramientas homólogas

Se realiza una comparación entre las herramientas ya existentes y las funcionalidades de la posible aplicación a desarrollar para configurar permisos, usuarios y grupos en Nova Servidores (ver Tabla 5):

Tabla 5: Comparación de herramientas homólogas

Fuente: Elaboración propia

	Términos comparativos					
Herramienta	**De pago**	**Fácil manejo**	**Licencia de pago**	**Compatible con varios Sistemas operativos**	**Administración remota**	**Configuración Usuario Grupos Permiso**
Users-admin	**sí**	**sí**	**no**	**sí**	**no**	**no**
Webmin	**no**	**sí**	**no**	**sí**	**sí**	**sí**
Zentyal	**sí**	**no**	**no**	**sí**	**no**	**no**

YaST	**no**	**no**	**sí**	**sí**	**sí**	**no**
Plesk	**sí**	**sí**	**sí**	**sí**	**sí**	**sí**
cPanel	**sí**	**sí**	**sí**	**sí**	**no**	**sí**
Aplicación por desarrollar	**no**	**sí**	**no**	**sí**	**sí**	**sí**

Luego de la comparación de las herramientas homólogas se puede concluir que ninguna satisface completamente la necesidad de configurar permisos, usuarios y grupos en Nova Servidores. En algunas se depende para acceder a estas de la web como Webmin, memorizar comandos cosas que se hace compleja para el usuario, algunas son de pago como Zentyal,, no son fáciles de usar como YaST, algunas utilizan licencias de pago como Cpanel mientras que la aplicación propuesta debe definirse por ser grata a la vista y fácil de manejar tanto para usuarios experimentados como para los menos experimentados, se planea que sea una aplicación totalmente gratis y accesible donde sea que esta se encuentre instalada, sencilla de administrar, con funcionamiento remoto, desarrollada con herramientas dominadas por el desarrollador y con tipo de metodología ágil, facilitando así el trabajo en Software Libre.

1.8 Tecnologías para el desarrollo de la propuesta de solución

Para el proceso de desarrollo de la aplicación de escritorio para la configuración de permisos, usuarios y grupos se han seleccionado un conjunto de herramientas y tecnologías que ayudan al diseño, la modelación e implementación de la solución propuesta, con el objetivo de obtener resultados positivos, las cuales fueron establecidas por la dirección de CESOL.

1.8.1 Lenguajes y herramientas para el modelado

Las herramientas CASE (Ingeniería de Software Asistida por Computadora, del inglés *Computer Aided Software Engineering*) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos en el ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, implementación de parte del código automáticamente

con el diseño dado, compilación automática, documentación o detección de errores entre otras (BELTRÁN, 2019).

Visual Paradigm v8.0

Herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue (para metodologías de ciclo completo). El software de modelado UML agiliza la construcción de aplicaciones de calidad. Propicia además un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación hasta el análisis y el diseño, además permite la correcta realización de los artefactos propuestos por la metodología AUP adaptada para la UCI (VARELA, 2017). Se utiliza una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (Community Edition, ya que existe la Enterprise, Professional). Fue diseñado para una ámplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. Se selecciona la herramienta CASE Visual Paradigm for UML 2.0 (PARADIGM, 2018) ya que es gratuita y se encuentra disponible para distribución Linux. Tiene una curva de aprendizaje muy pronunciada, ofreciendo conectividad limitada con otras aplicaciones por lo que se debe trabajar prácticamente solo en su entorno.

1.8.2 Marcos de trabajo

Un marco de trabajo o *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener (DÍAZ, 2018).

ElectronJS v13.0.1

Es un marco de código abierto desarrollado por GitHub3 para crear aplicaciones de escritorio utilizando HTML, CSS y JavaScript. Una de las ventajas principales del uso de Electron es que, dado que se basa en tecnologías web, es multiplataforma, lo que permite implementar aplicaciones para Linux, MacOS y Windows, con el mismo código. También cuenta con elementos nativos, como menús y notificaciones, así como herramientas de desarrollo útiles para la depuración y el informe de errores (ELECTRO, 2019). Es la plataforma que se utiliza para desarrollar la aplicación de escritorio y gracias a que es el mismo lenguaje que la web podemos compartir módulos entre nuestro frontend y backend con la aplicación de escritorio sin problemas. Se hace la selección de este marco de trabajo porque su capacidad de multiplataforma nos

permite convertir tecnologías web en aplicaciones para Linux que es en este caso el sistema operativo a operar.

React o reactJS v16.8.6

Es una librería JavaScript OpenSource que crea interfaces de usuario. Diseñada para desarrollar aplicaciones web complejas que utilizan un gran intercambio de datos. React solo se preocupa de la interfaz de usuario de tu app; si se piensa en el paradigma de software Modelo-Vista-Controlador (MVC) React es únicamente la Vista. Puede utilizarse combinando con otras librerías JavaScript o con frameworks como AngularJS, Ember o Backbone. También genera una gran biblioteca de extensiones, las cuales en muchas ocasiones no se ocupan solo de la interfaz de usuario, sino que te ayudarán a complementarlas necesidades de la App (JIMENEZ, 2015). Se utiliza esta librería principalmente por que genera el DOM (Modelado de objeto de documento, estructura de los elementos que se generan en el navegador web la cargar una página de forma dinámica).

.NET v4.8

Es un framework que hace un énfasis en la transparencia de , con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones. Basada en ella, la empresa intenta desarrollar una estrategia horizontal que integre sus productos, desde el sistema operativo hasta las herramientas de mercado. Ofrece una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo (GONZALEZ, 2020). Se elige este framework ya que realiza un control automático del código para que este sea seguro, así como el código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio.

1.8.3 Marco de diseño

Tailwind CSS v2.2

Es un marco CSS de utilidad para crear rápidamente interfaces de usuario personalizadas, de bajo nivel altamente personalizable que le brinda todos los componentes básicos que necesita para crear diseños a medida sin ningún estilo molesto con opiniones que tenga que luchar para anular. No impone especificaciones de diseño o cómo debería verse su sitio, simplemente reúne pequeños componentes

para construir una interfaz de usuario única. Lo que Tailwind simplemente hace es tomar un archivo CSS "sin procesar", procesar este archivo CSS sobre un archivo de configuración y generar una salida (RAPPIN, 2021). Se selecciona este marco de diseño debido a que crea pequeñas utilidades con un conjunto definido de opciones que permiten una fácil integración de clases existentes directamente en el código HTML.

1.8.4 Entorno de Desarrollo Integrado (IDE)

Un IDE (del inglés *Integrated Development Enviroment*) es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI) (ZORRILLA, 2016).

Visual Studio Code v1.62.2

Visual Studio Code es un editor de código ligero y potente que está disponible para los sistemas operativos de Windows, Linux y MacOS. Viene con soporte incorporado para JavaScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes como: C++, C#, Java, Python, PHP, GO, entre otros. La elección de esta herramienta se debe a que, desde sus inicios, Angular ha estado muy ligado a este IDE por lo que se ha adoptado como el preferido por la comunidad a la hora de desarrollar aplicaciones de este framework. Las características más importantes de este programa es la adopción de la tecnología de autocompletado IntelliSense propias de las versiones completas de Visual Studio, la ligereza con respecto a otros competidores y el hecho de que, a diferencia de las demás versiones de Visual Studio, es gratuito. (BALLESTER, 2018). Se selecciona esta herramienta por su fácil manejo por parte del desarrollador y su biblioteca de extensiones, las cuales dan un sin número de opciones para ser más eficiente a la hora de estar programando. Desde extensiones de otros lenguajes de programación como diferentes herramientas para visualizar tu código de manera más eficiente.

1.8.5 Herramienta de control de versiones

Se llama control de versiones a los métodos y herramientas disponibles para controlar todo lo referente a los cambios en el tiempo de un archivo. Es un método estándar para mantener esta memoria haciendo además que sea útil para el desarrollo futuro. Un sistema de control de versiones (*VCS* del inglés *Version Sistem Control*,) también significa generalmente que, al perder archivos, será posible recuperarlos fácilmente (BORRELL, 2006).

Git v2.33

Git maneja sus datos como un conjunto de copias instantáneas de un sistema de archivos miniatura. Cada vez que se confirma un cambio, o se guarda el estado del proyecto en el, él básicamente toma una foto del aspecto de todos los archivos en ese momento y guarda una referencia a esa copia instantánea. Para ser eficiente, si los archivos no se han modificado, no almacena el archivo de nuevo, sino un enlace al archivo anterior idéntico que ya tiene almacenado, maneja sus datos como una secuencia de copias instantáneas. Se utiliza Git para controlar los cambios realizados en las versiones de la herramienta en desarrollo. (CHACON, 2014)

Lenguaje Unificado de Modelado

El Lenguaje de Modelado Unificado (UML, en *inglés Unified Modeling Language*) es un lenguaje que se centra en la representación gráfica de un sistema, por lo que permite construir, modelar y diseñar dicho sistema. Un modelo UML está compuesto por 3 clases de bloques de construcción: los elementos (representaciones abstractas de cosas reales o ficticias); las relaciones (relacionan los elementos entre sí); y los diagramas (son colecciones de los elementos con sus relaciones) (LUCIDCHART, 2019). Se utiliza para comprender mejor el sistema que queremos desarrollar mediante la descripción de modelos una de sus características fundamentales.

Su utilidad se basa en la incorporación de toda una serie de diagramas y notaciones gráficas y textuales destinadas a mostrar el sistema desde las diferentes perspectivas, que pueden utilizarse en las diferentes fases del ciclo de desarrollo del software. No dispone de una metodología de desarrollo, por lo cual de apoya en metodologías de terceros para el desarrollo de los proyectos en este caso es coherente con la metodología AUP-UCI siendo esta de enfoque ágil. Mientras que el Visual Paradimg es compatible con el UML ya equilibra propiedades, centra la arquitectura en minimizar riesgos y organizar el desarrollo.

1.8.6 Lenguajes

El lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión (VALIENTE, 2015). A continuación, se plantean los lenguajes utilizados para el desarrollo de la aplicación de escritorio para configuración de permisos, usuarios y grupos.

HTML5 v1.1.2

HTML (Lenguaje de Formato de Documentos para Hipertexto, del inglés *HyperText Markup Language*,) es un lenguaje que se utiliza para la construcción de sitios web y aplicaciones. Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. Permite scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son JavaScript y PHP. La versión 5 de HTML establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos (PARADIGM, 2018). Provee básicamente tres características: estructura, estilo y funcionalidad. Se utiliza HTML5 en la creación de los *templates* de la propuesta de solución, para establecer su estructura y contenido.

CCS3 v1.10.2

CSS3 (Hojas de estilo en cascada, del inglés *Cascading Stylesheets*) es un lenguaje de diseño gráfico creado por W3C (Consorcio Mundial de la Red, del inglés *World Wide Web Consortium*), para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de las páginas web e interfaces de usuario escritas en HTML o XHTML. Puede ser aplicado a cualquier documento .xml, .xhtml, .svg, .xul, .rss, entre otros. También permite aplicar estilos no visuales, como las hojas de estilo auditivas (W3C, 2019)**.** Es un lenguaje sencillo**,** utiliza selectores para definir los elementos html para y reglas para aplicar los estilos. Este lenguaje también cumple una función vital para el diseño web hoy en día, y es que es el responsable de la funcionalidad responsiva, el decir, que se ajuste el diseño de la página web según la pantalla en la que se visualiza. Se emplea CCS3 utilizará CSS para dar estilo a los *templates* de la propuesta de solución. Maneja unidades de medida y colores rigen la aplicación a desarrollar.

JavaScript v1.8.0

JavaScript es un lenguaje de programación (de los denominados lenguajes de scripting (script se traduce como guión, literalmente)) que se utiliza principalmente para crear páginas web dinámicas, con algunos efectos realmente interesantes y que mejoren considerablemente su aspecto. Está diseñado para ser usado en conjunción con HTML, Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos

intermedios (EGUILUZ, 2020). Es sencillo y rápido, los resultados pueden ser muy satisfactorios y aunque el lenguaje tenga algunas limitaciones, permite al programador controlar lo que ocurre en la página. Se utiliza JavaScript para la implementación de la lógica del negocio. Permite escribir código fuente que será analizado por un ordenador, así como es una es una herramienta de fácil manejo para el desarrollador.

1.9 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 (PALMA, 2013).

1.9.1 Metodología de desarrollo de software variación de AUP para la UCI

La UCI desarrolló una versión de la metodología de desarrollo de software AUP (del inglés, *Agile Unified Process*), de forma tal que se adapte al ciclo de vida definido para la actividad productiva. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma, y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, denominada Ejecución, y agregándose también una nueva fase denominada Cierre (SÁNCHEZ, 2015).

Metodología Variación de AUP está formada por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad, las cuales contienen las características de las cuatro fases (Inicio, Elaboración, Construcción y Transición) propuestas en AUP. Las características de las fases de la metodología de la universidad son (Sánchez, 2015):

- **Inicio:**

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

- **Ejecución:**

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el

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

- **Cierre:**

En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Se enmarca en el Escenario número 4 de la metodología antes definida el cual se modela mediante historias de usuario (HU), ya que es factible aplicarlo a proyectos bien definidos que ya hayan evaluado el negocio a informatizar, además el cliente siempre acompaña al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Debido a que el proyecto no es extenso, se muestra un alto vínculo entre el usuario y el desarrollador; y la aplicación a desarrollar está bien definida.

- **Análisis y diseño:**

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

- **Implementación:**

En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

- **Pruebas internas:**

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas (SÁNCHEZ, 2015).

1.10 Conclusiones parciales

En este capítulo se trataron los elementos teóricos que dan sustento a la propuesta de solución al problema planteado. Queda detallado el análisis de fuentes bibliográficas referentes al tema de configuración de permisos, usuarios y grupos lo cual permitió contextualizar los principales términos tratados en el capítulo y la investigación en general e identificar las funcionalidades que se convertirán luego en requisitos funcionales que debe poseer la propuesta de solución. La selección de las tecnologías, metodología y herramientas con soporte multiplataforma y basadas en software libre, posibilitó obtener una base tecnológica enfocada en las políticas de desarrollo definidas en la UCI.

Capítulo 2: Análisis y diseño de la propuesta de una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota

En el presente capítulo, se desarrolla el análisis y diseño de la propuesta de solución haciendo uso de los artefactos que propone la metodología Variación de *AUP* para la UCI, en el escenario 4. Contiene un modelo conceptual que permite comprender el contexto del negocio. Se definen los requisitos a través de la especificación de requisitos de software y su descripción mediante historias de usuario. Se describe la arquitectura y los patrones de diseño utilizados en el desarrollo de la propuesta de solución.

2.1 Propuesta de solución

La propuesta de solución de la presente investigación consiste en una aplicación de escritorio que puede instalarse en cualquier computadora conectada a la red que cuente con sistema operativo GNU/Linux Nova y desde ella configurar de forma remota y centralizada, los permisos, usuarios y grupos en todos los servidores de la institución. La aplicación permitirá un mejor control de los permisos otorgados a usuarios y grupos desde el rol de administrador, a la vez se perfecciona este control de acceso y da soluciones a dificultades planteadas en la situación problemática; y es el resultado de la solución de diseño dada al problema de investigación. En el epígrafe 2.2 se muestra el modelo conceptual realizado para comprender el contexto del negocio.

2.2 Modelo conceptual

Un modelo conceptual contiene la descripción de cómo se relacionan los conceptos que intervienen en el negocio, como principales características tiene (LARMAN, 2003):

- Mejora la comprensión de una persona del tema que se está modelando.
- Comunica detalles entre personas que necesitan conocerlos.
- Proporciona un punto de referencia para que personas como diseñadores elaboren planes específicos.
- Proporciona un documento al que se puede hacer referencia en el futuro y utilizar cuando las personas trabajan juntas.

Como relaciones entre conceptos se tiene:

Informalmente ⟶ idea, cosa u objeto.

Formalmente ⟶ puede considerarse en términos de:

Símbolos: Palabras o imágenes representando un concepto.

Definición: La descripción formal de un concepto.

Extensión: El conjunto de ejemplos a los cuales se aplica el concepto.

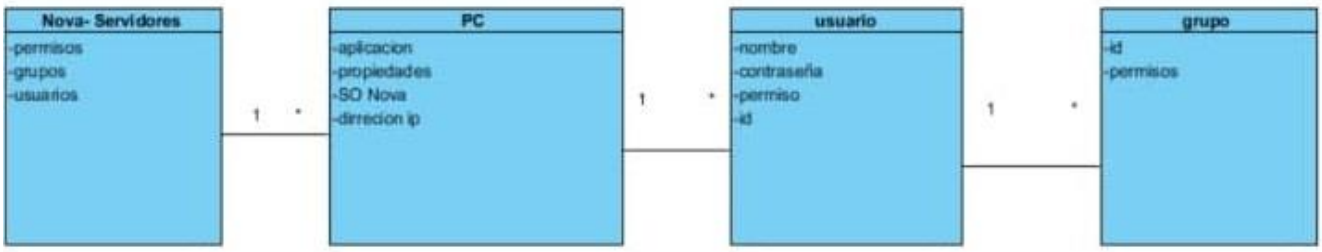

Figura 7: Modelo conceptual del negocio a informatizar.

(Fuente: Elaboración propia)

Usuario: utiliza la PC y todas sus funciones.

Nova-Servidores: distribución cubana de GNU/LINUX que almacena configuraciones del sistema de archivos realizadas por el cliente mediante la aplicación.

PC: es un ordenador que contiene la distribución cubana de GNU/Linux Nova y la aplicación a desarrollar.

Grupo: conjuntos de usuarios que utilizan la PC y todas sus funciones.

2.3 Requisitos de Software

2.3.1 Fuentes para la obtención de requisitos

- Modelo conceptual (Figura 7)
- Especialistas de CESOL.

- Estudio de Sistemas Homólogos.

2.3.2 Técnicas de identificación de requisitos

Construcción de prototipos

Es una manera de construir una versión inicial de la solución – un prototipo –. Se lo muestras al cliente, quien te da requerimientos adicionales. Los prototipos suelen consistir en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación. Fomentan el desarrollo de ideas y permiten a los usuarios mejorar las especificaciones de requerimientos. La técnica se aplica a través del prototipado de interfaz de usuario para conocer cuáles son las características que debían tener las interfaces de la herramienta a desarrollar. La aplicación de las técnicas descritas permite comprender con profundidad el problema de investigación y facilitó la identificación de las funcionalidades y características de la propuesta de solución, definidas en el siguiente epígrafe.

Entrevistas frente a frente

Es una técnica muy utilizada para obtener información cualitativa, como opiniones o descripciones subjetivas de actividades. Algunos aspectos importantes a tener en cuenta al realizar la entrevista es la preparación, ya que deben quedar bien definidos cuáles son los contenidos que se desean obtener. Además, se recomienda utilizar preguntas abiertas, donde los entrevistados puedan elaborar y dar detalles, más allá de simplemente responder “si” o “no”. La técnica se aplicó a través de un cuestionario de preguntas realizado al cliente y a 10 especialistas del centro CESOL (ver Anexo 2) para obtener la información necesaria referente a los requerimientos de la herramienta a desarrollar.

Observación

Por medio de esta técnica se obtiene información sobre la forma en que se efectúan las actividades. La técnica se aplica a través de una guía de observación, la cual permite observar la manera en que se llevan a cabo los procesos y verificar que realmente se sigan todos los pasos especificados. Los observadores experimentados saben qué buscar y cómo evaluar la relevancia de lo que observan.

Después de aplicada la entrevista y con la obtención de requisitos, se observa el proceso de visualización de los detalles técnicos llevado a cabo por los especialistas de CESOL (ver Anexo 3) lo que permite refinar e incorporar nuevos requisitos, presentes en la Tabla 6: Requisitos funcionales.

2.3.3 Requisitos de software

Los requisitos funcionales para un sistema describen lo que debe hacer el sistema. Estos requisitos dependen del tipo de software que se está desarrollando, los usuarios esperados del software y el enfoque general adoptado por la organización al redactar los requisitos. Cuando se expresan como requisitos del usuario, los requisitos funcionales deben estar escritos en lenguaje natural para que los usuarios y administradores del sistema puedan entenderlos. Los requisitos funcionales están escritos para desarrolladores de sistemas. Deben describir las funciones del sistema, sus entradas y salidas, y las excepciones en detalle (SOMMERVILLE, 2011). Los requisitos funcionales de la aplicación de escritorio a desarrollar, se muestran en la Tabla 6.

Tabla 6: Requisitos Funcionales.

Fuente: Elaboración propia.

No.	Nombre del Requisito Funcional	Descripción	Prioridad
1	Adicionar usuario	Permite añadir un usuario al sistema, y tiene en cuenta: nombre, identificador, contraseña.	Alta
2	Modificar usuario	Permite modificar un usuario creado previamente en el sistema y tiene en cuenta: nombre, identificador, contraseña, entre otros.	Alta
3	Eliminar usuario	Permite eliminar un usuario creado previamente en el sistema.	Alta
4	Cambiar contraseña de usuario	Permite cambiar la contraseña del usuario	Media
5	Mostrar listado de usuarios	Permite mostrar un listado de usuarios	Baja
6	Adicionar grupo	Permite añadir un grupo y tiene en	Alta

		cuenta: identificador, nombre, entre otros.	
7	Modificar grupo	Permite modificar los atributos de un grupo que ya ha sido creado en el sistema.	Alta
8	Eliminar Grupo	Permite eliminar grupo	Alta
9	Mostrar listado de grupo	Permite visualizar el listado de los grupos del sistema	Baja
10	Otorgar permiso de grupo	Permite otorgar permiso de grupo	Alta
11	Eliminar permiso de grupo	Permite eliminar permiso de grupo	Alta
12	Añadir usuario dentro de un grupo	Permite añadir usuario dentro del grupo del sistema.	Alta
13	Eliminar un usuario dentro de un grupo	Permite eliminar un usuario dentro del grupo del sistema.	Media
14	Asignar permiso de usuario	Permite asignar permiso de usuario creado dentro del sistema.	Alta
15	Eliminar permiso de usuario	Permite Eliminar permiso de usuario creado dentro del sistema.	Alta
16	Otorgar permiso de grupo	Permite otorgar permiso a grupos creados dentro del sistema.	Alta
17	Conectarse remotamente	Permite conectarse remotamente a otra máquina.	Alta

2.3.4 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Son propiedades o cualidades que el producto debe tener. A continuación, se describen los requisitos no funcionales del sistema (SOMMERVILLE, 2011).

La definición de los requisitos no funcionales de la propuesta de solución se realizó mediante el estándar de calidad ISO-25010 (Organización Internacional de Normalización, del inglés *International Organization for Standarization*), propuesto en el producto de trabajo “Especificación de requisitos de software” del expediente de proyecto utilizado en la actividad productiva de la universidad. A continuación, se presenta en la Tabla 7 el listado de los requisitos no funcionales de la propuesta de solución.

Tabla 7: Requisitos no funcionales.

Fuente: Elaboración propia.

No.	**Clasificación de Requisito**	**Descripción**
RnF1	Confiabilidad	La aplicación de configuración de permisos, usuarios y grupos debe ser capaz de recuperarse ante fallos de Software
RnF2	Interoperabilidad	La herramienta debe ser compatible con Nova Servidores.
RnF3	Usabilidad	En la aplicación se deben visualizar todos los mensajes en idioma español. La tipografía debe ser uniforme, de un tamaño adecuado.
RnF4	Usabilidad	La aplicación debe ofrecer una interfaz amigable, fácil de operar. Las interfaces deben poseer un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios la utilización de la misma.
RnF5	Seguridad	La aplicación debe poder proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o modificar los mismos, y las personas o sistemas autorizados tengan el acceso a ellos

Se utilizaron las historias de usuario para describir las características de los requisitos funcionales definidos anteriormente.

2.4 Historias de Usuario

Se utilizaron las historias de usuario para describir las características de los requisitos funcionales definidos anteriormente. Las Historias de Usuario especifican las tareas que debe realizar el sistema, lo que equivale a los casos de uso en el Proceso Unificado. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo (GONZÁLEZ, 2016). A continuación, en la Tabla 08, se muestra la Historia de Usuario Adicionar usuario, el resto se especifican en el Anexo 4.

Tabla 8: Historia de usuario Adicionar usuario

Fuente: Elaboración propia.

Historia de Usuario	
Número: 1	**Nombre del requisito:** Adicionar usuario
Programador: Anaely Trimiño	**Iteración:** 1
Prioridad: Alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 20 horas
Descripción: La funcionalidad comienza cuando el usuario 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:

Figura 8: Interfaz gráfica de Adicionar

(Fuente: Elaboración propia)

2.5 Arquitectura de software

Se utiliza Modelo/Vista/Controlador o MVC (*del inglés Model/View/Controller*) el mismo está dado por el hecho de que, el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas

Definición de las partes (FERNÁNDEZ, 2012).:

- El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

- La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa preferentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo.
- El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

El MVC en React solo se manifiesta explícitamente la vista en la carpeta views, pero se puede combinar fácilmente con otro framework en este caso .net para el desarrollo de la aplicación, los controladores se llamarían “Ver controlador" o "Vista del controlador" estar situados en la carpeta HomePageController, y los modelos directamente en la clase node.modules.

La figura 9 representa la separación de las clases de la aplicación de escritorio para configurar permisos usuarios grupos en Nova Servidores según la Liberia de React JS y el framerwork .Net, en el paquete controlador tenemos las clases controladoras user_controler.js, group_controler.js y permisos_controler.js en el paquete modelo se encuentras las entidades Usuario, grupos y permisos y en el paquete Vista tenemos listar_grupo, otorgar_permiso_grupo, lista_usuario, otorgar_permiso_usuario, app.test, app e index.

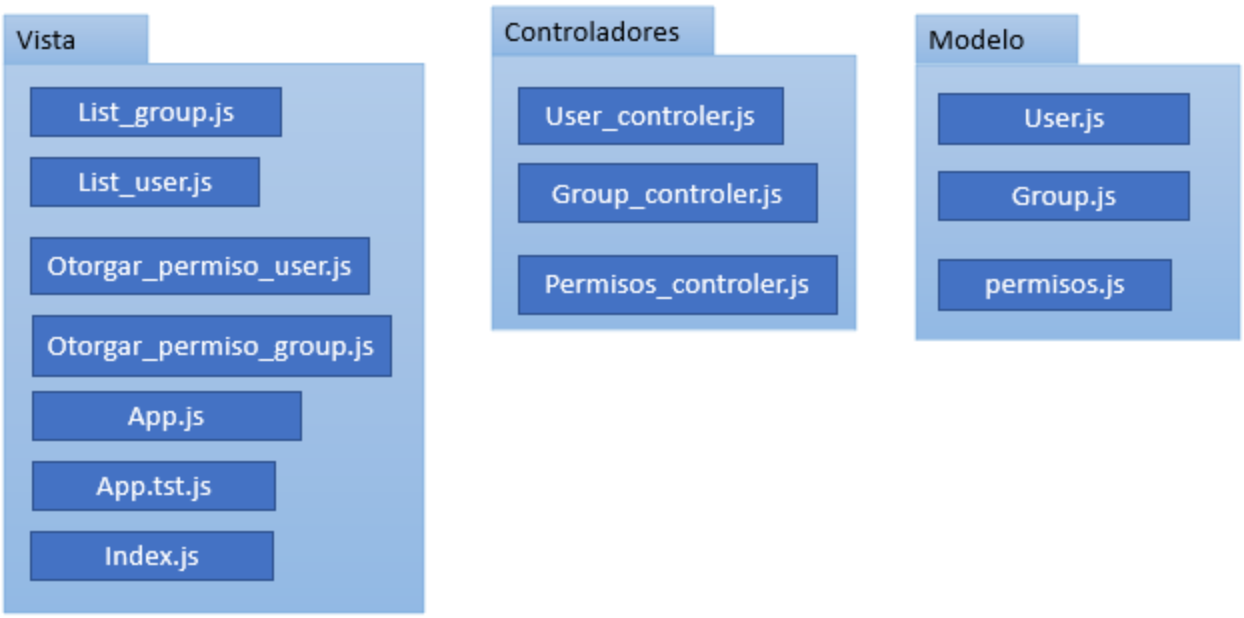

Figura 9: Uso del patrón arquitectónico en la aplicación

Fuente: (elaboración propia).

Diagrama de paquetes: muestra las agrupaciones lógicas en que está dividido el sistema, así como las dependencias entre dichas agrupaciones. A continuación, en la Figura 11 se muestra el diagrama de paquetes de la propuesta de solución (CAMPOS, 2017).

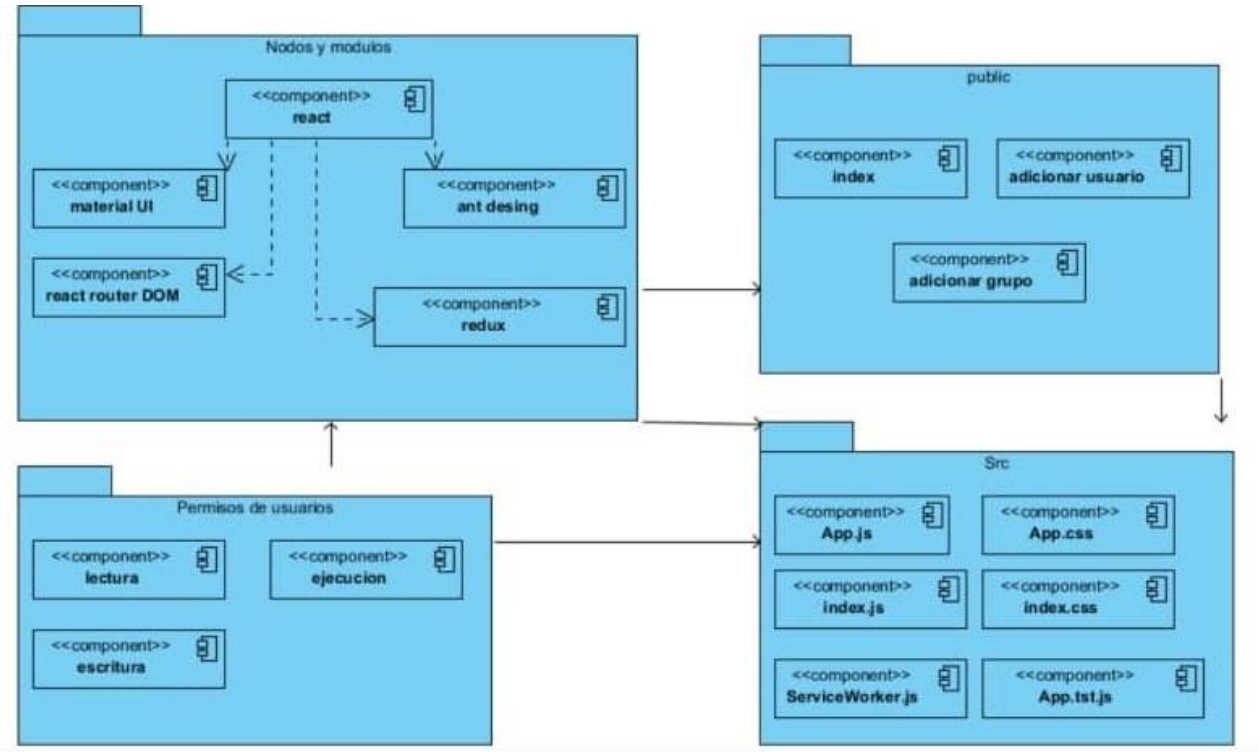

Figura 10: Diagrama de paquetes de la aplicación

Fuente: Elaboración Propia

2.6 Patrones de diseño

Los patrones de diseño son principios generales de soluciones que aplican ciertos estilos que ayudan a la creación de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (GONZÁLEZ, 2016). En el modelo del diseño de la aplicación se aplican los Patrones Generales de Software para Asignar Responsabilidades (GRASP, del inglés *General Responsibility Assignment Software Patterns*) y GoF (del inglés *Gang of Four*).

GRASP

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. A continuación, se describen los patrones utilizados.

Creador: guía la asignación de responsabilidades relacionadas con la creación de 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 se evidencia en las clases *select user.jsx* y list user.jsx, en ambas mediante el método created() el cual es responsable de la creación de las instancias de la clase config.jsx.

Experto: se utiliza para la asignación de responsabilidades relacionadas con 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 se encuentra presente en la clase *config.jsx* la cual maneja toda la información referente el contenido dentro de su archivo de configuración. *select react.jsx* y *list react.jsx* importan las etiquetas presentes en la clase *config.jsx* y de esta manera implementar los atributos de los archivos de configuración.

Bajo Acoplamiento: el objetivo de este patrón es lograr la menor dependencia entre clases, de manera que las modificaciones realizadas en algunas de ellas afecten lo menos posible el funcionamiento del sistema. Asigna responsabilidades de control de flujo del sistema a clases específicas Este patrón se evidencia en las *config.jsx* y *conf_crontroller.jsx,* entre ambas clases existe una sola relación, por lo que los cambios realizados en ellas tienen poca influencia en la otras.

Alta Cohesión: la cohesión es una medida del grado de focalización de las responsabilidades de una clase. Permite que las clases sean fáciles de entender, mantener y reutilizar. Este patrón se evidencia en la clase *select react.jsx*, la cual crea un atributo de los archivos de configuración y por tanto es responsable de las modificaciones sobre la configuración de la misma.

Patrones GoF: son de asignación de responsabilidades y se pueden categorizar en tres grupos teniendo en cuenta su propósito: creacionales, estructurales y de comportamiento.

Patrón Solitario (Singleton): es de tipo creacional con el objetivo de garantizar la existencia de una única instancia para una clase y posibilitar el 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.

2.7 Diagramas de clases de diseño

Representan las especificaciones de las clases e interfaces de software. Entre la información que representa se encuentran clases, interfaces con sus operaciones y constantes, métodos, navegabilidad y dependencias. Durante el diseño del sistema, el diagrama de clase del diseño se enfoca a los detalles de la implementación y refleja el funcionamiento de la aplicación en términos lógicos (MARTÍNEZ y HERNÁNDEZ 2013).

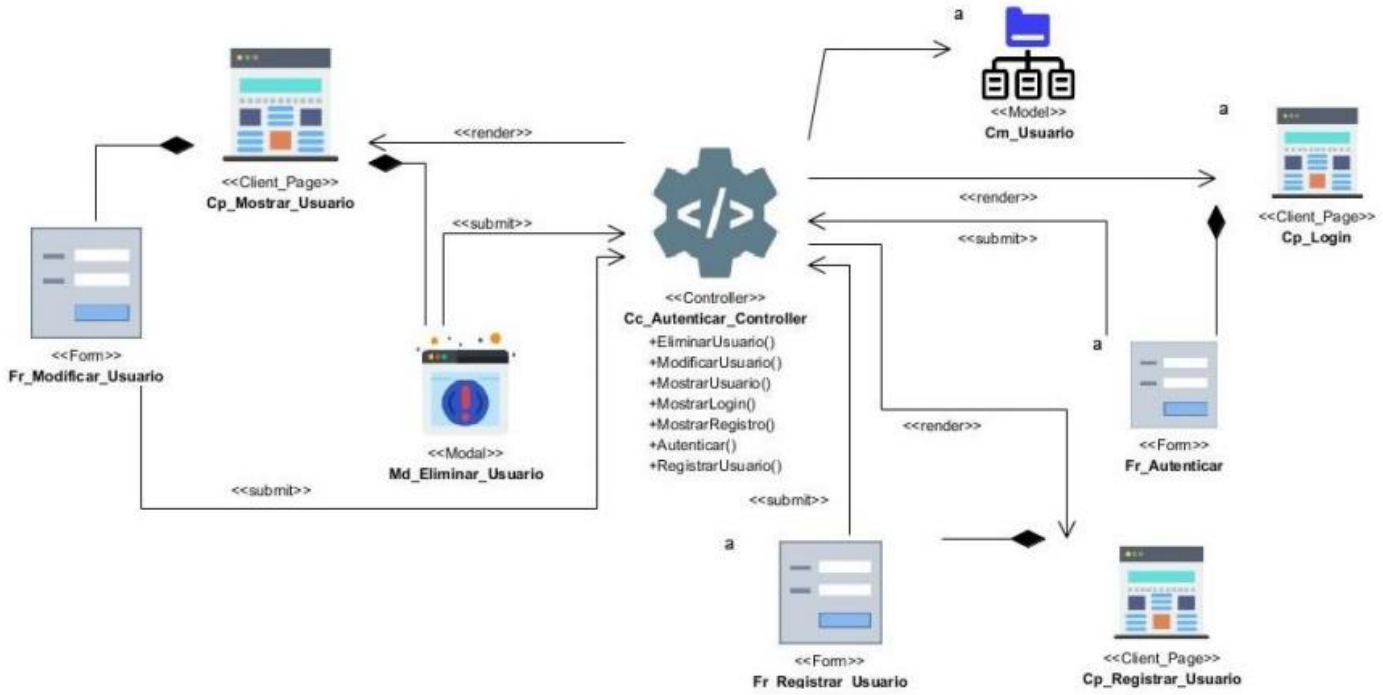

Figura 11: Diagrama de clases de diseño

Fuente: (Elaboración Propia)

2.8 Conclusiones parciales

A partir del estudio realizado en el capítulo se desarrolló el análisis y diseño de la propuesta de solución haciendo uso de los artefactos que propone la metodología Variación de AUP para la UCI, en el escenario 4; y los resultados obtenidos, arribando como conclusiones que la especificación de los requisitos funcionales y no funcionales los cuales definen las funciones ofrecidas por la aplicación, así como su correcto funcionamiento y guían la codificación de la aplicación. La definición de la arquitectura y los

patrones de diseño propuestos, permite establecer las bases para fomentar las buenas prácticas de programación durante la fase de implementación.

Capítulo 3: Implementación y pruebas de la propuesta de una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores de forma remota.

El presente capítulo contiene los productos de trabajos que se realizan en la disciplina de implementación y prueba, así como los estándares de codificación, diagrama de despliegue, diagrama de componentes y un ejemplo de la interfaz gráfica de usuario de la implementación de la propuesta de solución. Se documentan las pruebas realizadas y la evaluación del índice de satisfacción grupal sobre la aplicación de escritorio para la configuración de permisos, usuarios y grupos e Nova Servidores.

3.1 Implementación

Una implementación es la realización de una especificación técnica o algoritmos como un programa, componente de software, u otro sistema de cómputo. También son una manifestación en el mundo real de las funciones de procesamiento y estructuras de la información. Muchas implementaciones son realizadas según una especificación o un estándar (PRESSMAN, 2010).

3.1.1 Estándares de codificación

Los estándares de codificación son reglas de codificación que permiten tener una programación homogénea, comprendiendo todos los aspectos de la generalización del código, pues la aplicación debe de estar implementada como si un único programador escribiera el código de una sola vez. La usabilidad de estos permite conservar el código fuente entendible y fácil de mantener, además de mejorar la forma en la que se programa (RAMOS, 2017). Los estándares de codificación que se emplean en la aplicación son los requeridos para ReactJS definidos por una Guía de estilo de Standard Library. A continuación, se presentan de forma general las principales prácticas utilizadas en el desarrollo de la aplicación (SOLIS, 2015).

- 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.

  ```
  function camelCaseToTitle(camelCase) {
  ```

```
        if (!camelCase) {
            return '';
```

- Indentación: la unidad de indentación de bloques de sentencias son 4 espacios.

```
module.exports = {
  webpack: (config, { isServer }) => {
    if (!isServer) {
      config.target = 'electron-renderer';
      config.node = {
        __dirname: true,
      };
    }

    return config;
  },
};
```

- Comentarios: deben añadir claridad al código, contar el por qué y no el cómo y ser concisos.

```
<option disabled value="">Selecciona modo de red</option>
```

- Declaraciones: se declara cada variable en una línea distinta para permitir que cada una se comente por separado.

```
var a;
let x;
const y;
```

- Longitud de la línea: Limitar todas las líneas a un máximo de 120 caracteres.

```
module.exports = {

  darkMode: false, // or 'media' or 'class'

  theme: {

    extend: {},
```

3.1.2 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 (OMG Unified Modeling Language, 2017). La aplicación se ejecuta en sobre Nova Escritorio desde donde se conecta a Nova Servidores para configurar los permisos, usuarios y grupos en cada uno. El usuario accede mediante la aplicación que emplea conexiones seguras con la terminal, accede al servidor y realiza las operaciones de gestión de usuarios, grupos, y permisos del sistema. A continuación, se muestra la distribución física de la aplicación:

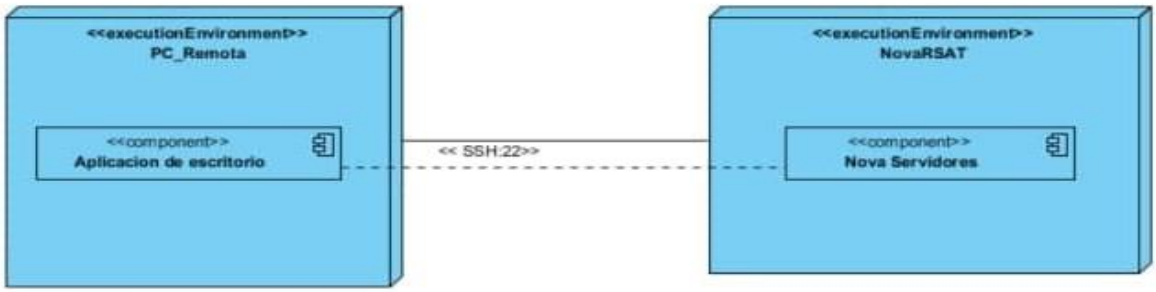

Figura 12: Diagrama de despliegue de la aplicación

Fuente: elaboración propia

3.2 Pruebas de software

Las pruebas de software consisten en la verificación del comportamiento de un programa en un conjunto finito de casos de prueba con diferentes ejecuciones. Consisten en una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación y calidad de un programa informático; probando el comportamiento del mismo (ZAPATA, 2013). La metodología variación de AUP

para la UCI define varios tipos como son pruebas internas, pruebas de liberación las cuales no se aplican en este caso pues son diseñadas y ejecutadas por una entidad certificadora de la calidad externa.

Dentro de las pruebas internas, a continuación, se describen los tipos de pruebas, métodos y técnicas que fueron realizadas a la propuesta de solución.

3.2.2 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas (RODRIGUEZ, 2015). Durante esta disciplina se aplicaron las pruebas unitarias y funcionales, las cuales se describen a continuación.

Pruebas Unitarias

Son ejecutadas por el equipo de desarrollo, consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controla suelen denominarse pruebas de módulos o pruebas de clases, siendo la convención definida por el lenguaje de programación la que influye en el término a utilizar (ZAPATA, 2013). Para poder emplear estas pruebas disponemos de herramientas que te permiten llevar a cabo está acción dependiendo del lenguaje de programación tiene sus propios programas para ejecutarlas en este caso PHPUnit en torno de pruebas unitarias en el lenguaje de programación PHP y Mocha y Chai nos permiten crear pruebas unitarias muy completas para el desarrollo en JavaScript.

Prueba Funcionales

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 9: Descripción de las variables para la prueba de validación.

Fuente: Elaboración propia

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 home**	Campo de selección	Si	Se habilita si el usuario no va a utilizar 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 10: Escenario 1.1 Adicionar usuario con los valores obligatorios

Fuente: Elaboración propia

V1	V2	V3	V4	V5	V6	V7	V8	V9
V	N/A	V	V	N/A	N/A	N/A	N/A	N/A
luis		**12345678**	**12345678**					
eva		**12345678**	**12345678**	**true**	**/home/luis**	**true**	**/bin/sh**	

Respuesta esperada: 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 11: Escenario 1.2 Adicionar usuario con los valores obligatorios vacíos.

Fuente: Elaboración propia

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 esperada: 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 12: Escenario 1.3 Adicionar usuario con los valores obligatorios incorrectos.

Fuente: Elaboración propia

V1	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**	**V8**	**V9**
V	N/A	V	V	N/A	N/A	N/A	N/A	N/A
Luis Garcia		Zaq12wsx,,	Zaq12wsx,,					
V	N/A	V	V	N/A	N/A	N/A	N/A	N/A
carlos		qazxcv123.	QAZXCV123					

Respuesta esperada: 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 13: Escenario 1.4 Adicionar usuario con los valores opcionales incorrectos.

Fuente: Elaboración propia

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

Respuesta esperada: 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.

3.4 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 siguiendo así un código homogéneo. La definición y puesta en práctica de pruebas unitarias y funcionales posibilitó la evaluación de la aplicación, así como la revisión de la correcta ejecución de las instrucciones y la validación del cumplimiento de los requisitos del sistema.

Conclusiones generales

La realización de la presente investigación permitió desarrollar una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores al dar respuesta a los planteamientos propuestos en la hipótesis descrita al inicio del trabajo. A partir de lo antes mencionado se concluye que:

- El estudio de la configuración de permisos, usuarios y grupos en los servidores GNU/Linux permitió comprender su funcionamiento y estructura. Así como a partir del análisis de las herramientas homólogas se logró identificar la necesidad de desarrollar una aplicación de escritorio para configurar permios, usuarios y grupos en Nova Servidores de forma remota.
- Mediante el diseño e implementación se logró una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores.
- Mediante a la aplicación de las diferentes pruebas de software queda evidenciado el correcto funcionamiento de la aplicación.

Recomendaciones

Una vez concluido el presente trabajo de diploma se recomienda:

- Utilizar el contenido de la investigación como base de referencia para el desarrollo de futuras aplicaciones para configurar usuarios, grupos y permisos en otras versiones de Nova y en otras distribuciones de GNU/Linux.
- Continuar el desarrollo de funcionales que sean requeridas en el futuro para asegurar la disponibilidad y usabilidad, así como la satisfacción de los usuarios potenciales.
- Aplicar las pruebas que se consideren faltantes en el capítulo 3.

Referencias Bibliográficas

AQUIA, Shadet. Módulo para el cifrado de los datos informáticos almacenados en NovaNAS. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2019.

ARIAS, Manuel. CISIAD. CISIAD. 2009. [En línea]. Disponible: [http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf. [Citado el: 27 agosto 2021.]

BALLESTER, Andrés. Desarrollo de una web app de carácter universitario orientada a la resolución de retos de programación, Trabajo Fin de Grado, Valencia, 2018.

BELTRÁN; Pedro. ¿Qué es una herramienta case? [En línea]. 2021. [Citado el: 27 junio 2021]. Disponible en: [https://www.academia.edu/28037284/Qu%C3%A9_es_una_herramienta_CASE].

BORRELL, Guillem. El control de versiones, 2006.

CAMPOS, Alejandro. Módulo de complementos para el servicio proxy de HMAST. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, junio, 2017.

CHACON, SCOTT y STRAUB, BEN. Pro Git. 2ª edición, Apress, 2014.

COOPER, Joe. The Book of Webmin, 2002.

DE OLIVEIRA, Cleusio. Sistema de control de acceso para entidades. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2020.

DOCUMENTACION cPanel. [En línea]. 2021. [Citado el: 1 de agosto de 2021.] Disponible en: [https://documentation.cpanel.net/display/ALD/Installation+FAQ].

EGUÍLIZ, Javier. Introducción a JavaScript. [En línea]. 2021. [Citado el: 17 junio 2021]. Disponible en: [https://www.jesusda.com/docs/ebooks/introduccion_javascript.pdf].

ELECTRON DOCUMENTATION. Documentación de Electron. [En línea]. 2021. [Citado el: 20 junio 2021]. Disponible en: [https://www.electronjs.org/docs].

FERNANDEZ, Yenisleidys y **DIAZ** Yanette Revista Telem@tica. Vol. 11. No. 1, enero-abril, 2012, p. 47-57.

GARCÍA, Eduardo. Programar es el futuro. ¿Qué es Vue.JS? [En línea]. 2021. [Citado el: 23 junio 2021] Disponible en: [https://codigofacilito.com/articulos/que-es-vue].

GARCÍA, Zuleimys. Módulo para la gestión de usuarios, grupos, procesos y repositorios del sistema GNU/Linux desde HMAST. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, junio, 2017.

GONZÁLEZ, Anabel. Entorno de Desarrollo Integrado para el lenguaje ensamblador. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2015.

GONZALEZ, Felipe y **VERA**, Yosel. MÓDULO DE HMAST PARA LA ADMINISTRACIÓN Y MIGRACIÓN HACIA SAMBA4 DEL SERVICIO DIRECTORIO ACTIVO. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, junio, 2016.

GONZÁLEZ, Laura Elena. Componente de cifrado de características biométricas utilizando plantillas cancelables. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2017.

GÓMEZ, Álvaro. Enciclopedia de la Seguridad Informática. Segunda Edición Actualizada. Editorial RA-MA, Madrid, 2011.

GOMEZ, Ramón Administración de servidores LINUX (Ubuntu/Fedora/Centos): Curso de formación para personal informático de la Universidad de Sevilla. 2014.

GONZALEZ, Felipe; **VERA** Yosel. Módulo de HMAST para la administración y migración hacia Samba4 del servicio directo activo. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, junio, 2016.

GONZALEZ, Jose Antonio; El lenguaje de programación C#.

HORSFORD, Rubén, **BAYARRE,** Héctor. Métodos y técnicas aplicados a la investigación en Atención de Salud, 2007.

JIMENEZ, Javi, Descubre React, 2015.

LARMAN, CRAIG. *UML y Patrones*. Madrid, España: Pearson Educación, S.A., 2003

LUCIDCHART Lucid Software Inc. ¿Qué es un modelo de base de datos? [En línea]. 2019. [Citado el: 22 junio 2021]. Disponible en: [https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-dedatos].

MARIN, Aymara; **TRUJILLO,** Yaimi; **BUEDO,** Denys. Marco de trabajo para gestionar actividades de calidad, Revista de Ciencias Informáticas, 2018, 12 (2), p. 74-88.

MARTÍNEZ, M.B.B. y **HERNÁNDEZ**, M.M.R., 2013. DIAGRAMA DE CLASE

MIYARES, Yoel. Administración del servicio antivirus desde la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST). Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, junio, 2014.

OFICINA NACIONAL DE NORMALIZACIÓN. Norma Cubana NC ISO/IEC 25023:2017. Ingeniería de Software y Sistemas – Requisitos de la Calidad y Evaluación de Software y Sistemas (SQUARE) – Medición de la Calidad del Producto de Software y Sistema. 2017. pp. 39.

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

PANEQUE, Isaac. ¡Libro Linux 4you! Edición 2013.

PALMA, Nurisel. Módulo para la administración de los servidores web en HMAST. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2013.

PALMA, Nurisel. Herramienta informática para la selección de Apache 2 y Nginx durante la migración a código abierto. Tesis para optar por el título de Máster en Informática Avanzada, Universidad de las Ciencias Informáticas, La Habana, 2019.

PÉREZ, Yoandy; **GARCÍA**, Abel; **GOÑI**, Ángel. Buenas Prácticas para la Migración a Código Abierto. La Habana, Ediciones Fututo, 2015.

PLESK International GmbH. Plesk, the power to simplify. [En línea]. [Citado el: 2 de agosto de 2015.] https://www.plesk.com/about-us.

PONS, Nicolas. Linux. Principios básicos de uso del sistema. 5ª edición. Barcelona, Ediciones ENI, 2016, p. 155-162.

RAFFINO, María Estela "Grupo". Argentina. [En línea]. 2020. [Citado: 03 de julio de 2021]. Disponible en: [https://concepto.de/grupo/].

RAULT, Raphaël, et al. Seguridad informática- Hacking Ético: Conocer el ataque para una mejor defensa (3ª edición). Barcelona Ediciones ENI, 2015, p. 402-403.

RAMOS, Kariné, et al. Experiencias del programa de mejora de procesos en la Universidad de las Ciencias Informáticas. Revista Cubana de Ciencias Informáticas, 2011, 5 (2): p. 1-16.

RAMOS, D., **NORIEGA**, R., **LAÍNEZ**, J.R. y **DURANGO,** A., 2017. *Curso de Ingeniería de Software: 2ª Edición*.

RAPPIN, Noel, Modern CSS withTailwind, 2021.

RIVERO, Maikel, AngularJs Paso a Paso, 2ª edición, 2016

RODRIGUEZ, Alejandro; **VIERA** Orlando. Módulo para la ejecución automática de scripts en los clientes desde el servidor de GRHS. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2015.

RODRIGUEZ, Miosotis Toledo. Módulo para la gestión de copias de seguridad en Nova 360. La Habana, 2019.

RODRIGUEZ, Tamara. Metodología de desarrollo para la actividad productiva de la UCI. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana. La Habana, 2015

SANCHEZ, Tamara. Metodología de desarrollo para la Actividad productiva de la UCI. La Habana, 2015.

SOMMERVILLE, 2011. *Software engineering* (9th Edition).

TORRES, Francisco; **PIZARRO**, Ana María. Linux para usuarios. Madrid, Ministerio de Educación, Cultura y Deporte, 2016.

SG BUZZ. Obtención de Requerimientos. Técnicas y Estrategia. [En línea]. [Citado el: 8 octubre 2021]. Disponible en: [https://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia].

SOLIS, Carlos. Manual del Guerrero: AngularJS,

TINAJERO, Marlon, **CATOTA,** Vicente, **CATOTA** Edgar La Tecnica de iadov. Niveles de satisfacción del cliente, 2019.

UBUNTU es.¿Qué es WebMin? [En línea] 6 de marzo de 2010. [Citado el: 4 de septiembre de 2021]. Disponible en: [http://www.ubuntu-es.org/node/129411].

VISUALPARADIGM. [En línea] [Citado el: 29 junio 2021]. Disponible en: [https://www.visual-paradigm.com-aboutus/newsreleases/].

VISUALStudioCod. [En línea] enero de 2020. [Citado el: 17 de febrero de 2020]. Disponible en: [https://code.visualstudio.com/updates/v1_42].

VUETIFY. Why vuetify. [En línea]. [Citado el: 20 de junio de 2021]. Disponible en: [https://vuetifyjs.com/en/introduction/why-vuetify/].

YAST/Introducción - OpenSUSE. [En línea] [Citado el: 4 de septiembre de 2021.] Disponible en: [https://es.opensuse.org/Portal:YaST/Introducci%C3%B3n].

ZENTYAL Linux Small Business Server. [En línea] 2017. [Citado el: 30 de septiembre de 2021]. Disponible en: [https://www.zentyal.com.es].

ZORRILLA, Luis Ernesto. Portal web del Observatorio Tecnológico de la Universidad de las Ciencias Informáticas. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2016.

Anexos

Anexo 1: Entrevista realizada a los especialistas de CESOL acerca del proceso de configuración de permisos, usuarios y grupos en las instituciones cubanas.

Estimado(a) especialista.

La presente entrevista tiene como objetivo conocer cómo se lleva a cabo actualmente el proceso de configuración de permisos, usuarios y grupos en las instituciones cubanas, una vez migrados los servicios telemáticos y por ende instalado Nova Servidores. Responda dentro de sus posibilidades las siguientes preguntas:

Pregunta 1: ¿Cómo se realiza la configuración de permisos, usuarios y grupos?

Pregunta 2: ¿Qué consecuencias tiene la forma en la que se realiza dicho proceso hoy?

Pregunta 3: ¿Tienen control sobre la configuración de permisos, usuarios y grupos en las instituciones cubanas?

a) De ser negativa la respuesta explique por qué.
b) De ser positiva explique mediante qué herramienta y si esta es satisfactoria.

Pregunta 4: ¿Cuántos usuarios cuentan actualmente con permisos de acceso?

Pregunta 5: ¿Cuántos grupos cuentan con permisos de acceso?

Pregunta 6: ¿Son suficientes los medios con los que cuenta para desarrollar una correcta configuración de permisos, usuarios y grupos?

Anexo 2: Entrevista realizada a especialistas del CESOL con el fin de conocer el funcionamiento actual de la configuración de permisos, usuarios y grupos en Nova servidores.

Objetivo: Conocer el proceso actual de configuración de permisos, usuarios y grupos en Nova Servidores.

Preguntas:

1. ¿Cuáles son los aspectos fundamentales a tener en cuenta cuando se configuran permisos a usuarios y grupos en Nova Servidores actualmente?

2. ¿Cuáles son los pasos a seguir para toda la administración de dichos permisos?

3. ¿Qué resultado se obtienen cuando el usuario o grupo accede a los permisos otorgados a este?

4. ¿Algunas restricciones deben cumplirse para poder acceder a dichos permisos? Si la respuesta es sí, ¿cuáles serían?

Anexo 3: Proceso para identificar los requisitos a utilizar para el desarrollo de una aplicación de escritorio para la configuración de permisos, usuarios y grupos en Nova Servidores.

Observador: Anaely Trimiño Haramboure

Lugar: Departamento de Servicios Integrales en Migración Asesoría y Soporte (SIMAYS) que pertenece al centro CESOL.

Objetivo: Identificar los requisitos fundamentales para la realización de la configuración de permisos, usuarios y grupos en Nova Servidores.

Preguntas:

1. ¿Cómo se realiza el proceso de configuración de permisos, usuarios y grupos en Nova Servidores?

2. ¿Cuáles son los pasos a seguir?

3. ¿Cuál es el resultado?

4 ¿Cuáles son los comandos para hacer este proceso?

5. ¿Usted se sabe de memoria todos los comandos?

6. ¿Es sencillo utilizar la consola o terminal?

Anexo 4: Historias de usuario

Tabla 14 Historia 17 Conectar remotamente

Fuente: (Elaboración propia)

Historia de Usuario	
Número: 17	**Nombre del requisito:** Conectar remotamente
Programador: Anaely Trimiño	**Iteración:** 1
Prioridad: Alta	**Tiempo Estimado:**1 horas
Riesgo en Desarrollo: No aplica	**Tiempo Real:** 1horas
Descripción: La aplicación permite al usuario conectarse remotamente desde una computadora a otra. El usuario selecciona la opción de "conectar" que aparece en la pantalla de inicio de la herramienta y automáticamente se muestra una interfaz para introducir el IP, usuario y contraseña de la maquina ala que vas acceder.	
Observaciones: Tener instalado el protocolo SSH.	

Tabla 15 Historia de usuario 7 Modificar grupo

Fuente: (Elaboración propia)

Historia de Usuario	
Número: 7	**Nombre del requisito:** Modificar grupo
Programador: Anaely Trimiño	**Iteración:** 1

Prioridad: Alta	**Tiempo Estimado:**10 horas
Riesgo en Desarrollo: Disponibilidad de la red	**Tiempo Real:** 10 horas
Descripción: La funcionalidad comienza cuando usuario de 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 modificar grupo:	

Figura 13: Interfaz gráfica de Modificar grupo

(Fuente: Elaboración propia)